Getting started

This section lists the different ways to set up and run Kubernetes. When you install Kubernetes, choose an installation type based on: ease of maintenance, security, control, available resources, and expertise required to operate and manage a cluster.

You can deploy a Kubernetes cluster on a local machine, cloud, on-prem datacenter, or choose a managed Kubernetes cluster. There are also custom solutions across a wide range of cloud providers, or bare metal environments.

Learning environment

If you're learning Kubernetes, use the tools supported by the Kubernetes community, or tools in the ecosystem to set up a Kubernetes cluster on a local machine.

Production environment

When evaluating a solution for a production environment, consider which aspects of operating a Kubernetes cluster (or *abstractions*) you want to manage yourself or offload to a provider.

Kubernetes Partners includes a list of Certified Kubernetes providers.

Release notes and version skew

Learning environment

Production environment

Best practices

Release notes and version skew

---

v1.20 Release Notes

Kubernetes version and version skew support policy

v1.20 Release Notes

v1.20.0

Documentation

Downloads for v1.20.0

filename	sha512 hash
kubernetes.tar.gz	ebfe49552bbda02807034488967b3b62bf9e3e507d56245e298c4c19090
kubernetes-src.tar.gz	bcbd67ed0bb77840828c08c6118ad0c9bf2bcda16763afaafd8731fd6ce

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	3609f6483f4244676162232b3294d7a2dc40ae5bdd86a842a05aa768f
kubernetes-client-linux-386.tar.gz	e06c08016a08137d39804383fdc33a40bb2567aa77d88a5c3fd5b9d93
kubernetes-client-linux-amd64.tar.gz	081472833601aa4fa78e79239f67833aa4efcb4efe714426cd01d4ddf
kubernetes-client-linux-arm.tar.gz	037f84a2f29fe62d266cab38ac5600d058cce12cbc4851bcf062fafba
kubernetes-client-linux-arm64.tar.gz	275727e1796791ca3cbe52aaa713a2660404eab6209466fdc1cfa8559
kubernetes-client-linux-ppc64le.tar.gz	7a9965293029e9fcdb2b7387467f022d2026953b8461e6c84182abf35
kubernetes-client-linux-s390x.tar.gz	85fc449ce1980f5f030cc32e8c8e2198c1cc91a448e04b15d27debc3c
kubernetes-client-windows-386.tar.gz	4c0a27dba1077aaee943e0eb7a787239dd697e1d968e78d1933c1e60b
kubernetes-client-windows-amd64.tar.gz	29336faf7c596539b8329afbbdceeddc843162501de4afee44a406162

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	fb56486a55dbf7dbacb53b1aaa690bae18d33d244c72a1e2dc95fb0fcce451

filename	sha512 hash
kubernetes-server-linux-arm.tar.gz	735ed9993071fe35b292bf06930ee3c0f889e3c7edb983195b1c8e4d71130
kubernetes-server-linux-arm64.tar.gz	ffab155531d5a9b82487ee1abf4f6ef49626ea58b2de340656a762e46cf3e
kubernetes-server-linux-ppc64le.tar.gz	9d5730d35c4ddfb4c5483173629fe55df35d1e535d96f02459468220ac2c9
kubernetes-server-linux-s390x.tar.gz	6e4c165306940e8b99dd6e590f8542e31aed23d2c7a6808af0357fa425cec

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	3e6c90561dd1c27fa1dff6953c503251c36001f7e0f8eff3ec918c74ae2d9
kubernetes-node-linux-arm.tar.gz	26db385d9ae9a97a1051a638e7e3de22c4bbff389d5a419fe40d5893f9e4f
kubernetes-node-linux-arm64.tar.gz	5b8b63f617e248432b7eb913285a8ef8ba028255216332c05db949666c3f9
kubernetes-node-linux-ppc64le.tar.gz	60da7715996b4865e390640525d6e98593ba3cd45c6caeea763aa5355a7f9
kubernetes-node-linux-s390x.tar.gz	9407dc55412bd04633f84fcefe3a1074f3eaa772a7cb9302242b8768d6189
kubernetes-node-windows-amd64.tar.gz	9d4261af343cc330e6359582f80dbd6efb57d41f882747a94bbf47b4f9329

Changelog since v1.19.0

What's New (Major Themes)

Dockershim deprecation

Docker as an underlying runtime is being deprecated. Docker-produced images will continue to work in your cluster with all runtimes, as they always have. The Kubernetes community has written a blog post about this in detail with a dedicated FAQ page for it.

External credential provider for client-go

The client-go credential plugins can now be passed in the current cluster information via the `KUBERNETES_EXEC_INFO` environment variable. Learn more about this on client-go credential plugins documentation.

CronJob controller v2 is available through feature gate

An alternative implementation of `CronJob` controller is now available as an alpha feature in this release, which has experimental performance improvement by using informers instead of polling. While this will be the default behavior in the future, you can try them in this release through a feature gate.

PID Limits graduates to General Availability

PID Limits features are now generally available on both `SupportNodePidsLimit` (node-to-pod PID isolation) and `SupportPodPidsLimit` (ability to limit PIDs per pod), after being enabled-by-default in beta stage for a year.

API Priority and Fairness graduates to Beta

Initially introduced in 1.18, Kubernetes 1.20 now enables API Priority and Fairness (APF) by default. This allows `kube-apiserver` to categorize incoming requests by priority levels.

IPv4/IPv6 run

IPv4/IPv6 dual-stack has been reimplemented for 1.20 to support dual-stack Services, based on user and community feedback. If your cluster has dual-stack enabled, you can create Services which can use IPv4, IPv6, or both, and you can change this setting for existing Services. Details are available in updated IPv4/IPv6 dual-stack docs, which cover the nuanced array of options.

We expect this implementation to progress from alpha to beta and GA in coming releases, so we're eager to have you comment about your dual-stack experiences in #k8s-dual-stack or in enhancements #563.

go1.15.5

go1.15.5 has been integrated to Kubernets project as of this release, including other infrastructure related updates on this effort.

CSI Volume Snapshot graduates to General Availability

CSI Volume Snapshot moves to GA in the 1.20 release. This feature provides a standard way to trigger volume snapshot operations in Kubernetes and allows Kubernetes users to incorporate snapshot operations in a portable manner on any Kubernetes environment regardless of supporting underlying storage providers. Additionally, these Kubernetes snapshot primitives act as

basic building blocks that unlock the ability to develop advanced, enterprise grade, storage administration features for Kubernetes: including application or cluster level backup solutions. Note that snapshot support will require Kubernetes distributors to bundle the Snapshot controller, Snapshot CRDs, and validation webhook. In addition, a CSI driver supporting the snapshot functionality must also be deployed on the cluster.

Non-recursive Volume Ownership (FSGroup) graduates to Beta

By default, the `fsgroup` setting, if specified, recursively updates permissions for every file in a volume on every mount. This can make mount, and pod startup, very slow if the volume has many files. This setting enables a pod to specify a `PodFSGroupChangePolicy` that indicates that volume ownership and permissions will be changed only when permission and ownership of the root directory does not match with expected permissions on the volume.

CSIDriver policy for FSGroup graduates to Beta

The FSGroup's CSIDriver Policy is now beta in 1.20. This allows CSIDrivers to explicitly indicate if they want Kubernetes to manage permissions and ownership for their volumes via `fsgroup`.

Security Improvements for CSI Drivers (Alpha)

In 1.20, we introduce a new alpha feature `CSIServiceAccountToken`. This feature allows CSI drivers to impersonate the pods that they mount the volumes for. This improves the security posture in the mounting process where the volumes are ACL'ed on the pods' service account without handing out unnecessary permissions to the CSI drivers' service account. This feature is especially important for secret-handling CSI drivers, such as the secrets-store-csi-driver. Since these tokens can be rotated and short-lived, this feature also provides a knob for CSI drivers to receive `NodePublishVolume` RPC calls periodically with the new token. This knob is also useful when volumes are short-lived, e.g. certificates.

Introducing Graceful Node Shutdown (Alpha)

The `GracefulNodeShutdown` feature is now in Alpha. This allows kubelet to be aware of node system shutdowns, enabling graceful termination of pods during a system shutdown. This feature can be enabled through feature gate.

Runtime log sanitation

Logs can now be configured to use runtime protection from leaking sensitive data. Details for this experimental feature is available in documentation.

Pod resource metrics

On-demand metrics calculation is now available through `/metrics/resources`. When enabled, the endpoint will report the requested resources and the desired limits of all running pods.

Introducing `RootCAConfigMap`

`RootCAConfigMap` graduates to Beta, seperating from `BoundServiceAccountTokenVolume`. The `kube-root-ca.crt` ConfigMap is now available to every namespace, by default. It contains the Certificate Authority bundle for verify kube-apiserver connections.

`kubectl debug` graduates to Beta

`kubectl alpha debug` graduates from alpha to beta in 1.20, becoming `kubectl debug`. `kubectl debug` provides support for common debugging workflows directly from kubectl. Troubleshooting scenarios supported in this release of `kubectl` include: Troubleshoot workloads that crash on startup by creating a copy of the pod that uses a different container image or command. Troubleshoot distroless containers by adding a new container with debugging tools, either in a new copy of the pod or using an ephemeral container. (Ephemeral containers are an alpha feature that are not enabled by default.) Troubleshoot on a node by creating a container running in the host namespaces and with access to the host's filesystem. Note that as a new builtin command, `kubectl debug` takes priority over any `kubectl` plugin named "debug". You will need to rename the affected plugin. Invocations using `kubectl alpha debug` are now deprecated and will be removed in a subsequent release. Update your scripts to use `kubectl debug` instead of `kubectl alpha debug`! For more information about kubectl debug, see Debugging Running Pods on the Kubernetes website, kubectl help debug, or reach out to SIG CLI by visiting #sig-cli or commenting on enhancement #1441.

Removing deprecated flags in kubeadm

`kubeadm` applies a number of deprecations and removals of deprecated features in this release. More details are available in the Urgent Upgrade Notes and Kind / Deprecation sections.

Pod Hostname as FQDN graduates to Beta

Previously introduced in 1.19 behind a feature gate, `SetHostnameAsFQDN` is now enabled by default. More details on this behavior is available in documentation for DNS for Services and Pods

`TokenRequest` / `TokenRequestProjection` graduates to General Availability

Service account tokens bound to pod is now a stable feature. The feature gates will be removed in 1.21 release. For more information, refer to notes below on the changelogs.

RuntimeClass feature graduates to General Availability.

The `node.k8s.io` API groups are promoted from `v1beta1` to `v1`. `v1beta1` is now deprecated and will be removed in a future release, please start using `v1`. (#95718, @SergeyKanzhelev) [SIG Apps, Auth, Node, Scheduling and Testing]

Cloud Controller Manager now exclusively shipped by Cloud Provider

Kubernetes will no longer ship an instance of the Cloud Controller Manager binary. Each Cloud Provider is expected to ship their own instance of this binary. Details for a Cloud Provider to create an instance of such a binary can be found under here. Anyone with questions on building a Cloud Controller Manager should reach out to SIG Cloud Provider. Questions about the Cloud Controller Manager on a Managed Kubernetes solution should go to the relevant Cloud Provider. Questions about the Cloud Controller Manager on a non managed solution can be brought up with SIG Cloud Provider.

Known Issues

Summary API in kubelet doesn't have accelerator metrics

Currently, cadvisor_stats_provider provides AcceleratorStats but cri_stats_provider does not. As a result, when using cri_stats_provider, kubelet's Summary API does not have accelerator metrics. There is an open work in progress to fix this.

Urgent Upgrade Notes

(No, really, you MUST read this before you upgrade)

- A bug was fixed in kubelet where exec probe timeouts were not respected. This may result in unexpected behavior since the default timeout (if not specified) is `1s` which may be too small for some exec probes. Ensure that pods relying on this behavior are updated to correctly handle probe timeouts. See configure probe section of the documentation for more details.
    - This change in behavior may be unexpected for some clusters and can be disabled by turning off the `ExecProbeTimeout` feature gate.

This gate will be locked and removed in future releases so that exec probe timeouts are always respected. (#94115, @andrewsykim) [SIG Node and Testing]

- RuntimeClass feature graduates to General Availability. Promote `node.k8s.io` API groups from `v1beta1` to `v1`. `v1beta1` is now deprecated and will be removed in a future release, please start using `v1`. (#95718, @SergeyKanzhelev) [SIG Apps, Auth, Node, Scheduling and Testing]

- API priority and fairness graduated to beta. 1.19 servers with APF turned on should not be run in a multi-server cluster with 1.20+ servers. (#96527, @adtac) [SIG API Machinery and Testing]

- For CSI drivers, kubelet no longer creates the target_path for NodePublishVolume in accordance with the CSI spec. Kubelet also no longer checks if staging and target paths are mounts or corrupted. CSI drivers need to be idempotent and do any necessary mount verification. (#88759, @andyzhangx) [SIG Storage]

- Kubeadm: http://git.k8s.io/enhancements/keps/sig-cluster-lifecycle/kubeadm/2067-rename-master-label-taint/README.md (#95382, @neolit123) [SIG Cluster Lifecycle]
  - The label applied to control-plane nodes "node-role.kubernetes.io/master" is now deprecated and will be removed in a future release after a GA deprecation period.
  - Introduce a new label "node-role.kubernetes.io/control-plane" that will be applied in parallel to "node-role.kubernetes.io/master" until the removal of the "node-role.kubernetes.io/master" label.
  - Make "kubeadm upgrade apply" add the "node-role.kubernetes.io/control-plane" label on existing nodes that only have the "node-role.kubernetes.io/master" label during upgrade.
  - Please adapt your tooling built on top of kubeadm to use the "node-role.kubernetes.io/control-plane" label.
  - The taint applied to control-plane nodes "node-role.kubernetes.io/master:NoSchedule" is now deprecated and will be removed in a future release after a GA deprecation period.
  - Apply toleration for a new, future taint "node-role.kubernetes.io/control-plane:NoSchedule" to the kubeadm CoreDNS / kube-dns managed manifests. Note that this taint is not yet applied to kubeadm control-plane nodes.
  - Please adapt your workloads to tolerate the same future taint preemptively.

- Kubeadm: improve the validation of serviceSubnet and podSubnet. ServiceSubnet has to be limited in size, due to implementation details, and the mask can not allocate more than 20 bits. PodSubnet validates against the corresponding cluster "--node-cidr-mask-size" of the kube-controller-manager, it fail if the values are not compatible. kubeadm no longer sets the node-mask automatically on IPv6 deployments, you must check that your IPv6 service subnet mask is compatible with the default node mask /64 or set it accordenly. Previously, for IPv6, if the podSubnet had a mask lower than /112, kubeadm calculated a node-

mask to be multiple of eight and splitting the available bits to maximise the number used for nodes. (#95723, @aojea) [SIG Cluster Lifecycle]

- The deprecated flag --experimental-kustomize is now removed from kubeadm commands. Use --experimental-patches instead, which was introduced in 1.19. Migration infromation available in --help description for --exprimental-patches. (#94871, @neolit123)

- Windows hyper-v container featuregate is deprecated in 1.20 and will be removed in 1.21 (#95505, @wawa0210) [SIG Node and Windows]

- The kube-apiserver ability to serve on an insecure port, deprecated since v1.10, has been removed. The insecure address flags `--address` and `--insecure-bind-address` have no effect in kube-apiserver and will be removed in v1.24. The insecure port flags `--port` and `--insecure-port` may only be set to 0 and will be removed in v1.24. (#95856, @knight42, [SIG API Machinery, Node, Testing])

- Add dual-stack Services (alpha). This is a BREAKING CHANGE to an alpha API. It changes the dual-stack API wrt Service from a single ipFamily field to 3 fields: ipFamilyPolicy (SingleStack, PreferDualStack, RequireDualStack), ipFamilies (a list of families assigned), and clusterIPs (inclusive of clusterIP). Most users do not need to set anything at all, defaulting will handle it for them. Services are single-stack unless the user asks for dual-stack. This is all gated by the "IPv6DualStack" feature gate. (#91824, @khenidak) [SIG API Machinery, Apps, CLI, Network, Node, Scheduling and Testing]

- `TokenRequest` and `TokenRequestProjection` are now GA features. The following flags are required by the API server:

    - `--service-account-issuer`, should be set to a URL identifying the API server that will be stable over the cluster lifetime.
    - `--service-account-key-file`, set to one or more files containing one or more public keys used to verify tokens.
    - `--service-account-signing-key-file`, set to a file containing a private key to use to sign service account tokens. Can be the same file given to `kube-controller-manager` with `--service-account-private-key-file`. (#95896, @zshihang) [SIG API Machinery, Auth, Cluster Lifecycle]

- kubeadm: make the command "kubeadm alpha kubeconfig user" accept a "--config" flag and remove the following flags:

    - apiserver-advertise-address / apiserver-bind-port: use either localAPIEndpoint from InitConfiguration or controlPlaneEndpoint from ClusterConfiguration.
    - cluster-name: use clusterName from ClusterConfiguration
    - cert-dir: use certificatesDir from ClusterConfiguration (#94879, @knight42) [SIG Cluster Lifecycle]

- Resolves non-deterministic behavior of the garbage collection controller when ownerReferences with incorrect data are encountered.

Events with a reason of `OwnerRefInvalidNamespace` are recorded when namespace mismatches between child and owner objects are detected. The kubectl-check-ownerreferences tool can be run prior to upgrading to locate existing objects with invalid ownerReferences.

    - A namespaced object with an ownerReference referencing a uid of a namespaced kind which does not exist in the same namespace is now consistently treated as though that owner does not exist, and the child object is deleted.
    - A cluster-scoped object with an ownerReference referencing a uid of a namespaced kind is now consistently treated as though that owner is not resolvable, and the child object is ignored by the garbage collector. (#92743, @liggitt) [SIG API Machinery, Apps and Testing]

Changes by Kind

Deprecation

- Docker support in the kubelet is now deprecated and will be removed in a future release. The kubelet uses a module called "dockershim" which implements CRI support for Docker and it has seen maintenance issues in the Kubernetes community. We encourage you to evaluate moving to a container runtime that is a full-fledged implementation of CRI (v1alpha1 or v1 compliant) as they become available. (#94624, @dims) [SIG Node]
- Kubeadm: deprecate self-hosting support. The experimental command "kubeadm alpha self-hosting" is now deprecated and will be removed in a future release. (#95125, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: graduate the "kubeadm alpha certs" command to a parent command "kubeadm certs". The command "kubeadm alpha certs" is deprecated and will be removed in a future release. Please migrate. (#94938, @yagonobre) [SIG Cluster Lifecycle]
- Kubeadm: remove the deprecated "kubeadm alpha kubelet config enable-dynamic" command. To continue using the feature please defer to the guide for "Dynamic Kubelet Configuration" at k8s.io. This change also removes the parent command "kubeadm alpha kubelet" as there are no more sub-commands under it for the time being. (#94668, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: remove the deprecated --kubelet-config flag for the command "kubeadm upgrade node" (#94869, @neolit123) [SIG Cluster Lifecycle]
- Kubectl: deprecate --delete-local-data (#95076, @dougsland) [SIG CLI, Cloud Provider and Scalability]
- Kubelet's deprecated endpoint `metrics/resource/v1alpha1` has been removed, please adopt `metrics/resource`. (#94272, @RainbowMango) [SIG Instrumentation and Node]
- Removes deprecated scheduler metrics DeprecatedSchedulingDuration, DeprecatedSchedulingAlgorithmPredicateEvaluationSecondsDuration, DeprecatedSchedulingAlgorithmPriorityEvaluationSecondsDuration (#94884, @arghya88) [SIG Instrumentation and Scheduling]

- Scheduler alpha metrics binding_duration_seconds and scheduling_algorithm_preemption_evaluation_seconds are deprecated, Both of those metrics are now covered as part of framework_extension_point_duration_seconds, the former as a PostFilter the latter and a Bind plugin. The plan is to remove both in 1.21 (#95001, @arghya88) [SIG Instrumentation and Scheduling]
- Support 'controlplane' as a valid EgressSelection type in the EgressSelectorConfiguration API. 'Master' is deprecated and will be removed in v1.22. (#95235, @andrewsykim) [SIG API Machinery]
- The v1alpha1 PodPreset API and admission plugin has been removed with no built-in replacement. Admission webhooks can be used to modify pods on creation. (#94090, @deads2k) [SIG API Machinery, Apps, CLI, Cloud Provider, Scalability and Testing]

API Change

- `TokenRequest` and `TokenRequestProjection` features have been promoted to GA. This feature allows generating service account tokens that are not visible in Secret objects and are tied to the lifetime of a Pod object. See https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/#service-account-token-volume-projection for details on configuring and using this feature. The `TokenRequest` and `TokenRequestProjection` feature gates will be removed in v1.21.
  - kubeadm's kube-apiserver Pod manifest now includes the following flags by default "--service-account-key-file", "--service-account-signing-key-file", "--service-account-issuer". (#93258, @zshihang) [SIG API Machinery, Auth, Cluster Lifecycle, Storage and Testing]
- A new `nofuzz` go build tag now disables gofuzz support. Release binaries enable this. (#92491, @BenTheElder) [SIG API Machinery]
- Add WindowsContainerResources and Annotations to CRI-API UpdateContainerResourcesRequest (#95741, @katiewasnothere) [SIG Node]
- Add a `serving` and `terminating` condition to the EndpointSlice API. `serving` tracks the readiness of endpoints regardless of their terminating state. This is distinct from `ready` since `ready` is only true when pods are not terminating. `terminating` is true when an endpoint is terminating. For pods this is any endpoint with a deletion timestamp. (#92968, @andrewsykim) [SIG Apps and Network]
- Add dual-stack Services (alpha). This is a BREAKING CHANGE to an alpha API. It changes the dual-stack API wrt Service from a single ipFamily field to 3 fields: ipFamilyPolicy (SingleStack, PreferDualStack, RequireDualStack), ipFamilies (a list of families assigned), and clusterIPs (inclusive of clusterIP). Most users do not need to set anything at all, defaulting will handle it for them. Services are single-stack unless the user asks for dual-stack. This is all gated by the "IPv6DualStack" feature gate. (#91824, @khenidak) [SIG API Machinery, Apps, CLI, Network, Node, Scheduling and Testing]
- Add support for hugepages to downward API (#86102, @derekwaynecarr) [SIG API Machinery, Apps, CLI, Network, Node, Scheduling and Testing]

- Adds kubelet alpha feature, `GracefulNodeShutdown` which makes kubelet aware of node system shutdowns and result in graceful termination of pods during a system shutdown. (#96129, @bobbypage) [SIG Node]
- AppProtocol is now GA for Endpoints and Services. The ServiceAppProtocol feature gate will be deprecated in 1.21. (#96327, @robscott) [SIG Apps and Network]
- Automatic allocation of NodePorts for services with type LoadBalancer can now be disabled by setting the (new) parameter Service.spec.allocateLoadBalancerNodePorts=false. The default is to allocate NodePorts for services with type LoadBalancer which is the existing behavior. (#92744, @uablrek) [SIG Apps and Network]
- Certain fields on Service objects will be automatically cleared when changing the service's `type` to a mode that does not need those fields. For example, changing from type=LoadBalancer to type=ClusterIP will clear the NodePort assignments, rather than forcing the user to clear them. (#95196, @thockin) [SIG API Machinery, Apps, Network and Testing]
- Document that ServiceTopology feature is required to use `service.spec.topologyKeys`. (#96528, @andrewsykim) [SIG Apps]
- EndpointSlice has a new NodeName field guarded by the EndpointSliceNodeName feature gate.
  - EndpointSlice topology field will be deprecated in an upcoming release.
  - EndpointSlice "IP" address type is formally removed after being deprecated in Kubernetes 1.17.
  - The discovery.k8s.io/v1alpha1 API is deprecated and will be removed in Kubernetes 1.21. (#96440, @robscott) [SIG API Machinery, Apps and Network]
- External facing API podresources is now available under k8s.io/kubelet/pkg/apis/ (#92632, @RenaudWasTaken) [SIG Node and Testing]
- Fewer candidates are enumerated for preemption to improve performance in large clusters. (#94814, @adtac)
- Fix conversions for custom metrics. (#94481, @wojtek-t) [SIG API Machinery and Instrumentation]
- GPU metrics provided by kubelet are now disabled by default. (#95184, @RenaudWasTaken)
- If BoundServiceAccountTokenVolume is enabled, 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` (#96273, @zshihang) [SIG API Machinery and Auth]
- Introduce alpha support for exec-based container registry credential provider plugins in the kubelet. (#94196, @andrewsykim) [SIG Node and Release]
- Introduces a metric source for HPAs which allows scaling based on container resource usage. (#90691, @arjunrn) [SIG API Machinery, Apps, Autoscaling and CLI]
- Kube-apiserver now deletes expired kube-apiserver Lease objects:
  - The feature is under feature gate `APIServerIdentity`.

    - A flag is added to kube-apiserver: `identity-lease-garbage-collection-check-period-seconds` (#95895, @roycaihw) [SIG API Machinery, Apps, Auth and Testing]
- Kube-controller-manager: volume plugins can be restricted from contacting local and loopback addresses by setting `--volume-host-allow-local-loopback=false`, or from contacting specific CIDR ranges by setting `--volume-host-cidr-denylist` (for example, `--volume-host-cidr-denylist=127.0.0.1/28,feed::/16`) (#91785, @mattcary) [SIG API Machinery, Apps, Auth, CLI, Network, Node, Storage and Testing]
- Migrate scheduler, controller-manager and cloud-controller-manager to use LeaseLock (#94603, @wojtek-t) [SIG API Machinery, Apps, Cloud Provider and Scheduling]
- Modify DNS-1123 error messages to indicate that RFC 1123 is not followed exactly (#94182, @mattfenwick) [SIG API Machinery, Apps, Auth, Network and Node]
- Move configurable fsgroup change policy for pods to beta (#96376, @gnufied) [SIG Apps and Storage]
- New flag is introduced, i.e. --topology-manager-scope=container|pod. The default value is the "container" scope. (#92967, @cezaryzukowski) [SIG Instrumentation, Node and Testing]
- New parameter `defaultingType` for `PodTopologySpread` plugin allows to use k8s defined or user provided default constraints (#95048, @alculquicondor) [SIG Scheduling]
- NodeAffinity plugin can be configured with AddedAffinity. (#96202, @alculquicondor) [SIG Node, Scheduling and Testing]
- Promote RuntimeClass feature to GA. Promote node.k8s.io API groups from v1beta1 to v1. (#95718, @SergeyKanzhelev) [SIG Apps, Auth, Node, Scheduling and Testing]
- Reminder: The labels "failure-domain.beta.kubernetes.io/zone" and "failure-domain.beta.kubernetes.io/region" are deprecated in favor of "topology.kubernetes.io/zone" and "topology.kubernetes.io/region" respectively. All users of the "failure-domain.beta..." labels should switch to the "topology..." equivalents. (#96033, @thockin) [SIG API Machinery, Apps, CLI, Cloud Provider, Network, Node, Scheduling, Storage and Testing]
- Server Side Apply now treats LabelSelector fields as atomic (meaning the entire selector is managed by a single writer and updated together), since they contain interrelated and inseparable fields that do not merge in intuitive ways. (#93901, @jpbetz) [SIG API Machinery, Auth, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation, Network, Node, Storage and Testing]
- Services will now have a `clusterIPs` field to go with `clusterIP`. `clusterIPs[0]` is a synonym for `clusterIP` and will be syncronized on create and update operations. (#95894, @thockin) [SIG Network]
- The ServiceAccountIssuerDiscovery feature gate is now Beta and enabled by default. (#91921, @mtaufen) [SIG Auth]
- The status of v1beta1 CRDs without "preserveUnknownFields:false" now shows a violation, "spec.preserveUnknownFields: Invalid value: true: must be false". (#93078, @vareti)
- The usage of mixed protocol values in the same LoadBalancer Service is possible if the new feature gate MixedProtocolLBService is enabled.

The feature gate is disabled by default. The user has to enable it for the API Server. (#94028, @janosi) [SIG API Machinery and Apps]
- This PR will introduce a feature gate CSIServiceAccountToken with two additional fields in `CSIDriverSpec`. (#93130, @zshihang) [SIG API Machinery, Apps, Auth, CLI, Network, Node, Storage and Testing]
- Users can try the cronjob controller v2 using the feature gate. This will be the default controller in future releases. (#93370, @alaypatel07) [SIG API Machinery, Apps, Auth and Testing]
- VolumeSnapshotDataSource moves to GA in 1.20 release (#95282, @xing-yang) [SIG Apps]
- WinOverlay feature graduated to beta (#94807, @ksubrmnn) [SIG Windows]

Feature

- **Additional documentation e.g., KEPs (Kubernetes Enhancement Proposals), usage docs, etc.**:

- A new metric `apiserver_request_filter_duration_seconds` has been introduced that measures request filter latency in seconds. (#95207, @tkashem) [SIG API Machinery and Instrumentation]

- A new set of alpha metrics are reported by the Kubernetes scheduler under the `/metrics/resources` endpoint that allow administrators to easily see the resource consumption (requests and limits for all resources on the pods) and compare it to actual pod usage or node capacity. (#94866, @smarterclayton) [SIG API Machinery, Instrumentation, Node and Scheduling]

- Add --experimental-logging-sanitization flag enabling runtime protection from leaking sensitive data in logs (#96370, @serathius) [SIG API Machinery, Cluster Lifecycle and Instrumentation]

- Add a StorageVersionAPI feature gate that makes API server update storageversions before serving certain write requests. This feature allows the storage migrator to manage storage migration for built-in resources. Enabling internal.apiserver.k8s.io/v1alpha1 API and APIServerIdentity feature gate are required to use this feature. (#93873, @roycaihw) [SIG API Machinery, Auth and Testing]

- Add a metric for time taken to perform recursive permission change (#95866, @JornShen) [SIG Instrumentation and Storage]

- Add a new `vSphere` metric: `cloudprovider_vsphere_vcenter_versions`. It's content show `vCenter` hostnames with the associated server version. (#94526, @Danil-Grigorev) [SIG Cloud Provider and Instrumentation]

- Add a new flag to set priority for the kubelet on Windows nodes so that workloads cannot overwhelm the node there by disrupting kubelet process. (#96051, @ravisantoshgudimetla) [SIG Node and Windows]

- Add feature to size memory backed volumes (#94444, @derekwaynecarr) [SIG Storage and Testing]
- Add foreground cascading deletion to kubectl with the new `kubectl delete foreground|background|orphan` option. (#93384, @zhouya0)
- Add metrics for azure service operations (route and loadbalancer). (#94124, @nilo19) [SIG Cloud Provider and Instrumentation]
- Add network rule support in Azure account creation. (#94239, @andyzhangx)
- Add node_authorizer_actions_duration_seconds metric that can be used to estimate load to node authorizer. (#92466, @mborsz) [SIG API Machinery, Auth and Instrumentation]
- Add pod_ based CPU and memory metrics to Kubelet's /metrics/resource endpoint (#95839, @egernst) [SIG Instrumentation, Node and Testing]
- Added `get-users` and `delete-user` to the `kubectl config` subcommand (#89840, @eddiezane) [SIG CLI]
- Added counter metric "apiserver_request_self" to count API server self-requests with labels for verb, resource, and subresource. (#94288, @LogicalShark) [SIG API Machinery, Auth, Instrumentation and Scheduling]
- Added new k8s.io/component-helpers repository providing shared helper code for (core) components. (#92507, @ingvagabund) [SIG Apps, Node, Release and Scheduling]
- Adds `create ingress` command to `kubectl` (#78153, @amimof) [SIG CLI and Network]
- Adds a headless service on node-local-cache addon. (#88412, @stafot) [SIG Cloud Provider and Network]
- Allow cross compilation of kubernetes on different platforms. (#94403, @bnrjee) [SIG Release]
- Azure: Support multiple services sharing one IP address (#94991, @nilo19) [SIG Cloud Provider]
- CRDs: For structural schemas, non-nullable null map fields will now be dropped and defaulted if a default is available. null items in list will continue being preserved, and fail validation if not nullable. (#95423, @apelisse) [SIG API Machinery]
- Changed: default "Accept: /" header added to HTTP probes. See https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#http-probes (https://github.com/kubernetes/website/pull/24756) (#95641, @fonsecas72) [SIG Network and Node]

- Client-go credential plugins can now be passed in the current cluster information via the KUBERNETES_EXEC_INFO environment variable. (#95489, @ankeesler) [SIG API Machinery and Auth]

- Command to start network proxy changes from 'KUBE_ENABLE_EGRESS_VIA_KONNECTIVITY_SERVICE ./cluster/kube-up.sh' to 'KUBE_ENABLE_KONNECTIVITY_SERVICE=true ./hack/kube-up.sh' (#92669, @Jefftree) [SIG Cloud Provider]

- Configure AWS LoadBalancer health check protocol via service annotations. (#94546, @kishorj)

- DefaultPodTopologySpread graduated to Beta. The feature gate is enabled by default. (#95631, @alculquicondor) [SIG Scheduling and Testing]

- E2e test for PodFsGroupChangePolicy (#96247, @saikat-royc) [SIG Storage and Testing]

- Ephemeral containers now apply the same API defaults as initContainers and containers (#94896, @wawa0210) [SIG Apps and CLI]

- Gradudate the Pod Resources API to G.A Introduces the pod_resources_endpoint_requests_total metric which tracks the total number of requests to the pod resources API (#92165, @RenaudWasTaken) [SIG Instrumentation, Node and Testing]

- In dual-stack bare-metal clusters, you can now pass dual-stack IPs to `kubelet --node-ip`. eg: `kubelet --node-ip 10.1.0.5,fd01::0005`. This is not yet supported for non-bare-metal clusters.

  In dual-stack clusters where nodes have dual-stack addresses, hostNetwork pods will now get dual-stack PodIPs. (#95239, @danwinship) [SIG Network and Node]

- Introduce api-extensions category which will return: mutating admission configs, validating admission configs, CRDs and APIServices when used in kubectl get, for example. (#95603, @soltysh) [SIG API Machinery]

- Introduces a new GCE specific cluster creation variable KUBE_PROXY_DISABLE. When set to true, this will skip over the creation of kube-proxy (whether the daemonset or static pod). This can be used to control the lifecycle of kube-proxy separately from the lifecycle of the nodes. (#91977, @varunmar) [SIG Cloud Provider]

- Kube-apiserver now maintains a Lease object to identify itself:

  - The feature is under feature gate `APIServerIdentity`.
  - Two flags are added to kube-apiserver: `identity-lease-duration-seconds`, `identity-lease-renew-interval-seconds` (#95533, @roycaihw) [SIG API Machinery]

- Kube-apiserver: The timeout used when making health check calls to etcd can now be configured with `--etcd-healthcheck-timeout`. The default timeout is 2 seconds, matching the previous behavior. (#93244, @Sh4d1) [SIG API Machinery]

- Kube-apiserver: added support for compressing rotated audit log files with `--audit-log-compress` (#94066, @lojies) [SIG API Machinery and Auth]

- Kubeadm now prints warnings instead of throwing errors if the current system time is outside of the NotBefore and NotAfter bounds of a loaded certificate. (#94504, @neolit123)

- Kubeadm: Add a preflight check that the control-plane node has at least 1700MB of RAM (#93275, @xlgao-zju) [SIG Cluster Lifecycle]

- Kubeadm: add the "--cluster-name" flag to the "kubeadm alpha kubeconfig user" to allow configuring the cluster name in the generated kubeconfig file (#93992, @prabhu43) [SIG Cluster Lifecycle]

- Kubeadm: add the "--kubeconfig" flag to the "kubeadm init phase upload-certs" command to allow users to pass a custom location for a kubeconfig file. (#94765, @zhanw15) [SIG Cluster Lifecycle]

- Kubeadm: make etcd pod request 100m CPU, 100Mi memory and 100Mi ephemeral_storage by default (#94479, @knight42) [SIG Cluster Lifecycle]

- Kubeadm: make the command "kubeadm alpha kubeconfig user" accept a "--config" flag and remove the following flags:

    - apiserver-advertise-address / apiserver-bind-port: use either localAPIEndpoint from InitConfiguration or controlPlaneEndpoint from ClusterConfiguration.
    - cluster-name: use clusterName from ClusterConfiguration
    - cert-dir: use certificatesDir from ClusterConfiguration (#94879, @knight42) [SIG Cluster Lifecycle]

- Kubectl create now supports creating ingress objects. (#94327, @rikatz) [SIG CLI and Network]

- Kubectl rollout history sts/sts-name --revision=some-revision will start showing the detailed view of the sts on that specified revision (#86506, @dineshba) [SIG CLI]

- Kubectl: Previously users cannot provide arguments to a external diff tool via KUBECTL_EXTERNAL_DIFF env. This release now allow users to specify args to KUBECTL_EXTERNAL_DIFF env. (#95292, @dougsland) [SIG CLI]

- Kubemark now supports both real and hollow nodes in a single cluster. (#93201, @ellistarn) [SIG Scalability]

Kubernetes E2E test image manifest lists now contain Windows images. (#77398, @claudiubelu) [SIG Testing and Windows]

- Kubernetes is now built using go1.15.2
  - build: Update to k/repo-infra@v0.1.1 (supports go1.15.2)
  - build: Use go-runner:buster-v2.0.1 (built using go1.15.1)
  - bazel: Replace --features with Starlark build settings flag
  - hack/lib/util.sh: some bash cleanups
    - switched one spot to use kube::logging
    - make kube::util::find-binary return an error when it doesn't find anything so that hack scripts fail fast instead of with '' binary not found errors.
    - this required deleting some genfeddoc stuff. the binary no longer exists in k/k repo since we removed federation/, and I don't see it in https://github.com/kubernetes-sigs/kubefed/ either. I'm assuming that it's gone for good now.
  - bazel: output go_binary rule directly from go_binary_conditional_pure

    From: @mikedanese: Instead of aliasing. Aliases are annoying in a number of ways. This is specifically bugging me now because they make the action graph harder to analyze programmatically. By using aliases here, we would need to handle potentially aliased go_binary targets and dereference to the effective target.

    The comment references an issue with `pure = select(...)` which appears to be resolved considering this now builds.
  - make kube::util::find-binary not dependent on bazel-out/ structure

    Implement an aspect that outputs go_build_mode metadata for go binaries, and use that during binary selection. (#94449, @justaugustus) [SIG Architecture, CLI, Cluster Lifecycle, Node, Release and Testing]
- Kubernetes is now built using go1.15.5
  - build: Update to k/repo-infra@v0.1.2 (supports go1.15.5) (#95776, @justaugustus) [SIG Cloud Provider, Instrumentation, Release and Testing]
- New default scheduling plugins order reduces scheduling and preemption latency when taints and node affinity are used (#95539, @soulxu) [SIG Scheduling]
- Only update Azure data disks when attach/detach (#94265, @andyzhangx) [SIG Cloud Provider]

- Promote SupportNodePidsLimit to GA to provide node-to-pod PID isolation. Promote SupportPodPidsLimit to GA to provide ability to limit PIDs per pod. (#94140, @derekwaynecarr)

- SCTP support in API objects (Pod, Service, NetworkPolicy) is now GA. Note that this has no effect on whether SCTP is enabled on nodes at the kernel level, and note that some cloud platforms and network plugins do not support SCTP traffic. (#95566, @danwinship) [SIG Apps and Network]

- Scheduler now ignores Pod update events if the resourceVersion of old and new Pods are identical. (#96071, @Huang-Wei) [SIG Scheduling]

- Scheduling Framework: expose Run[Pre]ScorePlugins functions to PreemptionHandle which can be used in PostFilter extention point. (#93534, @everpeace) [SIG Scheduling and Testing]

- SelectorSpreadPriority maps to PodTopologySpread plugin when DefaultPodTopologySpread feature is enabled (#95448, @alculquicondor) [SIG Scheduling]

- Send GCE node startup scripts logs to console and journal. (#95311, @karan)

- SetHostnameAsFQDN has been graduated to Beta and therefore it is enabled by default. (#95267, @javidiaz) [SIG Node]

- Support [service.beta.kubernetes.io/azure-pip-ip-tags] annotations to allow customers to specify ip-tags to influence public-ip creation in Azure [Tag1=Value1, Tag2=Value2, etc.] (#94114, @MarcPow) [SIG Cloud Provider]

- Support custom tags for cloud provider managed resources (#96450, @nilo19) [SIG Cloud Provider]

- Support customize load balancer health probe protocol and request path (#96338, @nilo19) [SIG Cloud Provider]

- Support for Windows container images (OS Versions: 1809, 1903, 1909, 2004) was added the pause:3.4 image. (#91452, @claudiubelu) [SIG Node, Release and Windows]

- Support multiple standard load balancers in one cluster (#96111, @nilo19) [SIG Cloud Provider]

- The beta `RootCAConfigMap` feature gate is enabled by default and causes kube-controller-manager to publish a "kube-root-ca.crt" ConfigMap to every namespace. This ConfigMap contains a CA bundle used for verifying connections to the kube-apiserver. (#96197, @zshihang) [SIG API Machinery, Apps, Auth and Testing]

- The kubelet_runtime_operations_duration_seconds metric buckets were set to 0.005 0.0125 0.03125 0.078125 0.1953125 0.48828125 1.220703125 3.0517578125 7.62939453125 19.073486328125

47.6837158203125 119.20928955078125 298.0232238769531 and 745.0580596923828 seconds (#96054, @alvaroaleman) [SIG Instrumentation and Node]

- There is a new pv_collector_total_pv_count metric that counts persistent volumes by the volume plugin name and volume mode. (#95719, @tsmetana) [SIG Apps, Instrumentation, Storage and Testing]
- Volume snapshot e2e test to validate PVC and VolumeSnapshotContent finalizer (#95863, @RaunakShah) [SIG Cloud Provider, Storage and Testing]
- Warns user when executing kubectl apply/diff to resource currently being deleted. (#95544, @SaiHarshaK) [SIG CLI]
- `kubectl alpha debug` has graduated to beta and is now `kubectl debug`. (#96138, @verb) [SIG CLI and Testing]
- `kubectl debug` gains support for changing container images when copying a pod for debugging, similar to how `kubectl set image` works. See `kubectl help debug` for more information. (#96058, @verb) [SIG CLI]

Documentation

- Fake dynamic client: document that List does not preserve TypeMeta in UnstructuredList (#95117, @andrewsykim) [SIG API Machinery]
- Kubelet: remove alpha warnings for CNI flags. (#94508, @andrewsykim) [SIG Network and Node]
- Updates docs and guidance on cloud provider InstancesV2 and Zones interface for external cloud providers:
  - removes experimental warning for InstancesV2
  - document that implementation of InstancesV2 will disable calls to Zones
  - deprecate Zones in favor of InstancesV2 (#96397, @andrewsykim) [SIG Cloud Provider]

Failing Test

- Resolves an issue running Ingress conformance tests on clusters which use finalizers on Ingress objects to manage releasing load balancer resources (#96742, @spencerhance) [SIG Network and Testing]
- The Conformance test "validates that there is no conflict between pods with same hostPort but different hostIP and protocol" now validates the connectivity to each hostPort, in addition to the functionality. (#96627, @aojea) [SIG Scheduling and Testing]

Bug or Regression

- Add kubectl wait --ignore-not-found flag (#90969, @zhouya0) [SIG CLI]

- Added support to kube-proxy for externalTrafficPolicy=Local setting via Direct Server Return (DSR) load balancers on Windows. (#93166, @elweb9858) [SIG Network]

- Alter wording to describe pods using a pvc (#95635, @RaunakShah) [SIG CLI]

- An issues preventing volume expand controller to annotate the PVC with `volume.kubernetes.io/storage-resizer` when the PVC StorageClass is already updated to the out-of-tree provisioner is now fixed. (#94489, @ialidzhikov) [SIG API Machinery, Apps and Storage]

- Azure ARM client: don't segfault on empty response and http error (#94078, @bpineau) [SIG Cloud Provider]

- Azure armclient backoff step defaults to 1 (no retry). (#94180, @feiskyer)

- Azure: fix a bug that kube-controller-manager would panic if wrong Azure VMSS name is configured (#94306, @knight42) [SIG Cloud Provider]

- Both apiserver_request_duration_seconds metrics and RequestReceivedTimestamp fields of an audit event now take into account the time a request spends in the apiserver request filters. (#94903, @tkashem)

- Build/lib/release: Explicitly use '--platform' in building server images

  When we switched to go-runner for building the apiserver, controller-manager, and scheduler server components, we no longer reference the individual architectures in the image names, specifically in the 'FROM' directive of the server image Dockerfiles.

  As a result, server images for non-amd64 images copy in the go-runner amd64 binary instead of the go-runner that matches that architecture.

  This commit explicitly sets the '--platform=linux/${arch}' to ensure we're pulling the correct go-runner arch from the manifest list.

  Before: `FROM ${base_image}`

  After: `FROM --platform=linux/${arch} ${base_image}` (#94552, @justaugustus) [SIG Release]

- Bump node-problem-detector version to v0.8.5 to fix OOM detection in with Linux kernels 5.1+ (#96716, @tosi3k) [SIG Cloud Provider, Scalability and Testing]

- CSIDriver object can be deployed during volume attachment. (#93710, @Jiawei0227) [SIG Apps, Node, Storage and Testing]

- Ceph RBD volume expansion now works even when ceph.conf was not provided. (#92027, @juliantaylor)

- Change plugin name in fsgroupapplymetrics of csi and flexvolume to distinguish different driver (#95892, @JornShen) [SIG Instrumentation, Storage and Testing]
- Change the calculation of pod UIDs so that static pods get a unique value - will cause all containers to be killed and recreated after in-place upgrade. (#87461, @bboreham) [SIG Node]
- Change the mount way from systemd to normal mount except ceph and glusterfs intree-volume. (#94916, @smileusd) [SIG Apps, Cloud Provider, Network, Node, Storage and Testing]
- Changes to timeout parameter handling in 1.20.0-beta.2 have been reverted to avoid breaking backwards compatibility with existing clients. (#96727, @liggitt) [SIG API Machinery and Testing]
- Clear UDP conntrack entry on endpoint changes when using nodeport (#71573, @JacobTanenbaum) [SIG Network]
- Cloud node controller: handle empty providerID from getProviderID (#95342, @nicolehanjing) [SIG Cloud Provider]
- Disable watchcache for events (#96052, @wojtek-t) [SIG API Machinery]
- Disabled `LocalStorageCapacityIsolation` feature gate is honored during scheduling. (#96092, @Huang-Wei) [SIG Scheduling]
- Do not fail sorting empty elements. (#94666, @soltysh) [SIG CLI]
- Dual-stack: make nodeipam compatible with existing single-stack clusters when dual-stack feature gate become enabled by default (#90439, @SataQiu) [SIG API Machinery]
- Duplicate owner reference entries in create/update/patch requests now get deduplicated by the API server. The client sending the request now receives a warning header in the API response. Clients should stop sending requests with duplicate owner references. The API server may reject such requests as early as 1.24. (#96185, @roycaihw) [SIG API Machinery and Testing]
- Endpoint slice controller now mirrors parent's service label to its corresponding endpoint slices. (#94443, @aojea)
- Ensure getPrimaryInterfaceID not panic when network interfaces for Azure VMSS are null (#94355, @feiskyer) [SIG Cloud Provider]
- Exposes and sets a default timeout for the SubjectAccessReview client for DelegatingAuthorizationOptions (#95725, @p0lyn0mial) [SIG API Machinery and Cloud Provider]
- Exposes and sets a default timeout for the TokenReview client for DelegatingAuthenticationOptions (#96217, @p0lyn0mial) [SIG API Machinery and Cloud Provider]

- Fix CVE-2020-8555 for Quobyte client connections. (#95206, @misterikkit) [SIG Storage]
- Fix IP fragmentation of UDP and TCP packets not supported issues on LoadBalancer rules (#96464, @nilo19) [SIG Cloud Provider]
- Fix a bug that DefaultPreemption plugin is disabled when using (legacy) scheduler policy. (#96439, @Huang-Wei) [SIG Scheduling and Testing]
- Fix a bug where loadbalancer deletion gets stuck because of missing resource group. (#93962, @phiphi282)
- Fix a concurrent map writes error in kubelet (#93773, @knight42) [SIG Node]
- Fix a panic in `kubectl debug` when a pod has multiple init or ephemeral containers. (#94580, @kiyoshim55)
- Fix a regression where kubeadm bails out with a fatal error when an optional version command line argument is supplied to the "kubeadm upgrade plan" command (#94421, @rosti) [SIG Cluster Lifecycle]
- Fix azure disk attach failure for disk size bigger than 4TB (#95463, @andyzhangx) [SIG Cloud Provider]
- Fix azure disk data loss issue on Windows when unmount disk (#95456, @andyzhangx) [SIG Cloud Provider and Storage]
- Fix azure file migration panic (#94853, @andyzhangx) [SIG Cloud Provider]
- Fix bug in JSON path parser where an error occurs when a range is empty (#95933, @brianpursley) [SIG API Machinery]
- Fix client-go prometheus metrics to correctly present the API path accessed in some environments. (#74363, @aanm) [SIG API Machinery]
- Fix detach azure disk issue when vm not exist (#95177, @andyzhangx) [SIG Cloud Provider]
- Fix etcd_object_counts metric reported by kube-apiserver (#94773, @tkashem) [SIG API Machinery]
- Fix incorrectly reported verbs for kube-apiserver metrics for CRD objects (#93523, @wojtek-t) [SIG API Machinery and Instrumentation]
- Fix k8s.io/apimachinery/pkg/api/meta.SetStatusCondition to update ObservedGeneration (#95961, @KnicKnic) [SIG API Machinery]
- Fix kubectl SchemaError on CRDs with schema using x-kubernetes-preserve-unknown-fields on array types. (#94888, @sttts) [SIG API Machinery]

- Fix memory leak in kube-apiserver when underlying time goes forth and back. (#96266, @chenyw1990) [SIG API Machinery]
- Fix missing csi annotations on node during parallel csinode update. (#94389, @pacoxu) [SIG Storage]
- Fix network_programming_latency metric reporting for Endpoints/EndpointSlice deletions, where we don't have correct timestamp (#95363, @wojtek-t) [SIG Network and Scalability]
- Fix paging issues when Azure API returns empty values with non-empty nextLink (#96211, @feiskyer) [SIG Cloud Provider]
- Fix pull image error from multiple ACRs using azure managed identity (#96355, @andyzhangx) [SIG Cloud Provider]
- Fix race condition on timeCache locks. (#94751, @auxten)
- Fix regression on `kubectl portforward` when TCP and UCP services were configured on the same port. (#94728, @amorenoz)
- Fix scheduler cache snapshot when a Node is deleted before its Pods (#95130, @alculquicondor) [SIG Scheduling]
- Fix the `cloudprovider_azure_api_request_duration_seconds` metric buckets to correctly capture the latency metrics. Previously, the majority of the calls would fall in the "+Inf" bucket. (#94873, @marwanad) [SIG Cloud Provider and Instrumentation]
- Fix vSphere volumes that could be erroneously attached to wrong node (#96224, @gnufied) [SIG Cloud Provider and Storage]
- Fix verb & scope reporting for kube-apiserver metrics (LIST reported instead of GET) (#95562, @wojtek-t) [SIG API Machinery and Testing]
- Fix vsphere detach failure for static PVs (#95447, @gnufied) [SIG Cloud Provider and Storage]
- Fix: azure disk resize error if source does not exist (#93011, @andyzhangx) [SIG Cloud Provider]
- Fix: detach azure disk broken on Azure Stack (#94885, @andyzhangx) [SIG Cloud Provider]
- Fix: resize Azure disk issue when it's in attached state (#96705, @andyzhangx) [SIG Cloud Provider]
- Fix: smb valid path error (#95583, @andyzhangx) [SIG Storage]
- Fix: use sensitiveOptions on Windows mount (#94126, @andyzhangx) [SIG Cloud Provider and Storage]
- Fixed a bug causing incorrect formatting of `kubectl describe ingress`. (#94985, @howardjohn) [SIG CLI and Network]

- Fixed a bug in client-go where new clients with customized `Dial`, `Proxy`, `GetCert` config may get stale HTTP transports. (#95427, @roycaihw) [SIG API Machinery]
- Fixed a bug that prevents kubectl to validate CRDs with schema using x-kubernetes-preserve-unknown-fields on object fields. (#96369, @gautierdelorme) [SIG API Machinery and Testing]
- Fixed a bug that prevents the use of ephemeral containers in the presence of a validating admission webhook. (#94685, @verb) [SIG Node and Testing]
- Fixed a bug where aggregator_unavailable_apiservice metrics were reported for deleted apiservices. (#96421, @dgrisonnet) [SIG API Machinery and Instrumentation]
- Fixed a bug where improper storage and comparison of endpoints led to excessive API traffic from the endpoints controller (#94112, @damemi) [SIG Apps, Network and Testing]
- Fixed a regression which prevented pods with `docker/default` seccomp annotations from being created in 1.19 if a PodSecurityPolicy was in place which did not allow `runtime/default` seccomp profiles. (#95985, @saschagrunert) [SIG Auth]
- Fixed bug in reflector that couldn't recover from "Too large resource version" errors with API servers 1.17.0-1.18.5 (#94316, @janeczku) [SIG API Machinery]
- Fixed bug where kubectl top pod output is not sorted when --sort-by and --containers flags are used together (#93692, @brianpursley) [SIG CLI]
- Fixed kubelet creating extra sandbox for pods with RestartPolicyOnFailure after all containers succeeded (#92614, @tnqn) [SIG Node and Testing]
- Fixes an issue proxying to ipv6 pods without specifying a port (#94834, @liggitt) [SIG API Machinery and Network]
- Fixes code generation for non-namespaced create subresources fake client test. (#96586, @Doude) [SIG API Machinery]
- Fixes high CPU usage in kubectl drain (#95260, @amandahla) [SIG CLI]
- For vSphere Cloud Provider, If VM of worker node is deleted, the node will also be deleted by node controller (#92608, @lubronzhan) [SIG Cloud Provider]
- Gracefully delete nodes when their parent scale set went missing (#95289, @bpineau) [SIG Cloud Provider]

- HTTP/2 connection health check is enabled by default in all Kubernetes clients. The feature should work out-of-the-box. If needed, users can tune the feature via the HTTP2_READ_IDLE_TIMEOUT_SECONDS and HTTP2_PING_TIMEOUT_SECONDS environment variables. The feature is disabled if HTTP2_READ_IDLE_TIMEOUT_SECONDS is set to 0. (#95981, @caesarxuchao) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Node]

- If the user specifies an invalid timeout in the request URL, the request will be aborted with an HTTP 400.

