

This section of the Kubernetes documentation contains references.

API Reference

- [Glossary](#) - a comprehensive, standardized list of Kubernetes terminology
- [Kubernetes API Reference](#)
- [One-page API Reference for Kubernetes v1.28](#)
- [Using The Kubernetes API](#) - overview of the API for Kubernetes.
- [API access control](#) - details on how Kubernetes controls API access
- [Well-Known Labels, Annotations and Taints](#)

Officially supported client libraries

To call the Kubernetes API from a programming language, you can use [client libraries](#). Officially supported client libraries:

- [Kubernetes Go client library](#)
- [Kubernetes Python client library](#)
- [Kubernetes Java client library](#)
- [Kubernetes JavaScript client library](#)
- [Kubernetes C# client library](#)
- [Kubernetes Haskell client library](#)

CLI

- [kubectl](#) - Main CLI tool for running commands and managing Kubernetes clusters.
 - [JSONPath](#) - Syntax guide for using [JSONPath expressions](#) with kubectl.
- [kubeadm](#) - CLI tool to easily provision a secure Kubernetes cluster.

Components

- [kubelet](#) - The primary agent that runs on each node. The kubelet takes a set of PodSpecs and ensures that the described containers are running and healthy.
- [kube-apiserver](#) - REST API that validates and configures data for API objects such as pods, services, replication controllers.
- [kube-controller-manager](#) - Daemon that embeds the core control loops shipped with Kubernetes.
- [kube-proxy](#) - Can do simple TCP/UDP stream forwarding or round-robin TCP/UDP forwarding across a set of back-ends.
- [kube-scheduler](#) - Scheduler that manages availability, performance, and capacity.
 - [Scheduler Policies](#)

- [Scheduler Profiles](#)
- List of [ports and protocols](#) that should be open on control plane and worker nodes

Config APIs

This section hosts the documentation for "unpublished" APIs which are used to configure kubernetes components or tools. Most of these APIs are not exposed by the API server in a RESTful way though they are essential for a user or an operator to use or manage a cluster.

- [kubeconfig \(v1\)](#)
- [kube-apiserver admission \(v1\)](#)
- [kube-apiserver configuration \(v1alpha1\)](#) and [kube-apiserver configuration \(v1beta1\)](#) and [kube-apiserver configuration \(v1\)](#)
- [kube-apiserver encryption \(v1\)](#)
- [kube-apiserver event rate limit \(v1alpha1\)](#)
- [kubelet configuration \(v1alpha1\)](#) and [kubelet configuration \(v1beta1\)](#) [kubelet configuration \(v1\)](#)
- [kubelet credential providers \(v1alpha1\)](#), [kubelet credential providers \(v1beta1\)](#) and [kubelet credential providers \(v1\)](#) [kube-scheduler configuration \(v1beta3\)](#) and [kube-scheduler configuration \(v1\)](#)
- [kube-controller-manager configuration \(v1alpha1\)](#)
- [kube-proxy configuration \(v1alpha1\)](#)
- [audit.k8s.io/v1 API](#)
- [Client authentication API \(v1beta1\)](#) and [Client authentication API \(v1\)](#)
- [WebhookAdmission configuration \(v1\)](#)
- [ImagePolicy API \(v1alpha1\)](#)

Config API for kubeadm

- [v1beta3](#)
- [v1beta4](#)

External APIs

These are the APIs defined by the Kubernetes project, but are not implemented by the core project:

- [Metrics API \(v1beta1\)](#)
- [Custom Metrics API \(v1beta2\)](#)
- [External Metrics API \(v1beta1\)](#)

Design Docs

An archive of the design docs for Kubernetes functionality. Good starting points are [Kubernetes Architecture](#) and [Kubernetes Design Overview](#).

API Overview

This section provides reference information for the Kubernetes API.

The REST API is the fundamental fabric of Kubernetes. All operations and communications between components, and external user commands are REST API calls that the API Server handles. Consequently, everything in the Kubernetes platform is treated as an API object and has a corresponding entry in the [API](#).

The [Kubernetes API reference](#) lists the API for Kubernetes version v1.28.

For general background information, read [The Kubernetes API](#). [Controlling Access to the Kubernetes API](#) describes how clients can authenticate to the Kubernetes API server, and how their requests are authorized.

API versioning

The JSON and Protobuf serialization schemas follow the same guidelines for schema changes. The following descriptions cover both formats.

The API versioning and software versioning are indirectly related. The [API and release versioning proposal](#) describes the relationship between API versioning and software versioning.

Different API versions indicate different levels of stability and support. You can find more information about the criteria for each level in the [API Changes documentation](#).

Here's a summary of each level:

- Alpha:
 - The version names contain alpha (for example, v1alpha1).
 - Built-in alpha API versions are disabled by default and must be explicitly enabled in the kube-apiserver configuration to be used.
 - The software may contain bugs. Enabling a feature may expose bugs.
 - Support for an alpha API may be dropped at any time without notice.
 - The API may change in incompatible ways in a later software release without notice.
 - The software is recommended for use only in short-lived testing clusters, due to increased risk of bugs and lack of long-term support.
- Beta:
 - The version names contain beta (for example, v2beta3).
 - Built-in beta API versions are disabled by default and must be explicitly enabled in the kube-apiserver configuration to be used (**except** for beta versions of APIs introduced prior to Kubernetes 1.22, which were enabled by default).
 - Built-in beta API versions have a maximum lifetime of 9 months or 3 minor releases (whichever is longer) from introduction to deprecation, and 9 months or 3 minor releases (whichever is longer) from deprecation to removal.
 - The software is well tested. Enabling a feature is considered safe.

- The support for a feature will not be dropped, though the details may change.
- The schema and/or semantics of objects may change in incompatible ways in a subsequent beta or stable API version. When this happens, migration instructions are provided. Adapting to a subsequent beta or stable API version may require editing or re-creating API objects, and may not be straightforward. The migration may require downtime for applications that rely on the feature.
- The software is not recommended for production uses. Subsequent releases may introduce incompatible changes. Use of beta API versions is required to transition to subsequent beta or stable API versions once the beta API version is deprecated and no longer served.

Note: Please try beta features and provide feedback. After the features exit beta, it may not be practical to make more changes.

- Stable:

- The version name is vX where X is an integer.
- Stable API versions remain available for all future releases within a Kubernetes major version, and there are no current plans for a major version revision of Kubernetes that removes stable APIs.

API groups

[API groups](#) make it easier to extend the Kubernetes API. The API group is specified in a REST path and in the apiVersion field of a serialized object.

There are several API groups in Kubernetes:

- The *core* (also called *legacy*) group is found at REST path /api/v1. The core group is not specified as part of the apiVersion field, for example, apiVersion: v1.
- The named groups are at REST path /apis/\$GROUP_NAME/\$VERSION and use apiVersion: \$GROUP_NAME/\$VERSION (for example, apiVersion: batch/v1). You can find the full list of supported API groups in [Kubernetes API reference](#).

Enabling or disabling API groups

Certain resources and API groups are enabled by default. You can enable or disable them by setting --runtime-config on the API server. The --runtime-config flag accepts comma separated <key>[=<value>] pairs describing the runtime configuration of the API server. If the =<value> part is omitted, it is treated as if =true is specified. For example:

- to disable batch/v1, set --runtime-config=batch/v1=false
- to enable batch/v2alpha1, set --runtime-config=batch/v2alpha1
- to enable a specific version of an API, such as storage.k8s.io/v1beta1/csistoragecapacities, set --runtime-config=storage.k8s.io/v1beta1/csistoragecapacities

Note: When you enable or disable groups or resources, you need to restart the API server and controller manager to pick up the --runtime-config changes.

Persistence

Kubernetes stores its serialized state in terms of the API resources by writing them into [etcd](#).

What's next

- Learn more about [API conventions](#)
- Read the design documentation for [aggregator](#)

Kubernetes API Concepts

The Kubernetes API is a resource-based (RESTful) programmatic interface provided via HTTP. It supports retrieving, creating, updating, and deleting primary resources via the standard HTTP verbs (POST, PUT, PATCH, DELETE, GET).

For some resources, the API includes additional subresources that allow fine grained authorization (such as separate views for Pod details and log retrievals), and can accept and serve those resources in different representations for convenience or efficiency.

Kubernetes supports efficient change notifications on resources via *watches*. Kubernetes also provides consistent list operations so that API clients can effectively cache, track, and synchronize the state of resources.

You can view the [API reference](#) online, or read on to learn about the API in general.

Kubernetes API terminology

Kubernetes generally leverages common RESTful terminology to describe the API concepts:

- A *resource type* is the name used in the URL (pods, namespaces, services)
- All resource types have a concrete representation (their object schema) which is called a *kind*
- A list of instances of a resource type is known as a *collection*
- A single instance of a resource type is called a *resource*, and also usually represents an *object*
- For some resource types, the API includes one or more *sub-resources*, which are represented as URI paths below the resource

Most Kubernetes API resource types are [objects](#) – they represent a concrete instance of a concept on the cluster, like a pod or namespace. A smaller number of API resource types are *virtual* in that they often represent operations on objects, rather than objects, such as a permission check (use a POST with a JSON-encoded body of SubjectAccessReview to the subjectaccessreviews resource), or the eviction sub-resource of a Pod (used to trigger [API-initiated eviction](#)).

Object names

All objects you can create via the API have a unique object [name](#) to allow idempotent creation and retrieval, except that virtual resource types may not have unique names if they are not retrievable, or do not rely on idempotency. Within a [namespace](#), only one object of a given kind

can have a given name at a time. However, if you delete the object, you can make a new object with the same name. Some objects are not namespaced (for example: Nodes), and so their names must be unique across the whole cluster.

API verbs

Almost all object resource types support the standard HTTP verbs - GET, POST, PUT, PATCH, and DELETE. Kubernetes also uses its own verbs, which are often written lowercase to distinguish them from HTTP verbs.

Kubernetes uses the term **list** to describe returning a [collection](#) of resources to distinguish from retrieving a single resource which is usually called a **get**. If you sent an HTTP GET request with the ?watch query parameter, Kubernetes calls this a **watch** and not a **get** (see [Efficient detection of changes](#) for more details).

For PUT requests, Kubernetes internally classifies these as either **create** or **update** based on the state of the existing object. An **update** is different from a **patch**; the HTTP verb for a **patch** is PATCH.

Resource URIs

All resource types are either scoped by the cluster (/apis/GROUP/VERSION/*) or to a namespace (/apis/GROUP/VERSION/namespaces/NAMESPACE/*). A namespace-scoped resource type will be deleted when its namespace is deleted and access to that resource type is controlled by authorization checks on the namespace scope.

Note: core resources use /api instead of /apis and omit the GROUP path segment.

Examples:

- /api/v1/namespaces
- /api/v1/pods
- /api/v1/namespaces/my-namespace/pods
- /apis/apps/v1/deployments
- /apis/apps/v1/namespaces/my-namespace/deployments
- /apis/apps/v1/namespaces/my-namespace/deployments/my-deployment

You can also access collections of resources (for example: listing all Nodes). The following paths are used to retrieve collections and resources:

- Cluster-scoped resources:
 - GET /apis/GROUP/VERSION/RESOURCETYPE - return the collection of resources of the resource type
 - GET /apis/GROUP/VERSION/RESOURCETYPE/NAME - return the resource with NAME under the resource type
- Namespace-scoped resources:
 - GET /apis/GROUP/VERSION/RESOURCETYPE - return the collection of all instances of the resource type across all namespaces
 - GET /apis/GROUP/VERSION/namespaces/NAMESPACE/RESOURCETYPE - return collection of all instances of the resource type in NAMESPACE

- GET /apis/GROUP/VERSION/namespaces/NAMESPACE/RESOURCETYPE/NAME - return the instance of the resource type with NAME in NAMESPACE

Since a namespace is a cluster-scoped resource type, you can retrieve the list (“collection”) of all namespaces with GET /api/v1/namespaces and details about a particular namespace with GET /api/v1/namespaces/NAME.

- Cluster-scoped subresource: GET /apis/GROUP/VERSION/RESOURCETYPE/NAME/SUBRESOURCE
- Namespace-scoped subresource: GET /apis/GROUP/VERSION/namespaces/NAMESPACE/RESOURCETYPE/NAME/SUBRESOURCE

The verbs supported for each subresource will differ depending on the object - see the [API reference](#) for more information. It is not possible to access sub-resources across multiple resources - generally a new virtual resource type would be used if that becomes necessary.

Efficient detection of changes

The Kubernetes API allows clients to make an initial request for an object or a collection, and then to track changes since that initial request: a **watch**. Clients can send a **list** or a **get** and then make a follow-up **watch** request.

To make this change tracking possible, every Kubernetes object has a `resourceVersion` field representing the version of that resource as stored in the underlying persistence layer. When retrieving a collection of resources (either namespace or cluster scoped), the response from the API server contains a `resourceVersion` value. The client can use that `resourceVersion` to initiate a **watch** against the API server.

When you send a **watch** request, the API server responds with a stream of changes. These changes itemize the outcome of operations (such as **create**, **delete**, and **update**) that occurred after the `resourceVersion` you specified as a parameter to the **watch** request. The overall **watch** mechanism allows a client to fetch the current state and then subscribe to subsequent changes, without missing any events.

If a client **watch** is disconnected then that client can start a new **watch** from the last returned `resourceVersion`; the client could also perform a fresh **get** / **list** request and begin again. See [Resource Version Semantics](#) for more detail.

For example:

1. List all of the pods in a given namespace.

```
GET /api/v1/namespaces/test/pods
---
200 OK
Content-Type: application/json

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {"resourceVersion": "10245"},
  "items": [...]
}
```

Starting from resource version 10245, receive notifications of any API operations (such as 2. **create**, **delete**, **patch** or **update**) that affect Pods in the *test* namespace. Each change notification is a JSON document. The HTTP response body (served as application/json) consists a series of JSON documents.

```
GET /api/v1/namespaces/test/pods?watch=1&resourceVersion=10245
---
200 OK
Transfer-Encoding: chunked
Content-Type: application/json

{
  "type": "ADDED",
  "object": {"kind": "Pod", "apiVersion": "v1", "metadata": {"resourceVersion": "10596", ...}, ...}
}
{
  "type": "MODIFIED",
  "object": {"kind": "Pod", "apiVersion": "v1", "metadata": {"resourceVersion": "11020", ...}, ...}
}
...
...
```

A given Kubernetes server will only preserve a historical record of changes for a limited time. Clusters using etcd 3 preserve changes in the last 5 minutes by default. When the requested **watch** operations fail because the historical version of that resource is not available, clients must handle the case by recognizing the status code 410 Gone, clearing their local cache, performing a new **get** or **list** operation, and starting the **watch** from the resourceVersion that was returned.

For subscribing to collections, Kubernetes client libraries typically offer some form of standard tool for this **list**-then-**watch** logic. (In the Go client library, this is called a Reflector and is located in the k8s.io/client-go/tools/cache package.)

Watch bookmarks

To mitigate the impact of short history window, the Kubernetes API provides a watch event named BOOKMARK. It is a special kind of event to mark that all changes up to a given resourceVersion the client is requesting have already been sent. The document representing the BOOKMARK event is of the type requested by the request, but only includes a .metadata.resourceVersion field. For example:

```
GET /api/v1/namespaces/test/pods?
watch=1&resourceVersion=10245&allowWatchBookmarks=true
---
200 OK
Transfer-Encoding: chunked
Content-Type: application/json

{
  "type": "ADDED",
  "object": {"kind": "Pod", "apiVersion": "v1", "metadata": {"resourceVersion": "10596", ...}, ...}
}
...
{
```

```
"type": "BOOKMARK",
"object": {"kind": "Pod", "apiVersion": "v1", "metadata": {"resourceVersion": "12746"} }
}
```

As a client, you can request BOOKMARK events by setting the allowWatchBookmarks=true query parameter to a **watch** request, but you shouldn't assume bookmarks are returned at any specific interval, nor can clients assume that the API server will send any BOOKMARK event even when requested.

Streaming lists

FEATURE STATE: Kubernetes v1.27 [alpha]

On large clusters, retrieving the collection of some resource types may result in a significant increase of resource usage (primarily RAM) on the control plane. In order to alleviate its impact and simplify the user experience of the **list + watch** pattern, Kubernetes v1.27 introduces as an alpha feature the support for requesting the initial state (previously requested via the **list** request) as part of the **watch** request.

Provided that the WatchList [feature gate](#) is enabled, this can be achieved by specifying sendInitialEvents=true as query string parameter in a **watch** request. If set, the API server starts the watch stream with synthetic init events (of type ADDED) to build the whole state of all existing objects followed by a [BOOKMARK event](#) (if requested via allowWatchBookmarks=true option). The bookmark event includes the resource version to which is synced. After sending the bookmark event, the API server continues as for any other **watch** request.

When you set sendInitialEvents=true in the query string, Kubernetes also requires that you set resourceVersionMatch to NotOlderThan value. If you provided resourceVersion in the query string without providing a value or don't provide it at all, this is interpreted as a request for *consistent read*; the bookmark event is sent when the state is synced at least to the moment of a consistent read from when the request started to be processed. If you specify resourceVersion (in the query string), the bookmark event is sent when the state is synced at least to the provided resource version.

Example

An example: you want to watch a collection of Pods. For that collection, the current resource version is 10245 and there are two pods: foo and bar. Then sending the following request (explicitly requesting *consistent read* by setting empty resource version using resourceVersion=) could result in the following sequence of events:

```
GET /api/v1/namespaces/test/pods?
watch=1&sendInitialEvents=true&allowWatchBookmarks=true&resourceVersion=&resourceVersionMatch=NotOlderThan
---
200 OK
Transfer-Encoding: chunked
Content-Type: application/json

{
  "type": "ADDED",
  "object": {"kind": "Pod", "apiVersion": "v1", "metadata": {"resourceVersion": "8467", "name": "
```

```
"foo"}, ...}
}
{
  "type": "ADDED",
  "object": {"kind": "Pod", "apiVersion": "v1", "metadata": {"resourceVersion": "5726", "name": "bar"}, ...}
}
{
  "type": "BOOKMARK",
  "object": {"kind": "Pod", "apiVersion": "v1", "metadata": {"resourceVersion": "10245"} }
}
...
<followed by regular watch stream starting from resourceVersion="10245">
```

Response compression

FEATURE STATE: Kubernetes v1.16 [beta]

APIResponseCompression is an option that allows the API server to compress the responses for **get** and **list** requests, reducing the network bandwidth and improving the performance of large-scale clusters. It is enabled by default since Kubernetes 1.16 and it can be disabled by including APIResponseCompression=false in the --feature-gates flag on the API server.

API response compression can significantly reduce the size of the response, especially for large resources or [collections](#). For example, a **list** request for pods can return hundreds of kilobytes or even megabytes of data, depending on the number of pods and their attributes. By compressing the response, the network bandwidth can be saved and the latency can be reduced.

To verify if APIResponseCompression is working, you can send a **get** or **list** request to the API server with an Accept-Encoding header, and check the response size and headers. For example:

```
GET /api/v1/pods
Accept-Encoding: gzip
---
200 OK
Content-Type: application/json
content-encoding: gzip
...
```

The content-encoding header indicates that the response is compressed with gzip.

Retrieving large results sets in chunks

FEATURE STATE: Kubernetes v1.9 [beta]

On large clusters, retrieving the collection of some resource types may result in very large responses that can impact the server and client. For instance, a cluster may have tens of thousands of Pods, each of which is equivalent to roughly 2 KiB of encoded JSON. Retrieving all pods across all namespaces may result in a very large response (10-20MB) and consume a large amount of server resources.

Provided that you don't explicitly disable the APIListChunking [feature gate](#), the Kubernetes API server supports the ability to break a single large collection request into many smaller chunks while preserving the consistency of the total request. Each chunk can be returned sequentially which reduces both the total size of the request and allows user-oriented clients to display results incrementally to improve responsiveness.

You can request that the API server handles a **list** by serving single collection using pages (which Kubernetes calls *chunks*). To retrieve a single collection in chunks, two query parameters limit and continue are supported on requests against collections, and a response field continue is returned from all **list** operations in the collection's metadata field. A client should specify the maximum results they wish to receive in each chunk with limit and the server will return up to limit resources in the result and include a continue value if there are more resources in the collection.

As an API client, you can then pass this continue value to the API server on the next request, to instruct the server to return the next page (*chunk*) of results. By continuing until the server returns an empty continue value, you can retrieve the entire collection.

Like a **watch** operation, a continue token will expire after a short amount of time (by default 5 minutes) and return a 410 Gone if more results cannot be returned. In this case, the client will need to start from the beginning or omit the limit parameter.

For example, if there are 1,253 pods on the cluster and you want to receive chunks of 500 pods at a time, request those chunks as follows:

1. List all of the pods on a cluster, retrieving up to 500 pods each time.

```
GET /api/v1/pods?limit=500
---
200 OK
Content-Type: application/json

{
  "kind": "PodList",
  "apiVersion": "v1",
  "metadata": {
    "resourceVersion": "10245",
    "continue": "ENCODED_CONTINUE_TOKEN",
    "remainingItemCount": 753,
    ...
  },
  "items": [...] // returns pods 1-500
}
```

2. Continue the previous call, retrieving the next set of 500 pods.

```
GET /api/v1/pods?limit=500&continue=ENCODED_CONTINUE_TOKEN
---
200 OK
Content-Type: application/json

{
  "kind": "PodList",
  "apiVersion": "v1",
```

```
"metadata": {  
    "resourceVersion": "10245",  
    "continue": "ENCODED_CONTINUE_TOKEN_2",  
    "remainingItemCount": 253,  
    ...  
},  
"items": [...] // returns pods 501-1000  
}
```

3. Continue the previous call, retrieving the last 253 pods.

```
GET /api/v1/pods?limit=500&continue=ENCODED_CONTINUE_TOKEN_2  
---  
200 OK  
Content-Type: application/json  
  
{  
    "kind": "PodList",  
    "apiVersion": "v1",  
    "metadata": {  
        "resourceVersion": "10245",  
        "continue": "", // continue token is empty because we have reached the end of the list  
        ...  
    },  
    "items": [...] // returns pods 1001-1253  
}
```

Notice that the resourceVersion of the collection remains constant across each request, indicating the server is showing you a consistent snapshot of the pods. Pods that are created, updated, or deleted after version 10245 would not be shown unless you make a separate **list** request without the continue token. This allows you to break large requests into smaller chunks and then perform a **watch** operation on the full set without missing any updates.

remainingItemCount is the number of subsequent items in the collection that are not included in this response. If the **list** request contained label or field [selectors](#) then the number of remaining items is unknown and the API server does not include a remainingItemCount field in its response. If the **list** is complete (either because it is not chunking, or because this is the last chunk), then there are no more remaining items and the API server does not include a remainingItemCount field in its response. The intended use of the remainingItemCount is estimating the size of a collection.

Collections

In Kubernetes terminology, the response you get from a **list** is a *collection*. However, Kubernetes defines concrete kinds for collections of different types of resource. Collections have a kind named for the resource kind, with List appended.

When you query the API for a particular type, all items returned by that query are of that type. For example, when you **list** Services, the collection response has kind set to [ServiceList](#); each item in that collection represents a single Service. For example:

```
GET /api/v1/services
```

```
{  
  "kind": "ServiceList",  
  "apiVersion": "v1",  
  "metadata": {  
    "resourceVersion": "2947301"  
  },  
  "items": [  
    {  
      "metadata": {  
        "name": "kubernetes",  
        "namespace": "default",  
        ...  
      }  
      "metadata": {  
        "name": "kube-dns",  
        "namespace": "kube-system",  
        ...  
      }  
    }  
  ]  
}
```

There are dozens of collection types (such as PodList, ServiceList, and NodeList) defined in the Kubernetes API. You can get more information about each collection type from the [Kubernetes API](#) documentation.

Some tools, such as kubectl, represent the Kubernetes collection mechanism slightly differently from the Kubernetes API itself. Because the output of kubectl might include the response from multiple **list** operations at the API level, kubectl represents a list of items using kind: List. For example:

```
kubectl get services -A -o yaml
```

```
apiVersion: v1  
kind: List  
metadata:  
  resourceVersion: ""  
  selfLink: ""  
items:  
- apiVersion: v1  
  kind: Service  
  metadata:  
    creationTimestamp: "2021-06-03T14:54:12Z"  
    labels:  
      component: apiserver  
      provider: kubernetes  
    name: kubernetes  
    namespace: default  
    ...  
- apiVersion: v1  
  kind: Service  
  metadata:  
    annotations:  
      prometheus.io/port: "9153"  
      prometheus.io/scrape: "true"  
    creationTimestamp: "2021-06-03T14:54:14Z"  
    labels:  
      k8s-app: kube-dns
```

```
kubernetes.io/cluster-service: "true"
kubernetes.io/name: CoreDNS
name: kube-dns
namespace: kube-system
```

Note:

Keep in mind that the Kubernetes API does not have a kind named List.

kind: List is a client-side, internal implementation detail for processing collections that might be of different kinds of object. Avoid depending on kind: List in automation or other code.

Receiving resources as Tables

When you run kubectl get, the default output format is a simple tabular representation of one or more instances of a particular resource type. In the past, clients were required to reproduce the tabular and describe output implemented in kubectl to perform simple lists of objects. A few limitations of that approach include non-trivial logic when dealing with certain objects. Additionally, types provided by API aggregation or third party resources are not known at compile time. This means that generic implementations had to be in place for types unrecognized by a client.

In order to avoid potential limitations as described above, clients may request the Table representation of objects, delegating specific details of printing to the server. The Kubernetes API implements standard HTTP content type negotiation: passing an Accept header containing a value of application/json;as=Table;g=meta.k8s.io;v=v1 with a GET call will request that the server return objects in the Table content type.

For example, list all of the pods on a cluster in the Table format.

```
GET /api/v1/pods
Accept: application/json;as=Table;g=meta.k8s.io;v=v1
---
200 OK
Content-Type: application/json

{
  "kind": "Table",
  "apiVersion": "meta.k8s.io/v1",
  ...
  "columnDefinitions": [
    ...
  ]
}
```

For API resource types that do not have a custom Table definition known to the control plane, the API server returns a default Table response that consists of the resource's name and creationTimestamp fields.

```
GET /apis/crd.example.com/v1alpha1/namespaces/default/resources
---
200 OK
Content-Type: application/json
```

```
...
{
  "kind": "Table",
  "apiVersion": "meta.k8s.io/v1",
  ...
  "columnDefinitions": [
    {
      "name": "Name",
      "type": "string",
      ...
    },
    {
      "name": "Created At",
      "type": "date",
      ...
    }
  ]
}
```

Not all API resource types support a Table response; for example, a [CustomResourceDefinitions](#) might not define field-to-table mappings, and an APIService that [extends the core Kubernetes API](#) might not serve Table responses at all. If you are implementing a client that uses the Table information and must work against all resource types, including extensions, you should make requests that specify multiple content types in the Accept header. For example:

```
Accept: application/json;as=Table;g=meta.k8s.io;v=v1, application/json
```

Alternate representations of resources

By default, Kubernetes returns objects serialized to JSON with content type application/json. This is the default serialization format for the API. However, clients may request the more efficient [Protobuf representation](#) of these objects for better performance at scale. The Kubernetes API implements standard HTTP content type negotiation: passing an Accept header with a GET call will request that the server tries to return a response in your preferred media type, while sending an object in Protobuf to the server for a PUT or POST call means that you must set the Content-Type header appropriately.

The server will return a response with a Content-Type header if the requested format is supported, or the 406 Not acceptable error if none of the media types you requested are supported. All built-in resource types support the application/json media type.

See the Kubernetes [API reference](#) for a list of supported content types for each API.

For example:

1. List all of the pods on a cluster in Protobuf format.

```
GET /api/v1/pods
Accept: application/vnd.kubernetes.protobuf
---
200 OK
Content-Type: application/vnd.kubernetes.protobuf
```

```
... binary encoded PodList object
```

2. Create a pod by sending Protobuf encoded data to the server, but request a response in JSON.

```
POST /api/v1/namespaces/test/pods
Content-Type: application/vnd.kubernetes.protobuf
Accept: application/json
... binary encoded Pod object
---
200 OK
Content-Type: application/json

{
  "kind": "Pod",
  "apiVersion": "v1",
  ...
}
```

Not all API resource types support Protobuf; specifically, Protobuf isn't available for resources that are defined as [CustomResourceDefinitions](#) or are served via the [aggregation layer](#). As a client, if you might need to work with extension types you should specify multiple content types in the request Accept header to support fallback to JSON. For example:

```
Accept: application/vnd.kubernetes.protobuf, application/json
```

Kubernetes Protobuf encoding

Kubernetes uses an envelope wrapper to encode Protobuf responses. That wrapper starts with a 4 byte magic number to help identify content in disk or in etcd as Protobuf (as opposed to JSON), and then is followed by a Protobuf encoded wrapper message, which describes the encoding and type of the underlying object and then contains the object.

The wrapper format is:

A four byte magic number prefix:

```
Bytes 0-3: "k8s\x00" [0x6b, 0x38, 0x73, 0x00]
```

An encoded Protobuf message with the following IDL:

```
message Unknown {
    // typeMeta should have the string values for "kind" and "apiVersion" as set on the JSON
    // object
    optional TypeMeta typeMeta = 1;

    // raw will hold the complete serialized object in protobuf. See the protobuf definitions in the
    // client libraries for a given kind.
    optional bytes raw = 2;

    // contentEncoding is encoding used for the raw data. Unspecified means no encoding.
    optional string contentEncoding = 3;

    // contentType is the serialization method used to serialize 'raw'. Unspecified means
```

```

application/vnd.kubernetes.protobuf and is usually
  // omitted.
  optional string contentType = 4;
}

message TypeMeta {
  // apiVersion is the group/version for this type
  optional string apiVersion = 1;
  // kind is the name of the object schema. A protobuf definition should exist for this object.
  optional string kind = 2;
}

```

Note: Clients that receive a response in application/vnd.kubernetes.protobuf that does not match the expected prefix should reject the response, as future versions may need to alter the serialization format in an incompatible way and will do so by changing the prefix.

Resource deletion

When you **delete** a resource this takes place in two phases.

1. *finalization*
2. removal

```
{
  "kind": "ConfigMap",
  "apiVersion": "v1",
  "metadata": {
    "finalizers": {"url.io/neat-finalization", "other-url.io/my-finalizer"},
    "deletionTimestamp": nil,
  }
}
```

When a client first sends a **delete** to request removal of a resource, the .metadata.deletionTimestamp is set to the current time. Once the .metadata.deletionTimestamp is set, external controllers that act on finalizers may start performing their cleanup work at any time, in any order.

Order is **not** enforced between finalizers because it would introduce significant risk of stuck .metadata.finalizers.

The .metadata.finalizers field is shared: any actor with permission can reorder it. If the finalizer list were processed in order, then this might lead to a situation in which the component responsible for the first finalizer in the list is waiting for some signal (field value, external system, or other) produced by a component responsible for a finalizer later in the list, resulting in a deadlock.

Without enforced ordering, finalizers are free to order amongst themselves and are not vulnerable to ordering changes in the list.

Once the last finalizer is removed, the resource is actually removed from etcd.

Single resource API

The Kubernetes API verbs **get**, **create**, **update**, **patch**, **delete** and **proxy** support single resources only. These verbs with single resource support have no support for submitting multiple resources together in an ordered or unordered list or transaction.

When clients (including kubectl) act on a set of resources, the client makes a series of single-resource API requests, then aggregates the responses if needed.

By contrast, the Kubernetes API verbs **list** and **watch** allow getting multiple resources, and **deletecollection** allows deleting multiple resources.

Field validation

Kubernetes always validates the type of fields. For example, if a field in the API is defined as a number, you cannot set the field to a text value. If a field is defined as an array of strings, you can only provide an array. Some fields allow you to omit them, other fields are required. Omitting a required field from an API request is an error.

If you make a request with an extra field, one that the cluster's control plane does not recognize, then the behavior of the API server is more complicated.

By default, the API server drops fields that it does not recognize from an input that it receives (for example, the JSON body of a PUT request).

There are two situations where the API server drops fields that you supplied in an HTTP request.

These situations are:

1. The field is unrecognized because it is not in the resource's OpenAPI schema. (One exception to this is for [CRDs](#) that explicitly choose not to prune unknown fields via `x-kubernetes-preserve-unknown-fields`).
2. The field is duplicated in the object.

Validation for unrecognized or duplicate fields

FEATURE STATE: Kubernetes v1.27 [stable]

From 1.25 onward, unrecognized or duplicate fields in an object are detected via validation on the server when you use HTTP verbs that can submit data (POST, PUT, and PATCH). Possible levels of validation are Ignore, Warn (default), and Strict.

Ignore

The API server succeeds in handling the request as it would without the erroneous fields being set, dropping all unknown and duplicate fields and giving no indication it has done so.

Warn

(Default) The API server succeeds in handling the request, and reports a warning to the client. The warning is sent using the `Warning`: response header, adding one warning item for each unknown or duplicate field. For more information about warnings and the Kubernetes API, see the blog article [Warning: Helpful Warnings Ahead](#).

Strict

The API server rejects the request with a 400 Bad Request error when it detects any unknown or duplicate fields. The response message from the API server specifies all the unknown or duplicate fields that the API server has detected.

The field validation level is set by the `fieldValidation` query parameter.

Note:

If you submit a request that specifies an unrecognized field, and that is also invalid for a different reason (for example, the request provides a string value where the API expects an integer for a known field), then the API server responds with a 400 Bad Request error, but will not provide any information on unknown or duplicate fields (only which fatal error it encountered first).

You always receive an error response in this case, no matter what field validation level you requested.

Tools that submit requests to the server (such as `kubectl`), might set their own defaults that are different from the `Warn` validation level that the API server uses by default.

The `kubectl` tool uses the `--validate` flag to set the level of field validation. It accepts the values `ignore`, `warn`, and `strict` while also accepting the values `true` (equivalent to `strict`) and `false` (equivalent to `ignore`). The default validation setting for `kubectl` is `--validate=true`, which means strict server-side field validation.

When `kubectl` cannot connect to an API server with field validation (API servers prior to Kubernetes 1.27), it will fall back to using client-side validation. Client-side validation will be removed entirely in a future version of `kubectl`.

Note: Prior to Kubernetes 1.25 `kubectl --validate` was used to toggle client-side validation on or off as a boolean flag.

Dry-run

FEATURE STATE: Kubernetes v1.18 [stable]

When you use HTTP verbs that can modify resources (`POST`, `PUT`, `PATCH`, and `DELETE`), you can submit your request in a *dry run* mode. Dry run mode helps to evaluate a request through the typical request stages (admission chain, validation, merge conflicts) up until persisting objects to storage. The response body for the request is as close as possible to a non-dry-run response. Kubernetes guarantees that dry-run requests will not be persisted in storage or have any other side effects.

Make a dry-run request

Dry-run is triggered by setting the `dryRun` query parameter. This parameter is a string, working as an enum, and the only accepted values are:

[no value set]

Allow side effects. You request this with a query string such as `?dryRun` or `?dryRun&pretty=true`. The response is the final object that would have been persisted, or an error if the request could not be fulfilled.

All

Every stage runs as normal, except for the final storage stage where side effects are prevented.

When you set ?dryRun=All, any relevant [admission controllers](#) are run, validating admission controllers check the request post-mutation, merge is performed on PATCH, fields are defaulted, and schema validation occurs. The changes are not persisted to the underlying storage, but the final object which would have been persisted is still returned to the user, along with the normal status code.

If the non-dry-run version of a request would trigger an admission controller that has side effects, the request will be failed rather than risk an unwanted side effect. All built in admission control plugins support dry-run. Additionally, admission webhooks can declare in their [configuration object](#) that they do not have side effects, by setting their sideEffects field to None.

Note: If a webhook actually does have side effects, then the sideEffects field should be set to "NoneOnDryRun". That change is appropriate provided that the webhook is also be modified to understand the DryRun field in AdmissionReview, and to prevent side effects on any request marked as dry runs.

Here is an example dry-run request that uses ?dryRun=All:

```
POST /api/v1/namespaces/test/pods?dryRun=All
Content-Type: application/json
Accept: application/json
```

The response would look the same as for non-dry-run request, but the values of some generated fields may differ.

Generated values

Some values of an object are typically generated before the object is persisted. It is important not to rely upon the values of these fields set by a dry-run request, since these values will likely be different in dry-run mode from when the real request is made. Some of these fields are:

- name: if generateName is set, name will have a unique random name
- creationTimestamp / deletionTimestamp: records the time of creation/deletion
- UID: [uniquely identifies](#) the object and is randomly generated (non-deterministic)
- resourceVersion: tracks the persisted version of the object
- Any field set by a mutating admission controller
- For the Service resource: Ports or IP addresses that the kube-apiserver assigns to Service objects

Dry-run authorization

Authorization for dry-run and non-dry-run requests is identical. Thus, to make a dry-run request, you must be authorized to make the non-dry-run request.

For example, to run a dry-run **patch** for a Deployment, you must be authorized to perform that **patch**. Here is an example of a rule for Kubernetes [RBAC](#) that allows patching Deployments:

```
rules:
- apiGroups: ["apps"]
```

```
resources: ["deployments"]
verbs: ["patch"]
```

See [Authorization Overview](#).

Updates to existing resources

Kubernetes provides several ways to update existing objects. You can read [choosing an update mechanism](#) to learn about which approach might be best for your use case.

You can overwrite (**update**) an existing resource - for example, a ConfigMap - using an HTTP PUT. For a PUT request, it is the client's responsibility to specify the resourceVersion (taking this from the object being updated). Kubernetes uses that resourceVersion information so that the API server can detect lost updates and reject requests made by a client that is out of date with the cluster. In the event that the resource has changed (the resourceVersion the client provided is stale), the API server returns a 409 Conflict error response.

Instead of sending a PUT request, the client can send an instruction to the API server to **patch** an existing resource. A **patch** is typically appropriate if the change that the client wants to make isn't conditional on the existing data. Clients that need effective detection of lost updates should consider making their request conditional on the existing resourceVersion (either HTTP PUT or HTTP PATCH), and then handle any retries that are needed in case there is a conflict.

The Kubernetes API supports four different PATCH operations, determined by their corresponding HTTP Content-Type header:

`application/apply-patch+yaml`

Server Side Apply YAML (a Kubernetes-specific extension, based on YAML). All JSON documents are valid YAML, so you can also submit JSON using this media type. See [Server Side Apply serialization](#) for more details.

To Kubernetes, this is a **create** operation if the object does not exist, or a **patch** operation if the object already exists.

`application/json-patch+json`

JSON Patch, as defined in [RFC6902](#). A JSON patch is a sequence of operations that are executed on the resource; for example `{"op": "add", "path": "/a/b/c", "value": ["foo", "bar"]}`. To Kubernetes, this is a **patch** operation.

A **patch** using `application/json-patch+json` can include conditions to validate consistency, allowing the operation to fail if those conditions are not met (for example, to avoid a lost update).

`application/merge-patch+json`

JSON Merge Patch, as defined in [RFC7386](#). A JSON Merge Patch is essentially a partial representation of the resource. The submitted JSON is combined with the current resource to create a new one, then the new one is saved.

To Kubernetes, this is a **patch** operation.

`application стратегиче-merge-patch+json`

Strategic Merge Patch (a Kubernetes-specific extension based on JSON). Strategic Merge Patch is a custom implementation of JSON Merge Patch. You can only use Strategic Merge Patch with built-in APIs, or with aggregated API servers that have special support for it. You cannot use `application/strategic-merge-patch+json` with any API defined using a [CustomResourceDefinition](#).

Note: The Kubernetes *server side apply* mechanism has superseded Strategic Merge Patch.

Kubernetes' [Server Side Apply](#) feature allows the control plane to track managed fields for newly created objects. Server Side Apply provides a clear pattern for managing field conflicts, offers server-side **apply** and **update** operations, and replaces the client-side functionality of kubectl apply.

For Server-Side Apply, Kubernetes treats the request as a **create** if the object does not yet exist, and a **patch** otherwise. For other requests that use PATCH at the HTTP level, the logical Kubernetes operation is always **patch**.

See [Server Side Apply](#) for more details.

Choosing an update mechanism

HTTP PUT to replace existing resource

The **update** (HTTP PUT) operation is simple to implement and flexible, but has drawbacks:

- You need to handle conflicts where the `resourceVersion` of the object changes between your client reading it and trying to write it back. Kubernetes always detects the conflict, but you as the client author need to implement retries.
- You might accidentally drop fields if you decode an object locally (for example, using `client-go`, you could receive fields that your client does not know how to handle - and then drop them as part of your update).
- If there's a lot of contention on the object (even on a field, or set of fields, that you're not trying to edit), you might have trouble sending the update. The problem is worse for larger objects and for objects with many fields.

HTTP PATCH using JSON Patch

A **patch** update is helpful, because:

- As you're only sending differences, you have less data to send in the PATCH request.
- You can make changes that rely on existing values, such as copying the value of a particular field into an annotation.
- Unlike with an **update** (HTTP PUT), making your change can happen right away even if there are frequent changes to unrelated fields): you usually would not need to retry.
 - You might still need to specify the `resourceVersion` (to match an existing object) if you want to be extra careful to avoid lost updates
 - It's still good practice to write in some retry logic in case of errors.
- You can use test conditions to carefully craft specific update conditions. For example, you can increment a counter without reading it if the existing value matches what you expect. You can do this with no lost update risk, even if the object has changed in other ways since you last wrote to it. (If the test condition fails, you can fall back to reading the current value and then write back the changed number).

However:

- you need more local (client) logic to build the patch; it helps a lot if you have a library implementation of JSON Patch, or even for making a JSON Patch specifically against Kubernetes
- as the author of client software, you need to be careful when building the patch (the HTTP request body) not to drop fields (the order of operations matters)

HTTP PATCH using Server-Side Apply

Server-Side Apply has some clear benefits:

- A single round trip: it rarely requires making a GET request first.
 - and you can still detect conflicts for unexpected changes
 - you have the option to force override a conflict, if appropriate
- Client implementations are easy to make
- You get an atomic create-or-update operation without extra effort (similar to UPSERT in some SQL dialects)

However:

- Server-Side Apply does not work at all for field changes that depend on a current value of the object
- You can only apply updates to objects. Some resources in the Kubernetes HTTP API are not objects (they do not have a .metadata field), and Server-Side Apply is only relevant for Kubernetes objects.

Resource versions

Resource versions are strings that identify the server's internal version of an object. Resource versions can be used by clients to determine when objects have changed, or to express data consistency requirements when getting, listing and watching resources. Resource versions must be treated as opaque by clients and passed unmodified back to the server.

You must not assume resource versions are numeric or collatable. API clients may only compare two resource versions for equality (this means that you must not compare resource versions for greater-than or less-than relationships).

resourceVersion fields in metadata

Clients find resource versions in resources, including the resources from the response stream for a **watch**, or when using **list** to enumerate resources.

[**v1.meta/ObjectMeta**](#) - The metadata.resourceVersion of a resource instance identifies the resource version the instance was last modified at.

[**v1.meta/ListMeta**](#) - The metadata.resourceVersion of a resource collection (the response to a **list**) identifies the resource version at which the collection was constructed.

resourceVersion parameters in query strings

The **get**, **list**, and **watch** operations support the resourceVersion parameter. From version v1.19, Kubernetes API servers also support the resourceVersionMatch parameter on *list* requests.

The API server interprets the resourceVersion parameter differently depending on the operation you request, and on the value of resourceVersion. If you set resourceVersionMatch then this also affects the way matching happens.

Semantics for get and list

For **get** and **list**, the semantics of resourceVersion are:

get:

resourceVersion unset	resourceVersion="0"	resourceVersion="{value other than 0}"
Most Recent	Any	Not older than

list:

From version v1.19, Kubernetes API servers support the resourceVersionMatch parameter on **list** requests. If you set both resourceVersion and resourceVersionMatch, the resourceVersionMatch parameter determines how the API server interprets resourceVersion.

You should always set the resourceVersionMatch parameter when setting resourceVersion on a **list** request. However, be prepared to handle the case where the API server that responds is unaware of resourceVersionMatch and ignores it.

Unless you have strong consistency requirements, using resourceVersionMatch=NotOlderThan and a known resourceVersion is preferable since it can achieve better performance and scalability of your cluster than leaving resourceVersion and resourceVersionMatch unset, which requires quorum read to be served.

Setting the resourceVersionMatch parameter without setting resourceVersion is not valid.

This table explains the behavior of **list** requests with various combinations of resourceVersion and resourceVersionMatch:

resourceVersionMatch and paging parameters for list

resourceVersionMatch param	paging params	resourceVersion not set	resourceVersion="0"	resourceVersion "{value other than 0}"
unset	limit unset	Most Recent	Any	Not older
unset	limit=<n>, continue unset	Most Recent	Any	Exact
unset	limit=<n>, continue=<token>	Continue Token, Exact	Invalid, treated as Continue Token, Exact	Invalid, H Request
resourceVersionMatch=Exact	limit unset	Invalid	Invalid	Exact
resourceVersionMatch=Exact	limit=<n>, continue unset	Invalid	Invalid	Exact
resourceVersionMatch=NotOlderThan	limit unset	Invalid	Any	Not older
resourceVersionMatch=NotOlderThan	limit=<n>, continue unset	Invalid	Any	Not older

Note: If your cluster's API server does not honor the resourceVersionMatch parameter, the behavior is the same as if you did not set it.

The meaning of the **get** and **list** semantics are:

Any

Return data at any resource version. The newest available resource version is preferred, but strong consistency is not required; data at any resource version may be served. It is

possible for the request to return data at a much older resource version that the client has previously observed, particularly in high availability configurations, due to partitions or stale caches. Clients that cannot tolerate this should not use this semantic.

Most recent

Return data at the most recent resource version. The returned data must be consistent (in detail: served from etcd via a quorum read).

Not older than

Return data at least as new as the provided resourceVersion. The newest available data is preferred, but any data not older than the provided resourceVersion may be served. For **list** requests to servers that honor the resourceVersionMatch parameter, this guarantees that the collection's .metadata.resourceVersion is not older than the requested resourceVersion, but does not make any guarantee about the .metadata.resourceVersion of any of the items in that collection.

Exact

Return data at the exact resource version provided. If the provided resourceVersion is unavailable, the server responds with HTTP 410 "Gone". For **list** requests to servers that honor the resourceVersionMatch parameter, this guarantees that the collection's .metadata.resourceVersion is the same as the resourceVersion you requested in the query string. That guarantee does not apply to the .metadata.resourceVersion of any items within that collection.

Continue Token, Exact

Return data at the resource version of the initial paginated **list** call. The returned *continue tokens* are responsible for keeping track of the initially provided resource version for all paginated **list** calls after the initial paginated **list**.

Note: When you **list** resources and receive a collection response, the response includes the [list metadata](#) of the collection as well as [object metadata](#) for each item in that collection. For individual objects found within a collection response, .metadata.resourceVersion tracks when that object was last updated, and not how up-to-date the object is when served.

When using resourceVersionMatch=NotOlderThan and limit is set, clients must handle HTTP 410 "Gone" responses. For example, the client might retry with a newer resourceVersion or fall back to resourceVersion="".

When using resourceVersionMatch=Exact and limit is unset, clients must verify that the collection's .metadata.resourceVersion matches the requested resourceVersion, and handle the case where it does not. For example, the client might fall back to a request with limit set.

Semantics for watch

For **watch**, the semantics of resource version are:

watch:

resourceVersion for watch

resourceVersion unset	resourceVersion="0"	resourceVersion="{value other than 0}"
Get State and Start at Most Recent	Get State and Start at Any	Start at Exact

The meaning of those **watch** semantics are:

Get State and Start at Any

Caution: Watches initialized this way may return arbitrarily stale data. Please review this semantic before using it, and favor the other semantics where possible.

Start a **watch** at any resource version; the most recent resource version available is preferred, but not required. Any starting resource version is allowed. It is possible for the **watch** to start at a much older resource version that the client has previously observed, particularly in high availability configurations, due to partitions or stale caches. Clients that cannot tolerate this apparent rewinding should not start a **watch** with this semantic. To establish initial state, the **watch** begins with synthetic "Added" events for all resource instances that exist at the starting resource version. All following watch events are for all changes that occurred after the resource version the **watch** started at.

Get State and Start at Most Recent

Start a **watch** at the most recent resource version, which must be consistent (in detail: served from etcd via a quorum read). To establish initial state, the **watch** begins with synthetic "Added" events of all resources instances that exist at the starting resource version. All following watch events are for all changes that occurred after the resource version the **watch** started at.

Start at Exact

Start a **watch** at an exact resource version. The watch events are for all changes after the provided resource version. Unlike "Get State and Start at Most Recent" and "Get State and Start at Any", the **watch** is not started with synthetic "Added" events for the provided resource version. The client is assumed to already have the initial state at the starting resource version since the client provided the resource version.

"410 Gone" responses

Servers are not required to serve all older resource versions and may return a HTTP 410 (Gone) status code if a client requests a resourceVersion older than the server has retained. Clients must be able to tolerate 410 (Gone) responses. See [Efficient detection of changes](#) for details on how to handle 410 (Gone) responses when watching resources.

If you request a resourceVersion outside the applicable limit then, depending on whether a request is served from cache or not, the API server may reply with a 410 Gone HTTP response.

Unavailable resource versions

Servers are not required to serve unrecognized resource versions. If you request **list** or **get** for a resource version that the API server does not recognize, then the API server may either:

- wait briefly for the resource version to become available, then timeout with a 504 (Gateway Timeout) if the provided resource versions does not become available in a reasonable amount of time;
- respond with a Retry-After response header indicating how many seconds a client should wait before retrying the request.

If you request a resource version that an API server does not recognize, the kube-apiserver additionally identifies its error responses with a "Too large resource version" message.

If you make a **watch** request for an unrecognized resource version, the API server may wait indefinitely (until the request timeout) for the resource version to become available.

Server-Side Apply

FEATURE STATE: Kubernetes v1.22 [stable]

Kubernetes supports multiple appliers collaborating to manage the fields of a single [object](#).

Server-Side Apply provides an optional mechanism for your cluster's control plane to track changes to an object's fields. At the level of a specific resource, Server-Side Apply records and tracks information about control over the fields of that object.

Server-Side Apply helps users and [controllers](#) manage their resources through declarative configuration. Clients can create and modify [objects](#) declaratively by submitting their *fully specified intent*.

A fully specified intent is a partial object that only includes the fields and values for which the user has an opinion. That intent either creates a new object (using default values for unspecified fields), or is [combined](#), by the API server, with the existing object.

[Comparison with Client-Side Apply](#) explains how Server-Side Apply differs from the original, client-side kubectl apply implementation.

Field management

The Kubernetes API server tracks *managed fields* for all newly created objects.

When trying to apply an object, fields that have a different value and are owned by another [manager](#) will result in a [conflict](#). This is done in order to signal that the operation might undo another collaborator's changes. Writes to objects with managed fields can be forced, in which case the value of any conflicted field will be overridden, and the ownership will be transferred.

Whenever a field's value does change, ownership moves from its current manager to the manager making the change.

Apply checks if there are any other field managers that also own the field. If the field is not owned by any other field managers, that field is set to its default value (if there is one), or otherwise is deleted from the object. The same rule applies to fields that are lists, associative lists, or maps.

For a user to manage a field, in the Server-Side Apply sense, means that the user relies on and expects the value of the field not to change. The user who last made an assertion about the value of a field will be recorded as the current field manager. This can be done by changing the field manager details explicitly using HTTP POST ([create](#)), PUT ([update](#)), or non-apply PATCH ([patch](#)). You can also declare and record a field manager by including a value for that field in a Server-Side Apply operation.

A Server-Side Apply **patch** request requires the client to provide its identity as a [field manager](#). When using Server-Side Apply, trying to change a field that is controlled by a different manager results in a rejected request unless the client forces an override. For details of overrides, see [Conflicts](#).

When two or more appliers set a field to the same value, they share ownership of that field. Any subsequent attempt to change the value of the shared field, by any of the appliers, results

in a conflict. Shared field owners may give up ownership of a field by making a Server-Side Apply **patch** request that doesn't include that field.

Field management details are stored in a `managedFields` field that is part of an object's [metadata](#).

If you remove a field from a manifest and apply that manifest, Server-Side Apply checks if there are any other field managers that also own the field. If the field is not owned by any other field managers, it is either deleted from the live object or reset to its default value, if it has one. The same rule applies to associative list or map items.

Compared to the (legacy) [kubectl.kubernetes.io/last-applied-configuration](#) annotation managed by kubectl, Server-Side Apply uses a more declarative approach, that tracks a user's (or client's) field management, rather than a user's last applied state. As a side effect of using Server-Side Apply, information about which field manager manages each field in an object also becomes available.

Example

A simple example of an object created using Server-Side Apply could look like this:

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: test-cm
  namespace: default
  labels:
    test-label: test
managedFields:
- manager: kubectl
  operation: Apply # note capitalization: "Apply" (or "Update")
  apiVersion: v1
  time: "2010-10-10T0:00:00Z"
  fieldsType: FieldsV1
  fieldsV1:
    f:metadata:
      f:labels:
        f:test-label: {}
      f:data:
        f:key: {}
data:
  key: some value
```

That example ConfigMap object contains a single field management record in `.metadata.managedFields`. The field management record consists of basic information about the managing entity itself, plus details about the fields being managed and the relevant operation (Apply or Update). If the request that last changed that field was a Server-Side Apply **patch** then the value of operation is Apply; otherwise, it is Update.

There is another possible outcome. A client could submit an invalid request body. If the fully specified intent does not produce a valid object, the request fails.

It is however possible to change `.metadata.managedFields` through an **update**, or through a **patch** operation that does not use Server-Side Apply. Doing so is highly discouraged, but might be a reasonable option to try if, for example, the `.metadata.managedFields` get into an inconsistent state (which should not happen in normal operations).

The format of `managedFields` is [described](#) in the Kubernetes API reference.

Caution: The `.metadata.managedFields` field is managed by the API server. You should avoid updating it manually.

Conflicts

A *conflict* is a special status error that occurs when an Apply operation tries to change a field that another manager also claims to manage. This prevents an applier from unintentionally overwriting the value set by another user. When this occurs, the applier has 3 options to resolve the conflicts:

- **Overwrite value, become sole manager:** If overwriting the value was intentional (or if the applier is an automated process like a controller) the applier should set the `force` query parameter to true (for `kubectl apply`, you use the `--force-conflicts` command line parameter), and make the request again. This forces the operation to succeed, changes the value of the field, and removes the field from all other managers' entries in `managedFields`.
- **Don't overwrite value, give up management claim:** If the applier doesn't care about the value of the field any more, the applier can remove it from their local model of the resource, and make a new request with that particular field omitted. This leaves the value unchanged, and causes the field to be removed from the applier's entry in `managedFields`.
- **Don't overwrite value, become shared manager:** If the applier still cares about the value of a field, but doesn't want to overwrite it, they can change the value of that field in their local model of the resource so as to match the value of the object on the server, and then make a new request that takes into account that local update. Doing so leaves the value unchanged, and causes that field's management to be shared by the applier along with all other field managers that already claimed to manage it.

Field managers

Managers identify distinct workflows that are modifying the object (especially useful on conflicts!), and can be specified through the [fieldManager](#) query parameter as part of a modifying request. When you `Apply` to a resource, the `fieldManager` parameter is required. For other updates, the API server infers a field manager identity from the "User-Agent:" HTTP header (if present).

When you use the `kubectl` tool to perform a Server-Side Apply operation, `kubectl` sets the manager identity to "kubectl" by default.

Serialization

At the protocol level, Kubernetes represents Server-Side Apply message bodies as [YAML](#), with the media type `application/apply-patch+yaml`.

Note:

Whether you are submitting JSON data or YAML data, use application/apply-patch+yaml as the Content-Type header value.

All JSON documents are valid YAML.

The serialization is the same as for Kubernetes objects, with the exception that clients are not required to send a complete object.

Here's an example of a Server-Side Apply message body (fully specified intent):

```
{  
  "apiVersion": "v1",  
  "kind": "ConfigMap"  
}
```

(this would make a no-change update, provided that it was sent as the body of a **patch** request to a valid v1/configmaps resource, and with the appropriate request Content-Type).

Operations in scope for field management

The Kubernetes API operations where field management is considered are:

1. Server-Side Apply (HTTP PATCH, with content type application/apply-patch+yaml)
2. Replacing an existing object (**update** to Kubernetes; PUT at the HTTP level)

Both operations update .metadata.managedFields, but behave a little differently.

Unless you specify a forced override, an apply operation that encounters field-level conflicts always fails; by contrast, if you make a change using **update** that would affect a managed field, a conflict never provokes failure of the operation.

All Server-Side Apply **patch** requests are required to identify themselves by providing a fieldManager query parameter, while the query parameter is optional for **update** operations. Finally, when using the Apply operation you cannot define managedFields in the body of the request that you submit.

An example object with multiple managers could look like this:

```
---  
apiVersion: v1  
kind: ConfigMap  
metadata:  
  name: test-cm  
  namespace: default  
  labels:  
    test-label: test  
managedFields:  
  - manager: kubectl  
    operation: Apply  
    apiVersion: v1  
    fields:  
      f:metadata:  
        f:labels:
```

```

f:test-label: {}
- manager: kube-controller-manager
  operation: Update
  apiVersion: v1
  time: '2019-03-30T16:00:00.000Z'
  fields:
    f:data:
      f:key: {}
data:
  key: new value

```

In this example, a second operation was run as an **update** by the manager called kube-controller-manager. The update request succeeded and changed a value in the data field, which caused that field's management to change to the kube-controller-manager.

If this update has instead been attempted using Server-Side Apply, the request would have failed due to conflicting ownership.

Merge strategy

The merging strategy, implemented with Server-Side Apply, provides a generally more stable object lifecycle. Server-Side Apply tries to merge fields based on the actor who manages them instead of overruling based on values. This way multiple actors can update the same object without causing unexpected interference.

When a user sends a *fully-specified intent* object to the Server-Side Apply endpoint, the server merges it with the live object favoring the value from the request body if it is specified in both places. If the set of items present in the applied config is not a superset of the items applied by the same user last time, each missing item not managed by any other appliers is removed. For more information about how an object's schema is used to make decisions when merging, see [sig.k8s.io/structured-merge-diff](#).

The Kubernetes API (and the Go code that implements that API for Kubernetes) allows defining *merge strategy markers*. These markers describe the merge strategy supported for fields within Kubernetes objects. For a [CustomResourceDefinition](#), you can set these markers when you define the custom resource.

Golang marker	OpenAPI extension	Possible values	Description
//+listType	x-kubernetes-list-type	atomic/set/map	Applicable to lists. set applies to lists that include only scalar elements. These elements must be unique. map applies to lists of nested types only. The key values (see listMapKey) must be unique in the list. atomic can apply to any list. If configured as atomic, the entire list is replaced during merge. At any point in time, a single manager owns the list. If set or map, different managers can manage entries separately.
//+listMapKey	x-kubernetes-list-map-keys	List of field names, e.g. ["port", "protocol"]	Only applicable when +listType=map. A list of field names whose values uniquely identify entries in the list. While there can be multiple keys, listMapKey is singular because keys need to

Golang marker	OpenAPI extension	Possible values	Description
			be specified individually in the Go type. The key fields must be scalars.
//+mapType	x-kubernetes-map-type	atomic/granular	Applicable to maps. atomic means that the map can only be entirely replaced by a single manager. granular means that the map supports separate managers updating individual fields.
//+structType	x-kubernetes-map-type	atomic/granular	Applicable to structs; otherwise same usage and OpenAPI annotation as //+mapType.

If `listType` is missing, the API server interprets a `patchStrategy=merge` marker as a `listType=map` and the corresponding `patchMergeKey` marker as a `listMapKey`.

The atomic list type is recursive.

(In the [Go](#) code for Kubernetes, these markers are specified as comments and code authors need not repeat them as field tags).

Custom resources and Server-Side Apply

By default, Server-Side Apply treats custom resources as unstructured data. All keys are treated the same as struct fields, and all lists are considered atomic.

If the CustomResourceDefinition defines a [schema](#) that contains annotations as defined in the previous [Merge Strategy](#) section, these annotations will be used when merging objects of this type.

Compatibility across topology changes

On rare occurrences, the author for a CustomResourceDefinition (CRD) or built-in may want to change the specific topology of a field in their resource, without incrementing its API version. Changing the topology of types, by upgrading the cluster or updating the CRD, has different consequences when updating existing objects. There are two categories of changes: when a field goes from map/set/granular to atomic, and the other way around.

When the `listType`, `mapType`, or `structType` changes from map/set/granular to atomic, the whole list, map, or struct of existing objects will end-up being owned by actors who owned an element of these types. This means that any further change to these objects would cause a conflict.

When a `listType`, `mapType`, or `structType` changes from atomic to map/set/granular, the API server is unable to infer the new ownership of these fields. Because of that, no conflict will be produced when objects have these fields updated. For that reason, it is not recommended to change a type from atomic to map/set/granular.

Take for example, the custom resource:

```
---
apiVersion: example.com/v1
kind: Foo
```

```

metadata:
  name: foo-sample
  managedFields:
    - manager: "manager-one"
      operation: Apply
    apiVersion: example.com/v1
  fields:
    f:spec:
      f:data: {}
spec:
  data:
    key1: val1
    key2: val2

```

Before spec.data gets changed from atomic to granular, manager-one owns the field spec.data, and all the fields within it (key1 and key2). When the CRD gets changed to make spec.data granular, manager-one continues to own the top-level field spec.data (meaning no other managers can delete the map called data without a conflict), but it no longer owns key1 and key2, so another manager can then modify or delete those fields without conflict.

Using Server-Side Apply in a controller

As a developer of a controller, you can use Server-Side Apply as a way to simplify the update logic of your controller. The main differences with a read-modify-write and/or patch are the following:

- the applied object must contain all the fields that the controller cares about.
- there is no way to remove fields that haven't been applied by the controller before (controller can still send a **patch** or **update** for these use-cases).
- the object doesn't have to be read beforehand; resourceVersion doesn't have to be specified.

It is strongly recommended for controllers to always force conflicts on objects that they own and manage, since they might not be able to resolve or act on these conflicts.

Transferring ownership

In addition to the concurrency controls provided by [conflict resolution](#), Server-Side Apply provides ways to perform coordinated field ownership transfers from users to controllers.

This is best explained by example. Let's look at how to safely transfer ownership of the replicas field from a user to a controller while enabling automatic horizontal scaling for a Deployment, using the HorizontalPodAutoscaler resource and its accompanying controller.

Say a user has defined Deployment with replicas set to the desired value:

[application/ssa/nginx-deployment.yaml](#)

```

apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment

```

```
labels:  
  app: nginx  
spec:  
  replicas: 3  
  selector:  
    matchLabels:  
      app: nginx  
template:  
  metadata:  
    labels:  
      app: nginx  
  spec:  
    containers:  
    - name: nginx  
      image: nginx:1.14.2
```

And the user has created the Deployment using Server-Side Apply, like so:

```
kubectl apply -f https://k8s.io/examples/application/ssa/nginx-deployment.yaml --server-side
```

Then later, automatic scaling is enabled for the Deployment; for example:

```
kubectl autoscale deployment nginx-deployment --cpu-percent=50 --min=1 --max=10
```

Now, the user would like to remove replicas from their configuration, so they don't accidentally fight with the HorizontalPodAutoscaler (HPA) and its controller. However, there is a race: it might take some time before the HPA feels the need to adjust `.spec.replicas`; if the user removes `.spec.replicas` before the HPA writes to the field and becomes its owner, then the API server would set `.spec.replicas` to 1 (the default replica count for Deployment). This is not what the user wants to happen, even temporarily - it might well degrade a running workload.

There are two solutions:

- (basic) Leave replicas in the configuration; when the HPA eventually writes to that field, the system gives the user a conflict over it. At that point, it is safe to remove from the configuration.
- (more advanced) If, however, the user doesn't want to wait, for example because they want to keep the cluster legible to their colleagues, then they can take the following steps to make it safe to remove replicas from their configuration:

First, the user defines a new manifest containing only the `replicas` field:

```
# Save this file as 'nginx-deployment-replicas-only.yaml'.  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: nginx-deployment  
spec:  
  replicas: 3
```

Note: The YAML file for SSA in this case only contains the fields you want to change. You are not supposed to provide a fully compliant Deployment manifest if you only want to modify the `spec.replicas` field using SSA.

The user applies that manifest using a private field manager name. In this example, the user picked handover-to-hpa:

```
kubectl apply -f nginx-deployment-replicas-only.yaml \
--server-side --field-manager=handover-to-hpa \
--validate=false
```

If the apply results in a conflict with the HPA controller, then do nothing. The conflict indicates the controller has claimed the field earlier in the process than it sometimes does.

At this point the user may remove the replicas field from their manifest:

[application/ssa/nginx-deployment-no-replicas.yaml](#)

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
```

Note that whenever the HPA controller sets the replicas field to a new value, the temporary field manager will no longer own any fields and will be automatically deleted. No further clean up is required.

Transferring ownership between managers

Field managers can transfer ownership of a field between each other by setting the field to the same value in both of their applied configurations, causing them to share ownership of the field. Once the managers share ownership of the field, one of them can remove the field from their applied configuration to give up ownership and complete the transfer to the other field manager.

Comparison with Client-Side Apply

Server-Side Apply is meant both as a replacement for the original client-side implementation of the kubectl apply subcommand, and as simple and effective mechanism for [controllers](#) to enact their changes.

Compared to the last-applied annotation managed by kubectl, Server-Side Apply uses a more declarative approach, which tracks an object's field management, rather than a user's last

applied state. This means that as a side effect of using Server-Side Apply, information about which field manager manages each field in an object also becomes available.

A consequence of the conflict detection and resolution implemented by Server-Side Apply is that an applier always has up to date field values in their local state. If they don't, they get a conflict the next time they apply. Any of the three options to resolve conflicts results in the applied configuration being an up to date subset of the object on the server's fields.

This is different from Client-Side Apply, where outdated values which have been overwritten by other users are left in an applier's local config. These values only become accurate when the user updates that specific field, if ever, and an applier has no way of knowing whether their next apply will overwrite other users' changes.

Another difference is that an applier using Client-Side Apply is unable to change the API version they are using, but Server-Side Apply supports this use case.

Migration between client-side and server-side apply

Upgrading from client-side apply to server-side apply

Client-side apply users who manage a resource with `kubectl apply` can start using server-side apply with the following flag.

```
kubectl apply --server-side [--dry-run=server]
```

By default, field management of the object transfers from client-side apply to `kubectl server-side apply`, without encountering conflicts.

Caution:

Keep the `last-applied-configuration` annotation up to date. The annotation infers client-side applies managed fields. Any fields not managed by client-side apply raise conflicts.

For example, if you used `kubectl scale` to update the `replicas` field after client-side apply, then this field is not owned by client-side apply and creates conflicts on `kubectl apply --server-side`.

This behavior applies to server-side apply with the `kubectl` field manager. As an exception, you can opt-out of this behavior by specifying a different, non-default field manager, as seen in the following example. The default field manager for `kubectl server-side apply` is `kubectl`.

```
kubectl apply --server-side --field-manager=my-manager [--dry-run=server]
```

Downgrading from server-side apply to client-side apply

If you manage a resource with `kubectl apply --server-side`, you can downgrade to client-side apply directly with `kubectl apply`.

Downgrading works because `kubectl Server-Side Apply` keeps the `last-applied-configuration` annotation up-to-date if you use `kubectl apply`.

This behavior applies to Server-Side Apply with the `kubectl` field manager. As an exception, you can opt-out of this behavior by specifying a different, non-default field manager, as seen in the following example. The default field manager for `kubectl server-side apply` is `kubectl`.

```
kubectl apply --server-side --field-manager=my-manager [--dry-run=server]
```

API implementation

The PATCH verb for a resource that supports Server-Side Apply can accept the unofficial application/apply-patch+yaml content type. Users of Server-Side Apply can send partially specified objects as YAML as the body of a PATCH request to the URI of a resource. When applying a configuration, you should always include all the fields that are important to the outcome (such as a desired state) that you want to define.

All JSON messages are valid YAML. Some clients specify Server-Side Apply requests using YAML request bodies that are also valid JSON.

Access control and permissions

Since Server-Side Apply is a type of PATCH, a principal (such as a Role for Kubernetes [RBAC](#)) requires the **patch** permission to edit existing resources, and also needs the **create** verb permission in order to create new resources with Server-Side Apply.

Clearing managedFields

It is possible to strip all managedFields from an object by overwriting them using a **patch** (JSON Merge Patch, Strategic Merge Patch, JSON Patch), or through an **update** (HTTP PUT); in other words, through every write operation other than **apply**. This can be done by overwriting the managedFields field with an empty entry. Two examples are:

```
PATCH /api/v1/namespaces/default/configmaps/example-cm
```

```
Accept: application/json
```

```
Content-Type: application/merge-patch+json
```

```
{  
  "metadata": {  
    "managedFields": [  
      {}  
    ]  
  }  
}
```

```
PATCH /api/v1/namespaces/default/configmaps/example-cm
```

```
Accept: application/json
```

```
Content-Type: application/json-patch+json
```

```
If-Match: 1234567890123456789
```

```
[{"op": "replace", "path": "/metadata/managedFields", "value": [{}]}]
```

This will overwrite the managedFields with a list containing a single empty entry that then results in the managedFields being stripped entirely from the object. Note that setting the managedFields to an empty list will not reset the field. This is on purpose, so managedFields never get stripped by clients not aware of the field.

In cases where the reset operation is combined with changes to other fields than the managedFields, this will result in the managedFields being reset first and the other changes

being processed afterwards. As a result the applier takes ownership of any fields updated in the same request.

Note: Server-Side Apply does not correctly track ownership on sub-resources that don't receive the resource object type. If you are using Server-Side Apply with such a sub-resource, the changed fields may not be tracked.

What's next

You can read about `managedFields` within the Kubernetes API reference for the [metadata](#) top level field.

Client Libraries

This page contains an overview of the client libraries for using the Kubernetes API from various programming languages.

To write applications using the [Kubernetes REST API](#), you do not need to implement the API calls and request/response types yourself. You can use a client library for the programming language you are using.

Client libraries often handle common tasks such as authentication for you. Most client libraries can discover and use the Kubernetes Service Account to authenticate if the API client is running inside the Kubernetes cluster, or can understand the [kubeconfig file](#) format to read the credentials and the API Server address.

Officially-supported Kubernetes client libraries

The following client libraries are officially maintained by [Kubernetes SIG API Machinery](#).

Language	Client Library	Sample Programs
C	github.com/kubernetes-client/c	browse
dotnet	github.com/kubernetes-client/csharp	browse
Go	github.com/kubernetes/client-go/	browse
Haskell	github.com/kubernetes-client/haskell	browse
Java	github.com/kubernetes-client/java	browse
JavaScript	github.com/kubernetes-client/javascript	browse
Perl	github.com/kubernetes-client/perl/	browse
Python	github.com/kubernetes-client/python/	browse
Ruby	github.com/kubernetes-client/ruby/	browse

Community-maintained client libraries

Note: This section links to third party projects that provide functionality required by Kubernetes. The Kubernetes project authors aren't responsible for these projects, which are listed alphabetically. To add a project to this list, read the [content guide](#) before submitting a change. [More information](#).

The following Kubernetes API client libraries are provided and maintained by their authors, not the Kubernetes team.

Language	Client Library
Clojure	github.com/yanatan16/clj-kubernetes-api
DotNet	github.com/tonnyeremin/kubernetes_gen
DotNet (RestSharp)	github.com/masroorhasan/Kubernetes.DotNet
Elixir	github.com/obmarg/kazan
Elixir	github.com/coryodaniel/k8s
Java (OSGi)	bitbucket.org/amdatulabs/amdatu-kubernetes
Java (Fabric8, OSGi)	github.com/fabric8io/kubernetes-client
Java	github.com/manusa/yakc
Lisp	github.com/brendandburns/cl-k8s
Lisp	github.com/xh4/cube
Node.js (TypeScript)	github.com/Goyoo/node-k8s-client
Node.js	github.com/ajpauwels/easy-k8s
Node.js	github.com/godaddy/kubernetes-client
Node.js	github.com/tenxcloud/node-kubernetes-client
Perl	metacpan.org/pod/Net::Kubernetes
PHP	github.com/allansun/kubernetes-php-client
PHP	github.com/maclof/kubernetes-client
PHP	github.com/travisghansen/kubernetes-client-php
PHP	github.com/renoki-co/php-k8s
Python	github.com/fiaas/k8s
Python	github.com/gt-system/lightkube
Python	github.com/kr8s-org/kr8s
Python	github.com/mnubo/kubernetes-py
Python	github.com/tomplus/kubernetes asyncio
Python	github.com/Frankkkkk/pykorm
Ruby	github.com/abonas/kubeclient
Ruby	github.com/k8s-ruby/k8s-ruby
Ruby	github.com/kontena/k8s-client
Rust	github.com/kube-rs/kube
Rust	github.com/ynqa/kubernetes-rust
Scala	github.com/hagay3/skuber
Scala	github.com/hnaderi/scala-k8s
Scala	github.com/joan38/kubernetes-client
Swift	github.com/swiftkube/client

Common Expression Language in Kubernetes

The [Common Expression Language \(CEL\)](#) is used in the Kubernetes API to declare validation rules, policy rules, and other constraints or conditions.

CEL expressions are evaluated directly in the [API server](#), making CEL a convenient alternative to out-of-process mechanisms, such as webhooks, for many extensibility use cases. Your CEL expressions continue to execute so long as the control plane's API server component remains available.

Language overview

The [CEL language](#) has a straightforward syntax that is similar to the expressions in C, C++, Java, JavaScript and Go.

CEL was designed to be embedded into applications. Each CEL "program" is a single expression that evaluates to a single value. CEL expressions are typically short "one-liners" that inline well into the string fields of Kubernetes API resources.

Inputs to a CEL program are "variables". Each Kubernetes API field that contains CEL declares in the API documentation which variables are available to use for that field. For example, in the `x-kubernetes-validations[i].rules` field of CustomResourceDefinitions, the `self` and `oldSelf` variables are available and refer to the previous and current state of the custom resource data to be validated by the CEL expression. Other Kubernetes API fields may declare different variables. See the API documentation of the API fields to learn which variables are available for that field.

Example CEL expressions:

Examples of CEL expressions and the purpose of each

Rule	Purpose
<code>self.minReplicas <= self.replicas && self.replicas <= self.maxReplicas</code>	Validate that the three fields defining replicas are ordered appropriately
'Available' in <code>self.stateCounts</code>	Validate that an entry with the 'Available' key exists in a map
<code>(self.list1.size() == 0) != (self.list2.size() == 0)</code>	Validate that one of two lists is non-empty, but not both
<code>self.envars.filter(e, e.name = 'MY_ENV').all(e, e.value.matches('^[a-zA-Z]*\$'))</code>	Validate the 'value' field of a listMap entry where key field 'name' is 'MY_ENV'
<code>has(self.expired) && self.created + self.ttl < self.expired</code>	Validate that 'expired' date is after a 'create' date plus a 'ttl' duration
<code>self.health.startsWith('ok')</code>	Validate a 'health' string field has the prefix 'ok'
<code>self.widgets.exists(w, w.key == 'x' && w.foo < 10)</code>	Validate that the 'foo' property of a listMap item with a key 'x' is less than 10
<code>type(self) == string ? self == '99%' : self == 42</code>	Validate an int-or-string field for both the int and string cases
<code>self.metadata.name == 'singleton'</code>	Validate that an object's name matches a specific value (making it a singleton)
<code>self.set1.all(e, !(e in self.set2))</code>	Validate that two listSets are disjoint
<code>self.names.size() == self.details.size() && self.names.all(n, n in self.details)</code>	Validate the 'details' map is keyed by the items in the 'names' listSet

CEL community libraries

Kubernetes CEL expressions have access to the following CEL community libraries:

- CEL standard functions, defined in the [list of standard definitions](#)
- CEL standard [macros](#)
- CEL [extended string function library](#)

Kubernetes CEL libraries

In addition to the CEL community libraries, Kubernetes includes CEL libraries that are available everywhere CEL is used in Kubernetes.

Kubernetes list library

The list library includes `indexOf` and `lastIndexOf`, which work similar to the `strings` functions of the same names. These functions either the first or last positional index of the provided element in the list.

The list library also includes `min`, `max` and `sum`. `Sum` is supported on all number types as well as the duration type. `Min` and `max` are supported on all comparable types.

`isSorted` is also provided as a convenience function and is supported on all comparable types.

Examples:

Examples of CEL expressions using list library functions

CEL Expression	Purpose
<code>names.isSorted()</code>	Verify that a list of names is kept in alphabetical order
<code>items.map(x, x.weight).sum() == 1.0</code>	Verify that the "weights" of a list of objects sum to 1.0
<code>lowPriorities.map(x, x.priority).max() < highPriorities.map(x, x.priority).min()</code>	Verify that two sets of priorities do not overlap
<code>names.indexOf('should-be-first') == 1</code>	Require that the first name in a list is a specific value

See the [Kubernetes List Library](#) godoc for more information.

Kubernetes regex library

In addition to the `matches` function provided by the CEL standard library, the regex library provides `find` and `findAll`, enabling a much wider range of regex operations.

Examples:

Examples of CEL expressions using regex library functions

CEL Expression	Purpose
<code>"abc 123".find('[0-9]*')</code>	Find the first number in a string
<code>"1, 2, 3, 4".findAll('[0-9]*').map(x, int(x)).sum() < 100</code>	Verify that the numbers in a string sum to less than 100

See the [Kubernetes regex library](#) godoc for more information.

Kubernetes URL library

To make it easier and safer to process URLs, the following functions have been added:

- `isURL(string)` checks if a string is a valid URL according to the [Go's net/url package](#). The string must be an absolute URL.
- `url(string)` URL converts a string to a URL or results in an error if the string is not a valid URL.

Once parsed via the `url` function, the resulting URL object has `getScheme`, `getHost`, `getHostname`, `getPort`, `getEscapedPath` and `getQuery` accessors.

Examples:

Examples of CEL expressions using URL library functions

CEL Expression	Purpose
<code>url('https://example.com:80/').getHost()</code>	Get the 'example.com:80' host part of the URL.
<code>url('https://example.com/path with spaces/').getEscapedPath()</code>	Returns '/path%20with%20spaces/'

See the [Kubernetes URL library](#) godoc for more information.

Kubernetes authorizer library

For CEL expressions in the API where a variable of type `Authorizer` is available, the authorizer may be used to perform authorization checks for the principal (authenticated user) of the request.

API resource checks are performed as follows:

1. Specify the group and resource to check: `Authorizer.group(string).resource(string) ResourceCheck`
2. Optionally call any combination of the following builder functions to further narrow the authorization check. Note that these functions return the receiver type and can be chained:
 - `ResourceCheck.subresource(string) ResourceCheck`
 - `ResourceCheck.namespace(string) ResourceCheck`
 - `ResourceCheck.name(string) ResourceCheck`
1. Call `ResourceCheck.check(verb string)` Decision to perform the authorization check.
2. Call `allowed() bool` or `reason() string` to inspect the result of the authorization check.

Non-resource authorization performed are used as follows:

1. specify only a path: `Authorizer.path(string) PathCheck`
2. Call `PathCheck.check(httpVerb string)` Decision to perform the authorization check.
3. Call `allowed() bool` or `reason() string` to inspect the result of the authorization check.

To perform an authorization check for a service account:

- `Authorizer.serviceAccount(namespace string, name string) Authorizer`

Examples of CEL expressions using URL library functions

CEL Expression	Purpose
<code>authorizer.group('').resource('pods').namespace('default').check('create').allowed()</code>	Returns true if the principal (user or service account) is allowed to create pods in the 'default' namespace.
<code>authorizer.path('/healthz').check('get').allowed()</code>	Checks if the principal (user or service account) is authorized to make HTTP GET requests to the / healthz API path.
<code>authorizer.serviceAccount('default', 'myserviceaccount').resource('deployments').check('delete').allowed()</code>	Checks if the service account is authorized to delete deployments.

See the [Kubernetes Authz library](#) godoc for more information.

Type checking

CEL is a [gradually typed language](#).

Some Kubernetes API fields contain fully type checked CEL expressions. For example, [CustomResourceDefinitions Validation Rules](#) are fully type checked.

Some Kubernetes API fields contain partially type checked CEL expressions. A partially type checked expression is an expression where some of the variables are statically typed but others are dynamically typed. For example, in the CEL expressions of [ValidatingAdmissionPolicies](#) the request variable is typed, but the object variable is dynamically typed. As a result, an expression containing `request.namex` would fail type checking because the namex field is not defined. However, `object.namex` would pass type checking even when the namex field is not defined for the resource kinds that object refers to, because object is dynamically typed.

The has() macro in CEL may be used in CEL expressions to check if a field of a dynamically typed variable is accessible before attempting to access the field's value. For example:

```
has(object.namex) ? object.namex == 'special' : request.name == 'special'
```

Type system integration

Table showing the relationship between OpenAPIv3 types and CEL types

OpenAPIv3 type	CEL type
'object' with Properties	object / "message type" (type(<object>) evaluates to selfType<uniqueNumber>.path.to.object.from.self)
'object' with AdditionalProperties	map
'object' with x-kubernetes-embedded-type	object / "message type", 'apiVersion', 'kind', 'metadata.name' and 'metadata.generateName' are implicitly included in schema
'object' with x-kubernetes-preserve-unknown-fields	object / "message type", unknown fields are NOT accessible in CEL expression
x-kubernetes-int-or-string	union of int or string, self.intOrString < 100 self.intOrString == '50%' evaluates to true for both 50 and "50%"
'array'	list
'array' with x-kubernetes-list-type=map	list with map based Equality & unique key guarantees
'array' with x-kubernetes-list-type=set	list with set based Equality & unique entry guarantees
'boolean'	boolean
'number' (all formats)	double
'integer' (all formats)	int (64)
<i>no equivalent</i>	uint (64)
'null'	null_type
'string'	string
'string' with format=byte (base64 encoded)	bytes
'string' with format=date	timestamp (google.protobuf.Timestamp)
'string' with format=datetime	timestamp (google.protobuf.Timestamp)
'string' with format=duration	duration (google.protobuf.Duration)

Also see: [CEL types](#), [OpenAPI types](#), [Kubernetes Structural Schemas](#).

Equality comparison for arrays with x-kubernetes-list-type of set or map ignores element order. For example [1, 2] == [2, 1] if the arrays represent Kubernetes set values.

Concatenation on arrays with x-kubernetes-list-type use the semantics of the list type:

- set: X + Y performs a union where the array positions of all elements in X are preserved and non-intersecting elements in Y are appended, retaining their partial order.
- map: X + Y performs a merge where the array positions of all keys in X are preserved but the values are overwritten by values in Y when the key sets of X and Y intersect. Elements in Y with non-intersecting keys are appended, retaining their partial order.

Escaping

Only Kubernetes resource property names of the form `[a-zA-Z_-/][a-zA-Z0-9_-/]*` are accessible from CEL. Accessible property names are escaped according to the following rules when accessed in the expression:

Table of CEL identifier escaping rules

escape sequence	property name equivalent
<code>_underscores_</code>	<code>_</code>
<code>_dot_</code>	<code>.</code>
<code>_dash_</code>	<code>-</code>
<code>_slash_</code>	<code>/</code>
<code>{keyword}</code>	CEL RESERVED keyword

When you escape any of CEL's **RESERVED** keywords you need to match the exact property name use the underscore escaping (for example, `int` in the word `sprint` would not be escaped and nor would it need to be).

Examples on escaping:

Examples escaped CEL identifiers

property name	rule with escaped property name
<code>namespace</code>	<code>self._ namespace_ > 0</code>
<code>x-prop</code>	<code>self.x dash_ prop > 0</code>
<code>redact_d</code>	<code>self.redact_underscores_d > 0</code>
<code>string</code>	<code>self.startsWith('kube')</code>

Resource constraints

CEL is non-Turing complete and offers a variety of production safety controls to limit execution time. CEL's *resource constraint* features provide feedback to developers about expression complexity and help protect the API server from excessive resource consumption during evaluation. CEL's resource constraint features are used to prevent CEL evaluation from consuming excessive API server resources.

A key element of the resource constraint features is a *cost unit* that CEL defines as a way of tracking CPU utilization. Cost units are independent of system load and hardware. Cost units are also deterministic; for any given CEL expression and input data, evaluation of the expression by the CEL interpreter will always result in the same cost.

Many of CEL's core operations have fixed costs. The simplest operations, such as comparisons (e.g. `<`) have a cost of 1. Some have a higher fixed cost, for example list literal declarations have a fixed base cost of 40 cost units.

Calls to functions implemented in native code approximate cost based on the time complexity of the operation. For example: operations that use regular expressions, such as `match` and `find`, are estimated using an approximated cost of $\text{length}(\text{regexString}) * \text{length}(\text{inputString})$. The approximated cost reflects the worst case time complexity of Go's RE2 implementation.

Runtime cost budget

All CEL expressions evaluated by Kubernetes are constrained by a runtime cost budget. The runtime cost budget is an estimate of actual CPU utilization computed by incrementing a cost unit counter while interpreting a CEL expression. If the CEL interpreter executes too many instructions, the runtime cost budget will be exceeded, execution of the expressions will be halted, and an error will result.

Some Kubernetes resources define an additional runtime cost budget that bounds the execution of multiple expressions. If the sum total of the cost of expressions exceed the budget, execution of the expressions will be halted, and an error will result. For example the validation of a custom resource has a *per-validation* runtime cost budget for all [Validation Rules](#) evaluated to validate the custom resource.

Estimated cost limits

For some Kubernetes resources, the API server may also check if worst case estimated running time of CEL expressions would be prohibitively expensive to execute. If so, the API server prevent the CEL expression from being written to API resources by rejecting create or update operations containing the CEL expression to the API resources. This feature offers a stronger assurance that CEL expressions written to the API resource will be evaluate at runtime without exceeding the runtime cost budget.

Kubernetes Deprecation Policy

This document details the deprecation policy for various facets of the system.

Kubernetes is a large system with many components and many contributors. As with any such software, the feature set naturally evolves over time, and sometimes a feature may need to be removed. This could include an API, a flag, or even an entire feature. To avoid breaking existing users, Kubernetes follows a deprecation policy for aspects of the system that are slated to be removed.

Deprecating parts of the API

Since Kubernetes is an API-driven system, the API has evolved over time to reflect the evolving understanding of the problem space. The Kubernetes API is actually a set of APIs, called "API groups", and each API group is independently versioned. [API versions](#) fall into 3 main tracks, each of which has different policies for deprecation:

Example	Track
v1	GA (generally available, stable)
v1beta1	Beta (pre-release)
v1alpha1	Alpha (experimental)

A given release of Kubernetes can support any number of API groups and any number of versions of each.

The following rules govern the deprecation of elements of the API. This includes:

- REST resources (aka API objects)
- Fields of REST resources
- Annotations on REST resources, including "beta" annotations but not including "alpha" annotations.
- Enumerated or constant values
- Component config structures

These rules are enforced between official releases, not between arbitrary commits to master or release branches.

Rule #1: API elements may only be removed by incrementing the version of the API group.

Once an API element has been added to an API group at a particular version, it can not be removed from that version or have its behavior significantly changed, regardless of track.

Note: For historical reasons, there are 2 "monolithic" API groups - "core" (no group name) and "extensions". Resources will incrementally be moved from these legacy API groups into more domain-specific API groups.

Rule #2: API objects must be able to round-trip between API versions in a given release without information loss, with the exception of whole REST resources that do not exist in some versions.

For example, an object can be written as v1 and then read back as v2 and converted to v1, and the resulting v1 resource will be identical to the original. The representation in v2 might be different from v1, but the system knows how to convert between them in both directions. Additionally, any new field added in v2 must be able to round-trip to v1 and back, which means v1 might have to add an equivalent field or represent it as an annotation.

Rule #3: An API version in a given track may not be deprecated in favor of a less stable API version.

- GA API versions can replace beta and alpha API versions.
- Beta API versions can replace earlier beta and alpha API versions, but *may not* replace GA API versions.
- Alpha API versions can replace earlier alpha API versions, but *may not* replace GA or beta API versions.

Rule #4a: API lifetime is determined by the API stability level

- GA API versions may be marked as deprecated, but must not be removed within a major version of Kubernetes
- Beta API versions are deprecated no more than 9 months or 3 minor releases after introduction (whichever is longer), and are no longer served 9 months or 3 minor releases after deprecation (whichever is longer)
- Alpha API versions may be removed in any release without prior deprecation notice

This ensures beta API support covers the [maximum supported version skew of 2 releases](#), and that APIs don't stagnate on unstable beta versions, accumulating production usage that will be disrupted when support for the beta API ends.

Note: There are no current plans for a major version revision of Kubernetes that removes GA APIs.

Note: Until [#52185](#) is resolved, no API versions that have been persisted to storage may be removed. Serving REST endpoints for those versions may be disabled (subject to the deprecation timelines in this document), but the API server must remain capable of decoding/converting previously persisted data from storage.

Rule #4b: The "preferred" API version and the "storage version" for a given group may not advance until after a release has been made that supports both the new version and the previous version

Users must be able to upgrade to a new release of Kubernetes and then roll back to a previous release, without converting anything to the new API version or suffering breakages (unless they explicitly used features only available in the newer version). This is particularly evident in the stored representation of objects.

All of this is best illustrated by examples. Imagine a Kubernetes release, version X, which introduces a new API group. A new Kubernetes release is made every approximately 4 months (3 per year). The following table describes which API versions are supported in a series of subsequent releases.

Release	API Versions	Preferred/ Storage Version	Notes
X	v1alpha1	v1alpha1	
X+1	v1alpha2	v1alpha2	<ul style="list-style-type: none">v1alpha1 is removed, "action required" relnote
X+2	v1beta1	v1beta1	<ul style="list-style-type: none">v1alpha2 is removed, "action required" relnote
X+3	v1beta2, v1beta1 (deprecated)	v1beta1	<ul style="list-style-type: none">v1beta1 is deprecated, "action required" relnote
X+4	v1beta2, v1beta1 (deprecated)	v1beta2	
X+5	v1, v1beta1 (deprecated), v1beta2 (deprecated)	v1beta2	<ul style="list-style-type: none">v1beta2 is deprecated, "action required" relnote
X+6	v1, v1beta2 (deprecated)	v1	<ul style="list-style-type: none">v1beta1 is removed, "action required" relnote
X+7	v1, v1beta2 (deprecated)	v1	
X+8	v2alpha1, v1	v1	<ul style="list-style-type: none">v1beta2 is removed, "action required" relnote

Release	API Versions	Preferred/ Storage Version	Notes
X+9	v2alpha2, v1	v1	<ul style="list-style-type: none"> v2alpha1 is removed, "action required" relnote
X+10	v2beta1, v1	v1	<ul style="list-style-type: none"> v2alpha2 is removed, "action required" relnote
X+11	v2beta2, v2beta1 (deprecated), v1	v1	<ul style="list-style-type: none"> v2beta1 is deprecated, "action required" relnote
X+12	v2, v2beta2 (deprecated), v2beta1 (deprecated), v1 (deprecated)	v1	<ul style="list-style-type: none"> v2beta2 is deprecated, "action required" relnote v1 is deprecated in favor of v2, but will not be removed
X+13	v2, v2beta1 (deprecated), v2beta2 (deprecated), v1 (deprecated)	v2	
X+14	v2, v2beta2 (deprecated), v1 (deprecated)	v2	<ul style="list-style-type: none"> v2beta1 is removed, "action required" relnote
X+15	v2, v1 (deprecated)	v2	<ul style="list-style-type: none"> v2beta2 is removed, "action required" relnote

REST resources (aka API objects)

Consider a hypothetical REST resource named Widget, which was present in API v1 in the above timeline, and which needs to be deprecated. We document and [announce](#) the deprecation in sync with release X+1. The Widget resource still exists in API version v1 (deprecated) but not in v2alpha1. The Widget resource continues to exist and function in releases up to and including X+8. Only in release X+9, when API v1 has aged out, does the Widget resource cease to exist, and the behavior get removed.

Starting in Kubernetes v1.19, making an API request to a deprecated REST API endpoint:

1. Returns a Warning header (as defined in [RFC7234, Section 5.5](#)) in the API response.
2. Adds a "k8s.io/deprecated":"true" annotation to the [audit event](#) recorded for the request.
3. Sets an apiserver_requested_deprecated_apis gauge metric to 1 in the kube-apiserver process. The metric has labels for group, version, resource, subresource that can be joined to the apiserver_request_total metric, and a removed_release label that indicates the Kubernetes release in which the API will no longer be served. The following Prometheus query returns information about requests made to deprecated APIs which will be removed in v1.22:

```
apiserver_requested_DEPRECATED_apis{removed_release="1.22"} * on(group,version,resource,subresource) group_right() apiserver_request_total
```

Fields of REST resources

As with whole REST resources, an individual field which was present in API v1 must exist and function until API v1 is removed. Unlike whole resources, the v2 APIs may choose a different representation for the field, as long as it can be round-tripped. For example a v1 field named "magnitude" which was deprecated might be named "deprecatedMagnitude" in API v2. When v1 is eventually removed, the deprecated field can be removed from v2.

Enumerated or constant values

As with whole REST resources and fields thereof, a constant value which was supported in API v1 must exist and function until API v1 is removed.

Component config structures

Component configs are versioned and managed similar to REST resources.

Future work

Over time, Kubernetes will introduce more fine-grained API versions, at which point these rules will be adjusted as needed.

Deprecating a flag or CLI

The Kubernetes system is comprised of several different programs cooperating. Sometimes, a Kubernetes release might remove flags or CLI commands (collectively "CLI elements") in these programs. The individual programs naturally sort into two main groups - user-facing and admin-facing programs, which vary slightly in their deprecation policies. Unless a flag is explicitly prefixed or documented as "alpha" or "beta", it is considered GA.

CLI elements are effectively part of the API to the system, but since they are not versioned in the same way as the REST API, the rules for deprecation are as follows:

Rule #5a: CLI elements of user-facing components (e.g. kubectl) must function after their announced deprecation for no less than:

- **GA: 12 months or 2 releases (whichever is longer)**
- **Beta: 3 months or 1 release (whichever is longer)**
- **Alpha: 0 releases**

Rule #5b: CLI elements of admin-facing components (e.g. kubelet) must function after their announced deprecation for no less than:

- **GA: 6 months or 1 release (whichever is longer)**
- **Beta: 3 months or 1 release (whichever is longer)**
- **Alpha: 0 releases**

Rule #6: Deprecated CLI elements must emit warnings (optionally disable) when used.

Deprecating a feature or behavior

Occasionally a Kubernetes release needs to deprecate some feature or behavior of the system that is not controlled by the API or CLI. In this case, the rules for deprecation are as follows:

Rule #7: Deprecated behaviors must function for no less than 1 year after their announced deprecation.

This does not imply that all changes to the system are governed by this policy. This applies only to significant, user-visible behaviors which impact the correctness of applications running on Kubernetes or that impact the administration of Kubernetes clusters, and which are being removed entirely.

An exception to the above rule is *feature gates*. Feature gates are key=value pairs that allow for users to enable/disable experimental features.

Feature gates are intended to cover the development life cycle of a feature - they are not intended to be long-term APIs. As such, they are expected to be deprecated and removed after a feature becomes GA or is dropped.

As a feature moves through the stages, the associated feature gate evolves. The feature life cycle matched to its corresponding feature gate is:

- Alpha: the feature gate is disabled by default and can be enabled by the user.
- Beta: the feature gate is enabled by default and can be disabled by the user.
- GA: the feature gate is deprecated (see "[Deprecation](#)") and becomes non-operational.
- GA, deprecation window complete: the feature gate is removed and calls to it are no longer accepted.

Deprecation

Features can be removed at any point in the life cycle prior to GA. When features are removed prior to GA, their associated feature gates are also deprecated.

When an invocation tries to disable a non-operational feature gate, the call fails in order to avoid unsupported scenarios that might otherwise run silently.

In some cases, removing pre-GA features requires considerable time. Feature gates can remain operational until their associated feature is fully removed, at which point the feature gate itself can be deprecated.

When removing a feature gate for a GA feature also requires considerable time, calls to feature gates may remain operational if the feature gate has no effect on the feature, and if the feature gate causes no errors.

Features intended to be disabled by users should include a mechanism for disabling the feature in the associated feature gate.

Versioning for feature gates is different from the previously discussed components, therefore the rules for deprecation are as follows:

Rule #8: Feature gates must be deprecated when the corresponding feature they control transitions a lifecycle stage as follows. Feature gates must function for no less than:

- Beta feature to GA: 6 months or 2 releases (whichever is longer)
- Beta feature to EOL: 3 months or 1 release (whichever is longer)
- Alpha feature to EOL: 0 releases

Rule #9: Deprecated feature gates must respond with a warning when used. When a feature gate is deprecated it must be documented in both in the release notes and the corresponding CLI help. Both warnings and documentation must indicate whether a feature gate is non-operational.

Deprecating a metric

Each component of the Kubernetes control-plane exposes metrics (usually the /metrics endpoint), which are typically ingested by cluster administrators. Not all metrics are the same: some metrics are commonly used as SLIs or used to determine SLOs, these tend to have greater import. Other metrics are more experimental in nature or are used primarily in the Kubernetes development process.

Accordingly, metrics fall under three stability classes (ALPHA, BETA STABLE); this impacts removal of a metric during a Kubernetes release. These classes are determined by the perceived importance of the metric. The rules for deprecating and removing a metric are as follows:

Rule #9a: Metrics, for the corresponding stability class, must function for no less than:

- STABLE: 4 releases or 12 months (whichever is longer)
- BETA: 2 releases or 8 months (whichever is longer)
- ALPHA: 0 releases

Rule #9b: Metrics, after their *announced deprecation*, must function for no less than:

- STABLE: 3 releases or 9 months (whichever is longer)
- BETA: 1 releases or 4 months (whichever is longer)
- ALPHA: 0 releases

Deprecated metrics will have their description text prefixed with a deprecation notice string '(Deprecated from x.y)' and a warning log will be emitted during metric registration. Like their stable undeprecated counterparts, deprecated metrics will be automatically registered to the metrics endpoint and therefore visible.

On a subsequent release (when the metric's deprecatedVersion is equal to `current_kubernetes_version - 3`), a deprecated metric will become a *hidden* metric. *Unlike* their deprecated counterparts, hidden metrics will *no longer* be automatically registered to the metrics endpoint (hence hidden). However, they can be explicitly enabled through a command line flag on the binary (`--show-hidden-metrics-for-version=`). This provides cluster admins an escape hatch to properly migrate off of a deprecated metric, if they were not able to react to the earlier deprecation warnings. Hidden metrics should be deleted after one release.

Exceptions

No policy can cover every possible situation. This policy is a living document, and will evolve over time. In practice, there will be situations that do not fit neatly into this policy, or for which this policy becomes a serious impediment. Such situations should be discussed with SIGs and project leaders to find the best solutions for those specific cases, always bearing in mind that Kubernetes is committed to being a stable system that, as much as possible, never breaks users. Exceptions will always be announced in all relevant release notes.

Deprecated API Migration Guide

As the Kubernetes API evolves, APIs are periodically reorganized or upgraded. When APIs evolve, the old API is deprecated and eventually removed. This page contains information you need to know when migrating from deprecated API versions to newer and more stable API versions.

Removed APIs by release

v1.29

The v1.29 release will stop serving the following deprecated API versions:

Flow control resources

The `flowcontrol.apiserver.k8s.io/v1beta2` API version of FlowSchema and PriorityLevelConfiguration will no longer be served in v1.29.

- Migrate manifests and API clients to use the `flowcontrol.apiserver.k8s.io/v1beta3` API version, available since v1.26.
- All existing persisted objects are accessible via the new API
- Notable changes in `flowcontrol.apiserver.k8s.io/v1beta3`:
 - The PriorityLevelConfiguration spec.limited.assuredConcurrencyShares field is renamed to spec.limited.nominalConcurrencyShares

v1.27

The v1.27 release stopped serving the following deprecated API versions:

CSIStrageCapacity

The `storage.k8s.io/v1beta1` API version of CSIStrageCapacity will no longer be served in v1.27.

- Migrate manifests and API clients to use the `storage.k8s.io/v1` API version, available since v1.24.
- All existing persisted objects are accessible via the new API
- No notable changes

v1.26

The **v1.26** release stopped serving the following deprecated API versions:

Flow control resources

The **flowcontrol.apiserver.k8s.io/v1beta1** API version of FlowSchema and PriorityLevelConfiguration is no longer served as of v1.26.

- Migrate manifests and API clients to use the **flowcontrol.apiserver.k8s.io/v1beta3** API version, available since v1.26.
- All existing persisted objects are accessible via the new API
- No notable changes

HorizontalPodAutoscaler

The **autoscaling/v2beta2** API version of HorizontalPodAutoscaler is no longer served as of v1.26.

- Migrate manifests and API clients to use the **autoscaling/v2** API version, available since v1.23.
- All existing persisted objects are accessible via the new API

v1.25

The **v1.25** release stopped serving the following deprecated API versions:

CronJob

The **batch/v1beta1** API version of CronJob is no longer served as of v1.25.

- Migrate manifests and API clients to use the **batch/v1** API version, available since v1.21.
- All existing persisted objects are accessible via the new API
- No notable changes

EndpointSlice

The **discovery.k8s.io/v1beta1** API version of EndpointSlice is no longer served as of v1.25.

- Migrate manifests and API clients to use the **discovery.k8s.io/v1** API version, available since v1.21.
- All existing persisted objects are accessible via the new API
- Notable changes in **discovery.k8s.io/v1**:
 - use per Endpoint nodeName field instead of deprecated topology["kubernetes.io/hostname"] field
 - use per Endpoint zone field instead of deprecated topology["topology.kubernetes.io/zone"] field
 - topology is replaced with the deprecatedTopology field which is not writable in v1

Event

The **events.k8s.io/v1beta1** API version of Event is no longer served as of v1.25.

- Migrate manifests and API clients to use the **events.k8s.io/v1** API version, available since v1.19.
- All existing persisted objects are accessible via the new API
- Notable changes in **events.k8s.io/v1**:
 - type is limited to Normal and Warning
 - involvedObject is renamed to regarding
 - action, reason, reportingController, and reportingInstance are required when creating new **events.k8s.io/v1** Events
 - use eventTime instead of the deprecated firstTimestamp field (which is renamed to deprecatedFirstTimestamp and not permitted in new **events.k8s.io/v1** Events)
 - use series.lastObservedTime instead of the deprecated lastTimestamp field (which is renamed to deprecatedLastTimestamp and not permitted in new **events.k8s.io/v1** Events)
 - use series.count instead of the deprecated count field (which is renamed to deprecatedCount and not permitted in new **events.k8s.io/v1** Events)
 - use reportingController instead of the deprecated source.component field (which is renamed to deprecatedSource.component and not permitted in new **events.k8s.io/v1** Events)
 - use reportingInstance instead of the deprecated source.host field (which is renamed to deprecatedSource.host and not permitted in new **events.k8s.io/v1** Events)

HorizontalPodAutoscaler

The **autoscaling/v2beta1** API version of HorizontalPodAutoscaler is no longer served as of v1.25.

- Migrate manifests and API clients to use the **autoscaling/v2** API version, available since v1.23.
- All existing persisted objects are accessible via the new API

PodDisruptionBudget

The **policy/v1beta1** API version of PodDisruptionBudget is no longer served as of v1.25.

- Migrate manifests and API clients to use the **policy/v1** API version, available since v1.21.
- All existing persisted objects are accessible via the new API
- Notable changes in **policy/v1**:
 - an empty spec.selector ({} written to a policy/v1 PodDisruptionBudget selects all pods in the namespace (in policy/v1beta1 an empty spec.selector selected no pods). An unset spec.selector selects no pods in either API version).

PodSecurityPolicy

PodSecurityPolicy in the **policy/v1beta1** API version is no longer served as of v1.25, and the PodSecurityPolicy admission controller will be removed.

Migrate to [Pod Security Admission](#) or a [3rd party admission webhook](#). For a migration guide, see [Migrate from PodSecurityPolicy to the Built-In PodSecurity Admission Controller](#). For more information on the deprecation, see [PodSecurityPolicy Deprecation: Past, Present, and Future](#).

RuntimeClass

RuntimeClass in the **node.k8s.io/v1beta1** API version is no longer served as of v1.25.

- Migrate manifests and API clients to use the **node.k8s.io/v1** API version, available since v1.20.
- All existing persisted objects are accessible via the new API
- No notable changes

v1.22

The **v1.22** release stopped serving the following deprecated API versions:

Webhook resources

The **admissionregistration.k8s.io/v1beta1** API version of MutatingWebhookConfiguration and ValidatingWebhookConfiguration is no longer served as of v1.22.

- Migrate manifests and API clients to use the **admissionregistration.k8s.io/v1** API version, available since v1.16.
- All existing persisted objects are accessible via the new APIs
- Notable changes:
 - webhooks[*].failurePolicy default changed from Ignore to Fail for v1
 - webhooks[*].matchPolicy default changed from Exact to Equivalent for v1
 - webhooks[*].timeoutSeconds default changed from 30s to 10s for v1
 - webhooks[*].sideEffects default value is removed, and the field made required, and only None and NoneOnDryRun are permitted for v1
 - webhooks[*].admissionReviewVersions default value is removed and the field made required for v1 (supported versions for AdmissionReview are v1 and v1beta1)
 - webhooks[*].name must be unique in the list for objects created via admissionregistration.k8s.io/v1

CustomResourceDefinition

The **apiextensions.k8s.io/v1beta1** API version of CustomResourceDefinition is no longer served as of v1.22.

- Migrate manifests and API clients to use the **apiextensions.k8s.io/v1** API version, available since v1.16.
- All existing persisted objects are accessible via the new API
- Notable changes:
 - spec.scope is no longer defaulted to Namespaced and must be explicitly specified
 - spec.version is removed in v1; use spec.versions instead
 - spec.validation is removed in v1; use spec.versions[*].schema instead
 - spec.subresources is removed in v1; use spec.versions[*].subresources instead
 - spec.additionalPrinterColumns is removed in v1; use spec.versions[*].additionalPrinterColumns instead

- spec.conversion.webhookClientConfig is moved to spec.conversion.webhook.clientConfig in v1
- spec.conversion.conversionReviewVersions is moved to spec.conversion.webhook.conversionReviewVersions in v1
- spec.versions[*].schema.openAPIV3Schema is now required when creating v1 CustomResourceDefinition objects, and must be a [structural schema](#)
- spec.preserveUnknownFields: true is disallowed when creating v1 CustomResourceDefinition objects; it must be specified within schema definitions as x-kubernetes-preserve-unknown-fields: true
- In additionalPrinterColumns items, the JSONPath field was renamed to jsonPath in v1 (fixes [#66531](#))

APIService

The **apiregistration.k8s.io/v1beta1** API version of APIService is no longer served as of v1.22.

- Migrate manifests and API clients to use the **apiregistration.k8s.io/v1** API version, available since v1.10.
- All existing persisted objects are accessible via the new API
- No notable changes

TokenReview

The **authentication.k8s.io/v1beta1** API version of TokenReview is no longer served as of v1.22.

- Migrate manifests and API clients to use the **authentication.k8s.io/v1** API version, available since v1.6.
- No notable changes

SubjectAccessReview resources

The **authorization.k8s.io/v1beta1** API version of LocalSubjectAccessReview, SelfSubjectAccessReview, SubjectAccessReview, and SelfSubjectRulesReview is no longer served as of v1.22.

- Migrate manifests and API clients to use the **authorization.k8s.io/v1** API version, available since v1.6.
- Notable changes:
 - spec.group was renamed to spec.groups in v1 (fixes [#32709](#))

CertificateSigningRequest

The **certificates.k8s.io/v1beta1** API version of CertificateSigningRequest is no longer served as of v1.22.

- Migrate manifests and API clients to use the **certificates.k8s.io/v1** API version, available since v1.19.
- All existing persisted objects are accessible via the new API

- Notable changes in certificates.k8s.io/v1:
 - For API clients requesting certificates:
 - spec.signerName is now required (see [known Kubernetes signers](#)), and requests for kubernetes.io/legacy-unknown are not allowed to be created via the certificates.k8s.io/v1 API
 - spec.usages is now required, may not contain duplicate values, and must only contain known usages
 - For API clients approving or signing certificates:
 - status.conditions may not contain duplicate types
 - status.conditions[*].status is now required
 - status.certificate must be PEM-encoded, and contain only CERTIFICATE blocks

Lease

The **coordination.k8s.io/v1beta1** API version of Lease is no longer served as of v1.22.

- Migrate manifests and API clients to use the **coordination.k8s.io/v1** API version, available since v1.14.
- All existing persisted objects are accessible via the new API
- No notable changes

Ingress

The **extensions/v1beta1** and **networking.k8s.io/v1beta1** API versions of Ingress is no longer served as of v1.22.

- Migrate manifests and API clients to use the **networking.k8s.io/v1** API version, available since v1.19.
- All existing persisted objects are accessible via the new API
- Notable changes:
 - spec.backend is renamed to spec.defaultBackend
 - The backend serviceName field is renamed to service.name
 - Numeric backend servicePort fields are renamed to service.port.number
 - String backend servicePort fields are renamed to service.port.name
 - pathType is now required for each specified path. Options are Prefix, Exact, and ImplementationSpecific. To match the undefined v1beta1 behavior, use ImplementationSpecific.

IngressClass

The **networking.k8s.io/v1beta1** API version of IngressClass is no longer served as of v1.22.

- Migrate manifests and API clients to use the **networking.k8s.io/v1** API version, available since v1.19.
- All existing persisted objects are accessible via the new API
- No notable changes

RBAC resources

The **rbac.authorization.k8s.io/v1beta1** API version of ClusterRole, ClusterRoleBinding, Role, and RoleBinding is no longer served as of v1.22.

- Migrate manifests and API clients to use the **rbac.authorization.k8s.io/v1** API version, available since v1.8.
- All existing persisted objects are accessible via the new APIs
- No notable changes

PriorityClass

The **scheduling.k8s.io/v1beta1** API version of PriorityClass is no longer served as of v1.22.

- Migrate manifests and API clients to use the **scheduling.k8s.io/v1** API version, available since v1.14.
- All existing persisted objects are accessible via the new API
- No notable changes

Storage resources

The **storage.k8s.io/v1beta1** API version of CSIDriver, CSINode, StorageClass, and VolumeAttachment is no longer served as of v1.22.

- Migrate manifests and API clients to use the **storage.k8s.io/v1** API version
 - CSIDriver is available in **storage.k8s.io/v1** since v1.19.
 - CSINode is available in **storage.k8s.io/v1** since v1.17
 - StorageClass is available in **storage.k8s.io/v1** since v1.6
 - VolumeAttachment is available in **storage.k8s.io/v1** v1.13
- All existing persisted objects are accessible via the new APIs
- No notable changes

v1.16

The **v1.16** release stopped serving the following deprecated API versions:

NetworkPolicy

The **extensions/v1beta1** API version of NetworkPolicy is no longer served as of v1.16.

- Migrate manifests and API clients to use the **networking.k8s.io/v1** API version, available since v1.8.
- All existing persisted objects are accessible via the new API

DaemonSet

The **extensions/v1beta1** and **apps/v1beta2** API versions of DaemonSet are no longer served as of v1.16.

- Migrate manifests and API clients to use the **apps/v1** API version, available since v1.9.
- All existing persisted objects are accessible via the new API

- Notable changes:
 - spec.templateGeneration is removed
 - spec.selector is now required and immutable after creation; use the existing template labels as the selector for seamless upgrades
 - spec.updateStrategy.type now defaults to RollingUpdate (the default in extensions/v1beta1 was OnDelete)

Deployment

The **extensions/v1beta1**, **apps/v1beta1**, and **apps/v1beta2** API versions of Deployment are no longer served as of v1.16.

- Migrate manifests and API clients to use the **apps/v1** API version, available since v1.9.
- All existing persisted objects are accessible via the new API
- Notable changes:
 - spec.rollbackTo is removed
 - spec.selector is now required and immutable after creation; use the existing template labels as the selector for seamless upgrades
 - spec.progressDeadlineSeconds now defaults to 600 seconds (the default in extensions/v1beta1 was no deadline)
 - spec.revisionHistoryLimit now defaults to 10 (the default in apps/v1beta1 was 2, the default in extensions/v1beta1 was to retain all)
 - maxSurge and maxUnavailable now default to 25% (the default in extensions/v1beta1 was 1)

StatefulSet

The **apps/v1beta1** and **apps/v1beta2** API versions of StatefulSet are no longer served as of v1.16.

- Migrate manifests and API clients to use the **apps/v1** API version, available since v1.9.
- All existing persisted objects are accessible via the new API
- Notable changes:
 - spec.selector is now required and immutable after creation; use the existing template labels as the selector for seamless upgrades
 - spec.updateStrategy.type now defaults to RollingUpdate (the default in apps/v1beta1 was OnDelete)

ReplicaSet

The **extensions/v1beta1**, **apps/v1beta1**, and **apps/v1beta2** API versions of ReplicaSet are no longer served as of v1.16.

- Migrate manifests and API clients to use the **apps/v1** API version, available since v1.9.
- All existing persisted objects are accessible via the new API
- Notable changes:
 - spec.selector is now required and immutable after creation; use the existing template labels as the selector for seamless upgrades

PodSecurityPolicy

The **extensions/v1beta1** API version of PodSecurityPolicy is no longer served as of v1.16.

- Migrate manifests and API client to use the **policy/v1beta1** API version, available since v1.10.
- Note that the **policy/v1beta1** API version of PodSecurityPolicy will be removed in v1.25.

What to do

Test with deprecated APIs disabled

You can test your clusters by starting an API server with specific API versions disabled to simulate upcoming removals. Add the following flag to the API server startup arguments:

```
--runtime-config=<group>/<version>=false
```

For example:

```
--runtime-config=admissionregistration.k8s.io/v1beta1=false,apiextensions.k8s.io/v1beta1,...
```

Locate use of deprecated APIs

Use [client warnings, metrics, and audit information available in 1.19+](#) to locate use of deprecated APIs.

Migrate to non-deprecated APIs

- Update custom integrations and controllers to call the non-deprecated APIs
- Change YAML files to reference the non-deprecated APIs

You can use the kubectl convert command to automatically convert an existing object:

```
kubectl convert -f <file> --output-version <group>/<version>.
```

For example, to convert an older Deployment to apps/v1, you can run:

```
kubectl convert -f ./my-deployment.yaml --output-version apps/v1
```

This conversion may use non-ideal default values. To learn more about a specific resource, check the Kubernetes [API reference](#).

Note:

The kubectl convert tool is not installed by default, although in fact it once was part of kubectl itself. For more details, you can read the [deprecation and removal issue](#) for the built-in subcommand.

To learn how to set up kubectl convert on your computer, visit the page that is right for your operating system: [Linux](#), [macOS](#), or [Windows](#).

Kubernetes API health endpoints

The Kubernetes [API server](#) provides API endpoints to indicate the current status of the API server. This page describes these API endpoints and explains how you can use them.

API endpoints for health

The Kubernetes API server provides 3 API endpoints (healthz, livez and readyz) to indicate the current status of the API server. The healthz endpoint is deprecated (since Kubernetes v1.16), and you should use the more specific livez and readyz endpoints instead. The livez endpoint can be used with the --livez-grace-period [flag](#) to specify the startup duration. For a graceful shutdown you can specify the --shutdown-delay-duration [flag](#) with the /readyz endpoint. Machines that check the healthz/livez/readyz of the API server should rely on the HTTP status code. A status code 200 indicates the API server is healthy/live/ready, depending on the called endpoint. The more verbose options shown below are intended to be used by human operators to debug their cluster or understand the state of the API server.

The following examples will show how you can interact with the health API endpoints.

For all endpoints, you can use the verbose parameter to print out the checks and their status. This can be useful for a human operator to debug the current status of the API server, it is not intended to be consumed by a machine:

```
curl -k https://localhost:6443/livez?verbose
```

or from a remote host with authentication:

```
kubectl get --raw='/readyz?verbose'
```

The output will look like this:

```
[+]ping ok
[+]log ok
[+]etcd ok
[+]poststarthook/start-kube-apiserver-admission-initializer ok
[+]poststarthook/generic-apiserver-start-informers ok
[+]poststarthook/start-apiextensions-informers ok
[+]poststarthook/start-apiextensions-controllers ok
[+]poststarthook/crd-informer-synced ok
[+]poststarthook/bootstrap-controller ok
[+]poststarthook/rbac/bootstrap-roles ok
[+]poststarthook/scheduling/bootstrap-system-priority-classes ok
[+]poststarthook/start-cluster-authentication-info-controller ok
[+]poststarthook/start-kube-aggregator-informers ok
[+]poststarthook/apiservice-registration-controller ok
[+]poststarthook/apiservice-status-available-controller ok
[+]poststarthook/kube-apiserver-autoregistration ok
[+]autoregister-completion ok
[+]poststarthook/apiservice-openapi-controller ok
healthz check passed
```

The Kubernetes API server also supports to exclude specific checks. The query parameters can also be combined like in this example:

```
curl -k 'https://localhost:6443/readyz?verbose&exclude=etcd'
```

The output show that the etcd check is excluded:

```
[+]ping ok
[+]log ok
[+]etcd excluded: ok
[+]poststarthook/start-kube-apiserver-admission-initializer ok
[+]poststarthook/generic-apiserver-start-informers ok
[+]poststarthook/start-apiextensions-informers ok
[+]poststarthook/start-apiextensions-controllers ok
[+]poststarthook/crd-informer-synced ok
[+]poststarthook/bootstrap-controller ok
[+]poststarthook/rbac/bootstrap-roles ok
[+]poststarthook/scheduling/bootstrap-system-priority-classes ok
[+]poststarthook/start-cluster-authentication-info-controller ok
[+]poststarthook/start-kube-aggregator-informers ok
[+]poststarthook/apiservice-registration-controller ok
[+]poststarthook/apiservice-status-available-controller ok
[+]poststarthook/kube-apiserver-autoregistration ok
[+]autoregister-completion ok
[+]poststarthook/apiservice-openapi-controller ok
[+]shutdown ok
healthz check passed
```

Individual health checks

FEATURE STATE: Kubernetes v1.28 [alpha]

Each individual health check exposes an HTTP endpoint and can be checked individually. The schema for the individual health checks is /livez/<healthcheck-name> where livez and readyz and be used to indicate if you want to check the liveness or the readiness of the API server. The <healthcheck-name> path can be discovered using the verbose flag from above and take the path between [+] and ok. These individual health checks should not be consumed by machines but can be helpful for a human operator to debug a system:

```
curl -k https://localhost:6443/livez/etcd
```

API Access Control

For an introduction to how Kubernetes implements and controls API access, read [Controlling Access to the Kubernetes API](#).

Reference documentation:

- [Authenticating](#)
 - [Authenticating with Bootstrap Tokens](#)
- [Admission Controllers](#)
 - [Dynamic Admission Control](#)

- [Authorization](#)
 - [Role Based Access Control](#)
 - [Attribute Based Access Control](#)
 - [Node Authorization](#)
 - [Webhook Authorization](#)
- [Certificate Signing Requests](#)
 - including [CSR approval](#) and [certificate signing](#)
- Service accounts
 - [Developer guide](#)
 - [Administration](#)
- [Kubelet Authentication & Authorization](#)
 - including kubelet [TLS bootstrapping](#)

Authenticating

This page provides an overview of authentication.

Users in Kubernetes

All Kubernetes clusters have two categories of users: service accounts managed by Kubernetes, and normal users.

It is assumed that a cluster-independent service manages normal users in the following ways:

- an administrator distributing private keys
- a user store like Keystone or Google Accounts
- a file with a list of usernames and passwords

In this regard, *Kubernetes does not have objects which represent normal user accounts*. Normal users cannot be added to a cluster through an API call.

Even though a normal user cannot be added via an API call, any user that presents a valid certificate signed by the cluster's certificate authority (CA) is considered authenticated. In this configuration, Kubernetes determines the username from the common name field in the 'subject' of the cert (e.g., "/CN=bob"). From there, the role based access control (RBAC) subsystem would determine whether the user is authorized to perform a specific operation on a resource. For more details, refer to the normal users topic in [certificate request](#) for more details about this.

In contrast, service accounts are users managed by the Kubernetes API. They are bound to specific namespaces, and created automatically by the API server or manually through API calls. Service accounts are tied to a set of credentials stored as Secrets, which are mounted into pods allowing in-cluster processes to talk to the Kubernetes API.

API requests are tied to either a normal user or a service account, or are treated as [anonymous requests](#). This means every process inside or outside the cluster, from a human user typing kubectl on a workstation, to kubelets on nodes, to members of the control plane, must authenticate when making requests to the API server, or be treated as an anonymous user.

Authentication strategies

Kubernetes uses client certificates, bearer tokens, or an authenticating proxy to authenticate API requests through authentication plugins. As HTTP requests are made to the API server, plugins attempt to associate the following attributes with the request:

- Username: a string which identifies the end user. Common values might be `kube-admin` or `jane@example.com`.
- UID: a string which identifies the end user and attempts to be more consistent and unique than username.
- Groups: a set of strings, each of which indicates the user's membership in a named logical collection of users. Common values might be `system:masters` or `devops-team`.
- Extra fields: a map of strings to list of strings which holds additional information authorizers may find useful.

All values are opaque to the authentication system and only hold significance when interpreted by an [authorizer](#).

You can enable multiple authentication methods at once. You should usually use at least two methods:

- service account tokens for service accounts
- at least one other method for user authentication.

When multiple authenticator modules are enabled, the first module to successfully authenticate the request short-circuits evaluation. The API server does not guarantee the order authenticators run in.

The `system:authenticated` group is included in the list of groups for all authenticated users.

Integrations with other authentication protocols (LDAP, SAML, Kerberos, alternate x509 schemes, etc) can be accomplished using an [authenticating proxy](#) or the [authentication webhook](#).

X509 client certificates

Client certificate authentication is enabled by passing the `--client-ca-file=SOMEFILE` option to API server. The referenced file must contain one or more certificate authorities to use to validate client certificates presented to the API server. If a client certificate is presented and verified, the common name of the subject is used as the user name for the request. As of Kubernetes 1.4, client certificates can also indicate a user's group memberships using the certificate's organization fields. To include multiple group memberships for a user, include multiple organization fields in the certificate.

For example, using the `openssl` command line tool to generate a certificate signing request:

```
openssl req -new -key jbeda.pem -out jbeda-csr.pem -subj "/CN=jbeda/O=app1/O=app2"
```

This would create a CSR for the username "jbeda", belonging to two groups, "app1" and "app2".

See [Managing Certificates](#) for how to generate a client cert.

Static token file

The API server reads bearer tokens from a file when given the `--token-auth-file=SOMEFILE` option on the command line. Currently, tokens last indefinitely, and the token list cannot be changed without restarting the API server.

The token file is a csv file with a minimum of 3 columns: token, user name, user uid, followed by optional group names.

Note:

If you have more than one group the column must be double quoted e.g.

```
token,user,uid,"group1,group2,group3"
```

Putting a bearer token in a request

When using bearer token authentication from an http client, the API server expects an `Authorization` header with a value of `Bearer <token>`. The bearer token must be a character sequence that can be put in an HTTP header value using no more than the encoding and quoting facilities of HTTP. For example: if the bearer token is `31ada4fd-adec-460c-809a-9e56ceb75269` then it would appear in an HTTP header as shown below.

```
Authorization: Bearer 31ada4fd-adec-460c-809a-9e56ceb75269
```

Bootstrap tokens

FEATURE STATE: Kubernetes v1.18 [stable]

To allow for streamlined bootstrapping for new clusters, Kubernetes includes a dynamically-managed Bearer token type called a *Bootstrap Token*. These tokens are stored as Secrets in the `kube-system` namespace, where they can be dynamically managed and created. Controller Manager contains a `TokenCleaner` controller that deletes bootstrap tokens as they expire.

The tokens are of the form `[a-z0-9]{6}.[a-z0-9]{16}`. The first component is a Token ID and the second component is the Token Secret. You specify the token in an HTTP header as follows:

```
Authorization: Bearer 781292.db7bc3a58fc5f07e
```

You must enable the Bootstrap Token Authenticator with the `--enable-bootstrap-token-auth` flag on the API Server. You must enable the `TokenCleaner` controller via the `--controllers` flag on the Controller Manager. This is done with something like `--controllers=*,tokencleaner`. `kubeadm` will do this for you if you are using it to bootstrap a cluster.

The authenticator authenticates as `system:bootstrap:<Token ID>`. It is included in the `system:bootstrappers` group. The naming and groups are intentionally limited to discourage users from using these tokens past bootstrapping. The user names and group can be used (and are used by `kubeadm`) to craft the appropriate authorization policies to support bootstrapping a cluster.

Please see [Bootstrap Tokens](#) for in depth documentation on the Bootstrap Token authenticator and controllers along with how to manage these tokens with `kubeadm`.

Service account tokens

A service account is an automatically enabled authenticator that uses signed bearer tokens to verify requests. The plugin takes two optional flags:

- `--service-account-key-file` File containing PEM-encoded x509 RSA or ECDSA private or public keys, used to verify ServiceAccount tokens. The specified file can contain multiple keys, and the flag can be specified multiple times with different files. If unspecified, `--tls-private-key-file` is used.
- `--service-account-lookup` If enabled, tokens which are deleted from the API will be revoked.

Service accounts are usually created automatically by the API server and associated with pods running in the cluster through the ServiceAccount [Admission Controller](#). Bearer tokens are mounted into pods at well-known locations, and allow in-cluster processes to talk to the API server. Accounts may be explicitly associated with pods using the `serviceAccountName` field of a PodSpec.

Note: `serviceAccountName` is usually omitted because this is done automatically.

```
apiVersion: apps/v1 # this apiVersion is relevant as of Kubernetes 1.9
kind: Deployment
metadata:
  name: nginx-deployment
  namespace: default
spec:
  replicas: 3
  template:
    metadata:
      # ...
    spec:
      serviceAccountName: bob-the-bot
      containers:
        - name: nginx
          image: nginx:1.14.2
```

Service account bearer tokens are perfectly valid to use outside the cluster and can be used to create identities for long standing jobs that wish to talk to the Kubernetes API. To manually create a service account, use the `kubectl create serviceaccount (NAME)` command. This creates a service account in the current namespace.

```
kubectl create serviceaccount jenkins
```

```
serviceaccount/jenkins created
```

Create an associated token:

```
kubectl create token jenkins
```

```
eyJhbGciOiJSUzI1NiIsImtp...
```

The created token is a signed JSON Web Token (JWT).

The signed JWT can be used as a bearer token to authenticate as the given service account. See [above](#) for how the token is included in a request. Normally these tokens are mounted into pods for in-cluster access to the API server, but can be used from outside the cluster as well.

Service accounts authenticate with the username system:serviceaccount:(NAMESPACE):(SERVICEACCOUNT), and are assigned to the groups system:serviceaccounts and system:serviceaccounts:(NAMESPACE).

Warning: Because service account tokens can also be stored in Secret API objects, any user with write access to Secrets can request a token, and any user with read access to those Secrets can authenticate as the service account. Be cautious when granting permissions to service accounts and read or write capabilities for Secrets.

OpenID Connect Tokens

[OpenID Connect](#) is a flavor of OAuth2 supported by some OAuth2 providers, notably Azure Active Directory, Salesforce, and Google. The protocol's main extension of OAuth2 is an additional field returned with the access token called an [ID Token](#). This token is a JSON Web Token (JWT) with well known fields, such as a user's email, signed by the server.

To identify the user, the authenticator uses the id_token (not the access_token) from the OAuth2 [token response](#) as a bearer token. See [above](#) for how the token is included in a request.

```
sequenceDiagram participant user as User participant idp as Identity Provider
participant kube as Kubectl participant api as API Server
user->>idp: 1. Log in to IdP
activate idp
idp-->>user: 2. Provide access_token, id_token, and refresh_token
deactivate idp
activate user
user->>kube: 3. Call Kubectl
with --token being the id_token
OR add tokens to .kube/config
deactivate user
activate kube
kube->>api: 4. Authorization: Bearer...
deactivate kube
activate api
api->>api: 5. Is JWT signature valid?
api->>api: 6. Has the JWT expired? (iat+exp)
api->>api: 7. User authorized?
api->>kube: 8. Authorized: Perform action and return result
deactivate api
activate kube
kube--x:user: 9. Return result
deactivate kube
```

JavaScript must be [enabled](#) to view this content

1. Log in to your identity provider
2. Your identity provider will provide you with an access_token, id_token and a refresh_token
3. When using kubectl, use your id_token with the --token flag or add it directly to your kubeconfig
4. kubectl sends your id_token in a header called Authorization to the API server
5. The API server will make sure the JWT signature is valid by checking against the certificate named in the configuration
6. Check to make sure the id_token hasn't expired
7. Make sure the user is authorized
8. Once authorized the API server returns a response to kubectl
9. kubectl provides feedback to the user

Since all of the data needed to validate who you are is in the `id_token`, Kubernetes doesn't need to "phone home" to the identity provider. In a model where every request is stateless this provides a very scalable solution for authentication. It does offer a few challenges:

1. Kubernetes has no "web interface" to trigger the authentication process. There is no browser or interface to collect credentials which is why you need to authenticate to your identity provider first.
2. The `id_token` can't be revoked, it's like a certificate so it should be short-lived (only a few minutes) so it can be very annoying to have to get a new token every few minutes.
3. To authenticate to the Kubernetes dashboard, you must use the `kubectl proxy` command or a reverse proxy that injects the `id_token`.

Configuring the API Server

To enable the plugin, configure the following flags on the API server:

Parameter	Description	Example	Required
<code>--oidc-issuer-url</code>	URL of the provider that allows the API server to discover public signing keys. Only URLs that use the <code>https://</code> scheme are accepted. This is typically the provider's discovery URL, changed to have an empty path	If the issuer's OIDC discovery URL is <code>https://accounts.provider.example/.well-known/openid-configuration</code> , the value should be <code>https://accounts.provider.example</code>	Yes
<code>--oidc-client-id</code>	A client id that all tokens must be issued for.	<code>kubernetes</code>	Yes
<code>--oidc-username-claim</code>	JWT claim to use as the user name. By default <code>sub</code> , which is expected to be a unique identifier of the end user. Admins can choose other claims, such as <code>email</code> or <code>name</code> , depending on their provider. However, claims other than <code>email</code> will be prefixed with the issuer URL to prevent naming clashes with other plugins.	<code>sub</code>	No
<code>--oidc-username-prefix</code>	Prefix prepended to username claims to prevent clashes with existing names (such as <code>system:users</code>). For example, the value <code>oidc:</code> will create usernames like <code>oidc:jane.doe</code> . If this flag isn't provided and <code>--oidc-username-claim</code> is a value other than <code>email</code> the prefix defaults to <code>(Issuer URL)</code> where <code>(Issuer URL)</code> is the value of <code>--oidc-issuer-url</code> . The value <code>-</code> can be used to disable all prefixing.	<code>oidc:</code>	No
		<code>groups</code>	No

Parameter	Description	Example	Required
--oidc-groups-claim	JWT claim to use as the user's group. If the claim is present it must be an array of strings.		
--oidc-groups-prefix	Prefix prepended to group claims to prevent clashes with existing names (such as system: groups). For example, the value oidc: will create group names like oidc:engineering and oidc:infra.	oidc:	No
--oidc-required-claim	A key=value pair that describes a required claim in the ID Token. If set, the claim is verified to be present in the ID Token with a matching value. Repeat this flag to specify multiple claims.	claim=value	No
--oidc-ca-file	The path to the certificate for the CA that signed your identity provider's web certificate. Defaults to the host's root CAs.	/etc/kubernetes/ssl/kc-ca.pem	No
--oidc-signing-algs	The signing algorithms accepted. Default is "RS256".	RS512	No

Importantly, the API server is not an OAuth2 client, rather it can only be configured to trust a single issuer. This allows the use of public providers, such as Google, without trusting credentials issued to third parties. Admins who wish to utilize multiple OAuth clients should explore providers which support the azp (authorized party) claim, a mechanism for allowing one client to issue tokens on behalf of another.

Kubernetes does not provide an OpenID Connect Identity Provider. You can use an existing public OpenID Connect Identity Provider (such as Google, or [others](#)). Or, you can run your own Identity Provider, such as [dex](#), [Keycloak](#), CloudFoundry [UAA](#), or Tremolo Security's [OpenUnison](#).

For an identity provider to work with Kubernetes it must:

1. Support [OpenID connect discovery](#); not all do.
2. Run in TLS with non-obsolete ciphers
3. Have a CA signed certificate (even if the CA is not a commercial CA or is self signed)

A note about requirement #3 above, requiring a CA signed certificate. If you deploy your own identity provider (as opposed to one of the cloud providers like Google or Microsoft) you MUST have your identity provider's web server certificate signed by a certificate with the CA flag set to TRUE, even if it is self signed. This is due to GoLang's TLS client implementation being very strict to the standards around certificate validation. If you don't have a CA handy, you can use the [gencert script](#) from the Dex team to create a simple CA and a signed certificate and key pair. Or you can use [this similar script](#) that generates SHA256 certs with a longer life and larger key size.

Setup instructions for specific systems:

- [UAA](#)

- [Dex](#)
- [OpenUnison](#)

Using kubectl

Option 1 - OIDC Authenticator

The first option is to use the kubectl oidc authenticator, which sets the id_token as a bearer token for all requests and refreshes the token once it expires. After you've logged into your provider, use kubectl to add your id_token, refresh_token, client_id, and client_secret to configure the plugin.

Providers that don't return an id_token as part of their refresh token response aren't supported by this plugin and should use "Option 2" below.

```
kubectl config set-credentials USER_NAME \
--auth-provider=oidc \
--auth-provider-arg=idp-issuer-url=( issuer url ) \
--auth-provider-arg=client-id=( your client id ) \
--auth-provider-arg=client-secret=( your client secret ) \
--auth-provider-arg=refresh-token=( your refresh token ) \
--auth-provider-arg=idp-certificate-authority=( path to your ca certificate ) \
--auth-provider-arg=id-token=( your id_token )
```

As an example, running the below command after authenticating to your identity provider:

```
kubectl config set-credentials mmosley \
--auth-provider=oidc \
--auth-provider-arg=idp-issuer-url=https://oidcidp.tremolo.lan:8443/auth/idp/OidcIdP \
--auth-provider-arg=client-id=kubernetes \
--auth-provider-arg=client-secret=1db158f6-177d-4d9c-8a8b-d36869918ec5 \
--auth-provider-arg=refresh-token=q1bKLFOyUiosTfawzA93TzZIDzH2TNa2SMm0zEiPKT
UwME6BkEo6Sql5yUWVBSWpKUGphaWpxSVAfekBOZbBhaEW+VlFUEVRGcluyVF5JT4+haZ
mPsluFoFu5XkpXk5BXqHega4GAXIF+ma+vmYpFcHe5eZR+slBFpZKtQA= \
--auth-provider-arg=idp-certificate-authority=/root/ca.pem \
--auth-provider-arg=id-token=eyJraWQiOjDTj1vaWRjaWRwLnRyZW1vbG8ubGFuLCBP
VT1EZW1vLCBPPVRybWVvbG8gU2VjdXJpdHksIEw9QXJsaW5ndG9uLCBTVD1WaXJnaW5pY
SwgQz1VUy1DTj1rdWJLWNhLTEyMDIxNDc5MjEwMzYwNzMyMTUyIwiYWxnIjoiUlMyNT
YifQ.eyJpc3MiOiJodHRwczovL29pZGNpZHAudHJlbW9sby5sYW46ODQ0My9hdXRoL2lkC9Pa
WRjSWRQIiwiYXVkJioia3ViZXJuZXRLcyIsImV4cCI6MTQ4MzU0OTUxMSwianRpIjoiMm96US1
5TXdFcHV4WDlHZUhQdy1hZyIsImlhdCI6MTQ4MzU0OTQ1MSwibmJmIjoxNDgzNTQ5MzMx
LCJzdWIiOiI0YWViMzdiYS1iNjQ1LTQ4ZmQtYWIzMC0xYTAxZWU0MWUyMTgifQ.w6p4J_6q
Q1HzTG9nrEOrubxIMb9K5hzcMPxc9IxPx2K4xO9l-
oFiUw93daH3m5pluP6K7eOE6txBuRVfEcpJSwlelsOsW8gb8VJcnzMS9EnZpeA0tW_p-
mnkFc3VcfyXuhe5R3G7aa5d8uHv70yJ9Y3-
UhjiN9EhpMdfPAoEB9fYKKkJRzF7utTTIPGrSaSU6d2pcpfYKaxIwePzEkT4DfcQthoZdy9ucNvvL
oi1DIC-
UocFD8HLs8LYKEqSxQvOcvnThbObJ9af71EwmuE21fO5KzMW20KtAeget1gnldOosPtz1G5Ewv
aQ401-RPQzPGMVBl0_zMCAwZttJ4knw
```

Which would produce the below configuration:

```

users:
- name: mmosley
  user:
    auth-provider:
      config:
        client-id: kubernetes
        client-secret: 1db158f6-177d-4d9c-8a8b-d36869918ec5
        id-token: eyJraWQiOjDTj1vaWRjaWRwLnRyZW1vbG8ubGFuLCBPVT1EZW1vLCBPPVR
ybWVvbG8gU2VjdXJpdHksIEw9QXjsaW5ndG9uLCBTVD1WaXJnaW5pYSwgQz1VUy1DTj1rd
WJlLNhLTEyMDIxNDc5MjEwMzYwNzMyMTUyIwiYWxnIjoiUlMyNTYifQ.eyJpc3MiOiJodH
RwczovL29pZGNpZHAudHJlbW9sby5sYW46ODQ0My9hdXRoL2lkC9PaWRjSWRQIwiYXVkJ
oia3ViZXJuZXRLcyIsImV4cCI6MTQ4MzU0OTUxMSwianRpIjoiMm96US15TXdFcHV4WDlHZU
hQdy1hZyIsImlhCI6MTQ4MzU0OTQ1MSwibmJmIjoxNDgzNTQ5MzMxLCJzdWIiOiI0YWViM
zdiYS1iNjQ1LTQ4ZmQtYWlzM0xYTAxZWU0MWUyMTgifQ.w6p4J_6qQ1HzTG9nrEOrubxIM
b9K5hzcMPxc9IxPx2K4xO9l-
oFiUw93daH3m5pluP6K7eOE6txBuRVfEcpJSwlelsOsW8gb8VJcnzMS9EnZpeA0tW_p-
mnkFc3VcfyXuhe5R3G7aa5d8uHv70yJ9Y3-
UhjiN9EhpMdfPAoEB9fYKKkJRzF7utTTIPGrSaSU6d2pcpfYKaxIwePzEkT4DfcQthoZdy9ucNvvL
oi1DIC-
UocFD8HLs8LYKEqSxQvOcvnThbObJ9af71EwmuE21fO5KzMW20KtAeget1gnldOosPtz1G5Eww
aQ401-RPQzPGMVBl0_zMC AwZttJ4knw
  idp-certificate-authority: /root/ca.pem
  idp-issuer-url: https://oidcidp.tremolo.lan:8443/auth/idp/OidcIdP
  refresh-token: q1bKLFOyUiOsTfawzA93TzZIDzH2TNa2SMm0zEiPKTUwME6BkEo6Sql5yU
WVBSWpKUGphaWpxSVAfekBOZbBhaEW+VlFUeVRGcluyVF5JT4+haZmPsluFoFu5XkpXk5B
Xq
  name: oidc

```

Once your id_token expires, kubectl will attempt to refresh your id_token using your refresh_token and client_secret storing the new values for the refresh_token and id_token in your .kube/config.

Option 2 - Use the --token Option

The kubectl command lets you pass in a token using the --token option. Copy and paste the id_token into this option:

```

kubectl --token=eyJhbGciOiJSUzI1NiJ9eyJpc3MiOiJodHRwczovL21sYi50cmVtb2xvLmxhbj04M
DQzL2F1dGgvaWRwL29pZGMiLCJhdWQiOjJrdWJlc5ldGVzIwiZXhwIjoxNDc0NTk2NjY5LCJ
qdGkiOjI2RDUzNXoxUEpFNjJOR3QxaWVYyM9RIiwiaWF0IjoxNDc0NTk2MzY5LCJuYmYiOjE0
NzQ1OTYyNDksInN1YiI6Im13aW5kdSIsInVzZXJfcmsZSI6WyJ1c2VycyIsIm5ldy1uYW1lc3BhY
2Utdmlld2VyIl0sImVtYWlsIjoiXdpbmR1QG5vbW9yZWplZGkuY29tIn0.f2As579n9VNoaKzoFdOQGmXkFKf1FMyNV0-va_B63jn-_nLGSCca_6IVMP8pO-
Zb4KvRqGyTP0r3HkHxYy5c81Anlh8ijarruczl-TK_yF5akjSTHFZD-0gRzlevBDiH8Q79NAr-
ky0P4iIXS8lY9Vnjch5MF74Zx0c3alKJHJUnnpjIACByfF2SCaYzbWFMUNat-K1PaUk5-
ujMBG7yYnr95xD-63n8CO8teGUAAEMx6zRjzhnhbzX-
ajwZLGwGUBT4WqjMs70-6a7_8gZmLZb2az1cZynkFRj2BaCkVT3A2RrjeEwZEtGXlMqKJ1_I2ul
rOVsYx01_yD35-rw get nodes

```

Webhook Token Authentication

Webhook authentication is a hook for verifying bearer tokens.

- `--authentication-token-webhook-config-file` a configuration file describing how to access the remote webhook service.
- `--authentication-token-webhook-cache-ttl` how long to cache authentication decisions. Defaults to two minutes.
- `--authentication-token-webhook-version` determines whether to use `authentication.k8s.io/v1beta1` or `authentication.k8s.io/v1` `TokenReview` objects to send/receive information from the webhook. Defaults to `v1beta1`.

The configuration file uses the [kubeconfig](#) file format. Within the file, clusters refers to the remote service and users refers to the API server webhook. An example would be:

```
# Kubernetes API version
apiVersion: v1
# kind of the API object
kind: Config
# clusters refers to the remote service.
clusters:
  - name: name-of-remote-authn-service
    cluster:
      certificate-authority: /path/to/ca.pem      # CA for verifying the remote service.
      server: https://authn.example.com/authenticate # URL of remote service to query. 'https'
recommended for production.

# users refers to the API server's webhook configuration.
users:
  - name: name-of-api-server
    user:
      client-certificate: /path/to/cert.pem # cert for the webhook plugin to use
      client-key: /path/to/key.pem        # key matching the cert

# kubeconfig files require a context. Provide one for the API server.
current-context: webhook
contexts:
- context:
    cluster: name-of-remote-authn-service
    user: name-of-api-server
    name: webhook
```

When a client attempts to authenticate with the API server using a bearer token as discussed [above](#), the authentication webhook POSTs a JSON-serialized `TokenReview` object containing the token to the remote service.

Note that webhook API objects are subject to the same [versioning compatibility rules](#) as other Kubernetes API objects. Implementers should check the `apiVersion` field of the request to ensure correct deserialization, and **must** respond with a `TokenReview` object of the same version as the request.

- [authentication.k8s.io/v1](#)
- [authentication.k8s.io/v1beta1](#)

Note: The Kubernetes API server defaults to sending authentication.k8s.io/v1beta1 token reviews for backwards compatibility. To opt into receiving authentication.k8s.io/v1 token reviews, the API server must be started with --authentication-token-webhook-version=v1.

```
{  
  "apiVersion": "authentication.k8s.io/v1",  
  "kind": "TokenReview",  
  "spec": {  
    # Opaque bearer token sent to the API server  
    "token": "014fbff9a07c...",  
  
    # Optional list of the audience identifiers for the server the token was presented to.  
    # Audience-aware token authenticators (for example, OIDC token authenticators)  
    # should verify the token was intended for at least one of the audiences in this list,  
    # and return the intersection of this list and the valid audiences for the token in the response  
    status.  
    # This ensures the token is valid to authenticate to the server it was presented to.  
    # If no audiences are provided, the token should be validated to authenticate to the  
    Kubernetes API server.  
    "audiences": ["https://myserver.example.com", "https://myserver.internal.example.com"]  
  }  
}
```

```
{  
  "apiVersion": "authentication.k8s.io/v1beta1",  
  "kind": "TokenReview",  
  "spec": {  
    # Opaque bearer token sent to the API server  
    "token": "014fbff9a07c...",  
  
    # Optional list of the audience identifiers for the server the token was presented to.  
    # Audience-aware token authenticators (for example, OIDC token authenticators)  
    # should verify the token was intended for at least one of the audiences in this list,  
    # and return the intersection of this list and the valid audiences for the token in the response  
    status.  
    # This ensures the token is valid to authenticate to the server it was presented to.  
    # If no audiences are provided, the token should be validated to authenticate to the  
    Kubernetes API server.  
    "audiences": ["https://myserver.example.com", "https://myserver.internal.example.com"]  
  }  
}
```

The remote service is expected to fill the status field of the request to indicate the success of the login. The response body's spec field is ignored and may be omitted. The remote service must return a response using the same TokenReview API version that it received. A successful validation of the bearer token would return:

- [authentication.k8s.io/v1](#)
- [authentication.k8s.io/v1beta1](#)

```
{  
  "apiVersion": "authentication.k8s.io/v1",  
  "kind": "TokenReview",
```

```
"status": {
    "authenticated": true,
    "user": {
        # Required
        "username": "janedoe@example.com",
        # Optional
        "uid": "42",
        # Optional group memberships
        "groups": ["developers", "qa"],
        # Optional additional information provided by the authenticator.
        # This should not contain confidential data, as it can be recorded in logs
        # or API objects, and is made available to admission webhooks.
        "extra": {
            "extrafield1": [
                "extravalue1",
                "extravalue2"
            ]
        }
    },
    # Optional list audience-aware token authenticators can return,
    # containing the audiences from the `spec.audiences` list for which the provided token was
    valid.
    # If this is omitted, the token is considered to be valid to authenticate to the Kubernetes API
    server.
    "audiences": ["https://myserver.example.com"]
}
}
```

```
{
    "apiVersion": "authentication.k8s.io/v1beta1",
    "kind": "TokenReview",
    "status": {
        "authenticated": true,
        "user": {
            # Required
            "username": "janedoe@example.com",
            # Optional
            "uid": "42",
            # Optional group memberships
            "groups": ["developers", "qa"],
            # Optional additional information provided by the authenticator.
            # This should not contain confidential data, as it can be recorded in logs
            # or API objects, and is made available to admission webhooks.
            "extra": {
                "extrafield1": [
                    "extravalue1",
                    "extravalue2"
                ]
            }
        },
        # Optional list audience-aware token authenticators can return,
        # containing the audiences from the `spec.audiences` list for which the provided token was
        valid.
    }
}
```

```
# If this is omitted, the token is considered to be valid to authenticate to the Kubernetes API server.  
    "audiences": ["https://myserver.example.com"]  
}  
}
```

An unsuccessful request would return:

- [authentication.k8s.io/v1](#)
- [authentication.k8s.io/v1beta1](#)

```
{  
  "apiVersion": "authentication.k8s.io/v1",  
  "kind": "TokenReview",  
  "status": {  
    "authenticated": false,  
    # Optionally include details about why authentication failed.  
    # If no error is provided, the API will return a generic Unauthorized message.  
    # The error field is ignored when authenticated=true.  
    "error": "Credentials are expired"  
  }  
}
```

```
{  
  "apiVersion": "authentication.k8s.io/v1beta1",  
  "kind": "TokenReview",  
  "status": {  
    "authenticated": false,  
    # Optionally include details about why authentication failed.  
    # If no error is provided, the API will return a generic Unauthorized message.  
    # The error field is ignored when authenticated=true.  
    "error": "Credentials are expired"  
  }  
}
```

Authenticating Proxy

The API server can be configured to identify users from request header values, such as X-Remote-User. It is designed for use in combination with an authenticating proxy, which sets the request header value.

- `--requestheader-username` Required, case-insensitive. Header names to check, in order, for the user identity. The first header containing a value is used as the username.
- `--requestheader-group` 1.6+. Optional, case-insensitive. "X-Remote-Group" is suggested. Header names to check, in order, for the user's groups. All values in all specified headers are used as group names.
- `--requestheader-extra-headers-prefix` 1.6+. Optional, case-insensitive. "X-Remote-Extra-" is suggested. Header prefixes to look for to determine extra information about the user (typically used by the configured authorization plugin). Any headers beginning with any of the specified prefixes have the prefix removed. The remainder of the header name is lowercased and [percent-decoded](#) and becomes the extra key, and the header value is the extra value.

Note: Prior to 1.11.3 (and 1.10.7, 1.9.11), the extra key could only contain characters which were legal in [HTTP header labels](#).

For example, with this configuration:

```
--requestheader-username=Headers=X-Remote-User  
--requestheader-group=Headers=X-Remote-Group  
--requestheader-extra=Headers-prefix=X-Remote-Extra-
```

this request:

```
GET / HTTP/1.1  
X-Remote-User: fido  
X-Remote-Group: dogs  
X-Remote-Group: dachshunds  
X-Remote-Extra-Acme.com%2Fproject: some-project  
X-Remote-Extra-Scope: openid  
X-Remote-Extra-Scope: profile
```

would result in this user info:

```
name: fido  
groups:  
- dogs  
- dachshunds  
extra:  
acme.com/project:  
- some-project  
scopes:  
- openid  
- profile
```

In order to prevent header spoofing, the authenticating proxy is required to present a valid client certificate to the API server for validation against the specified CA before the request headers are checked. **WARNING:** do **not** reuse a CA that is used in a different context unless you understand the risks and the mechanisms to protect the CA's usage.

- `--requestheader-client-ca-file` Required. PEM-encoded certificate bundle. A valid client certificate must be presented and validated against the certificate authorities in the specified file before the request headers are checked for user names.
- `--requestheader-allowed-names` Optional. List of Common Name values (CNs). If set, a valid client certificate with a CN in the specified list must be presented before the request headers are checked for user names. If empty, any CN is allowed.

Anonymous requests

When enabled, requests that are not rejected by other configured authentication methods are treated as anonymous requests, and given a username of `system:anonymous` and a group of `system:unauthenticated`.

For example, on a server with token authentication configured, and anonymous access enabled, a request providing an invalid bearer token would receive a 401 Unauthorized error. A request providing no bearer token would be treated as an anonymous request.

In 1.5.1-1.5.x, anonymous access is disabled by default, and can be enabled by passing the --anonymous-auth=true option to the API server.

In 1.6+, anonymous access is enabled by default if an authorization mode other than AlwaysAllow is used, and can be disabled by passing the --anonymous-auth=false option to the API server. Starting in 1.6, the ABAC and RBAC authorizers require explicit authorization of the system:anonymous user or the system:unauthenticated group, so legacy policy rules that grant access to the * user or * group do not include anonymous users.

User impersonation

A user can act as another user through impersonation headers. These let requests manually override the user info a request authenticates as. For example, an admin could use this feature to debug an authorization policy by temporarily impersonating another user and seeing if a request was denied.

Impersonation requests first authenticate as the requesting user, then switch to the impersonated user info.

- A user makes an API call with their credentials *and* impersonation headers.
- API server authenticates the user.
- API server ensures the authenticated users have impersonation privileges.
- Request user info is replaced with impersonation values.
- Request is evaluated, authorization acts on impersonated user info.

The following HTTP headers can be used to performing an impersonation request:

- Impersonate-User: The username to act as.
- Impersonate-Group: A group name to act as. Can be provided multiple times to set multiple groups. Optional. Requires "Impersonate-User".
- Impersonate-Extra-(extra name): A dynamic header used to associate extra fields with the user. Optional. Requires "Impersonate-User". In order to be preserved consistently, (extra name) must be lower-case, and any characters which aren't [legal in HTTP header labels](#) MUST be utf8 and [percent-encoded](#).
- Impersonate-Uid: A unique identifier that represents the user being impersonated. Optional. Requires "Impersonate-User". Kubernetes does not impose any format requirements on this string.

Note: Prior to 1.11.3 (and 1.10.7, 1.9.11), (extra name) could only contain characters which were [legal in HTTP header labels](#).

Note: Impersonate-Uid is only available in versions 1.22.0 and higher.

An example of the impersonation headers used when impersonating a user with groups:

```
Impersonate-User: jane.doe@example.com
Impersonate-Group: developers
Impersonate-Group: admins
```

An example of the impersonation headers used when impersonating a user with a UID and extra fields:

```
Impersonate-User: jane.doe@example.com
Impersonate-Extra-dn: cn=jane,ou=engineers,dc=example,dc=com
```

```
Impersonate-Extra-acme.com%2Fproject: some-project  
Impersonate-Extra-scopes: view  
Impersonate-Extra-scopes: development  
Impersonate-Uid: 06f6ce97-e2c5-4ab8-7ba5-7654dd08d52b
```

When using kubectl set the --as flag to configure the Impersonate-User header, set the --as-group flag to configure the Impersonate-Group header.

```
kubectl drain mynode
```

```
Error from server (Forbidden): User "clark" cannot get nodes at the cluster scope. (get nodes mynode)
```

Set the --as and --as-group flag:

```
kubectl drain mynode --as=superman --as-group=system:masters
```

```
node/mynode cordoned  
node/mynode drained
```

Note: kubectl cannot impersonate extra fields or UIDs.

To impersonate a user, group, user identifier (UID) or extra fields, the impersonating user must have the ability to perform the "impersonate" verb on the kind of attribute being impersonated ("user", "group", "uid", etc.). For clusters that enable the RBAC authorization plugin, the following ClusterRole encompasses the rules needed to set user and group impersonation headers:

```
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRole  
metadata:  
  name: impersonator  
rules:  
- apiGroups: [""]  
  resources: ["users", "groups", "serviceaccounts"]  
  verbs: ["impersonate"]
```

For impersonation, extra fields and impersonated UIDs are both under the "authentication.k8s.io" apiGroup. Extra fields are evaluated as sub-resources of the resource "userextras". To allow a user to use impersonation headers for the extra field "scopes" and for UIDs, a user should be granted the following role:

```
apiVersion: rbac.authorization.k8s.io/v1  
kind: ClusterRole  
metadata:  
  name: scopes-and-uid-impersonator  
rules:  
# Can set "Impersonate-Extra-scopes" header and the "Impersonate-Uid" header.  
- apiGroups: ["authentication.k8s.io"]  
  resources: ["userextras/scopes", "uids"]  
  verbs: ["impersonate"]
```

The values of impersonation headers can also be restricted by limiting the set of resourceNames a resource can take.

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: limited-impersonator
rules:
# Can impersonate the user "jane.doe@example.com"
- apiGroups: [""]
  resources: ["users"]
  verbs: ["impersonate"]
  resourceNames: ["jane.doe@example.com"]

# Can impersonate the groups "developers" and "admins"
- apiGroups: [""]
  resources: ["groups"]
  verbs: ["impersonate"]
  resourceNames: ["developers", "admins"]

# Can impersonate the extras field "scopes" with the values "view" and "development"
- apiGroups: ["authentication.k8s.io"]
  resources: ["userextras/scopes"]
  verbs: ["impersonate"]
  resourceNames: ["view", "development"]

# Can impersonate the uid "06f6ce97-e2c5-4ab8-7ba5-7654dd08d52b"
- apiGroups: ["authentication.k8s.io"]
  resources: ["uids"]
  verbs: ["impersonate"]
  resourceNames: ["06f6ce97-e2c5-4ab8-7ba5-7654dd08d52b"]

```

Note: Impersonating a user or group allows you to perform any action as if you were that user or group; for that reason, impersonation is not namespace scoped. If you want to allow impersonation using Kubernetes RBAC, this requires using a ClusterRole and a ClusterRoleBinding, not a Role and RoleBinding.

client-go credential plugins

FEATURE STATE: Kubernetes v1.22 [stable]

k8s.io/client-go and tools using it such as kubectl and kubelet are able to execute an external command to receive user credentials.

This feature is intended for client side integrations with authentication protocols not natively supported by k8s.io/client-go (LDAP, Kerberos, OAuth2, SAML, etc.). The plugin implements the protocol specific logic, then returns opaque credentials to use. Almost all credential plugin use cases require a server side component with support for the [webhook token authenticator](#) to interpret the credential format produced by the client plugin.

Example use case

In a hypothetical use case, an organization would run an external service that exchanges LDAP credentials for user specific, signed tokens. The service would also be capable of responding to

[webhook token authenticator](#) requests to validate the tokens. Users would be required to install a credential plugin on their workstation.

To authenticate against the API:

- The user issues a kubectl command.
- Credential plugin prompts the user for LDAP credentials, exchanges credentials with external service for a token.
- Credential plugin returns token to client-go, which uses it as a bearer token against the API server.
- API server uses the [webhook token authenticator](#) to submit a TokenReview to the external service.
- External service verifies the signature on the token and returns the user's username and groups.

Configuration

Credential plugins are configured through [kubectl config files](#) as part of the user fields.

- [client.authentication.k8s.io/v1](#)
- [client.authentication.k8s.io/v1beta1](#)

```
apiVersion: v1
kind: Config
users:
- name: my-user
  user:
    exec:
      # Command to execute. Required.
      command: "example-client-go-exec-plugin"

      # API version to use when decoding the ExecCredentials resource. Required.
      #
      # The API version returned by the plugin MUST match the version listed here.
      #
      # To integrate with tools that support multiple versions (such as client.authentication.k8s.io/v1beta1),

      # set an environment variable, pass an argument to the tool that indicates which version the exec plugin expects,
      # or read the version from the ExecCredential object in the KUBERNETES_EXEC_INFO environment variable.
      apiVersion: "client.authentication.k8s.io/v1"

      # Environment variables to set when executing the plugin. Optional.
      env:
      - name: "FOO"
        value: "bar"

      # Arguments to pass when executing the plugin. Optional.
      args:
      - "arg1"
      - "arg2"
```

```

# Text shown to the user when the executable doesn't seem to be present. Optional.
installHint: |
  example-client-go-exec-plugin is required to authenticate
  to the current cluster. It can be installed:

  On macOS: brew install example-client-go-exec-plugin

  On Ubuntu: apt-get install example-client-go-exec-plugin

  On Fedora: dnf install example-client-go-exec-plugin

  ...

# Whether or not to provide cluster information, which could potentially contain
# very large CA data, to this exec plugin as a part of the KUBERNETES_EXEC_INFO
# environment variable.
provideClusterInfo: true

# The contract between the exec plugin and the standard input I/O stream. If the
# contract cannot be satisfied, this plugin will not be run and an error will be
# returned. Valid values are "Never" (this exec plugin never uses standard input),
# "IfAvailable" (this exec plugin wants to use standard input if it is available),
# or "Always" (this exec plugin requires standard input to function). Required.
interactiveMode: Never

clusters:
- name: my-cluster
  cluster:
    server: "https://172.17.4.100:6443"
    certificate-authority: "/etc/kubernetes/ca.pem"
    extensions:
      - name: client.authentication.k8s.io/exec # reserved extension name for per cluster exec
        config
        this: can be provided via the KUBERNETES_EXEC_INFO environment variable upon
        setting provideClusterInfo
        you: ["can", "put", "anything", "here"]
  contexts:
- name: my-cluster
  context:
    cluster: my-cluster
    user: my-user
current-context: my-cluster

```

```

apiVersion: v1
kind: Config
users:
- name: my-user
  user:
    exec:
      # Command to execute. Required.
      command: "example-client-go-exec-plugin"

```

```

# API version to use when decoding the ExecCredentials resource. Required.
#
# The API version returned by the plugin MUST match the version listed here.
#
# To integrate with tools that support multiple versions (such as client.authentication.k8s.io/v1),
v1),

# set an environment variable, pass an argument to the tool that indicates which version the exec plugin expects,
    # or read the version from the ExecCredential object in the KUBERNETES_EXEC_INFO environment variable.
apiVersion: "client.authentication.k8s.io/v1beta1"

# Environment variables to set when executing the plugin. Optional.
env:
- name: "FOO"
  value: "bar"

# Arguments to pass when executing the plugin. Optional.
args:
- "arg1"
- "arg2"

# Text shown to the user when the executable doesn't seem to be present. Optional.
installHint: |
  example-client-go-exec-plugin is required to authenticate to the current cluster. It can be installed:

  On macOS: brew install example-client-go-exec-plugin

  On Ubuntu: apt-get install example-client-go-exec-plugin

  On Fedora: dnf install example-client-go-exec-plugin

  ...

# Whether or not to provide cluster information, which could potentially contain very large CA data, to this exec plugin as a part of the KUBERNETES_EXEC_INFO environment variable.
provideClusterInfo: true

# The contract between the exec plugin and the standard input I/O stream. If the contract cannot be satisfied, this plugin will not be run and an error will be returned. Valid values are "Never" (this exec plugin never uses standard input), "IfAvailable" (this exec plugin wants to use standard input if it is available), or "Always" (this exec plugin requires standard input to function). Optional. Defaults to "IfAvailable".
interactiveMode: Never

clusters:
- name: my-cluster
  cluster:
    server: "https://172.17.4.100:6443"

```

```

certificate-authority: "/etc/kubernetes/ca.pem"
extensions:
  - name: client.authentication.k8s.io/exec # reserved extension name for per cluster exec
config
  extension:
    arbitrary: config
    this: can be provided via the KUBERNETES_EXEC_INFO environment variable upon
setting provideClusterInfo
    you: ["can", "put", "anything", "here"]
contexts:
- name: my-cluster
context:
  cluster: my-cluster
  user: my-user
current-context: my-cluster

```

Relative command paths are interpreted as relative to the directory of the config file. If KUBECONFIG is set to /home/jane/kubeconfig and the exec command is ./bin/example-client-go-exec-plugin, the binary /home/jane/bin/example-client-go-exec-plugin is executed.

```

- name: my-user
user:
exec:
  # Path relative to the directory of the kubeconfig
  command: "./bin/example-client-go-exec-plugin"
  apiVersion: "client.authentication.k8s.io/v1"
  interactiveMode: Never

```

Input and output formats

The executed command prints an ExecCredential object to stdout. k8s.io/client-go authenticates against the Kubernetes API using the returned credentials in the status. The executed command is passed an ExecCredential object as input via the KUBERNETES_EXEC_INFO environment variable. This input contains helpful information like the expected API version of the returned ExecCredential object and whether or not the plugin can use stdin to interact with the user.

When run from an interactive session (i.e., a terminal), stdin can be exposed directly to the plugin. Plugins should use the spec.interactive field of the input ExecCredential object from the KUBERNETES_EXEC_INFO environment variable in order to determine if stdin has been provided. A plugin's stdin requirements (i.e., whether stdin is optional, strictly required, or never used in order for the plugin to run successfully) is declared via the user.exec.interactiveMode field in the [kubeconfig](#) (see table below for valid values). The user.exec.interactiveMode field is optional in client.authentication.k8s.io/v1beta1 and required in client.authentication.k8s.io/v1.

interactiveMode values

interactiveMode Value	Meaning
Never	This exec plugin never needs to use standard input, and therefore the exec plugin will be run regardless of whether standard input is available for user input.
IfAvailable	This exec plugin would like to use standard input if it is available, but can still operate if standard input is not available. Therefore, the exec plugin

interactiveMode Value	Meaning
	will be run regardless of whether stdin is available for user input. If standard input is available for user input, then it will be provided to this exec plugin.
Always	This exec plugin requires standard input in order to run, and therefore the exec plugin will only be run if standard input is available for user input. If standard input is not available for user input, then the exec plugin will not be run and an error will be returned by the exec plugin runner.

To use bearer token credentials, the plugin returns a token in the status of the [ExecCredential](#)

- [client.authentication.k8s.io/v1](#)
- [client.authentication.k8s.io/v1beta1](#)

```
{
  "apiVersion": "client.authentication.k8s.io/v1",
  "kind": "ExecCredential",
  "status": {
    "token": "my-bearer-token"
  }
}
```

```
{
  "apiVersion": "client.authentication.k8s.io/v1beta1",
  "kind": "ExecCredential",
  "status": {
    "token": "my-bearer-token"
  }
}
```

Alternatively, a PEM-encoded client certificate and key can be returned to use TLS client auth. If the plugin returns a different certificate and key on a subsequent call, k8s.io/client-go will close existing connections with the server to force a new TLS handshake.

If specified, clientKeyData and clientCertificateData must both must be present.

clientCertificateData may contain additional intermediate certificates to send to the server.

- [client.authentication.k8s.io/v1](#)
- [client.authentication.k8s.io/v1beta1](#)

```
{
  "apiVersion": "client.authentication.k8s.io/v1",
  "kind": "ExecCredential",
  "status": {
    "clientCertificateData": "-----BEGIN CERTIFICATE-----\n...\n-----END CERTIFICATE-----",
    "clientKeyData": "-----BEGIN RSA PRIVATE KEY-----\n...\n-----END RSA PRIVATE KEY-----"
  }
}
```

```
{
  "apiVersion": "client.authentication.k8s.io/v1beta1",
  "kind": "ExecCredential",
```

```
"status": {  
    "clientCertificateData": "-----BEGIN CERTIFICATE-----\n...\\n-----END CERTIFICATE-----",  
    "clientKeyData": "-----BEGIN RSA PRIVATE KEY-----\n...\\n-----END RSA PRIVATE KEY-----"  
}  
}
```

Optionally, the response can include the expiry of the credential formatted as a [RFC 3339](#) timestamp.

Presence or absence of an expiry has the following impact:

- If an expiry is included, the bearer token and TLS credentials are cached until the expiry time is reached, or if the server responds with a 401 HTTP status code, or when the process exits.
 - If an expiry is omitted, the bearer token and TLS credentials are cached until the server responds with a 401 HTTP status code or until the process exits.
 - [client.authentication.k8s.io/v1](#)
 - [client.authentication.k8s.io/v1beta1](#)

```
{  
  "apiVersion": "client.authentication.k8s.io/v1",  
  "kind": "ExecCredential",  
  "status": {  
    "token": "my-bearer-token",  
    "expirationTimestamp": "2018-03-05T17:30:20-08:00"  
  }  
}
```

```
{  
  "apiVersion": "client.authentication.k8s.io/v1beta1",  
  "kind": "ExecCredential",  
  "status": {  
    "token": "my-bearer-token",  
    "expirationTimestamp": "2018-03-05T17:30:20-08:00"  
  }  
}
```

To enable the exec plugin to obtain cluster-specific information, set `provideClusterInfo` on the `user.exec` field in the [kubeconfig](#). The plugin will then be supplied this cluster-specific information in the `KUBERNETES_EXEC_INFO` environment variable. Information from this environment variable can be used to perform cluster-specific credential acquisition logic. The following ExecCredential manifest describes a cluster information sample.

- [client.authentication.k8s.io/v1](#)
 - [client.authentication.k8s.io/v1beta1](#)

```
{  
  "apiVersion": "client.authentication.k8s.io/v1",  
  "kind": "ExecCredential",  
  "spec": {  
    "cluster": {  
      "server": "https://172.17.4.100:6443",  
      "certificate-authority-data": "LS0t..."},  
    "image": "quay.io/jetstack/certifier:v1.1.0",  
    "imagePullPolicy": "IfNotPresent",  
    "path": "/tmp/certifier",  
    "port": 6443  
  }  
}
```

```

"config": {
    "arbitrary": "config",
    "this": "can be provided via the KUBERNETES_EXEC_INFO environment variable upon
setting provideClusterInfo",
    "you": ["can", "put", "anything", "here"]
}
},
"interactive": true
}

{
"apiVersion": "client.authentication.k8s.io/v1beta1",
"kind": "ExecCredential",
"spec": {
"cluster": {
"server": "https://172.17.4.100:6443",
"certificate-authority-data": "LS0t...",
"config": {
"arbitrary": "config",
"this": "can be provided via the KUBERNETES_EXEC_INFO environment variable upon
setting provideClusterInfo",
"you": ["can", "put", "anything", "here"]
}
},
"interactive": true
}
}

```

API access to authentication information for a client

FEATURE STATE: Kubernetes v1.28 [stable]

If your cluster has the API enabled, you can use the SelfSubjectReview API to find out how your Kubernetes cluster maps your authentication information to identify you as a client. This works whether you are authenticating as a user (typically representing a real person) or as a ServiceAccount.

SelfSubjectReview objects do not have any configurable fields. On receiving a request, the Kubernetes API server fills the status with the user attributes and returns it to the user.

Request example (the body would be a SelfSubjectReview):

POST /apis/authentication.k8s.io/v1/selfsubjectreviews

```
{
"apiVersion": "authentication.k8s.io/v1",
"kind": "SelfSubjectReview"
}
```

Response example:

```
{
  "apiVersion": "authentication.k8s.io/v1",
  "kind": "SelfSubjectReview",
  "status": {
    "userInfo": {
      "name": "jane.doe",
      "uid": "b6c7cf4-f166-11ec-8ea0-0242ac120002",
      "groups": [
        "viewers",
        "editors",
        "system:authenticated"
      ],
      "extra": {
        "provider_id": ["token.company.example"]
      }
    }
  }
}
```

For convenience, the `kubectl auth whoami` command is present. Executing this command will produce the following output (yet different user attributes will be shown):

- Simple output example

ATTRIBUTE	VALUE
Username	jane.doe
Groups	[system:authenticated]

- Complex example including extra attributes

ATTRIBUTE	VALUE
Username	jane.doe
UID	b79dbf30-0c6a-11ed-861d-0242ac120002
Groups	[students teachers system:authenticated]
Extra: skills	[reading learning]
Extra: subjects	[math sports]

By providing the `output` flag, it is also possible to print the JSON or YAML representation of the result:

- [JSON](#)
- [YAML](#)

```
{
  "apiVersion": "authentication.k8s.io/v1",
  "kind": "SelfSubjectReview",
  "status": {
    "userInfo": {
      "username": "jane.doe",
      "uid": "b79dbf30-0c6a-11ed-861d-0242ac120002",
      "groups": [
        "students",
        "teachers",
        "system:authenticated"
      ]
    }
  }
}
```

```
],
"extra": {
  "skills": [
    "reading",
    "learning"
  ],
  "subjects": [
    "math",
    "sports"
  ]
}
}
```

```
apiVersion: authentication.k8s.io/v1
kind: SelfSubjectReview
status:
userInfo:
  username: jane.doe
  uid: b79dbf30-0c6a-11ed-861d-0242ac120002
  groups:
    - students
    - teachers
    - system:authenticated
  extra:
    skills:
      - reading
      - learning
    subjects:
      - math
      - sports
```

This feature is extremely useful when a complicated authentication flow is used in a Kubernetes cluster, for example, if you use [webhook token authentication](#) or [authenticating proxy](#).

Note: The Kubernetes API server fills the userInfo after all authentication mechanisms are applied, including [impersonation](#). If you, or an authentication proxy, make a SelfSubjectReview using impersonation, you see the user details and properties for the user that was impersonated.

By default, all authenticated users can create SelfSubjectReview objects when the APISelfSubjectReview feature is enabled. It is allowed by the system:basic-user cluster role.

Note:

You can only make SelfSubjectReview requests if:

- the APISelfSubjectReview [feature gate](#) is enabled for your cluster (not needed for Kubernetes 1.28, but older Kubernetes versions might not offer this feature gate, or might default it to be off)
- (if you are running a version of Kubernetes older than v1.28) the API server for your cluster has the authentication.k8s.io/v1alpha1 or authentication.k8s.io/v1beta1 [API group](#) enabled.

What's next

- Read the [client authentication reference \(v1beta1\)](#)
- Read the [client authentication reference \(v1\)](#)

Authenticating with Bootstrap Tokens

FEATURE STATE: Kubernetes v1.18 [stable]

Bootstrap tokens are a simple bearer token that is meant to be used when creating new clusters or joining new nodes to an existing cluster. It was built to support [kubeadm](#), but can be used in other contexts for users that wish to start clusters without kubeadm. It is also built to work, via RBAC policy, with the [Kubelet TLS Bootstrapping](#) system.

Bootstrap Tokens Overview

Bootstrap Tokens are defined with a specific type (`bootstrap.kubernetes.io/token`) of secrets that lives in the `kube-system` namespace. These Secrets are then read by the Bootstrap Authenticator in the API Server. Expired tokens are removed with the `TokenCleaner` controller in the Controller Manager. The tokens are also used to create a signature for a specific ConfigMap used in a "discovery" process through a `BootstrapSigner` controller.

Token Format

Bootstrap Tokens take the form of `abcdef.0123456789abcdef`. More formally, they must match the regular expression `[a-z0-9]{6}\.[a-z0-9]{16}`.

The first part of the token is the "Token ID" and is considered public information. It is used when referring to a token without leaking the secret part used for authentication. The second part is the "Token Secret" and should only be shared with trusted parties.

Enabling Bootstrap Token Authentication

The Bootstrap Token authenticator can be enabled using the following flag on the API server:

```
--enable-bootstrap-token-auth
```

When enabled, bootstrapping tokens can be used as bearer token credentials to authenticate requests against the API server.

```
Authorization: Bearer 07401b.f395accd246ae52d
```

Tokens authenticate as the username `system:bootstrap:<token id>` and are members of the group `system:bootstrappers`. Additional groups may be specified in the token's Secret.

Expired tokens can be deleted automatically by enabling the `tokencleaner` controller on the controller manager.

```
--controllers=*,tokencleaner
```

Bootstrap Token Secret Format

Each valid token is backed by a secret in the kube-system namespace. You can find the full design doc [here](#).

Here is what the secret looks like.

```
apiVersion: v1
kind: Secret
metadata:
  # Name MUST be of form "bootstrap-token-<token id>"
  name: bootstrap-token-07401b
  namespace: kube-system

  # Type MUST be 'bootstrap.kubernetes.io/token'
  type: bootstrap.kubernetes.io/token
  stringData:
    # Human readable description. Optional.
    description: "The default bootstrap token generated by 'kubeadm init'."

    # Token ID and secret. Required.
    token-id: 07401b
    token-secret: f395accd246ae52d

    # Expiration. Optional.
    expiration: 2017-03-10T03:22:11Z

    # Allowed usages.
    usage-bootstrap-authentication: "true"
    usage-bootstrap-signing: "true"

    # Extra groups to authenticate the token as. Must start with "system:bootstrappers:"
    auth-extra-groups: system:bootstrappers:worker,system:bootstrappers:ingress
```

The type of the secret must be bootstrap.kubernetes.io/token and the name must be bootstrap-token-<token id>. It must also exist in the kube-system namespace.

The usage-bootstrap-* members indicate what this secret is intended to be used for. A value must be set to true to be enabled.

- usage-bootstrap-authentication indicates that the token can be used to authenticate to the API server as a bearer token.
- usage-bootstrap-signing indicates that the token may be used to sign the cluster-info ConfigMap as described below.

The expiration field controls the expiry of the token. Expired tokens are rejected when used for authentication and ignored during ConfigMap signing. The expiry value is encoded as an absolute UTC time using RFC3339. Enable the tokencleaner controller to automatically delete expired tokens.

Token Management with kubeadm

You can use the kubeadm tool to manage tokens on a running cluster. See the [kubeadm token docs](#) for details.

ConfigMap Signing

In addition to authentication, the tokens can be used to sign a ConfigMap. This is used early in a cluster bootstrap process before the client trusts the API server. The signed ConfigMap can be authenticated by the shared token.

Enable ConfigMap signing by enabling the bootstrapsigner controller on the Controller Manager.

```
--controllers=*,bootstrapsigner
```

The ConfigMap that is signed is cluster-info in the kube-public namespace. The typical flow is that a client reads this ConfigMap while unauthenticated and ignoring TLS errors. It then validates the payload of the ConfigMap by looking at a signature embedded in the ConfigMap.

The ConfigMap may look like this:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: cluster-info
  namespace: kube-public
data:
  jws-kubeconfig-07401b: eyJhbGciOiJIUzI1NiIsImtpZCI6IjA3NDAxYj9..tYEfbo6zDNo40MQE07
  aZcQX2m3EB2rO3NuXtxVMyM9U
  kubeconfig: |
    apiVersion: v1
    clusters:
    - cluster:
        certificate-authority-data: <really long certificate data>
        server: https://10.138.0.2:6443
        name: ""
    contexts: []
    current-context: ""
    kind: Config
    preferences: {}
    users: []
```

The kubeconfig member of the ConfigMap is a config file with only the cluster information filled out. The key thing being communicated here is the certificate-authority-data. This may be expanded in the future.

The signature is a JWS signature using the "detached" mode. To validate the signature, the user should encode the kubeconfig payload according to JWS rules (base64 encoded while discarding any trailing =). That encoded payload is then used to form a whole JWS by inserting it between the 2 dots. You can verify the JWS using the HS256 scheme (HMAC-SHA256) with the full token (e.g. 07401b.f395accd246ae52d) as the shared secret. Users *must* verify that HS256 is used.

Warning: Any party with a bootstrapping token can create a valid signature for that token. When using ConfigMap signing it's discouraged to share the same token with many clients, since a compromised client can potentially man-in-the middle another client relying on the signature to bootstrap TLS trust.

Consult the [kubeadm implementation details](#) section for more information.

Certificates and Certificate Signing Requests

Kubernetes certificate and trust bundle APIs enable automation of [X.509](#) credential provisioning by providing a programmatic interface for clients of the Kubernetes API to request and obtain X.509 [certificates](#) from a Certificate Authority (CA).

There is also experimental (alpha) support for distributing [trust bundles](#).

Certificate signing requests

FEATURE STATE: Kubernetes v1.19 [stable]

A CertificateSigningRequest (CSR) resource is used to request that a certificate be signed by a denoted signer, after which the request may be approved or denied before finally being signed.

Request signing process

The CertificateSigningRequest resource type allows a client to ask for an X.509 certificate be issued, based on a signing request. The CertificateSigningRequest object includes a PEM-encoded PKCS#10 signing request in the spec.request field. The CertificateSigningRequest denotes the signer (the recipient that the request is being made to) using the spec.signerName field. Note that spec.signerName is a required key after API version certificates.k8s.io/v1. In Kubernetes v1.22 and later, clients may optionally set the spec.expirationSeconds field to request a particular lifetime for the issued certificate. The minimum valid value for this field is 600, i.e. ten minutes.

Once created, a CertificateSigningRequest must be approved before it can be signed. Depending on the signer selected, a CertificateSigningRequest may be automatically approved by a [controller](#). Otherwise, a CertificateSigningRequest must be manually approved either via the REST API (or client-go) or by running kubectl certificate approve. Likewise, a CertificateSigningRequest may also be denied, which tells the configured signer that it must not sign the request.

For certificates that have been approved, the next step is signing. The relevant signing controller first validates that the signing conditions are met and then creates a certificate. The signing controller then updates the CertificateSigningRequest, storing the new certificate into the status.certificate field of the existing CertificateSigningRequest object. The status.certificate field is either empty or contains a X.509 certificate, encoded in PEM format. The CertificateSigningRequest status.certificate field is empty until the signer does this.

Once the `status.certificate` field has been populated, the request has been completed and clients can now fetch the signed certificate PEM data from the `CertificateSigningRequest` resource. The signers can instead deny certificate signing if the approval conditions are not met.

In order to reduce the number of old `CertificateSigningRequest` resources left in a cluster, a garbage collection controller runs periodically. The garbage collection removes `CertificateSigningRequests` that have not changed state for some duration:

- Approved requests: automatically deleted after 1 hour
- Denied requests: automatically deleted after 1 hour
- Failed requests: automatically deleted after 1 hour
- Pending requests: automatically deleted after 24 hours
- All requests: automatically deleted after the issued certificate has expired

Certificate signing authorization

To allow creating a `CertificateSigningRequest` and retrieving any `CertificateSigningRequest`:

- Verbs: `create, get, list, watch, group: certificates.k8s.io, resource: certificatesigningrequests`

For example:

[access/certificate-signing-request/clusterrole-create.yaml](#)

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: csr-creator
rules:
- apiGroups:
  - certificates.k8s.io
  resources:
  - certificatesigningrequests
  verbs:
  - create
  - get
  - list
  - watch
```

To allow approving a `CertificateSigningRequest`:

- Verbs: `get, list, watch, group: certificates.k8s.io, resource: certificatesigningrequests`
- Verbs: `update, group: certificates.k8s.io, resource: certificatesigningrequests/approval`
- Verbs: `approve, group: certificates.k8s.io, resource: signers, resourceName: <signerNameDomain>/<signerNamePath> or <signerNameDomain>/*`

For example:

[access/certificate-signing-request/clusterrole-approve.yaml](#)

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
```

```

name: csr-approver
rules:
- apiGroups:
  - certificates.k8s.io
  resources:
  - certificatesigningrequests
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - certificates.k8s.io
  resources:
  - certificatesigningrequests/approval
  verbs:
  - update
- apiGroups:
  - certificates.k8s.io
  resources:
  - signers
  resourceNames:
  - example.com/my-signer-name
# example.com/* can be used to authorize for all signers in the 'example.com' domain
  verbs:
  - approve

```

To allow signing a CertificateSigningRequest:

- Verbs: get, list, watch, group: certificates.k8s.io, resource: certificatesigningrequests
- Verbs: update, group: certificates.k8s.io, resource: certificatesigningrequests/status
- Verbs: sign, group: certificates.k8s.io, resource: signers, resourceName: <signerNameDomain>/<signerNamePath> or <signerNameDomain>/*

[access/certificate-signing-request/clusterrole-sign.yaml](#)

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: csr-signer
rules:
- apiGroups:
  - certificates.k8s.io
  resources:
  - certificatesigningrequests
  verbs:
  - get
  - list
  - watch
- apiGroups:
  - certificates.k8s.io
  resources:
  - certificatesigningrequests/status

```

```

verbs:
- update
- apiGroups:
  - certificates.k8s.io
resources:
- signers
resourceNames:
- example.com/my-signer-name
# example.com/* can be used to authorize for all signers in the 'example.com' domain
verbs:
- sign

```

Signers

Signers abstractly represent the entity or entities that might sign, or have signed, a security certificate.

Any signer that is made available for outside a particular cluster should provide information about how the signer works, so that consumers can understand what that means for CertificateSigningRequests and (if enabled) [ClusterTrustBundles](#).

This includes:

1. **Trust distribution:** how trust anchors (CA certificates or certificate bundles) are distributed.
2. **Permitted subjects:** any restrictions on and behavior when a disallowed subject is requested.
3. **Permitted x509 extensions:** including IP subjectAltNames, DNS subjectAltNames, Email subjectAltNames, URI subjectAltNames etc, and behavior when a disallowed extension is requested.
4. **Permitted key usages / extended key usages:** any restrictions on and behavior when usages different than the signer-determined usages are specified in the CSR.
5. **Expiration/certificate lifetime:** whether it is fixed by the signer, configurable by the admin, determined by the CSR spec.expirationSeconds field, etc and the behavior when the signer-determined expiration is different from the CSR spec.expirationSeconds field.
6. **CA bit allowed/disallowed:** and behavior if a CSR contains a request a for a CA certificate when the signer does not permit it.

Commonly, the status.certificate field of a CertificateSigningRequest contains a single PEM-encoded X.509 certificate once the CSR is approved and the certificate is issued. Some signers store multiple certificates into the status.certificate field. In that case, the documentation for the signer should specify the meaning of additional certificates; for example, this might be the certificate plus intermediates to be presented during TLS handshakes.

If you want to make the *trust anchor* (root certificate) available, this should be done separately from a CertificateSigningRequest and its status.certificate field. For example, you could use a ClusterTrustBundle.

The PKCS#10 signing request format does not have a standard mechanism to specify a certificate expiration or lifetime. The expiration or lifetime therefore has to be set through the spec.expirationSeconds field of the CSR object. The built-in signers use the ClusterSigningDuration configuration option, which defaults to 1 year, (the --cluster-signing-duration command-line flag of the kube-controller-manager) as the default when no

`spec.expirationSeconds` is specified. When `spec.expirationSeconds` is specified, the minimum of `spec.expirationSeconds` and `ClusterSigningDuration` is used.

Note: The `spec.expirationSeconds` field was added in Kubernetes v1.22. Earlier versions of Kubernetes do not honor this field. Kubernetes API servers prior to v1.22 will silently drop this field when the object is created.

Kubernetes signers

Kubernetes provides built-in signers that each have a well-known signerName:

1. `kubernetes.io/kube-apiserver-client`: signs certificates that will be honored as client certificates by the API server. Never auto-approved by [kube-controller-manager](#).
 1. Trust distribution: signed certificates must be honored as client certificates by the API server. The CA bundle is not distributed by any other means.
 2. Permitted subjects - no subject restrictions, but approvers and signers may choose not to approve or sign. Certain subjects like cluster-admin level users or groups vary between distributions and installations, but deserve additional scrutiny before approval and signing. The `CertificateSubjectRestriction` admission plugin is enabled by default to restrict `system:masters`, but it is often not the only cluster-admin subject in a cluster.
 3. Permitted x509 extensions - honors `subjectAltName` and key usage extensions and discards other extensions.
 4. Permitted key usages - must include `["client auth"]`. Must not include key usages beyond `["digital signature", "key encipherment", "client auth"]`.
 5. Expiration/certificate lifetime - for the `kube-controller-manager` implementation of this signer, set to the minimum of the `--cluster-signing-duration` option or, if specified, the `spec.expirationSeconds` field of the CSR object.
 6. CA bit allowed/disallowed - not allowed.
2. `kubernetes.io/kube-apiserver-client-kubelet`: signs client certificates that will be honored as client certificates by the API server. May be auto-approved by [kube-controller-manager](#).
 1. Trust distribution: signed certificates must be honored as client certificates by the API server. The CA bundle is not distributed by any other means.
 2. Permitted subjects - organizations are exactly `["system:nodes"]`, common name starts with `"system:node:"`.
 3. Permitted x509 extensions - honors key usage extensions, forbids `subjectAltName` extensions and drops other extensions.
 4. Permitted key usages - `["key encipherment", "digital signature", "client auth"]` or `["digital signature", "client auth"]`.
 5. Expiration/certificate lifetime - for the `kube-controller-manager` implementation of this signer, set to the minimum of the `--cluster-signing-duration` option or, if specified, the `spec.expirationSeconds` field of the CSR object.
 6. CA bit allowed/disallowed - not allowed.

- kubernetes.io/kubelet-serving: signs serving certificates that are honored as a valid
3. kubelet serving certificate by the API server, but has no other guarantees. Never auto-approved by [kube-controller-manager](#).
1. Trust distribution: signed certificates must be honored by the API server as valid to terminate connections to a kubelet. The CA bundle is not distributed by any other means.
 2. Permitted subjects - organizations are exactly ["system:nodes"], common name starts with "system:node:".
 3. Permitted x509 extensions - honors key usage and DNSName/IPAddress subjectAltName extensions, forbids EmailAddress and URI subjectAltName extensions, drops other extensions. At least one DNS or IP subjectAltName must be present.
 4. Permitted key usages - ["key encipherment", "digital signature", "server auth"] or ["digital signature", "server auth"].
 5. Expiration/certificate lifetime - for the kube-controller-manager implementation of this signer, set to the minimum of the --cluster-signing-duration option or, if specified, the spec.expirationSeconds field of the CSR object.
 6. CA bit allowed/disallowed - not allowed.
4. kubernetes.io/legacy-unknown: has no guarantees for trust at all. Some third-party distributions of Kubernetes may honor client certificates signed by it. The stable CertificateSigningRequest API (version certificates.k8s.io/v1 and later) does not allow to set the signerName as kubernetes.io/legacy-unknown. Never auto-approved by [kube-controller-manager](#).

1. Trust distribution: None. There is no standard trust or distribution for this signer in a Kubernetes cluster.
2. Permitted subjects - any
3. Permitted x509 extensions - honors subjectAltName and key usage extensions and discards other extensions.
4. Permitted key usages - any
5. Expiration/certificate lifetime - for the kube-controller-manager implementation of this signer, set to the minimum of the --cluster-signing-duration option or, if specified, the spec.expirationSeconds field of the CSR object.
6. CA bit allowed/disallowed - not allowed.

The kube-controller-manager implements [control plane signing](#) for each of the built in signers. Failures for all of these are only reported in kube-controller-manager logs.

Note: The spec.expirationSeconds field was added in Kubernetes v1.22. Earlier versions of Kubernetes do not honor this field. Kubernetes API servers prior to v1.22 will silently drop this field when the object is created.

Distribution of trust happens out of band for these signers. Any trust outside of those described above are strictly coincidental. For instance, some distributions may honor kubernetes.io/legacy-unknown as client certificates for the kube-apiserver, but this is not a standard. None of these usages are related to ServiceAccount token secrets .data[ca.crt] in any way. That CA bundle is only guaranteed to verify a connection to the API server using the default service (kubernetes.default.svc).

Custom signers

You can also introduce your own custom signer, which should have a similar prefixed name but using your own domain name. For example, if you represent an open source project that uses the domain open-fictional.example then you might use issuer.open-fictional.example/service-mesh as a signer name.

A custom signer uses the Kubernetes API to issue a certificate. See [API-based signers](#).

Signing

Control plane signer

The Kubernetes control plane implements each of the [Kubernetes signers](#), as part of the kube-controller-manager.

Note: Prior to Kubernetes v1.18, the kube-controller-manager would sign any CSRs that were marked as approved.

Note: The spec.expirationSeconds field was added in Kubernetes v1.22. Earlier versions of Kubernetes do not honor this field. Kubernetes API servers prior to v1.22 will silently drop this field when the object is created.

API-based signers

Users of the REST API can sign CSRs by submitting an UPDATE request to the status subresource of the CSR to be signed.

As part of this request, the status.certificate field should be set to contain the signed certificate. This field contains one or more PEM-encoded certificates.

All PEM blocks must have the "CERTIFICATE" label, contain no headers, and the encoded data must be a BER-encoded ASN.1 Certificate structure as described in [section 4 of RFC5280](#).

Example certificate content:

```
-----BEGIN CERTIFICATE-----
MIIDgjCCAmqgAwIBAgIUC1N1EJ4Qnsd322BhDPRwmg3b/oAwDQYJKoZIhvcNAQEL
BQAwXDELMakGA1UEBhMCeHgxCjAIBgNVBAgMAXgxCjAIBgNVBAcMAXgxCjAIBgNV
BAoMAXgxCjAIBgNVBAsMAXgxCzAJBgNVBAMMAMnhMRAwDgYJKoZIhvcNAQkBFgF4
MB4XDTIwMDcwNjIyMDcwMFoXTI1MDcwNTIyMDcwMFowNzEVMBMGA1UEChMMc3lz
dGVtOm5vZGVzMR4wHAYDVQQDExVzeXN0ZW06bm9kZToxMjcuMC4wLjEwggEiMA0G
CSqGSib3DQEBAQUAA4IBDwAwggEKAoIBAQDne5X2eQ1JcLZkKvhzCR4HxI9+ZmU3
+e1zfOywLdoQxrPi+o4hVsUH3q0y52BmA7u1yehHDRSaq9u62cmi5ekgXhXHzGmm
kmW5n0itRECV3SFsSm2DSghRKf0mm6iTYHWDHzUXKdm9lPPWoSOxoR5oqOsm3JEh
Q7Et13wrvTJqBMJo1GTwQuF+HYOku0NF/DLqbZIcpI08yQKyrBgYz2uO51/oNp8a
sTCsV4OUfyHhx2BBLUo4g4SptHFySTBwlRWBnSjZPOhmN74JcpTLB4J5f4iEeA7
2QytZfADckG4wVkhH3C2EJUmRtFIBVirwDn39GXkSGlnvnMgf3uLZ6zNAgMBAAGj
YTBfMA4GA1UdDwEB/wQEAwIFoDATBgNVHSUEDDAKBggBgfBQcDAjAMBgNVHRMB
Af8EAjAAMB0GA1UdDgQWBBTRE12hW54lkQBDeVCcd2f2VSIB1DALBgNVHREEBDAC
ggAwDQYJKoZIhvcNAQELBQADggEBABpZjuIKTq8pCaX8dMEGPWtAykgLsTcD2jYr
L0/TCrqmuialiUa42jQt2OVsVP/L8ofFunj/KjpQU0bvKJPLMRKtmxbhXuQCQi1
qCRkp8o93mHvEz3mTUN+D1cfQ2fpsBENLnpS0F4G/JyY2Vrh19/X8+mImMEK5eOy
```

```
o0BMby7byUj98WmcUvNCiXbC6F45QTmkwEhMqWns0JZQY+/XeDhEcg+lJvz9Eyo2  
aGgPsye1o3DpyXnyfJWAWMhOz7cikS5X2adesbgI86PhEHBXPIJ1v13ZdfCExmdd  
M1flPhLyR54fGaY+7/X8P9AZzPefAkwizeXwe9ii6/a08vWoiE4=  
-----END CERTIFICATE-----
```

Non-PEM content may appear before or after the CERTIFICATE PEM blocks and is unvalidated, to allow for explanatory text as described in [section 5.2 of RFC7468](#).

When encoded in JSON or YAML, this field is base-64 encoded. A CertificateSigningRequest containing the example certificate above would look like this:

```
apiVersion: certificates.k8s.io/v1  
kind: CertificateSigningRequest  
...  
status:  
certificate: "LS0tLS1CRUdJTiBDRVJUSUZJQ0FURS0tLS0tCk1JS..."
```

Approval or rejection

Before a [signer](#) issues a certificate based on a CertificateSigningRequest, the signer typically checks that the issuance for that CSR has been *approved*.

Control plane automated approval

The kube-controller-manager ships with a built-in approver for certificates with a signerName of kubernetes.io/kube-apiserver-client-kubelet that delegates various permissions on CSRs for node credentials to authorization. The kube-controller-manager POSTs SubjectAccessReview resources to the API server in order to check authorization for certificate approval.

Approval or rejection using kubectl

A Kubernetes administrator (with appropriate permissions) can manually approve (or deny) CertificateSigningRequests by using the kubectl certificate approve and kubectl certificate deny commands.

To approve a CSR with kubectl:

```
kubectl certificate approve <certificate-signing-request-name>
```

Likewise, to deny a CSR:

```
kubectl certificate deny <certificate-signing-request-name>
```

Approval or rejection using the Kubernetes API

Users of the REST API can approve CSRs by submitting an UPDATE request to the approval subresource of the CSR to be approved. For example, you could write an [operator](#) that watches for a particular kind of CSR and then sends an UPDATE to approve them.

When you make an approval or rejection request, set either the Approved or Denied status condition based on the state you determine:

For Approved CSRs:

```
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
...
status:
  conditions:
    - lastUpdateTime: "2020-02-08T11:37:35Z"
      lastTransitionTime: "2020-02-08T11:37:35Z"
      message: Approved by my custom approver controller
      reason: ApprovedByMyPolicy # You can set this to any string
      type: Approved
```

For Denied CSRs:

```
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
...
status:
  conditions:
    - lastUpdateTime: "2020-02-08T11:37:35Z"
      lastTransitionTime: "2020-02-08T11:37:35Z"
      message: Denied by my custom approver controller
      reason: DeniedByMyPolicy # You can set this to any string
      type: Denied
```

It's usual to set status.conditions.reason to a machine-friendly reason code using TitleCase; this is a convention but you can set it to anything you like. If you want to add a note for human consumption, use the status.conditions.message field.

Cluster trust bundles

FEATURE STATE: Kubernetes v1.27 [alpha]

Note: In Kubernetes 1.28, you must enable the ClusterTrustBundles [feature gate](#) and the certificates.k8s.io/v1alpha1 [API group](#) in order to use this API.

A ClusterTrustBundles is a cluster-scoped object for distributing X.509 trust anchors (root certificates) to workloads within the cluster. They're designed to work well with the [signer](#) concept from CertificateSigningRequests.

ClusterTrustBundles can be used in two modes: [signer-linked](#) and [signer-unlinked](#).

Common properties and validation

All ClusterTrustBundle objects have strong validation on the contents of their trustBundle field. That field must contain one or more X.509 certificates, DER-serialized, each wrapped in a PEM CERTIFICATE block. The certificates must parse as valid X.509 certificates.

Esoteric PEM features like inter-block data and intra-block headers are either rejected during object validation, or can be ignored by consumers of the object. Additionally, consumers are allowed to reorder the certificates in the bundle with their own arbitrary but stable ordering.

ClusterTrustBundle objects should be considered world-readable within the cluster. If your cluster uses [RBAC](#) authorization, all ServiceAccounts have a default grant that allows them to **get**, **list**, and **watch** all ClusterTrustBundle objects. If you use your own authorization

mechanism and you have enabled ClusterTrustBundles in your cluster, you should set up an equivalent rule to make these objects public within the cluster, so that they work as intended.

If you do not have permission to list cluster trust bundles by default in your cluster, you can impersonate a service account you have access to in order to see available ClusterTrustBundles:

```
kubectl get clustertrustbundles --as='system:serviceaccount:mynamespace:default'
```

Signer-linked ClusterTrustBundles

Signer-linked ClusterTrustBundles are associated with a *signer name*, like this:

```
apiVersion: certificates.k8s.io/v1alpha1
kind: ClusterTrustBundle
metadata:
  name: example.com:mysigner:foo
spec:
  signerName: example.com/mysigner
  trustBundle: "<... PEM data ...>"
```

These ClusterTrustBundles are intended to be maintained by a signer-specific controller in the cluster, so they have several security features:

- To create or update a signer-linked ClusterTrustBundle, you must be permitted to **attest** on the signer (custom authorization verb attest, API group certificates.k8s.io; resource path signers). You can configure authorization for the specific resource name `<signerNameDomain>/<signerNamePath>` or match a pattern such as `<signerNameDomain>/*`.
- Signer-linked ClusterTrustBundles **must** be named with a prefix derived from their `spec.signerName` field. Slashes (/) are replaced with colons (:), and a final colon is appended. This is followed by an arbitrary name. For example, the signer `example.com/mysigner` can be linked to a ClusterTrustBundle `example.com:mysigner:<arbitrary-name>`.

Signer-linked ClusterTrustBundles will typically be consumed in workloads by a combination of a [field selector](#) on the signer name, and a separate [label selector](#).

Signer-unlinked ClusterTrustBundles

Signer-unlinked ClusterTrustBundles have an empty `spec.signerName` field, like this:

```
apiVersion: certificates.k8s.io/v1alpha1
kind: ClusterTrustBundle
metadata:
  name: foo
spec:
  # no signerName specified, so the field is blank
  trustBundle: "<... PEM data ...>"
```

They are primarily intended for cluster configuration use cases. Each signer-unlinked ClusterTrustBundle is an independent object, in contrast to the customary grouping behavior of signer-linked ClusterTrustBundles.

Signer-unlinked ClusterTrustBundles have no attest verb requirement. Instead, you control access to them directly using the usual mechanisms, such as role-based access control.

To distinguish them from signer-linked ClusterTrustBundles, the names of signer-unlinked ClusterTrustBundles **must not** contain a colon (:).

How to issue a certificate for a user

A few steps are required in order to get a normal user to be able to authenticate and invoke an API. First, this user must have a certificate issued by the Kubernetes cluster, and then present that certificate to the Kubernetes API.

Create private key

The following scripts show how to generate PKI private key and CSR. It is important to set CN and O attribute of the CSR. CN is the name of the user and O is the group that this user will belong to. You can refer to [RBAC](#) for standard groups.

```
openssl genrsa -out myuser.key 2048
openssl req -new -key myuser.key -out myuser.csr -subj "/CN=myuser"
```

Create a CertificateSigningRequest

Create a CertificateSigningRequest and submit it to a Kubernetes Cluster via kubectl. Below is a script to generate the CertificateSigningRequest.

```
cat <<EOF | kubectl apply -f -
apiVersion: certificates.k8s.io/v1
kind: CertificateSigningRequest
metadata:
  name: myuser
spec:
  request:
LS0tLS1CRUdjTiBDRVJUSUZJQ0FURSBDRVNUlS0tLS0KTUlJQ1ZqQ0NBVDRDQVFb0V
URVBNTBHQTFRUF3d0dZVzVuWld4aE1JSUJJakFOQmdrcWhraUc5dzBCQVFFRgpBQU9D
QVE4QU1JSUJDZ0tDQVFFQTByczhJTHRhdTYxakx2dHhWTTSVIRWMNDHWlJTWWw0dW1
uVWo4RElaWjBOCnR2MUZtRVFSd3VoaUzsOFEzcWl0Qm0wMUFSMkNJVXBGd2ZzSjZ4MXF3
ckJzVkhZbGlBNVhwRVpZM3ExcGswSDQKM3Z3aGJlK1o2MVNrVHF5SVBYUUwrTWM5T1Ns
bm0xb0R2N0NtSkZNMUIMRVi3QTVGZnZKOEdFRj6dHBoaUIFMwpub1dtdHNZb3JuT2wzc21
HQ2ZGZzR4Zmd4eW8ybmlneFNVekl1bXNnVm9PM2ttT0x1RVF6cXpkakJ3TFJXbWIECklmMX
BMWoyalVnald4UkhCM1gyWnVVV1d1T09PZnpXM01LaE8ybHEvZi9DdS8wYk83c0x0MCt3U
2ZMSU91TFcKcW90b1ZtRmxMMytqTy82WDNDKzBERhk5aUtwbXjjVDNwgZLemE1dHJRSU
RBUUFCb0FBd0RRWUpLb1pJaHzjTgpBUUVMQIFBRGdnRUJBR05WdmVIOGR4ZzNvK21VeVR
kbmFjVmQ1N24zSkExdnZEU1JWREkyQTZ1eXN3ZFp1L1BVCKkwZXpZWVF0RVNnSk1IRmQy
cVVNMjNuNVJsSXJ3R0xuUXFISUh5VStWWHhsdnZsRnpNOVpEWllSTmU3QljvYXgKQV1EdUI
5STZXT3FYbkFvczFqRmxNUG5NbFpqdU5kSGxpT1BjTU1oNndLaTZzZfhpVStHYTJ2RUVLY01j
SVUyRgpvU2djUWdMYTk0aEpacGk3ZnNMdm1OQUxoT045UHdNMGM1dVJVejV4T0dGMUtC
bWRSeEgvbUNOS2JKYjFRQm1HCkkwYitEUEdaTktXTU0xMzhIQXdoV0tkNjVoVHdYOWl4V3Z
HMkh4TG1WQzg0L1BHT0tWQW9FNkpsYWFHdTlQVmkkdjlOSjVaZlZrcXdCd0hKbzZXdk9xV
lA3SVFjZmg3d0drWm89Ci0tLS0tRU5EIENFUlJRklDQVRFIFJFUVVFU1QtLS0tLQo=
  signerName: kubernetes.io/kube-apiserver-client
  expirationSeconds: 86400 # one day
```

```
usages:  
- client auth  
EOF
```

Some points to note:

- usages has to be 'client auth'
- expirationSeconds could be made longer (i.e. 864000 for ten days) or shorter (i.e. 3600 for one hour)
- request is the base64 encoded value of the CSR file content. You can get the content using this command:

```
cat myuser.csr | base64 | tr -d "\n"
```

Approve the CertificateSigningRequest

Use kubectl to create a CSR and approve it.

Get the list of CSRs:

```
kubectl get csr
```

Approve the CSR:

```
kubectl certificate approve myuser
```

Get the certificate

Retrieve the certificate from the CSR:

```
kubectl get csr/myuser -o yaml
```

The certificate value is in Base64-encoded format under status.certificate.

Export the issued certificate from the CertificateSigningRequest.

```
kubectl get csr myuser -o jsonpath='{.status.certificate}'| base64 -d > myuser.crt
```

Create Role and RoleBinding

With the certificate created it is time to define the Role and RoleBinding for this user to access Kubernetes cluster resources.

This is a sample command to create a Role for this new user:

```
kubectl create role developer --verb=create --verb=get --verb=list --verb=update --verb=delete  
--resource=pods
```

This is a sample command to create a RoleBinding for this new user:

```
kubectl create rolebinding developer-binding-myuser --role=developer --user=myuser
```

Add to kubeconfig

The last step is to add this user into the kubeconfig file.

First, you need to add new credentials:

```
kubectl config set-credentials myuser --client-key=myuser.key --client-certificate=myuser.crt --embed-certs=true
```

Then, you need to add the context:

```
kubectl config set-context myuser --cluster=kubernetes --user=myuser
```

To test it, change the context to myuser:

```
kubectl config use-context myuser
```

What's next

- Read [Manage TLS Certificates in a Cluster](#)
- View the source code for the kube-controller-manager built in [signer](#)
- View the source code for the kube-controller-manager built in [approver](#)
- For details of X.509 itself, refer to [RFC 5280](#) section 3.1
- For information on the syntax of PKCS#10 certificate signing requests, refer to [RFC 2986](#)

Admission Controllers Reference

This page provides an overview of Admission Controllers.

What are they?

An *admission controller* is a piece of code that intercepts requests to the Kubernetes API server prior to persistence of the object, but after the request is authenticated and authorized.

Admission controllers may be *validating*, *mutating*, or both. Mutating controllers may modify objects related to the requests they admit; validating controllers may not.

Admission controllers limit requests to create, delete, modify objects. Admission controllers can also block custom verbs, such as a request connect to a Pod via an API server proxy. Admission controllers do *not* (and cannot) block requests to read (**get**, **watch** or **list**) objects.

The admission controllers in Kubernetes 1.28 consist of the [list](#) below, are compiled into the kube-apiserver binary, and may only be configured by the cluster administrator. In that list, there are two special controllers: `MutatingAdmissionWebhook` and `ValidatingAdmissionWebhook`. These execute the mutating and validating (respectively) [admission control webhooks](#) which are configured in the API.

Admission control phases

The admission control process proceeds in two phases. In the first phase, mutating admission controllers are run. In the second phase, validating admission controllers are run. Note again that some of the controllers are both.

If any of the controllers in either phase reject the request, the entire request is rejected immediately and an error is returned to the end-user.

Finally, in addition to sometimes mutating the object in question, admission controllers may sometimes have side effects, that is, mutate related resources as part of request processing. Incrementing quota usage is the canonical example of why this is necessary. Any such side-effect needs a corresponding reclamation or reconciliation process, as a given admission controller does not know for sure that a given request will pass all of the other admission controllers.

Why do I need them?

Several important features of Kubernetes require an admission controller to be enabled in order to properly support the feature. As a result, a Kubernetes API server that is not properly configured with the right set of admission controllers is an incomplete server and will not support all the features you expect.

How do I turn on an admission controller?

The Kubernetes API server flag `enable-admission-plugins` takes a comma-delimited list of admission control plugins to invoke prior to modifying objects in the cluster. For example, the following command line enables the `NamespaceLifecycle` and the `LimitRanger` admission control plugins:

```
kube-apiserver --enable-admission-plugins=NamespaceLifecycle,LimitRanger ...
```

Note: Depending on the way your Kubernetes cluster is deployed and how the API server is started, you may need to apply the settings in different ways. For example, you may have to modify the `systemd` unit file if the API server is deployed as a `systemd` service, you may modify the manifest file for the API server if Kubernetes is deployed in a self-hosted way.

How do I turn off an admission controller?

The Kubernetes API server flag `disable-admission-plugins` takes a comma-delimited list of admission control plugins to be disabled, even if they are in the list of plugins enabled by default.

```
kube-apiserver --disable-admission-plugins=PodNodeSelector,AlwaysDeny ...
```

Which plugins are enabled by default?

To see which admission plugins are enabled:

```
kube-apiserver -h | grep enable-admission-plugins
```

In Kubernetes 1.28, the default ones are:

CertificateApproval, CertificateSigning, CertificateSubjectRestriction, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, LimitRanger, MutatingAdmissionWebhook, NamespaceLifecycle, PersistentVolumeClaimResize, PodSecurity, Priority, ResourceQuota, RuntimeClass, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy, ValidatingAdmissionWebhook

Note: The [ValidatingAdmissionPolicy](#) admission plugin is enabled by default, but is only active if you enable the ValidatingAdmissionPolicy [feature gate](#) and the admissionregistration.k8s.io/v1alpha1 API.

What does each admission controller do?

AlwaysAdmit

FEATURE STATE: Kubernetes v1.13 [deprecated]

Type: Validating.

This admission controller allows all pods into the cluster. It is **deprecated** because its behavior is the same as if there were no admission controller at all.

AlwaysDeny

FEATURE STATE: Kubernetes v1.13 [deprecated]

Type: Validating.

Rejects all requests. AlwaysDeny is **deprecated** as it has no real meaning.

AlwaysPullImages

Type: Mutating and Validating.

This admission controller modifies every new Pod to force the image pull policy to Always. This is useful in a multitenant cluster so that users can be assured that their private images can only be used by those who have the credentials to pull them. Without this admission controller, once an image has been pulled to a node, any pod from any user can use it by knowing the image's name (assuming the Pod is scheduled onto the right node), without any authorization check against the image. When this admission controller is enabled, images are always pulled prior to starting containers, which means valid credentials are required.

CertificateApproval

Type: Validating.

This admission controller observes requests to approve CertificateSigningRequest resources and performs additional authorization checks to ensure the approving user has permission to **approve** certificate requests with the spec.signerName requested on the CertificateSigningRequest resource.

See [Certificate Signing Requests](#) for more information on the permissions required to perform different actions on CertificateSigningRequest resources.

CertificateSigning

Type: Validating.

This admission controller observes updates to the status.certificate field of CertificateSigningRequest resources and performs an additional authorization checks to ensure the signing user has permission to **sign** certificate requests with the spec.signerName requested on the CertificateSigningRequest resource.

See [Certificate Signing Requests](#) for more information on the permissions required to perform different actions on CertificateSigningRequest resources.

CertificateSubjectRestriction

Type: Validating.

This admission controller observes creation of CertificateSigningRequest resources that have a spec.signerName of kubernetes.io/kube-apiserver-client. It rejects any request that specifies a 'group' (or 'organization attribute') of system:masters.

DefaultIngressClass

Type: Mutating.

This admission controller observes creation of Ingress objects that do not request any specific ingress class and automatically adds a default ingress class to them. This way, users that do not request any special ingress class do not need to care about them at all and they will get the default one.

This admission controller does not do anything when no default ingress class is configured. When more than one ingress class is marked as default, it rejects any creation of Ingress with an error and an administrator must revisit their IngressClass objects and mark only one as default (with the annotation "ingressclass.kubernetes.io/is-default-class"). This admission controller ignores any Ingress updates; it acts only on creation.

See the [Ingress](#) documentation for more about ingress classes and how to mark one as default.

DefaultStorageClass

Type: Mutating.

This admission controller observes creation of PersistentVolumeClaim objects that do not request any specific storage class and automatically adds a default storage class to them. This way, users that do not request any special storage class do not need to care about them at all and they will get the default one.

This admission controller does not do anything when no default storage class is configured. When more than one storage class is marked as default, it rejects any creation of PersistentVolumeClaim with an error and an administrator must revisit their StorageClass

objects and mark only one as default. This admission controller ignores any PersistentVolumeClaim updates; it acts only on creation.

See [persistent volume](#) documentation about persistent volume claims and storage classes and how to mark a storage class as default.

DefaultTolerationSeconds

Type: Mutating.

This admission controller sets the default forgiveness toleration for pods to tolerate the taints `notready:NoExecute` and `unreachable:NoExecute` based on the k8s-apiserver input parameters `default-not-ready-toleration-seconds` and `default-unreachable-toleration-seconds` if the pods don't already have toleration for taints `node.kubernetes.io/not-ready:NoExecute` or `node.kubernetes.io/unreachable:NoExecute`. The default value for `default-not-ready-toleration-seconds` and `default-unreachable-toleration-seconds` is 5 minutes.

DenyServiceExternalIPs

Type: Validating.

This admission controller rejects all net-new usage of the Service field externalIPs. This feature is very powerful (allows network traffic interception) and not well controlled by policy. When enabled, users of the cluster may not create new Services which use externalIPs and may not add new values to externalIPs on existing Service objects. Existing uses of externalIPs are not affected, and users may remove values from externalIPs on existing Service objects.

Most users do not need this feature at all, and cluster admins should consider disabling it. Clusters that do need to use this feature should consider using some custom policy to manage usage of it.

This admission controller is disabled by default.

EventRateLimit

FEATURE STATE: Kubernetes v1.13 [alpha]

Type: Validating.

This admission controller mitigates the problem where the API server gets flooded by requests to store new Events. The cluster admin can specify event rate limits by:

- Enabling the EventRateLimit admission controller;
- Referencing an EventRateLimit configuration file from the file provided to the API server's command line flag `--admission-control-config-file`:

```
apiVersion: apiserver.config.k8s.io/v1
kind: AdmissionConfiguration
plugins:
  - name: EventRateLimit
    path: eventconfig.yaml
...
```

There are four types of limits that can be specified in the configuration:

- Server: All Event requests (creation or modifications) received by the API server share a single bucket.
- Namespace: Each namespace has a dedicated bucket.
- User: Each user is allocated a bucket.
- SourceAndObject: A bucket is assigned by each combination of source and involved object of the event.

Below is a sample eventconfig.yaml for such a configuration:

```
apiVersion: eventratelimit.admission.k8s.io/v1alpha1
kind: Configuration
limits:
  - type: Namespace
    qps: 50
    burst: 100
    cacheSize: 2000
  - type: User
    qps: 10
    burst: 50
```

See the [EventRateLimit Config API \(v1alpha1\)](#) for more details.

This admission controller is disabled by default.

ExtendedResourceToleration

Type: Mutating.

This plug-in facilitates creation of dedicated nodes with extended resources. If operators want to create dedicated nodes with extended resources (like GPUs, FPGAs etc.), they are expected to [taint the node](#) with the extended resource name as the key. This admission controller, if enabled, automatically adds tolerations for such taints to pods requesting extended resources, so users don't have to manually add these tolerations.

This admission controller is disabled by default.

ImagePolicyWebhook

Type: Validating.

The ImagePolicyWebhook admission controller allows a backend webhook to make admission decisions.

This admission controller is disabled by default.

Configuration file format

ImagePolicyWebhook uses a configuration file to set options for the behavior of the backend. This file may be json or yaml and has the following format:

```
imagePolicy:  
  kubeConfigFile: /path/to/kubeconfig/for/backend  
  # time in s to cache approval  
  allowTTL: 50  
  # time in s to cache denial  
  denyTTL: 50  
  # time in ms to wait between retries  
  retryBackoff: 500  
  # determines behavior if the webhook backend fails  
  defaultAllow: true
```

Reference the ImagePolicyWebhook configuration file from the file provided to the API server's command line flag --admission-control-config-file:

```
apiVersion: apiserver.config.k8s.io/v1  
kind: AdmissionConfiguration  
plugins:  
  - name: ImagePolicyWebhook  
    path: imagepolicyconfig.yaml  
...
```

Alternatively, you can embed the configuration directly in the file:

```
apiVersion: apiserver.config.k8s.io/v1  
kind: AdmissionConfiguration  
plugins:  
  - name: ImagePolicyWebhook  
    configuration:  
      imagePolicy:  
        kubeConfigFile: <path-to-kubeconfig-file>  
        allowTTL: 50  
        denyTTL: 50  
        retryBackoff: 500  
        defaultAllow: true
```

The ImagePolicyWebhook config file must reference a [kubeconfig](#) formatted file which sets up the connection to the backend. It is required that the backend communicate over TLS.

The kubeconfig file's cluster field must point to the remote service, and the user field must contain the returned authorizer.

```
# clusters refers to the remote service.  
clusters:  
  - name: name-of-remote-imagepolicy-service  
    cluster:  
      certificate-authority: /path/to/ca.pem  # CA for verifying the remote service.  
      server: https://images.example.com/policy # URL of remote service to query. Must use  
'https'.
```

```
# users refers to the API server's webhook configuration.  
users:  
  - name: name-of-api-server  
    user:
```

```
client-certificate: /path/to/cert.pem # cert for the webhook admission controller to use  
client-key: /path/to/key.pem      # key matching the cert
```

For additional HTTP configuration, refer to the [kubeconfig](#) documentation.

Request payloads

When faced with an admission decision, the API Server POSTs a JSON serialized `imagepolicy.k8s.io/v1alpha1 ImageReview` object describing the action. This object contains fields describing the containers being admitted, as well as any pod annotations that match `*.image-policy.k8s.io/*`.

Note: The webhook API objects are subject to the same versioning compatibility rules as other Kubernetes API objects. Implementers should be aware of looser compatibility promises for alpha objects and check the `apiVersion` field of the request to ensure correct deserialization. Additionally, the API Server must enable the `imagepolicy.k8s.io/v1alpha1` API extensions group (`--runtime-config=imagepolicy.k8s.io/v1alpha1=true`).

An example request body:

```
{  
  "apiVersion": "imagepolicy.k8s.io/v1alpha1",  
  "kind": "ImageReview",  
  "spec": {  
    "containers": [  
      {  
        "image": "myrepo/myimage:v1"  
      },  
      {  
        "image": "myrepo/  
myimage@sha256:beb6bd6a68f114c1dc2ea4b28db81bdf91de202a9014972bec5e4d9171d90ed"  
      }  
    ],  
    "annotations": {  
      "mycluster.image-policy.k8s.io/ticket-1234": "break-glass"  
    },  
    "namespace": "mynamespace"  
  }  
}
```

The remote service is expected to fill the `status` field of the request and respond to either allow or disallow access. The response body's `spec` field is ignored, and may be omitted. A permissive response would return:

```
{  
  "apiVersion": "imagepolicy.k8s.io/v1alpha1",  
  "kind": "ImageReview",  
  "status": {  
    "allowed": true  
  }  
}
```

To disallow access, the service would return:

```
{  
  "apiVersion": "imagepolicy.k8s.io/v1alpha1",  
  "kind": "ImageReview",  
  "status": {  
    "allowed": false,  
    "reason": "image currently blacklisted"  
  }  
}
```

For further documentation refer to the [imagepolicy.v1alpha1 API](#).

Extending with Annotations

All annotations on a Pod that match `*.image-policy.k8s.io/*` are sent to the webhook. Sending annotations allows users who are aware of the image policy backend to send extra information to it, and for different backends implementations to accept different information.

Examples of information you might put here are:

- request to "break glass" to override a policy, in case of emergency.
- a ticket number from a ticket system that documents the break-glass request
- provide a hint to the policy server as to the imageID of the image being provided, to save it a lookup

In any case, the annotations are provided by the user and are not validated by Kubernetes in any way.

LimitPodHardAntiAffinityTopology

Type: Validating.

This admission controller denies any pod that defines AntiAffinity topology key other than `kubernetes.io/hostname` in `requiredDuringSchedulingRequiredDuringExecution`.

This admission controller is disabled by default.

LimitRanger

Type: Mutating and Validating.

This admission controller will observe the incoming request and ensure that it does not violate any of the constraints enumerated in the LimitRange object in a Namespace. If you are using LimitRange objects in your Kubernetes deployment, you MUST use this admission controller to enforce those constraints. LimitRanger can also be used to apply default resource requests to Pods that don't specify any; currently, the default LimitRanger applies a 0.1 CPU requirement to all Pods in the default namespace.

See the [LimitRange API reference](#) and the [example of LimitRange](#) for more details.

MutatingAdmissionWebhook

Type: Mutating.

This admission controller calls any mutating webhooks which match the request. Matching webhooks are called in serial; each one may modify the object if it desires.

This admission controller (as implied by the name) only runs in the mutating phase.

If a webhook called by this has side effects (for example, decrementing quota) it *must* have a reconciliation system, as it is not guaranteed that subsequent webhooks or validating admission controllers will permit the request to finish.

If you disable the MutatingAdmissionWebhook, you must also disable the MutatingWebhookConfiguration object in the admissionregistration.k8s.io/v1 group/version via the --runtime-config flag, both are on by default.

Use caution when authoring and installing mutating webhooks

- Users may be confused when the objects they try to create are different from what they get back.
- Built in control loops may break when the objects they try to create are different when read back.
 - Setting originally unset fields is less likely to cause problems than overwriting fields set in the original request. Avoid doing the latter.
- Future changes to control loops for built-in resources or third-party resources may break webhooks that work well today. Even when the webhook installation API is finalized, not all possible webhook behaviors will be guaranteed to be supported indefinitely.

NamespaceAutoProvision

Type: Mutating.

This admission controller examines all incoming requests on namespaced resources and checks if the referenced namespace does exist. It creates a namespace if it cannot be found. This admission controller is useful in deployments that do not want to restrict creation of a namespace prior to its usage.

NamespaceExists

Type: Validating.

This admission controller checks all requests on namespaced resources other than Namespace itself. If the namespace referenced from a request doesn't exist, the request is rejected.

NamespaceLifecycle

Type: Validating.

This admission controller enforces that a Namespace that is undergoing termination cannot have new objects created in it, and ensures that requests in a non-existent Namespace are rejected. This admission controller also prevents deletion of three system reserved namespaces default, kube-system, kube-public.

A Namespace deletion kicks off a sequence of operations that remove all objects (pods, services, etc.) in that namespace. In order to enforce integrity of that process, we strongly recommend running this admission controller.

NodeRestriction

Type: Validating.

This admission controller limits the Node and Pod objects a kubelet can modify. In order to be limited by this admission controller, kubelets must use credentials in the system:nodes group, with a username in the form system:node:<nodeName>. Such kubelets will only be allowed to modify their own Node API object, and only modify Pod API objects that are bound to their node. kubelets are not allowed to update or remove taints from their Node API object.

The NodeRestriction admission plugin prevents kubelets from deleting their Node API object, and enforces kubelet modification of labels under the kubernetes.io/ or k8s.io/ prefixes as follows:

- **Prevents** kubelets from adding/removing/updating labels with a node-restriction.kubernetes.io/ prefix. This label prefix is reserved for administrators to label their Node objects for workload isolation purposes, and kubelets will not be allowed to modify labels with that prefix.
- **Allows** kubelets to add/remove/update these labels and label prefixes:
 - kubernetes.io/hostname
 - kubernetes.io/arch
 - kubernetes.io/os
 - beta.kubernetes.io/instance-type
 - node.kubernetes.io/instance-type
 - failure-domain.beta.kubernetes.io/region (deprecated)
 - failure-domain.beta.kubernetes.io/zone (deprecated)
 - topology.kubernetes.io/region
 - topology.kubernetes.io/zone
 - kubelet.kubernetes.io/-prefixed labels
 - node.kubernetes.io/-prefixed labels

Use of any other labels under the kubernetes.io or k8s.io prefixes by kubelets is reserved, and may be disallowed or allowed by the NodeRestriction admission plugin in the future.

Future versions may add additional restrictions to ensure kubelets have the minimal set of permissions required to operate correctly.

OwnerReferencesPermissionEnforcement

Type: Validating.

This admission controller protects the access to the metadata.ownerReferences of an object so that only users with **delete** permission to the object can change it. This admission controller also protects the access to metadata.ownerReferences[x].blockOwnerDeletion of an object, so that only users with **update** permission to the finalizers subresource of the referenced *owner* can change it.

PersistentVolumeClaimResize

FEATURE STATE: Kubernetes v1.24 [stable]

Type: Validating.

This admission controller implements additional validations for checking incoming PersistentVolumeClaim resize requests.

Enabling the PersistentVolumeClaimResize admission controller is recommended. This admission controller prevents resizing of all claims by default unless a claim's StorageClass explicitly enables resizing by setting allowVolumeExpansion to true.

For example: all PersistentVolumeClaims created from the following StorageClass support volume expansion:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gluster-vol-default
provisioner: kubernetes.io/glusterfs
parameters:
  resturl: "http://192.168.10.100:8080"
  restuser: ""
  secretNamespace: ""
  secretName: ""
allowVolumeExpansion: true
```

For more information about persistent volume claims, see [PersistentVolumeClaims](#).

PersistentVolumeLabel

FEATURE STATE: Kubernetes v1.13 [deprecated]

Type: Mutating.

This admission controller automatically attaches region or zone labels to PersistentVolumes as defined by the cloud provider (for example, Azure or GCP). It helps ensure the Pods and the PersistentVolumes mounted are in the same region and/or zone. If the admission controller doesn't support automatic labelling your PersistentVolumes, you may need to add the labels manually to prevent pods from mounting volumes from a different zone. PersistentVolumeLabel is **deprecated** as labeling for persistent volumes has been taken over by the [cloud-controller-manager](#).

This admission controller is disabled by default.

PodNodeSelector

FEATURE STATE: Kubernetes v1.5 [alpha]

Type: Validating.

This admission controller defaults and limits what node selectors may be used within a namespace by reading a namespace annotation and a global configuration.

This admission controller is disabled by default.

Configuration file format

PodNodeSelector uses a configuration file to set options for the behavior of the backend. Note that the configuration file format will move to a versioned file in a future release. This file may be json or yaml and has the following format:

```
podNodeSelectorPluginConfig:  
  clusterDefaultNodeSelector: name-of-node-selector  
  namespace1: name-of-node-selector  
  namespace2: name-of-node-selector
```

Reference the PodNodeSelector configuration file from the file provided to the API server's command line flag --admission-control-config-file:

```
apiVersion: apiserver.config.k8s.io/v1  
kind: AdmissionConfiguration  
plugins:  
- name: PodNodeSelector  
  path: podnodedeleteor.yaml  
...
```

Configuration Annotation Format

PodNodeSelector uses the annotation key `scheduler.alpha.kubernetes.io/node-selector` to assign node selectors to namespaces.

```
apiVersion: v1  
kind: Namespace  
metadata:  
  annotations:  
    scheduler.alpha.kubernetes.io/node-selector: name-of-node-selector  
  name: namespace3
```

Internal Behavior

This admission controller has the following behavior:

1. If the Namespace has an annotation with a key `scheduler.alpha.kubernetes.io/node-selector`, use its value as the node selector.
2. If the namespace lacks such an annotation, use the `clusterDefaultNodeSelector` defined in the PodNodeSelector plugin configuration file as the node selector.
3. Evaluate the pod's node selector against the namespace node selector for conflicts.
Conflicts result in rejection.
4. Evaluate the pod's node selector against the namespace-specific allowed selector defined in the plugin configuration file. Conflicts result in rejection.

Note: PodNodeSelector allows forcing pods to run on specifically labeled nodes. Also see the PodTolerationRestriction admission plugin, which allows preventing pods from running on specifically tainted nodes.

PodSecurity

FEATURE STATE: Kubernetes v1.25 [stable]

Type: Validating.

The PodSecurity admission controller checks new Pods before they are admitted, determines if it should be admitted based on the requested security context and the restrictions on permitted [Pod Security Standards](#) for the namespace that the Pod would be in.

See the [Pod Security Admission](#) documentation for more information.

PodSecurity replaced an older admission controller named PodSecurityPolicy.

PodTolerationRestriction

FEATURE STATE: Kubernetes v1.7 [alpha]

Type: Mutating and Validating.

The PodTolerationRestriction admission controller verifies any conflict between tolerations of a pod and the tolerations of its namespace. It rejects the pod request if there is a conflict. It then merges the tolerations annotated on the namespace into the tolerations of the pod. The resulting tolerations are checked against a list of allowed tolerations annotated to the namespace. If the check succeeds, the pod request is admitted otherwise it is rejected.

If the namespace of the pod does not have any associated default tolerations or allowed tolerations annotated, the cluster-level default tolerations or cluster-level list of allowed tolerations are used instead if they are specified.

Tolerations to a namespace are assigned via the scheduler.alpha.kubernetes.io/defaultTolerations annotation key. The list of allowed tolerations can be added via the scheduler.alpha.kubernetes.io/tolerationsWhitelist annotation key.

Example for namespace annotations:

```
apiVersion: v1
kind: Namespace
metadata:
  name: apps-that-need-nodes-exclusively
  annotations:
    scheduler.alpha.kubernetes.io/defaultTolerations: '[{"operator": "Exists", "effect": "NoSchedule", "key": "dedicated-node"}]'
    scheduler.alpha.kubernetes.io/tolerationsWhitelist: '[{"operator": "Exists", "effect": "NoSchedule", "key": "dedicated-node"}]'
```

This admission controller is disabled by default.

Priority

Type: Mutating and Validating.

The priority admission controller uses the priorityClassName field and populates the integer value of the priority. If the priority class is not found, the Pod is rejected.

ResourceQuota

Type: Validating.

This admission controller will observe the incoming request and ensure that it does not violate any of the constraints enumerated in the ResourceQuota object in a Namespace. If you are using ResourceQuota objects in your Kubernetes deployment, you MUST use this admission controller to enforce quota constraints.

See the [ResourceQuota API reference](#) and the [example of Resource Quota](#) for more details.

RuntimeClass

Type: Mutating and Validating.

If you define a RuntimeClass with [Pod overhead](#) configured, this admission controller checks incoming Pods. When enabled, this admission controller rejects any Pod create requests that have the overhead already set. For Pods that have a RuntimeClass configured and selected in their .spec, this admission controller sets .spec.overhead in the Pod based on the value defined in the corresponding RuntimeClass.

See also [Pod Overhead](#) for more information.

SecurityContextDeny

Type: Validating.

FEATURE STATE: Kubernetes v1.27 [deprecated]

Caution:

The Kubernetes project recommends that you **do not use** the SecurityContextDeny admission controller.

The SecurityContextDeny admission controller plugin is deprecated and disabled by default. It will be removed in a future version. If you choose to enable the SecurityContextDeny admission controller plugin, you must enable the SecurityContextDeny feature gate as well.

The SecurityContextDeny admission plugin is deprecated because it is outdated and incomplete; it may be unusable or not do what you would expect. As implemented, this plugin is unable to restrict all security-sensitive attributes of the Pod API. For example, the privileged and ephemeralContainers fields were never restricted by this plugin.

The [Pod Security Admission](#) plugin enforcing the [Pod Security Standards](#) Restricted profile captures what this plugin was trying to achieve in a better and up-to-date way.

This admission controller will deny any Pod that attempts to set the following [SecurityContext](#) fields:

- .spec.securityContext.supplementalGroups
- .spec.securityContext.seLinuxOptions
- .spec.securityContext.runAsUser
- .spec.securityContext.fsGroup
- .spec.(init)Containers[*].securityContext.seLinuxOptions
- .spec.(init)Containers[*].securityContext.runAsUser

For more historical context on this plugin, see [The birth of PodSecurityPolicy](#) from the Kubernetes blog article about PodSecurityPolicy and its removal. The article details the PodSecurityPolicy historical context and the birth of the securityContext field for Pods.

ServiceAccount

Type: Mutating and Validating.

This admission controller implements automation for [serviceAccounts](#). The Kubernetes project strongly recommends enabling this admission controller. You should enable this admission controller if you intend to make any use of Kubernetes ServiceAccount objects.

Regarding the annotation kubernetes.io/enforce-mountable-secrets: While the annotation's name suggests it only concerns the mounting of Secrets, its enforcement also extends to other ways Secrets are used in the context of a Pod. Therefore, it is crucial to ensure that all the referenced secrets are correctly specified in the ServiceAccount.

StorageObjectInUseProtection

Type: Mutating.

The StorageObjectInUseProtection plugin adds the kubernetes.io/pvc-protection or kubernetes.io/pv-protection finalizers to newly created Persistent Volume Claims (PVCs) or Persistent Volumes (PV). In case a user deletes a PVC or PV the PVC or PV is not removed until the finalizer is removed from the PVC or PV by PVC or PV Protection Controller. Refer to the [Storage Object in Use Protection](#) for more detailed information.

TaintNodesByCondition

Type: Mutating.

This admission controller [taints](#) newly created Nodes as NotReady and NoSchedule. That tainting avoids a race condition that could cause Pods to be scheduled on new Nodes before their taints were updated to accurately reflect their reported conditions.

ValidatingAdmissionPolicy

Type: Validating.

[This admission controller](#) implements the CEL validation for incoming matched requests. It is enabled when both feature gate validatingadmissionpolicy and admissionregistration.k8s.io/v1alpha1 group/version are enabled. If any of the ValidatingAdmissionPolicy fails, the request fails.

ValidatingAdmissionWebhook

Type: Validating.

This admission controller calls any validating webhooks which match the request. Matching webhooks are called in parallel; if any of them rejects the request, the request fails. This admission controller only runs in the validation phase; the webhooks it calls may not mutate

the object, as opposed to the webhooks called by the MutatingAdmissionWebhook admission controller.

If a webhook called by this has side effects (for example, decrementing quota) it *must* have a reconciliation system, as it is not guaranteed that subsequent webhooks or other validating admission controllers will permit the request to finish.

If you disable the ValidatingAdmissionWebhook, you must also disable the ValidatingWebhookConfiguration object in the admissionregistration.k8s.io/v1 group/version via the --runtime-config flag.

Is there a recommended set of admission controllers to use?

Yes. The recommended admission controllers are enabled by default (shown [here](#)), so you do not need to explicitly specify them. You can enable additional admission controllers beyond the default set using the --enable-admission-plugins flag (**order doesn't matter**).

Dynamic Admission Control

In addition to [compiled-in admission plugins](#), admission plugins can be developed as extensions and run as webhooks configured at runtime. This page describes how to build, configure, use, and monitor admission webhooks.

What are admission webhooks?

Admission webhooks are HTTP callbacks that receive admission requests and do something with them. You can define two types of admission webhooks, [validating admission webhook](#) and [mutating admission webhook](#). Mutating admission webhooks are invoked first, and can modify objects sent to the API server to enforce custom defaults. After all object modifications are complete, and after the incoming object is validated by the API server, validating admission webhooks are invoked and can reject requests to enforce custom policies.

Note: Admission webhooks that need to guarantee they see the final state of the object in order to enforce policy should use a validating admission webhook, since objects can be modified after being seen by mutating webhooks.

Experimenting with admission webhooks

Admission webhooks are essentially part of the cluster control-plane. You should write and deploy them with great caution. Please read the [user guides](#) for instructions if you intend to write/deploy production-grade admission webhooks. In the following, we describe how to quickly experiment with admission webhooks.

Prerequisites

- Ensure that MutatingAdmissionWebhook and ValidatingAdmissionWebhook admission controllers are enabled. [Here](#) is a recommended set of admission controllers to enable in general.

Ensure that the admissionregistration.k8s.io/v1 API is enabled.

•

Write an admission webhook server

Please refer to the implementation of the [admission webhook server](#) that is validated in a Kubernetes e2e test. The webhook handles the AdmissionReview request sent by the API servers, and sends back its decision as an AdmissionReview object in the same version it received.

See the [webhook request](#) section for details on the data sent to webhooks.

See the [webhook response](#) section for the data expected from webhooks.

The example admission webhook server leaves the ClientAuth field [empty](#), which defaults to NoClientCert. This means that the webhook server does not authenticate the identity of the clients, supposedly API servers. If you need mutual TLS or other ways to authenticate the clients, see how to [authenticate API servers](#).

Deploy the admission webhook service

The webhook server in the e2e test is deployed in the Kubernetes cluster, via the [deployment API](#). The test also creates a [service](#) as the front-end of the webhook server. See [code](#).

You may also deploy your webhooks outside of the cluster. You will need to update your webhook configurations accordingly.

Configure admission webhooks on the fly

You can dynamically configure what resources are subject to what admission webhooks via [ValidatingWebhookConfiguration](#) or [MutatingWebhookConfiguration](#).

The following is an example ValidatingWebhookConfiguration, a mutating webhook configuration is similar. See the [webhook configuration](#) section for details about each config field.

