Edit This Page

Setup

Use this page to find the type of solution that best fits your needs.

Deciding where to run Kubernetes depends on what resources you have available and how much flexibility you need. You can run Kubernetes almost anywhere, from your laptop to VMs on a cloud provider to a rack of bare metal servers. You can also set up a fully-managed cluster by running a single command or craft your own customized cluster on your bare metal servers.

- Local-machine Solutions
- Hosted Solutions
- Turnkey – Cloud Solutions
- Turnkey – On-Premises Solutions
- Custom Solutions
- What's next

Local-machine Solutions

A local-machine solution is an easy way to get started with Kubernetes. You can create and test Kubernetes clusters without worrying about consuming cloud resources and quotas.

You should pick a local solution if you want to:

- Try or start learning about Kubernetes
- Develop and test clusters locally

Pick a local-machine solution.

Hosted Solutions

Hosted solutions are a convenient way to create and maintain Kubernetes clusters. They manage and operate your clusters so you don't have to.

You should pick a hosted solution if you:

- Want a fully-managed solution
- Want to focus on developing your apps or services

- Don't have dedicated site reliability engineering (SRE) team but want high availability
- Don't have resources to host and monitor your clusters

Pick a hosted solution.

Turnkey – Cloud Solutions

These solutions allow you to create Kubernetes clusters with only a few commands and are actively developed and have active community support. They can also be hosted on a range of Cloud IaaS providers, but they offer more freedom and flexibility in exchange for effort.

You should pick a turnkey cloud solution if you:

- Want more control over your clusters than the hosted solutions allow
- Want to take on more operations ownership

Pick a turnkey cloud solution

Turnkey – On-Premises Solutions

These solutions allow you to create Kubernetes clusters on your internal, secure, cloud network with only a few commands.

You should pick a on-prem turnkey cloud solution if you:

- Want to deploy clusters on your private cloud network
- Have a dedicated SRE team
- Have the resources to host and monitor your clusters

Pick an on-prem turnkey cloud solution.

Custom Solutions

Custom solutions give you the most freedom over your clusters but require the most expertise. These solutions range from bare-metal to cloud providers on different operating systems.

Pick a custom solution.

What's next

Go to Picking the Right Solution for a complete list of solutions.

Feedback

Was this page helpful?

Yes

No

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

Edit This Page

Page last modified on October 04, 2018 at 3:37 AM PST by Trivial typo fixes (#10386) (Page History)

Edit This Page

Picking the Right Solution

Kubernetes can run on various platforms: from your laptop, to VMs on a cloud provider, to a rack of bare metal servers. The effort required to set up a cluster varies from running a single command to crafting your own customized cluster. Use this guide to choose a solution that fits your needs.

If you just want to "kick the tires" on Kubernetes, use the local Docker-based solutions.

When you are ready to scale up to more machines and higher availability, a hosted solution is the easiest to create and maintain.

Turnkey cloud solutions require only a few commands to create and cover a wide range of cloud providers. On-Premises turnkey cloud solutions have the simplicity of the turnkey cloud solution combined with the security of your own private network.

If you already have a way to configure hosting resources, use kubeadm to easily bring up a cluster with a single command per machine.

Custom solutions vary from step-by-step instructions to general advice for setting up a Kubernetes cluster from scratch.

- Local-machine Solutions
- Hosted Solutions
- Turnkey Cloud Solutions
- On-Premises turnkey cloud solutions

- Custom Solutions
- Table of Solutions

Local-machine Solutions

Community Supported Tools

- Minikube is a method for creating a local, single-node Kubernetes cluster for development and testing. Setup is completely automated and doesn't require a cloud provider account.
- Kubeadm-dind is a multi-node (while minikube is single-node) Kubernetes cluster which only requires a docker daemon. It uses docker-in-docker technique to spawn the Kubernetes cluster.
- Kubernetes IN Docker is a tool for running local Kubernetes clusters using Docker container "nodes". It is primarily designed for testing Kubernetes 1.11+. You can use it to create multi-node or multi-control-plane Kubernetes clusters.

Ecosystem Tools

- Docker Desktop is an easy-to-install application for your Mac or Windows environment that enables you to start coding and deploying in containers in minutes on a single-node Kubernetes cluster.
- Minishift installs the community version of the Kubernetes enterprise platform OpenShift for local development & testing. It offers an all-in-one VM (`minishift start`) for Windows, macOS, and Linux. The container start is based on `oc cluster up` (Linux only). You can also install the included add-ons.
- MicroK8s provides a single command installation of the latest Kubernetes release on a local machine for development and testing. Setup is quick, fast (~30 sec) and supports many plugins including Istio with a single command.
- IBM Cloud Private-CE (Community Edition) can use VirtualBox on your machine to deploy Kubernetes to one or more VMs for development and test scenarios. Scales to full multi-node cluster.
- IBM Cloud Private-CE (Community Edition) on Linux Containers is a Terraform/Packer/BASH based Infrastructure as Code (IaC) scripts to create a seven node (1 Boot, 1 Master, 1 Management, 1 Proxy and 3 Workers) LXD cluster on Linux Host.
- Ubuntu on LXD supports a nine-instance deployment on localhost.

Hosted Solutions

- AppsCode.com provides managed Kubernetes clusters for various public clouds, including AWS and Google Cloud Platform.
- APPUiO runs an OpenShift public cloud platform, supporting any Kubernetes workload. Additionally APPUiO offers Private Managed OpenShift Clusters, running on any public or private cloud.
- Amazon Elastic Container Service for Kubernetes offers managed Kubernetes service.
- Azure Kubernetes Service offers managed Kubernetes clusters.
- Containership Kubernetes Engine (CKE) intuitive Kubernetes cluster provisioning and management on GCP, Azure, AWS, Packet, and DigitalOcean. Seamless version upgrades, autoscaling, metrics, workload creation, and more.
- DigitalOcean Kubernetes offers managed Kubernetes service.
- Giant Swarm offers managed Kubernetes clusters in their own datacenter, on-premises, or on public clouds.
- Google Kubernetes Engine offers managed Kubernetes clusters.
- IBM Cloud Kubernetes Service offers managed Kubernetes clusters with isolation choice, operational tools, integrated security insight into images and containers, and integration with Watson, IoT, and data.
- Kubermatic provides managed Kubernetes clusters for various public clouds, including AWS and Digital Ocean, as well as on-premises with OpenStack integration.
- Kublr offers enterprise-grade secure, scalable, highly reliable Kubernetes clusters on AWS, Azure, GCP, and on-premise. It includes out-of-the-box backup and disaster recovery, multi-cluster centralized logging and monitoring, and built-in alerting.
- KubeSail is an easy, free way to try Kubernetes.
- Madcore.Ai is devops-focused CLI tool for deploying Kubernetes infrastructure in AWS. Master, auto-scaling group nodes with spot-instances, ingress-ssl-lego, Heapster, and Grafana.
- Nutanix Karbon is a multi-cluster, highly available Kubernetes management and operational platform that simplifies the provisioning, operations, and lifecycle management of Kubernetes.
- OpenShift Dedicated offers managed Kubernetes clusters powered by OpenShift.
- OpenShift Online provides free hosted access for Kubernetes applications.

- Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE) is a fully-managed, scalable, and highly available service that you can use to deploy your containerized applications to the cloud.
- Platform9 offers managed Kubernetes on-premises or on any public cloud, and provides 24/7 health monitoring and alerting. (Kube2go, a web-UI driven Kubernetes cluster deployment service Platform9 released, has been integrated to Platform9 Sandbox.)
- Stackpoint.io provides Kubernetes infrastructure automation and management for multiple public clouds.
- SysEleven MetaKube offers managed Kubernetes as a service powered on our OpenStack public cloud. It includes lifecycle management, administration dashboards, monitoring, autoscaling and much more.
- VEXXHOST VEXXHOST proudly offers Certified Kubernetes on our public cloud, which also happens to be the largest OpenStack public cloud in Canada.
- VMware Cloud PKS is an enterprise Kubernetes-as-a-Service offering in the VMware Cloud Services portfolio that provides easy to use, secure by default, cost effective, SaaS-based Kubernetes clusters.

Turnkey Cloud Solutions

These solutions allow you to create Kubernetes clusters on a range of Cloud IaaS providers with only a few commands. These solutions are actively developed and have active community support.

- Agile Stacks
- Alibaba Cloud
- APPUiO
- AWS
- Azure
- CenturyLink Cloud
- Conjure-up Kubernetes with Ubuntu on AWS, Azure, Google Cloud, Oracle Cloud
- Containership
- Docker Enterprise
- Gardener
- Giant Swarm
- Google Compute Engine (GCE)
- IBM Cloud
- Kontena Pharos
- Kubermatic
- Kublr
- Madcore.Ai

- Nirmata
- Nutanix Karbon
- Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE)
- Pivotal Container Service
- Rancher 2.0
- Stackpoint.io
- Supergiant.io
- VEXXHOST
- VMware Cloud PKS
- VMware Enterprise PKS

On-Premises turnkey cloud solutions

These solutions allow you to create Kubernetes clusters on your internal, secure, cloud network with only a few commands.

- Agile Stacks
- APPUiO
- Docker Enterprise
- Giant Swarm
- GKE On-Prem | Google Cloud
- IBM Cloud Private
- Kontena Pharos
- Kubermatic
- Kublr
- Mirantis Cloud Platform
- Nirmata
- OpenShift Container Platform (OCP) by Red Hat
- Pivotal Container Service
- Rancher 2.0
- SUSE CaaS Platform
- SUSE Cloud Application Platform
- VMware Enterprise PKS

Custom Solutions

Kubernetes can run on a wide range of Cloud providers and bare-metal environments, and with many base operating systems.

If you can find a guide below that matches your needs, use it.

Universal

If you already have a way to configure hosting resources, use kubeadm to bring up a cluster with a single command per machine.

Cloud

These solutions are combinations of cloud providers and operating systems not covered by the above solutions.

- Cloud Foundry Container Runtime (CFCR)
- Gardener
- Kublr
- Kubernetes on Ubuntu
- Kubespray
- Rancher Kubernetes Engine (RKE)
- VMware Essential PKS

On-Premises VMs

- Cloud Foundry Container Runtime (CFCR)
- CloudStack (uses Ansible)
- Fedora (Multi Node) (uses Fedora and flannel)
- Nutanix AHV
- OpenShift Container Platform (OCP) Kubernetes platform by Red Hat
- oVirt
- VMware Essential PKS
- VMware vSphere
- VMware vSphere, OpenStack, or Bare Metal (uses Juju, Ubuntu and flannel)

Bare Metal

- Digital Rebar
- Docker Enterprise
- Fedora (Single Node)
- Fedora (Multi Node)
- Kubernetes on Ubuntu
- OpenShift Container Platform (OCP) Kubernetes platform by Red Hat
- VMware Essential PKS

Integrations

These solutions provide integration with third-party schedulers, resource managers, and/or lower level platforms.

- DCOS
  - Community Edition DCOS uses AWS
  - Enterprise Edition DCOS supports cloud hosting, on-premises VMs, and bare metal

Table of Solutions

Below is a table of all of the solutions listed above.

IaaS Provider	Config. Mgmt.	OS
Agile Stacks	Terraform	CoreOS
Alibaba Cloud Container Service For Kubernetes	ROS	CentOS
any	any	multi-su
any	any	any
any	any	any
any	RKE	multi-su
any	Gardener Cluster-Operator	multi-su
AppsCode.com	Saltstack	Debian
AWS	CoreOS	CoreOS
AWS	Saltstack	Debian
AWS	kops	Debian
AWS	Juju	Ubuntu
Azure	Juju	Ubuntu
Azure (IaaS)		Ubuntu
Azure Kubernetes Service		Ubuntu
Bare-metal	custom	CentOS
Bare-metal	custom	Fedora
Bare-metal	custom	Fedora
Bare Metal	Juju	Ubuntu
Bare-metal	custom	Ubuntu
CloudStack	Ansible	CoreOS
DCOS	Marathon	CoreOS
Digital Rebar	kubeadm	any
Docker Enterprise	custom	multi-su
Giant Swarm		CoreOS
GCE	CoreOS	CoreOS
GCE	Juju	Ubuntu
GCE	Saltstack	Debian
Google Kubernetes Engine		
IBM Cloud Kubernetes Service		Ubuntu

IaaS Provider	Config. Mgmt.	OS
IBM Cloud Kubernetes Service		Ubuntu
IBM Cloud Private	Ansible	multi-su
Kublr	custom	multi-su
Kubermatic		multi-su
KVM	custom	Fedora
libvirt	custom	Fedora
lxd	Juju	Ubuntu
Madcore.Ai	Jenkins DSL	Ubuntu
Mirantis Cloud Platform	Salt	Ubuntu
Oracle Cloud Infrastructure	Juju	Ubuntu
Oracle Cloud Infrastructure Container Engine for Kubernetes (OKE)		
oVirt		
Platform9		multi-su
Rackspace	custom	CoreOS
Red Hat OpenShift	Ansible & CoreOS	RHEL &
Stackpoint.io		multi-su
Vagrant	CoreOS	CoreOS
VMware vSphere	any	multi-su
VMware vSphere	Juju	Ubuntu
VMware Cloud PKS		Photon
VMware Enterprise PKS	BOSH	Ubuntu
VMware Essential PKS	any	multi-su

Definition of columns

- **IaaS Provider** is the product or organization which provides the virtual or physical machines (nodes) that Kubernetes runs on.
- **OS** is the base operating system of the nodes.
- **Config. Mgmt.** is the configuration management system that helps install and maintain Kubernetes on the nodes.
- **Networking** is what implements the networking model. Those with networking type *none* may not support more than a single node, or may support multiple VM nodes in a single physical node.
- **Support Levels**
  - **Project**: Kubernetes committers regularly use this configuration, so it usually works with the latest release of Kubernetes.
  - **Commercial**: A commercial offering with its own support arrangements.
  - **Community**: Actively supported by community contributions. May not work with recent releases of Kubernetes.
  - **Inactive**: Not actively maintained. Not recommended for first-time Kubernetes users, and may be removed.

- **Notes** has other relevant information, such as the version of Kubernetes used.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 20, 2019 at 8:43 PM PST by Update table (#13317) (Page History)

Edit This Page

v1.13 Release Notes

- v1.13.0
  - Downloads for v1.13.0
    - Client Binaries
    - Server Binaries
    - Node Binaries
- Kubernetes 1.13 Release Notes
  - Security Content
  - Urgent Upgrade Notes
    - (No, really, you MUST do this before you upgrade)
  - Known Issues
  - Deprecations
  - Major Themes
    - SIG API Machinery
    - SIG Auth
    - SIG AWS
    - SIG Azure
    - SIG Big Data
    - SIG CLI
    - SIG Cloud Provider

        * SIG Cluster Lifecycle
        * SIG IBM Cloud
        * SIG Multicluster
        * SIG Network
        * SIG Node
        * SIG Openstack
        * SIG Scalability
        * SIG Scheduling
        * SIG Service Catalog
        * SIG Storage
        * SIG UI
        * SIG VMWare
        * SIG Windows
    - New Features
    - Release Notes From SIGs
        * SIG API Machinery
        * SIG Auth
        * SIG Autoscaling
        * SIG AWS
        * SIG Azure
        * SIG CLI
        * SIG Cloud Provider
        * SIG Cluster Lifecycle
        * SIG GCP
        * SIG Network
        * SIG Node
        * SIG OpenStack
        * SIG Release
        * SIG Scheduling
        * SIG Storage
        * SIG Windows
    - External Dependencies
- v1.13.0-rc.2
    - Downloads for v1.13.0-rc.2
        * Client Binaries
        * Server Binaries
        * Node Binaries
    - Changelog since v1.13.0-rc.1
        * Other notable changes
- v1.13.0-rc.1
    - Downloads for v1.13.0-rc.1
        * Client Binaries
        * Server Binaries
        * Node Binaries
    - Changelog since v1.13.0-beta.2
        * Other notable changes

v1.13.0

Documentation

Downloads for v1.13.0

filename	sha512 hash
kubernetes.tar.gz	7b6a81c9f1b852b1e889c1b62281569a4b8853c79e5675b0910d941dfa7863c97f244f6d
kubernetes-src.tar.gz	844b9fbba21374dd190c8f12dd0e5b3303dd2cd7ad25f241d6f7e46f74adf6987afad021

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	0c010351acb660a75122feb876c9887d46ec2cb466872dd073b7f5b2
kubernetes-client-darwin-amd64.tar.gz	c2c40bd202900124f4e9458b067a1e1fc040030dc84ce9bcc6a5beb2
kubernetes-client-linux-386.tar.gz	5f5449be103b103d72a4e2b1028ab014cf7f74781166327f2ae284e4
kubernetes-client-linux-amd64.tar.gz	61a6cd3b1fb34507e0b762a45da09d88e34921985970a2ba594e0e5a
kubernetes-client-linux-arm.tar.gz	dd5591e2b88c347759a138c4d2436a0f5252341d0e8c9fbab16b8f15
kubernetes-client-linux-arm64.tar.gz	894ed30261598ebf3485f3575e95f85e3c353f4d834bf9a6ea53b265
kubernetes-client-linux-ppc64le.tar.gz	6c26c807fc730ea736fda75dc57ac73395ba78bb828fffeee18b385b
kubernetes-client-linux-s390x.tar.gz	41e6e972de77c0bde22fdd779ea64e731b60f32e97e78a024f33fc3e
kubernetes-client-windows-386.tar.gz	442229e5030452901b924a94e7a879d4085597a4f201a5b3fc5ac980
kubernetes-client-windows-amd64.tar.gz	a11a8e8e732e7292781b9cb1de6e3e41683f95fb3fefc2b1a7b5fb1f

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	a8e3d457e5bcc1c09eeb66111e8dd049d6ba048c3c0fa90a61814291af
kubernetes-server-linux-arm.tar.gz	4e17494767000256775e4dd33c0a9b2d152bd4b5fba9f343b6dfeb5746
kubernetes-server-linux-arm64.tar.gz	0ddd0cf0ff56cebfa89efb1972cc2bc6916e824c2af56cfd330ac5638c
kubernetes-server-linux-ppc64le.tar.gz	b93828560224e812ed21b57fea5458fa8560745cfec96fc1677b258393
kubernetes-server-linux-s390x.tar.gz	154d565329d5ba52cdb7c3d43d8854b7a9b8e34803c4df6b3e6ae74c1a

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	9d18ba5f0c3b09edcf29397a496a1e908f4906087be3792989285630c
kubernetes-node-linux-arm.tar.gz	959b04ff7b8690413e01bffeabaab2119794dedf06b7aae1743e49988
kubernetes-node-linux-arm64.tar.gz	b5c18e8c9e28cf276067c871446720d86b6f162e22c3a5e9343cdbc68
kubernetes-node-linux-ppc64le.tar.gz	63e3504d3b115fdf3396968afafd1107b98e5a1a15b7c042a87f5a9cf
kubernetes-node-linux-s390x.tar.gz	21c5c2721febf7fddeada9569f3ecbd059267e5d2cc325d98fb74faf1
kubernetes-node-windows-amd64.tar.gz	3e73d3ecff14b4c85a71bb6cf91b1ab7d9c3075c64bd5ce6863562ab1

Kubernetes 1.13 Release Notes

Security Content

- CVE-2018-1002105, a critical security issue in the Kubernetes API Server, is resolved in v1.13.0 (and in v1.10.11, v1.11.5, and v1.12.3). We recommend all clusters running previous versions update to one of these releases immediately. See issue #71411 for details.

Urgent Upgrade Notes

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

Before upgrading to Kubernetes 1.13, you must keep the following in mind:

- kube-apiserver
  - The deprecated `etcd2` storage backend has been removed. Before upgrading a kube-apiserver using `--storage-backend=etcd2`, etcd v2 data must be migrated to the v3 storage backend, and kube-apiserver invocations changed to use `--storage-backend=etcd3`. Please consult the installation procedure used to set up etcd for specific migration instructions. Backups prior to upgrade are always a good practice, but since the etcd2 to etcd3 migration is not reversible, an etcd backup prior to migration is essential.
  - The deprecated `--etcd-quorum-read` flag has been removed. Quorum reads are now always enabled when fetching data from etcd. Remove the `--etcd-quorum-read` flag from kube-apiserver invocations before upgrading.
- kube-controller-manager
  - The deprecated `--insecure-experimental-approve-all-kubelet-csrs-for-group` flag has been removed.
- kubelet
  - The deprecated `--google-json-key` flag has been removed. Remove the `--google-json-key` flag from kubelet invocations before upgrading. (#69354, @yujuhong)
  - DaemonSet pods now make use of scheduling features that require kubelets to be at 1.11 or above. Ensure all kubelets in the cluster are at 1.11 or above before upgrading kube-controller-manager to 1.13.
  - The schema for the alpha `CSINodeInfo` CRD has been split into `spec` and `status` fields, and new fields `status.available` and `status.volumePluginMechanism` added. Clusters using the previous alpha schema must delete and recreate the CRD using the new schema. (#70515, @davidz627)
- kube-scheduler dropped support for configuration files with apiVersion `componentconfig/v1alpha1`. Ensure kube-scheduler is configured

using command-line flags or a configuration file with apiVersion `kubescheduler.config.k8s.io/v1alpha1` before upgrading to 1.13.
- kubectl
  - The deprecated command `run-container` has been removed. Invocations should use `kubectl run` instead (#70728, @Pingan2017)
- client-go releases will no longer have bootstrap (k8s.io/client-go/tools/bootstrap) related code. Any reference to it will break. Please redirect all references to k8s.io/bootstrap instead. (#67356, @yliaog)
- Kubernetes cannot distinguish between GCE Zonal PDs and Regional PDs with the same name. To workaround this issue, precreate PDs with unique names. PDs that are dynamically provisioned do not encounter this issue. (#70716, @msau42)

Known Issues

- If kubelet plugin registration for a driver fails, kubelet will not retry. The driver must delete and recreate the driver registration socket in order to force kubelet to attempt registration again. Restarting only the driver container may not be sufficient to trigger recreation of the socket, instead a pod restart may be required. (#71487)
- In some cases, a Flex volume resize may leave a PVC with erroneous Resizing condition even after volume has been successfully expanded. Users may choose to delete the condition, but it is not required. (#71470)
- The CSI driver-registrar external sidecar container v1.0.0-rc2 is known to take up to 1 minute to start in some cases. We expect this issue to be resolved in a future release of the sidecar container. For verification, please see the release notes of future releases of the external sidecar container. (#76)
- When using IPV6-only, be sure to use `proxy-mode=iptables` as `proxy-mode=ipvs` is known to not work. (#68437)

Deprecations

- kube-apiserver
  - The `--service-account-api-audiences` flag is deprecated in favor of `--api-audiences`. The old flag is accepted with a warning but will be removed in a future release. (#70105, @mikedanese)
  - The `--experimental-encryption-provider-config` flag is deprecated in favor of `--encryption-provider-config`. The old flag is accepted with a warning but will be removed in 1.14. (#71206, @stlaz)
  - As part of graduating the etcd encryption feature to beta, the configuration file referenced by `--encryption-provider-config` now uses `kind: EncryptionConfiguration` and `apiVersion:`

apiserver.config.k8s.io/v1. Support for kind: EncryptionConfig and apiVersion: v1 is deprecated and will be removed in a future release. (#67383, @stlaz)
  - The --deserialization-cache-size flag is deprecated, and will be removed in a future release. The flag is inactive since the etcd2 storage backend was removed. (#69842, @liggitt)
  - The Node authorization mode no longer allows kubelets to delete their Node API objects (prior to 1.11, in rare circumstances related to cloudprovider node ID changes, kubelets would attempt to delete/recreate their Node object at startup) (#71021, @liggitt)
  - The built-in system:csi-external-provisioner and system:csi-external-attacher cluster roles are deprecated and will not be auto-created in a future release. CSI deployments should provide their own RBAC role definitions with required permissions. (#69868, @pohly)
  - The built-in system:aws-cloud-provider cluster role is deprecated and will not be auto-created in a future release. Deployments using the AWS cloud provider should grant required permissions to the aws-cloud-provider service account in the kube-system namespace as part of deployment. (#66635, @wgliang)
- kubelet
  - Use of the beta plugin registration directory {kubelet_root_dir}/plugins/ for registration of external drivers via the kubelet plugin registration protocol is deprecated in favor of {kubelet_root_dir}/plugins_registry/. Support for the old directory is planned to be removed in v1.15. Device plugin and CSI storage drivers should switch to the new directory prior to v1.15. Only CSI storage drivers that support 0.x versions of the CSI API are allowed in the old directory. (#70494 by @RenaudWasTaken and #71314 by @saad-ali)
  - With the release of the CSI 1.0 API, support for CSI drivers using 0.3 and older releases of the CSI API is deprecated, and is planned to be removed in Kubernetes v1.15. CSI drivers should be updated to support the CSI 1.0 API, and deployed in the new kubelet plugin registration directory ({kubelet_root_dir}/plugins_registry/) once all nodes in the cluster are at 1.13 or higher (#71020 and #71314, both by @saad-ali)
  - Use of the --node-labels flag to set labels under the kubernetes.io/ and k8s.io/ prefix will be subject to restriction by the NodeRestriction admission plugin in future releases. See admission plugin documentation for allowed labels. (#68267, @liggitt)
- kube-scheduler
  - The alpha critical pod annotation (scheduler.alpha.kubernetes.io/critical-pod) is deprecated. Pod priority should be used instead to mark pods as critical. (#70298, @bsalamat)
- The following features are now GA, and the associated feature gates are deprecated and will be removed in a future release:
  - CSIPersistentVolume

  - GCERegionalPersistentDisk
  - KubeletPluginsWatcher
  - VolumeScheduling
- kubeadm
  - The DynamicKubeletConfig feature gate is deprecated. The functionality is still accessible by using the kubeadm alpha kubelet enable-dynamic command.
  - The command `kubeadm config print-defaults` is deprecated in favor of `kubeadm config print init-defaults` and `kubeadm config print join-defaults` (#69617, @rosti)
  - support for the `v1alpha3` configuration file format is deprecated and will be removed in 1.14. Use `kubeadm config migrate` to migrate `v1alpha3` configuration files to `v1beta1`, which provides improvements in image repository management, addons configuration, and other areas. The documentation for `v1beta1` can be found here: https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta1
- The `node.status.volumes.attached.devicePath` field is deprecated for CSI volumes and will not be set in future releases (#71095, @msau42)
- kubectl
  - The `kubectl convert` command is deprecated and will be removed in a future release (#70820, @seans3)
- Support for passing unknown provider names to the E2E test binaries is deprecated and will be removed in a future release. Use `--provider=skeleton` (no ssh access) or `--provider=local` (local cluster with ssh) instead. (#70141, @pohly)

Major Themes

SIG API Machinery

For the 1.13 release, SIG API Machinery is happy to announce that the dry-run functionality is now beta.

SIG Auth

With this release we've made several important enhancements to core SIG Auth areas. In the authorization category, we've further reduced Kubelet privileges by restricting node self-updates of labels to a whitelisted selection and by disallowing kubelets from deleting their Node API object. In authentication, we added alpha-level support for automounting improved service account tokens through projected volumes. We also enabled audience validation in TokenReview for the new tokens for improved scoping. Under audit logging, the new

alpha-level "dynamic audit configuration" adds support for dynamically registering webhooks to receive a stream of audit events. Finally, we've enhanced secrets protection by graduating etcd encryption out of experimental.

SIG AWS

In v1.13 we worked on tighter integrations of Kubernetes API objects with AWS services. These include three out-of-tree alpha feature releases:

1) Alpha for AWS ALB (Application Load Balancer) integration to Kubernetes Ingress resources. 2) Alpha for CSI specification 0.3 integration to AWS EBS (Elastic Block Store) 3) Alpha for the cloudprovider-aws cloud controller manager binary. Additionally we added aws-k8s-tester, deployer interface for kubetest, to the test-infra repository. This plugin allowed us to integrate Prow to the 3 subprojects defined above in order to provide CI signal for all 3 features. The CI signal is visible here under SIG-AWS.

For detailed release notes on the three alpha features from SIG AWS, please refer to the following Changelogs:

- aws-alb-ingress-controller v1.0.0
- aws-ebs-csi-driver v0.1
- cloudprovider-aws external v0.1.0

SIG Azure

For 1.13 SIG Azure was focused on adding additional Azure Disk support for Ultra SSD, Standard SSD, and Premium Azure Files. Azure Availability Zones and cross resource group nodes were also moved from Alpha to Beta in 1.13.

SIG Big Data

During the 1.13 release cycle, SIG Big Data has been focused on community engagements relating to 3rd-party project integrations with Kubernetes. There have been no impacts on the 1.13 release.

SIG CLI

Over the course of 1.13 release SIG CLI mostly focused on stabilizing the items we've been working on over the past releases such as server-side printing and its support in kubectl, as well as finishing kubectl diff which is based on server-side dry-run feature. We've continued separating kubectl code to prepare for extraction out of main repository. Finally, thanks to the awesome support and

feedback from community we've managed to promote the new plugin mechanism to Beta.

SIG Cloud Provider

For v1.13, SIG Cloud Provider has been focused on stabilizing the common APIs and interfaces consumed by cloud providers today. This involved auditing the cloud provider APIs for anything that should be deprecated as well as adding changes where necessary. In addition, SIG Cloud Provider has begun exploratory work around having a "cloud provider" e2e test suite which can be used to test common cloud provider functionalities with resources such as nodes and load balancers.

We are also continuing our long running effort to extract all the existing cloud providers that live in k8s.io/kubernetes into their own respective repos. Along with this migration, we are slowly transitioning users to use the cloud-controller-manager for any cloud provider features instead of the kube-controller-manager.

SIG Cluster Lifecycle

For 1.13 SIG Cluster Lifecycle is pleased to announce the long awaited promotion of kubeadm to stable GA, and the promotion of kubeadm's configuration API to `v1beta1`. In this release the SIG again focused on further improving the user experience on cluster creation and also fixing a number of bugs and other assorted improvements.

Some notable changes in kubeadm since Kubernetes 1.12:

- kubeadm's configuration API is now `v1beta1`. The new configuration format provides improvements in - image repository management, addons configuration, and other areas. We encourage `v1alpha3` users to migrate to this configuration API using `kubeadm config migrate`, as `v1alpha3` will be removed in 1.14. The documentation for `v1beta1` can be found here: https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta1
- kubeadm has graduated `kubeadm alpha phase` commands to `kubeadm init phase`. This means that the phases of creating a control-plane node are now tightly integrated as part of the `init` command. Alpha features, not yet ready for GA are still kept under `kubeadm alpha` and we appreciate feedback on them.
- `kubeadm init` and `kubeadm init phase` now have a `--image-repository` flag, improving support for environments with limited access to official kubernetes repository.
- The DynamicKubeletConfig and SelfHosting functionality was moved outside of `kubeadm init` and feature gates and is now exposed under `kubeadm alpha`.

- Kubeadm init phase certs now support the `--csr-only` option, simplifying custom CA creation.
- `kubeadm join --experimental-control-plane` now automatically adds a new etcd member for `local etcd` mode, further simplifying required tasks for HA clusters setup.
- Improvements were made to `kubeadm reset` related to cleaning etcd and notifying the user about the state of iptables.
- kubeadm commands now print warnings if input YAML documents contain unknown or duplicate fields.
- kubeadm now properly recognizes Docker 18.09.0 and newer, but still treats 18.06 as the default supported version.
- kubeadm now automatically sets the `--pod-infra-container-image` flag when starting the kubelet.

SIG IBM Cloud

The IBM Cloud SIG was focused on defining its charter and working towards moving its cloud provider code to an external repository with a goal to have this work done by the end of Kubernetes 1.14 release cycle. In the SIG meetings, we also made sure to share updates on the latest Kubernetes developments in the IBM Cloud like the availability of Kubernetes v1.12.2 in the IBM Cloud Kubernetes Service (IKS). The SIG updates were provided in the Kubernetes community weekly call and at the KubeCon China 2018.

