OpenShift Container Platform 4.18

API overview

Overview content for the OpenShift Container Platform API

Last Updated: 2025-06-04

OpenShift Container Platform 4.18 API overview

Overview content for the OpenShift Container Platform API

Legal Notice

Abstract

This document provides overview information for OpenShift Container Platform API.

Table of Contents

CHAPTER 1. UNDERSTANDING API TIERS

IMPORTANT

This guidance does not cover layered OpenShift Container Platform offerings.

API tiers for bare-metal configurations also apply to virtualized configurations except for any feature that directly interacts with hardware. Those features directly related to hardware have no application operating environment (AOE) compatibility level beyond that which is provided by the hardware vendor. For example, applications that rely on Graphics Processing Units (GPU) features are subject to the AOE compatibility provided by the GPU vendor driver.

API tiers in a cloud environment for cloud specific integration points have no API or AOE compatibility level beyond that which is provided by the hosting cloud vendor. For example, APIs that exercise dynamic management of compute, ingress, or storage are dependent upon the underlying API capabilities exposed by the cloud platform. Where a cloud vendor modifies a prerequisite API, Red Hat will provide commercially reasonable efforts to maintain support for the API with the capability presently offered by the cloud infrastructure vendor.

Red Hat requests that application developers validate that any behavior they depend on is explicitly defined in the formal API documentation to prevent introducing dependencies on unspecified implementation-specific behavior or dependencies on bugs in a particular implementation of an API. For example, new releases of an ingress router may not be compatible with older releases if an application uses an undocumented API or relies on undefined behavior.

1.1. API TIERS

All commercially supported APIs, components, and features are associated under one of the following support levels:

API tier 1

APIs and application operating environments (AOEs) are stable within a major release. They may be deprecated within a major release, but they will not be removed until a subsequent major release.

API tier 2

APIs and AOEs are stable within a major release for a minimum of 9 months or 3 minor releases from the announcement of deprecation, whichever is longer.

API tier 3

This level applies to languages, tools, applications, and optional Operators included with OpenShift Container Platform through Operator Hub. Each component will specify a lifetime during which the API and AOE will be supported. Newer versions of language runtime specific components will attempt to be as API and AOE compatible from minor version to minor version as possible. Minor version to minor version compatibility is not guaranteed, however.

Components and developer tools that receive continuous updates through the Operator Hub, referred to as Operators and operands, should be considered API tier 3. Developers should use caution and understand how these components may change with each minor release. Users are encouraged to consult the compatibility guidelines documented by the component.

API tier 4

No compatibility is provided. API and AOE can change at any point. These capabilities should not be used by applications needing long-term support.

It is common practice for Operators to use custom resource definitions (CRDs) internally to accomplish a task. These objects are not meant for use by actors external to the Operator and are intended to be hidden. If any CRD is not meant for use by actors external to the Operator, the **operators.operatorframework.io/internal-objects** annotation in the Operators **ClusterServiceVersion** (CSV) should be specified to signal that the corresponding resource is internal use only and the CRD may be explicitly labeled as tier 4.

1.2. MAPPING API TIERS TO API GROUPS

For each API tier defined by Red Hat, we provide a mapping table for specific API groups where the upstream communities are committed to maintain forward compatibility. Any API group that does not specify an explicit compatibility level and is not specifically discussed below is assigned API tier 3 by default except for **v1alpha1** APIs which are assigned tier 4 by default.

1.2.1. Support for Kubernetes API groups

API groups that end with the suffix ***.k8s.io** or have the form **version.<name>** with no suffix are governed by the Kubernetes deprecation policy and follow a general mapping between API version exposed and corresponding support tier unless otherwise specified.

API version example	API tier
v1	Tier 1
v1beta1	Tier 2
v1alpha1	Tier 4

1.2.2. Support for OpenShift API groups

API groups that end with the suffix ***.openshift.io** are governed by the OpenShift Container Platform deprecation policy and follow a general mapping between API version exposed and corresponding compatibility level unless otherwise specified.