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
metadata:
  name: "pod-policy.example.com"
webhooks:
- name: "pod-policy.example.com"
  rules:
  - apiGroups: [""]
    apiVersions: ["v1"]
    operations: ["CREATE"]
    resources: ["pods"]
    scope: "Namespaced"
  clientConfig:
    service:
      namespace: "example-namespace"
      name: "example-service"
      caBundle: <CA_BUNDLE>
  admissionReviewVersions: ["v1"]
```

```
sideEffects: None  
timeoutSeconds: 5
```

Note: You must replace the <CA_BUNDLE> in the above example by a valid CA bundle which is a PEM-encoded (field value is Base64 encoded) CA bundle for validating the webhook's server certificate.

The scope field specifies if only cluster-scoped resources ("Cluster") or namespace-scoped resources ("Namespaced") will match this rule. "*" means that there are no scope restrictions.

Note: When using clientConfig.service, the server cert must be valid for <svc_name>.<svc_namespace>.svc.

Note: Default timeout for a webhook call is 10 seconds, You can set the timeout and it is encouraged to use a short timeout for webhooks. If the webhook call times out, the request is handled according to the webhook's failure policy.

When an API server receives a request that matches one of the rules, the API server sends an admissionReview request to webhook as specified in the clientConfig.

After you create the webhook configuration, the system will take a few seconds to honor the new configuration.

Authenticate API servers

If your admission webhooks require authentication, you can configure the API servers to use basic auth, bearer token, or a cert to authenticate itself to the webhooks. There are three steps to complete the configuration.

- When starting the API server, specify the location of the admission control configuration file via the --admission-control-config-file flag.
- In the admission control configuration file, specify where the MutatingAdmissionWebhook controller and ValidatingAdmissionWebhook controller should read the credentials. The credentials are stored in kubeConfig files (yes, the same schema that's used by kubectl), so the field name is kubeConfigFile. Here is an example admission control configuration file:

- [apiserver.config.k8s.io/v1](#)
- [apiserver.k8s.io/v1alpha1](#)

```
apiVersion: apiserver.config.k8s.io/v1  
kind: AdmissionConfiguration  
plugins:  
- name: ValidatingAdmissionWebhook  
  configuration:  
    apiVersion: apiserver.config.k8s.io/v1  
    kind: WebhookAdmissionConfiguration  
    kubeConfigFile: "<path-to-kubeconfig-file>"  
- name: MutatingAdmissionWebhook  
  configuration:  
    apiVersion: apiserver.config.k8s.io/v1  
    kind: WebhookAdmissionConfiguration  
    kubeConfigFile: "<path-to-kubeconfig-file>"
```

```

# Deprecated in v1.17 in favor of apiserver.config.k8s.io/v1
apiVersion: apiserver.k8s.io/v1alpha1
kind: AdmissionConfiguration
plugins:
- name: ValidatingAdmissionWebhook
  configuration:
    # Deprecated in v1.17 in favor of apiserver.config.k8s.io/v1,
    kind=WebhookAdmissionConfiguration
    apiVersion: apiserver.config.k8s.io/v1alpha1
    kind: WebhookAdmission
    kubeConfigFile: "<path-to-kubeconfig-file>"
- name: MutatingAdmissionWebhook
  configuration:
    # Deprecated in v1.17 in favor of apiserver.config.k8s.io/v1,
    kind=WebhookAdmissionConfiguration
    apiVersion: apiserver.config.k8s.io/v1alpha1
    kind: WebhookAdmission
    kubeConfigFile: "<path-to-kubeconfig-file>"
```

For more information about AdmissionConfiguration, see the [AdmissionConfiguration \(v1\) reference](#). See the [webhook configuration](#) section for details about each config field.

In the kubeConfig file, provide the credentials:

```

apiVersion: v1
kind: Config
users:
# name should be set to the DNS name of the service or the host (including port) of the URL the
# webhook is configured to speak to.
# If a non-443 port is used for services, it must be included in the name when configuring 1.16+
# API servers.
#
# For a webhook configured to speak to a service on the default port (443), specify the DNS
# name of the service:
# - name: webhook1.ns1.svc
#   user: ...
#
# For a webhook configured to speak to a service on non-default port (e.g. 8443), specify the
# DNS name and port of the service in 1.16+:
# - name: webhook1.ns1.svc:8443
#   user: ...
#
# and optionally create a second stanza using only the DNS name of the service for
# compatibility with 1.15 API servers:
# - name: webhook1.ns1.svc
#   user: ...
#
# For webhooks configured to speak to a URL, match the host (and port) specified in the
# webhook's URL. Examples:
# A webhook with `url: https://www.example.com`:
# - name: www.example.com
#   user: ...
#
# A webhook with `url: https://www.example.com:443`:
```

```

# - name: www.example.com:443
#   user: ...
#
# A webhook with `url: https://www.example.com:8443`:
# - name: www.example.com:8443
#   user: ...
#
- name: 'webhook1.ns1.svc'
  user:
    client-certificate-data: "<pem encoded certificate>"
    client-key-data: "<pem encoded key>"
  # The `name` supports using * to wildcard-match prefixing segments.
- name: '*.webhook-company.org'
  user:
    password: "<password>"
    username: "<name>"
  # '*' is the default match.
- name: '*'
  user:
    token: "<token>"
```

Of course you need to set up the webhook server to handle these authentication requests.

Webhook request and response

Request

Webhooks are sent as POST requests, with Content-Type: application/json, with an AdmissionReview API object in the admission.k8s.io API group serialized to JSON as the body.

Webhooks can specify what versions of AdmissionReview objects they accept with the admissionReviewVersions field in their configuration:

```

apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  admissionReviewVersions: ["v1", "v1beta1"]
```

admissionReviewVersions is a required field when creating webhook configurations. Webhooks are required to support at least one AdmissionReview version understood by the current and previous API server.

API servers send the first AdmissionReview version in the admissionReviewVersions list they support. If none of the versions in the list are supported by the API server, the configuration will not be allowed to be created. If an API server encounters a webhook configuration that was previously created and does not support any of the AdmissionReview versions the API server knows how to send, attempts to call to the webhook will fail and be subject to the [failure policy](#).

This example shows the data contained in an AdmissionReview object for a request to update the scale subresource of an apps/v1 Deployment:

```
apiVersion: admission.k8s.io/v1
kind: AdmissionReview
request:
  # Random uid uniquely identifying this admission call
  uid: 705ab4f5-6393-11e8-b7cc-42010a800002

  # Fully-qualified group/version/kind of the incoming object
  kind:
    group: autoscaling
    version: v1
    kind: Scale

  # Fully-qualified group/version/kind of the resource being modified
  resource:
    group: apps
    version: v1
    resource: deployments

  # subresource, if the request is to a subresource
  subResource: scale

  # Fully-qualified group/version/kind of the incoming object in the original request to the API
  # server.
  # This only differs from `kind` if the webhook specified `matchPolicy: Equivalent` and the
  # original request to the API server was converted to a version the webhook registered for.
  requestKind:
    group: autoscaling
    version: v1
    kind: Scale

  # Fully-qualified group/version/kind of the resource being modified in the original request to
  # the API server.
  # This only differs from `resource` if the webhook specified `matchPolicy: Equivalent` and the
  # original request to the API server was converted to a version the webhook registered for.
  requestResource:
    group: apps
    version: v1
    resource: deployments

  # subresource, if the request is to a subresource

  # This only differs from `subResource` if the webhook specified `matchPolicy: Equivalent` and the
  # original request to the API server was converted to a version the webhook registered for.
  requestSubResource: scale

  # Name of the resource being modified
  name: my-deployment

  # Namespace of the resource being modified, if the resource is namespaced (or is a Namespace
  # object)
  namespace: my-namespace
```

```

# operation can be CREATE, UPDATE, DELETE, or CONNECT
operation: UPDATE

userInfo:
  # Username of the authenticated user making the request to the API server
  username: admin

  # UID of the authenticated user making the request to the API server
  uid: 014fbff9a07c

  # Group memberships of the authenticated user making the request to the API server
  groups:
    - system:authenticated
    - my-admin-group
  # Arbitrary extra info associated with the user making the request to the API server.
  # This is populated by the API server authentication layer and should be included
  # if any SubjectAccessReview checks are performed by the webhook.

extra:
  some-key:
    - some-value1
    - some-value2

# object is the new object being admitted.
# It is null for DELETE operations.
object:
  apiVersion: autoscaling/v1
  kind: Scale

# oldObject is the existing object.
# It is null for CREATE and CONNECT operations.
oldObject:
  apiVersion: autoscaling/v1
  kind: Scale

# options contains the options for the operation being admitted, like meta.k8s.io/v1
CreateOptions, UpdateOptions, or DeleteOptions.
# It is null for CONNECT operations.
options:
  apiVersion: meta.k8s.io/v1
  kind: UpdateOptions

# dryRun indicates the API request is running in dry run mode and will not be persisted.
# Webhooks with side effects should avoid actuating those side effects when dryRun is true.
# See http://k8s.io/docs/reference/using-api/api-concepts/#make-a-dry-run-request for more
details.
dryRun: False

```

Response

Webhooks respond with a 200 HTTP status code, Content-Type: application/json, and a body containing an AdmissionReview object (in the same version they were sent), with the response stanza populated, serialized to JSON.

At a minimum, the response stanza must contain the following fields:

- uid, copied from the request.uid sent to the webhook
- allowed, either set to true or false

Example of a minimal response from a webhook to allow a request:

```
{  
  "apiVersion": "admission.k8s.io/v1",  
  "kind": "AdmissionReview",  
  "response": {  
    "uid": "<value from request.uid>",  
    "allowed": true  
  }  
}
```

Example of a minimal response from a webhook to forbid a request:

```
{  
  "apiVersion": "admission.k8s.io/v1",  
  "kind": "AdmissionReview",  
  "response": {  
    "uid": "<value from request.uid>",  
    "allowed": false  
  }  
}
```

When rejecting a request, the webhook can customize the http code and message returned to the user using the status field. The specified status object is returned to the user. See the [API documentation](#) for details about the status type. Example of a response to forbid a request, customizing the HTTP status code and message presented to the user:

```
{  
  "apiVersion": "admission.k8s.io/v1",  
  "kind": "AdmissionReview",  
  "response": {  
    "uid": "<value from request.uid>",  
    "allowed": false,  
    "status": {  
      "code": 403,  
      "message": "You cannot do this because it is Tuesday and your name starts with A"  
    }  
  }  
}
```

When allowing a request, a mutating admission webhook may optionally modify the incoming object as well. This is done using the patch and patchType fields in the response. The only currently supported patchType is JSONPatch. See [JSON patch](#) documentation for more details. For patchType: JSONPatch, the patch field contains a base64-encoded array of JSON patch operations.

As an example, a single patch operation that would set spec.replicas would be [{"op": "add", "path": "/spec/replicas", "value": 3}]

Base64-encoded, this would be
W3sib3AiOiAiYWRkIiwgInBhdGgiOiAiL3NwZWMvcmVwbGljYXMiLCAdmFsdWUiOiAzfV0=

So a webhook response to add that label would be:

```
{  
  "apiVersion": "admission.k8s.io/v1",  
  "kind": "AdmissionReview",  
  "response": {  
    "uid": "<value from request.uid>",  
    "allowed": true,  
    "patchType": "JSONPatch",  
    "patch": "W3sib3AiOiAiYWRkIiwgInBhdGgiOiAiL3NwZWMvcmVwbGljYXMiLCAdmFsdW  
UiOiAzfV0="  
  }  
}
```

Admission webhooks can optionally return warning messages that are returned to the requesting client in HTTP Warning headers with a warning code of 299. Warnings can be sent with allowed or rejected admission responses.

If you're implementing a webhook that returns a warning:

- Don't include a "Warning:" prefix in the message
- Use warning messages to describe problems the client making the API request should correct or be aware of
- Limit warnings to 120 characters if possible

Caution: Individual warning messages over 256 characters may be truncated by the API server before being returned to clients. If more than 4096 characters of warning messages are added (from all sources), additional warning messages are ignored.

```
{  
  "apiVersion": "admission.k8s.io/v1",  
  "kind": "AdmissionReview",  
  "response": {  
    "uid": "<value from request.uid>",  
    "allowed": true,  
    "warnings": [  
      "duplicate envvar entries specified with name MY_ENV",  
      "memory request less than 4MB specified for container mycontainer, which will not start  
successfully"  
    ]  
  }  
}
```

Webhook configuration

To register admission webhooks, create MutatingWebhookConfiguration or ValidatingWebhookConfiguration API objects. The name of a MutatingWebhookConfiguration or a ValidatingWebhookConfiguration object must be a valid [DNS subdomain name](#).

Each configuration can contain one or more webhooks. If multiple webhooks are specified in a single configuration, each must be given a unique name. This is required in order to make resulting audit logs and metrics easier to match up to active configurations.

Each webhook defines the following things.

Matching requests: rules

Each webhook must specify a list of rules used to determine if a request to the API server should be sent to the webhook. Each rule specifies one or more operations, apiGroups, apiVersions, and resources, and a resource scope:

- operations lists one or more operations to match. Can be "CREATE", "UPDATE", "DELETE", "CONNECT", or "*" to match all.
- apiGroups lists one or more API groups to match. "" is the core API group. "*" matches all API groups.
- apiVersions lists one or more API versions to match. "*" matches all API versions.
- resources lists one or more resources to match.
 - "*" matches all resources, but not subresources.
 - "*/%" matches all resources and subresources.
 - "pods/%" matches all subresources of pods.
 - "*/status" matches all status subresources.
- scope specifies a scope to match. Valid values are "Cluster", "Namespaced", and "*". Subresources match the scope of their parent resource. Default is "*".
 - "Cluster" means that only cluster-scoped resources will match this rule (Namespace API objects are cluster-scoped).
 - "Namespaced" means that only namespaced resources will match this rule.
 - "*" means that there are no scope restrictions.

If an incoming request matches one of the specified operations, groups, versions, resources, and scope for any of a webhook's rules, the request is sent to the webhook.

Here are other examples of rules that could be used to specify which resources should be intercepted.

Match CREATE or UPDATE requests to apps/v1 and apps/v1beta1 deployments and replicasesets:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
...
webhooks:
- name: my-webhook.example.com
  rules:
  - operations: ["CREATE", "UPDATE"]
    apiGroups: ["apps"]
    apiVersions: ["v1", "v1beta1"]
    resources: ["deployments", "replicasesets"]
```

```
scope: "Namespaced"
```

```
...
```

Match create requests for all resources (but not subresources) in all API groups and versions:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  rules:
    - operations: ["CREATE"]
      apiGroups: ["*"]
      apiVersions: ["*"]
      resources: ["*"]
      scope: "*"
```

Match update requests for all status subresources in all API groups and versions:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  rules:
    - operations: ["UPDATE"]
      apiGroups: ["*"]
      apiVersions: ["*"]
      resources: ["/status"]
      scope: "*"
```

Matching requests: objectSelector

Webhooks may optionally limit which requests are intercepted based on the labels of the objects they would be sent, by specifying an objectSelector. If specified, the objectSelector is evaluated against both the object and oldObject that would be sent to the webhook, and is considered to match if either object matches the selector.

A null object (oldObject in the case of create, or newObj in the case of delete), or an object that cannot have labels (like a DeploymentRollback or a PodProxyOptions object) is not considered to match.

Use the object selector only if the webhook is opt-in, because end users may skip the admission webhook by setting the labels.

This example shows a mutating webhook that would match a CREATE of any resource (but not subresources) with the label foo: bar:

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  objectSelector:
    matchLabels:
      foo: bar
  rules:
```

```
- operations: ["CREATE"]
  apiGroups: ["*"]
  apiVersions: ["*"]
  resources: ["*"]
  scope: "*"
```

See [labels concept](#) for more examples of label selectors.

Matching requests: namespaceSelector

Webhooks may optionally limit which requests for namespaced resources are intercepted, based on the labels of the containing namespace, by specifying a namespaceSelector.

The namespaceSelector decides whether to run the webhook on a request for a namespaced resource (or a Namespace object), based on whether the namespace's labels match the selector. If the object itself is a namespace, the matching is performed on object.metadata.labels. If the object is a cluster scoped resource other than a Namespace, namespaceSelector has no effect.

This example shows a mutating webhook that matches a CREATE of any namespaced resource inside a namespace that does not have a "runlevel" label of "0" or "1":

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
  - name: my-webhook.example.com
    namespaceSelector:
      matchExpressions:
        - key: runlevel
          operator: NotIn
          values: ["0","1"]
    rules:
      - operations: ["CREATE"]
        apiGroups: ["*"]
        apiVersions: ["*"]
        resources: ["*"]
        scope: "Namespaced"
```

This example shows a validating webhook that matches a CREATE of any namespaced resource inside a namespace that is associated with the "environment" of "prod" or "staging":

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
  - name: my-webhook.example.com
    namespaceSelector:
      matchExpressions:
        - key: environment
          operator: In
          values: ["prod","staging"]
    rules:
      - operations: ["CREATE"]
        apiGroups: ["*"]
        apiVersions: ["*"]
```

```
resources: ["*"]
scope: "Namespaced"
```

See [labels concept](#) for more examples of label selectors.

Matching requests: matchPolicy

API servers can make objects available via multiple API groups or versions.

For example, if a webhook only specified a rule for some API groups/versions (like apiGroups: ["apps"], apiVersions:["v1","v1beta1"]), and a request was made to modify the resource via another API group/version (like extensions/v1beta1), the request would not be sent to the webhook.

The matchPolicy lets a webhook define how its rules are used to match incoming requests. Allowed values are Exact or Equivalent.

- Exact means a request should be intercepted only if it exactly matches a specified rule.
- Equivalent means a request should be intercepted if modifies a resource listed in rules, even via another API group or version.

In the example given above, the webhook that only registered for apps/v1 could use matchPolicy:

- matchPolicy: Exact would mean the extensions/v1beta1 request would not be sent to the webhook
- matchPolicy: Equivalent means the extensions/v1beta1 request would be sent to the webhook (with the objects converted to a version the webhook had specified: apps/v1)

Specifying Equivalent is recommended, and ensures that webhooks continue to intercept the resources they expect when upgrades enable new versions of the resource in the API server.

When a resource stops being served by the API server, it is no longer considered equivalent to other versions of that resource that are still served. For example, extensions/v1beta1 deployments were first deprecated and then removed (in Kubernetes v1.16).

Since that removal, a webhook with a apiGroups:["extensions"], apiVersions:["v1beta1"], resources:["deployments"] rule does not intercept deployments created via apps/v1 APIs. For that reason, webhooks should prefer registering for stable versions of resources.

This example shows a validating webhook that intercepts modifications to deployments (no matter the API group or version), and is always sent an apps/v1 Deployment object:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  matchPolicy: Equivalent
  rules:
  - operations: ["CREATE", "UPDATE", "DELETE"]
    apiGroups: ["apps"]
    apiVersions: ["v1"]
    resources: ["deployments"]
    scope: "Namespaced"
```

The matchPolicy for an admission webhooks defaults to Equivalent.

Matching requests: matchConditions

FEATURE STATE: Kubernetes v1.28 [beta]

You can define *match conditions* for webhooks if you need fine-grained request filtering. These conditions are useful if you find that match rules, objectSelectors and namespaceSelectors still doesn't provide the filtering you want over when to call out over HTTP. Match conditions are [CEL expressions](#). All match conditions must evaluate to true for the webhook to be called.

Here is an example illustrating a few different uses for match conditions:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  matchPolicy: Equivalent
  rules:
    - operations: ['CREATE','UPDATE']
      apiGroups: ['*']
      apiVersions: ['*']
      resources: ['*']
  failurePolicy: 'Ignore' # Fail-open (optional)
  sideEffects: None
  clientConfig:
    service:
      namespace: my-namespace
      name: my-webhook
      caBundle: '<omitted>'
# You can have up to 64 matchConditions per webhook
  matchConditions:
    - name: 'exclude-leases' # Each match condition must have a unique name
      expression: '!request.resource.group == "coordination.k8s.io" &&
request.resource.resource == "leases"' # Match non-lease resources.
    - name: 'exclude-kubelet-requests'
      expression: '!("system:nodes" in request.userInfo.groups)' # Match requests made by non-
node users.
    - name: 'rbac' # Skip RBAC requests, which are handled by the second webhook.
      expression: 'request.resource.group != "rbac.authorization.k8s.io"'
# This example illustrates the use of the 'authorizer'. The authorization check is more
expensive
# than a simple expression, so in this example it is scoped to only RBAC requests by using a
second
# webhook. Both webhooks can be served by the same endpoint.
- name: rbac.my-webhook.example.com
  matchPolicy: Equivalent
  rules:
    - operations: ['CREATE','UPDATE']
      apiGroups: ['rbac.authorization.k8s.io']
      apiVersions: ['*']
      resources: ['*']
```

```

failurePolicy: 'Fail' # Fail-closed (the default)
sideEffects: None
clientConfig:
  service:
    namespace: my-namespace
    name: my-webhook
    caBundle: '<omitted>'
# You can have up to 64 matchConditions per webhook
matchConditions:
  - name: 'breakglass'
    # Skip requests made by users authorized to 'breakglass' on this webhook.
    # The 'breakglass' API verb does not need to exist outside this check.
    expression: '!'
authorizer.group("admissionregistration.k8s.io").resource("validatingwebhookconfigurations").name("my-webhook.example.com").check("breakglass").allowed()

```

Note: You can define up to 64 elements in the matchConditions field per webhook.

Match conditions have access to the following CEL variables:

- object - The object from the incoming request. The value is null for DELETE requests. The object version may be converted based on the [matchPolicy](#).
- oldObject - The existing object. The value is null for CREATE requests.
- request - The request portion of the [AdmissionReview](#), excluding object and oldObject.
- authorizer - A CEL Authorizer. May be used to perform authorization checks for the principal (authenticated user) of the request. See [Authz](#) in the Kubernetes CEL library documentation for more details.
- authorizer.requestResource - A shortcut for an authorization check configured with the request resource (group, resource, (subresource), namespace, name).

For more information on CEL expressions, refer to the [Common Expression Language in Kubernetes reference](#).

In the event of an error evaluating a match condition the webhook is never called. Whether to reject the request is determined as follows:

1. If **any** match condition evaluated to false (regardless of other errors), the API server skips the webhook.
2. Otherwise:
 - for [failurePolicy: Fail](#), reject the request (without calling the webhook).
 - for [failurePolicy: Ignore](#), proceed with the request but skip the webhook.

Contacting the webhook

Once the API server has determined a request should be sent to a webhook, it needs to know how to contact the webhook. This is specified in the clientConfig stanza of the webhook configuration.

Webhooks can either be called via a URL or a service reference, and can optionally include a custom CA bundle to use to verify the TLS connection.

URL

url gives the location of the webhook, in standard URL form (scheme://host:port/path).

The host should not refer to a service running in the cluster; use a service reference by specifying the service field instead. The host might be resolved via external DNS in some API servers (e.g., kube-apiserver cannot resolve in-cluster DNS as that would be a layering violation). host may also be an IP address.

Please note that using localhost or 127.0.0.1 as a host is risky unless you take great care to run this webhook on all hosts which run an API server which might need to make calls to this webhook. Such installations are likely to be non-portable or not readily run in a new cluster.

The scheme must be "https"; the URL must begin with "https://".

Attempting to use a user or basic auth (for example user:password@) is not allowed. Fragments (#...) and query parameters (...) are also not allowed.

Here is an example of a mutating webhook configured to call a URL (and expects the TLS certificate to be verified using system trust roots, so does not specify a caBundle):

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  clientConfig:
    url: "https://my-webhook.example.com:9443/my-webhook-path"
```

Service reference

The service stanza inside clientConfig is a reference to the service for this webhook. If the webhook is running within the cluster, then you should use service instead of url. The service namespace and name are required. The port is optional and defaults to 443. The path is optional and defaults to "/".

Here is an example of a mutating webhook configured to call a service on port "1234" at the subpath "/my-path", and to verify the TLS connection against the ServerName my-service-name.my-service-namespace.svc using a custom CA bundle:

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  clientConfig:
    caBundle: <CA_BUNDLE>
    service:
      namespace: my-service-namespace
      name: my-service-name
      path: /my-path
      port: 1234
```

Note: You must replace the <CA_BUNDLE> in the above example by a valid CA bundle which is a PEM-encoded CA bundle for validating the webhook's server certificate.

Side effects

Webhooks typically operate only on the content of the AdmissionReview sent to them. Some webhooks, however, make out-of-band changes as part of processing admission requests.

Webhooks that make out-of-band changes ("side effects") must also have a reconciliation mechanism (like a controller) that periodically determines the actual state of the world, and adjusts the out-of-band data modified by the admission webhook to reflect reality. This is because a call to an admission webhook does not guarantee the admitted object will be persisted as is, or at all. Later webhooks can modify the content of the object, a conflict could be encountered while writing to storage, or the server could power off before persisting the object.

Additionally, webhooks with side effects must skip those side-effects when dryRun: true admission requests are handled. A webhook must explicitly indicate that it will not have side-effects when run with dryRun, or the dry-run request will not be sent to the webhook and the API request will fail instead.

Webhooks indicate whether they have side effects using the sideEffects field in the webhook configuration:

- None: calling the webhook will have no side effects.
- NoneOnDryRun: calling the webhook will possibly have side effects, but if a request with dryRun: true is sent to the webhook, the webhook will suppress the side effects (the webhook is dryRun-aware).

Here is an example of a validating webhook indicating it has no side effects on dryRun: true requests:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
  - name: my-webhook.example.com
    sideEffects: NoneOnDryRun
```

Timeouts

Because webhooks add to API request latency, they should evaluate as quickly as possible. timeoutSeconds allows configuring how long the API server should wait for a webhook to respond before treating the call as a failure.

If the timeout expires before the webhook responds, the webhook call will be ignored or the API call will be rejected based on the [failure policy](#).

The timeout value must be between 1 and 30 seconds.

Here is an example of a validating webhook with a custom timeout of 2 seconds:

```
apiVersion: admissionregistration.k8s.io/v1
kind: ValidatingWebhookConfiguration
webhooks:
  - name: my-webhook.example.com
    timeoutSeconds: 2
```

The timeout for an admission webhook defaults to 10 seconds.

Reinvocation policy

A single ordering of mutating admissions plugins (including webhooks) does not work for all cases (see <https://issue.k8s.io/64333> as an example). A mutating webhook can add a new sub-structure to the object (like adding a container to a pod), and other mutating plugins which have already run may have opinions on those new structures (like setting an `imagePullPolicy` on all containers).

To allow mutating admission plugins to observe changes made by other plugins, built-in mutating admission plugins are re-run if a mutating webhook modifies an object, and mutating webhooks can specify a `reinvocationPolicy` to control whether they are reinvoked as well.

`reinvocationPolicy` may be set to `Never` or `IfNeeded`. It defaults to `Never`.

- `Never`: the webhook must not be called more than once in a single admission evaluation.
- `IfNeeded`: the webhook may be called again as part of the admission evaluation if the object being admitted is modified by other admission plugins after the initial webhook call.

The important elements to note are:

- The number of additional invocations is not guaranteed to be exactly one.
- If additional invocations result in further modifications to the object, webhooks are not guaranteed to be invoked again.
- Webhooks that use this option may be reordered to minimize the number of additional invocations.
- To validate an object after all mutations are guaranteed complete, use a validating admission webhook instead (recommended for webhooks with side-effects).

Here is an example of a mutating webhook opting into being re-invoked if later admission plugins modify the object:

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  reinvocationPolicy: IfNeeded
```

Mutating webhooks must be [idempotent](#), able to successfully process an object they have already admitted and potentially modified. This is true for all mutating admission webhooks, since any change they can make in an object could already exist in the user-provided object, but it is essential for webhooks that opt into reinvocation.

Failure policy

`failurePolicy` defines how unrecognized errors and timeout errors from the admission webhook are handled. Allowed values are `Ignore` or `Fail`.

- `Ignore` means that an error calling the webhook is ignored and the API request is allowed to continue.
- `Fail` means that an error calling the webhook causes the admission to fail and the API request to be rejected.

Here is a mutating webhook configured to reject an API request if errors are encountered calling the admission webhook:

```
apiVersion: admissionregistration.k8s.io/v1
kind: MutatingWebhookConfiguration
webhooks:
- name: my-webhook.example.com
  failurePolicy: Fail
```

The default failurePolicy for an admission webhooks is Fail.

Monitoring admission webhooks

The API server provides ways to monitor admission webhook behaviors. These monitoring mechanisms help cluster admins to answer questions like:

1. Which mutating webhook mutated the object in a API request?
2. What change did the mutating webhook applied to the object?
3. Which webhooks are frequently rejecting API requests? What's the reason for a rejection?

Mutating webhook auditing annotations

Sometimes it's useful to know which mutating webhook mutated the object in a API request, and what change did the webhook apply.

The Kubernetes API server performs [auditing](#) on each mutating webhook invocation. Each invocation generates an auditing annotation capturing if a request object is mutated by the invocation, and optionally generates an annotation capturing the applied patch from the webhook admission response. The annotations are set in the audit event for given request on given stage of its execution, which is then pre-processed according to a certain policy and written to a backend.

The audit level of a event determines which annotations get recorded:

- At Metadata audit level or higher, an annotation with key mutation.webhook.admission.k8s.io/round_{round idx}_index_{order idx} gets logged with JSON payload indicating a webhook gets invoked for given request and whether it mutated the object or not.

For example, the following annotation gets recorded for a webhook being reinvoked. The webhook is ordered the third in the mutating webhook chain, and didn't mutated the request object during the invocation.

```
# the audit event recorded
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "annotations": {
    "mutation.webhook.admission.k8s.io/round_1_index_2": "{\"configuration\":\"my-mutating-webhook-configuration.example.com\",\"webhook\":\"my-
```

```

webhook.example.com\", \"mutated\": false}"
    # other annotations
    ...
}
# other fields
...
}

# the annotation value serialized
{
  "configuration": "my-mutating-webhook-configuration.example.com",
  "webhook": "my-webhook.example.com",
  "mutated": false
}

```

The following annotation gets recorded for a webhook being invoked in the first round. The webhook is ordered the first in the mutating webhook chain, and mutated the request object during the invocation.

```

# the audit event recorded
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "annotations": {
    "mutation.webhook.admission.k8s.io/round_0_index_0": "{\"configuration\":\"my-mutating-webhook-configuration.example.com\",\"webhook\":\"my-webhook-always-mutate.example.com\",\"mutated\": true}"
      # other annotations
      ...
    }
  # other fields
  ...
}

```

```

# the annotation value serialized
{
  "configuration": "my-mutating-webhook-configuration.example.com",
  "webhook": "my-webhook-always-mutate.example.com",
  "mutated": true
}

```

- At Request audit level or higher, an annotation with key patch.webhook.admission.k8s.io/round_{round idx}_index_{order idx} gets logged with JSON payload indicating a webhook gets invoked for given request and what patch gets applied to the request object.

For example, the following annotation gets recorded for a webhook being reinvoked. The webhook is ordered the fourth in the mutating webhook chain, and responded with a JSON patch which got applied to the request object.

```

# the audit event recorded
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",

```

```

"annotations": {
    "patch.webhook.admission.k8s.io/round_1_index_3": "{\"configuration\":\"my-other-
mutating-webhook-configuration.example.com\",\"webhook\":\"my-webhook-always-
mutate.example.com\",\"patch\":[{\"op\":\"add\",\"path\":\"/data/mutation-stage\",\"value\":
\"yes\"]},\"patchType\":\"JSONPatch\"}"
        # other annotations
        ...
    }
    # other fields
    ...
}
# the annotation value deserialized
{
    "configuration": "my-other-mutating-webhook-configuration.example.com",
    "webhook": "my-webhook-always-mutate.example.com",
    "patchType": "JSONPatch",
    "patch": [
        {
            "op": "add",
            "path": "/data/mutation-stage",
            "value": "yes"
        }
    ]
}

```

Admission webhook metrics

The API server exposes Prometheus metrics from the /metrics endpoint, which can be used for monitoring and diagnosing API server status. The following metrics record status related to admission webhooks.

API server admission webhook rejection count

Sometimes it's useful to know which admission webhooks are frequently rejecting API requests, and the reason for a rejection.

The API server exposes a Prometheus counter metric recording admission webhook rejections. The metrics are labelled to identify the causes of webhook rejection(s):

- name: the name of the webhook that rejected a request.
- operation: the operation type of the request, can be one of CREATE, UPDATE, DELETE and CONNECT.
- type: the admission webhook type, can be one of admit and validating.
- error_type: identifies if an error occurred during the webhook invocation that caused the rejection. Its value can be one of:
 - calling_webhook_error: unrecognized errors or timeout errors from the admission webhook happened and the webhook's [Failure policy](#) is set to Fail.

- no_error: no error occurred. The webhook rejected the request with allowed: false in the admission response. The metrics label rejection_code records the .status.code set in the admission response.
 - apiserver_internal_error: an API server internal error happened.
- rejection_code: the HTTP status code set in the admission response when a webhook rejected a request.

Example of the rejection count metrics:

```
# HELP apiserver_admission_webhook_rejection_count [ALPHA] Admission webhook
rejection count, identified by name and broken out for each admission type (validating or
admit) and operation. Additional labels specify an error type (calling_webhook_error or
apiserver_internal_error if an error occurred; no_error otherwise) and optionally a non-zero
rejection code if the webhook rejects the request with an HTTP status code (honored by the
apiserver when the code is greater or equal to 400). Codes greater than 600 are truncated to 600,
to keep the metrics cardinality bounded.
# TYPE apiserver_admission_webhook_rejection_count counter
apiserver_admission_webhook_rejection_count{error_type="calling_webhook_error",name="al
ways-timeout-
webhook.example.com",operation="CREATE",rejection_code="0",type="validating"} 1
apiserver_admission_webhook_rejection_count{error_type="calling_webhook_error",name="in
valid-admission-response-
webhook.example.com",operation="CREATE",rejection_code="0",type="validating"} 1
apiserver_admission_webhook_rejection_count{error_type="no_error",name="deny-unwanted-
configmap-data.example.com",operation="CREATE",rejection_code="400",type="validating"} 13
```

Best practices and warnings

Idempotence

An idempotent mutating admission webhook is able to successfully process an object it has already admitted and potentially modified. The admission can be applied multiple times without changing the result beyond the initial application.

Example of idempotent mutating admission webhooks:

1. For a CREATE pod request, set the field .spec.securityContext.runAsNonRoot of the pod to true, to enforce security best practices.
2. For a CREATE pod request, if the field .spec.containers[].resources.limits of a container is not set, set default resource limits.
3. For a CREATE pod request, inject a sidecar container with name foo-sidecar if no container with the name foo-sidecar already exists.

In the cases above, the webhook can be safely reinvoked, or admit an object that already has the fields set.

Example of non-idempotent mutating admission webhooks:

1. For a CREATE pod request, inject a sidecar container with name foo-sidecar suffixed with the current timestamp (e.g. foo-sidecar-19700101-000000).
2. For a CREATE/UPDATE pod request, reject if the pod has label "env" set, otherwise add an "env": "prod" label to the pod.
3. For a CREATE pod request, blindly append a sidecar container named foo-sidecar without looking to see if there is already a foo-sidecar container in the pod.

In the first case above, reinvoking the webhook can result in the same sidecar being injected multiple times to a pod, each time with a different container name. Similarly the webhook can inject duplicated containers if the sidecar already exists in a user-provided pod.

In the second case above, reinvoking the webhook will result in the webhook failing on its own output.

In the third case above, reinvoking the webhook will result in duplicated containers in the pod spec, which makes the request invalid and rejected by the API server.

Intercepting all versions of an object

It is recommended that admission webhooks should always intercept all versions of an object by setting `.webhooks[]`.`matchPolicy` to `Equivalent`. It is also recommended that admission webhooks should prefer registering for stable versions of resources. Failure to intercept all versions of an object can result in admission policies not being enforced for requests in certain versions. See [Matching requests: matchPolicy](#) for examples.

Availability

It is recommended that admission webhooks should evaluate as quickly as possible (typically in milliseconds), since they add to API request latency. It is encouraged to use a small timeout for webhooks. See [Timeouts](#) for more detail.

It is recommended that admission webhooks should leverage some format of load-balancing, to provide high availability and performance benefits. If a webhook is running within the cluster, you can run multiple webhook backends behind a service to leverage the load-balancing that service supports.

Guaranteeing the final state of the object is seen

Admission webhooks that need to guarantee they see the final state of the object in order to enforce policy should use a validating admission webhook, since objects can be modified after being seen by mutating webhooks.

For example, a mutating admission webhook is configured to inject a sidecar container with name "foo-sidecar" on every CREATE pod request. If the sidecar *must* be present, a validating admission webhook should also be configured to intercept CREATE pod requests, and validate that a container with name "foo-sidecar" with the expected configuration exists in the to-be-created object.

Avoiding deadlocks in self-hosted webhooks

A webhook running inside the cluster might cause deadlocks for its own deployment if it is configured to intercept resources required to start its own pods.

For example, a mutating admission webhook is configured to admit CREATE pod requests only if a certain label is set in the pod (e.g. "env": "prod"). The webhook server runs in a deployment which doesn't set the "env" label. When a node that runs the webhook server pods becomes unhealthy, the webhook deployment will try to reschedule the pods to another node. However the requests will get rejected by the existing webhook server since the "env" label is unset, and the migration cannot happen.

It is recommended to exclude the namespace where your webhook is running with a [namespaceSelector](#).

Side effects

It is recommended that admission webhooks should avoid side effects if possible, which means the webhooks operate only on the content of the AdmissionReview sent to them, and do not make out-of-band changes. The .webhooks[].sideEffects field should be set to None if a webhook doesn't have any side effect.

If side effects are required during the admission evaluation, they must be suppressed when processing an AdmissionReview object with dryRun set to true, and the .webhooks[].sideEffects field should be set to NoneOnDryRun. See [Side effects](#) for more detail.

Avoiding operating on the kube-system namespace

The kube-system namespace contains objects created by the Kubernetes system, e.g. service accounts for the control plane components, pods like kube-dns. Accidentally mutating or rejecting requests in the kube-system namespace may cause the control plane components to stop functioning or introduce unknown behavior. If your admission webhooks don't intend to modify the behavior of the Kubernetes control plane, exclude the kube-system namespace from being intercepted using a [namespaceSelector](#).

Managing Service Accounts

A *ServiceAccount* provides an identity for processes that run in a Pod.

A process inside a Pod can use the identity of its associated service account to authenticate to the cluster's API server.

For an introduction to service accounts, read [configure service accounts](#).

This task guide explains some of the concepts behind ServiceAccounts. The guide also explains how to obtain or revoke tokens that represent ServiceAccounts.

Before you begin

You need to have a Kubernetes cluster, and the kubectl command-line tool must be configured to communicate with your cluster. It is recommended to run this tutorial on a cluster with at

least two nodes that are not acting as control plane hosts. If you do not already have a cluster, you can create one by using [minikube](#) or you can use one of these Kubernetes playgrounds:

- [Killercoda](#)
- [Play with Kubernetes](#)

To be able to follow these steps exactly, ensure you have a namespace named `exampelens`. If you don't, create one by running:

```
kubectl create namespace exampelens
```

User accounts versus service accounts

Kubernetes distinguishes between the concept of a user account and a service account for a number of reasons:

- User accounts are for humans. Service accounts are for application processes, which (for Kubernetes) run in containers that are part of pods.
- User accounts are intended to be global: names must be unique across all namespaces of a cluster. No matter what namespace you look at, a particular username that represents a user represents the same user. In Kubernetes, service accounts are namespaced: two different namespaces can contain `ServiceAccounts` that have identical names.
- Typically, a cluster's user accounts might be synchronised from a corporate database, where new user account creation requires special privileges and is tied to complex business processes. By contrast, service account creation is intended to be more lightweight, allowing cluster users to create service accounts for specific tasks on demand. Separating `ServiceAccount` creation from the steps to onboard human users makes it easier for workloads to follow the principle of least privilege.
- Auditing considerations for humans and service accounts may differ; the separation makes that easier to achieve.
- A configuration bundle for a complex system may include definition of various service accounts for components of that system. Because service accounts can be created without many constraints and have namespaced names, such configuration is usually portable.

Bound service account token volume mechanism

FEATURE STATE: Kubernetes v1.22 [stable]

By default, the Kubernetes control plane (specifically, the [ServiceAccount admission controller](#)) adds a [projected volume](#) to Pods, and this volume includes a token for Kubernetes API access.

Here's an example of how that looks for a launched Pod:

```
...
- name: kube-api-access-<random-suffix>
  projected:
    sources:
      - serviceAccountToken:
          path: token # must match the path the app expects
      - configMap:
          items:
            - key: ca.crt
```

```
    path: ca.crt
    name: kube-root-ca.crt
  - downwardAPI:
    items:
      - fieldRef:
          apiVersion: v1
          fieldPath: metadata.namespace
        path: namespace
```

That manifest snippet defines a projected volume that consists of three sources. In this case, each source also represents a single path within that volume. The three sources are:

1. A serviceAccountToken source, that contains a token that the kubelet acquires from kube-apiserver. The kubelet fetches time-bound tokens using the TokenRequest API. A token served for a TokenRequest expires either when the pod is deleted or after a defined lifespan (by default, that is 1 hour). The kubelet also refreshes that token before the token expires. The token is bound to the specific Pod and has the kube-apiserver as its audience. This mechanism superseded an earlier mechanism that added a volume based on a Secret, where the Secret represented the ServiceAccount for the Pod, but did not expire.
2. A configMap source. The ConfigMap contains a bundle of certificate authority data. Pods can use these certificates to make sure that they are connecting to your cluster's kube-apiserver (and not to middlebox or an accidentally misconfigured peer).
3. A downwardAPI source that looks up the name of the namespace containing the Pod, and makes that name information available to application code running inside the Pod.

Any container within the Pod that mounts this particular volume can access the above information.

Note: There is no specific mechanism to invalidate a token issued via TokenRequest. If you no longer trust a bound service account token for a Pod, you can delete that Pod. Deleting a Pod expires its bound service account tokens.

Manual Secret management for ServiceAccounts

Versions of Kubernetes before v1.22 automatically created credentials for accessing the Kubernetes API. This older mechanism was based on creating token Secrets that could then be mounted into running Pods.

In more recent versions, including Kubernetes v1.28, API credentials are [obtained directly](#) using the [TokenRequest](#) API, and are mounted into Pods using a projected volume. The tokens obtained using this method have bounded lifetimes, and are automatically invalidated when the Pod they are mounted into is deleted.

You can still [manually create](#) a Secret to hold a service account token; for example, if you need a token that never expires.

Once you manually create a Secret and link it to a ServiceAccount, the Kubernetes control plane automatically populates the token into that Secret.

Note: Although the manual mechanism for creating a long-lived ServiceAccount token exists, using [TokenRequest](#) to obtain short-lived API access tokens is recommended instead.

Control plane details

ServiceAccount controller

A ServiceAccount controller manages the ServiceAccounts inside namespaces, and ensures a ServiceAccount named "default" exists in every active namespace.

Token controller

The service account token controller runs as part of kube-controller-manager. This controller acts asynchronously. It:

- watches for ServiceAccount deletion and deletes all corresponding ServiceAccount token Secrets.
- watches for ServiceAccount token Secret addition, and ensures the referenced ServiceAccount exists, and adds a token to the Secret if needed.
- watches for Secret deletion and removes a reference from the corresponding ServiceAccount if needed.

You must pass a service account private key file to the token controller in the kube-controller-manager using the --service-account-private-key-file flag. The private key is used to sign generated service account tokens. Similarly, you must pass the corresponding public key to the kube-apiserver using the --service-account-key-file flag. The public key will be used to verify the tokens during authentication.

ServiceAccount admission controller

The modification of pods is implemented via a plugin called an [Admission Controller](#). It is part of the API server. This admission controller acts synchronously to modify pods as they are created. When this plugin is active (and it is by default on most distributions), then it does the following when a Pod is created:

1. If the pod does not have a .spec.serviceAccountName set, the admission controller sets the name of the ServiceAccount for this incoming Pod to default.
2. The admission controller ensures that the ServiceAccount referenced by the incoming Pod exists. If there is no ServiceAccount with a matching name, the admission controller rejects the incoming Pod. That check applies even for the default ServiceAccount.
3. Provided that neither the ServiceAccount's automountServiceAccountToken field nor the Pod's automountServiceAccountToken field is set to false:
 - the admission controller mutates the incoming Pod, adding an extra [volume](#) that contains a token for API access.
 - the admission controller adds a volumeMount to each container in the Pod, skipping any containers that already have a volume mount defined for the path /var/run/secrets/kubernetes.io/serviceaccount. For Linux containers, that volume is mounted at /var/run/secrets/kubernetes.io/serviceaccount; on Windows nodes, the mount is at the equivalent path.
4. If the spec of the incoming Pod doesn't already contain any imagePullSecrets, then the admission controller adds imagePullSecrets, copying them from the ServiceAccount.

TokenRequest API

FEATURE STATE: Kubernetes v1.22 [stable]

You use the [TokenRequest](#) subresource of a ServiceAccount to obtain a time-bound token for that ServiceAccount. You don't need to call this to obtain an API token for use within a container, since the kubelet sets this up for you using a *projected volume*.

If you want to use the TokenRequest API from kubectl, see [Manually create an API token for a ServiceAccount](#).

The Kubernetes control plane (specifically, the ServiceAccount admission controller) adds a projected volume to Pods, and the kubelet ensures that this volume contains a token that lets containers authenticate as the right ServiceAccount.

(This mechanism superseded an earlier mechanism that added a volume based on a Secret, where the Secret represented the ServiceAccount for the Pod but did not expire.)

Here's an example of how that looks for a launched Pod:

```
...
- name: kube-api-access-<random-suffix>
  projected:
    defaultMode: 420 # decimal equivalent of octal 0644
    sources:
      - serviceAccountToken:
          expirationSeconds: 3607
          path: token
      - configMap:
          items:
            - key: ca.crt
              path: ca.crt
            name: kube-root-ca.crt
      - downwardAPI:
          items:
            - fieldRef:
                apiVersion: v1
                fieldPath: metadata.namespace
                path: namespace
```

That manifest snippet defines a projected volume that combines information from three sources:

1. A serviceAccountToken source, that contains a token that the kubelet acquires from kube-apiserver. The kubelet fetches time-bound tokens using the TokenRequest API. A token served for a TokenRequest expires either when the pod is deleted or after a defined lifespan (by default, that is 1 hour). The token is bound to the specific Pod and has the kube-apiserver as its audience.
2. A configMap source. The ConfigMap contains a bundle of certificate authority data. Pods can use these certificates to make sure that they are connecting to your cluster's kube-apiserver (and not to middlebox or an accidentally misconfigured peer).
3. A downwardAPI source. This downwardAPI volume makes the name of the namespace containing the Pod available to application code running inside the Pod.

Any container within the Pod that mounts this volume can access the above information.

Create additional API tokens

Caution: Only create long-lived API tokens if the [token request](#) mechanism is not suitable. The token request mechanism provides time-limited tokens; because these expire, they represent a lower risk to information security.

To create a non-expiring, persisted API token for a ServiceAccount, create a Secret of type kubernetes.io/service-account-token with an annotation referencing the ServiceAccount. The control plane then generates a long-lived token and updates that Secret with that generated token data.

Here is a sample manifest for such a Secret:

[secret/serviceaccount/mysecretname.yaml](#)

```
apiVersion: v1
kind: Secret
type: kubernetes.io/service-account-token
metadata:
  name: mysecretname
  annotations:
    kubernetes.io/service-account.name: myserviceaccount
```

To create a Secret based on this example, run:

```
kubectl -n exemplens create -f https://k8s.io/examples/secret/serviceaccount/
mysecretname.yaml
```

To see the details for that Secret, run:

```
kubectl -n exemplens describe secret mysecretname
```

The output is similar to:

```
Name:      mysecretname
Namespace: exemplens
Labels:    <none>
Annotations:  kubernetes.io/service-account.name=myserviceaccount
              kubernetes.io/service-account.uid=8a85c4c4-8483-11e9-bc42-526af7764f64

Type:  kubernetes.io/service-account-token

Data
=====
ca.crt:   1362 bytes
namespace: 9 bytes
token:    ...
```

If you launch a new Pod into the exemplens namespace, it can use the myserviceaccount service-account-token Secret that you just created.

Delete/invalidate a ServiceAccount token

If you know the name of the Secret that contains the token you want to remove:

```
kubectl delete secret name-of-secret
```

Otherwise, first find the Secret for the ServiceAccount.

```
# This assumes that you already have a namespace named 'examplens'  
kubectl -n examplens get serviceaccount/example-automated-thing -o yaml
```

The output is similar to:

```
apiVersion: v1  
kind: ServiceAccount  
metadata:  
  annotations:  
    kubectl.kubernetes.io/last-applied-configuration: |  
      {"apiVersion":"v1","kind":"ServiceAccount","metadata":{"annotations":{},"name":"example-  
automated-thing","namespace":"examplens"}}  
  creationTimestamp: "2019-07-21T07:07:07Z"  
  name: example-automated-thing  
  namespace: examplens  
  resourceVersion: "777"  
  selfLink: /api/v1/namespaces/examplens/serviceaccounts/example-automated-thing  
  uid: f23fd170-66f2-4697-b049-e1e266b7f835  
secrets:  
  - name: example-automated-thing-token-zyxwv
```

Then, delete the Secret you now know the name of:

```
kubectl -n examplens delete secret/example-automated-thing-token-zyxwv
```

Clean up

If you created a namespace `examplens` to experiment with, you can remove it:

```
kubectl delete namespace examplens
```

What's next

- Read more details about [projected volumes](#).

Authorization Overview

Learn more about Kubernetes authorization, including details about creating policies using the supported authorization modules.

In Kubernetes, you must be authenticated (logged in) before your request can be authorized (granted permission to access). For information about authentication, see [Controlling Access to the Kubernetes API](#).

Kubernetes expects attributes that are common to REST API requests. This means that Kubernetes authorization works with existing organization-wide or cloud-provider-wide access control systems which may handle other APIs besides the Kubernetes API.

Determine Whether a Request is Allowed or Denied

Kubernetes authorizes API requests using the API server. It evaluates all of the request attributes against all policies and allows or denies the request. All parts of an API request must be allowed by some policy in order to proceed. This means that permissions are denied by default.

(Although Kubernetes uses the API server, access controls and policies that depend on specific fields of specific kinds of objects are handled by Admission Controllers.)

When multiple authorization modules are configured, each is checked in sequence. If any authorizer approves or denies a request, that decision is immediately returned and no other authorizer is consulted. If all modules have no opinion on the request, then the request is denied. A deny returns an HTTP status code 403.

Review Your Request Attributes

Kubernetes reviews only the following API request attributes:

- **user** - The user string provided during authentication.
- **group** - The list of group names to which the authenticated user belongs.
- **extra** - A map of arbitrary string keys to string values, provided by the authentication layer.
- **API** - Indicates whether the request is for an API resource.
- **Request path** - Path to miscellaneous non-resource endpoints like /api or /healthz.
- **API request verb** - API verbs like get, list, create, update, patch, watch, delete, and deletecollection are used for resource requests. To determine the request verb for a resource API endpoint, see [Determine the request verb](#).
- **HTTP request verb** - Lowercased HTTP methods like get, post, put, and delete are used for non-resource requests.
- **Resource** - The ID or name of the resource that is being accessed (for resource requests only) -- For resource requests using get, update, patch, and delete verbs, you must provide the resource name.
- **Subresource** - The subresource that is being accessed (for resource requests only).
- **Namespace** - The namespace of the object that is being accessed (for namespaced resource requests only).
- **API group** - The [API Group](#) being accessed (for resource requests only). An empty string designates the [core API group](#).

Determine the Request Verb

Non-resource requests Requests to endpoints other than /api/v1/... or /apis/<group>/<version>/... are considered "non-resource requests", and use the lower-cased HTTP method of the request as the verb. For example, a GET request to endpoints like /api or /healthz would use get as the verb.

Resource requests To determine the request verb for a resource API endpoint, review the HTTP verb used and whether or not the request acts on an individual resource or a collection of resources:

HTTP verb	request verb
POST	create
GET, HEAD	get (for individual resources), list (for collections, including full object content), watch (for watching an individual resource or collection of resources)
PUT	update
PATCH	patch
DELETE	delete (for individual resources), deletecollection (for collections)

Caution: The get, list and watch verbs can all return the full details of a resource. In terms of the returned data they are equivalent. For example, list on secrets will still reveal the data attributes of any returned resources.

Kubernetes sometimes checks authorization for additional permissions using specialized verbs. For example:

- [RBAC](#)
 - bind and escalate verbs on roles and clusterroles resources in the rbac.authorization.k8s.io API group.
- [Authentication](#)
 - impersonate verb on users, groups, and serviceaccounts in the core API group, and the userextras in the authentication.k8s.io API group.

Authorization Modes

The Kubernetes API server may authorize a request using one of several authorization modes:

- **Node** - A special-purpose authorization mode that grants permissions to kubelets based on the pods they are scheduled to run. To learn more about using the Node authorization mode, see [Node Authorization](#).
- **ABAC** - Attribute-based access control (ABAC) defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together. The policies can use any type of attributes (user attributes, resource attributes, object, environment attributes, etc). To learn more about using the ABAC mode, see [ABAC Mode](#).
- **RBAC** - Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise. In this context, access is the ability of an individual user to perform a specific task, such as view, create, or modify a file. To learn more about using the RBAC mode, see [RBAC Mode](#)
 - When specified RBAC (Role-Based Access Control) uses the rbac.authorization.k8s.io API group to drive authorization decisions, allowing admins to dynamically configure permission policies through the Kubernetes API.
 - To enable RBAC, start the apiserver with --authorization-mode=RBAC.
- **Webhook** - A WebHook is an HTTP callback: an HTTP POST that occurs when something happens; a simple event-notification via HTTP POST. A web application implementing WebHooks will POST a message to a URL when certain things happen. To learn more about using the Webhook mode, see [Webhook Mode](#).

Checking API Access

kubectl provides the auth can-i subcommand for quickly querying the API authorization layer. The command uses the SelfSubjectAccessReview API to determine if the current user can perform a given action, and works regardless of the authorization mode used.

```
kubectl auth can-i create deployments --namespace dev
```

The output is similar to this:

```
yes
```

```
kubectl auth can-i create deployments --namespace prod
```

The output is similar to this:

```
no
```

Administrators can combine this with [user impersonation](#) to determine what action other users can perform.

```
kubectl auth can-i list secrets --namespace dev --as dave
```

The output is similar to this:

```
no
```

Similarly, to check whether a ServiceAccount named dev-sa in Namespace dev can list Pods in the Namespace target:

```
kubectl auth can-i list pods \  
  --namespace target \  
  --as system:serviceaccount:dev:dev-sa
```

The output is similar to this:

```
yes
```

SelfSubjectAccessReview is part of the authorization.k8s.io API group, which exposes the API server authorization to external services. Other resources in this group include:

- SubjectAccessReview - Access review for any user, not only the current one. Useful for delegating authorization decisions to the API server. For example, the kubelet and extension API servers use this to determine user access to their own APIs.
- LocalSubjectAccessReview - Like SubjectAccessReview but restricted to a specific namespace.
- SelfSubjectRulesReview - A review which returns the set of actions a user can perform within a namespace. Useful for users to quickly summarize their own access, or for UIs to hide/show actions.

These APIs can be queried by creating normal Kubernetes resources, where the response "status" field of the returned object is the result of the query.

```
kubectl create -f - -o yaml << EOF  
apiVersion: authorization.k8s.io/v1
```

```
kind: SelfSubjectAccessReview
spec:
  resourceAttributes:
    group: apps
    resource: deployments
    verb: create
    namespace: dev
EOF
```

The generated SelfSubjectAccessReview is:

```
apiVersion: authorization.k8s.io/v1
kind: SelfSubjectAccessReview
metadata:
  creationTimestamp: null
spec:
  resourceAttributes:
    group: apps
    resource: deployments
    namespace: dev
    verb: create
status:
  allowed: true
  denied: false
```

Using Flags for Your Authorization Module

You must include a flag in your policy to indicate which authorization module your policies include:

The following flags can be used:

- `--authorization-mode=ABAC` Attribute-Based Access Control (ABAC) mode allows you to configure policies using local files.
- `--authorization-mode=RBAC` Role-based access control (RBAC) mode allows you to create and store policies using the Kubernetes API.
- `--authorization-mode=Webhook` WebHook is an HTTP callback mode that allows you to manage authorization using a remote REST endpoint.
- `--authorization-mode=Node` Node authorization is a special-purpose authorization mode that specifically authorizes API requests made by kubelets.
- `--authorization-mode=AlwaysDeny` This flag blocks all requests. Use this flag only for testing.
- `--authorization-mode=AlwaysAllow` This flag allows all requests. Use this flag only if you do not require authorization for your API requests.

You can choose more than one authorization module. Modules are checked in order so an earlier module has higher priority to allow or deny a request.

Privilege escalation via workload creation or edits

Users who can create/edit pods in a namespace, either directly or through a [controller](#) such as an operator, could escalate their privileges in that namespace.

Caution: System administrators, use care when granting access to create or edit workloads. Details of how these can be misused are documented in [escalation paths](#)

Escalation paths

- Mounting arbitrary secrets in that namespace
 - Can be used to access secrets meant for other workloads
 - Can be used to obtain a more privileged service account's service account token
- Using arbitrary Service Accounts in that namespace
 - Can perform Kubernetes API actions as another workload (impersonation)
 - Can perform any privileged actions that Service Account has
- Mounting configmaps meant for other workloads in that namespace
 - Can be used to obtain information meant for other workloads, such as DB host names.
- Mounting volumes meant for other workloads in that namespace
 - Can be used to obtain information meant for other workloads, and change it.

Caution: System administrators should be cautious when deploying CRDs that change the above areas. These may open privilege escalation paths. This should be considered when deciding on your RBAC controls.

What's next

- To learn more about Authentication, see [Authentication](#) in [Controlling Access to the Kubernetes API](#).
- To learn more about Admission Control, see [Using Admission Controllers](#).

Using RBAC Authorization

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within your organization.

RBAC authorization uses the rbac.authorization.k8s.io [API group](#) to drive authorization decisions, allowing you to dynamically configure policies through the Kubernetes API.

To enable RBAC, start the [API server](#) with the --authorization-mode flag set to a comma-separated list that includes RBAC; for example:

```
kube-apiserver --authorization-mode=Example,RBAC --other-options --more-options
```

API objects

The RBAC API declares four kinds of Kubernetes object: *Role*, *ClusterRole*, *RoleBinding* and *ClusterRoleBinding*. You can describe or amend the RBAC [objects](#) using tools such as kubectl, just like any other Kubernetes object.

Caution: These objects, by design, impose access restrictions. If you are making changes to a cluster as you learn, see [privilege escalation prevention and bootstrapping](#) to understand how those restrictions can prevent you making some changes.

Role and ClusterRole

An RBAC *Role* or *ClusterRole* contains rules that represent a set of permissions. Permissions are purely additive (there are no "deny" rules).

A Role always sets permissions within a particular [namespace](#); when you create a Role, you have to specify the namespace it belongs in.

ClusterRole, by contrast, is a non-namespaced resource. The resources have different names (Role and ClusterRole) because a Kubernetes object always has to be either namespaced or not namespaced; it can't be both.

ClusterRoles have several uses. You can use a ClusterRole to:

1. define permissions on namespaced resources and be granted access within individual namespace(s)
2. define permissions on namespaced resources and be granted access across all namespaces
3. define permissions on cluster-scoped resources

If you want to define a role within a namespace, use a Role; if you want to define a role cluster-wide, use a ClusterRole.

Role example

Here's an example Role in the "default" namespace that can be used to grant read access to [pods](#):

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

ClusterRole example

A ClusterRole can be used to grant the same permissions as a Role. Because ClusterRoles are cluster-scoped, you can also use them to grant access to:

- cluster-scoped resources (like [nodes](#))
- non-resource endpoints (like [/healthz](#))
- namespaced resources (like Pods), across all namespaces

For example: you can use a ClusterRole to allow a particular user to run `kubectl get pods --all-namespaces`

Here is an example of a ClusterRole that can be used to grant read access to [secrets](#) in any particular namespace, or across all namespaces (depending on how it is [bound](#)):

```

apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  # "namespace" omitted since ClusterRoles are not namespaced
  name: secret-reader
rules:
- apiGroups: [""]
  #
  # at the HTTP level, the name of the resource for accessing Secret
  # objects is "secrets"
  resources: ["secrets"]
  verbs: ["get", "watch", "list"]

```

The name of a Role or a ClusterRole object must be a valid [path segment name](#).

RoleBinding and ClusterRoleBinding

A role binding grants the permissions defined in a role to a user or set of users. It holds a list of *subjects* (users, groups, or service accounts), and a reference to the role being granted. A RoleBinding grants permissions within a specific namespace whereas a ClusterRoleBinding grants that access cluster-wide.

A RoleBinding may reference any Role in the same namespace. Alternatively, a RoleBinding can reference a ClusterRole and bind that ClusterRole to the namespace of the RoleBinding. If you want to bind a ClusterRole to all the namespaces in your cluster, you use a ClusterRoleBinding.

The name of a RoleBinding or ClusterRoleBinding object must be a valid [path segment name](#).

RoleBinding examples

Here is an example of a RoleBinding that grants the "pod-reader" Role to the user "jane" within the "default" namespace. This allows "jane" to read pods in the "default" namespace.

```

apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "jane" to read pods in the "default" namespace.
# You need to already have a Role named "pod-reader" in that namespace.
kind: RoleBinding
metadata:
  name: read-pods
  namespace: default
subjects:
# You can specify more than one "subject"
- kind: User
  name: jane # "name" is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  # "roleRef" specifies the binding to a Role / ClusterRole
  kind: Role #this must be Role or ClusterRole
  name: pod-reader # this must match the name of the Role or ClusterRole you wish to bind to
  apiGroup: rbac.authorization.k8s.io

```

A RoleBinding can also reference a ClusterRole to grant the permissions defined in that ClusterRole to resources inside the RoleBinding's namespace. This kind of reference lets you define a set of common roles across your cluster, then reuse them within multiple namespaces.

For instance, even though the following RoleBinding refers to a ClusterRole, "dave" (the subject, case sensitive) will only be able to read Secrets in the "development" namespace, because the RoleBinding's namespace (in its metadata) is "development".

```
apiVersion: rbac.authorization.k8s.io/v1
# This role binding allows "dave" to read secrets in the "development" namespace.
# You need to already have a ClusterRole named "secret-reader".
kind: RoleBinding
metadata:
  name: read-secrets
  #
  # The namespace of the RoleBinding determines where the permissions are granted.
  # This only grants permissions within the "development" namespace.
  namespace: development
subjects:
- kind: User
  name: dave # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

ClusterRoleBinding example

To grant permissions across a whole cluster, you can use a ClusterRoleBinding. The following ClusterRoleBinding allows any user in the group "manager" to read secrets in any namespace.

```
apiVersion: rbac.authorization.k8s.io/v1
# This cluster role binding allows anyone in the "manager" group to read secrets in any
namespace.
kind: ClusterRoleBinding
metadata:
  name: read-secrets-global
subjects:
- kind: Group
  name: manager # Name is case sensitive
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: secret-reader
  apiGroup: rbac.authorization.k8s.io
```

After you create a binding, you cannot change the Role or ClusterRole that it refers to. If you try to change a binding's roleRef, you get a validation error. If you do want to change the roleRef for a binding, you need to remove the binding object and create a replacement.

There are two reasons for this restriction:

1. Making roleRef immutable allows granting someone update permission on an existing binding object, so that they can manage the list of subjects, without being able to change the role that is granted to those subjects.
2. A binding to a different role is a fundamentally different binding. Requiring a binding to be deleted/recreated in order to change the roleRef ensures the full list of subjects in the binding is intended to be granted the new role (as opposed to enabling or accidentally modifying only the roleRef without verifying all of the existing subjects should be given the new role's permissions).

The kubectl auth reconcile command-line utility creates or updates a manifest file containing RBAC objects, and handles deleting and recreating binding objects if required to change the role they refer to. See [command usage and examples](#) for more information.

Referring to resources

In the Kubernetes API, most resources are represented and accessed using a string representation of their object name, such as pods for a Pod. RBAC refers to resources using exactly the same name that appears in the URL for the relevant API endpoint. Some Kubernetes APIs involve a *subresource*, such as the logs for a Pod. A request for a Pod's logs looks like:

```
GET /api/v1/namespaces/{namespace}/pods/{name}/log
```

In this case, pods is the namespaced resource for Pod resources, and log is a subresource of pods. To represent this in an RBAC role, use a slash (/) to delimit the resource and subresource. To allow a subject to read pods and also access the log subresource for each of those Pods, you write:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-and-pod-logs-reader
rules:
- apiGroups: []
  resources: ["pods", "pods/log"]
  verbs: ["get", "list"]
```

You can also refer to resources by name for certain requests through the resourceNames list. When specified, requests can be restricted to individual instances of a resource. Here is an example that restricts its subject to only get or update a [ConfigMap](#) named my-configmap:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: configmap-updater
rules:
- apiGroups: []
  # at the HTTP level, the name of the resource for accessing ConfigMap
  # objects is "configmaps"
```

```
resources: ["configmaps"]
resourceNames: ["my-configmap"]
verbs: ["update", "get"]
```

Note: You cannot restrict create or deletecollection requests by their resource name. For create, this limitation is because the name of the new object may not be known at authorization time. If you restrict list or watch by resourceName, clients must include a metadata.name field selector in their list or watch request that matches the specified resourceName in order to be authorized. For example, kubectl get configmaps --field-selector=metadata.name=my-configmap

Rather than referring to individual resources, apiGroups, and verbs, you can use the wildcard * symbol to refer to all such objects. For nonResourceURLs, you can use the wildcard * as a suffix glob match. For resourceNames, an empty set means that everything is allowed. Here is an example that allows access to perform any current and future action on all current and future resources in the example.com API group. This is similar to the built-in cluster-admin role.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: example.com-superuser # DO NOT USE THIS ROLE, IT IS JUST AN EXAMPLE
rules:
- apiGroups: ["example.com"]
  resources: ["*"]
  verbs: ["*"]
```

Caution: Using wildcards in resource and verb entries could result in overly permissive access being granted to sensitive resources. For instance, if a new resource type is added, or a new subresource is added, or a new custom verb is checked, the wildcard entry automatically grants access, which may be undesirable. The [principle of least privilege](#) should be employed, using specific resources and verbs to ensure only the permissions required for the workload to function correctly are applied.

Aggregated ClusterRoles

You can *aggregate* several ClusterRoles into one combined ClusterRole. A controller, running as part of the cluster control plane, watches for ClusterRole objects with an aggregationRule set. The aggregationRule defines a label [selector](#) that the controller uses to match other ClusterRole objects that should be combined into the rules field of this one.

Caution: The control plane overwrites any values that you manually specify in the rules field of an aggregate ClusterRole. If you want to change or add rules, do so in the ClusterRole objects that are selected by the aggregationRule.

Here is an example aggregated ClusterRole:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring
aggregationRule:
  clusterRoleSelectors:
  - matchLabels:
```

```
rbac.example.com/aggregate-to-monitoring: "true"
rules: [] # The control plane automatically fills in the rules
```

If you create a new ClusterRole that matches the label selector of an existing aggregated ClusterRole, that change triggers adding the new rules into the aggregated ClusterRole. Here is an example that adds rules to the "monitoring" ClusterRole, by creating another ClusterRole labeled rbac.example.com/aggregate-to-monitoring: true.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: monitoring-endpoints
  labels:
    rbac.example.com/aggregate-to-monitoring: "true"
# When you create the "monitoring-endpoints" ClusterRole,
# the rules below will be added to the "monitoring" ClusterRole.
rules:
- apiGroups: []
  resources: ["services", "endpointslices", "pods"]
  verbs: ["get", "list", "watch"]
```

The [default user-facing roles](#) use ClusterRole aggregation. This lets you, as a cluster administrator, include rules for custom resources, such as those served by [CustomResourceDefinitions](#) or aggregated API servers, to extend the default roles.

For example: the following ClusterRoles let the "admin" and "edit" default roles manage the custom resource named CronTab, whereas the "view" role can perform only read actions on CronTab resources. You can assume that CronTab objects are named "crontabs" in URLs as seen by the API server.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: aggregate-cron-tabs-edit
  labels:
    # Add these permissions to the "admin" and "edit" default roles.
    rbac.authorization.k8s.io/aggregate-to-admin: "true"
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
rules:
- apiGroups: ["stable.example.com"]
  resources: ["crontabs"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
---
kind: ClusterRole
apiVersion: rbac.authorization.k8s.io/v1
metadata:
  name: aggregate-cron-tabs-view
  labels:
    # Add these permissions to the "view" default role.
    rbac.authorization.k8s.io/aggregate-to-view: "true"
rules:
- apiGroups: ["stable.example.com"]
```

```
resources: ["crontabs"]
verbs: ["get", "list", "watch"]
```

Role examples

The following examples are excerpts from Role or ClusterRole objects, showing only the rules section.

Allow reading "pods" resources in the core [API Group](#):

```
rules:
- apiGroups: [""]
  #
  # at the HTTP level, the name of the resource for accessing Pod
  # objects is "pods"
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
```

Allow reading/writing Deployments (at the HTTP level: objects with "deployments" in the resource part of their URL) in the "apps" API groups:

```
rules:
- apiGroups: ["apps"]
  #
  # at the HTTP level, the name of the resource for accessing Deployment
  # objects is "deployments"
  resources: ["deployments"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Allow reading Pods in the core API group, as well as reading or writing Job resources in the "batch" API group:

```
rules:
- apiGroups: [""]
  #
  # at the HTTP level, the name of the resource for accessing Pod
  # objects is "pods"
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["batch"]
  #
  # at the HTTP level, the name of the resource for accessing Job
  # objects is "jobs"
  resources: ["jobs"]
  verbs: ["get", "list", "watch", "create", "update", "patch", "delete"]
```

Allow reading a ConfigMap named "my-config" (must be bound with a RoleBinding to limit to a single ConfigMap in a single namespace):

```
rules:
- apiGroups: [""]
  #
  # at the HTTP level, the name of the resource for accessing ConfigMap
```

```
# objects is "configmaps"
resources: ["configmaps"]
resourceNames: ["my-config"]
verbs: ["get"]
```

Allow reading the resource "nodes" in the core group (because a Node is cluster-scoped, this must be in a ClusterRole bound with a ClusterRoleBinding to be effective):

```
rules:
- apiGroups: [""]
#
# at the HTTP level, the name of the resource for accessing Node
# objects is "nodes"
resources: ["nodes"]
verbs: ["get", "list", "watch"]
```

Allow GET and POST requests to the non-resource endpoint /healthz and all subpaths (must be in a ClusterRole bound with a ClusterRoleBinding to be effective):

```
rules:
- nonResourceURLs: ["/healthz", "/healthz/*"] # '*' in a nonResourceURL is a suffix glob match
  verbs: ["get", "post"]
```

Referring to subjects

A RoleBinding or ClusterRoleBinding binds a role to subjects. Subjects can be groups, users or [ServiceAccounts](#).

Kubernetes represents usernames as strings. These can be: plain names, such as "alice"; email-style names, like "bob@example.com"; or numeric user IDs represented as a string. It is up to you as a cluster administrator to configure the [authentication modules](#) so that authentication produces usernames in the format you want.

Caution: The prefix system: is reserved for Kubernetes system use, so you should ensure that you don't have users or groups with names that start with system: by accident. Other than this special prefix, the RBAC authorization system does not require any format for usernames.

In Kubernetes, Authenticator modules provide group information. Groups, like users, are represented as strings, and that string has no format requirements, other than that the prefix system: is reserved.

[ServiceAccounts](#) have names prefixed with system:serviceaccount:, and belong to groups that have names prefixed with system:serviceaccounts:.

Note:

- system:serviceaccount: (singular) is the prefix for service account usernames.
- system:serviceaccounts: (plural) is the prefix for service account groups.

RoleBinding examples

The following examples are RoleBinding excerpts that only show the subjects section.

For a user named alice@example.com:

```
subjects:  
- kind: User  
  name: "alice@example.com"  
  apiGroup: rbac.authorization.k8s.io
```

For a group named frontend-admins:

```
subjects:  
- kind: Group  
  name: "frontend-admins"  
  apiGroup: rbac.authorization.k8s.io
```

For the default service account in the "kube-system" namespace:

```
subjects:  
- kind: ServiceAccount  
  name: default  
  namespace: kube-system
```

For all service accounts in the "qa" namespace:

```
subjects:  
- kind: Group  
  name: system:serviceaccounts:qa  
  apiGroup: rbac.authorization.k8s.io
```

For all service accounts in any namespace:

```
subjects:  
- kind: Group  
  name: system:serviceaccounts  
  apiGroup: rbac.authorization.k8s.io
```

For all authenticated users:

```
subjects:  
- kind: Group  
  name: system:authenticated  
  apiGroup: rbac.authorization.k8s.io
```

For all unauthenticated users:

```
subjects:  
- kind: Group  
  name: system:unauthenticated  
  apiGroup: rbac.authorization.k8s.io
```

For all users:

```
subjects:  
- kind: Group  
  name: system:authenticated  
  apiGroup: rbac.authorization.k8s.io  
- kind: Group
```

```
name: system:unauthenticated
apiGroup: rbac.authorization.k8s.io
```

Default roles and role bindings

API servers create a set of default ClusterRole and ClusterRoleBinding objects. Many of these are system: prefixed, which indicates that the resource is directly managed by the cluster control plane. All of the default ClusterRoles and ClusterRoleBindings are labeled with kubernetes.io/bootstrapping=rbac-defaults.

Caution: Take care when modifying ClusterRoles and ClusterRoleBindings with names that have a system: prefix. Modifications to these resources can result in non-functional clusters.

Auto-reconciliation

At each start-up, the API server updates default cluster roles with any missing permissions, and updates default cluster role bindings with any missing subjects. This allows the cluster to repair accidental modifications, and helps to keep roles and role bindings up-to-date as permissions and subjects change in new Kubernetes releases.

To opt out of this reconciliation, set the rbac.authorization.kubernetes.io/autoupdate annotation on a default cluster role or rolebinding to false. Be aware that missing default permissions and subjects can result in non-functional clusters.

Auto-reconciliation is enabled by default if the RBAC authorizer is active.

API discovery roles

Default role bindings authorize unauthenticated and authenticated users to read API information that is deemed safe to be publicly accessible (including CustomResourceDefinitions). To disable anonymous unauthenticated access, add --anonymous-auth=false to the API server configuration.

To view the configuration of these roles via kubectl run:

```
kubectl get clusterroles system:discovery -o yaml
```

Note: If you edit that ClusterRole, your changes will be overwritten on API server restart via [auto-reconciliation](#). To avoid that overwriting, either do not manually edit the role, or disable auto-reconciliation.

Kubernetes RBAC API discovery roles

Default ClusterRole	Default ClusterRoleBinding	Description
system:basic-user	system:authenticated group	Allows a user read-only access to basic information about themselves. Prior to v1.14, this role was also bound to system:unauthenticated by default.
system:discovery	system:authenticated group	Allows read-only access to API discovery endpoints needed to discover and negotiate an API level. Prior to v1.14, this role was also bound to system:unauthenticated by default.

Default ClusterRole	Default ClusterRoleBinding	Description
system:public-info-viewer	system:authenticated and system:unauthenticated groups	Allows read-only access to non-sensitive information about the cluster. Introduced in Kubernetes v1.14.

User-facing roles

Some of the default ClusterRoles are not system: prefixed. These are intended to be user-facing roles. They include super-user roles (cluster-admin), roles intended to be granted cluster-wide using ClusterRoleBindings, and roles intended to be granted within particular namespaces using RoleBindings (admin, edit, view).

User-facing ClusterRoles use [ClusterRole aggregation](#) to allow admins to include rules for custom resources on these ClusterRoles. To add rules to the admin, edit, or view roles, create a ClusterRole with one or more of the following labels:

```
metadata:
  labels:
    rbac.authorization.k8s.io/aggregate-to-admin: "true"
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
    rbac.authorization.k8s.io/aggregate-to-view: "true"
```

Default ClusterRole	Default ClusterRoleBinding	Description
cluster-admin	system:masters group	Allows super-user access to perform any action on any resource. When used in a ClusterRoleBinding , it gives full control over every resource in the cluster and in all namespaces. When used in a RoleBinding , it gives full control over every resource in the role binding's namespace, including the namespace itself.
admin	None	Allows admin access, intended to be granted within a namespace using a RoleBinding . If used in a RoleBinding , allows read/write access to most resources in a namespace, including the ability to create roles and role bindings within the namespace. This role does not allow write access to resource quota or to the namespace itself. This role also does not allow write access to EndpointSlices (or Endpoints) in clusters created using Kubernetes v1.22+. More information is available in the "Write Access for EndpointSlices and Endpoints" section .
edit	None	Allows read/write access to most objects in a namespace. This role does not allow viewing or modifying roles or role bindings. However, this role allows accessing Secrets and running Pods as any ServiceAccount in the namespace, so it can be used to gain the API access levels of any ServiceAccount in the namespace. This role also does not allow write access to EndpointSlices (or

Default ClusterRole	Default ClusterRoleBinding	Description
		Endpoints) in clusters created using Kubernetes v1.22+. More information is available in the "Write Access for EndpointSlices and Endpoints" section .
view	None	Allows read-only access to see most objects in a namespace. It does not allow viewing roles or role bindings. This role does not allow viewing Secrets, since reading the contents of Secrets enables access to ServiceAccount credentials in the namespace, which would allow API access as any ServiceAccount in the namespace (a form of privilege escalation).

Core component roles

Default ClusterRole	Default ClusterRoleBinding	Description
system:kube-scheduler	system:kube-scheduler user	Allows access to the resources required by the scheduler component.
system:volumer-scheduler	system:kube-scheduler user	Allows access to the volume resources required by the kube-scheduler component.
system:kube-controller-manager	system:kube-controller-manager user	Allows access to the resources required by the controller manager component. The permissions required by individual controllers are detailed in the controller roles .
system:node	None	Allows access to resources required by the kubelet, including read access to all secrets, and write access to all pod status objects . You should use the Node authorizer and NodeRestriction admission plugin instead of the system:node role, and allow granting API access to kubelets based on the Pods scheduled to run on them. The system:node role only exists for compatibility with Kubernetes clusters upgraded from versions prior to v1.8.
system:node-proxier	system:kube-proxy user	Allows access to the resources required by the kube-proxy component.

Other component roles

Default ClusterRole	Default ClusterRoleBinding	Description
	None	

Default ClusterRole	Default ClusterRoleBinding	Description
system:auth-delegator		Allows delegated authentication and authorization checks. This is commonly used by add-on API servers for unified authentication and authorization.
system:heapster	None	Role for the Heapster component (deprecated).
system:kube-aggregator	None	Role for the kube-aggregator component.
system:kube-dns	kube-dns service account in the kube-system namespace	Role for the kube-dns component.
system:kubelet-api-admin	None	Allows full access to the kubelet API.
system:node-bootstrapper	None	Allows access to the resources required to perform kubelet TLS bootstrapping .
system:node-problem-detector	None	Role for the node-problem-detector component.
system:persistent-volume-provisioner	None	Allows access to the resources required by most dynamic volume provisioners .
system:monitoring	system:monitoring group	Allows read access to control-plane monitoring endpoints (i.e. kube-apiserver liveness and readiness endpoints (/healthz, /livez, /readyz), the individual health-check endpoints (/healthz/*, /livez/*, /readyz/*), and /metrics). Note that individual health check endpoints and the metric endpoint may expose sensitive information.

Roles for built-in controllers

The Kubernetes [controller manager](#) runs [controllers](#) that are built in to the Kubernetes control plane. When invoked with --use-service-account-credentials, kube-controller-manager starts each controller using a separate service account. Corresponding roles exist for each built-in controller, prefixed with system:controller:. If the controller manager is not started with --use-service-account-credentials, it runs all control loops using its own credential, which must be granted all the relevant roles. These roles include:

- system:controller:attachdetach-controller
- system:controller:certificate-controller
- system:controller:clusterrole-aggregation-controller
- system:controller:cronjob-controller
- system:controller:daemon-set-controller
- system:controller:deployment-controller
- system:controller:disruption-controller
- system:controller:endpoint-controller
- system:controller:expand-controller
- system:controller:generic-garbage-collector
- system:controller:horizontal-pod-autoscaler
- system:controller:job-controller

- system:controller:namespace-controller
- system:controller:node-controller
- system:controller:persistent-volume-binder
- system:controller:pod-garbage-collector
- system:controller:pv-protection-controller
- system:controller:pvc-protection-controller
- system:controller:replicaset-controller
- system:controller:replication-controller
- system:controller:resourcequota-controller
- system:controller:root-ca-cert-publisher
- system:controller:route-controller
- system:controller:service-account-controller
- system:controller:service-controller
- system:controller:statefulset-controller
- system:controller:ttl-controller

Privilege escalation prevention and bootstrapping

The RBAC API prevents users from escalating privileges by editing roles or role bindings. Because this is enforced at the API level, it applies even when the RBAC authorizer is not in use.

Restrictions on role creation or update

You can only create/update a role if at least one of the following things is true:

1. You already have all the permissions contained in the role, at the same scope as the object being modified (cluster-wide for a ClusterRole, within the same namespace or cluster-wide for a Role).
2. You are granted explicit permission to perform the escalate verb on the roles or clusterroles resource in the rbac.authorization.k8s.io API group.

For example, if user-1 does not have the ability to list Secrets cluster-wide, they cannot create a ClusterRole containing that permission. To allow a user to create/update roles:

1. Grant them a role that allows them to create/update Role or ClusterRole objects, as desired.
2. Grant them permission to include specific permissions in the roles they create/update:
 - implicitly, by giving them those permissions (if they attempt to create or modify a Role or ClusterRole with permissions they themselves have not been granted, the API request will be forbidden)
 - or explicitly allow specifying any permission in a Role or ClusterRole by giving them permission to perform the escalate verb on roles or clusterroles resources in the rbac.authorization.k8s.io API group

Restrictions on role binding creation or update

You can only create/update a role binding if you already have all the permissions contained in the referenced role (at the same scope as the role binding) *or* if you have been authorized to perform the bind verb on the referenced role. For example, if user-1 does not have the ability to

list Secrets cluster-wide, they cannot create a ClusterRoleBinding to a role that grants that permission. To allow a user to create/update role bindings:

1. Grant them a role that allows them to create/update RoleBinding or ClusterRoleBinding objects, as desired.
2. Grant them permissions needed to bind a particular role:
 - implicitly, by giving them the permissions contained in the role.
 - explicitly, by giving them permission to perform the bind verb on the particular Role (or ClusterRole).

For example, this ClusterRole and RoleBinding would allow user-1 to grant other users the admin, edit, and view roles in the namespace user-1-namespace:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: role-grantor
rules:
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["rolebindings"]
  verbs: ["create"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["clusterroles"]
  verbs: ["bind"]
# omit resourceNames to allow binding any ClusterRole
resourceNames: ["admin","edit","view"]
---
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: role-grantor-binding
  namespace: user-1-namespace
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: role-grantor
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: User
  name: user-1
```

When bootstrapping the first roles and role bindings, it is necessary for the initial user to grant permissions they do not yet have. To bootstrap initial roles and role bindings:

- Use a credential with the "system:masters" group, which is bound to the "cluster-admin" super-user role by the default bindings.

Command-line utilities

kubectl create role

Creates a Role object defining permissions within a single namespace. Examples:

- Create a Role named "pod-reader" that allows users to perform get, watch and list on pods:

```
kubectl create role pod-reader --verb=get --verb=list --verb=watch --resource=pods
```

- Create a Role named "pod-reader" with resourceNames specified:

```
kubectl create role pod-reader --verb=get --resource=pods --resource-name=readablepod  
--resource-name=anotherpod
```

- Create a Role named "foo" with apiGroups specified:

```
kubectl create role foo --verb=get,list,watch --resource=replicaset.apps
```

- Create a Role named "foo" with subresource permissions:

```
kubectl create role foo --verb=get,list,watch --resource=pods,pods/status
```

- Create a Role named "my-component-lease-holder" with permissions to get/update a resource with a specific name:

```
kubectl create role my-component-lease-holder --verb=get,list,watch,update --resource=le  
ase --resource-name=my-component
```

kubectl create clusterrole

Creates a ClusterRole. Examples:

- Create a ClusterRole named "pod-reader" that allows user to perform get, watch and list on pods:

```
kubectl create clusterrole pod-reader --verb=get,list,watch --resource=pods
```

- Create a ClusterRole named "pod-reader" with resourceNames specified:

```
kubectl create clusterrole pod-reader --verb=get --resource=pods --resource-name=readab  
lepod --resource-name=anotherpod
```

- Create a ClusterRole named "foo" with apiGroups specified:

```
kubectl create clusterrole foo --verb=get,list,watch --resource=replicaset.apps
```

- Create a ClusterRole named "foo" with subresource permissions:

```
kubectl create clusterrole foo --verb=get,list,watch --resource=pods,pods/status
```

- Create a ClusterRole named "foo" with nonResourceURL specified:

```
kubectl create clusterrole "foo" --verb=get --non-resource-url=/logs/*
```

- Create a ClusterRole named "monitoring" with an aggregationRule specified:

```
kubectl create clusterrole monitoring --aggregation-rule="rbac.example.com/aggregate-to-monitoring=true"
```

kubectl create rolebinding

Grants a Role or ClusterRole within a specific namespace. Examples:

- Within the namespace "acme", grant the permissions in the "admin" ClusterRole to a user named "bob":

```
kubectl create rolebinding bob-admin-binding --clusterrole=admin --user=bob --namespace=acme
```

- Within the namespace "acme", grant the permissions in the "view" ClusterRole to the service account in the namespace "acme" named "myapp":

```
kubectl create rolebinding myapp-view-binding --clusterrole=view --serviceaccount=acme:myapp --namespace=acme
```

- Within the namespace "acme", grant the permissions in the "view" ClusterRole to a service account in the namespace "myappnamespace" named "myapp":

```
kubectl create rolebinding myappnamespace-myapp-view-binding --clusterrole=view --serviceaccount=myappnamespace:myapp --namespace=acme
```

kubectl create clusterrolebinding

Grants a ClusterRole across the entire cluster (all namespaces). Examples:

- Across the entire cluster, grant the permissions in the "cluster-admin" ClusterRole to a user named "root":

```
kubectl create clusterrolebinding root-cluster-admin-binding --clusterrole=cluster-admin --user=root
```

- Across the entire cluster, grant the permissions in the "system:node-proxier" ClusterRole to a user named "system:kube-proxy":

```
kubectl create clusterrolebinding kube-proxy-binding --clusterrole=system:node-proxier --user=system:kube-proxy
```

- Across the entire cluster, grant the permissions in the "view" ClusterRole to a service account named "myapp" in the namespace "acme":

```
kubectl create clusterrolebinding myapp-view-binding --clusterrole=view --serviceaccount=acme:myapp
```

kubectl auth reconcile

Creates or updates rbac.authorization.k8s.io/v1 API objects from a manifest file.

Missing objects are created, and the containing namespace is created for namespaced objects, if required.

Existing roles are updated to include the permissions in the input objects, and remove extra permissions if `--remove-extra-permissions` is specified.

Existing bindings are updated to include the subjects in the input objects, and remove extra subjects if `--remove-extra-subjects` is specified.

Examples:

- Test applying a manifest file of RBAC objects, displaying changes that would be made:

```
kubectl auth reconcile -f my-rbac-rules.yaml --dry-run=client
```

- Apply a manifest file of RBAC objects, preserving any extra permissions (in roles) and any extra subjects (in bindings):

```
kubectl auth reconcile -f my-rbac-rules.yaml
```

- Apply a manifest file of RBAC objects, removing any extra permissions (in roles) and any extra subjects (in bindings):

```
kubectl auth reconcile -f my-rbac-rules.yaml --remove-extra-subjects --remove-extra-permissions
```

ServiceAccount permissions

Default RBAC policies grant scoped permissions to control-plane components, nodes, and controllers, but grant *no permissions* to service accounts outside the `kube-system` namespace (beyond discovery permissions given to all authenticated users).

This allows you to grant particular roles to particular ServiceAccounts as needed. Fine-grained role bindings provide greater security, but require more effort to administrate. Broader grants can give unnecessary (and potentially escalating) API access to ServiceAccounts, but are easier to administrate.

In order from most secure to least secure, the approaches are:

1. Grant a role to an application-specific service account (best practice)

This requires the application to specify a `serviceAccountName` in its pod spec, and for the service account to be created (via the API, application manifest, `kubectl create serviceaccount`, etc.).

For example, grant read-only permission within "my-namespace" to the "my-sa" service account:

```
kubectl create rolebinding my-sa-view \
--clusterrole=view \
--serviceaccount=my-namespace:my-sa \
--namespace=my-namespace
```

2. Grant a role to the "default" service account in a namespace

If an application does not specify a serviceAccountName, it uses the "default" service account.

Note: Permissions given to the "default" service account are available to any pod in the namespace that does not specify a serviceAccountName.

For example, grant read-only permission within "my-namespace" to the "default" service account:

```
kubectl create rolebinding default-view \  
  --clusterrole=view \  
  --serviceaccount=my-namespace:default \  
  --namespace=my-namespace
```

Many [add-ons](#) run as the "default" service account in the kube-system namespace. To allow those add-ons to run with super-user access, grant cluster-admin permissions to the "default" service account in the kube-system namespace.

Caution: Enabling this means the kube-system namespace contains Secrets that grant super-user access to your cluster's API.

```
kubectl create clusterrolebinding add-on-cluster-admin \  
  --clusterrole=cluster-admin \  
  --serviceaccount=kube-system:default
```

3. Grant a role to all service accounts in a namespace

If you want all applications in a namespace to have a role, no matter what service account they use, you can grant a role to the service account group for that namespace.

For example, grant read-only permission within "my-namespace" to all service accounts in that namespace:

```
kubectl create rolebinding serviceaccounts-view \  
  --clusterrole=view \  
  --group=system:serviceaccounts:my-namespace \  
  --namespace=my-namespace
```

4. Grant a limited role to all service accounts cluster-wide (discouraged)

If you don't want to manage permissions per-namespace, you can grant a cluster-wide role to all service accounts.

For example, grant read-only permission across all namespaces to all service accounts in the cluster:

```
kubectl create clusterrolebinding serviceaccounts-view \  
  --clusterrole=view \  
  --group=system:serviceaccounts
```

5. Grant super-user access to all service accounts cluster-wide (strongly discouraged)

If you don't care about partitioning permissions at all, you can grant super-user access to all service accounts.

Warning: This allows any application full access to your cluster, and also grants any user with read access to Secrets (or the ability to create any pod) full access to your cluster.

```
kubectl create clusterrolebinding serviceaccounts-cluster-admin \
--clusterrole=cluster-admin \
--group=system:serviceaccounts
```

Write access for EndpointSlices and Endpoints

Kubernetes clusters created before Kubernetes v1.22 include write access to EndpointSlices (and Endpoints) in the aggregated "edit" and "admin" roles. As a mitigation for [CVE-2021-25740](#), this access is not part of the aggregated roles in clusters that you create using Kubernetes v1.22 or later.

Existing clusters that have been upgraded to Kubernetes v1.22 will not be subject to this change. The [CVE announcement](#) includes guidance for restricting this access in existing clusters.

If you want new clusters to retain this level of access in the aggregated roles, you can create the following ClusterRole:

[access/endpoints-aggregated.yaml](#)

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  annotations:
    kubernetes.io/description: |->
      Add endpoints write permissions to the edit and admin roles. This was
      removed by default in 1.22 because of CVE-2021-25740. See
      https://issue.k8s.io/103675. This can allow writers to direct LoadBalancer
      or Ingress implementations to expose backend IPs that would not otherwise
      be accessible, and can circumvent network policies or security controls
      intended to prevent/isolate access to those backends.
      EndpointSlices were never included in the edit or admin roles, so there
      is nothing to restore for the EndpointSlice API.
  labels:
    rbac.authorization.k8s.io/aggregate-to-edit: "true"
  name: custom:aggregate-to-edit:endpoints # you can change this if you wish
rules:
  - apiGroups: []
    resources: ["endpoints"]
    verbs: ["create", "delete", "deletecollection", "patch", "update"]
```

Upgrading from ABAC

Clusters that originally ran older Kubernetes versions often used permissive ABAC policies, including granting full API access to all service accounts.

Default RBAC policies grant scoped permissions to control-plane components, nodes, and controllers, but grant *no permissions* to service accounts outside the kube-system namespace (beyond discovery permissions given to all authenticated users).

While far more secure, this can be disruptive to existing workloads expecting to automatically receive API permissions. Here are two approaches for managing this transition:

Parallel authorizers

Run both the RBAC and ABAC authorizers, and specify a policy file that contains the [legacy ABAC policy](#):

```
--authorization-mode=...,RBAC,ABAC --authorization-policy-file=mypolicy.json
```

To explain that first command line option in detail: if earlier authorizers, such as Node, deny a request, then the RBAC authorizer attempts to authorize the API request. If RBAC also denies that API request, the ABAC authorizer is then run. This means that any request allowed by *either* the RBAC or ABAC policies is allowed.

When the kube-apiserver is run with a log level of 5 or higher for the RBAC component (`--vmodule=rbac*=5` or `--v=5`), you can see RBAC denials in the API server log (prefixed with RBAC). You can use that information to determine which roles need to be granted to which users, groups, or service accounts.

Once you have [granted roles to service accounts](#) and workloads are running with no RBAC denial messages in the server logs, you can remove the ABAC authorizer.

Permissive RBAC permissions

You can replicate a permissive ABAC policy using RBAC role bindings.

Warning:

The following policy allows **ALL** service accounts to act as cluster administrators. Any application running in a container receives service account credentials automatically, and could perform any action against the API, including viewing secrets and modifying permissions. This is not a recommended policy.

```
kubectl create clusterrolebinding permissive-binding \
--clusterrole=cluster-admin \
--user=admin \
--user=kubelet \
--group=system:serviceaccounts
```

After you have transitioned to use RBAC, you should adjust the access controls for your cluster to ensure that these meet your information security needs.

Using ABAC Authorization

Attribute-based access control (ABAC) defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together.

Policy File Format

To enable ABAC mode, specify `--authorization-policy-file=SOME_FILENAME` and `--authorization-mode=ABAC` on startup.

The file format is [one JSON object per line](#). There should be no enclosing list or map, only one map per line.

Each line is a "policy object", where each such object is a map with the following properties:

- Versioning properties:
 - `apiVersion`, type string; valid values are "abac.authorization.kubernetes.io/v1beta1". Allows versioning and conversion of the policy format.
 - `kind`, type string: valid values are "Policy". Allows versioning and conversion of the policy format.
- `spec` property set to a map with the following properties:
 - Subject-matching properties:
 - `user`, type string; the user-string from `--token-auth-file`. If you specify `user`, it must match the username of the authenticated user.
 - `group`, type string; if you specify `group`, it must match one of the groups of the authenticated user. `system:authenticated` matches all authenticated requests. `system:unauthenticated` matches all unauthenticated requests.
 - Resource-matching properties:
 - `apiGroup`, type string; an API group.
 - Ex: `apps`, `networking.k8s.io`
 - Wildcard: `*` matches all API groups.
 - `namespace`, type string; a namespace.
 - Ex: `kube-system`
 - Wildcard: `*` matches all resource requests.
 - `resource`, type string; a resource type
 - Ex: `pods`, `deployments`
 - Wildcard: `*` matches all resource requests.
 - Non-resource-matching properties:
 - `nonResourcePath`, type string; non-resource request paths.
 - Ex: `/version` or `/apis`
 - Wildcard:
 - `*` matches all non-resource requests.
 - `/foo/*` matches all subpaths of `/foo/`.
 - `readonly`, type boolean, when true, means that the Resource-matching policy only applies to get, list, and watch operations, Non-resource-matching policy only applies to get operation.

Note:

An unset property is the same as a property set to the zero value for its type (e.g. empty string, 0, false). However, unset should be preferred for readability.

In the future, policies may be expressed in a JSON format, and managed via a REST interface.

Authorization Algorithm

A request has attributes which correspond to the properties of a policy object.

When a request is received, the attributes are determined. Unknown attributes are set to the zero value of its type (e.g. empty string, 0, false).

A property set to `"*"` will match any value of the corresponding attribute.

The tuple of attributes is checked for a match against every policy in the policy file. If at least one line matches the request attributes, then the request is authorized (but may fail later validation).

To permit any authenticated user to do something, write a policy with the group property set to `"system:authenticated"`.

To permit any unauthenticated user to do something, write a policy with the group property set to `"system:unauthenticated"`.

To permit a user to do anything, write a policy with the `apiGroup`, `namespace`, `resource`, and `nonResourcePath` properties set to `"*"`.

Kubectl

Kubectl uses the `/api` and `/apis` endpoints of `apiserver` to discover served resource types, and validates objects sent to the API by create/update operations using schema information located at `/openapi/v2`.

When using ABAC authorization, those special resources have to be explicitly exposed via the `nonResourcePath` property in a policy (see [examples](#) below):

- `/api`, `/api/*`, `/apis`, and `/apis/*` for API version negotiation.
- `/version` for retrieving the server version via kubectl version.
- `/swaggerapi/*` for create/update operations.

To inspect the HTTP calls involved in a specific kubectl operation you can turn up the verbosity:

```
kubectl --v=8 version
```

Examples

1. Alice can do anything to all resources:

```
{"apiVersion": "abac.authorization.kubernetes.io/v1beta1", "kind": "Policy", "spec": {"user": "alice", "namespace": "*", "resource": "*", "apiGroup": "*"}}
```

2. The kubelet can read any pods:

```
{"apiVersion": "abac.authorization.kubernetes.io/v1beta1", "kind": "Policy", "spec": {"user": "kubelet", "namespace": "*", "resource": "pods", "readonly": true}}
```

3. The kubelet can read and write events:

```
{"apiVersion": "abac.authorization.kubernetes.io/v1beta1", "kind": "Policy", "spec": {"user": "kubelet", "namespace": "*", "resource": "events"}}
```

Bob can just read pods in namespace "projectCaribou":

4.

```
{"apiVersion": "abac.authorization.kubernetes.io/v1beta1", "kind": "Policy", "spec": {"user": "bob", "namespace": "projectCaribou", "resource": "pods", "readonly": true}}
```

5. Anyone can make read-only requests to all non-resource paths:

```
{"apiVersion": "abac.authorization.kubernetes.io/v1beta1", "kind": "Policy", "spec": {"group": "system:authenticated", "readonly": true, "nonResourcePath": "*"}}, {"apiVersion": "abac.authorization.kubernetes.io/v1beta1", "kind": "Policy", "spec": {"group": "system:unauthenticated", "readonly": true, "nonResourcePath": "*"}},
```

[Complete file example](#)

A quick note on service accounts

Every service account has a corresponding ABAC username, and that service account's username is generated according to the naming convention:

```
system:serviceaccount:<namespace>:<serviceaccountname>
```

Creating a new namespace leads to the creation of a new service account in the following format:

```
system:serviceaccount:<namespace>:default
```

For example, if you wanted to grant the default service account (in the kube-system namespace) full privilege to the API using ABAC, you would add this line to your policy file:

```
{"apiVersion": "abac.authorization.kubernetes.io/v1beta1", "kind": "Policy", "spec": {"user": "system:serviceaccount:kube-system:default", "namespace": "*", "resource": "*", "apiGroup": "*"}},
```

The apiserver will need to be restarted to pick up the new policy lines.

Using Node Authorization

Node authorization is a special-purpose authorization mode that specifically authorizes API requests made by kubelets.

Overview

The Node authorizer allows a kubelet to perform API operations. This includes:

Read operations:

- services
- endpoints
- nodes
- pods
- secrets, configmaps, persistent volume claims and persistent volumes related to pods bound to the kubelet's node

Write operations:

- nodes and node status (enable the NodeRestriction admission plugin to limit a kubelet to modify its own node)
- pods and pod status (enable the NodeRestriction admission plugin to limit a kubelet to modify pods bound to itself)
- events

Auth-related operations:

- read/write access to the [CertificateSigningRequests API](#) for TLS bootstrapping
- the ability to create TokenReviews and SubjectAccessReviews for delegated authentication/authorization checks

In future releases, the node authorizer may add or remove permissions to ensure kubelets have the minimal set of permissions required to operate correctly.

In order to be authorized by the Node authorizer, kubelets must use a credential that identifies them as being in the system:nodes group, with a username of system:node:<nodeName>. This group and user name format match the identity created for each kubelet as part of [kubelet TLS bootstrapping](#).

The value of <nodeName> **must** match precisely the name of the node as registered by the kubelet. By default, this is the host name as provided by hostname, or overridden via the [kubelet option](#) --hostname-override. However, when using the --cloud-provider kubelet option, the specific hostname may be determined by the cloud provider, ignoring the local hostname and the --hostname-override option. For specifics about how the kubelet determines the hostname, see the [kubelet options reference](#).

To enable the Node authorizer, start the apiserver with --authorization-mode=Node.

To limit the API objects kubelets are able to write, enable the [NodeRestriction](#) admission plugin by starting the apiserver with --enable-admission-plugins=...,NodeRestriction,...

Migration considerations

Kubelets outside the system:nodes group

Kubelets outside the system:nodes group would not be authorized by the Node authorization mode, and would need to continue to be authorized via whatever mechanism currently authorizes them. The node admission plugin would not restrict requests from these kubelets.

Kubelets with undifferentiated usernames

In some deployments, kubelets have credentials that place them in the system:nodes group, but do not identify the particular node they are associated with, because they do not have a username in the system:node:... format. These kubelets would not be authorized by the Node authorization mode, and would need to continue to be authorized via whatever mechanism currently authorizes them.

The NodeRestriction admission plugin would ignore requests from these kubelets, since the default node identifier implementation would not consider that a node identity.

Mapping PodSecurityPolicies to Pod Security Standards

The tables below enumerate the configuration parameters on PodSecurityPolicy objects, whether the field mutates and/or validates pods, and how the configuration values map to the [Pod Security Standards](#).

For each applicable parameter, the allowed values for the [Baseline](#) and [Restricted](#) profiles are listed. Anything outside the allowed values for those profiles would fall under the [Privileged](#) profile. "No opinion" means all values are allowed under all Pod Security Standards.

For a step-by-step migration guide, see [Migrate from PodSecurityPolicy to the Built-In PodSecurity Admission Controller](#).

PodSecurityPolicy Spec

The fields enumerated in this table are part of the PodSecurityPolicySpec, which is specified under the .spec field path.

Mapping PodSecurityPolicySpec fields to Pod Security Standards

PodSecurityPolicySpec	Type	Pod Security Standards Equivalent
privileged	Validating	Baseline & Restricted: false / undefined / nil
defaultAddCapabilities	Mutating & Validating	Requirements match allowedCapabiliites below.
allowedCapabilities	Validating	Baseline: subset of <ul style="list-style-type: none">• AUDIT_WRITE• CHOWN• DAC_OVERRIDE• FOWNER• FSETID• KILL• MKNOD• NET_BIND_SERVICE• SETFCAP• SETGID• SETPCAP• SETUID• SYS_CHROOT Restricted: empty / undefined / nil OR a list containing <i>only</i> NET_BIND_SERVICE
requiredDropCapabilities	Mutating & Validating	Baseline: no opinion Restricted: must include ALL

volumes	Validating	<p>Baseline: anything except</p> <ul style="list-style-type: none"> hostPath * <p>Restricted: subset of</p> <ul style="list-style-type: none"> configMap csi downwardAPI emptyDir ephemeral persistentVolumeClaim projected secret
hostNetwork	Validating	Baseline & Restricted: false / undefined / nil
hostPorts	Validating	Baseline & Restricted: undefined / nil / empty
hostPID	Validating	Baseline & Restricted: false / undefined / nil
hostIPC	Validating	Baseline & Restricted: false / undefined / nil
seLinux	Mutating & Validating	<p>Baseline & Restricted: seLinux.rule is MustRunAs, with the following options</p> <ul style="list-style-type: none"> user is unset ("" / undefined / nil) role is unset ("" / undefined / nil) type is unset or one of: container_t, container_init_t, container_kvm_t level is anything
runAsUser	Mutating & Validating	<p>Baseline: Anything</p> <p>Restricted: rule is MustRunAsNonRoot</p>
runAsGroup	Mutating (MustRunAs) & Validating	<i>No opinion</i>
supplementalGroups	Mutating & Validating	<i>No opinion</i>
fsGroup	Mutating & Validating	<i>No opinion</i>
readOnlyRootFilesystem	Mutating & Validating	<i>No opinion</i>
defaultAllowPrivilegeEscalation	Mutating	<i>No opinion (non-validating)</i>
allowPrivilegeEscalation	Mutating & Validating	<i>Only mutating if set to false</i>

		Baseline: No opinion Restricted: false
allowedHostPaths	Validating	<i>No opinion (volumes takes precedence)</i>
allowedFlexVolumes	Validating	<i>No opinion (volumes takes precedence)</i>
allowedCSIDrivers	Validating	<i>No opinion (volumes takes precedence)</i>
allowedUnsafeSysctls	Validating	Baseline & Restricted: undefined / nil / empty
forbiddenSysctls	Validating	<i>No opinion</i>
allowedProcMountTypes (alpha feature)	Validating	Baseline & Restricted: ["Default"] OR undefined / nil / empty
runtimeClass .defaultRuntimeClassName	Mutating	<i>No opinion</i>
runtimeClass .allowedRuntimeClassNames	Validating	<i>No opinion</i>

PodSecurityPolicy annotations

The [annotations](#) enumerated in this table can be specified under .metadata.annotations on the PodSecurityPolicy object.

Mapping PodSecurityPolicy annotations to Pod Security Standards

PSP Annotation	Type	Pod Security Standards Equivalent
seccomp.security.alpha.kubernetes.io /defaultProfileName	Mutating	<i>No opinion</i>
seccomp.security.alpha.kubernetes.io /allowedProfileNames	Validating	Baseline: "runtime/default," (Trailing comma to allow unset) Restricted: "runtime/default" (No trailing comma) <i>localhost/* values are also permitted for both Baseline & Restricted.</i>
apparmor.security.beta.kubernetes.io /defaultProfileName	Mutating	<i>No opinion</i>
apparmor.security.beta.kubernetes.io /allowedProfileNames	Validating	Baseline: "runtime/default," (Trailing comma to allow unset) Restricted: "runtime/default" (No trailing comma) <i>localhost/* values are also permitted for both Baseline & Restricted.</i>

Webhook Mode

A WebHook is an HTTP callback: an HTTP POST that occurs when something happens; a simple event-notification via HTTP POST. A web application implementing WebHooks will POST a message to a URL when certain things happen.

When specified, mode Webhook causes Kubernetes to query an outside REST service when determining user privileges.

Configuration File Format

Mode Webhook requires a file for HTTP configuration, specify by the --authorization-webhook-config-file=SOME_FILENAME flag.

The configuration file uses the [kubeconfig](#) file format. Within the file "users" refers to the API Server webhook and "clusters" refers to the remote service.

A configuration example which uses HTTPS client auth:

```
# Kubernetes API version
apiVersion: v1
# kind of the API object
kind: Config
# clusters refers to the remote service.
clusters:
- name: name-of-remote-authz-service
  cluster:
    # CA for verifying the remote service.
    certificate-authority: /path/to/ca.pem
    # URL of remote service to query. Must use 'https'. May not include parameters.
    server: https://authz.example.com/authorize

# users refers to the API Server's webhook configuration.
users:
- name: name-of-api-server
  user:
    client-certificate: /path/to/cert.pem # cert for the webhook plugin to use
    client-key: /path/to/key.pem      # key matching the cert

# kubeconfig files require a context. Provide one for the API Server.
current-context: webhook
contexts:
- context:
  cluster: name-of-remote-authz-service
  user: name-of-api-server
  name: webhook
```

Request Payloads

When faced with an authorization decision, the API Server POSTs a JSON- serialized `authorization.k8s.io/v1beta1 SubjectAccessReview` object describing the action. This object

contains fields describing the user attempting to make the request, and either details about the resource being accessed or requests attributes.

Note that webhook API objects are subject to the same [versioning compatibility rules](#) as other Kubernetes API objects. Implementers should be aware of looser compatibility promises for beta objects and check the "apiVersion" field of the request to ensure correct deserialization. Additionally, the API Server must enable the authorization.k8s.io/v1beta1 API extensions group (--runtime-config=authorization.k8s.io/v1beta1=true).

An example request body:

```
{  
  "apiVersion": "authorization.k8s.io/v1beta1",  
  "kind": "SubjectAccessReview",  
  "spec": {  
    "resourceAttributes": {  
      "namespace": "kittensandponies",  
      "verb": "get",  
      "group": "unicorn.example.org",  
      "resource": "pods"  
    },  
    "user": "jane",  
    "group": [  
      "group1",  
      "group2"  
    ]  
  }  
}
```

The remote service is expected to fill the status field of the request and respond to either allow or disallow access. The response body's spec field is ignored and may be omitted. A permissive response would return:

```
{  
  "apiVersion": "authorization.k8s.io/v1beta1",  
  "kind": "SubjectAccessReview",  
  "status": {  
    "allowed": true  
  }  
}
```

For disallowing access there are two methods.

The first method is preferred in most cases, and indicates the authorization webhook does not allow, or has "no opinion" about the request, but if other authorizers are configured, they are given a chance to allow the request. If there are no other authorizers, or none of them allow the request, the request is forbidden. The webhook would return:

```
{  
  "apiVersion": "authorization.k8s.io/v1beta1",  
  "kind": "SubjectAccessReview",  
  "status": {  
    "allowed": false,  
    "reason": "user does not have read access to the namespace"  
  }
```

```
}
```

The second method denies immediately, short-circuiting evaluation by other configured authorizers. This should only be used by webhooks that have detailed knowledge of the full authorizer configuration of the cluster. The webhook would return:

```
{
  "apiVersion": "authorization.k8s.io/v1beta1",
  "kind": "SubjectAccessReview",
  "status": {
    "allowed": false,
    "denied": true,
    "reason": "user does not have read access to the namespace"
  }
}
```

Access to non-resource paths are sent as:

```
{
  "apiVersion": "authorization.k8s.io/v1beta1",
  "kind": "SubjectAccessReview",
  "spec": {
    "nonResourceAttributes": {
      "path": "/debug",
      "verb": "get"
    },
    "user": "jane",
    "group": [
      "group1",
      "group2"
    ]
  }
}
```

Non-resource paths include: /api, /apis, /metrics, /logs, /debug, /healthz, /livez, /openapi/v2, /readyz, and /version. Clients require access to /api, /api/*, /apis, /apis/*, and /version to discover what resources and versions are present on the server. Access to other non-resource paths can be disallowed without restricting access to the REST api.

For further documentation refer to the authorization.v1beta1 API objects and [webhook.go](#).

Kubelet authentication/authorization

Overview

A kubelet's HTTPS endpoint exposes APIs which give access to data of varying sensitivity, and allow you to perform operations with varying levels of power on the node and within containers.

This document describes how to authenticate and authorize access to the kubelet's HTTPS endpoint.

Kubelet authentication

By default, requests to the kubelet's HTTPS endpoint that are not rejected by other configured authentication methods are treated as anonymous requests, and given a username of system:anonymous and a group of system:unauthenticated.

To disable anonymous access and send 401 Unauthorized responses to unauthenticated requests:

- start the kubelet with the --anonymous-auth=false flag

To enable X509 client certificate authentication to the kubelet's HTTPS endpoint:

- start the kubelet with the --client-ca-file flag, providing a CA bundle to verify client certificates with
- start the apiserver with --kubelet-client-certificate and --kubelet-client-key flags
- see the [apiserver authentication documentation](#) for more details

To enable API bearer tokens (including service account tokens) to be used to authenticate to the kubelet's HTTPS endpoint:

- ensure the authentication.k8s.io/v1beta1 API group is enabled in the API server
- start the kubelet with the --authentication-token-webhook and --kubeconfig flags
- the kubelet calls the TokenReview API on the configured API server to determine user information from bearer tokens

Kubelet authorization

Any request that is successfully authenticated (including an anonymous request) is then authorized. The default authorization mode is AlwaysAllow, which allows all requests.

There are many possible reasons to subdivide access to the kubelet API:

- anonymous auth is enabled, but anonymous users' ability to call the kubelet API should be limited
- bearer token auth is enabled, but arbitrary API users' (like service accounts) ability to call the kubelet API should be limited
- client certificate auth is enabled, but only some of the client certificates signed by the configured CA should be allowed to use the kubelet API

To subdivide access to the kubelet API, delegate authorization to the API server:

- ensure the authorization.k8s.io/v1beta1 API group is enabled in the API server
- start the kubelet with the --authorization-mode=Webhook and the --kubeconfig flags
- the kubelet calls the SubjectAccessReview API on the configured API server to determine whether each request is authorized

The kubelet authorizes API requests using the same [request attributes](#) approach as the apiserver.

The verb is determined from the incoming request's HTTP verb:

HTTP verb	request verb
POST	create
GET, HEAD	get
PUT	update
PATCH	patch
DELETE	delete

The resource and subresource is determined from the incoming request's path:

Kubelet API	resource	subresource
/stats/*	nodes	stats
/metrics/*	nodes	metrics
/logs/*	nodes	log
/spec/*	nodes	spec
<i>all others</i>	nodes	proxy

The namespace and API group attributes are always an empty string, and the resource name is always the name of the kubelet's Node API object.

When running in this mode, ensure the user identified by the `--kubelet-client-certificate` and `--kubelet-client-key` flags passed to the apiserver is authorized for the following attributes:

- `verb=*, resource=nodes, subresource=proxy`
- `verb=*, resource=nodes, subresource=stats`
- `verb=*, resource=nodes, subresource=log`
- `verb=*, resource=nodes, subresource=spec`
- `verb=*, resource=nodes, subresource=metrics`

TLS bootstrapping

In a Kubernetes cluster, the components on the worker nodes - kubelet and kube-proxy - need to communicate with Kubernetes control plane components, specifically kube-apiserver. In order to ensure that communication is kept private, not interfered with, and ensure that each component of the cluster is talking to another trusted component, we strongly recommend using client TLS certificates on nodes.

The normal process of bootstrapping these components, especially worker nodes that need certificates so they can communicate safely with kube-apiserver, can be a challenging process as it is often outside of the scope of Kubernetes and requires significant additional work. This in turn, can make it challenging to initialize or scale a cluster.

In order to simplify the process, beginning in version 1.4, Kubernetes introduced a certificate request and signing API. The proposal can be found [here](#).

This document describes the process of node initialization, how to set up TLS client certificate bootstrapping for kubelets, and how it works.

Initialization process

When a worker node starts up, the kubelet does the following:

1. Look for its kubeconfig file
2. Retrieve the URL of the API server and credentials, normally a TLS key and signed certificate from the kubeconfig file
3. Attempt to communicate with the API server using the credentials.

Assuming that the kube-apiserver successfully validates the kubelet's credentials, it will treat the kubelet as a valid node, and begin to assign pods to it.

Note that the above process depends upon:

- Existence of a key and certificate on the local host in the kubeconfig
- The certificate having been signed by a Certificate Authority (CA) trusted by the kube-apiserver

All of the following are responsibilities of whoever sets up and manages the cluster:

1. Creating the CA key and certificate
2. Distributing the CA certificate to the control plane nodes, where kube-apiserver is running
3. Creating a key and certificate for each kubelet; strongly recommended to have a unique one, with a unique CN, for each kubelet
4. Signing the kubelet certificate using the CA key
5. Distributing the kubelet key and signed certificate to the specific node on which the kubelet is running

The TLS Bootstrapping described in this document is intended to simplify, and partially or even completely automate, steps 3 onwards, as these are the most common when initializing or scaling a cluster.

Bootstrap initialization

In the bootstrap initialization process, the following occurs:

1. kubelet begins
2. kubelet sees that it does *not* have a kubeconfig file
3. kubelet searches for and finds a bootstrap-kubeconfig file
4. kubelet reads its bootstrap file, retrieving the URL of the API server and a limited usage "token"
5. kubelet connects to the API server, authenticates using the token
6. kubelet now has limited credentials to create and retrieve a certificate signing request (CSR)
7. kubelet creates a CSR for itself with the signerName set to kubernetes.io/kube-apiserver-client-kubelet
8. CSR is approved in one of two ways:
 - If configured, kube-controller-manager automatically approves the CSR
 - If configured, an outside process, possibly a person, approves the CSR using the Kubernetes API or via kubectl
9. Certificate is created for the kubelet
10. Certificate is issued to the kubelet

11. kubelet retrieves the certificate
12. kubelet creates a proper kubeconfig with the key and signed certificate
13. kubelet begins normal operation
14. Optional: if configured, kubelet automatically requests renewal of the certificate when it is close to expiry
15. The renewed certificate is approved and issued, either automatically or manually, depending on configuration.

The rest of this document describes the necessary steps to configure TLS Bootstrapping, and its limitations.

Configuration

To configure for TLS bootstrapping and optional automatic approval, you must configure options on the following components:

- kube-apiserver
- kube-controller-manager
- kubelet
- in-cluster resources: ClusterRoleBinding and potentially ClusterRole

In addition, you need your Kubernetes Certificate Authority (CA).

Certificate Authority

As without bootstrapping, you will need a Certificate Authority (CA) key and certificate. As without bootstrapping, these will be used to sign the kubelet certificate. As before, it is your responsibility to distribute them to control plane nodes.

For the purposes of this document, we will assume these have been distributed to control plane nodes at /var/lib/kubernetes/ca.pem (certificate) and /var/lib/kubernetes/ca-key.pem (key). We will refer to these as "Kubernetes CA certificate and key".

All Kubernetes components that use these certificates - kubelet, kube-apiserver, kube-controller-manager - assume the key and certificate to be PEM-encoded.

kube-apiserver configuration

The kube-apiserver has several requirements to enable TLS bootstrapping:

- Recognizing CA that signs the client certificate
- Authenticating the bootstrapping kubelet to the system:bootstrappers group
- Authorize the bootstrapping kubelet to create a certificate signing request (CSR)

Recognizing client certificates

This is normal for all client certificate authentication. If not already set, add the --client-ca-file=FILENAME flag to the kube-apiserver command to enable client certificate authentication, referencing a certificate authority bundle containing the signing certificate, for example --client-ca-file=/var/lib/kubernetes/ca.pem.

Initial bootstrap authentication

In order for the bootstrapping kubelet to connect to kube-apiserver and request a certificate, it must first authenticate to the server. You can use any [authenticator](#) that can authenticate the kubelet.

While any authentication strategy can be used for the kubelet's initial bootstrap credentials, the following two authenticators are recommended for ease of provisioning.

1. [Bootstrap Tokens](#)
2. [Token authentication file](#)

Using bootstrap tokens is a simpler and more easily managed method to authenticate kubelets, and does not require any additional flags when starting kube-apiserver.

Whichever method you choose, the requirement is that the kubelet be able to authenticate as a user with the rights to:

1. create and retrieve CSRs
2. be automatically approved to request node client certificates, if automatic approval is enabled.

A kubelet authenticating using bootstrap tokens is authenticated as a user in the group system:bootstrappers, which is the standard method to use.

As this feature matures, you should ensure tokens are bound to a Role Based Access Control (RBAC) policy which limits requests (using the [bootstrap token](#)) strictly to client requests related to certificate provisioning. With RBAC in place, scoping the tokens to a group allows for great flexibility. For example, you could disable a particular bootstrap group's access when you are done provisioning the nodes.

Bootstrap tokens

Bootstrap tokens are described in detail [here](#). These are tokens that are stored as secrets in the Kubernetes cluster, and then issued to the individual kubelet. You can use a single token for an entire cluster, or issue one per worker node.

The process is two-fold:

1. Create a Kubernetes secret with the token ID, secret and scope(s).
2. Issue the token to the kubelet

From the kubelet's perspective, one token is like another and has no special meaning. From the kube-apiserver's perspective, however, the bootstrap token is special. Due to its type, namespace and name, kube-apiserver recognizes it as a special token, and grants anyone authenticating with that token special bootstrap rights, notably treating them as a member of the system:bootstrappers group. This fulfills a basic requirement for TLS bootstrapping.

The details for creating the secret are available [here](#).

If you want to use bootstrap tokens, you must enable it on kube-apiserver with the flag:

```
--enable-bootstrap-token-auth=true
```

Token authentication file

kube-apiserver has the ability to accept tokens as authentication. These tokens are arbitrary but should represent at least 128 bits of entropy derived from a secure random number generator (such as /dev/urandom on most modern Linux systems). There are multiple ways you can generate a token. For example:

```
head -c 16 /dev/urandom | od -An -t x | tr -d ''
```

This will generate tokens that look like 02b50b05283e98dd0fd71db496ef01e8.

The token file should look like the following example, where the first three values can be anything and the quoted group name should be as depicted:

```
02b50b05283e98dd0fd71db496ef01e8,kubelet-bootstrap,10001,"system:bootstrappers"
```

Add the `--token-auth-file=FILENAME` flag to the kube-apiserver command (in your systemd unit file perhaps) to enable the token file. See docs [here](#) for further details.

Authorize kubelet to create CSR

Now that the bootstrapping node is *authenticated* as part of the system:bootstrappers group, it needs to be *authorized* to create a certificate signing request (CSR) as well as retrieve it when done. Fortunately, Kubernetes ships with a ClusterRole with precisely these (and only these) permissions, system:node-bootstrapper.

To do this, you only need to create a ClusterRoleBinding that binds the system:bootstrappers group to the cluster role system:node-bootstrapper.

```
# enable bootstrapping nodes to create CSR
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: create-csrs-for-bootstrapping
subjects:
- kind: Group
  name: system:bootstrappers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: system:node-bootstrapper
  apiGroup: rbac.authorization.k8s.io
```

kube-controller-manager configuration

While the apiserver receives the requests for certificates from the kubelet and authenticates those requests, the controller-manager is responsible for issuing actual signed certificates.

The controller-manager performs this function via a certificate-issuing control loop. This takes the form of a [cfssl](#) local signer using assets on disk. Currently, all certificates issued have one year validity and a default set of key usages.

In order for the controller-manager to sign certificates, it needs the following:

- access to the "Kubernetes CA key and certificate" that you created and distributed
- enabling CSR signing

Access to key and certificate

As described earlier, you need to create a Kubernetes CA key and certificate, and distribute it to the control plane nodes. These will be used by the controller-manager to sign the kubelet certificates.

Since these signed certificates will, in turn, be used by the kubelet to authenticate as a regular kubelet to kube-apiserver, it is important that the CA provided to the controller-manager at this stage also be trusted by kube-apiserver for authentication. This is provided to kube-apiserver with the flag `--client-ca-file=FILENAME` (for example, `--client-ca-file=/var/lib/kubernetes/ca.pem`), as described in the kube-apiserver configuration section.

To provide the Kubernetes CA key and certificate to kube-controller-manager, use the following flags:

```
--cluster-signing-cert-file="/etc/path/to/kubernetes/ca/ca.crt" --cluster-signing-key-file="/etc/path/to/kubernetes/ca/ca.key"
```

For example:

```
--cluster-signing-cert-file="/var/lib/kubernetes/ca.pem" --cluster-signing-key-file="/var/lib/kubernetes/ca-key.pem"
```

The validity duration of signed certificates can be configured with flag:

```
--cluster-signing-duration
```

Approval

In order to approve CSRs, you need to tell the controller-manager that it is acceptable to approve them. This is done by granting RBAC permissions to the correct group.

There are two distinct sets of permissions:

- `nodeclient`: If a node is creating a new certificate for a node, then it does not have a certificate yet. It is authenticating using one of the tokens listed above, and thus is part of the group `system:bootstrappers`.
- `selfnodeclient`: If a node is renewing its certificate, then it already has a certificate (by definition), which it uses continuously to authenticate as part of the group `system:nodes`.

To enable the kubelet to request and receive a new certificate, create a ClusterRoleBinding that binds the group in which the bootstrapping node is a member `system:bootstrappers` to the ClusterRole that grants it permission, `system:certificates.k8s.io:certificatesigningrequests:nodeclient`:

```
# Approve all CSRs for the group "system:bootstrappers"
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
```

```

name: auto-approve-csrs-for-group
subjects:
- kind: Group
  name: system:bootstrappers
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: system:certificates.k8s.io:certificatesigningrequests:nodelclient
  apiGroup: rbac.authorization.k8s.io

```

To enable the kubelet to renew its own client certificate, create a ClusterRoleBinding that binds the group in which the fully functioning node is a member system:nodes to the ClusterRole that grants it permission, system:certificates.k8s.io:certificatesigningrequests:selfnodelclient:

```

# Approve renewal CSRs for the group "system:nodes"
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: auto-approve-renewals-for-nodes
subjects:
- kind: Group
  name: system:nodes
  apiGroup: rbac.authorization.k8s.io
roleRef:
  kind: ClusterRole
  name: system:certificates.k8s.io:certificatesigningrequests:selfnodelclient
  apiGroup: rbac.authorization.k8s.io

```

The csrapproving controller that ships as part of [kube-controller-manager](#) and is enabled by default. The controller uses the [SubjectAccessReview API](#) to determine if a given user is authorized to request a CSR, then approves based on the authorization outcome. To prevent conflicts with other approvers, the built-in approver doesn't explicitly deny CSRs. It only ignores unauthorized requests. The controller also prunes expired certificates as part of garbage collection.

kubelet configuration

Finally, with the control plane nodes properly set up and all of the necessary authentication and authorization in place, we can configure the kubelet.

The kubelet requires the following configuration to bootstrap:

- A path to store the key and certificate it generates (optional, can use default)
- A path to a kubeconfig file that does not yet exist; it will place the bootstrapped config file here
- A path to a bootstrap kubeconfig file to provide the URL for the server and bootstrap credentials, e.g. a bootstrap token
- Optional: instructions to rotate certificates

The bootstrap kubeconfig should be in a path available to the kubelet, for example /var/lib/kubelet/bootstrap-kubeconfig.

Its format is identical to a normal kubeconfig file. A sample file might look as follows:

```
apiVersion: v1
kind: Config
clusters:
- cluster:
  certificate-authority: /var/lib/kubernetes/ca.pem
  server: https://my.server.example.com:6443
  name: bootstrap
contexts:
- context:
  cluster: bootstrap
  user: kubelet-bootstrap
  name: bootstrap
current-context: bootstrap
preferences: {}
users:
- name: kubelet-bootstrap
  user:
    token: 07401b.f395accd246ae52d
```

The important elements to note are:

- **certificate-authority**: path to a CA file, used to validate the server certificate presented by kube-apiserver
- **server**: URL to kube-apiserver
- **token**: the token to use

The format of the token does not matter, as long as it matches what kube-apiserver expects. In the above example, we used a bootstrap token. As stated earlier, *any* valid authentication method can be used, not only tokens.

Because the bootstrap kubeconfig *is* a standard kubeconfig, you can use kubectl to generate it. To create the above example file:

```
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig set-cluster bootstrap --
server='https://my.server.example.com:6443' --certificate-authority=/var/lib/kubernetes/ca.pem
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig set-credentials kubelet-
bootstrap --token=07401b.f395accd246ae52d
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig set-context bootstrap --
user=kubelet-bootstrap --cluster=bootstrap
kubectl config --kubeconfig=/var/lib/kubelet/bootstrap-kubeconfig use-context bootstrap
```

To indicate to the kubelet to use the bootstrap kubeconfig, use the following kubelet flag:

```
--bootstrap-kubeconfig="/var/lib/kubelet/bootstrap-kubeconfig" --kubeconfig="/var/lib/kubelet/
kubeconfig"
```

When starting the kubelet, if the file specified via `--kubeconfig` does not exist, the bootstrap kubeconfig specified via `--bootstrap-kubeconfig` is used to request a client certificate from the API server. On approval of the certificate request and receipt back by the kubelet, a kubeconfig file referencing the generated key and obtained certificate is written to the path specified by `--kubeconfig`. The certificate and key file will be placed in the directory specified by `--cert-dir`.

Client and serving certificates

All of the above relate to kubelet *client* certificates, specifically, the certificates a kubelet uses to authenticate to kube-apiserver.

A kubelet also can use *serving* certificates. The kubelet itself exposes an https endpoint for certain features. To secure these, the kubelet can do one of:

- use provided key and certificate, via the --tls-private-key-file and --tls-cert-file flags
- create self-signed key and certificate, if a key and certificate are not provided
- request serving certificates from the cluster server, via the CSR API

The client certificate provided by TLS bootstrapping is signed, by default, for client auth only, and thus cannot be used as serving certificates, or server auth.

However, you *can* enable its server certificate, at least partially, via certificate rotation.

Certificate rotation

Kubernetes v1.8 and higher kubelet implements features for enabling rotation of its client and/or serving certificates. Note, rotation of serving certificate is a **beta** feature and requires the RotateKubeletServerCertificate feature flag on the kubelet (enabled by default).

You can configure the kubelet to rotate its client certificates by creating new CSRs as its existing credentials expire. To enable this feature, use the rotateCertificates field of [kubelet configuration file](#) or pass the following command line argument to the kubelet (deprecated):

```
--rotate-certificates
```

Enabling RotateKubeletServerCertificate causes the kubelet **both** to request a serving certificate after bootstrapping its client credentials **and** to rotate that certificate. To enable this behavior, use the field serverTLSBootstrap of the [kubelet configuration file](#) or pass the following command line argument to the kubelet (deprecated):

```
--rotate-server-certificates
```

Note:

The CSR approving controllers implemented in core Kubernetes do not approve node *serving* certificates for [security reasons](#). To use RotateKubeletServerCertificate operators need to run a custom approving controller, or manually approve the serving certificate requests.

A deployment-specific approval process for kubelet serving certificates should typically only approve CSRs which:

1. are requested by nodes (ensure the spec.username field is of the form system:node:<nodeName> and spec.groups contains system:nodes)
2. request usages for a serving certificate (ensure spec.usages contains server auth, optionally contains digital signature and key encipherment, and contains no other usages)
3. only have IP and DNS subjectAltNames that belong to the requesting node, and have no URI and Email subjectAltNames (parse the x509 Certificate Signing Request in spec.request to verify subjectAltNames)

Other authenticating components

All of TLS bootstrapping described in this document relates to the kubelet. However, other components may need to communicate directly with kube-apiserver. Notable is kube-proxy, which is part of the Kubernetes node components and runs on every node, but may also include other components such as monitoring or networking.

Like the kubelet, these other components also require a method of authenticating to kube-apiserver. You have several options for generating these credentials:

- The old way: Create and distribute certificates the same way you did for kubelet before TLS bootstrapping
- DaemonSet: Since the kubelet itself is loaded on each node, and is sufficient to start base services, you can run kube-proxy and other node-specific services not as a standalone process, but rather as a daemonset in the kube-system namespace. Since it will be in-cluster, you can give it a proper service account with appropriate permissions to perform its activities. This may be the simplest way to configure such services.

kubectl approval

CSRs can be approved outside of the approval flows built into the controller manager.

The signing controller does not immediately sign all certificate requests. Instead, it waits until they have been flagged with an "Approved" status by an appropriately-privileged user. This flow is intended to allow for automated approval handled by an external approval controller or the approval controller implemented in the core controller-manager. However cluster administrators can also manually approve certificate requests using kubectl. An administrator can list CSRs with `kubectl get csr` and describe one in detail with `kubectl describe csr <name>`. An administrator can approve or deny a CSR with `kubectl certificate approve <name>` and `kubectl certificate deny <name>`.

Validating Admission Policy

FEATURE STATE: Kubernetes v1.28 [beta]

This page provides an overview of Validating Admission Policy.

What is Validating Admission Policy?

Validating admission policies offer a declarative, in-process alternative to validating admission webhooks.

Validating admission policies use the Common Expression Language (CEL) to declare the validation rules of a policy. Validation admission policies are highly configurable, enabling policy authors to define policies that can be parameterized and scoped to resources as needed by cluster administrators.

What Resources Make a Policy

A policy is generally made up of three resources:

- The `ValidatingAdmissionPolicy` describes the abstract logic of a policy (think: "this policy makes sure a particular label is set to a particular value").
- A `ValidatingAdmissionPolicyBinding` links the above resources together and provides scoping. If you only want to require an owner label to be set for Pods, the binding is where you would specify this restriction.
- A parameter resource provides information to a `ValidatingAdmissionPolicy` to make it a concrete statement (think "the owner label must be set to something that ends in `.company.com`"). A native type such as `ConfigMap` or a CRD defines the schema of a parameter resource. `ValidatingAdmissionPolicy` objects specify what `Kind` they are expecting for their parameter resource.

At least a `ValidatingAdmissionPolicy` and a corresponding `ValidatingAdmissionPolicyBinding` must be defined for a policy to have an effect.

If a `ValidatingAdmissionPolicy` does not need to be configured via parameters, simply leave `spec.paramKind` in `ValidatingAdmissionPolicy` not specified.

Before you begin

- Ensure the `ValidatingAdmissionPolicy` [feature gate](#) is enabled.
- Ensure that the `admissionregistration.k8s.io/v1beta1` API is enabled.

Getting Started with Validating Admission Policy

Validating Admission Policy is part of the cluster control-plane. You should write and deploy them with great caution. The following describes how to quickly experiment with Validating Admission Policy.

Creating a `ValidatingAdmissionPolicy`

The following is an example of a `ValidatingAdmissionPolicy`.

[`validatingadmissionpolicy/basic-example-policy.yaml`](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicy
metadata:
  name: "demo-policy.example.com"
spec:
  failurePolicy: Fail
  matchConstraints:
  - resourceRules:
    - apiGroups: ["apps"]
      apiVersions: ["v1"]
      operations: ["CREATE", "UPDATE"]
```

```
resources: ["deployments"]
validations:
  - expression: "object.spec.replicas <= 5"
```

spec.validations contains CEL expressions which use the [Common Expression Language \(CEL\)](#) to validate the request. If an expression evaluates to false, the validation check is enforced according to the spec.failurePolicy field.

Note: You can quickly test CEL expressions in [CEL Playground](#).

To configure a validating admission policy for use in a cluster, a binding is required. The following is an example of a ValidatingAdmissionPolicyBinding.:

[validatingadmissionpolicy/basic-example-binding.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "demo-binding-test.example.com"
spec:
  policyName: "demo-policy.example.com"
  validationActions: [Deny]
  matchResources:
    namespaceSelector:
      matchLabels:
        environment: test
```

When trying to create a deployment with replicas set not satisfying the validation expression, an error will return containing message:

```
ValidatingAdmissionPolicy 'demo-policy.example.com' with binding 'demo-binding-test.example.com' denied request: failed expression: object.spec.replicas <= 5
```

The above provides a simple example of using ValidatingAdmissionPolicy without a parameter configured.

Validation actions

Each ValidatingAdmissionPolicyBinding must specify one or more validationActions to declare how validations of a policy are enforced.

The supported validationActions are:

- Deny: Validation failure results in a denied request.
- Warn: Validation failure is reported to the request client as a [warning](#).
- Audit: Validation failure is included in the audit event for the API request.

For example, to both warn clients about a validation failure and to audit the validation failures, use:

```
validationActions: [Warn, Audit]
```

Deny and Warn may not be used together since this combination needlessly duplicates the validation failure both in the API response body and the HTTP warning headers.

A validation that evaluates to false is always enforced according to these actions. Failures defined by the failurePolicy are enforced according to these actions only if the failurePolicy is set to Fail (or not specified), otherwise the failures are ignored.

See [Audit Annotations: validation failures](#) for more details about the validation failure audit annotation.

Parameter resources

Parameter resources allow a policy configuration to be separate from its definition. A policy can define paramKind, which outlines GVK of the parameter resource, and then a policy binding ties a policy by name (via policyName) to a particular parameter resource via paramRef.

If parameter configuration is needed, the following is an example of a ValidatingAdmissionPolicy with parameter configuration.

[validatingadmissionpolicy/policy-with-param.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicy
metadata:
  name: "replicalimit-policy.example.com"
spec:
  failurePolicy: Fail
  paramKind:
    apiVersion: rules.example.com/v1
    kind: ReplicaLimit
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments"]
  validations:
    - expression: "object.spec.replicas <= params.maxReplicas"
      reason: Invalid
```

The spec.paramKind field of the ValidatingAdmissionPolicy specifies the kind of resources used to parameterize this policy. For this example, it is configured by ReplicaLimit custom resources. Note in this example how the CEL expression references the parameters via the CEL params variable, e.g. params.maxReplicas. spec.matchConstraints specifies what resources this policy is designed to validate. Note that the native types such like ConfigMap could also be used as parameter reference.

The spec.validations fields contain CEL expressions. If an expression evaluates to false, the validation check is enforced according to the spec.failurePolicy field.

The validating admission policy author is responsible for providing the ReplicaLimit parameter CRD.

To configure an validating admission policy for use in a cluster, a binding and parameter resource are created. The following is an example of a ValidatingAdmissionPolicyBinding that uses a **cluster-wide** param - the same param will be used to validate every resource request that matches the binding:

[validatingadmissionpolicy/binding-with-param.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "replicalimit-binding-test.example.com"
spec:
  policyName: "replicalimit-policy.example.com"
  validationActions: [Deny]
  paramRef:
    name: "replica-limit-test.example.com"
    namespace: "default"
matchResources:
  namespaceSelector:
    matchLabels:
      environment: test
```

Notice this binding applies a parameter to the policy for all resources which are in the test environment.

The parameter resource could be as following:

[validatingadmissionpolicy/replicalimit-param.yaml](#)

```
apiVersion: rules.example.com/v1
kind: ReplicaLimit
metadata:
  name: "replica-limit-test.example.com"
  namesapce: "default"
maxReplicas: 3
```

This policy parameter resource limits deployments to a max of 3 replicas.

An admission policy may have multiple bindings. To bind all other environments to have a maxReplicas limit of 100, create another ValidatingAdmissionPolicyBinding:

[validatingadmissionpolicy/binding-with-param-prod.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicyBinding
metadata:
  name: "replicalimit-binding-nontest"
spec:
  policyName: "replicalimit-policy.example.com"
  validationActions: [Deny]
  paramRef:
    name: "replica-limit-prod.example.com"
    namespace: "default"
matchResources:
  namespaceSelector:
    matchExpressions:
      - key: environment
        operator: NotIn
```

```
values:  
- test
```

Notice this binding applies a different parameter to resources which are not in the test environment.

And have a parameter resource:

[validatingadmissionpolicy/replicalimit-param-prod.yaml](#)

```
apiVersion: rules.example.com/v1  
kind: ReplicaLimit  
metadata:  
  name: "replica-limit-prod.example.com"  
maxReplicas: 100
```

For each admission request, the API server evaluates CEL expressions of each (policy, binding, param) combination that match the request. For a request to be admitted it must pass **all** evaluations.

If multiple bindings match the request, the policy will be evaluated for each, and they must all pass evaluation for the policy to be considered passed.

If multiple parameters match a single binding, the policy rules will be evaluated for each param, and they too must all pass for the binding to be considered passed. Bindings can have overlapping match criteria. The policy is evaluated for each matching binding-parameter combination. A policy may even be evaluated multiple times if multiple bindings match it, or a single binding that matches multiple parameters.

The params object representing a parameter resource will not be set if a parameter resource has not been bound, so for policies requiring a parameter resource, it can be useful to add a check to ensure one has been bound. A parameter resource will not be bound and params will be null if paramKind of the policy, or paramRef of the binding are not specified.

For the use cases require parameter configuration, we recommend to add a param check in spec.validations[0].expression:

```
- expression: "params != null"  
  message: "params missing but required to bind to this policy"
```

Optional parameters

It can be convenient to be able to have optional parameters as part of a parameter resource, and only validate them if present. CEL provides has(), which checks if the key passed to it exists. CEL also implements Boolean short-circuiting. If the first half of a logical OR evaluates to true, it won't evaluate the other half (since the result of the entire OR will be true regardless).

Combining the two, we can provide a way to validate optional parameters:

```
!has(params.optionalNumber) || (params.optionalNumber >= 5 && params.optionalNumber <= 10)
```

Here, we first check that the optional parameter is present with `!has(params.optionalNumber)`.

- If `optionalNumber` hasn't been defined, then the expression short-circuits since `!has(params.optionalNumber)` will evaluate to true.
- If `optionalNumber` has been defined, then the latter half of the CEL expression will be evaluated, and `optionalNumber` will be checked to ensure that it contains a value between 5 and 10 inclusive.

Per-namespace Parameters

As the author of a `ValidatingAdmissionPolicy` and its `ValidatingAdmissionPolicyBinding`, you can choose to specify cluster-wide, or per-namespace parameters. If you specify a namespace for the binding's `paramRef`, the control plane only searches for parameters in that namespace.

However, if namespace is not specified in the `ValidatingAdmissionPolicyBinding`, the API server can search for relevant parameters in the namespace that a request is against. For example, if you make a request to modify a `ConfigMap` in the default namespace and there is a relevant `ValidatingAdmissionPolicyBinding` with no namespace set, then the API server looks for a parameter object in default. This design enables policy configuration that depends on the namespace of the resource being manipulated, for more fine-tuned control.

Parameter selector

In addition to specifying a parameter by name, you may choose instead to specify label selector, such that all resources of the policy's `paramKind`, and the param's namespace (if applicable) that match the label selector are selected for evaluation. See [selector](#) for more information on how label selectors match resources.

If multiple parameters are found to meet the condition, the policy's rules are evaluated for each parameter found and the results will be ANDed together.

If namespace is provided, only objects of the `paramKind` in the provided namespace are eligible for selection. Otherwise, when namespace is empty and `paramKind` is namespace-scoped, the namespace used in the request being admitted will be used.

Authorization checks

We introduced the authorization check for parameter resources. User is expected to have read access to the resources referenced by `paramKind` in `ValidatingAdmissionPolicy` and `paramRef` in `ValidatingAdmissionPolicyBinding`.

Note that if a resource in `paramKind` fails resolving via the restmapper, read access to all resources of groups is required.

Failure Policy

`failurePolicy` defines how mis-configurations and CEL expressions evaluating to error from the admission policy are handled. Allowed values are `Ignore` or `Fail`.

- `Ignore` means that an error calling the `ValidatingAdmissionPolicy` is ignored and the API request is allowed to continue.

- Fail means that an error calling the ValidatingAdmissionPolicy causes the admission to fail and the API request to be rejected.

Note that the failurePolicy is defined inside ValidatingAdmissionPolicy:

[validatingadmissionpolicy/failure-policy-ignore.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicy
spec:
...
failurePolicy: Ignore # The default is "Fail"
validations:
- expression: "object.spec.xyz == params.x"
```

Validation Expression

spec.validations[i].expression represents the expression which will be evaluated by CEL. To learn more, see the [CEL language specification](#) CEL expressions have access to the contents of the Admission request/response, organized into CEL variables as well as some other useful variables:

- 'object' - The object from the incoming request. The value is null for DELETE requests.
- 'oldObject' - The existing object. The value is null for CREATE requests.
- 'request' - Attributes of the [admission request](#).
- 'params' - Parameter resource referred to by the policy binding being evaluated. The value is null if ParamKind is not specified.
- namespaceObject - The namespace, as a Kubernetes resource, that the incoming object belongs to. The value is null if the incoming object is cluster-scoped.
- authorizer - A CEL Authorizer. May be used to perform authorization checks for the principal (authenticated user) of the request. See [Authz](#) in the Kubernetes CEL library documentation for more details.
- authorizer.requestResource - A shortcut for an authorization check configured with the request resource (group, resource, (subresource), namespace, name).

The apiVersion, kind, metadata.name and metadata.generateName are always accessible from the root of the object. No other metadata properties are accessible.

Equality on arrays with list type of 'set' or 'map' ignores element order, i.e. [1, 2] == [2, 1]. Concatenation on arrays with x-kubernetes-list-type use the semantics of the list type:

- 'set': X + Y performs a union where the array positions of all elements in X are preserved and non-intersecting elements in Y are appended, retaining their partial order.
- 'map': X + Y performs a merge where the array positions of all keys in X are preserved but the values are overwritten by values in Y when the key sets of X and Y intersect. Elements in Y with non-intersecting keys are appended, retaining their partial order.

Validation expression examples

Expression	Purpose
object.minReplicas <= object.replicas && object.replicas <= object.maxReplicas	Validate that the three fields defining replicas are ordered appropriately
'Available' in object.stateCounts	

Expression	Purpose
	Validate that an entry with the 'Available' key exists in a map
(size(object.list1) == 0) != (size(object.list2) == 0)	Validate that one of two lists is non-empty, but not both
!(‘MY_KEY’ in object.map1) object[‘MY_KEY’].matches(‘^ [a-zA-Z]*\$’)	Validate the value of a map for a specific key, if it is in the map
object.envars.filter(e, e.name == ‘MY_ENV’).all(e, e.value.matches(‘^ [a-zA-Z]*\$’))	Validate the ‘value’ field of a listMap entry where key field ‘name’ is ‘MY_ENV’
has(object.expired) && object.created + object.ttl < object.expired	Validate that ‘expired’ date is after a ‘create’ date plus a ‘ttl’ duration
object.health.startsWith(‘ok’)	Validate a ‘health’ string field has the prefix ‘ok’
object.widgets.exists(w, w.key == ‘x’ && w.foo < 10)	Validate that the ‘foo’ property of a listMap item with a key ‘x’ is less than 10
type(object) == string ? object == ‘100%’ : object == 1000	Validate an int-or-string field for both the int and string cases
object.metadata.name.startsWith(object.prefix)	Validate that an object’s name has the prefix of another field value
object.set1.all(e, !(e in object.set2))	Validate that two listSets are disjoint
size(object.names) == size(object.details) && object.names.all(n, n in object.details)	Validate the ‘details’ map is keyed by the items in the ‘names’ listSet
size(object.clusters.filter(c, c.name == object.primary)) == 1	Validate that the ‘primary’ property has one and only one occurrence in the ‘clusters’ listMap

Read [Supported evaluation on CEL](#) for more information about CEL rules.

spec.validation[i].reason represents a machine-readable description of why this validation failed. If this is the first validation in the list to fail, this reason, as well as the corresponding HTTP response code, are used in the HTTP response to the client. The currently supported reasons are: Unauthorized, Forbidden, Invalid, RequestEntityTooLarge. If not set, StatusReasonInvalid is used in the response to the client.

Matching requests: matchConditions

You can define *match conditions* for a ValidatingAdmissionPolicy if you need fine-grained request filtering. These conditions are useful if you find that match rules, objectSelectors and namespaceSelectors still doesn’t provide the filtering you want. Match conditions are [CEL expressions](#). All match conditions must evaluate to true for the resource to be evaluated.

Here is an example illustrating a few different uses for match conditions:

[access/validating-admission-policy-match-conditions.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1alpha1
kind: ValidatingAdmissionPolicy
metadata:
  name: "demo-policy.example.com"
```

```

spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["*"]
        apiVersions: ["*"]
        operations: ["CREATE", "UPDATE"]
        resources: ["*"]
  matchConditions:
    - name: 'exclude-leases' # Each match condition must have a unique name
      expression: '!(request.resource.group == "coordination.k8s.io" && request.resource.resource == "leases")' # Match non-lease resources.
    - name: 'exclude-kubelet-requests'
      expression: '!("system:nodes" in request.userInfo.groups)' # Match requests made by non-node users.
    - name: 'rbac' # Skip RBAC requests.
      expression: 'request.resource.group != "rbac.authorization.k8s.io"'
  validations:
    - expression: "!object.metadata.name.contains('demo') || object.metadata.namespace == 'demo'"

```

Match conditions have access to the same CEL variables as validation expressions.

In the event of an error evaluating a match condition the policy is not evaluated. Whether to reject the request is determined as follows:

1. If **any** match condition evaluated to false (regardless of other errors), the API server skips the policy.
2. Otherwise:
 - for [failurePolicy: Fail](#), reject the request (without evaluating the policy).
 - for [failurePolicy: Ignore](#), proceed with the request but skip the policy.

Audit annotations

`auditAnnotations` may be used to include audit annotations in the audit event of the API request.

For example, here is an admission policy with an audit annotation:

[access/validating-admission-policy-audit-annotation.yaml](#)

```

apiVersion: admissionregistration.k8s.io/v1alpha1
kind: ValidatingAdmissionPolicy
metadata:
  name: "demo-policy.example.com"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]

```

```

resources: ["deployments"]
validations:
  - key: "high-replica-count"
    expression: "object.spec.replicas > 50"
    messageExpression: "Deployment spec.replicas set to ' + string(object.spec.replicas)"

```

When an API request is validated with this admission policy, the resulting audit event will look like:

```

# the audit event recorded
{
  "kind": "Event",
  "apiVersion": "audit.k8s.io/v1",
  "annotations": {
    "demo-policy.example.com/high-replica-count": "Deployment spec.replicas set to 128"
    # other annotations
    ...
  }
  # other fields
  ...
}

```

In this example the annotation will only be included if the spec.replicas of the Deployment is more than 50, otherwise the CEL expression evaluates to null and the annotation will not be included.

Note that audit annotation keys are prefixed by the name of the ValidatingAdmissionWebhook and a /. If another admission controller, such as an admission webhook, uses the exact same audit annotation key, the value of the first admission controller to include the audit annotation will be included in the audit event and all other values will be ignored.

Message expression

To return a more friendly message when the policy rejects a request, we can use a CEL expression to composite a message with spec.validations[i].messageExpression. Similar to the validation expression, a message expression has access to object, oldObject, request, params, and namespaceObject. Unlike validations, message expression must evaluate to a string.

For example, to better inform the user of the reason of denial when the policy refers to a parameter, we can have the following validation:

[access/deployment-replicas-policy.yaml](#)

```

apiVersion: admissionregistration.k8s.io/v1alpha1
kind: ValidatingAdmissionPolicy
metadata:
  name: "deploy-replica-policy.example.com"
spec:
  paramKind:
    apiVersion: rules.example.com/v1
    kind: ReplicaLimit
  matchConstraints:
    resourceRules:

```

```
- apiGroups: ["apps"]
  apiVersions: ["v1"]
  operations: ["CREATE", "UPDATE"]
  resources: ["deployments"]
validations:
- expression: "object.spec.replicas <= params.maxReplicas"
  messageExpression: "object.spec.replicas must be no greater than ' + string(params.maxReplicas)"
  reason: Invalid
```

After creating a params object that limits the replicas to 3 and setting up the binding, when we try to create a deployment with 5 replicas, we will receive the following message.

```
$ kubectl create deploy --image=nginx nginx --replicas=5
error: failed to create deployment: deployments.apps "nginx" is forbidden:
ValidatingAdmissionPolicy 'deploy-replica-policy.example.com' with binding 'demo-binding-test.example.com' denied request: object.spec.replicas must be no greater than 3
```

This is more informative than a static message of "too many replicas".

The message expression takes precedence over the static message defined in spec.validations[i].message if both are defined. However, if the message expression fails to evaluate, the static message will be used instead. Additionally, if the message expression evaluates to a multi-line string, the evaluation result will be discarded and the static message will be used if present. Note that static message is validated against multi-line strings.

Type checking

When a policy definition is created or updated, the validation process parses the expressions it contains and reports any syntax errors, rejecting the definition if any errors are found. Afterward, the referred variables are checked for type errors, including missing fields and type confusion, against the matched types of spec.matchConstraints. The result of type checking can be retrieved from status.typeChecking. The presence of status.typeChecking indicates the completion of type checking, and an empty status.typeChecking means that no errors were detected.

For example, given the following policy definition:

[validatingadmissionpolicy/typechecking.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicy
metadata:
  name: "deploy-replica-policy.example.com"
spec:
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
        resources: ["deployments"]
  validations:
```

```
- expression: "object.replicas > 1" # should be "object.spec.replicas > 1"
  message: "must be replicated"
  reason: Invalid
```

The status will yield the following information:

```
status:
typeChecking:
expressionWarnings:
- fieldRef: spec.validations[0].expression
warning: |-  
  apps/v1, Kind=Deployment: ERROR: <input>:1:7: undefined field 'replicas'  
  | object.replicas > 1  
  | .....^
```

If multiple resources are matched in spec.matchConstraints, all of matched resources will be checked against. For example, the following policy definition

[validatingadmissionpolicy/typechecking-multiple-match.yaml](#)

```
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicy
metadata:
  name: "replica-policy.example.com"
spec:
  matchConstraints:
    resourceRules:
    - apiGroups: ["apps"]
      apiVersions: ["v1"]
      operations: ["CREATE", "UPDATE"]
      resources: ["deployments", "replicaset"]
  validations:
  - expression: "object.replicas > 1" # should be "object.spec.replicas > 1"
    message: "must be replicated"
    reason: Invalid
```

will have multiple types and type checking result of each type in the warning message.

```
status:
typeChecking:
expressionWarnings:
- fieldRef: spec.validations[0].expression
warning: |-  
  apps/v1, Kind=Deployment: ERROR: <input>:1:7: undefined field 'replicas'  
  | object.replicas > 1  
  | .....^  
  apps/v1, Kind=ReplicaSet: ERROR: <input>:1:7: undefined field 'replicas'  
  | object.replicas > 1  
  | .....^
```

Type Checking has the following limitation:

- No wildcard matching. If spec.matchConstraints.resourceRules contains "*" in any of apiGroups, apiVersions or resources, the types that "*" matches will not be checked.

- The number of matched types is limited to 10. This is to prevent a policy that manually specifying too many types. to consume excessive computing resources. In the order of ascending group, version, and then resource, 11th combination and beyond are ignored.
- Type Checking does not affect the policy behavior in any way. Even if the type checking detects errors, the policy will continue to evaluate. If errors do occur during evaluate, the failure policy will decide its outcome.
- Type Checking does not apply to CRDs, including matched CRD types and reference of paramKind. The support for CRDs will come in future release.

Variable composition

If an expression grows too complicated, or part of the expression is reusable and computationally expensive to evaluate, you can extract some part of the expressions into variables. A variable is a named expression that can be referred later in variables in other expressions.

```
spec:
variables:
- name: foo
  expression: "foo" in object.spec.metadata.labels ? object.spec.metadata.labels['foo'] : 'default'
validations:
- expression: variables.foo == 'bar'
```

A variable is lazily evaluated when it is first referred. Any error that occurs during the evaluation will be reported during the evaluation of the referring expression. Both the result and potential error are memorized and count only once towards the runtime cost.

The order of variables are important because a variable can refer to other variables that are defined before it. This ordering prevents circular references.

The following is a more complex example of enforcing that image repo names match the environment defined in its namespace.

[access/image-matches-namespace-environment.policy.yaml](#)

```
# This policy enforces that all containers of a deployment has the image repo match the
environment label of its namespace.
# Except for "exempt" deployments, or any containers that do not belong to the "example.com"
organization (e.g. common sidecars).
# For example, if the namespace has a label of {"environment": "staging"}, all container images
must be either staging.example.com/*
# or do not contain "example.com" at all, unless the deployment has {"exempt": "true"} label.
apiVersion: admissionregistration.k8s.io/v1beta1
kind: ValidatingAdmissionPolicy
metadata:
  name: "image-matches-namespace-environment.policy.example.com"
spec:
  failurePolicy: Fail
  matchConstraints:
    resourceRules:
      - apiGroups: ["apps"]
        apiVersions: ["v1"]
        operations: ["CREATE", "UPDATE"]
```

```

resources: ["deployments"]
variables:
- name: environment
  expression: "'environment' in namespaceObject.metadata.labels ?
namespaceObject.metadata.labels['environment'] : 'prod'"
- name: exempt
  expression: "'exempt' in object.metadata.labels && object.metadata.labels['exempt'] == 'true'"
- name: containers
  expression: "object.spec.template.spec.containers"
- name: containersToCheck
  expression: "variables.containers.filter(c, c.image.contains('example.com/'))"
validations:
- expression: "variables.exempt || variables.containersToCheck.all(c,
c.image.startsWith(variables.environment + ':'))"
  messageExpression: "only " + variables.environment + ' images are allowed in namespace ' +
namespaceObject.metadata.name"

```

With the policy bound to the namespace default, which is labeled environment: prod, the following attempt to create a deployment would be rejected.

```
kubectl create deploy --image=dev.example.com/nginx invalid
```

The error message is similar to this.

```

error: failed to create deployment: deployments.apps "invalid" is forbidden:
ValidatingAdmissionPolicy 'image-matches-namespace-environment.policy.example.com' with
binding 'demo-binding-test.example.com' denied request: only prod images are allowed in
namespace default

```

Well-Known Labels, Annotations and Taints

Kubernetes reserves all labels and annotations in the kubernetes.io and k8s.io namespaces.

This document serves both as a reference to the values and as a coordination point for assigning values.

Labels, annotations and taints used on API objects

`apf.kubernetes.io/autoupdate-spec`

Type: Annotation

Example: `apf.kubernetes.io/autoupdate-spec: "true"`

Used on: [FlowSchema and PriorityLevelConfiguration Objects](#)

If this annotation is set to true on a FlowSchema or PriorityLevelConfiguration, the spec for that object is managed by the kube-apiserver. If the API server does not recognize an APF object, and you annotate it for automatic update, the API server deletes the entire object.

Otherwise, the API server does not manage the object spec. For more details, read [Maintenance of the Mandatory and Suggested Configuration Objects](#).

app.kubernetes.io/component

Type: Label

Example: app.kubernetes.io/component: "database"

Used on: All Objects (typically used on [workload resources](#)).

The component within the application architecture.

One of the [recommended labels](#).

app.kubernetes.io/created-by (deprecated)

Type: Label

Example: app.kubernetes.io/created-by: "controller-manager"

Used on: All Objects (typically used on [workload resources](#)).

The controller/user who created this resource.

Note: Starting from v1.9, this label is deprecated.

app.kubernetes.io/instance

Type: Label

Example: app.kubernetes.io/instance: "mysql-abcxzy"

Used on: All Objects (typically used on [workload resources](#)).

A unique name identifying the instance of an application. To assign a non-unique name, use [app.kubernetes.io/name](#).

One of the [recommended labels](#).

app.kubernetes.io/managed-by

Type: Label

Example: app.kubernetes.io/managed-by: "helm"

Used on: All Objects (typically used on [workload resources](#)).

The tool being used to manage the operation of an application.

One of the [recommended labels](#).

app.kubernetes.io/name

Type: Label

Example: app.kubernetes.io/name: "mysql"

Used on: All Objects (typically used on [workload resources](#)).

The name of the application.

One of the [recommended labels](#).

app.kubernetes.io/part-of

Type: Label

Example: app.kubernetes.io/part-of: "wordpress"

Used on: All Objects (typically used on [workload resources](#)).

The name of a higher-level application this object is part of.

One of the [recommended labels](#).

app.kubernetes.io/version

Type: Label

Example: app.kubernetes.io/version: "5.7.21"

Used on: All Objects (typically used on [workload resources](#)).

The current version of the application.

Common forms of values include:

- [semantic version](#)
- the Git [revision hash](#) for the source code.

One of the [recommended labels](#).

applyset.kubernetes.io/additional-namespaces (alpha)

Type: Annotation

Example: applyset.kubernetes.io/additional-namespaces: "namespace1,namespace2"

Used on: Objects being used as ApplySet parents.

Use of this annotation is Alpha. For Kubernetes version 1.28, you can use this annotation on Secrets, ConfigMaps, or custom resources if the [CustomResourceDefinition](#) defining them has the applyset.kubernetes.io/is-parent-type label.

Part of the specification used to implement [ApplySet-based pruning in kubectl](#). This annotation is applied to the parent object used to track an ApplySet to extend the scope of the ApplySet beyond the parent object's own namespace (if any). The value is a comma-separated list of the names of namespaces other than the parent's namespace in which objects are found.

applyset.kubernetes.io/contains-group-resources (alpha)

Type: Annotation

Example: applyset.kubernetes.io/contains-group-resources: "certificates.cert-manager.io,configmaps,deployments.apps,secrets,services"

Used on: Objects being used as ApplySet parents.

Use of this annotation is Alpha. For Kubernetes version 1.28, you can use this annotation on Secrets, ConfigMaps, or custom resources if the CustomResourceDefinition defining them has the applyset.kubernetes.io/is-parent-type label.

Part of the specification used to implement [ApplySet-based pruning in kubectl](#). This annotation is applied to the parent object used to track an ApplySet to optimize listing of ApplySet member objects. It is optional in the ApplySet specification, as tools can perform discovery or use a different optimization. However, as of Kubernetes version 1.28, it is required by kubectl. When present, the value of this annotation must be a comma separated list of the group-kinds, in the fully-qualified name format, i.e. <resource>.<group>.

applyset.kubernetes.io/id (alpha)

Type: Label

Example: applyset.kubernetes.io/id:
"applyset-0eFHV8ySqp7XoShsGvyWFQD3s96yqwHmzc4e0HR1dsY-v1"

Used on: Objects being used as ApplySet parents.

Use of this label is Alpha. For Kubernetes version 1.28, you can use this label on Secrets, ConfigMaps, or custom resources if the CustomResourceDefinition defining them has the applyset.kubernetes.io/is-parent-type label.

Part of the specification used to implement [ApplySet-based pruning in kubectl](#). This label is what makes an object an ApplySet parent object. Its value is the unique ID of the ApplySet, which is derived from the identity of the parent object itself. This ID **must** be the base64 encoding (using the URL safe encoding of RFC4648) of the hash of the group-kind-name-namespace of the object it is on, in the form:

<base64(sha256(<name>.<namespace>.<kind>.<group>))>. There is no relation between the value of this label and object UID.

applyset.kubernetes.io/is-parent-type (alpha)

Type: Label

Example: applyset.kubernetes.io/is-parent-type: "true"

Used on: Custom Resource Definition (CRD)

Use of this label is Alpha. Part of the specification used to implement [ApplySet-based pruning in kubectl](#). You can set this label on a CustomResourceDefinition (CRD) to identify the custom resource type it defines (not the CRD itself) as an allowed parent for an ApplySet. The only permitted value for this label is "true"; if you want to mark a CRD as not being a valid parent for ApplySets, omit this label.

applyset.kubernetes.io/part-of (alpha)

Type: Label

Example: applyset.kubernetes.io/part-of:

"applyset-0eFHV8ySqp7XoShsGvyWFQD3s96yqwHmzc4e0HR1dsY-v1"

Used on: All objects.

Use of this label is Alpha. Part of the specification used to implement [ApplySet-based pruning in kubectl](#). This label is what makes an object a member of an ApplySet. The value of the label **must** match the value of the applyset.kubernetes.io/id label on the parent object.

applyset.kubernetes.io/tooling (alpha)

Type: Annotation

Example: applyset.kubernetes.io/tooling: "kubectl/v1.28"

Used on: Objects being used as ApplySet parents.

Use of this annotation is Alpha. For Kubernetes version 1.28, you can use this annotation on Secrets, ConfigMaps, or custom resources if the CustomResourceDefinition defining them has the applyset.kubernetes.io/is-parent-type label.

Part of the specification used to implement [ApplySet-based pruning in kubectl](#). This annotation is applied to the parent object used to track an ApplySet to indicate which tooling manages that ApplySet. Tooling should refuse to mutate ApplySets belonging to other tools. The value must be in the format <toolname>/<semver>.

apps.kubernetes.io/pod-index (beta)

Type: Label

Example: apps.kubernetes.io/pod-index: "0"

Used on: Pod

When a StatefulSet controller creates a Pod for the StatefulSet, it sets this label on that Pod. The value of the label is the ordinal index of the pod being created.

See [Pod Index Label](#) in the StatefulSet topic for more details. Note the [PodIndexLabel](#) feature gate must be enabled for this label to be added to pods.

cluster-autoscaler.kubernetes.io/safe-to-evict

Type: Annotation

Example: cluster-autoscaler.kubernetes.io/safe-to-evict: "true"

Used on: Pod

When this annotation is set to "true", the cluster autoscaler is allowed to evict a Pod even if other rules would normally prevent that. The cluster autoscaler never evicts Pods that have this annotation explicitly set to "false"; you could set that on an important Pod that you want to keep running. If this annotation is not set then the cluster autoscaler follows its Pod-level behavior.

config.kubernetes.io/local-config

Type: Annotation

Example: config.kubernetes.io/local-config: "true"

Used on: All objects

This annotation is used in manifests to mark an object as local configuration that should not be submitted to the Kubernetes API.

A value of "true" for this annotation declares that the object is only consumed by client-side tooling and should not be submitted to the API server.

A value of "false" can be used to declare that the object should be submitted to the API server even when it would otherwise be assumed to be local.

This annotation is part of the Kubernetes Resource Model (KRM) Functions Specification, which is used by Kustomize and similar third-party tools. For example, Kustomize removes objects with this annotation from its final build output.

container.apparmor.security.beta.kubernetes.io/* (beta)

Type: Annotation

Example: container.apparmor.security.beta.kubernetes.io/my-container: my-custom-profile

Used on: Pods

This annotation allows you to specify the AppArmor security profile for a container within a Kubernetes pod. To learn more, see the [AppArmor](#) tutorial. The tutorial illustrates using AppArmor to restrict a container's abilities and access.

The profile specified dictates the set of rules and restrictions that the containerized process must adhere to. This helps enforce security policies and isolation for your containers.

internal.config.kubernetes.io/* (reserved prefix)

Type: Annotation

Used on: All objects

This prefix is reserved for internal use by tools that act as orchestrators in accordance with the Kubernetes Resource Model (KRM) Functions Specification. Annotations with this prefix are

internal to the orchestration process and are not persisted to the manifests on the filesystem. In other words, the orchestrator tool should set these annotations when reading files from the local filesystem and remove them when writing the output of functions back to the filesystem.

A KRM function **must not** modify annotations with this prefix, unless otherwise specified for a given annotation. This enables orchestrator tools to add additional internal annotations, without requiring changes to existing functions.

internal.config.kubernetes.io/path

Type: Annotation

Example: `internal.config.kubernetes.io/path: "relative/file/path.yaml"`

Used on: All objects

This annotation records the slash-delimited, OS-agnostic, relative path to the manifest file the object was loaded from. The path is relative to a fixed location on the filesystem, determined by the orchestrator tool.

This annotation is part of the Kubernetes Resource Model (KRM) Functions Specification, which is used by Kustomize and similar third-party tools.

A KRM Function **should not** modify this annotation on input objects unless it is modifying the referenced files. A KRM Function **may** include this annotation on objects it generates.

internal.config.kubernetes.io/index

Type: Annotation

Example: `internal.config.kubernetes.io/index: "2"`

Used on: All objects

This annotation records the zero-indexed position of the YAML document that contains the object within the manifest file the object was loaded from. Note that YAML documents are separated by three dashes (--) and can each contain one object. When this annotation is not specified, a value of 0 is implied.

This annotation is part of the Kubernetes Resource Model (KRM) Functions Specification, which is used by Kustomize and similar third-party tools.

A KRM Function **should not** modify this annotation on input objects unless it is modifying the referenced files. A KRM Function **may** include this annotation on objects it generates.

kubernetes.io/arch

Type: Label

Example: `kubernetes.io/arch: "amd64"`

Used on: Node

The Kubelet populates this with runtime.GOARCH as defined by Go. This can be handy if you are mixing ARM and x86 nodes.

kubernetes.io/os

Type: Label

Example: kubernetes.io/os: "linux"

Used on: Node, Pod

For nodes, the kubelet populates this with runtime.GOOS as defined by Go. This can be handy if you are mixing operating systems in your cluster (for example: mixing Linux and Windows nodes).

You can also set this label on a Pod. Kubernetes allows you to set any value for this label; if you use this label, you should nevertheless set it to the Go runtime.GOOS string for the operating system that this Pod actually works with.

When the kubernetes.io/os label value for a Pod does not match the label value on a Node, the kubelet on the node will not admit the Pod. However, this is not taken into account by the kube-scheduler. Alternatively, the kubelet refuses to run a Pod where you have specified a Pod OS, if this isn't the same as the operating system for the node where that kubelet is running. Just look for [Pods OS](#) for more details.

kubernetes.io/metadata.name

Type: Label

Example: kubernetes.io/metadata.name: "mynamespace"

Used on: Namespaces

The Kubernetes API server (part of the [control plane](#)) sets this label on all namespaces. The label value is set to the name of the namespace. You can't change this label's value.

This is useful if you want to target a specific namespace with a label [selector](#).

kubernetes.io/limit-ranger

Type: Annotation

Example: kubernetes.io/limit-ranger: "LimitRanger plugin set: cpu, memory request for container nginx; cpu, memory limit for container nginx"

Used on: Pod

Kubernetes by default doesn't provide any resource limit, that means unless you explicitly define limits, your container can consume unlimited CPU and memory. You can define a default request or default limit for pods. You do this by creating a LimitRange in the relevant namespace. Pods deployed after you define a LimitRange will have these limits applied to them. The annotation kubernetes.io/limit-ranger records that resource defaults were specified for the Pod, and they were applied successfully. For more details, read about [LimitRanges](#).

addonmanager.kubernetes.io/mode

Type: Label

Example: addonmanager.kubernetes.io/mode: "Reconcile"

Used on: All objects

To specify how an add-on should be managed, you can use the addonmanager.kubernetes.io/mode label. This label can have one of three values: Reconcile, EnsureExists, or Ignore.

- **Reconcile:** Addon resources will be periodically reconciled with the expected state. If there are any differences, the add-on manager will recreate, reconfigure or delete the resources as needed. This is the default mode if no label is specified.
- **EnsureExists:** Addon resources will be checked for existence only but will not be modified after creation. The add-on manager will create or re-create the resources when there is no instance of the resource with that name.
- **Ignore:** Addon resources will be ignored. This mode is useful for add-ons that are not compatible with the add-on manager or that are managed by another controller.

For more details, see [Addon-manager](#).

beta.kubernetes.io/arch (deprecated)

Type: Label

This label has been deprecated. Please use [kubernetes.io/arch](#) instead.

beta.kubernetes.io/os (deprecated)

Type: Label

This label has been deprecated. Please use [kubernetes.io/os](#) instead.

kube-aggregator.kubernetes.io/automanged

Type: Label

Example: kube-aggregator.kubernetes.io/automanged: "onstart"

Used on: APIService

The kube-apiserver sets this label on any APIService object that the API server has created automatically. The label marks how the control plane should manage that APIService. You should not add, modify, or remove this label by yourself.

Note: Automanged APIService objects are deleted by kube-apiserver when it has no built-in or custom resource API corresponding to the API group/version of the APIService.

There are two possible values:

- **onstart:** The APIService should be reconciled when an API server starts up, but not otherwise.
- **true:** The API server should reconcile this APIService continuously.

service.alpha.kubernetes.io/tolerate-unready-endpoints (deprecated)

Type: Annotation

Used on: StatefulSet

This annotation on a Service denotes if the Endpoints controller should go ahead and create Endpoints for unready Pods. Endpoints of these Services retain their DNS records and continue receiving traffic for the Service from the moment the kubelet starts all containers in the pod and marks it *Running*, til the kubelet stops all containers and deletes the pod from the API server.

kubernetes.io/hostname

Type: Label

Example: kubernetes.io/hostname: "ip-172-20-114-199.ec2.internal"

Used on: Node

The Kubelet populates this label with the hostname of the node. Note that the hostname can be changed from the "actual" hostname by passing the --hostname-override flag to the kubelet.

This label is also used as part of the topology hierarchy. See [topology.kubernetes.io/zone](#) for more information.

kubernetes.io/change-cause

Type: Annotation

Example: kubernetes.io/change-cause: "kubectl edit --record deployment foo"

Used on: All Objects

This annotation is a best guess at why something was changed.

It is populated when adding --record to a kubectl command that may change an object.

kubernetes.io/description

Type: Annotation

Example: kubernetes.io/description: "Description of K8s object."

Used on: All Objects

This annotation is used for describing specific behaviour of given object.

kubernetes.io/enforce-mountable-secrets

Type: Annotation

Example: kubernetes.io/enforce-mountable-secrets: "true"

Used on: ServiceAccount

The value for this annotation must be **true** to take effect. When you set this annotation to "true", Kubernetes enforces the following rules for Pods running as this ServiceAccount:

1. Secrets mounted as volumes must be listed in the ServiceAccount's secrets field.
2. Secrets referenced in envFrom for containers (including sidecar containers and init containers) must also be listed in the ServiceAccount's secrets field. If any container in a Pod references a Secret not listed in the ServiceAccount's secrets field (and even if the reference is marked as optional), then the Pod will fail to start, and an error indicating the non-compliant secret reference will be generated.
3. Secrets referenced in a Pod's imagePullSecrets must be present in the ServiceAccount's imagePullSecrets field, the Pod will fail to start, and an error indicating the non-compliant image pull secret reference will be generated.

When you create or update a Pod, these rules are checked. If a Pod doesn't follow them, it won't start and you'll see an error message. If a Pod is already running and you change the kubernetes.io/enforce-mountable-secrets annotation to true, or you edit the associated ServiceAccount to remove the reference to a Secret that the Pod is already using, the Pod continues to run.

node.kubernetes.io/exclude-from-external-load-balancers

Type: Label

Example: node.kubernetes.io/exclude-from-external-load-balancers

Used on: Node

Kubernetes automatically enables the ServiceNodeExclusion feature gate on the clusters it creates. With this feature gate enabled on a cluster, you can add labels to particular worker nodes to exclude them from the list of backend servers. The following command can be used to exclude a worker node from the list of backend servers in a backend set:

```
kubectl label nodes <node-name> node.kubernetes.io/exclude-from-external-load-balancers=true
```

controller.kubernetes.io/pod-deletion-cost

Type: Annotation

Example: controller.kubernetes.io/pod-deletion-cost: "10"

Used on: Pod

This annotation is used to set [Pod Deletion Cost](#) which allows users to influence ReplicaSet downscaling order. The annotation value parses into an int32 type.

cluster-autoscaler.kubernetes.io/enable-ds-eviction

Type: Annotation

Example: cluster-autoscaler.kubernetes.io/enable-ds-eviction: "true"

Used on: Pod

This annotation controls whether a DaemonSet pod should be evicted by a ClusterAutoscaler. This annotation needs to be specified on DaemonSet pods in a DaemonSet manifest. When this annotation is set to "true", the ClusterAutoscaler is allowed to evict a DaemonSet Pod, even if other rules would normally prevent that. To disallow the ClusterAutoscaler from evicting DaemonSet pods, you can set this annotation to "false" for important DaemonSet pods. If this annotation is not set, then the ClusterAutoscaler follows its overall behavior (i.e evict the DaemonSets based on its configuration).

Note: This annotation only impacts DaemonSet Pods.

kubernetes.io/ingress-bandwidth

Type: Annotation

Example: kubernetes.io/ingress-bandwidth: 10M

Used on: Pod

You can apply quality-of-service traffic shaping to a pod and effectively limit its available bandwidth. Ingress traffic to a Pod is handled by shaping queued packets to effectively handle data. To limit the bandwidth on a Pod, write an object definition JSON file and specify the data traffic speed using kubernetes.io/ingress-bandwidth annotation. The unit used for specifying ingress rate is bits per second, as a [Quantity](#). For example, 10M means 10 megabits per second.

Note: Ingress traffic shaping annotation is an experimental feature. If you want to enable traffic shaping support, you must add the bandwidth plugin to your CNI configuration file (default /etc/cni/net.d) and ensure that the binary is included in your CNI bin dir (default /opt/cni/bin).

kubernetes.io/egress-bandwidth

Type: Annotation

Example: kubernetes.io/egress-bandwidth: 10M

Used on: Pod

Egress traffic from a Pod is handled by policing, which simply drops packets in excess of the configured rate. The limits you place on a Pod do not affect the bandwidth of other Pods. To limit the bandwidth on a Pod, write an object definition JSON file and specify the data traffic speed using kubernetes.io/egress-bandwidth annotation. The unit used for specifying egress rate is bits per second, as a [Quantity](#). For example, 10M means 10 megabits per second.

Note: Egress traffic shaping annotation is an experimental feature. If you want to enable traffic shaping support, you must add the bandwidth plugin to your CNI configuration file (default /etc/cni/net.d) and ensure that the binary is included in your CNI bin dir (default /opt/cni/bin).

beta.kubernetes.io/instance-type (deprecated)

Type: Label

Note: Starting in v1.17, this label is deprecated in favor of [node.kubernetes.io/instance-type](#).

node.kubernetes.io/instance-type

Type: Label

Example: node.kubernetes.io/instance-type: "m3.medium"

Used on: Node

The Kubelet populates this with the instance type as defined by the cloud provider. This will be set only if you are using a cloud provider. This setting is handy if you want to target certain workloads to certain instance types, but typically you want to rely on the Kubernetes scheduler to perform resource-based scheduling. You should aim to schedule based on properties rather than on instance types (for example: require a GPU, instead of requiring a g2.2xlarge).

failure-domain.beta.kubernetes.io/region (deprecated)

Type: Label

Note: Starting in v1.17, this label is deprecated in favor of [topology.kubernetes.io/region](#).

failure-domain.beta.kubernetes.io/zone (deprecated)

Type: Label

Note: Starting in v1.17, this label is deprecated in favor of [topology.kubernetes.io/zone](#).

pv.kubernetes.io/bind-completed

Type: Annotation

Example: pv.kubernetes.io/bind-completed: "yes"

Used on: PersistentVolumeClaim

When this annotation is set on a PersistentVolumeClaim (PVC), that indicates that the lifecycle of the PVC has passed through initial binding setup. When present, that information changes how the control plane interprets the state of PVC objects. The value of this annotation does not matter to Kubernetes.

pv.kubernetes.io/bound-by-controller

Type: Annotation

Example: pv.kubernetes.io/bound-by-controller: "yes"

Used on: PersistentVolume, PersistentVolumeClaim

If this annotation is set on a PersistentVolume or PersistentVolumeClaim, it indicates that a storage binding (PersistentVolume → PersistentVolumeClaim, or PersistentVolumeClaim → PersistentVolume) was installed by the [controller](#). If the annotation isn't set, and there is a storage binding in place, the absence of that annotation means that the binding was done manually. The value of this annotation does not matter.

pv.kubernetes.io/provisioned-by

Type: Annotation

Example: pv.kubernetes.io/provisioned-by: "kubernetes.io/rbd"

Used on: PersistentVolume

This annotation is added to a PersistentVolume(PV) that has been dynamically provisioned by Kubernetes. Its value is the name of volume plugin that created the volume. It serves both users (to show where a PV comes from) and Kubernetes (to recognize dynamically provisioned PVs in its decisions).

pv.kubernetes.io/migrated-to

Type: Annotation

Example: pv.kubernetes.io/migrated-to: pd.csi.storage.gke.io

Used on: PersistentVolume, PersistentVolumeClaim

It is added to a PersistentVolume(PV) and PersistentVolumeClaim(PVC) that is supposed to be dynamically provisioned/deleted by its corresponding CSI driver through the CSIMigration feature gate. When this annotation is set, the Kubernetes components will "stand-down" and the external-provisioner will act on the objects.

statefulset.kubernetes.io/pod-name

Type: Label

Example: statefulset.kubernetes.io/pod-name: "mystatefulset-7"

Used on: Pod

When a StatefulSet controller creates a Pod for the StatefulSet, the control plane sets this label on that Pod. The value of the label is the name of the Pod being created.

See [Pod Name Label](#) in the StatefulSet topic for more details.

scheduler.alpha.kubernetes.io/node-selector

Type: Annotation

Example: scheduler.alpha.kubernetes.io/node-selector: "name-of-node-selector"

Used on: Namespace

The [PodNodeSelector](#) uses this annotation key to assign node selectors to pods in namespaces.

topology.kubernetes.io/region

Type: Label

Example: topology.kubernetes.io/region: "us-east-1"

Used on: Node, PersistentVolume

See [topology.kubernetes.io/zone](#).

topology.kubernetes.io/zone

Type: Label

Example: topology.kubernetes.io/zone: "us-east-1c"

Used on: Node, PersistentVolume

On Node: The kubelet or the external cloud-controller-manager populates this with the information from the cloud provider. This will be set only if you are using a cloud provider. However, you can consider setting this on nodes if it makes sense in your topology.

On PersistentVolume: topology-aware volume provisioners will automatically set node affinity constraints on a PersistentVolume.

A zone represents a logical failure domain. It is common for Kubernetes clusters to span multiple zones for increased availability. While the exact definition of a zone is left to infrastructure implementations, common properties of a zone include very low network latency within a zone, no-cost network traffic within a zone, and failure independence from other zones. For example, nodes within a zone might share a network switch, but nodes in different zones should not.

A region represents a larger domain, made up of one or more zones. It is uncommon for Kubernetes clusters to span multiple regions. While the exact definition of a zone or region is left to infrastructure implementations, common properties of a region include higher network latency between them than within them, non-zero cost for network traffic between them, and failure independence from other zones or regions. For example, nodes within a region might share power infrastructure (e.g. a UPS or generator), but nodes in different regions typically would not.

Kubernetes makes a few assumptions about the structure of zones and regions:

1. regions and zones are hierarchical: zones are strict subsets of regions and no zone can be in 2 regions
2. zone names are unique across regions; for example region "africa-east-1" might be comprised of zones "africa-east-1a" and "africa-east-1b"

It should be safe to assume that topology labels do not change. Even though labels are strictly mutable, consumers of them can assume that a given node is not going to be moved between zones without being destroyed and recreated.

Kubernetes can use this information in various ways. For example, the scheduler automatically tries to spread the Pods in a ReplicaSet across nodes in a single-zone cluster (to reduce the impact of node failures, see [kubernetes.io/hostname](#)). With multiple-zone clusters, this spreading behavior also applies to zones (to reduce the impact of zone failures). This is achieved via *SelectorSpreadPriority*.

SelectorSpreadPriority is a best effort placement. If the zones in your cluster are heterogeneous (for example: different numbers of nodes, different types of nodes, or different pod resource requirements), this placement might prevent equal spreading of your Pods across zones. If desired, you can use homogenous zones (same number and types of nodes) to reduce the probability of unequal spreading.

The scheduler (through the *VolumeZonePredicate* predicate) also will ensure that Pods, that claim a given volume, are only placed into the same zone as that volume. Volumes cannot be attached across zones.

If `PersistentVolumeLabel` does not support automatic labeling of your `PersistentVolumes`, you should consider adding the labels manually (or adding support for `PersistentVolumeLabel`). With `PersistentVolumeLabel`, the scheduler prevents Pods from mounting volumes in a different zone. If your infrastructure doesn't have this constraint, you don't need to add the zone labels to the volumes at all.

volume.beta.kubernetes.io/storage-provisioner (deprecated)

Type: Annotation

Example: `volume.beta.kubernetes.io/storage-provisioner: "k8s.io/minikube-hostpath"`

Used on: `PersistentVolumeClaim`

This annotation has been deprecated since v1.23. See [volume.kubernetes.io/storage-provisioner](#).

volume.beta.kubernetes.io/storage-class (deprecated)

Type: Annotation

Example: `volume.beta.kubernetes.io/storage-class: "example-class"`

Used on: `PersistentVolume`, `PersistentVolumeClaim`

This annotation can be used for `PersistentVolume(PV)` or `PersistentVolumeClaim(PVC)` to specify the name of [StorageClass](#). When both the `storageClassName` attribute and the `volume.beta.kubernetes.io/storage-class` annotation are specified, the annotation `volume.beta.kubernetes.io/storage-class` takes precedence over the `storageClassName` attribute.

This annotation has been deprecated. Instead, set the [storageClassName field](#) for the `PersistentVolumeClaim` or `PersistentVolume`.

volume.beta.kubernetes.io/mount-options (deprecated)

Type: Annotation

Example : `volume.beta.kubernetes.io/mount-options: "ro,soft"`

Used on: `PersistentVolume`

A Kubernetes administrator can specify additional [mount options](#) for when a `PersistentVolume` is mounted on a node.

volume.kubernetes.io/storage-provisioner

Type: Annotation

Used on: PersistentVolumeClaim

This annotation is added to a PVC that is supposed to be dynamically provisioned. Its value is the name of a volume plugin that is supposed to provision a volume for this PVC.

volume.kubernetes.io/selected-node

Type: Annotation

Used on: PersistentVolumeClaim

This annotation is added to a PVC that is triggered by a scheduler to be dynamically provisioned. Its value is the name of the selected node.

volumes.kubernetes.io/controller-managed-attach-detach

Type: Annotation

Used on: Node

If a node has the annotation volumes.kubernetes.io/controller-managed-attach-detach, its storage attach and detach operations are being managed by the *volume attach/detach controller*.

The value of the annotation isn't important.

node.kubernetes.io/windows-build

Type: Label

Example: node.kubernetes.io/windows-build: "10.0.17763"

Used on: Node

When the kubelet is running on Microsoft Windows, it automatically labels its Node to record the version of Windows Server in use.

The label's value is in the format "MajorVersion.MinorVersion.BuildNumber".

service.kubernetes.io/headless

Type: Label

Example: service.kubernetes.io/headless: ""

Used on: Service

The control plane adds this label to an Endpoints object when the owning Service is headless.

service.kubernetes.io/topology-aware-hints (deprecated)

Example: `service.kubernetes.io/topology-aware-hints: "Auto"`

Used on: Service

This annotation was used for enabling *topology aware hints* on Services. Topology aware hints have since been renamed: the concept is now called [topology aware routing](#). Setting the annotation to Auto, on a Service, configured the Kubernetes control plane to add topology hints on EndpointSlices associated with that Service. You can also explicitly set the annotation to Disabled.

If you are running a version of Kubernetes older than 1.28, check the documentation for that Kubernetes version to see how topology aware routing works in that release.

There are no other valid values for this annotation. If you don't want topology aware hints for a Service, don't add this annotation.

service.kubernetes.io/topology-mode

Type: Annotation

Example: `service.kubernetes.io/topology-mode: Auto`

Used on: Service

This annotation provides a way to define how Services handle network topology; for example, you can configure a Service so that Kubernetes prefers keeping traffic between a client and server within a single topology zone. In some cases this can help reduce costs or improve network performance.

See [Topology Aware Routing](#) for more details.

kubernetes.io/service-name

Type: Label

Example: `kubernetes.io/service-name: "my-website"`

Used on: EndpointSlice

Kubernetes associates [EndpointSlices](#) with [Services](#) using this label.

This label records the [name](#) of the Service that the EndpointSlice is backing. All EndpointSlices should have this label set to the name of their associated Service.

kubernetes.io/service-account.name

Type: Annotation

Example: `kubernetes.io/service-account.name: "sa-name"`

Used on: Secret

This annotation records the [name](#) of the ServiceAccount that the token (stored in the Secret of type kubernetes.io/service-account-token) represents.

kubernetes.io/service-account.uid

Type: Annotation

Example: kubernetes.io/service-account.uid: da68f9c6-9d26-11e7-b84e-002dc52800da

Used on: Secret

This annotation records the [unique ID](#) of the ServiceAccount that the token (stored in the Secret of type kubernetes.io/service-account-token) represents.

kubernetes.io/legacy-token-last-used

Type: Label

Example: kubernetes.io/legacy-token-last-used: 2022-10-24

Used on: Secret

The control plane only adds this label to Secrets that have the type kubernetes.io/service-account-token. The value of this label records the date (ISO 8601 format, UTC time zone) when the control plane last saw a request where the client authenticated using the service account token.

If a legacy token was last used before the cluster gained the feature (added in Kubernetes v1.26), then the label isn't set.

endpointslice.kubernetes.io/managed-by

Type: Label

Example: endpointslice.kubernetes.io/managed-by: "controller"

Used on: EndpointSlices

The label is used to indicate the controller or entity that manages the EndpointSlice. This label aims to enable different EndpointSlice objects to be managed by different controllers or entities within the same cluster.

endpointslice.kubernetes.io/skip-mirror

Type: Label

Example: endpointslice.kubernetes.io/skip-mirror: "true"

Used on: Endpoints

The label can be set to "true" on an Endpoints resource to indicate that the EndpointSliceMirroring controller should not mirror this resource with EndpointSlices.

service.kubernetes.io/service-proxy-name

Type: Label

Example: service.kubernetes.io/service-proxy-name: "foo-bar"

Used on: Service

The kube-proxy has this label for custom proxy, which delegates service control to custom proxy.

experimental.windows.kubernetes.io/isolation-type (deprecated)

Type: Annotation

Example: experimental.windows.kubernetes.io/isolation-type: "hyperv"

Used on: Pod

The annotation is used to run Windows containers with Hyper-V isolation.

Note: Starting from v1.20, this annotation is deprecated. Experimental Hyper-V support was removed in 1.21.

ingressclass.kubernetes.io/is-default-class

Type: Annotation

Example: ingressclass.kubernetes.io/is-default-class: "true"

Used on: IngressClass

When a IngressClass resource has this annotation set to "true", new Ingress resource without a class specified will be assigned this default class.

kubernetes.io/ingress.class (deprecated)

Type: Annotation

Used on: Ingress

Note: Starting in v1.18, this annotation is deprecated in favor of spec.ingressClassName.

storageclass.kubernetes.io/is-default-class

Type: Annotation

Example: storageclass.kubernetes.io/is-default-class: "true"

Used on: StorageClass

When a single StorageClass resource has this annotation set to "true", new PersistentVolumeClaim resource without a class specified will be assigned this default class.

alpha.kubernetes.io/provided-node-ip (alpha)

Type: Annotation

Example: alpha.kubernetes.io/provided-node-ip: "10.0.0.1"

Used on: Node

The kubelet can set this annotation on a Node to denote its configured IPv4 and/or IPv6 address.

When kubelet is started with the --cloud-provider flag set to any value (includes both external and legacy in-tree cloud providers), it sets this annotation on the Node to denote an IP address set from the command line flag (--node-ip). This IP is verified with the cloud provider as valid by the cloud-controller-manager.

batch.kubernetes.io/job-completion-index

Type: Annotation, Label

Example: batch.kubernetes.io/job-completion-index: "3"

Used on: Pod

The Job controller in the kube-controller-manager sets this as a label and annotation for Pods created with Indexed [completion mode](#).

Note the [PodIndexLabel](#) feature gate must be enabled for this to be added as a pod **label**, otherwise it will just be an annotation.

batch.kubernetes.io/cronjob-scheduled-timestamp

Type: Annotation

Example: batch.kubernetes.io/cronjob-scheduled-timestamp: "2016-05-19T03:00:00-07:00"

Used on: Jobs and Pods controlled by CronJobs

This annotation is used to record the original (expected) creation timestamp for a Job, when that Job is part of a CronJob. The control plane sets the value to that timestamp in RFC3339 format. If the Job belongs to a CronJob with a timezone specified, then the timestamp is in that timezone. Otherwise, the timestamp is in controller-manager's local time.

kubectl.kubernetes.io/default-container

Type: Annotation

Example: kubectl.kubernetes.io/default-container: "front-end-app"

The value of the annotation is the container name that is default for this Pod. For example, kubectl logs or kubectl exec without -c or --container flag will use this default container.

kubectl.kubernetes.io/default-logs-container (deprecated)

Type: Annotation

Example: kubectl.kubernetes.io/default-logs-container: "front-end-app"

The value of the annotation is the container name that is the default logging container for this Pod. For example, kubectl logs without -c or --container flag will use this default container.

Note: This annotation is deprecated. You should use the [kubectl.kubernetes.io/default-container](#) annotation instead. Kubernetes versions 1.25 and newer ignore this annotation.

kubectl.kubernetes.io/last-applied-configuration

Type: Annotation

Example: *see following snippet*

```
kubectl.kubernetes.io/last-applied-configuration: >
  {"apiVersion": "apps/v1", "kind": "Deployment", "metadata": {"annotations": {}}, "name": "example", "namespace": "default", "spec": {"selector": {"matchLabels": {"app.kubernetes.io/name": "foo"}}, "template": {"metadata": {"labels": {"app.kubernetes.io/name": "foo"}}, "spec": {"containers": [{"image": "container-registry.example/foo-bar:1.42", "name": "foo-bar", "ports": [{"containerPort": 42}]}]}}}}}
```

Used on: all objects

The kubectl command line tool uses this annotation as a legacy mechanism to track changes. That mechanism has been superseded by [Server-side apply](#).

endpoints.kubernetes.io/over-capacity

Type: Annotation

Example: endpoints.kubernetes.io/over-capacity:truncated

Used on: Endpoints

The [control plane](#) adds this annotation to an [Endpoints](#) object if the associated [Service](#) has more than 1000 backing endpoints. The annotation indicates that the Endpoints object is over capacity and the number of endpoints has been truncated to 1000.

If the number of backend endpoints falls below 1000, the control plane removes this annotation.

control-plane.alpha.kubernetes.io/leader (deprecated)

Type: Annotation

```
Example: control-plane.alpha.kubernetes.io/
leader={"holderIdentity":"controller-0","leaseDurationSeconds":15,"acquireTime":"2023-01-19T13:12:57Z","renewTime":"2023-01-19T13:13:54Z","leaderTransitions":1}
```

Used on: Endpoints

The [control plane](#) previously set annotation on an [Endpoints](#) object. This annotation provided the following detail:

- Who is the current leader.
- The time when the current leadership was acquired.
- The duration of the lease (of the leadership) in seconds.
- The time the current lease (the current leadership) should be renewed.
- The number of leadership transitions that happened in the past.

Kubernetes now uses [Leases](#) to manage leader assignment for the Kubernetes control plane.

batch.kubernetes.io/job-tracking (deprecated)

Type: Annotation

Example: batch.kubernetes.io/job-tracking: ""

Used on: Jobs

The presence of this annotation on a Job used to indicate that the control plane is [tracking the Job status using finalizers](#). Adding or removing this annotation no longer has an effect (Kubernetes v1.27 and later) All Jobs are tracked with finalizers.

job-name (deprecated)

Type: Label

Example: job-name: "pi"

Used on: Jobs and Pods controlled by Jobs

Note: Starting from Kubernetes 1.27, this label is deprecated. Kubernetes 1.27 and newer ignore this label and use the prefixed job-name label.

controller-uid (deprecated)

Type: Label

Example: controller-uid: "\$UID"

Used on: Jobs and Pods controlled by Jobs

Note: Starting from Kubernetes 1.27, this label is deprecated. Kubernetes 1.27 and newer ignore this label and use the prefixed controller-uid label.

batch.kubernetes.io/job-name

Type: Label

Example: batch.kubernetes.io/job-name: "pi"

Used on: Jobs and Pods controlled by Jobs

This label is used as a user-friendly way to get Pods corresponding to a Job. The job-name comes from the name of the Job and allows for an easy way to get Pods corresponding to the Job.

batch.kubernetes.io/controller-uid

Type: Label

Example: batch.kubernetes.io/controller-uid: "\$UID"

Used on: Jobs and Pods controlled by Jobs

This label is used as a programmatic way to get all Pods corresponding to a Job. The controller-uid is a unique identifier that gets set in the selector field so the Job controller can get all the corresponding Pods.

scheduler.alpha.kubernetes.io/defaultTolerations

Type: Annotation

Example: scheduler.alpha.kubernetes.io/defaultTolerations: '[{"operator": "Equal", "value": "value1", "effect": "NoSchedule", "key": "dedicated-node"}]'

Used on: Namespace

This annotation requires the [PodTolerationRestriction](#) admission controller to be enabled. This annotation key allows assigning tolerations to a namespace and any new pods created in this namespace would get these tolerations added.

scheduler.alpha.kubernetes.io/tolerationsWhitelist

Type: Annotation

Example: scheduler.alpha.kubernetes.io/tolerationsWhitelist: '[{"operator": "Exists", "effect": "NoSchedule", "key": "dedicated-node"}]'

Used on: Namespace

This annotation is only useful when the (Alpha) [PodTolerationRestriction](#) admission controller is enabled. The annotation value is a JSON document that defines a list of allowed tolerations for the namespace it annotates. When you create a Pod or modify its tolerations, the API server checks the tolerations to see if they are mentioned in the allow list. The pod is admitted only if the check succeeds.

scheduler.alpha.kubernetes.io/preferAvoidPods (deprecated)

Type: Annotation

Used on: Node

This annotation requires the [NodePreferAvoidPods scheduling plugin](#) to be enabled. The plugin is deprecated since Kubernetes 1.22. Use [Taints and Tolerations](#) instead.

node.kubernetes.io/not-ready

Type: Taint

Example: node.kubernetes.io/not-ready: "NoExecute"

Used on: Node

The Node controller detects whether a Node is ready by monitoring its health and adds or removes this taint accordingly.

node.kubernetes.io/unreachable

Type: Taint

Example: node.kubernetes.io/unreachable: "NoExecute"

Used on: Node

The Node controller adds the taint to a Node corresponding to the [NodeCondition](#) Ready being Unknown.

node.kubernetes.io/unschedulable

Type: Taint

Example: node.kubernetes.io/unschedulable: "NoSchedule"

Used on: Node

The taint will be added to a node when initializing the node to avoid race condition.

node.kubernetes.io/memory-pressure

Type: Taint

Example: node.kubernetes.io/memory-pressure: "NoSchedule"

Used on: Node

The kubelet detects memory pressure based on memory.available and allocatableMemory.available observed on a Node. The observed values are then compared to the corresponding thresholds that can be set on the kubelet to determine if the Node condition and taint should be added/removed.

node.kubernetes.io/disk-pressure

Type: Taint

Example: node.kubernetes.io/disk-pressure : "NoSchedule"

Used on: Node

The kubelet detects disk pressure based on imagefs.available, imagefs.inodesFree, nodefs.available and nodefs.inodesFree(Linux only) observed on a Node. The observed values are then compared to the corresponding thresholds that can be set on the kubelet to determine if the Node condition and taint should be added/removed.

node.kubernetes.io/network-unavailable

Type: Taint

Example: node.kubernetes.io/network-unavailable: "NoSchedule"

Used on: Node

This is initially set by the kubelet when the cloud provider used indicates a requirement for additional network configuration. Only when the route on the cloud is configured properly will the taint be removed by the cloud provider.

node.kubernetes.io/pid-pressure

Type: Taint

Example: node.kubernetes.io/pid-pressure: "NoSchedule"

Used on: Node

The kubelet checks D-value of the size of /proc/sys/kernel/pid_max and the PIDs consumed by Kubernetes on a node to get the number of available PIDs that referred to as the pid.available metric. The metric is then compared to the corresponding threshold that can be set on the kubelet to determine if the node condition and taint should be added/removed.

node.kubernetes.io/out-of-service

Type: Taint

Example: node.kubernetes.io/out-of-service:NoExecute

Used on: Node

A user can manually add the taint to a Node marking it out-of-service. If the NodeOutOfServiceVolumeDetach [feature gate](#) is enabled on kube-controller-manager, and a Node is marked out-of-service with this taint, the Pods on the node will be forcefully deleted if there are no matching tolerations on it and volume detach operations for the Pods terminating on the node will happen immediately. This allows the Pods on the out-of-service node to recover quickly on a different node.

Caution: Refer to [Non-graceful node shutdown](#) for further details about when and how to use this taint.

node.cloudprovider.kubernetes.io/uninitialized

Type: Taint

Example: node.cloudprovider.kubernetes.io/uninitialized: "NoSchedule"

Used on: Node

Sets this taint on a Node to mark it as unusable, when kubelet is started with the "external" cloud provider, until a controller from the cloud-controller-manager initializes this Node, and then removes the taint.

node.cloudprovider.kubernetes.io/shutdown

Type: Taint

Example: node.cloudprovider.kubernetes.io/shutdown: "NoSchedule"

Used on: Node

If a Node is in a cloud provider specified shutdown state, the Node gets tainted accordingly with node.cloudprovider.kubernetes.io/shutdown and the taint effect of NoSchedule.

feature.node.kubernetes.io/*

Type: Label

Example: feature.node.kubernetes.io/network-sriov.capable: "true"

Used on: Node

These labels are used by the Node Feature Discovery (NFD) component to advertise features on a node. All built-in labels use the feature.node.kubernetes.io label namespace and have the format feature.node.kubernetes.io/<feature-name>: "true". NFD has many extension points for creating vendor and application-specific labels. For details, see the [customization guide](#).

nfd.node.kubernetes.io/master.version

Type: Annotation

Example: nfd.node.kubernetes.io/master.version: "v0.6.0"

Used on: Node

For node(s) where the Node Feature Discovery (NFD) [master](#) is scheduled, this annotation records the version of the NFD master. It is used for informative use only.

nfd.node.kubernetes.io/worker.version

Type: Annotation

Example: nfd.node.kubernetes.io/worker.version: "v0.4.0"

Used on: Nodes

This annotation records the version for a Node Feature Discovery's [worker](#) if there is one running on a node. It's used for informative use only.

nfd.node.kubernetes.io/feature-labels

Type: Annotation

Example: nfd.node.kubernetes.io/feature-labels: "cpu-cpuid.ADX,cpu-cpuid.AESNI,cpu-hardware_multithreading,kernel-version.full"

Used on: Nodes

This annotation records a comma-separated list of node feature labels managed by [Node Feature Discovery](#) (NFD). NFD uses this for an internal mechanism. You should not edit this annotation yourself.

nfd.node.kubernetes.io/extended-resources

Type: Annotation

Example: nfd.node.kubernetes.io/extended-resources: "accelerator.acme.example/q500,example.com/coprocessor-fx5"

Used on: Nodes

This annotation records a comma-separated list of [extended resources](#) managed by [Node Feature Discovery](#) (NFD). NFD uses this for an internal mechanism. You should not edit this annotation yourself.

nfd.node.kubernetes.io/node-name

Type: Label

Example: nfd.node.kubernetes.io/node-name: node-1

Used on: Nodes

It specifies which node the NodeFeature object is targeting. Creators of NodeFeature objects must set this label and consumers of the objects are supposed to use the label for filtering features designated for a certain node.

Note: These Node Feature Discovery (NFD) labels or annotations only apply to the nodes where NFD is running. To learn more about NFD and its components go to its official [documentation](#).

service.beta.kubernetes.io/aws-load-balancer-access-log-emit-interval (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-access-log-emit-interval: "5"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures the load balancer for a Service based on this annotation. The value determines how often the load balancer writes log entries. For example, if you set the value to 5, the log writes occur 5 seconds apart.

service.beta.kubernetes.io/aws-load-balancer-access-log-enabled (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-access-log-enabled: "false"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures the load balancer for a Service based on this annotation. Access logging is enabled if you set the annotation to "true".

service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-access-log-enabled: example

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures the load balancer for a Service based on this annotation. The load balancer writes logs to an S3 bucket with the name you specify.

service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-prefix (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-access-log-enabled: "/example"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures the load balancer for a Service based on this annotation. The load balancer writes log objects with the prefix that you specify.

service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-additional-resource-tags: "Environment=demo,Project=example"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures tags (an AWS concept) for a load balancer based on the comma-separated key/value pairs in the value of this annotation.

service.beta.kubernetes.io/aws-load-balancer-alpn-policy (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-alpn-policy: HTTP2Optional

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-attributes (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-attributes:
"deletion_protection.enabled=true"

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-backend-protocol (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-backend-protocol: tcp

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures the load balancer listener based on the value of this annotation.

service.beta.kubernetes.io/aws-load-balancer-connection-draining-enabled (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-connection-draining-enabled: "false"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures the load balancer based on this annotation. The load balancer's connection draining setting depends on the value you set.

service.beta.kubernetes.io/aws-load-balancer-connection-draining-timeout (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-connection-draining-timeout: "60"

Used on: Service

If you configure [connection draining](#) for a Service of type: LoadBalancer, and you use the AWS cloud, the integration configures the draining period based on this annotation. The value you set determines the draining timeout in seconds.

service.beta.kubernetes.io/aws-load-balancer-ip-address-type (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-ip-address-type: ipv4

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-connection-idle-timeout (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-connection-idle-timeout: "60"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The load balancer has a configured idle timeout period (in seconds) that applies to its connections. If no data has been sent or received by the time that the idle timeout period elapses, the load balancer closes the connection.

service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-cross-zone-load-balancing-enabled: "true"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. If you set this annotation to "true", each load balancer node distributes requests evenly across the registered targets in all enabled [availability zones](#). If you disable cross-zone load balancing, each load balancer node distributes requests evenly across the registered targets in its availability zone only.

service.beta.kubernetes.io/aws-load-balancer-eip-allocations (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-eip-allocations: "eipalloc-01bcdef23bcdef456,eipalloc-def1234abc4567890"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The value is a comma-separated list of elastic IP address allocation IDs.

This annotation is only relevant for Services of type: LoadBalancer, where the load balancer is an AWS Network Load Balancer.

service.beta.kubernetes.io/aws-load-balancer-extra-security-groups (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-extra-security-groups: "sg-12abcd3456,sg-34dcba6543"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value is a comma-separated list of extra AWS VPC security groups to configure for the load balancer.

service.beta.kubernetes.io/aws-load-balancer-healthcheck-healthy-threshold (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-healthcheck-healthy-threshold: "3"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value specifies the number of successive successful health checks required for a backend to be considered healthy for traffic.

service.beta.kubernetes.io/aws-load-balancer-healthcheck-interval (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-healthcheck-interval: "30"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value specifies the interval, in seconds, between health check probes made by the load balancer.

service.beta.kubernetes.io/aws-load-balancer-healthcheck-path (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-healthcheck-path: /healthcheck

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value determines the path part of the URL that is used for HTTP health checks.

service.beta.kubernetes.io/aws-load-balancer-healthcheck-port (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-healthcheck-port: "24"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value determines which port the load balancer connects to when performing health checks.

service.beta.kubernetes.io/aws-load-balancer-healthcheck-protocol (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-healthcheck-protocol: TCP

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value determines how the load balancer checks the health of backend targets.

service.beta.kubernetes.io/aws-load-balancer-healthcheck-timeout (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-healthcheck-timeout: "3"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value specifies the number of seconds before a probe that hasn't yet succeeded is automatically treated as having failed.

service.beta.kubernetes.io/aws-load-balancer-healthcheck-unhealthy-threshold (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-healthcheck-unhealthy-threshold: "3"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. The annotation value specifies the number of successive unsuccessful health checks required for a backend to be considered unhealthy for traffic.

service.beta.kubernetes.io/aws-load-balancer-internal (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-internal: "true"

Used on: Service

The cloud controller manager integration with AWS elastic load balancing configures a load balancer based on this annotation. When you set this annotation to "true", the integration configures an internal load balancer.

If you use the [AWS load balancer controller](#), see [service.beta.kubernetes.io/aws-load-balancer-scheme](#).

service.beta.kubernetes.io/aws-load-balancer-manage-backend-security-group-rules (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-manage-backend-security-group-rules: "true"

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-name (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-name: my-elb

Used on: Service

If you set this annotation on a Service, and you also annotate that Service with service.beta.kubernetes.io/aws-load-balancer-type: "external", and you use the [AWS load](#)

[balancer controller](#) in your cluster, then the AWS load balancer controller sets the name of that load balancer to the value you set for *this* annotation.

See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-nlb-target-type (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-nlb-target-type: "true"

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-private-ipv4-addresses (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-private-ipv4-addresses: "198.51.100.0,198.51.100.64"

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-proxy-protocol (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-proxy-protocol: "*"

Used on: Service

The official Kubernetes integration with AWS elastic load balancing configures a load balancer based on this annotation. The only permitted value is "*", which indicates that the load balancer should wrap TCP connections to the backend Pod with the PROXY protocol.

service.beta.kubernetes.io/aws-load-balancer-scheme (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-scheme: internal

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-security-groups (deprecated)

Example: service.beta.kubernetes.io/aws-load-balancer-security-groups: "sg-53fae93f,sg-8725gr62r"

Used on: Service

The AWS load balancer controller uses this annotation to specify a comma separated list of security groups you want to attach to an AWS load balancer. Both name and ID of security are supported where name matches a Name tag, not the groupName attribute.

When this annotation is added to a Service, the load-balancer controller attaches the security groups referenced by the annotation to the load balancer. If you omit this annotation, the AWS load balancer controller automatically creates a new security group and attaches it to the load balancer.

Note: Kubernetes v1.27 and later do not directly set or read this annotation. However, the AWS load balancer controller (part of the Kubernetes project) does still use the service.beta.kubernetes.io/aws-load-balancer-security-groups annotation.

service.beta.kubernetes.io/load-balancer-source-ranges (deprecated)

Example: service.beta.kubernetes.io/load-balancer-source-ranges: "192.0.2.0/25"

Used on: Service

The [AWS load balancer controller](#) uses this annotation. You should set .spec.loadBalancerSourceRanges for the Service instead.

service.beta.kubernetes.io/aws-load-balancer-ssl-cert (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-ssl-cert: "arn:aws:acm:us-east-1:123456789012:certificate/12345678-1234-1234-1234-123456789012"

Used on: Service

The official integration with AWS elastic load balancing configures TLS for a Service of type: LoadBalancer based on this annotation. The value of the annotation is the AWS Resource Name (ARN) of the X.509 certificate that the load balancer listener should use.

(The TLS protocol is based on an older technology that abbreviates to SSL.)

service.beta.kubernetes.io/aws-load-balancer-ssl-negotiation-policy (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-ssl-negotiation-policy: ELBSecurityPolicy-TLS-1-2-2017-01

The official integration with AWS elastic load balancing configures TLS for a Service of type: LoadBalancer based on this annotation. The value of the annotation is the name of an AWS policy for negotiating TLS with a client peer.

service.beta.kubernetes.io/aws-load-balancer-ssl-ports (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-ssl-ports: "***"

The official integration with AWS elastic load balancing configures TLS for a Service of type: LoadBalancer based on this annotation. The value of the annotation is either "***", which means that all the load balancer's ports should use TLS, or it is a comma separated list of port numbers.

service.beta.kubernetes.io/aws-load-balancer-subnets (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-subnets: "private-a,private-b"

Kubernetes' official integration with AWS uses this annotation to configure a load balancer and determine in which AWS availability zones to deploy the managed load balancing service. The value is either a comma separated list of subnet names, or a comma separated list of subnet IDs.

service.beta.kubernetes.io/aws-load-balancer-target-group-attributes (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-target-group-attributes: "stickiness.enabled=true,stickiness.type=source_ip"

Used on: Service

The [AWS load balancer controller](#) uses this annotation. See [annotations](#) in the AWS load balancer controller documentation.

service.beta.kubernetes.io/aws-load-balancer-target-node-labels (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-target-node-labels: "kubernetes.io/os=Linux,topology.kubernetes.io/region=us-east-2"

Kubernetes' official integration with AWS uses this annotation to determine which nodes in your cluster should be considered as valid targets for the load balancer.

service.beta.kubernetes.io/aws-load-balancer-type (beta)

Example: service.beta.kubernetes.io/aws-load-balancer-type: external

Kubernetes' official integrations with AWS use this annotation to determine whether the AWS cloud provider integration should manage a Service of type: LoadBalancer.

There are two permitted values:

nlb

the cloud controller manager configures a Network Load Balancer

external

the cloud controller manager does not configure any load balancer

If you deploy a Service of type: LoadBalancer on AWS, and you don't set any service.beta.kubernetes.io/aws-load-balancer-type annotation, the AWS integration deploys a classic Elastic Load Balancer. This behavior, with no annotation present, is the default unless you specify otherwise.

When you set this annotation to external on a Service of type: LoadBalancer, and your cluster has a working deployment of the AWS Load Balancer controller, then the AWS Load Balancer controller attempts to deploy a load balancer based on the Service specification.

Caution: Do not modify or add the service.beta.kubernetes.io/aws-load-balancer-type annotation on an existing Service object. See the AWS documentation on this topic for more details.

pod-security.kubernetes.io/enforce

Type: Label

Example: pod-security.kubernetes.io/enforce: "baseline"

Used on: Namespace

Value **must** be one of privileged, baseline, or restricted which correspond to [Pod Security Standard](#) levels. Specifically, the enforce label *prohibits* the creation of any Pod in the labeled Namespace which does not meet the requirements outlined in the indicated level.

See [Enforcing Pod Security at the Namespace Level](#) for more information.

pod-security.kubernetes.io/enforce-version

Type: Label

Example: pod-security.kubernetes.io/enforce-version: "1.28"

Used on: Namespace

Value **must** be latest or a valid Kubernetes version in the format v<major>.<minor>. This determines the version of the [Pod Security Standard](#) policies to apply when validating a Pod.

See [Enforcing Pod Security at the Namespace Level](#) for more information.

pod-security.kubernetes.io/audit

Type: Label

Example: pod-security.kubernetes.io/audit: "baseline"

Used on: Namespace

Value **must** be one of privileged, baseline, or restricted which correspond to [Pod Security Standard](#) levels. Specifically, the audit label does not prevent the creation of a Pod in the labeled Namespace which does not meet the requirements outlined in the indicated level, but adds an annotation to the Pod.

See [Enforcing Pod Security at the Namespace Level](#) for more information.

pod-security.kubernetes.io/audit-version

Type: Label

Example: pod-security.kubernetes.io/audit-version: "1.28"

Used on: Namespace

Value **must** be latest or a valid Kubernetes version in the format v<major>.<minor>. This determines the version of the [Pod Security Standard](#) policies to apply when validating a Pod.

See [Enforcing Pod Security at the Namespace Level](#) for more information.

pod-security.kubernetes.io/warn

Type: Label

Example: pod-security.kubernetes.io/warn: "baseline"

Used on: Namespace

Value **must** be one of privileged, baseline, or restricted which correspond to [Pod Security Standard](#) levels. Specifically, the warn label does not prevent the creation of a Pod in the labeled Namespace which does not meet the requirements outlined in the indicated level, but returns a warning to the user after doing so. Note that warnings are also displayed when creating or updating objects that contain Pod templates, such as Deployments, Jobs, StatefulSets, etc.

See [Enforcing Pod Security at the Namespace Level](#) for more information.

pod-security.kubernetes.io/warn-version

Type: Label

Example: pod-security.kubernetes.io/warn-version: "1.28"

Used on: Namespace

Value **must** be latest or a valid Kubernetes version in the format v<major>.<minor>. This determines the version of the [Pod Security Standard](#) policies to apply when validating a submitted Pod. Note that warnings are also displayed when creating or updating objects that contain Pod templates, such as Deployments, Jobs, StatefulSets, etc.

See [Enforcing Pod Security at the Namespace Level](#) for more information.

rbac.authorization.kubernetes.io/autoupdate

Type: Annotation

Example: rbac.authorization.kubernetes.io/autoupdate: "false"

Used on: ClusterRole, ClusterRoleBinding, Role, RoleBinding

When this annotation is set to "true" on default RBAC objects created by the API server, they are automatically updated at server start to add missing permissions and subjects (extra permissions and subjects are left in place). To prevent autoupdating a particular role or rolebinding, set this annotation to "false". If you create your own RBAC objects and set this annotation to "false", kubectl auth reconcile (which allows reconciling arbitrary RBAC objects in a [manifest](#)) respects this annotation and does not automatically add missing permissions and subjects.

kubernetes.io/psp (deprecated)

Type: Annotation

Example: kubernetes.io/psp: restricted

Used on: Pod

This annotation was only relevant if you were using [PodSecurityPolicy](#) objects. Kubernetes v1.28 does not support the PodSecurityPolicy API.

When the PodSecurityPolicy admission controller admitted a Pod, the admission controller modified the Pod to have this annotation. The value of the annotation was the name of the PodSecurityPolicy that was used for validation.

seccomp.security.alpha.kubernetes.io/pod (non-functional)

Type: Annotation

Used on: Pod

Kubernetes before v1.25 allowed you to configure seccomp behavior using this annotation. See [Restrict a Container's Syscalls with seccomp](#) to learn the supported way to specify seccomp restrictions for a Pod.

container.seccomp.security.alpha.kubernetes.io/[NAME] (non-functional)

Type: Annotation

Used on: Pod

Kubernetes before v1.25 allowed you to configure seccomp behavior using this annotation. See [Restrict a Container's Syscalls with seccomp](#) to learn the supported way to specify seccomp restrictions for a Pod.

snapshot.storage.kubernetes.io/allow-volume-mode-change

Type: Annotation

Example: snapshot.storage.kubernetes.io/allow-volume-mode-change: "true"

Used on: VolumeSnapshotContent

Value can either be true or false. This determines whether a user can modify the mode of the source volume when a PersistentVolumeClaim is being created from a VolumeSnapshot.

Refer to [Converting the volume mode of a Snapshot](#) and the [Kubernetes CSI Developer Documentation](#) for more information.

scheduler.alpha.kubernetes.io/critical-pod (deprecated)

Type: Annotation

Example: scheduler.alpha.kubernetes.io/critical-pod: ""

Used on: Pod

This annotation lets Kubernetes control plane know about a Pod being a critical Pod so that the descheduler will not remove this Pod.

Note: Starting in v1.16, this annotation was removed in favor of [Pod Priority](#).

Annotations used for audit

- [authorization.k8s.io/decision](#)
- [authorization.k8s.io/reason](#)
- [insecure-sha1.invalid-cert.kubernetes.io/\\$hostname](#)
- [missing-san.invalid-cert.kubernetes.io/\\$hostname](#)
- [pod-security.kubernetes.io/audit-violations](#)
- [pod-security.kubernetes.io/enforce-policy](#)
- [pod-security.kubernetes.io/exempt](#)

See more details on [Audit Annotations](#).

kubeadm

kubeadm.alpha.kubernetes.io/cri-socket

Type: Annotation

Example: kubeadm.alpha.kubernetes.io/cri-socket: unix:///run/containerd/container.sock

Used on: Node

Annotation that kubeadm uses to preserve the CRI socket information given to kubeadm at init/join time for later use. kubeadm annotates the Node object with this information. The annotation remains "alpha", since ideally this should be a field in KubeletConfiguration instead.

kubeadm.kubernetes.io/etcđ.advertise-client-urls

Type: Annotation

Example: kubeadm.kubernetes.io/etcđ.advertise-client-urls: https://172.17.0.18:2379

Used on: Pod

Annotation that kubeadm places on locally managed etcd Pods to keep track of a list of URLs where etcd clients should connect to. This is used mainly for etcd cluster health check purposes.

kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint

Type: Annotation

Example: kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: https://172.17.0.18:6443

Used on: Pod

Annotation that kubeadm places on locally managed kube-apiserver Pods to keep track of the exposed advertise address/port endpoint for that API server instance.

kubeadm.kubernetes.io/component-config.hash

Type: Annotation

Example: `kubeadm.kubernetes.io/component-config.hash`:
2c26b46b68ffc68ff99b453c1d30413413422d706483bfa0f98a5e886266e7ae

Used on: ConfigMap

Annotation that kubeadm places on ConfigMaps that it manages for configuring components. It contains a hash (SHA-256) used to determine if the user has applied settings different from the kubeadm defaults for a particular component.

node-role.kubernetes.io/control-plane

Type: Label

Used on: Node

A marker label to indicate that the node is used to run control plane components. The kubeadm tool applies this label to the control plane nodes that it manages. Other cluster management tools typically also set this taint.

You can label control plane nodes with this label to make it easier to schedule Pods only onto these nodes, or to avoid running Pods on the control plane. If this label is set, the [EndpointSlice controller](#) ignores that node while calculating Topology Aware Hints.

node-role.kubernetes.io/control-plane

Type: Taint

Example: `node-role.kubernetes.io/control-plane:NoSchedule`

Used on: Node

Taint that kubeadm applies on control plane nodes to restrict placing Pods and allow only specific pods to schedule on them.

If this Taint is applied, control plane nodes allow only critical workloads to be scheduled onto them. You can manually remove this taint with the following command on a specific node.

```
kubectl taint nodes <node-name> node-role.kubernetes.io/control-plane:NoSchedule-
```

node-role.kubernetes.io/master (deprecated)

Type: Taint

Used on: Node

Example: `node-role.kubernetes.io/master:NoSchedule`

Taint that kubeadm previously applied on control plane nodes to allow only critical workloads to schedule on them. Replaced by the [node-role.kubernetes.io/control-plane](#) taint. kubeadm no longer sets or uses this deprecated taint.

Audit Annotations

This page serves as a reference for the audit annotations of the kubernetes.io namespace. These annotations apply to Event object from API group audit.k8s.io.

Note: The following annotations are not used within the Kubernetes API. When you [enable auditing](#) in your cluster, audit event data is written using Event from API group audit.k8s.io. The annotations apply to audit events. Audit events are different from objects in the [Event API](#) (API group events.k8s.io).

pod-security.kubernetes.io/exempt

Example: pod-security.kubernetes.io/exempt: namespace

Value **must** be one of user, namespace, or runtimeClass which correspond to [Pod Security Exemption](#) dimensions. This annotation indicates on which dimension was based the exemption from the PodSecurity enforcement.

pod-security.kubernetes.io/enforce-policy

Example: pod-security.kubernetes.io/enforce-policy: restricted:latest

Value **must** be privileged:<version>, baseline:<version>, restricted:<version> which correspond to [Pod Security Standard](#) levels accompanied by a version which **must** be latest or a valid Kubernetes version in the format v<MAJOR>.<MINOR>. This annotations informs about the enforcement level that allowed or denied the pod during PodSecurity admission.

See [Pod Security Standards](#) for more information.

pod-security.kubernetes.io/audit-violations

Example: pod-security.kubernetes.io/audit-violations: would violate PodSecurity "restricted:latest": allowPrivilegeEscalation != false (container "example" must set securityContext.allowPrivilegeEscalation=false), ...

Value details an audit policy violation, it contains the [Pod Security Standard](#) level that was transgressed as well as the specific policies on the fields that were violated from the PodSecurity enforcement.

See [Pod Security Standards](#) for more information.

authorization.k8s.io/decision

Example: authorization.k8s.io/decision: "forbid"

This annotation indicates whether or not a request was authorized in Kubernetes audit logs.

See [Auditing](#) for more information.

authorization.k8s.io/reason

Example: authorization.k8s.io/reason: "Human-readable reason for the decision"

This annotation gives reason for the [decision](#) in Kubernetes audit logs.

See [Auditing](#) for more information.

missing-san.invalid-cert.kubernetes.io/\$hostname

Example:

missing-san.invalid-cert.kubernetes.io/example-svc.example-namespace.svc: "relies on a legacy Common Name field instead of the SAN extension for subject validation"

Used by Kubernetes version v1.24 and later

This annotation indicates a webhook or aggregated API server is using an invalid certificate that is missing subjectAltNames. Support for these certificates was disabled by default in Kubernetes 1.19, and removed in Kubernetes 1.23.

Requests to endpoints using these certificates will fail. Services using these certificates should replace them as soon as possible to avoid disruption when running in Kubernetes 1.23+ environments.

There's more information about this in the Go documentation: [X.509 CommonName deprecation](#).

insecure-sha1.invalid-cert.kubernetes.io/\$hostname

Example: insecure-sha1.invalid-cert.kubernetes.io/example-svc.example-namespace.svc: "uses an insecure SHA-1 signature"

Used by Kubernetes version v1.24 and later

This annotation indicates a webhook or aggregated API server is using an insecure certificate signed with a SHA-1 hash. Support for these insecure certificates is disabled by default in Kubernetes 1.24, and will be removed in a future release.

Services using these certificates should replace them as soon as possible, to ensure connections are secured properly and to avoid disruption in future releases.

There's more information about this in the Go documentation: [Rejecting SHA-1 certificates](#).

validation.policy.admission.k8s.io/validation_failure

Example: validation.policy.admission.k8s.io/validation_failure: '[{"message": "Invalid value", "policy": "policy.example.com", {"binding": "policybinding.example.com", {"expressionIndex": "1", "validationActions": ["Audit"]}}]

Used by Kubernetes version v1.27 and later.

This annotation indicates that a admission policy validation evaluted to false for an API request, or that the validation resulted in an error while the policy was configured with failurePolicy: Fail.

The value of the annotation is a JSON object. The message in the JSON provides the message about the validation failure.

The policy, binding and expressionIndex in the JSON identifies the name of the ValidatingAdmissionPolicy, the name of the ValidatingAdmissionPolicyBinding and the index in the policy validations of the CEL expressions that failed, respectively.

The validationActions shows what actions were taken for this validation failure. See [Validating Admission Policy](#) for more details about validationActions.

Kubernetes API

Kubernetes' API is the application that serves Kubernetes functionality through a RESTful interface and stores the state of the cluster.

Kubernetes resources and "records of intent" are all stored as API objects, and modified via RESTful calls to the API. The API allows configuration to be managed in a declarative way. Users can interact with the Kubernetes API directly, or via tools like kubectl. The core Kubernetes API is flexible and can also be extended to support custom resources.

[Workload Resources](#)

[Service Resources](#)

[Config and Storage Resources](#)

[Authentication Resources](#)

[Authorization Resources](#)

[Policy Resources](#)

[Extend Resources](#)

[Cluster Resources](#)

[Common Definitions](#)

[Other Resources](#)

[Common Parameters](#)

Workload Resources

[Pod](#)

Pod is a collection of containers that can run on a host.

[PodTemplate](#)

PodTemplate describes a template for creating copies of a predefined pod.

[ReplicationController](#)

ReplicationController represents the configuration of a replication controller.

[ReplicaSet](#)

ReplicaSet ensures that a specified number of pod replicas are running at any given time.

[Deployment](#)

Deployment enables declarative updates for Pods and ReplicaSets.

[StatefulSet](#)

StatefulSet represents a set of pods with consistent identities.

[ControllerRevision](#)

ControllerRevision implements an immutable snapshot of state data.

[DaemonSet](#)

DaemonSet represents the configuration of a daemon set.

[Job](#)

Job represents the configuration of a single job.

[CronJob](#)

CronJob represents the configuration of a single cron job.

[HorizontalPodAutoscaler](#)

configuration of a horizontal pod autoscaler.

[HorizontalPodAutoscaler](#)

HorizontalPodAutoscaler is the configuration for a horizontal pod autoscaler, which automatically manages the replica count of any resource implementing the scale subresource based on the metrics specified.

[PriorityClass](#)

PriorityClass defines mapping from a priority class name to the priority integer value.

[PodSchedulingContext v1alpha2](#)

PodSchedulingContext objects hold information that is needed to schedule a Pod with ResourceClaims that use "WaitForFirstConsumer" allocation mode.

[ResourceClaim v1alpha2](#)

ResourceClaim describes which resources are needed by a resource consumer.

[ResourceClaimTemplate v1alpha2](#)

ResourceClaimTemplate is used to produce ResourceClaim objects.

[ResourceClass v1alpha2](#)

ResourceClass is used by administrators to influence how resources are allocated.

Pod

Pod is a collection of containers that can run on a host.