    - If the user specifies a timeout in the request URL that exceeds the maximum request deadline allowed by the apiserver, the request will be aborted with an HTTP 400. (#96061, @tkashem) [SIG API Machinery, Network and Testing]

- If we set SelectPolicy MinPolicySelect on scaleUp behavior or scaleDown behavior,Horizontal Pod Autoscaler doesn`t automatically scale the number of pods correctly (#95647, @JoshuaAndrew) [SIG Apps and Autoscaling]

- Ignore apparmor for non-linux operating systems (#93220, @wawa0210) [SIG Node and Windows]

- Ignore root user check when windows pod starts (#92355, @wawa0210) [SIG Node and Windows]

- Improve error messages related to nodePort endpoint changes conntrack entries cleanup. (#96251, @ravens) [SIG Network]

- In dual-stack clusters, kubelet will now set up both IPv4 and IPv6 iptables rules, which may fix some problems, eg with HostPorts. (#94474, @danwinship) [SIG Network and Node]

- Increase maximum IOPS of AWS EBS io1 volume to current maximum (64,000). (#90014, @jacobmarble)

- Ipvs: ensure selected scheduler kernel modules are loaded (#93040, @cmluciano) [SIG Network]

- K8s.io/apimachinery: runtime.DefaultUnstructuredConverter.FromUnstructured now handles converting integer fields to typed float values (#93250, @liggitt) [SIG API Machinery]

- Kube-proxy now trims extra spaces found in loadBalancerSourceRanges to match Service validation. (#94107, @robscott) [SIG Network]

- Kubeadm ensures "kubeadm reset" does not unmount the root "/var/lib/kubelet" directory if it is mounted by the user. (#93702, @thtanaka)

- Kubeadm now makes sure the etcd manifest is regenerated upon upgrade even when no etcd version change takes place (#94395, @rosti) [SIG Cluster Lifecycle]

- Kubeadm now warns (instead of error out) on missing "ca.key" files for root CA, front-proxy CA and etcd CA, during "kubeadm join --control-plane" if the user has provided all certificates, keys and kubeconfig files which require signing with the given CA keys. (#94988, @neolit123)
- Kubeadm: add missing "--experimental-patches" flag to "kubeadm init phase control-plane" (#95786, @Sh4d1) [SIG Cluster Lifecycle]
- Kubeadm: avoid a panic when determining if the running version of CoreDNS is supported during upgrades (#94299, @zouyee) [SIG Cluster Lifecycle]
- Kubeadm: ensure the etcd data directory is created with 0700 permissions during control-plane init and join (#94102, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: fix coredns migration should be triggered when there are newdefault configs during kubeadm upgrade (#96907, @pacoxu) [SIG Cluster Lifecycle]
- Kubeadm: fix the bug that kubeadm tries to call 'docker info' even if the CRI socket was for another CR (#94555, @SataQiu) [SIG Cluster Lifecycle]
- Kubeadm: for Docker as the container runtime, make the "kubeadm reset" command stop containers before removing them (#94586, @BedivereZero) [SIG Cluster Lifecycle]
- Kubeadm: make the kubeconfig files for the kube-controller-manager and kube-scheduler use the LocalAPIEndpoint instead of the ControlPlaneEndpoint. This makes kubeadm clusters more reseliant to version skew problems during immutable upgrades: https://kubernetes.io/docs/setup/release/version-skew-policy/#kube-controller-manager-kube-scheduler-and-cloud-controller-manager (#94398, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: relax the validation of kubeconfig server URLs. Allow the user to define custom kubeconfig server URLs without erroring out during validation of existing kubeconfig files (e.g. when using external CA mode). (#94816, @neolit123) [SIG Cluster Lifecycle]
- Kubectl: print error if users place flags before plugin name (#92343, @knight42) [SIG CLI]
- Kubelet: assume that swap is disabled when `/proc/swaps` does not exist (#93931, @SataQiu) [SIG Node]
- New Azure instance types do now have correct max data disk count information. (#94340, @ialidzhikov) [SIG Cloud Provider and Storage]
- Port mapping now allows the same `containerPort` of different containers to different `hostPort` without naming the mapping explicitly. (#94494, @SergeyKanzhelev)

- Print go stack traces at -v=4 and not -v=2 (#94663, @soltysh) [SIG CLI]
- Recreate EndpointSlices on rapid Service creation. (#94730, @robscott)
- Reduce volume name length for vsphere volumes (#96533, @gnufied) [SIG Storage]
- Remove ready file and its directory (which is created during volume SetUp) during emptyDir volume TearDown. (#95770, @jingxu97) [SIG Storage]
- Reorganized iptables rules to fix a performance issue (#95252, @tssurya) [SIG Network]
- Require feature flag CustomCPUCFSQuotaPeriod if setting a non-default cpuCFSQuotaPeriod in kubelet config. (#94687, @karan) [SIG Node]
- Resolves a regression in 1.19+ with workloads targeting deprecated beta os/arch labels getting stuck in NodeAffinity status on node startup. (#96810, @liggitt) [SIG Node]
- Resolves non-deterministic behavior of the garbage collection controller when ownerReferences with incorrect data are encountered. Events with a reason of `OwnerRefInvalidNamespace` are recorded when namespace mismatches between child and owner objects are detected. The kubectl-check-ownerreferences tool can be run prior to upgrading to locate existing objects with invalid ownerReferences.
    - A namespaced object with an ownerReference referencing a uid of a namespaced kind which does not exist in the same namespace is now consistently treated as though that owner does not exist, and the child object is deleted.
    - A cluster-scoped object with an ownerReference referencing a uid of a namespaced kind is now consistently treated as though that owner is not resolvable, and the child object is ignored by the garbage collector. (#92743, @liggitt) [SIG API Machinery, Apps and Testing]
- Skip [k8s.io/kubernetes@v1.19.0/test/e2e/storage/testsuites/base.go:162]: Driver azure-disk doesn't support snapshot type DynamicSnapshot -- skipping skip [k8s.io/kubernetes@v1.19.0/test/e2e/storage/testsuites/base.go:185]: Driver azure-disk doesn't support ntfs -- skipping (#96144, @qinpingli) [SIG Storage and Testing]
- StatefulSet Controller now waits for PersistentVolumeClaim deletion before creating pods. (#93457, @ymmt2005)
- StreamWatcher now calls HandleCrash at appropriate sequence. (#93108, @lixiaobing1)

- Support the node label `node.kubernetes.io/exclude-from-external-load-balancers` (#95542, @nilo19) [SIG Cloud Provider]
- The AWS network load balancer attributes can now be specified during service creation (#95247, @kishorj) [SIG Cloud Provider]
- The `/debug/api_priority_and_fairness/dump_requests` path at an apiserver will no longer return a phantom line for each exempt priority level. (#93406, @MikeSpreitzer) [SIG API Machinery]
- The kube-apiserver will no longer serve APIs that should have been deleted in GA non-alpha levels. Alpha levels will continue to serve the removed APIs so that CI doesn't immediately break. (#96525, @deads2k) [SIG API Machinery]
- The kubelet recognizes the --containerd-namespace flag to configure the namespace used by cadvisor. (#87054, @changyaowei) [SIG Node]
- Unhealthy pods covered by PDBs can be successfully evicted if enough healthy pods are available. (#94381, @michaelgugino) [SIG Apps]
- Update Calico to v3.15.2 (#94241, @lmm) [SIG Cloud Provider]
- Update default etcd server version to 3.4.13 (#94287, @jingyih) [SIG API Machinery, Cloud Provider, Cluster Lifecycle and Testing]
- Update max azure data disk count map (#96308, @andyzhangx) [SIG Cloud Provider and Storage]
- Update the PIP when it is not in the Succeeded provisioning state during the LB update. (#95748, @nilo19) [SIG Cloud Provider]
- Update the frontend IP config when the service's `pipName` annotation is changed (#95813, @nilo19) [SIG Cloud Provider]
- Update the route table tag in the route reconcile loop (#96545, @nilo19) [SIG Cloud Provider]
- Use NLB Subnet CIDRs instead of VPC CIDRs in Health Check SG Rules (#93515, @t0rr3sp3dr0) [SIG Cloud Provider]
- Users will see increase in time for deletion of pods and also guarantee that removal of pod from api server would mean deletion of all the resources from container runtime. (#92817, @kmala) [SIG Node]
- Very large patches may now be specified to `kubectl patch` with the `--patch-file` flag instead of including them directly on the command line. The `--patch` and `--patch-file` flags are mutually exclusive. (#93548, @smarterclayton) [SIG CLI]
- Volume binding: report UnschedulableAndUnresolvable status instead of an error when bound PVs not found (#95541, @cofyc) [SIG Apps, Scheduling and Storage]

- Warn instead of fail when creating Roles and ClusterRoles with custom verbs via kubectl (#92492, @eddiezane) [SIG CLI]
- When creating a PVC with the volume.beta.kubernetes.io/storage-provisioner annotation already set, the PV controller might have incorrectly deleted the newly provisioned PV instead of binding it to the PVC, depending on timing and system load. (#95909, @pohly) [SIG Apps and Storage]
- [kubectl] Fail when local source file doesn't exist (#90333, @bamarni) [SIG CLI]

Other (Cleanup or Flake)

- **Additional documentation e.g., KEPs (Kubernetes Enhancement Proposals), usage docs, etc.**:

  ([#96443](https://github.com/kubernetes/kubernetes/pull/96443), [@alaypatel07](https://github.com/alaypatel07)) [SIG Apps]
- --redirect-container-streaming is no longer functional. The flag will be removed in v1.22 (#95935, @tallclair) [SIG Node]
- A new metric `requestAbortsTotal` has been introduced that counts aborted requests for each `group`, `version`, `verb`, `resource`, `subresource` and `scope`. (#95002, @p0lyn0mial) [SIG API Machinery, Cloud Provider, Instrumentation and Scheduling]
- API priority and fairness metrics use snake_case in label names (#96236, @adtac) [SIG API Machinery, Cluster Lifecycle, Instrumentation and Testing]
- Add fine grained debugging to intra-pod conformance test to troubleshoot networking issues for potentially unhealthy nodes when running conformance or sonobuoy tests. (#93837, @jayunit100)
- Add the following metrics:
  - network_plugin_operations_total
  - network_plugin_operations_errors_total (#93066, @AnishShah)
- Adds a bootstrapping ClusterRole, ClusterRoleBinding and group for /metrics, /livez/, /readyz/, & /healthz/- endpoints. (#93311, @logicalhan) [SIG API Machinery, Auth, Cloud Provider and Instrumentation]
- AdmissionReview objects sent for the creation of Namespace API objects now populate the `namespace` attribute consistently (previously the `namespace` attribute was empty for Namespace creation via POST requests, and populated for Namespace creation via server-side-apply PATCH requests) (#95012, @nodo) [SIG API Machinery and Testing]
- Applies translations on all command descriptions (#95439, @HerrNaN) [SIG CLI]

Base-images: Update to debian-iptables:buster-v1.3.0

-
  - Uses iptables 1.8.5
  - base-images: Update to debian-base:buster-v1.2.0
  - cluster/images/etcd: Build etcd:3.4.13-1 image
    - Uses debian-base:buster-v1.2.0 (#94733, @justaugustus) [SIG API Machinery, Release and Testing]

- Changed: default "Accept-Encoding" header removed from HTTP probes. See https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#http-probes (#96127, @fonsecas72) [SIG Network and Node]

- Client-go header logging (at verbosity levels >= 9) now masks `Authorization` header contents (#95316, @sfowl) [SIG API Machinery]

- Decrease warning message frequency on setting volume ownership for configmap/secret. (#92878, @jvanz)

- Enhance log information of verifyRunAsNonRoot, add pod, container information (#94911, @wawa0210) [SIG Node]

- Fix func name NewCreateCreateDeploymentOptions (#91931, @lixiaobing1) [SIG CLI]

- Fix kubelet to properly log when a container is started. Previously, kubelet may log that container is dead and was restarted when it was actually started for the first time. This behavior only happened on pods with initContainers and regular containers. (#91469, @rata)

- Fixes the message about no auth for metrics in scheduler. (#94035, @zhouya0) [SIG Scheduling]

- Generators for services are removed from kubectl (#95256, @Git-Jiro) [SIG CLI]

- Introduce kubectl-convert plugin. (#96190, @soltysh) [SIG CLI and Testing]

- Kube-scheduler now logs processed component config at startup (#96426, @damemi) [SIG Scheduling]

- Kubeadm: Separate argument key/value in log msg (#94016, @mrueg) [SIG Cluster Lifecycle]

- Kubeadm: remove the CoreDNS check for known image digests when applying the addon (#94506, @neolit123) [SIG Cluster Lifecycle]

- Kubeadm: update the default pause image version to 1.4.0 on Windows. With this update the image supports Windows versions 1809 (2019LTS), 1903, 1909, 2004 (#95419, @jsturtevant) [SIG Cluster Lifecycle and Windows]

- Kubectl: the `generator` flag of `kubectl autoscale` has been deprecated and has no effect, it will be removed in a feature release (#92998, @SataQiu) [SIG CLI]
- Lock ExternalPolicyForExternalIP to default, this feature gate will be removed in 1.22. (#94581, @knabben) [SIG Network]
- Mask ceph RBD adminSecrets in logs when logLevel >= 4. (#95245, @sfowl)
- Remove offensive words from kubectl cluster-info command. (#95202, @rikatz)
- Remove support for "ci/k8s-master" version label in kubeadm, use "ci/latest" instead. See kubernetes/test-infra#18517. (#93626, @vikkyomkar)
- Remove the dependency of csi-translation-lib module on apiserver/cloud-provider/controller-manager (#95543, @wawa0210) [SIG Release]
- Scheduler framework interface moved from pkg/scheduler/framework/v1alpha to pkg/scheduler/framework (#95069, @farah) [SIG Scheduling, Storage and Testing]
- Service.beta.kubernetes.io/azure-load-balancer-disable-tcp-reset is removed. All Standard load balancers will always enable tcp resets. (#94297, @MarcPow) [SIG Cloud Provider]
- Stop propagating SelfLink (deprecated in 1.16) in kube-apiserver (#94397, @wojtek-t) [SIG API Machinery and Testing]
- Strip unnecessary security contexts on Windows (#93475, @ravisantoshgudimetla) [SIG Node, Testing and Windows]
- To ensure the code be strong, add unit test for GetAddressAndDialer (#93180, @FreeZhang61) [SIG Node]
- UDP and SCTP protocols can left stale connections that need to be cleared to avoid services disruption, but they can cause problems that are hard to debug. Kubernetes components using a loglevel greater or equal than 4 will log the conntrack operations and its output, to show the entries that were deleted. (#95694, @aojea) [SIG Network]
- Update CNI plugins to v0.8.7 (#94367, @justaugustus) [SIG Cloud Provider, Network, Node, Release and Testing]
- Update cri-tools to v1.19.0 (#94307, @xmudrii) [SIG Cloud Provider]
- Update etcd client side to v3.4.13 (#94259, @jingyih) [SIG API Machinery and Cloud Provider]

- Users will now be able to configure all supported values for AWS NLB health check interval and thresholds for new resources. (#96312, @kishorj) [SIG Cloud Provider]
- V1helpers.MatchNodeSelectorTerms now accepts just a Node and a list of Terms (#95871, @damemi) [SIG Apps, Scheduling and Storage]
- Vsphere: improve logging message on node cache refresh event (#95236, @andrewsykim) [SIG Cloud Provider]
- `MatchNodeSelectorTerms` function moved to `k8s.io/component-helpers` (#95531, @damemi) [SIG Apps, Scheduling and Storage]
- `kubectl api-resources` now prints the API version (as 'API group/version', same as output of `kubectl api-versions`). The column APIGROUP is now APIVERSION (#95253, @sallyom) [SIG CLI]
- `kubectl get ingress` now prefers the `networking.k8s.io/v1` over `extensions/v1beta1` (deprecated since v1.14). To explicitly request the deprecated version, use `kubectl get ingress.v1beta1.extensions`. (#94309, @liggitt) [SIG API Machinery and CLI]

Dependencies

Added

- cloud.google.com/go/firestore: v1.1.0
- github.com/Azure/go-autorest: v14.2.0+incompatible
- github.com/armon/go-metrics: f0300d1
- github.com/armon/go-radix: 7fddfc3
- github.com/bketelsen/crypt: 5cbc8cc
- github.com/form3tech-oss/jwt-go: v3.2.2+incompatible
- github.com/fvbommel/sortorder: v1.0.1
- github.com/hashicorp/consul/api: v1.1.0
- github.com/hashicorp/consul/sdk: v0.1.1
- github.com/hashicorp/errwrap: v1.0.0
- github.com/hashicorp/go-cleanhttp: v0.5.1
- github.com/hashicorp/go-immutable-radix: v1.0.0
- github.com/hashicorp/go-msgpack: v0.5.3
- github.com/hashicorp/go-multierror: v1.0.0
- github.com/hashicorp/go-rootcerts: v1.0.0
- github.com/hashicorp/go-sockaddr: v1.0.0
- github.com/hashicorp/go-uuid: v1.0.1
- github.com/hashicorp/go.net: v0.0.1
- github.com/hashicorp/logutils: v1.0.0
- github.com/hashicorp/mdns: v1.0.0
- github.com/hashicorp/memberlist: v0.1.3
- github.com/hashicorp/serf: v0.8.2
- github.com/jmespath/go-jmespath/internal/testify: v1.5.1
- github.com/mitchellh/cli: v1.0.0
- github.com/mitchellh/go-testing-interface: v1.0.0
- github.com/mitchellh/gox: v0.4.0

- github.com/mitchellh/iochan: v1.0.0
- github.com/pascaldekloe/goe: 57f6aae
- github.com/posener/complete: v1.1.1
- github.com/ryanuber/columnize: 9b3edd6
- github.com/sean-/seed: e2103e2
- github.com/subosito/gotenv: v1.2.0
- github.com/willf/bitset: d5bec33
- gopkg.in/ini.v1: v1.51.0
- gopkg.in/yaml.v3: 9f266ea
- rsc.io/quote/v3: v3.1.0
- rsc.io/sampler: v1.3.0

Changed

- cloud.google.com/go/bigquery: v1.0.1 â†’ v1.4.0
- cloud.google.com/go/datastore: v1.0.0 â†’ v1.1.0
- cloud.google.com/go/pubsub: v1.0.1 â†’ v1.2.0
- cloud.google.com/go/storage: v1.0.0 â†’ v1.6.0
- cloud.google.com/go: v0.51.0 â†’ v0.54.0
- github.com/Azure/go-autorest/autorest/adal: v0.8.2 â†’ v0.9.5
- github.com/Azure/go-autorest/autorest/date: v0.2.0 â†’ v0.3.0
- github.com/Azure/go-autorest/autorest/mocks: v0.3.0 â†’ v0.4.1
- github.com/Azure/go-autorest/autorest: v0.9.6 â†’ v0.11.1
- github.com/Azure/go-autorest/logger: v0.1.0 â†’ v0.2.0
- github.com/Azure/go-autorest/tracing: v0.5.0 â†’ v0.6.0
- github.com/Microsoft/go-winio: fc70bd9 â†’ v0.4.15
- github.com/aws/aws-sdk-go: v1.28.2 â†’ v1.35.24
- github.com/blang/semver: v3.5.0+incompatible â†’ v3.5.1+incompatible
- github.com/checkpoint-restore/go-criu/v4: v4.0.2 â†’ v4.1.0
- github.com/containerd/containerd: v1.3.3 â†’ v1.4.1
- github.com/containerd/ttrpc: v1.0.0 â†’ v1.0.2
- github.com/containerd/typeurl: v1.0.0 â†’ v1.0.1
- github.com/coreos/etcd: v3.3.10+incompatible â†’ v3.3.13+incompatible
- github.com/docker/docker: aa6a989 â†’ bd33bbf
- github.com/go-gl/glfw/v3.3/glfw: 12ad95a â†’ 6f7a984
- github.com/golang/groupcache: 215e871 â†’ 8c9f03a
- github.com/golang/mock: v1.3.1 â†’ v1.4.1
- github.com/golang/protobuf: v1.4.2 â†’ v1.4.3
- github.com/google/cadvisor: v0.37.0 â†’ v0.38.5
- github.com/google/go-cmp: v0.4.0 â†’ v0.5.2
- github.com/google/pprof: d4f498a â†’ 1ebb73c
- github.com/google/uuid: v1.1.1 â†’ v1.1.2
- github.com/gorilla/mux: v1.7.3 â†’ v1.8.0
- github.com/gorilla/websocket: v1.4.0 â†’ v1.4.2
- github.com/jmespath/go-jmespath: c2b33e8 â†’ v0.4.0
- github.com/karrick/godirwalk: v1.7.5 â†’ v1.16.1
- github.com/opencontainers/go-digest: v1.0.0-rc1 â†’ v1.0.0
- github.com/opencontainers/runc: 819fcc6 â†’ v1.0.0-rc92
- github.com/opencontainers/runtime-spec: 237cc4f â†’ 4d89ac9
- github.com/opencontainers/selinux: v1.5.2 â†’ v1.6.0
- github.com/prometheus/procfs: v0.1.3 â†’ v0.2.0

- github.com/quobyte/api: v0.1.2 â†’ v0.1.8
- github.com/spf13/cobra: v1.0.0 â†’ v1.1.1
- github.com/spf13/viper: v1.4.0 â†’ v1.7.0
- github.com/storageos/go-api: 343b3ef â†’ v2.2.0+incompatible
- github.com/stretchr/testify: v1.4.0 â†’ v1.6.1
- github.com/vishvananda/netns: 52d707b â†’ db3c7e5
- go.etcd.io/etcd: 17cef6e â†’ dd1b699
- go.opencensus.io: v0.22.2 â†’ v0.22.3
- golang.org/x/crypto: 75b2880 â†’ 7f63de1
- golang.org/x/exp: da58074 â†’ 6cc2880
- golang.org/x/lint: fdd1cda â†’ 738671d
- golang.org/x/net: ab34263 â†’ 69a7880
- golang.org/x/oauth2: 858c2ad â†’ bf48bf1
- golang.org/x/sys: ed371f2 â†’ 5cba982
- golang.org/x/text: v0.3.3 â†’ v0.3.4
- golang.org/x/time: 555d28b â†’ 3af7569
- golang.org/x/xerrors: 9bdfabe â†’ 5ec99f8
- google.golang.org/api: v0.15.1 â†’ v0.20.0
- google.golang.org/genproto: cb27e3a â†’ 8816d57
- google.golang.org/grpc: v1.27.0 â†’ v1.27.1
- google.golang.org/protobuf: v1.24.0 â†’ v1.25.0
- honnef.co/go/tools: v0.0.1-2019.2.3 â†’ v0.0.1-2020.1.3
- k8s.io/gengo: 8167cfd â†’ 83324d8
- k8s.io/klog/v2: v2.2.0 â†’ v2.4.0
- k8s.io/kube-openapi: 6aeccd4 â†’ d219536
- k8s.io/system-validators: v1.1.2 â†’ v1.2.0
- k8s.io/utils: d5654de â†’ 67b214c
- sigs.k8s.io/apiserver-network-proxy/konnectivity-client: v0.0.9 â†’ v0.0.14
- sigs.k8s.io/structured-merge-diff/v4: v4.0.1 â†’ v4.0.2

Removed

- github.com/armon/consul-api: eb2c6b5
- github.com/go-ini/ini: v1.9.0
- github.com/ugorji/go: v1.1.4
- github.com/xlab/handysort: fb3537e
- github.com/xordataexchange/crypt: b2862e3
- vbom.ml/util: db5cfe1

Dependencies

Added

- cloud.google.com/go/firestore: v1.1.0
- github.com/Azure/go-autorest: v14.2.0+incompatible
- github.com/armon/go-metrics: f0300d1
- github.com/armon/go-radix: 7fddfc3
- github.com/bketelsen/crypt: 5cbc8cc
- github.com/form3tech-oss/jwt-go: v3.2.2+incompatible
- github.com/fvbommel/sortorder: v1.0.1

- github.com/hashicorp/consul/api: v1.1.0
- github.com/hashicorp/consul/sdk: v0.1.1
- github.com/hashicorp/errwrap: v1.0.0
- github.com/hashicorp/go-cleanhttp: v0.5.1
- github.com/hashicorp/go-immutable-radix: v1.0.0
- github.com/hashicorp/go-msgpack: v0.5.3
- github.com/hashicorp/go-multierror: v1.0.0
- github.com/hashicorp/go-rootcerts: v1.0.0
- github.com/hashicorp/go-sockaddr: v1.0.0
- github.com/hashicorp/go-uuid: v1.0.1
- github.com/hashicorp/go.net: v0.0.1
- github.com/hashicorp/logutils: v1.0.0
- github.com/hashicorp/mdns: v1.0.0
- github.com/hashicorp/memberlist: v0.1.3
- github.com/hashicorp/serf: v0.8.2
- github.com/jmespath/go-jmespath/internal/testify: v1.5.1
- github.com/mitchellh/cli: v1.0.0
- github.com/mitchellh/go-testing-interface: v1.0.0
- github.com/mitchellh/gox: v0.4.0
- github.com/mitchellh/iochan: v1.0.0
- github.com/pascaldekloe/goe: 57f6aae
- github.com/posener/complete: v1.1.1
- github.com/ryanuber/columnize: 9b3edd6
- github.com/sean-/seed: e2103e2
- github.com/subosito/gotenv: v1.2.0
- github.com/willf/bitset: d5bec33
- gopkg.in/ini.v1: v1.51.0
- gopkg.in/yaml.v3: 9f266ea
- rsc.io/quote/v3: v3.1.0
- rsc.io/sampler: v1.3.0

Changed

- cloud.google.com/go/bigquery: v1.0.1 â†’ v1.4.0
- cloud.google.com/go/datastore: v1.0.0 â†’ v1.1.0
- cloud.google.com/go/pubsub: v1.0.1 â†’ v1.2.0
- cloud.google.com/go/storage: v1.0.0 â†’ v1.6.0
- cloud.google.com/go: v0.51.0 â†’ v0.54.0
- github.com/Azure/go-autorest/autorest/adal: v0.8.2 â†’ v0.9.5
- github.com/Azure/go-autorest/autorest/date: v0.2.0 â†’ v0.3.0
- github.com/Azure/go-autorest/autorest/mocks: v0.3.0 â†’ v0.4.1
- github.com/Azure/go-autorest/autorest: v0.9.6 â†’ v0.11.1
- github.com/Azure/go-autorest/logger: v0.1.0 â†’ v0.2.0
- github.com/Azure/go-autorest/tracing: v0.5.0 â†’ v0.6.0
- github.com/Microsoft/go-winio: fc70bd9 â†’ v0.4.15
- github.com/aws/aws-sdk-go: v1.28.2 â†’ v1.35.24
- github.com/blang/semver: v3.5.0+incompatible â†’ v3.5.1+incompatible
- github.com/checkpoint-restore/go-criu/v4: v4.0.2 â†’ v4.1.0
- github.com/containerd/containerd: v1.3.3 â†’ v1.4.1
- github.com/containerd/ttrpc: v1.0.0 â†’ v1.0.2
- github.com/containerd/typeurl: v1.0.0 â†’ v1.0.1

- github.com/coreos/etcd: v3.3.10+incompatible â†’ v3.3.13+incompatible
- github.com/docker/docker: aa6a989 â†’ bd33bbf
- github.com/go-gl/glfw/v3.3/glfw: 12ad95a â†’ 6f7a984
- github.com/golang/groupcache: 215e871 â†’ 8c9f03a
- github.com/golang/mock: v1.3.1 â†’ v1.4.1
- github.com/golang/protobuf: v1.4.2 â†’ v1.4.3
- github.com/google/cadvisor: v0.37.0 â†’ v0.38.5
- github.com/google/go-cmp: v0.4.0 â†’ v0.5.2
- github.com/google/pprof: d4f498a â†’ 1ebb73c
- github.com/google/uuid: v1.1.1 â†’ v1.1.2
- github.com/gorilla/mux: v1.7.3 â†’ v1.8.0
- github.com/gorilla/websocket: v1.4.0 â†’ v1.4.2
- github.com/jmespath/go-jmespath: c2b33e8 â†’ v0.4.0
- github.com/karrick/godirwalk: v1.7.5 â†’ v1.16.1
- github.com/opencontainers/go-digest: v1.0.0-rc1 â†’ v1.0.0
- github.com/opencontainers/runc: 819fcc6 â†’ v1.0.0-rc92
- github.com/opencontainers/runtime-spec: 237cc4f â†’ 4d89ac9
- github.com/opencontainers/selinux: v1.5.2 â†’ v1.6.0
- github.com/prometheus/procfs: v0.1.3 â†’ v0.2.0
- github.com/quobyte/api: v0.1.2 â†’ v0.1.8
- github.com/spf13/cobra: v1.0.0 â†’ v1.1.1
- github.com/spf13/viper: v1.4.0 â†’ v1.7.0
- github.com/storageos/go-api: 343b3ef â†’ v2.2.0+incompatible
- github.com/stretchr/testify: v1.4.0 â†’ v1.6.1
- github.com/vishvananda/netns: 52d707b â†’ db3c7e5
- go.etcd.io/etcd: 17cef6e â†’ dd1b699
- go.opencensus.io: v0.22.2 â†’ v0.22.3
- golang.org/x/crypto: 75b2880 â†’ 7f63de1
- golang.org/x/exp: da58074 â†’ 6cc2880
- golang.org/x/lint: fdd1cda â†’ 738671d
- golang.org/x/net: ab34263 â†’ 69a7880
- golang.org/x/oauth2: 858c2ad â†’ bf48bf1
- golang.org/x/sys: ed371f2 â†’ 5cba982
- golang.org/x/text: v0.3.3 â†’ v0.3.4
- golang.org/x/time: 555d28b â†’ 3af7569
- golang.org/x/xerrors: 9bdfabe â†’ 5ec99f8
- google.golang.org/api: v0.15.1 â†’ v0.20.0
- google.golang.org/genproto: cb27e3a â†’ 8816d57
- google.golang.org/grpc: v1.27.0 â†’ v1.27.1
- google.golang.org/protobuf: v1.24.0 â†’ v1.25.0
- honnef.co/go/tools: v0.0.1-2019.2.3 â†’ v0.0.1-2020.1.3
- k8s.io/gengo: 8167cfd â†’ 83324d8
- k8s.io/klog/v2: v2.2.0 â†’ v2.4.0
- k8s.io/kube-openapi: 6aeccd4 â†’ d219536
- k8s.io/system-validators: v1.1.2 â†’ v1.2.0
- k8s.io/utils: d5654de â†’ 67b214c
- sigs.k8s.io/apiserver-network-proxy/konnectivity-client: v0.0.9 â†’ v0.0.14
- sigs.k8s.io/structured-merge-diff/v4: v4.0.1 â†’ v4.0.2

Removed

- github.com/armon/consul-api: eb2c6b5
- github.com/go-ini/ini: v1.9.0
- github.com/ugorji/go: v1.1.4
- github.com/xlab/handysort: fb3537e
- github.com/xordataexchange/crypt: b2862e3
- vbom.ml/util: db5cfe1

v1.20.0-rc.0

Downloads for v1.20.0-rc.0

Source Code

filename	sha512 hash
kubernetes.tar.gz	acfee8658831f9503fccda0904798405434f17be7064a361a9f34c6ed04
kubernetes-src.tar.gz	9d962f8845e1fa221649cf0c0e178f0f03808486c49ea15ab5ec67861ec

Client binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	062b57f1a450fe01d6184f104d81d376bdf5720010412821e315fd9b
kubernetes-client-linux-386.tar.gz	86e96d2c2046c5e62e02bef30a6643f25e01f1b3eba256cab7dd6125
kubernetes-client-linux-amd64.tar.gz	619d3afb9ce902368390e71633396010e88e87c5fd848e3adc71571c
kubernetes-client-linux-arm.tar.gz	60965150a60ab3d05a248339786e0c7da4b89a04539c3719737b13c
kubernetes-client-linux-arm64.tar.gz	688e064f4ef6a17189dbb5af468c279b9de35e215c40500fb97b1d46
kubernetes-client-linux-ppc64le.tar.gz	47b8abc02b42b3b1de67da184921b5801d7e3cb09befac840c85913
kubernetes-client-linux-s390x.tar.gz	971b41d3169f30e6c412e0254c180636abb7ccc8dcee6641b0e9877
kubernetes-client-windows-386.tar.gz	2d34e8387e31531d9aca5655f2f0d18e75b01825dc1c39b7beb73a7b
kubernetes-client-windows-amd64.tar.gz	f909640f4140693bb871936f10a40e79b43502105d0adb318b35bb7a

Server binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	0ea4458ae34108c633b4d48f1f128c6274dbc82b613492e78b3e0a2f656a
kubernetes-server-linux-arm.tar.gz	aef6a4d457faa29936603370f29a8523bb274211c3cb5101bd31aaf469c91
kubernetes-server-linux-arm64.tar.gz	4829f473e9d60f9929ad17c70fdc2b6b6509ed75418be0b23a75b2858094
kubernetes-server-linux-ppc64le.tar.gz	9ab0790d382a3e28df1c013762c09da0085449cfd09d176d80be932806c2
kubernetes-server-linux-s390x.tar.gz	98670b587e299856dd9821b7517a35f9a65835b915b153de08b66c54d82

Node binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	699e9c8d1837198312eade8eb6fec390f6a2fea9e08207d2f58e8bb6e3e79
kubernetes-node-linux-arm.tar.gz	f3b5eab0669490e3cd7e802693daf3555d08323dfff6e73a881fce00fed469
kubernetes-node-linux-arm64.tar.gz	e5012f77363561a609aaf791baaa17d09009819c4085a57132e5feb53662
kubernetes-node-linux-ppc64le.tar.gz	2a6d6501620b1a9838dff05c66a40260cc22154a28027813346eb16e18c3
kubernetes-node-linux-s390x.tar.gz	5eca02777519e31428a1e5842fe540b813fb8c929c341bbc71dcfd60d98d
kubernetes-node-windows-amd64.tar.gz	8ace02e7623dff894e863a2e0fa7dfb916368431d1723170713fe82e334c0

Changelog since v1.20.0-beta.2

Changes by Kind

Feature

- Kubernetes is now built using go1.15.5
  - build: Update to k/repo-infra@v0.1.2 (supports go1.15.5) (#95776, @justaugustus) [SIG Cloud Provider, Instrumentation, Release and Testing]

Failing Test

- Resolves an issue running Ingress conformance tests on clusters which use finalizers on Ingress objects to manage releasing load balancer resources (#96742, @spencerhance) [SIG Network and Testing]
- The Conformance test "validates that there is no conflict between pods with same hostPort but different hostIP and protocol" now validates the connectivity to each hostPort, in addition to the functionality. (#96627, @aojea) [SIG Scheduling and Testing]

Bug or Regression

- Bump node-problem-detector version to v0.8.5 to fix OOM detection in with Linux kernels 5.1+ (#96716, @tosi3k) [SIG Cloud Provider, Scalability and Testing]
- Changes to timeout parameter handling in 1.20.0-beta.2 have been reverted to avoid breaking backwards compatibility with existing clients. (#96727, @liggitt) [SIG API Machinery and Testing]
- Duplicate owner reference entries in create/update/patch requests now get deduplicated by the API server. The client sending the request now receives a warning header in the API response. Clients should stop sending requests with duplicate owner references. The API server may reject such requests as early as 1.24. (#96185, @roycaihw) [SIG API Machinery and Testing]
- Fix: resize Azure disk issue when it's in attached state (#96705, @andyzhangx) [SIG Cloud Provider]
- Fixed a bug where aggregator_unavailable_apiservice metrics were reported for deleted apiservices. (#96421, @dgrisonnet) [SIG API Machinery and Instrumentation]
- Fixes code generation for non-namespaced create subresources fake client test. (#96586, @Doude) [SIG API Machinery]
- HTTP/2 connection health check is enabled by default in all Kubernetes clients. The feature should work out-of-the-box. If needed, users can tune the feature via the HTTP2_READ_IDLE_TIMEOUT_SECONDS and HTTP2_PING_TIMEOUT_SECONDS environment variables. The feature is disabled if HTTP2_READ_IDLE_TIMEOUT_SECONDS is set to 0. (#95981, @caesarxuchao) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Node]

- Kubeadm: fix coredns migration should be triggered when there are newdefault configs during kubeadm upgrade (#96907, @pacoxu) [SIG Cluster Lifecycle]
- Reduce volume name length for vsphere volumes (#96533, @gnufied) [SIG Storage]
- Resolves a regression in 1.19+ with workloads targeting deprecated beta os/arch labels getting stuck in NodeAffinity status on node startup. (#96810, @liggitt) [SIG Node]

Dependencies

Added

Nothing has changed.

Changed

- github.com/google/cadvisor: v0.38.4 â†’ v0.38.5

Removed

Nothing has changed.

v1.20.0-beta.2

Downloads for v1.20.0-beta.2

Source Code

filename	sha512 hash
kubernetes.tar.gz	fe769280aa623802a949b6a35fbddadbba1d6f9933a54132a35625683
kubernetes-src.tar.gz	ce1c8d97c52e5189af335d673bd7e99c564816f6adebf249838f7e3f0e

Client binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	d6c14bd0f6702f4bbdf14a6abdfa4e5936de5b4efee38aa86c2bd7272
kubernetes-client-linux-386.tar.gz	b923c44cb0acb91a8f6fd442c2168aa6166c848f5d037ce50a7cb115
kubernetes-client-linux-amd64.tar.gz	8cae14146a9034dcd4e9d69d5d700f195a77aac35f629a148960ae02
kubernetes-client-linux-arm.tar.gz	1f54e5262a0432945ead57fcb924e6bfedd9ea76db1dd9ebd946787a

filename	sha512 hash
kubernetes-client-linux-arm64.tar.gz	31cf79c01e4878a231b4881fe3ed5ef790bd5fb5419388438d3f8c6a2
kubernetes-client-linux-ppc64le.tar.gz	2527948c40be2e16724d939316ad5363f15aa22ebf42d59359d8b6f7
kubernetes-client-linux-s390x.tar.gz	b777ad764b3a46651ecb0846e5b7f860bb2c1c4bd4d0fcc468c6cffb
kubernetes-client-windows-386.tar.gz	8a2f58aaab01be9fe298e4d01456536047cbdd39a37d3e325c1f69ce
kubernetes-client-windows-amd64.tar.gz	2f69cda177a178df149f5de66b7dba7f5ce14c1ffeb7c8d7dc4130c701

Server binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	3ecaac0213d369eab691ac55376821a80df5013cb12e1263f18d1c236a9e
kubernetes-server-linux-arm.tar.gz	580030b57ff207e177208fec0801a43389cae10cc2c9306327d354e7be6a0
kubernetes-server-linux-arm64.tar.gz	3e3286bd54671549fbef0dfdaaf1da99bc5c3efb32cc8d1e1985d9926520c
kubernetes-server-linux-ppc64le.tar.gz	9fa051e7e97648e97e26b09ab6d26be247b41b1a5938d2189204c9e6688
kubernetes-server-linux-s390x.tar.gz	fa85d432eff586f30975c95664ac130b9f5ae02dc52b97613ed7a41324496

Node binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	86e631f95fe670b467ead2b88d34e0364eaa275935af433d27cc378d82dc
kubernetes-node-linux-arm.tar.gz	a8754ff58a0e902397056b8615ab49af07aca347ba7cc4a812c238e38122
kubernetes-node-linux-arm64.tar.gz	28d727d7d08e2c856c9b4a574ef2dbf9e37236a0555f7ec5258b4284fa058
kubernetes-node-linux-ppc64le.tar.gz	a1283449f1a0b155c11449275e9371add544d0bdd4609d6dc737ed5f7dd2

filename	sha512 hash
kubernetes-node-linux-s390x.tar.gz	5806028ba15a6a9c54a34f90117bc3181428dbb0e7ced30874c9f4a953ea
kubernetes-node-windows-amd64.tar.gz	d5327e3b7916c78777b9b69ba0f3758c3a8645c67af80114a0ae52babd7a

Changelog since v1.20.0-beta.1

Urgent Upgrade Notes

(No, really, you MUST read this before you upgrade)

- A bug was fixed in kubelet where exec probe timeouts were not respected. Ensure that pods relying on this behavior are updated to correctly handle probe timeouts.

This change in behavior may be unexpected for some clusters and can be disabled by turning off the ExecProbeTimeout feature gate. This gate will be locked and removed in future releases so that exec probe timeouts are always respected. (#94115, @andrewsykim) [SIG Node and Testing]

- For CSI drivers, kubelet no longer creates the target_path for NodePublishVolume in accordance with the CSI spec. Kubelet also no longer checks if staging and target paths are mounts or corrupted. CSI drivers need to be idempotent and do any necessary mount verification. (#88759, @andyzhangx) [SIG Storage]

- Kubeadm:

- The label applied to control-plane nodes "node-role.kubernetes.io/master" is now deprecated and will be removed in a future release after a GA deprecation period.

- Introduce a new label "node-role.kubernetes.io/control-plane" that will be applied in parallel to "node-role.kubernetes.io/master" until the removal of the "node-role.kubernetes.io/master" label.

- Make "kubeadm upgrade apply" add the "node-role.kubernetes.io/control-plane" label on existing nodes that only have the "node-role.kubernetes.io/master" label during upgrade.

- Please adapt your tooling built on top of kubeadm to use the "node-role.kubernetes.io/control-plane" label.

- The taint applied to control-plane nodes "node-role.kubernetes.io/master:NoSchedule" is now deprecated and will be removed in a future release after a GA deprecation period.

- Apply toleration for a new, future taint "node-role.kubernetes.io/control-plane:NoSchedule" to the kubeadm CoreDNS / kube-dns managed manifests. Note that this taint is not yet applied to kubeadm control-plane nodes.
- Please adapt your workloads to tolerate the same future taint preemptively.

For more details see: http://git.k8s.io/enhancements/keps/sig-cluster-lifecycle/kubeadm/2067-rename-master-label-taint/README.md (#95382, @neolit123) [SIG Cluster Lifecycle]

Changes by Kind

Deprecation

- Docker support in the kubelet is now deprecated and will be removed in a future release. The kubelet uses a module called "dockershim" which implements CRI support for Docker and it has seen maintenance issues in the Kubernetes community. We encourage you to evaluate moving to a container runtime that is a full-fledged implementation of CRI (v1alpha1 or v1 compliant) as they become available. (#94624, @dims) [SIG Node]
- Kubectl: deprecate --delete-local-data (#95076, @dougsland) [SIG CLI, Cloud Provider and Scalability]

API Change

- API priority and fairness graduated to beta 1.19 servers with APF turned on should not be run in a multi-server cluster with 1.20+ servers. (#96527, @adtac) [SIG API Machinery and Testing]
- Add LoadBalancerIPMode feature gate (#92312, @Sh4d1) [SIG Apps, CLI, Cloud Provider and Network]
- Add WindowsContainerResources and Annotations to CRI-API UpdateContainerResourcesRequest (#95741, @katiewasnothere) [SIG Node]
- Add a 'serving' and `terminating` condition to the EndpointSlice API.

  `serving` tracks the readiness of endpoints regardless of their terminating state. This is distinct from `ready` since `ready` is only true when pods are not terminating. `terminating` is true when an endpoint is terminating. For pods this is any endpoint with a deletion timestamp. (#92968, @andrewsykim) [SIG Apps and Network]
- Add support for hugepages to downward API (#86102, @derekwaynecarr) [SIG API Machinery, Apps, CLI, Network, Node, Scheduling and Testing]

- Adds kubelet alpha feature, `GracefulNodeShutdown` which makes kubelet aware of node system shutdowns and result in graceful termination of pods during a system shutdown. (#96129, @bobbypage) [SIG Node]

- AppProtocol is now GA for Endpoints and Services. The ServiceAppProtocol feature gate will be deprecated in 1.21. (#96327, @robscott) [SIG Apps and Network]

- Automatic allocation of NodePorts for services with type LoadBalancer can now be disabled by setting the (new) parameter Service.spec.allocateLoadBalancerNodePorts=false. The default is to allocate NodePorts for services with type LoadBalancer which is the existing behavior. (#92744, @uablrek) [SIG Apps and Network]

- Document that ServiceTopology feature is required to use `service.spec.topologyKeys`. (#96528, @andrewsykim) [SIG Apps]

- EndpointSlice has a new NodeName field guarded by the EndpointSliceNodeName feature gate.