API version example	API tier
apps.openshift.io/v1	Tier 1
authorization.openshift.io/v1	Tier 1, some tier 1 deprecated
build.openshift.io/v1	Tier 1, some tier 1 deprecated
config.openshift.io/v1	Tier 1
image.openshift.io/v1	Tier 1
network.openshift.io/v1	Tier 1
network.operator.openshift.io/v1	Tier 1

API version example	API tier
oauth.openshift.io/v1	Tier 1
imagecontentsourcepolicy.operator.openshift.io/v1alpha1	Tier 1
project.openshift.io/v1	Tier 1
quota.openshift.io/v1	Tier 1
route.openshift.io/v1	Tier 1
quota.openshift.io/v1	Tier 1
security.openshift.io/v1	Tier 1 except for **RangeAllocation** (tier 4) and ***Reviews** (tier 2)
template.openshift.io/v1	Tier 1
console.openshift.io/v1	Tier 2

1.2.3. Support for Monitoring API groups

API groups that end with the suffix **monitoring.coreos.com** have the following mapping:

API version example	API tier
v1	Tier 1
v1alpha1	Tier 1
v1beta1	Tier 1

1.2.4. Support for Operator Lifecycle Manager API groups

Operator Lifecycle Manager (OLM) provides APIs that include API groups with the suffix **operators.coreos.com**. These APIs have the following mapping:

API version example	API tier
v2	Tier 1
v1	Tier 1
v1alpha1	Tier 1

1.3. API DEPRECATION POLICY

OpenShift Container Platform is composed of many components sourced from many upstream communities. It is anticipated that the set of components, the associated API interfaces, and correlated features will evolve over time and might require formal deprecation in order to remove the capability.

1.3.1. Deprecating parts of the API

OpenShift Container Platform is a distributed system where multiple components interact with a shared state managed by the cluster control plane through a set of structured APIs. Per Kubernetes conventions, each API presented by OpenShift Container Platform is associated with a group identifier and each API group is independently versioned. Each API group is managed in a distinct upstream community including Kubernetes, Metal3, Multus, Operator Framework, Open Cluster Management, OpenShift itself, and more.

While each upstream community might define their own unique deprecation policy for a given API group and version, Red Hat normalizes the community specific policy to one of the compatibility levels defined prior based on our integration in and awareness of each upstream community to simplify end-user consumption and support.

The deprecation policy and schedule for APIs vary by compatibility level.

The deprecation policy covers all elements of the API including:

- REST resources, also known as API objects
- Fields of REST resources
- Annotations on REST resources, excluding version-specific qualifiers
- Enumerated or constant values

Other than the most recent API version in each group, older API versions must be supported after their announced deprecation for a duration of no less than:

API tier	Duration
Tier 1	Stable within a major release. They may be deprecated within a major release, but they will not be removed until a subsequent major release.
Tier 2	9 months or 3 releases from the announcement of deprecation, whichever is longer.
Tier 3	See the component-specific schedule.
Tier 4	None. No compatibility is guaranteed.

The following rules apply to all tier 1 APIs:

- API elements can only be removed by incrementing the version of the group.
- API objects must be able to round-trip between API versions without information loss, with the

exception of whole REST resources that do not exist in some versions. In cases where equivalent fields do not exist between versions, data will be preserved in the form of annotations during conversion.

- API versions in a given group can not deprecate until a new API version at least as stable is released, except in cases where the entire API object is being removed.

1.3.2. Deprecating CLI elements

Client-facing CLI commands are not versioned in the same way as the API, but are user-facing component systems. The two major ways a user interacts with a CLI are through a command or flag, which is referred to in this context as CLI elements.

All CLI elements default to API tier 1 unless otherwise noted or the CLI depends on a lower tier API.

	Element	API tier
Generally available (GA)	Flags and commands	Tier 1
Technology Preview	Flags and commands	Tier 3
Developer Preview	Flags and commands	Tier 4

1.3.3. Deprecating an entire component

The duration and schedule for deprecating an entire component maps directly to the duration associated with the highest API tier of an API exposed by that component. For example, a component that surfaced APIs with tier 1 and 2 could not be removed until the tier 1 deprecation schedule was met.