```
apiVersion: v1  
import "k8s.io/api/core/v1"
```

Pod

Pod is a collection of containers that can run on a host. This resource is created by clients and scheduled onto hosts.

- **apiVersion:** v1

- **kind:** Pod

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([PodSpec](#))

Specification of the desired behavior of the pod. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([PodStatus](#))

Most recently observed status of the pod. This data may not be up to date. Populated by the system. Read-only. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

PodSpec

PodSpec is a description of a pod.

Containers

- **containers** ([][Container](#)), required

Patch strategy: merge on key name

List of containers belonging to the pod. Containers cannot currently be added or removed. There must be at least one container in a Pod. Cannot be updated.

- **initContainers** ([][Container](#))

Patch strategy: merge on key name

List of initialization containers belonging to the pod. Init containers are executed in order prior to containers being started. If any init container fails, the pod is considered to have failed and is handled according to its restartPolicy. The name for an init container or normal container must be unique among all containers. Init containers may not have Lifecycle actions, Readiness probes, Liveness probes, or Startup probes. The resourceRequirements of an init container are taken into account during scheduling by finding the highest request/limit for each resource type, and then using the max of that value or the sum of the normal containers. Limits are applied to init containers in a similar fashion. Init containers cannot currently be added or removed. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>

- **ephemeralContainers** ([][EphemeralContainer](#))

Patch strategy: merge on key name

List of ephemeral containers run in this pod. Ephemeral containers may be run in an existing pod to perform user-initiated actions such as debugging. This list cannot be specified when creating a pod, and it cannot be modified by updating the pod spec. In order to add an ephemeral container to an existing pod, use the pod's ephemeralcontainers subresource.

- **imagePullSecrets** ([][LocalObjectReference](#))

Patch strategy: merge on key name

ImagePullSecrets is an optional list of references to secrets in the same namespace to use for pulling any of the images used by this PodSpec. If specified, these secrets will be passed to individual puller implementations for them to use. More info: <https://kubernetes.io/docs/concepts/containers/images#specifying-imagepullsecrets-on-a-pod>

- **enableServiceLinks** (boolean)

EnableServiceLinks indicates whether information about services should be injected into pod's environment variables, matching the syntax of Docker links. Optional: Defaults to true.

- **os** (PodOS)

Specifies the OS of the containers in the pod. Some pod and container fields are restricted if this is set.

If the OS field is set to linux, the following fields must be unset: -
securityContext.windowsOptions

If the OS field is set to windows, following fields must be unset: - spec.hostPID - spec.hostIPC - spec.hostUsers - spec.securityContext.seLinuxOptions - spec.securityContext.seccompProfile - spec.securityContext.fsGroup - spec.securityContext.fsGroupChangePolicy - spec.securityContext.sysctls - spec.shareProcessNamespace - spec.securityContext.runAsUser - spec.securityContext.runAsGroup - spec.securityContext.supplementalGroups - spec.containers[].securityContext.seLinuxOptions - spec.containers[].securityContext.seccompProfile - spec.containers[].securityContext.capabilities - spec.containers[].securityContext.readOnlyRootFilesystem - spec.containers[].securityContext.privileged - spec.containers[].securityContext.allowPrivilegeEscalation - spec.containers[].securityContext.procMount - spec.containers[].securityContext.runAsUser - spec.containers[*].securityContext.runAsGroup

PodOS defines the OS parameters of a pod.

- **os.name** (string), required

Name is the name of the operating system. The currently supported values are linux and windows. Additional value may be defined in future and can be one of: <https://github.com/opencontainers/runtime-spec/blob/master/config.md#platform-specific-configuration> Clients should expect to handle additional values and treat unrecognized values in this field as os: null

Volumes

- **volumes** ([]Volume)

Patch strategies: retainKeys, merge on key name

List of volumes that can be mounted by containers belonging to the pod. More info: <https://kubernetes.io/docs/concepts/storage/volumes>

Scheduling

- **nodeSelector** (map[string]string)

NodeSelector is a selector which must be true for the pod to fit on a node. Selector which must match a node's labels for the pod to be scheduled on that node. More info: <https://kubernetes.io/docs/concepts/configuration/assign-pod-node/>

nodeName (string)

- NodeName is a request to schedule this pod onto a specific node. If it is non-empty, the scheduler simply schedules this pod onto that node, assuming that it fits resource requirements.

• **affinity** (Affinity)

If specified, the pod's scheduling constraints

Affinity is a group of affinity scheduling rules.

◦ **affinity.nodeAffinity** ([NodeAffinity](#))

Describes node affinity scheduling rules for the pod.

◦ **affinity.podAffinity** ([PodAffinity](#))

Describes pod affinity scheduling rules (e.g. co-locate this pod in the same node, zone, etc. as some other pod(s)).

◦ **affinity.podAntiAffinity** ([PodAntiAffinity](#))

Describes pod anti-affinity scheduling rules (e.g. avoid putting this pod in the same node, zone, etc. as some other pod(s)).

• **tolerations** ([]Toleration)

If specified, the pod's tolerations.

The pod this Toleration is attached to tolerates any taint that matches the triple <key,value,effect> using the matching operator .

◦ **tolerations.key** (string)

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

◦ **tolerations.operator** (string)

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

◦ **tolerations.value** (string)

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

◦ **tolerations.effect** (string)

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

◦ **tolerations.tolerationSeconds** (int64)

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

- **schedulerName** (string)

If specified, the pod will be dispatched by specified scheduler. If not specified, the pod will be dispatched by default scheduler.

- **runtimeClassName** (string)

RuntimeClassName refers to a RuntimeClass object in the node.k8s.io group, which should be used to run this pod. If no RuntimeClass resource matches the named class, the pod will not be run. If unset or empty, the "legacy" RuntimeClass will be used, which is an implicit class with an empty definition that uses the default runtime handler. More info: <https://git.k8s.io/enhancements/keps/sig-node/585-runtime-class>

- **priorityClassName** (string)

If specified, indicates the pod's priority. "system-node-critical" and "system-cluster-critical" are two special keywords which indicate the highest priorities with the former being the highest priority. Any other name must be defined by creating a PriorityClass object with that name. If not specified, the pod priority will be default or zero if there is no default.

- **priority** (int32)

The priority value. Various system components use this field to find the priority of the pod. When Priority Admission Controller is enabled, it prevents users from setting this field. The admission controller populates this field from PriorityClassName. The higher the value, the higher the priority.

- **preemptionPolicy** (string)

PreemptionPolicy is the Policy for preempting pods with lower priority. One of Never, PreemptLowerPriority. Defaults to PreemptLowerPriority if unset.

- **topologySpreadConstraints** ([]TopologySpreadConstraint)

Patch strategy: merge on key topologyKey

Map: unique values on keys topologyKey, whenUnsatisfiable will be kept during a merge

TopologySpreadConstraints describes how a group of pods ought to spread across topology domains. Scheduler will schedule pods in a way which abides by the constraints. All topologySpreadConstraints are ANDed.

TopologySpreadConstraint specifies how to spread matching pods among the given topology.

- **topologySpreadConstraints.maxSkew** (int32), required

MaxSkew describes the degree to which pods may be unevenly distributed. When whenUnsatisfiable=DoNotSchedule, it is the maximum permitted difference between the number of matching pods in the target topology and the global

minimum. The global minimum is the minimum number of matching pods in an eligible domain or zero if the number of eligible domains is less than MinDomains. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 2/2/1: In this case, the global minimum is 1. | zone1 | zone2 | zone3 || P P | P P | P | - if MaxSkew is 1, incoming pod can only be scheduled to zone3 to become 2/2/2; scheduling it onto zone1(zone2) would make the ActualSkew(3-1) on zone1(zone2) violate MaxSkew(1). - if MaxSkew is 2, incoming pod can be scheduled onto any zone. When whenUnsatisfiable=ScheduleAnyway, it is used to give higher precedence to topologies that satisfy it. It's a required field. Default value is 1 and 0 is not allowed.

- **topologySpreadConstraints.topologyKey** (string), required

TopologyKey is the key of node labels. Nodes that have a label with this key and identical values are considered to be in the same topology. We consider each <key, value> as a "bucket", and try to put balanced number of pods into each bucket. We define a domain as a particular instance of a topology. Also, we define an eligible domain as a domain whose nodes meet the requirements of nodeAffinityPolicy and nodeTaintsPolicy. e.g. If TopologyKey is "kubernetes.io/hostname", each Node is a domain of that topology. And, if TopologyKey is "topology.kubernetes.io/zone", each zone is a domain of that topology. It's a required field.

- **topologySpreadConstraints.whenUnsatisfiable** (string), required

WhenUnsatisfiable indicates how to deal with a pod if it doesn't satisfy the spread constraint. - DoNotSchedule (default) tells the scheduler not to schedule it. - ScheduleAnyway tells the scheduler to schedule the pod in any location, but giving higher precedence to topologies that would help reduce the skew. A constraint is considered "Unsatisfiable" for an incoming pod if and only if every possible node assignment for that pod would violate "MaxSkew" on some topology. For example, in a 3-zone cluster, MaxSkew is set to 1, and pods with the same labelSelector spread as 3/1/1: | zone1 | zone2 | zone3 || P P P | P | P | If WhenUnsatisfiable is set to DoNotSchedule, incoming pod can only be scheduled to zone2(zone3) to become 3/2/1(3/1/2) as ActualSkew(2-1) on zone2(zone3) satisfies MaxSkew(1). In other words, the cluster can still be imbalanced, but scheduler won't make it *more* imbalanced. It's a required field.

- **topologySpreadConstraints.labelSelector** ([LabelSelector](#))

LabelSelector is used to find matching pods. Pods that match this label selector are counted to determine the number of pods in their corresponding topology domain.

- **topologySpreadConstraints.matchLabelKeys** ([]string)

Atomic: will be replaced during a merge

MatchLabelKeys is a set of pod label keys to select the pods over which spreading will be calculated. The keys are used to lookup values from the incoming pod labels, those key-value labels are ANDed with labelSelector to select the group of existing pods over which spreading will be calculated for the incoming pod. The same key is forbidden to exist in both MatchLabelKeys and LabelSelector.

MatchLabelKeys cannot be set when LabelSelector isn't set. Keys that don't exist in the incoming pod labels will be ignored. A null or empty list means only match against labelSelector.

This is a beta field and requires the MatchLabelKeysInPodTopologySpread feature gate to be enabled (enabled by default).

- **topologySpreadConstraints.minDomains** (int32)

MinDomains indicates a minimum number of eligible domains. When the number of eligible domains with matching topology keys is less than minDomains, Pod Topology Spread treats "global minimum" as 0, and then the calculation of Skew is performed. And when the number of eligible domains with matching topology keys equals or greater than minDomains, this value has no effect on scheduling. As a result, when the number of eligible domains is less than minDomains, scheduler won't schedule more than maxSkew Pods to those domains. If value is nil, the constraint behaves as if MinDomains is equal to 1. Valid values are integers greater than 0. When value is not nil, WhenUnsatisfiable must be DoNotSchedule.

For example, in a 3-zone cluster, MaxSkew is set to 2, MinDomains is set to 5 and pods with the same labelSelector spread as 2/2/2: | zone1 | zone2 | zone3 | | P P | P P | P P | The number of domains is less than 5(MinDomains), so "global minimum" is treated as 0. In this situation, new pod with the same labelSelector cannot be scheduled, because computed skew will be 3(3 - 0) if new Pod is scheduled to any of the three zones, it will violate MaxSkew.

This is a beta field and requires the MinDomainsInPodTopologySpread feature gate to be enabled (enabled by default).

- **topologySpreadConstraints.nodeAffinityPolicy** (string)

NodeAffinityPolicy indicates how we will treat Pod's nodeAffinity/nodeSelector when calculating pod topology spread skew. Options are: - Honor: only nodes matching nodeAffinity/nodeSelector are included in the calculations. - Ignore: nodeAffinity/nodeSelector are ignored. All nodes are included in the calculations.

If this value is nil, the behavior is equivalent to the Honor policy. This is a beta-level feature default enabled by the NodeInclusionPolicyInPodTopologySpread feature flag.

- **topologySpreadConstraints.nodeTaintsPolicy** (string)

NodeTaintsPolicy indicates how we will treat node taints when calculating pod topology spread skew. Options are: - Honor: nodes without taints, along with tainted nodes for which the incoming pod has a toleration, are included. - Ignore: node taints are ignored. All nodes are included.

If this value is nil, the behavior is equivalent to the Ignore policy. This is a beta-level feature default enabled by the NodeInclusionPolicyInPodTopologySpread feature flag.

- **overhead** (map[string][Quantity](#))

Overhead represents the resource overhead associated with running a pod for a given RuntimeClass. This field will be autopopulated at admission time by the RuntimeClass admission controller. If the RuntimeClass admission controller is enabled, overhead must not be set in Pod create requests. The RuntimeClass admission controller will reject Pod create requests which have the overhead already set. If RuntimeClass is configured and

selected in the PodSpec, Overhead will be set to the value defined in the corresponding RuntimeClass, otherwise it will remain unset and treated as zero. More info: <https://git.k8s.io/enhancements/keps/sig-node/688-pod-overhead/README.md>

Lifecycle

- **restartPolicy** (string)

Restart policy for all containers within the pod. One of Always, OnFailure, Never. In some contexts, only a subset of those values may be permitted. Default to Always. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle/#restart-policy>

- **terminationGracePeriodSeconds** (int64)

Optional duration in seconds the pod needs to terminate gracefully. May be decreased in delete request. Value must be non-negative integer. The value zero indicates stop immediately via the kill signal (no opportunity to shut down). If this value is nil, the default grace period will be used instead. The grace period is the duration in seconds after the processes running in the pod are sent a termination signal and the time when the processes are forcibly halted with a kill signal. Set this value longer than the expected cleanup time for your process. Defaults to 30 seconds.

- **activeDeadlineSeconds** (int64)

Optional duration in seconds the pod may be active on the node relative to StartTime before the system will actively try to mark it failed and kill associated containers. Value must be a positive integer.

- **readinessGates** ([]PodReadinessGate)

If specified, all readiness gates will be evaluated for pod readiness. A pod is ready when all its containers are ready AND all conditions specified in the readiness gates have status equal to "True" More info: <https://git.k8s.io/enhancements/keps/sig-network/580-pod-readiness-gates>

PodReadinessGate contains the reference to a pod condition

- **readinessGates.conditionType** (string), required

ConditionType refers to a condition in the pod's condition list with matching type.

Hostname and Name resolution

- **hostname** (string)

Specifies the hostname of the Pod If not specified, the pod's hostname will be set to a system-defined value.

- **setHostnameAsFQDN** (boolean)

If true the pod's hostname will be configured as the pod's FQDN, rather than the leaf name (the default). In Linux containers, this means setting the FQDN in the hostname field of the kernel (the nodename field of struct utsname). In Windows containers, this means setting the registry value of hostname for the registry key

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters to FQDN. If a pod does not have FQDN, this has no effect. Default to false.

- **subdomain** (string)

If specified, the fully qualified Pod hostname will be "<hostname>.<subdomain>.<pod namespace>.svc.<cluster domain>". If not specified, the pod will not have a domainname at all.

- **hostAliases** ([]HostAlias)

Patch strategy: merge on key ip

HostAliases is an optional list of hosts and IPs that will be injected into the pod's hosts file if specified. This is only valid for non-hostNetwork pods.

HostAlias holds the mapping between IP and hostnames that will be injected as an entry in the pod's hosts file.

- **hostAliases.hostnames** ([]string)

Hostnames for the above IP address.

- **hostAliases.ip** (string)

IP address of the host file entry.

- **dnsConfig** (PodDNSConfig)

Specifies the DNS parameters of a pod. Parameters specified here will be merged to the generated DNS configuration based on DNSPolicy.

PodDNSConfig defines the DNS parameters of a pod in addition to those generated from DNSPolicy.

- **dnsConfig.nameservers** ([]string)

A list of DNS name server IP addresses. This will be appended to the base nameservers generated from DNSPolicy. Duplicated nameservers will be removed.

- **dnsConfig.options** ([]PodDNSConfigOption)

A list of DNS resolver options. This will be merged with the base options generated from DNSPolicy. Duplicated entries will be removed. Resolution options given in Options will override those that appear in the base DNSPolicy.

PodDNSConfigOption defines DNS resolver options of a pod.

- **dnsConfig.options.name** (string)

Required.

- **dnsConfig.options.value** (string)

- **dnsConfig.searches** ([]string)

A list of DNS search domains for host-name lookup. This will be appended to the base search paths generated from DNSPolicy. Duplicated search paths will be removed.

- **dnsPolicy** (string)

Set DNS policy for the pod. Defaults to "ClusterFirst". Valid values are 'ClusterFirstWithHostNet', 'ClusterFirst', 'Default' or 'None'. DNS parameters given in DNSConfig will be merged with the policy selected with DNSPolicy. To have DNS options set along with hostNetwork, you have to specify DNS policy explicitly to 'ClusterFirstWithHostNet'.

Hosts namespaces

- **hostNetwork** (boolean)

Host networking requested for this pod. Use the host's network namespace. If this option is set, the ports that will be used must be specified. Default to false.

- **hostPID** (boolean)

Use the host's pid namespace. Optional: Default to false.

- **hostIPC** (boolean)

Use the host's ipc namespace. Optional: Default to false.

- **shareProcessNamespace** (boolean)

Share a single process namespace between all of the containers in a pod. When this is set containers will be able to view and signal processes from other containers in the same pod, and the first process in each container will not be assigned PID 1. HostPID and ShareProcessNamespace cannot both be set. Optional: Default to false.

Service account

- **serviceAccountName** (string)

ServiceAccountName is the name of the ServiceAccount to use to run this pod. More info: <https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/>

- **automountServiceAccountToken** (boolean)

AutomountServiceAccountToken indicates whether a service account token should be automatically mounted.

Security context

- **securityContext** (PodSecurityContext)

SecurityContext holds pod-level security attributes and common container settings. Optional: Defaults to empty. See type description for default values of each field.

PodSecurityContext holds pod-level security attributes and common container settings. Some fields are also present in container.securityContext. Field values of container.securityContext take precedence over field values of PodSecurityContext.

- **securityContext.runAsUser** (int64)

The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.runAsNonRoot** (boolean)

Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

- **securityContext.runAsGroup** (int64)

The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.supplementalGroups** ([]int64)

A list of groups applied to the first process run in each container, in addition to the container's primary GID, the fsGroup (if specified), and group memberships defined in the container image for the uid of the container process. If unspecified, no additional groups are added to any container. Note that group memberships defined in the container image for the uid of the container process are still effective, even if they are not included in this list. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.fsGroup** (int64)

A special supplemental group that applies to all containers in a pod. Some volume types allow the Kubelet to change the ownership of that volume to be owned by the pod:

1. The owning GID will be the FSGroup 2. The setgid bit is set (new files created in the volume will be owned by FSGroup) 3. The permission bits are OR'd with rw-rw----

If unset, the Kubelet will not modify the ownership and permissions of any volume. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.fsGroupChangePolicy** (string)

fsGroupChangePolicy defines behavior of changing ownership and permission of the volume before being exposed inside Pod. This field will only apply to volume types which support fsGroup based ownership(and permissions). It will have no

effect on ephemeral volume types such as: secret, configmaps and emptydir. Valid values are "OnRootMismatch" and "Always". If not specified, "Always" is used. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.seccompProfile** (SeccompProfile)

The seccomp options to use by the containers in this pod. Note that this field cannot be set when spec.os.name is windows.

SeccompProfile defines a pod/container's seccomp profile settings. Only one profile source may be set.

- **securityContext.seccompProfile.type** (string), required

type indicates which kind of seccomp profile will be applied. Valid options are:

Localhost - a profile defined in a file on the node should be used.

RuntimeDefault - the container runtime default profile should be used.

Unconfined - no profile should be applied.

- **securityContext.seccompProfile.localhostProfile** (string)

localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.

- **securityContext.seLinuxOptions** (SELinuxOptions)

The SELinux context to be applied to all containers. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in SecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence for that container. Note that this field cannot be set when spec.os.name is windows.

SELinuxOptions are the labels to be applied to the container

- **securityContext.seLinuxOptions.level** (string)

Level is SELinux level label that applies to the container.

- **securityContext.seLinuxOptions.role** (string)

Role is a SELinux role label that applies to the container.

- **securityContext.seLinuxOptions.type** (string)

Type is a SELinux type label that applies to the container.

- **securityContext.seLinuxOptions.user** (string)

User is a SELinux user label that applies to the container.

- **securityContext.sysctls** ([]Sysctl)

Sysctls hold a list of namespaced sysctls used for the pod. Pods with unsupported sysctls (by the container runtime) might fail to launch. Note that this field cannot be set when spec.os.name is windows.

Sysctl defines a kernel parameter to be set

- **securityContext.sysctls.name** (string), required

Name of a property to set

- **securityContext.sysctls.value** (string), required

Value of a property to set

- **securityContext.windowsOptions** (WindowsSecurityContextOptions)

The Windows specific settings applied to all containers. If unspecified, the options within a container's SecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

WindowsSecurityContextOptions contain Windows-specific options and credentials.

- **securityContext.windowsOptions.gmsaCredentialSpec** (string)

GMSACredentialSpec is where the GMSA admission webhook (<https://github.com/kubernetes-sigs/windows-gmsa>) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.

- **securityContext.windowsOptions.gmsaCredentialSpecName** (string)

GMSACredentialSpecName is the name of the GMSA credential spec to use.

- **securityContext.windowsOptions.hostProcess** (boolean)

HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

- **securityContext.windowsOptions.runAsUserName** (string)

The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

Alpha level

- **hostUsers** (boolean)

Use the host's user namespace. Optional: Default to true. If set to true or not present, the pod will be run in the host user namespace, useful for when the pod needs a feature only available to the host user namespace, such as loading a kernel module with

CAP_SYS_MODULE. When set to false, a new usersns is created for the pod. Setting false is useful for mitigating container breakout vulnerabilities even allowing users to run their containers as root without actually having root privileges on the host. This field is alpha-level and is only honored by servers that enable the UserNamespacesSupport feature.

- **resourceClaims** ([]PodResourceClaim)

Patch strategies: retainKeys, merge on key name

Map: unique values on key name will be kept during a merge

ResourceClaims defines which ResourceClaims must be allocated and reserved before the Pod is allowed to start. The resources will be made available to those containers which consume them by name.

This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.

This field is immutable.

PodResourceClaim references exactly one ResourceClaim through a ClaimSource. It adds a name to it that uniquely identifies the ResourceClaim inside the Pod. Containers that need access to the ResourceClaim reference it with this name.

- **resourceClaims.name** (string), required

Name uniquely identifies this resource claim inside the pod. This must be a DNS_LABEL.

- **resourceClaims.source** (ClaimSource)

Source describes where to find the ResourceClaim.

*ClaimSource describes a reference to a ResourceClaim.

Exactly one of these fields should be set. Consumers of this type must treat an empty object as if it has an unknown value.*

- **resourceClaims.source.resourceClaimName** (string)

ResourceClaimName is the name of a ResourceClaim object in the same namespace as this pod.

- **resourceClaims.source.resourceClaimTemplateName** (string)

ResourceClaimTemplateName is the name of a ResourceClaimTemplate object in the same namespace as this pod.

The template will be used to create a new ResourceClaim, which will be bound to this pod. When this pod is deleted, the ResourceClaim will also be deleted. The pod name and resource name, along with a generated component, will be used to form a unique name for the ResourceClaim, which will be recorded in pod.status.resourceClaimStatuses.

This field is immutable and no changes will be made to the corresponding ResourceClaim by the control plane after creating the ResourceClaim.

schedulingGates ([]PodSchedulingGate)

- *Patch strategy: merge on key name*

Map: unique values on key name will be kept during a merge

SchedulingGates is an opaque list of values that if specified will block scheduling the pod. If schedulingGates is not empty, the pod will stay in the SchedulingGated state and the scheduler will not attempt to schedule the pod.

SchedulingGates can only be set at pod creation time, and be removed only afterwards.

This is a beta feature enabled by the PodSchedulingReadiness feature gate.

PodSchedulingGate is associated to a Pod to guard its scheduling.

- **schedulingGates.name** (string), required

Name of the scheduling gate. Each scheduling gate must have a unique name field.

Deprecated

- **serviceAccount** (string)

DeprecatedServiceAccount is a deprecated alias for ServiceAccountName. Deprecated: Use serviceAccountName instead.

Container

A single application container that you want to run within a pod.

- **name** (string), required

Name of the container specified as a DNS_LABEL. Each container in a pod must have a unique name (DNS_LABEL). Cannot be updated.

Image

- **image** (string)

Container image name. More info: <https://kubernetes.io/docs/concepts/containers/images>
This field is optional to allow higher level config management to default or override container images in workload controllers like Deployments and StatefulSets.

- **imagePullPolicy** (string)

Image pull policy. One of Always, Never, IfNotPresent. Defaults to Always if :latest tag is specified, or IfNotPresent otherwise. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/containers/images#updating-images>

Entrypoint

- **command** ([]string)

Entrypoint array. Not executed within a shell. The container image's ENTRYPOINT is used if this is not provided. Variable references `$(VAR_NAME)` are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double `$$` are reduced to a single `$`, which allows for escaping the `$(VAR_NAME)` syntax: i.e. `"$$(VAR_NAME)"` will produce the string literal `"$ $(VAR_NAME)"`. Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: <https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell>

- **args** ([]string)

Arguments to the entrypoint. The container image's CMD is used if this is not provided. Variable references `$(VAR_NAME)` are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double `$$` are reduced to a single `$`, which allows for escaping the `$(VAR_NAME)` syntax: i.e. `"$$(VAR_NAME)"` will produce the string literal `"$(VAR_NAME)"`. Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: <https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell>

- **workingDir** (string)

Container's working directory. If not specified, the container runtime's default will be used, which might be configured in the container image. Cannot be updated.

Ports

- **ports** ([]ContainerPort)

Patch strategy: merge on key containerPort

Map: unique values on keys containerPort, protocol will be kept during a merge

List of ports to expose from the container. Not specifying a port here DOES NOT prevent that port from being exposed. Any port which is listening on the default "0.0.0.0" address inside a container will be accessible from the network. Modifying this array with strategic merge patch may corrupt the data. For more information See <https://github.com/kubernetes/kubernetes/issues/108255>. Cannot be updated.

ContainerPort represents a network port in a single container.

- **ports.containerPort** (int32), required

Number of port to expose on the pod's IP address. This must be a valid port number, $0 < x < 65536$.

- **ports.hostIP** (string)

What host IP to bind the external port to.

ports.hostPort (int32)

- Number of port to expose on the host. If specified, this must be a valid port number, $0 < x < 65536$. If HostNetwork is specified, this must match ContainerPort. Most containers do not need this.

◦ **ports.name** (string)

If specified, this must be an IANA_SVC_NAME and unique within the pod. Each named port in a pod must have a unique name. Name for the port that can be referred to by services.

◦ **ports.protocol** (string)

Protocol for port. Must be UDP, TCP, or SCTP. Defaults to "TCP".

Environment variables

• **env** ([]EnvVar)

Patch strategy: merge on key name

List of environment variables to set in the container. Cannot be updated.

EnvVar represents an environment variable present in a Container.

◦ **env.name** (string), required

Name of the environment variable. Must be a C_IDENTIFIER.

◦ **env.value** (string)

Variable references \$(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double \$\$ are reduced to a single \$, which allows for escaping the \$(VAR_NAME) syntax: i.e. "\$\$(VAR_NAME)" will produce the string literal "\$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".

◦ **env.valueFrom** (EnvVarSource)

Source for the environment variable's value. Cannot be used if value is not empty.

EnvVarSource represents a source for the value of an EnvVar.

▪ **env.valueFrom.configMapKeyRef** (ConfigMapKeySelector)

Selects a key of a ConfigMap.

Selects a key from a ConfigMap.

▪ **env.valueFrom.configMapKeyRef.key** (string), required

The key to select.

env.valueFrom.configMapKeyRef.name (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

▪ **env.valueFrom.configMapKeyRef.optional** (boolean)

Specify whether the ConfigMap or its key must be defined

▪ **env.valueFrom.fieldRef** ([ObjectFieldSelector](#))

Selects a field of the pod: supports metadata.name, metadata.namespace, metadata.labels['\<KEY>'], metadata.annotations['\<KEY>'], spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

▪ **env.valueFrom.resourceFieldRef** ([ResourceFieldSelector](#))

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

▪ **env.valueFrom.secretKeyRef** ([SecretKeySelector](#))

Selects a key of a secret in the pod's namespace

SecretKeySelector selects a key of a Secret.

▪ **env.valueFrom.secretKeyRef.key** (string), required

The key of the secret to select from. Must be a valid secret key.

▪ **env.valueFrom.secretKeyRef.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

▪ **env.valueFrom.secretKeyRef.optional** (boolean)

Specify whether the Secret or its key must be defined

• **envFrom** ([]EnvFromSource)

List of sources to populate environment variables in the container. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the container is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

EnvFromSource represents the source of a set of ConfigMaps

◦ **envFrom.configMapRef** ([ConfigMapEnvSource](#))

The ConfigMap to select from

*ConfigMapEnvSource selects a ConfigMap to populate the environment variables with.

The contents of the target ConfigMap's Data field will represent the key-value pairs as environment variables.*

- **envFrom.configMapRef.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **envFrom.configMapRef.optional** (boolean)

Specify whether the ConfigMap must be defined

- **envFrom.prefix** (string)

An optional identifier to prepend to each key in the ConfigMap. Must be a C_IDENTIFIER.

- **envFrom.secretRef** (SecretEnvSource)

The Secret to select from

*SecretEnvSource selects a Secret to populate the environment variables with.

The contents of the target Secret's Data field will represent the key-value pairs as environment variables.*

- **envFrom.secretRef.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **envFrom.secretRef.optional** (boolean)

Specify whether the Secret must be defined

Volumes

- **volumeMounts** ([]VolumeMount)

Patch strategy: merge on key mountPath

Pod volumes to mount into the container's filesystem. Cannot be updated.

VolumeMount describes a mounting of a Volume within a container.

- **volumeMounts.mountPath** (string), required

Path within the container at which the volume should be mounted. Must not contain ':'.

- **volumeMounts.name** (string), required

This must match the Name of a Volume.

- **volumeMounts.mountPropagation** (string)

mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10.

- **volumeMounts.readOnly** (boolean)

Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.

- **volumeMounts.subPath** (string)

Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).

- **volumeMounts.subPathExpr** (string)

Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references \$ (VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

- **volumeDevices** ([]VolumeDevice)

Patch strategy: merge on key devicePath

volumeDevices is the list of block devices to be used by the container.

volumeDevice describes a mapping of a raw block device within a container.

- **volumeDevices.devicePath** (string), required

devicePath is the path inside of the container that the device will be mapped to.

- **volumeDevices.name** (string), required

name must match the name of a persistentVolumeClaim in the pod

Resources

- **resources** (ResourceRequirements)

Compute Resources required by this container. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

ResourceRequirements describes the compute resource requirements.

- **resources.claims** ([]ResourceClaim)

Map: unique values on key name will be kept during a merge

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.

This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.

This field is immutable. It can only be set for containers.

ResourceClaim references one entry in PodSpec.ResourceClaims.

- **resources.claims.name** (string), required

Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.

- **resources.limits** (map[string][Quantity](#))

Limits describes the maximum amount of compute resources allowed. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

- **resources.requests** (map[string][Quantity](#))

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

- **resizePolicy** ([]ContainerResizePolicy)

Atomic: will be replaced during a merge

Resources resize policy for the container.

ContainerResizePolicy represents resource resize policy for the container.

- **resizePolicy.resourceName** (string), required

Name of the resource to which this resource resize policy applies. Supported values: cpu, memory.

- **resizePolicy.restartPolicy** (string), required

Restart policy to apply when specified resource is resized. If not specified, it defaults to NotRequired.

Lifecycle

- **lifecycle** (Lifecycle)

Actions that the management system should take in response to container lifecycle events. Cannot be updated.

Lifecycle describes actions that the management system should take in response to container lifecycle events. For the PostStart and PreStop lifecycle handlers, management of the container blocks until the action is complete, unless the container process fails, in which case the handler is aborted.

- **lifecycle.postStart** ([LifecycleHandler](#))

PostStart is called immediately after a container is created. If the handler fails, the container is terminated and restarted according to its restart policy. Other management of the container blocks until the hook completes. More info: <https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks>

- **lifecycle.preStop** ([LifecycleHandler](#))

PreStop is called immediately before a container is terminated due to an API request or management event such as liveness/startup probe failure, preemption, resource contention, etc. The handler is not called if the container crashes or exits. The Pod's termination grace period countdown begins before the PreStop hook is executed. Regardless of the outcome of the handler, the container will eventually terminate within the Pod's termination grace period (unless delayed by finalizers). Other management of the container blocks until the hook completes or until the termination grace period is reached. More info: <https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks>

- **terminationMessagePath** (string)

Optional: Path at which the file to which the container's termination message will be written is mounted into the container's filesystem. Message written is intended to be brief final status, such as an assertion failure message. Will be truncated by the node if greater than 4096 bytes. The total message length across all containers will be limited to 12kb. Defaults to /dev/termination-log. Cannot be updated.

- **terminationMessagePolicy** (string)

Indicate how the termination message should be populated. File will use the contents of terminationMessagePath to populate the container status message on both success and failure. FallbackToLogsOnError will use the last chunk of container log output if the termination message file is empty and the container exited with an error. The log output is limited to 2048 bytes or 80 lines, whichever is smaller. Defaults to File. Cannot be updated.

- **livenessProbe** ([Probe](#))

Periodic probe of container liveness. Container will be restarted if the probe fails. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes>

- **readinessProbe** ([Probe](#))

Periodic probe of container service readiness. Container will be removed from service endpoints if the probe fails. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes>

- **startupProbe** ([Probe](#))

StartupProbe indicates that the Pod has successfully initialized. If specified, no other probes are executed until this completes successfully. If this probe fails, the Pod will be restarted, just as if the livenessProbe failed. This can be used to provide different probe parameters at the beginning of a Pod's lifecycle, when it might take a long time to load

data or warm a cache, than during steady-state operation. This cannot be updated. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes>

- **restartPolicy** (string)

RestartPolicy defines the restart behavior of individual containers in a pod. This field may only be set for init containers, and the only allowed value is "Always". For non-init containers or when this field is not specified, the restart behavior is defined by the Pod's restart policy and the container type. Setting the RestartPolicy as "Always" for the init container will have the following effect: this init container will be continually restarted on exit until all regular containers have terminated. Once all regular containers have completed, all init containers with restartPolicy "Always" will be shut down. This lifecycle differs from normal init containers and is often referred to as a "sidecar" container. Although this init container still starts in the init container sequence, it does not wait for the container to complete before proceeding to the next init container. Instead, the next init container starts immediately after this init container is started, or after any startupProbe has successfully completed.

Security Context

- **securityContext** (SecurityContext)

SecurityContext defines the security options the container should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext. More info: <https://kubernetes.io/docs/tasks/configure-pod-container/security-context/>

SecurityContext holds security configuration that will be applied to a container. Some fields are present in both SecurityContext and PodSecurityContext. When both are set, the values in SecurityContext take precedence.

- **securityContext.runAsUser** (int64)

The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.runAsNonRoot** (boolean)

Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

- **securityContext.runAsGroup** (int64)

The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.readOnlyRootFilesystem** (boolean)

Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.procMount** (string)

procMount denotes the type of proc mount to use for the containers. The default is DefaultProcMount which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.privileged** (boolean)

Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.allowPrivilegeEscalation** (boolean)

AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1) run as Privileged 2) has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.

- **securityContext.capabilities** (Capabilities)

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Adds and removes POSIX capabilities from running containers.

- **securityContext.capabilities.add** ([]string)

Added capabilities

- **securityContext.capabilities.drop** ([]string)

Removed capabilities

- **securityContext.seccompProfile** (SeccompProfile)

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

SeccompProfile defines a pod/container's seccomp profile settings. Only one profile source may be set.

- **securityContext.seccompProfile.type** (string), required

type indicates which kind of seccomp profile will be applied. Valid options are:

Localhost - a profile defined in a file on the node should be used.
RuntimeDefault - the container runtime default profile should be used.
Unconfined - no profile should be applied.

- **securityContext.seccompProfile.localhostProfile** (string)

localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.

- **securityContext.seLinuxOptions** (SELinuxOptions)

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

SELinuxOptions are the labels to be applied to the container

- **securityContext.seLinuxOptions.level** (string)

Level is SELinux level label that applies to the container.

- **securityContext.seLinuxOptions.role** (string)

Role is a SELinux role label that applies to the container.

- **securityContext.seLinuxOptions.type** (string)

Type is a SELinux type label that applies to the container.

- **securityContext.seLinuxOptions.user** (string)

User is a SELinux user label that applies to the container.

- **securityContext.windowsOptions** (WindowsSecurityContextOptions)

The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

WindowsSecurityContextOptions contain Windows-specific options and credentials.

- **securityContext.windowsOptions.gmsaCredentialSpec** (string)

GMSACredentialSpec is where the GMSA admission webhook (<https://github.com/kubernetes-sigs/windows-gmsa>) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.

- **securityContext.windowsOptions.gmsaCredentialSpecName** (string)

GMSACredentialSpecName is the name of the GMSA credential spec to use.

securityContext.windowsOptions.hostProcess (boolean)

HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

▪ securityContext.windowsOptions.runAsUserName (string)

The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

Debugging

• stdin (boolean)

Whether this container should allocate a buffer for stdin in the container runtime. If this is not set, reads from stdin in the container will always result in EOF. Default is false.

• stdinOnce (boolean)

Whether the container runtime should close the stdin channel after it has been opened by a single attach. When stdin is true the stdin stream will remain open across multiple attach sessions. If stdinOnce is set to true, stdin is opened on container start, is empty until the first client attaches to stdin, and then remains open and accepts data until the client disconnects, at which time stdin is closed and remains closed until the container is restarted. If this flag is false, a container processes that reads from stdin will never receive an EOF. Default is false

• tty (boolean)

Whether this container should allocate a TTY for itself, also requires 'stdin' to be true. Default is false.

EphemeralContainer

An EphemeralContainer is a temporary container that you may add to an existing Pod for user-initiated activities such as debugging. Ephemeral containers have no resource or scheduling guarantees, and they will not be restarted when they exit or when a Pod is removed or restarted. The kubelet may evict a Pod if an ephemeral container causes the Pod to exceed its resource allocation.

To add an ephemeral container, use the ephemeralcontainers subresource of an existing Pod. Ephemeral containers may not be removed or restarted.

• name (string), required

Name of the ephemeral container specified as a DNS_LABEL. This name must be unique among all containers, init containers and ephemeral containers.

targetContainerName (string)

- If set, the name of the container from PodSpec that this ephemeral container targets. The ephemeral container will be run in the namespaces (IPC, PID, etc) of this container. If not set then the ephemeral container uses the namespaces configured in the Pod spec.

The container runtime must implement support for this feature. If the runtime does not support namespace targeting then the result of setting this field is undefined.

Image

- **image** (string)

Container image name. More info: <https://kubernetes.io/docs/concepts/containers/images>

- **imagePullPolicy** (string)

Image pull policy. One of Always, Never, IfNotPresent. Defaults to Always if :latest tag is specified, or IfNotPresent otherwise. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/containers/images#updating-images>

Entrypoint

- **command** ([]string)

Entrypoint array. Not executed within a shell. The image's ENTRYPOINT is used if this is not provided. Variable references \$(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double \$\$ are reduced to a single \$, which allows for escaping the \$(VAR_NAME) syntax: i.e. "\$\$(VAR_NAME)" will produce the string literal "\$ \$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: <https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell>

- **args** ([]string)

Arguments to the entrypoint. The image's CMD is used if this is not provided. Variable references \$(VAR_NAME) are expanded using the container's environment. If a variable cannot be resolved, the reference in the input string will be unchanged. Double \$\$ are reduced to a single \$, which allows for escaping the \$(VAR_NAME) syntax: i.e. "\$\$(VAR_NAME)" will produce the string literal "\$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Cannot be updated. More info: <https://kubernetes.io/docs/tasks/inject-data-application/define-command-argument-container/#running-a-command-in-a-shell>

- **workingDir** (string)

Container's working directory. If not specified, the container runtime's default will be used, which might be configured in the container image. Cannot be updated.

Environment variables

- **env** ([]EnvVar)

Patch strategy: merge on key name

List of environment variables to set in the container. Cannot be updated.

EnvVar represents an environment variable present in a Container.

- **env.name** (string), required

Name of the environment variable. Must be a C_IDENTIFIER.

- **env.value** (string)

Variable references \$(VAR_NAME) are expanded using the previously defined environment variables in the container and any service environment variables. If a variable cannot be resolved, the reference in the input string will be unchanged. Double \$\$ are reduced to a single \$, which allows for escaping the \$(VAR_NAME) syntax: i.e. "\$\$(VAR_NAME)" will produce the string literal "\$(VAR_NAME)". Escaped references will never be expanded, regardless of whether the variable exists or not. Defaults to "".

- **env.valueFrom** (EnvVarSource)

Source for the environment variable's value. Cannot be used if value is not empty.

EnvVarSource represents a source for the value of an EnvVar.

- **env.valueFrom.configMapKeyRef** (ConfigMapKeySelector)

Selects a key of a ConfigMap.

Selects a key from a ConfigMap.

- **env.valueFrom.configMapKeyRef.key** (string), required

The key to select.

- **env.valueFrom.configMapKeyRef.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **env.valueFrom.configMapKeyRef.optional** (boolean)

Specify whether the ConfigMap or its key must be defined

- **env.valueFrom.fieldRef** ([ObjectFieldSelector](#))

Selects a field of the pod: supports metadata.name, metadata.namespace, metadata.labels['<KEY>'], metadata.annotations['<KEY>'], spec.nodeName, spec.serviceAccountName, status.hostIP, status.podIP, status.podIPs.

env.valueFrom.resourceFieldRef (ResourceFieldSelector)

- Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, limits.ephemeral-storage, requests.cpu, requests.memory and requests.ephemeral-storage) are currently supported.

- **env.valueFrom.secretKeyRef (SecretKeySelector)**

Selects a key of a secret in the pod's namespace

SecretKeySelector selects a key of a Secret.

- **env.valueFrom.secretKeyRef.key (string), required**

The key of the secret to select from. Must be a valid secret key.

- **env.valueFrom.secretKeyRef.name (string)**

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **env.valueFrom.secretKeyRef.optional (boolean)**

Specify whether the Secret or its key must be defined

- **envFrom ([]EnvFromSource)**

List of sources to populate environment variables in the container. The keys defined within a source must be a C_IDENTIFIER. All invalid keys will be reported as an event when the container is starting. When a key exists in multiple sources, the value associated with the last source will take precedence. Values defined by an Env with a duplicate key will take precedence. Cannot be updated.

EnvFromSource represents the source of a set of ConfigMaps

- **envFrom.configMapRef (ConfigMapEnvSource)**

The ConfigMap to select from

*ConfigMapEnvSource selects a ConfigMap to populate the environment variables with.

The contents of the target ConfigMap's Data field will represent the key-value pairs as environment variables.*

- **envFrom.configMapRef.name (string)**

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **envFrom.configMapRef.optional (boolean)**

Specify whether the ConfigMap must be defined

- **envFrom.prefix (string)**

An optional identifier to prepend to each key in the ConfigMap. Must be a C_IDENTIFIER.

- **envFrom.secretRef** (SecretEnvSource)

The Secret to select from

*SecretEnvSource selects a Secret to populate the environment variables with.

The contents of the target Secret's Data field will represent the key-value pairs as environment variables.*

- **envFrom.secretRef.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **envFrom.secretRef.optional** (boolean)

Specify whether the Secret must be defined

Volumes

- **volumeMounts** ([]VolumeMount)

Patch strategy: merge on key mountPath

Pod volumes to mount into the container's filesystem. Subpath mounts are not allowed for ephemeral containers. Cannot be updated.

VolumeMount describes a mounting of a Volume within a container.

- **volumeMounts.mountPath** (string), required

Path within the container at which the volume should be mounted. Must not contain ':'.

- **volumeMounts.name** (string), required

This must match the Name of a Volume.

- **volumeMounts.mountPropagation** (string)

mountPropagation determines how mounts are propagated from the host to container and the other way around. When not set, MountPropagationNone is used. This field is beta in 1.10.

- **volumeMounts.readOnly** (boolean)

Mounted read-only if true, read-write otherwise (false or unspecified). Defaults to false.

- **volumeMounts.subPath** (string)

Path within the volume from which the container's volume should be mounted. Defaults to "" (volume's root).

- **volumeMounts.subPathExpr** (string)

Expanded path within the volume from which the container's volume should be mounted. Behaves similarly to SubPath but environment variable references \$ (VAR_NAME) are expanded using the container's environment. Defaults to "" (volume's root). SubPathExpr and SubPath are mutually exclusive.

- **volumeDevices** ([]VolumeDevice)

Patch strategy: merge on key devicePath

volumeDevices is the list of block devices to be used by the container.

volumeDevice describes a mapping of a raw block device within a container.

- **volumeDevices.devicePath** (string), required

devicePath is the path inside of the container that the device will be mapped to.

- **volumeDevices.name** (string), required

name must match the name of a persistentVolumeClaim in the pod

Resources

- **resizePolicy** ([]ContainerResizePolicy)

Atomic: will be replaced during a merge

Resources resize policy for the container.

ContainerResizePolicy represents resource resize policy for the container.

- **resizePolicy.resourceName** (string), required

Name of the resource to which this resource resize policy applies. Supported values: cpu, memory.

- **resizePolicy.restartPolicy** (string), required

Restart policy to apply when specified resource is resized. If not specified, it defaults to NotRequired.

Lifecycle

- **terminationMessagePath** (string)

Optional: Path at which the file to which the container's termination message will be written is mounted into the container's filesystem. Message written is intended to be brief final status, such as an assertion failure message. Will be truncated by the node if greater than 4096 bytes. The total message length across all containers will be limited to 12kb. Defaults to /dev/termination-log. Cannot be updated.

terminationMessagePolicy (string)

- Indicate how the termination message should be populated. File will use the contents of terminationMessagePath to populate the container status message on both success and failure. FallbackToLogsOnError will use the last chunk of container log output if the termination message file is empty and the container exited with an error. The log output is limited to 2048 bytes or 80 lines, whichever is smaller. Defaults to File. Cannot be updated.

- **restartPolicy** (string)

Restart policy for the container to manage the restart behavior of each container within a pod. This may only be set for init containers. You cannot set this field on ephemeral containers.

Debugging

- **stdin** (boolean)

Whether this container should allocate a buffer for stdin in the container runtime. If this is not set, reads from stdin in the container will always result in EOF. Default is false.

- **stdinOnce** (boolean)

Whether the container runtime should close the stdin channel after it has been opened by a single attach. When stdin is true the stdin stream will remain open across multiple attach sessions. If stdinOnce is set to true, stdin is opened on container start, is empty until the first client attaches to stdin, and then remains open and accepts data until the client disconnects, at which time stdin is closed and remains closed until the container is restarted. If this flag is false, a container processes that reads from stdin will never receive an EOF. Default is false

- **tty** (boolean)

Whether this container should allocate a TTY for itself, also requires 'stdin' to be true. Default is false.

Security context

- **securityContext** (SecurityContext)

Optional: SecurityContext defines the security options the ephemeral container should be run with. If set, the fields of SecurityContext override the equivalent fields of PodSecurityContext.

SecurityContext holds security configuration that will be applied to a container. Some fields are present in both SecurityContext and PodSecurityContext. When both are set, the values in SecurityContext take precedence.

- **securityContext.runAsUser** (int64)

The UID to run the entrypoint of the container process. Defaults to user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both

SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.runAsNonRoot** (boolean)

Indicates that the container must run as a non-root user. If true, the Kubelet will validate the image at runtime to ensure that it does not run as UID 0 (root) and fail to start the container if it does. If unset or false, no such validation will be performed. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

- **securityContext.runAsGroup** (int64)

The GID to run the entrypoint of the container process. Uses runtime default if unset. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.readOnlyRootFilesystem** (boolean)

Whether this container has a read-only root filesystem. Default is false. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.procMount** (string)

procMount denotes the type of proc mount to use for the containers. The default is DefaultProcMount which uses the container runtime defaults for readonly paths and masked paths. This requires the ProcMountType feature flag to be enabled. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.privileged** (boolean)

Run container in privileged mode. Processes in privileged containers are essentially equivalent to root on the host. Defaults to false. Note that this field cannot be set when spec.os.name is windows.

- **securityContext.allowPrivilegeEscalation** (boolean)

AllowPrivilegeEscalation controls whether a process can gain more privileges than its parent process. This bool directly controls if the no_new_privs flag will be set on the container process. AllowPrivilegeEscalation is true always when the container is: 1) run as Privileged 2) has CAP_SYS_ADMIN Note that this field cannot be set when spec.os.name is windows.

- **securityContext.capabilities** (Capabilities)

The capabilities to add/drop when running containers. Defaults to the default set of capabilities granted by the container runtime. Note that this field cannot be set when spec.os.name is windows.

Adds and removes POSIX capabilities from running containers.

- **securityContext.capabilities.add** ([]string)

Added capabilities

securityContext.capabilities.drop ([]string)

- Removed capabilities

- **securityContext.seccompProfile** (SeccompProfile)

The seccomp options to use by this container. If seccomp options are provided at both the pod & container level, the container options override the pod options. Note that this field cannot be set when spec.os.name is windows.

SeccompProfile defines a pod/container's seccomp profile settings. Only one profile source may be set.

- **securityContext.seccompProfile.type** (string), required

type indicates which kind of seccomp profile will be applied. Valid options are:

Localhost - a profile defined in a file on the node should be used.

RuntimeDefault - the container runtime default profile should be used.

Unconfined - no profile should be applied.

- **securityContext.seccompProfile.localhostProfile** (string)

localhostProfile indicates a profile defined in a file on the node should be used. The profile must be preconfigured on the node to work. Must be a descending path, relative to the kubelet's configured seccomp profile location. Must be set if type is "Localhost". Must NOT be set for any other type.

- **securityContext.seLinuxOptions** (SELinuxOptions)

The SELinux context to be applied to the container. If unspecified, the container runtime will allocate a random SELinux context for each container. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is windows.

SELinuxOptions are the labels to be applied to the container

- **securityContext.seLinuxOptions.level** (string)

Level is SELinux level label that applies to the container.

- **securityContext.seLinuxOptions.role** (string)

Role is a SELinux role label that applies to the container.

- **securityContext.seLinuxOptions.type** (string)

Type is a SELinux type label that applies to the container.

- **securityContext.seLinuxOptions.user** (string)

User is a SELinux user label that applies to the container.

securityContext.windowsOptions (WindowsSecurityContextOptions)

- The Windows specific settings applied to all containers. If unspecified, the options from the PodSecurityContext will be used. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence. Note that this field cannot be set when spec.os.name is linux.

WindowsSecurityContextOptions contain Windows-specific options and credentials.

- **securityContext.windowsOptions.gmsaCredentialSpec** (string)

GMSACredentialSpec is where the GMSA admission webhook (<https://github.com/kubernetes-sigs/windows-gmsa>) inlines the contents of the GMSA credential spec named by the GMSACredentialSpecName field.

- **securityContext.windowsOptions.gmsaCredentialSpecName** (string)

GMSACredentialSpecName is the name of the GMSA credential spec to use.

- **securityContext.windowsOptions.hostProcess** (boolean)

HostProcess determines if a container should be run as a 'Host Process' container. All of a Pod's containers must have the same effective HostProcess value (it is not allowed to have a mix of HostProcess containers and non-HostProcess containers). In addition, if HostProcess is true then HostNetwork must also be set to true.

- **securityContext.windowsOptions.runAsUserName** (string)

The UserName in Windows to run the entrypoint of the container process. Defaults to the user specified in image metadata if unspecified. May also be set in PodSecurityContext. If set in both SecurityContext and PodSecurityContext, the value specified in SecurityContext takes precedence.

Not allowed

- **ports** ([]ContainerPort)

Patch strategy: merge on key containerPort

Map: unique values on keys containerPort, protocol will be kept during a merge

Ports are not allowed for ephemeral containers.

ContainerPort represents a network port in a single container.

- **ports.containerPort** (int32), required

Number of port to expose on the pod's IP address. This must be a valid port number, $0 < x < 65536$.

- **ports.hostIP** (string)

What host IP to bind the external port to.

ports.hostPort (int32)

- Number of port to expose on the host. If specified, this must be a valid port number, $0 < x < 65536$. If HostNetwork is specified, this must match ContainerPort. Most containers do not need this.

◦ **ports.name** (string)

If specified, this must be an IANA_SVC_NAME and unique within the pod. Each named port in a pod must have a unique name. Name for the port that can be referred to by services.

◦ **ports.protocol** (string)

Protocol for port. Must be UDP, TCP, or SCTP. Defaults to "TCP".

• **resources** (ResourceRequirements)

Resources are not allowed for ephemeral containers. Ephemeral containers use spare resources already allocated to the pod.

ResourceRequirements describes the compute resource requirements.

◦ **resources.claims** ([]ResourceClaim)

Map: unique values on key name will be kept during a merge

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.

This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.

This field is immutable. It can only be set for containers.

ResourceClaim references one entry in PodSpec.ResourceClaims.

▪ **resources.claims.name** (string), required

Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.

◦ **resources.limits** (map[string][Quantity](#))

Limits describes the maximum amount of compute resources allowed. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

◦ **resources.requests** (map[string][Quantity](#))

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

lifecycle (Lifecycle)

- Lifecycle is not allowed for ephemeral containers.

Lifecycle describes actions that the management system should take in response to container lifecycle events. For the PostStart and PreStop lifecycle handlers, management of the container blocks until the action is complete, unless the container process fails, in which case the handler is aborted.

- **lifecycle.postStart** ([LifecycleHandler](#))

PostStart is called immediately after a container is created. If the handler fails, the container is terminated and restarted according to its restart policy. Other management of the container blocks until the hook completes. More info: <https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks>

- **lifecycle.preStop** ([LifecycleHandler](#))

PreStop is called immediately before a container is terminated due to an API request or management event such as liveness/startup probe failure, preemption, resource contention, etc. The handler is not called if the container crashes or exits. The Pod's termination grace period countdown begins before the PreStop hook is executed. Regardless of the outcome of the handler, the container will eventually terminate within the Pod's termination grace period (unless delayed by finalizers). Other management of the container blocks until the hook completes or until the termination grace period is reached. More info: <https://kubernetes.io/docs/concepts/containers/container-lifecycle-hooks/#container-hooks>

- **livenessProbe** ([Probe](#))

Probes are not allowed for ephemeral containers.

- **readinessProbe** ([Probe](#))

Probes are not allowed for ephemeral containers.

- **startupProbe** ([Probe](#))

Probes are not allowed for ephemeral containers.

LifecycleHandler

LifecycleHandler defines a specific action that should be taken in a lifecycle hook. One and only one of the fields, except TCPSocket must be specified.

- **exec** ([ExecAction](#))

Exec specifies the action to take.

ExecAction describes a "run in container" action.

- **exec.command** ([]string)

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('!', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

- **httpGet** (HTTPGetAction)

HTTPGet specifies the http request to perform.

HTTPGetAction describes an action based on HTTP Get requests.

- **httpGet.port** (IntOrString), required

Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **httpGet.host** (string)

Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.

- **httpGet.httpHeaders** ([]HTTPHeader)

Custom headers to set in the request. HTTP allows repeated headers.

HTTPHeader describes a custom header to be used in HTTP probes

- **httpGet.httpHeaders.name** (string), required

The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.

- **httpGet.httpHeaders.value** (string), required

The header field value

- **httpGet.path** (string)

Path to access on the HTTP server.

- **httpGet.scheme** (string)

Scheme to use for connecting to the host. Defaults to HTTP.

- **tcpSocket** (TCPSocketAction)

Deprecated. TCPSocket is NOT supported as a LifecycleHandler and kept for the backward compatibility. There are no validation of this field and lifecycle hooks will fail in runtime when tcp handler is specified.

TCPSocketAction describes an action based on opening a socket

- **tcpSocket.port** (IntOrString), required

Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **tcpSocket.host** (string)

Optional: Host name to connect to, defaults to the pod IP.

NodeAffinity

Node affinity is a group of node affinity scheduling rules.

- **preferredDuringSchedulingIgnoredDuringExecution** ([]PreferredSchedulingTerm)

The scheduler will prefer to schedule pods to nodes that satisfy the affinity expressions specified by this field, but it may choose a node that violates one or more of the expressions. The node that is most preferred is the one with the greatest sum of weights, i.e. for each node that meets all of the scheduling requirements (resource request, requiredDuringScheduling affinity expressions, etc.), compute a sum by iterating through the elements of this field and adding "weight" to the sum if the node matches the corresponding matchExpressions; the node(s) with the highest sum are the most preferred.

An empty preferred scheduling term matches all objects with implicit weight 0 (i.e. it's a no-op). A null preferred scheduling term matches no objects (i.e. is also a no-op).

- **preferredDuringSchedulingIgnoredDuringExecution.preference** (NodeSelectorTerm), required

A node selector term, associated with the corresponding weight.

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

- **preferredDuringSchedulingIgnoredDuringExecution.preference.matchExpressions** ([]NodeSelectorRequirement)

A list of node selector requirements by node's labels.

- **preferredDuringSchedulingIgnoredDuringExecution.preference.matchFields** ([]NodeSelectorRequirement)

A list of node selector requirements by node's fields.

- **preferredDuringSchedulingIgnoredDuringExecution.weight** (int32), required

Weight associated with matching the corresponding nodeSelectorTerm, in the range 1-100.

- **requiredDuringSchedulingIgnoredDuringExecution** (NodeSelector)

If the affinity requirements specified by this field are not met at scheduling time, the pod will not be scheduled onto the node. If the affinity requirements specified by this field cease to be met at some point during pod execution (e.g. due to an update), the system may or may not try to eventually evict the pod from its node.

A node selector represents the union of the results of one or more label queries over a set of nodes; that is, it represents the OR of the selectors represented by the node selector terms.

- **requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms** ([]NodeSelectorTerm), required

Required. A list of node selector terms. The terms are ORed.

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

- **requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms.matchExpressions** ([]NodeSelectorRequirement)

A list of node selector requirements by node's labels.

- **requiredDuringSchedulingIgnoredDuringExecution.nodeSelectorTerms.matchFields** ([]NodeSelectorRequirement)

A list of node selector requirements by node's fields.

PodAffinity

Pod affinity is a group of inter pod affinity scheduling rules.

- **preferredDuringSchedulingIgnoredDuringExecution** ([]WeightedPodAffinityTerm)

The scheduler will prefer to schedule pods to nodes that satisfy the affinity expressions specified by this field, but it may choose a node that violates one or more of the expressions. The node that is most preferred is the one with the greatest sum of weights, i.e. for each node that meets all of the scheduling requirements (resource request, requiredDuringScheduling affinity expressions, etc.), compute a sum by iterating through the elements of this field and adding "weight" to the sum if the node has pods which matches the corresponding podAffinityTerm; the node(s) with the highest sum are the most preferred.

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm** (PodAffinityTerm), required

Required. A pod affinity term, associated with the corresponding weight.

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.topologyKey** (string), required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.labelSelector** ([LabelSelector](#))

A label query over a set of resources, in this case pods.

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.namespaceSelector** ([LabelSelector](#))

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.namespaces** ([]string)

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

- **preferredDuringSchedulingIgnoredDuringExecution.weight** (int32), required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

- **requiredDuringSchedulingIgnoredDuringExecution** ([]PodAffinityTerm)

If the affinity requirements specified by this field are not met at scheduling time, the pod will not be scheduled onto the node. If the affinity requirements specified by this field cease to be met at some point during pod execution (e.g. due to a pod label update), the system may or may not try to eventually evict the pod from its node. When there are multiple elements, the lists of nodes corresponding to each podAffinityTerm are intersected, i.e. all terms must be satisfied.

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity)

with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

- **requiredDuringSchedulingIgnoredDuringExecution.topologyKey** (string), required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

- **requiredDuringSchedulingIgnoredDuringExecution.labelSelector** ([LabelSelector](#))

A label query over a set of resources, in this case pods.

- **requiredDuringSchedulingIgnoredDuringExecution.namespaceSelector** ([LabelSelector](#))

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

- **requiredDuringSchedulingIgnoredDuringExecution.namespaces** ([]string)

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

PodAntiAffinity

Pod anti affinity is a group of inter pod anti affinity scheduling rules.

- **preferredDuringSchedulingIgnoredDuringExecution** ([]WeightedPodAffinityTerm)

The scheduler will prefer to schedule pods to nodes that satisfy the anti-affinity expressions specified by this field, but it may choose a node that violates one or more of the expressions. The node that is most preferred is the one with the greatest sum of weights, i.e. for each node that meets all of the scheduling requirements (resource request, requiredDuringScheduling anti-affinity expressions, etc.), compute a sum by iterating through the elements of this field and adding "weight" to the sum if the node has pods which matches the corresponding podAffinityTerm; the node(s) with the highest sum are the most preferred.

The weights of all of the matched WeightedPodAffinityTerm fields are added per-node to find the most preferred node(s)

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm** (PodAffinityTerm), required

Required. A pod affinity term, associated with the corresponding weight.

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity) with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.topologyKey** (string), required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.labelSelector** ([LabelSelector](#))

A label query over a set of resources, in this case pods.

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.namespaces** ([LabelSelector](#))

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector (()) matches all namespaces.

- **preferredDuringSchedulingIgnoredDuringExecution.podAffinityTerm.namespaces** ([]string)

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

- **preferredDuringSchedulingIgnoredDuringExecution.weight** (int32), required

weight associated with matching the corresponding podAffinityTerm, in the range 1-100.

- **requiredDuringSchedulingIgnoredDuringExecution** ([]PodAffinityTerm)

If the anti-affinity requirements specified by this field are not met at scheduling time, the pod will not be scheduled onto the node. If the anti-affinity requirements specified by this field cease to be met at some point during pod execution (e.g. due to a pod label update), the system may or may not try to eventually evict the pod from its node. When there are multiple elements, the lists of nodes corresponding to each podAffinityTerm are intersected, i.e. all terms must be satisfied.

Defines a set of pods (namely those matching the labelSelector relative to the given namespace(s)) that this pod should be co-located (affinity) or not co-located (anti-affinity)

with, where co-located is defined as running on a node whose value of the label with key matches that of any node on which a pod of the set of pods is running

- **requiredDuringSchedulingIgnoredDuringExecution.topologyKey** (string), required

This pod should be co-located (affinity) or not co-located (anti-affinity) with the pods matching the labelSelector in the specified namespaces, where co-located is defined as running on a node whose value of the label with key topologyKey matches that of any node on which any of the selected pods is running. Empty topologyKey is not allowed.

- **requiredDuringSchedulingIgnoredDuringExecution.labelSelector** ([LabelSelector](#))

A label query over a set of resources, in this case pods.

- **requiredDuringSchedulingIgnoredDuringExecution.namespaceSelector** ([LabelSelector](#))

A label query over the set of namespaces that the term applies to. The term is applied to the union of the namespaces selected by this field and the ones listed in the namespaces field. null selector and null or empty namespaces list means "this pod's namespace". An empty selector ({}) matches all namespaces.

- **requiredDuringSchedulingIgnoredDuringExecution.namespaces** ([]string)

namespaces specifies a static list of namespace names that the term applies to. The term is applied to the union of the namespaces listed in this field and the ones selected by namespaceSelector. null or empty namespaces list and null namespaceSelector means "this pod's namespace".

Probe

Probe describes a health check to be performed against a container to determine whether it is alive or ready to receive traffic.

- **exec** (ExecAction)

Exec specifies the action to take.

ExecAction describes a "run in container" action.

- **exec.command** ([]string)

Command is the command line to execute inside the container, the working directory for the command is root ('/') in the container's filesystem. The command is simply exec'd, it is not run inside a shell, so traditional shell instructions ('|', etc) won't work. To use a shell, you need to explicitly call out to that shell. Exit status of 0 is treated as live/healthy and non-zero is unhealthy.

- **httpGet** (HTTPGetAction)

HTTPGet specifies the http request to perform.

HTTPGetAction describes an action based on HTTP Get requests.

- **httpGet.port** (IntOrString), required

Name or number of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **httpGet.host** (string)

Host name to connect to, defaults to the pod IP. You probably want to set "Host" in httpHeaders instead.

- **httpGet.httpHeaders** ([]HTTPHeader)

Custom headers to set in the request. HTTP allows repeated headers.

HTTPHeader describes a custom header to be used in HTTP probes

- **httpGet.httpHeaders.name** (string), required

The header field name. This will be canonicalized upon output, so case-variant names will be understood as the same header.

- **httpGet.httpHeaders.value** (string), required

The header field value

- **httpGet.path** (string)

Path to access on the HTTP server.

- **httpGet.scheme** (string)

Scheme to use for connecting to the host. Defaults to HTTP.

- **tcpSocket** (TCPSocketAction)

TCPSocket specifies an action involving a TCP port.

TCPSocketAction describes an action based on opening a socket

- **tcpSocket.port** (IntOrString), required

Number or name of the port to access on the container. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

tcpSocket.host (string)

- - Optional: Host name to connect to, defaults to the pod IP.

• **initialDelaySeconds** (int32)

Number of seconds after the container has started before liveness probes are initiated.

More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes>

• **terminationGracePeriodSeconds** (int64)

Optional duration in seconds the pod needs to terminate gracefully upon probe failure. The grace period is the duration in seconds after the processes running in the pod are sent a termination signal and the time when the processes are forcibly halted with a kill signal. Set this value longer than the expected cleanup time for your process. If this value is nil, the pod's terminationGracePeriodSeconds will be used. Otherwise, this value overrides the value provided by the pod spec. Value must be non-negative integer. The value zero indicates stop immediately via the kill signal (no opportunity to shut down). This is a beta field and requires enabling ProbeTerminationGracePeriod feature gate. Minimum value is 1. spec.terminationGracePeriodSeconds is used if unset.

• **periodSeconds** (int32)

How often (in seconds) to perform the probe. Default to 10 seconds. Minimum value is 1.

• **timeoutSeconds** (int32)

Number of seconds after which the probe times out. Defaults to 1 second. Minimum value is 1. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#container-probes>

• **failureThreshold** (int32)

Minimum consecutive failures for the probe to be considered failed after having succeeded. Defaults to 3. Minimum value is 1.

• **successThreshold** (int32)

Minimum consecutive successes for the probe to be considered successful after having failed. Defaults to 1. Must be 1 for liveness and startup. Minimum value is 1.

• **grpc** (GRPCAction)

GRPC specifies an action involving a GRPC port.

**

◦ **grpc.port** (int32), required

Port number of the gRPC service. Number must be in the range 1 to 65535.

◦ **grpc.service** (string)

Service is the name of the service to place in the gRPC HealthCheckRequest (see <https://github.com/grpc/grpc/blob/master/doc/health-checking.md>).

If this is not specified, the default behavior is defined by gRPC.

PodStatus

PodStatus represents information about the status of a pod. Status may trail the actual state of a system, especially if the node that hosts the pod cannot contact the control plane.

- **nominatedNodeName** (string)

nominatedNodeName is set only when this pod preempts other pods on the node, but it cannot be scheduled right away as preemption victims receive their graceful termination periods. This field does not guarantee that the pod will be scheduled on this node. Scheduler may decide to place the pod elsewhere if other nodes become available sooner. Scheduler may also decide to give the resources on this node to a higher priority pod that is created after preemption. As a result, this field may be different than PodSpec.nodeName when the pod is scheduled.

- **hostIP** (string)

hostIP holds the IP address of the host to which the pod is assigned. Empty if the pod has not started yet. A pod can be assigned to a node that has a problem in kubelet which in turns mean that HostIP will not be updated even if there is a node is assigned to pod

- **hostIPs** ([]HostIP)

Patch strategy: merge on key ip

Atomic: will be replaced during a merge

hostIPs holds the IP addresses allocated to the host. If this field is specified, the first entry must match the hostIP field. This list is empty if the pod has not started yet. A pod can be assigned to a node that has a problem in kubelet which in turns means that HostIPs will not be updated even if there is a node is assigned to this pod.

HostIP represents a single IP address allocated to the host.

- **hostIPs.ip** (string)

IP is the IP address assigned to the host

- **startTime** (Time)

RFC 3339 date and time at which the object was acknowledged by the Kubelet. This is before the Kubelet pulled the container image(s) for the pod.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **phase** (string)

The phase of a Pod is a simple, high-level summary of where the Pod is in its lifecycle. The conditions array, the reason and message fields, and the individual container status arrays contain more detail about the pod's status. There are five possible phase values:

Pending: The pod has been accepted by the Kubernetes system, but one or more of the container images has not been created. This includes time before being scheduled as well as time spent downloading images over the network, which could take a while. Running: The pod has been bound to a node, and all of the containers have been created. At least one container is still running, or is in the process of starting or restarting. Succeeded: All containers in the pod have terminated in success, and will not be restarted. Failed: All containers in the pod have terminated, and at least one container has terminated in failure. The container either exited with non-zero status or was terminated by the system. Unknown: For some reason the state of the pod could not be obtained, typically due to an error in communicating with the host of the pod.

More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-phase>

- **message** (string)

A human readable message indicating details about why the pod is in this condition.

- **reason** (string)

A brief CamelCase message indicating details about why the pod is in this state. e.g. 'Evicted'

- **podIP** (string)

podIP address allocated to the pod. Routable at least within the cluster. Empty if not yet allocated.

- **podIPs** ([]PodIP)

Patch strategy: merge on key ip

podIPs holds the IP addresses allocated to the pod. If this field is specified, the 0th entry must match the podIP field. Pods may be allocated at most 1 value for each of IPv4 and IPv6. This list is empty if no IPs have been allocated yet.

PodIP represents a single IP address allocated to the pod.

- **podIPs.ip** (string)

IP is the IP address assigned to the pod

- **conditions** ([]PodCondition)

Patch strategy: merge on key type

Current service state of pod. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-conditions>

PodCondition contains details for the current condition of this pod.

- **conditions.status** (string), required

Status is the status of the condition. Can be True, False, Unknown. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-conditions>

- **conditions.type** (string), required

Type is the type of the condition. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-conditions>

- **conditions.lastProbeTime** (Time)

Last time we probed the condition.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.lastTransitionTime** (Time)

Last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

Human-readable message indicating details about last transition.

- **conditions.reason** (string)

Unique, one-word, CamelCase reason for the condition's last transition.

- **qosClass** (string)

The Quality of Service (QOS) classification assigned to the pod based on resource requirements See PodQOSClass type for available QOS classes More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-qos/#quality-of-service-classes>

- **initContainerStatuses** ([]ContainerStatus)

The list has one entry per init container in the manifest. The most recent successful init container will have ready = true, the most recently started container will have startTime set. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-and-container-status>

ContainerStatus contains details for the current status of this container.

- **containerStatuses** ([]ContainerStatus)

The list has one entry per container in the manifest. More info: <https://kubernetes.io/docs/concepts/workloads/pods/pod-lifecycle#pod-and-container-status>

ContainerStatus contains details for the current status of this container.

- **ephemeralContainerStatuses** ([]ContainerStatus)

Status for any ephemeral containers that have run in this pod.

ContainerStatus contains details for the current status of this container.

- **resourceClaimStatuses** ([]PodResourceClaimStatus)

Patch strategies: retainKeys, merge on key name

Map: unique values on key name will be kept during a merge

Status of resource claims.

PodResourceClaimStatus is stored in the PodStatus for each PodResourceClaim which references a ResourceClaimTemplate. It stores the generated name for the corresponding ResourceClaim.

- **resourceClaimStatuses.name** (string), required

Name uniquely identifies this resource claim inside the pod. This must match the name of an entry in pod.spec.resourceClaims, which implies that the string must be a DNS_LABEL.

- **resourceClaimStatuses.resourceClaimName** (string)

ResourceClaimName is the name of the ResourceClaim that was generated for the Pod in the namespace of the Pod. If this is unset, then generating a ResourceClaim was not necessary. The pod.spec.resourceClaims entry can be ignored in this case.

- **resize** (string)

Status of resources resize desired for pod's containers. It is empty if no resources resize is pending. Any changes to container resources will automatically set this to "Proposed"

PodList

PodList is a list of Pods.

- **items** ([]Pod), required

List of pods. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md>

- **apiVersion** (string)

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>

- **kind** (string)

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In

CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

Operations

get read the specified Pod

HTTP Request

GET /api/v1/namespaces/{namespace}/pods/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Pod
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Pod](#)): OK

401: Unauthorized

get read ephemeralcontainers of the specified Pod

HTTP Request

GET /api/v1/namespaces/{namespace}/pods/{name}/ephemeralcontainers

Parameters

- **name** (*in path*): string, required
 - name of the Pod
- **namespace** (*in path*): string, required
 - [namespace](#)

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Pod](#)): OK

401: Unauthorized

get read log of the specified Pod

HTTP Request

GET /api/v1/namespaces/{namespace}/pods/{name}/log

Parameters

- **name** (*in path*): string, required

name of the Pod

- **namespace** (*in path*): string, required

[namespace](#)

- **container** (*in query*): string

The container for which to stream logs. Defaults to only container if there is one container in the pod.

- **follow** (*in query*): boolean

Follow the log stream of the pod. Defaults to false.

- **insecureSkipTLSVerifyBackend** (*in query*): boolean

insecureSkipTLSVerifyBackend indicates that the apiserver should not confirm the validity of the serving certificate of the backend it is connecting to. This will make the HTTPS connection between the apiserver and the backend insecure. This means the apiserver cannot verify the log data it is receiving came from the real kubelet. If the kubelet is configured to verify the apiserver's TLS credentials, it does not mean the connection to the real kubelet is vulnerable to a man in the middle attack (e.g. an attacker could not intercept the actual log data coming from the real kubelet).

- **limitBytes** (*in query*): integer

If set, the number of bytes to read from the server before terminating the log output. This may not display a complete final line of logging, and may return slightly more or slightly less than the specified limit.

pretty (*in query*): string

- [pretty](#)

- **previous** (*in query*): boolean

Return previous terminated container logs. Defaults to false.

- **sinceSeconds** (*in query*): integer

A relative time in seconds before the current time from which to show logs. If this value precedes the time a pod was started, only logs since the pod start will be returned. If this value is in the future, no logs will be returned. Only one of sinceSeconds or sinceTime may be specified.

- **tailLines** (*in query*): integer

If set, the number of lines from the end of the logs to show. If not specified, logs are shown from the creation of the container or sinceSeconds or sinceTime

- **timestamps** (*in query*): boolean

If true, add an RFC3339 or RFC3339Nano timestamp at the beginning of every line of log output. Defaults to false.

Response

200 (string): OK

401: Unauthorized

get read status of the specified Pod

HTTP Request

GET /api/v1/namespaces/{namespace}/pods/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Pod

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Pod](#)): OK

401: Unauthorized

list list or watch objects of kind Pod

HTTP Request

GET /api/v1/namespaces/{namespace}/pods

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

timeoutSeconds (*in query*): integer

- [timeoutSeconds](#)

• **watch** (*in query*): boolean

[watch](#)

Response

200 ([PodList](#)): OK

401: Unauthorized

list list or watch objects of kind Pod

HTTP Request

GET /api/v1/pods

Parameters

• **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

• **continue** (*in query*): string

[continue](#)

• **fieldSelector** (*in query*): string

[fieldSelector](#)

• **labelSelector** (*in query*): string

[labelSelector](#)

• **limit** (*in query*): integer

[limit](#)

• **pretty** (*in query*): string

[pretty](#)

• **resourceVersion** (*in query*): string

[resourceVersion](#)

• **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

sendInitialEvents (*in query*): boolean

-

[sendInitialEvents](#)

• **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

• **watch** (*in query*): boolean

[watch](#)

Response

200 ([PodList](#)): OK

401: Unauthorized

create create a Pod

HTTP Request

POST /api/v1/namespaces/{namespace}/pods

Parameters

• **namespace** (*in path*): string, required

[namespace](#)

• **body**: [Pod](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([Pod](#)): OK

201 ([Pod](#)): Created

202 ([Pod](#)): Accepted

401: Unauthorized

update replace the specified Pod

HTTP Request

PUT /api/v1/namespaces/{namespace}/pods/{name}

Parameters

- **name** (*in path*): string, required

name of the Pod

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Pod](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Pod](#)): OK

201 ([Pod](#)): Created

401: Unauthorized

update replace ephemeralcontainers of the specified Pod

HTTP Request

PUT /api/v1/namespaces/{namespace}/pods/{name}/ephemeralcontainers

Parameters

- **name** (*in path*): string, required
name of the Pod
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Pod](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Pod](#)): OK

201 ([Pod](#)): Created

401: Unauthorized

update replace status of the specified Pod

HTTP Request

PUT /api/v1/namespaces/{namespace}/pods/{name}/status

Parameters

- **name** (*in path*): string, required
name of the Pod
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Pod](#), required

- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Pod](#)): OK

201 ([Pod](#)): Created

401: Unauthorized

patch partially update the specified Pod

HTTP Request

PATCH /api/v1/namespaces/{namespace}/pods/{name}

Parameters

- **name** (*in path*): string, required
name of the Pod
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)

- **force** (*in query*): boolean

- [force](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([Pod](#)): OK

201 ([Pod](#)): Created

401: Unauthorized

patch partially update ephemeralcontainers of the specified Pod

HTTP Request

PATCH /api/v1/namespaces/{namespace}/pods/{name}/ephemeralcontainers

Parameters

- **name** (*in path*): string, required

- name of the Pod

- **namespace** (*in path*): string, required

- [namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

- [dryRun](#)

- **fieldManager** (*in query*): string

- [fieldManager](#)

- **fieldValidation** (*in query*): string

- [fieldValidation](#)

- **force** (*in query*): boolean

- [force](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([Pod](#)): OK

201 ([Pod](#)): Created

401: Unauthorized

patch partially update status of the specified Pod

HTTP Request

PATCH /api/v1/namespaces/{namespace}/pods/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the Pod
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Pod](#)): OK

201 ([Pod](#)): Created

401: Unauthorized

delete delete a Pod

HTTP Request

DELETE /api/v1/namespaces/{namespace}/pods/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Pod
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Pod](#)): OK

202 ([Pod](#)): Accepted

401: Unauthorized

deletecollection delete collection of Pod

HTTP Request

DELETE /api/v1/namespaces/{namespace}/pods

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

PodTemplate

PodTemplate describes a template for creating copies of a predefined pod.

```
apiVersion: v1  
import "k8s.io/api/core/v1"
```

PodTemplate

PodTemplate describes a template for creating copies of a predefined pod.

- **apiVersion:** v1
- **kind:** PodTemplate
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **template** ([PodTemplateSpec](#))

Template defines the pods that will be created from this pod template. <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

PodTemplateSpec

PodTemplateSpec describes the data a pod should have when created from a template

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([PodSpec](#))

Specification of the desired behavior of the pod. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

PodTemplateList

PodTemplateList is a list of PodTemplates.

- **apiVersion:** v1
- **kind:** PodTemplateList

metadata ([ListMeta](#))

- Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>
- **items** ([][PodTemplate](#)), required

List of pod templates

Operations

get read the specified PodTemplate

HTTP Request

GET /api/v1/namespaces/{namespace}/podtemplates/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PodTemplate
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PodTemplate](#)): OK

401: Unauthorized

list list or watch objects of kind PodTemplate

HTTP Request

GET /api/v1/namespaces/{namespace}/podtemplates

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)
- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([PodTemplateList](#)): OK

401: Unauthorized

list list or watch objects of kind PodTemplate

HTTP Request

GET /api/v1/podtemplates

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([PodTemplateList](#)): OK

401: Unauthorized

create create a PodTemplate

HTTP Request

POST /api/v1/namespaces/{namespace}/podtemplates

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [PodTemplate](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PodTemplate](#)): OK

201 ([PodTemplate](#)): Created

202 ([PodTemplate](#)): Accepted

401: Unauthorized

update replace the specified PodTemplate

HTTP Request

PUT /api/v1/namespaces/{namespace}/podtemplates/{name}

Parameters

- **name** (*in path*): string, required

name of the PodTemplate

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [PodTemplate](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PodTemplate](#)): OK

201 ([PodTemplate](#)): Created

401: Unauthorized

patch partially update the specified PodTemplate

HTTP Request

PATCH /api/v1/namespaces/{namespace}/podtemplates/{name}

Parameters

- **name** (*in path*): string, required
name of the PodTemplate
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)

- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PodTemplate](#)): OK

201 ([PodTemplate](#)): Created

401: Unauthorized

delete delete a PodTemplate

HTTP Request

DELETE /api/v1/namespaces/{namespace}/podtemplates/{name}

Parameters

- **name** (*in path*): string, required
name of the PodTemplate
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([PodTemplate](#)): OK

202 ([PodTemplate](#)): Accepted

401: Unauthorized

deletecollection delete collection of PodTemplate

HTTP Request

DELETE /api/v1/namespaces/{namespace}/podtemplates

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ReplicationController

ReplicationController represents the configuration of a replication controller.

```
apiVersion: v1  
import "k8s.io/api/core/v1"
```

ReplicationController

ReplicationController represents the configuration of a replication controller.

- **apiVersion**: v1
- **kind**: ReplicationController
- **metadata** ([ObjectMeta](#))

If the Labels of a ReplicationController are empty, they are defaulted to be the same as the Pod(s) that the replication controller manages. Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([ReplicationControllerSpec](#))

Spec defines the specification of the desired behavior of the replication controller. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([ReplicationControllerStatus](#))

Status is the most recently observed status of the replication controller. This data may be out of date by some window of time. Populated by the system. Read-only. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

ReplicationControllerSpec

ReplicationControllerSpec is the specification of a replication controller.

- **selector** (map[string]string)

Selector is a label query over pods that should match the Replicas count. If Selector is empty, it is defaulted to the labels present on the Pod template. Label keys and values that must match in order to be controlled by this replication controller, if empty defaulted to labels on Pod template. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors>

- **template** ([PodTemplateSpec](#))

Template is the object that describes the pod that will be created if insufficient replicas are detected. This takes precedence over a TemplateRef. The only allowed template.spec.restartPolicy value is "Always". More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller#pod-template>

- **replicas** (int32)

Replicas is the number of desired replicas. This is a pointer to distinguish between explicit zero and unspecified. Defaults to 1. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller#what-is-a-replicationcontroller>

- **minReadySeconds** (int32)

Minimum number of seconds for which a newly created pod should be ready without any of its container crashing, for it to be considered available. Defaults to 0 (pod will be considered available as soon as it is ready)

ReplicationControllerStatus

ReplicationControllerStatus represents the current status of a replication controller.

- **replicas** (int32), required

Replicas is the most recently observed number of replicas. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller#what-is-a-replicationcontroller>

- **availableReplicas** (int32)

The number of available replicas (ready for at least minReadySeconds) for this replication controller.

readyReplicas (int32)

- The number of ready replicas for this replication controller.

fullyLabeledReplicas (int32)

The number of pods that have labels matching the labels of the pod template of the replication controller.

conditions ([]ReplicationControllerCondition)

Patch strategy: merge on key type

Represents the latest available observations of a replication controller's current state.

ReplicationControllerCondition describes the state of a replication controller at a certain point.

- **conditions.status** (string), required

Status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

Type of replication controller condition.

- **conditions.lastTransitionTime** (Time)

The last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

A human readable message indicating details about the transition.

- **conditions.reason** (string)

The reason for the condition's last transition.

observedGeneration (int64)

ObservedGeneration reflects the generation of the most recently observed replication controller.

ReplicationControllerList

ReplicationControllerList is a collection of replication controllers.

- **apiVersion**: v1

- **kind**: ReplicationControllerList

metadata ([ListMeta](#))

- Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>
- **items** ([][ReplicationController](#)), required

List of replication controllers. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller>

Operations

get read the specified ReplicationController

HTTP Request

GET /api/v1/namespaces/{namespace}/replicationcontrollers/{name}

Parameters

- **name** (*in path*): string, required
name of the ReplicationController
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ReplicationController](#)): OK

401: Unauthorized

get read status of the specified ReplicationController

HTTP Request

GET /api/v1/namespaces/{namespace}/replicationcontrollers/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ReplicationController

namespace (*in path*): string, required

- [namespace](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ReplicationController](#)): OK

401: Unauthorized

list list or watch objects of kind ReplicationController

HTTP Request

GET /api/v1/namespaces/{namespace}/replicationcontrollers

Parameters

• **namespace** (*in path*): string, required

[namespace](#)

• **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

• **continue** (*in query*): string

[continue](#)

• **fieldSelector** (*in query*): string

[fieldSelector](#)

• **labelSelector** (*in query*): string

[labelSelector](#)

• **limit** (*in query*): integer

[limit](#)

• **pretty** (*in query*): string

[pretty](#)

• **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ReplicationControllerList](#)): OK

401: Unauthorized

list list or watch objects of kind ReplicationController

HTTP Request

GET /api/v1/replicationcontrollers

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ReplicationControllerList](#)): OK

401: Unauthorized

create create a ReplicationController

HTTP Request

POST /api/v1/namespaces/{namespace}/replicationcontrollers

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [ReplicationController](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ReplicationController](#)): OK

201 ([ReplicationController](#)): Created

202 ([ReplicationController](#)): Accepted

401: Unauthorized

update replace the specified ReplicationController

HTTP Request

PUT /api/v1/namespaces/{namespace}/replicationcontrollers/{name}

Parameters

- **name** (*in path*): string, required

name of the ReplicationController

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ReplicationController](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ReplicationController](#)): OK

201 ([ReplicationController](#)): Created

401: Unauthorized

update replace status of the specified ReplicationController

HTTP Request

PUT /api/v1/namespaces/{namespace}/replicationcontrollers/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ReplicationController
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [ReplicationController](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ReplicationController](#)): OK

201 ([ReplicationController](#)): Created

401: Unauthorized

patch partially update the specified ReplicationController

HTTP Request

PATCH /api/v1/namespaces/{namespace}/replicationcontrollers/{name}

Parameters

- **name** (*in path*): string, required
name of the ReplicationController

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ReplicationController](#)): OK

201 ([ReplicationController](#)): Created

401: Unauthorized

patch partially update status of the specified ReplicationController

HTTP Request

PATCH /api/v1/namespaces/{namespace}/replicationcontrollers/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the ReplicationController
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ReplicationController](#)): OK

201 ([ReplicationController](#)): Created

401: Unauthorized

delete delete a ReplicationController

HTTP Request

DELETE /api/v1/namespaces/{namespace}/replicationcontrollers/{name}

Parameters

- **name** (*in path*): string, required
name of the ReplicationController
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ReplicationController

HTTP Request

DELETE /api/v1/namespaces/{namespace}/replicationcontrollers

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ReplicaSet

ReplicaSet ensures that a specified number of pod replicas are running at any given time.

apiVersion: apps/v1

```
import "k8s.io/api/apps/v1"
```

ReplicaSet

ReplicaSet ensures that a specified number of pod replicas are running at any given time.

- **apiVersion**: apps/v1

- **kind**: ReplicaSet

- **metadata** ([ObjectMeta](#))

If the Labels of a ReplicaSet are empty, they are defaulted to be the same as the Pod(s) that the ReplicaSet manages. Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([ReplicaSetSpec](#))

Spec defines the specification of the desired behavior of the ReplicaSet. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([ReplicasetStatus](#))

Status is the most recently observed status of the ReplicaSet. This data may be out of date by some window of time. Populated by the system. Read-only. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

ReplicaSetSpec

ReplicaSetSpec is the specification of a ReplicaSet.

- **selector** ([LabelSelector](#)), required

Selector is a label query over pods that should match the replica count. Label keys and values that must match in order to be controlled by this replica set. It must match the pod template's labels. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors>

- **template** ([PodTemplateSpec](#))

Template is the object that describes the pod that will be created if insufficient replicas are detected. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller#pod-template>

- **replicas** (int32)

Replicas is the number of desired replicas. This is a pointer to distinguish between explicit zero and unspecified. Defaults to 1. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/#what-is-a-replicationcontroller>

- **minReadySeconds** (int32)

Minimum number of seconds for which a newly created pod should be ready without any of its container crashing, for it to be considered available. Defaults to 0 (pod will be considered available as soon as it is ready)

ReplicasetStatus

ReplicasetStatus represents the current status of a ReplicaSet.

- **replicas** (int32), required

Replicas is the most recently observed number of replicas. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller/#what-is-a-replicationcontroller>

- **availableReplicas** (int32)

The number of available replicas (ready for at least minReadySeconds) for this replica set.

- **readyReplicas** (int32)

readyReplicas is the number of pods targeted by this ReplicaSet with a Ready Condition.

- **fullyLabeledReplicas** (int32)

The number of pods that have labels matching the labels of the pod template of the replicaset.

- **conditions** ([]ReplicaSetCondition)

Patch strategy: merge on key type

Represents the latest available observations of a replica set's current state.

ReplicaSetCondition describes the state of a replica set at a certain point.

- **conditions.status** (string), required

Status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

Type of replica set condition.

- **conditions.lastTransitionTime** (Time)

The last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

A human readable message indicating details about the transition.

- **conditions.reason** (string)

The reason for the condition's last transition.

- **observedGeneration** (int64)

ObservedGeneration reflects the generation of the most recently observed ReplicaSet.

ReplicaSetList

ReplicaSetList is a collection of ReplicaSets.

- **apiVersion**: apps/v1

- **kind**: ReplicaSetList

metadata ([ListMeta](#))

- Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>
- **items** ([][ReplicaSet](#)), required

List of ReplicaSets. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller>

Operations

get read the specified ReplicaSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/replicasets/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ReplicaSet
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ReplicaSet](#)): OK

401: Unauthorized

get read status of the specified ReplicaSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/replicasets/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the ReplicaSet

namespace (*in path*): string, required

- [namespace](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ReplicaSet](#)): OK

401: Unauthorized

list list or watch objects of kind ReplicaSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/replicasets

Parameters

• **namespace** (*in path*): string, required

[namespace](#)

• **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

• **continue** (*in query*): string

[continue](#)

• **fieldSelector** (*in query*): string

[fieldSelector](#)

• **labelSelector** (*in query*): string

[labelSelector](#)

• **limit** (*in query*): integer

[limit](#)

• **pretty** (*in query*): string

[pretty](#)

• **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ReplicaSetList](#)): OK

401: Unauthorized

list list or watch objects of kind ReplicaSet

HTTP Request

GET /apis/apps/v1/replicasets

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ReplicaSetList](#)): OK

401: Unauthorized

create create a ReplicaSet

HTTP Request

POST /apis/apps/v1/namespaces/{namespace}/replicasets

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [ReplicaSet](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ReplicaSet](#)): OK

201 ([ReplicaSet](#)): Created

202 ([ReplicaSet](#)): Accepted

401: Unauthorized

update replace the specified ReplicaSet

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/replicasets/{name}

Parameters

- **name** (*in path*): string, required

name of the ReplicaSet

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ReplicaSet](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ReplicaSet](#)): OK

201 ([ReplicaSet](#)): Created

401: Unauthorized

update replace status of the specified ReplicaSet

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/replicasets/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ReplicaSet
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [ReplicaSet](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ReplicaSet](#)): OK

201 ([ReplicaSet](#)): Created

401: Unauthorized

patch partially update the specified ReplicaSet

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/replicasets/{name}

Parameters

- **name** (*in path*): string, required
name of the ReplicaSet

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ReplicaSet](#)): OK

201 ([ReplicaSet](#)): Created

401: Unauthorized

patch partially update status of the specified ReplicaSet

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/replicasets/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the ReplicaSet
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ReplicaSet](#)): OK

201 ([ReplicaSet](#)): Created

401: Unauthorized

delete delete a ReplicaSet

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/replicasets/{name}

Parameters

- **name** (*in path*): string, required

name of the ReplicaSet

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ReplicaSet

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/replicasets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Deployment

Deployment enables declarative updates for Pods and ReplicaSets.

apiVersion: apps/v1

import "k8s.io/api/apps/v1"

Deployment

Deployment enables declarative updates for Pods and ReplicaSets.

- **apiVersion**: apps/v1

- **kind**: Deployment

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([DeploymentSpec](#))

Specification of the desired behavior of the Deployment.

status ([DeploymentStatus](#))

- Most recently observed status of the Deployment.

DeploymentSpec

DeploymentSpec is the specification of the desired behavior of the Deployment.

• **selector** ([LabelSelector](#)), required

Label selector for pods. Existing ReplicaSets whose pods are selected by this will be the ones affected by this deployment. It must match the pod template's labels.

• **template** ([PodTemplateSpec](#)), required

Template describes the pods that will be created. The only allowed template.spec.restartPolicy value is "Always".

• **replicas** (int32)

Number of desired pods. This is a pointer to distinguish between explicit zero and not specified. Defaults to 1.

• **minReadySeconds** (int32)

Minimum number of seconds for which a newly created pod should be ready without any of its container crashing, for it to be considered available. Defaults to 0 (pod will be considered available as soon as it is ready)

• **strategy** (DeploymentStrategy)

Patch strategy: retainKeys

The deployment strategy to use to replace existing pods with new ones.

DeploymentStrategy describes how to replace existing pods with new ones.

◦ **strategy.type** (string)

Type of deployment. Can be "Recreate" or "RollingUpdate". Default is RollingUpdate.

◦ **strategy.rollingUpdate** (RollingUpdateDeployment)

Rolling update config params. Present only if DeploymentStrategyType = RollingUpdate.

Spec to control the desired behavior of rolling update.

▪ **strategy.rollingUpdate.maxSurge** (IntOrString)

The maximum number of pods that can be scheduled above the desired number of pods. Value can be an absolute number (ex: 5) or a percentage of desired pods (ex: 10%). This can not be 0 if MaxUnavailable is 0. Absolute

number is calculated from percentage by rounding up. Defaults to 25%. Example: when this is set to 30%, the new ReplicaSet can be scaled up immediately when the rolling update starts, such that the total number of old and new pods do not exceed 130% of desired pods. Once old pods have been killed, new ReplicaSet can be scaled up further, ensuring that total number of pods running at any time during the update is at most 130% of desired pods.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **strategy.rollingUpdate.maxUnavailable** (IntOrString)

The maximum number of pods that can be unavailable during the update. Value can be an absolute number (ex: 5) or a percentage of desired pods (ex: 10%). Absolute number is calculated from percentage by rounding down. This can not be 0 if MaxSurge is 0. Defaults to 25%. Example: when this is set to 30%, the old ReplicaSet can be scaled down to 70% of desired pods immediately when the rolling update starts. Once new pods are ready, old ReplicaSet can be scaled down further, followed by scaling up the new ReplicaSet, ensuring that the total number of pods available at all times during the update is at least 70% of desired pods.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **revisionHistoryLimit** (int32)

The number of old ReplicaSets to retain to allow rollback. This is a pointer to distinguish between explicit zero and not specified. Defaults to 10.

- **progressDeadlineSeconds** (int32)

The maximum time in seconds for a deployment to make progress before it is considered to be failed. The deployment controller will continue to process failed deployments and a condition with a ProgressDeadlineExceeded reason will be surfaced in the deployment status. Note that progress will not be estimated during the time a deployment is paused. Defaults to 600s.

- **paused** (boolean)

Indicates that the deployment is paused.

DeploymentStatus

DeploymentStatus is the most recently observed status of the Deployment.

- **replicas** (int32)

Total number of non-terminated pods targeted by this deployment (their labels match the selector).

- **availableReplicas** (int32)

Total number of available pods (ready for at least minReadySeconds) targeted by this deployment.

- **readyReplicas** (int32)

readyReplicas is the number of pods targeted by this Deployment with a Ready Condition.

- **unavailableReplicas** (int32)

Total number of unavailable pods targeted by this deployment. This is the total number of pods that are still required for the deployment to have 100% available capacity. They may either be pods that are running but not yet available or pods that still have not been created.

- **updatedReplicas** (int32)

Total number of non-terminated pods targeted by this deployment that have the desired template spec.

- **collisionCount** (int32)

Count of hash collisions for the Deployment. The Deployment controller uses this field as a collision avoidance mechanism when it needs to create the name for the newest ReplicaSet.

- **conditions** ([]DeploymentCondition)

Patch strategy: merge on key type

Represents the latest available observations of a deployment's current state.

DeploymentCondition describes the state of a deployment at a certain point.

- **conditions.status** (string), required

Status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

Type of deployment condition.

- **conditions.lastTransitionTime** (Time)

Last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.lastUpdateTime** (Time)

The last time this condition was updated.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

A human readable message indicating details about the transition.

- **conditions.reason** (string)

The reason for the condition's last transition.

- **observedGeneration** (int64)

The generation observed by the deployment controller.

DeploymentList

DeploymentList is a list of Deployments.

- **apiVersion**: apps/v1

- **kind**: DeploymentList

- **metadata** ([ListMeta](#))

Standard list metadata.

- **items** ([][Deployment](#)), required

Items is the list of Deployments.

Operations

get read the specified Deployment

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/deployments/{name}

Parameters

- **name** (*in path*): string, required

name of the Deployment

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Deployment](#)): OK

401: Unauthorized

get read status of the specified Deployment

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/deployments/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Deployment

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Deployment](#)): OK

401: Unauthorized

list list or watch objects of kind Deployment

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/deployments

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([DeploymentList](#)): OK

401: Unauthorized

list list or watch objects of kind Deployment

HTTP Request

GET /apis/apps/v1/deployments

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([DeploymentList](#)): OK

401: Unauthorized

create create a Deployment

HTTP Request

POST /apis/apps/v1/namespaces/{namespace}/deployments

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Deployment](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Deployment](#)): OK

201 ([Deployment](#)): Created

202 ([Deployment](#)): Accepted

401: Unauthorized

update replace the specified Deployment

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/deployments/{name}

Parameters

- **name** (*in path*): string, required

name of the Deployment

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Deployment](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Deployment](#)): OK

201 ([Deployment](#)): Created

401: Unauthorized

update replace status of the specified Deployment

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/deployments/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Deployment

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Deployment](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

fieldValidation (*in query*): string

- [fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([Deployment](#)): OK

201 ([Deployment](#)): Created

401: Unauthorized

patch partially update the specified Deployment

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/deployments/{name}

Parameters

• **name** (*in path*): string, required

name of the Deployment

• **namespace** (*in path*): string, required

[namespace](#)

• **body**: [Patch](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **force** (*in query*): boolean

[force](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([Deployment](#)): OK

201 ([Deployment](#)): Created

401: Unauthorized

patch partially update status of the specified Deployment

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/deployments/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the Deployment
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Deployment](#)): OK

201 ([Deployment](#)): Created

401: Unauthorized

delete delete a Deployment

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/deployments/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Deployment
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Deployment

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/deployments

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

StatefulSet

StatefulSet represents a set of pods with consistent identities.

```
apiVersion: apps/v1  
import "k8s.io/api/apps/v1"
```

StatefulSet

StatefulSet represents a set of pods with consistent identities. Identities are defined as:

- Network: A single stable DNS and hostname.
- Storage: As many VolumeClaims as requested.

The StatefulSet guarantees that a given network identity will always map to the same storage identity.

- **apiVersion**: apps/v1
- **kind**: StatefulSet
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([StatefulSetSpec](#))

Spec defines the desired identities of pods in this set.

- **status** ([StatefulSetStatus](#))

Status is the current status of Pods in this StatefulSet. This data may be out of date by some window of time.

StatefulSetSpec

A StatefulSetSpec is the specification of a StatefulSet.

- **serviceName** (string), required

serviceName is the name of the service that governs this StatefulSet. This service must exist before the StatefulSet, and is responsible for the network identity of the set. Pods get DNS/hostnames that follow the pattern: pod-specific-string.serviceName.default.svc.cluster.local where "pod-specific-string" is managed by the StatefulSet controller.

- **selector** ([LabelSelector](#)), required

selector is a label query over pods that should match the replica count. It must match the pod template's labels. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors>

- **template** ([PodTemplateSpec](#)), required

template is the object that describes the pod that will be created if insufficient replicas are detected. Each pod stamped out by the StatefulSet will fulfill this Template, but have a unique identity from the rest of the StatefulSet. Each pod will be named with the format <statefulsetname>-<podindex>. For example, a pod in a StatefulSet named "web" with index number "3" would be named "web-3". The only allowed template.spec.restartPolicy value is "Always".

- **replicas** (int32)

replicas is the desired number of replicas of the given Template. These are replicas in the sense that they are instantiations of the same Template, but individual replicas also have a consistent identity. If unspecified, defaults to 1.

- **updateStrategy** ([StatefulSetUpdateStrategy](#))

updateStrategy indicates the StatefulSetUpdateStrategy that will be employed to update Pods in the StatefulSet when a revision is made to Template.

StatefulSetUpdateStrategy indicates the strategy that the StatefulSet controller will use to perform updates. It includes any additional parameters necessary to perform the update for the indicated strategy.

- **updateStrategy.type** (string)

Type indicates the type of the StatefulSetUpdateStrategy. Default is RollingUpdate.

- **updateStrategy.rollingUpdate** ([RollingUpdateStatefulSetStrategy](#))

RollingUpdate is used to communicate parameters when Type is RollingUpdateStatefulSetStrategyType.

RollingUpdateStatefulSetStrategy is used to communicate parameter for RollingUpdateStatefulSetStrategyType.

- **updateStrategy.rollingUpdate.maxUnavailable** ([IntOrString](#))

The maximum number of pods that can be unavailable during the update. Value can be an absolute number (ex: 5) or a percentage of desired pods (ex: 10%). Absolute number is calculated from percentage by rounding up. This can not be 0. Defaults to 1. This field is alpha-level and is only honored by servers that enable the MaxUnavailableStatefulSet feature. The field applies to all pods in the range 0 to Replicas-1. That means if there is any unavailable pod in the range 0 to Replicas-1, it will be counted towards MaxUnavailable.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

updateStrategy.rollingUpdate.partition (int32)

Partition indicates the ordinal at which the StatefulSet should be partitioned for updates. During a rolling update, all pods from ordinal Replicas-1 to Partition are updated. All pods from ordinal Partition-1 to 0 remain untouched. This is helpful in being able to do a canary based deployment. The default value is 0.

• **podManagementPolicy** (string)

podManagementPolicy controls how pods are created during initial scale up, when replacing pods on nodes, or when scaling down. The default policy is OrderedReady, where pods are created in increasing order (pod-0, then pod-1, etc) and the controller will wait until each pod is ready before continuing. When scaling down, the pods are removed in the opposite order. The alternative policy is Parallel which will create pods in parallel to match the desired scale without waiting, and on scale down will delete all pods at once.

• **revisionHistoryLimit** (int32)

revisionHistoryLimit is the maximum number of revisions that will be maintained in the StatefulSet's revision history. The revision history consists of all revisions not represented by a currently applied StatefulSetSpec version. The default value is 10.

• **volumeClaimTemplates** ([][PersistentVolumeClaim](#))

volumeClaimTemplates is a list of claims that pods are allowed to reference. The StatefulSet controller is responsible for mapping network identities to claims in a way that maintains the identity of a pod. Every claim in this list must have at least one matching (by name) volumeMount in one container in the template. A claim in this list takes precedence over any volumes in the template, with the same name.

• **minReadySeconds** (int32)

Minimum number of seconds for which a newly created pod should be ready without any of its container crashing for it to be considered available. Defaults to 0 (pod will be considered available as soon as it is ready)

• **persistentVolumeClaimRetentionPolicy**

(StatefulSetPersistentVolumeClaimRetentionPolicy)

persistentVolumeClaimRetentionPolicy describes the lifecycle of persistent volume claims created from volumeClaimTemplates. By default, all persistent volume claims are created as needed and retained until manually deleted. This policy allows the lifecycle to be altered, for example by deleting persistent volume claims when their stateful set is deleted, or when their pod is scaled down. This requires the StatefulSetAutoDeletePVC feature gate to be enabled, which is alpha. +optional

StatefulSetPersistentVolumeClaimRetentionPolicy describes the policy used for PVCs created from the StatefulSet VolumeClaimTemplates.

◦ **persistentVolumeClaimRetentionPolicy.whenDeleted** (string)

WhenDeleted specifies what happens to PVCs created from StatefulSet VolumeClaimTemplates when the StatefulSet is deleted. The default policy of Retain

causes PVCs to not be affected by StatefulSet deletion. The Delete policy causes those PVCs to be deleted.

- **persistentVolumeClaimRetentionPolicy.whenScaled** (string)

WhenScaled specifies what happens to PVCs created from StatefulSet VolumeClaimTemplates when the StatefulSet is scaled down. The default policy of Retain causes PVCs to not be affected by a scaledown. The Delete policy causes the associated PVCs for any excess pods above the replica count to be deleted.

- **ordinals** (StatefulSetOrdinals)

ordinals controls the numbering of replica indices in a StatefulSet. The default ordinals behavior assigns a "0" index to the first replica and increments the index by one for each additional replica requested. Using the ordinals field requires the StatefulSetStartOrdinal feature gate to be enabled, which is beta.

StatefulSetOrdinals describes the policy used for replica ordinal assignment in this StatefulSet.

- **ordinals.start** (int32)

start is the number representing the first replica's index. It may be used to number replicas from an alternate index (eg: 1-indexed) over the default 0-indexed names, or to orchestrate progressive movement of replicas from one StatefulSet to another. If set, replica indices will be in the range: [.spec.ordinals.start, .spec.ordinals.start + .spec.replicas]. If unset, defaults to 0. Replica indices will be in the range: [0, .spec.replicas).

StatefulSetStatus

StatefulSetStatus represents the current state of a StatefulSet.

- **replicas** (int32), required

replicas is the number of Pods created by the StatefulSet controller.

- **readyReplicas** (int32)

readyReplicas is the number of pods created for this StatefulSet with a Ready Condition.

- **currentReplicas** (int32)

currentReplicas is the number of Pods created by the StatefulSet controller from the StatefulSet version indicated by currentRevision.

- **updatedReplicas** (int32)

updatedReplicas is the number of Pods created by the StatefulSet controller from the StatefulSet version indicated by updateRevision.

- **availableReplicas** (int32)

Total number of available pods (ready for at least minReadySeconds) targeted by this statefulset.

- **collisionCount** (int32)

collisionCount is the count of hash collisions for the StatefulSet. The StatefulSet controller uses this field as a collision avoidance mechanism when it needs to create the name for the newest ControllerRevision.

- **conditions** ([]StatefulSetCondition)

Patch strategy: merge on key type

Represents the latest available observations of a statefulset's current state.

StatefulSetCondition describes the state of a statefulset at a certain point.

- **conditions.status** (string), required

Status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

Type of statefulset condition.

- **conditions.lastTransitionTime** (Time)

Last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

A human readable message indicating details about the transition.

- **conditions.reason** (string)

The reason for the condition's last transition.

- **currentRevision** (string)

currentRevision, if not empty, indicates the version of the StatefulSet used to generate Pods in the sequence [0,currentReplicas).

- **updateRevision** (string)

updateRevision, if not empty, indicates the version of the StatefulSet used to generate Pods in the sequence [replicas-updatedReplicas,replicas)

- **observedGeneration** (int64)

observedGeneration is the most recent generation observed for this StatefulSet. It corresponds to the StatefulSet's generation, which is updated on mutation by the API Server.

StatefulSetList

StatefulSetList is a collection of StatefulSets.

- **apiVersion**: apps/v1

- **kind**: StatefulSetList

- **metadata** ([ListMeta](#))

Standard list's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][StatefulSet](#)), required

Items is the list of stateful sets.

Operations

get read the specified StatefulSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/statefulsets/{name}

Parameters

- **name** (*in path*): string, required

name of the StatefulSet

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([StatefulSet](#)): OK

401: Unauthorized

get read status of the specified StatefulSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/statefulsets/{name}/status

Parameters

- **name** (*in path*): string, required
name of the StatefulSet
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([StatefulSet](#)): OK

401: Unauthorized

list list or watch objects of kind StatefulSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/statefulsets

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)

- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([StatefulSetList](#)): OK

401: Unauthorized

list list or watch objects of kind StatefulSet

HTTP Request

GET /apis/apps/v1/statefulsets

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)

- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)
- **watch** (*in query*): boolean
 - [watch](#)

Response

200 ([StatefulSetList](#)): OK

401: Unauthorized

create create a StatefulSet

HTTP Request

POST /apis/apps/v1/namespaces/{namespace}/statefulsets

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [StatefulSet](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([StatefulSet](#)): OK

201 ([StatefulSet](#)): Created

202 ([StatefulSet](#)): Accepted

401: Unauthorized

update replace the specified StatefulSet

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/statefulsets/{name}

Parameters

- **name** (*in path*): string, required

name of the StatefulSet

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [StatefulSet](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([StatefulSet](#)): OK

201 ([StatefulSet](#)): Created

401: Unauthorized

update replace status of the specified StatefulSet

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/statefulsets/{name}/status

Parameters

- **name** (*in path*): string, required

- name of the StatefulSet

- **namespace** (*in path*): string, required

- [namespace](#)

- **body**: [StatefulSet](#), required

- **dryRun** (*in query*): string

- [dryRun](#)

- **fieldManager** (*in query*): string

- [fieldManager](#)

- **fieldValidation** (*in query*): string

- [fieldValidation](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([StatefulSet](#)): OK

201 ([StatefulSet](#)): Created

401: Unauthorized

patch partially update the specified StatefulSet

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/statefulsets/{name}

Parameters

- **name** (*in path*): string, required
 - name of the StatefulSet
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([StatefulSet](#)): OK

201 ([StatefulSet](#)): Created

401: Unauthorized

patch partially update status of the specified StatefulSet

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/statefulsets/{name}/status

Parameters

- **name** (*in path*): string, required
name of the StatefulSet
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([StatefulSet](#)): OK

201 ([StatefulSet](#)): Created

401: Unauthorized

delete delete a StatefulSet

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/statefulsets/{name}

Parameters

- **name** (*in path*): string, required
name of the StatefulSet
- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of StatefulSet

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/statefulsets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ControllerRevision

ControllerRevision implements an immutable snapshot of state data.

```
apiVersion: apps/v1
```

```
import "k8s.io/api/apps/v1"
```

ControllerRevision

ControllerRevision implements an immutable snapshot of state data. Clients are responsible for serializing and deserializing the objects that contain their internal state. Once a ControllerRevision has been successfully created, it can not be updated. The API Server will fail validation of all requests that attempt to mutate the Data field. ControllerRevisions may, however, be deleted. Note that, due to its use by both the DaemonSet and StatefulSet controllers for update and rollback, this object is beta. However, it may be subject to name and representation changes in future releases, and clients should not depend on its stability. It is primarily for internal use by controllers.

- **apiVersion:** apps/v1

- **kind:** ControllerRevision

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **revision** (int64), required

Revision indicates the revision of the state represented by Data.

- **data** (RawExtension)

Data is the serialized representation of the state.

*RawExtension is used to hold extensions in external versions.

To use this, make a field which has RawExtension as its type in your external, versioned struct, and Object in your internal struct. You also need to register your various plugin types.

// Internal package:

```
type MyAPIObject struct { runtime.TypeMeta json:",inline" MyPlugin runtime.Object  
json:"myPlugin" }
```

```
type PluginA struct { AOption string json:"aOption" }
```

// External package:

```
type MyAPIObject struct { runtime.TypeMeta json:",inline" MyPlugin  
runtime.RawExtension json:"myPlugin" }
```

```
type PluginA struct { AOption string json:"aOption" }
```

// On the wire, the JSON will look something like this:

```
{ "kind":"MyAPIObject", "apiVersion":"v1", "myPlugin": { "kind":"PluginA", "aOption":"foo",  
}, }
```

So what happens? Decode first uses json or yaml to unmarshal the serialized data into your external MyAPIObject. That causes the raw JSON to be stored, but not unpacked. The next step is to copy (using pkg/conversion) into the internal struct. The runtime package's DefaultScheme has conversion functions installed which will unpack the JSON stored in RawExtension, turning it into the correct object type, and storing it in the Object. (TODO: In the case where the object is of an unknown type, a runtime.Unknown object will be created and stored.)*

ControllerRevisionList

ControllerRevisionList is a resource containing a list of ControllerRevision objects.

- **apiVersion**: apps/v1
- **kind**: ControllerRevisionList
- **metadata** ([ListMeta](#))

More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][ControllerRevision](#)), required

Items is the list of ControllerRevisions

Operations

get read the specified ControllerRevision

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/controllerrevisions/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ControllerRevision
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ControllerRevision](#)): OK

401: Unauthorized

list list or watch objects of kind ControllerRevision

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/controllerrevisions

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ControllerRevisionList](#)): OK

401: Unauthorized

list list or watch objects of kind ControllerRevision

HTTP Request

GET /apis/apps/v1/controllerrevisions

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ControllerRevisionList](#)): OK

401: Unauthorized

create create a ControllerRevision

HTTP Request

POST /apis/apps/v1/namespaces/{namespace}/controllerrevisions

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ControllerRevision](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ControllerRevision](#)): OK

201 ([ControllerRevision](#)): Created

202 ([ControllerRevision](#)): Accepted

401: Unauthorized

update replace the specified ControllerRevision

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/controllerrevisions/{name}

Parameters

- **name** (*in path*): string, required
name of the ControllerRevision
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [ControllerRevision](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ControllerRevision](#)): OK

201 ([ControllerRevision](#)): Created

401: Unauthorized

patch partially update the specified ControllerRevision

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/controllerrevisions/{name}

Parameters

- **name** (*in path*): string, required
name of the ControllerRevision

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ControllerRevision](#)): OK

201 ([ControllerRevision](#)): Created

401: Unauthorized

delete delete a ControllerRevision

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/controllerrevisions/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ControllerRevision
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ControllerRevision

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/controllerrevisions

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

DaemonSet

DaemonSet represents the configuration of a daemon set.

apiVersion: apps/v1

import "k8s.io/api/apps/v1"

DaemonSet

DaemonSet represents the configuration of a daemon set.

-
- **apiVersion**: apps/v1

- **kind**: DaemonSet

metadata ([ObjectMeta](#))

- Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

• spec ([DaemonSetSpec](#))

The desired behavior of this daemon set. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

• status ([DaemonSetStatus](#))

The current status of this daemon set. This data may be out of date by some window of time. Populated by the system. Read-only. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

DaemonSetSpec

DaemonSetSpec is the specification of a daemon set.

• selector ([LabelSelector](#)), required

A label query over pods that are managed by the daemon set. Must match in order to be controlled. It must match the pod template's labels. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors>

• template ([PodTemplateSpec](#)), required

An object that describes the pod that will be created. The DaemonSet will create exactly one copy of this pod on every node that matches the template's node selector (or on every node if no node selector is specified). The only allowed template.spec.restartPolicy value is "Always". More info: <https://kubernetes.io/docs/concepts/workloads/controllers/replicationcontroller#pod-template>

• minReadySeconds (int32)

The minimum number of seconds for which a newly created DaemonSet pod should be ready without any of its container crashing, for it to be considered available. Defaults to 0 (pod will be considered available as soon as it is ready).

• updateStrategy ([DaemonSetUpdateStrategy](#))

An update strategy to replace existing DaemonSet pods with new pods.

DaemonSetUpdateStrategy is a struct used to control the update strategy for a DaemonSet.

◦ updateStrategy.type (string)

Type of daemon set update. Can be "RollingUpdate" or "OnDelete". Default is RollingUpdate.

◦ updateStrategy.rollingUpdate ([RollingUpdateDaemonSet](#))

Rolling update config params. Present only if type = "RollingUpdate".

Spec to control the desired behavior of daemon set rolling update.

- **updateStrategy.rollingUpdate.maxSurge** (IntOrString)

The maximum number of nodes with an existing available DaemonSet pod that can have an updated DaemonSet pod during an update. Value can be an absolute number (ex: 5) or a percentage of desired pods (ex: 10%). This can not be 0 if MaxUnavailable is 0. Absolute number is calculated from percentage by rounding up to a minimum of 1. Default value is 0. Example: when this is set to 30%, at most 30% of the total number of nodes that should be running the daemon pod (i.e. status.desiredNumberScheduled) can have their a new pod created before the old pod is marked as deleted. The update starts by launching new pods on 30% of nodes. Once an updated pod is available (Ready for at least minReadySeconds) the old DaemonSet pod on that node is marked deleted. If the old pod becomes unavailable for any reason (Ready transitions to false, is evicted, or is drained) an updated pod is immediately created on that node without considering surge limits.

Allowing surge implies the possibility that the resources consumed by the daemonset on any given node can double if the readiness check fails, and so resource intensive daemonsets should take into account that they may cause evictions during disruption.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **updateStrategy.rollingUpdate.maxUnavailable** (IntOrString)

The maximum number of DaemonSet pods that can be unavailable during the update. Value can be an absolute number (ex: 5) or a percentage of total number of DaemonSet pods at the start of the update (ex: 10%). Absolute number is calculated from percentage by rounding up. This cannot be 0 if MaxSurge is 0. Default value is 1. Example: when this is set to 30%, at most 30% of the total number of nodes that should be running the daemon pod (i.e. status.desiredNumberScheduled) can have their pods stopped for an update at any given time. The update starts by stopping at most 30% of those DaemonSet pods and then brings up new DaemonSet pods in their place. Once the new pods are available, it then proceeds onto other DaemonSet pods, thus ensuring that at least 70% of original number of DaemonSet pods are available at all times during the update.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **revisionHistoryLimit** (int32)

The number of old history to retain to allow rollback. This is a pointer to distinguish between explicit zero and not specified. Defaults to 10.

DaemonSetStatus

DaemonSetStatus represents the current status of a daemon set.

- **numberReady** (int32), required

numberReady is the number of nodes that should be running the daemon pod and have one or more of the daemon pod running with a Ready Condition.

- **numberAvailable** (int32)

The number of nodes that should be running the daemon pod and have one or more of the daemon pod running and available (ready for at least spec.minReadySeconds)

- **numberUnavailable** (int32)

The number of nodes that should be running the daemon pod and have none of the daemon pod running and available (ready for at least spec.minReadySeconds)

- **numberMisscheduled** (int32), required

The number of nodes that are running the daemon pod, but are not supposed to run the daemon pod. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>

- **desiredNumberScheduled** (int32), required

The total number of nodes that should be running the daemon pod (including nodes correctly running the daemon pod). More info: <https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>

- **currentNumberScheduled** (int32), required

The number of nodes that are running at least 1 daemon pod and are supposed to run the daemon pod. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/>

- **updatedNumberScheduled** (int32)

The total number of nodes that are running updated daemon pod

- **collisionCount** (int32)

Count of hash collisions for the DaemonSet. The DaemonSet controller uses this field as a collision avoidance mechanism when it needs to create the name for the newest ControllerRevision.

- **conditions** ([]DaemonSetCondition)

Patch strategy: merge on key type

Represents the latest available observations of a DaemonSet's current state.

DaemonSetCondition describes the state of a DaemonSet at a certain point.

- **conditions.status** (string), required
 - Status of the condition, one of True, False, Unknown.
- **conditions.type** (string), required
 - Type of DaemonSet condition.
- **conditions.lastTransitionTime** (Time)
 - Last time the condition transitioned from one status to another.
Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.
- **conditions.message** (string)
 - A human readable message indicating details about the transition.
- **conditions.reason** (string)
 - The reason for the condition's last transition.

- **observedGeneration** (int64)

The most recent generation observed by the daemon set controller.

DaemonSetList

DaemonSetList is a collection of daemon sets.

- **apiVersion**: apps/v1
- **kind**: DaemonSetList
- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][DaemonSet](#)), required

A list of daemon sets.

Operations

get read the specified DaemonSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/daemonsets/{name}

Parameters

- **name** (*in path*): string, required
name of the DaemonSet
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([DaemonSet](#)): OK

401: Unauthorized

get read status of the specified DaemonSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/daemonsets/{name}/status

Parameters

- **name** (*in path*): string, required
name of the DaemonSet
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([DaemonSet](#)): OK

401: Unauthorized

list list or watch objects of kind DaemonSet

HTTP Request

GET /apis/apps/v1/namespaces/{namespace}/daemonsets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([DaemonSetList](#)): OK

401: Unauthorized

list list or watch objects of kind DaemonSet

HTTP Request

GET /apis/apps/v1/daemonsets

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

watch (*in query*): boolean

- [watch](#)

Response

200 ([DaemonSetList](#)): OK

401: Unauthorized

create create a DaemonSet

HTTP Request

POST /apis/apps/v1/namespaces/{namespace}/daemonsets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DaemonSet](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([DaemonSet](#)): OK

201 ([DaemonSet](#)): Created

202 ([DaemonSet](#)): Accepted

401: Unauthorized

update replace the specified DaemonSet

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/daemonsets/{name}

Parameters

- **name** (*in path*): string, required
 - name of the DaemonSet
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DaemonSet](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([DaemonSet](#)): OK

201 ([DaemonSet](#)): Created

401: Unauthorized

update replace status of the specified DaemonSet

HTTP Request

PUT /apis/apps/v1/namespaces/{namespace}/daemonsets/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the DaemonSet

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DaemonSet](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([DaemonSet](#)): OK

201 ([DaemonSet](#)): Created

401: Unauthorized

patch partially update the specified DaemonSet

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/daemonsets/{name}

Parameters

- **name** (*in path*): string, required
 - name of the DaemonSet
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([DaemonSet](#)): OK

201 ([DaemonSet](#)): Created

401: Unauthorized

patch partially update status of the specified DaemonSet

HTTP Request

PATCH /apis/apps/v1/namespaces/{namespace}/daemonsets/{name}/status

Parameters

- **name** (*in path*): string, required

name of the DaemonSet

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([DaemonSet](#)): OK

201 ([DaemonSet](#)): Created

401: Unauthorized

delete delete a DaemonSet

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/daemonsets/{name}

Parameters

- **name** (*in path*): string, required

name of the DaemonSet

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of DaemonSet

HTTP Request

DELETE /apis/apps/v1/namespaces/{namespace}/daemonsets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

sendInitialEvents (*in query*): boolean

-

[sendInitialEvents](#)

• **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Job

Job represents the configuration of a single job.

apiVersion: batch/v1

```
import "k8s.io/api/batch/v1"
```

Job

Job represents the configuration of a single job.

• **apiVersion**: batch/v1

• **kind**: Job

• **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

• **spec** ([JobSpec](#))

Specification of the desired behavior of a job. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

• **status** ([JobStatus](#))

Current status of a job. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

JobSpec

JobSpec describes how the job execution will look like.

Replicas

- **template** ([PodTemplateSpec](#)), required

Describes the pod that will be created when executing a job. The only allowed template.spec.restartPolicy values are "Never" or "OnFailure". More info: <https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/>

- **parallelism** (int32)

Specifies the maximum desired number of pods the job should run at any given time. The actual number of pods running in steady state will be less than this number when ((.spec.completions - .status.successful) < .spec.parallelism), i.e. when the work left to do is less than max parallelism. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/>

Lifecycle

- **completions** (int32)

Specifies the desired number of successfully finished pods the job should be run with. Setting to null means that the success of any pod signals the success of all pods, and allows parallelism to have any positive value. Setting to 1 means that parallelism is limited to 1 and the success of that pod signals the success of the job. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/>

- **completionMode** (string)

completionMode specifies how Pod completions are tracked. It can be NonIndexed (default) or Indexed.

NonIndexed means that the Job is considered complete when there have been .spec.completions successfully completed Pods. Each Pod completion is homologous to each other.

Indexed means that the Pods of a Job get an associated completion index from 0 to (.spec.completions - 1), available in the annotation batch.kubernetes.io/job-completion-index. The Job is considered complete when there is one successfully completed Pod for each index. When value is Indexed, .spec.completions must be specified and .spec.parallelism must be less than or equal to 10^5 . In addition, The Pod name takes the form \$(job-name)-\$(index)-\$(random-string), the Pod hostname takes the form \$(job-name)-\$(index).

More completion modes can be added in the future. If the Job controller observes a mode that it doesn't recognize, which is possible during upgrades due to version skew, the controller skips updates for the Job.

- **backoffLimit** (int32)

Specifies the number of retries before marking this job failed. Defaults to 6

- **activeDeadlineSeconds** (int64)

Specifies the duration in seconds relative to the startTime that the job may be continuously active before the system tries to terminate it; value must be positive integer. If a Job is suspended (at creation or through an update), this timer will effectively be stopped and reset when the Job is resumed again.

- **ttlSecondsAfterFinished** (int32)

ttlSecondsAfterFinished limits the lifetime of a Job that has finished execution (either Complete or Failed). If this field is set, ttlSecondsAfterFinished after the Job finishes, it is eligible to be automatically deleted. When the Job is being deleted, its lifecycle guarantees (e.g. finalizers) will be honored. If this field is unset, the Job won't be automatically deleted. If this field is set to zero, the Job becomes eligible to be deleted immediately after it finishes.

- **suspend** (boolean)

suspend specifies whether the Job controller should create Pods or not. If a Job is created with suspend set to true, no Pods are created by the Job controller. If a Job is suspended after creation (i.e. the flag goes from false to true), the Job controller will delete all active Pods associated with this Job. Users must design their workload to gracefully handle this. Suspending a Job will reset the StartTime field of the Job, effectively resetting the ActiveDeadlineSeconds timer too. Defaults to false.

Selector

- **selector** ([LabelSelector](#))

A label query over pods that should match the pod count. Normally, the system sets this field for you. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors>

- **manualSelector** (boolean)

manualSelector controls generation of pod labels and pod selectors. Leave manualSelector unset unless you are certain what you are doing. When false or unset, the system picks labels unique to this job and appends those labels to the pod template. When true, the user is responsible for picking unique labels and specifying the selector. Failure to pick a unique label may cause this and other jobs to not function correctly. However, You may see manualSelector=true in jobs that were created with the old extensions/v1beta1 API. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/#specifying-your-own-pod-selector>

Beta level

- **podFailurePolicy** (PodFailurePolicy)

Specifies the policy of handling failed pods. In particular, it allows to specify the set of actions and conditions which need to be satisfied to take the associated action. If empty, the default behaviour applies - the counter of failed pods, represented by the job's .status.failed field, is incremented and it is checked against the backoffLimit. This field cannot be used in combination with restartPolicy=OnFailure.

This field is beta-level. It can be used when the JobPodFailurePolicy feature gate is enabled (enabled by default).

PodFailurePolicy describes how failed pods influence the backoffLimit.

- **podFailurePolicy.rules** ([]PodFailurePolicyRule), required

Atomic: will be replaced during a merge

A list of pod failure policy rules. The rules are evaluated in order. Once a rule matches a Pod failure, the remaining of the rules are ignored. When no rule matches the Pod failure, the default handling applies - the counter of pod failures is incremented and it is checked against the backoffLimit. At most 20 elements are allowed.

PodFailurePolicyRule describes how a pod failure is handled when the requirements are met. One of onExitCodes and onPodConditions, but not both, can be used in each rule.

- **podFailurePolicy.rules.action** (string), required

Specifies the action taken on a pod failure when the requirements are satisfied. Possible values are:

- FailJob: indicates that the pod's job is marked as Failed and all running pods are terminated.
- FailIndex: indicates that the pod's index is marked as Failed and will not be restarted. This value is alpha-level. It can be used when the JobBackoffLimitPerIndex feature gate is enabled (disabled by default).
- Ignore: indicates that the counter towards the .backoffLimit is not incremented and a replacement pod is created.
- Count: indicates that the pod is handled in the default way - the counter towards the .backoffLimit is incremented. Additional values are considered to be added in the future. Clients should react to an unknown action by skipping the rule.

- **podFailurePolicy.rules.onPodConditions** ([]PodFailurePolicyOnPodConditionsPattern), required

Atomic: will be replaced during a merge

Represents the requirement on the pod conditions. The requirement is represented as a list of pod condition patterns. The requirement is satisfied if at least one pattern matches an actual pod condition. At most 20 elements are allowed.

PodFailurePolicyOnPodConditionsPattern describes a pattern for matching an actual pod condition type.

- **podFailurePolicy.rules.onPodConditions.status** (string), required

Specifies the required Pod condition status. To match a pod condition it is required that the specified status equals the pod condition status. Defaults to True.

podFailurePolicy.rules.onPodConditions.type (string), required

Specifies the required Pod condition type. To match a pod condition it is required that specified type equals the pod condition type.

▪ **podFailurePolicy.rules.onExitCodes**

(PodFailurePolicyOnExitCodesRequirement)

Represents the requirement on the container exit codes.

PodFailurePolicyOnExitCodesRequirement describes the requirement for handling a failed pod based on its container exit codes. In particular, it lookups the .state.terminated.exitCode for each app container and init container status, represented by the .status.containerStatuses and .status.initContainerStatuses fields in the Pod status, respectively. Containers completed with success (exit code 0) are excluded from the requirement check.

▪ **podFailurePolicy.rules.onExitCodes.operator** (string), required

Represents the relationship between the container exit code(s) and the specified values. Containers completed with success (exit code 0) are excluded from the requirement check. Possible values are:

- In: the requirement is satisfied if at least one container exit code (might be multiple if there are multiple containers not restricted by the 'containerName' field) is in the set of specified values.
- NotIn: the requirement is satisfied if at least one container exit code (might be multiple if there are multiple containers not restricted by the 'containerName' field) is not in the set of specified values. Additional values are considered to be added in the future. Clients should react to an unknown operator by assuming the requirement is not satisfied.

▪ **podFailurePolicy.rules.onExitCodes.values** ([]int32), required

Set: unique values will be kept during a merge

Specifies the set of values. Each returned container exit code (might be multiple in case of multiple containers) is checked against this set of values with respect to the operator. The list of values must be ordered and must not contain duplicates. Value '0' cannot be used for the In operator. At least one element is required. At most 255 elements are allowed.

▪ **podFailurePolicy.rules.onExitCodes.containerName** (string)

Restricts the check for exit codes to the container with the specified name. When null, the rule applies to all containers. When specified, it should match one the container or initContainer names in the pod template.

Alpha level

- **backoffLimitPerIndex** (int32)

Specifies the limit for the number of retries within an index before marking this index as failed. When enabled the number of failures per index is kept in the pod's batch.kubernetes.io/job-index-failure-count annotation. It can only be set when Job's completionMode=Indexed, and the Pod's restart policy is Never. The field is immutable. This field is alpha-level. It can be used when the JobBackoffLimitPerIndex feature gate is enabled (disabled by default).

- **maxFailedIndexes** (int32)

Specifies the maximal number of failed indexes before marking the Job as failed, when backoffLimitPerIndex is set. Once the number of failed indexes exceeds this number the entire Job is marked as Failed and its execution is terminated. When left as null the job continues execution of all of its indexes and is marked with the Complete Job condition. It can only be specified when backoffLimitPerIndex is set. It can be null or up to completions. It is required and must be less than or equal to 10^4 when is completions greater than 10^5 . This field is alpha-level. It can be used when the JobBackoffLimitPerIndex feature gate is enabled (disabled by default).

- **podReplacementPolicy** (string)

podReplacementPolicy specifies when to create replacement Pods. Possible values are: - TerminatingOrFailed means that we recreate pods when they are terminating (has a metadata.deletionTimestamp) or failed.

- Failed means to wait until a previously created Pod is fully terminated (has phase Failed or Succeeded) before creating a replacement Pod.

When using podFailurePolicy, Failed is the the only allowed value. TerminatingOrFailed and Failed are allowed values when podFailurePolicy is not in use. This is an alpha field. Enable JobPodReplacementPolicy to be able to use this field.

JobStatus

JobStatus represents the current state of a Job.

- **startTime** (Time)

Represents time when the job controller started processing a job. When a Job is created in the suspended state, this field is not set until the first time it is resumed. This field is reset every time a Job is resumed from suspension. It is represented in RFC3339 form and is in UTC.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **completionTime** (Time)

Represents time when the job was completed. It is not guaranteed to be set in happens-before order across separate operations. It is represented in RFC3339 form and is in UTC. The completion time is only set when the job finishes successfully.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

active (int32)

- The number of pending and running pods.

• **failed** (int32)

The number of pods which reached phase Failed.

• **succeeded** (int32)

The number of pods which reached phase Succeeded.

• **completedIndexes** (string)

completedIndexes holds the completed indexes when .spec.completionMode = "Indexed" in a text format. The indexes are represented as decimal integers separated by commas. The numbers are listed in increasing order. Three or more consecutive numbers are compressed and represented by the first and last element of the series, separated by a hyphen. For example, if the completed indexes are 1, 3, 4, 5 and 7, they are represented as "1,3-5,7".

• **conditions** ([]JobCondition)

Patch strategy: merge on key type

Atomic: will be replaced during a merge

The latest available observations of an object's current state. When a Job fails, one of the conditions will have type "Failed" and status true. When a Job is suspended, one of the conditions will have type "Suspended" and status true; when the Job is resumed, the status of this condition will become false. When a Job is completed, one of the conditions will have type "Complete" and status true. More info: <https://kubernetes.io/docs/concepts/workloads/controllers/jobs-run-to-completion/>

JobCondition describes current state of a job.

◦ **conditions.status** (string), required

Status of the condition, one of True, False, Unknown.

◦ **conditions.type** (string), required

Type of job condition, Complete or Failed.

◦ **conditions.lastProbeTime** (Time)

Last time the condition was checked.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

◦ **conditions.lastTransitionTime** (Time)

Last time the condition transit from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

Human readable message indicating details about last transition.

- **conditions.reason** (string)

(brief) reason for the condition's last transition.

- **unaccountedTerminatedPods** (UnaccountedTerminatedPods)

unaccountedTerminatedPods holds the UIDs of Pods that have terminated but the job controller hasn't yet accounted for in the status counters.

The job controller creates pods with a finalizer. When a pod terminates (succeeded or failed), the controller does three steps to account for it in the job status:

1. Add the pod UID to the arrays in this field.
2. Remove the pod finalizer.
3. Remove the pod UID from the arrays while increasing the corresponding counter.

Old jobs might not be tracked using this field, in which case the field remains null.

UnaccountedTerminatedPods holds UIDs of Pods that have terminated but haven't been accounted in Job status counters.

- **unaccountedTerminatedPods.failed** ([]string)

Set: unique values will be kept during a merge

failed holds UIDs of failed Pods.

- **unaccountedTerminatedPods.succeeded** ([]string)

Set: unique values will be kept during a merge

succeeded holds UIDs of succeeded Pods.

Beta level

- **ready** (int32)

The number of pods which have a Ready condition.

This field is beta-level. The job controller populates the field when the feature gate JobReadyPods is enabled (enabled by default).

Alpha level

- **failedIndexes** (string)

FailedIndexes holds the failed indexes when backoffLimitPerIndex=true. The indexes are represented in the text format analogous as for the completedIndexes field, ie. they are

kept as decimal integers separated by commas. The numbers are listed in increasing order. Three or more consecutive numbers are compressed and represented by the first and last element of the series, separated by a hyphen. For example, if the failed indexes are 1, 3, 4, 5 and 7, they are represented as "1,3-5,7". This field is alpha-level. It can be used when the JobBackoffLimitPerIndex feature gate is enabled (disabled by default).

- **terminating** (int32)

The number of pods which are terminating (in phase Pending or Running and have a deletionTimestamp).

This field is alpha-level. The job controller populates the field when the feature gate JobPodReplacementPolicy is enabled (disabled by default).

JobList

JobList is a collection of jobs.

- **apiVersion**: batch/v1

- **kind**: JobList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][Job](#)), required

items is the list of Jobs.

Operations

get read the specified Job

HTTP Request

GET /apis/batch/v1/namespaces/{namespace}/jobs/{name}

Parameters

- **name** (*in path*): string, required

name of the Job

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Job](#)): OK

401: Unauthorized

get read status of the specified Job

HTTP Request

GET /apis/batch/v1/namespaces/{namespace}/jobs/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Job

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Job](#)): OK

401: Unauthorized

list list or watch objects of kind Job

HTTP Request

GET /apis/batch/v1/namespaces/{namespace}/jobs

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([JobList](#)): OK

401: Unauthorized

list list or watch objects of kind Job

HTTP Request

GET /apis/batch/v1/jobs

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([JobList](#)): OK

401: Unauthorized

create create a Job

HTTP Request

POST /apis/batch/v1/namespaces/{namespace}/jobs

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Job](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Job](#)): OK

201 ([Job](#)): Created

202 ([Job](#)): Accepted

401: Unauthorized

update replace the specified Job

HTTP Request

PUT /apis/batch/v1/namespaces/{namespace}/jobs/{name}

Parameters

- **name** (*in path*): string, required

name of the Job

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Job](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Job](#)): OK

201 ([Job](#)): Created

401: Unauthorized

update replace status of the specified Job

HTTP Request

PUT /apis/batch/v1/namespaces/{namespace}/jobs/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Job

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Job](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Job](#)): OK

201 ([Job](#)): Created

401: Unauthorized

patch partially update the specified Job

HTTP Request

PATCH /apis/batch/v1/namespaces/{namespace}/jobs/{name}

Parameters

- **name** (*in path*): string, required

name of the Job

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Job](#)): OK

201 ([Job](#)): Created

401: Unauthorized

patch partially update status of the specified Job

HTTP Request

PATCH /apis/batch/v1/namespaces/{namespace}/jobs/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the Job
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Job](#)): OK

201 ([Job](#)): Created

401: Unauthorized

delete delete a Job

HTTP Request

DELETE /apis/batch/v1/namespaces/{namespace}/jobs/{name}

Parameters

- **name** (*in path*): string, required
name of the Job
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Job

HTTP Request

DELETE /apis/batch/v1/namespaces/{namespace}/jobs

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)

- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

CronJob

CronJob represents the configuration of a single cron job.

```
apiVersion: batch/v1  
import "k8s.io/api/batch/v1"
```

CronJob

CronJob represents the configuration of a single cron job.

- **apiVersion:** batch/v1

- **kind:** CronJob

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([CronJobSpec](#))

Specification of the desired behavior of a cron job, including the schedule. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([CronJobStatus](#))

Current status of a cron job. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

CronJobSpec

CronJobSpec describes how the job execution will look like and when it will actually run.

- **jobTemplate** ([JobTemplateSpec](#)), required

Specifies the job that will be created when executing a CronJob.

JobTemplateSpec describes the data a Job should have when created from a template

- **jobTemplate.metadata** ([ObjectMeta](#))

Standard object's metadata of the jobs created from this template. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **jobTemplate.spec** ([JobSpec](#))

Specification of the desired behavior of the job. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **schedule** (string), required

The schedule in Cron format, see <https://en.wikipedia.org/wiki/Cron>.

- **timeZone** (string)

The time zone name for the given schedule, see https://en.wikipedia.org/wiki/List_of_tz_database_time_zones. If not specified, this will default to the time zone of the kube-controller-manager process. The set of valid time zone names and the time zone offset is loaded from the system-wide time zone database by the API server during CronJob validation and the controller manager during execution. If no system-wide time zone database can be found a bundled version of the database is used instead. If the time zone name becomes invalid during the lifetime of a CronJob or due to a change in host configuration, the controller will stop creating new new Jobs and will create a system event with the reason UnknownTimeZone. More information can be found in <https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/#time-zones>

- **concurrencyPolicy** (string)

Specifies how to treat concurrent executions of a Job. Valid values are:

- "Allow" (default): allows CronJobs to run concurrently; - "Forbid": forbids concurrent runs, skipping next run if previous run hasn't finished yet; - "Replace": cancels currently running job and replaces it with a new one

- **startingDeadlineSeconds** (int64)

Optional deadline in seconds for starting the job if it misses scheduled time for any reason. Missed jobs executions will be counted as failed ones.

- **suspend** (boolean)

This flag tells the controller to suspend subsequent executions, it does not apply to already started executions. Defaults to false.

- **successfulJobsHistoryLimit** (int32)

The number of successful finished jobs to retain. Value must be non-negative integer. Defaults to 3.

- **failedJobsHistoryLimit** (int32)

The number of failed finished jobs to retain. Value must be non-negative integer. Defaults to 1.

CronJobStatus

CronJobStatus represents the current state of a cron job.

- **active** ([][ObjectReference](#))

Atomic: will be replaced during a merge

A list of pointers to currently running jobs.

lastScheduleTime (Time)

- Information when was the last time the job was successfully scheduled.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

lastSuccessfulTime (Time)

Information when was the last time the job successfully completed.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

CronJobList

CronJobList is a collection of cron jobs.

- **apiVersion**: batch/v1

- **kind**: CronJobList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][CronJob](#)), required

items is the list of CronJobs.

Operations

get read the specified CronJob

HTTP Request

GET /apis/batch/v1/namespaces/{namespace}/cronjobs/{name}

Parameters

- **name** (*in path*): string, required

name of the CronJob

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CronJob](#)): OK

401: Unauthorized

get read status of the specified CronJob

HTTP Request

GET /apis/batch/v1/namespaces/{namespace}/cronjobs/{name}/status

Parameters

- **name** (*in path*): string, required

name of the CronJob

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CronJob](#)): OK

401: Unauthorized

list list or watch objects of kind CronJob

HTTP Request

GET /apis/batch/v1/namespaces/{namespace}/cronjobs

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([CronJobList](#)): OK

401: Unauthorized

list list or watch objects of kind CronJob

HTTP Request

GET /apis/batch/v1/cronjobs

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([CronJobList](#)): OK

401: Unauthorized

create create a CronJob

HTTP Request

POST /apis/batch/v1/namespaces/{namespace}/cronjobs

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [CronJob](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CronJob](#)): OK

201 ([CronJob](#)): Created

202 ([CronJob](#)): Accepted

401: Unauthorized

update replace the specified CronJob

HTTP Request

PUT /apis/batch/v1/namespaces/{namespace}/cronjobs/{name}

Parameters

- **name** (*in path*): string, required
name of the CronJob
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [CronJob](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CronJob](#)): OK

201 ([CronJob](#)): Created

401: Unauthorized

update replace status of the specified CronJob

HTTP Request

PUT /apis/batch/v1/namespaces/{namespace}/cronjobs/{name}/status

Parameters

- **name** (*in path*): string, required

name of the CronJob

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [CronJob](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CronJob](#)): OK

201 ([CronJob](#)): Created

401: Unauthorized

patch partially update the specified CronJob

HTTP Request

PATCH /apis/batch/v1/namespaces/{namespace}/cronjobs/{name}

Parameters

- **name** (*in path*): string, required

name of the CronJob

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CronJob](#)): OK

201 ([CronJob](#)): Created

401: Unauthorized

patch partially update status of the specified CronJob

HTTP Request

PATCH /apis/batch/v1/namespaces/{namespace}/cronjobs/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the CronJob
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([CronJob](#)): OK

201 ([CronJob](#)): Created

401: Unauthorized

delete delete a CronJob

HTTP Request

DELETE /apis/batch/v1/namespaces/{namespace}/cronjobs/{name}

Parameters

- **name** (*in path*): string, required
name of the CronJob
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of CronJob

HTTP Request

DELETE /apis/batch/v1/namespaces/{namespace}/cronjobs

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)

- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

HorizontalPodAutoscaler

configuration of a horizontal pod autoscaler.

```
apiVersion: autoscaling/v1  
import "k8s.io/api/autoscaling/v1"
```

HorizontalPodAutoscaler

configuration of a horizontal pod autoscaler.

- **apiVersion**: autoscaling/v1
- **kind**: HorizontalPodAutoscaler
- **metadata** ([ObjectMeta](#))

Standard object metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([HorizontalPodAutoscalerSpec](#))

spec defines the behaviour of autoscaler. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>.

- **status** ([HorizontalPodAutoscalerStatus](#))

status is the current information about the autoscaler.

HorizontalPodAutoscalerSpec

specification of a horizontal pod autoscaler.

- **maxReplicas** (int32), required

maxReplicas is the upper limit for the number of pods that can be set by the autoscaler; cannot be smaller than MinReplicas.

- **scaleTargetRef** (CrossVersionObjectReference), required

reference to scaled resource; horizontal pod autoscaler will learn the current resource consumption and will set the desired number of pods by using its Scale subresource.

CrossVersionObjectReference contains enough information to let you identify the referred resource.

- **scaleTargetRef.kind** (string), required

kind is the kind of the referent; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **scaleTargetRef.name** (string), required

name is the name of the referent; More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **scaleTargetRef.apiVersion** (string)
 - apiVersion is the API version of the referent
- **minReplicas** (int32)

minReplicas is the lower limit for the number of replicas to which the autoscaler can scale down. It defaults to 1 pod. minReplicas is allowed to be 0 if the alpha feature gate HPA ScaleToZero is enabled and at least one Object or External metric is configured. Scaling is active as long as at least one metric value is available.
- **targetCPUUtilizationPercentage** (int32)

targetCPUUtilizationPercentage is the target average CPU utilization (represented as a percentage of requested CPU) over all the pods; if not specified the default autoscaling policy will be used.

HorizontalPodAutoscalerStatus

current status of a horizontal pod autoscaler

- **currentReplicas** (int32), required

currentReplicas is the current number of replicas of pods managed by this autoscaler.
- **desiredReplicas** (int32), required

desiredReplicas is the desired number of replicas of pods managed by this autoscaler.
- **currentCPUUtilizationPercentage** (int32)

currentCPUUtilizationPercentage is the current average CPU utilization over all pods, represented as a percentage of requested CPU, e.g. 70 means that an average pod is using now 70% of its requested CPU.
- **lastScaleTime** (Time)

lastScaleTime is the last time the HorizontalPodAutoscaler scaled the number of pods; used by the autoscaler to control how often the number of pods is changed.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.
- **observedGeneration** (int64)

observedGeneration is the most recent generation observed by this autoscaler.

HorizontalPodAutoscalerList

list of horizontal pod autoscaler objects.

- **apiVersion**: autoscaling/v1

- **kind**: HorizontalPodAutoscalerList
- **metadata** ([ListMeta](#))
 - Standard list metadata.
- **items** ([][HorizontalPodAutoscaler](#)), required
 - items is the list of horizontal pod autoscaler objects.

Operations

get read the specified HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required
 - name of the HorizontalPodAutoscaler
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

401: Unauthorized

get read status of the specified HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the HorizontalPodAutoscaler

namespace (*in path*): string, required

- [namespace](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

401: Unauthorized

list list or watch objects of kind HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers

Parameters

• **namespace** (*in path*): string, required

[namespace](#)

• **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

• **continue** (*in query*): string

[continue](#)

• **fieldSelector** (*in query*): string

[fieldSelector](#)

• **labelSelector** (*in query*): string

[labelSelector](#)

• **limit** (*in query*): integer

[limit](#)

• **pretty** (*in query*): string

[pretty](#)

• **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([HorizontalPodAutoscalerList](#)): OK

401: Unauthorized

list list or watch objects of kind HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v1/horizontalpodautoscalers

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([HorizontalPodAutoscalerList](#)): OK

401: Unauthorized

create create a HorizontalPodAutoscaler

HTTP Request

POST /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [HorizontalPodAutoscaler](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string

[pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

202 ([HorizontalPodAutoscaler](#)): Accepted

401: Unauthorized

update replace the specified HorizontalPodAutoscaler

HTTP Request

PUT /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required

name of the HorizontalPodAutoscaler

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [HorizontalPodAutoscaler](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

update replace status of the specified HorizontalPodAutoscaler

HTTP Request

PUT /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers/{name}/status

Parameters

- **name** (*in path*): string, required
name of the HorizontalPodAutoscaler
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [HorizontalPodAutoscaler](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

patch partially update the specified HorizontalPodAutoscaler

HTTP Request

PATCH /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required
name of the HorizontalPodAutoscaler

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

patch partially update status of the specified HorizontalPodAutoscaler

HTTP Request

PATCH /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the HorizontalPodAutoscaler
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

delete delete a HorizontalPodAutoscaler

HTTP Request

DELETE /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required

name of the HorizontalPodAutoscaler

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of HorizontalPodAutoscaler

HTTP Request

DELETE /apis/autoscaling/v1/namespaces/{namespace}/horizontalpodautoscalers

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

HorizontalPodAutoscaler

HorizontalPodAutoscaler is the configuration for a horizontal pod autoscaler, which automatically manages the replica count of any resource implementing the scale subresource based on the metrics specified.

```
apiVersion: autoscaling/v2
```

```
import "k8s.io/api/autoscaling/v2"
```

HorizontalPodAutoscaler

HorizontalPodAutoscaler is the configuration for a horizontal pod autoscaler, which automatically manages the replica count of any resource implementing the scale subresource based on the metrics specified.

-
- **apiVersion**: autoscaling/v2

- **kind**: HorizontalPodAutoscaler

- **metadata** ([ObjectMeta](#))

metadata is the standard object metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

spec ([HorizontalPodAutoscalerSpec](#))

- spec is the specification for the behaviour of the autoscaler. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>.
- **status** ([HorizontalPodAutoscalerStatus](#))

status is the current information about the autoscaler.

HorizontalPodAutoscalerSpec

HorizontalPodAutoscalerSpec describes the desired functionality of the HorizontalPodAutoscaler.

- **maxReplicas** (int32), required

maxReplicas is the upper limit for the number of replicas to which the autoscaler can scale up. It cannot be less than minReplicas.

- **scaleTargetRef** (CrossVersionObjectReference), required

scaleTargetRef points to the target resource to scale, and is used to the pods for which metrics should be collected, as well as to actually change the replica count.

CrossVersionObjectReference contains enough information to let you identify the referred resource.

- **scaleTargetRef.kind** (string), required

kind is the kind of the referent; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **scaleTargetRef.name** (string), required

name is the name of the referent; More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **scaleTargetRef.apiVersion** (string)

apiVersion is the API version of the referent

- **minReplicas** (int32)

minReplicas is the lower limit for the number of replicas to which the autoscaler can scale down. It defaults to 1 pod. minReplicas is allowed to be 0 if the alpha feature gate HPAutoScaleToZero is enabled and at least one Object or External metric is configured. Scaling is active as long as at least one metric value is available.

- **behavior** (HorizontalPodAutoscalerBehavior)

behavior configures the scaling behavior of the target in both Up and Down directions (scaleUp and scaleDown fields respectively). If not set, the default HPAScalingRules for scale up and scale down are used.

HorizontalPodAutoscalerBehavior configures the scaling behavior of the target in both Up and Down directions (scaleUp and scaleDown fields respectively).

- **behavior.scaleDown** (HPAScalingRules)

scaleDown is scaling policy for scaling Down. If not set, the default value is to allow to scale down to minReplicas pods, with a 300 second stabilization window (i.e., the highest recommendation for the last 300sec is used).

HPAScalingRules configures the scaling behavior for one direction. These Rules are applied after calculating DesiredReplicas from metrics for the HPA. They can limit the scaling velocity by specifying scaling policies. They can prevent flapping by specifying the stabilization window, so that the number of replicas is not set instantly, instead, the safest value from the stabilization window is chosen.

- **behavior.scaleDown.policies** ([]HPAScalingPolicy)

Atomic: will be replaced during a merge

policies is a list of potential scaling polices which can be used during scaling. At least one policy must be specified, otherwise the HPAScalingRules will be discarded as invalid

HPAScalingPolicy is a single policy which must hold true for a specified past interval.

- **behavior.scaleDown.policies.type** (string), required

type is used to specify the scaling policy.

- **behavior.scaleDown.policies.value** (int32), required

value contains the amount of change which is permitted by the policy. It must be greater than zero

- **behavior.scaleDown.policies.periodSeconds** (int32), required

periodSeconds specifies the window of time for which the policy should hold true. PeriodSeconds must be greater than zero and less than or equal to 1800 (30 min).

- **behavior.scaleDown.selectPolicy** (string)

selectPolicy is used to specify which policy should be used. If not set, the default value Max is used.

- **behavior.scaleDown.stabilizationWindowSeconds** (int32)

stabilizationWindowSeconds is the number of seconds for which past recommendations should be considered while scaling up or scaling down. StabilizationWindowSeconds must be greater than or equal to zero and less than or equal to 3600 (one hour). If not set, use the default values: - For scale up: 0 (i.e. no stabilization is done). - For scale down: 300 (i.e. the stabilization window is 300 seconds long).

behavior.scaleUp (HPAScalingRules)

- scaleUp is scaling policy for scaling Up. If not set, the default value is the higher of:
 - increase no more than 4 pods per 60 seconds
 - double the number of pods per 60 seconds No stabilization is used.

HPAScalingRules configures the scaling behavior for one direction. These Rules are applied after calculating DesiredReplicas from metrics for the HPA. They can limit the scaling velocity by specifying scaling policies. They can prevent flapping by specifying the stabilization window, so that the number of replicas is not set instantly, instead, the safest value from the stabilization window is chosen.

▪ **behavior.scaleUp.policies** ([]HPAScalingPolicy)

Atomic: will be replaced during a merge

policies is a list of potential scaling polices which can be used during scaling. At least one policy must be specified, otherwise the HPAScalingRules will be discarded as invalid

HPAScalingPolicy is a single policy which must hold true for a specified past interval.

▪ **behavior.scaleUp.policies.type** (string), required

type is used to specify the scaling policy.

▪ **behavior.scaleUp.policies.value** (int32), required

value contains the amount of change which is permitted by the policy. It must be greater than zero

▪ **behavior.scaleUp.policies.periodSeconds** (int32), required

periodSeconds specifies the window of time for which the policy should hold true. PeriodSeconds must be greater than zero and less than or equal to 1800 (30 min).

▪ **behavior.scaleUp.selectPolicy** (string)

selectPolicy is used to specify which policy should be used. If not set, the default value Max is used.

▪ **behavior.scaleUp.stabilizationWindowSeconds** (int32)

stabilizationWindowSeconds is the number of seconds for which past recommendations should be considered while scaling up or scaling down. StabilizationWindowSeconds must be greater than or equal to zero and less than or equal to 3600 (one hour). If not set, use the default values: - For scale up: 0 (i.e. no stabilization is done). - For scale down: 300 (i.e. the stabilization window is 300 seconds long).

• **metrics** ([]MetricSpec)

Atomic: will be replaced during a merge

metrics contains the specifications for which to use to calculate the desired replica count (the maximum replica count across all metrics will be used). The desired replica count is calculated multiplying the ratio between the target value and the current value by the current number of pods. Ergo, metrics used must decrease as the pod count is increased, and vice-versa. See the individual metric source types for more information about how each type of metric must respond. If not set, the default metric will be set to 80% average CPU utilization.

MetricSpec specifies how to scale based on a single metric (only type and one other matching field should be set at once).

- **metrics.type** (string), required

type is the type of metric source. It should be one of "ContainerResource", "External", "Object", "Pods" or "Resource", each mapping to a matching field in the object. Note: "ContainerResource" type is available on when the feature-gate HPAContainerMetrics is enabled

- **metrics.containerResource** (ContainerResourceMetricSource)

containerResource refers to a resource metric (such as those specified in requests and limits) known to Kubernetes describing a single container in each pod of the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source. This is an alpha feature and can be enabled by the HPAContainerMetrics feature flag.

ContainerResourceMetricSource indicates how to scale on a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). The values will be averaged together before being compared to the target. Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source. Only one "target" type should be set.

- **metrics.containerResource.container** (string), required

container is the name of the container in the pods of the scaling target

- **metrics.containerResource.name** (string), required

name is the name of the resource in question.

- **metrics.containerResource.target** (MetricTarget), required

target specifies the target value for the given metric

MetricTarget defines the target value, average value, or average utilization of a specific metric

- **metrics.containerResource.target.type** (string), required

type represents whether the metric type is Utilization, Value, or AverageValue

metrics.containerResource.target.averageUtilization (int32)

averageUtilization is the target value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods. Currently only valid for Resource metric source type

▪ **metrics.containerResource.target.averageValue** ([Quantity](#))

averageValue is the target value of the average of the metric across all relevant pods (as a quantity)

▪ **metrics.containerResource.target.value** ([Quantity](#))

value is the target value of the metric (as a quantity).

◦ **metrics.external** (ExternalMetricSource)

external refers to a global metric that is not associated with any Kubernetes object. It allows autoscaling based on information coming from components running outside of cluster (for example length of queue in cloud messaging service, or QPS from loadbalancer running outside of cluster).

ExternalMetricSource indicates how to scale on a metric not associated with any Kubernetes object (for example length of queue in cloud messaging service, or QPS from loadbalancer running outside of cluster).

▪ **metrics.external.metric** (MetricIdentifier), required

metric identifies the target metric by name and selector

MetricIdentifier defines the name and optionally selector for a metric

▪ **metrics.external.metric.name** (string), required

name is the name of the given metric

▪ **metrics.external.metric.selector** ([LabelSelector](#))

selector is the string-encoded form of a standard kubernetes label selector for the given metric When set, it is passed as an additional parameter to the metrics server for more specific metrics scoping. When unset, just the metricName will be used to gather metrics.

▪ **metrics.external.target** (MetricTarget), required

target specifies the target value for the given metric

MetricTarget defines the target value, average value, or average utilization of a specific metric

▪ **metrics.external.target.type** (string), required

type represents whether the metric type is Utilization, Value, or AverageValue

metrics.external.target.averageUtilization (int32)

averageUtilization is the target value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods. Currently only valid for Resource metric source type

▪ **metrics.external.target.averageValue** ([Quantity](#))

averageValue is the target value of the average of the metric across all relevant pods (as a quantity)

▪ **metrics.external.target.value** ([Quantity](#))

value is the target value of the metric (as a quantity).

◦ **metrics.object** (ObjectMetricSource)

object refers to a metric describing a single kubernetes object (for example, hits-per-second on an Ingress object).

ObjectMetricSource indicates how to scale on a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).

▪ **metrics.object.describedObject** (CrossVersionObjectReference), required

describedObject specifies the descriptions of a object,such as kind,name apiVersion

CrossVersionObjectReference contains enough information to let you identify the referred resource.

▪ **metrics.object.describedObject.kind** (string), required

kind is the kind of the referent; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

▪ **metrics.object.describedObject.name** (string), required

name is the name of the referent; More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

▪ **metrics.object.describedObject.apiVersion** (string)

apiVersion is the API version of the referent

▪ **metrics.object.metric** (MetricIdentifier), required

metric identifies the target metric by name and selector

MetricIdentifier defines the name and optionally selector for a metric

▪ **metrics.object.metric.name** (string), required

name is the name of the given metric

metrics.object.metric.selector ([LabelSelector](#))

selector is the string-encoded form of a standard kubernetes label selector for the given metric When set, it is passed as an additional parameter to the metrics server for more specific metrics scoping. When unset, just the metricName will be used to gather metrics.

▪ **metrics.object.target** (MetricTarget), required

target specifies the target value for the given metric

MetricTarget defines the target value, average value, or average utilization of a specific metric

▪ **metrics.object.target.type** (string), required

type represents whether the metric type is Utilization, Value, or AverageValue

▪ **metrics.object.target.averageUtilization** (int32)

averageUtilization is the target value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods. Currently only valid for Resource metric source type

▪ **metrics.object.target.averageValue** ([Quantity](#))

averageValue is the target value of the average of the metric across all relevant pods (as a quantity)

▪ **metrics.object.target.value** ([Quantity](#))

value is the target value of the metric (as a quantity).

◦ **metrics.pods** (PodsMetricSource)

pods refers to a metric describing each pod in the current scale target (for example, transactions-processed-per-second). The values will be averaged together before being compared to the target value.

PodsMetricSource indicates how to scale on a metric describing each pod in the current scale target (for example, transactions-processed-per-second). The values will be averaged together before being compared to the target value.

▪ **metrics.pods.metric** (MetricIdentifier), required

metric identifies the target metric by name and selector

MetricIdentifier defines the name and optionally selector for a metric

▪ **metrics.pods.metric.name** (string), required

name is the name of the given metric

metrics.pods.metric.selector ([LabelSelector](#))

selector is the string-encoded form of a standard kubernetes label selector for the given metric When set, it is passed as an additional parameter to the metrics server for more specific metrics scoping. When unset, just the metricName will be used to gather metrics.

- **metrics.pods.target** (MetricTarget), required

target specifies the target value for the given metric

MetricTarget defines the target value, average value, or average utilization of a specific metric

- **metrics.pods.target.type** (string), required

type represents whether the metric type is Utilization, Value, or AverageValue

- **metrics.pods.target.averageUtilization** (int32)

averageUtilization is the target value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods. Currently only valid for Resource metric source type

- **metrics.pods.target.averageValue** ([Quantity](#))

averageValue is the target value of the average of the metric across all relevant pods (as a quantity)

- **metrics.pods.target.value** ([Quantity](#))

value is the target value of the metric (as a quantity).

- **metrics.resource** (ResourceMetricSource)

resource refers to a resource metric (such as those specified in requests and limits) known to Kubernetes describing each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

ResourceMetricSource indicates how to scale on a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). The values will be averaged together before being compared to the target. Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source. Only one "target" type should be set.

- **metrics.resource.name** (string), required

name is the name of the resource in question.

- **metrics.resource.target** (MetricTarget), required

target specifies the target value for the given metric

MetricTarget defines the target value, average value, or average utilization of a specific metric

- **metrics.resource.target.type** (string), required

type represents whether the metric type is Utilization, Value, or AverageValue

- **metrics.resource.target.averageUtilization** (int32)

averageUtilization is the target value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods. Currently only valid for Resource metric source type

- **metrics.resource.target.averageValue** ([Quantity](#))

averageValue is the target value of the average of the metric across all relevant pods (as a quantity)

- **metrics.resource.target.value** ([Quantity](#))

value is the target value of the metric (as a quantity).

HorizontalPodAutoscalerStatus

HorizontalPodAutoscalerStatus describes the current status of a horizontal pod autoscaler.

- **desiredReplicas** (int32), required

desiredReplicas is the desired number of replicas of pods managed by this autoscaler, as last calculated by the autoscaler.

- **conditions** ([]HorizontalPodAutoscalerCondition)

Patch strategy: merge on key type

Map: unique values on key type will be kept during a merge

conditions is the set of conditions required for this autoscaler to scale its target, and indicates whether or not those conditions are met.

HorizontalPodAutoscalerCondition describes the state of a HorizontalPodAutoscaler at a certain point.

- **conditions.status** (string), required

status is the status of the condition (True, False, Unknown)

- **conditions.type** (string), required

type describes the current condition

conditions.lastTransitionTime (Time)

- lastTransitionTime is the last time the condition transitioned from one status to another

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

message is a human-readable explanation containing details about the transition

- **conditions.reason** (string)

reason is the reason for the condition's last transition.

• **currentMetrics** ([]MetricStatus)

Atomic: will be replaced during a merge

currentMetrics is the last read state of the metrics used by this autoscaler.

MetricStatus describes the last-read state of a single metric.

- **currentMetrics.type** (string), required

type is the type of metric source. It will be one of "ContainerResource", "External", "Object", "Pods" or "Resource", each corresponds to a matching field in the object. Note: "ContainerResource" type is available on when the feature-gate HPAContainerMetrics is enabled

- **currentMetrics.containerResource** (ContainerResourceMetricStatus)

container resource refers to a resource metric (such as those specified in requests and limits) known to Kubernetes describing a single container in each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

ContainerResourceMetricStatus indicates the current value of a resource metric known to Kubernetes, as specified in requests and limits, describing a single container in each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

- **currentMetrics.containerResource.container** (string), required

container is the name of the container in the pods of the scaling target

- **currentMetrics.containerResource.current** (MetricValueStatus), required

current contains the current value for the given metric

MetricValueStatus holds the current value for a metric

- **currentMetrics.containerResource.current.averageUtilization** (int32)

currentAverageUtilization is the current value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods.

- **currentMetrics.containerResource.current.averageValue** ([Quantity](#))

averageValue is the current value of the average of the metric across all relevant pods (as a quantity)

- **currentMetrics.containerResource.current.value** ([Quantity](#))

value is the current value of the metric (as a quantity).

- **currentMetrics.containerResource.name** (string), required

name is the name of the resource in question.

- **currentMetrics.external** (ExternalMetricStatus)

external refers to a global metric that is not associated with any Kubernetes object. It allows autoscaling based on information coming from components running outside of cluster (for example length of queue in cloud messaging service, or QPS from loadbalancer running outside of cluster).

ExternalMetricStatus indicates the current value of a global metric not associated with any Kubernetes object.

- **currentMetrics.external.current** (MetricValueStatus), required

current contains the current value for the given metric

MetricValueStatus holds the current value for a metric

- **currentMetrics.external.current.averageUtilization** (int32)

currentAverageUtilization is the current value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods.

- **currentMetrics.external.current.averageValue** ([Quantity](#))

averageValue is the current value of the average of the metric across all relevant pods (as a quantity)

- **currentMetrics.external.current.value** ([Quantity](#))

value is the current value of the metric (as a quantity).

- **currentMetrics.external.metric** (MetricIdentifier), required

metric identifies the target metric by name and selector

MetricIdentifier defines the name and optionally selector for a metric

- **currentMetrics.external.metric.name** (string), required

name is the name of the given metric

- **currentMetrics.external.metric.selector** ([LabelSelector](#))

selector is the string-encoded form of a standard kubernetes label selector for the given metric When set, it is passed as an additional parameter to the metrics server for more specific metrics scoping. When unset, just the metricName will be used to gather metrics.

- **currentMetrics.object** (ObjectMetricStatus)

object refers to a metric describing a single kubernetes object (for example, hits-per-second on an Ingress object).

ObjectMetricStatus indicates the current value of a metric describing a kubernetes object (for example, hits-per-second on an Ingress object).

- **currentMetrics.object.current** (MetricValueStatus), required

current contains the current value for the given metric

MetricValueStatus holds the current value for a metric

- **currentMetrics.object.current.averageUtilization** (int32)

currentAverageUtilization is the current value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods.

- **currentMetrics.object.current.averageValue** ([Quantity](#))

averageValue is the current value of the average of the metric across all relevant pods (as a quantity)

- **currentMetrics.object.current.value** ([Quantity](#))

value is the current value of the metric (as a quantity).

- **currentMetrics.object.describedObject** (CrossVersionObjectReference), required

DescribedObject specifies the descriptions of a object,such as kind,name apiVersion

CrossVersionObjectReference contains enough information to let you identify the referred resource.

- **currentMetrics.object.describedObject.kind** (string), required

kind is the kind of the referent; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **currentMetrics.object.describedObject.name** (string), required
name is the name of the referent; More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **currentMetrics.object.describedObject.apiVersion** (string)
apiVersion is the API version of the referent

- **currentMetrics.object.metric** (MetricIdentifier), required

metric identifies the target metric by name and selector

MetricIdentifier defines the name and optionally selector for a metric

- **currentMetrics.object.metric.name** (string), required

name is the name of the given metric

- **currentMetrics.object.metric.selector** ([LabelSelector](#))

selector is the string-encoded form of a standard kubernetes label selector for the given metric When set, it is passed as an additional parameter to the metrics server for more specific metrics scoping. When unset, just the metricName will be used to gather metrics.

- **currentMetrics.pods** (PodsMetricStatus)

pods refers to a metric describing each pod in the current scale target (for example, transactions-processed-per-second). The values will be averaged together before being compared to the target value.

PodsMetricStatus indicates the current value of a metric describing each pod in the current scale target (for example, transactions-processed-per-second).

- **currentMetrics.pods.current** (MetricValueStatus), required

current contains the current value for the given metric

MetricValueStatus holds the current value for a metric

- **currentMetrics.pods.current.averageUtilization** (int32)

currentAverageUtilization is the current value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods.

- **currentMetrics.pods.current.averageValue** ([Quantity](#))

averageValue is the current value of the average of the metric across all relevant pods (as a quantity)

currentMetrics.pods.current.value ([Quantity](#))

- value is the current value of the metric (as a quantity).
- **currentMetrics.pods.metric** (MetricIdentifier), required

metric identifies the target metric by name and selector

MetricIdentifier defines the name and optionally selector for a metric

- **currentMetrics.pods.metric.name** (string), required
- name is the name of the given metric
- **currentMetrics.pods.metric.selector** ([LabelSelector](#))

selector is the string-encoded form of a standard kubernetes label selector for the given metric When set, it is passed as an additional parameter to the metrics server for more specific metrics scoping. When unset, just the metricName will be used to gather metrics.

◦ **currentMetrics.resource** (ResourceMetricStatus)

resource refers to a resource metric (such as those specified in requests and limits) known to Kubernetes describing each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

ResourceMetricStatus indicates the current value of a resource metric known to Kubernetes, as specified in requests and limits, describing each pod in the current scale target (e.g. CPU or memory). Such metrics are built in to Kubernetes, and have special scaling options on top of those available to normal per-pod metrics using the "pods" source.

- **currentMetrics.resource.current** (MetricValueStatus), required

current contains the current value for the given metric

MetricValueStatus holds the current value for a metric

- **currentMetrics.resource.current.averageUtilization** (int32)

currentAverageUtilization is the current value of the average of the resource metric across all relevant pods, represented as a percentage of the requested value of the resource for the pods.

- **currentMetrics.resource.current.averageValue** ([Quantity](#))

averageValue is the current value of the average of the metric across all relevant pods (as a quantity)

- **currentMetrics.resource.current.value** ([Quantity](#))

value is the current value of the metric (as a quantity).

- **currentMetrics.resource.name** (string), required
 - name is the name of the resource in question.
- **currentReplicas** (int32)
 - currentReplicas is current number of replicas of pods managed by this autoscaler, as last seen by the autoscaler.
- **lastScaleTime** (Time)
 - lastScaleTime is the last time the HorizontalPodAutoscaler scaled the number of pods, used by the autoscaler to control how often the number of pods is changed.
Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.
- **observedGeneration** (int64)
 - observedGeneration is the most recent generation observed by this autoscaler.

HorizontalPodAutoscalerList

HorizontalPodAutoscalerList is a list of horizontal pod autoscaler objects.

- **apiVersion**: autoscaling/v2
- **kind**: HorizontalPodAutoscalerList
- **metadata** ([ListMeta](#))
 - metadata is the standard list metadata.
- **items** ([][HorizontalPodAutoscaler](#)), required
 - items is the list of horizontal pod autoscaler objects.

Operations

get read the specified HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required
 - name of the HorizontalPodAutoscaler

- **namespace** (*in path*): string, required

- [namespace](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

401: Unauthorized

get read status of the specified HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}/status

Parameters

- **name** (*in path*): string, required

- name of the HorizontalPodAutoscaler

- **namespace** (*in path*): string, required

- [namespace](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

401: Unauthorized

list list or watch objects of kind HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers

Parameters

- **namespace** (*in path*): string, required

- [namespace](#)

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([HorizontalPodAutoscalerList](#)): OK

401: Unauthorized

list list or watch objects of kind HorizontalPodAutoscaler

HTTP Request

GET /apis/autoscaling/v2/horizontalpodautoscalers

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([HorizontalPodAutoscalerList](#)): OK

401: Unauthorized

create create a HorizontalPodAutoscaler

HTTP Request

POST /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [HorizontalPodAutoscaler](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

202 ([HorizontalPodAutoscaler](#)): Accepted

401: Unauthorized

update replace the specified HorizontalPodAutoscaler

HTTP Request

PUT /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required

name of the HorizontalPodAutoscaler

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [HorizontalPodAutoscaler](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

update replace status of the specified HorizontalPodAutoscaler

HTTP Request

PUT /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}/status

Parameters

- **name** (*in path*): string, required

name of the HorizontalPodAutoscaler

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [HorizontalPodAutoscaler](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

patch partially update the specified HorizontalPodAutoscaler

HTTP Request

PATCH /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required
 - name of the HorizontalPodAutoscaler
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

patch partially update status of the specified HorizontalPodAutoscaler

HTTP Request

PATCH /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the HorizontalPodAutoscaler
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([HorizontalPodAutoscaler](#)): OK

201 ([HorizontalPodAutoscaler](#)): Created

401: Unauthorized

delete delete a HorizontalPodAutoscaler

HTTP Request

DELETE /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers/{name}

Parameters

- **name** (*in path*): string, required
 - name of the HorizontalPodAutoscaler
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of HorizontalPodAutoscaler

HTTP Request

DELETE /apis/autoscaling/v2/namespaces/{namespace}/horizontalpodautoscalers

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

PriorityClass

PriorityClass defines mapping from a priority class name to the priority integer value.

```
apiVersion: scheduling.k8s.io/v1
```

```
import "k8s.io/api/scheduling/v1"
```

PriorityClass

PriorityClass defines mapping from a priority class name to the priority integer value. The value can be any valid integer.

- **apiVersion:** scheduling.k8s.io/v1

- **kind:** PriorityClass

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **value** (int32), required

value represents the integer value of this priority class. This is the actual priority that pods receive when they have the name of this class in their pod spec.

- **description** (string)

description is an arbitrary string that usually provides guidelines on when this priority class should be used.

- **globalDefault** (boolean)

globalDefault specifies whether this PriorityClass should be considered as the default priority for pods that do not have any priority class. Only one PriorityClass can be marked as globalDefault. However, if more than one PriorityClasses exists with their globalDefault field set to true, the smallest value of such global default PriorityClasses will be used as the default priority.

- **preemptionPolicy** (string)

preemptionPolicy is the Policy for preempting pods with lower priority. One of Never, PreemptLowerPriority. Defaults to PreemptLowerPriority if unset.

PriorityClassList

PriorityClassList is a collection of priority classes.

- **apiVersion:** scheduling.k8s.io/v1

kind: PriorityclassList

- **metadata** ([ListMeta](#))

Standard list metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][PriorityClass](#)), required

items is the list of PriorityClasses

Operations

get read the specified PriorityClass

HTTP Request

GET /apis/scheduling.k8s.io/v1/priorityclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the PriorityClass

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityClass](#)): OK

401: Unauthorized

list list or watch objects of kind PriorityClass

HTTP Request

GET /apis/scheduling.k8s.io/v1/priorityclasses

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([PriorityClassList](#)): OK

401: Unauthorized

create create a PriorityClass

HTTP Request

POST /apis/scheduling.k8s.io/v1/priorityclasses

Parameters

- **body**: [PriorityClass](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityClass](#)): OK

201 ([PriorityClass](#)): Created

202 ([PriorityClass](#)): Accepted

401: Unauthorized

update replace the specified PriorityClass

HTTP Request

PUT /apis/scheduling.k8s.io/v1/priorityclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the PriorityClass

- **body**: [PriorityClass](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityClass](#)): OK

201 ([PriorityClass](#)): Created

401: Unauthorized

patch partially update the specified PriorityClass

HTTP Request

PATCH /apis/scheduling.k8s.io/v1/priorityclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the PriorityClass

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityClass](#)): OK

201 ([PriorityClass](#)): Created

401: Unauthorized

delete delete a PriorityClass

HTTP Request

DELETE /apis/scheduling.k8s.io/v1/priorityclasses/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PriorityClass
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of PriorityClass

HTTP Request

DELETE /apis/scheduling.k8s.io/v1/priorityclasses

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

PodSchedulingContext v1alpha2

PodSchedulingContext objects hold information that is needed to schedule a Pod with ResourceClaims that use "WaitForFirstConsumer" allocation mode.

apiVersion: resource.k8s.io/v1alpha2

```
import "k8s.io/api/resource/v1alpha2"
```

PodSchedulingContext

PodSchedulingContext objects hold information that is needed to schedule a Pod with ResourceClaims that use "WaitForFirstConsumer" allocation mode.

This is an alpha type and requires enabling the DynamicResourceAllocation feature gate.

- **apiVersion**: resource.k8s.io/v1alpha2

- **kind**: PodSchedulingContext

- **metadata** ([ObjectMeta](#))

Standard object metadata

- **spec** ([PodSchedulingContextSpec](#)), required

Spec describes where resources for the Pod are needed.

- **status** ([PodSchedulingContextStatus](#))

Status describes where resources for the Pod can be allocated.

PodSchedulingContextSpec

PodSchedulingContextSpec describes where resources for the Pod are needed.

- **potentialNodes** ([]string)

Set: unique values will be kept during a merge

PotentialNodes lists nodes where the Pod might be able to run.

The size of this field is limited to 128. This is large enough for many clusters. Larger clusters may need more attempts to find a node that suits all pending resources. This may get increased in the future, but not reduced.

- **selectedNode** (string)

SelectedNode is the node for which allocation of ResourceClaims that are referenced by the Pod and that use "WaitForFirstConsumer" allocation is to be attempted.

PodSchedulingContextStatus

PodSchedulingContextStatus describes where resources for the Pod can be allocated.

- **resourceClaims** ([]ResourceClaimSchedulingStatus)

Map: unique values on key name will be kept during a merge

ResourceClaims describes resource availability for each pod.spec.resourceClaim entry where the corresponding ResourceClaim uses "WaitForFirstConsumer" allocation mode.

ResourceClaimSchedulingStatus contains information about one particular ResourceClaim with "WaitForFirstConsumer" allocation mode.

- **resourceClaims.name** (string)

Name matches the pod.spec.resourceClaims[*].Name field.

- **resourceClaims.unsuitableNodes** ([]string)

Set: unique values will be kept during a merge

UnsuitableNodes lists nodes that the ResourceClaim cannot be allocated for.

The size of this field is limited to 128, the same as for PodSchedulingSpec.PotentialNodes. This may get increased in the future, but not reduced.

PodSchedulingContextList

PodSchedulingContextList is a collection of Pod scheduling objects.

- **apiVersion**: resource.k8s.io/v1alpha2

- **kind**: PodSchedulingContextList

- **metadata** ([ListMeta](#))

Standard list metadata

- **items** ([][PodSchedulingContext](#)), required

Items is the list of PodSchedulingContext objects.

Operations

get read the specified PodSchedulingContext

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts/{name}

Parameters

- **name** (*in path*): string, required

name of the PodSchedulingContext

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PodSchedulingContext](#)): OK

401: Unauthorized

get read status of the specified PodSchedulingContext

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts/{name}/status

Parameters

- **name** (*in path*): string, required

name of the PodSchedulingContext

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PodSchedulingContext](#)): OK

401: Unauthorized

list list or watch objects of kind PodSchedulingContext

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([PodSchedulingContextList](#)): OK

401: Unauthorized

list list or watch objects of kind PodSchedulingContext

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/podschedulingcontexts

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([PodSchedulingContextList](#)): OK

401: Unauthorized

create create a PodSchedulingContext

HTTP Request

POST /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [PodSchedulingContext](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PodSchedulingContext](#)): OK

201 ([PodSchedulingContext](#)): Created

202 ([PodSchedulingContext](#)): Accepted

401: Unauthorized

update replace the specified PodSchedulingContext

HTTP Request

PUT /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts/{name}

Parameters

- **name** (*in path*): string, required
name of the PodSchedulingContext
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [PodSchedulingContext](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PodSchedulingContext](#)): OK

201 ([PodSchedulingContext](#)): Created

401: Unauthorized

update replace status of the specified PodSchedulingContext

HTTP Request

PUT /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts/{name}/status

Parameters

- **name** (*in path*): string, required
name of the PodSchedulingContext
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [PodSchedulingContext](#), required

- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PodSchedulingContext](#)): OK

201 ([PodSchedulingContext](#)): Created

401: Unauthorized

patch partially update the specified PodSchedulingContext

HTTP Request

PATCH /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts/{name}

Parameters

- **name** (*in path*): string, required

name of the PodSchedulingContext
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)

- **force** (*in query*): boolean

- [force](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([PodSchedulingContext](#)): OK

201 ([PodSchedulingContext](#)): Created

401: Unauthorized

patch partially update status of the specified PodSchedulingContext

HTTP Request

PATCH /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts/{name}/status

Parameters

- **name** (*in path*): string, required

- name of the PodSchedulingContext

- **namespace** (*in path*): string, required

- [namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

- [dryRun](#)

- **fieldManager** (*in query*): string

- [fieldManager](#)

- **fieldValidation** (*in query*): string

- [fieldValidation](#)

- **force** (*in query*): boolean

- [force](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([PodSchedulingContext](#)): OK

201 ([PodSchedulingContext](#)): Created

401: Unauthorized

delete delete a PodSchedulingContext

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PodSchedulingContext
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([PodSchedulingContext](#)): OK

202 ([PodSchedulingContext](#)): Accepted

401: Unauthorized

deletecollection delete collection of PodSchedulingContext

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/podschedulingcontexts

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ResourceClaim v1alpha2

ResourceClaim describes which resources are needed by a resource consumer.

```
apiVersion: resource.k8s.io/v1alpha2
```

```
import "k8s.io/api/resource/v1alpha2"
```

ResourceClaim

ResourceClaim describes which resources are needed by a resource consumer. Its status tracks whether the resource has been allocated and what the resulting attributes are.

This is an alpha type and requires enabling the DynamicResourceAllocation feature gate.

- **apiVersion**: resource.k8s.io/v1alpha2

- **kind**: ResourceClaim

- **metadata** ([ObjectMeta](#))

Standard object metadata

- **spec** ([ResourceClaimSpec](#)), required

Spec describes the desired attributes of a resource that then needs to be allocated. It can only be set once when creating the ResourceClaim.

- **status** ([ResourceClaimStatus](#))

Status describes whether the resource is available and with which attributes.

ResourceClaimSpec

ResourceClaimSpec defines how a resource is to be allocated.

- **resourceClassName** (string), required

ResourceClassName references the driver and additional parameters via the name of a ResourceClass that was created as part of the driver deployment.

- **allocationMode** (string)

Allocation can start immediately or when a Pod wants to use the resource. "WaitForFirstConsumer" is the default.

- **parametersRef** (ResourceClaimParametersReference)

ParametersRef references a separate object with arbitrary parameters that will be used by the driver when allocating a resource for the claim.

The object must be in the same namespace as the ResourceClaim.

ResourceClaimParametersReference contains enough information to let you locate the parameters for a ResourceClaim. The object must be in the same namespace as the ResourceClaim.

- **parametersRef.kind** (string), required

Kind is the type of resource being referenced. This is the same value as in the parameter object's metadata, for example "ConfigMap".

- **parametersRef.name** (string), required

Name is the name of resource being referenced.

- **parametersRef.apiGroup** (string)

APIGroup is the group for the resource being referenced. It is empty for the core API. This matches the group in the APIVersion that is used when creating the resources.

ResourceClaimStatus

ResourceClaimStatus tracks whether the resource has been allocated and what the resulting attributes are.

- **allocation** (AllocationResult)

Allocation is set by the resource driver once a resource or set of resources has been allocated successfully. If this is not specified, the resources have not been allocated yet.

AllocationResult contains attributes of an allocated resource.

- **allocation.availableOnNodes** (NodeSelector)

This field will get set by the resource driver after it has allocated the resource to inform the scheduler where it can schedule Pods using the ResourceClaim.

Setting this field is optional. If null, the resource is available everywhere.

A node selector represents the union of the results of one or more label queries over a set of nodes; that is, it represents the OR of the selectors represented by the node selector terms.

- **allocation.availableOnNodes.nodeSelectorTerms** (`[]NodeSelectorTerm`), required

Required. A list of node selector terms. The terms are ORed.

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

- **allocation.availableOnNodes.nodeSelectorTerms.matchExpressions** (`([]NodeSelectorRequirement)`)

A list of node selector requirements by node's labels.

- **allocation.availableOnNodes.nodeSelectorTerms.matchFields** (`([]NodeSelectorRequirement)`)

A list of node selector requirements by node's fields.

- **allocation.resourceHandles** (`[]ResourceHandle`)

Atomic: will be replaced during a merge

ResourceHandles contain the state associated with an allocation that should be maintained throughout the lifetime of a claim. Each ResourceHandle contains data that should be passed to a specific kubelet plugin once it lands on a node. This data is returned by the driver after a successful allocation and is opaque to Kubernetes. Driver documentation may explain to users how to interpret this data if needed.

Setting this field is optional. It has a maximum size of 32 entries. If null (or empty), it is assumed this allocation will be processed by a single kubelet plugin with no ResourceHandle data attached. The name of the kubelet plugin invoked will match the DriverName set in the ResourceClaimStatus this AllocationResult is embedded in.

ResourceHandle holds opaque resource data for processing by a specific kubelet plugin.

- **allocation.resourceHandles.data** (string)

Data contains the opaque data associated with this ResourceHandle. It is set by the controller component of the resource driver whose name matches the DriverName set in the ResourceClaimStatus this ResourceHandle is embedded in. It is set at allocation time and is intended for processing by the kubelet plugin whose name matches the DriverName set in this ResourceHandle.

The maximum size of this field is 16KiB. This may get increased in the future, but not reduced.

- **allocation.resourceHandles.driverName** (string)

DriverName specifies the name of the resource driver whose kubelet plugin should be invoked to process this ResourceHandle's data once it lands on a node. This may differ from the DriverName set in ResourceClaimStatus this ResourceHandle is embedded in.

- **allocation.shareable** (boolean)

Shareable determines whether the resource supports more than one consumer at a time.

- **deallocationRequested** (boolean)

DeallocationRequested indicates that a ResourceClaim is to be deallocated.

The driver then must deallocate this claim and reset the field together with clearing the Allocation field.

While DeallocationRequested is set, no new consumers may be added to ReservedFor.

- **driverName** (string)

DriverName is a copy of the driver name from the ResourceClass at the time when allocation started.

- **reservedFor** ([]ResourceClaimConsumerReference)

Map: unique values on key uid will be kept during a merge

ReservedFor indicates which entities are currently allowed to use the claim. A Pod which references a ResourceClaim which is not reserved for that Pod will not be started.

There can be at most 32 such reservations. This may get increased in the future, but not reduced.

ResourceClaimConsumerReference contains enough information to let you locate the consumer of a ResourceClaim. The user must be a resource in the same namespace as the ResourceClaim.

- **reservedFor.name** (string), required

Name is the name of resource being referenced.

- **reservedFor.resource** (string), required

Resource is the type of resource being referenced, for example "pods".

- **reservedFor.uid** (string), required

UID identifies exactly one incarnation of the resource.

- **reservedFor.apiGroup** (string)

APIGroup is the group for the resource being referenced. It is empty for the core API. This matches the group in the APIVersion that is used when creating the resources.

ResourceClaimList

ResourceClaimList is a collection of claims.

- **apiVersion**: resource.k8s.io/v1alpha2

- **kind**: ResourceClaimList

- **metadata** ([ListMeta](#))

Standard list metadata

- **items** ([][ResourceClaim](#)), required

Items is the list of resource claims.

Operations

get read the specified ResourceClaim

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims/{name}

Parameters

- **name** (*in path*): string, required

name of the ResourceClaim

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ResourceClaim](#)): OK

401: Unauthorized

get read status of the specified ResourceClaim

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ResourceClaim
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceClaim](#)): OK

401: Unauthorized

list list or watch objects of kind ResourceClaim

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)

[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ResourceClaimList](#)): OK

401: Unauthorized

list list or watch objects of kind ResourceClaim

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/resourceclaims

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ResourceClaimList](#)): OK

401: Unauthorized

create create a ResourceClaim

HTTP Request

POST /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ResourceClaim](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

fieldValidation (*in query*): string

- [fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ResourceClaim](#)): OK

201 ([ResourceClaim](#)): Created

202 ([ResourceClaim](#)): Accepted

401: Unauthorized

update replace the specified ResourceClaim

HTTP Request

PUT /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims/{name}

Parameters

• **name** (*in path*): string, required

name of the ResourceClaim

• **namespace** (*in path*): string, required

[namespace](#)

• **body**: [ResourceClaim](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ResourceClaim](#)): OK

201 ([ResourceClaim](#)): Created

401: Unauthorized

update replace status of the specified ResourceClaim

HTTP Request

PUT /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims/{name}/status

Parameters

- **name** (*in path*): string, required
[name](#)
name of the ResourceClaim
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [ResourceClaim](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceClaim](#)): OK

201 ([ResourceClaim](#)): Created

401: Unauthorized

patch partially update the specified ResourceClaim

HTTP Request

PATCH /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims/{name}

Parameters

- **name** (*in path*): string, required
name of the ResourceClaim
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceClaim](#)): OK

201 ([ResourceClaim](#)): Created

401: Unauthorized

patch partially update status of the specified ResourceClaim

HTTP Request

PATCH /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ResourceClaim
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceClaim](#)): OK

201 ([ResourceClaim](#)): Created

401: Unauthorized

delete delete a ResourceClaim

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims/{name}

Parameters

- **name** (*in path*): string, required
name of the ResourceClaim
- **namespace** (*in path*): string, required

[namespace](#)

- **body:** [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([ResourceClaim](#)): OK

202 ([ResourceClaim](#)): Accepted

401: Unauthorized

deletecollection delete collection of ResourceClaim

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaims

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body:** [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ResourceClaimTemplate v1alpha2

ResourceClaimTemplate is used to produce ResourceClaim objects.

```
apiVersion: resource.k8s.io/v1alpha2
```

```
import "k8s.io/api/resource/v1alpha2"
```

ResourceClaimTemplate

ResourceClaimTemplate is used to produce ResourceClaim objects.

-
- **apiVersion**: resource.k8s.io/v1alpha2

- **kind**: ResourceClaimTemplate

- **metadata** ([ObjectMeta](#))

Standard object metadata

- **spec** ([ResourceClaimTemplateSpec](#)), required

Describes the ResourceClaim that is to be generated.

This field is immutable. A ResourceClaim will get created by the control plane for a Pod when needed and then not get updated anymore.

ResourceClaimTemplateSpec

ResourceClaimTemplateSpec contains the metadata and fields for a ResourceClaim.

- **spec** ([ResourceClaimSpec](#)), required

Spec for the ResourceClaim. The entire content is copied unchanged into the ResourceClaim that gets created from this template. The same fields as in a ResourceClaim are also valid here.

- **metadata** ([ObjectMeta](#))

ObjectMeta may contain labels and annotations that will be copied into the PVC when creating it. No other fields are allowed and will be rejected during validation.

ResourceClaimTemplateList

ResourceClaimTemplateList is a collection of claim templates.

- **apiVersion**: resource.k8s.io/v1alpha2

- **kind**: ResourceClaimTemplateList

- **metadata** ([ListMeta](#))

Standard list metadata

- **items** ([][ResourceClaimTemplate](#)), required

Items is the list of resource claim templates.

Operations

get read the specified ResourceClaimTemplate

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaimtemplates/{name}

Parameters

- **name** (*in path*): string, required
name of the ResourceClaimTemplate
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceClaimTemplate](#)): OK

401: Unauthorized

list list or watch objects of kind ResourceClaimTemplate

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaimtemplates

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)

- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ResourceClaimTemplateList](#)): OK

401: Unauthorized

list list or watch objects of kind ResourceClaimTemplate

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/resourceclaimtemplates

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)

- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)
- **watch** (*in query*): boolean
 - [watch](#)

Response

200 ([ResourceClaimTemplateList](#)): OK

401: Unauthorized

create create a ResourceClaimTemplate

HTTP Request

POST /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaimtemplates

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [ResourceClaimTemplate](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ResourceClaimTemplate](#)): OK

201 ([ResourceClaimTemplate](#)): Created

202 ([ResourceClaimTemplate](#)): Accepted

401: Unauthorized

update replace the specified ResourceClaimTemplate

HTTP Request

PUT /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaimtemplates/{name}

Parameters

- **name** (*in path*): string, required

name of the ResourceClaimTemplate

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ResourceClaimTemplate](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([ResourceClaimTemplate](#)): OK

201 ([ResourceClaimTemplate](#)): Created

401: Unauthorized

patch partially update the specified ResourceClaimTemplate

HTTP Request

PATCH /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaimtemplates/{name}

Parameters

- **name** (*in path*): string, required

- name of the ResourceClaimTemplate

- **namespace** (*in path*): string, required

- namespace

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

- dryRun

- **fieldManager** (*in query*): string

- fieldManager

- **fieldValidation** (*in query*): string

- fieldValidation

- **force** (*in query*): boolean

- force

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([ResourceClaimTemplate](#)): OK

201 ([ResourceClaimTemplate](#)): Created

401: Unauthorized

delete delete a ResourceClaimTemplate

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaimtemplates/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ResourceClaimTemplate
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([ResourceClaimTemplate](#)): OK

202 ([ResourceClaimTemplate](#)): Accepted

401: Unauthorized

deletecollection delete collection of ResourceClaimTemplate

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/namespaces/{namespace}/resourceclaimtemplates

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ResourceClass v1alpha2

ResourceClass is used by administrators to influence how resources are allocated.

```
apiVersion: resource.k8s.io/v1alpha2
```

```
import "k8s.io/api/resource/v1alpha2"
```

ResourceClass

ResourceClass is used by administrators to influence how resources are allocated.

This is an alpha type and requires enabling the DynamicResourceAllocation feature gate.

- **apiVersion**: resource.k8s.io/v1alpha2

- **kind**: ResourceClass

- **metadata** ([ObjectMeta](#))

Standard object metadata

- **driverName** (string), required

DriverName defines the name of the dynamic resource driver that is used for allocation of a ResourceClaim that uses this class.

Resource drivers have a unique name in forward domain order (acme.example.com).

- **parametersRef** (ResourceClassParametersReference)

ParametersRef references an arbitrary separate object that may hold parameters that will be used by the driver when allocating a resource that uses this class. A dynamic resource driver can distinguish between parameters stored here and those stored in ResourceClaimSpec.

ResourceClassParametersReference contains enough information to let you locate the parameters for a ResourceClass.

- **parametersRef.kind** (string), required

Kind is the type of resource being referenced. This is the same value as in the parameter object's metadata.

- **parametersRef.name** (string), required

Name is the name of resource being referenced.

- **parametersRef.apiGroup** (string)

APIGroup is the group for the resource being referenced. It is empty for the core API. This matches the group in the APIVersion that is used when creating the resources.

- **parametersRef.namespace** (string)

Namespace that contains the referenced resource. Must be empty for cluster-scoped resources and non-empty for namespaced resources.

- **suitableNodes** (NodeSelector)

Only nodes matching the selector will be considered by the scheduler when trying to find a Node that fits a Pod when that Pod uses a ResourceClaim that has not been allocated yet.

Setting this field is optional. If null, all nodes are candidates.

A node selector represents the union of the results of one or more label queries over a set of nodes; that is, it represents the OR of the selectors represented by the node selector terms.

- **suitableNodes.nodeSelectorTerms** ([]NodeSelectorTerm), required

Required. A list of node selector terms. The terms are ORed.

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

- **suitableNodes.nodeSelectorTerms.matchExpressions** ([][NodeSelectorRequirement](#))

A list of node selector requirements by node's labels.

- **suitableNodes.nodeSelectorTerms.matchFields** ([][NodeSelectorRequirement](#))

A list of node selector requirements by node's fields.

ResourceClassList

ResourceClassList is a collection of classes.

-
- **apiVersion**: resource.k8s.io/v1alpha2
 - **kind**: ResourceClassList

metadata ([ListMeta](#))

- Standard list metadata
- **items** ([][ResourceClass](#)), required
 - Items is the list of resource classes.

Operations

get read the specified ResourceClass

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/resourceclasses/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ResourceClass
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ResourceClass](#)): OK

401: Unauthorized

list list or watch objects of kind ResourceClass

HTTP Request

GET /apis/resource.k8s.io/v1alpha2/resourceclasses

Parameters

- **allowWatchBookmarks** (*in query*): boolean
 - [allowWatchBookmarks](#)
- **continue** (*in query*): string
 - [continue](#)
- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ResourceClassList](#)): OK

401: Unauthorized

create create a ResourceClass

HTTP Request

POST /apis/resource.k8s.io/v1alpha2/resourceclasses

Parameters

- **body**: [ResourceClass](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceClass](#)): OK

201 ([ResourceClass](#)): Created

202 ([ResourceClass](#)): Accepted

401: Unauthorized

update replace the specified ResourceClass

HTTP Request

PUT /apis/resource.k8s.io/v1alpha2/resourceclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the ResourceClass
- **body**: [ResourceClass](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceClass](#)): OK

201 ([ResourceClass](#)): Created

401: Unauthorized

patch partially update the specified ResourceClass

HTTP Request

PATCH /apis/resource.k8s.io/v1alpha2/resourceclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the ResourceClass

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ResourceClass](#)): OK

201 ([ResourceClass](#)): Created

401: Unauthorized

delete delete a ResourceClass

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/resourceclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the ResourceClass
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([ResourceClass](#)): OK

202 ([ResourceClass](#)): Accepted

401: Unauthorized

deletecollection delete collection of ResourceClass

HTTP Request

DELETE /apis/resource.k8s.io/v1alpha2/resourceclasses

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Service Resources

Service

Service is a named abstraction of software service (for example, mysql) consisting of local port (for example 3306) that the proxy listens on, and the selector that determines which pods will answer requests sent through the proxy.

Endpoints

Endpoints is a collection of endpoints that implement the actual service.

EndpointSlice

EndpointSlice represents a subset of the endpoints that implement a service.

Ingress

Ingress is a collection of rules that allow inbound connections to reach the endpoints defined by a backend.

IngressClass

IngressClass represents the class of the Ingress, referenced by the Ingress Spec.

Service

Service is a named abstraction of software service (for example, mysql) consisting of local port (for example 3306) that the proxy listens on, and the selector that determines which pods will answer requests sent through the proxy.

```
apiVersion: v1  
import "k8s.io/api/core/v1"
```

Service

Service is a named abstraction of software service (for example, mysql) consisting of local port (for example 3306) that the proxy listens on, and the selector that determines which pods will answer requests sent through the proxy.

-
- **apiVersion:** v1
 - **kind:** Service
 - **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([ServiceSpec](#))

Spec defines the behavior of a service. <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([ServiceStatus](#))

Most recently observed status of the service. Populated by the system. Read-only. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

ServiceSpec

ServiceSpec describes the attributes that a user creates on a service.

- **selector** (map[string]string)

Route service traffic to pods with label keys and values matching this selector. If empty or not present, the service is assumed to have an external process managing its endpoints, which Kubernetes will not modify. Only applies to types ClusterIP, NodePort, and LoadBalancer. Ignored if type is ExternalName. More info: <https://kubernetes.io/docs/concepts/services-networking/service/>

- **ports** ([]ServicePort)

Patch strategy: merge on key port

Map: unique values on keys port, protocol will be kept during a merge

The list of ports that are exposed by this service. More info: <https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies>

ServicePort contains information on service's port.

- **ports.port** (int32), required

The port that will be exposed by this service.

- **ports.targetPort** (IntOrString)

Number or name of the port to access on the pods targeted by the service. Number must be in the range 1 to 65535. Name must be an IANA_SVC_NAME. If this is a string, it will be looked up as a named port in the target Pod's container ports. If this is not specified, the value of the 'port' field is used (an identity map). This field is ignored for services with clusterIP=None, and should be omitted or set equal to the 'port' field. More info: <https://kubernetes.io/docs/concepts/services-networking/service/#defining-a-service>

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **ports.protocol** (string)

The IP protocol for this port. Supports "TCP", "UDP", and "SCTP". Default is TCP.

ports.name (string)

◦

The name of this port within the service. This must be a DNS_LABEL. All ports within a ServiceSpec must have unique names. When considering the endpoints for a Service, this must match the 'name' field in the EndpointPort. Optional if only one ServicePort is defined on this service.

◦ **ports.nodePort** (int32)

The port on each node on which this service is exposed when type is NodePort or LoadBalancer. Usually assigned by the system. If a value is specified, in-range, and not in use it will be used, otherwise the operation will fail. If not specified, a port will be allocated if this Service requires one. If this field is specified when creating a Service which does not need it, creation will fail. This field will be wiped when updating a Service to no longer need it (e.g. changing type from NodePort to ClusterIP). More info: <https://kubernetes.io/docs/concepts/services-networking/service/#type-nodeport>

◦ **ports.appProtocol** (string)

The application protocol for this port. This is used as a hint for implementations to offer richer behavior for protocols that they understand. This field follows standard Kubernetes label syntax. Valid values are either:

- Un-prefixed protocol names - reserved for IANA standard service names (as per RFC-6335 and <https://www.iana.org/assignments/service-names>).
- Kubernetes-defined prefixed names:
 - 'kubernetes.io/h2c' - HTTP/2 over cleartext as described in <https://www.rfc-editor.org/rfc/rfc7540>
 - 'kubernetes.io/ws' - WebSocket over cleartext as described in <https://www.rfc-editor.org/rfc/rfc6455>
 - 'kubernetes.io/wss' - WebSocket over TLS as described in <https://www.rfc-editor.org/rfc/rfc6455>
- Other protocols should use implementation-defined prefixed names such as mycompany.com/my-custom-protocol.

• **type** (string)

type determines how the Service is exposed. Defaults to ClusterIP. Valid options are ExternalName, ClusterIP, NodePort, and LoadBalancer. "ClusterIP" allocates a cluster-internal IP address for load-balancing to endpoints. Endpoints are determined by the selector or if that is not specified, by manual construction of an Endpoints object or EndpointSlice objects. If clusterIP is "None", no virtual IP is allocated and the endpoints are published as a set of endpoints rather than a virtual IP. "NodePort" builds on ClusterIP and allocates a port on every node which routes to the same endpoints as the clusterIP. "LoadBalancer" builds on NodePort and creates an external load-balancer (if supported in the current cloud) which routes to the same endpoints as the clusterIP. "ExternalName" aliases this service to the specified externalName. Several other fields do not apply to ExternalName services. More info: <https://kubernetes.io/docs/concepts/services-networking/service/#publishing-services-service-types>

ipFamilies ([]string)

- *Atomic: will be replaced during a merge*

IPFamilies is a list of IP families (e.g. IPv4, IPv6) assigned to this service. This field is usually assigned automatically based on cluster configuration and the ipFamilyPolicy field. If this field is specified manually, the requested family is available in the cluster, and ipFamilyPolicy allows it, it will be used; otherwise creation of the service will fail. This field is conditionally mutable: it allows for adding or removing a secondary IP family, but it does not allow changing the primary IP family of the Service. Valid values are "IPv4" and "IPv6". This field only applies to Services of types ClusterIP, NodePort, and LoadBalancer, and does apply to "headless" services. This field will be wiped when updating a Service to type ExternalName.

This field may hold a maximum of two entries (dual-stack families, in either order). These families must correspond to the values of the clusterIPs field, if specified. Both clusterIPs and ipFamilies are governed by the ipFamilyPolicy field.

- **ipFamilyPolicy (string)**

IPFamilyPolicy represents the dual-stack-ness requested or required by this Service. If there is no value provided, then this field will be set to SingleStack. Services can be "SingleStack" (a single IP family), "PreferDualStack" (two IP families on dual-stack configured clusters or a single IP family on single-stack clusters), or "RequireDualStack" (two IP families on dual-stack configured clusters, otherwise fail). The ipFamilies and clusterIPs fields depend on the value of this field. This field will be wiped when updating a service to type ExternalName.

- **clusterIP (string)**

clusterIP is the IP address of the service and is usually assigned randomly. If an address is specified manually, is in-range (as per system configuration), and is not in use, it will be allocated to the service; otherwise creation of the service will fail. This field may not be changed through updates unless the type field is also being changed to ExternalName (which requires this field to be blank) or the type field is being changed from ExternalName (in which case this field may optionally be specified, as described above). Valid values are "None", empty string (""), or a valid IP address. Setting this to "None" makes a "headless service" (no virtual IP), which is useful when direct endpoint connections are preferred and proxying is not required. Only applies to types ClusterIP, NodePort, and LoadBalancer. If this field is specified when creating a Service of type ExternalName, creation will fail. This field will be wiped when updating a Service to type ExternalName. More info: <https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies>

- **clusterIPs ([]string)**

- *Atomic: will be replaced during a merge*

ClusterIPs is a list of IP addresses assigned to this service, and are usually assigned randomly. If an address is specified manually, is in-range (as per system configuration), and is not in use, it will be allocated to the service; otherwise creation of the service will fail. This field may not be changed through updates unless the type field is also being changed to ExternalName (which requires this field to be empty) or the type field is being changed from ExternalName (in which case this field may optionally be specified, as

describe above). Valid values are "None", empty string (""), or a valid IP address. Setting this to "None" makes a "headless service" (no virtual IP), which is useful when direct endpoint connections are preferred and proxying is not required. Only applies to types ClusterIP, NodePort, and LoadBalancer. If this field is specified when creating a Service of type ExternalName, creation will fail. This field will be wiped when updating a Service to type ExternalName. If this field is not specified, it will be initialized from the clusterIP field. If this field is specified, clients must ensure that clusterIPs[0] and clusterIP have the same value.

This field may hold a maximum of two entries (dual-stack IPs, in either order). These IPs must correspond to the values of the ipFamilies field. Both clusterIPs and ipFamilies are governed by the ipFamilyPolicy field. More info: <https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies>

- **externalIPs** ([]string)

externalIPs is a list of IP addresses for which nodes in the cluster will also accept traffic for this service. These IPs are not managed by Kubernetes. The user is responsible for ensuring that traffic arrives at a node with this IP. A common example is external load-balancers that are not part of the Kubernetes system.

- **sessionAffinity** (string)

Supports "ClientIP" and "None". Used to maintain session affinity. Enable client IP based session affinity. Must be ClientIP or None. Defaults to None. More info: <https://kubernetes.io/docs/concepts/services-networking/service/#virtual-ips-and-service-proxies>

- **loadBalancerIP** (string)

Only applies to Service Type: LoadBalancer. This feature depends on whether the underlying cloud-provider supports specifying the loadBalancerIP when a load balancer is created. This field will be ignored if the cloud-provider does not support the feature. Deprecated: This field was under-specified and its meaning varies across implementations. Using it is non-portable and it may not support dual-stack. Users are encouraged to use implementation-specific annotations when available.

- **loadBalancerSourceRanges** ([]string)

If specified and supported by the platform, this will restrict traffic through the cloud-provider load-balancer will be restricted to the specified client IPs. This field will be ignored if the cloud-provider does not support the feature." More info: <https://kubernetes.io/docs/tasks/access-application-cluster/create-external-load-balancer/>

- **loadBalancerClass** (string)

loadBalancerClass is the class of the load balancer implementation this Service belongs to. If specified, the value of this field must be a label-style identifier, with an optional prefix, e.g. "internal-vip" or "example.com/internal-vip". Unprefixed names are reserved for end-users. This field can only be set when the Service type is 'LoadBalancer'. If not set, the default load balancer implementation is used, today this is typically done through the cloud provider integration, but should apply for any default implementation. If set, it is assumed that a load balancer implementation is watching for Services with a matching class. Any default load balancer implementation (e.g. cloud providers) should ignore

Services that set this field. This field can only be set when creating or updating a Service to type 'LoadBalancer'. Once set, it can not be changed. This field will be wiped when a service is updated to a non 'LoadBalancer' type.

- **externalName** (string)

externalName is the external reference that discovery mechanisms will return as an alias for this service (e.g. a DNS CNAME record). No proxying will be involved. Must be a lowercase RFC-1123 hostname (<https://tools.ietf.org/html/rfc1123>) and requires type to be "ExternalName".

- **externalTrafficPolicy** (string)

externalTrafficPolicy describes how nodes distribute service traffic they receive on one of the Service's "externally-facing" addresses (NodePorts, ExternalIPs, and LoadBalancer IPs). If set to "Local", the proxy will configure the service in a way that assumes that external load balancers will take care of balancing the service traffic between nodes, and so each node will deliver traffic only to the node-local endpoints of the service, without masquerading the client source IP. (Traffic mistakenly sent to a node with no endpoints will be dropped.) The default value, "Cluster", uses the standard behavior of routing to all endpoints evenly (possibly modified by topology and other features). Note that traffic sent to an External IP or LoadBalancer IP from within the cluster will always get "Cluster" semantics, but clients sending to a NodePort from within the cluster may need to take traffic policy into account when picking a node.

- **internalTrafficPolicy** (string)

InternalTrafficPolicy describes how nodes distribute service traffic they receive on the ClusterIP. If set to "Local", the proxy will assume that pods only want to talk to endpoints of the service on the same node as the pod, dropping the traffic if there are no local endpoints. The default value, "Cluster", uses the standard behavior of routing to all endpoints evenly (possibly modified by topology and other features).

- **healthCheckNodePort** (int32)

healthCheckNodePort specifies the healthcheck nodePort for the service. This only applies when type is set to LoadBalancer and externalTrafficPolicy is set to Local. If a value is specified, is in-range, and is not in use, it will be used. If not specified, a value will be automatically allocated. External systems (e.g. load-balancers) can use this port to determine if a given node holds endpoints for this service or not. If this field is specified when creating a Service which does not need it, creation will fail. This field will be wiped when updating a Service to no longer need it (e.g. changing type). This field cannot be updated once set.

- **publishNotReadyAddresses** (boolean)

publishNotReadyAddresses indicates that any agent which deals with endpoints for this Service should disregard any indications of ready/not-ready. The primary use case for setting this field is for a StatefulSet's Headless Service to propagate SRV DNS records for its Pods for the purpose of peer discovery. The Kubernetes controllers that generate Endpoints and EndpointSlice resources for Services interpret this to mean that all endpoints are considered "ready" even if the Pods themselves are not. Agents which consume only Kubernetes generated endpoints through the Endpoints or EndpointSlice resources can safely assume this behavior.

sessionAffinityConfig (SessionAffinityConfig)

- sessionAffinityConfig contains the configurations of session affinity.

SessionAffinityConfig represents the configurations of session affinity.

- **sessionAffinityConfig.clientIP** (ClientIPConfig)

clientIP contains the configurations of Client IP based session affinity.

ClientIPConfig represents the configurations of Client IP based session affinity.

- **sessionAffinityConfig.clientIP.timeoutSeconds** (int32)

timeoutSeconds specifies the seconds of ClientIP type session sticky time.

The value must be $>0 \ \&\& \leq 86400$ (for 1 day) if ServiceAffinity == "ClientIP".
Default value is 10800(for 3 hours).

- **allocateLoadBalancerNodePorts** (boolean)

allocateLoadBalancerNodePorts defines if NodePorts will be automatically allocated for services with type LoadBalancer. Default is "true". It may be set to "false" if the cluster load-balancer does not rely on NodePorts. If the caller requests specific NodePorts (by specifying a value), those requests will be respected, regardless of this field. This field may only be set for services with type LoadBalancer and will be cleared if the type is changed to any other type.

ServiceStatus

ServiceStatus represents the current status of a service.

- **conditions** ([]Condition)

Patch strategy: merge on key type

Map: unique values on key type will be kept during a merge

Current service state

Condition contains details for one aspect of the current state of this API Resource.

- **conditions.lastTransitionTime** (Time), required

lastTransitionTime is the last time the condition transitioned from one status to another. This should be when the underlying condition changed. If that is not known, then using the time when the API field changed is acceptable.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string), required

message is a human readable message indicating details about the transition. This may be an empty string.

- **conditions.reason** (string), required

reason contains a programmatic identifier indicating the reason for the condition's last transition. Producers of specific condition types may define expected values and meanings for this field, and whether the values are considered a guaranteed API. The value should be a CamelCase string. This field may not be empty.

- **conditions.status** (string), required

status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

type of condition in CamelCase or in foo.example.com/CamelCase.

- **conditions.observedGeneration** (int64)

observedGeneration represents the .metadata.generation that the condition was set based upon. For instance, if .metadata.generation is currently 12, but the .status.conditions[x].observedGeneration is 9, the condition is out of date with respect to the current state of the instance.

- **loadBalancer** (LoadBalancerStatus)

LoadBalancer contains the current status of the load-balancer, if one is present.

LoadBalancerStatus represents the status of a load-balancer.

- **loadBalancer.ingress** ([]LoadBalancerIngress)

Ingress is a list containing ingress points for the load-balancer. Traffic intended for the service should be sent to these ingress points.

LoadBalancerIngress represents the status of a load-balancer ingress point: traffic intended for the service should be sent to an ingress point.

- **loadBalancer.ingress.hostname** (string)

Hostname is set for load-balancer ingress points that are DNS based (typically AWS load-balancers)

- **loadBalancer.ingress.ip** (string)

IP is set for load-balancer ingress points that are IP based (typically GCE or OpenStack load-balancers)

- **loadBalancer.ingress.ipMode** (string)

IPMode specifies how the load-balancer IP behaves, and may only be specified when the ip field is specified. Setting this to "VIP" indicates that traffic is delivered to the node with the destination set to the load-balancer's IP and port. Setting this to "Proxy" indicates that traffic is delivered to the

node or pod with the destination set to the node's IP and node port or the pod's IP and port. Service implementations may use this information to adjust traffic routing.

- **loadBalancer.ingress.ports** ([]PortStatus)

Atomic: will be replaced during a merge

Ports is a list of records of service ports If used, every port defined in the service should have an entry in it

**

- **loadBalancer.ingress.ports.port** (int32), required

Port is the port number of the service port of which status is recorded here

- **loadBalancer.ingress.ports.protocol** (string), required

Protocol is the protocol of the service port of which status is recorded here The supported values are: "TCP", "UDP", "SCTP"

- **loadBalancer.ingress.ports.error** (string)

Error is to record the problem with the service port The format of the error shall comply with the following rules: - built-in error values shall be specified in this file and those shall use CamelCase names

- cloud provider specific error values must have names that comply with the format foo.example.com/CamelCase.

ServiceList

ServiceList holds a list of services.

- **apiVersion:** v1

- **kind:** ServiceList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][Service](#)), required

List of services

Operations

get read the specified Service

HTTP Request

GET /api/v1/namespaces/{namespace}/services/{name}

Parameters

- **name** (*in path*): string, required
name of the Service
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Service](#)): OK

401: Unauthorized

get read status of the specified Service

HTTP Request

GET /api/v1/namespaces/{namespace}/services/{name}/status

Parameters

- **name** (*in path*): string, required
name of the Service
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Service](#)): OK

401: Unauthorized

list list or watch objects of kind Service

HTTP Request

GET /api/v1/namespaces/{namespace}/services

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ServiceList](#)): OK

401: Unauthorized

list list or watch objects of kind Service

HTTP Request

GET /api/v1/services

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

watch (*in query*): boolean

- [watch](#)

Response

200 ([ServiceList](#)): OK

401: Unauthorized

create create a Service

HTTP Request

POST /api/v1/namespaces/{namespace}/services

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Service](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Service](#)): OK

201 ([Service](#)): Created

202 ([Service](#)): Accepted

401: Unauthorized

update replace the specified Service

HTTP Request

PUT /api/v1/namespaces/{namespace}/services/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Service
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Service](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Service](#)): OK

201 ([Service](#)): Created

401: Unauthorized

update replace status of the specified Service

HTTP Request

PUT /api/v1/namespaces/{namespace}/services/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the Service

namespace (*in path*): string, required

- [namespace](#)

- **body**: [Service](#), required

- **dryRun** (*in query*): string

- [dryRun](#)

- **fieldManager** (*in query*): string

- [fieldManager](#)

- **fieldValidation** (*in query*): string

- [fieldValidation](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([Service](#)): OK

201 ([Service](#)): Created

401: Unauthorized

patch partially update the specified Service

HTTP Request

PATCH /api/v1/namespaces/{namespace}/services/{name}

Parameters

- **name** (*in path*): string, required

- name of the Service

- **namespace** (*in path*): string, required

- [namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

- [dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Service](#)): OK

201 ([Service](#)): Created

401: Unauthorized

patch partially update status of the specified Service

HTTP Request

PATCH /api/v1/namespaces/{namespace}/services/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Service

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Service](#)): OK

201 ([Service](#)): Created

401: Unauthorized

delete delete a Service

HTTP Request

DELETE /api/v1/namespaces/{namespace}/services/{name}

Parameters

- **name** (*in path*): string, required

name of the Service

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Service](#)): OK

202 ([Service](#)): Accepted

401: Unauthorized

deletecollection delete collection of Service

HTTP Request

DELETE /api/v1/namespaces/{namespace}/services

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

sendInitialEvents (*in query*): boolean

-

[sendInitialEvents](#)

• **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Endpoints

Endpoints is a collection of endpoints that implement the actual service.

apiVersion: v1

```
import "k8s.io/api/core/v1"
```

Endpoints

Endpoints is a collection of endpoints that implement the actual service. Example:

```
Name: "mysvc",
Subsets: [
  {
    Addresses: [{"ip": "10.10.1.1"}, {"ip": "10.10.2.2"}],
    Ports: [{"name": "a", "port": 8675}, {"name": "b", "port": 309}]
  },
  {
    Addresses: [{"ip": "10.10.3.3"}],
    Ports: [{"name": "a", "port": 93}, {"name": "b", "port": 76}]
  },
]
```

• **apiVersion**: v1

• **kind**: Endpoints

• **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

• **subsets** ([]EndpointSubset)

The set of all endpoints is the union of all subsets. Addresses are placed into subsets according to the IPs they share. A single address with multiple ports, some of which are

ready and some of which are not (because they come from different containers) will result in the address being displayed in different subsets for the different ports. No address will appear in both Addresses and NotReadyAddresses in the same subset. Sets of addresses and ports that comprise a service.

*EndpointSubset is a group of addresses with a common set of ports. The expanded set of endpoints is the Cartesian product of Addresses x Ports. For example, given:

```
{ Addresses: [{"ip": "10.10.1.1"}, {"ip": "10.10.2.2"}], Ports: [{"name": "a", "port": 8675}, {"name": "b", "port": 309}] }
```

The resulting set of endpoints can be viewed as:

a: [10.10.1.1:8675, 10.10.2.2:8675], b: [10.10.1.1:309, 10.10.2.2:309]*

- **subsets.addresses** ([]EndpointAddress)

IP addresses which offer the related ports that are marked as ready. These endpoints should be considered safe for load balancers and clients to utilize.

EndpointAddress is a tuple that describes single IP address.

- **subsets.addresses.ip** (string), required

The IP of this endpoint. May not be loopback (127.0.0.0/8 or ::1), link-local (169.254.0.0/16 or fe80::/10), or link-local multicast (224.0.0.0/24 or ff02::/16).

- **subsets.addresses.hostname** (string)

The Hostname of this endpoint

- **subsets.addresses.nodeName** (string)

Optional: Node hosting this endpoint. This can be used to determine endpoints local to a node.

- **subsets.addresses.targetRef** ([ObjectReference](#))

Reference to object providing the endpoint.

- **subsets.notReadyAddresses** ([]EndpointAddress)

IP addresses which offer the related ports but are not currently marked as ready because they have not yet finished starting, have recently failed a readiness check, or have recently failed a liveness check.

EndpointAddress is a tuple that describes single IP address.

- **subsets.notReadyAddresses.ip** (string), required

The IP of this endpoint. May not be loopback (127.0.0.0/8 or ::1), link-local (169.254.0.0/16 or fe80::/10), or link-local multicast (224.0.0.0/24 or ff02::/16).

- **subsets.notReadyAddresses.hostname** (string)

The Hostname of this endpoint

subsets.notReadyAddresses.nodeName (string)

- Optional: Node hosting this endpoint. This can be used to determine endpoints local to a node.

- **subsets.notReadyAddresses.targetRef** ([ObjectReference](#))

Reference to object providing the endpoint.

- **subsets.ports** ([]EndpointPort)

Port numbers available on the related IP addresses.

EndpointPort is a tuple that describes a single port.

- **subsets.ports.port** (int32), required

The port number of the endpoint.

- **subsets.ports.protocol** (string)

The IP protocol for this port. Must be UDP, TCP, or SCTP. Default is TCP.

- **subsets.ports.name** (string)

The name of this port. This must match the 'name' field in the corresponding ServicePort. Must be a DNS_LABEL. Optional only if one port is defined.

- **subsets.ports.appProtocol** (string)

The application protocol for this port. This is used as a hint for implementations to offer richer behavior for protocols that they understand. This field follows standard Kubernetes label syntax. Valid values are either:

- Un-prefixed protocol names - reserved for IANA standard service names (as per RFC-6335 and <https://www.iana.org/assignments/service-names>).

- Kubernetes-defined prefixed names:

- 'kubernetes.io/h2c' - HTTP/2 over cleartext as described in <https://www.rfc-editor.org/rfc/rfc7540>

- 'kubernetes.io/ws' - WebSocket over cleartext as described in <https://www.rfc-editor.org/rfc/rfc6455>

- 'kubernetes.io/wss' - WebSocket over TLS as described in <https://www.rfc-editor.org/rfc/rfc6455>

- Other protocols should use implementation-defined prefixed names such as mycompany.com/my-custom-protocol.

EndpointsList

EndpointsList is a list of endpoints.

-
- **apiVersion**: v1
 - **kind**: EndpointsList
 - **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][Endpoints](#)), required

List of endpoints.

Operations

get read the specified Endpoints

HTTP Request

GET /api/v1/namespaces/{namespace}/endpoints/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Endpoints
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Endpoints](#)): OK

401: Unauthorized

list list or watch objects of kind Endpoints

HTTP Request

GET /api/v1/namespaces/{namespace}/endpoints

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([EndpointsList](#)): OK

401: Unauthorized

list list or watch objects of kind Endpoints

HTTP Request

GET /api/v1/endpoints

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([EndpointsList](#)): OK

401: Unauthorized

create create Endpoints

HTTP Request

POST /api/v1/namespaces/{namespace}/endpoints

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Endpoints](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Endpoints](#)): OK

201 ([Endpoints](#)): Created

202 ([Endpoints](#)): Accepted

401: Unauthorized

update replace the specified Endpoints

HTTP Request

PUT /api/v1/namespaces/{namespace}/endpoints/{name}

Parameters

- **name** (*in path*): string, required
name of the Endpoints
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Endpoints](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Endpoints](#)): OK

201 ([Endpoints](#)): Created

401: Unauthorized

patch partially update the specified Endpoints

HTTP Request

PATCH /api/v1/namespaces/{namespace}/endpoints/{name}

Parameters

- **name** (*in path*): string, required
name of the Endpoints
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required

- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Endpoints](#)): OK

201 ([Endpoints](#)): Created

401: Unauthorized

delete delete Endpoints

HTTP Request

DELETE /api/v1/namespaces/{namespace}/endpoints/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Endpoints
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)

- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Endpoints

HTTP Request

DELETE /api/v1/namespaces/{namespace}/endpoints

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)

- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

EndpointSlice

EndpointSlice represents a subset of the endpoints that implement a service.

apiVersion: discovery.k8s.io/v1

import "k8s.io/api/discovery/v1"

EndpointSlice

EndpointSlice represents a subset of the endpoints that implement a service. For a given service there may be multiple EndpointSlice objects, selected by labels, which must be joined to produce the full set of endpoints.

-
- **apiVersion**: discovery.k8s.io/v1
 - **kind**: EndpointSlice
 - **metadata** ([ObjectMeta](#))

Standard object's metadata.

addressType (string), required

- addressType specifies the type of address carried by this EndpointSlice. All addresses in this slice must be the same type. This field is immutable after creation. The following address types are currently supported:
 - * IPv4: Represents an IPv4 Address.
 - * IPv6: Represents an IPv6 Address.
 - * FQDN: Represents a Fully Qualified Domain Name.

• **endpoints** ([]Endpoint), required

Atomic: will be replaced during a merge

endpoints is a list of unique endpoints in this slice. Each slice may include a maximum of 1000 endpoints.

Endpoint represents a single logical "backend" implementing a service.

◦ **endpoints.addresses** ([]string), required

Set: unique values will be kept during a merge

addresses of this endpoint. The contents of this field are interpreted according to the corresponding EndpointSlice addressType field. Consumers must handle different types of addresses in the context of their own capabilities. This must contain at least one address but no more than 100. These are all assumed to be fungible and clients may choose to only use the first element. Refer to: <https://issue.k8s.io/106267>

◦ **endpoints.conditions** (EndpointConditions)

conditions contains information about the current status of the endpoint.

EndpointConditions represents the current condition of an endpoint.

▪ **endpoints.conditions.ready** (boolean)

ready indicates that this endpoint is prepared to receive traffic, according to whatever system is managing the endpoint. A nil value indicates an unknown state. In most cases consumers should interpret this unknown state as ready. For compatibility reasons, ready should never be "true" for terminating endpoints, except when the normal readiness behavior is being explicitly overridden, for example when the associated Service has set the publishNotReadyAddresses flag.

▪ **endpoints.conditions.serving** (boolean)

serving is identical to ready except that it is set regardless of the terminating state of endpoints. This condition should be set to true for a ready endpoint that is terminating. If nil, consumers should defer to the ready condition.

▪ **endpoints.conditions.terminating** (boolean)

terminating indicates that this endpoint is terminating. A nil value indicates an unknown state. Consumers should interpret this unknown state to mean that the endpoint is not terminating.

endpoints.deprecatedTopology (map[string]string)

- deprecatedTopology contains topology information part of the v1beta1 API. This field is deprecated, and will be removed when the v1beta1 API is removed (no sooner than kubernetes v1.24). While this field can hold values, it is not writable through the v1 API, and any attempts to write to it will be silently ignored. Topology information can be found in the zone and nodeName fields instead.

◦ **endpoints.hints** (EndpointHints)

hints contains information associated with how an endpoint should be consumed.

EndpointHints provides hints describing how an endpoint should be consumed.

▪ **endpoints.hints.forZones** ([]ForZone)

Atomic: will be replaced during a merge

forZones indicates the zone(s) this endpoint should be consumed by to enable topology aware routing.

ForZone provides information about which zones should consume this endpoint.

▪ **endpoints.hints.forZones.name** (string), required

name represents the name of the zone.

◦ **endpoints.hostname** (string)

hostname of this endpoint. This field may be used by consumers of endpoints to distinguish endpoints from each other (e.g. in DNS names). Multiple endpoints which use the same hostname should be considered fungible (e.g. multiple A values in DNS). Must be lowercase and pass DNS Label (RFC 1123) validation.

◦ **endpoints.nodeName** (string)

nodeName represents the name of the Node hosting this endpoint. This can be used to determine endpoints local to a Node.

◦ **endpoints.targetRef** ([ObjectReference](#))

targetRef is a reference to a Kubernetes object that represents this endpoint.

◦ **endpoints.zone** (string)

zone is the name of the Zone this endpoint exists in.

• **ports** ([]EndpointPort)

Atomic: will be replaced during a merge

ports specifies the list of network ports exposed by each endpoint in this slice. Each port must have a unique name. When ports is empty, it indicates that there are no defined ports. When a port is defined with a nil port value, it indicates "all ports". Each slice may include a maximum of 100 ports.

EndpointPort represents a Port used by an EndpointSlice

- **ports.port** (int32)

port represents the port number of the endpoint. If this is not specified, ports are not restricted and must be interpreted in the context of the specific consumer.

- **ports.protocol** (string)

protocol represents the IP protocol for this port. Must be UDP, TCP, or SCTP.
Default is TCP.

- **ports.name** (string)

name represents the name of this port. All ports in an EndpointSlice must have a unique name. If the EndpointSlice is derived from a Kubernetes service, this corresponds to the Service.ports[].name. Name must either be an empty string or pass DNS_LABEL validation: * must be no more than 63 characters long. * must consist of lower case alphanumeric characters or '-'. * must start and end with an alphanumeric character. Default is empty string.

- **ports.appProtocol** (string)

The application protocol for this port. This is used as a hint for implementations to offer richer behavior for protocols that they understand. This field follows standard Kubernetes label syntax. Valid values are either:

- Un-prefixed protocol names - reserved for IANA standard service names (as per RFC-6335 and <https://www.iana.org/assignments/service-names>).
- Kubernetes-defined prefixed names:
 - 'kubernetes.io/h2c' - HTTP/2 over cleartext as described in <https://www.rfc-editor.org/rfc/rfc7540>
 - 'kubernetes.io/ws' - WebSocket over cleartext as described in <https://www.rfc-editor.org/rfc/rfc6455>
 - 'kubernetes.io/wss' - WebSocket over TLS as described in <https://www.rfc-editor.org/rfc/rfc6455>
- Other protocols should use implementation-defined prefixed names such as mycompany.com/my-custom-protocol.

EndpointSliceList

EndpointSliceList represents a list of endpoint slices

- **apiVersion**: discovery.k8s.io/v1
- **kind**: EndpointSliceList
- **metadata** ([ListMeta](#))

Standard list metadata.

items ([][EndpointSlice](#)), required

- items is the list of endpoint slices

Operations

get read the specified EndpointSlice

HTTP Request

GET /apis/discovery.k8s.io/v1/namespaces/{namespace}/endpointslices/{name}

Parameters

- **name** (*in path*): string, required

name of the EndpointSlice

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([EndpointSlice](#)): OK

401: Unauthorized

list list or watch objects of kind EndpointSlice

HTTP Request

GET /apis/discovery.k8s.io/v1/namespaces/{namespace}/endpointslices

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([EndpointSliceList](#)): OK

401: Unauthorized

list list or watch objects of kind EndpointSlice

HTTP Request

GET /apis/discovery.k8s.io/v1/endpointslices

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([EndpointSliceList](#)): OK

401: Unauthorized

create create an EndpointSlice

HTTP Request

POST /apis/discovery.k8s.io/v1/namespaces/{namespace}/endpointslices

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [EndpointSlice](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([EndpointSlice](#)): OK

201 ([EndpointSlice](#)): Created

202 ([EndpointSlice](#)): Accepted

401: Unauthorized

update replace the specified EndpointSlice

HTTP Request

PUT /apis/discovery.k8s.io/v1/namespaces/{namespace}/endpointslices/{name}

Parameters

- **name** (*in path*): string, required
name of the EndpointSlice
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [EndpointSlice](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([EndpointSlice](#)): OK

201 ([EndpointSlice](#)): Created

401: Unauthorized

patch partially update the specified EndpointSlice

HTTP Request

PATCH /apis/discovery.k8s.io/v1/namespaces/{namespace}/endpointslices/{name}

Parameters

- **name** (*in path*): string, required
name of the EndpointSlice
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([EndpointSlice](#)): OK

201 ([EndpointSlice](#)): Created

401: Unauthorized

delete delete an EndpointSlice

HTTP Request

DELETE /apis/discovery.k8s.io/v1/namespaces/{namespace}/endpointslices/{name}

Parameters

- **name** (*in path*): string, required

name of the EndpointSlice

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of EndpointSlice

HTTP Request

DELETE /apis/discovery.k8s.io/v1/namespaces/{namespace}/endpointslices

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

sendInitialEvents (*in query*): boolean

- [sendInitialEvents](#)

timeoutSeconds (*in query*): integer

- [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Ingress

Ingress is a collection of rules that allow inbound connections to reach the endpoints defined by a backend.

apiVersion: networking.k8s.io/v1

import "k8s.io/api/networking/v1"

Ingress

Ingress is a collection of rules that allow inbound connections to reach the endpoints defined by a backend. An Ingress can be configured to give services externally-reachable urls, load balance traffic, terminate SSL, offer name based virtual hosting etc.

- **apiVersion**: networking.k8s.io/v1

- **kind**: Ingress

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([IngressSpec](#))

spec is the desired state of the Ingress. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([IngressStatus](#))

status is the current state of the Ingress. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

IngressSpec

IngressSpec describes the Ingress the user wishes to exist.

- **defaultBackend** ([IngressBackend](#))

defaultBackend is the backend that should handle requests that don't match any rule. If Rules are not specified, DefaultBackend must be specified. If DefaultBackend is not set, the handling of requests that do not match any of the rules will be up to the Ingress controller.

- **ingressClassName** (string)

ingressClassName is the name of an IngressClass cluster resource. Ingress controller implementations use this field to know whether they should be serving this Ingress resource, by a transitive connection (controller -> IngressClass -> Ingress resource). Although the kubernetes.io/ingress.class annotation (simple constant name) was never formally defined, it was widely supported by Ingress controllers to create a direct binding between Ingress controller and Ingress resources. Newly created Ingress resources should prefer using the field. However, even though the annotation is officially deprecated, for backwards compatibility reasons, ingress controllers should still honor that annotation if present.

- **rules** ([]IngressRule)

Atomic: will be replaced during a merge

rules is a list of host rules used to configure the Ingress. If unspecified, or no rule matches, all traffic is sent to the default backend.

IngressRule represents the rules mapping the paths under a specified host to the related backend services. Incoming requests are first evaluated for a host match, then routed to the backend associated with the matching IngressRuleValue.

- **rules.host** (string)

host is the fully qualified domain name of a network host, as defined by RFC 3986. Note the following deviations from the "host" part of the URI as defined in RFC 3986: 1. IPs are not allowed. Currently an IngressRuleValue can only apply to the IP in the Spec of the parent Ingress. 2. The : delimiter is not respected because ports are not allowed. Currently the port of an Ingress is implicitly :80 for http and :443 for https. Both these may change in the future. Incoming requests are matched against the host before the IngressRuleValue. If the host is unspecified, the Ingress routes all traffic based on the specified IngressRuleValue.

host can be "precise" which is a domain name without the terminating dot of a network host (e.g. "foo.bar.com") or "wildcard", which is a domain name prefixed with a single wildcard label (e.g. ".foo.com"). The wildcard character "`*`" must appear by itself as the first DNS label and matches only a single label. You cannot have a wildcard label by itself (e.g. Host == "`*`"). Requests will be matched against the Host field in the following way: 1. If host is precise, the request matches this rule if the http host header is equal to Host. 2. If host is a wildcard, then the request matches

this rule if the http host header is to equal to the suffix (removing the first label) of the wildcard rule.

- **rules.http** (`HTTPIngressRuleValue`)

HTTPIngressRuleValue is a list of http selectors pointing to backends. In the example: `http:///? -> backend` where parts of the url correspond to RFC 3986, this resource will be used to match against everything after the last '/' and before the first '?' or '#'.

- **rules.http.paths** (`[]HTTPIngressPath`), required

Atomic: will be replaced during a merge

paths is a collection of paths that map requests to backends.

HTTPIngressPath associates a path with a backend. Incoming urls matching the path are forwarded to the backend.

- **rules.http.paths.backend** ([IngressBackend](#)), required

backend defines the referenced service endpoint to which the traffic will be forwarded to.

- **rules.http.paths.pathType** (string), required

pathType determines the interpretation of the path matching.

PathType can be one of the following values:
* Exact: Matches the URL path exactly.
* Prefix: Matches based on a URL path prefix split by '/'. Matching is done on a path element by element basis. A path element refers to the list of labels in the path split by the '/' separator. A request is a match for path p if every p is an element-wise prefix of p of the request path. Note that if the last element of the path is a substring of the last element in request path, it is not a match (e.g. /foo/bar matches /foo/bar/baz, but does not match /foo/barbaz).

- ImplementationSpecific: Interpretation of the Path matching is up to the IngressClass. Implementations can treat this as a separate PathType or treat it identically to Prefix or Exact path types. Implementations are required to support all path types.

- **rules.http.paths.path** (string)

path is matched against the path of an incoming request. Currently it can contain characters disallowed from the conventional "path" part of a URL as defined by RFC 3986. Paths must begin with a '/' and must be present when using PathType with value "Exact" or "Prefix".

- **tls** (`[]IngressTLS`)

Atomic: will be replaced during a merge

tls represents the TLS configuration. Currently the Ingress only supports a single TLS port, 443. If multiple members of this list specify different hosts, they will be multiplexed on the same port according to the hostname specified through the SNI TLS extension, if the ingress controller fulfilling the ingress supports SNI.

IngressTLS describes the transport layer security associated with an ingress.

- **tls.hosts** ([]string)

Atomic: will be replaced during a merge

hosts is a list of hosts included in the TLS certificate. The values in this list must match the name/s used in the tlsSecret. Defaults to the wildcard host setting for the loadbalancer controller fulfilling this Ingress, if left unspecified.

- **tls.secretName** (string)

secretName is the name of the secret used to terminate TLS traffic on port 443. Field is left optional to allow TLS routing based on SNI hostname alone. If the SNI host in a listener conflicts with the "Host" header field used by an IngressRule, the SNI host is used for termination and value of the "Host" header is used for routing.

IngressBackend

IngressBackend describes all endpoints for a given service and port.

- **resource** ([TypedLocalObjectReference](#))

resource is an ObjectRef to another Kubernetes resource in the namespace of the Ingress object. If resource is specified, a service.Name and service.Port must not be specified. This is a mutually exclusive setting with "Service".

- **service** (IngressServiceBackend)

service references a service as a backend. This is a mutually exclusive setting with "Resource".

IngressServiceBackend references a Kubernetes Service as a Backend.

- **service.name** (string), required

name is the referenced service. The service must exist in the same namespace as the Ingress object.

- **service.port** (ServiceBackendPort)

port of the referenced service. A port name or port number is required for a IngressServiceBackend.

ServiceBackendPort is the service port being referenced.

- **service.port.name** (string)

name is the name of the port on the Service. This is a mutually exclusive setting with "Number".

- **service.port.number** (int32)

number is the numerical port number (e.g. 80) on the Service. This is a mutually exclusive setting with "Name".

IngressStatus

IngressStatus describe the current state of the Ingress.

- **loadBalancer** (IngressLoadBalancerStatus)

loadBalancer contains the current status of the load-balancer.

IngressLoadBalancerStatus represents the status of a load-balancer.

- **loadBalancer.ingress** ([]IngressLoadBalancerIngress)

ingress is a list containing ingress points for the load-balancer.

IngressLoadBalancerIngress represents the status of a load-balancer ingress point.

- **loadBalancer.ingress.hostname** (string)

hostname is set for load-balancer ingress points that are DNS based.

- **loadBalancer.ingress.ip** (string)

ip is set for load-balancer ingress points that are IP based.

- **loadBalancer.ingress.ports** ([]IngressPortStatus)

Atomic: will be replaced during a merge

ports provides information about the ports exposed by this LoadBalancer.

IngressPortStatus represents the error condition of a service port

- **loadBalancer.ingress.ports.port** (int32), required

port is the port number of the ingress port.

- **loadBalancer.ingress.ports.protocol** (string), required

protocol is the protocol of the ingress port. The supported values are: "TCP", "UDP", "SCTP"

- **loadBalancer.ingress.ports.error** (string)

error is to record the problem with the service port. The format of the error shall comply with the following rules: - built-in error values shall be specified in this file and those shall use CamelCase names

- cloud provider specific error values must have names that comply with the format foo.example.com/CamelCase.

IngressList

IngressList is a collection of Ingress.

- **items** ([][Ingress](#)), required

items is the list of Ingress.

- **apiVersion** (string)

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>

- **kind** (string)

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **metadata** ([ListMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

Operations

get read the specified Ingress

HTTP Request

GET /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses/{name}

Parameters

- **name** (*in path*): string, required

name of the Ingress

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Ingress](#)): OK

401: Unauthorized

get read status of the specified Ingress

HTTP Request

GET /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses/{name}/status

Parameters

- **name** (*in path*): string, required
name of the Ingress
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Ingress](#)): OK

401: Unauthorized

list list or watch objects of kind Ingress

HTTP Request

GET /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)

- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)
- **watch** (*in query*): boolean
 - [watch](#)

Response

200 ([IngressList](#)): OK

401: Unauthorized

list list or watch objects of kind Ingress

HTTP Request

GET /apis/networking.k8s.io/v1/ingresses

Parameters

- **allowWatchBookmarks** (*in query*): boolean
 - [allowWatchBookmarks](#)

- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([IngressList](#)): OK

401: Unauthorized

create create an Ingress

HTTP Request

POST /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Ingress](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Ingress](#)): OK

201 ([Ingress](#)): Created

202 ([Ingress](#)): Accepted

401: Unauthorized

update replace the specified Ingress

HTTP Request

PUT /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses/{name}

Parameters

- **name** (*in path*): string, required

name of the Ingress

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Ingress](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Ingress](#)): OK

201 ([Ingress](#)): Created

401: Unauthorized

update replace status of the specified Ingress

HTTP Request

PUT /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Ingress

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Ingress](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Ingress](#)): OK

201 ([Ingress](#)): Created

401: Unauthorized

patch partially update the specified Ingress

HTTP Request

PATCH /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses/{name}

Parameters

- **name** (*in path*): string, required

name of the Ingress

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Ingress](#)): OK

201 ([Ingress](#)): Created

401: Unauthorized

patch partially update status of the specified Ingress

HTTP Request

PATCH /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the Ingress
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Ingress](#)): OK

201 ([Ingress](#)): Created

401: Unauthorized

delete delete an Ingress

HTTP Request

DELETE /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses/{name}

Parameters

- **name** (*in path*): string, required
name of the Ingress
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Ingress

HTTP Request

DELETE /apis/networking.k8s.io/v1/namespaces/{namespace}/ingresses

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)

- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

IngressClass

IngressClass represents the class of the Ingress, referenced by the Ingress Spec.

```
apiVersion: networking.k8s.io/v1  
import "k8s.io/api/networking/v1"
```

IngressClass

IngressClass represents the class of the Ingress, referenced by the Ingress Spec. The `ingressclass.kubernetes.io/is-default-class` annotation can be used to indicate that an IngressClass should be considered default. When a single IngressClass resource has this annotation set to true, new Ingress resources without a class specified will be assigned this default class.

- **apiVersion**: networking.k8s.io/v1
- **kind**: IngressClass
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([IngressClassSpec](#))

spec is the desired state of the IngressClass. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

IngressClassSpec

IngressClassSpec provides information about the class of an Ingress.

- **controller** (string)

controller refers to the name of the controller that should handle this class. This allows for different "flavors" that are controlled by the same controller. For example, you may have different parameters for the same implementing controller. This should be specified as a domain-prefixed path no more than 250 characters in length, e.g. "acme.io/ingress-controller". This field is immutable.

- **parameters** ([IngressClassParametersReference](#))

parameters is a link to a custom resource containing additional configuration for the controller. This is optional if the controller does not require extra parameters.

IngressClassParametersReference identifies an API object. This can be used to specify a cluster or namespace-scoped resource.

- **parameters.kind** (string), required

kind is the type of resource being referenced.

- **parameters.name** (string), required

name is the name of resource being referenced.

- **parameters.apiGroup** (string)

apiGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.

- **parameters.namespace** (string)

namespace is the namespace of the resource being referenced. This field is required when scope is set to "Namespace" and must be unset when scope is set to "Cluster".

- **parameters.scope** (string)

scope represents if this refers to a cluster or namespace scoped resource. This may be set to "Cluster" (default) or "Namespace".

IngressClassList

IngressClassList is a collection of IngressClasses.

- **apiVersion**: networking.k8s.io/v1

- **kind**: IngressClassList

- **metadata** ([ListMeta](#))

Standard list metadata.

- **items** ([][IngressClass](#)), required

items is the list of IngressClasses.

Operations

get read the specified IngressClass

HTTP Request

GET /apis/networking.k8s.io/v1/ingressclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the IngressClass

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([IngressClass](#)): OK

401: Unauthorized

list list or watch objects of kind IngressClass

HTTP Request

GET /apis/networking.k8s.io/v1/ingressclasses

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([IngressClassList](#)): OK

401: Unauthorized

create create an IngressClass

HTTP Request

POST /apis/networking.k8s.io/v1/ingressclasses

Parameters

- **body**: [IngressClass](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([IngressClass](#)): OK

201 ([IngressClass](#)): Created

202 ([IngressClass](#)): Accepted

401: Unauthorized

update replace the specified IngressClass

HTTP Request

PUT /apis/networking.k8s.io/v1/ingressclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the IngressClass
- **body**: [IngressClass](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([IngressClass](#)): OK

201 ([IngressClass](#)): Created

401: Unauthorized

patch partially update the specified IngressClass

HTTP Request

PATCH /apis/networking.k8s.io/v1/ingressclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the IngressClass
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([IngressClass](#)): OK

201 ([IngressClass](#)): Created

401: Unauthorized

delete delete an IngressClass

HTTP Request

DELETE /apis/networking.k8s.io/v1/ingressclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the IngressClass
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of IngressClass

HTTP Request

DELETE /apis/networking.k8s.io/v1/ingressclasses

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Config and Storage Resources

[ConfigMap](#)

ConfigMap holds configuration data for pods to consume.

[Secret](#)

Secret holds secret data of a certain type.

[Volume](#)

Volume represents a named volume in a pod that may be accessed by any container in the pod.

[PersistentVolumeClaim](#)

PersistentVolumeClaim is a user's request for and claim to a persistent volume.

[PersistentVolume](#)

PersistentVolume (PV) is a storage resource provisioned by an administrator.

[StorageClass](#)

StorageClass describes the parameters for a class of storage for which PersistentVolumes can be dynamically provisioned.

[VolumeAttachment](#)

VolumeAttachment captures the intent to attach or detach the specified volume to/from the specified node.

[CSIDriver](#)

CSIDriver captures information about a Container Storage Interface (CSI) volume driver deployed on the cluster.

[CSINode](#)

CSINode holds information about all CSI drivers installed on a node.

[CSIStorageCapacity](#)

CSIStorageCapacity stores the result of one CSI GetCapacity call.

ConfigMap

ConfigMap holds configuration data for pods to consume.

```
apiVersion: v1
```

```
import "k8s.io/api/core/v1"
```

ConfigMap

ConfigMap holds configuration data for pods to consume.

- **apiVersion:** v1
- **kind:** ConfigMap
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **binaryData** (map[string][]byte)

BinaryData contains the binary data. Each key must consist of alphanumeric characters, '-', '_' or '.'. BinaryData can contain byte sequences that are not in the UTF-8 range. The keys stored in BinaryData must not overlap with the ones in the Data field, this is enforced during validation process. Using this field will require 1.10+ apiserver and kubelet.

- **data** (map[string]string)

Data contains the configuration data. Each key must consist of alphanumeric characters, '-', '_' or '.'. Values with non-UTF-8 byte sequences must use the BinaryData field. The keys stored in Data must not overlap with the keys in the BinaryData field, this is enforced during validation process.

- **immutable** (boolean)

Immutable, if set to true, ensures that data stored in the ConfigMap cannot be updated (only object metadata can be modified). If not set to true, the field can be modified at any time. Defaulted to nil.

ConfigMapList

ConfigMapList is a resource containing a list of ConfigMap objects.

- **apiVersion**: v1
- **kind**: ConfigMapList
- **metadata** ([ListMeta](#))

More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][ConfigMap](#)), required

Items is the list of ConfigMaps.

Operations

get read the specified ConfigMap

HTTP Request

GET /api/v1/namespaces/{namespace}/configmaps/{name}

Parameters

- **name** (*in path*): string, required
name of the ConfigMap
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ConfigMap](#)): OK

401: Unauthorized

list list or watch objects of kind ConfigMap

HTTP Request

GET /api/v1/namespaces/{namespace}/configmaps

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

timeoutSeconds (*in query*): integer

- [timeoutSeconds](#)

• **watch** (*in query*): boolean

[watch](#)

Response

200 ([ConfigMapList](#)): OK

401: Unauthorized

list list or watch objects of kind ConfigMap

HTTP Request

GET /api/v1/configmaps

Parameters

• **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

• **continue** (*in query*): string

[continue](#)

• **fieldSelector** (*in query*): string

[fieldSelector](#)

• **labelSelector** (*in query*): string

[labelSelector](#)

• **limit** (*in query*): integer

[limit](#)

• **pretty** (*in query*): string

[pretty](#)

• **resourceVersion** (*in query*): string

[resourceVersion](#)

• **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

sendInitialEvents (*in query*): boolean

-

[sendInitialEvents](#)

• **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

• **watch** (*in query*): boolean

[watch](#)

Response

200 ([ConfigMapList](#)): OK

401: Unauthorized

create create a ConfigMap

HTTP Request

POST /api/v1/namespaces/{namespace}/configmaps

Parameters

• **namespace** (*in path*): string, required

[namespace](#)

• **body**: [ConfigMap](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ConfigMap](#)): OK

201 ([ConfigMap](#)): Created

202 ([ConfigMap](#)): Accepted

401: Unauthorized

update replace the specified ConfigMap

HTTP Request

PUT /api/v1/namespaces/{namespace}/configmaps/{name}

Parameters

- **name** (*in path*): string, required

name of the ConfigMap

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ConfigMap](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ConfigMap](#)): OK

201 ([ConfigMap](#)): Created

401: Unauthorized

patch partially update the specified ConfigMap

HTTP Request

PATCH /api/v1/namespaces/{namespace}/configmaps/{name}

Parameters

- **name** (*in path*): string, required
name of the ConfigMap
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ConfigMap](#)): OK

201 ([ConfigMap](#)): Created

401: Unauthorized

delete delete a ConfigMap

HTTP Request

DELETE /api/v1/namespaces/{namespace}/configmaps/{name}

Parameters

- **name** (*in path*): string, required
name of the ConfigMap
- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ConfigMap

HTTP Request

DELETE /api/v1/namespaces/{namespace}/configmaps

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Secret

Secret holds secret data of a certain type.