SIG Multicluster

Moving Federation v2 from Alpha towards Beta has been the focus of our effort over the past quarter. To this end we engaged with end users, and successfully enlisted additional contributors from companies including IBM, Amadeus, Cisco and others. Federation v2 provides a suite of decoupled API's and re-usable components for building multi-cluster control planes. We plan to start releasing Beta components in late 2018. In addition, more minor updates were made to our cluster-registry and multi-cluster ingress sub-projects.

SIG Network

For 1.13, the areas of focus were in IPv6, DNS improvements and some smaller items: CoreDNS is now the default cluster DNS passing all of the scale/resource usage tests Node-local DNS cache feature is available in Alpha. This feature deploys a lightweight DNS caching Daemonset that avoids the conntrack and converts queries from UDP to more reliable TCP. PodReady++ feature now has `kubectl` CLI support.

Progress was made towards finalizing the IPv6 dual stack support KEP and support for topological routing of services.

SIG Node

SIG Node focused on stability and performance improvements in the 1.13 release. A new alpha feature is introduced to improve the mechanism that nodes heartbeat back to the control plane. The `NodeLease` feature results in the node using a `Lease` resource in the `kube-node-lease` namespace that is renewed periodically. The `NodeStatus` that was used previously to heartbeat back to the control plane is only updated when it changes. This reduces load on the control plane for large clusters. The Kubelet plugin registration mechanism, which enables automatic discovery of external plugins (including CSI and device plugins) has been promoted to stable in this release (introduced as alpha in 1.11 and promoted to beta in 1.12).

SIG Openstack

The major theme for the SIG OpenStack release is the work-in-progress for removing the in-tree provider. This work, being done in conjunction with SIG Cloud Provider, is focusing on moving internal APIs that the OpenStack (and other providers) depends upon to staging to guarantee API stability. This work also included abstracting the in-tree Cinder API and refactoring code to the external Cinder provider to remove additional Cinder volume provider code.

Additional work was also done to implement an OpenStack driver for the Cluster API effort lead by SIG Cluster Lifecycle. For the external Cloud-Provider-OpenStack code, the SIG largely focused on bug fixes and updates to match K8s 1.13 development.

SIG Scalability

SIG Scalability has mostly focused on stability and deflaking our tests, investing into framework for writing scalability tests (ClusterLoader v2) with a goal to migrate all tests to it by the end of 2018 and on the work towards extending definition of Kubernetes scalability by providing more/better user-friendly SLIs/SLOs.

SIG Scheduling

SIG Scheduling has mostly focused on stability in 1.13 and has postponed some of the major features to the next versions. There are still two notable changes: 1. TaintBasedEviction is moved to Beta and will be enabled by default. With this

feature enabled, condition taints are automatically added to the nodes and pods can add tolerations for them if needed. 2. Pod critical annotation is deprecated. Pods should use pod priority instead of the annotation.

It is worth noting again that kube-scheduler will use apiVersion `kubescheduler.config.k8s.io/v1alpha1` instead of `componentconfig/v1alpha1` in its configuration files in 1.13.

SIG Service Catalog

The Service Plan Defaults feature is still under active development. We continue to improve the UX for the svcat CLI, specifically filling in gaps for the new Namespaced Service Broker feature.

SIG Storage

Over the last year, SIG Storage has been focused on adding support for the Container Storage Interface (CSI) to Kubernetes. The specification recently moved to 1.0, and on the heels of this achievement, Kubernetes v1.13 moves CSI support for PersistentVolumes to GA.

With CSI the Kubernetes volume layer becomes truly extensible, allowing third party storage developers to write drivers making their storage systems available in Kubernetes without having to touch the core code.

CSI was first introduction as alpha in Kubernetes v1.9 and moved to beta in Kubernetes v1.10.

You can find a list of sample and production drivers in the CSI Documentation.

SIG Storage also moves support for Block Volumes to beta (introduced as alpha in v1.9) and support for Topology Aware Volume Scheduling to stable (introduced as alpha in v1.9 and promoted to beta in 1.10).

SIG UI

The migration to the newest version of Angular is still under active development as it is most important thing on the roadmap at the moment. We are getting closer to to the new release. We continue fixing bugs and adding other improvements.

SIG VMWare

Major focus for SIG VMware for this release is the work on moving internal APIs that the vSphere provider depends upon to staging to guarantee API stability. This work is being done in conjunction with SIG Cloud Provider and includes

the creation of a brand new vsphere-csi plugin to replace the current volume functionalities in-tree.

Additional work was also done to implement a vSphere provider for the Cluster API effort lead by SIG Cluster Lifecycle. For the out-of-tree vSphere cloud provider, the SIG largely focused on bug fixes and updates to match K8s 1.13 development.

SIG Windows

SIG Windows focused on improving reliability for Windows and Kubernetes support

New Features

- kubelet: When node lease feature is enabled, kubelet reports node status to api server only if there is some change or it didn't report over last report interval. (#69753, @wangzhen127)
- vSphereVolume implements Raw Block Volume Support (#68761, @fanzhangio)
- CRD supports multi-version Schema, Subresources and AdditionalPrintColumns (NOTE that CRDs created prior to 1.13 populated the top-level additionalPrinterColumns field by default. To apply an updated that changes to per-version additionalPrinterColumns, the top-level additionalPrinterColumns field must be explicitly set to null). (#70211, @roycaihw)
- New addon in addon manager that automatically installs CSI CRDs if CSIDriverRegistry or CSINodeInfo feature gates are true. (#70193, @saad-ali)
- Delegated authorization can now allow unrestricted access for `system:masters` like the main kube-apiserver (#70671, @deads2k)
- Added dns capabilities for Windows CNI plugins: (#67435, @feiskyer)
- kube-apiserver: `--audit-webhook-version` and `--audit-log-version` now default to `audit.k8s.io/v1` if unspecified (#70476, @charrywanganthony)
- kubeadm: timeoutForControlPlane is introduced as part of the API Server config, that controls the timeout for the wait for control plane to be up. Default value is 4 minutes. (#70480, @rosti)
- `--api-audiences` now defaults to the `--service-account-issuer` if the issuer is provided but the API audience is not. (#70308, @mikedanese)
- Added support for projected volume in describe function (#70158, @WanLinghao)
- kubeadm now automatically creates a new stacked etcd member when joining a new control plane node (does not applies to external etcd) (#69486, @fabriziopandini)

- Display the usage of ephemeral-storage when using `kubectl describe node` (#70268, @Pingan2017)
- Added functionality to enable br_netfilter and ip_forward for debian packages to improve kubeadm support for CRI runtime besides Docker. (#70152, @ashwanikhemani)
- Added regions ap-northeast-3 and eu-west-3 to the list of well known AWS regions. (#70252, @nckturner)
- kubeadm: Implemented preflight check to ensure that number of CPUs (#70048, @bart0sh)
- CoreDNS is now the default DNS server in kube-up deployments. (#69883, @chrisohaver)
- Opt out of chowning and chmoding from kubectl cp. (#69573, @bjhaid)
- Failed to provision volume with StorageClass "azurefile-premium": failed to create share andy-mg1121-dynamic-pvc-1a7b2813-d1b7-11e8-9e96-000d3a03e16b in account f7228f99bcde411e8ba4900: failed to create file share, err: storage: service returned error: StatusCode=400, ErrorCode=InvalidHeaderValue, ErrorMessage=The value for one of the HTTP headers is not in the correct format. (#69718, @andyzhangx)
- `TaintBasedEvictions` feature is promoted to beta. (#69824, @Huang-Wei)
- Fixed https://github.com/kubernetes/client-go/issues/478 by adding support for JSON Patch in client-go/dynamic/fake (#69330, @vaikas-google)
- Dry-run is promoted to Beta and will be enabled by default. (#69644, @apelisse)
- `kubectl get priorityclass` now prints value column by default. (#69431, @Huang-Wei)
- Added a new container based image for running e2e tests (#69368, @dims)
- The `LC_ALL` and `LC_MESSAGES` env vars can now be used to set desired locale for `kubectl` while keeping `LANG` unchanged. (#69500, @m1kola)
- NodeLifecycleController: Now node lease renewal is treated as the heartbeat signal from the node, in addition to NodeStatus Update. (#69241, @wangzhen127)
- Added dynamic shared informers to write generic, non-generated controllers (#69308, @p0lyn0mial)
- Upgraded to etcd 3.3 client (#69322, @jpbetz)
- It is now possible to use named ports in the `kubectl port-forward` command (#69477, @m1kola)
- `kubectl wait` now supports condition value checks other than true using `--for condition=available=false` (#69295, @deads2k)
- Updated defaultbackend image to 1.5. Users should concentrate on updating scripts to the new version. (#69120, @aledbf)
- Bumped Dashboard version to v1.10.0 (#68450, @jeefy)
- Added env variables to control CPU requests of kube-controller-manager and kube-scheduler. (#68823, @loburm)
- PodSecurityPolicy objects now support a `MayRunAs` rule for `fsGroup` and `supplementalGroups` options. This allows specifying ranges of

allowed GIDs for pods/containers without forcing a default GID the way `MustRunAs` does. This means that a container to which such a policy applies to won't use any fsGroup/supplementalGroup GID if not explicitly specified, yet a specified GID must still fall in the GID range according to the policy. (#65135, @stlaz)

- Upgrade Stackdriver Logging Agent addon image to 0.6-1.6.0-1 to use Fluentd v1.2. This provides nanoseconds timestamp granularity for logs. (#70954, @qingling128)
- When the BoundServiceAccountTokenVolumes Alpha feature is enabled, ServiceAccount volumes now use a projected volume source and their names have the prefix "kube-api-access". (#69848, @mikedanese)
- Raw block volume support is promoted to beta, and enabled by default. This is accessible via the `volumeDevices` container field in pod specs, and the `volumeMode` field in persistent volume and persistent volume claims definitions. (#71167, @msau42)
- TokenReview now supports audience validation of tokens with audiences other than the kube-apiserver. (#62692, @mikedanese)
- StatefulSet is supported in `kubectl autoscale` command (#71103, @Pingan2017)
- Kubernetes v1.13 moves support for Container Storage Interface to GA. As part of this move Kubernetes now supports CSI v1.0.0 and deprecates support for CSI 0.3 and older releases. Older CSI drivers must be updated to CSI 1.0 and moved to the new kubelet plugin registration directory in order to work with Kubernetes 1.15+. (#71020, @saad-ali)
- Added option to create CSRs instead of certificates for kubeadm init phase certs and kubeadm alpha certs renew (#70809, @liztio)
- Added a kubelet socket which serves an grpc service containing the devices used by containers on the node. (#70508, @dashpole)
- Added DynamicAuditing feature which allows for the configuration of audit webhooks through the use of an AuditSink API object. (#67257, @pbarker)
- The kube-apiserver's healthz now takes in an optional query parameter which allows you to disable health checks from causing healthz failures. (#70676, @logicalhan)
- Introduced support for running a nodelocal dns cache. It is disabled by default, can be enabled by setting KUBE_ENABLE_NODELOCAL_DNS=true (#70555, @prameshj)
- Added readiness gates in extended output for pods (#70775, @freehan)
- Added `Ready` column and improve human-readable output of Deployments and StatefulSets (#70466, @Pingan2017)
- Added `kubelet_container_log_size_bytes` metric representing the log file size of a container. (#70749, @brancz)
- NodeLifecycleController: When node lease feature is enabled, node lease will be deleted when the corresponding node is deleted. (#70034, @wangzhen127)
- GCERegionalPersistentDisk feature is GA now! (#70716, @jingxu97)

- Added secure port 10259 to the kube-scheduler (enabled by default) and deprecate old insecure port 10251. Without further flags self-signed certs are created on startup in memory. (#69663, @sttts)

Release Notes From SIGs

SIG API Machinery

- The OwnerReferencesPermissionEnforcement admission plugin now checks authorization for the correct scope (namespaced or cluster-scoped) of the owner resource type. Previously, it always checked permissions at the same scope as the child resource. (#70389, @caesarxuchao)
- OpenAPI spec now correctly marks delete request's body parameter as optional (#70032, @iamneha)
- The rules for incrementing `metadata.generation` of custom resources changed: (#69059, @caesarxuchao)
  - If the custom resource participates the spec/status convention, the metadata.generation of the CR increments when there is any change, except for the changes to the metadata or the changes to the status.
  - If the custom resource does not participate the spec/status convention, the metadata.generation of the CR increments when there is any change to the CR, except for changes to the metadata.
  - A custom resource is considered to participate the spec/status convention if and only if the "CustomResourceSubresources" feature gate is turned on and the CRD has `.spec.subresources.status={}`.
- Fixed patch/update operations on multi-version custom resources (#70087, @liggitt)
- Reduced memory utilization of admission webhook metrics by removing resource related labels. (#69895, @jpbetz)
- Kubelet can now parse PEM file containing both TLS certificate and key in arbitrary order. Previously key was always required to be first. (#69536, @awly)
- Code-gen: Removed lowercasing for project imports (#68484, @jsturtevant)
- Fixed client cert setup in delegating authentication logic (#69430, @DirectXMan12)
- OpenAPI spec and API reference now reflect dryRun query parameter for POST/PUT/PATCH operations (#69359, @roycaihw)
- Fixed the sample-apiserver so that its BanFlunder admission plugin can be used. (#68417, @MikeSpreitzer)
- APIService availability related to networking glitches are corrected faster (#68678, @deads2k)
- Fixed an issue with stuck connections handling error responses (#71412, @liggitt)

- apiserver: fixed handling and logging of panics in REST handlers (#71076, @liggitt)
- kube-controller-manager no longer removes ownerReferences from ResourceQuota objects (#70035, @liggitt)
- "unfinished_work_microseconds" is added to the workqueue metrics; it can be used to detect stuck worker threads. (kube-controller-manager runs many workqueues.) (#70884, @lavalamp)
- Timeouts set in ListOptions for clients are also be respected locally (#70998, @deads2k)
- Added support for CRD conversion webhook (#67006, @mbohlool)
- client-go: fixed sending oversized data frames to spdystreams in remotecommand.NewSPDYExecutor (#70999, @liggitt)
- Fixed missing flags in `-controller-manager --help`. (#71298, @stewart-yu)
- Fixed missing flags in `kube-apiserver --help`. (#70204, @imjching)
- The caBundle and service fields in admission webhook API objects now correctly indicate they are optional (#70138, @liggitt)
- Fixed an issue with stuck connections handling error responses (#71419, @liggitt)
- kube-controller-manager and cloud-controller-manager now hold generated serving certificates in-memory unless a writeable location is specified with –cert-dir (#69884, @liggitt)
- CCM server will not listen insecurely if secure port is specified (#68982, @aruneli)
- List operations against the API now return internal server errors instead of partially complete lists when a value cannot be transformed from storage. The updated behavior is consistent with all other operations that require transforming data from storage such as watch and get. (#69399, @mikedanese)

SIG Auth

- API Server can be configured to reject requests that cannot be audit-logged. (#65763, @x13n)
- Go clients created from a kubeconfig that specifies a TokenFile now periodically reload the token from the specified file. (#70606, @mikedanese)
- When `--rotate-server-certificates` is enabled, kubelet will no longer request a new certificate on startup if the current certificate on disk is satisfactory. (#69991, @agunnerson-ibm)
- Added dynamic audit configuration api (#67547, @pbarker)
- Added ability to control primary GID of containers through Pod Spec and PodSecurityPolicy (#67802, @krmayankk)
- kube-apiserver: the `NodeRestriction` admission plugin now prevents kubelets from modifying `Node` labels prefixed with `node-restriction.kubernetes.io/`. The `node-restriction.kubernetes.io/` label prefix is reserved for clus-

ter administrators to use for labeling `Node` objects to target workloads to nodes in a way that kubelets cannot modify or spoof. (#68267, @liggitt)

SIG Autoscaling

- Updated Cluster Autoscaler version to 1.13.0. See the Release Notes for more information. (#71513, @losipiuk)

SIG AWS

- `service.beta.kubernetes.io/aws-load-balancer-internal` now supports true and false values, previously it only supported non-empty strings (#69436, @mcrute)
- Added `service.beta.kubernetes.io/aws-load-balancer-security-groups` annotation to set the security groups to the AWS ELB to be the only ones specified in the annotation in case this is present (does not add `0.0.0.0/0`). (#62774, @Raffo)

SIG Azure

- Ensured orphan public IPs on Azure deleted when service recreated with the same name. (#70463, @feiskyer)
- Improved Azure instance metadata handling by adding caches. (#70353, @feiskyer)
- Corrected check for non-Azure managed nodes with the Azure cloud provider (#70135, @marc-sensenich)
- Fixed azure disk attach/detach failed forever issue (#71377, @andyzhangx)
- DisksAreAttached –> getNodeDataDisks–> GetDataDisks –> getVirtualMachine –> vmCache.Get (#71495, @andyzhangx)

SIG CLI

- `kubectl apply` can now change a deployment strategy from rollout to recreate without explicitly clearing the rollout-related fields (#70436, @liggitt)
- The `kubectl plugin list` command now displays discovered plugin paths in the same order as they are found in a user's PATH variable. (#70443, @juanvallejo)
- `kubectl get` no longer exits before printing all of its results if an error is found (#70311, @juanvallejo)
- Fixed a runtime error occuring when sorting the output of `kubectl get` with empty results (#70740, @mfpierre)

- kubectl: support multiple arguments for cordon/uncordon and drain (#68655, @goodluckbot)
- Fixed ability for admin/edit/view users to see controller revisions, needed for kubectl rollout commands (#70699, @liggitt)
- `kubectl rollout undo` now returns errors when attempting to rollback a deployment to a non-existent revision (#70039, @liggitt)
- kubectl run now generates apps/v1 deployments by default (#71006, @liggitt)
- The "kubectl cp" command now supports path shortcuts (../) in remote paths. (#65189, @juanvallejo)
- Fixed dry-run output in kubectl apply –prune (#69344, @zegl)
- The kubectl wait command must handle when a watch returns an error vs closing by printing out the error and retrying the watch. (#69389, @smarterclayton)
- kubectl: support multiple arguments for cordon/uncordon and drain (#68655, @goodluckbot)

SIG Cloud Provider

- Added deprecation warning for all cloud providers (#69171, @andrewsykim)

SIG Cluster Lifecycle

- kubeadm: Updates version of CoreDNS to 1.2.6 (#70796, @detiber)
- kubeadm: Validate kubeconfig files in case of external CA mode. (#70537, @yagonobre)
- kubeadm: The writable config file option for extra volumes is renamed to readOnly with a reversed meaning. With readOnly defaulted to false (as in pod specs). (#70495, @rosti)
- kubeadm: Multiple API server endpoints support upon join is removed as it is now redundant. (#69812, @rosti)
- `kubeadm reset` now cleans up custom etcd data path (#70003, @yagonobre)
- kubeadm: Fixed unnecessary upgrades caused by undefined order of Volumes and VolumeMounts in manifests (#70027, @bart0sh)
- kubeadm: Fixed node join taints. (#69846, @andrewrynhard)
- Fixed cluster autoscaler addon permissions so it can access batch/job. (#69858, @losipiuk)
- kubeadm: JoinConfiguration now houses the discovery options in a nested Discovery structure, which in turn has a couple of other nested structures to house more specific options (BootstrapTokenDiscovery and FileDiscovery) (#67763, @rosti)

- kubeadm: Fixed a possible scenario where kubeadm can pull much newer control-plane images (#69301, @neolit123)
- kubeadm now allows mixing of init/cluster and join configuration in a single YAML file (although a warning gets printed in this case). (#69426, @rosti)
- kubeadm: Added a `v1beta1` API. (#69289, @fabriziopandini)
- kubeadm init correctly uses `--node-name` and `--cri-socket` when `--config` option is also used (#71323, @bart0sh)
- kubeadm: Always pass spec.nodeName as `--hostname-override` for kube-proxy (#71283, @Klaven)
- `kubeadm join` correctly uses `--node-name` and `--cri-socket` when `--config` option is also used (#71270, @bart0sh)
- kubeadm now supports the `--image-repository` flag for customizing what registry to pull images from (#71135, @luxas)
- kubeadm: The writable config file option for extra volumes is renamed to readOnly with a reversed meaning. With readOnly defaulted to false (as in pod specs). (#70495, @rosti)
- kubeadm: Multiple API server endpoints support upon join is removed as it is now redundant. (#69812, @rosti)
- kubeadm: JoinConfiguration now houses the discovery options in a nested Discovery structure, which in turn has a couple of other nested structures to house more specific options (BootstrapTokenDiscovery and FileDiscovery) (#67763, @rosti)
- kubeadm: Added a `v1beta1` API. (#69289, @fabriziopandini)
- kubeadm: Use `advertise-client-urls` instead of `listen-client-urls` as and `etcd-servers` options for apiserver. (#69827, @tomkukral)
- Kubeadm now respects the custom image registry configuration across joins and upgrades. Kubeadm passes the custom registry to the kubelet for a custom pause container. (#70603, @chuckha)
- `kubeadm reset` now outputs instructions about manual iptables rules cleanup. (#70874, @rdodev)
- kubeadm: remove the AuditPolicyConfiguration feature gate (#70807, @Klaven)
- kubeadm pre-pulls Etcd image only if external Etcd is not used and (#70743, @bart0sh)
- kubeadm: UnifiedControlPlaneImage is replaced by UseHyperKubeImage boolean value. (#70793, @rosti)
- For kube-up and derived configurations, CoreDNS will honor master taints, for consistency with kube-dns behavior. (#70868, @justinsb)
- Recognize newer docker versions without -ce/-ee suffix: 18.09.0 (#71001, @thomas-riccardi)
- Any external provider should be aware the cloud-provider interface should be imported from :- (#68310, @cheftako)
- Fixed 'kubeadm upgrade' infinite loop waiting for pod restart (#69886, @bart0sh)
- Bumped addon-manager to v8.8 (#69337, @MrHohn)

- GCE: Filter out spammy audit logs from cluster autoscaler. (#70696, @loburm)
- GCE: Enable by default audit logging truncating backend. (#68288, @loburm)
- Bumped cluster-proportional-autoscaler to 1.3.0 (#69338, @MrHohn)
- Updated defaultbackend to v1.5 (#69334, @bowei)

SIG GCP

- Added tolerations for Stackdriver Logging and Metadata Agents. (#69737, @qingling128)
- Enabled insertId generation, and updated Stackdriver Logging Agent image to 0.5-1.5.36-1-k8s. This help reduce log duplication and guarantee log order. (#68920, @qingling128)
- Updated crictl to v1.12.0 (#69033, @feiskyer)

SIG Network

- Corrected family type (inet6) for ipsets in ipv6-only clusters (#68436, @uablrek)
- kube-proxy argument `hostname-override` can be used to override hostname defined in the configuration file (#69340, @stevesloka)
- CoreDNS correctly implements DNS spec for Services with externalNames that look like IP addresses. Kube-dns does not follow the spec for the same case, resulting in a behavior change when moving from Kube-dns to CoreDNS. See: coredns/coredns#2324
- IPVS proxier now set net/ipv4/vs/conn_reuse_mode to 0 by default, which will highly improve IPVS proxier performance. (#71114, @Lion-Wei)
- CoreDNS is now version 1.2.6 (#70799, @rajansandeep)
- Addon configuration is introduced in the kubeadm config API, while feature flag CoreDNS is now deprecated. (#70024, @fabriziopandini)

SIG Node

- Fixed a bug in previous releases where a pod could be placed inside another pod's cgroup when specifying –cgroup-root (#70678, @dashpole)
- Optimized calculating stats when only CPU and Memory stats are returned from Kubelet stats/summary http endpoint. (#68841, @krzysztof-jastrzebski)
- kubelet now supports `log-file` option to write logs directly to a specific file (#70917, @dims)
- Do not detach volume if mount in progress (#71145, @gnufied)

- The runtimeHandler field on the RuntimeClass resource now accepts the empty string. (#69550, @tallclair)
- kube-apiserver: fixes `procMount` field incorrectly being marked as required in openapi schema (#69694, @jessfraz)

SIG OpenStack

- Fixed cloud-controller-manager crash when using OpenStack provider and PersistentVolume initializing controller (#70459, @mvladev)

SIG Release

- Use debian-base instead of busybox as base image for server images (#70245, @ixdy)
- Images for cloud-controller-manager, kube-apiserver, kube-controller-manager, and kube-scheduler now contain a minimal /etc/nsswitch.conf and should respect /etc/hosts for lookups (#69238, @BenTheElder)

SIG Scheduling

- Added metrics for volume scheduling operations (#59529, @wackxu)
- Improved memory use and performance when processing large numbers of pods containing tolerations (#65350, @liggitt)
- Fixed a bug in the scheduler that could cause the scheduler to go to an infinite loop when all nodes in a zone are removed. (#69758, @bsalamat)
- Clear pod binding cache on bind error to make sure stale pod binding cache will not be used. (#71212, @cofyc)
- Fixed a scheduler panic due to internal cache inconsistency (#71063, @Huang-Wei)
- Report kube-scheduler unhealthy if leader election is deadlocked. (#71085, @bsalamat)
- Fixed a potential bug that scheduler preempts unnecessary pods. (#70898, @Huang-Wei)

SIG Storage

- Fixed CSI volume limits not showing up in node's capacity and allocatable (#70540, @gnufied)
- CSI drivers now have access to mountOptions defined on the storage class when attaching volumes. (#67898, @bswartz)
- change default azure file mount permission to 0777 (#69854, @andyzhangx)
- Fixed subpath in containerized kubelet. (#69565, @jsafrane)
- Fixed panic on iSCSI volume tear down. (#69140, @jsafrane)

- CSIPersistentVolume feature, i.e. PersistentVolumes with CSIPersistentVolumeSource, is GA. (#69929, @jsafrane)
- Fixed CSIDriver API object to allow missing fields. (#69331, @jsafrane)
- Flex volume plugins now support expandvolume (to increase underlying volume capacity) and expanfs (resize filesystem) commands that Flex plugin authors can implement to support expanding in use Flex PersistentVolumes (#67851, @aniket-s-kulkarni)
- Enabled AttachVolumeLimit feature (#69225, @gnufied)
- The default storage class annotation for the storage addons has been changed to use the GA variant (#68345, @smelchior)
- GlusterFS PersistentVolumes sources can now reference endpoints in any namespace using the `spec.glusterfs.endpointsNamespace` field. Ensure all kubelets are upgraded to 1.13+ before using this capability. (#60195, @humblec)
- Fixed GetVolumeLimits log flushing issue (#69558, @andyzhangx)
- The `MountPropagation` feature is unconditionally enabled in v1.13, and can no longer be disabled. (#68230, @bertinatto)

SIG Windows

- `kubelet --system-reserved` and `--kube-reserved` are supported now on Windows nodes (#69960, @feiskyer)
- Windows runtime endpoints is now switched to `npipe:////./pipe/dockershim` from `tcp://localhost:3735`. (#69516, @feiskyer)
- Fixed service issues with named targetPort for Windows (#70076, @feiskyer)
- Handle Windows named pipes in host mounts. (#69484, @ddebroy)
- Fixed inconsistency in windows kernel proxy when updating HNS policy. (#68923, @delulu)

External Dependencies

- Default etcd server is unchanged at v3.2.24 since Kubernetes 1.12. (#68318)
- The list of validated docker versions remain unchanged at 1.11.1, 1.12.1, 1.13.1, 17.03, 17.06, 17.09, 18.06 since Kubernetes 1.12. (#68495)
- The default Go version was updated to 1.11.2. (#70665)
- The minimum supported Go version was updated to 1.11.2 (#69386)
- CNI is unchanged at v0.6.0 since Kubernetes 1.10 (#51250)
- CSI is updated to 1.0.0. Pre-1.0.0 API support is now deprecated. (#71020])
- The dashboard add-on has been updated to v1.10.0. (#68450)
- Heapster remains at v1.6.0-beta, but is now retired in Kubernetes 1.13 (#67074)

- Cluster Autoscaler has been upgraded to v1.13.0 (#71513)
- kube-dns is unchanged at v1.14.13 since Kubernetes 1.12 (#68900)
- Influxdb is unchanged at v1.3.3 since Kubernetes 1.10 (#53319)
- Grafana is unchanged at v4.4.3 since Kubernetes 1.10 (#53319)
- Kibana has been upgraded to v6.3.2. (#67582)
- CAdvisor has been updated to v0.32.0 (#70964)
- fluentd-gcp-scaler has been updated to v0.5.0 (#68837)
- Fluentd in fluentd-elasticsearch is unchanged at v1.2.4 since Kubernetes 1.11 (#67434)
- fluentd-elasticsearch has been updated to v2.2.1 (#68012)
- The fluent-plugin-kubernetes_metadata_filter plugin in fluentd-elasticsearch is unchanged at 2.0.0 since Kubernetes 1.12 (#67544)
- fluentd-gcp has been updated to v3.2.0 (#70954)
- OIDC authentication is unchanged at coreos/go-oidc v2 since Kubernetes 1.10 (#58544)
- Calico was updated to v3.3.1 (#70932)
- Upgraded crictl on GCE to v1.12.0 (#69033)
- CoreDNS has been updated to v1.2.6 (#70799)
- event-exporter has been updated to v0.2.3 (#67691)
- Es-image remains unchanged at Elasticsearch 6.3.2 since Kubernetes 1.12 (#67484)
- metrics-server remains unchanged at v0.3.1 since Kubernetes 1.12 (#68746)
- GLBC remains unchanged at v1.2.3 since Kubernetes 1.12 (#66793)
- Ingress-gce remains unchanged at v1.2.3 since Kubernetes 1.12 (#66793)
- ip-masq-agen remains unchanged at v2.1.1 since Kubernetes 1.12 (#67916)
- v1.13.0-rc.2
- v1.13.0-rc.1
- v1.13.0-beta.2
- v1.13.0-beta.1
- v1.13.0-alpha.3
- v1.13.0-alpha.2
- v1.13.0-alpha.1

v1.13.0-rc.2

Documentation

Downloads for v1.13.0-rc.2

filename	sha512 hash
kubernetes.tar.gz	`12fbaf943ae72711cd93c9955719ec1773a229dbb8f86a44fcda179229beb82add4dc1a5`

filename	sha512 hash
kubernetes-src.tar.gz	8e94f0fe73909610e85c201bb1ba4f66fd55ca2b4ded77217a4dfad2874d402cc1cc9420

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	ac555f5d1e6b88fa4de1e06e0a1ebd372582f97c526c938334a8c63f
kubernetes-client-darwin-amd64.tar.gz	2eae428a0e4bcb2237343d7ac1e431ccfc1f7037622bb3131ad8d48a
kubernetes-client-linux-386.tar.gz	89e671679b4516f184f7fd5ea0fe2a9ab0245fab34447625786bf558
kubernetes-client-linux-amd64.tar.gz	61f6513722e9c485300b822d6fc5998927bbffa18862d2d3f177a7c7
kubernetes-client-linux-arm.tar.gz	ef0e5fd4bf2074dfd3cf54d45307550273695906baca3533a9d23424
kubernetes-client-linux-arm64.tar.gz	d34bb9ce9bfe2a5375fd58920e63b4eef818348719dba460f3583843
kubernetes-client-linux-ppc64le.tar.gz	4dc4e4a5e166e63360ba86e1278bbe75212ac7c3f60ba30425a1c565
kubernetes-client-linux-s390x.tar.gz	d27675f4753469cd5e31faed13a1ea9654c25d38b0d96c1340215fd2
kubernetes-client-windows-386.tar.gz	9d6e6de2d4a55eaeebd7fa6b861548e0768381d50838430722b56636
kubernetes-client-windows-amd64.tar.gz	30b2da5c015ef88b9efcf90bffe0498d367df7c126b65f2e878af263

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	8180f2b788249fe65f7f1d3ee431ac758ede29a6349db312afbee080ff
kubernetes-server-linux-arm.tar.gz	e9165284a0b82a9ab88dad05f43bfe1bebecad3bb1c7118475c3426e0b
kubernetes-server-linux-arm64.tar.gz	03797c021ebed3b08835e72eed405c57aaacce972bbbbf88bf49310efb
kubernetes-server-linux-ppc64le.tar.gz	ceb49af22e3b518f3ba27c1e7de28e577e2735175e84a6d203f1f8766e
kubernetes-server-linux-s390x.tar.gz	bee4752e8a52e217ae1ffcfbc263453c724de684b4d463d5ddb24a3a30

