

入门

本节介绍了设置和运行 Kubernetes 环境的不同选项。

不同的 Kubernetes 解决方案满足不同的要求：易于维护、安全性、可控制性、可用资源以及操作和管理 Kubernetes 集群所需的专业知识。

可以在本地机器、云、本地数据中心上部署 Kubernetes 集群，或选择一个托管的 Kubernetes 集群。还可以跨各种云提供商或裸机环境创建自定义解决方案。

更简单地说，可以在学习和生产环境中创建一个 Kubernetes 集群。

学习环境

如果正打算学习 Kubernetes，请使用基于 Docker 的解决方案：Docker 是 Kubernetes 社区支持或生态系统中用来在本地计算机上设置 Kubernetes 集群的一种工具。

生产环境

在评估生产环境的解决方案时，请考虑要管理自己 Kubernetes 集群（抽象层面）的哪些方面或将其转移给提供商。

[Kubernetes 合作伙伴](#) 包括一个 [已认证的 Kubernetes](#) 提供商列表。

[Kubernetes 发行说明和版本偏差](#)

[学习环境](#)

[生产环境](#)

[最佳实践](#)

Kubernetes 发行说明和版本偏差

[v1.18 发布说明](#)

[Kubernetes 版本及版本偏差支持策略](#)

v1.18 发布说明

v1.18.0

[Documentation](#)

Downloads for v1.18.0

filename	sha512 hash
kubernetes.tar.gz	cd5b86a3947a4f2cea6d857743ab2009be127d782b6f2eb4d37d88918a5e4
kubernetes-src.tar.gz	fb42cf133355ef18f67c8c4bb555aa1f284906c06e21fa41646e086d34ece77

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	26df342ef65745df12fa52931358e7f744111b6fe1e0bddb8c3c6598faf73a
kubernetes-client-darwin-amd64.tar.gz	803a0fed122ef6b85f7a120b5485723eaade765b7bc8306d0c0da03bd3df
kubernetes-client-linux-386.tar.gz	110844511b70f9f3ebb92c15105e6680a05a562cd83f79ce2d2e25c2dd70
kubernetes-client-linux-amd64.tar.gz	594ca3eadc7974ec4d9e4168453e36ca434812167ef8359086cd64d048df
kubernetes-client-linux-arm.tar.gz	d3627b763606557a6c9a5766c34198ec00b3a3cd72a55bc2cb47731060d
kubernetes-client-linux-arm64.tar.gz	ba9056eff1452cbdaef699efbf88f74f5309b3f7808d372ebf6918442d0c9fe
kubernetes-client-linux-ppc64le.tar.gz	f80fb3769358cb20820ff1a1ce9994de5ed194aabe6c73fb8b8048bffc394c
kubernetes-client-linux-s390x.tar.gz	a9b658108b6803d60fa3cd4e76d9e58bf75201017164fe54054b7ccadbb6
kubernetes-client-windows-386.tar.gz	18adffab5d1be146906fd8531f4eae7153576aac235150ce2da05aee5ae16
kubernetes-client-windows-amd64.tar.gz	162396256429cef07154f817de2a6b67635c770311f414e38b1e2db25961

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	a92f8d201973d5dfa44a398e95fcf6a7b4feeb1ef879ab3fee1c54370e21f59f725
kubernetes-server-linux-arm.tar.gz	62fbff3256bc0a83f70244b09149a8d7870d19c2c4b6dee8ca2714fc7388da340
kubernetes-server-linux-arm64.tar.gz	842910a7013f61a60d670079716b207705750d55a9e4f1f93696d19d39e19164
kubernetes-server-linux-ppc64le.tar.gz	95c5b952ac1c4127a5c3b519b664972ee1fb5e8e902551ce71c04e26ad44b39d
kubernetes-server-linux-s390x.tar.gz	a46522d2119a0fd58074564c1fa95dd8a929a79006b82ba3c4245611da8d2db9

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	f714f80feecb0756410f27efb4cf4a1b5232be0444fbec9f25cb85a7cccdcb5b
kubernetes-node-linux-arm.tar.gz	806000b5f6d723e24e2f12d19d1b9b3d16c74b855f51c7063284adf1fcc57a965
kubernetes-node-linux-arm64.tar.gz	c207e9ab60587d135897b5366af79efe9d2833f33401e469b2a4e0d74ecd2cf6b
kubernetes-node-linux-ppc64le.tar.gz	a542ed5ed02722af44ef12d1602f363fcd4e93cf704da2ea5d994463824856796
kubernetes-node-linux-s390x.tar.gz	651e0db73ee67869b2ae93cb0574168e4bd7918290fc5662a6b12b708fa62828
kubernetes-node-windows-amd64.tar.gz	d726ed904f9f7fe7e8831df621dc9094b87e767410a129aa675ee08417b662dde

Changelog since v1.17.0

A complete changelog for the release notes is now hosted in a customizable format at <https://relnotes.k8s.io>. Check it out and please give us your feedback!

What's New (Major Themes)

Kubernetes Topology Manager Moves to Beta - Align Up!

A beta feature of Kubernetes in release 1.18, the [Topology Manager feature](#) enables NUMA alignment of CPU and devices (such as SR-IOV VFs) that will allow your workload to run in an environment optimized for low-latency. Prior to the introduction of the Topology Manager, the CPU and Device Manager would make resource allocation decisions independent of each other. This could result in undesirable allocations on multi-socket systems, causing degraded performance on latency critical applications.

Serverside Apply - Beta 2

Server-side Apply was promoted to Beta in 1.16, but is now introducing a second Beta in 1.18. This new version will track and manage changes to fields of all new Kubernetes objects, allowing you to know what changed your resources and when.

Extending Ingress with and replacing a deprecated annotation with IngressClass

In Kubernetes 1.18, there are two significant additions to Ingress: A new `pathType` field and a new `IngressClass` resource. The `pathType` field allows specifying how paths should be matched. In addition to the default `ImplementationSpecific` type, there are new `Exact` and `Prefix` path types.

The `IngressClass` resource is used to describe a type of Ingress within a Kubernetes cluster. Ingresses can specify the class they are associated with by using a new `ingressClassName` field on Ingresses. This new resource and field replace the deprecated `kubernetes.io/ingress.class` annotation.

SIG CLI introduces kubect! debug

SIG CLI was debating the need for a debug utility for quite some time already. With the development of [ephemeral containers](#), it became more obvious how we can support developers with tooling built on top of `kubect! exec`. The addition of the `kubect! debug` [command](#) (it is alpha but your feedback is more than welcome), allows developers to easily debug their Pods inside the cluster. We think this addition is invaluable. This command allows one to create a temporary container which runs next to the Pod one is trying to examine, but also attaches to the console for interactive troubleshooting.

Introducing Windows CSI support alpha for Kubernetes

With the release of Kubernetes 1.18, an alpha version of CSI Proxy for Windows is getting released. CSI proxy enables non-privileged (pre-approved) containers to perform privileged storage operations on Windows.

CSI drivers can now be supported in Windows by leveraging CSI proxy. SIG Storage made a lot of progress in the 1.18 release. In particular, the following storage features are moving to GA in Kubernetes 1.18:

- Raw Block Support: Allow volumes to be surfaced as block devices inside containers instead of just mounted filesystems.
- Volume Cloning: Duplicate a PersistentVolumeClaim and underlying storage volume using the Kubernetes API via CSI.
- CSIDriver Kubernetes API Object: Simplifies CSI driver discovery and allows CSI Drivers to customize Kubernetes behavior.

SIG Storage is also introducing the following new storage features as alpha in Kubernetes 1.18:

- Windows CSI Support: Enabling containerized CSI node plugins in Windows via new [CSIProxy](#)
- Recursive Volume Ownership OnRootMismatch Option: Add a new "OnRootMismatch" policy that can help shorten the mount time for volumes that require ownership change and have many directories and files.

Other notable announcements

SIG Network is moving IPv6 to Beta in Kubernetes 1.18, after incrementing significantly the test coverage with new CI jobs.

NodeLocal DNSCache is an add-on that runs a dnsCache pod as a daemonset to improve clusterDNS performance and reliability. The feature has been in Alpha since 1.13 release. The SIG Network is announcing the GA graduation of Node Local DNSCache [#1351](#)

Known Issues

No Known Issues Reported

Urgent Upgrade Notes

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

kube-apiserver:

- in an `--encryption-provider-config` config file, an explicit `cacheSize: 0` parameter previously silently defaulted to caching 1000 keys. In Kubernetes 1.18, this now returns a config validation error. To disable caching, you can specify a negative `cacheSize` value in Kubernetes 1.18+.
- consumers of the 'certificatesigningrequests/approval' API must now have permission to 'approve' CSRs for the specific signer requested by the CSR. More information on the new `signerName` field and the required authorization can be found at <https://kubernetes.io/docs/>

[reference/access-authn-authz/certificate-signing-requests#authorization](#) (#88246, @munnerz) [SIG API Machinery, Apps, Auth, CLI, Node and Testing]

- The following features are unconditionally enabled and the corresponding --feature-gates flags have been removed: PodPriority, TaintNodesByCondition, ResourceQuotaScopeSelectors and ScheduleDaemonSetPods (#86210, @draveness) [SIG Apps and Scheduling]

kubelet:

- --enable-cadvisor-endpoints is now disabled by default. If you need access to the cAdvisor v1 Json API please enable it explicitly in the kubelet command line. Please note that this flag was deprecated in 1.15 and will be removed in 1.19. (#87440, @dims) [SIG Instrumentation, Node and Testing]
- Promote CSIMigrationOpenStack to Beta (off by default since it requires installation of the OpenStack Cinder CSI Driver. The in-tree AWS OpenStack Cinder driver "kubernetes.io/cinder" was deprecated in 1.16 and will be removed in 1.20. Users should enable CSIMigration + CSIMigrationOpenStack features and install the OpenStack Cinder CSI Driver (<https://github.com/kubernetes-sigs/cloud-provider-openstack>) to avoid disruption to existing Pod and PVC objects at that time. Users should start using the OpenStack Cinder CSI Driver directly for any new volumes. (#85637, @dims) [SIG Cloud Provider]

kubectl:

- kubectl and k8s.io/client-go no longer default to a server address of http://localhost:8080. If you own one of these legacy clusters, you are *strongly* encouraged to secure your server. If you cannot secure your server, you can set the \$KUBERNETES_MASTER environment variable to http://localhost:8080 to continue defaulting the server address. kubectl users can also set the server address using the --server flag, or in a kubeconfig file specified via --kubeconfig or \$KUBECONFIG. (#86173, @soltys) [SIG API Machinery, CLI and Testing]
- kubectl run has removed the previously deprecated generators, along with flags unrelated to creating pods. kubectl run now only creates pods. See specific kubectl create subcommands to create objects other than pods. (#87077, @soltys) [SIG Architecture, CLI and Testing]
- The deprecated command kubectl rolling-update has been removed (#88057, @julianvmodesto) [SIG Architecture, CLI and Testing]

client-go:

- Signatures on methods in generated clientsets, dynamic, metadata, and scale clients have been modified to accept context.Context as a first argument. Signatures of Create, Update, and Patch methods have been updated to accept CreateOptions, UpdateOptions and PatchOptions respectively. Signatures of Delete and DeleteCollection methods now

accept DeleteOptions by value instead of by reference. Generated clientsets with the previous interface have been added in new "deprecated" packages to allow incremental migration to the new APIs. The deprecated packages will be removed in the 1.21 release. A tool is available at <http://sigs.k8s.io/clientgofix> to rewrite method invocations to the new signatures.

- The following deprecated metrics are removed, please convert to the corresponding metrics:
 - The following replacement metrics are available from v1.14.0:
 - `rest_client_request_latency_seconds` -> `rest_client_request_duration_seconds`
 - `scheduler_scheduling_latency_seconds` -> `scheduler_scheduling_duration_seconds`
 - `docker_operations` -> `docker_operations_total`
 - `docker_operations_latency_microseconds` -> `docker_operations_duration_seconds`
 - `docker_operations_errors` -> `docker_operations_errors_total`
 - `docker_operations_timeout` -> `docker_operations_timeout_total`
 - `network_plugin_operations_latency_microseconds` -> `network_plugin_operations_duration_seconds`
 - `kubelet_pod_worker_latency_microseconds` -> `kubelet_pod_worker_duration_seconds`
 - `kubelet_pod_start_latency_microseconds` -> `kubelet_pod_start_duration_seconds`
 - `kubelet_cgroup_manager_latency_microseconds` -> `kubelet_cgroup_manager_duration_seconds`
 - `kubelet_pod_worker_start_latency_microseconds` -> `kubelet_pod_worker_start_duration_seconds`
 - `kubelet_pleg_relist_latency_microseconds` -> `kubelet_pleg_relist_duration_seconds`
 - `kubelet_pleg_relist_interval_microseconds` -> `kubelet_pleg_relist_interval_seconds`
 - `kubelet_eviction_stats_age_microseconds` -> `kubelet_eviction_stats_age_seconds`
 - `kubelet_runtime_operations` -> `kubelet_runtime_operations_total`
 - `kubelet_runtime_operations_latency_microseconds` -> `kubelet_runtime_operations_duration_seconds`
 - `kubelet_runtime_operations_errors` -> `kubelet_runtime_operations_errors_total`
 - `kubelet_device_plugin_registration_count` -> `kubelet_device_plugin_registration_total`
 - `kubelet_device_plugin_alloc_latency_microseconds` -> `kubelet_device_plugin_alloc_duration_seconds`
 - `scheduler_e2e_scheduling_latency_microseconds` -> `scheduler_e2e_scheduling_duration_seconds`

- scheduler_scheduling_algorithm_latency_microseconds -> scheduler_scheduling_algorithm_duration_seconds
- scheduler_scheduling_algorithm_predicate_evaluation -> scheduler_scheduling_algorithm_predicate_evaluation_seconds
- scheduler_scheduling_algorithm_priority_evaluation -> scheduler_scheduling_algorithm_priority_evaluation_seconds
- scheduler_scheduling_algorithm_preemption_evaluation -> scheduler_scheduling_algorithm_preemption_evaluation_seconds
- scheduler_binding_latency_microseconds -> scheduler_binding_duration_seconds
- kubeproxy_sync_proxy_rules_latency_microseconds -> kubeproxy_sync_proxy_rules_duration_seconds
- apiserver_request_latencies -> apiserver_request_duration_seconds
- apiserver_dropped_requests -> apiserver_dropped_requests_total
- etcd_request_latencies_summary -> etcd_request_duration_seconds
- apiserver_storage_transformation_latencies_microseconds -> apiserver_storage_transformation_duration_seconds
- apiserver_storage_data_key_generation_latencies_microseconds -> apiserver_storage_data_key_generation_duration_seconds
- apiserver_request_count -> apiserver_request_total
- apiserver_request_latencies_summary
- The following replacement metrics are available from v1.15.0:
 - apiserver_storage_transformation_failures_total -> apiserver_storage_transformation_operations_total ([#76496](#), [@danielqsj](#)) [SIG API Machinery, Cluster Lifecycle, Instrumentation, Network, Node and Scheduling]

Changes by Kind

Deprecation

kube-apiserver:

- the following deprecated APIs can no longer be served:
 - All resources under apps/v1beta1 and apps/v1beta2 - use apps/v1 instead
 - daemonsets, deployments, replicaset resources under extensions/v1beta1 - use apps/v1 instead
 - networkpolicies resources under extensions/v1beta1 - use networking.k8s.io/v1 instead
 - podsecuritypolicies resources under extensions/v1beta1 - use policy/v1beta1 instead ([#85903](#), [@liggitt](#)) [SIG API Machinery, Apps, Cluster Lifecycle, Instrumentation and Testing]

kube-controller-manager:

- Azure service annotation `service.beta.kubernetes.io/azure-load-balancer-disable-tcp-reset` has been deprecated. Its support would be removed in a future release. ([#88462](#), [@feiskyer](#)) [SIG Cloud Provider]

kubelet:

- The `StreamingProxyRedirects` feature and `--redirect-container-streaming` flag are deprecated, and will be removed in a future release. The default behavior (proxy streaming requests through the kubelet) will be the only supported option. If you are setting `--redirect-container-streaming=true`, then you must migrate off this configuration. The flag will no longer be able to be enabled starting in v1.20. If you are not setting the flag, no action is necessary. ([#88290](#), [@tallclair](#)) [SIG API Machinery and Node]
- resource metrics endpoint `/metrics/resource/v1alpha1` as well as all metrics under this endpoint have been deprecated. Please convert to the following metrics emitted by endpoint `/metrics/resource`:
- `scrape_error` --> `scrape_error`
- `node_cpu_usage_seconds_total` --> `node_cpu_usage_seconds`
- `node_memory_working_set_bytes` --> `node_memory_working_set_bytes`
- `container_cpu_usage_seconds_total` --> `container_cpu_usage_seconds`
- `container_memory_working_set_bytes` --> `container_memory_working_set_bytes`
- `scrape_error` --> `scrape_error`
([#86282](#), [@RainbowMango](#)) [SIG Node]
- In a future release, kubelet will no longer create the CSI `NodePublishVolume` target directory, in accordance with the CSI specification. CSI drivers may need to be updated accordingly to properly create and process the target path. ([#75535](#)) [SIG Storage]

kube-proxy:

- `--healthz-port` and `--metrics-port` flags are deprecated, please use `--healthz-bind-address` and `--metrics-bind-address` instead ([#88512](#), [@SataQiu](#)) [SIG Network]
- a new `EndpointSliceProxying` feature gate has been added to control the use of `EndpointSlices` in kube-proxy. The `EndpointSlice` feature gate that used to control this behavior no longer affects kube-proxy. This feature has been disabled by default. ([#86137](#), [@roboscott](#))

kubeadm:

- command line option `"kubelet-version"` for `kubeadm upgrade node` has been deprecated and will be removed in a future release. ([#87942](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
- deprecate the usage of the experimental flag `'--use-api'` under the `'kubeadm alpha certs renew'` command. ([#88827](#), [@neolit123](#)) [SIG Cluster Lifecycle]

- kube-dns is deprecated and will not be supported in a future version ([#86574](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
- the ClusterStatus struct present in the kubeadm-config ConfigMap is deprecated and will be removed in a future version. It is going to be maintained by kubeadm until it gets removed. The same information can be found on etcd and kube-apiserver pod annotations, kubeadm.kubernetes.io/etcd.advertise-client-urls and kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint respectively. ([#87656](#), [@ereslibre](#)) [SIG Cluster Lifecycle]

kubectl:

- the boolean and unset values for the --dry-run flag are deprecated and a value --dry-run=server|client|none will be required in a future version. ([#87580](#), [@julianvmodesto](#)) [SIG CLI]
- kubectl apply --server-dry-run is deprecated and replaced with --dry-run=server ([#87580](#), [@julianvmodesto](#)) [SIG CLI]

add-ons:

- Remove cluster-monitoring addon ([#85512](#), [@serathius](#)) [SIG Cluster Lifecycle, Instrumentation, Scalability and Testing]

kube-scheduler:

- The scheduling_duration_seconds summary metric is deprecated ([#86586](#), [@xiaoanyunfei](#)) [SIG Scheduling]
- The scheduling_algorithm_predicate_evaluation_seconds and scheduling_algorithm_priority_evaluation_seconds metrics are deprecated, replaced by framework_extension_point_duration_seconds[extension_point="Filter"] and framework_extension_point_duration_seconds[extension_point="Score"]. ([#86584](#), [@xiaoanyunfei](#)) [SIG Scheduling]
- AlwaysCheckAllPredicates is deprecated in scheduler Policy API. ([#86369](#), [@Huang-Wei](#)) [SIG Scheduling]

Other deprecations:

- The k8s.io/node-api component is no longer updated. Instead, use the RuntimeClass types located within k8s.io/api, and the generated clients located within k8s.io/client-go ([#87503](#), [@liggitt](#)) [SIG Node and Release]
- Removed the 'client' label from apiserver_request_total. ([#87669](#), [@logicalhan](#)) [SIG API Machinery and Instrumentation]

API Change

New API types/versions:

- A new IngressClass resource has been added to enable better Ingress configuration. ([#88509](#), [@robscott](#)) [SIG API Machinery, Apps, CLI, Network, Node and Testing]
- The CSIDriver API has graduated to storage.k8s.io/v1, and is now available for use. ([#84814](#), [@huffmanca](#)) [SIG Storage]

New API fields:

- autoscaling/v2beta2 HorizontalPodAutoscaler added a spec.behavior field that allows scale behavior to be configured. Behaviors are specified separately for scaling up and down. In each direction a stabilization window can be specified as well as a list of policies and how to select amongst them. Policies can limit the absolute number of pods added or removed, or the percentage of pods added or removed. ([#74525](#), [@gliush](#)) [SIG API Machinery, Apps, Autoscaling and CLI]
- Ingress:
 - spec.ingressClassName replaces the deprecated kubernetes.io/ingress.class annotation, and allows associating an Ingress object with a particular controller.
 - path definitions added a pathType field to allow indicating how the specified path should be matched against incoming requests. Valid values are Exact, Prefix, and ImplementationSpecific ([#88587](#), [@cmluciano](#)) [SIG Apps, Cluster Lifecycle and Network]
- The alpha feature AnyVolumeDataSource enables PersistentVolumeClaim objects to use the spec.dataSource field to reference a custom type as a data source ([#88636](#), [@bswartz](#)) [SIG Apps and Storage]
- The alpha feature ConfigurableFSGroupPolicy enables v1 Pods to specify a spec.securityContext.fsGroupChangePolicy policy to control how file permissions are applied to volumes mounted into the pod. ([#88488](#), [@gnufied](#)) [SIG Storage]
- The alpha feature ServiceAppProtocol enables setting an appProtocol field in ServicePort and EndpointPort definitions. ([#88503](#), [@robscott](#)) [SIG Apps and Network]
- The alpha feature ImmutableEphemeralVolumes enables an immutable field in both Secret and ConfigMap objects to mark their contents as immutable. ([#86377](#), [@wojtek-t](#)) [SIG Apps, CLI and Testing]

Other API changes:

- The beta feature ServerSideApply enables tracking and managing changed fields for all new objects, which means there will be managedFields in metadata with the list of managers and their owned fields.
- The alpha feature ServiceAccountIssuerDiscovery enables publishing OIDC discovery information and service account token verification keys at /.well-known/openid-configuration and /openid/v1/jwks endpoints by

- API servers configured to issue service account tokens. ([#80724](#), [@cceckman](#)) [SIG API Machinery, Auth, Cluster Lifecycle and Testing]
- CustomResourceDefinition schemas that use x-kubernetes-list-map-keys to specify properties that uniquely identify list items must make those properties required or have a default value, to ensure those properties are present for all list items. See <https://kubernetes.io/docs/reference/using-api/api-concepts/#merge-strategy> for details. ([#88076](#), [@eloyekunle](#)) [SIG API Machinery and Testing]
 - CustomResourceDefinition schemas that use x-kubernetes-list-type: map or x-kubernetes-list-type: set now enable validation that the list items in the corresponding custom resources are unique. ([#84920](#), [@sttts](#)) [SIG API Machinery]

Configuration file changes:

kube-apiserver:

- The `--egress-selector-config-file` configuration file now accepts an `apiserver.k8s.io/v1beta1 EgressSelectorConfiguration` configuration object, and has been updated to allow specifying HTTP or GRPC connections to the network proxy ([#87179](#), [@Jefftree](#)) [SIG API Machinery, Cloud Provider and Cluster Lifecycle]

kube-scheduler:

- A `kubescheduler.config.k8s.io/v1alpha2` configuration file version is now accepted, with support for multiple scheduling profiles ([#87628](#), [@alculquicondor](#)) [SIG Scheduling]
 - `HardPodAffinityWeight` moved from a top level `ComponentConfig` parameter to a `PluginConfig` parameter of `InterPodAffinity` Plugin in `kubescheduler.config.k8s.io/v1alpha2` ([#88002](#), [@alculquicondor](#)) [SIG Scheduling and Testing]
 - Kube-scheduler can run more than one scheduling profile. Given a pod, the profile is selected by using its `.spec.schedulerName`. ([#88285](#), [@alculquicondor](#)) [SIG Apps, Scheduling and Testing]
 - Scheduler Extenders can now be configured in the `v1alpha2` component config ([#88768](#), [@damemi](#)) [SIG Release, Scheduling and Testing]
 - The `PostFilter` of scheduler framework is renamed to `PreScore` in `kubescheduler.config.k8s.io/v1alpha2`. ([#87751](#), [@skilxn-go](#)) [SIG Scheduling and Testing]

kube-proxy:

- Added kube-proxy flags `--ipvs-tcp-timeout`, `--ipvs-tcpfin-timeout`, `--ipvs-udp-timeout` to configure IPVS connection timeouts. ([#85517](#), [@andrewsykim](#)) [SIG Cluster Lifecycle and Network]

- Added optional `--detect-local-mode` flag to kube-proxy. Valid values are "ClusterCIDR" (default matching previous behavior) and "NodeCIDR" ([#87748](#), [@satyasm](#)) [SIG Cluster Lifecycle, Network and Scheduling]
- Kube-controller-manager and kube-scheduler expose profiling by default to match the kube-apiserver. Use `--enable-profiling=false` to disable. ([#88663](#), [@deads2k](#)) [SIG API Machinery, Cloud Provider and Scheduling]
- Kubelet pod resources API now provides the information about active pods only. ([#79409](#), [@takmatsu](#)) [SIG Node]
- New flag `--endpointslice-updates-batch-period` in kube-controller-manager can be used to reduce the number of endpointslice updates generated by pod changes. ([#88745](#), [@mborsz](#)) [SIG API Machinery, Apps and Network]
- New flag `--show-hidden-metrics-for-version` in kube-proxy, kubelet, kube-controller-manager, and kube-scheduler can be used to show all hidden metrics that are deprecated in the previous minor release. ([#85279](#), [@RainbowMango](#)) [SIG Cluster Lifecycle and Network]

Features graduated to beta:

- StartupProbe ([#83437](#), [@matthyx](#)) [SIG Node, Scalability and Testing]

Features graduated to GA:

- VolumePVCDataSource ([#88686](#), [@j-griffith](#)) [SIG Storage]
- TaintBasedEvictions ([#87487](#), [@skilxn-go](#)) [SIG API Machinery, Apps, Node, Scheduling and Testing]
- BlockVolume and CSIBlockVolume ([#88673](#), [@jsafrane](#)) [SIG Storage]
- Windows RunAsUserName ([#87790](#), [@marosset](#)) [SIG Apps and Windows]
- The following feature gates are removed, because the associated features were unconditionally enabled in previous releases: CustomResourceValidation, CustomResourceSubresources, CustomResourceWebhookConversion, CustomResourcePublishOpenAPI, CustomResourceDefaulting ([#87475](#), [@liggitt](#)) [SIG API Machinery]

Feature

- API request throttling (due to a high rate of requests) is now reported in client-go logs at log level 2. The messages are of the form: Throttling request took 1.50705208s, request: GET:<URL> The presence of these messages may indicate to the administrator the need to tune the cluster accordingly. ([#87740](#), [@jennybuckley](#)) [SIG API Machinery]
- Add support for mount options to the FC volume plugin ([#87499](#), [@ejweber](#)) [SIG Storage]
- Added a config-mode flag in azure auth module to enable getting AAD token without spn: prefix in audience claim. When it's not specified, the default behavior doesn't change. ([#87630](#), [@weinong](#)) [SIG API Machinery, Auth, CLI and Cloud Provider]

- Allow for configuration of CoreDNS replica count ([#85837](#), [@pickledrick](#)) [SIG Cluster Lifecycle]
- Allow user to specify resource using --filename flag when invoking kubectl exec ([#88460](#), [@soltys](#)) [SIG CLI and Testing]
- Apiserver added a new flag --goaway-chance which is the fraction of requests that will be closed gracefully(GOAWAY) to prevent HTTP/2 clients from getting stuck on a single apiserver. ([#88567](#), [@answer1991](#)) [SIG API Machinery]
- Azure Cloud Provider now supports using Azure network resources (Virtual Network, Load Balancer, Public IP, Route Table, Network Security Group, etc.) in different AAD Tenant and Subscription than those for the Kubernetes cluster. To use the feature, please reference <https://github.com/kubernetes-sigs/cloud-provider-azure/blob/master/docs/cloud-provider-config.md#35;host-network-resources-in-different-aad-tenant-and-subscription>. ([#88384](#), [@bowen5](#)) [SIG Cloud Provider]
- Azure VMSS/VMSSVM clients now suppress requests on throttling ([#86740](#), [@feiskyer](#)) [SIG Cloud Provider]
- Azure cloud provider cache TTL is configurable, list of the azure cloud provider is as following:
 - "availabilitySetNodesCacheTTLInSeconds"
 - "vmssCacheTTLInSeconds"
 - "vmssVirtualMachinesCacheTTLInSeconds"
 - "vmCacheTTLInSeconds"
 - "loadBalancerCacheTTLInSeconds"
 - "nsgCacheTTLInSeconds"
 - "routeTableCacheTTLInSeconds" ([#86266](#), [@zqingqing1](#)) [SIG Cloud Provider]
- Azure global rate limit is switched to per-client. A set of new rate limit configure options are introduced, including routeRateLimit, SubnetsRateLimit, InterfaceRateLimit, RouteTableRateLimit, LoadBalancerRateLimit, PublicIPAddressRateLimit, SecurityGroupRateLimit, VirtualMachineRateLimit, StorageAccountRateLimit, DiskRateLimit, SnapshotRateLimit, VirtualMachineScaleSetRateLimit and VirtualMachineSizeRateLimit. The original rate limit options would be default values for those new client's rate limiter. ([#86515](#), [@feiskyer](#)) [SIG Cloud Provider]
- Azure network and VM clients now suppress requests on throttling ([#87122](#), [@feiskyer](#)) [SIG Cloud Provider]
- Azure storage clients now suppress requests on throttling ([#87306](#), [@feiskyer](#)) [SIG Cloud Provider]
- Azure: add support for single stack IPv6 ([#88448](#), [@aramase](#)) [SIG Cloud Provider]
- DefaultConstraints can be specified for PodTopologySpread Plugin in the scheduler's ComponentConfig ([#88671](#), [@alculquicondor](#)) [SIG Scheduling]
- DisableAvailabilitySetNodes is added to avoid VM list for VMSS clusters. It should only be used when vmType is "vmss" and all the nodes (including control plane nodes) are VMSS virtual machines. ([#87685](#), [@feiskyer](#)) [SIG Cloud Provider]
- Elasticsearch supports automatically setting the advertise address ([#85944](#), [@SataQiu](#)) [SIG Cluster Lifecycle and Instrumentation]

- EndpointSlices will now be enabled by default. A new EndpointSliceProxying feature gate determines if kube-proxy will use EndpointSlices, this is disabled by default. ([#86137](#), [@roboscott](#)) [SIG Network]
- Kube-proxy: Added dual-stack IPv4/IPv6 support to the iptables proxier. ([#82462](#), [@vllry](#)) [SIG Network]
- Kubeadm now supports automatic calculations of dual-stack node cidr masks to kube-controller-manager. ([#85609](#), [@Arvinderpal](#)) [SIG Cluster Lifecycle]
- Kubeadm: add a upgrade health check that deploys a Job ([#81319](#), [@neolit123](#)) [SIG Cluster Lifecycle]
- Kubeadm: add the experimental feature gate PublicKeysECDSA that can be used to create a cluster with ECDSA certificates from "kubeadm init". Renewal of existing ECDSA certificates is also supported using "kubeadm alpha certs renew", but not switching between the RSA and ECDSA algorithms on the fly or during upgrades. ([#86953](#), [@rojkov](#)) [SIG API Machinery, Auth and Cluster Lifecycle]
- Kubeadm: implemented structured output of 'kubeadm config images list' command in JSON, YAML, Go template and JsonPath formats ([#86810](#), [@bart0sh](#)) [SIG Cluster Lifecycle]
- Kubeadm: on kubeconfig certificate renewal, keep the embedded CA in sync with the one on disk ([#88052](#), [@neolit123](#)) [SIG Cluster Lifecycle]
- Kubeadm: reject a node joining the cluster if a node with the same name already exists ([#81056](#), [@neolit123](#)) [SIG Cluster Lifecycle]
- Kubeadm: support Windows specific kubelet flags in kubeadm-flags.env ([#88287](#), [@gab-satchi](#)) [SIG Cluster Lifecycle and Windows]
- Kubeadm: support automatic retry after failing to pull image ([#86899](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
- Kubeadm: upgrade supports fallback to the nearest known etcd version if an unknown k8s version is passed ([#88373](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
- Kubectl/drain: add disable-eviction option. Force drain to use delete, even if eviction is supported. This will bypass checking PodDisruptionBudgets, and should be used with caution. ([#85571](#), [@michaelgugino](#)) [SIG CLI]
- Kubectl/drain: add skip-wait-for-delete-timeout option. If a pod's DeletionTimestamp is older than N seconds, skip waiting for the pod. Seconds must be greater than 0 to skip. ([#85577](#), [@michaelgugino](#)) [SIG CLI]
- Option preConfiguredBackendPoolLoadBalancerTypes is added to azure cloud provider for the pre-configured load balancers, possible values: "", "internal", "external", "all" ([#86338](#), [@gossion](#)) [SIG Cloud Provider]
- PodTopologySpread plugin now excludes terminatingPods when making scheduling decisions. ([#87845](#), [@Huang-Wei](#)) [SIG Scheduling]
- Provider/azure: Network security groups can now be in a separate resource group. ([#87035](#), [@CecileRobertMichon](#)) [SIG Cloud Provider]
- SafeSysctlWhitelist: add net.ipv4.ping_group_range ([#85463](#), [@AkihiroSuda](#)) [SIG Auth]
- Scheduler framework permit plugins now run at the end of the scheduling cycle, after reserve plugins. Waiting on permit will remain in the beginning of the binding cycle. ([#88199](#), [@mateuszlitwin](#)) [SIG Scheduling]

- Scheduler: Add DefaultBinder plugin ([#87430](#), [@alculquicondor](#)) [SIG Scheduling and Testing]
- Skip default spreading scoring plugin for pods that define TopologySpreadConstraints ([#87566](#), [@skilxn-go](#)) [SIG Scheduling]
- The kubectl --dry-run flag now accepts the values 'client', 'server', and 'none', to support client-side and server-side dry-run strategies. The boolean and unset values for the --dry-run flag are deprecated and a value will be required in a future version. ([#87580](#), [@julianvmodesto](#)) [SIG CLI]
- Support server-side dry-run in kubectl with --dry-run=server for commands including apply, patch, create, run, annotate, label, set, autoscale, drain, rollout undo, and expose. ([#87714](#), [@julianvmodesto](#)) [SIG API Machinery, CLI and Testing]
- Add --dry-run=server|client to kubectl delete, taint, replace ([#88292](#), [@julianvmodesto](#)) [SIG CLI and Testing]
- The feature PodTopologySpread (feature gate EvenPodsSpread) has been enabled by default in 1.18. ([#88105](#), [@Huang-Wei](#)) [SIG Scheduling and Testing]
- The kubelet and the default docker runtime now support running ephemeral containers in the Linux process namespace of a target container. Other container runtimes must implement support for this feature before it will be available for that runtime. ([#84731](#), [@verb](#)) [SIG Node]
- The underlying format of the CPMManager state file has changed. Upgrades should be seamless, but any third-party tools that rely on reading the previous format need to be updated. ([#84462](#), [@klueska](#)) [SIG Node and Testing]
- Update CNI version to v0.8.5 ([#78819](#), [@justaugustus](#)) [SIG API Machinery, Cluster Lifecycle, Network, Release and Testing]
- Webhooks have alpha support for network proxy ([#85870](#), [@Jefftree](#)) [SIG API Machinery, Auth and Testing]
- When client certificate files are provided, reload files for new connections, and close connections when a certificate changes. ([#79083](#), [@jackkleeman](#)) [SIG API Machinery, Auth, Node and Testing]
- When deleting objects using kubectl with the --force flag, you are no longer required to also specify --grace-period=0. ([#87776](#), [@brianpursley](#)) [SIG CLI]
- Windows nodes on GCE can use virtual TPM-based authentication to the control plane. ([#85466](#), [@pjh](#)) [SIG Cluster Lifecycle]
- You can now pass "--node-ip ::" to kubelet to indicate that it should autodetect an IPv6 address to use as the node's primary address. ([#85850](#), [@danwinship](#)) [SIG Cloud Provider, Network and Node]
- kubectl now contains a kubectl alpha debug command. This command allows attaching an ephemeral container to a running pod for the purposes of debugging. ([#88004](#), [@verb](#)) [SIG CLI]
- TLS Server Name overrides can now be specified in a kubeconfig file and via --tls-server-name in kubectl ([#88769](#), [@deads2k](#)) [SIG API Machinery, Auth and CLI]

Metrics:

- Add `rest_client_rate_limiter_duration_seconds` metric to component-base to track client side rate limiter latency in seconds. Broken down by verb and URL. ([#88134](#), [@jennybuckley](#)) [SIG API Machinery, Cluster Lifecycle and Instrumentation]
- Added two client certificate metrics for exec auth:
 - `rest_client_certificate_expiration_seconds` a gauge reporting the lifetime of the current client certificate. Reports the time of expiry in seconds since January 1, 1970 UTC.
 - `rest_client_certificate_rotation_age` a histogram reporting the age of a just rotated client certificate in seconds. ([#84382](#), [@sambdavidson](#)) [SIG API Machinery, Auth, Cluster Lifecycle and Instrumentation]
- Controller manager serve workqueue metrics ([#87967](#), [@zhan849](#)) [SIG API Machinery]
- Following metrics have been turned off:
 - `kubelet_pod_worker_latency_microseconds`
 - `kubelet_pod_start_latency_microseconds`
 - `kubelet_cgroup_manager_latency_microseconds`
 - `kubelet_pod_worker_start_latency_microseconds`
 - `kubelet_pleg_relist_latency_microseconds`
 - `kubelet_pleg_relist_interval_microseconds`
 - `kubelet_eviction_stats_age_microseconds`
 - `kubelet_runtime_operations`
 - `kubelet_runtime_operations_latency_microseconds`
 - `kubelet_runtime_operations_errors`
 - `kubelet_device_plugin_registration_count`
 - `kubelet_device_plugin_alloc_latency_microseconds`
 - `kubelet_docker_operations`
 - `kubelet_docker_operations_latency_microseconds`
 - `kubelet_docker_operations_errors`
 - `kubelet_docker_operations_timeout`
 - `network_plugin_operations_latency_microseconds` ([#83841](#), [@RainbowMango](#)) [SIG Network and Node]
- Kube-apiserver metrics will now include request counts, latencies, and response sizes for `/healthz`, `/livez`, and `/readyz` requests. ([#83598](#), [@jktomer](#)) [SIG API Machinery]
- Kubelet now exports a `server_expiration_renew_failure` and `client_expiration_renew_failure` metric counter if the certificate rotations cannot be performed. ([#84614](#), [@rphillips](#)) [SIG API Machinery, Auth, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation, Node and Release]
- Kubelet: the metric `process_start_time_seconds` be marked as with the ALPHA stability level. ([#85446](#), [@RainbowMango](#)) [SIG API Machinery, Cluster Lifecycle, Instrumentation and Node]
- New metric `kubelet_pleg_last_seen_seconds` to aid diagnosis of PLEG not healthy issues. ([#86251](#), [@bboreham](#)) [SIG Node]

Other (Bug, Cleanup or Flake)

- Fixed a regression with clients prior to 1.15 not being able to update podIP in pod status, or podCIDR in node spec, against ≥ 1.16 API servers ([#88505](#), [@liggitt](#)) [SIG Apps and Network]
- Fixed "kubectl describe statefulsets.apps" printing garbage for rolling update partition ([#85846](#), [@phil9909](#)) [SIG CLI]
- Add a event to PV when filesystem on PV does not match actual filesystem on disk ([#86982](#), [@gnuified](#)) [SIG Storage]
- Add azure disk WriteAccelerator support ([#87945](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Add delays between goroutines for vm instance update ([#88094](#), [@aramase](#)) [SIG Cloud Provider]
- Add init containers log to cluster dump info. ([#88324](#), [@zhouya0](#)) [SIG CLI]
- Addons: elasticsearch discovery supports IPv6 ([#85543](#), [@SataQiu](#)) [SIG Cluster Lifecycle and Instrumentation]
- Adds "volume.beta.kubernetes.io/migrated-to" annotation to PV's and PVC's when they are migrated to signal external provisioners to pick up those objects for Provisioning and Deleting. ([#87098](#), [@davidz627](#)) [SIG Storage]
- All api-server log request lines in a more greppable format. ([#87203](#), [@lavalamp](#)) [SIG API Machinery]
- Azure VMSS LoadBalancerBackendAddressPools updating has been improved with sequential-sync + concurrent-async requests. ([#88699](#), [@feiskyer](#)) [SIG Cloud Provider]
- Azure cloud provider now obtains AAD token who audience claim will not have spn: prefix ([#87590](#), [@weinong](#)) [SIG Cloud Provider]
- AzureFile and CephFS use the new Mount library that prevents logging of sensitive mount options. ([#88684](#), [@saad-ali](#)) [SIG Storage]
- Bind dns-horizontal containers to linux nodes to avoid Windows scheduling on kubernetes cluster includes linux nodes and windows nodes ([#83364](#), [@wawa0210](#)) [SIG Cluster Lifecycle and Windows]
- Bind kube-dns containers to linux nodes to avoid Windows scheduling ([#83358](#), [@wawa0210](#)) [SIG Cluster Lifecycle and Windows]
- Bind metadata-agent containers to linux nodes to avoid Windows scheduling on kubernetes cluster includes linux nodes and windows nodes ([#83363](#), [@wawa0210](#)) [SIG Cluster Lifecycle, Instrumentation and Windows]
- Bind metrics-server containers to linux nodes to avoid Windows scheduling on kubernetes cluster includes linux nodes and windows nodes ([#83362](#), [@wawa0210](#)) [SIG Cluster Lifecycle, Instrumentation and Windows]
- Bug fixes: Make sure we include latest packages node #351 (@caseydavenport) ([#84163](#), [@david-tigera](#)) [SIG Cluster Lifecycle]
- CPU limits are now respected for Windows containers. If a node is over-provisioned, no weighting is used, only limits are respected. ([#86101](#), [@PatrickLang](#)) [SIG Node, Testing and Windows]
- Changed core_pattern on COS nodes to be an absolute path. ([#86329](#), [@mml](#)) [SIG Cluster Lifecycle and Node]

- Client-go certificate manager rotation gained the ability to preserve optional intermediate chains accompanying issued certificates ([#88744](#), [@jackkleeman](#)) [SIG API Machinery and Auth]
- Cloud provider config CloudProviderBackoffMode has been removed since it won't be used anymore. ([#88463](#), [@feiskyer](#)) [SIG Cloud Provider]
- Conformance image now depends on stretch-slim instead of debian-hyperkube-base as that image is being deprecated and removed. ([#88702](#), [@dims](#)) [SIG Cluster Lifecycle, Release and Testing]
- Deprecate --generator flag from kubectl create commands ([#88655](#), [@soltys](#)) [SIG CLI]
- During initialization phase (preflight), kubeadm now verifies the presence of the conntrack executable ([#85857](#), [@hnanni](#)) [SIG Cluster Lifecycle]
- EndpointSlice should not contain endpoints for terminating pods ([#89056](#), [@andrewsykim](#)) [SIG Apps and Network]
- Evictions due to pods breaching their ephemeral storage limits are now recorded by the kubelet_evictions metric and can be alerted on. ([#87906](#), [@smarterclayton](#)) [SIG Node]
- Filter published OpenAPI schema by making nullable, required fields non-required in order to avoid kubectl to wrongly reject null values. ([#85722](#), [@sttts](#)) [SIG API Machinery]
- Fix /readyz to return error immediately after a shutdown is initiated, before the --shutdown-delay-duration has elapsed. ([#88911](#), [@tkashem](#)) [SIG API Machinery]
- Fix API Server potential memory leak issue in processing watch request. ([#85410](#), [@answer1991](#)) [SIG API Machinery]
- Fix EndpointSlice controller race condition and ensure that it handles external changes to EndpointSlices. ([#85703](#), [@robscott](#)) [SIG Apps and Network]
- Fix IPv6 addresses lost issue in pure ipv6 vsphere environment ([#86001](#), [@hubv](#)) [SIG Cloud Provider]
- Fix LoadBalancer rule checking so that no unexpected LoadBalancer updates are made ([#85990](#), [@feiskyer](#)) [SIG Cloud Provider]
- Fix a bug in kube-proxy that caused it to crash when using load balancers with a different IP family ([#87117](#), [@aojea](#)) [SIG Network]
- Fix a bug in port-forward: named port not working with service ([#85511](#), [@oke-py](#)) [SIG CLI]
- Fix a bug in the dual-stack IPVS proxier where stale IPv6 endpoints were not being cleaned up ([#87695](#), [@andrewsykim](#)) [SIG Network]
- Fix a bug that orphan revision cannot be adopted and statefulset cannot be synced ([#86801](#), [@likakuli](#)) [SIG Apps]
- Fix a bug where ExternalTrafficPolicy is not applied to service ExternalIPs. ([#88786](#), [@freehan](#)) [SIG Network]
- Fix a bug where kubenet fails to parse the tc output. ([#83572](#), [@chendojts](#)) [SIG Network]
- Fix a regression in kubenet that prevent pods to obtain ip addresses ([#85993](#), [@chendojts](#)) [SIG Network and Node]
- Fix azure file AuthorizationFailure ([#85475](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fix bug where EndpointSlice controller would attempt to modify shared objects. ([#85368](#), [@robscott](#)) [SIG API Machinery, Apps and Network]

- Fix handling of aws-load-balancer-security-groups annotation. Security-Groups assigned with this annotation are no longer modified by kubernetes which is the expected behaviour of most users. Also no unnecessary Security-Groups are created anymore if this annotation is used. ([#83446](#), [@Elias481](#)) [SIG Cloud Provider]
- Fix invalid VMSS updates due to incorrect cache ([#89002](#), [@ArchangelSDY](#)) [SIG Cloud Provider]
- Fix isCurrentInstance for Windows by removing the dependency of hostname. ([#89138](#), [@feiskyer](#)) [SIG Cloud Provider]
- Fix issue #85805 about a resource not found in azure cloud provider when LoadBalancer specified in another resource group. ([#86502](#), [@levimm](#)) [SIG Cloud Provider]
- Fix kubectl annotate error when local=true is set ([#86952](#), [@zhouya0](#)) [SIG CLI]
- Fix kubectl create deployment image name ([#86636](#), [@zhouya0](#)) [SIG CLI]
- Fix kubectl drain ignore daemonsets and others. ([#87361](#), [@zhouya0](#)) [SIG CLI]
- Fix missing "apiVersion" for "involvedObject" in Events for Nodes. ([#87537](#), [@uthark](#)) [SIG Apps and Node]
- Fix nil pointer dereference in azure cloud provider ([#85975](#), [@ldx](#)) [SIG Cloud Provider]
- Fix regression in statefulset conversion which prevents applying a statefulset multiple times. ([#87706](#), [@liggitt](#)) [SIG Apps and Testing]
- Fix route conflicted operations when updating multiple routes together ([#88209](#), [@feiskyer](#)) [SIG Cloud Provider]
- Fix that prevents repeated fetching of PVC/PV objects by kubelet when processing of pod volumes fails. While this prevents hammering API server in these error scenarios, it means that some errors in processing volume(s) for a pod could now take up to 2-3 minutes before retry. ([#88141](#), [@tedyu](#)) [SIG Node and Storage]
- Fix the bug PIP's DNS is deleted if no DNS label service annotation isn't set. ([#87246](#), [@nilo19](#)) [SIG Cloud Provider]
- Fix control plane hosts rolling upgrade causing thundering herd of LISTs on etcd leading to control plane unavailability. ([#86430](#), [@wojtek-t](#)) [SIG API Machinery, Node and Testing]
- Fix: add azure disk migration support for CSINode ([#88014](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fix: add non-retriable errors in azure clients ([#87941](#), [@andyzhangx](#)) [SIG Cloud Provider]
- Fix: add remediation in azure disk attach/detach ([#88444](#), [@andyzhangx](#)) [SIG Cloud Provider]
- Fix: azure data disk should use same key as os disk by default ([#86351](#), [@andyzhangx](#)) [SIG Cloud Provider]
- Fix: azure disk could not mounted on Standard_DC4s/DC2s instances ([#86612](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fix: azure file mount timeout issue ([#88610](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fix: check disk status before disk azure disk ([#88360](#), [@andyzhangx](#)) [SIG Cloud Provider]
- Fix: corrupted mount point in csi driver ([#88569](#), [@andyzhangx](#)) [SIG Storage]

- Fix: get azure disk lun timeout issue ([#88158](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fix: update azure disk max count ([#88201](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fixed "requested device X but found Y" attach error on AWS. ([#85675](#), [@jsafrane](#)) [SIG Cloud Provider and Storage]
- Fixed NetworkPolicy validation that Except values are accepted when they are outside the CIDR range. ([#86578](#), [@tnqn](#)) [SIG Network]
- Fixed a bug in the TopologyManager. Previously, the TopologyManager would only guarantee alignment if container creation was serialized in some way. Alignment is now guaranteed under all scenarios of container creation. ([#87759](#), [@klueska](#)) [SIG Node]
- Fixed a bug which could prevent a provider ID from ever being set for node if an error occurred determining the provider ID when the node was added. ([#87043](#), [@zjs](#)) [SIG Apps and Cloud Provider]
- Fixed a data race in the kubelet image manager that can cause static pod workers to silently stop working. ([#88915](#), [@roycaiwh](#)) [SIG Node]
- Fixed a panic in the kubelet cleaning up pod volumes ([#86277](#), [@tedyu](#)) [SIG Storage]
- Fixed a regression where the kubelet would fail to update the ready status of pods. ([#84951](#), [@tedyu](#)) [SIG Node]
- Fixed an issue that could cause the kubelet to incorrectly run concurrent pod reconciliation loops and crash. ([#89055](#), [@tedyu](#)) [SIG Node]
- Fixed block CSI volume cleanup after timeouts. ([#88660](#), [@jsafrane](#)) [SIG Storage]
- Fixed cleaning of CSI raw block volumes. ([#87978](#), [@jsafrane](#)) [SIG Storage]
- Fixed AWS Cloud Provider attempting to delete LoadBalancer security group it didn't provision, and fixed AWS Cloud Provider creating a default LoadBalancer security group even if annotation `service.beta.kubernetes.io/aws-load-balancer-security-groups` is present because the intended behavior of `aws-load-balancer-security-groups` is to replace all security groups assigned to the load balancer. ([#84265](#), [@bhagwat070919](#)) [SIG Cloud Provider]
- Fixed two scheduler metrics (`pending_pods` and `schedule_attempts_total`) not being recorded ([#87692](#), [@everpeace](#)) [SIG Scheduling]
- Fixes an issue with kubelet-reported pod status on deleted/recreated pods. ([#86320](#), [@liggitt](#)) [SIG Node]
- Fixes conversion error in multi-version custom resources that could cause metadata.generation to increment on no-op patches or updates of a custom resource. ([#88995](#), [@liggitt](#)) [SIG API Machinery]
- Fixes issue where AAD token obtained by kubectl is incompatible with on-behalf-of flow and oidc. The audience claim before this fix has "spn:" prefix. After this fix, "spn:" prefix is omitted. ([#86412](#), [@weinong](#)) [SIG API Machinery, Auth and Cloud Provider]
- Fixes an issue where you can't attach more than 15 GCE Persistent Disks to c2, n2, m1, m2 machine types. ([#88602](#), [@yuga711](#)) [SIG Storage]