```
apiVersion: v1
```

```
import "k8s.io/api/core/v1"
```

Secret

Secret holds secret data of a certain type. The total bytes of the values in the Data field must be less than MaxSecretSize bytes.

- **apiVersion:** v1

- **kind:** Secret

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **data** (map[string][]byte)

Data contains the secret data. Each key must consist of alphanumeric characters, '-', '_' or '..'. The serialized form of the secret data is a base64 encoded string, representing the arbitrary (possibly non-string) data value here. Described in <https://tools.ietf.org/html/rfc4648#section-4>

- **immutable** (boolean)

Immutable, if set to true, ensures that data stored in the Secret cannot be updated (only object metadata can be modified). If not set to true, the field can be modified at any time. Defaulted to nil.

- **stringData** (map[string]string)

stringData allows specifying non-binary secret data in string form. It is provided as a write-only input field for convenience. All keys and values are merged into the data field on write, overwriting any existing values. The stringData field is never output when reading from the API.

- **type** (string)

Used to facilitate programmatic handling of secret data. More info: <https://kubernetes.io/docs/concepts/configuration/secret/#secret-types>

SecretList

SecretList is a list of Secret.

- **apiVersion:** v1

- **kind:** SecretList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

items ([][Secret](#)), required

- Items is a list of secret objects. More info: <https://kubernetes.io/docs/concepts/configuration/secret>

Operations

get read the specified Secret

HTTP Request

GET /api/v1/namespaces/{namespace}/secrets/{name}

Parameters

- **name** (*in path*): string, required

name of the Secret

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Secret](#)): OK

401: Unauthorized

list list or watch objects of kind Secret

HTTP Request

GET /api/v1/namespaces/{namespace}/secrets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([SecretList](#)): OK

401: Unauthorized

list list or watch objects of kind Secret

HTTP Request

GET /api/v1/secrets

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([SecretList](#)): OK

401: Unauthorized

create create a Secret

HTTP Request

POST /api/v1/namespaces/{namespace}/secrets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Secret](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Secret](#)): OK

201 ([Secret](#)): Created

202 ([Secret](#)): Accepted

401: Unauthorized

update replace the specified Secret

HTTP Request

PUT /api/v1/namespaces/{namespace}/secrets/{name}

Parameters

- **name** (*in path*): string, required

name of the Secret

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Secret](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Secret](#)): OK

201 ([Secret](#)): Created

401: Unauthorized

patch partially update the specified Secret

HTTP Request

PATCH /api/v1/namespaces/{namespace}/secrets/{name}

Parameters

- **name** (*in path*): string, required

name of the Secret

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Secret](#)): OK

201 ([Secret](#)): Created

401: Unauthorized

delete delete a Secret

HTTP Request

DELETE /api/v1/namespaces/{namespace}/secrets/{name}

Parameters

- **name** (*in path*): string, required

name of the Secret

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Secret

HTTP Request

DELETE /api/v1/namespaces/{namespace}/secrets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Volume

Volume represents a named volume in a pod that may be accessed by any container in the pod.

import "k8s.io/api/core/v1"

Volume

Volume represents a named volume in a pod that may be accessed by any container in the pod.

- **name** (string), required

name of the volume. Must be a DNS_LABEL and unique within the pod. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

Exposed Persistent volumes

- **persistentVolumeClaim** (PersistentVolumeClaimVolumeSource)

persistentVolumeClaimVolumeSource represents a reference to a PersistentVolumeClaim in the same namespace. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims>

PersistentVolumeClaimVolumeSource references the user's PVC in the same namespace. This volume finds the bound PV and mounts that volume for the pod. A PersistentVolumeClaimVolumeSource is, essentially, a wrapper around another type of volume that is owned by someone else (the system).

- **persistentVolumeClaim.claimName** (string), required

claimName is the name of a PersistentVolumeClaim in the same namespace as the pod using this volume. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims>

- **persistentVolumeClaim.readOnly** (boolean)

readOnly Will force the ReadOnly setting in VolumeMounts. Default false.

Projections

- **configMap** (ConfigMapVolumeSource)

configMap represents a configMap that should populate this volume

*Adapts a ConfigMap into a volume.

The contents of the target ConfigMap's Data field will be presented in a volume as files using the keys in the Data field as the file names, unless the items element is populated with specific mappings of keys to paths. ConfigMap volumes support ownership management and SELinux relabeling.*

- **configMap.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **configMap.optional** (boolean)

optional specify whether the ConfigMap or its keys must be defined

- **configMap.defaultMode** (int32)

defaultMode is optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

- **configMap.items** ([]KeyToPath)

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

- **secret** (SecretVolumeSource)

secret represents a secret that should populate this volume. More info: <https://kubernetes.io/docs/concepts/storage/volumes#secret>

*Adapts a Secret into a volume.

The contents of the target Secret's Data field will be presented in a volume as files using the keys in the Data field as the file names. Secret volumes support ownership management and SELinux relabeling.*

- **secret.secretName** (string)

secretName is the name of the secret in the pod's namespace to use. More info: <https://kubernetes.io/docs/concepts/storage/volumes#secret>

secret.optional (boolean)

- optional field specify whether the Secret or its keys must be defined
- **secret.defaultMode** (int32)

defaultMode is Optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

- **secret.items** ([][KeyToPath](#))

items If unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '!'.

• **downwardAPI** (DownwardAPIVolumeSource)

downwardAPI represents downward API about the pod that should populate this volume

DownwardAPIVolumeSource represents a volume containing downward API info. Downward API volumes support ownership management and SELinux relabeling.

- **downwardAPI.defaultMode** (int32)

Optional: mode bits to use on created files by default. Must be a Optional: mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Defaults to 0644. Directories within the path are not affected by this setting. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

- **downwardAPI.items** ([][DownwardAPIVolumeFile](#))

Items is a list of downward API volume file

• **projected** (ProjectedVolumeSource)

projected items for all in one resources secrets, configmaps, and downward API

Represents a projected volume source

- **projected.defaultMode** (int32)

defaultMode are the mode bits used to set permissions on created files by default. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. Directories within the path are not affected by this setting. This might be

in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

- **projected.sources** ([]VolumeProjection)

sources is the list of volume projections

Projection that may be projected along with other supported volume types

- **projected.sources.configMap** (ConfigMapProjection)

configMap information about the configMap data to project

*Adapts a ConfigMap into a projected volume.

The contents of the target ConfigMap's Data field will be presented in a projected volume as files using the keys in the Data field as the file names, unless the items element is populated with specific mappings of keys to paths. Note that this is identical to a configmap volume source without the default mode.*

- **projected.sources.configMap.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **projected.sources.configMap.optional** (boolean)

optional specify whether the ConfigMap or its keys must be defined

- **projected.sources.configMap.items** ([]KeyToPath)

items if unspecified, each key-value pair in the Data field of the referenced ConfigMap will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the ConfigMap, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

- **projected.sources.downwardAPI** (DownwardAPIProjection)

downwardAPI information about the downwardAPI data to project

Represents downward API info for projecting into a projected volume. Note that this is identical to a downwardAPI volume source without the default mode.

- **projected.sources.downwardAPI.items** ([]DownwardAPIVolumeFile)

Items is a list of DownwardAPIVolume file

- **projected.sources.secret** (SecretProjection)

secret information about the secret data to project

*Adapts a secret into a projected volume.

The contents of the target Secret's Data field will be presented in a projected volume as files using the keys in the Data field as the file names. Note that this is identical to a secret volume source without the default mode.*

- **projected.sources.secret.name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

- **projected.sources.secret.optional** (boolean)

optional field specify whether the Secret or its key must be defined

- **projected.sources.secret.items** ([][KeyToPath](#))

items if unspecified, each key-value pair in the Data field of the referenced Secret will be projected into the volume as a file whose name is the key and content is the value. If specified, the listed keys will be projected into the specified paths, and unlisted keys will not be present. If a key is specified which is not present in the Secret, the volume setup will error unless it is marked optional. Paths must be relative and may not contain the '..' path or start with '..'.

- **projected.sources.serviceAccountToken**

(ServiceAccountTokenProjection)

serviceAccountToken is information about the serviceAccountToken data to project

ServiceAccountTokenProjection represents a projected service account token volume. This projection can be used to insert a service account token into the pods runtime filesystem for use against APIs (Kubernetes API Server or otherwise).

- **projected.sources.serviceAccountToken.path** (string), required

path is the path relative to the mount point of the file to project the token into.

- **projected.sources.serviceAccountToken.audience** (string)

audience is the intended audience of the token. A recipient of a token must identify itself with an identifier specified in the audience of the token, and otherwise should reject the token. The audience defaults to the identifier of the apiserver.

- **projected.sources.serviceAccountToken.expirationSeconds**
(int64)

expirationSeconds is the requested duration of validity of the service account token. As the token approaches expiration, the kubelet volume plugin will proactively rotate the service account token. The kubelet will start trying to rotate the token if the token is older than 80 percent

of its time to live or if the token is older than 24 hours. Defaults to 1 hour and must be at least 10 minutes.

Local / Temporary Directory

- **emptyDir** (EmptyDirVolumeSource)

emptyDir represents a temporary directory that shares a pod's lifetime. More info: <https://kubernetes.io/docs/concepts/storage/volumes#emptydir>

Represents an empty directory for a pod. Empty directory volumes support ownership management and SELinux relabeling.

- **emptyDir.medium** (string)

medium represents what type of storage medium should back this directory. The default is "" which means to use the node's default medium. Must be an empty string (default) or Memory. More info: <https://kubernetes.io/docs/concepts/storage/volumes#emptydir>

- **emptyDir.sizeLimit** ([Quantity](#))

sizeLimit is the total amount of local storage required for this EmptyDir volume. The size limit is also applicable for memory medium. The maximum usage on memory medium EmptyDir would be the minimum value between the SizeLimit specified here and the sum of memory limits of all containers in a pod. The default is nil which means that the limit is undefined. More info: <https://kubernetes.io/docs/concepts/storage/volumes#emptydir>

- **hostPath** (HostPathVolumeSource)

hostPath represents a pre-existing file or directory on the host machine that is directly exposed to the container. This is generally used for system agents or other privileged things that are allowed to see the host machine. Most containers will NOT need this. More info: <https://kubernetes.io/docs/concepts/storage/volumes#hostpath>

Represents a host path mapped into a pod. Host path volumes do not support ownership management or SELinux relabeling.

- **hostPath.path** (string), required

path of the directory on the host. If the path is a symlink, it will follow the link to the real path. More info: <https://kubernetes.io/docs/concepts/storage/volumes#hostpath>

- **hostPath.type** (string)

type for HostPath Volume Defaults to "" More info: <https://kubernetes.io/docs/concepts/storage/volumes#hostpath>

Persistent volumes

- **awsElasticBlockStore** (AWSElasticBlockStoreVolumeSource)

awsElasticBlockStore represents an AWS Disk resource that is attached to a kubelet's host machine and then exposed to the pod. More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

*Represents a Persistent Disk resource in AWS.

An AWS EBS disk must exist before mounting to a container. The disk must also be in the same AWS zone as the kubelet. An AWS EBS disk can only be mounted as read/write once. AWS EBS volumes support ownership management and SELinux relabeling.*

- **awsElasticBlockStore.volumeID** (string), required

volumeID is unique ID of the persistent disk resource in AWS (Amazon EBS volume). More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

- **awsElasticBlockStore.fsType** (string)

fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

- **awsElasticBlockStore.partition** (int32)

partition is the partition in the volume that you want to mount. If omitted, the default is to mount by volume name. Examples: For volume /dev/sda1, you specify the partition as "1". Similarly, the volume partition for /dev/sda is "0" (or you can leave the property empty).

- **awsElasticBlockStore.readOnly** (boolean)

readOnly value true will force the readOnly setting in VolumeMounts. More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

- **azureDisk** (AzureDiskVolumeSource)

azureDisk represents an Azure Data Disk mount on the host and bind mount to the pod.

AzureDisk represents an Azure Data Disk mount on the host and bind mount to the pod.

- **azureDisk.diskName** (string), required

diskName is the Name of the data disk in the blob storage

- **azureDisk.diskURI** (string), required

diskURI is the URI of data disk in the blob storage

- **azureDisk.cachingMode** (string)

cachingMode is the Host Caching mode: None, Read Only, Read Write.

- **azureDisk.fsType** (string)

fsType is Filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **azureDisk.kind** (string)

kind expected values are Shared: multiple blob disks per storage account Dedicated: single blob disk per storage account Managed: azure managed data disk (only in managed availability set). defaults to shared

- **azureDisk.readOnly** (boolean)

readOnly Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **azureFile** (AzureFileVolumeSource)

azureFile represents an Azure File Service mount on the host and bind mount to the pod.

AzureFile represents an Azure File Service mount on the host and bind mount to the pod.

- **azureFile.secretName** (string), required

secretName is the name of secret that contains Azure Storage Account Name and Key

- **azureFile.shareName** (string), required

shareName is the azure share Name

- **azureFile.readOnly** (boolean)

readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **cephfs** (CephFSVolumeSource)

cephFS represents a Ceph FS mount on the host that shares a pod's lifetime

Represents a Ceph Filesystem mount that lasts the lifetime of a pod Cephfs volumes do not support ownership management or SELinux relabeling.

- **cephfs.monitors** ([]string), required

monitors is Required: Monitors is a collection of Ceph monitors More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>

- **cephfs.path** (string)

path is Optional: Used as the mounted root, rather than the full Ceph tree, default is /

- **cephfs.readOnly** (boolean)

readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts. More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>

- **cephfs.secretFile** (string)

secretFile is Optional: SecretFile is the path to key ring for User, default is /etc/ceph/user.secret More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>

- **cephfs.secretRef** ([LocalObjectReference](#))

secretRef is Optional: SecretRef is reference to the authentication secret for User, default is empty. More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>

- **cephfs.user** (string)

user is optional: User is the rados user name, default is admin More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>

- **cinder** ([CinderVolumeSource](#))

cinder represents a cinder volume attached and mounted on kubelets host machine. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

Represents a cinder volume resource in Openstack. A Cinder volume must exist before mounting to a container. The volume must also be in the same region as the kubelet. Cinder volumes support ownership management and SELinux relabeling.

- **cinder.volumeID** (string), required

volumeID used to identify the volume in cinder. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

- **cinder.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

- **cinder.readOnly** (boolean)

readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

- **cinder.secretRef** ([LocalObjectReference](#))

secretRef is optional: points to a secret object containing parameters used to connect to OpenStack.

- **csi** ([CSIVolumeSource](#))

csi (Container Storage Interface) represents ephemeral storage that is handled by certain external CSI drivers (Beta feature).

Represents a source location of a volume to mount, managed by an external CSI driver

- **csi.driver** (string), required

driver is the name of the CSI driver that handles this volume. Consult with your admin for the correct name as registered in the cluster.

- **csi.fsType** (string)

fsType to mount. Ex. "ext4", "xfs", "ntfs". If not provided, the empty value is passed to the associated CSI driver which will determine the default filesystem to apply.

- **csi.nodePublishSecretRef** ([LocalObjectReference](#))

nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secret references are passed.

- **csi.readOnly** (boolean)

readOnly specifies a read-only configuration for the volume. Defaults to false (read/write).

- **csi.volumeAttributes** (map[string]string)

volumeAttributes stores driver-specific properties that are passed to the CSI driver. Consult your driver's documentation for supported values.

- **ephemeral** (EphemeralVolumeSource)

ephemeral represents a volume that is handled by a cluster storage driver. The volume's lifecycle is tied to the pod that defines it - it will be created before the pod starts, and deleted when the pod is removed.

Use this if: a) the volume is only needed while the pod runs, b) features of normal volumes like restoring from snapshot or capacity tracking are needed, c) the storage driver is specified through a storage class, and d) the storage driver supports dynamic volume provisioning through a PersistentVolumeClaim (see EphemeralVolumeSource for more information on the connection between this volume type and PersistentVolumeClaim).

Use PersistentVolumeClaim or one of the vendor-specific APIs for volumes that persist for longer than the lifecycle of an individual pod.

Use CSI for light-weight local ephemeral volumes if the CSI driver is meant to be used that way - see the documentation of the driver for more information.

A pod can use both types of ephemeral volumes and persistent volumes at the same time.

Represents an ephemeral volume that is handled by a normal storage driver.

- **ephemeral.volumeClaimTemplate** ([PersistentVolumeClaimTemplate](#))

Will be used to create a stand-alone PVC to provision the volume. The pod in which this EphemeralVolumeSource is embedded will be the owner of the PVC, i.e. the PVC will be deleted together with the pod. The name of the PVC will be \<pod name>-<volume name> where \<volume name> is the name from the PodSpec.Volumes array entry. Pod validation will reject the pod if the concatenated name is not valid for a PVC (for example, too long).

An existing PVC with that name that is not owned by the pod will *not* be used for the pod to avoid using an unrelated volume by mistake. Starting the pod is then blocked until the unrelated PVC is removed. If such a pre-created PVC is meant to be used by the pod, the PVC has to be updated with an owner reference to the pod once the pod exists. Normally this should not be necessary, but it may be useful when manually reconstructing a broken cluster.

This field is read-only and no changes will be made by Kubernetes to the PVC after it has been created.

Required, must not be nil.

PersistentVolumeClaimTemplate is used to produce PersistentVolumeClaim objects as part of an EphemeralVolumeSource.

- **ephemeral.volumeClaimTemplate.spec** ([PersistentVolumeClaimSpec](#)), required

The specification for the PersistentVolumeClaim. The entire content is copied unchanged into the PVC that gets created from this template. The same fields as in a PersistentVolumeClaim are also valid here.

- **ephemeral.volumeClaimTemplate.metadata** ([ObjectMeta](#))

May contain labels and annotations that will be copied into the PVC when creating it. No other fields are allowed and will be rejected during validation.

- **fc** ([FCVolumeSource](#))

fc represents a Fibre Channel resource that is attached to a kubelet's host machine and then exposed to the pod.

Represents a Fibre Channel volume. Fibre Channel volumes can only be mounted as read/write once. Fibre Channel volumes support ownership management and SELinux relabeling.

- **fc.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **fc.lun** (int32)

lun is Optional: FC target lun number

fc.readOnly (boolean)

- readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

◦ **fc.targetWWNs** ([]string)

targetWWNs is Optional: FC target worldwide names (WWNs)

◦ **fc.wwids** ([]string)

wwids Optional: FC volume world wide identifiers (wwids) Either wwid or combination of targetWWNs and lun must be set, but not both simultaneously.

• **flexVolume** (FlexVolumeSource)

flexVolume represents a generic volume resource that is provisioned/attached using an exec based plugin.

FlexVolume represents a generic volume resource that is provisioned/attached using an exec based plugin.

◦ **flexVolume.driver** (string), required

driver is the name of the driver to use for this volume.

◦ **flexVolume.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". The default filesystem depends on FlexVolume script.

◦ **flexVolume.options** (map[string]string)

options is Optional: this field holds extra command options if any.

◦ **flexVolume.readOnly** (boolean)

readOnly is Optional: defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

◦ **flexVolume.secretRef** ([LocalObjectReference](#))

secretRef is Optional: secretRef is reference to the secret object containing sensitive information to pass to the plugin scripts. This may be empty if no secret object is specified. If the secret object contains more than one secret, all secrets are passed to the plugin scripts.

• **flocker** (FlockerVolumeSource)

flocker represents a Flocker volume attached to a kubelet's host machine. This depends on the Flocker control service being running

Represents a Flocker volume mounted by the Flocker agent. One and only one of datasetName and datasetUUID should be set. Flocker volumes do not support ownership management or SELinux relabeling.

- **flocker.datasetName** (string)

datasetName is Name of the dataset stored as metadata -> name on the dataset for Flocker should be considered as deprecated

- **flocker.datasetUUID** (string)

datasetUUID is the UUID of the dataset. This is unique identifier of a Flocker dataset

- **gcePersistentDisk** (GCEPersistentDiskVolumeSource)

gcePersistentDisk represents a GCE Disk resource that is attached to a kubelet's host machine and then exposed to the pod. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

*Represents a Persistent Disk resource in Google Compute Engine.

A GCE PD must exist before mounting to a container. The disk must also be in the same GCE project and zone as the kubelet. A GCE PD can only be mounted as read/write once or read-only many times. GCE PDs support ownership management and SELinux relabeling.*

- **gcePersistentDisk.pdName** (string), required

pdName is unique name of the PD resource in GCE. Used to identify the disk in GCE. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **gcePersistentDisk.fsType** (string)

fsType is filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **gcePersistentDisk.partition** (int32)

partition is the partition in the volume that you want to mount. If omitted, the default is to mount by volume name. Examples: For volume /dev/sda1, you specify the partition as "1". Similarly, the volume partition for /dev/sda is "0" (or you can leave the property empty). More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **gcePersistentDisk.readOnly** (boolean)

readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **glusterfs** (GlusterfsVolumeSource)

glusterfs represents a Glusterfs mount on the host that shares a pod's lifetime. More info: <https://examples.k8s.io/volumes/glusterfs/README.md>

Represents a Glusterfs mount that lasts the lifetime of a pod. Glusterfs volumes do not support ownership management or SELinux relabeling.

- **glusterfs.endpoints** (string), required

endpoints is the endpoint name that details Glusterfs topology. More info: <https://examples.k8s.io/volumes/glusterfs/README.md#create-a-pod>

- **glusterfs.path** (string), required

path is the Glusterfs volume path. More info: <https://examples.k8s.io/volumes/glusterfs/README.md#create-a-pod>

- **glusterfs.readOnly** (boolean)

readOnly here will force the Glusterfs volume to be mounted with read-only permissions. Defaults to false. More info: <https://examples.k8s.io/volumes/glusterfs/README.md#create-a-pod>

- **iscsi** (ISCSIVolumeSource)

iscsi represents an ISCSI Disk resource that is attached to a kubelet's host machine and then exposed to the pod. More info: <https://examples.k8s.io/volumes/iscsi/README.md>

Represents an ISCSI disk. ISCSI volumes can only be mounted as read/write once. ISCSI volumes support ownership management and SELinux relabeling.

- **iscsi.iqn** (string), required

iqn is the target iSCSI Qualified Name.

- **iscsi.lun** (int32), required

lun represents iSCSI Target Lun number.

- **iscsi.targetPortal** (string), required

targetPortal is iSCSI Target Portal. The Portal is either an IP or ip_addr:port if the port is other than default (typically TCP ports 860 and 3260).

- **iscsi.chapAuthDiscovery** (boolean)

chapAuthDiscovery defines whether support iSCSI Discovery CHAP authentication

- **iscsi.chapAuthSession** (boolean)

chapAuthSession defines whether support iSCSI Session CHAP authentication

- **iscsi.fsType** (string)

fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples:

"ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#iscsi>

- **iscsi.initiatorName** (string)

initiatorName is the custom iSCSI Initiator Name. If initiatorName is specified with iscsiInterface simultaneously, new iSCSI interface <target portal>:<volume name> will be created for the connection.

- **iscsi.iscsiInterface** (string)

iscsiInterface is the interface Name that uses an iSCSI transport. Defaults to 'default' (tcp).

- **iscsi.portals** ([]string)

portals is the iSCSI Target Portal List. The portal is either an IP or ip_addr:port if the port is other than default (typically TCP ports 860 and 3260).

- **iscsi.readOnly** (boolean)

readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false.

- **iscsi.secretRef** ([LocalObjectReference](#))

secretRef is the CHAP Secret for iSCSI target and initiator authentication

- **nfs** (NFSVolumeSource)

nfs represents an NFS mount on the host that shares a pod's lifetime More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

Represents an NFS mount that lasts the lifetime of a pod. NFS volumes do not support ownership management or SELinux relabeling.

- **nfs.path** (string), required

path that is exported by the NFS server. More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

- **nfs.server** (string), required

server is the hostname or IP address of the NFS server. More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

- **nfs.readOnly** (boolean)

readOnly here will force the NFS export to be mounted with read-only permissions. Defaults to false. More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

- **photonPersistentDisk** (PhotonPersistentDiskVolumeSource)

photonPersistentDisk represents a PhotonController persistent disk attached and mounted on kubelets host machine

Represents a Photon Controller persistent disk resource.

- **photonPersistentDisk.pdID** (string), required

pdID is the ID that identifies Photon Controller persistent disk

- **photonPersistentDisk.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **portworxVolume** (PortworxVolumeSource)

portworxVolume represents a portworx volume attached and mounted on kubelets host machine

PortworxVolumeSource represents a Portworx volume resource.

- **portworxVolume.volumeID** (string), required

volumeID uniquely identifies a Portworx volume

- **portworxVolume.fsType** (string)

fSType represents the filesystem type to mount Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs". Implicitly inferred to be "ext4" if unspecified.

- **portworxVolume.readOnly** (boolean)

readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **quobyte** (QuobyteVolumeSource)

quobyte represents a Quobyte mount on the host that shares a pod's lifetime

Represents a Quobyte mount that lasts the lifetime of a pod. Quobyte volumes do not support ownership management or SELinux relabeling.

- **quobyte.registry** (string), required

registry represents a single or multiple Quobyte Registry services specified as a string as host:port pair (multiple entries are separated with commas) which acts as the central registry for volumes

- **quobyte.volume** (string), required

volume is a string that references an already created Quobyte volume by name.

- **quobyte.group** (string)

group to map volume access to Default is no group

- **quobyte.readOnly** (boolean)

readOnly here will force the Quobyte volume to be mounted with read-only permissions. Defaults to false.

- **quobyte.tenant** (string)

tenant owning the given Quobyte volume in the Backend Used with dynamically provisioned Quobyte volumes, value is set by the plugin

- **quobyte.user** (string)

user to map volume access to Defaults to serviceaccount user

- **rbd** (RBDVolumeSource)

rbd represents a Rados Block Device mount on the host that shares a pod's lifetime. More info: <https://examples.k8s.io/volumes/rbd/README.md>

Represents a Rados Block Device mount that lasts the lifetime of a pod. RBD volumes support ownership management and SELinux relabeling.

- **rbd.image** (string), required

image is the rados image name. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.monitors** ([]string), required

monitors is a collection of Ceph monitors. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.fsType** (string)

fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#rbd>

- **rbd.keyring** (string)

keyring is the path to key ring for RBDUser. Default is /etc/ceph/keyring. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.pool** (string)

pool is the rados pool name. Default is rbd. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.readOnly** (boolean)

readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.secretRef** ([LocalObjectReference](#))

secretRef is name of the authentication secret for RBDUser. If provided overrides keyring. Default is nil. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.user** (string)

user is the rados user name. Default is admin. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **scaleIO** (ScaleIOPersistentVolumeSource)

scaleIO represents a ScaleIO persistent volume attached and mounted on Kubernetes nodes.

ScaleIOPersistentVolumeSource represents a persistent ScaleIO volume

- **scaleIO.gateway** (string), required

gateway is the host address of the ScaleIO API Gateway.

- **scaleIO.secretRef** ([LocalObjectReference](#)), required

secretRef references to the secret for ScaleIO user and other sensitive information. If this is not provided, Login operation will fail.

- **scaleIO.system** (string), required

system is the name of the storage system as configured in ScaleIO.

- **scaleIO.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Default is "xfs".

- **scaleIO.protectionDomain** (string)

protectionDomain is the name of the ScaleIO Protection Domain for the configured storage.

- **scaleIO.readOnly** (boolean)

readOnly Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **scaleIO.sslEnabled** (boolean)

sslEnabled Flag enable/disable SSL communication with Gateway, default false

- **scaleIO.storageMode** (string)

storageMode indicates whether the storage for a volume should be ThickProvisioned or ThinProvisioned. Default is ThinProvisioned.

- **scaleIO.storagePool** (string)

storagePool is the ScaleIO Storage Pool associated with the protection domain.

scaleIO.volumeName (string)

- volumeName is the name of a volume already created in the ScaleIO system that is associated with this volume source.

• **storageos** (StorageOSVolumeSource)

storageOS represents a StorageOS volume attached and mounted on Kubernetes nodes.

Represents a StorageOS persistent volume resource.

◦ **storageos.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

◦ **storageos.readOnly** (boolean)

readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

◦ **storageos.secretRef** ([LocalObjectReference](#))

secretRef specifies the secret to use for obtaining the StorageOS API credentials. If not specified, default values will be attempted.

◦ **storageos.volumeName** (string)

volumeName is the human-readable name of the StorageOS volume. Volume names are only unique within a namespace.

◦ **storageos.volumeNamespace** (string)

volumeNamespace specifies the scope of the volume within StorageOS. If no namespace is specified then the Pod's namespace will be used. This allows the Kubernetes name scoping to be mirrored within StorageOS for tighter integration. Set VolumeName to any name to override the default behaviour. Set to "default" if you are not using namespaces within StorageOS. Namespaces that do not pre-exist within StorageOS will be created.

• **vsphereVolume** (VsphereVirtualDiskVolumeSource)

vsphereVolume represents a vSphere volume attached and mounted on kubelets host machine

Represents a vSphere volume resource.

◦ **vsphereVolume.volumePath** (string), required

volumePath is the path that identifies vSphere volume vmdk

◦ **vsphereVolume.fsType** (string)

`fsType` is filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **vsphereVolume.storagePolicyID** (string)

`storagePolicyID` is the storage Policy Based Management (SPBM) profile ID associated with the `StoragePolicyName`.

- **vsphereVolume.storagePolicyName** (string)

`storagePolicyName` is the storage Policy Based Management (SPBM) profile name.

Deprecated

- **gitRepo** (GitRepoVolumeSource)

`gitRepo` represents a git repository at a particular revision. DEPRECATED: `GitRepo` is deprecated. To provision a container with a git repo, mount an `EmptyDir` into an `InitContainer` that clones the repo using git, then mount the `EmptyDir` into the Pod's container.

*Represents a volume that is populated with the contents of a git repository. Git repo volumes do not support ownership management. Git repo volumes support SELinux relabeling.

DEPRECATED: `GitRepo` is deprecated. To provision a container with a git repo, mount an `EmptyDir` into an `InitContainer` that clones the repo using git, then mount the `EmptyDir` into the Pod's container.*

- **gitRepo.repository** (string), required

`repository` is the URL

- **gitRepo.directory** (string)

`directory` is the target directory name. Must not contain or start with '..'. If '.' is supplied, the volume directory will be the git repository. Otherwise, if specified, the volume will contain the git repository in the subdirectory with the given name.

- **gitRepo.revision** (string)

`revision` is the commit hash for the specified revision.

DownwardAPIVolumeFile

`DownwardAPIVolumeFile` represents information to create the file containing the pod field

- **path** (string), required

Required: Path is the relative path name of the file to be created. Must not be absolute or contain the '.' path. Must be utf-8 encoded. The first item of the relative path must not start with '..'

fieldRef ([ObjectFieldSelector](#))

- Required: Selects a field of the pod: only annotations, labels, name and namespace are supported.
- **mode** (int32)

Optional: mode bits used to set permissions on this file, must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

resourceFieldRef ([ResourceFieldSelector](#))

Selects a resource of the container: only resources limits and requests (limits.cpu, limits.memory, requests.cpu and requests.memory) are currently supported.

KeyToPath

Maps a string key to a path within a volume.

- **key** (string), required

key is the key to project.

- **path** (string), required

path is the relative path of the file to map the key to. May not be an absolute path. May not contain the path element '..'. May not start with the string '!'.

- **mode** (int32)

mode is Optional: mode bits used to set permissions on this file. Must be an octal value between 0000 and 0777 or a decimal value between 0 and 511. YAML accepts both octal and decimal values, JSON requires decimal values for mode bits. If not specified, the volume defaultMode will be used. This might be in conflict with other options that affect the file mode, like fsGroup, and the result can be other mode bits set.

PersistentVolumeClaim

PersistentVolumeClaim is a user's request for and claim to a persistent volume.

```
apiVersion: v1
```

```
import "k8s.io/api/core/v1"
```

PersistentVolumeClaim

PersistentVolumeClaim is a user's request for and claim to a persistent volume

-
- **apiVersion:** v1
 - **kind:** PersistentVolumeClaim
 - **metadata** ([ObjectMeta](#))
Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>
 - **spec** ([PersistentVolumeClaimSpec](#))
spec defines the desired characteristics of a volume requested by a pod author. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims>
 - **status** ([PersistentVolumeClaimStatus](#))
status represents the current information/status of a persistent volume claim. Read-only. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims>

PersistentVolumeClaimSpec

PersistentVolumeClaimSpec describes the common attributes of storage devices and allows a Source for provider-specific attributes

- **accessModes** ([]string)
accessModes contains the desired access modes the volume should have. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1>
- **selector** ([LabelSelector](#))
selector is a label query over volumes to consider for binding.
- **resources** ([ResourceRequirements](#))
resources represents the minimum resources the volume should have. If RecoverVolumeExpansionFailure feature is enabled users are allowed to specify resource requirements that are lower than previous value but must still be higher than capacity recorded in the status field of the claim. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#resources>

ResourceRequirements describes the compute resource requirements.

- **resources.claims** ([]ResourceClaim)

Map: unique values on key name will be kept during a merge

Claims lists the names of resources, defined in spec.resourceClaims, that are used by this container.

This is an alpha field and requires enabling the DynamicResourceAllocation feature gate.

This field is immutable. It can only be set for containers.

ResourceClaim references one entry in PodSpec.ResourceClaims.

- **resources.claims.name** (string), required

Name must match the name of one entry in pod.spec.resourceClaims of the Pod where this field is used. It makes that resource available inside a container.

- **resources.limits** (map[string][Quantity](#))

Limits describes the maximum amount of compute resources allowed. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

- **resources.requests** (map[string][Quantity](#))

Requests describes the minimum amount of compute resources required. If Requests is omitted for a container, it defaults to Limits if that is explicitly specified, otherwise to an implementation-defined value. Requests cannot exceed Limits. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

- **volumeName** (string)

volumeName is the binding reference to the PersistentVolume backing this claim.

- **storageClassName** (string)

storageClassName is the name of the StorageClass required by the claim. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#class-1>

- **volumeMode** (string)

volumeMode defines what type of volume is required by the claim. Value of Filesystem is implied when not included in claim spec.

Beta level

- **dataSource** ([TypedLocalObjectReference](#))

dataSource field can be used to specify either:

- * An existing VolumeSnapshot object (snapshot.storage.k8s.io/VolumeSnapshot)
- * An existing PVC (PersistentVolumeClaim) If the provisioner or an external controller can support the specified data source, it will create a new volume based on the contents of the specified data source. When the AnyVolumeDataSource feature gate is enabled, dataSource contents will be copied to dataSourceRef, and dataSourceRef contents will be copied to dataSource when dataSourceRef.namespace is not specified. If the namespace is specified, then dataSourceRef will not be copied to dataSource.

- **dataSourceRef** ([TypedObjectReference](#))

dataSourceRef specifies the object from which to populate the volume with data, if a non-empty volume is desired. This may be any object from a non-empty API group (non core object) or a PersistentVolumeClaim object. When this field is specified, volume binding will only succeed if the type of the specified object matches some installed volume populator or dynamic provisioner. This field will replace the functionality of the dataSource field and as such if both fields are non-empty, they must have the same value. For backwards compatibility, when namespace isn't specified in dataSourceRef, both fields (dataSource and dataSourceRef) will be set to the same value automatically if one of them is empty and the other is non-empty. When namespace is specified in dataSourceRef, dataSource isn't set to the same value and must be empty. There are three important differences between dataSource and dataSourceRef: * While dataSource only allows two specific types of objects, dataSourceRef allows any non-core object, as well as PersistentVolumeClaim objects.

- While dataSource ignores disallowed values (dropping them), dataSourceRef preserves all values, and generates an error if a disallowed value is specified.
- While dataSource only allows local objects, dataSourceRef allows objects in any namespaces. (Beta) Using this field requires the AnyVolumeDataSource feature gate to be enabled. (Alpha) Using the namespace field of dataSourceRef requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

**

- **dataSourceRef.kind** (string), required

Kind is the type of resource being referenced

- **dataSourceRef.name** (string), required

Name is the name of resource being referenced

- **dataSourceRef.apiGroup** (string)

APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.

- **dataSourceRef.namespace** (string)

Namespace is the namespace of resource being referenced. Note that when a namespace is specified, a gateway.networking.k8s.io/ReferenceGrant object is required in the referent namespace to allow that namespace's owner to accept the reference. See the ReferenceGrant documentation for details. (Alpha) This field requires the CrossNamespaceVolumeDataSource feature gate to be enabled.

PersistentVolumeClaimStatus

PersistentVolumeClaimStatus is the current status of a persistent volume claim.

-
- **accessModes** ([]string)

accessModes contains the actual access modes the volume backing the PVC has. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes-1>

allocatedResourceStatuses (map[string]string)

- allocatedResourceStatuses stores status of resource being resized for the given PVC. Key names follow standard Kubernetes label syntax. Valid values are either:
* Un-prefixed keys: - storage - the capacity of the volume.
* Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource"
Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used.

ClaimResourceStatus can be in any of following states:

- ControllerResizeInProgress: State set when resize controller starts resizing the volume in control-plane.
- ControllerResizeFailed: State set when resize has failed in resize controller with a terminal error.
- NodeResizePending: State set when resize controller has finished resizing the volume but further resizing of volume is needed on the node.
- NodeResizeInProgress: State set when kubelet starts resizing the volume.
- NodeResizeFailed: State set when resizing has failed in kubelet with a terminal error. Transient errors don't set NodeResizeFailed. For example: if expanding a PVC for more capacity - this field can be one of the following states:
- pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeInProgress"
- pvc.status.allocatedResourceStatus['storage'] = "ControllerResizeFailed"
- pvc.status.allocatedResourceStatus['storage'] = "NodeResizePending"
- pvc.status.allocatedResourceStatus['storage'] = "NodeResizeInProgress"
- pvc.status.allocatedResourceStatus['storage'] = "NodeResizeFailed"

When this field is not set, it means that no resize operation is in progress for the given PVC.

A controller that receives PVC update with previously unknown resourceName or ClaimResourceStatus should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC.

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

allocatedResources (map[string][Quantity](#))

allocatedResources tracks the resources allocated to a PVC including its capacity. Key names follow standard Kubernetes label syntax. Valid values are either:
* Un-prefixed keys: - storage - the capacity of the volume.
* Custom resources must use implementation-defined prefixed names such as "example.com/my-custom-resource"
Apart from above values - keys that are unprefixed or have kubernetes.io prefix are considered reserved and hence may not be used.

Capacity reported here may be larger than the actual capacity when a volume expansion operation is requested. For storage quota, the larger value from allocatedResources and PVC.spec.resources is used. If allocatedResources is not set, PVC.spec.resources alone is used for quota calculation. If a volume expansion capacity request is lowered, allocatedResources is only lowered if there are no expansion operations in progress and if the actual volume capacity is equal or lower than the requested capacity.

A controller that receives PVC update with previously unknown resourceName should ignore the update for the purpose it was designed. For example - a controller that only is responsible for resizing capacity of the volume, should ignore PVC updates that change other valid resources associated with PVC.

This is an alpha field and requires enabling RecoverVolumeExpansionFailure feature.

- **capacity** (map[string][Quantity](#))

- capacity represents the actual resources of the underlying volume.

- **conditions** ([]PersistentVolumeClaimCondition)

Patch strategy: merge on key type

conditions is the current Condition of persistent volume claim. If underlying persistent volume is being resized then the Condition will be set to 'ResizeStarted'.

PersistentVolumeClaimCondition contains details about state of pvc

- **conditions.status** (string), required

- **conditions.type** (string), required

- **conditions.lastProbeTime** (Time)

lastProbeTime is the time we probed the condition.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.lastTransitionTime** (Time)

lastTransitionTime is the time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

message is the human-readable message indicating details about last transition.

- **conditions.reason** (string)

reason is a unique, this should be a short, machine understandable string that gives the reason for condition's last transition. If it reports "ResizeStarted" that means the underlying persistent volume is being resized.

- **phase** (string)

phase represents the current phase of PersistentVolumeClaim.

PersistentVolumeClaimList

PersistentVolumeClaimList is a list of PersistentVolumeClaim items.

- **apiVersion:** v1

kind: PersistentVolumeClaimList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][PersistentVolumeClaim](#)), required

items is a list of persistent volume claims. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistentvolumeclaims>

Operations

get read the specified PersistentVolumeClaim

HTTP Request

GET /api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}

Parameters

- **name** (*in path*): string, required

name of the PersistentVolumeClaim

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PersistentVolumeClaim](#)): OK

401: Unauthorized

get read status of the specified PersistentVolumeClaim

HTTP Request

GET /api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}/status

Parameters

- **name** (*in path*): string, required

name of the PersistentVolumeClaim

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PersistentVolumeClaim](#)): OK

401: Unauthorized

list list or watch objects of kind PersistentVolumeClaim

HTTP Request

GET /api/v1/namespaces/{namespace}/persistentvolumeclaims

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([PersistentVolumeClaimList](#)): OK

401: Unauthorized

list list or watch objects of kind PersistentVolumeClaim

HTTP Request

GET /api/v1/persistentvolumeclaims

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([PersistentVolumeClaimList](#)): OK

401: Unauthorized

create create a PersistentVolumeClaim

HTTP Request

POST /api/v1/namespaces/{namespace}/persistentvolumeclaims

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [PersistentVolumeClaim](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([PersistentVolumeClaim](#)): OK

201 ([PersistentVolumeClaim](#)): Created

202 ([PersistentVolumeClaim](#)): Accepted

401: Unauthorized

update replace the specified PersistentVolumeClaim

HTTP Request

PUT /api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}

Parameters

- **name** (*in path*): string, required

- name of the PersistentVolumeClaim

- **namespace** (*in path*): string, required

- [namespace](#)

- **body**: [PersistentVolumeClaim](#), required

- **dryRun** (*in query*): string

- [dryRun](#)

- **fieldManager** (*in query*): string

- [fieldManager](#)

- **fieldValidation** (*in query*): string

- [fieldValidation](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([PersistentVolumeClaim](#)): OK

201 ([PersistentVolumeClaim](#)): Created

401: Unauthorized

update replace status of the specified PersistentVolumeClaim

HTTP Request

PUT /api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the PersistentVolumeClaim
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [PersistentVolumeClaim](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PersistentVolumeClaim](#)): OK

201 ([PersistentVolumeClaim](#)): Created

401: Unauthorized

patch partially update the specified PersistentVolumeClaim

HTTP Request

PATCH /api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}

Parameters

- **name** (*in path*): string, required
name of the PersistentVolumeClaim
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PersistentVolumeClaim](#)): OK

201 ([PersistentVolumeClaim](#)): Created

401: Unauthorized

patch partially update status of the specified PersistentVolumeClaim

HTTP Request

PATCH /api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}/status

Parameters

- **name** (*in path*): string, required
name of the PersistentVolumeClaim
- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PersistentVolumeClaim](#)): OK

201 ([PersistentVolumeClaim](#)): Created

401: Unauthorized

delete delete a PersistentVolumeClaim

HTTP Request

DELETE /api/v1/namespaces/{namespace}/persistentvolumeclaims/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PersistentVolumeClaim
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)

- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([PersistentVolumeClaim](#)): OK

202 ([PersistentVolumeClaim](#)): Accepted

401: Unauthorized

deletecollection delete collection of PersistentVolumeClaim

HTTP Request

DELETE /api/v1/namespaces/{namespace}/persistentvolumeclaims

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)

- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

PersistentVolume

PersistentVolume (PV) is a storage resource provisioned by an administrator.

```
apiVersion: v1
```

```
import "k8s.io/api/core/v1"
```

PersistentVolume

PersistentVolume (PV) is a storage resource provisioned by an administrator. It is analogous to a node. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes>

-
- **apiVersion:** v1
 - **kind:** PersistentVolume
 - **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([PersistentVolumeSpec](#))

spec defines a specification of a persistent volume owned by the cluster. Provisioned by an administrator. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistent-volumes>

- **status** ([PersistentVolumeStatus](#))

status represents the current information/status for the persistent volume. Populated by the system. Read-only. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#persistent-volumes>

PersistentVolumeSpec

PersistentVolumeSpec is the specification of a persistent volume.

- **accessModes** ([]string)

accessModes contains all ways the volume can be mounted. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#access-modes>

- **capacity** (map[string][Quantity](#))

capacity is the description of the persistent volume's resources and capacity. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#capacity>

- **claimRef** ([ObjectReference](#))

claimRef is part of a bi-directional binding between PersistentVolume and PersistentVolumeClaim. Expected to be non-nil when bound. claim.VolumeName is the authoritative bind between PV and PVC. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#binding>

- **mountOptions** ([]string)

mountOptions is the list of mount options, e.g. ["ro", "soft"]. Not validated - mount will simply fail if one is invalid. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes/#mount-options>

- **nodeAffinity** ([VolumeNodeAffinity](#))

nodeAffinity defines constraints that limit what nodes this volume can be accessed from. This field influences the scheduling of pods that use this volume.

VolumeNodeAffinity defines constraints that limit what nodes this volume can be accessed from.

- **nodeAffinity.required** (NodeSelector)

required specifies hard node constraints that must be met.

A node selector represents the union of the results of one or more label queries over a set of nodes; that is, it represents the OR of the selectors represented by the node selector terms.

- **nodeAffinity.required.nodeSelectorTerms** ([]NodeSelectorTerm), required
Required. A list of node selector terms. The terms are ORed.

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

- **nodeAffinity.required.nodeSelectorTerms.matchExpressions** ([]NodeSelectorRequirement)
A list of node selector requirements by node's labels.
- **nodeAffinity.required.nodeSelectorTerms.matchFields** ([]NodeSelectorRequirement)
A list of node selector requirements by node's fields.

- **persistentVolumeReclaimPolicy** (string)

persistentVolumeReclaimPolicy defines what happens to a persistent volume when released from its claim. Valid options are Retain (default for manually created PersistentVolumes), Delete (default for dynamically provisioned PersistentVolumes), and Recycle (deprecated). Recycle must be supported by the volume plugin underlying this PersistentVolume. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#reclaiming>

- **storageClassName** (string)

storageClassName is the name of StorageClass to which this persistent volume belongs. Empty value means that this volume does not belong to any StorageClass.

- **volumeMode** (string)

volumeMode defines if a volume is intended to be used with a formatted filesystem or to remain in raw block state. Value of Filesystem is implied when not included in spec.

Local

- **hostPath** (HostPathVolumeSource)

hostPath represents a directory on the host. Provisioned by a developer or tester. This is useful for single-node development and testing only! On-host storage is not supported in any way and WILL NOT WORK in a multi-node cluster. More info: <https://kubernetes.io/docs/concepts/storage/volumes#hostpath>

Represents a host path mapped into a pod. Host path volumes do not support ownership management or SELinux relabeling.

- **hostPath.path** (string), required

path of the directory on the host. If the path is a symlink, it will follow the link to the real path. More info: <https://kubernetes.io/docs/concepts/storage/volumes#hostpath>

- **hostPath.type** (string)

type for HostPath Volume Defaults to "" More info: <https://kubernetes.io/docs/concepts/storage/volumes#hostpath>

- **local** (LocalVolumeSource)

local represents directly-attached storage with node affinity

Local represents directly-attached storage with node affinity (Beta feature)

- **local.path** (string), required

path of the full path to the volume on the node. It can be either a directory or block device (disk, partition, ...).

- **local.fsType** (string)

fsType is the filesystem type to mount. It applies only when the Path is a block device. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". The default value is to auto-select a filesystem if unspecified.

Persistent volumes

- **awsElasticBlockStore** (AWSElasticBlockStoreVolumeSource)

awsElasticBlockStore represents an AWS Disk resource that is attached to a kubelet's host machine and then exposed to the pod. More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

*Represents a Persistent Disk resource in AWS.

An AWS EBS disk must exist before mounting to a container. The disk must also be in the same AWS zone as the kubelet. An AWS EBS disk can only be mounted as read/write once. AWS EBS volumes support ownership management and SELinux relabeling.*

- **awsElasticBlockStore.volumeID** (string), required

volumeID is unique ID of the persistent disk resource in AWS (Amazon EBS volume). More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

- **awsElasticBlockStore.fsType** (string)

fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

- **awsElasticBlockStore.partition** (int32)

partition is the partition in the volume that you want to mount. If omitted, the default is to mount by volume name. Examples: For volume /dev/sda1, you specify the partition as "1". Similarly, the volume partition for /dev/sda is "0" (or you can leave the property empty).

- **awsElasticBlockStore.readOnly** (boolean)

readOnly value true will force the readOnly setting in VolumeMounts. More info: <https://kubernetes.io/docs/concepts/storage/volumes#awselasticblockstore>

- **azureDisk** (AzureDiskVolumeSource)

azureDisk represents an Azure Data Disk mount on the host and bind mount to the pod.

AzureDisk represents an Azure Data Disk mount on the host and bind mount to the pod.

- **azureDisk.diskName** (string), required

diskName is the Name of the data disk in the blob storage

- **azureDisk.diskURI** (string), required

diskURI is the URI of data disk in the blob storage

- **azureDisk.cachingMode** (string)

cachingMode is the Host Caching mode: None, Read Only, Read Write.

- **azureDisk.fsType** (string)

fsType is Filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **azureDisk.kind** (string)

kind expected values are Shared: multiple blob disks per storage account Dedicated: single blob disk per storage account Managed: azure managed data disk (only in managed availability set). defaults to shared

- **azureDisk.readOnly** (boolean)

readOnly Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **azureFile** (AzureFilePersistentVolumeSource)

azureFile represents an Azure File Service mount on the host and bind mount to the pod.

AzureFile represents an Azure File Service mount on the host and bind mount to the pod.

- **azureFile.secretName** (string), required

secretName is the name of secret that contains Azure Storage Account Name and Key

- **azureFile.shareName** (string), required
 - shareName is the azure Share Name
 - **azureFile.readOnly** (boolean)

readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
 - **azureFile.secretNamespace** (string)

secretNamespace is the namespace of the secret that contains Azure Storage Account Name and Key default is the same as the Pod
- **cephfs** (CephFSPersistentVolumeSource)

cephFS represents a Ceph FS mount on the host that shares a pod's lifetime

Represents a Ceph Filesystem mount that lasts the lifetime of a pod Cephfs volumes do not support ownership management or SELinux relabeling.

 - **cephfs.monitors** ([]string), required

monitors is Required: Monitors is a collection of Ceph monitors More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>
 - **cephfs.path** (string)

path is Optional: Used as the mounted root, rather than the full Ceph tree, default is /
 - **cephfs.readOnly** (boolean)

readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts. More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>
 - **cephfs.secretFile** (string)

secretFile is Optional: SecretFile is the path to key ring for User, default is /etc/ceph/user.secret More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>
 - **cephfs.secretRef** (SecretReference)

secretRef is Optional: SecretRef is reference to the authentication secret for User, default is empty. More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

 - **cephfs.secretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **cephfs.secretRef.namespace** (string)

- namespace defines the space within which the secret name must be unique.

- **cephfs.user** (string)

- user is Optional: User is the rados user name, default is admin More info: <https://examples.k8s.io/volumes/cephfs/README.md#how-to-use-it>

- **cinder** (CinderPersistentVolumeSource)

- cinder represents a cinder volume attached and mounted on kubelets host machine. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

Represents a cinder volume resource in Openstack. A Cinder volume must exist before mounting to a container. The volume must also be in the same region as the kubelet. Cinder volumes support ownership management and SELinux relabeling.

- **cinder.volumeID** (string), required

- volumeID used to identify the volume in cinder. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

- **cinder.fsType** (string)

- fsType Filesystem type to mount. Must be a filesystem type supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

- **cinder.readOnly** (boolean)

- readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts. More info: <https://examples.k8s.io/mysql-cinder-pd/README.md>

- **cinder.secretRef** (SecretReference)

- secretRef is Optional: points to a secret object containing parameters used to connect to OpenStack.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **cinder.secretRef.name** (string)

- name is unique within a namespace to reference a secret resource.

- **cinder.secretRef.namespace** (string)

- namespace defines the space within which the secret name must be unique.

- **csi** (CSIPersistentVolumeSource)

- csi represents storage that is handled by an external CSI driver (Beta feature).

Represents storage that is managed by an external CSI volume driver (Beta feature)

- **csi.driver** (string), required

driver is the name of the driver to use for this volume. Required.

- **csi.volumeHandle** (string), required

volumeHandle is the unique volume name returned by the CSI volume plugin's CreateVolume to refer to the volume on all subsequent calls. Required.

- **csi.controllerExpandSecretRef** (SecretReference)

controllerExpandSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI ControllerExpandVolume call. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secrets are passed.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **csi.controllerExpandSecretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **csi.controllerExpandSecretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **csi.controllerPublishSecretRef** (SecretReference)

controllerPublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI ControllerPublishVolume and ControllerUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secrets are passed.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **csi.controllerPublishSecretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **csi.controllerPublishSecretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **csi.fsType** (string)

fsType to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs".

- **csi.nodeExpandSecretRef** (SecretReference)

nodeExpandSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodeExpandVolume call. This is a beta field which is enabled default by CSINodeExpandSecret feature gate. This field is optional, may be omitted if no secret is required. If the secret object contains more than one secret, all secrets are passed.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **csi.nodeExpandSecretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **csi.nodeExpandSecretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **csi.nodePublishSecretRef** (SecretReference)

nodePublishSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodePublishVolume and NodeUnpublishVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secrets are passed.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **csi.nodePublishSecretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **csi.nodePublishSecretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **csi.nodeStageSecretRef** (SecretReference)

nodeStageSecretRef is a reference to the secret object containing sensitive information to pass to the CSI driver to complete the CSI NodeStageVolume and NodeStageVolume and NodeUnstageVolume calls. This field is optional, and may be empty if no secret is required. If the secret object contains more than one secret, all secrets are passed.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **csi.nodeStageSecretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **csi.nodeStageSecretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **csi.readOnly** (boolean)

readOnly value to pass to ControllerPublishVolumeRequest. Defaults to false (read/write).

- **csi.volumeAttributes** (map[string]string)

volumeAttributes of the volume to publish.

- **fc** (FCVolumeSource)

fc represents a Fibre Channel resource that is attached to a kubelet's host machine and then exposed to the pod.

Represents a Fibre Channel volume. Fibre Channel volumes can only be mounted as read/write once. Fibre Channel volumes support ownership management and SELinux relabeling.

- **fc.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **fc.lun** (int32)

lun is Optional: FC target lun number

- **fc.readOnly** (boolean)

readOnly is Optional: Defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **fc.targetWWNs** ([]string)

targetWWNs is Optional: FC target worldwide names (WWNs)

- **fc.wwids** ([]string)

wwids Optional: FC volume world wide identifiers (wwids) Either wwid or combination of targetWWNs and lun must be set, but not both simultaneously.

- **flexVolume** (FlexPersistentVolumeSource)

flexVolume represents a generic volume resource that is provisioned/attached using an exec based plugin.

FlexPersistentVolumeSource represents a generic persistent volume resource that is provisioned/attached using an exec based plugin.

- **flexVolume.driver** (string), required

driver is the name of the driver to use for this volume.

- **flexVolume.fsType** (string)

fsType is the Filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". The default filesystem depends on FlexVolume script.

flexVolume.options (map[string]string)

- - options is Optional: this field holds extra command options if any.
 - **flexVolume.readOnly** (boolean)

readOnly is Optional: defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **flexVolume.secretRef** (SecretReference)

secretRef is Optional: SecretRef is reference to the secret object containing sensitive information to pass to the plugin scripts. This may be empty if no secret object is specified. If the secret object contains more than one secret, all secrets are passed to the plugin scripts.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **flexVolume.secretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **flexVolume.secretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

• **flocker** (FlockerVolumeSource)

flocker represents a Flocker volume attached to a kubelet's host machine and exposed to the pod for its usage. This depends on the Flocker control service being running

Represents a Flocker volume mounted by the Flocker agent. One and only one of datasetName and datasetUUID should be set. Flocker volumes do not support ownership management or SELinux relabeling.

- **flocker.datasetName** (string)

datasetName is Name of the dataset stored as metadata -> name on the dataset for Flocker should be considered as deprecated

- **flocker.datasetUUID** (string)

datasetUUID is the UUID of the dataset. This is unique identifier of a Flocker dataset

• **gcePersistentDisk** (GCEPersistentDiskVolumeSource)

gcePersistentDisk represents a GCE Disk resource that is attached to a kubelet's host machine and then exposed to the pod. Provisioned by an admin. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

*Represents a Persistent Disk resource in Google Compute Engine.

A GCE PD must exist before mounting to a container. The disk must also be in the same GCE project and zone as the kubelet. A GCE PD can only be mounted as read/write once or read-only many times. GCE PDs support ownership management and SELinux relabeling.*

- **gcePersistentDisk.pdName** (string), required

pdName is unique name of the PD resource in GCE. Used to identify the disk in GCE. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **gcePersistentDisk.fsType** (string)

fsType is filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **gcePersistentDisk.partition** (int32)

partition is the partition in the volume that you want to mount. If omitted, the default is to mount by volume name. Examples: For volume /dev/sda1, you specify the partition as "1". Similarly, the volume partition for /dev/sda is "0" (or you can leave the property empty). More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **gcePersistentDisk.readOnly** (boolean)

readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false. More info: <https://kubernetes.io/docs/concepts/storage/volumes#gcepersistentdisk>

- **glusterfs** (GlusterfsPersistentVolumeSource)

glusterfs represents a Glusterfs volume that is attached to a host and exposed to the pod. Provisioned by an admin. More info: <https://examples.k8s.io/volumes/glusterfs/README.md>

Represents a Glusterfs mount that lasts the lifetime of a pod. Glusterfs volumes do not support ownership management or SELinux relabeling.

- **glusterfs.endpoints** (string), required

endpoints is the endpoint name that details Glusterfs topology. More info: <https://examples.k8s.io/volumes/glusterfs/README.md#create-a-pod>

- **glusterfs.path** (string), required

path is the Glusterfs volume path. More info: <https://examples.k8s.io/volumes/glusterfs/README.md#create-a-pod>

- **glusterfs.endpointsNamespace** (string)

endpointsNamespace is the namespace that contains Glusterfs endpoint. If this field is empty, the EndpointNamespace defaults to the same namespace as the bound

PVC. More info: <https://examples.k8s.io/volumes/glusterfs/README.md#create-a-pod>

- **glusterfs.readOnly** (boolean)

readOnly here will force the Glusterfs volume to be mounted with read-only permissions. Defaults to false. More info: <https://examples.k8s.io/volumes/glusterfs/README.md#create-a-pod>

- **iscsi** (ISCSIPersistentVolumeSource)

iscsi represents an iSCSI Disk resource that is attached to a kubelet's host machine and then exposed to the pod. Provisioned by an admin.

ISCSIPersistentVolumeSource represents an iSCSI disk. iSCSI volumes can only be mounted as read/write once. iSCSI volumes support ownership management and SELinux relabeling.

- **iscsi.iqn** (string), required

iqn is Target iSCSI Qualified Name.

- **iscsi.lun** (int32), required

lun is iSCSI Target Lun number.

- **iscsi.targetPortal** (string), required

targetPortal is iSCSI Target Portal. The Portal is either an IP or ip_addr:port if the port is other than default (typically TCP ports 860 and 3260).

- **iscsi.chapAuthDiscovery** (boolean)

chapAuthDiscovery defines whether support iSCSI Discovery CHAP authentication

- **iscsi.chapAuthSession** (boolean)

chapAuthSession defines whether support iSCSI Session CHAP authentication

- **iscsi.fsType** (string)

fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#iscsi>

- **iscsi.initiatorName** (string)

initiatorName is the custom iSCSI Initiator Name. If initiatorName is specified with iscsiInterface simultaneously, new iSCSI interface <target portal>:<volume name> will be created for the connection.

- **iscsi.iscsiInterface** (string)

iscsiInterface is the interface Name that uses an iSCSI transport. Defaults to 'default' (tcp).

iscsi.portals ([]string)

- portals is the iSCSI Target Portal List. The Portal is either an IP or ip_addr:port if the port is other than default (typically TCP ports 860 and 3260).

- **iscsi.readOnly** (boolean)

readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false.

- **iscsi.secretRef** (SecretReference)

secretRef is the CHAP Secret for iSCSI target and initiator authentication

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **iscsi.secretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **iscsi.secretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **nfs** (NFSVolumeSource)

nfs represents an NFS mount on the host. Provisioned by an admin. More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

Represents an NFS mount that lasts the lifetime of a pod. NFS volumes do not support ownership management or SELinux relabeling.

- **nfs.path** (string), required

path that is exported by the NFS server. More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

- **nfs.server** (string), required

server is the hostname or IP address of the NFS server. More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

- **nfs.readOnly** (boolean)

readOnly here will force the NFS export to be mounted with read-only permissions. Defaults to false. More info: <https://kubernetes.io/docs/concepts/storage/volumes#nfs>

- **photonPersistentDisk** (PhotonPersistentDiskVolumeSource)

photonPersistentDisk represents a PhotonController persistent disk attached and mounted on kubelets host machine

Represents a Photon Controller persistent disk resource.

- **photonPersistentDisk.pdID** (string), required

pdID is the ID that identifies Photon Controller persistent disk

- **photonPersistentDisk.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **portworxVolume** (PortworxVolumeSource)

portworxVolume represents a portworx volume attached and mounted on kubelets host machine

PortworxVolumeSource represents a Portworx volume resource.

- **portworxVolume.volumeID** (string), required

volumeID uniquely identifies a Portworx volume

- **portworxVolume.fsType** (string)

fSType represents the filesystem type to mount Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs". Implicitly inferred to be "ext4" if unspecified.

- **portworxVolume.readOnly** (boolean)

readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.

- **quobyte** (QuobyteVolumeSource)

quobyte represents a Quobyte mount on the host that shares a pod's lifetime

Represents a Quobyte mount that lasts the lifetime of a pod. Quobyte volumes do not support ownership management or SELinux relabeling.

- **quobyte.registry** (string), required

registry represents a single or multiple Quobyte Registry services specified as a string as host:port pair (multiple entries are separated with commas) which acts as the central registry for volumes

- **quobyte.volume** (string), required

volume is a string that references an already created Quobyte volume by name.

- **quobyte.group** (string)

group to map volume access to Default is no group

- **quobyte.readOnly** (boolean)

readOnly here will force the Quobyte volume to be mounted with read-only permissions. Defaults to false.

quobyte.tenant (string)

- tenant owning the given Quobyte volume in the Backend Used with dynamically provisioned Quobyte volumes, value is set by the plugin

- **quobyte.user** (string)

user to map volume access to Defaults to serviceaccount user

- **rbd** (RBDPersistentVolumeSource)

rbd represents a Rados Block Device mount on the host that shares a pod's lifetime. More info: <https://examples.k8s.io/volumes/rbd/README.md>

Represents a Rados Block Device mount that lasts the lifetime of a pod. RBD volumes support ownership management and SELinux relabeling.

- **rbd.image** (string), required

image is the rados image name. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.monitors** ([]string), required

monitors is a collection of Ceph monitors. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.fsType** (string)

fsType is the filesystem type of the volume that you want to mount. Tip: Ensure that the filesystem type is supported by the host operating system. Examples: "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified. More info: <https://kubernetes.io/docs/concepts/storage/volumes#rbd>

- **rbd.keyring** (string)

keyring is the path to key ring for RBDUser. Default is /etc/ceph/keyring. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.pool** (string)

pool is the rados pool name. Default is rbd. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.readOnly** (boolean)

readOnly here will force the ReadOnly setting in VolumeMounts. Defaults to false. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **rbd.secretRef** (SecretReference)

secretRef is name of the authentication secret for RBDUser. If provided overrides keyring. Default is nil. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **rbd.secretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **rbd.secretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **rbd.user** (string)

user is the rados user name. Default is admin. More info: <https://examples.k8s.io/volumes/rbd/README.md#how-to-use-it>

- **scaleIO** (ScaleIOPersistentVolumeSource)

scaleIO represents a ScaleIO persistent volume attached and mounted on Kubernetes nodes.

ScaleIOPersistentVolumeSource represents a persistent ScaleIO volume

- **scaleIO.gateway** (string), required

gateway is the host address of the ScaleIO API Gateway.

- **scaleIO.secretRef** (SecretReference), required

secretRef references to the secret for ScaleIO user and other sensitive information. If this is not provided, Login operation will fail.

SecretReference represents a Secret Reference. It has enough information to retrieve secret in any namespace

- **scaleIO.secretRef.name** (string)

name is unique within a namespace to reference a secret resource.

- **scaleIO.secretRef.namespace** (string)

namespace defines the space within which the secret name must be unique.

- **scaleIO.system** (string), required

system is the name of the storage system as configured in ScaleIO.

- **scaleIO.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Default is "xfs"

- **scaleIO.protectionDomain** (string)

protectionDomain is the name of the ScaleIO Protection Domain for the configured storage.

- **scaleIO.readOnly** (boolean)
 - readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
 - **scaleIO.sslEnabled** (boolean)

sslEnabled is the flag to enable/disable SSL communication with Gateway, default false
 - **scaleIO.storageMode** (string)

storageMode indicates whether the storage for a volume should be ThickProvisioned or ThinProvisioned. Default is ThinProvisioned.
 - **scaleIO.storagePool** (string)

storagePool is the ScaleIO Storage Pool associated with the protection domain.
 - **scaleIO.volumeName** (string)

volumeName is the name of a volume already created in the ScaleIO system that is associated with this volume source.
- **storageos** (StorageOSPersistentVolumeSource)

storageOS represents a StorageOS volume that is attached to the kubelet's host machine and mounted into the pod More info: <https://examples.k8s.io/volumes/storageos/README.md>

Represents a StorageOS persistent volume resource.

 - **storageos.fsType** (string)

fsType is the filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.
 - **storageos.readOnly** (boolean)

readOnly defaults to false (read/write). ReadOnly here will force the ReadOnly setting in VolumeMounts.
 - **storageos.secretRef** ([ObjectReference](#))

secretRef specifies the secret to use for obtaining the StorageOS API credentials. If not specified, default values will be attempted.
 - **storageos.volumeName** (string)

volumeName is the human-readable name of the StorageOS volume. Volume names are only unique within a namespace.
 - **storageos.volumeNamespace** (string)

volumeNamespace specifies the scope of the volume within StorageOS. If no namespace is specified then the Pod's namespace will be used. This allows the Kubernetes name scoping to be mirrored within StorageOS for tighter integration. Set VolumeName to any name to override the default behaviour. Set to "default" if you are not using namespaces within StorageOS. Namespaces that do not pre-exist within StorageOS will be created.

- **vsphereVolume** (`VsphereVirtualDiskVolumeSource`)

vsphereVolume represents a vSphere volume attached and mounted on kubelets host machine

Represents a vSphere volume resource.

- **vsphereVolume.volumePath** (string), required

volumePath is the path that identifies vSphere volume vmdk

- **vsphereVolume.fsType** (string)

fsType is filesystem type to mount. Must be a filesystem type supported by the host operating system. Ex. "ext4", "xfs", "ntfs". Implicitly inferred to be "ext4" if unspecified.

- **vsphereVolume.storagePolicyID** (string)

storagePolicyID is the storage Policy Based Management (SPBM) profile ID associated with the StoragePolicyName.

- **vsphereVolume.storagePolicyName** (string)

storagePolicyName is the storage Policy Based Management (SPBM) profile name.

PersistentVolumeStatus

PersistentVolumeStatus is the current status of a persistent volume.

- **lastPhaseTransitionTime** (`Time`)

lastPhaseTransitionTime is the time the phase transitioned from one to another and automatically resets to current time everytime a volume phase transitions. This is an alpha field and requires enabling PersistentVolumeLastPhaseTransitionTime feature.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **message** (string)

message is a human-readable message indicating details about why the volume is in this state.

- **phase** (string)

phase indicates if a volume is available, bound to a claim, or released by a claim. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#phase>

- **reason** (string)

reason is a brief CamelCase string that describes any failure and is meant for machine parsing and tidy display in the CLI.

PersistentVolumeList

PersistentVolumeList is a list of PersistentVolume items.

- **apiVersion**: v1
- **kind**: PersistentVolumeList
- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][PersistentVolume](#)), required

items is a list of persistent volumes. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes>

Operations

get read the specified PersistentVolume

HTTP Request

GET /api/v1/persistentvolumes/{name}

Parameters

- **name** (*in path*): string, required

name of the PersistentVolume

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PersistentVolume](#)): OK

401: Unauthorized

get read status of the specified PersistentVolume

HTTP Request

GET /api/v1/persistentvolumes/{name}/status

Parameters

- **name** (*in path*): string, required

name of the PersistentVolume

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PersistentVolume](#)): OK

401: Unauthorized

list list or watch objects of kind PersistentVolume

HTTP Request

GET /api/v1/persistentvolumes

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([PersistentVolumeList](#)): OK

401: Unauthorized

create create a PersistentVolume

HTTP Request

POST /api/v1/persistentvolumes

Parameters

- **body**: [PersistentVolume](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PersistentVolume](#)): OK

201 ([PersistentVolume](#)): Created

202 ([PersistentVolume](#)): Accepted

401: Unauthorized

update replace the specified PersistentVolume

HTTP Request

PUT /api/v1/persistentvolumes/{name}

Parameters

- **name** (*in path*): string, required
name of the PersistentVolume
- **body**: [PersistentVolume](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PersistentVolume](#)): OK

201 ([PersistentVolume](#)): Created

401: Unauthorized

update replace status of the specified PersistentVolume

HTTP Request

PUT /api/v1/persistentvolumes/{name}/status

Parameters

- **name** (*in path*): string, required
name of the PersistentVolume
- **body**: [PersistentVolume](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PersistentVolume](#)): OK

201 ([PersistentVolume](#)): Created

401: Unauthorized

patch partially update the specified PersistentVolume

HTTP Request

PATCH /api/v1/persistentvolumes/{name}

Parameters

- **name** (*in path*): string, required
name of the PersistentVolume
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PersistentVolume](#)): OK

201 ([PersistentVolume](#)): Created

401: Unauthorized

patch partially update status of the specified PersistentVolume

HTTP Request

PATCH /api/v1/persistentvolumes/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the PersistentVolume
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PersistentVolume](#)): OK

201 ([PersistentVolume](#)): Created

401: Unauthorized

delete delete a PersistentVolume

HTTP Request

DELETE /api/v1/persistentvolumes/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PersistentVolume
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([PersistentVolume](#)): OK

202 ([PersistentVolume](#)): Accepted

401: Unauthorized

deletecollection delete collection of PersistentVolume

HTTP Request

DELETE /api/v1/persistentvolumes

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

StorageClass

StorageClass describes the parameters for a class of storage for which PersistentVolumes can be dynamically provisioned.

```
apiVersion: storage.k8s.io/v1
```

```
import "k8s.io/api/storage/v1"
```

StorageClass

StorageClass describes the parameters for a class of storage for which PersistentVolumes can be dynamically provisioned.

StorageClasses are non-namespaced; the name of the storage class according to etcd is in ObjectMeta.Name.

- **apiVersion:** storage.k8s.io/v1
- **kind:** StorageClass
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **provisioner** (string), required

provisioner indicates the type of the provisioner.

- **allowVolumeExpansion** (boolean)

allowVolumeExpansion shows whether the storage class allow volume expand.

- **allowedTopologies** ([]TopologySelectorTerm)

Atomic: will be replaced during a merge

allowedTopologies restrict the node topologies where volumes can be dynamically provisioned. Each volume plugin defines its own supported topology specifications. An empty TopologySelectorTerm list means there is no topology restriction. This field is only honored by servers that enable the VolumeScheduling feature.

A topology selector term represents the result of label queries. A null or empty topology selector term matches no objects. The requirements of them are ANDed. It provides a subset of functionality as NodeSelectorTerm. This is an alpha feature and may change in the future.

- **allowedTopologies.matchLabelExpressions** ([]TopologySelectorLabelRequirement)

A list of topology selector requirements by labels.

A topology selector requirement is a selector that matches given label. This is an alpha feature and may change in the future.

- **allowedTopologies.matchLabelExpressions.key** (string), required

The label key that the selector applies to.

- **allowedTopologies.matchLabelExpressions.values** ([]string), required

An array of string values. One value must match the label to be selected.
Each entry in Values is ORed.

- **mountOptions** ([]string)

mountOptions controls the mountOptions for dynamically provisioned PersistentVolumes of this storage class. e.g. ["ro", "soft"]. Not validated - mount of the PVs will simply fail if one is invalid.

- **parameters** (map[string]string)

parameters holds the parameters for the provisioner that should create volumes of this storage class.

- **reclaimPolicy** (string)

reclaimPolicy controls the reclaimPolicy for dynamically provisioned PersistentVolumes of this storage class. Defaults to Delete.

- **volumeBindingMode** (string)

volumeBindingMode indicates how PersistentVolumeClaims should be provisioned and bound. When unset, VolumeBindingImmediate is used. This field is only honored by servers that enable the VolumeScheduling feature.

StorageClassList

StorageClassList is a collection of storage classes.

- **apiVersion**: storage.k8s.io/v1
- **kind**: StorageClassList
- **metadata** ([ListMeta](#))

Standard list metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][StorageClass](#)), required

items is the list of StorageClasses

Operations

get read the specified StorageClass

HTTP Request

GET /apis/storage.k8s.io/v1/storageclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the StorageClass

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([StorageClass](#)): OK

401: Unauthorized

list list or watch objects of kind StorageClass

HTTP Request

GET /apis/storage.k8s.io/v1/storageclasses

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([StorageClassList](#)): OK

401: Unauthorized

create create a StorageClass

HTTP Request

POST /apis/storage.k8s.io/v1/storageclasses

Parameters

- **body**: [StorageClass](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

[pretty](#)

Response

200 ([StorageClass](#)): OK

201 ([StorageClass](#)): Created

202 ([StorageClass](#)): Accepted

401: Unauthorized

update replace the specified StorageClass

HTTP Request

PUT /apis/storage.k8s.io/v1/storageclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the StorageClass

- **body**: [StorageClass](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([StorageClass](#)): OK

201 ([StorageClass](#)): Created

401: Unauthorized

patch partially update the specified StorageClass

HTTP Request

PATCH /apis/storage.k8s.io/v1/storageclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the StorageClass
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([StorageClass](#)): OK

201 ([StorageClass](#)): Created

401: Unauthorized

delete delete a StorageClass

HTTP Request

DELETE /apis/storage.k8s.io/v1/storageclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the StorageClass

- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([StorageClass](#)): OK

202 ([StorageClass](#)): Accepted

401: Unauthorized

deletecollection delete collection of StorageClass

HTTP Request

DELETE /apis/storage.k8s.io/v1/storageclasses

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

VolumeAttachment

VolumeAttachment captures the intent to attach or detach the specified volume to/from the specified node.

```
apiVersion: storage.k8s.io/v1
```

```
import "k8s.io/api/storage/v1"
```

VolumeAttachment

VolumeAttachment captures the intent to attach or detach the specified volume to/from the specified node.

VolumeAttachment objects are non-namespaced.

-
- **apiVersion**: storage.k8s.io/v1

- **kind**: VolumeAttachment

- **metadata** ([ObjectMeta](#))

Standard object metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([VolumeAttachmentSpec](#)), required

spec represents specification of the desired attach/detach volume behavior. Populated by the Kubernetes system.

- **status** ([VolumeAttachmentStatus](#))

status represents status of the VolumeAttachment request. Populated by the entity completing the attach or detach operation, i.e. the external-attacher.

VolumeAttachmentSpec

VolumeAttachmentSpec is the specification of a VolumeAttachment request.

- **attacher** (string), required

attacher indicates the name of the volume driver that MUST handle this request. This is the name returned by GetPluginName().

- **nodeName** (string), required

nodeName represents the node that the volume should be attached to.

- **source** ([VolumeAttachmentSource](#)), required

source represents the volume that should be attached.

VolumeAttachmentSource represents a volume that should be attached. Right now only PersistentVolumes can be attached via external attacher, in future we may allow also inline volumes in pods. Exactly one member can be set.

- **source.inlineVolumeSpec** ([PersistentVolumeSpec](#))

inlineVolumeSpec contains all the information necessary to attach a persistent volume defined by a pod's inline VolumeSource. This field is populated only for the CSIMigration feature. It contains translated fields from a pod's inline VolumeSource to a PersistentVolumeSpec. This field is beta-level and is only honored by servers that enabled the CSIMigration feature.

- **source.persistentVolumeName** (string)

persistentVolumeName represents the name of the persistent volume to attach.

VolumeAttachmentStatus

VolumeAttachmentStatus is the status of a VolumeAttachment request.

- **attached** (boolean), required

attached indicates the volume is successfully attached. This field must only be set by the entity completing the attach operation, i.e. the external-attacher.

- **attachError** (VolumeError)

attachError represents the last error encountered during attach operation, if any. This field must only be set by the entity completing the attach operation, i.e. the external-attacher.

VolumeError captures an error encountered during a volume operation.

- **attachError.message** (string)

message represents the error encountered during Attach or Detach operation. This string may be logged, so it should not contain sensitive information.

- **attachError.time** (Time)

time represents the time the error was encountered.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **attachmentMetadata** (map[string]string)

attachmentMetadata is populated with any information returned by the attach operation, upon successful attach, that must be passed into subsequent WaitForAttach or Mount calls. This field must only be set by the entity completing the attach operation, i.e. the external-attacher.

- **detachError** (VolumeError)

detachError represents the last error encountered during detach operation, if any. This field must only be set by the entity completing the detach operation, i.e. the external-attacher.

VolumeError captures an error encountered during a volume operation.

- **detachError.message** (string)

message represents the error encountered during Attach or Detach operation. This string may be logged, so it should not contain sensitive information.

- **detachError.time** (Time)

time represents the time the error was encountered.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

VolumeAttachmentList

VolumeAttachmentList is a collection of VolumeAttachment objects.

- **apiVersion**: storage.k8s.io/v1
- **kind**: VolumeAttachmentList
- **metadata** ([ListMeta](#))
Standard list metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>
- **items** ([][VolumeAttachment](#)), required
items is the list of VolumeAttachments

Operations

get read the specified VolumeAttachment

HTTP Request

GET /apis/storage.k8s.io/v1/volumeattachments/{name}

Parameters

- **name** (*in path*): string, required
name of the VolumeAttachment
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([VolumeAttachment](#)): OK

401: Unauthorized

get read status of the specified VolumeAttachment

HTTP Request

GET /apis/storage.k8s.io/v1/volumeattachments/{name}/status

Parameters

- **name** (*in path*): string, required

name of the VolumeAttachment

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([VolumeAttachment](#)): OK

401: Unauthorized

list list or watch objects of kind VolumeAttachment

HTTP Request

GET /apis/storage.k8s.io/v1/volumeattachments

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([VolumeAttachmentList](#)): OK

401: Unauthorized

create create a VolumeAttachment

HTTP Request

POST /apis/storage.k8s.io/v1/volumeattachments

Parameters

- **body**: [VolumeAttachment](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([VolumeAttachment](#)): OK

201 ([VolumeAttachment](#)): Created

202 ([VolumeAttachment](#)): Accepted

401: Unauthorized

update replace the specified VolumeAttachment

HTTP Request

PUT /apis/storage.k8s.io/v1/volumeattachments/{name}

Parameters

- **name** (*in path*): string, required

name of the VolumeAttachment

- **body**: [VolumeAttachment](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([VolumeAttachment](#)): OK

201 ([VolumeAttachment](#)): Created

401: Unauthorized

update replace status of the specified VolumeAttachment

HTTP Request

PUT /apis/storage.k8s.io/v1/volumeattachments/{name}/status

Parameters

- **name** (*in path*): string, required
name of the VolumeAttachment
- **body**: [VolumeAttachment](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([VolumeAttachment](#)): OK

201 ([VolumeAttachment](#)): Created

401: Unauthorized

patch partially update the specified VolumeAttachment

HTTP Request

PATCH /apis/storage.k8s.io/v1/volumeattachments/{name}

Parameters

- **name** (*in path*): string, required
name of the VolumeAttachment
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([VolumeAttachment](#)): OK

201 ([VolumeAttachment](#)): Created

401: Unauthorized

patch partially update status of the specified VolumeAttachment

HTTP Request

PATCH /apis/storage.k8s.io/v1/volumeattachments/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the VolumeAttachment
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([VolumeAttachment](#)): OK

201 ([VolumeAttachment](#)): Created

401: Unauthorized

delete delete a VolumeAttachment

HTTP Request

DELETE /apis/storage.k8s.io/v1/volumeattachments/{name}

Parameters

- **name** (*in path*): string, required
name of the VolumeAttachment
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([VolumeAttachment](#)): OK

202 ([VolumeAttachment](#)): Accepted

401: Unauthorized

deletecollection delete collection of VolumeAttachment

HTTP Request

DELETE /apis/storage.k8s.io/v1/volumeattachments

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

CSIDriver

CSIDriver captures information about a Container Storage Interface (CSI) volume driver deployed on the cluster.

```
apiVersion: storage.k8s.io/v1
```

```
import "k8s.io/api/storage/v1"
```

CSIDriver

CSIDriver captures information about a Container Storage Interface (CSI) volume driver deployed on the cluster. Kubernetes attach/detach controller uses this object to determine whether attach is required. Kubelet uses this object to determine whether pod information needs to be passed on mount. CSIDriver objects are non-namespaced.

- **apiVersion:** storage.k8s.io/v1
- **kind:** CSIDriver
- **metadata** ([ObjectMeta](#))

Standard object metadata. metadata.Name indicates the name of the CSI driver that this object refers to; it MUST be the same name returned by the CSI GetPluginName() call for that driver. The driver name must be 63 characters or less, beginning and ending with an alphanumeric character ([a-z0-9A-Z]) with dashes (-), dots (.), and alphanumerics between. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([CSIDriverSpec](#)), required

spec represents the specification of the CSI Driver.

CSIDriverSpec

CSIDriverSpec is the specification of a CSIDriver.

- **attachRequired** (boolean)

attachRequired indicates this CSI volume driver requires an attach operation (because it implements the CSI ControllerPublishVolume() method), and that the Kubernetes attach/detach controller should call the attach volume interface which checks the volumeattachment status and waits until the volume is attached before proceeding to mounting. The CSI external-attacher coordinates with CSI volume driver and updates the volumeattachment status when the attach operation is complete. If the CSIDriverRegistry feature gate is enabled and the value is specified to false, the attach operation will be skipped. Otherwise the attach operation will be called.

This field is immutable.

- **fsGroupPolicy** (string)

fsGroupPolicy defines if the underlying volume supports changing ownership and permission of the volume before being mounted. Refer to the specific FSGroupPolicy values for additional details.

This field is immutable.

Defaults to ReadWriteOnceWithFSType, which will examine each volume to determine if Kubernetes should modify ownership and permissions of the volume. With the default policy the defined fsGroup will only be applied if a fstype is defined and the volume's access mode contains ReadWriteOnce.

- **podInfoOnMount** (boolean)

podInfoOnMount indicates this CSI volume driver requires additional pod information (like podName, podUID, etc.) during mount operations, if set to true. If set to false, pod information will not be passed on mount. Default is false.

The CSI driver specifies podInfoOnMount as part of driver deployment. If true, Kubelet will pass pod information as VolumeContext in the CSI NodePublishVolume() calls. The CSI driver is responsible for parsing and validating the information passed in as VolumeContext.

The following VolumeConext will be passed if podInfoOnMount is set to true. This list might grow, but the prefix will be used.
"csi.storage.k8s.io/pod.name": pod.Name
"csi.storage.k8s.io/pod.namespace": pod.Namespace
"csi.storage.k8s.io/pod.uid": string(pod.UID)
"csi.storage.k8s.io/ephemeral": "true" if the volume is an ephemeral inline volume defined by a CSIVolumeSource, otherwise "false"

"csi.storage.k8s.io/ephemeral" is a new feature in Kubernetes 1.16. It is only required for drivers which support both the "Persistent" and "Ephemeral" VolumeLifecycleMode. Other drivers can leave pod info disabled and/or ignore this field. As Kubernetes 1.15 doesn't support this field, drivers can only support one mode when deployed on such a cluster and the deployment determines which mode that is, for example via a command line parameter of the driver.

This field is immutable.

- **requiresRepublish** (boolean)

requiresRepublish indicates the CSI driver wants NodePublishVolume being periodically called to reflect any possible change in the mounted volume. This field defaults to false.

Note: After a successful initial NodePublishVolume call, subsequent calls to NodePublishVolume should only update the contents of the volume. New mount points will not be seen by a running container.

- **seLinuxMount** (boolean)

seLinuxMount specifies if the CSI driver supports "-o context" mount option.

When "true", the CSI driver must ensure that all volumes provided by this CSI driver can be mounted separately with different -o context options. This is typical for storage backends that provide volumes as filesystems on block devices or as independent shared volumes. Kubernetes will call NodeStage / NodePublish with "-o context=xyz" mount option when mounting a ReadWriteOncePod volume used in Pod that has explicitly set SELinux context. In the future, it may be expanded to other volume AccessModes. In any case, Kubernetes will ensure that the volume is mounted only with a single SELinux context.

When "false", Kubernetes won't pass any special SELinux mount options to the driver. This is typical for volumes that represent subdirectories of a bigger shared filesystem.

Default is "false".

- **storageCapacity** (boolean)

storageCapacity indicates that the CSI volume driver wants pod scheduling to consider the storage capacity that the driver deployment will report by creating CSIStrageCapacity objects with capacity information, if set to true.

The check can be enabled immediately when deploying a driver. In that case, provisioning new volumes with late binding will pause until the driver deployment has published some suitable CSIStrageCapacity object.

Alternatively, the driver can be deployed with the field unset or false and it can be flipped later when storage capacity information has been published.

This field was immutable in Kubernetes <= 1.22 and now is mutable.

- **tokenRequests** ([]TokenRequest)

Atomic: will be replaced during a merge

tokenRequests indicates the CSI driver needs pods' service account tokens it is mounting volume for to do necessary authentication. Kubelet will pass the tokens in VolumeContext in the CSI NodePublishVolume calls. The CSI driver should parse and validate the following VolumeContext: "csi.storage.k8s.io/serviceAccount.tokens": { "<audience>": { "token": <token>, "expirationTimestamp": <expiration timestamp in RFC3339>, }, ... }

Note: Audience in each TokenRequest should be different and at most one token is empty string. To receive a new token after expiry, RequiresRepublish can be used to trigger NodePublishVolume periodically.

TokenRequest contains parameters of a service account token.

- **tokenRequests.audience** (string), required

audience is the intended audience of the token in "TokenRequestSpec". It will default to the audiences of kube apiserver.

- **tokenRequests.expirationSeconds** (int64)

expirationSeconds is the duration of validity of the token in "TokenRequestSpec". It has the same default value of "ExpirationSeconds" in "TokenRequestSpec".

volumeLifecycleModes ([]string)

- *Set: unique values will be kept during a merge*

volumeLifecycleModes defines what kind of volumes this CSI volume driver supports. The default if the list is empty is "Persistent", which is the usage defined by the CSI specification and implemented in Kubernetes via the usual PV/PVC mechanism.

The other mode is "Ephemeral". In this mode, volumes are defined inline inside the pod spec with CSIVolumeSource and their lifecycle is tied to the lifecycle of that pod. A driver has to be aware of this because it is only going to get a NodePublishVolume call for such a volume.

For more information about implementing this mode, see <https://kubernetes-csi.github.io/docs/ephemeral-local-volumes.html> A driver can support one or more of these modes and more modes may be added in the future.

This field is beta. This field is immutable.

CSIDriverList

CSIDriverList is a collection of CSIDriver objects.

- **apiVersion:** storage.k8s.io/v1

- **kind:** CSIDriverList

- **metadata** ([ListMeta](#))

Standard list metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items ([]CSIDriver)**, required

items is the list of CSIDriver

Operations

get read the specified CSIDriver

HTTP Request

GET /apis/storage.k8s.io/v1/csidrivers/{name}

Parameters

- **name** (*in path*): string, required

name of the CSIDriver

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([CSIDriver](#)): OK

401: Unauthorized

list list or watch objects of kind CSIDriver

HTTP Request

GET /apis/storage.k8s.io/v1/csidrivers

Parameters

- **allowWatchBookmarks** (*in query*): boolean

- [allowWatchBookmarks](#)

- **continue** (*in query*): string

- [continue](#)

- **fieldSelector** (*in query*): string

- [fieldSelector](#)

- **labelSelector** (*in query*): string

- [labelSelector](#)

- **limit** (*in query*): integer

- [limit](#)

- **pretty** (*in query*): string

- [pretty](#)

- **resourceVersion** (*in query*): string

- [resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

- [resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

- [sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)
- **watch** (*in query*): boolean
 - [watch](#)

Response

200 ([CSIDriverList](#)): OK

401: Unauthorized

create create a CSIDriver

HTTP Request

POST /apis/storage.k8s.io/v1/csidrivers

Parameters

- **body**: [CSIDriver](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([CSIDriver](#)): OK

201 ([CSIDriver](#)): Created

202 ([CSIDriver](#)): Accepted

401: Unauthorized

update replace the specified CSIDriver

HTTP Request

PUT /apis/storage.k8s.io/v1/csidrivers/{name}

Parameters

- **name** (*in path*): string, required
name of the CSIDriver
- **body**: [CSIDriver](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CSIDriver](#)): OK

201 ([CSIDriver](#)): Created

401: Unauthorized

patch partially update the specified CSIDriver

HTTP Request

PATCH /apis/storage.k8s.io/v1/csidrivers/{name}

Parameters

- **name** (*in path*): string, required
name of the CSIDriver
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CSIDriver](#)): OK

201 ([CSIDriver](#)): Created

401: Unauthorized

delete delete a CSIDriver

HTTP Request

DELETE /apis/storage.k8s.io/v1/csidrivers/{name}

Parameters

- **name** (*in path*): string, required
name of the CSIDriver
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([CSIDriver](#)): OK

202 ([CSIDriver](#)): Accepted

401: Unauthorized

deletecollection delete collection of CSIDriver

HTTP Request

DELETE /apis/storage.k8s.io/v1/csidrivers

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

CSINode

CSINode holds information about all CSI drivers installed on a node.

apiVersion: storage.k8s.io/v1

```
import "k8s.io/api/storage/v1"
```

CSINode

CSINode holds information about all CSI drivers installed on a node. CSI drivers do not need to create the CSINode object directly. As long as they use the node-driver-registrar sidecar container, the kubelet will automatically populate the CSINode object for the CSI driver as part of kubelet plugin registration. CSINode has the same name as a node. If the object is missing, it means either there are no CSI Drivers available on the node, or the Kubelet version is low enough that it doesn't create this object. CSINode has an OwnerReference that points to the corresponding node object.

-
- **apiVersion**: storage.k8s.io/v1

- **kind**: CSINode

- **metadata** ([ObjectMeta](#))

Standard object's metadata. metadata.name must be the Kubernetes node name.

- **spec** ([CSINodeSpec](#)), required

spec is the specification of CSINode

CSINodeSpec

CSINodeSpec holds information about the specification of all CSI drivers installed on a node

- **drivers** ([]CSINodeDriver), required

Patch strategy: merge on key name

drivers is a list of information of all CSI Drivers existing on a node. If all drivers in the list are uninstalled, this can become empty.

CSINodeDriver holds information about the specification of one CSI driver installed on a node

- **drivers.name** (string), required

name represents the name of the CSI driver that this object refers to. This MUST be the same name returned by the CSI GetPluginName() call for that driver.

- **drivers.nodeID** (string), required

nodeID of the node from the driver point of view. This field enables Kubernetes to communicate with storage systems that do not share the same nomenclature for nodes. For example, Kubernetes may refer to a given node as "node1", but the storage system may refer to the same node as "nodeA". When Kubernetes issues a command to the storage system to attach a volume to a specific node, it can use this field to refer to the node name using the ID that the storage system will understand, e.g. "nodeA" instead of "node1". This field is required.

- **drivers.allocatable** (VolumeNodeResources)

allocatable represents the volume resources of a node that are available for scheduling. This field is beta.

VolumeNodeResources is a set of resource limits for scheduling of volumes.

- **drivers.allocatable.count** (int32)

count indicates the maximum number of unique volumes managed by the CSI driver that can be used on a node. A volume that is both attached and mounted on a node is considered to be used once, not twice. The same rule applies for a unique volume that is shared among multiple pods on the same node. If this field is not specified, then the supported number of volumes on this node is unbounded.

- **drivers.topologyKeys** ([]string)

topologyKeys is the list of keys supported by the driver. When a driver is initialized on a cluster, it provides a set of topology keys that it understands (e.g. "company.com/zone", "company.com/region"). When a driver is initialized on a node, it provides the same topology keys along with values. Kubelet will expose these topology keys as labels on its own node object. When Kubernetes does topology aware provisioning, it can use this list to determine which labels it should

retrieve from the node object and pass back to the driver. It is possible for different nodes to use different topology keys. This can be empty if driver does not support topology.

CSINodeList

CSINodeList is a collection of CSINode objects.

- **apiVersion**: storage.k8s.io/v1
- **kind**: CSINodeList
- **metadata** ([ListMeta](#))

Standard list metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][CSINode](#)), required

items is the list of CSINode

Operations

get read the specified CSINode

HTTP Request

GET /apis/storage.k8s.io/v1/csinodes/{name}

Parameters

- **name** (*in path*): string, required
name of the CSINode
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CSINode](#)): OK

401: Unauthorized

list list or watch objects of kind CSINode

HTTP Request

GET /apis/storage.k8s.io/v1/csinodes

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([CSINodeList](#)): OK

401: Unauthorized

create create a CSINode

HTTP Request

POST /apis/storage.k8s.io/v1/csinodes

Parameters

- **body**: [CSINode](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CSINode](#)): OK

201 ([CSINode](#)): Created

202 ([CSINode](#)): Accepted

401: Unauthorized

update replace the specified CSINode

HTTP Request

PUT /apis/storage.k8s.io/v1/csinodes/{name}

Parameters

- **name** (*in path*): string, required

name of the CSINode

- **body**: [CSINode](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CSINode](#)): OK

201 ([CSINode](#)): Created

401: Unauthorized

patch partially update the specified CSINode

HTTP Request

PATCH /apis/storage.k8s.io/v1/csinodes/{name}

Parameters

- **name** (*in path*): string, required

name of the CSINode

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

- [force](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([CSINode](#)): OK

201 ([CSINode](#)): Created

401: Unauthorized

delete delete a CSINode

HTTP Request

DELETE /apis/storage.k8s.io/v1/csinodes/{name}

Parameters

- **name** (*in path*): string, required

- name of the CSINode

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

- [dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

- [gracePeriodSeconds](#)

- **pretty** (*in query*): string

- [pretty](#)

- **propagationPolicy** (*in query*): string

- [propagationPolicy](#)

Response

200 ([CSINode](#)): OK

202 ([CSINode](#)): Accepted

401: Unauthorized

deletecollection delete collection of CSINode

HTTP Request

DELETE /apis/storage.k8s.io/v1/csinodes

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

CSIStorageCapacity

CSIStorageCapacity stores the result of one CSI GetCapacity call.

```
apiVersion: storage.k8s.io/v1  
import "k8s.io/api/storage/v1"
```

CSIStorageCapacity

CSIStorageCapacity stores the result of one CSI GetCapacity call. For a given StorageClass, this describes the available capacity in a particular topology segment. This can be used when considering where to instantiate new PersistentVolumes.

For example this can express things like: - StorageClass "standard" has "1234 GiB" available in "topology.kubernetes.io/zone=us-east1" - StorageClass "localssd" has "10 GiB" available in "kubernetes.io/hostname=knode-abc123"

The following three cases all imply that no capacity is available for a certain combination: - no object exists with suitable topology and storage class name - such an object exists, but the capacity is unset - such an object exists, but the capacity is zero

The producer of these objects can decide which approach is more suitable.

They are consumed by the kube-scheduler when a CSI driver opts into capacity-aware scheduling with CSIDriverSpec.StorageCapacity. The scheduler compares the MaximumVolumeSize against the requested size of pending volumes to filter out unsuitable nodes. If MaximumVolumeSize is unset, it falls back to a comparison against the less precise Capacity. If that is also unset, the scheduler assumes that capacity is insufficient and tries some other node.

-
- **apiVersion:** storage.k8s.io/v1
 - **kind:** CSIStorageCapacity
 - **metadata** ([ObjectMeta](#))

Standard object's metadata. The name has no particular meaning. It must be a DNS subdomain (dots allowed, 253 characters). To ensure that there are no conflicts with other CSI drivers on the cluster, the recommendation is to use csisc-<uuid>, a generated name, or a reverse-domain name which ends with the unique CSI driver name.

Objects are namespaced.

More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **storageClassName** (string), required

storageClassName represents the name of the StorageClass that the reported capacity applies to. It must meet the same requirements as the name of a StorageClass object (non-empty, DNS subdomain). If that object no longer exists, the CSIStrorageCapacity object is obsolete and should be removed by its creator. This field is immutable.

- **capacity** ([Quantity](#))

capacity is the value reported by the CSI driver in its GetCapacityResponse for a GetCapacityRequest with topology and parameters that match the previous fields.

The semantic is currently (CSI spec 1.2) defined as: The available capacity, in bytes, of the storage that can be used to provision volumes. If not set, that information is currently unavailable.

- **maximumVolumeSize** ([Quantity](#))

maximumVolumeSize is the value reported by the CSI driver in its GetCapacityResponse for a GetCapacityRequest with topology and parameters that match the previous fields.

This is defined since CSI spec 1.4.0 as the largest size that may be used in a CreateVolumeRequest.capacity_range.required_bytes field to create a volume with the same parameters as those in GetCapacityRequest. The corresponding value in the Kubernetes API is ResourceRequirements.Requests in a volume claim.

- **nodeTopology** ([LabelSelector](#))

nodeTopology defines which nodes have access to the storage for which capacity was reported. If not set, the storage is not accessible from any node in the cluster. If empty, the storage is accessible from all nodes. This field is immutable.

CSIStrorageCapacityList

CSIStrorageCapacityList is a collection of CSIStrorageCapacity objects.

- **apiVersion**: storage.k8s.io/v1
- **kind**: CSIStrorageCapacityList
- **metadata** ([ListMeta](#))

Standard list metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][CSIStrorageCapacity](#)), required

Map: unique values on key name will be kept during a merge

items is the list of CSIStrorageCapacity objects.

Operations

get read the specified CSIStrageCapacity

HTTP Request

GET /apis/storage.k8s.io/v1/namespaces/{namespace}/csistoragecapacities/{name}

Parameters

- **name** (*in path*): string, required
name of the CSIStrageCapacity
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CSIStrageCapacity](#)): OK

401: Unauthorized

list list or watch objects of kind CSIStrageCapacity

HTTP Request

GET /apis/storage.k8s.io/v1/namespaces/{namespace}/csistoragecapacities

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)

- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)
- **watch** (*in query*): boolean
 - [watch](#)

Response

200 ([CSIStorageCapacityList](#)): OK

401: Unauthorized

list list or watch objects of kind CSIStorageCapacity

HTTP Request

GET /apis/storage.k8s.io/v1/csistoragecapacities

Parameters

- **allowWatchBookmarks** (*in query*): boolean
 - [allowWatchBookmarks](#)
- **continue** (*in query*): string
 - [continue](#)

- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([CSIStorageCapacityList](#)): OK

401: Unauthorized

create create a CSIStorageCapacity

HTTP Request

POST /apis/storage.k8s.io/v1/namespaces/{namespace}/csistoragecapacities

Parameters

- **namespace** (*in path*): string, required
[namespace](#)

- **body**: [CSIStorageCapacity](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CSIStorageCapacity](#)): OK

201 ([CSIStorageCapacity](#)): Created

202 ([CSIStorageCapacity](#)): Accepted

401: Unauthorized

update replace the specified CSIStorageCapacity

HTTP Request

PUT /apis/storage.k8s.io/v1/namespaces/{namespace}/csistoragecapacities/{name}

Parameters

- **name** (*in path*): string, required
 - name of the CSIStorageCapacity
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [CSIStorageCapacity](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)

fieldValidation (*in query*): string

- [fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([CSIStrageCapacity](#)): OK

201 ([CSIStrageCapacity](#)): Created

401: Unauthorized

patch partially update the specified CSIStrageCapacity

HTTP Request

PATCH /apis/storage.k8s.io/v1/namespaces/{namespace}/csistoragecapacities/{name}

Parameters

• **name** (*in path*): string, required

name of the CSIStrageCapacity

• **namespace** (*in path*): string, required

[namespace](#)

• **body**: [Patch](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **force** (*in query*): boolean

[force](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([CSIStrageCapacity](#)): OK

201 ([CSIStrageCapacity](#)): Created

401: Unauthorized

delete delete a CSIStrageCapacity

HTTP Request

DELETE /apis/storage.k8s.io/v1/namespaces/{namespace}/csistoragecapacities/{name}

Parameters

- **name** (*in path*): string, required
name of the CSIStrageCapacity
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of CSIStrorageCapacity

HTTP Request

DELETE /apis/storage.k8s.io/v1/namespaces/{namespace}/csistoragecapacities

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Authentication Resources

[ServiceAccount](#)

ServiceAccount binds together:

- * a name, understood by users, and perhaps by peripheral systems, for an identity
- * a principal that can be authenticated and authorized
- * a set of secrets.

[TokenRequest](#)

TokenRequest requests a token for a given service account.

[TokenReview](#)

TokenReview attempts to authenticate a token to a known user.

[CertificateSigningRequest](#)

CertificateSigningRequest objects provide a mechanism to obtain x509 certificates by submitting a certificate signing request, and having it asynchronously approved and issued.

[ClusterTrustBundle v1alpha1](#)

ClusterTrustBundle is a cluster-scoped container for X.

[SelfSubjectReview](#)

SelfSubjectReview contains the user information that the kube-apiserver has about the user making this request.

ServiceAccount

ServiceAccount binds together:

- * a name, understood by users, and perhaps by peripheral systems, for an identity
- * a principal that can be authenticated and authorized
- * a set of secrets.

apiVersion: v1

```
import "k8s.io/api/core/v1"
```

ServiceAccount

ServiceAccount binds together:

- * a name, understood by users, and perhaps by peripheral systems, for an identity
- * a principal that can be authenticated and authorized
- * a set of secrets

- **apiVersion:** v1
- **kind:** ServiceAccount
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **automountServiceAccountToken** (boolean)

AutomountServiceAccountToken indicates whether pods running as this service account should have an API token automatically mounted. Can be overridden at the pod level.

- **imagePullSecrets** ([] [LocalObjectReference](#))

ImagePullSecrets is a list of references to secrets in the same namespace to use for pulling any images in pods that reference this ServiceAccount. ImagePullSecrets are distinct from Secrets because Secrets can be mounted in the pod, but ImagePullSecrets are only accessed by the kubelet. More info: <https://kubernetes.io/docs/concepts/containers/images/#specifying-imagepullsecrets-on-a-pod>

- **secrets** ([] [ObjectReference](#))

Patch strategy: merge on key name

Secrets is a list of the secrets in the same namespace that pods running using this ServiceAccount are allowed to use. Pods are only limited to this list if this service account has a "kubernetes.io/enforce-mountable-secrets" annotation set to "true". This field should not be used to find auto-generated service account token secrets for use outside of pods. Instead, tokens can be requested directly using the TokenRequest API, or service account token secrets can be manually created. More info: <https://kubernetes.io/docs/concepts/configuration/secret>

ServiceAccountList

ServiceAccountList is a list of ServiceAccount objects

- **apiVersion:** v1
- **kind:** ServiceAccountList
- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][ServiceAccount](#)), required

List of ServiceAccounts. More info: <https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/>

Operations

get read the specified ServiceAccount

HTTP Request

GET /api/v1/namespaces/{namespace}/serviceaccounts/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ServiceAccount
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ServiceAccount](#)): OK

401: Unauthorized

list list or watch objects of kind ServiceAccount

HTTP Request

GET /api/v1/namespaces/{namespace}/serviceaccounts

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)
- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ServiceAccountList](#)): OK

401: Unauthorized

list list or watch objects of kind ServiceAccount

HTTP Request

GET /api/v1/serviceaccounts

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ServiceAccountList](#)): OK

401: Unauthorized

create create a ServiceAccount

HTTP Request

POST /api/v1/namespaces/{namespace}/serviceaccounts

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ServiceAccount](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ServiceAccount](#)): OK

201 ([ServiceAccount](#)): Created

202 ([ServiceAccount](#)): Accepted

401: Unauthorized

update replace the specified ServiceAccount

HTTP Request

PUT /api/v1/namespaces/{namespace}/serviceaccounts/{name}

Parameters

- **name** (*in path*): string, required

name of the ServiceAccount

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ServiceAccount](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ServiceAccount](#)): OK

201 ([ServiceAccount](#)): Created

401: Unauthorized

patch partially update the specified ServiceAccount

HTTP Request

PATCH /api/v1/namespaces/{namespace}/serviceaccounts/{name}

Parameters

- **name** (*in path*): string, required

name of the ServiceAccount

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ServiceAccount](#)): OK

201 ([ServiceAccount](#)): Created

401: Unauthorized

delete delete a ServiceAccount

HTTP Request

DELETE /api/v1/namespaces/{namespace}/serviceaccounts/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ServiceAccount
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([ServiceAccount](#)): OK

202 ([ServiceAccount](#)): Accepted

401: Unauthorized

deletecollection delete collection of ServiceAccount

HTTP Request

DELETE /api/v1/namespaces/{namespace}/serviceaccounts

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

TokenRequest

TokenRequest requests a token for a given service account.

```
apiVersion: authentication.k8s.io/v1
```

```
import "k8s.io/api/authentication/v1"
```

TokenRequest

TokenRequest requests a token for a given service account.

- **apiVersion**: authentication.k8s.io/v1
- **kind**: TokenRequest
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([TokenRequestSpec](#)), required

Spec holds information about the request being evaluated

- **status** ([TokenRequestStatus](#))

Status is filled in by the server and indicates whether the token can be authenticated.

TokenRequestSpec

TokenRequestSpec contains client provided parameters of a token request.

- **audiences** ([]string), required

Audiences are the intended audiences of the token. A recipient of a token must identify themselves with an identifier in the list of audiences of the token, and otherwise should reject the token. A token issued for multiple audiences may be used to authenticate against any of the audiences listed but implies a high degree of trust between the target audiences.

- **boundObjectRef** (BoundObjectReference)

BoundObjectRef is a reference to an object that the token will be bound to. The token will only be valid for as long as the bound object exists. NOTE: The API server's TokenReview endpoint will validate the BoundObjectRef, but other audiences may not. Keep ExpirationSeconds small if you want prompt revocation.

BoundObjectReference is a reference to an object that a token is bound to.

- **boundObjectRef.apiVersion** (string)

API version of the referent.

- **boundObjectRef.kind** (string)

Kind of the referent. Valid kinds are 'Pod' and 'Secret'.

- **boundObjectRef.name** (string)

Name of the referent.

- **boundObjectRef.uid** (string)

UID of the referent.

- **expirationSeconds** (int64)

ExpirationSeconds is the requested duration of validity of the request. The token issuer may return a token with a different validity duration so a client needs to check the 'expiration' field in a response.

TokenRequestStatus

TokenRequestStatus is the result of a token request.

- **expirationTimestamp** (Time), required

ExpirationTimestamp is the time of expiration of the returned token.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **token** (string), required

Token is the opaque bearer token.

Operations

create create token of a ServiceAccount

HTTP Request

POST /api/v1/namespaces/{namespace}/serviceaccounts/{name}/token

Parameters

- **name** (*in path*): string, required

name of the TokenRequest

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [TokenRequest](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([TokenRequest](#)): OK

201 ([TokenRequest](#)): Created

202 ([TokenRequest](#)): Accepted

401: Unauthorized

TokenReview

TokenReview attempts to authenticate a token to a known user.

```
apiVersion: authentication.k8s.io/v1  
import "k8s.io/api/authentication/v1"
```

TokenReview

TokenReview attempts to authenticate a token to a known user. Note: TokenReview requests may be cached by the webhook token authenticator plugin in the kube-apiserver.

- **apiVersion**: authentication.k8s.io/v1
- **kind**: TokenReview
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([TokenReviewSpec](#)), required

Spec holds information about the request being evaluated

- **status** ([TokenReviewStatus](#))

Status is filled in by the server and indicates whether the request can be authenticated.

TokenReviewSpec

TokenReviewSpec is a description of the token authentication request.

- **audiences** ([]string)

Audiences is a list of the identifiers that the resource server presented with the token identifies as. Audience-aware token authenticators will verify that the token was intended for at least one of the audiences in this list. If no audiences are provided, the audience will default to the audience of the Kubernetes apiserver.

- **token** (string)

Token is the opaque bearer token.

TokenReviewStatus

TokenReviewStatus is the result of the token authentication request.

- **audiences** ([]string)

Audiences are audience identifiers chosen by the authenticator that are compatible with both the TokenReview and token. An identifier is any identifier in the intersection of the TokenReviewSpec audiences and the token's audiences. A client of the TokenReview API that sets the spec.audiences field should validate that a compatible audience identifier is returned in the status.audiences field to ensure that the TokenReview server is audience aware. If a TokenReview returns an empty status.audience field where status.authenticated is "true", the token is valid against the audience of the Kubernetes API server.

- **authenticated** (boolean)

Authenticated indicates that the token was associated with a known user.

- **error** (string)

Error indicates that the token couldn't be checked

- **user** (UserInfo)

User is the UserInfo associated with the provided token.

UserInfo holds the information about the user needed to implement the user.Info interface.

- **user.extra** (map[string][]string)

Any additional information provided by the authenticator.

- **user.groups** ([]string)

The names of groups this user is a part of.

- **user.uid** (string)

A unique value that identifies this user across time. If this user is deleted and another user by the same name is added, they will have different UIDs.

- **user.username** (string)

The name that uniquely identifies this user among all active users.

Operations

create create a TokenReview

HTTP Request

POST /apis/authentication.k8s.io/v1/tokenreviews

Parameters

- **body**: [TokenReview](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([TokenReview](#)): OK

201 ([TokenReview](#)): Created

202 ([TokenReview](#)): Accepted

401: Unauthorized

CertificateSigningRequest

CertificateSigningRequest objects provide a mechanism to obtain x509 certificates by submitting a certificate signing request, and having it asynchronously approved and issued.

```
apiVersion: certificates.k8s.io/v1
```

```
import "k8s.io/api/certificates/v1"
```

CertificateSigningRequest

CertificateSigningRequest objects provide a mechanism to obtain x509 certificates by submitting a certificate signing request, and having it asynchronously approved and issued.

Kubelets use this API to obtain:

1. client certificates to authenticate to kube-apiserver (with the "kubernetes.io/kube-apiserver-client-kubelet" signerName).
2. serving certificates for TLS endpoints kube-apiserver can connect to securely (with the "kubernetes.io/kubelet-serving" signerName).

This API can be used to request client certificates to authenticate to kube-apiserver (with the "kubernetes.io/kube-apiserver-client" signerName), or to obtain certificates from custom non-Kubernetes signers.

- **apiVersion**: certificates.k8s.io/v1
- **kind**: CertificateSigningRequest
- **metadata** ([ObjectMeta](#))
- **spec** ([CertificateSigningRequestSpec](#)), required

spec contains the certificate request, and is immutable after creation. Only the request, signerName, expirationSeconds, and usages fields can be set on creation. Other fields are derived by Kubernetes and cannot be modified by users.

- **status** ([CertificateSigningRequestStatus](#))

status contains information about whether the request is approved or denied, and the certificate issued by the signer, or the failure condition indicating signer failure.

CertificateSigningRequestSpec

CertificateSigningRequestSpec contains the certificate request.

- **request** ([]byte), required

Atomic: will be replaced during a merge

request contains an x509 certificate signing request encoded in a "CERTIFICATE REQUEST" PEM block. When serialized as JSON or YAML, the data is additionally base64-encoded.

- **signerName** (string), required

signerName indicates the requested signer, and is a qualified name.

List/watch requests for CertificateSigningRequests can filter on this field using a "spec.signerName=NAME" fieldSelector.

Well-known Kubernetes signers are:

1. "kubernetes.io/kube-apiserver-client": issues client certificates that can be used to authenticate to kube-apiserver. Requests for this signer are never auto-approved by kube-controller-manager, can be issued by the "csrsigning" controller in kube-controller-manager.
2. "kubernetes.io/kube-apiserver-client-kubelet": issues client certificates that kubelets use to authenticate to kube-apiserver. Requests for this signer can be auto-approved by the "csrapproving" controller in kube-controller-manager, and can be issued by the "csrsigning" controller in kube-controller-manager.
3. "kubernetes.io/kubelet-serving" issues serving certificates that kubelets use to serve TLS endpoints, which kube-apiserver can connect to securely. Requests for this

signer are never auto-approved by kube-controller-manager, and can be issued by the "csrsigning" controller in kube-controller-manager.

More details are available at <https://k8s.io/docs/reference/access-authn-authz/certificate-signing-requests/#kubernetes-signers>

Custom signerNames can also be specified. The signer defines:

1. Trust distribution: how trust (CA bundles) are distributed.
2. Permitted subjects: and behavior when a disallowed subject is requested.
3. Required, permitted, or forbidden x509 extensions in the request (including whether subjectAltNames are allowed, which types, restrictions on allowed values) and behavior when a disallowed extension is requested.
4. Required, permitted, or forbidden key usages / extended key usages.
5. Expiration/certificate lifetime: whether it is fixed by the signer, configurable by the admin.
6. Whether or not requests for CA certificates are allowed.

- **expirationSeconds** (int32)

expirationSeconds is the requested duration of validity of the issued certificate. The certificate signer may issue a certificate with a different validity duration so a client must check the delta between the notBefore and and notAfter fields in the issued certificate to determine the actual duration.

The v1.22+ in-tree implementations of the well-known Kubernetes signers will honor this field as long as the requested duration is not greater than the maximum duration they will honor per the --cluster-signing-duration CLI flag to the Kubernetes controller manager.

Certificate signers may not honor this field for various reasons:

1. Old signer that is unaware of the field (such as the in-tree implementations prior to v1.22)
2. Signer whose configured maximum is shorter than the requested duration
3. Signer whose configured minimum is longer than the requested duration

The minimum valid value for expirationSeconds is 600, i.e. 10 minutes.

- **extra** (map[string][]string)

extra contains extra attributes of the user that created the CertificateSigningRequest. Populated by the API server on creation and immutable.

- **groups** ([]string)

Atomic: will be replaced during a merge

groups contains group membership of the user that created the CertificateSigningRequest. Populated by the API server on creation and immutable.

- **uid** (string)

uid contains the uid of the user that created the CertificateSigningRequest. Populated by the API server on creation and immutable.

usages ([]string)

- *Atomic: will be replaced during a merge*

usages specifies a set of key usages requested in the issued certificate.

Requests for TLS client certificates typically request: "digital signature", "key encipherment", "client auth".

Requests for TLS serving certificates typically request: "key encipherment", "digital signature", "server auth".

Valid values are: "signing", "digital signature", "content commitment", "key encipherment", "key agreement", "data encipherment", "cert sign", "crl sign", "encipher only", "decipher only", "any", "server auth", "client auth", "code signing", "email protection", "s/mime", "ipsec end system", "ipsec tunnel", "ipsec user", "timestamping", "ocsp signing", "microsoft sgc", "netscape sgc"

• username (string)

username contains the name of the user that created the CertificateSigningRequest. Populated by the API server on creation and immutable.

CertificateSigningRequestStatus

CertificateSigningRequestStatus contains conditions used to indicate approved/denied/failed status of the request, and the issued certificate.

• certificate ([]byte)

- *Atomic: will be replaced during a merge*

certificate is populated with an issued certificate by the signer after an Approved condition is present. This field is set via the /status subresource. Once populated, this field is immutable.

If the certificate signing request is denied, a condition of type "Denied" is added and this field remains empty. If the signer cannot issue the certificate, a condition of type "Failed" is added and this field remains empty.

Validation requirements:

1. certificate must contain one or more PEM blocks.
2. All PEM blocks must have the "CERTIFICATE" label, contain no headers, and the encoded data must be a BER-encoded ASN.1 Certificate structure as described in section 4 of RFC5280.
3. Non-PEM content may appear before or after the "CERTIFICATE" PEM blocks and is unvalidated, to allow for explanatory text as described in section 5.2 of RFC7468.

If more than one PEM block is present, and the definition of the requested spec.signerName does not indicate otherwise, the first block is the issued certificate, and subsequent blocks should be treated as intermediate certificates and presented in TLS handshakes.

The certificate is encoded in PEM format.

When serialized as JSON or YAML, the data is additionally base64-encoded, so it consists of:

```
base64(  
----BEGIN CERTIFICATE----  
...  
----END CERTIFICATE----  
)
```

- **conditions** ([]CertificateSigningRequestCondition)

Map: unique values on key type will be kept during a merge

conditions applied to the request. Known conditions are "Approved", "Denied", and "Failed".

CertificateSigningRequestCondition describes a condition of a CertificateSigningRequest object

- **conditions.status** (string), required

status of the condition, one of True, False, Unknown. Approved, Denied, and Failed conditions may not be "False" or "Unknown".

- **conditions.type** (string), required

type of the condition. Known conditions are "Approved", "Denied", and "Failed".

An "Approved" condition is added via the /approval subresource, indicating the request was approved and should be issued by the signer.

A "Denied" condition is added via the /approval subresource, indicating the request was denied and should not be issued by the signer.

A "Failed" condition is added via the /status subresource, indicating the signer failed to issue the certificate.

Approved and Denied conditions are mutually exclusive. Approved, Denied, and Failed conditions cannot be removed once added.

Only one condition of a given type is allowed.

- **conditions.lastTransitionTime** (Time)

lastTransitionTime is the time the condition last transitioned from one status to another. If unset, when a new condition type is added or an existing condition's status is changed, the server defaults this to the current time.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.lastUpdateTime** (Time)

lastUpdateTime is the time of the last update to this condition

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

message contains a human readable message with details about the request state

- **conditions.reason** (string)

reason indicates a brief reason for the request state

CertificateSigningRequestList

CertificateSigningRequestList is a collection of CertificateSigningRequest objects

- **apiVersion**: certificates.k8s.io/v1

- **kind**: CertificateSigningRequestList

- **metadata** ([ListMeta](#))

- **items** ([][CertificateSigningRequest](#)), required

items is a collection of CertificateSigningRequest objects

Operations

get read the specified CertificateSigningRequest

HTTP Request

GET /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}

Parameters

- **name** (*in path*): string, required

name of the CertificateSigningRequest

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

401: Unauthorized

get read approval of the specified CertificateSigningRequest

HTTP Request

GET /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}/approval

Parameters

- **name** (*in path*): string, required
 - name of the CertificateSigningRequest
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

401: Unauthorized

get read status of the specified CertificateSigningRequest

HTTP Request

GET /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the CertificateSigningRequest
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

401: Unauthorized

list list or watch objects of kind CertificateSigningRequest

HTTP Request

GET /apis/certificates.k8s.io/v1/certificatesigningrequests

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([CertificateSigningRequestList](#)): OK

401: Unauthorized

create create a CertificateSigningRequest

HTTP Request

POST /apis/certificates.k8s.io/v1/certificatesigningrequests

Parameters

- **body**: [CertificateSigningRequest](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

201 ([CertificateSigningRequest](#)): Created

202 ([CertificateSigningRequest](#)): Accepted

401: Unauthorized

update replace the specified CertificateSigningRequest

HTTP Request

PUT /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}

Parameters

- **name** (*in path*): string, required
name of the CertificateSigningRequest
- **body**: [CertificateSigningRequest](#), required
- **dryRun** (*in query*): string
[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

201 ([CertificateSigningRequest](#)): Created

401: Unauthorized

update replace approval of the specified CertificateSigningRequest

HTTP Request

PUT /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}/approval

Parameters

- **name** (*in path*): string, required
name of the CertificateSigningRequest
- **body**: [CertificateSigningRequest](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

201 ([CertificateSigningRequest](#)): Created

401: Unauthorized

update replace status of the specified CertificateSigningRequest

HTTP Request

PUT /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}/status

Parameters

- **name** (*in path*): string, required

name of the CertificateSigningRequest

- **body**: [CertificateSigningRequest](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

201 ([CertificateSigningRequest](#)): Created

401: Unauthorized

patch partially update the specified CertificateSigningRequest

HTTP Request

PATCH /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}

Parameters

- **name** (*in path*): string, required

name of the CertificateSigningRequest

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

201 ([CertificateSigningRequest](#)): Created

401: Unauthorized

patch partially update approval of the specified CertificateSigningRequest

HTTP Request

PATCH /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}/approval

Parameters

- **name** (*in path*): string, required

name of the CertificateSigningRequest

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

201 ([CertificateSigningRequest](#)): Created

401: Unauthorized

patch partially update status of the specified CertificateSigningRequest

HTTP Request

PATCH /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}/status

Parameters

- **name** (*in path*): string, required

name of the CertificateSigningRequest
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([CertificateSigningRequest](#)): OK

201 ([CertificateSigningRequest](#)): Created

401: Unauthorized

delete delete a CertificateSigningRequest

HTTP Request

DELETE /apis/certificates.k8s.io/v1/certificatesigningrequests/{name}

Parameters

- **name** (*in path*): string, required
 - name of the CertificateSigningRequest
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of CertificateSigningRequest

HTTP Request

DELETE /apis/certificates.k8s.io/v1/certificatesigningrequests

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

ClusterTrustBundle v1alpha1

ClusterTrustBundle is a cluster-scoped container for X.

```
apiVersion: certificates.k8s.io/v1alpha1
import "k8s.io/api/certificates/v1alpha1"
```

ClusterTrustBundle

ClusterTrustBundle is a cluster-scoped container for X.509 trust anchors (root certificates).

ClusterTrustBundle objects are considered to be readable by any authenticated user in the cluster, because they can be mounted by pods using the clusterTrustBundle projection. All service accounts have read access to ClusterTrustBundles by default. Users who only have namespace-level access to a cluster can read ClusterTrustBundles by impersonating a serviceaccount that they have access to.

It can be optionally associated with a particular signer, in which case it contains one valid set of trust anchors for that signer. Signers may have multiple associated ClusterTrustBundles; each is an independent set of trust anchors for that signer. Admission control is used to enforce that only users with permissions on the signer can create or modify the corresponding bundle.

- **apiVersion:** certificates.k8s.io/v1alpha1

- **kind:** ClusterTrustBundle

- **metadata** ([ObjectMeta](#))

metadata contains the object metadata.

- **spec** ([ClusterTrustBundleSpec](#)), required

spec contains the signer (if any) and trust anchors.

ClusterTrustBundleSpec

ClusterTrustBundleSpec contains the signer and trust anchors.

- **trustBundle** (string), required

trustBundle contains the individual X.509 trust anchors for this bundle, as PEM bundle of PEM-wrapped, DER-formatted X.509 certificates.

The data must consist only of PEM certificate blocks that parse as valid X.509 certificates. Each certificate must include a basic constraints extension with the CA bit set. The API server will reject objects that contain duplicate certificates, or that use PEM block headers.

Users of ClusterTrustBundles, including Kubelet, are free to reorder and deduplicate certificate blocks in this file according to their own logic, as well as to drop PEM block headers and inter-block data.

- **signerName** (string)

signerName indicates the associated signer, if any.

In order to create or update a ClusterTrustBundle that sets signerName, you must have the following cluster-scoped permission: group=certificates.k8s.io resource=signers resourceName=<the signer name> verb=attest.

If signerName is not empty, then the ClusterTrustBundle object must be named with the signer name as a prefix (translating slashes to colons). For example, for the signer name example.com/foo, valid ClusterTrustBundle object names include example.com:foo:abc and example.com:foo:v1.

If signerName is empty, then the ClusterTrustBundle object's name must not have such a prefix.

List/watch requests for ClusterTrustBundles can filter on this field using a spec.signerName=NAME field selector.

ClusterTrustBundleList

ClusterTrustBundleList is a collection of ClusterTrustBundle objects

- **apiVersion**: certificates.k8s.io/v1alpha1

- **kind**: ClusterTrustBundleList

- **metadata** ([ListMeta](#))

metadata contains the list metadata.

- **items** ([][ClusterTrustBundle](#)), required

items is a collection of ClusterTrustBundle objects

Operations

get read the specified ClusterTrustBundle

HTTP Request

GET /apis/certificates.k8s.io/v1alpha1/clustertrustbundles/{name}

Parameters

- **name** (*in path*): string, required

name of the ClusterTrustBundle

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterTrustBundle](#)): OK

401: Unauthorized

list list or watch objects of kind ClusterTrustBundle

HTTP Request

GET /apis/certificates.k8s.io/v1alpha1/clustertrustbundles

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ClusterTrustBundleList](#)): OK

401: Unauthorized

create create a ClusterTrustBundle

HTTP Request

POST /apis/certificates.k8s.io/v1alpha1/clustertrustbundles

Parameters

- **body**: [ClusterTrustBundle](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterTrustBundle](#)): OK

201 ([ClusterTrustBundle](#)): Created

202 ([ClusterTrustBundle](#)): Accepted

401: Unauthorized

update replace the specified ClusterTrustBundle

HTTP Request

PUT /apis/certificates.k8s.io/v1alpha1/clustertrustbundles/{name}

Parameters

- **name** (*in path*): string, required
name of the ClusterTrustBundle
- **body**: [ClusterTrustBundle](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ClusterTrustBundle](#)): OK

201 ([ClusterTrustBundle](#)): Created

401: Unauthorized

patch partially update the specified ClusterTrustBundle

HTTP Request

PATCH /apis/certificates.k8s.io/v1alpha1/clustertrustbundles/{name}

Parameters

- **name** (*in path*): string, required
name of the ClusterTrustBundle
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ClusterTrustBundle](#)): OK

201 ([ClusterTrustBundle](#)): Created

401: Unauthorized

delete delete a ClusterTrustBundle

HTTP Request

DELETE /apis/certificates.k8s.io/v1alpha1/clustertrustbundles/{name}

Parameters

- **name** (*in path*): string, required
name of the ClusterTrustBundle
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ClusterTrustBundle

HTTP Request

DELETE /apis/certificates.k8s.io/v1alpha1/clustertrustbundles

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

SelfSubjectReview

SelfSubjectReview contains the user information that the kube-apiserver has about the user making this request.

```
apiVersion: authentication.k8s.io/v1  
import "k8s.io/api/authentication/v1"
```

SelfSubjectReview

SelfSubjectReview contains the user information that the kube-apiserver has about the user making this request. When using impersonation, users will receive the user info of the user being impersonated. If impersonation or request header authentication is used, any extra keys will have their case ignored and returned as lowercase.

-
- **apiVersion**: authentication.k8s.io/v1
 - **kind**: SelfSubjectReview
 - **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **status** ([SelfSubjectReviewStatus](#))

Status is filled in by the server with the user attributes.

SelfSubjectReviewStatus

SelfSubjectReviewStatus is filled by the kube-apiserver and sent back to a user.

- **userInfo** (UserInfo)

User attributes of the user making this request.

UserInfo holds the information about the user needed to implement the user.Info interface.

- **userInfo.extra** (map[string][]string)

Any additional information provided by the authenticator.

- **userInfo.groups** ([]string)

The names of groups this user is a part of.

- **userInfo.uid** (string)

A unique value that identifies this user across time. If this user is deleted and another user by the same name is added, they will have different UIDs.

- **userInfo.username** (string)

The name that uniquely identifies this user among all active users.

Operations

create create a SelfSubjectReview

HTTP Request

POST /apis/authentication.k8s.io/v1/selfsubjectreviews

Parameters

- **body**: [SelfSubjectReview](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

pretty (*in query*): string

- [pretty](#)

Response

200 ([SelfSubjectReview](#)): OK

201 ([SelfSubjectReview](#)): Created

202 ([SelfSubjectReview](#)): Accepted

401: Unauthorized

Authorization Resources

[LocalSubjectAccessReview](#)

LocalSubjectAccessReview checks whether or not a user or group can perform an action in a given namespace.

[SelfSubjectAccessReview](#)

SelfSubjectAccessReview checks whether or the current user can perform an action.

[SelfSubjectRulesReview](#)

SelfSubjectRulesReview enumerates the set of actions the current user can perform within a namespace.

[SubjectAccessReview](#)

SubjectAccessReview checks whether or not a user or group can perform an action.

[ClusterRole](#)

ClusterRole is a cluster level, logical grouping of PolicyRules that can be referenced as a unit by a RoleBinding or ClusterRoleBinding.

[ClusterRoleBinding](#)

ClusterRoleBinding references a ClusterRole, but not contain it.

[Role](#)

Role is a namespaced, logical grouping of PolicyRules that can be referenced as a unit by a RoleBinding.

[RoleBinding](#)

RoleBinding references a role, but does not contain it.

LocalSubjectAccessReview

LocalSubjectAccessReview checks whether or not a user or group can perform an action in a given namespace.

```
apiVersion: authorization.k8s.io/v1
```

```
import "k8s.io/api/authorization/v1"
```

LocalSubjectAccessReview

LocalSubjectAccessReview checks whether or not a user or group can perform an action in a given namespace. Having a namespace scoped resource makes it much easier to grant namespace scoped policy that includes permissions checking.

- **apiVersion:** authorization.k8s.io/v1

- **kind:** LocalSubjectAccessReview

- **metadata** ([ObjectMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([SubjectAccessReviewSpec](#)), required

Spec holds information about the request being evaluated. spec.namespace must be equal to the namespace you made the request against. If empty, it is defaulted.

- **status** ([SubjectAccessReviewStatus](#))

Status is filled in by the server and indicates whether the request is allowed or not

Operations

create create a LocalSubjectAccessReview

HTTP Request

```
POST /apis/authorization.k8s.io/v1/namespaces/{namespace}/localsubjectaccessreviews
```

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [LocalSubjectAccessReview](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([LocalSubjectAccessReview](#)): OK

201 ([LocalSubjectAccessReview](#)): Created

202 ([LocalSubjectAccessReview](#)): Accepted

401: Unauthorized

SelfSubjectAccessReview

SelfSubjectAccessReview checks whether or the current user can perform an action.

```
apiVersion: authorization.k8s.io/v1
```

```
import "k8s.io/api/authorization/v1"
```

SelfSubjectAccessReview

SelfSubjectAccessReview checks whether or the current user can perform an action. Not filling in a spec.namespace means "in all namespaces". Self is a special case, because users should always be able to check whether they can perform an action

-
- **apiVersion**: authorization.k8s.io/v1

- **kind**: SelfSubjectAccessReview

- **metadata** ([ObjectMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

spec ([SelfSubjectAccessReviewSpec](#)), required

- Spec holds information about the request being evaluated. user and groups must be empty

- **status** ([SubjectAccessReviewStatus](#))

Status is filled in by the server and indicates whether the request is allowed or not

SelfSubjectAccessReviewSpec

SelfSubjectAccessReviewSpec is a description of the access request. Exactly one of ResourceAuthorizationAttributes and NonResourceAuthorizationAttributes must be set

- **nonResourceAttributes** (NonResourceAttributes)

NonResourceAttributes describes information for a non-resource access request

NonResourceAttributes includes the authorization attributes available for non-resource requests to the Authorizer interface

- **nonResourceAttributes.path** (string)

Path is the URL path of the request

- **nonResourceAttributes.verb** (string)

Verb is the standard HTTP verb

- **resourceAttributes** (ResourceAttributes)

ResourceAuthorizationAttributes describes information for a resource access request

ResourceAttributes includes the authorization attributes available for resource requests to the Authorizer interface

- **resourceAttributes.group** (string)

Group is the API Group of the Resource. "*" means all.

- **resourceAttributes.name** (string)

Name is the name of the resource being requested for a "get" or deleted for a "delete". "" (empty) means all.

- **resourceAttributes.namespace** (string)

Namespace is the namespace of the action being requested. Currently, there is no distinction between no namespace and all namespaces "" (empty) is defaulted for LocalSubjectAccessReviews "" (empty) is empty for cluster-scoped resources "" (empty) means "all" for namespace scoped resources from a SubjectAccessReview or SelfSubjectAccessReview

- **resourceAttributes.resource** (string)

Resource is one of the existing resource types. "*" means all.

- **resourceAttributes.subresource** (string)

Subresource is one of the existing resource types. "" means none.

- **resourceAttributes.verb** (string)

Verb is a kubernetes resource API verb, like: get, list, watch, create, update, delete, proxy. "*" means all.

- **resourceAttributes.version** (string)

Version is the API Version of the Resource. "*" means all.

Operations

create create a SelfSubjectAccessReview

HTTP Request

POST /apis/authorization.k8s.io/v1/selfsubjectaccessreviews

Parameters

- **body**: [SelfSubjectAccessReview](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([SelfSubjectAccessReview](#)): OK

201 ([SelfSubjectAccessReview](#)): Created

202 ([SelfSubjectAccessReview](#)): Accepted

401: Unauthorized

SelfSubjectRulesReview

SelfSubjectRulesReview enumerates the set of actions the current user can perform within a namespace.

```
apiVersion: authorization.k8s.io/v1  
import "k8s.io/api/authorization/v1"
```

SelfSubjectRulesReview

SelfSubjectRulesReview enumerates the set of actions the current user can perform within a namespace. The returned list of actions may be incomplete depending on the server's authorization mode, and any errors experienced during the evaluation. SelfSubjectRulesReview should be used by UIs to show/hide actions, or to quickly let an end user reason about their permissions. It should NOT Be used by external systems to drive authorization decisions as this raises confused deputy, cache lifetime/revocation, and correctness concerns.

SubjectAccessReview, and LocalAccessReview are the correct way to defer authorization decisions to the API server.

- **apiVersion:** authorization.k8s.io/v1

- **kind:** SelfSubjectRulesReview

- **metadata** ([ObjectMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([SelfSubjectRulesReviewSpec](#)), required

Spec holds information about the request being evaluated.

- **status** ([SubjectRulesReviewStatus](#))

Status is filled in by the server and indicates the set of actions a user can perform.

SubjectRulesReviewStatus contains the result of a rules check. This check can be incomplete depending on the set of authorizers the server is configured with and any errors experienced during evaluation. Because authorization rules are additive, if a rule appears in a list it's safe to assume the subject has that permission, even if that list is incomplete.

- **status.incomplete** (boolean), required

Incomplete is true when the rules returned by this call are incomplete. This is most commonly encountered when an authorizer, such as an external authorizer, doesn't support rules evaluation.

- **status.nonResourceRules** ([]NonResourceRule), required

NonResourceRules is the list of actions the subject is allowed to perform on non-resources. The list ordering isn't significant, may contain duplicates, and possibly be incomplete.

NonResourceRule holds information that describes a rule for the non-resource

- **status.nonResourceRules.verbs** ([]string), required

Verb is a list of kubernetes non-resource API verbs, like: get, post, put, delete, patch, head, options. "*" means all.

- **status.nonResourceRules.nonResourceURLs** ([]string)

NonResourceURLs is a set of partial urls that a user should have access to. *s are allowed, but only as the full, final step in the path. "" means all.*

- **status.resourceRules** ([]ResourceRule), required

ResourceRules is the list of actions the subject is allowed to perform on resources. The list ordering isn't significant, may contain duplicates, and possibly be incomplete.

ResourceRule is the list of actions the subject is allowed to perform on resources. The list ordering isn't significant, may contain duplicates, and possibly be incomplete.

- **status.resourceRules.verbs** ([]string), required

Verb is a list of kubernetes resource API verbs, like: get, list, watch, create, update, delete, proxy. "*" means all.

- **status.resourceRules.apiGroups** ([]string)

APIGroups is the name of the APIGroup that contains the resources. If multiple API groups are specified, any action requested against one of the enumerated resources in any API group will be allowed. "*" means all.

- **status.resourceRules.resourceNames** ([]string)

ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed. "*" means all.

- **status.resourceRules.resources** ([]string)

Resources is a list of resources this rule applies to. *"" means all in the specified apiGroups. "/foo" represents the subresource 'foo' for all resources in the specified apiGroups.*

- **status.evaluationError** (string)

EvaluationError can appear in combination with Rules. It indicates an error occurred during rule evaluation, such as an authorizer that doesn't support rule evaluation, and that ResourceRules and/or NonResourceRules may be incomplete.

SelfSubjectRulesReviewSpec

SelfSubjectRulesReviewSpec defines the specification for SelfSubjectRulesReview.

- **namespace** (string)

Namespace to evaluate rules for. Required.

Operations

create create a SelfSubjectRulesReview

HTTP Request

POST /apis/authorization.k8s.io/v1/selfsubjectrulesreviews

Parameters

- **body**: [SelfSubjectRulesReview](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([SelfSubjectRulesReview](#)): OK

201 ([SelfSubjectRulesReview](#)): Created

202 ([SelfSubjectRulesReview](#)): Accepted

401: Unauthorized

SubjectAccessReview

SubjectAccessReview checks whether or not a user or group can perform an action.

```
apiVersion: authorization.k8s.io/v1
```

```
import "k8s.io/api/authorization/v1"
```

SubjectAccessReview

SubjectAccessReview checks whether or not a user or group can perform an action.

- **apiVersion:** authorization.k8s.io/v1

- **kind:** SubjectAccessReview

- **metadata** ([ObjectMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([SubjectAccessReviewSpec](#)), required

Spec holds information about the request being evaluated

- **status** ([SubjectAccessReviewStatus](#))

Status is filled in by the server and indicates whether the request is allowed or not

SubjectAccessReviewSpec

SubjectAccessReviewSpec is a description of the access request. Exactly one of ResourceAuthorizationAttributes and NonResourceAuthorizationAttributes must be set

- **extra** (map[string][]string)

Extra corresponds to the user.Info.GetExtra() method from the authenticator. Since that is input to the authorizer it needs a reflection here.

- **groups** ([]string)

Groups is the groups you're testing for.

- **nonResourceAttributes** (NonResourceAttributes)

NonResourceAttributes describes information for a non-resource access request

NonResourceAttributes includes the authorization attributes available for non-resource requests to the Authorizer interface

- **nonResourceAttributes.path** (string)

Path is the URL path of the request

- **nonResourceAttributes.verb** (string)

Verb is the standard HTTP verb

- **resourceAttributes** (ResourceAttributes)

ResourceAuthorizationAttributes describes information for a resource access request

ResourceAttributes includes the authorization attributes available for resource requests to the Authorizer interface

- **resourceAttributes.group** (string)

Group is the API Group of the Resource. "*" means all.

- **resourceAttributes.name** (string)

Name is the name of the resource being requested for a "get" or deleted for a "delete". "" (empty) means all.

- **resourceAttributes.namespace** (string)

Namespace is the namespace of the action being requested. Currently, there is no distinction between no namespace and all namespaces "" (empty) is defaulted for LocalSubjectAccessReviews "" (empty) is empty for cluster-scoped resources "" (empty) means "all" for namespace scoped resources from a SubjectAccessReview or SelfSubjectAccessReview

- **resourceAttributes.resource** (string)

Resource is one of the existing resource types. "*" means all.

- **resourceAttributes.subresource** (string)

Subresource is one of the existing resource types. "" means none.

- **resourceAttributes.verb** (string)

Verb is a kubernetes resource API verb, like: get, list, watch, create, update, delete, proxy. "*" means all.

- **resourceAttributes.version** (string)

Version is the API Version of the Resource. "*" means all.

- **uid** (string)

UID information about the requesting user.

- **user** (string)

User is the user you're testing for. If you specify "User" but not "Groups", then is it interpreted as "What if User were not a member of any groups"

SubjectAccessReviewStatus

SubjectAccessReviewStatus

- **allowed** (boolean), required

Allowed is required. True if the action would be allowed, false otherwise.

- **denied** (boolean)

Denied is optional. True if the action would be denied, otherwise false. If both allowed is false and denied is false, then the authorizer has no opinion on whether to authorize the action. Denied may not be true if Allowed is true.

- **evaluationError** (string)

EvaluationError is an indication that some error occurred during the authorization check. It is entirely possible to get an error and be able to continue determine authorization status in spite of it. For instance, RBAC can be missing a role, but enough roles are still present and bound to reason about the request.

- **reason** (string)

Reason is optional. It indicates why a request was allowed or denied.

Operations

create create a SubjectAccessReview

HTTP Request

POST /apis/authorization.k8s.io/v1/subjectaccessreviews

Parameters

- **body**: [SubjectAccessReview](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([SubjectAccessReview](#)): OK

201 ([SubjectAccessReview](#)): Created

202 ([SubjectAccessReview](#)): Accepted

401: Unauthorized

ClusterRole

ClusterRole is a cluster level, logical grouping of PolicyRules that can be referenced as a unit by a RoleBinding or ClusterRoleBinding.

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
import "k8s.io/api/rbac/v1"
```

ClusterRole

ClusterRole is a cluster level, logical grouping of PolicyRules that can be referenced as a unit by a RoleBinding or ClusterRoleBinding.

- **apiVersion:** rbac.authorization.k8s.io/v1

- **kind:** ClusterRole

- **metadata** ([ObjectMeta](#))

Standard object's metadata.

- **aggregationRule** ([AggregationRule](#))

AggregationRule is an optional field that describes how to build the Rules for this ClusterRole. If AggregationRule is set, then the Rules are controller managed and direct changes to Rules will be stomped by the controller.

AggregationRule describes how to locate ClusterRoles to aggregate into the ClusterRole

- **aggregationRule.clusterRoleSelectors** ([] [LabelSelector](#))

ClusterRoleSelectors holds a list of selectors which will be used to find ClusterRoles and create the rules. If any of the selectors match, then the ClusterRole's permissions will be added

- **rules** ([]PolicyRule)

Rules holds all the PolicyRules for this ClusterRole

PolicyRule holds information that describes a policy rule, but does not contain information about who the rule applies to or which namespace the rule applies to.

- **rules.apiGroups** ([]string)

APIGroups is the name of the APIGroup that contains the resources. If multiple API groups are specified, any action requested against one of the enumerated resources in any API group will be allowed. "" represents the core API group and "*" represents all API groups.

- **rules.resources** ([]string)

Resources is a list of resources this rule applies to. "*" represents all resources.

- **rules.verbs** ([]string), required

Verbs is a list of Verbs that apply to ALL the ResourceKinds contained in this rule. "*" represents all verbs.

- **rules.resourceNames** ([]string)

ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed.

- **rules.nonResourceURLs** ([]string)

NonResourceURLs is a set of partial urls that a user should have access to. *'s are allowed, but only as the full, final step in the path Since non-resource URLs are not namespaced, this field is only applicable for ClusterRoles referenced from a ClusterRoleBinding. Rules can either apply to API resources (such as "pods" or "secrets") or non-resource URL paths (such as "/api"), but not both.

ClusterRoleList

ClusterRoleList is a collection of ClusterRoles

- **apiVersion:** rbac.authorization.k8s.io/v1

- **kind:** ClusterRoleList

- **metadata** ([ListMeta](#))

Standard object's metadata.

- **items** ([][ClusterRole](#)), required

Items is a list of ClusterRoles

Operations

get read the specified ClusterRole

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/clusterroles/{name}

Parameters

- **name** (*in path*): string, required

name of the ClusterRole

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterRole](#)): OK

401: Unauthorized

list list or watch objects of kind ClusterRole

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/clusterroles

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ClusterRoleList](#)): OK

401: Unauthorized

create create a ClusterRole

HTTP Request

POST /apis/rbac.authorization.k8s.io/v1/clusterroles

Parameters

- **body**: [ClusterRole](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ClusterRole](#)): OK

201 ([ClusterRole](#)): Created

202 ([ClusterRole](#)): Accepted

401: Unauthorized

update replace the specified ClusterRole

HTTP Request

PUT /apis/rbac.authorization.k8s.io/v1/clusterroles/{name}

Parameters

- **name** (*in path*): string, required

name of the ClusterRole

- **body**: [ClusterRole](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterRole](#)): OK

201 ([ClusterRole](#)): Created

401: Unauthorized

patch partially update the specified ClusterRole

HTTP Request

PATCH /apis/rbac.authorization.k8s.io/v1/clusterroles/{name}

Parameters

- **name** (*in path*): string, required
name of the ClusterRole
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ClusterRole](#)): OK

201 ([ClusterRole](#)): Created

401: Unauthorized

delete delete a ClusterRole

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/clusterroles/{name}

Parameters

- **name** (*in path*): string, required
name of the ClusterRole
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ClusterRole

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/clusterroles

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)

- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ClusterRoleBinding

ClusterRoleBinding references a ClusterRole, but not contain it.

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
import "k8s.io/api/rbac/v1"
```

ClusterRoleBinding

ClusterRoleBinding references a ClusterRole, but not contain it. It can reference a ClusterRole in the global namespace, and adds who information via Subject.

-
- **apiVersion**: rbac.authorization.k8s.io/v1
 - **kind**: ClusterRoleBinding
 - **metadata** ([ObjectMeta](#))
 - Standard object's metadata.
 - **roleRef** (RoleRef), required

RoleRef can only reference a ClusterRole in the global namespace. If the RoleRef cannot be resolved, the Authorizer must return an error. This field is immutable.

RoleRef contains information that points to the role being used

- **roleRef.apiGroup** (string), required

APIGroup is the group for the resource being referenced

- **roleRef.kind** (string), required

Kind is the type of resource being referenced

- **roleRef.name** (string), required

Name is the name of resource being referenced

- **subjects** ([]Subject)

Subjects holds references to the objects the role applies to.

Subject contains a reference to the object or user identities a role binding applies to. This can either hold a direct API object reference, or a value for non-objects such as user and group names.

- **subjects.kind** (string), required

Kind of object being referenced. Values defined by this API group are "User", "Group", and "ServiceAccount". If the Authorizer does not recognize the kind value, the Authorizer should report an error.

- **subjects.name** (string), required

Name of the object being referenced.

- **subjects.apiGroup** (string)

APIGroup holds the API group of the referenced subject. Defaults to "" for ServiceAccount subjects. Defaults to "rbac.authorization.k8s.io" for User and Group subjects.

- **subjects.namespace** (string)

Namespace of the referenced object. If the object kind is non-namespace, such as "User" or "Group", and this value is not empty the Authorizer should report an error.

ClusterRoleBindingList

ClusterRoleBindingList is a collection of ClusterRoleBindings

- **apiVersion**: rbac.authorization.k8s.io/v1
- **kind**: ClusterRoleBindingList

metadata ([ListMeta](#))

- Standard object's metadata.
- **items** ([][ClusterRoleBinding](#)), required
 - Items is a list of ClusterRoleBindings

Operations

get read the specified ClusterRoleBinding

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ClusterRoleBinding
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ClusterRoleBinding](#)): OK

401: Unauthorized

list list or watch objects of kind ClusterRoleBinding

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/clusterrolebindings

Parameters

- **allowWatchBookmarks** (*in query*): boolean
 - [allowWatchBookmarks](#)
- **continue** (*in query*): string
 - [continue](#)
- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ClusterRoleBindingList](#)): OK

401: Unauthorized

create create a ClusterRoleBinding

HTTP Request

POST /apis/rbac.authorization.k8s.io/v1/clusterrolebindings

Parameters

- **body**: [ClusterRoleBinding](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ClusterRoleBinding](#)): OK

201 ([ClusterRoleBinding](#)): Created

202 ([ClusterRoleBinding](#)): Accepted

401: Unauthorized

update replace the specified ClusterRoleBinding

HTTP Request

PUT /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ClusterRoleBinding
- **body**: [ClusterRoleBinding](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ClusterRoleBinding](#)): OK

201 ([ClusterRoleBinding](#)): Created

401: Unauthorized

patch partially update the specified ClusterRoleBinding

HTTP Request

PATCH /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/{name}

Parameters

- **name** (*in path*): string, required
name of the ClusterRoleBinding
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ClusterRoleBinding](#)): OK

201 ([ClusterRoleBinding](#)): Created

401: Unauthorized

delete delete a ClusterRoleBinding

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/clusterrolebindings/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ClusterRoleBinding
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ClusterRoleBinding

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/clusterrolebindings

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Role

Role is a namespaced, logical grouping of PolicyRules that can be referenced as a unit by a RoleBinding.

apiVersion: rbac.authorization.k8s.io/v1

```
import "k8s.io/api/rbac/v1"
```

Role

Role is a namespaced, logical grouping of PolicyRules that can be referenced as a unit by a RoleBinding.

- **apiVersion:** rbac.authorization.k8s.io/v1

- **kind:** Role

- **metadata** ([ObjectMeta](#))

Standard object's metadata.

- **rules** ([]PolicyRule)

Rules holds all the PolicyRules for this Role

PolicyRule holds information that describes a policy rule, but does not contain information about who the rule applies to or which namespace the rule applies to.

- **rules.apiGroups** ([]string)

APIGroups is the name of the APIGroup that contains the resources. If multiple API groups are specified, any action requested against one of the enumerated resources in any API group will be allowed. "" represents the core API group and "*" represents all API groups.

- **rules.resources** ([]string)

Resources is a list of resources this rule applies to. '*' represents all resources.

- **rules.verbs** ([]string), required

Verbs is a list of Verbs that apply to ALL the ResourceKinds contained in this rule. '*' represents all verbs.

- **rules.resourceNames** ([]string)

ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed.

- **rules.nonResourceURLs** ([]string)

NonResourceURLs is a set of partial urls that a user should have access to. *'s are allowed, but only as the full, final step in the path Since non-resource URLs are not namespaced, this field is only applicable for ClusterRoles referenced from a ClusterRoleBinding. Rules can either apply to API resources (such as "pods" or "secrets") or non-resource URL paths (such as "/api"), but not both.

RoleList

RoleList is a collection of Roles

- **apiVersion:** rbac.authorization.k8s.io/v1

- **kind:** RoleList

- **metadata** ([ListMeta](#))

Standard object's metadata.

- **items** ([][Role](#)), required

Items is a list of Roles

Operations

get read the specified Role

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/roles/{name}

Parameters

- **name** (*in path*): string, required

name of the Role

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Role](#)): OK

401: Unauthorized

list list or watch objects of kind Role

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/roles

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([RoleList](#)): OK

401: Unauthorized

list list or watch objects of kind Role

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/roles

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([RoleList](#)): OK

401: Unauthorized

create create a Role

HTTP Request

POST /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/roles

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Role](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Role](#)): OK

201 ([Role](#)): Created

202 ([Role](#)): Accepted

401: Unauthorized

update replace the specified Role

HTTP Request

PUT /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/roles/{name}

Parameters

- **name** (*in path*): string, required
name of the Role
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Role](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Role](#)): OK

201 ([Role](#)): Created

401: Unauthorized

patch partially update the specified Role

HTTP Request

PATCH /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/roles/{name}

Parameters

- **name** (*in path*): string, required
name of the Role
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required

- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Role](#)): OK

201 ([Role](#)): Created

401: Unauthorized

delete delete a Role

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/roles/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Role
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)

- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Role

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/roles

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)

- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

RoleBinding

RoleBinding references a role, but does not contain it.

```
apiVersion: rbac.authorization.k8s.io/v1
```

```
import "k8s.io/api/rbac/v1"
```

RoleBinding

RoleBinding references a role, but does not contain it. It can reference a Role in the same namespace or a ClusterRole in the global namespace. It adds who information via Subjects and namespace information by which namespace it exists in. RoleBindings in a given namespace only have effect in that namespace.

-
- **apiVersion**: rbac.authorization.k8s.io/v1
 - **kind**: RoleBinding
 - **metadata** ([ObjectMeta](#))

Standard object's metadata.

roleRef (RoleRef), required

- RoleRef can reference a Role in the current namespace or a ClusterRole in the global namespace. If the RoleRef cannot be resolved, the Authorizer must return an error. This field is immutable.

RoleRef contains information that points to the role being used

◦ **roleRef.apiGroup** (string), required

APIGroup is the group for the resource being referenced

◦ **roleRef.kind** (string), required

Kind is the type of resource being referenced

◦ **roleRef.name** (string), required

Name is the name of resource being referenced

• **subjects** ([]Subject)

Subjects holds references to the objects the role applies to.

Subject contains a reference to the object or user identities a role binding applies to. This can either hold a direct API object reference, or a value for non-objects such as user and group names.

◦ **subjects.kind** (string), required

Kind of object being referenced. Values defined by this API group are "User", "Group", and "ServiceAccount". If the Authorizer does not recognize the kind value, the Authorizer should report an error.

◦ **subjects.name** (string), required

Name of the object being referenced.

◦ **subjects.apiGroup** (string)

APIGroup holds the API group of the referenced subject. Defaults to "" for ServiceAccount subjects. Defaults to "rbac.authorization.k8s.io" for User and Group subjects.

◦ **subjects.namespace** (string)

Namespace of the referenced object. If the object kind is non-namespace, such as "User" or "Group", and this value is not empty the Authorizer should report an error.

RoleBindingList

RoleBindingList is a collection of RoleBindings

-
- **apiVersion**: rbac.authorization.k8s.io/v1
 - **kind**: RoleBindingList
 - **metadata** ([ListMeta](#))

Standard object's metadata.

- **items** ([][RoleBinding](#)), required

Items is a list of RoleBindings

Operations

get read the specified RoleBinding

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/rolebindings/{name}

Parameters

- **name** (*in path*): string, required
 - name of the RoleBinding
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([RoleBinding](#)): OK

401: Unauthorized

list list or watch objects of kind RoleBinding

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/rolebindings

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([RoleBindingList](#)): OK

401: Unauthorized

list list or watch objects of kind RoleBinding

HTTP Request

GET /apis/rbac.authorization.k8s.io/v1/rolebindings

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([RoleBindingList](#)): OK

401: Unauthorized

create create a RoleBinding

HTTP Request

POST /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/rolebindings

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [RoleBinding](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([RoleBinding](#)): OK

201 ([RoleBinding](#)): Created

202 ([RoleBinding](#)): Accepted

401: Unauthorized

update replace the specified RoleBinding

HTTP Request

PUT /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/rolebindings/{name}

Parameters

- **name** (*in path*): string, required
name of the RoleBinding
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [RoleBinding](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([RoleBinding](#)): OK

201 ([RoleBinding](#)): Created

401: Unauthorized

patch partially update the specified RoleBinding

HTTP Request

PATCH /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/rolebindings/{name}

Parameters

- **name** (*in path*): string, required
name of the RoleBinding
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required

- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([RoleBinding](#)): OK

201 ([RoleBinding](#)): Created

401: Unauthorized

delete delete a RoleBinding

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/rolebindings/{name}

Parameters

- **name** (*in path*): string, required
 - name of the RoleBinding
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)

- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of RoleBinding

HTTP Request

DELETE /apis/rbac.authorization.k8s.io/v1/namespaces/{namespace}/rolebindings

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)

- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Policy Resources

[LimitRange](#)

LimitRange sets resource usage limits for each kind of resource in a Namespace.

[ResourceQuota](#)

ResourceQuota sets aggregate quota restrictions enforced per namespace.

[NetworkPolicy](#)

NetworkPolicy describes what network traffic is allowed for a set of Pods.

[PodDisruptionBudget](#)

PodDisruptionBudget is an object to define the max disruption that can be caused to a collection of pods.

[IPAddress v1alpha1](#)

IPAddress represents a single IP of a single IP Family.

LimitRange

LimitRange sets resource usage limits for each kind of resource in a Namespace.

apiVersion: v1

```
import "k8s.io/api/core/v1"
```

LimitRange

LimitRange sets resource usage limits for each kind of resource in a Namespace.

- **apiVersion:** v1
- **kind:** LimitRange
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([LimitRangeSpec](#))

Spec defines the limits enforced. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

LimitRangeSpec

LimitRangeSpec defines a min/max usage limit for resources that match on kind.

- **limits** ([]LimitRangeItem), required

Limits is the list of LimitRangeItem objects that are enforced.

LimitRangeItem defines a min/max usage limit for any resource that matches on kind.

- **limits.type** (string), required

Type of resource that this limit applies to.

- **limits.default** (map[string][Quantity](#))

Default resource requirement limit value by resource name if resource limit is omitted.

- **limits.defaultRequest** (map[string][Quantity](#))

DefaultRequest is the default resource requirement request value by resource name if resource request is omitted.

- **limits.max** (map[string][Quantity](#))

Max usage constraints on this kind by resource name.

- **limits.maxLimitRequestRatio** (map[string][Quantity](#))

MaxLimitRequestRatio if specified, the named resource must have a request and limit that are both non-zero where limit divided by request is less than or equal to the enumerated value; this represents the max burst for the named resource.

- **limits.min** (map[string][Quantity](#))

Min usage constraints on this kind by resource name.

LimitRangeList

LimitRangeList is a list of LimitRange items.

- **apiVersion:** v1
- **kind:** LimitRangeList
- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][LimitRange](#)), required

Items is a list of LimitRange objects. More info: <https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/>

Operations

get read the specified LimitRange

HTTP Request

GET /api/v1/namespaces/{namespace}/limitranges/{name}

Parameters

- **name** (*in path*): string, required

name of the LimitRange

- **namespace** (*in path*): string, required

[namespace](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([LimitRange](#)): OK

401: Unauthorized

list list or watch objects of kind LimitRange

HTTP Request

GET /api/v1/namespaces/{namespace}/limitranges

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([LimitRangeList](#)): OK

401: Unauthorized

list list or watch objects of kind LimitRange

HTTP Request

GET /api/v1/limitranges

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([LimitRangeList](#)): OK

401: Unauthorized

create create a LimitRange

HTTP Request

POST /api/v1/namespaces/{namespace}/limitranges

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [LimitRange](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([LimitRange](#)): OK

201 ([LimitRange](#)): Created

202 ([LimitRange](#)): Accepted

401: Unauthorized

update replace the specified LimitRange

HTTP Request

PUT /api/v1/namespaces/{namespace}/limitranges/{name}

Parameters

- **name** (*in path*): string, required

name of the LimitRange

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [LimitRange](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([LimitRange](#)): OK

201 ([LimitRange](#)): Created

401: Unauthorized

patch partially update the specified LimitRange

HTTP Request

PATCH /api/v1/namespaces/{namespace}/limitranges/{name}

Parameters

- **name** (*in path*): string, required
 - name of the LimitRange
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([LimitRange](#)): OK

201 ([LimitRange](#)): Created

401: Unauthorized

delete delete a LimitRange

HTTP Request

DELETE /api/v1/namespaces/{namespace}/limitranges/{name}

Parameters

- **name** (*in path*): string, required
name of the LimitRange
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of LimitRange

HTTP Request

DELETE /api/v1/namespaces/{namespace}/limitranges

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)

- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ResourceQuota

ResourceQuota sets aggregate quota restrictions enforced per namespace.

```
apiVersion: v1  
import "k8s.io/api/core/v1"
```

ResourceQuota

ResourceQuota sets aggregate quota restrictions enforced per namespace

- **apiVersion:** v1
- **kind:** ResourceQuota
- **metadata** ([ObjectMeta](#))
Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>
- **spec** ([ResourceQuotaSpec](#))
Spec defines the desired quota. <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>
- **status** ([ResourceQuotaStatus](#))
Status defines the actual enforced quota and its current usage. <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

ResourceQuotaSpec

ResourceQuotaSpec defines the desired hard limits to enforce for Quota.

- **hard** (map[string][Quantity](#))
hard is the set of desired hard limits for each named resource. More info: <https://kubernetes.io/docs/concepts/policy/resource-quotas/>
- **scopeSelector** (ScopeSelector)
scopeSelector is also a collection of filters like scopes that must match each object tracked by a quota but expressed using ScopeSelectorOperator in combination with possible values. For a resource to match, both scopes AND scopeSelector (if specified in spec), must be matched.
A scope selector represents the AND of the selectors represented by the scoped-resource selector requirements.
 - **scopeSelector.matchExpressions** ([]ScopedResourceSelectorRequirement)
A list of scope selector requirements by scope of the resources.

A *scoped-resource selector requirement* is a selector that contains values, a scope name, and an operator that relates the scope name and values.

- **scopeSelector.matchExpressions.operator** (string), required

Represents a scope's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist.

- **scopeSelector.matchExpressions.scopeName** (string), required

The name of the scope that the selector applies to.

- **scopeSelector.matchExpressions.values** ([]string)

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

- **scopes** ([]string)

A collection of filters that must match each object tracked by a quota. If not specified, the quota matches all objects.

ResourceQuotaStatus

ResourceQuotaStatus defines the enforced hard limits and observed use.

- **hard** (map[string][Quantity](#))

Hard is the set of enforced hard limits for each named resource. More info: <https://kubernetes.io/docs/concepts/policy/resource-quotas/>

- **used** (map[string][Quantity](#))

Used is the current observed total usage of the resource in the namespace.

ResourceQuotaList

ResourceQuotaList is a list of ResourceQuota items.

- **apiVersion:** v1

- **kind:** ResourceQuotaList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][ResourceQuota](#)), required

Items is a list of ResourceQuota objects. More info: <https://kubernetes.io/docs/concepts/policy/resource-quotas/>

Operations

get read the specified ResourceQuota

HTTP Request

GET /api/v1/namespaces/{namespace}/resourcequotas/{name}

Parameters

- **name** (*in path*): string, required
name of the ResourceQuota
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceQuota](#)): OK

401: Unauthorized

get read status of the specified ResourceQuota

HTTP Request

GET /api/v1/namespaces/{namespace}/resourcequotas/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ResourceQuota
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceQuota](#)): OK

401: Unauthorized

list list or watch objects of kind ResourceQuota

HTTP Request

GET /api/v1/namespaces/{namespace}/resourcequotas

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

timeoutSeconds (*in query*): integer

- [timeoutSeconds](#)

• **watch** (*in query*): boolean

[watch](#)

Response

200 ([ResourceQuotaList](#)): OK

401: Unauthorized

list list or watch objects of kind ResourceQuota

HTTP Request

GET /api/v1/resourcequotas

Parameters

• **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

• **continue** (*in query*): string

[continue](#)

• **fieldSelector** (*in query*): string

[fieldSelector](#)

• **labelSelector** (*in query*): string

[labelSelector](#)

• **limit** (*in query*): integer

[limit](#)

• **pretty** (*in query*): string

[pretty](#)

• **resourceVersion** (*in query*): string

[resourceVersion](#)

• **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

sendInitialEvents (*in query*): boolean

•

[sendInitialEvents](#)

• **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

• **watch** (*in query*): boolean

[watch](#)

Response

200 ([ResourceQuotaList](#)): OK

401: Unauthorized

create create a ResourceQuota

HTTP Request

POST /api/v1/namespaces/{namespace}/resourcequotas

Parameters

• **namespace** (*in path*): string, required

[namespace](#)

• **body**: [ResourceQuota](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ResourceQuota](#)): OK

201 ([ResourceQuota](#)): Created

202 ([ResourceQuota](#)): Accepted

401: Unauthorized

update replace the specified ResourceQuota

HTTP Request

PUT /api/v1/namespaces/{namespace}/resourcequotas/{name}

Parameters

- **name** (*in path*): string, required

name of the ResourceQuota

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [ResourceQuota](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ResourceQuota](#)): OK

201 ([ResourceQuota](#)): Created

401: Unauthorized

update replace status of the specified ResourceQuota

HTTP Request

PUT /api/v1/namespaces/{namespace}/resourcequotas/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ResourceQuota
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [ResourceQuota](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ResourceQuota](#)): OK

201 ([ResourceQuota](#)): Created

401: Unauthorized

patch partially update the specified ResourceQuota

HTTP Request

PATCH /api/v1/namespaces/{namespace}/resourcequotas/{name}

Parameters

- **name** (*in path*): string, required
name of the ResourceQuota
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required

- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ResourceQuota](#)): OK

201 ([ResourceQuota](#)): Created

401: Unauthorized

patch partially update status of the specified ResourceQuota

HTTP Request

PATCH /api/v1/namespaces/{namespace}/resourcequotas/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the ResourceQuota
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ResourceQuota](#)): OK

201 ([ResourceQuota](#)): Created

401: Unauthorized

delete delete a ResourceQuota

HTTP Request

DELETE /api/v1/namespaces/{namespace}/resourcequotas/{name}

Parameters

- **name** (*in path*): string, required
 - name of the [ResourceQuota](#)
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([ResourceQuota](#)): OK

202 ([ResourceQuota](#)): Accepted

401: Unauthorized

deletecollection delete collection of ResourceQuota

HTTP Request

DELETE /api/v1/namespaces/{namespace}/resourcequotas

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

NetworkPolicy

NetworkPolicy describes what network traffic is allowed for a set of Pods.

```
apiVersion: networking.k8s.io/v1
```

```
import "k8s.io/api/networking/v1"
```

NetworkPolicy

NetworkPolicy describes what network traffic is allowed for a set of Pods

- **apiVersion**: networking.k8s.io/v1
- **kind**: NetworkPolicy
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([NetworkPolicySpec](#))

spec represents the specification of the desired behavior for this NetworkPolicy.

NetworkPolicySpec

NetworkPolicySpec provides the specification of a NetworkPolicy

- **podSelector** ([LabelSelector](#)), required

podSelector selects the pods to which this NetworkPolicy object applies. The array of ingress rules is applied to any pods selected by this field. Multiple network policies can select the same set of pods. In this case, the ingress rules for each are combined additively. This field is NOT optional and follows standard label selector semantics. An empty podSelector matches all pods in this namespace.

- **policyTypes** ([]string)

policyTypes is a list of rule types that the NetworkPolicy relates to. Valid options are ["Ingress"], ["Egress"], or ["Ingress", "Egress"]. If this field is not specified, it will default based on the existence of ingress or egress rules; policies that contain an egress section are assumed to affect egress, and all policies (whether or not they contain an ingress section) are assumed to affect ingress. If you want to write an egress-only policy, you must explicitly specify policyTypes ["Egress"]. Likewise, if you want to write a policy that specifies that no egress is allowed, you must specify a policyTypes value that include "Egress" (since such a policy would not include an egress section and would otherwise default to just ["Ingress"]). This field is beta-level in 1.8

- **ingress** ([]NetworkPolicyIngressRule)

ingress is a list of ingress rules to be applied to the selected pods. Traffic is allowed to a pod if there are no NetworkPolicies selecting the pod (and cluster policy otherwise allows the traffic), OR if the traffic source is the pod's local node, OR if the traffic matches at least one ingress rule across all of the NetworkPolicy objects whose podSelector matches the pod. If this field is empty then this NetworkPolicy does not allow any traffic (and serves solely to ensure that the pods it selects are isolated by default)

NetworkPolicyIngressRule describes a particular set of traffic that is allowed to the pods matched by a NetworkPolicySpec's podSelector. The traffic must match both ports and from.

- **ingress.from** ([]NetworkPolicyPeer)

from is a list of sources which should be able to access the pods selected for this rule. Items in this list are combined using a logical OR operation. If this field is empty or missing, this rule matches all sources (traffic not restricted by source). If this field is present and contains at least one item, this rule allows traffic only if the traffic matches at least one item in the from list.

NetworkPolicyPeer describes a peer to allow traffic to/from. Only certain combinations of fields are allowed

- **ingress.from.ipBlock** (IPBlock)

ipBlock defines policy on a particular IPBlock. If this field is set then neither of the other fields can be.

IPBlock describes a particular CIDR (Ex. "192.168.1.0/24","2001:db8::/64") that is allowed to the pods matched by a NetworkPolicySpec's podSelector. The except entry describes CIDRs that should not be included within this rule.

- **ingress.from.ipBlock.cidr** (string), required

cidr is a string representing the IPBlock Valid examples are "192.168.1.0/24" or "2001:db8::/64"

- **ingress.from.ipBlock.except ([]string)**

except is a slice of CIDRs that should not be included within an IPBlock Valid examples are "192.168.1.0/24" or "2001:db8::/64" Except values will be rejected if they are outside the cidr range

- **ingress.from.namespaceSelector ([LabelSelector](#))**

namespaceSelector selects namespaces using cluster-scoped labels. This field follows standard label selector semantics; if present but empty, it selects all namespaces.

If podSelector is also set, then the NetworkPolicyPeer as a whole selects the pods matching podSelector in the namespaces selected by namespaceSelector. Otherwise it selects all pods in the namespaces selected by namespaceSelector.

- **ingress.from.podSelector ([LabelSelector](#))**

podSelector is a label selector which selects pods. This field follows standard label selector semantics; if present but empty, it selects all pods.

If namespaceSelector is also set, then the NetworkPolicyPeer as a whole selects the pods matching podSelector in the Namespaces selected by NamespaceSelector. Otherwise it selects the pods matching podSelector in the policy's own namespace.

- **ingress.ports ([]NetworkPolicyPort)**

ports is a list of ports which should be made accessible on the pods selected for this rule. Each item in this list is combined using a logical OR. If this field is empty or missing, this rule matches all ports (traffic not restricted by port). If this field is present and contains at least one item, then this rule allows traffic only if the traffic matches at least one port in the list.

NetworkPolicyPort describes a port to allow traffic on

- **ingress.ports.port (IntOrString)**

port represents the port on the given protocol. This can either be a numerical or named port on a pod. If this field is not provided, this matches all port names and numbers. If present, only traffic on the specified protocol AND port will be matched.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **ingress.ports.endPort (int32)**

endPort indicates that the range of ports from port to endPort if set, inclusive, should be allowed by the policy. This field cannot be defined if the port field is not defined or if the port field is defined as a named (string) port. The endPort must be equal or greater than port.

- **ingress.ports.protocol** (string)

protocol represents the protocol (TCP, UDP, or SCTP) which traffic must match. If not specified, this field defaults to TCP.

- **egress** ([]NetworkPolicyEgressRule)

egress is a list of egress rules to be applied to the selected pods. Outgoing traffic is allowed if there are no NetworkPolicies selecting the pod (and cluster policy otherwise allows the traffic), OR if the traffic matches at least one egress rule across all of the NetworkPolicy objects whose podSelector matches the pod. If this field is empty then this NetworkPolicy limits all outgoing traffic (and serves solely to ensure that the pods it selects are isolated by default). This field is beta-level in 1.8

NetworkPolicyEgressRule describes a particular set of traffic that is allowed out of pods matched by a NetworkPolicySpec's podSelector. The traffic must match both ports and to. This type is beta-level in 1.8

- **egress.to** ([]NetworkPolicyPeer)

to is a list of destinations for outgoing traffic of pods selected for this rule. Items in this list are combined using a logical OR operation. If this field is empty or missing, this rule matches all destinations (traffic not restricted by destination). If this field is present and contains at least one item, this rule allows traffic only if the traffic matches at least one item in the to list.

NetworkPolicyPeer describes a peer to allow traffic to/from. Only certain combinations of fields are allowed

- **egress.to.ipBlock** (IPBlock)

ipBlock defines policy on a particular IPBlock. If this field is set then neither of the other fields can be.

IPBlock describes a particular CIDR (Ex. "192.168.1.0/24","2001:db8::/64") that is allowed to the pods matched by a NetworkPolicySpec's podSelector. The except entry describes CIDRs that should not be included within this rule.

- **egress.to.ipBlock.cidr** (string), required

cidr is a string representing the IPBlock Valid examples are "192.168.1.0/24" or "2001:db8::/64"

- **egress.to.ipBlock.exception** ([]string)

except is a slice of CIDRs that should not be included within an IPBlock Valid examples are "192.168.1.0/24" or "2001:db8::/64" Except values will be rejected if they are outside the cidr range

- **egress.to.namespaceSelector** ([LabelSelector](#))

namespaceSelector selects namespaces using cluster-scoped labels. This field follows standard label selector semantics; if present but empty, it selects all namespaces.

If podSelector is also set, then the NetworkPolicyPeer as a whole selects the pods matching podSelector in the namespaces selected by namespaceSelector. Otherwise it selects all pods in the namespaces selected by namespaceSelector.

- **egress.to.podSelector** ([LabelSelector](#))

podSelector is a label selector which selects pods. This field follows standard label selector semantics; if present but empty, it selects all pods.

If namespaceSelector is also set, then the NetworkPolicyPeer as a whole selects the pods matching podSelector in the Namespaces selected by NamespaceSelector. Otherwise it selects the pods matching podSelector in the policy's own namespace.

- **egress.ports** ([]NetworkPolicyPort)

ports is a list of destination ports for outgoing traffic. Each item in this list is combined using a logical OR. If this field is empty or missing, this rule matches all ports (traffic not restricted by port). If this field is present and contains at least one item, then this rule allows traffic only if the traffic matches at least one port in the list.

NetworkPolicyPort describes a port to allow traffic on

- **egress.ports.port** (IntOrString)

port represents the port on the given protocol. This can either be a numerical or named port on a pod. If this field is not provided, this matches all port names and numbers. If present, only traffic on the specified protocol AND port will be matched.

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **egress.ports.endPort** (int32)

endPort indicates that the range of ports from port to endPort if set, inclusive, should be allowed by the policy. This field cannot be defined if the port field is not defined or if the port field is defined as a named (string) port. The endPort must be equal or greater than port.

- **egress.ports.protocol** (string)

protocol represents the protocol (TCP, UDP, or SCTP) which traffic must match. If not specified, this field defaults to TCP.

NetworkPolicyList

NetworkPolicyList is a list of NetworkPolicy objects.

- **apiVersion**: networking.k8s.io/v1
- **kind**: NetworkPolicyList
- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][NetworkPolicy](#)), required

items is a list of schema objects.

Operations

get read the specified NetworkPolicy

HTTP Request

GET /apis/networking.k8s.io/v1/namespaces/{namespace}/networkpolicies/{name}

Parameters

- **name** (*in path*): string, required
 - name of the NetworkPolicy
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([NetworkPolicy](#)): OK

401: Unauthorized

list list or watch objects of kind NetworkPolicy

HTTP Request

GET /apis/networking.k8s.io/v1/namespaces/{namespace}/networkpolicies

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([NetworkPolicyList](#)): OK

401: Unauthorized

list list or watch objects of kind NetworkPolicy

HTTP Request

GET /apis/networking.k8s.io/v1/networkpolicies

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

watch (*in query*): boolean

- [watch](#)

Response

200 ([NetworkPolicyList](#)): OK

401: Unauthorized

create create a NetworkPolicy

HTTP Request

POST /apis/networking.k8s.io/v1/namespaces/{namespace}/networkpolicies

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [NetworkPolicy](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([NetworkPolicy](#)): OK

201 ([NetworkPolicy](#)): Created

202 ([NetworkPolicy](#)): Accepted

401: Unauthorized

update replace the specified NetworkPolicy

HTTP Request

PUT /apis/networking.k8s.io/v1/namespaces/{namespace}/networkpolicies/{name}

Parameters

- **name** (*in path*): string, required
name of the NetworkPolicy
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [NetworkPolicy](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([NetworkPolicy](#)): OK

201 ([NetworkPolicy](#)): Created

401: Unauthorized

patch partially update the specified NetworkPolicy

HTTP Request

PATCH /apis/networking.k8s.io/v1/namespaces/{namespace}/networkpolicies/{name}

Parameters

- **name** (*in path*): string, required
name of the NetworkPolicy

- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([NetworkPolicy](#)): OK

201 ([NetworkPolicy](#)): Created

401: Unauthorized

delete delete a NetworkPolicy

HTTP Request

DELETE /apis/networking.k8s.io/v1/namespaces/{namespace}/networkpolicies/{name}

Parameters

- **name** (*in path*): string, required
 - name of the NetworkPolicy
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of NetworkPolicy

HTTP Request

DELETE /apis/networking.k8s.io/v1/namespaces/{namespace}/networkpolicies

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

PodDisruptionBudget

PodDisruptionBudget is an object to define the max disruption that can be caused to a collection of pods.

```
apiVersion: policy/v1
```

```
import "k8s.io/api/policy/v1"
```

PodDisruptionBudget

PodDisruptionBudget is an object to define the max disruption that can be caused to a collection of pods

-
- **apiVersion**: policy/v1

kind: PodDisruptionBudget

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([PodDisruptionBudgetSpec](#))

Specification of the desired behavior of the PodDisruptionBudget.

- **status** ([PodDisruptionBudgetStatus](#))

Most recently observed status of the PodDisruptionBudget.

PodDisruptionBudgetSpec

PodDisruptionBudgetSpec is a description of a PodDisruptionBudget.

- **maxUnavailable** (IntOrString)

An eviction is allowed if at most "maxUnavailable" pods selected by "selector" are unavailable after the eviction, i.e. even in absence of the evicted pod. For example, one can prevent all voluntary evictions by specifying 0. This is a mutually exclusive setting with "minAvailable".

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **minAvailable** (IntOrString)

An eviction is allowed if at least "minAvailable" pods selected by "selector" will still be available after the eviction, i.e. even in the absence of the evicted pod. So for example you can prevent all voluntary evictions by specifying "100%".

IntOrString is a type that can hold an int32 or a string. When used in JSON or YAML marshalling and unmarshalling, it produces or consumes the inner type. This allows you to have, for example, a JSON field that can accept a name or number.

- **selector** ([LabelSelector](#))

Label query over pods whose evictions are managed by the disruption budget. A null selector will match no pods, while an empty ({}) selector will select all pods within the namespace.

- **unhealthyPodEvictionPolicy** (string)

UnhealthyPodEvictionPolicy defines the criteria for when unhealthy pods should be considered for eviction. Current implementation considers healthy pods, as pods that have status.conditions item with type="Ready",status="True".

Valid policies are IfHealthyBudget and AlwaysAllow. If no policy is specified, the default behavior will be used, which corresponds to the IfHealthyBudget policy.

IfHealthyBudget policy means that running pods (`status.phase="Running"`), but not yet healthy can be evicted only if the guarded application is not disrupted (`status.currentHealthy` is at least equal to `status.desiredHealthy`). Healthy pods will be subject to the PDB for eviction.

AlwaysAllow policy means that all running pods (`status.phase="Running"`), but not yet healthy are considered disrupted and can be evicted regardless of whether the criteria in a PDB is met. This means perspective running pods of a disrupted application might not get a chance to become healthy. Healthy pods will be subject to the PDB for eviction.

Additional policies may be added in the future. Clients making eviction decisions should disallow eviction of unhealthy pods if they encounter an unrecognized policy in this field.

This field is beta-level. The eviction API uses this field when the feature gate `PDBUnhealthyPodEvictionPolicy` is enabled (enabled by default).

PodDisruptionBudgetStatus

`PodDisruptionBudgetStatus` represents information about the status of a `PodDisruptionBudget`. Status may trail the actual state of a system.

- **currentHealthy** (int32), required

current number of healthy pods

- **desiredHealthy** (int32), required

minimum desired number of healthy pods

- **disruptionsAllowed** (int32), required

Number of pod disruptions that are currently allowed.

- **expectedPods** (int32), required

total number of pods counted by this disruption budget

- **conditions** ([]Condition)

Patch strategy: merge on key type

Map: unique values on key type will be kept during a merge

Conditions contain conditions for PDB. The disruption controller sets the `DisruptionAllowed` condition. The following are known values for the `reason` field (additional reasons could be added in the future): - `SyncFailed`: The controller

encountered an error and wasn't able to compute the number of allowed disruptions. Therefore no disruptions are allowed and the status of the condition will be False.

- InsufficientPods: The number of pods are either at or below the number required by the PodDisruptionBudget. No disruptions are allowed and the status of the condition will be False.
- SufficientPods: There are more pods than required by the PodDisruptionBudget. The condition will be True, and the number of allowed disruptions are provided by the disruptionsAllowed property.

Condition contains details for one aspect of the current state of this API Resource.

- **conditions.lastTransitionTime** (Time), required

lastTransitionTime is the last time the condition transitioned from one status to another. This should be when the underlying condition changed. If that is not known, then using the time when the API field changed is acceptable.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string), required

message is a human readable message indicating details about the transition. This may be an empty string.

- **conditions.reason** (string), required

reason contains a programmatic identifier indicating the reason for the condition's last transition. Producers of specific condition types may define expected values and meanings for this field, and whether the values are considered a guaranteed API. The value should be a CamelCase string. This field may not be empty.

- **conditions.status** (string), required

status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

type of condition in CamelCase or in foo.example.com/CamelCase.

- **conditions.observedGeneration** (int64)

observedGeneration represents the .metadata.generation that the condition was set based upon. For instance, if .metadata.generation is currently 12, but the .status.conditions[x].observedGeneration is 9, the condition is out of date with respect to the current state of the instance.

- **disruptedPods** (map[string]Time)

DisruptedPods contains information about pods whose eviction was processed by the API server eviction subresource handler but has not yet been observed by the PodDisruptionBudget controller. A pod will be in this map from the time when the API server processed the eviction request to the time when the pod is seen by PDB controller

as having been marked for deletion (or after a timeout). The key in the map is the name of the pod and the value is the time when the API server processed the eviction request. If the deletion didn't occur and a pod is still there it will be removed from the list automatically by PodDisruptionBudget controller after some time. If everything goes smooth this map should be empty for the most of the time. Large number of entries in the map may indicate problems with pod deletions.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **observedGeneration** (int64)

Most recent generation observed when updating this PDB status. DisruptionsAllowed and other status information is valid only if observedGeneration equals to PDB's object generation.

PodDisruptionBudgetList

PodDisruptionBudgetList is a collection of PodDisruptionBudgets.

- **apiVersion**: policy/v1
- **kind**: PodDisruptionBudgetList
- **metadata** ([ListMeta](#))
Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>
- **items** ([][PodDisruptionBudget](#)), required
Items is a list of PodDisruptionBudgets

Operations

get read the specified PodDisruptionBudget

HTTP Request

GET /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets/{name}

Parameters

- **name** (*in path*): string, required
name of the PodDisruptionBudget
- **namespace** (*in path*): string, required
[namespace](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([PodDisruptionBudget](#)): OK

401: Unauthorized

get read status of the specified PodDisruptionBudget

HTTP Request

GET /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets/{name}/status

Parameters

- **name** (*in path*): string, required

- name of the PodDisruptionBudget

- **namespace** (*in path*): string, required

- [namespace](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([PodDisruptionBudget](#)): OK

401: Unauthorized

list list or watch objects of kind PodDisruptionBudget

HTTP Request

GET /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets

Parameters

- **namespace** (*in path*): string, required

- [namespace](#)

- **allowWatchBookmarks** (*in query*): boolean

- [allowWatchBookmarks](#)

- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([PodDisruptionBudgetList](#)): OK

401: Unauthorized

list list or watch objects of kind PodDisruptionBudget

HTTP Request

GET /apis/policy/v1/poddisruptionbudgets

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([PodDisruptionBudgetList](#)): OK

401: Unauthorized

create create a PodDisruptionBudget

HTTP Request

POST /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [PodDisruptionBudget](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PodDisruptionBudget](#)): OK

201 ([PodDisruptionBudget](#)): Created

202 ([PodDisruptionBudget](#)): Accepted

401: Unauthorized

update replace the specified PodDisruptionBudget

HTTP Request

PUT /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets/{name}

Parameters

- **name** (*in path*): string, required

name of the PodDisruptionBudget

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [PodDisruptionBudget](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PodDisruptionBudget](#)): OK

201 ([PodDisruptionBudget](#)): Created

401: Unauthorized

update replace status of the specified PodDisruptionBudget

HTTP Request

PUT /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets/{name}/status

Parameters

- **name** (*in path*): string, required

name of the PodDisruptionBudget

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [PodDisruptionBudget](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PodDisruptionBudget](#)): OK

201 ([PodDisruptionBudget](#)): Created

401: Unauthorized

patch partially update the specified PodDisruptionBudget

HTTP Request

PATCH /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PodDisruptionBudget
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PodDisruptionBudget](#)): OK

201 ([PodDisruptionBudget](#)): Created

401: Unauthorized

patch partially update status of the specified PodDisruptionBudget

HTTP Request

PATCH /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets/{name}/status

Parameters

- **name** (*in path*): string, required
[name](#)
name of the PodDisruptionBudget
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PodDisruptionBudget](#)): OK

201 ([PodDisruptionBudget](#)): Created

401: Unauthorized

delete delete a PodDisruptionBudget

HTTP Request

DELETE /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PodDisruptionBudget
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of PodDisruptionBudget

HTTP Request

DELETE /apis/policy/v1/namespaces/{namespace}/poddisruptionbudgets

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

IPAddress v1alpha1

IPAddress represents a single IP of a single IP Family.

```
apiVersion: networking.k8s.io/v1alpha1
```

```
import "k8s.io/api/networking/v1alpha1"
```

IPAddress

IPAddress represents a single IP of a single IP Family. The object is designed to be used by APIs that operate on IP addresses. The object is used by the Service core API for allocation of IP addresses. An IP address can be represented in different formats, to guarantee the uniqueness of the IP, the name of the object is the IP address in canonical format, four decimal digits separated by dots suppressing leading zeros for IPv4 and the representation defined by RFC 5952 for IPv6. Valid: 192.168.1.5 or 2001:db8::1 or 2001:db8:aaaa:bbbb:cccc:dddd:eeee:1 Invalid: 10.01.2.3 or 2001:db8:0:0:0::1

- **apiVersion:** networking.k8s.io/v1alpha1
- **kind:** IPAddress
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([IPAddressSpec](#))

spec is the desired state of the IPAddress. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

IPAddressSpec

IPAddressSpec describe the attributes in an IP Address.

- **parentRef** ([ParentReference](#))

ParentRef references the resource that an IPAddress is attached to. An IPAddress must reference a parent object.

ParentReference describes a reference to a parent object.

- **parentRef.group** (string)

Group is the group of the object being referenced.

- **parentRef.name** (string)

Name is the name of the object being referenced.

parentRef.namespace (string)

◦ Namespace is the namespace of the object being referenced.

◦ **parentRef.resource** (string)

Resource is the resource of the object being referenced.

◦ **parentRef.uid** (string)

UID is the uid of the object being referenced.

IPAddressList

IPAddressList contains a list of IPAddress.

- **apiVersion**: networking.k8s.io/v1alpha1

- **kind**: IPAddressList

- **metadata** ([ListMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][IPAddress](#)), required

items is the list of IPAddresses.

Operations

get read the specified IPAddress

HTTP Request

GET /apis/networking.k8s.io/v1alpha1/ipaddresses/{name}

Parameters

- **name** (*in path*): string, required

name of the IPAddress

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([IPAddress](#)): OK

401: Unauthorized

list list or watch objects of kind IPAddress

HTTP Request

GET /apis/networking.k8s.io/v1alpha1/ipaddresses

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([IPAddressList](#)): OK

401: Unauthorized

create create an IPAddress

HTTP Request

POST /apis/networking.k8s.io/v1alpha1/ipaddresses

Parameters

- **body**: [IPAddress](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([IPAddress](#)): OK

201 ([IPAddress](#)): Created

202 ([IPAddress](#)): Accepted

401: Unauthorized

update replace the specified IPAddress

HTTP Request

PUT /apis/networking.k8s.io/v1alpha1/ipaddresses/{name}

Parameters

- **name** (*in path*): string, required

name of the IPAddress

- **body**: [IPAddress](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([IPAddress](#)): OK

201 ([IPAddress](#)): Created

401: Unauthorized

patch partially update the specified IPAddress

HTTP Request

PATCH /apis/networking.k8s.io/v1alpha1/ipaddresses/{name}

Parameters

- **name** (*in path*): string, required

name of the IPAddress

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

force (*in query*): boolean

- [force](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([IPAddress](#)): OK

201 ([IPAddress](#)): Created

401: Unauthorized

delete delete an IPAddress

HTTP Request

DELETE /apis/networking.k8s.io/v1alpha1/ipaddresses/{name}

Parameters

• **name** (*in path*): string, required

name of the IPAddress

• **body**: [DeleteOptions](#)

• **dryRun** (*in query*): string

[dryRun](#)

• **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

• **pretty** (*in query*): string

[pretty](#)

• **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of IPAddress

HTTP Request

DELETE /apis/networking.k8s.io/v1alpha1/ipaddresses

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Extend Resources

[CustomResourceDefinition](#)

CustomResourceDefinition represents a resource that should be exposed on the API server.

[MutatingWebhookConfiguration](#)

MutatingWebhookConfiguration describes the configuration of an admission webhook that accept or reject and may change the object.

[ValidatingWebhookConfiguration](#)

ValidatingWebhookConfiguration describes the configuration of an admission webhook that accept or reject and object without changing it.

[ValidatingAdmissionPolicy v1beta1](#)

ValidatingAdmissionPolicy describes the definition of an admission validation policy that accepts or rejects an object without changing it.

CustomResourceDefinition

CustomResourceDefinition represents a resource that should be exposed on the API server.

apiVersion: apiextensions.k8s.io/v1

```
import "k8s.io/apiextensions-apiserver/pkg/apis/apiextensions/v1"
```

CustomResourceDefinition

CustomResourceDefinition represents a resource that should be exposed on the API server. Its name MUST be in the format <.spec.name>.<.spec.group>.

- **apiVersion:** apiextensions.k8s.io/v1
- **kind:** CustomResourceDefinition
- **metadata** ([ObjectMeta](#))

Standard object's metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([CustomResourceDefinitionSpec](#)), required

spec describes how the user wants the resources to appear

- **status** ([CustomResourceDefinitionStatus](#))

status indicates the actual state of the CustomResourceDefinition

CustomResourceDefinitionSpec

CustomResourceDefinitionSpec describes how a user wants their resource to appear

- **group** (string), required

group is the API group of the defined custom resource. The custom resources are served under /apis/\<group>/.... Must match the name of the CustomResourceDefinition (in the form \<names.plural>.\<group>).

- **names** ([CustomResourceDefinitionNames](#)), required

names specify the resource and kind names for the custom resource.

CustomResourceDefinitionNames indicates the names to serve this CustomResourceDefinition

- **names.kind** (string), required

kind is the serialized kind of the resource. It is normally CamelCase and singular. Custom resource instances will use this value as the kind attribute in API calls.

- **names.plural** (string), required

plural is the plural name of the resource to serve. The custom resources are served under /apis/\<group>/\<version>/.../\<plural>. Must match the name of the CustomResourceDefinition (in the form \<names.plural>.\<group>). Must be all lowercase.

- **names.categories** ([]string)

categories is a list of grouped resources this custom resource belongs to (e.g. 'all'). This is published in API discovery documents, and used by clients to support invocations like kubectl get all.

- **names.listKind** (string)

listKind is the serialized kind of the list for this resource. Defaults to "kindList".

- **names.shortNames** ([]string)

shortNames are short names for the resource, exposed in API discovery documents, and used by clients to support invocations like `kubectl get \<shortname>`. It must be all lowercase.

- **names.singular** (string)

singular is the singular name of the resource. It must be all lowercase. Defaults to lowercased kind.

- **scope** (string), required

scope indicates whether the defined custom resource is cluster- or namespace-scoped. Allowed values are Cluster and Namespaced.

- **versions** ([]CustomResourceDefinitionVersion), required

versions is the list of all API versions of the defined custom resource. Version names are used to compute the order in which served versions are listed in API discovery. If the version string is "kube-like", it will sort above non "kube-like" version strings, which are ordered lexicographically. "Kube-like" versions start with a "v", then are followed by a number (the major version), then optionally the string "alpha" or "beta" and another number (the minor version). These are sorted first by GA > beta > alpha (where GA is a version with no suffix such as beta or alpha), and then by comparing major version, then minor version. An example sorted list of versions: v10, v2, v1, v11beta2, v10beta3, v3beta1, v12alpha1, v11alpha2, foo1, foo10.

CustomResourceDefinitionVersion describes a version for CRD.

- **versions.name** (string), required

name is the version name, e.g. "v1", "v2beta1", etc. The custom resources are served under this version at `/apis/\<group>/\<version>/...` if served is true.

- **versions.served** (boolean), required

served is a flag enabling/disabling this version from being served via REST APIs

- **versions.storage** (boolean), required

storage indicates this version should be used when persisting custom resources to storage. There must be exactly one version with storage=true.

- **versions.additionalPrinterColumns** ([]CustomResourceColumnDefinition)

additionalPrinterColumns specifies additional columns returned in Table output. See <https://kubernetes.io/docs/reference/using-api/api-concepts/#receiving-resources-as-tables> for details. If no columns are specified, a single column displaying the age of the custom resource is used.

CustomResourceColumnDefinition specifies a column for server side printing.

- **versions.additionalPrinterColumns.jsonPath** (string), required

jsonPath is a simple JSON path (i.e. with array notation) which is evaluated against each custom resource to produce the value for this column.

versions.additionalPrinterColumns.name (string), required

- name is a human readable name for the column.

▪ versions.additionalPrinterColumns.type (string), required

type is an OpenAPI type definition for this column. See <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#data-types> for details.

▪ versions.additionalPrinterColumns.description (string)

description is a human readable description of this column.

▪ versions.additionalPrinterColumns.format (string)

format is an optional OpenAPI type definition for this column. The 'name' format is applied to the primary identifier column to assist in clients identifying column is the resource name. See <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#data-types> for details.

▪ versions.additionalPrinterColumns.priority (int32)

priority is an integer defining the relative importance of this column compared to others. Lower numbers are considered higher priority. Columns that may be omitted in limited space scenarios should be given a priority greater than 0.

◦ **versions.deprecated** (boolean)

deprecated indicates this version of the custom resource API is deprecated. When set to true, API requests to this version receive a warning header in the server response. Defaults to false.

◦ **versions.deprecationWarning** (string)

deprecationWarning overrides the default warning returned to API clients. May only be set when deprecated is true. The default warning indicates this version is deprecated and recommends use of the newest served version of equal or greater stability, if one exists.

◦ **versions.schema** (CustomResourceValidation)

schema describes the schema used for validation, pruning, and defaulting of this version of the custom resource.

CustomResourceValidation is a list of validation methods for CustomResources.

▪ versions.schema.openAPIV3Schema ([JSONSchemaProps](#))

openAPIV3Schema is the OpenAPI v3 schema to use for validation and pruning.

◦ **versions.subresources** (CustomResourceSubresources)

subresources specify what subresources this version of the defined custom resource have.

CustomResourceSubresources defines the status and scale subresources for CustomResources.

- **versions.subresources.scale** (`CustomResourceSubresourceScale`)

scale indicates the custom resource should serve a /scale subresource that returns an autoscaling/v1 Scale object.

CustomResourceSubresourceScale defines how to serve the scale subresource for CustomResources.

- **versions.subresources.scale.specReplicasPath** (string), required

specReplicasPath defines the JSON path inside of a custom resource that corresponds to Scale spec.replicas. Only JSON paths without the array notation are allowed. Must be a JSON Path under .spec. If there is no value under the given path in the custom resource, the /scale subresource will return an error on GET.

- **versions.subresources.scale.statusReplicasPath** (string), required

statusReplicasPath defines the JSON path inside of a custom resource that corresponds to Scale status.replicas. Only JSON paths without the array notation are allowed. Must be a JSON Path under .status. If there is no value under the given path in the custom resource, the status.replicas value in the /scale subresource will default to 0.

- **versions.subresources.scale.labelSelectorPath** (string)

labelSelectorPath defines the JSON path inside of a custom resource that corresponds to Scale status.selector. Only JSON paths without the array notation are allowed. Must be a JSON Path under .status or .spec. Must be set to work with HorizontalPodAutoscaler. The field pointed by this JSON path must be a string field (not a complex selector struct) which contains a serialized label selector in string form. More info: <https://kubernetes.io/docs/tasks/access-kubernetes-api/custom-resources/custom-resource-definitions#scale-subresource> If there is no value under the given path in the custom resource, the status.selector value in the /scale subresource will default to the empty string.

- **versions.subresources.status** (`CustomResourceSubresourceStatus`)

status indicates the custom resource should serve a /status subresource. When enabled: 1. requests to the custom resource primary endpoint ignore changes to the status stanza of the object. 2. requests to the custom resource /status subresource ignore changes to anything other than the status stanza of the object.

*CustomResourceSubresourceStatus defines how to serve the status subresource for CustomResources. Status is represented by the .status JSON path inside of a CustomResource. When set, * exposes a /status subresource for the custom*

*resource * PUT requests to the /status subresource take a custom resource object, and ignore changes to anything except the status stanza * PUT/POST/PATCH requests to the custom resource ignore changes to the status stanza*

- **conversion** (CustomResourceConversion)

conversion defines conversion settings for the CRD.

CustomResourceConversion describes how to convert different versions of a CR.

- **conversion.strategy** (string), required

strategy specifies how custom resources are converted between versions. Allowed values are: - "None": The converter only change the apiVersion and would not touch any other field in the custom resource. - "Webhook": API Server will call to an external webhook to do the conversion. Additional information is needed for this option. This requires spec.preserveUnknownFields to be false, and spec.conversion.webhook to be set.

- **conversion.webhook** (WebhookConversion)

webhook describes how to call the conversion webhook. Required when strategy is set to "Webhook".

WebhookConversion describes how to call a conversion webhook

- **conversion.webhook.conversionReviewVersions** ([]string), required

conversionReviewVersions is an ordered list of preferred ConversionReview versions the Webhook expects. The API server will use the first version in the list which it supports. If none of the versions specified in this list are supported by API server, conversion will fail for the custom resource. If a persisted Webhook configuration specifies allowed versions and does not include any versions known to the API Server, calls to the webhook will fail.

- **conversion.webhook.clientConfig** (WebhookClientConfig)

clientConfig is the instructions for how to call the webhook if strategy is Webhook.

WebhookClientConfig contains the information to make a TLS connection with the webhook.

- **conversion.webhook.clientConfig.caBundle** ([]byte)

caBundle is a PEM encoded CA bundle which will be used to validate the webhook's server certificate. If unspecified, system trust roots on the apiserver are used.

- **conversion.webhook.clientConfig.service** (ServiceReference)

service is a reference to the service for this webhook. Either service or url must be specified.

If the webhook is running within the cluster, then you should use service.

ServiceReference holds a reference to Service.legacy.k8s.io

- **conversion.webhook.clientConfig.service.name** (string), required

name is the name of the service. Required

- **conversion.webhook.clientConfig.service.namespace** (string), required

namespace is the namespace of the service. Required

- **conversion.webhook.clientConfig.service.path** (string)

path is an optional URL path at which the webhook will be contacted.

- **conversion.webhook.clientConfig.service.port** (int32)

port is an optional service port at which the webhook will be contacted. port should be a valid port number (1-65535, inclusive). Defaults to 443 for backward compatibility.

- **conversion.webhook.clientConfig.url** (string)

url gives the location of the webhook, in standard URL form (scheme://host:port/path). Exactly one of url or service must be specified.

The host should not refer to a service running in the cluster; use the service field instead. The host might be resolved via external DNS in some apiservers (e.g., kube-apiserver cannot resolve in-cluster DNS as that would be a layering violation). host may also be an IP address.

Please note that using localhost or 127.0.0.1 as a host is risky unless you take great care to run this webhook on all hosts which run an apiserver which might need to make calls to this webhook. Such installs are likely to be non-portable, i.e., not easy to turn up in a new cluster.

The scheme must be "https"; the URL must begin with "https://".

A path is optional, and if present may be any string permissible in a URL. You may use the path to pass an arbitrary string to the webhook, for example, a cluster identifier.

Attempting to use a user or basic auth e.g. "user:password@" is not allowed. Fragments ("#...") and query parameters ("?...") are not allowed, either.

- **preserveUnknownFields** (boolean)

`preserveUnknownFields` indicates that object fields which are not specified in the OpenAPI schema should be preserved when persisting to storage. `apiVersion`, `kind`, `metadata` and known fields inside `metadata` are always preserved. This field is deprecated in favor of setting `x-preserve-unknown-fields` to true in `spec.versions[*].schema.openAPIV3Schema`. See <https://kubernetes.io/docs/tasks/extend-kubernetes/custom-resources/custom-resource-definitions/#field-pruning> for details.

JSONSchemaProps

`JSONSchemaProps` is a JSON-Schema following Specification Draft 4 (<http://json-schema.org/>).

- **\$ref** (string)
- **\$schema** (string)
- **additionalItems** (JSONSchemaPropsOrBool)

`JSONSchemaPropsOrBool` represents `JSONSchemaProps` or a boolean value. Defaults to true for the boolean property.

- **additionalProperties** (JSONSchemaPropsOrBool)

`JSONSchemaPropsOrBool` represents `JSONSchemaProps` or a boolean value. Defaults to true for the boolean property.

- **allOf** ([][JSONSchemaProps](#))
- **anyOf** ([][JSONSchemaProps](#))
- **default** (JSON)

`default` is a default value for undefined object fields. Defaulting is a beta feature under the `CustomResourceDefaulting` feature gate. Defaulting requires `spec.preserveUnknownFields` to be false.

`JSON` represents any valid `JSON` value. These types are supported: `bool`, `int64`, `float64`, `string`, `[]interface{}`, `map[string]interface{}` and `nil`.

- **definitions** (map[string][JSONSchemaProps](#))
- **dependencies** (map[string]JSONSchemaPropsOrStringArray)

`JSONSchemaPropsOrStringArray` represents a `JSONSchemaProps` or a string array.

- **description** (string)
- **enum** ([]JSON)

`JSON` represents any valid `JSON` value. These types are supported: `bool`, `int64`, `float64`, `string`, `[]interface{}`, `map[string]interface{}` and `nil`.

- **example** (JSON)

JSON represents any valid JSON value. These types are supported: bool, int64, float64, string, []interface{}, map[string]interface{} and nil.

- **exclusiveMaximum** (boolean)
- **exclusiveMinimum** (boolean)
- **externalDocs** (ExternalDocumentation)

ExternalDocumentation allows referencing an external resource for extended documentation.

- **externalDocs.description** (string)
- **externalDocs.url** (string)

- **format** (string)

format is an OpenAPI v3 format string. Unknown formats are ignored. The following formats are validated:

- bsonobjectid: a bson object ID, i.e. a 24 characters hex string - uri: an URI as parsed by Golang net/url.ParseRequestURI - email: an email address as parsed by Golang net/mail.ParseAddress - hostname: a valid representation for an Internet host name, as defined by RFC 1034, section 3.1 [RFC1034]. - ipv4: an IPv4 IP as parsed by Golang net.ParseIP - ipv6: an IPv6 IP as parsed by Golang net.ParseIP - cidr: a CIDR as parsed by Golang net.ParseCIDR - mac: a MAC address as parsed by Golang net.ParseMAC - uuid: an UUID that allows uppercase defined by the regex (?i)^[0-9a-f]{8}-?[0-9a-f]{4}-?[0-9a-f]{4}-?[0-9a-f]{4}-?[0-9a-f]{12}\$ - uuid3: an UUID3 that allows uppercase defined by the regex (?i)^[0-9a-f]{8}-?[0-9a-f]{4}-?3[0-9a-f]{3}-?[0-9a-f]{4}-?[0-9a-f]{12}\$ - uuid4: an UUID4 that allows uppercase defined by the regex (?i)^[0-9a-f]{8}-?[0-9a-f]{4}-?4[0-9a-f]{3}-?[89ab][0-9a-f]{3}-?[0-9a-f]{12}\$ - uuid5: an UUID5 that allows uppercase defined by the regex (?i)^[0-9a-f]{8}-?[0-9a-f]{4}-?5[0-9a-f]{3}-?[89ab][0-9a-f]{3}-?[0-9a-f]{12}\$ - isbn: an ISBN10 or ISBN13 number string like "0321751043" or "978-0321751041" - isbn10: an ISBN10 number string like "0321751043" - isbn13: an ISBN13 number string like "978-0321751041" - creditcard: a credit card number defined by the regex ^(?:4[0-9]{12}(?:[0-9]{3})?|5[1-5][0-9]{14}|6(?:011|5[0-9][0-9])[0-9]{12}|3[47][0-9]{13}|3(?:0[0-5]|68)[0-9]{11}|(?:2131|1800|35\d{3})\d{11})\$ with any non digit characters mixed in - ssn: a U.S. social security number following the regex ^\d{3}[-]?\d{2}[-]?\d{4}\$ - hexcolor: an hexadecimal color code like "#FFFFFF: following the regex ^#?(#[0-9a-fA-F]{3}|#[0-9a-fA-F]{6})\$ - rgbcolor: an RGB color code like rgb like "rgb(255,255,255)" - byte: base64 encoded binary data - password: any kind of string - date: a date string like "2006-01-02" as defined by full-date in RFC3339 - duration: a duration string like "22 ns" as parsed by Golang time.ParseDuration or compatible with Scala duration format - datetime: a date time string like "2014-12-15T19:30:20.000Z" as defined by date-time in RFC3339.

- **id** (string)
- **items** (JSONSchemaPropsOrArray)

JSONSchemaPropsOrArray represents a value that can either be a JSONSchemaProps or an array of JSONSchemaProps. Mainly here for serialization purposes.

- **maxItems** (int64)
- **maxLength** (int64)
- **maxProperties** (int64)
- **maximum** (double)
- **minItems** (int64)
- **minLength** (int64)
- **minProperties** (int64)
- **minimum** (double)
- **multipleOf** (double)
- **not** ([JSONSchemaProps](#))
- **nullable** (boolean)
- **oneOf** ([][JSONSchemaProps](#))
- **pattern** (string)
- **patternProperties** (map[string][JSONSchemaProps](#))
- **properties** (map[string][JSONSchemaProps](#))
- **required** ([]string)
- **title** (string)
- **type** (string)
- **uniqueItems** (boolean)
- **x-kubernetes-embedded-resource** (boolean)

x-kubernetes-embedded-resource defines that the value is an embedded Kubernetes runtime.Object, with TypeMeta and ObjectMeta. The type must be object. It is allowed to further restrict the embedded object. kind, apiVersion and metadata are validated automatically. **x-kubernetes-preserve-unknown-fields** is allowed to be true, but does not have to be if the object is fully specified (up to kind, apiVersion, metadata).

- **x-kubernetes-int-or-string** (boolean)

x-kubernetes-int-or-string specifies that this value is either an integer or a string. If this is true, an empty type is allowed and type as child of anyOf is permitted if following one of the following patterns:

1. anyOf:
 - type: integer
 - type: string

2. allOf:

- anyOf:
 - type: integer
 - type: string
- ... zero or more

• **x-kubernetes-list-map-keys** ([]string)

x-kubernetes-list-map-keys annotates an array with the x-kubernetes-list-type map by specifying the keys used as the index of the map.

This tag MUST only be used on lists that have the "x-kubernetes-list-type" extension set to "map". Also, the values specified for this attribute must be a scalar typed field of the child structure (no nesting is supported).

The properties specified must either be required or have a default value, to ensure those properties are present for all list items.

• **x-kubernetes-list-type** (string)

x-kubernetes-list-type annotates an array to further describe its topology. This extension must only be used on lists and may have 3 possible values:

1. atomic: the list is treated as a single entity, like a scalar. Atomic lists will be entirely replaced when updated. This extension may be used on any type of list (struct, scalar, ...).
2. set: Sets are lists that must not have multiple items with the same value. Each value must be a scalar, an object with x-kubernetes-map-type atomic or an array with x-kubernetes-list-type atomic.
3. map: These lists are like maps in that their elements have a non-index key used to identify them. Order is preserved upon merge. The map tag must only be used on a list with elements of type object. Defaults to atomic for arrays.

• **x-kubernetes-map-type** (string)

x-kubernetes-map-type annotates an object to further describe its topology. This extension must only be used when type is object and may have 2 possible values:

1. granular: These maps are actual maps (key-value pairs) and each fields are independent from each other (they can each be manipulated by separate actors). This is the default behaviour for all maps.
2. atomic: the list is treated as a single entity, like a scalar. Atomic maps will be entirely replaced when updated.

• **x-kubernetes-preserve-unknown-fields** (boolean)

x-kubernetes-preserve-unknown-fields stops the API server decoding step from pruning fields which are not specified in the validation schema. This affects fields recursively, but switches back to normal pruning behaviour if nested properties or additionalProperties are specified in the schema. This can either be true or undefined. False is forbidden.

• **x-kubernetes-validations** ([]ValidationRule)

Patch strategy: merge on key rule

Map: unique values on key rule will be kept during a merge

x-kubernetes-validations describes a list of validation rules written in the CEL expression language. This field is an alpha-level. Using this field requires the feature gate CustomResourceValidationExpressions to be enabled.

ValidationRule describes a validation rule written in the CEL expression language.

- **x-kubernetes-validations.rule** (string), required

Rule represents the expression which will be evaluated by CEL. ref: <https://github.com/google/cel-spec> The Rule is scoped to the location of the x-kubernetes-validations extension in the schema. The self variable in the CEL expression is bound to the scoped value. Example: - Rule scoped to the root of a resource with a status subresource: {"rule": "self.status.actual <= self.spec.maxDesired"}

If the Rule is scoped to an object with properties, the accessible properties of the object are field selectable via self.field and field presence can be checked via has(self.field). Null valued fields are treated as absent fields in CEL expressions. If the Rule is scoped to an object with additionalProperties (i.e. a map) the value of the map are accessible via self[mapKey], map containment can be checked via mapKey in self and all entries of the map are accessible via CEL macros and functions such as self.all(...). If the Rule is scoped to an array, the elements of the array are accessible via self[i] and also by macros and functions. If the Rule is scoped to a scalar, self is bound to the scalar value. Examples: - Rule scoped to a map of objects: {"rule": "self.components['Widget'].priority < 10"} - Rule scoped to a list of integers: {"rule": "self.values.all(value, value >= 0 && value < 100)"} - Rule scoped to a string value: {"rule": "self.startsWith('kube')"}

The apiVersion, kind, metadata.name and metadata.generateName are always accessible from the root of the object and from any x-kubernetes-embedded-resource annotated objects. No other metadata properties are accessible.

Unknown data preserved in custom resources via x-kubernetes-preserve-unknown-fields is not accessible in CEL expressions. This includes: - Unknown field values that are preserved by object schemas with x-kubernetes-preserve-unknown-fields. - Object properties where the property schema is of an "unknown type". An "unknown type" is recursively defined as:

- A schema with no type and x-kubernetes-preserve-unknown-fields set to true
- An array where the items schema is of an "unknown type"
- An object where the additionalProperties schema is of an "unknown type"

Only property names of the form [a-zA-Z_-.][a-zA-Z0-9_-.]* are accessible. Accessible property names are escaped according to the following rules when accessed in the expression: - **" escapes to 'underscores' - !' escapes to 'dot' - '-' escapes to 'dash' - '/' escapes to 'slash' - Property names that exactly match a CEL RESERVED keyword escape to '{keyword}_"**. The keywords are: "true", "false", "null", "in", "as", "break", "const", "continue", "else", "for", "function", "if", "import", "let", "loop", "package", "namespace", "return". Examples:

- Rule accessing a property named "namespace": {"rule": "self.namespace > 0"}
- Rule accessing a property named "x-prop": {"rule": "self.x_dash_prop > 0"}

- Rule accessing a property named "redact__d": {"rule": "self.redact__underscores__d > 0"}

Equality on arrays with x-kubernetes-list-type of 'set' or 'map' ignores element order, i.e. [1, 2] == [2, 1]. Concatenation on arrays with x-kubernetes-list-type use the semantics of the list type:

- 'set': X + Y performs a union where the array positions of all elements in X are preserved and non-intersecting elements in Y are appended, retaining their partial order.
- 'map': X + Y performs a merge where the array positions of all keys in X are preserved but the values are overwritten by values in Y when the key sets of X and Y intersect. Elements in Y with non-intersecting keys are appended, retaining their partial order.

- **x-kubernetes-validations.fieldPath** (string)

fieldPath represents the field path returned when the validation fails. It must be a relative JSON path (i.e. with array notation) scoped to the location of this x-kubernetes-validations extension in the schema and refer to an existing field. e.g. when validation checks if a specific attribute foo under a map testMap, the fieldPath could be set to .testMap.foo If the validation checks two lists must have unique attributes, the fieldPath could be set to either of the list: e.g. .testList It does not support list numeric index. It supports child operation to refer to an existing field currently. Refer to [JSONPath support in Kubernetes](#) for more info. Numeric index of array is not supported. For field name which contains special characters, use ['specialName'] to refer the field name. e.g. for attribute foo.34\$ appears in a list testList, the fieldPath could be set to .testList['foo.34\$']

- **x-kubernetes-validations.message** (string)

Message represents the message displayed when validation fails. The message is required if the Rule contains line breaks. The message must not contain line breaks. If unset, the message is "failed rule: {Rule}" . e.g. "must be a URL with the host matching spec.host"

- **x-kubernetes-validations.messageExpression** (string)

MessageExpression declares a CEL expression that evaluates to the validation failure message that is returned when this rule fails. Since messageExpression is used as a failure message, it must evaluate to a string. If both message and messageExpression are present on a rule, then messageExpression will be used if validation fails. If messageExpression results in a runtime error, the runtime error is logged, and the validation failure message is produced as if the messageExpression field were unset. If messageExpression evaluates to an empty string, a string with only spaces, or a string that contains line breaks, then the validation failure message will also be produced as if the messageExpression field were unset, and the fact that messageExpression produced an empty string/string with only spaces/string with line breaks will be logged. messageExpression has access to all the same variables as the rule; the only difference is the return type. Example: "x must be less than max (+string(self.max)+)"

- **x-kubernetes-validations.reason** (string)

reason provides a machine-readable validation failure reason that is returned to the caller when a request fails this validation rule. The HTTP status code returned to the caller will match the reason of the reason of the first failed validation rule. The currently supported reasons are: "FieldValueInvalid", "FieldValueForbidden", "FieldValueRequired", "FieldValueDuplicate". If not set, default to use "FieldValueInvalid". All future added reasons must be accepted by clients when reading this value and unknown reasons should be treated as FieldValueInvalid.

CustomResourceDefinitionStatus

CustomResourceDefinitionStatus indicates the state of the CustomResourceDefinition

- **acceptedNames** (CustomResourceDefinitionNames)

acceptedNames are the names that are actually being used to serve discovery. They may be different than the names in spec.

CustomResourceDefinitionNames indicates the names to serve this CustomResourceDefinition

- **acceptedNames.kind** (string), required

kind is the serialized kind of the resource. It is normally CamelCase and singular. Custom resource instances will use this value as the kind attribute in API calls.

- **acceptedNames.plural** (string), required

plural is the plural name of the resource to serve. The custom resources are served under /apis/\<group>/\<version>/.../\<plural>. Must match the name of the CustomResourceDefinition (in the form \<names.plural>.\<group>). Must be all lowercase.

- **acceptedNames.categories** ([]string)

categories is a list of grouped resources this custom resource belongs to (e.g. 'all'). This is published in API discovery documents, and used by clients to support invocations like kubectl get all.

- **acceptedNames.listKind** (string)

listKind is the serialized kind of the list for this resource. Defaults to "kindList".

- **acceptedNames.shortNames** ([]string)

shortNames are short names for the resource, exposed in API discovery documents, and used by clients to support invocations like kubectl get \<shortname>. It must be all lowercase.

- **acceptedNames.singular** (string)

singular is the singular name of the resource. It must be all lowercase. Defaults to lowercased kind.

conditions ([]CustomResourceDefinitionCondition)

- *Map: unique values on key type will be kept during a merge*

conditions indicate state for particular aspects of a CustomResourceDefinition

CustomResourceDefinitionCondition contains details for the current condition of this pod.

- **conditions.status** (string), required

status is the status of the condition. Can be True, False, Unknown.

- **conditions.type** (string), required

type is the type of the condition. Types include Established, NamesAccepted and Terminating.

- **conditions.lastTransitionTime** (Time)

lastTransitionTime last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

message is a human-readable message indicating details about last transition.

- **conditions.reason** (string)

reason is a unique, one-word, CamelCase reason for the condition's last transition.

- **storedVersions** ([]string)

storedVersions lists all versions of CustomResources that were ever persisted. Tracking these versions allows a migration path for stored versions in etcd. The field is mutable so a migration controller can finish a migration to another version (ensuring no old objects are left in storage), and then remove the rest of the versions from this list. Versions may not be removed from spec.versions while they exist in this list.

CustomResourceDefinitionList

CustomResourceDefinitionList is a list of CustomResourceDefinition objects.

- **items** ([]CustomResourceDefinition), required

items list individual CustomResourceDefinition objects

- **apiVersion** (string)

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject

unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>

- **kind** (string)

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **metadata** ([ListMeta](#))

Standard object's metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

Operations

get read the specified CustomResourceDefinition

HTTP Request

GET /apis/apiextensions.k8s.io/v1/customresourcedefinitions/{name}

Parameters

- **name** (*in path*): string, required
name of the CustomResourceDefinition
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CustomResourceDefinition](#)): OK

401: Unauthorized

get read status of the specified CustomResourceDefinition

HTTP Request

GET /apis/apiextensions.k8s.io/v1/customresourcedefinitions/{name}/status

Parameters

- **name** (*in path*): string, required
name of the CustomResourceDefinition

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([CustomResourceDefinition](#)): OK

401: Unauthorized

list list or watch objects of kind CustomResourceDefinition

HTTP Request

GET /apis/apiextensions.k8s.io/v1/customresourcedefinitions

Parameters

- **allowWatchBookmarks** (*in query*): boolean

- [allowWatchBookmarks](#)

- **continue** (*in query*): string

- [continue](#)

- **fieldSelector** (*in query*): string

- [fieldSelector](#)

- **labelSelector** (*in query*): string

- [labelSelector](#)

- **limit** (*in query*): integer

- [limit](#)

- **pretty** (*in query*): string

- [pretty](#)

- **resourceVersion** (*in query*): string

- [resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

- [resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

- [sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)
- **watch** (*in query*): boolean
 - [watch](#)

Response

200 ([CustomResourceDefinitionList](#)): OK

401: Unauthorized

create create a CustomResourceDefinition

HTTP Request

POST /apis/apiextensions.k8s.io/v1/customresourcedefinitions

Parameters

- **body**: [CustomResourceDefinition](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([CustomResourceDefinition](#)): OK

201 ([CustomResourceDefinition](#)): Created

202 ([CustomResourceDefinition](#)): Accepted

401: Unauthorized

update replace the specified CustomResourceDefinition

HTTP Request

PUT /apis/apiextensions.k8s.io/v1/customresourcedefinitions/{name}

Parameters

- **name** (*in path*): string, required
name of the CustomResourceDefinition
- **body**: [CustomResourceDefinition](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([CustomResourceDefinition](#)): OK

201 ([CustomResourceDefinition](#)): Created

401: Unauthorized

update replace status of the specified CustomResourceDefinition

HTTP Request

PUT /apis/apiextensions.k8s.io/v1/customresourcedefinitions/{name}/status

Parameters

- **name** (*in path*): string, required
name of the CustomResourceDefinition
- **body**: [CustomResourceDefinition](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CustomResourceDefinition](#)): OK

201 ([CustomResourceDefinition](#)): Created

401: Unauthorized

patch partially update the specified CustomResourceDefinition

HTTP Request

PATCH /apis/apiextensions.k8s.io/v1/customresourcedefinitions/{name}

Parameters

- **name** (*in path*): string, required

name of the CustomResourceDefinition

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CustomResourceDefinition](#)): OK

201 ([CustomResourceDefinition](#)): Created

401: Unauthorized

patch partially update status of the specified CustomResourceDefinition

HTTP Request

PATCH /apis/apiextensions.k8s.io/v1/customresourcedefinitions/{name}/status

Parameters

- **name** (*in path*): string, required

name of the CustomResourceDefinition

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([CustomResourceDefinition](#)): OK

201 ([CustomResourceDefinition](#)): Created

401: Unauthorized

delete delete a CustomResourceDefinition

HTTP Request

DELETE /apis/apiextensions.k8s.io/v1/customresourcedefinitions/{name}

Parameters

- **name** (*in path*): string, required
name of the CustomResourceDefinition
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of CustomResourceDefinition

HTTP Request

DELETE /apis/apiextensions.k8s.io/v1/customresourcedefinitions

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

MutatingWebhookConfiguration

MutatingWebhookConfiguration describes the configuration of and admission webhook that accept or reject and may change the object.

apiVersion: admissionregistration.k8s.io/v1

```
import "k8s.io/api/admissionregistration/v1"
```

MutatingWebhookConfiguration

MutatingWebhookConfiguration describes the configuration of an admission webhook that accept or reject and may change the object.

- **apiVersion:** admissionregistration.k8s.io/v1
- **kind:** MutatingWebhookConfiguration
- **metadata** ([ObjectMeta](#))

Standard object metadata; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>.

- **webhooks** ([]MutatingWebhook)

Patch strategy: merge on key name

Webhooks is a list of webhooks and the affected resources and operations.

MutatingWebhook describes an admission webhook and the resources and operations it applies to.

- **webhooks.admissionReviewVersions** ([]string), required

AdmissionReviewVersions is an ordered list of preferred AdmissionReview versions the Webhook expects. API server will try to use first version in the list which it supports. If none of the versions specified in this list supported by API server, validation will fail for this object. If a persisted webhook configuration specifies allowed versions and does not include any versions known to the API Server, calls to the webhook will fail and be subject to the failure policy.

- **webhooks.clientConfig** (WebhookClientConfig), required

ClientConfig defines how to communicate with the hook. Required

WebhookClientConfig contains the information to make a TLS connection with the webhook

- **webhooks.clientConfig.caBundle** ([]byte)

caBundle is a PEM encoded CA bundle which will be used to validate the webhook's server certificate. If unspecified, system trust roots on the apiserver are used.

- **webhooks.clientConfig.service** (ServiceReference)

service is a reference to the service for this webhook. Either service or url must be specified.

If the webhook is running within the cluster, then you should use service.

ServiceReference holds a reference to Service.legacy.k8s.io

- **webhooks.clientConfig.service.name** (string), required

name is the name of the service. Required

- **webhooks.clientConfig.service.namespace** (string), required

namespace is the namespace of the service. Required

- **webhooks.clientConfig.service.path** (string)

path is an optional URL path which will be sent in any request to this service.

- **webhooks.clientConfig.service.port** (int32)

If specified, the port on the service that hosting webhook. Default to 443 for backward compatibility. port should be a valid port number (1-65535, inclusive).

- **webhooks.clientConfig.url** (string)

url gives the location of the webhook, in standard URL form (scheme://host:port/path). Exactly one of url or service must be specified.

The host should not refer to a service running in the cluster; use the service field instead. The host might be resolved via external DNS in some apiservers (e.g., kube-apiserver cannot resolve in-cluster DNS as that would be a layering violation). host may also be an IP address.

Please note that using localhost or 127.0.0.1 as a host is risky unless you take great care to run this webhook on all hosts which run an apiserver which might need to make calls to this webhook. Such installs are likely to be non-portable, i.e., not easy to turn up in a new cluster.

The scheme must be "https"; the URL must begin with "https://".

A path is optional, and if present may be any string permissible in a URL. You may use the path to pass an arbitrary string to the webhook, for example, a cluster identifier.

Attempting to use a user or basic auth e.g. "user:password@" is not allowed. Fragments ("#...") and query parameters ("?...") are not allowed, either.

- **webhooks.name** (string), required

The name of the admission webhook. Name should be fully qualified, e.g., imagepolicy.kubernetes.io, where "imagepolicy" is the name of the webhook, and kubernetes.io is the name of the organization. Required.

- **webhooks.sideEffects** (string), required

SideEffects states whether this webhook has side effects. Acceptable values are: None, NoneOnDryRun (webhooks created via v1beta1 may also specify Some or

Unknown). Webhooks with side effects MUST implement a reconciliation system, since a request may be rejected by a future step in the admission chain and the side effects therefore need to be undone. Requests with the dryRun attribute will be auto-rejected if they match a webhook with sideEffects == Unknown or Some.

- **webhooks.failurePolicy** (string)

FailurePolicy defines how unrecognized errors from the admission endpoint are handled - allowed values are Ignore or Fail. Defaults to Fail.

- **webhooks.matchConditions** ([]MatchCondition)

Patch strategy: merge on key name

Map: unique values on key name will be kept during a merge

MatchConditions is a list of conditions that must be met for a request to be sent to this webhook. Match conditions filter requests that have already been matched by the rules, namespaceSelector, and objectSelector. An empty list of matchConditions matches all requests. There are a maximum of 64 match conditions allowed.

The exact matching logic is (in order):

1. If ANY matchCondition evaluates to FALSE, the webhook is skipped.
2. If ALL matchConditions evaluate to TRUE, the webhook is called.
3. If any matchCondition evaluates to an error (but none are FALSE):
 - If failurePolicy=Fail, reject the request
 - If failurePolicy=Ignore, the error is ignored and the webhook is skipped

This is a beta feature and managed by the AdmissionWebhookMatchConditions feature gate.

MatchCondition represents a condition which must be fulfilled for a request to be sent to a webhook.

- **webhooks.matchConditions.expression** (string), required

Expression represents the expression which will be evaluated by CEL. Must evaluate to bool. CEL expressions have access to the contents of the AdmissionRequest and Authorizer, organized into CEL variables:

'object' - The object from the incoming request. The value is null for DELETE requests.
'oldObject' - The existing object. The value is null for CREATE requests.
'request' - Attributes of the admission request(/pkg/apis/admission/types.go#AdmissionRequest).
'authorizer' - A CEL Authorizer. May be used to perform authorization checks for the principal (user or service account) of the request. See <https://pkg.go.dev/k8s.io/apiserver/pkg/cel/library#Authz>
'authorizer.requestResource' - A CEL ResourceCheck constructed from the 'authorizer' and configured with the request resource. Documentation on CEL: <https://kubernetes.io/docs/reference/using-api/cel/>

Required.

- **webhooks.matchConditions.name** (string), required

Name is an identifier for this match condition, used for strategic merging of MatchConditions, as well as providing an identifier for logging purposes. A good name should be descriptive of the associated expression. Name must be a qualified name consisting of alphanumeric characters, '-' , '' or '.', and must start and end with an alphanumeric character (e.g. 'MyName', or 'my.name', or '123-abc', regex used for validation is '([A-Za-z0-9][-A-Za-z0-9.]*)?[A-Za-z0-9]') with an optional DNS subdomain prefix and '/' (e.g. 'example.com/MyName')

Required.

- **webhooks.matchPolicy** (string)

matchPolicy defines how the "rules" list is used to match incoming requests. Allowed values are "Exact" or "Equivalent".

- Exact: match a request only if it exactly matches a specified rule. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, but "rules" only included apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would not be sent to the webhook.
- Equivalent: match a request if modifies a resource listed in rules, even via another API group or version. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, and "rules" only included apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would be converted to apps/v1 and sent to the webhook.

Defaults to "Equivalent"

- **webhooks.namespaceSelector** ([LabelSelector](#))

NamespaceSelector decides whether to run the webhook on an object based on whether the namespace for that object matches the selector. If the object itself is a namespace, the matching is performed on object.metadata.labels. If the object is another cluster scoped resource, it never skips the webhook.

For example, to run the webhook on any objects whose namespace is not associated with "runlevel" of "0" or "1"; you will set the selector as follows:
"namespaceSelector": { "matchExpressions": [{ "key": "runlevel", "operator": "NotIn", "values": ["0", "1"] }] }

If instead you want to only run the webhook on any objects whose namespace is associated with the "environment" of "prod" or "staging"; you will set the selector as follows: "namespaceSelector": { "matchExpressions": [{ "key": "environment", "operator": "In", "values": ["prod", "staging"] }] }

See <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/> for more examples of label selectors.

Default to the empty LabelSelector, which matches everything.

- **webhooks.objectSelector** ([LabelSelector](#))

ObjectSelector decides whether to run the webhook based on if the object has matching labels. objectSelector is evaluated against both the oldObject and newObject that would be sent to the webhook, and is considered to match if either object matches the selector. A null object (oldObject in the case of create, or newObject in the case of delete) or an object that cannot have labels (like a DeploymentRollback or a PodProxyOptions object) is not considered to match. Use the object selector only if the webhook is opt-in, because end users may skip the admission webhook by setting the labels. Default to the empty LabelSelector, which matches everything.

- **webhooks.reinvocationPolicy** (string)

reinvocationPolicy indicates whether this webhook should be called multiple times as part of a single admission evaluation. Allowed values are "Never" and "IfNeeded".

Never: the webhook will not be called more than once in a single admission evaluation.

IfNeeded: the webhook will be called at least one additional time as part of the admission evaluation if the object being admitted is modified by other admission plugins after the initial webhook call. Webhooks that specify this option *must* be idempotent, able to process objects they previously admitted. Note: * the number of additional invocations is not guaranteed to be exactly one. * if additional invocations result in further modifications to the object, webhooks are not guaranteed to be invoked again. * webhooks that use this option may be reordered to minimize the number of additional invocations. * to validate an object after all mutations are guaranteed complete, use a validating admission webhook instead.

Defaults to "Never".

- **webhooks.rules** ([]RuleWithOperations)

Rules describes what operations on what resources/subresources the webhook cares about. The webhook cares about an operation if it matches *any* Rule. However, in order to prevent ValidatingAdmissionWebhooks and MutatingAdmissionWebhooks from putting the cluster in a state which cannot be recovered from without completely disabling the plugin, ValidatingAdmissionWebhooks and MutatingAdmissionWebhooks are never called on admission requests for ValidatingWebhookConfiguration and MutatingWebhookConfiguration objects.

RuleWithOperations is a tuple of Operations and Resources. It is recommended to make sure that all the tuple expansions are valid.

- **webhooks.rules.apiGroups** ([]string)

Atomic: will be replaced during a merge

APIGroups is the API groups the resources belong to. '' is all groups. If '' is present, the length of the slice must be one. Required.

- **webhooks.rules.apiVersions** ([]string)

Atomic: will be replaced during a merge

APIVersions is the API versions the resources belong to. "*" is all versions.* If " is present, the length of the slice must be one. Required.

- **webhooks.rules.operations** ([]string)

Atomic: will be replaced during a merge

Operations is the operations the admission hook cares about - CREATE, UPDATE, DELETE, CONNECT or * for all of those operations and any future admission operations that are added. If '*' is present, the length of the slice must be one. Required.

- **webhooks.rules.resources** ([]string)

Atomic: will be replaced during a merge

Resources is a list of resources this rule applies to.

For example: 'pods' means pods. 'pods/log' means the log subresource of pods. "*" means all resources, but not subresources.* 'pods/' means all subresources of pods. '/scale' means all scale subresources. '*' means all resources and their subresources.

If wildcard is present, the validation rule will ensure resources do not overlap with each other.

Depending on the enclosing object, subresources might not be allowed. Required.

- **webhooks.rules.scope** (string)

scope specifies the scope of this rule. Valid values are "Cluster", "Namespaced", and "*" "Cluster" means that only cluster-scoped resources will match this rule. Namespace API objects are cluster-scoped. "Namespaced" means that only namespaced resources will match this rule.* " " means that there are no scope restrictions. Subresources match the scope of their parent resource. Default is "*".

- **webhooks.timeoutSeconds** (int32)

TimeoutSeconds specifies the timeout for this webhook. After the timeout passes, the webhook call will be ignored or the API call will fail based on the failure policy. The timeout value must be between 1 and 30 seconds. Default to 10 seconds.

MutatingWebhookConfigurationList

MutatingWebhookConfigurationList is a list of MutatingWebhookConfiguration.

-
- **apiVersion:** admissionregistration.k8s.io/v1
 - **kind:** MutatingWebhookConfigurationList
 - **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([] [MutatingWebhookConfiguration](#)), required

List of MutatingWebhookConfiguration.

Operations

get read the specified MutatingWebhookConfiguration

HTTP Request

GET /apis/admissionregistration.k8s.io/v1/mutatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required
name of the MutatingWebhookConfiguration
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([MutatingWebhookConfiguration](#)): OK

401: Unauthorized

list list or watch objects of kind MutatingWebhookConfiguration

HTTP Request

GET /apis/admissionregistration.k8s.io/v1/mutatingwebhookconfigurations

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)

- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)
- **watch** (*in query*): boolean
 - [watch](#)

Response

200 ([MutatingWebhookConfigurationList](#)): OK

401: Unauthorized

create create a MutatingWebhookConfiguration

HTTP Request

POST /apis/admissionregistration.k8s.io/v1/mutatingwebhookconfigurations

Parameters

- **body**: [MutatingWebhookConfiguration](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([MutatingWebhookConfiguration](#)): OK

201 ([MutatingWebhookConfiguration](#)): Created

202 ([MutatingWebhookConfiguration](#)): Accepted

401: Unauthorized

update replace the specified MutatingWebhookConfiguration

HTTP Request

PUT /apis/admissionregistration.k8s.io/v1/mutatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required

name of the MutatingWebhookConfiguration

- **body**: [MutatingWebhookConfiguration](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([MutatingWebhookConfiguration](#)): OK

201 ([MutatingWebhookConfiguration](#)): Created

401: Unauthorized

patch partially update the specified MutatingWebhookConfiguration

HTTP Request

PATCH /apis/admissionregistration.k8s.io/v1/mutatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required

name of the MutatingWebhookConfiguration

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([MutatingWebhookConfiguration](#)): OK

201 ([MutatingWebhookConfiguration](#)): Created

401: Unauthorized

delete delete a MutatingWebhookConfiguration

HTTP Request

DELETE /apis/admissionregistration.k8s.io/v1/mutatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required
name of the MutatingWebhookConfiguration
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of MutatingWebhookConfiguration

HTTP Request

DELETE /apis/admissionregistration.k8s.io/v1/mutatingwebhookconfigurations

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ValidatingWebhookConfiguration

ValidatingWebhookConfiguration describes the configuration of and admission webhook that accept or reject and object without changing it.

apiVersion: admissionregistration.k8s.io/v1

import "k8s.io/api/admissionregistration/v1"

ValidatingWebhookConfiguration

ValidatingWebhookConfiguration describes the configuration of an admission webhook that accept or reject an object without changing it.

- **apiVersion**: admissionregistration.k8s.io/v1
- **kind**: ValidatingWebhookConfiguration
- **metadata** ([ObjectMeta](#))

Standard object metadata; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>.

- **webhooks** ([]ValidatingWebhook)

Patch strategy: merge on key name

Webhooks is a list of webhooks and the affected resources and operations.

ValidatingWebhook describes an admission webhook and the resources and operations it applies to.

- **webhooks.admissionReviewVersions** ([]string), required

AdmissionReviewVersions is an ordered list of preferred AdmissionReview versions the Webhook expects. API server will try to use first version in the list which it supports. If none of the versions specified in this list supported by API server, validation will fail for this object. If a persisted webhook configuration specifies allowed versions and does not include any versions known to the API Server, calls to the webhook will fail and be subject to the failure policy.

- **webhooks.clientConfig** (WebhookClientConfig), required

ClientConfig defines how to communicate with the hook. Required

WebhookClientConfig contains the information to make a TLS connection with the webhook

- **webhooks.clientConfig.caBundle** ([]byte)

caBundle is a PEM encoded CA bundle which will be used to validate the webhook's server certificate. If unspecified, system trust roots on the apiserver are used.

- **webhooks.clientConfig.service** (ServiceReference)

service is a reference to the service for this webhook. Either service or url must be specified.

If the webhook is running within the cluster, then you should use service.

ServiceReference holds a reference to Service.legacy.k8s.io

- **webhooks.clientConfig.service.name** (string), required

name is the name of the service. Required

- **webhooks.clientConfig.service.namespace** (string), required

namespace is the namespace of the service. Required

- **webhooks.clientConfig.service.path** (string)

path is an optional URL path which will be sent in any request to this service.

- **webhooks.clientConfig.service.port** (int32)

If specified, the port on the service that hosting webhook. Default to 443 for backward compatibility. port should be a valid port number (1-65535, inclusive).

- **webhooks.clientConfig.url** (string)

url gives the location of the webhook, in standard URL form (scheme://host:port/path). Exactly one of url or service must be specified.

The host should not refer to a service running in the cluster; use the service field instead. The host might be resolved via external DNS in some apiservers (e.g., kube-apiserver cannot resolve in-cluster DNS as that would be a layering violation). host may also be an IP address.

Please note that using localhost or 127.0.0.1 as a host is risky unless you take great care to run this webhook on all hosts which run an apiserver which might need to make calls to this webhook. Such installs are likely to be non-portable, i.e., not easy to turn up in a new cluster.

The scheme must be "https"; the URL must begin with "https://".

A path is optional, and if present may be any string permissible in a URL. You may use the path to pass an arbitrary string to the webhook, for example, a cluster identifier.

Attempting to use a user or basic auth e.g. "user:password@" is not allowed. Fragments ("#...") and query parameters ("?...") are not allowed, either.

- **webhooks.name** (string), required

The name of the admission webhook. Name should be fully qualified, e.g., imagepolicy.kubernetes.io, where "imagepolicy" is the name of the webhook, and kubernetes.io is the name of the organization. Required.

- **webhooks.sideEffects** (string), required

SideEffects states whether this webhook has side effects. Acceptable values are: None, NoneOnDryRun (webhooks created via v1beta1 may also specify Some or

Unknown). Webhooks with side effects MUST implement a reconciliation system, since a request may be rejected by a future step in the admission chain and the side effects therefore need to be undone. Requests with the dryRun attribute will be auto-rejected if they match a webhook with sideEffects == Unknown or Some.

- **webhooks.failurePolicy** (string)

FailurePolicy defines how unrecognized errors from the admission endpoint are handled - allowed values are Ignore or Fail. Defaults to Fail.

- **webhooks.matchConditions** ([]MatchCondition)

Patch strategy: merge on key name

Map: unique values on key name will be kept during a merge

MatchConditions is a list of conditions that must be met for a request to be sent to this webhook. Match conditions filter requests that have already been matched by the rules, namespaceSelector, and objectSelector. An empty list of matchConditions matches all requests. There are a maximum of 64 match conditions allowed.

The exact matching logic is (in order):

1. If ANY matchCondition evaluates to FALSE, the webhook is skipped.
2. If ALL matchConditions evaluate to TRUE, the webhook is called.
3. If any matchCondition evaluates to an error (but none are FALSE):
 - If failurePolicy=Fail, reject the request
 - If failurePolicy=Ignore, the error is ignored and the webhook is skipped

This is a beta feature and managed by the AdmissionWebhookMatchConditions feature gate.

MatchCondition represents a condition which must be fulfilled for a request to be sent to a webhook.

- **webhooks.matchConditions.expression** (string), required

Expression represents the expression which will be evaluated by CEL. Must evaluate to bool. CEL expressions have access to the contents of the AdmissionRequest and Authorizer, organized into CEL variables:

'object' - The object from the incoming request. The value is null for DELETE requests.
'oldObject' - The existing object. The value is null for CREATE requests.
'request' - Attributes of the admission request(/pkg/apis/admission/types.go#AdmissionRequest).
'authorizer' - A CEL Authorizer. May be used to perform authorization checks for the principal (user or service account) of the request. See <https://pkg.go.dev/k8s.io/apiserver/pkg/cel/library#Authz>
'authorizer.requestResource' - A CEL ResourceCheck constructed from the 'authorizer' and configured with the request resource. Documentation on CEL: <https://kubernetes.io/docs/reference/using-api/cel/>

Required.

- **webhooks.matchConditions.name** (string), required

Name is an identifier for this match condition, used for strategic merging of MatchConditions, as well as providing an identifier for logging purposes. A good name should be descriptive of the associated expression. Name must be a qualified name consisting of alphanumeric characters, '-' , '' or '.', and must start and end with an alphanumeric character (e.g. 'MyName', or 'my.name', or '123-abc', regex used for validation is '([A-Za-z0-9][-A-Za-z0-9.]*)?[A-Za-z0-9]') with an optional DNS subdomain prefix and '/' (e.g. 'example.com/MyName')

Required.

- **webhooks.matchPolicy** (string)

matchPolicy defines how the "rules" list is used to match incoming requests. Allowed values are "Exact" or "Equivalent".

- Exact: match a request only if it exactly matches a specified rule. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, but "rules" only included apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would not be sent to the webhook.
- Equivalent: match a request if modifies a resource listed in rules, even via another API group or version. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, and "rules" only included apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would be converted to apps/v1 and sent to the webhook.

Defaults to "Equivalent"

- **webhooks.namespaceSelector** ([LabelSelector](#))

NamespaceSelector decides whether to run the webhook on an object based on whether the namespace for that object matches the selector. If the object itself is a namespace, the matching is performed on object.metadata.labels. If the object is another cluster scoped resource, it never skips the webhook.

For example, to run the webhook on any objects whose namespace is not associated with "runlevel" of "0" or "1"; you will set the selector as follows:
"namespaceSelector": { "matchExpressions": [{ "key": "runlevel", "operator": "NotIn", "values": ["0", "1"] }] }

If instead you want to only run the webhook on any objects whose namespace is associated with the "environment" of "prod" or "staging"; you will set the selector as follows: "namespaceSelector": { "matchExpressions": [{ "key": "environment", "operator": "In", "values": ["prod", "staging"] }] }

See <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels> for more examples of label selectors.

Default to the empty LabelSelector, which matches everything.

- **webhooks.objectSelector** ([LabelSelector](#))

ObjectSelector decides whether to run the webhook based on if the object has matching labels. objectSelector is evaluated against both the oldObject and newObject that would be sent to the webhook, and is considered to match if either object matches the selector. A null object (oldObject in the case of create, or newObject in the case of delete) or an object that cannot have labels (like a DeploymentRollback or a PodProxyOptions object) is not considered to match. Use the object selector only if the webhook is opt-in, because end users may skip the admission webhook by setting the labels. Default to the empty LabelSelector, which matches everything.

- **webhooks.rules** ([]RuleWithOperations)

Rules describes what operations on what resources/subresources the webhook cares about. The webhook cares about an operation if it matches *any* Rule. However, in order to prevent ValidatingAdmissionWebhooks and MutatingAdmissionWebhooks from putting the cluster in a state which cannot be recovered from without completely disabling the plugin, ValidatingAdmissionWebhooks and MutatingAdmissionWebhooks are never called on admission requests for ValidatingWebhookConfiguration and MutatingWebhookConfiguration objects.

RuleWithOperations is a tuple of Operations and Resources. It is recommended to make sure that all the tuple expansions are valid.

- **webhooks.rules.apiGroups** ([]string)

Atomic: will be replaced during a merge

APIGroups is the API groups the resources belong to. '' is all groups. If '' is present, the length of the slice must be one. Required.

- **webhooks.rules.apiVersions** ([]string)

Atomic: will be replaced during a merge

APIVersions is the API versions the resources belong to. '' is all versions. If '' is present, the length of the slice must be one. Required.

- **webhooks.rules.operations** ([]string)

Atomic: will be replaced during a merge

Operations is the operations the admission hook cares about - CREATE, UPDATE, DELETE, CONNECT or * for all of those operations and any future admission operations that are added. If '*' is present, the length of the slice must be one. Required.

- **webhooks.rules.resources** ([]string)

Atomic: will be replaced during a merge

Resources is a list of resources this rule applies to.

For example: 'pods' means pods. 'pods/log' means the log subresource of pods. '' means all resources, but not subresources. 'pods/' means all subresources

of pods. '/scale' means all scale subresources. '*' means all resources and their subresources.

If wildcard is present, the validation rule will ensure resources do not overlap with each other.

Depending on the enclosing object, subresources might not be allowed.
Required.

- **webhooks.rules.scope** (string)

scope specifies the scope of this rule. Valid values are "Cluster", "Namespaced", and "" "*Cluster*" means that only cluster-scoped resources will match this rule. Namespace API objects are cluster-scoped. "Namespaced" means that only namespaced resources will match this rule. "" means that there are no scope restrictions. Subresources match the scope of their parent resource. Default is "*".

- **webhooks.timeoutSeconds** (int32)

TimeoutSeconds specifies the timeout for this webhook. After the timeout passes, the webhook call will be ignored or the API call will fail based on the failure policy. The timeout value must be between 1 and 30 seconds. Default to 10 seconds.

ValidatingWebhookConfigurationList

ValidatingWebhookConfigurationList is a list of ValidatingWebhookConfiguration.

- **apiVersion**: admissionregistration.k8s.io/v1

- **kind**: ValidatingWebhookConfigurationList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][ValidatingWebhookConfiguration](#)), required

List of ValidatingWebhookConfiguration.

Operations

get read the specified ValidatingWebhookConfiguration

HTTP Request

GET /apis/admissionregistration.k8s.io/v1/validatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required
name of the ValidatingWebhookConfiguration
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ValidatingWebhookConfiguration](#)): OK

401: Unauthorized

list list or watch objects of kind ValidatingWebhookConfiguration

HTTP Request

GET /apis/admissionregistration.k8s.io/v1/validatingwebhookconfigurations

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ValidatingWebhookConfigurationList](#)): OK

401: Unauthorized

create create a ValidatingWebhookConfiguration

HTTP Request

POST /apis/admissionregistration.k8s.io/v1/validatingwebhookconfigurations

Parameters

- **body**: [ValidatingWebhookConfiguration](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ValidatingWebhookConfiguration](#)): OK

201 ([ValidatingWebhookConfiguration](#)): Created

202 ([ValidatingWebhookConfiguration](#)): Accepted

401: Unauthorized

update replace the specified ValidatingWebhookConfiguration

HTTP Request

PUT /apis/admissionregistration.k8s.io/v1/validatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ValidatingWebhookConfiguration
- **body**: [ValidatingWebhookConfiguration](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ValidatingWebhookConfiguration](#)): OK

201 ([ValidatingWebhookConfiguration](#)): Created

401: Unauthorized

patch partially update the specified ValidatingWebhookConfiguration

HTTP Request

PATCH /apis/admissionregistration.k8s.io/v1/validatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ValidatingWebhookConfiguration

- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ValidatingWebhookConfiguration](#)): OK

201 ([ValidatingWebhookConfiguration](#)): Created

401: Unauthorized

delete delete a ValidatingWebhookConfiguration

HTTP Request

DELETE /apis/admissionregistration.k8s.io/v1/validatingwebhookconfigurations/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ValidatingWebhookConfiguration
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ValidatingWebhookConfiguration

HTTP Request

DELETE /apis/admissionregistration.k8s.io/v1/validatingwebhookconfigurations

Parameters

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

ValidatingAdmissionPolicy v1beta1

ValidatingAdmissionPolicy describes the definition of an admission validation policy that accepts or rejects an object without changing it.