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	b368989bbb8ab4d29b51d5d4d71d073b0ceb39614c944859dcd14c330
kubernetes-node-linux-arm.tar.gz	404b7b74a1e0d0fed9088a7e9461e02cfd9a6992c554baa125b7a361a
kubernetes-node-linux-arm64.tar.gz	fa531b1675a778c572a2175fb1bed00e78dc589f638f2096b3b5c9d3c
kubernetes-node-linux-ppc64le.tar.gz	a7ecc1f63e632c1b4f9b312babd6882ec966420bf4f8346edf80495fc
kubernetes-node-linux-s390x.tar.gz	a7171ed95de943a0ac5a32da4458e8d4366eb1fadbe426bebc371d2bb
kubernetes-node-windows-amd64.tar.gz	8a3a71d142b99fb200c4c1c9c0fa4dc6a3b64a0b506dc37dc3d832a94

Changelog since v1.13.0-rc.1

Other notable changes

- Update Cluster Autoscaler version to 1.13.0. Release notes: https://github.com/kubernetes/autoscaler/releases/tag/cluster-autoscaler-1.13.0 (#71513, @losipiuk)
- fix detach azure disk issue due to dirty cache (#71495, @andyzhangx)

v1.13.0-rc.1

Documentation

Downloads for v1.13.0-rc.1

filename	sha512 hash
kubernetes.tar.gz	`1c047e4edcf3553a568679e6e5083988b06df9d938f299a9193c72ad96a9c439a1f47f98`
kubernetes-src.tar.gz	`d2fd47c38abd29a2037b9e2a3a958ec250e2c6ae77532f6e935a6422bd626485fd720932`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`44d0733359be5036953775e12fc1723e4c64452a24a8c3b522c8a624`
kubernetes-client-darwin-amd64.tar.gz	`2acd37ed234271b0ff9c30273261e4b127309a1bc91a006b7a07e1a9`
kubernetes-client-linux-386.tar.gz	`5fe07ea2f776086df0e9447b7e6b0863c5b3af71f5aff8e302087e24`
kubernetes-client-linux-amd64.tar.gz	`7541d5850d74156862e5fe00817bd954d2b49b2c0cf15abe5cde3440`
kubernetes-client-linux-arm.tar.gz	`122121d3e469b6e33cc3fd910b32a5a94b9d3479f0367c54fbc4e7f1`
kubernetes-client-linux-arm64.tar.gz	`5e3d415db4239f27461c4ea404903cfc762084d5c1e84f9ed8bc0325`
kubernetes-client-linux-ppc64le.tar.gz	`8651f4161569913b616695bdd1a41c4b177cbfb4773fbca649b3e979`
kubernetes-client-linux-s390x.tar.gz	`920b81f6bbc7e7d4fa2f9c61fbc6f529621f2f134dbbb0f407866ffd`
kubernetes-client-windows-386.tar.gz	`0d49277cb7c36e5538d4c1c0fd6e6a69da7cd73c226f5869b29fad1e`
kubernetes-client-windows-amd64.tar.gz	`34ae587e2d439f925d1e324d2bbff3a751bb73b18e98b13c93e5742e`

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	`7030ef7463bef0871e524a5233d23c5f8aee18ac92e96555910ddc7a89`
kubernetes-server-linux-arm.tar.gz	`ccd1f413ad357581a904d1ff67f3e376be7882bd72efb13657f8aa1191`
kubernetes-server-linux-arm64.tar.gz	`ff589f5b6c56713818edda8ae9b39b17dfbf34e881c09736f722de5d70`

filename	sha512 hash
kubernetes-server-linux-ppc64le.tar.gz	`f748985751bf403bc7b1f9160ce937cd2915552b27c3c79764a66789dc`
kubernetes-server-linux-s390x.tar.gz	`b3b0075948d72784defe94073dff251b79083aa46b4f2941902675766`

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	`01907a104c043607985053571183b7bdccf655f847d1dd9d8991cd2c4`
kubernetes-node-linux-arm.tar.gz	`dbf1801c456312698253767dd36b186fb4e503a03454cd16bba68a1ec`
kubernetes-node-linux-arm64.tar.gz	`15f3259370f1419fcc372a28faa9a3caae5f2c89ee76286c14ea62d61`
kubernetes-node-linux-ppc64le.tar.gz	`00dc7f5bd40d045baeb72d5dcfb302b8566aacc23cd7de1b877724e11`
kubernetes-node-linux-s390x.tar.gz	`2b80e4dffa0b8bdc0305d1263c06320918541f3a7b6519123752b89be`
kubernetes-node-windows-amd64.tar.gz	`600b442a1665e39621fce03ad07b162e2353cc8bc982cad849dab7e1c`

Changelog since v1.13.0-beta.2

Other notable changes

- CVE-2018-1002105: Fix critical security issue in kube-apiserver upgrade request proxy handler (#71411, @liggitt)
- Update Cluster Autoscaler version to 1.13.0-rc.2. Release notes: https://github.com/kubernetes/autoscaler/releases/tag/cluster-autoscaler-1.13.0-rc.2 (#71452, @losipiuk)
- Upgrade Stackdriver Logging Agent addon image to 0.6-1.6.0-1 to use Fluentd v1.2. This provides nanoseconds timestamp granularity for logs. (#70954, @qingling128)
- fixes a runtime error occuring when sorting the output of `kubectl get` with empty results (#70740, @mfpierre)
- fix azure disk attach/detach failed forever issue (#71377, @andyzhangx)
- Do not detach volume if mount in progress (#71145, @gnufied)

v1.13.0-beta.2

Documentation

Downloads for v1.13.0-beta.2

filename	sha512 hash
kubernetes.tar.gz	`e8607473e2b3946a3655fa895c2b7dee74818b4c2701047fee5343ab6b2f2aa3d97b19b1`

filename	sha512 hash
kubernetes-src.tar.gz	6ca15ad729a82b41587e1dbbd4e9ad5447e202e8e7ee8c01c411090031ee3feb83f0cc65

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	5727218280ea7c68350aa5cf04e3d3c346f97d462e3f60f5196e2735
kubernetes-client-darwin-amd64.tar.gz	3e3975a41da08135dc654a40acb86ce862b1f56a9361e0c38c9c99c5
kubernetes-client-linux-386.tar.gz	26cfa99fbe09b20ebe3d2aebb4d08f0f9f2661d5533b94daf6c83547
kubernetes-client-linux-amd64.tar.gz	42204953b02af81bb5f695c957aca9fa382609447ada5e3a9701da3e
kubernetes-client-linux-arm.tar.gz	c680c94699b0b319b654a4c1c0a9b7fc387c44fb22744f30049142b1
kubernetes-client-linux-arm64.tar.gz	aa997b3428979ba2652fd251c4c5ece87043472ebe2ee15d8a179e69
kubernetes-client-linux-ppc64le.tar.gz	684dfc462d84d3902e322535997e57f7874003ab17c41508c057bc7c
kubernetes-client-linux-s390x.tar.gz	ff98b3a23dfe436a12843eb388be9568cbc29c9328648a1d166518aa
kubernetes-client-windows-386.tar.gz	6897a0f59fb409526dae9c86680702f3d2a1dc68d145504ed2e98b05
kubernetes-client-windows-amd64.tar.gz	6ed67eecb2b79ace8d428cbd4d07ef7d52ba4e5b3b44eb59d46aff99

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	351292b217c1c49b5c0241da11b4be0929a5d1645bec7dd05051930df8
kubernetes-server-linux-arm.tar.gz	88f166a7b5a3f9d9c19a5b911adb6e8e4cac1a3323b83d681f13aaf7bb
kubernetes-server-linux-arm64.tar.gz	fb4868a939eca18de17e0b606d1ab127712e277e01c02ffa96138a5397
kubernetes-server-linux-ppc64le.tar.gz	47a4e8e96c1e8a8cc37eabd19194b9d174fa93c3feaf1384895f89c5c6
kubernetes-server-linux-s390x.tar.gz	4e0823d1da55a71f001fcb07511a7b3416641ea93bfbd56b1e1e435c0a

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	e21964063b80f52e387cd35826f3081ad0a3b62608d182e008b8b76f5
kubernetes-node-linux-arm.tar.gz	cb665911af59a1cf86e5d66a4cdc134dc412e9e479dd89fa0bbbaeb83
kubernetes-node-linux-arm64.tar.gz	c172126829aea38e2238af6b62035abad6ed08d041175b0bf99792b7c
kubernetes-node-linux-ppc64le.tar.gz	0367940078ea9b4d46778b8406840fd2925f612304b5fa5b675fc07d5
kubernetes-node-linux-s390x.tar.gz	74382ed862ae099b91ce6056b85b7ee4f075fbdb4e737a8448c92e20f
kubernetes-node-windows-amd64.tar.gz	9164c4eae920c727965caae046e1b2daabf4822e2dee2260697b22e52

Changelog since v1.13.0-beta.1

Other notable changes

- Fix missing flags in kube-apiserver –help. (#70204, @imjching)
- kubeadm init correctly uses –node-name and –cri-socket when –config option is also used (#71323, @bart0sh)
- API server flag `--experimental-encryption-provider-config` was renamed to `--encryption-provider-config`. The old flag is accepted with a warning but will be removed in 1.14. (#71206, @stlaz)
- Fix missing flags in *-controller-manager –help. (#71298, @stewart-yu)
- Clear pod binding cache on bind error to make sure stale pod binding cache will not be used. (#71212, @cofyc)
- kubeadm: always pass spec.nodeName as –hostname-override for kube-proxy (#71283, @Klaven)
- kubeadm join correctly uses –node-name and –cri-socket when –config option is also used (#71270, @bart0sh)
- apiserver can be configured to reject requests that cannot be audit-logged. (#65763, @x13n)
- Kubelet Device Plugin Registration directory changed from from `{kubelet_root_dir}/plugins/` to `{kubelet_root_dir}/plugins_registry/`. Any drivers (CSI or device plugin) that were using the old path must be updated to work with this version. (#70494, @RenaudWasTaken)
- When the BoundServiceAccountTokenVolumes Alpha feature is enabled, ServiceAccount volumes now use a projected volume source and their names have the prefix “kube-api-access”. (#69848, @mikedanese)

v1.13.0-beta.1

Documentation

Downloads for v1.13.0-beta.1

filename	sha512 hash
kubernetes.tar.gz	`78245b2357a5eeafd193d28f86655327edce7bbc4da142c826eba5f5c05a624cd30b2551`
kubernetes-src.tar.gz	`880c5a8b16215bc58b30792247470304802 0b38be1d41672425cd07bdcf0626a88f04a08`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`0f804c77ef6122b4b6586a507179fe0f1a383752342b3e5575e09223`

filename	sha512 hash
kubernetes-client-darwin-amd64.tar.gz	0bdbd8003bcbecb4494b4778411e7d057067e78a99a7e8e8e45a3982
kubernetes-client-linux-386.tar.gz	522795df77ff8543251232863cb36fe2d501671e04a5279a112aa3ff
kubernetes-client-linux-amd64.tar.gz	b6481bae237e6971f7b9cc039d3b7e62d49ddd48d52dd979432fa031
kubernetes-client-linux-arm.tar.gz	45b8fa2557bb742a8ce16e0a69fa64fe898509418c6f9099a24bf1ab
kubernetes-client-linux-arm64.tar.gz	475b823a5e2c4c6e1bc49f35fbef45d1fc6e6279f5335762bad05d0f
kubernetes-client-linux-ppc64le.tar.gz	bc289b249051e9918f8f842bb98bf4d0b8951709fe5b65c2185f04b7
kubernetes-client-linux-s390x.tar.gz	0935e0ad23a61d570de087e72f22bc3da2a34c19bb5aea0ab342f916
kubernetes-client-windows-386.tar.gz	4833425ff040983b841722a00edd2cfa56f85099658ae04890c4e226
kubernetes-client-windows-amd64.tar.gz	156a5328834055f7b9732c762cc917cfdbf2d2fc67dd80ba89ae7dcb

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	9435be5cced10252954be579408e2a253eb51dd7b649417f1e91679bce
kubernetes-server-linux-arm.tar.gz	8dc3d4a0c09830831efd77fdf193ed9ccc1247bd981b4811192cac38cf
kubernetes-server-linux-arm64.tar.gz	f2549f87f21ea44c5d776a706c59bb2ea61d7f2cca304850aa6ad5b09c
kubernetes-server-linux-ppc64le.tar.gz	0c0671eaf7cf7262c95411930311bb4610f89583431738149f0ee7f8f6
kubernetes-server-linux-s390x.tar.gz	ff24909b0b044924d241d6aeac9e9b4f0696c0ca7e973d56a874b02b61

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	235b1c4348b5779ca71a5f63121ff6a162db02bb24b4d815ec73412af
kubernetes-node-linux-arm.tar.gz	cd23813419a74983bdd3a3104e20684e947ef7302dcfa1802132439b2
kubernetes-node-linux-arm64.tar.gz	fb6283dae828f8d9275a05c6a4ea27bc1136e8e8253b5ddac52c82548
kubernetes-node-linux-ppc64le.tar.gz	f910922d422b65f6b6a8d7762a23048991695496c0fc07c3dc4f1a81e
kubernetes-node-linux-s390x.tar.gz	d895fed57caf038afe0087ff44d2adfdd8955d18135adad9935952702
kubernetes-node-windows-amd64.tar.gz	4bc09e54935d2cb4d2bad7db06831d040cc03906d8934575932ed6eab

Changelog since v1.13.0-alpha.3

Action Required

- ACTION REQUIRED: The Node.Status.Volumes.Attached.DevicePath fields is deprecated for CSI volumes and will be unset in a future release (#71095, @msau42)

Other notable changes

- Raw block volume support is promoted to beta, and enabled by default. This is accessible via the `volumeDevices` container field in pod specs, and the `volumeMode` field in persistent volume and persistent volume claims definitions. (#71167, @msau42)
- Fix a scheduler panic due to internal cache inconsistency (#71063, @Huang-Wei)
- Fix a potential bug that scheduler preempts unnecessary pods. (#70898, @Huang-Wei)
- The API server encryption configuration file format has graduated to stable and moved to `apiVersion: apiserver.config.k8s.io/v1` and `kind: EncryptionConfiguration`. (#67383, @stlaz)
- kubelet now supports `log-file` option to write logs directly to a specific file (#70917, @dims)
- kubeadm now supports the `--image-repository` flag for customizing what registry to pull images from (#71135, @luxas)
- timeouts set in ListOptions for clients will also be respected locally (#70998, @deads2k)
- IPVS proxier now set net/ipv4/vs/conn_reuse_mode to 0 by default, which will highly improve IPVS proxier performance. (#71114, @Lion-Wei)
- StatefulSet is supported in `kubectl autoscale` command (#71103, @Pingan2017)
- Report kube-scheduler unhealthy if leader election is deadlocked. (#71085, @bsalamat)
- apiserver: fixes handling and logging of panics in REST handlers (#71076, @liggitt)
- kubelets are no longer allowed to delete their own Node API object. Prior to 1.11, in rare circumstances related to cloudprovider node ID changes, kubelets would attempt to delete/recreate their Node object at startup. Kubelets older than 1.11 are not supported running against a v1.13+ API server. If an unsupported legacy kubelet encounters this situation, a cluster admin can remove the Node object: (#71021, @liggitt) * `kubectl delete node/<nodeName>`
  - or grant self-deletion permission explicitly:
    * `kubectl create clusterrole self-deleting-nodes --verb=delete --resource=nodes`
    * `kubectl create clusterrolebinding self-deleting-nodes --clusterrole=self-deleting-nodes --group=system:nodes`
- Kubernetes v1.13 moves support for Container Storage Interface to GA. As part of this move Kubernetes now supports CSI v1.0.0 and drops support for CSI 0.3 and older releases. Older CSI drivers must be updated to CSI 1.0 in order to work with Kubernetes 1.13+. (#71020, @saad-ali)
- Remove deprecated kubectl command aliases 'run-container' (#70728,

@Pingan2017)

- kubeadm: enable strict unmarshaling of YAML configuration files and show warnings for unknown and duplicate fields. (#70901, @neolit123)
- For kube-up and derived configurations, CoreDNS will honor master taints, for consistency with kube-dns behavior. (#70868, @justinsb)
- CoreDNS is now version 1.2.6 (#70799, @rajansandeep)
- kubeadm: Use `advertise-client-urls` instead of `listen-client-urls` as and `etcd-servers` options for apiserver. (#69827, @tomkukral)
- Add option to create CSRs instead of certificates for kubeadm init phase certs and kubeadm alpha certs renew (#70809, @liztio)
- Add a kubelet socket which serves an grpc service containing the devices used by containers on the node. (#70508, @dashpole)
- kube-apiserver: the `NodeRestriction` admission plugin now prevents kubelets from modifying `Node` labels prefixed with `node-restriction.kubernetes.io/`. The `node-restriction.kubernetes.io/` label prefix is reserved for cluster administrators to use for labeling `Node` objects to target workloads to nodes in a way that kubelets cannot modify or spoof. (#68267, @liggitt)
  - kubelet: it is now deprecated to use the `--node-labels` flag to set `kubernetes.io/` and `k8s.io/`-prefixed labels other than the following labels:
    * `kubernetes.io/hostname`
    * `kubernetes.io/instance-type`
    * `kubernetes.io/os`
    * `kubernetes.io/arch`
    * `beta.kubernetes.io/instance-type`
    * `beta.kubernetes.io/os`
    * `beta.kubernetes.io/arch`
    * `failure-domain.kubernetes.io/zone`
    * `failure-domain.kubernetes.io/region`
    * `failure-domain.beta.kubernetes.io/zone`
    * `failure-domain.beta.kubernetes.io/region`
    * `[*.]kubelet.kubernetes.io/*`
    * `[*.]node.kubernetes.io/*`
  - Setting other `kubernetes.io/`- and `k8s.io/`-prefixed labels using the `--node-labels` flag will produce a warning in v1.13, and be disallowed in v1.15. Setting labels that are not prefixed with `kubernetes.io/` or `k8s.io/` is still permitted.
- Adds DynamicAuditing feature which allows for the configuration of audit webhooks through the use of an AuditSink API object. (#67257, @pbarker)
- The Kubelet plugin registration mechanism used by device plugins and CSI plugins is now GA (#70559, @vladimirvivien)
- CSIPersistentVolume feature, i.e. PersistentVolumes with CSIPersistentVolumeSource, is GA. (#69929, @jsafrane)
  - CSIPersistentVolume feature gate is now deprecated and will be removed according to deprecation policy.

- kubectl: support multiple arguments for cordon/uncordon and drain (#68655, @goodluckbot)
- The kube-apiserver's healthz now takes in an optional query parameter which allows you to disable health checks from causing healthz failures. (#70676, @logicalhan)
- client-go: fixes sending oversized data frames to spdystreams in remotecommand.NewSPDYExecutor (#70999, @liggitt)
- kube-controller-manager no longer removes ownerReferences from ResourceQuota objects (#70035, @liggitt)
- Introduces support for running a nodelocal dns cache. It is disabled by default, can be enabled by setting KUBE_ENABLE_NODELOCAL_DNS=true (#70555, @prameshj)
  - An ip address is required for the cache instance to listen for requests on, default is a link local ip address of value 169.254.20.10
- Fix dry-run output in kubectl apply –prune (#69344, @zegl)
- kubectl run now generates apps/v1 deployments by default (#71006, @liggitt)
- `kubeadm reset` now outputs instructions about manual iptables rules cleanup. (#70874, @rdodev)
- Recognize newer docker versions without -ce/-ee suffix: 18.09.0 (#71001, @thomas-riccardi)
- "unfinished_work_microseconds" is added to the workqueue metrics; it can be used to detect stuck worker threads. (kube-controller-manager runs many workqueues.) (#70884, @lavalamp)
- add readiness gates in extended output for pods (#70775, @freehan)
- add `Ready` column and improve human-readable output of Deployments and StatefulSets (#70466, @Pingan2017)
- Kubeadm now respects the custom image registry configuration across joins and upgrades. Kubeadm passes the custom registry to the kubelet for a custom pause container. (#70603, @chuckha)
- kubeadm: deprecate the DynamicKubeletConfig feature gate. The functionality is still accessible by using the kubeadm alpha kubelet enable-dynamic command. (#70849, @yagonobre)
- Add `kubelet_container_log_size_bytes` metric representing the log file size of a container. (#70749, @brancz)
- kubeadm: remove the AuditPolicyConfiguration feature gate (#70807, @Klaven)
- Kubeadm: attributes for join –control-plane workflow are now grouped into a dedicated JoinControlPlane struct (#70870, @fabriziopandini)
- Addon configuration is introduced in the kubeadm config API, while feature flag CoreDNS is now deprecated. (#70024, @fabriziopandini)
- Fixes ability for admin/edit/view users to see controller revisions, needed for kubectl rollout commands (#70699, @liggitt)
- kubeadm pre-pulls Etcd image only if external Etcd is not used and (#70743, @bart0sh)
  - –etcd-upgrade=false is not specified

- Add support for CRD conversion webhook (#67006, @mbohlool)
- Delete node lease if the corresponding node is deleted (#70034, @wangzhen127)
- In a future release the kubectl convert command will be deprecated. (#70820, @seans3)
- kubeadm: UnifiedControlPlaneImage is replaced by UseHyperKubeImage boolean value. (#70793, @rosti)
- kubeadm v1beta1 API: InitConfiguration.APIEndpoint has been renamed to .LocalAPIEndpoint (#70761, @luxas)
- Breaking change: CSINodeInfo split into Spec and Status. New fields Available and VolumePluginMechanism added to CSINodeInfo csi-api object. CSIDriverInfo no longer deleted on Driver uninstallation, instead Available flag is set to false. (#70515, @davidz627)
- GCERegionalPersistentDisk feature is GA now! (#70716, @jingxu97)
- Add secure port 10259 to the kube-scheduler (enabled by default) and deprecate old insecure port 10251. Without further flags self-signed certs are created on startup in memory. (#69663, @sttts)
- `--feature-gates` argument has been removed from the `kubeadm join` command. Feature gates will be retrieved from the cluster configuration during the join process. (#70755, @ereslibre)
- [kubeadm] Updates version of CoreDNS to 1.2.6 (#70796, @detiber)
- kubelet: When node lease feature is enabled, kubelet reports node status to api server only if there is some change or it didn't report over last report interval. (#69753, @wangzhen127)
- Self hosted is no longer supported in the standard workflow. The feature flags have been removed and your self hosted cluster is no longer able to upgrade via kubeadm. (#69878, @Klaven)
- vSphereVolume implements Raw Block Volume Support (#68761, @fanzhangio)
- [GCE] Filter out spammy audit logs from cluster autoscaler. (#70696, @loburm)
- CRD supports multi-version Schema, Subresources and AdditionalPrintColumns (NOTE that CRDs created prior to 1.13 populated the top-level additionalPrinterColumns field by default. To apply an update that changes to per-version additionalPrinterColumns, the top-level additionalPrinterColumns field must be explicitly set to null). (#70211, @roycaihw)
- Fixes a bug in previous releases where a pod could be placed inside another pod's cgroup when specifying –cgroup-root (#70678, @dashpole)
- Upgrade golang.org/x/net image to release-branch.go1.10 (#70663, @wenjiaswe)
- New addon in addon manager that automatically installs CSI CRDs if CSIDriverRegistry or CSINodeInfo feature gates are true. (#70193, @saad-ali)
- delegated authorization can now allow unrestricted access for `system:masters` like the main kube-apiserver (#70671, @deads2k)
- Update to use go1.11.2 (#70665, @cblecker)

- Add dns capabilities for Windows CNI plugins: (#67435, @feiskyer)
  - “dns” {
  - “servers”: [“10.0.0.10”],
  - “searches”: [“default.svc.cluster.local”,“svc.cluster.local”,“cluster.local”],
  - “options”: []
  - }
- The VolumeScheduling feature is GA. The VolumeScheduling feature gate is deprecated and will be removed in a future release. (#70673, @msau42)
- Go clients created from a kubeconfig that specifies a TokenFile now periodically reload the token from the specified file. (#70606, @mikedanese)
- kubeadm: validate kubeconfig files in case of external CA mode. (#70537, @yagonobre)
- kube-apiserver: `--audit-webhook-version` and `--audit-log-version` now default to `audit.k8s.io/v1` if unspecified (#70476, @charrywanganthony)
- kubeadm: timeoutForControlPlane is introduced as part of the API Server config, that controls the timeout for the wait for control plane to be up. Default value is 4 minutes. (#70480, @rosti)
- kubeadm: The writable config file option for extra volumes is renamed to readOnly with a reversed meaning. With readOnly defaulted to false (as in pod specs). (#70495, @rosti)
- remove retry operation on attach/detach azure disk (#70568, @andyzhangx)
- Fix CSI volume limits not showing up in node's capacity and allocatable (#70540, @gnufied)
- Flex volume plugins now support expandvolume (to increase underlying volume capacity) and expanfs (resize filesystem) commands that Flex plugin authors can implement to support expanding in use Flex PersistentVolumes (#67851, @aniket-s-kulkarni)
- kubeadm: Control plane component configs are separated into ClusterConfiguration sub-structs. (#70371, @rosti)
- The `MountPropagation` feature is unconditionally enabled in v1.13, and can no longer be disabled. (#68230, @bertinatto)
- add azure UltraSSD, StandardSSD disk type support (#70477, @andyzhangx)
- The OwnerReferencesPermissionEnforcement admission plugin now checks authorization for the correct scope (namespaced or cluster-scoped) of the owner resource type. Previously, it always checked permissions at the same scope as the child resource. (#70389, @caesarxuchao)
- Ensure orphan public IPs on Azure deleted when service recreated with the same name. (#70463, @feiskyer)
- `kubectl apply` can now change a deployment strategy from rollout to recreate without explicitly clearing the rollout-related fields (#70436, @liggitt)
- Fix cloud-controller-manager crash when using OpenStack provider and PersistentVolume initializing controller (#70459, @mvladev)

v1.13.0-alpha.3

Documentation

Downloads for v1.13.0-alpha.3

filename	sha512 hash
kubernetes.tar.gz	1d50cfd34306ace7354516125c45f8c546bba3ca5081af2b21969b535967d302821c06e7
kubernetes-src.tar.gz	bf097b99d7b9af15bc1d592ee3782da1e811d8eb68dc9ae9d287589ce9174d3743beaf51

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	77778ae2887eda52ee716fb0e843c17b2705b1284a67cdf53f91292e
kubernetes-client-darwin-amd64.tar.gz	b3399767df12b71ee4b7b30126bd8001a0c1396161eb7535d797fd58
kubernetes-client-linux-386.tar.gz	5ef0d318ff8da28c332ae25164e5a441272d2ee8ef2ac26438a47fe3
kubernetes-client-linux-amd64.tar.gz	1f429eae5b0b1e39b1d4d30e3220a82d0ae6672a6f4b34a05246c3ef
kubernetes-client-linux-arm.tar.gz	5583aecdc9b4a54a4aa904fc1de66400f50628969e31b5a63ab1d3b6
kubernetes-client-linux-arm64.tar.gz	2453b9100c06b11e8c424d59cfd1c5e111c22b596191a9cfb0b330d1
kubernetes-client-linux-ppc64le.tar.gz	4991ec4c19a82d50caed78bc8db51e7cdcd1f2896dfcaa45d84f347a
kubernetes-client-linux-s390x.tar.gz	c55f2802afb2e5d261bb26b6c396df8ebe6b95913ddab1e124cf177f
kubernetes-client-windows-386.tar.gz	df78465267e35ef078c3c0fd33f8898a9df26fbf411df3ed3283fbdc
kubernetes-client-windows-amd64.tar.gz	5b93fdaaa931ef8e24196e53c484f91ef9e50b7d11e1053ccb61b2d6

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	922a93ce677e686e594c11db75e1969c995b23062bba511bff4a43d3a5
kubernetes-server-linux-arm.tar.gz	5dd550b58dedf25df020e66f1526e80c50b46d2df3ddd241bd02b6ebf1
kubernetes-server-linux-arm64.tar.gz	3e1037e71d85a74cd5d40dd836bd442b2dcc457f8ccc8247e4537f3dec
kubernetes-server-linux-ppc64le.tar.gz	a89c46b558613ad09efe44a81574ad18157a787d1e9c5d09c98d3911b4
kubernetes-server-linux-s390x.tar.gz	47a68668e38ac1b8cb801f4bff3b15060cd88801f446ebfbf06125dbc9

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	74d5d46ac6ba336fa8aaf55d0a15860f6ebde2ff58d377ca93063593c
kubernetes-node-linux-arm.tar.gz	90372cb5270ffe6179d5d7efd3fff6aa029f73853805038fef1a6f926

filename	sha512 hash
kubernetes-node-linux-arm64.tar.gz	`09693303a1a8489d9599d32f7fbf549d18f31eb53671fa2ed342fe508`
kubernetes-node-linux-ppc64le.tar.gz	`cbdb3b9ffd9be524ec0b38d72b0545b6dd1b3b789747f41a661fd7cbe`
kubernetes-node-linux-s390x.tar.gz	`fc296b386bc03bf10773559118cd4a3d5be3d4c296f09748507fac812`
kubernetes-node-windows-amd64.tar.gz	`ae79c62fcb0654a62606d65cf131188d93e4a10787a862e7b0363269`

Changelog since v1.13.0-alpha.2

Other notable changes

- kubelet –system-reserved and –kube-reserved are supported now on Windows nodes (#69960, @feiskyer)
- CSI drivers now have access to mountOptions defined on the storage class when attaching volumes. (#67898, @bswartz)
- The `kubectl plugin list` command will now display discovered plugin paths in the same order as they are found in a user's PATH variable. (#70443, @juanvallejo)
- Handle Windows named pipes in host mounts. (#69484, @ddebroy)
- kubeadm: Multiple API server endpoints support upon join is removed as it is now redundant. (#69812, @rosti)
- OpenAPI spec marks delete request's body parameter as optional (#70032, @iamneha)
- kube-controller-manager and cloud-controller-manager now hold generated serving certificates in-memory unless a writeable location is specified with –cert-dir (#69884, @liggitt)
- Scheduler only activates unschedulable pods if node's scheduling related properties change. (#70366, @mlmhl)
- –api-audiences now defaults to the –service-account-issuer if the issuer is provided but the API audience is not. (#70308, @mikedanese)
- Refactor scheduler_test.go to use a fake k8s client. (#70290, @tossmilestone)
- `kubectl rollout undo` now returns errors when attempting to rollback a deployment to a non-existent revision (#70039, @liggitt)
  - `kubectl rollout undo` no longer uses the deprecated extensions/v1beta1 rollback API, which means that Events are no longer emitted when rolling back a deployment
- - The builtin system:csi-external-provisioner and system:csi-external-attacher cluster roles (#69868, @pohly)
  - are deprecated and will not be updated for deployments of CSI sidecar container versions >= 0.4.
  - Deployments with the current CSI sidecar containers have to provide their own RBAC
  - definitions. The reason is that the rules depend on how the sidecar containers are used,