API tier	Duration
Tier 1	Stable within a major release. They may be deprecated within a major release, but they will not be removed until a subsequent major release.
Tier 2	9 months or 3 releases from the announcement of deprecation, whichever is longer.
Tier 3	See the component-specific schedule.
Tier 4	None. No compatibility is guaranteed.

CHAPTER 2. UNDERSTANDING API COMPATIBILITY GUIDELINES

IMPORTANT

This guidance does not cover layered OpenShift Container Platform offerings.

2.1. API COMPATIBILITY GUIDELINES

Red Hat recommends that application developers adopt the following principles in order to improve compatibility with OpenShift Container Platform:

- Use APIs and components with support tiers that match the application's need.
- Build applications using the published client libraries where possible.
- Applications are only guaranteed to run correctly if they execute in an environment that is as new as the environment it was built to execute against. An application that was built for OpenShift Container Platform 4.14 is not guaranteed to function properly on OpenShift Container Platform 4.13.
- Do not design applications that rely on configuration files provided by system packages or other components. These files can change between versions unless the upstream community is explicitly committed to preserving them. Where appropriate, depend on any Red Hat provided interface abstraction over those configuration files in order to maintain forward compatibility. Direct file system modification of configuration files is discouraged, and users are strongly encouraged to integrate with an Operator provided API where available to avoid dual-writer conflicts.
- Do not depend on API fields prefixed with **unsupported<FieldName>** or annotations that are not explicitly mentioned in product documentation.
- Do not depend on components with shorter compatibility guarantees than your application.
- Do not perform direct storage operations on the etcd server. All etcd access must be performed via the api-server or through documented backup and restore procedures.

Red Hat recommends that application developers follow the compatibility guidelines defined by Red Hat Enterprise Linux (RHEL). OpenShift Container Platform strongly recommends the following guidelines when building an application or hosting an application on the platform:

- Do not depend on a specific Linux kernel or OpenShift Container Platform version.
- Avoid reading from **proc**, **sys**, and **debug** file systems, or any other pseudo file system.
- Avoid using **ioctls** to directly interact with hardware.
- Avoid direct interaction with **cgroups** in order to not conflict with OpenShift Container Platform host-agents that provide the container execution environment.

NOTE

During the lifecycle of a release, Red Hat makes commercially reasonable efforts to maintain API and application operating environment (AOE) compatibility across all minor releases and z-stream releases. If necessary, Red Hat might make exceptions to this compatibility goal for critical impact security or other significant issues.

2.2. API COMPATIBILITY EXCEPTIONS

The following are exceptions to compatibility in OpenShift Container Platform:

RHEL CoreOS file system modifications not made with a supported Operator
No assurances are made at this time that a modification made to the host operating file system is preserved across minor releases except for where that modification is made through the public interface exposed via a supported Operator, such as the Machine Config Operator or Node Tuning Operator.

Modifications to cluster infrastructure in cloud or virtualized environments
No assurances are made at this time that a modification to the cloud hosting environment that supports the cluster is preserved except for where that modification is made through a public interface exposed in the product or is documented as a supported configuration. Cluster infrastructure providers are responsible for preserving their cloud or virtualized infrastructure except for where they delegate that authority to the product through an API.

Functional defaults between an upgraded cluster and a new installation
No assurances are made at this time that a new installation of a product minor release will have the same functional defaults as a version of the product that was installed with a prior minor release and upgraded to the equivalent version. For example, future versions of the product may provision cloud infrastructure with different defaults than prior minor versions. In addition, different default security choices may be made in future versions of the product than those made in past versions of the product. Past versions of the product will forward upgrade, but preserve legacy choices where appropriate specifically to maintain backwards compatibility.

Usage of API fields that have the prefix "unsupported” or undocumented annotations
Select APIs in the product expose fields with the prefix **unsupported<FieldName>**. No assurances are made at this time that usage of this field is supported across releases or within a release. Product support can request a customer to specify a value in this field when debugging specific problems, but its usage is not supported outside of that interaction. Usage of annotations on objects that are not explicitly documented are not assured support across minor releases.