```
apiVersion: admissionregistration.k8s.io/v1beta1
import "k8s.io/api/admissionregistration/v1beta1"
```

ValidatingAdmissionPolicy

ValidatingAdmissionPolicy describes the definition of an admission validation policy that accepts or rejects an object without changing it.

-
- **apiVersion**: admissionregistration.k8s.io/v1beta1
 - **kind**: ValidatingAdmissionPolicy
 - **metadata** ([ObjectMeta](#))

Standard object metadata; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>.

- **spec** (ValidatingAdmissionPolicySpec)

Specification of the desired behavior of the ValidatingAdmissionPolicy.

ValidatingAdmissionPolicySpec is the specification of the desired behavior of the *AdmissionPolicy*.

- **spec.auditAnnotations** ([]AuditAnnotation)

Atomic: will be replaced during a merge

auditAnnotations contains CEL expressions which are used to produce audit annotations for the audit event of the API request. validations and *auditAnnotations* may not both be empty; at least one of validations or *auditAnnotations* is required.

AuditAnnotation describes how to produce an audit annotation for an API request.

- **spec.auditAnnotations.key** (string), required

key specifies the audit annotation key. The audit annotation keys of a *ValidatingAdmissionPolicy* must be unique. The key must be a qualified name ([A-Za-z0-9][A-Za-z0-9_.]*) no more than 63 bytes in length.

The key is combined with the resource name of the *ValidatingAdmissionPolicy* to construct an audit annotation key: "{*ValidatingAdmissionPolicy* name}/{key}".

If an admission webhook uses the same resource name as this *ValidatingAdmissionPolicy* and the same audit annotation key, the annotation key will be identical. In this case, the first annotation written with the key will be included in the audit event and all subsequent annotations with the same key will be discarded.

Required.

- **spec.auditAnnotations.valueExpression** (string), required

valueExpression represents the expression which is evaluated by CEL to produce an audit annotation value. The expression must evaluate to either a string or null value. If the expression evaluates to a string, the audit annotation is included with the string value. If the expression evaluates to null or empty string the audit annotation will be omitted. The *valueExpression* may be no longer than 5kb in length. If the result of the *valueExpression* is more than 10kb in length, it will be truncated to 10kb.

If multiple *ValidatingAdmissionPolicyBinding* resources match an API request, then the *valueExpression* will be evaluated for each binding. All unique values produced by the *valueExpressions* will be joined together in a comma-separated list.

Required.

- **spec.failurePolicy** (string)

failurePolicy defines how to handle failures for the admission policy. Failures can occur from CEL expression parse errors, type check errors, runtime errors and invalid or mis-configured policy definitions or bindings.

A policy is invalid if spec.paramKind refers to a non-existent Kind. A binding is invalid if spec.paramRef.name refers to a non-existent resource.

failurePolicy does not define how validations that evaluate to false are handled.

When failurePolicy is set to Fail, ValidatingAdmissionPolicyBinding validationActions define how failures are enforced.

Allowed values are Ignore or Fail. Defaults to Fail.

- **spec.matchConditions** ([]MatchCondition)

Patch strategy: merge on key name

Map: unique values on key name will be kept during a merge

MatchConditions is a list of conditions that must be met for a request to be validated. Match conditions filter requests that have already been matched by the rules, namespaceSelector, and objectSelector. An empty list of matchConditions matches all requests. There are a maximum of 64 match conditions allowed.

If a parameter object is provided, it can be accessed via the params handle in the same manner as validation expressions.

The exact matching logic is (in order):

1. If ANY matchCondition evaluates to FALSE, the policy is skipped.
2. If ALL matchConditions evaluate to TRUE, the policy is evaluated.
3. If any matchCondition evaluates to an error (but none are FALSE):
 - If failurePolicy=Fail, reject the request
 - If failurePolicy=Ignore, the policy is skipped

MatchCondition represents a condition which must be fulfilled for a request to be sent to a webhook.

- **spec.matchConditions.expression** (string), required

Expression represents the expression which will be evaluated by CEL. Must evaluate to bool. CEL expressions have access to the contents of the AdmissionRequest and Authorizer, organized into CEL variables:

'object' - The object from the incoming request. The value is null for DELETE requests.
'oldObject' - The existing object. The value is null for CREATE requests.
'request' - Attributes of the admission request(/pkg/apis/admission/types.go#AdmissionRequest).
'authorizer' - A CEL Authorizer. May be used to perform authorization checks for the principal (user or service account) of the request. See <https://pkg.go.dev/k8s.io/apiserver/pkg/cel/library#Authz>
'authorizer.requestResource' - A CEL ResourceCheck constructed from the 'authorizer' and configured with the request resource. Documentation on CEL: <https://kubernetes.io/docs/reference/using-api/cel/>

Required.

- **spec.matchConditions.name** (string), required

Name is an identifier for this match condition, used for strategic merging of MatchConditions, as well as providing an identifier for logging purposes. A good name should be descriptive of the associated expression. Name must be a qualified name consisting of alphanumeric characters, '-' , '' or '.', and must start and end with an alphanumeric character (e.g. 'MyName', or 'my.name', or '123-abc', regex used for validation is '([A-Za-z0-9][-A-Za-z0-9.]*)?[A-Za-z0-9]') with an optional DNS subdomain prefix and '/' (e.g. 'example.com/MyName')

Required.

- **spec.matchConstraints** (MatchResources)

MatchConstraints specifies what resources this policy is designed to validate. The AdmissionPolicy cares about a request if it matches *all* Constraints. However, in order to prevent clusters from being put into an unstable state that cannot be recovered from via the API ValidatingAdmissionPolicy cannot match ValidatingAdmissionPolicy and ValidatingAdmissionPolicyBinding. Required.

MatchResources decides whether to run the admission control policy on an object based on whether it meets the match criteria. The exclude rules take precedence over include rules (if a resource matches both, it is excluded)

- **spec.matchConstraints.excludeResourceRules**
([]NamedRuleWithOperations)

Atomic: will be replaced during a merge

ExcludeResourceRules describes what operations on what resources/ subresources the ValidatingAdmissionPolicy should not care about. The exclude rules take precedence over include rules (if a resource matches both, it is excluded)

NamedRuleWithOperations is a tuple of Operations and Resources with ResourceNames.

- **spec.matchConstraints.excludeResourceRules.apiGroups**
([]string)

Atomic: will be replaced during a merge

APIGroups is the API groups the resources belong to. '' is all groups. If '' is present, the length of the slice must be one. Required.

- **spec.matchConstraints.excludeResourceRules.apiVersions**
([]string)

Atomic: will be replaced during a merge

APIVersions is the API versions the resources belong to. '' is all versions. If '' is present, the length of the slice must be one. Required.

- **spec.matchConstraints.excludeResourceRules.operations**
([]string)

Atomic: will be replaced during a merge

Operations is the operations the admission hook cares about - CREATE, UPDATE, DELETE, CONNECT or '*' for all of those operations and any future admission operations that are added. If '*' is present, the length of the slice must be one. Required.

- **spec.matchConstraints.excludeResourceRules.resourceNames** ([]string)

Atomic: will be replaced during a merge

ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed.

- **spec.matchConstraints.excludeResourceRules.resources** ([]string)

Atomic: will be replaced during a merge

Resources is a list of resources this rule applies to.

For example: 'pods' means pods. 'pods/log' means the log subresource of pods. '*' means all resources, but not subresources. 'pods/' means all subresources of pods. '/scale' means all scale subresources. '/'* means all resources and their subresources.

If wildcard is present, the validation rule will ensure resources do not overlap with each other.

Depending on the enclosing object, subresources might not be allowed. Required.

- **spec.matchConstraints.excludeResourceRules.scope** (string)

scope specifies the scope of this rule. Valid values are "Cluster", "Namespaced", and "" "*Cluster*" means that only cluster-scoped resources will match this rule. Namespace API objects are cluster-scoped.

"Namespaced" means that only namespaced resources will match this rule. "" means that there are no scope restrictions. Subresources match the scope of their parent resource. Default is "".

- **spec.matchConstraints.matchPolicy** (string)

matchPolicy defines how the "MatchResources" list is used to match incoming requests. Allowed values are "Exact" or "Equivalent".

- Exact: match a request only if it exactly matches a specified rule. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, but "rules" only included apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would not be sent to the ValidatingAdmissionPolicy.

- Equivalent: match a request if modifies a resource listed in rules, even via another API group or version. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, and

"rules" only included apiGroups:["apps"], apiVersions:["v1"], resources:["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would be converted to apps/v1 and sent to the ValidatingAdmissionPolicy.

Defaults to "Equivalent"

- **spec.matchConstraints.namespaceSelector** ([LabelSelector](#))

NamespaceSelector decides whether to run the admission control policy on an object based on whether the namespace for that object matches the selector. If the object itself is a namespace, the matching is performed on object.metadata.labels. If the object is another cluster scoped resource, it never skips the policy.

For example, to run the webhook on any objects whose namespace is not associated with "runlevel" of "0" or "1"; you will set the selector as follows:
"namespaceSelector": { "matchExpressions": [{ "key": "runlevel", "operator": "NotIn", "values": ["0", "1"] }] }

If instead you want to only run the policy on any objects whose namespace is associated with the "environment" of "prod" or "staging"; you will set the selector as follows: "namespaceSelector": { "matchExpressions": [{ "key": "environment", "operator": "In", "values": ["prod", "staging"] }] }

See <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/> for more examples of label selectors.

Default to the empty LabelSelector, which matches everything.

- **spec.matchConstraints.objectSelector** ([LabelSelector](#))

ObjectSelector decides whether to run the validation based on if the object has matching labels. objectSelector is evaluated against both the oldObject and newObject that would be sent to the cel validation, and is considered to match if either object matches the selector. A null object (oldObject in the case of create, or newObject in the case of delete) or an object that cannot have labels (like a DeploymentRollback or a PodProxyOptions object) is not considered to match. Use the object selector only if the webhook is opt-in, because end users may skip the admission webhook by setting the labels.
Default to the empty LabelSelector, which matches everything.

- **spec.matchConstraints.resourceRules** ([]NamedRuleWithOperations)

Atomic: will be replaced during a merge

ResourceRules describes what operations on what resources/subresources the ValidatingAdmissionPolicy matches. The policy cares about an operation if it matches *any* Rule.

NamedRuleWithOperations is a tuple of Operations and Resources with ResourceNames.

- **spec.matchConstraints.resourceRules.apiGroups** ([]string)

Atomic: will be replaced during a merge

APIGroups is the API groups the resources belong to. "*" is all groups. If*" is present, the length of the slice must be one. Required.

- **spec.matchConstraints.resourceRules.apiVersions ([]string)**

Atomic: will be replaced during a merge

APIVersions is the API versions the resources belong to. "*" is all versions. If*" is present, the length of the slice must be one. Required.

- **spec.matchConstraints.resourceRules.operations ([]string)**

Atomic: will be replaced during a merge

Operations is the operations the admission hook cares about - CREATE, UPDATE, DELETE, CONNECT or * for all of those operations and any future admission operations that are added. If '*' is present, the length of the slice must be one. Required.

- **spec.matchConstraints.resourceRules.resourceNames ([]string)**

Atomic: will be replaced during a merge

ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed.

- **spec.matchConstraints.resourceRules.resources ([]string)**

Atomic: will be replaced during a merge

Resources is a list of resources this rule applies to.

For example: 'pods' means pods. 'pods/log' means the log subresource of pods. "*" means all resources, but not subresources. 'pods/' means all subresources of pods. '/scale' means all scale subresources. '/' means all resources and their subresources.*

If wildcard is present, the validation rule will ensure resources do not overlap with each other.

Depending on the enclosing object, subresources might not be allowed. Required.

- **spec.matchConstraints.resourceRules.scope (string)**

scope specifies the scope of this rule. Valid values are "Cluster", "Namespaced", and "" "*" Cluster" means that only cluster-scoped resources will match this rule. Namespace API objects are cluster-scoped. "Namespaced" means that only namespaced resources will match this rule. "" means that there are no scope restrictions. Subresources match the scope of their parent resource. Default is "*".*

- **spec.paramKind (ParamKind)**

ParamKind specifies the kind of resources used to parameterize this policy. If absent, there are no parameters for this policy and the param CEL variable will not be provided to validation expressions. If ParamKind refers to a non-existent kind, this policy definition is mis-configured and the FailurePolicy is applied. If paramKind is specified but paramRef is unset in ValidatingAdmissionPolicyBinding, the params variable will be null.

ParamKind is a tuple of Group Kind and Version.

- **spec.paramKind.apiVersion** (string)

APIVersion is the API group version the resources belong to. In format of "group/version". Required.

- **spec.paramKind.kind** (string)

Kind is the API kind the resources belong to. Required.

- **spec.validations** ([]Validation)

Atomic: will be replaced during a merge

Validations contain CEL expressions which is used to apply the validation. Validations and AuditAnnotations may not both be empty; a minimum of one Validations or AuditAnnotations is required.

Validation specifies the CEL expression which is used to apply the validation.

- **spec.validations.expression** (string), required

Expression represents the expression which will be evaluated by CEL. ref: <https://github.com/google/cel-spec> CEL expressions have access to the contents of the API request/response, organized into CEL variables as well as some other useful variables:

- 'object' - The object from the incoming request. The value is null for DELETE requests. - 'oldObject' - The existing object. The value is null for CREATE requests. - 'request' - Attributes of the API request([ref](#)). - 'params' - Parameter resource referred to by the policy binding being evaluated. Only populated if the policy has a ParamKind. - 'namespaceObject' - The namespace object that the incoming object belongs to. The value is null for cluster-scoped resources. - 'variables' - Map of composed variables, from its name to its lazily evaluated value. For example, a variable named 'foo' can be accessed as 'variables.foo'.
- 'authorizer' - A CEL Authorizer. May be used to perform authorization checks for the principal (user or service account) of the request. See <https://pkg.go.dev/k8s.io/apiserver/pkg/cel/library#Authz>
- 'authorizer.requestResource' - A CEL ResourceCheck constructed from the 'authorizer' and configured with the request resource.

The apiVersion, kind, metadata.name and metadata.generateName are always accessible from the root of the object. No other metadata properties are accessible.

Only property names of the form [a-zA-Z_-.][a-zA-Z0-9_-.]* are accessible. Accessible property names are escaped according to the following rules when accessed in the expression: - " escapes to 'underscores' - '.' escapes to 'dot' - '-' escapes to 'dash' - '/' escapes to 'slash' - **Property names that exactly match a CEL RESERVED keyword escape to '{keyword}__'**. The keywords are: "true", "false", "null", "in", "as", "break", "const", "continue", "else", "for", "function", "if", "import", "let", "loop", "package", "namespace", "return". Examples:

- Expression accessing a property named "namespace": {"Expression": "object.namespace > 0"}
- Expression accessing a property named "x-prop": {"Expression": "object.x_dash_prop > 0"}
- Expression accessing a property named "redact_d": {"Expression": "object.redact_underscores_d > 0"}

Equality on arrays with list type of 'set' or 'map' ignores element order, i.e. [1, 2] == [2, 1]. Concatenation on arrays with x-kubernetes-list-type use the semantics of the list type:

- 'set': X + Y performs a union where the array positions of all elements in X are preserved and non-intersecting elements in Y are appended, retaining their partial order.
- 'map': X + Y performs a merge where the array positions of all keys in X are preserved but the values are overwritten by values in Y when the key sets of X and Y intersect. Elements in Y with non-intersecting keys are appended, retaining their partial order. Required.

- **spec.validations.message** (string)

Message represents the message displayed when validation fails. The message is required if the Expression contains line breaks. The message must not contain line breaks. If unset, the message is "failed rule: {Rule}". e.g. "must be a URL with the host matching spec.host" If the Expression contains line breaks. Message is required. The message must not contain line breaks. If unset, the message is "failed Expression: {Expression}".

- **spec.validations.messageExpression** (string)

messageExpression declares a CEL expression that evaluates to the validation failure message that is returned when this rule fails. Since messageExpression is used as a failure message, it must evaluate to a string. If both message and messageExpression are present on a validation, then messageExpression will be used if validation fails. If messageExpression results in a runtime error, the runtime error is logged, and the validation failure message is produced as if the messageExpression field were unset. If messageExpression evaluates to an empty string, a string with only spaces, or a string that contains line breaks, then the validation failure message will also be produced as if the messageExpression field were unset, and the fact that messageExpression produced an empty string/string with only spaces/string with line breaks will be logged. messageExpression has access to all the same variables as the expression except for 'authorizer' and 'authorizer.requestResource'. Example: "object.x must be less than max (+string(params.max)+)"

- **spec.validations.reason** (string)

- Reason represents a machine-readable description of why this validation failed. If this is the first validation in the list to fail, this reason, as well as the corresponding HTTP response code, are used in the HTTP response to the client. The currently supported reasons are: "Unauthorized", "Forbidden", "Invalid", "RequestEntityTooLarge". If not set, StatusReasonInvalid is used in the response to the client.

- **spec.variables** ([]Variable)

Patch strategy: merge on key name

Map: unique values on key name will be kept during a merge

Variables contain definitions of variables that can be used in composition of other expressions. Each variable is defined as a named CEL expression. The variables defined here will be available under variables in other expressions of the policy except MatchConditions because MatchConditions are evaluated before the rest of the policy.

The expression of a variable can refer to other variables defined earlier in the list but not those after. Thus, Variables must be sorted by the order of first appearance and acyclic.

Variable is the definition of a variable that is used for composition. A variable is defined as a named expression.

- **spec.variables.expression** (string), required

- Expression is the expression that will be evaluated as the value of the variable. The CEL expression has access to the same identifiers as the CEL expressions in Validation.

- **spec.variables.name** (string), required

- Name is the name of the variable. The name must be a valid CEL identifier and unique among all variables. The variable can be accessed in other expressions through variables. For example, if name is "foo", the variable will be available as variables.foo

- **status** (ValidatingAdmissionPolicyStatus)

The status of the ValidatingAdmissionPolicy, including warnings that are useful to determine if the policy behaves in the expected way. Populated by the system. Read-only.

ValidatingAdmissionPolicyStatus represents the status of an admission validation policy.

- **status.conditions** ([]Condition)

Map: unique values on key type will be kept during a merge

The conditions represent the latest available observations of a policy's current state.

Condition contains details for one aspect of the current state of this API Resource.

- **status.conditions.lastTransitionTime** (Time), required

lastTransitionTime is the last time the condition transitioned from one status to another. This should be when the underlying condition changed. If that is not known, then using the time when the API field changed is acceptable.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **status.conditions.message** (string), required

message is a human readable message indicating details about the transition. This may be an empty string.

- **status.conditions.reason** (string), required

reason contains a programmatic identifier indicating the reason for the condition's last transition. Producers of specific condition types may define expected values and meanings for this field, and whether the values are considered a guaranteed API. The value should be a CamelCase string. This field may not be empty.

- **status.conditions.status** (string), required

status of the condition, one of True, False, Unknown.

- **status.conditions.type** (string), required

type of condition in CamelCase or in foo.example.com/CamelCase.

- **status.conditions.observedGeneration** (int64)

observedGeneration represents the .metadata.generation that the condition was set based upon. For instance, if .metadata.generation is currently 12, but the .status.conditions[x].observedGeneration is 9, the condition is out of date with respect to the current state of the instance.

- **status.observedGeneration** (int64)

The generation observed by the controller.

- **status.typeChecking** (TypeChecking)

The results of type checking for each expression. Presence of this field indicates the completion of the type checking.

TypeChecking contains results of type checking the expressions in the ValidatingAdmissionPolicy

- **status.typeChecking.expressionWarnings** ([]ExpressionWarning)

Atomic: will be replaced during a merge

The type checking warnings for each expression.

ExpressionWarning is a warning information that targets a specific expression.

- **status.typeChecking.expressionWarnings.fieldRef** (string), required

The path to the field that refers the expression. For example, the reference to the expression of the first item of validations is "spec.validations[0].expression"

- **status.typeChecking.expressionWarnings.warning** (string), required

The content of type checking information in a human-readable form. Each line of the warning contains the type that the expression is checked against, followed by the type check error from the compiler.

ValidatingAdmissionPolicyList

ValidatingAdmissionPolicyList is a list of ValidatingAdmissionPolicy.

- **apiVersion** (string)

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>

- **items** ([][ValidatingAdmissionPolicy](#))

List of ValidatingAdmissionPolicy.

- **kind** (string)

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

ValidatingAdmissionPolicyBinding

ValidatingAdmissionPolicyBinding binds the ValidatingAdmissionPolicy with parameterized resources. ValidatingAdmissionPolicyBinding and parameter CRDs together define how cluster administrators configure policies for clusters.

For a given admission request, each binding will cause its policy to be evaluated N times, where N is 1 for policies/bindings that don't use params, otherwise N is the number of parameters selected by the binding.

The CEL expressions of a policy must have a computed CEL cost below the maximum CEL budget. Each evaluation of the policy is given an independent CEL cost budget. Adding/removing policies, bindings, or params can not affect whether a given (policy, binding, param) combination is within its own CEL budget.

- **apiVersion** (string)

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>

- **kind** (string)

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **metadata** ([ObjectMeta](#))

Standard object metadata; More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>.

- **spec** (ValidatingAdmissionPolicyBindingSpec)

Specification of the desired behavior of the ValidatingAdmissionPolicyBinding.

ValidatingAdmissionPolicyBindingSpec is the specification of the ValidatingAdmissionPolicyBinding.

- **spec.matchResources** (MatchResources)

MatchResources declares what resources match this binding and will be validated by it. Note that this is intersected with the policy's matchConstraints, so only requests that are matched by the policy can be selected by this. If this is unset, all resources matched by the policy are validated by this binding. When resourceRules is unset, it does not constrain resource matching. If a resource is matched by the other fields of this object, it will be validated. Note that this is differs from ValidatingAdmissionPolicy matchConstraints, where resourceRules are required.

MatchResources decides whether to run the admission control policy on an object based on whether it meets the match criteria. The exclude rules take precedence over include rules (if a resource matches both, it is excluded)

- **spec.matchResources.excludeResourceRules** ([]NamedRuleWithOperations)

Atomic: will be replaced during a merge

ExcludeResourceRules describes what operations on what resources/subresources the ValidatingAdmissionPolicy should not care about. The exclude rules take precedence over include rules (if a resource matches both, it is excluded)

NamedRuleWithOperations is a tuple of Operations and Resources with ResourceNames.

- **spec.matchResources.excludeResourceRules.apiGroups ([]string)**

Atomic: will be replaced during a merge

APIGroups is the API groups the resources belong to. '' is all groups. If '' is present, the length of the slice must be one. Required.

- **spec.matchResources.excludeResourceRules.apiVersions ([]string)**

Atomic: will be replaced during a merge

APIVersions is the API versions the resources belong to. '' is all versions. If '' is present, the length of the slice must be one. Required.

- **spec.matchResources.excludeResourceRules.operations ([]string)**

Atomic: will be replaced during a merge

Operations is the operations the admission hook cares about - CREATE, UPDATE, DELETE, CONNECT or * for all of those operations and any future admission operations that are added. If '*' is present, the length of the slice must be one. Required.

- **spec.matchResources.excludeResourceRules.resourceNames ([]string)**

Atomic: will be replaced during a merge

ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed.

- **spec.matchResources.excludeResourceRules.resources ([]string)**

Atomic: will be replaced during a merge

Resources is a list of resources this rule applies to.

For example: 'pods' means pods. 'pods/log' means the log subresource of pods. '' means all resources, but not subresources. 'pods/' means all subresources of pods. '/scale' means all scale subresources. '/'* means all resources and their subresources.

If wildcard is present, the validation rule will ensure resources do not overlap with each other.

Depending on the enclosing object, subresources might not be allowed. Required.

- **spec.matchResources.excludeResourceRules.scope** (string)

scope specifies the scope of this rule. Valid values are "Cluster", "Namespaced", and "" "*Cluster*" means that only cluster-scoped resources will match this rule. Namespace API objects are cluster-scoped. "*Namespaced*" means that only namespaced resources will match this rule. "" means that there are no scope restrictions. Subresources match the scope of their parent resource. Default is "*".

- **spec.matchResources.matchPolicy** (string)

matchPolicy defines how the "MatchResources" list is used to match incoming requests. Allowed values are "Exact" or "Equivalent".

- Exact: match a request only if it exactly matches a specified rule. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, but "rules" only included apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would not be sent to the ValidatingAdmissionPolicy.
- Equivalent: match a request if modifies a resource listed in rules, even via another API group or version. For example, if deployments can be modified via apps/v1, apps/v1beta1, and extensions/v1beta1, and "rules" only included apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"], a request to apps/v1beta1 or extensions/v1beta1 would be converted to apps/v1 and sent to the ValidatingAdmissionPolicy.

Defaults to "Equivalent"

- **spec.matchResources.namespaceSelector** ([LabelSelector](#))

NamespaceSelector decides whether to run the admission control policy on an object based on whether the namespace for that object matches the selector. If the object itself is a namespace, the matching is performed on object.metadata.labels. If the object is another cluster scoped resource, it never skips the policy.

For example, to run the webhook on any objects whose namespace is not associated with "runlevel" of "0" or "1"; you will set the selector as follows: "namespaceSelector": { "matchExpressions": [{ "key": "runlevel", "operator": "NotIn", "values": ["0", "1"] }] }

If instead you want to only run the policy on any objects whose namespace is associated with the "environment" of "prod" or "staging"; you will set the selector as follows: "namespaceSelector": { "matchExpressions": [{ "key": "environment", "operator": "In", "values": ["prod", "staging"] }] }

See <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/> for more examples of label selectors.

Default to the empty LabelSelector, which matches everything.

- **spec.matchResources.objectSelector** ([LabelSelector](#))

ObjectSelector decides whether to run the validation based on if the object has matching labels. objectSelector is evaluated against both the oldObject and newObject that would be sent to the cel validation, and is considered to match if either object matches the selector. A null object (oldObject in the case of create, or newObject in the case of delete) or an object that cannot have labels (like a DeploymentRollback or a PodProxyOptions object) is not considered to match. Use the object selector only if the webhook is opt-in, because end users may skip the admission webhook by setting the labels. Default to the empty LabelSelector, which matches everything.

- **spec.matchResources.resourceRules** ([]NamedRuleWithOperations)

Atomic: will be replaced during a merge

ResourceRules describes what operations on what resources/subresources the ValidatingAdmissionPolicy matches. The policy cares about an operation if it matches *any* Rule.

NamedRuleWithOperations is a tuple of Operations and Resources with ResourceNames.

- **spec.matchResources.resourceRules.apiGroups** ([]string)

Atomic: will be replaced during a merge

APIGroups is the API groups the resources belong to. "`"` is all groups. If "`*`" is present, the length of the slice must be one. Required.

- **spec.matchResources.resourceRules.apiVersions** ([]string)

Atomic: will be replaced during a merge

APIVersions is the API versions the resources belong to. "`"` is all versions. If "`*`" is present, the length of the slice must be one. Required.

- **spec.matchResources.resourceRules.operations** ([]string)

Atomic: will be replaced during a merge

Operations is the operations the admission hook cares about - CREATE, UPDATE, DELETE, CONNECT or `*` for all of those operations and any future admission operations that are added. If `'*'` is present, the length of the slice must be one. Required.

- **spec.matchResources.resourceRules.resourceNames** ([]string)

Atomic: will be replaced during a merge

ResourceNames is an optional white list of names that the rule applies to. An empty set means that everything is allowed.

spec.matchResources.resourceRules.resources ([]string)

Atomic: will be replaced during a merge

Resources is a list of resources this rule applies to.

For example: 'pods' means pods. 'pods/log' means the log subresource of pods. '' means all resources, but not subresources. 'pods/' means all subresources of pods. '/scale' means all scale subresources. '/*' means all resources and their subresources.

If wildcard is present, the validation rule will ensure resources do not overlap with each other.

Depending on the enclosing object, subresources might not be allowed.
Required.

▪ spec.matchResources.resourceRules.scope (string)

scope specifies the scope of this rule. Valid values are "Cluster", "Namespaced", and "" "*Cluster*" means that only cluster-scoped resources will match this rule. Namespace API objects are cluster-scoped. "*Namespaced*" means that only namespaced resources will match this rule. "" means that there are no scope restrictions. Subresources match the scope of their parent resource. Default is "*".

◦ **spec.paramRef (ParamRef)**

paramRef specifies the parameter resource used to configure the admission control policy. It should point to a resource of the type specified in ParamKind of the bound ValidatingAdmissionPolicy. If the policy specifies a ParamKind and the resource referred to by ParamRef does not exist, this binding is considered misconfigured and the FailurePolicy of the ValidatingAdmissionPolicy applied. If the policy does not specify a ParamKind then this field is ignored, and the rules are evaluated without a param.

ParamRef describes how to locate the params to be used as input to expressions of rules applied by a policy binding.

▪ spec.paramRef.name (string)

name is the name of the resource being referenced.

One of name or selector must be set, but name and selector are mutually exclusive properties. If one is set, the other must be unset.

A single parameter used for all admission requests can be configured by setting the name field, leaving selector blank, and setting namespace if paramKind is namespace-scoped.

▪ spec.paramRef.namespace (string)

namespace is the namespace of the referenced resource. Allows limiting the search for params to a specific namespace. Applies to both name and selector fields.

A per-namespace parameter may be used by specifying a namespace-scoped paramKind in the policy and leaving this field empty.

- If paramKind is cluster-scoped, this field MUST be unset. Setting this field results in a configuration error.
- If paramKind is namespace-scoped, the namespace of the object being evaluated for admission will be used when this field is left unset. Take care that if this is left empty the binding must not match any cluster-scoped resources, which will result in an error.

- **spec.paramRef.parameterNotFoundAction** (string)

parameterNotFoundAction controls the behavior of the binding when the resource exists, and name or selector is valid, but there are no parameters matched by the binding. If the value is set to Allow, then no matched parameters will be treated as successful validation by the binding. If set to Deny, then no matched parameters will be subject to the failurePolicy of the policy.

Allowed values are Allow or Deny

Required

- **spec.paramRef.selector** ([LabelSelector](#))

selector can be used to match multiple param objects based on their labels. Supply selector: {} to match all resources of the ParamKind.

If multiple params are found, they are all evaluated with the policy expressions and the results are ANDed together.

One of name or selector must be set, but name and selector are mutually exclusive properties. If one is set, the other must be unset.

- **spec.policyName** (string)

PolicyName references a ValidatingAdmissionPolicy name which the ValidatingAdmissionPolicyBinding binds to. If the referenced resource does not exist, this binding is considered invalid and will be ignored Required.

- **spec.validationActions** ([]string)

Set: unique values will be kept during a merge

validationActions declares how Validations of the referenced ValidatingAdmissionPolicy are enforced. If a validation evaluates to false it is always enforced according to these actions.

Failures defined by the ValidatingAdmissionPolicy's FailurePolicy are enforced according to these actions only if the FailurePolicy is set to Fail, otherwise the failures are ignored. This includes compilation errors, runtime errors and misconfigurations of the policy.

`validationActions` is declared as a set of action values. Order does not matter.
`validationActions` may not contain duplicates of the same action.

The supported actions values are:

"Deny" specifies that a validation failure results in a denied request.

"Warn" specifies that a validation failure is reported to the request client in HTTP Warning headers, with a warning code of 299. Warnings can be sent both for allowed or denied admission responses.

"Audit" specifies that a validation failure is included in the published audit event for the request. The audit event will contain a `validation.policy.admission.k8s.io/validation_failure` audit annotation with a value containing the details of the validation failures, formatted as a JSON list of objects, each with the following fields:

- `message`: The validation failure message string
- `policy`: The resource name of the `ValidatingAdmissionPolicy`
- `binding`: The resource name of the `ValidatingAdmissionPolicyBinding`
- `expressionIndex`: The index of the failed validations in the `ValidatingAdmissionPolicy`
- `validationActions`: The enforcement actions enacted for the validation failure

Example audit annotation:

```
"validation.policy.admission.k8s.io/validation_failure": "[{"message": "Invalid value", "policy": "policy.example.com", "binding": "policybinding.example.com", "expressionIndex": "1", "validationActions": ["Audit"]}]"
```

Clients should expect to handle additional values by ignoring any values not recognized.

"Deny" and "Warn" may not be used together since this combination needlessly duplicates the validation failure both in the API response body and the HTTP warning headers.

Required.

Operations

get read the specified ValidatingAdmissionPolicy

HTTP Request

GET /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies/{name}

Parameters

- **name** (*in path*): string, required

name of the `ValidatingAdmissionPolicy`

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ValidatingAdmissionPolicy](#)): OK

401: Unauthorized

get read status of the specified ValidatingAdmissionPolicy

HTTP Request

GET /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the ValidatingAdmissionPolicy
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ValidatingAdmissionPolicy](#)): OK

401: Unauthorized

list list or watch objects of kind ValidatingAdmissionPolicy

HTTP Request

GET /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies

Parameters

- **allowWatchBookmarks** (*in query*): boolean
 - [allowWatchBookmarks](#)
- **continue** (*in query*): string
 - [continue](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)

- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([ValidatingAdmissionPolicyList](#)): OK

401: Unauthorized

create create a ValidatingAdmissionPolicy

HTTP Request

POST /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies

Parameters

- **body**: [ValidatingAdmissionPolicy](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ValidatingAdmissionPolicy](#)): OK

201 ([ValidatingAdmissionPolicy](#)): Created

202 ([ValidatingAdmissionPolicy](#)): Accepted

401: Unauthorized

update replace the specified ValidatingAdmissionPolicy

HTTP Request

PUT /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies/{name}

Parameters

- **name** (*in path*): string, required

name of the ValidatingAdmissionPolicy

- **body**: [ValidatingAdmissionPolicy](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ValidatingAdmissionPolicy](#)): OK

201 ([ValidatingAdmissionPolicy](#)): Created

401: Unauthorized

update replace status of the specified ValidatingAdmissionPolicy

HTTP Request

PUT /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies/{name}/status

Parameters

- **name** (*in path*): string, required
name of the ValidatingAdmissionPolicy
- **body**: [ValidatingAdmissionPolicy](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([ValidatingAdmissionPolicy](#)): OK

201 ([ValidatingAdmissionPolicy](#)): Created

401: Unauthorized

patch partially update the specified ValidatingAdmissionPolicy

HTTP Request

PATCH /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies/{name}

Parameters

- **name** (*in path*): string, required
name of the ValidatingAdmissionPolicy
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ValidatingAdmissionPolicy](#)): OK

201 ([ValidatingAdmissionPolicy](#)): Created

401: Unauthorized

patch partially update status of the specified ValidatingAdmissionPolicy

HTTP Request

PATCH /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies/{name}/status

Parameters

- **name** (*in path*): string, required

name of the ValidatingAdmissionPolicy

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ValidatingAdmissionPolicy](#)): OK

201 ([ValidatingAdmissionPolicy](#)): Created

401: Unauthorized

delete delete a ValidatingAdmissionPolicy

HTTP Request

DELETE /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies/{name}

Parameters

- **name** (*in path*): string, required

name of the ValidatingAdmissionPolicy

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ValidatingAdmissionPolicy

HTTP Request

DELETE /apis/admissionregistration.k8s.io/v1beta1/validatingadmissionpolicies

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Cluster Resources

[Node](#)

Node is a worker node in Kubernetes.

[Namespace](#)

Namespace provides a scope for Names.

[Event](#)

Event is a report of an event somewhere in the cluster.

[APIService](#)

APIService represents a server for a particular GroupVersion.

[Lease](#)

Lease defines a lease concept.

[RuntimeClass](#)

RuntimeClass defines a class of container runtime supported in the cluster.

[FlowSchema v1beta3](#)

FlowSchema defines the schema of a group of flows.

[PriorityLevelConfiguration v1beta3](#)

PriorityLevelConfiguration represents the configuration of a priority level.

[Binding](#)

Binding ties one object to another; for example, a pod is bound to a node by a scheduler.

[ComponentStatus](#)

ComponentStatus (and ComponentStatusList) holds the cluster validation info.

[ClusterCIDR v1alpha1](#)

ClusterCIDR represents a single configuration for per-Node Pod CIDR allocations when the MultiCIDRRRangeAllocator is enabled (see the config for kube-controller-manager).

Node

Node is a worker node in Kubernetes.

```
apiVersion: v1
```

```
import "k8s.io/api/core/v1"
```

Node

Node is a worker node in Kubernetes. Each node will have a unique identifier in the cache (i.e. in etcd).

- **apiVersion:** v1

- **kind:** Node

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([NodeSpec](#))

Spec defines the behavior of a node. <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([NodeStatus](#))

Most recently observed status of the node. Populated by the system. Read-only. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

NodeSpec

NodeSpec describes the attributes that a node is created with.

- **configSource** ([NodeConfigSource](#))

Deprecated: Previously used to specify the source of the node's configuration for the DynamicKubeletConfig feature. This feature is removed.

NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

- **configSource.configMap** (ConfigMapNodeConfigSource)

ConfigMap is a reference to a Node's ConfigMap

ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: <https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration>

- **configSource.configMap.kubeletConfigKey** (string), required

KubeletConfigKey declares which key of the referenced ConfigMap corresponds to the KubeletConfiguration structure. This field is required in all cases.

- **configSource.configMap.name** (string), required

Name is the metadata.name of the referenced ConfigMap. This field is required in all cases.

- **configSource.configMap.namespace** (string), required

Namespace is the metadata.namespace of the referenced ConfigMap. This field is required in all cases.

- **configSource.configMap.resourceVersion** (string)

ResourceVersion is the metadata.ResourceVersion of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

- **configSource.configMap.uid** (string)

UID is the metadata.UID of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

- **externalID** (string)

Deprecated. Not all kubelets will set this field. Remove field after 1.13. see: <https://issues.k8s.io/61966>

- **podCIDR** (string)

PodCIDR represents the pod IP range assigned to the node.

- **podCIDRs** ([]string)

podCIDRs represents the IP ranges assigned to the node for usage by Pods on that node. If this field is specified, the 0th entry must match the podCIDR field. It may contain at most 1 value for each of IPv4 and IPv6.

providerID (string)

- ID of the node assigned by the cloud provider in the format: <ProviderName>://<ProviderSpecificNodeID>

• **taints** ([]Taint)

If specified, the node's taints.

The node this Taint is attached to has the "effect" on any pod that does not tolerate the Taint.

- **taints.effect** (string), required

Required. The effect of the taint on pods that do not tolerate the taint. Valid effects are NoSchedule, PreferNoSchedule and NoExecute.

- **taints.key** (string), required

Required. The taint key to be applied to a node.

- **taints.timeAdded** (Time)

TimeAdded represents the time at which the taint was added. It is only written for NoExecute taints.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **taints.value** (string)

The taint value corresponding to the taint key.

• **unschedulable** (boolean)

Unschedulable controls node schedulability of new pods. By default, node is schedulable. More info: <https://kubernetes.io/docs/concepts/nodes/node/#manual-node-administration>

NodeStatus

NodeStatus is information about the current status of a node.

• **addresses** ([]NodeAddress)

Patch strategy: merge on key type

List of addresses reachable to the node. Queried from cloud provider, if available. More info: <https://kubernetes.io/docs/concepts/nodes/node/#addresses> Note: This field is declared as mergeable, but the merge key is not sufficiently unique, which can cause data corruption when it is merged. Callers should instead use a full-replacement patch. See <https://pr.k8s.io/79391> for an example. Consumers should assume that addresses can change during the lifetime of a Node. However, there are some exceptions where this

may not be possible, such as Pods that inherit a Node's address in its own status or consumers of the downward API (status.hostIP).

NodeAddress contains information for the node's address.

- **addresses.address** (string), required

The node address.

- **addresses.type** (string), required

Node address type, one of Hostname, ExternalIP or InternalIP.

- **allocatable** (map[string][Quantity](#))

Allocatable represents the resources of a node that are available for scheduling. Defaults to Capacity.

- **capacity** (map[string][Quantity](#))

Capacity represents the total resources of a node. More info: <https://kubernetes.io/docs/concepts/storage/persistent-volumes#capacity>

- **conditions** ([]NodeCondition)

Patch strategy: merge on key type

Conditions is an array of current observed node conditions. More info: <https://kubernetes.io/docs/concepts/nodes/node/#condition>

NodeCondition contains condition information for a node.

- **conditions.status** (string), required

Status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

Type of node condition.

- **conditions.lastHeartbeatTime** (Time)

Last time we got an update on a given condition.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.lastTransitionTime** (Time)

Last time the condition transit from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- conditions.message** (string)
 - Human readable message indicating details about last transition.
- **conditions.reason** (string)
 - (brief) reason for the condition's last transition.
- **config** (NodeConfigStatus)
 - Status of the config assigned to the node via the dynamic Kubelet config feature.

NodeConfigStatus describes the status of the config assigned by Node.Spec.ConfigSource.

 - **config.active** (NodeConfigSource)
 - Active reports the checkpointed config the node is actively using. Active will represent either the current version of the Assigned config, or the current LastKnownGood config, depending on whether attempting to use the Assigned config results in an error.

NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

 - **config.active.configMap** (ConfigMapNodeConfigSource)
 - ConfigMap is a reference to a Node's ConfigMap

ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: <https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration>

 - **config.active.configMap.kubeletConfigKey** (string), required
 - KubeletConfigKey declares which key of the referenced ConfigMap corresponds to the KubeletConfiguration structure. This field is required in all cases.
 - **config.active.configMap.name** (string), required
 - Name is the metadata.name of the referenced ConfigMap. This field is required in all cases.
 - **config.active.configMap.namespace** (string), required
 - Namespace is the metadata.namespace of the referenced ConfigMap. This field is required in all cases.
 - **config.active.configMap.resourceVersion** (string)
 - ResourceVersion is the metadata.ResourceVersion of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

config.active.configMap.uid (string)

UID is the metadata.UID of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

◦ **config.assigned** (NodeConfigSource)

Assigned reports the checkpointed config the node will try to use. When Node.Spec.ConfigSource is updated, the node checkpoints the associated config payload to local disk, along with a record indicating intended config. The node refers to this record to choose its config checkpoint, and reports this record in Assigned. Assigned only updates in the status after the record has been checkpointed to disk. When the Kubelet is restarted, it tries to make the Assigned config the Active config by loading and validating the checkpointed payload identified by Assigned.

NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

▪ **config.assigned.configMap** (ConfigMapNodeConfigSource)

ConfigMap is a reference to a Node's ConfigMap

ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: <https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration>

▪ **config.assigned.configMap.kubeletConfigKey** (string), required

KubeletConfigKey declares which key of the referenced ConfigMap corresponds to the KubeletConfiguration structure. This field is required in all cases.

▪ **config.assigned.configMap.name** (string), required

Name is the metadata.name of the referenced ConfigMap. This field is required in all cases.

▪ **config.assigned.configMap.namespace** (string), required

Namespace is the metadata.namespace of the referenced ConfigMap. This field is required in all cases.

▪ **config.assigned.configMap.resourceVersion** (string)

ResourceVersion is the metadata.ResourceVersion of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

▪ **config.assigned.configMap.uid** (string)

UID is the metadata.UID of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

config.error (string)

- Error describes any problems reconciling the Spec.ConfigSource to the Active config. Errors may occur, for example, attempting to checkpoint Spec.ConfigSource to the local Assigned record, attempting to checkpoint the payload associated with Spec.ConfigSource, attempting to load or validate the Assigned config, etc. Errors may occur at different points while syncing config. Earlier errors (e.g. download or checkpointing errors) will not result in a rollback to LastKnownGood, and may resolve across Kubelet retries. Later errors (e.g. loading or validating a checkpointed config) will result in a rollback to LastKnownGood. In the latter case, it is usually possible to resolve the error by fixing the config assigned in Spec.ConfigSource. You can find additional information for debugging by searching the error message in the Kubelet log. Error is a human-readable description of the error state; machines can check whether or not Error is empty, but should not rely on the stability of the Error text across Kubelet versions.

◦ **config.lastKnownGood** (NodeConfigSource)

LastKnownGood reports the checkpointed config the node will fall back to when it encounters an error attempting to use the Assigned config. The Assigned config becomes the LastKnownGood config when the node determines that the Assigned config is stable and correct. This is currently implemented as a 10-minute soak period starting when the local record of Assigned config is updated. If the Assigned config is Active at the end of this period, it becomes the LastKnownGood. Note that if Spec.ConfigSource is reset to nil (use local defaults), the LastKnownGood is also immediately reset to nil, because the local default config is always assumed good. You should not make assumptions about the node's method of determining config stability and correctness, as this may change or become configurable in the future.

NodeConfigSource specifies a source of node configuration. Exactly one subfield (excluding metadata) must be non-nil. This API is deprecated since 1.22

▪ **config.lastKnownGood.configMap** (ConfigMapNodeConfigSource)

ConfigMap is a reference to a Node's ConfigMap

ConfigMapNodeConfigSource contains the information to reference a ConfigMap as a config source for the Node. This API is deprecated since 1.22: <https://git.k8s.io/enhancements/keps/sig-node/281-dynamic-kubelet-configuration>

▪ **config.lastKnownGood.configMap.kubeletConfigKey** (string), required

KubeletConfigKey declares which key of the referenced ConfigMap corresponds to the KubeletConfiguration structure. This field is required in all cases.

▪ **config.lastKnownGood.configMap.name** (string), required

Name is the metadata.name of the referenced ConfigMap. This field is required in all cases.

▪ **config.lastKnownGood.configMap.namespace** (string), required

Namespace is the metadata.namespace of the referenced ConfigMap. This field is required in all cases.

- **config.lastKnownGood.configMap.resourceVersion** (string)

ResourceVersion is the metadata.ResourceVersion of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

- **config.lastKnownGood.configMap.uid** (string)

UID is the metadata.UID of the referenced ConfigMap. This field is forbidden in Node.Spec, and required in Node.Status.

- **daemonEndpoints** (NodeDaemonEndpoints)

Endpoints of daemons running on the Node.

NodeDaemonEndpoints lists ports opened by daemons running on the Node.

- **daemonEndpoints.kubeletEndpoint** (DaemonEndpoint)

Endpoint on which Kubelet is listening.

DaemonEndpoint contains information about a single Daemon endpoint.

- **daemonEndpoints.kubeletEndpoint.Port** (int32), required

Port number of the given endpoint.

- **images** ([]ContainerImage)

List of container images on this node

Describe a container image

- **images.names** ([]string)

Names by which this image is known. e.g. ["kubernetes.example/hyperkube:v1.0.7", "cloud-vendor.registry.example/cloud-vendor/hyperkube:v1.0.7"]

- **images.sizeBytes** (int64)

The size of the image in bytes.

- **nodeInfo** (NodeSystemInfo)

Set of ids/uuids to uniquely identify the node. More info: <https://kubernetes.io/docs/concepts/nodes/node/#info>

NodeSystemInfo is a set of ids/uuids to uniquely identify the node.

- **nodeInfo.architecture** (string), required

The Architecture reported by the node

- **nodeInfo.bootID** (string), required
 - Boot ID reported by the node.
- **nodeInfo.containerRuntimeVersion** (string), required

ContainerRuntime Version reported by the node through runtime remote API (e.g. containerd://1.4.2).
- **nodeInfo.kernelVersion** (string), required

Kernel Version reported by the node from 'uname -r' (e.g. 3.16.0-0.bpo.4-amd64).
- **nodeInfo.kubeProxyVersion** (string), required

KubeProxy Version reported by the node.
- **nodeInfo.kubeletVersion** (string), required

Kubelet Version reported by the node.
- **nodeInfo.machineID** (string), required

MachineID reported by the node. For unique machine identification in the cluster this field is preferred. Learn more from man(5) machine-id: <http://man7.org/linux/man-pages/man5/machine-id.5.html>
- **nodeInfo.operatingSystem** (string), required

The Operating System reported by the node
- **nodeInfo.osImage** (string), required

OS Image reported by the node from /etc/os-release (e.g. Debian GNU/Linux 7 (wheezy)).
- **nodeInfo.systemUUID** (string), required

SystemUUID reported by the node. For unique machine identification MachineID is preferred. This field is specific to Red Hat hosts https://access.redhat.com/documentation/en-us/red_hat_subscription_management/1/html/rhsm/uuid
- **phase** (string)

NodePhase is the recently observed lifecycle phase of the node. More info: <https://kubernetes.io/docs/concepts/nodes/node/#phase> The field is never populated, and now is deprecated.
- **volumesAttached** ([]AttachedVolume)

List of volumes that are attached to the node.
AttachedVolume describes a volume attached to a node

 - **volumesAttached.devicePath** (string), required

DevicePath represents the device path where the volume should be available

- **volumesAttached.name** (string), required

Name of the attached volume

- **volumesInUse** ([]string)

List of attachable volumes in use (mounted) by the node.

NodeList

NodeList is the whole list of all Nodes which have been registered with master.

- **apiVersion**: v1

- **kind**: NodeList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][Node](#)), required

List of nodes

Operations

get read the specified Node

HTTP Request

GET /api/v1/nodes/{name}

Parameters

- **name** (*in path*): string, required

name of the Node

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Node](#)): OK

401: Unauthorized

get read status of the specified Node

HTTP Request

GET /api/v1/nodes/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Node

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Node](#)): OK

401: Unauthorized

list list or watch objects of kind Node

HTTP Request

GET /api/v1/nodes

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([NodeList](#)): OK

401: Unauthorized

create create a Node

HTTP Request

POST /api/v1/nodes

Parameters

- **body**: [Node](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Node](#)): OK

201 ([Node](#)): Created

202 ([Node](#)): Accepted

401: Unauthorized

update replace the specified Node

HTTP Request

PUT /api/v1/nodes/{name}

Parameters

- **name** (*in path*): string, required

name of the Node

- **body**: [Node](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Node](#)): OK

201 ([Node](#)): Created

401: Unauthorized

update replace status of the specified Node

HTTP Request

PUT /api/v1/nodes/{name}/status

Parameters

- **name** (*in path*): string, required
name of the Node
- **body**: [Node](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Node](#)): OK

201 ([Node](#)): Created

401: Unauthorized

patch partially update the specified Node

HTTP Request

PATCH /api/v1/nodes/{name}

Parameters

- **name** (*in path*): string, required
name of the Node
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)

- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Node](#)): OK

201 ([Node](#)): Created

401: Unauthorized

patch partially update status of the specified Node

HTTP Request

PATCH /api/v1/nodes/{name}/status

Parameters

- **name** (*in path*): string, required
 - name of the Node
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **force** (*in query*): boolean
 - [force](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Node](#)): OK

201 ([Node](#)): Created

401: Unauthorized

delete delete a Node

HTTP Request

DELETE /api/v1/nodes/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Node
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Node

HTTP Request

DELETE /api/v1/nodes

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

Response

200 ([Status](#)): OK

Namespace

Namespace provides a scope for Names.

```
apiVersion: v1
```

```
import "k8s.io/api/core/v1"
```

Namespace

Namespace provides a scope for Names. Use of multiple namespaces is optional.

- **apiVersion:** v1
- **kind:** Namespace
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([NamespaceSpec](#))

Spec defines the behavior of the Namespace. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([NamespaceStatus](#))

Status describes the current status of a Namespace. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

NamespaceSpec

NamespaceSpec describes the attributes on a Namespace.

- **finalizers** ([]string)

Finalizers is an opaque list of values that must be empty to permanently remove object from storage. More info: <https://kubernetes.io/docs/tasks/administer-cluster/namespaces/>

NamespaceStatus

NamespaceStatus is information about the current status of a Namespace.

- **conditions** ([]NamespaceCondition)

Patch strategy: merge on key type

Represents the latest available observations of a namespace's current state.

NamespaceCondition contains details about state of namespace.

- **conditions.status** (string), required

Status of the condition, one of True, False, Unknown.

- **conditions.type** (string), required

Type of namespace controller condition.

- **conditions.lastTransitionTime** (Time)

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

- **conditions.reason** (string)

- **phase** (string)

Phase is the current lifecycle phase of the namespace. More info: <https://kubernetes.io/docs/tasks/administer-cluster/namespaces/>

NamespaceList

NamespaceList is a list of Namespaces.

- **apiVersion**: v1

- **kind**: NamespaceList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **items** ([][Namespace](#)), required

Items is the list of Namespace objects in the list. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>

Operations

get read the specified Namespace

HTTP Request

GET /api/v1/namespaces/{name}

Parameters

- **name** (*in path*): string, required
name of the Namespace
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Namespace](#)): OK

401: Unauthorized

get read status of the specified Namespace

HTTP Request

GET /api/v1/namespaces/{name}/status

Parameters

- **name** (*in path*): string, required
name of the Namespace
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Namespace](#)): OK

401: Unauthorized

list list or watch objects of kind Namespace

HTTP Request

GET /api/v1/namespaces

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([NamespaceList](#)): OK

401: Unauthorized

create create a Namespace

HTTP Request

POST /api/v1/namespaces

Parameters

- **body**: [Namespace](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Namespace](#)): OK

201 ([Namespace](#)): Created

202 ([Namespace](#)): Accepted

401: Unauthorized

update replace the specified Namespace

HTTP Request

PUT /api/v1/namespaces/{name}

Parameters

- **name** (*in path*): string, required
name of the Namespace
- **body**: [Namespace](#), required
- **dryRun** (*in query*): string
[dryRun](#)

- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Namespace](#)): OK

201 ([Namespace](#)): Created

401: Unauthorized

update replace finalize of the specified Namespace

HTTP Request

PUT /api/v1/namespaces/{name}/finalize

Parameters

- **name** (*in path*): string, required
name of the Namespace
- **body**: [Namespace](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Namespace](#)): OK

201 ([Namespace](#)): Created

401: Unauthorized

update replace status of the specified Namespace

HTTP Request

PUT /api/v1/namespaces/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Namespace

- **body**: [Namespace](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Namespace](#)): OK

201 ([Namespace](#)): Created

401: Unauthorized

patch partially update the specified Namespace

HTTP Request

PATCH /api/v1/namespaces/{name}

Parameters

- **name** (*in path*): string, required

name of the Namespace

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Namespace](#)): OK

201 ([Namespace](#)): Created

401: Unauthorized

patch partially update status of the specified Namespace

HTTP Request

PATCH /api/v1/namespaces/{name}/status

Parameters

- **name** (*in path*): string, required

name of the Namespace

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

fieldValidation (*in query*): string

- [fieldValidation](#)

• **force** (*in query*): boolean

[force](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([Namespace](#)): OK

201 ([Namespace](#)): Created

401: Unauthorized

delete delete a Namespace

HTTP Request

DELETE /api/v1/namespaces/{name}

Parameters

• **name** (*in path*): string, required

name of the Namespace

• **body**: [DeleteOptions](#)

• **dryRun** (*in query*): string

[dryRun](#)

• **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

• **pretty** (*in query*): string

[pretty](#)

• **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

Event

Event is a report of an event somewhere in the cluster.

```
apiVersion: events.k8s.io/v1
```

```
import "k8s.io/api/events/v1"
```

Event

Event is a report of an event somewhere in the cluster. It generally denotes some state change in the system. Events have a limited retention time and triggers and messages may evolve with time. Event consumers should not rely on the timing of an event with a given Reason reflecting a consistent underlying trigger, or the continued existence of events with that Reason. Events should be treated as informative, best-effort, supplemental data.

- **apiVersion:** events.k8s.io/v1

- **kind:** Event

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **eventTime** (MicroTime), required

eventTime is the time when this Event was first observed. It is required.

MicroTime is version of Time with microsecond level precision.

- **action** (string)

action is what action was taken/failed regarding to the regarding object. It is machine-readable. This field cannot be empty for new Events and it can have at most 128 characters.

- **deprecatedCount** (int32)

deprecatedCount is the deprecated field assuring backward compatibility with core.v1 Event type.

- **deprecatedFirstTimestamp** (Time)

deprecatedFirstTimestamp is the deprecated field assuring backward compatibility with core.v1 Event type.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **deprecatedLastTimestamp** (Time)

deprecatedLastTimestamp is the deprecated field assuring backward compatibility with core.v1 Event type.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **deprecatedSource** (EventSource)

deprecatedSource is the deprecated field assuring backward compatibility with core.v1 Event type.

EventSource contains information for an event.

- **deprecatedSource.component** (string)

Component from which the event is generated.

- **deprecatedSource.host** (string)

Node name on which the event is generated.

- **note** (string)

note is a human-readable description of the status of this operation. Maximal length of the note is 1kB, but libraries should be prepared to handle values up to 64kB.

- **reason** (string)

reason is why the action was taken. It is human-readable. This field cannot be empty for new Events and it can have at most 128 characters.

- **regarding** ([ObjectReference](#))

regarding contains the object this Event is about. In most cases it's an Object reporting controller implements, e.g. ReplicaSetController implements ReplicaSets and this event is emitted because it acts on some changes in a ReplicaSet object.

- **related** ([ObjectReference](#))

related is the optional secondary object for more complex actions. E.g. when regarding object triggers a creation or deletion of related object.

- **reportingController** (string)

reportingController is the name of the controller that emitted this Event, e.g. kubernetes.io/kubelet. This field cannot be empty for new Events.

- **reportingInstance** (string)

reportingInstance is the ID of the controller instance, e.g. kubelet-xyzf. This field cannot be empty for new Events and it can have at most 128 characters.

series (EventSeries)

- series is data about the Event series this event represents or nil if it's a singleton Event.

EventSeries contain information on series of events, i.e. thing that was/is happening continuously for some time. How often to update the EventSeries is up to the event reporters. The default event reporter in "k8s.io/client-go/tools/events/event_broadcaster.go" shows how this struct is updated on heartbeats and can guide customized reporter implementations.

- **series.count** (int32), required

count is the number of occurrences in this series up to the last heartbeat time.

- **series.lastObservedTime** (MicroTime), required

lastObservedTime is the time when last Event from the series was seen before last heartbeat.

MicroTime is version of Time with microsecond level precision.

- **type** (string)

type is the type of this event (Normal, Warning), new types could be added in the future. It is machine-readable. This field cannot be empty for new Events.

EventList

EventList is a list of Event objects.

- **apiVersion**: events.k8s.io/v1

- **kind**: EventList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][Event](#)), required

items is a list of schema objects.

Operations

get read the specified Event

HTTP Request

GET /apis/events.k8s.io/v1/namespaces/{namespace}/events/{name}

Parameters

- **name** (*in path*): string, required
name of the Event
- **namespace** (*in path*): string, required
[namespace](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Event](#)): OK

401: Unauthorized

list list or watch objects of kind Event

HTTP Request

GET /apis/events.k8s.io/v1/namespaces/{namespace}/events

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)

[pretty](#)

- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([EventList](#)): OK

401: Unauthorized

list list or watch objects of kind Event

HTTP Request

GET /apis/events.k8s.io/v1/events

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([EventList](#)): OK

401: Unauthorized

create create an Event

HTTP Request

POST /apis/events.k8s.io/v1/namespaces/{namespace}/events

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Event](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

fieldValidation (*in query*): string

- [fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([Event](#)): OK

201 ([Event](#)): Created

202 ([Event](#)): Accepted

401: Unauthorized

update replace the specified Event

HTTP Request

PUT /apis/events.k8s.io/v1/namespaces/{namespace}/events/{name}

Parameters

• **name** (*in path*): string, required

name of the Event

• **namespace** (*in path*): string, required

[namespace](#)

• **body**: [Event](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([Event](#)): OK

201 ([Event](#)): Created

401: Unauthorized

patch partially update the specified Event

HTTP Request

PATCH /apis/events.k8s.io/v1/namespaces/{namespace}/events/{name}

Parameters

- **name** (*in path*): string, required

name of the Event

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Event](#)): OK

201 ([Event](#)): Created

401: Unauthorized

delete delete an Event

HTTP Request

DELETE /apis/events.k8s.io/v1/namespaces/{namespace}/events/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Event
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Event

HTTP Request

DELETE /apis/events.k8s.io/v1/namespaces/{namespace}/events

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
 - [labelSelector](#)
- **limit** (*in query*): integer
 - [limit](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)
- **resourceVersion** (*in query*): string
 - [resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
 - [resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
 - [sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
 - [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

APIService

APIService represents a server for a particular GroupVersion.

```
apiVersion: apiregistration.k8s.io/v1  
import "k8s.io/kube-aggregator/pkg/apis/apiregistration/v1"
```

APIService

APIService represents a server for a particular GroupVersion. Name must be "version.group".

- **apiVersion:** apiregistration.k8s.io/v1
- **kind:** APIService
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([APIServiceSpec](#))

Spec contains information for locating and communicating with a server

- **status** ([APIServiceStatus](#))

Status contains derived information about an API server

APIServiceSpec

APIServiceSpec contains information for locating and communicating with a server. Only https is supported, though you are able to disable certificate verification.

- **groupPriorityMinimum** (int32), required

GroupPriorityMinimum is the priority this group should have at least. Higher priority means that the group is preferred by clients over lower priority ones. Note that other versions of this group might specify even higher GroupPriorityMinimum values such that the whole group gets a higher priority. The primary sort is based on GroupPriorityMinimum, ordered highest number to lowest (20 before 10). The secondary sort is based on the alphabetical comparison of the name of the object. (v1.bar before v1.foo) We'd recommend something like: *.k8s.io (except extensions) at 18000 and PaaSes (OpenShift, Deis) are recommended to be in the 2000s

- **versionPriority** (int32), required

VersionPriority controls the ordering of this API version inside of its group. Must be greater than zero. The primary sort is based on VersionPriority, ordered highest to lowest (20 before 10). Since it's inside of a group, the number can be small, probably in the 10s.

In case of equal version priorities, the version string will be used to compute the order inside a group. If the version string is "kube-like", it will sort above non "kube-like" version strings, which are ordered lexicographically. "Kube-like" versions start with a "v", then are followed by a number (the major version), then optionally the string "alpha" or "beta" and another number (the minor version). These are sorted first by GA > beta > alpha (where GA is a version with no suffix such as beta or alpha), and then by comparing major version, then minor version. An example sorted list of versions: v10, v2, v1, v11beta2, v10beta3, v3beta1, v12alpha1, v11alpha2, foo1, foo10.

- **caBundle** ([]byte)

Atomic: will be replaced during a merge

CABundle is a PEM encoded CA bundle which will be used to validate an API server's serving certificate. If unspecified, system trust roots on the apiserver are used.

- **group** (string)

Group is the API group name this server hosts

- **insecureSkipTLSVerify** (boolean)

InsecureSkipTLSVerify disables TLS certificate verification when communicating with this server. This is strongly discouraged. You should use the CABundle instead.

- **service** (ServiceReference)

Service is a reference to the service for this API server. It must communicate on port 443. If the Service is nil, that means the handling for the API groupversion is handled locally on this server. The call will simply delegate to the normal handler chain to be fulfilled.

ServiceReference holds a reference to Service.legacy.k8s.io

- **service.name** (string)

Name is the name of the service

- **service.namespace** (string)

Namespace is the namespace of the service

- **service.port** (int32)

If specified, the port on the service that hosting webhook. Default to 443 for backward compatibility. port should be a valid port number (1-65535, inclusive).

- **version** (string)

Version is the API version this server hosts. For example, "v1"

APIServiceStatus

APIServiceStatus contains derived information about an API server

-
- **conditions** ([]APIServiceCondition)

Patch strategy: merge on key type

Map: unique values on key type will be kept during a merge

Current service state of apiService.

APIServiceCondition describes the state of an APIService at a particular point

- **conditions.status** (string), required

Status is the status of the condition. Can be True, False, Unknown.

- **conditions.type** (string), required

Type is the type of the condition.

- **conditions.lastTransitionTime** (Time)

Last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

Human-readable message indicating details about last transition.

- **conditions.reason** (string)

Unique, one-word, CamelCase reason for the condition's last transition.

APIServiceList

APIServiceList is a list of APIService objects.

- **apiVersion**: apiregistration.k8s.io/v1

- **kind**: APIServiceList

- **metadata** ([ListMeta](#))

Standard list metadata More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([]APIService), required

Items is the list of APIService

Operations

get read the specified APIService

HTTP Request

GET /apis/apiregistration.k8s.io/v1/apiservices/{name}

Parameters

- **name** (*in path*): string, required
name of the APIService
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([APIService](#)): OK

401: Unauthorized

get read status of the specified APIService

HTTP Request

GET /apis/apiregistration.k8s.io/v1/apiservices/{name}/status

Parameters

- **name** (*in path*): string, required
name of the APIService
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([APIService](#)): OK

401: Unauthorized

list list or watch objects of kind APIService

HTTP Request

GET /apis/apiregistration.k8s.io/v1/apiservices

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([APIServiceList](#)): OK

401: Unauthorized

create create an APIService

HTTP Request

POST /apis/apiregistration.k8s.io/v1/apiservices

Parameters

- **body**: [APIService](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([APIService](#)): OK

201 ([APIService](#)): Created

202 ([APIService](#)): Accepted

401: Unauthorized

update replace the specified APIService

HTTP Request

PUT /apis/apiregistration.k8s.io/v1/apiservices/{name}

Parameters

- **name** (*in path*): string, required

name of the APIService

- **body**: [APIService](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([APIService](#)): OK

201 ([APIService](#)): Created

401: Unauthorized

update replace status of the specified APIService

HTTP Request

PUT /apis/apiregistration.k8s.io/v1/apiservices/{name}/status

Parameters

- **name** (*in path*): string, required

name of the APIService

- **body**: [APIService](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([APIService](#)): OK

201 ([APIService](#)): Created

401: Unauthorized

patch partially update the specified APIService

HTTP Request

PATCH /apis/apiregistration.k8s.io/v1/apiservices/{name}

Parameters

- **name** (*in path*): string, required

- name of the APIService

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

- [dryRun](#)

- **fieldManager** (*in query*): string

- [fieldManager](#)

- **fieldValidation** (*in query*): string

- [fieldValidation](#)

- **force** (*in query*): boolean

- [force](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([APIService](#)): OK

201 ([APIService](#)): Created

401: Unauthorized

patch partially update status of the specified APIService

HTTP Request

PATCH /apis/apiregistration.k8s.io/v1/apiservices/{name}/status

Parameters

- **name** (*in path*): string, required
name of the APIService
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **force** (*in query*): boolean
[force](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([APIService](#)): OK

201 ([APIService](#)): Created

401: Unauthorized

delete delete an APIService

HTTP Request

DELETE /apis/apiregistration.k8s.io/v1/apiservices/{name}

Parameters

- **name** (*in path*): string, required
name of the APIService

- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **pretty** (*in query*): string
 - [pretty](#)
- **propagationPolicy** (*in query*): string
 - [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of APIService

HTTP Request

DELETE /apis/apiregistration.k8s.io/v1/apiservices

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
 - [continue](#)
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldSelector** (*in query*): string
 - [fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
 - [gracePeriodSeconds](#)
- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Lease

Lease defines a lease concept.

apiVersion: coordination.k8s.io/v1

import "k8s.io/api/coordination/v1"

Lease

Lease defines a lease concept.

-
- **apiVersion**: coordination.k8s.io/v1

- **kind**: Lease

metadata ([ObjectMeta](#))

- More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

• spec ([LeaseSpec](#))

spec contains the specification of the Lease. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

LeaseSpec

LeaseSpec is a specification of a Lease.

• acquireTime (MicroTime)

acquireTime is a time when the current lease was acquired.

MicroTime is version of Time with microsecond level precision.

• holderIdentity (string)

holderIdentity contains the identity of the holder of a current lease.

• leaseDurationSeconds (int32)

leaseDurationSeconds is a duration that candidates for a lease need to wait to force acquire it. This is measure against time of last observed renewTime.

• leaseTransitions (int32)

leaseTransitions is the number of transitions of a lease between holders.

• renewTime (MicroTime)

renewTime is a time when the current holder of a lease has last updated the lease.

MicroTime is version of Time with microsecond level precision.

LeaseList

LeaseList is a list of Lease objects.

• apiVersion: coordination.k8s.io/v1

• kind: LeaseList

• metadata ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][Lease](#)), required
 - items is a list of schema objects.

Operations

get read the specified Lease

HTTP Request

GET /apis/coordination.k8s.io/v1/namespaces/{namespace}/leases/{name}

Parameters

- **name** (*in path*): string, required
 - name of the Lease
- **namespace** (*in path*): string, required
 - [namespace](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Lease](#)): OK

401: Unauthorized

list list or watch objects of kind Lease

HTTP Request

GET /apis/coordination.k8s.io/v1/namespaces/{namespace}/leases

Parameters

- **namespace** (*in path*): string, required
 - [namespace](#)
- **allowWatchBookmarks** (*in query*): boolean
 - [allowWatchBookmarks](#)
- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([LeaseList](#)): OK

401: Unauthorized

list list or watch objects of kind Lease

HTTP Request

GET /apis/coordination.k8s.io/v1/leases

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([LeaseList](#)): OK

401: Unauthorized

create create a Lease

HTTP Request

POST /apis/coordination.k8s.io/v1/namespaces/{namespace}/leases

Parameters

- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Lease](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([Lease](#)): OK

201 ([Lease](#)): Created

202 ([Lease](#)): Accepted

401: Unauthorized

update replace the specified Lease

HTTP Request

PUT /apis/coordination.k8s.io/v1/namespaces/{namespace}/leases/{name}

Parameters

- **name** (*in path*): string, required
name of the Lease
- **namespace** (*in path*): string, required
[namespace](#)
- **body**: [Lease](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Lease](#)): OK

201 ([Lease](#)): Created

401: Unauthorized

patch partially update the specified Lease

HTTP Request

PATCH /apis/coordination.k8s.io/v1/namespaces/{namespace}/leases/{name}

Parameters

- **name** (*in path*): string, required

name of the Lease

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([Lease](#)): OK

201 ([Lease](#)): Created

401: Unauthorized

delete delete a Lease

HTTP Request

DELETE /apis/coordination.k8s.io/v1/namespaces/{namespace}/leases/{name}

Parameters

- **name** (*in path*): string, required

name of the Lease

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of Lease

HTTP Request

DELETE /apis/coordination.k8s.io/v1/namespaces/{namespace}/leases

Parameters

- **namespace** (*in path*): string, required

[namespace](#)

- **body**: [DeleteOptions](#)

- **continue** (*in query*): string

[continue](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

sendInitialEvents (*in query*): boolean

-

[sendInitialEvents](#)

• **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

RuntimeClass

RuntimeClass defines a class of container runtime supported in the cluster.

apiVersion: node.k8s.io/v1

```
import "k8s.io/api/node/v1"
```

RuntimeClass

RuntimeClass defines a class of container runtime supported in the cluster. The RuntimeClass is used to determine which container runtime is used to run all containers in a pod.

RuntimeClasses are manually defined by a user or cluster provisioner, and referenced in the PodSpec. The Kubelet is responsible for resolving the RuntimeClassName reference before running the pod. For more details, see <https://kubernetes.io/docs/concepts/containers/runtime-class/>

- **apiVersion**: node.k8s.io/v1

- **kind**: RuntimeClass

- **metadata** ([ObjectMeta](#))

More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **handler** (string), required

handler specifies the underlying runtime and configuration that the CRI implementation will use to handle pods of this class. The possible values are specific to the node & CRI configuration. It is assumed that all handlers are available on every node, and handlers of the same name are equivalent on every node. For example, a handler called "runc" might specify that the runc OCI runtime (using native Linux containers) will be used to run the containers in a pod. The Handler must be lowercase, conform to the DNS Label (RFC 1123) requirements, and is immutable.

- **overhead** (Overhead)

overhead represents the resource overhead associated with running a pod for a given RuntimeClass. For more details, see <https://kubernetes.io/docs/concepts/scheduling-eviction/pod-overhead/>

Overhead structure represents the resource overhead associated with running a pod.

- **overhead.podFixed** (map[string][Quantity](#))

podFixed represents the fixed resource overhead associated with running a pod.

- **scheduling** (Scheduling)

scheduling holds the scheduling constraints to ensure that pods running with this RuntimeClass are scheduled to nodes that support it. If scheduling is nil, this RuntimeClass is assumed to be supported by all nodes.

Scheduling specifies the scheduling constraints for nodes supporting a RuntimeClass.

- **scheduling.nodeSelector** (map[string]string)

nodeSelector lists labels that must be present on nodes that support this RuntimeClass. Pods using this RuntimeClass can only be scheduled to a node matched by this selector. The RuntimeClass nodeSelector is merged with a pod's existing nodeSelector. Any conflicts will cause the pod to be rejected in admission.

- **scheduling.tolerations** ([]Toleration)

Atomic: will be replaced during a merge

tolerations are appended (excluding duplicates) to pods running with this RuntimeClass during admission, effectively unioning the set of nodes tolerated by the pod and the RuntimeClass.

The pod this Toleration is attached to tolerates any taint that matches the triple <key,value,effect> using the matching operator .

- **scheduling.tolerations.key** (string)

Key is the taint key that the toleration applies to. Empty means match all taint keys. If the key is empty, operator must be Exists; this combination means to match all values and all keys.

- **scheduling.tolerations.operator** (string)

Operator represents a key's relationship to the value. Valid operators are Exists and Equal. Defaults to Equal. Exists is equivalent to wildcard for value, so that a pod can tolerate all taints of a particular category.

- **scheduling.tolerations.value** (string)

Value is the taint value the toleration matches to. If the operator is Exists, the value should be empty, otherwise just a regular string.

- **scheduling.tolerations.effect** (string)

Effect indicates the taint effect to match. Empty means match all taint effects. When specified, allowed values are NoSchedule, PreferNoSchedule and NoExecute.

- **scheduling.tolerations.tolerationSeconds** (int64)

TolerationSeconds represents the period of time the toleration (which must be of effect NoExecute, otherwise this field is ignored) tolerates the taint. By default, it is not set, which means tolerate the taint forever (do not evict). Zero and negative values will be treated as 0 (evict immediately) by the system.

RuntimeClassList

RuntimeClassList is a list of RuntimeClass objects.

- **apiVersion**: node.k8s.io/v1

- **kind**: RuntimeClassList

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][RuntimeClass](#)), required

items is a list of schema objects.

Operations

get read the specified RuntimeClass

HTTP Request

GET /apis/node.k8s.io/v1/runtimeclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the RuntimeClass

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([RuntimeClass](#)): OK

401: Unauthorized

list list or watch objects of kind RuntimeClass

HTTP Request

GET /apis/node.k8s.io/v1/runtimeclasses

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([RuntimeClassList](#)): OK

401: Unauthorized

create create a RuntimeClass

HTTP Request

POST /apis/node.k8s.io/v1/runtimeclasses

Parameters

- **body**: [RuntimeClass](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([RuntimeClass](#)): OK

201 ([RuntimeClass](#)): Created

202 ([RuntimeClass](#)): Accepted

401: Unauthorized

update replace the specified RuntimeClass

HTTP Request

PUT /apis/node.k8s.io/v1/runtimeclasses/{name}

Parameters

- **name** (*in path*): string, required

name of the RuntimeClass

- **body**: [RuntimeClass](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([RuntimeClass](#)): OK

201 ([RuntimeClass](#)): Created

401: Unauthorized

patch partially update the specified RuntimeClass

HTTP Request

PATCH /apis/node.k8s.io/v1/runtimeclasses/{name}

Parameters

- **name** (*in path*): string, required
name of the RuntimeClass
- **body**: [Patch](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)

- **force** (*in query*): boolean

- [force](#)

- **pretty** (*in query*): string

- [pretty](#)

Response

200 ([RuntimeClass](#)): OK

201 ([RuntimeClass](#)): Created

401: Unauthorized

delete delete a RuntimeClass

HTTP Request

DELETE /apis/node.k8s.io/v1/runtimeclasses/{name}

Parameters

- **name** (*in path*): string, required

- name of the RuntimeClass

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

- [dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

- [gracePeriodSeconds](#)

- **pretty** (*in query*): string

- [pretty](#)

- **propagationPolicy** (*in query*): string

- [propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of RuntimeClass

HTTP Request

DELETE /apis/node.k8s.io/v1/runtimeclasses

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

FlowSchema v1beta3

FlowSchema defines the schema of a group of flows.

apiVersion: flowcontrol.apiserver.k8s.io/v1beta3

import "k8s.io/api/flowcontrol/v1beta3"

FlowSchema

FlowSchema defines the schema of a group of flows. Note that a flow is made up of a set of inbound API requests with similar attributes and is identified by a pair of strings: the name of the FlowSchema and a "flow distinguisher".

- **apiVersion:** flowcontrol.apiserver.k8s.io/v1beta3

- **kind:** FlowSchema

- **metadata** ([ObjectMeta](#))

metadata is the standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([FlowSchemaSpec](#))

spec is the specification of the desired behavior of a FlowSchema. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([FlowSchemaStatus](#))

status is the current status of a FlowSchema. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

FlowSchemaSpec

FlowSchemaSpec describes how the FlowSchema's specification looks like.

- **priorityLevelConfiguration** (PriorityLevelConfigurationReference), required

priorityLevelConfiguration should reference a PriorityLevelConfiguration in the cluster. If the reference cannot be resolved, the FlowSchema will be ignored and marked as invalid in its status. Required.

PriorityLevelConfigurationReference contains information that points to the "request-priority" being used.

- **priorityLevelConfiguration.name** (string), required

name is the name of the priority level configuration being referenced Required.

- **distinguisherMethod** (FlowDistinguisherMethod)

distinguisherMethod defines how to compute the flow distinguisher for requests that match this schema. nil specifies that the distinguisher is disabled and thus will always be the empty string.

FlowDistinguisherMethod specifies the method of a flow distinguisher.

- **distinguisherMethod.type** (string), required

type is the type of flow distinguisher method The supported types are "ByUser" and "ByNamespace". Required.

- **matchingPrecedence** (int32)

matchingPrecedence is used to choose among the FlowSchemas that match a given request. The chosen FlowSchema is among those with the numerically lowest (which we take to be logically highest) MatchingPrecedence. Each MatchingPrecedence value must be ranged in [1,10000]. Note that if the precedence is not specified, it will be set to 1000 as default.

- **rules** (>[]PolicyRulesWithSubjects)

Atomic: will be replaced during a merge

rules describes which requests will match this flow schema. This FlowSchema matches a request if and only if at least one member of rules matches the request. if it is an empty slice, there will be no requests matching the FlowSchema.

PolicyRulesWithSubjects prescribes a test that applies to a request to an apiserver. The test considers the subject making the request, the verb being requested, and the resource to be acted upon. This PolicyRulesWithSubjects matches a request if and only if both (a) at least one member of subjects matches the request and (b) at least one member of resourceRules or nonResourceRules matches the request.

- **rules.subjects** ([]Subject), required

Atomic: will be replaced during a merge

subjects is the list of normal user, serviceaccount, or group that this rule cares about. There must be at least one member in this slice. A slice that includes both the system:authenticated and system:unauthenticated user groups matches every request. Required.

Subject matches the originator of a request, as identified by the request authentication system. There are three ways of matching an originator; by user, group, or service account.

- **rules.subjects.kind** (string), required

kind indicates which one of the other fields is non-empty. Required

- **rules.subjects.group** (GroupSubject)

group matches based on user group name.

GroupSubject holds detailed information for group-kind subject.

- **rules.subjects.group.name** (string), required

name is the user group that matches, or "*" to match all user groups.

See <https://github.com/kubernetes/apiserver/blob/master/pkg/authentication/user/user.go> for some well-known group names.

Required.

- **rules.subjects.serviceAccount** (ServiceAccountSubject)

serviceAccount matches ServiceAccounts.

ServiceAccountSubject holds detailed information for service-account-kind subject.

- **rules.subjects.serviceAccount.name** (string), required

name is the name of matching ServiceAccount objects, or "*" to match regardless of name. Required.

- **rules.subjects.serviceAccount.namespace** (string), required

namespace is the namespace of matching ServiceAccount objects. Required.

- **rules.subjects.user** (UserSubject)

user matches based on username.

UserSubject holds detailed information for user-kind subject.

- **rules.subjects.user.name** (string), required

name is the username that matches, or "*" to match all usernames. Required.

- **rules.nonResourceRules** ([]NonResourcePolicyRule)

Atomic: will be replaced during a merge

nonResourceRules is a list of NonResourcePolicyRules that identify matching requests according to their verb and the target non-resource URL.

NonResourcePolicyRule is a predicate that matches non-resource requests according to their verb and the target non-resource URL. A *NonResourcePolicyRule* matches a request if and only if both (a) at least one member of verbs matches the request and (b) at least one member of nonResourceURLs matches the request.

- **rules.nonResourceRules.nonResourceURLs** ([]string), required

Set: unique values will be kept during a merge

nonResourceURLs is a set of url prefixes that a user should have access to and may not be empty. For example:

- "/healthz" is legal
- "/hea*" is illegal
- "/hea" is legal but matches nothing
- "/hea/*" also matches nothing
- "/healthz/" matches all per-component health checks. "" matches all non-resource urls. if it is present, it must be the only entry. Required.

- **rules.nonResourceRules.verbs** ([]string), required

Set: unique values will be kept during a merge

verbs is a list of matching verbs and may not be empty. "*" matches all verbs. If it is present, it must be the only entry. Required.

- **rules.resourceRules** ([]ResourcePolicyRule)

Atomic: will be replaced during a merge

resourceRules is a slice of ResourcePolicyRules that identify matching requests according to their verb and the target resource. At least one of resourceRules and nonResourceRules has to be non-empty.

ResourcePolicyRule is a predicate that matches some resource requests, testing the request's verb and the target resource. A *ResourcePolicyRule* matches a resource request if and only if: (a) at least one member of verbs matches the request, (b) at least one member of apiGroups matches the request, (c) at least one member of resources matches the request, and (d) either (d1) the request does not specify a namespace (i.e., Namespace=="") and clusterScope is true or (d2) the request specifies a namespace and least one member of namespaces matches the request's namespace.

- **rules.resourceRules.apiGroups** ([]string), required

Set: unique values will be kept during a merge

apiGroups is a list of matching API groups and may not be empty. "*" matches all API groups and, if present, must be the only entry. Required.

- **rules.resourceRules.resources** ([]string), required

Set: unique values will be kept during a merge

resources is a list of matching resources (i.e., lowercase and plural) with, if desired, subresource. For example, ["services", "nodes/status"]. This list may

not be empty. "*" matches all resources and, if present, must be the only entry. Required.

- **rules.resourceRules.verbs** ([]string), required

Set: unique values will be kept during a merge

verbs is a list of matching verbs and may not be empty. "*" matches all verbs and, if present, must be the only entry. Required.

- **rules.resourceRules.clusterScope** (boolean)

clusterScope indicates whether to match requests that do not specify a namespace (which happens either because the resource is not namespaced or the request targets all namespaces). If this field is omitted or false then the namespaces field must contain a non-empty list.

- **rules.resourceRules.namespaces** ([]string)

Set: unique values will be kept during a merge

namespaces is a list of target namespaces that restricts matches. A request that specifies a target namespace matches only if either (a) this list contains that target namespace or (b) this list contains "". Note that "" matches any specified namespace but does not match a request that *does not specify* a namespace (see the clusterScope field for that). This list may be empty, but only if clusterScope is true.

FlowSchemaStatus

FlowSchemaStatus represents the current state of a FlowSchema.

- **conditions** ([]FlowSchemaCondition)

Patch strategy: merge on key type

Map: unique values on key type will be kept during a merge

conditions is a list of the current states of FlowSchema.

FlowSchemaCondition describes conditions for a FlowSchema.

- **conditions.lastTransitionTime** (Time)

lastTransitionTime is the last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **conditions.message** (string)

message is a human-readable message indicating details about last transition.

- **conditions.reason** (string)

reason is a unique, one-word, CamelCase reason for the condition's last transition.

- **conditions.status** (string)

status is the status of the condition. Can be True, False, Unknown. Required.

- **conditions.type** (string)

type is the type of the condition. Required.

FlowSchemaList

FlowSchemaList is a list of FlowSchema objects.

- **apiVersion**: flowcontrol.apiserver.k8s.io/v1beta3

- **kind**: FlowSchemaList

- **metadata** ([ListMeta](#))

metadata is the standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][FlowSchema](#)), required

items is a list of FlowSchemas.

Operations

get read the specified FlowSchema

HTTP Request

GET /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas/{name}

Parameters

- **name** (*in path*): string, required

name of the FlowSchema

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([FlowSchema](#)): OK

401: Unauthorized

get read status of the specified FlowSchema

HTTP Request

GET /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas/{name}/status

Parameters

- **name** (*in path*): string, required

name of the FlowSchema

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([FlowSchema](#)): OK

401: Unauthorized

list list or watch objects of kind FlowSchema

HTTP Request

GET /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([FlowSchemaList](#)): OK

401: Unauthorized

create create a FlowSchema

HTTP Request

POST /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas

Parameters

- **body**: [FlowSchema](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([FlowSchema](#)): OK

201 ([FlowSchema](#)): Created

202 ([FlowSchema](#)): Accepted

401: Unauthorized

update replace the specified FlowSchema

HTTP Request

PUT /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas/{name}

Parameters

- **name** (*in path*): string, required
name of the FlowSchema
- **body**: [FlowSchema](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([FlowSchema](#)): OK

201 ([FlowSchema](#)): Created

401: Unauthorized

update replace status of the specified FlowSchema

HTTP Request

PUT /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas/{name}/status

Parameters

- **name** (*in path*): string, required
name of the FlowSchema
- **body**: [FlowSchema](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([FlowSchema](#)): OK

201 ([FlowSchema](#)): Created

401: Unauthorized

patch partially update the specified FlowSchema

HTTP Request

PATCH /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas/{name}

Parameters

- **name** (*in path*): string, required
name of the FlowSchema
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([FlowSchema](#)): OK

201 ([FlowSchema](#)): Created

401: Unauthorized

patch partially update status of the specified FlowSchema

HTTP Request

PATCH /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas/{name}/status

Parameters

- **name** (*in path*): string, required

name of the FlowSchema

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([FlowSchema](#)): OK

201 ([FlowSchema](#)): Created

401: Unauthorized

delete delete a FlowSchema

HTTP Request

DELETE /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas/{name}

Parameters

- **name** (*in path*): string, required

name of the FlowSchema

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of FlowSchema

HTTP Request

DELETE /apis/flowcontrol.apiserver.k8s.io/v1beta3/flowschemas

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

PriorityLevelConfiguration v1beta3

PriorityLevelConfiguration represents the configuration of a priority level.

apiVersion: flowcontrol.apiserver.k8s.io/v1beta3

import "k8s.io/api/flowcontrol/v1beta3"

PriorityLevelConfiguration

PriorityLevelConfiguration represents the configuration of a priority level.

- **apiVersion**: flowcontrol.apiserver.k8s.io/v1beta3
- **kind**: PriorityLevelConfiguration
- **metadata** ([ObjectMeta](#))

metadata is the standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([PriorityLevelConfigurationSpec](#))

spec is the specification of the desired behavior of a "request-priority". More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

- **status** ([PriorityLevelConfigurationStatus](#))

status is the current status of a "request-priority". More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

PriorityLevelConfigurationSpec

PriorityLevelConfigurationSpec specifies the configuration of a priority level.

- **type** (string), required

type indicates whether this priority level is subject to limitation on request execution. A value of "Exempt" means that requests of this priority level are not subject to a limit (and thus are never queued) and do not detract from the capacity made available to other

priority levels. A value of "Limited" means that (a) requests of this priority level *are* subject to limits and (b) some of the server's limited capacity is made available exclusively to this priority level. Required.

- **exempt** (ExemptPriorityLevelConfiguration)

exempt specifies how requests are handled for an exempt priority level. This field MUST be empty if type is "Limited". This field MAY be non-empty if type is "Exempt". If empty and type is "Exempt" then the default values for ExemptPriorityLevelConfiguration apply.

ExemptPriorityLevelConfiguration describes the configurable aspects of the handling of exempt requests. In the mandatory exempt configuration object the values in the fields here can be modified by authorized users, unlike the rest of the spec.

- **exempt.lendablePercent** (int32)

lendablePercent prescribes the fraction of the level's NominalCL that can be borrowed by other priority levels. This value of this field must be between 0 and 100, inclusive, and it defaults to 0. The number of seats that other levels can borrow from this level, known as this level's LendableConcurrencyLimit (LendableCL), is defined as follows.

$$\text{LendableCL}(i) = \text{round}(\text{NominalCL}(i) * \text{lendablePercent}(i)/100.0)$$

- **exempt.nominalConcurrencyShares** (int32)

nominalConcurrencyShares (NCS) contributes to the computation of the NominalConcurrencyLimit (NominalCL) of this level. This is the number of execution seats nominally reserved for this priority level. This DOES NOT limit the dispatching from this priority level but affects the other priority levels through the borrowing mechanism. The server's concurrency limit (ServerCL) is divided among all the priority levels in proportion to their NCS values:

$$\text{NominalCL}(i) = \text{ceil}(\text{ServerCL} * \text{NCS}(i) / \text{sum_ncs}) \quad \text{sum_ncs} = \text{sum}[\text{priority level } k] \text{NCS}(k)$$

Bigger numbers mean a larger nominal concurrency limit, at the expense of every other priority level. This field has a default value of zero.

- **limited** (LimitedPriorityLevelConfiguration)

limited specifies how requests are handled for a Limited priority level. This field must be non-empty if and only if type is "Limited".

*LimitedPriorityLevelConfiguration specifies how to handle requests that are subject to limits. It addresses two issues:

- How are requests for this priority level limited?
- What should be done with requests that exceed the limit?*
- **limited.borrowingLimitPercent** (int32)

borrowingLimitPercent, if present, configures a limit on how many seats this priority level can borrow from other priority levels. The limit is known as this

level's BorrowingConcurrencyLimit (BorrowingCL) and is a limit on the total number of seats that this level may borrow at any one time. This field holds the ratio of that limit to the level's nominal concurrency limit. When this field is non-nil, it must hold a non-negative integer and the limit is calculated as follows.

$$\text{BorrowingCL}(i) = \text{round}(\text{NominalCL}(i) * \text{borrowingLimitPercent}(i)/100.0)$$

The value of this field can be more than 100, implying that this priority level can borrow a number of seats that is greater than its own nominal concurrency limit (NominalCL). When this field is left nil, the limit is effectively infinite.

- **limited.lendablePercent** (int32)

lendablePercent prescribes the fraction of the level's NominalCL that can be borrowed by other priority levels. The value of this field must be between 0 and 100, inclusive, and it defaults to 0. The number of seats that other levels can borrow from this level, known as this level's LendableConcurrencyLimit (LendableCL), is defined as follows.

$$\text{LendableCL}(i) = \text{round}(\text{NominalCL}(i) * \text{lendablePercent}(i)/100.0)$$

- **limited.limitResponse** (LimitResponse)

limitResponse indicates what to do with requests that can not be executed right now

LimitResponse defines how to handle requests that can not be executed right now.

- **limited.limitResponse.type** (string), required

type is "Queue" or "Reject". "Queue" means that requests that can not be executed upon arrival are held in a queue until they can be executed or a queuing limit is reached. "Reject" means that requests that can not be executed upon arrival are rejected. Required.

- **limited.limitResponse.queuing** (QueuingConfiguration)

queuing holds the configuration parameters for queuing. This field may be non-empty only if type is "Queue".

QueuingConfiguration holds the configuration parameters for queuing

- **limited.limitResponse.queuing.handSize** (int32)

handSize is a small positive number that configures the shuffle sharding of requests into queues. When enqueueing a request at this priority level the request's flow identifier (a string pair) is hashed and the hash value is used to shuffle the list of queues and deal a hand of the size specified here. The request is put into one of the shortest queues in that hand. handSize must be no larger than queues, and should be significantly smaller (so that a few heavy flows do not saturate most of the queues). See the user-facing documentation for more extensive guidance on setting this field. This field has a default value of 8.

limited.limitResponse.queuing.queueLengthLimit (int32)

queueLengthLimit is the maximum number of requests allowed to be waiting in a given queue of this priority level at a time; excess requests are rejected. This value must be positive. If not specified, it will be defaulted to 50.

▪ **limited.limitResponse.queuing.queues** (int32)

queues is the number of queues for this priority level. The queues exist independently at each apiserver. The value must be positive. Setting it to 1 effectively precludes shufflesharding and thus makes the distinguisher method of associated flow schemas irrelevant. This field has a default value of 64.

◦ **limited.nominalConcurrencyShares** (int32)

nominalConcurrencyShares (NCS) contributes to the computation of the NominalConcurrencyLimit (NominalCL) of this level. This is the number of execution seats available at this priority level. This is used both for requests dispatched from this priority level as well as requests dispatched from other priority levels borrowing seats from this level. The server's concurrency limit (ServerCL) is divided among the Limited priority levels in proportion to their NCS values:

$$\text{NominalCL}(i) = \text{ceil}(\text{ServerCL} * \text{NCS}(i) / \text{sum_ncs})$$
$$\text{sum_ncs} = \text{sum}[\text{priority level } k] \text{NCS}(k)$$

Bigger numbers mean a larger nominal concurrency limit, at the expense of every other priority level. This field has a default value of 30.

PriorityLevelConfigurationStatus

PriorityLevelConfigurationStatus represents the current state of a "request-priority".

• **conditions** ([]PriorityLevelConfigurationCondition)

Patch strategy: merge on key type

Map: unique values on key type will be kept during a merge

conditions is the current state of "request-priority".

PriorityLevelConfigurationCondition defines the condition of priority level.

◦ **conditions.lastTransitionTime** (Time)

lastTransitionTime is the last time the condition transitioned from one status to another.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- conditions.message** (string)
 - message is a human-readable message indicating details about last transition.
- **conditions.reason** (string)
 - reason is a unique, one-word, CamelCase reason for the condition's last transition.
- **conditions.status** (string)
 - status is the status of the condition. Can be True, False, Unknown. Required.
- **conditions.type** (string)
 - type is the type of the condition. Required.

PriorityLevelConfigurationList

PriorityLevelConfigurationList is a list of PriorityLevelConfiguration objects.

- **apiVersion**: flowcontrol.apiserver.k8s.io/v1beta3
- **kind**: PriorityLevelConfigurationList
- **metadata** ([ListMeta](#))

metadata is the standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **items** ([][PriorityLevelConfiguration](#)), required

items is a list of request-priorities.

Operations

get read the specified PriorityLevelConfiguration

HTTP Request

GET /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations/{name}

Parameters

- **name** (*in path*): string, required
 - name of the PriorityLevelConfiguration
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([PriorityLevelConfiguration](#)): OK

401: Unauthorized

get read status of the specified PriorityLevelConfiguration

HTTP Request

GET /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations/{name}/status

Parameters

- **name** (*in path*): string, required
name of the PriorityLevelConfiguration
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PriorityLevelConfiguration](#)): OK

401: Unauthorized

list list or watch objects of kind PriorityLevelConfiguration

HTTP Request

GET /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations

Parameters

- **allowWatchBookmarks** (*in query*): boolean
[allowWatchBookmarks](#)
- **continue** (*in query*): string
[continue](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **labelSelector** (*in query*): string
[labelSelector](#)

- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer
[timeoutSeconds](#)
- **watch** (*in query*): boolean
[watch](#)

Response

200 ([PriorityLevelConfigurationList](#)): OK

401: Unauthorized

create create a PriorityLevelConfiguration

HTTP Request

POST /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations

Parameters

- **body**: [PriorityLevelConfiguration](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityLevelConfiguration](#)): OK

201 ([PriorityLevelConfiguration](#)): Created

202 ([PriorityLevelConfiguration](#)): Accepted

401: Unauthorized

update replace the specified PriorityLevelConfiguration

HTTP Request

PUT /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations/{name}

Parameters

- **name** (*in path*): string, required

name of the PriorityLevelConfiguration

- **body**: [PriorityLevelConfiguration](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityLevelConfiguration](#)): OK

201 ([PriorityLevelConfiguration](#)): Created

401: Unauthorized

update replace status of the specified PriorityLevelConfiguration

HTTP Request

PUT /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations/{name}/status

Parameters

- **name** (*in path*): string, required
name of the PriorityLevelConfiguration
- **body**: [PriorityLevelConfiguration](#), required
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldManager** (*in query*): string
[fieldManager](#)
- **fieldValidation** (*in query*): string
[fieldValidation](#)
- **pretty** (*in query*): string
[pretty](#)

Response

200 ([PriorityLevelConfiguration](#)): OK

201 ([PriorityLevelConfiguration](#)): Created

401: Unauthorized

patch partially update the specified PriorityLevelConfiguration

HTTP Request

PATCH /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations/{name}

Parameters

- **name** (*in path*): string, required
name of the PriorityLevelConfiguration
- **body**: [Patch](#), required
- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityLevelConfiguration](#)): OK

201 ([PriorityLevelConfiguration](#)): Created

401: Unauthorized

patch partially update status of the specified PriorityLevelConfiguration

HTTP Request

PATCH /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations/{name}/status

Parameters

- **name** (*in path*): string, required

name of the PriorityLevelConfiguration

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([PriorityLevelConfiguration](#)): OK

201 ([PriorityLevelConfiguration](#)): Created

401: Unauthorized

delete delete a PriorityLevelConfiguration

HTTP Request

DELETE /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations/{name}

Parameters

- **name** (*in path*): string, required

name of the PriorityLevelConfiguration

- **body**: [DeleteOptions](#)

- **dryRun** (*in query*): string

[dryRun](#)

- **gracePeriodSeconds** (*in query*): integer

[gracePeriodSeconds](#)

- **pretty** (*in query*): string

[pretty](#)

- **propagationPolicy** (*in query*): string

[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of PriorityLevelConfiguration

HTTP Request

DELETE /apis/flowcontrol.apiserver.k8s.io/v1beta3/prioritylevelconfigurations

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **labelSelector** (*in query*): string
[labelSelector](#)
- **limit** (*in query*): integer
[limit](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)
- **resourceVersion** (*in query*): string
[resourceVersion](#)
- **resourceVersionMatch** (*in query*): string
[resourceVersionMatch](#)
- **sendInitialEvents** (*in query*): boolean
[sendInitialEvents](#)
- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Binding

Binding ties one object to another; for example, a pod is bound to a node by a scheduler.

apiVersion: v1

```
import "k8s.io/api/core/v1"
```

Binding

Binding ties one object to another; for example, a pod is bound to a node by a scheduler.
Deprecated in 1.7, please use the bindings subresource of pods instead.

- **apiVersion:** v1
- **kind:** Binding
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **target** ([ObjectReference](#)), required

The target object that you want to bind to the standard object.

Operations

create create a Binding

HTTP Request

POST /api/v1/namespaces/{namespace}/bindings

Parameters

- **namespace** (*in path*): string, required
[namespace](#)

- **body**: [Binding](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)
- **fieldValidation** (*in query*): string
 - [fieldValidation](#)
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([Binding](#)): OK

201 ([Binding](#)): Created

202 ([Binding](#)): Accepted

401: Unauthorized

create create binding of a Pod

HTTP Request

POST /api/v1/namespaces/{namespace}/pods/{name}/binding

Parameters

- **name** (*in path*): string, required
 - name of the Binding
- **namespace** (*in path*): string, required
 - [namespace](#)
- **body**: [Binding](#), required
- **dryRun** (*in query*): string
 - [dryRun](#)
- **fieldManager** (*in query*): string
 - [fieldManager](#)

- **fieldValidation** (*in query*): string
- [fieldValidation](#)
- **pretty** (*in query*): string
- [pretty](#)

Response

200 ([Binding](#)): OK

201 ([Binding](#)): Created

202 ([Binding](#)): Accepted

401: Unauthorized

ComponentStatus

ComponentStatus (and ComponentStatusList) holds the cluster validation info.

apiVersion: v1

import "k8s.io/api/core/v1"

ComponentStatus

ComponentStatus (and ComponentStatusList) holds the cluster validation info. Deprecated: This API is deprecated in v1.19+

- **apiVersion**: v1
- **kind**: ComponentStatus
- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **conditions** ([]ComponentCondition)

Patch strategy: merge on key type

List of component conditions observed

Information about the condition of a component.

- **conditions.status** (string), required

Status of the condition for a component. Valid values for "Healthy": "True", "False", or "Unknown".

- conditions.type** (string), required
 - Type of condition for a component. Valid value: "Healthy"
- **conditions.error** (string)
 - Condition error code for a component. For example, a health check error code.
- **conditions.message** (string)
 - Message about the condition for a component. For example, information about a health check.

ComponentStatusList

Status of all the conditions for the component as a list of ComponentStatus objects. Deprecated: This API is deprecated in v1.19+

- **apiVersion**: v1
- **kind**: ComponentStatusList
- **metadata** ([ListMeta](#))
 - Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>
- **items** ([][ComponentStatus](#)), required
 - List of ComponentStatus objects.

Operations

get read the specified ComponentStatus

HTTP Request

GET /api/v1/componentstatuses/{name}

Parameters

- **name** (*in path*): string, required
 - name of the ComponentStatus
- **pretty** (*in query*): string
 - [pretty](#)

Response

200 ([ComponentStatus](#)): OK

401: Unauthorized

list list objects of kind ComponentStatus

HTTP Request

GET /api/v1/componentstatuses

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

watch (*in query*): boolean

- [watch](#)

Response

200 ([ComponentStatusList](#)): OK

401: Unauthorized

ClusterCIDR v1alpha1

ClusterCIDR represents a single configuration for per-Node Pod CIDR allocations when the MultiCIDRRRangeAllocator is enabled (see the config for kube-controller-manager).

```
apiVersion: networking.k8s.io/v1alpha1
```

```
import "k8s.io/api/networking/v1alpha1"
```

ClusterCIDR

ClusterCIDR represents a single configuration for per-Node Pod CIDR allocations when the MultiCIDRRRangeAllocator is enabled (see the config for kube-controller-manager). A cluster may have any number of ClusterCIDR resources, all of which will be considered when allocating a CIDR for a Node. A ClusterCIDR is eligible to be used for a given Node when the node selector matches the node in question and has free CIDRs to allocate. In case of multiple matching ClusterCIDR resources, the allocator will attempt to break ties using internal heuristics, but any ClusterCIDR whose node selector matches the Node may be used.

-
- **apiVersion:** networking.k8s.io/v1alpha1

- **kind:** ClusterCIDR

- **metadata** ([ObjectMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

- **spec** ([ClusterCIDRSpec](#))

spec is the desired state of the ClusterCIDR. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

ClusterCIDRSpec

ClusterCIDRSpec defines the desired state of ClusterCIDR.

-
- **perNodeHostBits** (int32), required

perNodeHostBits defines the number of host bits to be configured per node. A subnet mask determines how much of the address is used for network bits and host bits. For example an IPv4 address of 192.168.0.0/24, splits the address into 24 bits for the network portion and 8 bits for the host portion. To allocate 256 IPs, set this field to 8 (a /24 mask for IPv4 or a /120 for IPv6). Minimum value is 4 (16 IPs). This field is immutable.

- **ipv4** (string)

ipv4 defines an IPv4 IP block in CIDR notation(e.g. "10.0.0.0/8"). At least one of ipv4 and ipv6 must be specified. This field is immutable.

- **ipv6** (string)

ipv6 defines an IPv6 IP block in CIDR notation(e.g. "2001:db8::/64"). At least one of ipv4 and ipv6 must be specified. This field is immutable.

- **nodeSelector** (NodeSelector)

nodeSelector defines which nodes the config is applicable to. An empty or nil nodeSelector selects all nodes. This field is immutable.

A node selector represents the union of the results of one or more label queries over a set of nodes; that is, it represents the OR of the selectors represented by the node selector terms.

- **nodeSelector.nodeSelectorTerms** ([]NodeSelectorTerm), required

Required. A list of node selector terms. The terms are ORed.

A null or empty node selector term matches no objects. The requirements of them are ANDed. The TopologySelectorTerm type implements a subset of the NodeSelectorTerm.

- **nodeSelector.nodeSelectorTerms.matchExpressions** ([]NodeSelectorRequirement)

A list of node selector requirements by node's labels.

- **nodeSelector.nodeSelectorTerms.matchFields** ([]NodeSelectorRequirement)

A list of node selector requirements by node's fields.

ClusterCIDRList

ClusterCIDRList contains a list of ClusterCIDR.

- **apiVersion**: networking.k8s.io/v1alpha1

- **kind**: ClusterCIDRList

- **metadata** ([ListMeta](#))

Standard object's metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

items ([][ClusterCIDR](#)), required

- items is the list of ClusterCIDRs.

Operations

get read the specified ClusterCIDR

HTTP Request

GET /apis/networking.k8s.io/v1alpha1/clustercidrs/{name}

Parameters

- **name** (*in path*): string, required

name of the ClusterCIDR

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterCIDR](#)): OK

401: Unauthorized

list list or watch objects of kind ClusterCIDR

HTTP Request

GET /apis/networking.k8s.io/v1alpha1/clustercidrs

Parameters

- **allowWatchBookmarks** (*in query*): boolean

[allowWatchBookmarks](#)

- **continue** (*in query*): string

[continue](#)

- **fieldSelector** (*in query*): string

[fieldSelector](#)

- **labelSelector** (*in query*): string

[labelSelector](#)

- **limit** (*in query*): integer

[limit](#)

- **pretty** (*in query*): string

[pretty](#)

- **resourceVersion** (*in query*): string

[resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

[resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

[sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

[timeoutSeconds](#)

- **watch** (*in query*): boolean

[watch](#)

Response

200 ([ClusterCIDRList](#)): OK

401: Unauthorized

create create a ClusterCIDR

HTTP Request

POST /apis/networking.k8s.io/v1alpha1/clustercidrs

Parameters

- **body**: [ClusterCIDR](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

fieldValidation (*in query*): string

- [fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterCIDR](#)): OK

201 ([ClusterCIDR](#)): Created

202 ([ClusterCIDR](#)): Accepted

401: Unauthorized

update replace the specified ClusterCIDR

HTTP Request

PUT /apis/networking.k8s.io/v1alpha1/clustercidrs/{name}

Parameters

• **name** (*in path*): string, required

name of the ClusterCIDR

• **body**: [ClusterCIDR](#), required

• **dryRun** (*in query*): string

[dryRun](#)

• **fieldManager** (*in query*): string

[fieldManager](#)

• **fieldValidation** (*in query*): string

[fieldValidation](#)

• **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterCIDR](#)): OK

201 ([ClusterCIDR](#)): Created

401: Unauthorized

patch partially update the specified ClusterCIDR

HTTP Request

PATCH /apis/networking.k8s.io/v1alpha1/clustercidrs/{name}

Parameters

- **name** (*in path*): string, required

name of the ClusterCIDR

- **body**: [Patch](#), required

- **dryRun** (*in query*): string

[dryRun](#)

- **fieldManager** (*in query*): string

[fieldManager](#)

- **fieldValidation** (*in query*): string

[fieldValidation](#)

- **force** (*in query*): boolean

[force](#)

- **pretty** (*in query*): string

[pretty](#)

Response

200 ([ClusterCIDR](#)): OK

201 ([ClusterCIDR](#)): Created

401: Unauthorized

delete delete a ClusterCIDR

HTTP Request

DELETE /apis/networking.k8s.io/v1alpha1/clustercidrs/{name}

Parameters

- **name** (*in path*): string, required
name of the ClusterCIDR
- **body**: [DeleteOptions](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **gracePeriodSeconds** (*in query*): integer
[gracePeriodSeconds](#)
- **pretty** (*in query*): string
[pretty](#)
- **propagationPolicy** (*in query*): string
[propagationPolicy](#)

Response

200 ([Status](#)): OK

202 ([Status](#)): Accepted

401: Unauthorized

deletecollection delete collection of ClusterCIDR

HTTP Request

DELETE /apis/networking.k8s.io/v1alpha1/clustercidrs

Parameters

- **body**: [DeleteOptions](#)
- **continue** (*in query*): string
[continue](#)
- **dryRun** (*in query*): string
[dryRun](#)
- **fieldSelector** (*in query*): string
[fieldSelector](#)

gracePeriodSeconds (*in query*): integer

- [gracePeriodSeconds](#)

- **labelSelector** (*in query*): string

- [labelSelector](#)

- **limit** (*in query*): integer

- [limit](#)

- **pretty** (*in query*): string

- [pretty](#)

- **propagationPolicy** (*in query*): string

- [propagationPolicy](#)

- **resourceVersion** (*in query*): string

- [resourceVersion](#)

- **resourceVersionMatch** (*in query*): string

- [resourceVersionMatch](#)

- **sendInitialEvents** (*in query*): boolean

- [sendInitialEvents](#)

- **timeoutSeconds** (*in query*): integer

- [timeoutSeconds](#)

Response

200 ([Status](#)): OK

401: Unauthorized

Common Definitions

[DeleteOptions](#)

DeleteOptions may be provided when deleting an API object.

[LabelSelector](#)

A label selector is a label query over a set of resources.

[ListMeta](#)

ListMeta describes metadata that synthetic resources must have, including lists and various status objects.

[LocalObjectReference](#)

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

[NodeSelectorRequirement](#)

A node selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

[ObjectFieldSelector](#)

ObjectFieldSelector selects an APIVersioned field of an object.

[ObjectMeta](#)

ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.

[ObjectReference](#)

ObjectReference contains enough information to let you inspect or modify the referred object.

[Patch](#)

Patch is provided to give a concrete name and type to the Kubernetes PATCH request body.

[Quantity](#)

Quantity is a fixed-point representation of a number.

[ResourceFieldSelector](#)

ResourceFieldSelector represents container resources (cpu, memory) and their output format.

[Status](#)

Status is a return value for calls that don't return other objects.

[TypedLocalObjectReference](#)

TypedLocalObjectReference contains enough information to let you locate the typed referenced object inside the same namespace.

DeleteOptions

DeleteOptions may be provided when deleting an API object.

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
```

DeleteOptions may be provided when deleting an API object.

- **apiVersion** (string)

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>

- **dryRun** ([]string)

When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

- **gracePeriodSeconds** (int64)

The duration in seconds before the object should be deleted. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace period for the specified type will be used. Defaults to a per object value if not specified. zero means delete immediately.

- **kind** (string)

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **orphanDependents** (boolean)

Deprecated: please use the PropagationPolicy, this field will be deprecated in 1.7. Should the dependent objects be orphaned. If true/false, the "orphan" finalizer will be added to/ removed from the object's finalizers list. Either this field or PropagationPolicy may be set, but not both.

- **preconditions** (Preconditions)

Must be fulfilled before a deletion is carried out. If not possible, a 409 Conflict status will be returned.

Preconditions must be fulfilled before an operation (update, delete, etc.) is carried out.

- **preconditions.resourceVersion** (string)

Specifies the target ResourceVersion

- **preconditions.uid** (string)
 - Specifies the target UID.
- **propagationPolicy** (string)

Whether and how garbage collection will be performed. Either this field or OrphanDependents may be set, but not both. The default policy is decided by the existing finalizer set in the metadata.finalizers and the resource-specific default policy. Acceptable values are: 'Orphan' - orphan the dependents; 'Background' - allow the garbage collector to delete the dependents in the background; 'Foreground' - a cascading policy that deletes all dependents in the foreground.

LabelSelector

A label selector is a label query over a set of resources.

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
```

A label selector is a label query over a set of resources. The result of matchLabels and matchExpressions are ANDed. An empty label selector matches all objects. A null label selector matches no objects.

- **matchExpressions** ([]LabelSelectorRequirement)

matchExpressions is a list of label selector requirements. The requirements are ANDed.

A label selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

- **matchExpressions.key** (string), required

key is the label key that the selector applies to.

- **matchExpressions.operator** (string), required

operator represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists and DoesNotExist.

- **matchExpressions.values** ([]string)

values is an array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. This array is replaced during a strategic merge patch.

- **matchLabels** (map[string]string)

matchLabels is a map of {key,value} pairs. A single {key,value} in the matchLabels map is equivalent to an element of matchExpressions, whose key field is "key", the operator is "In", and the values array contains only "value". The requirements are ANDed.

ListMeta

ListMeta describes metadata that synthetic resources must have, including lists and various status objects.

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
```

ListMeta describes metadata that synthetic resources must have, including lists and various status objects. A resource may have only one of {ObjectMeta, ListMeta}.

- **continue** (string)

continue may be set if the user set a limit on the number of items returned, and indicates that the server has more data available. The value is opaque and may be used to issue another request to the endpoint that served this list to retrieve the next set of available objects. Continuing a consistent list may not be possible if the server configuration has changed or more than a few minutes have passed. The resourceVersion field returned when using this continue value will be identical to the value in the first response, unless you have received this token from an error message.

- **remainingItemCount** (int64)

remainingItemCount is the number of subsequent items in the list which are not included in this list response. If the list request contained label or field selectors, then the number of remaining items is unknown and the field will be left unset and omitted during serialization. If the list is complete (either because it is not chunking or because this is the last chunk), then there are no more remaining items and this field will be left unset and omitted during serialization. Servers older than v1.15 do not set this field. The intended use of the remainingItemCount is *estimating* the size of a collection. Clients should not rely on the remainingItemCount to be set or to be exact.

- **resourceVersion** (string)

String that identifies the server's internal version of this object that can be used by clients to determine when objects have changed. Value must be treated as opaque by clients and passed unmodified back to the server. Populated by the system. Read-only. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency>

- **selfLink** (string)

Deprecated: selfLink is a legacy read-only field that is no longer populated by the system.

LocalObjectReference

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
```

LocalObjectReference contains enough information to let you locate the referenced object inside the same namespace.

- **name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

NodeSelectorRequirement

A node selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

```
import "k8s.io/api/core/v1"
```

A node selector requirement is a selector that contains values, a key, and an operator that relates the key and values.

- **key** (string), required

The label key that the selector applies to.

- **operator** (string), required

Represents a key's relationship to a set of values. Valid operators are In, NotIn, Exists, DoesNotExist, Gt, and Lt.

- **values** ([]string)

An array of string values. If the operator is In or NotIn, the values array must be non-empty. If the operator is Exists or DoesNotExist, the values array must be empty. If the operator is Gt or Lt, the values array must have a single element, which will be interpreted as an integer. This array is replaced during a strategic merge patch.

ObjectFieldSelector

ObjectFieldSelector selects an APIVersioned field of an object.

```
import "k8s.io/api/core/v1"
```

ObjectFieldSelector selects an APIVersioned field of an object.

- **fieldPath** (string), required

Path of the field to select in the specified API version.

- **apiVersion** (string)

Version of the schema the FieldPath is written in terms of, defaults to "v1".

ObjectMeta

ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
```

ObjectMeta is metadata that all persisted resources must have, which includes all objects users must create.

- **name** (string)

Name must be unique within a namespace. Is required when creating resources, although some resources may allow a client to request the generation of an appropriate name automatically. Name is primarily intended for creation idempotence and configuration definition. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names#names>

- **generateName** (string)

GenerateName is an optional prefix, used by the server, to generate a unique name ONLY IF the Name field has not been provided. If this field is used, the name returned to the client will be different than the name passed. This value will also be combined with a unique suffix. The provided value has the same validation rules as the Name field, and may be truncated by the length of the suffix required to make the value unique on the server.

If this field is specified and the generated name exists, the server will return a 409.

Applied only if Name is not specified. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#idempotency>

- **namespace** (string)

Namespace defines the space within which each name must be unique. An empty namespace is equivalent to the "default" namespace, but "default" is the canonical representation. Not all objects are required to be scoped to a namespace - the value of this field for those objects will be empty.

Must be a DNS_LABEL. Cannot be updated. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces>

- **labels** (map[string]string)

Map of string keys and values that can be used to organize and categorize (scope and select) objects. May match selectors of replication controllers and services. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/labels>

- **annotations** (map[string]string)

Annotations is an unstructured key value map stored with a resource that may be set by external tools to store and retrieve arbitrary metadata. They are not queryable and should

be preserved when modifying objects. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/annotations>

System

- **finalizers** ([]string)

Must be empty before the object is deleted from the registry. Each entry is an identifier for the responsible component that will remove the entry from the list. If the deletionTimestamp of the object is non-nil, entries in this list can only be removed. Finalizers may be processed and removed in any order. Order is NOT enforced because it introduces significant risk of stuck finalizers. finalizers is a shared field, any actor with permission can reorder it. If the finalizer list is processed in order, then this can lead to a situation in which the component responsible for the first finalizer in the list is waiting for a signal (field value, external system, or other) produced by a component responsible for a finalizer later in the list, resulting in a deadlock. Without enforced ordering finalizers are free to order amongst themselves and are not vulnerable to ordering changes in the list.

- **managedFields** ([]ManagedFieldsEntry)

ManagedFields maps workflow-id and version to the set of fields that are managed by that workflow. This is mostly for internal housekeeping, and users typically shouldn't need to set or understand this field. A workflow can be the user's name, a controller's name, or the name of a specific apply path like "ci-cd". The set of fields is always in the version that the workflow used when modifying the object.

ManagedFieldsEntry is a workflow-id, a FieldSet and the group version of the resource that the fieldset applies to.

- **managedFields.apiVersion** (string)

APIVersion defines the version of this resource that this field set applies to. The format is "group/version" just like the top-level APIVersion field. It is necessary to track the version of a field set because it cannot be automatically converted.

- **managedFields.fieldsType** (string)

FieldsType is the discriminator for the different fields format and version. There is currently only one possible value: "FieldsV1"

- **managedFields.fieldsV1** (FieldsV1)

FieldsV1 holds the first JSON version format as described in the "FieldsV1" type.

*FieldsV1 stores a set of fields in a data structure like a Trie, in JSON format.

Each key is either a '.' representing the field itself, and will always map to an empty set, or a string representing a sub-field or item. The string will follow one of these four formats: 'f:', where is the name of a field in a struct, or key in a map 'v:', where is the exact json formatted value of a list item 'i:', where is position of a item in a list 'k:', where is a map of a list item's key fields to their unique values If a key maps to an empty Fields value, the field that key represents is part of the set.

The exact format is defined in sigs.k8s.io/structured-merge-diff*

- **managedFields.manager** (string)

Manager is an identifier of the workflow managing these fields.

- **managedFields.operation** (string)

Operation is the type of operation which lead to this ManagedFieldsEntry being created. The only valid values for this field are 'Apply' and 'Update'.

- **managedFields.subresource** (string)

Subresource is the name of the subresource used to update that object, or empty string if the object was updated through the main resource. The value of this field is used to distinguish between managers, even if they share the same name. For example, a status update will be distinct from a regular update using the same manager name. Note that the APIVersion field is not related to the Subresource field and it always corresponds to the version of the main resource.

- **managedFields.time** (Time)

Time is the timestamp of when the ManagedFields entry was added. The timestamp will also be updated if a field is added, the manager changes any of the owned fields value or removes a field. The timestamp does not update when a field is removed from the entry because another manager took it over.

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **ownerReferences** ([]OwnerReference)

Patch strategy: merge on key uid

List of objects depended by this object. If ALL objects in the list have been deleted, this object will be garbage collected. If this object is managed by a controller, then an entry in this list will point to this controller, with the controller field set to true. There cannot be more than one managing controller.

OwnerReference contains enough information to let you identify an owning object. An owning object must be in the same namespace as the dependent, or be cluster-scoped, so there is no namespace field.

- **ownerReferences.apiVersion** (string), required

API version of the referent.

- **ownerReferences.kind** (string), required

Kind of the referent. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **ownerReferences.name** (string), required

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names#names>

- **ownerReferences.uid** (string), required

UID of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names#uids>

- **ownerReferences.blockOwnerDeletion** (boolean)

If true, AND if the owner has the "foregroundDeletion" finalizer, then the owner cannot be deleted from the key-value store until this reference is removed. See <https://kubernetes.io/docs/concepts/architecture/garbage-collection/#foreground-deletion> for how the garbage collector interacts with this field and enforces the foreground deletion. Defaults to false. To set this field, a user needs "delete" permission of the owner, otherwise 422 (Unprocessable Entity) will be returned.

- **ownerReferences.controller** (boolean)

If true, this reference points to the managing controller.

Read-only

- **creationTimestamp** (Time)

CreationTimestamp is a timestamp representing the server time when this object was created. It is not guaranteed to be set in happens-before order across separate operations. Clients may not set this value. It is represented in RFC3339 form and is in UTC.

Populated by the system. Read-only. Null for lists. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata>

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **deletionGracePeriodSeconds** (int64)

Number of seconds allowed for this object to gracefully terminate before it will be removed from the system. Only set when deletionTimestamp is also set. May only be shortened. Read-only.

- **deletionTimestamp** (Time)

DeletionTimestamp is RFC 3339 date and time at which this resource will be deleted. This field is set by the server when a graceful deletion is requested by the user, and is not directly settable by a client. The resource is expected to be deleted (no longer visible from resource lists, and not reachable by name) after the time in this field, once the finalizers list is empty. As long as the finalizers list contains items, deletion is blocked. Once the deletionTimestamp is set, this value may not be unset or be set further into the future, although it may be shortened or the resource may be deleted prior to this time. For example, a user may request that a pod is deleted in 30 seconds. The Kubelet will react by sending a graceful termination signal to the containers in the pod. After that 30 seconds, the Kubelet will send a hard termination signal (SIGKILL) to the container and after cleanup, remove the pod from the API. In the presence of network partitions, this object

may still exist after this timestamp, until an administrator or automated process can determine the resource is fully terminated. If not set, graceful deletion of the object has not been requested.

Populated by the system when a graceful deletion is requested. Read-only. More info: <https://git.k8s.io/community/contributors-devel/sig-architecture/api-conventions.md#metadata>

Time is a wrapper around time.Time which supports correct marshaling to YAML and JSON. Wrappers are provided for many of the factory methods that the time package offers.

- **generation** (int64)

A sequence number representing a specific generation of the desired state. Populated by the system. Read-only.

- **resourceVersion** (string)

An opaque value that represents the internal version of this object that can be used by clients to determine when objects have changed. May be used for optimistic concurrency, change detection, and the watch operation on a resource or set of resources. Clients must treat these values as opaque and passed unmodified back to the server. They may only be valid for a particular resource or set of resources.

Populated by the system. Read-only. Value must be treated as opaque by clients and . More info: <https://git.k8s.io/community/contributors-devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency>

- **selfLink** (string)

Deprecated: selfLink is a legacy read-only field that is no longer populated by the system.

- **uid** (string)

UID is the unique in time and space value for this object. It is typically generated by the server on successful creation of a resource and is not allowed to change on PUT operations.

Populated by the system. Read-only. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names#uids>

ObjectReference

ObjectReference contains enough information to let you inspect or modify the referred object.

```
import "k8s.io/api/core/v1"
```

ObjectReference contains enough information to let you inspect or modify the referred object.

- **apiVersion** (string)

API version of the referent.

fieldPath (string)

- If referring to a piece of an object instead of an entire object, this string should contain a valid JSON/Go field access statement, such as `desiredState.manifest.containers[2]`. For example, if the object reference is to a container within a pod, this would take on a value like: `"spec.containers{name}"` (where "name" refers to the name of the container that triggered the event) or if no container name is specified `"spec.containers[2]"` (container with index 2 in this pod). This syntax is chosen only to have some well-defined way of referencing a part of an object.

• **kind** (string)

Name of the referent. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

• **name** (string)

Name of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#names>

• **namespace** (string)

Namespace of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/>

• **resourceVersion** (string)

Specific resourceVersion to which this reference is made, if any. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#concurrency-control-and-consistency>

• **uid** (string)

UID of the referent. More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names/#uids>

Patch

Patch is provided to give a concrete name and type to the Kubernetes PATCH request body.

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
```

Patch is provided to give a concrete name and type to the Kubernetes PATCH request body.

Quantity

Quantity is a fixed-point representation of a number.

```
import "k8s.io/apimachinery/pkg/api/resource"
```

Quantity is a fixed-point representation of a number. It provides convenient marshaling/unmarshaling in JSON and YAML, in addition to String() and AsInt64() accessors.

The serialization format is:

(Note that \<suffix> may be empty, from the "" case in \<decimalSI>.)

```
\<digit> ::= 0 | 1 | ... | 9 \<digits> ::= \<digit> | \<digit>\<digits> \<number> ::= \<digits> | \<digits>.\<digits> | \<digits>. | .\<digits> \<sign> ::= "+" | "-"
\<signedNumber> ::= \<number> | \<sign>\<number> \<suffix> ::= \<binarySI> |
\<decimalExponent> | \<decimalSI> \<binarySI> ::= Ki | Mi | Gi | Ti | Pi | Ei
```

(International System of units; See: <http://physics.nist.gov/cuu/Units/binary.html>)

```
\<decimalSI> ::= m | "" | k | M | G | T | P | E
```

(Note that 1024 = 1Ki but 1000 = 1k; I didn't choose the capitalization.)

```
\<decimalExponent> ::= "e" \<signedNumber> | "E" \<signedNumber> ````
```

No matter which of the three exponent forms is used, no quantity may represent a number greater than $2^{63}-1$ in magnitude, nor may it have more than 3 decimal places. Numbers larger or more precise will be capped or rounded up. (E.g.: 0.1m will rounded up to 1m.) This may be extended in the future if we require larger or smaller quantities.

When a Quantity is parsed from a string, it will remember the type of suffix it had, and will use the same type again when it is serialized.

Before serializing, Quantity will be put in "canonical form". This means that Exponent/suffix will be adjusted up or down (with a corresponding increase or decrease in Mantissa) such that:

- No precision is lost - No fractional digits will be emitted - The exponent (or suffix) is as large as possible.

The sign will be omitted unless the number is negative.

Examples:

- 1.5 will be serialized as "1500m" - 1.5Gi will be serialized as "1536Mi"

Note that the quantity will NEVER be internally represented by a floating point number. That is the whole point of this exercise.

Non-canonical values will still parse as long as they are well formed, but will be re-emitted in their canonical form. (So always use canonical form, or don't diff.)

This format is intended to make it difficult to use these numbers without writing some sort of special handling code in the hopes that that will cause implementors to also use a fixed point implementation.

<hr>

ResourceFieldSelector

ResourceFieldSelector represents container resources (cpu, memory) and their output format.

```
import "k8s.io/api/core/v1"
```

ResourceFieldSelector represents container resources (cpu, memory) and their output format

- **resource** (string), required

Required: resource to select

- **containerName** (string)

Container name: required for volumes, optional for env vars

- **divisor** ([Quantity](#))

Specifies the output format of the exposed resources, defaults to "1"

Status

Status is a return value for calls that don't return other objects.

```
import "k8s.io/apimachinery/pkg/apis/meta/v1"
```

Status is a return value for calls that don't return other objects.

- **apiVersion** (string)

APIVersion defines the versioned schema of this representation of an object. Servers should convert recognized schemas to the latest internal value, and may reject unrecognized values. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#resources>

- **code** (int32)

Suggested HTTP return code for this status, 0 if not set.

- **details** (StatusDetails)

Extended data associated with the reason. Each reason may define its own extended details. This field is optional and the data returned is not guaranteed to conform to any schema except that defined by the reason type.

StatusDetails is a set of additional properties that MAY be set by the server to provide additional information about a response. The Reason field of a Status object defines what attributes will be set. Clients must ignore fields that do not match the defined type of each attribute, and should assume that any attribute may be empty, invalid, or under defined.

- **details.causes** ([]StatusCause)

The Causes array includes more details associated with the StatusReason failure. Not all StatusReasons may provide detailed causes.

StatusCause provides more information about an api.Status failure, including cases when multiple errors are encountered.

- **details.causes.field** (string)

The field of the resource that has caused this error, as named by its JSON serialization. May include dot and postfix notation for nested attributes. Arrays are zero-indexed. Fields may appear more than once in an array of causes due to fields having multiple errors. Optional.

Examples: "name" - the field "name" on the current resource "items[0].name" - the field "name" on the first array entry in "items"

- **details.causes.message** (string)

A human-readable description of the cause of the error. This field may be presented as-is to a reader.

- **details.causes.reason** (string)

A machine-readable description of the cause of the error. If this value is empty there is no information available.

- **details.group** (string)

The group attribute of the resource associated with the status StatusReason.

- **details.kind** (string)

The kind attribute of the resource associated with the status StatusReason. On some operations may differ from the requested resource Kind. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **details.name** (string)

The name attribute of the resource associated with the status StatusReason (when there is a single name which can be described).

- **details.retryAfterSeconds** (int32)

If specified, the time in seconds before the operation should be retried. Some errors may indicate the client must take an alternate action - for those errors this field may indicate how long to wait before taking the alternate action.

- **details.uid** (string)

UID of the resource. (when there is a single resource which can be described). More info: <https://kubernetes.io/docs/concepts/overview/working-with-objects/names#uids>

- **kind** (string)

Kind is a string value representing the REST resource this object represents. Servers may infer this from the endpoint the client submits requests to. Cannot be updated. In CamelCase. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **message** (string)

A human-readable description of the status of this operation.

- **metadata** ([ListMeta](#))

Standard list metadata. More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds>

- **reason** (string)

A machine-readable description of why this operation is in the "Failure" status. If this value is empty there is no information available. A Reason clarifies an HTTP status code but does not override it.

- **status** (string)

Status of the operation. One of: "Success" or "Failure". More info: <https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#spec-and-status>

TypedLocalObjectReference

TypedLocalObjectReference contains enough information to let you locate the typed referenced object inside the same namespace.

```
import "k8s.io/api/core/v1"
```

TypedLocalObjectReference contains enough information to let you locate the typed referenced object inside the same namespace.

- **kind** (string), required

Kind is the type of resource being referenced

- **name** (string), required

Name is the name of resource being referenced

- **apiGroup** (string)

APIGroup is the group for the resource being referenced. If APIGroup is not specified, the specified Kind must be in the core API group. For any other third-party types, APIGroup is required.

Other Resources

[ValidatingAdmissionPolicyBindingList v1beta1](#)

ValidatingAdmissionPolicyBindingList v1beta1

```
apiVersion: admissionregistration.k8s.io/v1beta1  
import "k8s.io/api/admissionregistration/v1beta1"
```

Common Parameters

allowWatchBookmarks

allowWatchBookmarks requests watch events with type "BOOKMARK". Servers that do not implement bookmarks may ignore this flag and bookmarks are sent at the server's discretion. Clients should not assume bookmarks are returned at any specific interval, nor may they assume the server will send any BOOKMARK event during a session. If this is not a watch, this field is ignored.

continue

The continue option should be set when retrieving more results from the server. Since this value is server defined, clients may only use the continue value from a previous query result with identical query parameters (except for the value of continue) and the server may reject a continue value it does not recognize. If the specified continue value is no longer valid whether due to expiration (generally five to fifteen minutes) or a configuration change on the server, the server will respond with a 410 ResourceExpired error together with a continue token. If the client needs a consistent list, it must restart their list without the continue field. Otherwise, the client may send another list request with the token received with the 410 error, the server will respond with a list starting from the next key, but from the latest snapshot, which is inconsistent from the previous list results - objects that are created, modified, or deleted after the first list request will be included in the response, as long as their keys are after the "next key".

This field is not supported when watch is true. Clients may start a watch from the last resourceVersion value returned by the server and not miss any modifications.

dryRun

When present, indicates that modifications should not be persisted. An invalid or unrecognized dryRun directive will result in an error response and no further processing of the request. Valid values are: - All: all dry run stages will be processed

fieldManager

fieldManager is a name associated with the actor or entity that is making these changes. The value must be less than or 128 characters long, and only contain printable characters, as defined by <https://golang.org/pkg/unicode/#IsPrint>.

fieldSelector

A selector to restrict the list of returned objects by their fields. Defaults to everything.

fieldValidation

fieldValidation instructs the server on how to handle objects in the request (POST/PUT/PATCH) containing unknown or duplicate fields. Valid values are:

- Ignore: This will ignore any unknown fields that are silently dropped from the object, and will ignore all but the last duplicate field that the decoder encounters. This is the default behavior prior to v1.23.
- Warn: This will send a warning via the standard warning response header for each unknown field that is dropped from the object, and for each duplicate field that is encountered. The request will still succeed if there are no other errors, and will only persist the last of any duplicate fields. This is the default in v1.23+.
- Strict: This will fail the request with a BadRequest error if any unknown fields would be dropped from the object, or if any duplicate fields are present. The error returned from the server will contain all unknown and duplicate fields encountered.

force

Force is going to "force" Apply requests. It means user will re-acquire conflicting fields owned by other people. Force flag must be unset for non-apply patch requests.

gracePeriodSeconds

The duration in seconds before the object should be deleted. Value must be non-negative integer. The value zero indicates delete immediately. If this value is nil, the default grace period for the specified type will be used. Defaults to a per object value if not specified. zero means delete immediately.

labelSelector

A selector to restrict the list of returned objects by their labels. Defaults to everything.

limit

limit is a maximum number of responses to return for a list call. If more items exist, the server will set the continue field on the list metadata to a value that can be used with the same initial query to retrieve the next set of results. Setting a limit may return fewer than the requested amount of items (up to zero items) in the event all requested objects are filtered out and clients should only use the presence of the continue field to determine whether more results are available. Servers may choose not to support the limit argument and will return all of the available results. If limit is specified and the continue field is empty, clients may assume that no more results are available. This field is not supported if watch is true.

The server guarantees that the objects returned when using continue will be identical to issuing a single list call without a limit - that is, no objects created, modified, or deleted after the first request is issued will be included in any subsequent continued requests. This is sometimes referred to as a consistent snapshot, and ensures that a client that is using limit to receive smaller chunks of a very large result can ensure they see all possible objects. If objects are updated during a chunked list the version of the object that was present at the time the first list result was calculated is returned.

namespace

object name and auth scope, such as for teams and projects

pretty

If 'true', then the output is pretty printed.

propagationPolicy

Whether and how garbage collection will be performed. Either this field or OrphanDependents may be set, but not both. The default policy is decided by the existing finalizer set in the metadata.finalizers and the resource-specific default policy. Acceptable values are: 'Orphan' - orphan the dependents; 'Background' - allow the garbage collector to delete the dependents in the background; 'Foreground' - a cascading policy that deletes all dependents in the foreground.

resourceVersion

resourceVersion sets a constraint on what resource versions a request may be served from. See <https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions> for details.

Defaults to unset

resourceVersionMatch

resourceVersionMatch determines how resourceVersion is applied to list calls. It is highly recommended that resourceVersionMatch be set for list calls where resourceVersion is set. See <https://kubernetes.io/docs/reference/using-api/api-concepts/#resource-versions> for details.

Defaults to unset

sendInitialEvents

sendInitialEvents=true may be set together with watch=true. In that case, the watch stream will begin with synthetic events to produce the current state of objects in the collection. Once all such events have been sent, a synthetic "Bookmark" event will be sent. The bookmark will report the ResourceVersion (RV) corresponding to the set of objects, and be marked with "k8s.io/initial-events-end": "true" annotation. Afterwards, the watch stream will proceed as usual, sending watch events corresponding to changes (subsequent to the RV) to objects watched.

When sendInitialEvents option is set, we require resourceVersionMatch option to also be set. The semantic of the watch request is as following: - resourceVersionMatch = NotOlderThan is interpreted as "data at least as new as the provided resourceVersion" and the bookmark event is send when the state is synced to a resourceVersion at least as fresh as the one provided by the ListOptions. If resourceVersion is unset, this is interpreted as "consistent read" and the bookmark event is send when the state is synced at least to the moment when request started being processed.

- resourceVersionMatch set to any other value or unset Invalid error is returned.

Defaults to true if resourceVersion="" or resourceVersion="0" (for backward compatibility reasons) and to false otherwise.

timeoutSeconds

Timeout for the list/watch call. This limits the duration of the call, regardless of any activity or inactivity.

watch

Watch for changes to the described resources and return them as a stream of add, update, and remove notifications. Specify resourceVersion.

Instrumentation

[CRI Pod & Container Metrics](#)

Collection of Pod & Container metrics via the CRI.

[Node metrics data](#)

Mechanisms for accessing metrics at node, volume, pod and container level, as seen by the kubelet.

[Kubernetes Metrics Reference](#)

Details of the metric data that Kubernetes components export.

Kubernetes Component SLI Metrics

FEATURE STATE: Kubernetes v1.27 [beta]

By default, Kubernetes 1.28 publishes Service Level Indicator (SLI) metrics for each Kubernetes component binary. This metric endpoint is exposed on the serving HTTPS port of each component, at the path /metrics/slis. The ComponentSLIs [feature gate](#) defaults to enabled for each Kubernetes component as of v1.27.

SLI Metrics

With SLI metrics enabled, each Kubernetes component exposes two metrics, labeled per healthcheck:

- a gauge (which represents the current state of the healthcheck)
- a counter (which records the cumulative counts observed for each healthcheck state)

You can use the metric information to calculate per-component availability statistics. For example, the API server checks the health of etcd. You can work out and report how available or unavailable etcd has been - as reported by its client, the API server.

The prometheus gauge data looks like this:

```
# HELP kubernetes_healthcheck [ALPHA] This metric records the result of a single
healthcheck.
# TYPE kubernetes_healthcheck gauge
kubernetes_healthcheck{name="autoregister-completion",type="healthz"} 1
kubernetes_healthcheck{name="autoregister-completion",type="readyz"} 1
kubernetes_healthcheck{name="etcd",type="healthz"} 1
kubernetes_healthcheck{name="etcd",type="readyz"} 1
kubernetes_healthcheck{name="etcd-readiness",type="readyz"} 1
kubernetes_healthcheck{name="informer-sync",type="readyz"} 1
kubernetes_healthcheck{name="log",type="healthz"} 1
kubernetes_healthcheck{name="log",type="readyz"} 1
kubernetes_healthcheck{name="ping",type="healthz"} 1
kubernetes_healthcheck{name="ping",type="readyz"} 1
```

While the counter data looks like this:

```

# HELP kubernetes_healthchecks_total [ALPHA] This metric records the results of all
healthcheck.
# TYPE kubernetes_healthchecks_total counter
kubernetes_healthchecks_total{name="autoregister-completion",status="error",type="readyz"} 1
kubernetes_healthchecks_total{name="autoregister-
completion",status="success",type="healthz"} 15
kubernetes_healthchecks_total{name="autoregister-completion",status="success",type="readyz"} 14
kubernetes_healthchecks_total{name="etcd",status="success",type="healthz"} 15
kubernetes_healthchecks_total{name="etcd",status="success",type="readyz"} 15
kubernetes_healthchecks_total{name="etcd-readiness",status="success",type="readyz"} 15
kubernetes_healthchecks_total{name="informer-sync",status="error",type="readyz"} 1
kubernetes_healthchecks_total{name="informer-sync",status="success",type="readyz"} 14
kubernetes_healthchecks_total{name="log",status="success",type="healthz"} 15
kubernetes_healthchecks_total{name="log",status="success",type="readyz"} 15
kubernetes_healthchecks_total{name="ping",status="success",type="healthz"} 15
kubernetes_healthchecks_total{name="ping",status="success",type="readyz"} 15

```

Using this data

The component SLIs metrics endpoint is intended to be scraped at a high frequency. Scraping at a high frequency means that you end up with greater granularity of the gauge's signal, which can be then used to calculate SLOs. The /metrics/slis endpoint provides the raw data necessary to calculate an availability SLO for the respective Kubernetes component.

CRI Pod & Container Metrics

Collection of Pod & Container metrics via the CRI.

FEATURE STATE: Kubernetes v1.23 [alpha]

The [kubelet](#) collects pod and container metrics via [cAdvisor](#). As an alpha feature, Kubernetes lets you configure the collection of pod and container metrics via the [Container Runtime Interface](#) (CRI). You must enable the `PodAndContainerStatsFromCRI` [feature gate](#) and use a compatible CRI implementation (containerd \geq 1.6.0, CRI-O \geq 1.23.0) to use the CRI based collection mechanism.

CRI Pod & Container Metrics

With `PodAndContainerStatsFromCRI` enabled, the kubelet polls the underlying container runtime for pod and container stats instead of inspecting the host system directly using cAdvisor. The benefits of relying on the container runtime for this information as opposed to direct collection with cAdvisor include:

- Potential improved performance if the container runtime already collects this information during normal operations. In this case, the data can be re-used instead of being aggregated again by the kubelet.
- It further decouples the kubelet and the container runtime allowing collection of metrics for container runtimes that don't run processes directly on the host with kubelet where they are observable by cAdvisor (for example: container runtimes that use virtualization).

Node metrics data

Mechanisms for accessing metrics at node, volume, pod and container level, as seen by the kubelet.

The [kubelet](#) gathers metric statistics at the node, volume, pod and container level, and emits this information in the [Summary API](#).

You can send a proxied request to the stats summary API via the Kubernetes API server.

Here is an example of a Summary API request for a node named minikube:

```
kubectl get --raw "/api/v1/nodes/minikube/proxy/stats/summary"
```

Here is the same API call using curl:

```
# You need to run "kubectl proxy" first  
# Change 8080 to the port that "kubectl proxy" assigns  
curl http://localhost:8080/api/v1/nodes/minikube/proxy/stats/summary
```

Note: Beginning with metrics-server 0.6.x, metrics-server queries the /metrics/resource kubelet endpoint, and not /stats/summary.

Summary metrics API source

By default, Kubernetes fetches node summary metrics data using an embedded [cAdvisor](#) that runs within the kubelet. If you enable the PodAndContainerStatsFromCRI [feature gate](#) in your cluster, and you use a container runtime that supports statistics access via [Container Runtime Interface](#) (CRI), then the kubelet [fetches Pod- and container-level metric data using CRI](#), and not via cAdvisor.

What's next

The task pages for [Troubleshooting Clusters](#) discuss how to use a metrics pipeline that rely on these data.

Kubernetes Metrics Reference

Details of the metric data that Kubernetes components export.

Metrics (v1.29)

This page details the metrics that different Kubernetes components export. You can query the metrics endpoint for these components using an HTTP scrape, and fetch the current metrics data in Prometheus format.

List of Stable Kubernetes Metrics

Stable metrics observe strict API contracts and no labels can be added or removed from stable metrics during their lifetime.

`apiserver_admission_controller_admission_duration_seconds`

Admission controller latency histogram in seconds, identified by name and broken out for each operation and API resource and type (validate or admit).

- Stability Level:STABLE
- Type: Histogram
- Labels:nameoperationrejectedtype

`apiserver_admission_step_admission_duration_seconds`

Admission sub-step latency histogram in seconds, broken out for each operation and API resource and step type (validate or admit).

- Stability Level:STABLE
- Type: Histogram
- Labels:operationrejectedtype

`apiserver_admission_webhook_admission_duration_seconds`

Admission webhook latency histogram in seconds, identified by name and broken out for each operation and API resource and type (validate or admit).

- Stability Level:STABLE
- Type: Histogram
- Labels:nameoperationrejectedtype

`apiserver_current_inflight_requests`

Maximal number of currently used inflight request limit of this apiserver per request kind in last second.

- Stability Level:STABLE
- Type: Gauge
- Labels:request_kind

`apiserver_longrunning_requests`

Gauge of all active long-running apiserver requests broken out by verb, group, version, resource, scope and component. Not all requests are tracked this way.

- Stability Level:STABLE
- Type: Gauge
- Labels:componentgroupresourcescopesubresourceverbversion

`apiserver_request_duration_seconds`

Response latency distribution in seconds for each verb, dry run value, group, version, resource, subresource, scope and component.

- Stability Level:STABLE
- Type: Histogram
- Labels:componentdry_rungroupresourcescopesubresourceverbversion

`apiserver_request_total`

Counter of apiserver requests broken out for each verb, dry run value, group, version, resource, scope, component, and HTTP response code.

- Stability Level:STABLE
- Type: Counter
- Labels:codecomponentdry_rungroupresourcescopesubresourceverbversion

apiserver_requested_DEPRECATED_apis

Gauge of deprecated APIs that have been requested, broken out by API group, version, resource, subresource, and removed_release.

- Stability Level:STABLE
- Type: Gauge
- Labels:groupremoved_releaseresourcesubresourceversion

apiserver_response_sizes

Response size distribution in bytes for each group, version, verb, resource, subresource, scope and component.

- Stability Level:STABLE
- Type: Histogram
- Labels:componentgroupresourcescopesubresourceverbversion

apiserver_storage_objects

Number of stored objects at the time of last check split by kind.

- Stability Level:STABLE
- Type: Gauge
- Labels:resource

container_cpu_usage_seconds_total

Cumulative cpu time consumed by the container in core-seconds

- Stability Level:STABLE
- Type: Custom
- Labels:containerpodnamespace

container_memory_working_set_bytes

Current working set of the container in bytes

- Stability Level:STABLE
- Type: Custom
- Labels:containerpodnamespace

container_start_time_seconds

Start time of the container since unix epoch in seconds

- Stability Level:STABLE
- Type: Custom
- Labels:containerpodnamespace

cronjob_controller_job_creation_skew_duration_seconds

Time between when a cronjob is scheduled to be run, and when the corresponding job is created

- Stability Level:STABLE
- Type: Histogram

job_controller_job_pods_finished_total

The number of finished Pods that are fully tracked

- Stability Level:STABLE
- Type: Counter
- Labels:completion_moderesult

job_controller_job_sync_duration_seconds

The time it took to sync a job

- Stability Level:STABLE
- Type: Histogram
- Labels:actioncompletion_moderesult

job_controller_job_syncs_total

The number of job syncs

- Stability Level:STABLE
- Type: Counter
- Labels:actioncompletion_moderesult

job_controller_jobs_finished_total

The number of finished jobs

- Stability Level:STABLE
- Type: Counter
- Labels:completion_modereasonresult

kube_pod_resource_limit

Resources limit for workloads on the cluster, broken down by pod. This shows the resource usage the scheduler and kubelet expect per pod for resources along with the unit for the resource if any.

- Stability Level:STABLE
- Type: Custom
- Labels:namespacepodnodeschedulerpriorityresourceunit

kube_pod_resource_request

Resources requested by workloads on the cluster, broken down by pod. This shows the resource usage the scheduler and kubelet expect per pod for resources along with the unit for the resource if any.

- Stability Level:STABLE
- Type: Custom
- Labels:namespacepodnodeschedulerpriorityresourceunit

node_collector_evictions_total

Number of Node evictions that happened since current instance of NodeController started.

- Stability Level:STABLE
- Type: Counter
- Labels:zone

node_cpu_usage_seconds_total

Cumulative cpu time consumed by the node in core-seconds

- Stability Level:STABLE
- Type: Custom

node_memory_working_set_bytes

Current working set of the node in bytes

- Stability Level:STABLE
- Type: Custom

pod_cpu_usage_seconds_total

Cumulative cpu time consumed by the pod in core-seconds

- Stability Level:STABLE
- Type: Custom
- Labels:podnamespace

pod_memory_working_set_bytes

Current working set of the pod in bytes

- Stability Level:STABLE
- Type: Custom
- Labels:podnamespace

resource_scrape_error

1 if there was an error while getting container metrics, 0 otherwise

- Stability Level:STABLE
- Type: Custom

scheduler_framework_extension_point_duration_seconds

Latency for running all plugins of a specific extension point.

- Stability Level:STABLE
- Type: Histogram
- Labels:extension_pointprofilestatus

scheduler_pending_pods

Number of pending pods, by the queue type. 'active' means number of pods in activeQ; 'backoff' means number of pods in backoffQ; 'unschedulable' means number of pods in unschedulablePods that the scheduler attempted to schedule and failed; 'gated' is the number of unschedulable pods that the scheduler never attempted to schedule because they are gated.

- Stability Level:STABLE
- Type: Gauge
- Labels:queue

`scheduler_pod_scheduling_attempts`

Number of attempts to successfully schedule a pod.

- Stability Level:STABLE
- Type: Histogram

`scheduler_pod_scheduling_duration_seconds`

E2e latency for a pod being scheduled which may include multiple scheduling attempts.

- Stability Level:STABLE
- Type: Histogram
- Labels:attempts
- Deprecated Versions:1.28.0

`scheduler_preemption_attempts_total`

Total preemption attempts in the cluster till now

- Stability Level:STABLE
- Type: Counter

`scheduler_preemption_victims`

Number of selected preemption victims

- Stability Level:STABLE
- Type: Histogram

`scheduler_queue_incoming_pods_total`

Number of pods added to scheduling queues by event and queue type.

- Stability Level:STABLE
- Type: Counter
- Labels:eventqueue

`scheduler_schedule_attempts_total`

Number of attempts to schedule pods, by the result. 'unschedulable' means a pod could not be scheduled, while 'error' means an internal scheduler problem.

- Stability Level:STABLE
- Type: Counter
- Labels:profileresult

`scheduler_scheduling_attempt_duration_seconds`

Scheduling attempt latency in seconds (scheduling algorithm + binding)

- Stability Level:STABLE
- Type: Histogram
- Labels:profileresult

List of Beta Kubernetes Metrics

Beta metrics observe a looser API contract than its stable counterparts. No labels can be removed from beta metrics during their lifetime, however, labels can be added while the metric is in the beta stage. This offers the assurance that beta metrics will honor existing dashboards and alerts, while allowing for amendments in the future.

`apiserver_flowcontrol_current_executing_requests`

Number of requests in initial (for a WATCH) or any (for a non-WATCH) execution stage in the API Priority and Fairness subsystem

- Stability Level:BETA
- Type: Gauge
- Labels:flow_schemapriority_level

`apiserver_flowcontrol_current_executing_seats`

Concurrency (number of seats) occupied by the currently executing (initial stage for a WATCH, any stage otherwise) requests in the API Priority and Fairness subsystem

- Stability Level:BETA
- Type: Gauge
- Labels:flow_schemapriority_level

`apiserver_flowcontrol_current_inqueue_requests`

Number of requests currently pending in queues of the API Priority and Fairness subsystem

- Stability Level:BETA
- Type: Gauge
- Labels:flow_schemapriority_level

`apiserver_flowcontrol_dispatched_requests_total`

Number of requests executed by API Priority and Fairness subsystem

- Stability Level:BETA
- Type: Counter
- Labels:flow_schemapriority_level

`apiserver_flowcontrol_nominal_limit_seats`

Nominal number of execution seats configured for each priority level

- Stability Level:BETA
- Type: Gauge
- Labels:priority_level

`apiserver_flowcontrol_rejected_requests_total`

Number of requests rejected by API Priority and Fairness subsystem

- Stability Level:BETA
- Type: Counter
- Labels:flow_schemapriority_levelreason

`apiserver_flowcontrol_request_wait_duration_seconds`

Length of time a request spent waiting in its queue

- Stability Level:BETA
- Type: Histogram
- Labels:executeflow_schemapriority_level

`disabled_metrics_total`

The count of disabled metrics.

- Stability Level:BETA

- Type: Counter

hidden_metrics_total

The count of hidden metrics.

- Stability Level:BETA
- Type: Counter

kubernetes_feature_enabled

This metric records the data about the stage and enablement of a k8s feature.

- Stability Level:BETA
- Type: Gauge
- Labels:namestage

kubernetes_healthcheck

This metric records the result of a single healthcheck.

- Stability Level:BETA
- Type: Gauge
- Labels:nametype

kubernetes_healthchecks_total

This metric records the results of all healthcheck.

- Stability Level:BETA
- Type: Counter
- Labels:namestatustype

registered_metrics_total

The count of registered metrics broken by stability level and deprecation version.

- Stability Level:BETA
- Type: Counter
- Labels:deprecated_versionstability_level

scheduler_pod_scheduling_sli_duration_seconds

E2e latency for a pod being scheduled, from the time the pod enters the scheduling queue and might involve multiple scheduling attempts.

- Stability Level:BETA
- Type: Histogram
- Labels:attempts

List of Alpha Kubernetes Metrics

Alpha metrics do not have any API guarantees. These metrics must be used at your own risk, subsequent versions of Kubernetes may remove these metrics altogether, or mutate the API in such a way that breaks existing dashboards and alerts.

aggregator_discovery_aggregation_count_total

Counter of number of times discovery was aggregated

- Stability Level:ALPHA

- Type: Counter

aggregator_openapi_v2_regeneration_count

Counter of OpenAPI v2 spec regeneration count broken down by causing APIService name and reason.

- Stability Level:ALPHA
- Type: Counter
- Labels:apiservicereason

aggregator_openapi_v2_regeneration_duration

Gauge of OpenAPI v2 spec regeneration duration in seconds.

- Stability Level:ALPHA
- Type: Gauge
- Labels:reason

aggregator_unavailable_apiservice

Gauge of APIServices which are marked as unavailable broken down by APIService name.

- Stability Level:ALPHA
- Type: Custom
- Labels:name

aggregator_unavailable_apiservice_total

Counter of APIServices which are marked as unavailable broken down by APIService name and reason.

- Stability Level:ALPHA
- Type: Counter
- Labels:namereason

apiextensions_openapi_v2_regeneration_count

Counter of OpenAPI v2 spec regeneration count broken down by causing CRD name and reason.

- Stability Level:ALPHA
- Type: Counter
- Labels:crdreason

apiextensions_openapi_v3_regeneration_count

Counter of OpenAPI v3 spec regeneration count broken down by group, version, causing CRD and reason.

- Stability Level:ALPHA
- Type: Counter
- Labels:crdgroupreasonversion

apiserver_admission_match_condition_evaluation_errors_total

Admission match condition evaluation errors count, identified by name of resource containing the match condition and broken out for each kind containing matchConditions (webhook or policy), operation and admission type (validate or admit).

- Stability Level:ALPHA
- Type: Counter

- Labels:kindnameoperationtype

apiserver_admission_match_condition_evaluation_seconds

Admission match condition evaluation time in seconds, identified by name and broken out for each kind containing matchConditions (webhook or policy), operation and type (validate or admit).

- Stability Level:ALPHA
- Type: Histogram
- Labels:kindnameoperationtype

apiserver_admission_match_condition_exclusions_total

Admission match condition evaluation exclusions count, identified by name of resource containing the match condition and broken out for each kind containing matchConditions (webhook or policy), operation and admission type (validate or admit).

- Stability Level:ALPHA
- Type: Counter
- Labels:kindnameoperationtype

apiserver_admission_step_admission_duration_seconds_summary

Admission sub-step latency summary in seconds, broken out for each operation and API resource and step type (validate or admit).

- Stability Level:ALPHA
- Type: Summary
- Labels:operationrejectedtype

apiserver_admission_webhook_fail_open_count

Admission webhook fail open count, identified by name and broken out for each admission type (validating or mutating).

- Stability Level:ALPHA
- Type: Counter
- Labels:nametype

apiserver_admission_webhook_rejection_count

Admission webhook rejection count, identified by name and broken out for each admission type (validating or admit) and operation. Additional labels specify an error type (calling_webhook_error or apiserver_internal_error if an error occurred; no_error otherwise) and optionally a non-zero rejection code if the webhook rejects the request with an HTTP status code (honored by the apiserver when the code is greater or equal to 400). Codes greater than 600 are truncated to 600, to keep the metrics cardinality bounded.

- Stability Level:ALPHA
- Type: Counter
- Labels:error_typeoperationrejection_codetype

apiserver_admission_webhook_request_total

Admission webhook request total, identified by name and broken out for each admission type (validating or mutating) and operation. Additional labels specify whether the request was

rejected or not and an HTTP status code. Codes greater than 600 are truncated to 600, to keep the metrics cardinality bounded.

- Stability Level:ALPHA
- Type: Counter
- Labels:codenameoperationrejectedtype

apiserver_audit_error_total

Counter of audit events that failed to be audited properly. Plugin identifies the plugin affected by the error.

- Stability Level:ALPHA
- Type: Counter
- Labels:plugin

apiserver_audit_event_total

Counter of audit events generated and sent to the audit backend.

- Stability Level:ALPHA
- Type: Counter

apiserver_audit_level_total

Counter of policy levels for audit events (1 per request).

- Stability Level:ALPHA
- Type: Counter
- Labels:level

apiserver_audit_requests_rejected_total

Counter of apiserver requests rejected due to an error in audit logging backend.

- Stability Level:ALPHA
- Type: Counter

apiserver_cache_list_fetched_objects_total

Number of objects read from watch cache in the course of serving a LIST request

- Stability Level:ALPHA
- Type: Counter
- Labels:indexresource_prefix

apiserver_cache_list_returned_objects_total

Number of objects returned for a LIST request from watch cache

- Stability Level:ALPHA
- Type: Counter
- Labels:resource_prefix

apiserver_cache_list_total

Number of LIST requests served from watch cache

- Stability Level:ALPHA
- Type: Counter
- Labels:indexresource_prefix

`apiserver_cel_compilation_duration_seconds`
CEL compilation time in seconds.

- Stability Level:ALPHA
- Type: Histogram

`apiserver_cel_evaluation_duration_seconds`
CEL evaluation time in seconds.

- Stability Level:ALPHA
- Type: Histogram

`apiserver_certificates_registry_csr_honored_duration_total`
Total number of issued CSRs with a requested duration that was honored, sliced by signer (only kubernetes.io signer names are specifically identified)

- Stability Level:ALPHA
- Type: Counter
- Labels:signerName

`apiserver_certificates_registry_csr_requested_duration_total`
Total number of issued CSRs with a requested duration, sliced by signer (only kubernetes.io signer names are specifically identified)

- Stability Level:ALPHA
- Type: Counter
- Labels:signerName

`apiserver_client_certificate_expiration_seconds`
Distribution of the remaining lifetime on the certificate used to authenticate a request.

- Stability Level:ALPHA
- Type: Histogram

`apiserver_conversion_webhook_duration_seconds`
Conversion webhook request latency

- Stability Level:ALPHA
- Type: Histogram
- Labels:failure_typeresult

`apiserver_conversion_webhook_request_total`
Counter for conversion webhook requests with success/failure and failure error type

- Stability Level:ALPHA
- Type: Counter
- Labels:failure_typeresult

`apiserver_crd_conversion_webhook_duration_seconds`
CRD webhook conversion duration in seconds

- Stability Level:ALPHA
- Type: Histogram
- Labels:crd_namefrom_versionsucceededto_version

`apiserver_current_inqueue_requests`

Maximal number of queued requests in this apiserver per request kind in last second.

- Stability Level:ALPHA
- Type: Gauge
- Labels:request_kind

`apiserver_delegated_authn_request_duration_seconds`

Request latency in seconds. Broken down by status code.

- Stability Level:ALPHA
- Type: Histogram
- Labels:code

`apiserver_delegated_authn_request_total`

Number of HTTP requests partitioned by status code.

- Stability Level:ALPHA
- Type: Counter
- Labels:code

`apiserver_delegated_authz_request_duration_seconds`

Request latency in seconds. Broken down by status code.

- Stability Level:ALPHA
- Type: Histogram
- Labels:code

`apiserver_delegated_authz_request_total`

Number of HTTP requests partitioned by status code.

- Stability Level:ALPHA
- Type: Counter
- Labels:code

`apiserver_egress_dialer_dial_duration_seconds`

Dial latency histogram in seconds, labeled by the protocol (http-connect or grpc), transport (tcp or uds)

- Stability Level:ALPHA
- Type: Histogram
- Labels:protocoltransport

`apiserver_egress_dialer_dial_failure_count`

Dial failure count, labeled by the protocol (http-connect or grpc), transport (tcp or uds), and stage (connect or proxy). The stage indicates at which stage the dial failed

- Stability Level:ALPHA
- Type: Counter
- Labels:protocolstagetransport

`apiserver_egress_dialer_dial_start_total`

Dial starts, labeled by the protocol (http-connect or grpc) and transport (tcp or uds).

- Stability Level:ALPHA

- Type: Counter
- Labels: protocoltransport

apiserver_encryption_config_controller_automatic_reload_failures_total
Total number of failed automatic reloads of encryption configuration.

- Stability Level: ALPHA
- Type: Counter

apiserver_encryption_config_controller_automatic_reload_last_timestamp_seconds
Timestamp of the last successful or failed automatic reload of encryption configuration.

- Stability Level: ALPHA
- Type: Gauge
- Labels: status

apiserver_encryption_config_controller_automatic_reload_success_total
Total number of successful automatic reloads of encryption configuration.

- Stability Level: ALPHA
- Type: Counter

apiserver_envelope_encryption_dek_cache_fill_percent
Percent of the cache slots currently occupied by cached DEKs.

- Stability Level: ALPHA
- Type: Gauge

apiserver_envelope_encryption_dek_cache_inter_arrival_time_seconds
Time (in seconds) of inter arrival of transformation requests.

- Stability Level: ALPHA
- Type: Histogram
- Labels: transformation_type

apiserver_envelope_encryption_dek_source_cache_size
Number of records in data encryption key (DEK) source cache. On a restart, this value is an approximation of the number of decrypt RPC calls the server will make to the KMS plugin.

- Stability Level: ALPHA
- Type: Gauge
- Labels: provider_name

apiserver_envelope_encryption_invalid_key_id_from_status_total
Number of times an invalid keyID is returned by the Status RPC call split by error.

- Stability Level: ALPHA
- Type: Counter
- Labels: errorprovider_name

apiserver_envelope_encryption_key_id_hash_last_timestamp_seconds
The last time in seconds when a keyID was used.

- Stability Level: ALPHA
- Type: Gauge

- Labels:key_id_hashprovider_nametransformation_type

apiserver_envelope_encryption_key_id_hash_status_last_timestamp_seconds
The last time in seconds when a keyID was returned by the Status RPC call.

- Stability Level:ALPHA
- Type: Gauge
- Labels:key_id_hashprovider_name

apiserver_envelope_encryption_key_id_hash_total
Number of times a keyID is used split by transformation type and provider.

- Stability Level:ALPHA
- Type: Counter
- Labels:key_id_hashprovider_nametransformation_type

apiserver_envelope_encryption_kms_operations_latency_seconds
KMS operation duration with gRPC error code status total.

- Stability Level:ALPHA
- Type: Histogram
- Labels:grpc_status_codemethod_nameprovider_name

apiserver_flowcontrol_current_inqueue_seats
Number of seats currently pending in queues of the API Priority and Fairness subsystem

- Stability Level:ALPHA
- Type: Gauge
- Labels:flow_schemapriority_level

apiserver_flowcontrol_current_limit_seats
current derived number of execution seats available to each priority level

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_current_r
R(time of last change)

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_demand_seats
Observations, at the end of every nanosecond, of (the number of seats each priority level could use) / (nominal number of seats for that level)

- Stability Level:ALPHA
- Type: TimingRatioHistogram
- Labels:priority_level

apiserver_flowcontrol_demand_seats_average

Time-weighted average, over last adjustment period, of demand_seats

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_demand_seats_high_watermark

High watermark, over last adjustment period, of demand_seats

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_demand_seats_smoothed

Smoothed seat demands

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_demand_seats_stdev

Time-weighted standard deviation, over last adjustment period, of demand_seats

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_dispatch_r

R(time of last dispatch)

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_epoch_advance_total

Number of times the queueset's progress meter jumped backward

- Stability Level:ALPHA
- Type: Counter
- Labels:priority_levelsuccess

apiserver_flowcontrol_latest_s

S(most recently dispatched request)

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_lower_limit_seats

Configured lower bound on number of execution seats available to each priority level

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_next_discounted_s_bounds

min and max, over queues, of S(oldest waiting request in queue) - estimated work in progress

- Stability Level:ALPHA
- Type: Gauge
- Labels:boundpriority_level

apiserver_flowcontrol_next_s_bounds

min and max, over queues, of S(oldest waiting request in queue)

- Stability Level:ALPHA
- Type: Gauge
- Labels:boundpriority_level

apiserver_flowcontrol_priority_level_request_utilization

Observations, at the end of every nanosecond, of number of requests (as a fraction of the relevant limit) waiting or in any stage of execution (but only initial stage for WATCHes)

- Stability Level:ALPHA
- Type: TimingRatioHistogram
- Labels:phasepriority_level

apiserver_flowcontrol_priority_level_seat_utilization

Observations, at the end of every nanosecond, of utilization of seats for any stage of execution (but only initial stage for WATCHes)

- Stability Level:ALPHA
- Type: TimingRatioHistogram
- Labels:priority_level
- Const Labels:phase:executing

apiserver_flowcontrol_read_vs_write_current_requests

Observations, at the end of every nanosecond, of the number of requests (as a fraction of the relevant limit) waiting or in regular stage of execution

- Stability Level:ALPHA
- Type: TimingRatioHistogram
- Labels:phaserequest_kind

apiserver_flowcontrol_request_concurrency_in_use

Concurrency (number of seats) occupied by the currently executing (initial stage for a WATCH, any stage otherwise) requests in the API Priority and Fairness subsystem

- Stability Level:ALPHA
- Type: Gauge
- Labels:flow_schemapriority_level
- Deprecated Versions:1.31.0

apiserver_flowcontrol_request_concurrency_limit

Nominal number of execution seats configured for each priority level

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

- Deprecated Versions:1.30.0

apiserver_flowcontrol_request_dispatch_no_accommodation_total

Number of times a dispatch attempt resulted in a non accommodation due to lack of available seats

- Stability Level:ALPHA
- Type: Counter
- Labels:flow_schemapriority_level

apiserver_flowcontrol_request_execution_seconds

Duration of initial stage (for a WATCH) or any (for a non-WATCH) stage of request execution in the API Priority and Fairness subsystem

- Stability Level:ALPHA
- Type: Histogram
- Labels:flow_schemapriority_leveltype

apiserver_flowcontrol_request_queue_length_after_enqueue

Length of queue in the API Priority and Fairness subsystem, as seen by each request after it is enqueued

- Stability Level:ALPHA
- Type: Histogram
- Labels:flow_schemapriority_level

apiserver_flowcontrol_seat_fair_frac

Fair fraction of server's concurrency to allocate to each priority level that can use it

- Stability Level:ALPHA
- Type: Gauge

apiserver_flowcontrol_target_seats

Seat allocation targets

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_upper_limit_seats

Configured upper bound on number of execution seats available to each priority level

- Stability Level:ALPHA
- Type: Gauge
- Labels:priority_level

apiserver_flowcontrol_watch_count_samples

count of watchers for mutating requests in API Priority and Fairness

- Stability Level:ALPHA
- Type: Histogram
- Labels:flow_schemapriority_level

apiserver_flowcontrol_work_estimated_seats

Number of estimated seats (maximum of initial and final seats) associated with requests in API Priority and Fairness

- Stability Level:ALPHA
- Type: Histogram
- Labels:flow_schemapriority_level

apiserver_init_events_total

Counter of init events processed in watch cache broken by resource type.

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

apiserver_kube_aggregator_x509_insecure_sha1_total

Counts the number of requests to servers with insecure SHA1 signatures in their serving certificate OR the number of connection failures due to the insecure SHA1 signatures (either/or, based on the runtime environment)

- Stability Level:ALPHA
- Type: Counter

apiserver_kube_aggregator_x509_missing_san_total

Counts the number of requests to servers missing SAN extension in their serving certificate OR the number of connection failures due to the lack of x509 certificate SAN extension missing (either/or, based on the runtime environment)

- Stability Level:ALPHA
- Type: Counter

apiserver_request_aborts_total

Number of requests which apiserver aborted possibly due to a timeout, for each group, version, verb, resource, subresource and scope

- Stability Level:ALPHA
- Type: Counter
- Labels:groupresourcescopesubresourceverbversion

apiserver_request_body_sizes

Apiserver request body sizes broken out by size.

- Stability Level:ALPHA
- Type: Histogram
- Labels:resourceverb

apiserver_request_filter_duration_seconds

Request filter latency distribution in seconds, for each filter type

- Stability Level:ALPHA
- Type: Histogram
- Labels:filter

apiserver_request_post_timeout_total

Tracks the activity of the request handlers after the associated requests have been timed out by the apiserver

- Stability Level:ALPHA
- Type: Counter
- Labels:sourcetstatus

`apiserver_request_sli_duration_seconds`

Response latency distribution (not counting webhook duration and priority & fairness queue wait times) in seconds for each verb, group, version, resource, subresource, scope and component.

- Stability Level:ALPHA
- Type: Histogram
- Labels:componentgroupresourcescopesubresourceverbversion

`apiserver_request_slo_duration_seconds`

Response latency distribution (not counting webhook duration and priority & fairness queue wait times) in seconds for each verb, group, version, resource, subresource, scope and component.

- Stability Level:ALPHA
- Type: Histogram
- Labels:componentgroupresourcescopesubresourceverbversion
- Deprecated Versions:1.27.0

`apiserver_request_terminations_total`

Number of requests which apiserver terminated in self-defense.

- Stability Level:ALPHA
- Type: Counter
- Labels:codecomponentgroupresourcescopesubresourceverbversion

`apiserver_request_timestamp_comparison_time`

Time taken for comparison of old vs new objects in UPDATE or PATCH requests

- Stability Level:ALPHA
- Type: Histogram
- Labels:code_path

`apiserver_rerouted_request_total`

Total number of requests that were proxied to a peer kube apiserver because the local apiserver was not capable of serving it

- Stability Level:ALPHA
- Type: Counter
- Labels:code

`apiserver_selfrequest_total`

Counter of apiserver self-requests broken out for each verb, API resource and subresource.

- Stability Level:ALPHA
- Type: Counter
- Labels:resourcesubresourceverb

`apiserver_storage_data_key_generation_duration_seconds`
Latencies in seconds of data encryption key(DEK) generation operations.

- Stability Level:ALPHA
- Type: Histogram

`apiserver_storage_data_key_generation_failures_total`
Total number of failed data encryption key(DEK) generation operations.

- Stability Level:ALPHA
- Type: Counter

`apiserver_storage_db_total_size_in_bytes`
Total size of the storage database file physically allocated in bytes.

- Stability Level:ALPHA
- Type: Gauge
- Labels:endpoint
- Deprecated Versions:1.28.0

`apiserver_storage_decode_errors_total`
Number of stored object decode errors split by object type

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

`apiserver_storage_envelope_transformation_cache_misses_total`
Total number of cache misses while accessing key decryption key(KEK).

- Stability Level:ALPHA
- Type: Counter

`apiserver_storage_events_received_total`
Number of etcd events received split by kind.

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

`apiserver_storage_list_evaluated_objects_total`
Number of objects tested in the course of serving a LIST request from storage

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

`apiserver_storage_list_fetched_objects_total`
Number of objects read from storage in the course of serving a LIST request

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

`apiserver_storage_list_returned_objects_total`

Number of objects returned for a LIST request from storage

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

apiserver_storage_list_total

Number of LIST requests served from storage

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

apiserver_storage_size_bytes

Size of the storage database file physically allocated in bytes.

- Stability Level:ALPHA
- Type: Custom
- Labels:cluster

apiserver_storage_transformation_duration_seconds

Latencies in seconds of value transformation operations.

- Stability Level:ALPHA
- Type: Histogram
- Labels:transformation_type transformer_prefix

apiserver_storage_transformation_operations_total

Total number of transformations. Successful transformation will have a status 'OK' and a varied status string when the transformation fails. This status and transformation_type fields may be used for alerting on encryption/decryption failure using transformation_type from_storage for decryption and to_storage for encryption

- Stability Level:ALPHA
- Type: Counter
- Labels:status transformation_type transformer_prefix

apiserver_terminated_watchers_total

Counter of watchers closed due to unresponsiveness broken by resource type.

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

apiserver_tls_handshake_errors_total

Number of requests dropped with 'TLS handshake error from' error

- Stability Level:ALPHA
- Type: Counter

apiserver_validating_admission_policy_check_duration_seconds

Validation admission latency for individual validation expressions in seconds, labeled by policy and further including binding, state and enforcement action taken.

- Stability Level:ALPHA

- Type: Histogram
- Labels: enforcement_actionpolicypolicy_bindingstate

apiserver_validating_admission_policy_check_total

Validation admission policy check total, labeled by policy and further identified by binding, enforcement action taken, and state.

- Stability Level: ALPHA
- Type: Counter
- Labels: enforcement_actionpolicypolicy_bindingstate

apiserver_validating_admission_policy_definition_total

Validation admission policy count total, labeled by state and enforcement action.

- Stability Level: ALPHA
- Type: Counter
- Labels: enforcement_actionstate

apiserver_watch_cache_events_dispatched_total

Counter of events dispatched in watch cache broken by resource type.

- Stability Level: ALPHA
- Type: Counter
- Labels: resource

apiserver_watch_cache_events_received_total

Counter of events received in watch cache broken by resource type.

- Stability Level: ALPHA
- Type: Counter
- Labels: resource

apiserver_watch_cache_initializations_total

Counter of watch cache initializations broken by resource type.

- Stability Level: ALPHA
- Type: Counter
- Labels: resource

apiserver_watch_events_sizes

Watch event size distribution in bytes

- Stability Level: ALPHA
- Type: Histogram
- Labels: groupkindversion

apiserver_watch_events_total

Number of events sent in watch clients

- Stability Level: ALPHA
- Type: Counter
- Labels: groupkindversion

apiserver_webhooks_x509_insecure_sha1_total

Counts the number of requests to servers with insecure SHA1 signatures in their serving certificate OR the number of connection failures due to the insecure SHA1 signatures (either/or, based on the runtime environment)

- Stability Level:ALPHA
- Type: Counter

apiserver_webhooks_x509_missing_san_total

Counts the number of requests to servers missing SAN extension in their serving certificate OR the number of connection failures due to the lack of x509 certificate SAN extension missing (either/or, based on the runtime environment)

- Stability Level:ALPHA
- Type: Counter

attach_detach_controller_attachdetach_controller_forced_detaches

Number of times the A/D Controller performed a forced detach

- Stability Level:ALPHA
- Type: Counter
- Labels:reason

attachdetach_controller_total_volumes

Number of volumes in A/D Controller

- Stability Level:ALPHA
- Type: Custom
- Labels:plugin_namestate

authenticated_user_requests

Counter of authenticated requests broken out by username.

- Stability Level:ALPHA
- Type: Counter
- Labels:username

authentication_attempts

Counter of authenticated attempts.

- Stability Level:ALPHA
- Type: Counter
- Labels:result

authentication_duration_seconds

Authentication duration in seconds broken out by result.

- Stability Level:ALPHA
- Type: Histogram
- Labels:result

authentication_token_cache_active_fetch_count

- Stability Level:ALPHA
- Type: Gauge
- Labels:status

`authentication_token_cache_fetch_total`

- Stability Level:ALPHA
- Type: Counter
- Labels:status

`authentication_token_cache_request_duration_seconds`

- Stability Level:ALPHA
- Type: Histogram
- Labels:status

`authentication_token_cache_request_total`

- Stability Level:ALPHA
- Type: Counter
- Labels:status

`authorization_attempts_total`

Counter of authorization attempts broken down by result. It can be either 'allowed', 'denied', 'no-opinion' or 'error'.

- Stability Level:ALPHA
- Type: Counter
- Labels:result

`authorization_duration_seconds`

Authorization duration in seconds broken out by result.

- Stability Level:ALPHA
- Type: Histogram
- Labels:result

`cloud_provider_webhook_request_duration_seconds`

Request latency in seconds. Broken down by status code.

- Stability Level:ALPHA
- Type: Histogram
- Labels:codewebhook

`cloud_provider_webhook_request_total`

Number of HTTP requests partitioned by status code.

- Stability Level:ALPHA
- Type: Counter
- Labels:codewebhook

`cloudprovider_azure_api_request_duration_seconds`

Latency of an Azure API call

- Stability Level:ALPHA
- Type: Histogram
- Labels:requestresource_groupsourcesubscription_id

`cloudprovider_azure_api_request_errors`

Number of errors for an Azure API call

- Stability Level:ALPHA
- Type: Counter
- Labels:requestresource_groupsourcesubscription_id

cloudprovider_azure_api_request_ratelimited_count

Number of rate limited Azure API calls

- Stability Level:ALPHA
- Type: Counter
- Labels:requestresource_groupsourcesubscription_id

cloudprovider_azure_api_request_throttled_count

Number of throttled Azure API calls

- Stability Level:ALPHA
- Type: Counter
- Labels:requestresource_groupsourcesubscription_id

cloudprovider_azure_op_duration_seconds

Latency of an Azure service operation

- Stability Level:ALPHA
- Type: Histogram
- Labels:requestresource_groupsourcesubscription_id

cloudprovider_azure_op_failure_count

Number of failed Azure service operations

- Stability Level:ALPHA
- Type: Counter
- Labels:requestresource_groupsourcesubscription_id

cloudprovider_gce_api_request_duration_seconds

Latency of a GCE API call

- Stability Level:ALPHA
- Type: Histogram
- Labels:regionrequestversionzone

cloudprovider_gce_api_request_errors

Number of errors for an API call

- Stability Level:ALPHA
- Type: Counter
- Labels:regionrequestversionzone

cloudprovider_vsphere_api_request_duration_seconds

Latency of vsphere api call

- Stability Level:ALPHA
- Type: Histogram
- Labels:request

cloudprovider_vsphere_api_request_errors
vsphere Api errors

- Stability Level:ALPHA
- Type: Counter
- Labels:request

cloudprovider_vsphere_operation_duration_seconds
Latency of vsphere operation call

- Stability Level:ALPHA
- Type: Histogram
- Labels:operation

cloudprovider_vsphere_operation_errors
vsphere operation errors

- Stability Level:ALPHA
- Type: Counter
- Labels:operation

cloudprovider_vsphere_vcenter_versions
Versions for connected vSphere vCenters

- Stability Level:ALPHA
- Type: Custom
- Labels:hostnameversionbuild

container_swap_usage_bytes

Current amount of the container swap usage in bytes. Reported only on non-windows systems

- Stability Level:ALPHA
- Type: Custom
- Labels:containerpodnamespace

csi_operations_seconds

Container Storage Interface operation duration with gRPC error code status total

- Stability Level:ALPHA
- Type: Histogram
- Labels:driver_namegrpc_status_codemethod_namemigrated

endpoint_slice_controller_changes
Number of EndpointSlice changes

- Stability Level:ALPHA
- Type: Counter
- Labels:operation

endpoint_slice_controller_desired_endpoint_slices

Number of EndpointSlices that would exist with perfect endpoint allocation

- Stability Level:ALPHA
- Type: Gauge

`endpoint_slice_controller_endpoints_added_per_sync`

Number of endpoints added on each Service sync

- Stability Level:ALPHA
- Type: Histogram

`endpoint_slice_controller_endpoints_desired`

Number of endpoints desired

- Stability Level:ALPHA
- Type: Gauge

`endpoint_slice_controller_endpoints_removed_per_sync`

Number of endpoints removed on each Service sync

- Stability Level:ALPHA
- Type: Histogram

`endpoint_slice_controller_endpointslices_changed_per_sync`

Number of EndpointSlices changed on each Service sync

- Stability Level:ALPHA
- Type: Histogram
- Labels:topology

`endpoint_slice_controller_num_endpoint_slices`

Number of EndpointSlices

- Stability Level:ALPHA
- Type: Gauge

`endpoint_slice_controller_syncs`

Number of EndpointSlice syncs

- Stability Level:ALPHA
- Type: Counter
- Labels:result

`endpoint_slice_mirroring_controller_addresses_skipped_per_sync`

Number of addresses skipped on each Endpoints sync due to being invalid or exceeding MaxEndpointsPerSubset

- Stability Level:ALPHA
- Type: Histogram

`endpoint_slice_mirroring_controller_changes`

Number of EndpointSlice changes

- Stability Level:ALPHA
- Type: Counter
- Labels:operation

`endpoint_slice_mirroring_controller_desired_endpoint_slices`

Number of EndpointSlices that would exist with perfect endpoint allocation

- Stability Level:ALPHA
- Type: Gauge

endpoint_slice_mirroring_controller_endpoints_added_per_sync

Number of endpoints added on each Endpoints sync

- Stability Level:ALPHA
- Type: Histogram

endpoint_slice_mirroring_controller_endpoints_desired

Number of endpoints desired

- Stability Level:ALPHA
- Type: Gauge

endpoint_slice_mirroring_controller_endpoints_removed_per_sync

Number of endpoints removed on each Endpoints sync

- Stability Level:ALPHA
- Type: Histogram

endpoint_slice_mirroring_controller_endpoints_sync_duration

Duration of syncEndpoints() in seconds

- Stability Level:ALPHA
- Type: Histogram

endpoint_slice_mirroring_controller_endpoints_updated_per_sync

Number of endpoints updated on each Endpoints sync

- Stability Level:ALPHA
- Type: Histogram

endpoint_slice_mirroring_controller_num_endpoint_slices

Number of EndpointSlices

- Stability Level:ALPHA
- Type: Gauge

ephemeral_volume_controller_create_failures_total

Number of PersistenVolumeClaims creation requests

- Stability Level:ALPHA
- Type: Counter

ephemeral_volume_controller_create_total

Number of PersistenVolumeClaims creation requests

- Stability Level:ALPHA
- Type: Counter

etcd_bookmark_counts

Number of etcd bookmarks (progress notify events) split by kind.

- Stability Level:ALPHA
- Type: Gauge
- Labels:resource

etcdb_lease_object_counts

Number of objects attached to a single etcd lease.

- Stability Level:ALPHA
- Type: Histogram

etcdb_request_duration_seconds

Etdc request latency in seconds for each operation and object type.

- Stability Level:ALPHA
- Type: Histogram
- Labels:operationtype

etcdb_request_errors_total

Etdc failed request counts for each operation and object type.

- Stability Level:ALPHA
- Type: Counter
- Labels:operationtype

etcdb_requests_total

Etdc request counts for each operation and object type.

- Stability Level:ALPHA
- Type: Counter
- Labels:operationtype

etcdb_version_info

Etdc server's binary version

- Stability Level:ALPHA
- Type: Gauge
- Labels:binary_version

field_validation_request_duration_seconds

Response latency distribution in seconds for each field validation value

- Stability Level:ALPHA
- Type: Histogram
- Labels:field_validation

force_cleaned_failed_volume_operation_errors_total

The number of volumes that failed force cleanup after their reconstruction failed during kubelet startup.

- Stability Level:ALPHA
- Type: Counter

force_cleaned_failed_volume_operations_total

The number of volumes that were force cleaned after their reconstruction failed during kubelet startup. This includes both successful and failed cleanups.

- Stability Level:ALPHA
- Type: Counter

`garbagecollector_controller_resources_sync_error_total`
Number of garbage collector resources sync errors

- Stability Level:ALPHA
- Type: Counter

`get_token_count`
Counter of total Token() requests to the alternate token source

- Stability Level:ALPHA
- Type: Counter

`get_token_fail_count`
Counter of failed Token() requests to the alternate token source

- Stability Level:ALPHA
- Type: Counter

`horizontal_pod_autoscaler_controller_metric_computation_duration_seconds`
The time(seconds) that the HPA controller takes to calculate one metric. The label 'action' should be either 'scale_down', 'scale_up', or 'none'. The label 'error' should be either 'spec', 'internal', or 'none'. The label 'metric_type' corresponds to HPA.spec.metrics[*].type

- Stability Level:ALPHA
- Type: Histogram
- Labels:actionerrormetric_type

`horizontal_pod_autoscaler_controller_metric_computation_total`
Number of metric computations. The label 'action' should be either 'scale_down', 'scale_up', or 'none'. Also, the label 'error' should be either 'spec', 'internal', or 'none'. The label 'metric_type' corresponds to HPA.spec.metrics[*].type

- Stability Level:ALPHA
- Type: Counter
- Labels:actionerrormetric_type

`horizontal_pod_autoscaler_controller_reconciliation_duration_seconds`
The time(seconds) that the HPA controller takes to reconcile once. The label 'action' should be either 'scale_down', 'scale_up', or 'none'. Also, the label 'error' should be either 'spec', 'internal', or 'none'. Note that if both spec and internal errors happen during a reconciliation, the first one to occur is reported in 'error' label.

- Stability Level:ALPHA
- Type: Histogram
- Labels:actionerror

`horizontal_pod_autoscaler_controller_reconciliations_total`
Number of reconciliations of HPA controller. The label 'action' should be either 'scale_down', 'scale_up', or 'none'. Also, the label 'error' should be either 'spec', 'internal', or 'none'. Note that if

both spec and internal errors happen during a reconciliation, the first one to occur is reported in `error` label.

- Stability Level:ALPHA
- Type: Counter
- Labels:actionerror

job_controller_pod_failures_handled_by_failure_policy_total

`The number of failed Pods handled by failure policy with, respect to the failure policy action applied based on the matched, rule. Possible values of the action label correspond to the, possible values for the failure policy rule action, which are:, "FailJob", "Ignore" and "Count".`

- Stability Level:ALPHA
- Type: Counter
- Labels:action

job_controller_terminated_pods_tracking_finalizer_total

`The number of terminated pods (phase=Failed|Succeeded), that have the finalizer batch.kubernetes.io/job-tracking, The event label can be "add" or "delete".`

- Stability Level:ALPHA
- Type: Counter
- Labels:event

kube_apiserver_clusterip_allocator_allocated_ips

Gauge measuring the number of allocated IPs for Services

- Stability Level:ALPHA
- Type: Gauge
- Labels:cidr

kube_apiserver_clusterip_allocator_allocation_errors_total

Number of errors trying to allocate Cluster IPs

- Stability Level:ALPHA
- Type: Counter
- Labels:cidrscope

kube_apiserver_clusterip_allocator_allocation_total

Number of Cluster IPs allocations

- Stability Level:ALPHA
- Type: Counter
- Labels:cidrscope

kube_apiserver_clusterip_allocator_available_ips

Gauge measuring the number of available IPs for Services

- Stability Level:ALPHA
- Type: Gauge
- Labels:cidr

kube_apiserver_nodeport_allocator_allocated_ports

Gauge measuring the number of allocated NodePorts for Services

- Stability Level:ALPHA
- Type: Gauge

`kube_apiserver_nodeport_allocator_available_ports`

Gauge measuring the number of available NodePorts for Services

- Stability Level:ALPHA
- Type: Gauge

`kube_apiserver_pod_logs_backend_tls_failure_total`

Total number of requests for pods/logs that failed due to kubelet server TLS verification

- Stability Level:ALPHA
- Type: Counter

`kube_apiserver_pod_logs_insecure_backend_total`

Total number of requests for pods/logs sliced by usage type: enforce_tls, skip_tls_allowed, skip_tls_denied

- Stability Level:ALPHA
- Type: Counter
- Labels:usage

`kube_apiserver_pod_logs_pods_logs_backend_tls_failure_total`

Total number of requests for pods/logs that failed due to kubelet server TLS verification

- Stability Level:ALPHA
- Type: Counter
- Deprecated Versions:1.27.0

`kube_apiserver_pod_logs_pods_logs_insecure_backend_total`

Total number of requests for pods/logs sliced by usage type: enforce_tls, skip_tls_allowed, skip_tls_denied

- Stability Level:ALPHA
- Type: Counter
- Labels:usage
- Deprecated Versions:1.27.0

`kubelet_active_pods`

The number of pods the kubelet considers active and which are being considered when admitting new pods. static is true if the pod is not from the apiserver.

- Stability Level:ALPHA
- Type: Gauge
- Labels:static

`kubelet_certificate_manager_client_expiration_renew_errors`

Counter of certificate renewal errors.

- Stability Level:ALPHA
- Type: Counter

kubelet_certificate_manager_client_ttl_seconds

Gauge of the TTL (time-to-live) of the Kubelet's client certificate. The value is in seconds until certificate expiry (negative if already expired). If client certificate is invalid or unused, the value will be +INF.

- Stability Level:ALPHA
- Type: Gauge

kubelet_certificate_manager_server_rotation_seconds

Histogram of the number of seconds the previous certificate lived before being rotated.

- Stability Level:ALPHA
- Type: Histogram

kubelet_certificate_manager_server_ttl_seconds

Gauge of the shortest TTL (time-to-live) of the Kubelet's serving certificate. The value is in seconds until certificate expiry (negative if already expired). If serving certificate is invalid or unused, the value will be +INF.

- Stability Level:ALPHA
- Type: Gauge

kubelet_cgroup_manager_duration_seconds

Duration in seconds for cgroup manager operations. Broken down by method.

- Stability Level:ALPHA
- Type: Histogram
- Labels:operation_type

kubelet_container_log_filesystem_used_bytes

Bytes used by the container's logs on the filesystem.

- Stability Level:ALPHA
- Type: Custom
- Labels:uidnamespacepodcontainer

kubelet_containers_per_pod_count

The number of containers per pod.

- Stability Level:ALPHA
- Type: Histogram

kubelet_cpu_manager_pinning_errors_total

The number of cpu core allocations which required pinning failed.

- Stability Level:ALPHA
- Type: Counter

kubelet_cpu_manager_pinning_requests_total

The number of cpu core allocations which required pinning.

- Stability Level:ALPHA
- Type: Counter

kubelet_credential_provider_plugin_duration

Duration of execution in seconds for credential provider plugin

- Stability Level:ALPHA
- Type: Histogram
- Labels:plugin_name

kubelet_credential_provider_plugin_errors

Number of errors from credential provider plugin

- Stability Level:ALPHA
- Type: Counter
- Labels:plugin_name

kubelet_desired_pods

The number of pods the kubelet is being instructed to run. static is true if the pod is not from the apiserver.

- Stability Level:ALPHA
- Type: Gauge
- Labels:static

kubelet_device_plugin_alloc_duration_seconds

Duration in seconds to serve a device plugin Allocation request. Broken down by resource name.

- Stability Level:ALPHA
- Type: Histogram
- Labels:resource_name

kubelet_device_plugin_registration_total

Cumulative number of device plugin registrations. Broken down by resource name.

- Stability Level:ALPHA
- Type: Counter
- Labels:resource_name

kubelet_evented_pleg_connection_error_count

The number of errors encountered during the establishment of streaming connection with the CRI runtime.

- Stability Level:ALPHA
- Type: Counter

kubelet_evented_pleg_connection_latency_seconds

The latency of streaming connection with the CRI runtime, measured in seconds.

- Stability Level:ALPHA
- Type: Histogram

kubelet_evented_pleg_connection_success_count

The number of times a streaming client was obtained to receive CRI Events.

- Stability Level:ALPHA
- Type: Counter

kubelet_eviction_stats_age_seconds

Time between when stats are collected, and when pod is evicted based on those stats by eviction signal

- Stability Level:ALPHA
- Type: Histogram
- Labels:eviction_signal

kubelet_evictions

Cumulative number of pod evictions by eviction signal

- Stability Level:ALPHA
- Type: Counter
- Labels:eviction_signal

kubelet_graceful_shutdown_end_time_seconds

Last graceful shutdown start time since unix epoch in seconds

- Stability Level:ALPHA
- Type: Gauge

kubelet_graceful_shutdown_start_time_seconds

Last graceful shutdown start time since unix epoch in seconds

- Stability Level:ALPHA
- Type: Gauge

kubelet_http_inflight_requests

Number of the inflight http requests

- Stability Level:ALPHA
- Type: Gauge
- Labels:long_runningmethodpathserver_type

kubelet_http_requests_duration_seconds

Duration in seconds to serve http requests

- Stability Level:ALPHA
- Type: Histogram
- Labels:long_runningmethodpathserver_type

kubelet_http_requests_total

Number of the http requests received since the server started

- Stability Level:ALPHA
- Type: Counter
- Labels:long_runningmethodpathserver_type

kubelet_lifecycle_handler_http_fallbacks_total

The number of times lifecycle handlers successfully fell back to http from https.

- Stability Level:ALPHA
- Type: Counter

kubelet_managed_ephemeral_containers

Current number of ephemeral containers in pods managed by this kubelet.

- Stability Level:ALPHA
- Type: Gauge

kubelet_mirror_pods

The number of mirror pods the kubelet will try to create (one per admitted static pod)

- Stability Level:ALPHA
- Type: Gauge

kubelet_node_name

The node's name. The count is always 1.

- Stability Level:ALPHA
- Type: Gauge
- Labels:node

kubelet_orphan_pod_cleaned_volumes

The total number of orphaned Pods whose volumes were cleaned in the last periodic sweep.

- Stability Level:ALPHA
- Type: Gauge

kubelet_orphan_pod_cleaned_volumes_errors

The number of orphaned Pods whose volumes failed to be cleaned in the last periodic sweep.

- Stability Level:ALPHA
- Type: Gauge

kubelet_orphaned_runtime_pods_total

Number of pods that have been detected in the container runtime without being already known to the pod worker. This typically indicates the kubelet was restarted while a pod was force deleted in the API or in the local configuration, which is unusual.

- Stability Level:ALPHA
- Type: Counter

kubelet_pleg_discard_events

The number of discard events in PLEG.

- Stability Level:ALPHA
- Type: Counter

kubelet_pleg_last_seen_seconds

Timestamp in seconds when PLEG was last seen active.

- Stability Level:ALPHA
- Type: Gauge

kubelet_pleg_relist_duration_seconds

Duration in seconds for relisting pods in PLEG.

- Stability Level:ALPHA
- Type: Histogram

kubelet_pleg_relist_interval_seconds
Interval in seconds between relisting in PLEG.

- Stability Level:ALPHA
- Type: Histogram

kubelet_pod_resources_endpoint_errors_get
Number of requests to the PodResource Get endpoint which returned error. Broken down by server api version.

- Stability Level:ALPHA
- Type: Counter
- Labels:server_api_version

kubelet_pod_resources_endpoint_errors_get_allocatable
Number of requests to the PodResource GetAllocatableResources endpoint which returned error. Broken down by server api version.

- Stability Level:ALPHA
- Type: Counter
- Labels:server_api_version

kubelet_pod_resources_endpoint_errors_list
Number of requests to the PodResource List endpoint which returned error. Broken down by server api version.

- Stability Level:ALPHA
- Type: Counter
- Labels:server_api_version

kubelet_pod_resources_endpoint_requests_get
Number of requests to the PodResource Get endpoint. Broken down by server api version.

- Stability Level:ALPHA
- Type: Counter
- Labels:server_api_version

kubelet_pod_resources_endpoint_requests_get_allocatable
Number of requests to the PodResource GetAllocatableResources endpoint. Broken down by server api version.

- Stability Level:ALPHA
- Type: Counter
- Labels:server_api_version

kubelet_pod_resources_endpoint_requests_list
Number of requests to the PodResource List endpoint. Broken down by server api version.

- Stability Level:ALPHA
- Type: Counter
- Labels:server_api_version

kubelet_pod_resources_endpoint_requests_total

Cumulative number of requests to the PodResource endpoint. Broken down by server api version.

- Stability Level:ALPHA
- Type: Counter
- Labels:server_api_version

kubelet_pod_start_duration_seconds

Duration in seconds from kubelet seeing a pod for the first time to the pod starting to run

- Stability Level:ALPHA
- Type: Histogram

kubelet_pod_start_sli_duration_seconds

Duration in seconds to start a pod, excluding time to pull images and run init containers, measured from pod creation timestamp to when all its containers are reported as started and observed via watch

- Stability Level:ALPHA
- Type: Histogram

kubelet_pod_status_sync_duration_seconds

Duration in seconds to sync a pod status update. Measures time from detection of a change to pod status until the API is successfully updated for that pod, even if multiple intervening changes to pod status occur.

- Stability Level:ALPHA
- Type: Histogram

kubelet_pod_worker_duration_seconds

Duration in seconds to sync a single pod. Broken down by operation type: create, update, or sync

- Stability Level:ALPHA
- Type: Histogram
- Labels:operation_type

kubelet_pod_worker_start_duration_seconds

Duration in seconds from kubelet seeing a pod to starting a worker.

- Stability Level:ALPHA
- Type: Histogram

kubelet_preemptions

Cumulative number of pod preemptions by preemption resource

- Stability Level:ALPHA
- Type: Counter
- Labels:preemption_signal

kubelet_restarted_pods_total

Number of pods that have been restarted because they were deleted and recreated with the same UID while the kubelet was watching them (common for static pods, extremely uncommon for API pods)

- Stability Level:ALPHA
- Type: Counter
- Labels:static

kubelet_run_podsandbox_duration_seconds

Duration in seconds of the run_podsandbox operations. Broken down by RuntimeClass.Handler.

- Stability Level:ALPHA
- Type: Histogram
- Labels:runtime_handler

kubelet_run_podsandbox_errors_total

Cumulative number of the run_podsandbox operation errors by RuntimeClass.Handler.

- Stability Level:ALPHA
- Type: Counter
- Labels:runtime_handler

kubelet_running_containers

Number of containers currently running

- Stability Level:ALPHA
- Type: Gauge
- Labels:container_state

kubelet_running_pods

Number of pods that have a running pod sandbox

- Stability Level:ALPHA
- Type: Gauge

kubelet_runtime_operations_duration_seconds

Duration in seconds of runtime operations. Broken down by operation type.

- Stability Level:ALPHA
- Type: Histogram
- Labels:operation_type

kubelet_runtime_operations_errors_total

Cumulative number of runtime operation errors by operation type.

- Stability Level:ALPHA
- Type: Counter
- Labels:operation_type

kubelet_runtime_operations_total

Cumulative number of runtime operations by operation type.

- Stability Level:ALPHA
- Type: Counter

- Labels:operation_type

kubelet_server_expiration_renew_errors

Counter of certificate renewal errors.

- Stability Level:ALPHA
- Type: Counter

kubelet_started_containers_errors_total

Cumulative number of errors when starting containers

- Stability Level:ALPHA
- Type: Counter
- Labels:codecontainer_type

kubelet_started_containers_total

Cumulative number of containers started

- Stability Level:ALPHA
- Type: Counter
- Labels:container_type

kubelet_started_host_process_containers_errors_total

Cumulative number of errors when starting hostprocess containers. This metric will only be collected on Windows.

- Stability Level:ALPHA
- Type: Counter
- Labels:codecontainer_type

kubelet_started_host_process_containers_total

Cumulative number of hostprocess containers started. This metric will only be collected on Windows.

- Stability Level:ALPHA
- Type: Counter
- Labels:container_type

kubelet_started_pods_errors_total

Cumulative number of errors when starting pods

- Stability Level:ALPHA
- Type: Counter

kubelet_started_pods_total

Cumulative number of pods started

- Stability Level:ALPHA
- Type: Counter

kubelet_topology_manager_admission_duration_ms

Duration in milliseconds to serve a pod admission request.

- Stability Level:ALPHA
- Type: Histogram

kubelet_topology_manager_admission_errors_total

The number of admission request failures where resources could not be aligned.

- Stability Level:ALPHA
- Type: Counter

kubelet_topology_manager_admission_requests_total

The number of admission requests where resources have to be aligned.

- Stability Level:ALPHA
- Type: Counter

kubelet_volume_metric_collection_duration_seconds

Duration in seconds to calculate volume stats

- Stability Level:ALPHA
- Type: Histogram
- Labels:metric_source

kubelet_volume_stats_available_bytes

Number of available bytes in the volume

- Stability Level:ALPHA
- Type: Custom
- Labels:namespacepersistentvolumeclaim

kubelet_volume_stats_capacity_bytes

Capacity in bytes of the volume

- Stability Level:ALPHA
- Type: Custom
- Labels:namespacepersistentvolumeclaim

kubelet_volume_stats_health_status_abnormal

Abnormal volume health status. The count is either 1 or 0. 1 indicates the volume is unhealthy, 0 indicates volume is healthy

- Stability Level:ALPHA
- Type: Custom
- Labels:namespacepersistentvolumeclaim

kubelet_volume_stats_inodes

Maximum number of inodes in the volume

- Stability Level:ALPHA
- Type: Custom
- Labels:namespacepersistentvolumeclaim

kubelet_volume_stats_inodes_free

Number of free inodes in the volume

- Stability Level:ALPHA
- Type: Custom
- Labels:namespacepersistentvolumeclaim

kubelet_volume_stats_inodes_used
Number of used inodes in the volume

- Stability Level: ALPHA
- Type: Custom
- Labels: namespacepersistentvolumeclaim

kubelet_volume_stats_used_bytes
Number of used bytes in the volume

- Stability Level: ALPHA
- Type: Custom
- Labels: namespacepersistentvolumeclaim

kubelet_working_pods

Number of pods the kubelet is actually running, broken down by lifecycle phase, whether the pod is desired, orphaned, or runtime only (also orphaned), and whether the pod is static. An orphaned pod has been removed from local configuration or force deleted in the API and consumes resources that are not otherwise visible.

- Stability Level: ALPHA
- Type: Gauge
- Labels: configlifecyclestatic

kubeproxy_network_programming_duration_seconds
In Cluster Network Programming Latency in seconds

- Stability Level: ALPHA
- Type: Histogram

kubeproxy_proxy_healthz_total
Cumulative proxy healthz HTTP status

- Stability Level: ALPHA
- Type: Counter
- Labels: code

kubeproxy_proxy_livez_total
Cumulative proxy livez HTTP status

- Stability Level: ALPHA
- Type: Counter
- Labels: code

kubeproxy_sync_full_proxy_rules_duration_seconds
SyncProxyRules latency in seconds for full resyncs

- Stability Level: ALPHA
- Type: Histogram

kubeproxy_sync_partial_proxy_rules_duration_seconds
SyncProxyRules latency in seconds for partial resyncs

- Stability Level: ALPHA
- Type: Histogram

kubeproxy_sync_proxy_rules_duration_seconds
SyncProxyRules latency in seconds

- Stability Level:ALPHA
- Type: Histogram

kubeproxy_sync_proxy_rules_endpoint_changes_pending
Pending proxy rules Endpoint changes

- Stability Level:ALPHA
- Type: Gauge

kubeproxy_sync_proxy_rules_endpoint_changes_total
Cumulative proxy rules Endpoint changes

- Stability Level:ALPHA
- Type: Counter

kubeproxy_sync_proxy_rules_iptables_last
Number of iptables rules written by kube-proxy in last sync

- Stability Level:ALPHA
- Type: Gauge
- Labels:table

kubeproxy_sync_proxy_rules_iptables_partial_restore_failures_total
Cumulative proxy iptables partial restore failures

- Stability Level:ALPHA
- Type: Counter

kubeproxy_sync_proxy_rules_iptables_restore_failures_total
Cumulative proxy iptables restore failures

- Stability Level:ALPHA
- Type: Counter

kubeproxy_sync_proxy_rules_iptables_total
Total number of iptables rules owned by kube-proxy

- Stability Level:ALPHA
- Type: Gauge
- Labels:table

kubeproxy_sync_proxy_rules_last_queued_timestamp_seconds
The last time a sync of proxy rules was queued

- Stability Level:ALPHA
- Type: Gauge

kubeproxy_sync_proxy_rules_last_timestamp_seconds
The last time proxy rules were successfully synced

- Stability Level:ALPHA
- Type: Gauge

kubeproxy_sync_proxy_rules_no_local_endpoints_total
Number of services with a Local traffic policy and no endpoints

- Stability Level:ALPHA
- Type: Gauge
- Labels:traffic_policy

kubeproxy_sync_proxy_rules_service_changes_pending
Pending proxy rules Service changes

- Stability Level:ALPHA
- Type: Gauge

kubeproxy_sync_proxy_rules_service_changes_total
Cumulative proxy rules Service changes

- Stability Level:ALPHA
- Type: Counter

kubernetes_build_info

A metric with a constant '1' value labeled by major, minor, git version, git commit, git tree state, build date, Go version, and compiler from which Kubernetes was built, and platform on which it is running.

- Stability Level:ALPHA
- Type: Gauge
- Labels:build_datecompilergit_commitgit_tree_stategit_versiongo_versionmajorminorplatform

leader_election_master_status

Gauge of if the reporting system is master of the relevant lease, 0 indicates backup, 1 indicates master. 'name' is the string used to identify the lease. Please make sure to group by name.

- Stability Level:ALPHA
- Type: Gauge
- Labels:name

node_authorizer_graph_actions_duration_seconds

Histogram of duration of graph actions in node authorizer.

- Stability Level:ALPHA
- Type: Histogram
- Labels:operation

node_collector_unhealthy_nodes_in_zone

Gauge measuring number of not Ready Nodes per zones.

- Stability Level:ALPHA
- Type: Gauge
- Labels:zone

node_collector_update_all_nodes_health_duration_seconds

Duration in seconds for NodeController to update the health of all nodes.

- Stability Level:ALPHA
- Type: Histogram

node_collector_update_node_health_duration_seconds

Duration in seconds for NodeController to update the health of a single node.

- Stability Level:ALPHA
- Type: Histogram

node_collector_zone_health

Gauge measuring percentage of healthy nodes per zone.

- Stability Level:ALPHA
- Type: Gauge
- Labels:zone

node_collector_zone_size

Gauge measuring number of registered Nodes per zones.

- Stability Level:ALPHA
- Type: Gauge
- Labels:zone

node_controller_cloud_provider_taint_removal_delay_seconds

Number of seconds after node creation when NodeController removed the cloud-provider taint of a single node.

- Stability Level:ALPHA
- Type: Histogram

node_controller_initial_node_sync_delay_seconds

Number of seconds after node creation when NodeController finished the initial synchronization of a single node.

- Stability Level:ALPHA
- Type: Histogram

node_ipam_controller_cidrset_allocation_tries_per_request

Number of endpoints added on each Service sync

- Stability Level:ALPHA
- Type: Histogram
- Labels:clusterCIDR

node_ipam_controller_cidrset_cidrs_allocations_total

Counter measuring total number of CIDR allocations.

- Stability Level:ALPHA
- Type: Counter
- Labels:clusterCIDR

node_ipam_controller_cidrset_cidrs_releases_total

Counter measuring total number of CIDR releases.

- Stability Level:ALPHA
- Type: Counter
- Labels:clusterCIDR

`node_ipam_controller_cidrset_usage_cidrs`
Gauge measuring percentage of allocated CIDRs.

- Stability Level:ALPHA
- Type: Gauge
- Labels:clusterCIDR

`node_ipam_controller_cirdset_max_cidrs`
Maximum number of CIDRs that can be allocated.

- Stability Level:ALPHA
- Type: Gauge
- Labels:clusterCIDR

`node_ipam_controller_multicidrset_allocation_tries_per_request`
Histogram measuring CIDR allocation tries per request.

- Stability Level:ALPHA
- Type: Histogram
- Labels:clusterCIDR

`node_ipam_controller_multicidrset_cidrs_allocations_total`
Counter measuring total number of CIDR allocations.

- Stability Level:ALPHA
- Type: Counter
- Labels:clusterCIDR

`node_ipam_controller_multicidrset_cidrs_releases_total`
Counter measuring total number of CIDR releases.

- Stability Level:ALPHA
- Type: Counter
- Labels:clusterCIDR

`node_ipam_controller_multicidrset_usage_cidrs`
Gauge measuring percentage of allocated CIDRs.

- Stability Level:ALPHA
- Type: Gauge
- Labels:clusterCIDR

`node_ipam_controller_multicirdset_max_cidrs`
Maximum number of CIDRs that can be allocated.

- Stability Level:ALPHA
- Type: Gauge
- Labels:clusterCIDR

`node_swap_usage_bytes`
Current swap usage of the node in bytes. Reported only on non-windows systems

- Stability Level:ALPHA
- Type: Custom

`number_of_l4_ilbs`

Number of L4 ILBs

- Stability Level:ALPHA
- Type: Gauge
- Labels:feature

`plugin_manager_total_plugins`

Number of plugins in Plugin Manager

- Stability Level:ALPHA
- Type: Custom
- Labels:socket_pathstate

`pod_gc_collector_force_delete_pod_errors_total`

Number of errors encountered when forcefully deleting the pods since the Pod GC Controller started.

- Stability Level:ALPHA
- Type: Counter
- Labels:namespacereason

`pod_gc_collector_force_delete_pods_total`

Number of pods that are being forcefully deleted since the Pod GC Controller started.

- Stability Level:ALPHA
- Type: Counter
- Labels:namespacereason

`pod_security_errors_total`

Number of errors preventing normal evaluation. Non-fatal errors may result in the latest restricted profile being used for evaluation.

- Stability Level:ALPHA
- Type: Counter
- Labels:fatalrequest_operationresourcesubresource

`pod_security_evaluations_total`

Number of policy evaluations that occurred, not counting ignored or exempt requests.

- Stability Level:ALPHA
- Type: Counter
- Labels:decisionmodepolicy_levelpolicy_versionrequest_operationresourcesubresource

`pod_security_exemptions_total`

Number of exempt requests, not counting ignored or out of scope requests.

- Stability Level:ALPHA
- Type: Counter
- Labels:request_operationresourcesubresource

`pod_swap_usage_bytes`

Current amount of the pod swap usage in bytes. Reported only on non-windows systems

- Stability Level:ALPHA

- Type: Custom
- Labels:podnamespace

prober_probe_duration_seconds

Duration in seconds for a probe response.

- Stability Level:ALPHA
- Type: Histogram
- Labels:containernamespacepodprobe_type

prober_probe_total

Cumulative number of a liveness, readiness or startup probe for a container by result.

- Stability Level:ALPHA
- Type: Counter
- Labels:containernamespacepodpod_uidprobe_typeresult

pv_collector_bound_pv_count

Gauge measuring number of persistent volume currently bound

- Stability Level:ALPHA
- Type: Custom
- Labels:storage_class

pv_collector_bound_pvc_count

Gauge measuring number of persistent volume claim currently bound

- Stability Level:ALPHA
- Type: Custom
- Labels:namespace

pv_collector_total_pv_count

Gauge measuring total number of persistent volumes

- Stability Level:ALPHA
- Type: Custom
- Labels:plugin_namevolume_mode

pv_collector_unbound_pv_count

Gauge measuring number of persistent volume currently unbound

- Stability Level:ALPHA
- Type: Custom
- Labels:storage_class

pv_collector_unbound_pvc_count

Gauge measuring number of persistent volume claim currently unbound

- Stability Level:ALPHA
- Type: Custom
- Labels:namespace

reconstruct_volume_operations_errors_total

The number of volumes that failed reconstruction from the operating system during kubelet startup.

- Stability Level:ALPHA
- Type: Counter

`reconstruct_volume_operations_total`

The number of volumes that were attempted to be reconstructed from the operating system during kubelet startup. This includes both successful and failed reconstruction.

- Stability Level:ALPHA
- Type: Counter

`replicaset_controller_sorting_deletion_age_ratio`

The ratio of chosen deleted pod's ages to the current youngest pod's age (at the time). Should be <2. The intent of this metric is to measure the rough efficacy of the LogarithmicScaleDown feature gate's effect on the sorting (and deletion) of pods when a replicaset scales down. This only considers Ready pods when calculating and reporting.

- Stability Level:ALPHA
- Type: Histogram

`resourceclaim_controller_create_attempts_total`

Number of ResourceClaims creation requests

- Stability Level:ALPHA
- Type: Counter

`resourceclaim_controller_create_failures_total`

Number of ResourceClaims creation request failures

- Stability Level:ALPHA
- Type: Counter

`rest_client_dns_resolution_duration_seconds`

DNS resolver latency in seconds. Broken down by host.

- Stability Level:ALPHA
- Type: Histogram
- Labels:host

`rest_client_exec_plugin_call_total`

Number of calls to an exec plugin, partitioned by the type of event encountered (no_error, plugin_execution_error, plugin_not_found_error, client_internal_error) and an optional exit code. The exit code will be set to 0 if and only if the plugin call was successful.

- Stability Level:ALPHA
- Type: Counter
- Labels:call_statuscode

`rest_client_exec_plugin_certificate_rotation_age`

Histogram of the number of seconds the last auth exec plugin client certificate lived before being rotated. If auth exec plugin client certificates are unused, histogram will contain no data.

- Stability Level:ALPHA

- Type: Histogram

`rest_client_exec_plugin_ttl_seconds`

Gauge of the shortest TTL (time-to-live) of the client certificate(s) managed by the auth exec plugin. The value is in seconds until certificate expiry (negative if already expired). If auth exec plugins are unused or manage no TLS certificates, the value will be +INF.

- Stability Level:ALPHA
- Type: Gauge

`rest_client_rate_limiter_duration_seconds`

Client side rate limiter latency in seconds. Broken down by verb, and host.

- Stability Level:ALPHA
- Type: Histogram
- Labels:hostverb

`rest_client_request_duration_seconds`

Request latency in seconds. Broken down by verb, and host.

- Stability Level:ALPHA
- Type: Histogram
- Labels:hostverb

`rest_client_request_retries_total`

Number of request retries, partitioned by status code, verb, and host.

- Stability Level:ALPHA
- Type: Counter
- Labels:codehostverb

`rest_client_request_size_bytes`

Request size in bytes. Broken down by verb and host.

- Stability Level:ALPHA
- Type: Histogram
- Labels:hostverb

`rest_client_requests_total`

Number of HTTP requests, partitioned by status code, method, and host.

- Stability Level:ALPHA
- Type: Counter
- Labels:codehostmethod

`rest_client_response_size_bytes`

Response size in bytes. Broken down by verb and host.

- Stability Level:ALPHA
- Type: Histogram
- Labels:hostverb

`rest_client_transport_cache_entries`

Number of transport entries in the internal cache.

- Stability Level:ALPHA
- Type: Gauge

`rest_client_transport_create_calls_total`

Number of calls to get a new transport, partitioned by the result of the operation hit: obtained from the cache, miss: created and added to the cache, uncacheable: created and not cached

- Stability Level:ALPHA
- Type: Counter
- Labels:result

`retroactive_storageclass_errors_total`

Total number of failed retroactive StorageClass assignments to persistent volume claim

- Stability Level:ALPHA
- Type: Counter

`retroactive_storageclass_total`

Total number of retroactive StorageClass assignments to persistent volume claim

- Stability Level:ALPHA
- Type: Counter

`root_ca_cert_publisher_sync_duration_seconds`

Number of namespace syncs happened in root ca cert publisher.

- Stability Level:ALPHA
- Type: Histogram
- Labels:code

`root_ca_cert_publisher_sync_total`

Number of namespace syncs happened in root ca cert publisher.

- Stability Level:ALPHA
- Type: Counter
- Labels:code

`running_managed_controllers`

Indicates where instances of a controller are currently running

- Stability Level:ALPHA
- Type: Gauge
- Labels:managername

`scheduler_goroutines`

Number of running goroutines split by the work they do such as binding.

- Stability Level:ALPHA
- Type: Gauge
- Labels:operation

`scheduler_permit_wait_duration_seconds`

Duration of waiting on permit.

- Stability Level:ALPHA
- Type: Histogram
- Labels:result

`scheduler_plugin_evaluation_total`

Number of attempts to schedule pods by each plugin and the extension point (available only in PreFilter and Filter.).

- Stability Level:ALPHA
- Type: Counter
- Labels:extension_pointpluginprofile

`scheduler_plugin_execution_duration_seconds`

Duration for running a plugin at a specific extension point.

- Stability Level:ALPHA
- Type: Histogram
- Labels:extension_pointpluginstatus

`scheduler_scheduler_cache_size`

Number of nodes, pods, and assumed (bound) pods in the scheduler cache.

- Stability Level:ALPHA
- Type: Gauge
- Labels:type

`scheduler_scheduling_algorithm_duration_seconds`

Scheduling algorithm latency in seconds

- Stability Level:ALPHA
- Type: Histogram

`scheduler_unschedulable_pods`

The number of unschedulable pods broken down by plugin name. A pod will increment the gauge for all plugins that caused it to not schedule and so this metric have meaning only when broken down by plugin.

- Stability Level:ALPHA
- Type: Gauge
- Labels:pluginprofile

`scheduler_volume_binder_cache_requests_total`

Total number for request volume binding cache

- Stability Level:ALPHA
- Type: Counter
- Labels:operation

`scheduler_volume_scheduling_stage_error_total`

Volume scheduling stage error count

- Stability Level:ALPHA
- Type: Counter

- Labels:operation

scrape_error

1 if there was an error while getting container metrics, 0 otherwise

- Stability Level:ALPHA
- Type: Custom
- Deprecated Versions:1.29.0

service_controller_loadbalancer_sync_total

A metric counting the amount of times any load balancer has been configured, as an effect of service/node changes on the cluster

- Stability Level:ALPHA
- Type: Counter

service_controller_nodesync_error_total

A metric counting the amount of times any load balancer has been configured and errored, as an effect of node changes on the cluster

- Stability Level:ALPHA
- Type: Counter

service_controller_nodesync_latency_seconds

A metric measuring the latency for nodesync which updates loadbalancer hosts on cluster node updates.

- Stability Level:ALPHA
- Type: Histogram

service_controller_update_loadbalancer_host_latency_seconds

A metric measuring the latency for updating each load balancer hosts.

- Stability Level:ALPHA
- Type: Histogram

serviceaccount_legacy_auto_token_uses_total

Cumulative auto-generated legacy tokens used

- Stability Level:ALPHA
- Type: Counter

serviceaccount_legacy_manual_token_uses_total

Cumulative manually created legacy tokens used

- Stability Level:ALPHA
- Type: Counter

serviceaccount_legacy_tokens_total

Cumulative legacy service account tokens used

- Stability Level:ALPHA
- Type: Counter

serviceaccount_stale_tokens_total

Cumulative stale projected service account tokens used

- Stability Level:ALPHA
- Type: Counter

serviceaccount_valid_tokens_total

Cumulative valid projected service account tokens used

- Stability Level:ALPHA
- Type: Counter

storage_count_attachable_volumes_in_use

Measure number of volumes in use

- Stability Level:ALPHA
- Type: Custom
- Labels:nodenamevolume_plugin

storage_operation_duration_seconds

Storage operation duration

- Stability Level:ALPHA
- Type: Histogram
- Labels:migratedoperation_namestatusvolume_plugin

ttl_after_finished_controller_job_deletion_duration_seconds

The time it took to delete the job since it became eligible for deletion

- Stability Level:ALPHA
- Type: Histogram

volume_manager_selinux_container_errors_total

Number of errors when kubelet cannot compute SELinux context for a container. Kubelet can't start such a Pod then and it will retry, therefore value of this metric may not represent the actual nr. of containers.

- Stability Level:ALPHA
- Type: Gauge

volume_manager_selinux_container_warnings_total

Number of errors when kubelet cannot compute SELinux context for a container that are ignored. They will become real errors when SELinuxMountReadWriteOncePod feature is expanded to all volume access modes.

- Stability Level:ALPHA
- Type: Gauge

volume_manager_selinux_pod_context_mismatch_errors_total

Number of errors when a Pod defines different SELinux contexts for its containers that use the same volume. Kubelet can't start such a Pod then and it will retry, therefore value of this metric may not represent the actual nr. of Pods.

- Stability Level:ALPHA
- Type: Gauge

`volume_manager_selinux_pod_context_mismatch_warnings_total`

Number of errors when a Pod defines different SELinux contexts for its containers that use the same volume. They are not errors yet, but they will become real errors when SELinuxMountReadWriteOncePod feature is expanded to all volume access modes.

- Stability Level:ALPHA
- Type: Gauge

`volume_manager_selinux_volume_context_mismatch_errors_total`

Number of errors when a Pod uses a volume that is already mounted with a different SELinux context than the Pod needs. Kubelet can't start such a Pod then and it will retry, therefore value of this metric may not represent the actual nr. of Pods.

- Stability Level:ALPHA
- Type: Gauge

`volume_manager_selinux_volume_context_mismatch_warnings_total`

Number of errors when a Pod uses a volume that is already mounted with a different SELinux context than the Pod needs. They are not errors yet, but they will become real errors when SELinuxMountReadWriteOncePod feature is expanded to all volume access modes.

- Stability Level:ALPHA
- Type: Gauge

`volume_manager_selinux_volumes_admitted_total`

Number of volumes whose SELinux context was fine and will be mounted with mount -o context option.

- Stability Level:ALPHA
- Type: Gauge

`volume_manager_total_volumes`

Number of volumes in Volume Manager

- Stability Level:ALPHA
- Type: Custom
- Labels:plugin_namestate

`volume_operation_total_errors`

Total volume operation errors

- Stability Level:ALPHA
- Type: Counter
- Labels:operation_nameplugin_name

`volume_operation_total_seconds`

Storage operation end to end duration in seconds

- Stability Level:ALPHA
- Type: Histogram
- Labels:operation_nameplugin_name

`watch_cache_capacity`

Total capacity of watch cache broken by resource type.

- Stability Level:ALPHA
- Type: Gauge
- Labels:resource

watch_cache_capacity_decrease_total

Total number of watch cache capacity decrease events broken by resource type.

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

watch_cache_capacity_increase_total

Total number of watch cache capacity increase events broken by resource type.

- Stability Level:ALPHA
- Type: Counter
- Labels:resource

workqueue_adds_total

Total number of adds handled by workqueue

- Stability Level:ALPHA
- Type: Counter
- Labels:name

workqueue_depth

Current depth of workqueue

- Stability Level:ALPHA
- Type: Gauge
- Labels:name

workqueue_longest_running_processor_seconds

How many seconds has the longest running processor for workqueue been running.

- Stability Level:ALPHA
- Type: Gauge
- Labels:name

workqueue_queue_duration_seconds

How long in seconds an item stays in workqueue before being requested.

- Stability Level:ALPHA
- Type: Histogram
- Labels:name

workqueue_retries_total

Total number of retries handled by workqueue

- Stability Level:ALPHA
- Type: Counter
- Labels:name

`workqueue_unfinished_work_seconds`

How many seconds of work has done that is in progress and hasn't been observed by `work_duration`. Large values indicate stuck threads. One can deduce the number of stuck threads by observing the rate at which this increases.

- Stability Level:ALPHA
- Type: Gauge
- Labels:name

`workqueue_work_duration_seconds`

How long in seconds processing an item from workqueue takes.

- Stability Level:ALPHA
- Type: Histogram
- Labels:name

Kubernetes Issues and Security

[Kubernetes Issue Tracker](#)

[Kubernetes Security and Disclosure Information](#)

[Official CVE Feed](#)

Kubernetes Issue Tracker

To report a security issue, please follow the [Kubernetes security disclosure process](#).

Work on Kubernetes code and public issues are tracked using [GitHub Issues](#).

- Official [list of known CVEs](#) (security vulnerabilities) that have been announced by the [Security Response Committee](#)
- [CVE-related GitHub issues](#)

Security-related announcements are sent to the kubernetes-security-announce@googlegroups.com mailing list.

Kubernetes Security and Disclosure Information

This page describes Kubernetes security and disclosure information.

Security Announcements

Join the kubernetes-security-announce group for emails about security and major API announcements.

Report a Vulnerability

We're extremely grateful for security researchers and users that report vulnerabilities to the Kubernetes Open Source Community. All reports are thoroughly investigated by a set of community volunteers.

To make a report, submit your vulnerability to the [Kubernetes bug bounty program](#). This allows triage and handling of the vulnerability with standardized response times.

You can also email the private security@kubernetes.io list with the security details and the details expected for [all Kubernetes bug reports](#).

You may encrypt your email to this list using the GPG keys of the [Security Response Committee members](#). Encryption using GPG is NOT required to make a disclosure.

When Should I Report a Vulnerability?

- You think you discovered a potential security vulnerability in Kubernetes
- You are unsure how a vulnerability affects Kubernetes
- You think you discovered a vulnerability in another project that Kubernetes depends on
 - For projects with their own vulnerability reporting and disclosure process, please report it directly there

When Should I NOT Report a Vulnerability?

- You need help tuning Kubernetes components for security
- You need help applying security related updates
- Your issue is not security related

Security Vulnerability Response

Each report is acknowledged and analyzed by Security Response Committee members within 3 working days. This will set off the [Security Release Process](#).

Any vulnerability information shared with Security Response Committee stays within Kubernetes project and will not be disseminated to other projects unless it is necessary to get the issue fixed.

As the security issue moves from triage, to identified fix, to release planning we will keep the reporter updated.

Public Disclosure Timing

A public disclosure date is negotiated by the Kubernetes Security Response Committee and the bug submitter. We prefer to fully disclose the bug as soon as possible once a user mitigation is available. It is reasonable to delay disclosure when the bug or the fix is not yet fully understood, the solution is not well-tested, or for vendor coordination. The timeframe for disclosure is from immediate (especially if it's already publicly known) to a few weeks. For a vulnerability with a straightforward mitigation, we expect report date to disclosure date to be on the order of 7 days. The Kubernetes Security Response Committee holds the final say when setting a disclosure date.

Official CVE Feed

FEATURE STATE: Kubernetes v1.27 [beta]

This is a community maintained list of official CVEs announced by the Kubernetes Security Response Committee. See [Kubernetes Security and Disclosure Information](#) for more details.

The Kubernetes project publishes a programmatically accessible feed of published security issues in [JSON feed](#) and [RSS feed](#) formats. You can access it by executing the following commands:

- [JSON feed](#)
- [RSS feed](#)

[Link to JSON format](#)

```
curl -Lv https://k8s.io/docs/reference/issues-security/official-cve-feed/index.json
```

[Link to RSS format](#)

```
curl -Lv https://k8s.io/docs/reference/issues-security/official-cve-feed/feed.xml
```

Official Kubernetes CVE List (last updated: 23 Nov 2023 22:36:56 UTC)

CVE ID	Issue Summary	CVE GitHub Issue URL
CVE-2023-5528	Insufficient input sanitization in in-tree storage plugin leads to privilege escalation on Windows nodes	#121879
CVE-2023-3955	Insufficient input sanitization on Windows nodes leads to privilege escalation	#119595
CVE-2023-3893	Insufficient input sanitization on kubernetes-csi-proxy leads to privilege escalation	#119594
CVE-2023-3676	Insufficient input sanitization on Windows nodes leads to privilege escalation	#119339
CVE-2023-2431	Bypass of seccomp profile enforcement	#118690
CVE-2023-2727, CVE-2023-2728	Bypassing policies imposed by the ImagePolicyWebhook and bypassing mountable secrets policy imposed by the ServiceAccount admission plugin	#118640
CVE-2023-2878	secrets-store-csi-driver discloses service account tokens in logs	#118419
CVE-2022-3294	Node address isn't always verified when proxying	#113757
CVE-2022-3162	Unauthorized read of Custom Resources	#113756
CVE-2022-3172	Aggregated API server can cause clients to be redirected (SSRF)	#112513
CVE-2021-25749	`runAsNonRoot` logic bypass for Windows containers	#112192
CVE-2021-25741	Symlink Exchange Can Allow Host Filesystem Access	#104980
CVE-2021-25737	Holes in EndpointSlice Validation Enable Host Network Hijack	#102106
CVE-2021-3121	Processes may panic upon receipt of malicious protobuf messages	#101435

CVE ID	Issue Summary	CVE GitHub Issue URL
CVE-2021-25735	Validating Admission Webhook does not observe some previous fields	#100096
CVE-2020-8554	Man in the middle using LoadBalancer or ExternalIPs	#97076
CVE-2020-8566	Ceph RBD adminSecrets exposed in logs when loglevel >= 4	#95624
CVE-2020-8565	Incomplete fix for CVE-2019-11250 allows for token leak in logs when logLevel >= 9	#95623
CVE-2020-8564	Docker config secrets leaked when file is malformed and log level >= 4	#95622
CVE-2020-8563	Secret leaks in kube-controller-manager when using vSphere provider	#95621
CVE-2020-8557	Node disk DOS by writing to container /etc/hosts	#93032
CVE-2020-8559	Privilege escalation from compromised node to cluster	#92914
CVE-2020-8558	Node setting allows for neighboring hosts to bypass localhost boundary	#92315
CVE-2020-8555	Half-Blind SSRF in kube-controller-manager	#91542
CVE-2020-10749	IPv4 only clusters susceptible to MitM attacks via IPv6 rogue router advertisements	#91507
CVE-2019-11254	kube-apiserver Denial of Service vulnerability from malicious YAML payloads	#89535
CVE-2020-8552	apiserver DoS (oom)	#89378
CVE-2020-8551	Kubelet DoS via API	#89377
CVE-2019-11251	kubectl cp symlink vulnerability	#87773
CVE-2018-1002102	Unvalidated redirect	#85867
CVE-2019-11255	CSI volume snapshot, cloning and resizing features can result in unauthorized volume data access or mutation	#85233
CVE-2019-11253	Kubernetes API Server JSON/YAML parsing vulnerable to resource exhaustion attack	#83253
CVE-2019-11250	Bearer tokens are revealed in logs	#81114
CVE-2019-11248	/debug/pprof exposed on kubelet's healthz port	#81023
CVE-2019-11249	Incomplete fixes for CVE-2019-1002101 and CVE-2019-11246, kubectl cp potential directory traversal	#80984
CVE-2019-11247	API server allows access to custom resources via wrong scope	#80983
CVE-2019-11245	container uid changes to root after first restart or if image is already pulled to the node	#78308
CVE-2019-11243	rest.AnonymousClientConfig() does not remove the serviceaccount credentials from config created by rest.InClusterConfig()	#76797
CVE-2019-11244	`kubectl -http-cache=<world-accessible dir>` creates world-writeable cached schema files	#76676
CVE-2019-1002100	json-patch requests can exhaust apiserver resources	#74534
CVE-2018-1002105	proxy request handling in kube-apiserver can leave vulnerable TCP connections	#71411
CVE-2018-1002101	smb mount security issue	#65750

CVE ID	Issue Summary	CVE GitHub Issue URL
CVE-2018-1002100	Kubectl copy doesn't check for paths outside of it's destination directory.	#61297
CVE-2017-1002102	atomic writer volume handling allows arbitrary file deletion in host filesystem	#60814
CVE-2017-1002101	subpath volume mount handling allows arbitrary file access in host filesystem	#60813
CVE-2017-1002100	Azure PV should be Private scope not Container scope	#47611
CVE-2017-1000056	PodSecurityPolicy admission plugin authorizes incorrectly	#43459

This feed is auto-refreshing with a noticeable but small lag (minutes to hours) from the time a CVE is announced to the time it is accessible in this feed.

The source of truth of this feed is a set of GitHub Issues, filtered by a controlled and restricted label official-cve-feed. The raw data is stored in a Google Cloud Bucket which is writable only by a small number of trusted members of the Community.

Node Reference Information

This section contains the following reference topics about nodes:

- the kubelet's [checkpoint API](#)
- a list of [Articles on dockershim Removal and on Using CRI-compatible Runtimes](#)
- [Node .status information](#)

You can also read node reference details from elsewhere in the Kubernetes documentation, including:

- [Node Metrics Data](#).
- [CRI Pod & Container Metrics](#).

Kubelet Checkpoint API

FEATURE STATE: Kubernetes v1.25 [alpha]

Checkpointing a container is the functionality to create a stateful copy of a running container. Once you have a stateful copy of a container, you could move it to a different computer for debugging or similar purposes.

If you move the checkpointed container data to a computer that's able to restore it, that restored container continues to run at exactly the same point it was checkpointed. You can also inspect the saved data, provided that you have suitable tools for doing so.

Creating a checkpoint of a container might have security implications. Typically a checkpoint contains all memory pages of all processes in the checkpointed container. This means that

everything that used to be in memory is now available on the local disk. This includes all private data and possibly keys used for encryption. The underlying CRI implementations (the container runtime on that node) should create the checkpoint archive to be only accessible by the root user. It is still important to remember if the checkpoint archive is transferred to another system all memory pages will be readable by the owner of the checkpoint archive.

Operations

post checkpoint the specified container

Tell the kubelet to checkpoint a specific container from the specified Pod.

Consult the [Kubelet authentication/authorization reference](#) for more information about how access to the kubelet checkpoint interface is controlled.

The kubelet will request a checkpoint from the underlying [CRI](#) implementation. In the checkpoint request the kubelet will specify the name of the checkpoint archive as `checkpoint-<podFullName>-<containerName>-<timestamp>.tar` and also request to store the checkpoint archive in the `checkpoints` directory below its root directory (as defined by `--root-dir`). This defaults to `/var/lib/kubelet/checkpoints`.

The checkpoint archive is in `tar` format, and could be listed using an implementation of [tar](#). The contents of the archive depend on the underlying CRI implementation (the container runtime on that node).

HTTP Request

POST /checkpoint/{namespace}/{pod}/{container}

Parameters

- **namespace** (*in path*): string, required

[Namespace](#)

- **pod** (*in path*): string, required

[Pod](#)

- **container** (*in path*): string, required

[Container](#)

- **timeout** (*in query*): integer

Timeout in seconds to wait until the checkpoint creation is finished. If zero or no timeout is specified the default [CRI](#) timeout value will be used. Checkpoint creation time depends directly on the used memory of the container. The more memory a container uses the more time is required to create the corresponding checkpoint.

Response

200: OK

401: Unauthorized

404: Not Found (if the ContainerCheckpoint feature gate is disabled)

404: Not Found (if the specified namespace, pod or container cannot be found)

500: Internal Server Error (if the CRI implementation encounter an error during checkpointing (see error message for further details))

500: Internal Server Error (if the CRI implementation does not implement the checkpoint CRI API (see error message for further details))

Articles on dockershim Removal and on Using CRI-compatible Runtimes

This is a list of articles and other pages that are either about the Kubernetes' deprecation and removal of *dockershim*, or about using CRI-compatible container runtimes, in connection with that removal.

Kubernetes project

- Kubernetes blog: [Dockershim Removal FAQ](#) (originally published 2020/12/02)
- Kubernetes blog: [Updated: Dockershim Removal FAQ](#) (updated published 2022/02/17)
- Kubernetes blog: [Kubernetes is Moving on From Dockershim: Commitments and Next Steps](#) (published 2022/01/07)
- Kubernetes blog: [Dockershim removal is coming. Are you ready?](#) (published 2021/11/12)
- Kubernetes documentation: [Migrating from dockershim](#)
- Kubernetes documentation: [Container Runtimes](#)
- Kubernetes enhancement proposal: [KEP-2221: Removing dockershim from kubelet](#)
- Kubernetes enhancement proposal issue: [Removing dockershim from kubelet \(k/enhancements#2221\)](#)

You can provide feedback via the GitHub issue [Dockershim removal feedback & issues](#). (*k/kubernetes/#106917*)

External sources

- Amazon Web Services EKS documentation: [Amazon EKS is ending support for Dockershim](#)

- CNCF conference video: [Lessons Learned Migrating Kubernetes from Docker to containerd Runtime](#) (Ana Caylin, at KubeCon Europe 2019)
- Docker.com blog: [What developers need to know about Docker, Docker Engine, and Kubernetes v1.20](#) (published 2020/12/04)
- "Google Open Source" channel on YouTube: [Learn Kubernetes with Google - Migrating from Dockershim to Containerd](#)
- Microsoft Apps on Azure blog: [Dockershim deprecation and AKS](#) (published 2022/01/21)
- Mirantis blog: [The Future of Dockershim is cri-dockerd](#) (published 2021/04/21)
- Mirantis: [Mirantis/cri-dockerd](#) Git repository (on GitHub)
- Tripwire: [How Dockershim's Forthcoming Deprecation Affects Your Kubernetes](#) (published 2021/07/01)

Node Labels Populated By The Kubelet

Kubernetes [nodes](#) come pre-populated with a standard set of [labels](#).

You can also set your own labels on nodes, either through the kubelet configuration or using the Kubernetes API.

Preset labels

The preset labels that Kubernetes sets on nodes are:

- [kubernetes.io/arch](#)
- [kubernetes.io/hostname](#)
- [kubernetes.io/os](#)
- [node.kubernetes.io/instance-type](#) (if known to the kubelet – Kubernetes may not have this information to set the label)
- [topology.kubernetes.io/region](#) (if known to the kubelet – Kubernetes may not have this information to set the label)
- [topology.kubernetes.io/zone](#) (if known to the kubelet – Kubernetes may not have this information to set the label)

Note: The value of these labels is cloud provider specific and is not guaranteed to be reliable. For example, the value of kubernetes.io/hostname may be the same as the node name in some environments and a different value in other environments.

What's next

- See [Well-Known Labels, Annotations and Taints](#) for a list of common labels.
- Learn how to [add a label to a node](#).

Kubelet Device Manager API Versions

This page provides details of version compatibility between the Kubernetes [device plugin API](#), and different versions of Kubernetes itself.

Compatibility matrix

	v1alpha1	v1beta1
Kubernetes 1.21	-	
Kubernetes 1.22	-	
Kubernetes 1.23	-	
Kubernetes 1.24	-	
Kubernetes 1.25	-	
Kubernetes 1.26	-	

Key:

- Exactly the same features / API objects in both device plugin API and the Kubernetes version.
- + The device plugin API has features or API objects that may not be present in the Kubernetes cluster, either because the device plugin API has added additional new API calls, or that the server has removed an old API call. However, everything they have in common (most other APIs) will work. Note that alpha APIs may vanish or change significantly between one minor release and the next.
- - The Kubernetes cluster has features the device plugin API can't use, either because server has added additional API calls, or that device plugin API has removed an old API call. However, everything they share in common (most APIs) will work.

Node Status

The status of a [node](#) in Kubernetes is a critical aspect of managing a Kubernetes cluster. In this article, we'll cover the basics of monitoring and maintaining node status to ensure a healthy and stable cluster.

Node status fields

A Node's status contains the following information:

- [Addresses](#)
- [Conditions](#)
- [Capacity and Allocatable](#)
- [Info](#)

You can use kubectl to view a Node's status and other details:

```
kubectl describe node <insert-node-name-here>
```

Each section of the output is described below.

Addresses

The usage of these fields varies depending on your cloud provider or bare metal configuration.

- HostName: The hostname as reported by the node's kernel. Can be overridden via the kubelet --hostname-override parameter.
- ExternalIP: Typically the IP address of the node that is externally routable (available from outside the cluster).
- InternalIP: Typically the IP address of the node that is routable only within the cluster.

Conditions

The conditions field describes the status of all Running nodes. Examples of conditions include:

Node conditions, and a description of when each condition applies.

Node Condition	Description
Ready	True if the node is healthy and ready to accept pods, False if the node is not healthy and is not accepting pods, and Unknown if the node controller has not heard from the node in the last node-monitor-grace-period (default is 40 seconds)
DiskPressure	True if pressure exists on the disk size—that is, if the disk capacity is low; otherwise False
MemoryPressure	True if pressure exists on the node memory—that is, if the node memory is low; otherwise False
PIDPressure	True if pressure exists on the processes—that is, if there are too many processes on the node; otherwise False
NetworkUnavailable	True if the network for the node is not correctly configured, otherwise False

Note: If you use command-line tools to print details of a cordoned Node, the Condition includes SchedulingDisabled. SchedulingDisabled is not a Condition in the Kubernetes API; instead, cordoned nodes are marked Unschedulable in their spec.

In the Kubernetes API, a node's condition is represented as part of the .status of the Node resource. For example, the following JSON structure describes a healthy node:

```
"conditions": [
  {
    "type": "Ready",
    "status": "True",
    "reason": "KubeletReady",
    "message": "kubelet is posting ready status",
    "lastHeartbeatTime": "2019-06-05T18:38:35Z",
    "lastTransitionTime": "2019-06-05T11:41:27Z"
  }
]
```

When problems occur on nodes, the Kubernetes control plane automatically creates [taints](#) that match the conditions affecting the node. An example of this is when the status of the Ready condition remains Unknown or False for longer than the kube-controller-manager's NodeMonitorGracePeriod, which defaults to 40 seconds. This will cause either an

`node.kubernetes.io/unreachable` taint, for an Unknown status, or a `node.kubernetes.io/not-ready` taint, for a False status, to be added to the Node.

These taints affect pending pods as the scheduler takes the Node's taints into consideration when assigning a pod to a Node. Existing pods scheduled to the node may be evicted due to the application of NoExecute taints. Pods may also have [tolerations](#) that let them schedule to and continue running on a Node even though it has a specific taint.

See [Taint Based Evictions](#) and [Taint Nodes by Condition](#) for more details.

Capacity and Allocatable

Describes the resources available on the node: CPU, memory, and the maximum number of pods that can be scheduled onto the node.

The fields in the capacity block indicate the total amount of resources that a Node has. The allocatable block indicates the amount of resources on a Node that is available to be consumed by normal Pods.

You may read more about capacity and allocatable resources while learning how to [reserve compute resources](#) on a Node.

Info

Describes general information about the node, such as kernel version, Kubernetes version (kubelet and kube-proxy version), container runtime details, and which operating system the node uses. The kubelet gathers this information from the node and publishes it into the Kubernetes API.

Heartbeats

Heartbeats, sent by Kubernetes nodes, help your cluster determine the availability of each node, and to take action when failures are detected.

For nodes there are two forms of heartbeats:

- updates to the `.status` of a Node
- [Lease](#) objects within the kube-node-lease [namespace](#). Each Node has an associated Lease object.

Compared to updates to `.status` of a Node, a Lease is a lightweight resource. Using Leases for heartbeats reduces the performance impact of these updates for large clusters.

The kubelet is responsible for creating and updating the `.status` of Nodes, and for updating their related Leases.

- The kubelet updates the node's `.status` either when there is change in status or if there has been no update for a configured interval. The default interval for `.status` updates to Nodes is 5 minutes, which is much longer than the 40 second default timeout for unreachable nodes.
- The kubelet creates and then updates its Lease object every 10 seconds (the default update interval). Lease updates occur independently from updates to the Node's `.status`. If the

Lease update fails, the kubelet retries, using exponential backoff that starts at 200 milliseconds and capped at 7 seconds.

Networking Reference

This section of the Kubernetes documentation provides reference details of Kubernetes networking.

[**Protocols for Services**](#)

[**Ports and Protocols**](#)

[**Virtual IPs and Service Proxies**](#)

Protocols for Services

If you configure a [Service](#), you can select from any network protocol that Kubernetes supports.

Kubernetes supports the following protocols with Services:

- [SCTP](#)
- [TCP](#) (*the default*)
- [UDP](#)

When you define a Service, you can also specify the [application protocol](#) that it uses.

This document details some special cases, all of them typically using TCP as a transport protocol:

- [HTTP](#) and [HTTPS](#)
- [PROXY protocol](#)
- [TLS](#) termination at the load balancer

Supported protocols

There are 3 valid values for the protocol of a port for a Service:

SCTP

FEATURE STATE: Kubernetes v1.20 [stable]

When using a network plugin that supports SCTP traffic, you can use SCTP for most Services. For type: LoadBalancer Services, SCTP support depends on the cloud provider offering this facility. (Most do not).

SCTP is not supported on nodes that run Windows.

Support for multihomed SCTP associations

The support of multihomed SCTP associations requires that the CNI plugin can support the assignment of multiple interfaces and IP addresses to a Pod.

NAT for multihomed SCTP associations requires special logic in the corresponding kernel modules.

TCP

You can use TCP for any kind of Service, and it's the default network protocol.

UDP

You can use UDP for most Services. For type: LoadBalancer Services, UDP support depends on the cloud provider offering this facility.

Special cases

HTTP

If your cloud provider supports it, you can use a Service in LoadBalancer mode to configure a load balancer outside of your Kubernetes cluster, in a special mode where your cloud provider's load balancer implements HTTP / HTTPS reverse proxying, with traffic forwarded to the backend endpoints for that Service.

Typically, you set the protocol for the Service to TCP and add an [annotation](#) (usually specific to your cloud provider) that configures the load balancer to handle traffic at the HTTP level. This configuration might also include serving HTTPS (HTTP over TLS) and reverse-proxying plain HTTP to your workload.

Note: You can also use an [Ingress](#) to expose HTTP/HTTPS Services.

You might additionally want to specify that the [application protocol](#) of the connection is http or https. Use http if the session from the load balancer to your workload is HTTP without TLS, and use https if the session from the load balancer to your workload uses TLS encryption.

PROXY protocol

If your cloud provider supports it, you can use a Service set to type: LoadBalancer to configure a load balancer outside of Kubernetes itself, that will forward connections wrapped with the [PROXY protocol](#).

The load balancer then sends an initial series of octets describing the incoming connection, similar to this example (PROXY protocol v1):

```
PROXY TCP4 192.0.2.202 10.0.42.7 12345 7\r\n
```

The data after the proxy protocol preamble are the original data from the client. When either side closes the connection, the load balancer also triggers a connection close and sends any remaining data where feasible.

Typically, you define a Service with the protocol to TCP. You also set an annotation, specific to your cloud provider, that configures the load balancer to wrap each incoming connection in the PROXY protocol.

TLS

If your cloud provider supports it, you can use a Service set to type: LoadBalancer as a way to set up external reverse proxying, where the connection from client to load balancer is TLS encrypted and the load balancer is the TLS server peer. The connection from the load balancer to your workload can also be TLS, or might be plain text. The exact options available to you depend on your cloud provider or custom Service implementation.

Typically, you set the protocol to TCP and set an annotation (usually specific to your cloud provider) that configures the load balancer to act as a TLS server. You would configure the TLS identity (as server, and possibly also as a client that connects to your workload) using mechanisms that are specific to your cloud provider.

Ports and Protocols

When running Kubernetes in an environment with strict network boundaries, such as on-premises datacenter with physical network firewalls or Virtual Networks in Public Cloud, it is useful to be aware of the ports and protocols used by Kubernetes components.

Control plane

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10259	kube-scheduler	Self
TCP	Inbound	10257	kube-controller-manager	Self

Although etcd ports are included in control plane section, you can also host your own etcd cluster externally or on custom ports.

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Services†	All

† Default port range for [NodePort Services](#).

All default port numbers can be overridden. When custom ports are used those ports need to be open instead of defaults mentioned here.

One common example is API server port that is sometimes switched to 443. Alternatively, the default port is kept as is and API server is put behind a load balancer that listens on 443 and routes the requests to API server on the default port.

Virtual IPs and Service Proxies

Every [node](#) in a Kubernetes [cluster](#) runs a [kube-proxy](#) (unless you have deployed your own alternative component in place of kube-proxy).

The kube-proxy component is responsible for implementing a *virtual IP* mechanism for [Services](#) of type other than [ExternalName](#).

A question that pops up every now and then is why Kubernetes relies on proxying to forward inbound traffic to backends. What about other approaches? For example, would it be possible to configure DNS records that have multiple A values (or AAAA for IPv6), and rely on round-robin name resolution?

There are a few reasons for using proxying for Services:

- There is a long history of DNS implementations not respecting record TTLs, and caching the results of name lookups after they should have expired.
- Some apps do DNS lookups only once and cache the results indefinitely.
- Even if apps and libraries did proper re-resolution, the low or zero TTLs on the DNS records could impose a high load on DNS that then becomes difficult to manage.

Later in this page you can read about how various kube-proxy implementations work. Overall, you should note that, when running kube-proxy, kernel level rules may be modified (for example, iptables rules might get created), which won't get cleaned up, in some cases until you reboot. Thus, running kube-proxy is something that should only be done by an administrator which understands the consequences of having a low level, privileged network proxying service on a computer. Although the kube-proxy executable supports a cleanup function, this function is not an official feature and thus is only available to use as-is.

Some of the details in this reference refer to an example: the backend [Pods](#) for a stateless image-processing workloads, running with three replicas. Those replicas are fungible—frontends do not care which backend they use. While the actual Pods that compose the backend set may change, the frontend clients should not need to be aware of that, nor should they need to keep track of the set of backends themselves.

Proxy modes

The kube-proxy starts up in different modes, which are determined by its configuration.

On Linux nodes, the available modes for kube-proxy are:

[iptables](#)

A mode where the kube-proxy configures packet forwarding rules using iptables, on Linux.

[ipvs](#)

a mode where the kube-proxy configures packet forwarding rules using ipvs.

There is only one mode available for kube-proxy on Windows:

[kernel space](#)

a mode where the kube-proxy configures packet forwarding rules in the Windows kernel

iptables proxy mode

This proxy mode is only available on Linux nodes.

In this mode, kube-proxy watches the Kubernetes [control plane](#) for the addition and removal of Service and EndpointSlice [objects](#). For each Service, it installs iptables rules, which capture traffic to the Service's clusterIP and port, and redirect that traffic to one of the Service's backend sets. For each endpoint, it installs iptables rules which select a backend Pod.

By default, kube-proxy in iptables mode chooses a backend at random.

Using iptables to handle traffic has a lower system overhead, because traffic is handled by Linux netfilter without the need to switch between userspace and the kernel space. This approach is also likely to be more reliable.

If kube-proxy is running in iptables mode and the first Pod that's selected does not respond, the connection fails. This is different from the old userspace mode: in that scenario, kube-proxy would detect that the connection to the first Pod had failed and would automatically retry with a different backend Pod.

You can use Pod [readiness probes](#) to verify that backend Pods are working OK, so that kube-proxy in iptables mode only sees backends that test out as healthy. Doing this means you avoid having traffic sent via kube-proxy to a Pod that's known to have failed.

Virtual IP mechanism for Services, using iptables mode

Example

As an example, consider the image processing application described [earlier](#) in the page. When the backend Service is created, the Kubernetes control plane assigns a virtual IP address, for example 10.0.0.1. For this example, assume that the Service port is 1234. All of the kube-proxy instances in the cluster observe the creation of the new Service.

When kube-proxy on a node sees a new Service, it installs a series of iptables rules which redirect from the virtual IP address to more iptables rules, defined per Service. The per-Service rules link to further rules for each backend endpoint, and the per- endpoint rules redirect traffic (using destination NAT) to the backends.

When a client connects to the Service's virtual IP address the iptables rule kicks in. A backend is chosen (either based on session affinity or randomly) and packets are redirected to the backend without rewriting the client IP address.

This same basic flow executes when traffic comes in through a node-port or through a load-balancer, though in those cases the client IP address does get altered.

Optimizing iptables mode performance

In large clusters (with tens of thousands of Pods and Services), the iptables mode of kube-proxy may take a long time to update the rules in the kernel when Services (or their EndpointSlices) change. You can adjust the syncing behavior of kube-proxy via options in the [iptables section](#) of the kube-proxy [configuration file](#) (which you specify via kube-proxy --config <path>):

```
...
iptables:
  minSyncPeriod: 1s
  syncPeriod: 30s
...
```

minSyncPeriod

The `minSyncPeriod` parameter sets the minimum duration between attempts to resynchronize iptables rules with the kernel. If it is 0s, then kube-proxy will always immediately synchronize the rules every time any Service or Endpoint changes. This works fine in very small clusters, but it results in a lot of redundant work when lots of things change in a small time period. For example, if you have a Service backed by a [Deployment](#) with 100 pods, and you delete the Deployment, then with `minSyncPeriod: 0s`, kube-proxy would end up removing the Service's endpoints from the iptables rules one by one, for a total of 100 updates. With a larger `minSyncPeriod`, multiple Pod deletion events would get aggregated together, so kube-proxy might instead end up making, say, 5 updates, each removing 20 endpoints, which will be much more efficient in terms of CPU, and result in the full set of changes being synchronized faster.

The larger the value of `minSyncPeriod`, the more work that can be aggregated, but the downside is that each individual change may end up waiting up to the full `minSyncPeriod` before being processed, meaning that the iptables rules spend more time being out-of-sync with the current API server state.

The default value of 1s should work well in most clusters, but in very large clusters it may be necessary to set it to a larger value. Especially, if kube-proxy's `sync_proxy_rules_duration_seconds` metric indicates an average time much larger than 1 second, then bumping up `minSyncPeriod` may make updates more efficient.

Updating legacy `minSyncPeriod` configuration

Older versions of kube-proxy updated all the rules for all Services on every sync; this led to performance issues (update lag) in large clusters, and the recommended solution was to set a larger `minSyncPeriod`. Since Kubernetes v1.28, the iptables mode of kube-proxy uses a more minimal approach, only making updates where Services or EndpointSlices have actually changed.

If you were previously overriding `minSyncPeriod`, you should try removing that override and letting kube-proxy use the default value (1s) or at least a smaller value than you were using before upgrading.

If you are not running kube-proxy from Kubernetes 1.28, check the behavior and associated advice for the version that you are actually running.

syncPeriod

The `syncPeriod` parameter controls a handful of synchronization operations that are not directly related to changes in individual Services and EndpointSlices. In particular, it controls how quickly kube-proxy notices if an external component has interfered with kube-proxy's iptables rules. In large clusters, kube-proxy also only performs certain cleanup operations once every `syncPeriod` to avoid unnecessary work.

For the most part, increasing syncPeriod is not expected to have much impact on performance, but in the past, it was sometimes useful to set it to a very large value (eg, 1h). This is no longer recommended, and is likely to hurt functionality more than it improves performance.

IPVS proxy mode

This proxy mode is only available on Linux nodes.

In ipvs mode, kube-proxy watches Kubernetes Services and EndpointSlices, calls netlink interface to create IPVS rules accordingly and synchronizes IPVS rules with Kubernetes Services and EndpointSlices periodically. This control loop ensures that IPVS status matches the desired state. When accessing a Service, IPVS directs traffic to one of the backend Pods.

The IPVS proxy mode is based on netfilter hook function that is similar to iptables mode, but uses a hash table as the underlying data structure and works in the kernel space. That means kube-proxy in IPVS mode redirects traffic with lower latency than kube-proxy in iptables mode, with much better performance when synchronizing proxy rules. Compared to the other proxy modes, IPVS mode also supports a higher throughput of network traffic.

IPVS provides more options for balancing traffic to backend Pods; these are:

- rr (Round Robin): Traffic is equally distributed amongst the backing servers.
- wrr (Weighted Round Robin): Traffic is routed to the backing servers based on the weights of the servers. Servers with higher weights receive new connections and get more requests than servers with lower weights.
- lc (Least Connection): More traffic is assigned to servers with fewer active connections.
- wlc (Weighted Least Connection): More traffic is routed to servers with fewer connections relative to their weights, that is, connections divided by weight.
- lblc (Locality based Least Connection): Traffic for the same IP address is sent to the same backing server if the server is not overloaded and available; otherwise the traffic is sent to servers with fewer connections, and keep it for future assignment.
- lblcr (Locality Based Least Connection with Replication): Traffic for the same IP address is sent to the server with least connections. If all the backing servers are overloaded, it picks up one with fewer connections and add it to the target set. If the target set has not changed for the specified time, the most loaded server is removed from the set, in order to avoid high degree of replication.
- sh (Source Hashing): Traffic is sent to a backing server by looking up a statically assigned hash table based on the source IP addresses.
- dh (Destination Hashing): Traffic is sent to a backing server by looking up a statically assigned hash table based on their destination addresses.
- sed (Shortest Expected Delay): Traffic forwarded to a backing server with the shortest expected delay. The expected delay is $(C + 1) / U$ if sent to a server, where C is the number of connections on the server and U is the fixed service rate (weight) of the server.
- nq (Never Queue): Traffic is sent to an idle server if there is one, instead of waiting for a fast one; if all servers are busy, the algorithm falls back to the sed behavior.

Note:

To run kube-proxy in IPVS mode, you must make IPVS available on the node before starting kube-proxy.

When kube-proxy starts in IPVS proxy mode, it verifies whether IPVS kernel modules are available. If the IPVS kernel modules are not detected, then kube-proxy falls back to running in iptables proxy mode.

Virtual IP address mechanism for Services, using IPVS mode

kernelspace proxy mode

This proxy mode is only available on Windows nodes.

The kube-proxy configures packet filtering rules in the Windows *Virtual Filtering Platform* (VFP), an extension to Windows vSwitch. These rules process encapsulated packets within the node-level virtual networks, and rewrite packets so that the destination IP address (and layer 2 information) is correct for getting the packet routed to the correct destination. The Windows VFP is analogous to tools such as Linux nftables or iptables. The Windows VFP extends the *Hyper-V Switch*, which was initially implemented to support virtual machine networking.

When a Pod on a node sends traffic to a virtual IP address, and the kube-proxy selects a Pod on a different node as the load balancing target, the kernelspace proxy mode rewrites that packet to be destined to the target backend Pod. The Windows *Host Networking Service* (HNS) ensures that packet rewriting rules are configured so that the return traffic appears to come from the virtual IP address and not the specific backend Pod.

Direct server return for kernelspace mode

FEATURE STATE: Kubernetes v1.14 [alpha]

As an alternative to the basic operation, a node that hosts the backend Pod for a Service can apply the packet rewriting directly, rather than placing this burden on the node where the client Pod is running. This is called *direct server return*.

To use this, you must run kube-proxy with the --enable-dsr command line argument **and** enable the WinDSR [feature gate](#).

Direct server return also optimizes the case for Pod return traffic even when both Pods are running on the same node.

Session affinity

In these proxy models, the traffic bound for the Service's IP:Port is proxied to an appropriate backend without the clients knowing anything about Kubernetes or Services or Pods.

If you want to make sure that connections from a particular client are passed to the same Pod each time, you can select the session affinity based on the client's IP addresses by setting .spec.sessionAffinity to ClientIP for a Service (the default is None).

Session stickiness timeout

You can also set the maximum session sticky time by setting `.spec.sessionAffinityConfig.clientIP.timeoutSeconds` appropriately for a Service. (the default value is 10800, which works out to be 3 hours).

Note: On Windows, setting the maximum session sticky time for Services is not supported.

IP address assignment to Services

Unlike Pod IP addresses, which actually route to a fixed destination, Service IPs are not actually answered by a single host. Instead, kube-proxy uses packet processing logic (such as Linux iptables) to define *virtual* IP addresses which are transparently redirected as needed.

When clients connect to the VIP, their traffic is automatically transported to an appropriate endpoint. The environment variables and DNS for Services are actually populated in terms of the Service's virtual IP address (and port).

Avoiding collisions

One of the primary philosophies of Kubernetes is that you should not be exposed to situations that could cause your actions to fail through no fault of your own. For the design of the Service resource, this means not making you choose your own IP address if that choice might collide with someone else's choice. That is an isolation failure.

In order to allow you to choose an IP address for your Services, we must ensure that no two Services can collide. Kubernetes does that by allocating each Service its own IP address from within the `service-cluster-ip-range` CIDR range that is configured for the [API Server](#).

IP address allocation tracking

To ensure each Service receives a unique IP, an internal allocator atomically updates a global allocation map in [etcd](#) prior to creating each Service. The map object must exist in the registry for Services to get IP address assignments, otherwise creations will fail with a message indicating an IP address could not be allocated.

In the control plane, a background controller is responsible for creating that map (needed to support migrating from older versions of Kubernetes that used in-memory locking). Kubernetes also uses controllers to check for invalid assignments (e.g. due to administrator intervention) and for cleaning up allocated IP addresses that are no longer used by any Services.

FEATURE STATE: Kubernetes v1.27 [alpha]

If you enable the MultiCIDRSvcAllocator [feature gate](#) and the [networking.k8s.io/v1alpha1 API group](#), the control plane replaces the existing etcd allocator with a new one, using IPAddress objects instead of an internal global allocation map. The ClusterIP address associated to each Service will have a referenced IPAddress object.

The background controller is also replaced by a new one to handle the new IPAddress objects and the migration from the old allocator model.

One of the main benefits of the new allocator is that it removes the size limitations for the service-cluster-ip-range, there is no limitations for IPv4 and for IPv6 users can use masks equal or larger than /64 (previously it was /108).

Users now will be able to inspect the IP addresses assigned to their Services, and Kubernetes extensions such as the [Gateway](#) API, can use this new IPAddress object kind to enhance the Kubernetes networking capabilities, going beyond the limitations of the built-in Service API.