  - which is defined by the deployment.
- Use debian-base instead of busybox as base image for server images (#70245, @ixdy)
- add support for projected volume in describe function (#70158, @WanLinghao)
- Speedup process lookup in /proc (#66367, @cpuguy83)
- Kubeadm reset now clean up custom etcd data path (#70003, @yagonobre)
- We changed when the `metadata.generation` of a custom resource (CR) increments. (#69059, @caesarxuchao)
  - If the CR participates the spec/status convention, the metadata.generation of the CR increments when there is any change, except for the changes to the metadata or the changes to the status.
  - If the CR does not participate the spec/status convention, the metadata.generation of the CR increments when there is any change to the CR, except for changes to the metadata.
  - A CR is considered to participate the spec/status convention if and only if the "CustomResourceSubresources" feature gate is turned on and the CRD has `.spec.subresources.status={}`.
- Improve Azure instance metadata handling by adding caches. (#70353, @feiskyer)
- adding cn-northwest-1 for AWS China Ningxia region (#70155, @pahud)
- "kubectl get" no longer exits before printing all of its results if an error is found (#70311, @juanvallejo)
- kubeadm now automatically creates a new stacked etcd member when joining a new control plane node (does not applies to external etcd) (#69486, @fabriziopandini)
- Critical pod annotation is deprecated. Pod priority should be used instead to mark pods as critical. (#70298, @bsalamat)
- Display the usage of ephemeral-storage when using `kubectl describe node` (#70268, @Pingan2017)
- Added functionality to enable br_netfilter and ip_forward for debian packages to improve kubeadm support for CRI runtime besides Docker. (#70152, @ashwanikhemani)
- Add regions ap-northeast-3 and eu-west-3 to the list of well known AWS regions. (#70252, @nckturner)
- Remove kube-controller-manager flag '–insecure-experimental-approve-all-kubelet-csrs-for-group'(deprecated in v1.7) (#69209, @Pingan2017)
- GCE/GKE load balancer health check default interval changes from 2 seconds to 8 seconds, unhealthyThreshold to 3. (#70099, @grayluck)
  - Health check parameters are configurable to be bigger than default values.
- The kubectl wait command must handle when a watch returns an error vs closing by printing out the error and retrying the watch. (#69389, @smarterclayton)
- Updates to use debian-iptables v11.0, debian-hyperkube-base 0.12.0, and

kube-addon-manager:v8.9. (#70209, @ixdy)
- Fixed patch/update operations on multi-version custom resources (#70087, @liggitt)
- When `--rotate-server-certificates` is enabled, kubelet will no longer request a new certificate on startup if the current certificate on disk is satisfactory. (#69991, @agunnerson-ibm)
- - Support for passing unknown provider names to the E2E test binaries is going to be deprecated. Use `--provider=skeleton` (no ssh access) or `--provider=local` (local cluster with ssh) instead. (#70141, @pohly)
- Add scheduler benchmark tests for PodAffinity and NodeAffinity. (#69898, @Huang-Wei)
- fix azure disk attachment error on Linux (#70002, @andyzhangx)

v1.13.0-alpha.2

Documentation

Downloads for v1.13.0-alpha.2

filename	sha512 hash
kubernetes.tar.gz	`cbe7ef29c7e7bbed82e173289f5f84d7a85ee4965cc5b7ccd16cf8236a3b8171bb8f5201`
kubernetes-src.tar.gz	`8b0b8e1b635cd849c2974d755fe174f0ce8fe8c690721d8ac6312683bbd2ca2c6f7eada3`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`fca661a5001e7f368374d0805f20910be24baa485bf4ae5d993185b9`
kubernetes-client-darwin-amd64.tar.gz	`d31dfea475981c7f7b758c7f201aa5b866db48d87942c79d0a12d464`
kubernetes-client-linux-386.tar.gz	`fecf8362c572fff48952fd2748ddcb9d375462cb484670cda4fda138`
kubernetes-client-linux-amd64.tar.gz	`136cb82ac94bcd791d56e997a948a7e1bee4af03bcc69ce9c835895c`
kubernetes-client-linux-arm.tar.gz	`e561c37895edef44614ecd59f497d393275ee62455b6269b169a8918`
kubernetes-client-linux-arm64.tar.gz	`c0d5eb49763e8bf50b5e8e3785c7889fecbd8bf7c0b3c18250fa894a`
kubernetes-client-linux-ppc64le.tar.gz	`a5a8c150af163e7c726662eeddfc3de8e43f123daaa100b8e82c9bc7`
kubernetes-client-linux-s390x.tar.gz	`fd162e0244e107f1892d79029f3452cdba84d8616ad1b15eebe197af`
kubernetes-client-windows-386.tar.gz	`e01fedec8f700e037bc43cb13bc916b85601cd1c9361a0f63fd27092`
kubernetes-client-windows-amd64.tar.gz	`d2601efcfa6a4ba8a017e9cac571fb454b21b7700a7b3f8e2fbabdd5`

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	4dda298d44bc309f250c067e9282eea37903838a140cf5abf6f861dca6
kubernetes-server-linux-arm.tar.gz	e9c3bdf60272399bc6f85a15bbc55cd69db389c223b275661ddcab4ae8
kubernetes-server-linux-arm64.tar.gz	d0a1701a34365f939799b6ea676129acdcfa1582bcf50e82a9751d9aaf
kubernetes-server-linux-ppc64le.tar.gz	1a23473960aaaa639e796020741b63c11dad8a93903926e80c871814b8
kubernetes-server-linux-s390x.tar.gz	293d0eb93e2ed641d0c1e26d58423670c04c307dddb034a9fc252043ab

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	e3cba71c2b2d151cdcc44937c1bea083ee0ceb829e7feb25cd37edd4c
kubernetes-node-linux-arm.tar.gz	28d7f1cf4fecdc72da7f5f19836cc06bb08182f8e8fb1641dc01e9299
kubernetes-node-linux-arm64.tar.gz	cfe22b11502cd857f0e277e7c1af08e6202f7ffc36f852c6154159bdc
kubernetes-node-linux-ppc64le.tar.gz	195a6785c49af419361a8901c99bb6613a6578a8eac5e8f08ec280776
kubernetes-node-linux-s390x.tar.gz	51f5b5ed47b50f5188d9e2f57b03555492d3e490494842247fa04fe81
kubernetes-node-windows-amd64.tar.gz	d2690d57cd485c0c7ebe425464ad59f2c7722870abd6f264ea7fae65a

Changelog since v1.13.0-alpha.1

Other notable changes

- Corrected family type (inet6) for ipsets in ipv6-only clusters (#68436, @uablrek)
- Corrects check for non-Azure managed nodes with the Azure cloud provider (#70135, @marc-sensenich)
- Windows runtime endpoints is now switched to 'npipe:////./pipe/dockershim' from 'tcp://localhost:3735'. (#69516, @feiskyer)
- The caBundle and service fields in admission webhook API objects now correctly indicate they are optional (#70138, @liggitt)
- The –service-account-api-audiences on kube-apiserver is deprecated in favor of –api-audiences. (#70105, @mikedanese)
- kubeadm: fix unnecessary upgrades caused by undefined order of Volumes and VolumeMounts in manifests (#70027, @bart0sh)
- kubeadm: Implemented preflight check to ensure that number of CPUs (#70048, @bart0sh)
  - on the master node is not less than required.
- Reduce memory utilization of admission webhook metrics by removing resource related labels. (#69895, @jpbetz)
- kubeadm: Introduce config print init/join-defaults that deprecate config print-defaults by decoupling init and join configs. (#69617, @rosti)
- Images based on debian-base no longer include the libsystemd0 package. This should have no user-facing impact. (#69995, @ixdy)

  - Additionally, the addon-manager image is updated to use kubectl v1.11.3.
- fix 'kubeadm upgrade' infinite loop waiting for pod restart (#69886, @bart0sh)
- add more logging for azure disk diagnostics (#70012, @andyzhangx)
- Fluentd: concatenate long logs (#68012, @desaintmartin)
- CoreDNS is now the default DNS server in kube-up deployments. (#69883, @chrisohaver)
- Optimizes calculating stats when only CPU and Memory stats are returned from Kubelet stats/summary http endpoint. (#68841, @krzysztof-jastrzebski)
- kubeadm: Fix node join taints. (#69846, @andrewrynhard)
- Opt out of chowning and chmoding from kubectl cp. (#69573, @bjhaid)
- support Azure premium file for azure file plugin (#69718, @andyzhangx)
- `TaintBasedEvictions` feature is promoted to beta. (#69824, @Huang-Wei)
- improves memory use and performance when processing large numbers of pods containing tolerations (#65350, @liggitt)
- Add dynamic audit configuration api (#67547, @pbarker)
- Promote resource limits priority function to beta (#69437, @ravisantoshgudimetla)
- Fix cluster autoscaler addon permissions so it can access batch/job. (#69858, @losipiuk)
- change default azure file mount permission to 0777 (#69854, @andyzhangx)
- kubeadm: JoinConfiguration now houses the discovery options in a nested Discovery structure, which in turn has a couple of other nested structures to house more specific options (BootstrapTokenDiscovery and FileDiscovery) (#67763, @rosti)
- Fix tests to use fsync instead of sync (#69755, @mrunalp)
- kube-proxy argument `hostname-override` can be used to override hostname defined in the configuration file (#69340, @stevesloka)
- kube-apiserver: the `--deserialization-cache-size` flag is no longer used, is deprecated, and will be removed in a future release (#69842, @liggitt)
- Add support for JSON patch in fake client (#69330, @vaikas-google)

v1.13.0-alpha.1

Documentation

Downloads for v1.13.0-alpha.1

filename	sha512 hash
kubernetes.tar.gz	9f8a34b54a22ea4d7925c2f8d0e0cb2e2005486b1ed89e594bc0100ec7202fc247b89c5c
kubernetes-src.tar.gz	a27a7c254d3677c823bd6fd1d0d5f9b1e78ccf807837173669a0079b0812a23444d646d8

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	d77d33c6d6357b99089f65e1c9ec3cabdcf526ec56e87bdee6b09a8c
kubernetes-client-darwin-amd64.tar.gz	5b4a586defa2ba0ea7c8893dedfe48cae52a2cd324bcb311a3877e27
kubernetes-client-linux-386.tar.gz	d50572fbb716393004ad2984a15043d2dfadedd16ae03a73fc856532
kubernetes-client-linux-amd64.tar.gz	12ab709e574228f170a2ee2686e18dcbfcf59f64599b2ab9047c2ed6
kubernetes-client-linux-arm.tar.gz	3a8c75b62cf9e6476417246d4aaeda5a13b74bc073444fc3649198b9
kubernetes-client-linux-arm64.tar.gz	0f5b5956850f11a826d59d226b6a22645ca1f63893cd33c17dfe004b
kubernetes-client-linux-ppc64le.tar.gz	06c60dd2e4e8d1ab45474a5b85345b4f644d0c1c66e167596c6c91bd
kubernetes-client-linux-s390x.tar.gz	4630e9e523beb02d8d3900c71b3306561c2d119d588399c93d578184
kubernetes-client-windows-386.tar.gz	0c0fcc9c492aceb00ff7fd3c10ba228c7bb10d6139b75ceecd8f8553
kubernetes-client-windows-amd64.tar.gz	3548a6d8618c6c7c8042ae8c3eb69654314392c46f839de24ab72d9f

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	9dbf2343ef9539b7d4d73949bcd9eef6f46ece59e97fa3390a0e695d0c
kubernetes-server-linux-arm.tar.gz	a985f3c302246df9bff4b927a2596d209c19fb2f245aa5cb5de189b6a9
kubernetes-server-linux-arm64.tar.gz	80d20df07e6a29b7aedccbd4e26c1c0565b2a1c3146e1a5bb2ebd2e8cf
kubernetes-server-linux-ppc64le.tar.gz	7d45ed3aa8b36e9e666b334ff3ed3de238caea34b4a92b5e1a61a6e722
kubernetes-server-linux-s390x.tar.gz	30698478fab2fe7daccac97917b0b21b018c194ec39b005728f8cddf77

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	4497d14ac81677b43f0b75a457890c1f3bb8745a39875f58d53c734be
kubernetes-node-linux-arm.tar.gz	a3b0357db50e0dec7b0474816fec287388adabc76cc309a40dee9bc73
kubernetes-node-linux-arm64.tar.gz	43af8ec4c5f2a1e2baa8cd13817e127fb6a3576dd811a30c4cc5f04d8
kubernetes-node-linux-ppc64le.tar.gz	840354219b3e59ed05b5b44cbbf4d45ccc4c0d74044e28c8a557ca75c
kubernetes-node-linux-s390x.tar.gz	796ca2e6855bd942a9a63d93f847ae62c5ee74195e041b60b89ee7d0e
kubernetes-node-windows-amd64.tar.gz	96d666e8446d09088bdcb440559035118dce07a2d9f5718856192fd80

Changelog since v1.12.0

Action Required

- kube-apiserver: the deprecated `--etcd-quorum-read` flag has been removed, and quorum reads are always enabled when fetching data from etcd. (#69527, @liggitt)
- Moved staging/src/k8s.io/client-go/tools/bootstrap to staging/src/k8s… (#67356, @yliaog)
- [action required] kubeadm: The `v1alpha2` config API has been removed. (#69055, @fabriziopandini)
  - Please convert your `v1alpha2` configuration files to `v1alpha3` using the
  - `kubeadm config migrate` command of kubeadm v1.12.x

Other notable changes

- Refactor factory_test.go to use a fake k8s client. (#69412, @tossmilestone)
- kubeadm: fix a case where fetching a kubernetesVersion from the internet still happened even if some commands don't need it. (#69645, @neolit123)
- Add tolerations for Stackdriver Logging and Metadata Agents. (#69737, @qingling128)
- Fix a bug in the scheduler that could cause the scheduler to go to an infinite loop when all nodes in a zone are removed. (#69758, @bsalamat)
- Dry-run is promoted to Beta and will be enabled by default. (#69644, @apelisse)
- `kubectl get priorityclass` now prints value column by default. (#69431, @Huang-Wei)
- Added a new container based image for running e2e tests (#69368, @dims)
- Remove the deprecated –google-json-key flag from kubelet. (#69354, @yujuhong)
- kube-apiserver: fixes `procMount` field incorrectly being marked as required in openapi schema (#69694, @jessfraz)
- The `LC_ALL` and `LC_MESSAGES` env vars can now be used to set desired locale for `kubectl` while keeping `LANG` unchanged. (#69500, @m1kola)
- Add ability to control primary GID of containers through Pod Spec and PodSecurityPolicy (#67802, @krmayankk)
- NodeLifecycleController: Now node lease renewal is treated as the heartbeat signal from the node, in addition to NodeStatus Update. (#69241, @wangzhen127)
- [GCE] Enable by default audit logging truncating backend. (#68288, @loburm)
- Enable insertId generation, and update Stackdriver Logging Agent image to 0.5-1.5.36-1-k8s. This help reduce log duplication and guarantee log

order. (#68920, @qingling128)
- Move NodeInfo utils into pkg/scheduler/cache. (#69495, @wgliang)
- adds dynamic shared informers to write generic, non-generated controllers (#69308, @p0lyn0mial)
- Move CacheComparer to pkg/scheduler/internal/cache/comparer. (#69317, @wgliang)
- Updating OWNERS list for vSphere Cloud Provider. (#69187, @Sandeep-Pissay)
- The default storage class annotation for the storage addons has been changed to use the GA variant (#68345, @smelchior)
- Upgrade to etcd 3.3 client (#69322, @jpbetz)
- fix GetVolumeLimits log flushing issue (#69558, @andyzhangx)
- It is now possible to use named ports in the `kubectl port-forward` command (#69477, @m1kola)
- kubeadm: fix a possible scenario where kubeadm can pull much newer control-plane images (#69301, @neolit123)
- test/e2e/e2e.test: (#69105, @pohly) * -viper-config can be used to set also the options defined by command line flags * the default config file is "e2e.yaml/toml/json/..." and the test starts when no such config is found (as before) but if -viper-config is used, the config file must exist * -viper-config can be used to select a file with full path, with or without file suffix * the csiImageVersion/Registry flags were renamed to storage.csi.imageVersion/Registry
- Move FakeCache to pkg/scheduler/internal/cache/fake. (#69318, @wgliang)
- The "kubectl cp" command now supports path shortcuts (../) in remote paths. (#65189, @juanvallejo)
- Fixed subpath in containerized kubelet. (#69565, @jsafrane)
- The runtimeHandler field on the RuntimeClass resource now accepts the empty string. (#69550, @tallclair)
- Kubelet can now parse PEM file containing both TLS certificate and key in arbitrary order. Previously key was always required to be first. (#69536, @awly)
- Scheduling conformance tests related to daemonsets should set the annotation that relaxes node selection restrictions, if any are set. This ensures conformance tests can run on a wider array of clusters. (#68793, @aveshagarwal)
- Replace Parallelize with function ParallelizeUntil and formally deprecate the Parallelize. (#68403, @wgliang)
- Move scheduler cache interface and implementation to pkg/scheduler/internal/cache. (#68968, @wgliang)
- Update to use go1.11.1 (#69386, @cblecker)
- Any external provider should be aware the cloud-provider interface should be imported from :- (#68310, @cheftako)
  - cloudprovider "k8s.io/cloud-provider"
- kubeadm: Fix a crash if the etcd local alpha phase is called when the

configuration contains an external etcd cluster (#69420, @ereslibre)
- kubeadm now allows mixing of init/cluster and join configuration in a single YAML file (although a warning gets printed in this case). (#69426, @rosti)
- Code-gen: Remove lowercasing for project imports (#68484, @jsturtevant)
- Fix client cert setup in delegating authentication logic (#69430, @DirectXMan12)
- service.beta.kubernetes.io/aws-load-balancer-internal now supports true and false values, previously it only supported non-empty strings (#69436, @mcrute)
- OpenAPI spec and API reference now reflect dryRun query parameter for POST/PUT/PATCH operations (#69359, @roycaihw)
- kubeadm: Add a `v1beta1` API. (#69289, @fabriziopandini)
- kube-apiserver has removed support for the `etcd2` storage backend (deprecated since v1.9). Existing clusters must migrate etcd v2 data to etcd v3 storage before upgrading to v1.13. (#69310, @liggitt)
- List operations against the API now return internal server errors instead of partially complete lists when a value cannot be transformed from storage. The updated behavior is consistent with all other operations that require transforming data from storage such as watch and get. (#69399, @mikedanese)
- `kubectl wait` now supports condition value checks other than true using `--for condition=available=false` (#69295, @deads2k)
- CCM server will not listen insecurely if secure port is specified (#68982, @aruneli)
- Bump cluster-proportional-autoscaler to 1.3.0 (#69338, @MrHohn)
  - - Rebase docker image on scratch.
- fix inconsistency in windows kernel proxy when updating HNS policy. (#68923, @delulu)
- Fixes the sample-apiserver so that its BanFlunder admission plugin can be used. (#68417, @MikeSpreitzer)
- Fixed CSIDriver API object to allow missing fields. (#69331, @jsafrane)
- Bump addon-manager to v8.8 (#69337, @MrHohn)
  - - Rebase docker image on debian-base:0.3.2.
- Update defaultbackend image to 1.5. Users should concentrate on updating scripts to the new version. (#69120, @aledbf)
- Bump Dashboard version to v1.10.0 (#68450, @jeefy)
- Fixed panic on iSCSI volume tear down. (#69140, @jsafrane)
- Update defaultbackend to v1.5 (#69334, @bowei)
- Remove unused chaosclient. (#68409, @wgliang)
- Enable AttachVolumeLimit feature (#69225, @gnufied)
- Update crictl to v1.12.0 (#69033, @feiskyer)
- Wait for pod failed event in subpath test. (#69300, @mrunalp)
- [GCP] Added env variables to control CPU requests of kube-controller-manager and kube-scheduler. (#68823, @loburm)

- Bump up pod short start timeout to 2 minutes. (#69291, @mrunalp)
- Use the mounted "/var/run/secrets/kubernetes.io/serviceaccount/token" as the token file for running in-cluster based e2e testing. (#69273, @dims)
- apiservice availability related to networking glitches are corrected faster (#68678, @deads2k)
- extract volume attachment status checking operation as a common function when attaching a CSI volume (#68931, @mlmhl)
- PodSecurityPolicy objects now support a `MayRunAs` rule for `fsGroup` and `supplementalGroups` options. This allows specifying ranges of allowed GIDs for pods/containers without forcing a default GID the way `MustRunAs` does. This means that a container to which such a policy applies to won't use any fsGroup/supplementalGroup GID if not explicitly specified, yet a specified GID must still fall in the GID range according to the policy. (#65135, @stlaz)
- Images for cloud-controller-manager, kube-apiserver, kube-controller-manager, and kube-scheduler now contain a minimal /etc/nsswitch.conf and should respect /etc/hosts for lookups (#69238, @BenTheElder)
- add deprecation warning for all cloud providers (#69171, @andrewsykim)
- IPVS proxier mode now support connection based graceful termination. (#66012, @Lion-Wei)
- Fix panic in kubectl rollout commands (#69150, @soltysh)
- Add fallbacks to ARM API when getting empty node IP from Azure IMDS (#69077, @feiskyer)
- Deduplicate PATH items when reading plugins. (#69089, @soltysh)
- Adds permissions for startup of an on-cluster kube-controller-manager (#69062, @dghubble)
- Fixes issue #68899 where pods might schedule on an unschedulable node. (#68984, @k82cn)
- Returns error if NodeGetInfo fails. (#68979, @xing-yang)
- Pod disruption budgets shouldn't be checked for terminal pods while evicting (#68892, @ravisantoshgudimetla)
- Fix scheduler crashes when Prioritize Map function returns error. (#68563, @DylanBLE)
- kubeadm: create control plane with ClusterFirstWithHostNet DNS policy (#68890, @andrewrynhard)
- Reduced excessive logging from fluentd-gcp-scaler. (#68837, @x13n)
- adds dynamic lister (#68748, @p0lyn0mial)
- kubectl: add the –no-headers flag to `kubectl top ...` (#67890, @WanLinghao)
- Restrict redirect following from the apiserver to same-host redirects, and ignore redirects in some cases. (#66516, @tallclair)
- Fixed pod cleanup when /var/lib/kubelet is a symlink. (#68741, @jsafrane)
- Add "only_cpu_and_memory" GET parameter to /stats/summary http handler in kubelet. If parameter is true then only cpu and memory will be present in response. (#67829, @krzysztof-jastrzebski)

- Start synchronizing pods after network is ready. (#68752, @krzysztofjastrzebski)
- kubectl has gained new –profile and –profile-output options to output go profiles (#68681, @dlespiau)
- Provides FSGroup capability on FlexVolume driver. It allows to disable the VolumeOwnership operation when volume is mounted (#68680, @benoitf)
- Apply _netdev mount option on bind mount (#68626, @gnufied)
- fix UnmountDevice failure on Windows (#68608, @andyzhangx)
- Allows changing nodeName in endpoint update. (#68575, @prameshj)
- kube-apiserver would return 400 Bad Request when it couldn't decode a json patch. (#68346, @CaoShuFeng)
  - kube-apiserver would return 422 Unprocessable Entity when a json patch couldn't be applied to one object.
- remove unused ReplicasetControllerOptions (#68121, @dixudx)
- Pass signals to fluentd process (#68064, @gianrubio)
- Flex drivers by default do not produce metrics. Flex plugins can enable metrics collection by setting the capability 'supportsMetrics' to true. Make sure the file system can support fs stat to produce metrics in this case. (#67508, @brahmaroutu)
- Use monotonically increasing generation to prevent scheduler equivalence cache race. (#67308, @cofyc)
- Fix kubelet service file permission warning (#66669, @daixiang0)
- Add prometheus metric for scheduling throughput. (#64526, @misterikkit)
- Get public IP for Azure vmss nodes. (#68498, @feiskyer)
- test/integration: add a basic test for covering CronJobs (#66937, @mortent)
- Make service environment variables optional (#68754, @bradhoekstra)

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 20, 2019 at 7:05 PM PST by Fix relative links issue in English content (#13307) (Page History)

Edit This Page

Building a release

You can either build a release from source or download a pre-built release. If you do not plan on developing Kubernetes itself, we suggest using a pre-built version of the current release, which can be found in the Release Notes.

The Kubernetes source code can be downloaded from the kubernetes/kubernetes repo.

- Building from source

Building from source

If you are simply building a release from source there is no need to set up a full golang environment as all building happens in a Docker container.

Building a release is simple.

```
git clone https://github.com/kubernetes/kubernetes.git
cd kubernetes
make release
```

For more details on the release process see the kubernetes/kubernetes `build` directory.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 08, 2019 at 5:04 AM PST by Update building-from-source.md (#12946) (Page History)

Edit This Page

v1.13 Release Notes

- v1.13.0
  - Downloads for v1.13.0
    - Client Binaries
    - Server Binaries
    - Node Binaries
- Kubernetes 1.13 Release Notes
  - Security Content
  - Urgent Upgrade Notes
    - (No, really, you MUST do this before you upgrade)
  - Known Issues
  - Deprecations
  - Major Themes
    - SIG API Machinery
    - SIG Auth
    - SIG AWS
    - SIG Azure
    - SIG Big Data
    - SIG CLI
    - SIG Cloud Provider
    - SIG Cluster Lifecycle
    - SIG IBM Cloud
    - SIG Multicluster
    - SIG Network
    - SIG Node
    - SIG Openstack
    - SIG Scalability
    - SIG Scheduling
    - SIG Service Catalog
    - SIG Storage
    - SIG UI
    - SIG VMWare
    - SIG Windows
  - New Features
  - Release Notes From SIGs

        * SIG API Machinery
        * SIG Auth
        * SIG Autoscaling
        * SIG AWS
        * SIG Azure
        * SIG CLI
        * SIG Cloud Provider
        * SIG Cluster Lifecycle
        * SIG GCP
        * SIG Network
        * SIG Node
        * SIG OpenStack
        * SIG Release
        * SIG Scheduling
        * SIG Storage
        * SIG Windows
    - External Dependencies
- v1.13.0-rc.2
    - Downloads for v1.13.0-rc.2
        * Client Binaries
        * Server Binaries
        * Node Binaries
    - Changelog since v1.13.0-rc.1
        * Other notable changes
- v1.13.0-rc.1
    - Downloads for v1.13.0-rc.1
        * Client Binaries
        * Server Binaries
        * Node Binaries
    - Changelog since v1.13.0-beta.2
        * Other notable changes
- v1.13.0-beta.2
    - Downloads for v1.13.0-beta.2
        * Client Binaries
        * Server Binaries
        * Node Binaries
    - Changelog since v1.13.0-beta.1
        * Other notable changes
- v1.13.0-beta.1
    - Downloads for v1.13.0-beta.1
        * Client Binaries
        * Server Binaries
        * Node Binaries
    - Changelog since v1.13.0-alpha.3
        * Action Required
        * Other notable changes

- v1.13.0-alpha.3
  - Downloads for v1.13.0-alpha.3
    - Client Binaries
    - Server Binaries
    - Node Binaries
  - Changelog since v1.13.0-alpha.2
    - Other notable changes
- v1.13.0-alpha.2
  - Downloads for v1.13.0-alpha.2
    - Client Binaries
    - Server Binaries
    - Node Binaries
  - Changelog since v1.13.0-alpha.1
    - Other notable changes
- v1.13.0-alpha.1
  - Downloads for v1.13.0-alpha.1
    - Client Binaries
    - Server Binaries
    - Node Binaries
  - Changelog since v1.12.0
    - Action Required
    - Other notable changes

v1.13.0

Documentation

Downloads for v1.13.0

filename	sha512 hash
kubernetes.tar.gz	`7b6a81c9f1b852b1e889c1b62281569a4b8853c79e5675b0910d941dfa7863c97f244f6d`
kubernetes-src.tar.gz	`844b9fbba21374dd190c8f12dd0e5b3303dd2cd7ad25f241d6f7e46f74adf6987afad021`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`0c010351acb660a75122feb876c9887d46ec2cb466872dd073b7f5b2`
kubernetes-client-darwin-amd64.tar.gz	`c2c40bd202900124f4e9458b067a1e1fc040030dc84ce9bcc6a5beb2`
kubernetes-client-linux-386.tar.gz	`5f5449be103b103d72a4e2b1028ab014cf7f74781166327f2ae284e4`
kubernetes-client-linux-amd64.tar.gz	`61a6cd3b1fb34507e0b762a45da09d88e34921985970a2ba594e0e5a`
kubernetes-client-linux-arm.tar.gz	`dd5591e2b88c347759a138c4d2436a0f5252341d0e8c9fbab16b8f15`

filename	sha512 hash
kubernetes-client-linux-arm64.tar.gz	894ed30261598ebf3485f3575e95f85e3c353f4d834bf9a6ea53b265
kubernetes-client-linux-ppc64le.tar.gz	6c26c807fc730ea736fda75dc57ac73395ba78bb828fffeee18b385b
kubernetes-client-linux-s390x.tar.gz	41e6e972de77c0bde22fdd779ea64e731b60f32e97e78a024f33fc3e
kubernetes-client-windows-386.tar.gz	442229e5030452901b924a94e7a879d4085597a4f201a5b3fc5ac980
kubernetes-client-windows-amd64.tar.gz	a11a8e8e732e7292781b9cb1de6e3e41683f95fb3fefc2b1a7b5fb1f

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	a8e3d457e5bcc1c09eeb66111e8dd049d6ba048c3c0fa90a61814291af
kubernetes-server-linux-arm.tar.gz	4e17494767000256775e4dd33c0a9b2d152bd4b5fba9f343b6dfeb5746
kubernetes-server-linux-arm64.tar.gz	0ddd0cf0ff56cebfa89efb1972cc2bc6916e824c2af56cfd330ac5638c
kubernetes-server-linux-ppc64le.tar.gz	b93828560224e812ed21b57fea5458fa8560745cfec96fc1677b258393
kubernetes-server-linux-s390x.tar.gz	154d565329d5ba52cdb7c3d43d8854b7a9b8e34803c4df6b3e6ae74c1a

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	9d18ba5f0c3b09edcf29397a496a1e908f4906087be3792989285630c
kubernetes-node-linux-arm.tar.gz	959b04ff7b8690413e01bffeabaab2119794dedf06b7aae1743e49988
kubernetes-node-linux-arm64.tar.gz	b5c18e8c9e28cf276067c871446720d86b6f162e22c3a5e9343cdbc68
kubernetes-node-linux-ppc64le.tar.gz	63e3504d3b115fdf3396968afafd1107b98e5a1a15b7c042a87f5a9cf
kubernetes-node-linux-s390x.tar.gz	21c5c2721febf7fddeada9569f3ecbd059267e5d2cc325d98fb74faf1
kubernetes-node-windows-amd64.tar.gz	3e73d3ecff14b4c85a71bb6cf91b1ab7d9c3075c64bd5ce6863562ab1

Kubernetes 1.13 Release Notes

Security Content

- CVE-2018-1002105, a critical security issue in the Kubernetes API Server, is resolved in v1.13.0 (and in v1.10.11, v1.11.5, and v1.12.3). We recommend all clusters running previous versions update to one of these releases immediately. See issue #71411 for details.