- Fixes kube-proxy when EndpointSlice feature gate is enabled on Windows. ([#86016](#), [@roboscott](#)) [SIG Auth and Network]
- Fixes kubelet crash in client certificate rotation cases ([#88079](#), [@liggitt](#)) [SIG API Machinery, Auth and Node]
- Fixes service account token admission error in clusters that do not run the service account token controller ([#87029](#), [@liggitt](#)) [SIG Auth]
- Fixes v1.17.0 regression in --service-cluster-ip-range handling with IPv4 ranges larger than 65536 IP addresses ([#86534](#), [@liggitt](#)) [SIG Network]
- Fixes wrong validation result of NetworkPolicy PolicyTypes ([#85747](#), [@tnqn](#)) [SIG Network]
- For subprotocol negotiation, both client and server protocol is required now. ([#86646](#), [@tedyu](#)) [SIG API Machinery and Node]
- For volumes that allow attaches across multiple nodes, attach and detach operations across different nodes are now executed in parallel. ([#88678](#), [@verult](#)) [SIG Storage]
- Garbage collector now can correctly orphan ControllerRevisions when StatefulSets are deleted with orphan propagation policy. ([#84984](#), [@cofyc](#)) [SIG Apps]
- Get-kube.sh uses the gcloud's current local GCP service account for auth when the provider is GCE or GKE instead of the metadata server default ([#88383](#), [@BenTheElder](#)) [SIG Cluster Lifecycle]
- Golang/x/net has been updated to bring in fixes for CVE-2020-9283 ([#88381](#), [@BenTheElder](#)) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle and Instrumentation]
- If a serving certificate's param specifies a name that is an IP for an SNI certificate, it will have priority for replying to server connections. ([#85308](#), [@deads2k](#)) [SIG API Machinery]
- Improved yaml parsing performance ([#85458](#), [@cjcullen](#)) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Node]
- Improves performance of the node authorizer ([#87696](#), [@liggitt](#)) [SIG Auth]
- In GKE alpha clusters it will be possible to use the service annotation cloud.google.com/network-tier: Standard ([#88487](#), [@zioproto](#)) [SIG Cloud Provider]
- Includes FSType when describing CSI persistent volumes. ([#85293](#), [@huffmanca](#)) [SIG CLI and Storage]
- Iptables/userspace proxy: improve performance by getting local addresses only once per sync loop, instead of for every external IP ([#85617](#), [@andrewsykim](#)) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Network]
- Kube-aggregator: always sets unavailableGauge metric to reflect the current state of a service. ([#87778](#), [@p0lyn0mial](#)) [SIG API Machinery]
- Kube-apiserver: fixed a conflict error encountered attempting to delete a pod with gracePeriodSeconds=0 and a resourceVersion precondition ([#85516](#), [@michaelgugino](#)) [SIG API Machinery]
- Kube-proxy no longer modifies shared EndpointSlices. ([#86092](#), [@roboscott](#)) [SIG Network]
- Kube-proxy: on dual-stack mode, if it is not able to get the IP Family of an endpoint, logs it with level InfoV(4) instead of Warning, avoiding

- flooding the logs for endpoints without addresses ([#88934](#), [@aojea](#)) [SIG Network]
- Kubeadm allows to configure single-stack clusters if dual-stack is enabled ([#87453](#), [@aojea](#)) [SIG API Machinery, Cluster Lifecycle and Network]
 - Kubeadm now includes CoreDNS version 1.6.7 ([#86260](#), [@rajansandeep](#)) [SIG Cluster Lifecycle]
 - Kubeadm upgrades always persist the etcd backup for stacked ([#86861](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
 - Kubeadm: 'kubeadm alpha kubelet config download' has been removed, please use 'kubeadm upgrade node phase kubelet-config' instead ([#87944](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
 - Kubeadm: Forward cluster name to the controller-manager arguments ([#85817](#), [@ereslibre](#)) [SIG Cluster Lifecycle]
 - Kubeadm: add support for the "ci/k8s-master" version label as a replacement for "ci-cross/*", which no longer exists. ([#86609](#), [@Pensu](#)) [SIG Cluster Lifecycle]
 - Kubeadm: apply further improvements to the tentative support for concurrent etcd member join. Fixes a bug where multiple members can receive the same hostname. Increase the etcd client dial timeout and retry timeout for add/remove/... operations. ([#87505](#), [@neolit123](#)) [SIG Cluster Lifecycle]
 - Kubeadm: don't write the kubelet environment file on "upgrade apply" ([#85412](#), [@boluisa](#)) [SIG Cluster Lifecycle]
 - Kubeadm: fix potential panic when executing "kubeadm reset" with a corrupted kubelet.conf file ([#86216](#), [@neolit123](#)) [SIG Cluster Lifecycle]
 - Kubeadm: fix the bug that 'kubeadm upgrade' hangs in single node cluster ([#88434](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
 - Kubeadm: make sure images are pre-pulled even if a tag did not change but their contents changed ([#85603](#), [@bart0sh](#)) [SIG Cluster Lifecycle]
 - Kubeadm: remove 'kubeadm upgrade node config' command since it was deprecated in v1.15, please use 'kubeadm upgrade node phase kubelet-config' instead ([#87975](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
 - Kubeadm: remove the deprecated CoreDNS feature-gate. It was set to "true" since v1.11 when the feature went GA. In v1.13 it was marked as deprecated and hidden from the CLI. ([#87400](#), [@neolit123](#)) [SIG Cluster Lifecycle]
 - Kubeadm: retry kubeadm-config ConfigMap creation or mutation if the apiserver is not responding. This will improve resiliency when joining new control plane nodes. ([#85763](#), [@ereslibre](#)) [SIG Cluster Lifecycle]
 - Kubeadm: tolerate whitespace when validating certificate authority PEM data in kubeconfig files ([#86705](#), [@neolit123](#)) [SIG Cluster Lifecycle]
 - Kubeadm: use bind-address option to configure the kube-controller-manager and kube-scheduler http probes ([#86493](#), [@aojea](#)) [SIG Cluster Lifecycle]
 - Kubeadm: uses the api-server AdvertiseAddress IP family to choose the etcd endpoint IP family for non external etcd clusters ([#85745](#), [@aojea](#)) [SIG Cluster Lifecycle]

- Kubectl cluster-info dump --output-directory=xxx now generates files with an extension depending on the output format. ([#82070](#), [@olivierlemasle](#)) [SIG CLI]
- Kubectl describe <type> and kubectl top pod will return a message saying "No resources found" or "No resources found in <namespace> namespace" if there are no results to display. ([#87527](#), [@brianpursley](#)) [SIG CLI]
- Kubectl drain node --dry-run will list pods that would be evicted or deleted ([#82660](#), [@sallyom](#)) [SIG CLI]
- Kubectl set resources will no longer return an error if passed an empty change for a resource. kubectl set subject will no longer return an error if passed an empty change for a resource. ([#85490](#), [@sallyom](#)) [SIG CLI]
- Kubelet metrics gathered through metrics-server or prometheus should no longer timeout for Windows nodes running more than 3 pods. ([#87730](#), [@marosset](#)) [SIG Node, Testing and Windows]
- Kubelet metrics have been changed to buckets. For example the exec/{podNamespace}/{podID}/{containerName} is now just exec. ([#87913](#), [@cheftako](#)) [SIG Node]
- Kubelets perform fewer unnecessary pod status update operations on the API server. ([#88591](#), [@smarterclayton](#)) [SIG Node and Scalability]
- Kubernetes will try to acquire the iptables lock every 100 msec during 5 seconds instead of every second. This is especially useful for environments using kube-proxy in iptables mode with a high churn rate of services. ([#85771](#), [@aojea](#)) [SIG Network]
- Limit number of instances in a single update to GCE target pool to 1000. ([#87881](#), [@wojtek-t](#)) [SIG Cloud Provider, Network and Scalability]
- Make Azure clients only retry on specified HTTP status codes ([#88017](#), [@feiskyer](#)) [SIG Cloud Provider]
- Make error message and service event message more clear ([#86078](#), [@feiskyer](#)) [SIG Cloud Provider]
- Minimize AWS NLB health check timeout when externalTrafficPolicy set to Local ([#73363](#), [@kellycampbell](#)) [SIG Cloud Provider]
- Pause image contains "Architecture" in non-amd64 images ([#87954](#), [@BenTheElder](#)) [SIG Release]
- Pause image upgraded to 3.2 in kubelet and kubeadm. ([#88173](#), [@BenTheElder](#)) [SIG CLI, Cluster Lifecycle, Node and Testing]
- Plugin/PluginConfig and Policy APIs are mutually exclusive when running the scheduler ([#88864](#), [@alculquicondor](#)) [SIG Scheduling]
- Remove FilteredNodesStatuses argument from PreScore's interface. ([#88189](#), [@skilxn-go](#)) [SIG Scheduling and Testing]
- Resolved a performance issue in the node authorizer index maintenance. ([#87693](#), [@liggitt](#)) [SIG Auth]
- Resolved regression in admission, authentication, and authorization webhook performance in v1.17.0-rc.1 ([#85810](#), [@liggitt](#)) [SIG API Machinery and Testing]
- Resolves performance regression in kubectl get all and in client-go discovery clients constructed using NewDiscoveryClientForConfig or NewDiscoveryClientForConfigOrDie. ([#86168](#), [@liggitt](#)) [SIG API Machinery]

- Reverted a kubectl azure auth module change where oidc claim spn: prefix was omitted resulting a breaking behavior with existing Azure AD OIDC enabled api-server ([#87507](#), [@weinong](#)) [SIG API Machinery, Auth and Cloud Provider]
- Shared informers are now more reliable in the face of network disruption. ([#86015](#), [@squeed](#)) [SIG API Machinery]
- Specifying PluginConfig for the same plugin more than once fails scheduler startup. Specifying extenders and configuring .ignoredResources for the NodeResourcesFit plugin fails ([#88870](#), [@alculquicondor](#)) [SIG Scheduling]
- Terminating a restartPolicy=Never pod no longer has a chance to report the pod succeeded when it actually failed. ([#88440](#), [@smarterclayton](#)) [SIG Node and Testing]
- The CSR signing cert/key pairs will be reloaded from disk like the kube-apiserver cert/key pairs ([#86816](#), [@deads2k](#)) [SIG API Machinery, Apps and Auth]
- The EventRecorder from k8s.io/client-go/tools/events will now create events in the default namespace (instead of kube-system) when the related object does not have it set. ([#88815](#), [@enj](#)) [SIG API Machinery]
- The audit event sourceIPs list will now always end with the IP that sent the request directly to the API server. ([#87167](#), [@tallclair](#)) [SIG API Machinery and Auth]
- The sample-apiserver aggregated conformance test has updated to use the Kubernetes v1.17.0 sample apiserver ([#84735](#), [@liggitt](#)) [SIG API Machinery, Architecture, CLI and Testing]
- To reduce chances of throttling, VM cache is set to nil when Azure node provisioning state is deleting ([#87635](#), [@feiskyer](#)) [SIG Cloud Provider]
- VMSS cache is added so that less chances of VMSS GET throttling ([#85885](#), [@nilo19](#)) [SIG Cloud Provider]
- Wait for kubelet & kube-proxy to be ready on Windows node within 10s ([#85228](#), [@YangLu1031](#)) [SIG Cluster Lifecycle]
- kubectl apply -f <file> --prune -n <namespace> should prune all resources not defined in the file in the cli specified namespace. ([#85613](#), [@MartinKaburu](#)) [SIG CLI]
- kubectl create clusterrolebinding creates rbac.authorization.k8s.io/v1 object ([#85889](#), [@oke-py](#)) [SIG CLI]
- kubectl diff now returns 1 only on diff finding changes, and >1 on kubectl errors. The "exit status code 1" message has also been muted. ([#87437](#), [@apelisse](#)) [SIG CLI and Testing]

Dependencies

- Update Calico to v3.8.4 ([#84163](#), [@david-tigera](#)) [SIG Cluster Lifecycle]
- Update aws-sdk-go dependency to v1.28.2 ([#87253](#), [@SaranBalaji90](#)) [SIG API Machinery and Cloud Provider]
- Update CNI version to v0.8.5 ([#78819](#), [@justaugustus](#)) [SIG Release, Testing, Network, Cluster Lifecycle and API Machinery]
- Update cri-tools to v1.17.0 ([#86305](#), [@saschagrunert](#)) [SIG Release and Cluster Lifecycle]
- Pause image upgraded to 3.2 in kubelet and kubeadm ([#88173](#), [@BenTheElder](#)) [SIG CLI, Node, Testing and Cluster Lifecycle]

- Update CoreDNS version to 1.6.7 in kubeadm ([#86260](#), [@rajansandeep](#))[SIG Cluster Lifecycle]
- Update golang.org/x/crypto to fix CVE-2020-9283 ([#8838](#), [@BenTheElder](#))[SIG CLI, Instrumentation, API Machinery, CLuster Lifecycle and Cloud Provider]
- Update Go to 1.13.8 ([#87648](#), [@ialidzhikov](#))[SIG Release and Testing]
- Update Cluster-Autoscaler to 1.18.0 ([#89095](#), [@losipiuk](#))[SIG Autoscaling and Cluster Lifecycle]

v1.18.0-rc.1

[Documentation](#)

Downloads for v1.18.0-rc.1

filename	sha512 hash
kubernetes.tar.gz	c17231d5de2e0677e8af8259baa11a388625821c79b86362049f2edb36640
kubernetes-src.tar.gz	e84ffad57c301f5d6e90f916b996d5abb0c987928c3ca6b1565f7b042588f83

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	1aea99923d492436b3eb91aaecffac94e5d0aa2b38a0930d266fda85c665
kubernetes-client-darwin-amd64.tar.gz	07fa7340a959740bd52b83ff44438bbd988e235277dad1e43f125f08ac85
kubernetes-client-linux-386.tar.gz	48cebd26448fdd47aa36257baa4c716a98fda055bbf6a05230f2a3fe3c1b9
kubernetes-client-linux-amd64.tar.gz	c3a5fedf263f07a07f59c01fea6c63c1e0b76ee8dc67c45b6c134255c28ed6
kubernetes-client-linux-arm.tar.gz	a6b11a55bd38583bbaac14931a6862f8ce6493afe30947ba29e5556654a5
kubernetes-client-linux-arm64.tar.gz	9e15331ac8010154a9b64f5488969fc8ee2f21059639896cb84c5cf4f05f4c
kubernetes-client-linux-ppc64le.tar.gz	f828fe6252678de9d4822e482f5873309ae9139b2db87298ab3273ce45d3
kubernetes-client-linux-s390x.tar.gz	19da4b45f0666c063934af616f3e7ed3caa99d4ee1e46d53efadc7a8a4d38
kubernetes-client-windows-386.tar.gz	775c9afb6cb3e7c4ba53e9f48a5df2cf207234a33059bd74448bc9f177dd1
kubernetes-client-windows-amd64.tar.gz	208d2595a5b57ac97aac75b4a2a6130f0c937f781a030bde1a432daf4bc5

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	dcf832eae04f9f52ff473754ef5cfe697b35f4dc1a282622c94fa10943c8c35f4a87
kubernetes-server-linux-arm.tar.gz	a04e34bea28eb1c8b492e8b1dd3c0dd87ebee71a7dbbef72be10a335e553361
kubernetes-server-linux-arm64.tar.gz	a6af086b07a8c2e498f32b43e6511bf6a5e6baf358c572c6910c8df17cd6cae94f
kubernetes-server-linux-ppc64le.tar.gz	5a960ef5ba0c255f587f2ac0b028cd03136dc91e4efc5d1becab46417852e5524
kubernetes-server-linux-s390x.tar.gz	0f32c7d9b14bc238b9a5764d8f00edc4d3bf36bcf06b340b81061424e6070768

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	27d8955d535d14f3f4dca501fd27e4f06fad84c6da878ea5332a5c83b6955667f6
kubernetes-node-linux-arm.tar.gz	0d56eccad63ba608335988e90b377fe8ae978b177dc836cdb803a5c99d99e8f3
kubernetes-node-linux-arm64.tar.gz	79bb9be66f9e892d866b28e5cc838245818edb9706981fab6ccbff493181b341c
kubernetes-node-linux-ppc64le.tar.gz	3e9e2c6f9a2747d828069511dce8b4034c773c2d122f005f4508e22518055c1e0
kubernetes-node-linux-s390x.tar.gz	4f96e018c336fa13bb6df6f7217fe46a2b5c47f806f786499c429604ccba2ebe55
kubernetes-node-windows-amd64.tar.gz	ab110d76d506746af345e5897ef4f6993d5f53ac818ba69a334f3641047351aa6

Changelog since v1.18.0-beta.2

Changes by Kind

API Change

- Removes ConfigMap as suggestion for IngressClass parameters ([#89093](#), [@roboscott](#)) [SIG Network]

Other (Bug, Cleanup or Flake)

- EndpointSlice should not contain endpoints for terminating pods ([#89056](#), [@andrewsykim](#)) [SIG Apps and Network]
- Fix a bug where ExternalTrafficPolicy is not applied to service ExternalIPs. ([#88786](#), [@freehan](#)) [SIG Network]
- Fix invalid VMSS updates due to incorrect cache ([#89002](#), [@ArchangelSDY](#)) [SIG Cloud Provider]
- Fix isCurrentInstance for Windows by removing the dependency of hostname. ([#89138](#), [@feiskyer](#)) [SIG Cloud Provider]
- Fixed a data race in kubelet image manager that can cause static pod workers to silently stop working. ([#88915](#), [@roycai hw](#)) [SIG Node]
- Fixed an issue that could cause the kubelet to incorrectly run concurrent pod reconciliation loops and crash. ([#89055](#), [@tedyu](#)) [SIG Node]
- Kube-proxy: on dual-stack mode, if it is not able to get the IP Family of an endpoint, logs it with level InfoV(4) instead of Warning, avoiding flooding the logs for endpoints without addresses ([#88934](#), [@aojea](#)) [SIG Network]
- Update Cluster Autoscaler to 1.18.0; changelog: <https://github.com/kubernetes/autoscaler/releases/tag/cluster-autoscaler-1.18.0> ([#89095](#), [@losipiuk](#)) [SIG Autoscaling and Cluster Lifecycle]

v1.18.0-beta.2

[Documentation](#)

Downloads for v1.18.0-beta.2

filename	sha512 hash
kubernetes.tar.gz	3017430ca17f8a3523669b4a02c39cedfc6c48b07281bc0a67a9fbe9d76547
kubernetes-src.tar.gz	c5fd60601380a99efff4458b1c9cf4dc02195f6f756b36e590e54dff68f7064da

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	7e49ede167b9271d4171e477fa21d267b2fb35f80869337d5b323198dc1
kubernetes-client-darwin-amd64.tar.gz	3f5cdf0e85eee7d0773e0ae2df1c61329dea90e0da92b02dae1ffd101008c
kubernetes-client-linux-386.tar.gz	b67b41c11bfecb88017c33fee21735c56f24cf6f7851b63c752495fc0fb56
kubernetes-client-linux-amd64.tar.gz	1fef2197cb80003e3a5c26f05e889af9d85fbbc23e27747944d2997ace4bf
kubernetes-client-linux-arm.tar.gz	84e5f4d9776490219ee94a84adccd5dfc7c0362eb330709771afcde95ec83
kubernetes-client-linux-arm64.tar.gz	ba613b114e0cca32fa21a3d10f845aa2f215d3af54e775f917ff93919f7dd7
kubernetes-client-linux-ppc64le.tar.gz	502a6938d8c4bbe04abbd19b59919d86765058ff72334848be4012cec49
kubernetes-client-linux-s390x.tar.gz	c24700e0ed2ef5c1d2dd282d638c88d90392ae90ea420837b39fd8e1cfc1
kubernetes-client-windows-386.tar.gz	0d4c5a741b052f790c8b0923c9586ee9906225e51cf4dc8a56fc303d4d61
kubernetes-client-windows-amd64.tar.gz	841ef2e306c0c9593f04d9528ee019bf3b667761227d9afc1d6ca8bf1aa56

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	b373df2e6ef55215e712315a5508e85a39126bd81b7b93c6b6305238919a88c7
kubernetes-server-linux-arm.tar.gz	b8103cb743c23076ce8dd7c2da01c8dd5a542fbac8480e82dc673139c8ee5ec4
kubernetes-server-linux-arm64.tar.gz	8f8f05cf64fb9c8d80cdcb4935b2d3e3edc48bdd303231ae12f93e3f4d9792374
kubernetes-server-linux-ppc64le.tar.gz	b313b911c46f2ec129537407af3f165f238e48caeb4b9e530783ffa3659304a544
kubernetes-server-linux-s390x.tar.gz	a1b6b06571141f507b12e5ef98efb88f4b6b9aba924722b2a74f11278d29a2972

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	20e02ca327543cd5b2568ead3d5de164cbfb2914ab6416106d906bf12fcfbc4e5
kubernetes-node-linux-arm.tar.gz	ecd817ef05d6284f9c6592b84b0a48ea31cf4487030c9fb36518474b2a33dad11
kubernetes-node-linux-arm64.tar.gz	0020d32b7908ffd5055c8b26a8b3033e4702f89efcffe3f6fcdb8a9921fa8eaaed
kubernetes-node-linux-ppc64le.tar.gz	e065411d66d486e7793449c1b2f5a412510b913bf7f4e728c0a20e275642b766
kubernetes-node-linux-s390x.tar.gz	082ee90413beaaea41d6cbe9a18f7d783a95852607f3b94190e0ca12aacdd97d
kubernetes-node-windows-amd64.tar.gz	fb5aca0cc36be703f9d4033eababd581bac5de8399c50594db087a99ed4cb56e

Changelog since v1.18.0-beta.1

Urgent Upgrade Notes

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

- kubectl no longer defaults to `http://localhost:8080`. If you own one of these legacy clusters, you are *strongly*- encouraged to secure your server. If you cannot secure your server, you can set `KUBERNETES_MASTER` if you were relying on that behavior and you're a client-go user. Set `-server`, `--kubeconfig` or `KUBECONFIG` to make it work in kubectl. ([#86173](#), [@soltys](#)) [SIG API Machinery, CLI and Testing]

Changes by Kind

Deprecation

- AlgorithmSource is removed from v1alpha2 Scheduler ComponentConfig ([#87999](#), [@damemi](#)) [SIG Scheduling]
- Kube-proxy: deprecate `--healthz-port` and `--metrics-port` flag, please use `--healthz-bind-address` and `--metrics-bind-address` instead ([#88512](#), [@SataQiu](#)) [SIG Network]

- Kubeadm: deprecate the usage of the experimental flag '--use-api' under the 'kubeadm alpha certs renew' command. ([#88827](#), [@neolit123](#)) [SIG Cluster Lifecycle]

API Change

- A new IngressClass resource has been added to enable better Ingress configuration. ([#88509](#), [@roboscott](#)) [SIG API Machinery, Apps, CLI, Network, Node and Testing]
- Added GenericPVCDataSource feature gate to enable using arbitrary custom resources as the data source for a PVC. ([#88636](#), [@bswartz](#)) [SIG Apps and Storage]
- Allow user to specify fsgroup permission change policy for pods ([#88488](#), [@gnufied](#)) [SIG Apps and Storage]
- BlockVolume and CSIBlockVolume features are now GA. ([#88673](#), [@jsafrane](#)) [SIG Apps, Node and Storage]
- CustomResourceDefinition schemas that use x-kubernetes-list-map-keys to specify properties that uniquely identify list items must make those properties required or have a default value, to ensure those properties are present for all list items. See <https://kubernetes.io/docs/reference/using-api/api-concepts/#merge-strategy> for details. ([#88076](#), [@eloyekunle](#)) [SIG API Machinery and Testing]
- Fixes a regression with clients prior to 1.15 not being able to update podIP in pod status, or podCIDR in node spec, against >= 1.16 API servers ([#88505](#), [@liggitt](#)) [SIG Apps and Network]
- Ingress: Add Exact and Prefix matching to Ingress PathTypes ([#88587](#), [@cmluciano](#)) [SIG Apps, Cluster Lifecycle and Network]
- Ingress: Add alternate backends via TypedLocalObjectReference ([#88775](#), [@cmluciano](#)) [SIG Apps and Network]
- Ingress: allow wildcard hosts in IngressRule ([#88858](#), [@cmluciano](#)) [SIG Network]
- Kube-controller-manager and kube-scheduler expose profiling by default to match the kube-apiserver. Use --enable-profiling=false to disable. ([#88663](#), [@deads2k](#)) [SIG API Machinery, Cloud Provider and Scheduling]
- Move TaintBasedEvictions feature gates to GA ([#87487](#), [@skilxn-go](#)) [SIG API Machinery, Apps, Node, Scheduling and Testing]
- New flag --endpointslice-updates-batch-period in kube-controller-manager can be used to reduce number of endpointslice updates generated by pod changes. ([#88745](#), [@mborsz](#)) [SIG API Machinery, Apps and Network]
- Scheduler Extenders can now be configured in the v1alpha2 component config ([#88768](#), [@damemi](#)) [SIG Release, Scheduling and Testing]
- The apiserver/v1alph1#EgressSelectorConfiguration API is now beta. ([#88502](#), [@caesarxuchao](#)) [SIG API Machinery]
- The storage.k8s.io/CSIDriver has moved to GA, and is now available for use. ([#84814](#), [@huffmanca](#)) [SIG API Machinery, Apps, Auth, Node, Scheduling, Storage and Testing]
- VolumePVCDataSource moves to GA in 1.18 release ([#88686](#), [@j-griffith](#)) [SIG Apps, CLI and Cluster Lifecycle]

Feature

- Add `rest_client_rate_limiter_duration_seconds` metric to component-base to track client side rate limiter latency in seconds. Broken down by verb and URL. ([#88134](#), [@jennybuckley](#)) [SIG API Machinery, Cluster Lifecycle and Instrumentation]
- Allow user to specify resource using `--filename` flag when invoking `kubectl exec` ([#88460](#), [@soltys](#)) [SIG CLI and Testing]
- Apiserver add a new flag `--goaway-chance` which is the fraction of requests that will be closed gracefully(GOAWAY) to prevent HTTP/2 clients from getting stuck on a single apiserver. After the connection closed(received GOAWAY), the client's other in-flight requests won't be affected, and the client will reconnect. The flag min value is 0 (off), max is .02 (1/50 requests); .001 (1/1000) is a recommended starting point. Clusters with single apiservers, or which don't use a load balancer, should NOT enable this. ([#88567](#), [@answer1991](#)) [SIG API Machinery]
- Azure: add support for single stack IPv6 ([#88448](#), [@aramase](#)) [SIG Cloud Provider]
- DefaultConstraints can be specified for the PodTopologySpread plugin in the component config ([#88671](#), [@alculquicondor](#)) [SIG Scheduling]
- Kubeadm: support Windows specific kubelet flags in `kubeadm-flags.env` ([#88287](#), [@gab-satchi](#)) [SIG Cluster Lifecycle and Windows]
- Kubectl cluster-info dump changed to only display a message telling you the location where the output was written when the output is not standard output. ([#88765](#), [@brianpursley](#)) [SIG CLI]
- Print NotReady when pod is not ready based on its conditions. ([#88240](#), [@soltys](#)) [SIG CLI]
- Scheduler Extender API is now located under `k8s.io/kube-scheduler/extender` ([#88540](#), [@damemi](#)) [SIG Release, Scheduling and Testing]
- Signatures on scale client methods have been modified to accept `context.Context` as a first argument. Signatures of `Get`, `Update`, and `Patch` methods have been updated to accept `GetOptions`, `UpdateOptions` and `PatchOptions` respectively. ([#88599](#), [@julianvmodesto](#)) [SIG API Machinery, Apps, Autoscaling and CLI]
- Signatures on the dynamic client methods have been modified to accept `context.Context` as a first argument. Signatures of `Delete` and `DeleteCollection` methods now accept `DeleteOptions` by value instead of by reference. ([#88906](#), [@liggitt](#)) [SIG API Machinery, Apps, CLI, Cluster Lifecycle, Storage and Testing]
- Signatures on the metadata client methods have been modified to accept `context.Context` as a first argument. Signatures of `Delete` and `DeleteCollection` methods now accept `DeleteOptions` by value instead of by reference. ([#88910](#), [@liggitt](#)) [SIG API Machinery, Apps and Testing]
- Webhooks will have alpha support for network proxy ([#85870](#), [@Jefftree](#)) [SIG API Machinery, Auth and Testing]
- When client certificate files are provided, reload files for new connections, and close connections when a certificate changes. ([#79083](#), [@jackkleeman](#)) [SIG API Machinery, Auth, Node and Testing]
- When deleting objects using `kubectl` with the `--force` flag, you are no longer required to also specify `--grace-period=0`. ([#87776](#), [@brianpursley](#)) [SIG CLI]

- `kubect` now contains a `kubect` alpha `debug` command. This command allows attaching an ephemeral container to a running pod for the purposes of debugging. ([#88004](#), [@verb](#)) [SIG CLI]

Documentation

- Update Japanese translation for `kubect` help ([#86837](#), [@inductor](#)) [SIG CLI and Docs]
- `kubect` plugin now prints a note how to install `krew` ([#88577](#), [@corneliusweig](#)) [SIG CLI]

Other (Bug, Cleanup or Flake)

- Azure VMSS LoadBalancerBackendAddressPools updating has been improved with sequential-sync + concurrent-async requests. ([#88699](#), [@feiskyer](#)) [SIG Cloud Provider]
- AzureFile and CephFS use new Mount library that prevents logging of sensitive mount options. ([#88684](#), [@saad-ali](#)) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Storage]
- Build: Enable kube-cross image-building on K8s Infra ([#88562](#), [@justaugustus](#)) [SIG Release and Testing]
- Client-go certificate manager rotation gained the ability to preserve optional intermediate chains accompanying issued certificates ([#88744](#), [@jackkleeman](#)) [SIG API Machinery and Auth]
- Conformance image now depends on stretch-slim instead of debian-hyperkube-base as that image is being deprecated and removed. ([#88702](#), [@dims](#)) [SIG Cluster Lifecycle, Release and Testing]
- Deprecate `--generator` flag from `kubect` create commands ([#88655](#), [@solysh](#)) [SIG CLI]
- FIX: prevent apiserver from panicking when failing to load audit webhook config file ([#88879](#), [@JoshVanL](#)) [SIG API Machinery and Auth]
- Fix `/readyz` to return error immediately after a shutdown is initiated, before the `--shutdown-delay-duration` has elapsed. ([#88911](#), [@tkashem](#)) [SIG API Machinery]
- Fix a bug where kubenet fails to parse the tc output. ([#83572](#), [@chendojs](#)) [SIG Network]
- Fix describe ingress annotations not sorted. ([#88394](#), [@zhouya0](#)) [SIG CLI]
- Fix handling of `aws-load-balancer-security-groups` annotation. Security-Groups assigned with this annotation are no longer modified by kubernetes which is the expected behaviour of most users. Also no

unnecessary Security-Groups are created anymore if this annotation is used. ([#83446](#), [@Elias481](#)) [SIG Cloud Provider]

- Fix kubectl create deployment image name ([#86636](#), [@zhouya0](#)) [SIG CLI]
- Fix missing "apiVersion" for "involvedObject" in Events for Nodes. ([#87537](#), [@uthark](#)) [SIG Apps and Node]
- Fix that prevents repeated fetching of PVC/PV objects by kubelet when processing of pod volumes fails. While this prevents hammering API server in these error scenarios, it means that some errors in processing volume(s) for a pod could now take up to 2-3 minutes before retry. ([#88141](#), [@tedyu](#)) [SIG Node and Storage]
- Fix: azure file mount timeout issue ([#88610](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fix: corrupted mount point in csi driver ([#88569](#), [@andyzhangx](#)) [SIG Storage]
- Fixed a bug in the TopologyManager. Previously, the TopologyManager would only guarantee alignment if container creation was serialized in some way. Alignment is now guaranteed under all scenarios of container creation. ([#87759](#), [@klueska](#)) [SIG Node]
- Fixed block CSI volume cleanup after timeouts. ([#88660](#), [@jsafrane](#)) [SIG Node and Storage]
- Fixes issue where you can't attach more than 15 GCE Persistent Disks to c2, n2, m1, m2 machine types. ([#88602](#), [@yuga711](#)) [SIG Storage]
- For volumes that allow attaches across multiple nodes, attach and detach operations across different nodes are now executed in parallel. ([#88678](#), [@verult](#)) [SIG Apps, Node and Storage]
- Hide kubectl.kubernetes.io/last-applied-configuration in describe command ([#88758](#), [@soltys](#)) [SIG Auth and CLI]
- In GKE alpha clusters it will be possible to use the service annotation cloud.google.com/network-tier: Standard ([#88487](#), [@zioproto](#)) [SIG Cloud Provider]
- Kubelets perform fewer unnecessary pod status update operations on the API server. ([#88591](#), [@smarterclayton](#)) [SIG Node and Scalability]
- Plugin/PluginConfig and Policy APIs are mutually exclusive when running the scheduler ([#88864](#), [@alculquicondor](#)) [SIG Scheduling]
- Specifying PluginConfig for the same plugin more than once fails scheduler startup.

Specifying extenders and configuring `.ignoredResources` for the `NodeResourcesFit` plugin fails ([#88870](#), [@alculquicondor](#)) [SIG Scheduling]

- Support TLS Server Name overrides in kubeconfig file and via `--tls-server-name` in `kubectl` ([#88769](#), [@deads2k](#)) [SIG API Machinery, Auth and CLI]
- Terminating a `restartPolicy=Never` pod no longer has a chance to report the pod succeeded when it actually failed. ([#88440](#), [@smarterclayton](#)) [SIG Node and Testing]
- The `EventRecorder` from `k8s.io/client-go/tools/events` will now create events in the default namespace (instead of `kube-system`) when the related object does not have it set. ([#88815](#), [@enj](#)) [SIG API Machinery]
- The audit event `sourceIPs` list will now always end with the IP that sent the request directly to the API server. ([#87167](#), [@tallclair](#)) [SIG API Machinery and Auth]
- Update to use `golang` 1.13.8 ([#87648](#), [@ialidzhikov](#)) [SIG Release and Testing]
- Validate `kube-proxy` flags `--ipvs-tcp-timeout`, `--ipvs-tcpfin-timeout`, `--ipvs-udp-timeout` ([#88657](#), [@chendojts](#)) [SIG Network]

v1.18.0-beta.1

[Documentation](#)

Downloads for v1.18.0-beta.1

filename	sha512 hash
kubernetes.tar.gz	7c182ca905b3a31871c01ab5fdaf46f074547536c7975e069ff230af0d402dfc
kubernetes-src.tar.gz	d104b8c792b1517bd730787678c71c8ee3b259de81449192a49a1c6e37a6f

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	bc337bb8f200a789be4b97ce99b9d7be78d35ebd64746307c28339dc462
kubernetes-client-darwin-amd64.tar.gz	38dfa5e0b0cfff39942c913a6bcb2ad8868ec43457d35cffba08217bb6e75
kubernetes-client-linux-386.tar.gz	8e63ec7ce29c69241120c037372c6c779e3f16253eabd612c7cbe6aa8932
kubernetes-client-linux-amd64.tar.gz	c1be9f184a7c3f896a785c41cd6ece9d90d8cb9b1f6088bdfb5557d8856c5

filename	sha512 hash
kubernetes-client-linux-arm.tar.gz	8eab02453cfd9e847632a774a0e0cf3a33c7619fb4ced7f1840e1f71444e8
kubernetes-client-linux-arm64.tar.gz	f7df0ec02d2e7e63278d5386e8153cfe2b691b864f17b6452cc824a5f328c
kubernetes-client-linux-ppc64le.tar.gz	36dd5b10addca678a518e6d052c9d6edf473e3f87388a2f03f714c93c5fbf
kubernetes-client-linux-s390x.tar.gz	5bdbb44b996ab4ccf3a383780270f5cfdbf174982c300723c8bddf0a48ae5
kubernetes-client-windows-386.tar.gz	5dea3d4c4e91ef889850143b361974250e99a3c526f5efee23ff9ccdc2ce
kubernetes-client-windows-amd64.tar.gz	db298e698391368703e6aea7f4345aec5a4b8c69f9d8ff6c99fb5804a6cea

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	c6284929dd5940e750b48db72ffbc09f73c5ec31ab3db283babb8e4e07cd8cbb
kubernetes-server-linux-arm.tar.gz	6fc9552cf082c54cc0833b19876117c87ba7feb5a12c7e57f71b52208daf03eaf3
kubernetes-server-linux-arm64.tar.gz	b794b9c399e548949b5bfb2fe71123e86c2034847b2c99aca34b6de718a35355
kubernetes-server-linux-ppc64le.tar.gz	fd daed7a54f97046a91c29534645811c6346e973e22950b2607b8c119c2377e9
kubernetes-server-linux-s390x.tar.gz	65951a534bb55069c7419f41cbcdfe2fae31541d8a3f9eca11fc2489addf281c5a

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	992059efb5cae7ed0ef55820368d854bad1c6d13a70366162cd3b5111ce24c37
kubernetes-node-linux-arm.tar.gz	c63ae0f8add5821ad267774314b8c8c1ffe3b785872bf278e721fd5dfdad1a5db
kubernetes-node-linux-arm64.tar.gz	47adb9ddf6eaf8f475b89f59ee16fbd5df183149a11ad1574eaa645b47a6d58ae

filename	sha512 hash
kubernetes-node-linux-ppc64le.tar.gz	a3bc4a165567c7b76a3e45ab7b102d6eb3ecf373eb048173f921a4964cf9be889
kubernetes-node-linux-s390x.tar.gz	109ddf37c748f69584c829db57107c3518defe005c11fcd2a1471845c15aae0a3
kubernetes-node-windows-amd64.tar.gz	a3a75d2696ad3136476ad7d811e8eabaff5111b90e592695e651d6111f819ebf0

Changelog since v1.18.0-beta.0

Urgent Upgrade Notes

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

- The StreamingProxyRedirects feature and `--redirect-container-streaming` flag are deprecated, and will be removed in a future release. The default behavior (proxy streaming requests through the kubelet) will be the only supported option. If you are setting `--redirect-container-streaming=true`, then you must migrate off this configuration. The flag will no longer be able to be enabled starting in v1.20. If you are not setting the flag, no action is necessary. ([#88290](#), [@talclair](#)) [SIG API Machinery and Node]

- Yes.

Feature Name: Support using network resources (VNet, LB, IP, etc.) in different AAD Tenant and Subscription than those for the cluster.

Changes in Pull Request:

1. Add properties `networkResourceTenantID` and `networkResourceSubscriptionID` in cloud provider auth config section, which indicates the location of network resources.
2. Add function `GetMultiTenantServicePrincipalToken` to fetch multi-tenant service principal token, which will be used by Azure VM/VMSS Clients in this feature.
3. Add function `GetNetworkResourceServicePrincipalToken` to fetch network resource service principal token, which will be used by Azure Network Resource (Load Balancer, Public IP, Route Table, Network Security Group and their sub level resources) Clients in this feature.
4. Related unit tests.

None.

User Documentation: In PR <https://github.com/kubernetes-sigs/cloud-provider-azure/pull/301> ([#88384](#), [@bowen5](#)) [SIG Cloud Provider]

Changes by Kind

Deprecation

- Azure service annotation `service.beta.kubernetes.io/azure-load-balancer-disable-tcp-reset` has been deprecated. Its support would be removed in a future release. ([#88462](#), [@feiskyer](#)) [SIG Cloud Provider]

API Change

- API additions to apiserver types ([#87179](#), [@Jefftree](#)) [SIG API Machinery, Cloud Provider and Cluster Lifecycle]
- Add Scheduling Profiles to `kubescheduler.config.k8s.io/v1alpha2` ([#88087](#), [@alculquicondor](#)) [SIG Scheduling and Testing]
- Added support for multiple sizes huge pages on a container level ([#84051](#), [@bart0sh](#)) [SIG Apps, Node and Storage]
- `AppProtocol` is a new field on Service and Endpoints resources, enabled with the `ServiceAppProtocol` feature gate. ([#88503](#), [@roboscott](#)) [SIG Apps and Network]
- Fixed missing validation of uniqueness of list items in lists with `x-kubernetes-list-type: map` or `x-kubernetes-list-type: set`` in CustomResources. ([#84920](#), [@sttts](#)) [SIG API Machinery]
- Introduces optional `--detect-local` flag to kube-proxy. Currently the only supported value is "cluster-cidr", which is the default if not specified. ([#87748](#), [@satyasm](#)) [SIG Cluster Lifecycle, Network and Scheduling]
- Kube-scheduler can run more than one scheduling profile. Given a pod, the profile is selected by using its `.spec.SchedulerName`. ([#88285](#), [@alculquicondor](#)) [SIG Apps, Scheduling and Testing]
- Moving Windows `RunAsUserName` feature to GA ([#87790](#), [@marosset](#)) [SIG Apps and Windows]

Feature

- Add `--dry-run` to `kubectl delete`, `taint`, `replace` ([#88292](#), [@julianvmodesto](#)) [SIG CLI and Testing]
- Add huge page stats to Allocated resources in "kubectl describe node" ([#80605](#), [@odinuge](#)) [SIG CLI]
- Kubeadm: The `ClusterStatus` struct present in the `kubeadm-config ConfigMap` is deprecated and will be removed on a future version. It is going to be maintained by kubeadm until it gets removed. The same information can be found on `etcd` and `kube-apiserver` pod annotations, `kubeadm.kubernetes.io/etcd.advertise-client-urls` and `kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint` respectively. ([#87656](#), [@ereslibre](#)) [SIG Cluster Lifecycle]
- Kubeadm: add the experimental feature gate `PublicKeysECDSA` that can be used to create a cluster with ECDSA certificates from "kubeadm init". Renewal of existing ECDSA certificates is also supported using

- "kubeadm alpha certs renew", but not switching between the RSA and ECDSA algorithms on the fly or during upgrades. ([#86953](#), [@rojkov](#)) [SIG API Machinery, Auth and Cluster Lifecycle]
- Kubeadm: on kubeconfig certificate renewal, keep the embedded CA in sync with the one on disk ([#88052](#), [@neolit123](#)) [SIG Cluster Lifecycle]
- Kubeadm: upgrade supports fallback to the nearest known etcd version if an unknown k8s version is passed ([#88373](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
- New flag `--show-hidden-metrics-for-version` in kube-scheduler can be used to show all hidden metrics that deprecated in the previous minor release. ([#84913](#), [@serathius](#)) [SIG Instrumentation and Scheduling]
- Scheduler framework permit plugins now run at the end of the scheduling cycle, after reserve plugins. Waiting on permit will remain in the beginning of the binding cycle. ([#88199](#), [@mateuszlitwin](#)) [SIG Scheduling]
- The kubelet and the default docker runtime now support running ephemeral containers in the Linux process namespace of a target container. Other container runtimes must implement this feature before it will be available in that runtime. ([#84731](#), [@verb](#)) [SIG Node]

Other (Bug, Cleanup or Flake)

- Add delays between goroutines for vm instance update ([#88094](#), [@aramase](#)) [SIG Cloud Provider]
- Add init containers log to cluster dump info. ([#88324](#), [@zhouya0](#)) [SIG CLI]
- CPU limits are now respected for Windows containers. If a node is over-provisioned, no weighting is used - only limits are respected. ([#86101](#), [@PatrickLang](#)) [SIG Node, Testing and Windows]
- Cloud provider config `CloudProviderBackoffMode` has been removed since it won't be used anymore. ([#88463](#), [@feiskyer](#)) [SIG Cloud Provider]
- Evictions due to pods breaching their ephemeral storage limits are now recorded by the kubelet `evictions` metric and can be alerted on. ([#87906](#), [@smarterclayton](#)) [SIG Node]
- Fix: add remediation in azure disk attach/detach ([#88444](#), [@andyzhangx](#)) [SIG Cloud Provider]
- Fix: check disk status before disk azure disk ([#88360](#), [@andyzhangx](#)) [SIG Cloud Provider]
- Fixed cleaning of CSI raw block volumes. ([#87978](#), [@jsafrane](#)) [SIG Storage]
- Get-kube.sh uses the gcloud's current local GCP service account for auth when the provider is GCE or GKE instead of the metadata server default ([#88383](#), [@BenTheElder](#)) [SIG Cluster Lifecycle]
- Golang/x/net has been updated to bring in fixes for CVE-2020-9283 ([#88381](#), [@BenTheElder](#)) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle and Instrumentation]
- Kubeadm now includes CoreDNS version 1.6.7 ([#86260](#), [@rajansandeep](#)) [SIG Cluster Lifecycle]
- Kubeadm: fix the bug that 'kubeadm upgrade' hangs in single node cluster ([#88434](#), [@SataQiu](#)) [SIG Cluster Lifecycle]

- Optimize kubectl version help info ([#88313](#), [@zhouya0](#)) [SIG CLI]
- Removes the deprecated command kubectl rolling-update ([#88057](#), [@julianvmodesto](#)) [SIG Architecture, CLI and Testing]

v1.18.0-alpha.5

[Documentation](#)

Downloads for v1.18.0-alpha.5

filename	sha512 hash
kubernetes.tar.gz	6452cac2b80721e9f577cb117c29b9ac6858812b4275c2becbf74312566f7d
kubernetes-src.tar.gz	e41d9d4dd6910a42990051fcdca4bf5d3999df46375abd27ffc56aae9b455a

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	5c95935863492b31d4aaa6be93260088dafa27663eb91edca980ca3a84
kubernetes-client-darwin-amd64.tar.gz	868faa578b3738604d8be62fae599ccc556799f1ce54807f1fe72599f20f8a
kubernetes-client-linux-386.tar.gz	76a89d1d30b476b47f8fb808e342f89608e5c1c1787c4c06f2d7e763f9482
kubernetes-client-linux-amd64.tar.gz	07ad96a09b44d1c707d7c68312c5d69b101a3424bf1e6e9400b2e7a3fba
kubernetes-client-linux-arm.tar.gz	c04fed9fa370a75c1b8e18b2be0821943bb9befcc784d14762ea3278e736
kubernetes-client-linux-arm64.tar.gz	4199147dea9954333df26d34248a1cb7b02ebbd6380ffcd42d9f9ed5fdab
kubernetes-client-linux-ppc64le.tar.gz	4f6d4d61d1c52d3253ca19031ebcd4bad06d19b68bbaaab5c8e8c590774
kubernetes-client-linux-s390x.tar.gz	e2a454151ae5dd891230fb516a3f73f73ab97832db66fd3d12e7f1657a56
kubernetes-client-windows-386.tar.gz	14b262ba3b71c41f545db2a017cf1746075ada5745a858d2a62bc9df7c5d
kubernetes-client-windows-amd64.tar.gz	26353c294755a917216664364b524982b7f5fc6aa832ce90134bb178df8a

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	ba77e0e7c610f59647c1b2601f82752964a0f54b7ad609a89b00fcfd553d0f0249
kubernetes-server-linux-arm.tar.gz	45e87b3e844ea26958b0b489e8c9b90900a3253000850f5ff9e87ffdcafb72ab8
kubernetes-server-linux-arm64.tar.gz	155e136e3124ead69c594eead3398d6cfdabb8f823c324880e8a7bbd1b570b05c
kubernetes-server-linux-ppc64le.tar.gz	3fa0fb8221da19ad9d03278961172b7fa29a618b30abfa55e7243bb937dede8d
kubernetes-server-linux-s390x.tar.gz	db3199c3d7ba0b326d71dc8b80f50b195e79e662f71386a3b2976d47d13d7b0

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	addcdfbad7f12647e6babb8eadf853a374605c8f18bf63f416fa4d3bf1b903aa20
kubernetes-node-linux-arm.tar.gz	b2ac54e0396e153523d116a2aaa32c919d6243931e0104cd47a23f546d710e7a
kubernetes-node-linux-arm64.tar.gz	7aab36f2735cba805e4fd109831a1af0f586a88db3f07581b6dc2a2aab90076b2
kubernetes-node-linux-ppc64le.tar.gz	a579936f07ebf86f69f297ac50ba4c34caf2c0b903f73190eb581c78382b05ef36c
kubernetes-node-linux-s390x.tar.gz	58fa0359ddd48835192fab1136a2b9b45d1927b04411502c269cda07cb8a8106
kubernetes-node-windows-amd64.tar.gz	9086c03cd92b440686cea6d8c4e48045cc46a43ab92ae0e70350b3f51804b9e2

Changelog since v1.18.0-alpha.3

Deprecation

- Kubeadm: command line option "kubelet-version" for kubeadm upgrade node has been deprecated and will be removed in a future release. ([#87942](#), [@SataQiu](#)) [SIG Cluster Lifecycle]

API Change

- Kubelet podresources API now provides the information about active pods only. ([#79409](#), [@takmatsu](#)) [SIG Node]
- Remove deprecated fields from .leaderElection in kubescheduler.config.k8s.io/v1alpha2 ([#87904](#), [@alculquicondor](#)) [SIG Scheduling]
- Signatures on generated clientset methods have been modified to accept context.Context as a first argument. Signatures of generated Create, Update, and Patch methods have been updated to accept CreateOptions, UpdateOptions and PatchOptions respectively. Clientsets that with the previous interface have been added in new "deprecated" packages to allow incremental migration to the new APIs. The deprecated packages will be removed in the 1.21 release. ([#87299](#), [@mikedanese](#)) [SIG API Machinery, Apps, Auth, Autoscaling, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation, Network, Node, Scheduling, Storage, Testing and Windows]
- The k8s.io/node-api component is no longer updated. Instead, use the RuntimeClass types located within k8s.io/api, and the generated clients located within k8s.io/client-go ([#87503](#), [@liggitt](#)) [SIG Node and Release]

Feature

- Add indexer for storage cacher ([#85445](#), [@shaloulcy](#)) [SIG API Machinery]
- Add support for mount options to the FC volume plugin ([#87499](#), [@ejweber](#)) [SIG Storage]
- Added a config-mode flag in azure auth module to enable getting AAD token without spn: prefix in audience claim. When it's not specified, the default behavior doesn't change. ([#87630](#), [@weinong](#)) [SIG API Machinery, Auth, CLI and Cloud Provider]
- Introduced BackoffManager interface for backoff management ([#87829](#), [@zhan849](#)) [SIG API Machinery]
- PodTopologySpread plugin now excludes terminatingPods when making scheduling decisions. ([#87845](#), [@Huang-Wei](#)) [SIG Scheduling]
- Promote CSIMigrationOpenStack to Beta (off by default since it requires installation of the OpenStack Cinder CSI Driver) The in-tree AWS OpenStack Cinder "kubernetes.io/cinder" was already deprecated a while ago and will be removed in 1.20. Users should enable CSIMigration + CSIMigrationOpenStack features and install the OpenStack Cinder CSI Driver (<https://github.com/kubernetes-sigs/>)

[cloud-provider-openstack](#)) to avoid disruption to existing Pod and PVC objects at that time. Users should start using the OpenStack Cinder CSI Driver directly for any new volumes. ([#85637](#), [@dims](#)) [SIG Cloud Provider]

Design

- The scheduler Permit extension point doesn't return a boolean value in its Allow() and Reject() functions. ([#87936](#), [@Huang-Wei](#)) [SIG Scheduling]

Other (Bug, Cleanup or Flake)

- Adds "volume.beta.kubernetes.io/migrated-to" annotation to PV's and PVC's when they are migrated to signal external provisioners to pick up those objects for Provisioning and Deleting. ([#87098](#), [@davidz627](#)) [SIG Apps and Storage]
- Fix a bug in the dual-stack IPVS proxier where stale IPv6 endpoints were not being cleaned up ([#87695](#), [@andrewsykim](#)) [SIG Network]
- Fix kubectl drain ignore daemonsets and others. ([#87361](#), [@zhouya0](#)) [SIG CLI]
- Fix: add azure disk migration support for CSINode ([#88014](#), [@andyzhangx](#)) [SIG Cloud Provider and Storage]
- Fix: add non-retriable errors in azure clients ([#87941](#), [@andyzhangx](#)) [SIG Cloud Provider]
- Fixed NetworkPolicy validation that Except values are accepted when they are outside the CIDR range. ([#86578](#), [@tnqn](#)) [SIG Network]
- Improves performance of the node authorizer ([#87696](#), [@liggitt](#)) [SIG Auth]
- Iptables/userspace proxy: improve performance by getting local addresses only once per sync loop, instead of for every external IP ([#85617](#), [@andrewsykim](#)) [SIG API Machinery, CLI, Cloud Provider, Cluster Lifecycle, Instrumentation and Network]
- Kube-aggregator: always sets unavailableGauge metric to reflect the current state of a service. ([#87778](#), [@p0lyn0mial](#)) [SIG API Machinery]
- Kubeadm allows to configure single-stack clusters if dual-stack is enabled ([#87453](#), [@aojea](#)) [SIG API Machinery, Cluster Lifecycle and Network]
- Kubeadm: 'kubeadm alpha kubelet config download' has been removed, please use 'kubeadm upgrade node phase kubelet-config' instead ([#87944](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
- Kubeadm: remove 'kubeadm upgrade node config' command since it was deprecated in v1.15, please use 'kubeadm upgrade node phase kubelet-config' instead ([#87975](#), [@SataQiu](#)) [SIG Cluster Lifecycle]
- Kubectl describe and kubectl top pod will return a message saying "No resources found" or "No resources found in namespace" if there are no results to display. ([#87527](#), [@brianpursley](#)) [SIG CLI]
- Kubelet metrics gathered through metrics-server or prometheus should no longer timeout for Windows nodes running more than 3 pods. ([#87730](#), [@marosset](#)) [SIG Node, Testing and Windows]

- Kubelet metrics have been changed to buckets. For example the `exec/{podNamespace}/{podID}/{containerName}` is now just `exec`. ([#87913](#), [@cheftako](#)) [SIG Node]
- Limit number of instances in a single update to GCE target pool to 1000. ([#87881](#), [@wojtek-t](#)) [SIG Cloud Provider, Network and Scalability]
- Make Azure clients only retry on specified HTTP status codes ([#88017](#), [@feiskyer](#)) [SIG Cloud Provider]
- Pause image contains "Architecture" in non-amd64 images ([#87954](#), [@BenTheElder](#)) [SIG Release]
- Pods that are considered for preemption and haven't started don't produce an error log. ([#87900](#), [@alculquicondor](#)) [SIG Scheduling]
- Prevent error message from being displayed when running `kubectl plugin list` and your path includes an empty string ([#87633](#), [@brianpursley](#)) [SIG CLI]
- `kubectl create clusterrolebinding` creates `rbac.authorization.k8s.io/v1` object ([#85889](#), [@oke-py](#)) [SIG CLI]

v1.18.0-alpha.4

[Documentation](#)

Important note about manual tag

Due to a [tagging bug in our Release Engineering tooling](#) during v1.18.0-alpha.3, we needed to push a manual tag (v1.18.0-alpha.4).