```
kubectl get services
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	2001:db8:1:2::1	<none>	443/TCP	3d1h

```
kubectl get ipaddresses
```

NAME	PARENTREF
2001:db8:1:2::1	services/default/kubernetes
2001:db8:1:2::a	services/kube-system/kube-dns

IP address ranges for Service virtual IP addresses

FEATURE STATE: Kubernetes v1.26 [stable]

Kubernetes divides the ClusterIP range into two bands, based on the size of the configured service-cluster-ip-range by using the following formula $\min(\max(16, \text{cidrSize} / 16), 256)$. That formula paraphrases as *never less than 16 or more than 256, with a graduated step function between them*.

Kubernetes prefers to allocate dynamic IP addresses to Services by choosing from the upper band, which means that if you want to assign a specific IP address to a type: ClusterIP Service, you should manually assign an IP address from the **lower** band. That approach reduces the risk of a conflict over allocation.

If you disable the ServiceIPStaticSubrange [feature gate](#) then Kubernetes uses a single shared pool for both manually and dynamically assigned IP addresses, that are used for type: ClusterIP Services.

Traffic policies

You can set the .spec.internalTrafficPolicy and .spec.externalTrafficPolicy fields to control how Kubernetes routes traffic to healthy (“ready”) backends.

Internal traffic policy

FEATURE STATE: Kubernetes v1.26 [stable]

You can set the .spec.internalTrafficPolicy field to control how traffic from internal sources is routed. Valid values are Cluster and Local. Set the field to Cluster to route internal traffic to all ready endpoints and Local to only route to ready node-local endpoints. If the traffic policy is Local and there are no node-local endpoints, traffic is dropped by kube-proxy.

External traffic policy

You can set the `.spec.externalTrafficPolicy` field to control how traffic from external sources is routed. Valid values are Cluster and Local. Set the field to Cluster to route external traffic to all ready endpoints and Local to only route to ready node-local endpoints. If the traffic policy is Local and there are no node-local endpoints, the kube-proxy does not forward any traffic for the relevant Service.

Traffic to terminating endpoints

FEATURE STATE: Kubernetes v1.28 [stable]

If the `ProxyTerminatingEndpoints` [feature gate](#) is enabled in kube-proxy and the traffic policy is Local, that node's kube-proxy uses a more complicated algorithm to select endpoints for a Service. With the feature enabled, kube-proxy checks if the node has local endpoints and whether or not all the local endpoints are marked as terminating. If there are local endpoints and **all** of them are terminating, then kube-proxy will forward traffic to those terminating endpoints. Otherwise, kube-proxy will always prefer forwarding traffic to endpoints that are not terminating.

This forwarding behavior for terminating endpoints exist to allow NodePort and LoadBalancer Services to gracefully drain connections when using `externalTrafficPolicy: Local`.

As a deployment goes through a rolling update, nodes backing a load balancer may transition from N to 0 replicas of that deployment. In some cases, external load balancers can send traffic to a node with 0 replicas in between health check probes. Routing traffic to terminating endpoints ensures that Node's that are scaling down Pods can gracefully receive and drain traffic to those terminating Pods. By the time the Pod completes termination, the external load balancer should have seen the node's health check failing and fully removed the node from the backend pool.

What's next

To learn more about Services, read [Connecting Applications with Services](#).

You can also:

- Read about [Services](#) as a concept
- Read about [Ingresses](#) as a concept
- Read the [API reference](#) for the Service API

Setup tools

[Kubeadm](#)

Kubeadm

Kubeadm is a tool built to provide kubeadm init and kubeadm join as best-practice "fast paths" for creating Kubernetes clusters.

kubeadm performs the actions necessary to get a minimum viable cluster up and running. By design, it cares only about bootstrapping, not about provisioning machines. Likewise, installing various nice-to-have addons, like the Kubernetes Dashboard, monitoring solutions, and cloud-specific addons, is not in scope.

Instead, we expect higher-level and more tailored tooling to be built on top of kubeadm, and ideally, using kubeadm as the basis of all deployments will make it easier to create conformant clusters.

How to install

To install kubeadm, see the [installation guide](#).

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node
- [kubeadm join](#) to bootstrap a Kubernetes worker node and join it to the cluster
- [kubeadm upgrade](#) to upgrade a Kubernetes cluster to a newer version
- [kubeadm config](#) if you initialized your cluster using kubeadm v1.7.x or lower, to configure your cluster for kubeadm upgrade
- [kubeadm token](#) to manage tokens for kubeadm join
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join
- [kubeadm certs](#) to manage Kubernetes certificates
- [kubeadm kubeconfig](#) to manage kubeconfig files
- [kubeadm version](#) to print the kubeadm version
- [kubeadm alpha](#) to preview a set of features made available for gathering feedback from the community

kubeadm init

This command initializes a Kubernetes control-plane node.

Run this command in order to set up the Kubernetes control plane

Synopsis

Run this command in order to set up the Kubernetes control plane

The "init" command executes the following phases:

preflight	Run pre-flight checks
certs	Certificate generation
/ca	Generate the self-signed Kubernetes CA to provision identities for other Kubernetes components

/apiserver	Generate the certificate for serving the Kubernetes API
/apiserver-kubelet-client	Generate the certificate for the API server to connect to kubelet
/front-proxy-ca	Generate the self-signed CA to provision identities for front proxy
/front-proxy-client	Generate the certificate for the front proxy client
/etcd-ca	Generate the self-signed CA to provision identities for etcd
/etcd-server	Generate the certificate for serving etcd
/etcd-peer	Generate the certificate for etcd nodes to communicate with each other
/etcd-healthcheck-client	Generate the certificate for liveness probes to healthcheck etcd
/apiserver-etcd-client	Generate the certificate the apiserver uses to access etcd
/sa	Generate a private key for signing service account tokens along with its public key
kubeconfig	Generate all kubeconfig files necessary to establish the control plane and the admin kubeconfig file
/admin	Generate a kubeconfig file for the admin to use and for kubeadm itself
/kubelet	Generate a kubeconfig file for the kubelet to use *only* for cluster bootstrapping purposes
/controller-manager	Generate a kubeconfig file for the controller manager to use
/scheduler	Generate a kubeconfig file for the scheduler to use
etcd	Generate static Pod manifest file for local etcd
/local	Generate the static Pod manifest file for a local, single-node local etcd instance
control-plane	Generate all static Pod manifest files necessary to establish the control plane
plane	
/apiserver	Generates the kube-apiserver static Pod manifest
/controller-manager	Generates the kube-controller-manager static Pod manifest
/scheduler	Generates the kube-scheduler static Pod manifest
kubelet-start	Write kubelet settings and (re)start the kubelet
upload-config	Upload the kubeadm and kubelet configuration to a ConfigMap
/kubeadm	Upload the kubeadm ClusterConfiguration to a ConfigMap
/kubelet	Upload the kubelet component config to a ConfigMap
upload-certs	Upload certificates to kubeadm-certs
mark-control-plane	Mark a node as a control-plane
bootstrap-token	Generates bootstrap tokens used to join a node to a cluster
kubelet-finalize	Updates settings relevant to the kubelet after TLS bootstrap
/experimental-cert-rotation	Enable kubelet client certificate rotation
addon	Install required addons for passing conformance tests
/coredns	Install the CoreDNS addon to a Kubernetes cluster
/kube-proxy	Install the kube-proxy addon to a Kubernetes cluster
show-join-command	Show the join command for control-plane and worker node

kubeadm init [flags]

Options

--apiserver-advertise-address string

The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.

--apiserver-bind-port int32 Default: 6443

Port for the API Server to bind to.

--apiserver-cert-extra-sans	strings
	Optional extra Subject Alternative Names (SANs) to use for the API Server serving certificate. Can be both IP addresses and DNS names.
--cert-dir	string Default: "/etc/kubernetes/pki"
	The path where to save and store the certificates.
--certificate-key	string
	Key used to encrypt the control-plane certificates in the kubeadm-certs Secret.
--config	string
	Path to a kubeadm configuration file.
--control-plane-endpoint	string
	Specify a stable IP address or DNS name for the control plane.
--cri-socket	string
	Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--dry-run	
	Don't apply any changes; just output what would be done.
--feature-gates	string
	A set of key=value pairs that describe feature gates for various features. Options are: EtcdLearnerMode=true false (ALPHA - default=false) PublicKeysECDSA=true false (ALPHA - default=false) RootlessControlPlane=true false (ALPHA - default=false) UpgradeAddonsBeforeControlPlane=true false (DEPRECATED - default=false)
-h, --help	
	help for init
--ignore-preflight-errors	strings
	A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
--image-repository	string Default: "registry.k8s.io"
	Choose a container registry to pull control plane images from
--kubernetes-version	string Default: "stable-1"

	Choose a specific Kubernetes version for the control plane.
--node-name string	Specify the node name.
--patches string	Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.
--pod-network-cidr string	Specify range of IP addresses for the pod network. If set, the control plane will automatically allocate CIDRs for every node.
--service-cidr string Default: "10.96.0.0/12"	Use alternative range of IP address for service VIPs.
--service-dns-domain string Default: "cluster.local"	Use alternative domain for services, e.g. "myorg.internal".
--skip-certificate-key-print	Don't print the key used to encrypt the control-plane certificates.
--skip-phases strings	List of phases to be skipped
--skip-token-print	Skip printing of the default bootstrap token generated by 'kubeadm init'.
--token string	The token to use for establishing bidirectional trust between nodes and control-plane nodes. The format is [a-z0-9]{6}.[a-z0-9]{16} - e.g. abcdef.0123456789abcdef
--token-ttl duration Default: 24h0m0s	The duration before the token is automatically deleted (e.g. 1s, 2m, 3h). If set to '0', the token will never expire

```
--upload-certs
```

Upload control-plane certificates to the kubeadm-certs Secret.

Options inherited from parent commands

```
--rootfs string
```

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Init workflow

kubeadm init bootstraps a Kubernetes control-plane node by executing the following steps:

1. Runs a series of pre-flight checks to validate the system state before making changes. Some checks only trigger warnings, others are considered errors and will exit kubeadm until the problem is corrected or the user specifies `--ignore-preflight-errors=<list-of-errors>`.
2. Generates a self-signed CA to set up identities for each component in the cluster. The user can provide their own CA cert and/or key by dropping it in the cert directory configured via `--cert-dir` (`/etc/kubernetes/pki` by default). The APIServer certs will have additional SAN entries for any `--apiserver-cert-extra-sans` arguments, lowercased if necessary.
3. Writes kubeconfig files in `/etc/kubernetes/` for the kubelet, the controller-manager and the scheduler to use to connect to the API server, each with its own identity, as well as an additional kubeconfig file for administration named `admin.conf`.
4. Generates static Pod manifests for the API server, controller-manager and scheduler. In case an external etcd is not provided, an additional static Pod manifest is generated for etcd.

Static Pod manifests are written to `/etc/kubernetes/manifests`; the kubelet watches this directory for Pods to create on startup.

Once control plane Pods are up and running, the kubeadm init sequence can continue.

5. Apply labels and taints to the control-plane node so that no additional workloads will run there.
6. Generates the token that additional nodes can use to register themselves with a control-plane in the future. Optionally, the user can provide a token via `--token`, as described in the [kubeadm token](#) docs.
7. Makes all the necessary configurations for allowing node joining with the [Bootstrap Tokens](#) and [TLS Bootstrap](#) mechanism:
 - Write a ConfigMap for making available all the information required for joining, and set up related RBAC access rules.

- Let Bootstrap Tokens access the CSR signing API.
- Configure auto-approval for new CSR requests.

See [kubeadm join](#) for additional info.

- Installs a DNS server (CoreDNS) and the kube-proxy addon components via the API server. In Kubernetes version 1.11 and later CoreDNS is the default DNS server. Please note that although the DNS server is deployed, it will not be scheduled until CNI is installed.

Warning: kube-dns usage with kubeadm is deprecated as of v1.18 and is removed in v1.21.

Using init phases with kubeadm

Kubeadm allows you to create a control-plane node in phases using the kubeadm init phase command.

To view the ordered list of phases and sub-phases you can call kubeadm init --help. The list will be located at the top of the help screen and each phase will have a description next to it. Note that by calling kubeadm init all of the phases and sub-phases will be executed in this exact order.

Some phases have unique flags, so if you want to have a look at the list of available options add --help, for example:

```
sudo kubeadm init phase control-plane controller-manager --help
```

You can also use --help to see the list of sub-phases for a certain parent phase:

```
sudo kubeadm init phase control-plane --help
```

kubeadm init also exposes a flag called --skip-phases that can be used to skip certain phases. The flag accepts a list of phase names and the names can be taken from the above ordered list.

An example:

```
sudo kubeadm init phase control-plane all --config=configfile.yaml
sudo kubeadm init phase etcd local --config=configfile.yaml
# you can now modify the control plane and etcd manifest files
sudo kubeadm init --skip-phases=control-plane,etcd --config=configfile.yaml
```

What this example would do is write the manifest files for the control plane and etcd in /etc/kubernetes/manifests based on the configuration in configfile.yaml. This allows you to modify the files and then skip these phases using --skip-phases. By calling the last command you will create a control plane node with the custom manifest files.

FEATURE STATE: Kubernetes v1.22 [beta]

Alternatively, you can use the skipPhases field under InitConfiguration.

Using kubeadm init with a configuration file

Caution: The config file is still considered beta and may change in future versions.

It's possible to configure kubeadm init with a configuration file instead of command line flags, and some more advanced features may only be available as configuration file options. This file is passed using the `--config` flag and it must contain a `ClusterConfiguration` structure and optionally more structures separated by `--config`. Mixing `--config` with others flags may not be allowed in some cases.

The default configuration can be printed out using the [kubeadm config print](#) command.

If your configuration is not using the latest version it is **recommended** that you migrate using the [kubeadm config migrate](#) command.

For more information on the fields and usage of the configuration you can navigate to our [API reference page](#).

Using kubeadm init with feature gates

Kubeadm supports a set of feature gates that are unique to kubeadm and can only be applied during cluster creation with kubeadm init. These features can control the behavior of the cluster. Feature gates are removed after a feature graduates to GA.

To pass a feature gate you can either use the `--feature-gates` flag for kubeadm init, or you can add items into the `featureGates` field when you pass a [configuration file](#) using `--config`.

Passing [feature gates for core Kubernetes components](#) directly to kubeadm is not supported. Instead, it is possible to pass them by [Customizing components with the kubeadm API](#).

List of feature gates:

kubeadm feature gates

Feature	Default	Alpha	Beta	GA
PublicKeysECDSA	false	1.19	-	-
RootlessControlPlane	false	1.22	-	-
EtcdLearnerMode	false	1.27	-	-

Note: Once a feature gate goes GA its value becomes locked to true by default.

Feature gate descriptions:

PublicKeysECDSA

Can be used to create a cluster that uses ECDSA certificates instead of the default RSA algorithm. Renewal of existing ECDSA certificates is also supported using kubeadm certs renew, but you cannot switch between the RSA and ECDSA algorithms on the fly or during upgrades.

RootlessControlPlane

Setting this flag configures the kubeadm deployed control plane component static Pod containers for kube-apiserver, kube-controller-manager, kube-scheduler and etcd to run as non-root users. If the flag is not set, those components run as root. You can change the value of this feature gate before you upgrade to a newer version of Kubernetes.

EtcdLearnerMode

With this feature gate enabled, when joining a new control plane node, a new etcd member will be created as a learner and promoted to a voting member only after the etcd data are fully aligned.

List of deprecated feature gates:

kubeadm deprecated feature gates

Feature	Default
UpgradeAddonsBeforeControlPlane	false

Feature gate descriptions:

UpgradeAddonsBeforeControlPlane

This is as a **disabled** feature gate that was introduced for Kubernetes v1.28, in order to allow reactivating a legacy and deprecated behavior during cluster upgrade. For kubeadm versions prior to v1.28, kubeadm upgrades cluster addons (including CoreDNS and kube-proxy) immediately during kubeadm upgrade apply, regardless of whether there are other control plane instances that have not been upgraded. This may cause compatibility problems. Since v1.28, kubeadm defaults to a mode that always checks whether all the control plane instances have been upgraded before starting to upgrade the addons. This behavior is applied to both kubeadm upgrade apply and kubeadm upgrade node. kubeadm determines whether a control plane instance has been upgraded by checking whether the image of the kube-apiserver Pod has been upgraded. You must perform control plane instances upgrade sequentially or at least ensure that the last control plane instance upgrade is not started until all the other control plane instances have been upgraded completely, and the addons upgrade will be performed after the last control plane instance is upgraded. The deprecated UpgradeAddonsBeforeControlPlane feature gate gives you a chance to keep the old upgrade behavior. You should not need this old behavior; if you do, you should consider changing your cluster or upgrade processes, as this feature gate will be removed in a future release.

List of removed feature gates:

kubeadm removed feature gates

Feature	Alpha	Beta	GA	Removed
UnversionedKubeletConfigMap	1.22	1.23	1.25	1.26
IPv6DualStack	1.16	1.21	1.23	1.24

Feature gate descriptions:

UnversionedKubeletConfigMap

This flag controls the name of the [ConfigMap](#) where kubeadm stores kubelet configuration data. With this flag not specified or set to true, the ConfigMap is named kubelet-config. If you set this flag to false, the name of the ConfigMap includes the major and minor version for Kubernetes (for example: kubelet-config-1.28). Kubeadm ensures that RBAC rules for reading and writing that ConfigMap are appropriate for the value you set. When kubeadm writes this ConfigMap (during kubeadm init or kubeadm upgrade apply), kubeadm respects the value of UnversionedKubeletConfigMap. When reading that ConfigMap (during kubeadm join, kubeadm reset, kubeadm upgrade ...), kubeadm attempts to use unversioned ConfigMap name first; if that does not succeed, kubeadm falls back to using the legacy (versioned) name for that ConfigMap.

IPv6DualStack

This flag helps to configure components dual stack when the feature is in progress. For more details on Kubernetes dual-stack support see [Dual-stack support with kubeadm](#).

Adding kube-proxy parameters

For information about kube-proxy parameters in the kubeadm configuration see:

- [kube-proxy reference](#)

For information about enabling IPVS mode with kubeadm see:

- [IPVS](#)

Passing custom flags to control plane components

For information about passing flags to control plane components see:

- [control-plane-flags](#)

Running kubeadm without an Internet connection

For running kubeadm without an Internet connection you have to pre-pull the required control-plane images.

You can list and pull the images using the kubeadm config images sub-command:

```
kubeadm config images list  
kubeadm config images pull
```

You can pass --config to the above commands with a [kubeadm configuration file](#) to control the kubernetesVersion and imageRepository fields.

All default registry.k8s.io images that kubeadm requires support multiple architectures.

Using custom images

By default, kubeadm pulls images from registry.k8s.io. If the requested Kubernetes version is a CI label (such as ci/latest) gcr.io/k8s-staging-ci-images is used.

You can override this behavior by using [kubeadm with a configuration file](#). Allowed customization are:

- To provide kubernetesVersion which affects the version of the images.
- To provide an alternative imageRepository to be used instead of registry.k8s.io.
- To provide a specific imageRepository and imageTag for etcd or CoreDNS.

Image paths between the default registry.k8s.io and a custom repository specified using imageRepository may differ for backwards compatibility reasons. For example, one image might have a subpath at registry.k8s.io/subpath/image, but be defaulted to my.customrepository.io/image when using a custom repository.

To ensure you push the images to your custom repository in paths that kubeadm can consume, you must:

- Pull images from the defaults paths at registry.k8s.io using kubeadm config images {list| pull}.
- Push images to the paths from kubeadm config images list --config=config.yaml, where config.yaml contains the custom imageRepository, and/or imageTag for etcd and CoreDNS.
- Pass the same config.yaml to kubeadm init.

Custom sandbox (pause) images

To set a custom image for these you need to configure this in your [container runtime](#) to use the image. Consult the documentation for your container runtime to find out how to change this setting; for selected container runtimes, you can also find advice within the [Container Runtimes](#) topic.

Uploading control-plane certificates to the cluster

By adding the flag --upload-certs to kubeadm init you can temporary upload the control-plane certificates to a Secret in the cluster. Please note that this Secret will expire automatically after 2 hours. The certificates are encrypted using a 32byte key that can be specified using --certificate-key. The same key can be used to download the certificates when additional control-plane nodes are joining, by passing --control-plane and --certificate-key to kubeadm join.

The following phase command can be used to re-upload the certificates after expiration:

```
kubeadm init phase upload-certs --upload-certs --config=SOME_YAML_FILE
```

Note: A predefined certificateKey can be provided in InitConfiguration when passing the [configuration file](#) with --config.

If a predefined certificate key is not passed to kubeadm init and kubeadm init phase upload-certs a new key will be generated automatically.

The following command can be used to generate a new key on demand:

```
kubeadm certs certificate-key
```

Certificate management with kubeadm

For detailed information on certificate management with kubeadm see [Certificate Management with kubeadm](#). The document includes information about using external CA, custom certificates and certificate renewal.

Managing the kubeadm drop-in file for the kubelet

The kubeadm package ships with a configuration file for running the kubelet by systemd. Note that the kubeadm CLI never touches this drop-in file. This drop-in file is part of the kubeadm DEB/RPM package.

For further information, see [Managing the kubeadm drop-in file for systemd](#).

Use kubeadm with CRI runtimes

By default kubeadm attempts to detect your container runtime. For more details on this detection, see the [kubeadm CRI installation guide](#).

Setting the node name

By default, kubeadm assigns a node name based on a machine's host address. You can override this setting with the `--node-name` flag. The flag passes the appropriate [--hostname-override](#) value to the kubelet.

Be aware that overriding the hostname can [interfere with cloud providers](#).

Automating kubeadm

Rather than copying the token you obtained from kubeadm init to each node, as in the [basic kubeadm tutorial](#), you can parallelize the token distribution for easier automation. To implement this automation, you must know the IP address that the control-plane node will have after it is started, or use a DNS name or an address of a load balancer.

1. Generate a token. This token must have the form <6 character string>.<16 character string>. More formally, it must match the regex: `[a-z0-9]{6}\.[a-z0-9]{16}`.

kubeadm can generate a token for you:

```
kubeadm token generate
```

2. Start both the control-plane node and the worker nodes concurrently with this token. As they come up they should find each other and form the cluster. The same `--token` argument can be used on both kubeadm init and kubeadm join.
3. Similar can be done for `--certificate-key` when joining additional control-plane nodes. The key can be generated using:

```
kubeadm certs certificate-key
```

Once the cluster is up, you can grab the admin credentials from the control-plane node at `/etc/kubernetes/admin.conf` and use that to talk to the cluster.

Note that this style of bootstrap has some relaxed security guarantees because it does not allow the root CA hash to be validated with `--discovery-token-ca-cert-hash` (since it's not generated when the nodes are provisioned). For details, see the [kubeadm join](#).

What's next

- [kubeadm init phase](#) to understand more about kubeadm init phases
- [kubeadm join](#) to bootstrap a Kubernetes worker node and join it to the cluster
- [kubeadm upgrade](#) to upgrade a Kubernetes cluster to a newer version
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join

kubeadm join

This command initializes a Kubernetes worker node and joins it to the cluster.

Run this on any machine you wish to join an existing cluster

Synopsis

When joining a kubeadm initialized cluster, we need to establish bidirectional trust. This is split into discovery (having the Node trust the Kubernetes Control Plane) and TLS bootstrap (having the Kubernetes Control Plane trust the Node).

There are 2 main schemes for discovery. The first is to use a shared token along with the IP address of the API server. The second is to provide a file - a subset of the standard kubeconfig file. The discovery/kubeconfig file supports token, client-go authentication plugins ("exec"), "tokenFile", and "authProvider". This file can be a local file or downloaded via an HTTPS URL. The forms are kubeadm join --discovery-token abcdef.1234567890abcdef 1.2.3.4:6443, kubeadm join --discovery-file path/to/file.conf, or kubeadm join --discovery-file https://url/file.conf. Only one form can be used. If the discovery information is loaded from a URL, HTTPS must be used. Also, in that case the host installed CA bundle is used to verify the connection.

If you use a shared token for discovery, you should also pass the --discovery-token-ca-cert-hash flag to validate the public key of the root certificate authority (CA) presented by the Kubernetes Control Plane. The value of this flag is specified as "<hash-type>:<hex-encoded-value>", where the supported hash type is "sha256". The hash is calculated over the bytes of the Subject Public Key Info (SPKI) object (as in RFC7469). This value is available in the output of "kubeadm init" or can be calculated using standard tools. The --discovery-token-ca-cert-hash flag may be repeated multiple times to allow more than one public key.

If you cannot know the CA public key hash ahead of time, you can pass the --discovery-token-unsafe-skip-ca-verification flag to disable this verification. This weakens the kubeadm security model since other nodes can potentially impersonate the Kubernetes Control Plane.

The TLS bootstrap mechanism is also driven via a shared token. This is used to temporarily authenticate with the Kubernetes Control Plane to submit a certificate signing request (CSR) for a locally created key pair. By default, kubeadm will set up the Kubernetes Control Plane to automatically approve these signing requests. This token is passed in with the --tls-bootstrap-token abcdef.1234567890abcdef flag.

Often times the same token is used for both parts. In this case, the --token flag can be used instead of specifying each token individually.

The "join [api-server-endpoint]" command executes the following phases:

preflight	Run join pre-flight checks
control-plane-prepare	Prepare the machine for serving a control plane
/download-certs	[EXPERIMENTAL] Download certificates shared among control-plane nodes from the kubeadm-certs Secret
/certs	Generate the certificates for the new control plane components
/kubeconfig	Generate the kubeconfig for the new control plane components
/control-plane	Generate the manifests for the new control plane components
kubelet-start	Write kubelet settings, certificates and (re)start the kubelet

```
control-plane-join  Join a machine as a control plane instance
/etcd           Add a new local etcd member
/update-status   Register the new control-plane node into the ClusterStatus maintained in
the kubeadm-config ConfigMap (DEPRECATED)
/mark-control-plane  Mark a node as a control-plane
```

```
kubeadm join [api-server-endpoint] [flags]
```

Options

--apiserver-advertise-address string
If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443
If the node should host a new control plane instance, the port for the API Server to bind to.
--certificate-key string
Use this key to decrypt the certificate secrets uploaded by init.
--config string
Path to a kubeadm configuration file.
--control-plane
Create a new control plane instance on this node
--cri-socket string
Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--discovery-file string
For file-based discovery, a file or URL from which to load cluster information.
--discovery-token string
For token-based discovery, the token used to validate cluster information fetched from the API server.
--discovery-token-ca-cert-hash strings
For token-based discovery, validate that the root CA public key matches this hash (format: "<type>:<value>").
--discovery-token-unsafe-skip-ca-verification

For token-based discovery, allow joining without --discovery-token-ca-cert-hash pinning.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for join
--ignore-preflight-errors strings
A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
--node-name string
Specify the node name.
--patches string
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.
--skip-phases strings
List of phases to be skipped
--tls-bootstrap-token string
Specify the token used to temporarily authenticate with the Kubernetes Control Plane while joining the node.
--token string
Use this token for both discovery-token and tls-bootstrap-token when those values are not provided.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

The join workflow

kubeadm join bootstraps a Kubernetes worker node or a control-plane node and adds it to the cluster. This action consists of the following steps for worker nodes:

1. kubeadm downloads necessary cluster information from the API server. By default, it uses the bootstrap token and the CA key hash to verify the authenticity of that data. The root CA can also be discovered directly via a file or URL.
2. Once the cluster information is known, kubelet can start the TLS bootstrapping process. The TLS bootstrap uses the shared token to temporarily authenticate with the Kubernetes API server to submit a certificate signing request (CSR); by default the control plane signs this CSR request automatically.
3. Finally, kubeadm configures the local kubelet to connect to the API server with the definitive identity assigned to the node.

For control-plane nodes additional steps are performed:

1. Downloading certificates shared among control-plane nodes from the cluster (if explicitly requested by the user).
2. Generating control-plane component manifests, certificates and kubeconfig.
3. Adding new local etcd member.

Using join phases with kubeadm

Kubeadm allows you join a node to the cluster in phases using kubeadm join phase.

To view the ordered list of phases and sub-phases you can call kubeadm join --help. The list will be located at the top of the help screen and each phase will have a description next to it. Note that by calling kubeadm join all of the phases and sub-phases will be executed in this exact order.

Some phases have unique flags, so if you want to have a look at the list of available options add --help, for example:

```
kubeadm join phase kubelet-start --help
```

Similar to the [kubeadm init phase](#) command, kubeadm join phase allows you to skip a list of phases using the --skip-phases flag.

For example:

```
sudo kubeadm join --skip-phases=preflight --config=config.yaml
```

FEATURE STATE: Kubernetes v1.22 [beta]

Alternatively, you can use the skipPhases field in JoinConfiguration.

Discovering what cluster CA to trust

The kubeadm discovery has several options, each with security tradeoffs. The right method for your environment depends on how you provision nodes and the security expectations you have about your network and node lifecycles.

Token-based discovery with CA pinning

This is the default mode in kubeadm. In this mode, kubeadm downloads the cluster configuration (including root CA) and validates it using the token as well as validating that the root CA public key matches the provided hash and that the API server certificate is valid under the root CA.

The CA key hash has the format sha256:<hex_encoded_hash>. By default, the hash value is printed at the end of the kubeadm init command or in the output from the kubeadm token create --print-join-command command. It is in a standard format (see [RFC7469](#)) and can also be calculated by 3rd party tools or provisioning systems. For example, using the OpenSSL CLI:

```
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>/dev/null | openssl dgst -sha256 -hex | sed 's/^.* //'
```

Example kubeadm join commands:

For worker nodes:

```
kubeadm join --discovery-token abcdef.1234567890abcdef --discovery-token-ca-cert-hash sha256:1234..cdef 1.2.3.4:6443
```

For control-plane nodes:

```
kubeadm join --discovery-token abcdef.1234567890abcdef --discovery-token-ca-cert-hash sha256:1234..cdef --control-plane 1.2.3.4:6443
```

You can also call join for a control-plane node with --certificate-key to copy certificates to this node, if the kubeadm init command was called with --upload-certs.

Advantages:

- Allows bootstrapping nodes to securely discover a root of trust for the control-plane node even if other worker nodes or the network are compromised.
- Convenient to execute manually since all of the information required fits into a single kubeadm join command.

Disadvantages:

- The CA hash is not normally known until the control-plane node has been provisioned, which can make it more difficult to build automated provisioning tools that use kubeadm. By generating your CA in beforehand, you may workaround this limitation.

Token-based discovery without CA pinning

This mode relies only on the symmetric token to sign (HMAC-SHA256) the discovery information that establishes the root of trust for the control-plane. To use the mode the joining

nodes must skip the hash validation of the CA public key, using --discovery-token-unsafe-skip-ca-verification. You should consider using one of the other modes if possible.

Example kubeadm join command:

```
kubeadm join --token abcdef.1234567890abcdef --discovery-token-unsafe-skip-ca-verification  
1.2.3.4:6443
```

Advantages:

- Still protects against many network-level attacks.
- The token can be generated ahead of time and shared with the control-plane node and worker nodes, which can then bootstrap in parallel without coordination. This allows it to be used in many provisioning scenarios.

Disadvantages:

- If an attacker is able to steal a bootstrap token via some vulnerability, they can use that token (along with network-level access) to impersonate the control-plane node to other bootstrapping nodes. This may or may not be an appropriate tradeoff in your environment.

File or HTTPS-based discovery

This provides an out-of-band way to establish a root of trust between the control-plane node and bootstrapping nodes. Consider using this mode if you are building automated provisioning using kubeadm. The format of the discovery file is a regular Kubernetes [kubeconfig](#) file.

In case the discovery file does not contain credentials, the TLS discovery token will be used.

Example kubeadm join commands:

- kubeadm join --discovery-file path/to/file.conf (local file)
- kubeadm join --discovery-file https://url/file.conf (remote HTTPS URL)

Advantages:

- Allows bootstrapping nodes to securely discover a root of trust for the control-plane node even if the network or other worker nodes are compromised.

Disadvantages:

- Requires that you have some way to carry the discovery information from the control-plane node to the bootstrapping nodes. If the discovery file contains credentials you must keep it secret and transfer it over a secure channel. This might be possible with your cloud provider or provisioning tool.

Use of custom kubelet credentials with kubeadm join

To allow kubeadm join to use predefined kubelet credentials and skip client TLS bootstrap and CSR approval for a new node:

1. From a working control plane node in the cluster that has /etc/kubernetes/pki/ca.key execute kubeadm kubeconfig user --org system:nodes --client-name system:node:\$NODE > kubelet.conf. \$NODE must be set to the name of the new node.
2. Modify the resulted kubelet.conf manually to adjust the cluster name and the server endpoint, or run kubeadm kubeconfig user --config (it accepts InitConfiguration).

If your cluster does not have the ca.key file, you must sign the embedded certificates in the kubelet.conf externally.

1. Copy the resulting kubelet.conf to /etc/kubernetes/kubelet.conf on the new node.
2. Execute kubeadm join with the flag --ignore-preflight-errors=FileAvailable--etc-kubernetes-kubelet.conf on the new node.

Securing your installation even more

The defaults for kubeadm may not work for everyone. This section documents how to tighten up a kubeadm installation at the cost of some usability.

Turning off auto-approval of node client certificates

By default, there is a CSR auto-applier enabled that basically approves any client certificate request for a kubelet when a Bootstrap Token was used when authenticating. If you don't want the cluster to automatically approve kubelet client certs, you can turn it off by executing this command:

```
kubectl delete clusterrolebinding kubeadm:node-autoapprove-bootstrap
```

After that, kubeadm join will block until the admin has manually approved the CSR in flight:

1. Using kubectl get csr, you can see that the original CSR is in the Pending state.

```
kubectl get csr
```

The output is similar to this:

NAME	AGE	REQUESTOR	CONDITION
node-csr-c69HXe7aYcqkS1bKmH4faEnHAWxn6i2bHZ2mD04jZyQ	18s		
system:bootstrap:878f07	Pending		

2. kubectl certificate approve allows the admin to approve CSR. This action tells a certificate signing controller to issue a certificate to the requestor with the attributes requested in the CSR.

```
kubectl certificate approve node-csr-c69HXe7aYcqkS1bKmH4faEnHAWxn6i2bHZ2mD04jZyQ
```

The output is similar to this:

```
certificatesigningrequest "node-csr-c69HXe7aYcqkS1bKmH4faEnHAWxn6i2bHZ2mD04jZyQ" approved
```

3. This would change the CRS resource to Active state.

```
kubectl get csr
```

The output is similar to this:

NAME	AGE	REQUESTOR	CONDITION
node-csr-c69HXe7aYcqkS1bKmH4faEnHAWxn6i2bHZ2mD04jZyQ	1m		
system:bootstrap:878f07	Approved,Issued		

This forces the workflow that kubeadm join will only succeed if kubectl certificate approve has been run.

Turning off public access to the cluster-info ConfigMap

In order to achieve the joining flow using the token as the only piece of validation information, a ConfigMap with some data needed for validation of the control-plane node's identity is exposed publicly by default. While there is no private data in this ConfigMap, some users might wish to turn it off regardless. Doing so will disable the ability to use the --discovery-token flag of the kubeadm join flow. Here are the steps to do so:

- Fetch the cluster-info file from the API Server:

```
kubectl -n kube-public get cm cluster-info -o jsonpath='{.data.kubeconfig}' | tee cluster-info.yaml
```

The output is similar to this:

```
apiVersion: v1
kind: Config
clusters:
- cluster:
  certificate-authority-data: <ca-cert>
  server: https://<ip>:<port>
  name: ""
contexts: []
current-context: ""
preferences: {}
users: []
```

- Use the cluster-info.yaml file as an argument to kubeadm join --discovery-file.
- Turn off public access to the cluster-info ConfigMap:

```
kubectl -n kube-public delete rolebinding kubeadm:bootstrap-signer-clusterinfo
```

These commands should be run after kubeadm init but before kubeadm join.

Using kubeadm join with a configuration file

Caution: The config file is still considered beta and may change in future versions.

It's possible to configure kubeadm join with a configuration file instead of command line flags, and some more advanced features may only be available as configuration file options. This file is passed using the --config flag and it must contain a JoinConfiguration structure. Mixing --config with others flags may not be allowed in some cases.

The default configuration can be printed out using the [kubeadm config print](#) command.

If your configuration is not using the latest version it is **recommended** that you migrate using the [kubeadm config migrate](#) command.

For more information on the fields and usage of the configuration you can navigate to our [API reference](#).

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node.
- [kubeadm token](#) to manage tokens for kubeadm join.
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join.

kubeadm upgrade

kubeadm upgrade is a user-friendly command that wraps complex upgrading logic behind one command, with support for both planning an upgrade and actually performing it.

kubeadm upgrade guidance

The steps for performing an upgrade using kubeadm are outlined in [this document](#). For older versions of kubeadm, please refer to older documentation sets of the Kubernetes website.

You can use kubeadm upgrade diff to see the changes that would be applied to static pod manifests.

In Kubernetes v1.15.0 and later, kubeadm upgrade apply and kubeadm upgrade node will also automatically renew the kubeadm managed certificates on this node, including those stored in kubeconfig files. To opt-out, it is possible to pass the flag --certificate-renewal=false. For more details about certificate renewal see the [certificate management documentation](#).

Note: The commands kubeadm upgrade apply and kubeadm upgrade plan have a legacy --config flag which makes it possible to reconfigure the cluster, while performing planning or upgrade of that particular control-plane node. Please be aware that the upgrade workflow was not designed for this scenario and there are reports of unexpected results.

kubeadm upgrade plan

Check which versions are available to upgrade to and validate whether your current cluster is upgradeable. To skip the internet check, pass in the optional [version] parameter

Synopsis

Check which versions are available to upgrade to and validate whether your current cluster is upgradeable. To skip the internet check, pass in the optional [version] parameter

```
kubeadm upgrade plan [version] [flags]
```

Options

--allow-experimental-upgrades	Show unstable versions of Kubernetes as an upgrade alternative and allow upgrading to an alpha/beta/release candidate versions of Kubernetes.
--allow-release-candidate-upgrades	Show release candidate versions of Kubernetes as an upgrade alternative and allow upgrading to a release candidate versions of Kubernetes.
--config string	Path to a kubeadm configuration file.
--feature-gates string	A set of key=value pairs that describe feature gates for various features. Options are: EtcLearnerMode=true false (ALPHA - default=false) PublicKeysECDSA=true false (ALPHA - default=false) RootlessControlPlane=true false (ALPHA - default=false) UpgradeAddonsBeforeControlPlane=true false (DEPRECATED - default=false)
-h, --help	help for plan
--ignore-preflight-errors strings	A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
--kubeconfig string Default: "/etc/kubernetes/admin.conf"	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
-o, --output string Default: "text"	EXPERIMENTAL: Output format. One of: text json yaml.
--print-config	

Specifies whether the configuration file that will be used in the upgrade should be printed or not.

--show-managed-fields

If true, keep the managedFields when printing objects in JSON or YAML format.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm upgrade apply

Upgrade your Kubernetes cluster to the specified version

Synopsis

Upgrade your Kubernetes cluster to the specified version

kubeadm upgrade apply [version]

Options

--allow-experimental-upgrades

Show unstable versions of Kubernetes as an upgrade alternative and allow upgrading to an alpha/beta/release candidate versions of Kubernetes.

--allow-release-candidate-upgrades

Show release candidate versions of Kubernetes as an upgrade alternative and allow upgrading to a release candidate versions of Kubernetes.

--certificate-renewal Default: true

Perform the renewal of certificates used by component changed during upgrades.

--config string

Path to a kubeadm configuration file.

--dry-run

Do not change any state, just output what actions would be performed.

--etcd-upgrade Default: true

Perform the upgrade of etcd.
--feature-gates string
A set of key=value pairs that describe feature gates for various features. Options are: EtcdLearnerMode=true false (ALPHA - default=false) PublicKeysECDSA=true false (ALPHA - default=false) RootlessControlPlane=true false (ALPHA - default=false) UpgradeAddonsBeforeControlPlane=true false (DEPRECATED - default=false)
-f, --force
Force upgrading although some requirements might not be met. This also implies non-interactive mode.
-h, --help
help for apply
--ignore-preflight-errors strings
A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--patches string
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.
--print-config
Specifies whether the configuration file that will be used in the upgrade should be printed or not.
-y, --yes
Perform the upgrade and do not prompt for confirmation (non-interactive mode).

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm upgrade diff

Show what differences would be applied to existing static pod manifests. See also: kubeadm upgrade apply --dry-run

Synopsis

Show what differences would be applied to existing static pod manifests. See also: kubeadm upgrade apply --dry-run

```
kubeadm upgrade diff [version] [flags]
```

Options

--api-server-manifest string Default: "/etc/kubernetes/manifests/kube-apiserver.yaml"
path to API server manifest
--config string
Path to a kubeadm configuration file.
-c, --context-lines int Default: 3
How many lines of context in the diff
--controller-manager-manifest string Default: "/etc/kubernetes/manifests/kube-controller-manager.yaml"
path to controller manifest
-h, --help
help for diff
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--scheduler-manifest string Default: "/etc/kubernetes/manifests/kube-scheduler.yaml"
path to scheduler manifest

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm upgrade node

Upgrade commands for a node in the cluster

Synopsis

Upgrade commands for a node in the cluster

The "node" command executes the following phases:

preflight Run upgrade node pre-flight checks
control-plane Upgrade the control plane instance deployed on this node, if any
kubelet-config Upgrade the kubelet configuration for this node

kubeadm upgrade node [flags]

Options

--certificate-renewal Default: true
Perform the renewal of certificates used by component changed during upgrades.
--dry-run
Do not change any state, just output the actions that would be performed.
--etcd-upgrade Default: true
Perform the upgrade of etcd.
-h, --help
help for node
--ignore-preflight-errors strings
A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--patches string	Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.
--skip-phases strings	List of phases to be skipped

Options inherited from parent commands

--rootfs string	[EXPERIMENTAL] The path to the 'real' host root filesystem.
-----------------	---

What's next

- [kubeadm config](#) if you initialized your cluster using kubeadm v1.7.x or lower, to configure your cluster for kubeadm upgrade

kubeadm config

During kubeadm init, kubeadm uploads the ClusterConfiguration object to your cluster in a ConfigMap called kubeadm-config in the kube-system namespace. This configuration is then read during kubeadm join, kubeadm reset and kubeadm upgrade.

You can use kubeadm config print to print the default static configuration that kubeadm uses for kubeadm init and kubeadm join.

Note: The output of the command is meant to serve as an example. You must manually edit the output of this command to adapt to your setup. Remove the fields that you are not certain about and kubeadm will try to default them on runtime by examining the host.

For more information on init and join navigate to [Using kubeadm init with a configuration file](#) or [Using kubeadm join with a configuration file](#).

For more information on using the kubeadm configuration API navigate to [Customizing components with the kubeadm API](#).

You can use kubeadm config migrate to convert your old configuration files that contain a deprecated API version to a newer, supported API version.

kubeadm config validate can be used for validating a configuration file.

kubeadm config images list and kubeadm config images pull can be used to list and pull the images that kubeadm requires.

kubeadm config print

Print configuration

Synopsis

This command prints configurations for subcommands provided. For details, see: <https://pkg.go.dev/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm#section-directories>

```
kubeadm config print [flags]
```

Options

-h, --help
help for print

Options inherited from parent commands

--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm config print init-defaults

Print default init configuration, that can be used for 'kubeadm init'

Synopsis

This command prints objects such as the default init configuration that is used for 'kubeadm init'.

Note that sensitive values like the Bootstrap Token fields are replaced with placeholder values like "abcdef.0123456789abcdef" in order to pass validation but not perform the real computation for creating a token.

```
kubeadm config print init-defaults [flags]
```

Options

--component-configs strings	A comma-separated list for component config API objects to print the default values for. Available values: [KubeProxyConfiguration KubeletConfiguration]. If this flag is not set, no component configs will be printed.
-h, --help	help for init-defaults

Options inherited from parent commands

--kubeconfig string Default: "/etc/kubernetes/admin.conf"	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--rootfs string	[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm config print join-defaults

Print default join configuration, that can be used for 'kubeadm join'

Synopsis

This command prints objects such as the default join configuration that is used for 'kubeadm join'.

Note that sensitive values like the Bootstrap Token fields are replaced with placeholder values like "abcdef.0123456789abcdef" in order to pass validation but not perform the real computation for creating a token.

```
kubeadm config print join-defaults [flags]
```

Options

--component-configs strings	A comma-separated list for component config API objects to print the default values for. Available values: [KubeProxyConfiguration KubeletConfiguration]. If this flag is not set, no component configs will be printed.
-h, --help	

```
help for join-defaults
```

Options inherited from parent commands

--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
---------------------	---------------------------------------

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm config migrate

Read an older version of the kubeadm configuration API types from a file, and output the similar config object for the newer version

Synopsis

This command lets you convert configuration objects of older versions to the latest supported version, locally in the CLI tool without ever touching anything in the cluster. In this version of kubeadm, the following API versions are supported:

- kubeadm.k8s.io/v1beta3

Further, kubeadm can only write out config of version "kubeadm.k8s.io/v1beta3", but read both types. So regardless of what version you pass to the --old-config parameter here, the API object will be read, deserialized, defaulted, converted, validated, and re-serialized when written to stdout or --new-config if specified.

In other words, the output of this command is what kubeadm actually would read internally if you submitted this file to "kubeadm init"

```
kubeadm config migrate [flags]
```

Options

--allow-experimental-api

Allow migration to experimental, unreleased APIs.

-h, --help

help for migrate

--new-config string

Path to the resulting equivalent kubeadm config file using the new API version. Optional, if not specified output will be sent to STDOUT.

--old-config string

Path to the kubeadm config file that is using an old API version and should be converted. This flag is mandatory.

Options inherited from parent commands

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm config validate

Read a file containing the kubeadm configuration API and report any validation problems

Synopsis

This command lets you validate a kubeadm configuration API file and report any warnings and errors. If there are no errors the exit status will be zero, otherwise it will be non-zero. Any unmarshaling problems such as unknown API fields will trigger errors. Unknown API versions and fields with invalid values will also trigger errors. Any other errors or warnings may be reported depending on contents of the input file.

In this version of kubeadm, the following API versions are supported:

- kubeadm.k8s.io/v1beta3

```
kubeadm config validate [flags]
```

Options

--allow-experimental-api

Allow validation of experimental, unreleased APIs.
--

--config string

Path to a kubeadm configuration file.

-h, --help

help for validate

Options inherited from parent commands

--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
---------------------	---------------------------------------

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm config images list

Print a list of images kubeadm will use. The configuration file is used in case any images or image repositories are customized

Synopsis

Print a list of images kubeadm will use. The configuration file is used in case any images or image repositories are customized

kubeadm config images list [flags]

Options

--allow-missing-template-keys	Default: true
-------------------------------	---------------

If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.

--config string

Path to a kubeadm configuration file.

-o, --experimental-output string	Default: "text"
----------------------------------	-----------------

Output format. One of: text|json|yaml|go-template|go-template-file|template|templatefile|jsonpath|jsonpath-as-json|jsonpath-file.

--feature-gates string

A set of key=value pairs that describe feature gates for various features. Options are:
EtcdLearnerMode=true|false (ALPHA - default=false)
PublicKeysECDSA=true|false (ALPHA - default=false)
RootlessControlPlane=true|false (ALPHA - default=false)
UpgradeAddonsBeforeControlPlane=true|false (DEPRECATED - default=false)

-h, --help

help for list

--image-repository string Default: "registry.k8s.io"

Choose a container registry to pull control plane images from

--kubernetes-version string Default: "stable-1"

Choose a specific Kubernetes version for the control plane.

--show-managed-fields

If true, keep the managedFields when printing objects in JSON or YAML format.

Options inherited from parent commands

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm config images pull

Pull images used by kubeadm

Synopsis

Pull images used by kubeadm

kubeadm config images pull [flags]

Options

--config string

Path to a kubeadm configuration file.
--cri-socket string
Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--feature-gates string
A set of key=value pairs that describe feature gates for various features. Options are: EtcdLearnerMode=true false (ALPHA - default=false) PublicKeysECDSA=true false (ALPHA - default=false) RootlessControlPlane=true false (ALPHA - default=false) UpgradeAddonsBeforeControlPlane=true false (DEPRECATED - default=false)
-h, --help
help for pull
--image-repository string Default: "registry.k8s.io"
Choose a container registry to pull control plane images from
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

What's next

- [kubeadm upgrade](#) to upgrade a Kubernetes cluster to a newer version

kubeadm reset

Performs a best effort revert of changes made by kubeadm init or kubeadm join.

Performs a best effort revert of changes made to this host by 'kubeadm init' or 'kubeadm join'

Synopsis

Performs a best effort revert of changes made to this host by 'kubeadm init' or 'kubeadm join'

The "reset" command executes the following phases:

preflight	Run reset pre-flight checks
remove-etcd-member	Remove a local etcd member.
cleanup-node	Run cleanup node.

kubeadm reset [flags]

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path to the directory where the certificates are stored. If specified, clean this directory.	
--cleanup-tmp-dir	
Cleanup the "/etc/kubernetes/tmp" directory	
--config string	
Path to a kubeadm configuration file.	
--cri-socket string	
Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.	
--dry-run	
Don't apply any changes; just output what would be done.	
-f, --force	
Reset the node without prompting for confirmation.	
-h, --help	
help for reset	
--ignore-preflight-errors strings	
A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.	
--kubeconfig string	Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--skip-phases strings

List of phases to be skipped

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Reset workflow

kubeadm reset is responsible for cleaning up a node local file system from files that were created using the kubeadm init or kubeadm join commands. For control-plane nodes reset also removes the local stacked etcd member of this node from the etcd cluster.

kubeadm reset phase can be used to execute the separate phases of the above workflow. To skip a list of phases you can use the --skip-phases flag, which works in a similar way to the kubeadm join and kubeadm init phase runners.

External etcd clean up

kubeadm reset will not delete any etcd data if external etcd is used. This means that if you run kubeadm init again using the same etcd endpoints, you will see state from previous clusters.

To wipe etcd data it is recommended you use a client like etcdctl, such as:

```
etcdctl del "" --prefix
```

See the [etcd documentation](#) for more information.

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node
- [kubeadm join](#) to bootstrap a Kubernetes worker node and join it to the cluster

kubeadm token

Bootstrap tokens are used for establishing bidirectional trust between a node joining the cluster and a control-plane node, as described in [authenticating with bootstrap tokens](#).

kubeadm init creates an initial token with a 24-hour TTL. The following commands allow you to manage such a token and also to create and manage new ones.

kubeadm token create

Create bootstrap tokens on the server

Synopsis

This command will create a bootstrap token for you. You can specify the usages for this token, the "time to live" and an optional human friendly description.

The [token] is the actual token to write. This should be a securely generated random token of the form "[a-z0-9]{6}.[a-z0-9]{16}". If no [token] is given, kubeadm will generate a random token instead.

```
kubeadm token create [token]
```

Options

--certificate-key string	
	When used together with '--print-join-command', print the full 'kubeadm join' flag needed to join the cluster as a control-plane. To create a new certificate key you must use 'kubeadm init phase upload-certs --upload-certs'.
--config string	Path to a kubeadm configuration file.
--description string	A human friendly description of how this token is used.
--groups strings Default: "system:bootstrappers:kubeadm:default-node-token"	Extra groups that this token will authenticate as when used for authentication. Must match "\Asystem:bootstrappers:[a-z0-9:-]{0,255}[a-z0-9]\z"
-h, --help	help for create
--print-join-command	Instead of printing only the token, print the full 'kubeadm join' flag needed to join the cluster using the token.
--ttl duration Default: 24h0m0s	The duration before the token is automatically deleted (e.g. 1s, 2m, 3h). If set to '0', the token will never expire
--usages strings Default: "signing,authentication"	

Describes the ways in which this token can be used. You can pass --usages multiple times or provide a comma separated list of options. Valid options: [signing,authentication]

Options inherited from parent commands

--dry-run

Whether to enable dry-run mode or not

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm token delete

Delete bootstrap tokens on the server

Synopsis

This command will delete a list of bootstrap tokens for you.

The [token-value] is the full Token of the form "[a-z0-9]{6}.[a-z0-9]{16}" or the Token ID of the form "[a-z0-9]{6}" to delete.

kubeadm token delete [token-value] ...

Options

-h, --help

help for delete

Options inherited from parent commands

--dry-run

Whether to enable dry-run mode or not

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm token generate

Generate and print a bootstrap token, but do not create it on the server

Synopsis

This command will print out a randomly-generated bootstrap token that can be used with the "init" and "join" commands.

You don't have to use this command in order to generate a token. You can do so yourself as long as it is in the format "[a-z0-9]{6}.[a-z0-9]{16}". This command is provided for convenience to generate tokens in the given format.

You can also use "kubeadm init" without specifying a token and it will generate and print one for you.

kubeadm token generate [flags]

Options

-h, --help

help for generate

Options inherited from parent commands

--dry-run

Whether to enable dry-run mode or not

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm token list

List bootstrap tokens on the server

Synopsis

This command will list all bootstrap tokens for you.

```
kubeadm token list [flags]
```

Options

--allow-missing-template-keys	Default: true
If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.	
-o, --experimental-output	string Default: "text"
Output format. One of: text json yaml go-template go-template-file template templatefile jsonpath jsonpath-as-json jsonpath-file.	
-h, --help	
help for list	
--show-managed-fields	
If true, keep the managedFields when printing objects in JSON or YAML format.	

Options inherited from parent commands

--dry-run	
Whether to enable dry-run mode or not	
--kubeconfig	string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.	
--rootfs	string
[EXPERIMENTAL] The path to the 'real' host root filesystem.	

What's next

- [kubeadm join](#) to bootstrap a Kubernetes worker node and join it to the cluster

kubeadm version

This command prints the version of kubeadm.

Print the version of kubeadm

Synopsis

Print the version of kubeadm

```
kubeadm version [flags]
```

Options

-h, --help
help for version
-o, --output string
Output format; available options are 'yaml', 'json' and 'short'

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm alpha

Caution: kubeadm alpha provides a preview of a set of features made available for gathering feedback from the community. Please try it out and give us feedback!

Currently there are no experimental commands under kubeadm alpha.

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node
- [kubeadm join](#) to connect a node to the cluster
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join

kubeadm certs

kubeadm certs provides utilities for managing certificates. For more details on how these commands can be used, see [Certificate Management with kubeadm](#).

kubeadm certs

A collection of operations for operating Kubernetes certificates.

- [overview](#)

Commands related to handling kubernetes certificates

Synopsis

Commands related to handling kubernetes certificates

kubeadm certs [flags]

Options

-h, --help
help for certs

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm certs renew

You can renew all Kubernetes certificates using the all subcommand or renew them selectively. For more details see [Manual certificate renewal](#).

- [renew](#)
- [all](#)
- [admin.conf](#)
- [apiserver-etcd-client](#)
- [apiserver-kubelet-client](#)
- [apiserver](#)
- [controller-manager.conf](#)
- [etcd-healthcheck-client](#)
- [etcd-peer](#)
- [etcd-server](#)
- [front-proxy-client](#)
- [scheduler.conf](#)

Renew certificates for a Kubernetes cluster

Synopsis

This command is not meant to be run on its own. See list of available subcommands.

```
kubeadm certs renew [flags]
```

Options

-h, --help
help for renew

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew all available certificates

Synopsis

Renew all known certificates necessary to run the control plane. Renewals are run unconditionally, regardless of expiration date. Renewals can also be run individually for more control.

```
kubeadm certs renew all [flags]
```

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save the certificates
--config string
Path to a kubeadm configuration file.
-h, --help
help for all
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate embedded in the kubeconfig file for the admin to use and for kubeadm itself

Synopsis

Renew the certificate embedded in the kubeconfig file for the admin to use and for kubeadm itself.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

```
kubeadm certs renew admin.conf [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save the certificates	
--config string	
Path to a kubeadm configuration file.	
-h, --help	
help for admin.conf	
--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate the apiserver uses to access etcd

Synopsis

Renew the certificate the apiserver uses to access etcd.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

```
kubeadm certs renew apiserver-etcd-client [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save the certificates	
--config string	
Path to a kubeadm configuration file.	
-h, --help	
help for apiserver-etcd-client	
--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate for the API server to connect to kubelet

Synopsis

Renew the certificate for the API server to connect to kubelet.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

```
kubeadm certs renew apiserver-kubelet-client [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save the certificates	
--config string	
Path to a kubeadm configuration file.	
-h, --help	
help for apiserver-kubelet-client	
--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate for serving the Kubernetes API

Synopsis

Renew the certificate for serving the Kubernetes API.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

```
kubeadm certs renew apiserver [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save the certificates	
--config string	
Path to a kubeadm configuration file.	
-h, --help	
help for apiserver	
--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate embedded in the kubeconfig file for the controller manager to use

Synopsis

Renew the certificate embedded in the kubeconfig file for the controller manager to use.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

```
kubeadm certs renew controller-manager.conf [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save the certificates	
--config string	
Path to a kubeadm configuration file.	
-h, --help	
help for controller-manager.conf	
--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate for liveness probes to healthcheck etcd

Synopsis

Renew the certificate for liveness probes to healthcheck etcd.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

```
kubeadm certs renew etcd-healthcheck-client [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save the certificates	

--config string
Path to a kubeadm configuration file.
-h, --help
help for etcd-healthcheck-client
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate for etcd nodes to communicate with each other

Synopsis

Renew the certificate for etcd nodes to communicate with each other.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

kubeadm certs renew etcd-peer [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save the certificates
--config string
Path to a kubeadm configuration file.
-h, --help

help for etcd-peer

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate for serving etcd

Synopsis

Renew the certificate for serving etcd.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

kubeadm certs renew etcd-server [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"

The path where to save the certificates

--config string

Path to a kubeadm configuration file.

-h, --help

help for etcd-server

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate for the front proxy client

Synopsis

Renew the certificate for the front proxy client.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

kubeadm certs renew front-proxy-client [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"

The path where to save the certificates

--config string

Path to a kubeadm configuration file.

-h, --help

help for front-proxy-client

--kubeconfig string Default: "/etc/kubernetes/admin.conf"

The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Renew the certificate embedded in the kubeconfig file for the scheduler manager to use

Synopsis

Renew the certificate embedded in the kubeconfig file for the scheduler manager to use.

Renewals run unconditionally, regardless of certificate expiration date; extra attributes such as SANs will be based on the existing file/certificates, there is no need to resupply them.

Renewal by default tries to use the certificate authority in the local PKI managed by kubeadm; as alternative it is possible to use K8s certificate API for certificate renewal, or as a last option, to generate a CSR request.

After renewal, in order to make changes effective, is required to restart control-plane components and eventually re-distribute the renewed certificate in case the file is used elsewhere.

```
kubeadm certs renew scheduler.conf [flags]
```

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save the certificates
--config string
Path to a kubeadm configuration file.
-h, --help
help for scheduler.conf
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm certs certificate-key

This command can be used to generate a new control-plane certificate key. The key can be passed as `--certificate-key` to [kubeadm init](#) and [kubeadm join](#) to enable the automatic copy of certificates when joining additional control-plane nodes.

- [certificate-key](#)

Generate certificate keys

Synopsis

This command will print out a secure randomly-generated certificate key that can be used with the "init" command.

You can also use "kubeadm init --upload-certs" without specifying a certificate key and it will generate and print one for you.

```
kubeadm certs certificate-key [flags]
```

Options

-h, --help
help for certificate-key

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm certs check-expiration

This command checks expiration for the certificates in the local PKI managed by kubeadm. For more details see [Check certificate expiration](#).

- [check-expiration](#)

Check certificates expiration for a Kubernetes cluster

Synopsis

Checks expiration for the certificates in the local PKI managed by kubeadm.

```
kubeadm certs check-expiration [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
	The path where to save the certificates
--config string	
	Path to a kubeadm configuration file.
-h, --help	
	help for check-expiration
--kubeconfig string	Default: "/etc/kubernetes/admin.conf"
	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm certs generate-csr

This command can be used to generate keys and CSRs for all control-plane certificates and kubeconfig files. The user can then sign the CSRs with a CA of their choice.

- [generate-csr](#)

Generate keys and certificate signing requests

Synopsis

Generates keys and certificate signing requests (CSRs) for all the certificates required to run the control plane. This command also generates partial kubeconfig files with private key data in the "users > user > client-key-data" field, and for each kubeconfig file an accompanying ".csr" file is created.

This command is designed for use in [Kubeadm External CA Mode](#). It generates CSRs which you can then submit to your external certificate authority for signing.

The PEM encoded signed certificates should then be saved alongside the key files, using ".crt" as the file extension, or in the case of kubeconfig files, the PEM encoded signed certificate should be base64 encoded and added to the kubeconfig file in the "users > user > client-certificate-data" field.

kubeadm certs generate-csr [flags]

Examples

```
# The following command will generate keys and CSRs for all control-plane certificates and  
kubeconfig files:  
kubeadm certs generate-csr --kubeconfig-dir /tmp/etc-k8s --cert-dir /tmp/etc-k8s/pki
```

Options

--cert-dir string	
	The path where to save the certificates
--config string	
	Path to a kubeadm configuration file.
-h, --help	
	help for generate-csr
--kubeconfig-dir string Default: "/etc/kubernetes"	
	The path where to save the kubeconfig file.

Options inherited from parent commands

--rootfs string	
	[EXPERIMENTAL] The path to the 'real' host root filesystem.

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node
- [kubeadm join](#) to connect a node to the cluster
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join

kubeadm init phase

kubeadm init phase enables you to invoke atomic steps of the bootstrap process. Hence, you can let kubeadm do some of the work and you can fill in the gaps if you wish to apply customization.

kubeadm init phase is consistent with the [kubeadm init workflow](#), and behind the scene both use the same code.

kubeadm init phase preflight

Using this command you can execute preflight checks on a control-plane node.

- [preflight](#)

Run pre-flight checks

Synopsis

Run pre-flight checks for kubeadm init.

```
kubeadm init phase preflight [flags]
```

Examples

```
# Run pre-flight checks for kubeadm init using a config file.  
kubeadm init phase preflight --config kubeadm-config.yaml
```

Options

--config string
Path to a kubeadm configuration file.
--cri-socket string
Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for preflight
--ignore-preflight-errors strings
A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase kubelet-start

This phase will write the kubelet configuration file and environment file and then start the kubelet.

- [kubelet-start](#)

Write kubelet settings and (re)start the kubelet

Synopsis

Write a file with KubeletConfiguration and an environment file with node specific kubelet settings, and then (re)start kubelet.

```
kubeadm init phase kubelet-start [flags]
```

Examples

```
# Writes a dynamic environment file with kubelet flags from a InitConfiguration file.  
kubeadm init phase kubelet-start --config config.yaml
```

Options

--config string
Path to a kubeadm configuration file.
--cri-socket string
Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for kubelet-start
--node-name string
Specify the node name.
--patches string
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json"

or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase certs

Can be used to create all required certificates by kubeadm.

- [certs](#)
- [all](#)
- [ca](#)
- [apiserver](#)
- [apiserver-kubelet-client](#)
- [front-proxy-ca](#)
- [front-proxy-client](#)
- [etcd-ca](#)
- [etcd-server](#)
- [etcd-peer](#)
- [healthcheck-client](#)
- [apiserver-etcd-client](#)
- [sa](#)

Certificate generation

Synopsis

This command is not meant to be run on its own. See list of available subcommands.

```
kubeadm init phase certs [flags]
```

Options

-h, --help

help for certs

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate all certificates

Synopsis

Generate all certificates

```
kubeadm init phase certs all [flags]
```

Options

--apiserver-advertise-address string	The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-cert-extra-sans strings	Optional extra Subject Alternative Names (SANs) to use for the API Server serving certificate. Can be both IP addresses and DNS names.
--cert-dir string Default: "/etc/kubernetes/pki"	The path where to save and store the certificates.
--config string	Path to a kubeadm configuration file.
--control-plane-endpoint string	Specify a stable IP address or DNS name for the control plane.
--dry-run	Don't apply any changes; just output what would be done.
-h, --help	help for all
--kubernetes-version string Default: "stable-1"	Choose a specific Kubernetes version for the control plane.
--service-cidr string Default: "10.96.0.0/12"	Use alternative range of IP address for service VIPs.
--service-dns-domain string Default: "cluster.local"	Use alternative domain for services, e.g. "myorg.internal".

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the self-signed Kubernetes CA to provision identities for other Kubernetes components

Synopsis

Generate the self-signed Kubernetes CA to provision identities for other Kubernetes components, and save them into ca.crt and ca.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

```
kubeadm init phase certs ca [flags]
```

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for ca
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificate for serving the Kubernetes API

Synopsis

Generate the certificate for serving the Kubernetes API, and save them into apiserver.crt and apiserver.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

```
kubeadm init phase certs apiserver [flags]
```

Options

--apiserver-advertise-address string	The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-cert-extra-sans strings	Optional extra Subject Alternative Names (SANs) to use for the API Server serving certificate. Can be both IP addresses and DNS names.
--cert-dir string Default: "/etc/kubernetes/pki"	The path where to save and store the certificates.
--config string	Path to a kubeadm configuration file.
--control-plane-endpoint string	Specify a stable IP address or DNS name for the control plane.
--dry-run	Don't apply any changes; just output what would be done.
-h, --help	help for apiserver
--kubernetes-version string Default: "stable-1"	Choose a specific Kubernetes version for the control plane.
--service-cidr string Default: "10.96.0.0/12"	Use alternative range of IP address for service VIPs.
--service-dns-domain string Default: "cluster.local"	

Use alternative domain for services, e.g. "myorg.internal".

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificate for the API server to connect to kubelet

Synopsis

Generate the certificate for the API server to connect to kubelet, and save them into apiserver-kubelet-client.crt and apiserver-kubelet-client.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

kubeadm init phase certs apiserver-kubelet-client [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"

The path where to save and store the certificates.

--config string

Path to a kubeadm configuration file.

--dry-run

Don't apply any changes; just output what would be done.

-h, --help

help for apiserver-kubelet-client

--kubernetes-version string Default: "stable-1"

Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the self-signed CA to provision identities for front proxy

Synopsis

Generate the self-signed CA to provision identities for front proxy, and save them into front-proxy-ca.crt and front-proxy-ca.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

```
kubeadm init phase certs front-proxy-ca [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.	
--config string	
Path to a kubeadm configuration file.	
--dry-run	
Don't apply any changes; just output what would be done.	
-h, --help	
help for front-proxy-ca	
--kubernetes-version string	Default: "stable-1"
Choose a specific Kubernetes version for the control plane.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificate for the front proxy client

Synopsis

Generate the certificate for the front proxy client, and save them into front-proxy-client.crt and front-proxy-client.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

```
kubeadm init phase certs front-proxy-client [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.	
--config string	
Path to a kubeadm configuration file.	
--dry-run	
Don't apply any changes; just output what would be done.	
-h, --help	
help for front-proxy-client	
--kubernetes-version string	Default: "stable-1"
Choose a specific Kubernetes version for the control plane.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the self-signed CA to provision identities for etcd

Synopsis

Generate the self-signed CA to provision identities for etcd, and save them into etcd/ca.crt and etcd/ca.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

```
kubeadm init phase certs etcd-ca [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.	
--config string	
Path to a kubeadm configuration file.	
--dry-run	
Don't apply any changes; just output what would be done.	
-h, --help	
help for etcd-ca	
--kubernetes-version string	Default: "stable-1"
Choose a specific Kubernetes version for the control plane.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificate for serving etcd

Synopsis

Generate the certificate for serving etcd, and save them into etcd/server.crt and etcd/server.key files.

Default SANs are localhost, 127.0.0.1, 127.0.0.1, ::1

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

```
kubeadm init phase certs etcd-server [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
-------------------	--------------------------------

The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for etcd-server
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificate for etcd nodes to communicate with each other

Synopsis

Generate the certificate for etcd nodes to communicate with each other, and save them into etcd/peer.crt and etcd/peer.key files.

Default SANs are localhost, 127.0.0.1, 127.0.0.1, ::1

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

kubeadm init phase certs etcd-peer [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.

--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for etcd-peer
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificate for liveness probes to healthcheck etcd

Synopsis

Generate the certificate for liveness probes to healthcheck etcd, and save them into etcd/healthcheck-client.crt and etcd/healthcheck-client.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

kubeadm init phase certs etcd-healthcheck-client [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for etcd-healthcheck-client
--kubernetes-version string Default: "stable-1"

Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificate the apiserver uses to access etcd

Synopsis

Generate the certificate the apiserver uses to access etcd, and save them into apiserver-etcd-client.crt and apiserver-etcd-client.key files.

If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

kubeadm init phase certs apiserver-etcd-client [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"

The path where to save and store the certificates.

--config string

Path to a kubeadm configuration file.

--dry-run

Don't apply any changes; just output what would be done.

-h, --help

help for apiserver-etcd-client

--kubernetes-version string Default: "stable-1"

Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate a private key for signing service account tokens along with its public key

Synopsis

Generate the private key for signing service account tokens along with its public key, and save them into sa.key and sa.pub files. If both files already exist, kubeadm skips the generation step and existing files will be used.

Alpha Disclaimer: this command is currently alpha.

```
kubeadm init phase certs sa [flags]
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.	
-h, --help	
help for sa	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase kubeconfig

You can create all required kubeconfig files by calling the all subcommand or call them individually.

- [kubeconfig](#)
- [all](#)
- [admin](#)
- [kubelet](#)
- [controller-manager](#)
- [scheduler](#)

Generate all kubeconfig files necessary to establish the control plane and the admin kubeconfig file

Synopsis

This command is not meant to be run on its own. See list of available subcommands.

```
kubeadm init phase kubeconfig [flags]
```

Options

-h, --help
help for kubeconfig

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate all kubeconfig files

Synopsis

Generate all kubeconfig files

```
kubeadm init phase kubeconfig all [flags]
```

Options

--apiserver-advertise-address string
The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443
Port for the API Server to bind to.
--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.
--control-plane-endpoint string
Specify a stable IP address or DNS name for the control plane.
--dry-run
Don't apply any changes; just output what would be done.

-h, --help
help for all
--kubeconfig-dir string Default: "/etc/kubernetes"
The path where to save the kubeconfig file.
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.
--node-name string
Specify the node name.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate a kubeconfig file for the admin to use and for kubeadm itself

Synopsis

Generate the kubeconfig file for the admin and for kubeadm itself, and save it to admin.conf file.

kubeadm init phase kubeconfig admin [flags]

Options

--apiserver-advertise-address string
The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443
Port for the API Server to bind to.
--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.

--control-plane-endpoint string
Specify a stable IP address or DNS name for the control plane.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for admin
--kubeconfig-dir string Default: "/etc/kubernetes"
The path where to save the kubeconfig file.
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate a kubeconfig file for the kubelet to use *only* for cluster bootstrapping purposes

Synopsis

Generate the kubeconfig file for the kubelet to use and save it to kubelet.conf file.

Please note that this should *only* be used for cluster bootstrapping purposes. After your control plane is up, you should request all kubelet credentials from the CSR API.

kubeadm init phase kubeconfig kubelet [flags]

Options

--apiserver-advertise-address string
The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443
Port for the API Server to bind to.
--cert-dir string Default: "/etc/kubernetes/pki"

The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.
--control-plane-endpoint string
Specify a stable IP address or DNS name for the control plane.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for kubelet
--kubeconfig-dir string Default: "/etc/kubernetes"
The path where to save the kubeconfig file.
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.
--node-name string
Specify the node name.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate a kubeconfig file for the controller manager to use

Synopsis

Generate the kubeconfig file for the controller manager to use and save it to controller-manager.conf file

kubeadm init phase kubeconfig controller-manager [flags]

Options

--apiserver-advertise-address string

The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443
Port for the API Server to bind to.
--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.
--control-plane-endpoint string
Specify a stable IP address or DNS name for the control plane.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for controller-manager
--kubeconfig-dir string Default: "/etc/kubernetes"
The path where to save the kubeconfig file.
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate a kubeconfig file for the scheduler to use

Synopsis

Generate the kubeconfig file for the scheduler to use and save it to scheduler.conf file.

kubeadm init phase kubeconfig scheduler [flags]

Options

--apiserver-advertise-address string	The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443	Port for the API Server to bind to.
--cert-dir string Default: "/etc/kubernetes/pki"	The path where to save and store the certificates.
--config string	Path to a kubeadm configuration file.
--control-plane-endpoint string	Specify a stable IP address or DNS name for the control plane.
--dry-run	Don't apply any changes; just output what would be done.
-h, --help	help for scheduler
--kubeconfig-dir string Default: "/etc/kubernetes"	The path where to save the kubeconfig file.
--kubernetes-version string Default: "stable-1"	Choose a specific Kubernetes version for the control plane.

Options inherited from parent commands

--rootfs string	[EXPERIMENTAL] The path to the 'real' host root filesystem.
-----------------	---

kubeadm init phase control-plane

Using this phase you can create all required static Pod files for the control plane components.

- [control-plane](#)
- [all](#)
- [apiserver](#)
- [controller-manager](#)
- [scheduler](#)

Generate all static Pod manifest files necessary to establish the control plane

Synopsis

This command is not meant to be run on its own. See list of available subcommands.

```
kubeadm init phase control-plane [flags]
```

Options

-h, --help
help for control-plane

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate all static Pod manifest files

Synopsis

Generate all static Pod manifest files

```
kubeadm init phase control-plane all [flags]
```

Examples

```
# Generates all static Pod manifest files for control plane components,  
# functionally equivalent to what is generated by kubeadm init.  
kubeadm init phase control-plane all  
  
# Generates all static Pod manifest files using options read from a configuration file.  
kubeadm init phase control-plane all --config config.yaml
```

Options

--apiserver-advertise-address string	The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443	Port for the API Server to bind to.
--apiserver-extra-args <comma-separated 'key=value' pairs>	A set of extra flags to pass to the API Server or override default ones in form of <flagname>=<value>
--cert-dir string Default: "/etc/kubernetes/pki"	The path where to save and store the certificates.
--config string	Path to a kubeadm configuration file.
--control-plane-endpoint string	Specify a stable IP address or DNS name for the control plane.
--controller-manager-extra-args <comma-separated 'key=value' pairs>	A set of extra flags to pass to the Controller Manager or override default ones in form of <flagname>=<value>
--dry-run	Don't apply any changes; just output what would be done.
--feature-gates string	A set of key=value pairs that describe feature gates for various features. Options are: EtcdLearnerMode=true false (ALPHA - default=false) PublicKeysECDSA=true false (ALPHA - default=false) RootlessControlPlane=true false (ALPHA - default=false) UpgradeAddonsBeforeControlPlane=true false (DEPRECATED - default=false)
-h, --help	help for all
--image-repository string Default: "registry.k8s.io"	

Choose a container registry to pull control plane images from
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.
--patches string
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.
--pod-network-cidr string
Specify range of IP addresses for the pod network. If set, the control plane will automatically allocate CIDRs for every node.
--scheduler-extra-args <comma-separated 'key=value' pairs>
A set of extra flags to pass to the Scheduler or override default ones in form of <flagname>=<value>
--service-cidr string Default: "10.96.0.0/12"
Use alternative range of IP address for service VIPs.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generates the kube-apiserver static Pod manifest

Synopsis

Generates the kube-apiserver static Pod manifest

kubeadm init phase control-plane apiserver [flags]

Options

--apiserver-advertise-address string

The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.

--apiserver-bind-port int32 Default: 6443

Port for the API Server to bind to.

--apiserver-extra-args <comma-separated 'key=value' pairs>

A set of extra flags to pass to the API Server or override default ones in form of
<flagname>=<value>

--cert-dir string Default: "/etc/kubernetes/pki"

The path where to save and store the certificates.

--config string

Path to a kubeadm configuration file.

--control-plane-endpoint string

Specify a stable IP address or DNS name for the control plane.

--dry-run

Don't apply any changes; just output what would be done.

--feature-gates string

A set of key=value pairs that describe feature gates for various features. Options are:
EtcLearnerMode=true|false (ALPHA - default=false)
PublicKeysECDSA=true|false (ALPHA - default=false)
RootlessControlPlane=true|false (ALPHA - default=false)
UpgradeAddonsBeforeControlPlane=true|false (DEPRECATED - default=false)

-h, --help

help for apiserver

--image-repository string Default: "registry.k8s.io"

Choose a container registry to pull control plane images from

--kubernetes-version string Default: "stable-1"

Choose a specific Kubernetes version for the control plane.

--patches string

Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

--service-cidr string Default: "10.96.0.0/12"

Use alternative range of IP address for service VIPs.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generates the kube-controller-manager static Pod manifest

Synopsis

Generates the kube-controller-manager static Pod manifest

kubeadm init phase control-plane controller-manager [flags]

Options

--cert-dir string Default: "/etc/kubernetes/pki"

The path where to save and store the certificates.

--config string

Path to a kubeadm configuration file.

--controller-manager-extra-args <comma-separated 'key=value' pairs>

A set of extra flags to pass to the Controller Manager or override default ones in form of <flagname>=<value>

--dry-run

Don't apply any changes; just output what would be done.

-h, --help

help for controller-manager

--image-repository string	Default: "registry.k8s.io"
Choose a container registry to pull control plane images from	
--kubernetes-version string	Default: "stable-1"
Choose a specific Kubernetes version for the control plane.	
--patches string	
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.	
--pod-network-cidr string	
Specify range of IP addresses for the pod network. If set, the control plane will automatically allocate CIDRs for every node.	

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generates the kube-scheduler static Pod manifest

Synopsis

Generates the kube-scheduler static Pod manifest

kubeadm init phase control-plane scheduler [flags]

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.	
--config string	
Path to a kubeadm configuration file.	
--dry-run	

Don't apply any changes; just output what would be done.

-h, --help

help for scheduler

--image-repository string Default: "registry.k8s.io"

Choose a container registry to pull control plane images from

--kubernetes-version string Default: "stable-1"

Choose a specific Kubernetes version for the control plane.

--patches string

Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

--scheduler-extra-args <comma-separated 'key=value' pairs>

A set of extra flags to pass to the Scheduler or override default ones in form of <flagname>=<value>

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase etcd

Use the following phase to create a local etcd instance based on a static Pod file.

- [etcd](#)
- [local](#)

Generate static Pod manifest file for local etcd

Synopsis

This command is not meant to be run on its own. See list of available subcommands.

kubeadm init phase etcd [flags]

Options

-h, --help
help for etcd

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the static Pod manifest file for a local, single-node local etcd instance

Synopsis

Generate the static Pod manifest file for a local, single-node local etcd instance

kubeadm init phase etcd local [flags]

Examples

```
# Generates the static Pod manifest file for etcd, functionally
# equivalent to what is generated by kubeadm init.
kubeadm init phase etcd local
```

```
# Generates the static Pod manifest file for etcd using options
# read from a configuration file.
kubeadm init phase etcd local --config config.yaml
```

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path where to save and store the certificates.
--config string
Path to a kubeadm configuration file.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for local
--image-repository string Default: "registry.k8s.io"

Choose a container registry to pull control plane images from

--patches string

Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase upload-config

You can use this command to upload the kubeadm configuration to your cluster. Alternatively, you can use [kubeadm config](#).

- [upload-config](#)
- [all](#)
- [kubeadm](#)
- [kubelet](#)

Upload the kubeadm and kubelet configuration to a ConfigMap

Synopsis

This command is not meant to be run on its own. See list of available subcommands.

kubeadm init phase upload-config [flags]

Options

-h, --help

help for upload-config

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Upload all configuration to a config map

Synopsis

Upload all configuration to a config map

kubeadm init phase upload-config all [flags]

Options

--config string	
	Path to a kubeadm configuration file.
--cri-socket string	
	Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--dry-run	
	Don't apply any changes; just output what would be done.
-h, --help	
	help for all
--kubeconfig string Default: "/etc/kubernetes/admin.conf"	
	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Upload the kubeadm ClusterConfiguration to a ConfigMap

Synopsis

Upload the kubeadm ClusterConfiguration to a ConfigMap called kubeadm-config in the kube-system namespace. This enables correct configuration of system components and a seamless user experience when upgrading.

Alternatively, you can use kubeadm config.

```
kubeadm init phase upload-config kubeadm [flags]
```

Examples

```
# upload the configuration of your cluster  
kubeadm init phase upload-config --config=myConfig.yaml
```

Options

--config string
Path to a kubeadm configuration file.
--cri-socket string
Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for kubeadm
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Upload the kubelet component config to a ConfigMap

Synopsis

Upload the kubelet configuration extracted from the kubeadm InitConfiguration object to a kubelet-config ConfigMap in the cluster

```
kubeadm init phase upload-config kubelet [flags]
```

Examples

```
# Upload the kubelet configuration from the kubeadm Config file to a ConfigMap in the cluster.  
kubeadm init phase upload-config kubelet --config kubeadm.yaml
```

Options

--config string	
	Path to a kubeadm configuration file.
--cri-socket string	
	Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--dry-run	
	Don't apply any changes; just output what would be done.
-h, --help	
	help for kubelet
--kubeconfig string Default: "/etc/kubernetes/admin.conf"	
	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string	
	[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase upload-certs

Use the following phase to upload control-plane certificates to the cluster. By default the certs and encryption key expire after two hours.

- [upload-certs](#)

Upload certificates to kubeadm-certs

Synopsis

Upload control plane certificates to the kubeadm-certs Secret

```
kubeadm init phase upload-certs [flags]
```

Options

--certificate-key string
Key used to encrypt the control-plane certificates in the kubeadm-certs Secret.
--config string
Path to a kubeadm configuration file.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for upload-certs
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--skip-certificate-key-print
Don't print the key used to encrypt the control-plane certificates.
--upload-certs
Upload control-plane certificates to the kubeadm-certs Secret.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase mark-control-plane

Use the following phase to label and taint the node as a control plane node.

- [mark-control-plane](#)

Mark a node as a control-plane

Synopsis

Mark a node as a control-plane

```
kubeadm init phase mark-control-plane [flags]
```

Examples

```
# Applies control-plane label and taint to the current node, functionally equivalent to what
executed by kubeadm init.
```

```
kubeadm init phase mark-control-plane --config config.yaml
```

```
# Applies control-plane label and taint to a specific node
```

```
kubeadm init phase mark-control-plane --node-name myNode
```

Options

--config string
Path to a kubeadm configuration file.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for mark-control-plane
--node-name string
Specify the node name.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase bootstrap-token

Use the following phase to configure bootstrap tokens.

- [bootstrap-token](#)

Generates bootstrap tokens used to join a node to a cluster

Synopsis

Bootstrap tokens are used for establishing bidirectional trust between a node joining the cluster and a control-plane node.

This command makes all the configurations required to make bootstrap tokens work and then creates an initial token.

```
kubeadm init phase bootstrap-token [flags]
```

Examples

```
# Make all the bootstrap token configurations and create an initial token, functionally
# equivalent to what generated by kubeadm init.
kubeadm init phase bootstrap-token
```

Options

--config string	
	Path to a kubeadm configuration file.
--dry-run	
	Don't apply any changes; just output what would be done.
-h, --help	
	help for bootstrap-token
--kubeconfig string Default: "/etc/kubernetes/admin.conf"	
	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--skip-token-print	
	Skip printing of the default bootstrap token generated by 'kubeadm init'.

Options inherited from parent commands

--rootfs string	
	[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm init phase kubelet-finalize

Use the following phase to update settings relevant to the kubelet after TLS bootstrap. You can use the all subcommand to run all kubelet-finalize phases.

- [kubelet-finalize](#)
- [kubelet-finalize-all](#)
- [kubelet-finalize-cert-rotation](#)

Updates settings relevant to the kubelet after TLS bootstrap

Synopsis

Updates settings relevant to the kubelet after TLS bootstrap

```
kubeadm init phase kubelet-finalize [flags]
```

Examples

```
# Updates settings relevant to the kubelet after TLS bootstrap"
kubeadm init phase kubelet-finalize all --config
```

Options

-h, --help
help for kubelet-finalize

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Run all kubelet-finalize phases

Synopsis

Run all kubelet-finalize phases

```
kubeadm init phase kubelet-finalize all [flags]
```

Examples

```
# Updates settings relevant to the kubelet after TLS bootstrap"
kubeadm init phase kubelet-finalize all --config
```

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
	The path where to save and store the certificates.
--config string	
	Path to a kubeadm configuration file.
--dry-run	
	Don't apply any changes; just output what would be done.
-h, --help	
	help for all

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Enable kubelet client certificate rotation

Synopsis

Enable kubelet client certificate rotation

kubeadm init phase kubelet-finalize experimental-cert-rotation [flags]

Options

--cert-dir string	Default: "/etc/kubernetes/pki"
	The path where to save and store the certificates.
--config string	
	Path to a kubeadm configuration file.
--dry-run	
	Don't apply any changes; just output what would be done.
-h, --help	

```
help for experimental-cert-rotation
```

Options inherited from parent commands

```
--rootfs string
```

```
[EXPERIMENTAL] The path to the 'real' host root filesystem.
```

kubeadm init phase addon

You can install all the available addons with the `all` subcommand, or install them selectively.

- [addon](#)
- [all](#)
- [coredns](#)
- [kube-proxy](#)

Install required addons for passing conformance tests

Synopsis

This command is not meant to be run on its own. See list of available subcommands.

```
kubeadm init phase addon [flags]
```

Options

```
-h, --help
```

```
help for addon
```

Options inherited from parent commands

```
--rootfs string
```

```
[EXPERIMENTAL] The path to the 'real' host root filesystem.
```

Install all the addons

Synopsis

Install all the addons

```
kubeadm init phase addon all [flags]
```

Options

--apiserver-advertise-address string	The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443	Port for the API Server to bind to.
--config string	Path to a kubeadm configuration file.
--control-plane-endpoint string	Specify a stable IP address or DNS name for the control plane.
--dry-run	Don't apply any changes; just output what would be done.
--feature-gates string	A set of key=value pairs that describe feature gates for various features. Options are: EtcdLearnerMode=true false (ALPHA - default=false) PublicKeysECDSA=true false (ALPHA - default=false) RootlessControlPlane=true false (ALPHA - default=false) UpgradeAddonsBeforeControlPlane=true false (DEPRECATED - default=false)
-h, --help	help for all
--imagerepository string Default: "registry.k8s.io"	Choose a container registry to pull control plane images from
--kubeconfig string Default: "/etc/kubernetes/admin.conf"	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--kubernetes-version string Default: "stable-1"	Choose a specific Kubernetes version for the control plane.
--pod-network-cidr string	

Specify range of IP addresses for the pod network. If set, the control plane will automatically allocate CIDRs for every node.

--service-cidr string Default: "10.96.0.0/12"

Use alternative range of IP address for service VIPs.

--service-dns-domain string Default: "cluster.local"

Use alternative domain for services, e.g. "myorg.internal".

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Install the CoreDNS addon to a Kubernetes cluster

Synopsis

Install the CoreDNS addon components via the API server. Please note that although the DNS server is deployed, it will not be scheduled until CNI is installed.

kubeadm init phase addon coredns [flags]

Options

--config string

Path to a kubeadm configuration file.

--dry-run

Don't apply any changes; just output what would be done.

--feature-gates string

A set of key=value pairs that describe feature gates for various features. Options are:
EtcdLearnerMode=true|false (ALPHA - default=false)
PublicKeysECDSA=true|false (ALPHA - default=false)
RootlessControlPlane=true|false (ALPHA - default=false)
UpgradeAddonsBeforeControlPlane=true|false (DEPRECATED - default=false)

-h, --help

help for coredns

--image-repository string Default: "registry.k8s.io"

Choose a container registry to pull control plane images from
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.
--print-manifest
Print the addon manifests to STDOUT instead of installing them
--service-cidr string Default: "10.96.0.0/12"
Use alternative range of IP address for service VIPs.
--service-dns-domain string Default: "cluster.local"
Use alternative domain for services, e.g. "myorg.internal".

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Install the kube-proxy addon to a Kubernetes cluster

Synopsis

Install the kube-proxy addon components via the API server.

kubeadm init phase addon kube-proxy [flags]

Options

--apiserver-advertise-address string
The IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443
Port for the API Server to bind to.

Path to a kubeadm configuration file.
--control-plane-endpoint string
Specify a stable IP address or DNS name for the control plane.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for kube-proxy
--image-repository string Default: "registry.k8s.io"
Choose a container registry to pull control plane images from
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--kubernetes-version string Default: "stable-1"
Choose a specific Kubernetes version for the control plane.
--pod-network-cidr string
Specify range of IP addresses for the pod network. If set, the control plane will automatically allocate CIDRs for every node.
--print-manifest
Print the addon manifests to STDOUT instead of installing them

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

For more details on each field in the v1beta3 configuration you can navigate to our [API reference pages](#).

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node

- [kubeadm join](#) to connect a node to the cluster
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join
- [kubeadm alpha](#) to try experimental functionality

kubeadm join phase

kubeadm join phase enables you to invoke atomic steps of the join process. Hence, you can let kubeadm do some of the work and you can fill in the gaps if you wish to apply customization.

kubeadm join phase is consistent with the [kubeadm join workflow](#), and behind the scene both use the same code.

kubeadm join phase

- [phase](#)

Use this command to invoke single phase of the join workflow

Synopsis

Use this command to invoke single phase of the join workflow

Options

-h, --help
help for phase

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm join phase preflight

Using this phase you can execute preflight checks on a joining node.

- [preflight](#)

Run join pre-flight checks

Synopsis

Run pre-flight checks for kubeadm join.

```
kubeadm join phase preflight [api-server-endpoint] [flags]
```

Examples

```
# Run join pre-flight checks using a config file.  
kubeadm join phase preflight --config kubeadm-config.yaml
```

Options

--apiserver-advertise-address string	If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443	If the node should host a new control plane instance, the port for the API Server to bind to.
--certificate-key string	Use this key to decrypt the certificate secrets uploaded by init.
--config string	Path to a kubeadm configuration file.
--control-plane	Create a new control plane instance on this node
--cri-socket string	Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--discovery-file string	For file-based discovery, a file or URL from which to load cluster information.
--discovery-token string	For token-based discovery, the token used to validate cluster information fetched from the API server.
--discovery-token-ca-cert-hash strings	For token-based discovery, validate that the root CA public key matches this hash (format: "<type>:<value>").
--discovery-token-unsafe-skip-ca-validation	For token-based discovery, allow joining without --discovery-token-ca-cert-hash pinning.

--dry-run	Don't apply any changes; just output what would be done.
-h, --help	help for preflight
--ignore-preflight-errors strings	A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
--node-name string	Specify the node name.
--tls-bootstrap-token string	Specify the token used to temporarily authenticate with the Kubernetes Control Plane while joining the node.
--token string	Use this token for both discovery-token and tls-bootstrap-token when those values are not provided.

Options inherited from parent commands

--rootfs string	[EXPERIMENTAL] The path to the 'real' host root filesystem.
-----------------	---

kubeadm join phase control-plane-prepare

Using this phase you can prepare a node for serving a control-plane.

- [control-plane-prepare](#)
- [all](#)
- [download-certs](#)
- [certs](#)
- [kubeconfig](#)
- [control-plane](#)

Prepare the machine for serving a control plane

Synopsis

Prepare the machine for serving a control plane

```
kubeadm join phase control-plane-prepare [flags]
```

Examples

```
# Prepares the machine for serving a control plane
kubeadm join phase control-plane-prepare all
```

Options

-h, --help
help for control-plane-prepare

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Prepare the machine for serving a control plane

Synopsis

Prepare the machine for serving a control plane

```
kubeadm join phase control-plane-prepare all [api-server-endpoint] [flags]
```

Options

--apiserver-advertise-address string
If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--apiserver-bind-port int32 Default: 6443
If the node should host a new control plane instance, the port for the API Server to bind to.
--certificate-key string
Use this key to decrypt the certificate secrets uploaded by init.
--config string
Path to a kubeadm configuration file.
--control-plane

Create a new control plane instance on this node
--discovery-file string
For file-based discovery, a file or URL from which to load cluster information.
--discovery-token string
For token-based discovery, the token used to validate cluster information fetched from the API server.
--discovery-token-ca-cert-hash strings
For token-based discovery, validate that the root CA public key matches this hash (format: "<type>:<value>").
--discovery-token-unsafe-skip-ca-verification
For token-based discovery, allow joining without --discovery-token-ca-cert-hash pinning.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for all
--node-name string
Specify the node name.
--patches string
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.
--tls-bootstrap-token string
Specify the token used to temporarily authenticate with the Kubernetes Control Plane while joining the node.
--token string

Use this token for both discovery-token and tls-bootstrap-token when those values are not provided.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

[EXPERIMENTAL] Download certificates shared among control-plane nodes from the kubeadm-certs Secret

Synopsis

[EXPERIMENTAL] Download certificates shared among control-plane nodes from the kubeadm-certs Secret

kubeadm join phase control-plane-prepare download-certs [api-server-endpoint] [flags]

Options

--certificate-key string

Use this key to decrypt the certificate secrets uploaded by init.

--config string

Path to a kubeadm configuration file.

--control-plane

Create a new control plane instance on this node

--discovery-file string

For file-based discovery, a file or URL from which to load cluster information.

--discovery-token string

For token-based discovery, the token used to validate cluster information fetched from the API server.

--discovery-token-ca-cert-hash strings

For token-based discovery, validate that the root CA public key matches this hash (format: "<type>:<value>").

--discovery-token-unsafe-skip-ca-verification

For token-based discovery, allow joining without --discovery-token-ca-cert-hash pinning.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for download-certs
--tls-bootstrap-token string
Specify the token used to temporarily authenticate with the Kubernetes Control Plane while joining the node.
--token string
Use this token for both discovery-token and tls-bootstrap-token when those values are not provided.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the certificates for the new control plane components

Synopsis

Generate the certificates for the new control plane components

kubeadm join phase control-plane-prepare certs [api-server-endpoint] [flags]

Options

--apiserver-advertise-address string
If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--config string
Path to a kubeadm configuration file.
--control-plane
Create a new control plane instance on this node

--discovery-file string
For file-based discovery, a file or URL from which to load cluster information.
--discovery-token string
For token-based discovery, the token used to validate cluster information fetched from the API server.
--discovery-token-ca-cert-hash strings
For token-based discovery, validate that the root CA public key matches this hash (format: "<type>:<value>").
--discovery-token-unsafe-skip-ca-verification
For token-based discovery, allow joining without --discovery-token-ca-cert-hash pinning.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for certs
--node-name string
Specify the node name.
--tls-bootstrap-token string
Specify the token used to temporarily authenticate with the Kubernetes Control Plane while joining the node.
--token string
Use this token for both discovery-token and tls-bootstrap-token when those values are not provided.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the kubeconfig for the new control plane components

Synopsis

Generate the kubeconfig for the new control plane components

```
kubeadm join phase control-plane-prepare kubeconfig [api-server-endpoint] [flags]
```

Options

--certificate-key string	
	Use this key to decrypt the certificate secrets uploaded by init.
--config string	
	Path to a kubeadm configuration file.
--control-plane	
	Create a new control plane instance on this node
--discovery-file string	
	For file-based discovery, a file or URL from which to load cluster information.
--discovery-token string	
	For token-based discovery, the token used to validate cluster information fetched from the API server.
--discovery-token-ca-cert-hash strings	
	For token-based discovery, validate that the root CA public key matches this hash (format: "<type>:<value>").
--discovery-token-unsafe-skip-ca-verification	
	For token-based discovery, allow joining without --discovery-token-ca-cert-hash pinning.
--dry-run	
	Don't apply any changes; just output what would be done.
-h, --help	
	help for kubeconfig
--tls-bootstrap-token string	
	Specify the token used to temporarily authenticate with the Kubernetes Control Plane while joining the node.

--token string

Use this token for both discovery-token and tls-bootstrap-token when those values are not provided.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Generate the manifests for the new control plane components

Synopsis

Generate the manifests for the new control plane components

kubeadm join phase control-plane-prepare control-plane [flags]

Options

--apiserver-advertise-address string

If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.

--apiserver-bind-port int32 Default: 6443

If the node should host a new control plane instance, the port for the API Server to bind to.

--config string

Path to a kubeadm configuration file.

--control-plane

Create a new control plane instance on this node

--dry-run

Don't apply any changes; just output what would be done.

-h, --help

help for control-plane

--patches string

Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm join phase kubelet-start

Using this phase you can write the kubelet settings, certificates and (re)start the kubelet.

- [kubelet-start](#)

Write kubelet settings, certificates and (re)start the kubelet

Synopsis

Write a file with KubeletConfiguration and an environment file with node specific kubelet settings, and then (re)start kubelet.

kubeadm join phase kubelet-start [api-server-endpoint] [flags]

Options

--config string

Path to a kubeadm configuration file.

--cri-socket string

Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.

--discovery-file string

For file-based discovery, a file or URL from which to load cluster information.

--discovery-token string

For token-based discovery, the token used to validate cluster information fetched from the API server.
--discovery-token-ca-cert-hash strings
For token-based discovery, validate that the root CA public key matches this hash (format: "<type>:<value>").
--discovery-token-unsafe-skip-ca-verification
For token-based discovery, allow joining without --discovery-token-ca-cert-hash pinning.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for kubelet-start
--node-name string
Specify the node name.
--patches string
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.
--tls-bootstrap-token string
Specify the token used to temporarily authenticate with the Kubernetes Control Plane while joining the node.
--token string
Use this token for both discovery-token and tls-bootstrap-token when those values are not provided.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm join phase control-plane-join

Using this phase you can join a node as a control-plane instance.

- [control-plane-join](#)
- [all](#)
- [etcd](#)
- [update-status](#)
- [mark-control-plane](#)

Join a machine as a control plane instance

Synopsis

Join a machine as a control plane instance

```
kubeadm join phase control-plane-join [flags]
```

Examples

```
# Joins a machine as a control plane instance
kubeadm join phase control-plane-join all
```

Options

-h, --help
help for control-plane-join

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Join a machine as a control plane instance

Synopsis

Join a machine as a control plane instance

```
kubeadm join phase control-plane-join all [flags]
```

Options

--apiserver-advertise-address string

If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.

--config string

Path to a kubeadm configuration file.

--control-plane

Create a new control plane instance on this node

--dry-run

Don't apply any changes; just output what would be done.

-h, --help

help for all

--node-name string

Specify the node name.

--patches string

Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

Add a new local etcd member

Synopsis

Add a new local etcd member

kubeadm join phase control-plane-join etcd [flags]

Options

--apiserver-advertise-address string	If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--config string	Path to a kubeadm configuration file.
--control-plane	Create a new control plane instance on this node
--dry-run	Don't apply any changes; just output what would be done.
-h, --help	help for etcd
--node-name string	Specify the node name.
--patches string	Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

Options inherited from parent commands

--rootfs string	[EXPERIMENTAL] The path to the 'real' host root filesystem.
-----------------	---

Register the new control-plane node into the ClusterStatus maintained in the kubeadm-config ConfigMap (DEPRECATED)

Synopsis

Register the new control-plane node into the ClusterStatus maintained in the kubeadm-config ConfigMap (DEPRECATED)

```
kubeadm join phase control-plane-join update-status [flags]
```

Options

--apiserver-advertise-address string
If the node should host a new control plane instance, the IP address the API Server will advertise it's listening on. If not set the default network interface will be used.
--config string
Path to a kubeadm configuration file.
--control-plane
Create a new control plane instance on this node
-h, --help
help for update-status
--node-name string
Specify the node name.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Mark a node as a control-plane

Synopsis

Mark a node as a control-plane

```
kubeadm join phase control-plane-join mark-control-plane [flags]
```

Options

--config string
Path to a kubeadm configuration file.
--control-plane
Create a new control plane instance on this node

--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for mark-control-plane
--node-name string
Specify the node name.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node
- [kubeadm join](#) to connect a node to the cluster
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join
- [kubeadm alpha](#) to try experimental functionality

kubeadm kubeconfig

kubeadm kubeconfig provides utilities for managing kubeconfig files.

For examples on how to use kubeadm kubeconfig user see [Generating kubeconfig files for additional users](#).

kubeadm kubeconfig

- [overview](#)

Kubeconfig file utilities

Synopsis

Kubeconfig file utilities.

Options

-h, --help

```
help for kubeconfig
```

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm kubeconfig user

This command can be used to output a kubeconfig file for an additional user.

- [user](#)

Output a kubeconfig file for an additional user

Synopsis

Output a kubeconfig file for an additional user.

```
kubeadm kubeconfig user [flags]
```

Examples

```
# Output a kubeconfig file for an additional user named foo
kubeadm kubeconfig user --client-name=foo
```

```
# Output a kubeconfig file for an additional user named foo using a kubeadm config file bar
kubeadm kubeconfig user --client-name=foo --config=bar
```

Options

--client-name string
The name of user. It will be used as the CN if client certificates are created

--config string
Path to a kubeadm configuration file.

-h, --help
help for user

--org strings

The organizations of the client certificate. It will be used as the O if client certificates are created

--token string

The token that should be used as the authentication mechanism for this kubeconfig, instead of client certificates

--validity-period duration Default: 8760h0m0s

The validity period of the client certificate. It is an offset from the current time.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm reset phase

kubeadm reset phase enables you to invoke atomic steps of the node reset process. Hence, you can let kubeadm do some of the work and you can fill in the gaps if you wish to apply customization.

kubeadm reset phase is consistent with the [kubeadm reset workflow](#), and behind the scene both use the same code.

kubeadm reset phase

- [phase](#)

Use this command to invoke single phase of the reset workflow

Synopsis

Use this command to invoke single phase of the reset workflow

Options

-h, --help

help for phase

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm reset phase preflight

Using this phase you can execute preflight checks on a node that is being reset.

- [preflight](#)

Run reset pre-flight checks

Synopsis

Run pre-flight checks for kubeadm reset.

kubeadm reset phase preflight [flags]

Options

--dry-run

Don't apply any changes; just output what would be done.

-f, --force

Reset the node without prompting for confirmation.

-h, --help

help for preflight

--ignore-preflight-errors strings

A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.

Options inherited from parent commands

--rootfs string

[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm reset phase remove-etcd-member

Using this phase you can remove this control-plane node's etcd member from the etcd cluster.

- [remove-etcd-member](#)

Remove a local etcd member.

Synopsis

Remove a local etcd member for a control plane node.

```
kubeadm reset phase remove-etcd-member [flags]
```

Options

--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for remove-etcd-member
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

kubeadm reset phase cleanup-node

Using this phase you can perform cleanup on this node.

- [cleanup-node](#)

Run cleanup node.

Synopsis

Run cleanup node.

```
kubeadm reset phase cleanup-node [flags]
```

Options

--cert-dir string Default: "/etc/kubernetes/pki"
The path to the directory where the certificates are stored. If specified, clean this directory.

--cleanup-tmp-dir
Cleanup the "/etc/kubernetes/tmp" directory
--cri-socket string
Path to the CRI socket to connect. If empty kubeadm will try to auto-detect this value; use this option only if you have more than one CRI installed or if you have non-standard CRI socket.
--dry-run
Don't apply any changes; just output what would be done.
-h, --help
help for cleanup-node

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node
- [kubeadm join](#) to connect a node to the cluster
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join
- [kubeadm alpha](#) to try experimental functionality

kubeadm upgrade phase

In v1.15.0, kubeadm introduced preliminary support for kubeadm upgrade node phases. Phases for other kubeadm upgrade sub-commands such as apply, could be added in the following releases.

kubeadm upgrade node phase

Using this phase you can choose to execute the separate steps of the upgrade of secondary control-plane or worker nodes. Please note that kubeadm upgrade apply still has to be called on a primary control-plane node.

- [phase](#)
- [preflight](#)
- [control-plane](#)
- [kubelet-config](#)

Use this command to invoke single phase of the node workflow

Synopsis

Use this command to invoke single phase of the node workflow

Options

-h, --help
help for phase

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Run upgrade node pre-flight checks

Synopsis

Run pre-flight checks for kubeadm upgrade node.

kubeadm upgrade node phase preflight [flags]

Options

-h, --help
help for preflight
--ignore-preflight-errors strings
A list of checks whose errors will be shown as warnings. Example: 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Upgrade the control plane instance deployed on this node, if any

Synopsis

Upgrade the control plane instance deployed on this node, if any

```
kubeadm upgrade node phase control-plane [flags]
```

Options

--certificate-renewal Default: true
Perform the renewal of certificates used by component changed during upgrades.
--dry-run
Do not change any state, just output the actions that would be performed.
--etcd-upgrade Default: true
Perform the upgrade of etcd.
-h, --help
help for control-plane
--kubeconfig string Default: "/etc/kubernetes/admin.conf"
The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--patches string
Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

Upgrade the kubelet configuration for this node

Synopsis

Download the kubelet configuration from the kubelet-config ConfigMap stored in the cluster

```
kubeadm upgrade node phase kubelet-config [flags]
```

Options

--dry-run	Do not change any state, just output the actions that would be performed.
-h, --help	help for kubelet-config
--kubeconfig string Default: "/etc/kubernetes/admin.conf"	The kubeconfig file to use when talking to the cluster. If the flag is not set, a set of standard locations can be searched for an existing kubeconfig file.
--patches string	Path to a directory that contains files named "target[suffix][+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic", "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

Options inherited from parent commands

--rootfs string
[EXPERIMENTAL] The path to the 'real' host root filesystem.

What's next

- [kubeadm init](#) to bootstrap a Kubernetes control-plane node
- [kubeadm join](#) to connect a node to the cluster
- [kubeadm reset](#) to revert any changes made to this host by kubeadm init or kubeadm join
- [kubeadm upgrade](#) to upgrade a kubeadm node
- [kubeadm alpha](#) to try experimental functionality

Implementation details

FEATURE STATE: Kubernetes v1.10 [stable]

kubeadm init and kubeadm join together provides a nice user experience for creating a best-practice but bare Kubernetes cluster from scratch. However, it might not be obvious *how* kubeadm does that.

This document provides additional details on what happen under the hood, with the aim of sharing knowledge on Kubernetes cluster best practices.

Core design principles

The cluster that kubeadm init and kubeadm join set up should be:

- **Secure:** It should adopt latest best-practices like:
 - enforcing RBAC
 - using the Node Authorizer
 - using secure communication between the control plane components
 - using secure communication between the API server and the kubelets
 - lock-down the kubelet API
 - locking down access to the API for system components like the kube-proxy and CoreDNS
 - locking down what a Bootstrap Token can access
- **User-friendly:** The user should not have to run anything more than a couple of commands:
 - kubeadm init
 - export KUBECONFIG=/etc/kubernetes/admin.conf
 - kubectl apply -f <network-of-choice.yaml>
 - kubeadm join --token <token> <endpoint>:<port>
- **Extendable:**
 - It should *not* favor any particular network provider. Configuring the cluster network is out-of-scope
 - It should provide the possibility to use a config file for customizing various parameters

Constants and well-known values and paths

In order to reduce complexity and to simplify development of higher level tools that build on top of kubeadm, it uses a limited set of constant values for well-known paths and file names.

The Kubernetes directory /etc/kubernetes is a constant in the application, since it is clearly the given path in a majority of cases, and the most intuitive location; other constants paths and file names are:

- /etc/kubernetes/manifests as the path where kubelet should look for static Pod manifests.
Names of static Pod manifests are:
 - etcd.yaml
 - kube-apiserver.yaml
 - kube-controller-manager.yaml
 - kube-scheduler.yaml
- /etc/kubernetes/ as the path where kubeconfig files with identities for control plane components are stored. Names of kubeconfig files are:
 - kubelet.conf (bootstrap-kubelet.conf during TLS bootstrap)
 - controller-manager.conf
 - scheduler.conf
 - admin.conf for the cluster admin and kubeadm itself

Names of certificates and key files :

- - ca.crt, ca.key for the Kubernetes certificate authority
 - apiserver.crt, apiserver.key for the API server certificate
 - apiserver-kubelet-client.crt, apiserver-kubelet-client.key for the client certificate used by the API server to connect to the kubelets securely
 - sa.pub, sa.key for the key used by the controller manager when signing ServiceAccount
 - front-proxy-ca.crt, front-proxy-ca.key for the front proxy certificate authority
 - front-proxy-client.crt, front-proxy-client.key for the front proxy client

kubeadm init workflow internal design

The kubeadm init [internal workflow](#) consists of a sequence of atomic work tasks to perform, as described in kubeadm init.

The [kubeadm init phase](#) command allows users to invoke each task individually, and ultimately offers a reusable and composable API/toolbox that can be used by other Kubernetes bootstrap tools, by any IT automation tool or by an advanced user for creating custom clusters.

Preflight checks

Kubeadm executes a set of preflight checks before starting the init, with the aim to verify preconditions and avoid common cluster startup problems. The user can skip specific preflight checks or all of them with the --ignore-preflight-errors option.

- [warning] If the Kubernetes version to use (specified with the --kubernetes-version flag) is at least one minor version higher than the kubeadm CLI version.
- Kubernetes system requirements:
 - if running on linux:
 - [error] if Kernel is older than the minimum required version
 - [error] if required cgroups subsystem aren't set up
 - [error] if the CRI endpoint does not answer
 - [error] if user is not root
 - [error] if the machine hostname is not a valid DNS subdomain
 - [warning] if the host name cannot be reached via network lookup
 - [error] if kubelet version is lower than the minimum kubelet version supported by kubeadm (current minor -1)
 - [error] if kubelet version is at least one minor higher than the required controlplane version (unsupported version skew)
 - [warning] if kubelet service does not exist or if it is disabled
 - [warning] if firewalld is active
 - [error] if API server bindPort or ports 10250/10251/10252 are used
 - [Error] if /etc/kubernetes/manifest folder already exists and it is not empty
 - [Error] if /proc/sys/net/bridge/bridge-nf-call-iptables file does not exist/does not contain 1
 - [Error] if advertise address is ipv6 and /proc/sys/net/bridge/bridge-nf-call-ip6tables does not exist/does not contain 1.
 - [Error] if swap is on
 - [Error] if conntrack, ip, iptables, mount, nsenter commands are not present in the command path

- [warning] if ebtables, ethtool, socat, tc, touch, crictl commands are not present in the command path
- [warning] if extra arg flags for API server, controller manager, scheduler contains some invalid options
- [warning] if connection to https://API.AdvertiseAddress:API.BindPort goes through proxy
- [warning] if connection to services subnet goes through proxy (only first address checked)
- [warning] if connection to Pods subnet goes through proxy (only first address checked)
- If external etcd is provided:
 - [Error] if etcd version is older than the minimum required version
 - [Error] if etcd certificates or keys are specified, but not provided
- If external etcd is NOT provided (and thus local etcd will be installed):
 - [Error] if ports 2379 is used
 - [Error] if Etcd.DataDir folder already exists and it is not empty
- If authorization mode is ABAC:
 - [Error] if abac_policy.json does not exist
- If authorization mode is WebHook
 - [Error] if webhook_authz.conf does not exist

Please note that:

1. Preflight checks can be invoked individually with the [kubeadm init phase preflight](#) command

Generate the necessary certificates

Kubeadm generates certificate and private key pairs for different purposes:

- A self signed certificate authority for the Kubernetes cluster saved into ca.crt file and ca.key private key file
- A serving certificate for the API server, generated using ca.crt as the CA, and saved into apiserver.crt file with its private key apiserver.key. This certificate should contain following alternative names:
 - The Kubernetes service's internal clusterIP (the first address in the services CIDR, e.g. 10.96.0.1 if service subnet is 10.96.0.0/12)
 - Kubernetes DNS names, e.g. kubernetes.default.svc.cluster.local if --service-dns-domain flag value is cluster.local, plus default DNS names kubernetes.default.svc, kubernetes.default, kubernetes
 - The node-name
 - The --apiserver-advertise-address
 - Additional alternative names specified by the user
- A client certificate for the API server to connect to the kubelets securely, generated using ca.crt as the CA and saved into apiserver-kubelet-client.crt file with its private key apiserver-kubelet-client.key. This certificate should be in the system:masters organization
- A private key for signing ServiceAccount Tokens saved into sa.key file along with its public key sa.pub
- A certificate authority for the front proxy saved into front-proxy-ca.crt file with its key front-proxy-ca.key

- A client cert for the front proxy client, generate using front-proxy-ca.crt as the CA and saved into front-proxy-client.crt file with its private keyfront-proxy-client.key

Certificates are stored by default in /etc/kubernetes/pki, but this directory is configurable using the --cert-dir flag.

Please note that:

1. If a given certificate and private key pair both exist, and its content is evaluated compliant with the above specs, the existing files will be used and the generation phase for the given certificate skipped. This means the user can, for example, copy an existing CA to /etc/kubernetes/pki/ca.{crt,key}, and then kubeadm will use those files for signing the rest of the certs. See also [using custom certificates](#)
2. Only for the CA, it is possible to provide the ca.crt file but not the ca.key file, if all other certificates and kubeconfig files already are in place kubeadm recognize this condition and activates the ExternalCA , which also implies the csrsignercontroller in controller-manager won't be started
3. If kubeadm is running in [external CA mode](#); all the certificates must be provided by the user, because kubeadm cannot generate them by itself
4. In case of kubeadm is executed in the --dry-run mode, certificates files are written in a temporary folder
5. Certificate generation can be invoked individually with the [kubeadm init phase certs all](#) command

Generate kubeconfig files for control plane components

Kubeadm generates kubeconfig files with identities for control plane components:

- A kubeconfig file for the kubelet to use during TLS bootstrap - /etc/kubernetes/bootstrap-kubelet.conf. Inside this file there is a bootstrap-token or embedded client certificates for authenticating this node with the cluster.

This client cert should:

- Be in the system:nodes organization, as required by the [Node Authorization module](#)
- Have the Common Name (CN) system:node:<hostname-lowercased>
- A kubeconfig file for controller-manager, /etc/kubernetes/controller-manager.conf; inside this file is embedded a client certificate with controller-manager identity. This client cert should have the CN system:kube-controller-manager, as defined by default [RBAC core components roles](#)
- A kubeconfig file for scheduler, /etc/kubernetes/scheduler.conf; inside this file is embedded a client certificate with scheduler identity. This client cert should have the CN system:kube-scheduler, as defined by default [RBAC core components roles](#)

Additionally, a kubeconfig file for kubeadm itself and the admin is generated and saved into the /etc/kubernetes/admin.conf file. The "admin" here is defined as the actual person(s) that is administering the cluster and wants to have full control (**root**) over the cluster. The embedded client certificate for admin should be in the system:masters organization, as defined by default [RBAC user facing role bindings](#). It should also include a CN. Kubeadm uses the kubernetes-admin CN.

Please note that:

1. ca.crt certificate is embedded in all the kubeconfig files.
2. If a given kubeconfig file exists, and its content is evaluated compliant with the above specs, the existing file will be used and the generation phase for the given kubeconfig skipped
3. If kubeadm is running in [ExternalCA mode](#), all the required kubeconfig must be provided by the user as well, because kubeadm cannot generate any of them by itself
4. In case of kubeadm is executed in the --dry-run mode, kubeconfig files are written in a temporary folder
5. Kubeconfig files generation can be invoked individually with the [kubeadm init phase kubeconfig all](#) command

Generate static Pod manifests for control plane components

Kubeadm writes static Pod manifest files for control plane components to /etc/kubernetes/manifests. The kubelet watches this directory for Pods to create on startup.

Static Pod manifest share a set of common properties:

- All static Pods are deployed on kube-system namespace
- All static Pods get tier:control-plane and component:{component-name} labels
- All static Pods use the system-node-critical priority class
- hostNetwork: true is set on all static Pods to allow control plane startup before a network is configured; as a consequence:
 - The address that the controller-manager and the scheduler use to refer the API server is 127.0.0.1
 - If using a local etcd server, etcd-servers address will be set to 127.0.0.1:2379
- Leader election is enabled for both the controller-manager and the scheduler
- Controller-manager and the scheduler will reference kubeconfig files with their respective, unique identities
- All static Pods get any extra flags specified by the user as described in [passing custom arguments to control plane components](#)
- All static Pods get any extra Volumes specified by the user (Host path)

Please note that:

1. All images will be pulled from registry.k8s.io by default. See [using custom images](#) for customizing the image repository
2. In case of kubeadm is executed in the --dry-run mode, static Pods files are written in a temporary folder
3. Static Pod manifest generation for control plane components can be invoked individually with the [kubeadm init phase control-plane all](#) command

API server

The static Pod manifest for the API server is affected by following parameters provided by the users:

- The apiserver-advertise-address and apiserver-bind-port to bind to; if not provided, those value defaults to the IP address of the default network interface on the machine and port 6443
- The service-cluster-ip-range to use for services
- If an external etcd server is specified, the etcd-servers address and related TLS settings (etcd-cafile, etcd-certfile, etcd-keyfile); if an external etcd server is not be provided, a local etcd will be used (via host network)
- If a cloud provider is specified, the corresponding --cloud-provider is configured, together with the --cloud-config path if such file exists (this is experimental, alpha and will be removed in a future version)

Other API server flags that are set unconditionally are:

- --insecure-port=0 to avoid insecure connections to the api server
- --enable-bootstrap-token-auth=true to enable the BootstrapTokenAuthenticator authentication module. See [TLS Bootstrapping](#) for more details
- --allow-privileged to true (required e.g. by kube proxy)
- --requestheader-client-ca-file to front-proxy-ca.crt
- --enable-admission-plugins to:
 - [NamespaceLifecycle](#) e.g. to avoid deletion of system reserved namespaces
 - [LimitRanger](#) and [ResourceQuota](#) to enforce limits on namespaces
 - [ServiceAccount](#) to enforce service account automation
 - [PersistentVolumeLabel](#) attaches region or zone labels to PersistentVolumes as defined by the cloud provider (This admission controller is deprecated and will be removed in a future version. It is not deployed by kubeadm by default with v1.9 onwards when not explicitly opting into using gce or aws as cloud providers)
 - [DefaultStorageClass](#) to enforce default storage class on PersistentVolumeClaim objects
 - [DefaultTolerationSeconds](#)
 - [NodeRestriction](#) to limit what a kubelet can modify (e.g. only pods on this node)
- --kubelet-preferred-address-types to InternalIP,ExternalIP,Hostname; this makes kubectl logs and other API server-kubelet communication work in environments where the hostnames of the nodes aren't resolvable
- Flags for using certificates generated in previous steps:
 - --client-ca-file to ca.crt
 - --tls-cert-file to apiserver.crt
 - --tls-private-key-file to apiserver.key
 - --kubelet-client-certificate to apiserver-kubelet-client.crt
 - --kubelet-client-key to apiserver-kubelet-client.key
 - --service-account-key-file to sa.pub
 - --requestheader-client-ca-file to front-proxy-ca.crt

- --proxy-client-cert-file to front-proxy-client.crt
- --proxy-client-key-file to front-proxy-client.key
- Other flags for securing the front proxy ([API Aggregation](#)) communications:
 - --requestheader-username-headers=X-Remote-User
 - --requestheader-group-headers=X-Remote-Group
 - --requestheader-extra-headers-prefix=X-Remote-Extra-
 - --requestheader-allowed-names=front-proxy-client

Controller manager

The static Pod manifest for the controller manager is affected by following parameters provided by the users:

- If kubeadm is invoked specifying a --pod-network-cidr, the subnet manager feature required for some CNI network plugins is enabled by setting:
 - --allocate-node-cidrs=true
 - --cluster-cidr and --node-cidr-mask-size flags according to the given CIDR
- If a cloud provider is specified, the corresponding --cloud-provider is specified, together with the --cloud-config path if such configuration file exists (this is experimental, alpha and will be removed in a future version)

Other flags that are set unconditionally are:

- --controllers enabling all the default controllers plus BootstrapSigner and TokenCleaner controllers for TLS bootstrap. See [TLS Bootstrapping](#) for more details
- --use-service-account-credentials to true
- Flags for using certificates generated in previous steps:
 - --root-ca-file to ca.crt
 - --cluster-signing-cert-file to ca.crt, if External CA mode is disabled, otherwise to ""
 - --cluster-signing-key-file to ca.key, if External CA mode is disabled, otherwise to ""
 - --service-account-private-key-file to sa.key

Scheduler

The static Pod manifest for the scheduler is not affected by parameters provided by the users.

Generate static Pod manifest for local etcd

If you specified an external etcd this step will be skipped, otherwise kubeadm generates a static Pod manifest file for creating a local etcd instance running in a Pod with following attributes:

- listen on localhost:2379 and use HostNetwork=true
- make a hostPath mount out from the dataDir to the host's filesystem
- Any extra flags specified by the user

Please note that:

1. The etcd container image will be pulled from registry.gcr.io by default. See [using custom images](#) for customizing the image repository.
2. If you run kubeadm in --dry-run mode, the etcd static Pod manifest is written into a temporary folder.
3. You can directly invoke static Pod manifest generation for local etcd, using the [kubeadm init phase etcd local](#) command.

Wait for the control plane to come up

kubeadm waits (upto 4m0s) until localhost:6443/healthz (kube-apiserver liveness) returns ok. However in order to detect deadlock conditions, kubeadm fails fast if localhost:10255/healthz (kubelet liveness) or localhost:10255/healthz/syncloop (kubelet readiness) don't return ok within 40s and 60s respectively.

kubeadm relies on the kubelet to pull the control plane images and run them properly as static Pods. After the control plane is up, kubeadm completes the tasks described in following paragraphs.

Save the kubeadm ClusterConfiguration in a ConfigMap for later reference

kubeadm saves the configuration passed to kubeadm init in a ConfigMap named kubeadm-config under kube-system namespace.

This will ensure that kubeadm actions executed in future (e.g kubeadm upgrade) will be able to determine the actual/current cluster state and make new decisions based on that data.

Please note that:

1. Before saving the ClusterConfiguration, sensitive information like the token is stripped from the configuration
2. Upload of control plane node configuration can be invoked individually with the command [kubeadm init phase upload-config](#).

Mark the node as control-plane

As soon as the control plane is available, kubeadm executes following actions:

- Labels the node as control-plane with node-role.kubernetes.io/control-plane=""
- Taints the node with node-role.kubernetes.io/control-plane:NoSchedule

Please note that the phase to mark the control-plane phase can be invoked individually with the [kubeadm init phase mark-control-plane](#) command.

- Taints the node with node-role.kubernetes.io/master:NoSchedule and node-role.kubernetes.io/control-plane:NoSchedule

Please note that:

1. The node-role.kubernetes.io/master taint is deprecated and will be removed in kubeadm version 1.25

2. Mark control-plane phase can be invoked individually with the command [kubeadm init phase mark-control-plane](#)

Configure TLS-Bootstrapping for node joining

Kubeadm uses [Authenticating with Bootstrap Tokens](#) for joining new nodes to an existing cluster; for more details see also [design proposal](#).

kubeadm init ensures that everything is properly configured for this process, and this includes following steps as well as setting API server and controller flags as already described in previous paragraphs.

Please note that:

1. TLS bootstrapping for nodes can be configured with the command [kubeadm init phase bootstrap-token](#), executing all the configuration steps described in following paragraphs; alternatively, each step can be invoked individually

Create a bootstrap token

kubeadm init create a first bootstrap token, either generated automatically or provided by the user with the --token flag; as documented in bootstrap token specification, token should be saved as secrets with name bootstrap-token-<token-id> under kube-system namespace.

Please note that:

1. The default token created by kubeadm init will be used to validate temporary user during TLS bootstrap process; those users will be member of system:bootstrappers:kubeadm:default-node-token group
2. The token has a limited validity, default 24 hours (the interval may be changed with the --token-ttl flag)
3. Additional tokens can be created with the [kubeadm token](#) command, that provide as well other useful functions for token management.

Allow joining nodes to call CSR API

Kubeadm ensures that users in system:bootstrappers:kubeadm:default-node-token group are able to access the certificate signing API.

This is implemented by creating a ClusterRoleBinding named kubeadm:kubelet-bootstrap between the group above and the default RBAC role system:node-bootstrapper.

Set up auto approval for new bootstrap tokens

Kubeadm ensures that the Bootstrap Token will get its CSR request automatically approved by the csrapprover controller.

This is implemented by creating ClusterRoleBinding named kubeadm:node-autoapprove-bootstrap between the system:bootstrappers:kubeadm:default-node-token group and the default role system:certificates.k8s.io:certificatesigningrequests:nodeclient.

The role system:certificates.k8s.io:certificatesigningrequests:nodeclient should be created as well, granting POST permission to /apis/certificates.k8s.io/certificatesigningrequests/nodeclient.

Set up nodes certificate rotation with auto approval

Kubeadm ensures that certificate rotation is enabled for nodes, and that new certificate request for nodes will get its CSR request automatically approved by the csrapprover controller.

This is implemented by creating ClusterRoleBinding named kubeadm:node-autoapprove-certificate-rotation between the system:nodes group and the default role system:certificates.k8s.io:certificatesigningrequests:selfnodeclient.

Create the public cluster-info ConfigMap

This phase creates the cluster-info ConfigMap in the kube-public namespace.

Additionally it creates a Role and a RoleBinding granting access to the ConfigMap for unauthenticated users (i.e. users in RBAC group system:unauthenticated).

Please note that:

1. The access to the cluster-info ConfigMap *is not* rate-limited. This may or may not be a problem if you expose your cluster's API server to the internet; worst-case scenario here is a DoS attack where an attacker uses all the in-flight requests the kube-apiserver can handle to serving the cluster-info ConfigMap.

Install addons

Kubeadm installs the internal DNS server and the kube-proxy addon components via the API server.

Please note that:

1. This phase can be invoked individually with the command [kubeadm init phase addon all proxy](#).

A ServiceAccount for kube-proxy is created in the kube-system namespace; then kube-proxy is deployed as a DaemonSet:

- The credentials (ca.crt and token) to the control plane come from the ServiceAccount
- The location (URL) of the API server comes from a ConfigMap
- The kube-proxy ServiceAccount is bound to the privileges in the system:node-proxier ClusterRole

DNS

- The CoreDNS service is named kube-dns. This is done to prevent any interruption in service when the user is switching the cluster DNS from kube-dns to CoreDNS the --config method described [here](#).
- A ServiceAccount for CoreDNS is created in the kube-system namespace.
- The coredns ServiceAccount is bound to the privileges in the system:coredns ClusterRole

In Kubernetes version 1.21, support for using kube-dns with kubeadm was removed. You can use CoreDNS with kubeadm even when the related Service is named kube-dns.

kubeadm join phases internal design

Similarly to kubeadm init, also kubeadm join internal workflow consists of a sequence of atomic work tasks to perform.

This is split into discovery (having the Node trust the Kubernetes Master) and TLS bootstrap (having the Kubernetes Master trust the Node).

see [Authenticating with Bootstrap Tokens](#) or the corresponding [design proposal](#).

Preflight checks

kubeadm executes a set of preflight checks before starting the join, with the aim to verify preconditions and avoid common cluster startup problems.

Please note that:

1. kubeadm join preflight checks are basically a subset kubeadm init preflight checks
2. Starting from 1.24, kubeadm uses crictl to communicate to all known CRI endpoints.
3. Starting from 1.9, kubeadm provides support for joining nodes running on Windows; in that case, linux specific controls are skipped.
4. In any case the user can skip specific preflight checks (or eventually all preflight checks) with the --ignore-preflight-errors option.

Discovery cluster-info

There are 2 main schemes for discovery. The first is to use a shared token along with the IP address of the API server. The second is to provide a file (that is a subset of the standard kubeconfig file).

Shared token discovery

If kubeadm join is invoked with --discovery-token, token discovery is used; in this case the node basically retrieves the cluster CA certificates from the cluster-info ConfigMap in the kube-public namespace.

In order to prevent "man in the middle" attacks, several steps are taken:

- First, the CA certificate is retrieved via insecure connection (this is possible because kubeadm init granted access to cluster-info users for system:unauthenticated)
- Then the CA certificate goes through following validation steps:
 - Basic validation: using the token ID against a JWT signature
 - Pub key validation: using provided --discovery-token-ca-cert-hash. This value is available in the output of kubeadm init or can be calculated using standard tools (the hash is calculated over the bytes of the Subject Public Key Info (SPKI) object as in RFC7469). The --discovery-token-ca-cert-hash flag may be repeated multiple times to allow more than one public key.

- As a additional validation, the CA certificate is retrieved via secure connection and then compared with the CA retrieved initially

Please note that:

1. Pub key validation can be skipped passing --discovery-token-unsafe-skip-ca-verification flag; This weakens the kubeadm security model since others can potentially impersonate the Kubernetes Master.

File/https discovery

If kubeadm join is invoked with --discovery-file, file discovery is used; this file can be a local file or downloaded via an HTTPS URL; in case of HTTPS, the host installed CA bundle is used to verify the connection.

With file discovery, the cluster CA certificates is provided into the file itself; in fact, the discovery file is a kubeconfig file with only server and certificate-authority-data attributes set, as described in [kubeadm join](#) reference doc; when the connection with the cluster is established, kubeadm try to access the cluster-info ConfigMap, and if available, uses it.

TLS Bootstrap

Once the cluster info are known, the file bootstrap-kubelet.conf is written, thus allowing kubelet to do TLS Bootstrapping.

The TLS bootstrap mechanism uses the shared token to temporarily authenticate with the Kubernetes API server to submit a certificate signing request (CSR) for a locally created key pair.

The request is then automatically approved and the operation completes saving ca.crt file and kubelet.conf file to be used by kubelet for joining the cluster, whilebootstrap-kubelet.conf is deleted.

Please note that:

- The temporary authentication is validated against the token saved during the kubeadm init process (or with additional tokens created with kubeadm token)
- The temporary authentication resolve to a user member of system:bootstrappers:kubeadm:default-node-token group which was granted access to CSR api during the kubeadm init process
- The automatic CSR approval is managed by the csrapprover controller, according with configuration done the kubeadm init process

Command line tool (kubectl)

Kubernetes provides a command line tool for communicating with a Kubernetes cluster's [control plane](#), using the Kubernetes API.

This tool is named kubectl.

For configuration, kubectl looks for a file named config in the \$HOME/.kube directory. You can specify other [kubeconfig](#) files by setting the KUBECONFIG environment variable or by setting the [--kubeconfig](#) flag.

This overview covers kubectl syntax, describes the command operations, and provides common examples. For details about each command, including all the supported flags and subcommands, see the [kubectl](#) reference documentation.

For installation instructions, see [Installing kubectl](#); for a quick guide, see the [cheat sheet](#). If you're used to using the docker command-line tool, [kubectl for Docker Users](#) explains some equivalent commands for Kubernetes.

Syntax

Use the following syntax to run kubectl commands from your terminal window:

```
kubectl [command] [TYPE] [NAME] [flags]
```

where command, TYPE, NAME, and flags are:

- command: Specifies the operation that you want to perform on one or more resources, for example create, get, describe, delete.
- TYPE: Specifies the [resource type](#). Resource types are case-insensitive and you can specify the singular, plural, or abbreviated forms. For example, the following commands produce the same output:

```
kubectl get pod pod1  
kubectl get pods pod1  
kubectl get po pod1
```

- NAME: Specifies the name of the resource. Names are case-sensitive. If the name is omitted, details for all resources are displayed, for example kubectl get pods.

When performing an operation on multiple resources, you can specify each resource by type and name or specify one or more files:

- To specify resources by type and name:

- To group resources if they are all the same type: TYPE1 name1 name2 name<#>. Example: kubectl get pod example-pod1 example-pod2
- To specify multiple resource types individually: TYPE1/name1 TYPE1/name2 TYPE2/name3 TYPE<#>/name<#>. Example: kubectl get pod/example-pod1 replicationcontroller/example-rc1

- To specify resources with one or more files: -f file1 -f file2 -f file<#>

- [Use YAML rather than JSON](#) since YAML tends to be more user-friendly, especially for configuration files.
Example: kubectl get -f ./pod.yaml

- flags: Specifies optional flags. For example, you can use the `-s` or `--server` flags to specify the address and port of the Kubernetes API server.

Caution: Flags that you specify from the command line override default values and any corresponding environment variables.

If you need help, run `kubectl help` from the terminal window.

In-cluster authentication and namespace overrides

By default `kubectl` will first determine if it is running within a pod, and thus in a cluster. It starts by checking for the `KUBERNETES_SERVICE_HOST` and `KUBERNETES_SERVICE_PORT` environment variables and the existence of a service account token file at `/var/run/secrets/kubernetes.io/serviceaccount/token`. If all three are found in-cluster authentication is assumed.

To maintain backwards compatibility, if the `POD_NAMESPACE` environment variable is set during in-cluster authentication it will override the default namespace from the service account token. Any manifests or tools relying on namespace defaulting will be affected by this.

POD_NAMESPACE environment variable

If the `POD_NAMESPACE` environment variable is set, cli operations on namespaced resources will default to the variable value. For example, if the variable is set to `seattle`, `kubectl get pods` would return pods in the `seattle` namespace. This is because pods are a namespaced resource, and no namespace was provided in the command. Review the output of `kubectl api-resources` to determine if a resource is namespaced.

Explicit use of `--namespace <value>` overrides this behavior.

How `kubectl` handles ServiceAccount tokens

If:

- there is Kubernetes service account token file mounted at `/var/run/secrets/kubernetes.io/serviceaccount/token`, and
- the `KUBERNETES_SERVICE_HOST` environment variable is set, and
- the `KUBERNETES_SERVICE_PORT` environment variable is set, and
- you don't explicitly specify a namespace on the `kubectl` command line

then `kubectl` assumes it is running in your cluster. The `kubectl` tool looks up the namespace of that ServiceAccount (this is the same as the namespace of the Pod) and acts against that namespace. This is different from what happens outside of a cluster; when `kubectl` runs outside a cluster and you don't specify a namespace, the `kubectl` command acts against the namespace set for the current context in your client configuration. To change the default namespace for your `kubectl` you can use the following command:

```
kubectl config set-context --current --namespace=<namespace-name>
```

Operations

The following table includes short descriptions and the general syntax for all of the `kubectl` operations:

Operation	Syntax	Description
alpha	kubectl alpha SUBCOMMAND [flags]	List the available commands that correspond to alpha features, which are not enabled in Kubernetes clusters by default.
annotate	kubectl annotate (-f FILENAME TYPE NAME TYPE/NAME) KEY_1=VAL_1 ... KEY_N=VAL_N [--overwrite] [--all] [--resource-version=version] [flags]	Add or update the annotations of one or more resources.
api-resources	kubectl api-resources [flags]	List the API resources that are available.
api-versions	kubectl api-versions [flags]	List the API versions that are available.
apply	kubectl apply -f FILENAME [flags]	Apply a configuration change to a resource from a file or stdin.
attach	kubectl attach POD -c CONTAINER [-i] [-t] [flags]	Attach to a running container either to view the output stream or interact with the container (stdin).
auth	kubectl auth [flags] [options]	Inspect authorization.
autoscale	kubectl autoscale (-f FILENAME TYPE NAME TYPE/NAME) [--min=MINPODS] --max=MAXPODS [--cpu-percent=CPU] [flags]	Automatically scale the set of pods that are managed by a replication controller.
certificate	kubectl certificate SUBCOMMAND [options]	Modify certificate resources.
cluster-info	kubectl cluster-info [flags]	Display endpoint information about the master and services in the cluster.
completion	kubectl completion SHELL [options]	Output shell completion code for the specified shell (bash or zsh).
config	kubectl config SUBCOMMAND [flags]	Modifies kubeconfig files. See the individual subcommands for details.
convert	kubectl convert -f FILENAME [options]	Convert config files between different API versions. Both YAML and JSON formats are accepted. Note - requires kubectl-convert plugin to be installed.
cordon	kubectl cordon NODE [options]	Mark node as unschedulable.
cp	kubectl cp <file-spec-src> <file-spec-dest> [options]	Copy files and directories to and from containers.
create	kubectl create -f FILENAME [flags]	Create one or more resources from a file or stdin.
delete	kubectl delete (-f FILENAME TYPE [NAME / NAME -l label --all]) [flags]	Delete resources either from a file, stdin, or specifying label selectors, names, resource selectors, or resources.
describe		

Operation	Syntax	Description
	kubectl describe (-f FILENAME TYPE [NAME_PREFIX /NAME -l label]) [flags]	Display the detailed state of one or more resources.
diff	kubectl diff -f FILENAME [flags]	Diff file or stdin against live configuration.
drain	kubectl drain NODE [options]	Drain node in preparation for maintenance.
edit	kubectl edit (-f FILENAME TYPE NAME TYPE/NAME) [flags]	Edit and update the definition of one or more resources on the server by using the default editor.
events	kubectl events	List events
exec	kubectl exec POD [-c CONTAINER] [-i] [-t] [flags] [-- COMMAND [args...]]	Execute a command against a container in a pod.
explain	kubectl explain TYPE [--recursive=false] [flags]	Get documentation of various resources. For instance pods, nodes, services, etc.
expose	kubectl expose (-f FILENAME TYPE NAME TYPE/NAME) [--port=port] [--protocol=TCP UDP] [--target-port=number-or-name] [--name=name] [--external-ip=external-ip-of-service] [--type=type] [flags]	Expose a replication controller, service, or pod as a new Kubernetes service.
get	kubectl get (-f FILENAME TYPE [NAME /NAME -l label]) [--watch] [--sort-by=FIELD] [[-o --output]=OUTPUT_FORMAT] [flags]	List one or more resources.
kustomize	kubectl kustomize <dir> [flags] [options]	List a set of API resources generated from instructions in a kustomization.yaml file. The argument must be the path to the directory containing the file, or a git repository URL with a path suffix specifying same with respect to the repository root.
label	kubectl label (-f FILENAME TYPE NAME TYPE/NAME) KEY_1=VAL_1 ... KEY_N=VAL_N [-overwrite] [--all] [--resource-version=version] [flags]	Add or update the labels of one or more resources.
logs	kubectl logs POD [-c CONTAINER] [--follow] [flags]	Print the logs for a container in a pod.
options	kubectl options	List of global command-line options, which apply to all commands.
patch	kubectl patch (-f FILENAME TYPE NAME TYPE/NAME) --patch PATCH [flags]	Update one or more fields of a resource by using the strategic merge patch process.
plugin	kubectl plugin [flags] [options]	Provides utilities for interacting with plugins.
port-forward	kubectl port-forward POD [LOCAL_PORT:]REMOTE_PORT [...] [LOCAL_PORT_N:]REMOTE_PORT_N] [flags]	Forward one or more local ports to a pod.

Operation	Syntax	Description
proxy	kubectl proxy [--port=PORT] [--www=static-dir] [-www-prefix=prefix] [--api-prefix=prefix] [flags]	Run a proxy to the Kubernetes API server.
replace	kubectl replace -f FILENAME	Replace a resource from a file or stdin.
rollout	kubectl rollout SUBCOMMAND [options]	Manage the rollout of a resource. Valid resource types include: deployments, daemonsets and statefulsets.
run	kubectl run NAME --image=image [--env="key=value"] [--port=port] [--dry-run=server client none] [--overrides=inline-json] [flags]	Run a specified image on the cluster.
scale	kubectl scale (-f FILENAME TYPE NAME TYPE/NAME) --replicas=COUNT [--resource-version=version] [--current-replicas=count] [flags]	Update the size of the specified replication controller.
set	kubectl set SUBCOMMAND [options]	Configure application resources.
taint	kubectl taint NODE NAME KEY_1=VAL_1:TAINTEFFECT_1 ... KEY_N=VAL_N:TAINTEFFECT_N [options]	Update the taints on one or more nodes.
top	`kubectl top (POD	NODE) [flags] [options]`
uncordon	kubectl uncordon NODE [options]	Mark node as schedulable.
version	kubectl version [--client] [flags]	Display the Kubernetes version running on the client and server.
wait	kubectl wait ([-f FILENAME] resource.group/resource.name resource.group [(-l label --all)]) [--for=delete --for condition=available] [options]	Experimental: Wait for a specific condition on one or many resources.

To learn more about command operations, see the [kubectl](#) reference documentation.

Resource types

The following table includes a list of all the supported resource types and their abbreviated aliases.

(This output can be retrieved from kubectl api-resources, and was accurate as of Kubernetes 1.25.0)

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
bindings		v1	true	Binding
componentstatuses	cs	v1	false	ComponentStatus
configmaps	cm	v1	true	ConfigMap
endpoints	ep	v1	true	Endpoints
events	ev	v1	true	Event
limitranges	limits	v1	true	LimitRange
namespaces	ns	v1	false	Namespace
nodes	no	v1	false	Node
persistentvolumeclaims	pvc	v1	true	PersistentVolumeClaim

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
persistentvolumes	pv	v1	false	PersistentVolume
pods	po	v1	true	Pod
podtemplates		v1	true	PodTemplate
replicationcontrollers	rc	v1	true	ReplicationController
resourcequotas	quota	v1	true	ResourceQuota
secrets		v1	true	Secret
serviceaccounts	sa	v1	true	ServiceAccount
services	svc	v1	true	Service
mutatingwebhookconfigurations		admissionregistration.k8s.io/v1	false	MutatingWebhookConfiguration
validatingwebhookconfigurations		admissionregistration.k8s.io/v1	false	ValidatingWebhookConfiguration
customresourcedefinitions	crd,crds	apiextensions.k8s.io/v1	false	CustomResourceDefinition
apiservices		apiregistration.k8s.io/v1	false	APIService
controllerrevisions		apps/v1	true	ControllerRevision
daemonsets	ds	apps/v1	true	DaemonSet
deployments	deploy	apps/v1	true	Deployment
replicasets	rs	apps/v1	true	ReplicaSet
statefulsets	sts	apps/v1	true	StatefulSet
tokenreviews		authentication.k8s.io/v1	false	TokenReview
localsubjectaccessreviews		authorization.k8s.io/v1	true	LocalSubjectAccessReview
selfsubjectaccessreviews		authorization.k8s.io/v1	false	SelfSubjectAccessReview
selfsubjectrulesreviews		authorization.k8s.io/v1	false	SelfSubjectRulesReview
subjectaccessreviews		authorization.k8s.io/v1	false	SubjectAccessReview
horizontalpodautoscalers	hpa	autoscaling/v2	true	HorizontalPodAutoscaler
cronjobs	cj	batch/v1	true	CronJob
jobs		batch/v1	true	Job
certificatesigningrequests	csr	certificates.k8s.io/v1	false	CertificateSigningRequest
leases		coordination.k8s.io/v1	true	Lease
endpointslices		discovery.k8s.io/v1	true	EndpointSlice
events	ev	events.k8s.io/v1	true	Event
flowschemas		flowcontrol.apiserver.k8s.io/v1beta2	false	FlowSchema
prioritylevelconfigurations		flowcontrol.apiserver.k8s.io/v1beta2	false	PriorityLevelConfiguration
ingressclasses		networking.k8s.io/v1	false	IngressClass
ingresses	ing	networking.k8s.io/v1	true	Ingress
networkpolicies	netpol	networking.k8s.io/v1	true	NetworkPolicy
runtimemelasses		node.k8s.io/v1	false	RuntimeClass
poddisruptionbudgets	pdb	policy/v1	true	PodDisruptionBudget
podsecuritypolicies	psp	policy/v1beta1	false	PodSecurityPolicy
clusterrolebindings		rbac.authorization.k8s.io/v1	false	ClusterRoleBinding
clusterroles		rbac.authorization.k8s.io/v1	false	ClusterRole
rolebindings		rbac.authorization.k8s.io/v1	true	RoleBinding
roles		rbac.authorization.k8s.io/v1	true	Role

NAME	SHORTNAMES	APIVERSION	NAMESPACED	KIND
priorityclasses	pc	scheduling.k8s.io/v1	false	PriorityClass
csidrivers		storage.k8s.io/v1	false	CSIDriver
csinodes		storage.k8s.io/v1	false	CSINode
csistoragecapacities		storage.k8s.io/v1	true	CSIStorageCapacity
storageclasses	sc	storage.k8s.io/v1	false	StorageClass
volumeattachments		storage.k8s.io/v1	false	VolumeAttachment

Output options

Use the following sections for information about how you can format or sort the output of certain commands. For details about which commands support the various output options, see the [kubectl](#) reference documentation.

Formatting output

The default output format for all kubectl commands is the human readable plain-text format. To output details to your terminal window in a specific format, you can add either the `-o` or `--output` flags to a supported kubectl command.

Syntax

```
kubectl [command] [TYPE] [NAME] -o <output_format>
```

Depending on the kubectl operation, the following output formats are supported:

Output format	Description
<code>-o custom-columns=<spec></code>	Print a table using a comma separated list of custom columns .
<code>-o custom-columns-file=<filename></code>	Print a table using the custom columns template in the <code><filename></code> file.
<code>-o json</code>	Output a JSON formatted API object.
<code>-o jsonpath=<template></code>	Print the fields defined in a jsonpath expression.
<code>-o jsonpath-file=<filename></code>	Print the fields defined by the jsonpath expression in the <code><filename></code> file.
<code>-o name</code>	Print only the resource name and nothing else.
<code>-o wide</code>	Output in the plain-text format with any additional information. For pods, the node name is included.
<code>-o yaml</code>	Output a YAML formatted API object.

Example

In this example, the following command outputs the details for a single pod as a YAML formatted object:

```
kubectl get pod web-pod-13je7 -o yaml
```

Remember: See the [kubectl](#) reference documentation for details about which output format is supported by each command.

Custom columns

To define custom columns and output only the details that you want into a table, you can use the `custom-columns` option. You can choose to define the custom columns inline or use a template file: `-o custom-columns=<spec>` or `-o custom-columns-file=<filename>`.

Examples

Inline:

```
kubectl get pods <pod-name> -o custom-columns=NAME:.metadata.name,RSRC:.metadata.resourceVersion
```

Template file:

```
kubectl get pods <pod-name> -o custom-columns-file=template.txt
```

where the `template.txt` file contains:

```
NAME      RSRC  
metadata.name metadata.resourceVersion
```

The result of running either command is similar to:

```
NAME      RSRC  
submit-queue 610995
```

Server-side columns

`kubectl` supports receiving specific column information from the server about objects. This means that for any given resource, the server will return columns and rows relevant to that resource, for the client to print. This allows for consistent human-readable output across clients used against the same cluster, by having the server encapsulate the details of printing.

This feature is enabled by default. To disable it, add the `--server-print=false` flag to the `kubectl get` command.

Examples

To print information about the status of a pod, use a command like the following:

```
kubectl get pods <pod-name> --server-print=false
```

The output is similar to:

```
NAME      AGE  
pod-name  1m
```

Sorting list objects

To output objects to a sorted list in your terminal window, you can add the `--sort-by` flag to a supported `kubectl` command. Sort your objects by specifying any numeric or string field with the `--sort-by` flag. To specify a field, use a [jsonpath](#) expression.

Syntax

```
kubectl [command] [TYPE] [NAME] --sort-by=<jsonpath_exp>
```

Example

To print a list of pods sorted by name, you run:

```
kubectl get pods --sort-by=.metadata.name
```

Examples: Common operations

Use the following set of examples to help you familiarize yourself with running the commonly used kubectl operations:

kubectl apply - Apply or Update a resource from a file or stdin.

```
# Create a service using the definition in example-service.yaml.  
kubectl apply -f example-service.yaml
```

```
# Create a replication controller using the definition in example-controller.yaml.  
kubectl apply -f example-controller.yaml
```

```
# Create the objects that are defined in any .yaml, .yml, or .json file within the <directory>  
directory.
```

```
kubectl apply -f <directory>
```

kubectl get - List one or more resources.

```
# List all pods in plain-text output format.  
kubectl get pods
```

```
# List all pods in plain-text output format and include additional information (such as node  
name).
```

```
kubectl get pods -o wide
```

```
# List the replication controller with the specified name in plain-text output format. Tip: You  
can shorten and replace the 'replicationcontroller' resource type with the alias 'rc'.
```

```
kubectl get replicationcontroller <rc-name>
```

```
# List all replication controllers and services together in plain-text output format.  
kubectl get rc,services
```

```
# List all daemon sets in plain-text output format.  
kubectl get ds
```

```
# List all pods running on node server01  
kubectl get pods --field-selector=spec.nodeName=server01
```

kubectl describe - Display detailed state of one or more resources, including the uninitialized ones by default.

```
# Display the details of the node with name <node-name>.  
kubectl describe nodes <node-name>  
  
# Display the details of the pod with name <pod-name>.  
kubectl describe pods/<pod-name>  
  
# Display the details of all the pods that are managed by the replication controller named <rc-name>.  
# Remember: Any pods that are created by the replication controller get prefixed with the name  
of the replication controller.  
kubectl describe pods <rc-name>  
  
# Describe all pods  
kubectl describe pods
```

Note: The kubectl get command is usually used for retrieving one or more resources of the same resource type. It features a rich set of flags that allows you to customize the output format using the -o or --output flag, for example. You can specify the -w or --watch flag to start watching updates to a particular object. The kubectl describe command is more focused on describing the many related aspects of a specified resource. It may invoke several API calls to the API server to build a view for the user. For example, the kubectl describe node command retrieves not only the information about the node, but also a summary of the pods running on it, the events generated for the node etc.

kubectl delete - Delete resources either from a file, stdin, or specifying label selectors, names, resource selectors, or resources.

```
# Delete a pod using the type and name specified in the pod.yaml file.  
kubectl delete -f pod.yaml
```

```
# Delete all the pods and services that have the label '<label-key>=<label-value>'.  
kubectl delete pods,services -l <label-key>=<label-value>
```

```
# Delete all pods, including uninitialized ones.  
kubectl delete pods --all
```

kubectl exec - Execute a command against a container in a pod.

```
# Get output from running 'date' from pod <pod-name>. By default, output is from the first  
container.  
kubectl exec <pod-name> -- date
```

```
# Get output from running 'date' in container <container-name> of pod <pod-name>.  
kubectl exec <pod-name> -c <container-name> -- date
```

```
# Get an interactive TTY and run /bin/bash from pod <pod-name>. By default, output is from  
the first container.  
kubectl exec -ti <pod-name> -- /bin/bash
```

kubectl logs - Print the logs for a container in a pod.

```
# Return a snapshot of the logs from pod <pod-name>.  
kubectl logs <pod-name>
```

```
# Start streaming the logs from pod <pod-name>. This is similar to the 'tail -f' Linux command.  
kubectl logs -f <pod-name>
```

kubectl diff - View a diff of the proposed updates to a cluster.

```
# Diff resources included in "pod.json".  
kubectl diff -f pod.json
```

```
# Diff file read from stdin.  
cat service.yaml | kubectl diff -f -
```

Examples: Creating and using plugins

Use the following set of examples to help you familiarize yourself with writing and using kubectl plugins:

```
# create a simple plugin in any language and name the resulting executable file  
# so that it begins with the prefix "kubectl-"  
cat ./kubectl-hello
```

```
#!/bin/sh  
  
# this plugin prints the words "hello world"  
echo "hello world"
```

With a plugin written, let's make it executable:

```
chmod a+x ./kubectl-hello  
  
# and move it to a location in our PATH  
sudo mv ./kubectl-hello /usr/local/bin  
sudo chown root:root /usr/local/bin  
  
# You have now created and "installed" a kubectl plugin.  
# You can begin using this plugin by invoking it from kubectl as if it were a regular command  
kubectl hello
```

```
hello world
```

```
# You can "uninstall" a plugin, by removing it from the folder in your  
# $PATH where you placed it  
sudo rm /usr/local/bin/kubectl-hello
```

In order to view all of the plugins that are available to kubectl, use the kubectl plugin list subcommand:

```
kubectl plugin list
```

The output is similar to:

```
The following kubectl-compatible plugins are available:
```

```
/usr/local/bin/kubectl-hello
```

```
/usr/local/bin/kubectl-foo  
/usr/local/bin/kubectl-bar
```

kubectl plugin list also warns you about plugins that are not executable, or that are shadowed by other plugins; for example:

```
sudo chmod -x /usr/local/bin/kubectl-foo # remove execute permission  
kubectl plugin list
```

The following kubectl-compatible plugins are available:

```
/usr/local/bin/kubectl-hello  
/usr/local/bin/kubectl-foo  
  - warning: /usr/local/bin/kubectl-foo identified as a plugin, but it is not executable  
/usr/local/bin/kubectl-bar
```

error: one plugin warning was found

You can think of plugins as a means to build more complex functionality on top of the existing kubectl commands:

```
cat ./kubectl-whoami
```

The next few examples assume that you already made kubectl-whoami have the following contents:

```
#!/bin/bash  
  
# this plugin makes use of the `kubectl config` command in order to output  
# information about the current user, based on the currently selected context  
kubectl config view --template='{{ range .contexts }}{{ if eq .name "$(kubectl config current-context)" }}Current user: {{ printf "%s\n" .context.user }}{{ end }}{{ end }}'
```

Running the above command gives you an output containing the user for the current context in your KUBECONFIG file:

```
# make the file executable  
sudo chmod +x ./kubectl-whoami  
  
# and move it into your PATH  
sudo mv ./kubectl-whoami /usr/local/bin  
  
kubectl whoami  
Current user: plugins-user
```

What's next

- Read the kubectl reference documentation:
 - the kubectl [command reference](#)
 - the [command line arguments](#) reference
- Learn about [kubectl usage conventions](#)
- Read about [JSONPath support](#) in kubectl

- Read about how to [extend kubectl with plugins](#)
 - To find out more about plugins, take a look at the [example CLI plugin](#).

kubectl Cheat Sheet

This page contains a list of commonly used kubectl commands and flags.

Note: These instructions are for Kubernetes v1.28. To check the version, use the kubectl version command.

Kubectl autocomplete

BASH

```
source <(kubectl completion bash) # set up autocomplete in bash into the current shell, bash-completion package should be installed first.  
echo "source <(kubectl completion bash)" >> ~/.bashrc  
# add autocomplete permanently to your bash shell.
```

You can also use a shorthand alias for kubectl that also works with completion:

```
alias k=kubectl  
complete -o default -F __start_kubectl k
```

ZSH

```
source <(kubectl completion zsh) # set up autocomplete in zsh into the current shell  
echo '[[ $commands[kubectl] ]] && source <(kubectl completion zsh)' >> ~/.zshrc # add autocomplete permanently to your zsh shell
```

FISH

Require kubectl version 1.23 or above.

```
echo 'kubectl completion fish | source' >> ~/.config/fish/config.fish # add kubectl autocomplete permanently to your fish shell
```

A note on --all-namespaces

Appending --all-namespaces happens frequently enough that you should be aware of the shorthand for --all-namespaces:

```
kubectl -A
```

Kubectl context and configuration

Set which Kubernetes cluster kubectl communicates with and modifies configuration information. See [Authenticating Across Clusters with kubeconfig](#) documentation for detailed config file information.

```

kubectl config view # Show Merged kubeconfig settings.

# use multiple kubeconfig files at the same time and view merged config
KUBECONFIG=~/.kube/config:~/.kube/kubconfig2

kubectl config view

# get the password for the e2e user
kubectl config view -o jsonpath='{.users[?(@.name == "e2e")].user.password}'

kubectl config view -o jsonpath='{.users[].name}'    # display the first user
kubectl config view -o jsonpath='{.users[*].name}'   # get a list of users
kubectl config get-contexts                         # display list of contexts
kubectl config current-context                      # display the current-context
kubectl config use-context my-cluster-name         # set the default context to my-cluster-name

kubectl config set-cluster my-cluster-name          # set a cluster entry in the kubeconfig

# configure the URL to a proxy server to use for requests made by this client in the kubeconfig
kubectl config set-cluster my-cluster-name --proxy-url=my-proxy-url

# add a new user to your kubeconf that supports basic auth
kubectl config set-credentials kubeuser/foo.kubernetes.com --username=kubeuser --
password=kubepassword

# permanently save the namespace for all subsequent kubectl commands in that context.
kubectl config set-context --current --namespace=ggckad-s2

# set a context utilizing a specific username and namespace.
kubectl config set-context gce --user=cluster-admin --namespace=foo \
&& kubectl config use-context gce

kubectl config unset users.foo                      # delete user foo

# short alias to set/show context/namespace (only works for bash and bash-compatible shells,
# current context to be set before using kn to set namespace)
alias kx='f() { [ "$1" ] && kubectl config use-context $1 || kubectl config current-context ; } ; f'
alias kn='f() { [ "$1" ] && kubectl config set-context --current --namespace $1 || kubectl config
view --minify | grep namespace | cut -d" " -f6 ; } ; f'

```

Kubectl apply

apply manages applications through files defining Kubernetes resources. It creates and updates resources in a cluster through running kubectl apply. This is the recommended way of managing Kubernetes applications on production. See [Kubectl Book](#).

Creating objects

Kubernetes manifests can be defined in YAML or JSON. The file extension .yaml, .yml, and .json can be used.

```

kubectl apply -f ./my-manifest.yaml      # create resource(s)
kubectl apply -f ./my1.yaml -f ./my2.yaml  # create from multiple files
kubectl apply -f ./dir                  # create resource(s) in all manifest files in dir
kubectl apply -f https://git.io/vPieo    # create resource(s) from url
kubectl create deployment nginx --image=nginx # start a single instance of nginx

# create a Job which prints "Hello World"
kubectl create job hello --image=busybox:1.28 -- echo "Hello World"

# create a CronJob that prints "Hello World" every minute
kubectl create cronjob hello --image=busybox:1.28 --schedule="*/1 * * * *" -- echo "Hello
World"

kubectl explain pods                  # get the documentation for pod manifests

# Create multiple YAML objects from stdin
kubectl apply -f - <<EOF
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep
spec:
  containers:
  - name: busybox
    image: busybox:1.28
    args:
    - sleep
    - "1000000"
---
apiVersion: v1
kind: Pod
metadata:
  name: busybox-sleep-less
spec:
  containers:
  - name: busybox
    image: busybox:1.28
    args:
    - sleep
    - "1000"
EOF

# Create a secret with several keys
kubectl apply -f - <<EOF
apiVersion: v1
kind: Secret
metadata:
  name: mysecret
type: Opaque
data:
  password: $(echo -n "s33msi4" | base64 -w0)
  username: $(echo -n "jane" | base64 -w0)
EOF

```

Viewing and finding resources

```
# Get commands with basic output
kubectl get services          # List all services in the namespace
kubectl get pods -all-namespaces # List all pods in all namespaces
kubectl get pods -o wide        # List all pods in the current namespace, with more
                               details
kubectl get deployment my-dep   # List a particular deployment
kubectl get pods                # List all pods in the namespace
kubectl get pod my-pod -o yaml  # Get a pod's YAML

# Describe commands with verbose output
kubectl describe nodes my-node
kubectl describe pods my-pod

# List Services Sorted by Name
kubectl get services --sort-by=.metadata.name

# List pods Sorted by Restart Count
kubectl get pods --sort-by=.status.containerStatuses[0].restartCount

# List PersistentVolumes sorted by capacity
kubectl get pv --sort-by=.spec.capacity.storage

# Get the version label of all pods with label app=cassandra
kubectl get pods --selector=app=cassandra -o \
  jsonpath='{.items[*].metadata.labels.version}'

# Retrieve the value of a key with dots, e.g. 'ca.crt'
kubectl get configmap myconfig \
  -o jsonpath='{.data.ca\.crt}'

# Retrieve a base64 encoded value with dashes instead of underscores.
kubectl get secret my-secret --template='{{index .data "key-name-with-dashes"}}'

# Get all worker nodes (use a selector to exclude results that have a label
# named 'node-role.kubernetes.io/control-plane')
kubectl get node --selector='!node-role.kubernetes.io/control-plane'

# Get all running pods in the namespace
kubectl get pods --field-selector=status.phase=Running

# Get ExternalIPs of all nodes
kubectl get nodes -o jsonpath='{.items[*].status.addresses[?(@.type=="ExternalIP")].address}'

# List Names of Pods that belong to Particular RC
# "jq" command useful for transformations that are too complex for jsonpath, it can be found at
# https://jqlang.github.io/jq/
sel=$(kubectl get rc my-rc --output=json | jq -r '.spec.selector | to_entries | .[] | "\\"(.key)=\\
(.value),\"")%?'
echo $(kubectl get pods --selector=$sel --output=jsonpath={.items..metadata.name})
```

```

# Show labels for all pods (or any other Kubernetes object that supports labelling)
kubectl get pods --show-labels

# Check which nodes are ready
JSONPATH='{range .items[*]}{@.metadata.name}:{range @.status.conditions[*]}{@.type}
={@.status};{end}{end}' \
&& kubectl get nodes -o jsonpath="$JSONPATH" | grep "Ready=True"

# Check which nodes are ready with custom-columns
kubectl get node -o custom-columns='NODE_NAME:.metadata.name,STATUS:.status.conditions
[?(@.type=="Ready")].status'

# Output decoded secrets without external tools
kubectl get secret my-secret -o go-template='{{range $k,$v := .data}}{{"### "}}{{{$k}}}{{"\n"}}{{$v|
base64decode}}{{"\n\n"}}{{end}}'

# List all Secrets currently in use by a pod
kubectl get pods -o json | jq '.items[].spec.containers[].env[]?.valueFrom.secretKeyRef.name' |
grep -v null | sort | uniq

# List all containerIDs of initContainer of all pods
# Helpful when cleaning up stopped containers, while avoiding removal of initContainers.
kubectl get pods --all-namespaces -o jsonpath='{range .items[*].status.initContainerStatuses[*]}
{.containerID}"\n'{end}' | cut -d/ -f3

# List Events sorted by timestamp
kubectl get events --sort-by=.metadata.creationTimestamp

# List all warning events
kubectl events --types=Warning

# Compares the current state of the cluster against the state that the cluster would be in if the
manifest was applied.
kubectl diff -f ./my-manifest.yaml

# Produce a period-delimited tree of all keys returned for nodes
# Helpful when locating a key within a complex nested JSON structure
kubectl get nodes -o json | jq -c 'paths|join(".")

# Produce a period-delimited tree of all keys returned for pods, etc
kubectl get pods -o json | jq -c 'paths|join(".")

# Produce ENV for all pods, assuming you have a default container for the pods, default
namespace and the `env` command is supported.
# Helpful when running any supported command across all pods, not just `env`
for pod in $(kubectl get po --output=jsonpath={.items..metadata.name}); do echo $pod &&
kubectl exec -it $pod -- env; done

# Get a deployment's status subresource
kubectl get deployment nginx-deployment --subresource=status

```

Updating resources

```
kubectl set image deployment/frontend www=image:v2          # Rolling update "www"
containers of "frontend" deployment, updating the image
kubectl rollout history deployment/frontend               # Check the history of deployments
including the revision
kubectl rollout undo deployment/frontend                 # Rollback to the previous deployment
kubectl rollout undo deployment/frontend --to-revision=2   # Rollback to a specific revision
kubectl rollout status -w deployment/frontend           # Watch rolling update status of
"frontend" deployment until completion
kubectl rollout restart deployment/frontend            # Rolling restart of the "frontend"
deployment

cat pod.json | kubectl replace -f -                  # Replace a pod based on the JSON passed
into stdin

# Force replace, delete and then re-create the resource. Will cause a service outage.
kubectl replace --force ./pod.json

# Create a service for a replicated nginx, which serves on port 80 and connects to the containers
on port 8000
kubectl expose rc nginx --port=80 --target-port=8000

# Update a single-container pod's image version (tag) to v4
kubectl get pod mypod -o yaml | sed 's/^(image: myimage):.*$/1:v4/' | kubectl replace -f -

kubectl label pods my-pod new-label=awesome          # Add a Label
kubectl label pods my-pod new-label-                  # Remove a label
kubectl label pods my-pod new-label=new-value --overwrite  # Overwrite an existing value
kubectl annotate pods my-pod icon-url=http://goo.gl/XXBTWq  # Add an annotation
kubectl annotate pods my-pod icon-                   # Remove annotation
kubectl autoscale deployment foo --min=2 --max=10      # Auto scale a deployment "foo"
```

Patching resources

```
# Partially update a node
kubectl patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}'

# Update a container's image; spec.containers[*].name is required because it's a merge key
kubectl patch pod valid-pod -p '{"spec":{"containers":[{"name":"kubernetes-serv-
hostname","image":"new image"}]}}'

# Update a container's image using a json patch with positional arrays
kubectl patch pod valid-pod --type='json' -p='[{"op": "replace", "path": "/spec/containers/0/
image", "value":"new image"}]'

# Disable a deployment livenessProbe using a json patch with positional arrays
kubectl patch deployment valid-deployment --type json -p='[{"op": "remove", "path": "/spec/
template/spec/containers/0/livenessProbe"}]'

# Add a new element to a positional array
```

```
kubectl patch sa default --type='json' -p='[{"op": "add", "path": "/secrets/1", "value": {"name": "whatever" }}]'
```

```
# Update a deployment's replica count by patching its scale subresource  
kubectl patch deployment nginx-deployment --subresource='scale' --type='merge' -p '{"spec": {"replicas":2}}'
```

Editing resources

Edit any API resource in your preferred editor.

```
kubectl edit svc/docker-registry          # Edit the service named docker-registry  
KUBE_EDITOR="nano" kubectl edit svc/docker-registry # Use an alternative editor
```

Scaling resources

```
kubectl scale --replicas=3 rs/foo           # Scale a replicaset named 'foo' to 3  
kubectl scale --replicas=3 -f foo.yaml      # Scale a resource specified in "foo.yaml"  
to 3  
kubectl scale --current-replicas=2 --replicas=3 deployment/mysql # If the deployment named  
mysql's current size is 2, scale mysql to 3  
kubectl scale --replicas=5 rc/foo rc/bar rc/baz    # Scale multiple replication controllers
```

Deleting resources

```
kubectl delete -f ./pod.json               # Delete a pod using the type and name  
specified in pod.json  
kubectl delete pod unwanted --now          # Delete a pod with no grace period  
kubectl delete pod,service baz foo         # Delete pods and services with same  
names "baz" and "foo"  
kubectl delete pods,services -l name=myLabel # Delete pods and services with  
label name=myLabel  
kubectl -n my-ns delete pod,svc --all       # Delete all pods and services in  
namespace my-ns,  
# Delete all pods matching the awk pattern1 or pattern2  
kubectl get pods -n mynamespace --no-headers=true | awk '/pattern1|pattern2/{print $1}' |  
xargs kubectl delete -n mynamespace pod
```

Interacting with running Pods

```
kubectl logs my-pod                      # dump pod logs (stdout)  
kubectl logs -l name=myLabel (stdout)        # dump pod logs, with label name=myLabel  
kubectl logs my-pod --previous             # dump pod logs (stdout) for a previous  
instantiation of a container  
kubectl logs my-pod -c my-container        # dump pod container logs (stdout, multi-  
container case)  
kubectl logs -l name=myLabel -c my-container # dump pod logs, with label name=myLabel  
(stdout)  
kubectl logs my-pod -c my-container --previous # dump pod container logs (stdout, multi-
```

container case) for a previous instantiation of a container

```
kubectl logs -f my-pod          # stream pod logs (stdout)
kubectl logs -f my-pod -c my-container    # stream pod container logs (stdout, multi-
container case)
kubectl logs -f -l name=myLabel --all-containers  # stream all pods logs with label
name=myLabel (stdout)
kubectl run -i --tty busybox --image=busybox:1.28 -- sh # Run pod as interactive shell
kubectl run nginx --image=nginx -n mynamespace   # Start a single instance of nginx pod in
the namespace of mynamespace
kubectl run nginx --image=nginx --dry-run=client -o yaml > pod.yaml
# Generate spec for running pod nginx and write it into a file
called pod.yaml
kubectl attach my-pod -i          # Attach to Running Container
kubectl port-forward my-pod 5000:6000
# Listen on port 5000 on the local machine and forward to port 6000 on my-pod
kubectl exec my-pod -- ls /        # Run command in existing pod (1 container case)
kubectl exec --stdin --tty my-pod -- /bin/sh  # Interactive shell access to a running pod (1
container case)
kubectl exec my-pod -c my-container -- ls /      # Run command in existing pod (multi-
container case)
kubectl top pod POD_NAME --containers       # Show metrics for a given pod and its
containers
kubectl top pod POD_NAME --sort-by(cpu)      # Show metrics for a given pod and sort it
by 'cpu' or 'memory'
```

Copying files and directories to and from containers

```
kubectl cp /tmp/foo_dir my-pod:/tmp/bar_dir      # Copy /tmp/foo_dir local directory to /  
tmp/bar_dir in a remote pod in the current namespace  
kubectl cp /tmp/foo my-pod:/tmp/bar -c my-container # Copy /tmp/foo local file to /tmp/bar  
in a remote pod in a specific container  
kubectl cp /tmp/foo my-namespace/my-pod:/tmp/bar    # Copy /tmp/foo local file to /tmp/bar  
in a remote pod in namespace my-namespace  
kubectl cp my-namespace/my-pod:/tmp/foo /tmp/bar     # Copy /tmp/foo from a remote pod  
to /tmp/bar locally
```

Note: kubectl cp requires that the 'tar' binary is present in your container image. If 'tar' is not present, kubectl cp will fail. For advanced use cases, such as symlinks, wildcard expansion or file mode preservation consider using kubectl exec.

```
tar cf - /tmp/foo | kubectl exec -i -n my-namespace my-pod -- tar xf - -C /tmp/bar      #  
Copy /tmp/foo local file to /tmp/bar in a remote pod in namespace my-namespace  
kubectl exec -n my-namespace my-pod -- tar cf - /tmp/foo | tar xf - -C /tmp/bar  # Copy /tmp/  
foo from a remote pod to /tmp/bar locally
```

Interacting with Deployments and Services

```
kubectl logs deploy/my-deployment          # dump Pod logs for a Deployment (single-container case)
kubectl logs deploy/my-deployment -c my-container    # dump Pod logs for a Deployment (multi-container case)
```

```
kubectl port-forward svc/my-service 5000          # listen on local port 5000 and forward to  
port 5000 on Service backend  
kubectl port-forward svc/my-service 5000:my-service-port # listen on local port 5000 and  
forward to Service target port with name <my-service-port>  
  
kubectl port-forward deploy/my-deployment 5000:6000    # listen on local port 5000 and  
forward to port 6000 on a Pod created by <my-deployment>  
kubectl exec deploy/my-deployment -- ls           # run command in first Pod and first  
container in Deployment (single- or multi-container cases)
```

Interacting with Nodes and cluster

```
kubectl cordon my-node                # Mark my-node as unschedulable  
kubectl drain my-node               # Drain my-node in preparation for  
maintenance  
kubectl uncordon my-node            # Mark my-node as schedulable  
kubectl top node my-node           # Show metrics for a given node  
kubectl cluster-info                # Display addresses of the master and  
services  
kubectl cluster-info dump           # Dump current cluster state to stdout  
kubectl cluster-info dump --output-directory=/path/to/cluster-state # Dump current cluster  
state to /path/to/cluster-state  
  
# View existing taints on which exist on current nodes.  
kubectl get nodes -o='custom-  
columns=NodeName:.metadata.name,TaintKey:.spec.taints[*].key,TaintValue:.spec.taints[*].valu  
e,TaintEffect:.spec.taints[*].effect'  
  
# If a taint with that key and effect already exists, its value is replaced as specified.  
kubectl taint nodes foo dedicated=special-user:NoSchedule
```

Resource types

List all supported resource types along with their shortnames, [API group](#), whether they are [namespaced](#), and [kind](#):

```
kubectl api-resources
```

Other operations for exploring API resources:

```
kubectl api-resources --namespaced=true   # All namespaced resources  
kubectl api-resources --namespaced=false  # All non-namespaced resources  
kubectl api-resources -o name            # All resources with simple output (only the resource  
name)  
kubectl api-resources -o wide             # All resources with expanded (aka "wide") output  
kubectl api-resources --verbs=list,get  
# All resources that support the "list" and "get" request verbs  
kubectl api-resources --api-group=extensions # All resources in the "extensions" API group
```

Formatting output

To output details to your terminal window in a specific format, add the `-o` (or `--output`) flag to a supported kubectl command.

Output format	Description
<code>-o=custom-columns=<spec></code>	Print a table using a comma separated list of custom columns
<code>-o=custom-columns-file=<filename></code>	Print a table using the custom columns template in the <code><filename></code> file
<code>-o=go-template=<template></code>	Print the fields defined in a golang template
<code>-o=go-template-file=<filename></code>	Print the fields defined by the golang template in the <code><filename></code> file
<code>-o=json</code>	Output a JSON formatted API object
<code>-o=jsonpath=<template></code>	Print the fields defined in a jsonpath expression
<code>-o=jsonpath-file=<filename></code>	Print the fields defined by the jsonpath expression in the <code><filename></code> file
<code>-o=name</code>	Print only the resource name and nothing else
<code>-o=wide</code>	Output in the plain-text format with any additional information, and for pods, the node name is included
<code>-o=yaml</code>	Output a YAML formatted API object

Examples using `-o=custom-columns`:

```
# All images running in a cluster
kubectl get pods -A -o=custom-columns='DATA:spec.containers[*].image'

# All images running in namespace: default, grouped by Pod
kubectl get pods --namespace default --output=custom-columns="NAME:.metadata.name,IMAGE:.spec.containers[*].image"

# All images excluding "registry.k8s.io/coredns:1.6.2"
kubectl get pods -A -o=custom-columns='DATA:spec.containers[?(@.image!="registry.k8s.io/coredns:1.6.2")].image'

# All fields under metadata regardless of name
kubectl get pods -A -o=custom-columns='DATA:metadata.*'
```

More examples in the kubectl [reference documentation](#).

Kubectl output verbosity and debugging

Kubectl verbosity is controlled with the `-v` or `--v` flags followed by an integer representing the log level. General Kubernetes logging conventions and the associated log levels are described [here](#).

Verbosity	Description
<code>--v=0</code>	Generally useful for this to <i>always</i> be visible to a cluster operator.
<code>--v=1</code>	A reasonable default log level if you don't want verbosity.
<code>--v=2</code>	

Verbosity	Description
	Useful steady state information about the service and important log messages that may correlate to significant changes in the system. This is the recommended default log level for most systems.
--v=3	Extended information about changes.
--v=4	Debug level verbosity.
--v=5	Trace level verbosity.
--v=6	Display requested resources.
--v=7	Display HTTP request headers.
--v=8	Display HTTP request contents.
--v=9	Display HTTP request contents without truncation of contents.

What's next

- Read the [kubectl overview](#) and learn about [JsonPath](#).
- See [kubectl](#) options.
- Also read [kubectl Usage Conventions](#) to understand how to use kubectl in reusable scripts.
- See more community [kubectl cheatsheets](#).

kubectl

Synopsis

kubectl controls the Kubernetes cluster manager.

Find more information at: <https://kubernetes.io/docs/reference/kubectl/overview/>

kubectl [flags]

Options

--add-dir-header	
--alsologtostderr	If true, adds the file directory to the header of the log messages
--as string	log to standard error as well as files
--as-group stringArray	Username to impersonate for the operation
--azure-container-registry-config string	Group to impersonate for the operation, this flag can be repeated to specify multiple groups.
--cache-dir string Default: "\$HOME/.kube/cache"	Path to the file containing Azure container registry configuration information.
	Default cache directory

--certificate-authority string
Path to a cert file for the certificate authority
--client-certificate string
Path to a client certificate file for TLS
--client-key string
Path to a client key file for TLS
--cloud-provider-gce-l7lb-src-cidrs cidrs Default: 130.211.0.0/22,35.191.0.0/16
CIDRs opened in GCE firewall for L7 LB traffic proxy & health checks
--cloud-provider-gce-lb-src-cidrs cidrs Default: 130.211.0.0/22,209.85.152.0/22,209.85.204.0/22,35.191.0.0/16
CIDRs opened in GCE firewall for L4 LB traffic proxy & health checks
--cluster string
The name of the kubeconfig cluster to use
--context string
The name of the kubeconfig context to use
--default-not-ready-toleration-seconds int Default: 300
Indicates the tolerationSeconds of the toleration for notReady:NoExecute that is added by default to every pod that does not already have such a toleration.
--default-unreachable-toleration-seconds int Default: 300
Indicates the tolerationSeconds of the toleration for unreachable:NoExecute that is added by default to every pod that does not already have such a toleration.
-h, --help
help for kubectl
--insecure-skip-tls-verify
If true, the server's certificate will not be checked for validity. This will make your HTTPS connections insecure
--kubeconfig string
Path to the kubeconfig file to use for CLI requests.
--log-backtrace-at traceLocation Default: :0
when logging hits line file:N, emit a stack trace
--log-dir string
If non-empty, write log files in this directory
--log-file string
If non-empty, use this log file
--log-file-max-size uint Default: 1800
Defines the maximum size a log file can grow to. Unit is megabytes. If the value is 0, the maximum file size is unlimited.
--log-flush-frequency duration Default: 5s
Maximum number of seconds between log flushes
--logtostderr Default: true
log to standard error instead of files
--match-server-version
Require server version to match client version
-n, --namespace string
If present, the namespace scope for this CLI request
--one-output

--log-level string	If true, only write logs to their native severity level (vs also writing to each lower severity level)
--password string	Password for basic authentication to the API server
--profile string Default: "none"	Name of profile to capture. One of (none cpu heap goroutine threadcreate block mutex)
--profile-output string Default: "profile.pprof"	Name of the file to write the profile to
--request-timeout string Default: "0"	The length of time to wait before giving up on a single server request. Non-zero values should contain a corresponding time unit (e.g. 1s, 2m, 3h). A value of zero means don't timeout requests.
-s, --server string	The address and port of the Kubernetes API server
--skip-headers	If true, avoid header prefixes in the log messages
--skip-log-headers	If true, avoid headers when opening log files
--stderrthreshold severity Default: 2	logs at or above this threshold go to stderr
--tls-server-name string	Server name to use for server certificate validation. If it is not provided, the hostname used to contact the server is used
--token string	Bearer token for authentication to the API server
--user string	The name of the kubeconfig user to use
--username string	Username for basic authentication to the API server
-v, --v Level	number for the log level verbosity
--version version[=true]	Print version information and quit
--vmodule moduleSpec	comma-separated list of pattern=N settings for file-filtered logging
--warnings-as-errors	Treat warnings received from the server as errors and exit with a non-zero exit code

Environment variables

KUBECONFIG	Path to the kubectl configuration ("kubeconfig") file. Default: "\$HOME/.kube/config"
KUBECTL_COMMAND_HEADERS	When set to false, turns off extra HTTP headers detailing invoked kubectl command (Kubernetes version v1.22 or later)
KUBECTL_EXPLAIN_OPENAPIV3	

Toggles whether calls to `kubectl explain` use the new OpenAPIv3 data source available. OpenAPIv3 is enabled by default since Kubernetes 1.24.
--

KUBECTL_ENABLE_CMD_SHADOW

When set to true, external plugins can be used as subcommands for builtin commands if subcommand does not exist. In alpha stage, this feature can only be used for create command(e.g. kubectl create networkpolicy).

KUBECTL_INTERACTIVE_DELETE

When set to true, the --interactive flag in the kubectl delete command will be activated, allowing users to preview and confirm resources before proceeding to delete by passing this flag.

See Also

- [kubectl annotate](#) - Update the annotations on a resource
- [kubectl api-resources](#) - Print the supported API resources on the server
- [kubectl api-versions](#) - Print the supported API versions on the server, in the form of "group/version"
- [kubectl apply](#) - Apply a configuration to a resource by filename or stdin
- [kubectl attach](#) - Attach to a running container
- [kubectl auth](#) - Inspect authorization
- [kubectl autoscale](#) - Auto-scale a Deployment, ReplicaSet, or ReplicationController
- [kubectl certificate](#) - Modify certificate resources.
- [kubectl cluster-info](#) - Display cluster info
- [kubectl completion](#) - Output shell completion code for the specified shell (bash or zsh)
- [kubectl config](#) - Modify kubeconfig files
- [kubectl cordon](#) - Mark node as unschedulable
- [kubectl cp](#) - Copy files and directories to and from containers.
- [kubectl create](#) - Create a resource from a file or from stdin.
- [kubectl debug](#) - Create debugging sessions for troubleshooting workloads and nodes
- [kubectl delete](#) - Delete resources by filenames, stdin, resources and names, or by resources and label selector
- [kubectl describe](#) - Show details of a specific resource or group of resources
- [kubectl diff](#) - Diff live version against would-be applied version
- [kubectl drain](#) - Drain node in preparation for maintenance
- [kubectl edit](#) - Edit a resource on the server
- [kubectl events](#) - List events
- [kubectl exec](#) - Execute a command in a container
- [kubectl explain](#) - Documentation of resources
- [kubectl expose](#) - Take a replication controller, service, deployment or pod and expose it as a new Kubernetes Service
- [kubectl get](#) - Display one or many resources
- [kubectl kustomize](#) - Build a kustomization target from a directory or a remote url.
- [kubectl label](#) - Update the labels on a resource
- [kubectl logs](#) - Print the logs for a container in a pod
- [kubectl options](#) - Print the list of flags inherited by all commands
- [kubectl patch](#) - Update field(s) of a resource
- [kubectl plugin](#) - Provides utilities for interacting with plugins.
- [kubectl port-forward](#) - Forward one or more local ports to a pod
- [kubectl proxy](#) - Run a proxy to the Kubernetes API server
- [kubectl replace](#) - Replace a resource by filename or stdin
- [kubectl rollout](#) - Manage the rollout of a resource

- [kubectl run](#) - Run a particular image on the cluster
- [kubectl scale](#) - Set a new size for a Deployment, ReplicaSet or Replication Controller
- [kubectl set](#) - Set specific features on objects
- [kubectl taint](#) - Update the taints on one or more nodes
- [kubectl top](#) - Display Resource (CPU/Memory/Storage) usage.
- [kubectl uncordon](#) - Mark node as schedulable
- [kubectl version](#) - Print the client and server version information
- [kubectl wait](#) - Experimental: Wait for a specific condition on one or many resources.

JSONPath Support

Kubectl supports JSONPath template.

JSONPath template is composed of JSONPath expressions enclosed by curly braces {}. Kubectl uses JSONPath expressions to filter on specific fields in the JSON object and format the output. In addition to the original JSONPath template syntax, the following functions and syntax are valid:

1. Use double quotes to quote text inside JSONPath expressions.
2. Use the range, end operators to iterate lists.
3. Use negative slice indices to step backwards through a list. Negative indices do not "wrap around" a list and are valid as long as $-index + listLength \geq 0$.

Note:

- The \$ operator is optional since the expression always starts from the root object by default.
- The result object is printed as its String() function.

Given the JSON input:

```
{
  "kind": "List",
  "items": [
    {
      "kind": "None",
      "metadata": {
        "name": "127.0.0.1",
        "labels": {
          "kubernetes.io/hostname": "127.0.0.1"
        }
      },
      "status": {
        "capacity": {"cpu": "4"},
        "addresses": [{"type": "LegacyHostIP", "address": "127.0.0.1"}]
      }
    },
    {
      "kind": "None",
      "metadata": {"name": "127.0.0.2"},
      "status": {
        "capacity": {"cpu": "4"}
      }
    }
  ]
}
```

```

"capacity":{"cpu":"8"},
"addresses":[
    {"type": "LegacyHostIP", "address":"127.0.0.2"},
    {"type": "another", "address":"127.0.0.3"}
]
}
],
"users":[
{
    "name": "myself",
    "user": {}
},
{
    "name": "e2e",
    "user": {"username": "admin", "password": "secret"}
}
]
}

```

Function	Description	Example	Result
text	the plain text	kind is {.kind}	kind is List
@	the current object	{@}	the same as input
. or []	child operator	{.kind}, {[kind]} or {[name\type]}	List
..	recursive descent	{..name}	127.0.0.1 127.0.0.2 myself e2e
*	wildcard. Get all objects	{.items[*].metadata.name}	[127.0.0.1 127.0.0.2]
[start:end:step]	subscript operator	{.users[0].name}	myself
[,]	union operator	{.items[*]['metadata.name', 'status.capacity']}	map[cpu:4] map[cpu:8]
?()	filter	{.users[?(@.name=="e2e")].user.password}	secret
range, end	iterate list	{range .items[*]}{.metadata.name}, {.status.capacity} {end}	[127.0.0.1, map[cpu:4]] [127.0.0.2, map[cpu:8]]
"	quote interpreted string	{range .items[*]}{.metadata.name}{\t}{end}	127.0.0.1 127.0.0.2
\	escape termination character	{.items[0].metadata.labels.kubernetes\\.io/hostname}	127.0.0.1

Examples using kubectl and JSONPath expressions:

```

kubectl get pods -o json
kubectl get pods -o=jsonpath='{@}'
kubectl get pods -o=jsonpath='{.items[0]}'

```

```
kubectl get pods -o=jsonpath='{.items[0].metadata.name}'  
kubectl get pods -o=jsonpath="{.items[*]['metadata.name', 'status.capacity']}"  
kubectl get pods -o=jsonpath='{range .items[*]}.metadata.name{"\t"}{.status.startTime}{\"\n\"}  
{end}'  
kubectl get pods -o=jsonpath='{.items[0].metadata.labels.kubernetes\\.io/hostname}'
```

Note:

On Windows, you must *double* quote any JSONPath template that contains spaces (not single quote as shown above for bash). This in turn means that you must use a single quote or escaped double quote around any literals in the template. For example:

```
kubectl get pods -o=jsonpath="{range .items[*]}.metadata.name{\\"t\"}{.status.startTime}{\"\n\"}  
{end}"  
kubectl get pods -o=jsonpath="{range .items[*]}.metadata.name{\\"\\t\\\"}{.status.startTime}{\"\n\"}  
{end}"
```

Note:

JSONPath regular expressions are not supported. If you want to match using regular expressions, you can use a tool such as jq.

```
# kubectl does not support regular expressions for JSONpath output  
# The following command does not work  
kubectl get pods -o jsonpath='{.items[?(@.metadata.name=~/^test$)].metadata.name}'  
  
# The following command achieves the desired result  
kubectl get pods -o json | jq -r '.items[] | select(.metadata.name | test("test-")).metadata.name'
```

kubectl for Docker Users

You can use the Kubernetes command line tool kubectl to interact with the API Server. Using kubectl is straightforward if you are familiar with the Docker command line tool. However, there are a few differences between the Docker commands and the kubectl commands. The following sections show a Docker sub-command and describe the equivalent kubectl command.

docker run

To run an nginx Deployment and expose the Deployment, see [kubectl create deployment](#).
docker:

```
docker run -d --restart=always -e DOMAIN=cluster --name nginx-app -p 80:80 nginx
```

```
55c103fa129692154a7652490236fee9be47d70a8dd562281ae7d2f9a339a6db
```

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
55c103fa1296	nginx	"nginx -g 'daemon of..."	9 seconds ago
0.0.0.0:80->80/tcp	nginx-app		Up 9 seconds

kubectl:

```
# start the pod running nginx
kubectl create deployment --image=nginx nginx-app
```

```
deployment.apps/nginx-app created
```

```
# add env to nginx-app
kubectl set env deployment/nginx-app DOMAIN=cluster
```

```
deployment.apps/nginx-app env updated
```

Note: kubectl commands print the type and name of the resource created or mutated, which can then be used in subsequent commands. You can expose a new Service after a Deployment is created.

```
# expose a port through with a service
kubectl expose deployment nginx-app --port=80 --name=nginx-http
```

```
service "nginx-http" exposed
```

By using kubectl, you can create a [Deployment](#) to ensure that N pods are running nginx, where N is the number of replicas stated in the spec and defaults to 1. You can also create a [service](#) with a selector that matches the pod labels. For more information, see [Use a Service to Access an Application in a Cluster](#).