Urgent Upgrade Notes

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

Before upgrading to Kubernetes 1.13, you must keep the following in mind:

- kube-apiserver
  - The deprecated `etcd2` storage backend has been removed. Before upgrading a kube-apiserver using `--storage-backend=etcd2`, etcd v2 data must be migrated to the v3 storage backend, and kube-apiserver invocations changed to use `--storage-backend=etcd3`. Please consult the installation procedure used to set up etcd for specific migration instructions. Backups prior to upgrade are always a good practice, but since the etcd2 to etcd3 migration is not reversible, an etcd backup prior to migration is essential.
  - The deprecated `--etcd-quorum-read` flag has been removed. Quorum reads are now always enabled when fetching data from etcd. Remove the `--etcd-quorum-read` flag from kube-apiserver invocations before upgrading.
- kube-controller-manager
  - The deprecated `--insecure-experimental-approve-all-kubelet-csrs-for-group` flag has been removed.
- kubelet
  - The deprecated `--google-json-key` flag has been removed. Remove the `--google-json-key` flag from kubelet invocations before upgrading. (#69354, @yujuhong)
  - DaemonSet pods now make use of scheduling features that require kubelets to be at 1.11 or above. Ensure all kubelets in the cluster are at 1.11 or above before upgrading kube-controller-manager to 1.13.
  - The schema for the alpha `CSINodeInfo` CRD has been split into `spec` and `status` fields, and new fields `status.available` and `status.volumePluginMechanism` added. Clusters using the previous alpha schema must delete and recreate the CRD using the new schema. (#70515, @davidz627)
- kube-scheduler dropped support for configuration files with apiVersion `componentconfig/v1alpha1`. Ensure kube-scheduler is configured using command-line flags or a configuration file with apiVersion `kubescheduler.config.k8s.io/v1alpha1` before upgrading to 1.13.
- kubectl
  - The deprecated command `run-container` has been removed. Invocations should use `kubectl run` instead (#70728, @Pingan2017)
- client-go releases will no longer have bootstrap (k8s.io/client-go/tools/bootstrap) related code. Any reference to it will break. Please redirect all references to k8s.io/bootstrap instead. (#67356, @yliaog)
- Kubernetes cannot distinguish between GCE Zonal PDs and Regional PDs with the same name. To workaround this issue, precreate PDs with unique

names. PDs that are dynamically provisioned do not encounter this issue. (#70716, @msau42)

Known Issues

- If kubelet plugin registration for a driver fails, kubelet will not retry. The driver must delete and recreate the driver registration socket in order to force kubelet to attempt registration again. Restarting only the driver container may not be sufficient to trigger recreation of the socket, instead a pod restart may be required. (#71487)
- In some cases, a Flex volume resize may leave a PVC with erroneous Resizing condition even after volume has been successfully expanded. Users may choose to delete the condition, but it is not required. (#71470)
- The CSI driver-registrar external sidecar container v1.0.0-rc2 is known to take up to 1 minute to start in some cases. We expect this issue to be resolved in a future release of the sidecar container. For verification, please see the release notes of future releases of the external sidecar container. (#76)
- When using IPV6-only, be sure to use `proxy-mode=iptables` as `proxy-mode=ipvs` is known to not work. (#68437)

Deprecations

- kube-apiserver
  - The `--service-account-api-audiences` flag is deprecated in favor of `--api-audiences`. The old flag is accepted with a warning but will be removed in a future release. (#70105, @mikedanese)
  - The `--experimental-encryption-provider-config` flag is deprecated in favor of `--encryption-provider-config`. The old flag is accepted with a warning but will be removed in 1.14. (#71206, @stlaz)
  - As part of graduating the etcd encryption feature to beta, the configuration file referenced by `--encryption-provider-config` now uses `kind: EncryptionConfiguration` and `apiVersion: apiserver.config.k8s.io/v1`. Support for `kind: EncryptionConfig` and `apiVersion: v1` is deprecated and will be removed in a future release. (#67383, @stlaz)
  - The `--deserialization-cache-size` flag is deprecated, and will be removed in a future release. The flag is inactive since the etcd2 storage backend was removed. (#69842, @liggitt)
  - The `Node` authorization mode no longer allows kubelets to delete their Node API objects (prior to 1.11, in rare circumstances related to cloudprovider node ID changes, kubelets would attempt to delete/recreate their Node object at startup) (#71021, @liggitt)

  - The built-in `system:csi-external-provisioner` and `system:csi-external-attacher` cluster roles are deprecated and will not be auto-created in a future release. CSI deployments should provide their own RBAC role definitions with required permissions. (#69868, @pohly)
  - The built-in `system:aws-cloud-provider` cluster role is deprecated and will not be auto-created in a future release. Deployments using the AWS cloud provider should grant required permissions to the `aws-cloud-provider` service account in the `kube-system` namespace as part of deployment. (#66635, @wgliang)
- kubelet
  - Use of the beta plugin registration directory `{kubelet_root_dir}/plugins/` for registration of external drivers via the kubelet plugin registration protocol is deprecated in favor of `{kubelet_root_dir}/plugins_registry/`. Support for the old directory is planned to be removed in v1.15. Device plugin and CSI storage drivers should switch to the new directory prior to v1.15. Only CSI storage drivers that support 0.x versions of the CSI API are allowed in the old directory. (#70494 by @RenaudWasTaken and #71314 by @saad-ali)
  - With the release of the CSI 1.0 API, support for CSI drivers using 0.3 and older releases of the CSI API is deprecated, and is planned to be removed in Kubernetes v1.15. CSI drivers should be updated to support the CSI 1.0 API, and deployed in the new kubelet plugin registration directory (`{kubelet_root_dir}/plugins_registry/`) once all nodes in the cluster are at 1.13 or higher (#71020 and #71314, both by @saad-ali)
  - Use of the `--node-labels` flag to set labels under the `kubernetes.io/` and `k8s.io/` prefix will be subject to restriction by the `NodeRestriction` admission plugin in future releases. See admission plugin documentation for allowed labels. (#68267, @liggitt)
- kube-scheduler
  - The alpha critical pod annotation (`scheduler.alpha.kubernetes.io/critical-pod`) is deprecated. Pod priority should be used instead to mark pods as critical. (#70298, @bsalamat)
- The following features are now GA, and the associated feature gates are deprecated and will be removed in a future release:
  - CSIPersistentVolume
  - GCERegionalPersistentDisk
  - KubeletPluginsWatcher
  - VolumeScheduling
- kubeadm
  - The DynamicKubeletConfig feature gate is deprecated. The functionality is still accessible by using the kubeadm alpha kubelet enable-dynamic command.
  - The command `kubeadm config print-defaults` is deprecated in favor of `kubeadm config print init-defaults` and `kubeadm config print join-defaults` (#69617, @rosti)

  - support for the `v1alpha3` configuration file format is deprecated and will be removed in 1.14. Use `kubeadm config migrate` to migrate `v1alpha3` configuration files to `v1beta1`, which provides improvements in image repository management, addons configuration, and other areas. The documentation for `v1beta1` can be found here: https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta1
- The `node.status.volumes.attached.devicePath` field is deprecated for CSI volumes and will not be set in future releases (#71095, @msau42)
- kubectl
  - The `kubectl convert` command is deprecated and will be removed in a future release (#70820, @seans3)
- Support for passing unknown provider names to the E2E test binaries is deprecated and will be removed in a future release. Use `--provider=skeleton` (no ssh access) or `--provider=local` (local cluster with ssh) instead. (#70141, @pohly)

Major Themes

SIG API Machinery

For the 1.13 release, SIG API Machinery is happy to announce that the dry-run functionality is now beta.

SIG Auth

With this release we've made several important enhancements to core SIG Auth areas. In the authorization category, we've further reduced Kubelet privileges by restricting node self-updates of labels to a whitelisted selection and by disallowing kubelets from deleting their Node API object. In authentication, we added alpha-level support for automounting improved service account tokens through projected volumes. We also enabled audience validation in TokenReview for the new tokens for improved scoping. Under audit logging, the new alpha-level "dynamic audit configuration" adds support for dynamically registering webhooks to receive a stream of audit events. Finally, we've enhanced secrets protection by graduating etcd encryption out of experimental.

SIG AWS

In v1.13 we worked on tighter integrations of Kubernetes API objects with AWS services. These include three out-of-tree alpha feature releases:

1) Alpha for AWS ALB (Application Load Balancer) integration to Kubernetes Ingress resources. 2) Alpha for CSI specification 0.3 integration to AWS EBS

(Elastic Block Store) 3) Alpha for the cloudprovider-aws cloud controller manager binary. Additionally we added aws-k8s-tester, deployer interface for kubetest, to the test-infra repository. This plugin allowed us to integrate Prow to the 3 subprojects defined above in order to provide CI signal for all 3 features. The CI signal is visible here under SIG-AWS.

For detailed release notes on the three alpha features from SIG AWS, please refer to the following Changelogs:

- aws-alb-ingress-controller v1.0.0
- aws-ebs-csi-driver v0.1
- cloudprovider-aws external v0.1.0

SIG Azure

For 1.13 SIG Azure was focused on adding additional Azure Disk support for Ultra SSD, Standard SSD, and Premium Azure Files. Azure Availability Zones and cross resource group nodes were also moved from Alpha to Beta in 1.13.

SIG Big Data

During the 1.13 release cycle, SIG Big Data has been focused on community engagements relating to 3rd-party project integrations with Kubernetes. There have been no impacts on the 1.13 release.

SIG CLI

Over the course of 1.13 release SIG CLI mostly focused on stabilizing the items we've been working on over the past releases such as server-side printing and its support in kubectl, as well as finishing kubectl diff which is based on server-side dry-run feature. We've continued separating kubectl code to prepare for extraction out of main repository. Finally, thanks to the awesome support and feedback from community we've managed to promote the new plugin mechanism to Beta.

SIG Cloud Provider

For v1.13, SIG Cloud Provider has been focused on stabilizing the common APIs and interfaces consumed by cloud providers today. This involved auditing the cloud provider APIs for anything that should be deprecated as well as adding changes where necessary. In addition, SIG Cloud Provider has begun exploratory work around having a "cloud provider" e2e test suite which can be used to test common cloud provider functionalities with resources such as nodes and load balancers.

We are also continuing our long running effort to extract all the existing cloud providers that live in k8s.io/kubernetes into their own respective repos. Along with this migration, we are slowly transitioning users to use the cloud-controller-manager for any cloud provider features instead of the kube-controller-manager.

SIG Cluster Lifecycle

For 1.13 SIG Cluster Lifecycle is pleased to announce the long awaited promotion of kubeadm to stable GA, and the promotion of kubeadm's configuration API to `v1beta1`. In this release the SIG again focused on further improving the user experience on cluster creation and also fixing a number of bugs and other assorted improvements.

Some notable changes in kubeadm since Kubernetes 1.12:

- kubeadm's configuration API is now `v1beta1`. The new configuration format provides improvements in - image repository management, addons configuration, and other areas. We encourage `v1alpha3` users to migrate to this configuration API using `kubeadm config migrate`, as `v1alpha3` will be removed in 1.14. The documentation for `v1beta1` can be found here: https://godoc.org/k8s.io/kubernetes/cmd/kubeadm/app/apis/kubeadm/v1beta1
- kubeadm has graduated `kubeadm alpha phase` commands to `kubeadm init phase`. This means that the phases of creating a control-plane node are now tightly integrated as part of the `init` command. Alpha features, not yet ready for GA are still kept under `kubeadm alpha` and we appreciate feedback on them.
- `kubeadm init` and `kubeadm init phase` now have a `--image-repository` flag, improving support for environments with limited access to official kubernetes repository.
- The DynamicKubeletConfig and SelfHosting functionality was moved outside of `kubeadm init` and feature gates and is now exposed under `kubeadm alpha`.
- Kubeadm init phase certs now support the `--csr-only` option, simplifying custom CA creation.
- `kubeadm join --experimental-control-plane` now automatically adds a new etcd member for `local etcd` mode, further simplifying required tasks for HA clusters setup.
- Improvements were made to `kubeadm reset` related to cleaning etcd and notifying the user about the state of iptables.
- kubeadm commands now print warnings if input YAML documents contain unknown or duplicate fields.
- kubeadm now properly recognizes Docker 18.09.0 and newer, but still treats 18.06 as the default supported version.
- kubeadm now automatically sets the `--pod-infra-container-image` flag when starting the kubelet.

SIG IBM Cloud

The IBM Cloud SIG was focused on defining its charter and working towards moving its cloud provider code to an external repository with a goal to have this work done by the end of Kubernetes 1.14 release cycle. In the SIG meetings, we also made sure to share updates on the latest Kubernetes developments in the IBM Cloud like the availability of Kubernetes v1.12.2 in the IBM Cloud Kubernetes Service (IKS). The SIG updates were provided in the Kubernetes community weekly call and at the KubeCon China 2018.

SIG Multicluster

Moving Federation v2 from Alpha towards Beta has been the focus of our effort over the past quarter. To this end we engaged with end users, and successfully enlisted additional contributors from companies including IBM, Amadeus, Cisco and others. Federation v2 provides a suite of decoupled API's and re-usable components for building multi-cluster control planes. We plan to start releasing Beta components in late 2018. In addition, more minor updates were made to our cluster-registry and multi-cluster ingress sub-projects.

SIG Network

For 1.13, the areas of focus were in IPv6, DNS improvements and some smaller items: CoreDNS is now the default cluster DNS passing all of the scale/resource usage tests Node-local DNS cache feature is available in Alpha. This feature deploys a lightweight DNS caching Daemonset that avoids the conntrack and converts queries from UDP to more reliable TCP. PodReady++ feature now has `kubectl` CLI support.

Progress was made towards finalizing the IPv6 dual stack support KEP and support for topological routing of services.

SIG Node

SIG Node focused on stability and performance improvements in the 1.13 release. A new alpha feature is introduced to improve the mechanism that nodes heartbeat back to the control plane. The `NodeLease` feature results in the node using a `Lease` resource in the `kube-node-lease` namespace that is renewed periodically. The `NodeStatus` that was used previously to heartbeat back to the control plane is only updated when it changes. This reduces load on the control plane for large clusters. The Kubelet plugin registration mechanism, which enables automatic discovery of external plugins (including CSI and device plugins) has been promoted to stable in this release (introduced as alpha in 1.11 and promoted to beta in 1.12).

SIG Openstack

The major theme for the SIG OpenStack release is the work-in-progress for removing the in-tree provider. This work, being done in conjunction with SIG Cloud Provider, is focusing on moving internal APIs that the OpenStack (and other providers) depends upon to staging to guarantee API stability. This work also included abstracting the in-tree Cinder API and refactoring code to the external Cinder provider to remove additional Cinder volume provider code.

Additional work was also done to implement an OpenStack driver for the Cluster API effort lead by SIG Cluster Lifecycle. For the external Cloud-Provider-OpenStack code, the SIG largely focused on bug fixes and updates to match K8s 1.13 development.

SIG Scalability

SIG Scalability has mostly focused on stability and deflaking our tests, investing into framework for writing scalability tests (ClusterLoader v2) with a goal to migrate all tests to it by the end of 2018 and on the work towards extending definition of Kubernetes scalability by providing more/better user-friendly SLIs/SLOs.

SIG Scheduling

SIG Scheduling has mostly focused on stability in 1.13 and has postponed some of the major features to the next versions. There are still two notable changes: 1. TaintBasedEviction is moved to Beta and will be enabled by default. With this feature enabled, condition taints are automatically added to the nodes and pods can add tolerations for them if needed. 2. Pod critical annotation is deprecated. Pods should use pod priority instead of the annotation.

It is worth noting again that kube-scheduler will use apiVersion `kubescheduler.config.k8s.io/v1alpha1` instead of `componentconfig/v1alpha1` in its configuration files in 1.13.

SIG Service Catalog

The Service Plan Defaults feature is still under active development. We continue to improve the UX for the svcat CLI, specifically filling in gaps for the new Namespaced Service Broker feature.

SIG Storage

Over the last year, SIG Storage has been focused on adding support for the Container Storage Interface (CSI) to Kubernetes. The specification recently

moved to 1.0, and on the heels of this achievement, Kubernetes v1.13 moves CSI support for PersistentVolumes to GA.

With CSI the Kubernetes volume layer becomes truly extensible, allowing third party storage developers to write drivers making their storage systems available in Kubernetes without having to touch the core code.

CSI was first introduction as alpha in Kubernetes v1.9 and moved to beta in Kubernetes v1.10.

You can find a list of sample and production drivers in the CSI Documentation.

SIG Storage also moves support for Block Volumes to beta (introduced as alpha in v1.9) and support for Topology Aware Volume Scheduling to stable (introduced as alpha in v1.9 and promoted to beta in 1.10).

SIG UI

The migration to the newest version of Angular is still under active development as it is most important thing on the roadmap at the moment. We are getting closer to to the new release. We continue fixing bugs and adding other improvements.

SIG VMWare

Major focus for SIG VMware for this release is the work on moving internal APIs that the vSphere provider depends upon to staging to guarantee API stability. This work is being done in conjunction with SIG Cloud Provider and includes the creation of a brand new vsphere-csi plugin to replace the current volume functionalities in-tree.

Additional work was also done to implement a vSphere provider for the Cluster API effort lead by SIG Cluster Lifecycle. For the out-of-tree vSphere cloud provider, the SIG largely focused on bug fixes and updates to match K8s 1.13 development.

SIG Windows

SIG Windows focused on improving reliability for Windows and Kubernetes support

New Features

- kubelet: When node lease feature is enabled, kubelet reports node status to api server only if there is some change or it didn't report over last report

interval. (#69753, @wangzhen127)
- vSphereVolume implements Raw Block Volume Support (#68761, @fanzhangio)
- CRD supports multi-version Schema, Subresources and AdditionalPrintColumns (NOTE that CRDs created prior to 1.13 populated the top-level additionalPrinterColumns field by default. To apply an updated that changes to per-version additionalPrinterColumns, the top-level additionalPrinterColumns field must be explicitly set to null). (#70211, @roycaihw)
- New addon in addon manager that automatically installs CSI CRDs if CSIDriverRegistry or CSINodeInfo feature gates are true. (#70193, @saad-ali)
- Delegated authorization can now allow unrestricted access for `system:masters` like the main kube-apiserver (#70671, @deads2k)
- Added dns capabilities for Windows CNI plugins: (#67435, @feiskyer)
- kube-apiserver: `--audit-webhook-version` and `--audit-log-version` now default to `audit.k8s.io/v1` if unspecified (#70476, @charrywanganthony)
- kubeadm: timeoutForControlPlane is introduced as part of the API Server config, that controls the timeout for the wait for control plane to be up. Default value is 4 minutes. (#70480, @rosti)
- `--api-audiences` now defaults to the `--service-account-issuer` if the issuer is provided but the API audience is not. (#70308, @mikedanese)
- Added support for projected volume in describe function (#70158, @WanLinghao)
- kubeadm now automatically creates a new stacked etcd member when joining a new control plane node (does not applies to external etcd) (#69486, @fabriziopandini)
- Display the usage of ephemeral-storage when using `kubectl describe node` (#70268, @Pingan2017)
- Added functionality to enable br_netfilter and ip_forward for debian packages to improve kubeadm support for CRI runtime besides Docker. (#70152, @ashwanikhemani)
- Added regions ap-northeast-3 and eu-west-3 to the list of well known AWS regions. (#70252, @nckturner)
- kubeadm: Implemented preflight check to ensure that number of CPUs (#70048, @bart0sh)
- CoreDNS is now the default DNS server in kube-up deployments. (#69883, @chrisohaver)
- Opt out of chowning and chmoding from kubectl cp. (#69573, @bjhaid)
- Failed to provision volume with StorageClass "azurefile-premium": failed to create share andy-mg1121-dynamic-pvc-1a7b2813-d1b7-11e8-9e96-000d3a03e16b in account f7228f99bcde411e8ba4900: failed to create file share, err: storage: service returned error: StatusCode=400, ErrorCode=InvalidHeaderValue, ErrorMessage=The value for one of the HTTP headers is not in the correct format. (#69718, @andyzhangx)
- `TaintBasedEvictions` feature is promoted to beta. (#69824, @Huang-

Wei)
- Fixed https://github.com/kubernetes/client-go/issues/478 by adding support for JSON Patch in client-go/dynamic/fake (#69330, @vaikas-google)
- Dry-run is promoted to Beta and will be enabled by default. (#69644, @apelisse)
- `kubectl get priorityclass` now prints value column by default. (#69431, @Huang-Wei)
- Added a new container based image for running e2e tests (#69368, @dims)
- The `LC_ALL` and `LC_MESSAGES` env vars can now be used to set desired locale for `kubectl` while keeping `LANG` unchanged. (#69500, @m1kola)
- NodeLifecycleController: Now node lease renewal is treated as the heartbeat signal from the node, in addition to NodeStatus Update. (#69241, @wangzhen127)
- Added dynamic shared informers to write generic, non-generated controllers (#69308, @p0lyn0mial)
- Upgraded to etcd 3.3 client (#69322, @jpbetz)
- It is now possible to use named ports in the `kubectl port-forward` command (#69477, @m1kola)
- `kubectl wait` now supports condition value checks other than true using `--for condition=available=false` (#69295, @deads2k)
- Updated defaultbackend image to 1.5. Users should concentrate on updating scripts to the new version. (#69120, @aledbf)
- Bumped Dashboard version to v1.10.0 (#68450, @jeefy)
- Added env variables to control CPU requests of kube-controller-manager and kube-scheduler. (#68823, @loburm)
- PodSecurityPolicy objects now support a `MayRunAs` rule for `fsGroup` and `supplementalGroups` options. This allows specifying ranges of allowed GIDs for pods/containers without forcing a default GID the way `MustRunAs` does. This means that a container to which such a policy applies to won't use any fsGroup/supplementalGroup GID if not explicitly specified, yet a specified GID must still fall in the GID range according to the policy. (#65135, @stlaz)
- Upgrade Stackdriver Logging Agent addon image to 0.6-1.6.0-1 to use Fluentd v1.2. This provides nanoseconds timestamp granularity for logs. (#70954, @qingling128)
- When the BoundServiceAccountTokenVolumes Alpha feature is enabled, ServiceAccount volumes now use a projected volume source and their names have the prefix "kube-api-access". (#69848, @mikedanese)
- Raw block volume support is promoted to beta, and enabled by default. This is accessible via the `volumeDevices` container field in pod specs, and the `volumeMode` field in persistent volume and persistent volume claims definitions. (#71167, @msau42)
- TokenReview now supports audience validation of tokens with audiences other than the kube-apiserver. (#62692, @mikedanese)
- StatefulSet is supported in `kubectl autoscale` command (#71103, @Pingan2017)

- Kubernetes v1.13 moves support for Container Storage Interface to GA. As part of this move Kubernetes now supports CSI v1.0.0 and deprecates support for CSI 0.3 and older releases. Older CSI drivers must be updated to CSI 1.0 and moved to the new kubelet plugin registration directory in order to work with Kubernetes 1.15+. (#71020, @saad-ali)
- Added option to create CSRs instead of certificates for kubeadm init phase certs and kubeadm alpha certs renew (#70809, @liztio)
- Added a kubelet socket which serves an grpc service containing the devices used by containers on the node. (#70508, @dashpole)
- Added DynamicAuditing feature which allows for the configuration of audit webhooks through the use of an AuditSink API object. (#67257, @pbarker)
- The kube-apiserver's healthz now takes in an optional query parameter which allows you to disable health checks from causing healthz failures. (#70676, @logicalhan)
- Introduced support for running a nodelocal dns cache. It is disabled by default, can be enabled by setting KUBE_ENABLE_NODELOCAL_DNS=true (#70555, @prameshj)
- Added readiness gates in extended output for pods (#70775, @freehan)
- Added `Ready` column and improve human-readable output of Deployments and StatefulSets (#70466, @Pingan2017)
- Added `kubelet_container_log_size_bytes` metric representing the log file size of a container. (#70749, @brancz)
- NodeLifecycleController: When node lease feature is enabled, node lease will be deleted when the corresponding node is deleted. (#70034, @wangzhen127)
- GCERegionalPersistentDisk feature is GA now! (#70716, @jingxu97)
- Added secure port 10259 to the kube-scheduler (enabled by default) and deprecate old insecure port 10251. Without further flags self-signed certs are created on startup in memory. (#69663, @sttts)

Release Notes From SIGs

SIG API Machinery

- The OwnerReferencesPermissionEnforcement admission plugin now checks authorization for the correct scope (namespaced or cluster-scoped) of the owner resource type. Previously, it always checked permissions at the same scope as the child resource. (#70389, @caesarxuchao)
- OpenAPI spec now correctly marks delete request's body parameter as optional (#70032, @iamneha)
- The rules for incrementing `metadata.generation` of custom resources changed: (#69059, @caesarxuchao)
  - If the custom resource participates the spec/status convention, the metadata.generation of the CR increments when there is any change,

except for the changes to the metadata or the changes to the status.
  - If the custom resource does not participate the spec/status convention, the metadata.generation of the CR increments when there is any change to the CR, except for changes to the metadata.
  - A custom resource is considered to participate the spec/status convention if and only if the "CustomResourceSubresources" feature gate is turned on and the CRD has `.spec.subresources.status={}`.
- Fixed patch/update operations on multi-version custom resources (#70087, @liggitt)
- Reduced memory utilization of admission webhook metrics by removing resource related labels. (#69895, @jpbetz)
- Kubelet can now parse PEM file containing both TLS certificate and key in arbitrary order. Previously key was always required to be first. (#69536, @awly)
- Code-gen: Removed lowercasing for project imports (#68484, @jsturtevant)
- Fixed client cert setup in delegating authentication logic (#69430, @DirectXMan12)
- OpenAPI spec and API reference now reflect dryRun query parameter for POST/PUT/PATCH operations (#69359, @roycaihw)
- Fixed the sample-apiserver so that its BanFlunder admission plugin can be used. (#68417, @MikeSpreitzer)
- APIService availability related to networking glitches are corrected faster (#68678, @deads2k)
- Fixed an issue with stuck connections handling error responses (#71412, @liggitt)
- apiserver: fixed handling and logging of panics in REST handlers (#71076, @liggitt)
- kube-controller-manager no longer removes ownerReferences from ResourceQuota objects (#70035, @liggitt)
- "unfinished_work_microseconds" is added to the workqueue metrics; it can be used to detect stuck worker threads. (kube-controller-manager runs many workqueues.) (#70884, @lavalamp)
- Timeouts set in ListOptions for clients are also be respected locally (#70998, @deads2k)
- Added support for CRD conversion webhook (#67006, @mbohlool)
- client-go: fixed sending oversized data frames to spdystreams in remotecommand.NewSPDYExecutor (#70999, @liggitt)
- Fixed missing flags in `-controller-manager --help`. (#71298, @stewart-yu)
- Fixed missing flags in `kube-apiserver --help`. (#70204, @imjching)
- The caBundle and service fields in admission webhook API objects now correctly indicate they are optional (#70138, @liggitt)
- Fixed an issue with stuck connections handling error responses (#71419, @liggitt)
- kube-controller-manager and cloud-controller-manager now hold gener-

ated serving certificates in-memory unless a writeable location is specified with –cert-dir (#69884, @liggitt)
- CCM server will not listen insecurely if secure port is specified (#68982, @aruneli)
- List operations against the API now return internal server errors instead of partially complete lists when a value cannot be transformed from storage. The updated behavior is consistent with all other operations that require transforming data from storage such as watch and get. (#69399, @mikedanese)

SIG Auth

- API Server can be configured to reject requests that cannot be audit-logged. (#65763, @x13n)
- Go clients created from a kubeconfig that specifies a TokenFile now periodically reload the token from the specified file. (#70606, @mikedanese)
- When `--rotate-server-certificates` is enabled, kubelet will no longer request a new certificate on startup if the current certificate on disk is satisfactory. (#69991, @agunnerson-ibm)
- Added dynamic audit configuration api (#67547, @pbarker)
- Added ability to control primary GID of containers through Pod Spec and PodSecurityPolicy (#67802, @krmayankk)
- kube-apiserver: the `NodeRestriction` admission plugin now prevents kubelets from modifying `Node` labels prefixed with `node-restriction.kubernetes.io/`. The `node-restriction.kubernetes.io/` label prefix is reserved for cluster administrators to use for labeling `Node` objects to target workloads to nodes in a way that kubelets cannot modify or spoof. (#68267, @liggitt)

SIG Autoscaling

- Updated Cluster Autoscaler version to 1.13.0. See the Release Notes for more information. (#71513, @losipiuk)

SIG AWS

- `service.beta.kubernetes.io/aws-load-balancer-internal` now supports true and false values, previously it only supported non-empty strings (#69436, @mcrute)
- Added `service.beta.kubernetes.io/aws-load-balancer-security-groups` annotation to set the security groups to the AWS ELB to be the only ones specified in the annotation in case this is present (does not add `0.0.0.0/0`). (#62774, @Raffo)

SIG Azure

- Ensured orphan public IPs on Azure deleted when service recreated with the same name. (#70463, @feiskyer)
- Improved Azure instance metadata handling by adding caches. (#70353, @feiskyer)
- Corrected check for non-Azure managed nodes with the Azure cloud provider (#70135, @marc-sensenich)
- Fixed azure disk attach/detach failed forever issue (#71377, @andyzhangx)
- DisksAreAttached –> getNodeDataDisks–> GetDataDisks –> getVirtual-Machine –> vmCache.Get (#71495, @andyzhangx)

SIG CLI

- `kubectl apply` can now change a deployment strategy from rollout to recreate without explicitly clearing the rollout-related fields (#70436, @liggitt)
- The `kubectl plugin list` command now displays discovered plugin paths in the same order as they are found in a user's PATH variable. (#70443, @juanvallejo)
- `kubectl get` no longer exits before printing all of its results if an error is found (#70311, @juanvallejo)
- Fixed a runtime error occuring when sorting the output of `kubectl get` with empty results (#70740, @mfpierre)
- kubectl: support multiple arguments for cordon/uncordon and drain (#68655, @goodluckbot)
- Fixed ability for admin/edit/view users to see controller revisions, needed for kubectl rollout commands (#70699, @liggitt)
- `kubectl rollout undo` now returns errors when attempting to rollback a deployment to a non-existent revision (#70039, @liggitt)
- kubectl run now generates apps/v1 deployments by default (#71006, @liggitt)
- The "kubectl cp" command now supports path shortcuts (../) in remote paths. (#65189, @juanvallejo)
- Fixed dry-run output in kubectl apply –prune (#69344, @zegl)
- The kubectl wait command must handle when a watch returns an error vs closing by printing out the error and retrying the watch. (#69389, @smarterclayton)
- kubectl: support multiple arguments for cordon/uncordon and drain (#68655, @goodluckbot)

SIG Cloud Provider

- Added deprecation warning for all cloud providers (#69171, @andrewsykim)

SIG Cluster Lifecycle

- kubeadm: Updates version of CoreDNS to 1.2.6 (#70796, @detiber)
- kubeadm: Validate kubeconfig files in case of external CA mode. (#70537, @yagonobre)
- kubeadm: The writable config file option for extra volumes is renamed to readOnly with a reversed meaning. With readOnly defaulted to false (as in pod specs). (#70495, @rosti)
- kubeadm: Multiple API server endpoints support upon join is removed as it is now redundant. (#69812, @rosti)
- `kubeadm reset` now cleans up custom etcd data path (#70003, @yagonobre)
- kubeadm: Fixed unnecessary upgrades caused by undefined order of Volumes and VolumeMounts in manifests (#70027, @bart0sh)
- kubeadm: Fixed node join taints. (#69846, @andrewrynhard)
- Fixed cluster autoscaler addon permissions so it can access batch/job. (#69858, @losipiuk)
- kubeadm: JoinConfiguration now houses the discovery options in a nested Discovery structure, which in turn has a couple of other nested structures to house more specific options (BootstrapTokenDiscovery and FileDiscovery) (#67763, @rosti)
- kubeadm: Fixed a possible scenario where kubeadm can pull much newer control-plane images (#69301, @neolit123)
- kubeadm now allows mixing of init/cluster and join configuration in a single YAML file (although a warning gets printed in this case). (#69426, @rosti)
- kubeadm: Added a `v1beta1` API. (#69289, @fabriziopandini)
- kubeadm init correctly uses `--node-name` and `--cri-socket` when `--config` option is also used (#71323, @bart0sh)
- kubeadm: Always pass spec.nodeName as `--hostname-override` for kube-proxy (#71283, @Klaven)
- `kubeadm join` correctly uses `--node-name` and `--cri-socket` when `--config` option is also used (#71270, @bart0sh)
- kubeadm now supports the `--image-repository` flag for customizing what registry to pull images from (#71135, @luxas)
- kubeadm: The writable config file option for extra volumes is renamed to readOnly with a reversed meaning. With readOnly defaulted to false (as in pod specs). (#70495, @rosti)
- kubeadm: Multiple API server endpoints support upon join is removed as it is now redundant. (#69812, @rosti)