No binaries have been produced or will be provided for v1.18.0-alpha.4.

The changelog for v1.18.0-alpha.4 is included as part of the [changelog since v1.18.0-alpha.3][`#changelog-since-v1.18.0-alpha.3`] section.

v1.18.0-alpha.3

[Documentation](#)

Downloads for v1.18.0-alpha.3

filename	sha512 hash
kubernetes.tar.gz	60bf3bfc23b428f53fd853bac18a4a905b980fcc0bacd35ccd6357a89cfc26e4
kubernetes-src.tar.gz	8adf1016565a7c93713ab6fa4293c2d13b4f6e4e1ec4dcba60bd71e218b4db

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	abb32e894e8280c772e96227b574da81cd1eac374b8d29158b7f222ed55
kubernetes-client-darwin-amd64.tar.gz	5e4b1a993264e256ec1656305de7c306094cae9781af8f1382df4ce4eed4
kubernetes-client-linux-386.tar.gz	68da39c2ae101d2b38f6137ceda07eb0c2124794982a62ef483245dbffb0
kubernetes-client-linux-amd64.tar.gz	dc236ffa8ad426620e50181419e9bebe3c161e953dbfb8a019f61b11286e
kubernetes-client-linux-arm.tar.gz	ab0a8bd6dc31ea160b731593cdc490b3cc03668b1141cf95310bd7060dc
kubernetes-client-linux-arm64.tar.gz	159ea083c601710d0d6aea423eeb346c99ffaf2abd137d35a53e87a07f5ca
kubernetes-client-linux-ppc64le.tar.gz	16b0459adfa26575d13be49ab53ac7f0ffd05e184e4e13d2dfbfe725d46bb
kubernetes-client-linux-s390x.tar.gz	d5aa1f5d89168995d2797eb839a04ce32560f405b38c1c0baaa0e313e477
kubernetes-client-windows-386.tar.gz	374e16a1e52009be88c94786f80174d82dff66399bf294c9bee18a2159c4
kubernetes-client-windows-amd64.tar.gz	5a94c1068c19271f810b994adad8e62fae03b3d4473c7c9e6d056995ff775

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	a677bec81f0eba75114b92ff955bac74512b47e53959d56a685dae5edd527283
kubernetes-server-linux-arm.tar.gz	2fb696f86ff13ebef5f3cf2b254bf41303644c5ea84a292782eac6123550702655
kubernetes-server-linux-arm64.tar.gz	738e95da9cfb8f1309479078098de1c38cef5e1dd5ee1129b77651a936a412b7
kubernetes-server-linux-ppc64le.tar.gz	7a85bfcbb2aa636df60c41879e96e788742ecd72040cb0db2a93418439c12521
kubernetes-server-linux-s390x.tar.gz	1f1cdb2efa3e7cac857203d8845df2fdaa5cf1f20df764efffff29371945ec58f6dee

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	4ccfced3f5ba4adfa58f4a9d1b2c5bdb3e89f9203ab0e27d11eb1c325ac323ebe
kubernetes-node-linux-arm.tar.gz	d695a69d18449062e4c129e54ec8384c573955f8108f4b78adc2ec929719f2196
kubernetes-node-linux-arm64.tar.gz	21df1da88c89000abc22f97e482c3aaa5ce53ec9628d83dda2e04a1d86c4d53b
kubernetes-node-linux-ppc64le.tar.gz	ff77e3aacb6ed9d89baed92ef542c8b5cec83151b6421948583cf608bca3b779d
kubernetes-node-linux-s390x.tar.gz	57d75b7977ec1a0f6e7ed96a304dbb3b8664910f42ca19aab319a9ec33535ff59
kubernetes-node-windows-amd64.tar.gz	63fdbb71773cfd73a914c498e69bb9eea3fc314366c99ffb8bd42ec5b4dae8076

Changelog since v1.18.0-alpha.2

Deprecation

- Remove all the generators from kubectl run. It will now only create pods. Additionally, deprecates all the flags that are not relevant anymore. ([#87077](#), [@solysh](#)) [SIG Architecture, SIG CLI, and SIG Testing]
- kubeadm: kube-dns is deprecated and will not be supported in a future version ([#86574](#), [@SataQiu](#)) [SIG Cluster Lifecycle]

API Change

- Add kubescheduler.config.k8s.io/v1alpha2 ([#87628](#), [@alculquicondor](#)) [SIG Scheduling]
- --enable-cadvisor-endpoints is now disabled by default. If you need access to the cAdvisor v1 Json API please enable it explicitly in the kubelet command line. Please note that this flag was deprecated in 1.15 and will be removed in 1.19. ([#87440](#), [@dims](#)) [SIG Instrumentation, SIG Node, and SIG Testing]
- The following feature gates are removed, because the associated features were unconditionally enabled in previous releases: CustomResourceValidation, CustomResourceSubresources, CustomResourceWebhookConversion, CustomResourcePublishOpenAPI, CustomResourceDefaulting ([#87475](#), [@liggitt](#)) [SIG API Machinery]

Feature

- aggregation api will have alpha support for network proxy ([#87515](#), [@Sh4d1](#)) [SIG API Machinery]
- API request throttling (due to a high rate of requests) is now reported in client-go logs at log level 2. The messages are of the form

Throttling request took 1.50705208s, request: GET:

The presence of these messages, may indicate to the administrator the need to tune the cluster accordingly. ([#87740](#), [@jennybuckley](#)) [SIG API Machinery]

- kubeadm: reject a node joining the cluster if a node with the same name already exists ([#81056](#), [@neolit123](#)) [SIG Cluster Lifecycle]
- disableAvailabilitySetNodes is added to avoid VM list for VMSS clusters. It should only be used when vmType is "vmss" and all the nodes (including masters) are VMSS virtual machines. ([#87685](#), [@feiskyer](#)) [SIG Cloud Provider]
- The kubectl --dry-run flag now accepts the values 'client', 'server', and 'none', to support client-side and server-side dry-run strategies. The boolean and unset values for the --dry-run flag are deprecated and a value will be required in a future version. ([#87580](#), [@julianvmodesto](#)) [SIG CLI]
- Add support for pre-allocated hugepages for more than one page size ([#82820](#), [@odinuge](#)) [SIG Apps]
- Update CNI version to v0.8.5 ([#78819](#), [@justaugustus](#)) [SIG API Machinery, SIG Cluster Lifecycle, SIG Network, SIG Release, and SIG Testing]
- Skip default spreading scoring plugin for pods that define TopologySpreadConstraints ([#87566](#), [@skilxn-go](#)) [SIG Scheduling]
- Added more details to taint toleration errors ([#87250](#), [@starizard](#)) [SIG Apps, and SIG Scheduling]
- Scheduler: Add DefaultBinder plugin ([#87430](#), [@alculquicondor](#)) [SIG Scheduling, and SIG Testing]
- Kube-apiserver metrics will now include request counts, latencies, and response sizes for /healthz, /livez, and /readyz requests. ([#83598](#), [@jktomer](#)) [SIG API Machinery]

Other (Bug, Cleanup or Flake)

- Fix the masters rolling upgrade causing thundering herd of LISTs on etcd leading to control plane unavailability. ([#86430](#), [@wojtek-t](#)) [SIG API Machinery, SIG Node, and SIG Testing]

- kubectl diff now returns 1 only on diff finding changes, and >1 on kubectl errors. The "exit status code 1" message as also been muted. ([#87437](#), [@apelisse](#)) [SIG CLI, and SIG Testing]
- To reduce chances of throttling, VM cache is set to nil when Azure node provisioning state is deleting ([#87635](#), [@feiskyer](#)) [SIG Cloud Provider]
- Fix regression in statefulset conversion which prevented applying a statefulset multiple times. ([#87706](#), [@liggitt](#)) [SIG Apps, and SIG Testing]
- fixed two scheduler metrics (pending_pods and schedule_attempts_total) not being recorded ([#87692](#), [@everpeace](#)) [SIG Scheduling]
- Resolved a performance issue in the node authorizer index maintenance. ([#87693](#), [@liggitt](#)) [SIG Auth]
- Removed the 'client' label from apiserver_request_total. ([#87669](#), [@logicalhan](#)) [SIG API Machinery, and SIG Instrumentation]
- (*"k8s.io/client-go/rest".Request).{Do,DoRaw,Stream,Watch} now require callers to pass a context.Context as an argument. The context is used for timeout and cancellation signaling and to pass supplementary information to round trippers in the wrapped transport chain. If you don't need any of this functionality, it is sufficient to pass a context created with context.Background() to these functions. The (*"k8s.io/client-go/rest".Request).Context method is removed now that all methods that execute a request accept a context directly. ([#87597](#), [@mikedanese](#)) [SIG API Machinery, SIG Apps, SIG Auth, SIG Autoscaling, SIG CLI, SIG Cloud Provider, SIG Cluster Lifecycle, SIG Instrumentation, SIG Network, SIG Node, SIG Scheduling, SIG Storage, and SIG Testing]
- For volumes that allow attaches across multiple nodes, attach and detach operations across different nodes are now executed in parallel. ([#87258](#), [@verult](#)) [SIG Apps, SIG Node, and SIG Storage]
- kubeadm: apply further improvements to the tentative support for concurrent etcd member join. Fixes a bug where multiple members can receive the same hostname. Increase the etcd client dial timeout and retry timeout for add/remove/... operations. ([#87505](#), [@neolit123](#)) [SIG Cluster Lifecycle]
- Reverted a kubectl azure auth module change where oidc claim spn: prefix was omitted resulting a breaking behavior with existing Azure AD OIDC enabled api-server ([#87507](#), [@weinong](#)) [SIG API Machinery, SIG Auth, and SIG Cloud Provider]
- Update cri-tools to v1.17.0 ([#86305](#), [@saschagrunert](#)) [SIG Cluster Lifecycle, and SIG Release]
- kubeadm: remove the deprecated CoreDNS feature-gate. It was set to "true" since v1.11 when the feature went GA. In v1.13 it was marked as deprecated and hidden from the CLI. ([#87400](#), [@neolit123](#)) [SIG Cluster Lifecycle]
- Shared informers are now more reliable in the face of network disruption. ([#86015](#), [@squeed](#)) [SIG API Machinery]
- the CSR signing cert/key pairs will be reloaded from disk like the kube-apiserver cert/key pairs ([#86816](#), [@deads2k](#)) [SIG API Machinery, SIG Apps, and SIG Auth]
- "kubectl describe statefulsets.apps" prints garbage for rolling update partition ([#85846](#), [@phil9909](#)) [SIG CLI]

v1.18.0-alpha.2

[Documentation](#)

Downloads for v1.18.0-alpha.2

filename	sha512 hash
kubernetes.tar.gz	7af83386b4b35353f0aa1bdaf73599eb08b1d1ca11ecc2c606854aff754db69
kubernetes-src.tar.gz	a14b02a0a0bde97795a836a8f5897b0ee6b43e010e13e43dd4cca80a5b962

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	427f214d47ded44519007de2ae87160c56c2920358130e474b768299751
kubernetes-client-darwin-amd64.tar.gz	861fd81ac3bd45765575bedf5e002a2294aba48ef9e15980fc7d6783985f7
kubernetes-client-linux-386.tar.gz	7d59b05d6247e2606a8321c72cd239713373d876dbb43b0fb7f1cb857fa
kubernetes-client-linux-amd64.tar.gz	7cdefb4e32bad9d2df5bb8e7e0a6f4dab2ae6b7afef5d801ac5c342d4effd
kubernetes-client-linux-arm.tar.gz	6212bbf0fa1d01ced77dcca2c4b76b73956cd3c6b70e0701c1fe0df5ff371
kubernetes-client-linux-arm64.tar.gz	1f0d9990700510165ee471acb2f88222f1b80e8f6deb351ce14cf50a70a98
kubernetes-client-linux-ppc64le.tar.gz	77e00ba12a32db81e96f8de84609de93f32c61bb3f53875a57496d213aa6
kubernetes-client-linux-s390x.tar.gz	a39ec2044bed5a4570e9c83068e0fc0ce923ccffa44380f8bbc3247426bea
kubernetes-client-windows-386.tar.gz	1a0ab88f9b7e34b60ab31d5538e97202a256ad8b7b7ed5070cae5f2f12d5
kubernetes-client-windows-amd64.tar.gz	1966eb5dfb78c1bc33aaa6389f32512e3aa92584250a0164182f3566c81d

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	f814d6a3872e4572aa4da297c29def4c1fad8eba0903946780b6bf9788c72b99d

filename	sha512 hash
kubernetes-server-linux-arm.tar.gz	56aa08225e546c92c2ff88ac57d3db7dd5e63640772ea72a429f080f706982713
kubernetes-server-linux-arm64.tar.gz	fb87128d905211ba097aa860244a376575ae2edbaca6e51402a24bc2964854b9
kubernetes-server-linux-ppc64le.tar.gz	6d21fbf39b9d3a0df9642407d6f698fabdc809aca83af197bceb58a81b2584607
kubernetes-server-linux-s390x.tar.gz	ddcda4dc360ca97705f71bf2a18ddacd7b7ddf77535b62e699e97a1b2dd24843

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	78915a9bde35c70c67014f0cea8754849db4f6a84491a3ad9678fd3bc0203e43a
kubernetes-node-linux-arm.tar.gz	3218e811abcb0cb09d80742def339be3916db5e9bbc62c0dc8e6d87085f7e3d9
kubernetes-node-linux-arm64.tar.gz	fa22de9c4440b8fb27f4e77a5a63c5e1c8aa8aa30bb79eda843b0f40498c21b8c
kubernetes-node-linux-ppc64le.tar.gz	bbda9b5cc66e8f13d235703b2a85e2c4f02fa16af047be4d27a3e198e11eb1170
kubernetes-node-linux-s390x.tar.gz	b2ed1eda013069adce2aac00b86d75b84e006cfce9bafac0b5a2bafcb60f8f2cb3
kubernetes-node-windows-amd64.tar.gz	bd8eb23dba711f31b5148257076b1bbe9629f2a75de213b2c779bd5b29279e9

Changelog since v1.18.0-alpha.1

Other notable changes

- Bump golang/mock version to v1.3.1 ([#87326](#), [@wawa0210](#))
- fix a bug that orphan revision cannot be adopted and statefulset cannot be synced ([#86801](#), [@likakuli](#))
- Azure storage clients now suppress requests on throttling ([#87306](#), [@feiskyer](#))
- Introduce Alpha field Immutable in both Secret and ConfigMap objects to mark their contents as immutable. The implementation is hidden

- behind feature gate `ImmutableEphemeralVolumes` (currently in Alpha stage). ([#86377](#), [@wojtekt](#))
- `EndpointSlices` will now be enabled by default. A new `EndpointSliceProxy` feature gate determines if kube-proxy will use `EndpointSlices`, this is disabled by default. ([#86137](#), [@roboscott](#))
 - `kubeadm` upgrades always persist the `etcd` backup for stacked ([#86861](#), [@SataQiu](#))
 - Fix the bug PIP's DNS is deleted if no DNS label service annotation isn't set. ([#87246](#), [@nilo19](#))
 - New flag `--show-hidden-metrics-for-version` in `kube-controller-manager` can be used to show all hidden metrics that deprecated in the previous minor release. ([#85281](#), [@RainbowMango](#))
 - Azure network and VM clients now suppress requests on throttling ([#87122](#), [@feiskyer](#))
 - `kubectl apply -f <file> --prune -n <namespace>` should prune all resources not defined in the file in the cli specified namespace. ([#85613](#), [@MartinKaburu](#))
 - Fixes service account token admission error in clusters that do not run the service account token controller ([#87029](#), [@liggitt](#))
 - `CustomResourceDefinition` status fields are no longer required for client validation when submitting manifests. ([#87213](#), [@hasheddan](#))
 - All apiservers log request lines in a more greppable format. ([#87203](#), [@lavalamp](#))
 - `provider/azure`: Network security groups can now be in a separate resource group. ([#87035](#), [@CecileRobertMichon](#))
 - Cleaned up the output from `kubectl describe CSINode <name>`. ([#85283](#), [@huffmanca](#))
 - Fixed the following ([#84265](#), [@bhagwat070919](#))
 - ■ AWS Cloud Provider attempts to delete `LoadBalancer` security group it didn't provision
 - ■ AWS Cloud Provider creates default `LoadBalancer` security group even if annotation `[service.beta.kubernetes.io/aws-load-balancer-security-groups]` is present
 - `kubelet`: resource metrics endpoint `/metrics/resource/v1alpha1` as well as all metrics under this endpoint have been deprecated. ([#86282](#), [@RainbowMango](#))
 - Please convert to the following metrics emitted by endpoint `/metrics/resource`:
 - ■ `scrape_error --> scrape_error`
 - ■ `node_cpu_usage_seconds_total --> node_cpu_usage_seconds`
 - ■ `node_memory_working_set_bytes --> node_memory_working_set_bytes`
 - ■ `container_cpu_usage_seconds_total --> container_cpu_usage_seconds`
 - ■ `container_memory_working_set_bytes --> container_memory_working_set_bytes`
 - ■ `scrape_error --> scrape_error`
 - You can now pass `--node-ip ::` to `kubelet` to indicate that it should autodetect an IPv6 address to use as the node's primary address. ([#85850](#), [@danwinship](#))

- kubeadm: support automatic retry after failing to pull image ([#86899](#), [@SataQiu](#))
- TODO ([#87044](#), [@jennybuckley](#))
- Improved yaml parsing performance ([#85458](#), [@cjcullen](#))
- Fixed a bug which could prevent a provider ID from ever being set for node if an error occurred determining the provider ID when the node was added. ([#87043](#), [@zjs](#))
- fix a regression in kubenet that prevent pods to obtain ip addresses ([#85993](#), [@chendotjs](#))
- Bind kube-dns containers to linux nodes to avoid Windows scheduling ([#83358](#), [@wawa0210](#))
- The following features are unconditionally enabled and the corresponding --feature-gates flags have been removed: PodPriority, TaintNodesByCondition, ResourceQuotaScopeSelectors and ScheduleDaemonSetPods ([#86210](#), [@draveness](#))
- Bind dns-horizontal containers to linux nodes to avoid Windows scheduling on kubernetes cluster includes linux nodes and windows nodes ([#83364](#), [@wawa0210](#))
- fix kubectl annotate error when local=true is set ([#86952](#), [@zhouya0](#))
- Bug fixes: ([#84163](#), [@david-tigera](#))
 - Make sure we include latest packages node #351 ([@caseydavenport](#))
- fix kubectl apply set-last-applied namespaces error ([#86474](#), [@zhouya0](#))
- Add VolumeBinder method to FrameworkHandle interface, which allows user to get the volume binder when implementing scheduler framework plugins. ([#86940](#), [@skilxn-go](#))
- elasticsearch supports automatically setting the advertise address ([#85944](#), [@SataQiu](#))
- If a serving certificates param specifies a name that is an IP for an SNI certificate, it will have priority for replying to server connections. ([#85308](#), [@deads2k](#))
- kube-proxy: Added dual-stack IPv4/IPv6 support to the iptables proxier. ([#82462](#), [@vllry](#))
- Azure VMSS/VMSSVM clients now suppress requests on throttling ([#86740](#), [@feiskyer](#))
- New metric kubelet_pleg_last_seen_seconds to aid diagnosis of PLEG not healthy issues. ([#86251](#), [@bboreham](#))
- For subprotocol negotiation, both client and server protocol is required now. ([#86646](#), [@tedyu](#))
- kubeadm: use bind-address option to configure the kube-controller-manager and kube-scheduler http probes ([#86493](#), [@aojea](#))
- Marked scheduler's metrics scheduling_algorithm_predicate_evaluation_seconds and ([#86584](#), [@xiaoanyunfei](#))
 - scheduling_algorithm_priority_evaluation_seconds as deprecated. Those are replaced by framework_extension_point_duration_seconds[extension_point="Filter"] and framework_extension_point_duration_seconds[extension_point="Score"] respectively.

- Marked scheduler's `scheduling_duration_seconds` Summary metric as deprecated ([#86586](#), [@xiaoanyunfei](#))
- Add instructions about how to bring up e2e test cluster ([#85836](#), [@YangLu1031](#))
- If a required flag is not provided to a command, the user will only see the required flag error message, instead of the entire usage menu. ([#86693](#), [@sallyom](#))
- kubeadm: tolerate whitespace when validating certificate authority PEM data in kubeconfig files ([#86705](#), [@neolit123](#))
- kubeadm: add support for the "ci/k8s-master" version label as a replacement for "ci-cross/*", which no longer exists. ([#86609](#), [@Pensu](#))
- Fix EndpointSlice controller race condition and ensure that it handles external changes to EndpointSlices. ([#85703](#), [@roboscott](#))
- Fix nil pointer dereference in azure cloud provider ([#85975](#), [@ldx](#))
- fix: azure disk could not mounted on Standard_DC4s/DC2s instances ([#86612](#), [@andyzhangx](#))
- Fixes v1.17.0 regression in `--service-cluster-ip-range` handling with IPv4 ranges larger than 65536 IP addresses ([#86534](#), [@liggitt](#))
- Adds back support for `AlwaysCheckAllPredicates` flag. ([#86496](#), [@ahg-g](#))
- Azure global rate limit is switched to per-client. A set of new rate limit configure options are introduced, including `routeRateLimit`, `SubnetsRateLimit`, `InterfaceRateLimit`, `RouteTableRateLimit`, `LoadBalancerRateLimit`, `PublicIPAddressRateLimit`, `SecurityGroupRateLimit`, `VirtualMachineRateLimit`, `StorageAccountRateLimit`, `DiskRateLimit`, `SnapshotRateLimit`, `VirtualMachineScaleSetRateLimit` and `VirtualMachineSizeRateLimit`. ([#86515](#), [@feiskyer](#))
 - The original rate limit options would be default values for those new client's rate limiter.
- Fix issue [#85805](#) about resource not found in azure cloud provider when lb specified in other resource group. ([#86502](#), [@levimm](#))
- `AlwaysCheckAllPredicates` is deprecated in scheduler Policy API. ([#86369](#), [@Huang-Wei](#))
- Kubernetes KMS provider for data encryption now supports disabling the in-memory data encryption key (DEK) cache by setting `cachesize` to a negative value. ([#86294](#), [@enj](#))
- option `preConfiguredBackendPoolLoadBalancerTypes` is added to azure cloud provider for the pre-configured load balancers, possible values: `"", "internal", "external", "all"` ([#86338](#), [@gossion](#))
- Promote `StartupProbe` to beta for 1.18 release ([#83437](#), [@matthyx](#))
- Fixes issue where AAD token obtained by `kubect`l is incompatible with on-behalf-of flow and oidc. ([#86412](#), [@weinong](#))
 - The audience claim before this fix has "spn:" prefix. After this fix, "spn:" prefix is omitted.
- change `CounterVec` to `Counter` about `PLEGDiscardEvent` ([#86167](#), [@yiyang5055](#))
- hollow-node do not use remote CRI anymore ([#86425](#), [@jkaniuk](#))
- hollow-node use fake CRI ([#85879](#), [@gongguan](#))

v1.18.0-alpha.1

[Documentation](#)

Downloads for v1.18.0-alpha.1

filename	sha512 hash
kubernetes.tar.gz	0c4904efc7f4f1436119c91dc1b6c93b3bd9c7490362a394bff10099c18e1e7
kubernetes-src.tar.gz	0a50fc6816c730ca5ae4c4f26d5ad7b049607d29f6a782a4e5b4b05ac50e01

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	c6d75f7f3f20bef17fc7564a619b54e6f4a673d041b7c9ec93663763a1cc8c
kubernetes-client-darwin-amd64.tar.gz	ca1f19db289933beace6daee6fc30af19b0e260634ef6e89f773464a05e24
kubernetes-client-linux-386.tar.gz	af2e673653eb39c3f24a54efc68e1055f9258bdf6cf8fea42faf42c05abefc2c
kubernetes-client-linux-amd64.tar.gz	9009032c3f94ac8a78c1322a28e16644ce3b20989eb762685a1819148aec
kubernetes-client-linux-arm.tar.gz	afba9595b37a3f2eead6e3418573f7ce093b55467dce4da0b8de86002857
kubernetes-client-linux-arm64.tar.gz	04fc3b2fe3f271807f0bc6c61be52456f26a1af904964400be819b7914519
kubernetes-client-linux-ppc64le.tar.gz	04c7edab874b33175ff7bebfff5b3a032bc6eb088fcd7387ffcd5b3fa71395
kubernetes-client-linux-s390x.tar.gz	499287dbbc33399a37b9f3b35e0124ff20b17b6619f25a207ee9c606ef263
kubernetes-client-windows-386.tar.gz	cf84aeddff00f126fb13c0436b116dd0464a625659e44c84bf863517db0400
kubernetes-client-windows-amd64.tar.gz	69f20558ccd5cd6dbaccf29307210db4e687af21f6d71f68c69d3a3976686

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	3f29df2ce904a0f10db4c1d7a425a36f420867b595da3fa158ae430bfead90def2

filename	sha512 hash
kubernetes-server-linux-arm.tar.gz	4a21073b2273d721fbf062c254840be5c8471a010bcc0c731b101729e36e61f63
kubernetes-server-linux-arm64.tar.gz	7f1cb6d721bedc90e28b16f99bea7e59f5ad6267c31ef39c14d34db6ad6aad87e
kubernetes-server-linux-ppc64le.tar.gz	8f2b552030b5274b1c2c7c166eacd5a14b0c6ca0f23042f4c52efe87e22a167ba4
kubernetes-server-linux-s390x.tar.gz	8d9f2c96f66edafb7c8b3aa90960d29b41471743842aede6b47b3b2e61f4306fb

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	84194cb081d1502f8ca68143569f9707d96f1a28fcf0c574ebd203321463a8b60
kubernetes-node-linux-arm.tar.gz	0091e108ab94fd8683b89c597c4fdc2fbf4920b007cfd5297072c44bc3a230dfe
kubernetes-node-linux-arm64.tar.gz	b7e85682cc2848a35d52fd6f01c247f039ee1b5dd03345713821ea10a7fa9939b
kubernetes-node-linux-ppc64le.tar.gz	cd1f0849e9c62b5d2c93ff0cebf58843e178d8a88317f45f76de0db5ae020b8027
kubernetes-node-linux-s390x.tar.gz	e1e697a34424c75d75415b613b81c8af5f64384226c5152d869f12fd7db1a3e25
kubernetes-node-windows-amd64.tar.gz	c725a19a4013c74e22383ad3fb4cb799b3e161c4318fdad066daf806730a89bc3

Changelog since v1.17.0

Action Required

- action required ([#85363](#), [@immutableT](#))
 - 1. Currently, if users were to explicitly specify CacheSize of 0 for KMS provider, they would end-up with a provider that caches up to 1000 keys. This PR changes this behavior.
 - Post this PR, when users supply 0 for CacheSize this will result in a validation error.

- 1. CacheSize type was changed from int32 to *int32. This allows defaulting logic to differentiate between cases where users explicitly supplied 0 vs. not supplied any value.
- 1. KMS Provider's endpoint (path to Unix socket) is now validated when the EncryptionConfiguration file is loaded. This used to be handled by the GRPCService.

Other notable changes

- fix: azure data disk should use same key as os disk by default ([#86351](#), [@andyzhangx](#))
- New flag --show-hidden-metrics-for-version in kube-proxy can be used to show all hidden metrics that deprecated in the previous minor release. ([#85279](#), [@RainbowMango](#))
- Remove cluster-monitoring addon ([#85512](#), [@serathius](#))
- Changed core_pattern on COS nodes to be an absolute path. ([#86329](#), [@mml](#))
- Track mount operations as uncertain if operation fails with non-final error ([#82492](#), [@gnufied](#))
- add kube-proxy flags --ipvs-tcp-timeout, --ipvs-tcpfin-timeout, --ipvs-udp-timeout to configure IPVS connection timeouts. ([#85517](#), [@andrewsykim](#))
- The sample-apiserver aggregated conformance test has updated to use the Kubernetes v1.17.0 sample apiserver ([#84735](#), [@liggitt](#))
- The underlying format of the CPUManager state file has changed. Upgrades should be seamless, but any third-party tools that rely on reading the previous format need to be updated. ([#84462](#), [@klueska](#))
- kubernetes will try to acquire the iptables lock every 100 msec during 5 seconds instead of every second. This is specially useful for environments using kube-proxy in iptables mode with a high churn rate of services. ([#85771](#), [@aojea](#))
- Fixed a panic in the kubelet cleaning up pod volumes ([#86277](#), [@tedyu](#))
- azure cloud provider cache TTL is configurable, list of the azure cloud provider is as following: ([#86266](#), [@zqingqing1](#))
 - ■ "availabilitySetNodesCacheTTLInSeconds"
 - ■ "vmssCacheTTLInSeconds"
 - ■ "vmssVirtualMachinesCacheTTLInSeconds"
 - ■ "vmCacheTTLInSeconds"
 - ■ "loadBalancerCacheTTLInSeconds"
 - ■ "nsgCacheTTLInSeconds"
 - ■ "routeTableCacheTTLInSeconds"
- Fixes kube-proxy when EndpointSlice feature gate is enabled on Windows. ([#86016](#), [@roboscott](#))
- Fixes wrong validation result of NetworkPolicy PolicyTypes ([#85747](#), [@tnqn](#))
- Fixes an issue with kubelet-reported pod status on deleted/recreated pods. ([#86320](#), [@liggitt](#))
- kube-apiserver no longer serves the following deprecated APIs: ([#85903](#), [@liggitt](#))
 - * All resources under apps/v1beta1 and apps/v1beta2
 - use apps/v1 instead * daemonsets, deployments, replicaset resources under extensions/v1beta1
 - use apps/v1 instead * networkpolicies

- resources under extensions/v1beta1 - use networking.k8s.io/v1 instead * podsecuritypolicies resources under extensions/v1beta1 - use policy/v1beta1 instead
- kubeadm: fix potential panic when executing "kubeadm reset" with a corrupted kubelet.conf file ([#86216](#), [@neolit123](#))
 - Fix a bug in port-forward: named port not working with service ([#85511](#), [@oke-py](#))
 - kube-proxy no longer modifies shared EndpointSlices. ([#86092](#), [@roboscott](#))
 - allow for configuration of CoreDNS replica count ([#85837](#), [@pickledrick](#))
 - Fixed a regression where the kubelet would fail to update the ready status of pods. ([#84951](#), [@tedyu](#))
 - Resolves performance regression in client-go discovery clients constructed using NewDiscoveryClientForConfig or NewDiscoveryClientForConfigOrDie. ([#86168](#), [@liggitt](#))
 - Make error message and service event message more clear ([#86078](#), [@feiskyer](#))
 - e2e-test-framework: add e2e test namespace dump if all tests succeed but the cleanup fails. ([#85542](#), [@schrodit](#))
 - SafeSysctlWhitelist: add net.ipv4.ping_group_range ([#85463](#), [@AkihiroSuda](#))
 - kubelet: the metric process_start_time_seconds be marked as with the ALPHA stability level. ([#85446](#), [@RainbowMango](#))
 - API request throttling (due to a high rate of requests) is now reported in the kubelet (and other component) logs by default. The messages are of the form ([#80649](#), [@RobertKrawitz](#))
 - Throttling request took 1.50705208s, request: GET:
 - The presence of large numbers of these messages, particularly with long delay times, may indicate to the administrator the need to tune the cluster accordingly.
 - Fix API Server potential memory leak issue in processing watch request. ([#85410](#), [@answer1991](#))
 - Verify kubelet & kube-proxy can recover after being killed on Windows nodes ([#84886](#), [@YangLu1031](#))
 - Fixed an issue that the scheduler only returns the first failure reason. ([#86022](#), [@Huang-Wei](#))
 - kubectl/drain: add skip-wait-for-delete-timeout option. ([#85577](#), [@michaelgugino](#))
 - If pod DeletionTimestamp older than N seconds, skip waiting for the pod. Seconds must be greater than 0 to skip.
 - Following metrics have been turned off: ([#83841](#), [@RainbowMango](#))
 - ■ kubelet_pod_worker_latency_microseconds
 - ■ kubelet_pod_start_latency_microseconds
 - ■ kubelet_cgroup_manager_latency_microseconds
 - ■ kubelet_pod_worker_start_latency_microseconds
 - ■ kubelet_pleg_relist_latency_microseconds
 - ■ kubelet_pleg_relist_interval_microseconds
 - ■ kubelet_eviction_stats_age_microseconds
 - ■ kubelet_runtime_operations
 - ■ kubelet_runtime_operations_latency_microseconds

- ■ kubelet_runtime_operations_errors
- ■ kubelet_device_plugin_registration_count
- ■ kubelet_device_plugin_alloc_latency_microseconds
- ■ kubelet_docker_operations
- ■ kubelet_docker_operations_latency_microseconds
- ■ kubelet_docker_operations_errors
- ■ kubelet_docker_operations_timeout
- ■ network_plugin_operations_latency_microseconds
- ◦ Renamed Kubelet metric
certificate_manager_server_expiration_seconds to
certificate_manager_server_ttl_seconds and changed to report the
second until expiration at read time rather than absolute time of
expiry. ([#85874](#), [@sambdavidson](#))
 - ■ Improved accuracy of Kubelet metric
rest_client_exec_plugin_ttl_seconds.
- Bind metadata-agent containers to linux nodes to avoid Windows
scheduling on kubernetes cluster includes linux nodes and windows
nodes ([#83363](#), [@wawa0210](#))
- Bind metrics-server containers to linux nodes to avoid Windows
scheduling on kubernetes cluster includes linux nodes and windows
nodes ([#83362](#), [@wawa0210](#))
- During initialization phase (preflight), kubeadm now verifies the
presence of the conntack executable ([#85857](#), [@hnanni](#))
- VMSS cache is added so that less chances of VMSS GET throttling
([#85885](#), [@nilo19](#))
- Update go-winio module version from 0.4.11 to 0.4.14 ([#85739](#),
[@wawa0210](#))
- Fix LoadBalancer rule checking so that no unexpected LoadBalancer
updates are made ([#85990](#), [@feiskyer](#))
- kubectl drain node --dry-run will list pods that would be evicted or
deleted ([#82660](#), [@sallyom](#))
- Windows nodes on GCE can use TPM-based authentication to the
master. ([#85466](#), [@pjh](#))
- kubectl/drain: add disable-eviction option. ([#85571](#), [@michaelgugino](#))
 - Force drain to use delete, even if eviction is supported. This will
bypass checking PodDisruptionBudgets, and should be used with
caution.
- kubeadm now errors out whenever a not supported component config
version is supplied for the kubelet and kube-proxy ([#85639](#), [@roster](#))
- Fixed issue with addon-resizer using deprecated extensions APIs
([#85793](#), [@bskiba](#))
- Includes FSType when describing CSI persistent volumes. ([#85293](#),
[@huffmanca](#))
- kubelet now exports a "server_expiration_renew_failure" and
"client_expiration_renew_failure" metric counter if the certificate
rotations cannot be performed. ([#84614](#), [@rphillips](#))
- kubeadm: don't write the kubelet environment file on "upgrade apply"
([#85412](#), [@boluisa](#))
- fix azure file AuthorizationFailure ([#85475](#), [@andyzhangx](#))
- Resolved regression in admission, authentication, and authorization
webhook performance in v1.17.0-rc.1 ([#85810](#), [@liggitt](#))

- kubeadm: uses the apiserver AdvertiseAddress IP family to choose the etcd endpoint IP family for non external etcd clusters ([#85745](#), [@aojea](#))
- kubeadm: Forward cluster name to the controller-manager arguments ([#85817](#), [@ereslibre](#))
- Fixed "requested device X but found Y" attach error on AWS. ([#85675](#), [@jsafrane](#))
- addons: elasticsearch discovery supports IPv6 ([#85543](#), [@SataQiu](#))
- kubeadm: retry kubeadm-config ConfigMap creation or mutation if the apiserver is not responding. This will improve resiliency when joining new control plane nodes. ([#85763](#), [@ereslibre](#))
- Update Cluster Autoscaler to 1.17.0; changelog: <https://github.com/kubernetes/autoscaler/releases/tag/cluster-autoscaler-1.17.0> ([#85610](#), [@losipiuk](#))
- Filter published OpenAPI schema by making nullable, required fields non-required in order to avoid kubectl to wrongly reject null values. ([#85722](#), [@sttts](#))
- kubectl set resources will no longer return an error if passed an empty change for a resource. ([#85490](#), [@sallyom](#))
 - kubectl set subject will no longer return an error if passed an empty change for a resource.
- kube-apiserver: fixed a conflict error encountered attempting to delete a pod with gracePeriodSeconds=0 and a resourceVersion precondition ([#85516](#), [@michaelgugino](#))
- kubeadm: add a upgrade health check that deploys a Job ([#81319](#), [@neolit123](#))
- kubeadm: make sure images are pre-pulled even if a tag did not change but their contents changed ([#85603](#), [@bart0sh](#))
- kube-apiserver: Fixes a bug that hidden metrics can not be enabled by the command-line option --show-hidden-metrics-for-version. ([#85444](#), [@RainbowMango](#))
- kubeadm now supports automatic calculations of dual-stack node cidr masks to kube-controller-manager. ([#85609](#), [@Arvinderpal](#))
- Fix bug where EndpointSlice controller would attempt to modify shared objects. ([#85368](#), [@roboscott](#))
- Use context to check client closed instead of http.CloseNotifier in processing watch request which will reduce 1 goroutine for each request if proto is HTTP/2.x . ([#85408](#), [@answer1991](#))
- kubeadm: reset raises warnings if it cannot delete folders ([#85265](#), [@SataQiu](#))
- Wait for kubelet & kube-proxy to be ready on Windows node within 10s ([#85228](#), [@YangLu1031](#))

反馈

此页是否对您有帮助？

是 否

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 [Stack Overflow](#)。在 GitHub 仓库上登记新的问题 [报告问题](#) 或者 [提出改进建议](#)。

Kubernetes 版本及版本偏差支持策略

本文描述 Kubernetes 各组件之间版本偏差支持策略。特定的集群部署工具可能会有额外的限制。

版本支持策略

Kubernetes 版本号格式为 **x.y.z**，其中 **x** 为大版本号，**y** 为小版本号，**z** 为补丁版本号。版本号格式遵循 [Semantic Versioning](#) 规则。更多信息，请参阅 [Kubernetes 发布版本](#)。

Kubernetes 项目会维护最近三个小版本分支（1.20, 1.19, 1.18）。Kubernetes 1.19 及更高的版本将获得大约1年的补丁支持。Kubernetes 1.18 及更早的版本获得大约9个月的补丁支持。

一些 bug 修复，包括安全修复，取决于其严重性和可行性，有可能会反向合并到这三个发布分支。补丁版本会[定期](#)或根据需要从这些分支中发布。最终是否发布是由[发布管理者](#)来决定的。如需了解更多信息，请查看 Kubernetes [补丁发布](#)。

版本偏差策略

kube-apiserver

在 [高可用（HA）集群](#) 中，多个 kube-apiserver 实例小版本号最多差1。

例如：

- 最新的 kube-apiserver 版本号如果是 **1.20**
- 则受支持的 kube-apiserver 版本号包括 **1.20** 和 **1.19**

kubelet

kubelet 版本号不能高于 kube-apiserver，最多可以比 kube-apiserver 低两个小版本。

例如：

- kube-apiserver 版本号如果是 **1.20**
- 受支持的 kubelet 版本将包括 **1.20**、**1.19** 和 **1.18**

说明：如果 HA 集群中多个 kube-apiserver 实例版本号不一致，相应的 kubelet 版本号可选范围也要减小。

例如：

- 如果 kube-apiserver 实例同时存在 **1.20** 和 **1.19**

- kubelet 的受支持版本将是 **1.19** 和 **1.18** (**1.20** 不再支持, 因为它比 **1.19** 版本的 kube-apiserver 更新)

kube-controller-manager、 kube-scheduler 和 cloud-controller-manager

kube-controller-manager、 kube-scheduler 和 cloud-controller-manager 版本不能高于 kube-apiserver 版本号。 最好它们的版本号与 kube-apiserver 保持一致, 但允许比 kube-apiserver 低一个小版本 (为了支持在线升级)。

例如 :

- 如果 kube-apiserver 版本号为 **1.20**
- kube-controller-manager、 kube-scheduler 和 cloud-controller-manager 版本支持 **1.20** 和 **1.19**

说明 : 如果在 HA 集群中, 多个 kube-apiserver 实例版本号不一致, 他们也可以跟任意一个 kube-apiserver 实例通信 (例如, 通过 load balancer), 但 kube-controller-manager、 kube-scheduler 和 cloud-controller-manager 版本可用范围会相应的减小。

例如 :

- kube-apiserver 实例同时存在 **1.20** 和 **1.19** 版本
- kube-controller-manager、 kube-scheduler 和 cloud-controller-manager 可以通过 load balancer 与所有的 kube-apiserver 通信
- kube-controller-manager、 kube-scheduler 和 cloud-controller-manager 可选版本为 **1.19** (**1.20** 不再支持, 因为它比 **1.19** 版本的 kube-apiserver 更新)

kubectrl

kubectrl 可以比 kube-apiserver 高一个小版本, 也可以低一个小版本。

例如 :

- 如果 kube-apiserver 当前是 **1.20** 版本
- kubectrl 则支持 **1.21**、 **1.20** 和 **1.19**

说明 : 如果 HA 集群中的多个 kube-apiserver 实例版本号不一致, 相应的 kubectrl 可用版本范围也会减小。

例如 :

- kube-apiserver 多个实例同时存在 **1.20** 和 **1.19**
- kubectrl 可选的版本为 **1.20** 和 **1.19** (其他版本不再支持, 因为它会比其中某个 kube-apiserver 实例高或低一个小版本)

支持的组件升级次序

组件之间支持的版本偏差会影响组件升级的顺序。 本节描述组件从版本 **1.19** 到 **1.20** 的升级次序。

kube-apiserver

前提条件：

- 单实例集群中，kube-apiserver 实例版本号须是 **1.19**
- 高可用（HA）集群中，所有的 kube-apiserver 实例版本号必须是 **1.19** 或 **1.20**（确保满足最新和最旧的实例小版本号相差不大于1）
- kube-controller-manager、kube-scheduler 和 cloud-controller-manager 版本号必须为 **1.19**（确保不高于 API server 的版本，且版本号相差不大于1）
- kubelet 实例版本号必须是 **1.19** 或 **1.18**（确保版本号不高于 API server，且版本号相差不大于2）
- 注册的 admission 插件必须能够处理新的 kube-apiserver 实例发送过来的数据：
 - ValidatingWebhookConfiguration 和 MutatingWebhookConfiguration 对象必须升级到可以处理 **1.20** 版本新加的 REST 资源（或使用 1.15 版本提供的 [matchPolicy: Equivalent 选项](#)）
 - 插件可以处理任何 **1.20** 版本新的 REST 资源数据和新加的字段

升级 kube-apiserver 到 **1.20**

说明：

根据 [API 弃用策略](#) 和 [API 变更指南](#)， kube-apiserver 不能跨小版本号升级，即使是单实例集群也不可以。

kube-controller-manager、kube-scheduler 和 cloud-controller-manager

前提条件：

- kube-apiserver 实例必须为 **1.20**（HA 集群中，所有的 kube-apiserver 实例必须在组件升级前完成升级）

升级 kube-controller-manager、kube-scheduler 和 cloud-controller-manager 到 **1.20**

kubelet

前提条件：

- kube-apiserver 实例必须为 **1.20** 版本

kubelet 可以升级到 **1.20**（或者停留在 **1.19** 或 **1.18**）

警告：

集群中 kubelet 版本号不建议比 kube-apiserver 低两个版本号：

- 它们必须升级到与 kube-apiserver 相差不超过 1 个小版本，才可以升级其他控制面组件
- 有可能使用低于 3 个在维护的小版本

kube-proxy

- kube-proxy 必须与节点上的 kubelet 的小版本相同
- kube-proxy 一定不能比 kube-apiserver 小版本更新
- kube-proxy 最多只能比 kube-apiserver 早两个小版本

例如：

如果 kube-proxy 的版本是 **1.18**：

- kubelet 版本必须相同，也是 **1.18**
- kube-apiserver 版本必须在 **1.18** 到 **1.20** 之间（闭区间）

反馈

此页是否对您有帮助？

是 否

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 [Stack Overflow](#). 在 GitHub 仓库上登记新的问题 [报告问题](#) 或者 [提出改进建议](#).