    - EndpointSlice topology field will be deprecated in an upcoming release.
    - EndpointSlice "IP" address type is formally removed after being deprecated in Kubernetes 1.17.
    - The discovery.k8s.io/v1alpha1 API is deprecated and will be removed in Kubernetes 1.21. (#96440, @robscott) [SIG API Machinery, Apps and Network]

- Fewer candidates are enumerated for preemption to improve performance in large clusters (#94814, @adtac) [SIG Scheduling]

- If BoundServiceAccountTokenVolume is enabled, 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` (#96273, @zshihang) [SIG API Machinery and Auth]

- Introduce alpha support for exec-based container registry credential provider plugins in the kubelet. (#94196, @andrewsykim) [SIG Node and Release]

- Kube-apiserver now deletes expired kube-apiserver Lease objects:

    - The feature is under feature gate `APIServerIdentity`.
    - A flag is added to kube-apiserver: `identity-lease-garbage-collection-check-period-seconds` (#95895, @roycaihw) [SIG API Machinery, Apps, Auth and Testing]

- Move configurable fsgroup change policy for pods to beta (#96376, @gnufied) [SIG Apps and Storage]

New flag is introduced, i.e. --topology-manager-scope=container|pod.
- The default value is the "container" scope. (#92967, @cezaryzukowski) [SIG Instrumentation, Node and Testing]

- NodeAffinity plugin can be configured with AddedAffinity. (#96202, @alculquicondor) [SIG Node, Scheduling and Testing]

- Promote RuntimeClass feature to GA. Promote node.k8s.io API groups from v1beta1 to v1. (#95718, @SergeyKanzhelev) [SIG Apps, Auth, Node, Scheduling and Testing]

- Reminder: The labels "failure-domain.beta.kubernetes.io/zone" and "failure-domain.beta.kubernetes.io/region" are deprecated in favor of "topology.kubernetes.io/zone" and "topology.kubernetes.io/region" respectively. All users of the "failure-domain.beta..." labels should switch to the "topology..." equivalents. (#96033, @thockin) [SIG API Machinery, Apps, CLI, Cloud Provider, Network, Node, Scheduling, Storage and Testing]

- The usage of mixed protocol values in the same LoadBalancer Service is possible if the new feature gate MixedProtocolLBSVC is enabled. "action required" The feature gate is disabled by default. The user has to enable it for the API Server. (#94028, @janosi) [SIG API Machinery and Apps]

- This PR will introduce a feature gate CSIServiceAccountToken with two additional fields in `CSIDriverSpec`. (#93130, @zshihang) [SIG API Machinery, Apps, Auth, CLI, Network, Node, Storage and Testing]

- Users can try the cronjob controller v2 using the feature gate. This will be the default controller in future releases. (#93370, @alaypatel07) [SIG API Machinery, Apps, Auth and Testing]

- VolumeSnapshotDataSource moves to GA in 1.20 release (#95282, @xing-yang) [SIG Apps]

Feature

- **Additional documentation e.g., KEPs (Kubernetes Enhancement Proposals), usage docs, etc.**: (#95896, @zshihang) [SIG API Machinery and Cluster Lifecycle]
- A new set of alpha metrics are reported by the Kubernetes scheduler under the `/metrics/resources` endpoint that allow administrators to easily see the resource consumption (requests and limits for all resources on the pods) and compare it to actual pod usage or node capacity. (#94866, @smarterclayton) [SIG API Machinery, Instrumentation, Node and Scheduling]
- Add --experimental-logging-sanitization flag enabling runtime protection from leaking sensitive data in logs (#96370, @serathius) [SIG API Machinery, Cluster Lifecycle and Instrumentation]
- Add a StorageVersionAPI feature gate that makes API server update storageversions before serving certain write requests. This feature

allows the storage migrator to manage storage migration for built-in resources. Enabling internal.apiserver.k8s.io/v1alpha1 API and APIServerIdentity feature gate are required to use this feature. (#93873, @roycaihw) [SIG API Machinery, Auth and Testing]
- Add a new `vSphere` metric: `cloudprovider_vsphere_vcenter_versions`. It's content show `vCenter` hostnames with the associated server version. (#94526, @Danil-Grigorev) [SIG Cloud Provider and Instrumentation]
- Add feature to size memory backed volumes (#94444, @derekwaynecarr) [SIG Storage and Testing]
- Add node_authorizer_actions_duration_seconds metric that can be used to estimate load to node authorizer. (#92466, @mborsz) [SIG API Machinery, Auth and Instrumentation]
- Add pod_ based CPU and memory metrics to Kubelet's /metrics/resource endpoint (#95839, @egernst) [SIG Instrumentation, Node and Testing]
- Adds a headless service on node-local-cache addon. (#88412, @stafot) [SIG Cloud Provider and Network]
- CRDs: For structural schemas, non-nullable null map fields will now be dropped and defaulted if a default is available. null items in list will continue being preserved, and fail validation if not nullable. (#95423, @apelisse) [SIG API Machinery]
- E2e test for PodFsGroupChangePolicy (#96247, @saikat-royc) [SIG Storage and Testing]
- Gradudate the Pod Resources API to G.A Introduces the pod_resources_endpoint_requests_total metric which tracks the total number of requests to the pod resources API (#92165, @RenaudWasTaken) [SIG Instrumentation, Node and Testing]
- Introduce api-extensions category which will return: mutating admission configs, validating admission configs, CRDs and APIServices when used in kubectl get, for example. (#95603, @soltysh) [SIG API Machinery]
- Kube-apiserver now maintains a Lease object to identify itself:
  - The feature is under feature gate `APIServerIdentity`.
  - Two flags are added to kube-apiserver: `identity-lease-duration-seconds`, `identity-lease-renew-interval-seconds` (#95533, @roycaihw) [SIG API Machinery]
- Kube-apiserver: The timeout used when making health check calls to etcd can now be configured with `--etcd-healthcheck-timeout`. The default timeout is 2 seconds, matching the previous behavior. (#93244, @Sh4d1) [SIG API Machinery]
- Kubectl: Previously users cannot provide arguments to a external diff tool via KUBECTL_EXTERNAL_DIFF env. This release now allow users to specify args to KUBECTL_EXTERNAL_DIFF env. (#95292, @dougsland) [SIG CLI]
- Scheduler now ignores Pod update events if the resourceVersion of old and new Pods are identical. (#96071, @Huang-Wei) [SIG Scheduling]
- Support custom tags for cloud provider managed resources (#96450, @nilo19) [SIG Cloud Provider]
- Support customize load balancer health probe protocol and request path (#96338, @nilo19) [SIG Cloud Provider]

- Support multiple standard load balancers in one cluster (#96111, @nilo19) [SIG Cloud Provider]
- The beta `RootCAConfigMap` feature gate is enabled by default and causes kube-controller-manager to publish a "kube-root-ca.crt" ConfigMap to every namespace. This ConfigMap contains a CA bundle used for verifying connections to the kube-apiserver. (#96197, @zshihang) [SIG API Machinery, Apps, Auth and Testing]
- The kubelet_runtime_operations_duration_seconds metric got additional buckets of 60, 300, 600, 900 and 1200 seconds (#96054, @alvaroaleman) [SIG Instrumentation and Node]
- There is a new pv_collector_total_pv_count metric that counts persistent volumes by the volume plugin name and volume mode. (#95719, @tsmetana) [SIG Apps, Instrumentation, Storage and Testing]
- Volume snapshot e2e test to validate PVC and VolumeSnapshotContent finalizer (#95863, @RaunakShah) [SIG Cloud Provider, Storage and Testing]
- Warns user when executing kubectl apply/diff to resource currently being deleted. (#95544, @SaiHarshaK) [SIG CLI]
- `kubectl alpha debug` has graduated to beta and is now `kubectl debug`. (#96138, @verb) [SIG CLI and Testing]
- `kubectl debug` gains support for changing container images when copying a pod for debugging, similar to how `kubectl set image` works. See `kubectl help debug` for more information. (#96058, @verb) [SIG CLI]

Documentation

- Updates docs and guidance on cloud provider InstancesV2 and Zones interface for external cloud providers:
  - removes experimental warning for InstancesV2
  - document that implementation of InstancesV2 will disable calls to Zones
  - deprecate Zones in favor of InstancesV2 (#96397, @andrewsykim) [SIG Cloud Provider]

Bug or Regression

- Change plugin name in fsgroupapplymetrics of csi and flexvolume to distinguish different driver (#95892, @JornShen) [SIG Instrumentation, Storage and Testing]
- Clear UDP conntrack entry on endpoint changes when using nodeport (#71573, @JacobTanenbaum) [SIG Network]
- Exposes and sets a default timeout for the TokenReview client for DelegatingAuthenticationOptions (#96217, @p0lyn0mial) [SIG API Machinery and Cloud Provider]
- Fix CVE-2020-8555 for Quobyte client connections. (#95206, @misterikkit) [SIG Storage]
- Fix IP fragmentation of UDP and TCP packets not supported issues on LoadBalancer rules (#96464, @nilo19) [SIG Cloud Provider]

- Fix a bug that DefaultPreemption plugin is disabled when using (legacy) scheduler policy. (#96439, @Huang-Wei) [SIG Scheduling and Testing]
- Fix bug in JSON path parser where an error occurs when a range is empty (#95933, @brianpursley) [SIG API Machinery]
- Fix client-go prometheus metrics to correctly present the API path accessed in some environments. (#74363, @aanm) [SIG API Machinery]
- Fix memory leak in kube-apiserver when underlying time goes forth and back. (#96266, @chenyw1990) [SIG API Machinery]
- Fix paging issues when Azure API returns empty values with non-empty nextLink (#96211, @feiskyer) [SIG Cloud Provider]
- Fix pull image error from multiple ACRs using azure managed identity (#96355, @andyzhangx) [SIG Cloud Provider]
- Fix vSphere volumes that could be erroneously attached to wrong node (#96224, @gnufied) [SIG Cloud Provider and Storage]
- Fixed a bug that prevents kubectl to validate CRDs with schema using x-kubernetes-preserve-unknown-fields on object fields. (#96369, @gautierdelorme) [SIG API Machinery and Testing]
- For vSphere Cloud Provider, If VM of worker node is deleted, the node will also be deleted by node controller (#92608, @lubronzhan) [SIG Cloud Provider]
- HTTP/2 connection health check is enabled by default in all Kubernetes clients. The feature should work out-of-the-box. If needed, users can tune the feature via the HTTP2_READ_IDLE_TIMEOUT_SECONDS and HTTP2_PING_TIMEOUT_SECONDS environment variables. The feature is disabled if HTTP2_READ_IDLE_TIMEOUT_SECONDS is set to 0. (#95981, @caesarxuchao) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Node]
- If the user specifies an invalid timeout in the request URL, the request will be aborted with an HTTP 400.
    - If the user specifies a timeout in the request URL that exceeds the maximum request deadline allowed by the apiserver, the request will be aborted with an HTTP 400. (#96061, @tkashem) [SIG API Machinery, Network and Testing]
- Improve error messages related to nodePort endpoint changes conntrack entries cleanup. (#96251, @ravens) [SIG Network]
- Print go stack traces at -v=4 and not -v=2 (#94663, @soltysh) [SIG CLI]
- Remove ready file and its directory (which is created during volume SetUp) during emptyDir volume TearDown. (#95770, @jingxu97) [SIG Storage]
- Resolves non-deterministic behavior of the garbage collection controller when ownerReferences with incorrect data are encountered. Events with a reason of `OwnerRefInvalidNamespace` are recorded when namespace mismatches between child and owner objects are detected.
    - A namespaced object with an ownerReference referencing a uid of a namespaced kind which does not exist in the same namespace is now consistently treated as though that owner does not exist, and the child object is deleted.
    - A cluster-scoped object with an ownerReference referencing a uid of a namespaced kind is now consistently treated as though that

owner is not resolvable, and the child object is ignored by the garbage collector. (#92743, @liggitt) [SIG API Machinery, Apps and Testing]
- Skip [k8s.io/kubernetes@v1.19.0/test/e2e/storage/testsuites/base.go:162]: Driver azure-disk doesn't support snapshot type DynamicSnapshot -- skipping skip [k8s.io/kubernetes@v1.19.0/test/e2e/storage/testsuites/base.go:185]: Driver azure-disk doesn't support ntfs -- skipping (#96144, @qinpingli) [SIG Storage and Testing]
- The AWS network load balancer attributes can now be specified during service creation (#95247, @kishorj) [SIG Cloud Provider]
- The kube-apiserver will no longer serve APIs that should have been deleted in GA non-alpha levels. Alpha levels will continue to serve the removed APIs so that CI doesn't immediately break. (#96525, @deads2k) [SIG API Machinery]
- Update max azure data disk count map (#96308, @andyzhangx) [SIG Cloud Provider and Storage]
- Update the route table tag in the route reconcile loop (#96545, @nilo19) [SIG Cloud Provider]
- Volume binding: report UnschedulableAndUnresolvable status instead of an error when bound PVs not found (#95541, @cofyc) [SIG Apps, Scheduling and Storage]
- [kubectl] Fail when local source file doesn't exist (#90333, @bamarni) [SIG CLI]

Other (Cleanup or Flake)

- Handle slow cronjob lister in cronjob controller v2 and improve memory footprint. (#96443, @alaypatel07) [SIG Apps]
- --redirect-container-streaming is no longer functional. The flag will be removed in v1.22 (#95935, @tallclair) [SIG Node]
- A new metric `requestAbortsTotal` has been introduced that counts aborted requests for each `group`, `version`, `verb`, `resource`, `subresource` and `scope`. (#95002, @p0lyn0mial) [SIG API Machinery, Cloud Provider, Instrumentation and Scheduling]
- API priority and fairness metrics use snake_case in label names (#96236, @adtac) [SIG API Machinery, Cluster Lifecycle, Instrumentation and Testing]
- Applies translations on all command descriptions (#95439, @HerrNaN) [SIG CLI]
- Changed: default "Accept-Encoding" header removed from HTTP probes. See https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#http-probes (#96127, @fonsecas72) [SIG Network and Node]
- Generators for services are removed from kubectl (#95256, @Git-Jiro) [SIG CLI]
- Introduce kubectl-convert plugin. (#96190, @soltysh) [SIG CLI and Testing]
- Kube-scheduler now logs processed component config at startup (#96426, @damemi) [SIG Scheduling]
- NONE (#96179, @bbyrne5) [SIG Network]

- Users will now be able to configure all supported values for AWS NLB health check interval and thresholds for new resources. (#96312, @kishorj) [SIG Cloud Provider]

Dependencies

Added

- cloud.google.com/go/firestore: v1.1.0
- github.com/armon/go-metrics: f0300d1
- github.com/armon/go-radix: 7fddfc3
- github.com/bketelsen/crypt: 5cbc8cc
- github.com/hashicorp/consul/api: v1.1.0
- github.com/hashicorp/consul/sdk: v0.1.1
- github.com/hashicorp/errwrap: v1.0.0
- github.com/hashicorp/go-cleanhttp: v0.5.1
- github.com/hashicorp/go-immutable-radix: v1.0.0
- github.com/hashicorp/go-msgpack: v0.5.3
- github.com/hashicorp/go-multierror: v1.0.0
- github.com/hashicorp/go-rootcerts: v1.0.0
- github.com/hashicorp/go-sockaddr: v1.0.0
- github.com/hashicorp/go-uuid: v1.0.1
- github.com/hashicorp/go.net: v0.0.1
- github.com/hashicorp/logutils: v1.0.0
- github.com/hashicorp/mdns: v1.0.0
- github.com/hashicorp/memberlist: v0.1.3
- github.com/hashicorp/serf: v0.8.2
- github.com/mitchellh/cli: v1.0.0
- github.com/mitchellh/go-testing-interface: v1.0.0
- github.com/mitchellh/gox: v0.4.0
- github.com/mitchellh/iochan: v1.0.0
- github.com/pascaldekloe/goe: 57f6aae
- github.com/posener/complete: v1.1.1
- github.com/ryanuber/columnize: 9b3edd6
- github.com/sean-/seed: e2103e2
- github.com/subosito/gotenv: v1.2.0
- github.com/willf/bitset: d5bec33
- gopkg.in/ini.v1: v1.51.0
- gopkg.in/yaml.v3: 9f266ea
- rsc.io/quote/v3: v3.1.0
- rsc.io/sampler: v1.3.0

Changed

- cloud.google.com/go/bigquery: v1.0.1 â†’ v1.4.0
- cloud.google.com/go/datastore: v1.0.0 â†’ v1.1.0
- cloud.google.com/go/pubsub: v1.0.1 â†’ v1.2.0
- cloud.google.com/go/storage: v1.0.0 â†’ v1.6.0
- cloud.google.com/go: v0.51.0 â†’ v0.54.0
- github.com/Microsoft/go-winio: fc70bd9 â†’ v0.4.15
- github.com/aws/aws-sdk-go: v1.35.5 â†’ v1.35.24

- github.com/blang/semver: v3.5.0+incompatible â†’ v3.5.1+incompatible
- github.com/checkpoint-restore/go-criu/v4: v4.0.2 â†’ v4.1.0
- github.com/containerd/containerd: v1.3.3 â†’ v1.4.1
- github.com/containerd/ttrpc: v1.0.0 â†’ v1.0.2
- github.com/containerd/typeurl: v1.0.0 â†’ v1.0.1
- github.com/coreos/etcd: v3.3.10+incompatible â†’ v3.3.13+incompatible
- github.com/docker/docker: aa6a989 â†’ bd33bbf
- github.com/go-gl/glfw/v3.3/glfw: 12ad95a â†’ 6f7a984
- github.com/golang/groupcache: 215e871 â†’ 8c9f03a
- github.com/golang/mock: v1.3.1 â†’ v1.4.1
- github.com/golang/protobuf: v1.4.2 â†’ v1.4.3
- github.com/google/cadvisor: v0.37.0 â†’ v0.38.4
- github.com/google/go-cmp: v0.4.0 â†’ v0.5.2
- github.com/google/pprof: d4f498a â†’ 1ebb73c
- github.com/google/uuid: v1.1.1 â†’ v1.1.2
- github.com/gorilla/mux: v1.7.3 â†’ v1.8.0
- github.com/gorilla/websocket: v1.4.0 â†’ v1.4.2
- github.com/karrick/godirwalk: v1.7.5 â†’ v1.16.1
- github.com/opencontainers/runc: 819fcc6 â†’ v1.0.0-rc92
- github.com/opencontainers/runtime-spec: 237cc4f â†’ 4d89ac9
- github.com/opencontainers/selinux: v1.5.2 â†’ v1.6.0
- github.com/prometheus/procfs: v0.1.3 â†’ v0.2.0
- github.com/quobyte/api: v0.1.2 â†’ v0.1.8
- github.com/spf13/cobra: v1.0.0 â†’ v1.1.1
- github.com/spf13/viper: v1.4.0 â†’ v1.7.0
- github.com/stretchr/testify: v1.4.0 â†’ v1.6.1
- github.com/vishvananda/netns: 52d707b â†’ db3c7e5
- go.opencensus.io: v0.22.2 â†’ v0.22.3
- golang.org/x/exp: da58074 â†’ 6cc2880
- golang.org/x/lint: fdd1cda â†’ 738671d
- golang.org/x/net: ab34263 â†’ 69a7880
- golang.org/x/oauth2: 858c2ad â†’ bf48bf1
- golang.org/x/sys: ed371f2 â†’ 5cba982
- golang.org/x/text: v0.3.3 â†’ v0.3.4
- golang.org/x/time: 555d28b â†’ 3af7569
- golang.org/x/xerrors: 9bdfabe â†’ 5ec99f8
- google.golang.org/api: v0.15.1 â†’ v0.20.0
- google.golang.org/genproto: cb27e3a â†’ 8816d57
- google.golang.org/grpc: v1.27.0 â†’ v1.27.1
- google.golang.org/protobuf: v1.24.0 â†’ v1.25.0
- honnef.co/go/tools: v0.0.1-2019.2.3 â†’ v0.0.1-2020.1.3
- k8s.io/gengo: 8167cfd â†’ 83324d8
- k8s.io/klog/v2: v2.2.0 â†’ v2.4.0
- k8s.io/kube-openapi: 8b50664 â†’ d219536
- k8s.io/utils: d5654de â†’ 67b214c
- sigs.k8s.io/apiserver-network-proxy/konnectivity-client: v0.0.12 â†’ v0.0.14
- sigs.k8s.io/structured-merge-diff/v4: b3cf1e8 â†’ v4.0.2

Removed

- github.com/armon/consul-api: eb2c6b5
- github.com/go-ini/ini: v1.9.0
- github.com/ugorji/go: v1.1.4
- github.com/xordataexchange/crypt: b2862e3

v1.20.0-beta.1

Downloads for v1.20.0-beta.1

Source Code

filename	sha512 hash
kubernetes.tar.gz	4eddf4850c2d57751696f352d0667309339090aeb30ff93e8db8a22c6c
kubernetes-src.tar.gz	59de5221162e9b6d88f5abbdb99765cb2b2e501498ea853fb65f2abe3

Client binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	d69ffed19b034a4221fc084e43ac293cf392e98febf5bf580f8d92307a
kubernetes-client-linux-386.tar.gz	1b542e165860c4adcd4550adc19b86c3db8cd75d2a1b8db17becc75
kubernetes-client-linux-amd64.tar.gz	90ad52785eecb43a6f9035b92b6ba39fc84e67f8bc91cf098e70f8cfdc
kubernetes-client-linux-arm.tar.gz	d0cb3322b056e1821679afa70728ffc0d3375e8f3326dabbe8185be2
kubernetes-client-linux-arm64.tar.gz	3aecc8197e0aa368408624add28a2dd5e73f0d8a48e5e33c19edf91d
kubernetes-client-linux-ppc64le.tar.gz	6ff145058f62d478b98f1e418e272555bfb5c7861834fbbf10a8fb334c
kubernetes-client-linux-s390x.tar.gz	ff7b8bb894076e05a3524f6327a4a6353b990466f3292e84c92826cb
kubernetes-client-windows-386.tar.gz	6c6dcac9c725605763a130b5a975f2b560aa976a5c809d4e0887900
kubernetes-client-windows-amd64.tar.gz	d12e3a29c960f0ddd1b9aabf5426ac1259863ac6c8f2be1736ebeb57

Server binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	904e8c049179e071c6caa65f525f465260bb4d4318a6dd9cc05be2172f39
kubernetes-server-linux-arm.tar.gz	5934959374868aed8d4294de84411972660bca7b2e952201a9403f37e40
kubernetes-server-linux-arm64.tar.gz	4c884585970f80dc5462d9a734d7d5be9558b36c6e326a8a3139423efbd7
kubernetes-server-linux-ppc64le.tar.gz	235b78b08440350dcb9f13b63f7722bd090c672d8e724ca5d409256e5a5c
kubernetes-server-linux-s390x.tar.gz	220fc9351702b3ecdcf79089892ceb26753a8a1deaf46922ffb3d3b62b999

Node binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	fe59d3a1f21c47bab126f689687657f77fbcb46a2caeef48eecd073b2b2287
kubernetes-node-linux-arm.tar.gz	93e545aa963cfd11e0b2c6d47669b5ef70c5a86ef80c3353c1a074396bff1c
kubernetes-node-linux-arm64.tar.gz	5e0f177f9bec406a668d4b37e69b191208551fdf289c82b5ec898959da4f8
kubernetes-node-linux-ppc64le.tar.gz	574412059e4d257eb904cd4892a075b6a2cde27adfa4976ee64c46d6768f
kubernetes-node-linux-s390x.tar.gz	b1ffaa6d7f77d89885c642663cb14a86f3e2ec2afd223e3bb2000962758cf0
kubernetes-node-windows-amd64.tar.gz	388983765213cf3bdc1f8b27103ed79e39028767e5f1571e35ed1f91ed100

Changelog since v1.20.0-beta.0

Changes by Kind

Deprecation

- ACTION REQUIRED: The kube-apiserver ability to serve on an insecure port, deprecated since v1.10, has been removed. The insecure address flags `--address` and `--insecure-bind-address` have no effect in kube-apiserver and will be removed in v1.24. The insecure port flags `--port` and `--insecure-port` may only be set to 0 and will be removed in v1.24. (#95856, @knight42) [SIG API Machinery, Node and Testing]

API Change

-
  - `TokenRequest` and `TokenRequestProjection` features have been promoted to GA. This feature allows generating service account tokens that are not visible in Secret objects and are tied to the lifetime of a Pod object. See https://kubernetes.io/docs/tasks/configure-pod-container/configure-service-account/#service-account-token-volume-projection for details on configuring and using this feature. The `TokenRequest` and `TokenRequestProjection` feature gates will be removed in v1.21.
  - kubeadm's kube-apiserver Pod manifest now includes the following flags by default "--service-account-key-file", "--service-account-signing-key-file", "--service-account-issuer". (#93258, @zshihang) [SIG API Machinery, Auth, Cluster Lifecycle, Storage and Testing]
- Certain fields on Service objects will be automatically cleared when changing the service's `type` to a mode that does not need those fields. For example, changing from type=LoadBalancer to type=ClusterIP will clear the NodePort assignments, rather than forcing the user to clear them. (#95196, @thockin) [SIG API Machinery, Apps, Network and Testing]
- Services will now have a `clusterIPs` field to go with `clusterIP`. `clusterIPs[0]` is a synonym for `clusterIP` and will be syncronized on create and update operations. (#95894, @thockin) [SIG Network]

Feature

- A new metric `apiserver_request_filter_duration_seconds` has been introduced that measures request filter latency in seconds. (#95207, @tkashem) [SIG API Machinery and Instrumentation]
- Add a new flag to set priority for the kubelet on Windows nodes so that workloads cannot overwhelm the node there by disrupting kubelet process. (#96051, @ravisantoshgudimetla) [SIG Node and Windows]
- Changed: default "Accept: /" header added to HTTP probes. See https://kubernetes.io/docs/tasks/configure-pod-container/configure-liveness-readiness-startup-probes/#http-probes (https://github.com/kubernetes/website/pull/24756) (#95641, @fonsecas72) [SIG Network and Node]

- Client-go credential plugins can now be passed in the current cluster information via the KUBERNETES_EXEC_INFO environment variable. (#95489, @ankeesler) [SIG API Machinery and Auth]
- Kube-apiserver: added support for compressing rotated audit log files with `--audit-log-compress` (#94066, @lojies) [SIG API Machinery and Auth]

Documentation

- Fake dynamic client: document that List does not preserve TypeMeta in UnstructuredList (#95117, @andrewsykim) [SIG API Machinery]

Bug or Regression

- Added support to kube-proxy for externalTrafficPolicy=Local setting via Direct Server Return (DSR) load balancers on Windows. (#93166, @elweb9858) [SIG Network]
- Disable watchcache for events (#96052, @wojtek-t) [SIG API Machinery]
- Disabled `LocalStorageCapacityIsolation` feature gate is honored during scheduling. (#96092, @Huang-Wei) [SIG Scheduling]
- Fix bug in JSON path parser where an error occurs when a range is empty (#95933, @brianpursley) [SIG API Machinery]
- Fix k8s.io/apimachinery/pkg/api/meta.SetStatusCondition to update ObservedGeneration (#95961, @KnicKnic) [SIG API Machinery]
- Fixed a regression which prevented pods with `docker/default` seccomp annotations from being created in 1.19 if a PodSecurityPolicy was in place which did not allow `runtime/default` seccomp profiles. (#95985, @saschagrunert) [SIG Auth]
- Kubectl: print error if users place flags before plugin name (#92343, @knight42) [SIG CLI]
- When creating a PVC with the volume.beta.kubernetes.io/storage-provisioner annotation already set, the PV controller might have incorrectly deleted the newly provisioned PV instead of binding it to the PVC, depending on timing and system load. (#95909, @pohly) [SIG Apps and Storage]

Other (Cleanup or Flake)

- Kubectl: the `generator` flag of `kubectl autoscale` has been deprecated and has no effect, it will be removed in a feature release (#92998, @SataQiu) [SIG CLI]
- V1helpers.MatchNodeSelectorTerms now accepts just a Node and a list of Terms (#95871, @damemi) [SIG Apps, Scheduling and Storage]
- `MatchNodeSelectorTerms` function moved to `k8s.io/component-helpers` (#95531, @damemi) [SIG Apps, Scheduling and Storage]

Dependencies

Added

Nothing has changed.

Changed

Nothing has changed.

Removed

Nothing has changed.

v1.20.0-beta.0

Downloads for v1.20.0-beta.0

Source Code

filename	sha512 hash
kubernetes.tar.gz	385e49e32bbd6996f07bcadbf42285755b8a8ef9826ee1ba42bd82c658
kubernetes-src.tar.gz	842e80f6dcad461426fb699de8a55fde8621d76a94e54288fe9939cc1a

Client binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	bde5e7d9ee3e79d1e69465a3ddb4bb36819a4f281b5c01a7976816d
kubernetes-client-linux-386.tar.gz	721bb8444c9e0d7a9f8461e3f5428882d76fcb3def6eb11b8e8e08fae
kubernetes-client-linux-amd64.tar.gz	71e4edc41afbd65f813e7ecbc22b27c95f248446f005e288d758138d
kubernetes-client-linux-arm.tar.gz	bbefc749156f63898973f2f7c7a6f1467481329fb430d641fe659b497
kubernetes-client-linux-arm64.tar.gz	9803190685058b4b64d002c2fbfb313308bcea4734ed53a8c340cfda
kubernetes-client-linux-ppc64le.tar.gz	bcdceea64cba1ae38ea2bab50d8fd77c53f6d673de12566050b0e3c2
kubernetes-client-linux-s390x.tar.gz	41e36d00867e90012d5d5adfabfaae8d9f5a9fd32f290811e3c368e11
kubernetes-client-windows-386.tar.gz	c50fec5aec2d0e742f851f25c236cb73e76f8fc73b0908049a10ae736

filename	sha512 hash
kubernetes-client-windows-amd64.tar.gz	0fd6777c349908b6d627e849ea2d34c048b8de41f7df8a19898623f5

Server binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	30d982424ca64bf0923503ae8195b2e2a59497096b2d9e58dfd491cd663
kubernetes-server-linux-arm.tar.gz	f08b62be9bc6f0745f820b0083c7a31eedb2ce370a037c768459a591921
kubernetes-server-linux-arm64.tar.gz	e3472b5b3dfae0a56e5363d52062b1e4a9fc227a05e0cf5ece38233b2c44
kubernetes-server-linux-ppc64le.tar.gz	06c254e0a62f755d31bc40093d86c44974f0a60308716cc3214a6b3c249a
kubernetes-server-linux-s390x.tar.gz	2edeb4411c26a0de057a66787091ab1044f71774a464aed898ffee26634a

Node binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	cc1d5b94b86070b5e7746d7aaeaeac3b3a5e5ebbff1ec33885f7eeab270a6
kubernetes-node-linux-arm.tar.gz	75e82c7c9122add3b24695b94dcb0723c52420c3956abf47511e37785aa4
kubernetes-node-linux-arm64.tar.gz	16ef27c40bf4d678a55fcd3d3f7d09f1597eec2cc58f9950946f0901e52b82
kubernetes-node-linux-ppc64le.tar.gz	939865f2c4cb6a8934f22a06223e416dec5f768ffc1010314586149470420
kubernetes-node-linux-s390x.tar.gz	bbfdd844075fb816079af7b73d99bc1a78f41717cdbadb043f6f5872b4dc4
kubernetes-node-windows-amd64.tar.gz	a2b3ea40086fd71aed71a4858fd3fc79fd1907bc9ea8048ff3c82ec56477b0

Changelog since v1.20.0-alpha.3

Urgent Upgrade Notes

(No, really, you MUST read this before you upgrade)

- Kubeadm: improve the validation of serviceSubnet and podSubnet. ServiceSubnet has to be limited in size, due to implementation details, and the mask can not allocate more than 20 bits. PodSubnet validates against the corresponding cluster "--node-cidr-mask-size" of the kube-controller-manager, it fail if the values are not compatible. kubeadm no longer sets the node-mask automatically on IPv6 deployments, you must check that your IPv6 service subnet mask is compatible with the default node mask /64 or set it accordenly. Previously, for IPv6, if the podSubnet had a mask lower than /112, kubeadm calculated a node-mask to be multiple of eight and splitting the available bits to maximise the number used for nodes. (#95723, @aojea) [SIG Cluster Lifecycle]
- Windows hyper-v container featuregate is deprecated in 1.20 and will be removed in 1.21 (#95505, @wawa0210) [SIG Node and Windows]

Changes by Kind

Deprecation

- Support 'controlplane' as a valid EgressSelection type in the EgressSelectorConfiguration API. 'Master' is deprecated and will be removed in v1.22. (#95235, @andrewsykim) [SIG API Machinery]

API Change

- Add dual-stack Services (alpha). This is a BREAKING CHANGE to an alpha API. It changes the dual-stack API wrt Service from a single ipFamily field to 3 fields: ipFamilyPolicy (SingleStack, PreferDualStack, RequireDualStack), ipFamilies (a list of families assigned), and clusterIPs (inclusive of clusterIP). Most users do not need to set anything at all, defaulting will handle it for them. Services are single-stack unless the user asks for dual-stack. This is all gated by the "IPv6DualStack" feature gate. (#91824, @khenidak) [SIG API Machinery, Apps, CLI, Network, Node, Scheduling and Testing]
- Introduces a metric source for HPAs which allows scaling based on container resource usage. (#90691, @arjunrn) [SIG API Machinery, Apps, Autoscaling and CLI]

Feature

- Add a metric for time taken to perform recursive permission change (#95866, @JornShen) [SIG Instrumentation and Storage]
- Allow cross compilation of kubernetes on different platforms. (#94403, @bnrjee) [SIG Release]

- Command to start network proxy changes from 'KUBE_ENABLE_EGRESS_VIA_KONNECTIVITY_SERVICE ./cluster/kube-up.sh' to 'KUBE_ENABLE_KONNECTIVITY_SERVICE=true ./hack/kube-up.sh' (#92669, @Jefftree) [SIG Cloud Provider]
- DefaultPodTopologySpread graduated to Beta. The feature gate is enabled by default. (#95631, @alculquicondor) [SIG Scheduling and Testing]
- Kubernetes E2E test image manifest lists now contain Windows images. (#77398, @claudiubelu) [SIG Testing and Windows]
- Support for Windows container images (OS Versions: 1809, 1903, 1909, 2004) was added the pause:3.4 image. (#91452, @claudiubelu) [SIG Node, Release and Windows]

Documentation

- Fake dynamic client: document that List does not preserve TypeMeta in UnstructuredList (#95117, @andrewsykim) [SIG API Machinery]

Bug or Regression

- Exposes and sets a default timeout for the SubjectAccessReview client for DelegatingAuthorizationOptions. (#95725, @p0lyn0mial) [SIG API Machinery and Cloud Provider]
- Alter wording to describe pods using a pvc (#95635, @RaunakShah) [SIG CLI]
- If we set SelectPolicy MinPolicySelect on scaleUp behavior or scaleDown behavior,Horizontal Pod Autoscaler doesn`t automatically scale the number of pods correctly (#95647, @JoshuaAndrew) [SIG Apps and Autoscaling]
- Ignore apparmor for non-linux operating systems (#93220, @wawa0210) [SIG Node and Windows]
- Ipvs: ensure selected scheduler kernel modules are loaded (#93040, @cmluciano) [SIG Network]
- Kubeadm: add missing "--experimental-patches" flag to "kubeadm init phase control-plane" (#95786, @Sh4d1) [SIG Cluster Lifecycle]
- Reorganized iptables rules to fix a performance issue (#95252, @tssurya) [SIG Network]
- Unhealthy pods covered by PDBs can be successfully evicted if enough healthy pods are available. (#94381, @michaelgugino) [SIG Apps]
- Update the PIP when it is not in the Succeeded provisioning state during the LB update. (#95748, @nilo19) [SIG Cloud Provider]
- Update the frontend IP config when the service's `pipName` annotation is changed (#95813, @nilo19) [SIG Cloud Provider]

Other (Cleanup or Flake)

- NO (#95690, @nikhita) [SIG Release]

Dependencies

Added

- github.com/form3tech-oss/jwt-go: v3.2.2+incompatible

Changed

- github.com/Azure/go-autorest/autorest/adal: v0.9.0 â†’ v0.9.5
- github.com/Azure/go-autorest/autorest/mocks: v0.4.0 â†’ v0.4.1
- golang.org/x/crypto: 75b2880 â†’ 7f63de1

Removed

Nothing has changed.

v1.20.0-alpha.3

Downloads for v1.20.0-alpha.3

Source Code

filename	sha512 hash
kubernetes.tar.gz	542cc9e0cd97732020491456402b6e2b4f54f2714007ee1374a7d3636
kubernetes-src.tar.gz	5e5d725294e552fd1d14fd6716d013222827ac2d4e2d11a7a1fdefb77b

Client binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	60004939727c75d0f06adc4449e16b43303941937c0e9ea9aca7d94
kubernetes-client-linux-386.tar.gz	7edba9c4f1bf38fdf1fa5bff2856c05c0e127333ce19b17edf3119dc9b
kubernetes-client-linux-amd64.tar.gz	db1818aa82d072cb3e32a2a988e66d76ecf7cebc6b8a29845fa2d6ec
kubernetes-client-linux-arm.tar.gz	d2922e70d22364b1f5a1e94a0c115f849fe2575b231b1ba268f73a9d
kubernetes-client-linux-arm64.tar.gz	2e3ae20e554c7d4fc3a8afdfcafe6bbc81d4c5e9aea036357baac7a3fd
kubernetes-client-linux-ppc64le.tar.gz	b54a34e572e6a86221577de376e6f7f9fcd82327f7fe94f2fc8d21f35d
kubernetes-client-linux-s390x.tar.gz	5be1b70dc437d3ba88cb0b89cd1bc555f79896c3f5b5f4fa0fb046a0d

filename	sha512 hash
kubernetes-client-windows-386.tar.gz	88cf3f66168ef3bf9a5d3d2275b7f33799406e8205f2c202997ebec23
kubernetes-client-windows-amd64.tar.gz	87d2d4ea1829da8cfa1a705a03ea26c759a03bd1c4d8b96f2c93264c

Server binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	7af691fc0b13a937797912374e3b3eeb88d5262e4eb7d4ebe92a3b64b3c2
kubernetes-server-linux-arm.tar.gz	557c47870ecf5c2090b2694c8f0c8e3b4ca23df5455a37945bd037bc6fb5b
kubernetes-server-linux-arm64.tar.gz	981de6cf7679d743cdeef1e894314357b6809013381480187050 4ef30564
kubernetes-server-linux-ppc64le.tar.gz	506578a21601ccff609ae757a55e68634c15cbfecbf13de972c96b32a155d
kubernetes-server-linux-s390x.tar.gz	af0cdcd4a77a7cc8060a076641615730a802f1f02dab084e41926023489ef

Node binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	2d92c61596296279de1efae23b2b707415565d9d50cd61a7231b8d10325
kubernetes-node-linux-arm.tar.gz	c298de9b5ac1b8778729a2d8e2793ff86743033254fbc27014333880b03c
kubernetes-node-linux-arm64.tar.gz	daa3c65afda6d7aff206c1494390bbcc205c2c6f8db04c10ca967a690578a
kubernetes-node-linux-ppc64le.tar.gz	05661908bb73bfcaf9c2eae96e9a6a793db5a7a100bce6df9e057985dd53a
kubernetes-node-linux-s390x.tar.gz	845e518e2c4ef0cef2c3b58f0b9ea5b5fe9b8a249717f789607752484c424
kubernetes-node-windows-amd64.tar.gz	530e536574ed2c3e5973d3c0f0fdd2b4d48ef681a7a7c02db13e60500166

Changelog since v1.20.0-alpha.2

Changes by Kind

API Change

- New parameter `defaultingType` for `PodTopologySpread` plugin allows to use k8s defined or user provided default constraints (#95048, @alculquicondor) [SIG Scheduling]

Feature

- Added new k8s.io/component-helpers repository providing shared helper code for (core) components. (#92507, @ingvagabund) [SIG Apps, Node, Release and Scheduling]
- Adds `create ingress` command to `kubectl` (#78153, @amimof) [SIG CLI and Network]
- Kubectl create now supports creating ingress objects. (#94327, @rikatz) [SIG CLI and Network]
- New default scheduling plugins order reduces scheduling and preemption latency when taints and node affinity are used (#95539, @soulxu) [SIG Scheduling]
- SCTP support in API objects (Pod, Service, NetworkPolicy) is now GA. Note that this has no effect on whether SCTP is enabled on nodes at the kernel level, and note that some cloud platforms and network plugins do not support SCTP traffic. (#95566, @danwinship) [SIG Apps and Network]
- Scheduling Framework: expose Run[Pre]ScorePlugins functions to PreemptionHandle which can be used in PostFilter extention point. (#93534, @everpeace) [SIG Scheduling and Testing]
- SelectorSpreadPriority maps to PodTopologySpread plugin when DefaultPodTopologySpread feature is enabled (#95448, @alculquicondor) [SIG Scheduling]
- SetHostnameAsFQDN has been graduated to Beta and therefore it is enabled by default. (#95267, @javidiaz) [SIG Node]

Bug or Regression

- An issues preventing volume expand controller to annotate the PVC with `volume.kubernetes.io/storage-resizer` when the PVC StorageClass is already updated to the out-of-tree provisioner is now fixed. (#94489, @ialidzhikov) [SIG API Machinery, Apps and Storage]
- Change the mount way from systemd to normal mount except ceph and glusterfs intree-volume. (#94916, @smileusd) [SIG Apps, Cloud Provider, Network, Node, Storage and Testing]
- Fix azure disk attach failure for disk size bigger than 4TB (#95463, @andyzhangx) [SIG Cloud Provider]
- Fix azure disk data loss issue on Windows when unmount disk (#95456, @andyzhangx) [SIG Cloud Provider and Storage]

- Fix verb & scope reporting for kube-apiserver metrics (LIST reported instead of GET) (#95562, @wojtek-t) [SIG API Machinery and Testing]
- Fix vsphere detach failure for static PVs (#95447, @gnufied) [SIG Cloud Provider and Storage]
- Fix: smb valid path error (#95583, @andyzhangx) [SIG Storage]
- Fixed a bug causing incorrect formatting of `kubectl describe ingress`. (#94985, @howardjohn) [SIG CLI and Network]
- Fixed a bug in client-go where new clients with customized `Dial`, `Proxy`, `GetCert` config may get stale HTTP transports. (#95427, @roycaihw) [SIG API Machinery]
- Fixes high CPU usage in kubectl drain (#95260, @amandahla) [SIG CLI]
- Support the node label `node.kubernetes.io/exclude-from-external-load-balancers` (#95542, @nilo19) [SIG Cloud Provider]

Other (Cleanup or Flake)

- Fix func name NewCreateCreateDeploymentOptions (#91931, @lixiaobing1) [SIG CLI]
- Kubeadm: update the default pause image version to 1.4.0 on Windows. With this update the image supports Windows versions 1809 (2019LTS), 1903, 1909, 2004 (#95419, @jsturtevant) [SIG Cluster Lifecycle and Windows]
- Upgrade snapshot controller to 3.0.0 (#95412, @saikat-royc) [SIG Cloud Provider]
- Remove the dependency of csi-translation-lib module on apiserver/cloud-provider/controller-manager (#95543, @wawa0210) [SIG Release]
- Scheduler framework interface moved from pkg/scheduler/framework/v1alpha to pkg/scheduler/framework (#95069, @farah) [SIG Scheduling, Storage and Testing]
- UDP and SCTP protocols can left stale connections that need to be cleared to avoid services disruption, but they can cause problems that are hard to debug. Kubernetes components using a loglevel greater or equal than 4 will log the conntrack operations and its output, to show the entries that were deleted. (#95694, @aojea) [SIG Network]

Dependencies

Added

Nothing has changed.

Changed

Nothing has changed.

Removed

Nothing has changed.

v1.20.0-alpha.2

Downloads for v1.20.0-alpha.2

Source Code

filename	sha512 hash
kubernetes.tar.gz	45089a4d26d56a5d613ecbea64e356869ac738eca3cc71d16b74ea8ae
kubernetes-src.tar.gz	646edd890d6df5858b90aaf68cc6e1b4589b8db09396ae921b5c400f2

Client binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	c136273883e24a2a50b5093b9654f01cdfe57b97461d34885af4a68c
kubernetes-client-linux-386.tar.gz	6ec59f1ed30569fa64ddb2d0de32b1ae04cda4ffe13f339050a7c9d7c
kubernetes-client-linux-amd64.tar.gz	7b40a4c087e2ea7f8d055f297fcd39a3f1cb6c866e7a3981a9408c3c3
kubernetes-client-linux-arm.tar.gz	cda9955feebea5acb8f2b5b87895d24894bbbbde47041453b1f926eb
kubernetes-client-linux-arm64.tar.gz	f65bd9241c7eb88a4886a285330f732448570aea4ededaebeabcf70d
kubernetes-client-linux-ppc64le.tar.gz	1e377599af100a81d027d9199365fb8208d443a8e0a97affff1a79dc1
kubernetes-client-linux-s390x.tar.gz	1cdee81478246aa7e7b80ae4efc7f070a5b058083ae278f59fad088b7
kubernetes-client-windows-386.tar.gz	d8774167c87b6844c348aa15e92d5033c528d6ab9e95d08a7cb22da
kubernetes-client-windows-amd64.tar.gz	f664b47d8daa6036f8154c1dc1f881bfe683bf57c39d9b491de3848c0

Server binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	d6fcb4600be0beb9de222a8da64c35fe22798a0da82d41401d34d0f0fc7e2
kubernetes-server-linux-arm.tar.gz	022a76cf10801f8afbabb509572479b68fdb4e683526fa0799cdbd9bab4d3

filename	sha512 hash
kubernetes-server-linux-arm64.tar.gz	0679aadd60bbf6f607e5befad74b5267eb2d4c1b55985cc25a97e0f4c5efb
kubernetes-server-linux-ppc64le.tar.gz	9f2cfeed543b515eafb60d9765a3afff4f3d323c0a5c8a0d75e3de25985b26
kubernetes-server-linux-s390x.tar.gz	937258704d7b9dcd91f35f2d34ee9dd38c18d9d4e867408c05281bfbbb91

Node binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	076165d745d47879de68f4404eaf432920884be48277eb409e84bf2c6175
kubernetes-node-linux-arm.tar.gz	1ff2e2e3e43af41118cdfb70c778e15035bbb1aca833ffd2db83c4bcd44f55
kubernetes-node-linux-arm64.tar.gz	b232c7359b8c635126899beee76998078eec7a1ef6758d92bcdebe8013b0
kubernetes-node-linux-ppc64le.tar.gz	51d415a068f554840f4c78d11a4fedebd7cb03c686b0ec864509b24f7a866
kubernetes-node-linux-s390x.tar.gz	b51c082d8af358233a088b632cf2f6c8cfe5421471c27f5dc9ba4839ae6ea
kubernetes-node-windows-amd64.tar.gz	91b9d26620a2dde67a0edead0039814efccbdfd54594dda3597aaced6d89

Changelog since v1.20.0-alpha.1

Changes by Kind

Deprecation

- Action-required: kubeadm: graduate the "kubeadm alpha certs" command to a parent command "kubeadm certs". The command "kubeadm alpha certs" is deprecated and will be removed in a future release. Please migrate. (#94938, @yagonobre) [SIG Cluster Lifecycle]
- Action-required: kubeadm: remove the deprecated feature --experimental-kustomize from kubeadm commands. The feature was replaced with --experimental-patches in 1.19. To migrate see the --help

description for the --experimental-patches flag. (#94871, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: deprecate self-hosting support. The experimental command "kubeadm alpha self-hosting" is now deprecated and will be removed in a future release. (#95125, @neolit123) [SIG Cluster Lifecycle]
- Removes deprecated scheduler metrics DeprecatedSchedulingDuration, DeprecatedSchedulingAlgorithmPredicateEvaluationSecondsDuration, DeprecatedSchedulingAlgorithmPriorityEvaluationSecondsDuration (#94884, @arghya88) [SIG Instrumentation and Scheduling]
- Scheduler alpha metrics binding_duration_seconds and scheduling_algorithm_preemption_evaluation_seconds are deprecated, Both of those metrics are now covered as part of framework_extension_point_duration_seconds, the former as a PostFilter the latter and a Bind plugin. The plan is to remove both in 1.21 (#95001, @arghya88) [SIG Instrumentation and Scheduling]

API Change

- GPU metrics provided by kubelet are now disabled by default (#95184, @RenaudWasTaken) [SIG Node]
- New parameter `defaultingType` for `PodTopologySpread` plugin allows to use k8s defined or user provided default constraints (#95048, @alculquicondor) [SIG Scheduling]
- Server Side Apply now treats LabelSelector fields as atomic (meaning the entire selector is managed by a single writer and updated together), since they contain interrelated and inseparable fields that do not merge in intuitive ways. (#93901, @jpbetz) [SIG API Machinery, Auth, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation, Network, Node, Storage and Testing]
- Status of v1beta1 CRDs without "preserveUnknownFields:false" will show violation "spec.preserveUnknownFields: Invalid value: true: must be false" (#93078, @vareti) [SIG API Machinery]

Feature

- Added `get-users` and `delete-user` to the `kubectl config` subcommand (#89840, @eddiezane) [SIG CLI]

- Added counter metric "apiserver_request_self" to count API server self-requests with labels for verb, resource, and subresource. (#94288, @LogicalShark) [SIG API Machinery, Auth, Instrumentation and Scheduling]

- Added new k8s.io/component-helpers repository providing shared helper code for (core) components. (#92507, @ingvagabund) [SIG Apps, Node, Release and Scheduling]

- Adds `create ingress` command to `kubectl` (#78153, @amimof) [SIG CLI and Network]

- Allow configuring AWS LoadBalancer health check protocol via service annotations (#94546, @kishorj) [SIG Cloud Provider]
- Azure: Support multiple services sharing one IP address (#94991, @nilo19) [SIG Cloud Provider]
- Ephemeral containers now apply the same API defaults as initContainers and containers (#94896, @wawa0210) [SIG Apps and CLI]
- In dual-stack bare-metal clusters, you can now pass dual-stack IPs to `kubelet --node-ip`. eg: `kubelet --node-ip 10.1.0.5,fd01::0005`. This is not yet supported for non-bare-metal clusters.