API availability per product installation topology
The OpenShift distribution will continue to evolve its supported installation topology, and not all APIs in one install topology will necessarily be included in another. For example, certain topologies may restrict read/write access to particular APIs if they are in conflict with the product installation topology or not include a particular API at all if not pertinent to that topology. APIs that exist in a given topology will be supported in accordance with the compatibility tiers defined above.

2.3. API COMPATIBILITY COMMON TERMINOLOGY

2.3.1. Application Programming Interface (API)

An API is a public interface implemented by a software program that enables it to interact with other software. In OpenShift Container Platform, the API is served from a centralized API server and is used as the hub for all system interaction.

2.3.2. Application Operating Environment (AOE)

An AOE is the integrated environment that executes the end-user application program. The AOE is a containerized environment that provides isolation from the host operating system (OS). At a minimum, AOE allows the application to run in an isolated manner from the host OS libraries and binaries, but still share the same OS kernel as all other containers on the host. The AOE is enforced at runtime and it describes the interface between an application and its operating environment. It includes intersection points between the platform, operating system and environment, with the user application including projection of downward API, DNS, resource accounting, device access, platform workload identity, isolation among containers, isolation between containers and host OS.

The AOE does not include components that might vary by installation, such as Container Network Interface (CNI) plugin selection or extensions to the product such as admission hooks. Components that integrate with the cluster at a level below the container environment might be subjected to additional variation between versions.

2.3.3. Compatibility in a virtualized environment

Virtual environments emulate bare-metal environments such that unprivileged applications that run on bare-metal environments will run, unmodified, in corresponding virtual environments. Virtual environments present simplified abstracted views of physical resources, so some differences might exist.

2.3.4. Compatibility in a cloud environment

OpenShift Container Platform might choose to offer integration points with a hosting cloud environment via cloud provider specific integrations. The compatibility of these integration points are specific to the guarantee provided by the native cloud vendor and its intersection with the OpenShift Container Platform compatibility window. Where OpenShift Container Platform provides an integration with a cloud environment natively as part of the default installation, Red Hat develops against stable cloud API endpoints to provide commercially reasonable support with forward looking compatibility that includes stable deprecation policies. Example areas of integration between the cloud provider and OpenShift Container Platform include, but are not limited to, dynamic volume provisioning, service load balancer integration, pod workload identity, dynamic management of compute, and infrastructure provisioned as part of initial installation.

2.3.5. Major, minor, and z-stream releases

A Red Hat major release represents a significant step in the development of a product. Minor releases appear more frequently within the scope of a major release and represent deprecation boundaries that might impact future application compatibility. A z-stream release is an update to a minor release which provides a stream of continuous fixes to an associated minor release. API and AOE compatibility is never broken in a z-stream release except when this policy is explicitly overridden in order to respond to an unforeseen security impact.

For example, in the release 4.13.2:

- 4 is the major release version
- 13 is the minor release version
- 2 is the z-stream release version

2.3.6. Extended user support (EUS)

A minor release in an OpenShift Container Platform major release that has an extended support window for critical bug fixes. Users are able to migrate between EUS releases by incrementally adopting minor versions between EUS releases. It is important to note that the deprecation policy is defined across minor releases and not EUS releases. As a result, an EUS user might have to respond to a deprecation when migrating to a future EUS while sequentially upgrading through each minor release.

2.3.7. Developer Preview

An optional product capability that is not officially supported by Red Hat, but is intended to provide a mechanism to explore early phase technology. By default, Developer Preview functionality is opt-in, and subject to removal at any time. Enabling a Developer Preview feature might render a cluster unsupportable dependent upon the scope of the feature.

If you are a Red( )Hat customer or partner and have feedback about these developer preview versions, file an issue by using the OpenShift Bugs tracker. Do not use the formal Red( )Hat support service ticket process. You can read more about support handling in the following knowledge article.