最后修改 October 10, 2020 at 11:06 AM PST: [fix outdated content \(77071553d\)](#)

安装工具

在你的计算机上设置 Kubernetes 工具。

kubectl

Kubernetes 命令行工具，kubectl，使得你可以对 Kubernetes 集群运行命令。你可以使用 kubectl 来部署应用、监测和管理集群资源以及查看日志。

关于如何下载和安装 kubectl 并配置其访问你的集群，可参阅 [安装和配置 kubectl](#)。

[查看 kubectl 安装和配置指南](#)

你也可以阅读 [kubectl 参考文档](#)。

kind

[kind](#) 让你能够在本地计算机上运行 Kubernetes。kind 要求你安装并配置好 [Docker](#)。

kind [快速入门](#) 页面展示了 开始使用 kind 所需要完成的操作。

[查看 kind 的快速入门指南](#)

minikube

与 kind 类似，[minikube](#) 是一个工具，能让你在本地运行 Kubernetes。minikube 在你本地的个人计算机（包括 Windows、macOS 和 Linux PC）运行一个单节点的 Kubernetes 集群，以便你来尝试 Kubernetes 或者开展每天的开发工作。

如果你关注如何安装此工具，可以按官方的 [Get Started!](#) 指南操作。

[查看 minikube 快速入门指南](#)

当你拥有了可工作的 minikube 时，就可以用它来 [运行示例应用](#)了。

kubeadm

你可以使用 [kubeadm](#) 工具来 创建和管理 Kubernetes 集群。该工具能够执行必要的动作并用一种用户友好的方式启动一个可用的、安全的集群。

[安装 kubeadm](#) 展示了如何安装 kubeadm 的过程。一旦安装了 kubeadm，你就可以使用它来 [创建一个集群](#)。

[查看 kubeadm 安装指南](#)

生产环境

[容器运行时](#)

[Turnkey 云解决方案](#)

[使用部署工具安装 Kubernetes](#)

[Windows Kubernetes](#)

容器运行时

你需要在集群内每个节点上安装一个[容器运行时](#) 以使 Pod 可以运行在上面。本文概述了所涉及的内容并描述了与节点设置相关的任务。

本文列出了在 Linux 上结合 Kubernetes 使用的几种通用容器运行时的详细信息：

- [containerd](#)
- [CRI-O](#)
- [Docker](#)

提示：对于其他操作系统，请查阅特定于你所使用平台的相关文档。

Cgroup 驱动程序

控制组用来约束分配给进程的资源。

当某个 Linux 系统发行版使用 [systemd](#) 作为其初始化系统时，初始化进程会生成并使用一个 root 控制组 (cgroup)，并充当 cgroup 管理器。Systemd 与 cgroup 集成紧密，并将为每个 systemd 单元分配一个 cgroup。你也可以配置容器运行时和 kubelet 使用 cgroupfs。连同 systemd 一起使用 cgroupfs 意味着将有两个不同的 cgroup 管理器。

单个 cgroup 管理器将简化分配资源的视图，并且默认情况下将对可用资源和使用中的资源具有更一致的视图。当有两个管理器共存于一个系统中时，最终将对这些资源产生两种视图。在此领域人们已经报告过一些案例，某些节点配置让 kubelet 和 docker 使用 cgroupfs，而节点上运行的其余进程则使用 systemd；这类节点在资源压力下会变得不稳定。

更改设置，令容器运行时和 kubelet 使用 systemd 作为 cgroup 驱动，以此使系统更为稳定。对于 Docker，设置 `native.cgroupdriver=systemd` 选项。

注意：非常不建议更改已加入集群的节点的 cgroup 驱动。如果 kubelet 已经使用某 cgroup 驱动的语义创建了 pod，更改运行时以使用别的 cgroup 驱动，当为现有 Pods 重新创建 PodSandbox 时会产生错误。重启 kubelet 也可能无法解决此类问题。如果你有切实可行的自动化方案，使用其他已更新配置的节点来替换该节点，或者使用自动化方案来重新安装。

容器运行时

注意：本部分链接到提供 Kubernetes 所需功能的第三方项目。Kubernetes 项目作者不负责这些项目。此页面遵循[CNCF 网站指南](#)，按字母顺序列出项目。要将项目添加到此列表中，请在提交更改之前阅读[内容指南](#)。

containerd

本节包含使用 containerd 作为 CRI 运行时的必要步骤。

使用以下命令在系统上安装容器：

安装和配置的先决条件：

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

```
sudo modprobe overlay
sudo modprobe br_netfilter
```

设置必需的 sysctl 参数，这些参数在重新启动后仍然存在。

```
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
```

安装 containerd:

- [Ubuntu 16.04](#)
- [Ubuntu 18.04/20.04](#)
- [Debian 9+](#)
- [CentOS/RHEL 7.4+](#)
- [Windows \(PowerShell\)](#)

(安装 containerd)

(设置仓库)

(安装软件包以允许 apt 通过 HTTPS 使用存储库)

```
sudo apt-get update && sudo apt-get install -y apt-transport-https ca-
certificates curl software-properties-common
```

安装 Docker 的官方 GPG 密钥

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key --
keyring /etc/apt/trusted.gpg.d/docker.gpg add -
```

新增 Docker apt 仓库。

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

安装 containerd

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

配置 containerd

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


重启 containerd

```
sudo systemctl restart containerd
```

安装 containerd

```
sudo apt-get update && sudo apt-get install -y containerd
```

配置 containerd

```
sudo mkdir -p /etc/containerd
```

```
sudo containerd config default | sudo tee /etc/containerd/config.toml
```

重启 containerd

```
sudo systemctl restart containerd
```

安装 containerd

配置仓库

安装软件包以使 apt 能够使用 HTTPS 访问仓库

```
sudo apt-get update && sudo apt-get install -y apt-transport-https ca-  
certificates curl software-properties-common
```

添加 Docker 的官方 GPG 密钥

```
curl -fsSL https://download.docker.com/linux/debian/gpg | sudo apt-key --  
keyring /etc/apt/trusted.gpg.d/docker.gpg add -
```

添加 Docker apt 仓库

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

安装 containerd

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

设置 containerd 的默认配置

```
sudo mkdir -p /etc/containerd
```

```
containerd config default | sudo tee /etc/containerd/config.toml
```

重启 containerd

```
sudo systemctl restart containerd
```

安装 containerd

设置仓库

安装所需包

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

添加 Docker 仓库

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

```
## 安装 containerd
```

```
sudo yum update -y && sudo yum install -y containerd.io
```

```
# 配置 containerd
```

```
sudo mkdir -p /etc/containerd
```

```
containerd config default | sudo tee /etc/containerd/config.toml
```

```
# 重启 containerd
```

```
sudo systemctl restart containerd
```

```
# 安装 containerd
```

```
# 下载 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
```

```
# 解压并配置
```

```
Copy-Item -Path ".\bin\" -Destination "$Env:ProgramFiles\containerd" -Recurse -Force
```

```
cd $Env:ProgramFiles\containerd\
```

```
.\containerd.exe config default | Out-File config.toml -Encoding ascii
```

```
# 检查配置文件，基于你可能想要调整的设置：
```

```
# - sandbox_image (kubernetes pause 镜像)
```

```
# - CNI 的 bin_dir 和 conf_dir 路径
```

```
Get-Content config.toml
```

```
# 启动 containerd
```

```
.\containerd.exe --register-service
```

```
Start-Service containerd
```

systemd

结合 runc 使用 systemd cgroup 驱动，在 /etc/containerd/config.toml 中设置

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

```
...
```

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

```
SystemdCgroup = true
```

当使用 kubeadm 时，请手动配置 [kubelet 的 cgroup 驱动](#)。

CRI-O

本节包含安装 CRI-O 作为容器运行时的必要步骤。

使用以下命令在系统中安装 CRI-O：

提示：CRI-O 的主要以及次要版本必须与 Kubernetes 的主要和次要版本相匹配。更多信息请查阅 [CRI-O 兼容性列表](#)。

安装以及配置的先决条件：

```
# 创建 .conf 文件，以便在系统启动时加载内核模块
cat <<EOF | sudo tee /etc/modules-load.d/crio.conf
overlay
br_netfilter
EOF

sudo modprobe overlay
sudo modprobe br_netfilter

# 设置必需的 sysctl 参数，这些参数在重新启动后仍然存在。
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](#)

在下列操作系统上安装 CRI-O, 使用下表中合适的值设置环境变量 OS:

操作系统	\$OS
Debian Unstable	Debian_Unstable
Debian Testing	Debian_Testing

然后，将 \$VERSION 设置为与你的 Kubernetes 相匹配的 CRI-O 版本。例如，如果你要安装 CRI-O 1.18, 请设置 VERSION=1.18. 你也可以安装一个特定的发行版本。例如要安装 1.18.3 版本，设置 VERSION=1.18:1.18.3.

然后执行

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

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

在下列操作系统上安装 CRI-O, 使用下表中合适的值设置环境变量 OS:

操作系统	\$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

然后, 将 \$VERSION 设置为与你的 Kubernetes 相匹配的 CRI-O 版本。例如, 如果你要安装 CRI-O 1.18, 请设置 VERSION=1.18. 你也可以安装一个特定的发行版本。例如要安装 1.18.3 版本, 设置 VERSION=1.18:1.18.3.

然后执行

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

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

在下列操作系统上安装 CRI-O, 使用下表中合适的值设置环境变量 OS:

操作系统	\$OS
Centos 8	CentOS_8
Centos 8 Stream	CentOS_8_Stream
Centos 7	CentOS_7

然后, 将 \$VERSION 设置为与你的 Kubernetes 相匹配的 CRI-O 版本。例如, 如果你要安装 CRI-O 1.18, 请设置 VERSION=1.18. 你也可以安装一个特定的发行版本。例如要安装 1.18.3 版本, 设置 VERSION=1.18:1.18.3.

然后执行

```
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.r
epo 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
```

将 \$VERSION 设置为与你的 Kubernetes 相匹配的 CRI-O 版本。例如, 如果要安装 CRI-O 1.18, 请设置 VERSION=1.18。你可以用下列命令查找可用的版本:

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

CRI-O 不支持在 Fedora 上固定到特定的版本。

然后执行

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

启动 CRI-O:

```
sudo systemctl daemon-reload
sudo systemctl start cri-o
```

更多信息请参阅 [CRI-O 安装指南](#)。

Docker

在你的所有节点上安装 Docker CE.

Kubernetes 发布说明中列出了 Docker 的哪些版本与该版本的 Kubernetes 相兼容。

在你的操作系统上使用如下命令安装 Docker:

- [Ubuntu 16.04+](#)
- [CentOS/RHEL 7.4+](#)

```
# (安装 Docker CE)
```

```
## 设置仓库:
```

```
### 安装软件包以允许 apt 通过 HTTPS 使用存储库
```

```
sudo apt-get update && sudo apt-get install -y \
  apt-transport-https ca-certificates curl software-properties-common gnupg2
```

```
### 新增 Docker 的官方 GPG 秘钥:
```

```
curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add --
keyring /etc/apt/trusted.gpg.d/docker.gpg -
```

```
### 添加 Docker apt 仓库:
```

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

```
## 安装 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)
```

```
# 设置 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
```



```
# 重启 docker.
```

```
sudo systemctl daemon-reload  
sudo systemctl restart docker
```

```
# (安装 Docker CE)
```

```
## 设置仓库
```

```
### 安装所需包
```

```
sudo yum install -y yum-utils device-mapper-persistent-data lvm2
```

```
### 新增 Docker 仓库
```

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

```
## 安装 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
```

```
## 创建 /etc/docker 目录
```

```
sudo mkdir /etc/docker
```

```
# 设置 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
```

```
# 重启 Docker
```

```
sudo systemctl daemon-reload  
sudo systemctl restart docker
```

如果你想开机即启动 docker 服务，执行以下命令：

```
sudo systemctl enable docker
```

请参阅[官方 Docker 安装指南](#) 获取更多的信息。

反馈

此页是否对您有帮助？

是 否

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 [Stack Overflow](#). 在 GitHub 仓库上登记新的问题 [报告问题](#) 或者 [提出改进建议](#).

最后修改 January 11, 2021 at 9:55 AM PST: [\[zh\] Resync setup/production-environment/container-runtimes.md \(fd12858bd\)](#)

Turnkey 云解决方案

本页列示 Kubernetes 认证解决方案供应商。在每一个供应商分页，你可以学习如何安装和设置生产就绪的集群。

```
<script src="https://landscape.cncf.io/iframeResizer.js"/></div><div
id="pre-footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您
有帮助？</p><button class="btn btn-primary mb-4 feedback-yes">是</
button> <button class="btn btn-primary mb-4 feedback-no">否</
button><p class="feedback-response feedback--response hidden">感谢反
馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 <a
target=" blank" rel="noopener" href="https://stackoverflow.com/questions/
tagged/kubernetes">Stack Overflow</a>. 在 GitHub 仓库上登记新的问题 <a
class="feedback-link" target=" blank" rel="noopener" href="https://
github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io">
报告问题</a> 或者 <a class="feedback-link" target=" blank" rel="noopener"
href="https://github.com/kubernetes/website/issues/new?
title=Improvement%20for%20k8s.io">提出改进建议</a>.</p></
div><script>const yes=document.querySelector('.feedback-yes');const
no=document.querySelector('.feedback-
no');document.querySelectorAll('.feedback-link').forEach(link=&gt;
{link.href=link.href+window.location.pathname;});const
sendFeedback=(value)=&gt;{if(!gtag){console.log('!gtag');}
gtag('event','click',
{'event category':'Helpful','event label':window.location.pathname,value});};const
disableButtons=()=&gt;{yes.disabled=true;yes.classList.add('feedback--
button disabled');no.disabled=true;no.classList.add('feedback--
button disabled');};yes.addEventListener('click',()=&gt;
{sendFeedback(1);disableButtons();document.querySelector('.feedback--
response').classList.remove('feedback--
response hidden');});no.addEventListener('click',()=&gt;
{sendFeedback(0);disableButtons();document.querySelector('.feedback--
response').classList.remove('feedback-response hidden');});</script><div
class="text-muted mt-5 pt-3 border-top">最后修改 November 14, 2020 at
10:39 AM PST: <a href="https://github.com/kubernetes/website/commit/
```

54d5270823ad391a8ef3b5bd0f9977789510063a">[zh] translate setup's Turnkey Cloud Solutions (54d527082)</div></div><div class="td-

content"><h1>使用部署工具安装 Kubernetes</h1><div class="section-index"><hr class="panel-line"/><div class="entry"><h5>使用 kubeadm 引导集群</h5><p/></div><div class="entry"><h5>使用 Kops 安装 Kubernetes</h5><p/></div><div class="entry"><h5>使用 Kubespray 安装 Kubernetes</h5><p/></div></div></div><div class="td-

content"><h1>使用 kubeadm 引导集群</h1><div class="section-index"><hr class="panel-line"/><div class="entry"><h5>安装 kubeadm</h5><p/></div><div class="entry"><h5>对 kubeadm 进行故障排查</h5><p/></div><div class="entry"><h5>使用 kubeadm 创建集群</h5><p/></div><div class="entry"><h5>使用 kubeadm 定制控制平面配置</h5><p/></div><div class="entry"><h5>高可用拓扑选项</h5><p/></div><div class="entry"><h5>利用 kubeadm 创建高可用集群</h5><p/></div><div class="entry"><h5>使用 kubeadm 创建一个高可用 etcd 集群</h5><p/></div><div class="entry"><h5>使用 kubeadm 配置集群中的每个 kubelet</h5><p/></div><div class="entry"><h5>配置您的 kubernetes 集群以自托管控制平台</h5><p/></div></div></div><div class="td-content"><h1>安装 kubeadm</h1><p>本页面显示如何安装 `kubeadm` 工具箱。有关在执行此安装过程后如何使用 kubeadm 创建集群的信息，请参见 使用 kubeadm 创建集群 页面。</p><h2 id="#准备开始">准备开始</h2>一台或多台运行着下列系统的机器：Ubuntu 16.04+Debian 9+CentOS 7Red Hat Enterprise Linux (RHEL) 7Fedora 25+HypriotOS v1.0.1+Flatcar Container Linux（使用 2512.3.0 版本测试通过）每台机器 2 GB 或更多的 RAM（如果少于这个数字将会影响你应用的运行内存）2 CPU 核或更多集群中的所有机器的网络彼此均能相互连接（公网和内网都可以）节点之中不可以有重复的主机名、MAC 地址或 product uuid。请参见这里了解更多详细信息。开启机器上的某些端口。请参见这里了解更多详细信息。禁用交换分区。为了保证 kubelet 正常工作，你 必须 禁用交换分区。<h2

产品

id="verify-mac-address">确保每个节点上 MAC 地址和 product uuid 的唯一性</h2>你可以使用命令 `ip link` 或 `ifconfig -a` 来获取网络接口的 MAC 地址可以使用 `sudo cat /sys/class/dmi/id/product uuid` 命令对 product uuid 校验<p>一般来讲，硬件设备会拥有唯一的地址，但是有些虚拟机的地址可能会重复。Kubernetes 使用这些值来唯一确定集群中的节点。如果这些值在每个节点上不唯一，可能会导致安装 [失败](https://github.com/kubernetes/kubeadm/issues/31)。</p><h2>

id="检查网络适配器">

检查网络适配器</h2><p>如果你有一个以上的网络适配器，同时你的 Kubernetes 组件通过默认路由不可达，我们建议你预先添加 IP 路由规则，这样 Kubernetes 集群就可以通过对应的适配器完成连接。</p><h2 id="允许iptables-检查桥接流量">允许

iptables 检查桥接流量</h2><p>确保 `br netfilter` 模块被加载。这一操作可以通过运行 `lsmod | grep br netfilter` 来完成。若要显式加载该模块，可执行 `sudo modprobe br netfilter`。</p><p>为了让你的 Linux 节点上的 iptables 能够正确地查看桥接流量，你需要确保在你的 `sysctl` 配置中将 `net.bridge.bridge-nf-call-iptables` 设置为 1。例如：</p><div class="highlight"><pre>

网络插件需求页面。</p><h2 id="check-required-ports">检查所需端口</h2><h3>

id="控制平面节点">控制平面节点</h3><table><thead><tr><th>协议</th><th>方向</th><th>端口范围</th><th>作用</th><th>使用者</th></tr></thead><tbody><tr><td>TCP</td><td>入站</td><td>6443</td><td>Kubernetes API 服务器</td><td>所有组件</td></tr><tr><td>TCP</td><td>入站</td><td>2379-2380</td><td>etcd 服务器客户端 API</td><td>kube-apiserver, etcd</td></tr><tr><td>TCP</td><td>入站</td><td>10250</td><td>Kubelet API</td><td>kubelet 自身、控制平面组件</td></tr><tr><td>TCP</td><td>入站</td><td>10251</td><td>kube-scheduler</td><td>kube-scheduler 自身</td></tr><tr><td>TCP</td><td>入站</td><td>10252</td><td>kube-controller-manager</td><td>kube-controller-manager 自身</td></tr></tbody></table><h3 id="工作节点">工作节点</h3><table><thead><tr><th>协议</th><th>方向</th><th>端口范围</th><th>作用</th><th>使用者</th></tr></thead><tbody><tr><td>TCP</td><td>入站</td><td>10250</td><td>Kubelet API</td><td>kubelet 自身、控制平面组件</td></tr><tr><td>TCP</td><td>入站</td><td>30000-32767</td><td></td><td></td></tr></tbody></table>

NodePort 服务†

所有组件

† [NodePort 服务](/zh/docs/concepts/services-networking/service/) 的默认端口范围。

使用 * 标记的任意端口号都可以被覆盖，所以你需要保证所定制的端口是开放的。

虽然控制平面节点已经包含了 etcd 的端口，你也可以使用自定义的外部 etcd 集群，或是指定自定义端口。

你使用的 Pod 网络插件 (见下) 也可能需要某些特定端口开启。由于各个 Pod 网络插件都有所不同，请参阅他们各自文档中对端口的要求。

安装 runtime

为了在 Pod 中运行容器，Kubernetes 使用 Container Runtime Interface (CRI)。

- Linux 节点
- 其它操作系统

容器运行时接口 (Container Runtime Interface, CRI)

来与你所选择的容器运行时交互。

如果你不指定运行时，则 kubeadm 会自动尝试检测到系统上已经安装的运行时，方法是扫描一组众所周知的 Unix 域套接字。下面的表格列举了一些容器运行时及其对应的套接字路径：

运行时	域套接字
Docker	/var/run/docker.sock
containerd	/run/containerd/containerd.sock
CRI-O	/var/run/crio/crio.sock

如果同时检测到 Docker 和 containerd，则优先选择 Docker。这是必然的，因为 Docker 18.09 附带了 containerd 并且两者都是可以检测到的，即使你仅安装了 Docker。如果检测到其他两个或多个运行时，kubeadm 输出错误信息并退出。

kubelet 通过内置的 `dockershim` CRI 实现与 Docker 集成。

参阅 [容器运行时](/zh/docs/setup/production-environment/container-runtimes/) 以了解更多信息。

Docker

作为容器运行时。kubelet 通过内置的 `dockershim` CRI 实现与 Docker 集成。参阅 [容器运行时](/zh/docs/setup/production-environment/container-runtimes/) 以了解更

多信息。

安装 kubeadm、kubenet 和 kubectl

你需要在每台机器上安装以下的软件包：

- `kubeadm`：用来初始化集群的指令。
- `kubelet`：在集群中的每个节点上用来启动 Pod 和容器等。
- `kubectl`：用来与集群通信的命令行工具。

`kubeadm` 不能 帮你安装或者管理 `kubelet` 或 `kubectl`，所以你需要 确保它们与通过 `kubeadm` 安装的控制平面的版本相匹配。如果不这样做，则存在发生版本偏差的风险，可能会导致一些预料之外的错误和问题。然而，控制平面与 `kubelet` 间的相差一个次要版本不一致是支持的，但 `kubelet` 的版本不可以超过 API 服务器的版本。例如，1.7.0 版本的 `kubelet` 可以完全兼容 1.8.0 版本的 API 服务器，反之则不可以。

有关安装 `kubectl` 的信息，请参阅[安装和设置 kubectl](/zh/docs/tasks/tools/install-kubectl/)文档。

警告：这些指南不包括系统升级时使用的所有 Kubernetes 程序包。这是因为 `kubeadm` 和 Kubernetes 有[特殊的升级注意事项](/zh/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/)。

关于版本偏差的更多信息，请参阅以下文档：

- Kubernetes [版本与版本间的偏差策略](/zh/docs/setup/release/version-skew-policy/)
- Kubeadm 特定的[版本偏差策略](/zh/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#version-skew-policy)

[Ubuntu、Debian 或 HypriotOS](#k8s-install-0)

[CentOS、RHEL 或 Fedora](#k8s-install-1)

[Fedora CoreOS 或 Flatcar Container Linux](#k8s-install-2)

```
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
```



```
</span><span style="color:#b44">gpgcheck=1 </span><span style="color:#b44">repo gpgcheck=1 </span><span style="color:#b44">gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.com/yum/doc/rpm-package-key.gpg </span><span style="color:#b44">exclude=kubelet kubeadm kubectl </span><span style="color:#b44">EOF</span> <span style="color:#080;font-style:italic"># 将 SELinux 设置为 permissive 模式 ( 相当于将其禁用 ) </span> setenforce <span style="color:#666">0</span> sed -i <span style="color:#b44">'s/^SELINUX=enforcing$/SELINUX=permissive/'</span> /etc/selinux/config yum install -y kubelet kubeadm kubectl --disableexcludes<span style="color:#666">=</span></span>kubernetes systemctl <span style="color:#a2f">enable</span> --now kubelet </code></pre></div><p><strong>请注意 : </strong></p><ul><li><p>通过运行命令 <code>setenforce 0</code> 和 <code>sed ...</code> 将 SELinux 设置为 permissive 模式 可以有效地将其禁用。这是允许容器访问主机文件系统所必需的，而这些操作时为了例如 Pod 网络工作正常。</p><p>你必须这么做，直到 kubelet 做出对 SELinux 的支持进行升级为止。</p></li><li><p>你可以保持 SELinux 处于弃用状态，前提是你知道如何配置它，不过这也意味着有些 配置是 kubeadm 所不支持的。</p></li></ul></div><div id="k8s-install-2" class="tab-pane" role="tabpanel" aria-labelledby="k8s-install-2"><p><p>安装 CNI 插件 ( 大多数 Pod 网络都需要 ) : </p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash"><span style="color:#b8860b">CNI VERSION</span><span style="color:#666">=</span><span style="color:#b44">"v0.8.2"</span></span><span style="color:#b44">"https://github.com/containernetworking/plugins/releases/download/</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">CNI VERSION</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">/"</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">".tgz"</span> | sudo tar -C /opt/cni/bin -xz </code></pre></div><p>定义要下载命令文件的目录。</p><blockquote class="note callout"><div><strong>说明 : </strong><p>DOWNLOAD DIR 变量必须被设置为一个可写入的目录。如果你在运行 Flatcar Container Linux , 可将 DOWNLOAD DIR 设置为 /opt/bin。</p></div></blockquote><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash"><span style="color:#b8860b">DOWNLOAD DIR</span><span style="color:#666">=</span><span>/usr/local/bin</span> sudo mkdir -p <span style="color:#b8860b">${DOWNLOAD DIR}</span> </code></pre></div><p>安装 crictl ( kubeadm/kubelet 容器运行时接口 ( CRI ) 所需 ) </p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash"><span style="color:#b8860b">CRICTL VERSION</span><span style="color:#666">=</span><span style="color:#b44">"v1.17.0"</span><span> curl -L <span style="color:#b44">"https://github.com/kubernetes-sigs/cri-tools/releases/download/</span><span style="color:#b68;font-weight:700">${</span>
```

```
span><span style="color:#b8860b">CRICTL VERSION</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">/cricctl-</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">CRICTL VERSION</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">-linux-amd64.tar.gz"</span> | sudo tar -C <span style="color:#b8860b">$DOWNLOAD DIR</span> -xz </code></pre></div><p>安装 <code>kubeadm</code>、<code>kubelet</code>、<code>kubectl</code> 并添加 <code>kubelet</code> 系统服务 : </p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash"><span style="color:#b8860b">RELEASE</span><span style="color:#666">=</span><span style="color:#b44">"</span><span style="color:#a2f;font-weight:700">$(</span>curl -sSL https://dl.k8s.io/release/stable.txt<span style="color:#a2f;font-weight:700">)</span><span style="color:#b44">"</span> <span style="color:#a2f">cd</span> <span style="color:#b8860b">$DOWNLOAD DIR</span> sudo curl -L --remote-name-all https://storage.googleapis.com/kubernetes-release/release/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">RELEASE</span><span style="color:#b68;font-weight:700">}</span>/bin/linux/amd64/<span style="color:#666">{</span><span>kubeadm,kubelet,kubectl<span style="color:#666">}</span><span> chmod +x <span style="color:#666">{</span><span>kubeadm,kubelet,kubectl<span style="color:#666">}</span> <span style="color:#b8860b">RELEASE VERSION</span><span style="color:#666">=</span><span style="color:#b44">"v0.4.0"</span><span> curl -sSL <span style="color:#b44">"https://raw.githubusercontent.com/kubernetes/release/</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">RELEASE VERSION</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">/cmd/kubepkg/templates/latest/deb/kubelet/lib/systemd/system/kubelet.service"</span> | sed <span style="color:#b44">"s:/usr/bin:</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">DOWNLOAD DIR</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">:g"</span> | sudo tee /etc/systemd/system/kubelet.service sudo mkdir -p /etc/systemd/system/kubelet.service.d curl -sSL <span style="color:#b44">"https://raw.githubusercontent.com/kubernetes/release/</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">RELEASE VERSION</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">/cmd/kubepkg/templates/latest/deb/kubeadm/10-kubeadm.conf"</span> | sed <span style="color:#b44">"s:/usr/bin:</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">DOWNLOAD DIR</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">:g"</span> | sudo tee /etc/systemd/system/kubelet.service.d/10-kubeadm.conf curl -sSL <span style="color:#b44">"https://raw.githubusercontent.com/kubernetes/kubernetes/</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">RELEASE</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">/</span>
```

```
build/debs/kubelet.service" | sed "s:/usr/bin:/opt/bin:g" & > /etc/systemd/system/kubelet.service mkdir -p /etc/systemd/system/kubelet.service.d curl -sSL "https://raw.githubusercontent.com/kubernetes/kubernetes/RELEASE" } } build/debs/10-kubeadm.conf" | sed "s:/usr/bin:/opt/bin:g" & > /etc/systemd/system/kubelet.service.d/10-kubeadm.conf
```

激活并启动 `kubelet` :

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

说明 :

Flatcar Container Linux 发行版会将 `/usr/` 目录挂载为一个只读文件系统。在启动引导你的集群之前，你需要执行一些额外的操作来配置一个可写入的目录。参见 [kubeadm 故障排查指南](/zh/docs/setup/production-environment/tools/kubeadm/troubleshooting-kubeadm/#usr-mounted-read-only) 以了解如何配置一个可写入的目录。

`kubelet` 现在每隔几秒就会重启，因为它陷入了一个等待 `kubeadm` 指令的死循环。

在控制平面节点上配置 `kubelet` 使用的 `cgroup` 驱动程序

使用 `Docker` 时，`kubeadm` 会自动为其检测 `cgroup` 驱动并在运行时对 `/var/lib/kubelet/kubeadm-flags.env` 文件进行配置。

如果你在使用不同的 CRI，你必须为 `kubeadm init` 传递 `cgroupDriver` 值，像这样：

```
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
cgroupDriver: cgroupfs
```

进一步的相关细节，可参阅 [使用配置文件来执行 `kubeadm init`](/zh/docs/reference/setup-tools/kubeadm/kubeadm-init/#config-file)。

请注意，你只需要在你的 `cgroup` 驱动程序不是 `cgroupfs` 时这么做，因为它已经是 `kubelet` 中的默认值。

说明 :

由于 `kubelet` 已经弃用了 `--cgroup-driver` 标志，如果你在配置文件 `/var/lib/kubelet/kubeadm-flags.env` 或者 `/etc/default/kubelet` (对于 RPM 而言是 `/etc/sysconfig/kubelet`) 包含此设置，请将其删除 并使用 `KubeletConfiguration` 作为替代 (默认存储于 `/var/lib/kubelet/config.yaml` 文件中)。

需要重新启动 `kubelet` :

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

自

动检测其他容器运行时（例如 CRI-O 和 containerd）的 cgroup 驱动的相关工作仍在进行中。

故障排查

如果你在使用 kubeadm 时遇到困难，请参阅我们的 [故障排查文档](/zh/docs/setup/production-environment/tools/kubeadm/troubleshooting-kubeadm/)。

接下来

- [使用 kubeadm 创建集群](/zh/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/)

反馈

此页是否对您有帮助？

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 [Stack Overflow](https://stackoverflow.com/questions/tagged/kubernetes)。在 GitHub 仓库上登记新的问题 [报告问题](https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io) 或者 [提出改进建议](https://github.com/kubernetes/website/issues/new?title=Improvement%20for%20k8s.io)。

id="在安装过程中没񧂞
btables-

或者其他类似的可&#x

在安装过程中没有找到 `etables` 或者其他类似的可执行文件</

h2><p>如果在运行 `kubeadm init` 命令时，遇到以下的警告</

p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh">[preflight] WARNING: etables not found in system path[preflight] WARNING: ethtool not found in system path</code></pre></div><p>那么或许在你的节点上缺失 `etables`、

`ethtool` 或者类似的可执行文件。你可以使用以下命令安装它们：</p>对于 Ubuntu/Debian 用户，运行 `apt install etables`

`ethtool` 命令。对于 CentOS/Fedora 用户，运行 `yum install etables ethtool` 命令。<h2

id="在安装过程中-

kubeadm-

一直等待控制平面&#x

在安装过程中，kubeadm 一直等待控制平面就绪</h2><p>如果你注意到

`kubeadm init` 在打印以下行后挂起：</p><div

class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh">[apiclient] Created API client, waiting for the control plane to become ready</code></pre></div><p>这可能

是由许多问题引起的。最常见的是：</p><p>网络连接问题。在继续之前，请检查你的计算机是否具有全部联通的网络连接。</p><p>kubelet 的默认 cgroup 驱动程序配置不同于 Docker 使用的配置。检查系统日志文件（例如 `/var/log/message`）或检查 `journalctl -u kubelet` 的输出。如果你看见以下内容：</p><div class="highlight"><pre

style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">error: failed to run Kubelet: failed to create kubelet: misconfiguration: kubelet cgroup driver: "systemd" is different from docker cgroup driver: "cgroupfs"</code></pre></div><p>有两种常见方法可解决 cgroup 驱动程序问题：</p><p>按照 此处 的说明再次安装 Docker。</p><p>更改 kubelet 配置以手动匹配 Docker cgroup 驱动程序，你可以参考

在主节点上配置 kubelet 要使用的 cgroup 驱动程序</p><p>控制平面上的 Docker 容器持续进入崩溃状态或（因其他原因）挂起。你可以运行 `docker ps` 命令来检查以及 `docker logs` 命令来检视每个容器的运行日志。</p><h2

id="当删除托管容器时-
kubeadm-阻塞">当删除托管容器时 kubeadm 阻塞</h2><p>如

果 Docker 停止并且不删除 Kubernetes 所管理的所有容器，可能发生以下情况：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-

size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-

```
lang="bash">sudo kubeadm reset <span style="color:#666">[</span>preflight<span style="color:#666">]</span> Running pre-flight checks <span style="color:#666">[</span>reset<span style="color:#666">]</span> Stopping the kubelet service <span style="color:#666">[</span>reset<span style="color:#666">]</span> Unmounting mounted directories in <span style="color:#b44">"/var/lib/kubelet"</span> <span style="color:#666">[</span>reset<span style="color:#666">]</span> Removing kubernetes-managed containers <span style="color:#666">(</span>block<span style="color:#666">)</span></pre></div><p>一个可行的解决方案是重新启动 Docker 服务，然后重新运行 <code>kubeadm reset</code> : </p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash">sudo systemctl restart docker.service sudo kubeadm reset </code></pre></div><p>检查 docker 的日志也可能有用 : </p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">journalctl -ul docker </code></pre></div><h2 id="pods-#x5904;#xE8E;-runcontainererror-crashloopbackoff-#x6216;#x8005;-error-#x72B6;#x6001;">Pods 处于 <code>RunContainerError</code>、<code>CrashLoopBackOff</code> 或者 <code>Error</code> 状态</h2><p>在 <code>kubeadm init</code> 命令运行后，系统中不应该有 pods 处于这类状态。</p><ul><li><p>在 <code>kubeadm init</code> 命令执行完后，如果有 pods 处于这些状态之一，请在 kubeadm 仓库提起一个 issue。<code>coredns</code> (或者 <code>kube-dns</code>) 应该处于 <code>Pending</code> 状态，直到你部署了网络解决方案为止。</p></li><li><p>如果在部署完网络解决方案之后，有 Pods 处于 <code>RunContainerError</code>、<code>CrashLoopBackOff</code> 或 <code>Error</code> 状态之一，并且<code>coredns</code> (或者 <code>kube-dns</code>) 仍处于 <code>Pending</code> 状态，那很可能是你安装的网络解决方案由于某种原因无法工作。你或许需要授予它更多的 RBAC 特权或使用较新的版本。请在 Pod Network 提供商的问题跟踪器中提交问题，然后在此处分类问题。</p></li><li><p>如果你安装的 Docker 版本早于 1.12.1，请在使用 <code>systemd</code> 来启动 <code>dockerd</code> 和重启 <code>docker</code> 时，删除 <code>MountFlags=slave</code> 选项。你可以在 <code>/usr/lib/systemd/system/docker.service</code> 中看到 MountFlags。MountFlags 可能会干扰 Kubernetes 挂载的卷，并使 Pods 处于 <code>CrashLoopBackOff</code> 状态。当 Kubernetes 不能找到 <code>var/run/secrets/kubernetes.io/serviceaccount</code> 文件时会发生错误。</p></li></ul><h2 id="coredns-#x6216;-kube-dns-#x505C;#xEDE;#x5728;-pending-#x72B6;#x6001;"><code>coredns</code> (或 <code>kube-dns</code>) 停滞在 <code>Pending</code> 状态</h2><p>这一行为是 <strong>预期之中</strong> 的，因为系统就是这么设计的。kubeadm 的网络供应商是中立的，因此管理员应该选择 <a href="/zh/docs/concepts/cluster-administration/addons/">安装 pod 的网络解决方案</a>。你必须完成 Pod 的网络配置，然后才能完全部署 CoreDNS。在网络被配置好之前，DNS 组件会一直处于 <code>Pending</code> 状态。</p><h2 id="hostport-#x670D;#x52A1;#x65E0;#x6CD5;#x5DE5;#x4F5C;"><code>HostPort</code> 服务无法工作</h2><p>此 <code>HostPort</code> 和 <code>HostIP</code> 功能是否可用取决于你的 Pod 网络配置。请联系 Pod 解决
```


方案的作者，以确认 `HostPort` 和 `HostIP` 功能是否可用。

已验证 Calico、Canal 和 Flannel CNI 驱动程序支持 HostPort。

有关更多信息，请参考 <https://github.com/containernetworking/plugins/blob/master/plugins/meta/portmap/README.md> CNI portmap 文档。

如果你的网络提供商不支持 portmap CNI 插件，你或许需要使用 [NodePort](/zh/docs/concepts/services-networking/service/#nodeport) 服务的功能 或者使用 `HostNetwork=true`。

无法通过其服务 IP 访问 Pod

许多网络附加组件尚未启用 [hairpin 模式](/zh/docs/tasks/debug-application-cluster/debug-service/#a-pod-cannot-reach-itself-via-service-ip) 该模式允许 Pod 通过其服务 IP 进行访问。这是与 [CNI](https://github.com/containernetworking/cni/issues/476) 有关的问题。请与网络附加组件提供商联系，以获取他们所提供的 hairpin 模式的最新状态。

如果你正在使用 VirtualBox (直接使用或者通过 Vagrant 使用)，你需要确保 `hostname -i` 返回一个可路由的 IP 地址。默认情况下，第一个接口连接不能路由的仅主机网络。解决方法是修改 `/etc/hosts`，请参考示例 [Vagrantfile](https://github.com/errordeveloper/k8s-playground/blob/22dd39dfc06111235620e6c4404a96ae146f26fd/Vagrantfile#L11)。

TLS 证书错误

以下错误指出证书可能不匹配。

```
# 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")
```

验证 `$HOME/.kube/config` 文件是否包含有效证书，并在必要时重新生成证书。在 kubeconfig 文件中的证书是 base64 编码的。该 `base64 -d` 命令可以用来解码证书，`openssl x509 -text -noout` 命令可以用于查看证书信息。

使用如下方法取消设置 `KUBECONFIG` 环境变量的值：

```
unset KUBECONFIG
```

或者将其设置为默认的 `KUBECONFIG` 位置：

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

另一个方法是覆盖 `kubeconfig` 的现有用户 "管理员"：

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

`u`

```
$(id -g style="color:#b8860b">$HOME /.kube/config </code> </pre> </div> </li> </ul> <h2 id="&#x5728;-vagrant-&#x4E2D;&#x4F7F;&#x7528;-flannel-&#x4F5C;&#x4E3A;-pod-&#x7F51;&#x7EDC;&#x65F6;&#x7684;&#x9ED8;&#x8BA4;-nic">在 Vagrant 中使用 flannel 作为 pod 网络时的默认 NIC </h2> <p>以下错误可能表明 Pod 网络中出现问题 : </p> <div class="highlight"> <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"> <code class="language-sh" data-lang="sh">Error from server <span style="color:#666"> </span> NotFound <span style="color:#666"> </span> : the server could not find the requested resource </code> </pre> </div> <ul> <li> <p>如果你正在 Vagrant 中使用 flannel 作为 pod 网络, 则必须指定 flannel 的默认接口名称. </p> <p>Vagrant 通常为所有 VM 分配两个接口. 第一个为所有主机分配了 IP 地址 <code>10.0.2.15 </code>, 用于获得 NATed 的外部流量. </p> <p>这可能会导致 flannel 出现问题, 它默认为主机上的第一个接口. 这导致所有主机认为它们具有 相同的公共 IP 地址. 为防止这种情况, 传递 <code>--iface eth1 </code> 标志给 flannel 以便选择第二个接口. </p> </li> </ul> <h2 id="&#x5BB9;&#x5668;&#x4F7F;&#x7528;&#x7684;&#x975E;&#x516C;&#x5171;-ip">容器使用的非公共 IP </h2> <p>在某些情况下 <code>kubectl logs </code> 和 <code>kubectl run </code> 命令或许会返回以下错误, 即便除此之外集群一切功能正常 : </p> <div class="highlight"> <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"> <code class="language-sh" data-lang="sh">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 </code> </pre> </div> <ul> <li> <p>这或许是由于 Kubernetes 使用的 IP 无法与看似相同的子网上的其他 IP 进行通信的缘故, 可能是由机器提供商的政策所导致的. </p> </li> <li> <p>Digital Ocean 既分配一个共有 IP 给 <code>eth0 </code>, 也分配一个私有 IP 在内部用作其浮动 IP 功能的锚点, 然而 <code>kubelet </code> 将选择后者作为节点的 <code>InternalIP </code> 而不是公共 IP </p> <p>使用 <code>ip addr show </code> 命令代替 <code>ifconfig </code> 命令去检查这种情况, 因为 <code>ifconfig </code> 命令 不会显示有问题的别名 IP 地址. 或者指定的 Digital Ocean 的 API 端口允许从 droplet 中 查询 anchor IP : </p> <div class="highlight"> <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"> <code class="language-sh" data-lang="sh">curl http://169.254.169.254/metadata/v1/interfaces/public/0/anchor ipv4/address </code> </pre> </div> <p>解决方法是通知 <code>kubelet </code> 使用哪个 <code>--node-ip </code>. 当使用 Digital Ocean 时, 可以是公网IP ( 分配给 <code>eth0 </code> 的 ), 或者是私网IP ( 分配给 <code>eth1 </code> 的 ). 私网 IP 是可选的. <a href="https://github.com/kubernetes/kubernetes/blob/release-1.13/cmd/kubeadm/app/apis/kubeadm/v1beta1/types.go">kubadm <code>NodeRegistrationOptions </code> 结构的 <code>KubeletExtraArgs </code> 部分 </a> 被用来处理这种情况. </p> <p>然后重启 <code>kubelet </code> : </p> <div class="highlight"> <pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"> <code class="language-shell" data-lang="shell">systemctl daemon-reload systemctl restart kubelet </code> </pre> </div> </li> </ul> <h2 id="coredns-pods-&#x6709;-crashloopbackoff-&#x6216;&#x8005;-error-
```

状态"><code>coredns</code> pods 有

<code>CrashLoopBackOff</code> 或者 <code>Error</code> 状态</

h2><p>如果有些节点运行的是旧版本的 Docker，同时启用了 SELinux，你或许会遇到 <code>coredns</code> pods 无法启动的情况。要解决此问题，你可以尝试以下

选项之一：</p><p>升级到 Docker 的较新版本。</p></

li><p>禁用 SELinux.</p><p>修改 <code>coredns</code> 部署以设置

<code>allowPrivilegeEscalation</code> 为 <code>true</code>：</p><div class="highlight"><pre style="background-color:#f8f8f8;-

moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-

lang="shell">kubectl -n kube-system get deployment coredns -o yaml |

\ <span

style="color:#b62;font-weight:700"/> sed 's/

allowPrivilegeEscalation: false/allowPrivilegeEscalation: true/g' |

\ <span

style="color:#b62;font-weight:700"/> kubectl apply -f - </code></pre></div><p>CoreDNS 处于 <code>CrashLoopBackOff</code> 时的另一个原因是

当 Kubernetes 中部署的 CoreDNS Pod 检测到环路时。有许多解决方法 可以避免在每次 CoreDNS 监测到循环并退出时，Kubernetes 尝试重启 CoreDNS Pod 的情况。</p><blockquote

class="warning callout"><div>警告： 禁用 SELinux 或设置 <code>allowPrivilegeEscalation</code> 为 <code>true</code> 可能会损害集群的安全性。</div></blockquote><h2 id="etcd-pods-

持续重启">etcd pods 持续重启</h2><p>如

果你遇到以下错误：</p><pre><code>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\""

运行 CentOS 7 就会出现这种问题。此版本的 Docker 会阻止 kubelet 在 etcd 容器中执行。</p><p>为解决此问题，请选择以下选项之一：</p><p>回滚到

早期版本的 Docker，例如 1.13.1-75</p><div class="highlight"><pre

style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:

4"><code class="language-shell" data-lang="shell">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 </code></pre></div><p>安装较新的推荐版本之一，例如 18.06:</p><div

class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-

tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">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 </code></

pre></div><h2

id="无法将以逗号分隔

component-extra-args-

标志内的参数">无法将以逗号

分隔的值列表传递给 <code>--component-extra-args</code> 标志内的参数</

h2><p><code>kubeadm init</code> 标志例如 <code>--component-extra-

`args` 允许你将自定义参数传递给像 kube-apiserver 这样的控制平面组件。然而，由于解析 (`mapStringString`) 的基础类型值，此机制将受到限制。

如果你决定传递一个支持多个逗号分隔值（例如 `--apiserver-extra-args "enable-admission-plugins=LimitRanger,NamespaceExists"`）参数，将出现 `flag: malformed pair, expect string=string` 错误。发生这种问题是因为参数列表 `--apiserver-extra-args` 预期的是 `key=value` 形式，而这里的 `NamespacesExists` 被误认为是缺少取值的键名。

一种解决方法是尝试分离 `key=value` 对，像这样：`--apiserver-extra-args "enable-admission-plugins=LimitRanger,enable-admission-plugins=NamespaceExists"` 但这将导致键 `enable-admission-plugins` 仅有值 `NamespaceExists`。

已知的解决方法是使用 kubeadm [配置](/zh/docs/setup/production-environment/tools/kubeadm/control-plane-flags/#apiserver-flags) 文件。

在节点被云控制管理器初始化之前，kube-proxy 就被调度了

在云环境场景中，可能出现在云控制管理器完成节点地址初始化之前，kube-proxy 就被调度到新节点了。这会导致 kube-proxy 无法正确获取节点的 IP 地址，并对管理负载平衡器的代理功能产生连锁反应。

在 kube-proxy Pod 中可以看到以下错误：

```
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
```

一种已知的解决方案是修补 kube-proxy DaemonSet，以允许在控制平面节点上调度它，而不管它们的条件如何，将其与其他节点保持隔离，直到它们的初始保护条件消除：

```
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" } ] } } } }
```

此问题的跟踪[在这里](https://github.com/kubernetes/kubeadm/issues/1027)。

NodeRegistration.Taints 字段在编组 kubeadm 配置时丢失

注意：这个[问题](https://github.com/kubernetes/kubeadm/issues/1358)仅适用于操控 kubeadm 数据类型的工具（例如，YAML 配置文件）。它将在 kubeadm API v1beta2 修复。

默认情况下，kubeadm 将 `node-role.kubernetes.io/master:NoSchedule` 污点应用于控制平面节点。如果你希望 kubeadm 不污染控制平面节点，并将

`InitConfiguration.NodeRegistration.Taints` 设置成空切片，则应在编组时省略该字段。如果省略该字段，则 kubeadm 将应用默认污点。

至少有两种解决方法：

- 使用 `node-role.kubernetes.io/master:PreferNoSchedule` 污点代替空切片。除非其他节点具有容量，[否则将在主节点上调度 Pods](/zh/docs/concepts/scheduling-eviction/taint-and-toleration/)。
- 在 kubeadm init 退出后删除污点：

```
tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-  
lang="shell">kubectl taint nodes NODE NAME node-role.kubernetes.io/  
master:NoSchedule- </code></pre></div></li></ol><div id="pre-  
footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您有帮助？  
</p><button class="btn btn-primary mb-4 feedback-yes">是</button>  
<button class="btn btn-primary mb-4 feedback-no">否</button><p  
class="feedback-response feedback-response hidden">感谢反馈。如果您有  
一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 <a  
target=" blank" rel="noopener" href="https://stackoverflow.com/questions/  
tagged/kubernetes">Stack Overflow</a>。在 GitHub 仓库上登记新的问题 <a  
class="feedback-link" target=" blank" rel="noopener" href="https://  
github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io">  
报告问题</a> 或者 <a class="feedback-link" target=" blank" rel="noopener"  
href="https://github.com/kubernetes/website/issues/new?  
title=Improvement%20for%20k8s.io">提出改进建议</a>。</p></  
div><script>const yes=document.querySelector('.feedback-yes');const  
no=document.querySelector('.feedback-  
no');document.querySelectorAll('.feedback--link').forEach(link=&gt;  
{link.href=link.href+window.location.pathname;});const  
sendFeedback=(value)=&gt;{if(!gtag){console.log('!gtag');}  
gtag('event','click',  
{'event category':'Helpful','event label':window.location.pathname,value});});const  
disableButtons=()=&gt;{yes.disabled=true;yes.classList.add('feedback-  
button disabled');no.disabled=true;no.classList.add('feedback-  
button disabled');};yes.addEventListener('click',()=&gt;  
{sendFeedback(1);disableButtons();document.querySelector('.feedback-  
response').classList.remove('feedback-  
response hidden');});no.addEventListener('click',()=&gt;  
{sendFeedback(0);disableButtons();document.querySelector('.feedback-  
response').classList.remove('feedback-response hidden');});</script><div  
class="text-muted mt-5 pt-3 border-top">最后修改 January 10, 2021 at 12:29  
AM PST: <a href="https://github.com/kubernetes/website/commit/  
4e0747d620cabb9d3c61112ff3a9116d866cf341">[zh] fixed link to  
configuring cgroup driver on control-plane node (4e0747d62)</a></div></  
div><div class="td-content"><h1>使用 kubeadm 创建集群</h1><p>创  
建一个符合最佳实践的最小化 Kubernetes 集群。事实上，你可以使用  
<code>kubeadm</code> 配置一个通过 <a href="https://kubernetes.io/blog/  
2017/10/software-conformance-certification">Kubernetes 一致性测试</a> 的  
集群。 <code>kubeadm</code> 还支持其他集群生命周期功能，例如 <a href="/  
zh/docs/reference/access-authn-authz/bootstrap-tokens/">启动引导令牌</a>  
和集群升级。</p><p>kubeadm 工具很棒，如果你需要：</p><ul><li>一个尝试  
Kubernetes 的简单方法。</li><li>一个现有用户可以自动设置集群并测试其应用程序  
的途径。</li><li>其他具有更大范围的生态系统和/或安装工具中的构建模块。</li></  
ul><p>你可以在各种机器上安装和使用 <code>kubeadm</code>：笔记本电脑，  
一组云服务器，Raspberry Pi 等。无论是部署到云还是本地，你都可以将  
<code>kubeadm</code> 集成到预配置系统中，例如 Ansible 或 Terraform。</  
p><h2 id="#&x51C6;&x5907;&x5F00;&x59CB;">准备开始</h2><p>要  
遵循本指南，你需要：</p><ul><li>一台或多台运行兼容 deb/rpm 的 Linux 操作系  
统的计算机；例如：Ubuntu 或 CentOS。</li><li>每台机器 2 GB 以上的内存，内
```

存不足时应用会受限制。

- 用作控制平面节点的计算机上至少有2个 CPU。
- 集群中所有计算机之间具有完全的网络连接。你可以使用公共网络或专用网络。

你还需要使用可以在新集群中部署特定 Kubernetes 版本对应的 `kubeadm`。

[Kubernetes 版本及版本倾斜支持策略](/zh/docs/setup/release/version-skew-policy/#supported-versions) 适用于 `kubeadm` 以及整个 Kubernetes。查阅该策略以了解支持哪些版本的 Kubernetes 和 `kubeadm`。该页面是为 Kubernetes v1.20 编写的。

`kubeadm` 工具的整体功能状态为一般可用性 (GA)。一些子功能仍在积极开发中。随着工具的发展, 创建集群的实现可能会略有变化, 但总体实现应相当稳定。

说明： 根据定义, 在 `kubeadm`

`alpha` 下的所有命令均在 alpha 级别上受支持。

目标

- 安装单个控制平面的 Kubernetes 集群
- 在集群上安装 Pod 网络, 以便你的 Pod 可以相互连通

操作指南

在你的主机上安装 `kubeadm`

查看 [安装 `kubeadm`](/zh/docs/setup/production-environment/tools/kubeadm/install-kubeadm/)。

说明： 如果你已经安装了 `kubeadm`, 执行 `apt-get update && apt-get upgrade` 或 `yum update` 以获取 `kubeadm` 的最新版本。

升级时, `kubelet` 每隔几秒钟重新启动一次, 在 `crashloop` 状态中等待 `kubeadm` 发布指令。 `crashloop` 状态是正常现象。初始化控制平面后, `kubelet` 将正常运行。

初始化控制平面节点

控制平面节点是运行控制平面组件的机器, 包括

[etcd](#)、[Kubernetes](#)

[Kubernetes API](#) (集群数据库) 和 [API Server](#) (命令行工具 `kubectl` 与之通信)。

如果计划将单个控制平面 `kubeadm` 集群升级成高可用, 你应该指定 `--control-plane-endpoint` 为所有控制平面节点设置共享端点。端点可以是负载均衡器的 DNS 名称或 IP 地址。

选择一个 Pod 网络插件, 并验证是否需要为 `kubeadm init` 传递参数。根据你选择的第三方网络插件, 你可能需要设置 `--pod-network-cidr` 的值。请参阅 [安装 Pod 网络附加组件](#)。

(可选) 从版本 1.14 开始, `kubeadm` 尝试使用一系列众所周知的域套接字路径来检

测 Linux 上的容器运行时。要使用不同的容器运行时，或者如果在预配置的节点上安装了多个容器，请为 `kubeadm init` 指定 `--cri-socket` 参数。请参阅[安装运行时](/zh/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#installing-runtime)。

(可选) 除非另有说明，否则 `kubeadm` 使用与默认网关关联的网络接口来设置此控制平面节点 API server 的广播地址。要使用其他网络接口，请为 `kubeadm init` 设置 `--apiserver-advertise-address=<ip-address>` 参数。要部署使用 IPv6 地址的 Kubernetes 集群，必须指定一个 IPv6 地址，例如 `--apiserver-advertise-address=fd00::101`。

(可选) 在 `kubeadm init` 之前运行 `kubeadm config images pull`，以验证与 gcr.io 容器镜像仓库的连通性。

要初始化控制平面节点，请运行：

```
kubeadm init <args>
```

关于 `apiserver-advertise-address` 和 `ControlPlaneEndpoint` 的注意事项

`--apiserver-advertise-address` 可用于为控制平面节点的 API server 设置广播地址，`--control-plane-endpoint` 可用于为所有控制平面节点设置共享端点。