  In dual-stack clusters where nodes have dual-stack addresses, hostNetwork pods will now get dual-stack PodIPs. (#95239, @danwinship) [SIG Network and Node]
- Introduces a new GCE specific cluster creation variable KUBE_PROXY_DISABLE. When set to true, this will skip over the creation of kube-proxy (whether the daemonset or static pod). This can be used to control the lifecycle of kube-proxy separately from the lifecycle of the nodes. (#91977, @varunmar) [SIG Cloud Provider]
- Kubeadm: do not throw errors if the current system time is outside of the NotBefore and NotAfter bounds of a loaded certificate. Print warnings instead. (#94504, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: make the command "kubeadm alpha kubeconfig user" accept a "--config" flag and remove the following flags:
  - apiserver-advertise-address / apiserver-bind-port: use either localAPIEndpoint from InitConfiguration or controlPlaneEndpoint from ClusterConfiguration.
  - cluster-name: use clusterName from ClusterConfiguration
  - cert-dir: use certificatesDir from ClusterConfiguration (#94879, @knight42) [SIG Cluster Lifecycle]
- Kubectl rollout history sts/sts-name --revision=some-revision will start showing the detailed view of the sts on that specified revision (#86506, @dineshba) [SIG CLI]
- Scheduling Framework: expose Run[Pre]ScorePlugins functions to PreemptionHandle which can be used in PostFilter extention point. (#93534, @everpeace) [SIG Scheduling and Testing]
- Send gce node startup scripts logs to console and journal (#95311, @karan) [SIG Cloud Provider and Node]
- Support kubectl delete orphan/foreground/background options (#93384, @zhouya0) [SIG CLI and Testing]

Bug or Regression

- Change the mount way from systemd to normal mount except ceph and glusterfs intree-volume. (#94916, @smileusd) [SIG Apps, Cloud Provider, Network, Node, Storage and Testing]
- Cloud node controller: handle empty providerID from getProviderID (#95342, @nicolehanjing) [SIG Cloud Provider]
- Fix a bug where the endpoint slice controller was not mirroring the parent service labels to its corresponding endpoint slices (#94443, @aojea) [SIG Apps and Network]
- Fix azure disk attach failure for disk size bigger than 4TB (#95463, @andyzhangx) [SIG Cloud Provider]
- Fix azure disk data loss issue on Windows when unmount disk (#95456, @andyzhangx) [SIG Cloud Provider and Storage]
- Fix detach azure disk issue when vm not exist (#95177, @andyzhangx) [SIG Cloud Provider]
- Fix network_programming_latency metric reporting for Endpoints/EndpointSlice deletions, where we don't have correct timestamp (#95363, @wojtek-t) [SIG Network and Scalability]
- Fix scheduler cache snapshot when a Node is deleted before its Pods (#95130, @alculquicondor) [SIG Scheduling]
- Fix vsphere detach failure for static PVs (#95447, @gnufied) [SIG Cloud Provider and Storage]
- Fixed a bug that prevents the use of ephemeral containers in the presence of a validating admission webhook. (#94685, @verb) [SIG Node and Testing]
- Gracefully delete nodes when their parent scale set went missing (#95289, @bpineau) [SIG Cloud Provider]
- In dual-stack clusters, kubelet will now set up both IPv4 and IPv6 iptables rules, which may fix some problems, eg with HostPorts. (#94474, @danwinship) [SIG Network and Node]
- Kubeadm: for Docker as the container runtime, make the "kubeadm reset" command stop containers before removing them (#94586, @BedivereZero) [SIG Cluster Lifecycle]
- Kubeadm: warn but do not error out on missing "ca.key" files for root CA, front-proxy CA and etcd CA, during "kubeadm join --control-plane" if the user has provided all certificates, keys and kubeconfig files which require signing with the given CA keys. (#94988, @neolit123) [SIG Cluster Lifecycle]
- Port mapping allows to map the same `containerPort` to multiple `hostPort` without naming the mapping explicitly. (#94494, @SergeyKanzhelev) [SIG Network and Node]
- Warn instead of fail when creating Roles and ClusterRoles with custom verbs via kubectl (#92492, @eddiezane) [SIG CLI]

Other (Cleanup or Flake)

- Added fine grained debugging to the intra-pod conformance test for helping easily resolve networking issues for nodes that might be unhealthy when running conformance or sonobuoy tests. (#93837, @jayunit100) [SIG Network and Testing]

- AdmissionReview objects sent for the creation of Namespace API objects now populate the `namespace` attribute consistently (previously the `namespace` attribute was empty for Namespace creation via POST requests, and populated for Namespace creation via server-side-apply PATCH requests) (#95012, @nodo) [SIG API Machinery and Testing]
- Client-go header logging (at verbosity levels >= 9) now masks `Authorization` header contents (#95316, @sfowl) [SIG API Machinery]
- Enhance log information of verifyRunAsNonRoot, add pod, container information (#94911, @wawa0210) [SIG Node]
- Errors from staticcheck: vendor/k8s.io/client-go/discovery/cached/memory/memcache_test.go: 94:2: this value of g is never used (SA4006) (#95098, @phunziker) [SIG API Machinery]
- Kubeadm: update the default pause image version to 1.4.0 on Windows. With this update the image supports Windows versions 1809 (2019LTS), 1903, 1909, 2004 (#95419, @jsturtevant) [SIG Cluster Lifecycle and Windows]
- Masks ceph RBD adminSecrets in logs when logLevel >= 4 (#95245, @sfowl) [SIG Storage]
- Upgrade snapshot controller to 3.0.0 (#95412, @saikat-royc) [SIG Cloud Provider]
- Remove offensive words from kubectl cluster-info command (#95202, @rikatz) [SIG Architecture, CLI and Testing]
- The following new metrics are available.
  - network_plugin_operations_total
  - network_plugin_operations_errors_total (#93066, @AnishShah) [SIG Instrumentation, Network and Node]
- Vsphere: improve logging message on node cache refresh event (#95236, @andrewsykim) [SIG Cloud Provider]
- `kubectl api-resources` now prints the API version (as 'API group/version', same as output of `kubectl api-versions`). The column APIGROUP is now APIVERSION (#95253, @sallyom) [SIG CLI]

Dependencies

Added

- github.com/jmespath/go-jmespath/internal/testify: v1.5.1

Changed

- github.com/aws/aws-sdk-go: v1.28.2 â†’ v1.35.5
- github.com/jmespath/go-jmespath: c2b33e8 â†’ v0.4.0
- k8s.io/kube-openapi: 6aeccd4 â†’ 8b50664
- sigs.k8s.io/apiserver-network-proxy/konnectivity-client: v0.0.9 â†’ v0.0.12
- sigs.k8s.io/structured-merge-diff/v4: v4.0.1 â†’ b3cf1e8

Removed

Nothing has changed.

v1.20.0-alpha.1

Downloads for v1.20.0-alpha.1

Source Code

filename	sha512 hash
kubernetes.tar.gz	e7daed6502ea07816274f2371f96fe1a446d0d7917df4454b722d9eb3b
kubernetes-src.tar.gz	e91213a0919647a1215d4691a63b12d89a3e74055463a8ebd71dc1a4

Client binaries

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	1f3add5f826fa989820d715ca38e8864b66f30b59c1abeacbb4bfb96b
kubernetes-client-linux-386.tar.gz	c62acdc8993b0a950d4b0ce0b45473bf96373d501ce61c88adf4007a
kubernetes-client-linux-amd64.tar.gz	1203ababfe00f9bc5be5c059324c17160a96530c1379a152db33564b
kubernetes-client-linux-arm.tar.gz	31860088596e12d739c7aed94556c2d1e217971699b950c8417a3ce
kubernetes-client-linux-arm64.tar.gz	8d469f37fe20d6e15b5debc13cce4c22e8b7a4f6a4ac787006b96507a
kubernetes-client-linux-ppc64le.tar.gz	0d62ee1729cd5884946b6c73701ad3a570fa4d642190ca0fe5c1db0f
kubernetes-client-linux-s390x.tar.gz	0fc0420e134ec0b8e0ab2654e1e102cebec47b48179703f1e1b79d51
kubernetes-client-windows-386.tar.gz	3fb53b5260f4888c77c0e4ff602bbcf6bf38c364d2769850afe2b8d8e8
kubernetes-client-windows-amd64.tar.gz	2f44c93463d6b5244ce0c82f147e7f32ec2233d0e29c64c3c5759e235

Server binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	ae82d14b1214e4100f0cc2c988308b3e1edd040a65267d0eddb9082409f7

filename	sha512 hash
kubernetes-server-linux-arm.tar.gz	9a2a5828b7d1ddb16cc19d573e99a4af642f84129408e6203eeeb0558e7b
kubernetes-server-linux-arm64.tar.gz	ed700dd226c999354ce05b73927388d36d08474c15333ae689427de15de
kubernetes-server-linux-ppc64le.tar.gz	abb7a9d726538be3ccf5057a0c63ff9732b616e213c6ebb81363f0c49f1e1
kubernetes-server-linux-s390x.tar.gz	3a51888af1bfdd2d5b0101d173ee589c1f39240e4428165f5f85c610344db

Node binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	d0f28e3c38ca59a7ff1bfecb48a1ce97116520355d9286afdca1200d346c1
kubernetes-node-linux-arm.tar.gz	ed9d3f13028beb3be39bce980c966f82c4b39dc73beaae38cc075fea5be30
kubernetes-node-linux-arm64.tar.gz	ad5b3268db365dcdded9a9a4bffc90c7df0f844000349accdf2b8fb5f1081e
kubernetes-node-linux-ppc64le.tar.gz	c4de2524e513996def5eeba7b83f7b406f17eaf89d4d557833a93bd035348
kubernetes-node-linux-s390x.tar.gz	9157b44e3e7bd5478af9f72014e54d1afa5cd19b984b4cd8b348b312c385
kubernetes-node-windows-amd64.tar.gz	8b40a43c5e6447379ad2ee8aac06e8028555e1b370a995f6001018a6241

Changelog since v1.20.0-alpha.0

Urgent Upgrade Notes

(No, really, you MUST read this before you upgrade)

- Azure blob disk feature(`kind`: `Shared`, `Dedicated`) has been deprecated, you should use `kind`: `Managed` in `kubernetes.io/azure-disk` storage class. (#92905, @andyzhangx) [SIG Cloud Provider and Storage]
- CVE-2020-8559 (Medium): Privilege escalation from compromised node to cluster. See https://github.com/kubernetes/kubernetes/issues/92914

for more details. The API Server will no longer proxy non-101 responses for upgrade requests. This could break proxied backends (such as an extension API server) that respond to upgrade requests with a non-101 response code. (#92941, @tallclair) [SIG API Machinery]

Changes by Kind

Deprecation

- Kube-apiserver: the componentstatus API is deprecated. This API provided status of etcd, kube-scheduler, and kube-controller-manager components, but only worked when those components were local to the API server, and when kube-scheduler and kube-controller-manager exposed unsecured health endpoints. Instead of this API, etcd health is included in the kube-apiserver health check and kube-scheduler/kube-controller-manager health checks can be made directly against those components' health endpoints. (#93570, @liggitt) [SIG API Machinery, Apps and Cluster Lifecycle]
- Kubeadm: deprecate the "kubeadm alpha kubelet config enable-dynamic" command. To continue using the feature please defer to the guide for "Dynamic Kubelet Configuration" at k8s.io. (#92881, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: remove the deprecated "kubeadm alpha kubelet config enable-dynamic" command. To continue using the feature please defer to the guide for "Dynamic Kubelet Configuration" at k8s.io. This change also removes the parent command "kubeadm alpha kubelet" as there are no more sub-commands under it for the time being. (#94668, @neolit123) [SIG Cluster Lifecycle]
- Kubeadm: remove the deprecated --kubelet-config flag for the command "kubeadm upgrade node" (#94869, @neolit123) [SIG Cluster Lifecycle]
- Kubelet's deprecated endpoint `metrics/resource/v1alpha1` has been removed, please adopt to `metrics/resource`. (#94272, @RainbowMango) [SIG Instrumentation and Node]
- The v1alpha1 PodPreset API and admission plugin has been removed with no built-in replacement. Admission webhooks can be used to modify pods on creation. (#94090, @deads2k) [SIG API Machinery, Apps, CLI, Cloud Provider, Scalability and Testing]

API Change

- A new `nofuzz` go build tag now disables gofuzz support. Release binaries enable this. (#92491, @BenTheElder) [SIG API Machinery]
- A new alpha-level field, `SupportsFsGroup`, has been introduced for CSIDrivers to allow them to specify whether they support volume ownership and permission modifications. The `CSIVolumeSupportFSGroup` feature gate must be enabled to allow this field to be used. (#92001, @huffmanca) [SIG API Machinery, CLI and Storage]
- Added pod version skew strategy for seccomp profile to synchronize the deprecated annotations with the new API Server fields. Please see the

corresponding section in the KEP for more detailed explanations. (#91408, @saschagrunert) [SIG Apps, Auth, CLI and Node]
- Adds the ability to disable Accelerator/GPU metrics collected by Kubelet (#91930, @RenaudWasTaken) [SIG Node]
- Custom Endpoints are now mirrored to EndpointSlices by a new EndpointSliceMirroring controller. (#91637, @robscott) [SIG API Machinery, Apps, Auth, Cloud Provider, Instrumentation, Network and Testing]
- External facing API podresources is now available under k8s.io/kubelet/pkg/apis/ (#92632, @RenaudWasTaken) [SIG Node and Testing]
- Fix conversions for custom metrics. (#94481, @wojtek-t) [SIG API Machinery and Instrumentation]
- Generic ephemeral volumes, a new alpha feature under the `GenericEphemeralVolume` feature gate, provide a more flexible alternative to `EmptyDir` volumes: as with `EmptyDir`, volumes are created and deleted for each pod automatically by Kubernetes. But because the normal provisioning process is used (`PersistentVolumeClaim`), storage can be provided by third-party storage vendors and all of the usual volume features work. Volumes don't need to be empt; for example, restoring from snapshot is supported. (#92784, @pohly) [SIG API Machinery, Apps, Auth, CLI, Instrumentation, Node, Scheduling, Storage and Testing]
- Kube-controller-manager: volume plugins can be restricted from contacting local and loopback addresses by setting `--volume-host-allow-local-loopback=false`, or from contacting specific CIDR ranges by setting `--volume-host-cidr-denylist` (for example, `--volume-host-cidr-denylist=127.0.0.1/28,feed::/16`) (#91785, @mattcary) [SIG API Machinery, Apps, Auth, CLI, Network, Node, Storage and Testing]
- Kubernetes is now built with golang 1.15.0-rc.1.
  - The deprecated, legacy behavior of treating the CommonName field on X.509 serving certificates as a host name when no Subject Alternative Names are present is now disabled by default. It can be temporarily re-enabled by adding the value x509ignoreCN=0 to the GODEBUG environment variable. (#93264, @justaugustus) [SIG API Machinery, Auth, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation, Network, Node, Release, Scalability, Storage and Testing]
- Migrate scheduler, controller-manager and cloud-controller-manager to use LeaseLock (#94603, @wojtek-t) [SIG API Machinery, Apps, Cloud Provider and Scheduling]
- Modify DNS-1123 error messages to indicate that RFC 1123 is not followed exactly (#94182, @mattfenwick) [SIG API Machinery, Apps, Auth, Network and Node]
- The ServiceAccountIssuerDiscovery feature gate is now Beta and enabled by default. (#91921, @mtaufen) [SIG Auth]
- The kube-controller-manager managed signers can now have distinct signing certificates and keys. See the help about `--cluster-signing-[signer-name]-{cert,key}-file`. `--cluster-signing-{cert,key}-file` is still the default. (#90822, @deads2k) [SIG API Machinery, Apps and Auth]

- When creating a networking.k8s.io/v1 Ingress API object, `spec.tls[*].secretName` values are required to pass validation rules for Secret API object names. (#93929, @liggitt) [SIG Network]
- WinOverlay feature graduated to beta (#94807, @ksubrmnn) [SIG Windows]

Feature

- ACTION REQUIRED : In CoreDNS v1.7.0, metrics names have been changed which will be backward incompatible with existing reporting formulas that use the old metrics' names. Adjust your formulas to the new names before upgrading.

  Kubeadm now includes CoreDNS version v1.7.0. Some of the major changes include:

  - Fixed a bug that could cause CoreDNS to stop updating service records.
  - Fixed a bug in the forward plugin where only the first upstream server is always selected no matter which policy is set.
  - Remove already deprecated options `resyncperiod` and `upstream` in the Kubernetes plugin.
  - Includes Prometheus metrics name changes (to bring them in line with standard Prometheus metrics naming convention). They will be backward incompatible with existing reporting formulas that use the old metrics' names.
  - The federation plugin (allows for v1 Kubernetes federation) has been removed. More details are available in https://coredns.io/2020/06/15/coredns-1.7.0-release/ (#92651, @rajansandeep) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle and Instrumentation]

- Add metrics for azure service operations (route and loadbalancer). (#94124, @nilo19) [SIG Cloud Provider and Instrumentation]

- Add network rule support in Azure account creation (#94239, @andyzhangx) [SIG Cloud Provider]

- Add tags support for Azure File Driver (#92825, @ZeroMagic) [SIG Cloud Provider and Storage]

- Added kube-apiserver metrics: apiserver_current_inflight_request_measures and, when API Priority and Fairness is enable, windowed_request_stats. (#91177, @MikeSpreitzer) [SIG API Machinery, Instrumentation and Testing]

- Audit events for API requests to deprecated API versions now include a `"k8s.io/deprecated": "true"` audit annotation. If a target removal release is identified, the audit event includes a `"k8s.io/removal-release": "<majorVersion>.<minorVersion>"` audit annotation as well. (#92842, @liggitt) [SIG API Machinery and Instrumentation]

- Cloud node-controller use InstancesV2 (#91319, @gongguan) [SIG Apps, Cloud Provider, Scalability and Storage]
- Kubeadm: Add a preflight check that the control-plane node has at least 1700MB of RAM (#93275, @xlgao-zju) [SIG Cluster Lifecycle]
- Kubeadm: add the "--cluster-name" flag to the "kubeadm alpha kubeconfig user" to allow configuring the cluster name in the generated kubeconfig file (#93992, @prabhu43) [SIG Cluster Lifecycle]
- Kubeadm: add the "--kubeconfig" flag to the "kubeadm init phase upload-certs" command to allow users to pass a custom location for a kubeconfig file. (#94765, @zhanw15) [SIG Cluster Lifecycle]
- Kubeadm: deprecate the "--csr-only" and "--csr-dir" flags of the "kubeadm init phase certs" subcommands. Please use "kubeadm alpha certs generate-csr" instead. This new command allows you to generate new private keys and certificate signing requests for all the control-plane components, so that the certificates can be signed by an external CA. (#92183, @wallrj) [SIG Cluster Lifecycle]
- Kubeadm: make etcd pod request 100m CPU, 100Mi memory and 100Mi ephemeral_storage by default (#94479, @knight42) [SIG Cluster Lifecycle]
- Kubemark now supports both real and hollow nodes in a single cluster. (#93201, @ellistarn) [SIG Scalability]
- Kubernetes is now built using go1.15.2
  - build: Update to k/repo-infra@v0.1.1 (supports go1.15.2)
  - build: Use go-runner:buster-v2.0.1 (built using go1.15.1)
  - bazel: Replace --features with Starlark build settings flag
  - hack/lib/util.sh: some bash cleanups
    - switched one spot to use kube::logging
    - make kube::util::find-binary return an error when it doesn't find anything so that hack scripts fail fast instead of with '' binary not found errors.
    - this required deleting some genfeddoc stuff. the binary no longer exists in k/k repo since we removed federation/, and I don't see it in https://github.com/kubernetes-sigs/kubefed/ either. I'm assuming that it's gone for good now.
  - bazel: output go_binary rule directly from go_binary_conditional_pure

    From: @mikedanese: Instead of aliasing. Aliases are annoying in a number of ways. This is specifically bugging me now because they make the action graph harder to analyze programmatically. By

using aliases here, we would need to handle potentially aliased go_binary targets and dereference to the effective target.

The comment references an issue with `pure = select(...)` which appears to be resolved considering this now builds.

- make kube::util::find-binary not dependent on bazel-out/ structure

  Implement an aspect that outputs go_build_mode metadata for go binaries, and use that during binary selection. (#94449, @justaugustus) [SIG Architecture, CLI, Cluster Lifecycle, Node, Release and Testing]

- Only update Azure data disks when attach/detach (#94265, @andyzhangx) [SIG Cloud Provider]

- Promote SupportNodePidsLimit to GA to provide node to pod pid isolation Promote SupportPodPidsLimit to GA to provide ability to limit pids per pod (#94140, @derekwaynecarr) [SIG Node and Testing]

- Rename pod_preemption_metrics to preemption_metrics. (#93256, @ahg-g) [SIG Instrumentation and Scheduling]

- Server-side apply behavior has been regularized in the case where a field is removed from the applied configuration. Removed fields which have no other owners are deleted from the live object, or reset to their default value if they have one. Safe ownership transfers, such as the transfer of a `replicas` field from a user to an HPA without resetting to the default value are documented in Transferring Ownership (#92661, @jpbetz) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Testing]

- Set CSIMigrationvSphere feature gates to beta. Users should enable CSIMigration + CSIMigrationvSphere features and install the vSphere CSI Driver (https://github.com/kubernetes-sigs/vsphere-csi-driver) to move workload from the in-tree vSphere plugin "kubernetes.io/vsphere-volume" to vSphere CSI Driver.

  Requires: vSphere vCenter/ESXi Version: 7.0u1, HW Version: VM version 15 (#92816, @divyenpatel) [SIG Cloud Provider and Storage]

- Support [service.beta.kubernetes.io/azure-pip-ip-tags] annotations to allow customers to specify ip-tags to influence public-ip creation in Azure [Tag1=Value1, Tag2=Value2, etc.] (#94114, @MarcPow) [SIG Cloud Provider]

- Support a smooth upgrade from client-side apply to server-side apply without conflicts, as well as support the corresponding downgrade. (#90187, @julianvmodesto) [SIG API Machinery and Testing]

- Trace output in apiserver logs is more organized and comprehensive. Traces are nested, and for all non-long running request endpoints, the entire filter chain is instrumented (e.g. authentication check is

included). (#88936, @jpbetz) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Scheduling]

- `kubectl alpha debug` now supports debugging nodes by creating a debugging container running in the node's host namespaces. (#92310, @verb) [SIG CLI]

Documentation

- Kubelet: remove alpha warnings for CNI flags. (#94508, @andrewsykim) [SIG Network and Node]

Failing Test

- Kube-proxy iptables min-sync-period defaults to 1 sec. Previously, it was 0. (#92836, @aojea) [SIG Network]

Bug or Regression

- A panic in the apiserver caused by the `informer-sync` health checker is now fixed. (#93600, @ialidzhikov) [SIG API Machinery]
- Add kubectl wait --ignore-not-found flag (#90969, @zhouya0) [SIG CLI]
- Adding fix to the statefulset controller to wait for pvc deletion before creating pods. (#93457, @ymmt2005) [SIG Apps]
- Azure ARM client: don't segfault on empty response and http error (#94078, @bpineau) [SIG Cloud Provider]
- Azure: fix a bug that kube-controller-manager would panic if wrong Azure VMSS name is configured (#94306, @knight42) [SIG Cloud Provider]
- Azure: per VMSS VMSS VMs cache to prevent throttling on clusters having many attached VMSS (#93107, @bpineau) [SIG Cloud Provider]
- Both apiserver_request_duration_seconds metrics and RequestReceivedTimestamp field of an audit event take into account the time a request spends in the apiserver request filters. (#94903, @tkashem) [SIG API Machinery, Auth and Instrumentation]
- Build/lib/release: Explicitly use '--platform' in building server images

  When we switched to go-runner for building the apiserver, controller-manager, and scheduler server components, we no longer reference the individual architectures in the image names, specifically in the 'FROM' directive of the server image Dockerfiles.

  As a result, server images for non-amd64 images copy in the go-runner amd64 binary instead of the go-runner that matches that architecture.

This commit explicitly sets the '--platform=linux/${arch}' to ensure we're pulling the correct go-runner arch from the manifest list.

Before: `FROM ${base_image}`

After: `FROM --platform=linux/${arch} ${base_image}` (#94552, @justaugustus) [SIG Release]

- CSIDriver object can be deployed during volume attachment. (#93710, @Jiawei0227) [SIG Apps, Node, Storage and Testing]
- CVE-2020-8557 (Medium): Node-local denial of service via container / etc/hosts file. See https://github.com/kubernetes/kubernetes/issues/93032 for more details. (#92916, @joelsmith) [SIG Node]
- Do not add nodes labeled with kubernetes.azure.com/managed=false to backend pool of load balancer. (#93034, @matthias50) [SIG Cloud Provider]
- Do not fail sorting empty elements. (#94666, @soltysh) [SIG CLI]
- Do not retry volume expansion if CSI driver returns FailedPrecondition error (#92986, @gnufied) [SIG Node and Storage]
- Dockershim security: pod sandbox now always run with `no-new-privileges` and `runtime/default` seccomp profile dockershim seccomp: custom profiles can now have smaller seccomp profiles when set at pod level (#90948, @pjbgf) [SIG Node]
- Dual-stack: make nodeipam compatible with existing single-stack clusters when dual-stack feature gate become enabled by default (#90439, @SataQiu) [SIG API Machinery]
- Endpoint controller requeues service after an endpoint deletion event occurs to confirm that deleted endpoints are undesired to mitigate the effects of an out of sync endpoint cache. (#93030, @swetharepakula) [SIG Apps and Network]
- EndpointSlice controllers now return immediately if they encounter an error creating, updating, or deleting resources. (#93908, @robscott) [SIG Apps and Network]
- EndpointSliceMirroring controller now copies labels from Endpoints to EndpointSlices. (#93442, @robscott) [SIG Apps and Network]
- EndpointSliceMirroring controller now mirrors Endpoints that do not have a Service associated with them. (#94171, @robscott) [SIG Apps, Network and Testing]
- Ensure backoff step is set to 1 for Azure armclient. (#94180, @feiskyer) [SIG Cloud Provider]
- Ensure getPrimaryInterfaceID not panic when network interfaces for Azure VMSS are null (#94355, @feiskyer) [SIG Cloud Provider]

- Eviction requests for pods that have a non-zero DeletionTimestamp will always succeed (#91342, @michaelgugino) [SIG Apps]
- Extended DSR loadbalancer feature in winkernel kube-proxy to HNS versions 9.3-9.max, 10.2+ (#93080, @elweb9858) [SIG Network]
- Fix HandleCrash order (#93108, @lixiaobing1) [SIG API Machinery]
- Fix a concurrent map writes error in kubelet (#93773, @knight42) [SIG Node]
- Fix a regression where kubeadm bails out with a fatal error when an optional version command line argument is supplied to the "kubeadm upgrade plan" command (#94421, @rosti) [SIG Cluster Lifecycle]
- Fix azure file migration panic (#94853, @andyzhangx) [SIG Cloud Provider]
- Fix bug where loadbalancer deletion gets stuck because of missing resource group #75198 (#93962, @phiphi282) [SIG Cloud Provider]
- Fix calling AttachDisk on a previously attached EBS volume (#93567, @gnufied) [SIG Cloud Provider, Storage and Testing]
- Fix detection of image filesystem, disk metrics for devicemapper, detection of OOM Kills on 5.0+ linux kernels. (#92919, @dashpole) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Node]
- Fix etcd_object_counts metric reported by kube-apiserver (#94773, @tkashem) [SIG API Machinery]
- Fix incorrectly reported verbs for kube-apiserver metrics for CRD objects (#93523, @wojtek-t) [SIG API Machinery and Instrumentation]
- Fix instance not found issues when an Azure Node is recreated in a short time (#93316, @feiskyer) [SIG Cloud Provider]
- Fix kube-apiserver /readyz to contain "informer-sync" check ensuring that internal informers are synced. (#93670, @wojtek-t) [SIG API Machinery and Testing]
- Fix kubectl SchemaError on CRDs with schema using x-kubernetes-preserve-unknown-fields on array types. (#94888, @sttts) [SIG API Machinery]
- Fix memory leak in EndpointSliceTracker for EndpointSliceMirroring controller. (#93441, @robscott) [SIG Apps and Network]
- Fix missing csi annotations on node during parallel csinode update. (#94389, @pacoxu) [SIG Storage]
- Fix the `cloudprovider_azure_api_request_duration_seconds` metric buckets to correctly capture the latency metrics. Previously, the

majority of the calls would fall in the "+Inf" bucket. (#94873, @marwanad) [SIG Cloud Provider and Instrumentation]

- Fix: azure disk resize error if source does not exist (#93011, @andyzhangx) [SIG Cloud Provider]
- Fix: detach azure disk broken on Azure Stack (#94885, @andyzhangx) [SIG Cloud Provider]
- Fix: determine the correct ip config based on ip family (#93043, @aramase) [SIG Cloud Provider]
- Fix: initial delay in mounting azure disk & file (#93052, @andyzhangx) [SIG Cloud Provider and Storage]
- Fix: use sensitiveOptions on Windows mount (#94126, @andyzhangx) [SIG Cloud Provider and Storage]
- Fixed Ceph RBD volume expansion when no ceph.conf exists (#92027, @juliantaylor) [SIG Storage]
- Fixed a bug where improper storage and comparison of endpoints led to excessive API traffic from the endpoints controller (#94112, @damemi) [SIG Apps, Network and Testing]
- Fixed a bug whereby the allocation of reusable CPUs and devices was not being honored when the TopologyManager was enabled (#93189, @klueska) [SIG Node]
- Fixed a panic in kubectl debug when pod has multiple init containers or ephemeral containers (#94580, @kiyoshim55) [SIG CLI]
- Fixed a regression that sometimes prevented `kubectl portforward` to work when TCP and UDP services were configured on the same port (#94728, @amorenoz) [SIG CLI]
- Fixed bug in reflector that couldn't recover from "Too large resource version" errors with API servers 1.17.0-1.18.5 (#94316, @janeczku) [SIG API Machinery]
- Fixed bug where kubectl top pod output is not sorted when --sort-by and --containers flags are used together (#93692, @brianpursley) [SIG CLI]
- Fixed kubelet creating extra sandbox for pods with RestartPolicyOnFailure after all containers succeeded (#92614, @tnqn) [SIG Node and Testing]
- Fixed memory leak in endpointSliceTracker (#92838, @tnqn) [SIG Apps and Network]
- Fixed node data lost in kube-scheduler for clusters with imbalance on number of nodes across zones (#93355, @maelk) [SIG Scheduling]

- Fixed the EndpointSliceController to correctly create endpoints for IPv6-only pods.

  Fixed the EndpointController to allow IPv6 headless services, if the IPv6DualStack feature gate is enabled, by specifying `ipFamily: IPv6` on the service. (This already worked with the EndpointSliceController.) (#91399, @danwinship) [SIG Apps and Network]

- Fixes a bug evicting pods after a taint with a limited tolerationSeconds toleration is removed from a node (#93722, @liggitt) [SIG Apps and Node]

- Fixes a bug where EndpointSlices would not be recreated after rapid Service recreation. (#94730, @robscott) [SIG Apps, Network and Testing]

- Fixes a race condition in kubelet pod handling (#94751, @auxten) [SIG Node]

- Fixes an issue proxying to ipv6 pods without specifying a port (#94834, @liggitt) [SIG API Machinery and Network]

- Fixes an issue that can result in namespaced custom resources being orphaned when their namespace is deleted, if the CRD defining the custom resource is removed concurrently with namespaces being deleted, then recreated. (#93790, @liggitt) [SIG API Machinery and Apps]

- Ignore root user check when windows pod starts (#92355, @wawa0210) [SIG Node and Windows]

- Increased maximum IOPS of AWS EBS io1 volumes to 64,000 (current AWS maximum). (#90014, @jacobmarble) [SIG Cloud Provider and Storage]

- K8s.io/apimachinery: runtime.DefaultUnstructuredConverter.FromUnstructured now handles converting integer fields to typed float values (#93250, @liggitt) [SIG API Machinery]

- Kube-aggregator certificates are dynamically loaded on change from disk (#92791, @p0lyn0mial) [SIG API Machinery]

- Kube-apiserver: fixed a bug returning inconsistent results from list requests which set a field or label selector and set a paging limit (#94002, @wojtek-t) [SIG API Machinery]

- Kube-apiserver: jsonpath expressions with consecutive recursive descent operators are no longer evaluated for custom resource printer columns (#93408, @joelsmith) [SIG API Machinery]

- Kube-proxy now trims extra spaces found in loadBalancerSourceRanges to match Service validation. (#94107, @robscott) [SIG Network]

- Kube-up now includes CoreDNS version v1.7.0. Some of the major changes include:
  - Fixed a bug that could cause CoreDNS to stop updating service records.
  - Fixed a bug in the forward plugin where only the first upstream server is always selected no matter which policy is set.
  - Remove already deprecated options `resyncperiod` and `upstream` in the Kubernetes plugin.
  - Includes Prometheus metrics name changes (to bring them in line with standard Prometheus metrics naming convention). They will be backward incompatible with existing reporting formulas that use the old metrics' names.
  - The federation plugin (allows for v1 Kubernetes federation) has been removed. More details are available in https://coredns.io/2020/06/15/coredns-1.7.0-release/ (#92718, @rajansandeep) [SIG Cloud Provider]

- Kubeadm now makes sure the etcd manifest is regenerated upon upgrade even when no etcd version change takes place (#94395, @rosti) [SIG Cluster Lifecycle]

- Kubeadm: avoid a panic when determining if the running version of CoreDNS is supported during upgrades (#94299, @zouyee) [SIG Cluster Lifecycle]

- Kubeadm: ensure "kubeadm reset" does not unmount the root "/var/lib/kubelet" directory if it is mounted by the user (#93702, @thtanaka) [SIG Cluster Lifecycle]

- Kubeadm: ensure the etcd data directory is created with 0700 permissions during control-plane init and join (#94102, @neolit123) [SIG Cluster Lifecycle]

- Kubeadm: fix the bug that kubeadm tries to call 'docker info' even if the CRI socket was for another CR (#94555, @SataQiu) [SIG Cluster Lifecycle]

- Kubeadm: make the kubeconfig files for the kube-controller-manager and kube-scheduler use the LocalAPIEndpoint instead of the ControlPlaneEndpoint. This makes kubeadm clusters more reseliant to version skew problems during immutable upgrades: https://kubernetes.io/docs/setup/release/version-skew-policy/#kube-controller-manager-kube-scheduler-and-cloud-controller-manager (#94398, @neolit123) [SIG Cluster Lifecycle]

- Kubeadm: relax the validation of kubeconfig server URLs. Allow the user to define custom kubeconfig server URLs without erroring out during validation of existing kubeconfig files (e.g. when using external CA mode). (#94816, @neolit123) [SIG Cluster Lifecycle]

- Kubeadm: remove duplicate DNS names and IP addresses from generated certificates (#92753, @QianChenglong) [SIG Cluster Lifecycle]
- Kubelet: assume that swap is disabled when `/proc/swaps` does not exist (#93931, @SataQiu) [SIG Node]
- Kubelet: fix race condition in pluginWatcher (#93622, @knight42) [SIG Node]
- Kuberuntime security: pod sandbox now always runs with `runtime/default` seccomp profile kuberuntime seccomp: custom profiles can now have smaller seccomp profiles when set at pod level (#90949, @pjbgf) [SIG Node]
- NONE (#71269, @DeliangFan) [SIG Node]
- New Azure instance types do now have correct max data disk count information. (#94340, @ialidzhikov) [SIG Cloud Provider and Storage]
- Pods with invalid Affinity/AntiAffinity LabelSelectors will now fail scheduling when these plugins are enabled (#93660, @damemi) [SIG Scheduling]
- Require feature flag CustomCPUCFSQuotaPeriod if setting a non-default cpuCFSQuotaPeriod in kubelet config. (#94687, @karan) [SIG Node]
- Reverted devicemanager for Windows node added in 1.19rc1. (#93263, @liggitt) [SIG Node and Windows]
- Scheduler bugfix: Scheduler doesn't lose pod information when nodes are quickly recreated. This could happen when nodes are restarted or quickly recreated reusing a nodename. (#93938, @alculquicondor) [SIG Scalability, Scheduling and Testing]
- The EndpointSlice controller now waits for EndpointSlice and Node caches to be synced before starting. (#94086, @robscott) [SIG Apps and Network]
- The `/debug/api_priority_and_fairness/dump_requests` path at an apiserver will no longer return a phantom line for each exempt priority level. (#93406, @MikeSpreitzer) [SIG API Machinery]
- The kubelet recognizes the --containerd-namespace flag to configure the namespace used by cadvisor. (#87054, @changyaowei) [SIG Node]
- The terminationGracePeriodSeconds from pod spec is respected for the mirror pod. (#92442, @tedyu) [SIG Node and Testing]
- Update Calico to v3.15.2 (#94241, @lmm) [SIG Cloud Provider]
- Update default etcd server version to 3.4.13 (#94287, @jingyih) [SIG API Machinery, Cloud Provider, Cluster Lifecycle and Testing]

- Updated Cluster Autoscaler to 1.19.0; (#93577, @vivekbagade) [SIG Autoscaling and Cloud Provider]
- Use NLB Subnet CIDRs instead of VPC CIDRs in Health Check SG Rules (#93515, @t0rr3sp3dr0) [SIG Cloud Provider]
- Users will see increase in time for deletion of pods and also guarantee that removal of pod from api server would mean deletion of all the resources from container runtime. (#92817, @kmala) [SIG Node]
- Very large patches may now be specified to `kubectl patch` with the `--patch-file` flag instead of including them directly on the command line. The `--patch` and `--patch-file` flags are mutually exclusive. (#93548, @smarterclayton) [SIG CLI]
- When creating a networking.k8s.io/v1 Ingress API object, `spec.rules[*].http` values are now validated consistently when the `host` field contains a wildcard. (#93954, @Miciah) [SIG CLI, Cloud Provider, Cluster Lifecycle, Instrumentation, Network, Storage and Testing]

Other (Cleanup or Flake)

- --cache-dir sets cache directory for both http and discovery, defaults to $HOME/.kube/cache (#92910, @soltysh) [SIG API Machinery and CLI]
- Adds a bootstrapping ClusterRole, ClusterRoleBinding and group for /metrics, /livez/, /readyz/, & /healthz/- endpoints. (#93311, @logicalhan) [SIG API Machinery, Auth, Cloud Provider and Instrumentation]
- Base-images: Update to debian-iptables:buster-v1.3.0
  - Uses iptables 1.8.5
  - base-images: Update to debian-base:buster-v1.2.0
  - cluster/images/etcd: Build etcd:3.4.13-1 image
    - Uses debian-base:buster-v1.2.0 (#94733, @justaugustus) [SIG API Machinery, Release and Testing]
- Build: Update to debian-base@v2.1.2 and debian-iptables@v12.1.1 (#93667, @justaugustus) [SIG API Machinery, Release and Testing]
- Build: Update to debian-base@v2.1.3 and debian-iptables@v12.1.2 (#93916, @justaugustus) [SIG API Machinery, Release and Testing]
- Build: Update to go-runner:buster-v2.0.0 (#94167, @justaugustus) [SIG Release]
- Fix kubelet to properly log when a container is started. Before, sometimes the log said that a container is dead and was restarted when it was started for the first time. This only happened when using pods with initContainers and regular containers. (#91469, @rata) [SIG Node]
- Fix: license issue in blob disk feature (#92824, @andyzhangx) [SIG Cloud Provider]
- Fixes the flooding warning messages about setting volume ownership for configmap/secret volumes (#92878, @jvanz) [SIG Instrumentation, Node and Storage]
- Fixes the message about no auth for metrics in scheduler. (#94035, @zhouya0) [SIG Scheduling]

- Kube-up: defaults to limiting critical pods to the kube-system namespace to match behavior prior to 1.17 (#93121, @liggitt) [SIG Cloud Provider and Scheduling]
- Kubeadm: Separate argument key/value in log msg (#94016, @mrueg) [SIG Cluster Lifecycle]
- Kubeadm: remove support for the "ci/k8s-master" version label. This label has been removed in the Kubernetes CI release process and would no longer work in kubeadm. You can use the "ci/latest" version label instead. See kubernetes/test-infra#18517 (#93626, @vikkyomkar) [SIG Cluster Lifecycle]
- Kubeadm: remove the CoreDNS check for known image digests when applying the addon (#94506, @neolit123) [SIG Cluster Lifecycle]
- Kubernetes is now built with go1.15.0 (#93939, @justaugustus) [SIG Release and Testing]
- Kubernetes is now built with go1.15.0-rc.2 (#93827, @justaugustus) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation, Node, Release and Testing]
- Lock ExternalPolicyForExternalIP to default, this feature gate will be removed in 1.22. (#94581, @knabben) [SIG Network]
- Service.beta.kubernetes.io/azure-load-balancer-disable-tcp-reset is removed. All Standard load balancers will always enable tcp resets. (#94297, @MarcPow) [SIG Cloud Provider]
- Stop propagating SelfLink (deprecated in 1.16) in kube-apiserver (#94397, @wojtek-t) [SIG API Machinery and Testing]
- Strip unnecessary security contexts on Windows (#93475, @ravisantoshgudimetla) [SIG Node, Testing and Windows]
- To ensure the code be strong, add unit test for GetAddressAndDialer (#93180, @FreeZhang61) [SIG Node]
- Update CNI plugins to v0.8.7 (#94367, @justaugustus) [SIG Cloud Provider, Network, Node, Release and Testing]
- Update Golang to v1.14.5
  - Update repo-infra to 0.0.7 (to support go1.14.5 and go1.13.13)
    - Includes:
      - bazelbuild/bazel-toolchains@3.3.2
      - bazelbuild/rules_go@v0.22.7 (#93088, @justaugustus) [SIG Release and Testing]
- Update Golang to v1.14.6
  - Update repo-infra to 0.0.8 (to support go1.14.6 and go1.13.14)
    - Includes:
      - bazelbuild/bazel-toolchains@3.4.0
      - bazelbuild/rules_go@v0.22.8 (#93198, @justaugustus) [SIG Release and Testing]
- Update cri-tools to v1.19.0 (#94307, @xmudrii) [SIG Cloud Provider]
- Update default etcd server version to 3.4.9 (#92349, @jingyih) [SIG API Machinery, Cloud Provider, Cluster Lifecycle and Testing]
- Update etcd client side to v3.4.13 (#94259, @jingyih) [SIG API Machinery and Cloud Provider]
- `kubectl get ingress` now prefers the `networking.k8s.io/v1` over `extensions/v1beta1` (deprecated since v1.14). To explicitly request the deprecated version, use `kubectl get ingress.v1beta1.extensions`. (#94309, @liggitt) [SIG API Machinery and CLI]

Dependencies

Added

- github.com/Azure/go-autorest: v14.2.0+incompatible
- github.com/fvbommel/sortorder: v1.0.1
- github.com/yuin/goldmark: v1.1.27
- sigs.k8s.io/structured-merge-diff/v4: v4.0.1

Changed

- github.com/Azure/go-autorest/autorest/adal: v0.8.2 â† v0.9.0
- github.com/Azure/go-autorest/autorest/date: v0.2.0 â† v0.3.0
- github.com/Azure/go-autorest/autorest/mocks: v0.3.0 â† v0.4.0
- github.com/Azure/go-autorest/autorest: v0.9.6 â† v0.11.1
- github.com/Azure/go-autorest/logger: v0.1.0 â† v0.2.0
- github.com/Azure/go-autorest/tracing: v0.5.0 â† v0.6.0
- github.com/Microsoft/hcsshim: v0.8.9 â† 5eafd15
- github.com/cilium/ebpf: 9f1617e â† 1c8d4c9
- github.com/containerd/cgroups: bf292b2 â† 0dbf7f0
- github.com/coredns/corefile-migration: v1.0.8 â† v1.0.10
- github.com/evanphx/json-patch: e83c0a1 â† v4.9.0+incompatible
- github.com/google/cadvisor: 8450c56 â† v0.37.0
- github.com/json-iterator/go: v1.1.9 â† v1.1.10
- github.com/opencontainers/go-digest: v1.0.0-rc1 â† v1.0.0
- github.com/opencontainers/runc: 1b94395 â† 819fcc6
- github.com/prometheus/client_golang: v1.6.0 â† v1.7.1
- github.com/prometheus/common: v0.9.1 â† v0.10.0
- github.com/prometheus/procfs: v0.0.11 â† v0.1.3
- github.com/rubiojr/go-vhd: 0bfd3b3 â† 02e2102
- github.com/storageos/go-api: 343b3ef â† v2.2.0+incompatible
- github.com/urfave/cli: v1.22.1 â† v1.22.2
- go.etcd.io/etcd: 54ba958 â† dd1b699
- golang.org/x/crypto: bac4c82 â† 75b2880
- golang.org/x/mod: v0.1.0 â† v0.3.0
- golang.org/x/net: d3edc99 â† ab34263
- golang.org/x/tools: c00d67e â† c1934b7
- k8s.io/kube-openapi: 656914f â† 6aeccd4
- k8s.io/system-validators: v1.1.2 â† v1.2.0
- k8s.io/utils: 6e3d28b â† d5654de

Removed

- github.com/godbus/dbus: ade71ed
- github.com/xlab/handysort: fb3537e
- sigs.k8s.io/structured-merge-diff/v3: v3.0.0
- vbom.ml/util: db5cfe1

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified December 09, 2020 at 9:56 AM PST: Update release notes for 1.20 release (99ee1442c)
Edit this page *Create child page* *Create an issue*

Kubernetes version and version skew support policy

This document describes the maximum version skew supported between various Kubernetes components. Specific cluster deployment tools may place additional restrictions on version skew.

Supported versions

*Kubernetes versions are expressed as **x.y.z**, where **x** is the major version, **y** is the minor version, and **z** is the patch version, following Semantic Versioning terminology. For more information, see Kubernetes Release Versioning.*

The Kubernetes project maintains release branches for the most recent three minor releases (1.20, 1.19, 1.18). Kubernetes 1.19 and newer receive approximately 1 year of patch support. Kubernetes 1.18 and older received approximately 9 months of patch support.

Applicable fixes, including security fixes, may be backported to those three release branches, depending on severity and feasibility. Patch releases are cut from those branches at a regular cadence, plus additional urgent releases, when required.

The Release Managers group owns this decision.

For more information, see the Kubernetes patch releases page.

Supported version skew

kube-apiserver

In highly-available (HA) clusters, the newest and oldest `kube-apiserver` instances must be within one minor version.

Example:

- *newest `kube-apiserver` is at **1.20***
- *other `kube-apiserver` instances are supported at **1.20** and **1.19***

kubelet

`kubelet` must not be newer than `kube-apiserver`, and may be up to two minor versions older.

Example:

- *`kube-apiserver` is at **1.20***
- *`kubelet` is supported at **1.20**, **1.19**, and **1.18***

  ***Note:** If version skew exists between `kube-apiserver` instances in an HA cluster, this narrows the allowed `kubelet` versions.*

Example:

- *`kube-apiserver` instances are at **1.20** and **1.19***
- *`kubelet` is supported at **1.19**, and **1.18** (**1.20** is not supported because that would be newer than the `kube-apiserver` instance at version **1.19**)*

kube-controller-manager, kube-scheduler, and cloud-controller-manager

`kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` must not be newer than the `kube-apiserver` instances they

communicate with. They are expected to match the `kube-apiserver` minor version, but may be up to one minor version older (to allow live upgrades).

Example:

- *`kube-apiserver` is at **1.20***
- *`kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` are supported at **1.20** and **1.19***

  ***Note:** If version skew exists between `kube-apiserver` instances in an HA cluster, and these components can communicate with any `kube-apiserver` instance in the cluster (for example, via a load balancer), this narrows the allowed versions of these components.*

Example:

- *`kube-apiserver` instances are at **1.20** and **1.19***
- *`kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` communicate with a load balancer that can route to any `kube-apiserver` instance*
- *`kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` are supported at **1.19** (**1.20** is not supported because that would be newer than the `kube-apiserver` instance at version **1.19**)*

kubectl

`kubectl` is supported within one minor version (older or newer) of `kube-apiserver`.

Example:

- *`kube-apiserver` is at **1.20***
- *`kubectl` is supported at **1.21**, **1.20**, and **1.19***

  ***Note:** If version skew exists between `kube-apiserver` instances in an HA cluster, this narrows the supported `kubectl` versions.*

Example:

- *`kube-apiserver` instances are at **1.20** and **1.19***
- *`kubectl` is supported at **1.20** and **1.19** (other versions would be more than one minor version skewed from one of the `kube-apiserver` components)*

Supported component upgrade order

*The supported version skew between components has implications on the order in which components must be upgraded. This section describes the order in which components must be upgraded to transition an existing cluster from version **1.19** to version **1.20**.*

kube-apiserver

Pre-requisites:

- *In a single-instance cluster, the existing `kube-apiserver` instance is **1.19***
- *In an HA cluster, all `kube-apiserver` instances are at **1.19** or **1.20** (this ensures maximum skew of 1 minor version between the oldest and newest `kube-apiserver` instance)*
- *The `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` instances that communicate with this server are at version **1.19** (this ensures they are not newer than the existing API server version, and are within 1 minor version of the new API server version)*
- *`kubelet` instances on all nodes are at version **1.19** or **1.18** (this ensures they are not newer than the existing API server version, and are within 2 minor versions of the new API server version)*
- *Registered admission webhooks are able to handle the data the new `kube-apiserver` instance will send them:*
  - *`ValidatingWebhookConfiguration` and `MutatingWebhookConfiguration` objects are updated to include any new versions of REST resources added in **1.20** (or use the `matchPolicy: Equivalent` option available in v1.15+)*
  - *The webhooks are able to handle any new versions of REST resources that will be sent to them, and any new fields added to existing versions in **1.20***

*Upgrade `kube-apiserver` to **1.20***

> ***Note:** Project policies for API deprecation and API change guidelines require `kube-apiserver` to not skip minor versions when upgrading, even in single-instance clusters.*

kube-controller-manager, kube-scheduler, and cloud-controller-manager

Pre-requisites:

- *The `kube-apiserver` instances these components communicate with are at **1.20** (in HA clusters in which these control plane components can communicate with any `kube-apiserver` instance in the cluster, all `kube-apiserver` instances must be upgraded before upgrading these components)*

*Upgrade `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` to **1.20***

kubelet

Pre-requisites:

- *The `kube-apiserver` instances the `kubelet` communicates with are at **1.20***

*Optionally upgrade `kubelet` instances to **1.20** (or they can be left at **1.19** or **1.18**)*

> ***Warning:***
>
> *Running a cluster with `kubelet` instances that are persistently two minor versions behind `kube-apiserver` is not recommended:*
>
> - *they must be upgraded within one minor version of `kube-apiserver` before the control plane can be upgraded*
> - *it increases the likelihood of running `kubelet` versions older than the three maintained minor releases*

kube-proxy

- *`kube-proxy` must be the same minor version as `kubelet` on the node.*
- *`kube-proxy` must not be newer than `kube-apiserver`.*
- *`kube-proxy` must be at most two minor versions older than `kube-apiserver`.*

Example:

*If `kube-proxy` version is **1.18**:*

- *`kubelet` version must be at the same minor version as **1.18**.*
- *`kube-apiserver` version must be between **1.18** and **1.20**, inclusive.*

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified August 15, 2020 at 9:55 PM PST: fix example in k8s.io/docs/setup/release/version-skew-policy (e0eedf9bd)
Edit this page Create child page Create an issue

Install Tools

Set up Kubernetes tools on your computer.

kubectl

The Kubernetes command-line tool, `kubectl`, allows you to run commands against Kubernetes clusters. You can use `kubectl` to deploy applications, inspect and manage cluster resources, and view logs.