- kubeadm: JoinConfiguration now houses the discovery options in a nested Discovery structure, which in turn has a couple of other nested structures to house more specific options (BootstrapTokenDiscovery and FileDiscovery) (#67763, @rosti)
- kubeadm: Added a `v1beta1` API. (#69289, @fabriziopandini)
- kubeadm: Use `advertise-client-urls` instead of `listen-client-urls` as and `etcd-servers` options for apiserver. (#69827, @tomkukral)
- Kubeadm now respects the custom image registry configuration across joins and upgrades. Kubeadm passes the custom registry to the kubelet for a custom pause container. (#70603, @chuckha)
- `kubeadm reset` now outputs instructions about manual iptables rules cleanup. (#70874, @rdodev)
- kubeadm: remove the AuditPolicyConfiguration feature gate (#70807, @Klaven)
- kubeadm pre-pulls Etcd image only if external Etcd is not used and (#70743, @bart0sh)
- kubeadm: UnifiedControlPlaneImage is replaced by UseHyperKubeImage boolean value. (#70793, @rosti)
- For kube-up and derived configurations, CoreDNS will honor master taints, for consistency with kube-dns behavior. (#70868, @justinsb)
- Recognize newer docker versions without -ce/-ee suffix: 18.09.0 (#71001, @thomas-riccardi)
- Any external provider should be aware the cloud-provider interface should be imported from :- (#68310, @cheftako)
- Fixed 'kubeadm upgrade' infinite loop waiting for pod restart (#69886, @bart0sh)
- Bumped addon-manager to v8.8 (#69337, @MrHohn)
- GCE: Filter out spammy audit logs from cluster autoscaler. (#70696, @loburm)
- GCE: Enable by default audit logging truncating backend. (#68288, @loburm)
- Bumped cluster-proportional-autoscaler to 1.3.0 (#69338, @MrHohn)
- Updated defaultbackend to v1.5 (#69334, @bowei)

SIG GCP

- Added tolerations for Stackdriver Logging and Metadata Agents. (#69737, @qingling128)
- Enabled insertId generation, and updated Stackdriver Logging Agent image to 0.5-1.5.36-1-k8s. This help reduce log duplication and guarantee log order. (#68920, @qingling128)
- Updated crictl to v1.12.0 (#69033, @feiskyer)

SIG Network

- Corrected family type (inet6) for ipsets in ipv6-only clusters (#68436, @uablrek)
- kube-proxy argument `hostname-override` can be used to override hostname defined in the configuration file (#69340, @stevesloka)
- CoreDNS correctly implements DNS spec for Services with externalNames that look like IP addresses. Kube-dns does not follow the spec for the same case, resulting in a behavior change when moving from Kube-dns to CoreDNS. See: coredns/coredns#2324
- IPVS proxier now set net/ipv4/vs/conn_reuse_mode to 0 by default, which will highly improve IPVS proxier performance. (#71114, @Lion-Wei)
- CoreDNS is now version 1.2.6 (#70799, @rajansandeep)
- Addon configuration is introduced in the kubeadm config API, while feature flag CoreDNS is now deprecated. (#70024, @fabriziopandini)

SIG Node

- Fixed a bug in previous releases where a pod could be placed inside another pod's cgroup when specifying –cgroup-root (#70678, @dashpole)
- Optimized calculating stats when only CPU and Memory stats are returned from Kubelet stats/summary http endpoint. (#68841, @krzysztofjastrzebski)
- kubelet now supports `log-file` option to write logs directly to a specific file (#70917, @dims)
- Do not detach volume if mount in progress (#71145, @gnufied)
- The runtimeHandler field on the RuntimeClass resource now accepts the empty string. (#69550, @tallclair)
- kube-apiserver: fixes `procMount` field incorrectly being marked as required in openapi schema (#69694, @jessfraz)

SIG OpenStack

- Fixed cloud-controller-manager crash when using OpenStack provider and PersistentVolume initializing controller (#70459, @mvladev)

SIG Release

- Use debian-base instead of busybox as base image for server images (#70245, @ixdy)
- Images for cloud-controller-manager, kube-apiserver, kube-controller-manager, and kube-scheduler now contain a minimal /etc/nsswitch.conf and should respect /etc/hosts for lookups (#69238, @BenTheElder)

SIG Scheduling

- Added metrics for volume scheduling operations (#59529, @wackxu)
- Improved memory use and performance when processing large numbers of pods containing tolerations (#65350, @liggitt)
- Fixed a bug in the scheduler that could cause the scheduler to go to an infinite loop when all nodes in a zone are removed. (#69758, @bsalamat)
- Clear pod binding cache on bind error to make sure stale pod binding cache will not be used. (#71212, @cofyc)
- Fixed a scheduler panic due to internal cache inconsistency (#71063, @Huang-Wei)
- Report kube-scheduler unhealthy if leader election is deadlocked. (#71085, @bsalamat)
- Fixed a potential bug that scheduler preempts unnecessary pods. (#70898, @Huang-Wei)

SIG Storage

- Fixed CSI volume limits not showing up in node's capacity and allocatable (#70540, @gnufied)
- CSI drivers now have access to mountOptions defined on the storage class when attaching volumes. (#67898, @bswartz)
- change default azure file mount permission to 0777 (#69854, @andyzhangx)
- Fixed subpath in containerized kubelet. (#69565, @jsafrane)
- Fixed panic on iSCSI volume tear down. (#69140, @jsafrane)
- CSIPersistentVolume feature, i.e. PersistentVolumes with CSIPersistentVolumeSource, is GA. (#69929, @jsafrane)
- Fixed CSIDriver API object to allow missing fields. (#69331, @jsafrane)
- Flex volume plugins now support expandvolume (to increase underlying volume capacity) and expanfs (resize filesystem) commands that Flex plugin authors can implement to support expanding in use Flex PersistentVolumes (#67851, @aniket-s-kulkarni)
- Enabled AttachVolumeLimit feature (#69225, @gnufied)
- The default storage class annotation for the storage addons has been changed to use the GA variant (#68345, @smelchior)
- GlusterFS PersistentVolumes sources can now reference endpoints in any namespace using the `spec.glusterfs.endpointsNamespace` field. Ensure all kubelets are upgraded to 1.13+ before using this capability. (#60195, @humblec)
- Fixed GetVolumeLimits log flushing issue (#69558, @andyzhangx)
- The `MountPropagation` feature is unconditionally enabled in v1.13, and can no longer be disabled. (#68230, @bertinatto)

SIG Windows

- `kubelet --system-reserved` and `--kube-reserved` are supported now on Windows nodes (#69960, @feiskyer)
- Windows runtime endpoints is now switched to `npipe:////./pipe/dockershim` from `tcp://localhost:3735`. (#69516, @feiskyer)
- Fixed service issues with named targetPort for Windows (#70076, @feiskyer)
- Handle Windows named pipes in host mounts. (#69484, @ddebroy)
- Fixed inconsistency in windows kernel proxy when updating HNS policy. (#68923, @delulu)

External Dependencies

- Default etcd server is unchanged at v3.2.24 since Kubernetes 1.12. (#68318)
- The list of validated docker versions remain unchanged at 1.11.1, 1.12.1, 1.13.1, 17.03, 17.06, 17.09, 18.06 since Kubernetes 1.12. (#68495)
- The default Go version was updated to 1.11.2. (#70665)
- The minimum supported Go version was updated to 1.11.2 (#69386)
- CNI is unchanged at v0.6.0 since Kubernetes 1.10 (#51250)
- CSI is updated to 1.0.0. Pre-1.0.0 API support is now deprecated. (#71020])
- The dashboard add-on has been updated to v1.10.0. (#68450)
- Heapster remains at v1.6.0-beta, but is now retired in Kubernetes 1.13 (#67074)
- Cluster Autoscaler has been upgraded to v1.13.0 (#71513)
- kube-dns is unchanged at v1.14.13 since Kubernetes 1.12 (#68900)
- Influxdb is unchanged at v1.3.3 since Kubernetes 1.10 (#53319)
- Grafana is unchanged at v4.4.3 since Kubernetes 1.10 (#53319)
- Kibana has been upgraded to v6.3.2. (#67582)
- CAdvisor has been updated to v0.32.0 (#70964)
- fluentd-gcp-scaler has been updated to v0.5.0 (#68837)
- Fluentd in fluentd-elasticsearch is unchanged at v1.2.4 since Kubernetes 1.11 (#67434)
- fluentd-elasticsearch has been updated to v2.2.1 (#68012)
- The fluent-plugin-kubernetes_metadata_filter plugin in fluentd-elasticsearch is unchanged at 2.0.0 since Kubernetes 1.12 (#67544)
- fluentd-gcp has been updated to v3.2.0 (#70954)
- OIDC authentication is unchanged at coreos/go-oidc v2 since Kubernetes 1.10 (#58544)
- Calico was updated to v3.3.1 (#70932)
- Upgraded crictl on GCE to v1.12.0 (#69033)
- CoreDNS has been updated to v1.2.6 (#70799)
- event-exporter has been updated to v0.2.3 (#67691)

- Es-image remains unchanged at Elasticsearch 6.3.2 since Kubernetes 1.12 (#67484)
- metrics-server remains unchanged at v0.3.1 since Kubernetes 1.12 (#68746)
- GLBC remains unchanged at v1.2.3 since Kubernetes 1.12 (#66793)
- Ingress-gce remains unchanged at v1.2.3 since Kubernetes 1.12 (#66793)
- ip-masq-agen remains unchanged at v2.1.1 since Kubernetes 1.12 (#67916)
- v1.13.0-rc.2
- v1.13.0-rc.1
- v1.13.0-beta.2
- v1.13.0-beta.1
- v1.13.0-alpha.3
- v1.13.0-alpha.2
- v1.13.0-alpha.1

v1.13.0-rc.2

Documentation

Downloads for v1.13.0-rc.2

filename	sha512 hash
kubernetes.tar.gz	`12fbaf943ae72711cd93c9955719ec1773a229dbb8f86a44fcda179229beb82add4dc1a5`
kubernetes-src.tar.gz	`8e94f0fe73909610e85c201bb1ba4f66fd55ca2b4ded77217a4dfad2874d402cc1cc9420`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`ac555f5d1e6b88fa4de1e06e0a1ebd372582f97c526c938334a8c63f`
kubernetes-client-darwin-amd64.tar.gz	`2eae428a0e4bcb2237343d7ac1e431ccfc1f7037622bb3131ad8d48a`
kubernetes-client-linux-386.tar.gz	`89e671679b4516f184f7fd5ea0fe2a9ab0245fab34447625786bf558`
kubernetes-client-linux-amd64.tar.gz	`61f6513722e9c485300b822d6fc5998927bbffa18862d2d3f177a7c7`
kubernetes-client-linux-arm.tar.gz	`ef0e5fd4bf2074dfd3cf54d45307550273695906baca3533a9d23424`
kubernetes-client-linux-arm64.tar.gz	`d34bb9ce9bfe2a5375fd58920e63b4eef818348719dba460f3583843`
kubernetes-client-linux-ppc64le.tar.gz	`4dc4e4a5e166e63360ba86e1278bbe75212ac7c3f60ba30425a1c565`
kubernetes-client-linux-s390x.tar.gz	`d27675f4753469cd5e31faed13a1ea9654c25d38b0d96c1340215fd2`
kubernetes-client-windows-386.tar.gz	`9d6e6de2d4a55eaeebd7fa6b861548e0768381d50838430722b56636`
kubernetes-client-windows-amd64.tar.gz	`30b2da5c015ef88b9efcf90bffe0498d367df7c126b65f2e878af263`

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	8180f2b788249fe65f7f1d3ee431ac758ede29a6349db312afbee080ff
kubernetes-server-linux-arm.tar.gz	e9165284a0b82a9ab88dad05f43bfe1bebecad3bb1c7118475c3426e0b
kubernetes-server-linux-arm64.tar.gz	03797c021ebed3b08835e72eed405c57aaacce972bbbbf88bf49310efb
kubernetes-server-linux-ppc64le.tar.gz	ceb49af22e3b518f3ba27c1e7de28e577e2735175e84a6d203f1f8766e
kubernetes-server-linux-s390x.tar.gz	bee4752e8a52e217ae1ffcfbc263453c724de684b4d463d5ddb24a3a30

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	b368989bbb8ab4d29b51d5d4d71d073b0ceb39614c944859dcd14c330
kubernetes-node-linux-arm.tar.gz	404b7b74a1e0d0fed9088a7e9461e02cfd9a6992c554baa125b7a361a
kubernetes-node-linux-arm64.tar.gz	fa531b1675a778c572a2175fb1bed00e78dc589f638f2096b3b5c9d3c
kubernetes-node-linux-ppc64le.tar.gz	a7ecc1f63e632c1b4f9b312babd6882ec966420bf4f8346edf80495fc
kubernetes-node-linux-s390x.tar.gz	a7171ed95de943a0ac5a32da4458e8d4366eb1fadbe426bebc371d2bb
kubernetes-node-windows-amd64.tar.gz	8a3a71d142b99fb200c4c1c9c0fa4dc6a3b64a0b506dc37dc3d832a94

Changelog since v1.13.0-rc.1

Other notable changes

- Update Cluster Autoscaler version to 1.13.0. Release notes: https://github.com/kubernetes/autoscaler/releases/tag/cluster-autoscaler-1.13.0 (#71513, @losipiuk)
- fix detach azure disk issue due to dirty cache (#71495, @andyzhangx)

v1.13.0-rc.1

Documentation

Downloads for v1.13.0-rc.1

filename	sha512 hash
kubernetes.tar.gz	1c047e4edcf3553a568679e6e5083988b06df9d938f299a9193c72ad96a9c439a1f47f98
kubernetes-src.tar.gz	d2fd47c38abd29a2037b9e2a3a958ec250e2c6ae77532f6e935a6422bd626485fd720932

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	44d0733359be5036953775e12fc1723e4c64452a24a8c3b522c8a624
kubernetes-client-darwin-amd64.tar.gz	2acd37ed234271b0ff9c30273261e4b127309a1bc91a006b7a07e1a9
kubernetes-client-linux-386.tar.gz	5fe07ea2f776086df0e9447b7e6b0863c5b3af71f5aff8e302087e24
kubernetes-client-linux-amd64.tar.gz	7541d5850d74156862e5fe00817bd954d2b49b2c0cf15abe5cde3440
kubernetes-client-linux-arm.tar.gz	122121d3e469b6e33cc3fd910b32a5a94b9d3479f0367c54fbc4e7f1
kubernetes-client-linux-arm64.tar.gz	5e3d415db4239f27461c4ea404903cfc762084d5c1e84f9ed8bc0325
kubernetes-client-linux-ppc64le.tar.gz	8651f4161569913b616695bdd1a41c4b177cbfb4773fbca649b3e979
kubernetes-client-linux-s390x.tar.gz	920b81f6bbc7e7d4fa2f9c61fbc6f529621f2f134dbbb0f407866ffd
kubernetes-client-windows-386.tar.gz	0d49277cb7c36e5538d4c1c0fd6e6a69da7cd73c226f5869b29fad1e
kubernetes-client-windows-amd64.tar.gz	34ae587e2d439f925d1e324d2bbff3a751bb73b18e98b13c93e5742e

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	7030ef7463bef0871e524a5233d23c5f8aee18ac92e96555910ddc7a89
kubernetes-server-linux-arm.tar.gz	ccd1f413ad357581a904d1ff67f3e376be7882bd72efb13657f8aa1191
kubernetes-server-linux-arm64.tar.gz	ff589f5b6c56713818edda8ae9b39b17dfbf34e881c09736f722de5d70
kubernetes-server-linux-ppc64le.tar.gz	f748985751bf403bc7b1f9160ce937cd2915552b27c3c79764a66789dc
kubernetes-server-linux-s390x.tar.gz	b3b0075948d72784defe94073dff251b79083aa46b4f2941902675766

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	01907a104c043607985053571183b7bdccf655f847d1dd9d8991cd2c4
kubernetes-node-linux-arm.tar.gz	dbf1801c456312698253767dd36b186fb4e503a03454cd16bba68a1ec
kubernetes-node-linux-arm64.tar.gz	15f3259370f1419fcc372a28faa9a3caae5f2c89ee76286c14ea62d61
kubernetes-node-linux-ppc64le.tar.gz	00dc7f5bd40d045baeb72d5dcfb302b8566aacc23cd7de1b877724e11
kubernetes-node-linux-s390x.tar.gz	2b80e4dffa0b8bdc0305d1263c06320918541f3a7b6519123752b89be
kubernetes-node-windows-amd64.tar.gz	600b442a1665e39621fce03ad07b162e2353cc8bc982cad849dab7e1c

Changelog since v1.13.0-beta.2

Other notable changes

- CVE-2018-1002105: Fix critical security issue in kube-apiserver upgrade request proxy handler (#71411, @liggitt)

- Update Cluster Autoscaler version to 1.13.0-rc.2. Release notes: https://github.com/kubernetes/autoscaler/releases/tag/cluster-autoscaler-1.13.0-rc.2 (#71452, @losipiuk)
- Upgrade Stackdriver Logging Agent addon image to 0.6-1.6.0-1 to use Fluentd v1.2. This provides nanoseconds timestamp granularity for logs. (#70954, @qingling128)
- fixes a runtime error occuring when sorting the output of `kubectl get` with empty results (#70740, @mfpierre)
- fix azure disk attach/detach failed forever issue (#71377, @andyzhangx)
- Do not detach volume if mount in progress (#71145, @gnufied)

v1.13.0-beta.2

Documentation

Downloads for v1.13.0-beta.2

filename	sha512 hash
kubernetes.tar.gz	`e8607473e2b3946a3655fa895c2b7dee74818b4c2701047fee5343ab6b2f2aa3d97b19b1`
kubernetes-src.tar.gz	`6ca15ad729a82b41587e1dbbd4e9ad5447e202e8e7ee8c01c411090031ee3feb83f0cc65`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`5727218280ea7c68350aa5cf04e3d3c346f97d462e3f60f5196e2735`
kubernetes-client-darwin-amd64.tar.gz	`3e3975a41da08135dc654a40acb86ce862b1f56a9361e0c38c9c99c5`
kubernetes-client-linux-386.tar.gz	`26cfa99fbe09b20ebe3d2aebb4d08f0f9f2661d5533b94daf6c83547`
kubernetes-client-linux-amd64.tar.gz	`42204953b02af81bb5f695c957aca9fa382609447ada5e3a9701da3e`
kubernetes-client-linux-arm.tar.gz	`c680c94699b0b319b654a4c1c0a9b7fc387c44fb22744f30049142b1`
kubernetes-client-linux-arm64.tar.gz	`aa997b3428979ba2652fd251c4c5ece87043472ebe2ee15d8a179e69`
kubernetes-client-linux-ppc64le.tar.gz	`684dfc462d84d3902e322535997e57f7874003ab17c41508c057bc7c`
kubernetes-client-linux-s390x.tar.gz	`ff98b3a23dfe436a12843eb388be9568cbc29c9328648a1d166518aa`
kubernetes-client-windows-386.tar.gz	`6897a0f59fb409526dae9c86680702f3d2a1dc68d145504ed2e98b05`
kubernetes-client-windows-amd64.tar.gz	`6ed67eecb2b79ace8d428cbd4d07ef7d52ba4e5b3b44eb59d46aff99`

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	`351292b217c1c49b5c0241da11b4be0929a5d1645bec7dd05051930df8`

filename	sha512 hash
kubernetes-server-linux-arm.tar.gz	`88f166a7b5a3f9d9c19a5b911adb6e8e4cac1a3323b83d681f13aaf7bb`
kubernetes-server-linux-arm64.tar.gz	`fb4868a939eca18de17e0b606d1ab127712e277e01c02ffa96138a5397`
kubernetes-server-linux-ppc64le.tar.gz	`47a4e8e96c1e8a8cc37eabd19194b9d174fa93c3feaf1384895f89c5c6`
kubernetes-server-linux-s390x.tar.gz	`4e0823d1da55a71f001fcb07511a7b3416641ea93bfbd56b1e1e435c0a`

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	`e21964063b80f52e387cd35826f3081ad0a3b62608d182e008b8b76f5`
kubernetes-node-linux-arm.tar.gz	`cb665911af59a1cf86e5d66a4cdc134dc412e9e479dd89fa0bbbaeb83`
kubernetes-node-linux-arm64.tar.gz	`c172126829aea38e2238af6b62035abad6ed08d041175b0bf99792b7c`
kubernetes-node-linux-ppc64le.tar.gz	`0367940078ea9b4d46778b8406840fd2925f612304b5fa5b675fc07d5`
kubernetes-node-linux-s390x.tar.gz	`74382ed862ae099b91ce6056b85b7ee4f075fbdb4e737a8448c92e20f`
kubernetes-node-windows-amd64.tar.gz	`9164c4eae920c727965caae046e1b2daabf4822e2dee2260697b22e52`

Changelog since v1.13.0-beta.1

Other notable changes

- Fix missing flags in kube-apiserver –help. (#70204, @imjching)
- kubeadm init correctly uses –node-name and –cri-socket when –config option is also used (#71323, @bart0sh)
- API server flag `--experimental-encryption-provider-config` was renamed to `--encryption-provider-config`. The old flag is accepted with a warning but will be removed in 1.14. (#71206, @stlaz)
- Fix missing flags in *-controller-manager –help. (#71298, @stewart-yu)
- Clear pod binding cache on bind error to make sure stale pod binding cache will not be used. (#71212, @cofyc)
- kubeadm: always pass spec.nodeName as –hostname-override for kube-proxy (#71283, @Klaven)
- kubeadm join correctly uses –node-name and –cri-socket when –config option is also used (#71270, @bart0sh)
- apiserver can be configured to reject requests that cannot be audit-logged. (#65763, @x13n)
- Kubelet Device Plugin Registration directory changed from from `{kubelet_root_dir}/plugins/` to `{kubelet_root_dir}/plugins_registry/`. Any drivers (CSI or device plugin) that were using the old path must be updated to work with this version. (#70494, @RenaudWasTaken)
- When the BoundServiceAccountTokenVolumes Alpha feature is enabled, ServiceAccount volumes now use a projected volume source and their names have the prefix “kube-api-access”. (#69848, @mikedanese)

v1.13.0-beta.1

Documentation

Downloads for v1.13.0-beta.1

filename	sha512 hash
kubernetes.tar.gz	78245b2357a5eeafd193d28f86655327edce7bbc4da142c826eba5f5c05a624cd30b2551
kubernetes-src.tar.gz	880c5a8b16215bc58b307922474703048020b38be1d41672425cd07bdcf0626a88f04a08

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	0f804c77ef6122b4b6586a507179fe0f1a383752342b3e5575e09223
kubernetes-client-darwin-amd64.tar.gz	0bdbd8003bcbecb4494b4778411e7d057067e78a99a7e8e8e45a3982
kubernetes-client-linux-386.tar.gz	522795df77ff8543251232863cb36fe2d501671e04a5279a112aa3ff
kubernetes-client-linux-amd64.tar.gz	b6481bae237e6971f7b9cc039d3b7e62d49ddd48d52dd979432fa031
kubernetes-client-linux-arm.tar.gz	45b8fa2557bb742a8ce16e0a69fa64fe898509418c6f9099a24bf1ab
kubernetes-client-linux-arm64.tar.gz	475b823a5e2c4c6e1bc49f35fbef45d1fc6e6279f5335762bad05d0f
kubernetes-client-linux-ppc64le.tar.gz	bc289b249051e9918f8f842bb98bf4d0b8951709fe5b65c2185f04b7
kubernetes-client-linux-s390x.tar.gz	0935e0ad23a61d570de087e72f22bc3da2a34c19bb5aea0ab342f916
kubernetes-client-windows-386.tar.gz	4833425ff040983b841722a00edd2cfa56f85099658ae04890c4e226
kubernetes-client-windows-amd64.tar.gz	156a5328834055f7b9732c762cc917cfdbf2d2fc67dd80ba89ae7dcb

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	9435be5cced10252954be579408e2a253eb51dd7b649417f1e91679bce
kubernetes-server-linux-arm.tar.gz	8dc3d4a0c09830831efd77fdf193ed9ccc1247bd981b4811192cac38cf
kubernetes-server-linux-arm64.tar.gz	f2549f87f21ea44c5d776a706c59bb2ea61d7f2cca304850aa6ad5b09c
kubernetes-server-linux-ppc64le.tar.gz	0c0671eaf7cf7262c95411930311bb4610f89583431738149f0ee7f8f6
kubernetes-server-linux-s390x.tar.gz	ff24909b0b044924d241d6aeac9e9b4f0696c0ca7e973d56a874b02b61

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	235b1c4348b5779ca71a5f63121ff6a162db02bb24b4d815ec73412af
kubernetes-node-linux-arm.tar.gz	cd23813419a74983bdd3a3104e20684e947ef7302dcfa1802132439b2

filename	sha512 hash
kubernetes-node-linux-arm64.tar.gz	`fb6283dae828f8d9275a05c6a4ea27bc1136e8e8253b5ddac52c82548`
kubernetes-node-linux-ppc64le.tar.gz	`f910922d422b65f6b6a8d7762a23048991695496c0fc07c3dc4f1a81e`
kubernetes-node-linux-s390x.tar.gz	`d895fed57caf038afe0087ff44d2adfdd8955d18135adad9935952702`
kubernetes-node-windows-amd64.tar.gz	`4bc09e54935d2cb4d2bad7db06831d040cc03906d8934575932ed6eab`

Changelog since v1.13.0-alpha.3

Action Required

- ACTION REQUIRED: The Node.Status.Volumes.Attached.DevicePath fields is deprecated for CSI volumes and will be unset in a future release (#71095, @msau42)

Other notable changes

- Raw block volume support is promoted to beta, and enabled by default. This is accessible via the `volumeDevices` container field in pod specs, and the `volumeMode` field in persistent volume and persistent volume claims definitions. (#71167, @msau42)
- Fix a scheduler panic due to internal cache inconsistency (#71063, @Huang-Wei)
- Fix a potential bug that scheduler preempts unnecessary pods. (#70898, @Huang-Wei)
- The API server encryption configuration file format has graduated to stable and moved to `apiVersion: apiserver.config.k8s.io/v1` and `kind: EncryptionConfiguration`. (#67383, @stlaz)
- kubelet now supports `log-file` option to write logs directly to a specific file (#70917, @dims)
- kubeadm now supports the `--image-repository` flag for customizing what registry to pull images from (#71135, @luxas)
- timeouts set in ListOptions for clients will also be respected locally (#70998, @deads2k)
- IPVS proxier now set net/ipv4/vs/conn_reuse_mode to 0 by default, which will highly improve IPVS proxier performance. (#71114, @Lion-Wei)
- StatefulSet is supported in `kubectl autoscale` command (#71103, @Pingan2017)
- Report kube-scheduler unhealthy if leader election is deadlocked. (#71085, @bsalamat)
- apiserver: fixes handling and logging of panics in REST handlers (#71076, @liggitt)

- kubelets are no longer allowed to delete their own Node API object. Prior to 1.11, in rare circumstances related to cloudprovider node ID changes, kubelets would attempt to delete/recreate their Node object at startup. Kubelets older than 1.11 are not supported running against a v1.13+ API server. If an unsupported legacy kubelet encounters this situation, a cluster admin can remove the Node object: (#71021, @liggitt) * `kubectl delete node/<nodeName>`
  - or grant self-deletion permission explicitly:
    * `kubectl create clusterrole self-deleting-nodes --verb=delete --resource=nodes`
    * `kubectl create clusterrolebinding self-deleting-nodes --clusterrole=self-deleting-nodes --group=system:nodes`
- Kubernetes v1.13 moves support for Container Storage Interface to GA. As part of this move Kubernetes now supports CSI v1.0.0 and drops support for CSI 0.3 and older releases. Older CSI drivers must be updated to CSI 1.0 in order to work with Kubernetes 1.13+. (#71020, @saad-ali)
- Remove deprecated kubectl command aliases 'run-container' (#70728, @Pingan2017)
- kubeadm: enable strict unmarshaling of YAML configuration files and show warnings for unknown and duplicate fields. (#70901, @neolit123)
- For kube-up and derived configurations, CoreDNS will honor master taints, for consistency with kube-dns behavior. (#70868, @justinsb)
- CoreDNS is now version 1.2.6 (#70799, @rajansandeep)
- kubeadm: Use `advertise-client-urls` instead of `listen-client-urls` as and `etcd-servers` options for apiserver. (#69827, @tomkukral)
- Add option to create CSRs instead of certificates for kubeadm init phase certs and kubeadm alpha certs renew (#70809, @liztio)
- Add a kubelet socket which serves an grpc service containing the devices used by containers on the node. (#70508, @dashpole)
- kube-apiserver: the `NodeRestriction` admission plugin now prevents kubelets from modifying `Node` labels prefixed with `node-restriction.kubernetes.io/`. The `node-restriction.kubernetes.io/` label prefix is reserved for cluster administrators to use for labeling `Node` objects to target workloads to nodes in a way that kubelets cannot modify or spoof. (#68267, @liggitt)
  - kubelet: it is now deprecated to use the `--node-labels` flag to set `kubernetes.io/` and `k8s.io/`-prefixed labels other than the following labels:
    * `kubernetes.io/hostname`
    * `kubernetes.io/instance-type`
    * `kubernetes.io/os`
    * `kubernetes.io/arch`
    * `beta.kubernetes.io/instance-type`
    * `beta.kubernetes.io/os`
    * `beta.kubernetes.io/arch`
    * `failure-domain.kubernetes.io/zone`
    * `failure-domain.kubernetes.io/region`

    * `failure-domain.beta.kubernetes.io/zone`
    * `failure-domain.beta.kubernetes.io/region`
    * `[*.]kubelet.kubernetes.io/*`
    * `[*.]node.kubernetes.io/*`
  - Setting other `kubernetes.io/`- and `k8s.io/`-prefixed labels using the `--node-labels` flag will produce a warning in v1.13, and be disallowed in v1.15. Setting labels that are not prefixed with `kubernetes.io/` or `k8s.io/` is still permitted.
- Adds DynamicAuditing feature which allows for the configuration of audit webhooks through the use of an AuditSink API object. (#67257, @pbarker)
- The Kubelet plugin registration mechanism used by device plugins and CSI plugins is now GA (#70559, @vladimirvivien)
- CSIPersistentVolume feature, i.e. PersistentVolumes with CSIPersistentVolumeSource, is GA. (#69929, @jsafrane)
  - CSIPersistentVolume feature gate is now deprecated and will be removed according to deprecation policy.
- kubectl: support multiple arguments for cordon/uncordon and drain (#68655, @goodluckbot)
- The kube-apiserver's healthz now takes in an optional query parameter which allows you to disable health checks from causing healthz failures. (#70676, @logicalhan)
- client-go: fixes sending oversized data frames to spdystreams in remotecommand.NewSPDYExecutor (#70999, @liggitt)
- kube-controller-manager no longer removes ownerReferences from ResourceQuota objects (#70035, @liggitt)
- Introduces support for running a nodelocal dns cache. It is disabled by default, can be enabled by setting KUBE_ENABLE_NODELOCAL_DNS=true (#70555, @prameshj)
  - An ip address is required for the cache instance to listen for requests on, default is a link local ip address of value 169.254.20.10
- Fix dry-run output in kubectl apply –prune (#69344, @zegl)
- kubectl run now generates apps/v1 deployments by default (#71006, @liggitt)
- `kubeadm reset` now outputs instructions about manual iptables rules cleanup. (#70874, @rdodev)
- Recognize newer docker versions without -ce/-ee suffix: 18.09.0 (#71001, @thomas-riccardi)
- "unfinished_work_microseconds" is added to the workqueue metrics; it can be used to detect stuck worker threads. (kube-controller-manager runs many workqueues.) (#70884, @lavalamp)
- add readiness gates in extended output for pods (#70775, @freehan)
- add `Ready` column and improve human-readable output of Deployments and StatefulSets (#70466, @Pingan2017)
- Kubeadm now respects the custom image registry configuration across joins and upgrades. Kubeadm passes the custom registry to the kubelet

for a custom pause container. (#70603, @chuckha)