`--control-plane-endpoint` 允许 IP 地址和可以映射到 IP 地址的 DNS 名称。请与你的网络管理员联系，以评估有关此类映射的可能解决方案。

这是一个示例映射：

```
192.168.0.102 cluster-endpoint
```

其中 `192.168.0.102` 是此节点的 IP 地址，`cluster-endpoint` 是映射到该 IP 的自定义 DNS 名称。这将允许你将 `--control-plane-endpoint=cluster-endpoint` 传递给 `kubeadm init`，并将相同的 DNS 名称传递给 `kubeadm join`。稍后你可以修改 `cluster-endpoint` 以指向高可用性方案中的负载均衡器的地址。

`kubeadm` 不支持将没有 `--control-plane-endpoint` 参数的单个控制平面集群转换为高可用性集群。

更多信息

有关 `kubeadm init` 参数的更多信息，请参见[kubeadm 参考指南](/zh/docs/reference/setup-tools/kubeadm/kubeadm-init/)。

要使用配置文件配置 `kubeadm init` 命令，请参见[带配置文件使用 kubeadm init](/zh/docs/reference/setup-tools/kubeadm/kubeadm-init/#config-file)。

要自定义控制平面组件，包括可选的对控制平面组件和 etcd 服务器的活动探针提供 IPv6 支持，请参阅[自定义参数](/zh/docs/setup/production-environment/tools/kubeadm/control-plane-flags/)。

要再次运行 `kubeadm init`，你必须首先[卸载集群](#tear-down)。

如果将具有不同架构的节点加入集群，请确保已部署的 DaemonSet 对这种体系结构具有容器镜像支持。

`kubeadm init` 首先运行一系列预检查以确保机器准备运行 Kubernetes。这些预检查会显示警告并在错误时退出。然后 `kubeadm init` 下载并安装集群控制平面组件。这可能会需要几分钟。完成之后你应该看到：

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

要使非 root 用户可以运行 kubectl，请运行以下命令，它们也是 kubectl init 输出的一部分：

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

或者，如果你是 root 用户，则可以运行：

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

记录 kubectl init 输出的 kubectl join 命令。你需要此命令将节点加入集群。

令牌用于控制平面节点和加入节点之间的相互身份验证。这里包含的令牌是密钥。确保它的安全，因为拥有此令牌的任何人都可以将经过身份验证的节点添加到你的集群中。可以使用 kubectl token 命令列出，创建和删除这些令牌。请参阅 [kubeadm 参考指南](/zh/docs/reference/setup-tools/kubeadm/kubeadm-token/)。

安装 Pod 网络附加组件

注意：

本节包含有关网络设置和部署顺序的重要信息。在继续之前，请仔细阅读所有建议。

你必须部署一个基于 Pod 网络插件的

容器网络接口 (CNI)，以便你的 Pod 可以相互通信。在安装网络之前，集群 DNS (CoreDNS) 将不会启动。

- 注意你的 Pod 网络不得与任何主机网络重叠：如果有重叠，你很可能遇到问题。（如果你发现网络插件的首选 Pod 网络与某些主机网络之间存在冲突，则应考虑使用一个合适的 CIDR 块来代替，然后在执行 kubectl init 时使用 --pod-network-cidr 参数并在你的网络插件的 YAML 中替换它）。
- 默认情况下，kubectl 将集群设置为使用 and 强制使用 RBAC（基于角色的访问控制）。确保你的 Pod 网络插件支持 RBAC，以及用于部署它的 manifests 也是如此。
- 如果要为集群使用 IPv6（双协议栈或仅单协议栈 IPv6 网络），请确保你的 Pod 网络插件支持 IPv6。IPv6 支持已在 CNI

[containernetworking/cni/releases/tag/v0.6.0">v0.6.0](#) 版本中添加。

说明：目前 Calico 是 kubeadm 项目中执行 e2e 测试的唯一 CNI 插件。如果你发现与 CNI 插件相关的问题，应在其各自的问题跟踪器中记录而不是在 kubeadm 或 kubernetes 问题跟踪器中记录。

一些外部项目为 Kubernetes 提供使用 CNI 的 Pod 网络，其中一些还支持 [网络策略](#)。请参阅实现 [Kubernetes 网络模型](#) 的附加组件列表。

你可以使用以下命令在控制平面节点或具有 kubeconfig 凭据的节点上安装 Pod 网络附加组件：

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

每个集群只能安装一个 Pod 网络。安装 Pod 网络后，您可以通过在 `kubectl get pods --all-namespaces` 输出中检查 CoreDNS Pod 是否 `Running` 来确认其是否正常运行。一旦 CoreDNS Pod 启用并运行，你就可以继续加入节点。如果您的网络无法正常工作或 CoreDNS 不在“运行中”状态，请查看 `kubeadm` 的 [故障排除指南](#)。

控制平面节点隔离

默认情况下，出于安全原因，你的集群不会在控制平面节点上调度 Pod。如果你希望能够在控制平面节点上调度 Pod，例如用于开发的单机 Kubernetes 集群，请运行：

```
kubectl taint nodes --all
```

输出看起来像：

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

这将从任何拥有 `node-role.kubernetes.io/master` 标记的节点中移除该标记，包括控制平面节点，这意味着调度程序将能够在任何地方调度 Pods。

加入节点

节点是你的工作负载（容器和 Pod 等）运行的地方。要将新节点添加到集群，请对每台计算机执行以下操作：

- SSH 到机器
- 成为 root（例如 `sudo su`）
- 运行 `kubeadm init` 输出的命令。例如：

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

如果没有令牌，可以通过在控制平面节点上运行以下命令来获取令牌：

```
kubeadm token list
```

输出类似于以下内容：

```
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

默认情况下，令牌会在24小时后过期。如果要在当前令牌

过期后将节点加入集群，则可以通过在控制平面节点上运行以下命令来创建新令牌：

```
<div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash">kubeadm token create</code></pre></div><p>输出类似于以下内容：</p><pre><code class="language-console" data-lang="console">5didvk.d09sbcov8ph2amjw</code></pre></div>
```

如果你没有 `--discovery-token-ca-cert-hash` 的值，则可以通过在控制平面节点上执行以下命令链来获取它：

```
<div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash">openssl x509 -pubkey -in /etc/kubernetes/pki/ca.crt | openssl rsa -pubin -outform der 2>&#x2D;/dev/null |<span style="color:#b62;font-weight:700">\</span><span style="color:#b62;font-weight:700"/> openssl dgst -sha256 -hex | sed <span style="color:#b44">'s/^.* //'</span></code></pre></div><p>输出类似于以下内容：</p><pre><code class="language-console" data-lang="console">8cb2de97839780a412b93877f8507ad6c94f73add17d5d7058e91741c9</code></pre></div>
```

说明： 要为 `<control-plane-host>;<control-plane-port>` 指定 IPv6 元组，必须将 IPv6 地址括在方括号中，例如：

```
<code>[fd00::101]:2073</code></div></blockquote><p>输出应类似于：</p></div>
```

```
<pre><code>[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.</code></pre><p>几秒钟后，当你在控制平面节点上执行<code>kubectl get nodes</code>，你会注意到该节点出现在输出中。</p><h3 id="&#x53EF;&#x9009;-&#x4ECE;&#x63A7;&#x5236;&#x5E73;&#x9762;&#x8282;&#x70B9;&#x4EE5;&#x5085;">选）从控制平面节点以外的计算机控制集群</h3><p>为了使 kubectl 在其他计算机（例如笔记本电脑）上与你的集群通信，你需要将管理员 kubeconfig 文件从控制平面节点复制到工作站，如下所示：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash">scp root@&lt;control-plane-host>;/etc/kubernetes/admin.conf . kubectl --kubeconfig ./admin.conf get nodes</code></pre></div><blockquote class="note callout"><div><strong>说明：</strong><p>上面的示例假定为 root 用户启用了 SSH 访问。如果不是这种情况，你可以使用<code>scp</code> 将 admin.conf 文件复制给其他允许访问的用户。</p><p>admin.conf 文件为用户提供了对集群的超级用户特权。该文件应谨慎使用。对于普通用户，建议生成一个你为其授予特权的唯一证书。你可以使用<code>kubeadm alpha kubeconfig user --client-name &lt;CN></code> 命令执行此操作。该命令会将 KubeConfig 文件打印到 STDOUT，你应该将其保存到文件并分发给用户。之后，使用<code>kubectl create (cluster)rolebinding</code> 授予特权。</p></div></blockquote><h3 id="&#x53EF;&#x9009;-&#x5C06;api&#x670D;&#x52A1;&#x5668;&#x4EE3;&#x7406;&#x5230;&#x672C;&#x5085;">选）将 API 服务器代理到本地主机</h3><p>如果要从集群外部连接到 API 服务器，则可以使用<code>kubectl proxy</code>：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-bash" data-lang="bash">scp root@&lt;control-plane-host>;/etc/kubernetes/admin.conf . kubectl --kubeconfig ./
```

admin.conf proxy

```
</pre></div><p>你现在可以在本地访问API服务器 http://localhost:8001/api/v1</p><h2 id="tear-down">清理</h2><p>如果你在集群中使用了一次性服务器进行测试，则可以关闭这些服务器，而无需进一步清理。你可以使用 kubectl config delete-cluster 删除对集群的本地引用。</p><p>但是，如果要更干净地取消配置群集，则应首先<a href="/docs/reference/generated/kubectl/kubectl-commands#drain">清空节点</a>并确保该节点为空，然后取消配置该节点。</p><h3 id="#x5220;&#x9664;&#x8282;&#x70B9;">删除节点</h3><p>使用适当的凭证与控制平面节点通信，运行：kubeadm 安装的状态：kubeadm init 或 kubeadm join 并加上适当的参数。</p><h3 id="#x6E05;&#x7406;&#x63A7;&#x5236;&#x5E73;&#x9762;">清理控制平面</h3><p>你可以在控制平面主机上使用 kubeadm reset 来触发尽力而为的清理。</p><p>有关此子命令及其选项的更多信息，请参见<a href="/zh/docs/reference/setup-tools/kubeadm/kubeadm-reset/"><code>kubeadm reset</code></a>参考文档。</p><h2 id="what-next">下一步</h2><ul><li>使用 <a href="https://github.com/heptio/sonobuoy">Sonobuoy</a> 验证集群是否正常运行</li><li><a id="lifecycle">有关使用kubeadm升级集群的详细信息，请参阅<a href="/zh/docs/tasks/administer-cluster/kubeadm/kubeadm-upgrade/">升级 kubeadm 集群</a>。</a></li><li>在<a href="/zh/docs/reference/setup-tools/kubeadm/kubeadm">kubeadm</a>参考文档</a>中了解有关高级 <code>kubeadm</code>用法的信息</li><li>了解有关Kubernetes<a href="/zh/docs/concepts/">概念</a>和<a href="/zh/docs/reference/kubectl/overview/"><code>kubectl</code></a>的更多信息。</li><li>有关Pod网络附加组件的更多列表，请参见<a href="/zh/docs/concepts/cluster-administration/networking/">集群网络</a>页面。</li><li><a id="other-addons">请参阅<a href="/zh/docs/concepts/cluster-administration/addons/">附加组件列表</a>以探索其他附加组件，包括用于 Kubernetes 集群的日志记录，监视，网络策略，可视化和控制的工具。</a></li><li>配置集群如何处理集群事件的日志以及在Pods中运行的应用程序。有关所涉及
```

内容的概述，请参见[日志架构](/zh/docs/concepts/cluster-administration/logging/)。

反馈

- 有关 bugs，访问 [kubeadm GitHub issue tracker](https://github.com/kubernetes/kubeadm/issues)

- 有关支持，访问 [#kubeadm](https://kubernetes.slack.com/messages/kubeadm/)

Slack 频道

- General SIG 集群生命周期开发 Slack 频道: [#sig-cluster-lifecycle](https://kubernetes.slack.com/messages/sig-cluster-lifecycle/)

- SIG 集群生命周期 [SIG information](https://github.com/kubernetes/community/tree/master/sig-cluster-lifecycle#readme)

- SIG 集群生命周期邮件列表: groups.google.com/forum/#!forum/kubernetes-sig-cluster-lifecycle

版本倾斜政策

版本 v1.20 的 kubeadm 工具可以使用版本 v1.20 或 v1.19 的控制平面部署集群。kubeadm v1.20 还可以升级现有的 kubeadm 创建的 v1.19 版本的集群。

由于没有未来，kubeadm CLI v1.20 可能会或可能无法部署 v1.21 集群。

这些资源提供了有关 kubelet 与控制平面以及其他 Kubernetes 组件之间受支持的版本倾斜的更多信息：

- [版本和版本偏斜政策](/zh/docs/setup/release/version-skew-policy/)

- [Kubeadm-specific 安装指南](/zh/docs/setup/production-environment/tools/kubeadm/install-kubeadm/#installing-kubeadm-kubelet-and-kubectl)

局限性

集群弹性

此处创建的集群具有单个控制平面节点，运行单个 etcd 数据库。这意味着如果控制平面节点发生故障，你的集群可能会丢失数据并且可能需要从头开始重新创建。

解决方法:

- 定期 [备份 etcd](https://coreos.com/etcd/docs/latest/admin_guide.html)。

kubeadm 配置的 etcd 数据目录位于控制平面节点上的 `/var/lib/etcd` 中。

- 使用多个控制平面节点。你可以阅读 [可选的高可用性拓扑](/zh/docs/setup/production-environment/tools/kubeadm/ha-topology/)

选择集群拓扑提供的 [高可用性](/zh/docs/setup/production-environment/tools/kubeadm/high-availability/)。

平台兼容性

kubeadm deb/rpm 软件包和二进制文件是为 amd64，arm (32-bit)，arm64，ppc64le 和 s390x 构建的遵循 [多平台提案](https://github.com/kubernetes/community/blob/master/contributors/design-proposals/multi-platform.md)。

从 v1.12 开始还支持用于控制平面和附加组件的多平台容器镜像。

只有一些网络提供商为所有平台提供解决方案。请查阅上方的 网络提供商清单或每个提供商的文档以确定提供商是否支持你选择的平台。

故障排除

如果你在使用 kubeadm 时遇到困难，请查阅我们的 [故障排除文档](/zh/docs/setup/production-environment/tools/kubeadm/troubleshooting-kubeadm/)。

反馈

此页是否对您有帮助？

是

否

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问

[Stack Overflow](https://stackoverflow.com/questions/tagged/kubernetes)。在 GitHub 仓库上登记新的问题

[报告问题](https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io) 或者

[报告问题](https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io) 或者

href="https://github.com/kubernetes/website/issues/new?title=Improvement%20for%20k8s.io">提出改进建议.</p></div><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><div class="text-muted mt-5 pt-3 border-top">最后修改 December 14, 2020 at 7:35 PM PST: modify `kubeadm init` link (7bfc91534)</div></div><div class="td-content"><h1>使用 kubeadm 定制控制平面配置</h1><div style="margin-top:10px;margin-bottom:10px">FEATURE STATE: <code>Kubernetes 1.12 [stable]</code></div><p>kubeadm <code>ClusterConfiguration</code> 对象公开了 <code>extraArgs</code> 字段，它可以覆盖传递给控制平面组件（如 APIServer、ControllerManager 和 Scheduler）的默认参数。各组件配置使用如下字段定义：</p><code>apiServer</code><code>controllerManager</code><code>scheduler</code><p><code>extraArgs</code> 字段由 <code>key: value</code> 对组成。要覆盖控制平面组件的参数:</p>将适当的字段添加到配置中。向字段添加要覆盖的参数值。用 <code>--config <YOUR CONFIG YAML></code> 运行 <code>kubeadm init</code>。<p>有关配置中的每个字段的详细信息，您可以导航到我们的 API 参考页面。</p><blockquote class="note callout"><div>说明：<p>您可以通过运行 <code>kubeadm config print init-defaults</code> 并将输出保存到您选择的文件中，以默认值形式生成 <code>ClusterConfiguration</code> 对象。</p></div></blockquote><h2 id="apiserver-参数">APIServer 参数</h2><p>有关详细信息，请参阅 kube-apiserver 参考文档。</p><p>使用示例：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml">apiVersion: kubeadm.k8s.io/v1beta2 kind: ClusterConfiguration kubernetesVersion: v1.16.0 </pre></div></div>

```
span><span style="color:#bbb"/><span style="color:green;font-weight:700">apiServer</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">extraArgs</span>:<span style="color:#bbb"> </span><span style="color:green;font-weight:700">advertise-address</span>:<span style="color:#bbb"> </span><span style="color:#666">192.168.0.103</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">anonymous-auth</span>:<span style="color:#bbb"> </span><span style="color:#b44">"false"</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">enable-admission-plugins</span>:<span style="color:#bbb"> </span><span>AlwaysPullImages,DefaultStorageClass<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">audit-log-path</span>:<span style="color:#bbb"> </span><span>/home/johndoe/audit.log<span style="color:#bbb"> </span></code></pre></div><h2 id="controllermanager">ControllerManager 参数</h2><p>有关详细信息，请参阅 <a href="/docs/reference/command-line-tools-reference/kube-controller-manager/">kube-controller-manager 参考文档</a>。</p><p>使用示例：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml"><span style="color:green;font-weight:700">apiVersion</span>:<span style="color:#bbb"> </span>kubeadm.k8s.io/v1beta2<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">kind</span>:<span style="color:#bbb"> </span>ClusterConfiguration<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">kubernetesVersion</span>:<span style="color:#bbb"> </span>v1.16.0<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">controllerManager</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">extraArgs</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">cluster-signing-key-file</span>:<span style="color:#bbb"> </span><span>/home/johndoe/keys/ca.key<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">bind-address</span>:<span style="color:#bbb"> </span><span style="color:#666">0.0.0.0</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">deployment-controller-sync-period</span>:<span style="color:#bbb"> </span><span style="color:#b44">"50"</span><span style="color:#bbb"> </span></code></pre></div><h2 id="scheduler">Scheduler 参数</h2><p>有关详细信息，请参阅 <a href="/docs/reference/command-line-tools-reference/kube-scheduler/">kube-scheduler 参考文档</a>。</p><p>使用示例：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml"><span style="color:green;font-weight:700">apiVersion</span>
```

```
span>:<span style="color:#bbb"> </span>kubeadm.k8s.io/v1beta2<span
style="color:#bbb"> </span><span style="color:#bbb"/><span
style="color:green;font-weight:700">kind</span>:<span
style="color:#bbb"> </span>ClusterConfiguration<span
style="color:#bbb"> </span><span style="color:#bbb"/><span
style="color:green;font-weight:700">kubernetesVersion</span>:<span
style="color:#bbb"> </span>v1.16.0<span style="color:#bbb"> </
span><span style="color:#bbb"/><span style="color:green;font-weight:
700">scheduler</span>:<span style="color:#bbb"> </span><span
style="color:#bbb"> </span><span style="color:green;font-weight:
700">extraArgs</span>:<span style="color:#bbb"> </span><span
style="color:#bbb"> </span><span style="color:green;font-weight:
700">address</span>:<span style="color:#bbb"> </span><span
style="color:#666">0.0.0.0</span><span style="color:#bbb"> </
span><span style="color:#bbb"> </span><span style="color:green;font-
weight:700">config</span>:<span style="color:#bbb"> </span>/home/
johndoe/schedconfig.yaml<span style="color:#bbb"> </span><span
style="color:#bbb"> </span><span style="color:green;font-weight:
700">kubeconfig</span>:<span style="color:#bbb"> </span>/home/
johndoe/kubeconfig.yaml<span style="color:#bbb"> </span></code></
pre></div><div id="pre-footer"><h2>反馈</h2><p class="feedback-
prompt">此页是否对您有帮助? </p><button class="btn btn-primary mb-4
feedback--yes">是</button> <button class="btn btn-primary mb-4
feedback--no">否</button><p class="feedback--response feedback--
response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、
需要答案的问题, 可以访问 <a target=" blank" rel="noopener" href="https://
stackoverflow.com/questions/tagged/kubernetes">Stack Overflow</a>. 在
GitHub 仓库上登记新的问题 <a class="feedback--link" target=" blank"
rel="noopener" href="https://github.com/kubernetes/website/issues/new?
title=Issue%20with%20k8s.io">报告问题</a> 或者 <a class="feedback--link"
target=" blank" rel="noopener" href="https://github.com/kubernetes/
website/issues/new?title=Improvement%20for%20k8s.io">提出改进建议</
a>.</p></div><script>const yes=document.querySelector('.feedback--
yes');const no=document.querySelector('.feedback--
no');document.querySelectorAll('.feedback--link').forEach(link=&gt;
{link.href=link.href+window.location.pathname;});const
sendFeedback=(value)=&gt;{if(!gtag){console.log('!gtag');}
gtag('event','click',
{'event category':'Helpful','event label':window.location.pathname,value});};const
disableButtons=()=&gt;{yes.disabled=true;yes.classList.add('feedback--
button disabled');no.disabled=true;no.classList.add('feedback--
button disabled');};yes.addEventListener('click',)=&gt;
{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><div
class="text-muted mt-5 pt-3 border-top">最后修改 June 01, 2020 at 9:23 AM
PST: <a href="https://github.com/kubernetes/website/commit/
4b35d4d401e6f637815d3421fabe0dc0dc0917c3">add zh pages
(4b35d4d40)</a></div></div><div class="td-content"><h1>高可用拓扑选项
```

本页面介绍了配置高可用（HA）Kubernetes 集群拓扑的两个选项。

您可以设置 HA 集群：

- 使用堆叠（stacked）控制平面节点，其中 etcd 节点与控制平面节点共存
- 使用外部 etcd 节点，其中 etcd 在与控制平面不同的节点上运行

在设置 HA 集群之前，您应该仔细考虑每种拓扑的优缺点。

说明： kubeadm 静态引导 etcd 集群。阅读 [etcd 集群指南](https://github.com/etcd-io/etcd/blob/release-3.4/Documentation/op-guide/clustering.md#static) 以获得更多详细信息。

堆叠（Stacked）etcd 拓扑

堆叠（Stacked）HA 集群是一种这样的[网络拓扑](https://en.wikipedia.org/wiki/Network_topology)，其中 etcd 分布式数据存储集群堆叠在 kubeadm 管理的控制平面节点上，作为控制平面的一个组件运行。

每个控制平面节点运行 `kube-apiserver`，`kube-scheduler` 和 `kube-controller-manager` 实例。

`kube-apiserver` 使用负载均衡器暴露给工作节点。

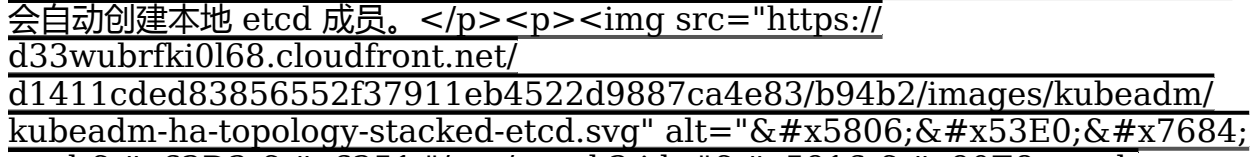
每个控制平面节点创建一个本地 etcd 成员（member），这个 etcd 成员只与该节点的 `kube-apiserver` 通信。这同样适用于本地 `kube-controller-manager` 和 `kube-scheduler` 实例。

这种拓扑将控制平面和 etcd 成员耦合在同一节点上。相对使用外部 etcd 集群，设置起来更简单，而且更易于副本管理。

然而，堆叠集群存在耦合失败的风险。如果一个节点发生故障，则 etcd 成员和控制平面实例都将丢失，并且冗余会受到影响。您可以通过添加更多控制平面节点来降低此风险。

因此，您应该为 HA 集群运行至少三个堆叠的控制平面节点。

这是 kubeadm 中的默认拓扑。当使用 `kubeadm init` 和 `kubeadm join --control-plane` 时，在控制平面节点上会自动创建本地 etcd 成员。



外部 etcd 拓扑

具有外部 etcd 的 HA 集群是一种这样的[网络拓扑](https://en.wikipedia.org/wiki/Network_topology)，其中 etcd 分布式数据存储集群在独立于控制平面节点的其他节点上运行。

就像堆叠的 etcd 拓扑一样，外部 etcd 拓扑中的每个控制平面节点都运行 `kube-apiserver`，`kube-scheduler` 和 `kube-controller-manager` 实例。同样，`kube-apiserver` 使用负载均衡器暴露给工作节点。但是，etcd 成员在不同的主机上运行，每个 etcd 主机与每个控制平面节点的 `kube-apiserver` 通信。

这种拓扑结构解耦了控制平面和 etcd 成员。因此，它提供了一种 HA 设置，其中失去控制平面实例或者 etcd 成员的影响较小，并且不会像堆叠的 HA 拓扑那样影响集群冗余。

但是，此拓扑需要两倍于堆叠 HA 拓扑的主机数量。

具有此拓扑的 HA 集群至少需要三个用于控制平面节点的主机和三个用于 etcd 节点的主机。



接下来

- [使用 kubeadm 设置高可用集群](/zh/docs/setup/production-environment/tools/kubeadm/high-availability/)

反馈

此页是否对您有帮助？

yes">是</button> <button class="btn btn-primary mb-4 feedback--no">否</button><p class="feedback--response feedback--response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 Stack Overflow。在 GitHub 仓库上登记新的问题 报告问题 或者 提出改进建议。</p></div><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><div class="text-muted mt-5 pt-3 border-top">最后修改 September 07, 2020 at 1:35 PM PST: [zh] Fix links in setup section (3fda142df)</div></div><div class="td-content"><h1>利用 kubeadm 创建高可用集群</h1><p>本文讲述了使用 kubeadm 设置一个高可用的 Kubernetes 集群的两种不同方式：</p>使用堆控制平面节点。这种方法所需基础设施较少。etcd 成员和控制平面节点位于同一位置。使用外部集群。这种方法所需基础设施较多。控制平面的节点和 etcd 成员是分开的。<p>在下一步之前，您应该仔细考虑哪种方法更好的满足您的应用程序和环境的需求。这是对比文档 讲述了每种方法的优缺点。</p><p>如果您在安装 HA 集群时遇到问题，请在 kubeadm 问题跟踪里向我们提供反馈。</p><p>您也可以阅读 升级文件</p><blockquote class="caution callout"><div>注意： 这篇文档没有讲述在云提供商上运行集群的问题。在云环境中，此处记录的方法不适用于类型为 LoadBalancer 的服务对象，或者具有动态的 PersistentVolumes。</div></blockquote><h2 id="#&x51C6;&x5907;&x5F00;&x59CB;">准备开始</h2><p>对于这两种方法，您都需要以下基础设施：</p>配置三台机器 kubeadm 的最低要求 给主节点配置三台机器 kubeadm 的最低要求 给工作节点在集群中，所有计算机之间的完全网络连接（公网或私网）所有机器上的 sudo 权限每台设备对系统中所有节点的 SSH 访问在所有机器上安装

`kubeadm` 和 `kubelet` , `kubect`

仅对于外部 etcd 集群来说, 您还需要 :

给 etcd 成员使用的另外三台机器

这两种方法的第一步

`kube-apiserver` 创建负载均衡器

说明 : 使用负载均衡器需要许多配置。您的集群搭建可能需要不同的配置。下面的例子只是其中的一方面配置。

创建一个名为 `kube-apiserver` 的负载均衡器解析

DNS。在云环境中, 应该将控制平面节点放置在 TCP 后面转发负

载平衡。该负载均衡器将流量分配给目标列表中所有运行状况良好的控制平面节点。健康检查 `apiserver` 是在 `kube-apiserver` 监听端口(默认值 `:6443`)上的一个 TCP 检查。

不建议在云环境中直接使用 IP 地址。负载均衡器必须能够在 `apiserver` 端口上与所有控制平面节点通信。它还必须允许其监听端口的传入流量。

确保负载均衡器的地址始终匹配 `kubeadm` 的 `ControlPlaneEndpoint` 地址。

阅读<https://github.com/kubernetes/kubeadm/blob/master/docs/ha-considerations.md#options-for-software-load-balancing>以获取更多详细信息。

添加第一个控制平面节点到负载均衡器并测试连接 :

```
nc -v LOAD_BALANCER_IP PORT
```

由于 `apiserver` 尚未运行, 预期会出现一个连接拒绝错误。然而超时意味着负载均衡器不能和控制平面节点通信。如果发生超时, 请重新配置负载均衡器与控制平面节点进行通信。

将其余控制平面节点添加到负载均衡器目标组。

使用堆控制平面和 etcd 节点

控制平面节点的第一步

```
sudo kubeadm init --control-plane-endpoint
```

```
"LOAD_BALANCER_DNS:LOAD_BALANCER_PORT"
```

```
--upload-certs
```

您可以使用 `--kubernetes-version` 标志来设置要使用的 Kubernetes 版本。建议将 `kubeadm`、`kebelet`、`kubect` 和 Kubernetes 的版本匹配。

这个 `--control-plane-endpoint` 标志应该被设置成负载均衡器的地址或 DNS 和端口。

这个 `--upload-certs` 标志用来将在所有控制平面实例之间的共享证书上传到集群。如果正好相反, 你更喜欢手动地通过控制平面节点或者使用自动化工具复制证书, 请删除此标志并参考如下部分[证书分配手册](#manual-certs)。

说明 : 标志 `kubeadm init`、`--config` 和 `--certificate-key` 不能混合使用, 因此如果您要使用

<https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta2> `kubeadm` 配置, 您必须在相应的配置文件 (位于 `InitConfiguration` 和 `JoinConfiguration` :

`controlPlane`) 添加 `certificateKey` 字段。

说明： 一些 CNI 网络插件如 Calico 需要 CIDR 例如 `192.168.0.0/16` 和一些像 Weave 没有。参考 [CNI 网络文档](/zh/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#pod-network)。通过传递 `--pod-network-cidr` 标志添加 pod CIDR，或者您可以使用 kubeadm 配置文件，在 `ClusterConfiguration` 的 `networking` 对象下设置 `podSubnet` 字段。

命令完成后，您应该会看到类似以下内容：

```
... 现在，您可以通过在根目录上运行以下命令来加入任意数量的控制平面节点：
kubeadm join 192.168.0.200:6443 --token 9vr73a.a8uxyaju799qwdjv --discovery-token-ca-cert-hash
sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720eff5359e26aec866
--control-plane --certificate-key
f8902e114ef118304e561c3ecd4d0b543adc226b7a07f675f56564185ffe0c07
```

请注意，证书密钥可以访问集群内敏感数据，请保密！为了安全起见，将在两个小时内删除上传的证书；如有必要，您可以使用 kubeadm 初始化上传证书阶段，之后重新加载证书。然后，您可以通过在根目录上运行以下命令来加入任意数量的工作节点：

```
kubeadm join 192.168.0.200:6443 --token 9vr73a.a8uxyaju799qwdjv --discovery-token-ca-cert-hash
sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720eff5359e26aec866
```

将此输出复制到文本文件。稍后您将需要它来将控制平面节点和辅助节点加入集群。

当 `--upload-certs` 与 `kubeadm init` 一起使用时，主控制平面的证书被加密并上传到 `kubeadm-certs` 密钥中。

要重新上传证书并生成新的解密密钥，请在已加入集群节点的控制平面上使用以下命令：

```
sudo kubeadm init phase upload-certs --upload-certs
```

您还可以在 `init` 期间指定自定义的 `--certificate-key`，以后可以由 `join` 使用。要生成这样的密钥，可以使用以下命令：

```
kubeadm alpha certs certificate-key
```

说明： `kubeadm-certs` 密钥和解密密钥会在两个小时后失效。

注意： 正如命令输出中所述，证书密钥可访问群集敏感数据，并将其保密！

应用您选择的 CNI 插件： [请遵循以下指示](/zh/docs/setup/production-environment/tools/kubeadm/create-cluster-kubeadm/#pod-network) 安装 CNI 提供程序。如果适用，请确保配置与 kubeadm 配置文件中指定的 Pod CIDR 相对应。

在此示例中，我们使用 Weave Net：

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 | tr -d '\n')
```

`style="color:#a2f;font-weight:700">)"</code></pre></div><p>输入以下内容，并查看 pods 的控制平面组件启动：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh">kubectl get pod -n kube-system -w</code></pre></div><h3 id="#其余控制平面节点制平面节点">其余控制平面节点的步骤</h3><blockquote class="note callout"><div>说明：从 kubeadm 1.15 版本开始，您可以并行加入多个控制平面节点。在此版本之前，您必须在第一个节点初始化后才能依序的增加新的控制平面节点。</div></blockquote><p>对于每个其他控制平面节点，您应该：</p><p>执行先前由第一个节点上的 <code>kubeadm init</code> 输出提供给您的 join 命令。它看起来应该像这样：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh">sudo kubeadm join 192.168.0.200:6443 --token 9vr73a.a8uxyaju799qwdjv --discovery-token-ca-cert-hash sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720eff5359e26aec866 --control-plane --certificate-key f8902e114ef118304e561c3ecd4d0b543adc226b7a07f675f56564185ffe0c07</code></pre></div>这个 <code>--control-plane</code> 命令通知 <code>kubeadm join</code> 创建一个新的控制平面。<code>--certificate-key ...</code> 将导致从集群中的 <code>kubeadm-certs</code> 秘钥下载控制平面证书并使用给定的密钥进行解密。<h2 id="#外部-etcd-节点">外部 etcd 节点</h2><p>使用外部 etcd 节点设置集群类似于用于堆叠 etcd 的过程，不同之处在于您应该首先设置 etcd，并在 kubeadm 配置文件中传递 etcd 信息。</p><h3 id="#设置-etcd-集群">设置 etcd 集群</h3><p>按照 这些指示 去设置 etcd 集群。</p><p>设置 SSH 在 这描述。</p><p>将以下文件从集群中的任何 etcd 节点复制到第一个控制平面节点：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh">exportCONTROL_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}"</pre></div>`

`style="color:#b68;font-weight:700">}`

`style="color:#b44">"`

- 用第一台控制平面机的 `user@host` 替换 `CONTROL PLANE` 的值。

设置第一个控制平面节点

用以下内容创建一个名为 `kubeadm-config.yaml` 的文件：

```
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
```

说明： 这里堆 etcd 和外部 etcd 之前的区别在于设置外部 etcd 需要一个 `etcd` 的

`external` 对象下带有 etcd 端点的配置文件。如果是堆 etcd 技术，

是自动管理的。

在您的集群中，将配置模板中的以下变量替换为适当值：

- `LOAD BALANCER DNS`

- `LOAD BALANCER PORT`

- `ETCD 0 IP`

- `ETCD 1 IP`

- `ETCD 2 IP`

以下的步骤与设置堆集群是相似的：

在节点上运行 `sudo kubeadm init`

`--config kubeadm-config.yaml --upload-certs` 命令。

编写输出联接命令，这些命令将返回到文本文件以供以后使用。

应用您选择的 CNI 插件。给定以下示例适用于 Weave Net：

```
kubect
l apply -f "https://cloud.weave.works/k8s/net?k8s-
version=$(
kubect
l version | base64 | tr -d '\n'
)
```

其他控制平面节点的步骤

步骤与设置堆 etcd 相同：

- 确保第一个控制平面节点已完全初始化。

- 使用保存到文本文件的连接命令将每个控制平面节点连接在一起。建议一次加入一个控制平面节点。

- 不要忘记默认情况下，`--certificate-key` 中的解密密钥会在两个小时后过期。

列举控制平面之后的常见任务

安装工作节点

您可以使用之前存储的命令将工作节点加入集群中 作为

`kubeadm init` 命令的输出：

```
sudo kubeadm join
192.168.0.200:6443 --token 9vr73a.a8uxyaju799qwdjv --discovery-token-ca-
cert-hash
sha256:7c2e69131a36ae2a042a339b33381c6d0d43887e2de83720eff5359e26aec866
```

```
4" >
```

```
4" >
```

```
4" >
```

```
4" >
```

```
4" >
```

手动证书分发

如果您选择不将 `kubeadm init` 与 `--upload-certs` 命令

一起使用，则意味着您将必须手动将证书从主控制平面节点复制到将要加入的控制平面节点上。

有许多方法可以实现这种操作。在下面的例子中我们使用 `ssh` 和 `scp`：

如果要在单独的一台计算机控制所有节点，则需要 SSH。

- 在您的主设备上启动 `ssh-agent`，要求该设备能访问系统中的所有其他节点：

```
eval $(ssh-agent)
```

- 将 SSH 身份添加到会话中：

```
ssh-add ~/.ssh/path to private key
```

- 检查节点间的 SSH 以确保连接是正常运行的

- SSH 到任何节点时，请确保添加 `-A` 标志：

```
ssh -A 10.0.0.7
```

- 当在任何节点上使用 `sudo` 时，请确保环境完善，以便使用 SSH 转发任务：

```
sudo -E -s
```

- 在所有节点上配置 SSH 之后，您应该在运行过 `kubeadm init` 命令的第一个控制平面节点上运行以下脚本。该脚本会将证书从第一个控制平面节点复制到另一个控制平面节点：

在以下示例中，用其他控制平面节点的 IP 地址替换 `CONTROL PLANE IPS`。

```
USER=ubuntu # 可自己设置 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@ $CONTROL PLANE IPS: scp /etc/kubernetes/pki/ca.key $USER@ $CONTROL PLANE IPS: scp /etc/kubernetes/pki/sa.key $USER@ $CONTROL PLANE IPS: scp /etc/kubernetes/pki/sa.pub $USER@ $CONTROL PLANE IPS: scp /etc/kubernetes/pki/front-proxy-ca.crt $USER@ $CONTROL PLANE IPS: scp /etc/kubernetes/pki/front-proxy-ca.key $
```

```
{</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">"</span>@<span style="color:#b8860b">$host</span>: scp /etc/kubernetes/pki/etcd/ca.crt <span style="color:#b44">"</span><span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">"</span>@<span style="color:#b8860b">$host</span>:etcd-ca.crt scp /etc/kubernetes/pki/etcd/ca.key <span style="color:#b44">"</span><span><span style="color:#b68;font-weight:700">${</span><span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span><span style="color:#b44">"</span>@<span style="color:#b8860b">$host</span>:etcd-ca.key <span style="color:#a2f;font-weight:700">done</span> </code></pre></div></li></ol><blockquote class="caution callout"><div><strong>注意：</strong> 只需要复制上面列表中的证书。kubeadm 将负责生成其余证书以及加入控制平面实例所需的 SAN。如果您错误地复制了所有证书，由于缺少所需的 SAN，创建其他节点可能会失败。</div></blockquote><ol><li><p>然后，在每个连接控制平面节点上，您必须先运行以下脚本，然后再运行 <code>kubeadm join</code>。该脚本会将先前复制的证书从主目录移动到 <code>/etc/kubernetes/pki</code>：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh"><span style="color:#b8860b">USER</span><span style="color:#666">=</span>ubuntu <span style="color:#080;font-style:italic"># 可自己设置</span> mkdir -p /etc/kubernetes/pki/etcd mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/ca.crt /etc/kubernetes/pki/ mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/ca.key /etc/kubernetes/pki/ mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/sa.pub /etc/kubernetes/pki/ mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/sa.key /etc/kubernetes/pki/ mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/front-proxy-ca.crt /etc/kubernetes/pki/ mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/front-proxy-ca.key /etc/kubernetes/pki/ mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt mv /home/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>/etcd-ca.key /etc/kubernetes/pki/etcd/ca.key </code></pre></div></li></ol><div id="pre-footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您有帮助？</p><button class="btn btn-
```

primary mb-4 feedback--yes">是</button> <button class="btn btn-primary mb-4 feedback--no">否</button><p class="feedback--response feedback--response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 Stack Overflow。在 GitHub 仓库上登记新的问题 报告问题 或者 提出改进建议.</p></div><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><div class="text-muted mt-5 pt-3 border-top">最后修改 September 07, 2020 at 1:35 PM PST: [zh] Fix links in setup section (3fda142df)</div></div><div class="td-content"><h1>使用 kubeadm 创建一个高可用 etcd 集群</h1><blockquote class="note callout"><div>说明：<p>在本指南中，当 kubeadm 用作为外部 etcd 节点管理工具，请注意 kubeadm 不计划支持此类节点的证书更换或升级。对于长期规划是使用 etcdadm 增强工具来管理这方面。</p></div></blockquote><p>默认情况下，kubeadm 运行单成员的 etcd 集群，该集群由控制面节点上的 kubelet 以静态 Pod 的方式进行管理。由于 etcd 集群只包含一个成员且不能在任一成员不可用时保持运行，所以这不是一种高可用设置。本任务，将告诉您如何在使用 kubeadm 创建一个 kubernetes 集群时创建一个外部 etcd：有三个成员的高可用 etcd 集群。</p><h2 id="准备开始">准备开始</h2>三个可以通过 2379 和 2380 端口相互通信的主机。本文档使用这些作为默认端口。不过，它们可以通过 kubeadm 的配置文件进行自定义。每个主机必须 安装有 docker、kubelet 和 kubeadm。一些可以用来在主机间复制文件的基础设施。例如 <code>ssh</code> 和 <code>scp</code> 就可以满足需求。<h2 id="建立集群">建立集群</h2><p>一般来说，是在一个节点上生成所有证书并且只分发这些必要的文件到其它节点上。</p><blockquote class="note callout"><div>说明：<p>kubeadm 包含生成下述证书所需的所有必要的密码学工具；在这个例子中，不需要其他加密工具。</p></div></blockquote><p>将 kubelet

配置为 etcd 的服务管理器。

由于 etcd 是首先创建的，因此您必须通过创建具有更高优先级的新文件来覆盖 kubeadm 提供的 kubelet 单元文件。

```
cat <&lt; 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

为 kubeadm 创建配置文件。

使用以下脚本为每个将要运行 etcd 成员的主机生成一个 kubeadm 配置文件。

```
# 使用 IP 或可解析的主机名替换 HOST0、HOST1 和 HOST2
```

```
export
```

```
HOST0
```

```
=
```

```
10.0.0.6
```

```
export
```

```
HOST1
```

```
=
```

```
10.0.0.7
```

```
export
```

```
HOST2
```

```
=
```

```
10.0.0.8
```

```
# 创建临时目录来存储将被分发到其它主机上的文件
```

```
mkdir -p /tmp/
```

```
{
```

```
HOST0
```

```
}
```

```
/ tmp/
```

```
{
```

```
HOST1
```

```
}
```

```
/ tmp/
```

```
{
```

```
HOST2
```

```
}
```

```
< ETCDHOSTS
```

```
=
```

```
{
```

```
HOST0
```

```
}
```

```
<
```

```
{
```

```
HOST1
```

```
}
```

```
<
```

```
HOST2
```

```
}
```

```
<
```

```
<
```

```
NAMES
```

```
=
```

```
"infra0"
```

```
"infra1"
```

```
"infra2"
```

```
<
```

```
for
```

```
i in
```

```
!ETCDHOSTS[@]
```

```

style="color:#b68;font-weight:700">}</span><span
style="color:#b44">"</span>; <span style="color:#a2f;font-weight:
700">do</span> <span style="color:#b8860b">HOST</span><span
style="color:#666">=</span><span style="color:#b68;font-weight:700">${
</span><span style="color:#b8860b">ETCDHOSTS</span>[<span
style="color:#b8860b">${i}</span>]<span style="color:#b68;font-weight:
700">}</span> <span style="color:#b8860b">NAME</span><span
style="color:#666">=</span><span style="color:#b68;font-weight:700">${
</span><span style="color:#b8860b">NAMES</span>[<span
style="color:#b8860b">${i}</span>]<span style="color:#b68;font-weight:
700">}</span> cat <span style="color:#b44">&lt;&lt; EOF &gt; /tmp/${
{HOST}}/kubeadmcfq.yaml </span><span style="color:#b44">apiVersion:
"kubeadm.k8s.io/v1beta2" </span><span style="color:#b44">kind:
ClusterConfiguration </span><span style="color:#b44">etcd: </
span><span style="color:#b44"> local: </span><span
style="color:#b44"> serverCertSANs: </span><span style="color:#b44"> -
"${HOST}" </span><span style="color:#b44"> peerCertSANs: </
span><span style="color:#b44"> - "${HOST}" </span><span
style="color:#b44"> extraArgs: </span><span style="color:#b44"> initial-
cluster: infra0=https://${ETCDHOSTS[0]}:2380,infra1=https://${
{ETCDHOSTS[1]}:2380,infra2=https://${ETCDHOSTS[2]}:2380 </
span><span style="color:#b44"> initial-cluster-state: new </span><span
style="color:#b44"> name: ${NAME} </span><span style="color:#b44">
listen-peer-urls: https://${HOST}:2380 </span><span style="color:#b44">
listen-client-urls: https://${HOST}:2379 </span><span
style="color:#b44"> advertise-client-urls: https://${HOST}:2379 </
span><span style="color:#b44"> initial-advertise-peer-urls: https://${
{HOST}:2380 </span><span style="color:#b44">EOF</span> <span
style="color:#a2f;font-weight:700">done</span> </code></pre></div></
li><li><p>生成证书颁发机构</p><p>如果您已经拥有 CA，那么唯一的操作是复制
CA 的 <code>crt</code> 和 <code>key</code> 文件到 <code>etc/
kubernetes/pki/etcd/ca.crt</code> 和 <code>etc/kubernetes/pki/etcd/
ca.key</code>。复制完这些文件后继续下一步，"为每个成员创建证书"。</p><p>
如果您还没有 CA，则在 <code>${HOST0}</code>（您为 kubeadm 生成配置文件的
位置）上运行此命令。</p><pre><code>kubeadm init phase certs etcd-ca </
code></pre><p>创建了如下两个文件</p><ul><li><code>/etc/kubernetes/
pki/etcd/ca.crt</code></li><li><code>/etc/kubernetes/pki/etcd/ca.key</
code></li></ul></li><li><p>为每个成员创建证书</p><div
class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-
tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh">kubeadm
init phase certs etcd-server --config<span style="color:#666">=</span>/
tmp/<span style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST2</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfq.yaml kubeadm init phase certs etcd-
peer --config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST2</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfq.yaml kubeadm init phase certs etcd-
healthcheck-client --config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST2</span><span style="color:#b68;font-

```

```
weight:700">}</span>/kubeadmcfgyaml kubeadm init phase certs
apiserver-etcd-client --config<span style="color:#666">=</span>/tmp/
<span style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST2</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml cp -R /etc/kubernetes/pki /tmp/
<span style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST2</span><span style="color:#b68;font-
weight:700">}</span>/ <span style="color:#080;font-style:italic"># 清理不
可重复使用的证书</span> find /etc/kubernetes/pki -not -name ca.crt -not -
name ca.key -type f -delete kubeadm init phase certs etcd-server --
config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST1</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml kubeadm init phase certs etcd-
peer --config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST1</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml kubeadm init phase certs etcd-
healthcheck-client --config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST1</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml kubeadm init phase certs
apiserver-etcd-client --config<span style="color:#666">=</span>/tmp/
<span style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST1</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml cp -R /etc/kubernetes/pki /tmp/
<span style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST1</span><span style="color:#b68;font-
weight:700">}</span>/ find /etc/kubernetes/pki -not -name ca.crt -not -
name ca.key -type f -delete kubeadm init phase certs etcd-server --
config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST0</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml kubeadm init phase certs etcd-
peer --config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST0</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml kubeadm init phase certs etcd-
healthcheck-client --config<span style="color:#666">=</span>/tmp/<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST0</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml kubeadm init phase certs
apiserver-etcd-client --config<span style="color:#666">=</span>/tmp/
<span style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST0</span><span style="color:#b68;font-
weight:700">}</span>/kubeadmcfgyaml <span style="color:#080;font-
style:italic"># 不需要移动 certs 因为它们是给 HOST0 使用的</span> <span
style="color:#080;font-style:italic"># 清理不应从此主机复制的证书</span>
find /tmp/<span style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST2</span><span style="color:#b68;font-
weight:700">}</span> -name ca.key -type f -delete find /tmp/<span
style="color:#b68;font-weight:700">${</span><span
```

```
style="color:#b8860b">HOST1</span><span style="color:#b68;font-weight:700">}</span> -name ca.key -type f -delete </code></pre></div></li><li><p>复制证书和 kubeadm 配置</p><p>证书已生成，现在必须将它们移动到对应的主机。</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh"><span style="color:#b8860b">USER</span><span style="color:#666">=</span><span>ubuntu <span style="color:#b8860b">HOST</span><span style="color:#666">=</span><span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">HOST1</span><span style="color:#b68;font-weight:700">}</span> scp -r /tmp/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">HOST</span><span style="color:#b68;font-weight:700">}</span>/* <span style="color:#b68;font-weight:700">${</span></span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>@<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">HOST</span><span style="color:#b68;font-weight:700">}</span>: ssh <span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">USER</span><span style="color:#b68;font-weight:700">}</span>@<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">HOST</span><span style="color:#b68;font-weight:700">}</span> USER@HOST $ sudo -Es root@HOST $ chown -R root:root pki root@HOST $ mv pki /etc/kubernetes/ </code></pre></div></li><li><p>确保已经所有预期的文件都存在</p><p><code>$HOST0</code> 所需文件的完整列表如下：</p><pre><code>/tmp/${HOST0} └─ kubeadmcfgyaml --- /etc/kubernetes/pki └─ apiserver-etcd-client.crt └─ apiserver-etcd-client.key └─ etcd └─ ca.crt └─ ca.key └─ healthcheck-client.crt └─ healthcheck-client.key └─ peer.crt └─ peer.key └─ server.crt └─ server.key </code></pre><p>在 <code>$HOST1</code>:</p><pre><code>$HOME └─ kubeadmcfgyaml --- /etc/kubernetes/pki └─ apiserver-etcd-client.crt └─ apiserver-etcd-client.key └─ etcd └─ ca.crt └─ healthcheck-client.crt └─ healthcheck-client.key └─ peer.crt └─ peer.key └─ server.crt └─ server.key </code></pre><p>在 <code>$HOST2</code></p><pre><code>$HOME └─ kubeadmcfgyaml --- /etc/kubernetes/pki └─ apiserver-etcd-client.crt └─ apiserver-etcd-client.key └─ etcd └─ ca.crt └─ healthcheck-client.crt └─ healthcheck-client.key └─ peer.crt └─ peer.key └─ server.crt └─ server.key </code></pre></li><li><p>创建静态 Pod 清单</p><p>既然证书和配置已经就绪，是时候去创建清单了。在每台主机上运行 <code>kubeadm</code> 命令来生成 etcd 使用的静态清单。</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-sh" data-lang="sh">root@HOST0 $ kubeadm init phase etcd <span style="color:#a2f">local</span> --config<span style="color:#666">=/tmp/<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">HOST0</span><span style="color:#b68;font-weight:700">}</span>/kubeadmcfgyaml root@HOST1 $ kubeadm init phase etcd <span style="color:#a2f">local</span> --config<span style="color:#666">=/home/ubuntu/kubeadmcfgyaml root@HOST2 $ kubeadm init phase etcd <span>
```