2.3.8. Technology Preview

An optional product capability that provides early access to upcoming product innovations to test functionality and provide feedback during the development process. The feature is not fully supported, might not be functionally complete, and is not intended for production use. Usage of a Technology Preview function requires explicit opt-in. Learn more about the Technology Preview Features Support Scope.

CHAPTER 3. EDITING KUBELET LOG LEVEL VERBOSITY AND GATHERING LOGS

To troubleshoot some issues with nodes, establish the kubelet's log level verbosity depending on the issue to be tracked.

3.1. MODIFYING THE KUBELET AS A ONE-TIME SCENARIO

To modify the kubelet in a one-time scenario without rebooting the node due to the change of **machine-config(spec":{"paused":false}})**, allowing you to modify the kubelet without affecting the service, follow this procedure.

Procedure

1. Connect to the node in debug mode:

```
$ oc debug node/<node>
```

```
$ chroot /host
```

   Alternatively, it is possible to SSH to the node and become root.

2. After access is established, check the default log level:

```
$ systemctl cat kubelet
```

   #### Example output

```
# /etc/systemd/system/kubelet.service.d/20-logging.conf
[Service]
Environment="KUBELET_LOG_LEVEL=2"
```

3. Define the new verbosity required in a new **/etc/systemd/system/kubelet.service.d/30-logging.conf** file, which overrides **/etc/systemd/system/kubelet.service.d/20-logging.conf**. In this example, the verbosity is changed from **2** to **8**:

```
$ echo -e "[Service]\nEnvironment=\"KUBELET_LOG_LEVEL=8\"" >
/etc/systemd/system/kubelet.service.d/30-logging.conf
```

4. Reload systemd and restart the service:

```
$ systemctl daemon-reload
```

```
$ systemctl restart kubelet
```

5. Gather the logs, and then revert the log level increase:

```
$ rm -f /etc/systemd/system/kubelet.service.d/30-logging.conf
```

```
$ systemctl daemon-reload
```

```
$ systemctl restart kubelet
```

3.2. PERSISTENT KUBELET LOG LEVEL CONFIGURATION

Procedure

- Use the following **MachineConfig** object for persistent kubelet log level configuration:

```
apiVersion: machineconfiguration.openshift.io/v1
kind: MachineConfig
metadata:
  labels:
    machineconfiguration.openshift.io/role: master
  name: 99-master-kubelet-loglevel
spec:
  config:
    ignition:
      version: 3.2.0
    systemd:
      units:
        - name: kubelet.service
          enabled: true
          dropins:
            - name: 30-logging.conf
              contents: |
                [Service]
                Environment="KUBELET_LOG_LEVEL=2"
```

Generally, it is recommended to apply **0-4** as debug-level logs and **5-8** as trace-level logs.

3.3. LOG VERBOSITY DESCRIPTIONS

Log verbosity	Description
--v=0	Always visible to an Operator.
--v=1	A reasonable default log level if you do not want verbosity.
--v=2	Useful steady state information about the service and important log messages that might correlate to significant changes in the system. This is the recommended default log level.
--v=3	Extended information about changes.
--v=4	Debug level verbosity.
--v=6	Display requested resources.
--v=7	Display HTTP request headers.

Log verbosity	Description
--v=8	Display HTTP request contents.

3.4. GATHERING KUBELET LOGS

Procedure

- After the kubelet's log level verbosity is configured properly, you can gather logs by running the following commands:

```
$ oc adm node-logs --role master -u kubelet
```

```
$ oc adm node-logs --role worker -u kubelet
```

Alternatively, inside the node, run the following command:

```
$ journalctl -b -f -u kubelet.service
```

- To collect master container logs, run the following command:

```
$ sudo tail -f /var/log/containers/*
```

- To directly gather the logs of all nodes, run the following command:

```
- for n in $(oc get node --no-headers | awk '{print $1}'); do oc adm node-logs $n | gzip >
$n.log.gz; done
```