- kubeadm: deprecate the DynamicKubeletConfig feature gate. The functionality is still accessible by using the kubeadm alpha kubelet enable-dynamic command. (#70849, @yagonobre)
- Add `kubelet_container_log_size_bytes` metric representing the log file size of a container. (#70749, @brancz)
- kubeadm: remove the AuditPolicyConfiguration feature gate (#70807, @Klaven)
- Kubeadm: attributes for join –control-plane workflow are now grouped into a dedicated JoinControlPlane struct (#70870, @fabriziopandini)
- Addon configuration is introduced in the kubeadm config API, while feature flag CoreDNS is now deprecated. (#70024, @fabriziopandini)
- Fixes ability for admin/edit/view users to see controller revisions, needed for kubectl rollout commands (#70699, @liggitt)
- kubeadm pre-pulls Etcd image only if external Etcd is not used and (#70743, @bart0sh)
  - –etcd-upgrade=false is not specified
- Add support for CRD conversion webhook (#67006, @mbohlool)
- Delete node lease if the corresponding node is deleted (#70034, @wangzhen127)
- In a future release the kubectl convert command will be deprecated. (#70820, @seans3)
- kubeadm: UnifiedControlPlaneImage is replaced by UseHyperKubeImage boolean value. (#70793, @rosti)
- kubeadm v1beta1 API: InitConfiguration.APIEndpoint has been renamed to .LocalAPIEndpoint (#70761, @luxas)
- Breaking change: CSINodeInfo split into Spec and Status. New fields Available and VolumePluginMechanism added to CSINodeInfo csi-api object. CSIDriverInfo no longer deleted on Driver uninstallation, instead Available flag is set to false. (#70515, @davidz627)
- GCERegionalPersistentDisk feature is GA now! (#70716, @jingxu97)
- Add secure port 10259 to the kube-scheduler (enabled by default) and deprecate old insecure port 10251. Without further flags self-signed certs are created on startup in memory. (#69663, @sttts)
- `--feature-gates` argument has been removed from the `kubeadm join` command. Feature gates will be retrieved from the cluster configuration during the join process. (#70755, @ereslibre)
- [kubeadm] Updates version of CoreDNS to 1.2.6 (#70796, @detiber)
- kubelet: When node lease feature is enabled, kubelet reports node status to api server only if there is some change or it didn't report over last report interval. (#69753, @wangzhen127)
- Self hosted is no longer supported in the standard workflow. The feature flags have been removed and your self hosted cluster is no longer able to upgrade via kubeadm. (#69878, @Klaven)
- vSphereVolume implements Raw Block Volume Support (#68761, @fanzhangio)

- [GCE] Filter out spammy audit logs from cluster autoscaler. (#70696, @loburm)
- CRD supports multi-version Schema, Subresources and AdditionalPrintColumns (NOTE that CRDs created prior to 1.13 populated the top-level additionalPrinterColumns field by default. To apply an update that changes to per-version additionalPrinterColumns, the top-level additionalPrinterColumns field must be explicitly set to null). (#70211, @roycaihw)
- Fixes a bug in previous releases where a pod could be placed inside another pod's cgroup when specifying –cgroup-root (#70678, @dashpole)
- Upgrade golang.org/x/net image to release-branch.go1.10 (#70663, @wenjiaswe)
- New addon in addon manager that automatically installs CSI CRDs if CSIDriverRegistry or CSINodeInfo feature gates are true. (#70193, @saad-ali)
- delegated authorization can now allow unrestricted access for `system:masters` like the main kube-apiserver (#70671, @deads2k)
- Update to use go1.11.2 (#70665, @cblecker)
- Add dns capabilities for Windows CNI plugins: (#67435, @feiskyer)
  - "dns" {
  - "servers": ["10.0.0.10"],
  - "searches": ["default.svc.cluster.local","svc.cluster.local","cluster.local"],
  - "options": []
  - }
- The VolumeScheduling feature is GA. The VolumeScheduling feature gate is deprecated and will be removed in a future release. (#70673, @msau42)
- Go clients created from a kubeconfig that specifies a TokenFile now periodically reload the token from the specified file. (#70606, @mikedanese)
- kubeadm: validate kubeconfig files in case of external CA mode. (#70537, @yagonobre)
- kube-apiserver: `--audit-webhook-version` and `--audit-log-version` now default to `audit.k8s.io/v1` if unspecified (#70476, @charrywanganthony)
- kubeadm: timeoutForControlPlane is introduced as part of the API Server config, that controls the timeout for the wait for control plane to be up. Default value is 4 minutes. (#70480, @rosti)
- kubeadm: The writable config file option for extra volumes is renamed to readOnly with a reversed meaning. With readOnly defaulted to false (as in pod specs). (#70495, @rosti)
- remove retry operation on attach/detach azure disk (#70568, @andyzhangx)
- Fix CSI volume limits not showing up in node's capacity and allocatable (#70540, @gnufied)
- Flex volume plugins now support expandvolume (to increase underlying volume capacity) and expanfs (resize filesystem) commands that Flex plugin authors can implement to support expanding in use Flex PersistentVolumes (#67851, @aniket-s-kulkarni)

- kubeadm: Control plane component configs are separated into ClusterConfiguration sub-structs. (#70371, @rosti)
- The `MountPropagation` feature is unconditionally enabled in v1.13, and can no longer be disabled. (#68230, @bertinatto)
- add azure UltraSSD, StandardSSD disk type support (#70477, @andyzhangx)
- The OwnerReferencesPermissionEnforcement admission plugin now checks authorization for the correct scope (namespaced or cluster-scoped) of the owner resource type. Previously, it always checked permissions at the same scope as the child resource. (#70389, @caesarxuchao)
- Ensure orphan public IPs on Azure deleted when service recreated with the same name. (#70463, @feiskyer)
- `kubectl apply` can now change a deployment strategy from rollout to recreate without explicitly clearing the rollout-related fields (#70436, @liggitt)
- Fix cloud-controller-manager crash when using OpenStack provider and PersistentVolume initializing controller (#70459, @mvladev)

v1.13.0-alpha.3

Documentation

Downloads for v1.13.0-alpha.3

filename	sha512 hash
kubernetes.tar.gz	`1d50cfd34306ace7354516125c45f8c546bba3ca5081af2b21969b535967d302821c06e7`
kubernetes-src.tar.gz	`bf097b99d7b9af15bc1d592ee3782da1e811d8eb68dc9ae9d287589ce9174d3743beaf51`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`77778ae2887eda52ee716fb0e843c17b2705b1284a67cdf53f91292e`
kubernetes-client-darwin-amd64.tar.gz	`b3399767df12b71ee4b7b30126bd8001a0c1396161eb7535d797fd58`
kubernetes-client-linux-386.tar.gz	`5ef0d318ff8da28c332ae25164e5a441272d2ee8ef2ac26438a47fe3`
kubernetes-client-linux-amd64.tar.gz	`1f429eae5b0b1e39b1d4d30e3220a82d0ae6672a6f4b34a05246c3ef`
kubernetes-client-linux-arm.tar.gz	`5583aecdc9b4a54a4aa904fc1de66400f50628969e31b5a63ab1d3b6`
kubernetes-client-linux-arm64.tar.gz	`2453b9100c06b11e8c424d59cfd1c5e111c22b596191a9cfb0b330d1`
kubernetes-client-linux-ppc64le.tar.gz	`4991ec4c19a82d50caed78bc8db51e7cdcd1f2896dfcaa45d84f347a`
kubernetes-client-linux-s390x.tar.gz	`c55f2802afb2e5d261bb26b6c396df8ebe6b95913ddab1e124cf177f`
kubernetes-client-windows-386.tar.gz	`df78465267e35ef078c3c0fd33f8898a9df26fbf411df3ed3283fbdc`

filename	sha512 hash
kubernetes-client-windows-amd64.tar.gz	`5b93fdaaa931ef8e24196e53c484f91ef9e50b7d11e1053ccb61b2d6`

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	`922a93ce677e686e594c11db75e1969c995b23062bba511bff4a43d3a5`
kubernetes-server-linux-arm.tar.gz	`5dd550b58dedf25df020e66f1526e80c50b46d2df3ddd241bd02b6ebf1`
kubernetes-server-linux-arm64.tar.gz	`3e1037e71d85a74cd5d40dd836bd442b2dcc457f8ccc8247e4537f3dec`
kubernetes-server-linux-ppc64le.tar.gz	`a89c46b558613ad09efe44a81574ad18157a787d1e9c5d09c98d3911b4`
kubernetes-server-linux-s390x.tar.gz	`47a68668e38ac1b8cb801f4bff3b15060cd88801f446ebfbf06125dbc9`

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	`74d5d46ac6ba336fa8aaf55d0a15860f6ebde2ff58d377ca93063593c`
kubernetes-node-linux-arm.tar.gz	`90372cb5270ffe6179d5d7efd3fff6aa029f73853805038fef1a6f926`
kubernetes-node-linux-arm64.tar.gz	`09693303a1a8489d9599d32f7fbf549d18f31eb53671fa2ed342fe508`
kubernetes-node-linux-ppc64le.tar.gz	`cbdb3b9ffd9be524ec0b38d72b0545b6dd1b3b789747f41a661fd7cbe`
kubernetes-node-linux-s390x.tar.gz	`fc296b386bc03bf10773559118cd4a3d5be3d4c296f09748507fac812`
kubernetes-node-windows-amd64.tar.gz	`ae79c62fcb0654a62606d65cf131188d93e4a10787a862e7b0363269`

Changelog since v1.13.0-alpha.2

Other notable changes

- kubelet –system-reserved and –kube-reserved are supported now on Windows nodes (#69960, @feiskyer)
- CSI drivers now have access to mountOptions defined on the storage class when attaching volumes. (#67898, @bswartz)
- The `kubectl plugin list` command will now display discovered plugin paths in the same order as they are found in a user's PATH variable. (#70443, @juanvallejo)
- Handle Windows named pipes in host mounts. (#69484, @ddebroy)
- kubeadm: Multiple API server endpoints support upon join is removed as it is now redundant. (#69812, @rosti)
- OpenAPI spec marks delete request's body parameter as optional (#70032, @iamneha)
- kube-controller-manager and cloud-controller-manager now hold generated serving certificates in-memory unless a writeable location is specified

with –cert-dir (#69884, @liggitt)
- Scheduler only activates unschedulable pods if node's scheduling related properties change. (#70366, @mlmhl)
- –api-audiences now defaults to the –service-account-issuer if the issuer is provided but the API audience is not. (#70308, @mikedanese)
- Refactor scheduler_test.go to use a fake k8s client. (#70290, @tossmilestone)
- `kubectl rollout undo` now returns errors when attempting to rollback a deployment to a non-existent revision (#70039, @liggitt)
  - `kubectl rollout undo` no longer uses the deprecated extensions/v1beta1 rollback API, which means that Events are no longer emitted when rolling back a deployment
- - The builtin system:csi-external-provisioner and system:csi-external-attacher cluster roles (#69868, @pohly)
  - are deprecated and will not be updated for deployments of CSI sidecar container versions >= 0.4.
  - Deployments with the current CSI sidecar containers have to provide their own RBAC
  - definitions. The reason is that the rules depend on how the sidecar containers are used,
  - which is defined by the deployment.
- Use debian-base instead of busybox as base image for server images (#70245, @ixdy)
- add support for projected volume in describe function (#70158, @WanLinghao)
- Speedup process lookup in /proc (#66367, @cpuguy83)
- Kubeadm reset now clean up custom etcd data path (#70003, @yagonobre)
- We changed when the `metadata.generation` of a custom resource (CR) increments. (#69059, @caesarxuchao)
  - If the CR participates the spec/status convention, the metadata.generation of the CR increments when there is any change, except for the changes to the metadata or the changes to the status.
  - If the CR does not participate the spec/status convention, the metadata.generation of the CR increments when there is any change to the CR, except for changes to the metadata.
  - A CR is considered to participate the spec/status convention if and only if the "CustomResourceSubresources" feature gate is turned on and the CRD has `.spec.subresources.status={}`.
- Improve Azure instance metadata handling by adding caches. (#70353, @feiskyer)
- adding cn-northwest-1 for AWS China Ningxia region (#70155, @pahud)
- "kubectl get" no longer exits before printing all of its results if an error is found (#70311, @juanvallejo)
- kubeadm now automatically creates a new stacked etcd member when joining a new control plane node (does not applies to external etcd) (#69486,

@fabriziopandini)
- Critical pod annotation is deprecated. Pod priority should be used instead to mark pods as critical. (#70298, @bsalamat)
- Display the usage of ephemeral-storage when using `kubectl describe node` (#70268, @Pingan2017)
- Added functionality to enable br_netfilter and ip_forward for debian packages to improve kubeadm support for CRI runtime besides Docker. (#70152, @ashwanikhemani)
- Add regions ap-northeast-3 and eu-west-3 to the list of well known AWS regions. (#70252, @nckturner)
- Remove kube-controller-manager flag '–insecure-experimental-approve-all-kubelet-csrs-for-group'(deprecated in v1.7) (#69209, @Pingan2017)
- GCE/GKE load balancer health check default interval changes from 2 seconds to 8 seconds, unhealthyThreshold to 3. (#70099, @grayluck)
  - Health check parameters are configurable to be bigger than default values.
- The kubectl wait command must handle when a watch returns an error vs closing by printing out the error and retrying the watch. (#69389, @smarterclayton)
- Updates to use debian-iptables v11.0, debian-hyperkube-base 0.12.0, and kube-addon-manager:v8.9. (#70209, @ixdy)
- Fixed patch/update operations on multi-version custom resources (#70087, @liggitt)
- When `--rotate-server-certificates` is enabled, kubelet will no longer request a new certificate on startup if the current certificate on disk is satisfactory. (#69991, @agunnerson-ibm)
- - Support for passing unknown provider names to the E2E test binaries is going to be deprecated. Use `--provider=skeleton` (no ssh access) or `--provider=local` (local cluster with ssh) instead. (#70141, @pohly)
- Add scheduler benchmark tests for PodAffinity and NodeAffinity. (#69898, @Huang-Wei)
- fix azure disk attachment error on Linux (#70002, @andyzhangx)

v1.13.0-alpha.2

Documentation

Downloads for v1.13.0-alpha.2

filename	sha512 hash
kubernetes.tar.gz	`cbe7ef29c7e7bbed82e173289f5f84d7a85ee4965cc5b7ccd16cf8236a3b8171bb8f5201`
kubernetes-src.tar.gz	`8b0b8e1b635cd849c2974d755fe174f0ce8fe8c690721d8ac6312683bbd2ca2c6f7eada3`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	fca661a5001e7f368374d0805f20910be24baa485bf4ae5d993185b9
kubernetes-client-darwin-amd64.tar.gz	d31dfea475981c7f7b758c7f201aa5b866db48d87942c79d0a12d464
kubernetes-client-linux-386.tar.gz	fecf8362c572fff48952fd2748ddcb9d375462cb484670cda4fda138
kubernetes-client-linux-amd64.tar.gz	136cb82ac94bcd791d56e997a948a7e1bee4af03bcc69ce9c835895c
kubernetes-client-linux-arm.tar.gz	e561c37895edef44614ecd59f497d393275ee62455b6269b169a8918
kubernetes-client-linux-arm64.tar.gz	c0d5eb49763e8bf50b5e8e3785c7889fecbd8bf7c0b3c18250fa894a
kubernetes-client-linux-ppc64le.tar.gz	a5a8c150af163e7c726662eeddfc3de8e43f123daaa100b8e82c9bc7
kubernetes-client-linux-s390x.tar.gz	fd162e0244e107f1892d79029f3452cdba84d8616ad1b15eebe197af
kubernetes-client-windows-386.tar.gz	e01fedec8f700e037bc43cb13bc916b85601cd1c9361a0f63fd27092
kubernetes-client-windows-amd64.tar.gz	d2601efcfa6a4ba8a017e9cac571fb454b21b7700a7b3f8e2fbabdd5

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	4dda298d44bc309f250c067e9282eea37903838a140cf5abf6f861dca6
kubernetes-server-linux-arm.tar.gz	e9c3bdf60272399bc6f85a15bbc55cd69db389c223b275661ddcab4ae8
kubernetes-server-linux-arm64.tar.gz	d0a1701a34365f939799b6ea676129acdcfa1582bcf50e82a9751d9aaf
kubernetes-server-linux-ppc64le.tar.gz	1a23473960aaaa639e796020741b63c11dad8a93903926e80c871814b8
kubernetes-server-linux-s390x.tar.gz	293d0eb93e2ed641d0c1e26d58423670c04c307dddb034a9fc252043ab

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	e3cba71c2b2d151cdcc44937c1bea083ee0ceb829e7feb25cd37edd4c
kubernetes-node-linux-arm.tar.gz	28d7f1cf4fecdc72da7f5f19836cc06bb08182f8e8fb1641dc01e9299
kubernetes-node-linux-arm64.tar.gz	cfe22b11502cd857f0e277e7c1af08e6202f7ffc36f852c6154159bdc
kubernetes-node-linux-ppc64le.tar.gz	195a6785c49af419361a8901c99bb6613a6578a8eac5e8f08ec280776
kubernetes-node-linux-s390x.tar.gz	51f5b5ed47b50f5188d9e2f57b03555492d3e490494842247fa04fe81
kubernetes-node-windows-amd64.tar.gz	d2690d57cd485c0c7ebe425464ad59f2c7722870abd6f264ea7fae65a

Changelog since v1.13.0-alpha.1

Other notable changes

- Corrected family type (inet6) for ipsets in ipv6-only clusters (#68436, @uablrek)

- Corrects check for non-Azure managed nodes with the Azure cloud provider (#70135, @marc-sensenich)
- Windows runtime endpoints is now switched to 'npipe:////./pipe/dockershim' from 'tcp://localhost:3735'. (#69516, @feiskyer)
- The caBundle and service fields in admission webhook API objects now correctly indicate they are optional (#70138, @liggitt)
- The –service-account-api-audiences on kube-apiserver is deprecated in favor of –api-audiences. (#70105, @mikedanese)
- kubeadm: fix unnecessary upgrades caused by undefined order of Volumes and VolumeMounts in manifests (#70027, @bart0sh)
- kubeadm: Implemented preflight check to ensure that number of CPUs (#70048, @bart0sh)
  - on the master node is not less than required.
- Reduce memory utilization of admission webhook metrics by removing resource related labels. (#69895, @jpbetz)
- kubeadm: Introduce config print init/join-defaults that deprecate config print-defaults by decoupling init and join configs. (#69617, @rosti)
- Images based on debian-base no longer include the libsystemd0 package. This should have no user-facing impact. (#69995, @ixdy)
  - Additionally, the addon-manager image is updated to use kubectl v1.11.3.
- fix 'kubeadm upgrade' infinite loop waiting for pod restart (#69886, @bart0sh)
- add more logging for azure disk diagnostics (#70012, @andyzhangx)
- Fluentd: concatenate long logs (#68012, @desaintmartin)
- CoreDNS is now the default DNS server in kube-up deployments. (#69883, @chrisohaver)
- Optimizes calculating stats when only CPU and Memory stats are returned from Kubelet stats/summary http endpoint. (#68841, @krzysztofjastrzebski)
- kubeadm: Fix node join taints. (#69846, @andrewrynhard)
- Opt out of chowning and chmoding from kubectl cp. (#69573, @bjhaid)
- support Azure premium file for azure file plugin (#69718, @andyzhangx)
- `TaintBasedEvictions` feature is promoted to beta. (#69824, @HuangWei)
- improves memory use and performance when processing large numbers of pods containing tolerations (#65350, @liggitt)
- Add dynamic audit configuration api (#67547, @pbarker)
- Promote resource limits priority function to beta (#69437, @ravisantoshgudimetla)
- Fix cluster autoscaler addon permissions so it can access batch/job. (#69858, @losipiuk)
- change default azure file mount permission to 0777 (#69854, @andyzhangx)
- kubeadm: JoinConfiguration now houses the discovery options in a nested Discovery structure, which in turn has a couple of other nested structures to house more specific options (BootstrapTokenDiscovery and FileDiscov-

ery) (#67763, @rosti)
- Fix tests to use fsync instead of sync (#69755, @mrunalp)
- kube-proxy argument `hostname-override` can be used to override hostname defined in the configuration file (#69340, @stevesloka)
- kube-apiserver: the `--deserialization-cache-size` flag is no longer used, is deprecated, and will be removed in a future release (#69842, @liggitt)
- Add support for JSON patch in fake client (#69330, @vaikas-google)

v1.13.0-alpha.1

Documentation

Downloads for v1.13.0-alpha.1

filename	sha512 hash
kubernetes.tar.gz	`9f8a34b54a22ea4d7925c2f8d0e0cb2e2005486b1ed89e594bc0100ec7202fc247b89c5c`
kubernetes-src.tar.gz	`a27a7c254d3677c823bd6fd1d0d5f9b1e78ccf807837173669a0079b0812a23444d646d8`

Client Binaries

filename	sha512 hash
kubernetes-client-darwin-386.tar.gz	`d77d33c6d6357b99089f65e1c9ec3cabdcf526ec56e87bdee6b09a8c`
kubernetes-client-darwin-amd64.tar.gz	`5b4a586defa2ba0ea7c8893dedfe48cae52a2cd324bcb311a3877e27`
kubernetes-client-linux-386.tar.gz	`d50572fbb716393004ad2984a15043d2dfadedd16ae03a73fc856532`
kubernetes-client-linux-amd64.tar.gz	`12ab709e574228f170a2ee2686e18dcbfcf59f64599b2ab9047c2ed6`
kubernetes-client-linux-arm.tar.gz	`3a8c75b62cf9e6476417246d4aaeda5a13b74bc073444fc3649198b9`
kubernetes-client-linux-arm64.tar.gz	`0f5b5956850f11a826d59d226b6a22645ca1f63893cd33c17dfe004b`
kubernetes-client-linux-ppc64le.tar.gz	`06c60dd2e4e8d1ab45474a5b85345b4f644d0c1c66e167596c6c91bd`
kubernetes-client-linux-s390x.tar.gz	`4630e9e523beb02d8d3900c71b3306561c2d119d588399c93d578184`
kubernetes-client-windows-386.tar.gz	`0c0fcc9c492aceb00ff7fd3c10ba228c7bb10d6139b75ceecd8f8553`
kubernetes-client-windows-amd64.tar.gz	`3548a6d8618c6c7c8042ae8c3eb69654314392c46f839de24ab72d9f`

Server Binaries

filename	sha512 hash
kubernetes-server-linux-amd64.tar.gz	`9dbf2343ef9539b7d4d73949bcd9eef6f46ece59e97fa3390a0e695d0c`
kubernetes-server-linux-arm.tar.gz	`a985f3c302246df9bff4b927a2596d209c19fb2f245aa5cb5de189b6a9`

filename	sha512 hash
kubernetes-server-linux-arm64.tar.gz	`80d20df07e6a29b7aedccbd4e26c1c0565b2a1c3146e1a5bb2ebd2e8cf`
kubernetes-server-linux-ppc64le.tar.gz	`7d45ed3aa8b36e9e666b334ff3ed3de238caea34b4a92b5e1a61a6e722`
kubernetes-server-linux-s390x.tar.gz	`30698478fab2fe7daccac97917b0b21b018c194ec39b005728f8cddf77`

Node Binaries

filename	sha512 hash
kubernetes-node-linux-amd64.tar.gz	`4497d14ac81677b43f0b75a457890c1f3bb8745a39875f58d53c734be`
kubernetes-node-linux-arm.tar.gz	`a3b0357db50e0dec7b0474816fec287388adabc76cc309a40dee9bc73`
kubernetes-node-linux-arm64.tar.gz	`43af8ec4c5f2a1e2baa8cd13817e127fb6a3576dd811a30c4cc5f04d8`
kubernetes-node-linux-ppc64le.tar.gz	`840354219b3e59ed05b5b44cbbf4d45ccc4c0d74044e28c8a557ca75c`
kubernetes-node-linux-s390x.tar.gz	`796ca2e6855bd942a9a63d93f847ae62c5ee74195e041b60b89ee7d0e`
kubernetes-node-windows-amd64.tar.gz	`96d666e8446d09088bdcb440559035118dce07a2d9f5718856192fd80`

Changelog since v1.12.0

Action Required

- kube-apiserver: the deprecated `--etcd-quorum-read` flag has been removed, and quorum reads are always enabled when fetching data from etcd. (#69527, @liggitt)
- Moved staging/src/k8s.io/client-go/tools/bootstrap to staging/src/k8s… (#67356, @yliaog)
- [action required] kubeadm: The `v1alpha2` config API has been removed. (#69055, @fabriziopandini)
  - Please convert your `v1alpha2` configuration files to `v1alpha3` using the
  - `kubeadm config migrate` command of kubeadm v1.12.x

Other notable changes

- Refactor factory_test.go to use a fake k8s client. (#69412, @tossmilestone)
- kubeadm: fix a case where fetching a kubernetesVersion from the internet still happened even if some commands don't need it. (#69645, @neolit123)
- Add tolerations for Stackdriver Logging and Metadata Agents. (#69737, @qingling128)
- Fix a bug in the scheduler that could cause the scheduler to go to an infinite loop when all nodes in a zone are removed. (#69758, @bsalamat)

- Dry-run is promoted to Beta and will be enabled by default. (#69644, @apelisse)
- `kubectl get priorityclass` now prints value column by default. (#69431, @Huang-Wei)
- Added a new container based image for running e2e tests (#69368, @dims)
- Remove the deprecated –google-json-key flag from kubelet. (#69354, @yujuhong)
- kube-apiserver: fixes `procMount` field incorrectly being marked as required in openapi schema (#69694, @jessfraz)
- The `LC_ALL` and `LC_MESSAGES` env vars can now be used to set desired locale for `kubectl` while keeping `LANG` unchanged. (#69500, @m1kola)
- Add ability to control primary GID of containers through Pod Spec and PodSecurityPolicy (#67802, @krmayankk)
- NodeLifecycleController: Now node lease renewal is treated as the heartbeat signal from the node, in addition to NodeStatus Update. (#69241, @wangzhen127)
- [GCE] Enable by default audit logging truncating backend. (#68288, @loburm)
- Enable insertId generation, and update Stackdriver Logging Agent image to 0.5-1.5.36-1-k8s. This help reduce log duplication and guarantee log order. (#68920, @qingling128)
- Move NodeInfo utils into pkg/scheduler/cache. (#69495, @wgliang)
- adds dynamic shared informers to write generic, non-generated controllers (#69308, @p0lyn0mial)
- Move CacheComparer to pkg/scheduler/internal/cache/comparer. (#69317, @wgliang)
- Updating OWNERS list for vSphere Cloud Provider. (#69187, @Sandeep-Pissay)
- The default storage class annotation for the storage addons has been changed to use the GA variant (#68345, @smelchior)
- Upgrade to etcd 3.3 client (#69322, @jpbetz)
- fix GetVolumeLimits log flushing issue (#69558, @andyzhangx)
- It is now possible to use named ports in the `kubectl port-forward` command (#69477, @m1kola)
- kubeadm: fix a possible scenario where kubeadm can pull much newer control-plane images (#69301, @neolit123)
- test/e2e/e2e.test: (#69105, @pohly) * -viper-config can be used to set also the options defined by command line flags * the default config file is “e2e.yaml/toml/json/…” and the test starts when no such config is found (as before) but if -viper-config is used, the config file must exist * -viper-config can be used to select a file with full path, with or without file suffix * the csiImageVersion/Registry flags were renamed to storage.csi.imageVersion/Registry
- Move FakeCache to pkg/scheduler/internal/cache/fake. (#69318, @wgliang)
- The “kubectl cp” command now supports path shortcuts (../) in remote

paths. (#65189, @juanvallejo)
- Fixed subpath in containerized kubelet. (#69565, @jsafrane)
- The runtimeHandler field on the RuntimeClass resource now accepts the empty string. (#69550, @tallclair)
- Kubelet can now parse PEM file containing both TLS certificate and key in arbitrary order. Previously key was always required to be first. (#69536, @awly)
- Scheduling conformance tests related to daemonsets should set the annotation that relaxes node selection restrictions, if any are set. This ensures conformance tests can run on a wider array of clusters. (#68793, @aveshagarwal)
- Replace Parallelize with function ParallelizeUntil and formally deprecate the Parallelize. (#68403, @wgliang)
- Move scheduler cache interface and implementation to pkg/scheduler/internal/cache. (#68968, @wgliang)
- Update to use go1.11.1 (#69386, @cblecker)
- Any external provider should be aware the cloud-provider interface should be imported from :- (#68310, @cheftako)
  - cloudprovider "k8s.io/cloud-provider"
- kubeadm: Fix a crash if the etcd local alpha phase is called when the configuration contains an external etcd cluster (#69420, @ereslibre)
- kubeadm now allows mixing of init/cluster and join configuration in a single YAML file (although a warning gets printed in this case). (#69426, @rosti)
- Code-gen: Remove lowercasing for project imports (#68484, @jsturtevant)
- Fix client cert setup in delegating authentication logic (#69430, @DirectXMan12)
- service.beta.kubernetes.io/aws-load-balancer-internal now supports true and false values, previously it only supported non-empty strings (#69436, @mcrute)
- OpenAPI spec and API reference now reflect dryRun query parameter for POST/PUT/PATCH operations (#69359, @roycaihw)
- kubeadm: Add a `v1beta1` API. (#69289, @fabriziopandini)
- kube-apiserver has removed support for the `etcd2` storage backend (deprecated since v1.9). Existing clusters must migrate etcd v2 data to etcd v3 storage before upgrading to v1.13. (#69310, @liggitt)
- List operations against the API now return internal server errors instead of partially complete lists when a value cannot be transformed from storage. The updated behavior is consistent with all other operations that require transforming data from storage such as watch and get. (#69399, @mikedanese)
- `kubectl wait` now supports condition value checks other than true using `--for condition=available=false` (#69295, @deads2k)
- CCM server will not listen insecurely if secure port is specified (#68982, @aruneli)

- Bump cluster-proportional-autoscaler to 1.3.0 (#69338, @MrHohn)
  - - Rebase docker image on scratch.
- fix inconsistency in windows kernel proxy when updating HNS policy. (#68923, @delulu)
- Fixes the sample-apiserver so that its BanFlunder admission plugin can be used. (#68417, @MikeSpreitzer)
- Fixed CSIDriver API object to allow missing fields. (#69331, @jsafrane)
- Bump addon-manager to v8.8 (#69337, @MrHohn)
  - - Rebase docker image on debian-base:0.3.2.
- Update defaultbackend image to 1.5. Users should concentrate on updating scripts to the new version. (#69120, @aledbf)
- Bump Dashboard version to v1.10.0 (#68450, @jeefy)
- Fixed panic on iSCSI volume tear down. (#69140, @jsafrane)
- Update defaultbackend to v1.5 (#69334, @bowei)
- Remove unused chaosclient. (#68409, @wgliang)
- Enable AttachVolumeLimit feature (#69225, @gnufied)
- Update crictl to v1.12.0 (#69033, @feiskyer)
- Wait for pod failed event in subpath test. (#69300, @mrunalp)
- [GCP] Added env variables to control CPU requests of kube-controller-manager and kube-scheduler. (#68823, @loburm)
- Bump up pod short start timeout to 2 minutes. (#69291, @mrunalp)
- Use the mounted "/var/run/secrets/kubernetes.io/serviceaccount/token" as the token file for running in-cluster based e2e testing. (#69273, @dims)
- apiservice availability related to networking glitches are corrected faster (#68678, @deads2k)
- extract volume attachment status checking operation as a common function when attaching a CSI volume (#68931, @mlmhl)
- PodSecurityPolicy objects now support a `MayRunAs` rule for `fsGroup` and `supplementalGroups` options. This allows specifying ranges of allowed GIDs for pods/containers without forcing a default GID the way `MustRunAs` does. This means that a container to which such a policy applies to won't use any fsGroup/supplementalGroup GID if not explicitly specified, yet a specified GID must still fall in the GID range according to the policy. (#65135, @stlaz)
- Images for cloud-controller-manager, kube-apiserver, kube-controller-manager, and kube-scheduler now contain a minimal /etc/nsswitch.conf and should respect /etc/hosts for lookups (#69238, @BenTheElder)
- add deprecation warning for all cloud providers (#69171, @andrewsykim)
- IPVS proxier mode now support connection based graceful termination. (#66012, @Lion-Wei)
- Fix panic in kubectl rollout commands (#69150, @soltysh)
- Add fallbacks to ARM API when getting empty node IP from Azure IMDS (#69077, @feiskyer)
- Deduplicate PATH items when reading plugins. (#69089, @soltysh)
- Adds permissions for startup of an on-cluster kube-controller-manager (#69062, @dghubble)

- Fixes issue #68899 where pods might schedule on an unschedulable node. (#68984, @k82cn)
- Returns error if NodeGetInfo fails. (#68979, @xing-yang)
- Pod disruption budgets shouldn't be checked for terminal pods while evicting (#68892, @ravisantoshgudimetla)
- Fix scheduler crashes when Prioritize Map function returns error. (#68563, @DylanBLE)
- kubeadm: create control plane with ClusterFirstWithHostNet DNS policy (#68890, @andrewrynhard)
- Reduced excessive logging from fluentd-gcp-scaler. (#68837, @x13n)
- adds dynamic lister (#68748, @p0lyn0mial)
- kubectl: add the –no-headers flag to `kubectl top ...` (#67890, @WanLinghao)
- Restrict redirect following from the apiserver to same-host redirects, and ignore redirects in some cases. (#66516, @tallclair)
- Fixed pod cleanup when /var/lib/kubelet is a symlink. (#68741, @jsafrane)
- Add "only_cpu_and_memory" GET parameter to /stats/summary http handler in kubelet. If parameter is true then only cpu and memory will be present in response. (#67829, @krzysztof-jastrzebski)
- Start synchronizing pods after network is ready. (#68752, @krzysztof-jastrzebski)
- kubectl has gained new –profile and –profile-output options to output go profiles (#68681, @dlespiau)
- Provides FSGroup capability on FlexVolume driver. It allows to disable the VolumeOwnership operation when volume is mounted (#68680, @benoitf)
- Apply _netdev mount option on bind mount (#68626, @gnufied)
- fix UnmountDevice failure on Windows (#68608, @andyzhangx)
- Allows changing nodeName in endpoint update. (#68575, @prameshj)
- kube-apiserver would return 400 Bad Request when it couldn't decode a json patch. (#68346, @CaoShuFeng)
  - kube-apiserver would return 422 Unprocessable Entity when a json patch couldn't be applied to one object.
- remove unused ReplicasetControllerOptions (#68121, @dixudx)
- Pass signals to fluentd process (#68064, @gianrubio)
- Flex drivers by default do not produce metrics. Flex plugins can enable metrics collection by setting the capability 'supportsMetrics' to true. Make sure the file system can support fs stat to produce metrics in this case. (#67508, @brahmaroutu)
- Use monotonically increasing generation to prevent scheduler equivalence cache race. (#67308, @cofyc)
- Fix kubelet service file permission warning (#66669, @daixiang0)
- Add prometheus metric for scheduling throughput. (#64526, @misterikkit)
- Get public IP for Azure vmss nodes. (#68498, @feiskyer)

- test/integration: add a basic test for covering CronJobs (#66937, @mortent)
- Make service environment variables optional (#68754, @bradhoekstra)

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 20, 2019 at 7:05 PM PST by Fix relative links issue in English content (#13307) (Page History)

Edit This Page

Set up a High Availability etcd cluster with kubeadm

Kubeadm defaults to running a single member etcd cluster in a static pod managed by the kubelet on the control plane node. This is not a high availability setup as the etcd cluster contains only one member and cannot sustain any members becoming unavailable. This task walks through the process of creating a high availability etcd cluster of three members that can be used as an external etcd when using kubeadm to set up a kubernetes cluster.