```
style="color:#a2f">local</span> --config<span style="color:#666">=</span>/home/ubuntu/kubeadmconfig.yaml </code></pre></div></li><li><p>
可选：检查群集运行状况</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code
class="language-sh" data-lang="sh">docker run --rm -it <span
style="color:#b62;font-weight:700">\ </span><span
style="color:#b62;font-weight:700"/>--net host <span
style="color:#b62;font-weight:700">\ </span><span
style="color:#b62;font-weight:700"/>-v /etc/kubernetes:/etc/kubernetes
k8s.gcr.io/etcd:<span style="color:#b68;font-weight:700">${</span>
span><span style="color:#b8860b">ETCD_TAG</span><span
style="color:#b68;font-weight:700">}</span> etcdctl <span
style="color:#b62;font-weight:700">\ </span><span
style="color:#b62;font-weight:700"/>--cert /etc/kubernetes/pki/etcd/peer.crt
<span style="color:#b62;font-weight:700">\ </span><span
style="color:#b62;font-weight:700"/>--key /etc/kubernetes/pki/etcd/peer.key
<span style="color:#b62;font-weight:700">\ </span><span
style="color:#b62;font-weight:700"/>--cacert /etc/kubernetes/pki/etcd/ca.crt
<span style="color:#b62;font-weight:700">\ </span><span
style="color:#b62;font-weight:700"/>--endpoints https://<span
style="color:#b68;font-weight:700">${</span><span
style="color:#b8860b">HOST0</span><span style="color:#b68;font-
weight:700">}</span>:2379 endpoint health --cluster ... https://<span
style="color:#666">[</span><span>HOST0 IP<span style="color:#666">]</span>
span>:2379 is healthy: successfully committed proposal: <span
style="color:#b8860b">took</span> <span style="color:#666">=</span>
16.283339ms https://<span style="color:#666">[</span><span>HOST1 IP<span
style="color:#666">]</span>:2379 is healthy: successfully committed
proposal: <span style="color:#b8860b">took</span> <span
style="color:#666">=</span> 19.44402ms https://<span
style="color:#666">[</span><span>HOST2 IP<span style="color:#666">]</span>
span>:2379 is healthy: successfully committed proposal: <span
style="color:#b8860b">took</span> <span style="color:#666">=</span>
35.926451ms </code></pre></div><ul><li>将 <code>${ETCD_TAG}</code>
<code> 设置为你的 etcd 镜像的版本标签，例如 <code>3.4.3-0</code>。要查看
kubeadm 使用的 etcd 镜像和标签，请执行 <code>kubeadm config images list --
kubernetes-version ${K8S_VERSION}</code>，其中 <code>${
{K8S_VERSION}</code> 是 <code>v1.17.0</code> 作为例子。</li></ul>
<ul><li>将 <code>${HOST0}</code> 设置为要测试的主机的 IP 地址</li>
</ul></li></ol><h2 id="&#x63A5;&#x4E0B;&#x6765;">接下来</h2>
<p>一旦拥有了一个正常工作的 3 成员的 etcd 集群，你就可以基于 <a href="/zh/docs/setup/production-environment/tools/kubeadm/high-availability/">使用
kubeadm 的外部 etcd 方法</a>，继续部署一个高可用的控制平面。</p><div
id="pre-footer"><h2>反馈</h2><p class="feedback--prompt">此页是否对您
有帮助？</p><button class="btn btn-primary mb-4 feedback--yes">是</button>
<button class="btn btn-primary mb-4 feedback--no">否</button>
<p class="feedback--response feedback--response hidden">感谢反
馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 <a
target=" blank" rel="noopener" href="https://stackoverflow.com/questions/tagged/kubernetes">Stack Overflow</a>。在 GitHub 仓库上登记新的问题 <a
class="feedback-link" target=" blank" rel="noopener" href="https://
```

github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io> 报告问题 或者 提出改进建议.</p></div><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><div class="text-muted mt-5 pt-3 border-top">最后修改 September 07, 2020 at 1:35 PM PST: [zh] Fix links in setup section (3fda142df)</div></div><div class="td-content"><h1>使用 kubeadm 配置集群中的每个 kubelet</h1><div style="margin-top: 10px;margin-bottom: 10px">FEATURE STATE: <code>Kubernetes 1.11 [stable]</code></div><p>kubeadm CLI 工具的生命周期与 kubelet解耦，它是一个守护程序，在 Kubernetes 集群中的每个节点上运行。当 Kubernetes 初始化或升级时，kubeadm CLI 工具由用户执行，而 kubelet 始终在后台运行。</p><p>由于 kubelet 是守护程序，因此需要通过某种初始化系统或服务管理器进行维护。当使用 DEB 或 RPM 安装 kubelet 时，配置系统去管理 kubelet。你可以改用其他服务管理器，但需要手动地配置。</p><p>集群中涉及的所有 kubelet 的一些配置细节都必须相同，而其他配置方面则需要基于每个 kubelet 进行设置，以适应给定机器的不同特性（例如操作系统、存储和网络）。你可以手动地管理 kubelet 的配置，但是 kubeadm 现在提供一种 <code>KubeletConfiguration</code> API 类型 用于集中管理 kubelet 的配置。</p><h2 id="kubelet-配置模式">Kubelet 配置模式</h2><p>以下各节讲述了通过使用 kubeadm 简化 kubelet 配置模式，而不是在每个节点上手动地管理 kubelet 配置。</p><h3 id="将集群级配置传播 kubelet-中">将集群级配置传播到每个 kubelet 中</h3><p>你可以通过使用 <code>kubeadm init</code> 和 <code>kubeadm join</code> 命令为 kubelet 提供默认值。有趣的示例包括使用其他 CRI 运行时或通过服务器设置不同的默认子网。</p><p>如果你想使用子网 <code>10.96.0.0/12</code> 作为默认的服务，你可以给 kubeadm 传递 <code>--service-cidr</code> 参数：</p><div class="highlight"><pre style="background-color: #f8f8f8;-moz-tab-size: 4;-o-tab-size: 4;tab-size: 4"><code class="language-bash" data-lang="bash">kubeadm init --service-cidr 10.96.0.0/12 </code></pre></div><p>现在，可以从该子网分配服务的虚拟 IP。你还需要通过 kubelet 使用 <code>--cluster-dns</code> 标志设置 DNS 地址。在集群中的每个管理器和节点上的 kubelet 的设置需要相同。kubelet 提供了一个版本化的结构化 API 对象，该对

象可以配置 kubelet 中的大多数参数，并将此配置推送到集群中正在运行的每个

kubelet 上。此对象被称为 **kubelet 的配置组件**。该配置组件

允许用户指定标志，例如用骆峰代表集群的 DNS IP 地址，如下所示：

```
apiVersion: kubelet.config.k8s.io/v1beta1
kind: KubeletConfiguration
clusterDNS: 10.96.0.10
```

有关组件配置的更多详细信息，请参阅 [#configure-kubelets-using-kubeadm](#) 本节。

提供指定实例的详细配置信息

由于硬件、操作系统、网络或者其他主机特定参数的差异。某些主机需要特定的 kubelet 配置。以下列表提供了一些示例。

- 由 kubelet 配置标志 `--resolv-conf` 指定的 DNS 解析文件的路径在操作系统之间可能有所不同，它取决于你是否使用

- `systemd-resolved`。如果此路径错误，则在其 kubelet 配置错误的节点上 DNS 解析也将失败。

- 除非你使用云驱动，否则默认情况下 Node API 对象的 `metadata.name` 会被设置为计算机的主机名。

- 如果你需要指定一个与机器的主机名不同的节点名称，你可以使用 `--hostname-override` 标志覆盖默认值。

- 当前，kubelet 无法自动检测 CRI 运行时使用的 cgroup 驱动程序，但是值 `--cgroup-driver` 必须与 CRI 运行时使用的 cgroup 驱动程序匹配，以确保 kubelet 的健康运行状况。

- 取决于你的集群所使用的 CRI 运行时，你可能需要为 kubelet 指定不同的标志。例如，当使用 Docker 时，你需要指定如 `--network-plugin=cni` 这类标志；但是如果你使用的是外部运行时，则需要指定

- `--container-runtime=remote` 并使用 `--container-runtime-endpoint=` 指定 CRI 端点。

你可以在服务管理器（例如 systemd）中设定某个 kubelet 的配置来指定这些参数。

使用 kubeadm 配置 kubelet

如果自定义的

`KubeletConfiguration` API 对象使用像 `kubeadm ... --config some-config-file.yaml` 这样的配置文件进行传递，则可以配置

kubeadm 启动的 kubelet。

通过调用 `kubeadm config print init-defaults --component-configs KubeletConfiguration`，你可以看到此结构中的所有默认值。

也可以阅读 <https://godoc.org/k8s.io/kubernetes/pkg/kubelet/apis/config#KubeletConfiguration> kubelet 配置组件的 API 参考来获取更多各个字段的更多信息。

当使用

`kubeadm init` 时的工作流程

当调用 `kubeadm init` 时，kubelet 配置被编组到磁盘上的 `/var/lib/kubelet/`

`config.yaml` 中，并且上传到集群中的 ConfigMap。ConfigMap 名为

`kubelet-config-1.X`，其中 `X` 是你正在初始化的 Kubernetes 版本的次版本。在集群中所有 kubelet 的基准集群范围内配置，将

kubelet 配置文件写入 `/etc/kubernetes/kubelet.conf` 中。此配置文件指向允许 kubelet 与 API 服务器通信的客户端证书。这解决了 [将集群级配置传播到每个 kubelet](#propagating-cluster-level-configuration-to-each-kubelet) 的需求。

该文档 [提供特定实例的配置详细信息](#providing-instance-specific-configuration-details) 是第二种解决模式，kubeadm 将环境文件写入 `/var/lib/kubelet/kubeadm-flags.env`，其中包含了一个标志列表，当 kubelet 启动时，该标志列表会传递给 kubelet 标志在文件中的显示方式如下：

```
KUBELET_KUBEADM_ARGS="--flag1=value1 --flag2=value2 ..."
```

除了启动 kubelet 时使用该标志外，该文件还包含动态参数，例如 cgroup 驱动程序以及是否使用其他 CRI 运行时 socket (`--cri-socket`)。

将这两个文件编组到磁盘后，如果使用 systemd，则 kubeadm 尝试运行以下两个命令：

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

如果重新加载和重新启动成功，则正常的 `kubeadm init` 工作流程将继续。

`kubeadm join` 的工作流程

当使用 `kubeadm join` 时，kubeadm 使用 Bootstrap Token 证书执行 TLS 引导，该引导会获取一份证书，该证书需要下载 `kubelet-config-1.X` ConfigMap 并把它写入 `/var/lib/kubelet/config.yaml` 中。动态环境文件的生成方式恰好与 `kubeadm init` 相同。

接下来，kubeadm 运行以下两个命令将新配置加载到 kubelet 中：

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

在 kubelet 加载新配置后，kubeadm 将写入 `/etc/kubernetes/bootstrap-kubelet.conf` KubeConfig 文件中，该文件包含 CA 证书和引导程序令牌。kubelet 使用这些证书执行 TLS 引导程序并获取唯一的凭据，该凭据被存储在 `/etc/kubernetes/kubelet.conf` 中。当此文件被写入后，kubelet 就完成了执行 TLS 引导程序。

kubelet 的 systemd 文件

`kubeadm` 中附带了有关系统如何运行 kubelet 的 systemd 配置文件。请注意 kubeadm CLI 命令不会修改此文件。

通过 `kubeadm` [DEB](https://github.com/kubernetes/release/blob/master/cmd/kubepkg/templates/latest/deb/kubeadm/10-kubeadm.conf) 或者 [RPM](https://github.com/kubernetes/release/blob/master/cmd/kubepkg/templates/latest/rpm/kubeadm/10-kubeadm.conf) 包安装的配置文件被写入 `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` 并由系统使用。它对原来的 [RPM 版本](https://github.com/kubernetes/release/blob/master/cmd/kubepkg/templates/latest/rpm/kubelet/kubelet.service) `kubelet.service` 或者

[kubernetes/release/blob/master/cmd/kubepkg/templates/latest/deb/kubelet/lib/systemd/system/kubelet.service](https://kubernetes.io/release/blob/master/cmd/kubepkg/templates/latest/deb/kubelet/lib/systemd/system/kubelet.service)>DEB 版本 <code>kubelet.service</code> 作了增强：

```
[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" # 这是 "kubeadm init" 和 "kubeadm join" 运行时生成的文件，动态地填充 KUBELET KUBEADM ARGS 变量
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env # 这是一个文件，用户在不得已下可以将其用作替代 kubelet args。 # 用户最好使用 .NodeRegistration.KubeletExtraArgs 对象在配置文件中替代。 #
```

二进制文件和软件&#x

`cri-tools` 从 github.com/kubernetes-sigs/cri-tools 仓库中安装 `/usr/bin/crictl` 可执行文件。

反馈

此页是否对您有帮助？

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 [Stack Overflow](https://stackoverflow.com/questions/tagged/kubernetes)。在 GitHub 仓库上登记新的问题 <https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io>

报告问题

或者 [提出改进建议](https://github.com/kubernetes/website/issues/new?title=Improvement%20for%20k8s.io).

，并等待生成的 Pod 运行。

- 自托管 Pod 运行后，将删除其关联的静态 Pod，然后 kubeadm 继续安装下一个组件。这将触发 kubelet 停止那些静态 Pod。

- 当原始静态控制平面停止时，新的自托管控制平面能够绑定到侦听端口并变为活动状态。

反馈

此页是否对您有帮助？

是否

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 [Stack Overflow](https://stackoverflow.com/questions/tagged/kubernetes)。在 GitHub 仓库上登记新的问题 [报告问题](https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io) 或者 [提出改进建议](https://github.com/kubernetes/website/issues/new?title=Improvement%20for%20k8s.io)。

id="#x521B;建集群">创建集群</h2><h3 id="1-5-安装-kops">(1/5) 安装 kops</h3><h4

id="#x524D;提条件">前提条件</h4><p>你必须安装 kubectl 才能使 kops 工作。</p><h4 id="#x5B89;装">安装</h4><p>从下载页面下载 kops (从源代码构建也很容易) :</p><p>在 macOS 上 :</p><div class="highlight"><pre

```


route53-&#x57DF;&#x540D;">(2/5) 为你的集群创建一个 route53 域名</h3><p>kops 在集群内部都使用 DNS 进行发现操作,因此你可以从客户端访问


```

kubernetes API 服务器。</p><p>kops 对集群名称有明显的要求:它应该是有效的 DNS 名称。这样一来,你就不会再使集群混乱,可以与同事明确共享集群,并且无需依赖记住 IP 地址即可访问群集。</p><p>你应该使用子域名来划分集群。作为示例,我们将使用域名 <code>useast1.dev.example.com</code>。然后,API 服务器端点

域名将为 <code>api.useast1.dev.example.com</code>。</p><p>Route53 托管区域可以服务子域名。你的托管区域可能是 <code>useast1.dev.example.com</code>, 还有 <code>dev.example.com</code> 甚至 <code>example.com</code>。kops 可以与以上任何一种配合使用,因此通常你出于组织原因选择不同的托管区域。例如,允许你在 <code>dev.example.com</code> 下创建记录,但不能在

<code>example.com</code> 下创建记录。</p><p>假设你使用 <code>dev.example.com</code> 作为托管区域。你可以使用 正常流程 或者使用诸如 <code>aws

<code>route53 create-hosted-zone --name dev.example.com --caller-reference 1</code> 之类的命令来创建该托管区域。</p><p>然后,你必须在父域名中设置你的

DNS 记录,以便该域名中的记录可以被解析。在这里,你将在

<code>example.com</code> 中为 <code>dev</code> 创建 DNS 记录。如果它是根域名,则可以在域名注册机构配置 DNS 记录。例如,你需要在购买

<code>example.com</code> 的地方配置 <code>example.com</code>。</p><p>这一步很容易搞砸(这是问题的第一大原因!)如果你安装了 dig 工具,则可以

通过运行以下步骤再次检查集群是否配置正确:</p><div

```
DNS dev.example.com </code></pre></div><p>你应该看到 Route53 分配了你的托管区域的 4 条 DNS 记录。</p><h3 id="3-5-&#x521B;&#x5EFA;&#x4E00;&#x4E2A;-s3-
```

存储桶来存储集群群">创建一个 S3 存储桶来存储集群状态</h3><p>kops 使你即使在安装后也可以管理集

群。为此，它必须跟踪已创建的集群及其配置、所使用的密钥等。此信息存储在 S3 存储桶中。S3 权限用于控制对存储桶的访问。

多个集群可以使用同一 S3 存储桶，并且你可以在管理同一集群的同事之间共享一个 S3 存储桶 - 这比传递 `kubecfg` 文件容易得多。但是有权访问 S3 存储桶的任何人都将拥有对所有集群的管理访问权限，因此你不想在运营团队之外共享它。

因此，通常每个运维团队都有一个 S3 存储桶（而且名称通常对应于上面托管区域的名称！）

在我们的示例中，我们选择 `dev.example.com` 作为托管区域，因此让我们选择 `clusters.dev.example.com` 作为 S3 存储桶名称。

- 导出 `AWS PROFILE` 文件（如果你需要选择一个配置文件用来使 AWS CLI 正常工作）
- 使用 `aws s3 mb s3://clusters.dev.example.com` 创建 S3 存储桶
- 你可以进行 `export KOPS STATE STORE=s3://clusters.dev.example.com` 操作，然后 `kops` 将默认使用此位置。我们建议将其放入你的 `bash profile` 文件或类似文件中。

4-5- 建立你的集群配置

运行 `"kops create cluster"` 以创建你的集群配置：

```
kops create cluster --zones=us-east-1c
useast1.dev.example.com
```

`kops` 将为你的集群创建配置。请注意，它仅创建配置，实际上并没有创建云资源 - 你将在下一步中使用 `kops update cluster` 进行配置。这使你有机会查看配置或进行更改。

它打印出可用于进一步探索的命令：

- 使用以下命令列出集群：`kops get cluster`
- 使用以下命令编辑该集群：`kops edit cluster useast1.dev.example.com`
- 使用以下命令编辑你的节点实例组：`kops edit ig --name = useast1.dev.example.com nodes`
- 使用以下命令编辑你的主实例组：`kops edit ig --name = useast1.dev.example.com master-us-east-1c`

如果你这是你第一次使用 `kops`，请花几分钟尝试一下！实例组是一组实例，将被注册为 `kubernetes` 节点。在 AWS 上，这是通过 `auto-scaling-groups` 实现的。你可以有多个实例组。例如，如果你想要的是混合实例和按需实例的节点，或者 GPU 和非 GPU 实例。

5-5- 在 AWS 中创建集群

运行 `"kops update cluster"` 以在 AWS 中创建集群：

```
kops update cluster useast1.dev.example.com --yes
```

这需要几秒钟的时间才能运行，但实际上集群可能需要几分钟才能准备就绪。每当更改集群配置时，都会使用 `kops update cluster` 工具。它将对配置进行的更改应用于你的集群 - 根据需要重新配置 AWS 或者 `kubernetes`。

例如，在你运行 `kops edit ig nodes` 之后，然后运行 `kops update cluster --yes` 应用你的配置，有时你还必须运行 `kops rolling-update cluster` 立即回滚更新配置。

如果没有 `--yes` 参数，`kops update cluster` 操作将向你显示其操作的预览效果。这对于生产集群很方便！

探索其他附加组件

请参阅[附加组件列表](/zh/docs/concepts/cluster-administration/addons/)探索其他附加组件，包括用于 `Kubernetes` 集群的日志记录、监视、网络策略、可视化和控制的工具。

清理

- 删除集群：`kops delete cluster useast1.dev.example.com --yes`

反馈

- Slack 频道: <https://>

kubernetes.slack.com/messages/kops-users/>#kops-usersGitHub Issues<h2 id="#&x63A5;&x4E0B;&x6765;">接下来</h2>了解有关 Kubernetes 的概念 和 <code>kubectl</code> 的更多信息。了解 <code>kops</code> 高级用法。请参阅 <code>kops</code> 文档 获取教程、最佳做法和高级配置选项。<div id="pre-footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您有帮助？</p><button class="btn btn-primary mb-4 feedback-yes">是</button> <button class="btn btn-primary mb-4 feedback-no">否</button><p class="feedback-response feedback-response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 Stack Overflow。在 GitHub 仓库上登记新的问题 报告问题 或者 提出改进建议。</p></div><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><div class="text-muted mt-5 pt-3 border-top">最后修改 October 30, 2020 at 4:49 PM PST: update broken link (f609a12f8)</div></div><div class="td-content"><h1>使用 Kubespray 安装 Kubernetes</h1><p>此快速入门有助于使用 Kubespray 安装在 GCE、Azure、OpenStack、AWS、vSphere、Packet (裸机)、Oracle Cloud Infrastructure (实验性) 或 Baremetal 上托管的 Kubernetes 集群。</p><p>Kubespray 是一个由 Ansible playbooks、清单 (inventory)、供应工具和通用 OS/ Kubernetes 集群配置管理任务的领域知识组成的。Kubespray 提供：</p>高可用性集群可组合属性支持大多数流行的 Linux 发行版Ubuntu 16.04、18.04、20.04CentOS / RHEL / Oracle Linux 7、8Debian Buster, Jessie, Stretch, Wheezy</div>

li>Fedora 31、32Fedora CoreOSopenSUSE Leap 15Kinvolk 的 Flatcar Container Linux持续集成测试<p>要选择最适合你的用例的工具，请阅读此比较以 kubeadm 和 kops 。</p><h2

id="#x521B;建集群">创建集群</h2><h3 id="1-5-满足下层设施要求">

满足下层设施要求</h3><p>按以下要求来配置服务器：</p>在将运行 Ansible 命令的计算机上安装 Ansible v2.9 和 python-netaddr运行 Ansible Playbook 需要 Jinja 2.11（或更高版本）目标服务器必须有权访问 Internet 才能拉取 Docker 镜像。否则，需要其他配置（请参见离线环境）目标服务器配置为允许 IPv4 转发你的 SSH 密钥必须复制到清单中的所有服务器部分防火墙不受管理，你将需要按照以前的方式实施自己的规则。为了避免在部署过程中出现任何问题，你应该禁用防火墙如果从非 root 用户帐户运行 kubespray，则应在目标服务器中配置正确的特权升级方法。然后应指定"ansible become" 标志或命令参数 "--become" 或 "-b"<p>Kubespray 提供以下实用程序来帮助你设置环境：</p>为以下云驱动提供的 Terraform 脚本：AWSOpenStackPacket<h3 id="2-5-

编写清单文件">（2/5）编写清单文件</h3><p>设置服务器后，请创建一个 Ansible 的清单文件。你可以手动执行此操作，也可以通过动态清单脚本执行此操作。有关更多信息，请参阅"建立你自己的清单"。</p><h3 id="3-5-

规划集群部署">（3/5）规划集群部署</h3><p>Kubespray 能够自定义部署的许多方面：</p>选择部署模式：kubeadm 或非 kubeadmCNI（网络）插件DNS 配置控制平面的选择：本机/可执行文件或容器化组件版本Calico 路由反射器组件运行时选项Dockercontainerd

Kubernetes

的轻量级容器运行&#x

data-toggle="tooltip" data-placement="top" href="https://cri-o.io/#what-is-cri-o" target=" blank" aria-label="CRI-O">CRI-O

证书生成方式<p>可以修改<a href="https://docs.ansible.com/

ansible/playbooks variables.html">变量文件以进行 Kubespray 定制。如果你刚刚开始使用 Kubespray，请考虑使用 Kubespray 默认设置来部署你的集群并探索 Kubernetes。</p><h3 id="4-5-

部署集群">(4/5) 部署集群</h3><p>接下

来，部署你的集群：</p><p>使用 <a href="https://github.com/kubernetes-

sigs/kubespray/blob/master/docs/getting-started.md#starting-custom-

deployment">ansible-playbook 进行集群部署。</p><div

class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-

tab-size:4;tab-size:4"><code class="language-shell" data-

lang="shell">ansible-playbook -i your/inventory/inventory.ini cluster.yml -b -

v \ <span

style="color:#b62;font-weight:700"/> --private-key<span

style="color:#666">=~/.ssh/private key </code></pre></div><p>大型部署 (超过 100 个节点) 可能需要<a href="https://github.com/

kubernetes-sigs/kubespray/blob/master/docs/large-deployments.md">特定的

调整，以获得最佳效果。</p><h3 id="5-5-

验证部署">(5/5) 验证部署</h3><p>Kubespray 提供了一种使用 <a href="https://github.com/kubernetes-

sigs/kubespray/blob/master/docs/netcheck.md">Netchecker 验证 Pod 间

连接和 DNS 解析的方法。Netchecker 确保 netchecker-agents pod 可以解析。

DNS 请求并在默认名称空间内对每个请求执行 ping 操作。这些 Pods 模仿其余工作负

载的类似行为，并用作集群运行状况指示器。</p><h2

id="集群操作">集群操作</h2><p>Kubespray 提供了其他 Playbooks 来管理集群：scale 和

upgrade。</p><h3

id="扩展集群">扩展集群</h3><p>你可以通过

运行 scale playbook 向集群中添加工作节点。有关更多信息，请参见 "<a

href="https://github.com/kubernetes-sigs/kubespray/blob/master/docs/

getting-started.md#adding-nodes">添加节点"。你可以通过运行 remove-

node playbook 来从集群中删除工作节点。有关更多信息，请参见 "<a

href="https://github.com/kubernetes-sigs/kubespray/blob/master/docs/

getting-started.md#remove-nodes">删除节点"。</p><h3

id="升级集群">升级集群</h3><p>你可以通过

运行 upgrade-cluster Playbook 来升级集群。有关更多信息，请参见 "<a

href="https://github.com/kubernetes-sigs/kubespray/blob/master/docs/

upgrades.md">升级"。</p><h2 id="清理">清理</h2><p>你可以通过 <a href="https://github.com/kubernetes-sigs/kubespray/

blob/master/reset.yml">reset Playbook 重置节点并清除所有与 Kubespray

一起安装的组件。</p><blockquote class="caution callout"><div>注

意： 运行 reset playbook 时，请确保不要意外地将生产集群作为目标！</div></blockquote><h2 id="反馈">反馈</h2><p>你可以通过 <a href="https://github.com/kubernetes-sigs/kubespray/

blob/master/reset.yml">reset Playbook 重置节点并清除所有与 Kubespray

一起安装的组件。</p><blockquote class="caution callout"><div>注

意： 运行 reset playbook 时，请确保不要意外地将生产集群作为目标！</div></blockquote><h2 id="反馈">反馈</h2><p>你可以通过 <a href="https://github.com/kubernetes-sigs/kubespray/

blob/master/reset.yml">reset Playbook 重置节点并清除所有与 Kubespray

一起安装的组件。</p><blockquote class="caution callout"><div>注

意： 运行 reset playbook 时，请确保不要意外地将生产集群作为目标！</div></blockquote><h2 id="反馈">反馈</h2><p>你可以通过 <a href="https://github.com/kubernetes-sigs/kubespray/

blob/master/reset.yml">reset Playbook 重置节点并清除所有与 Kubespray

一起安装的组件。</p><blockquote class="caution callout"><div>注

意： 运行 reset playbook 时，请确保不要意外地将生产集群作为目标！</div></blockquote><h2 id="反馈">反馈</h2><p>你可以通过 <a href="https://github.com/kubernetes-sigs/kubespray/

blob/master/reset.yml">reset Playbook 重置节点并清除所有与 Kubespray

一起安装的组件。</p><blockquote class="caution callout"><div>注

意： 运行 reset playbook 时，请确保不要意外地将生产集群作为目标！</div></blockquote><h2 id="反馈">反馈</h2><p>你可以通过 <a href="https://github.com/kubernetes-sigs/kubespray/

blob/master/reset.yml">reset Playbook 重置节点并清除所有与 Kubespray

一起安装的组件。</p><blockquote class="caution callout"><div>注

意： 运行 reset playbook 时，请确保不要意外地将生产集群作为目标！</div></blockquote><h2 id="反馈">反馈</h2><p>你可以通过 <a href="https://github.com/kubernetes-sigs/kubespray/

id="#x63A5;下来">接下来</h2><p>查看有关 Kubespray 的 路线图的计划工作。</p><div id="pre-footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您有帮助？</p><button class="btn btn-primary mb-4 feedback-yes">是</button> <button class="btn btn-primary mb-4 feedback-no">否</button><p class="feedback-response feedback-response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 Stack Overflow。在 GitHub 仓库上登记新的问题 报告问题 或者 提出改进建议。</p></div><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><div class="text-muted mt-5 pt-3 border-top">最后修改 December 29, 2020 at 5:24 PM PST: typo fix (f54fdec6)</div></div><div class="td-content"><h1>Windows Kubernetes</h1><div class="section-index"><hr class="panel-line"/><div class="entry"><h5>Kubernetes 对 Windows 的支持</h5><p></p></div><div class="entry"><h5>Kubernetes 中调度 Windows 容器的指南</h5><p></p></div></div></div><div class="td-content"><h1>Kubernetes 对 Windows 的支持</h1><p>在很多组织中，其服务和应用的很大比例是 Windows 应用。Windows 容器提供了一种对进程和包依赖关系 进行封装的现代方式，这使得用户更容易采用 DevOps 实践，令 Windows 应用同样遵从 云原生模式。Kubernetes 已经成为事实上的标准容器编排器，Kubernetes 1.14 发行版本中包含了将 Windows 容器调度到 Kubernetes 集群中 Windows 节点上的生产级支持，从而使得巨大的 Windows 应用生态圈能够充分利用 Kubernetes 的能力。对于同时投入基于 Windows 应用和 Linux 应用的组织而言，他们不必寻找不同的编排系统 来管理其工作负载，其跨部署的运维效率得以大幅提升，而不必关心所用操作系统。</p><h2 id="windows-containers-in-kubernetes">kubernetes 中的 Windows 容器</h2><p>若要在 Kubernetes 中

启用对 Windows 容器的编排，只需在现有的 Linux 集群中包含 Windows 节点。在 Kubernetes 上调度 [Pods](/docs/concepts/workloads/pods/pod-overview/ "Pod") 中的 Windows 容器与调用基于 Linux 的容器一样简单、一样容易。

为了运行 Windows 容器，你的 Kubernetes 集群必须包含多个操作系统，控制面节点运行 Linux，工作节点则可以根据负载需要运行 Windows 或 Linux。Windows Server 2019 是唯一被支持的 Windows 操作系统，在 Windows 上启用 [Kubernetes 节点](https://github.com/kubernetes/community/blob/master/contributors/design-proposals/architecture/architecture.md#the-kubernetes-node) 支持（包括 kubelet，[容器运行时](https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/containerd)，以及 kube-proxy）。关于 Windows 发行版渠道的详细讨论，可参见 [Microsoft 文档](https://docs.microsoft.com/en-us/windows-server/get-started-19/servicing-channels-19)。

说明： Kubernetes 控制面，包括[主控组件](https://docs.microsoft.com/en-us/windows-server/get-started-19/servicing-channels-19)，继续在 Linux 上运行。目前没有支持完全是 Windows 节点的 Kubernetes 集群的计划。

说明： 在本文中，当我们讨论 Windows 容器时，我们所指的是具有进程隔离能力的 Windows 容器。具有 [Hyper-V 隔离能力](https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/hyperv-container) 的 Windows 容器计划在将来发行版本中推出。

支持的功能与局限性

支持的功能

Windows 操作系统版本支持

参考下面的表格，了解 Kubernetes 中支持的 Windows 操作系统。同一个异构的 Kubernetes 集群中可以同时包含 Windows 和 Linux 工作节点。Windows 容器仅能调度到 Windows 节点，Linux 容器则只能调度到 Linux 节点。

Kubernetes 版本	Windows Server LTSC 版本	Windows Server SAC 版本
Kubernetes v1.14	Windows Server 2019	Windows Server ver 1809
Kubernetes v1.15	Windows Server 2019	Windows Server ver 1809
Kubernetes v1.16	Windows Server 2019	Windows Server ver 1809
Kubernetes v1.17	Windows Server 2019	Windows Server ver 1809
Kubernetes v1.18	Windows Server 2019	Windows Server ver 1809, Windows Server ver 1903, Windows Server ver 1909
Kubernetes v1.19	Windows Server 2019	Windows Server ver 1909, Windows Server ver 2004

关于不同的 Windows Server 版本的服务渠道，包括其支持模式等相关信息可以在 [Windows Server servicing channels](https://docs.microsoft.com/en-us/windows-server/get-started-19/servicing-channels-19) 找到。

我们并不指望所有 Windows 客户都为其应用频繁地更新操作系统。对应用的更新是向集群中引入新代码的根本原因。对于想要更新运行于 Kubernetes 之上的容器

中操作系统的客户，我们会在添加对新操作系统版本的支持时提供指南和分步的操作指令。该指南会包含与集群节点一起来升级用户应用的建议升级步骤。Windows 节点遵从 Kubernetes [版本偏差策略](/zh/docs/setup/release/version-skew-policy/)（节点到控制面的版本控制），与 Linux 节点的现行策略相同。

Windows Server 主机操作系统会受 [Windows Server](https://www.microsoft.com/en-us/cloud-platform/windows-server-pricing) 授权策略控制。Windows 容器镜像则遵从 [Windows 容器的补充授权条款](https://docs.microsoft.com/en-us/virtualization/windowscontainers/images-eula) 约定。

带进程隔离的 Windows 容器受一些严格的兼容性规则约束，[其中宿主 OS 版本必须与容器基准镜像的 OS 版本相同](https://docs.microsoft.com/en-us/virtualization/windowscontainers/deploy-containers/version-compatibility)。一旦我们在 Kubernetes 中支持带 Hyper-V 隔离的 Windows 容器，这一约束和兼容性规则也会发生改变。

计算

从 API 和 kubectl 的角度，Windows 容器的表现在很大程度上与基于 Linux 的容器是相同的。不过也有一些与关键功能相关的差别值得注意，这些差别列举于 [局限性](#limitations) 小节中。

关键性的 Kubernetes 元素在 Windows 下与其在 Linux 下工作方式相同。我们在本节中讨论一些关键性的负载支撑组件及其在 Windows 中的映射。

[Pods](/zh/docs/concepts/workloads/pods/)

Pod 是 Kubernetes 中最基本的构造模块，是 Kubernetes 对象模型中你可以创建或部署的最小、最简单单元。你不可在同一 Pod 中部署 Windows 和 Linux 容器。Pod 中的所有容器都会被调度到同一节点（Node），而每个节点代表的是一种特定的平台和体系结构。Windows 容器支持 Pod 的以下能力、属性和事件：

- 在带进程隔离和卷共享支持的 Pod 中运行一个或多个容器
- Pod 状态字段
- 就绪态（Readiness）和活跃性（Liveness）探针
- postStart 和 preStop 容器生命周期事件
- ConfigMap、Secrets：用作环境变量或卷
- emptyDir 卷
- 从宿主系统挂载命名管道
- 资源限制

[控制器（Controllers）](/zh/docs/concepts/workloads/controllers/)

Kubernetes 控制器处理 Pod 的期望状态。Windows 容器支持以下负载控制器：

- ReplicaSet
- ReplicationController
- Deployment
- StatefulSet
- DaemonSet
- Job
- CronJob

[服务（Services）](/zh/docs/concepts/services-networking/service/)

Kubernetes Service 是一种抽象对象，用来定义 Pod 的一个逻辑集合及用来访问这些 Pod 的策略。Service 有时也称作微服务（Micro-service）。你可以使用服务来实现跨操作系统的连接。在 Windows 系统中，服务可以使用下面的类型、属性和能力：

- Service 环境变量
- NodePort
- ClusterIP
- LoadBalancer
- ExternalName
- 无头（Headless）服务

Pods、控制器和服务是在 Kubernetes 上管理 Windows 负载的关键元素。不过，在一个动态的云原生环境中，这些元素本身还不足以用来正确管理 Windows 负载的生命周期。我们为此添加了如下功能特性：

- Pod 和容器的度量（Metrics）
- 对水平 Pod 自动扩展的支持
- 对 kubectl exec 命令的支持
- 资源配额
- 调度器抢占

容器运行时

Docker EE

FEATURE STATE: `Kubernetes v1.14 [stable]`

Docker EE-basic 19.03+ 是建议所有 Windows Server 版本采用的容器运行时。该容器运行时能够与 kubelet 中的 dockershim 代码协同工作。

CRI-ContainerD

10px;margin-bottom:10px">FEATURE STATE: <code>Kubernetes v1.19 [beta]</code></div><blockquote class="caution

callout"><div>注意： 在 ContainerD 上使用 GMSA 访问 Windows 网络共享资源时，有一个 已知的局限，需要内核补丁来解决。你可以在关注 Microsoft Windows Containers 问题跟踪 来跟进相关的更新。</div></blockquote><p><a class="glossary-tooltip"

title="#x5F3A;调简单性、健壡"

data-toggle="tooltip" data-placement="top" href="https://containerd.io/docs/" target=" blank" aria-label="ContainerD">ContainerD 1.4.0-

beta.2+ 也可在 Windows Kubernetes 节点上用作容器运行时。</p><p>在

Windows 对 ContainerD 的最初支持是在 Kubernetes v1.18 加入的。Windows 上

ContainerD 的进展可以在 <a href="https://github.com/kubernetes/

enhancements/issues/1001">enhancements#1001 跟进。</p><p>你可

以进一步了解如何<a href="/zh/docs/setup/production-environment/container-

runtimes/#install-containerd">在 Windows 上安装 ContainerD。</p><h4

id="persistent-storage">持久性存储</h4><p>使用 Kubernetes <a href="/zh/

docs/concepts/storage/volumes/">卷，对数据持久性和 Pod 卷 共享有需求的

复杂应用也可以部署到 Kubernetes 上。管理与特定存储后端或协议相关的持久卷时，

相关的操作包括：对卷的配备（Provisioning）、去配（De-provisioning）和调整大小，

将卷挂接到 Kubernetes 节点或从节点上解除挂接，将卷挂载到需要持久数据的

Pod 中的某容器或从容器上卸载。负责实现为特定存储后端或协议实现卷管理动作的代

码以 Kubernetes 卷 <a href="/zh/docs/concepts/storage/volumes/#types-of-

volumes">插件的形式发布。Windows 支持以下大类的 Kubernetes 卷插件：

</p><h5 id="in-tree-volume-plugins">树内卷插件</h5><p>与树内卷插件（In-

Tree Volume Plugin）相关的代码都作为核心 Kubernetes 代码基的一部分发布。树

内卷插件的部署不需要安装额外的脚本，也不需要额外部署独立的 容器化插件组件。这

些插件可以处理：对应存储后端上存储卷的配备、去配和尺寸更改，将卷挂接到

Kubernetes 或从其上解挂，以及将卷挂载到 Pod 中各个容器上或从其上 卸载。以下树

内插件支持 Windows 节点：</p><a href="/zh/docs/concepts/

storage/volumes/#awselasticblockstore">awsElasticBlockStore</

li><a href="/zh/docs/concepts/storage/volumes/

#azuredisk">azureDisk<a href="/zh/docs/concepts/storage/

volumes/#azurefile">azureFile<a href="/zh/docs/concepts/

storage/volumes/#gcepersistentdisk">gcePersistentDisk<a

href="/zh/docs/concepts/storage/volumes/

#vspherevolume">vsphereVolume<h5 id="flexvolume-

plugins">FlexVolume 插件</h5><p>与 <a href="/docs/concepts/storage/

volumes/#flexVolume">FlexVolume 插件相关的代码是作为 树外（Out-of-

tree）脚本或可执行文件来发布的，因此需要在宿主系统上直接部署。FlexVolume 插

件处理将卷挂接到 Kubernetes 节点或从其上解挂、将卷挂载到 Pod 中 各个容器上或

从其上卸载等操作。对于与 FlexVolume 插件相关联的持久卷的配备和 去配操作，可以

通过外部的配置程序来处理。这类配置程序通常与 FlexVolume 插件 相分离。下面的

FlexVolume <a href="https://github.com/Microsoft/K8s-Storage-Plugins/

tree/master/flexvolume/windows">插件 可以以 PowerShell 脚本的形式部署

到宿主系统上，支持 Windows 节点：</p><a href="https://

github.com/microsoft/K8s-Storage-Plugins/tree/master/flexvolume/windows/

plugins/microsoft.com~smb.cmd">SMB<a href="https://

github.com/microsoft/K8s-Storage-Plugins/tree/master/flexvolume/windows/

plugins/microsoft.com~iscsi.cmd">iSCSI<h5 id="csi-plugins">CSI 插件</h5><div style="margin-top:10px;margin-bottom:10px">FEATURE STATE: <code>Kubernetes v1.16 [alpha]</code></div><p>与 CSI 插件相关联的代码作为 树外脚本和可执行文件来发布且通常发布为容器镜像形式，并使用 DaemonSet 和 StatefulSet 这类标准的 Kubernetes 构造体来部署。CSI 插件处理 Kubernetes 中的很多卷管理操作：对卷的配备、去配和调整大小，将卷挂接到 Kubernetes 节点或从节点上解除挂接，将卷挂载到需要持久数据的 Pod 中的某容器或从容器上卸载，使用快照和克隆来备份或恢复持久数据。CSI 插件通常包含节点插件（以 DaemonSet 形式运行于各节点上）和控制器插件。</p><p>CSI 节点插件（尤其是那些通过块设备或者共享文件系统形式来提供持久卷的插件）需要执行很多特权级操作，例如扫描磁盘设备、挂载文件系统等等。这些操作在不同的宿主操作系统上差别较大。对于 Linux 工作节点而言，容器化的 CSI 节点插件通常部署为特权级的容器。对于 Windows 工作节点而言，容器化的 CSI 节点插件的特权操作通过 csi-proxy 来支持；csi-proxy 是一个社区管理的、独立的可执行文件，需要预安装在每个 Windows 节点之上。请参考你要部署的 CSI 插件的部署指南以进一步了解其细节。</p><h4 id="networking">联网</h4><p>Windows 容器的联网是通过 CNI 插件 来暴露出来的。Windows 容器的联网行为与虚拟机的联网行为类似。每个容器有一块虚拟的网络适配器（vNIC）连接到 Hyper-V 的虚拟交换机（vSwitch）。宿主的联网服务（Host Networking Service，HNS）和宿主计算服务（Host Compute Service，HCS）协同工作，创建容器并将容器的虚拟网卡连接到网络上。HCS 负责管理容器，HNS 则负责管理网络资源，例如：

- 虚拟网络（包括创建 vSwitch）
- 端点（Endpoint）/ vNIC
- 名字空间（Namespace）
- 策略（报文封装、负载均衡规则、访问控制列表、网络地址转译规则等等）

支持的服务规约类型如下：

- NodePort
- ClusterIP
- LoadBalancer
- ExternalName

| 网络驱动 | 容器报文修改 | 网络插件 | 网络插件特点 | |----------|--------|---|--| | L2bridge | | | 容器挂接到外部 vSwitch 上。容器挂接到下层网络之上，但由于容器的 MAC 地址在入站和出站时被重写，物理网络不需要这些地址。 | | | | | MAC 地址被重写为宿主系统的 MAC 地址，IP 地址也可能依据 HNS OutboundNAT 策略重写为宿主的 IP 地址。 | | | | win-bridge、Azure-CNI；Flannel 宿主网关（host-gateway）使用 win-bridge | win-bridge 使用二层桥接（L2bridge）网络模式，将容器连接到下层宿主系统上，从而提供最佳性能。需要用户定义的路由（User-Defined Routes，UDR）才能实现节点间的连接。 | | L2Tunnel | | | 这是二 |

层桥接的一种特殊情形，但仅被用于 Azure 上。所有报文都被发送到虚拟化环境中的宿主主机上并根据 SDN 策略进行处理。

MAC 地址被改写，IP 地址在下层网络上可见。	Azure-CNI	Azure-CNI 使得容器能够与 Azure vNET 集成，并允许容器利用 Azure 虚拟网络 所提供的功能特性集合。例如，可以安全地连接到 Azure 服务上或者使用 Azure NSG。你可以参考 azure-cni 所提供的一些示例。
覆盖网络 (Kubernetes 中为 Windows 提供的覆盖网络支持处于 alpha 阶段)	每个容器会获得一个连接到外部 vSwitch 的虚拟网卡 (vNIC)。每个覆盖网络都有自己的、通过定制 IP 前缀来定义的 IP 子网。覆盖网络驱动使用 VXLAN 封装。封装于外层包头内。	Win-overlay 、Flannel VXLAN (使用 win-overlay)
当 (比如出于安全原因) 期望虚拟容器网络与下层宿主网络隔离时，应该使用 win-overlay。如果你的数据中心可用 IP 地址受限，覆盖网络允许你在不同的网络中复用 IP 地址 (每个覆盖网络有不同的 VNID 标签)。这一选项要求在 Windows Server 2009 上安装 KB4489899 补丁。	透明网络 (ovn-kubernetes 的特殊用例)	需要一个外部 vSwitch。容器挂接到某外部 vSwitch 上，该 vSwitch 通过逻辑网络 (逻辑交换机和路由器) 允许 Pod 间通信。
报文或者通过 GENEVE 来封装，或者通过 STT 隧道来封装，以便能够到达不在同一宿主系统上的每个 Pod。报文通过 OVN 网络控制器所提供的隧道元数据信息来判定是转发还是丢弃。	北-南向通信通过 NAT 网络地址转译来实现。	ovn-kubernetes
通过 Ansible 来部署 。	所发布的 ACL 可以通过 Kubernetes 策略来应用实施。支持 IPAM。负载均衡能力不依赖 kube-proxy。网络地址转译 (NAT) 也不需要 iptables 或 netsh。	
NAT (未在 Kubernetes 中使用)	容器获得一个连接到某内部 vSwitch 的 vNIC 接口。DNS/DHCP 服务通过名为 WinNAT 的内部组件来提供。	MAC 地址和 IP 地址都被重写为宿主系统的 MAC 地址和 IP 地址。
nat	列在此表中仅出于完整性考虑	

如前所述，[Flannel](https://github.com/coreos/flannel) CNI [meta 插件](https://github.com/containernetworking/plugins/tree/master/plugins/meta/flannel) 在 Windows 上也是 [被支持](https://github.com/containernetworking/plugins/tree/master/plugins/meta/flannel#windows-support-experimental) 的，方法是通过 [VXLAN 网络后端](https://github.com/coreos/flannel/blob/master/Documentation/backends.md#vxlan) (**alpha 阶段**：委托给 win-overlay) 和 [主机-网关 \(host-gateway\) 网络后端](https://github.com/coreos/flannel/blob/master/Documentation/backends.md#host-gw) (稳定版本；委托给 win-bridge 实

现)。此插件支持将操作委托给所引用的 CNI 插件 (win-overlay、win-bridge) 之一, 从而能够与 Windows 上的 Flannel 守护进程 (Flanneld) 一同工作, 自动为节点分配子网租期, 创建 HNS 网络。该插件读入其自身的配置文件 (cni.conf), 并将其与 Flanneld 所生成的 subnet.env 文件中的环境变量整合, 之后将其操作委托给所引用的 CNI 插件之一以完成网络发现, 并将包含节点所被分配的子网信息的正确配置发送给 IPAM 插件 (例如 host-local)。

对于节点、Pod 和服务对象, 可针对 TCP/UDP 流量支持以下网络数据流:

- Pod -> Pod (IP 寻址)
- Pod -> Pod (名字寻址)
- Pod -> 服务 (集群 IP)
- Pod -> 服务 (部分限定域名, 仅适用于名称中不包含"."的情形)
- Pod -> 服务 (全限定域名)
- Pod -> 集群外部 (IP 寻址)
- Pod -> 集群外部 (DNS 寻址)
- 节点 -> Pod
- 节点 -> 节点

IP 地址管理 (IPAM)

Windows 上支持以下 IPAM 选项:

- [host-local](https://github.com/kubernetes-networking/plugins/tree/master/plugins/ipam/host-local)
- HNS IPAM (Inbox 平台 IPAM, 未指定 IPAM 时的默认设置)
- [Azure-vnet-ipam](https://github.com/Azure/azure-container-networking/blob/master/docs/ipam.md) (仅适用于 azure-cni)

负载均衡与服务

在 Windows 系统上, 你可以使用以下配置来设定服务和负载均衡行为:

功能特性	描述	支持的 Kubernetes 版本	支持的 Windows OS 版本	如何启用
会话亲和性	确保来自特定客户的连接每次都被交给同一 Pod。	v1.19+	Windows Server vNext Insider Preview Build 19551 或更高版本	将 <code>service.spec.sessionAffinity</code> 设置为 "ClientIP"
直接服务器返回	这是一种负载均衡模式, IP 地址的修正和负载均衡地址转译 (LBNAT) 直接在容器的 vSwitch 端口上处理; 服务流量到达时, 其源端 IP 地址设置为来源 Pod 的 IP。这种方案的延迟很低且可伸缩性好。	v1.15+	Windows Server 2004 版	为 kube-proxy 设置标志: <code>--feature-gates="WinDSR=true" --enable-dsr=true</code>
保留目标地址	对服务流量略过 DNAT 步骤, 这样就可以在到达后端 Pod 的报文中保留目标服务的虚拟 IP 地址。这一配置也会确保入站报文的客户端 IP 地址也被保留下来。	v1.15+	Windows Server 1903 或更高版本	在服务注解中设置 <code>"preserve-destination": "true"</code> 并启用 kube-proxy 中的 DSR 标志。
IPv4/IPv6 双栈网络	在集群内外同时支持原生的 IPv4-到-IPv4 和 IPv6-到-IPv6 通信。	v1.19+	Windows Server vNext Insider Preview Build 19603 或更高版本	参见 IPv4/IPv6 dual-stack

IPv4/IPv6 双栈支持

你可以通过使用 `IPv6DualStack` [特性门控](#) 来为 `l2bridge` 网络启用 IPv4/IPv6 双栈联网支持。进一步的细节可参见 [启用 IPv4/IPv6 双协议栈](#)。对 Windows 而言, 在 Kubernetes 中使用 IPv6 需要 Windows Server vNext Insider Preview Build 19603 或更高版本。

目前 Windows 上的覆盖网络 (VXLAN) 还不

支持双协议栈联网。

局限性

控制面

在 Kubernetes 架构和节点阵列中仅支持将 Windows 作为工作节点使用。这意味着 Kubernetes 集群必须总是包含 Linux 主控节点，零个或者多个 Linux 工作节点以及零个或者多个 Windows 工作节点。

计算

资源管理与进程隔离

Linux 上使用 Linux 控制组

(CGroups) 作为 Pod 的边界，以实现资源控制。容器都创建于这一边界之内，从而实现网络、进程和文件系统的隔离。控制组 CGroups API 可用来收集 CPU、I/O 和内存的统计信息。与此相比，Windows 为每个容器创建一个带有系统名字空间过滤设置的 Job 对象，以容纳容器中的所有进程并提供其与宿主系统间的逻辑隔离。没有现成的名字空间过滤设置是无法运行 Windows 容器的。这也意味着，系统特权无法在宿主环境中评估，因而 Windows 上也就不存在特权容器。归咎于独立存在的安全账号管理器 (Security Account Manager, SAM)，容器也不能获得宿主系统上的任何身份标识。

操作系统限制

Windows 有着严格的兼容性规则，宿主 OS 的版本必须与容器基准镜像 OS 的版本匹配。目前仅支持容器操作系统为 Windows Server 2019 的 Windows 容器。对于容器的 Hyper-V 隔离、允许一定程度上的 Windows 容器镜像版本向后兼容性等等，都是将来版本计划的一部分。

功能特性限制

- 终止宽限期 (Termination Grace Period)：未实现
- 单文件映射：将用 CRI-ContainerD 来实现
- 终止消息

- 特权容器：Windows 容器当前不支持
- 巨页 (Huge Pages)：Windows 容器当前不支持
- 现有的节点问题探测器 (Node Problem Detector) 仅适用于 Linux，且要求使用特权容器。一般而言，我们不设想此探测器能用于 Windows 节点，因为 Windows 不支持特权容器。
- 并非支持共享名字空间的所有功能特性 (参见 API 节以了解详细信息)

内存预留与处理

Windows 不像 Linux 一样有一个内存耗尽 (Out-of-memory) 进程杀手 (Process Killer) 机制。Windows 总是将用户态的内存分配视为虚拟请求，页面文件 (Pagefile) 是必需的。这一差异的直接结果是 Windows 不会像 Linux 那样出现内存耗尽的状况，系统会将进程内存页面写入磁盘而不会因内存耗尽而终止进程。当内存被过量使用且所有物理内存都被用光时，系统的换页行为会导致性能下降。

通过一个两步的过程是有可能将内存用量限制在一个合理的范围的。首先，使用 kubelet 参数 `--kubelet-reserve` 与/或 `--system-reserve` 来划分节点上的内存用量 (各容器之外)。这样做会减少节点可分配内存 ([NodeAllocatable](/zh/docs/tasks/administer-cluster/reserve-compute-resources/#node-allocatable))。在你部署工作负载时，对容器使用资源限制 (必须仅设置 limits 或者让 limits 等于 requests 值)。这也会从 NodeAllocatable 中耗掉部分内存量，从而避免在节点 负荷已满时调度器继续向节点添加 Pods。

避免过量分配的最佳实践是为 kubelet 配置至少 2 GB 的系统预留内存，以供 Windows、Docker 和 Kubernetes 进程使用。

参数的不同行为描述如下：

- `--kubelet-reserve`、`--system-reserve` 和 `--eviction-hard` 标志会更新节点可分配内存量
- 未实现通过使用 `--enforce-node-allocatable` 来完成的 Pod 驱逐
- 未实现通过使用 `--eviction-hard` 和 `--eviction-soft` 来完成的 Pod 驱逐
- `MemoryPressure` 状况未实现

- `kubelet` 不会采取措施来执行基于 OOM 的驱逐动作
- Windows 节点上运行的 kubelet 没有内存约束。`--kubelet-reserve` 和 `--system-reserve` 不会为 kubelet 或宿主系

统上运行的进程设限。这意味着 kubelet 或宿主系统上的进程可能导致内存资源紧张，而这一情况既不受节点可分配量影响，也不会被调度器感知。

存储

Windows 上包含一个分层的文件系统来挂载容器的分层，并会基于 NTFS 来创建一个 拷贝文件系统。容器中的所有文件路径都仅在该容器的上下文内完成解析。

- 卷挂载仅可针对容器中的目录进行，不可针对独立的文件
- 卷挂载无法将文件或目录投射回宿主文件系统
- 不支持只读文件系统，因为 Windows 注册表和 SAM 数据库总是需要写访问权限。不过，Windows 支持只读的卷。
- 不支持卷的用户掩码和访问许可，因为宿主与容器之间并不共享 SAM，二者之间不存在映射关系。所有访问许可都是在容器上下文中解析的。

因此，Windows 节点上不支持以下存储功能：

- 卷的子路径挂载；只能在 Windows 容器上挂载整个卷。
- 为 Secret 执行子路径挂载
- 宿主挂载投射
- 默认访问模式（因为该特性依赖 UID/GID）
- 只读的根文件系统；映射的卷仍然支持 `readOnly`
- 块设备映射
- 将内存作为存储介质
- 类似 UUID/GUID、每用户不同的 Linux 文件系统访问许可等文件系统特性
- 基于 NFS 的存储和卷支持
- 扩充已挂载卷（resizefs）

联网

Windows 容器联网与 Linux 联网有着非常重要的差别。 [Microsoft documentation for Windows Container Networking](https://docs.microsoft.com/en-us/virtualization/windowscontainers/container-networking/architecture) 中包含额外的细节和背景信息。

Windows 宿主联网服务和虚拟交换机实现了名字空间隔离，可以根据需要为 Pod 或容器 创建虚拟的网络接口（NICs）。不过，很多类似 DNS、路由、度量值之类的配置数据都 保存在 Windows 注册表数据库中而不是像 Linux 一样保存在 `/etc/...` 文件中。Windows 为容器提供的注册表与宿主系统的注册表是分离的，因此类似于将 `/etc/resolv.conf` 文件从宿主系统映射到容器中的做法不会产生与 Linux 系统相同的效果。这些信息必须在容器内部使用 Windows API 来配置。因此，CNI 实现需要调用 HNS，而不是依赖文件映射来将网络细节传递到 Pod 或容器中。

Windows 节点不支持以下联网功能：

- Windows Pod 不能使用宿主网络模式
- 从节点本地访问 NodePort 会失败（但从其他节点或外部客户端可访问）
- Windows Server 的未来版本中会支持从节点访问服务的 VIPs
- kube-proxy 的覆盖网络支持是 Alpha 特性。此外，它要求在 Windows Server 2019 上安装 [KB4482887](https://support.microsoft.com/en-us/help/4482887/windows-10-update-kb4482887) 补丁
- 本地流量策略和 DSR（保留目标地址）模式
- 连接到 l2bridge、l2tunnel 或覆盖网络的 Windows 容器不支持使用 IPv6 协议栈通信。要使得这些网络驱动能够支持 IPv6 地址需要在 Windows 平台上开展大量的工作，还需要在 Kubernetes 侧修改 kubelet、kube-proxy 以及 CNI 插件。
- 通过 win-overlay、win-bridge 和 Azure-CNI 插件使用 ICMP 协议向集群外通信。尤其是，Windows 数据面（[VFP](https://www.microsoft.com/en-us/research/project/azure-virtual-filtering-platform/)）不支持转换 ICMP 报文。这意味着：
 - 指向同一网络内目标地址的 ICMP 报文（例如 Pod 之间的 ping 通信）是可以工作的，没有局限性
 - TCP/UDP 报文可以正常工作，没有局限性
 - 指向远程网络的 ICMP 报文（例如，从 Pod 中 ping 外部互联网的通信）无法被转换，因此也无法被路由回到其源点。
 - 由于 TCP/UDP 包仍可被转换，用户可以将 `ping <目标>` 操作替换为 `curl <目标>` 以便能够调试与外部世界的网络连接。

Kubernetes v1.15 中添加了以下功能特性：

- `kubectl port-forward`

CNI 插件

Windows 参考网络插件 win-bridge 和 win-overlay 当前未实现 [CNI spec](https://github.com/kubernetes/networking/cni/blob/master/SPEC.md) v0.4.0，原

因是缺少检查用 (CHECK) 的实现。

- Windows 上的 Flannel VXLAN CNI 有以下局限性：
 - 其设计上不支持从节点到 Pod 的连接。只有在 Flannel v0.12.0 或更高版本后才有可能访问本地 Pods。
 - 我们被限制只能使用 VNI 4096 和 UDP 端口 4789。VNI 的限制正在被解决，会在将来的版本中消失 (开源的 Flannel 更改)。参见官方的 <https://github.com/coreos/flannel/blob/master/Documentation/backends.md#vxlan> 后端文档以了解关于这些参数的详细信息。

DNS

- 不支持 DNS 的 ClusterFirstWithHostNet 配置。Windows 将所有包含 "." 的名字视为全限定域名 (FQDN)，因而不会对其执行部分限定域名 (PQDN) 解析。
- 在 Linux 上，你可以有一个 DNS 后缀列表供解析部分限定域名时使用。在 Windows 上，我们只有一个 DNS 后缀，即与该 Pod 名字空间相关联的 DNS 后缀 (例如 `mydns.svc.cluster.local`)。Windows 可以解析全限定域名、或者恰好可用该后缀来解析的服务名称。例如，在 default 名字空间中生成的 Pod 会获得 DNS 后缀 `default.svc.cluster.local`。在 Windows Pod 中，你可以解析 `kubernetes.default.svc.cluster.local` 和 `kubernetes`，但无法解析二者之间的形式，如 `kubernetes.default` 或 `kubernetes.default.svc`。
- 在 Windows 上，可以使用的 DNS 解析程序有很多。由于这些解析程序彼此之间会有轻微的行为差别，建议使用 `Resolve-DNSName` 工具来完成名字查询解析。

IPv6

Windows 上的 Kubernetes 不支持单协议栈的"只用 IPv6"联网选项。不过，系统支持在 IPv4/IPv6 双协议栈的 Pod 和节点上运行单协议家族的服务。更多细节可参阅 [IPv4/IPv6 双协议栈联网](#) 一节。

会话亲和性

不支持使用 `service.spec.sessionAffinityConfig.clientIP.timeoutSeconds` 来为 Windows 服务设置最大会话粘滞时间。

安全性

Secret 以明文形式写入节点的卷中 (而不是像 Linux 那样写入内存或 tmpfs 中)。这意味着客户必须做以下两件事：

- 使用文件访问控制列表来保护 Secret 文件所在的位置
- 使用 <https://docs.microsoft.com/en-us/windows/security/information-protection/bitlocker/bitlocker-how-to-deploy-on-windows-server> BitLocker 来执行卷层面的加密

Windows 上目前不支持 </zh/docs/concepts/policy/pod-security-policy/#users-and-groups> `RunAsUser`。一种替代方案是在为容器打包时创建本地账号。将来的版本中可能会添加对 `RunAsUser` 的支持。

不支持特定于 Linux 的 Pod 安全上下文特权，例如 SELinux、AppArmor、Seccomp、权能字 (POSIX 权能字) 等等。

此外，如前所述，Windows 不支持特权容器。

API

对 Windows 而言，大多数 Kubernetes API 的工作方式没有变化。一些不易察觉的差别通常体现在 OS 和容器运行时上的不同。在某些场合，负载 API (如 Pod 或 Container) 的某些属性在设计时假定其在 Linux 上实现，因此会无法在 Windows 上运行。

在较高层面，不同的 OS 概念有：

- 身份标识 - Linux 使用证书类型来表示用户 ID (UID) 和组 ID (GID)。用户和组名没有特定标准，它们仅是 `/etc/groups` 或 `/etc/passwd` 中的别名表项，会映射回 UID+GID。Windows 使用一个更大的二进制安全标识符 (SID)，保存在 Windows 安全访问管理器 (Security Access Manager, SAM) 数据库中。此数据库并不在宿主系统与容器间，或者任意两个容器之间共享。
- 文件许可 - Windows 使用基于 SID 的访问控制列表，而不是基于 UID+GID 的访问权限位掩码。
- 文件路径 - Windows 上的习惯是使用 `\` 而非 `/`。Go 语言的 IO 库通常能够同时接受二者，并做出正确判断。不过当你在指定要在容器内解析的路径或命令行时，可能需要

使用 `\`。信号 - Windows 交互式应用以不同方式来处理终止事件，并可实现以下方式之一或组合：

- UI 线程处理包含 WM_CLOSE 在内的良定的消息
- 控制台应用使用控制处理程序来处理 Ctrl-C 或 Ctrl-Break
- 服务会注册服务控制处理程序，接受 SERVICE_CONTROL_STOP 控制代码

退出代码遵从相同的习惯，0 表示成功，非 0 值表示失败。特定的错误代码在 Windows 和 Linux 上可能会不同。不过，从 Kubernetes 组件（kubelet、kube-proxy）所返回的退出代码是没有变化的。

`V1.Container`

- `v1.Container.ResourceRequirements.limits.cpu` 和 `v1.Container.ResourceRequirements.limits.memory` - Windows 不对 CPU 分配设置硬性的限制。与之相反，Windows 使用一个份额（share）系统。基于毫核（millicores）的现有字段值会被缩放为相对的份额值，供 Windows 调度器使用。参见 <https://github.com/kubernetes/kubernetes/blob/master/pkg/kubelet/kuberuntime/helpers/windows.go> 和 <https://docs.microsoft.com/en-us/virtualization/windowscontainers/manage-containers/resource-controls> Microsoft 文档中关于资源控制的部分。
- Windows 容器运行时中没有实现巨页支持，因此相关特性不可用。巨页支持需要 <https://docs.microsoft.com/en-us/windows/desktop/Memory/large-page-support> 判定用户的特权 而这一特性无法在容器级别配置。
- `v1.Container.ResourceRequirements.requests.cpu` 和 `v1.Container.ResourceRequirements.requests.memory` - 请求值会从节点可分配资源中扣除，从而可用来避免节点上的资源过量分配。但是，它们无法用来在一个已经过量分配的节点上提供资源保障。如果操作员希望彻底避免过量分配，作为最佳实践，他们就需要为所有容器设置资源请求值。
- `v1.Container.SecurityContext.allowPrivilegeEscalation` - 在 Windows 上无法实现，对应的权能无一可在 Windows 上生效。
- `v1.Container.SecurityContext.Capabilities` - Windows 上未实现 POSIX 权能机制
- `v1.Container.SecurityContext.privileged` - Windows 不支持特权容器
- `v1.Container.SecurityContext.procMount` - Windows 不包含 `/proc` 文件系统
- `v1.Container.SecurityContext.readOnlyRootFilesystem` - 在 Windows 上无法实现，要在容器内使用注册表或运行系统进程就必需写访问权限。
- `v1.Container.SecurityContext.runAsGroup` - 在 Windows 上无法实现，没有 GID 支持
- `v1.Container.SecurityContext.runAsNonRoot` - Windows 上没有 root 用户。与之最接近的等价用户是 `ContainerAdministrator`，而该身份标识在节点上并不存在。
- `v1.Container.SecurityContext.runAsUser` - 在 Windows 上无法实现，因为没有作为整数支持的 GID。
- `v1.Container.SecurityContext.seLinuxOptions` - 在 Windows 上无法实现，因为没有 SELinux
- `V1.Container.terminationMessagePath` - 因为 Windows 不支持单个文件的映射，这一功能在 Windows 上也受限。默认值 `/dev/termination-log` 在 Windows 上也无法使用因为对应路径在 Windows 上不存在。

`V1.Pod`

`v1.Pod.hostIPC`、`v1.Pod.hostPID` - Windows 不支持共享宿主系统的名字空间

`v1.Pod.hostNetwork` - Windows 操作系统不支持共享宿主网络

`v1.Pod.dnsPolicy` - 不支持 `ClusterFirstWithHostNet`，因为 Windows 不支持宿主网络

`v1.Pod.podSecurityContext` - 参见下面的 `v1.PodSecurityContext`

`v1.Pod.shareProcessNamespace` - 此为 Beta 特性且依赖于 Windows 上未实现的 Linux 名字空间。Windows 无法共享进程名字空间或者容器的根文件系统。只能共享网络。

`v1.Pod.terminationGracePeriodSeconds` - 这一特性未在 Windows 版本的 Docker 中完全实现。参见[问题报告](https://github.com/moby/moby/issues/25982)。目前实现的行为是向 ENTRYPOINT 进程发送 CTRL SHUTDOWN EVENT 时间，之后 Windows 默认等待 5 秒钟，并最终使用正常的 Windows 关机行为关闭所有进程。这里的 5 秒钟默认值实际上保存在[容器内](https://github.com/moby/moby/issues/25982#issuecomment-426441183)的 Windows 注册表中，因此可以在构造容器时重载。

`v1.Pod.volumeDevices` - 此为 Beta 特性且未在 Windows 上实现。Windows 无法挂接原生的块设备到 Pod 中。

`v1.Pod.volumes` - `emptyDir`、`secret`、`configMap` 和 `hostPath` 都可正常工作且在 TestGrid 中测试。

- `v1.emptyDir.volumeSource` - Windows 上节点的默认介质是磁盘。不支持将内存作为介质，因为 Windows 不支持内置的 RAM 磁盘。

`v1.VolumeMount.mountPropagation` - Windows 上不支持挂载传播。

`v1-podsecuritycontext`

`V1.PodSecurityContext` 的所有选项在 Windows 上都无法工作。这些选项列在下面仅供参考。

- `v1.PodSecurityContext.seLinuxOptions` - Windows 上无 SELinux
- `v1.PodSecurityContext.runAsUser` - 提供 UID；Windows 不支持
- `v1.PodSecurityContext.runAsGroup` - 提供 GID；Windows 不支持
- `v1.PodSecurityContext.runAsNonRoot` - Windows 上没有 root 用户 最接近的等价账号是 `ContainerAdministrator`，而该身份标识在节点上不存在
- `v1.PodSecurityContext.supplementalGroups` - 提供 GID；Windows 不支持
- `v1.PodSecurityContext.sysctls` - 这些是 Linux sysctl 接口的一部分；Windows 上没有等价机制。

troubleshooting

获取帮助和故障排查

对你的 Kubernetes 集群进行排查的主要帮助信息来源应该是 [这份文档](/docs/tasks/debug-application-cluster/troubleshooting/)。该文档中包含了一些额外的、特定于 Windows 系统的故障排查帮助信息。Kubernetes 中日志是故障排查的一个重要元素。确保你在尝试从其他贡献者那里获得故障排查帮助时提供日志信息。你可以按照 SIG-Windows [贡献指南和收集日志](https://github.com/kubernetes/community/blob/master/sig-windows/CONTRIBUTING.md#gathering-logs) 所给的指令来操作。

我怎样知道 `start.ps1` 是否已成功完成？

你应该能看到节点上运行的 kubelet、kube-proxy 和（如果你选择 Flannel 作为联网方案）flanneld 宿主代理进程，它们的运行日志显示在不同的 PowerShell 窗口中。此外，你的 Windows 节点应该在你的 Kubernetes 集群 列举为 "Ready" 节点。

start="2"><p>我可以将 Kubernetes 节点进程配置为服务运行在后台么？</p><p>kubelet 和 kube-proxy 都已经被配置为以本地 Windows 服务运行，并且在出现失效事件（例如进程意外结束）时通过自动重启服务来提供一定的弹性。你有两种办法将这些节点组件配置为服务。</p><p>以本地 Windows 服务的形式</p><p>Kubelet 和 kube-proxy 可以用 `sc.exe` 以本地 Windows 服务的形式运行：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell"># 用两个单独的命令为 kubelet 和 kube-proxy 创建服务 sc.exe create <组件名称> binPath= "<可执行文件路径> -service <其它参数>";# 请注意如果参数中包含空格，必须使用转义 sc.exe create kubelet binPath= "C:\kubelet.exe --service --hostname-override 'minion' <其它参数>";# 启动服务 Start-Service kubelet Start-Service kube-proxy # 停止服务 Stop-Service kubelet (-Force) Stop-Service kube-proxy (-Force) # 查询服务状态 Get-Service kubelet Get-Service kube-proxy </code></pre></div><ol start="2"><p>使用 nssm.exe</p><p>你也总是可以使用替代的服务管理器，例如nssm.exe，来为你在后台运行这些进程（<code>flanneld</code>、<code>kubelet</code> 和 <code>kube-proxy</code>）。你可以使用这一示例脚本，利用<code>nssm.exe</code> 将 <code>kubelet</code>、<code>kube-proxy</code> 和 <code>flanneld.exe</code> 注册为要在后台运行的 Windows 服务。</p><div class="highlight"><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 <网络模式> -ManagementIP <Windows 节点 IP> -ClusterCIDR <集群子网> -KubeDnsServiceIP <kube-dns 服务 IP> -LogDir <日志目录> # NetworkMode = 网络模式 l2bridge (flannel host-gw , 也是默认值) 或 overlay (flannel vxlan) 选做网络方案 # ManagementIP = 分配给 Windows 节点的 IP 地址。你可以使用 ipconfig 得到此值 # ClusterCIDR = 集群子网范围 (默认值为 10.244.0.0/16) # KubeDnsServiceIP = Kubernetes DNS 服务 IP (默认值为 10.96.0.10) # LogDir = kubelet 和 kube-proxy 的日志会被重定向到这一目录中的对应输出文件，默认值为 `C:\k`。 </code></pre></div><p>若以上所引用的脚本不适合，你可以使用下面的例子手动配置 <code>nssm.exe</code>。</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-powershell" data-lang="powershell"># 注册 flanneld.exe nssm install flanneld C:\flannel\flanneld.exe nssm</div>