CHAPTER 4. API INDEX

API	API group
AdminNetworkPolicy	policy.networking.k8s.io/v1alpha1
AdminPolicyBasedExternalRoute	k8s.ovn.org/v1
AlertingRule	monitoring.openshift.io/v1
Alertmanager	monitoring.coreos.com/v1
AlertmanagerConfig	monitoring.coreos.com/v1beta1
AlertRelabelConfig	monitoring.openshift.io/v1
APIRequestCount	apiserver.openshift.io/v1
APIServer	config.openshift.io/v1
APIService	apiregistration.k8s.io/v1
AppliedClusterResourceQuota	quota.openshift.io/v1
Authentication	config.openshift.io/v1
Authentication	operator.openshift.io/v1
BareMetalHost	metal3.io/v1alpha1
BaselineAdminNetworkPolicy	policy.networking.k8s.io/v1alpha1
Binding	v1
BMCEventSubscription	metal3.io/v1alpha1
BrokerTemplateInstance	template.openshift.io/v1
Build	build.openshift.io/v1
Build	config.openshift.io/v1
BuildConfig	build.openshift.io/v1
BuildLog	build.openshift.io/v1
BuildRequest	build.openshift.io/v1

API	API group
CatalogSource	operators.coreos.com/v1alpha1
CertificateSigningRequest	certificates.k8s.io/v1
CloudCredential	operator.openshift.io/v1
CloudPrivateIPConfig	cloud.network.openshift.io/v1
ClusterAutoscaler	autoscaling.openshift.io/v1
ClusterCSIDriver	operator.openshift.io/v1
ClusterOperator	config.openshift.io/v1
ClusterResourceQuota	quota.openshift.io/v1
ClusterRole	authorization.openshift.io/v1
ClusterRole	rbac.authorization.k8s.io/v1
ClusterRoleBinding	authorization.openshift.io/v1
ClusterRoleBinding	rbac.authorization.k8s.io/v1
ClusterServiceVersion	operators.coreos.com/v1alpha1
ClusterVersion	config.openshift.io/v1
ComponentStatus	v1
Config	imageregistry.operator.openshift.io/v1
Config	operator.openshift.io/v1
Config	samples.operator.openshift.io/v1
ConfigMap	v1
Console	config.openshift.io/v1
Console	operator.openshift.io/v1
ConsoleCLIDownload	console.openshift.io/v1
ConsoleExternalLogLink	console.openshift.io/v1

API	API group
ConsoleLink	console.openshift.io/v1
ConsoleNotification	console.openshift.io/v1
ConsolePlugin	console.openshift.io/v1
ConsoleQuickStart	console.openshift.io/v1
ConsoleSample	console.openshift.io/v1
ConsoleYAMLSample	console.openshift.io/v1
ContainerRuntimeConfig	machineconfiguration.openshift.io/v1
ControllerConfig	machineconfiguration.openshift.io/v1
ControllerRevision	apps/v1
ControlPlaneMachineSet	machine.openshift.io/v1
CredentialsRequest	cloudcredential.openshift.io/v1
CronJob	batch/v1
CSIDriver	storage.k8s.io/v1
CSINode	storage.k8s.io/v1
CSISnapshotController	operator.openshift.io/v1
CSIStorageCapacity	storage.k8s.io/v1
CustomResourceDefinition	apiextensions.k8s.io/v1
DaemonSet	apps/v1
DataImage	metal3.io/v1alpha1
Deployment	apps/v1
DeploymentConfig	apps.openshift.io/v1
DeploymentConfigRollback	apps.openshift.io/v1
DeploymentLog	apps.openshift.io/v1

API	API group
DeploymentRequest	apps.openshift.io/v1
DNS	config.openshift.io/v1
DNS	operator.openshift.io/v1
DNSRecord	ingress.operator.openshift.io/v1
EgressFirewall	k8s.ovn.org/v1
EgressIP	k8s.ovn.org/v1
EgressQoS	k8s.ovn.org/v1
EgressRouter	network.operator.openshift.io/v1
EgressService	k8s.ovn.org/v1
Endpoints	v1
EndpointSlice	discovery.k8s.io/v1
Etcd	operator.openshift.io/v1
Event	v1
Event	events.k8s.io/v1
Eviction	policy/v1
FeatureGate	config.openshift.io/v1
FirmwareSchema	metal3.io/v1alpha1
FlowSchema	flowcontrol.apiserver.k8s.io/v1
Group	user.openshift.io/v1
HardwareData	metal3.io/v1alpha1
HelmChartRepository	helm.openshift.io/v1beta1
HorizontalPodAutoscaler	autoscaling/v2
HostFirmwareComponents	metal3.io/v1alpha1