See Install and Set Up `kubectl` for information about how to download and install `kubectl` and set it up for accessing your cluster.

View kubectl Install and Set Up Guide

You can also read the `kubectl` reference documentation.

kind

`kind` lets you run Kubernetes on your local computer. This tool requires that you have Docker installed and configured.

The kind Quick Start page shows you what you need to do to get up and running with kind.

View kind Quick Start Guide

minikube

Like `kind`, `minikube` is a tool that lets you run Kubernetes locally. `minikube` runs a single-node Kubernetes cluster on your personal computer (including Windows, macOS and Linux PCs) so that you can try out Kubernetes, or for daily development work.

You can follow the official Get Started! guide if your focus is on getting the tool installed.

View minikube Get Started! Guide

Once you have `minikube` working, you can use it to run a sample application.

kubeadm

You can use the kubeadm tool to create and manage Kubernetes clusters. It performs the actions necessary to get a minimum viable, secure cluster up and running in a user friendly way.

Installing kubeadm shows you how to install kubeadm. Once installed, you can use it to create a cluster.

View kubeadm Install Guide

Production environment

Container runtimes

Installing Kubernetes with deployment tools

Turnkey Cloud Solutions

Windows in Kubernetes

Container runtimes

You need to install a container runtime into each node in the cluster so that Pods can run there. This page outlines what is involved and describes related tasks for setting up nodes.

This page lists details for using several common container runtimes with Kubernetes, on Linux:

- *containerd*
- *CRI-O*
- *Docker*

> ***Note:** For other operating systems, look for documentation specific to your platform.*

Cgroup drivers

Control groups are used to constrain resources that are allocated to processes.

When systemd is chosen as the init system for a Linux distribution, the init process generates and consumes a root control group (`cgroup`) and acts as a

cgroup manager. Systemd has a tight integration with cgroups and allocates a cgroup per systemd unit. It's possible to configure your container runtime and the kubelet to use `cgroupfs`. Using `cgroupfs` alongside systemd means that there will be two different cgroup managers.

A single cgroup manager simplifies the view of what resources are being allocated and will by default have a more consistent view of the available and in-use resources. When there are two cgroup managers on a system, you end up with two views of those resources. In the field, people have reported cases where nodes that are configured to use `cgroupfs` for the kubelet and Docker, but `systemd` for the rest of the processes, become unstable under resource pressure.

Changing the settings such that your container runtime and kubelet use `systemd` as the cgroup driver stabilized the system. To configure this for Docker, set `native.cgroupdriver=systemd`.

> ***Caution:***
>
> *Changing the cgroup driver of a Node that has joined a cluster is strongly not recommended.*
> *If the kubelet has created Pods using the semantics of one cgroup driver, changing the container runtime to another cgroup driver can cause errors when trying to re-create the Pod sandbox for such existing Pods. Restarting the kubelet may not solve such errors.*
>
> *If you have automation that makes it feasible, replace the node with another using the updated configuration, or reinstall it using automation.*

Container runtimes

> ***Caution:*** *This section links to third party projects that provide functionality required by Kubernetes. The Kubernetes project authors aren't responsible for these projects. This page follows CNCF website guidelines by listing projects alphabetically. To add a project to this list, read the content guide before submitting a change.*

containerd

This section contains the necessary steps to use `containerd` as CRI runtime.

Use the following commands to install Containerd on your system:

Install and configure prerequisites:

```
cat <<EOF | sudo tee /etc/modules-load.d/containerd.conf
overlay
```

```
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# Setup required sysctl params, these persist across reboots.
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

# Apply sysctl params without reboot
sudo sysctl --system
```

Install containerd:

- *Ubuntu 16.04*
- *CentOS/RHEL 7.4+*
- *Windows (PowerShell)*

```
# (Install containerd)
## Set up the repository
### Install packages to allow apt to use a repository over HTTPS
sudo apt-get update && sudo apt-get install -y apt-transport-
https ca-certificates curl software-properties-common
```

```
## Add Docker's official GPG key
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key --keyring /etc/apt/trusted.gpg.d/docker.gpg add -
```

```
## Add Docker apt repository.
sudo add-apt-repository \
    "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
    $(lsb_release -cs) \
    stable"
```

```
## Install containerd
sudo apt-get update && sudo apt-get install -y containerd.io
```

```
# Configure containerd
sudo mkdir -p /etc/containerd
sudo containerd config default | sudo tee /etc/containerd/
config.toml
```

```
# Restart containerd
sudo systemctl restart containerd
```

```
# (Install containerd)
## Set up the repository
### Install required packages
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
## Add docker repository
sudo yum-config-manager \
    --add-repo \
    https://download.docker.com/linux/centos/docker-ce.repo

## Install containerd
sudo yum update -y && sudo yum install -y containerd.io

## Configure containerd
sudo mkdir -p /etc/containerd
sudo containerd config default > /etc/containerd/config.toml

# Restart containerd
sudo systemctl restart containerd

# (Install containerd)
# download containerd
cmd /c curl -OL https://github.com/containerd/containerd/
releases/download/v1.4.1/containerd-1.4.1-windows-amd64.tar.gz
cmd /c tar xvf .\containerd-1.4.1-windows-amd64.tar.gz

# extract and configure
Copy-Item -Path ".\bin\" -Destination "$Env:ProgramFiles\containe
rd" -Recurse -Force
cd $Env:ProgramFiles\containerd\
.\containerd.exe config default | Out-File config.toml -Encoding
ascii

# review the configuration. depending on setup you may want to
adjust:
# - the sandbox_image (kubernetes pause image)
# - cni bin_dir and conf_dir locations
Get-Content config.toml

# start containerd
.\containerd.exe --register-service
Start-Service containerd
```

systemd

To use the `systemd` cgroup driver in `/etc/containerd/config.toml` with `runc`, set

```
[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc]
  ...

[plugins."io.containerd.grpc.v1.cri".containerd.runtimes.runc.opt
ions]
    SystemdCgroup = true
```

When using kubeadm, manually configure the [cgroup driver for kubelet](#).

CRI-O

This section contains the necessary steps to install CRI-O as a container runtime.

Use the following commands to install CRI-O on your system:

> ***Note:*** *The CRI-O major and minor versions must match the Kubernetes major and minor versions. For more information, see the CRI-O compatibility matrix.*

Install and configure prerequisites:

```
sudo modprobe overlay
sudo modprobe br_netfilter

# Set up required sysctl params, these persist across reboots.
cat <<EOF | sudo tee /etc/sysctl.d/99-kubernetes-cri.conf
net.bridge.bridge-nf-call-iptables  = 1
net.ipv4.ip_forward                 = 1
net.bridge.bridge-nf-call-ip6tables = 1
EOF

sudo sysctl --system
```

- *Debian*
- *Ubuntu*
- *CentOS*
- *openSUSE Tumbleweed*
- *Fedora*

To install CRI-O on the following operating systems, set the environment variable `OS` to the appropriate value from the following table:

Operating system	**`$OS`**
Debian Unstable	`Debian_Unstable`
Debian Testing	`Debian_Testing`

Then, set `$VERSION` to the CRI-O version that matches your Kubernetes version. For instance, if you want to install CRI-O 1.18, set `VERSION=1.18`. You can pin your installation to a specific release. To install version 1.18.3, set `VERSION=1.18:1.18.3`.

Then run

```
cat <<EOF | sudo tee /etc/apt/sources.list.d/
devel:kubic:libcontainers:stable.list
deb https://download.opensuse.org/repositories/devel:/kubic:/
libcontainers:/stable/$OS/ /
EOF
cat <<EOF | sudo tee /etc/apt/sources.list.d/
```

```
devel:kubic:libcontainers:stable:cri-o:$VERSION.list
deb http://download.opensuse.org/repositories/devel:/kubic:/
libcontainers:/stable:/cri-o:/$VERSION/$OS/ /
EOF

curl -L https://download.opensuse.org/repositories/
devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/Release.key
| sudo apt-key --keyring /etc/apt/trusted.gpg.d/
libcontainers.gpg add -
curl -L https://download.opensuse.org/repositories/devel:/kubic:/
libcontainers:/stable/$OS/Release.key | sudo apt-key --keyring /
etc/apt/trusted.gpg.d/libcontainers.gpg add -

sudo apt-get update
sudo apt-get install cri-o cri-o-runc
```

To install on the following operating systems, set the environment variable `OS` to the appropriate field in the following table:

Operating system	$OS
Ubuntu 20.04	xUbuntu_20.04
Ubuntu 19.10	xUbuntu_19.10
Ubuntu 19.04	xUbuntu_19.04
Ubuntu 18.04	xUbuntu_18.04

Then, set `$VERSION` to the CRI-O version that matches your Kubernetes version. For instance, if you want to install CRI-O 1.18, set `VERSION=1.18`. You can pin your installation to a specific release. To install version 1.18.3, set `VERSION=1.18:1.18.3`.

Then run

```
cat <<EOF | sudo tee /etc/apt/sources.list.d/
devel:kubic:libcontainers:stable.list
deb https://download.opensuse.org/repositories/devel:/kubic:/
libcontainers:/stable/$OS/ /
EOF
cat <<EOF | sudo tee /etc/apt/sources.list.d/
devel:kubic:libcontainers:stable:cri-o:$VERSION.list
deb http://download.opensuse.org/repositories/devel:/kubic:/
libcontainers:/stable:/cri-o:/$VERSION/$OS/ /
EOF

curl -L https://download.opensuse.org/repositories/devel:/kubic:/
libcontainers:/stable/$OS/Release.key | sudo apt-key --keyring /
etc/apt/trusted.gpg.d/libcontainers.gpg add -
curl -L https://download.opensuse.org/repositories/
devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/Release.key
| sudo apt-key --keyring /etc/apt/trusted.gpg.d/libcontainers-
cri-o.gpg add -
```

```
sudo apt-get update
sudo apt-get install cri-o cri-o-runc
```

To install on the following operating systems, set the environment variable `OS` to the appropriate field in the following table:

Operating system	**`$OS`**
Centos 8	`CentOS_8`
Centos 8 Stream	`CentOS_8_Stream`
Centos 7	`CentOS_7`

Then, set `$VERSION` to the CRI-O version that matches your Kubernetes version. For instance, if you want to install CRI-O 1.18, set `VERSION=1.18`. You can pin your installation to a specific release. To install version 1.18.3, set `VERSION=1.18:1.18.3`.

Then run

```
sudo curl -L -o /etc/yum.repos.d/
devel:kubic:libcontainers:stable.repo https://
download.opensuse.org/repositories/devel:/kubic:/libcontainers:/
stable/$OS/devel:kubic:libcontainers:stable.repo
sudo curl -L -o /etc/yum.repos.d/
devel:kubic:libcontainers:stable:cri-o:$VERSION.repo https://
download.opensuse.org/repositories/
devel:kubic:libcontainers:stable:cri-o:$VERSION/$OS/
devel:kubic:libcontainers:stable:cri-o:$VERSION.repo
sudo yum install cri-o
```

```
sudo zypper install cri-o
```

Set `$VERSION` to the CRI-O version that matches your Kubernetes version. For instance, if you want to install CRI-O 1.18, `VERSION=1.18`.

You can find available versions with:

```
sudo dnf module list cri-o
```

CRI-O does not support pinning to specific releases on Fedora.

Then run

```
sudo dnf module enable cri-o:$VERSION
sudo dnf install cri-o
```

Start CRI-O:

```
sudo systemctl daemon-reload
sudo systemctl start crio
```

Refer to the [CRI-O installation guide] for more information.

Docker

On each of your nodes, install Docker CE.

The Kubernetes release notes list which versions of Docker are compatible with that version of Kubernetes.

Use the following commands to install Docker on your system:

- *Ubuntu 16.04+*
- *CentOS/RHEL 7.4+*

```
# (Install Docker CE)
## Set up the repository:
### Install packages to allow apt to use a repository over HTTPS
sudo apt-get update && sudo apt-get install -y \
  apt-transport-https ca-certificates curl software-properties-
common gnupg2
```

```
# Add Docker's official GPG key:
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo
apt-key --keyring /etc/apt/trusted.gpg.d/docker.gpg add -
```

```
# Add the Docker apt repository:
sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"
```

```
# Install Docker CE
sudo apt-get update && sudo apt-get install -y \
  containerd.io=1.2.13-2 \
  docker-ce=5:19.03.11~3-0~ubuntu-$(lsb_release -cs) \
  docker-ce-cli=5:19.03.11~3-0~ubuntu-$(lsb_release -cs)
```

```
# Set up the Docker daemon
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF
```

```
# Create /etc/systemd/system/docker.service.d
sudo mkdir -p /etc/systemd/system/docker.service.d
```

```
# Restart Docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

```
# (Install Docker CE)
## Set up the repository
### Install required packages
sudo yum install -y yum-utils device-mapper-persistent-data lvm2

## Add the Docker repository
sudo yum-config-manager --add-repo \
  https://download.docker.com/linux/centos/docker-ce.repo

# Install Docker CE
sudo yum update -y && sudo yum install -y \
  containerd.io-1.2.13 \
  docker-ce-19.03.11 \
  docker-ce-cli-19.03.11

## Create /etc/docker
sudo mkdir /etc/docker

# Set up the Docker daemon
cat <<EOF | sudo tee /etc/docker/daemon.json
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
    "max-size": "100m"
  },
  "storage-driver": "overlay2",
  "storage-opts": [
    "overlay2.override_kernel_check=true"
  ]
}
EOF

# Create /etc/systemd/system/docker.service.d
sudo mkdir -p /etc/systemd/system/docker.service.d

# Restart Docker
sudo systemctl daemon-reload
sudo systemctl restart docker
```

If you want the `docker` service to start on boot, run the following command:

```
sudo systemctl enable docker
```

Refer to the [official Docker installation guides] for more information.

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified November 20, 2020 at 9:16 AM PST: Update container-runtimes.md (07a8b3315)
Edit this page Create child page Create an issue

Installing Kubernetes with deployment tools

Bootstrapping clusters with kubeadm

Installing kubeadm

This page shows how to install the `kubeadm` toolbox. For information how to create a cluster with kubeadm once you have performed this installation process, see the Using kubeadm to Create a Cluster page.

Before you begin

- *One or more machines running one of:*
  - *Ubuntu 16.04+*
  - *Debian 9+*

    - *CentOS 7*
    - *Red Hat Enterprise Linux (RHEL) 7*
    - *Fedora 25+*
    - *HypriotOS v1.0.1+*
    - *Flatcar Container Linux (tested with 2512.3.0)*
- *2 GB or more of RAM per machine (any less will leave little room for your apps)*
- *2 CPUs or more*
- *Full network connectivity between all machines in the cluster (public or private network is fine)*
- *Unique hostname, MAC address, and product_uuid for every node. See here for more details.*
- *Certain ports are open on your machines. See here for more details.*
- *Swap disabled. You **MUST** disable swap in order for the kubelet to work properly.*

Verify the MAC address and product_uuid are unique for every node

- *You can get the MAC address of the network interfaces using the command `ip link` or `ifconfig -a`*
- *The product_uuid can be checked by using the command `sudo cat /sys/class/dmi/id/product_uuid`*

It is very likely that hardware devices will have unique addresses, although some virtual machines may have identical values. Kubernetes uses these values to uniquely identify the nodes in the cluster. If these values are not unique to each node, the installation process may fail.

Check network adapters

If you have more than one network adapter, and your Kubernetes components are not reachable on the default route, we recommend you add IP route(s) so Kubernetes cluster addresses go via the appropriate adapter.

Letting iptables see bridged traffic

Make sure that the `br_netfilter` module is loaded. This can be done by running `lsmod | grep br_netfilter`. To load it explicitly call `sudo modprobe br_netfilter`.

As a requirement for your Linux Node's iptables to correctly see bridged traffic, you should ensure `net.bridge.bridge-nf-call-iptables` is set to 1 in your `sysctl` config, e.g.

```
cat <<EOF | sudo tee /etc/sysctl.d/k8s.conf
net.bridge.bridge-nf-call-ip6tables = 1
net.bridge.bridge-nf-call-iptables = 1
EOF
sudo sysctl --system
```

For more details please see the Network Plugin Requirements page.

Check required ports

Control-plane node(s)

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	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	30000-32767	NodePort Servicesâ€	All

â€ Default port range for NodePort Services.

*Any port numbers marked with * are overridable, so you will need to ensure any custom ports you provide are also open.*

Although etcd ports are included in control-plane nodes, you can also host your own etcd cluster externally or on custom ports.

The pod network plugin you use (see below) may also require certain ports to be open. Since this differs with each pod network plugin, please see the documentation for the plugins about what port(s) those need.

Installing runtime

To run containers in Pods, Kubernetes uses a container runtime.

- *Linux nodes*
- *other operating systems*

By default, Kubernetes uses the Container Runtime Interface (CRI) to interface with your chosen container runtime.

If you don't specify a runtime, kubeadm automatically tries to detect an installed container runtime by scanning through a list of well known Unix domain sockets. The following table lists container runtimes and their associated socket paths:

Runtime	Path to Unix domain socket
Docker	`/var/run/docker.sock`
containerd	`/run/containerd/containerd.sock`
CRI-O	`/var/run/crio/crio.sock`

If both Docker and containerd are detected, Docker takes precedence. This is needed because Docker 18.09 ships with containerd and both are detectable even if you only installed Docker. If any other two or more runtimes are detected, kubeadm exits with an error.

The kubelet integrates with Docker through the built-in `dockershim` CRI implementation.

See container runtimes for more information.

By default, kubeadm uses Docker as the container runtime. The kubelet integrates with Docker through the built-in `dockershim` CRI implementation.

See container runtimes for more information.

Installing kubeadm, kubelet and kubectl

You will install these packages on all of your machines:

- *`kubeadm`: the command to bootstrap the cluster.*
- *`kubelet`: the component that runs on all of the machines in your cluster and does things like starting pods and containers.*
- *`kubectl`: the command line util to talk to your cluster.*

*kubeadm **will not** install or manage `kubelet` or `kubectl` for you, so you will need to ensure they match the version of the Kubernetes control plane you want kubeadm to install for you. If you do not, there is a risk of a version skew occurring that can lead to unexpected, buggy behaviour. However, one minor version skew between the kubelet and the control plane is supported, but the kubelet version may never exceed the API server version. For example, kubelets running 1.7.0 should be fully compatible with a 1.8.0 API server, but not vice versa.*

For information about installing `kubectl`, see Install and set up kubectl.

> ***Warning:** These instructions exclude all Kubernetes packages from any system upgrades. This is because kubeadm and Kubernetes require special attention to upgrade.*

For more information on version skews, see:

- *Kubernetes version and version-skew policy*

- *Kubeadm-specific [version skew policy](#)*

- *[Ubuntu, Debian or HypriotOS](#)*
- *[CentOS, RHEL or Fedora](#)*
- *[Fedora CoreOS or Flatcar Container Linux](#)*

```
sudo apt-get update && sudo apt-get install -y apt-transport-
https curl
curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg |
sudo apt-key add -
cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF
sudo apt-get update
sudo apt-get install -y kubelet kubeadm kubectl
sudo apt-mark hold kubelet kubeadm kubectl
```

```
cat <<EOF | sudo tee /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-
el7-\$basearch
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg
https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg
exclude=kubelet kubeadm kubectl
EOF

# Set SELinux in permissive mode (effectively disabling it)
sudo setenforce 0
sudo sed -i 's/^SELINUX=enforcing$/SELINUX=permissive/' /etc/
selinux/config

sudo yum install -y kubelet kubeadm kubectl --disableexcludes=kub
ernetes

sudo systemctl enable --now kubelet
```

Notes:

- *Setting SELinux in permissive mode by running `setenforce 0` and `sed ...` effectively disables it. This is required to allow containers to access the host filesystem, which is needed by pod networks for example. You have to do this until SELinux support is improved in the kubelet.*

- *You can leave SELinux enabled if you know how to configure it but it may require settings that are not supported by kubeadm.*

Install CNI plugins (required for most pod network):

```
CNI_VERSION="v0.8.2"
sudo mkdir -p /opt/cni/bin
curl -L "https://github.com/containernetworking/plugins/releases/
download/${CNI_VERSION}/cni-plugins-linux-amd64-${CNI_VERSION}.tg
z" | sudo tar -C /opt/cni/bin -xz
```

Define the directory to download command files

> ***Note:*** *The DOWNLOAD_DIR variable must be set to a writable directory. If you are running Flatcar Container Linux, set DOWNLOAD_DIR=/opt/bin.*

```
DOWNLOAD_DIR=/usr/local/bin
sudo mkdir -p $DOWNLOAD_DIR
```

Install crictl (required for kubeadm / Kubelet Container Runtime Interface (CRI))

```
CRICTL_VERSION="v1.17.0"
curl -L "https://github.com/kubernetes-sigs/cri-tools/releases/
download/${CRICTL_VERSION}/crictl-${CRICTL_VERSION}-linux-
amd64.tar.gz" | sudo tar -C $DOWNLOAD_DIR -xz
```

Install `kubeadm`, `kubelet`, `kubectl` and add a `kubelet` systemd service:

```
RELEASE="$(curl -sSL https://dl.k8s.io/release/stable.txt)"
cd $DOWNLOAD_DIR
sudo curl -L --remote-name-all https://storage.googleapis.com/
kubernetes-release/release/${RELEASE}/bin/linux/amd64/{kubeadm,ku
belet,kubectl}
sudo chmod +x {kubeadm,kubelet,kubectl}

RELEASE_VERSION="v0.4.0"
curl -sSL "https://raw.githubusercontent.com/kubernetes/release/$
{RELEASE_VERSION}/cmd/kubepkg/templates/latest/deb/kubelet/lib/
systemd/system/kubelet.service" | sed "s:/usr/bin:$
{DOWNLOAD_DIR}:g" | sudo tee /etc/systemd/system/kubelet.service
sudo mkdir -p /etc/systemd/system/kubelet.service.d
curl -sSL "https://raw.githubusercontent.com/kubernetes/release/$
{RELEASE_VERSION}/cmd/kubepkg/templates/latest/deb/kubeadm/10-
kubeadm.conf" | sed "s:/usr/bin:${DOWNLOAD_DIR}:g" | sudo tee /
etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

Enable and start `kubelet`:

```
systemctl enable --now kubelet
```

> ***Note:*** *The Flatcar Container Linux distribution mounts the `/usr` directory as a read-only filesystem. Before bootstrapping your cluster, you need to take additional steps to configure a writable directory. See the Kubeadm Troubleshooting guide to learn how to set up a writable directory.*

The kubelet is now restarting every few seconds, as it waits in a crashloop for kubeadm to tell it what to do.

Configure cgroup driver used by kubelet on control-plane node

When using Docker, kubeadm will automatically detect the cgroup driver for the kubelet and set it in the `/var/lib/kubelet/config.yaml` file during runtime.

If you are using a different CRI, you must pass your `cgroupDriver` value to `kubeadm init`, like so:

```
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
cgroupDriver: <value>
```

For further details, please read Using kubeadm init with a configuration file.

Please mind, that you **only** have to do that if the cgroup driver of your CRI is not `cgroupfs`, because that is the default value in the kubelet already.

> **Note:** Since `--cgroup-driver` flag has been deprecated by kubelet, if you have that in `/var/lib/kubelet/kubeadm-flags.env` or `/etc/default/kubelet`(`/etc/sysconfig/kubelet` for RPMs), please remove it and use the KubeletConfiguration instead (stored in `/var/lib/kubelet/config.yaml` by default).

Restarting the kubelet is required:

```
sudo systemctl daemon-reload
sudo systemctl restart kubelet
```

The automatic detection of cgroup driver for other container runtimes like CRI-O and containerd is work in progress.

Troubleshooting

If you are running into difficulties with kubeadm, please consult our troubleshooting docs.

What's next

- Using kubeadm to Create a Cluster

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified December 08, 2020 at 5:39 PM PST: Add sudo when running systemctl (5f6b3af86)
Edit this page Create child page Create an issue

Troubleshooting kubeadm

As with any program, you might run into an error installing or running kubeadm. This page lists some common failure scenarios and have provided steps that can help you understand and fix the problem.

If your problem is not listed below, please follow the following steps:

- *If you think your problem is a bug with kubeadm:*
  - *Go to github.com/kubernetes/kubeadm and search for existing issues.*
  - *If no issue exists, please open one and follow the issue template.*
- *If you are unsure about how kubeadm works, you can ask on Slack in `#kubeadm`, or open a question on StackOverflow. Please include relevant tags like `#kubernetes` and `#kubeadm` so folks can help you.*

Not possible to join a v1.18 Node to a v1.17 cluster due to missing RBAC

In v1.18 kubeadm added prevention for joining a Node in the cluster if a Node with the same name already exists. This required adding RBAC for the bootstrap-token user to be able to GET a Node object.

However this causes an issue where `kubeadm join` from v1.18 cannot join a cluster created by kubeadm v1.17.

To workaround the issue you have two options:

Execute `kubeadm init phase bootstrap-token` on a control-plane node using kubeadm v1.18. Note that this enables the rest of the bootstrap-token permissions as well.

or

Apply the following RBAC manually using `kubectl apply -f ...`:

```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  name: kubeadm:get-nodes
rules:
- apiGroups:
  - ""
  resources:
  - nodes
  verbs:
  - get
---
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  name: kubeadm:get-nodes
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: kubeadm:get-nodes
subjects:
- apiGroup: rbac.authorization.k8s.io
  kind: Group
  name: system:bootstrappers:kubeadm:default-node-token
```

`ebtables` or some similar executable not found during installation

If you see the following warnings while running `kubeadm init`

```
[preflight] WARNING: ebtables not found in system path
[preflight] WARNING: ethtool not found in system path
```

Then you may be missing `ebtables`, `ethtool` or a similar executable on your node. You can install them with the following commands:

- *For Ubuntu/Debian users, run `apt install ebtables ethtool`.*
- *For CentOS/Fedora users, run `yum install ebtables ethtool`.*

kubeadm blocks waiting for control plane during installation

If you notice that `kubeadm init` hangs after printing out the following line:

```
[apiclient] Created API client, waiting for the control plane to become ready
```

This may be caused by a number of problems. The most common are:

- *network connection problems. Check that your machine has full network connectivity before continuing.*
- *the default cgroup driver configuration for the kubelet differs from that used by Docker. Check the system log file (e.g. `/var/log/message`) or examine the output from `journalctl -u kubelet`. If you see something like the following:*

  ```
  error: failed to run Kubelet: failed to create kubelet: misconfiguration: kubelet cgroup driver: "systemd" is different from docker cgroup driver: "cgroupfs"
  ```

  There are two common ways to fix the cgroup driver problem:

1. *Install Docker again following instructions here.*
2. *Change the kubelet config to match the Docker cgroup driver manually, you can refer to Configure cgroup driver used by kubelet on Master Node*

- *control plane Docker containers are crashlooping or hanging. You can check this by running `docker ps` and investigating each container by running `docker logs`.*

kubeadm blocks when removing managed containers

The following could happen if Docker halts and does not remove any Kubernetes-managed containers:

```
sudo kubeadm reset
[preflight] Running pre-flight checks
[reset] Stopping the kubelet service
[reset] Unmounting mounted directories in "/var/lib/kubelet"
[reset] Removing kubernetes-managed containers
(block)
```

A possible solution is to restart the Docker service and then re-run `kubeadm reset`:

```
sudo systemctl restart docker.service
sudo kubeadm reset
```

Inspecting the logs for docker may also be useful:

```
journalctl -u docker
```

Pods in `RunContainerError`, `CrashLoopBackOff` or `Error` state

Right after `kubeadm init` there should not be any pods in these states.

- *If there are pods in one of these states right after `kubeadm init`, please open an issue in the kubeadm repo. `coredns` (or `kube-dns`) should be in the `Pending` state until you have deployed the network solution.*
- *If you see Pods in the `RunContainerError`, `CrashLoopBackOff` or `Error` state after deploying the network solution and nothing happens to `coredns` (or `kube-dns`), it's very likely that the Pod Network solution that you installed is somehow broken. You might have to grant it more RBAC privileges or use a newer version. Please file an issue in the Pod Network providers' issue tracker and get the issue triaged there.*
- *If you install a version of Docker older than 1.12.1, remove the `MountFlags=slave` option when booting `dockerd` with `systemd` and restart `docker`. You can see the MountFlags in `/usr/lib/systemd/system/docker.service`. MountFlags can interfere with volumes mounted by Kubernetes, and put the Pods in `CrashLoopBackOff` state. The error happens when Kubernetes does not find `var/run/secrets/kubernetes.io/serviceaccount` files.*

`coredns` (or `kube-dns`) is stuck in the `Pending` state

*This is **expected** and part of the design. kubeadm is network provider-agnostic, so the admin should [install the pod network solution] of choice. You have to install a Pod Network before CoreDNS may be deployed fully. Hence the `Pending` state before the network is set up.*

`HostPort` services do not work

The `HostPort` and `HostIP` functionality is available depending on your Pod Network provider. Please contact the author of the Pod Network solution to find out whether `HostPort` and `HostIP` functionality are available.

Calico, Canal, and Flannel CNI providers are verified to support HostPort.

For more information, see the [CNI portmap documentation].

If your network provider does not support the portmap CNI plugin, you may need to use the [NodePort feature of services] or use `HostNetwork=true`.

Pods are not accessible via their Service IP

- Many network add-ons do not yet enable hairpin mode which allows pods to access themselves via their Service IP. This is an issue related to CNI. Please contact the network add-on provider to get the latest status of their support for hairpin mode.
- If you are using VirtualBox (directly or via Vagrant), you will need to ensure that `hostname -i` returns a routable IP address. By default the first interface is connected to a non-routable host-only network. A work around is to modify `/etc/hosts`, see this Vagrantfile for an example.

TLS certificate errors

The following error indicates a possible certificate mismatch.

```
# kubectl get pods
Unable to connect to the server: x509: certificate signed by
unknown authority (possibly because of "crypto/rsa: verification
error" while trying to verify candidate authority certificate
"kubernetes")
```

- Verify that the `$HOME/.kube/config` file contains a valid certificate, and regenerate a certificate if necessary. The certificates in a kubeconfig file are base64 encoded. The `base64 --decode` command can be used to decode the certificate and `openssl x509 -text -noout` can be used for viewing the certificate information.
- Unset the `KUBECONFIG` environment variable using:

  ```
  unset KUBECONFIG
  ```

  Or set it to the default `KUBECONFIG` location:

  ```
  export KUBECONFIG=/etc/kubernetes/admin.conf
  ```

- Another workaround is to overwrite the existing `kubeconfig` for the "admin" user:

  ```
  mv  $HOME/.kube $HOME/.kube.bak
  mkdir $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config
  ```

Default NIC When using flannel as the pod network in Vagrant

The following error might indicate that something was wrong in the pod network:

```
Error from server (NotFound): the server could not find the
requested resource
```

- *If you're using flannel as the pod network inside Vagrant, then you will have to specify the default interface name for flannel.*

  Vagrant typically assigns two interfaces to all VMs. The first, for which all hosts are assigned the IP address `10.0.2.15`, is for external traffic that gets NATed.

  This may lead to problems with flannel, which defaults to the first interface on a host. This leads to all hosts thinking they have the same public IP address. To prevent this, pass the `--iface eth1` flag to flannel so that the second interface is chosen.

Non-public IP used for containers

In some situations `kubectl logs` and `kubectl run` commands may return with the following errors in an otherwise functional cluster:

```
Error from server: Get https://10.19.0.41:10250/containerLogs/
default/mysql-ddc65b868-glc5m/mysql: dial tcp 10.19.0.41:10250:
getsockopt: no route to host
```

- *This may be due to Kubernetes using an IP that can not communicate with other IPs on the seemingly same subnet, possibly by policy of the machine provider.*
- *DigitalOcean assigns a public IP to `eth0` as well as a private one to be used internally as anchor for their floating IP feature, yet `kubelet` will pick the latter as the node's `InternalIP` instead of the public one.*

  Use `ip addr show` to check for this scenario instead of `ifconfig` because `ifconfig` will not display the offending alias IP address. Alternatively an API endpoint specific to DigitalOcean allows to query for the anchor IP from the droplet:

  ```
  curl http://169.254.169.254/metadata/v1/interfaces/public/0/
  anchor_ipv4/address
  ```

  The workaround is to tell `kubelet` which IP to use using `--node-ip`. When using DigitalOcean, it can be the public one (assigned to `eth0`) or the private one (assigned to `eth1`) should you want to use the optional private network. The [`KubeletExtraArgs` section of the kubeadm `NodeRegistrationOptions` structure]() can be used for this.

  Then restart `kubelet`:

  ```
  systemctl daemon-reload
  systemctl restart kubelet
  ```

`coredns` pods have `CrashLoopBackOff` or `Error` state

If you have nodes that are running SELinux with an older version of Docker you might experience a scenario where the `coredns` pods are not starting. To solve that you can try one of the following options:

- Upgrade to a newer version of Docker.
- Disable SELinux.
- Modify the `coredns` deployment to set `allowPrivilegeEscalation` to `true`:

```
kubectl -n kube-system get deployment coredns -o yaml | \
  sed 's/allowPrivilegeEscalation: false/
allowPrivilegeEscalation: true/g' | \
  kubectl apply -f -
```

Another cause for CoreDNS to have `CrashLoopBackOff` is when a CoreDNS Pod deployed in Kubernetes detects a loop. A number of workarounds are available to avoid Kubernetes trying to restart the CoreDNS Pod every time CoreDNS detects the loop and exits.

> **Warning:** Disabling SELinux or setting `allowPrivilegeEscalation` to `true` can compromise the security of your cluster.

etcd pods restart continually

If you encounter the following error:

```
rpc error: code = 2 desc = oci runtime error: exec failed:
container_linux.go:247: starting container process caused
"process_linux.go:110: decoding init error from pipe caused
\"read parent: connection reset by peer\""
```

this issue appears if you run CentOS 7 with Docker 1.13.1.84. This version of Docker can prevent the kubelet from executing into the etcd container.

To work around the issue, choose one of these options:

- Roll back to an earlier version of Docker, such as 1.13.1-75

```
yum downgrade docker-1.13.1-75.git8633870.el7.centos.x86_64
docker-client-1.13.1-75.git8633870.el7.centos.x86_64 docker-
common-1.13.1-75.git8633870.el7.centos.x86_64
```

- Install one of the more recent recommended versions, such as 18.06:

```
sudo yum-config-manager --add-repo https://download.docker.com/
linux/centos/docker-ce.repo
yum install docker-ce-18.06.1.ce-3.el7.x86_64
```

Not possible to pass a comma separated list of values to arguments inside a `--component-extra-args` flag

`kubeadm init` flags such as `--component-extra-args` allow you to pass custom arguments to a control-plane component like the kube-apiserver. However, this mechanism is limited due to the underlying type used for parsing the values (`mapStringString`).

If you decide to pass an argument that supports multiple, comma-separated values such as `--apiserver-extra-args "enable-admission-plugins=LimitRanger,NamespaceExists"` this flag will fail with `flag: malformed pair, expect string=string`. This happens because the list of arguments for `--apiserver-extra-args` expects `key=value` pairs and in this case `NamespacesExists` is considered as a key that is missing a value.

Alternatively, you can try separating the `key=value` pairs like so: `--apiserver-extra-args "enable-admission-plugins=LimitRanger,enable-admission-plugins=NamespaceExists"` but this will result in the key `enable-admission-plugins` only having the value of `NamespaceExists`.

A known workaround is to use the kubeadm [configuration file](#).

kube-proxy scheduled before node is initialized by cloud-controller-manager

In cloud provider scenarios, kube-proxy can end up being scheduled on new worker nodes before the cloud-controller-manager has initialized the node addresses. This causes kube-proxy to fail to pick up the node's IP address properly and has knock-on effects to the proxy function managing load balancers.

The following error can be seen in kube-proxy Pods:

```
server.go:610] Failed to retrieve node IP: host IP unknown;
known addresses: []
proxier.go:340] invalid nodeIP, initializing kube-proxy with
127.0.0.1 as nodeIP
```

A known solution is to patch the kube-proxy DaemonSet to allow scheduling it on control-plane nodes regardless of their conditions, keeping it off of other nodes until their initial guarding conditions abate:

```
kubectl -n kube-system patch ds kube-proxy -p='{ "spec":
{ "template": { "spec": { "tolerations": [ { "key":
"CriticalAddonsOnly", "operator": "Exists" }, { "effect":
"NoSchedule", "key": "node-role.kubernetes.io/
master" } ] } } } }'
```

The tracking issue for this problem is here.

The NodeRegistration.Taints field is omitted when marshalling kubeadm configuration

Note: This issue only applies to tools that marshal kubeadm types (e.g. to a YAML configuration file). It will be fixed in kubeadm API v1beta2.

By default, kubeadm applies the `node-role.kubernetes.io/master:NoSchedule` taint to control-plane nodes. If you prefer kubeadm to not taint the control-plane node, and set `InitConfiguration.NodeRegistration.Taints` to an empty slice, the field will be omitted when marshalling. When the field is omitted, kubeadm applies the default taint.

There are at least two workarounds:

1. *Use the `node-role.kubernetes.io/master:PreferNoSchedule` taint instead of an empty slice. Pods will get scheduled on masters, unless other nodes have capacity.*

2. *Remove the taint after kubeadm init exits:*

```
kubectl taint nodes NODE_NAME node-role.kubernetes.io/master:NoSchedule-
```

`/usr` is mounted read-only on nodes

On Linux distributions such as Fedora CoreOS, the directory `/usr` is mounted as a read-only filesystem. For flex-volume support, Kubernetes components like the kubelet and kube-controller-manager use the default path of `/usr/libexec/kubernetes/kubelet-plugins/volume/exec/`, yet the flex-volume directory must be writeable for the feature to work.

To workaround this issue you can configure the flex-volume directory using the kubeadm configuration file.

On the primary control-plane Node (created using `kubeadm init`) pass the following file using `--config`:

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: InitConfiguration
nodeRegistration:
  kubeletExtraArgs:
    volume-plugin-dir: "/opt/libexec/kubernetes/kubelet-plugins/volume/exec/"
---
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
controllerManager:
  extraArgs:
```

```
    flex-volume-plugin-dir: "/opt/libexec/kubernetes/kubelet-
plugins/volume/exec/"
```

On joining Nodes:

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: JoinConfiguration
nodeRegistration:
  kubeletExtraArgs:
    volume-plugin-dir: "/opt/libexec/kubernetes/kubelet-plugins/
volume/exec/"
```

Alternatively, you can modify `/etc/fstab` to make the `/usr` mount writeable, but please be advised that this is modifying a design principle of the Linux distribution.

`kubeadm upgrade plan` prints out `context deadline exceeded` error message

This error message is shown when upgrading a Kubernetes cluster with `kubeadm` in the case of running an external etcd. This is not a critical bug and happens because older versions of kubeadm perform a version check on the external etcd cluster. You can proceed with `kubeadm upgrade apply ...`.

This issue is fixed as of version 1.19.

`kubeadm reset` unmounts `/var/lib/kubelet`

If `/var/lib/kubelet` is being mounted, performing a `kubeadm reset` will effectively unmount it.

To workaround the issue, re-mount the `/var/lib/kubelet` directory after performing the `kubeadm reset` operation.

This is a regression introduced in kubeadm 1.15. The issue is fixed in 1.20.

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified September 02, 2020 at 10:38 AM PST: kubeadm reset unmounts /var/lib/kubelet (5c7280bb6)

Creating a cluster with kubeadm

Creating a minimum viable Kubernetes cluster that conforms to best practices. In fact, you can use `kubeadm` to set up a cluster that will pass the Kubernetes Conformance tests. `kubeadm` also supports other cluster lifecycle functions, such as bootstrap tokens and cluster upgrades.

The `kubeadm` tool is good if you need:

- *A simple way for you to try out Kubernetes, possibly for the first time.*
- *A way for existing users to automate setting up a cluster and test their application.*

- *A building block in other ecosystem and/or installer tools with a larger scope.*

You can install and use `kubeadm` on various machines: your laptop, a set of cloud servers, a Raspberry Pi, and more. Whether you're deploying into the cloud or on-premises, you can integrate `kubeadm` into provisioning systems such as Ansible or Terraform.

Before you begin

To follow this guide, you need:

- *One or more machines running a deb/rpm-compatible Linux OS; for example: Ubuntu or CentOS.*
- *2 GiB or more of RAM per machine--any less leaves little room for your apps.*
- *At least 2 CPUs on the machine that you use as a control-plane node.*
- *Full network connectivity among all machines in the cluster. You can use either a public or a private network.*

You also need to use a version of `kubeadm` that can deploy the version of Kubernetes that you want to use in your new cluster.

Kubernetes' version and version skew support policy applies to `kubeadm` as well as to Kubernetes overall. Check that policy to learn about what versions of Kubernetes and `kubeadm` are supported. This page is written for Kubernetes v1.20.

The `kubeadm` tool's overall feature state is General Availability (GA). Some sub-features are still under active development. The implementation of creating the cluster may change slightly as the tool evolves, but the overall implementation should be pretty stable.

> ***Note:** Any commands under `kubeadm alpha` are, by definition, supported on an alpha level.*

Objectives

- *Install a single control-plane Kubernetes cluster*
- *Install a Pod network on the cluster so that your Pods can talk to each other*

Instructions

Installing kubeadm on your hosts

See "Installing kubeadm".

> ***Note:***

If you have already installed kubeadm, run `apt-get update && apt-get upgrade` or `yum update` to get the latest version of kubeadm.

When you upgrade, the kubelet restarts every few seconds as it waits in a crashloop for kubeadm to tell it what to do. This crashloop is expected and normal. After you initialize your control-plane, the kubelet runs normally.

Initializing your control-plane node

The control-plane node is the machine where the control plane components run, including etcd (the cluster database) and the API Server (which the kubectl command line tool communicates with).

1. *(Recommended) If you have plans to upgrade this single control-plane `kubeadm` cluster to high availability you should specify the `--control-plane-endpoint` to set the shared endpoint for all control-plane nodes. Such an endpoint can be either a DNS name or an IP address of a load-balancer.*
2. *Choose a Pod network add-on, and verify whether it requires any arguments to be passed to `kubeadm init`. Depending on which third-party provider you choose, you might need to set the `--pod-network-cidr` to a provider-specific value. See Installing a Pod network add-on.*
3. *(Optional) Since version 1.14, `kubeadm` tries to detect the container runtime on Linux by using a list of well known domain socket paths. To use different container runtime or if there are more than one installed on the provisioned node, specify the `--cri-socket` argument to `kubeadm init`. See Installing runtime.*
4. *(Optional) Unless otherwise specified, `kubeadm` uses the network interface associated with the default gateway to set the advertise address for this particular control-plane node's API server. To use a different network interface, specify the `--apiserver-advertise-address=<ip-address>` argument to `kubeadm init`. To deploy an IPv6 Kubernetes cluster using IPv6 addressing, you must specify an IPv6 address, for example `--apiserver-advertise-address=fd00::101`*
5. *(Optional) Run `kubeadm config images pull` prior to `kubeadm init` to verify connectivity to the gcr.io container image registry.*

To initialize the control-plane node run:

```
kubeadm init <args>
```

Considerations about apiserver-advertise-address and ControlPlaneEndpoint

While `--apiserver-advertise-address` can be used to set the advertise address for this particular control-plane node's API server, `--control-plane-endpoint` can be used to set the shared endpoint for all control-plane nodes.

`--control-plane-endpoint` allows both IP addresses and DNS names that can map to IP addresses. Please contact your network administrator to evaluate possible solutions with respect to such mapping.

Here is an example mapping:

```
192.168.0.102 cluster-endpoint
```

Where `192.168.0.102` is the IP address of this node and `cluster-endpoint` is a custom DNS name that maps to this IP. This will allow you to pass `--control-plane-endpoint=cluster-endpoint` to `kubeadm init` and pass the same DNS name to `kubeadm join`. Later you can modify `cluster-endpoint` to point to the address of your load-balancer in an high availability scenario.

Turning a single control plane cluster created without `--control-plane-endpoint` into a highly available cluster is not supported by kubeadm.

More information

For more information about `kubeadm init` arguments, see the kubeadm reference guide.

To configure `kubeadm init` with a configuration file see Using kubeadm init with a configuration file.

To customize control plane components, including optional IPv6 assignment to liveness probe for control plane components and etcd server, provide extra arguments to each component as documented in custom arguments.

To run `kubeadm init` again, you must first tear down the cluster.

If you join a node with a different architecture to your cluster, make sure that your deployed DaemonSets have container image support for this architecture.

`kubeadm init` first runs a series of prechecks to ensure that the machine is ready to run Kubernetes. These prechecks expose warnings and exit on errors. `kubeadm init` then downloads and installs the cluster control plane components. This may take several minutes. After it finishes you should see:

```
Your Kubernetes control-plane has initialized successfully!

To start using your cluster, you need to run the following as a
regular user:

  mkdir -p $HOME/.kube
  sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
  sudo chown $(id -u):$(id -g) $HOME/.kube/config

You should now deploy a Pod network to the cluster.
Run "kubectl apply -f [podnetwork].yaml" with one of the options
listed at:
  /docs/concepts/cluster-administration/addons/
```

```
You can now join any number of machines by running the following
on each node
as root:

  kubeadm join <control-plane-host>:<control-plane-port> --token
<token> --discovery-token-ca-cert-hash sha256:<hash>
```

To make kubectl work for your non-root user, run these commands, which are also part of the `kubeadm init` output:

```
mkdir -p $HOME/.kube
sudo cp -i /etc/kubernetes/admin.conf $HOME/.kube/config
sudo chown $(id -u):$(id -g) $HOME/.kube/config
```

Alternatively, if you are the `root` user, you can run:

```
export KUBECONFIG=/etc/kubernetes/admin.conf
```

Make a record of the `kubeadm join` command that `kubeadm init` outputs. You need this command to join nodes to your cluster.

The token is used for mutual authentication between the control-plane node and the joining nodes. The token included here is secret. Keep it safe, because anyone with this token can add authenticated nodes to your cluster. These tokens can be listed, created, and deleted with the `kubeadm token` command. See the kubeadm reference guide.

Installing a Pod network add-on

> ***Caution:***
>
> *This section contains important information about networking setup and deployment order. Read all of this advice carefully before proceeding.*
>
> ***You must deploy a Container Network Interface (CNI) based Pod network add-on so that your Pods can communicate with each other. Cluster DNS (CoreDNS) will not start up before a network is installed.***
>
> - *Take care that your Pod network must not overlap with any of the host networks: you are likely to see problems if there is any overlap. (If you find a collision between your network plugin's preferred Pod network and some of your host networks, you should think of a suitable CIDR block to use instead, then use that during `kubeadm init` with `--pod-network-cidr` and as a replacement in your network plugin's YAML).*

- By default, `kubeadm` sets up your cluster to use and enforce use of RBAC (role based access control). Make sure that your Pod network plugin supports RBAC, and so do any manifests that you use to deploy it.
- If you want to use IPv6--either dual-stack, or single-stack IPv6 only networking--for your cluster, make sure that your Pod network plugin supports IPv6. IPv6 support was added to CNI in v0.6.0.

> **Note:** Currently Calico is the only CNI plugin that the kubeadm project performs e2e tests against. If you find an issue related to a CNI plugin you should log a ticket in its respective issue tracker instead of the kubeadm or kubernetes issue trackers.

Several external projects provide Kubernetes Pod networks using CNI, some of which also support Network Policy.

See a list of add-ons that implement the Kubernetes networking model.

You can install a Pod network add-on with the following command on the control-plane node or a node that has the kubeconfig credentials:

```
kubectl apply -f <add-on.yaml>
```

You can install only one Pod network per cluster.

Once a Pod network has been installed, you can confirm that it is working by checking that the CoreDNS Pod is `Running` in the output of `kubectl get pods --all-namespaces`. And once the CoreDNS Pod is up and running, you can continue by joining your nodes.

If your network is not working or CoreDNS is not in the `Running` state, check out the troubleshooting guide for `kubeadm`.

Control plane node isolation

By default, your cluster will not schedule Pods on the control-plane node for security reasons. If you want to be able to schedule Pods on the control-plane node, for example for a single-machine Kubernetes cluster for development, run:

```
kubectl taint nodes --all node-role.kubernetes.io/master-
```

With output looking something like:

```
node "test-01" untainted
taint "node-role.kubernetes.io/master:" not found
taint "node-role.kubernetes.io/master:" not found
```

This will remove the `node-role.kubernetes.io/master` taint from any nodes that have it, including the control-plane node, meaning that the scheduler will then be able to schedule Pods everywhere.