```

>set </span>flanneld AppParameters --
kubeconfig>-file</span>=c:\k\config --
iface=&lt;ManagementIP&gt; --ip-masq=1 --kube-subnet-mgr=1 nssm
>set </span>flanneld AppEnvironmentExtra
NODE_NAME=&lt;主机名&gt; nssm >set </
span>flanneld AppDirectory C:\flannel nssm >start
</span>flanneld ># 注册
kubelet.exe</span> ># Microsoft
在 mcr.microsoft.com/k8s/core/pause:1.2.0 发布其 pause 基础设施容器</span>
nssm install kubelet C:\k\kubelet.exe nssm >set </
span>kubelet AppParameters --hostname-override=&lt;hostname&gt; --v=6
--pod-infra-container-image=mcr.microsoft.com/k8s/core/pause>:</
span>1.2.0 --resolv-conf=>""</span> --allow-
privileged=true --enable-debugging-handlers --cluster-dns=&lt;DNS 服务
IP&gt; --cluster-domain=cluster.local --kubeconfig=c:\k\config --hairpin-
mode=promiscuous-bridge --image-pull-progress-deadline=20m --cgroups-
per-qos=false --log-dir=&lt;log directory&gt; --logtostderr=false --enforce-
node-allocatable=>""</span> --network-
plugin=cni --cni-bin-dir=c:\k\cni --cni-conf-dir=c:\k\cni\config nssm >set </span>kubelet AppDirectory C:\k nssm >start </span>kubelet ># 注册 kube-proxy.exe (l2bridge / host-gw)</span> nssm install
kube-proxy C:\k\kube-proxy.exe nssm >set </
span>kube-proxy AppDirectory c:\k nssm >set </
span>kube-proxy AppParameters --v=4 --proxy-mode=kernel-space --
hostname-override=&lt;主机名&gt;--kubeconfig=c:\k\config --enable-
dsr=false --log-dir=&lt;日志目录&gt; --logtostderr=false nssm.exe >set </span>kube-proxy AppEnvironmentExtra
KUBE_NETWORK=cbr0 nssm >set </span>kube-
proxy DependOnService kubelet nssm >start </
span>kube-proxy ># 注册 kube-
proxy.exe (overlay / vxlan)</span> nssm install kube-proxy C:\k\kube-
proxy.exe nssm >set </span>kube-proxy
AppDirectory c:\k nssm >set </span>kube-proxy
AppParameters --v=4 --proxy-mode=kernel-space --feature-gates=>"WinOverlay=true"</span> --hostname-override=&lt;主
机名&gt; --kubeconfig=c:\k\config --network-name=vxlan0 --source-vip=&lt;
源端 VIP&gt; --enable-dsr=false --log-dir=&lt;日志目录&gt; --
logtostderr=false nssm >set </span>kube-proxy
DependOnService kubelet nssm >start </
span>kube-proxy </code></pre></div><p>作为初始的故障排查操作，你可以使
用在 <a href="https://nssm.cc/">nssm.exe</a> 中使用下面的标志 以便将标准输
出和标准错误输出重定向到一个输出文件：</p><div class="highlight"><pre
style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:
4"><code class="language-powershell" data-lang="powershell">nssm
>set </span>&lt;服务名称&gt; AppStdout C:
\k\mysvc.log nssm >set </span>&lt;服务名称&gt;
AppStderr C:\k\mysvc.log </code></pre></div><p>要了解更多的细节，可参
见官方的 <a href="https://nssm.cc/usage">nssm 用法</a> 文档。</p></li></
ol></li></ol><ol start="3"><li><p>我的 Windows Pods 无法连接网络</
p><p>如果你在使用虚拟机，请确保 VM 网络适配器均已开启 MAC 侦听

```

(Spoofing) 。

- 我的 Windows Pods 无法 ping 外部资源

Windows Pods 目前没有为 ICMP 协议提供出站规则。不过 TCP/UDP 是支持的。尝试与集群外资源连接时，可以将 `ping <IP>` 命令替换为对应的 `curl <IP>` 命令。

如果你还遇到问题，很可能你在 <https://github.com/Microsoft/SDN/blob/master/Kubernetes/flannel/l2bridge/cni/config/cni.conf> 中的网络配置值得额外的注意。你总是可以编辑这一静态文件。配置的更新会应用到所有新创建的 Kubernetes 资源上。

Kubernetes 网络的需求之一（参见 </zh/docs/concepts/cluster-administration/networking/> Kubernetes 模型）是集群内部无需网络地址转译（NAT）即可实现通信。为了符合这一要求，对所有我们不希望出站时发生 NAT 的通信都存在一个 [ExceptionList](https://github.com/Microsoft/SDN/blob/master/Kubernetes/flannel/l2bridge/cni/config/cni.conf#L20)。然而这也意味着你需要将你要查询的外部 IP 从 ExceptionList 中移除。只有这时，从你的 Windows Pod 发起的网络请求才会被正确地通过 SNAT 转换以接收到来自外部世界的响应。就此而言，你在 `cni.conf` 中的 `ExceptionList` 应该看起来像这样：

```
ExceptionList: [ "10.244.0.0/16", # 集群子网 "10.96.0.0/12", # 服务子网 "10.127.130.0/24" # 管理（主机）子网 ]
```
- 我的 Windows 节点无法访问 NodePort 服务

从节点自身发起的本地 NodePort 请求会失败。这是一个已知的局限。NodePort 服务的访问从其他节点或者外部客户端都可正常进行。
- 容器的 vNICs 和 HNS 端点被删除了

这一问题可能因为 `hostname-override` 参数未能传递给 [kube-proxy](/docs/reference/command-line-tools-reference/kube-proxy/) 而导致。解决这一问题时，用户需要按如下方式将主机名传递给 kube-proxy：

```
C:\k\kube-proxy.exe --hostname-override=$(hostname)
```
- 使用 Flannel 时，我的节点在重新加入集群时遇到问题

无论何时，当一个之前被删除的节点被重新添加到集群时，flannelD 都会将为节点分配一个新的 Pod 子网。用户需要将下面路径中的老的 Pod 子网配置文件删除：

```
Remove-Item C:\k\SourceVip.json
Remove-Item C:\k\SourceVipRequest.json
```
- 在启动了 `start.ps1` 之后，flanneld 一直停滞在 "Waiting for the Network to be created" 状态

关于这一 [正在被分析的问题](https://github.com/coreos/flannel/issues/1066) 有很多的报告；最可能的一种原因是关于何时设置 Flannel 网络的管理 IP 的时间问题。一种解决办法是重新启动 `start.ps1` 或者按如下方式手动重启之：

```
PS C:> [Environment]::SetEnvironmentVariable("NODE_NAME", "&lt;Windows 工作节点主机名&gt;")
PS C:> C:\flannel\flanneld.exe --
```

```
kubeconfig<span style="color:#666">-file</span>=c:\k\config --
iface=&lt;Windows 工作节点 IP&gt; --ip-masq=1 --kube-subnet-mgr=1 </
code></pre></div></li></ol><ol start="9"><li><p>我的 Windows Pods
无法启动，因为缺少 <code>/run/flannel/subnet.env</code> 文件</p><p>这表明 Flannel 网络未能正确启动。你可以尝试重启 flanneld.exe 或者将文件手动地从
Kubernetes 主控节点的 <code>/run/flannel/subnet.env</code> 路径复制到
Windows 工作节点的 <code>C:\run\flannel\subnet.env</code> 路径，并将
<code>FLANNEL SUBNET</code> 行改为一个不同的数值。例如，如果期望节
点子网为 <code>10.244.4.1/24</code>：</p><div class="highlight"><pre
style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:
4"><code class="language-env" data-lang="env"><span
style="color:#b8860b">FLANNEL NETWORK</span><span
style="color:#666">=</span>10.244.0.0/16 <span
style="color:#b8860b">FLANNEL SUBNET</span><span
style="color:#666">=</span>10.244.4.1/24 <span
style="color:#b8860b">FLANNEL MTU</span><span
style="color:#666">=</span><span style="color:#666">1500</span>
<span style="color:#b8860b">FLANNEL IPMASQ</span><span
style="color:#666">=</span><span style="color:#a2f">true</span> </
code></pre></div></li></ol><ol start="10"><li><p>我的 Windows 节点
无法使用服务 IP 访问我的服务</p><p>这是 Windows 上当前网络协议栈的一个已知的
限制。Windows Pods 能够访问服务 IP。</p></li></ol><ol
start="11"><li><p>启动 kubelet 时找不到网络适配器</p><p>Windows 网络堆
栈需要一个虚拟的适配器，这样 Kubernetes 网络才能工作。如果下面的命令（在管理
员 Shell 中）没有任何返回结果，证明虚拟网络创建（kubelet 正常工作的必要前提之
一）失败了：</p><div class="highlight"><pre style="background-
color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code
class="language-powershell" data-lang="powershell"><span
style="color:#a2f">Get-HnsNetwork</span> | ? Name <span
style="color:#666">-ieq</span> <span style="color:#b44">"cbr0"</span>
<span style="color:#a2f">Get-NetAdapter</span> | ? Name <span
style="color:#666">-Like</span> <span style="color:#b44">"vEthernet
(Ethernet*"</span> </code></pre></div><p>当宿主系统的网络适配器名称不
是 "Ethernet" 时，通常值得更改 <code>start.ps1</code> 脚本中的 <a
href="https://github.com/microsoft/SDN/blob/master/Kubernetes/flannel/
start.ps1#L7">InterfaceName</a> 参数来重试。否则可以查验 <code>start-
kubelet.ps1</code> 的输出，看看是否在虚拟网络创建过程中报告了其他错误。</
p></li></ol><ol start="12"><li><p>我的 Pods 停滞在 "Container Creating"
状态或者反复重启</p><p>检查你的 pause 镜像是与你的 OS 版本兼容的。<a
href="https://docs.microsoft.com/en-us/virtualization/windowscontainers/
kubernetes/deploying-resources">这里的指令</a> 假定你的 OS 和容器版本都是
1803。如果你安装的是更新版本的 Windows，比如说 某个 Insider 构造版本，你需要
相应地调整要使用的镜像。请参照 Microsoft 的 <a href="https://
hub.docker.com/u/microsoft/"> Docker 仓库</a> 了解镜像。不管怎样，pause 镜
像的 Dockerfile 和示例服务都期望镜像的标签为 <code>:latest</code>。</
p><p>从 Kubernetes v1.14 版本起，Microsoft 开始在
<code>mcr.microsoft.com/k8s/core/pause:1.2.0</code> 发布其 pause 基础设
施容器。</p></li></ol><ol start="13"><li><p>DNS 解析无法正常工作</
p><p>参阅 Windows 上 <a href="#dns-limitations">DNS 相关的局限</a>
节。</p></li></ol><ol start="14"><li><p><code>kubect
l port-forward</code>
```

`>` 失败，错误信息为 "unable to do port forwarding: wincat not found" `</p><p>`此功能是在 Kubernetes v1.15 中实现的，pause 基础设施容器为 `<code>mcr.microsoft.com/k8s/core/pause:1.2.0</code>`。请确保你使用的是这些版本或者更新版本。如果你想要自行构造你自己的 pause 基础设施容器，要确保其中包含了 `wincat</p><ol start="15"><p>`我的 Kubernetes 安装失败，因为我的 Windows Server 节点在防火墙后面`</p><p>`如果你处于防火墙之后，那么必须定义如下 PowerShell 环境变量：`</p><div class="highlight"><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></div><ol start="15"><p><code>pause</code>`容器是什么？`</p><p>`在一个 Kubernetes Pod 中，一个基础设施容器，或称 "pause" 容器，会被首先创建出来，用以托管容器端点。属于同一 Pod 的容器，包括基础设施容器和工作容器，会共享相同的网络名字空间和端点（相同的 IP 和端口空间）。我们需要 pause 容器来工作容器崩溃或重启的状况，以确保不会丢失任何网络配置。`</p><p>`"pause"（基础设施）镜像托管在 Microsoft Container Registry (MCR) 上。你可以使用 `<code>docker pull mcr.microsoft.com/k8s/core/pause:1.2.0</code>` 来访问它。要了解进一步的细节，可参阅 `DOCKERFILE`。`</p><h3 id="further-investigation">进一步探究</h3><p>`如果以上步骤未能解决你遇到的问题，你可以通过以下方式获得在 Kubernetes 中的 Windows 节点上运行 Windows 容器的帮助：`</p>StackOverflow Windows Server Container 主题Kubernetes 官方论坛 discuss.kubernetes.ioKubernetes Slack #SIG-Windows 频道<h2 id="reporting-issues-and-feature-requests">报告问题和功能需求</h2><p>`如果你遇到看起来像是软件缺陷的问题，或者你想要提起某种功能需求，请使用 `GitHub 问题跟踪系统`。你可以在 `GitHub` 上发起 Issue 并将其指派给 SIG-Windows。你应该首先搜索 Issue 列表，看看是否该 Issue 以前曾经被报告过，以评论形式将你在该 Issue 上的体验追加进去，并附上额外的日志信息。SIG-Windows Slack 频道也是一个获得初步支持的好渠道，可以在生成新的 Ticket 之前对一些想法进行故障分析。`</p><p>`在登记软件缺陷时，请给出如何重现该问题的详细信息，例如：`</p>Kubernetes 版本：kubectl 版本环境细节：云平台、OS 版本、网络选型和配置情况以及 Docker 版本重现该问题的详细步骤相关的日志通过为该`

Issue 添加 `<sig windows>` 评论为其添加 `<sig/windows>` 标签，进而引起 SIG-Windows 成员的注意。

接下来

在我们的未来蓝图中包含很多功能特性（要实现）。下面是一个浓缩的简要列表，不过我们鼓励你查看我们的 [roadmap 项目](https://github.com/orgs/kubernetes/projects/8) 并通过 [贡献](https://github.com/kubernetes/community/blob/master/sig-windows/) 的方式 帮助我们 把 Windows 支持做得更好。

Hyper-V 隔离

要满足 Kubernetes 中 Windows 容器的如下用例，需要利用 Hyper-V 隔离：

- 在 Pod 之间实施基于监管程序（Hypervisor）的隔离，以增强安全性
- 出于向后兼容需要，允许添加运行新 Windows Server 版本的节点时不必重新创建容器
- 为 Pod 设置特定的 CPU/NUMA 配置
- 实施内存隔离与预留

现有的 Hyper-V 隔离支持是添加自 v1.10 版本的实验性功能特性，会在未来版本中弃用，向前文所提到的 CRI-ContainerD 和 RuntimeClass 特性倾斜。要使用当前的功能特性并创建 Hyper-V 隔离的容器，需要在启动 kubelet 时设置特性门控

`HyperVContainer=true`，同时为 Pod 添加注解

`experimental.windows.kubernetes.io/isolation-type=hyperv`。在实验性实现版本中，此功能特性限制每个 Pod 中只能包含一个容器。

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: iis
  spec:
    selector:
      matchLabels:
        app: iis
    replicas: 3
    template:
      metadata:
        labels:
          app: iis
      annotations:
```

```
style="color:#bbb"> </span><span style="color:#bbb"> </span><span
style="color:green;font-weight:700">experimental.windows.kubernetes.io/
isolation-type</span>:<span style="color:#bbb"> </span>hyperv<span
style="color:#bbb"> </span><span style="color:#bbb"> </span><span
style="color:green;font-weight:700">spec</span>:<span
style="color:#bbb"> </span><span style="color:#bbb"> </span><span
style="color:green;font-weight:700">containers</span>:<span
style="color:#bbb"> </span><span style="color:#bbb"> </span>- <span
style="color:green;font-weight:700">name</span>:<span
style="color:#bbb"> </span>iis<span style="color:#bbb"> </span><span
style="color:#bbb"> </span><span style="color:green;font-weight:
700">image</span>:<span style="color:#bbb"> </span>microsoft/
iis<span style="color:#bbb"> </span><span style="color:#bbb"> </
span><span style="color:green;font-weight:700">ports</span>:<span
style="color:#bbb"> </span><span style="color:#bbb"> </span>- <span
style="color:green;font-weight:700">containerPort</span>:<span
style="color:#bbb"> </span><span style="color:#666">80</span><span
style="color:#bbb"> </span></code></pre></div><h3 id="deployment-
with-kubeadm-and-cluster-api">使用 kubeadm 和 Cluster API 来部署</
h3><p>kubeadm 已经成为用户部署 Kubernetes 集群的事实标准。kubeadm 对
Windows 节点的支持目前还在开发过程中，不过你可以阅读相关的 <a href="/zh/
docs/tasks/administer-cluster/kubeadm/adding-windows-nodes/">指南</a>。
我们也在投入资源到 Cluster API，以确保 Windows 节点被正确配置。</p><h3
id="a-few-other-key-features">若干其他关键功能</h3><ul><li>为组管理的服
务账号（Group Managed Service Accounts，GMSA）提供 Beta 支持</li><li>
添加更多的 CNI 支持</li><li>实现更多的存储插件</li></ul><div id="pre-
footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您有帮助？
</p><button class="btn btn-primary mb-4 feedback-yes">是</button>
<button class="btn btn-primary mb-4 feedback-no">否</button><p
class="feedback-response feedback-response hidden">感谢反馈。如果您有
一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 <a
target=" blank" rel="noopener" href="https://stackoverflow.com/questions/
tagged/kubernetes">Stack Overflow</a>。在 GitHub 仓库上登记新的问题 <a
class="feedback-link" target=" blank" rel="noopener" href="https://
github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io">
报告问题</a> 或者 <a class="feedback-link" target=" blank" rel="noopener"
href="https://github.com/kubernetes/website/issues/new?
title=Improvement%20for%20k8s.io">提出改进建议</a>。</p></
div><script>const yes=document.querySelector('.feedback-yes');const
no=document.querySelector('.feedback-
no');document.querySelectorAll('.feedback-link').forEach(link=&gt;{
link.href=link.href+window.location.pathname;});const
sendFeedback=(value)=&gt;{if(!gtag){console.log('!gtag');}
gtag('event','click',
{'event category':'Helpful','event label':window.location.pathname,value});};const
disableButtons=()=&gt;{yes.disabled=true;yes.classList.add('feedback-
button disabled');no.disabled=true;no.classList.add('feedback-
button disabled');};yes.addEventListener('click',)=&gt;{
sendFeedback(1);disableButtons();document.querySelector('.feedback-
response').classList.remove('feedback-
response hidden');});no.addEventListener('click',)=&gt;
```



```
{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response hidden');});</script><div class="text-muted mt-5 pt-3 border-top">最后修改 October 22, 2020 at 11:26 AM PST: <a href="https://github.com/kubernetes/website/commit/4abcae9498e5c0ad005d6511dc81064520d475ff">[zh] Translate docs/tasks/configure-pod-container/configure-gmsa.md (4abcae949)</a></div></div><div class="td-content"><h1>Kubernetes 中调度 Windows 容器的指南</h1><p>Windows 应用程序构成了许多组织中运行的服务和应用程序的很大一部分。本指南将引导您完成在 Kubernetes 中配置和部署 Windows 容器的步骤。</p><h2 id="#x76EE;#x6807;">目标</h2><ul><li>配置一个示例 deployment 以在 Windows 节点上运行 Windows 容器</li><li>(可选) 使用组托管服务帐户 (GMSA) 为您的 Pod 配置 Active Directory 身份</li></ul><h2 id="#x5728;#x4F60;#x5F00;#x59CB;#x4E4B;#x524D;">在你开始之前</h2><ul><li>创建一个 Kubernetes 集群，其中包括一个 <a href="/zh/docs/tasks/administer-cluster/kubeadm/adding-windows-nodes/">运行 Windows 服务器的主节点和工作节点</a></li><li>重要的是要注意，对于 Linux 和 Windows 容器，在 Kubernetes 上创建和部署服务和工作负载的行为几乎相同。与集群接口的 <a href="/zh/docs/reference/kubectl/overview/">Kubectl 命令</a>相同。提供以下部分中的示例只是为了快速启动 Windows 容器的使用体验。</li></ul><h2 id="#x5165;#x95E8;-#x90E8;#x7F72;-windows-#x5BB9;#x5668;">入门：部署 Windows 容器</h2><p>要在 Kubernetes 上部署 Windows 容器，您必须首先创建一个示例应用程序。下面的示例 YAML 文件创建了一个简单的 Web 服务器应用程序。创建一个名为 <code>win-webserver.yaml</code> 的服务规约，其内容如下：</p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-yaml" data-lang="yaml"><span style="color:green;font-weight:700">apiVersion</span>:<span style="color:#bbb"> </span><span style="color:#bbb">v1</span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:green;font-weight:700">kind</span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:green;font-weight:700">Service</span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:green;font-weight:700">metadata</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:green;font-weight:700">name</span>:<span style="color:#bbb"> </span><span style="color:#bbb">win-webserver</span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:green;font-weight:700">labels</span>:<span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:#bbb"></span><span style="color:green;font-weight:700">app</span>:<span style="color:#bbb"> </span><span style="color:#bbb">win-webserver</span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:green;font-weight:700">spec</span>:<span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:#bbb"></span><span style="color:green;font-weight:700">ports</span>:<span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:#bbb"></span><span style="color:#080;font-style:italic"># the port that this service should serve on</span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:#bbb"></span><span style="color:green;font-weight:700">port</span>:<span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:#666">80</span><span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:green;font-weight:700">targetPort</span>:<span style="color:#bbb"> </span><span style="color:#bbb"></span><span style="color:#666">80</span><span style="color:#bbb"> </span><span style="color:#bbb"></span></pre></div></div>
```

[illegible]

```

span>windowserver<span style="color:#bbb"> </span><span
style="color:#bbb"> </span><span style="color:green;font-weight:
700">image</span>:<span style="color:#bbb"> </
span>mcr.microsoft.com/windows/servercore:ltsc2019<span
style="color:#bbb"> </span><span style="color:#bbb"> </span><span
style="color:green;font-weight:700">command</span>:<span
style="color:#bbb"> </span><span style="color:#bbb"> </span>-
powershell.exe<span style="color:#bbb"> </span><span
style="color:#bbb"> </span>- -command<span style="color:#bbb"> </
span><span style="color:#bbb"> </span>- <span
style="color:#b44">"&lt;#code used from https://gist.github.com/
19WAS85/5424431#&gt; ; $$listener = New-Object
System.Net.HttpListener ; $$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('&gt; {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='&lt;html&gt;&lt;body&gt;&lt;H1&gt;Windows Container Web
Server&lt;/H1&gt;' ; $$callerCountsString='' ; $$callerCounts.Keys | % { $
$callerCountsString+=''&lt;p&gt;IP {0} callerCount {1} ' -f $
$ip[1].IPAddress,$$callerCounts.Item($$ ) } ; $$footer='&lt;/body&gt;&lt;/
html&gt;' ; $$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() ; $$responseStatus = $response.StatusCode ; Write-
Host('&lt; {0}' -f $$responseStatus) } ; "</span><span
style="color:#bbb"> </span><span style="color:#bbb"> </span><span
style="color:green;font-weight:700">nodeSelector</span>:<span
style="color:#bbb"> </span><span style="color:#bbb"> </span><span
style="color:green;font-weight:700">kubernetes.io/os</span>:<span
style="color:#bbb"> </span>windows<span style="color:#bbb"> </
span></code></pre></div><blockquote class="note
callout"><div><strong>说明：</strong> 端口映射也是支持的，但为简单起见，
在此示例中容器端口 80 直接暴露给服务。</div></blockquote><ol><li><p>检
查所有节点是否健康：</p><div class="highlight"><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></div></li><li><p>部署服务并观察 pod 更新：</p><div
class="highlight"><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></div><p>正确部署服务后，两个 Pod 都标记为"Ready"。要退
出 watch 命令，请按 Ctrl + C。</p></li><li><p>检查部署是否成功。验证：</
p><ul><li>Windows 节点上每个 Pod 有两个容器，使用 <code>docker ps</
code></li><li>Linux 主机列出两个 Pod，使用 <code>kubectl get pods</

```

`>`

- 跨网络的节点到 Pod 通信，从 Linux 主服务器 `curl` 您的 pod IPs 的端口80，以检查 Web 服务器响应
- Pod 到 Pod 的通信，使用 `docker exec` 或 `kubectl exec` 在 pod 之间（以及跨主机，如果您有多个 Windows 节点）进行 ping 操作
- 服务到 Pod 的通信，从 Linux 主服务器和各个 Pod 中 `curl` 虚拟服务 IP（在 `kubectl get services` 下可见）
- 服务发现，使用 Kubernetes `curl` 服务名称[默认 DNS 后缀](/zh/docs/concepts/services-networking/dns-pod-service/#services)
- 入站连接，从 Linux 主服务器或集群外部的计算机 `curl` NodePort
- 出站连接，使用 `kubectl exec` 从 Pod 内部 `curl` 外部 IP

说明： 由于当前平台对 Windows 网络堆栈的限制，Windows 容器主机无法访问在其上调度的服务的 IP。只有 Windows pods 才能访问服务 IP。

使用可配置的容器用户名

从 Kubernetes v1.16 开始，可以为 Windows 容器配置与其镜像默认值不同的用户名来运行其入口点和进程。此能力的实现方式和 Linux 容器有些不同。在[此处](/zh/docs/tasks/configure-pod-container/configure-runasusername/)可了解更多信息。

使用组托管服务帐户管理工作负载身份

从 Kubernetes v1.14 开始，可以将 Windows 容器工作负载配置为使用组托管服务帐户（GMSA）。组托管服务帐户是 Active Directory 帐户的一种特定类型，它提供自动密码管理，简化的服务主体名称（SPN）管理以及将管理委派给跨多台服务器的其他管理员的功能。配置了 GMSA 的容器可以访问外部 Active Directory 域资源，同时携带通过 GMSA 配置的身份。在[此处](/zh/docs/tasks/configure-pod-container/configure-gmsa/)了解有关为 Windows 容器配置和使用 GMSA 的更多信息。

污点和容忍度

目前，用户需要将 Linux 和 Windows 工作负载运行在各自特定的操作系统的节点上，因而需要结合使用污点和节点选择算符。这可能仅给 Windows 用户造成不便。推荐的方法概述如下，其主要目标之一是该方法不应破坏与现有 Linux 工作负载的兼容性。

确保特定操作系统的工作负载落在适当的容器主机上

用户可以使用污点和容忍度确保 Windows 容器可以调度在适当的主机上。目前所有 Kubernetes 节点都具有以下默认标签：

- `kubernetes.io/os = [windows|linux]`
- `kubernetes.io/arch = [amd64|arm64|...]`

如果 Pod 规范未指定诸如 `"kubernetes.io/os": windows` 之类的

nodeSelector，则该 Pod 可能会被调度到任何主机（Windows 或 Linux）上。这是有问题的，因为 Windows 容器只能在 Windows 上运行，而 Linux 容器只能在 Linux 上运行。最佳实践是使用 nodeSelector。

但是，我们了解到，在许多情况下，用户都有既存的大量的 Linux 容器部署，以及一个现成的配置生态系统，例如社区 Helm charts，以及程序化 Pod 生成案例，例如 Operators。在这些情况下，您可能不会愿意更改配置添加 nodeSelector。替代方法是使用污点。由于 kubelet 可以在注册期间设置污点，因此可以轻松修改它，使其仅在 Windows 上运行时自动添加污点。

例如：`--register-with-taints='os=windows:NoSchedule'`

向所有 Windows 节点添加污点后，Kubernetes 将不会在它们上调度任何负载（包括现有的 Linux Pod）。为了使某 Windows Pod 调度到 Windows 节点上，该 Pod 既需要 nodeSelector 选择 Windows，也需要合适的匹配的容忍度设置。

```
code class="language-yaml" data-
```

```
lang="yaml"><span style="color:green;font-weight:700">nodeSelector</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">kubernetes.io/os</span>:<span style="color:#bbb"> </span>windows<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">node.kubernetes.io/windows-build</span>:<span style="color:#bbb"> </span><span style="color:#b44">'10.0.17763'</span><span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">tolerations</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span>- <span style="color:green;font-weight:700">key</span>:<span style="color:#bbb"> </span><span style="color:#b44">"os"</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">operator</span>:<span style="color:#bbb"> </span><span style="color:#b44">"Equal"</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">value</span>:<span style="color:#bbb"> </span><span style="color:#b44">"windows"</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">effect</span>:<span style="color:#bbb"> </span><span style="color:#b44">"NoSchedule"</span><span style="color:#bbb"> </span></span></code></pre></div><h3
```

id="处理同一集群中的

windows-版本">处理同一集群中的多个 Windows 版本</p></h3><p>每个 Pod 使用的 Windows Server 版本必须与该节点的 Windows Server

版本相匹配。如果要在同一集群中使用多个 Windows Server 版本，则应该设置其他节点标签和 nodeSelector。</p><p>Kubernetes 1.17 自动添加了一个新标签

<code>node.kubernetes.io/windows-build</code> 来简化此操作。如果您运行的是旧版本，则建议手动将此标签添加到 Windows 节点。</p><p>此标签反映了需要兼容的 Windows 主要、次要和内部版本号。以下是当前每个 Windows Server 版本使用的值。</p><table><thead><tr><th>产品名称</th><th>内部编号</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="使用-runtimeclass-简化">使用

RuntimeClass 简化</h3><p>RuntimeClass 可用于简化使用污点和容忍度的过程。集群管理员可以创建 <code>RuntimeClass</code> 对象，用于封装这些污点和容忍度。</p><p>将此文件保存到 <code>runtimeClasses.yml</code> 文件。它包括适用于 Windows 操作系统、体系结构和版本的 <code>nodeSelector</code>。</p><div class="highlight"><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:

RuntimeClass 可用于简化使用污点和容忍度的过程。集群管理员可以创建 <code>RuntimeClass</code> 对象，用于封装这些污点和容忍度。</p><p>将此文件保存到 <code>runtimeClasses.yml</code> 文件。它包括适用于 Windows 操作系统、体系结构和版本的 <code>nodeSelector</code>。</p><div class="highlight"><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:

class="language-yaml" data-lang="yaml">apiVersion: node.k8s.io/v1 kind: RuntimeClass metadata:

class="language-yaml" data-lang="yaml">apiVersion: node.k8s.io/v1 kind: RuntimeClass metadata:

class="language-yaml" data-lang="yaml">apiVersion: node.k8s.io/v1 kind: RuntimeClass metadata:

class="language-yaml" data-lang="yaml">apiVersion: node.k8s.io/v1 kind: RuntimeClass metadata:

```
style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">name</span>:<span style="color:#bbb"> </span>windows-2019<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">handler</span>:<span style="color:#bbb"> </span><span style="color:#b44">'docker'</span><span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">scheduling</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">nodeSelector</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">kubernetes.io/os</span>:<span style="color:#bbb"> </span><span style="color:#b44">'windows'</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">kubernetes.io/arch</span>:<span style="color:#bbb"> </span><span style="color:#b44">'amd64'</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">node.kubernetes.io/windows-build</span>:<span style="color:#bbb"> </span><span style="color:#b44">'10.0.17763'</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">tolerations</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">effect</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">NoSchedule</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">key</span>:<span style="color:#bbb"> </span>os<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">operator</span>:<span style="color:#bbb"> </span>Equal<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">value</span>:<span style="color:#bbb"> </span><span style="color:#b44">"windows"</span><span style="color:#bbb"> </span></code></pre></div></li></ol><ol start="2"><li>集群管理员运行<code>kubectl create -f runtimeClasses.yml</code> 操作</li><li>根据需要向 Pod 规约中添加 <code>runtimeClassName: windows-2019</code></li></ol><p>例如 : </p><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code>class="language-yaml" data-lang="yaml"><span style="color:green;font-weight:700">apiVersion</span>:<span style="color:#bbb"> </span>apps/v1<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">kind</span>:<span style="color:#bbb"> </span>Deployment<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">metadata</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">name</span>:<span style="color:#bbb"> </span>iis-2019<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">labels</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">app</span>:<span style="color:#bbb"> </span>iis-2019<span style="color:#bbb"> </span></code></pre></div>
```

spec: replicas: 1 template: metadata: name: iis-2019 labels: app: iis-2019 spec: runtimeClassName: windows-2019 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:


```
style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">matchLabels</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">app</span>:<span style="color:#bbb"> </span>iis-2019<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:#00f;font-weight:700">---</span><span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">apiVersion</span>:<span style="color:#bbb"> </span>v1<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">kind</span>:<span style="color:#bbb"> </span>Service<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">metadata</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">name</span>:<span style="color:#bbb"> </span>iis<span style="color:#bbb"> </span><span style="color:#bbb"/><span style="color:green;font-weight:700">spec</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">type</span>:<span style="color:#bbb"> </span>LoadBalancer<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">ports</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">protocol</span>:<span style="color:#bbb"> </span>TCP<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">port</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:#666">80</span><span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">selector</span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </span><span style="color:green;font-weight:700">app</span>:<span style="color:#bbb"> </span>iis-2019<span style="color:#bbb"> </span></span></code></pre><div id="pre-footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您有帮助？</p><button class="btn btn-primary mb-4 feedback--yes">是</button> <button class="btn btn-primary mb-4 feedback--no">否</button><p class="feedback-response feedback--response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 <a target=" blank" rel="noopener" href="https://stackoverflow.com/questions/tagged/kubernetes">Stack Overflow</a>。在 GitHub 仓库上登记新的问题 <a class="feedback--link" target=" blank" rel="noopener" href="https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io">报告问题</a> 或者 <a class="feedback--link" target=" blank" rel="noopener" href="https://github.com/kubernetes/website/issues/new?title=Improvement%20for%20k8s.io">提出改进建议</a>.</p></div><script>const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=&gt;{link.href=link.href+window.location.pathname;});const sendFeedback=(value)=&gt;{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><div
class="text-muted mt-5 pt-3 border-top">最后修改 December 31, 2020 at
11:21 AM PST: <a href="https://github.com/kubernetes/website/commit/
21d86f2aa8abd86a9f4b369e164bd94e748079bf">Incorrect apiVersion
(21d86f2aa)</a></div></div><div class="td-content"><h1>最佳实践</
h1><div class="section-index"><hr class="panel-line"/><div
class="entry"><h5><a href="/zh/docs/setup/best-practices/multiple-
zones/">运行于多区环境</a></h5><p/></div><div class="entry"><h5><a
href="/zh/docs/setup/best-practices/cluster-large/">创建大型集群</a></
h5><p/></div><div class="entry"><h5><a href="/zh/docs/setup/best-
practices/node-conformance/">校验节点设置</a></h5><p/></div><div
class="entry"><h5><a href="/zh/docs/setup/best-practices/
certificates/">PKI 证书和要求</a></h5><p/></div></div></div><div
class="td-content"><h1>运行于多区环境</h1><p>本页描述如何在多个区
( Zone ) 中运行集群。</p><h2 id="#x4ECB;&#x7ECD;">介绍</
h2><p>Kubernetes 1.2 添加了跨多个失效区 ( Failure Zone ) 运行同一集群的能力
( GCE 把它们称作"区 ( Zones ) " , AWS 把它们称作"可用区 ( Availability
Zones ) " , 这里我们用"区 ( Zones ) "指代它们)。此能力是更广泛的集群联邦
( Cluster Federation ) 特性的一个轻量级版本。集群联邦之前有一个昵称 <a
href="https://github.com/kubernetes/community/blob/master/contributors/
design-proposals/multicluster/federation.md">"Ubernetes"</a>)。完全的集
群联邦可以将运行在多个区域 ( Region ) 或云供应商 ( 或本地数据中心 ) 的多个
Kubernetes 集群组合起来。不过,很多用户仅仅是希望在同一云厂商平台的多个区域
运行一个可用性更好的集群,而这恰恰是 1.2 引入的多区支持所带来的特性 ( 此特性之
前有一个昵称 "Ubernetes Lite" )。</p><p>多区支持有意实现的有局限性:可以在
跨多个区域运行同一 Kubernetes 集群,但只能 在同一区域 ( Region ) 和云厂商平
台。目前仅自动支持 GCE 和 AWS,尽管为其他云平台 或裸金属平台添加支持页相对容
易,只需要确保节点和卷上添加合适的标签即可。</p><h2
id="#x529F;&#x80FD;">功能</h2><p>节点启动时, <code>kubelet</
code> 自动向其上添加区信息标签。</p><p>在单区 ( Single-Zone ) 集群中,
Kubernetes 会自动将副本控制器或服务中的 Pod 分布到不同节点,以降低节点失效的
影响。在多区集群中,这一分布负载的行为被扩展到跨区分布,以降低区失效的影响,
跨区分布的能力是通过 <code>SelectorSpreadPriority</code> 实现的。此放置策
略亦仅仅是 尽力而为,所以如果你的集群所跨区是异质的 ( 例如,节点个数不同、节点
类型 不同或者 Pod 资源需求不同 ), 放置策略都可能无法完美地跨区完成 Pod 的 均衡
分布。如果需要,你可以使用同质区 ( 节点个数和类型相同 ) 以降低不均衡 分布的可
能性。</p><p>持久卷被创建时, <code>PersistentVolumeLabel</code> 准入控
制器会自动为其添加区标签。调度器使用 <code>VolumeZonePredicate</code>
断言确保申领某给定卷的 Pod 只会被放到 该卷所在的区。这是因为卷不可以跨区挂载。
</p><h2 id="#x5C40;&#x9650;&#x6027;">局限性</h2><p>多区支持有一
些很重要的局限性:</p><ul><li>我们假定不同的区之间在网络上彼此距离很近,所
以我们不执行可感知区的路由。尤其是,即使某些负责提供该服务的 Pod 与客户端位于
```

同一区，通过服务末端 进入的流量可能会跨区，因而会导致一些额外的延迟和开销。

- 卷与区之间的亲和性仅适用于 PV 持久卷。例如，如果你直接在 Pod 规约中指定某 EBS 卷，这种亲和性支持就无法工作。
- 集群无法跨多个云平台或者地理区域运行。这类功能需要完整的联邦特性支持。

尽管你的节点位于多个区中，`kube-up` 脚本目前默认只能构造一个主控节点。尽管服务是高可用的，能够忍受失去某个区的问题，控制面位于某一个区中。希望运行高可用控制面的用户应该遵照 [高可用性](/zh/docs/setup/production-environment/tools/kubeadm/high-availability/) 中的指令构建。

卷局限性

以下局限性通过 [拓扑感知的卷绑定](/zh/docs/concepts/storage/storage-classes/#volume-binding-mode) 解决：

- 使用动态卷供应时，StatefulSet 卷的跨区分布目前与 Pod 亲和性和反亲和性策略不兼容。
- 如果 StatefulSet 的名字中包含连字符 ("-")，卷的跨区分布可能无法实现存储的 跨区同一分布。
- 当在一个 Deployment 或 Pod 规约中指定多个 PVC 申领时，则需要为某特定区域 配置 StorageClass，或者在某一特定区域中需要静态供应 PV 卷。另一种解决方案是使用 StatefulSet，确保给定副本的所有卷都从同一区中供应。

演练

我们现在准备对在 GCE 和 AWS 上配置和使用多区集群进行演练。为了完成此演练，你需要设置 `MULTIZONE=true` 来启动一个完整的集群，之后指定 `KUBE_USE_EXISTING_MASTER=true` 并再次运行 `kube-up` 添加其他区中的节点。

建立集群

和往常一样创建集群，不过需要设置 MULTIZONE，以便告诉集群需要管理多个区。这里我们在 `us-central1-a` 创建节点。

GCE:

```
curl -sS https://get.k8s.io | MULTIZONE=true KUBERNETES_PROVIDER=gce KUBE_GCE_ZONE=us-central1-a NUM_NODES=3 bash
```

AWS:

```
curl -sS https://get.k8s.io | MULTIZONE=true KUBERNETES_PROVIDER=aws KUBE_AWS_ZONE=us-west-2a NUM_NODES=3 bash
```

这一步骤和往常一样启动一个集群，不过尽管 `MULTIZONE=true` 标志已经启用了多区功能特性支持，集群仍然运行在一个区内。

节点已被打标签

查看节点，你会看到节点上已经有了区信息标签。目前这些节点都在 `us-central1-a` (GCE) 或 `us-west-2a` (AWS)。对于区域 (Region)，标签为 `topology.kubernetes.io/region`，对于区 (Zone)，标签为 `topology.kubernetes.io/zone`：

```
kubectl get nodes --show-labels
```

输出类似于：

```
NAME STATUS ROLES AGE VERSION LABELS
kubernetes-master Ready,SchedulingDisabled <none> 6m v1.13.0 beta.kubernetes.io/instance-type=n1-standard-1,topology.kubernetes.io/region=us-central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/hostname=kubernetes-master
kubernetes-minion-87j9 Ready <none> 6m v1.13.0 beta.kubernetes.io/instance-type=n1-standard-2,topology.kubernetes.io/region=us-central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/hostname=kubernetes-minion-87j9
kubernetes-minion-9vly Ready <none> 6m v1.13.0 beta.kubernetes.io/instance-type=n1-standard-2,topology.kubernetes.io/region=us-central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/hostname=kubernetes-minion-9vly
kubernetes-minion-a12q Ready <none> 6m v1.13.0 beta.kubernetes.io/instance-type=n1-standard-2,topology.kubernetes.io/region=us-central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/hostname=kubernetes-minion-a12q
```

添加第二个区中的节点

让我们向现有集群中添加另外一组节点，复用现有的主控节点，但运行在不同的区 (`us-central1-b` 或 `us-west-2b`)。我们再次运行 `kube-up`，不过设置 `KUBE_USE_EXISTING_MASTER=true`。 `kube-up` 不会创建新的主控节点，而会复用之前创建的主控节点。

GCE:

```
KUBE_USE_EXISTING_MASTER=true
MULTIZONE=true
KUBERNETES_PROVIDER=gce
KUBE_GCE_ZONE=us-central1-b
NUM_NODES=3
```

kubernetes/cluster/kube-up.sh

在 AWS 上，我们还需要为额外的子网指定网络 CIDR，以及主控节点的内部 IP 地址：

让我们向现有集群中添加另外一组节点，复用现有的主控节点，但运行在不同的区 (`us-central1-b` 或 `us-west-2b`)。我们再次运行 `kube-up`，不过设置 `KUBE_USE_EXISTING_MASTER=true`。 `kube-up` 不会创建新的主控节点，而会复用之前创建的主控节点。

AWS:

```
KUBE_USE_EXISTING_MASTER=true
MULTIZONE=true
KUBERNETES_PROVIDER=aws
KUBE_AWS_ZONE=us-central1-b
NUM_NODES=3
```

kubernetes/cluster/kube-up.sh

```
style="color:#666">=</span><span style="color:#a2f">>true</span>
<span style="color:#b8860b">MULTIZONE</span><span
style="color:#666">=</span><span style="color:#a2f">>true</span>
<span style="color:#b8860b">KUBERNETES PROVIDER</span><span
style="color:#666">=</span>aws <span
style="color:#b8860b">KUBE AWS ZONE</span><span
style="color:#666">=</span>us-west-2b <span
style="color:#b8860b">NUM NODES</span><span
style="color:#666">=</span><span style="color:#666">3</span> <span
style="color:#b8860b">KUBE SUBNET CIDR</span><span
style="color:#666">=</span>172.20.1.0/24 <span
style="color:#b8860b">MASTER INTERNAL IP</span><span
style="color:#666">=</span>172.20.0.9 kubernetes/cluster/kube-up.sh </
code></pre></div><p>再次查看节点，你会看到新启动了三个节点并且其标签表明
运行在 <code>us-central1-b</code> 区：</p><div class="highlight"><pre
style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:
4"><code class="language-shell" data-lang="shell">kubectl get nodes --
show-labels </code></pre></div><p>输出类似于：</
p><pre><code>NAME STATUS ROLES AGE VERSION LABELS
kubernetes-master Ready,SchedulingDisabled &lt;none> 16m v1.13.0
beta.kubernetes.io/instance-type=n1-standard-1,topology.kubernetes.io/
region=us-central1,topology.kubernetes.io/zone=us-central1-
a,kubernetes.io/hostname=kubernetes-master kubernetes-minion-281d
Ready &lt;none> 2m v1.13.0 beta.kubernetes.io/instance-type=n1-
standard-2,topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-b,kubernetes.io/
hostname=kubernetes-minion-281d kubernetes-minion-87j9 Ready
&lt;none> 16m v1.13.0 beta.kubernetes.io/instance-type=n1-
standard-2,topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/
hostname=kubernetes-minion-87j9 kubernetes-minion-9v1v Ready
&lt;none> 16m v1.13.0 beta.kubernetes.io/instance-type=n1-
standard-2,topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/
hostname=kubernetes-minion-9v1v kubernetes-minion-a12q Ready
&lt;none> 17m v1.13.0 beta.kubernetes.io/instance-type=n1-
standard-2,topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/
hostname=kubernetes-minion-a12q kubernetes-minion-pp2f Ready
&lt;none> 2m v1.13.0 beta.kubernetes.io/instance-type=n1-
standard-2,topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-b,kubernetes.io/
hostname=kubernetes-minion-pp2f kubernetes-minion-wf8i Ready
&lt;none> 2m v1.13.0 beta.kubernetes.io/instance-type=n1-
standard-2,topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-b,kubernetes.io/
hostname=kubernetes-minion-wf8i </code></pre><h3
id="&#x5377;&#x4EB2;&#x548C;&#x6027;">卷亲和性</h3><p>通过动态卷
供应创建一个卷（只有 PV 持久卷支持区亲和性）：</p><div
class="highlight"><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 - &&EOF
{
  apiVersion: "v1",
  kind: "PersistentVolumeClaim",
  metadata: {
    name: "claim1",
    annotations: {
      volume.alpha.kubernetes.io/storage-class: "foo"
    },
    spec: {
      accessModes: [
        "ReadWriteOnce"
      ],
      resources: {
        requests: {
          storage: "5Gi"
        }
      }
    }
  }
}
EOF
```

说明：
Kubernetes 1.3 及以上版本会将动态 PV 申领散布到所配置的各个区。在 1.2 版本中，动态持久卷总是在集群主控节点所在的区（这里的 `us-central1-a` 或 `us-west-2a`），对应的 Issue ([#23330](https://github.com/kubernetes/kubernetes/issues/23330)) 在 1.3 及以上版本中已经解决。

现在我们来验证 Kubernetes 自动为 PV 打上了所在区或区域的标签：

```
kubectl get pv --show-labels
```

输出类似于：

```
NAME
CAPACITY ACCESSMODES RECLAIM POLICY STATUS CLAIM
STORAGECLASS REASON AGE LABELS pv-gce-mj4gm 5Gi RWO Retain
Bound default/claim1 manual 46s topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-a
```

现在我们将创建一个使用 PVC 申领的 Pod。由于 GCE PD 或 AWS EBS 卷都不能跨区挂载，这意味着 Pod 只能创建在卷所在的区：

```
kubectl apply -f -
&&EOF
```

```
apiVersion:
v1
kind:
Pod
metadata:
  name: mypod
spec:
  containers:
```

```

weight:700">image</span>:<span style="color:#bbb"> </
span>nginx<span style="color:#bbb"> </span><span style="color:#bbb">
</span><span style="color:green;font-weight:700">volumeMounts</
span>:<span style="color:#bbb"> </span><span style="color:#bbb"> </
span>- <span style="color:green;font-weight:700">mountPath</
span>:<span style="color:#bbb"> </span><span style="color:#b44">"/
var/www/html"</span><span style="color:#bbb"> </span><span
style="color:#bbb"> </span><span style="color:green;font-weight:
700">name</span>:<span style="color:#bbb"> </span>mypd<span
style="color:#bbb"> </span><span style="color:#bbb"> </span><span
style="color:green;font-weight:700">volumes</span>:<span
style="color:#bbb"> </span><span style="color:#bbb"> </span>- <span
style="color:green;font-weight:700">name</span>:<span
style="color:#bbb"> </span>mypd<span style="color:#bbb"> </
span><span style="color:#bbb"> </span><span style="color:green;font-
weight:700">persistentVolumeClaim</span>:<span style="color:#bbb"> </
span><span style="color:#bbb"> </span><span style="color:green;font-
weight:700">claimName</span>:<span style="color:#bbb"> </
span>claim1<span style="color:#bbb"> </span><span style="color:#bbb"/
>EOF<span style="color:#bbb"> </span></code></pre></div><p>注意
Pod 自动创建在卷所在的区，因为云平台提供商一般不允许跨区挂载存储卷。</p><div
class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-
tab-size:4;tab-size:4"><code class="language-shell" data-
lang="shell">kubectl describe pod mypod | grep Node </code></pre></
div><pre><code>Node: kubernetes-minion-9vlv/10.240.0.5 </code></
pre><p>检查节点标签：</p><div class="highlight"><pre
style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:
4"><code class="language-shell" data-lang="shell">kubectl get node
kubernetes-minion-9vlv --show-labels </code></pre></
div><pre><code>NAME STATUS AGE VERSION LABELS kubernetes-
minion-9vlv Ready 22m v1.6.0+fff5156 beta.kubernetes.io/instance-type=n1-
standard-2,topology.kubernetes.io/region=us-
central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/
hostname=kubernetes-minion-9vlv </code></pre><h3 id="pod-
&#x8DE8;&#x533A;&#x5206;&#x5E03;">Pod 跨区分布</h3><p>同一副本
控制器或服务的多个 Pod 会自动完成跨区分布。首先，我们现在第三个区启动一些节
点：</p><p>GCE:</p><div class="highlight"><pre style="background-
color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code
class="language-shell" data-lang="shell"><span
style="color:#b8860b">KUBE_USE_EXISTING_MASTER</span><span
style="color:#666">=</span><span style="color:#a2f">true</span>
<span style="color:#b8860b">MULTIZONE</span><span
style="color:#666">=</span><span style="color:#a2f">true</span>
<span style="color:#b8860b">KUBERNETES_PROVIDER</span><span
style="color:#666">=</span>gce <span
style="color:#b8860b">KUBE_GCE_ZONE</span><span
style="color:#666">=</span>us-central1-f <span
style="color:#b8860b">NUM_NODES</span><span
style="color:#666">=</span><span style="color:#666">3</span>
kubernetes/cluster/kube-up.sh </code></pre></div><p>AWS:</p><div
class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-

```



```
tab-size:4;tab-size:4"><code class="language-shell" data-  
lang="shell"><span  
style="color:#b8860b">KUBE USE EXISTING MASTER</span><span  
style="color:#666">=</span><span style="color:#a2f">>true</span>  
<span style="color:#b8860b">MULTIZONE</span><span  
style="color:#666">=</span><span style="color:#a2f">>true</span>  
<span style="color:#b8860b">KUBERNETES PROVIDER</span><span  
style="color:#666">=</span>aws <span  
style="color:#b8860b">KUBE AWS ZONE</span><span  
style="color:#666">=</span>us-west-2c <span  
style="color:#b8860b">NUM NODES</span><span  
style="color:#666">=</span><span style="color:#666">3</span> <span  
style="color:#b8860b">KUBE SUBNET CIDR</span><span  
style="color:#666">=</span>172.20.2.0/24 <span  
style="color:#b8860b">MASTER INTERNAL IP</span><span  
style="color:#666">=</span>172.20.0.9 kubernetes/cluster/kube-up.sh </  
code></pre></div><p>验证你现在有来自三个区的节点：</p><div  
class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-  
tab-size:4;tab-size:4"><code class="language-shell" data-  
lang="shell">kubectl get nodes --show-labels </code></pre></div><p>创  
建 <code>guestbook-go</code> 示例，其中包含副本个数为 3 的 RC，运行一个简  
单的 Web 应用：</p><div class="highlight"><pre style="background-  
color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code  
class="language-shell" data-lang="shell">find kubernetes/examples/  
guestbook-go/ -name <span style="color:#b44">'*.json'</span> | xargs -l  
<span style="color:#666">{}</span> kubectl apply -f <span  
style="color:#666">{}</span> </code></pre></div><p>Pod 应该跨三个区  
分布：</p><div class="highlight"><pre style="background-color:#f8f8f8;-  
moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-  
lang="shell">kubectl describe pod -l <span style="color:#b8860b">app</  
span><span style="color:#666">=</span>guestbook | grep Node </  
code></pre></div><pre><code>Node: kubernetes-minion-9vlv/10.240.0.5  
Node: kubernetes-minion-281d/10.240.0.8 Node: kubernetes-minion-olsh/  
10.240.0.11 </code></pre><div class="highlight"><pre  
style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:  
4"><code class="language-shell" data-lang="shell">kubectl get node  
kubernetes-minion-9vlv kubernetes-minion-281d kubernetes-minion-olsh --  
show-labels </code></pre></div><pre><code>NAME STATUS ROLES  
AGE VERSION LABELS kubernetes-minion-9vlv Ready &lt;none>; 34m  
v1.13.0 beta.kubernetes.io/instance-type=n1-  
standard-2,topology.kubernetes.io/region=us-  
central1,topology.kubernetes.io/zone=us-central1-a,kubernetes.io/  
hostname=kubernetes-minion-9vlv kubernetes-minion-281d Ready  
&lt;none>; 20m v1.13.0 beta.kubernetes.io/instance-type=n1-  
standard-2,topology.kubernetes.io/region=us-  
central1,topology.kubernetes.io/zone=us-central1-b,kubernetes.io/  
hostname=kubernetes-minion-281d kubernetes-minion-olsh Ready  
&lt;none>; 3m v1.13.0 beta.kubernetes.io/instance-type=n1-  
standard-2,topology.kubernetes.io/region=us-  
central1,topology.kubernetes.io/zone=us-central1-f,kubernetes.io/  
hostname=kubernetes-minion-olsh </code></pre><p>负载均衡器也会跨集群
```

中的所有区；`guestbook-go` 示例中包含了一个负载均衡 服务的例子：

```
<div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">kubectl describe service guestbook | grep LoadBalancer.Ingress </code></pre></div><p>输出类似于：</pre><pre><code>LoadBalancer Ingress: 130.211.126.21 </code></pre><p>设置上面的 IP 地址：</pre><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell"><span style="color:#a2f">export</span> <span style="color:#b8860b">IP</span><span><span style="color:#666">=</span>130.211.126.21 </code></pre></div><p>使用 curl 访问该 IP：</pre><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell">curl -s http://<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">IP</span><span style="color:#b68;font-weight:700">}</span>:3000/env | grep HOSTNAME </code></pre></div><p>输出类似于：</pre><pre><code> "HOSTNAME": "guestbook-44sep", </code></pre><p>如果多次尝试该命令：</pre><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell"><span style="color:#666">(</span><span style="color:#a2f;font-weight:700">for</span> i in <span style="color:#b44">`</span>seq 20<span style="color:#b44">`</span>; <span style="color:#a2f;font-weight:700">do</span> curl -s http://<span style="color:#b68;font-weight:700">${</span><span style="color:#b8860b">IP</span><span style="color:#b68;font-weight:700">}</span>:3000/env | grep HOSTNAME; <span style="color:#a2f;font-weight:700">done</span><span><span style="color:#666">)</span> | sort | uniq </code></pre></div><p>输出类似于：</pre><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell"><span style="color:#b44">"HOSTNAME"</span>: <span style="color:#b44">"guestbook-44sep"</span>, <span style="color:#b44">"HOSTNAME"</span>: <span style="color:#b44">"guestbook-hum5n"</span>, <span style="color:#b44">"HOSTNAME"</span>: <span style="color:#b44">"guestbook-ppm40"</span>, </code></pre></div><p>负载均衡器正确地选择不同的 Pod，即使它们跨了多个区。</pre><h3 id="&#x505C;&#x6B62;&#x96C6;&#x7FA4;">停止集群</h3><p>当完成以上工作之后，清理任务现场：</pre><p>GCE:</pre><div class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:4"><code class="language-shell" data-lang="shell"><span style="color:#b8860b">KUBERNETES PROVIDER</span><span style="color:#666">=</span>gce <span style="color:#b8860b">KUBE USE EXISTING MASTER</span><span style="color:#666">=</span><span style="color:#a2f">true</span><span style="color:#b8860b">KUBE GCE ZONE</span><span style="color:#666">=</span>us-central1-f kubernetes/cluster/kube-down.sh <span style="color:#b8860b">KUBERNETES PROVIDER</span><span style="color:#666">=</span>gce <span></pre></div>
```

```
style="color:#b8860b">KUBE USE EXISTING MASTER</span><span
style="color:#666">=</span><span style="color:#a2f">>true</span>
<span style="color:#b8860b">KUBE GCE ZONE</span><span
style="color:#666">=</span>us-central1-b kubernetes/cluster/kube-
down.sh <span style="color:#b8860b">KUBERNETES PROVIDER</
span><span style="color:#666">=</span>gce <span
style="color:#b8860b">KUBE GCE ZONE</span><span
style="color:#666">=</span>us-central1-a kubernetes/cluster/kube-
down.sh </code></pre></div><p>AWS:</p><div class="highlight"><pre
style="background-color:#f8f8f8;-moz-tab-size:4;-o-tab-size:4;tab-size:
4"><code class="language-shell" data-lang="shell"><span
style="color:#b8860b">KUBERNETES PROVIDER</span><span
style="color:#666">=</span>aws <span
style="color:#b8860b">KUBE USE EXISTING MASTER</span><span
style="color:#666">=</span><span style="color:#a2f">>true</span>
<span style="color:#b8860b">KUBE AWS ZONE</span><span
style="color:#666">=</span>us-west-2c kubernetes/cluster/kube-down.sh
<span style="color:#b8860b">KUBERNETES PROVIDER</span><span
style="color:#666">=</span>aws <span
style="color:#b8860b">KUBE USE EXISTING MASTER</span><span
style="color:#666">=</span><span style="color:#a2f">>true</span>
<span style="color:#b8860b">KUBE AWS ZONE</span><span
style="color:#666">=</span>us-west-2b kubernetes/cluster/kube-down.sh
<span style="color:#b8860b">KUBERNETES PROVIDER</span><span
style="color:#666">=</span>aws <span
style="color:#b8860b">KUBE AWS ZONE</span><span
style="color:#666">=</span>us-west-2a kubernetes/cluster/kube-down.sh
</code></pre></div><div id="pre-footer"><h2>反馈</h2><p
class="feedback-prompt">此页是否对您有帮助? </p><button class="btn btn-
primary mb-4 feedback--yes">是</button> <button class="btn btn-primary
mb-4 feedback--no">否</button><p class="feedback-response feedback--
response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、
需要答案的问题，可以访问 <a target=" blank" rel="noopener" href="https://
stackoverflow.com/questions/tagged/kubernetes">Stack Overflow</a>。在
GitHub 仓库上登记新的问题 <a class="feedback--link" target=" blank"
rel="noopener" href="https://github.com/kubernetes/website/issues/new?
title=Issue%20with%20k8s.io">报告问题</a> 或者 <a class="feedback--link"
target=" blank" rel="noopener" href="https://github.com/kubernetes/
website/issues/new?title=Improvement%20for%20k8s.io">提出改进建议</
a>.</p></div><script>const yes=document.querySelector('.feedback--
yes');const no=document.querySelector('.feedback--
no');document.querySelectorAll('.feedback--link').forEach(link=&gt;{
{link.href=link.href+window.location.pathname;});const
sendFeedback=(value)=&gt;{if(!gtag){console.log('!gtag');}
gtag('event','click',
{'event category':'Helpful','event label':window.location.pathname,value});});const
disableButtons=()=&gt;{yes.disabled=true;yes.classList.add('feedback--
button disabled');no.disabled=true;no.classList.add('feedback--
button disabled');};yes.addEventListener('click',)=&gt;{
{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><div
class="text-muted mt-5 pt-3 border-top">最后修改 November 04, 2020 at
9:06 AM PST: <a href="https://github.com/kubernetes/website/commit/
7e7423a474c56f4587fbac42c9bc4a066fce14ea">Make trivial non-en
changes for KEP 1659 (7e7423a47)</div></div><div class="td-
content"><h1>创建大型集群</h1><h2 id="#支持">支持</
h2><p>在 v1.20 版本中，Kubernetes 支持的最大节点数为 5000。更具体地说，我
们支持满足以下所有条件的配置：</p>节点数不超过
5000Pod 总数不超过 150000容器总数不超过 300000</
li>每个节点的 pod 数量不超过 100
<nav
id="TableOfContents">支持</
a>设定配额问题</
li>Etcd 存储主
控节点大小和主控组件插件资源
允
许启动时次要节点失败</nav><h2
id="#设定">设定</h2><p>集群是一组运行着 Kubernetes 代理
的节点（物理机或者虚拟机），这些节点由主控节点（集群级控制面）控制。</p><p>
通常，集群中的节点数由特定于云平台的配置文件 <code>config-default.sh</
code>（可以参考 <a href="https://releases.k8s.io/master/cluster/gce/config-
default.sh">GCE 平台的 <code>config-default.sh</code>）中的
<code>NUM NODES</code> 参数控制。</p><p>但是，在许多云供应商的平台
上，仅将该值更改为非常大的值，可能会导致安装脚本运行失败。例如，在 GCE，由于
配额问题，集群会启动失败。</p><p>因此，在创建大型 Kubernetes 集群时，必须
考虑以下问题。</p><h3 id="#配额问题">配额
问题</h3><p>为了避免遇到云供应商配额问题，在创建具有大规模节点的集群时，请
考虑：</p>增加诸如 CPU，IP 等资源的配额。例如，在 GCE，
您需要增加以下资源的配额：CPUsVM 实例永久磁盘
总量使用中的 IP 地址防火墙规则转发规则
路由目标池由于某些云供应商会对虚拟机
的创建进行流控，因此需要对设置脚本进行更改，使其以较小的批次启动新的节点，并且
之间有等待时间。<h3 id="etcd-存储">Etcd 存储</
h3><p>为了提高大规模集群的性能，我们将事件存储在专用的 etcd 实例中。</
p><p>在创建集群时，现有 salt 脚本可以：</p>启动并配置其它 etcd 实
例配置 API 服务器以使用 etcd 存储事件<h3
id="#主控节点大小和主">主
控节点大小和主控组件</h3><p>在 GCE/Google Kubernetes Engine 和 AWS
上，<code>kube-up</code> 会根据节点数量自动为您集群中的 master 节点配置适
当的虚拟机大小。在其它云供应商的平台上，您将需要手动配置它。作为参考，我们在
GCE 上使用的规格为：</p>1-5 个节点：n1-standard-16-10
个节点：n1-standard-211-100 个节点：n1-standard-4</
li>101-250 个节点：n1-standard-8251-500 个节点：n1-
standard-16超过 500 节点：n1-standard-32<p>在 AWS
上使用的规格为</p>1-5 个节点：m3.medium6-10 个节点：

m3.large

- 11-100 个节点：m3.xlarge
- 101-250 个节点：m3.2xlarge
- 251-500 个节点：c4.4xlarge
- 超过 500 节点：c4.8xlarge

说明：

在 Google Kubernetes Engine 上，主控节点的大小会根据集群的大小自动调整。更多有关信息，请参阅 <https://cloudplatform.googleblog.com/2017/11/Cutting-Cluster-Management-Fees-on-Google-Kubernetes-Engine.html> 此博客文章。

在 AWS 上，主控节点的规格是在集群启动时设置的，并且，即使以后通过手动删除或添加节点的方式使集群扩容或扩容，主控节点的大小也不会更改。

插件资源

为了防止内存泄漏或 [集群插件](https://releases.k8s.io/master/cluster/addons) 中的其它资源问题导致节点上所有可用资源被消耗，Kubernetes 限制了插件容器可以消耗的 CPU 和内存资源（请参阅 PR [#10653](http://pr.k8s.io/10653/files) 和 [#10778](http://pr.k8s.io/10778/files)）。例如：

```
containers:
- name:
  fluentd-cloud-logging
  image:
  k8s.gcr.io/fluentd-gcp:1.16
  resources:
  limits:
  cpu:
  100m
  memory:
  200Mi
```

本数量有助于处理增加的负载，但是，由于每个副本的负载也略有增加，因此也请考虑增加 CPU/内存限制）：

- [elasticsearch](https://releases.k8s.io/master/cluster/addons/fluentd-elasticsearch/es-statefulset.yaml)

- 根据集群的规模，如果使用了以下插件，限制其内存和 CPU 上限（这些插件在每个节点上都有一个副本，但是 CPU/内存使用量也会随集群负载/规模而略有增加）：
 - [FluentD 和 ElasticSearch 插件](https://releases.k8s.io/master/cluster/addons/fluentd-elasticsearch/fluentd-es-ds.yaml)
 - [FluentD 和 GCP 插件](https://releases.k8s.io/master/cluster/addons/fluentd-gcp/fluentd-gcp-ds.yaml)

Heapster 的资源限制与您集群的初始大小有关（请参阅 [#16185](https://issue.k8s.io/16185) 和 [#22940](http://issue.k8s.io/22940)）。如果您发现 Heapster 资源不足，您应该调整堆内存请求的计算公式（有关详细信息，请参阅相关 PR）。</p><p>关于如何检测插件容器是否达到资源限制，参见 [计算资源的故障排除](/zh/docs/concepts/configuration/manage-resources-containers/#troubleshooting) 部分。</p><p>未来，我们期望根据集群规模大小来设置所有群集附加资源限制，并在集群扩缩容时动态调整它们。我们欢迎您来实现这些功能。</p><h3

id="#x5141;许启动时次要节&

允许启动时次要节点失败</h3><p>出于各种原因（更多详细信息，请参见 [#18969](https://github.com/kubernetes/kubernetes/issues/18969)），在 `kube-up.sh` 中设置很大的 `NUM NODES` 时，可能会由于少数节点无法正常启动而失败。此时，您有两个选择：重新启动集群（运行 `kube-down.sh`，然后再运行 `kube-up.sh`），或者在运行 `kube-up.sh` 之前将环境变量 `ALLOWED NOTREADY NODES` 设置为您认为合适的任何值。采取后者时，即使运行成功的节点数量少于 `NUM NODES`，`kube-up.sh` 仍可以运行成功。根据失败的原因，这些节点可能会稍后加入集群，又或者群集的大小保持在 `NUM NODES-ALLOWED NOTREADY NODES`。</p><div id="pre-footer"><h2>

反馈</h2><p class="feedback--prompt">此页是否对您有帮助？</p><button class="btn btn-primary mb-4 feedback--yes">是</button> <button class="btn btn-primary mb-4 feedback--no">否</button><p class="feedback--response feedback--response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问 [Stack Overflow](https://stackoverflow.com/questions/tagged/kubernetes)。在 GitHub 仓库上登记新的问题 [报告问题](https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io) 或者 [提出改进建议](https://github.com/kubernetes/website/issues/new?title=Improvement%20for%20k8s.io)。</p></div><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--

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',()=>&gt;
{sendFeedback(1);disableButtons();document.querySelector('.feedback--
response').classList.remove('feedback--
response hidden');});no.addEventListener('click',()=>&gt;
{sendFeedback(0);disableButtons();document.querySelector('.feedback--
response').classList.remove('feedback-response hidden');});</script><div
class="text-muted mt-5 pt-3 border-top">最后修改 September 07, 2020 at
4:32 PM PST: <a href="https://github.com/kubernetes/website/commit/
73415d973b747187b3809ebd12c07ea918f00a89">[zh] Fix links in setup
section (2) (73415d973)</a></div></div><div class="td-content"><h1>校
验节点设置</h1><nav id="TableOfContents"><ul><li><a
href="#&#x8282;&#x70B9;&#x4E00;&#x81F4;&#x6027;&#x6D4B;&#x8BD5;">
节点一致性测试</a></li><li><a
href="#&#x8282;&#x70B9;&#x7684;&#x524D;&#x63D0;&#x6761;&#x4EF6;">
节点的前提条件</a></li><li><a
href="#&#x8FD0;&#x884C;&#x8282;&#x70B9;&#x4E00;&#x81F4;&#x6027;&#x6D4B;&#x8BD5;">
运行节点一致性测试</a></li><li><a
href="#&#x9488;&#x5BF9;&#x5176;&#x4ED6;&#x786C;&#x4EF6;&#x4F53;&#x70B9;">
针对其他硬件体系结构运行节点一致性测试</a></li><li><a
href="#&#x8FD0;&#x884C;&#x7279;&#x5B9A;&#x7684;&#x6D4B;&#x8BD5;">
运行特定的测试</a></li><li><a href="#&#x6CE8;&#x610F;">注意</a></
li></ul></nav><h2
id="#&#x8282;&#x70B9;&#x4E00;&#x81F4;&#x6027;&#x6D4B;&#x8BD5;">
节点一致性测试</h2><p><em>节点一致性测试</em> 是一个容器化的测试框架，
提供了针对节点的系统验证和功能测试。</p><p>该测试主要检测节点是否满足
Kubernetes 的最低要求，通过检测的节点有资格加入 Kubernetes 集群。</p><h2
id="#&#x8282;&#x70B9;&#x7684;&#x524D;&#x63D0;&#x6761;&#x4EF6;">
节点的前提条件</h2><p>要运行节点一致性测试，节点必须满足与标准 Kubernetes
节点相同的前提条件。节点至少应安装以下守护程序：</p><ul><li>容器运行时
(Docker)</li><li>Kubelet</li></ul><h2
id="#&#x8FD0;&#x884C;&#x8282;&#x70B9;&#x4E00;&#x81F4;&#x6027;&#x6D4B;&#x8BD5;">
运行节点一致性测试</h2><p>要运行节点一致性测试，请执行以下步骤：</
p><ol><li>得出 kubelet 的 <code>--kubeconfig</code> 的值；例如：
<code>--kubeconfig=/var/lib/kubelet/config.yaml</code>。由于测试框架启动了
本地控制平面来测试 kubelet，因此使用 <code>http://localhost:8080</code> 作
为API 服务器的 URL。一些其他的 kubelet 命令行参数可能会被用到：
<ul><li><code>--pod-cidr</code>：如果使用 <code>kubenet</code>，需
要为 Kubelet 任意指定一个 CIDR，例如 <code>--pod-cidr=10.180.0.0/24</
code>。</li><li><code>--cloud-provider</code>：如果使用 <code>--
cloud-provider=gce</code>，需要移除这个参数 来运行测试。</li></ul></li></
ol><ol start="2"><li><p>使用以下命令运行节点一致性测试：</p><div
class="highlight"><pre style="background-color:#f8f8f8;-moz-tab-size:4;-o-
tab-size:4;tab-size:4"><code class="language-shell" data-
lang="shell"><span style="color:#080;font-style:italic"># $CONFIG DIR 是
您 Kubelet 的 pod manifest 路径。</span> <span style="color:#080;font-
style:italic"># $LOG DIR 是测试的输出路径。</span> sudo docker run -it --rm
--privileged --net<span style="color:#666">=</span>host <span
style="color:#b62;font-weight:700">\ </span><span
style="color:#b62;font-weight:700">/> -v /:/rootfs -v <span
style="color:#b8860b">$CONFIG DIR</span>:<span
```


`style="color:#b8860b">$CONFIG DIR -v $LOG DIR:/var/result \ k8s.gcr.io/node-test:0.2 </code></pre></div><h2 id="针对其他硬件体系" >针对其他硬件体系结构运行节点一致性测试</h2><p>Kubernetes 也为其他硬件体系结构的系统提供了节点一致性测试的 Docker 镜像：</p><p><table><thead><tr><th>架构</th><th style="text-align:center">镜像</th><th></th></tr></thead><tbody><tr><td>amd64</td><td style="text-align:center">node-test-amd64</td><td></td></tr><tr><td>arm</td><td style="text-align:center">node-test-arm</td><td></td></tr><tr><td>arm64</td><td style="text-align:center">node-test-arm64</td><td></td></tr></tbody></table><h2 id="运行特定的测试" >运行特定的测试</h2><p>要运行特定测试，请使用您希望运行的测试的特定表达式覆盖环境变量 <code>FOCUS</code>。</p><div class="highlight"><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></div><p>要跳过特定的测试，请使用您希望跳过的测试的常规表达式覆盖环境变量 <code>SKIP</code>。</p><div class="highlight"><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 SKIP=MirrorPod \ # 运行除 MirrorPod 测试外的所有一致性测试内容 k8s.gcr.io/node-test:0.2 </code></pre></div><p>节点一致性测试是节点端到端测试的容器化版本。</p><p>默认情况下，它会运行所有一致性测试。</p><p>理论上，只要合理地配置容器和挂载所需的卷，就可以运行任何的节点端到端测试用例。但是这里强烈建议只运行一致性测试，因为运行非一致性测试需`

要很多复杂的配置。

注意

- 测试会在节点上遗留一些 Docker 镜像，包括节点一致性测试本身的镜像和功能测试相关的镜像。
- 测试会在节点上遗留一些死的容器。这些容器是在功能测试的过程中创建的。

反馈

此页是否对您有帮助？

感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问

[Stack Overflow](https://stackoverflow.com/questions/tagged/kubernetes)。在 GitHub 仓库上登记新的问题

[报告问题](https://github.com/kubernetes/website/issues/new?title=Issue%20with%20k8s.io) 或者 [提出改进建议](https://github.com/kubernetes/website/issues/new?title=Improvement%20for%20k8s.io)

```
const yes=document.querySelector('.feedback--yes');const no=document.querySelector('.feedback--no');document.querySelectorAll('.feedback--link').forEach(link=&gt;{link.href=link.href+window.location.pathname;});const
```

```
sendFeedback=(value)=&gt;{if(!gtag){console.log('!gtag');}gtag('event','click',
```

```
{'event category':'Helpful','event label':window.location.pathname,value});});const disableButtons=()=&gt;{yes.disabled=true;yes.classList.add('feedback--button disabled');no.disabled=true;no.classList.add('feedback--button disabled');};yes.addEventListener('click',()=&gt;
```

```
{sendFeedback(1);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response hidden');});no.addEventListener('click',()=&gt;
```

```
{sendFeedback(0);disableButtons();document.querySelector('.feedback--response').classList.remove('feedback--response hidden');});
```

最后修改 December 06, 2020 at 11:31 PM PST: [\[zh\] Update node-conformance.md \(980d7f124\)](https://github.com/kubernetes/website/commit/980d7f124315124c35cde888cb7ac95fcd254551)

PKI 证书和要求

Kubernetes 需要 PKI 证书才能进行基于 TLS 的身份验证。如果你是使用 [kubeadm](/zh/docs/reference/setup-tools/kubeadm/kubeadm/) 安装的 Kubernetes，则会自动生成集群所需的证书。你还可以生成自己的证书。例如，不将私钥存储在 API 服务器上，可以让私钥更加安全。此页面说明了集群必需的证书。

集群是如何使用证书的

Kubernetes 需要 PKI 才能执行以下操作：

- Kubelet 的客户端证书，用于 API 服务器身份验证
- API 服务器端点的证书
- 集群管理员的客户端证书，用于 API 服务器身份认证
- API 服务器的客户端证书，用于和 Kubelet 的会话
- API 服务器的客户端证书，用于和 etcd 的会话
- 控制器管理器的客户端证书/kubeconfig，用于和 API 服务器的会话
- 调度器的客户端证书/kubeconfig，用于和 API 服务器的会话
- [前端代理](/zh/docs/tasks/extend-kubernetes/configure-aggregation-layer/) 的客户端及服务端证书

说明： 只有当你运行 kube-proxy 并要支持 [前端代理](/zh/docs/tasks/extend-kubernetes/) 的客户端及服务端证书

setup-extension-api-server/">扩展 API 服务器

时，才需要 `front-proxy` 证书

etcd 还实现了双向 TLS 来对客户端和其他对等节点进行身份验证。

证书存放的位置

如果你是通过 kubeadm 安装的 Kubernetes，所有证书都存放在 `/etc/kubernetes/pki` 目录下。本文所有相关的路径都是基于该路径的相对路径。

手动配置证书

如果你不想通过 kubeadm 生成这些必需的证书，你可以通过下面两种方式之一来手动创建它们。

单根 CA

你可以创建一个单根 CA，由管理员控制器它。该根 CA 可以创建多个中间 CA，并将所有进一步的创建委托给 Kubernetes。

需要这些 CA：

路径	默认 CN	描述
ca.crt,key	kubernetes-ca	Kubernetes 通用 CA
etcd/ca.crt,key	etcd-ca	与 etcd 相关的所有功能
front-proxy-ca.crt,key	kubernetes-front-proxy-ca	用于

前端代理

上面的 CA 之外，还需要获取用于服务账户管理的密钥对，也就是 `sa.key` 和 `sa.pub`。

所有的证书

如果你不想将 CA 的私钥拷贝至你的集群中，你也可以自己生成全部的证书。

需要这些证书：

默认 CN	父级 CA	O (位于 Subject 中)	类型	主机 (SAN)
kube-etcd	etcd-ca		server, client	<code>localhost</code> , <code>127.0.0.1</code>
kube-etcd-peer	etcd-ca		server, client	<code>&lt;hostname></code> , <code>&lt;Host IP></code> , <code>localhost</code> , <code>127.0.0.1</code>
kube-etcd-healthcheck-client	etcd-ca		client	
kube-apiserver-etcd-client	etcd-ca			
system:masters	client			
kube-apiserver	kubernetes-ca		server	<code>&lt;hostname></code> , <code>&lt;Host IP></code> , <code>&lt;advertise IP></code> , <code>[1]</code>
kube-apiserver-kubelet-client	kubernetes-ca			
system:masters	client			
front-proxy-client	kubernetes-front-proxy-ca		client	

[1]: 用来连接到集群的不同 IP 或 DNS 名 (就像 `kubeadm` 为负载均衡所使用的固定 IP 或 DNS 名, `kubernetes`、`kubernetes.default`、`kubernetes.default.svc`、`kubernetes.default.svc.cluster`、

`kubernetes.default.svc.cluster.local`)。其中，`kind` 对应一种或多种类型的 `x509` 密钥用途：

kind	密钥用途
------	------

server	数字签名、密钥加密、服务端认证
client	数字签名、密钥加密、客户端认证

说明：

上面列出的 Hosts/SAN 是推荐的配置方式；如果需要特殊安装，则可以在所有服务器证书上添加其他 SAN。

说明：

对于 kubeadm 用户：

- 不使用私钥，将证书复制到集群 CA 的方案，在 kubeadm 文档中将这种方案称为外部 CA。
- 如果将以上列表与 kubeadm 生成的 PKI 进行比较，你会注意到，如果使用外部 etcd，则不会生成 `kube-etcd`、`kube-etcd-peer` 和 `kube-etcd-healthcheck-client` 证书。

证书路径

证书应放置在建议的路径中（以便 [kubeadm](/zh/docs/reference/setup-tools/kubeadm/kubeadm/) 使用）。无论使用什么位置，都应使用给定的参数指定路径。

默认 CN	建议的密钥路径	建议的证书路径	命令	密钥参数	证书参数
etcd-ca	etcd/ca.key	etcd/ca.crt	kube-apiserver	--etcd-cafile	
kube-apiserver-etcd-client	apiserver-etcd-client.key	apiserver-etcd-client.crt	kube-apiserver	--etcd-keyfile	--etcd-certfile
kubernetes-ca	ca.key	ca.crt	kube-apiserver		--client-ca-file
kubernetes-ca	ca.key	ca.crt	kube-controller-manager	--cluster-signing-key-file	--client-ca-file, --root-ca-file, --cluster-signing-cert-file
kube-apiserver	apiserver.key	apiserver.crt	kube-apiserver	--tls-private-key-file	--tls-cert-file
kube-apiserver-kubelet-client	apiserver-kubelet-client.key	apiserver-kubelet-client.crt	kube-apiserver	--kubelet-client-key	--kubelet-client-certificate
front-proxy-ca	front-proxy-ca.key	front-proxy-ca.crt	kube-apiserver		--requestheader-client-ca-file
front-proxy-ca	front-proxy-ca.key	front-proxy-ca.crt	kube-controller-manager		--requestheader-client-ca-file
front-proxy-client	front-proxy-client.key	front-proxy-client.crt	kube-apiserver	--proxy-client-key-file	--proxy-client-cert-file
etcd-ca	etcd/ca.key	etcd/ca.crt	etcd		--trusted-ca-file, --peer-trusted-ca-file
kube-etcd	etcd/server.key	etcd/server.crt	etcd	--key-file	--cert-file
kube-etcd-peer	etcd/peer.key	etcd/peer.crt	etcd	--peer-key-file	--peer-cert-file
etcd-ca	etcd/ca.crt	etcdctl	etcdctl		--cacert
kube-etcd-healthcheck-client	etcd/healthcheck-client.key	etcd/healthcheck-client.crt	etcdctl	--key	--cert

注意事项同样适用于服务帐户密钥对：

私钥路径	公钥路径	命令
------	------	----

th><th>参数</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="#x4E3A;用户帐户配置证">为用户帐户配置证书</h2><p>你必须手动配置以下管理员帐户和服务帐户：</p><p><table><thead><tr><th>文件名</th><th>凭据名称</th><th>默认 CN</th><th>O (位于 Subject 中)</th></tr></thead><tbody><tr><td>admin.conf</td><td>default-admin</td><td><td>kubernetes-admin</td><td>system:masters</td></tr><tr><td>kubelet.conf</td><td>default-auth</td><td><td>system:node:<code><nodeName></code> (参阅注释)</td><td>system:nodes</td></tr><tr><td>controller-manager.conf</td><td>default-controller-manager</td><td>system:kube-controller-manager</td><td></td></tr><tr><td>scheduler.conf</td><td>default-scheduler</td><td>system:kube-scheduler</td><td></td></tr></tbody></table><blockquote class="note callout"><div>说明：<code>kubelet.conf</code> 中 <code><nodeName></code> 的值必须 与 kubelet 向 apiserver 注册时提供的节点名称的值完全匹配。有关更多详细信息，请阅读节点授权。</div></blockquote><p>对于每个配置，请都使用给定的 CN 和 O 生成 x509 证书/密钥偶对。</p><p>为每个配置运行下面的 <code>kubect</code> 命令：</p><div class="highlight"><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></div><p>这些文件用途如下：</p><p><table><thead><tr><th>文件名</th><th>命令</th><th>说明</th></tr></thead><tbody><tr><td>admin.conf</td><td>kubect</td><td>配置集群的管理员</td></tr><tr><td>kubelet.conf</td><td>kubelet</td><td>集群中的每个节点都需要一份</td></tr><tr><td>controller-manager.conf</td><td>kube-controller-manager</td><td>必需添加到<code>manifests/kube-controller-manager.yaml</code> 清单中</td></tr><tr><td>scheduler.conf</td><td>kube-scheduler</td><td>必需添加到<code>manifests/kube-scheduler.yaml</code> 清单中</td></tr></tbody></table><div id="pre-footer"><h2>反馈</h2><p class="feedback-prompt">此页是否对您有帮助？</p><button class="btn btn-primary mb-4 feedback-yes">是</button> <button class="btn btn-primary mb-4 feedback-no">否</button><p class="feedback-response feedback--

response hidden">感谢反馈。如果您有一个关于如何使用 Kubernetes 的特定的、需要答案的问题，可以访问