By default images run in the background, similar to docker run -d To run things in the foreground, use [kubectl run](#) to create pod:

```
kubectl run [-i] [--tty] --attach <name> --image=<image>
```

Unlike docker run ..., if you specify --attach, then you attach stdin, stdout and stderr. You cannot control which streams are attached (docker -a ...). To detach from the container, you can type the escape sequence Ctrl+P followed by Ctrl+Q.

docker ps

To list what is currently running, see [kubectl get](#).

docker:

```
docker ps -a
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
14636241935f ago	ubuntu:16.04	"echo test"	5 seconds ago
55c103fa1296	cocky_fermi		Exited (0) 5 seconds
55c103fa1296	nginx	"nginx -g 'daemon off...'"	About a minute ago
	0.0.0.0:80->80/tcp	nginx-app	Up About a minute

kubectl:

```
kubectl get po
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-app-8df569cb7-4gd89	1/1	Running	0	3m
ubuntu	0/1	Completed	0	20s

docker attach

To attach a process that is already running in a container, see [kubectl attach](#).

docker:

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
55c103fa1296	nginx	"nginx -g 'daemon of..."	5 minutes ago
0.0.0.0:80->80/tcp	nginx-app		Up 5 minutes

```
docker attach 55c103fa1296
```

```
...
```

kubectl:

```
kubectl get pods
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-app-5jyvm	1/1	Running	0	10m

```
kubectl attach -it nginx-app-5jyvm
```

```
...
```

To detach from the container, you can type the escape sequence Ctrl+P followed by Ctrl+Q.

docker exec

To execute a command in a container, see [kubectl exec](#).

docker:

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED
STATUS	PORTS	NAMES	
55c103fa1296	nginx	"nginx -g 'daemon of..."	6 minutes ago
0.0.0.0:80->80/tcp	nginx-app		Up 6 minutes

```
docker exec 55c103fa1296 cat /etc/hostname
```

```
55c103fa1296
```

kubectl:

```
kubectl get po
```

```
NAME      READY  STATUS  RESTARTS  AGE
nginx-app-5jyvm  1/1    Running   0          10m
```

```
kubectl exec nginx-app-5jyvm -- cat /etc/hostname
```

```
nginx-app-5jyvm
```

To use interactive commands.

docker:

```
docker exec -ti 55c103fa1296 /bin/sh
# exit
```

kubectl:

```
kubectl exec -ti nginx-app-5jyvm -- /bin/sh
# exit
```

For more information, see [Get a Shell to a Running Container](#).

docker logs

To follow stdout/stderr of a process that is running, see [kubectl logs](#).

docker:

```
docker logs -f a9e
```

```
192.168.9.1 - - [14/Jul/2015:01:04:02 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.35.0" "-"
192.168.9.1 - - [14/Jul/2015:01:04:03 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.35.0" "-"
```

kubectl:

```
kubectl logs -f nginx-app-zibvs
```

```
10.240.63.110 - - [14/Jul/2015:01:09:01 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.26.0" "-"
10.240.63.110 - - [14/Jul/2015:01:09:02 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.26.0" "-"
```

There is a slight difference between pods and containers; by default pods do not terminate if their processes exit. Instead the pods restart the process. This is similar to the docker run option `--restart=always` with one major difference. In docker, the output for each invocation of the process is concatenated, but for Kubernetes, each invocation is separate. To see the output from a previous run in Kubernetes, do this:

```
kubectl logs --previous nginx-app-zibvs
```

```
10.240.63.110 - - [14/Jul/2015:01:09:01 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.26.0" "-"
10.240.63.110 - - [14/Jul/2015:01:09:02 +0000] "GET / HTTP/1.1" 200 612 "-" "curl/7.26.0" "-"
```

For more information, see [Logging Architecture](#).

docker stop and docker rm

To stop and delete a running process, see [kubectl delete](#).

docker:

```
docker ps
```

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS	NAMES			
a9ec34d98787	nginx	"nginx -g 'daemon off;"	22 hours ago	Up 22 hours
0.0.0.0:80->80/tcp, 443/tcp	nginx-app			

```
docker stop a9ec34d98787
```

```
a9ec34d98787
```

```
docker rm a9ec34d98787
```

```
a9ec34d98787
```

kubectl:

```
kubectl get deployment nginx-app
```

NAME	READY	UP-TO-DATE	AVAILABLE	AGE
nginx-app	1/1	1	1	2m

```
kubectl get po -l app=nginx-app
```

NAME	READY	STATUS	RESTARTS	AGE
nginx-app-2883164633-aklf7	1/1	Running	0	2m

```
kubectl delete deployment nginx-app
```

```
deployment "nginx-app" deleted
```

```
kubectl get po -l app=nginx-app
```

```
# Return nothing
```

Note: When you use kubectl, you don't delete the pod directly. You have to first delete the Deployment that owns the pod. If you delete the pod directly, the Deployment recreates the pod.

docker login

There is no direct analog of docker login in kubectl. If you are interested in using Kubernetes with a private registry, see [Using a Private Registry](#).

docker version

To get the version of client and server, see [kubectl version](#).

docker:

```
docker version
```

```
Client version: 1.7.0
Client API version: 1.19
Go version (client): go1.4.2
Git commit (client): 0baf609
OS/Arch (client): linux/amd64
Server version: 1.7.0
Server API version: 1.19
Go version (server): go1.4.2
Git commit (server): 0baf609
OS/Arch (server): linux/amd64
```

kubectl:

```
kubectl version
```

```
Client Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.9+a3d1dfa6f4335",
GitCommit:"9b77fed11a9843ce3780f70dd251e92901c43072", GitTreeState:"dirty",
BuildDate:"2017-08-29T20:32:58Z", OpenPaasKubernetesVersion:"v1.03.02", GoVersion:"go1.7.5",
Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"6", GitVersion:"v1.6.9+a3d1dfa6f4335",
GitCommit:"9b77fed11a9843ce3780f70dd251e92901c43072", GitTreeState:"dirty",
BuildDate:"2017-08-29T20:32:58Z", OpenPaasKubernetesVersion:"v1.03.02", GoVersion:"go1.7.5",
Compiler:"gc", Platform:"linux/amd64"}
```

docker info

To get miscellaneous information about the environment and configuration, see [kubectl cluster-info](#).

docker:

```
docker info
```

```
Containers: 40
Images: 168
Storage Driver: aufs
Root Dir: /usr/local/google/docker/aufs
Backing Filesystem: extfs
Dirs: 248
Dirperm1 Supported: false
Execution Driver: native-0.2
Logging Driver: json-file
Kernel Version: 3.13.0-53-generic
Operating System: Ubuntu 14.04.2 LTS
CPUs: 12
Total Memory: 31.32 GiB
Name: k8s-is-fun.mtv.corp.google.com
ID: ADUV:GCYR:B3VJ:HMPO:LNPQ:KD5S:YKFQ:76VN:IANZ:7TFV:ZBF4:BYJO
WARNING: No swap limit support
```

kubectl:

```
kubectl cluster-info
```

```
Kubernetes master is running at https://203.0.113.141
KubeDNS is running at https://203.0.113.141/api/v1/namespaces/kube-system/services/kube-dns/proxy
kubernetes-dashboard is running at https://203.0.113.141/api/v1/namespaces/kube-system/services/kubernetes-dashboard/proxy
Grafana is running at https://203.0.113.141/api/v1/namespaces/kube-system/services/monitoring-grafana/proxy
Heapster is running at https://203.0.113.141/api/v1/namespaces/kube-system/services/monitoring-heapster/proxy
InfluxDB is running at https://203.0.113.141/api/v1/namespaces/kube-system/services/monitoring-influxdb/proxy
```

kubectl Usage Conventions

Recommended usage conventions for kubectl.

Using kubectl in Reusable Scripts

For a stable output in a script:

- Request one of the machine-oriented output forms, such as `-o name`, `-o json`, `-o yaml`, `-o go-template`, or `-o jsonpath`.
- Fully-qualify the version. For example, `jobs.v1.batch/myjob`. This will ensure that kubectl does not use its default version that can change over time.
- Don't rely on context, preferences, or other implicit states.

Subresources

- You can use the `--subresource` beta flag for kubectl commands like `get`, `patch`, `edit` and `replace` to fetch and update subresources for all resources that support them. Currently, only the `status` and `scale` subresources are supported.
 - For kubectl `edit`, the `scale` subresource is not supported. If you use `--subresource` with kubectl `edit` and specify `scale` as the subresource, the command will error out.
- The API contract against a subresource is identical to a full resource. While updating the `status` subresource to a new value, keep in mind that the subresource could be potentially reconciled by a controller to a different value.

Best Practices

kubectl run

For kubectl run to satisfy infrastructure as code:

- Tag the image with a version-specific tag and don't move that tag to a new version. For example, use `:v1234`, `v1.2.3`, `r03062016-1-4`, rather than `:latest` (For more information, see [Best Practices for Configuration](#)).

- Check in the script for an image that is heavily parameterized.
- Switch to configuration files checked into source control for features that are needed, but not expressible via kubectl run flags.

You can use the `--dry-run=client` flag to preview the object that would be sent to your cluster, without really submitting it.

kubectl apply

- You can use kubectl apply to create or update resources. For more information about using kubectl apply to update resources, see [Kubectl Book](#).

Component tools

Feature Gates

[Feature Gates \(removed\)](#)

[kubelet](#)

[kube-apiserver](#)

[kube-controller-manager](#)

[kube-proxy](#)

[kube-scheduler](#)

Feature Gates

This page contains an overview of the various feature gates an administrator can specify on different Kubernetes components.

See [feature stages](#) for an explanation of the stages for a feature.

Overview

Feature gates are a set of key=value pairs that describe Kubernetes features. You can turn these features on or off using the `--feature-gates` command line flag on each Kubernetes component.

Each Kubernetes component lets you enable or disable a set of feature gates that are relevant to that component. Use `-h` flag to see a full set of feature gates for all components. To set feature gates for a component, such as kubelet, use the `--feature-gates` flag assigned to a list of feature pairs:

```
--feature-gates=...,GracefulNodeShutdown=true
```

The following tables are a summary of the feature gates that you can set on different Kubernetes components.

- The "Since" column contains the Kubernetes release when a feature is introduced or its release stage is changed.
- The "Until" column, if not empty, contains the last Kubernetes release in which you can still use a feature gate.
- If a feature is in the Alpha or Beta state, you can find the feature listed in the [Alpha/Beta feature gate table](#).
- If a feature is stable you can find all stages for that feature listed in the [Graduated/Deprecated feature gate table](#).
- The [Graduated/Deprecated feature gate table](#) also lists deprecated and withdrawn features.

Note: For a reference to old feature gates that are removed, please refer to [feature gates removed](#).

Feature gates for Alpha or Beta features

Feature gates for features in Alpha or Beta states

Feature	Default	Stage	Since	Until
APIListChunking	false	Alpha	1.8	1.8
APIListChunking	true	Beta	1.9	
APIPriorityAndFairness	false	Alpha	1.18	1.19
APIPriorityAndFairness	true	Beta	1.20	
APIResponseCompression	false	Alpha	1.7	1.15
APIResponseCompression	true	Beta	1.16	
APIServerIdentity	false	Alpha	1.20	1.25
APIServerIdentity	true	Beta	1.26	
APIServerTracing	false	Alpha	1.22	1.26
APIServerTracing	true	Beta	1.27	
AdmissionWebhookMatchConditions	false	Alpha	1.27	1.27
AdmissionWebhookMatchConditions	true	Beta	1.28	
AggregatedDiscoveryEndpoint	false	Alpha	1.26	1.26
AggregatedDiscoveryEndpoint	true	Beta	1.27	
AnyVolumeDataSource	false	Alpha	1.18	1.23
AnyVolumeDataSource	true	Beta	1.24	
AppArmor	true	Beta	1.4	
CPUManagerPolicyAlphaOptions	false	Alpha	1.23	
CPUManagerPolicyBetaOptions	true	Beta	1.23	
CPUManagerPolicyOptions	false	Alpha	1.22	1.22
CPUManagerPolicyOptions	true	Beta	1.23	
CRDValidationRatcheting	false	Alpha	1.28	
CSIMigrationPortworx	false	Alpha	1.23	1.24
CSIMigrationPortworx	false	Beta	1.25	
CSINodeExpandSecret	false	Alpha	1.25	1.26
CSINodeExpandSecret	true	Beta	1.27	
CSIVolumeHealth	false	Alpha	1.21	

Feature	Default	Stage	Since	Until
CloudControllerManagerWebhook	false	Alpha	1.27	
CloudDualStackNodeIPs	false	Alpha	1.27	
ClusterTrustBundle	false	Alpha	1.27	
ComponentSLIs	false	Alpha	1.26	1.26
ComponentSLIs	true	Beta	1.27	
ConsistentListFromCache	false	Alpha	1.28	
ContainerCheckpoint	false	Alpha	1.25	
ContextualLogging	false	Alpha	1.24	
CronJobsScheduledAnnotation	true	Beta	1.28	
CrossNamespaceVolumeDataSource	false	Alpha	1.26	
CustomCPUUCFSQuotaPeriod	false	Alpha	1.12	
CustomResourceValidationExpressions	false	Alpha	1.23	1.24
CustomResourceValidationExpressions	true	Beta	1.25	
DevicePluginCDIDevices	false	Alpha	1.28	
DisableCloudProviders	false	Alpha	1.22	
DisableKubeletCloudCredentialProviders	false	Alpha	1.23	
DynamicResourceAllocation	false	Alpha	1.26	
ElasticIndexedJob	true	Beta	1.27	
EventedPLEG	false	Alpha	1.26	1.26
EventedPLEG	false	Beta	1.27	
GracefulNodeShutdown	false	Alpha	1.20	1.20
GracefulNodeShutdown	true	Beta	1.21	
GracefulNodeShutdownBasedOnPodPriority	false	Alpha	1.23	1.23
GracefulNodeShutdownBasedOnPodPriority	true	Beta	1.24	
HPAContainerMetrics	false	Alpha	1.20	1.26
HPAContainerMetrics	true	Beta	1.27	
HPAScaleToZero	false	Alpha	1.16	
HonorPVReclaimPolicy	false	Alpha	1.23	
InPlacePodVerticalScaling	false	Alpha	1.27	
InTreePluginAWSUnregister	false	Alpha	1.21	
InTreePluginAzureDiskUnregister	false	Alpha	1.21	
InTreePluginAzureFileUnregister	false	Alpha	1.21	
InTreePluginGCEUnregister	false	Alpha	1.21	
InTreePluginOpenStackUnregister	false	Alpha	1.21	
InTreePluginPortworxUnregister	false	Alpha	1.23	
InTreePluginvSphereUnregister	false	Alpha	1.21	
JobBackoffLimitPerIndex	false	Alpha	1.28	
JobPodFailurePolicy	false	Alpha	1.25	1.25
JobPodFailurePolicy	true	Beta	1.26	
JobPodReplacementPolicy	false	Alpha	1.28	
JobReadyPods	false	Alpha	1.23	1.23
JobReadyPods	true	Beta	1.24	
KMSv2	false	Alpha	1.25	1.26
KMSv2	true	Beta	1.27	

Feature	Default	Stage	Since	Until
KMSv2KDF	false	Beta	1.28	
KubeProxyDrainingTerminatingNodes	false	Alpha	1.28	
KubeletCgroupDriverFromCRI	false	Alpha	1.28	
KubeletInUserNamespace	false	Alpha	1.22	
KubeletPodResourcesDynamicResources	false	Alpha	1.27	
KubeletPodResourcesGet	false	Alpha	1.27	
KubeletTracing	false	Alpha	1.25	1.26
KubeletTracing	true	Beta	1.27	
LegacyServiceAccountTokenCleanUp	false	Alpha	1.28	
LocalStorageCapacityIsolationFSQuotaMonitoring	false	Alpha	1.15	-
LogarithmicScaleDown	false	Alpha	1.21	1.21
LogarithmicScaleDown	true	Beta	1.22	
LoggingAlphaOptions	false	Alpha	1.24	-
LoggingBetaOptions	true	Beta	1.24	-
MatchLabelKeysInPodTopologySpread	false	Alpha	1.25	1.26
MatchLabelKeysInPodTopologySpread	true	Beta	1.27	-
MaxUnavailableStatefulSet	false	Alpha	1.24	
MemoryManager	false	Alpha	1.21	1.21
MemoryManager	true	Beta	1.22	
MemoryQoS	false	Alpha	1.22	
MinDomainsInPodTopologySpread	false	Alpha	1.24	1.24
MinDomainsInPodTopologySpread	false	Beta	1.25	1.26
MinDomainsInPodTopologySpread	true	Beta	1.27	
MultiCIDRRangeAllocator	false	Alpha	1.25	
MultiCIDRServiceAllocator	false	Alpha	1.27	
NewVolumeManagerReconstruction	false	Beta	1.27	1.27
NewVolumeManagerReconstruction	true	Beta	1.28	
NodeInclusionPolicyInPodTopologySpread	false	Alpha	1.25	1.25
NodeInclusionPolicyInPodTopologySpread	true	Beta	1.26	
NodeLogQuery	false	Alpha	1.27	
NodeSwap	false	Alpha	1.22	1.27
NodeSwap	false	Beta	1.28	
OpenAPIEnums	false	Alpha	1.23	1.23
OpenAPIEnums	true	Beta	1.24	
PDBUnhealthyPodEvictionPolicy	false	Alpha	1.26	1.26
PDBUnhealthyPodEvictionPolicy	true	Beta	1.27	
PersistentVolumeLastPhaseTransistionTime	false	Alpha	1.28	
PodAndContainerStatsFromCRI	false	Alpha	1.23	
PodDeletionCost	false	Alpha	1.21	1.21
PodDeletionCost	true	Beta	1.22	
PodDisruptionConditions	false	Alpha	1.25	1.25
PodDisruptionConditions	true	Beta	1.26	
PodHostIPs	false	Alpha	1.28	
PodIndexLabel	true	Beta	1.28	

Feature	Default	Stage	Since	Until
PodReadyToStartContainersCondition	false	Alpha	1.28	
PodSchedulingReadiness	false	Alpha	1.26	1.26
PodSchedulingReadiness	true	Beta	1.27	
ProcMountType	false	Alpha	1.12	
QOSReserved	false	Alpha	1.11	
ReadWriteOncePod	false	Alpha	1.22	1.26
ReadWriteOncePod	true	Beta	1.27	
RecoverVolumeExpansionFailure	false	Alpha	1.23	
RemainingItemCount	false	Alpha	1.15	1.15
RemainingItemCount	true	Beta	1.16	
RotateKubeletServerCertificate	false	Alpha	1.7	1.11
RotateKubeletServerCertificate	true	Beta	1.12	
SELinuxMountReadWriteOncePod	false	Alpha	1.25	1.26
SELinuxMountReadWriteOncePod	false	Beta	1.27	1.27
SELinuxMountReadWriteOncePod	true	Beta	1.28	
SchedulerQueueingHints	true	Beta	1.28	
SecurityContextDeny	false	Alpha	1.27	
ServiceNodePortStaticSubrange	false	Alpha	1.27	1.27
ServiceNodePortStaticSubrange	true	Beta	1.28	
SidecarContainers	false	Alpha	1.28	
SizeMemoryBackedVolumes	false	Alpha	1.20	1.21
SizeMemoryBackedVolumes	true	Beta	1.22	
SkipReadOnlyValidationGCE	false	Alpha	1.28	
StableLoadBalancerNodeSet	true	Beta	1.27	
StatefulSetAutoDeletePVC	false	Alpha	1.23	1.26
StatefulSetAutoDeletePVC	false	Beta	1.27	
StatefulSetStartOrdinal	false	Alpha	1.26	1.26
StatefulSetStartOrdinal	true	Beta	1.27	
StorageVersionAPI	false	Alpha	1.20	
StorageVersionHash	false	Alpha	1.14	1.14
StorageVersionHash	true	Beta	1.15	
TopologyAwareHints	false	Alpha	1.21	1.22
TopologyAwareHints	false	Beta	1.23	1.23
TopologyAwareHints	true	Beta	1.24	
TopologyManagerPolicyAlphaOptions	false	Alpha	1.26	
TopologyManagerPolicyBetaOptions	false	Beta	1.26	1.27
TopologyManagerPolicyBetaOptions	true	Beta	1.28	
TopologyManagerPolicyOptions	false	Alpha	1.26	1.27
TopologyManagerPolicyOptions	true	Beta	1.28	
UnknownVersionInteroperabilityProxy	false	Alpha	1.28	
UserNamespacesSupport	false	Alpha	1.28	
ValidatingAdmissionPolicy	false	Alpha	1.26	1.27
ValidatingAdmissionPolicy	false	Beta	1.28	
VolumeCapacityPriority	false	Alpha	1.21	

Feature	Default	Stage	Since	Until
WatchList	false	Alpha	1.27	
WinDSR	false	Alpha	1.14	
WinOverlay	false	Alpha	1.14	1.19
WinOverlay	true	Beta	1.20	
WindowsHostNetwork	true	Alpha	1.26	

Feature gates for graduated or deprecated features

Feature Gates for Graduated or Deprecated Features

Feature	Default	Stage	Since	Until
APISelfSubjectReview	false	Alpha	1.26	1.26
APISelfSubjectReview	true	Beta	1.27	1.27
APISelfSubjectReview	true	GA	1.28	-
CPUManager	false	Alpha	1.8	1.9
CPUManager	true	Beta	1.10	1.25
CPUManager	true	GA	1.26	-
CSIMigrationAzureFile	false	Alpha	1.15	1.20
CSIMigrationAzureFile	false	Beta	1.21	1.23
CSIMigrationAzureFile	true	Beta	1.24	1.25
CSIMigrationAzureFile	true	GA	1.26	
CSIMigrationRBD	false	Alpha	1.23	1.27
CSIMigrationRBD	false	Deprecated	1.28	
CSIMigrationvSphere	false	Alpha	1.18	1.18
CSIMigrationvSphere	false	Beta	1.19	1.24
CSIMigrationvSphere	true	Beta	1.25	1.25
CSIMigrationvSphere	true	GA	1.26	-
ConsistentHTTPGetHandlers	true	GA	1.25	-
CronJobTimeZone	false	Alpha	1.24	1.24
CronJobTimeZone	true	Beta	1.25	1.26
CronJobTimeZone	true	GA	1.27	-
DaemonSetUpdateSurge	false	Alpha	1.21	1.21
DaemonSetUpdateSurge	true	Beta	1.22	1.24
DaemonSetUpdateSurge	true	GA	1.25	
DefaultHostNetworkHostPortsInPodTemplates	false	Deprecated	1.28	
DownwardAPIHugePages	false	Alpha	1.20	1.20
DownwardAPIHugePages	false	Beta	1.21	1.21
DownwardAPIHugePages	true	Beta	1.22	1.26
DownwardAPIHugePages	true	GA	1.27	
EfficientWatchResumption	false	Alpha	1.20	1.20
EfficientWatchResumption	true	Beta	1.21	1.23
EfficientWatchResumption	true	GA	1.24	
ExecProbeTimeout	true	GA	1.20	
ExpandedDNSConfig	false	Alpha	1.22	1.25
ExpandedDNSConfig	true	Beta	1.26	1.27
ExpandedDNSConfig	true	GA	1.28	

Feature	Default	Stage	Since	Until
ExperimentalHostUserNamespaceDefaulting	false	Beta	1.5	1.27
ExperimentalHostUserNamespaceDefaulting	false	Deprecated	1.28	
GRPCContainerProbe	false	Alpha	1.23	1.23
GRPCContainerProbe	true	Beta	1.24	1.26
GRPCContainerProbe	true	GA	1.27	
IPTablesOwnershipCleanup	false	Alpha	1.25	1.26
IPTablesOwnershipCleanup	true	Beta	1.27	1.27
IPTablesOwnershipCleanup	true	GA	1.28	
InTreePluginRBDUnregister	false	Alpha	1.23	1.27
InTreePluginRBDUnregister	false	Deprecated	1.28	
JobMutableNodeSchedulingDirectives	true	Beta	1.23	1.26
JobMutableNodeSchedulingDirectives	true	GA	1.27	
JobTrackingWithFinalizers	false	Alpha	1.22	1.22
JobTrackingWithFinalizers	false	Beta	1.23	1.24
JobTrackingWithFinalizers	true	Beta	1.25	1.25
JobTrackingWithFinalizers	true	GA	1.26	
KMSv1	true	Deprecated	1.28	
KubeletPodResources	false	Alpha	1.13	1.14
KubeletPodResources	true	Beta	1.15	1.27
KubeletPodResources	true	GA	1.28	
KubeletPodResourcesGetAllocatable	false	Alpha	1.21	1.22
KubeletPodResourcesGetAllocatable	true	Beta	1.23	1.27
KubeletPodResourcesGetAllocatable	true	GA	1.28	
LegacyServiceAccountTokenNoAutoGeneration	true	Beta	1.24	1.25
LegacyServiceAccountTokenNoAutoGeneration	true	GA	1.26	
LegacyServiceAccountTokenTracking	false	Alpha	1.26	1.26
LegacyServiceAccountTokenTracking	true	Beta	1.27	1.27
LegacyServiceAccountTokenTracking	true	GA	1.28	
MinimizeIPTablesRestore	false	Alpha	1.26	1.26
MinimizeIPTablesRestore	true	Beta	1.27	1.27
MinimizeIPTablesRestore	true	GA	1.28	
NodeOutOfServiceVolumeDetach	false	Alpha	1.24	1.25
NodeOutOfServiceVolumeDetach	true	Beta	1.26	1.27
NodeOutOfServiceVolumeDetach	true	GA	1.28	
OpenAPIV3	false	Alpha	1.23	1.23
OpenAPIV3	true	Beta	1.24	1.26
OpenAPIV3	true	GA	1.27	
ProbeTerminationGracePeriod	false	Alpha	1.21	1.21
ProbeTerminationGracePeriod	false	Beta	1.22	1.24
ProbeTerminationGracePeriod	true	Beta	1.25	1.27
ProbeTerminationGracePeriod	true	GA	1.28	
ProxyTerminatingEndpoints	false	Alpha	1.22	1.25
ProxyTerminatingEndpoints	true	Beta	1.26	1.27
ProxyTerminatingEndpoints	true	GA	1.28	

Feature	Default	Stage	Since	Until
RemoveSelfLink	false	Alpha	1.16	1.19
RemoveSelfLink	true	Beta	1.20	1.23
RemoveSelfLink	true	GA	1.24	
RetroactiveDefaultStorageClass	false	Alpha	1.25	1.25
RetroactiveDefaultStorageClass	true	Beta	1.26	1.27
RetroactiveDefaultStorageClass	true	GA	1.28	
SeccompDefault	false	Alpha	1.22	1.24
SeccompDefault	true	Beta	1.25	1.26
SeccompDefault	true	GA	1.27	-
ServerSideApply	false	Alpha	1.14	1.15
ServerSideApply	true	Beta	1.16	1.21
ServerSideApply	true	GA	1.22	-
ServerSideFieldValidation	false	Alpha	1.23	1.24
ServerSideFieldValidation	true	Beta	1.25	1.26
ServerSideFieldValidation	true	GA	1.27	-
TopologyManager	false	Alpha	1.16	1.17
TopologyManager	true	Beta	1.18	1.26
TopologyManager	true	GA	1.27	-
WatchBookmark	false	Alpha	1.15	1.15
WatchBookmark	true	Beta	1.16	1.16
WatchBookmark	true	GA	1.17	-

Using a feature

Feature stages

A feature can be in *Alpha*, *Beta* or *GA* stage. An *Alpha* feature means:

- Disabled by default.
- Might be buggy. Enabling the feature may expose bugs.
- Support for feature may be dropped at any time without notice.
- The API may change in incompatible ways in a later software release without notice.
- Recommended for use only in short-lived testing clusters, due to increased risk of bugs and lack of long-term support.

A *Beta* feature means:

- Usually enabled by default. Beta API groups are [disabled by default](#).
- The feature is well tested. Enabling the feature is considered safe.
- Support for the overall feature will not be dropped, though details may change.
- The schema and/or semantics of objects may change in incompatible ways in a subsequent beta or stable release. When this happens, we will provide instructions for migrating to the next version. This may require deleting, editing, and re-creating API objects. The editing process may require some thought. This may require downtime for applications that rely on the feature.
- Recommended for only non-business-critical uses because of potential for incompatible changes in subsequent releases. If you have multiple clusters that can be upgraded independently, you may be able to relax this restriction.

Note: Please do try *Beta* features and give feedback on them! After they exit beta, it may not be practical for us to make more changes.

A *General Availability* (GA) feature is also referred to as a *stable* feature. It means:

- The feature is always enabled; you cannot disable it.
- The corresponding feature gate is no longer needed.
- Stable versions of features will appear in released software for many subsequent versions.

List of feature gates

Each feature gate is designed for enabling/disabling a specific feature:

- AdmissionWebhookMatchConditions: Enable [match conditions](#) on mutating & validating admission webhooks.
- APIListChunking: Enable the API clients to retrieve (LIST or GET) resources from API server in chunks.
- APIPriorityAndFairness: Enable managing request concurrency with prioritization and fairness at each server. (Renamed from RequestManagement)
- APIResponseCompression: Compress the API responses for LIST or GET requests.
- APISelfSubjectReview: Activate the SelfSubjectReview API which allows users to see the requesting subject's authentication information. See [API access to authentication information for a client](#) for more details.
- APIServerIdentity: Assign each API server an ID in a cluster, using a [Lease](#).
- APIServerTracing: Add support for distributed tracing in the API server. See [Traces for Kubernetes System Components](#) for more details.
- AggregatedDiscoveryEndpoint: Enable a single HTTP endpoint /discovery/<version> which supports native HTTP caching with ETags containing all APIResources known to the API server.
- AnyVolumeDataSource: Enable use of any custom resource as the DataSource of a [PVC](#).
- AppArmor: Enable use of AppArmor mandatory access control for Pods running on Linux nodes. See [AppArmor Tutorial](#) for more details.
- CPUManager: Enable container level CPU affinity support, see [CPU Management Policies](#).
- CPUManagerPolicyAlphaOptions: This allows fine-tuning of CPUManager policies, experimental, Alpha-quality options. This feature gate guards *a group* of CPUManager options whose quality level is alpha. This feature gate will never graduate to beta or stable.
- CPUManagerPolicyBetaOptions: This allows fine-tuning of CPUManager policies, experimental, Beta-quality options. This feature gate guards *a group* of CPUManager options whose quality level is beta. This feature gate will never graduate to stable.

`CPUManagerPolicyOptions`: Allow fine-tuning of CPUManager policies.

- `CSIMigrationAzureFile`: Enables shims and translation logic to route volume operations from the Azure-File in-tree plugin to AzureFile CSI plugin. Supports falling back to in-tree AzureFile plugin for mount operations to nodes that have the feature disabled or that do not have AzureFile CSI plugin installed and configured. Does not support falling back for provision operations, for those the CSI plugin must be installed and configured. Requires CSIMigration feature flag enabled.
- `CSIMigrationRBD`: Enables shims and translation logic to route volume operations from the RBD in-tree plugin to Ceph RBD CSI plugin. Requires CSIMigration and `csiMigrationRBD` feature flags enabled and Ceph CSI plugin installed and configured in the cluster. This flag has been deprecated in favor of the `InTreePluginRBDUnregister` feature flag which prevents the registration of in-tree RBD plugin.
- `CSIMigrationvSphere`: Enables shims and translation logic to route volume operations from the vSphere in-tree plugin to vSphere CSI plugin. Supports falling back to in-tree vSphere plugin for mount operations to nodes that have the feature disabled or that do not have vSphere CSI plugin installed and configured. Does not support falling back for provision operations, for those the CSI plugin must be installed and configured. Requires CSIMigration feature flag enabled.
- `CSIMigrationPortworx`: Enables shims and translation logic to route volume operations from the Portworx in-tree plugin to Portworx CSI plugin. Requires Portworx CSI driver to be installed and configured in the cluster.
- `CSINodeExpandSecret`: Enable passing secret authentication data to a CSI driver for use during a `NodeExpandVolume` CSI operation.
- `CSIVolumeHealth`: Enable support for CSI volume health monitoring on node.
- `CloudControllerManagerWebhook`: Enable webhooks in cloud controller manager.
- `CloudDualStackNodeIPs`: Enables dual-stack kubelet --node-ip with external cloud providers. See [Configure IPv4/IPv6 dual-stack](#) for more details.
- `ClusterTrustBundle`: Enable ClusterTrustBundle objects and kubelet integration.
- `ComponentSLIs`: Enable the /metrics/slis endpoint on Kubernetes components like kubelet, kube-scheduler, kube-proxy, kube-controller-manager, cloud-controller-manager allowing you to scrape health check metrics.
- `ConsistentHTTPGetHandlers`: Normalize HTTP get URL and Header passing for lifecycle handlers with probers.
- `ConsistentListFromCache`: Allow the API server to serve consistent lists from cache.
- `ContainerCheckpoint`: Enables the kubelet checkpoint API. See [Kubelet Checkpoint API](#) for more details.
- `ContextualLogging`: When you enable this feature gate, Kubernetes components that support contextual logging add extra detail to log output.
- `CronJobsScheduledAnnotation`: Set the scheduled job time as an [annotation](#) on Jobs that were created on behalf of a CronJob.

CronJobTimeZone: Allow the use of the timeZone optional field in [CronJobs](#).

- CRDValidationRatcheting: Enable updates to custom resources to contain violations of their OpenAPI schema if the offending portions of the resource update did not change. See [Validation Ratcheting](#) for more details.
- CrossNamespaceVolumeDataSource: Enable the usage of cross namespace volume data source to allow you to specify a source namespace in the dataSourceRef field of a PersistentVolumeClaim.
- CustomCPUCFSQuotaPeriod: Enable nodes to change cpuCFSQuotaPeriod in [kubelet config](#).
- CustomResourceValidationExpressions: Enable expression language validation in CRD which will validate customer resource based on validation rules written in the x-kubernetes-validations extension.
- DaemonSetUpdateSurge: Enables the DaemonSet workloads to maintain availability during update per node. See [Perform a Rolling Update on a DaemonSet](#).
- DefaultHostNetworkHostPortsInPodTemplates: Changes when the default value of PodSpec.containers[*].ports[*].hostPort is assigned. The default is to only set a default value in Pods. Enabling this means a default will be assigned even to embedded PodSpecs (e.g. in a Deployment), which is the historical default.
- DevicePluginCDIDevices: Enable support to CDI device IDs in the [Device Plugin](#) API.
- DisableCloudProviders: Disables any functionality in kube-apiserver, kube-controller-manager and kubelet related to the --cloud-provider component flag.
- DisableKubeletCloudCredentialProviders: Disable the in-tree functionality in kubelet to authenticate to a cloud provider container registry for image pull credentials.
- DownwardAPIHugePages: Enables usage of hugepages in [downward API](#).
- DynamicResourceAllocation: Enables support for resources with custom parameters and a lifecycle that is independent of a Pod.
- ElasticIndexedJob: Enables Indexed Jobs to be scaled up or down by mutating both spec.completions and spec.parallelism together such that spec.completions == spec.parallelism. See docs on [elastic Indexed Jobs](#) for more details.
- EfficientWatchResumption: Allows for storage-originated bookmark (progress notify) events to be delivered to the users. This is only applied to watch operations.
- EventedPLEG: Enable support for the kubelet to receive container life cycle events from the [container runtime](#) via an extension to [CRI](#). (PLEG is an abbreviation for “Pod lifecycle event generator”). For this feature to be useful, you also need to enable support for container lifecycle events in each container runtime running in your cluster. If the container runtime does not announce support for container lifecycle events then the kubelet automatically switches to the legacy generic PLEG mechanism, even if you have this feature gate enabled.

`ExecProbeTimeout`: Ensure kubelet respects exec probe timeouts. This feature gate exists • in case any of your existing workloads depend on a now-corrected fault where Kubernetes ignored exec probe timeouts. See [readiness probes](#).

- `ExpandedDNSConfig`: Enable kubelet and kube-apiserver to allow more DNS search paths and longer list of DNS search paths. This feature requires container runtime support (containerd: v1.5.6 or higher, CRI-O: v1.22 or higher). See [Expanded DNS Configuration](#).
- `ExperimentalHostUserNamespaceDefaulting`: Enabling the defaulting user namespace to host. This is for containers that are using other host namespaces, host mounts, or containers that are privileged or using specific non-namespaced capabilities (e.g. MKNODE, SYS_MODULE etc.). This should only be enabled if user namespace remapping is enabled in the Docker daemon.
- `GracefulNodeShutdown`: Enables support for graceful shutdown in kubelet. During a system shutdown, kubelet will attempt to detect the shutdown event and gracefully terminate pods running on the node. See [Graceful Node Shutdown](#) for more details.
- `GracefulNodeShutdownBasedOnPodPriority`: Enables the kubelet to check Pod priorities when shutting down a node gracefully.
- `GRPCContainerProbe`: Enables the gRPC probe method for liveness, readiness and startup probes. See [Configure Liveness, Readiness and Startup Probes](#).
- `HonorPVReclaimPolicy`: Honor persistent volume reclaim policy when it is Delete irrespective of PV-PVC deletion ordering. For more details, check the [PersistentVolume deletion protection finalizer](#) documentation.
- `HPAContainerMetrics`: Enable the HorizontalPodAutoscaler to scale based on metrics from individual containers in target pods.
- `HPAScaleToZero`: Enables setting minReplicas to 0 for HorizontalPodAutoscaler resources when using custom or external metrics.
- `IPTablesOwnershipCleanup`: This causes kubelet to no longer create legacy iptables rules.
- `InPlacePodVerticalScaling`: Enables in-place Pod vertical scaling.
- `InTreePluginAWSUnregister`: Stops registering the aws-ebs in-tree plugin in kubelet and volume controllers.
- `InTreePluginAzureDiskUnregister`: Stops registering the azuredisk in-tree plugin in kubelet and volume controllers.
- `InTreePluginAzureFileUnregister`: Stops registering the azurefile in-tree plugin in kubelet and volume controllers.
- `InTreePluginGCEUnregister`: Stops registering the gce-pd in-tree plugin in kubelet and volume controllers.
- `InTreePluginOpenStackUnregister`: Stops registering the OpenStack cinder in-tree plugin in kubelet and volume controllers.
- `InTreePluginPortworxUnregister`: Stops registering the Portworx in-tree plugin in kubelet and volume controllers.

- InTreePluginRBDUnregister: Stops registering the RBD in-tree plugin in kubelet and volume controllers.
- InTreePluginvSphereUnregister: Stops registering the vSphere in-tree plugin in kubelet and volume controllers.
- JobMutableNodeSchedulingDirectives: Allows updating node scheduling directives in the pod template of [Job](#).
- JobBackoffLimitPerIndex: Allows specifying the maximal number of pod retries per index in Indexed jobs.
- JobPodFailurePolicy: Allow users to specify handling of pod failures based on container exit codes and pod conditions.
- JobPodReplacementPolicy: Allows you to specify pod replacement for terminating pods in a [Job](#).
- JobReadyPods: Enables tracking the number of Pods that have a Ready [condition](#). The count of Ready pods is recorded in the [status](#) of a [Job](#) status.
- JobTrackingWithFinalizers: Enables tracking [Job](#) completions without relying on Pods remaining in the cluster indefinitely. The Job controller uses Pod finalizers and a field in the Job status to keep track of the finished Pods to count towards completion.
- KMSv1: Enables KMS v1 API for encryption at rest. See [Using a KMS Provider for data encryption](#) for more details.
- KMSv2: Enables KMS v2 API for encryption at rest. See [Using a KMS Provider for data encryption](#) for more details.
- KMSv2KDF: Enables KMS v2 to generate single use data encryption keys. See [Using a KMS Provider for data encryption](#) for more details. If the KMSv2 feature gate is not enabled in your cluster, the value of the KMSv2KDF feature gate has no effect.
- KubeProxyDrainingTerminatingNodes: Implement connection draining for terminating nodes for externalTrafficPolicy: Cluster services.
- KubeletCgroupDriverFromCRI: Enable detection of the kubelet cgroup driver configuration option from the [CRI](#). You can use this feature gate on nodes with a kubelet that supports the feature gate and where there is a CRI container runtime that supports the RuntimeConfig CRI call. If both CRI and kubelet support this feature, the kubelet ignores the cgroupDriver configuration setting (or deprecated --cgroup-driver command line argument). If you enable this feature gate and the container runtime doesn't support it, the kubelet falls back to using the driver configured using the cgroupDriver configuration setting. See [Configuring a cgroup driver](#) for more details.
- KubeletInUserNamespace: Enables support for running kubelet in a [user namespace](#). See [Running Kubernetes Node Components as a Non-root User](#).
- KubeletPodResources: Enable the kubelet's pod resources gRPC endpoint. See [Support Device Monitoring](#) for more details.
- KubeletPodResourcesGet: Enable the Get gRPC endpoint on kubelet's for Pod resources. This API augments the [resource allocation reporting](#).

KubeletPodResourcesGetAllocatable: Enable the kubelet's pod resources

- GetAllocatableResources functionality. This API augments the [resource allocation reporting](#).

- KubeletPodResourcesDynamicResources: Extend the kubelet's pod resources gRPC endpoint to include resources allocated in ResourceClaims via DynamicResourceAllocation API. See [resource allocation reporting](#) for more details. with informations about the allocatable resources, enabling clients to properly track the free compute resources on a node.

- KubeletTracing: Add support for distributed tracing in the kubelet. When enabled, kubelet CRI interface and authenticated http servers are instrumented to generate OpenTelemetry trace spans. See [Traces for Kubernetes System Components](#) for more details.

- LegacyServiceAccountTokenNoAutoGeneration: Stop auto-generation of Secret-based [service account tokens](#).

- LegacyServiceAccountTokenCleanUp: Enable cleaning up Secret-based [service account tokens](#) when they are not used in a specified time (default to be one year).

- LegacyServiceAccountTokenTracking: Track usage of Secret-based [service account tokens](#).

- LocalStorageCapacityIsolationFSQuotaMonitoring: When LocalStorageCapacityIsolation is enabled for [local ephemeral storage](#) and the backing filesystem for [emptyDir volumes](#) supports project quotas and they are enabled, use project quotas to monitor [emptyDir volume](#) storage consumption rather than filesystem walk for better performance and accuracy.

- LogarithmicScaleDown: Enable semi-random selection of pods to evict on controller scaledown based on logarithmic bucketing of pod timestamps.

- LoggingAlphaOptions: Allow fine-tuning of experimental, alpha-quality logging options.

- LoggingBetaOptions: Allow fine-tuning of experimental, beta-quality logging options.

- MatchLabelKeysInPodTopologySpread: Enable the matchLabelKeys field for [Pod topology spread constraints](#).

- MaxUnavailableStatefulSet: Enables setting the maxUnavailable field for the [rolling update strategy](#) of a StatefulSet. The field specifies the maximum number of Pods that can be unavailable during the update.

- MemoryManager: Allows setting memory affinity for a container based on NUMA topology.

- MemoryQoS: Enable memory protection and usage throttle on pod / container using cgroup v2 memory controller.

- MinDomainsInPodTopologySpread: Enable minDomains in [Pod topology spread constraints](#).

- MinimizeIPTablesRestore: Enables new performance improvement logics in the kube-proxy iptables mode.

- MultiCIDRRRangeAllocator: Enables the MultiCIDR range allocator.
- MultiCIDRServiceAllocator: Track IP address allocations for Service cluster IPs using IPAddress objects.
- NewVolumeManagerReconstruction: Enables improved discovery of mounted volumes during kubelet startup. Since this code has been significantly refactored, we allow to opt-out in case kubelet gets stuck at the startup or is not unmounting volumes from terminated Pods. Note that this refactoring was behind SELinuxMountReadWriteOncePod alpha feature gate in Kubernetes 1.25.

Before Kubernetes v1.25, the kubelet used different default behavior for discovering mounted volumes during the kubelet startup. If you disable this feature gate (it's enabled by default), you select the legacy discovery behavior.

In Kubernetes v1.25 and v1.26, this behavior toggle was part of the SELinuxMountReadWriteOncePod feature gate.

- NodeInclusionPolicyInPodTopologySpread: Enable using nodeAffinityPolicy and nodeTaintsPolicy in [Pod topology spread constraints](#) when calculating pod topology spread skew.
- NodeLogQuery: Enables querying logs of node services using the /logs endpoint.
- NodeOutOfServiceVolumeDetach: When a Node is marked out-of-service using the node.kubernetes.io/out-of-service taint, Pods on the node will be forcefully deleted if they can not tolerate this taint, and the volume detach operations for Pods terminating on the node will happen immediately. The deleted Pods can recover quickly on different nodes.
- NodeSwap: Enable the kubelet to allocate swap memory for Kubernetes workloads on a node. Must be used with KubeletConfiguration.failSwapOn set to false. For more details, please see [swap memory](#).
- OpenAPIEnums: Enables populating "enum" fields of OpenAPI schemas in the spec returned from the API server.
- OpenAPIV3: Enables the API server to publish OpenAPI v3.
- PDBUnhealthyPodEvictionPolicy: Enables the unhealthyPodEvictionPolicy field of a PodDisruptionBudget. This specifies when unhealthy pods should be considered for eviction. Please see [Unhealthy Pod Eviction Policy](#) for more details.
- PersistentVolumeLastPhaseTransitionTime: Adds a new field to PersistentVolume which holds a timestamp of when the volume last transitioned its phase.
- PodAndContainerStatsFromCRI: Configure the kubelet to gather container and pod stats from the CRI container runtime rather than gathering them from cAdvisor. As of 1.26, this also includes gathering metrics from CRI and emitting them over /metrics/cadvisor (rather than having cAdvisor emit them directly).
- PodDeletionCost: Enable the [Pod Deletion Cost](#) feature which allows users to influence ReplicaSet downscaling order.
- PodDisruptionConditions: Enables support for appending a dedicated pod condition indicating that the pod is being deleted due to a disruption.

PodHostIPs: Enable the status.hostIPs field for pods and the [downward API](#). The field lets you expose host IP addresses to workloads.

- PodIndexLabel: Enables the Job controller and StatefulSet controller to add the pod index as a label when creating new pods. See [Job completion mode docs](#) and [StatefulSet pod index label docs](#) for more details.
- PodReadyToStartContainersCondition: Enable the kubelet to mark the [PodReadyToStartContainers](#) condition on pods. This was previously (1.25-1.27) known as PodHasNetworkCondition.
- PodSchedulingReadiness: Enable setting schedulingGates field to control a Pod's [scheduling readiness](#).
- ProbeTerminationGracePeriod: Enable [setting probe-level terminationGracePeriodSeconds](#) on pods. See the [enhancement proposal](#) for more details.
- ProcMountType: Enables control over the type proc mounts for containers by setting the procMount field of a SecurityContext.
- ProxyTerminatingEndpoints: Enable the kube-proxy to handle terminating endpoints when ExternalTrafficPolicy=Local.
- QOSReserved: Allows resource reservations at the QoS level preventing pods at lower QoS levels from bursting into resources requested at higher QoS levels (memory only for now).
- ReadWriteOncePod: Enables the usage of ReadWriteOncePod PersistentVolume access mode.
- RecoverVolumeExpansionFailure: Enables users to edit their PVCs to smaller sizes so as they can recover from previously issued volume expansion failures. See [Recovering from Failure when Expanding Volumes](#) for more details.
- RemainingItemCount: Allow the API servers to show a count of remaining items in the response to a [chunking list request](#).
- RemoveSelfLink: Sets the .metadata.selfLink field to blank (empty string) for all objects and collections. This field has been deprecated since the Kubernetes v1.16 release. When this feature is enabled, the .metadata.selfLink field remains part of the Kubernetes API, but is always unset.
- RetroactiveDefaultStorageClass: Allow assigning StorageClass to unbound PVCs retroactively.
- RotateKubeletServerCertificate: Enable the rotation of the server TLS certificate on the kubelet. See [kubelet configuration](#) for more details.
- SELinuxMountReadWriteOncePod: Speeds up container startup by allowing kubelet to mount volumes for a Pod directly with the correct SELinux label instead of changing each file on the volumes recursively. The initial implementation focused on ReadWriteOncePod volumes.
- SchedulerQueueingHints: Enables [the scheduler's queueing hints enhancement](#), which benefits to reduce the useless requeueing. The scheduler retries scheduling pods if

something changes in the cluster that could make the pod scheduled. Queueing hints are internal signals that allow the scheduler to filter the changes in the cluster that are relevant to the unscheduled pod, based on previous scheduling attempts.

- SeccompDefault: Enables the use of RuntimeDefault as the default seccomp profile for all workloads. The seccomp profile is specified in the securityContext of a Pod and/or a Container.
- SecurityContextDeny: This gate signals that the SecurityContextDeny admission controller is deprecated.
- ServerSideApply: Enables the [Server Side Apply \(SSA\)](#) feature on the API Server.
- ServerSideFieldValidation: Enables server-side field validation. This means the validation of resource schema is performed at the API server side rather than the client side (for example, the kubectl create or kubectl apply command line).
- ServiceNodePortStaticSubrange: Enables the use of different port allocation strategies for NodePort Services. For more details, see [reserve NodePort ranges to avoid collisions](#).
- SidecarContainers: Allow setting the restartPolicy of an init container to Always so that the container becomes a sidecar container (restartable init containers). See [Sidecar containers and restartPolicy](#) for more details.
- SizeMemoryBackedVolumes: Enable kubelets to determine the size limit for memory-backed volumes (mainly emptyDir volumes).
- SkipReadOnlyValidationGCE: Skip validation for GCE, will enable in the next version.
- StableLoadBalancerNodeSet: Enables less load balancer re-configurations by the service controller (KCCM) as an effect of changing node state.
- StatefulSetStartOrdinal: Allow configuration of the start ordinal in a StatefulSet. See [Start ordinal](#) for more details.
- StorageVersionAPI: Enable the [storage version API](#).
- StorageVersionHash: Allow API servers to expose the storage version hash in the discovery.
- TopologyAwareHints: Enables topology aware routing based on topology hints in EndpointSlices. See [Topology Aware Hints](#) for more details.
- TopologyManager: Enable a mechanism to coordinate fine-grained hardware resource assignments for different components in Kubernetes. See [Control Topology Management Policies on a node](#).
- TopologyManagerPolicyAlphaOptions: Allow fine-tuning of topology manager policies, experimental, Alpha-quality options. This feature gate guards *a group* of topology manager options whose quality level is alpha. This feature gate will never graduate to beta or stable.
- TopologyManagerPolicyBetaOptions: Allow fine-tuning of topology manager policies, experimental, Beta-quality options. This feature gate guards *a group* of topology manager options whose quality level is beta. This feature gate will never graduate to stable.

- TopologyManagerPolicyOptions: Allow fine-tuning of topology manager policies.
- UnknownVersionInteroperabilityProxy: Proxy resource requests to the correct peer kube-apiserver when multiple kube-apiservers exist at varied versions. See [Mixed version proxy](#) for more information.
- UserNamespacesSupport: Enable user namespace support for Pods. Before Kubernetes v1.28, this feature gate was named UserNamespacesStatelessPodsSupport.
- ValidatingAdmissionPolicy: Enable [ValidatingAdmissionPolicy](#) support for CEL validations be used in Admission Control.
- VolumeCapacityPriority: Enable support for prioritizing nodes in different topologies based on available PV capacity.
- WatchBookmark: Enable support for watch bookmark events.
- WatchList : Enable support for [streaming initial state of objects in watch requests](#).
- WinDSR: Allows kube-proxy to create DSR loadbalancers for Windows.
- WinOverlay: Allows kube-proxy to run in overlay mode for Windows.
- WindowsHostNetwork: Enables support for joining Windows containers to a hosts' network namespace.

What's next

- The [deprecation policy](#) for Kubernetes explains the project's approach to removing features and components.
- Since Kubernetes 1.24, new beta APIs are not enabled by default. When enabling a beta feature, you will also need to enable any associated API resources. For example, to enable a particular resource like storage.k8s.io/v1beta1/csistoragecapacities, set --runtime-config=storage.k8s.io/v1beta1/csistoragecapacities. See [API Versioning](#) for more details on the command line flags.

Feature Gates (removed)

This page contains list of feature gates that have been removed. The information on this page is for reference. A removed feature gate is different from a GA'ed or deprecated one in that a removed one is no longer recognized as a valid feature gate. However, a GA'ed or a deprecated feature gate is still recognized by the corresponding Kubernetes components although they are unable to cause any behavior differences in a cluster.

For feature gates that are still recognized by the Kubernetes components, please refer to the [Alpha/Beta feature gate table](#) or the [Graduated/Deprecated feature gate table](#)

Feature gates that are removed

In the following table:

- The "From" column contains the Kubernetes release when a feature is introduced or its release stage is changed.
- The "To" column, if not empty, contains the last Kubernetes release in which you can still use a feature gate. If the feature stage is either "Deprecated" or "GA", the "To" column is the Kubernetes release when the feature is removed.

Feature Gates Removed

Feature	Default	Stage	From	To
Accelerators	false	Alpha	1.6	1.10
Accelerators	-	Deprecated	1.11	1.11
AdvancedAuditing	false	Alpha	1.7	1.7
AdvancedAuditing	true	Beta	1.8	1.11
AdvancedAuditing	true	GA	1.12	1.27
AffinityInAnnotations	false	Alpha	1.6	1.7
AffinityInAnnotations	-	Deprecated	1.8	1.8
AllowExtTrafficLocalEndpoints	false	Beta	1.4	1.6
AllowExtTrafficLocalEndpoints	true	GA	1.7	1.9
AllowInsecureBackendProxy	true	Beta	1.17	1.20
AllowInsecureBackendProxy	true	GA	1.21	1.25
AttachVolumeLimit	false	Alpha	1.11	1.11
AttachVolumeLimit	true	Beta	1.12	1.16
AttachVolumeLimit	true	GA	1.17	1.21
BalanceAttachedNodeVolumes	false	Alpha	1.11	1.21
BalanceAttachedNodeVolumes	false	Deprecated	1.22	1.22
BlockVolume	false	Alpha	1.9	1.12
BlockVolume	true	Beta	1.13	1.17
BlockVolume	true	GA	1.18	1.21
BoundServiceAccountTokenVolume	false	Alpha	1.13	1.20
BoundServiceAccountTokenVolume	true	Beta	1.21	1.21
BoundServiceAccountTokenVolume	true	GA	1.22	1.23
CRIContainerLogRotation	false	Alpha	1.10	1.10
CRIContainerLogRotation	true	Beta	1.11	1.20
CRIContainerLogRotation	true	GA	1.21	1.22
CSIBlockVolume	false	Alpha	1.11	1.13
CSIBlockVolume	true	Beta	1.14	1.17
CSIBlockVolume	true	GA	1.18	1.21
CSIDriverRegistry	false	Alpha	1.12	1.13
CSIDriverRegistry	true	Beta	1.14	1.17
CSIDriverRegistry	true	GA	1.18	1.21
CSIInlineVolume	false	Alpha	1.15	1.15
CSIInlineVolume	true	Beta	1.16	1.24
CSIInlineVolume	true	GA	1.25	1.26
CSIMigration	false	Alpha	1.14	1.16

Feature	Default	Stage	From	To
CSIMigration	true	Beta	1.17	1.24
CSIMigration	true	GA	1.25	1.26
CSIMigrationAWS	false	Alpha	1.14	1.16
CSIMigrationAWS	false	Beta	1.17	1.22
CSIMigrationAWS	true	Beta	1.23	1.24
CSIMigrationAWS	true	GA	1.25	1.26
CSIMigrationAWSComplete	false	Alpha	1.17	1.20
CSIMigrationAWSComplete	-	Deprecated	1.21	1.21
CSIMigrationAzureDisk	false	Alpha	1.15	1.18
CSIMigrationAzureDisk	false	Beta	1.19	1.22
CSIMigrationAzureDisk	true	Beta	1.23	1.23
CSIMigrationAzureDisk	true	GA	1.24	1.26
CSIMigrationAzureDiskComplete	false	Alpha	1.17	1.20
CSIMigrationAzureDiskComplete	-	Deprecated	1.21	1.21
CSIMigrationAzureFileComplete	false	Alpha	1.17	1.20
CSIMigrationAzureFileComplete	-	Deprecated	1.21	1.21
CSIMigrationGCE	false	Alpha	1.14	1.16
CSIMigrationGCE	false	Beta	1.17	1.22
CSIMigrationGCE	true	Beta	1.23	1.24
CSIMigrationGCE	true	GA	1.25	1.27
CSIMigrationGCEComplete	false	Alpha	1.17	1.20
CSIMigrationGCEComplete	-	Deprecated	1.21	1.21
CSIMigrationOpenStack	false	Alpha	1.14	1.17
CSIMigrationOpenStack	true	Beta	1.18	1.23
CSIMigrationOpenStack	true	GA	1.24	1.25
CSIMigrationOpenStackComplete	false	Alpha	1.17	1.20
CSIMigrationOpenStackComplete	-	Deprecated	1.21	1.21
CSIMigrationvSphereComplete	false	Beta	1.19	1.21
CSIMigrationvSphereComplete	-	Deprecated	1.22	1.22
CSINodeInfo	false	Alpha	1.12	1.13
CSINodeInfo	true	Beta	1.14	1.16
CSINodeInfo	true	GA	1.17	1.22
CSIPersistentVolume	false	Alpha	1.9	1.9
CSIPersistentVolume	true	Beta	1.10	1.12
CSIPersistentVolume	true	GA	1.13	1.16
CSIServiceAccountToken	false	Alpha	1.20	1.20
CSIServiceAccountToken	true	Beta	1.21	1.21
CSIServiceAccountToken	true	GA	1.22	1.24
CSIStorageCapacity	false	Alpha	1.19	1.20
CSIStorageCapacity	true	Beta	1.21	1.23
CSIStorageCapacity	true	GA	1.24	1.27
CSIVolumeFSGroupPolicy	false	Alpha	1.19	1.19
CSIVolumeFSGroupPolicy	true	Beta	1.20	1.22
CSIVolumeFSGroupPolicy	true	GA	1.23	1.25

Feature	Default	Stage	From	To
CSRDURATION	true	Beta	1.22	1.23
CSRDURATION	true	GA	1.24	1.25
ConfigurableFSGroupPolicy	false	Alpha	1.18	1.19
ConfigurableFSGroupPolicy	true	Beta	1.20	1.22
ConfigurableFSGroupPolicy	true	GA	1.23	1.25
ControllerManagerLeaderMigration	false	Alpha	1.21	1.21
ControllerManagerLeaderMigration	true	Beta	1.22	1.23
ControllerManagerLeaderMigration	true	GA	1.24	1.26
CronJobControllerV2	false	Alpha	1.20	1.20
CronJobControllerV2	true	Beta	1.21	1.21
CronJobControllerV2	true	GA	1.22	1.23
CustomPodDNS	false	Alpha	1.9	1.9
CustomPodDNS	true	Beta	1.10	1.13
CustomPodDNS	true	GA	1.14	1.16
CustomResourceDefaulting	false	Alpha	1.15	1.15
CustomResourceDefaulting	true	Beta	1.16	1.16
CustomResourceDefaulting	true	GA	1.17	1.18
CustomResourcePublishOpenAPI	false	Alpha	1.14	1.14
CustomResourcePublishOpenAPI	true	Beta	1.15	1.15
CustomResourcePublishOpenAPI	true	GA	1.16	1.18
CustomResourceSubresources	false	Alpha	1.10	1.10
CustomResourceSubresources	true	Beta	1.11	1.15
CustomResourceSubresources	true	GA	1.16	1.18
CustomResourceValidation	false	Alpha	1.8	1.8
CustomResourceValidation	true	Beta	1.9	1.15
CustomResourceValidation	true	GA	1.16	1.18
CustomResourceWebhookConversion	false	Alpha	1.13	1.14
CustomResourceWebhookConversion	true	Beta	1.15	1.15
CustomResourceWebhookConversion	true	GA	1.16	1.18
DaemonSetUpdateSurge	false	Alpha	1.21	1.21
DaemonSetUpdateSurge	true	Beta	1.22	1.24
DaemonSetUpdateSurge	true	GA	1.25	1.26
DefaultPodTopologySpread	false	Alpha	1.19	1.19
DefaultPodTopologySpread	true	Beta	1.20	1.23
DefaultPodTopologySpread	true	GA	1.24	1.25
DelegateFSGroupToCSIDriver	false	Alpha	1.22	1.22
DelegateFSGroupToCSIDriver	true	Beta	1.23	1.25
DelegateFSGroupToCSIDriver	true	GA	1.26	1.27
DevicePlugins	false	Alpha	1.8	1.9
DevicePlugins	true	Beta	1.10	1.25
DevicePlugins	true	GA	1.26	1.27
DisableAcceleratorUsageMetrics	false	Alpha	1.19	1.19
DisableAcceleratorUsageMetrics	true	Beta	1.20	1.24
DisableAcceleratorUsageMetrics	true	GA	1.25	1.27

Feature	Default	Stage	From	To
DryRun	false	Alpha	1.12	1.12
DryRun	true	Beta	1.13	1.18
DryRun	true	GA	1.19	1.27
DynamicAuditing	false	Alpha	1.13	1.18
DynamicAuditing	-	Deprecated	1.19	1.19
DynamicKubeletConfig	false	Alpha	1.4	1.10
DynamicKubeletConfig	true	Beta	1.11	1.21
DynamicKubeletConfig	false	Deprecated	1.22	1.25
DynamicProvisioningScheduling	false	Alpha	1.11	1.11
DynamicProvisioningScheduling	-	Deprecated	1.12	-
DynamicVolumeProvisioning	true	Alpha	1.3	1.7
DynamicVolumeProvisioning	true	GA	1.8	1.12
EnableAggregatedDiscoveryTimeout	true	Deprecated	1.16	1.17
EnableEquivalenceClassCache	false	Alpha	1.8	1.12
EnableEquivalenceClassCache	-	Deprecated	1.13	1.23
EndpointSlice	false	Alpha	1.16	1.16
EndpointSlice	false	Beta	1.17	1.17
EndpointSlice	true	Beta	1.18	1.20
EndpointSlice	true	GA	1.21	1.24
EndpointSliceNodeName	false	Alpha	1.20	1.20
EndpointSliceNodeName	true	GA	1.21	1.24
EndpointSliceProxying	false	Alpha	1.18	1.18
EndpointSliceProxying	true	Beta	1.19	1.21
EndpointSliceProxying	true	GA	1.22	1.24
EndpointSliceTerminatingCondition	false	Alpha	1.20	1.21
EndpointSliceTerminatingCondition	true	Beta	1.22	1.25
EndpointSliceTerminatingCondition	true	GA	1.26	1.27
EphemeralContainers	false	Alpha	1.16	1.22
EphemeralContainers	true	Beta	1.23	1.24
EphemeralContainers	true	GA	1.25	1.26
EvenPodsSpread	false	Alpha	1.16	1.17
EvenPodsSpread	true	Beta	1.18	1.18
EvenPodsSpread	true	GA	1.19	1.21
ExpandCSIVolumes	false	Alpha	1.14	1.15
ExpandCSIVolumes	true	Beta	1.16	1.23
ExpandCSIVolumes	true	GA	1.24	1.26
ExpandInUsePersistentVolumes	false	Alpha	1.11	1.14
ExpandInUsePersistentVolumes	true	Beta	1.15	1.23
ExpandInUsePersistentVolumes	true	GA	1.24	1.26
ExpandPersistentVolumes	false	Alpha	1.8	1.10
ExpandPersistentVolumes	true	Beta	1.11	1.23
ExpandPersistentVolumes	true	GA	1.24	1.26
ExperimentalCriticalPodAnnotation	false	Alpha	1.5	1.12
ExperimentalCriticalPodAnnotation	false	Deprecated	1.13	1.16

Feature	Default	Stage	From	To
ExternalPolicyForExternalIP	true	GA	1.18	1.22
GCERegionalPersistentDisk	true	Beta	1.10	1.12
GCERegionalPersistentDisk	true	GA	1.13	1.16
GenericEphemeralVolume	false	Alpha	1.19	1.20
GenericEphemeralVolume	true	Beta	1.21	1.22
GenericEphemeralVolume	true	GA	1.23	1.24
HugePageStorageMediumSize	false	Alpha	1.18	1.18
HugePageStorageMediumSize	true	Beta	1.19	1.21
HugePageStorageMediumSize	true	GA	1.22	1.24
HugePages	false	Alpha	1.8	1.9
HugePages	true	Beta	1.10	1.13
HugePages	true	GA	1.14	1.16
HyperVContainer	false	Alpha	1.10	1.19
HyperVContainer	false	Deprecated	1.20	1.20
IPv6DualStack	false	Alpha	1.15	1.20
IPv6DualStack	true	Beta	1.21	1.22
IPv6DualStack	true	GA	1.23	1.24
IdentifyPodOS	false	Alpha	1.23	1.23
IdentifyPodOS	true	Beta	1.24	1.24
IdentifyPodOS	true	GA	1.25	1.26
ImmutableEphemeralVolumes	false	Alpha	1.18	1.18
ImmutableEphemeralVolumes	true	Beta	1.19	1.20
ImmutableEphemeralVolumes	true	GA	1.21	1.24
IndexedJob	false	Alpha	1.21	1.21
IndexedJob	true	Beta	1.22	1.23
IndexedJob	true	GA	1.24	1.25
IngressClassNameSpacedParams	false	Alpha	1.21	1.21
IngressClassNameSpacedParams	true	Beta	1.22	1.22
IngressClassNameSpacedParams	true	GA	1.23	1.24
Initializers	false	Alpha	1.7	1.13
Initializers	-	Deprecated	1.14	1.14
KMSv1	true	Deprecated	1.28	
KubeletConfigFile	false	Alpha	1.8	1.9
KubeletConfigFile	-	Deprecated	1.10	1.10
KubeletCredentialProviders	false	Alpha	1.20	1.23
KubeletCredentialProviders	true	Beta	1.24	1.25
KubeletCredentialProviders	true	GA	1.26	1.28
KubeletPluginsWatcher	false	Alpha	1.11	1.11
KubeletPluginsWatcher	true	Beta	1.12	1.12
KubeletPluginsWatcher	true	GA	1.13	1.16
LegacyNodeRoleBehavior	false	Alpha	1.16	1.18
LegacyNodeRoleBehavior	true	Beta	1.19	1.20
LegacyNodeRoleBehavior	false	GA	1.21	1.22
LocalStorageCapacityIsolation	false	Alpha	1.7	1.9

Feature	Default	Stage	From	To
LocalStorageCapacityIsolation	true	Beta	1.10	1.24
LocalStorageCapacityIsolation	true	GA	1.25	1.26
MixedProtocolLBService	false	Alpha	1.20	1.23
MixedProtocolLBService	true	Beta	1.24	1.25
MixedProtocolLBService	true	GA	1.26	1.27
MountContainers	false	Alpha	1.9	1.16
MountContainers	false	Deprecated	1.17	1.17
MountPropagation	false	Alpha	1.8	1.9
MountPropagation	true	Beta	1.10	1.11
MountPropagation	true	GA	1.12	1.14
NamespaceDefaultLabelName	true	Beta	1.21	1.21
NamespaceDefaultLabelName	true	GA	1.22	1.23
NetworkPolicyEndPort	false	Alpha	1.21	1.21
NetworkPolicyEndPort	true	Beta	1.22	1.24
NetworkPolicyEndPort	true	GA	1.25	1.26
NetworkPolicyStatus	false	Alpha	1.24	1.27
NodeDisruptionExclusion	false	Alpha	1.16	1.18
NodeDisruptionExclusion	true	Beta	1.19	1.20
NodeDisruptionExclusion	true	GA	1.21	1.22
NodeLease	false	Alpha	1.12	1.13
NodeLease	true	Beta	1.14	1.16
NodeLease	true	GA	1.17	1.23
NonPreemptingPriority	false	Alpha	1.15	1.18
NonPreemptingPriority	true	Beta	1.19	1.23
NonPreemptingPriority	true	GA	1.24	1.25
PVCProtection	false	Alpha	1.9	1.9
PVCProtection	-	Deprecated	1.10	1.10
PersistentLocalVolumes	false	Alpha	1.7	1.9
PersistentLocalVolumes	true	Beta	1.10	1.13
PersistentLocalVolumes	true	GA	1.14	1.16
PodAffinityNamespaceSelector	false	Alpha	1.21	1.21
PodAffinityNamespaceSelector	true	Beta	1.22	1.23
PodAffinityNamespaceSelector	true	GA	1.24	1.25
PodDisruptionBudget	false	Alpha	1.3	1.4
PodDisruptionBudget	true	Beta	1.5	1.20
PodDisruptionBudget	true	GA	1.21	1.25
PodHasNetworkCondition	false	Alpha	1.25	1.27
PodOverhead	false	Alpha	1.16	1.17
PodOverhead	true	Beta	1.18	1.23
PodOverhead	true	GA	1.24	1.25
PodPriority	false	Alpha	1.8	1.10
PodPriority	true	Beta	1.11	1.13
PodPriority	true	GA	1.14	1.18
PodReadinessGates	false	Alpha	1.11	1.11

Feature	Default	Stage	From	To
PodReadinessGates	true	Beta	1.12	1.13
PodReadinessGates	true	GA	1.14	1.16
PodSecurity	false	Alpha	1.22	1.22
PodSecurity	true	Beta	1.23	1.24
PodSecurity	true	GA	1.25	1.27
PodShareProcessNamespace	false	Alpha	1.10	1.11
PodShareProcessNamespace	true	Beta	1.12	1.16
PodShareProcessNamespace	true	GA	1.17	1.19
PreferNominatedNode	false	Alpha	1.21	1.21
PreferNominatedNode	true	Beta	1.22	1.23
PreferNominatedNode	true	GA	1.24	1.25
RequestManagement	false	Alpha	1.15	1.16
RequestManagement	-	Deprecated	1.17	1.17
ResourceLimitsPriorityFunction	false	Alpha	1.9	1.18
ResourceLimitsPriorityFunction	-	Deprecated	1.19	1.19
ResourceQuotaScopeSelectors	false	Alpha	1.11	1.11
ResourceQuotaScopeSelectors	true	Beta	1.12	1.16
ResourceQuotaScopeSelectors	true	GA	1.17	1.18
RootCAConfigMap	false	Alpha	1.13	1.19
RootCAConfigMap	true	Beta	1.20	1.20
RootCAConfigMap	true	GA	1.21	1.22
RotateKubeletClientCertificate	true	Beta	1.8	1.18
RotateKubeletClientCertificate	true	GA	1.19	1.21
RunAsGroup	true	Beta	1.14	1.20
RunAsGroup	true	GA	1.21	1.22
RuntimeClass	false	Alpha	1.12	1.13
RuntimeClass	true	Beta	1.14	1.19
RuntimeClass	true	GA	1.20	1.24
SCTPSupport	false	Alpha	1.12	1.18
SCTPSupport	true	Beta	1.19	1.19
SCTPSupport	true	GA	1.20	1.22
ScheduleDaemonSetPods	false	Alpha	1.11	1.11
ScheduleDaemonSetPods	true	Beta	1.12	1.16
ScheduleDaemonSetPods	true	GA	1.17	1.18
SelectorIndex	false	Alpha	1.18	1.18
SelectorIndex	true	Beta	1.19	1.19
SelectorIndex	true	GA	1.20	1.25
ServiceAccountIssuerDiscovery	false	Alpha	1.18	1.19
ServiceAccountIssuerDiscovery	true	Beta	1.20	1.20
ServiceAccountIssuerDiscovery	true	GA	1.21	1.23
ServiceAppProtocol	false	Alpha	1.18	1.18
ServiceAppProtocol	true	Beta	1.19	1.19
ServiceAppProtocol	true	GA	1.20	1.22
ServiceIPStaticSubrange	false	Alpha	1.24	1.24

Feature	Default	Stage	From	To
ServiceIPStaticSubrange	true	Beta	1.25	1.25
ServiceIPStaticSubrange	true	GA	1.26	1.27
ServiceInternalTrafficPolicy	false	Alpha	1.21	1.21
ServiceInternalTrafficPolicy	true	Beta	1.22	1.25
ServiceInternalTrafficPolicy	true	GA	1.26	1.27
ServiceLBNodePortControl	false	Alpha	1.20	1.21
ServiceLBNodePortControl	true	Beta	1.22	1.23
ServiceLBNodePortControl	true	GA	1.24	1.25
ServiceLoadBalancerClass	false	Alpha	1.21	1.21
ServiceLoadBalancerClass	true	Beta	1.22	1.23
ServiceLoadBalancerClass	true	GA	1.24	1.25
ServiceLoadBalancerFinalizer	false	Alpha	1.15	1.15
ServiceLoadBalancerFinalizer	true	Beta	1.16	1.16
ServiceLoadBalancerFinalizer	true	GA	1.17	1.20
ServiceNodeExclusion	false	Alpha	1.8	1.18
ServiceNodeExclusion	true	Beta	1.19	1.20
ServiceNodeExclusion	true	GA	1.21	1.22
ServiceTopology	false	Alpha	1.17	1.19
ServiceTopology	false	Deprecated	1.20	1.22
SetHostnameAsFQDN	false	Alpha	1.19	1.19
SetHostnameAsFQDN	true	Beta	1.20	1.21
SetHostnameAsFQDN	true	GA	1.22	1.24
StartupProbe	false	Alpha	1.16	1.17
StartupProbe	true	Beta	1.18	1.19
StartupProbe	true	GA	1.20	1.23
StatefulSetMinReadySeconds	false	Alpha	1.22	1.22
StatefulSetMinReadySeconds	true	Beta	1.23	1.24
StatefulSetMinReadySeconds	true	GA	1.25	1.26
StorageObjectInUseProtection	true	Beta	1.10	1.10
StorageObjectInUseProtection	true	GA	1.11	1.24
StreamingProxyRedirects	false	Beta	1.5	1.5
StreamingProxyRedirects	true	Beta	1.6	1.17
StreamingProxyRedirects	true	Deprecated	1.18	1.21
StreamingProxyRedirects	false	Deprecated	1.22	1.24
SupportIPVSProxyMode	false	Alpha	1.8	1.8
SupportIPVSProxyMode	false	Beta	1.9	1.9
SupportIPVSProxyMode	true	Beta	1.10	1.10
SupportIPVSProxyMode	true	GA	1.11	1.20
SupportNodePidsLimit	false	Alpha	1.14	1.14
SupportNodePidsLimit	true	Beta	1.15	1.19
SupportNodePidsLimit	true	GA	1.20	1.23
SupportPodPidsLimit	false	Alpha	1.10	1.13
SupportPodPidsLimit	true	Beta	1.14	1.19
SupportPodPidsLimit	true	GA	1.20	1.23

Feature	Default	Stage	From	To
SuspendJob	false	Alpha	1.21	1.21
SuspendJob	true	Beta	1.22	1.23
SuspendJob	true	GA	1.24	1.25
Sysctls	true	Beta	1.11	1.20
Sysctls	true	GA	1.21	1.22
TTLAfterFinished	false	Alpha	1.12	1.20
TTLAfterFinished	true	Beta	1.21	1.22
TTLAfterFinished	true	GA	1.23	1.24
TaintBasedEvictions	false	Alpha	1.6	1.12
TaintBasedEvictions	true	Beta	1.13	1.17
TaintBasedEvictions	true	GA	1.18	1.20
TaintNodesByCondition	false	Alpha	1.8	1.11
TaintNodesByCondition	true	Beta	1.12	1.16
TaintNodesByCondition	true	GA	1.17	1.18
TokenRequest	false	Alpha	1.10	1.11
TokenRequest	true	Beta	1.12	1.19
TokenRequest	true	GA	1.20	1.21
TokenRequestProjection	false	Alpha	1.11	1.11
TokenRequestProjection	true	Beta	1.12	1.19
TokenRequestProjection	true	GA	1.20	1.21
UserNamespacesStatelessPodsSupport	false	Alpha	1.25	1.27
ValidateProxyRedirects	false	Alpha	1.12	1.13
ValidateProxyRedirects	true	Beta	1.14	1.21
ValidateProxyRedirects	true	Deprecated	1.22	1.24
VolumePVCDatasource	false	Alpha	1.15	1.15
VolumePVCDatasource	true	Beta	1.16	1.17
VolumePVCDatasource	true	GA	1.18	1.21
VolumeScheduling	false	Alpha	1.9	1.9
VolumeScheduling	true	Beta	1.10	1.12
VolumeScheduling	true	GA	1.13	1.16
VolumeSnapshotDatasource	false	Alpha	1.12	1.16
VolumeSnapshotDatasource	true	Beta	1.17	1.19
VolumeSnapshotDatasource	true	GA	1.20	1.22
VolumeSubpath	true	GA	1.10	1.24
VolumeSubpathEnvExpansion	false	Alpha	1.14	1.14
VolumeSubpathEnvExpansion	true	Beta	1.15	1.16
VolumeSubpathEnvExpansion	true	GA	1.17	1.24
WarningHeaders	true	Beta	1.19	1.21
WarningHeaders	true	GA	1.22	1.24
WindowsEndpointSliceProxying	false	Alpha	1.19	1.20
WindowsEndpointSliceProxying	true	Beta	1.21	1.21
WindowsEndpointSliceProxying	true	GA	1.22	1.24
WindowsGMSA	false	Alpha	1.14	1.15
WindowsGMSA	true	Beta	1.16	1.17

Feature	Default	Stage	From	To
WindowsGMSA	true	GA	1.18	1.20
WindowsHostProcessContainers	false	Alpha	1.22	1.22
WindowsHostProcessContainers	true	Beta	1.23	1.25
WindowsHostProcessContainers	true	GA	1.26	1.27
WindowsRunAsUserName	false	Alpha	1.16	1.16
WindowsRunAsUserName	true	Beta	1.17	1.17
WindowsRunAsUserName	true	GA	1.18	1.20

Descriptions for removed feature gates

- Accelerators: Provided an early form of plugin to enable Nvidia GPU support when using Docker Engine; no longer available. See [Device Plugins](#) for an alternative.
- AffinityInAnnotations: Enable setting [Pod affinity or anti-affinity](#).
- AdvancedAuditing: Enable [advanced auditing](#)
- AllowExtTrafficLocalEndpoints: Enable a service to route external requests to node local endpoints.
- AllowInsecureBackendProxy: Enable the users to skip TLS verification of kubelets on Pod log requests.
- AttachVolumeLimit: Enable volume plugins to report limits on number of volumes that can be attached to a node. See [dynamic volume limits](#) for more details.
- BalanceAttachedNodeVolumes: Include volume count on node to be considered for balanced resource allocation while scheduling. A node which has closer CPU, memory utilization, and volume count is favored by the scheduler while making decisions.
- BlockVolume: Enable the definition and consumption of raw block devices in Pods. See [Raw Block Volume Support](#) for more details.
- BoundServiceAccountTokenVolume: Migrate ServiceAccount volumes to use a projected volume consisting of a ServiceAccountTokenVolumeProjection. Cluster admins can use metric serviceaccount_stale_tokens_total to monitor workloads that are depending on the extended tokens. If there are no such workloads, turn off extended tokens by starting kube-apiserver with flag --service-account-extend-token-expiration=false. Check [Bound Service Account Tokens](#) for more details.
- CRIContainerLogRotation: Enable container log rotation for CRI container runtime. The default max size of a log file is 10MB and the default max number of log files allowed for a container is 5. These values can be configured in the kubelet config. See [logging at node level](#) for more details.
- CSIBlockVolume: Enable external CSI volume drivers to support block storage. See [csi raw block volume support](#) for more details.
- CSIDriverRegistry: Enable all logic related to the CSIDriver API object in csi.storage.k8s.io.

- CSIIInlineVolume: Enable CSI Inline volumes support for pods.
- CSIMigration: Enables shims and translation logic to route volume operations from in-tree plugins to corresponding pre-installed CSI plugins
- CSIMigrationAWS: Enables shims and translation logic to route volume operations from the AWS-EBS in-tree plugin to EBS CSI plugin. Supports falling back to in-tree EBS plugin for mount operations to nodes that have the feature disabled or that do not have EBS CSI plugin installed and configured. Does not support falling back for provision operations, for those the CSI plugin must be installed and configured.
- CSIMigrationAWSComplete: Stops registering the EBS in-tree plugin in kubelet and volume controllers and enables shims and translation logic to route volume operations from the AWS-EBS in-tree plugin to EBS CSI plugin. Requires CSIMigration and CSIMigrationAWS feature flags enabled and EBS CSI plugin installed and configured on all nodes in the cluster. This flag has been deprecated in favor of the InTreePluginAWSUnregister feature flag which prevents the registration of in-tree EBS plugin.
- CSIMigrationAzureDisk: Enables shims and translation logic to route volume operations from the Azure-Disk in-tree plugin to AzureDisk CSI plugin. Supports falling back to in-tree AzureDisk plugin for mount operations to nodes that have the feature disabled or that do not have AzureDisk CSI plugin installed and configured. Does not support falling back for provision operations, for those the CSI plugin must be installed and configured. Requires CSIMigration feature flag enabled.
- CSIMigrationAzureDiskComplete: Stops registering the Azure-Disk in-tree plugin in kubelet and volume controllers and enables shims and translation logic to route volume operations from the Azure-Disk in-tree plugin to AzureDisk CSI plugin. Requires CSIMigration and CSIMigrationAzureDisk feature flags enabled and AzureDisk CSI plugin installed and configured on all nodes in the cluster. This flag has been deprecated in favor of the InTreePluginAzureDiskUnregister feature flag which prevents the registration of in-tree AzureDisk plugin.
- CSIMigrationAzureFileComplete: Stops registering the Azure-File in-tree plugin in kubelet and volume controllers and enables shims and translation logic to route volume operations from the Azure-File in-tree plugin to AzureFile CSI plugin. Requires CSIMigration and CSIMigrationAzureFile feature flags enabled and AzureFile CSI plugin installed and configured on all nodes in the cluster. This flag has been deprecated in favor of the InTreePluginAzureFileUnregister feature flag which prevents the registration of in-tree AzureFile plugin.
- CSIMigrationGCE: Enables shims and translation logic to route volume operations from the GCE-PD in-tree plugin to PD CSI plugin. Supports falling back to in-tree GCE plugin for mount operations to nodes that have the feature disabled or that do not have PD CSI plugin installed and configured. Does not support falling back for provision operations, for those the CSI plugin must be installed and configured. Requires CSIMigration feature flag enabled.
- CSIMigrationGCEComplete: Stops registering the GCE-PD in-tree plugin in kubelet and volume controllers and enables shims and translation logic to route volume operations from the GCE-PD in-tree plugin to PD CSI plugin. Requires CSIMigration and CSIMigrationGCE feature flags enabled and PD CSI plugin installed and configured on all nodes in the cluster. This flag has been deprecated in favor of the

InTreePluginGCEUnregister feature flag which prevents the registration of in-tree GCE PD plugin.

- CSIMigrationOpenStack: Enables shims and translation logic to route volume operations from the Cinder in-tree plugin to Cinder CSI plugin. Supports falling back to in-tree Cinder plugin for mount operations to nodes that have the feature disabled or that do not have Cinder CSI plugin installed and configured. Does not support falling back for provision operations, for those the CSI plugin must be installed and configured. Requires CSIMigration feature flag enabled.
- CSIMigrationOpenStackComplete: Stops registering the Cinder in-tree plugin in kubelet and volume controllers and enables shims and translation logic to route volume operations from the Cinder in-tree plugin to Cinder CSI plugin. Requires CSIMigration and CSIMigrationOpenStack feature flags enabled and Cinder CSI plugin installed and configured on all nodes in the cluster. This flag has been deprecated in favor of the InTreePluginOpenStackUnregister feature flag which prevents the registration of in-tree openstack cinder plugin.
- CSIMigrationvSphereComplete: Stops registering the vSphere in-tree plugin in kubelet and volume controllers and enables shims and translation logic to route volume operations from the vSphere in-tree plugin to vSphere CSI plugin. Requires CSIMigration and CSIMigrationvSphere feature flags enabled and vSphere CSI plugin installed and configured on all nodes in the cluster. This flag has been deprecated in favor of the InTreePluginvSphereUnregister feature flag which prevents the registration of in-tree vsphere plugin.
- CSINodeInfo: Enable all logic related to the CSINodeInfo API object in `csi.storage.k8s.io`.
- CSIPersistentVolume: Enable discovering and mounting volumes provisioned through a [CSI \(Container Storage Interface\)](#) compatible volume plugin.
- CSIServiceAccountToken: Enable CSI drivers to receive the pods' service account token that they mount volumes for. See [Token Requests](#).
- CSIStorageCapacity: Enables CSI drivers to publish storage capacity information and the Kubernetes scheduler to use that information when scheduling pods. See [Storage Capacity](#). Check the [csi volume type](#) documentation for more details.
- CSIVolumeFSGroupPolicy: Allows CSIDrivers to use the `fsGroupPolicy` field. This field controls whether volumes created by a CSIDriver support volume ownership and permission modifications when these volumes are mounted.
- CSRDuration: Allows clients to request a duration for certificates issued via the Kubernetes CSR API.
- ConfigurableFSGroupPolicy: Allows user to configure volume permission change policy for `fsGroups` when mounting a volume in a Pod. See [Configure volume permission and ownership change policy for Pods](#) for more details.
- CronJobControllerV2: Use an alternative implementation of the [CronJob](#) controller. Otherwise, version 1 of the same controller is selected.
- ControllerManagerLeaderMigration: Enables Leader Migration for [kube-controller-manager](#) and [cloud-controller-manager](#) which allows a cluster operator to live migrate

controllers from the kube-controller-manager into an external controller-manager (e.g. the cloud-controller-manager) in an HA cluster without downtime.

- CustomPodDNS: Enable customizing the DNS settings for a Pod using its dnsConfig property. Check [Pod's DNS Config](#) for more details.
- CustomResourceDefaulting: Enable CRD support for default values in OpenAPI v3 validation schemas.
- CustomResourcePublishOpenAPI: Enables publishing of CRD OpenAPI specs.
- CustomResourceSubresources: Enable /status and /scale subresources on resources created from [CustomResourceDefinition](#).
- CustomResourceValidation: Enable schema based validation on resources created from [CustomResourceDefinition](#).
- CustomResourceWebhookConversion: Enable webhook-based conversion on resources created from [CustomResourceDefinition](#).
- DaemonSetUpdateSurge: Enables the DaemonSet workloads to maintain availability during update per node. See [Perform a Rolling Update on a DaemonSet](#).
- DefaultPodTopologySpread: Enables the use of PodTopologySpread scheduling plugin to do [default spreading](#).
- DelegateFSGroupToCSIDriver: If supported by the CSI driver, delegates the role of applying fsGroup from a Pod's securityContext to the driver by passing fsGroup through the NodeStageVolume and NodePublishVolume CSI calls.
- DevicePlugins: Enable the [device-plugins](#) based resource provisioning on nodes.
- DisableAcceleratorUsageMetrics: [Disable accelerator metrics collected by the kubelet](#).
- DryRun: Enable server-side [dry run](#) requests so that validation, merging, and mutation can be tested without committing.
- DynamicAuditing: Used to enable dynamic auditing before v1.19.
- DynamicKubeletConfig: Enable the dynamic configuration of kubelet. The feature is no longer supported outside of supported skew policy. The feature gate was removed from kubelet in 1.24.
- DynamicProvisioningScheduling: Extend the default scheduler to be aware of volume topology and handle PV provisioning. This feature was superseded by the VolumeScheduling feature in v1.12.
- DynamicVolumeProvisioning: Enable the [dynamic provisioning](#) of persistent volumes to Pods.
- EnableAggregatedDiscoveryTimeout: Enable the five second timeout on aggregated discovery calls.
- EnableEquivalenceClassCache: Enable the scheduler to cache equivalence of nodes when scheduling Pods.

- EndpointSlice: Enables EndpointSlices for more scalable and extensible network endpoints. See [Enabling EndpointSlices](#).
- EndpointSliceNodeName: Enables EndpointSlice nodeName field.
- EndpointSliceProxying: When enabled, kube-proxy running on Linux will use EndpointSlices as the primary data source instead of Endpoints, enabling scalability and performance improvements. See [Enabling Endpoint Slices](#).
- EndpointSliceTerminatingCondition: Enables EndpointSlice terminating and serving condition fields.
- EphemeralContainers: Enable the ability to add [ephemeral containers](#) to running Pods.
- EvenPodsSpread: Enable pods to be scheduled evenly across topology domains. See [Pod Topology Spread Constraints](#).
- ExpandCSIVolumes: Enable the expanding of CSI volumes.
- ExpandInUsePersistentVolumes: Enable expanding in-use PVCs. See [Resizing an in-use PersistentVolumeClaim](#).
- ExpandPersistentVolumes: Enable the expanding of persistent volumes. See [Expanding Persistent Volumes Claims](#).
- ExperimentalCriticalPodAnnotation: Enable annotating specific pods as *critical* so that their [scheduling is guaranteed](#). This feature is deprecated by Pod Priority and Preemption as of v1.13.
- ExternalPolicyForExternalIP: Fix a bug where ExternalTrafficPolicy is not applied to Service ExternalIPs.
- GCERegionalPersistentDisk: Enable the regional PD feature on GCE.
- GenericEphemeralVolume: Enables ephemeral, inline volumes that support all features of normal volumes (can be provided by third-party storage vendors, storage capacity tracking, restore from snapshot, etc.). See [Ephemeral Volumes](#).
- HugePageStorageMediumSize: Enable support for multiple sizes pre-allocated [huge pages](#).
- HugePages: Enable the allocation and consumption of pre-allocated [huge pages](#).
- HyperVContainer: Enable [Hyper-V isolation](#) for Windows containers.
- IPv6DualStack: Enable [dual stack](#) support for IPv6.
- IdentifyPodOS: Allows the Pod OS field to be specified. This helps in identifying the OS of the pod authoritatively during the API server admission time. In Kubernetes 1.28, the allowed values for the pod.spec.os.name are windows and linux.
- ImmutableEphemeralVolumes: Allows for marking individual Secrets and ConfigMaps as immutable for better safety and performance.
- IndexedJob: Allows the [Job](#) controller to manage Pod completions per completion index.

- IngressClassNamespacedParams: Allow namespace-scoped parameters reference in IngressClass resource. This feature adds two fields - Scope and Namespace to IngressClass.spec.parameters.
- Initializers: Allow asynchronous coordination of object creation using the Initializers admission plugin.
- KubeletConfigFile: Enable loading kubelet configuration from a file specified using a config file. See [setting kubelet parameters via a config file](#) for more details.
- KubeletCredentialProviders: Enable kubelet exec credential providers for image pull credentials.
- KubeletPluginsWatcher: Enable probe-based plugin watcher utility to enable kubelet to discover plugins such as [CSI volume drivers](#).
- LegacyNodeRoleBehavior: When disabled, legacy behavior in service load balancers and node disruption will ignore the node-role.kubernetes.io/master label in favor of the feature-specific labels provided by NodeDisruptionExclusion and ServiceNodeExclusion.
- LocalStorageCapacityIsolation: Enable the consumption of [local ephemeral storage](#) and also the sizeLimit property of an [emptyDir volume](#).
- MixedProtocolLBService: Enable using different protocols in the same LoadBalancer type Service instance.
- MountContainers: Enable using utility containers on host as the volume mounter.
- MountPropagation: Enable sharing volume mounted by one container to other containers or pods. For more details, please see [mount propagation](#).
- NamespaceDefaultLabelName: Configure the API Server to set an immutable [label](#) kubernetes.io/metadata.name on all namespaces, containing the namespace name.
- NetworkPolicyStatus: Enable the status subresource for NetworkPolicy objects.
- NodeDisruptionExclusion: Enable use of the Node label node.kubernetes.io/exclude-disruption which prevents nodes from being evacuated during zone failures.
- NodeLease: Enable the new Lease API to report node heartbeats, which could be used as a node health signal.
- NonPreemptingPriority: Enable preemptionPolicy field for PriorityClass and Pod.
- PVCProtection: Enable the prevention of a PersistentVolumeClaim (PVC) from being deleted when it is still used by any Pod.
- PersistentLocalVolumes: Enable the usage of local volume type in Pods. Pod affinity has to be specified if requesting a local volume.
- PodAffinityNamespaceSelector: Enable the [Pod Affinity Namespace Selector](#) and [CrossNamespacePodAffinity](#) quota scope features.
- PodDisruptionBudget: Enable the [PodDisruptionBudget](#) feature.

- PodHasNetworkCondition: Enable the kubelet to mark the [PodHasNetwork](#) condition on pods. This was renamed to PodReadyToStartContainersCondition in 1.28.
- PodOverhead: Enable the [PodOverhead](#) feature to account for pod overheads.
- PodPriority: Enable the descheduling and preemption of Pods based on their [priorities](#).
- PodReadinessGates: Enable the setting of PodReadinessGate field for extending Pod readiness evaluation. See [Pod readiness gate](#) for more details.
- PodSecurity: Enables the PodSecurity admission plugin.
- PodShareProcessNamespace: Enable the setting of shareProcessNamespace in a Pod for sharing a single process namespace between containers running in a pod. More details can be found in [Share Process Namespace between Containers in a Pod](#).
- PreferNominatedNode: This flag tells the scheduler whether the nominated nodes will be checked first before looping through all the other nodes in the cluster.
- RequestManagement: Enables managing request concurrency with prioritization and fairness at each API server. Deprecated by APIPriorityAndFairness since 1.17.
- ResourceLimitsPriorityFunction: Enable a scheduler priority function that assigns a lowest possible score of 1 to a node that satisfies at least one of the input Pod's cpu and memory limits. The intent is to break ties between nodes with same scores.
- ResourceQuotaScopeSelectors: Enable resource quota scope selectors.
- RootCACConfigMap: Configure the kube-controller-manager to publish a [ConfigMap](#) named kube-root-ca.crt to every namespace. This ConfigMap contains a CA bundle used for verifying connections to the kube-apiserver. See [Bound Service Account Tokens](#) for more details.
- RotateKubeletClientCertificate: Enable the rotation of the client TLS certificate on the kubelet. See [kubelet configuration](#) for more details.
- RunAsGroup: Enable control over the primary group ID set on the init processes of containers.
- RuntimeClass: Enable the [RuntimeClass](#) feature for selecting container runtime configurations.
- SCTPSupport: Enables the *SCTP* protocol value in Pod, Service, Endpoints, EndpointSlice, and NetworkPolicy definitions.
- ScheduleDaemonSetPods: Enable DaemonSet Pods to be scheduled by the default scheduler instead of the DaemonSet controller.
- SelectorIndex: Allows label and field based indexes in API server watch cache to accelerate list operations.
- ServiceAccountIssuerDiscovery: Enable OIDC discovery endpoints (issuer and JWKS URLs) for the service account issuer in the API server. See [Configure Service Accounts for Pods](#) for more details.

- ServiceAppProtocol: Enables the appProtocol field on Services and Endpoints.
- ServiceIPStaticSubrange: Enables a strategy for Services ClusterIP allocations, whereby the ClusterIP range is subdivided. Dynamic allocated ClusterIP addresses will be allocated preferably from the upper range allowing users to assign static ClusterIPs from the lower range with a low risk of collision. See [Avoiding collisions](#) for more details.
- ServiceInternalTrafficPolicy: Enables the internalTrafficPolicy field on Services.
- ServiceLoadBalancerClass: Enables the loadBalancerClass field on Services. See [Specifying class of load balancer implementation](#) for more details.
- ServiceLoadBalancerFinalizer: Enable finalizer protection for Service load balancers.
- ServiceLBNodePortControl: Enables the allocateLoadBalancerNodePorts field on Services.
- ServiceNodeExclusion: Enable the exclusion of nodes from load balancers created by a cloud provider. A node is eligible for exclusion if labelled with "node.kubernetes.io/exclude-from-external-load-balancers".
- ServiceTopology: Enable service to route traffic based upon the Node topology of the cluster.
- SetHostnameAsFQDN: Enable the ability of setting Fully Qualified Domain Name(FQDN) as the hostname of a pod. See [Pod's setHostnameAsFQDN field](#).
- StartupProbe: Enable the [startup](#) probe in the kubelet.
- StatefulSetMinReadySeconds: Allows minReadySeconds to be respected by the StatefulSet controller.
- StorageObjectInUseProtection: Postpone the deletion of PersistentVolume or PersistentVolumeClaim objects if they are still being used.
- StreamingProxyRedirects: Instructs the API server to intercept (and follow) redirects from the backend (kubelet) for streaming requests. Examples of streaming requests include the exec, attach and port-forward requests.
- SupportIPVSProxyMode: Enable providing in-cluster service load balancing using IPVS. See [service proxies](#) for more details.
- SupportNodePidsLimit: Enable the support to limiting PIDs on the Node. The parameter pid=<number> in the --system-reserved and --kube-reserved options can be specified to ensure that the specified number of process IDs will be reserved for the system as a whole and for Kubernetes system daemons respectively.
- SupportPodPidsLimit: Enable the support to limiting PIDs in Pods.
- SuspendJob: Enable support to suspend and resume Jobs. For more details, see [the Jobs docs](#).
- Sysctls: Enable support for namespaced kernel parameters (sysctls) that can be set for each pod. See [sysctls](#) for more details.

- TTLAfterFinished: Allow a [TTL controller](#) to clean up resources after they finish execution.
- TaintBasedEvictions: Enable evicting pods from nodes based on taints on Nodes and tolerations on Pods. See [taints and tolerations](#) for more details.
- TaintNodesByCondition: Enable automatic tainting nodes based on [node conditions](#).
- TokenRequest: Enable the TokenRequest endpoint on service account resources.
- TokenRequestProjection: Enable the injection of service account tokens into a Pod through a [projected volume](#).
- UserNamespacesStatelessPodsSupport: Enable user namespace support for stateless Pods. This flag was renamed on newer releases to UserNamespacesSupport.
- ValidateProxyRedirects: This flag controls whether the API server should validate that redirects are only followed to the same host. Only used if the StreamingProxyRedirects flag is enabled.
- VolumePVCDatasource: Enable support for specifying an existing PVC as a DataSource.
- VolumeScheduling: Enable volume topology aware scheduling and make the PersistentVolumeClaim (PVC) binding aware of scheduling decisions. It also enables the usage of [local](#) volume type when used together with the PersistentLocalVolumes feature gate.
- VolumeSnapshotDataSource: Enable volume snapshot data source support.
- VolumeSubpath: Allow mounting a subpath of a volume in a container.
- VolumeSubpathEnvExpansion: Enable subPathExpr field for expanding environment variables into a subPath.
- WarningHeaders: Allow sending warning headers in API responses.
- WindowsEndpointSliceProxying: When enabled, kube-proxy running on Windows will use EndpointSlices as the primary data source instead of Endpoints, enabling scalability and performance improvements. See [Enabling Endpoint Slices](#).
- WindowsGMSA: Enables passing of GMSA credential specs from pods to container runtimes.
- WindowsHostProcessContainers: Enables support for Windows HostProcess containers.
- WindowsRunAsUserName : Enable support for running applications in Windows containers with as a non-default user. See [Configuring RunAsUserName](#) for more details.

kubelet

Synopsis

The kubelet is the primary "node agent" that runs on each node. It can register the node with the apiserver using one of: the hostname; a flag to override the hostname; or specific logic for a cloud provider.

The kubelet works in terms of a PodSpec. A PodSpec is a YAML or JSON object that describes a pod. The kubelet takes a set of PodSpecs that are provided through various mechanisms (primarily through the apiserver) and ensures that the containers described in those PodSpecs are running and healthy. The kubelet doesn't manage containers which were not created by Kubernetes.

Other than from a PodSpec from the apiserver, there are two ways that a container manifest can be provided to the kubelet.

- File: Path passed as a flag on the command line. Files under this path will be monitored periodically for updates. The monitoring period is 20s by default and is configurable via a flag.
- HTTP endpoint: HTTP endpoint passed as a parameter on the command line. This endpoint is checked every 20 seconds (also configurable with a flag).

kubelet [flags]

Options

--address string Default: 0.0.0.0

The IP address for the kubelet to serve on (set to 0.0.0.0 or :: for listening on all interfaces and IP address fam

--allowed-unsafe-sysctls strings

Comma-separated whitelist of unsafe sysctls or unsafe sysctl patterns (ending in *). Use these at your own ri

--anonymous-auth Default: true

Enables anonymous requests to the kubelet server. Requests that are not rejected by another authentication

--authentication-token-webhook

Use the TokenReview API to determine authentication for bearer tokens. (DEPRECATED: This parameter sh

--authentication-token-webhook-cache-ttl duration Default: 2m0s

The duration to cache responses from the webhook token authenticator. (DEPRECATED: This parameter sh

--authorization-mode string Default: AlwaysAllow

Authorization mode for kubelet server. Valid options are "AlwaysAllow" or "Webhook". Webhook mode uses

--authorization-webhook-cache-authorized-ttl duration Default: 5m0s

The duration to cache 'authorized' responses from the webhook authorizer. (DEPRECATED: This parameter

--authorization-webhook-cache-unauthorized-ttl duration Default: 30s

The duration to cache 'unauthorized' responses from the webhook authorizer. (DEPRECATED: This parameter

--bootstrap-kubeconfig string

Path to a kubeconfig file that will be used to get client certificate for kubelet. If the file specified by --kubec

--cert-dir string Default: /var/lib/kubelet/pki

The directory where the TLS certs are located. If --tls-cert-file and --tls-private-key-file are provided, this flag

--cgroup-driver string	Default: cgroupfs
	Driver that the kubelet uses to manipulate cgroups on the host. Possible values: "cgroupfs", "systemd". (DEPRECATED)
--cgroup-root string	Default: "
	Optional root cgroup to use for pods. This is handled by the container runtime on a best effort basis. Default is "".
--cgroups-per-qos	Default: true
	Enable creation of QoS cgroup hierarchy, if true, top level QoS and pod cgroups are created. (DEPRECATED)
--client-ca-file string	
	If set, any request presenting a client certificate signed by one of the authorities in the client-ca-file is authorized.
--cloud-config string	
	The path to the cloud provider configuration file. Empty string for no configuration file. (DEPRECATED: will be removed in v1.10)
--cloud-provider string	
	The provider for cloud services. Set to empty string for running with no cloud provider. If set, the cloud provider configuration file will be loaded.
--cluster-dns strings	
	Comma-separated list of DNS server IP address. This value is used for containers DNS server in case of Pods.
	Note: all DNS servers appearing in the list MUST serve the same set of records otherwise name resolution will fail.
--cluster-domain string	
	Domain for this cluster. If set, kubelet will configure all containers to search this domain in addition to the hosts' domain.
--config string	
	The kubelet will load its initial configuration from this file. The path may be absolute or relative; relative paths are relative to the kubelet's configuration directory.
--config-dir string	Default: "
	Path to a directory to specify drop-ins, allows the user to optionally specify additional configs to overwrite via files in the directory.
	Note: Set the 'KUBELET_CONFIG_DROPIN_DIR_ALPHA' environment variable to specify the directory.
--container-log-max-files int32	Default: 5
	<Warning: Beta feature> Set the maximum number of container log files that can be present for a container.
--container-log-max-size string	Default: 10Mi
	<Warning: Beta feature> Set the maximum size (e.g. 10Mi) of container log file before it is rotated. (DEPRECATED)
--container-runtime-endpoint string	
	The endpoint of remote runtime service. UNIX domain sockets are supported on Linux, while 'npipe' and 'tcp' are supported on Windows.
--contention-profiling	
	Enable block profiling, if profiling is enabled. (DEPRECATED: This parameter should be set via the config file.)
--cpu-cfs-quota	Default: true
	Enable CPU CFS quota enforcement for containers that specify CPU limits. (DEPRECATED: This parameter should be set via the config file.)
--cpu-cfs-quota-period duration	Default: 100ms
	Sets CPU CFS quota period value, cpu.cfs_period_us, defaults to Linux Kernel default. (DEPRECATED: This parameter should be set via the config file.)
--cpu-manager-policy string	Default: none
	The CPU manager policy to use. Possible values: "none", "static". (DEPRECATED: This parameter should be set via the config file.)
--cpu-manager-policy-options string	
	A set of 'key=value' CPU manager policy options to use, to fine tune their behaviour. If not supplied, keep the Linux Kernel default.
--cpu-manager-reconcile-period duration	Default: 10s
	<Warning: Alpha feature> CPU manager reconciliation period. Examples: "10s", or "1m". If not supplied, default is 10s.
--enable-controller-attach-detach	Default: true
	Enables the Attach/Detach controller to manage attachment/detachment of volumes scheduled to this node.
--enable-debugging-handlers	Default: true
	Enables server endpoints for log collection and local running of containers and commands. (DEPRECATED)
--enable-server	Default: true
	Enable the kubelet's server. (DEPRECATED: This parameter should be set via the config file specified by the --config parameter.)

--enforce-node-allocatable	strings	Default: pods
A comma separated list of levels of node allocatable enforcement to be enforced by kubelet. Acceptable options are: hard, soft, soft-qos.		
--event-burst	int32	Default: 100
Maximum size of a bursty event records, temporarily allows event records to burst to this number, while still respecting the --event-qps limit.		
--event-qps	int32	Default: 50
QPS to limit event creations. The number must be >= 0. If 0 will use default QPS (50). (DEPRECATED: This parameter is deprecated and will be removed in a future version.)		
--eviction-hard	string	Default: imagefs.available<15%,memory.available<100Mi,nodefs.available<10%
A set of eviction thresholds (e.g. "memory.available<1Gi") that if met would trigger a pod eviction. On a Linux system, this is expressed as memory pressure.		
--eviction-max-pod-grace-period	int32	Default: 300
Maximum allowed grace period (in seconds) to use when terminating pods in response to a soft eviction threshold.		
--eviction-minimum-reclaim	string	A set of minimum reclaims (e.g. "imagefs.available=2Gi") that describes the minimum amount of resource that must be freed before an eviction can occur.
--eviction-pressure-transition-period	duration	Default: 5m0s
Duration for which the kubelet has to wait before transitioning out of an eviction pressure condition. (DEPRECATED: This parameter is deprecated and will be removed in a future version.)		
--eviction-soft	string	A set of eviction thresholds (e.g. "memory.available<1.5Gi") that if met over a corresponding grace period would trigger a soft eviction.
--eviction-soft-grace-period	string	A set of eviction grace periods (e.g. "memory.available=1m30s") that correspond to how long a soft eviction will wait before transitioning to a hard eviction.
--exit-on-lock-contention		Whether kubelet should exit upon lock-file contention.
--experimental-allocatable-ignore-eviction		Default: false
When set to true, hard eviction thresholds will be ignored while calculating node allocatable. See here for more information.		
--experimental-mounter-path	string	Default: mount
[Experimental] Path of mounter binary. Leave empty to use the default mount. (DEPRECATED: will be removed in a future version.)		
--fail-swap-on		Default: true
Makes the kubelet fail to start if swap is enabled on the node. (DEPRECATED: This parameter should be set in /etc/default/kubelet)		
--feature-gates	<A list of 'key=true/false' pairs>	A set of key=value pairs that describe feature gates for alpha/experimental features. Options are: APIListChunking=true false (BETA - default=true) APIPriorityAndFairness=true false (BETA - default=true) APIResponseCompression=true false (BETA - default=true) APIServerIdentity=true false (BETA - default=true) APIServerTracing=true false (BETA - default=true) AdmissionWebhookMatchConditions=true false (BETA - default=true) AggregatedDiscoveryEndpoint=true false (BETA - default=true) AllAlpha=true false (ALPHA - default=false) AllBeta=true false (BETA - default=false) AnyVolumeDataSource=true false (BETA - default=true) AppArmor=true false (BETA - default=true) CPUManagerPolicyAlphaOptions=true false (ALPHA - default=false) CPUManagerPolicyBetaOptions=true false (BETA - default=true) CPUManagerPolicyOptions=true false (BETA - default=true) CRDValidationRatcheting=true false (ALPHA - default=false) CSIMigrationPortworx=true false (BETA - default=false) CSINodeExpandSecret=true false (BETA - default=true) CSIVolumeHealth=true false (ALPHA - default=false) CloudControllerManagerWebhook=true false (ALPHA - default=false) CloudDualStackNodeIPs=true false (ALPHA - default=false)

ClusterTrustBundle=true|false (ALPHA - default=false)
ComponentSLIs=true|false (BETA - default=true)
ConsistentListFromCache=true|false (ALPHA - default=false)
ContainerCheckpoint=true|false (ALPHA - default=false)
ContextualLogging=true|false (ALPHA - default=false)
CronJobsScheduledAnnotation=true|false (BETA - default=true)
CrossNamespaceVolumeDataSource=true|false (ALPHA - default=false)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - default=false)
CustomResourceValidationExpressions=true|false (BETA - default=true)
DevicePluginCDIDevices=true|false (ALPHA - default=false)
DisableCloudProviders=true|false (ALPHA - default=false)
DisableKubeletCloudCredentialProviders=true|false (ALPHA - default=false)
DynamicResourceAllocation=true|false (ALPHA - default=false)
ElasticIndexedJob=true|false (BETA - default=true)
EventedPLEG=true|false (BETA - default=false)
GracefulNodeShutdown=true|false (BETA - default=true)
GracefulNodeShutdownBasedOnPodPriority=true|false (BETA - default=true)
HPAContainerMetrics=true|false (BETA - default=true)
HPAScaleToZero=true|false (ALPHA - default=false)
HonorPVReclaimPolicy=true|false (ALPHA - default=false)
InPlacePodVerticalScaling=true|false (ALPHA - default=false)
InTreePluginAWSUnregister=true|false (ALPHA - default=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - default=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - default=false)
InTreePluginGCEUnregister=true|false (ALPHA - default=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - default=false)
InTreePluginPortworxUnregister=true|false (ALPHA - default=false)
InTreePluginvSphereUnregister=true|false (ALPHA - default=false)
JobBackoffLimitPerIndex=true|false (ALPHA - default=false)
JobPodFailurePolicy=true|false (BETA - default=true)
JobPodReplacementPolicy=true|false (ALPHA - default=false)
JobReadyPods=true|false (BETA - default=true)
KMSv2=true|false (BETA - default=true)
KMSv2KDF=true|false (BETA - default=false)
KubeProxyDrainingTerminatingNodes=true|false (ALPHA - default=false)
KubeletCgroupDriverFromCRI=true|false (ALPHA - default=false)
KubeletInUserNamespace=true|false (ALPHA - default=false)
KubeletPodResourcesDynamicResources=true|false (ALPHA - default=false)
KubeletPodResourcesGet=true|false (ALPHA - default=false)
KubeletTracing=true|false (BETA - default=true)
LegacyServiceAccountTokenCleanUp=true|false (ALPHA - default=false)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - default=false)
LogarithmicScaleDown=true|false (BETA - default=true)
LoggingAlphaOptions=true|false (ALPHA - default=false)
LoggingBetaOptions=true|false (BETA - default=true)
MatchLabelKeysInPodTopologySpread=true|false (BETA - default=true)
MaxUnavailableStatefulSet=true|false (ALPHA - default=false)
MemoryManager=true|false (BETA - default=true)
MemoryQoS=true|false (ALPHA - default=false)
MinDomainsInPodTopologySpread=true|false (BETA - default=true)
MultiCIDRRangeAllocator=true|false (ALPHA - default=false)
MultiCIDRSvcAllocator=true|false (ALPHA - default=false)
NewVolumeManagerReconstruction=true|false (BETA - default=true)

NodeInclusionPolicyInPodTopologySpread=true false (BETA - default=true)
NodeLogQuery=true false (ALPHA - default=false)
NodeSwap=true false (BETA - default=false)
OpenAPIEnums=true false (BETA - default=true)
PDBUnhealthyPodEvictionPolicy=true false (BETA - default=true)
PersistentVolumeLastPhaseTransitionTime=true false (ALPHA - default=false)
PodAndContainerStatsFromCRI=true false (ALPHA - default=false)
PodDeletionCost=true false (BETA - default=true)
PodDisruptionConditions=true false (BETA - default=true)
PodHostIPs=true false (ALPHA - default=false)
PodIndexLabel=true false (BETA - default=true)
PodReadyToStartContainersCondition=true false (ALPHA - default=false)
PodSchedulingReadiness=true false (BETA - default=true)
ProcMountType=true false (ALPHA - default=false)
QOSReserved=true false (ALPHA - default=false)
ReadWriteOncePod=true false (BETA - default=true)
RecoverVolumeExpansionFailure=true false (ALPHA - default=false)
RemainingItemCount=true false (BETA - default=true)
RotateKubeletServerCertificate=true false (BETA - default=true)
SELinuxMountReadWriteOncePod=true false (BETA - default=true)
SchedulerQueueingHints=true false (BETA - default=true)
SecurityContextDeny=true false (ALPHA - default=false)
ServiceNodePortStaticSubrange=true false (BETA - default=true)
SidecarContainers=true false (ALPHA - default=false)
SizeMemoryBackedVolumes=true false (BETA - default=true)
SkipReadOnlyValidationGCE=true false (ALPHA - default=false)
StableLoadBalancerNodeSet=true false (BETA - default=true)
StatefulSetAutoDeletePVC=true false (BETA - default=true)
StatefulSetStartOrdinal=true false (BETA - default=true)
StorageVersionAPI=true false (ALPHA - default=false)
StorageVersionHash=true false (BETA - default=true)
TopologyAwareHints=true false (BETA - default=true)
TopologyManagerPolicyAlphaOptions=true false (ALPHA - default=false)
TopologyManagerPolicyBetaOptions=true false (BETA - default=false)
TopologyManagerPolicyOptions=true false (BETA - default=true)
UnknownVersionInteroperabilityProxy=true false (ALPHA - default=false)
UserNamespacesStatelessPodsSupport=true false (ALPHA - default=false)
ValidatingAdmissionPolicy=true false (BETA - default=false)
VolumeCapacityPriority=true false (ALPHA - default=false)
WatchList=true false (ALPHA - default=false)
WinDSR=true false (ALPHA - default=false)
WinOverlay=true false (BETA - default=true)
WindowsHostNetwork=true false (ALPHA - default=true)(DEPRECATED: This parameter should be set via the --file-check-frequency duration Default: 20s)
Duration between checking config files for new data. (DEPRECATED: This parameter should be set via the --file-check-frequency duration Default: 20s)
--hairpin-mode string Default: promiscuous-bridge
How should the kubelet setup hairpin NAT. This allows endpoints of a Service to load balance back to themselfs.
--healthz-bind-address string Default: 127.0.0.1
The IP address for the healthz server to serve on (set to "0.0.0.0" or ":" for listening in all interfaces and IP family).
--healthz-port int32 Default: 10248
The port of the localhost healthz endpoint (set to 0 to disable). (DEPRECATED: This parameter should be set via the --healthz-port int32 Default: 10248)

-h, --help
help for kubelet
--hostname-override string
If non-empty, will use this string as identification instead of the actual hostname. If --cloud-provider is set, th
--http-check-frequency duration Default: 20s
Duration between checking HTTP for new data. (DEPRECATED: This parameter should be set via the config
--image-credential-provider-bin-dir string
The path to the directory where credential provider plugin binaries are located.
--image-credential-provider-config string
The path to the credential provider plugin config file.
--image-gc-high-threshold int32 Default: 85
The percent of disk usage after which image garbage collection is always run. Values must be within the ran
--image-gc-low-threshold int32 Default: 80
The percent of disk usage before which image garbage collection is never run. Lowest disk usage to garbage
--image-service-endpoint string
The endpoint of remote image service. If not specified, it will be the same with --container-runtime-endpoi
--keep-terminated-pod-volumes
Keep terminated pod volumes mounted to the node after the pod terminates. Can be useful for debugging v
--kernel-memcg-notification
If enabled, the kubelet will integrate with the kernel memcg notification to determine if memory eviction th
--kube-api-burst int32 Default: 100
Burst to use while talking with kubernetes API server. The number must be >= 0. If 0 will use default burst (
--kube-api-content-type string Default: application/vnd.kubernetes.protobuf
Content type of requests sent to apiserver. (DEPRECATED: This parameter should be set via the config file s
--kube-api-qps int32 Default: 50
QPS to use while talking with kubernetes API server. The number must be >= 0. If 0 will use default QPS (50)
--kube-reserved string Default: <None>
A set of <resource name>=<resource quantity> (e.g. "cpu=200m,memory=500Mi,ephemeral-storage=1Gi,pid=
--kube-reserved-cgroup string Default: "
Absolute name of the top level cgroup that is used to manage kubernetes components for which compute re
--kubeconfig string
Path to a kubeconfig file, specifying how to connect to the API server. Providing --kubeconfig enables API se
--kubelet-cgroups string
Optional absolute name of cgroups to create and run the kubelet in. (DEPRECATED: This parameter should
--local-storage-capacity-isolation> Default: true
If true, local ephemeral storage isolation is enabled. Otherwise, local storage isolation feature will be disabled
--lock-file string
<Warning: Alpha feature> The path to file for kubelet to use as a lock file.
--log-flush-frequency duration Default: 5s
Maximum number of seconds between log flushes.
--log-json-info-buffer-size string Default: '0'
[Alpha] In JSON format with split output streams, the info messages can be buffered for a while to increase p
--log-json-split-stream
[Alpha] In JSON format, write error messages to stderr and info messages to stdout. The default is to write a
--logging-format string Default: text

Sets the log format. Permitted formats: "json" (gated by LoggingBetaOptions, "text"). (DEPRECATED: This parameter should be set via the config file)
--make-iptables-util-chains Default: true
If true, kubelet will ensure iptables utility rules are present on host. (DEPRECATED: This parameter should be set via the config file)
--manifest-url string
URL for accessing additional Pod specifications to run. (DEPRECATED: This parameter should be set via the config file)
--manifest-url-header string
Comma-separated list of HTTP headers to use when accessing the URL provided to --manifest-url. Multiple values are separated by commas.
--max-open-files int Default: 1000000
Number of files that can be opened by kubelet process. (DEPRECATED: This parameter should be set via the config file)
--max-pods int32 Default: 110
Number of Pods that can run on this kubelet. (DEPRECATED: This parameter should be set via the config file)
--maximum-dead-containers int32 Default: -1
Maximum number of old instances of containers to retain globally. Each container takes up some disk space.
--maximum-dead-containers-per-container int32 Default: 1
Maximum number of old instances to retain per container. Each container takes up some disk space. (DEPRECATED: This parameter should be set via the config file)
--memory-manager-policy string Default: None
Memory Manager policy to use. Possible values: "None", "Static". (DEPRECATED: This parameter should be set via the config file)
--minimum-container-ttl-duration duration
Minimum age for a finished container before it is garbage collected. Examples: "300ms", "10s" or "2h45m". (DEPRECATED: This parameter should be set via the config file)
--minimum-image-ttl-duration duration Default: 2m0s
Minimum age for an unused image before it is garbage collected. Examples: "300ms", "10s" or "2h45m". (DEPRECATED: This parameter should be set via the config file)
--node-ip string
IP address (or comma-separated dual-stack IP addresses) of the node. If unset, kubelet will use the node's default IP address.
--node-labels <key=value pairs>
<Warning: Alpha feature>Labels to add when registering the node in the cluster. Labels must be key=value pairs.
--node-status-max-images int32 Default: 50
The maximum number of images to report in node.status.images. If -1 is specified, no cap will be applied. (DEPRECATED: This parameter should be set via the config file)
--node-status-update-frequency duration Default: 10s
Specifies how often kubelet posts node status to master. Note: be cautious when changing the constant, it may affect the system stability.
--oom-score-adj int32 Default: -999
The oom-score-adj value for kubelet process. Values must be within the range [-1000, 1000]. (DEPRECATED: This parameter should be set via the config file)
--pod-cidr string
The CIDR to use for pod IP addresses, only used in standalone mode. In cluster mode, this is obtained from the network plugin.
--pod-infra-container-image string Default: registry.k8s.io/pause:3.9
Specified image will not be pruned by the image garbage collector. CRI implementations have their own configuration for this.
--pod-manifest-path string
Path to the directory containing static pod files to run, or the path to a single static pod file. Files starting with a dot are ignored.
--pod-max-pids int Default: -1
Set the maximum number of processes per pod. If -1, the kubelet defaults to the node allocatable PID capacity.
--pods-per-core int32
Number of Pods per core that can run on this kubelet. The total number of pods on this kubelet cannot exceed the number of cores times the number of pods per core.
--port int32 Default: 10250
The port for the kubelet to serve on. (DEPRECATED: This parameter should be set via the config file specified in the --config flag)
--protect-kernel-defaults
Default kubelet behaviour for kernel tuning. If set, kubelet errors if any of kernel tunables is different than kernel default.

--provider-id string	Unique identifier for identifying the node in a machine database, i.e cloud provider. (DEPRECATED: This parameter is deprecated and will be removed in a future release.)
--qos-reserved string	<Warning: Alpha feature> A set of <resource name>=<percentage> (e.g. "memory=50%") pairs that describe the percentage of a resource to be reserved for pods.
--read-only-port int32 Default: 10255	The read-only port for the kubelet to serve on with no authentication/authorization (set to 0 to disable). (DEPRECATED: This parameter is deprecated and will be removed in a future release.)
--register-node Default: true	Register the node with the API server. If --kubeconfig is not provided, this flag is irrelevant, as the kubelet will register with the master by default.
--register-schedulable Default: true	Register the node as schedulable. Won't have any effect if --register-node is false. (DEPRECATED: will be removed in a future release.)
--register-with-taints string	Register the node with the given list of taints (comma separated <key>=<value>:<effect>). No-op if --register-node is false.
--registry-burst int32 Default: 10	Maximum size of a bursty pulls, temporarily allows pulls to burst to this number, while still not exceeding --registry-qps.
--registry-qps int32 Default: 5	If > 0, limit registry pull QPS to this value. If 0, unlimited. (DEPRECATED: This parameter should be set via the environment variable KUBELET_REGISTRY_QPS.)
--reserved-cpus string	A comma-separated list of CPUs or CPU ranges that are reserved for system and kubernetes usage. This specifies the list of CPUs reserved for the kubelet.
--reserved-memory string	A comma-separated list of memory reservations for NUMA nodes. (e.g. "--reserved-memory 0:memory=1Gi,1:memory=1Gi").
--resolv-conf string Default: /etc/resolv.conf	Resolver configuration file used as the basis for the container DNS resolution configuration. (DEPRECATED: This parameter is deprecated and will be removed in a future release.)
--root-dir string Default: /var/lib/kubelet	Directory path for managing kubelet files (volume mounts, etc).
--rotate-certificates	Auto rotate the kubelet client certificates by requesting new certificates from the kube-apiserver when the current ones expire.
--rotate-server-certificates	Auto-request and rotate the kubelet serving certificates by requesting new certificates from the kube-apiserver when the current ones expire.
--runonce	If true, exit after spawning pods from local manifests or remote urls. Exclusive with --enable-server (DEPRECATED: This parameter is deprecated and will be removed in a future release.)
--runtime-cgroups string	Optional absolute name of cgroups to create and run the runtime in.
--runtime-request-timeout duration Default: 2m0s	Timeout of all runtime requests except long running request - pull, logs, exec and attach. When timeout exceeds, the request will be terminated.
--seccomp-default Default: false	Enable the use of RuntimeDefault as the default seccomp profile for all workloads.
--serialize-image-pulls Default: true	Pull images one at a time. We recommend *not* changing the default value on nodes that run docker daemon.
--streaming-connection-idle-timeout duration Default: 4h0m0s	Maximum time a streaming connection can be idle before the connection is automatically closed. 0 indicates infinite.
--sync-frequency duration Default: 1m0s	Max period between synchronizing running containers and config. (DEPRECATED: This parameter should be removed in a future release.)
--system-cgroups string	Optional absolute name of cgroups in which to place all non-kernel processes that are not already inside a cgroup.
--system-reserved string Default: <none>	

--system-reserved-cgroup string	Default: "
--tls-cert-file string	Absolute name of the top level cgroup that is used to manage non-kubernetes components for which compute resources are reserved.
--tls-cipher-suites string	File containing x509 certificate used for serving HTTPS (with intermediate certs, if any, concatenated after the root cert).
--tls-min-version string	Comma-separated list of cipher suites for the server. If omitted, the default Go cipher suites will be used. Preferred values: TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256 TLS_RSA_WITH_AES_256_GCM_SHA384 Insecure values: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA (DEPRECATED: This parameter should be set via the config file specified by the kubelet's --config flag. See kubernetes.io/docs/concepts/services-networking/configuring-tls-for-the-api-server)
--tls-private-key-file string	Minimum TLS version supported. Possible values: "VersionTLS10", "VersionTLS11", "VersionTLS12", "VersionTLS13".
--topology-manager-policy string	Topology Manager policy to use. Possible values: "none", "best-effort", "restricted", "single-numa-node". (DEPRECATED: This parameter should be set via the config file specified by the kubelet's --config flag. See kubernetes.io/docs/concepts/services-networking/configuring-topology-awareness)
--topology-manager-policy-options string	A set of <key>=<value> topology manager policy options to use, to fine tune their behaviour. If not supplied, the default values will be used.
--topology-manager-scope string	Default: container
-v, --v Level	Scope to which topology hints are applied. Topology manager collects hints from hint providers and applies them to pods.
--version version[=true]	Number for the log level verbosity
--vmodule <A list of 'pattern=N' strings>	Print version information and quit; --version=vX.Y.Z... sets the reported version.
--volume-plugin-dir string	Default: /usr/libexec/kubernetes/kubelet-plugins/volume/exec/
--volume-stats-agg-period duration	The full path of the directory in which to search for additional third party volume plugins. (DEPRECATED: This parameter should be set via the config file specified by the kubelet's --config flag. See kubernetes.io/docs/concepts/storage/persistent-volumes#third-party-volume-plugins)
--volume-stats-interval duration	Default: 1m0s
--volume-stats-size limit	Specifies interval for kubelet to calculate and cache the volume disk usage for all pods and volumes. To disable, set to 0.

kube-apiserver

Synopsis

The Kubernetes API server validates and configures data for the api objects which include pods, services, replicationcontrollers, and others. The API Server services REST operations and provides the frontend to the cluster's shared state through which all other components interact.

kube-apiserver [flags]

Options

--admission-control-config-file string
--

File with admission control configuration.
--advertise-address string
The IP address on which to advertise the apiserver to members of the cluster. This address must be reachable by the rest of the cluster. If blank, the --bind-address will be used. If --bind-address is unspecified, the host's default interface will be used.
--aggregator-reject-forwarding-redirect Default: true
Aggregator reject forwarding redirect response back to client.
--allow-metric-labels stringToString Default: []
The map from metric-label to value allow-list of this label. The key's format is <MetricName>,<LabelName>. The value's format is <allowed_value>,<allowed_value>...e.g. metric1,label1='v1,v2,v3', metric1,label2='v1,v2,v3' metric2,label1='v1,v2,v3'.
--allow-privileged
If true, allow privileged containers. [default=false]
--anonymous-auth Default: true
Enables anonymous requests to the secure port of the API server. Requests that are not rejected by another authentication method are treated as anonymous requests. Anonymous requests have a username of system:anonymous, and a group name of system:unauthenticated.
--api-audiences strings
Identifiers of the API. The service account token authenticator will validate that tokens used against the API are bound to at least one of these audiences. If the --service-account-issuer flag is configured and this flag is not, this field defaults to a single element list containing the issuer URL.
--audit-log-batch-buffer-size int Default: 10000
The size of the buffer to store events before batching and writing. Only used in batch mode.
--audit-log-batch-max-size int Default: 1
The maximum size of a batch. Only used in batch mode.
--audit-log-batch-max-wait duration
The amount of time to wait before force writing the batch that hadn't reached the max size. Only used in batch mode.
--audit-log-batch-throttle-burst int

Maximum number of requests sent at the same moment if ThrottleQPS was not utilized before. Only used in batch mode.
--audit-log-batch-throttle-enable
Whether batching throttling is enabled. Only used in batch mode.
--audit-log-batch-throttle-qps float
Maximum average number of batches per second. Only used in batch mode.
--audit-log-compress
If set, the rotated log files will be compressed using gzip.
--audit-log-format string Default: "json"
Format of saved audits. "legacy" indicates 1-line text format for each event. "json" indicates structured json format. Known formats are legacy,json.
--audit-log-maxage int
The maximum number of days to retain old audit log files based on the timestamp encoded in their filename.
--audit-log-maxbackup int
The maximum number of old audit log files to retain. Setting a value of 0 will mean there's no restriction on the number of files.
--audit-log-maxsize int
The maximum size in megabytes of the audit log file before it gets rotated.
--audit-log-mode string Default: "blocking"
Strategy for sending audit events. Blocking indicates sending events should block server responses. Batch causes the backend to buffer and write events asynchronously. Known modes are batch,blocking,blocking-strict.
--audit-log-path string
If set, all requests coming to the apiserver will be logged to this file. '-' means standard out.
--audit-log-truncate-enabled
Whether event and batch truncating is enabled.
--audit-log-truncate-max-batch-size int Default: 10485760

Maximum size of the batch sent to the underlying backend. Actual serialized size can be several hundreds of bytes greater. If a batch exceeds this limit, it is split into several batches of smaller size.
--audit-log-truncate-max-event-size int Default: 102400
Maximum size of the audit event sent to the underlying backend. If the size of an event is greater than this number, first request and response are removed, and if this doesn't reduce the size enough, event is discarded.
--audit-log-version string Default: "audit.k8s.io/v1"
API group and version used for serializing audit events written to log.
--audit-policy-file string
Path to the file that defines the audit policy configuration.
--audit-webhook-batch-buffer-size int Default: 10000
The size of the buffer to store events before batching and writing. Only used in batch mode.
--audit-webhook-batch-max-size int Default: 400
The maximum size of a batch. Only used in batch mode.
--audit-webhook-batch-max-wait duration Default: 30s
The amount of time to wait before force writing the batch that hadn't reached the max size. Only used in batch mode.
--audit-webhook-batch-throttle-burst int Default: 15
Maximum number of requests sent at the same moment if ThrottleQPS was not utilized before. Only used in batch mode.
--audit-webhook-batch-throttle-enable Default: true
Whether batching throttling is enabled. Only used in batch mode.
--audit-webhook-batch-throttle-qps float Default: 10
Maximum average number of batches per second. Only used in batch mode.
--audit-webhook-config-file string
Path to a kubeconfig formatted file that defines the audit webhook configuration.
--audit-webhook-initial-backoff duration Default: 10s

	The amount of time to wait before retrying the first failed request.
--audit-webhook-mode	string Default: "batch"
	Strategy for sending audit events. Blocking indicates sending events should block server responses. Batch causes the backend to buffer and write events asynchronously. Known modes are batch,blocking,blocking-strict.
--audit-webhook-truncate-enabled	
	Whether event and batch truncating is enabled.
--audit-webhook-truncate-max-batch-size	int Default: 10485760
	Maximum size of the batch sent to the underlying backend. Actual serialized size can be several hundreds of bytes greater. If a batch exceeds this limit, it is split into several batches of smaller size.
--audit-webhook-truncate-max-event-size	int Default: 102400
	Maximum size of the audit event sent to the underlying backend. If the size of an event is greater than this number, first request and response are removed, and if this doesn't reduce the size enough, event is discarded.
--audit-webhook-version	string Default: "audit.k8s.io/v1"
	API group and version used for serializing audit events written to webhook.
--authentication-token-webhook-cache-ttl	duration Default: 2m0s
	The duration to cache responses from the webhook token authenticator.
--authentication-token-webhook-config-file	string
	File with webhook configuration for token authentication in kubeconfig format. The API server will query the remote service to determine authentication for bearer tokens.
--authentication-token-webhook-version	string Default: "v1beta1"
	The API version of the authentication.k8s.io TokenReview to send to and expect from the webhook.
--authorization-mode	strings Default: "AlwaysAllow"
	Ordered list of plug-ins to do authorization on secure port. Comma-delimited list of: AlwaysAllow,AlwaysDeny,ABAC,Webhook,RBAC,Node.
--authorization-policy-file	string

	File with authorization policy in json line by line format, used with --authorization-mode=ABAC, on the secure port.
--authorization-webhook-cache-authorized-ttl duration	Default: 5m0s
	The duration to cache 'authorized' responses from the webhook authorizer.
--authorization-webhook-cache-unauthorized-ttl duration	Default: 30s
	The duration to cache 'unauthorized' responses from the webhook authorizer.
--authorization-webhook-config-file string	
	File with webhook configuration in kubeconfig format, used with --authorization-mode=Webhook. The API server will query the remote service to determine access on the API server's secure port.
--authorization-webhook-version string	Default: "v1beta1"
	The API version of the authorization.k8s.io SubjectAccessReview to send to and expect from the webhook.
--azure-container-registry-config string	
	Path to the file containing Azure container registry configuration information.
--bind-address string	Default: 0.0.0.0
	The IP address on which to listen for the --secure-port port. The associated interface(s) must be reachable by the rest of the cluster, and by CLI/web clients. If blank or an unspecified address (0.0.0.0 or ::), all interfaces and IP address families will be used.
--cert-dir string	Default: "/var/run/kubernetes"
	The directory where the TLS certs are located. If --tls-cert-file and --tls-private-key-file are provided, this flag will be ignored.
--client-ca-file string	
	If set, any request presenting a client certificate signed by one of the authorities in the client-ca-file is authenticated with an identity corresponding to the CommonName of the client certificate.
--cloud-config string	
	The path to the cloud provider configuration file. Empty string for no configuration file.
--cloud-provider string	
	The provider for cloud services. Empty string for no provider.

--cloud-provider-gce-l7lb-src-cidrs	cidrs Default: 130.211.0.0/22,35.191.0.0/16
CIDRs opened in GCE firewall for L7 LB traffic proxy & health checks	
--contention-profiling	
Enable block profiling, if profiling is enabled	
--cors-allowed-origins	
List of allowed origins for CORS, comma separated. An allowed origin can be a regular expression to support subdomain matching. If this list is empty CORS will not be enabled. Please ensure each expression matches the entire hostname by anchoring to the start with '^' or including the '// prefix, and by anchoring to the end with '\$' or including the ':' port separator suffix. Examples of valid expressions are '//example.com(: \$)' and '^https://example.com(: \$)'	
--debug-socket-path	string
Use an unprotected (no authn/authz) unix-domain socket for profiling with the given path	
--default-not-ready-toleration-seconds	int Default: 300
Indicates the tolerationSeconds of the toleration for notReady:NoExecute that is added by default to every pod that does not already have such a toleration.	
--default-unreachable-toleration-seconds	int Default: 300
Indicates the tolerationSeconds of the toleration for unreachable:NoExecute that is added by default to every pod that does not already have such a toleration.	
--delete-collection-workers	int Default: 1
Number of workers spawned for DeleteCollection call. These are used to speed up namespace cleanup.	
--disable-admission-plugins	
admission plugins that should be disabled although they are in the default enabled plugins list (NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, ClusterTrustBundleAttest, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionPolicy, ValidatingAdmissionWebhook, ResourceQuota). Comma-delimited list of admission plugins: AlwaysAdmit, AlwaysDeny, AlwaysPullImages, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, DenyServiceExternalIPs, EventRateLimit, ExtendedResourceToleration, ImagePolicyWebhook, LimitPodHardAntiAffinityTopology, LimitRanger, MutatingAdmissionWebhook, NamespaceAutoProvision, NamespaceExists, NamespaceLifecycle, NodeRestriction, OwnerReferencesPermissionEnforcement, PersistentVolumeClaimResize, PersistentVolumeLabel, PodNodeSelector, PodSecurity,	

PodTolerationRestriction, Priority, ResourceQuota, RuntimeClass, SecurityContextDeny, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy, ValidatingAdmissionWebhook. The order of plugins in this flag does not matter.
--disabled-metrics strings
This flag provides an escape hatch for misbehaving metrics. You must provide the fully qualified metric name in order to disable it. Disclaimer: disabling metrics is higher in precedence than showing hidden metrics.
--egress-selector-config-file string
File with apiserver egress selector configuration.
--enable-admission-plugins strings
admission plugins that should be enabled in addition to default enabled ones (NamespaceLifecycle, LimitRanger, ServiceAccount, TaintNodesByCondition, PodSecurity, Priority, DefaultTolerationSeconds, DefaultStorageClass, StorageObjectInUseProtection, PersistentVolumeClaimResize, RuntimeClass, CertificateApproval, CertificateSigning, ClusterTrustBundleAttest, CertificateSubjectRestriction, DefaultIngressClass, MutatingAdmissionWebhook, ValidatingAdmissionPolicy, ValidatingAdmissionWebhook, ResourceQuota). Comma-delimited list of admission plugins: AlwaysAdmit, AlwaysDeny, AlwaysPullImages, CertificateApproval, CertificateSigning, CertificateSubjectRestriction, ClusterTrustBundleAttest, DefaultIngressClass, DefaultStorageClass, DefaultTolerationSeconds, DenyServiceExternalIPs, EventRateLimit, ExtendedResourceToleration, ImagePolicyWebhook, LimitPodHardAntiAffinityTopology, LimitRanger, MutatingAdmissionWebhook, NamespaceAutoProvision, NamespaceExists, NamespaceLifecycle, NodeRestriction, OwnerReferencesPermissionEnforcement, PersistentVolumeClaimResize, PersistentVolumeLabel, PodNodeSelector, PodSecurity, PodTolerationRestriction, Priority, ResourceQuota, RuntimeClass, SecurityContextDeny, ServiceAccount, StorageObjectInUseProtection, TaintNodesByCondition, ValidatingAdmissionPolicy, ValidatingAdmissionWebhook. The order of plugins in this flag does not matter.
--enable-aggregator-routing
Turns on aggregator routing requests to endpoints IP rather than cluster IP.
--enable-bootstrap-token-auth
Enable to allow secrets of type 'bootstrap.kubernetes.io/token' in the 'kube-system' namespace to be used for TLS bootstrapping authentication.
--enable-garbage-collector Default: true
Enables the generic garbage collector. MUST be synced with the corresponding flag of the kube-controller-manager.
--enable-priority-and-fairness Default: true

If true and the APIPriorityAndFairness feature gate is enabled, replace the max-in-flight handler with an enhanced one that queues and dispatches with priority and fairness
--encryption-provider-config string
The file containing configuration for encryption providers to be used for storing secrets in etcd
--encryption-provider-config-automatic-reload
Determines if the file set by --encryption-provider-config should be automatically reloaded if the disk contents change. Setting this to true disables the ability to uniquely identify distinct KMS plugins via the API server healthz endpoints.
--endpoint-reconciler-type string Default: "lease"
Use an endpoint reconciler (master-count, lease, none) master-count is deprecated, and will be removed in a future version.
--etcd-cafile string
SSL Certificate Authority file used to secure etcd communication.
--etcd-certfile string
SSL certification file used to secure etcd communication.
--etcd-compaction-interval duration Default: 5m0s
The interval of compaction requests. If 0, the compaction request from apiserver is disabled.
--etcd-count-metric-poll-period duration Default: 1m0s
Frequency of polling etcd for number of resources per type. 0 disables the metric collection.
--etcd-db-metric-poll-interval duration Default: 30s
The interval of requests to poll etcd and update metric. 0 disables the metric collection
--etcd-healthcheck-timeout duration Default: 2s
The timeout to use when checking etcd health.
--etcd-keyfile string
SSL key file used to secure etcd communication.
--etcd-prefix string Default: "/registry"
The prefix to prepend to all resource paths in etcd.
--etcd-readycheck-timeout duration Default: 2s

The timeout to use when checking etcd readiness
--etcd-servers strings
List of etcd servers to connect with (scheme://ip:port), comma separated.
--etcd-servers-overrides strings
Per-resource etcd servers overrides, comma separated. The individual override format: group/resource#servers, where servers are URLs, semicolon separated. Note that this applies only to resources compiled into this server binary.
--event-ttl duration Default: 1h0m0s
Amount of time to retain events.
--external-hostname string
The hostname to use when generating externalized URLs for this master (e.g. Swagger API Docs or OpenID Discovery).
--feature-gates <comma-separated 'key=True False' pairs>
A set of key=value pairs that describe feature gates for alpha/experimental features. Options are: APIListChunking=true false (BETA - default=true) APIPriorityAndFairness=true false (BETA - default=true) APIResponseCompression=true false (BETA - default=true) APIServerIdentity=true false (BETA - default=true) APIServerTracing=true false (BETA - default=true) AdmissionWebhookMatchConditions=true false (BETA - default=true) AggregatedDiscoveryEndpoint=true false (BETA - default=true) AllAlpha=true false (ALPHA - default=false) AllBeta=true false (BETA - default=false) AnyVolumeDataSource=true false (BETA - default=true) AppArmor=true false (BETA - default=true) CPUManagerPolicyAlphaOptions=true false (ALPHA - default=false) CPUManagerPolicyBetaOptions=true false (BETA - default=true) CPUManagerPolicyOptions=true false (BETA - default=true) CRDValidationRatcheting=true false (ALPHA - default=false) CSIMigrationPortworx=true false (BETA - default=false) CSINodeExpandSecret=true false (BETA - default=true) CSIVolumeHealth=true false (ALPHA - default=false) CloudControllerManagerWebhook=true false (ALPHA - default=false) CloudDualStackNodeIPs=true false (ALPHA - default=false) ClusterTrustBundle=true false (ALPHA - default=false) ComponentSLIs=true false (BETA - default=true) ConsistentListFromCache=true false (ALPHA - default=false) ContainerCheckpoint=true false (ALPHA - default=false) ContextualLogging=true false (ALPHA - default=false) CronJobsScheduledAnnotation=true false (BETA - default=true)

CrossNamespaceVolumeDataSource=true|false (ALPHA - default=false)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - default=false)
CustomResourceValidationExpressions=true|false (BETA - default=true)
DevicePluginCDIDevices=true|false (ALPHA - default=false)
DisableCloudProviders=true|false (ALPHA - default=false)
DisableKubeletCloudCredentialProviders=true|false (ALPHA - default=false)
DynamicResourceAllocation=true|false (ALPHA - default=false)
ElasticIndexedJob=true|false (BETA - default=true)
EventedPLEG=true|false (BETA - default=false)
GracefulNodeShutdown=true|false (BETA - default=true)
GracefulNodeShutdownBasedOnPodPriority=true|false (BETA - default=true)
HPAContainerMetrics=true|false (BETA - default=true)
HPAScaleToZero=true|false (ALPHA - default=false)
HonorPVReclaimPolicy=true|false (ALPHA - default=false)
InPlacePodVerticalScaling=true|false (ALPHA - default=false)
InTreePluginAWSUnregister=true|false (ALPHA - default=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - default=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - default=false)
InTreePluginGCEUnregister=true|false (ALPHA - default=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - default=false)
InTreePluginPortworxUnregister=true|false (ALPHA - default=false)
InTreePluginvSphereUnregister=true|false (ALPHA - default=false)
JobBackoffLimitPerIndex=true|false (ALPHA - default=false)
JobPodFailurePolicy=true|false (BETA - default=true)
JobPodReplacementPolicy=true|false (ALPHA - default=false)
JobReadyPods=true|false (BETA - default=true)
KMSv2=true|false (BETA - default=true)
KMSv2KDF=true|false (BETA - default=false)
KubeProxyDrainingTerminatingNodes=true|false (ALPHA - default=false)
KubeletCgroupDriverFromCRI=true|false (ALPHA - default=false)
KubeletInUserNamespace=true|false (ALPHA - default=false)
KubeletPodResourcesDynamicResources=true|false (ALPHA - default=false)
KubeletPodResourcesGet=true|false (ALPHA - default=false)
KubeletTracing=true|false (BETA - default=true)
LegacyServiceAccountTokenCleanUp=true|false (ALPHA - default=false)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - default=false)
LogarithmicScaleDown=true|false (BETA - default=true)
LoggingAlphaOptions=true|false (ALPHA - default=false)
LoggingBetaOptions=true|false (BETA - default=true)
MatchLabelKeysInPodTopologySpread=true|false (BETA - default=true)
MaxUnavailableStatefulSet=true|false (ALPHA - default=false)
MemoryManager=true|false (BETA - default=true)
MemoryQoS=true|false (ALPHA - default=false)
MinDomainsInPodTopologySpread=true|false (BETA - default=true)
MultiCIDRRangeAllocator=true|false (ALPHA - default=false)
MultiCIDRSvcAllocator=true|false (ALPHA - default=false)
NewVolumeManagerReconstruction=true|false (BETA - default=true)
NodeInclusionPolicyInPodTopologySpread=true|false (BETA - default=true)
NodeLogQuery=true|false (ALPHA - default=false)
NodeSwap=true|false (BETA - default=false)
OpenAPIEnums=true|false (BETA - default=true)
PDBUnhealthyPodEvictionPolicy=true|false (BETA - default=true)
PersistentVolumeLastPhaseTransitionTime=true|false (ALPHA - default=false)

PodAndContainerStatsFromCRI=true|false (ALPHA - default=false)
PodDeletionCost=true|false (BETA - default=true)
PodDisruptionConditions=true|false (BETA - default=true)
PodHostIPs=true|false (ALPHA - default=false)
PodIndexLabel=true|false (BETA - default=true)
PodReadyToStartContainersCondition=true|false (ALPHA - default=false)
PodSchedulingReadiness=true|false (BETA - default=true)
ProcMountType=true|false (ALPHA - default=false)
QOSReserved=true|false (ALPHA - default=false)
ReadWriteOncePod=true|false (BETA - default=true)
RecoverVolumeExpansionFailure=true|false (ALPHA - default=false)
RemainingItemCount=true|false (BETA - default=true)
RotateKubeletServerCertificate=true|false (BETA - default=true)
SELinuxMountReadWriteOncePod=true|false (BETA - default=true)
SchedulerQueueingHints=true|false (BETA - default=true)
SecurityContextDeny=true|false (ALPHA - default=false)
ServiceNodePortStaticSubrange=true|false (BETA - default=true)
SidecarContainers=true|false (ALPHA - default=false)
SizeMemoryBackedVolumes=true|false (BETA - default=true)
SkipReadOnlyValidationGCE=true|false (ALPHA - default=false)
StableLoadBalancerNodeSet=true|false (BETA - default=true)
StatefulSetAutoDeletePVC=true|false (BETA - default=true)
StatefulSetStartOrdinal=true|false (BETA - default=true)
StorageVersionAPI=true|false (ALPHA - default=false)
StorageVersionHash=true|false (BETA - default=true)
TopologyAwareHints=true|false (BETA - default=true)
TopologyManagerPolicyAlphaOptions=true|false (ALPHA - default=false)
TopologyManagerPolicyBetaOptions=true|false (BETA - default=true)
TopologyManagerPolicyOptions=true|false (BETA - default=true)
UnknownVersionInteroperabilityProxy=true|false (ALPHA - default=false)
UserNamespacesSupport=true|false (ALPHA - default=false)
ValidatingAdmissionPolicy=true|false (BETA - default=false)
VolumeCapacityPriority=true|false (ALPHA - default=false)
WatchList=true|false (ALPHA - default=false)
WinDSR=true|false (ALPHA - default=false)
WinOverlay=true|false (BETA - default=true)
WindowsHostNetwork=true|false (ALPHA - default=true)

--goaway-chance float

To prevent HTTP/2 clients from getting stuck on a single apiserver, randomly close a connection (GOAWAY). The client's other in-flight requests won't be affected, and the client will reconnect, likely landing on a different apiserver after going through the load balancer again. This argument sets the fraction of requests that will be sent a GOAWAY. Clusters with single apiservers, or which don't use a load balancer, should NOT enable this. Min is 0 (off), Max is .02 (1/50 requests); .001 (1/1000) is a recommended starting point.

-h, --help

help for kube-apiserver

--http2-max-streams-per-connection int

The limit that the server gives to clients for the maximum number of streams in an HTTP/2 connection. Zero means to use golang's default.
--kubelet-certificate-authority string
Path to a cert file for the certificate authority.
--kubelet-client-certificate string
Path to a client cert file for TLS.
--kubelet-client-key string
Path to a client key file for TLS.
--kubelet-preferred-address-types strings Default: "Hostname,InternalDNS,InternalIP,ExternalDNS,ExternalIP"
List of the preferred NodeAddressTypes to use for kubelet connections.
--kubelet-timeout duration Default: 5s
Timeout for kubelet operations.
--kubernetes-service-node-port int
If non-zero, the Kubernetes master service (which apiserver creates/maintains) will be of type NodePort, using this as the value of the port. If zero, the Kubernetes master service will be of type ClusterIP.
--lease-reuse-duration-seconds int Default: 60
The time in seconds that each lease is reused. A lower value could avoid large number of objects reusing the same lease. Notice that a too small value may cause performance problems at storage layer.
--livez-grace-period duration
This option represents the maximum amount of time it should take for apiserver to complete its startup sequence and become live. From apiserver's start time to when this amount of time has elapsed, /livez will assume that unfinished post-start hooks will complete successfully and therefore return true.
--log-flush-frequency duration Default: 5s
Maximum number of seconds between log flushes
--logging-format string Default: "text"
Sets the log format. Permitted formats: "text".

--max-connection-bytes-per-sec int	If non-zero, throttle each user connection to this number of bytes/sec. Currently only applies to long-running requests.
--max-mutating-requests-inflight int Default: 200	This and --max-requests-inflight are summed to determine the server's total concurrency limit (which must be positive) if --enable-priority-and-fairness is true. Otherwise, this flag limits the maximum number of mutating requests in flight, or a zero value disables the limit completely.
--max-requests-inflight int Default: 400	This and --max-mutating-requests-inflight are summed to determine the server's total concurrency limit (which must be positive) if --enable-priority-and-fairness is true. Otherwise, this flag limits the maximum number of non-mutating requests in flight, or a zero value disables the limit completely.
--min-request-timeout int Default: 1800	An optional field indicating the minimum number of seconds a handler must keep a request open before timing it out. Currently only honored by the watch request handler, which picks a randomized value above this number as the connection timeout, to spread out load.
--oidc-ca-file string	If set, the OpenID server's certificate will be verified by one of the authorities in the oidc-ca-file, otherwise the host's root CA set will be used.
--oidc-client-id string	The client ID for the OpenID Connect client, must be set if oidc-issuer-url is set.
--oidc-groups-claim string	If provided, the name of a custom OpenID Connect claim for specifying user groups. The claim value is expected to be a string or array of strings. This flag is experimental, please see the authentication documentation for further details.
--oidc-groups-prefix string	If provided, all groups will be prefixed with this value to prevent conflicts with other authentication strategies.
--oidc-issuer-url string	The URL of the OpenID issuer, only HTTPS scheme will be accepted. If set, it will be used to verify the OIDC JSON Web Token (JWT).
--oidc-required-claim <comma-separated 'key=value' pairs>	

A key=value pair that describes a required claim in the ID Token. If set, the claim is verified to be present in the ID Token with a matching value. Repeat this flag to specify multiple claims.

--oidc-signing-algs strings Default: "RS256"

Comma-separated list of allowed JOSE asymmetric signing algorithms. JWTs with a supported 'alg' header values are: RS256, RS384, RS512, ES256, ES384, ES512, PS256, PS384, PS512. Values are defined by RFC 7518 <https://tools.ietf.org/html/rfc7518#section-3.1>.

--oidc-username-claim string Default: "sub"

The OpenID claim to use as the user name. Note that claims other than the default ('sub') is not guaranteed to be unique and immutable. This flag is experimental, please see the authentication documentation for further details.

--oidc-username-prefix string

If provided, all usernames will be prefixed with this value. If not provided, username claims other than 'email' are prefixed by the issuer URL to avoid clashes. To skip any prefixing, provide the value '-'.

--peer-advertise-ip string

If set and the UnknownVersionInteroperabilityProxy feature gate is enabled, this IP will be used by peer kube-apiservers to proxy requests to this kube-apiserver when the request cannot be handled by the peer due to version skew between the kube-apiservers. This flag is only used in clusters configured with multiple kube-apiservers for high availability.

--peer-advertise-port string

If set and the UnknownVersionInteroperabilityProxy feature gate is enabled, this port will be used by peer kube-apiservers to proxy requests to this kube-apiserver when the request cannot be handled by the peer due to version skew between the kube-apiservers. This flag is only used in clusters configured with multiple kube-apiservers for high availability.

--peer-ca-file string

If set and the UnknownVersionInteroperabilityProxy feature gate is enabled, this file will be used to verify serving certificates of peer kube-apiservers. This flag is only used in clusters configured with multiple kube-apiservers for high availability.

--permit-address-sharing

If true, SO_REUSEADDR will be used when binding the port. This allows binding to wildcard IPs like 0.0.0.0 and specific IPs in parallel, and it avoids waiting for the kernel to release sockets in TIME_WAIT state. [default=false]

--permit-port-sharing

If true, SO_REUSEPORT will be used when binding the port, which allows more than one instance to bind on the same address and port. [default=false]
--profiling Default: true
Enable profiling via web interface host:port/debug/pprof/
--proxy-client-cert-file string
Client certificate used to prove the identity of the aggregator or kube-apiserver when it must call out during a request. This includes proxying requests to a user api-server and calling out to webhook admission plugins. It is expected that this cert includes a signature from the CA in the --requestheader-client-ca-file flag. That CA is published in the 'extension-apiserver-authentication' configmap in the kube-system namespace. Components receiving calls from kube-aggregator should use that CA to perform their half of the mutual TLS verification.
--proxy-client-key-file string
Private key for the client certificate used to prove the identity of the aggregator or kube-apiserver when it must call out during a request. This includes proxying requests to a user api-server and calling out to webhook admission plugins.
--request-timeout duration Default: 1m0s
An optional field indicating the duration a handler must keep a request open before timing it out. This is the default request timeout for requests but may be overridden by flags such as --min-request-timeout for specific types of requests.
--requestheader-allowed-names strings
List of client certificate common names to allow to provide usernames in headers specified by --requestheader-username-headers. If empty, any client certificate validated by the authorities in --requestheader-client-ca-file is allowed.
--requestheader-client-ca-file string
Root certificate bundle to use to verify client certificates on incoming requests before trusting usernames in headers specified by --requestheader-username-headers. WARNING: generally do not depend on authorization being already done for incoming requests.
--requestheader-extra-headers-prefix strings
List of request header prefixes to inspect. X-Remote-Extra- is suggested.
--requestheader-group-headers strings
List of request headers to inspect for groups. X-Remote-Group is suggested.
--requestheader-username-headers strings

List of request headers to inspect for usernames. X-Remote-User is common.

--runtime-config <comma-separated 'key=value' pairs>

A set of key=value pairs that enable or disable built-in APIs. Supported options are:
v1=true|false for the core API group
<group>/<version>=true|false for a specific API group and version (e.g. apps/v1=true)
api/all=true|false controls all API versions
api/ga=true|false controls all API versions of the form v[0-9]+
api/beta=true|false controls all API versions of the form v[0-9]+beta[0-9]+
api/alpha=true|false controls all API versions of the form v[0-9]+alpha[0-9]+
api/legacy is deprecated, and will be removed in a future version

--secure-port int Default: 6443

The port on which to serve HTTPS with authentication and authorization. It cannot be switched off with 0.

--service-account-extend-token-expiration Default: true

Turns on projected service account expiration extension during token generation, which helps safe transition from legacy token to bound service account token feature. If this flag is enabled, admission injected tokens would be extended up to 1 year to prevent unexpected failure during transition, ignoring value of service-account-max-token-expiration.

--service-account-issuer strings

Identifier of the service account token issuer. The issuer will assert this identifier in "iss" claim of issued tokens. This value is a string or URI. If this option is not a valid URI per the OpenID Discovery 1.0 spec, the ServiceAccountIssuerDiscovery feature will remain disabled, even if the feature gate is set to true. It is highly recommended that this value comply with the OpenID spec: https://openid.net/specs/openid-connect-discovery-1_0.html. In practice, this means that service-account-issuer must be an https URL. It is also highly recommended that this URL be capable of serving OpenID discovery documents at {service-account-issuer}/.well-known/openid-configuration. When this flag is specified multiple times, the first is used to generate tokens and all are used to determine which issuers are accepted.

--service-account-jwks-uri string

Overrides the URI for the JSON Web Key Set in the discovery doc served at /.well-known/openid-configuration. This flag is useful if the discovery doc and key set are served to relying parties from a URL other than the API server's external (as auto-detected or overridden with external-hostname).

--service-account-key-file strings

File containing PEM-encoded x509 RSA or ECDSA private or public keys, used to verify ServiceAccount tokens. The specified file can contain multiple keys, and the flag can be specified multiple times with different files. If unspecified, --tls-private-key-file is used. Must be specified when --service-account-signing-key-file is provided

--service-account-lookup	Default: true
If true, validate ServiceAccount tokens exist in etcd as part of authentication.	
--service-account-max-token-expiration	duration
The maximum validity duration of a token created by the service account token issuer. If an otherwise valid TokenRequest with a validity duration larger than this value is requested, a token will be issued with a validity duration of this value.	
--service-account-signing-key-file	string
Path to the file that contains the current private key of the service account token issuer. The issuer will sign issued ID tokens with this private key.	
--service-cluster-ip-range	string
A CIDR notation IP range from which to assign service cluster IPs. This must not overlap with any IP ranges assigned to nodes or pods. Max of two dual-stack CIDRs is allowed.	
--service-node-port-range	<a string in the form 'N1-N2'> Default: 30000-32767
A port range to reserve for services with NodePort visibility. This must not overlap with the ephemeral port range on nodes. Example: '30000-32767'. Inclusive at both ends of the range.	
--show-hidden-metrics-for-version	string
The previous version for which you want to show hidden metrics. Only the previous minor version is meaningful, other values will not be allowed. The format is <major>.<minor>, e.g.: '1.16'. The purpose of this format is make sure you have the opportunity to notice if the next release hides additional metrics, rather than being surprised when they are permanently removed in the release after that.	
--shutdown-delay-duration	duration
Time to delay the termination. During that time the server keeps serving requests normally. The endpoints /healthz and /livez will return success, but /readyz immediately returns failure. Graceful termination starts after this delay has elapsed. This can be used to allow load balancer to stop sending traffic to this server.	
--shutdown-send-retry-after	
If true the HTTP Server will continue listening until all non long running request(s) in flight have been drained, during this window all incoming requests will be rejected with a status code 429 and a 'Retry-After' response header, in addition 'Connection: close' response header is set in order to tear down the TCP connection when idle.	
--shutdown-watch-termination-grace-period	duration
This option, if set, represents the maximum amount of grace period the apiserver will wait for active watch request(s) to drain during the graceful server shutdown window.	

--storage-backend string	The storage backend for persistence. Options: 'etcd3' (default).
--storage-media-type string Default: "application/vnd.kubernetes.protobuf"	The media type to use to store objects in storage. Some resources or storage backends may only support a specific media type and will ignore this setting. Supported media types: [application/json, application/yaml, application/vnd.kubernetes.protobuf]
--strict-transport-security-directives strings	List of directives for HSTS, comma separated. If this list is empty, then HSTS directives will not be added. Example: 'max-age=31536000,includeSubDomains,preload'
--tls-cert-file string	File containing the default x509 Certificate for HTTPS. (CA cert, if any, concatenated after server cert). If HTTPS serving is enabled, and --tls-cert-file and --tls-private-key-file are not provided, a self-signed certificate and key are generated for the public address and saved to the directory specified by --cert-dir.
--tls-cipher-suites strings	<p>Comma-separated list of cipher suites for the server. If omitted, the default Go cipher suites will be used.</p> <p>Preferred values: TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_GCM_SHA384.</p> <p>Insecure values: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_RC4_128_SHA.</p>
--tls-min-version string	

Minimum TLS version supported. Possible values: VersionTLS10, VersionTLS11, VersionTLS12, VersionTLS13
--tls-private-key-file string
File containing the default x509 private key matching --tls-cert-file.
--tls-sni-cert-key string
A pair of x509 certificate and private key file paths, optionally suffixed with a list of domain patterns which are fully qualified domain names, possibly with prefixed wildcard segments. The domain patterns also allow IP addresses, but IPs should only be used if the apiserver has visibility to the IP address requested by a client. If no domain patterns are provided, the names of the certificate are extracted. Non-wildcard matches trump over wildcard matches, explicit domain patterns trump over extracted names. For multiple key/certificate pairs, use the --tls-sni-cert-key multiple times. Examples: "example.crt,example.key" or "foo.crt,foo.key:*.foo.com,foo.com".
--token-auth-file string
If set, the file that will be used to secure the secure port of the API server via token authentication.
--tracing-config-file string
File with apiserver tracing configuration.
-v, --v int
number for the log level verbosity
--version version[=true]
--version, --version=raw prints version information and quits; --version=vX.Y.Z... sets the reported version
--vmodule pattern=N,...
comma-separated list of pattern=N settings for file-filtered logging (only works for text log format)
--watch-cache Default: true
Enable watch caching in the apiserver
--watch-cache-sizes strings
Watch cache size settings for some resources (pods, nodes, etc.), comma separated. The individual setting format: resource[.group]#size, where resource is lowercase plural (no version), group is omitted for resources of apiVersion v1 (the legacy core API) and included for others, and size is a number. This option is only meaningful for resources built into the

apiserver, not ones defined by CRDs or aggregated from external servers, and is only consulted if the watch-cache is enabled. The only meaningful size setting to supply here is zero, which means to disable watch caching for the associated resource; all non-zero values are equivalent and mean to not disable watch caching for that resource

kube-controller-manager

Synopsis

The Kubernetes controller manager is a daemon that embeds the core control loops shipped with Kubernetes. In applications of robotics and automation, a control loop is a non-terminating loop that regulates the state of the system. In Kubernetes, a controller is a control loop that watches the shared state of the cluster through the apiserver and makes changes attempting to move the current state towards the desired state. Examples of controllers that ship with Kubernetes today are the replication controller, endpoints controller, namespace controller, and serviceaccounts controller.

kube-controller-manager [flags]

Options

--allocate-node-cidrs	Should CIDRs for Pods be allocated and set on the cloud provider.
--allow-metric-labels stringToString Default: []	The map from metric-label to value allow-list of this label. The key's format is <MetricName>,<LabelName>. The value's format is <allowed_value>,<allowed_value>...e.g. metric1,label1='v1,v2,v3', metric1,label2='v1,v2,v3' metric2,label1='v1,v2,v3'.
--attach-detach-reconcile-sync-period duration Default: 1m0s	The reconciler sync wait time between volume attach detach. This duration must be larger than one second, and increasing this value from the default may allow for volumes to be mismatched with pods.
--authentication-kubeconfig string	kubeconfig file pointing at the 'core' kubernetes server with enough rights to create tokenreviews.authentication.k8s.io. This is optional. If empty, all token requests are considered to be anonymous and no client CA is looked up in the cluster.
--authentication-skip-lookup	If false, the authentication-kubeconfig will be used to lookup missing authentication configuration from the cluster.
--authentication-token-webhook-cache-ttl duration Default: 10s	

	The duration to cache responses from the webhook token authenticator.
--authentication-tolerate-lookup-failure	If true, failures to look up missing authentication configuration from the cluster are not considered fatal. Note that this can result in authentication that treats all requests as anonymous.
--authorization-always-allow-paths	strings Default: "/healthz,/readyz,/livez"
	A list of HTTP paths to skip during authorization, i.e. these are authorized without contacting the 'core' kubernetes server.
--authorization-kubeconfig	string
	kubeconfig file pointing at the 'core' kubernetes server with enough rights to create subjectaccessreviews.authorization.k8s.io. This is optional. If empty, all requests not skipped by authorization are forbidden.
--authorization-webhook-cache-authorized-ttl	duration Default: 10s
	The duration to cache 'authorized' responses from the webhook authorizer.
--authorization-webhook-cache-unauthorized-ttl	duration Default: 10s
	The duration to cache 'unauthorized' responses from the webhook authorizer.
--azure-container-registry-config	string
	Path to the file containing Azure container registry configuration information.
--bind-address	string Default: 0.0.0.0
	The IP address on which to listen for the --secure-port port. The associated interface(s) must be reachable by the rest of the cluster, and by CLI/web clients. If blank or an unspecified address (0.0.0.0 or ::), all interfaces and IP address families will be used.
--cert-dir	string
	The directory where the TLS certs are located. If --tls-cert-file and --tls-private-key-file are provided, this flag will be ignored.
--cidr-allocator-type	string Default: "RangeAllocator"
	Type of CIDR allocator to use
--client-ca-file	string

If set, any request presenting a client certificate signed by one of the authorities in the client-ca-file is authenticated with an identity corresponding to the CommonName of the client certificate.
--cloud-config string
The path to the cloud provider configuration file. Empty string for no configuration file.
--cloud-provider string
The provider for cloud services. Empty string for no provider.
--cluster-cidr string
CIDR Range for Pods in cluster. Requires --allocate-node-cidrs to be true
--cluster-name string Default: "kubernetes"
The instance prefix for the cluster.
--cluster-signing-cert-file string
Filename containing a PEM-encoded X509 CA certificate used to issue cluster-scoped certificates. If specified, no more specific --cluster-signing-* flag may be specified.
--cluster-signing-duration duration Default: 8760h0m0s
The max length of duration signed certificates will be given. Individual CSRs may request shorter certs by setting spec.expirationSeconds.
--cluster-signing-key-file string
Filename containing a PEM-encoded RSA or ECDSA private key used to sign cluster-scoped certificates. If specified, no more specific --cluster-signing-* flag may be specified.
--cluster-signing-kube-apiserver-client-cert-file string
Filename containing a PEM-encoded X509 CA certificate used to issue certificates for the kubernetes.io/kube-apiserver-client signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--cluster-signing-kube-apiserver-client-key-file string
Filename containing a PEM-encoded RSA or ECDSA private key used to sign certificates for the kubernetes.io/kube-apiserver-client signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--cluster-signing-kubelet-client-cert-file string

	Filename containing a PEM-encoded X509 CA certificate used to issue certificates for the kubernetes.io/kube-apiserver-client-kubelet signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--cluster-signing-kubelet-client-key-file	string
	Filename containing a PEM-encoded RSA or ECDSA private key used to sign certificates for the kubernetes.io/kube-apiserver-client-kubelet signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--cluster-signing-kubelet-serving-cert-file	string
	Filename containing a PEM-encoded X509 CA certificate used to issue certificates for the kubernetes.io/kubelet-serving signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--cluster-signing-kubelet-serving-key-file	string
	Filename containing a PEM-encoded RSA or ECDSA private key used to sign certificates for the kubernetes.io/kubelet-serving signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--cluster-signing-legacy-unknown-cert-file	string
	Filename containing a PEM-encoded X509 CA certificate used to issue certificates for the kubernetes.io/legacy-unknown signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--cluster-signing-legacy-unknown-key-file	string
	Filename containing a PEM-encoded RSA or ECDSA private key used to sign certificates for the kubernetes.io/legacy-unknown signer. If specified, --cluster-signing-{cert,key}-file must not be set.
--concurrent-cron-job-syncs	int32 Default: 5
	The number of cron job objects that are allowed to sync concurrently. Larger number = more responsive jobs, but more CPU (and network) load
--concurrent-deployment-syncs	int32 Default: 5
	The number of deployment objects that are allowed to sync concurrently. Larger number = more responsive deployments, but more CPU (and network) load
--concurrent-endpoint-syncs	int32 Default: 5
	The number of endpoint syncing operations that will be done concurrently. Larger number = faster endpoint updating, but more CPU (and network) load
--concurrent-ephemeralvolume-syncs	int32 Default: 5

	The number of ephemeral volume syncing operations that will be done concurrently. Larger number = faster ephemeral volume updating, but more CPU (and network) load
--concurrent-gc-syncs int32	Default: 20
	The number of garbage collector workers that are allowed to sync concurrently.
--concurrent-horizontal-pod-autoscaler-syncs int32	Default: 5
	The number of horizontal pod autoscaler objects that are allowed to sync concurrently. Larger number = more responsive horizontal pod autoscaler objects processing, but more CPU (and network) load.
--concurrent-job-syncs int32	Default: 5
	The number of job objects that are allowed to sync concurrently. Larger number = more responsive jobs, but more CPU (and network) load
--concurrent-namespace-syncs int32	Default: 10
	The number of namespace objects that are allowed to sync concurrently. Larger number = more responsive namespace termination, but more CPU (and network) load
--concurrent-rc-syncs int32	Default: 5
	The number of replication controllers that are allowed to sync concurrently. Larger number = more responsive replica management, but more CPU (and network) load
--concurrent-replicaset-syncs int32	Default: 5
	The number of replica sets that are allowed to sync concurrently. Larger number = more responsive replica management, but more CPU (and network) load
--concurrent-resource-quota-syncs int32	Default: 5
	The number of resource quotas that are allowed to sync concurrently. Larger number = more responsive quota management, but more CPU (and network) load
--concurrent-service-endpoint-syncs int32	Default: 5
	The number of service endpoint syncing operations that will be done concurrently. Larger number = faster endpoint slice updating, but more CPU (and network) load. Defaults to 5.
--concurrent-service-syncs int32	Default: 1
	The number of services that are allowed to sync concurrently. Larger number = more responsive service management, but more CPU (and network) load
--concurrent-serviceaccount-token-syncs int32	Default: 5

	The number of service account token objects that are allowed to sync concurrently. Larger number = more responsive token generation, but more CPU (and network) load
--concurrent-statefulset-syncs int32	Default: 5
	The number of statefulset objects that are allowed to sync concurrently. Larger number = more responsive statefulsets, but more CPU (and network) load
--concurrent-ttl-after-finished-syncs int32	Default: 5
	The number of ttl-after-finished-controller workers that are allowed to sync concurrently.
--concurrent-validating-admission-policy-status-syncs int32	Default: 5
	The number of ValidatingAdmissionPolicyStatusController workers that are allowed to sync concurrently.
--configure-cloud-routes	Default: true
	Should CIDRs allocated by allocate-node-cidrs be configured on the cloud provider.
--contention-profiling	
	Enable block profiling, if profiling is enabled
--controller-start-interval duration	
	Interval between starting controller managers.
--controllers strings	Default: "*" A list of controllers to enable. '*' enables all on-by-default controllers, 'foo' enables the controller named 'foo', '-foo' disables the controller named 'foo'. All controllers: bootstrap-signer-controller, certificatesigningrequest-approving-controller, certificatesigningrequest-cleaner-controller, certificatesigningrequest-signing-controller, cloud-node-lifecycle-controller, clusterrole-aggregation-controller, cronjob-controller, daemonset-controller, deployment-controller, disruption-controller, endpoints-controller, endpointslice-controller, endpointslice-mirroring-controller, ephemeral-volume-controller, garbage-collector-controller, horizontal-pod-autoscaler-controller, job-controller, namespace-controller, node-ipam-controller, node-lifecycle-controller, node-route-controller, persistentvolume-attach-detach-controller, persistentvolume-binder-controller, persistentvolume-expander-controller, persistentvolume-protection-controller, persistentvolumeclaim-protection-controller, pod-garbage-collector-controller, replicaset-controller, replicationcontroller-controller, resourcequota-controller, root-ca-certificate-publisher-controller, service-lb-controller, serviceaccount-controller, serviceaccount-token-controller, statefulset-controller, token-cleaner-controller, ttl-after-finished-controller, ttl-controller Disabled-by-default controllers: bootstrap-signer-controller, token-cleaner-controller
--disable-attach-detach-reconcile-sync	

	Disable volume attach/detach reconciler sync. Disabling this may cause volumes to be mismatched with pods. Use wisely.
--disabled-metrics	strings
	This flag provides an escape hatch for misbehaving metrics. You must provide the fully qualified metric name in order to disable it. Disclaimer: disabling metrics is higher in precedence than showing hidden metrics.
--enable-dynamic-provisioning	Default: true
	Enable dynamic provisioning for environments that support it.
--enable-garbage-collector	Default: true
	Enables the generic garbage collector. MUST be synced with the corresponding flag of the kube-apiserver.
--enable-hostpath-provisioner	
	Enable HostPath PV provisioning when running without a cloud provider. This allows testing and development of provisioning features. HostPath provisioning is not supported in any way, won't work in a multi-node cluster, and should not be used for anything other than testing or development.
--enable-leader-migration	
	Whether to enable controller leader migration.
--endpoint-updates-batch-period	duration
	The length of endpoint updates batching period. Processing of pod changes will be delayed by this duration to join them with potential upcoming updates and reduce the overall number of endpoints updates. Larger number = higher endpoint programming latency, but lower number of endpoints revision generated
--endpointslice-updates-batch-period	duration
	The length of endpoint slice updates batching period. Processing of pod changes will be delayed by this duration to join them with potential upcoming updates and reduce the overall number of endpoints updates. Larger number = higher endpoint programming latency, but lower number of endpoints revision generated
--external-cloud-volume-plugin	string
	The plugin to use when cloud provider is set to external. Can be empty, should only be set when cloud-provider is external. Currently used to allow node-ipam-controller, persistentvolume-binder-controller, persistentvolume-expander-controller and attach-detach-controller to work for in-tree cloud providers.
--feature-gates	<comma-separated 'key=True False' pairs>

A set of key=value pairs that describe feature gates for alpha/experimental features. Options are:

APIListChunking=true|false (BETA - default=true)
APIPriorityAndFairness=true|false (BETA - default=true)
APIResponseCompression=true|false (BETA - default=true)
APIServerIdentity=true|false (BETA - default=true)
APIServerTracing=true|false (BETA - default=true)
AdmissionWebhookMatchConditions=true|false (BETA - default=true)
AggregatedDiscoveryEndpoint=true|false (BETA - default=true)
AllAlpha=true|false (ALPHA - default=false)
AllBeta=true|false (BETA - default=false)
AnyVolumeDataSource=true|false (BETA - default=true)
AppArmor=true|false (BETA - default=true)
CPUManagerPolicyAlphaOptions=true|false (ALPHA - default=false)
CPUManagerPolicyBetaOptions=true|false (BETA - default=true)
CPUManagerPolicyOptions=true|false (BETA - default=true)
CRDValidationRatcheting=true|false (ALPHA - default=false)
CSIMigrationPortworx=true|false (BETA - default=false)
CSINodeExpandSecret=true|false (BETA - default=true)
CSIVolumeHealth=true|false (ALPHA - default=false)
CloudControllerManagerWebhook=true|false (ALPHA - default=false)
CloudDualStackNodeIPs=true|false (ALPHA - default=false)
ClusterTrustBundle=true|false (ALPHA - default=false)
ComponentSLIs=true|false (BETA - default=true)
ConsistentListFromCache=true|false (ALPHA - default=false)
ContainerCheckpoint=true|false (ALPHA - default=false)
ContextualLogging=true|false (ALPHA - default=false)
CronJobsScheduledAnnotation=true|false (BETA - default=true)
CrossNamespaceVolumeDataSource=true|false (ALPHA - default=false)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - default=false)
CustomResourceValidationExpressions=true|false (BETA - default=true)
DevicePluginCDIDevices=true|false (ALPHA - default=false)
DisableCloudProviders=true|false (ALPHA - default=false)
DisableKubeletCloudCredentialProviders=true|false (ALPHA - default=false)
DynamicResourceAllocation=true|false (ALPHA - default=false)
ElasticIndexedJob=true|false (BETA - default=true)
EventedPLEG=true|false (BETA - default=false)
GracefulNodeShutdown=true|false (BETA - default=true)
GracefulNodeShutdownBasedOnPodPriority=true|false (BETA - default=true)
HPAContainerMetrics=true|false (BETA - default=true)
HPAScaleToZero=true|false (ALPHA - default=false)
HonorPVReclaimPolicy=true|false (ALPHA - default=false)
InPlacePodVerticalScaling=true|false (ALPHA - default=false)
InTreePluginAWSUnregister=true|false (ALPHA - default=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - default=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - default=false)
InTreePluginGCEUnregister=true|false (ALPHA - default=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - default=false)
InTreePluginPortworxUnregister=true|false (ALPHA - default=false)
InTreePluginvSphereUnregister=true|false (ALPHA - default=false)
JobBackoffLimitPerIndex=true|false (ALPHA - default=false)
JobPodFailurePolicy=true|false (BETA - default=true)

JobPodReplacementPolicy=true|false (ALPHA - default=false)
JobReadyPods=true|false (BETA - default=true)
KMSv2=true|false (BETA - default=true)
KMSv2KDF=true|false (BETA - default=false)
KubeProxyDrainingTerminatingNodes=true|false (ALPHA - default=false)
KubeletCgroupDriverFromCRI=true|false (ALPHA - default=false)
KubeletInUserNamespace=true|false (ALPHA - default=false)
KubeletPodResourcesDynamicResources=true|false (ALPHA - default=false)
KubeletPodResourcesGet=true|false (ALPHA - default=false)
KubeletTracing=true|false (BETA - default=true)
LegacyServiceAccountTokenCleanUp=true|false (ALPHA - default=false)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - default=false)
LogarithmicScaleDown=true|false (BETA - default=true)
LoggingAlphaOptions=true|false (ALPHA - default=false)
LoggingBetaOptions=true|false (BETA - default=true)
MatchLabelKeysInPodTopologySpread=true|false (BETA - default=true)
MaxUnavailableStatefulSet=true|false (ALPHA - default=false)
MemoryManager=true|false (BETA - default=true)
MemoryQoS=true|false (ALPHA - default=false)
MinDomainsInPodTopologySpread=true|false (BETA - default=true)
MultiCIDRRangeAllocator=true|false (ALPHA - default=false)
MultiCIDRSvcAllocator=true|false (ALPHA - default=false)
NewVolumeManagerReconstruction=true|false (BETA - default=true)
NodeInclusionPolicyInPodTopologySpread=true|false (BETA - default=true)
NodeLogQuery=true|false (ALPHA - default=false)
NodeSwap=true|false (BETA - default=false)
OpenAPIEnums=true|false (BETA - default=true)
PDBUnhealthyPodEvictionPolicy=true|false (BETA - default=true)
PersistentVolumeLastPhaseTransitionTime=true|false (ALPHA - default=false)
PodAndContainerStatsFromCRI=true|false (ALPHA - default=false)
PodDeletionCost=true|false (BETA - default=true)
PodDisruptionConditions=true|false (BETA - default=true)
PodHostIPs=true|false (ALPHA - default=false)
PodIndexLabel=true|false (BETA - default=true)
PodReadyToStartContainersCondition=true|false (ALPHA - default=false)
PodSchedulingReadiness=true|false (BETA - default=true)
ProcMountType=true|false (ALPHA - default=false)
QOSReserved=true|false (ALPHA - default=false)
ReadWriteOncePod=true|false (BETA - default=true)
RecoverVolumeExpansionFailure=true|false (ALPHA - default=false)
RemainingItemCount=true|false (BETA - default=true)
RotateKubeletServerCertificate=true|false (BETA - default=true)
SELinuxMountReadWriteOncePod=true|false (BETA - default=true)
SchedulerQueueingHints=true|false (BETA - default=true)
SecurityContextDeny=true|false (ALPHA - default=false)
ServiceNodePortStaticSubrange=true|false (BETA - default=true)
SidecarContainers=true|false (ALPHA - default=false)
SizeMemoryBackedVolumes=true|false (BETA - default=true)
SkipReadOnlyValidationGCE=true|false (ALPHA - default=false)
StableLoadBalancerNodeSet=true|false (BETA - default=true)
StatefulSetAutoDeletePVC=true|false (BETA - default=true)
StatefulSetStartOrdinal=true|false (BETA - default=true)
StorageVersionAPI=true|false (ALPHA - default=false)

	StorageVersionHash=true false (BETA - default=true) TopologyAwareHints=true false (BETA - default=true) TopologyManagerPolicyAlphaOptions=true false (ALPHA - default=false) TopologyManagerPolicyBetaOptions=true false (BETA - default=true) TopologyManagerPolicyOptions=true false (BETA - default=true) UnknownVersionInteroperabilityProxy=true false (ALPHA - default=false) UserNamespacesSupport=true false (ALPHA - default=false) ValidatingAdmissionPolicy=true false (BETA - default=false) VolumeCapacityPriority=true false (ALPHA - default=false) WatchList=true false (ALPHA - default=false) WinDSR=true false (ALPHA - default=false) WinOverlay=true false (BETA - default=true) WindowsHostNetwork=true false (ALPHA - default=true)
--flex-volume-plugin-dir	string Default: "/usr/libexec/kubernetes/kubelet-plugins/volume/exec/"
	Full path of the directory in which the flex volume plugin should search for additional third party volume plugins.
-h, --help	
	help for kube-controller-manager
--horizontal-pod-autoscaler-cpu-initialization-period	duration Default: 5m0s
	The period after pod start when CPU samples might be skipped.
--horizontal-pod-autoscaler-downscale-stabilization	duration Default: 5m0s
	The period for which autoscaler will look backwards and not scale down below any recommendation it made during that period.
--horizontal-pod-autoscaler-initial-readiness-delay	duration Default: 30s
	The period after pod start during which readiness changes will be treated as initial readiness.
--horizontal-pod-autoscaler-sync-period	duration Default: 15s
	The period for syncing the number of pods in horizontal pod autoscaler.
--horizontal-pod-autoscaler-tolerance	float Default: 0.1
	The minimum change (from 1.0) in the desired-to-actual metrics ratio for the horizontal pod autoscaler to consider scaling.
--http2-max-streams-per-connection	int
	The limit that the server gives to clients for the maximum number of streams in an HTTP/2 connection. Zero means to use golang's default.
--kube-api-burst	int32 Default: 30

Burst to use while talking with kubernetes apiserver.
--kube-api-content-type string Default: "application/vnd.kubernetes.protobuf"
Content type of requests sent to apiserver.
--kube-api-qps float Default: 20
QPS to use while talking with kubernetes apiserver.
--kubeconfig string
Path to kubeconfig file with authorization and master location information (the master location can be overridden by the master flag).
--large-cluster-size-threshold int32 Default: 50
Number of nodes from which node-lifecycle-controller treats the cluster as large for the eviction logic purposes. --secondary-node-eviction-rate is implicitly overridden to 0 for clusters this size or smaller.
--leader-elect Default: true
Start a leader election client and gain leadership before executing the main loop. Enable this when running replicated components for high availability.
--leader-elect-lease-duration duration Default: 15s
The duration that non-leader candidates will wait after observing a leadership renewal until attempting to acquire leadership of a led but unrenewed leader slot. This is effectively the maximum duration that a leader can be stopped before it is replaced by another candidate. This is only applicable if leader election is enabled.
--leader-elect-renew-deadline duration Default: 10s
The interval between attempts by the acting master to renew a leadership slot before it stops leading. This must be less than the lease duration. This is only applicable if leader election is enabled.
--leader-elect-resource-lock string Default: "leases"
The type of resource object that is used for locking during leader election. Supported options are 'leases', 'endpointsleases' and 'configmapsleases'.
--leader-elect-resource-name string Default: "kube-controller-manager"
The name of resource object that is used for locking during leader election.
--leader-elect-resource-namespace string Default: "kube-system"

	The namespace of resource object that is used for locking during leader election.
--leader-elect-retry-period duration	Default: 2s
	The duration the clients should wait between attempting acquisition and renewal of a leadership. This is only applicable if leader election is enabled.
--leader-migration-config string	
	Path to the config file for controller leader migration, or empty to use the value that reflects default configuration of the controller manager. The config file should be of type LeaderMigrationConfiguration, group controllermanager.config.k8s.io, version v1alpha1.
--legacy-service-account-token-clean-up-period duration	Default: 8760h0m0s
	The period of time since the last usage of an legacy service account token before it can be deleted.
--log-flush-frequency duration	Default: 5s
	Maximum number of seconds between log flushes
--logging-format string	Default: "text"
	Sets the log format. Permitted formats: "text".
--master string	
	The address of the Kubernetes API server (overrides any value in kubeconfig).
--max-endpoints-per-slice int32	Default: 100
	The maximum number of endpoints that will be added to an EndpointSlice. More endpoints per slice will result in less endpoint slices, but larger resources. Defaults to 100.
--min-resync-period duration	Default: 12h0m0s
	The resync period in reflectors will be random between MinResyncPeriod and 2*MinResyncPeriod.
--mirroring-concurrent-service-endpoint-syncs int32	Default: 5
	The number of service endpoint syncing operations that will be done concurrently by the endpointslice-mirroring-controller. Larger number = faster endpoint slice updating, but more CPU (and network) load. Defaults to 5.
--mirroring-endpointslice-updates-batch-period duration	
	The length of EndpointSlice updates batching period for endpointslice-mirroring-controller. Processing of EndpointSlice changes will be delayed by this duration to join them with potential upcoming updates and reduce the overall number of EndpointSlice updates. Larger

number = higher endpoint programming latency, but lower number of endpoints revision generated

--mirroring-max-endpoints-per-subset int32 Default: 1000

The maximum number of endpoints that will be added to an EndpointSlice by the endpointslice-mirroring-controller. More endpoints per slice will result in less endpoint slices, but larger resources. Defaults to 100.

--namespace-sync-period duration Default: 5m0s

The period for syncing namespace life-cycle updates

--node-cidr-mask-size int32

Mask size for node cidr in cluster. Default is 24 for IPv4 and 64 for IPv6.

--node-cidr-mask-size-ipv4 int32

Mask size for IPv4 node cidr in dual-stack cluster. Default is 24.

--node-cidr-mask-size-ipv6 int32

Mask size for IPv6 node cidr in dual-stack cluster. Default is 64.

--node-eviction-rate float Default: 0.1

Number of nodes per second on which pods are deleted in case of node failure when a zone is healthy (see --unhealthy-zone-threshold for definition of healthy/unhealthy). Zone refers to entire cluster in non-multizone clusters.

--node-monitor-grace-period duration Default: 40s

Amount of time which we allow running Node to be unresponsive before marking it unhealthy. Must be N times more than kubelet's nodeStatusUpdateFrequency, where N means number of retries allowed for kubelet to post node status.

--node-monitor-period duration Default: 5s

The period for syncing NodeStatus in cloud-node-lifecycle-controller.

--node-startup-grace-period duration Default: 1m0s

Amount of time which we allow starting Node to be unresponsive before marking it unhealthy.

--permit-address-sharing

If true, SO_REUSEADDR will be used when binding the port. This allows binding to wildcard IPs like 0.0.0.0 and specific IPs in parallel, and it avoids waiting for the kernel to release sockets in TIME_WAIT state. [default=false]

--permit-port-sharing	If true, SO_REUSEPORT will be used when binding the port, which allows more than one instance to bind on the same address and port. [default=false]
--profiling Default: true	Enable profiling via web interface host:port/debug/pprof/
--pv-recycler-increment-timeout-nfs int32 Default: 30	the increment of time added per Gi to ActiveDeadlineSeconds for an NFS scrubber pod
--pv-recycler-minimum-timeout-hostpath int32 Default: 60	The minimum ActiveDeadlineSeconds to use for a HostPath Recycler pod. This is for development and testing only and will not work in a multi-node cluster.
--pv-recycler-minimum-timeout-nfs int32 Default: 300	The minimum ActiveDeadlineSeconds to use for an NFS Recycler pod
--pv-recycler-pod-template-filename string	The file path to a pod definition used as a template for HostPath persistent volume recycling. This is for development and testing only and will not work in a multi-node cluster.
--pv-recycler-pod-template-filename-nfs string	The file path to a pod definition used as a template for NFS persistent volume recycling
--pv-recycler-timeout-increment-hostpath int32 Default: 30	the increment of time added per Gi to ActiveDeadlineSeconds for a HostPath scrubber pod. This is for development and testing only and will not work in a multi-node cluster.
--pvclaimbinder-sync-period duration Default: 15s	The period for syncing persistent volumes and persistent volume claims
--requestheader-allowed-names strings	List of client certificate common names to allow to provide usernames in headers specified by --requestheader-username-headers. If empty, any client certificate validated by the authorities in --requestheader-client-ca-file is allowed.
--requestheader-client-ca-file string	Root certificate bundle to use to verify client certificates on incoming requests before trusting usernames in headers specified by --requestheader-username-headers. WARNING: generally do not depend on authorization being already done for incoming requests.

--requestheader-extra-headers-prefix	strings	Default: "x-remote-extra-"
List of request header prefixes to inspect. X-Remote-Extra- is suggested.		
--requestheader-group-headers	strings	Default: "x-remote-group"
List of request headers to inspect for groups. X-Remote-Group is suggested.		
--requestheader-username-headers	strings	Default: "x-remote-user"
List of request headers to inspect for usernames. X-Remote-User is common.		
--resource-quota-sync-period	duration	Default: 5m0s
The period for syncing quota usage status in the system		
--root-ca-file	string	
If set, this root certificate authority will be included in service account's token secret. This must be a valid PEM-encoded CA bundle.		
--route-reconciliation-period	duration	Default: 10s
The period for reconciling routes created for Nodes by cloud provider.		
--secondary-node-eviction-rate	float	Default: 0.01
Number of nodes per second on which pods are deleted in case of node failure when a zone is unhealthy (see --unhealthy-zone-threshold for definition of healthy/unhealthy). Zone refers to entire cluster in non-multizone clusters. This value is implicitly overridden to 0 if the cluster size is smaller than --large-cluster-size-threshold.		
--secure-port	int	Default: 10257
The port on which to serve HTTPS with authentication and authorization. If 0, don't serve HTTPS at all.		
--service-account-private-key-file	string	
Filename containing a PEM-encoded private RSA or ECDSA key used to sign service account tokens.		
--service-cluster-ip-range	string	
CIDR Range for Services in cluster. Requires --allocate-node-cidrs to be true		
--show-hidden-metrics-for-version	string	
The previous version for which you want to show hidden metrics. Only the previous minor version is meaningful, other values will not be allowed. The format is <major>.<minor>, e.g.: '1.16'. The purpose of this format is make sure you have the opportunity to notice if the next		

release hides additional metrics, rather than being surprised when they are permanently removed in the release after that.

--terminated-pod-gc-threshold int32 Default: 12500

Number of terminated pods that can exist before the terminated pod garbage collector starts deleting terminated pods. If <= 0, the terminated pod garbage collector is disabled.

--tls-cert-file string

File containing the default x509 Certificate for HTTPS. (CA cert, if any, concatenated after server cert). If HTTPS serving is enabled, and --tls-cert-file and --tls-private-key-file are not provided, a self-signed certificate and key are generated for the public address and saved to the directory specified by --cert-dir.

--tls-cipher-suites strings

Comma-separated list of cipher suites for the server. If omitted, the default Go cipher suites will be used.

Preferred values: TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_GCM_SHA384.

Insecure values: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_RC4_128_SHA.

--tls-min-version string

Minimum TLS version supported. Possible values: VersionTLS10, VersionTLS11, VersionTLS12, VersionTLS13

--tls-private-key-file string

File containing the default x509 private key matching --tls-cert-file.

--tls-sni-cert-key string

A pair of x509 certificate and private key file paths, optionally suffixed with a list of domain patterns which are fully qualified domain names, possibly with prefixed wildcard segments. The domain patterns also allow IP addresses, but IPs should only be used if the apiserver has visibility to the IP address requested by a client. If no domain patterns are provided, the names of the certificate are extracted. Non-wildcard matches trump over wildcard matches, explicit domain patterns trump over extracted names. For multiple key/certificate pairs, use the --tls-sni-cert-key multiple times. Examples: "example.crt,example.key" or "foo.crt,foo.key:*.foo.com,foo.com".

--unhealthy-zone-threshold float Default: 0.55

Fraction of Nodes in a zone which needs to be not Ready (minimum 3) for zone to be treated as unhealthy.

--use-service-account-credentials

If true, use individual service account credentials for each controller.

-v, --v int

number for the log level verbosity

--version version[=true]

--version, --version=raw prints version information and quits; --version=vX.Y.Z... sets the reported version

--vmodule pattern=N,...

comma-separated list of pattern=N settings for file-filtered logging (only works for text log format)

kube-proxy

Synopsis

The Kubernetes network proxy runs on each node. This reflects services as defined in the Kubernetes API on each node and can do simple TCP, UDP, and SCTP stream forwarding or round robin TCP, UDP, and SCTP forwarding across a set of backends. Service cluster IPs and ports are currently found through Docker-links-compatible environment variables specifying ports opened by the service proxy. There is an optional addon that provides cluster DNS for these cluster IPs. The user must create a service with the apiserver API to configure the proxy.

kube-proxy [flags]

Options

--add_dir_header

If true, adds the file directory to the header of the log messages
--alsologtostderr
log to standard error as well as files (no effect when -logtostderr=true)
--bind-address string Default: 0.0.0.0
The IP address for the proxy server to serve on (set to '0.0.0.0' for all IPv4 interfaces and '::' for all IPv6 interfaces). This parameter is ignored if a config file is specified by --config.
--bind-address-hard-fail
If true kube-proxy will treat failure to bind to a port as fatal and exit
--boot_id_file string Default: "/proc/sys/kernel/random/boot_id"
Comma-separated list of files to check for boot-id. Use the first one that exists.
--cleanup
If true cleanup iptables and ipvs rules and exit.
--cluster-cidr string
The CIDR range of pods in the cluster. When configured, traffic sent to a Service cluster IP from outside this range will be masqueraded and traffic sent from pods to an external LoadBalancer IP will be directed to the respective cluster IP instead. For dual-stack clusters, a comma-separated list is accepted with at least one CIDR per IP family (IPv4 and IPv6). This parameter is ignored if a config file is specified by --config.
--config string
The path to the configuration file.
--config-sync-period duration Default: 15m0s
How often configuration from the apiserver is refreshed. Must be greater than 0.
--conntrack-max-per-core int32 Default: 32768
Maximum number of NAT connections to track per CPU core (0 to leave the limit as-is and ignore conntrack-min).
--conntrack-min int32 Default: 131072
Minimum number of conntrack entries to allocate, regardless of conntrack-max-per-core (set conntrack-max-per-core=0 to leave the limit as-is).
--conntrack-tcp-timeout-close-wait duration Default: 1h0m0s

NAT timeout for TCP connections in the CLOSE_WAIT state
--conntrack-tcp-timeout-established duration Default: 24h0m0s
Idle timeout for established TCP connections (0 to leave as-is)
--detect-local-mode LocalMode
Mode to use to detect local traffic. This parameter is ignored if a config file is specified by --config.
--feature-gates <comma-separated 'key=True False' pairs>
A set of key=value pairs that describe feature gates for alpha/experimental features. Options are: APIListChunking=true false (BETA - default=true) APIPriorityAndFairness=true false (BETA - default=true) APIResponseCompression=true false (BETA - default=true) APIServerIdentity=true false (BETA - default=true) APIServerTracing=true false (BETA - default=true) AdmissionWebhookMatchConditions=true false (BETA - default=true) AggregatedDiscoveryEndpoint=true false (BETA - default=true) AllAlpha=true false (ALPHA - default=false) AllBeta=true false (BETA - default=false) AnyVolumeDataSource=true false (BETA - default=true) AppArmor=true false (BETA - default=true) CPUManagerPolicyAlphaOptions=true false (ALPHA - default=false) CPUManagerPolicyBetaOptions=true false (BETA - default=true) CPUManagerPolicyOptions=true false (BETA - default=true) CRDValidationRatcheting=true false (ALPHA - default=false) CSIMigrationPortworx=true false (BETA - default=false) CSINodeExpandSecret=true false (BETA - default=true) CSIVolumeHealth=true false (ALPHA - default=false) CloudControllerManagerWebhook=true false (ALPHA - default=false) CloudDualStackNodeIPs=true false (ALPHA - default=false) ClusterTrustBundle=true false (ALPHA - default=false) ComponentSLIs=true false (BETA - default=true) ConsistentListFromCache=true false (ALPHA - default=false) ContainerCheckpoint=true false (ALPHA - default=false) ContextualLogging=true false (ALPHA - default=false) CronJobsScheduledAnnotation=true false (BETA - default=true) CrossNamespaceVolumeDataSource=true false (ALPHA - default=false) CustomCPUCFSQuotaPeriod=true false (ALPHA - default=false) CustomResourceValidationExpressions=true false (BETA - default=true) DevicePluginCDIDevices=true false (ALPHA - default=false) DisableCloudProviders=true false (ALPHA - default=false) DisableKubeletCloudCredentialProviders=true false (ALPHA - default=false) DynamicResourceAllocation=true false (ALPHA - default=false) ElasticIndexedJob=true false (BETA - default=true) EventedPLEG=true false (BETA - default=false) GracefulNodeShutdown=true false (BETA - default=true) GracefulNodeShutdownBasedOnPodPriority=true false (BETA - default=true)

HPAContainerMetrics=true|false (BETA - default=true)
HPAScaleToZero=true|false (ALPHA - default=false)
HonorPVReclaimPolicy=true|false (ALPHA - default=false)
InPlacePodVerticalScaling=true|false (ALPHA - default=false)
InTreePluginAWSUnregister=true|false (ALPHA - default=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - default=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - default=false)
InTreePluginGCEUnregister=true|false (ALPHA - default=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - default=false)
InTreePluginPortworxUnregister=true|false (ALPHA - default=false)
InTreePluginInvSphereUnregister=true|false (ALPHA - default=false)
JobBackoffLimitPerIndex=true|false (ALPHA - default=false)
JobPodFailurePolicy=true|false (BETA - default=true)
JobPodReplacementPolicy=true|false (ALPHA - default=false)
JobReadyPods=true|false (BETA - default=true)
KMSv2=true|false (BETA - default=true)
KMSv2KDF=true|false (BETA - default=false)
KubeProxyDrainingTerminatingNodes=true|false (ALPHA - default=false)
KubeletCgroupDriverFromCRI=true|false (ALPHA - default=false)
KubeletInUserNamespace=true|false (ALPHA - default=false)
KubeletPodResourcesDynamicResources=true|false (ALPHA - default=false)
KubeletPodResourcesGet=true|false (ALPHA - default=false)
KubeletTracing=true|false (BETA - default=true)
LegacyServiceAccountTokenCleanUp=true|false (ALPHA - default=false)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - default=false)
LogarithmicScaleDown=true|false (BETA - default=true)
LoggingAlphaOptions=true|false (ALPHA - default=false)
LoggingBetaOptions=true|false (BETA - default=true)
MatchLabelKeysInPodTopologySpread=true|false (BETA - default=true)
MaxUnavailableStatefulSet=true|false (ALPHA - default=false)
MemoryManager=true|false (BETA - default=true)
MemoryQoS=true|false (ALPHA - default=false)
MinDomainsInPodTopologySpread=true|false (BETA - default=true)
MultiCIDRRangeAllocator=true|false (ALPHA - default=false)
MultiCIDRSvcAllocator=true|false (ALPHA - default=false)
NewVolumeManagerReconstruction=true|false (BETA - default=true)
NodeInclusionPolicyInPodTopologySpread=true|false (BETA - default=true)
NodeLogQuery=true|false (ALPHA - default=false)
NodeSwap=true|false (BETA - default=false)
OpenAPIEnums=true|false (BETA - default=true)
PDBUnhealthyPodEvictionPolicy=true|false (BETA - default=true)
PersistentVolumeLastPhaseTransitionTime=true|false (ALPHA - default=false)
PodAndContainerStatsFromCRI=true|false (ALPHA - default=false)
PodDeletionCost=true|false (BETA - default=true)
PodDisruptionConditions=true|false (BETA - default=true)
PodHostIPs=true|false (ALPHA - default=false)
PodIndexLabel=true|false (BETA - default=true)
PodReadyToStartContainersCondition=true|false (ALPHA - default=false)
PodSchedulingReadiness=true|false (BETA - default=true)
ProcMountType=true|false (ALPHA - default=false)
QOSReserved=true|false (ALPHA - default=false)
ReadWriteOncePod=true|false (BETA - default=true)
RecoverVolumeExpansionFailure=true|false (ALPHA - default=false)

RemainingItemCount=true|false (BETA - default=true)
RotateKubeletServerCertificate=true|false (BETA - default=true)
SELinuxMountReadWriteOncePod=true|false (BETA - default=true)
SchedulerQueueingHints=true|false (BETA - default=true)
SecurityContextDeny=true|false (ALPHA - default=false)
ServiceNodePortStaticSubrange=true|false (BETA - default=true)
SidecarContainers=true|false (ALPHA - default=false)
SizeMemoryBackedVolumes=true|false (BETA - default=true)
SkipReadOnlyValidationGCE=true|false (ALPHA - default=false)
StableLoadBalancerNodeSet=true|false (BETA - default=true)
StatefulSetAutoDeletePVC=true|false (BETA - default=true)
StatefulSetStartOrdinal=true|false (BETA - default=true)
StorageVersionAPI=true|false (ALPHA - default=false)
StorageVersionHash=true|false (BETA - default=true)
TopologyAwareHints=true|false (BETA - default=true)
TopologyManagerPolicyAlphaOptions=true|false (ALPHA - default=false)
TopologyManagerPolicyBetaOptions=true|false (BETA - default=true)
TopologyManagerPolicyOptions=true|false (BETA - default=true)
UnknownVersionInteroperabilityProxy=true|false (ALPHA - default=false)
UserNamespacesSupport=true|false (ALPHA - default=false)
ValidatingAdmissionPolicy=true|false (BETA - default=false)
VolumeCapacityPriority=true|false (ALPHA - default=false)
WatchList=true|false (ALPHA - default=false)
WinDSR=true|false (ALPHA - default=false)
WinOverlay=true|false (BETA - default=true)
WindowsHostNetwork=true|false (ALPHA - default=true)
This parameter is ignored if a config file is specified by --config.

--healthz-bind-address ipport Default: 0.0.0.0:10256

The IP address with port for the health check server to serve on (set to '0.0.0.0:10256' for all IPv4 interfaces and '[::]:10256' for all IPv6 interfaces). Set empty to disable. This parameter is ignored if a config file is specified by --config.

-h, --help

help for kube-proxy

--hostname-override string

If non-empty, will use this string as identification instead of the actual hostname.

--iptables-localhost-nodeports Default: true

If false Kube-proxy will disable the legacy behavior of allowing NodePort services to be accessed via localhost, This only applies to iptables mode and ipv4.

--iptables-masquerade-bit int32 Default: 14

If using the pure iptables proxy, the bit of the fwmark space to mark packets requiring SNAT with. Must be within the range [0, 31].

--iptables-min-sync-period duration	Default: 1s
The minimum interval of how often the iptables rules can be refreshed as endpoints and services change (e.g. '5s', '1m', '2h22m').	
--iptables-sync-period duration	Default: 30s
The maximum interval of how often iptables rules are refreshed (e.g. '5s', '1m', '2h22m'). Must be greater than 0.	
--ipvs-exclude-cidrs strings	
A comma-separated list of CIDR's which the ipvs proxier should not touch when cleaning up IPVS rules.	
--ipvs-min-sync-period duration	
The minimum interval of how often the ipvs rules can be refreshed as endpoints and services change (e.g. '5s', '1m', '2h22m').	
--ipvs-scheduler string	
The ipvs scheduler type when proxy mode is ipvs	
--ipvs-strict-arp	
Enable strict ARP by setting arp_ignore to 1 and arp_announce to 2	
--ipvs-sync-period duration	Default: 30s
The maximum interval of how often ipvs rules are refreshed (e.g. '5s', '1m', '2h22m'). Must be greater than 0.	
--ipvs-tcp-timeout duration	
The timeout for idle IPVS TCP connections, 0 to leave as-is. (e.g. '5s', '1m', '2h22m').	
--ipvs-tcpfin-timeout duration	
The timeout for IPVS TCP connections after receiving a FIN packet, 0 to leave as-is. (e.g. '5s', '1m', '2h22m').	
--ipvs-udp-timeout duration	
The timeout for IPVS UDP packets, 0 to leave as-is. (e.g. '5s', '1m', '2h22m').	
--kube-api-burst int32	Default: 10
Burst to use while talking with kubernetes apiserver	
--kube-api-content-type string	Default: "application/vnd.kubernetes.protobuf"

Content type of requests sent to apiserver.
--kube-api-qps float Default: 5
QPS to use while talking with kubernetes apiserver
--kubeconfig string
Path to kubeconfig file with authorization information (the master location can be overridden by the master flag).
--log-flush-frequency duration Default: 5s
Maximum number of seconds between log flushes
--log_backtrace_at <a string in the form 'file:N'> Default: :0
when logging hits line file:N, emit a stack trace
--log_dir string
If non-empty, write log files in this directory (no effect when -logtostderr=true)
--log_file string
If non-empty, use this log file (no effect when -logtostderr=true)
--log_file_max_size uint Default: 1800
Defines the maximum size a log file can grow to (no effect when -logtostderr=true). Unit is megabytes. If the value is 0, the maximum file size is unlimited.
--logging-format string Default: "text"
Sets the log format. Permitted formats: "text".
--logtostderr Default: true
log to standard error instead of files
--machine_id_file string Default: "/etc/machine-id,/var/lib/dbus/machine-id"
Comma-separated list of files to check for machine-id. Use the first one that exists.
--masquerade-all
If using the pure iptables proxy, SNAT all traffic sent via Service cluster IPs (this not commonly needed)
--master string

The address of the Kubernetes API server (overrides any value in kubeconfig)
--metrics-bind-address ipport Default: 127.0.0.1:10249
The IP address with port for the metrics server to serve on (set to '0.0.0.0:10249' for all IPv4 interfaces and '[::]:10249' for all IPv6 interfaces). Set empty to disable. This parameter is ignored if a config file is specified by --config.
--nodeport-addresses strings
A string slice of values which specify the addresses to use for NodePorts. Values may be valid IP blocks (e.g. 1.2.3.0/24, 1.2.3.4/32). The default empty string slice ([]) means to use all local addresses. This parameter is ignored if a config file is specified by --config.
--one_output
If true, only write logs to their native severity level (vs also writing to each lower severity level; no effect when -logtostderr=true)
--oom-score-adj int32 Default: -999
The oom-score-adj value for kube-proxy process. Values must be within the range [-1000, 1000]. This parameter is ignored if a config file is specified by --config.
--pod-bridge-interface string
A bridge interface name in the cluster. Kube-proxy considers traffic as local if originating from an interface which matches the value. This argument should be set if DetectLocalMode is set to BridgeInterface.
--pod-interface-name-prefix string
An interface prefix in the cluster. Kube-proxy considers traffic as local if originating from interfaces that match the given prefix. This argument should be set if DetectLocalMode is set to InterfaceNamePrefix.
--profiling
If true enables profiling via web interface on /debug/pprof handler. This parameter is ignored if a config file is specified by --config.
--proxy-mode ProxyMode
Which proxy mode to use: on Linux this can be 'iptables' (default) or 'ipvs'. On Windows the only supported value is 'kernelspace'.This parameter is ignored if a config file is specified by --config.
--proxy-port-range port-range

Range of host ports (beginPort-endPort, single port or beginPort+offset, inclusive) that may be consumed in order to proxy service traffic. If (unspecified, 0, or 0-0) then ports will be randomly chosen.

--show-hidden-metrics-for-version string

The previous version for which you want to show hidden metrics. Only the previous minor version is meaningful, other values will not be allowed. The format is <major>.<minor>, e.g.: '1.16'. The purpose of this format is make sure you have the opportunity to notice if the next release hides additional metrics, rather than being surprised when they are permanently removed in the release after that. This parameter is ignored if a config file is specified by --config.

--skip_headers

If true, avoid header prefixes in the log messages

--skip_log_headers

If true, avoid headers when opening log files (no effect when -logtostderr=true)

--stderrthreshold int Default: 2

logs at or above this threshold go to stderr when writing to files and stderr (no effect when -logtostderr=true or -alsologtostderr=false)

-v, --v int

number for the log level verbosity

--version version[=true]

--version, --version=raw prints version information and quits; --version=vX.Y.Z... sets the reported version

--vmodule pattern=N,...

comma-separated list of pattern=N settings for file-filtered logging (only works for text log format)

--write-config-to string

If set, write the default configuration values to this file and exit.

kube-scheduler

Synopsis

The Kubernetes scheduler is a control plane process which assigns Pods to Nodes. The scheduler determines which Nodes are valid placements for each Pod in the scheduling queue according to constraints and available resources. The scheduler then ranks each valid Node and binds the Pod to a suitable Node. Multiple different schedulers may be used within a cluster; kube-scheduler is the reference implementation. See [scheduling](#) for more information about scheduling and the kube-scheduler component.

kube-scheduler [flags]

Options

--allow-metric-labels stringToString Default: []
The map from metric-label to value allow-list of this label. The key's format is <MetricName>,<LabelName>. The value's format is <allowed_value>,<allowed_value>...e.g. metric1,label1='v1,v2,v3', metric1,label2='v1,v2,v3' metric2,label1='v1,v2,v3'.
--authentication-kubeconfig string
kubeconfig file pointing at the 'core' kubernetes server with enough rights to create tokenreviews.authentication.k8s.io. This is optional. If empty, all token requests are considered to be anonymous and no client CA is looked up in the cluster.
--authentication-skip-lookup
If false, the authentication-kubeconfig will be used to lookup missing authentication configuration from the cluster.
--authentication-token-webhook-cache-ttl duration Default: 10s
The duration to cache responses from the webhook token authenticator.
--authentication-tolerate-lookup-failure Default: true
If true, failures to look up missing authentication configuration from the cluster are not considered fatal. Note that this can result in authentication that treats all requests as anonymous.
--authorization-always-allow-paths strings Default: "/healthz,/readyz,/livez"
A list of HTTP paths to skip during authorization, i.e. these are authorized without contacting the 'core' kubernetes server.
--authorization-kubeconfig string

kubeconfig file pointing at the 'core' kubernetes server with enough rights to create subjectaccessreviews.authorization.k8s.io. This is optional. If empty, all requests not skipped by authorization are forbidden.
--authorization-webhook-cache-authorized-ttl duration Default: 10s
The duration to cache 'authorized' responses from the webhook authorizer.
--authorization-webhook-cache-unauthorized-ttl duration Default: 10s
The duration to cache 'unauthorized' responses from the webhook authorizer.
--azure-container-registry-config string
Path to the file containing Azure container registry configuration information.
--bind-address string Default: 0.0.0.0
The IP address on which to listen for the --secure-port port. The associated interface(s) must be reachable by the rest of the cluster, and by CLI/web clients. If blank or an unspecified address (0.0.0.0 or ::), all interfaces and IP address families will be used.
--cert-dir string
The directory where the TLS certs are located. If --tls-cert-file and --tls-private-key-file are provided, this flag will be ignored.
--client-ca-file string
If set, any request presenting a client certificate signed by one of the authorities in the client-ca-file is authenticated with an identity corresponding to the CommonName of the client certificate.
--config string
The path to the configuration file.
--contention-profiling Default: true
DEPRECATED: enable block profiling, if profiling is enabled. This parameter is ignored if a config file is specified in --config.
--disabled-metrics strings
This flag provides an escape hatch for misbehaving metrics. You must provide the fully qualified metric name in order to disable it. Disclaimer: disabling metrics is higher in precedence than showing hidden metrics.
--feature-gates <comma-separated 'key=True False' pairs>

A set of key=value pairs that describe feature gates for alpha/experimental features. Options are:

APIListChunking=true|false (BETA - default=true)
APIPriorityAndFairness=true|false (BETA - default=true)
APIResponseCompression=true|false (BETA - default=true)
APIServerIdentity=true|false (BETA - default=true)
APIServerTracing=true|false (BETA - default=true)
AdmissionWebhookMatchConditions=true|false (BETA - default=true)
AggregatedDiscoveryEndpoint=true|false (BETA - default=true)
AllAlpha=true|false (ALPHA - default=false)
AllBeta=true|false (BETA - default=false)
AnyVolumeDataSource=true|false (BETA - default=true)
AppArmor=true|false (BETA - default=true)
CPUManagerPolicyAlphaOptions=true|false (ALPHA - default=false)
CPUManagerPolicyBetaOptions=true|false (BETA - default=true)
CPUManagerPolicyOptions=true|false (BETA - default=true)
CRDValidationRatcheting=true|false (ALPHA - default=false)
CSIMigrationPortworx=true|false (BETA - default=false)
CSINodeExpandSecret=true|false (BETA - default=true)
CSIVolumeHealth=true|false (ALPHA - default=false)
CloudControllerManagerWebhook=true|false (ALPHA - default=false)
CloudDualStackNodeIPs=true|false (ALPHA - default=false)
ClusterTrustBundle=true|false (ALPHA - default=false)
ComponentSLIs=true|false (BETA - default=true)
ConsistentListFromCache=true|false (ALPHA - default=false)
ContainerCheckpoint=true|false (ALPHA - default=false)
ContextualLogging=true|false (ALPHA - default=false)
CronJobsScheduledAnnotation=true|false (BETA - default=true)
CrossNamespaceVolumeDataSource=true|false (ALPHA - default=false)
CustomCPUCFSQuotaPeriod=true|false (ALPHA - default=false)
CustomResourceValidationExpressions=true|false (BETA - default=true)
DevicePluginCDIDevices=true|false (ALPHA - default=false)
DisableCloudProviders=true|false (ALPHA - default=false)
DisableKubeletCloudCredentialProviders=true|false (ALPHA - default=false)
DynamicResourceAllocation=true|false (ALPHA - default=false)
ElasticIndexedJob=true|false (BETA - default=true)
EventedPLEG=true|false (BETA - default=false)
GracefulNodeShutdown=true|false (BETA - default=true)
GracefulNodeShutdownBasedOnPodPriority=true|false (BETA - default=true)
HPAContainerMetrics=true|false (BETA - default=true)
HPAScaleToZero=true|false (ALPHA - default=false)
HonorPVReclaimPolicy=true|false (ALPHA - default=false)
InPlacePodVerticalScaling=true|false (ALPHA - default=false)
InTreePluginAWSUnregister=true|false (ALPHA - default=false)
InTreePluginAzureDiskUnregister=true|false (ALPHA - default=false)
InTreePluginAzureFileUnregister=true|false (ALPHA - default=false)
InTreePluginGCEUnregister=true|false (ALPHA - default=false)
InTreePluginOpenStackUnregister=true|false (ALPHA - default=false)
InTreePluginPortworxUnregister=true|false (ALPHA - default=false)
InTreePluginvSphereUnregister=true|false (ALPHA - default=false)
JobBackoffLimitPerIndex=true|false (ALPHA - default=false)
JobPodFailurePolicy=true|false (BETA - default=true)

JobPodReplacementPolicy=true|false (ALPHA - default=false)
JobReadyPods=true|false (BETA - default=true)
KMSv2=true|false (BETA - default=true)
KMSv2KDF=true|false (BETA - default=false)
KubeProxyDrainingTerminatingNodes=true|false (ALPHA - default=false)
KubeletCgroupDriverFromCRI=true|false (ALPHA - default=false)
KubeletInUserNamespace=true|false (ALPHA - default=false)
KubeletPodResourcesDynamicResources=true|false (ALPHA - default=false)
KubeletPodResourcesGet=true|false (ALPHA - default=false)
KubeletTracing=true|false (BETA - default=true)
LegacyServiceAccountTokenCleanUp=true|false (ALPHA - default=false)
LocalStorageCapacityIsolationFSQuotaMonitoring=true|false (ALPHA - default=false)
LogarithmicScaleDown=true|false (BETA - default=true)
LoggingAlphaOptions=true|false (ALPHA - default=false)
LoggingBetaOptions=true|false (BETA - default=true)
MatchLabelKeysInPodTopologySpread=true|false (BETA - default=true)
MaxUnavailableStatefulSet=true|false (ALPHA - default=false)
MemoryManager=true|false (BETA - default=true)
MemoryQoS=true|false (ALPHA - default=false)
MinDomainsInPodTopologySpread=true|false (BETA - default=true)
MultiCIDRRangeAllocator=true|false (ALPHA - default=false)
MultiCIDRSvcAllocator=true|false (ALPHA - default=false)
NewVolumeManagerReconstruction=true|false (BETA - default=true)
NodeInclusionPolicyInPodTopologySpread=true|false (BETA - default=true)
NodeLogQuery=true|false (ALPHA - default=false)
NodeSwap=true|false (BETA - default=false)
OpenAPIEnums=true|false (BETA - default=true)
PDBUnhealthyPodEvictionPolicy=true|false (BETA - default=true)
PersistentVolumeLastPhaseTransitionTime=true|false (ALPHA - default=false)
PodAndContainerStatsFromCRI=true|false (ALPHA - default=false)
PodDeletionCost=true|false (BETA - default=true)
PodDisruptionConditions=true|false (BETA - default=true)
PodHostIPs=true|false (ALPHA - default=false)
PodIndexLabel=true|false (BETA - default=true)
PodReadyToStartContainersCondition=true|false (ALPHA - default=false)
PodSchedulingReadiness=true|false (BETA - default=true)
ProcMountType=true|false (ALPHA - default=false)
QOSReserved=true|false (ALPHA - default=false)
ReadWriteOncePod=true|false (BETA - default=true)
RecoverVolumeExpansionFailure=true|false (ALPHA - default=false)
RemainingItemCount=true|false (BETA - default=true)
RotateKubeletServerCertificate=true|false (BETA - default=true)
SELinuxMountReadWriteOncePod=true|false (BETA - default=true)
SchedulerQueueingHints=true|false (BETA - default=true)
SecurityContextDeny=true|false (ALPHA - default=false)
ServiceNodePortStaticSubrange=true|false (BETA - default=true)
SidecarContainers=true|false (ALPHA - default=false)
SizeMemoryBackedVolumes=true|false (BETA - default=true)
SkipReadOnlyValidationGCE=true|false (ALPHA - default=false)
StableLoadBalancerNodeSet=true|false (BETA - default=true)
StatefulSetAutoDeletePVC=true|false (BETA - default=true)
StatefulSetStartOrdinal=true|false (BETA - default=true)
StorageVersionAPI=true|false (ALPHA - default=false)

StorageVersionHash=true|false (BETA - default=true)
TopologyAwareHints=true|false (BETA - default=true)
TopologyManagerPolicyAlphaOptions=true|false (ALPHA - default=false)
TopologyManagerPolicyBetaOptions=true|false (BETA - default=true)
TopologyManagerPolicyOptions=true|false (BETA - default=true)
UnknownVersionInteroperabilityProxy=true|false (ALPHA - default=false)
UserNamespacesSupport=true|false (ALPHA - default=false)
ValidatingAdmissionPolicy=true|false (BETA - default=false)
VolumeCapacityPriority=true|false (ALPHA - default=false)
WatchList=true|false (ALPHA - default=false)
WinDSR=true|false (ALPHA - default=false)
WinOverlay=true|false (BETA - default=true)
WindowsHostNetwork=true|false (ALPHA - default=true)

-h, --help

help for kube-scheduler

--http2-max-streams-per-connection int

The limit that the server gives to clients for the maximum number of streams in an HTTP/2 connection. Zero means to use golang's default.

--kube-api-burst int32 Default: 100

DEPRECATED: burst to use while talking with kubernetes apiserver. This parameter is ignored if a config file is specified in --config.

--kube-api-content-type string Default: "application/vnd.kubernetes.protobuf"

DEPRECATED: content type of requests sent to apiserver. This parameter is ignored if a config file is specified in --config.

--kube-api-qps float Default: 50

DEPRECATED: QPS to use while talking with kubernetes apiserver. This parameter is ignored if a config file is specified in --config.

--kubeconfig string

DEPRECATED: path to kubeconfig file with authorization and master location information. This parameter is ignored if a config file is specified in --config.

--leader-elect Default: true

Start a leader election client and gain leadership before executing the main loop. Enable this when running replicated components for high availability.

--leader-elect-lease-duration duration Default: 15s

The duration that non-leader candidates will wait after observing a leadership renewal until attempting to acquire leadership of a led but unrenewed leader slot. This is effectively the

maximum duration that a leader can be stopped before it is replaced by another candidate. This is only applicable if leader election is enabled.

--leader-elect-renew-deadline duration Default: 10s

The interval between attempts by the acting master to renew a leadership slot before it stops leading. This must be less than the lease duration. This is only applicable if leader election is enabled.

--leader-elect-resource-lock string Default: "leases"

The type of resource object that is used for locking during leader election. Supported options are 'leases', 'endpointsleases' and 'configmapsleases'.

--leader-elect-resource-name string Default: "kube-scheduler"

The name of resource object that is used for locking during leader election.

--leader-elect-resource-namespace string Default: "kube-system"

The namespace of resource object that is used for locking during leader election.

--leader-elect-retry-period duration Default: 2s

The duration the clients should wait between attempting acquisition and renewal of a leadership. This is only applicable if leader election is enabled.

--log-flush-frequency duration Default: 5s

Maximum number of seconds between log flushes

--logging-format string Default: "text"

Sets the log format. Permitted formats: "text".

--master string

The address of the Kubernetes API server (overrides any value in kubeconfig)

--permit-address-sharing

If true, SO_REUSEADDR will be used when binding the port. This allows binding to wildcard IPs like 0.0.0.0 and specific IPs in parallel, and it avoids waiting for the kernel to release sockets in TIME_WAIT state. [default=false]

--permit-port-sharing

If true, SO_REUSEPORT will be used when binding the port, which allows more than one instance to bind on the same address and port. [default=false]

--pod-max-in-unschedulable-pods-duration duration Default: 5m0s

DEPRECATED: the maximum time a pod can stay in unschedulablePods. If a pod stays in unschedulablePods for longer than this value, the pod will be moved from unschedulablePods to backoffQ or activeQ. This flag is deprecated and will be removed in 1.26

--profiling Default: true

DEPRECATED: enable profiling via web interface host:port/debug/pprof/. This parameter is ignored if a config file is specified in --config.

--requestheader-allowed-names strings

List of client certificate common names to allow to provide usernames in headers specified by --requestheader-username-headers. If empty, any client certificate validated by the authorities in --requestheader-client-ca-file is allowed.

--requestheader-client-ca-file string

Root certificate bundle to use to verify client certificates on incoming requests before trusting usernames in headers specified by --requestheader-username-headers. WARNING: generally do not depend on authorization being already done for incoming requests.

--requestheader-extra-headers-prefix strings Default: "x-remote-extra-"

List of request header prefixes to inspect. X-Remote-Extra- is suggested.

--requestheader-group-headers strings Default: "x-remote-group"

List of request headers to inspect for groups. X-Remote-Group is suggested.

--requestheader-username-headers strings Default: "x-remote-user"

List of request headers to inspect for usernames. X-Remote-User is common.

--secure-port int Default: 10259

The port on which to serve HTTPS with authentication and authorization. If 0, don't serve HTTPS at all.

--show-hidden-metrics-for-version string

The previous version for which you want to show hidden metrics. Only the previous minor version is meaningful, other values will not be allowed. The format is <major>.<minor>, e.g.: '1.16'. The purpose of this format is make sure you have the opportunity to notice if the next release hides additional metrics, rather than being surprised when they are permanently removed in the release after that.

--tls-cert-file string

File containing the default x509 Certificate for HTTPS. (CA cert, if any, concatenated after server cert). If HTTPS serving is enabled, and --tls-cert-file and --tls-private-key-file are not

provided, a self-signed certificate and key are generated for the public address and saved to the directory specified by --cert-dir.

--tls-cipher-suites strings

Comma-separated list of cipher suites for the server. If omitted, the default Go cipher suites will be used.

Preferred values: TLS_AES_128_GCM_SHA256, TLS_AES_256_GCM_SHA384, TLS_CHACHA20_POLY1305_SHA256, TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_ECDSA_WITH_CHACHA20_POLY1305_SHA256, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256, TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305, TLS_ECDHE_RSA_WITH_CHACHA20_POLY1305_SHA256, TLS_RSA_WITH_AES_128_CBC_SHA, TLS_RSA_WITH_AES_128_GCM_SHA256, TLS_RSA_WITH_AES_256_CBC_SHA, TLS_RSA_WITH_AES_256_GCM_SHA384.

Insecure values: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_ECDSA_WITH_RC4_128_SHA, TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA256, TLS_ECDHE_RSA_WITH_RC4_128_SHA, TLS_RSA_WITH_3DES_EDE_CBC_SHA, TLS_RSA_WITH_AES_128_CBC_SHA256, TLS_RSA_WITH_RC4_128_SHA.

--tls-min-version string

Minimum TLS version supported. Possible values: VersionTLS10, VersionTLS11, VersionTLS12, VersionTLS13

--tls-private-key-file string

File containing the default x509 private key matching --tls-cert-file.

--tls-sni-cert-key string

A pair of x509 certificate and private key file paths, optionally suffixed with a list of domain patterns which are fully qualified domain names, possibly with prefixed wildcard segments. The domain patterns also allow IP addresses, but IPs should only be used if the apiserver has visibility to the IP address requested by a client. If no domain patterns are provided, the names of the certificate are extracted. Non-wildcard matches trump over wildcard matches, explicit domain patterns trump over extracted names. For multiple key/certificate pairs, use the --tls-sni-cert-key multiple times. Examples: "example.crt,example.key" or "foo.crt,foo.key:*.foo.com,foo.com".

-v, --v int

number for the log level verbosity
--version version[=true]
--version, --version=raw prints version information and quits; --version=vX.Y.Z... sets the reported version
--vmodule pattern=N,...
comma-separated list of pattern=N settings for file-filtered logging (only works for text log format)
--write-config-to string
If set, write the configuration values to this file and exit.

Debug cluster

Flow control

Flow control

API Priority and Fairness controls the behavior of the Kubernetes API server in an overload situation. You can find more information about it in the [API Priority and Fairness](#) documentation.

Diagnostics

Every HTTP response from an API server with the priority and fairness feature enabled has two extra headers: X-Kubernetes-PF-FlowSchema-UID and X-Kubernetes-PF-PriorityLevel-UID, noting the flow schema that matched the request and the priority level to which it was assigned, respectively. The API objects' names are not included in these headers (to avoid revealing details in case the requesting user does not have permission to view them). When debugging, you can use a command such as:

```
kubectl get flowschemas -o custom-columns="uid:{metadata.uid},name:{metadata.name}"
kubectl get prioritylevelconfigurations -o custom-columns="uid:{metadata.uid},name:{metadata.name}"
```

to get a mapping of UIDs to names for both FlowSchemas and PriorityLevelConfigurations.

Debug endpoints

With the APIPriorityAndFairness feature enabled, the kube-apiserver serves the following additional paths at its HTTP(S) ports.

You need to ensure you have permissions to access these endpoints. You don't have to do anything if you are using admin. Permissions can be granted if needed following the [RBAC](#) doc to access /debug/api_priority_and_fairness/ by specifying nonResourceURLs.

- /debug/api_priority_and_fairness/dump_priority_levels - a listing of all the priority levels and the current state of each. You can fetch like this:

```
kubectl get --raw /debug/api_priority_and_fairness/dump_priority_levels
```

The output will be in CSV and similar to this:

PriorityLevelName	ActiveQueues	IsIdle	IsQuiescing	WaitingRequests	ExecutingRequests	DispatchedRequests	RejectedRequests	TimedoutRequests	CancelledRequests
catch-all	0	true	false	0	0	1	0	0	0
0	0								
exempt	0	true	false	0	0	0	0	0	0
0	0								
global-default	0	true	false	0	0	46	0	0	0
0	0								
leader-election	0	true	false	0	0	4	0	0	0
0	0								
node-high	0	true	false	0	0	34	0	0	0
0	0								
system	0	true	false	0	0	48	0	0	0
0	0								
workload-high	0	true	false	0	0	500	0	0	0
0	0								
workload-low	0	true	false	0	0	0	0	0	0
0	0								

Explanation for selected column names:

- IsQuiescing indicates if this priority level will be removed when its queues have been drained.
- /debug/api_priority_and_fairness/dump_queues - a listing of all the queues and their current state. You can fetch like this:

```
kubectl get --raw /debug/api_priority_and_fairness/dump_queues
```

The output will be in CSV and similar to this:

PriorityLevelName	Index	PendingRequests	ExecutingRequests	SeatsInUse	NextDispatchR	InitialSeatsSum	MaxSeatsSum	TotalWorkSum
workload-low	14	27	0	0	77.64342019ss	270	270	0.81000000ss
workload-low	74	26	0	0	76.95387841ss	260	260	0.78000000ss
...								
leader-election	0	0	0	0	5088.87053833ss	0	0	0.00000000ss
leader-election	1	0	0	0	0.00000000ss	0	0	0.00000000ss

```

...
workload-high, 0, 0, 0, 0, 0.0000000ss, 0, 0,
0.0000000ss
workload-high, 1, 0, 0, 0, 1119.44936475ss, 0,
0, 0.0000000ss

```

Explanation for selected column names:

- NextDispatchR: The R progress meter reading, in units of seat-seconds, at which the next request will be dispatched.
- InitialSeatsSum: The sum of InitialSeats associated with all requests in a given queue.
- MaxSeatsSum: The sum of MaxSeats associated with all requests in a given queue.
- TotalWorkSum: The sum of total work, in units of seat-seconds, of all waiting requests in a given queue.

Note: seat-second (abbreviate as ss) is a measure of work, in units of seat-seconds, in the APF world.

- /debug/api_priority_and_fairness/dump_requests - a listing of all the requests including requests waiting in a queue and requests being executing. You can fetch like this:

```
kubectl get --raw /debug/api_priority_and_fairness/dump_requests
```

The output will be in CSV and similar to this:

```

PriorityLevelName, FlowSchemaName, QueueIndex, RequestIndexInQueue,
FlowDistingsher, ArriveTime, InitialSeats, FinalSeats,
AdditionalLatency, StartTime
exempt, exempt, -1, -1, ,
2023-07-15T04:51:25.596404345Z, 1, 0, 0s,
2023-07-15T04:51:25.596404345Z
workload-low, service-accounts, 14, 0,
system:serviceaccount:default:loadtest, 2023-07-18T00:12:51.386556253Z, 10, 0,
0s, 0001-01-01T00:00:00Z
workload-low, service-accounts, 14, 1,
system:serviceaccount:default:loadtest, 2023-07-18T00:12:51.487092539Z, 10, 0,
0s, 0001-01-01T00:00:00Z

```

You can get a more detailed listing with a command like this:

```
kubectl get --raw '/debug/api_priority_and_fairness/dump_requests?
includeRequestDetails=1'
```

The output will be in CSV and similar to this:

```

PriorityLevelName, FlowSchemaName, QueueIndex, RequestIndexInQueue,
FlowDistingsher, ArriveTime, InitialSeats, FinalSeats,
AdditionalLatency, StartTime, UserName, Verb,
APIPath, Namespace, Name, APIVersion, Resource, SubResource
exempt, exempt, -1, -1, ,
2023-07-15T04:51:25.596404345Z, 1, 0, 0s,
2023-07-15T04:51:25.596404345Z, system:serviceaccount:system:admin, list, /api/v1/
namespaces/kube-stress/configmaps, kube-stress, , v1, configmaps,

```

```

workload-low,    service-accounts, 14,      0,
system:serviceaccount:default:loadtest, 2023-07-18T00:13:08.986534842Z, 10,      0,
0s,      0001-01-01T00:00:00Z,      system:serviceaccount:default:loadtest, list, /
api/v1/namespaces/kube-stress/configmaps, kube-stress, ,   v1,      configmaps,
workload-low,    service-accounts, 14,      1,
system:serviceaccount:default:loadtest, 2023-07-18T00:13:09.086476021Z, 10,      0,
0s,      0001-01-01T00:00:00Z,      system:serviceaccount:default:loadtest, list, /
api/v1/namespaces/kube-stress/configmaps, kube-stress, ,   v1,      configmaps,

```

Explanation for selected column names:

- QueueIndex: The index of the queue. It will be -1 for priority levels without queues.
- RequestIndexInQueue: The index in the queue for a given request. It will be -1 for executing requests.
- InitialSeats: The number of seats will be occupied during the initial (normal) stage of execution of the request.
- FinalSeats: The number of seats will be occupied during the final stage of request execution, accounting for the associated WATCH notifications.
- AdditionalLatency: The extra time taken during the final stage of request execution. FinalSeats will be occupied during this time period. It does not mean any latency that a user will observe.
- StartTime: The time a request starts to execute. It will be 0001-01-01T00:00:00Z for queued requests.

Debug logging

At -v=3 or more verbosity, the API server outputs an httplog line for every request in the API server log, and it includes the following attributes.

- apf_fs: the name of the flow schema to which the request was classified.
- apf_pl: the name of the priority level for that flow schema.
- apf_iseats: the number of seats determined for the initial (normal) stage of execution of the request.
- apf_fseats: the number of seats determined for the final stage of execution (accounting for the associated watch notifications) of the request.
- apf_additionalLatency: the duration of the final stage of execution of the request.

At higher levels of verbosity there will be log lines exposing details of how APF handled the request, primarily for debugging purposes.

Response headers

APF adds the following two headers to each HTTP response message. They won't appear in the audit log. They can be viewed from the client side. For client using klog, use verbosity -v=8 or higher to view these headers.

- X-Kubernetes-PF-FlowSchema-UID holds the UID of the FlowSchema object to which the corresponding request was classified.
- X-Kubernetes-PF-PriorityLevel-UID holds the UID of the PriorityLevelConfiguration object associated with that FlowSchema.

What's next

For background information on design details for API priority and fairness, see the [enhancement proposal](#).

Configuration APIs

[Client Authentication \(v1\)](#)

[Client Authentication \(v1beta1\)](#)

[Event Rate Limit Configuration \(v1alpha1\)](#)

[Image Policy API \(v1alpha1\)](#)

[kube-apiserver Admission \(v1\)](#)

[kube-apiserver Audit Configuration \(v1\)](#)

[kube-apiserver Configuration \(v1\)](#)

[kube-apiserver Configuration \(v1alpha1\)](#)

[kube-apiserver Configuration \(v1beta1\)](#)

[kube-apiserver Encryption Configuration \(v1\)](#)

[kube-controller-manager Configuration \(v1alpha1\)](#)

[kube-proxy Configuration \(v1alpha1\)](#)

[kube-scheduler Configuration \(v1\)](#)

[kube-scheduler Configuration \(v1beta3\)](#)

[kubeadm Configuration \(v1beta3\)](#)

[kubeadm Configuration \(v1beta4\)](#)

[kubecfg \(v1\)](#)

[Kubelet Configuration \(v1\)](#)

[Kubelet Configuration \(v1alpha1\)](#)

[Kubelet Configuration \(v1beta1\)](#)

[Kubelet CredentialProvider \(v1\)](#)

[Kubelet CredentialProvider \(v1alpha1\)](#)

[Kubelet CredentialProvider \(v1beta1\)](#)

[WebhookAdmission Configuration \(v1\)](#)

Client Authentication (v1)

Resource Types

- [ExecCredential](#)

ExecCredential

ExecCredential is used by exec-based plugins to communicate credentials to HTTP transports.

Field	Description
apiVersion string	client.authentication.k8s.io/v1
kind string	ExecCredential
spec [Required] ExecCredentialSpec	Spec holds information passed to the plugin by the transport.
status ExecCredentialStatus	Status is filled in by the plugin and holds the credentials that the transport should use to contact the API.

Cluster

Appears in:

- [ExecCredentialSpec](#)

Cluster contains information to allow an exec plugin to communicate with the kubernetes cluster being authenticated to.

To ensure that this struct contains everything someone would need to communicate with a kubernetes cluster (just like they would via a kubeconfig), the fields should shadow "k8s.io/client-go/tools/clientcmd/api/v1".Cluster, with the exception of CertificateAuthority, since CA data will always be passed to the plugin as bytes.

Field	Description
server [Required] string	Server is the address of the kubernetes cluster (https://hostname:port).

Field	Description
tls-server-name string	TLS Server Name is passed to the server for SNI and is used in the client to check server certificates against. If ServerName is empty, the hostname used to contact the server is used.
insecure-skip-tls-verify bool	Insecure Skip TLS Verify skips the validity check for the server's certificate. This will make your HTTPS connections insecure.
certificate-authority-data []byte	CAData contains PEM-encoded certificate authority certificates. If empty, system roots should be used.
proxy-url string	ProxyURL is the URL to the proxy to be used for all requests to this cluster.
disable-compression bool	DisableCompression allows client to opt-out of response compression for all requests to the server. This is useful to speed up requests (specifically lists) when client-server network bandwidth is ample, by saving time on compression (server-side) and decompression (client-side): https://github.com/kubernetes/kubernetes/issues/112296 .
config k8s.io/apimachinery/pkg/runtime.RawExtension	<p>Config holds additional config data that is specific to the exec plugin with regards to the cluster being authenticated to.</p> <p>This data is sourced from the clientcmd Cluster object's extensions[client.authentication.k8s.io/exec] field:</p> <p>clusters:</p> <ul style="list-style-type: none"> name: my-cluster cluster: ... extensions: <ul style="list-style-type: none"> name: client.authentication.k8s.io/exec # reserved extension name for per cluster exec config extension: audience: 06e3fb18de8 # arbitrary config <p>In some environments, the user config may be exactly the same across many clusters (i.e. call this exec plugin) minus some details that are specific to each cluster such as the audience. This field allows the per cluster config to be directly specified with the cluster info. Using this field to store secret data is not recommended as one of the prime benefits of exec plugins is that no secrets need to be stored directly in the kubeconfig.</p>

ExecCredentialSpec

Appears in:

- [ExecCredential](#)

ExecCredentialSpec holds request and runtime specific information provided by the transport.

Field	Description
cluster Cluster	Cluster contains information to allow an exec plugin to communicate with the kubernetes cluster being authenticated to. Note that Cluster is non-nil only when provideClusterInfo is set to true in the exec provider config (i.e., ExecConfig.ProvideClusterInfo).
interactive [Required] bool	Interactive declares whether stdin has been passed to this exec plugin.

ExecCredentialStatus

Appears in:

- [ExecCredential](#)

ExecCredentialStatus holds credentials for the transport to use.

Token and ClientKeyData are sensitive fields. This data should only be transmitted in-memory between client and exec plugin process. Exec plugin itself should at least be protected via file permissions.

Field	Description
expirationTimestamp meta/v1.Time	ExpirationTimestamp indicates a time when the provided credentials expire.
token [Required] string	Token is a bearer token used by the client for request authentication.
clientCertificateData [Required] string	PEM-encoded client TLS certificates (including intermediates, if any).
clientKeyData [Required] string	PEM-encoded private key for the above certificate.

Client Authentication (v1beta1)

Resource Types

- [ExecCredential](#)

ExecCredential

ExecCredential is used by exec-based plugins to communicate credentials to HTTP transports.

Field	Description
apiVersion string	client.authentication.k8s.io/v1beta1
kind string	ExecCredential
spec [Required] ExecCredentialSpec	Spec holds information passed to the plugin by the transport.
status ExecCredentialStatus	Status is filled in by the plugin and holds the credentials that the transport should use to contact the API.

Cluster

Appears in:

- [ExecCredentialSpec](#)

Cluster contains information to allow an exec plugin to communicate with the kubernetes cluster being authenticated to.

To ensure that this struct contains everything someone would need to communicate with a kubernetes cluster (just like they would via a kubeconfig), the fields should shadow "k8s.io/client-go/tools/clientcmd/api/v1".Cluster, with the exception of CertificateAuthority, since CA data will always be passed to the plugin as bytes.

Field	Description
server [Required] string	Server is the address of the kubernetes cluster (https:// hostname:port).
tls-server-name string	TLSServerName is passed to the server for SNI and is used in the client to check server certificates against. If ServerName is empty, the hostname used to contact the server is used.
insecure-skip-tls-verify bool	InsecureSkipTLSVerify skips the validity check for the server's certificate. This will make your HTTPS connections insecure.
certificate-authority-data []byte	CAData contains PEM-encoded certificate authority certificates. If empty, system roots should be used.

Field	Description
proxy-url string	ProxyURL is the URL to the proxy to be used for all requests to this cluster.
disable-compression bool	DisableCompression allows client to opt-out of response compression for all requests to the server. This is useful to speed up requests (specifically lists) when client-server network bandwidth is ample, by saving time on compression (server-side) and decompression (client-side): https://github.com/kubernetes/kubernetes/issues/112296 .
config k8s.io/apimachinery/pkg/runtime.RawExtension	<p>Config holds additional config data that is specific to the exec plugin with regards to the cluster being authenticated to.</p> <p>This data is sourced from the clientcmd Cluster object's extensions[client.authentication.k8s.io/exec] field:</p> <p>clusters:</p> <ul style="list-style-type: none"> • name: my-cluster cluster: ... extensions: <ul style="list-style-type: none"> ◦ name: client.authentication.k8s.io/exec # reserved extension name for per cluster exec config extension: audience: 06e3fbd18de8 # arbitrary config <p>In some environments, the user config may be exactly the same across many clusters (i.e. call this exec plugin) minus some details that are specific to each cluster such as the audience. This field allows the per cluster config to be directly specified with the cluster info. Using this field to store secret data is not recommended as one of the prime benefits of exec plugins is that no secrets need to be stored directly in the kubeconfig.</p>

ExecCredentialSpec

Appears in:

- [ExecCredential](#)

ExecCredentialSpec holds request and runtime specific information provided by the transport.

Field	Description
cluster Cluster	Cluster contains information to allow an exec plugin to communicate with the kubernetes cluster being authenticated to. Note that Cluster is non-nil only when provideClusterInfo is set to true in the exec provider config (i.e., ExecConfig.ProvideClusterInfo).
interactive [Required] bool	Interactive declares whether stdin has been passed to this exec plugin.

ExecCredentialStatus

Appears in:

- [ExecCredential](#)

ExecCredentialStatus holds credentials for the transport to use.

Token and ClientKeyData are sensitive fields. This data should only be transmitted in-memory between client and exec plugin process. Exec plugin itself should at least be protected via file permissions.

Field	Description
expirationTimestamp meta/v1.Time	ExpirationTimestamp indicates a time when the provided credentials expire.
token [Required] string	Token is a bearer token used by the client for request authentication.
clientCertificateData [Required] string	PEM-encoded client TLS certificates (including intermediates, if any).
clientKeyData [Required] string	PEM-encoded private key for the above certificate.

Event Rate Limit Configuration (v1alpha1)

Resource Types

- [Configuration](#)

Configuration

Configuration provides configuration for the EventRateLimit admission controller.

Field	Description
apiVersion string	eventratelimit.admission.k8s.io/v1alpha1
kind string	Configuration
limits [Required] []Limit	limits are the limits to place on event queries received. Limits can be placed on events received server-wide, per namespace, per user, and per source+object. At least one limit is required.

Limit

Appears in:

- [Configuration](#)

Limit is the configuration for a particular limit type

Field	Description
type [Required] LimitType	type is the type of limit to which this configuration applies
qps [Required] int32	qps is the number of event queries per second that are allowed for this type of limit. The qps and burst fields are used together to determine if a particular event query is accepted. The qps determines how many queries are accepted once the burst amount of queries has been exhausted.
burst [Required] int32	burst is the burst number of event queries that are allowed for this type of limit. The qps and burst fields are used together to determine if a particular event query is accepted. The burst determines the maximum size of the allowance granted for a particular bucket. For example, if the burst is 10 and the qps is 3, then the admission control will accept 10 queries before blocking any queries. Every second, 3 more queries will be allowed. If some of that allowance is not used, then it will roll over to the next second, until the maximum allowance of 10 is reached.
cacheSize int32	cacheSize is the size of the LRU cache for this type of limit. If a bucket is evicted from the cache, then the allowance for that bucket is reset. If more queries are later received for an evicted bucket, then that bucket will re-enter the cache with a clean slate, giving that bucket a full allowance of burst queries. The default cache size is 4096. If limitType is 'server', then cacheSize is ignored.

LimitType

(Alias of string)

Appears in:

- [Limit](#)

LimitType is the type of the limit (e.g., per-namespace)

Image Policy API (v1alpha1)

Resource Types

- [ImageReview](#)

ImageReview

ImageReview checks if the set of images in a pod are allowed.

Field	Description
apiVersion string	imagepolicy.k8s.io/v1alpha1
kind string	ImageReview
metadata meta / v1.ObjectMeta	Standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata Refer to the Kubernetes API documentation for the fields of the metadata field.
spec [Required] ImageReviewSpec	Spec holds information about the pod being evaluated
status ImageReviewStatus	Status is filled in by the backend and indicates whether the pod should be allowed.

ImageReviewContainerSpec

Appears in:

- [ImageReviewSpec](#)

ImageReviewContainerSpec is a description of a container within the pod creation request.

Field	Description
image string	This can be in the form image:tag or image@SHA:012345679abcdef.

ImageReviewSpec

Appears in:

- [ImageReview](#)

ImageReviewSpec is a description of the pod creation request.

Field	Description
containers [] ImageReviewContainerSpec	Containers is a list of a subset of the information in each container of the Pod being created.
annotations map[string]string	Annotations is a list of key-value pairs extracted from the Pod's annotations. It only includes keys which match the pattern <code>*.image-policy.k8s.io/*</code> . It is up to each webhook backend to determine how to interpret these annotations, if at all.
namespace string	Namespace is the namespace the pod is being created in.

ImageReviewStatus

Appears in:

- [ImageReview](#)

ImageReviewStatus is the result of the review for the pod creation request.

Field	Description
allowed [Required] bool	Allowed indicates that all images were allowed to be run.
reason string	Reason should be empty unless Allowed is false in which case it may contain a short description of what is wrong. Kubernetes may truncate excessively long errors when displaying to the user.
auditAnnotations map[string]string	AuditAnnotations will be added to the attributes object of the admission controller request using 'AddAnnotation'. The keys should be prefix-less (i.e., the admission controller will add an appropriate prefix).

kube-apiserver Admission (v1)

Resource Types

- [AdmissionReview](#)

AdmissionReview

AdmissionReview describes an admission review request/response.

Field	Description
apiVersion string	admission.k8s.io/v1
kind string	AdmissionReview
request AdmissionRequest	Request describes the attributes for the admission request.
response AdmissionResponse	Response describes the attributes for the admission response.

AdmissionRequest

Appears in:

- [AdmissionReview](#)

AdmissionRequest describes the admission.Attributes for the admission request.

Field	Description
uid [Required] k8s.io/apimachinery/pkg/types.UID	UID is an identifier for the individual request/response. It allows us to distinguish instances of requests which are otherwise identical (parallel requests, requests when earlier requests did not modify etc) The UID is meant to track the round trip (request/response) between the KAS and the WebHook, not the user request. It is suitable for correlating log entries between the webhook and apiserver, for either auditing or debugging.
kind [Required] meta/v1.GroupVersionKind	Kind is the fully-qualified type of object being submitted (for example, v1.Pod or autoscaling.v1.Scale)
resource [Required] meta/v1.GroupVersionResource	Resource is the fully-qualified resource being requested (for example, v1.pods)
subResource string	SubResource is the subresource being requested, if any (for example, "status" or "scale")
requestKind meta/v1.GroupVersionKind	RequestKind is the fully-qualified type of the original API request (for example, v1.Pod or autoscaling.v1.Scale). If this is specified and differs from the value in "kind", an equivalent match and conversion was performed. For example, if deployments can be modified via apps/v1 and apps/v1beta1, and a webhook registered a rule of apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"] and matchPolicy: Equivalent, an API request to apps/v1beta1 deployments would be

Field	Description
	<p>converted and sent to the webhook with kind: {group:"apps", version:"v1", kind:"Deployment"} (matching the rule the webhook registered for), and requestKind: {group:"apps", version:"v1beta1", kind:"Deployment"} (indicating the kind of the original API request).</p> <p>See documentation for the "matchPolicy" field in the webhook configuration type for more details.</p>
requestResource <u>meta/</u> v1.GroupVersionResource	<p>RequestResource is the fully-qualified resource of the original API request (for example, v1.pods). If this is specified and differs from the value in "resource", an equivalent match and conversion was performed.</p> <p>For example, if deployments can be modified via apps/v1 and apps/v1beta1, and a webhook registered a rule of apiGroups:["apps"], apiVersions:["v1"], resources: ["deployments"] and matchPolicy: Equivalent, an API request to apps/v1beta1 deployments would be converted and sent to the webhook with resource: {group:"apps", version:"v1", resource:"deployments"} (matching the resource the webhook registered for), and requestResource: {group:"apps", version:"v1beta1", resource:"deployments"} (indicating the resource of the original API request).</p> <p>See documentation for the "matchPolicy" field in the webhook configuration type.</p>
requestSubResource string	<p>RequestSubResource is the name of the subresource of the original API request, if any (for example, "status" or "scale") If this is specified and differs from the value in "subResource", an equivalent match and conversion was performed. See documentation for the "matchPolicy" field in the webhook configuration type.</p>
name string	<p>Name is the name of the object as presented in the request. On a CREATE operation, the client may omit name and rely on the server to generate the name. If that is the case, this field will contain an empty string.</p>
namespace string	<p>Namespace is the namespace associated with the request (if any).</p>
operation [Required] Operation	<p>Operation is the operation being performed. This may be different than the operation requested. e.g. a patch can result in either a CREATE or UPDATE Operation.</p>
userInfo [Required] authentication/v1.UserInfo	<p>Userinfo is information about the requesting user</p>

Field	Description
object k8s.io/apimachinery/pkg/runtime.RawExtension	Object is the object from the incoming request.
oldObject k8s.io/apimachinery/pkg/runtime.RawExtension	OldObject is the existing object. Only populated for DELETE and UPDATE requests.
dryRun bool	DryRun indicates that modifications will definitely not be persisted for this request. Defaults to false.
options k8s.io/apimachinery/pkg/runtime.RawExtension	Options is the operation option structure of the operation being performed. e.g. meta.k8s.io/v1.DeleteOptions or meta.k8s.io/v1.CreateOptions. This may be different than the options the caller provided. e.g. for a patch request the performed Operation might be a CREATE, in which case the Options will be a meta.k8s.io/v1.CreateOptions even though the caller provided meta.k8s.io/v1.PatchOptions.

AdmissionResponse

Appears in:

- [AdmissionReview](#)

AdmissionResponse describes an admission response.

Field	Description
uid [Required] k8s.io/apimachinery/pkg/types.UID	UID is an identifier for the individual request/response. This must be copied over from the corresponding AdmissionRequest.
allowed [Required] bool	Allowed indicates whether or not the admission request was permitted.
status meta/v1.Status	Result contains extra details into why an admission request was denied. This field IS NOT consulted in any way if "Allowed" is "true".
patch []byte	The patch body. Currently we only support "JSONPatch" which implements RFC 6902.
patchType PatchType	The type of Patch. Currently we only allow "JSONPatch".
auditAnnotations map[string]string	AuditAnnotations is an unstructured key value map set by remote admission controller (e.g. error=image-blacklisted).

Field	Description
	MutatingAdmissionWebhook and ValidatingAdmissionWebhook admission controller will prefix the keys with admission webhook name (e.g. imagepolicy.example.com/error=image-blacklisted). AuditAnnotations will be provided by the admission webhook to add additional context to the audit log for this request.
warnings []string	warnings is a list of warning messages to return to the requesting API client. Warning messages describe a problem the client making the API request should correct or be aware of. Limit warnings to 120 characters if possible. Warnings over 256 characters and large numbers of warnings may be truncated.

Operation

(Alias of string)

Appears in:

- [AdmissionRequest](#)

Operation is the type of resource operation being checked for admission control

PatchType

(Alias of string)

Appears in:

- [AdmissionResponse](#)

PatchType is the type of patch being used to represent the mutated object

kube-apiserver Audit Configuration (v1)

Resource Types

- [Event](#)
- [EventList](#)
- [Policy](#)
- [PolicyList](#)

Event

Appears in:

- [EventList](#)

Event captures all the information that can be included in an API audit log.

Field	Description
apiVersion string	audit.k8s.io/v1
kind string	Event
level [Required] Level	AuditLevel at which event was generated
auditID [Required] k8s.io/apimachinery/pkg/types.UID	Unique audit ID, generated for each request.
stage [Required] Stage	Stage of the request handling when this event instance was generated.
requestURI [Required] string	RequestURI is the request URI as sent by the client to a server.
verb [Required] string	Verb is the kubernetes verb associated with the request. For non-resource requests, this is the lower-cased HTTP method.
user [Required] authentication/v1.UserInfo	Authenticated user information.
impersonatedUser authentication/v1.UserInfo	Impersonated user information.
sourceIPs []string	Source IPs, from where the request originated and intermediate proxies. The source IPs are listed from (in order): <ol style="list-style-type: none">1. X-Forwarded-For request header IPs2. X-Real-Ip header, if not present in the X-Forwarded-For list3. The remote address for the connection, if it doesn't match the last IP in the list up to here (X-Forwarded-For or X-Real-Ip). Note: All but the last IP can be arbitrarily set by the client.
userAgent string	UserAgent records the user agent string reported by the client. Note that the UserAgent is provided by the client, and must not be trusted.
objectRef ObjectReference	Object reference this request is targeted at. Does not apply for List-type requests, or non-resource requests.
responseStatus meta/v1.Status	The response status, populated even when the ResponseObject is not a Status type. For successful responses, this will only include

Field	Description
	the Code and StatusSuccess. For non-status type error responses, this will be auto-populated with the error Message.
requestObject k8s.io/apimachinery/pkg/runtime.Unknown	API object from the request, in JSON format. The RequestObject is recorded as-is in the request (possibly re-encoded as JSON), prior to version conversion, defaulting, admission or merging. It is an external versioned object type, and may not be a valid object on its own. Omitted for non-resource requests. Only logged at Request Level and higher.
responseObject k8s.io/apimachinery/pkg/runtime.Unknown	API object returned in the response, in JSON. The ResponseObject is recorded after conversion to the external type, and serialized as JSON. Omitted for non-resource requests. Only logged at Response Level.
requestReceivedTimestamp meta/v1.MicroTime	Time the request reached the apiserver.
stageTimestamp meta/v1.MicroTime	Time the request reached current audit stage.
annotations map[string]string	Annotations is an unstructured key value map stored with an audit event that may be set by plugins invoked in the request serving chain, including authentication, authorization and admission plugins. Note that these annotations are for the audit event, and do not correspond to the metadata.annotations of the submitted object. Keys should uniquely identify the informing component to avoid name collisions (e.g. podsecuritypolicy.admission.k8s.io/policy). Values should be short. Annotations are included in the Metadata level.

EventList

EventList is a list of audit Events.

Field	Description
apiVersion string	audit.k8s.io/v1
kind string	EventList
metadata meta/v1.ListMeta	No description provided.
items [Required] []Event	No description provided.

Policy

Appears in:

- [PolicyList](#)

Policy defines the configuration of audit logging, and the rules for how different request categories are logged.

Field	Description
apiVersion string	audit.k8s.io/v1
kind string	Policy
metadata meta/v1.ObjectMeta	ObjectMeta is included for interoperability with API infrastructure. Refer to the Kubernetes API documentation for the fields of the metadata field.
rules [Required] []PolicyRule	Rules specify the audit Level a request should be recorded at. A request may match multiple rules, in which case the FIRST matching rule is used. The default audit level is None, but can be overridden by a catch-all rule at the end of the list. PolicyRules are strictly ordered.
omitStages []Stage	OmitStages is a list of stages for which no events are created. Note that this can also be specified per rule in which case the union of both are omitted.
omitManagedFields bool	OmitManagedFields indicates whether to omit the managed fields of the request and response bodies from being written to the API audit log. This is used as a global default - a value of 'true' will omit the managed fields, otherwise the managed fields will be included in the API audit log. Note that this can also be specified per rule in which case the value specified in a rule will override the global default.

PolicyList

PolicyList is a list of audit Policies.

Field	Description
apiVersion string	audit.k8s.io/v1
kind string	PolicyList
metadata meta/v1.ListMeta	No description provided.
	No description provided.

Field	Description
items [Required] []Policy	

GroupResources

Appears in:

- [PolicyRule](#)

GroupResources represents resource kinds in an API group.

Field	Description
group string	Group is the name of the API group that contains the resources. The empty string represents the core API group.
resources []string	<p>Resources is a list of resources this rule applies to.</p> <p>For example: 'pods' matches pods. 'pods/log' matches the log subresource of pods. '*' matches all resources and their subresources. 'pods/*' matches all subresources of pods. '*/scale' matches all scale subresources.</p> <p>If wildcard is present, the validation rule will ensure resources do not overlap with each other.</p> <p>An empty list implies all resources and subresources in this API groups apply.</p>
resourceNames []string	ResourceNames is a list of resource instance names that the policy matches. Using this field requires Resources to be specified. An empty list implies that every instance of the resource is matched.

Level

(Alias of string)

Appears in:

- [Event](#)
- [PolicyRule](#)

Level defines the amount of information logged during auditing

ObjectReference

Appears in:

- [Event](#)

ObjectReference contains enough information to let you inspect or modify the referred object.

Field	Description
resource string	No description provided.
namespace string	No description provided.
name string	No description provided.
uid k8s.io/apimachinery/pkg/types.UID	No description provided.
apiGroup string	APIGroup is the name of the API group that contains the referred object. The empty string represents the core API group.
apiVersion string	APIVersion is the version of the API group that contains the referred object.
resourceVersion string	No description provided.
subresource string	No description provided.

PolicyRule

Appears in:

- [Policy](#)

PolicyRule maps requests based off metadata to an audit Level. Requests must match the rules of every field (an intersection of rules).

Field	Description
level [Required] Level	The Level that requests matching this rule are recorded at.
users []string	The users (by authenticated user name) this rule applies to. An empty list implies every user.
userGroups []string	The user groups this rule applies to. A user is considered matching if it is a member of any of the UserGroups. An empty list implies every user group.
verbs []string	The verbs that match this rule. An empty list implies every verb.

Field	Description
resources [] GroupResources	Resources that this rule matches. An empty list implies all kinds in all API groups.
namespaces []string	Namespaces that this rule matches. The empty string "" matches non-namespaced resources. An empty list implies every namespace.
nonResourceURLs []string	NonResourceURLs is a set of URL paths that should be audited. *'s are allowed, but only as the full, final step in the path. Examples: "/metrics" - Log requests for apiserver metrics "/healthz*" - Log all health checks
omitStages [] Stage	OmitStages is a list of stages for which no events are created. Note that this can also be specified policy wide in which case the union of both are omitted. An empty list means no restrictions will apply.
omitManagedFields bool	OmitManagedFields indicates whether to omit the managed fields of the request and response bodies from being written to the API audit log. <ul style="list-style-type: none"> • a value of 'true' will drop the managed fields from the API audit log • a value of 'false' indicates that the managed fields should be included in the API audit log Note that the value, if specified, in this rule will override the global default If a value is not specified then the global default specified in Policy.OmitManagedFields will stand.

Stage

(Alias of string)

Appears in:

- [Event](#)
- [Policy](#)
- [PolicyRule](#)

Stage defines the stages in request handling that audit events may be generated.

kube-apiserver Configuration (v1)

Package v1 is the v1 version of the API.

Resource Types

- [AdmissionConfiguration](#)

AdmissionConfiguration

AdmissionConfiguration provides versioned configuration for admission controllers.

Field	Description
apiVersion string	apiserver.config.k8s.io/v1
kind string	AdmissionConfiguration
plugins []AdmissionPluginConfiguration	Plugins allows specifying a configuration per admission control plugin.

AdmissionPluginConfiguration

Appears in:

- [AdmissionConfiguration](#)

AdmissionPluginConfiguration provides the configuration for a single plug-in.

Field	Description
name [Required] string	Name is the name of the admission controller. It must match the registered admission plugin name.
path string	Path is the path to a configuration file that contains the plugin's configuration
configuration k8s.io/apimachinery/pkg/runtime.Unknown	Configuration is an embedded configuration object to be used as the plugin's configuration. If present, it will be used instead of the path to the configuration file.

kube-apiserver Configuration (v1alpha1)

Package v1alpha1 is the v1alpha1 version of the API.

Resource Types

- [AdmissionConfiguration](#)
- [EgressSelectorConfiguration](#)
- [TracingConfiguration](#)

TracingConfiguration

Appears in:

- [KubeletConfiguration](#)
- [TracingConfiguration](#)

TracingConfiguration provides versioned configuration for OpenTelemetry tracing clients.

Field	Description
endpoint string	Endpoint of the collector this component will report traces to. The connection is insecure, and does not currently support TLS. Recommended is unset, and endpoint is the otlp grpc default, localhost:4317.
samplingRatePerMillion int32	SamplingRatePerMillion is the number of samples to collect per million spans. Recommended is unset. If unset, sampler respects its parent span's sampling rate, but otherwise never samples.

AdmissionConfiguration

AdmissionConfiguration provides versioned configuration for admission controllers.

Field	Description
apiVersion string	apiserver.k8s.io/v1alpha1
kind string	AdmissionConfiguration
plugins []AdmissionPluginConfiguration	Plugins allows specifying a configuration per admission control plugin.

EgressSelectorConfiguration

EgressSelectorConfiguration provides versioned configuration for egress selector clients.

Field	Description
apiVersion string	apiserver.k8s.io/v1alpha1
kind string	EgressSelectorConfiguration
egressSelections [Required] []EgressSelection	connectionServices contains a list of egress selection client configurations

TracingConfiguration

TracingConfiguration provides versioned configuration for tracing clients.

Field	Description
apiVersion string	apiserver.k8s.io/v1alpha1
kind string	TracingConfiguration
TracingConfiguration [Required] TracingConfiguration	(Members of TracingConfiguration are embedded into this type.) Embed the component config tracing configuration struct

AdmissionPluginConfiguration

Appears in:

- [AdmissionConfiguration](#)

AdmissionPluginConfiguration provides the configuration for a single plug-in.

Field	Description
name [Required] string	Name is the name of the admission controller. It must match the registered admission plugin name.
path string	Path is the path to a configuration file that contains the plugin's configuration
configuration k8s.io/apimachinery/pkg/runtime.Unknown	Configuration is an embedded configuration object to be used as the plugin's configuration. If present, it will be used instead of the path to the configuration file.

Connection

Appears in:

- [EgressSelection](#)

Connection provides the configuration for a single egress selection client.

Field	Description
proxyProtocol [Required] ProtocolType	Protocol is the protocol used to connect from client to the konnectivity server.

Field	Description
transport Transport	Transport defines the transport configurations we use to dial to the konnectivity server. This is required if ProxyProtocol is HTTPConnect or GRPC.

EgressSelection

Appears in:

- [EgressSelectorConfiguration](#)

EgressSelection provides the configuration for a single egress selection client.

Field	Description
name [Required] string	name is the name of the egress selection. Currently supported values are "controlplane", "master", "etcd" and "cluster" The "master" egress selector is deprecated in favor of "controlplane"
connection [Required] Connection	connection is the exact information used to configure the egress selection

ProtocolType

(Alias of string)

Appears in:

- [Connection](#)

ProtocolType is a set of valid values for Connection.ProtocolType

TCPTransport

Appears in:

- [Transport](#)

TCPTransport provides the information to connect to konnectivity server via TCP

Field	Description
url [Required] string	URL is the location of the konnectivity server to connect to. As an example it might be "https://127.0.0.1:8131"

Field	Description
tlsConfig TLSConfig	TLSConfig is the config needed to use TLS when connecting to konnectivity server

TLSConfig

Appears in:

- [TCPTransport](#)

TLSConfig provides the authentication information to connect to konnectivity server Only used with TCPTransport

Field	Description
caBundle string	caBundle is the file location of the CA to be used to determine trust with the konnectivity server. Must be absent/empty if TCPTransport.URL is prefixed with http:// If absent while TCPTransport.URL is prefixed with https://, default to system trust roots.
clientKey string	clientKey is the file location of the client key to be used in mtls handshakes with the konnectivity server. Must be absent/empty if TCPTransport.URL is prefixed with http:// Must be configured if TCPTransport.URL is prefixed with https://
clientCert string	clientCert is the file location of the client certificate to be used in mtls handshakes with the konnectivity server. Must be absent/empty if TCPTransport.URL is prefixed with http:// Must be configured if TCPTransport.URL is prefixed with https://

Transport

Appears in:

- [Connection](#)

Transport defines the transport configurations we use to dial to the konnectivity server

Field	Description
tcp TCPTransport	TCP is the TCP configuration for communicating with the konnectivity server via TCP ProxyProtocol of GRPC is not supported with TCP transport at the moment Requires at least one of TCP or UDS to be set
uds UDSTransport	UDS is the UDS configuration for communicating with the konnectivity server via UDS Requires at least one of TCP or UDS to be set

UDSTransport

Appears in:

- [Transport](#)

UDSTransport provides the information to connect to konnectivity server via UDS

Field	Description
udsName [Required] string	UDSName is the name of the unix domain socket to connect to konnectivity server This does not use a unix:// prefix. (Eg: /etc/srv/kubernetes/konnectivity-server/konnectivity-server.socket)

kube-apiserver Configuration (v1beta1)

Package v1beta1 is the v1beta1 version of the API.

Resource Types

- [EgressSelectorConfiguration](#)
- [TracingConfiguration](#)

TracingConfiguration

Appears in:

- [KubeletConfiguration](#)
- [TracingConfiguration](#)
- [TracingConfiguration](#)

TracingConfiguration provides versioned configuration for OpenTelemetry tracing clients.

Field	Description
endpoint string	Endpoint of the collector this component will report traces to. The connection is insecure, and does not currently support TLS. Recommended is unset, and endpoint is the otlp grpc default, localhost:4317.
samplingRatePerMillion int32	SamplingRatePerMillion is the number of samples to collect per million spans. Recommended is unset. If unset, sampler respects its parent span's sampling rate, but otherwise never samples.

EgressSelectorConfiguration

EgressSelectorConfiguration provides versioned configuration for egress selector clients.

Field	Description
apiVersion string	apiserver.k8s.io/v1beta1
kind string	EgressSelectorConfiguration
egressSelections [Required] []EgressSelection	connectionServices contains a list of egress selection client configurations

TracingConfiguration

TracingConfiguration provides versioned configuration for tracing clients.

Field	Description
apiVersion string	apiserver.k8s.io/v1beta1
kind string	TracingConfiguration
TracingConfiguration [Required] TracingConfiguration	(Members of TracingConfiguration are embedded into this type.) Embed the component config tracing configuration struct

Connection

Appears in:

- [EgressSelection](#)

Connection provides the configuration for a single egress selection client.

Field	Description
proxyProtocol [Required] ProtocolType	Protocol is the protocol used to connect from client to the konnectivity server.
transport Transport	Transport defines the transport configurations we use to dial to the konnectivity server. This is required if ProxyProtocol is HTTPConnect or GRPC.

EgressSelection

Appears in:

- [EgressSelectorConfiguration](#)

EgressSelection provides the configuration for a single egress selection client.

Field	Description
name [Required] string	name is the name of the egress selection. Currently supported values are "controlplane", "master", "etcd" and "cluster" The "master" egress selector is deprecated in favor of "controlplane"
connection [Required] Connection	connection is the exact information used to configure the egress selection

ProtocolType

(Alias of string)

Appears in:

- [Connection](#)

ProtocolType is a set of valid values for Connection.ProtocolType

TCPTransport

Appears in:

- [Transport](#)

TCPTransport provides the information to connect to konnectivity server via TCP

Field	Description
url [Required] string	URL is the location of the konnectivity server to connect to. As an example it might be "https://127.0.0.1:8131"
tlsConfig TLSConfig	TLSConfig is the config needed to use TLS when connecting to konnectivity server

TLSConfig

Appears in:

- [TCPTransport](#)

TLSCConfig provides the authentication information to connect to konnectivity server Only used with TCPTransport

Field	Description
caBundle string	caBundle is the file location of the CA to be used to determine trust with the konnectivity server. Must be absent/empty if TCPTransport.URL is prefixed with http:// If absent while TCPTransport.URL is prefixed with https://, default to system trust roots.
clientKey string	clientKey is the file location of the client key to be used in mtls handshakes with the konnectivity server. Must be absent/empty if TCPTransport.URL is prefixed with http:// Must be configured if TCPTransport.URL is prefixed with https://
clientCert string	clientCert is the file location of the client certificate to be used in mtls handshakes with the konnectivity server. Must be absent/empty if TCPTransport.URL is prefixed with http:// Must be configured if TCPTransport.URL is prefixed with https://

Transport

Appears in:

- [Connection](#)

Transport defines the transport configurations we use to dial to the konnectivity server

Field	Description
tcp TCPTransport	TCP is the TCP configuration for communicating with the konnectivity server via TCP ProxyProtocol of GRPC is not supported with TCP transport at the moment Requires at least one of TCP or UDS to be set
uds UDSTransport	UDS is the UDS configuration for communicating with the konnectivity server via UDS Requires at least one of TCP or UDS to be set

UDSTransport

Appears in:

- [Transport](#)

UDSTransport provides the information to connect to konnectivity server via UDS

Field	Description
udsName [Required] string	

Field	Description
	UDSName is the name of the unix domain socket to connect to konnectivity server This does not use a unix:// prefix. (Eg: /etc/srv/kubernetes/konnectivity-server/konnectivity-server.socket)

kube-apiserver Encryption Configuration (v1)

Package v1 is the v1 version of the API.

Resource Types

- [EncryptionConfiguration](#)

EncryptionConfiguration

EncryptionConfiguration stores the complete configuration for encryption providers. It also allows the use of wildcards to specify the resources that should be encrypted. Use '*'<group> to encrypt all resources within a group or '**' to encrypt all resources. '*' can be used to encrypt all resource in the core group. '**' will encrypt all resources, even custom resources that are added after API server start. Use of wildcards that overlap within the same resource list or across multiple entries are not allowed since part of the configuration would be ineffective. Resource lists are processed in order, with earlier lists taking precedence.

Example:

```
kind: EncryptionConfiguration
apiVersion: apiserver.config.k8s.io/v1
resources:
- resources:
  - events
  providers:
  - identity: {} # do not encrypt events even though ** is specified below
- resources:
  - secrets
  - configmaps
  - pandas.awesome.bears.example
  providers:
  - aescbc:
    keys:
    - name: key1
      secret: c2VjcmV0IGlzIHNIY3VyZQ==
- resources:
  - '*.apps'
  providers:
  - aescbc:
    keys:
```

```

- name: key2
  secret: c2VjcmV0IGlzIHNIY3VyZSwgb3IgaXMgaXQ/Cg==
- resources:
  - '*'
providers:
- aescbc:
  keys:
  - name: key3
    secret: c2VjcmV0IGlzIHNIY3VyZSwgSSB0aGluaw==

```

Field	Description
apiVersion string	apiserver.config.k8s.io/v1
kind string	EncryptionConfiguration
resources [Required] []ResourceConfiguration	resources is a list containing resources, and their corresponding encryption providers.

AESConfiguration

Appears in:

- [ProviderConfiguration](#)

AESConfiguration contains the API configuration for an AES transformer.

Field	Description
keys [Required] []Key	keys is a list of keys to be used for creating the AES transformer. Each key has to be 32 bytes long for AES-CBC and 16, 24 or 32 bytes for AES-GCM.

IdentityConfiguration

Appears in:

- [ProviderConfiguration](#)

IdentityConfiguration is an empty struct to allow identity transformer in provider configuration.

KMSConfiguration

Appears in:

- [ProviderConfiguration](#)

KMSConfiguration contains the name, cache size and path to configuration file for a KMS based envelope transformer.

Field	Description
apiVersion string	apiVersion of KeyManagementService
name [Required] string	name is the name of the KMS plugin to be used.
cachesize int32	cachesize is the maximum number of secrets which are cached in memory. The default value is 1000. Set to a negative value to disable caching. This field is only allowed for KMS v1 providers.
endpoint [Required] string	endpoint is the gRPC server listening address, for example "unix:///var/run/kms-provider.sock".
timeout meta/v1.Duration	timeout for gRPC calls to kms-plugin (ex. 5s). The default is 3 seconds.

Key

Appears in:

- [AESConfiguration](#)
- [SecretboxConfiguration](#)

Key contains name and secret of the provided key for a transformer.

Field	Description
name [Required] string	name is the name of the key to be used while storing data to disk.
secret [Required] string	secret is the actual key, encoded in base64.

ProviderConfiguration

Appears in:

- [ResourceConfiguration](#)

ProviderConfiguration stores the provided configuration for an encryption provider.

Field	Description
aesgcm [Required] AESConfiguration	aesgcm is the configuration for the AES-GCM transformer.

Field	Description
aescbc [Required] AESConfiguration	aescbc is the configuration for the AES-CBC transformer.
secretbox [Required] SecretboxConfiguration	secretbox is the configuration for the Secretbox based transformer.
identity [Required] IdentityConfiguration	identity is the (empty) configuration for the identity transformer.
kms [Required] KMSConfiguration	kms contains the name, cache size and path to configuration file for a KMS based envelope transformer.

ResourceConfiguration

Appears in:

- [EncryptionConfiguration](#)

ResourceConfiguration stores per resource configuration.

Field	Description
resources [Required] []string	resources is a list of kubernetes resources which have to be encrypted. The resource names are derived from resource or resource.group of the group/version/resource. eg: pandas.awesome.bears.example is a custom resource with 'group': awesome.bears.example, 'resource': pandas. Use '*' to encrypt all resources and '*.<group>' to encrypt all resources in a specific group. eg: '*.awesome.bears.example' will encrypt all resources in the group 'awesome.bears.example'. eg: '*' will encrypt all resources in the core group (such as pods, configmaps, etc).
providers [Required] []ProviderConfiguration	providers is a list of transformers to be used for reading and writing the resources to disk. eg: aesgcm, aescbc, secretbox, identity, kms.

SecretboxConfiguration

Appears in:

- [ProviderConfiguration](#)

SecretboxConfiguration contains the API configuration for an Secretbox transformer.

Field	Description
keys [Required] []Key	keys is a list of keys to be used for creating the Secretbox transformer. Each key has to be 32 bytes long.

kube-controller-manager Configuration (v1alpha1)

Resource Types

- [CloudControllerManagerConfiguration](#)
- [LeaderMigrationConfiguration](#)
- [KubeControllerManagerConfiguration](#)

NodeControllerConfiguration

Appears in:

- [CloudControllerManagerConfiguration](#)

NodeControllerConfiguration contains elements describing NodeController.

Field	Description
ConcurrentNodeSyncs [Required] int32	ConcurrentNodeSyncs is the number of workers concurrently synchronizing nodes

ServiceControllerConfiguration

Appears in:

- [CloudControllerManagerConfiguration](#)
- [KubeControllerManagerConfiguration](#)

ServiceControllerConfiguration contains elements describing ServiceController.

Field	Description
ConcurrentServiceSyncs [Required] int32	concurrentServiceSyncs is the number of services that are allowed to sync concurrently. Larger number = more responsive service management, but more CPU (and network) load.

CloudControllerManagerConfiguration

CloudControllerManagerConfiguration contains elements describing cloud-controller manager.

Field	Description
apiVersion string	cloudcontrollermanager.config.k8s.io/v1alpha1 CloudControllerManagerConfiguration

Field	Description
kind string	
Generic [Required] GenericControllerManagerConfiguration	Generic holds configuration for a generic controller-manager
KubeCloudShared [Required] KubeCloudSharedConfiguration	KubeCloudSharedConfiguration holds configuration for shared related features both in cloud controller manager and kube-controller manager.
NodeController [Required] NodeControllerConfiguration	NodeController holds configuration for node controller related features.
ServiceController [Required] ServiceControllerConfiguration	ServiceControllerConfiguration holds configuration for ServiceController related features.
NodeStatusUpdateFrequency [Required] meta/v1.Duration	NodeStatusUpdateFrequency is the frequency at which the controller updates nodes' status
Webhook [Required] WebhookConfiguration	Webhook is the configuration for cloud-controller-manager hosted webhooks

CloudProviderConfiguration

Appears in:

- [KubeCloudSharedConfiguration](#)

CloudProviderConfiguration contains basically elements about cloud provider.

Field	Description
Name [Required] string	Name is the provider for cloud services.
CloudConfigFile [Required] string	cloudConfigFile is the path to the cloud provider configuration file.

KubeCloudSharedConfiguration

Appears in:

- [CloudControllerManagerConfiguration](#)

[KubeControllerManagerConfiguration](#)

- KubeCloudSharedConfiguration contains elements shared by both kube-controller manager and cloud-controller manager, but not genericconfig.

Field	Description
CloudProvider [Required] CloudProviderConfiguration	CloudProviderConfiguration holds configuration for CloudProvider related features.
ExternalCloudVolumePlugin [Required] string	externalCloudVolumePlugin specifies the plugin to use when cloudProvider is "external". It is currently used by the in repo cloud providers to handle node and volume control in the KCM.
UseServiceAccountCredentials [Required] bool	useServiceAccountCredentials indicates whether controllers should be run with individual service account credentials.
AllowUntaggedCloud [Required] bool	run with untagged cloud instances
RouteReconciliationPeriod [Required] meta/v1.Duration	routeReconciliationPeriod is the period for reconciling routes created for Nodes by cloud provider..
NodeMonitorPeriod [Required] meta/v1.Duration	nodeMonitorPeriod is the period for syncing NodeStatus in NodeController.
ClusterName [Required] string	clusterName is the instance prefix for the cluster.
ClusterCIDR [Required] string	clusterCIDR is CIDR Range for Pods in cluster.
AllocateNodeCIDRs [Required] bool	AllocateNodeCIDRs enables CIDRs for Pods to be allocated and, if ConfigureCloudRoutes is true, to be set on the cloud provider.
CIDRAlocatorType [Required] string	CIDRAlocatorType determines what kind of pod CIDR allocator will be used.
ConfigureCloudRoutes [Required] bool	configureCloudRoutes enables CIDRs allocated with allocateNodeCIDRs to be configured on the cloud provider.
NodeSyncPeriod [Required] meta/v1.Duration	nodeSyncPeriod is the period for syncing nodes from cloudprovider. Longer periods will result in fewer calls to

Field	Description
	cloud provider, but may delay addition of new nodes to cluster.

WebhookConfiguration

Appears in:

- [CloudControllerManagerConfiguration](#)

WebhookConfiguration contains configuration related to cloud-controller-manager hosted webhooks

Field	Description
Webhooks [Required] []string	Webhooks is the list of webhooks to enable or disable '*' means "all enabled by default webhooks" 'foo' means "enable 'foo'" '-foo' means "disable 'foo'" first item for a particular name wins

LeaderMigrationConfiguration

Appears in:

- [GenericControllerManagerConfiguration](#)

LeaderMigrationConfiguration provides versioned configuration for all migrating leader locks.

Field	Description
apiVersion string	controllermanager.config.k8s.io/v1alpha1
kind string	LeaderMigrationConfiguration
leaderName [Required] string	LeaderName is the name of the leader election resource that protects the migration E.g. 1-20-KCM-to-1-21-CCM
resourceLock [Required] string	ResourceLock indicates the resource object type that will be used to lock Should be "leases" or "endpoints"
controllerLeaders [Required] []ControllerLeaderConfiguration	ControllerLeaders contains a list of migrating leader lock configurations

ControllerLeaderConfiguration

Appears in:

- [LeaderMigrationConfiguration](#)

ControllerLeaderConfiguration provides the configuration for a migrating leader lock.

Field	Description
name [Required] string	Name is the name of the controller being migrated E.g. service-controller, route-controller, cloud-node-controller, etc
component [Required] string	Component is the name of the component in which the controller should be running. E.g. kube-controller-manager, cloud-controller-manager, etc Or '*' meaning the controller can be run under any component that participates in the migration

GenericControllerManagerConfiguration

Appears in:

- [CloudControllerManagerConfiguration](#)
- [KubeControllerManagerConfiguration](#)

GenericControllerManagerConfiguration holds configuration for a generic controller-manager.

Field	Description
Port [Required] int32	port is the port that the controller-manager's http service runs on.
Address [Required] string	address is the IP address to serve on (set to 0.0.0.0 for all interfaces).
MinResyncPeriod [Required] meta/v1.Duration	minResyncPeriod is the resync period in reflectors; will be random between minResyncPeriod and 2*minResyncPeriod.
ClientConnection [Required] ClientConnectionConfiguration	ClientConnection specifies the kubeconfig file and client connection settings for the proxy server to use when communicating with the apiserver.
ControllerStartInterval [Required] meta/v1.Duration	How long to wait between starting controller managers

Field	Description
LeaderElection [Required] LeaderElectionConfiguration	leaderElection defines the configuration of leader election client.
Controllers [Required] []string	Controllers is the list of controllers to enable or disable '*' means "all enabled by default controllers" 'foo' means "enable 'foo'" '-foo' means "disable 'foo'" first item for a particular name wins
Debugging [Required] DebuggingConfiguration	DebuggingConfiguration holds configuration for Debugging related features.
LeaderMigrationEnabled [Required] bool	LeaderMigrationEnabled indicates whether Leader Migration should be enabled for the controller manager.
LeaderMigration [Required] LeaderMigrationConfiguration	LeaderMigration holds the configuration for Leader Migration.

KubeControllerManagerConfiguration

KubeControllerManagerConfiguration contains elements describing kube-controller manager.

Field	Description
apiVersion string	kubecontrollermanager.config.k8s.io/v1alpha1
kind string	KubeControllerManagerConfiguration
Generic [Required] GenericControllerManagerConfiguration	Generic holds configuration for a generic controller manager
KubeCloudShared [Required] KubeCloudSharedConfiguration	KubeCloudSharedConfiguration holds configuration shared related features both in cloud controller manager and kube-controller manager.
AttachDetachController [Required] AttachDetachControllerConfiguration	AttachDetachControllerConfiguration holds configuration for AttachDetachController related features.
CSRSigningController [Required] CSRSigningControllerConfiguration	CSRSigningControllerConfiguration holds configuration for CSRSigningController related features.

Field	Description
DaemonSetController [Required] DaemonSetControllerConfiguration	DaemonSetControllerConfiguration holds configuration for DaemonSetController related features.
DeploymentController [Required] DeploymentControllerConfiguration	DeploymentControllerConfiguration holds configuration for DeploymentController related features.
StatefulSetController [Required] StatefulSetControllerConfiguration	StatefulSetControllerConfiguration holds configuration for StatefulSetController related features.
DeprecatedController [Required] DeprecatedControllerConfiguration	DeprecatedControllerConfiguration holds configuration for some deprecated features.
EndpointController [Required] EndpointControllerConfiguration	EndpointControllerConfiguration holds configuration for EndpointController related features.
EndpointSliceController [Required] EndpointSliceControllerConfiguration	EndpointSliceControllerConfiguration holds configuration for EndpointSliceController related features.
EndpointSliceMirroringController [Required] EndpointSliceMirroringControllerConfiguration	EndpointSliceMirroringControllerConfiguration holds configuration for EndpointSliceMirroringController related features.
EphemeralVolumeController [Required] EphemeralVolumeControllerConfiguration	EphemeralVolumeControllerConfiguration holds configuration for EphemeralVolumeController related features.
GarbageCollectorController [Required] GarbageCollectorControllerConfiguration	GarbageCollectorControllerConfiguration holds configuration for GarbageCollectorController related features.
HPAController [Required] HPAControllerConfiguration	HPAControllerConfiguration holds configuration for HPAController related features.
JobController [Required] JobControllerConfiguration	JobControllerConfiguration holds configuration for JobController related features.
CronJobController [Required] CronJobControllerConfiguration	CronJobControllerConfiguration holds configuration for CronJobController related features.

Field	Description
LegacySATokenCleaner [Required] LegacySATokenCleanerConfiguration	LegacySATokenCleanerConfiguration holds configuration for LegacySATokenCleaner related features.
NamespaceController [Required] NamespaceControllerConfiguration	NamespaceControllerConfiguration holds configuration for NamespaceController related features.
NodeIPAMController [Required] NodeIPAMControllerConfiguration	NodeIPAMControllerConfiguration holds configuration for NodeIPAMController related features.
NodeLifecycleController [Required] NodeLifecycleControllerConfiguration	NodeLifecycleControllerConfiguration holds configuration for NodeLifecycleController related features.
PersistentVolumeBinderController [Required] PersistentVolumeBinderControllerConfiguration	PersistentVolumeBinderControllerConfiguration holds configuration for PersistentVolumeBinderController related features.
PodGCController [Required] PodGCControllerConfiguration	PodGCControllerConfiguration holds configuration for PodGCController related features.
ReplicaSetController [Required] ReplicaSetControllerConfiguration	ReplicaSetControllerConfiguration holds configuration for ReplicaSet related features.
ReplicationController [Required] ReplicationControllerConfiguration	ReplicationControllerConfiguration holds configuration for ReplicationController related features.
ResourceQuotaController [Required] ResourceQuotaControllerConfiguration	ResourceQuotaControllerConfiguration holds configuration for ResourceQuotaController related features.
SAController [Required] SAControllerConfiguration	SAControllerConfiguration holds configuration for ServiceAccountController related features.
ServiceController [Required] ServiceControllerConfiguration	ServiceControllerConfiguration holds configuration for ServiceController related features.
TTLAfterFinishedController [Required] TTLAfterFinishedControllerConfiguration	TTLAfterFinishedControllerConfiguration holds configuration for TTLAfterFinishedController related features.

Field	Description
ValidatingAdmissionPolicyStatusController [Required] ValidatingAdmissionPolicyStatusControllerConfiguration	ValidatingAdmissionPolicyStatusControllerConfig holds configuration for ValidatingAdmissionPolicyStatusController related features.

AttachDetachControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

AttachDetachControllerConfiguration contains elements describing AttachDetachController.

Field	Description
DisableAttachDetachReconcilerSync [Required] bool	Reconciler runs a periodic loop to reconcile the desired state of the with the actual state of the world by triggering attach detach operations. This flag enables or disables reconcile. Is false by default, and thus enabled.
ReconcilerSyncLoopPeriod [Required] meta/v1.Duration	ReconcilerSyncLoopPeriod is the amount of time the reconciler sync states loop wait between successive executions. Is set to 5 sec by default.

CSRSigningConfiguration

Appears in:

- [CSRSigningControllerConfiguration](#)

CSRSigningConfiguration holds information about a particular CSR signer

Field	Description
CertFile [Required] string	certFile is the filename containing a PEM-encoded X509 CA certificate used to issue certificates
KeyFile [Required] string	keyFile is the filename containing a PEM-encoded RSA or ECDSA private key used to issue certificates

CSRSigningControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

CSRSigningControllerConfiguration contains elements describing CSRSigningController.

Field	Description
ClusterSigningCertFile [Required] string	clusterSigningCertFile is the filename containing a PEM-encoded X509 CA certificate used to issue cluster-scoped certificates
ClusterSigningKeyFile [Required] string	clusterSigningKeyFile is the filename containing a PEM-encoded RSA or ECDSA private key used to issue cluster-scoped certificates
KubeletServingSignerConfiguration [Required] CSRSigningConfiguration	kubeletServingSignerConfiguration holds the certificate and key used to issue certificates for the kubernetes.io/kubelet-serving signer
KubeletClientSignerConfiguration [Required] CSRSigningConfiguration	kubeletClientSignerConfiguration holds the certificate and key used to issue certificates for the kubernetes.io/kube-apiserver-client-kubelet
KubeAPIServerClientSignerConfiguration [Required] CSRSigningConfiguration	kubeAPIServerClientSignerConfiguration holds the certificate and key used to issue certificates for the kubernetes.io/kube-apiserver-client
LegacyUnknownSignerConfiguration [Required] CSRSigningConfiguration	legacyUnknownSignerConfiguration holds the certificate and key used to issue certificates for the kubernetes.io/legacy-unknown
ClusterSigningDuration [Required] meta/v1.Duration	clusterSigningDuration is the max length of duration signed certificates will be given. Individual CSRs may request shorter certs by setting spec.expirationSeconds.

CronJobControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

CronJobControllerConfiguration contains elements describing CronJob2Controller.

Field	Description
ConcurrentCronJobSyncs [Required] int32	concurrentCronJobSyncs is the number of job objects that are allowed to sync concurrently. Larger number = more responsive jobs, but more CPU (and network) load.

DaemonSetControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

DaemonSetControllerConfiguration contains elements describing DaemonSetController.

Field	Description
ConcurrentDaemonSetSyncs [Required] int32	concurrentDaemonSetSyncs is the number of daemonset objects that are allowed to sync concurrently. Larger number = more responsive daemonset, but more CPU (and network) load.

DeploymentControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

DeploymentControllerConfiguration contains elements describing DeploymentController.

Field	Description
ConcurrentDeploymentSyncs [Required] int32	concurrentDeploymentSyncs is the number of deployment objects that are allowed to sync concurrently. Larger number = more responsive deployments, but more CPU (and network) load.

DeprecatedControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

DeprecatedControllerConfiguration contains elements be deprecated.

EndpointControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

EndpointControllerConfiguration contains elements describing EndpointController.

Field	Description
ConcurrentEndpointSyncs [Required] int32	concurrentEndpointSyncs is the number of endpoint syncing operations that will be done concurrently. Larger number = faster endpoint updating, but more CPU (and network) load.
EndpointUpdatesBatchPeriod [Required] meta/v1.Duration	EndpointUpdatesBatchPeriod describes the length of endpoint updates batching period. Processing of pod changes will be delayed by this duration to join them with potential upcoming updates and reduce the overall number of endpoints updates.

EndpointSliceControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

EndpointSliceControllerConfiguration contains elements describing EndpointSliceController.

Field	Description
ConcurrentServiceEndpointSyncs [Required] int32	concurrentServiceEndpointSyncs is the number of service endpoint syncing operations that will be done concurrently. Larger number = faster endpoint slice updating, but more CPU (and network) load.
MaxEndpointsPerSlice [Required] int32	maxEndpointsPerSlice is the maximum number of endpoints that will be added to an EndpointSlice. More endpoints per slice will result in fewer and larger endpoint slices, but larger resources.
EndpointUpdatesBatchPeriod [Required] meta/v1.Duration	EndpointUpdatesBatchPeriod describes the length of endpoint updates batching period. Processing of pod changes will be delayed by this duration to join them with potential upcoming updates and reduce the overall number of endpoints updates.

EndpointSliceMirroringControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

EndpointSliceMirroringControllerConfiguration contains elements describing EndpointSliceMirroringController.

Field	Description
MirroringConcurrentServiceEndpointSyncs [Required] int32	mirroringConcurrentServiceEndpointSyncs is the number of service endpoint syncing operations that will be done concurrently. Larger number = faster endpoint slice updating, but more CPU (and network) load.
MirroringMaxEndpointsPerSubset [Required] int32	mirroringMaxEndpointsPerSubset is the maximum number of endpoints that will be mirrored to an EndpointSlice for an EndpointSubset.
MirroringEndpointUpdatesBatchPeriod [Required] meta/v1.Duration	mirroringEndpointUpdatesBatchPeriod can be used to batch EndpointSlice updates. All updates triggered by EndpointSlice changes will be delayed by up to 'mirroringEndpointUpdatesBatchPeriod'. If other addresses in the same Endpoints resource change in that period, they will be batched to a single EndpointSlice update. Default 0 value means that each Endpoints update triggers an EndpointSlice update.

EphemeralVolumeControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

EphemeralVolumeControllerConfiguration contains elements describing EphemeralVolumeController.

Field	Description
ConcurrentEphemeralVolumeSyncs [Required] int32	ConcurrentEphemeralVolumeSyncs is the number of ephemeral volume syncing operations that will be done concurrently. Larger number = faster ephemeral volume updating, but more CPU (and network) load.

GarbageCollectorControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

GarbageCollectorControllerConfiguration contains elements describing GarbageCollectorController.

Field	Description
EnableGarbageCollector [Required] bool	enables the generic garbage collector. MUST be synced with the corresponding flag of the kube-apiserver. WARNING: the generic garbage collector is an alpha feature.
ConcurrentGCSyncs [Required] int32	concurrentGCSyncs is the number of garbage collector workers that are allowed to sync concurrently.
GCIgnoredResources [Required] []GroupResource	gCIgnoredResources is the list of GroupResources that garbage collection should ignore.

GroupResource

Appears in:

- [GarbageCollectorControllerConfiguration](#)

GroupResource describes an group resource.

Field	Description
Group [Required] string	group is the group portion of the GroupResource.
Resource [Required] string	resource is the resource portion of the GroupResource.

HPAControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

HPAControllerConfiguration contains elements describing HPAController.

Field	Description
ConcurrentHorizontalPodAutoscalerSyncs [Required] int32	ConcurrentHorizontalPodAutoscalerSyncs is the number of HPA objects that are allowed to sync

Field	Description
	concurrently. Larger number = more responsive HPA processing, but more CPU (and network) load.
HorizontalPodAutoscalerSyncPeriod [Required] meta/v1.Duration	HorizontalPodAutoscalerSyncPeriod is the period for syncing the number of pods in horizontal pod autoscaler.
HorizontalPodAutoscalerUpscaleForbiddenWindow [Required] meta/v1.Duration	HorizontalPodAutoscalerUpscaleForbiddenWindow is a period after which next upscale allowed.
HorizontalPodAutoscalerDownscaleStabilizationWindow [Required] meta/v1.Duration	HorizontalPodAutoscalerDownscaleStabilizationWindow is a period for which autoscaler will look backward and not scale down below any recommendation it makes during that period.
HorizontalPodAutoscalerDownscaleForbiddenWindow [Required] meta/v1.Duration	HorizontalPodAutoscalerDownscaleForbiddenWindow is a period after which next downscale allowed.
HorizontalPodAutoscalerTolerance [Required] float64	HorizontalPodAutoscalerTolerance is the tolerance value when resource usage suggests upscaling/downscaling.
HorizontalPodAutoscalerCPUInitializationPeriod [Required] meta/v1.Duration	HorizontalPodAutoscalerCPUInitializationPeriod is a period after pod start when CPU samples might be skipped.
HorizontalPodAutoscalerInitialReadinessDelay [Required] meta/v1.Duration	HorizontalPodAutoscalerInitialReadinessDelay is a period after pod start during which readiness changes are treated as readiness being set for the first time. The effect of this is that HPA will disregard CPU samples from unready pods that had last readiness change during that period.

JobControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

JobControllerConfiguration contains elements describing JobController.

Field	Description
ConcurrentJobSyncs [Required] int32	

Field	Description
	concurrentJobSyncs is the number of job objects that are allowed to sync concurrently. Larger number = more responsive jobs, but more CPU (and network) load.

LegacySATokenCleanerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

LegacySATokenCleanerConfiguration contains elements describing LegacySATokenCleaner

Field	Description
CleanUpPeriod [Required] meta/v1.Duration	CleanUpPeriod is the period of time since the last usage of an auto-generated service account token before it can be deleted.

NamespaceControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

NamespaceControllerConfiguration contains elements describing NamespaceController.

Field	Description
NamespaceSyncPeriod [Required] meta/v1.Duration	namespaceSyncPeriod is the period for syncing namespace life-cycle updates.
ConcurrentNamespaceSyncs [Required] int32	concurrentNamespaceSyncs is the number of namespace objects that are allowed to sync concurrently.

NodeIPAMControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

NodeIPAMControllerConfiguration contains elements describing NodeIpamController.

Field	Description
ServiceCIDR [Required] string	serviceCIDR is CIDR Range for Services in cluster.

Field	Description
SecondaryServiceCIDR [Required] string	secondaryServiceCIDR is CIDR Range for Services in cluster. This is used in dual stack clusters. SecondaryServiceCIDR must be of different IP family than ServiceCIDR
NodeCIDRMaskSize [Required] int32	NodeCIDRMaskSize is the mask size for node cidr in cluster.
NodeCIDRMaskSizeIPv4 [Required] int32	NodeCIDRMaskSizeIPv4 is the mask size for node cidr in dual-stack cluster.
NodeCIDRMaskSizeIPv6 [Required] int32	NodeCIDRMaskSizeIPv6 is the mask size for node cidr in dual-stack cluster.

NodeLifecycleControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

NodeLifecycleControllerConfiguration contains elements describing NodeLifecycleController.

Field	Description
NodeEvictionRate [Required] float32	nodeEvictionRate is the number of nodes per second on which pods are deleted in case of node failure when a zone is healthy
SecondaryNodeEvictionRate [Required] float32	secondaryNodeEvictionRate is the number of nodes per second on which pods are deleted in case of node failure when a zone is unhealthy
NodeStartupGracePeriod [Required] meta/v1.Duration	nodeStartupGracePeriod is the amount of time which we allow starting a node to be unresponsive before marking it unhealthy.
NodeMonitorGracePeriod [Required] meta/v1.Duration	nodeMontiorGracePeriod is the amount of time which we allow a running node to be unresponsive before marking it unhealthy. Must be N times more than kubelet's nodeStatusUpdateFrequency, where N means number of retries allowed for kubelet to post node status.
PodEvictionTimeout [Required] meta/v1.Duration	podEvictionTimeout is the grace period for deleting pods on failed nodes.

Field	Description
LargeClusterSizeThreshold [Required] int32	secondaryNodeEvictionRate is implicitly overridden to 0 for clusters smaller than or equal to largeClusterSizeThreshold
UnhealthyZoneThreshold [Required] float32	Zone is treated as unhealthy in nodeEvictionRate and secondaryNodeEvictionRate when at least unhealthyZoneThreshold (no less than 3) of Nodes in the zone are NotReady

PersistentVolumeBinderControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

PersistentVolumeBinderControllerConfiguration contains elements describing PersistentVolumeBinderController.

Field	Description
PVClaimBinderSyncPeriod [Required] meta/v1.Duration	pvClaimBinderSyncPeriod is the period for syncing persistent volumes and persistent volume claims.
VolumeConfiguration [Required] VolumeConfiguration	volumeConfiguration holds configuration for volume related features.
VolumeHostCIDRDenylist [Required] []string	DEPRECATED: VolumeHostCIDRDenylist is a list of CIDRs that should not be reachable by the controller from plugins.
VolumeHostAllowLocalLoopback [Required] bool	DEPRECATED: VolumeHostAllowLocalLoopback indicates if local loopback hosts (127.0.0.1, etc) should be allowed from plugins.

PersistentVolumeRecyclerConfiguration

Appears in:

- [VolumeConfiguration](#)

PersistentVolumeRecyclerConfiguration contains elements describing persistent volume plugins.

Field	Description
MaximumRetry [Required] int32	maximumRetry is number of retries the PV recycler will execute on failure to recycle PV.
MinimumTimeoutNFS [Required] int32	minimumTimeoutNFS is the minimum ActiveDeadlineSeconds to use for an NFS Recycler pod.
PodTemplateFilePathNFS [Required] string	podTemplateFilePathNFS is the file path to a pod definition used as a template for NFS persistent volume recycling
IncrementTimeoutNFS [Required] int32	incrementTimeoutNFS is the increment of time added per Gi to ActiveDeadlineSeconds for an NFS scrubber pod.
PodTemplateFilePathHostPath [Required] string	podTemplateFilePathHostPath is the file path to a pod definition used as a template for HostPath persistent volume recycling. This is for development and testing only and will not work in a multi-node cluster.
MinimumTimeoutHostPath [Required] int32	minimumTimeoutHostPath is the minimum ActiveDeadlineSeconds to use for a HostPath Recycler pod. This is for development and testing only and will not work in a multi-node cluster.
IncrementTimeoutHostPath [Required] int32	incrementTimeoutHostPath is the increment of time added per Gi to ActiveDeadlineSeconds for a HostPath scrubber pod. This is for development and testing only and will not work in a multi-node cluster.

PodGCControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

PodGCControllerConfiguration contains elements describing PodGCController.

Field	Description
TerminatedPodGCThreshold [Required] int32	terminatedPodGCThreshold is the number of terminated pods that can exist before the terminated pod garbage collector starts deleting terminated pods. If <= 0, the terminated pod garbage collector is disabled.

ReplicaSetControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

ReplicaSetControllerConfiguration contains elements describing ReplicaSetController.

Field	Description
ConcurrentRSSyncs [Required] int32	concurrentRSSyncs is the number of replica sets that are allowed to sync concurrently. Larger number = more responsive replica management, but more CPU (and network) load.

ReplicationControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

ReplicationControllerConfiguration contains elements describing ReplicationController.

Field	Description
ConcurrentRCSyncs [Required] int32	concurrentRCSyncs is the number of replication controllers that are allowed to sync concurrently. Larger number = more responsive replica management, but more CPU (and network) load.

ResourceQuotaControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

ResourceQuotaControllerConfiguration contains elements describing ResourceQuotaController.

Field	Description
ResourceQuotaSyncPeriod [Required] meta/v1.Duration	resourceQuotaSyncPeriod is the period for syncing quota usage status in the system.
ConcurrentResourceQuotaSyncs [Required] int32	concurrentResourceQuotaSyncs is the number of resource quotas that are allowed to sync concurrently. Larger number = more responsive quota management, but more CPU (and network) load.

SAControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

SAControllerConfiguration contains elements describing ServiceAccountController.

Field	Description
ServiceAccountKeyFile [Required] string	serviceAccountKeyFile is the filename containing a PEM-encoded private RSA key used to sign service account tokens.
ConcurrentSATokenSyncs [Required] int32	concurrentSATokenSyncs is the number of service account token syncing operations that will be done concurrently.
RootCAFile [Required] string	rootCAFile is the root certificate authority will be included in service account's token secret. This must be a valid PEM-encoded CA bundle.

StatefulSetControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

StatefulSetControllerConfiguration contains elements describing StatefulSetController.

Field	Description
ConcurrentStatefulSetSyncs [Required] int32	concurrentStatefulSetSyncs is the number of statefulset objects that are allowed to sync concurrently. Larger number = more responsive statefulsets, but more CPU (and network) load.

TTLAfterFinishedControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

TTLAfterFinishedControllerConfiguration contains elements describing TTLAfterFinishedController.

Field	Description
ConcurrentTTLSyncs [Required] int32	concurrentTTLSyncs is the number of TTL-after-finished collector workers that are allowed to sync concurrently.

ValidatingAdmissionPolicyStatusControllerConfiguration

Appears in:

- [KubeControllerManagerConfiguration](#)

ValidatingAdmissionPolicyStatusControllerConfiguration contains elements describing ValidatingAdmissionPolicyStatusController.

Field	Description
ConcurrentPolicySyncs [Required] int32	ConcurrentPolicySyncs is the number of policy objects that are allowed to sync concurrently. Larger number = quicker type checking, but more CPU (and network) load. The default value is 5.

VolumeConfiguration

Appears in:

- [PersistentVolumeBinderControllerConfiguration](#)

VolumeConfiguration contains *all* enumerated flags meant to configure all volume plugins. From this config, the controller-manager binary will create many instances of volume.VolumeConfig, each containing only the configuration needed for that plugin which are then passed to the appropriate plugin. The ControllerManager binary is the only part of the code which knows what plugins are supported and which flags correspond to each plugin.

Field	Description
EnableHostPathProvisioning [Required] bool	enableHostPathProvisioning enables HostPath PV provisioning when running without a cloud provider. This allows testing and development of provisioning features. HostPath provisioning is not supported in any way, won't work in a multi-node cluster, and should not be used for anything other than testing or development.
EnableDynamicProvisioning [Required] bool	enableDynamicProvisioning enables the provisioning of volumes when running within an environment that supports dynamic provisioning. Defaults to true.
PersistentVolumeRecyclerConfiguration [Required] PersistentVolumeRecyclerConfiguration	persistentVolumeRecyclerConfiguration holds configuration for persistent volume plugins.

Field	Description
FlexVolumePluginDir [Required] string	volumePluginDir is the full path of the directory in which the flex volume plugin should search for additional third party volume plugins

kube-proxy Configuration (v1alpha1)

Resource Types

- [KubeProxyConfiguration](#)

ClientConnectionConfiguration

Appears in:

- [KubeProxyConfiguration](#)
- [KubeSchedulerConfiguration](#)
- [KubeSchedulerConfiguration](#)
- [GenericControllerManagerConfiguration](#)

ClientConnectionConfiguration contains details for constructing a client.

Field	Description
kubeconfig [Required] string	kubeconfig is the path to a KubeConfig file.
acceptContentTypes [Required] string	acceptContentTypes defines the Accept header sent by clients when connecting to a server, overriding the default value of 'application/json'. This field will control all connections to the server used by a particular client.
contentType [Required] string	contentType is the content type used when sending data to the server from this client.
qps [Required] float32	qps controls the number of queries per second allowed for this connection.
burst [Required] int32	burst allows extra queries to accumulate when a client is exceeding its rate.

DebuggingConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)
- [KubeSchedulerConfiguration](#)
- [GenericControllerManagerConfiguration](#)

DebuggingConfiguration holds configuration for Debugging related features.

Field	Description
enableProfiling [Required] bool	enableProfiling enables profiling via web interface host:port/debug/pprof/
enableContentionProfiling [Required] bool	enableContentionProfiling enables block profiling, if enableProfiling is true.

LeaderElectionConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)
- [KubeSchedulerConfiguration](#)
- [GenericControllerManagerConfiguration](#)

LeaderElectionConfiguration defines the configuration of leader election clients for components that can run with leader election enabled.

Field	Description
leaderElect [Required] bool	leaderElect enables a leader election client to gain leadership before executing the main loop. Enable this when running replicated components for high availability.
leaseDuration [Required] meta/v1.Duration	leaseDuration is the duration that non-leader candidates will wait after observing a leadership renewal until attempting to acquire leadership of a led but unrenewed leader slot. This is effectively the maximum duration that a leader can be stopped before it is replaced by another candidate. This is only applicable if leader election is enabled.

Field	Description
renewDeadline [Required] meta/v1.Duration	renewDeadline is the interval between attempts by the acting master to renew a leadership slot before it stops leading. This must be less than or equal to the lease duration. This is only applicable if leader election is enabled.
retryPeriod [Required] meta/v1.Duration	retryPeriod is the duration the clients should wait between attempting acquisition and renewal of a leadership. This is only applicable if leader election is enabled.
resourceLock [Required] string	resourceLock indicates the resource object type that will be used to lock during leader election cycles.
resourceName [Required] string	resourceName indicates the name of resource object that will be used to lock during leader election cycles.
resourceNamespace [Required] string	resourceName indicates the namespace of resource object that will be used to lock during leader election cycles.

KubeProxyConfiguration

KubeProxyConfiguration contains everything necessary to configure the Kubernetes proxy server.

Field	Description
apiVersion string	kubeproxy.config.k8s.io/v1alpha1
kind string	KubeProxyConfiguration
featureGates [Required] map[string]bool	featureGates is a map of feature names to bools that enable or disable alpha/experimental features.
bindAddress [Required] string	bindAddress is the IP address for the proxy server to serve on (set to 0.0.0.0 for all interfaces)
healthzBindAddress [Required] string	healthzBindAddress is the IP address and port for the health check server to serve on, defaulting to 0.0.0.0:10256
metricsBindAddress [Required] string	metricsBindAddress is the IP address and port for the metrics server to serve on, defaulting to 127.0.0.1:10249 (set to 0.0.0.0 for all interfaces)

Field	Description
bindAddressHardFail [Required] bool	bindAddressHardFail, if true, kube-proxy will treat failure to bind to a port as fatal and exit
enableProfiling [Required] bool	enableProfiling enables profiling via web interface on / debug/pprof handler. Profiling handlers will be handled by metrics server.
clusterCIDR [Required] string	clusterCIDR is the CIDR range of the pods in the cluster. It is used to bridge traffic coming from outside of the cluster. If not provided, no off-cluster bridging will be performed.
hostnameOverride [Required] string	hostnameOverride, if non-empty, will be used as the identity instead of the actual hostname.
clientConnection [Required] ClientConnectionConfiguration	clientConnection specifies the kubeconfig file and client connection settings for the proxy server to use when communicating with the apiserver.
iptables [Required] KubeProxyIPTablesConfiguration	iptables contains iptables-related configuration options.
ipvs [Required] KubeProxyIPVSConfiguration	ipvs contains ipvs-related configuration options.
oomScoreAdj [Required] int32	oomScoreAdj is the oom-score-adj value for kube-proxy process. Values must be within the range [-1000, 1000]
mode [Required] ProxyMode	mode specifies which proxy mode to use.
portRange [Required] string	portRange is the range of host ports (beginPort-endPort, inclusive) that may be consumed in order to proxy service traffic. If unspecified (0-0) then ports will be randomly chosen.
conntrack [Required] KubeProxyConntrackConfiguration	conntrack contains conntrack-related configuration options.
configSyncPeriod [Required] meta/v1.Duration	configSyncPeriod is how often configuration from the apiserver is refreshed. Must be greater than 0.
nodePortAddresses [Required] []string	nodePortAddresses is the --nodeport-addresses value for kube-proxy process. Values must be valid IP blocks. These

Field	Description
	values are as a parameter to select the interfaces where nodeport works. In case someone would like to expose a service on localhost for local visit and some other interfaces for particular purpose, a list of IP blocks would do that. If set it to "127.0.0.0/8", kube-proxy will only select the loopback interface for NodePort. If set it to a non-zero IP block, kube-proxy will filter that down to just the IPs that applied to the node. An empty string slice is meant to select all network interfaces.
winkernel [Required] KubeProxyWinkernelConfiguration	winkernel contains winkernel-related configuration options.
showHiddenMetricsForVersion [Required] string	ShowHiddenMetricsForVersion is the version for which you want to show hidden metrics.
detectLocalMode [Required] LocalMode	DetectLocalMode determines mode to use for detecting local traffic, defaults to LocalModeClusterCIDR
detectLocal [Required] DetectLocalConfiguration	DetectLocal contains optional configuration settings related to DetectLocalMode.
logging [Required] LoggingConfiguration	logging specifies the options of logging. Refer to Logs Options for more information.

DetectLocalConfiguration

Appears in:

- [KubeProxyConfiguration](#)

DetectLocalConfiguration contains optional settings related to DetectLocalMode option

Field	Description
bridgeInterface [Required] string	BridgeInterface is a string argument which represents a single bridge interface name. Kube-proxy considers traffic as local if originating from this given bridge. This argument should be set if DetectLocalMode is set to LocalModeBridgeInterface.

Field	Description
interfaceNamePrefix [Required] string	InterfaceNamePrefix is a string argument which represents a single interface prefix name. Kube-proxy considers traffic as local if originating from one or more interfaces which match the given prefix. This argument should be set if DetectLocalMode is set to LocalModeInterfaceNamePrefix.

KubeProxyConntrackConfiguration

Appears in:

- [KubeProxyConfiguration](#)

KubeProxyConntrackConfiguration contains conntrack settings for the Kubernetes proxy server.

Field	Description
maxPerCore [Required] int32	maxPerCore is the maximum number of NAT connections to track per CPU core (0 to leave the limit as-is and ignore min).
min [Required] int32	min is the minimum value of connect-tracking records to allocate, regardless of conntrackMaxPerCore (set maxPerCore=0 to leave the limit as-is).
tcpEstablishedTimeout [Required] meta/v1.Duration	tcpEstablishedTimeout is how long an idle TCP connection will be kept open (e.g. '2s'). Must be greater than 0 to set.
tcpCloseWaitTimeout [Required] meta/v1.Duration	tcpCloseWaitTimeout is how long an idle conntrack entry in CLOSE_WAIT state will remain in the conntrack table. (e.g. '60s'). Must be greater than 0 to set.

KubeProxyIPTablesConfiguration

Appears in:

- [KubeProxyConfiguration](#)

KubeProxyIPTablesConfiguration contains iptables-related configuration details for the Kubernetes proxy server.

Field	Description
masqueradeBit [Required] int32	masqueradeBit is the bit of the iptables fwmark space to use for SNAT if using the pure iptables proxy mode. Values must be within the range [0, 31].
masqueradeAll [Required] bool	masqueradeAll tells kube-proxy to SNAT everything if using the pure iptables proxy mode.
localhostNodePorts [Required] bool	LocalhostNodePorts tells kube-proxy to allow service NodePorts to be accessed via localhost (iptables mode only)
syncPeriod [Required] meta/v1.Duration	syncPeriod is the period that iptables rules are refreshed (e.g. '5s', '1m', '2h22m'). Must be greater than 0.
minSyncPeriod [Required] meta/v1.Duration	minSyncPeriod is the minimum period that iptables rules are refreshed (e.g. '5s', '1m', '2h22m').

KubeProxyIPVSConfiguration

Appears in:

- [KubeProxyConfiguration](#)

KubeProxyIPVSConfiguration contains ipvs-related configuration details for the Kubernetes proxy server.

Field	Description
syncPeriod [Required] meta/v1.Duration	syncPeriod is the period that ipvs rules are refreshed (e.g. '5s', '1m', '2h22m'). Must be greater than 0.
minSyncPeriod [Required] meta/v1.Duration	minSyncPeriod is the minimum period that ipvs rules are refreshed (e.g. '5s', '1m', '2h22m').
scheduler [Required] string	ipvs scheduler
excludeCIDRs [Required] []string	excludeCIDRs is a list of CIDR's which the ipvs proxier should not touch when cleaning up ipvs services.
strictARP [Required] bool	strict ARP configure arp_ignore and arp_announce to avoid answering ARP queries from kube-ipvs0 interface

Field	Description
tcpTimeout [Required] meta/v1.Duration	tcpTimeout is the timeout value used for idle IPVS TCP sessions. The default value is 0, which preserves the current timeout value on the system.
tcpFinTimeout [Required] meta/v1.Duration	tcpFinTimeout is the timeout value used for IPVS TCP sessions after receiving a FIN. The default value is 0, which preserves the current timeout value on the system.
udpTimeout [Required] meta/v1.Duration	udpTimeout is the timeout value used for IPVS UDP packets. The default value is 0, which preserves the current timeout value on the system.

KubeProxyWinkernelConfiguration

Appears in:

- [KubeProxyConfiguration](#)

KubeProxyWinkernelConfiguration contains Windows/HNS settings for the Kubernetes proxy server.

Field	Description
networkName [Required] string	networkName is the name of the network kube-proxy will use to create endpoints and policies
sourceVip [Required] string	sourceVip is the IP address of the source VIP endpoint used for NAT when loadbalancing
enableDSR [Required] bool	enableDSR tells kube-proxy whether HNS policies should be created with DSR
rootHnsEndpointName [Required] string	RootHnsEndpointName is the name of hnsendpoint that is attached to l2bridge for root network namespace
forwardHealthCheckVip [Required] bool	ForwardHealthCheckVip forwards service VIP for health check port on Windows

LocalMode

(Alias of string)

Appears in:

- [KubeProxyConfiguration](#)

LocalMode represents modes to detect local traffic from the node

ProxyMode

(Alias of string)

Appears in:

- [KubeProxyConfiguration](#)

ProxyMode represents modes used by the Kubernetes proxy server.

Currently, two modes of proxy are available on Linux platforms: 'iptables' and 'ipvs'. One mode of proxy is available on Windows platforms: 'kernelspace'.

If the proxy mode is unspecified, the best-available proxy mode will be used (currently this is iptables on Linux and kernelspace on Windows). If the selected proxy mode cannot be used (due to lack of kernel support, missing userspace components, etc) then kube-proxy will exit with an error.

kube-scheduler Configuration (v1)

Resource Types

- [DefaultPreemptionArgs](#)
- [InterPodAffinityArgs](#)
- [KubeSchedulerConfiguration](#)
- [NodeAffinityArgs](#)
- [NodeResourcesBalancedAllocationArgs](#)
- [NodeResourcesFitArgs](#)
- [PodTopologySpreadArgs](#)
- [VolumeBindingArgs](#)

ClientConnectionConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)
- [KubeSchedulerConfiguration](#)

ClientConnectionConfiguration contains details for constructing a client.

Field	Description
	kubeconfig is the path to a KubeConfig file.

Field	Description
kubeconfig [Required] string	
acceptContentTypes [Required] string	acceptContentTypes defines the Accept header sent by clients when connecting to a server, overriding the default value of 'application/json'. This field will control all connections to the server used by a particular client.
contentType [Required] string	contentType is the content type used when sending data to the server from this client.
qps [Required] float32	qps controls the number of queries per second allowed for this connection.
burst [Required] int32	burst allows extra queries to accumulate when a client is exceeding its rate.

DebuggingConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)
- [KubeSchedulerConfiguration](#)

DebuggingConfiguration holds configuration for Debugging related features.

Field	Description
enableProfiling [Required] bool	enableProfiling enables profiling via web interface host:port/debug/pprof/
enableContentionProfiling [Required] bool	enableContentionProfiling enables block profiling, if enableProfiling is true.

LeaderElectionConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)
- [KubeSchedulerConfiguration](#)

LeaderElectionConfiguration defines the configuration of leader election clients for components that can run with leader election enabled.

Field	Description
leaderElect [Required] bool	leaderElect enables a leader election client to gain leadership before executing the main loop. Enable this when running replicated components for high availability.
leaseDuration [Required] meta/v1.Duration	leaseDuration is the duration that non-leader candidates will wait after observing a leadership renewal until attempting to acquire leadership of a led but unrenewed leader slot. This is effectively the maximum duration that a leader can be stopped before it is replaced by another candidate. This is only applicable if leader election is enabled.
renewDeadline [Required] meta/v1.Duration	renewDeadline is the interval between attempts by the acting master to renew a leadership slot before it stops leading. This must be less than or equal to the lease duration. This is only applicable if leader election is enabled.
retryPeriod [Required] meta/v1.Duration	retryPeriod is the duration the clients should wait between attempting acquisition and renewal of a leadership. This is only applicable if leader election is enabled.
resourceLock [Required] string	resourceLock indicates the resource object type that will be used to lock during leader election cycles.
resourceName [Required] string	resourceName indicates the name of resource object that will be used to lock during leader election cycles.
resourceNamespace [Required] string	resourceName indicates the namespace of resource object that will be used to lock during leader election cycles.

DefaultPreemptionArgs

DefaultPreemptionArgs holds arguments used to configure the DefaultPreemption plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	DefaultPreemptionArgs
minCandidateNodesPercentage [Required] int32	MinCandidateNodesPercentage is the minimum number of candidates to shortlist when dry running preemption as a

Field	Description
	percentage of number of nodes. Must be in the range [0, 100]. Defaults to 10% of the cluster size if unspecified.
minCandidateNodesAbsolute [Required] int32	MinCandidateNodesAbsolute is the absolute minimum number of candidates to shortlist. The likely number of candidates enumerated for dry running preemption is given by the formula: numCandidates = max(numNodes * minCandidateNodesPercentage, minCandidateNodesAbsolute) We say "likely" because there are other factors such as PDB violations that play a role in the number of candidates shortlisted. Must be at least 0 nodes. Defaults to 100 nodes if unspecified.

InterPodAffinityArgs

InterPodAffinityArgs holds arguments used to configure the InterPodAffinity plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	InterPodAffinityArgs
hardPodAffinityWeight [Required] int32	HardPodAffinityWeight is the scoring weight for existing pods with a matching hard affinity to the incoming pod.
ignorePreferredTermsOfExistingPods [Required] bool	IgnorePreferredTermsOfExistingPods configures the scheduler to ignore existing pods' preferred affinity rules when scoring candidate nodes, unless the incoming pod has inter-pod affinities.

KubeSchedulerConfiguration

KubeSchedulerConfiguration configures a scheduler

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	KubeSchedulerConfiguration
parallelism [Required] int32	Parallelism defines the amount of parallelism in algorithms for scheduling a Pods. Must be greater than 0. Defaults to 16

Field	Description
leaderElection [Required] LeaderElectionConfiguration	LeaderElection defines the configuration of leader election client.
clientConnection [Required] ClientConnectionConfiguration	ClientConnection specifies the kubeconfig file and client connection settings for the proxy server to use when communicating with the apiserver.
DebuggingConfiguration [Required] DebuggingConfiguration	(Members of DebuggingConfiguration are embedded into this type.) DebuggingConfiguration holds configuration for Debugging related features TODO: We might wanna make this a substruct like Debugging componentbaseconfigv1alpha1.DebuggingConfiguration
percentageOfNodesToScore [Required] int32	PercentageOfNodesToScore is the percentage of all nodes that once found feasible for running a pod, the scheduler stops its search for more feasible nodes in the cluster. This helps improve scheduler's performance. Scheduler always tries to find at least "minFeasibleNodesToFind" feasible nodes no matter what the value of this flag is. Example: if the cluster size is 500 nodes and the value of this flag is 30, then scheduler stops finding further feasible nodes once it finds 150 feasible ones. When the value is 0, default percentage (5%--50% based on the size of the cluster) of the nodes will be scored. It is overridden by profile level PercentageofNodesToScore.
podInitialBackoffSeconds [Required] int64	PodInitialBackoffSeconds is the initial backoff for unschedulable pods. If specified, it must be greater than 0. If this value is null, the default value (1s) will be used.
podMaxBackoffSeconds [Required] int64	PodMaxBackoffSeconds is the max backoff for unschedulable pods. If specified, it must be greater than podInitialBackoffSeconds. If this value is null, the default value (10s) will be used.
profiles [Required] []KubeSchedulerProfile	Profiles are scheduling profiles that kube-scheduler supports. Pods can choose to be scheduled under a particular profile by setting its associated scheduler name. Pods that don't specify any scheduler name are scheduled with the "default-scheduler" profile, if present here.

Field	Description
extenders [Required] []Extender	Extenders are the list of scheduler extenders, each holding the values of how to communicate with the extender. These extenders are shared by all scheduler profiles.
delayCacheUntilActive [Required] bool	DelayCacheUntilActive specifies when to start caching. If this is true and leader election is enabled, the scheduler will wait to fill informer caches until it is the leader. Doing so will have slower failover with the benefit of lower memory overhead while waiting to become leader. Defaults to false.

NodeAffinityArgs

NodeAffinityArgs holds arguments to configure the NodeAffinity plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	NodeAffinityArgs
addedAffinity core/ v1.NodeAffinity	AddedAffinity is applied to all Pods additionally to the NodeAffinity specified in the PodSpec. That is, Nodes need to satisfy AddedAffinity AND .spec.NodeAffinity. AddedAffinity is empty by default (all Nodes match). When AddedAffinity is used, some Pods with affinity requirements that match a specific Node (such as Daemonset Pods) might remain unschedulable.

NodeResourcesBalancedAllocationArgs

NodeResourcesBalancedAllocationArgs holds arguments used to configure NodeResourcesBalancedAllocation plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	NodeResourcesBalancedAllocationArgs
resources [Required] []ResourceSpec	Resources to be managed, the default is "cpu" and "memory" if not specified.

NodeResourcesFitArgs

NodeResourcesFitArgs holds arguments used to configure the NodeResourcesFit plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	NodeResourcesFitArgs
ignoredResources [Required] []string	IgnoredResources is the list of resources that NodeResources fit filter should ignore. This doesn't apply to scoring.
ignoredResourceGroups [Required] []string	IgnoredResourceGroups defines the list of resource groups that NodeResources fit filter should ignore. e.g. if group is ["example.com"], it will ignore all resource names that begin with "example.com", such as "example.com/aaa" and "example.com/bbb". A resource group name can't contain '/'. This doesn't apply to scoring.
scoringStrategy [Required] ScoringStrategy	ScoringStrategy selects the node resource scoring strategy. The default strategy is LeastAllocated with an equal "cpu" and "memory" weight.

PodTopologySpreadArgs

PodTopologySpreadArgs holds arguments used to configure the PodTopologySpread plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	PodTopologySpreadArgs
defaultConstraints [] core/v1.TopologySpreadConstraint	DefaultConstraints defines topology spread constraints to be applied to Pods that don't define any in pod.spec.topologySpreadConstraints. .defaultConstraints[*].labelSelector must be empty, as they are deduced from the Pod's membership to Services, ReplicationControllers, ReplicaSets or StatefulSets. When empty, .defaultingType must be "List".
defaultingType PodTopologySpreadConstraintsDefaulting	<p>DefaultingType determines how .defaultConstraints are deduced. one of "System" or "List".</p> <ul style="list-style-type: none"> "System": Use kubernetes defined constraints that spread Pods among Nodes and Zones. "List": Use constraints defined in .defaultConstraints. <p>Defaults to "System".</p>

VolumeBindingArgs

VolumeBindingArgs holds arguments used to configure the VolumeBinding plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1
kind string	VolumeBindingArgs
bindTimeoutSeconds [Required] int64	BindTimeoutSeconds is the timeout in seconds in volume binding operation. Value must be non-negative integer. The value zero indicates no waiting. If this value is nil, the default value (600) will be used.
shape []UtilizationShapePoint	Shape specifies the points defining the score function shape, which is used to score nodes based on the utilization of statically provisioned PVs. The utilization is calculated by dividing the total requested storage of the pod by the total capacity of feasible PVs on each node. Each point contains utilization (ranges from 0 to 100) and its associated score (ranges from 0 to 10). You can turn the priority by specifying different scores for different utilization numbers. The default shape points are: <ol style="list-style-type: none">1. 0 for 0 utilization2. 10 for 100 utilization All points must be sorted in increasing order by utilization.

Extender

Appears in:

- [KubeSchedulerConfiguration](#)

Extender holds the parameters used to communicate with the extender. If a verb is unspecified/empty, it is assumed that the extender chose not to provide that extension.

Field	Description
urlPrefix [Required] string	URLPrefix at which the extender is available
filterVerb [Required] string	Verb for the filter call, empty if not supported. This verb is appended to the URLPrefix when issuing the filter call to extender.

Field	Description
preemptVerb [Required] string	Verb for the preempt call, empty if not supported. This verb is appended to the URLPrefix when issuing the preempt call to extender.
prioritizeVerb [Required] string	Verb for the prioritize call, empty if not supported. This verb is appended to the URLPrefix when issuing the prioritize call to extender.
weight [Required] int64	The numeric multiplier for the node scores that the prioritize call generates. The weight should be a positive integer
bindVerb [Required] string	Verb for the bind call, empty if not supported. This verb is appended to the URLPrefix when issuing the bind call to extender. If this method is implemented by the extender, it is the extender's responsibility to bind the pod to apiserver. Only one extender can implement this function.
enableHTTPS [Required] bool	EnableHTTPS specifies whether https should be used to communicate with the extender
tlsConfig [Required] ExtenderTLSConfig	TLSConfig specifies the transport layer security config
httpTimeout [Required] meta/v1.Duration	HTTPTimeout specifies the timeout duration for a call to the extender. Filter timeout fails the scheduling of the pod. Prioritize timeout is ignored, k8s/other extenders priorities are used to select the node.
nodeCacheCapable [Required] bool	NodeCacheCapable specifies that the extender is capable of caching node information, so the scheduler should only send minimal information about the eligible nodes assuming that the extender already cached full details of all nodes in the cluster
managedResources []ExtenderManagedResource	<p>ManagedResources is a list of extended resources that are managed by this extender.</p> <ul style="list-style-type: none"> • A pod will be sent to the extender on the Filter, Prioritize and Bind (if the extender is the binder) phases iff the pod requests at least one of the extended resources in this list. If empty or unspecified, all pods will be sent to this extender. • If IgnoredByScheduler is set to true for a resource, kube-scheduler will skip checking the resource in predicates.

Field	Description
ignorable [Required] bool	Ignorable specifies if the extender is ignorable, i.e. scheduling should not fail when the extender returns an error or is not reachable.

ExtenderManagedResource

Appears in:

- [Extender](#)

ExtenderManagedResource describes the arguments of extended resources managed by an extender.

Field	Description
name [Required] string	Name is the extended resource name.
ignoredByScheduler [Required] bool	IgnoredByScheduler indicates whether kube-scheduler should ignore this resource when applying predicates.

ExtenderTLSConfig

Appears in:

- [Extender](#)

ExtenderTLSConfig contains settings to enable TLS with extender

Field	Description
insecure [Required] bool	Server should be accessed without verifying the TLS certificate. For testing only.
serverName [Required] string	ServerName is passed to the server for SNI and is used in the client to check server certificates against. If ServerName is empty, the hostname used to contact the server is used.
certFile [Required] string	Server requires TLS client certificate authentication
keyFile [Required] string	Server requires TLS client certificate authentication
	Trusted root certificates for server

Field	Description
caFile [Required] string	
certData [Required] []byte	CertData holds PEM-encoded bytes (typically read from a client certificate file). CertData takes precedence over CertFile
keyData [Required] []byte	KeyData holds PEM-encoded bytes (typically read from a client certificate key file). KeyData takes precedence over KeyFile
caData [Required] []byte	CAData holds PEM-encoded bytes (typically read from a root certificates bundle). CAData takes precedence over CAFile

KubeSchedulerProfile

Appears in:

- [KubeSchedulerConfiguration](#)

KubeSchedulerProfile is a scheduling profile.

Field	Description
schedulerName [Required] string	SchedulerName is the name of the scheduler associated to this profile. If SchedulerName matches with the pod's "spec.schedulerName", then the pod is scheduled with this profile.
percentageOfNodesToScore [Required] int32	PercentageOfNodesToScore is the percentage of all nodes that once found feasible for running a pod, the scheduler stops its search for more feasible nodes in the cluster. This helps improve scheduler's performance. Scheduler always tries to find at least "minFeasibleNodesToFind" feasible nodes no matter what the value of this flag is. Example: if the cluster size is 500 nodes and the value of this flag is 30, then scheduler stops finding further feasible nodes once it finds 150 feasible ones. When the value is 0, default percentage (5%--50% based on the size of the cluster) of the nodes will be scored. It will override global PercentageOfNodesToScore. If it is empty, global PercentageOfNodesToScore will be used.
plugins [Required] Plugins	Plugins specify the set of plugins that should be enabled or disabled. Enabled plugins are the ones that should be enabled in addition to the default plugins. Disabled plugins are any of the default plugins that should be disabled. When no enabled or disabled plugin is specified for an extension point, default plugins for that extension point will be used if there is any. If a QueueSort

Field	Description
	plugin is specified, the same QueueSort Plugin and PluginConfig must be specified for all profiles.
pluginConfig [Required] []PluginConfig	PluginConfig is an optional set of custom plugin arguments for each plugin. Omitting config args for a plugin is equivalent to using the default config for that plugin.

Plugin

Appears in:

- [PluginSet](#)

Plugin specifies a plugin name and its weight when applicable. Weight is used only for Score plugins.

Field	Description
name [Required] string	Name defines the name of plugin
weight [Required] int32	Weight defines the weight of plugin, only used for Score plugins.

PluginConfig

Appears in:

- [KubeSchedulerProfile](#)

PluginConfig specifies arguments that should be passed to a plugin at the time of initialization. A plugin that is invoked at multiple extension points is initialized once. Args can have arbitrary structure. It is up to the plugin to process these Args.

Field	Description
name [Required] string	Name defines the name of plugin being configured
args [Required] k8s.io/apimachinery/pkg/runtime.RawExtension	Args defines the arguments passed to the plugins at the time of initialization. Args can have arbitrary structure.

PluginSet

Appears in:

- [Plugins](#)

PluginSet specifies enabled and disabled plugins for an extension point. If an array is empty, missing, or nil, default plugins at that extension point will be used.

Field	Description
enabled [Required] []Plugin	Enabled specifies plugins that should be enabled in addition to default plugins. If the default plugin is also configured in the scheduler config file, the weight of plugin will be overridden accordingly. These are called after default plugins and in the same order specified here.
disabled [Required] []Plugin	Disabled specifies default plugins that should be disabled. When all default plugins need to be disabled, an array containing only one "*" should be provided.

Plugins

Appears in:

- [KubeSchedulerProfile](#)

Plugins include multiple extension points. When specified, the list of plugins for a particular extension point are the only ones enabled. If an extension point is omitted from the config, then the default set of plugins is used for that extension point. Enabled plugins are called in the order specified here, after default plugins. If they need to be invoked before default plugins, default plugins must be disabled and re-enabled here in desired order.

Field	Description
preEnqueue [Required] PluginSet	PreEnqueue is a list of plugins that should be invoked before adding pods to the scheduling queue.
queueSort [Required] PluginSet	QueueSort is a list of plugins that should be invoked when sorting pods in the scheduling queue.
preFilter [Required] PluginSet	PreFilter is a list of plugins that should be invoked at "PreFilter" extension point of the scheduling framework.
filter [Required] PluginSet	Filter is a list of plugins that should be invoked when filtering out nodes that cannot run the Pod.
postFilter [Required] PluginSet	PostFilter is a list of plugins that are invoked after filtering phase, but only when no feasible nodes were found for the pod.
preScore [Required] PluginSet	PreScore is a list of plugins that are invoked before scoring.

Field	Description
score [Required] PluginSet	Score is a list of plugins that should be invoked when ranking nodes that have passed the filtering phase.
reserve [Required] PluginSet	Reserve is a list of plugins invoked when reserving/unreserving resources after a node is assigned to run the pod.
permit [Required] PluginSet	Permit is a list of plugins that control binding of a Pod. These plugins can prevent or delay binding of a Pod.
preBind [Required] PluginSet	PreBind is a list of plugins that should be invoked before a pod is bound.
bind [Required] PluginSet	Bind is a list of plugins that should be invoked at "Bind" extension point of the scheduling framework. The scheduler call these plugins in order. Scheduler skips the rest of these plugins as soon as one returns success.
postBind [Required] PluginSet	PostBind is a list of plugins that should be invoked after a pod is successfully bound.
multiPoint [Required] PluginSet	<p>MultiPoint is a simplified config section to enable plugins for all valid extension points. Plugins enabled through MultiPoint will automatically register for every individual extension point the plugin has implemented. Disabling a plugin through MultiPoint disables that behavior. The same is true for disabling "*" through MultiPoint (no default plugins will be automatically registered). Plugins can still be disabled through their individual extension points.</p> <p>In terms of precedence, plugin config follows this basic hierarchy</p> <ol style="list-style-type: none"> 1. Specific extension points 2. Explicitly configured MultiPoint plugins 3. The set of default plugins, as MultiPoint plugins This implies that a higher precedence plugin will run first and overwrite any settings within MultiPoint. Explicitly user-configured plugins also take a higher precedence over default plugins. Within this hierarchy, an Enabled setting takes precedence over Disabled. For example, if a plugin is set in both multiPoint.Enabled and multiPoint.Disabled, the plugin will be enabled. Similarly, including multiPoint.Disabled = '*' and multiPoint.Enabled = pluginA will still register that specific plugin through MultiPoint. This follows the same behavior as all other extension point configurations.

PodTopologySpreadConstraintsDefaulting

(Alias of string)

Appears in:

- [PodTopologySpreadArgs](#)

PodTopologySpreadConstraintsDefaulting defines how to set default constraints for the PodTopologySpread plugin.

RequestedToCapacityRatioParam

Appears in:

- [ScoringStrategy](#)

RequestedToCapacityRatioParam define RequestedToCapacityRatio parameters

Field	Description
shape [Required] []UtilizationShapePoint	Shape is a list of points defining the scoring function shape.

ResourceSpec

Appears in:

- [NodeResourcesBalancedAllocationArgs](#)
- [ScoringStrategy](#)

ResourceSpec represents a single resource.

Field	Description
name [Required] string	Name of the resource.
weight [Required] int64	Weight of the resource.

ScoringStrategy

Appears in:

- [NodeResourcesFitArgs](#)

ScoringStrategy define ScoringStrategyType for node resource plugin

Field	Description
type [Required] ScoringStrategyType	Type selects which strategy to run.
resources [Required] []ResourceSpec	

Field	Description
	Resources to consider when scoring. The default resource set includes "cpu" and "memory" with an equal weight. Allowed weights go from 1 to 100. Weight defaults to 1 if not specified or explicitly set to 0.
requestedToCapacityRatio [Required] RequestedToCapacityRatioParam	Arguments specific to RequestedToCapacityRatio strategy.

ScoringStrategyType

(Alias of string)

Appears in:

- [ScoringStrategy](#)

ScoringStrategyType the type of scoring strategy used in NodeResourcesFit plugin.

UtilizationShapePoint

Appears in:

- [VolumeBindingArgs](#)
- [RequestedToCapacityRatioParam](#)

UtilizationShapePoint represents single point of priority function shape.

Field	Description
utilization [Required] int32	Utilization (x axis). Valid values are 0 to 100. Fully utilized node maps to 100.
score [Required] int32	Score assigned to given utilization (y axis). Valid values are 0 to 10.

kube-scheduler Configuration (v1beta3)

Resource Types

- [DefaultPreemptionArgs](#)
- [InterPodAffinityArgs](#)
- [KubeSchedulerConfiguration](#)
- [NodeAffinityArgs](#)
- [NodeResourcesBalancedAllocationArgs](#)
- [NodeResourcesFitArgs](#)

- [PodTopologySpreadArgs](#)
- [VolumeBindingArgs](#)

ClientConnectionConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)

ClientConnectionConfiguration contains details for constructing a client.

Field	Description
kubeconfig [Required] string	kubeconfig is the path to a KubeConfig file.
acceptContentTypes [Required] string	acceptContentTypes defines the Accept header sent by clients when connecting to a server, overriding the default value of 'application/json'. This field will control all connections to the server used by a particular client.
contentType [Required] string	contentType is the content type used when sending data to the server from this client.
qps [Required] float32	qps controls the number of queries per second allowed for this connection.
burst [Required] int32	burst allows extra queries to accumulate when a client is exceeding its rate.

DebuggingConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)

DebuggingConfiguration holds configuration for Debugging related features.

Field	Description
enableProfiling [Required] bool	enableProfiling enables profiling via web interface host:port/debug/pprof/
enableContentionProfiling [Required] bool	enableContentionProfiling enables block profiling, if enableProfiling is true.

LeaderElectionConfiguration

Appears in:

- [KubeSchedulerConfiguration](#)

LeaderElectionConfiguration defines the configuration of leader election clients for components that can run with leader election enabled.

Field	Description
leaderElect [Required] bool	leaderElect enables a leader election client to gain leadership before executing the main loop. Enable this when running replicated components for high availability.
leaseDuration [Required] meta/v1.Duration	leaseDuration is the duration that non-leader candidates will wait after observing a leadership renewal until attempting to acquire leadership of a led but unrenewed leader slot. This is effectively the maximum duration that a leader can be stopped before it is replaced by another candidate. This is only applicable if leader election is enabled.
renewDeadline [Required] meta/v1.Duration	renewDeadline is the interval between attempts by the acting master to renew a leadership slot before it stops leading. This must be less than or equal to the lease duration. This is only applicable if leader election is enabled.
retryPeriod [Required] meta/v1.Duration	retryPeriod is the duration the clients should wait between attempting acquisition and renewal of a leadership. This is only applicable if leader election is enabled.
resourceLock [Required] string	resourceLock indicates the resource object type that will be used to lock during leader election cycles.
resourceName [Required] string	resourceName indicates the name of resource object that will be used to lock during leader election cycles.
resourceNamespace [Required] string	resourceName indicates the namespace of resource object that will be used to lock during leader election cycles.

DefaultPreemptionArgs

DefaultPreemptionArgs holds arguments used to configure the DefaultPreemption plugin.

Field	Description
	kubescheduler.config.k8s.io/v1beta3

Field	Description
apiVersion string	
kind string	DefaultPreemptionArgs
minCandidateNodesPercentage [Required] int32	MinCandidateNodesPercentage is the minimum number of candidates to shortlist when dry running preemption as a percentage of number of nodes. Must be in the range [0, 100]. Defaults to 10% of the cluster size if unspecified.
minCandidateNodesAbsolute [Required] int32	MinCandidateNodesAbsolute is the absolute minimum number of candidates to shortlist. The likely number of candidates enumerated for dry running preemption is given by the formula: numCandidates = max(numNodes * minCandidateNodesPercentage, minCandidateNodesAbsolute) We say "likely" because there are other factors such as PDB violations that play a role in the number of candidates shortlisted. Must be at least 0 nodes. Defaults to 100 nodes if unspecified.

InterPodAffinityArgs

InterPodAffinityArgs holds arguments used to configure the InterPodAffinity plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1beta3
kind string	InterPodAffinityArgs
hardPodAffinityWeight [Required] int32	HardPodAffinityWeight is the scoring weight for existing pods with a matching hard affinity to the incoming pod.
ignorePreferredTermsOfExistingPods [Required] bool	IgnorePreferredTermsOfExistingPods configures the scheduler to ignore existing pods' preferred affinity rules when scoring candidate nodes, unless the incoming pod has inter-pod affinities.

KubeSchedulerConfiguration

KubeSchedulerConfiguration configures a scheduler

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1beta3

Field	Description
kind string	KubeSchedulerConfiguration
parallelism [Required] int32	Parallelism defines the amount of parallelism in algorithms for scheduling a Pods. Must be greater than 0. Defaults to 16
leaderElection [Required] LeaderElectionConfiguration	LeaderElection defines the configuration of leader election client.
clientConnection [Required] ClientConnectionConfiguration	ClientConnection specifies the kubeconfig file and client connection settings for the proxy server to use when communicating with the apiserver.
DebuggingConfiguration [Required] DebuggingConfiguration	(Members of DebuggingConfiguration are embedded into this type.) DebuggingConfiguration holds configuration for Debugging related features TODO: We might wanna make this a substruct like Debugging componentbaseconfigv1alpha1.DebuggingConfiguration
percentageOfNodesToScore [Required] int32	PercentageOfNodesToScore is the percentage of all nodes that once found feasible for running a pod, the scheduler stops its search for more feasible nodes in the cluster. This helps improve scheduler's performance. Scheduler always tries to find at least "minFeasibleNodesToFind" feasible nodes no matter what the value of this flag is. Example: if the cluster size is 500 nodes and the value of this flag is 30, then scheduler stops finding further feasible nodes once it finds 150 feasible ones. When the value is 0, default percentage (5%--50% based on the size of the cluster) of the nodes will be scored.
podInitialBackoffSeconds [Required] int64	PodInitialBackoffSeconds is the initial backoff for unschedulable pods. If specified, it must be greater than 0. If this value is null, the default value (1s) will be used.
podMaxBackoffSeconds [Required] int64	PodMaxBackoffSeconds is the max backoff for unschedulable pods. If specified, it must be greater than podInitialBackoffSeconds. If this value is null, the default value (10s) will be used.

Field	Description
profiles [Required] [] KubeSchedulerProfile	Profiles are scheduling profiles that kube-scheduler supports. Pods can choose to be scheduled under a particular profile by setting its associated scheduler name. Pods that don't specify any scheduler name are scheduled with the "default-scheduler" profile, if present here.
extenders [Required] [] Extender	Extenders are the list of scheduler extenders, each holding the values of how to communicate with the extender. These extenders are shared by all scheduler profiles.

NodeAffinityArgs

NodeAffinityArgs holds arguments to configure the NodeAffinity plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1beta3
kind string	NodeAffinityArgs
addedAffinity core/ v1.NodeAffinity	AddedAffinity is applied to all Pods additionally to the NodeAffinity specified in the PodSpec. That is, Nodes need to satisfy AddedAffinity AND .spec.NodeAffinity. AddedAffinity is empty by default (all Nodes match). When AddedAffinity is used, some Pods with affinity requirements that match a specific Node (such as Daemonset Pods) might remain unschedulable.

NodeResourcesBalancedAllocationArgs

NodeResourcesBalancedAllocationArgs holds arguments used to configure NodeResourcesBalancedAllocation plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1beta3
kind string	NodeResourcesBalancedAllocationArgs
resources [Required] [] ResourceSpec	Resources to be managed, the default is "cpu" and "memory" if not specified.

NodeResourcesFitArgs

NodeResourcesFitArgs holds arguments used to configure the NodeResourcesFit plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1beta3
kind string	NodeResourcesFitArgs
ignoredResources [Required] []string	IgnoredResources is the list of resources that NodeResources fit filter should ignore. This doesn't apply to scoring.
ignoredResourceGroups [Required] []string	IgnoredResourceGroups defines the list of resource groups that NodeResources fit filter should ignore. e.g. if group is ["example.com"], it will ignore all resource names that begin with "example.com", such as "example.com/aaa" and "example.com/bbb". A resource group name can't contain '/'. This doesn't apply to scoring.
scoringStrategy [Required] ScoringStrategy	ScoringStrategy selects the node resource scoring strategy. The default strategy is LeastAllocated with an equal "cpu" and "memory" weight.

PodTopologySpreadArgs

PodTopologySpreadArgs holds arguments used to configure the PodTopologySpread plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1beta3
kind string	PodTopologySpreadArgs
defaultConstraints [] core/v1.TopologySpreadConstraint	DefaultConstraints defines topology spread constraints to be applied to Pods that don't define any in pod.spec.topologySpreadConstraints. .defaultConstraints[*].labelSelector must be empty, as they are deduced from the Pod's membership to Services, ReplicationControllers, ReplicaSets or StatefulSets. When empty, .defaultingType must be "List".
defaultingType PodTopologySpreadConstraintsDefaulting	<p>DefaultingType determines how .defaultConstraints are deduced. one of "System" or "List".</p> <ul style="list-style-type: none"> "System": Use kubernetes defined constraints that spread Pods among Nodes and Zones. "List": Use constraints defined in .defaultConstraints. <p>Defaults to "System".</p>

VolumeBindingArgs

VolumeBindingArgs holds arguments used to configure the VolumeBinding plugin.

Field	Description
apiVersion string	kubescheduler.config.k8s.io/v1beta3
kind string	VolumeBindingArgs
bindTimeoutSeconds [Required] int64	BindTimeoutSeconds is the timeout in seconds in volume binding operation. Value must be non-negative integer. The value zero indicates no waiting. If this value is nil, the default value (600) will be used.
shape []UtilizationShapePoint	Shape specifies the points defining the score function shape, which is used to score nodes based on the utilization of statically provisioned PVs. The utilization is calculated by dividing the total requested storage of the pod by the total capacity of feasible PVs on each node. Each point contains utilization (ranges from 0 to 100) and its associated score (ranges from 0 to 10). You can turn the priority by specifying different scores for different utilization numbers. The default shape points are: <ol style="list-style-type: none">1. 0 for 0 utilization2. 10 for 100 utilization All points must be sorted in increasing order by utilization.

Extender

Appears in:

- [KubeSchedulerConfiguration](#)

Extender holds the parameters used to communicate with the extender. If a verb is unspecified/empty, it is assumed that the extender chose not to provide that extension.

Field	Description
urlPrefix [Required] string	URLPrefix at which the extender is available
filterVerb [Required] string	Verb for the filter call, empty if not supported. This verb is appended to the URLPrefix when issuing the filter call to extender.

Field	Description
preemptVerb [Required] string	Verb for the preempt call, empty if not supported. This verb is appended to the URLPrefix when issuing the preempt call to extender.
prioritizeVerb [Required] string	Verb for the prioritize call, empty if not supported. This verb is appended to the URLPrefix when issuing the prioritize call to extender.
weight [Required] int64	The numeric multiplier for the node scores that the prioritize call generates. The weight should be a positive integer
bindVerb [Required] string	Verb for the bind call, empty if not supported. This verb is appended to the URLPrefix when issuing the bind call to extender. If this method is implemented by the extender, it is the extender's responsibility to bind the pod to apiserver. Only one extender can implement this function.
enableHTTPS [Required] bool	EnableHTTPS specifies whether https should be used to communicate with the extender
tlsConfig [Required] ExtenderTLSConfig	TLSConfig specifies the transport layer security config
httpTimeout [Required] meta/v1.Duration	HTTPTimeout specifies the timeout duration for a call to the extender. Filter timeout fails the scheduling of the pod. Prioritize timeout is ignored, k8s/other extenders priorities are used to select the node.
nodeCacheCapable [Required] bool	NodeCacheCapable specifies that the extender is capable of caching node information, so the scheduler should only send minimal information about the eligible nodes assuming that the extender already cached full details of all nodes in the cluster
managedResources []ExtenderManagedResource	<p>ManagedResources is a list of extended resources that are managed by this extender.</p> <ul style="list-style-type: none"> A pod will be sent to the extender on the Filter, Prioritize and Bind (if the extender is the binder) phases iff the pod requests at least one of the extended resources in this list. If empty or unspecified, all pods will be sent to this extender. If IgnoredByScheduler is set to true for a resource, kube-scheduler will skip checking the resource in predicates.

Field	Description
ignorable [Required] bool	Ignorable specifies if the extender is ignorable, i.e. scheduling should not fail when the extender returns an error or is not reachable.

ExtenderManagedResource

Appears in:

- [Extender](#)

ExtenderManagedResource describes the arguments of extended resources managed by an extender.

Field	Description
name [Required] string	Name is the extended resource name.
ignoredByScheduler [Required] bool	IgnoredByScheduler indicates whether kube-scheduler should ignore this resource when applying predicates.

ExtenderTLSConfig

Appears in:

- [Extender](#)

ExtenderTLSConfig contains settings to enable TLS with extender

Field	Description
insecure [Required] bool	Server should be accessed without verifying the TLS certificate. For testing only.
serverName [Required] string	ServerName is passed to the server for SNI and is used in the client to check server certificates against. If ServerName is empty, the hostname used to contact the server is used.
certFile [Required] string	Server requires TLS client certificate authentication
keyFile [Required] string	Server requires TLS client certificate authentication
	Trusted root certificates for server

Field	Description
caFile [Required] string	
certData [Required] []byte	CertData holds PEM-encoded bytes (typically read from a client certificate file). CertData takes precedence over CertFile
keyData [Required] []byte	KeyData holds PEM-encoded bytes (typically read from a client certificate key file). KeyData takes precedence over KeyFile
caData [Required] []byte	CAData holds PEM-encoded bytes (typically read from a root certificates bundle). CAData takes precedence over CAFile

KubeSchedulerProfile

Appears in:

- [KubeSchedulerConfiguration](#)

KubeSchedulerProfile is a scheduling profile.

Field	Description
schedulerName [Required] string	SchedulerName is the name of the scheduler associated to this profile. If SchedulerName matches with the pod's "spec.schedulerName", then the pod is scheduled with this profile.
plugins [Required] Plugins	Plugins specify the set of plugins that should be enabled or disabled. Enabled plugins are the ones that should be enabled in addition to the default plugins. Disabled plugins are any of the default plugins that should be disabled. When no enabled or disabled plugin is specified for an extension point, default plugins for that extension point will be used if there is any. If a QueueSort plugin is specified, the same QueueSort Plugin and PluginConfig must be specified for all profiles.
pluginConfig [Required] []PluginConfig	PluginConfig is an optional set of custom plugin arguments for each plugin. Omitting config args for a plugin is equivalent to using the default config for that plugin.

Plugin

Appears in:

- [PluginSet](#)

Plugin specifies a plugin name and its weight when applicable. Weight is used only for Score plugins.

Field	Description
name [Required] string	Name defines the name of plugin
weight [Required] int32	Weight defines the weight of plugin, only used for Score plugins.

PluginConfig

Appears in:

- [KubeSchedulerProfile](#)

PluginConfig specifies arguments that should be passed to a plugin at the time of initialization. A plugin that is invoked at multiple extension points is initialized once. Args can have arbitrary structure. It is up to the plugin to process these Args.

Field	Description
name [Required] string	Name defines the name of plugin being configured
args [Required] k8s.io/apimachinery/pkg/runtime.RawExtension	Args defines the arguments passed to the plugins at the time of initialization. Args can have arbitrary structure.

PluginSet

Appears in:

- [Plugins](#)

PluginSet specifies enabled and disabled plugins for an extension point. If an array is empty, missing, or nil, default plugins at that extension point will be used.

Field	Description
enabled [Required] []Plugin	Enabled specifies plugins that should be enabled in addition to default plugins. If the default plugin is also configured in the scheduler config file, the weight of plugin will be overridden accordingly. These are called after default plugins and in the same order specified here.
disabled [Required] []Plugin	Disabled specifies default plugins that should be disabled. When all default plugins need to be disabled, an array containing only one "*" should be provided.

Plugins

Appears in:

- [KubeSchedulerProfile](#)

Plugins include multiple extension points. When specified, the list of plugins for a particular extension point are the only ones enabled. If an extension point is omitted from the config, then the default set of plugins is used for that extension point. Enabled plugins are called in the order specified here, after default plugins. If they need to be invoked before default plugins, default plugins must be disabled and re-enabled here in desired order.

Field	Description
preEnqueue [Required] PluginSet	PreEnqueue is a list of plugins that should be invoked before adding pods to the scheduling queue.
queueSort [Required] PluginSet	QueueSort is a list of plugins that should be invoked when sorting pods in the scheduling queue.
preFilter [Required] PluginSet	PreFilter is a list of plugins that should be invoked at "PreFilter" extension point of the scheduling framework.
filter [Required] PluginSet	Filter is a list of plugins that should be invoked when filtering out nodes that cannot run the Pod.
postFilter [Required] PluginSet	PostFilter is a list of plugins that are invoked after filtering phase, but only when no feasible nodes were found for the pod.
preScore [Required] PluginSet	PreScore is a list of plugins that are invoked before scoring.
score [Required] PluginSet	Score is a list of plugins that should be invoked when ranking nodes that have passed the filtering phase.
reserve [Required] PluginSet	Reserve is a list of plugins invoked when reserving/unreserving resources after a node is assigned to run the pod.
permit [Required] PluginSet	Permit is a list of plugins that control binding of a Pod. These plugins can prevent or delay binding of a Pod.
preBind [Required] PluginSet	PreBind is a list of plugins that should be invoked before a pod is bound.

Field	Description
bind [Required] PluginSet	Bind is a list of plugins that should be invoked at "Bind" extension point of the scheduling framework. The scheduler call these plugins in order. Scheduler skips the rest of these plugins as soon as one returns success.
postBind [Required] PluginSet	PostBind is a list of plugins that should be invoked after a pod is successfully bound.
multiPoint [Required] PluginSet	<p>MultiPoint is a simplified config section to enable plugins for all valid extension points. Plugins enabled through MultiPoint will automatically register for every individual extension point the plugin has implemented. Disabling a plugin through MultiPoint disables that behavior. The same is true for disabling "*" through MultiPoint (no default plugins will be automatically registered). Plugins can still be disabled through their individual extension points.</p> <p>In terms of precedence, plugin config follows this basic hierarchy</p> <ol style="list-style-type: none"> 1. Specific extension points 2. Explicitly configured MultiPoint plugins 3. The set of default plugins, as MultiPoint plugins This implies that a higher precedence plugin will run first and overwrite any settings within MultiPoint. Explicitly user-configured plugins also take a higher precedence over default plugins. Within this hierarchy, an Enabled setting takes precedence over Disabled. For example, if a plugin is set in both multiPoint.Enabled and multiPoint.Disabled, the plugin will be enabled. Similarly, including multiPoint.Disabled = '*' and multiPoint.Enabled = pluginA will still register that specific plugin through MultiPoint. This follows the same behavior as all other extension point configurations.

PodTopologySpreadConstraintsDefaulting

(Alias of string)

Appears in:

- [PodTopologySpreadArgs](#)

PodTopologySpreadConstraintsDefaulting defines how to set default constraints for the PodTopologySpread plugin.

RequestedToCapacityRatioParam

Appears in:

- [ScoringStrategy](#)

RequestedToCapacityRatioParam define RequestedToCapacityRatio parameters

Field	Description
shape [Required] []UtilizationShapePoint	Shape is a list of points defining the scoring function shape.

ResourceSpec

Appears in:

- [NodeResourcesBalancedAllocationArgs](#)
- [ScoringStrategy](#)

ResourceSpec represents a single resource.

Field	Description
name [Required] string	Name of the resource.
weight [Required] int64	Weight of the resource.

ScoringStrategy

Appears in:

- [NodeResourcesFitArgs](#)

ScoringStrategy define ScoringStrategyType for node resource plugin

Field	Description
type [Required] ScoringStrategyType	Type selects which strategy to run.
resources [Required] []ResourceSpec	Resources to consider when scoring. The default resource set includes "cpu" and "memory" with an equal weight. Allowed weights go from 1 to 100. Weight defaults to 1 if not specified or explicitly set to 0.
requestedToCapacityRatio [Required] RequestedToCapacityRatioParam	Arguments specific to RequestedToCapacityRatio strategy.

ScoringStrategyType

(Alias of string)

Appears in:

- [ScoringStrategy](#)

ScoringStrategyType the type of scoring strategy used in NodeResourcesFit plugin.

UtilizationShapePoint

Appears in:

- [VolumeBindingArgs](#)
- [RequestedToCapacityRatioParam](#)

UtilizationShapePoint represents single point of priority function shape.

Field	Description
utilization [Required] int32	Utilization (x axis). Valid values are 0 to 100. Fully utilized node maps to 100.
score [Required] int32	Score assigned to given utilization (y axis). Valid values are 0 to 10.

kubeadm Configuration (v1beta3)

Overview

Package v1beta3 defines the v1beta3 version of the kubeadm configuration file format. This version improves on the v1beta2 format by fixing some minor issues and adding a few new fields.

A list of changes since v1beta2:

- The deprecated "ClusterConfiguration.useHyperKubeImage" field has been removed. Kubeadm no longer supports the hyperkube image.
- The "ClusterConfiguration.DNS.Type" field has been removed since CoreDNS is the only supported DNS server type by kubeadm.
- Include "datapolicy" tags on the fields that hold secrets. This would result in the field values to be omitted when API structures are printed with klog.
- Add "InitConfiguration.SkipPhases", "JoinConfiguration.SkipPhases" to allow skipping a list of phases during kubeadm init/join command execution.
- Add "InitConfiguration.NodeRegistration.ImagePullPolicy" and "JoinConfiguration.NodeRegistration.ImagePullPolicy" to allow specifying the images pull policy during kubeadm "init" and "join". The value must be one of "Always", "Never" or "IfNotPresent". "IfNotPresent" is the default, which has been the existing behavior prior to this addition.
- Add "InitConfiguration.Patches.Directory", "JoinConfiguration.Patches.Directory" to allow the user to configure a directory from which to take patches for components deployed by kubeadm.

- Move the BootstrapToken* API and related utilities out of the "kubeadm" API group to a new group "bootstraptoken". The kubeadm API version v1beta3 no longer contains the BootstrapToken* structures.

Migration from old kubeadm config versions

- kubeadm v1.15.x and newer can be used to migrate from v1beta1 to v1beta2.
- kubeadm v1.22.x and newer no longer support v1beta1 and older APIs, but can be used to migrate v1beta2 to v1beta3.
- kubeadm v1.27.x and newer no longer support v1beta2 and older APIs,

Basics

The preferred way to configure kubeadm is to pass an YAML configuration file with the --config option. Some of the configuration options defined in the kubeadm config file are also available as command line flags, but only the most common/simple use case are supported with this approach.

A kubeadm config file could contain multiple configuration types separated using three dashes (---).

kubeadm supports the following configuration types:

```
apiVersion: kubeadm.k8s.io/v1beta3
kind: InitConfiguration

apiVersion: kubeadm.k8s.io/v1beta3
kind: ClusterConfiguration

apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration

apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration

apiVersion: kubeadm.k8s.io/v1beta3
kind: JoinConfiguration
```

To print the defaults for "init" and "join" actions use the following commands:

```
kubeadm config print init-defaults
kubeadm config print join-defaults
```

The list of configuration types that must be included in a configuration file depends by the action you are performing (init or join) and by the configuration options you are going to use (defaults or advanced customization).

If some configuration types are not provided, or provided only partially, kubeadm will use default values; defaults provided by kubeadm includes also enforcing consistency of values across components when required (e.g. --cluster-cidr flag on controller manager and clusterCIDR on kube-proxy).

Users are always allowed to override default values, with the only exception of a small subset of setting with relevance for security (e.g. enforce authorization-mode Node and RBAC on api server).

If the user provides a configuration types that is not expected for the action you are performing, kubeadm will ignore those types and print a warning.

Kubeadm init configuration types

When executing kubeadm init with the --config option, the following configuration types could be used: InitConfiguration, ClusterConfiguration, KubeProxyConfiguration, KubeletConfiguration, but only one between InitConfiguration and ClusterConfiguration is mandatory.

```
apiVersion: kubeadm.k8s.io/v1beta3
kind: InitConfiguration
bootstrapTokens:
...
nodeRegistration:
...
```

The InitConfiguration type should be used to configure runtime settings, that in case of kubeadm init are the configuration of the bootstrap token and all the setting which are specific to the node where kubeadm is executed, including:

- NodeRegistration, that holds fields that relate to registering the new node to the cluster; use it to customize the node name, the CRI socket to use or any other settings that should apply to this node only (e.g. the node ip).
- LocalAPIEndpoint, that represents the endpoint of the instance of the API server to be deployed on this node; use it e.g. to customize the API server advertise address.

```
apiVersion: kubeadm.k8s.io/v1beta3
kind: ClusterConfiguration
networking:
...
etcd:
...
apiServer:
  extraArgs:
    ...
extraVolumes:
...
...
```

The ClusterConfiguration type should be used to configure cluster-wide settings, including settings for:

- networking that holds configuration for the networking topology of the cluster; use it e.g. to customize Pod subnet or services subnet.
- etcd: use it e.g. to customize the local etcd or to configure the API server for using an external etcd cluster.

- kube-apiserver, kube-scheduler, kube-controller-manager configurations; use it to customize control-plane components by adding customized setting or overriding kubeadm default settings.

```
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
...

```

The KubeProxyConfiguration type should be used to change the configuration passed to kube-proxy instances deployed in the cluster. If this object is not provided or provided only partially, kubeadm applies defaults.

See <https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/> or <https://pkg.go.dev/k8s.io/kube-proxy/config/v1alpha1#KubeProxyConfiguration> for kube-proxy official documentation.

```
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
...

```

The KubeletConfiguration type should be used to change the configurations that will be passed to all kubelet instances deployed in the cluster. If this object is not provided or provided only partially, kubeadm applies defaults.

See <https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/> or <https://pkg.go.dev/k8s.io/kubelet/config/v1beta1#KubeletConfiguration> for kubelet official documentation.

Here is a fully populated example of a single YAML file containing multiple configuration types to be used during a kubeadm init run.

```
apiVersion: kubeadm.k8s.io/v1beta3
kind: InitConfiguration
bootstrapTokens:
  - token: "9a08jv.c0izixklcxtmnze7"
    description: "kubeadm bootstrap token"
    ttl: "24h"
  - token: "783bde.3f89s0fje9f38fhf"
    description: "another bootstrap token"
    usages:
      - authentication
      - signing
    groups:
      - system:bootstrappers:kubeadm:default-node-token
nodeRegistration:
  name: "ec2-10-100-0-1"
  criSocket: "/var/run/dockershim.sock"
  taints:
    - key: "kubeadmNode"
      value: "someValue"
      effect: "NoSchedule"
kubeletExtraArgs:
  v: 4
ignorePreflightErrors:

```

```
- IsPrivilegedUser
imagePullPolicy: "IfNotPresent"
localAPIEndpoint:
  advertiseAddress: "10.100.0.1"
  bindPort: 6443
certificateKey: "e6a2eb8581237ab72a4f494f30285ec12a9694d750b9785706a83bfcbdd2204"
skipPhases:
  - addon/kube-proxy
---
apiVersion: kubeadm.k8s.io/v1beta3
kind: ClusterConfiguration
etcd:
  # one of local or external
  local:
    imageRepository: "registry.k8s.io"
    imageTag: "3.2.24"
    dataDir: "/var/lib/etcd"
    extraArgs:
      listen-client-urls: "http://10.100.0.1:2379"
    serverCertSANs:
      - "ec2-10-100-0-1.compute-1.amazonaws.com"
  peerCertSANs:
    - "10.100.0.1"
  # external:
  #   endpoints:
  #     - "10.100.0.1:2379"
  #     - "10.100.0.2:2379"
  #   caFile: "/etcd/kubernetes/pki/etcd/etcd-ca.crt"
  #   certFile: "/etcd/kubernetes/pki/etcd/etcd.crt"
  #   keyFile: "/etcd/kubernetes/pki/etcd/etcd.key"
networking:
  serviceSubnet: "10.96.0.0/16"
  podSubnet: "10.244.0.0/24"
  dnsDomain: "cluster.local"
kubernetesVersion: "v1.21.0"
controlPlaneEndpoint: "10.100.0.1:6443"
apiServer:
  extraArgs:
    authorization-mode: "Node,RBAC"
  extraVolumes:
    - name: "some-volume"
      hostPath: "/etc/some-path"
      mountPath: "/etc/some-pod-path"
      readOnly: false
      pathType: File
certSANs:
  - "10.100.1.1"
  - "ec2-10-100-0-1.compute-1.amazonaws.com"
timeoutForControlPlane: 4m0s
controllerManager:
  extraArgs:
    "node-cidr-mask-size": "20"
  extraVolumes:
```

```

- name: "some-volume"
  hostPath: "/etc/some-path"
  mountPath: "/etc/some-pod-path"
  readOnly: false
  pathType: File
scheduler:
extraArgs:
  bind-address: "10.100.0.1"
extraVolumes:
- name: "some-volume"
  hostPath: "/etc/some-path"
  mountPath: "/etc/some-pod-path"
  readOnly: false
  pathType: File
certificatesDir: "/etc/kubernetes/pki"
imageRepository: "registry.k8s.io"
clusterName: "example-cluster"
---
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
# kubelet specific options here
---
apiVersion: kubeproxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
# kube-proxy specific options here

```

Kubeadm join configuration types

When executing kubeadm join with the --config option, the JoinConfiguration type should be provided.

```

apiVersion: kubeadm.k8s.io/v1beta3
kind: JoinConfiguration
...

```

The JoinConfiguration type should be used to configure runtime settings, that in case of kubeadm join are the discovery method used for accessing the cluster info and all the setting which are specific to the node where kubeadm is executed, including:

- nodeRegistration, that holds fields that relate to registering the new node to the cluster; use it to customize the node name, the CRI socket to use or any other settings that should apply to this node only (e.g. the node ip).
- apiEndpoint, that represents the endpoint of the instance of the API server to be eventually deployed on this node.

Resource Types

- [ClusterConfiguration](#)
- [InitConfiguration](#)
- [JoinConfiguration](#)

BootstrapToken

Appears in:

- [InitConfiguration](#)

BootstrapToken describes one bootstrap token, stored as a Secret in the cluster

Field	Description
token [Required] BootstrapTokenString	token is used for establishing bidirectional trust between nodes and control-planes. Used for joining nodes in the cluster.
description string	description sets a human-friendly message why this token exists and what it's used for, so other administrators can know its purpose.
ttl meta/v1.Duration	ttl defines the time to live for this token. Defaults to 24h. expires and ttl are mutually exclusive.
expires meta/v1.Time	expires specifies the timestamp when this token expires. Defaults to being set dynamically at runtime based on the ttl. expires and ttl are mutually exclusive.
usages []string	usages describes the ways in which this token can be used. Can by default be used for establishing bidirectional trust, but that can be changed here.
groups []string	groups specifies the extra groups that this token will authenticate as when/if used for authentication

BootstrapTokenString

Appears in:

- [BootstrapToken](#)

BootstrapTokenString is a token of the format abcdefabcdef0123456789 that is used for both validation of the practically of the API server from a joining node's point of view and as an authentication method for the node in the bootstrap phase of "kubeadm join". This token is and should be short-lived.

Field	Description
- [Required] string	No description provided.
- [Required] string	No description provided.

ClusterConfiguration

ClusterConfiguration contains cluster-wide configuration for a kubeadm cluster

Field	Description
apiVersion string	kubeadm.k8s.io/v1beta3
kind string	ClusterConfiguration
etcd Etcd	etcd holds the configuration for etcd.
networking Networking	networking holds configuration for the networking topology of the cluster.
kubernetesVersion string	kubernetesVersion is the target version of the control plane.
controlPlaneEndpoint string	controlPlaneEndpoint sets a stable IP address or DNS name for the control plane. It can be a valid IP address or a RFC-1123 DNS subdomain, both with optional TCP port. In case the controlPlaneEndpoint is not specified, the advertiseAddress + bindPort are used; in case the controlPlaneEndpoint is specified but without a TCP port, the bindPort is used. Possible usages are: <ul style="list-style-type: none">• In a cluster with more than one control plane instances, this field should be assigned the address of the external load balancer in front of the control plane instances.• In environments with enforced node recycling, the controlPlaneEndpoint could be used for assigning a stable DNS to the control plane.
apiServer APIServer	apiServer contains extra settings for the API server.
controllerManager ControlPlaneComponent	controllerManager contains extra settings for the controller manager.
scheduler ControlPlaneComponent	scheduler contains extra settings for the scheduler.
dns DNS	dns defines the options for the DNS add-on installed in the cluster.
certificatesDir string	certificatesDir specifies where to store or look for all required certificates.
imageRepository string	imageRepository sets the container registry to pull images from. If empty, registry.k8s.io will be used by default. In case of kubernetes

Field	Description
	version is a CI build (kubernetes version starts with ci/) gcr.io/k8s-staging-ci-images will be used as a default for control plane components and for kube-proxy, while registry.k8s.io will be used for all the other images.
featureGates map[string]bool	featureGates contains the feature gates enabled by the user.
clusterName string	The cluster name.

InitConfiguration

InitConfiguration contains a list of elements that is specific "kubeadm init"-only runtime information. kubeadm init-only information. These fields are solely used the first time kubeadm init runs. After that, the information in the fields IS NOT uploaded to the kubeadm-config ConfigMap that is used by kubeadm upgrade for instance. These fields must be omitted.

Field	Description
apiVersion string	kubeadm.k8s.io/v1beta3
kind string	InitConfiguration
bootstrapTokens []BootstrapToken	bootstrapTokens is respected at kubeadm init time and describes a set of Bootstrap Tokens to create. This information IS NOT uploaded to the kubeadm cluster configmap, partly because of its sensitive nature
nodeRegistration NodeRegistrationOptions	nodeRegistration holds fields that relate to registering the new control-plane node to the cluster.
localAPIEndpoint APIEndpoint	localAPIEndpoint represents the endpoint of the API server instance that's deployed on this control plane node. In HA setups, this differs from ClusterConfiguration.controlPlaneEndpoint in the sense that controlPlaneEndpoint is the global endpoint for the cluster, which then load-balances the requests to each individual API server. This configuration object lets you customize what IP/DNS name and port the local API server advertises it's accessible on. By default, kubeadm tries to auto-detect the IP of the default interface and use that, but in case that process fails you may set the desired value here.
certificateKey string	certificateKey sets the key with which certificates and keys are encrypted prior to being uploaded in a Secret in the cluster during the uploadcerts init phase.

Field	Description
skipPhases []string	skipPhases is a list of phases to skip during command execution. The list of phases can be obtained with the kubeadm init --help command. The flag "--skip-phases" takes precedence over this field.
patches Patches	patches contains options related to applying patches to components deployed by kubeadm during kubeadm init.

JoinConfiguration

JoinConfiguration contains elements describing a particular node.

Field	Description
apiVersion string	kubeadm.k8s.io/v1beta3
kind string	JoinConfiguration
nodeRegistration NodeRegistrationOptions	nodeRegistration holds fields that relate to registering the new control-plane node to the cluster.
caCertPath string	caCertPath is the path to the SSL certificate authority used to secure communications between a node and the control-plane. Defaults to "/etc/kubernetes/pki/ca.crt".
discovery [Required] Discovery	discovery specifies the options for the kubelet to use during the TLS bootstrap process.
controlPlane JoinControlPlane	controlPlane defines the additional control plane instance to be deployed on the joining node. If nil, no additional control plane instance will be deployed.
skipPhases []string	skipPhases is a list of phases to skip during command execution. The list of phases can be obtained with the kubeadm join --help command. The flag --skip-phases takes precedence over this field.
patches Patches	patches contains options related to applying patches to components deployed by kubeadm during kubeadm join.

APIEndpoint

Appears in:

- [InitConfiguration](#)
- [JoinControlPlane](#)

APIEndpoint struct contains elements of API server instance deployed on a node.

Field	Description
advertiseAddress string	advertiseAddress sets the IP address for the API server to advertise.
bindPort int32	bindPort sets the secure port for the API Server to bind to. Defaults to 6443.

APIServer

Appears in:

- [ClusterConfiguration](#)

APIServer holds settings necessary for API server deployments in the cluster

Field	Description
ControlPlaneComponent [Required] ControlPlaneComponent	(Members of ControlPlaneComponent are embedded into this type.) No description provided.
certSANs []string	certSANs sets extra Subject Alternative Names (SANs) for the API Server signing certificate.
timeoutForControlPlane meta/v1.Duration	timeoutForControlPlane controls the timeout that we wait for API server to appear.

BootstrapTokenDiscovery

Appears in:

- [Discovery](#)

BootstrapTokenDiscovery is used to set the options for bootstrap token based discovery

Field	Description
token [Required] string	token is a token used to validate cluster information fetched from the control-plane.

Field	Description
apiServerEndpoint string	apiServerEndpoint is an IP or domain name to the API server from which information will be fetched.
caCertHashes []string	caCertHashes specifies a set of public key pins to verify when token-based discovery is used. The root CA found during discovery must match one of these values. Specifying an empty set disables root CA pinning, which can be unsafe. Each hash is specified as <type>:<value>, where the only currently supported type is "sha256". This is a hex-encoded SHA-256 hash of the Subject Public Key Info (SPKI) object in DER-encoded ASN.1. These hashes can be calculated using, for example, OpenSSL.
unsafeSkipCAVerification bool	unsafeSkipCAVerification allows token-based discovery without CA verification via caCertHashes. This can weaken the security of kubeadm since other nodes can impersonate the control-plane.

ControlPlaneComponent

Appears in:

- [ClusterConfiguration](#)
- [APIServer](#)

ControlPlaneComponent holds settings common to control plane component of the cluster

Field	Description
extraArgs map[string]string	extraArgs is an extra set of flags to pass to the control plane component. A key in this map is the flag name as it appears on the command line except without leading dash(es).
extraVolumes []HostPathMount	extraVolumes is an extra set of host volumes, mounted to the control plane component.

DNS

Appears in:

- [ClusterConfiguration](#)

DNS defines the DNS addon that should be used in the cluster

Field	Description
	(Members of ImageMeta are embedded into this type.)

Field	Description
ImageMeta [Required] ImageMeta	imageMeta allows to customize the image used for the DNS component.

Discovery

Appears in:

- [JoinConfiguration](#)

Discovery specifies the options for the kubelet to use during the TLS Bootstrap process.

Field	Description
bootstrapToken BootstrapTokenDiscovery	bootstrapToken is used to set the options for bootstrap token based discovery. bootstrapToken and file are mutually exclusive.
file FileDiscovery	file is used to specify a file or URL to a kubeconfig file from which to load cluster information. bootstrapToken and file are mutually exclusive.
tlsBootstrapToken string	tlsBootstrapToken is a token used for TLS bootstrapping. If bootstrapToken is set, this field is defaulted to .bootstrapToken.token, but can be overridden. If file is set, this field must be set in case the KubeConfigFile does not contain any other authentication information
timeout meta/v1.Duration	timeout modifies the discovery timeout.

Etcd

Appears in:

- [ClusterConfiguration](#)

Etcd contains elements describing Etcd configuration.

Field	Description
local LocalEtcd	local provides configuration knobs for configuring the local etcd instance. local and external are mutually exclusive.
external ExternalEtcd	external describes how to connect to an external etcd cluster. local and external are mutually exclusive.

ExternalEtcd

Appears in:

- [Etcd](#)

ExternalEtcd describes an external etcd cluster. Kubeadm has no knowledge of where certificate files live and they must be supplied.

Field	Description
endpoints [Required] []string	endpoints contains the list of etcd members.
caFile [Required] string	caFile is an SSL Certificate Authority (CA) file used to secure etcd communication. Required if using a TLS connection.
certFile [Required] string	certFile is an SSL certification file used to secure etcd communication. Required if using a TLS connection.
keyFile [Required] string	keyFile is an SSL key file used to secure etcd communication. Required if using a TLS connection.

FileDiscovery

Appears in:

- [Discovery](#)

FileDiscovery is used to specify a file or URL to a kubeconfig file from which to load cluster information.

Field	Description
kubeConfigPath [Required] string	kubeConfigPath is used to specify the actual file path or URL to the kubeconfig file from which to load cluster information.

HostPathMount

Appears in:

- [ControlPlaneComponent](#)

HostPathMount contains elements describing volumes that are mounted from the host.

Field	Description
name [Required] string	name is the name of the volume inside the Pod template.
hostPath [Required] string	hostPath is the path in the host that will be mounted inside the Pod.
mountPath [Required] string	mountPath is the path inside the Pod where hostPath will be mounted.
readOnly bool	readOnly controls write access to the volume.
pathType core/v1.HostPathType	pathType is the type of the hostPath.

ImageMeta

Appears in:

- [DNS](#)
- [LocalEtcd](#)

ImageMeta allows to customize the image used for components that are not originated from the Kubernetes/Kubernetes release process

Field	Description
imageRepository string	imageRepository sets the container registry to pull images from. If not set, the imageRepository defined in ClusterConfiguration will be used instead.
imageTag string	imageTag allows to specify a tag for the image. In case this value is set, kubeadm does not change automatically the version of the above components during upgrades.

JoinControlPlane

Appears in:

- [JoinConfiguration](#)

JoinControlPlane contains elements describing an additional control plane instance to be deployed on the joining node.

Field	Description
localAPIEndpoint APIEndpoint	localAPIEndpoint represents the endpoint of the API server instance to be deployed on this node.
certificateKey string	certificateKey is the key that is used for decryption of certificates after they are downloaded from the secret upon joining a new control plane node. The corresponding encryption key is in the InitConfiguration.

LocalEtcd

Appears in:

- [Etcd](#)

LocalEtcd describes that kubeadm should run an etcd cluster locally

Field	Description
ImageMeta [Required] ImageMeta	(Members of ImageMeta are embedded into this type.) ImageMeta allows to customize the container used for etcd.
dataDir [Required] string	dataDir is the directory etcd will place its data. Defaults to "/var/lib/etcd".
extraArgs map[string]string	extraArgs are extra arguments provided to the etcd binary when run inside a static Pod. A key in this map is the flag name as it appears on the command line except without leading dash(es).
serverCertSANs []string	serverCertSANs sets extra Subject Alternative Names (SANs) for the etcd server signing certificate.
peerCertSANs []string	peerCertSANs sets extra Subject Alternative Names (SANs) for the etcd peer signing certificate.

Networking

Appears in:

- [ClusterConfiguration](#)

Networking contains elements describing cluster's networking configuration

Field	Description
serviceSubnet string	

Field	Description
	serviceSubnet is the subnet used by Kubernetes Services. Defaults to "10.96.0.0/12".
podSubnet string	podSubnet is the subnet used by Pods.
dnsDomain string	dnsDomain is the DNS domain used by Kubernetes Services. Defaults to "cluster.local".

NodeRegistrationOptions

Appears in:

- [InitConfiguration](#)
- [JoinConfiguration](#)

NodeRegistrationOptions holds fields that relate to registering a new control-plane or node to the cluster, either via "kubeadm init" or "kubeadm join"

Field	Description
name string	name is the .metadata.name field of the Node API object that will be created in this kubeadm init or kubeadm join operation. This field is also used in the CommonName field of the kubelet's client certificate to the API server. Defaults to the hostname of the node if not provided.
criSocket string	criSocket is used to retrieve container runtime info. This information will be annotated to the Node API object, for later re-use.
taints [Required] []core/v1.Taint	taints specifies the taints the Node API object should be registered with. If this field is unset, i.e. nil, it will be defaulted with a control-plane taint for control-plane nodes. If you don't want to taint your control-plane node, set this field to an empty list, i.e. taints: [] in the YAML file. This field is solely used for Node registration.
kubeletExtraArgs map[string]string	kubeletExtraArgs passes through extra arguments to the kubelet. The arguments here are passed to the kubelet command line via the environment file kubeadm writes at runtime for the kubelet to source. This overrides the generic base-level configuration in the kubelet-config ConfigMap. Flags have higher priority when parsing. These values are local and specific to the node kubeadm is executing on. A key in this map is the flag name as it appears on the command line except without leading dash(es).

Field	Description
ignorePreflightErrors []string	ignorePreflightErrors provides a list of pre-flight errors to be ignored when the current node is registered, e.g. IsPrivilegedUser, Swap. Value all ignores errors from all checks.
imagePullPolicy core/v1.PullPolicy	imagePullPolicy specifies the policy for image pulling during kubeadm "init" and "join" operations. The value of this field must be one of "Always", "IfNotPresent" or "Never". If this field is not set, kubeadm will default it to "IfNotPresent", or pull the required images if not present on the host.

Patches

Appears in:

- [InitConfiguration](#)
- [JoinConfiguration](#)

Patches contains options related to applying patches to components deployed by kubeadm.

Field	Description
directory string	directory is a path to a directory that contains files named "target[suffix] [+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd". "patchtype" can be one of "strategic" "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

kubeadm Configuration (v1beta4)

Overview

Package v1beta4 defines the v1beta4 version of the kubeadm configuration file format. This version improves on the v1beta3 format by fixing some minor issues and adding a few new fields.

A list of changes since v1beta3:

- TODO <https://github.com/kubernetes/kubeadm/issues/2890>
- Support custom environment variables in control plane components under ClusterConfiguration. Use APIServer.ExtraEnvs, ControllerManager.ExtraEnvs, Scheduler.ExtraEnvs, Etcd.Local.ExtraEnvs.
- The ResetConfiguration API type is now supported in v1beta4. Users are able to reset a node by passing a --config file to kubeadm reset.

Migration from old kubeadm config versions

- kubeadm v1.15.x and newer can be used to migrate from v1beta1 to v1beta2.
- kubeadm v1.22.x and newer no longer support v1beta1 and older APIs, but can be used to migrate v1beta2 to v1beta3.
- kubeadm v1.27.x and newer no longer support v1beta2 and older APIs.
- TODO: <https://github.com/kubernetes/kubeadm/issues/2890> add version that can be used to convert to v1beta4

Basics

The preferred way to configure kubeadm is to pass an YAML configuration file with the `--config` option. Some of the configuration options defined in the kubeadm config file are also available as command line flags, but only the most common/simple use case are supported with this approach.

A kubeadm config file could contain multiple configuration types separated using three dashes (---).

kubeadm supports the following configuration types:

```
apiVersion: kubeadm.k8s.io/v1beta4
kind: InitConfiguration
```

```
apiVersion: kubeadm.k8s.io/v1beta4
kind: ClusterConfiguration
```

```
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
```

```
apiVersion: kubeProxy.config.k8s.io/v1alpha1
kind: KubeProxyConfiguration
```

```
apiVersion: kubeadm.k8s.io/v1beta4
kind: JoinConfiguration
```

To print the defaults for "init" and "join" actions use the following commands:

```
kubeadm config print init-defaults
kubeadm config print join-defaults
```

The list of configuration types that must be included in a configuration file depends by the action you are performing (init or join) and by the configuration options you are going to use (defaults or advanced customization).

If some configuration types are not provided, or provided only partially, kubeadm will use default values; defaults provided by kubeadm includes also enforcing consistency of values across components when required (e.g. --cluster-cidr flag on controller manager and clusterCIDR on kube-proxy).

Users are always allowed to override default values, with the only exception of a small subset of setting with relevance for security (e.g. enforce authorization-mode Node and RBAC on api server).

If the user provides a configuration types that is not expected for the action you are performing, kubeadm will ignore those types and print a warning.

Kubeadm init configuration types

When executing kubeadm init with the `--config` option, the following configuration types could be used: InitConfiguration, ClusterConfiguration, KubeProxyConfiguration, KubeletConfiguration, but only one between InitConfiguration and ClusterConfiguration is mandatory.

```
apiVersion: kubeadm.k8s.io/v1beta4
kind: InitConfiguration
bootstrapTokens:
  ...
nodeRegistration:
  ...
```

The InitConfiguration type should be used to configure runtime settings, that in case of kubeadm init are the configuration of the bootstrap token and all the setting which are specific to the node where kubeadm is executed, including:

- NodeRegistration, that holds fields that relate to registering the new node to the cluster; use it to customize the node name, the CRI socket to use or any other settings that should apply to this node only (e.g. the node ip).
- LocalAPIEndpoint, that represents the endpoint of the instance of the API server to be deployed on this node; use it e.g. to customize the API server advertise address.

```
apiVersion: kubeadm.k8s.io/v1beta4
kind: ClusterConfiguration
networking:
  ...
etcld:
  ...
apiServer:
```

```
extraArgs:  
...  
extraVolumes:  
...  
...
```

The ClusterConfiguration type should be used to configure cluster-wide settings, including settings for:

- networking that holds configuration for the networking topology of the cluster; use it e.g. to customize Pod subnet or services subnet.
- etcd: use it e.g. to customize the local etcd or to configure the API server for using an external etcd cluster.
- kube-apiserver, kube-scheduler, kube-controller-manager configurations; use it to customize control-plane components by adding customized setting or overriding kubeadm default settings.

```
apiVersion: kubeproxy.config.k8s.io/v1alpha1  
kind: KubeProxyConfiguration
```

```
...
```

The KubeProxyConfiguration type should be used to change the configuration passed to kube-proxy instances deployed in the cluster. If this object is not provided or provided only partially, kubeadm applies defaults.

See <https://kubernetes.io/docs/reference/command-line-tools-reference/kube-proxy/> or <https://pkg.go.dev/k8s.io/kube-proxy/config/v1alpha1#KubeProxyConfiguration> for kube-proxy official documentation.

```
apiVersion: kubelet.config.k8s.io/v1beta1  
kind: KubeletConfiguration
```

```
...
```

The KubeletConfiguration type should be used to change the configurations that will be passed to all kubelet instances deployed in the cluster. If this object is not provided or provided only partially, kubeadm applies defaults.

See <https://kubernetes.io/docs/reference/command-line-tools-reference/kubelet/> or <https://pkg.go.dev/k8s.io/kubelet/config/v1beta1#KubeletConfiguration> for kubelet official documentation.

Here is a fully populated example of a single YAML file containing multiple configuration types to be used during a kubeadm init run.

```
apiVersion: kubeadm.k8s.io/v1beta4  
kind: InitConfiguration  
bootstrapTokens:
```

```
- token: "9a08jv.c0izixklcxtmnze7"
  description: "kubeadm bootstrap token"
  ttl: "24h"
- token: "783bde.3f89s0fje9f38fhf"
  description: "another bootstrap token"
  usages:
- authentication
- signing
  groups:
- system:bootstrappers:kubeadm:default-node-token
```

nodeRegistration:

```
  name: "ec2-10-100-0-1"
  criSocket: "unix:///var/run/containerd/containerd.sock"
  taints:
    - key: "kubeadmNode"
      value: "someValue"
      effect: "NoSchedule"
  kubeletExtraArgs:
    v: 4
  ignorePreflightErrors:
    - IsPrivilegedUser
  imagePullPolicy: "IfNotPresent"
```

localAPIEndpoint:

```
  advertiseAddress: "10.100.0.1"
  bindPort: 6443
```

```
certificateKey: "e6a2eb8581237ab72a4f494f30285ec12a9694d750b9785706a83bfccbb2204"
skipPhases:
```

```
  - addon/kube-proxy
```

```
---
```

```
apiVersion: kubeadm.k8s.io/v1beta4
```

```
kind: ClusterConfiguration
```

```
etcd:
```

```
  # one of local or external
  local:
    imageRepository: "registry.k8s.io"
    imageTag: "3.2.24"
    dataDir: "/var/lib/etcd"
    extraArgs:
      listen-client-urls: "http://10.100.0.1:2379"
    serverCertSANs:
      - "ec2-10-100-0-1.compute-1.amazonaws.com"
    peerCertSANs:
      - "10.100.0.1"
  # external:
    # endpoints:
    # - "10.100.0.1:2379"
```

```
# - "10.100.0.2:2379"  
# caFile: "/etc/kubernetes/pki/etcd/etcd-ca.crt"  
# certFile: "/etc/kubernetes/pki/etcd/etcd.crt"  
# keyFile: "/etc/kubernetes/pki/etcd/etcd.key"
```

networking:

```
serviceSubnet: "10.96.0.0/16"  
podSubnet: "10.244.0.0/24"  
dnsDomain: "cluster.local"
```

kubernetesVersion: "v1.21.0"

controlPlaneEndpoint: "10.100.0.1:6443"

apiServer:

```
extraArgs:  
  authorization-mode: "Node,RBAC"  
extraVolumes:
```

```
- name: "some-volume"  
  hostPath: "/etc/some-path"  
  mountPath: "/etc/some-pod-path"  
  readOnly: false  
  pathType: File
```

certSANs:

```
- "10.100.1.1"  
- "ec2-10-100-0-1.compute-1.amazonaws.com"
```

timeoutForControlPlane: 4m0s

controllerManager:

```
extraArgs:  
  "node-cidr-mask-size": "20"
```

extraVolumes:

```
- name: "some-volume"  
  hostPath: "/etc/some-path"  
  mountPath: "/etc/some-pod-path"  
  readOnly: false  
  pathType: File
```

scheduler:

```
extraArgs:  
  address: "10.100.0.1"  
extraVolumes:  
- name: "some-volume"  
  hostPath: "/etc/some-path"  
  mountPath: "/etc/some-pod-path"  
  readOnly: false  
  pathType: File
```

certificatesDir: "/etc/kubernetes/pki"

imageRepository: "registry.k8s.io"

clusterName: "example-cluster"

```
---  
apiVersion: kubelet.config.k8s.io/v1beta1  
kind: KubeletConfiguration  
# kubelet specific options here  
---  
apiVersion: kubeproxy.config.k8s.io/v1alpha1  
kind: KubeProxyConfiguration  
# kube-proxy specific options here
```

Kubeadm join configuration types

When executing kubeadm join with the --config option, the JoinConfiguration type should be provided.

```
apiVersion: kubeadm.k8s.io/v1beta4  
kind: JoinConfiguration
```

```
...
```

The JoinConfiguration type should be used to configure runtime settings, that in case of kubeadm join are the discovery method used for accessing the cluster info and all the setting which are specific to the node where kubeadm is executed, including:

- nodeRegistration, that holds fields that relate to registering the new node to the cluster; use it to customize the node name, the CRI socket to use or any other settings that should apply to this node only (e.g. the node ip).
- `apiEndpoint“, that represents the endpoint of the instance of the API server to be eventually deployed on this node.

Resource Types

- [ClusterConfiguration](#)
- [InitConfiguration](#)
- [JoinConfiguration](#)
- [ResetConfiguration](#)

BootstrapToken

Appears in:

- [InitConfiguration](#)
- [InitConfiguration](#)

BootstrapToken describes one bootstrap token, stored as a Secret in the cluster

Field	Description
token [Required] BootstrapTokenString	token is used for establishing bidirectional trust between nodes and control-planes. Used for joining nodes in the cluster.
description string	description sets a human-friendly message why this token exists and what it's used for, so other administrators can know its purpose.
ttl meta/v1.Duration	ttl defines the time to live for this token. Defaults to 24h. expires and ttl are mutually exclusive.
expires meta/v1.Time	expires specifies the timestamp when this token expires. Defaults to being set dynamically at runtime based on the ttl. expires and ttl are mutually exclusive.
usages []string	usages describes the ways in which this token can be used. Can by default be used for establishing bidirectional trust, but that can be changed here.
groups []string	groups specifies the extra groups that this token will authenticate as when/if used for authentication

BootstrapTokenString

Appears in:

- [BootstrapToken](#)

BootstrapTokenString is a token of the format abcdef.abcdef0123456789 that is used for both validation of the practically of the API server from a joining node's point of view and as an authentication method for the node in the bootstrap phase of "kubeadm join". This token is and should be short-lived.

Field	Description
- [Required] string	No description provided.
- [Required] string	No description provided.

ClusterConfiguration

ClusterConfiguration contains cluster-wide configuration for a kubeadm cluster

Field	Description
	kubeadm.k8s.io/v1beta4

Field	Description
apiVersion string	
kind string	ClusterConfiguration
etcd Etcd	Etcd holds configuration for etcd.
networking Networking	Networking holds configuration for the networking topology of the cluster.
kubernetesVersion string	KubernetesVersion is the target version of the control plane.
controlPlaneEndpoint string	ControlPlaneEndpoint sets a stable IP address or DNS name for the control plane; it can be a valid IP address or a RFC-1123 DNS subdomain, both with optional TCP port. In case the ControlPlaneEndpoint is not specified, the AdvertiseAddress + BindPort are used; in case the ControlPlaneEndpoint is specified but without a TCP port, the BindPort is used. Possible usages are: e.g. In a cluster with more than one control plane instances, this field should be assigned the address of the external load balancer in front of the control plane instances. e.g. in environments with enforced node recycling, the ControlPlaneEndpoint could be used for assigning a stable DNS to the control plane.
apiServer APIServer	APIServer contains extra settings for the API server control plane component
controllerManager ControlPlaneComponent	ControllerManager contains extra settings for the controller manager control plane component
scheduler ControlPlaneComponent	Scheduler contains extra settings for the scheduler control plane component
dns DNS	DNS defines the options for the DNS add-on installed in the cluster.
certificatesDir string	CertificatesDir specifies where to store or look for all required certificates.
imageRepository string	ImageRepository sets the container registry to pull images from. If empty, registry.k8s.io will be used by default; in case of kubernetes version is a CI build (kubernetes version starts with ci/) gcr.io/k8s-staging-ci-images will be used as a default for control plane

Field	Description
	components and for kube-proxy, while registry.k8s.io will be used for all the other images.
featureGates map[string]bool	FeatureGates enabled by the user.
clusterName string	The cluster name

InitConfiguration

InitConfiguration contains a list of elements that is specific "kubeadm init"-only runtime information.

Field	Description
apiVersion string	kubeadm.k8s.io/v1beta4
kind string	InitConfiguration
bootstrapTokens []BootstrapToken	BootstrapTokens is respected at kubeadm init time and describes a set of Bootstrap Tokens to create. This information IS NOT uploaded to the kubeadm cluster configmap, partly because of its sensitive nature
nodeRegistration NodeRegistrationOptions	NodeRegistration holds fields that relate to registering the new control-plane node to the cluster
localAPIEndpoint APIEndpoint	LocalAPIEndpoint represents the endpoint of the API server instance that's deployed on this control plane node In HA setups, this differs from ClusterConfiguration.ControlPlaneEndpoint in the sense that ControlPlaneEndpoint is the global endpoint for the cluster, which then loadbalances the requests to each individual API server. This configuration object lets you customize what IP/DNS name and port the local API server advertises it's accessible on. By default, kubeadm tries to auto-detect the IP of the default interface and use that, but in case that process fails you may set the desired value here.
certificateKey string	CertificateKey sets the key with which certificates and keys are encrypted prior to being uploaded in a secret in the cluster during the uploadcerts init phase.
skipPhases []string	SkipPhases is a list of phases to skip during command execution. The list of phases can be obtained with the "kubeadm init --help" command. The flag "--skip-phases" takes precedence over this field.

Field	Description
patches Patches	Patches contains options related to applying patches to components deployed by kubeadm during "kubeadm init".

JoinConfiguration

JoinConfiguration contains elements describing a particular node.

Field	Description
apiVersion string	kubeadm.k8s.io/v1beta4
kind string	JoinConfiguration
nodeRegistration NodeRegistrationOptions	NodeRegistration holds fields that relate to registering the new control-plane node to the cluster
caCertPath string	CACertPath is the path to the SSL certificate authority used to secure communications between node and control-plane. Defaults to "/etc/kubernetes/pki/ca.crt".
discovery [Required] Discovery	Discovery specifies the options for the kubelet to use during the TLS Bootstrap process
controlPlane JoinControlPlane	ControlPlane defines the additional control plane instance to be deployed on the joining node. If nil, no additional control plane instance will be deployed.
skipPhases []string	SkipPhases is a list of phases to skip during command execution. The list of phases can be obtained with the "kubeadm join --help" command. The flag "--skip-phases" takes precedence over this field.
patches Patches	Patches contains options related to applying patches to components deployed by kubeadm during "kubeadm join".

ResetConfiguration

ResetConfiguration contains a list of fields that are specifically "kubeadm reset"-only runtime information.

Field	Description
apiVersion string	kubeadm.k8s.io/v1beta4

Field	Description
kind string	ResetConfiguration
cleanupTmpDir bool	CleanupTmpDir specifies whether the "/etc/kubernetes/tmp" directory should be cleaned during the reset process.
certificatesDir string	CertificatesDir specifies the directory where the certificates are stored. If specified, it will be cleaned during the reset process.
criSocket string	CRISocket is used to retrieve container runtime info and used for the removal of the containers. If CRISocket is not specified by flag or config file, kubeadm will try to detect one valid CRISocket instead.
dryRun bool	DryRun tells if the dry run mode is enabled, don't apply any change if it is and just output what would be done.
force bool	Force flag instructs kubeadm to reset the node without prompting for confirmation.
ignorePreflightErrors []string	IgnorePreflightErrors provides a slice of pre-flight errors to be ignored during the reset process, e.g. 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
skipPhases []string	SkipPhases is a list of phases to skip during command execution. The list of phases can be obtained with the "kubeadm reset phase --help" command.

APIEndpoint

Appears in:

- [InitConfiguration](#)
- [JoinControlPlane](#)

APIEndpoint struct contains elements of API server instance deployed on a node.

Field	Description
advertiseAddress string	AdvertiseAddress sets the IP address for the API server to advertise.
bindPort int32	BindPort sets the secure port for the API Server to bind to. Defaults to 6443.

APIServer

Appears in:

- [ClusterConfiguration](#)

APIServer holds settings necessary for API server deployments in the cluster

Field	Description
ControlPlaneComponent [Required] ControlPlaneComponent	(Members of ControlPlaneComponent are embedded into this type.) No description provided.
certSANs []string	CertSANs sets extra Subject Alternative Names for the API Server signing cert.
timeoutForControlPlane meta/v1.Duration	TimeoutForControlPlane controls the timeout that we use for API server to appear

BootstrapTokenDiscovery

Appears in:

- [Discovery](#)

BootstrapTokenDiscovery is used to set the options for bootstrap token based discovery

Field	Description
token [Required] string	Token is a token used to validate cluster information fetched from the control-plane.
apiServerEndpoint string	APIServerEndpoint is an IP or domain name to the API server from which info will be fetched.
caCertHashes []string	CA Cert Hashes specifies a set of public key pins to verify when token-based discovery is used. The root CA found during discovery must match one of these values. Specifying an empty set disables root CA pinning, which can be unsafe. Each hash is specified as ":" , where the only currently supported type is "sha256". This is a hex-encoded SHA-256 hash of the Subject Public Key Info (SPKI) object in DER-encoded ASN.1. These hashes can be calculated using, for example, OpenSSL.

Field	Description
unsafeSkipCAVerification bool	UnsafeSkipCAVerification allows token-based discovery without CA verification via CACertHashes. This can weaken the security of kubeadm since other nodes can impersonate the control-plane.

ControlPlaneComponent

Appears in:

- [ClusterConfiguration](#)
- [APIServer](#)

ControlPlaneComponent holds settings common to control plane component of the cluster

Field	Description
extraArgs map[string]string	ExtraArgs is an extra set of flags to pass to the control plane component. A key in this map is the flag name as it appears on the command line except without leading dash(es). TODO: This is temporary and ideally we would like to switch all components to use ComponentConfig + ConfigMaps.
extraVolumes []HostPathMount	ExtraVolumes is an extra set of host volumes, mounted to the control plane component.
extraEnvs []core/v1.EnvVar	ExtraEnvs is an extra set of environment variables to pass to the control plane component. Environment variables passed using ExtraEnvs will override any existing environment variables, or *_proxy environment variables that kubeadm adds by default.

DNS

Appears in:

- [ClusterConfiguration](#)

DNS defines the DNS addon that should be used in the cluster

Field	Description
ImageMeta [Required] ImageMeta	(Members of ImageMeta are embedded into this type.) ImageMeta allows to customize the image used for the DNS component

Discovery

Appears in:

- [JoinConfiguration](#)

Discovery specifies the options for the kubelet to use during the TLS Bootstrap process

Field	Description
bootstrapToken BootstrapTokenDiscovery	BootstrapToken is used to set the options for bootstrap token based discovery BootstrapToken and File are mutually exclusive
file FileDiscovery	File is used to specify a file or URL to a kubeconfig file from which to load cluster information BootstrapToken and File are mutually exclusive
tlsBootstrapToken string	TLSBootstrapToken is a token used for TLS bootstrapping. If .BootstrapToken is set, this field is defaulted to .BootstrapToken.Token, but can be overridden. If .File is set, this field must be set in case the KubeConfigFile does not contain any other authentication information
timeout meta/v1.Duration	Timeout modifies the discovery timeout

Etcd

Appears in:

- [ClusterConfiguration](#)

Etcd contains elements describing Etcd configuration.

Field	Description
local LocalEtcd	Local provides configuration knobs for configuring the local etcd instance Local and External are mutually exclusive
external ExternalEtcd	External describes how to connect to an external etcd cluster Local and External are mutually exclusive

ExternalEtcd

Appears in:

- [Etcd](#)

ExternalEtcd describes an external etcd cluster. Kubeadm has no knowledge of where certificate files live and they must be supplied.

Field	Description
endpoints [Required] []string	Endpoints of etcd members. Required for ExternalEtcd.
caFile [Required] string	CAFile is an SSL Certificate Authority file used to secure etcd communication. Required if using a TLS connection.
certFile [Required] string	CertFile is an SSL certification file used to secure etcd communication. Required if using a TLS connection.
keyFile [Required] string	KeyFile is an SSL key file used to secure etcd communication. Required if using a TLS connection.

FileDiscovery

Appears in:

- [Discovery](#)

FileDiscovery is used to specify a file or URL to a kubeconfig file from which to load cluster information

Field	Description
kubeConfigPath [Required] string	KubeConfigPath is used to specify the actual file path or URL to the kubeconfig file from which to load cluster information

HostPathMount

Appears in:

- [ControlPlaneComponent](#)

HostPathMount contains elements describing volumes that are mounted from the host.

Field	Description
name [Required] string	Name of the volume inside the pod template.
hostPath [Required] string	HostPath is the path in the host that will be mounted inside the pod.

Field	Description
mountPath [Required] string	MountPath is the path inside the pod where hostPath will be mounted.
readOnly bool	ReadOnly controls write access to the volume
pathType core/v1.HostPathType	PathType is the type of the HostPath.

ImageMeta

Appears in:

- [DNS](#)
- [LocalEtcd](#)

ImageMeta allows to customize the image used for components that are not originated from the Kubernetes/Kubernetes release process

Field	Description
imageRepository string	ImageRepository sets the container registry to pull images from. if not set, the ImageRepository defined in ClusterConfiguration will be used instead.
imageTag string	ImageTag allows to specify a tag for the image. In case this value is set, kubeadm does not change automatically the version of the above components during upgrades.

JoinControlPlane

Appears in:

- [JoinConfiguration](#)

JoinControlPlane contains elements describing an additional control plane instance to be deployed on the joining node.

Field	Description
localAPIEndpoint APIEndpoint	LocalAPIEndpoint represents the endpoint of the API server instance to be deployed on this node.
certificateKey string	CertificateKey is the key that is used for decryption of certificates after they are downloaded from the secret upon joining a new control plane node. The corresponding encryption key is in the InitConfiguration.

LocalEtcd

Appears in:

- [Etc](#)

LocalEtcd describes that kubeadm should run an etcd cluster locally

Field	Description
ImageMeta [Required] ImageMeta	(Members of ImageMeta are embedded into this type.) ImageMeta allows to customize the container used for etcd
dataDir [Required] string	DataDir is the directory etcd will place its data. Defaults to "/var/lib/etcd".
extraArgs map[string]string	ExtraArgs are extra arguments provided to the etcd binary when run inside a static pod. A key in this map is the flag name as it appears on the command line except without leading dash(es).
extraEnvs []core/v1.EnvVar	ExtraEnvs is an extra set of environment variables to pass to the control plane component. Environment variables passed using ExtraEnvs will override any existing environment variables, or *_proxy environment variables that kubeadm adds by default.
serverCertSANs []string	ServerCertSANs sets extra Subject Alternative Names for the etcd server signing cert.
peerCertSANs []string	PeerCertSANs sets extra Subject Alternative Names for the etcd peer signing cert.

Networking

Appears in:

- [ClusterConfiguration](#)

Networking contains elements describing cluster's networking configuration

Field	Description
serviceSubnet string	ServiceSubnet is the subnet used by k8s services. Defaults to "10.96.0.0/12".
podSubnet string	PodSubnet is the subnet used by pods.

Field	Description
dnsDomain string	DNSDomain is the dns domain used by k8s services. Defaults to "cluster.local".

NodeRegistrationOptions

Appears in:

- [InitConfiguration](#)
- [JoinConfiguration](#)

NodeRegistrationOptions holds fields that relate to registering a new control-plane or node to the cluster, either via "kubeadm init" or "kubeadm join"

Field	Description
name string	Name is the .Metadata.Name field of the Node API object that will be created in this kubeadm init or kubeadm join operation. This field is also used in the CommonName field of the kubelet's client certificate to the API server. Defaults to the hostname of the node if not provided.
criSocket string	CRISocket is used to retrieve container runtime info. This information will be annotated to the Node API object, for later re-use
taints [Required] []core/v1.Taint	Taints specifies the taints the Node API object should be registered with. If this field is unset, i.e. nil, it will be defaulted with a control-plane taint for control-plane nodes. If you don't want to taint your control-plane node, set this field to an empty slice, i.e. taints: [] in the YAML file. This field is solely used for Node registration.
kubeletExtraArgs map[string]string	KubeletExtraArgs passes through extra arguments to the kubelet. The arguments here are passed to the kubelet command line via the environment file kubeadm writes at runtime for the kubelet to source. This overrides the generic base-level configuration in the kubelet-config ConfigMap Flags have higher priority when parsing. These values are local and specific to the node kubeadm is executing on. A key in this map is the flag name as it appears on the command line except without leading dash(es).
ignorePreflightErrors []string	IgnorePreflightErrors provides a slice of pre-flight errors to be ignored when the current node is registered, e.g. 'IsPrivilegedUser,Swap'. Value 'all' ignores errors from all checks.
imagePullPolicy core/v1.PullPolicy	ImagePullPolicy specifies the policy for image pulling during kubeadm "init" and "join" operations. The value of this field must be one of "Always", "IfNotPresent" or "Never". If this field is unset kubeadm will

Field	Description
	default it to "IfNotPresent", or pull the required images if not present on the host.

Patches

Appears in:

- [InitConfiguration](#)
- [JoinConfiguration](#)

Patches contains options related to applying patches to components deployed by kubeadm.

Field	Description
directory string	Directory is a path to a directory that contains files named "target[suffix] [+patchtype].extension". For example, "kube-apiserver0+merge.yaml" or just "etcd.json". "target" can be one of "kube-apiserver", "kube-controller-manager", "kube-scheduler", "etcd", "kubeletconfiguration". "patchtype" can be one of "strategic" "merge" or "json" and they match the patch formats supported by kubectl. The default "patchtype" is "strategic". "extension" must be either "json" or "yaml". "suffix" is an optional string that can be used to determine which patches are applied first alpha-numerically.

kubeconfig (v1)

Resource Types

- [Config](#)

Config

Config holds the information needed to build connect to remote kubernetes clusters as a given user

Field	Description
apiVersion string	/v1
kind string	Config
kind string	Legacy field from pkg/api/types.go TypeMeta. TODO(jlowdermilk): remove this after eliminating downstream dependencies.
apiVersion string	

Field	Description
	Legacy field from pkg/api/types.go TypeMeta. TODO(jlowdermilk): remove this after eliminating downstream dependencies.
preferences [Required] Preferences	Preferences holds general information to be use for cli interactions
clusters [Required] []NamedCluster	Clusters is a map of referencable names to cluster configs
users [Required] []NamedAuthInfo	AuthInfos is a map of referencable names to user configs
contexts [Required] []NamedContext	Contexts is a map of referencable names to context configs
current-context [Required] string	CurrentContext is the name of the context that you would like to use by default
extensions []NamedExtension	Extensions holds additional information. This is useful for extenders so that reads and writes don't clobber unknown fields

AuthInfo

Appears in:

- [NamedAuthInfo](#)

AuthInfo contains information that describes identity information. This is use to tell the kubernetes cluster who you are.

Field	Description
client-certificate string	ClientCertificate is the path to a client cert file for TLS.
client-certificate-data []byte	ClientCertificateData contains PEM-encoded data from a client cert file for TLS. Overrides ClientCertificate
client-key string	ClientKey is the path to a client key file for TLS.
client-key-data []byte	ClientKeyData contains PEM-encoded data from a client key file for TLS. Overrides ClientKey
token string	Token is the bearer token for authentication to the kubernetes cluster.

Field	Description
tokenFile string	TokenFile is a pointer to a file that contains a bearer token (as described above). If both Token and TokenFile are present, Token takes precedence.
as string	Impersonate is the username to impersonate. The name matches the flag.
as-uid string	ImpersonateUID is the uid to impersonate.
as-groups []string	ImpersonateGroups is the groups to impersonate.
as-user-extra map[string][]string	ImpersonateUserExtra contains additional information for impersonated user.
username string	Username is the username for basic authentication to the kubernetes cluster.
password string	Password is the password for basic authentication to the kubernetes cluster.
auth-provider AuthProviderConfig	AuthProvider specifies a custom authentication plugin for the kubernetes cluster.
exec ExecConfig	Exec specifies a custom exec-based authentication plugin for the kubernetes cluster.
extensions [] NamedExtension	Extensions holds additional information. This is useful for extenders so that reads and writes don't clobber unknown fields

AuthProviderConfig

Appears in:

- [AuthInfo](#)

AuthProviderConfig holds the configuration for a specified auth provider.

Field	Description
name [Required] string	No description provided.
config [Required] map[string]string	No description provided.

Cluster

Appears in:

- [NamedCluster](#)

Cluster contains information about how to communicate with a kubernetes cluster

Field	Description
server [Required] string	Server is the address of the kubernetes cluster (https://hostname:port).
tls-server-name string	TLS Server Name is used to check server certificate. If TLS Server Name is empty, the hostname used to contact the server is used.
insecure-skip-tls-verify bool	Insecure Skip TLS Verify skips the validity check for the server's certificate. This will make your HTTPS connections insecure.
certificate-authority string	Certificate Authority is the path to a cert file for the certificate authority.
certificate-authority-data []byte	Certificate Authority Data contains PEM-encoded certificate authority certificates. Overrides Certificate Authority
proxy-url string	Proxy URL is the URL to the proxy to be used for all requests made by this client. URLs with "http", "https", and "socks5" schemes are supported. If this configuration is not provided or the empty string, the client attempts to construct a proxy configuration from http_proxy and https_proxy environment variables. If these environment variables are not set, the client does not attempt to proxy requests. socks5 proxying does not currently support spdy streaming endpoints (exec, attach, port forward).
disable-compression bool	Disable Compression allows client to opt-out of response compression for all requests to the server. This is useful to speed up requests (specifically lists) when client-server network bandwidth is ample, by saving time on compression (server-side) and decompression (client-side): https://github.com/kubernetes/kubernetes/issues/112296 .
extensions []NamedExtension	Extensions holds additional information. This is useful for extenders so that reads and writes don't clobber unknown fields

Context

Appears in:

- [NamedContext](#)

Context is a tuple of references to a cluster (how do I communicate with a kubernetes cluster), a user (how do I identify myself), and a namespace (what subset of resources do I want to work with)

Field	Description
cluster [Required] string	Cluster is the name of the cluster for this context
user [Required] string	AuthInfo is the name of the authInfo for this context
namespace string	Namespace is the default namespace to use on unspecified requests
extensions []NamedExtension	Extensions holds additional information. This is useful for extenders so that reads and writes don't clobber unknown fields

ExecConfig

Appears in:

- [AuthInfo](#)

ExecConfig specifies a command to provide client credentials. The command is exec'd and outputs structured stdout holding credentials.

See the client.authentication.k8s.io API group for specifications of the exact input and output format

Field	Description
command [Required] string	Command to execute.
args []string	Arguments to pass to the command when executing it.
env []ExecEnvVar	Env defines additional environment variables to expose to the process. These are unioned with the host's environment, as well as variables client-go uses to pass argument to the plugin.

Field	Description
apiVersion [Required] string	Preferred input version of the ExecInfo. The returned ExecCredentials MUST use the same encoding version as the input.
installHint [Required] string	This text is shown to the user when the executable doesn't seem to be present. For example, brew install foo-cli might be a good InstallHint for foo-cli on Mac OS systems.
provideClusterInfo [Required] bool	ProvideClusterInfo determines whether or not to provide cluster information, which could potentially contain very large CA data, to this exec plugin as a part of the KUBERNETES_EXEC_INFO environment variable. By default, it is set to false. Package k8s.io/client-go/tools/auth/exec provides helper methods for reading this environment variable.
interactiveMode ExecInteractiveMode	<p>InteractiveMode determines this plugin's relationship with standard input. Valid values are "Never" (this exec plugin never uses standard input), "IfAvailable" (this exec plugin wants to use standard input if it is available), or "Always" (this exec plugin requires standard input to function). See ExecInteractiveMode values for more details.</p> <p>If APIVersion is client.authentication.k8s.io/v1alpha1 or client.authentication.k8s.io/v1beta1, then this field is optional and defaults to "IfAvailable" when unset. Otherwise, this field is required.</p>

ExecEnvVar

Appears in:

- [ExecConfig](#)

ExecEnvVar is used for setting environment variables when executing an exec-based credential plugin.

Field	Description
name [Required] string	No description provided.
value [Required] string	No description provided.

ExecInteractiveMode

(Alias of string)

Appears in:

- [ExecConfig](#)

ExecInteractiveMode is a string that describes an exec plugin's relationship with standard input.

NamedAuthInfo

Appears in:

- [Config](#)

NamedAuthInfo relates nicknames to auth information

Field	Description
name [Required] string	Name is the nickname for this AuthInfo
user [Required] AuthInfo	AuthInfo holds the auth information

NamedCluster

Appears in:

- [Config](#)

NamedCluster relates nicknames to cluster information

Field	Description
name [Required] string	Name is the nickname for this Cluster
cluster [Required] Cluster	Cluster holds the cluster information

NamedContext

Appears in:

- [Config](#)

NamedContext relates nicknames to context information

Field	Description
name [Required] string	Name is the nickname for this Context
context [Required] Context	Context holds the context information

NamedExtension

Appears in:

- [Config](#)
- [AuthInfo](#)
- [Cluster](#)
- [Context](#)
- [Preferences](#)

NamedExtension relates nicknames to extension information

Field	Description
name [Required] string	Name is the nickname for this Extension
extension [Required] k8s.io/apimachinery/pkg/runtime.RawExtension	Extension holds the extension information

Preferences

Appears in:

- [Config](#)

Field	Description
colors bool	No description provided.
extensions []NamedExtension	Extensions holds additional information. This is useful for extenders so that reads and writes don't clobber unknown fields

Kubelet Configuration (v1)

Resource Types

- [CredentialProviderConfig](#)

CredentialProviderConfig

CredentialProviderConfig is the configuration containing information about each exec credential provider. Kubelet reads this configuration from disk and enables each provider as specified by the CredentialProvider type.

Field	Description
apiVersion string	kubelet.config.k8s.io/v1
kind string	CredentialProviderConfig
providers [Required] []CredentialProvider	providers is a list of credential provider plugins that will be enabled by the kubelet. Multiple providers may match against a single image, in which case credentials from all providers will be returned to the kubelet. If multiple providers are called for a single image, the results are combined. If providers return overlapping auth keys, the value from the provider earlier in this list is used.

CredentialProvider

Appears in:

- [CredentialProviderConfig](#)

CredentialProvider represents an exec plugin to be invoked by the kubelet. The plugin is only invoked when an image being pulled matches the images handled by the plugin (see matchImages).

Field	Description
name [Required] string	name is the required name of the credential provider. It must match the name of the provider executable as seen by the kubelet. The executable must be in the kubelet's bin directory (set by the --image-credential-provider-bin-dir flag).
matchImages [Required] []string	<p>matchImages is a required list of strings used to match against images in order to determine if this provider should be invoked. If one of the strings matches the requested image from the kubelet, the plugin will be invoked and given a chance to provide credentials. Images are expected to contain the registry domain and URL path.</p> <p>Each entry in matchImages is a pattern which can optionally contain a port and a path. Globs can be used in the domain, but not in the port or the path. Globs are supported as subdomains like '*.k8s.io' or 'k8s.*.io', and top-level-domains such as 'k8s.*'. Matching partial subdomains like 'app.k8s.io' is also supported. Each glob can only match a single subdomain segment, so *.io does not match *.k8s.io.</p> <p>A match exists between an image and a matchImage when all of the below are true:</p> <ul style="list-style-type: none"> • Both contain the same number of domain parts and each part matches. • The URL path of an imageMatch must be a prefix of the target image URL path.

Field	Description
	<ul style="list-style-type: none"> If the imageMatch contains a port, then the port must match in the image as well. <p>Example values of matchImages:</p> <ul style="list-style-type: none"> 123456789.dkr.ecr.us-east-1.amazonaws.com *.azurecr.io gcr.io **.registry.io registry.io:8080/path
defaultCacheDuration [Required] meta/v1.Duration	defaultCacheDuration is the default duration the plugin will cache credentials in-memory if a cache duration is not provided in the plugin response. This field is required.
apiVersion [Required] string	Required input version of the exec CredentialProviderRequest. The returned CredentialProviderResponse MUST use the same encoding version as the input. Current supported values are: <ul style="list-style-type: none"> credentialprovider.kubelet.k8s.io/v1
args []string	Arguments to pass to the command when executing it.
env [] ExecEnvVar	Env defines additional environment variables to expose to the process. These are unioned with the host's environment, as well as variables client-go uses to pass argument to the plugin.

ExecEnvVar

Appears in:

- [CredentialProvider](#)

ExecEnvVar is used for setting environment variables when executing an exec-based credential plugin.

Field	Description
name [Required] string	No description provided.
value [Required] string	No description provided.

Kubelet Configuration (v1alpha1)

Resource Types

- [CredentialProviderConfig](#)

CredentialProviderConfig

CredentialProviderConfig is the configuration containing information about each exec credential provider. Kubelet reads this configuration from disk and enables each provider as specified by the CredentialProvider type.

Field	Description
apiVersion string	kubelet.config.k8s.io/v1alpha1
kind string	CredentialProviderConfig
providers [Required] []CredentialProvider	providers is a list of credential provider plugins that will be enabled by the kubelet. Multiple providers may match against a single image, in which case credentials from all providers will be returned to the kubelet. If multiple providers are called for a single image, the results are combined. If providers return overlapping auth keys, the value from the provider earlier in this list is used.

CredentialProvider

Appears in:

- [CredentialProviderConfig](#)

CredentialProvider represents an exec plugin to be invoked by the kubelet. The plugin is only invoked when an image being pulled matches the images handled by the plugin (see matchImages).

Field	Description
name [Required] string	name is the required name of the credential provider. It must match the name of the provider executable as seen by the kubelet. The executable must be in the kubelet's bin directory (set by the --image-credential-provider-bin-dir flag).
matchImages [Required] []string	matchImages is a required list of strings used to match against images in order to determine if this provider should be invoked. If one of the strings matches the requested image from the kubelet, the plugin will be invoked and given a chance to provide credentials. Images are expected to contain the registry domain and URL path.

Field	Description
	<p>Each entry in matchImages is a pattern which can optionally contain a port and a path. Globs can be used in the domain, but not in the port or the path. Globs are supported as subdomains like *.k8s.io or k8s.*.io, and top-level-domains such as k8s.*. Matching partial subdomains like app*.k8s.io is also supported. Each glob can only match a single subdomain segment, so *.io does not match *.k8s.io.</p> <p>A match exists between an image and a matchImage when all of the below are true:</p> <ul style="list-style-type: none"> • Both contain the same number of domain parts and each part matches. • The URL path of an imageMatch must be a prefix of the target image URL path. • If the imageMatch contains a port, then the port must match in the image as well. <p>Example values of matchImages:</p> <ul style="list-style-type: none"> • 123456789.dkr.ecr.us-east-1.amazonaws.com • *.azurecr.io • gcr.io • *.*.registry.io • registry.io:8080/path
defaultCacheDuration [Required] meta/v1.Duration	<p>defaultCacheDuration is the default duration the plugin will cache credentials in-memory if a cache duration is not provided in the plugin response. This field is required.</p>
apiVersion [Required] string	<p>Required input version of the exec CredentialProviderRequest. The returned CredentialProviderResponse MUST use the same encoding version as the input. Current supported values are:</p> <ul style="list-style-type: none"> • credentialprovider.kubelet.k8s.io/v1alpha1
args []string	<p>Arguments to pass to the command when executing it.</p>
env []ExecEnvVar	<p>Env defines additional environment variables to expose to the process. These are unioned with the host's environment, as well as variables client-go uses to pass argument to the plugin.</p>

ExecEnvVar

Appears in:

- [CredentialProvider](#)

ExecEnvVar is used for setting environment variables when executing an exec-based credential plugin.

Field	Description
name [Required] string	No description provided.
value [Required] string	No description provided.

Kubelet Configuration (v1beta1)

Resource Types

- [CredentialProviderConfig](#)
- [KubeletConfiguration](#)
- [SerializedNodeConfigSource](#)

FormatOptions

Appears in:

- [LoggingConfiguration](#)

FormatOptions contains options for the different logging formats.

Field	Description
json [Required] JSONOptions	[Alpha] JSON contains options for logging format "json". Only available when the LoggingAlphaOptions feature gate is enabled.

JSONOptions

Appears in:

- [FormatOptions](#)

JSONOptions contains options for logging format "json".

Field	Description
splitStream [Required] bool	[Alpha] SplitStream redirects error messages to stderr while info messages go to stdout, with buffering. The default is to write both to stdout, without buffering. Only available when the LoggingAlphaOptions feature gate is enabled.
infoBufferSize [Required] k8s.io/apimachinery/pkg/api/resource.QuantityValue	[Alpha] InfoBufferSize sets the size of the info stream when using split streams. The default is zero, which disables buffering.

Field	Description
	Only available when the LoggingAlphaOptions feature gate is enabled.

LogFormatFactory

LogFormatFactory provides support for a certain additional, non-default log format.

LoggingConfiguration

Appears in:

- [KubeletConfiguration](#)

LoggingConfiguration contains logging options.

Field	Description
format [Required] string	Format Flag specifies the structure of log messages. default value of format is text
flushFrequency [Required] TimeOrMetaDuration	Maximum time between log flushes. If a string, parsed as a duration (i.e. "1s") If an int, the maximum number of nanoseconds (i.e. 1s = 1000000000). Ignored if the selected logging backend writes log messages without buffering.
verbosity [Required] VerbosityLevel	Verbosity is the threshold that determines which log messages are logged. Default is zero which logs only the most important messages. Higher values enable additional messages. Error messages are always logged.
vmodule [Required] VMModuleConfiguration	VModule overrides the verbosity threshold for individual files. Only supported for "text" log format.
options [Required] FormatOptions	[Alpha] Options holds additional parameters that are specific to the different logging formats. Only the options for the selected format get used, but all of them get validated. Only available when the LoggingAlphaOptions feature gate is enabled.

LoggingOptions

LoggingOptions can be used with ValidateAndApplyWithOptions to override certain global defaults.

Field	Description
ErrorStream [Required] io.Writer	ErrorStream can be used to override the os.Stderr default.
InfoStream [Required] io.Writer	InfoStream can be used to override the os.Stdout default.

TimeOrMetaDuration

Appears in:

- [LoggingConfiguration](#)

TimeOrMetaDuration is present only for backwards compatibility for the flushFrequency field, and new fields should use metav1.Duration.

Field	Description
Duration [Required] meta/v1.Duration	Duration holds the duration
- [Required] bool	SerializeAsString controls whether the value is serialized as a string or an integer

TracingConfiguration

Appears in:

- [KubeletConfiguration](#)

TracingConfiguration provides versioned configuration for OpenTelemetry tracing clients.

Field	Description
endpoint string	Endpoint of the collector this component will report traces to. The connection is insecure, and does not currently support TLS. Recommended is unset, and endpoint is the otlp grpc default, localhost:4317.
samplingRatePerMillion int32	SamplingRatePerMillion is the number of samples to collect per million spans. Recommended is unset. If unset, sampler respects its parent span's sampling rate, but otherwise never samples.

VModuleConfiguration

(Alias of []k8s.io/component-base/logs/api/v1.VModuleItem)

Appears in:

- [LoggingConfiguration](#)

VModuleConfiguration is a collection of individual file names or patterns and the corresponding verbosity threshold.

VerbosityLevel

(Alias of uint32)

Appears in:

- [LoggingConfiguration](#)

VerbosityLevel represents a klog or logr verbosity threshold.

CredentialProviderConfig

CredentialProviderConfig is the configuration containing information about each exec credential provider. Kubelet reads this configuration from disk and enables each provider as specified by the CredentialProvider type.

Field	Description
apiVersion string	kubelet.config.k8s.io/v1beta1
kind string	CredentialProviderConfig
providers [Required] []CredentialProvider	providers is a list of credential provider plugins that will be enabled by the kubelet. Multiple providers may match against a single image, in which case credentials from all providers will be returned to the kubelet. If multiple providers are called for a single image, the results are combined. If providers return overlapping auth keys, the value from the provider earlier in this list is used.

KubeletConfiguration

KubeletConfiguration contains the configuration for the Kubelet

Field	Description
apiVersion string	kubelet.config.k8s.io/v1beta1
kind string	KubeletConfiguration
enableServer [Required] bool	enableServer enables Kubelet's secured server. Note: Kubelet's insecure port is controlled by the readOnlyPort option. Default: true

Field	Description
staticPodPath string	staticPodPath is the path to the directory containing local (static) pods to run, or the path to a single static pod file. Default: ""
syncFrequency meta/v1.Duration	syncFrequency is the max period between synchronizing running containers and config. Default: "1m"
fileCheckFrequency meta/v1.Duration	fileCheckFrequency is the duration between checking config files for new data. Default: "20s"
httpCheckFrequency meta/v1.Duration	httpCheckFrequency is the duration between checking http for new data. Default: "20s"
staticPodURL string	staticPodURL is the URL for accessing static pods to run. Default: ""
staticPodURLHeader map[string][]string	staticPodURLHeader is a map of slices with HTTP headers to use when accessing the podURL. Default: nil
address string	address is the IP address for the Kubelet to serve on (set to 0.0.0.0 for all interfaces). Default: "0.0.0.0"
port int32	port is the port for the Kubelet to serve on. The port number must be between 1 and 65535, inclusive. Default: 10250
readOnlyPort int32	readOnlyPort is the read-only port for the Kubelet to serve on with no authentication/authorization. The port number must be between 1 and 65535, inclusive. Setting this field to 0 disables the read-only service. Default: 0 (disabled)

Field	Description
tlsCertFile string	tlsCertFile is the file containing x509 Certificate for HTTPS. (CA cert, if any, concatenated after server cert). If tlsCertFile and tlsPrivateKeyFile are not provided, a self-signed certificate and key are generated for the public address and saved to the directory passed to the Kubelet's --cert-dir flag. Default: ""
tlsPrivateKeyFile string	tlsPrivateKeyFile is the file containing x509 private key matching tlsCertFile. Default: ""
tlsCipherSuites []string	tlsCipherSuites is the list of allowed cipher suites for the server. Note that TLS 1.3 ciphersuites are not configurable. Values are from tls package constants (https://golang.org/pkg/crypto/tls/#pkg-constants). Default: nil
tlsMinVersion string	tlsMinVersion is the minimum TLS version supported. Values are from tls package constants (https://golang.org/pkg/crypto/tls/#pkg-constants). Default: ""
rotateCertificates bool	rotateCertificates enables client certificate rotation. The Kubelet will request a new certificate from the certificates.k8s.io API. This requires an approver to approve the certificate signing requests. Default: false
serverTLSBootstrap bool	serverTLSBootstrap enables server certificate bootstrap. Instead of self signing a serving certificate, the Kubelet will request a certificate from the 'certificates.k8s.io' API. This requires an approver to approve the certificate signing requests (CSR). The RotateKubeletServerCertificate feature must be enabled when setting this field. Default: false
authentication KubeletAuthentication	authentication specifies how requests to the Kubelet's server are authenticated. Defaults: anonymous: enabled: false webhook: enabled: true cacheTTL: "2m"
authorization KubeletAuthorization	authorization specifies how requests to the Kubelet's server are authorized. Defaults: mode:

Field	Description
	Webhook webhook: cacheAuthorizedTTL: "5m" cacheUnauthorizedTTL: "30s"
registryPullQPS int32	registryPullQPS is the limit of registry pulls per second. The value must not be a negative number. Setting it to 0 means no limit. Default: 5
registryBurst int32	registryBurst is the maximum size of bursty pulls, temporarily allows pulls to burst to this number, while still not exceeding registryPullQPS. The value must not be a negative number. Only used if registryPullQPS is greater than 0. Default: 10
eventRecordQPS int32	eventRecordQPS is the maximum event creations per second. If 0, there is no limit enforced. The value cannot be a negative number. Default: 50
eventBurst int32	eventBurst is the maximum size of a burst of event creations, temporarily allows event creations to burst to this number, while still not exceeding eventRecordQPS. This field canot be a negative number and it is only used when eventRecordQPS > 0. Default: 100
enableDebuggingHandlers bool	enableDebuggingHandlers enables server endpoints for log access and local running of containers and commands, including the exec, attach, logs, and portforward features. Default: true
enableContentionProfiling bool	enableContentionProfiling enables block profiling, if enableDebuggingHandlers is true. Default: false
healthzPort int32	healthzPort is the port of the localhost healthz endpoint (set to 0 to disable). A valid number is between 1 and 65535. Default: 10248
healthzBindAddress string	healthzBindAddress is the IP address for the healthz server to serve on. Default: "127.0.0.1"

Field	Description
oomScoreAdj int32	oomScoreAdj is The oom-score-adj value for kubelet process. Values must be within the range [-1000, 1000]. Default: -999
clusterDomain string	clusterDomain is the DNS domain for this cluster. If set, kubelet will configure all containers to search this domain in addition to the host's search domains. Default: ""
clusterDNS []string	clusterDNS is a list of IP addresses for the cluster DNS server. If set, kubelet will configure all containers to use this for DNS resolution instead of the host's DNS servers. Default: nil
streamingConnectionIdleTimeout meta/v1.Duration	streamingConnectionIdleTimeout is the maximum time a streaming connection can be idle before the connection is automatically closed. Default: "4h"
nodeStatusUpdateFrequency meta/v1.Duration	nodeStatusUpdateFrequency is the frequency that kubelet computes node status. If node lease feature is not enabled, it is also the frequency that kubelet posts node status to master. Note: When node lease feature is not enabled, be cautious when changing the constant, it must work with nodeMonitorGracePeriod in nodecontroller. Default: "10s"
nodeStatusReportFrequency meta/v1.Duration	nodeStatusReportFrequency is the frequency that kubelet posts node status to master if node status does not change. Kubelet will ignore this frequency and post node status immediately if any change is detected. It is only used when node lease feature is enabled. nodeStatusReportFrequency's default value is 5m. But if nodeStatusUpdateFrequency is set explicitly, nodeStatusReportFrequency's default value will be set to nodeStatusUpdateFrequency for backward compatibility. Default: "5m"
nodeLeaseDurationSeconds int32	nodeLeaseDurationSeconds is the duration the Kubelet will set on its corresponding Lease. NodeLease provides an indicator of node health by having the Kubelet create and periodically renew a lease, named after the node, in the kube-node-lease namespace. If the lease expires, the node can be

Field	Description
	considered unhealthy. The lease is currently renewed every 10s, per KEP-0009. In the future, the lease renewal interval may be set based on the lease duration. The field value must be greater than 0. Default: 40
imageMinimumGCAge meta/v1.Duration	imageMinimumGCAge is the minimum age for an unused image before it is garbage collected. Default: "2m"
imageGCHighThresholdPercent int32	imageGCHighThresholdPercent is the percent of disk usage after which image garbage collection is always run. The percent is calculated by dividing this field value by 100, so this field must be between 0 and 100, inclusive. When specified, the value must be greater than imageGCLowThresholdPercent. Default: 85
imageGCLowThresholdPercent int32	imageGCLowThresholdPercent is the percent of disk usage before which image garbage collection is never run. Lowest disk usage to garbage collect to. The percent is calculated by dividing this field value by 100, so the field value must be between 0 and 100, inclusive. When specified, the value must be less than imageGCHighThresholdPercent. Default: 80
volumeStatsAggPeriod meta/v1.Duration	volumeStatsAggPeriod is the frequency for calculating and caching volume disk usage for all pods. Default: "1m"
kubeletCgroups string	kubeletCgroups is the absolute name of cgroups to isolate the kubelet in Default: ""
systemCgroups string	systemCgroups is absolute name of cgroups in which to place all non-kernel processes that are not already in a container. Empty for no container. Rolling back the flag requires a reboot. The cgroupRoot must be specified if this field is not empty. Default: ""
cgroupRoot string	cgroupRoot is the root cgroup to use for pods. This is handled by the container runtime on a best effort basis.

Field	Description
cgroupsPerQOS bool	cgroupsPerQOS enable QoS based CGroup hierarchy: top level CGroups for QoS classes and all Burstable and BestEffort Pods are brought up under their specific top level QoS CGroup. Default: true
cgroupDriver string	cgroupDriver is the driver kubelet uses to manipulate CGroups on the host (cgroupfs or systemd). Default: "cgroupfs"
cpuManagerPolicy string	cpuManagerPolicy is the name of the policy to use. Requires the CPUManager feature gate to be enabled. Default: "None"
cpuManagerPolicyOptions map[string]string	cpuManagerPolicyOptions is a set of key=value which allows to set extra options to fine tune the behaviour of the cpu manager policies. Requires both the "CPUManager" and "CPUManagerPolicyOptions" feature gates to be enabled. Default: nil
cpuManagerReconcilePeriod meta/v1.Duration	cpuManagerReconcilePeriod is the reconciliation period for the CPU Manager. Requires the CPUManager feature gate to be enabled. Default: "10s"
memoryManagerPolicy string	memoryManagerPolicy is the name of the policy to use by memory manager. Requires the MemoryManager feature gate to be enabled. Default: "none"
topologyManagerPolicy string	<p>topologyManagerPolicy is the name of the topology manager policy to use. Valid values include:</p> <ul style="list-style-type: none"> • restricted: kubelet only allows pods with optimal NUMA node alignment for requested resources; • best-effort: kubelet will favor pods with NUMA alignment of CPU and device resources; • none: kubelet has no knowledge of NUMA alignment of a pod's CPU and device resources.

Field	Description
	<ul style="list-style-type: none"> • single-numa-node: kubelet only allows pods with a single NUMA alignment of CPU and device resources. <p>Default: "none"</p>
topologyManagerScope string	<p>topologyManagerScope represents the scope of topology hint generation that topology manager requests and hint providers generate. Valid values include:</p> <ul style="list-style-type: none"> • container: topology policy is applied on a per-container basis. • pod: topology policy is applied on a per-pod basis. <p>Default: "container"</p>
topologyManagerPolicyOptions map[string]string	<p>TopologyManagerPolicyOptions is a set of key=value which allows to set extra options to fine tune the behaviour of the topology manager policies. Requires both the "TopologyManager" and "TopologyManagerPolicyOptions" feature gates to be enabled. Default: nil</p>
qosReserved map[string]string	<p>qosReserved is a set of resource name to percentage pairs that specify the minimum percentage of a resource reserved for exclusive use by the guaranteed QoS tier. Currently supported resources: "memory" Requires the QOSReserved feature gate to be enabled. Default: nil</p>
runtimeRequestTimeout meta/v1.Duration	<p>runtimeRequestTimeout is the timeout for all runtime requests except long running requests - pull, logs, exec and attach. Default: "2m"</p>
hairpinMode string	<p>hairpinMode specifies how the Kubelet should configure the container bridge for hairpin packets. Setting this flag allows endpoints in a Service to loadbalance back to themselves if they should try to access their own Service. Values:</p> <ul style="list-style-type: none"> • "promiscuous-bridge": make the container bridge promiscuous. • "hairpin-veth": set the hairpin flag on container veth interfaces. • "none": do nothing.

Field	Description
	Generally, one must set --hairpin-mode=hairpin-veth to achieve hairpin NAT, because promiscuous-bridge assumes the existence of a container bridge named cbr0. Default: "promiscuous-bridge"
maxPods int32	maxPods is the maximum number of Pods that can run on this Kubelet. The value must be a non-negative integer. Default: 110
podCIDR string	podCIDR is the CIDR to use for pod IP addresses, only used in standalone mode. In cluster mode, this is obtained from the control plane. Default: ""
podPidsLimit int64	podPidsLimit is the maximum number of PIDs in any pod. Default: -1
resolvConf string	resolvConf is the resolver configuration file used as the basis for the container DNS resolution configuration. If set to the empty string, will override the default and effectively disable DNS lookups. Default: "/etc/resolv.conf"
runOnce bool	runOnce causes the Kubelet to check the API server once for pods, run those in addition to the pods specified by static pod files, and exit. Default: false
cpuCFSQuota bool	cpuCFSQuota enables CPU CFS quota enforcement for containers that specify CPU limits. Default: true
cpuCFSQuotaPeriod meta/v1.Duration	cpuCFSQuotaPeriod is the CPU CFS quota period value, cpu.cfs_period_us. The value must be between 1 ms and 1 second, inclusive. Requires the CustomCPUCFSQuotaPeriod feature gate to be enabled. Default: "100ms"

Field	Description
nodeStatusMaxImages int32	nodeStatusMaxImages caps the number of images reported in Node.status.images. The value must be greater than -2. Note: If -1 is specified, no cap will be applied. If 0 is specified, no image is returned. Default: 50
maxOpenFiles int64	maxOpenFiles is Number of files that can be opened by Kubelet process. The value must be a non-negative number. Default: 1000000
contentType string	contentType is contentType of requests sent to apiserver. Default: "application/vnd.kubernetes.protobuf"
kubeAPIQPS int32	kubeAPIQPS is the QPS to use while talking with kubernetes apiserver. Default: 50
kubeAPIBurst int32	kubeAPIBurst is the burst to allow while talking with kubernetes API server. This field cannot be a negative number. Default: 100
serializeImagePulls bool	serializeImagePulls when enabled, tells the Kubelet to pull images one at a time. We recommend <i>not</i> changing the default value on nodes that run docker daemon with version < 1.9 or an Aufs storage backend. Issue #10959 has more details. Default: true
maxParallelImagePulls int32	MaxParallelImagePulls sets the maximum number of image pulls in parallel. This field cannot be set if SerializeImagePulls is true. Setting it to nil means no limit. Default: nil
evictionHard map[string]string	evictionHard is a map of signal names to quantities that defines hard eviction thresholds. For example: {"memory.available": "300Mi"}. To explicitly disable, pass a 0% or 100% threshold on an arbitrary resource. Default: memory.available: "100Mi" nodefs.available: "10%" nodefs.inodesFree: "5%" imagefs.available: "15%"
evictionSoft map[string]string	

Field	Description
	evictionSoft is a map of signal names to quantities that defines soft eviction thresholds. For example: {"memory.available": "300Mi"}. Default: nil
evictionSoftGracePeriod map[string]string	evictionSoftGracePeriod is a map of signal names to quantities that defines grace periods for each soft eviction signal. For example: {"memory.available": "30s"}. Default: nil
evictionPressureTransitionPeriod meta/v1.Duration	evictionPressureTransitionPeriod is the duration for which the kubelet has to wait before transitioning out of an eviction pressure condition. Default: "5m"
evictionMaxPodGracePeriod int32	evictionMaxPodGracePeriod is the maximum allowed grace period (in seconds) to use when terminating pods in response to a soft eviction threshold being met. This value effectively caps the Pod's terminationGracePeriodSeconds value during soft evictions. Note: Due to issue #64530, the behavior has a bug where this value currently just overrides the grace period during soft eviction, which can increase the grace period from what is set on the Pod. This bug will be fixed in a future release. Default: 0
evictionMinimumReclaim map[string]string	evictionMinimumReclaim is a map of signal names to quantities that defines minimum reclaims, which describe the minimum amount of a given resource the kubelet will reclaim when performing a pod eviction while that resource is under pressure. For example: {"imagefs.available": "2Gi"}. Default: nil
podsPerCore int32	podsPerCore is the maximum number of pods per core. Cannot exceed maxPods. The value must be a non-negative integer. If 0, there is no limit on the number of Pods. Default: 0
enableControllerAttachDetach bool	enableControllerAttachDetach enables the Attach/Detach controller to manage attachment/detachment of volumes scheduled to this node, and disables kubelet from executing any attach/detach operations. Note: attaching/detaching CSI volumes

Field	Description
	is not supported by the kubelet, so this option needs to be true for that use case. Default: true
protectKernelDefaults bool	protectKernelDefaults, if true, causes the Kubelet to error if kernel flags are not as it expects. Otherwise the Kubelet will attempt to modify kernel flags to match its expectation. Default: false
makeIPTablesUtilChains bool	makeIPTablesUtilChains, if true, causes the Kubelet to create the KUBE-IPTABLES-HINT chain in iptables as a hint to other components about the configuration of iptables on the system. Default: true
iptablesMasqueradeBit int32	iptablesMasqueradeBit formerly controlled the creation of the KUBE-MARK-MASQ chain. Deprecated: no longer has any effect. Default: 14
iptablesDropBit int32	iptablesDropBit formerly controlled the creation of the KUBE-MARK-DROP chain. Deprecated: no longer has any effect. Default: 15
featureGates map[string]bool	featureGates is a map of feature names to bools that enable or disable experimental features. This field modifies piecemeal the built-in default values from "k8s.io/kubernetes/pkg/features/kube_features.go". Default: nil
failSwapOn bool	failSwapOn tells the Kubelet to fail to start if swap is enabled on the node. Default: true
memorySwap MemorySwapConfiguration	memorySwap configures swap memory available to container workloads.
containerLogMaxSize string	containerLogMaxSize is a quantity defining the maximum size of the container log file before it is rotated. For example: "5Mi" or "256Ki". Default: "10Mi"
containerLogMaxFiles int32	containerLogMaxFiles specifies the maximum number of container log files that can be present for a container. Default: 5

Field	Description
<u>configMapAndSecretChangeDetectionStrategy</u> <u>ResourceChangeDetectionStrategy</u>	<p>configMapAndSecretChangeDetectionStrategy is a mode in which ConfigMap and Secret managers are running. Valid values include:</p> <ul style="list-style-type: none"> • Get: kubelet fetches necessary objects directly from the API server; • Cache: kubelet uses TTL cache for object fetched from the API server; • Watch: kubelet uses watches to observe changes to objects that are in its interest. <p>Default: "Watch"</p>
systemReserved map[string]string	systemReserved is a set of ResourceName=ResourceQuantity (e.g. cpu=200m, memory=150G) pairs that describe resources reserved for non-kubernetes components. Currently only cpu and memory are supported. See http://kubernetes.io/docs/user-guide/compute-resources for more detail. Default: nil
kubeReserved map[string]string	kubeReserved is a set of ResourceName=ResourceQuantity (e.g. cpu=200m, memory=150G) pairs that describe resources reserved for kubernetes system components. Currently cpu, memory and local storage for root file system are supported. See https://kubernetes.io/docs/concepts/configuration/manage-resources-containers/ for more details. Default: nil
reservedSystemCPUs [Required] string	The reservedSystemCPUs option specifies the CPU list reserved for the host level system threads and kubernetes related threads. This provide a "static" CPU list rather than the "dynamic" list by systemReserved and kubeReserved. This option does not support systemReservedCgroup or kubeReservedCgroup.
showHiddenMetricsForVersion string	showHiddenMetricsForVersion is the previous version for which you want to show hidden metrics. Only the previous minor version is meaningful, other values will not be allowed. The format is <major>.<minor>, e.g.: 1.16. The purpose of this format is make sure you have the opportunity to notice if the next release hides

Field	Description
	additional metrics, rather than being surprised when they are permanently removed in the release after that. Default: ""
systemReservedCgroup string	systemReservedCgroup helps the kubelet identify absolute name of top level CGroup used to enforce systemReserved compute resource reservation for OS system daemons. Refer to Node Allocatable doc for more information. Default: ""
kubeReservedCgroup string	kubeReservedCgroup helps the kubelet identify absolute name of top level CGroup used to enforce KubeReserved compute resource reservation for Kubernetes node system daemons. Refer to Node Allocatable doc for more information. Default: ""
enforceNodeAllocatable []string	This flag specifies the various Node Allocatable enforcements that Kubelet needs to perform. This flag accepts a list of options. Acceptable options are none, pods, system-reserved and kube-reserved. If none is specified, no other options may be specified. When system-reserved is in the list, systemReservedCgroup must be specified. When kube-reserved is in the list, kubeReservedCgroup must be specified. This field is supported only when cgroupsPerQOS is set to true. Refer to Node Allocatable for more information. Default: ["pods"]
allowedUnsafeSysctls []string	A comma separated whitelist of unsafe sysctls or sysctl patterns (ending in *). Unsafe sysctl groups are kernel.shm*, kernel.msg*, kernel.sem, fs.mqueue.*, and net.*. For example: "kernel.msg*,net.ipv4.route.min_pmtu" Default: []
volumePluginDir string	volumePluginDir is the full path of the directory in which to search for additional third party volume plugins. Default: "/usr/libexec/kubernetes/kubelet-plugins/volume/exec/"
providerID string	providerID, if set, sets the unique ID of the instance that an external provider (i.e. cloudprovider) can use to identify a specific node. Default: ""
kernelMemcgNotification bool	

Field	Description
	kernelMemcgNotification, if set, instructs the kubelet to integrate with the kernel memcg notification for determining if memory eviction thresholds are exceeded rather than polling. Default: false
logging [Required] LoggingConfiguration	logging specifies the options of logging. Refer to Logs Options for more information. Default: Format: text
enableSystemLogHandler bool	enableSystemLogHandler enables system logs via web interface host:port/logs/ Default: true
enableSystemLogQuery bool	enableSystemLogQuery enables the node log query feature on the /logs endpoint. EnableSystemLogHandler has to be enabled in addition for this feature to work. Default: false
shutdownGracePeriod meta/v1.Duration	shutdownGracePeriod specifies the total duration that the node should delay the shutdown and total grace period for pod termination during a node shutdown. Default: "0s"
shutdownGracePeriodCriticalPods meta/v1.Duration	shutdownGracePeriodCriticalPods specifies the duration used to terminate critical pods during a node shutdown. This should be less than shutdownGracePeriod. For example, if shutdownGracePeriod=30s, and shutdownGracePeriodCriticalPods=10s, during a node shutdown the first 20 seconds would be reserved for gracefully terminating normal pods, and the last 10 seconds would be reserved for terminating critical pods. Default: "0s"
shutdownGracePeriodByPodPriority []ShutdownGracePeriodByPodPriority	shutdownGracePeriodByPodPriority specifies the shutdown grace period for Pods based on their associated priority class value. When a shutdown request is received, the Kubelet will initiate shutdown on all pods running on the node with a grace period that depends on the priority of the pod, and then wait for all pods to exit. Each entry in the array represents the graceful shutdown time a pod with a priority class value that lies in the range of that value and the next higher entry in the list when the node is shutting down. For

Field	Description
	<p>example, to allow critical pods 10s to shutdown, priority\geq10000 pods 20s to shutdown, and all remaining pods 30s to shutdown.</p> <p>shutdownGracePeriodByPodPriority:</p> <ul style="list-style-type: none"> • priority: 2000000000 shutdownGracePeriodSeconds: 10 • priority: 10000 shutdownGracePeriodSeconds: 20 • priority: 0 shutdownGracePeriodSeconds: 30 <p>The time the Kubelet will wait before exiting will at most be the maximum of all shutdownGracePeriodSeconds for each priority class range represented on the node. When all pods have exited or reached their grace periods, the Kubelet will release the shutdown inhibit lock. Requires the GracefulNodeShutdown feature gate to be enabled. This configuration must be empty if either ShutdownGracePeriod or ShutdownGracePeriodCriticalPods is set. Default: nil</p>
reservedMemory []MemoryReservation	<p>reservedMemory specifies a comma-separated list of memory reservations for NUMA nodes. The parameter makes sense only in the context of the memory manager feature. The memory manager will not allocate reserved memory for container workloads. For example, if you have a NUMA0 with 10Gi of memory and the reservedMemory was specified to reserve 1Gi of memory at NUMA0, the memory manager will assume that only 9Gi is available for allocation. You can specify a different amount of NUMA node and memory types. You can omit this parameter at all, but you should be aware that the amount of reserved memory from all NUMA nodes should be equal to the amount of memory specified by the node allocatable. If at least one node allocatable parameter has a non-zero value, you will need to specify at least one NUMA node. Also, avoid specifying:</p> <ol style="list-style-type: none"> 1. Duplicates, the same NUMA node, and memory type, but with a different value. 2. zero limits for any memory type. 3. NUMAs nodes IDs that do not exist under the machine. 4. memory types except for memory and hugepages-

Field	Description
	Default: nil
enableProfilingHandler bool	enableProfilingHandler enables profiling via web interface host:port/debug/pprof/ Default: true
enableDebugFlagsHandler bool	enableDebugFlagsHandler enables flags endpoint via web interface host:port/debug flags/v Default: true
seccompDefault bool	SeccompDefault enables the use of RuntimeDefault as the default seccomp profile for all workloads. Default: false
memoryThrottlingFactor float64	MemoryThrottlingFactor specifies the factor multiplied by the memory limit or node allocatable memory when setting the cgroupv2 memory.high value to enforce MemoryQoS. Decreasing this factor will set lower high limit for container cgroups and put heavier reclaim pressure while increasing will put less reclaim pressure. See https://kep.k8s.io/2570 for more details. Default: 0.9
registerWithTaints []core/v1.Taint	registerWithTaints are an array of taints to add to a node object when the kubelet registers itself. This only takes effect when registerNode is true and upon the initial registration of the node. Default: nil
registerNode bool	registerNode enables automatic registration with the apiserver. Default: true
tracing TracingConfiguration	Tracing specifies the versioned configuration for OpenTelemetry tracing clients. See https://kep.k8s.io/2832 for more details. Default: nil
localStorageCapacityIsolation bool	LocalStorageCapacityIsolation enables local ephemeral storage isolation feature. The default setting is true. This feature allows users to set request/limit for container's ephemeral storage and manage it in a similar way as cpu and memory. It also allows setting sizeLimit for emptyDir volume, which will trigger pod eviction if disk usage from

Field	Description
	the volume exceeds the limit. This feature depends on the capability of detecting correct root file system disk usage. For certain systems, such as kind rootless, if this capability cannot be supported, the feature LocalStorageCapacityIsolation should be disabled. Once disabled, user should not set request/limit for container's ephemeral storage, or sizeLimit for emptyDir. Default: true
containerRuntimeEndpoint [Required] string	ContainerRuntimeEndpoint is the endpoint of container runtime. Unix Domain Sockets are supported on Linux, while npipe and tcp endpoints are supported on Windows. Examples:'unix:///path/to/runtime.sock', 'npipe://./pipe/runtime'
imageServiceEndpoint string	ImageServiceEndpoint is the endpoint of container image service. Unix Domain Socket are supported on Linux, while npipe and tcp endpoints are supported on Windows. Examples:'unix:///path/to/runtime.sock', 'npipe://./pipe/runtime'. If not specified, the value in containerRuntimeEndpoint is used.

SerializedNodeConfigSource

SerializedNodeConfigSource allows us to serialize v1.NodeConfigSource. This type is used internally by the Kubelet for tracking checkpointed dynamic configs. It exists in the kubeletconfig API group because it is classified as a versioned input to the Kubelet.

Field	Description
apiVersion string	kubelet.config.k8s.io/v1beta1
kind string	SerializedNodeConfigSource
source core/v1.NodeConfigSource	source is the source that we are serializing.

CredentialProvider

Appears in:

- [CredentialProviderConfig](#)

CredentialProvider represents an exec plugin to be invoked by the kubelet. The plugin is only invoked when an image being pulled matches the images handled by the plugin (see matchImages).

Field	Description
name [Required] string	<p>name is the required name of the credential provider. It must match the name of the provider executable as seen by the kubelet. The executable must be in the kubelet's bin directory (set by the --image-credential-provider-bin-dir flag).</p>
matchImages [Required] []string	<p>matchImages is a required list of strings used to match against images in order to determine if this provider should be invoked. If one of the strings matches the requested image from the kubelet, the plugin will be invoked and given a chance to provide credentials. Images are expected to contain the registry domain and URL path.</p> <p>Each entry in matchImages is a pattern which can optionally contain a port and a path. Globs can be used in the domain, but not in the port or the path. Globs are supported as subdomains like '<code>*.k8s.io</code>' or '<code>k8s.*.io</code>', and top-level-domains such as '<code>k8s.*</code>'. Matching partial subdomains like '<code>app*.k8s.io</code>' is also supported. Each glob can only match a single subdomain segment, so '<code>*.io</code>' does not match '<code>*.k8s.io</code>'.</p> <p>A match exists between an image and a matchImage when all of the below are true:</p> <ul style="list-style-type: none"> • Both contain the same number of domain parts and each part matches. • The URL path of an imageMatch must be a prefix of the target image URL path. • If the imageMatch contains a port, then the port must match in the image as well. <p>Example values of matchImages:</p> <ul style="list-style-type: none"> • <code>123456789.dkr.ecr.us-east-1.amazonaws.com</code> • <code>*.azurecr.io</code> • <code>gcr.io</code> • <code>**.registry.io</code> • <code>registry.io:8080/path</code>
defaultCacheDuration [Required] meta/v1.Duration	defaultCacheDuration is the default duration the plugin will cache credentials in-memory if a cache duration is not provided in the plugin response. This field is required.

Field	Description
apiVersion [Required] string	Required input version of the exec CredentialProviderRequest. The returned CredentialProviderResponse MUST use the same encoding version as the input. Current supported values are: <ul style="list-style-type: none"> • credentialprovider.kubelet.k8s.io/v1beta1
args []string	Arguments to pass to the command when executing it.
env []ExecEnvVar	Env defines additional environment variables to expose to the process. These are unioned with the host's environment, as well as variables client-go uses to pass argument to the plugin.

ExecEnvVar

Appears in:

- [CredentialProvider](#)

ExecEnvVar is used for setting environment variables when executing an exec-based credential plugin.

Field	Description
name [Required] string	No description provided.
value [Required] string	No description provided.

KubeletAnonymousAuthentication

Appears in:

- [KubeletAuthentication](#)

Field	Description
enabled bool	enabled allows anonymous requests to the kubelet server. Requests that are not rejected by another authentication method are treated as anonymous requests. Anonymous requests have a username of system:anonymous, and a group name of system:unauthenticated.

KubeletAuthentication

Appears in:

- [KubeletConfiguration](#)

Field	Description
x509 KubeletX509Authentication	x509 contains settings related to x509 client certificate authentication.
webhook KubeletWebhookAuthentication	webhook contains settings related to webhook bearer token authentication.
anonymous KubeletAnonymousAuthentication	anonymous contains settings related to anonymous authentication.

KubeletAuthorization

Appears in:

- [KubeletConfiguration](#)

Field	Description
mode KubeletAuthorizationMode	mode is the authorization mode to apply to requests to the kubelet server. Valid values are AlwaysAllow and Webhook. Webhook mode uses the SubjectAccessReview API to determine authorization.
webhook KubeletWebhookAuthorization	webhook contains settings related to Webhook authorization.

KubeletAuthorizationMode

(Alias of string)

Appears in:

- [KubeletAuthorization](#)

KubeletWebhookAuthentication

Appears in:

- [KubeletAuthentication](#)

Field	Description
enabled bool	enabled allows bearer token authentication backed by the tokenreviews.authentication.k8s.io API.

Field	Description
cacheTTL meta/v1.Duration	cacheTTL enables caching of authentication results

KubeletWebhookAuthorization

Appears in:

- [KubeletAuthorization](#)

Field	Description
cacheAuthorizedTTL meta/v1.Duration	cacheAuthorizedTTL is the duration to cache 'authorized' responses from the webhook authorizer.
cacheUnauthorizedTTL meta/v1.Duration	cacheUnauthorizedTTL is the duration to cache 'unauthorized' responses from the webhook authorizer.

KubeletX509Authentication

Appears in:

- [KubeletAuthentication](#)

Field	Description
clientCAFile string	clientCAFile is the path to a PEM-encoded certificate bundle. If set, any request presenting a client certificate signed by one of the authorities in the bundle is authenticated with a username corresponding to the CommonName, and groups corresponding to the Organization in the client certificate.

MemoryReservation

Appears in:

- [KubeletConfiguration](#)

MemoryReservation specifies the memory reservation of different types for each NUMA node

Field	Description
numaNode [Required] int32	No description provided.
limits [Required] core/v1.ResourceList	No description provided.

MemorySwapConfiguration

Appears in:

- [KubeletConfiguration](#)

Field	Description
swapBehavior string	swapBehavior configures swap memory available to container workloads. May be one of "", "LimitedSwap": workload combined memory and swap usage cannot exceed pod memory limit "UnlimitedSwap": workloads can use unlimited swap, up to the allocatable limit.

ResourceChangeDetectionStrategy

(Alias of string)

Appears in:

- [KubeletConfiguration](#)

ResourceChangeDetectionStrategy denotes a mode in which internal managers (secret, configmap) are discovering object changes.

ShutdownGracePeriodByPodPriority

Appears in:

- [KubeletConfiguration](#)

ShutdownGracePeriodByPodPriority specifies the shutdown grace period for Pods based on their associated priority class value

Field	Description
priority [Required] int32	priority is the priority value associated with the shutdown grace period
shutdownGracePeriodSeconds [Required] int64	shutdownGracePeriodSeconds is the shutdown grace period in seconds

Kubelet CredentialProvider (v1)

Resource Types

- [CredentialProviderRequest](#)
- [CredentialProviderResponse](#)

CredentialProviderRequest

CredentialProviderRequest includes the image that the kubelet requires authentication for. Kubelet will pass this request object to the plugin via stdin. In general, plugins should prefer responding with the same apiVersion they were sent.

Field	Description
apiVersion string	credentialprovider.kubelet.k8s.io/v1
kind string	CredentialProviderRequest
image [Required] string	image is the container image that is being pulled as part of the credential provider plugin request. Plugins may optionally parse the image to extract any information required to fetch credentials.

CredentialProviderResponse

CredentialProviderResponse holds credentials that the kubelet should use for the specified image provided in the original request. Kubelet will read the response from the plugin via stdout. This response should be set to the same apiVersion as CredentialProviderRequest.

Field	Description
apiVersion string	credentialprovider.kubelet.k8s.io/v1
kind string	CredentialProviderResponse
cacheKeyType [Required] PluginCacheKeyType	cacheKeyType indicates the type of caching key to use based on the image provided in the request. There are three valid values for the cache key type: Image, Registry, and Global. If an invalid value is specified, the response will NOT be used by the kubelet.
cacheDuration meta/v1.Duration	cacheDuration indicates the duration the provided credentials should be cached for. The kubelet will use this field to set the in-memory cache duration for credentials in the AuthConfig. If null, the kubelet will use defaultCacheDuration provided in CredentialProviderConfig. If set to 0, the kubelet will not cache the provided AuthConfig.
auth map[string]k8s.io/kubelet/pkg/apis/credentialprovider/v1.AuthConfig	auth is a map containing authentication information passed into the kubelet. Each key is a match image string (more on this below). The corresponding authConfig value should be valid for all images that match against this key. A plugin should set this field to null if no valid credentials can be returned for the requested image. Each key in the map is a pattern which can optionally contain a port and a path. Globs can be used in the domain, but not in the port or

Field	Description
	<p>the path. Globs are supported as subdomains like '*.k8s.io' or 'k8s.*.io', and top-level-domains such as 'k8s.*'. Matching partial subdomains like 'app*.k8s.io' is also supported. Each glob can only match a single subdomain segment, so *.io does not match *.k8s.io.</p> <p>The kubelet will match images against the key when all of the below are true:</p> <ul style="list-style-type: none"> • Both contain the same number of domain parts and each part matches. • The URL path of an imageMatch must be a prefix of the target image URL path. • If the imageMatch contains a port, then the port must match in the image as well. <p>When multiple keys are returned, the kubelet will traverse all keys in reverse order so that:</p> <ul style="list-style-type: none"> • longer keys come before shorter keys with the same prefix • non-wildcard keys come before wildcard keys with the same prefix. <p>For any given match, the kubelet will attempt an image pull with the provided credentials, stopping after the first successfully authenticated pull.</p> <p>Example keys:</p> <ul style="list-style-type: none"> • 123456789.dkr.ecr.us-east-1.amazonaws.com • *.azurecr.io • gcr.io • *.*.registry.io • registry.io:8080/path

AuthConfig

Appears in:

- [CredentialProviderResponse](#)

AuthConfig contains authentication information for a container registry. Only username/password based authentication is supported today, but more authentication mechanisms may be added in the future.

Field	Description
username [Required] string	username is the username used for authenticating to the container registry An empty username is valid.

Field	Description
password [Required] string	password is the password used for authenticating to the container registry An empty password is valid.

PluginCacheKeyType

(Alias of string)

Appears in:

- [CredentialProviderResponse](#)

Kubelet CredentialProvider (v1alpha1)

Resource Types

- [CredentialProviderRequest](#)
- [CredentialProviderResponse](#)

CredentialProviderRequest

CredentialProviderRequest includes the image that the kubelet requires authentication for. Kubelet will pass this request object to the plugin via stdin. In general, plugins should prefer responding with the same apiVersion they were sent.

Field	Description
apiVersion string	credentialprovider.kubelet.k8s.io/v1alpha1
kind string	CredentialProviderRequest
image [Required] string	image is the container image that is being pulled as part of the credential provider plugin request. Plugins may optionally parse the image to extract any information required to fetch credentials.

CredentialProviderResponse

CredentialProviderResponse holds credentials that the kubelet should use for the specified image provided in the original request. Kubelet will read the response from the plugin via stdout. This response should be set to the same apiVersion as CredentialProviderRequest.

Field	Description
apiVersion string	credentialprovider.kubelet.k8s.io/v1alpha1
	CredentialProviderResponse

Field	Description
kind string	
cacheKeyType [Required] PluginCacheKeyType	cacheKeyType indicates the type of caching key to use based on the image provided in the request. There are three valid values for the cache key type: Image, Registry, and Global. If an invalid value is specified, the response will NOT be used by the kubelet.
cacheDuration meta/v1.Duration	cacheDuration indicates the duration the provided credentials should be cached for. The kubelet will use this field to set the in-memory cache duration for credentials in the AuthConfig. If null, the kubelet will use defaultCacheDuration provided in CredentialProviderConfig. If set to 0, the kubelet will not cache the provided AuthConfig.
auth map[string]k8s.io/kubelet/pkg/apis/credentialprovider/v1alpha1.AuthConfig	<p>auth is a map containing authentication information passed into the kubelet. Each key is a match image string (more on this below). The corresponding authConfig value should be valid for all images that match against this key. A plugin should set this field to null if no valid credentials can be returned for the requested image.</p> <p>Each key in the map is a pattern which can optionally contain a port and a path. Globs can be used in the domain, but not in the port or the path. Globs are supported as subdomains like '*.k8s.io' or 'k8s.*.io', and top-level-domains such as 'k8s.*'. Matching partial subdomains like 'app*.k8s.io' is also supported. Each glob can only match a single subdomain segment, so *.io does not match *.k8s.io.</p> <p>The kubelet will match images against the key when all of the below are true:</p> <ul style="list-style-type: none"> • Both contain the same number of domain parts and each part matches. • The URL path of an imageMatch must be a prefix of the target image URL path. • If the imageMatch contains a port, then the port must match in the image as well. <p>When multiple keys are returned, the kubelet will traverse all keys in reverse order so that:</p> <ul style="list-style-type: none"> • longer keys come before shorter keys with the same prefix • non-wildcard keys come before wildcard keys with the same prefix. <p>For any given match, the kubelet will attempt an image pull with the provided credentials, stopping after the first successfully authenticated pull.</p>

Field	Description
	<p>Example keys:</p> <ul style="list-style-type: none"> • 123456789.dkr.ecr.us-east-1.amazonaws.com • *.azurecr.io • gcr.io • **.registry.io • registry.io:8080/path

AuthConfig

Appears in:

- [CredentialProviderResponse](#)

AuthConfig contains authentication information for a container registry. Only username/password based authentication is supported today, but more authentication mechanisms may be added in the future.

Field	Description
username [Required] string	username is the username used for authenticating to the container registry An empty username is valid.
password [Required] string	password is the password used for authenticating to the container registry An empty password is valid.

PluginCacheKeyType

(Alias of string)

Appears in:

- [CredentialProviderResponse](#)

Kubelet CredentialProvider (v1beta1)

Resource Types

- [CredentialProviderRequest](#)
- [CredentialProviderResponse](#)

CredentialProviderRequest

CredentialProviderRequest includes the image that the kubelet requires authentication for. Kubelet will pass this request object to the plugin via stdin. In general, plugins should prefer responding with the same apiVersion they were sent.

Field	Description
apiVersion string	credentialprovider.kubelet.k8s.io/v1beta1
kind string	CredentialProviderRequest
image [Required] string	image is the container image that is being pulled as part of the credential provider plugin request. Plugins may optionally parse the image to extract any information required to fetch credentials.

CredentialProviderResponse

CredentialProviderResponse holds credentials that the kubelet should use for the specified image provided in the original request. Kubelet will read the response from the plugin via stdout. This response should be set to the same apiVersion as CredentialProviderRequest.

Field	Description
apiVersion string	credentialprovider.kubelet.k8s.io/v1beta1
kind string	CredentialProviderResponse
cacheKeyType [Required] PluginCacheKeyType	cacheKeyType indicates the type of caching key to use based on the image provided in the request. There are three valid values for the cache key type: Image, Registry, and Global. If an invalid value is specified, the response will NOT be used by the kubelet.
cacheDuration meta/v1.Duration	cacheDuration indicates the duration the provided credentials should be cached for. The kubelet will use this field to set the in-memory cache duration for credentials in the AuthConfig. If null, the kubelet will use defaultCacheDuration provided in CredentialProviderConfig. If set to 0, the kubelet will not cache the provided AuthConfig.
auth map[string]k8s.io/kubelet/pkg/apis/credentialprovider/v1beta1.AuthConfig	auth is a map containing authentication information passed into the kubelet. Each key is a match image string (more on this below). The corresponding authConfig value should be valid for all images that match against this key. A plugin should set this field to null if no valid credentials can be returned for the requested image. Each key in the map is a pattern which can optionally contain a port and a path. Globs can be used in the domain, but not in the

Field	Description
	<p>port or the path. Globs are supported as subdomains like '*.k8s.io' or 'k8s.*.io', and top-level-domains such as 'k8s.*'. Matching partial subdomains like 'app*.k8s.io' is also supported. Each glob can only match a single subdomain segment, so *.io does not match *.k8s.io.</p> <p>The kubelet will match images against the key when all of the below are true:</p> <ul style="list-style-type: none"> • Both contain the same number of domain parts and each part matches. • The URL path of an imageMatch must be a prefix of the target image URL path. • If the imageMatch contains a port, then the port must match in the image as well. <p>When multiple keys are returned, the kubelet will traverse all keys in reverse order so that:</p> <ul style="list-style-type: none"> • longer keys come before shorter keys with the same prefix • non-wildcard keys come before wildcard keys with the same prefix. <p>For any given match, the kubelet will attempt an image pull with the provided credentials, stopping after the first successfully authenticated pull.</p> <p>Example keys:</p> <ul style="list-style-type: none"> • 123456789.dkr.ecr.us-east-1.amazonaws.com • *.azurecr.io • gcr.io • **.registry.io • registry.io:8080/path

AuthConfig

Appears in:

- [CredentialProviderResponse](#)

AuthConfig contains authentication information for a container registry. Only username/password based authentication is supported today, but more authentication mechanisms may be added in the future.

Field	Description
username [Required] string	username is the username used for authenticating to the container registry An empty username is valid.

Field	Description
password [Required] string	password is the password used for authenticating to the container registry An empty password is valid.

PluginCacheKeyType

(Alias of string)

Appears in:

- [CredentialProviderResponse](#)

WebhookAdmission Configuration (v1)

Package v1 is the v1 version of the API.

Resource Types

- [WebhookAdmission](#)

WebhookAdmission

WebhookAdmission provides configuration for the webhook admission controller.

Field	Description
apiVersion string	apiserver.config.k8s.io/v1
kind string	WebhookAdmission
kubeConfigFile [Required] string	KubeConfigFile is the path to the kubeconfig file.

External APIs

[Kubernetes Custom Metrics \(v1beta2\)](#)

[Kubernetes External Metrics \(v1beta1\)](#)

[Kubernetes Metrics \(v1beta1\)](#)

Kubernetes Custom Metrics (v1beta2)

Package v1beta2 is the v1beta2 version of the custom_metrics API.

Resource Types

- [MetricListOptions](#)
- [MetricValue](#)
- [MetricValueList](#)

MetricListOptions

MetricListOptions is used to select metrics by their label selectors

Field	Description
apiVersion string	custom.metrics.k8s.io/v1beta2
kind string	MetricListOptions
labelSelector string	A selector to restrict the list of returned objects by their labels. Defaults to everything.
metricLabelSelector string	A selector to restrict the list of returned metrics by their labels

MetricValue

Appears in:

- [MetricValueList](#)

MetricValue is the metric value for some object

Field	Description
apiVersion string	custom.metrics.k8s.io/v1beta2
kind string	MetricValue
describedObject [Required] core/v1.ObjectReference	a reference to the described object
metric [Required] MetricIdentifier	No description provided.
timestamp [Required] meta/v1.Time	indicates the time at which the metrics were produced

Field	Description
windowSeconds [Required] int64	indicates the window ([Timestamp-Window, Timestamp]) from which these metrics were calculated, when returning rate metrics calculated from cumulative metrics (or zero for non-calculated instantaneous metrics).
value [Required] k8s.io/apimachinery/pkg/api/resource.Quantity	the value of the metric for this

MetricValueList

MetricValueList is a list of values for a given metric for some set of objects

Field	Description
apiVersion string	custom.metrics.k8s.io/v1beta2
kind string	MetricValueList
metadata [Required] meta/v1.ListMeta	No description provided.
items [Required] []MetricValue	the value of the metric across the described objects

MetricIdentifier

Appears in:

- [MetricValue](#)

MetricIdentifier identifies a metric by name and, optionally, selector

Field	Description
name [Required] string	name is the name of the given metric
selector meta/v1.LabelSelector	selector represents the label selector that could be used to select this metric, and will generally just be the selector passed in to the query used to fetch this metric. When left blank, only the metric's Name will be used to gather metrics.

Kubernetes External Metrics (v1beta1)

Package v1beta1 is the v1beta1 version of the external metrics API.

Resource Types

- [ExternalMetricValue](#)
- [ExternalMetricValueList](#)

ExternalMetricValue

Appears in:

- [ExternalMetricValueList](#)

ExternalMetricValue is a metric value for external metric A single metric value is identified by metric name and a set of string labels. For one metric there can be multiple values with different sets of labels.

Field	Description
apiVersion string	external.metrics.k8s.io/v1beta1
kind string	ExternalMetricValue
metricName [Required] string	the name of the metric
metricLabels [Required] map[string]string	a set of labels that identify a single time series for the metric
timestamp [Required] meta/v1.Time	indicates the time at which the metrics were produced
window [Required] int64	indicates the window ([Timestamp-Window, Timestamp]) from which these metrics were calculated, when returning rate metrics calculated from cumulative metrics (or zero for non-calculated instantaneous metrics).
value [Required] k8s.io/apimachinery/pkg/api/resource.Quantity	the value of the metric

ExternalMetricValueList

ExternalMetricValueList is a list of values for a given metric for some set labels

Field	Description
apiVersion string	external.metrics.k8s.io/v1beta1
kind string	ExternalMetricValueList
	No description provided.

Field	Description
metadata [Required] meta/v1.ListMeta	
items [Required] []ExternalMetricValue	value of the metric matching a given set of labels

Kubernetes Metrics (v1beta1)

Package v1beta1 is the v1beta1 version of the metrics API.

Resource Types

- [NodeMetrics](#)
- [NodeMetricsList](#)
- [PodMetrics](#)
- [PodMetricsList](#)

NodeMetrics

Appears in:

- [NodeMetricsList](#)

NodeMetrics sets resource usage metrics of a node.

Field	Description
apiVersion string	metrics.k8s.io/v1beta1
kind string	NodeMetrics
metadata meta/v1.ObjectMeta	Standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata Refer to the Kubernetes API documentation for the fields of the metadata field.
timestamp [Required] meta/v1.Time	The following fields define time interval from which metrics were collected from the interval [Timestamp-Window, Timestamp].
window [Required] meta/v1.Duration	No description provided.
usage [Required] core/v1.ResourceList	The memory usage is the memory working set.

NodeMetricsList

NodeMetricsList is a list of NodeMetrics.

Field	Description
apiVersion string	metrics.k8s.io/v1beta1
kind string	NodeMetricsList
metadata [Required] meta/v1.ListMeta	Standard list metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
items [Required] []NodeMetrics	List of node metrics.

PodMetrics

Appears in:

- [PodMetricsList](#)

PodMetrics sets resource usage metrics of a pod.

Field	Description
apiVersion string	metrics.k8s.io/v1beta1
kind string	PodMetrics
metadata meta/v1.ObjectMeta	Standard object's metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#metadata Refer to the Kubernetes API documentation for the fields of the metadata field.
timestamp [Required] meta/v1.Time	The following fields define time interval from which metrics were collected from the interval [Timestamp-Window, Timestamp].
window [Required] meta/v1.Duration	No description provided.
containers [Required] []ContainerMetrics	Metrics for all containers are collected within the same time window.

PodMetricsList

PodMetricsList is a list of PodMetrics.

Field	Description
apiVersion string	metrics.k8s.io/v1beta1
kind string	PodMetricsList
metadata [Required] meta/v1.ListMeta	Standard list metadata. More info: https://git.k8s.io/community/contributors/devel/sig-architecture/api-conventions.md#types-kinds
items [Required] []PodMetrics	List of pod metrics.

ContainerMetrics

Appears in:

- [PodMetrics](#)

ContainerMetrics sets resource usage metrics of a container.

Field	Description
name [Required] string	Container name corresponding to the one from pod.spec.containers.
usage [Required] core/v1.ResourceList	The memory usage is the memory working set.

Scheduling

[Scheduler Configuration](#)

[Scheduling Policies](#)

Scheduler Configuration

FEATURE STATE: Kubernetes v1.25 [stable]

You can customize the behavior of the kube-scheduler by writing a configuration file and passing its path as a command line argument.

A scheduling Profile allows you to configure the different stages of scheduling in the [kube-scheduler](#). Each stage is exposed in an extension point. Plugins provide scheduling behaviors by implementing one or more of these extension points.

You can specify scheduling profiles by running `kube-scheduler --config <filename>`, using the KubeSchedulerConfiguration [v1](#) struct.

A minimal configuration looks as follows:

```
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
clientConnection:
  kubeconfig: /etc/srv/kubernetes/kube-scheduler/kubeconfig
```

Note: KubeSchedulerConfiguration [v1beta3](#) is deprecated in v1.26 and will be removed in v1.29. Please migrate KubeSchedulerConfiguration to [v1](#).

Profiles

A scheduling Profile allows you to configure the different stages of scheduling in the [kube-scheduler](#). Each stage is exposed in an [extension point](#). [Plugins](#) provide scheduling behaviors by implementing one or more of these extension points.

You can configure a single instance of kube-scheduler to run [multiple profiles](#).

Extension points

Scheduling happens in a series of stages that are exposed through the following extension points:

1. queueSort: These plugins provide an ordering function that is used to sort pending Pods in the scheduling queue. Exactly one queue sort plugin may be enabled at a time.
2. preFilter: These plugins are used to pre-process or check information about a Pod or the cluster before filtering. They can mark a pod as unschedulable.
3. filter: These plugins are the equivalent of Predicates in a scheduling Policy and are used to filter out nodes that can not run the Pod. Filters are called in the configured order. A pod is marked as unschedulable if no nodes pass all the filters.
4. postFilter: These plugins are called in their configured order when no feasible nodes were found for the pod. If any postFilter plugin marks the Pod *schedulable*, the remaining plugins are not called.
5. preScore: This is an informational extension point that can be used for doing pre-scoring work.
6. score: These plugins provide a score to each node that has passed the filtering phase. The scheduler will then select the node with the highest weighted scores sum.
7. reserve: This is an informational extension point that notifies plugins when resources have been reserved for a given Pod. Plugins also implement an Unreserve call that gets called in the case of failure during or after Reserve.
8. permit: These plugins can prevent or delay the binding of a Pod.
9. preBind: These plugins perform any work required before a Pod is bound.
10. bind: The plugins bind a Pod to a Node. bind plugins are called in order and once one has done the binding, the remaining plugins are skipped. At least one bind plugin is required.
11. postBind: This is an informational extension point that is called after a Pod has been bound.
12. multiPoint: This is a config-only field that allows plugins to be enabled or disabled for all of their applicable extension points simultaneously.

For each extension point, you could disable specific [default plugins](#) or enable your own. For example:

```

apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
  - plugins:
      score:
        disabled:
          - name: PodTopologySpread
        enabled:
          - name: MyCustomPluginA
            weight: 2
          - name: MyCustomPluginB
            weight: 1

```

You can use `*` as name in the disabled array to disable all default plugins for that extension point. This can also be used to rearrange plugins order, if desired.

Scheduling plugins

The following plugins, enabled by default, implement one or more of these extension points:

- ImageLocality: Favors nodes that already have the container images that the Pod runs. Extension points: score.
- TaintToleration: Implements [taints and tolerations](#). Implements extension points: filter, preScore, score.
- NodeName: Checks if a Pod spec node name matches the current node. Extension points: filter.
- NodePorts: Checks if a node has free ports for the requested Pod ports. Extension points: preFilter, filter.
- NodeAffinity: Implements [node selectors](#) and [node affinity](#). Extension points: filter, score.
- PodTopologySpread: Implements [Pod topology spread](#). Extension points: preFilter, filter, preScore, score.
- NodeUnschedulable: Filters out nodes that have `.spec.unschedulable` set to true. Extension points: filter.
- NodeResourcesFit: Checks if the node has all the resources that the Pod is requesting. The score can use one of three strategies: LeastAllocated (default), MostAllocated and RequestedToCapacityRatio. Extension points: preFilter, filter, score.
- NodeResourcesBalancedAllocation: Favors nodes that would obtain a more balanced resource usage if the Pod is scheduled there. Extension points: score.
- VolumeBinding: Checks if the node has or if it can bind the requested [volumes](#). Extension points: preFilter, filter, reserve, preBind, score.

Note: score extension point is enabled when `VolumeCapacityPriority` feature is enabled. It prioritizes the smallest PVs that can fit the requested volume size.

- VolumeRestrictions: Checks that volumes mounted in the node satisfy restrictions that are specific to the volume provider. Extension points: filter.
- VolumeZone: Checks that volumes requested satisfy any zone requirements they might have. Extension points: filter.
- NodeVolumeLimits: Checks that CSI volume limits can be satisfied for the node. Extension points: filter.
- EBSLimits: Checks that AWS EBS volume limits can be satisfied for the node. Extension points: filter.
- GCEPDLimits: Checks that GCP-PD volume limits can be satisfied for the node. Extension points: filter.

- AzureDiskLimits: Checks that Azure disk volume limits can be satisfied for the node. Extension points: filter.
- InterPodAffinity: Implements [inter-Pod affinity and anti-affinity](#). Extension points: preFilter, filter, preScore, score.
- PrioritySort: Provides the default priority based sorting. Extension points: queueSort.
- DefaultBinder: Provides the default binding mechanism. Extension points: bind.
- DefaultPreemption: Provides the default preemption mechanism. Extension points: postFilter.

You can also enable the following plugins, through the component config APIs, that are not enabled by default:

- CinderLimits: Checks that [OpenStack Cinder](#) volume limits can be satisfied for the node. Extension points: filter.

Multiple profiles

You can configure kube-scheduler to run more than one profile. Each profile has an associated scheduler name and can have a different set of plugins configured in its [extension points](#).

With the following sample configuration, the scheduler will run with two profiles: one with the default plugins and one with all scoring plugins disabled.

```
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
  - schedulerName: default-scheduler
  - schedulerName: no-scoring-scheduler
    plugins:
      preScore:
        disabled:
          - name: '*'
      score:
        disabled:
          - name: '*'
```

Pods that want to be scheduled according to a specific profile can include the corresponding scheduler name in its .spec.schedulerName.

By default, one profile with the scheduler name default-scheduler is created. This profile includes the default plugins described above. When declaring more than one profile, a unique scheduler name for each of them is required.

If a Pod doesn't specify a scheduler name, kube-apiserver will set it to default-scheduler. Therefore, a profile with this scheduler name should exist to get those pods scheduled.

Note: Pod's scheduling events have .spec.schedulerName as the ReportingController. Events for leader election use the scheduler name of the first profile in the list.

Note: All profiles must use the same plugin in the queueSort extension point and have the same configuration parameters (if applicable). This is because the scheduler only has one pending pods queue.

Plugins that apply to multiple extension points

Starting from kubescheduler.config.k8s.io/v1beta3, there is an additional field in the profile config, multiPoint, which allows for easily enabling or disabling a plugin across several extension points. The intent of multiPoint config is to simplify the configuration needed for users and administrators when using custom profiles.

Consider a plugin, MyPlugin, which implements the preScore, score, preFilter, and filter extension points. To enable MyPlugin for all its available extension points, the profile config looks like:

```
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: multipoint-scheduler
  plugins:
    multiPoint:
      enabled:
        - name: MyPlugin
```

This would equate to manually enabling MyPlugin for all of its extension points, like so:

```
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: non-multipoint-scheduler
  plugins:
    preScore:
      enabled:
        - name: MyPlugin
    score:
      enabled:
        - name: MyPlugin
    preFilter:
      enabled:
        - name: MyPlugin
    filter:
      enabled:
        - name: MyPlugin
```

One benefit of using multiPoint here is that if MyPlugin implements another extension point in the future, the multiPoint config will automatically enable it for the new extension.

Specific extension points can be excluded from MultiPoint expansion using the disabled field for that extension point. This works with disabling default plugins, non-default plugins, or with the wildcard (*) to disable all plugins. An example of this, disabling Score and PreScore, would be:

```
apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: non-multipoint-scheduler
  plugins:
    multiPoint:
      disabled:
        - name: Score
        - name: PreScore
```

```

enabled:
- name: 'MyPlugin'
preScore:
disabled:
- name: '*'
score:
disabled:
- name: '*'

```

Starting from kubescheduler.config.k8s.io/v1beta3, all [default plugins](#) are enabled internally through MultiPoint. However, individual extension points are still available to allow flexible reconfiguration of the default values (such as ordering and Score weights). For example, consider two Score plugins DefaultScore1 and DefaultScore2, each with a weight of 1. They can be reordered with different weights like so:

```

apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: multipoint-scheduler
  plugins:
    score:
      enabled:
      - name: 'DefaultScore2'
        weight: 5

```

In this example, it's unnecessary to specify the plugins in MultiPoint explicitly because they are default plugins. And the only plugin specified in Score is DefaultScore2. This is because plugins set through specific extension points will always take precedence over MultiPoint plugins. So, this snippet essentially re-orders the two plugins without needing to specify both of them.

The general hierarchy for precedence when configuring MultiPoint plugins is as follows:

1. Specific extension points run first, and their settings override whatever is set elsewhere
2. Plugins manually configured through MultiPoint and their settings
3. Default plugins and their default settings

To demonstrate the above hierarchy, the following example is based on these plugins:

Plugin	Extension Points
DefaultQueueSort	QueueSort
CustomQueueSort	QueueSort
DefaultPlugin1	Score, Filter
DefaultPlugin2	Score
CustomPlugin1	Score, Filter
CustomPlugin2	Score, Filter

A valid sample configuration for these plugins would be:

```

apiVersion: kubescheduler.config.k8s.io/v1
kind: KubeSchedulerConfiguration
profiles:
- schedulerName: multipoint-scheduler
  plugins:

```

```

multiPoint:
  enabled:
    - name: 'CustomQueueSort'
    - name: 'CustomPlugin1'
      weight: 3
    - name: 'CustomPlugin2'
  disabled:
    - name: 'DefaultQueueSort'
filter:
  disabled:
    - name: 'DefaultPlugin1'
score:
  enabled:
    - name: 'DefaultPlugin2'

```

Note that there is no error for re-declaring a MultiPoint plugin in a specific extension point. The re-declaration is ignored (and logged), as specific extension points take precedence.

Besides keeping most of the config in one spot, this sample does a few things:

- Enables the custom queueSort plugin and disables the default one
- Enables CustomPlugin1 and CustomPlugin2, which will run first for all of their extension points
- Disables DefaultPlugin1, but only for filter
- Reorders DefaultPlugin2 to run first in score (even before the custom plugins)

In versions of the config before v1beta3, without multiPoint, the above snippet would equate to this:

```

apiVersion: kubescheduler.config.k8s.io/v1beta2
kind: KubeSchedulerConfiguration
profiles:
  - schedulerName: multipoint-scheduler
    plugins:
      # Disable the default QueueSort plugin
      queueSort:
        enabled:
          - name: 'CustomQueueSort'
        disabled:
          - name: 'DefaultQueueSort'

      # Enable custom Filter plugins
      filter:
        enabled:
          - name: 'CustomPlugin1'
          - name: 'CustomPlugin2'
          - name: 'DefaultPlugin2'
        disabled:
          - name: 'DefaultPlugin1'

      # Enable and reorder custom score plugins
      score:

```

```
enabled:
- name: 'DefaultPlugin2'
  weight: 1
- name: 'DefaultPlugin1'
  weight: 3
```

While this is a complicated example, it demonstrates the flexibility of MultiPoint config as well as its seamless integration with the existing methods for configuring extension points.

Scheduler configuration migrations

- [v1beta1 → v1beta2](#)
- [v1beta2 → v1beta3](#)
- [v1beta3 → v1](#)

- With the v1beta2 configuration version, you can use a new score extension for the NodeResourcesFit plugin. The new extension combines the functionalities of the NodeResourcesLeastAllocated, NodeResourcesMostAllocated and RequestedToCapacityRatio plugins. For example, if you previously used the NodeResourcesMostAllocated plugin, you would instead use NodeResourcesFit (enabled by default) and add a pluginConfig with a scoreStrategy that is similar to:

```
apiVersion: kubescheduler.config.k8s.io/v1beta2
kind: KubeSchedulerConfiguration
profiles:
- pluginConfig:
  - args:
    scoringStrategy:
      resources:
      - name: cpu
        weight: 1
      type: MostAllocated
    name: NodeResourcesFit
```

- The scheduler plugin NodeLabel is deprecated; instead, use the [NodeAffinity](#) plugin (enabled by default) to achieve similar behavior.
- The scheduler plugin ServiceAffinity is deprecated; instead, use the [InterPodAffinity](#) plugin (enabled by default) to achieve similar behavior.
- The scheduler plugin NodePreferAvoidPods is deprecated; instead, use [node taints](#) to achieve similar behavior.
- A plugin enabled in a v1beta2 configuration file takes precedence over the default configuration for that plugin.
- Invalid host or port configured for scheduler healthz and metrics bind address will cause validation failure.
- Three plugins' weight are increased by default:
 - InterPodAffinity from 1 to 2
 - NodeAffinity from 1 to 2
 - TaintToleration from 1 to 3

- The scheduler plugin SelectorSpread is removed, instead, use the PodTopologySpread plugin (enabled by default) to achieve similar behavior.

What's next

- Read the [kube-scheduler reference](#)
- Learn about [scheduling](#)
- Read the [kube-scheduler configuration \(v1beta2\)](#) reference
- Read the [kube-scheduler configuration \(v1beta3\)](#) reference
- Read the [kube-scheduler configuration \(v1\)](#) reference

Scheduling Policies

In Kubernetes versions before v1.23, a scheduling policy can be used to specify the *predicates* and *priorities* process. For example, you can set a scheduling policy by running `kube-scheduler --policy-config-file <filename>` or `kube-scheduler --policy-configmap <ConfigMap>`.

This scheduling policy is not supported since Kubernetes v1.23. Associated flags `policy-config-file`, `policy-configmap`, `policy-configmap-namespace` and `use-legacy-policy-config` are also not supported. Instead, use the [Scheduler Configuration](#) to achieve similar behavior.

What's next

- Learn about [scheduling](#)
- Learn about [kube-scheduler Configuration](#)
- Read the [kube-scheduler configuration reference \(v1\)](#)

Other Tools

Kubernetes contains several tools to help you work with the Kubernetes system.

cricctl

[cricctl](#) is a command-line interface for inspecting and debugging [CRI](#)-compatible container runtimes.

Dashboard

[Dashboard](#), the web-based user interface of Kubernetes, allows you to deploy containerized applications to a Kubernetes cluster, troubleshoot them, and manage the cluster and its resources itself.

Helm

This item links to a third party project or product that is not part of Kubernetes itself. [More information](#)

[Helm](#) is a tool for managing packages of pre-configured Kubernetes resources. These packages are known as *Helm charts*.

Use Helm to:

- Find and use popular software packaged as Kubernetes charts
- Share your own applications as Kubernetes charts
- Create reproducible builds of your Kubernetes applications
- Intelligently manage your Kubernetes manifest files
- Manage releases of Helm packages

Kompose

[Kompose](#) is a tool to help Docker Compose users move to Kubernetes.

Use Kompose to:

- Translate a Docker Compose file into Kubernetes objects
- Go from local Docker development to managing your application via Kubernetes
- Convert v1 or v2 Docker Compose yaml files or [Distributed Application Bundles](#)

Kui

[Kui](#) is a GUI tool that takes your normal kubectl command line requests and responds with graphics.

Kui takes the normal kubectl command line requests and responds with graphics. Instead of ASCII tables, Kui provides a GUI rendering with tables that you can sort.

Kui lets you:

- Directly click on long, auto-generated resource names instead of copying and pasting
- Type in kubectl commands and see them execute, even sometimes faster than kubectl itself
- Query a [Job](#) and see its execution rendered as a waterfall diagram
- Click through resources in your cluster using a tabbed UI

Minikube

[minikube](#) is a tool that runs a single-node Kubernetes cluster locally on your workstation for development and testing purposes.

Mapping from dockercli to crictl

Note: This section links to third party projects that provide functionality required by Kubernetes. The Kubernetes project authors aren't responsible for these projects, which are listed alphabetically. To add a project to this list, read the [content guide](#) before submitting a change. [More information.](#)

cricl is a command-line interface for [CRI](#)-compatible container runtimes. You can use it to inspect and debug container runtimes and applications on a Kubernetes node. crictl and its source are hosted in the [cri-tools](#) repository.

This page provides a reference for mapping common commands for the docker command-line tool into the equivalent commands for crictl.

Mapping from docker CLI to crictl

The exact versions for the mapping table are for docker CLI v1.40 and crictl v1.19.0. This list is not exhaustive. For example, it doesn't include experimental docker CLI commands.

Note: The output format of crictl is similar to docker CLI, despite some missing columns for some CLI. Make sure to check output for the specific command if your command output is being parsed programmatically.

Retrieve debugging information

mapping from docker cli to crictl - retrieve debugging information

docker cli	crictl	Description	Unsupported Features
attach	attach	Attach to a running container	--detach-keys, --sig-proxy
exec	exec	Run a command in a running container	--privileged, --user, --detach-keys
images	images	List images	
info	info	Display system-wide information	
inspect	inspect, inspecti	Return low-level information on a container, image or task	
logs	logs	Fetch the logs of a container	--details
ps	ps	List containers	
stats	stats	Display a live stream of container(s) resource usage statistics	Column: NET/BLOCK I/O, PIDs
version	version	Show the runtime (Docker, ContainerID, or others) version information	

Perform Changes

mapping from docker cli to crictl - perform changes

docker cli	crictl	Description	Unsupported Features
create	create	Create a new container	
kill	stop (timeout = 0)	Kill one or more running container	--signal
pull	pull	Pull an image or a repository from a registry	--all-tags, --disable-content-trust
rm	rm	Remove one or more containers	

docker cli	cricctl	Description	Unsupported Features
rmi	rmi	Remove one or more images	
run	run	Run a command in a new container	
start	start	Start one or more stopped containers	--detach-keys
stop	stop	Stop one or more running containers	
update	update	Update configuration of one or more containers	--restart, --blkio-weight and some other resource limit not supported by CRI.

Supported only in cricctl

mapping from docker cli to cricctl - supported only in cricctl

cricctl	Description
imagefsinfo	Return image filesystem info
inspectp	Display the status of one or more pods
port-forward	Forward local port to a pod
pods	List pods
runp	Run a new pod
rmp	Remove one or more pods
stopp	Stop one or more running pods