Joining your nodes

The nodes are where your workloads (containers and Pods, etc) run. To add new nodes to your cluster do the following for each machine:

- *SSH to the machine*
- *Become root (e.g. `sudo su -`)*
- *Run the command that was output by `kubeadm init`. For example:*

```
kubeadm join --token <token> <control-plane-host>:<control-plane-port> --discovery-token-ca-cert-hash sha256:<hash>
```

If you do not have the token, you can get it by running the following command on the control-plane node:

```
kubeadm token list
```

The output is similar to this:

```
TOKEN                    TTL  EXPIRES
USAGES           DESCRIPTION            EXTRA GROUPS
8ewj1p.9r9hcjoqgajrj4gi  23h  2018-06-12T02:51:28Z
authentication,  The default bootstrap  system:

signing          token generated by     bootstrappers:

   'kubeadm init'.        kubeadm:

                          default-node-token
```

By default, tokens expire after 24 hours. If you are joining a node to the cluster after the current token has expired, you can create a new token by running the following command on the control-plane node:

```
kubeadm token create
```

The output is similar to this:

```
5didvk.d09sbcov8ph2amjw
```

If you don't have the value of `--discovery-token-ca-cert-hash`, you can get it by running the following command chain on the control-plane node:

```
openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl
rsa -pubin -outform der 2>/dev/null | \
   openssl dgst -sha256 -hex | sed 's/^.* //'
```

The output is similar to:

```
8cb2de97839780a412b93877f8507ad6c94f73add17d5d7058e91741c9d5ec78
```

> ***Note:*** *To specify an IPv6 tuple for `<control-plane-host>:<control-plane-port>`, IPv6 address must be enclosed in square brackets, for example: `[fd00::101]:2073`.*

The output should look something like:

```
[preflight] Running pre-flight checks

... (log output of join workflow) ...

Node join complete:
* Certificate signing request sent to control-plane and response
  received.
* Kubelet informed of new secure connection details.

Run 'kubectl get nodes' on control-plane to see this machine
join.
```

A few seconds later, you should notice this node in the output from `kubectl get nodes` when run on the control-plane node.

(Optional) Controlling your cluster from machines other than the control-plane node

In order to get a kubectl on some other computer (e.g. laptop) to talk to your cluster, you need to copy the administrator kubeconfig file from your control-plane node to your workstation like this:

```
scp root@<control-plane-host>:/etc/kubernetes/admin.conf .
kubectl --kubeconfig ./admin.conf get nodes
```

> ***Note:***
>
> *The example above assumes SSH access is enabled for root. If that is not the case, you can copy the `admin.conf` file to be accessible by some other user and `scp` using that other user instead.*
>
> *The `admin.conf` file gives the user superuser privileges over the cluster. This file should be used sparingly. For normal users, it's recommended to generate an unique credential to which you grant privileges. You can do this with the `kubeadm alpha kubeconfig user --client-name <CN>` command. That command will print out a KubeConfig file to STDOUT which you should save to a file and distribute to your user. After that, grant privileges by using `kubectl create (cluster)rolebinding`.*

(Optional) Proxying API Server to localhost

If you want to connect to the API Server from outside the cluster you can use `kubectl proxy`:

```
scp root@<control-plane-host>:/etc/kubernetes/admin.conf .
kubectl --kubeconfig ./admin.conf proxy
```

You can now access the API Server locally at `http://localhost:8001/api/v1`

Clean up

If you used disposable servers for your cluster, for testing, you can switch those off and do no further clean up. You can use `kubectl config delete-cluster` to delete your local references to the cluster.

However, if you want to deprovision your cluster more cleanly, you should first drain the node and make sure that the node is empty, then deconfigure the node.

Remove the node

Talking to the control-plane node with the appropriate credentials, run:

```
kubectl drain <node name> --delete-local-data --force --ignore-daemonsets
```

Before removing the node, reset the state installed by `kubeadm`:

```
kubeadm reset
```

The reset process does not reset or clean up iptables rules or IPVS tables. If you wish to reset iptables, you must do so manually:

```
iptables -F && iptables -t nat -F && iptables -t mangle -F && iptables -X
```

If you want to reset the IPVS tables, you must run the following command:

```
ipvsadm -C
```

Now remove the node:

```
kubectl delete node <node name>
```

If you wish to start over simply run `kubeadm init` or `kubeadm join` with the appropriate arguments.

Clean up the control plane

You can use `kubeadm reset` on the control plane host to trigger a best-effort clean up.

See the `kubeadm reset` reference documentation for more information about this subcommand and its options.

What's next

- *Verify that your cluster is running properly with Sonobuoy*
- *See Upgrading kubeadm clusters for details about upgrading your cluster using `kubeadm`.*
- *Learn about advanced `kubeadm` usage in the kubeadm reference documentation*
- *Learn more about Kubernetes concepts and `kubectl`.*
- *See the Cluster Networking page for a bigger list of Pod network add-ons.*
- *See the list of add-ons to explore other add-ons, including tools for logging, monitoring, network policy, visualization & control of your Kubernetes cluster.*
- *Configure how your cluster handles logs for cluster events and from applications running in Pods. See Logging Architecture for an overview of what is involved.*

Feedback

- *For bugs, visit the kubeadm GitHub issue tracker*
- *For support, visit the #kubeadm Slack channel*
- *General SIG Cluster Lifecycle development Slack channel: #sig-cluster-lifecycle*
- *SIG Cluster Lifecycle SIG information*
- *SIG Cluster Lifecycle mailing list: kubernetes-sig-cluster-lifecycle*

Version skew policy

The `kubeadm` tool of version v1.20 may deploy clusters with a control plane of version v1.20 or v1.19. `kubeadm` v1.20 can also upgrade an existing kubeadm-created cluster of version v1.19.

Due to that we can't see into the future, kubeadm CLI v1.20 may or may not be able to deploy v1.21 clusters.

These resources provide more information on supported version skew between kubelets and the control plane, and other Kubernetes components:

- *Kubernetes version and version-skew policy*
- *Kubeadm-specific installation guide*

Limitations

Cluster resilience

The cluster created here has a single control-plane node, with a single etcd database running on it. This means that if the control-plane node fails, your cluster may lose data and may need to be recreated from scratch.

Workarounds:

- Regularly back up etcd. The etcd data directory configured by kubeadm is at `/var/lib/etcd` on the control-plane node.
- Use multiple control-plane nodes. You can read Options for Highly Available topology to pick a cluster topology that provides high-availability.

Platform compatibility

kubeadm deb/rpm packages and binaries are built for amd64, arm (32-bit), arm64, ppc64le, and s390x following the multi-platform proposal.

Multiplatform container images for the control plane and addons are also supported since v1.12.

Only some of the network providers offer solutions for all platforms. Please consult the list of network providers above or the documentation from each provider to figure out whether the provider supports your chosen platform.

Troubleshooting

If you are running into difficulties with kubeadm, please consult our troubleshooting docs.

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified November 05, 2020 at 7:44 PM PST: kubeadm: remove general output from "kubeadm init" (dd402eff7)
Edit this page Create child page Create an issue

Customizing control plane configuration with kubeadm

FEATURE STATE: `Kubernetes v1.12 [stable]`

The kubeadm `ClusterConfiguration` object exposes the field `extraArgs` that can override the default flags passed to control plane components such as the APIServer, ControllerManager and Scheduler. The components are defined using the following fields:

- `apiServer`
- `controllerManager`
- `scheduler`

The `extraArgs` field consist of `key: value` pairs. To override a flag for a control plane component:

1. Add the appropriate fields to your configuration.
2. Add the flags to override to the field.
3. Run `kubeadm init` with `--config <YOUR CONFIG YAML>`.

For more details on each field in the configuration you can navigate to our API reference pages.

> ***Note:*** You can generate a `ClusterConfiguration` object with default values by running `kubeadm config print init-defaults` and saving the output to a file of your choice.

APIServer flags

For details, see the reference documentation for kube-apiserver.

Example usage:

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
kubernetesVersion: v1.16.0
apiServer:
  extraArgs:
    advertise-address: 192.168.0.103
    anonymous-auth: "false"
    enable-admission-plugins: AlwaysPullImages,DefaultStorageClas
s
    audit-log-path: /home/johndoe/audit.log
```

ControllerManager flags

For details, see the reference documentation for kube-controller-manager.

Example usage:

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
kubernetesVersion: v1.16.0
controllerManager:
  extraArgs:
    cluster-signing-key-file: /home/johndoe/keys/ca.key
    bind-address: 0.0.0.0
    deployment-controller-sync-period: "50"
```

Scheduler flags

For details, see the reference documentation for kube-scheduler.

Example usage:

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
kubernetesVersion: v1.16.0
scheduler:
  extraArgs:
    address: 0.0.0.0
    config: /home/johndoe/schedconfig.yaml
    kubeconfig: /home/johndoe/kubeconfig.yaml
```

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified May 30, 2020 at 3:10 PM PST: add en pages (ecc27bbbe)
Edit this page Create child page Create an issue

- *APIServer flags*
- *ControllerManager flags*
- *Scheduler flags*

Options for Highly Available topology

This page explains the two options for configuring the topology of your highly available (HA) Kubernetes clusters.

You can set up an HA cluster:

- *With stacked control plane nodes, where etcd nodes are colocated with control plane nodes*
- *With external etcd nodes, where etcd runs on separate nodes from the control plane*

You should carefully consider the advantages and disadvantages of each topology before setting up an HA cluster.

> ***Note:** kubeadm bootstraps the etcd cluster statically. Read the etcd Clustering Guide for more details.*

Stacked etcd topology

A stacked HA cluster is a topology where the distributed data storage cluster provided by etcd is stacked on top of the cluster formed by the nodes managed by kubeadm that run control plane components.

Each control plane node runs an instance of the `kube-apiserver`, `kube-scheduler`, and `kube-controller-manager`. The `kube-apiserver` is exposed to worker nodes using a load balancer.

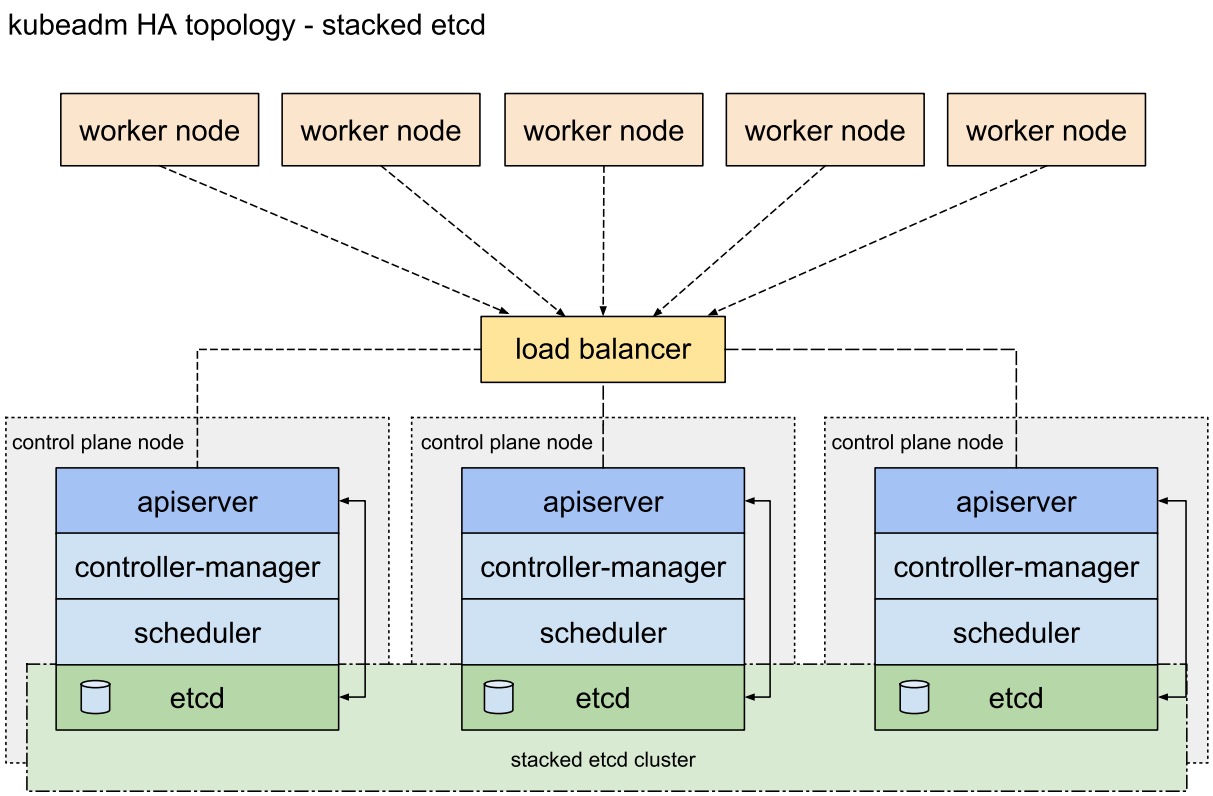
Each control plane node creates a local etcd member and this etcd member communicates only with the `kube-apiserver` of this node. The same applies to the local `kube-controller-manager` and `kube-scheduler` instances.

This topology couples the control planes and etcd members on the same nodes. It is simpler to set up than a cluster with external etcd nodes, and simpler to manage for replication.

However, a stacked cluster runs the risk of failed coupling. If one node goes down, both an etcd member and a control plane instance are lost, and redundancy is compromised. You can mitigate this risk by adding more control plane nodes.

You should therefore run a minimum of three stacked control plane nodes for an HA cluster.

This is the default topology in kubeadm. A local etcd member is created automatically on control plane nodes when using `kubeadm init` and `kubeadm join --control-plane`.



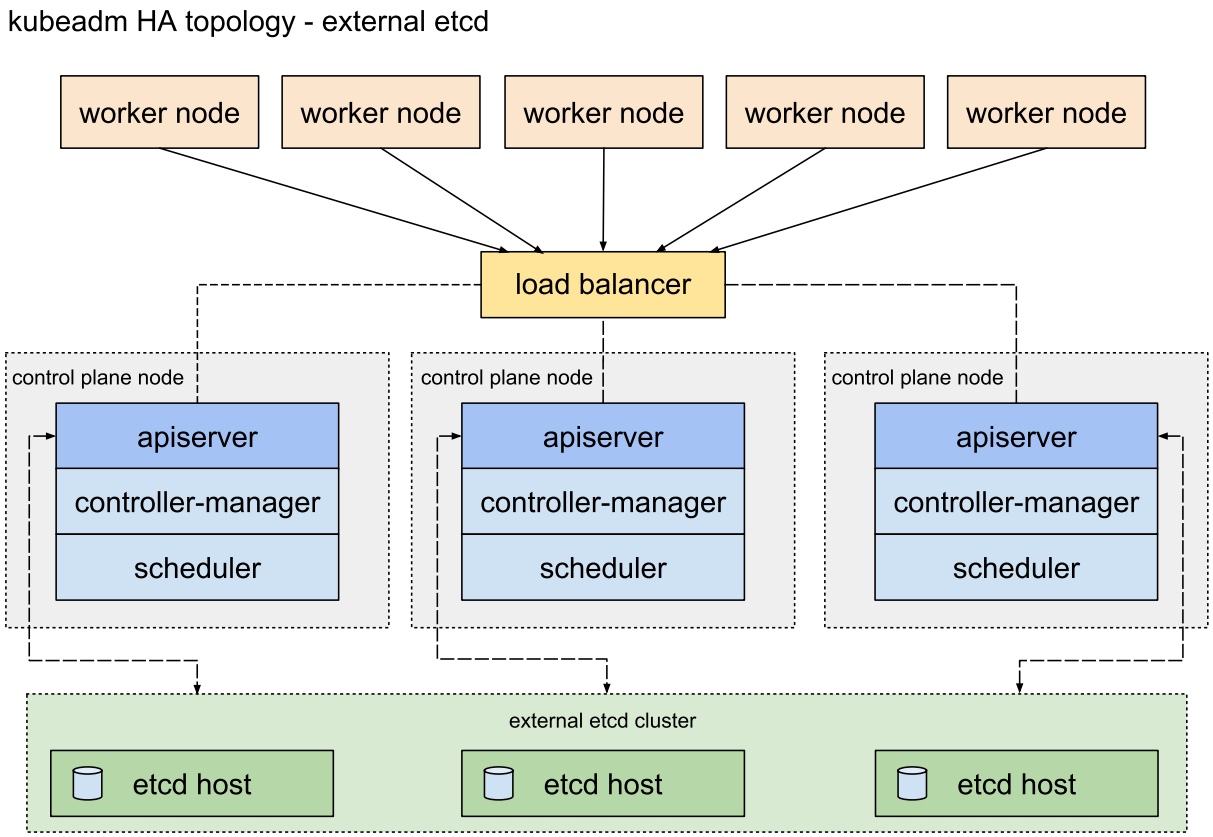
External etcd topology

An HA cluster with external etcd is a topology where the distributed data storage cluster provided by etcd is external to the cluster formed by the nodes that run control plane components.

Like the stacked etcd topology, each control plane node in an external etcd topology runs an instance of the `kube-apiserver`, `kube-scheduler`, and `kube-controller-manager`. And the `kube-apiserver` is exposed to worker nodes using a load balancer. However, etcd members run on separate hosts, and each etcd host communicates with the `kube-apiserver` of each control plane node.

This topology decouples the control plane and etcd member. It therefore provides an HA setup where losing a control plane instance or an etcd member has less impact and does not affect the cluster redundancy as much as the stacked HA topology.

However, this topology requires twice the number of hosts as the stacked HA topology. A minimum of three hosts for control plane nodes and three hosts for etcd nodes are required for an HA cluster with this topology.



What's next

- *Set up a highly available cluster with kubeadm*

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified May 30, 2020 at 3:10 PM PST: add en pages (ecc27bbbe)

Creating Highly Available clusters with kubeadm

This page explains two different approaches to setting up a highly available Kubernetes cluster using kubeadm:

- With stacked control plane nodes. This approach requires less infrastructure. The etcd members and control plane nodes are co-located.
- With an external etcd cluster. This approach requires more infrastructure. The control plane nodes and etcd members are separated.

Before proceeding, you should carefully consider which approach best meets the needs of your applications and environment. This comparison topic outlines the advantages and disadvantages of each.

If you encounter issues with setting up the HA cluster, please provide us with feedback in the kubeadm issue tracker.

See also The upgrade documentation.

> **Caution:** This page does not address running your cluster on a cloud provider. In a cloud environment, neither approach documented here works with Service objects of type LoadBalancer, or with dynamic PersistentVolumes.

Before you begin

For both methods you need this infrastructure:

- Three machines that meet kubeadm's minimum requirements for the control-plane nodes
- Three machines that meet kubeadm's minimum requirements for the workers
- Full network connectivity between all machines in the cluster (public or private network)
- sudo privileges on all machines
- SSH access from one device to all nodes in the system
- `kubeadm` and `kubelet` installed on all machines. `kubectl` is optional.

For the external etcd cluster only, you also need:

- Three additional machines for etcd members

First steps for both methods

Create load balancer for kube-apiserver

> **Note:** There are many configurations for load balancers. The following example is only one option. Your cluster requirements may need a different configuration.

1. Create a kube-apiserver load balancer with a name that resolves to DNS.
    - In a cloud environment you should place your control plane nodes behind a TCP forwarding load balancer. This load balancer distributes traffic to all healthy control plane nodes in its target list. The health check for an apiserver is a TCP check on the port the kube-apiserver listens on (default value `:6443`).
    - It is not recommended to use an IP address directly in a cloud environment.
    - The load balancer must be able to communicate with all control plane nodes on the apiserver port. It must also allow incoming traffic on its listening port.
    - Make sure the address of the load balancer always matches the address of kubeadm's `ControlPlaneEndpoint`.
    - Read the Options for Software Load Balancing guide for more details.
2. Add the first control plane nodes to the load balancer and test the connection:

    ```
    nc -v LOAD_BALANCER_IP PORT
    ```

    - A connection refused error is expected because the apiserver is not yet running. A timeout, however, means the load balancer cannot communicate with the control plane node. If a timeout occurs, reconfigure the load balancer to communicate with the control plane node.
3. Add the remaining control plane nodes to the load balancer target group.

Stacked control plane and etcd nodes

Steps for the first control plane node

1. Initialize the control plane:

```
sudo kubeadm init --control-plane-endpoint "LOAD_BALANCER_DNS
:LOAD_BALANCER_PORT" --upload-certs
```

- *You can use the `--kubernetes-version` flag to set the Kubernetes version to use. It is recommended that the versions of kubeadm, kubelet, kubectl and Kubernetes match.*

- *The `--control-plane-endpoint` flag should be set to the address or DNS and port of the load balancer.*

- *The `--upload-certs` flag is used to upload the certificates that should be shared across all the control-plane instances to the cluster. If instead, you prefer to copy certs across control-plane nodes manually or using automation tools, please remove this flag and refer to Manual certificate distribution section below.*

  ***Note:** The `kubeadm init` flags `--config` and `--certificate-key` cannot be mixed, therefore if you want to use the kubeadm configuration you must add the `certificateKey` field in the appropriate config locations (under `InitConfiguration` and `JoinConfiguration: controlPlane`).*

  ***Note:** Some CNI network plugins require additional configuration, for example specifying the pod IP CIDR, while others do not. See the CNI network documentation. To add a pod CIDR pass the flag `--pod-network-cidr`, or if you are using a kubeadm configuration file set the `podSubnet` field under the `networking` object of `ClusterConfiguration`.*

- *The output looks similar to:*

```
...
You can now join any number of control-plane node by
running the following command on each as a root:
    kubeadm join 192.168.0.200:6443 --token
9vr73a.a8uxyaju799qwdjv --discovery-token-ca-cert-hash
sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720e
ff5359e26aec866 --control-plane --certificate-key
f8902e114ef118304e561c3ecd4d0b543adc226b7a07f675f5656418
5ffe0c07

Please note that the certificate-key gives access to
cluster sensitive data, keep it secret!
As a safeguard, uploaded-certs will be deleted in two
hours; If necessary, you can use kubeadm init phase
upload-certs to reload certs afterward.

Then you can join any number of worker nodes by running
the following on each as root:
    kubeadm join 192.168.0.200:6443 --token
9vr73a.a8uxyaju799qwdjv --discovery-token-ca-cert-hash
sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720e
ff5359e26aec866
```

- *Copy this output to a text file. You will need it later to join control plane and worker nodes to the cluster.*
- *When `--upload-certs` is used with `kubeadm init`, the certificates of the primary control plane are encrypted and uploaded in the `kubeadm-certs` Secret.*
- *To re-upload the certificates and generate a new decryption key, use the following command on a control plane node that is already joined to the cluster:*

  ```
  sudo kubeadm init phase upload-certs --upload-certs
  ```

- *You can also specify a custom `--certificate-key` during `init` that can later be used by `join`. To generate such a key you can use the following command:*

  ```
  kubeadm certs certificate-key
  ```

  Note: *The `kubeadm-certs` Secret and decryption key expire after two hours.*

  Caution: *As stated in the command output, the certificate key gives access to cluster sensitive data, keep it secret!*

2. *Apply the CNI plugin of your choice: [Follow these instructions] to install the CNI provider. Make sure the configuration corresponds to the Pod CIDR specified in the kubeadm configuration file if applicable.*

   In this example we are using Weave Net:

   ```
   kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')"
   ```

3. *Type the following and watch the pods of the control plane components get started:*

   ```
   kubectl get pod -n kube-system -w
   ```

Steps for the rest of the control plane nodes

Note: *Since kubeadm version 1.15 you can join multiple control-plane nodes in parallel. Prior to this version, you must join new control plane nodes sequentially, only after the first node has finished initializing.*

For each additional control plane node you should:

1. *Execute the join command that was previously given to you by the `kubeadm init` output on the first node. It should look something like this:*

   ```
   sudo kubeadm join 192.168.0.200:6443 --token 9vr73a.a8uxyaju799qwdjv --discovery-token-ca-cert-hash sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720eff535
   ```

```
9e26aec866 --control-plane --certificate-key
f8902e114ef118304e561c3ecd4d0b543adc226b7a07f675f56564185ffe0
c07
```

- The `--control-plane` flag tells `kubeadm join` to create a new control plane.
- The `--certificate-key ...` will cause the control plane certificates to be downloaded from the `kubeadm-certs` Secret in the cluster and be decrypted using the given key.

External etcd nodes

Setting up a cluster with external etcd nodes is similar to the procedure used for stacked etcd with the exception that you should setup etcd first, and you should pass the etcd information in the kubeadm config file.

Set up the etcd cluster

1. Follow [these instructions] to set up the etcd cluster.

2. Setup SSH as described [here].

3. Copy the following files from any etcd node in the cluster to the first control plane node:

```
export CONTROL_PLANE="ubuntu@10.0.0.7"
scp /etc/kubernetes/pki/etcd/ca.crt "${CONTROL_PLANE}":
scp /etc/kubernetes/pki/apiserver-etcd-client.crt "${CONTROL_
PLANE}":
scp /etc/kubernetes/pki/apiserver-etcd-client.key "${CONTROL_
PLANE}":
```

   - Replace the value of `CONTROL_PLANE` with the `user@host` of the first control-plane node.

Set up the first control plane node

1. Create a file called `kubeadm-config.yaml` with the following contents:

```
apiVersion: kubeadm.k8s.io/v1beta2
kind: ClusterConfiguration
kubernetesVersion: stable
controlPlaneEndpoint: "LOAD_BALANCER_DNS:LOAD_BALANCER_PORT"
etcd:
    external:
        endpoints:
        - https://ETCD_0_IP:2379
        - https://ETCD_1_IP:2379
        - https://ETCD_2_IP:2379
        caFile: /etc/kubernetes/pki/etcd/ca.crt
        certFile: /etc/kubernetes/pki/apiserver-etcd-
client.crt
```

```
        keyFile: /etc/kubernetes/pki/apiserver-etcd-
client.key
```

Note: *The difference between stacked etcd and external etcd here is that the external etcd setup requires a configuration file with the etcd endpoints under the* `external` *object for* `etcd`*. In the case of the stacked etcd topology this is managed automatically.*

- *Replace the following variables in the config template with the appropriate values for your cluster:*

```
- `LOAD_BALANCER_DNS`
- `LOAD_BALANCER_PORT`
- `ETCD_0_IP`
- `ETCD_1_IP`
- `ETCD_2_IP`
```

The following steps are similar to the stacked etcd setup:

1. *Run* `sudo kubeadm init --config kubeadm-config.yaml --upload-certs` *on this node.*
2. *Write the output join commands that are returned to a text file for later use.*
3. *Apply the CNI plugin of your choice. The given example is for Weave Net:*

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-
version=$(kubectl version | base64 | tr -d '\n')"
```

Steps for the rest of the control plane nodes

The steps are the same as for the stacked etcd setup:

- *Make sure the first control plane node is fully initialized.*
- *Join each control plane node with the join command you saved to a text file. It's recommended to join the control plane nodes one at a time.*
- *Don't forget that the decryption key from* `--certificate-key` *expires after two hours, by default.*

Common tasks after bootstrapping control plane

Install workers

Worker nodes can be joined to the cluster with the command you stored previously as the output from the `kubeadm init` *command:*

```
sudo kubeadm join 192.168.0.200:6443 --token
9vr73a.a8uxyaju799qwdjv --discovery-token-ca-cert-hash
```

```
sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720eff5359e26aec866
```

Manual certificate distribution

If you choose to not use `kubeadm init` with the `--upload-certs` flag this means that you are going to have to manually copy the certificates from the primary control plane node to the joining control plane nodes.

There are many ways to do this. In the following example we are using `ssh` and `scp`:

SSH is required if you want to control all nodes from a single machine.

1. Enable ssh-agent on your main device that has access to all other nodes in the system:

   ```
   eval $(ssh-agent)
   ```

2. Add your SSH identity to the session:

   ```
   ssh-add ~/.ssh/path_to_private_key
   ```

3. SSH between nodes to check that the connection is working correctly.

   - When you SSH to any node, make sure to add the `-A` flag:

     ```
     ssh -A 10.0.0.7
     ```

   - When using sudo on any node, make sure to preserve the environment so SSH forwarding works:

     ```
     sudo -E -s
     ```

4. After configuring SSH on all the nodes you should run the following script on the first control plane node after running `kubeadm init`. This script will copy the certificates from the first control plane node to the other control plane nodes:

   In the following example, replace `CONTROL_PLANE_IPS` with the IP addresses of the other control plane nodes.

   ```
   USER=ubuntu # customizable
   CONTROL_PLANE_IPS="10.0.0.7 10.0.0.8"
   for host in ${CONTROL_PLANE_IPS}; do
       scp /etc/kubernetes/pki/ca.crt "${USER}"@$host:
       scp /etc/kubernetes/pki/ca.key "${USER}"@$host:
       scp /etc/kubernetes/pki/sa.key "${USER}"@$host:
       scp /etc/kubernetes/pki/sa.pub "${USER}"@$host:
       scp /etc/kubernetes/pki/front-proxy-ca.crt "${USER}"@$hos
   t:
       scp /etc/kubernetes/pki/front-proxy-ca.key "${USER}"@$hos
   t:
   ```

```
    scp /etc/kubernetes/pki/etcd/ca.crt "${USER}"@$host:etcd-ca.crt
    # Quote this line if you are using external etcd
    scp /etc/kubernetes/pki/etcd/ca.key "${USER}"@$host:etcd-ca.key
done
```

> **Caution:** Copy only the certificates in the above list. kubeadm will take care of generating the rest of the certificates with the required SANs for the joining control-plane instances. If you copy all the certificates by mistake, the creation of additional nodes could fail due to a lack of required SANs.

5. Then on each joining control plane node you have to run the following script before running `kubeadm join`. This script will move the previously copied certificates from the home directory to `/etc/kubernetes/pki`:

```
USER=ubuntu # customizable
mkdir -p /etc/kubernetes/pki/etcd
mv /home/${USER}/ca.crt /etc/kubernetes/pki/
mv /home/${USER}/ca.key /etc/kubernetes/pki/
mv /home/${USER}/sa.pub /etc/kubernetes/pki/
mv /home/${USER}/sa.key /etc/kubernetes/pki/
mv /home/${USER}/front-proxy-ca.crt /etc/kubernetes/pki/
mv /home/${USER}/front-proxy-ca.key /etc/kubernetes/pki/
mv /home/${USER}/etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
# Quote this line if you are using external etcd
mv /home/${USER}/etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
```

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified November 12, 2020 at 9:28 PM PST: kubeadm: promote the "kubeadm certs" command to GA (#24410) (d0c6d303c)
Edit this page Create child page Create an issue

Set up a High Availability etcd cluster with kubeadm

> **Note:** While kubeadm is being used as the management tool for external etcd nodes in this guide, please note that kubeadm does not plan to support certificate rotation or upgrades for such nodes. The long term plan is to empower the tool etcdadm to manage these aspects.

Kubeadm defaults to running a single member etcd cluster in a static pod managed by the kubelet on the control plane node. This is not a high availability setup as the etcd cluster contains only one member and cannot sustain any members becoming unavailable. This task walks through the process of creating a high availability etcd cluster of three members that can be used as an external etcd when using kubeadm to set up a kubernetes cluster.

Before you begin

- Three hosts that can talk to each other over ports 2379 and 2380. This document assumes these default ports. However, they are configurable through the kubeadm config file.
- Each host must have docker, kubelet, and kubeadm installed.
- Each host should have access to the Kubernetes container image registry (`k8s.gcr.io`) or list/pull the required etcd image using `kubeadm config images list/pull`. This guide will setup etcd instances as static pods managed by a kubelet.
- Some infrastructure to copy files between hosts. For example `ssh` and `scp` can satisfy this requirement.

Setting up the cluster

The general approach is to generate all certs on one node and only distribute the necessary files to the other nodes.

> **Note:** kubeadm contains all the necessary crytographic machinery to generate the certificates described below; no other cryptographic tooling is required for this example.

1. Configure the kubelet to be a service manager for etcd.

Note: *You must do this on every host where etcd should be running.*

Since etcd was created first, you must override the service priority by creating a new unit file that has higher precedence than the kubeadm-provided kubelet unit file.

```
cat << EOF > /etc/systemd/system/kubelet.service.d/20-etcd-service-manager.conf
[Service]
ExecStart=
#  Replace "systemd" with the cgroup driver of your container runtime. The default value in the kubelet is "cgroupfs".
ExecStart=/usr/bin/kubelet --address=127.0.0.1 --pod-manifest-path=/etc/kubernetes/manifests --cgroup-driver=systemd
Restart=always
EOF

systemctl daemon-reload
systemctl restart kubelet
```

Check the kubelet status to ensure it is running.

```
systemctl status kubelet
```

2. *Create configuration files for kubeadm.*

   Generate one kubeadm configuration file for each host that will have an etcd member running on it using the following script.

```
# Update HOST0, HOST1, and HOST2 with the IPs or resolvable names of your hosts
export HOST0=10.0.0.6
export HOST1=10.0.0.7
export HOST2=10.0.0.8

# Create temp directories to store files that will end up on other hosts.
mkdir -p /tmp/${HOST0}/ /tmp/${HOST1}/ /tmp/${HOST2}/

ETCDHOSTS=(${HOST0} ${HOST1} ${HOST2})
NAMES=("infra0" "infra1" "infra2")

for i in "${!ETCDHOSTS[@]}"; do
HOST=${ETCDHOSTS[$i]}
NAME=${NAMES[$i]}
cat << EOF > /tmp/${HOST}/kubeadmcfg.yaml
apiVersion: "kubeadm.k8s.io/v1beta2"
```

```
kind: ClusterConfiguration
etcd:
    local:
        serverCertSANs:
        - "${HOST}"
        peerCertSANs:
        - "${HOST}"
        extraArgs:
            initial-cluster: ${NAMES[0]}=https://$
{ETCDHOSTS[0]}:2380,${NAMES[1]}=https://${ETCDHOSTS[1]}:
2380,${NAMES[2]}=https://${ETCDHOSTS[2]}:2380
            initial-cluster-state: new
            name: ${NAME}
            listen-peer-urls: https://${HOST}:2380
            listen-client-urls: https://${HOST}:2379
            advertise-client-urls: https://${HOST}:2379
            initial-advertise-peer-urls: https://${HOST}:2380
EOF
done
```

3. *Generate the certificate authority*

   If you already have a CA then the only action that is copying the CA's `crt` and `key` file to `/etc/kubernetes/pki/etcd/ca.crt` and `/etc/kubernetes/pki/etcd/ca.key`. After those files have been copied, proceed to the next step, "Create certificates for each member".

   If you do not already have a CA then run this command on `$HOST0` (where you generated the configuration files for kubeadm).

   ```
   kubeadm init phase certs etcd-ca
   ```

   This creates two files

   - `/etc/kubernetes/pki/etcd/ca.crt`
   - `/etc/kubernetes/pki/etcd/ca.key`

4. *Create certificates for each member*

   ```
   kubeadm init phase certs etcd-server --config=/tmp/${HOST2}/
   kubeadmcfg.yaml
   kubeadm init phase certs etcd-peer --config=/tmp/${HOST2}/
   kubeadmcfg.yaml
   kubeadm init phase certs etcd-healthcheck-client --config=/
   tmp/${HOST2}/kubeadmcfg.yaml
   kubeadm init phase certs apiserver-etcd-client --config=/tmp/
   ${HOST2}/kubeadmcfg.yaml
   cp -R /etc/kubernetes/pki /tmp/${HOST2}/
   # cleanup non-reusable certificates
   find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key
   -type f -delete

   kubeadm init phase certs etcd-server --config=/tmp/${HOST1}/
   ```

```
kubeadmcfg.yaml
kubeadm init phase certs etcd-peer --config=/tmp/${HOST1}/
kubeadmcfg.yaml
kubeadm init phase certs etcd-healthcheck-client --config=/
tmp/${HOST1}/kubeadmcfg.yaml
kubeadm init phase certs apiserver-etcd-client --config=/tmp/
${HOST1}/kubeadmcfg.yaml
cp -R /etc/kubernetes/pki /tmp/${HOST1}/
find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key
-type f -delete

kubeadm init phase certs etcd-server --config=/tmp/${HOST0}/
kubeadmcfg.yaml
kubeadm init phase certs etcd-peer --config=/tmp/${HOST0}/
kubeadmcfg.yaml
kubeadm init phase certs etcd-healthcheck-client --config=/
tmp/${HOST0}/kubeadmcfg.yaml
kubeadm init phase certs apiserver-etcd-client --config=/tmp/
${HOST0}/kubeadmcfg.yaml
# No need to move the certs because they are for HOST0

# clean up certs that should not be copied off this host
find /tmp/${HOST2} -name ca.key -type f -delete
find /tmp/${HOST1} -name ca.key -type f -delete
```

5. *Copy certificates and kubeadm configs*

   The certificates have been generated and now they must be moved to their respective hosts.

```
USER=ubuntu
HOST=${HOST1}
scp -r /tmp/${HOST}/* ${USER}@${HOST}:
ssh ${USER}@${HOST}
USER@HOST $ sudo -Es
root@HOST $ chown -R root:root pki
root@HOST $ mv pki /etc/kubernetes/
```

6. *Ensure all expected files exist*

   The complete list of required files on `$HOST0` is:

```
/tmp/${HOST0}
â””â”€â”€ kubeadmcfg.yaml
---
/etc/kubernetes/pki
â”œâ”€â”€ apiserver-etcd-client.crt
â”œâ”€â”€ apiserver-etcd-client.key
â””â”€â”€ etcd
    â”œâ”€â”€ ca.crt
    â”œâ”€â”€ ca.key
    â”œâ”€â”€ healthcheck-client.crt
    â”œâ”€â”€ healthcheck-client.key
```

```
    â”œâ”€â”€ peer.crt
    â”œâ”€â”€ peer.key
    â”œâ”€â”€ server.crt
    â””â”€â”€ server.key
```

On `$HOST1`:

```
$HOME
â””â”€â”€ kubeadmcfg.yaml
---
/etc/kubernetes/pki
â”œâ”€â”€ apiserver-etcd-client.crt
â”œâ”€â”€ apiserver-etcd-client.key
â””â”€â”€ etcd
    â”œâ”€â”€ ca.crt
    â”œâ”€â”€ healthcheck-client.crt
    â”œâ”€â”€ healthcheck-client.key
    â”œâ”€â”€ peer.crt
    â”œâ”€â”€ peer.key
    â”œâ”€â”€ server.crt
    â””â”€â”€ server.key
```

On `$HOST2`

```
$HOME
â””â”€â”€ kubeadmcfg.yaml
---
/etc/kubernetes/pki
â”œâ”€â”€ apiserver-etcd-client.crt
â”œâ”€â”€ apiserver-etcd-client.key
â””â”€â”€ etcd
    â”œâ”€â”€ ca.crt
    â”œâ”€â”€ healthcheck-client.crt
    â”œâ”€â”€ healthcheck-client.key
    â”œâ”€â”€ peer.crt
    â”œâ”€â”€ peer.key
    â”œâ”€â”€ server.crt
    â””â”€â”€ server.key
```

7. *Create the static pod manifests*

   Now that the certificates and configs are in place it's time to create the manifests. On each host run the `kubeadm` command to generate a static manifest for etcd.

   ```
   root@HOST0 $ kubeadm init phase etcd local --config=/tmp/${HOST0}/kubeadmcfg.yaml
   root@HOST1 $ kubeadm init phase etcd local --config=/home/ubuntu/kubeadmcfg.yaml
   root@HOST2 $ kubeadm init phase etcd local --config=/home/ubuntu/kubeadmcfg.yaml
   ```

8. *Optional: Check the cluster health*

```
docker run --rm -it \
--net host \
-v /etc/kubernetes:/etc/kubernetes k8s.gcr.io/etcd:${ETCD_TAG} etcdctl \
--cert /etc/kubernetes/pki/etcd/peer.crt \
--key /etc/kubernetes/pki/etcd/peer.key \
--cacert /etc/kubernetes/pki/etcd/ca.crt \
--endpoints https://${HOST0}:2379 endpoint health --cluster
...
https://[HOST0 IP]:2379 is healthy: successfully committed proposal: took = 16.283339ms
https://[HOST1 IP]:2379 is healthy: successfully committed proposal: took = 19.44402ms
https://[HOST2 IP]:2379 is healthy: successfully committed proposal: took = 35.926451ms
```

- Set `${ETCD_TAG}` to the version tag of your etcd image. For example `3.4.3-0`. To see the etcd image and tag that kubeadm uses execute `kubeadm config images list --kubernetes-version ${K8S_VERSION}`, where `${K8S_VERSION}` is for example `v1.17.0`
- Set `${HOST0}`to the IP address of the host you are testing.

What's next

Once you have a working 3 member etcd cluster, you can continue setting up a highly available control plane using the external etcd method with kubeadm.

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified November 16, 2020 at 12:39 PM PST: kubeadm: add instruction to check kubelet status (74243e939)
Edit this page Create child page Create an issue

Configuring each kubelet in your cluster using kubeadm

FEATURE STATE: `Kubernetes v1.11 [stable]`

The lifecycle of the kubeadm CLI tool is decoupled from the kubelet, which is a daemon that runs on each node within the Kubernetes cluster. The kubeadm CLI tool is executed by the user when Kubernetes is initialized or upgraded, whereas the kubelet is always running in the background.

Since the kubelet is a daemon, it needs to be maintained by some kind of an init system or service manager. When the kubelet is installed using DEBs or RPMs, systemd is configured to manage the kubelet. You can use a different service manager instead, but you need to configure it manually.

Some kubelet configuration details need to be the same across all kubelets involved in the cluster, while other configuration aspects need to be set on a per-kubelet basis to accommodate the different characteristics of a given machine (such as OS, storage, and networking). You can manage the configuration of your kubelets manually, but kubeadm now provides a `KubeletConfiguration` *API type for managing your kubelet configurations centrally.*

Kubelet configuration patterns

The following sections describe patterns to kubelet configuration that are simplified by using kubeadm, rather than managing the kubelet configuration for each Node manually.

Propagating cluster-level configuration to each kubelet

You can provide the kubelet with default values to be used by `kubeadm init` *and* `kubeadm join` *commands. Interesting examples include using a different CRI runtime or setting the default subnet used by services.*

If you want your services to use the subnet `10.96.0.0/12` *as the default for services, you can pass the* `--service-cidr` *parameter to kubeadm:*

```
kubeadm init --service-cidr 10.96.0.0/12
```

Virtual IPs for services are now allocated from this subnet. You also need to set the DNS address used by the kubelet, using the `--cluster-dns` *flag. This setting needs to be the same for every kubelet on every manager and Node in the cluster. The kubelet provides a versioned, structured API object that can configure most parameters in the kubelet and push out this configuration to each running kubelet in the cluster. This object is called* ***the kubelet's ComponentConfig****. The ComponentConfig allows the user to specify flags such as the cluster DNS IP addresses expressed as a list of values to a camelCased key, illustrated by the following example:*

```
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
clusterDNS:
- 10.96.0.10
```

For more details on the ComponentConfig have a look at this section.

Providing instance-specific configuration details

Some hosts require specific kubelet configurations due to differences in hardware, operating system, networking, or other host-specific parameters. The following list provides a few examples.

- The path to the DNS resolution file, as specified by the `--resolv-conf` kubelet configuration flag, may differ among operating systems, or depending on whether you are using `systemd-resolved`. If this path is wrong, DNS resolution will fail on the Node whose kubelet is configured incorrectly.
- The Node API object `.metadata.name` is set to the machine's hostname by default, unless you are using a cloud provider. You can use the `--hostname-override` flag to override the default behavior if you need to specify a Node name different from the machine's hostname.
- Currently, the kubelet cannot automatically detect the cgroup driver used by the CRI runtime, but the value of `--cgroup-driver` must match the cgroup driver used by the CRI runtime to ensure the health of the kubelet.
- Depending on the CRI runtime your cluster uses, you may need to specify different flags to the kubelet. For instance, when using Docker, you need to specify flags such as `--network-plugin=cni`, but if you are using an external runtime, you need to specify `--container-runtime=remote` and specify the CRI endpoint using the `--container-runtime-endpoint=<path>`.

You can specify these flags by configuring an individual kubelet's configuration in your service manager, such as systemd.

Configure kubelets using kubeadm

It is possible to configure the kubelet that kubeadm will start if a custom `KubeletConfiguration` API object is passed with a configuration file like so `kubeadm ... --config some-config-file.yaml`.

By calling `kubeadm config print init-defaults --component-configs KubeletConfiguration` you can see all the default values for this structure.

Also have a look at the API reference for the kubelet ComponentConfig for more information on the individual fields.

Workflow when using `kubeadm init`

When you call `kubeadm init`, the kubelet configuration is marshalled to disk at `/var/lib/kubelet/config.yaml`, and also uploaded to a ConfigMap in the cluster. The ConfigMap is named `kubelet-config-1.X`, where `X` is the minor version of the Kubernetes version you are initializing. A kubelet configuration file is also written to `/etc/kubernetes/kubelet.conf` with the baseline cluster-wide configuration for all kubelets in the cluster. This configuration file points to the client certificates that allow the kubelet to communicate with the API server. This addresses the need to propagate cluster-level configuration to each kubelet.

To address the second pattern of providing instance-specific configuration details, kubeadm writes an environment file to `/var/lib/kubelet/kubeadm-flags.env`, which contains a list of flags to pass to the kubelet when it starts. The flags are presented in the file like this:

```
KUBELET_KUBEADM_ARGS="--flag1=value1 --flag2=value2 ..."
```

In addition to the flags used when starting the kubelet, the file also contains dynamic parameters such as the cgroup driver and whether to use a different CRI runtime socket (`--cri-socket`).

After marshalling these two files to disk, kubeadm attempts to run the following two commands, if you are using systemd:

```
systemctl daemon-reload && systemctl restart kubelet
```

If the reload and restart are successful, the normal `kubeadm init` workflow continues.

Workflow when using `kubeadm join`

When you run `kubeadm join`, kubeadm uses the Bootstrap Token credential to perform a TLS bootstrap, which fetches the credential needed to download the `kubelet-config-1.X` ConfigMap and writes it to `/var/lib/kubelet/config.yaml`. The dynamic environment file is generated in exactly the same way as `kubeadm init`.

Next, `kubeadm` runs the following two commands to load the new configuration into the kubelet:

```
systemctl daemon-reload && systemctl restart kubelet
```

After the kubelet loads the new configuration, kubeadm writes the `/etc/kubernetes/bootstrap-kubelet.conf` KubeConfig file, which contains a CA certificate and Bootstrap Token. These are used by the kubelet to perform the TLS Bootstrap and obtain a unique credential, which is stored in `/etc/kubernetes/kubelet.conf`. When this file is written, the kubelet has finished performing the TLS Bootstrap.

The kubelet drop-in file for systemd

`kubeadm` ships with configuration for how systemd should run the kubelet. Note that the kubeadm CLI command never touches this drop-in file.

This configuration file installed by the `kubeadm` DEB or RPM package is written to `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` and is used by systemd. It augments the basic `kubelet.service` for RPM or `kubelet.service` for DEB:

```
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/
kubernetes/bootstrap-kubelet.conf
--kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/
config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generate
at runtime, populating
the KUBELET_KUBEADM_ARGS variable dynamically
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the
kubelet args as a last resort. Preferably,
# the user should use the .NodeRegistration.KubeletExtraArgs
object in the configuration files instead.
# KUBELET_EXTRA_ARGS should be sourced from this file.
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS
$KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_ARGS $KUBELET_EXTRA_ARGS
```

This file specifies the default locations for all of the files managed by kubeadm for the kubelet.

- The KubeConfig file to use for the TLS Bootstrap is `/etc/kubernetes/bootstrap-kubelet.conf`, but it is only used if `/etc/kubernetes/kubelet.conf` does not exist.
- The KubeConfig file with the unique kubelet identity is `/etc/kubernetes/kubelet.conf`.
- The file containing the kubelet's ComponentConfig is `/var/lib/kubelet/config.yaml`.
- The dynamic environment file that contains `KUBELET_KUBEADM_ARGS` is sourced from `/var/lib/kubelet/kubeadm-flags.env`.
- The file that can contain user-specified flag overrides with `KUBELET_EXTRA_ARGS` is sourced from `/etc/default/kubelet` (for DEBs), or `/etc/sysconfig/kubelet` (for RPMs). `KUBELET_EXTRA_ARGS` is last in the flag chain and has the highest priority in the event of conflicting settings.

Kubernetes binaries and package contents

The DEB and RPM packages shipped with the Kubernetes releases are:

Package name	Description
`kubeadm`	Installs the `/usr/bin/kubeadm` CLI tool and the kubelet drop-in file for the kubelet.
`kubelet`	Installs the kubelet binary in `/usr/bin` and CNI binaries in `/opt/cni/bin`.
`kubectl`	Installs the `/usr/bin/kubectl` binary.
`cri-tools`	Installs the `/usr/bin/crictl` binary from the cri-tools git repository.

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified October 17, 2020 at 3:21 PM PST: update kubernetes-incubator references (a8b6551c2)
Edit this page Create child page Create an issue

Configuring your kubernetes cluster to self-host the control plane

Self-hosting the Kubernetes control plane

kubeadm allows you to experimentally create a self-hosted Kubernetes control plane. This means that key components such as the API server, controller manager, and scheduler run as DaemonSet pods configured via the Kubernetes API instead of static pods configured in the kubelet via static files.

To create a self-hosted cluster see the kubeadm alpha selfhosting pivot command.