API	API group
HostFirmwareSettings	metal3.io/v1alpha1
Identity	user.openshift.io/v1
Image	config.openshift.io/v1
Image	image.openshift.io/v1
ImageContentPolicy	config.openshift.io/v1
ImageContentSourcePolicy	operator.openshift.io/v1alpha1
ImageDigestMirrorSet	config.openshift.io/v1
ImagePruner	imageregistry.operator.openshift.io/v1
ImageSignature	image.openshift.io/v1
ImageStream	image.openshift.io/v1
ImageStreamImage	image.openshift.io/v1
ImageStreamImport	image.openshift.io/v1
ImageStreamLayers	image.openshift.io/v1
ImageStreamMapping	image.openshift.io/v1
ImageStreamTag	image.openshift.io/v1
ImageTag	image.openshift.io/v1
ImageTagMirrorSet	config.openshift.io/v1
Infrastructure	config.openshift.io/v1
Ingress	config.openshift.io/v1
Ingress	networking.k8s.io/v1
IngressClass	networking.k8s.io/v1
IngressController	operator.openshift.io/v1
InsightsOperator	operator.openshift.io/v1

API	API group
InstallPlan	operators.coreos.com/v1alpha1
IPAddress	ipam.cluster.x-k8s.io/v1beta1
IPAddressClaim	ipam.cluster.x-k8s.io/v1beta1
IPPool	whereabouts.cni.cncf.io/v1alpha1
Job	batch/v1
KubeAPIServer	operator.openshift.io/v1
KubeControllerManager	operator.openshift.io/v1
KubeletConfig	machineconfiguration.openshift.io/v1
KubeScheduler	operator.openshift.io/v1
KubeStorageVersionMigrator	operator.openshift.io/v1
Lease	coordination.k8s.io/v1
LimitRange	v1
LocalResourceAccessReview	authorization.openshift.io/v1
LocalSubjectAccessReview	authorization.k8s.io/v1
LocalSubjectAccessReview	authorization.openshift.io/v1
Machine	machine.openshift.io/v1beta1
MachineAutoscaler	autoscaling.openshift.io/v1beta1
MachineConfig	machineconfiguration.openshift.io/v1
MachineConfigPool	machineconfiguration.openshift.io/v1
MachineConfiguration	operator.openshift.io/v1
MachineHealthCheck	machine.openshift.io/v1beta1
MachineSet	machine.openshift.io/v1beta1
Metal3Remediation	infrastructure.cluster.x-k8s.io/v1beta1

API	API group
Metal3RemediationTemplate	infrastructure.cluster.x-k8s.io/v1beta1
MultiNetworkPolicy	k8s.cni.cncf.io/v1beta1
MutatingWebhookConfiguration	admissionregistration.k8s.io/v1
Namespace	v1
Network	config.openshift.io/v1
Network	operator.openshift.io/v1
NetworkAttachmentDefinition	k8s.cni.cncf.io/v1
NetworkPolicy	networking.k8s.io/v1
Node	v1
Node	config.openshift.io/v1
NodeMetrics	metrics.k8s.io/v1beta1
OAuth	config.openshift.io/v1
OAuthAccessToken	oauth.openshift.io/v1
OAuthAuthorizeToken	oauth.openshift.io/v1
OAuthClient	oauth.openshift.io/v1
OAuthClientAuthorization	oauth.openshift.io/v1
OLMConfig	operators.coreos.com/v1
OpenShiftAPIServer	operator.openshift.io/v1
OpenShiftControllerManager	operator.openshift.io/v1
Operator	operators.coreos.com/v1
OperatorCondition	operators.coreos.com/v2
OperatorGroup	operators.coreos.com/v1
OperatorHub	config.openshift.io/v1