Caveats

> ***Caution:*** *This feature pivots your cluster into an unsupported state, rendering kubeadm unable to manage you cluster any longer. This includes* `kubeadm upgrade`*.*

1. *Self-hosting in 1.8 and later has some important limitations. In particular, a self-hosted cluster cannot recover from a reboot of the control-plane node without manual intervention.*

2. *By default, self-hosted control plane Pods rely on credentials loaded from* `hostPath` *volumes. Except for initial creation, these credentials are not managed by kubeadm.*

3. *The self-hosted portion of the control plane does not include etcd, which still runs as a static Pod.*

Process

The self-hosting bootstrap process is documented in the kubeadm design document.

In summary, `kubeadm alpha selfhosting` *works as follows:*

1. *Waits for this bootstrap static control plane to be running and healthy. This is identical to the* `kubeadm init` *process without self-hosting.*

2. *Uses the static control plane Pod manifests to construct a set of DaemonSet manifests that will run the self-hosted control plane. It also modifies these manifests where necessary, for example adding new volumes for secrets.*

3. *Creates DaemonSets in the* `kube-system` *namespace and waits for the resulting Pods to be running.*

4. *Once self-hosted Pods are operational, their associated static Pods are deleted and kubeadm moves on to install the next component. This triggers kubelet to stop those static Pods.*

5. *When the original static control plane stops, the new self-hosted control plane is able to bind to listening ports and become active.*

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Installing Kubernetes with kops

This quickstart shows you how to easily install a Kubernetes cluster on AWS. It uses a tool called `kops`.

kops is an automated provisioning system:

- Fully automated installation
- Uses DNS to identify clusters
- Self-healing: everything runs in Auto-Scaling Groups
- Multiple OS support (Debian, Ubuntu 16.04 supported, CentOS & RHEL, Amazon Linux and CoreOS) - see the images.md
- High-Availability support - see the high_availability.md
- Can directly provision, or generate terraform manifests - see the terraform.md

Before you begin

- You must have kubectl installed.
- You must install `kops` on a 64-bit (AMD64 and Intel 64) device architecture.
- You must have an AWS account, generate IAM keys and configure them. The IAM user will need adequate permissions.

Creating a cluster

(1/5) Install kops

Installation

Download kops from the releases page (it is also easy to build from source):

- macOS
- Linux

Download the latest release with the command:

```
curl -LO https://github.com/kubernetes/kops/releases/download/$(c
url -s https://api.github.com/repos/kubernetes/kops/releases/
latest | grep tag_name | cut -d '"' -f 4)/kops-darwin-amd64
```

To download a specific version, replace the following portion of the command with the specific kops version.

```
$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f 4)
```

For example, to download kops version v1.15.0 type:

```
curl -LO  https://github.com/kubernetes/kops/releases/download/1.15.0/kops-darwin-amd64
```

Make the kops binary executable.

```
chmod +x kops-darwin-amd64
```

Move the kops binary in to your PATH.

```
sudo mv kops-darwin-amd64 /usr/local/bin/kops
```

You can also install kops using Homebrew.

```
brew update && brew install kops
```

Download the latest release with the command:

```
curl -LO https://github.com/kubernetes/kops/releases/download/$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f 4)/kops-linux-amd64
```

To download a specific version of kops, replace the following portion of the command with the specific kops version.

```
$(curl -s https://api.github.com/repos/kubernetes/kops/releases/latest | grep tag_name | cut -d '"' -f 4)
```

For example, to download kops version v1.15.0 type:

```
curl -LO  https://github.com/kubernetes/kops/releases/download/1.15.0/kops-linux-amd64
```

Make the kops binary executable

```
chmod +x kops-linux-amd64
```

Move the kops binary in to your PATH.

```
sudo mv kops-linux-amd64 /usr/local/bin/kops
```

You can also install kops using Homebrew.

```
brew update && brew install kops
```

(2/5) Create a route53 domain for your cluster

kops uses DNS for discovery, both inside the cluster and outside, so that you can reach the kubernetes API server from clients.

kops has a strong opinion on the cluster name: it should be a valid DNS name. By doing so you will no longer get your clusters confused, you can share clusters with your colleagues unambiguously, and you can reach them without relying on remembering an IP address.

You can, and probably should, use subdomains to divide your clusters. As our example we will use `useast1.dev.example.com`. The API server endpoint will then be `api.useast1.dev.example.com`.

A Route53 hosted zone can serve subdomains. Your hosted zone could be `useast1.dev.example.com`, but also `dev.example.com` or even `example.com`. kops works with any of these, so typically you choose for organization reasons (e.g. you are allowed to create records under `dev.example.com`, but not under `example.com`).

Let's assume you're using `dev.example.com` as your hosted zone. You create that hosted zone using the [normal process](), or with a command such as `aws route53 create-hosted-zone --name dev.example.com --caller-reference 1`.

You must then set up your NS records in the parent domain, so that records in the domain will resolve. Here, you would create NS records in `example.com` for `dev`. If it is a root domain name you would configure the NS records at your domain registrar (e.g. `example.com` would need to be configured where you bought `example.com`).

This step is easy to mess up (it is the #1 cause of problems!) You can double-check that your cluster is configured correctly if you have the dig tool by running:

```
dig NS dev.example.com
```

You should see the 4 NS records that Route53 assigned your hosted zone.

(3/5) Create an S3 bucket to store your clusters state

kops lets you manage your clusters even after installation. To do this, it must keep track of the clusters that you have created, along with their configuration, the keys they are using etc. This information is stored in an S3 bucket. S3 permissions are used to control access to the bucket.

Multiple clusters can use the same S3 bucket, and you can share an S3 bucket between your colleagues that administer the same clusters - this is much easier than passing around kubecfg files. But anyone with access to the S3 bucket will have administrative access to all your clusters, so you don't want to share it beyond the operations team.

So typically you have one S3 bucket for each ops team (and often the name will correspond to the name of the hosted zone above!)

In our example, we chose `dev.example.com` as our hosted zone, so let's pick `clusters.dev.example.com` as the S3 bucket name.

- *Export `AWS_PROFILE` (if you need to select a profile for the AWS CLI to work)*
- *Create the S3 bucket using `aws s3 mb s3://clusters.dev.example.com`*
- *You can `export KOPS_STATE_STORE=s3://clusters.dev.example.com` and then kops will use this location by default. We suggest putting this in your bash profile or similar.*

(4/5) Build your cluster configuration

Run `kops create cluster` to create your cluster configuration:

```
kops create cluster --zones=us-east-1c useast1.dev.example.com
```

kops will create the configuration for your cluster. Note that it only creates the configuration, it does not actually create the cloud resources - you'll do that in the next step with a `kops update cluster`. This give you an opportunity to review the configuration or change it.

It prints commands you can use to explore further:

- *List your clusters with: `kops get cluster`*
- *Edit this cluster with: `kops edit cluster useast1.dev.example.com`*
- *Edit your node instance group: `kops edit ig --name=useast1.dev.example.com nodes`*
- *Edit your master instance group: `kops edit ig --name=useast1.dev.example.com master-us-east-1c`*

If this is your first time using kops, do spend a few minutes to try those out! An instance group is a set of instances, which will be registered as kubernetes nodes. On AWS this is implemented via auto-scaling-groups. You can have several instance groups, for example if you wanted nodes that are a mix of spot and on-demand instances, or GPU and non-GPU instances.

(5/5) Create the cluster in AWS

Run "kops update cluster" to create your cluster in AWS:

```
kops update cluster useast1.dev.example.com --yes
```

That takes a few seconds to run, but then your cluster will likely take a few minutes to actually be ready. `kops update cluster` will be the tool you'll use whenever you change the configuration of your cluster; it applies the changes you have made to the configuration to your cluster - reconfiguring AWS or kubernetes as needed.

For example, after you `kops edit ig nodes`, then `kops update cluster --yes` to apply your configuration, and sometimes you will also have to `kops rolling-update cluster` to roll out the configuration immediately.

Without `--yes`, `kops update cluster` will show you a preview of what it is going to do. This is handy for production clusters!

Explore other add-ons

See the list of add-ons to explore other add-ons, including tools for logging, monitoring, network policy, visualization, and control of your Kubernetes cluster.

Cleanup

- *To delete your cluster: `kops delete cluster useast1.dev.example.com --yes`*

What's next

- *Learn more about Kubernetes concepts and `kubectl`.*
- *Learn more about `kops` advanced usage for tutorials, best practices and advanced configuration options.*
- *Follow `kops` community discussions on Slack: community discussions*
- *Contribute to `kops` by addressing or raising an issue GitHub Issues*

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified July 20, 2020 at 4:17 PM PST: Replace redirected links with the real targets (c4add100f)
Edit this page Create child page Create an issue

Installing Kubernetes with Kubespray

This quickstart helps to install a Kubernetes cluster hosted on GCE, Azure, OpenStack, AWS, vSphere, Packet (bare metal), Oracle Cloud Infrastructure (Experimental) or Baremetal with Kubespray.

Kubespray is a composition of Ansible playbooks, inventory, provisioning tools, and domain knowledge for generic OS/Kubernetes clusters configuration management tasks. Kubespray provides:

- *a highly available cluster*
- *composable attributes*
- *support for most popular Linux distributions*
  - *Ubuntu 16.04, 18.04, 20.04*
  - *CentOS/RHEL/Oracle Linux 7, 8*
  - *Debian Buster, Jessie, Stretch, Wheezy*
  - *Fedora 31, 32*
  - *Fedora CoreOS*
  - *openSUSE Leap 15*
  - *Flatcar Container Linux by Kinvolk*
- *continuous integration tests*

To choose a tool which best fits your use case, read this comparison to kubeadm and kops.

Creating a cluster

(1/5) Meet the underlay requirements

Provision servers with the following requirements:

- ***Ansible v2.9 and python-netaddr is installed on the machine that will run Ansible commands***
- ***Jinja 2.11 (or newer) is required to run the Ansible Playbooks***
- *The target servers must have access to the Internet in order to pull docker images. Otherwise, additional configuration is required (See Offline Environment)*
- *The target servers are configured to allow **IPv4 forwarding***
- ***Your ssh key must be copied** to all the servers part of your inventory*
- *The **firewalls are not managed**, you'll need to implement your own rules the way you used to. in order to avoid any issue during deployment you should disable your firewall*
- *If kubespray is ran from non-root user account, correct privilege escalation method should be configured in the target servers. Then the `ansible_become` flag or command parameters `--become` or `-b` should be specified*

Kubespray provides the following utilities to help provision your environment:

- *Terraform scripts for the following cloud providers:*
  - *AWS*
  - *OpenStack*
  - *Packet*

(2/5) Compose an inventory file

After you provision your servers, create an inventory file for Ansible. You can do this manually or via a dynamic inventory script. For more information, see "Building your own inventory".

(3/5) Plan your cluster deployment

Kubespray provides the ability to customize many aspects of the deployment:

- *Choice deployment mode: kubeadm or non-kubeadm*
- *CNI (networking) plugins*
- *DNS configuration*
- *Choice of control plane: native/binary or containerized*
- *Component versions*
- *Calico route reflectors*
- *Component runtime options*
  - *Docker*
  - *containerd*
  - *CRI-O*
- *Certificate generation methods*

Kubespray customizations can be made to a variable file. If you are just getting started with Kubespray, consider using the Kubespray defaults to deploy your cluster and explore Kubernetes.

(4/5) Deploy a Cluster

Next, deploy your cluster:

Cluster deployment using ansible-playbook.

```
ansible-playbook -i your/inventory/inventory.ini cluster.yml -b -v \
  --private-key=~/.ssh/private_key
```

Large deployments (100+ nodes) may require specific adjustments for best results.

(5/5) Verify the deployment

Kubespray provides a way to verify inter-pod connectivity and DNS resolve with Netchecker. Netchecker ensures the netchecker-agents pods can

resolve DNS requests and ping each over within the default namespace. Those pods mimic similar behavior of the rest of the workloads and serve as cluster health indicators.

Cluster operations

Kubespray provides additional playbooks to manage your cluster: scale and upgrade.

Scale your cluster

You can add worker nodes from your cluster by running the scale playbook. For more information, see "Adding nodes". You can remove worker nodes from your cluster by running the remove-node playbook. For more information, see "Remove nodes".

Upgrade your cluster

You can upgrade your cluster by running the upgrade-cluster playbook. For more information, see "Upgrades".

Cleanup

You can reset your nodes and wipe out all components installed with Kubespray via the reset playbook.

> ***Caution:*** *When running the reset playbook, be sure not to accidentally target your production cluster!*

Feedback

- *Slack Channel: #kubespray (You can get your invite here)*
- *GitHub Issues*

What's next

Check out planned work on Kubespray's roadmap.

Feedback

Was this page helpful?

Yes No

Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Last modified September 10, 2020 at 9:47 AM PST: Update Kubespray page (a5cbc1f31)
Edit this page Create child page Create an issue

Turnkey Cloud Solutions

This page provides a list of Kubernetes certified solution providers. From each provider page, you can learn how to install and setup production ready clusters.

<script src="https://landscape.cncf.io/iframeResizer.js"/> <h2>Feedback</h2> Was this page helpful? <button class="btn btn-primary mb-4 feedback--yes">Yes</button> <button class="btn btn-primary mb-4 feedback--no">No</button> Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement. <script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=>{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=>{if(!gtag){console.log('!gtag');}gtag('event','click',{'event_category':'Helpful','event_label':window.location.pathname,value});};const disableButtons=()=>{yes.disabled=true;yes.classList.add('feedback--button__disabled');no.disabled=true;no.classList.add('feedback--button__disabled');};yes.addEventListener('click',()=>{sendFeedback(1);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});no.addEventListener('click',()=>

{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});</script> Last modified November 03, 2020 at 4:24 PM PST: modify width of iframe (a8f07b6a1) <i class="fa fa-edit fa-fw"/>Edit this page <i class="fa fa-edit fa-fw"/>Create child page <i class="fab fa-github fa-fw"/>Create an issue <h1>Windows in Kubernetes</h1> <hr class="panel-line"/> <h5>Intro to Windows support in Kubernetes</h5> <h5>Guide for scheduling Windows containers in Kubernetes</h5> <h1>Intro to Windows support in Kubernetes</h1> Windows applications constitute a large portion of the services and applications that run in many organizations. Windows containers provide a modern way to encapsulate processes and package dependencies, making it easier to use DevOps practices and follow cloud native patterns for Windows applications. Kubernetes has become the defacto standard container orchestrator, and the release of Kubernetes 1.14 includes production support for scheduling Windows containers on Windows nodes in a Kubernetes cluster, enabling a vast ecosystem of Windows applications to leverage the power of Kubernetes. Organizations with investments in Windows-based applications and Linux-based applications don't have to look for separate orchestrators to manage their workloads, leading to increased operational efficiencies across their deployments, regardless of operating system. <h2 id="windows-containers-in-kubernetes">Windows containers in Kubernetes</h2> To enable the orchestration of Windows containers in Kubernetes, simply include Windows nodes in your existing Linux cluster. Scheduling Windows containers in <a class="glossary-tooltip" title="A Pod represents a set of running containers in your cluster." data-toggle="tooltip" data-placement="top" href="/docs/concepts/workloads/

pods/" target="_blank" aria-label="Pods">Pods on Kubernetes is as simple and easy as scheduling Linux-based containers. In order to run Windows containers, your Kubernetes cluster must include multiple operating systems, with control plane nodes running Linux and workers running either Windows or Linux depending on your workload needs. Windows Server 2019 is the only Windows operating system supported, enabling Kubernetes Node on Windows (including kubelet, container runtime, and kube-proxy). For a detailed explanation of Windows distribution channels see the Microsoft documentation. <blockquote class="note callout"> Note: The Kubernetes control plane, including the master components, continues to run on Linux. There are no plans to have a Windows-only Kubernetes cluster. </blockquote><blockquote class="note callout"> Note: In this document, when we talk about Windows containers we mean Windows containers with process isolation. Windows containers with Hyper-V isolation is planned for a future release. </blockquote><h2 id="supported-functionality-and-limitations">Supported Functionality and Limitations</h2><h3 id="supported-functionality">Supported Functionality</h3><h4 id="windows-os-version-support">Windows OS Version Support</h4> Refer to the following table for Windows operating system support in Kubernetes. A single heterogeneous Kubernetes cluster can have both Windows and Linux worker nodes. Windows containers have to be scheduled on Windows nodes and Linux containers on Linux nodes. <table><thead><tr><th>Kubernetes version</th><th>Windows Server LTSC releases</th><th>Windows Server SAC releases</th><th></th></tr></thead><tbody><tr><td>Kubernetes v1.17</td><td>Windows Server 2019</td><td>Windows Server ver 1809</td><td></td></tr><tr><td>Kubernetes v1.18</td><td>Windows Server 2019</td><td>Windows Server ver 1809, Windows Server ver 1903, Windows Server ver 1909</td><td></td></tr><tr><td>Kubernetes v1.19</td><td>Windows Server 2019</td><td>Windows Server ver 1909, Windows Server ver 2004</td><td></td></tr><tr><td>Kubernetes v1.20</td><td>Windows Server 2019</td><td>Windows Server ver 1909, Windows Server ver 2004</td><td></td></tr></tbody></table> <blockquote class="note callout"> Note: Information on the different Windows Server servicing channels including their support models can be found at Windows Server servicing channels. </blockquote><blockquote class="note callout"> Note: We don't expect all Windows customers to update the operating system for their apps frequently. Upgrading your applications is what dictates and necessitates upgrading or introducing new nodes to the cluster. For the customers that chose to

upgrade their operating system for containers running on Kubernetes, we will offer guidance and step-by-step instructions when we add support for a new operating system version. This guidance will include recommended upgrade procedures for upgrading user applications together with cluster nodes. Windows nodes adhere to Kubernetes version-skew policy (node to control plane versioning) the same way as Linux nodes do today. </blockquote><blockquote class="note callout"> Note: The Windows Server Host Operating System is subject to the Windows Server licensing. The Windows Container images are subject to the Supplemental License Terms for Windows containers. </blockquote><blockquote class="note callout"> Note: Windows containers with process isolation have strict compatibility rules, where the host OS version must match the container base image OS version. Once we support Windows containers with Hyper-V isolation in Kubernetes, the limitation and compatibility rules will change. </blockquote><h4 id="pause-image">Pause Image</h4> Microsoft maintains a Windows pause infrastructure container at <code>mcr.microsoft.com/oss/kubernetes/pause:1.4.1</code>. <h4 id="compute">Compute</h4> From an API and kubectl perspective, Windows containers behave in much the same way as Linux-based containers. However, there are some notable differences in key functionality which are outlined in the limitation section. Key Kubernetes elements work the same way in Windows as they do in Linux. In this section, we talk about some of the key workload enablers and how they map to Windows. Pods A Pod is the basic building block of Kubernetes-the smallest and simplest unit in the Kubernetes object model that you create or deploy. You may not deploy Windows and Linux containers in the same Pod. All containers in a Pod are scheduled onto a single Node where each Node represents a specific platform and architecture. The following Pod capabilities, properties and events are supported with Windows containers: Single or multiple containers per Pod with process isolation and volume sharing
Pod status fields
Readiness and Liveness probes
postStart & preStop container lifecycle events
ConfigMap, Secrets: as environment variables or volumes
EmptyDir
Named pipe host mounts
Resource limits Controllers Kubernetes controllers handle the desired state of Pods. The following workload controllers are supported with Windows containers: ReplicaSet
ReplicationController
Deployments
StatefulSets
DaemonSet
Job
CronJob Services A Kubernetes Service is an abstraction which defines a logical set of Pods and a policy by which to access them - sometimes called a micro-service. You can use services for cross-operating

system connectivity. In Windows, services can utilize the following types, properties and capabilities: Service Environment variables
NodePort
ClusterIP
LoadBalancer
ExternalName
Headless services Pods, Controllers and Services are critical elements to managing Windows workloads on Kubernetes. However, on their own they are not enough to enable the proper lifecycle management of Windows workloads in a dynamic cloud native environment. We added support for the following features: Pod and container metrics
Horizontal Pod Autoscaler support
kubectl Exec
Resource Quotas
Scheduler preemption
<h4 id="container-runtime">Container Runtime</h4><h5 id="docker-ee">Docker EE</h5> FEATURE STATE: <code>Kubernetes v1.14 [stable]</code> Docker EE-basic 19.03+ is the recommended container runtime for all Windows Server versions. This works with the dockershim code included in the kubelet. <h5 id="cri-containerd">CRI-ContainerD</h5> FEATURE STATE: <code>Kubernetes v1.20 [stable]</code> ContainerD 1.4.0+ can also be used as the container runtime for Windows Kubernetes nodes. Learn how to install ContainerD on a Windows. <blockquote class="caution callout"> Caution: There is a known limitation when using GMSA with ContainerD to access Windows network shares which requires a kernel patch. Updates to address this limitation are currently available for Windows Server, Version 2004 and will be available for Windows Server 2019 in early 2021. Check for updates on the Microsoft Windows Containers issue tracker. </blockquote><h4 id="persistent-storage">Persistent Storage</h4> Kubernetes volumes enable complex applications, with data persistence and Pod volume sharing requirements, to be deployed on Kubernetes. Management of persistent volumes associated with a specific storage back-end or protocol includes actions such as: provisioning/de-provisioning/resizing of volumes, attaching/detaching a volume to/from a Kubernetes node and mounting/dismounting a volume to/from individual containers in a pod that needs to persist data. The code implementing these volume management actions for a specific storage back-end or protocol is shipped in the form of a Kubernetes volume plugin. The following broad classes of Kubernetes volume plugins are supported on Windows: <h5 id="in-tree-volume-plugins">In-tree Volume Plugins</h5> Code associated with in-tree volume plugins ship as part of the core Kubernetes code base. Deployment of in-tree volume plugins do not require installation of additional scripts or deployment of separate containerized plugin components. These plugins can handle: provisioning/de-provisioning and resizing of volumes in the storage

backend, attaching/detaching of volumes to/from a Kubernetes node and mounting/dismounting a volume to/from individual containers in a pod. The following in-tree plugins support Windows nodes: awsElasticBlockStore
azureDisk
azureFile
gcePersistentDisk
vsphereVolume
<h5 id="flexvolume-plugins">FlexVolume Plugins</h5> Code associated with FlexVolume plugins ship as out-of-tree scripts or binaries that need to be deployed directly on the host. FlexVolume plugins handle attaching/detaching of volumes to/from a Kubernetes node and mounting/dismounting a volume to/from individual containers in a pod. Provisioning/De-provisioning of persistent volumes associated with FlexVolume plugins may be handled through an external provisioner that is typically separate from the FlexVolume plugins. The following FlexVolume plugins, deployed as powershell scripts on the host, support Windows nodes: SMB
iSCSI
<h5 id="csi-plugins">CSI Plugins</h5> FEATURE STATE: <code>Kubernetes v1.19 [beta]</code> Code associated with CSI plugins ship as out-of-tree scripts and binaries that are typically distributed as container images and deployed using standard Kubernetes constructs like DaemonSets and StatefulSets. CSI plugins handle a wide range of volume management actions in Kubernetes: provisioning/de-provisioning/resizing of volumes, attaching/detaching of volumes to/from a Kubernetes node and mounting/dismounting a volume to/from individual containers in a pod, backup/restore of persistent data using snapshots and cloning. CSI plugins typically consist of node plugins (that run on each node as a DaemonSet) and controller plugins. CSI node plugins (especially those associated with persistent volumes exposed as either block devices or over a shared file-system) need to perform various privileged operations like scanning of disk devices, mounting of file systems, etc. These operations differ for each host operating system. For Linux worker nodes, containerized CSI node plugins are typically deployed as privileged containers. For Windows worker nodes, privileged operations for containerized CSI node plugins is supported using csi-proxy, a community-managed, stand-alone binary that needs to be pre-installed on each Windows node. Please refer to the deployment guide of the CSI plugin you wish to deploy for further details. <h4

id="networking">Networking</h4> Networking for Windows containers is exposed through CNI plugins. Windows containers function similarly to virtual machines in regards to networking. Each container has a virtual network adapter (vNIC) which is connected to a Hyper-V virtual switch (vSwitch). The Host Networking Service (HNS) and the Host Compute Service (HCS) work together to create containers and attach container vNICs to networks. HCS is responsible for the management of containers whereas HNS is responsible for the management of networking resources such as: Virtual networks (including creation of vSwitches)
Endpoints / vNICs
Namespaces
Policies (Packet encapsulations, Load-balancing rules, ACLs, NAT'ing rules, etc.) The following service spec types are supported: NodePort
ClusterIP
LoadBalancer
ExternalName
<h5 id="network-modes">Network modes</h5> Windows supports five different networking drivers/modes: L2bridge, L2tunnel, Overlay, Transparent, and NAT. In a heterogeneous cluster with Windows and Linux worker nodes, you need to select a networking solution that is compatible on both Windows and Linux. The following out-of-tree plugins are supported on Windows, with recommendations on when to use each CNI: <table><thead><tr><th>Network Driver</th><th>Description</th><th>Container Packet Modifications</th><th>Network Plugins</th><th>Network Plugin Characteristics</th></tr></thead><tbody><tr><td>L2bridge</td><td>Containers are attached to an external vSwitch. Containers are attached to the underlay network, although the physical network doesn't need to learn the container MACs because they are rewritten on ingress/egress.</td><td>MAC is rewritten to host MAC, IP may be rewritten to host IP using HNS OutboundNAT policy.</td><td>win-bridge, Azure-CNI, Flannel host-gateway uses win-bridge</td><td>win-bridge uses L2bridge network mode, connects containers to the underlay of hosts, offering best performance. Requires user-defined routes (UDR) for inter-node connectivity.</td></tr><tr><td>L2Tunnel</td><td>This is a special case of l2bridge, but only used on Azure. All packets are sent to the virtualization host where SDN policy is applied.</td><td>MAC rewritten, IP visible on the underlay network</td><td>Azure-CNI</td><td>Azure-CNI allows integration of containers with Azure vNET, and allows them to leverage the set of capabilities that Azure Virtual Network provides. For example, securely connect to Azure services or use Azure NSGs. See azure-cni for some examples</td></tr><tr><td>Overlay (Overlay networking for Windows in Kubernetes is in alpha stage)</td><td>Containers are given a vNIC connected to an external vSwitch. Each overlay network gets its own IP subnet, defined by a custom IP prefix.The overlay network driver uses VXLAN encapsulation.</td><td>Encapsulated with an outer header.</td><td><a href="https://github.com/containernetworking/plugins/tree/

master/plugins/main/windows/win-overlay">Win-overlay, Flannel VXLAN (uses win-overlay)</td><td>win-overlay should be used when virtual container networks are desired to be isolated from underlay of hosts (e.g. for security reasons). Allows for IPs to be re-used for different overlay networks (which have different VNID tags) if you are restricted on IPs in your datacenter. This option requires KB4489899 on Windows Server 2019.</td></tr><tr><td>Transparent (special use case for ovn-kubernetes)</td><td>Requires an external vSwitch. Containers are attached to an external vSwitch which enables intra-pod communication via logical networks (logical switches and routers).</td><td>Packet is encapsulated either via GENEVE or STT tunneling to reach pods which are not on the same host.
Packets are forwarded or dropped via the tunnel metadata information supplied by the ovn network controller.
NAT is done for north-south communication.</td><td>ovn-kubernetes</td><td>Deploy via ansible. Distributed ACLs can be applied via Kubernetes policies. IPAM support. Load-balancing can be achieved without kube-proxy. NATing is done without using iptables/netsh.</td></tr><tr><td>NAT (not used in Kubernetes)</td><td>Containers are given a vNIC connected to an internal vSwitch. DNS/DHCP is provided using an internal component called WinNAT</td><td>MAC and IP is rewritten to host MAC/IP.</td><td>nat</td><td>Included here for completeness</td></tr></tbody></table> As outlined above, the Flannel CNI meta plugin is also supported on Windows via the VXLAN network backend (alpha support ; delegates to win-overlay) and host-gateway network backend (stable support; delegates to win-bridge). This plugin supports delegating to one of the reference CNI plugins (win-overlay, win-bridge), to work in conjunction with Flannel daemon on Windows (Flanneld) for automatic node subnet lease assignment and HNS network creation. This plugin reads in its own configuration file (cni.conf), and aggregates it with the environment variables from the FlannelD generated subnet.env file. It then delegates to one of the reference CNI plugins for network plumbing, and sends the correct configuration containing the node-assigned subnet to the IPAM plugin (e.g. host-local). For the node, pod, and service objects, the following network flows are supported for TCP/UDP traffic: Pod -> Pod (IP)
Pod -> Pod (Name)
Pod -

> Service (Cluster IP)
Pod -> Service (PQDN, but only if there are no ".")
Pod -> Service (FQDN)
Pod -> External (IP)
Pod -> External (DNS)
Node -> Pod
Pod -> Node
<h5 id="ipam">IP address management (IPAM)</h5> The following IPAM options are supported on Windows: Host-local
HNS IPAM (Inbox platform IPAM, this is a fallback when no IPAM is set)
Azure-vnet-ipam (for azure-cni only)
<h5 id="load-balancing-and-services">Load balancing and Services</h5> On Windows, you can use the following settings to configure Services and load balancing behavior: <table><caption style="display:none">Windows Service Settings</caption><thead><tr><th>Feature</th><th>Description</th><th>Supported Kubernetes version</th><th>Supported Windows OS build</th><th>How to enable</th></tr></thead><tbody><tr><td>Session affinity</td><td>Ensures that connections from a particular client are passed to the same Pod each time.</td><td>v1.19+</td><td>Windows Server vNext Insider Preview Build 19551 (or higher)</td><td>Set <code>service.spec.sessionAffinity</code> to "ClientIP"</td></tr><tr><td>Direct Server Return</td><td>Load balancing mode where the IP address fixups and the LBNAT occurs at the container vSwitch port directly; service traffic arrives with the source IP set as the originating pod IP. Promises lower latency and scalability.</td><td>v1.15+</td><td>Windows Server, version 2004</td><td>Set the following flags in kube-proxy: <code>--feature-gates="WinDSR=true" --enable-dsr=true</code></td></tr><tr><td>Preserve-Destination</td><td>Skips DNAT of service traffic, thereby preserving the virtual IP of the target service in packets reaching the backend Pod. This setting will also ensure that the client IP of incoming packets get preserved.</td><td>v1.15+</td><td>Windows Server, version 1903 (or higher)</td><td>Set <code>"preserve-destination": "true"</code> in service annotations and enable DSR flags in kube-proxy.</td></tr><tr><td>IPv4/IPv6 dual-stack networking</td><td>Native IPv4-to-IPv4 in parallel with IPv6-to-IPv6 communications to, from, and within a cluster</td><td>v1.19+</td><td>Windows Server vNext Insider Preview Build 19603 (or higher)</td><td>See IPv4/IPv6 dual-stack</td></tr></tbody></table><h4 id="ipv4-ipv6-dual-stack">IPv4/IPv6 dual-stack</h4> You can enable IPv4/IPv6 dual-stack networking for <code>l2bridge</code> networks using the <code>IPv6DualStack</code> feature gate. See enable IPv4/IPv6 dual stack for more details. <blockquote class="note callout"> Note: On Windows, using IPv6 with Kubernetes require Windows Server, version 2004 (kernel version 10.0.19041.610) or later. </blockquote><blockquote class="note callout"> Note: Overlay (VXLAN) networks on Windows do not support dual-stack networking today. </

blockquote><h3 id="limitations">Limitations</h3><h4 id="control-plane">Control Plane</h4> Windows is only supported as a worker node in the Kubernetes architecture and component matrix. This means that a Kubernetes cluster must always include Linux master nodes, zero or more Linux worker nodes, and zero or more Windows worker nodes. <h4 id="compute-compute-limitations">Compute {compute-limitations}</h4><h5 id="resource-management-and-process-isolation">Resource management and process isolation</h5> Linux cgroups are used as a pod boundary for resource controls in Linux. Containers are created within that boundary for network, process and file system isolation. The cgroups APIs can be used to gather cpu/io/memory stats. In contrast, Windows uses a Job object per container with a system namespace filter to contain all processes in a container and provide logical isolation from the host. There is no way to run a Windows container without the namespace filtering in place. This means that system privileges cannot be asserted in the context of the host, and thus privileged containers are not available on Windows. Containers cannot assume an identity from the host because the Security Account Manager (SAM) is separate. <h5 id="operating-system-restrictions">Operating System Restrictions</h5> Windows has strict compatibility rules, where the host OS version must match the container base image OS version. Only Windows containers with a container operating system of Windows Server 2019 are supported. Hyper-V isolation of containers, enabling some backward compatibility of Windows container image versions, is planned for a future release. <h5 id="feature-restrictions">Feature Restrictions</h5>
TerminationGracePeriod: requires CRI-containerD
Single file mapping: to be implemented with CRI-ContainerD
Termination message: to be implemented with CRI-ContainerD
Privileged Containers: not currently supported in Windows containers
HugePages: not currently supported in Windows containers
The existing node problem detector is Linux-only and requires privileged containers. In general, we don't expect this to be used on Windows because privileged containers are not supported
Not all features of shared namespaces are supported (see API section for more details)
<h5 id="memory-reservations-and-handling">Memory Reservations and Handling</h5> Windows does not have an out-of-memory process killer as Linux does. Windows always treats all user-mode memory allocations as virtual, and pagefiles are mandatory. The net effect is that Windows won't reach out of memory conditions the same way Linux does, and processes page to disk instead of being subject to out of memory (OOM) termination. If memory is over-provisioned and all physical memory is exhausted, then paging can slow down performance. Keeping memory usage within reasonable bounds is possible with a two-step process. First, use the kubelet parameters <code>--kubelet-reserve</code> and/or <code>--system-reserve</code> to account for memory usage on the node (outside of containers). This reduces NodeAllocatable). As you deploy workloads, use resource limits (must set only limits or limits must equal requests) on containers. This also subtracts from NodeAllocatable and prevents the scheduler from adding more pods once a node is full. A best practice to avoid over-provisioning is to configure the kubelet with a system reserved memory of at least 2GB to account for Windows, Docker, and Kubernetes processes.</

p> The behavior of the flags behave differently as described below: <code>--kubelet-reserve</code>, <code>--system-reserve</code> , and <code>--eviction-hard</code> flags update Node Allocatable
Eviction by using <code>--enforce-node-allocable</code> is not implemented
Eviction by using <code>--eviction-hard</code> and <code>--eviction-soft</code> are not implemented
MemoryPressure Condition is not implemented
There are no OOM eviction actions taken by the kubelet
Kubelet running on the windows node does not have memory restrictions. <code>--kubelet-reserve</code> and <code>--system-reserve</code> do not set limits on kubelet or processes running on the host. This means kubelet or a process on the host could cause memory resource starvation outside the node-allocatable and scheduler
An additional flag to set the priority of the kubelet process is availabe on the Windows nodes called <code>--windows-priorityclass</code>. This flag allows kubelet process to get more CPU time slices when compared to other processes running on the Windows host. More information on the allowable values and their meaning is available at Windows Priority Classes. In order for kubelet to always have enough CPU cycles it is recommended to set this flag to <code>ABOVE_NORMAL_PRIORITY_CLASS</code> and above
<h4 id="storage">Storage</h4> Windows has a layered filesystem driver to mount container layers and create a copy filesystem based on NTFS. All file paths in the container are resolved only within the context of that container. With Docker Volume mounts can only target a directory in the container, and not an individual file. This limitation does not exist with CRI-containerD.
Volume mounts cannot project files or directories back to the host filesystem
Read-only filesystems are not supported because write access is always required for the Windows registry and SAM database. However, read-only volumes are supported
Volume user-masks and permissions are not available. Because the SAM is not shared between the host & container, there's no mapping between them. All permissions are resolved within the context of the container As a result, the following storage functionality is not supported on Windows nodes Volume subpath mounts. Only the entire volume can be mounted in a Windows container.
Subpath volume mounting for Secrets
Host mount projection
DefaultMode (due to UID/GID dependency)
Read-only root filesystem. Mapped volumes still support readOnly
Block device mapping
Memory as the storage medium
File system features like uui/guid, per-user Linux filesystem permissions
NFS based storage/volume support
Expanding the mounted volume (resizefs)
<h4 id="networking-networking-limitations">Networking {networking-limitations}</h4> Windows Container Networking differs in some important ways from Linux networking. The Microsoft documentation for Windows Container Networking contains additional details and background. The Windows host networking service and virtual switch implement namespacing and can create virtual NICs as needed for a pod or container.

However, many configurations such as DNS, routes, and metrics are stored in the Windows registry database rather than /etc/... files as they are on Linux. The Windows registry for the container is separate from that of the host, so concepts like mapping /etc/resolv.conf from the host into a container don't have the same effect they would on Linux. These must be configured using Windows APIs run in the context of that container. Therefore CNI implementations need to call the HNS instead of relying on file mappings to pass network details into the pod or container. The following networking functionality is not supported on Windows nodes Host networking mode is not available for Windows pods
Local NodePort access from the node itself fails (works for other nodes or external clients)
Accessing service VIPs from nodes will be available with a future release of Windows Server
Overlay networking support in kube-proxy is an alpha release. In addition, it requires KB4482887 to be installed on Windows Server 2019
Local Traffic Policy and DSR mode
Windows containers connected to l2bridge, l2tunnel, or overlay networks do not support communicating over the IPv6 stack. There is outstanding Windows platform work required to enable these network drivers to consume IPv6 addresses and subsequent Kubernetes work in kubelet, kube-proxy, and CNI plugins.
Outbound communication using the ICMP protocol via the win-overlay, win-bridge, and Azure-CNI plugin. Specifically, the Windows data plane (VFP) doesn't support ICMP packet transpositions. This means:
ICMP packets directed to destinations within the same network (e.g. pod to pod communication via ping) work as expected and without any limitations
TCP/UDP packets work as expected and without any limitations
ICMP packets directed to pass through a remote network (e.g. pod to external internet communication via ping) cannot be transposed and thus will not be routed back to their source
Since TCP/UDP packets can still be transposed, one can substitute <code>ping <destination></code> with <code>curl <destination></code> to be able to debug connectivity to the outside world. These features were added in Kubernetes v1.15: <code>kubectl port-forward</code>
<h5 id="cni-plugins">CNI Plugins</h5>
Windows reference network plugins win-bridge and win-overlay do not currently implement CNI spec v0.4.0 due to missing "CHECK" implementation.
The Flannel VXLAN CNI has the following limitations on Windows:
Node-pod connectivity isn't possible by design. It's only possible for local pods with Flannel v0.12.0 (or higher).
We are restricted to using VNI 4096 and UDP port 4789. The VNI limitation is being worked on and will be overcome in a future release (open-source flannel changes). See the official Flannel VXLAN backend docs for more details on these parameters.
<h5 id="dns-limitations">DNS</h5>
ClusterFirstWithHostNet is not supported for DNS. Windows treats all names with a '.' as a FQDN and skips PQDN resolution
On Linux, you have a DNS suffix list, which is used when trying to resolve

PQDNs. On Windows, we only have 1 DNS suffix, which is the DNS suffix associated with that pod's namespace (mydns.svc.cluster.local for example). Windows can resolve FQDNs and services or names resolvable with just that suffix. For example, a pod spawned in the default namespace, will have the DNS suffix default.svc.cluster.local. On a Windows pod, you can resolve both kubernetes.default.svc.cluster.local and kubernetes, but not the in-betweens, like kubernetes.default or kubernetes.default.svc.
On Windows, there are multiple DNS resolvers that can be used. As these come with slightly different behaviors, using the <code>Resolve-DNSName</code> utility for name query resolutions is recommended.
<h5 id="ipv6">IPv6</h5> Kubernetes on Windows does not support single-stack "IPv6-only" networking. However,dual-stack IPv4/IPv6 networking for pods and nodes with single-family services is supported. See IPv4/IPv6 dual-stack networking for more details. <h5 id="session-affinity">Session affinity</h5> Setting the maximum session sticky time for Windows services using <code>service.spec.sessionAffinityConfig.clientIP.timeoutSeconds</code> is not supported. <h5 id="security">Security</h5> Secrets are written in clear text on the node's volume (as compared to tmpfs/in-memory on linux). This means customers have to do two things Use file ACLs to secure the secrets file location
Use volume-level encryption using BitLocker RunAsUsername can be specified for Windows Pod's or Container's to execute the Container processes as a node-default user. This is roughly equivalent to RunAsUser. Linux specific pod security context privileges such as SELinux, AppArmor, Seccomp, Capabilities (POSIX Capabilities), and others are not supported. In addition, as mentioned already, privileged containers are not supported on Windows. <h4 id="api">API</h4> There are no differences in how most of the Kubernetes APIs work for Windows. The subtleties around what's different come down to differences in the OS and container runtime. In certain situations, some properties on workload APIs such as Pod or Container were designed with an assumption that they are implemented on Linux, failing to run on Windows. At a high level, these OS concepts are different: Identity - Linux uses userID (UID) and groupID (GID) which are represented as integer types. User and group names are not canonical - they are just an alias in <code>/etc/groups</code> or <code>/etc/passwd</code> back to UID+GID. Windows uses a larger binary security identifier (SID) which is stored in the Windows Security Access Manager (SAM) database. This database is not shared between the host and containers, or between containers.
File permissions - Windows uses an access control list based on SIDs, rather than a bitmask of permissions and UID+GID
File paths - convention on Windows is to use <code>\</code> instead of <code>/</code>. The Go IO libraries typically accept both and just make it work, but when you're setting a path or command line that's interpreted inside a container, <code>\</code> may be needed.
Signals - Windows interactive

apps handle termination differently, and can implement one or more of these:
A UI thread handles well-defined messages including WM_CLOSE
Console apps handle ctrl-c or ctrl-break using a Control Handler
Services register a Service Control Handler function that can accept SERVICE_CONTROL_STOP control codes Exit Codes follow the same convention where 0 is success, nonzero is failure. The specific error codes may differ across Windows and Linux. However, exit codes passed from the Kubernetes components (kubelet, kube-proxy) are unchanged. <h5 id="v1-container">V1.Container</h5>
V1.Container.ResourceRequirements.limits.cpu and V1.Container.ResourceRequirements.limits.memory - Windows doesn't use hard limits for CPU allocations. Instead, a share system is used. The existing fields based on millicores are scaled into relative shares that are followed by the Windows scheduler. see: kuberuntime/helpers_windows.go, see: resource controls in Microsoft docs
Huge pages are not implemented in the Windows container runtime, and are not available. They require asserting a user privilege that's not configurable for containers.
V1.Container.ResourceRequirements.requests.cpu and V1.Container.ResourceRequirements.requests.memory - Requests are subtracted from node available resources, so they can be used to avoid overprovisioning a node. However, they cannot be used to guarantee resources in an overprovisioned node. They should be applied to all containers as a best practice if the operator wants to avoid overprovisioning entirely.
V1.Container.SecurityContext.allowPrivilegeEscalation - not possible on Windows, none of the capabilities are hooked up
V1.Container.SecurityContext.Capabilities - POSIX capabilities are not implemented on Windows
V1.Container.SecurityContext.privileged - Windows doesn't support privileged containers
V1.Container.SecurityContext.procMount - Windows doesn't have a /proc filesystem
V1.Container.SecurityContext.readOnlyRootFilesystem - not possible on Windows, write access is required for registry & system processes to run inside the container
V1.Container.SecurityContext.runAsGroup - not possible on Windows, no GID support
V1.Container.SecurityContext.runAsNonRoot - Windows does not have a root user. The closest equivalent is ContainerAdministrator which is an identity that doesn't exist on the node.
V1.Container.SecurityContext.runAsUser - not possible on Windows, no UID support as int.
V1.Container.SecurityContext.seLinuxOptions - not possible on Windows, no SELinux
V1.Container.terminationMessagePath - this has some limitations in that Windows doesn't support mapping single files. The default value is /dev/termination-log, which does work because it does not exist on Windows by default.
<h5 id="v1-pod">V1.Pod</

h5>
V1.Pod.hostIPC, v1.pod.hostpid - host namespace sharing is not possible on Windows
V1.Pod.hostNetwork - There is no Windows OS support to share the host network
V1.Pod.dnsPolicy - ClusterFirstWithHostNet - is not supported because Host Networking is not supported on Windows.
V1.Pod.podSecurityContext - see V1.PodSecurityContext below
V1.Pod.shareProcessNamespace - this is a beta feature, and depends on Linux namespaces which are not implemented on Windows. Windows cannot share process namespaces or the container's root filesystem. Only the network can be shared.
V1.Pod.terminationGracePeriodSeconds - this is not fully implemented in Docker on Windows, see: reference. The behavior today is that the ENTRYPOINT process is sent CTRL_SHUTDOWN_EVENT, then Windows waits 5 seconds by default, and finally shuts down all processes using the normal Windows shutdown behavior. The 5 second default is actually in the Windows registry inside the container, so it can be overridden when the container is built.
V1.Pod.volumeDevices - this is a beta feature, and is not implemented on Windows. Windows cannot attach raw block devices to pods.
V1.Pod.volumes - EmptyDir, Secret, ConfigMap, HostPath - all work and have tests in TestGrid
V1.emptyDirVolumeSource - the Node default medium is disk on Windows. Memory is not supported, as Windows does not have a built-in RAM disk.
V1.VolumeMount.mountPropagation - mount propagation is not supported on Windows.
<h5 id="v1-podsecuritycontext">V1.PodSecurityContext</h5> None of the PodSecurityContext fields work on Windows. They're listed here for reference. V1.PodSecurityContext.SELinuxOptions - SELinux is not available on Windows
V1.PodSecurityContext.RunAsUser - provides a UID, not available on Windows
V1.PodSecurityContext.RunAsGroup - provides a GID, not available on Windows
V1.PodSecurityContext.RunAsNonRoot - Windows does not have a root user. The closest equivalent is ContainerAdministrator which is an identity that doesn't exist on the node.
V1.PodSecurityContext.SupplementalGroups - provides GID, not available on Windows
V1.PodSecurityContext.Sysctls - these are part of the Linux sysctl interface. There's no equivalent on Windows.
<h2 id="troubleshooting">Getting Help and Troubleshooting</h2> Your main source of help for troubleshooting your Kubernetes cluster should start with this section. Some additional, Windows-specific troubleshooting help is included in this section. Logs are an important element of troubleshooting issues in Kubernetes. Make sure to include them any time you seek troubleshooting assistance from other contributors. Follow the instructions in the SIG-Windows contributing guide on gathering logs. How do I know start.ps1 completed successfully? You should see kubelet, kube-proxy, and (if you chose Flannel as your networking solution) flanneld host-agent processes running on your node, with running logs being displayed in separate PowerShell windows. In addition to this, your Windows node should be listed as "Ready"

in your Kubernetes cluster. Can I configure the Kubernetes node processes to run in the background as services? Kubelet and kube-proxy are already configured to run as native Windows Services, offering resiliency by re-starting the services automatically in the event of failure (for example a process crash). You have two options for configuring these node components as services. As native Windows Services Kubelet & kube-proxy can be run as native Windows Services using <code>sc.exe</code>. <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell"># Create the services for kubelet and kube-proxy in two separate commands sc.exe create <component_name> binPath= "<path_to_binary> --service <other_args>" # Please note that if the arguments contain spaces, they must be escaped. sc.exe create kubelet binPath= "C:\kubelet.exe --service --hostname-override 'minion' <other_args>" # Start the services Start-Service kubelet Start-Service kube-proxy # Stop the service Stop-Service kubelet (-Force) Stop-Service kube-proxy (-Force) # Query the service status Get-Service kubelet Get-Service kube-proxy </code></pre> Using nssm.exe You can also always use alternative service managers like nssm.exe to run these processes (flanneld, kubelet & kube-proxy) in the background for you. You can use this sample script, leveraging nssm.exe to register kubelet, kube-proxy, and flanneld.exe to run as Windows services in the background. <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell">register-svc.ps1 -NetworkMode <Network mode> -ManagementIP <Windows Node IP> -ClusterCIDR <Cluster subnet> -KubeDnsServiceIP <Kube-dns Service IP> -LogDir <Directory to place logs> # NetworkMode = The network mode l2bridge (flannel host-gw, also the default value) or overlay (flannel vxlan) chosen as a network solution # ManagementIP = The IP address assigned to the Windows node. You can use ipconfig to find this # ClusterCIDR = The cluster subnet range. (Default value 10.244.0.0/16) # KubeDnsServiceIP = The Kubernetes DNS service IP (Default value 10.96.0.10) # LogDir = The directory where kubelet and kube-proxy logs are redirected into their respective output files (Default value C:\k) </code></pre> If the above referenced script is not suitable, you can manually configure nssm.exe using the

following examples. <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell"># Register flanneld.exe nssm install flanneld C:\flannel\flanneld.exe nssm set flanneld AppParameters --kubeconfig-file=c:\k\config --iface=<ManagementIP> --ip-masq=1 --kube-subnet-mgr=1 nssm set flanneld AppEnvironmentExtra NODE_NAME=<hostname> nssm set flanneld AppDirectory C:\flannel nssm start flanneld # Register kubelet.exe # Microsoft releases the pause infrastructure container at mcr.microsoft.com/oss/kubernetes/pause:1.4.1 nssm install kubelet C:\k\kubelet.exe nssm set kubelet AppParameters --hostname-override=<hostname> --v=6 --pod-infra-container-image=mcr.microsoft.com/oss/kubernetes/pause:1.4.1 --resolv-conf="" --allow-privileged=true --enable-debugging-handlers --cluster-dns=<DNS-service-IP> --cluster-domain=cluster.local --kubeconfig=c:\k\config --hairpin-mode=promiscuous-bridge --image-pull-progress-deadline=20m --cgroups-per-qos=false --log-dir=<log directory> --logtostderr=false --enforce-node-allocatable="" --network-plugin=cni --cni-bin-dir=c:\k\cni --cni-conf-dir=c:\k\cni\config nssm set kubelet AppDirectory C:\k nssm start kubelet # Register kube-proxy.exe (l2bridge / host-gw) nssm install kube-proxy C:\k\kube-proxy.exe nssm set kube-proxy AppDirectory c:\k nssm set kube-proxy AppParameters --v=4 --proxy-mode=kernelspace --hostname-override=<hostname>--kubeconfig=c:\k\config --enable-dsr=false --log-dir=<log directory> --logtostderr=false nssm.exe set kube-proxy AppEnvironmentExtra KUBE_NETWORK=cbr0 nssm set kube-proxy DependOnService kubelet nssm start kube-proxy # Register kube-proxy.exe (overlay / vxlan) nssm install kube-proxy C:\k\kube-proxy.exe nssm set kube-proxy AppDirectory c:\k nssm set kube-proxy AppParameters --v=4 --proxy-mode=kernelspace --feature-gates="WinOverlay=true" --hostname-override=<hostname> --kubeconfig=c:\k\config --network-name=vxlan0 --source-vip=<source-vip> --enable-dsr=false --log-dir=<log directory> --logtostderr=false nssm set kube-proxy DependOnService kubelet nssm start kube-proxy </code></pre> For initial troubleshooting, you can use the following flags in nssm.exe to redirect stdout and stderr to a output file: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell">nssm <span