API	API group
OperatorPKI	network.operator.openshift.io/v1
OverlappingRangeIPReservation	whereabouts.cni.cncf.io/v1alpha1
PackageManifest	packages.operators.coreos.com/v1
PerformanceProfile	performance.openshift.io/v2
PersistentVolume	v1
PersistentVolumeClaim	v1
Pod	v1
PodDisruptionBudget	policy/v1
PodMetrics	metrics.k8s.io/v1beta1
PodMonitor	monitoring.coreos.com/v1
PodNetworkConnectivityCheck	controlplane.operator.openshift.io/v1alpha1
PodSecurityPolicyReview	security.openshift.io/v1
PodSecurityPolicySelfSubjectReview	security.openshift.io/v1
PodSecurityPolicySubjectReview	security.openshift.io/v1
PodTemplate	v1
PreprovisioningImage	metal3.io/v1alpha1
PriorityClass	scheduling.k8s.io/v1
PriorityLevelConfiguration	flowcontrol.apiserver.k8s.io/v1
Probe	monitoring.coreos.com/v1
Profile	tuned.openshift.io/v1
Project	config.openshift.io/v1
Project	project.openshift.io/v1

API	API group
ProjectHelmChartRepository	helm.openshift.io/v1beta1
ProjectRequest	project.openshift.io/v1
Prometheus	monitoring.coreos.com/v1
PrometheusRule	monitoring.coreos.com/v1
Provisioning	metal3.io/v1alpha1
Proxy	config.openshift.io/v1
RangeAllocation	security.openshift.io/v1
ReplicaSet	apps/v1
ReplicationController	v1
ResourceAccessReview	authorization.openshift.io/v1
ResourceQuota	v1
Role	authorization.openshift.io/v1
Role	rbac.authorization.k8s.io/v1
RoleBinding	authorization.openshift.io/v1
RoleBinding	rbac.authorization.k8s.io/v1
RoleBindingRestriction	authorization.openshift.io/v1
Route	route.openshift.io/v1
RuntimeClass	node.k8s.io/v1
Scale	autoscaling/v1
Scheduler	config.openshift.io/v1
Secret	v1
SecretList	image.openshift.io/v1

API	API group
SecurityContextConstraints	security.openshift.io/v1
SelfSubjectAccessReview	authorization.k8s.io/v1
SelfSubjectReview	authentication.k8s.io/v1
SelfSubjectRulesReview	authorization.k8s.io/v1
SelfSubjectRulesReview	authorization.openshift.io/v1
Service	v1
ServiceAccount	v1
ServiceCA	operator.openshift.io/v1
ServiceMonitor	monitoring.coreos.com/v1
StatefulSet	apps/v1
Storage	operator.openshift.io/v1
StorageClass	storage.k8s.io/v1
StorageState	migration.k8s.io/v1alpha1
StorageVersionMigration	migration.k8s.io/v1alpha1
SubjectAccessReview	authorization.k8s.io/v1
SubjectAccessReview	authorization.openshift.io/v1
SubjectRulesReview	authorization.openshift.io/v1
Subscription	operators.coreos.com/v1alpha1
Template	template.openshift.io/v1
TemplateInstance	template.openshift.io/v1
ThanosRuler	monitoring.coreos.com/v1
TokenRequest	authentication.k8s.io/v1

API	API group
TokenReview	authentication.k8s.io/v1
Tuned	tuned.openshift.io/v1
User	user.openshift.io/v1
UserIdentityMapping	user.openshift.io/v1
UserOAuthAccessToken	oauth.openshift.io/v1
ValidatingAdmissionPolicy	admissionregistration.k8s.io/v1
ValidatingAdmissionPolicyBinding	admissionregistration.k8s.io/v1
ValidatingWebhookConfiguration	admissionregistration.k8s.io/v1
VolumeAttachment	storage.k8s.io/v1
VolumeSnapshot	snapshot.storage.k8s.io/v1
VolumeSnapshotClass	snapshot.storage.k8s.io/v1
VolumeSnapshotContent	snapshot.storage.k8s.io/v1