style="color:#a2f">set <Service Name> AppStdout C:\k\mysvc.log nssm set <Service Name> AppStderr C:\k\mysvc.log </code></pre> For additional details, see official nssm usage docs. My Windows Pods do not have network connectivity If you are using virtual machines, ensure that MAC spoofing is enabled on all the VM network adapter(s). My Windows Pods cannot ping external resources Windows Pods do not have outbound rules programmed for the ICMP protocol today. However, TCP/UDP is supported. When trying to demonstrate connectivity to resources outside of the cluster, please substitute <code>ping <IP></code> with corresponding <code>curl <IP></code> commands. If you are still facing problems, most likely your network configuration in cni.conf deserves some extra attention. You can always edit this static file. The configuration update will apply to any newly created Kubernetes resources. One of the Kubernetes networking requirements (see Kubernetes model) is for cluster communication to occur without NAT internally. To honor this requirement, there is an ExceptionList for all the communication where we do not want outbound NAT to occur. However, this also means that you need to exclude the external IP you are trying to query from the ExceptionList. Only then will the traffic originating from your Windows pods be SNAT'ed correctly to receive a response from the outside world. In this regard, your ExceptionList in <code>cni.conf</code> should look as follows: <pre><code class="language-conf" data-lang="conf">"ExceptionList": ["10.244.0.0/16", # Cluster subnet "10.96.0.0/12", # Service subnet "10.127.130.0/24" # Management (host) subnet] </code></pre> My Windows node cannot access NodePort service Local NodePort access from the node itself fails. This is a known limitation. NodePort access works from other nodes or external clients. vNICs and HNS endpoints of containers are being deleted This issue can be caused when the <code>hostname-override</code> parameter is not passed to kube-proxy. To resolve it, users need to pass the hostname to kube-proxy as follows: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell">C:\k\kube-proxy.exe --hostname-override=$(hostname) </code></pre> With flannel my nodes are having issues after rejoining a cluster Whenever a previously deleted node is being re-joined to the cluster, flannelD tries to assign a new pod subnet to the node. Users should remove the old pod subnet configuration files in the following paths: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell">Remove-Item C:\k\SourceVip.json Remove-Item C:\k\SourceVipRequest.json </code></pre> After

launching <code>start.ps1</code>, flanneld is stuck in "Waiting for the Network to be created" There are numerous reports of this issue which are being investigated; most likely it is a timing issue for when the management IP of the flannel network is set. A workaround is to simply relaunch start.ps1 or relaunch it manually as follows: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell">PS C:> [Environment]::SetEnvironmentVariable("NODE_NAME", "<Windows_Worker_Hostname>") PS C:> C:\flannel\flanneld.exe --kubeconfig-file=c:\k\config --iface=<Windows_Worker_Node_IP> --ip-masq=1 --kube-subnet-mgr=1 </code></pre> My Windows Pods cannot launch because of missing <code>/run/flannel/subnet.env</code> This indicates that Flannel didn't launch correctly. You can either try to restart flanneld.exe or you can copy the files over manually from <code>/run/flannel/subnet.env</code> on the Kubernetes master to <code>C:\run\flannel\subnet.env</code> on the Windows worker node and modify the <code>FLANNEL_SUBNET</code> row to a different number. For example, if node subnet 10.244.4.1/24 is desired: <pre><code class="language-env" data-lang="env">FLANNEL_NETWORK=10.244.0.0/16 FLANNEL_SUBNET=10.244.4.1/24 FLANNEL_MTU=1500 FLANNEL_IPMASQ=true </code></pre> My Windows node cannot access my services using the service IP This is a known limitation of the current networking stack on Windows. Windows Pods are able to access the service IP however. No network adapter is found when starting kubelet The Windows networking stack needs a virtual adapter for Kubernetes networking to work. If the following commands return no results (in an admin shell), virtual network creation â€” a necessary prerequisite for Kubelet to work â€” has failed: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell">Get-HnsNetwork | ? Name -ieq "cbr0" Get-NetAdapter | ? Name -Like "vEthernet (Ethernet*" </code></pre> Often it is worthwhile to modify the InterfaceName parameter of the start.ps1 script, in cases where the host's network adapter isn't "Ethernet". Otherwise, consult the output of the <code>start-kubelet.ps1</code> script to see if there are errors during virtual network creation. My Pods are stuck at "Container Creating" or restarting over and over Check that your pause image is compatible with your OS version. The instructions assume that both the OS and the containers are version 1803. If you have a later version of Windows, such as

an Insider build, you need to adjust the images accordingly. Please refer to the Microsoft's Docker repository for images. Regardless, both the pause image Dockerfile and the sample service expect the image to be tagged as :latest. DNS resolution is not properly working Check the DNS limitations for Windows in this section. <code>kubectl port-forward</code> fails with "unable to do port forwarding: wincat not found" This was implemented in Kubernetes 1.15 by including wincat.exe in the pause infrastructure container <code>mcr.microsoft.com/oss/kubernetes/pause:1.4.1</code>. Be sure to use these versions or newer ones. If you would like to build your own pause infrastructure container be sure to include wincat. My Kubernetes installation is failing because my Windows Server node is behind a proxy If you are behind a proxy, the following PowerShell environment variables must be defined: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-PowerShell" data-lang="PowerShell">[Environment]::SetEnvironmentVariable("HTTP_PROXY", "http://proxy.example.com:80/", [EnvironmentVariableTarget]::Machine) [Environment]::SetEnvironmentVariable("HTTPS_PROXY", "http://proxy.example.com:443/", [EnvironmentVariableTarget]::Machine) </code></pre> What is a <code>pause</code> container? In a Kubernetes Pod, an infrastructure or "pause" container is first created to host the container endpoint. Containers that belong to the same pod, including infrastructure and worker containers, share a common network namespace and endpoint (same IP and port space). Pause containers are needed to accommodate worker containers crashing or restarting without losing any of the networking configuration. The "pause" (infrastructure) image is hosted on Microsoft Container Registry (MCR). You can access it using <code>mcr.microsoft.com/oss/kubernetes/pause:1.4.1</code>. For more details, see the DOCKERFILE. <h3 id="further-investigation">Further investigation</h3> If these steps don't resolve your problem, you can get help running Windows containers on Windows nodes in Kubernetes through: StackOverflow Windows Server Container topic
Kubernetes Official Forum discuss.kubernetes.io
Kubernetes Slack #SIG-Windows Channel
<h2 id="reporting-issues-and-feature-requests">Reporting Issues and Feature Requests</h2> If you have what looks like a bug, or you would like to make a feature request, please use the GitHub issue tracking system. You can open issues on <a href="https://github.com/kubernetes/kubernetes/issues/new/

choose">GitHub and assign them to SIG-Windows. You should first search the list of issues in case it was reported previously and comment with your experience on the issue and add additional logs. SIG-Windows Slack is also a great avenue to get some initial support and troubleshooting ideas prior to creating a ticket. If filing a bug, please include detailed information about how to reproduce the problem, such as: Kubernetes version: kubectl version
Environment details: Cloud provider, OS distro, networking choice and configuration, and Docker version
Detailed steps to reproduce the problem
Relevant logs
Tag the issue sig/windows by commenting on the issue with <code>/sig windows</code> to bring it to a SIG-Windows member's attention
<h2 id="what-s-next">What's next</h2> We have a lot of features in our roadmap. An abbreviated high level list is included below, but we encourage you to view our roadmap project and help us make Windows support better by contributing. <h3 id="hyper-v-isolation">Hyper-V isolation</h3> Hyper-V isolation is requried to enable the following use cases for Windows containers in Kubernetes: Hypervisor-based isolation between pods for additional security
Backwards compatibility allowing a node to run a newer Windows Server version without requiring containers to be rebuilt
Specific CPU/NUMA settings for a pod
Memory isolation and reservations Hyper-V isolation support will be added in a later release and will require CRI-Containerd. <h3 id="deployment-with-kubeadm-and-cluster-api">Deployment with kubeadm and cluster API</h3> Kubeadm is becoming the de facto standard for users to deploy a Kubernetes cluster. Windows node support in kubeadm is currently a work-in-progress but a guide is available here. We are also making investments in cluster API to ensure Windows nodes are properly provisioned. <h2>Feedback</h2> Was this page helpful? <button class="btn btn-primary mb-4 feedback--yes">Yes</button> <button class="btn btn-primary mb-4 feedback--no">No</button> Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement. <script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=>{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=>{if(!gtag){console.log('!gtag');}gtag('event','click',

{'event_category':'Helpful','event_label':window.location.pathname,value});};const disableButtons=()=>{yes.disabled=true;yes.classList.add('feedback--button__disabled');no.disabled=true;no.classList.add('feedback--button__disabled');};yes.addEventListener('click',()=>{sendFeedback(1);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});no.addEventListener('click',()=>{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});</script> Last modified November 20, 2020 at 11:02 PM PST: Misc updates to intro-windows-in-kubernetes.md (7181cb54c) <i class="fa fa-edit fa-fw"/>Edit this page <i class="fa fa-edit fa-fw"/>Create child page <i class="fab fa-github fa-fw"/>Create an issue <nav id="TableOfContents">
Windows containers in Kubernetes
Supported Functionality and Limitations
Supported Functionality
Limitations
Getting Help and Troubleshooting
Further investigation
Reporting Issues and Feature Requests
What's next
Hyper-V isolation
Deployment with kubeadm and cluster API
</nav> <h1>Guide for scheduling Windows containers in Kubernetes</h1> Windows applications constitute a large portion of the services and applications that run in many organizations. This guide walks you through the steps to configure and deploy a Windows container in Kubernetes. <h2 id="objectives">Objectives</h2>
Configure an example

deployment to run Windows containers on the Windows node
(Optional) Configure an Active Directory Identity for your Pod using Group Managed Service Accounts (GMSA)
<h2 id="before-you-begin">Before you begin</h2>
Create a Kubernetes cluster that includes a master and a worker node running Windows Server
It is important to note that creating and deploying services and workloads on Kubernetes behaves in much the same way for Linux and Windows containers. Kubectl commands to interface with the cluster are identical. The example in the section below is provided simply to jumpstart your experience with Windows containers.
<h2 id="getting-started-deploying-a-windows-container">Getting Started: Deploying a Windows container</h2> To deploy a Windows container on Kubernetes, you must first create an example application. The example YAML file below creates a simple webserver application. Create a service spec named <code>win-webserver.yaml</code> with the contents below: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml">apiVersion: v1
kind: Service
metadata:
 name:
win-webserver
 labels:
 app:
win-webserver
spec:
 ports:
 # the port that this service should serve on
- port:
80

targetPort:
80

selector:

app:
win-webserver

type:
NodePort
<span

style="color:#bbb"/>--- apiVersion: apps/v1 kind: Deployment metadata: labels: app: win-webserver name: win-webserver spec: replicas: 2 selector: matchLabels: app: win-webserver template: metadata: labels: app: win-webserver name: win-webserver spec: containers: - name: windowswebserver image: mcr.microsoft.com/windows/servercore:ltsc2019 command: - powershell.exe - -command<span

style="color:#bbb"> - "<#code used from https://gist.github.com/19WAS85/5424431#> ; $$listener = New-Object System.Net.HttpListener ; $$listener.Prefixes.Add('http://*:80/') ; $$listener.Start() ; $$callerCounts = @{} ; Write-Host('Listening at http://*:80/') ; while ($$listener.IsListening) { ;$$context = $$listener.GetContext() ; $$requestUrl = $$context.Request.Url ;$$clientIP = $$context.Request.RemoteEndPoint.Address ;$$response = $$context.Response ;Write-Host '' ;Write-Host('> {0}' -f $$requestUrl) ; ;$$count = 1 ;$$k=$$callerCounts.Get_Item($$clientIP) ;if ($$k -ne $$null) { $$count += $$k } ;$$callerCounts.Set_Item($$clientIP, $$count) ;$$ip=(Get-NetAdapter | Get-NetIpAddress); $$header='<html><body><H1>Windows Container Web Server</H1>' ;$$callerCountsString='' ;$$callerCounts.Keys | % { $$callerCountsString+='<p>IP {0} callerCount {1} ' -f $$ip[1].IPAddress,$$callerCounts.Item($$_) } ;$$footer='</body></html>' ;$$content='{0}{1}{2}' -f $$header,$$callerCountsString,$$footer ;Write-Output $$content ;$$buffer = [System.Text.Encoding]::UTF8.GetBytes($$content) ;$$response.ContentLength64 = $$buffer.Length ;$$response.OutputStream.Write($$buffer, 0, $$buffer.Length) ;$$response.Close() ;$$responseStatus = $$response.StatusCode ;Write-Host('< {0}' -f $$responseStatus) } ; " nodeSelector: kubernetes.io/os: windows </code></pre> <blockquote class="note callout"> Note: Port mapping is also supported, but for simplicity in this example the container port 80 is exposed directly to the service. </blockquote> Check that all nodes are healthy: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash">kubectl get nodes </code></pre> Deploy the service and watch for pod updates: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash">kubectl apply -f win-webserver.yaml kubectl get pods -o wide -w </code></pre> When the service is deployed correctly both Pods are marked as Ready. To exit the watch command, press Ctrl+C. Check that the deployment succeeded. To verify: Two containers per pod on the Windows node, use <code>docker ps</code> Two pods listed from the Linux master, use <code>kubectl get pods</code> Node-to-pod communication across the network, <code>curl</code> port 80 of your pod IPs from the Linux master to check for a web server response Pod-to-pod communication, ping between pods (and across hosts, if you have more than one Windows node) using docker exec or kubectl exec Service-to-pod communication, <code>curl</code> the virtual service IP (seen under <code>kubectl get services</code>) from the Linux master and from individual pods Service discovery, <code>curl</code> the service

name with the Kubernetes default DNS suffix
Inbound connectivity, <code>curl</code> the NodePort from the Linux master or machines outside of the cluster
Outbound connectivity, <code>curl</code> external IPs from inside the pod using kubectl exec
<blockquote class="note callout"> Note: Windows container hosts are not able to access the IP of services scheduled on them due to current platform limitations of the Windows networking stack. Only Windows pods are able to access service IPs. </blockquote><h2 id="observability">Observability</h2><h3 id="capturing-logs-from-workloads">Capturing logs from workloads</h3> Logs are an important element of observability; they enable users to gain insights into the operational aspect of workloads and are a key ingredient to troubleshooting issues. Because Windows containers and workloads inside Windows containers behave differently from Linux containers, users had a hard time collecting logs, limiting operational visibility. Windows workloads for example are usually configured to log to ETW (Event Tracing for Windows) or push entries to the application event log. LogMonitor, an open source tool by Microsoft, is the recommended way to monitor configured log sources inside a Windows container. LogMonitor supports monitoring event logs, ETW providers, and custom application logs, piping them to STDOUT for consumption by <code>kubectl logs <pod></code>. Follow the instructions in the LogMonitor GitHub page to copy its binaries and configuration files to all your containers and add the necessary entrypoints for LogMonitor to push your logs to STDOUT. <h2 id="using-configurable-container-usernames">Using configurable Container usernames</h2> Starting with Kubernetes v1.16, Windows containers can be configured to run their entrypoints and processes with different usernames than the image defaults. The way this is achieved is a bit different from the way it is done for Linux containers. Learn more about it here. <h2 id="managing-workload-identity-with-group-managed-service-accounts">Managing Workload Identity with Group Managed Service Accounts</h2> Starting with Kubernetes v1.14, Windows container workloads can be configured to use Group Managed Service Accounts (GMSA). Group Managed Service Accounts are a specific type of Active Directory account that provides automatic password management, simplified service principal name (SPN) management, and the ability to delegate the management to other administrators across multiple servers. Containers configured with a GMSA can access external Active Directory Domain resources while carrying the identity configured with the GMSA. Learn more about configuring and using GMSA for Windows containers here. <h2 id="taints-and-tolerations">Taints and Tolerations</h2> Users today need to use some combination of taints and node selectors in order to keep Linux and Windows workloads on their respective OS-specific nodes. This likely imposes a burden only on Windows users. The recommended approach is outlined below, with one of its main goals being that this approach should not break compatibility for existing Linux workloads. <h3 id="ensuring-os-specific-workloads-land-on-the-

appropriate-container-host">Ensuring OS-specific workloads land on the appropriate container host</h3> Users can ensure Windows containers can be scheduled on the appropriate host using Taints and Tolerations. All Kubernetes nodes today have the following default labels: kubernetes.io/os = [windows|linux] kubernetes.io/arch = [amd64|arm64|...] If a Pod specification does not specify a nodeSelector like <code>"kubernetes.io/os": windows</code>, it is possible the Pod can be scheduled on any host, Windows or Linux. This can be problematic since a Windows container can only run on Windows and a Linux container can only run on Linux. The best practice is to use a nodeSelector. However, we understand that in many cases users have a pre-existing large number of deployments for Linux containers, as well as an ecosystem of off-the-shelf configurations, such as community Helm charts, and programmatic Pod generation cases, such as with Operators. In those situations, you may be hesitant to make the configuration change to add nodeSelectors. The alternative is to use Taints. Because the kubelet can set Taints during registration, it could easily be modified to automatically add a taint when running on Windows only. For example: <code>--register-with-taints='os=windows:NoSchedule'</code> By adding a taint to all Windows nodes, nothing will be scheduled on them (that includes existing Linux Pods). In order for a Windows Pod to be scheduled on a Windows node, it would need both the nodeSelector to choose Windows, and the appropriate matching toleration. <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml">nodeSelector: kubernetes.io/os: windows node.kubernetes.io/windows-build: '10.0.17763' tolerations: - key: "os" operator: "Equal" value: "windows" effect: "NoSchedule" </code></pre> <h3 id="handling-multiple-windows-versions-in-the-same-cluster">Handling multiple Windows versions in the same cluster</h3> The Windows Server version used by each pod must match that of the node. If you want to use multiple Windows Server versions in the same cluster, then you should

set additional node labels and nodeSelectors. Kubernetes 1.17 automatically adds a new label <code>node.kubernetes.io/windows-build</code> to simplify this. If you're running an older version, then it's recommended to add this label manually to Windows nodes. This label reflects the Windows major, minor, and build number that need to match for compatibility. Here are values used today for each Windows Server version. <table><thead><tr><th>Product Name</th><th>Build Number(s)</th></tr></thead><tbody><tr><td>Windows Server 2019</td><td>10.0.17763</td></tr><tr><td>Windows Server version 1809</td><td>10.0.17763</td></tr><tr><td>Windows Server version 1903</td><td>10.0.18362</td></tr></tbody></table><h3 id="simplifying-with-runtimeclass">Simplifying with RuntimeClass</h3> RuntimeClass can be used to simplify the process of using taints and tolerations. A cluster administrator can create a <code>RuntimeClass</code> object which is used to encapsulate these taints and tolerations. Save this file to <code>runtimeClasses.yml</code>. It includes the appropriate <code>nodeSelector</code> for the Windows OS, architecture, and version. <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml">apiVersion: node.k8s.io/v1 kind: RuntimeClass metadata: name: windows-2019 handler: 'docker' scheduling: nodeSelector: kubernetes.io/os: 'windows' kubernetes.io/arch: 'amd64' node.kubernetes.io/windows-build: '10.0.17763' tolerations: - effect:<span

style="color:#bbb"> NoSchedule key: os operator: Equal value: "windows" </code></pre> Run <code>kubectl create -f runtimeClasses.yml</code> using as a cluster administrator
Add <code>runtimeClassName: windows-2019</code> as appropriate to Pod specs For example: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml">apiVersion: apps/v1 kind: Deployment metadata: name: iis-2019 labels: app: iis-2019 spec: replicas: 1 template: metadata: name: iis-2019 labels: app: iis-2019 spec: runtimeClassName: windows-2019 </

span>containers: - name: iis image: mcr.microsoft.com/windows/servercore/iis:windowsservercore-ltsc2019 resources: limits: cpu: 1 memory: 800Mi requests: cpu: .1 memory: 300Mi ports: - containerPort: 80 selector: matchLabels: app: iis-2019 --- apiVersion: v1 kind: Service metadata: name: iis spec: type: LoadBalancer <span style="color:#a2f;font-

weight:700">ports: - protocol: TCP port: 80 selector: app: iis-2019 </code></pre> <h2>Feedback</h2> Was this page helpful? <button class="btn btn-primary mb-4 feedback--yes">Yes</button> <button class="btn btn-primary mb-4 feedback--no">No</button> Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement. <script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=>{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=>{if(!gtag){console.log('!gtag');}gtag('event','click',{'event_category':'Helpful','event_label':window.location.pathname,value});};const disableButtons=()=>{yes.disabled=true;yes.classList.add('feedback--button__disabled');no.disabled=true;no.classList.add('feedback--button__disabled');};yes.addEventListener('click',()=>{sendFeedback(1);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});no.addEventListener('click',()=>{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});</script> Last modified November 23, 2020 at 4:02 PM PST: Fix broken link (d38bbe24e) <i class="fa fa-edit fa-fw"/>Edit this page <a href="https://github.com/kubernetes/website/new/master/content/en/docs/setup/production-environment/windows/user-guide-windows-containers.md?filename=change-me.md&value=---%0Atitle%3A+

%22Long+Page+Title%22%0AlinkTitle%3A+%22Short+Nav+Title%22%0Aweight%3A+100%0Adescription%3A+%3E-%0A+++++Page+description+for+heading+and+indexes.%0A---%0A%0A%23%23+Heading%0A%0AEdit+this+template+to+create+your+new+page.%0A%0A%2A+Give+it+a+good+name%2C+ending+in+%60.md%60+-+e.g.+%60getting-started.md%60%0A%2A+Edit+the+%22front+matter%22+section+at+the+top+of+the+page+%28weight+controls+how+its+ordered+amongst+other+pages+in+the+same+direct%28%3C80+characters%3B+use+the+extended+description+field+for+more+detail%0A" target="_blank"><i class="fa fa-edit fa-fw"/>Create child page <i class="fab fa-github fa-fw"/>Create an issue <nav id="TableOfContents">
Objectives
Before you begin
Getting Started: Deploying a Windows container
Observability
Capturing logs from workloads
Using configurable Container usernames
Managing Workload Identity with Group Managed Service Accounts
Taints and Tolerations
Ensuring OS-specific workloads land on the appropriate container host
Handling multiple Windows versions in the same cluster
Simplifying with RuntimeClass
</nav> <h1>Best practices</h1> <hr class="panel-line"/> <h5>Considerations for large clusters</h5><p/> <h5>Running in multiple zones</h5><p/> <h5>Validate node setup</h5><p/> <h5>PKI certificates and requirements</h5><p/> <h1>Considerations for large clusters</h1><p>A cluster is a set of nodes (physical or virtual machines) running Kubernetes agents, managed by the control plane. Kubernetes v1.20 supports clusters with up to 5000 nodes. More specifically, Kubernetes is designed to accommodate configurations that meet

all of the following criteria: No more than 100 pods per node
No more than 5000 nodes
No more than 150000 total pods
No more than 300000 total containers You can scale your cluster by adding or removing nodes. The way you do this depends on how your cluster is deployed. <h2 id="quota-issues">Cloud provider resource quotas</h2> To avoid running into cloud provider quota issues, when creating a cluster with many nodes, consider: Request a quota increase for cloud resources such as:
Computer instances
CPUs
Storage volumes
In-use IP addresses
Packet filtering rule sets
Number of load balancers
Network subnets
Log streams
Gate the cluster scaling actions to brings up new nodes in batches, with a pause between batches, because some cloud providers rate limit the creation of new instances.
<h2 id="control-plane-components">Control plane components</h2> For a large cluster, you need a control plane with sufficient compute and other resources. Typically you would run one or two control plane instances per failure zone, scaling those instances vertically first and then scaling horizontally after reaching the point of falling returns to (vertical) scale. You should run at least one instance per failure zone to provide fault-tolerance. Kubernetes nodes do not automatically steer traffic towards control-plane endpoints that are in the same failure zone; however, your cloud provider might have its own mechanisms to do this. For example, using a managed load balancer, you configure the load balancer to send traffic that originates from the kubelet and Pods in failure zone A, and direct that traffic only to the control plane hosts that are also in zone A. If a single control-plane host or endpoint failure zone A goes offline, that means that all the control-plane traffic for nodes in zone A is now being sent between zones. Running multiple control plane hosts in each zone makes that outcome less likely. <h3 id="etcd-storage">etcd storage</h3> To improve performance of large clusters, you can store Event objects in a separate dedicated etcd instance. When creating a cluster, you can (using custom tooling): start and configure additional etcd instance
configure the API server to use it for storing events
<h2 id="addon-resources">Addon resources</h2> Kubernetes resource limits help to minimise the impact of memory leaks and other ways that pods and containers can impact on other components. These resource limits can and should apply to addon just as they apply to application workloads. For example, you can set CPU and memory limits for a logging component: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml"> ...   </

span>containers: - name: fluentd-cloud-logging image: fluent/fluentd-kubernetes-daemonset:v1 resources: limits: cpu: 100m memory: 200Mi </code></pre> Addons' default limits are typically based on data collected from experience running each addon on small or medium Kubernetes clusters. When running on large clusters, addons often consume more of some resources than their default limits. If a large cluster is deployed without adjusting these values, the addon(s) may continuously get killed because they keep hitting the memory limit. Alternatively, the addon may run but with poor performance due to CPU time slice restrictions. To avoid running into cluster addon resource issues, when creating a cluster with many nodes, consider the following: Some addons scale vertically - there is one replica of the addon for the cluster or serving a whole failure zone. For these addons, increase requests and limits as you scale out your cluster.
Many addons scale horizontally - you add capacity by running more pods - but with a very large cluster you may also need to raise CPU or memory limits slightly. The VerticalPodAutoscaler can run in recommender mode to provide suggested figures for requests and limits.
Some addons run as one copy per node, controlled by a DaemonSet: for example, a node-level log aggregator. Similar to the case with horizontally-scaled addons, you may also need to raise CPU or memory limits slightly.
<h2 id="what-s-next">What's next</h2> <code>VerticalPodAutoscaler</code> is a custom resource that you can deploy into your cluster to help you manage resource requests and limits for pods.
Visit Vertical Pod Autoscaler to learn more about <code>VerticalPodAutoscaler</code> and how you can use it to scale cluster components, including cluster-critical addons. The cluster autoscaler integrates with a number of cloud providers to help you run the right number of nodes for the level of resource demand in your cluster. <h2>Feedback</h2> Was this page helpful? <button class="btn btn-primary mb-4 feedback--

yes">Yes</button> <button class="btn btn-primary mb-4 feedback--no">No</button> Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement. <script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=>{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=>{if(!gtag){console.log('!gtag');}gtag('event','click',{'event_category':'Helpful','event_label':window.location.pathname,value});};const disableButtons=()=>{yes.disabled=true;yes.classList.add('feedback--button__disabled');no.disabled=true;no.classList.add('feedback--button__disabled');};yes.addEventListener('click',()=>{sendFeedback(1);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});no.addEventListener('click',()=>{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});</script> Last modified October 08, 2020 at 2:40 AM PST: Add advice about control plane resilience for large clusters (f80591272) <i class="fa fa-edit fa-fw"/>Edit this page <i class="fa fa-edit fa-fw"/>Create child page <i class="fab fa-github fa-fw"/>Create an issue <nav id="TableOfContents">
Cloud provider

resource quotas
Control plane components
etcd storage
Addon resources
What's next
</nav> <h1>Running in multiple zones</h1> This page describes running Kubernetes across multiple zones. <h2 id="background">Background</h2> Kubernetes is designed so that a single Kubernetes cluster can run across multiple failure zones, typically where these zones fit within a logical grouping called a region. Major cloud providers define a region as a set of failure zones (also called availability zones) that provide a consistent set of features: within a region, each zone offers the same APIs and services. Typical cloud architectures aim to minimize the chance that a failure in one zone also impairs services in another zone. <h2 id="control-plane-behavior">Control plane behavior</h2> All control plane components support running as a pool of interchangable resources, replicated per component. When you deploy a cluster control plane, place replicas of control plane components across multiple failure zones. If availability is an important concern, select at least three failure zones and replicate each individual control plane component (API server, scheduler, etcd, cluster controller manager) across at least three failure zones. If you are running a cloud controller manager then you should also replicate this across all the failure zones you selected. <blockquote class="note callout"> Note: Kubernetes does not provide cross-zone resilience for the API server endpoints. You can use various techniques to improve availability for the cluster API server, including DNS round-robin, SRV records, or a third-party load balancing solution with health checking. </blockquote><h2 id="node-behavior">Node behavior</h2> Kubernetes automatically spreads the Pods for workload resources (such as Deployment or StatefulSet) across different nodes in a cluster. This spreading helps reduce the impact of failures. When nodes start up, the kubelet on each node automatically adds labels to the Node object that represents that specific kubelet in the Kubernetes API. These labels can include zone information. If your cluster spans multiple zones or regions, you can use node labels in conjunction with Pod topology spread constraints to control how Pods are spread across your cluster among fault domains: regions, zones, and even

specific nodes. These hints enable the scheduler to place Pods for better expected availability, reducing the risk that a correlated failure affects your whole workload. For example, you can set a constraint to make sure that the 3 replicas of a StatefulSet are all running in different zones to each other, whenever that is feasible. You can define this declaratively without explicitly defining which availability zones are in use for each workload. <h3 id="distributing-nodes-across-zones">Distributing nodes across zones</h3> Kubernetes' core does not create nodes for you; you need to do that yourself, or use a tool such as the Cluster API to manage nodes on your behalf. Using tools such as the Cluster API you can define sets of machines to run as worker nodes for your cluster across multiple failure domains, and rules to automatically heal the cluster in case of whole-zone service disruption. <h2 id="manual-zone-assignment-for-pods">Manual zone assignment for Pods</h2> You can apply node selector constraints to Pods that you create, as well as to Pod templates in workload resources such as Deployment, StatefulSet, or Job. <h2 id="storage-access-for-zones">Storage access for zones</h2> When persistent volumes are created, the <code>PersistentVolumeLabel</code> admission controller automatically adds zone labels to any PersistentVolumes that are linked to a specific zone. The scheduler then ensures, through its <code>NoVolumeZoneConflict</code> predicate, that pods which claim a given PersistentVolume are only placed into the same zone as that volume. You can specify a StorageClass for PersistentVolumeClaims that specifies the failure domains (zones) that the storage in that class may use. To learn about configuring a StorageClass that is aware of failure domains or zones, see Allowed topologies. <h2 id="networking">Networking</h2> By itself, Kubernetes does not include zone-aware networking. You can use a network plugin to configure cluster networking, and that network solution might have zone-specific elements. For example, if your cloud provider supports Services with <code>type=LoadBalancer</code>, the load balancer might only send traffic to Pods running in the same zone as the load balancer element processing a given connection. Check your cloud provider's documentation for details. For custom or on-premises deployments,

similar considerations apply. Service and Ingress behavior, including handling of different failure zones, does vary depending on exactly how your cluster is set up. <h2 id="fault-recovery">Fault recovery</h2> When you set up your cluster, you might also need to consider whether and how your setup can restore service if all the failure zones in a region go off-line at the same time. For example, do you rely on there being at least one node able to run Pods in a zone?
Make sure that any cluster-critical repair work does not rely on there being at least one healthy node in your cluster. For example: if all nodes are unhealthy, you might need to run a repair Job with a special toleration so that the repair can complete enough to bring at least one node into service. Kubernetes doesn't come with an answer for this challenge; however, it's something to consider. <h2 id="what-s-next">What's next</h2> To learn how the scheduler places Pods in a cluster, honoring the configured constraints, visit Scheduling and Eviction. <h2>Feedback</h2> Was this page helpful? <button class="btn btn-primary mb-4 feedback--yes">Yes</button> <button class="btn btn-primary mb-4 feedback--no">No</button> Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement. <script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=>{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=>{if(!gtag){console.log('!gtag');}gtag('event','click',{'event_category':'Helpful','event_label':window.location.pathname,value});};const disableButtons=()=>{yes.disabled=true;yes.classList.add('feedback--button__disabled');no.disabled=true;no.classList.add('feedback--button__disabled');};yes.addEventListener('click',()=>{sendFeedback(1);disableButtons();document.querySelector('.feedback--

response').classList.remove('feedback--response__hidden');});no.addEventListener('click',()=>{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});</script> Last modified December 02, 2020 at 12:52 PM PST: Revise multiple zones (a6e8f8bca) <i class="fa fa-edit fa-fw"/>Edit this page <i class="fa fa-edit fa-fw"/>Create child page <i class="fab fa-github fa-fw"/>Create an issue <nav id="TableOfContents">
Background
Control plane behavior
Node behavior
Distributing nodes across zones
Manual zone assignment for Pods
Storage access for zones
Networking
Fault recovery
What's next
</nav> <h1>Validate node setup</h1><h2 id="node-conformance-test">Node Conformance Test</h2> Node conformance test is a containerized test framework that provides a system verification and functionality test for a node. The test validates whether the node meets the minimum requirements for Kubernetes; a node that passes the test is qualified to join a Kubernetes cluster. <h2 id="node-prerequisite">Node Prerequisite</h2> To run node conformance test, a node must satisfy the same prerequisites as a standard Kubernetes node. At a minimum, the node should have the following daemons installed: Container Runtime (Docker)
Kubelet
<h2 id="running-node-conformance-test">Running Node Conformance Test</h2> To run the node conformance test, perform the following steps: Work out the value of the <code>--kubeconfig</code> option for the kubelet; for example: <code>--kubeconfig=/var/lib/kubelet/config.yaml</code>. Because the test framework starts a local control plane

to test the kubelet, use <code>http://localhost:8080</code> as the URL of the API server. There are some other kubelet command line parameters you may want to use:
<code>--pod-cidr</code>: If you are using <code>kubenet</code>, you should specify an arbitrary CIDR to Kubelet, for example <code>--pod-cidr=10.180.0.0/24</code>.
<code>--cloud-provider</code>: If you are using <code>--cloud-provider=gce</code>, you should remove the flag to run the test.
<ol start="2">
Run the node conformance test with command: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell"># $CONFIG_DIR is the pod manifest path of your Kubelet. # $LOG_DIR is the test output path. sudo docker run -it --rm --privileged --net=host \ -v /:/rootfs -v $CONFIG_DIR:$CONFIG_DIR -v $LOG_DIR:/var/result \ k8s.gcr.io/node-test:0.2 </code></pre> <h2 id="running-node-conformance-test-for-other-architectures">Running Node Conformance Test for Other Architectures</h2> Kubernetes also provides node conformance test docker images for other architectures: <table><thead><tr><th>Arch</th><th align="center">Image</th></tr></thead><tbody><tr><td>amd64</td><td align="center">node-test-amd64</td></tr><tr><td>arm</td><td align="center">node-test-arm</td></tr><tr><td>arm64</td><td align="center">node-test-arm64</td></tr></tbody></table><h2 id="running-selected-test">Running Selected Test</h2> To run specific tests, overwrite the environment variable <code>FOCUS</code> with the regular expression of tests you want to run. <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">sudo docker run -it --rm --privileged --net=host \ -v /:/rootfs:ro -v $CONFIG_DIR:$CONFIG_DIR -v $LOG_DIR:/var/result \ -e FOCUS=MirrorPod \ # Only run MirrorPod test k8s.gcr.io/node-test:0.2 </code></pre> To skip specific tests, overwrite the environment variable <code>SKIP</code> with the regular expression of tests you want to skip. <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">sudo docker run -it --rm --privileged --net=host \ <span

style="color:#b62;font-weight:700"/> -v /:/rootfs:ro -v $CONFIG_DIR:$CONFIG_DIR -v $LOG_DIR:/var/result \ -e SKIP=MirrorPod \ # Run all conformance tests but skip MirrorPod test k8s.gcr.io/node-test:0.2 </code></pre> Node conformance test is a containerized version of node e2e test. By default, it runs all conformance tests. Theoretically, you can run any node e2e test if you configure the container and mount required volumes properly. But it is strongly recommended to only run conformance test, because it requires much more complex configuration to run non-conformance test. <h2 id="caveats">Caveats</h2>
The test leaves some docker images on the node, including the node conformance test image and images of containers used in the functionality test.
The test leaves dead containers on the node. These containers are created during the functionality test. <h2>Feedback</h2> Was this page helpful? <button class="btn btn-primary mb-4 feedback--yes">Yes</button> <button class="btn btn-primary mb-4 feedback--no">No</button> Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement. <script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=>{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=>{if(!gtag){console.log('!gtag');}gtag('event','click',{'event_category':'Helpful','event_label':window.location.pathname,value});};const disableButtons=()=>{yes.disabled=true;yes.classList.add('feedback--button__disabled');no.disabled=true;no.classList.add('feedback--button__disabled');};yes.addEventListener('click',()=>{sendFeedback(1);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});no.addEventListener('click',()=>{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response__hidden');});</script> Last modified October 13, 2020 at 10:17 AM PST: <a href="https://github.com/kubernetes/website/commit/

e2457876419a4cbb792350b1faaa9849d5f7000b">fix node-conformance apiserver adress error (e24578764) <i class="fa fa-edit fa-fw"/>Edit this page <i class="fa fa-edit fa-fw"/>Create child page <i class="fab fa-github fa-fw"/>Create an issue <nav id="TableOfContents">
Node Conformance Test
Node Prerequisite
Running Node Conformance Test
Running Node Conformance Test for Other Architectures
Running Selected Test
Caveats
</nav> <h1>PKI certificates and requirements</h1> Kubernetes requires PKI certificates for authentication over TLS. If you install Kubernetes with kubeadm, the certificates that your cluster requires are automatically generated. You can also generate your own certificates -- for example, to keep your private keys more secure by not storing them on the API server. This page explains the certificates that your cluster requires. <h2 id="how-certificates-are-used-by-your-cluster">How certificates are used by your cluster</h2> Kubernetes requires PKI for the following operations: Client certificates for the kubelet to authenticate to the API server
Server certificate for the API server endpoint
Client certificates for administrators of the cluster to authenticate to the API server
Client certificates for the API server to talk to the kubelets
Client certificate for the API server to talk to etcd
Client certificate/kubeconfig for the controller manager to talk to the API server
Client certificate/kubeconfig for the scheduler to talk to the API server.
Client and server certificates for the front-proxy
<blockquote class="note callout"> Note: <code>front-proxy</code> certificates are required only if you run kube-proxy to support an extension API server. </

blockquote> etcd also implements mutual TLS to authenticate clients and peers. <h2 id="where-certificates-are-stored">Where certificates are stored</h2> If you install Kubernetes with kubeadm, certificates are stored in <code>/etc/kubernetes/pki</code>. All paths in this documentation are relative to that directory. <h2 id="configure-certificates-manually">Configure certificates manually</h2> If you don't want kubeadm to generate the required certificates, you can create them in either of the following ways. <h3 id="single-root-ca">Single root CA</h3> You can create a single root CA, controlled by an administrator. This root CA can then create multiple intermediate CAs, and delegate all further creation to Kubernetes itself. Required CAs: <table><thead><tr><th>path</th><th>Default CN</th><th>description</th></tr></thead><tbody><tr><td>ca.crt,key</td><td>kubernetes-ca</td><td>Kubernetes general CA</td></tr><tr><td>etcd/ca.crt,key</td><td>etcd-ca</td><td>For all etcd-related functions</td></tr><tr><td>front-proxy-ca.crt,key</td><td>kubernetes-front-proxy-ca</td><td>For the front-end proxy</td></tr></tbody></table> On top of the above CAs, it is also necessary to get a public/private key pair for service account management, <code>sa.key</code> and <code>sa.pub</code>. <h3 id="all-certificates">All certificates</h3> If you don't wish to copy the CA private keys to your cluster, you can generate all certificates yourself. Required certificates: <table><thead><tr><th>Default CN</th><th>Parent CA</th><th>O (in Subject)</th><th>kind</th><th>hosts (SAN)</th></tr></thead><tbody><tr><td>kube-etcd</td><td>etcd-ca</td><td/><td>server, client</td><td><code>localhost</code>, <code>127.0.0.1</code></td></tr><tr><td>kube-etcd-peer</td><td>etcd-ca</td><td/><td>server, client</td><td><code><hostname></code>, <code><Host_IP></code>, <code>localhost</code>, <code>127.0.0.1</code></td></tr><tr><td>kube-etcd-healthcheck-client</td><td>etcd-ca</td><td/><td>client</td><td/></tr><tr><td>kube-apiserver-etcd-client</td><td>etcd-ca</td><td>system:masters</td><td>client</td><td/></tr><tr><td>kube-apiserver</td><td>kubernetes-ca</td><td/><td>server</td><td><code><hostname></code>, <code><Host_IP></code>, <code><advertise_IP></code>, <code>[1]</code></td></tr><tr><td>kube-apiserver-kubelet-client</td><td>kubernetes-ca</td><td>system:masters</td><td>client</td><td/></tr><tr><td>front-proxy-client</td><td>kubernetes-front-proxy-ca</td><td/><td>client</td><td/></tr></tbody></table> [1]: any other IP or DNS name you contact your cluster on (as used by kubeadm the load balancer stable IP and/or DNS name, <code>kubernetes</code>, <code>kubernetes.default</code>, <code>kubernetes.default.svc</code>, <code>kubernetes.default.svc.cluster</code>, <code>kubernetes.default.svc.cluster.local</code>) where <code>kind</code> maps to one or more of the x509 key usage types: <table><thead><tr><th>kind</th><th>Key usage</th></tr></thead><tbody><tr><td>server</td><td>digital signature, key encipherment, server auth</td></tr><tr><td>client</td><td>digital

signature, key encipherment, client auth</td></tr></tbody></table><blockquote class="note callout"> Note: Hosts/SAN listed above are the recommended ones for getting a working cluster; if required by a specific setup, it is possible to add additional SANs on all the server certificates. </blockquote><blockquote class="note callout"> Note: For kubeadm users only: The scenario where you are copying to your cluster CA certificates without private keys is referred as external CA in the kubeadm documentation.
If you are comparing the above list with a kubeadm generated PKI, please be aware that <code>kube-etcd</code>, <code>kube-etcd-peer</code> and <code>kube-etcd-healthcheck-client</code> certificates are not generated in case of external etcd. </blockquote><h3 id="certificate-paths">Certificate paths</h3> Certificates should be placed in a recommended path (as used by kubeadm). Paths should be specified using the given argument regardless of location. <table><thead><tr><th>Default CN</th><th>recommended key path</th><th>recommended cert path</th><th>command</th><th>key argument</th><th>cert argument</th></tr></thead><tbody><tr><td>etcd-ca</td><td>etcd/ca.key</td><td>etcd/ca.crt</td><td>kube-apiserver</td><td/><td>--etcd-cafile</td></tr><tr><td>kube-apiserver-etcd-client</td><td>apiserver-etcd-client.key</td><td>apiserver-etcd-client.crt</td><td>kube-apiserver</td><td>--etcd-keyfile</td><td>--etcd-certfile</td></tr><tr><td>kubernetes-ca</td><td>ca.key</td><td>ca.crt</td><td>kube-apiserver</td><td/><td>--client-ca-file</td></tr><tr><td>kubernetes-ca</td><td>ca.key</td><td>ca.crt</td><td>kube-controller-manager</td><td>--cluster-signing-key-file</td><td>--client-ca-file, --root-ca-file, --cluster-signing-cert-file</td></tr><tr><td>kube-apiserver</td><td>apiserver.key</td><td>apiserver.crt</td><td>kube-apiserver</td><td>--tls-private-key-file</td><td>--tls-cert-file</td></tr><tr><td>kube-apiserver-kubelet-client</td><td>apiserver-kubelet-client.key</td><td>apiserver-kubelet-client.crt</td><td>kube-apiserver</td><td>--kubelet-client-key</td><td>--kubelet-client-certificate</td></tr><tr><td>front-proxy-ca</td><td>front-proxy-ca.key</td><td>front-proxy-ca.crt</td><td>kube-apiserver</td><td/><td>--requestheader-client-ca-file</td></tr><tr><td>front-proxy-ca</td><td>front-proxy-ca.key</td><td>front-proxy-ca.crt</td><td>kube-controller-manager</td><td/><td>--requestheader-client-ca-file</td></tr><tr><td>front-proxy-client</td><td>front-proxy-client.key</td><td>front-proxy-client.crt</td><td>kube-apiserver</td><td>--proxy-client-key-file</td><td>--proxy-client-cert-file</td></tr><tr><td>etcd-ca</td><td>etcd/ca.key</td><td>etcd/ca.crt</td><td>etcd</td><td/><td>--trusted-ca-file, --peer-trusted-ca-file</td></tr><tr><td>kube-etcd</td><td>etcd/server.key</td><td>etcd/server.crt</td><td>etcd</td><td>--key-file</td><td>--cert-file</td></tr><tr><td>kube-etcd-peer</td><td>etcd/peer.key</td><td>etcd/peer.crt</td><td>etcd</td><td>--peer-key-file</td><td>--peer-cert-file</td></tr><tr><td>etcd-ca</td><td/><td>etcd/ca.crt</td><td>etcdctl</td><td/><td>--cacert</td></tr><tr><td>kube-etcd-healthcheck-client</td><td>etcd/healthcheck-client.key</td><td>etcd/healthcheck-client.crt</td><td>etcdctl</td><td>--key</td><td>--cert</

td></tr></tbody></table> Same considerations apply for the service account key pair: <table><thead><tr><th>private key path</th><th>public key path</th><th>command</th><th>argument</th></tr></thead><tbody><tr><td>sa.key</td><td/><td>kube-controller-manager</td><td>--service-account-private-key-file</td></tr><tr><td/><td>sa.pub</td><td>kube-apiserver</td><td>--service-account-key-file</td></tr></tbody></table><h2 id="configure-certificates-for-user-accounts">Configure certificates for user accounts</h2> You must manually configure these administrator account and service accounts: <table><thead><tr><th>filename</th><th>credential name</th><th>Default CN</th><th>O (in Subject)</th></tr></thead><tbody><tr><td>admin.conf</td><td>default-admin</td><td>kubernetes-admin</td><td>system:masters</td></tr><tr><td>kubelet.conf</td><td>default-auth</td><td>system:node:<code><nodeName></code> (see note)</td><td>system:nodes</td></tr><tr><td>controller-manager.conf</td><td>default-controller-manager</td><td>system:kube-controller-manager</td><td/></tr><tr><td>scheduler.conf</td><td>default-scheduler</td><td>system:kube-scheduler</td><td/></tr></tbody></table><blockquote class="note callout"> Note: The value of <code><nodeName></code> for <code>kubelet.conf</code> must match precisely the value of the node name provided by the kubelet as it registers with the apiserver. For further details, read the Node Authorization. </blockquote> For each config, generate an x509 cert/key pair with the given CN and O. Run <code>kubectl</code> as follows for each config: <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">KUBECONFIG=<filename> kubectl config set-cluster default-cluster --server=https://<host ip>:6443 --certificate-authority <path-to-kubernetes-ca> --embed-certs KUBECONFIG=<filename> kubectl config set-credentials <credential-name> --client-key <path-to-key>.pem --client-certificate <path-to-cert>.pem --embed-certs KUBECONFIG=<filename> kubectl config set-context default-system --cluster default-cluster --user <credential-name> KUBECONFIG=<filename> kubectl config use-context default-system </code></pre> These files are used as follows: <table><thead><tr><th>filename</th><th>command</th><th>comment</th></tr></thead><tbody><tr><td>admin.conf</td><td>kubectl</td><td>Configures administrator user for the cluster</td></tr><tr><td>kubelet.conf</td><td>kubelet</td><td>One required for each node in the cluster.</td></tr><tr><td>controller-manager.conf</td><td>kube-controller-manager</td><td>Must be added to manifest in <code>manifests/kube-controller-manager.yaml</code></td></tr><tr><td>scheduler.conf</td><td>kube-scheduler</td><td>Must be added to manifest in <code>manifests/kube-scheduler.yaml</code></td></

tr></tbody></table> <h2>Feedback</h2> Was this page helpful? <button class="btn btn-primary mb-4 feedback--yes">Yes</button> <button class="btn btn-primary mb-4 feedback--no">No</button> Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement. <script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=>{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=>{if(!gtag){console.log('!gtag');}gtag('event','click',{'event_category':'Helpful','event_label':window.location.pathname,value});};const disableButtons=()=>{yes.disabled=true;yes.classList.add('feedback--button--disabled');no.disabled=true;no.classList.add('feedback--button--disabled');};yes.addEventListener('click',()=>{sendFeedback(1);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response--hidden');});no.addEventListener('click',()=>{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response--hidden');});</script> Last modified August 28, 2020 at 3:40 PM PST: Fix arguments for service account key pair (a2a175560) <i class="fa fa-edit fa-fw"/>Edit this page <i class="fa fa-edit fa-fw"/>Create child page <i class="fab fa-github fa-fw"/>Create an issue <nav

id="TableOfContents">
How certificates are used by your cluster
Where certificates are stored
Configure certificates manually
Single root CA
All certificates
Certificate paths
Configure certificates for user accounts
</nav>