- Before you begin
- Setting up the cluster
- What's next

Before you begin

- Three hosts that can talk to each other over ports 2379 and 2380. This document assumes these default ports. However, they are configurable through the kubeadm config file.

- Each host must have docker, kubelet, and kubeadm installed.
- Some infrastructure to copy files between hosts. For example `ssh` and `scp` can satisfy this requirement.

Setting up the cluster

The general approach is to generate all certs on one node and only distribute the *necessary* files to the other nodes.

> **Note:** kubeadm contains all the necessary crytographic machinery to generate the certificates described below; no other cryptographic tooling is required for this example.

1. Configure the kubelet to be a service manager for etcd.

   Since etcd was created first, you must override the service priority by creating a new unit file that has higher precedence than the kubeadm-provided kubelet unit file.

   ```
   cat << EOF > /etc/systemd/system/kubelet.service.d/20-etcd-service-manager.conf
   [Service]
   ExecStart=
   ExecStart=/usr/bin/kubelet --address=127.0.0.1 --pod-manifest-path=/etc/kubernetes/mani
   Restart=always
   EOF

   systemctl daemon-reload
   systemctl restart kubelet
   ```

2. Create configuration files for kubeadm.

   Generate one kubeadm configuration file for each host that will have an etcd member running on it using the following script.

   ```
   # Update HOST0, HOST1, and HOST2 with the IPs or resolvable names of your hosts
   export HOST0=10.0.0.6
   export HOST1=10.0.0.7
   export HOST2=10.0.0.8

   # Create temp directories to store files that will end up on other hosts.
   mkdir -p /tmp/${HOST0}/ /tmp/${HOST1}/ /tmp/${HOST2}/

   ETCDHOSTS=(${HOST0} ${HOST1} ${HOST2})
   NAMES=("infra0" "infra1" "infra2")

   for i in "${!ETCDHOSTS[@]}"; do
   HOST=${ETCDHOSTS[$i]}
   NAME=${NAMES[$i]}
   ```

```
cat << EOF > /tmp/${HOST}/kubeadmcfg.yaml
apiVersion: "kubeadm.k8s.io/v1beta1"
kind: ClusterConfiguration
etcd:
    local:
        serverCertSANs:
        - "${HOST}"
        peerCertSANs:
        - "${HOST}"
        extraArgs:
            initial-cluster: ${NAMES[0]}=https://${ETCDHOSTS[0]}:2380,${NAMES[1]}=https
            initial-cluster-state: new
            name: ${NAME}
            listen-peer-urls: https://${HOST}:2380
            listen-client-urls: https://${HOST}:2379
            advertise-client-urls: https://${HOST}:2379
            initial-advertise-peer-urls: https://${HOST}:2380
EOF
done
```

3. Generate the certificate authority

   If you already have a CA then the only action that is copying the CA's `crt` and `key` file to `/etc/kubernetes/pki/etcd/ca.crt` and `/etc/kubernetes/pki/etcd/ca.key`. After those files have been copied, proceed to the next step, "Create certificates for each member".

   If you do not already have a CA then run this command on `$HOST0` (where you generated the configuration files for kubeadm).

   ```
   kubeadm init phase certs etcd-ca
   ```

   This creates two files

   - `/etc/kubernetes/pki/etcd/ca.crt`
   - `/etc/kubernetes/pki/etcd/ca.key`

4. Create certificates for each member

   ```
   kubeadm init phase certs etcd-server --config=/tmp/${HOST2}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-peer --config=/tmp/${HOST2}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST2}/kubeadmcfg.yaml
   kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST2}/kubeadmcfg.yaml
   cp -R /etc/kubernetes/pki /tmp/${HOST2}/
   # cleanup non-reusable certificates
   find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete

   kubeadm init phase certs etcd-server --config=/tmp/${HOST1}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-peer --config=/tmp/${HOST1}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST1}/kubeadmcfg.yaml
   ```

```
kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST1}/kubeadmcfg.yaml
cp -R /etc/kubernetes/pki /tmp/${HOST1}/
find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete

kubeadm init phase certs etcd-server --config=/tmp/${HOST0}/kubeadmcfg.yaml
kubeadm init phase certs etcd-peer --config=/tmp/${HOST0}/kubeadmcfg.yaml
kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST0}/kubeadmcfg.yaml
kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST0}/kubeadmcfg.yaml
# No need to move the certs because they are for HOST0

# clean up certs that should not be copied off this host
find /tmp/${HOST2} -name ca.key -type f -delete
find /tmp/${HOST1} -name ca.key -type f -delete
```

5. Copy certificates and kubeadm configs

   The certificates have been generated and now they must be moved to their respective hosts.

   ```
   USER=ubuntu
   HOST=${HOST1}
   scp -r /tmp/${HOST}/* ${USER}@${HOST}:
   ssh ${USER}@${HOST}
   USER@HOST $ sudo -Es
   root@HOST $ chown -R root:root pki
   root@HOST $ mv pki /etc/kubernetes/
   ```

6. Ensure all expected files exist

   The complete list of required files on `$HOST0` is:

   ```
   /tmp/${HOST0}
     kubeadmcfg.yaml
   ---
   /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
   ```

   On `$HOST1`:

   ```
   $HOME
   ```

```
  kubeadmcfg.yaml
---
/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
```

On `$HOST2`

```
$HOME
  kubeadmcfg.yaml
---
/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
```

7. Create the static pod manifests

   Now that the certificates and configs are in place it's time to create the manifests. On each host run the `kubeadm` command to generate a static manifest for etcd.

```
root@HOST0 $ kubeadm init phase etcd local --config=/tmp/${HOST0}/kubeadmcfg.yaml
root@HOST1 $ kubeadm init phase etcd local --config=/home/ubuntu/kubeadmcfg.yaml
root@HOST2 $ kubeadm init phase etcd local --config=/home/ubuntu/kubeadmcfg.yaml
```

8. Optional: Check the cluster health

```
docker run --rm -it \
--net host \
-v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd:${ETCD_TAG} etcdctl \
--cert-file /etc/kubernetes/pki/etcd/peer.crt \
--key-file /etc/kubernetes/pki/etcd/peer.key \
--ca-file /etc/kubernetes/pki/etcd/ca.crt \
```

```
--endpoints https://${HOST0}:2379 cluster-health
...
cluster is healthy
```

- Set `${ETCD_TAG}` to the version tag of your etcd image. For example `v3.2.24`.
- Set `${HOST0}`to the IP address of the host you are testing.

What's next

Once you have a working 3 member etcd cluster, you can continue setting up a highly available control plane using the external etcd method with kubeadm.

Feedback

Was this page helpful?

Yes

No

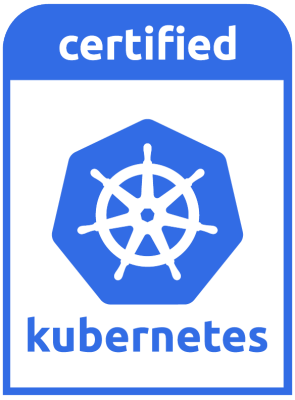
Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Create an Issue Edit This Page

Page last modified on February 11, 2019 at 1:35 AM PST by docs: fix typo (#12568) (Page History)

Edit This Page

Installing kubeadm



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

Before you begin

- One or more machines running one of:
  - Ubuntu 16.04+
  - Debian 9
  - CentOS 7
  - RHEL 7
  - Fedora 25/26 (best-effort)
  - HypriotOS v1.0.1+
  - Container Linux (tested with 1800.6.0)
- 2 GB or more of RAM per machine (any less will leave little room for your apps)
- 2 CPUs or more
- Full network connectivity between all machines in the cluster (public or private network is fine)
- Unique hostname, MAC address, and product_uuid for every node. See here for more details.

- Certain ports are open on your machines. See here for more details.
- Swap disabled. You **MUST** disable swap in order for the kubelet to work properly.

Verify the MAC address and product_uuid are unique for every node

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

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

Check network adapters

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

Check required ports

Master node(s)

Protocol	Direction	Port Range	Purpose	Used By
TCP	Inbound	6443*	Kubernetes API server	All
TCP	Inbound	2379-2380	etcd server client API	kube-apiserver, etcd
TCP	Inbound	10250	Kubelet API	Self, Control plane
TCP	Inbound	10251	kube-scheduler	Self
TCP	Inbound	10252	kube-controller-manager	Self

Worker node(s)

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

** Default port range for NodePort Services.

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

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

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

Installing runtime

Since v1.6.0, Kubernetes has enabled the use of CRI, Container Runtime Interface, by default. The container runtime used by default is Docker, which is enabled through the built-in `dockershim` CRI implementation inside of the `kubelet`.

Other CRI-based runtimes include:

- containerd (CRI plugin built into containerd)
- cri-o
- frakti

Refer to the CRI installation instructions for more information.

Installing kubeadm, kubelet and kubectl

You will install these packages on all of your machines:

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

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

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

For more information on version skews, see:

- Kubernetes version and version-skew policy
- Kubeadm-specific version skew policy

- Ubuntu, Debian or HypriotOS
- CentOS, RHEL or Fedora
- Container Linux

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

```
cat <<EOF > /etc/yum.repos.d/kubernetes.repo
[kubernetes]
name=Kubernetes
baseurl=https://packages.cloud.google.com/yum/repos/kubernetes-el7-x86_64
enabled=1
gpgcheck=1
repo_gpgcheck=1
gpgkey=https://packages.cloud.google.com/yum/doc/yum-key.gpg https://packages.cloud.google.c
exclude=kube*
EOF

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

yum install -y kubelet kubeadm kubectl --disableexcludes=kubernetes

systemctl enable --now kubelet
```

Note:

- Setting SELinux in permissive mode by running `setenforce 0` and `sed ...` effectively disables it. This is required to allow containers to access the host filesystem, which is needed by pod networks for example. You have to do this until SELinux support is improved in the kubelet.
- Some users on RHEL/CentOS 7 have reported issues with traffic being routed incorrectly due to iptables being bypassed. You should ensure

`net.bridge.bridge-nf-call-iptables` is set to 1 in your `sysctl` config, e.g.

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

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

Install CNI plugins (required for most pod network):

```
CNI_VERSION="v0.6.0"
mkdir -p /opt/cni/bin
curl -L "https://github.com/containernetworking/plugins/releases/download/${CNI_VERSION}/cni
```

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

```
CRICTL_VERSION="v1.11.1"
mkdir -p /opt/bin
curl -L "https://github.com/kubernetes-incubator/cri-tools/releases/download/${CRICTL_VERSIO
```

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

```
RELEASE="$(curl -sSL https://dl.k8s.io/release/stable.txt)"

mkdir -p /opt/bin
cd /opt/bin
curl -L --remote-name-all https://storage.googleapis.com/kubernetes-release/release/${RELEAS
chmod +x {kubeadm,kubelet,kubectl}

curl -sSL "https://raw.githubusercontent.com/kubernetes/kubernetes/${RELEASE}/build/debs/kub
mkdir -p /etc/systemd/system/kubelet.service.d
curl -sSL "https://raw.githubusercontent.com/kubernetes/kubernetes/${RELEASE}/build/debs/10-
```

Enable and start `kubelet`:

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

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

Configure cgroup driver used by kubelet on Master Node

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

runtime.

If you are using a different CRI, you have to modify the file `/etc/default/kubelet` with your `cgroup-driver` value, like so:

```
KUBELET_EXTRA_ARGS=--cgroup-driver=<value>
```

This file will be used by `kubeadm init` and `kubeadm join` to source extra user defined arguments for the kubelet.

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

Restarting the kubelet is required:

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

Troubleshooting

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

What's next

- Using kubeadm to Create a Cluster

Feedback

Was this page helpful?

Yes

No

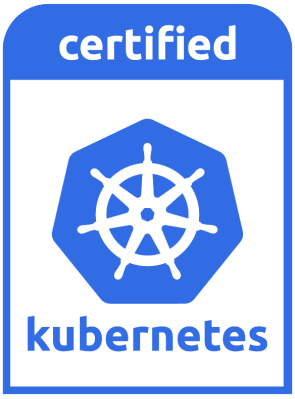
Thanks for the feedback. If you have a specific, answerable question about how to use Kubernetes, ask it on Stack Overflow. Open an issue in the GitHub repo if you want to report a problem or suggest an improvement.

Create an Issue Edit This Page

Page last modified on February 22, 2019 at 1:24 AM PST by Make k8s.io/docs/home to support i18n (#12570) (Page History)

Edit This Page

Creating a single master cluster with kubeadm



kubeadm helps you bootstrap a minimum viable Kubernetes cluster that conforms to best practices. With kubeadm, your cluster should pass Kubernetes Conformance tests. Kubeadm also supports other cluster lifecycle functions, such as upgrades, downgrade, and managing bootstrap tokens.

Because you can install kubeadm on various types of machine (e.g. laptop, server, Raspberry Pi, etc.), it's well suited for integration with provisioning systems such as Terraform or Ansible.

kubeadm's simplicity means it can serve a wide range of use cases:

- New users can start with kubeadm to try Kubernetes out for the first time.
- Users familiar with Kubernetes can spin up clusters with kubeadm and test their applications.
- Larger projects can include kubeadm as a building block in a more complex system that can also include other installer tools.

kubeadm is designed to be a simple way for new users to start trying Kubernetes out, possibly for the first time, a way for existing users to test their application on and stitch together a cluster easily, and also to be a building block in other ecosystem and/or installer tool with a larger scope.

You can install *kubeadm* very easily on operating systems that support installing deb or rpm packages. The responsible SIG for kubeadm, SIG Cluster Lifecycle, provides these packages pre-built for you, but you may also build them from source for other OSes.

kubeadm Maturity

Area	Maturity Level

Area	Maturity Level
Command line UX	GA
Implementation	GA
Config file API	beta
CoreDNS	GA
kubeadm alpha subcommands	alpha
High availability	alpha
DynamicKubeletConfig	alpha
Self-hosting	alpha

kubeadm's overall feature state is **GA**. Some sub-features, like the configuration file API are still under active development. The implementation of creating the cluster may change slightly as the tool evolves, but the overall implementation should be pretty stable. Any commands under `kubeadm alpha` are by definition, supported on an alpha level.

Support timeframes

Kubernetes releases are generally supported for nine months, and during that period a patch release may be issued from the release branch if a severe bug or security issue is found. Here are the latest Kubernetes releases and the support timeframe; which also applies to `kubeadm`.

Kubernetes version	Release month	End-of-life-month
v1.6.x	March 2017	December 2017
v1.7.x	June 2017	March 2018
v1.8.x	September 2017	June 2018
v1.9.x	December 2017	September 2018
v1.10.x	March 2018	December 2018
v1.11.x	June 2018	March 2019
v1.12.x	September 2018	June 2019
v1.13.x	December 2018	September 2019

- Before you begin
- Objectives
- Instructions
- Tear down
- Maintaining a cluster
- Explore other add-ons
- What's next

- Feedback
- Version skew policy
- kubeadm works on multiple platforms
- Limitations
- Troubleshooting

Before you begin

- One or more machines running a deb/rpm-compatible OS, for example Ubuntu or CentOS
- 2 GB or more of RAM per machine. Any less leaves little room for your apps.
- 2 CPUs or more on the master
- Full network connectivity among all machines in the cluster. A public or private network is fine.

Objectives

- Install a single master Kubernetes cluster or high availability cluster
- Install a Pod network on the cluster so that your Pods can talk to each other

Instructions

Installing kubeadm on your hosts

See "Installing kubeadm".

> **Note:**
>
> If you have already installed kubeadm, run `apt-get update && apt-get upgrade` or `yum update` to get the latest version of kubeadm.
>
> When you upgrade, the kubelet restarts every few seconds as it waits in a crashloop for kubeadm to tell it what to do. This crashloop is expected and normal. After you initialize your master, the kubelet runs normally.

Initializing your master

The master is the machine where the control plane components run, including etcd (the cluster database) and the API server (which the kubectl CLI communicates with).

1. Choose a pod network add-on, and verify whether it requires any arguments to be passed to kubeadm initialization. Depending on which third-party provider you choose, you might need to set the `--pod-network-cidr` to a provider-specific value. See Installing a pod network add-on.
2. (Optional) Unless otherwise specified, kubeadm uses the network interface associated with the default gateway to advertise the master's IP. To use a different network interface, specify the `--apiserver-advertise-address=<ip-address>` argument to `kubeadm init`. To deploy an IPv6 Kubernetes cluster using IPv6 addressing, you must specify an IPv6 address, for example `--apiserver-advertise-address=fd00::101`
3. (Optional) Run `kubeadm config images pull` prior to `kubeadm init` to verify connectivity to gcr.io registries.

Now run:

```
kubeadm init <args>
```

More information

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

For a complete list of configuration options, see the configuration file documentation.

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

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

If you join a node with a different architecture to your cluster, create a separate Deployment or DaemonSet for `kube-proxy` and `kube-dns` on the node. This is because the Docker images for these components do not currently support multi-architecture.

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

```
[init] Using Kubernetes version: vX.Y.Z
[preflight] Running pre-flight checks
[preflight] Pulling images required for setting up a Kubernetes cluster
[preflight] This might take a minute or two, depending on the speed of your internet connect
[preflight] You can also perform this action in beforehand using 'kubeadm config images pull
[kubelet-start] Writing kubelet environment file with flags to file "/var/lib/kubelet/kubead
[kubelet-start] Writing kubelet configuration to file "/var/lib/kubelet/config.yaml"
```

```
[kubelet-start] Activating the kubelet service
[certs] Using certificateDir folder "/etc/kubernetes/pki"
[certs] Generating "etcd/ca" certificate and key
[certs] Generating "etcd/server" certificate and key
[certs] etcd/server serving cert is signed for DNS names [kubeadm-master localhost] and IPs
[certs] Generating "etcd/healthcheck-client" certificate and key
[certs] Generating "etcd/peer" certificate and key
[certs] etcd/peer serving cert is signed for DNS names [kubeadm-master localhost] and IPs [1
[certs] Generating "apiserver-etcd-client" certificate and key
[certs] Generating "ca" certificate and key
[certs] Generating "apiserver" certificate and key
[certs] apiserver serving cert is signed for DNS names [kubeadm-master kubernetes kubernetes
[certs] Generating "apiserver-kubelet-client" certificate and key
[certs] Generating "front-proxy-ca" certificate and key
[certs] Generating "front-proxy-client" certificate and key
[certs] Generating "sa" key and public key
[kubeconfig] Using kubeconfig folder "/etc/kubernetes"
[kubeconfig] Writing "admin.conf" kubeconfig file
[kubeconfig] Writing "kubelet.conf" kubeconfig file
[kubeconfig] Writing "controller-manager.conf" kubeconfig file
[kubeconfig] Writing "scheduler.conf" kubeconfig file
[control-plane] Using manifest folder "/etc/kubernetes/manifests"
[control-plane] Creating static Pod manifest for "kube-apiserver"
[control-plane] Creating static Pod manifest for "kube-controller-manager"
[control-plane] Creating static Pod manifest for "kube-scheduler"
[etcd] Creating static Pod manifest for local etcd in "/etc/kubernetes/manifests"
[wait-control-plane] Waiting for the kubelet to boot up the control plane as static Pods fro
[apiclient] All control plane components are healthy after 31.501735 seconds
[uploadconfig] storing the configuration used in ConfigMap "kubeadm-config" in the "kube-sys
[kubelet] Creating a ConfigMap "kubelet-config-X.Y" in namespace kube-system with the config
[patchnode] Uploading the CRI Socket information "/var/run/dockershim.sock" to the Node API
[mark-control-plane] Marking the node kubeadm-master as control-plane by adding the label "n
[mark-control-plane] Marking the node kubeadm-master as control-plane by adding the taints
[bootstrap-token] Using token: <token>
[bootstrap-token] Configuring bootstrap tokens, cluster-info ConfigMap, RBAC Roles
[bootstraptoken] configured RBAC rules to allow Node Bootstrap tokens to post CSRs in order
[bootstraptoken] configured RBAC rules to allow the csrapprover controller automatically app
[bootstraptoken] configured RBAC rules to allow certificate rotation for all node client cer
[bootstraptoken] creating the "cluster-info" ConfigMap in the "kube-public" namespace
[addons] Applied essential addon: CoreDNS
[addons] Applied essential addon: kube-proxy

Your Kubernetes master has initialized successfully!

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

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

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

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

  kubeadm join <master-ip>:<master-port> --token <token> --discovery-token-ca-cert-hash sha2
```

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

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

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

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

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

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

Installing a pod network add-on

> **Caution:** This section contains important information about installation and deployment order. Read it carefully before proceeding.

You must install a pod network add-on so that your pods can communicate with each other.

The network must be deployed before any applications. Also, CoreDNS will not start up before a network is installed. kubeadm only supports Container Network Interface (CNI) based networks (and does not support kubenet).

Several projects provide Kubernetes pod networks using CNI, some of which also support Network Policy. See the add-ons page for a complete list of available network add-ons. - IPv6 support was added in CNI v0.6.0. - CNI bridge and

local-ipam are the only supported IPv6 network plugins in Kubernetes version 1.9.

Note that kubeadm sets up a more secure cluster by default and enforces use of RBAC. Make sure that your network manifest supports RBAC.

Also, beware, that your Pod network must not overlap with any of the host networks as this can cause issues. If you find a collision between your network plugin's preferred Pod network and some of your host networks, you should think of a suitable CIDR replacement and use that during `kubeadm init` with `--pod-network-cidr` and as a replacement in your network plugin's YAML.

You can install a pod network add-on with the following command:

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

You can install only one pod network per cluster.

- Choose one...
- Calico
- Canal
- Cilium
- Flannel
- Kube-router
- Romana
- Weave Net
- JuniperContrail/TungstenFabric
- Contiv-VPP

Please select one of the tabs to see installation instructions for the respective third-party Pod Network Provider.

For more information about using Calico, see Quickstart for Calico on Kubernetes, Installing Calico for policy and networking, and other related resources.

For Calico to work correctly, you need to pass `--pod-network-cidr=192.168.0.0/16` to `kubeadm init` or update the `calico.yml` file to match your Pod network. Note that Calico works on `amd64`, `arm64`, and `ppc64le` only.

```
kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installatio
kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installatio
```

Canal uses Calico for policy and Flannel for networking. Refer to the Calico documentation for the official getting started guide.

For Canal to work correctly, `--pod-network-cidr=10.244.0.0/16` has to be passed to `kubeadm init`. Note that Canal works on `amd64` only.

```
kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installatio
kubectl apply -f https://docs.projectcalico.org/v3.3/getting-started/kubernetes/installatio
```

For more information about using Cilium with Kubernetes, see Kubernetes Install guide for Cilium.

These commands will deploy Cilium with its own etcd managed by etcd operator.

Note: If you are running kubeadm in a single node please untaint it so that etcd-operator pods can be scheduled in the control-plane node.

```
kubectl taint nodes <node-name> node-role.kubernetes.io/master:NoSchedule-
```

To deploy Cilium you just need to run:

```
kubectl create -f https://raw.githubusercontent.com/cilium/cilium/v1.4/examples/kubernetes/1
```

Once all Cilium pods are marked as `READY`, you start using your cluster.

```
$ kubectl get pods -n kube-system --selector=k8s-app=cilium
NAME           READY   STATUS    RESTARTS   AGE
cilium-drxkl   1/1     Running   0          18m
```

For `flannel` to work correctly, you must pass `--pod-network-cidr=10.244.0.0/16` to `kubeadm init`.

Set `/proc/sys/net/bridge/bridge-nf-call-iptables` to `1` by running `sysctl net.bridge.bridge-nf-call-iptables=1` to pass bridged IPv4 traffic to iptables' chains. This is a requirement for some CNI plugins to work, for more information please see here.

Note that `flannel` works on `amd64`, `arm`, `arm64`, `ppc64le` and `s390x` under Linux. Windows (`amd64`) is claimed as supported in v0.11.0 but the usage is undocumented.

```
kubectl apply -f https://raw.githubusercontent.com/coreos/flannel/a70459be0084506e4ec919aa1c
```

For more information about `flannel`, see the CoreOS flannel repository on GitHub.

Set `/proc/sys/net/bridge/bridge-nf-call-iptables` to `1` by running `sysctl net.bridge.bridge-nf-call-iptables=1` to pass bridged IPv4 traffic to iptables' chains. This is a requirement for some CNI plugins to work, for more information please see here.

Kube-router relies on kube-controller-manager to allocate pod CIDR for the nodes. Therefore, use `kubeadm init` with the `--pod-network-cidr` flag.

Kube-router provides pod networking, network policy, and high-performing IP Virtual Server(IPVS)/Linux Virtual Server(LVS) based service proxy.

For information on setting up Kubernetes cluster with Kube-router using kubeadm, please see official setup guide.

Set `/proc/sys/net/bridge/bridge-nf-call-iptables` to `1` by running `sysctl net.bridge.bridge-nf-call-iptables=1` to pass bridged IPv4 traffic to iptables' chains. This is a requirement for some CNI plugins to work, for more information please see here.

The official Romana set-up guide is here.

Romana works on `amd64` only.

```
kubectl apply -f https://raw.githubusercontent.com/romana/romana/master/containerize/specs/r
```

Set `/proc/sys/net/bridge/bridge-nf-call-iptables` to `1` by running `sysctl net.bridge.bridge-nf-call-iptables=1` to pass bridged IPv4 traffic to iptables' chains. This is a requirement for some CNI plugins to work, for more information please see here.

The official Weave Net set-up guide is here.

Weave Net works on `amd64`, `arm`, `arm64` and `ppc64le` without any extra action required. Weave Net sets hairpin mode by default. This allows Pods to access themselves via their Service IP address if they don't know their PodIP.

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | base64 |
```

Provides overlay SDN solution, delivering multicloud networking, hybrid cloud networking, simultaneous overlay-underlay support, network policy enforcement, network isolation, service chaining and flexible load balancing.

There are multiple, flexible ways to install JuniperContrail/TungstenFabric CNI.

Kindly refer to this quickstart: TungstenFabric

Contiv-VPP employs a programmable CNF vSwitch based on FD.io VPP, offering feature-rich & high-performance cloud-native networking and services.

It implements k8s services and network policies in the user space (on VPP).

Please refer to this installation guide: Contiv-VPP Manual Installation

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

If your network is not working or CoreDNS is not in the Running state, check out our troubleshooting docs.

Control plane node isolation

By default, your cluster will not schedule pods on the master for security reasons. If you want to be able to schedule pods on the master, e.g. for a single-machine Kubernetes cluster for development, run:

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

With output looking something like:

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

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

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

Joining your nodes

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

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

```
kubeadm join --token <token> <master-ip>:<master-port> --discovery-token-ca-cert-hash sha256
```

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

```
kubeadm token list
```

The output is similar to this:

```
TOKEN                     TTL  EXPIRES              USAGES           DESCRIPTION            E
8ewj1p.9r9hcjoqgajrj4gi  23h  2018-06-12T02:51:28Z authentication,  The default bootstrap  s
                                                   signing          token generated by     b
                                                                    'kubeadm init'.        k
                                                                                           c
```

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

```
kubeadm token create
```

The output is similar to this:

```
5didvk.d09sbcov8ph2amjw
```

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

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

The output is similar to this:

```
8cb2de97839780a412b93877f8507ad6c94f73add17d5d7058e91741c9d5ec78
```

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

The output should look something like:

```
[preflight] Running pre-flight checks

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

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

Run 'kubectl get nodes' on the master to see this machine join.
```

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

(Optional) Controlling your cluster from machines other than the master

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

```
scp root@<master ip>:/etc/kubernetes/admin.conf .
kubectl --kubeconfig ./admin.conf get nodes
```

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

(Optional) Proxying API Server to localhost

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

```
scp root@<master ip>:/etc/kubernetes/admin.conf .
kubectl --kubeconfig ./admin.conf proxy
```

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

Tear down

To undo what kubeadm did, you should first drain the node and make sure that the node is empty before shutting it down.

Talking to the master with the appropriate credentials, run:

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

Then, on the node being removed, reset all kubeadm installed state:

```
kubeadm reset
```

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

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

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

```
ipvsadm -C
```

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

More options and information about the `kubeadm reset command`.

Maintaining a cluster

Instructions for maintaining kubeadm clusters (e.g. upgrades,downgrades, etc.) can be found here.

Explore other add-ons

See the list of add-ons to explore other add-ons, including tools for logging, monitoring, network policy, visualization & control of your Kubernetes cluster.

What's next

- Verify that your cluster is running properly with Sonobuoy
- Learn about kubeadm's advanced usage in the kubeadm reference documentation
- Learn more about Kubernetes concepts and `kubectl`.

- Configure log rotation. You can use **logrotate** for that. When using Docker, you can specify log rotation options for Docker daemon, for example `--log-driver=json-file --log-opt=max-size=10m --log-opt=max-file=5`. See Configure and troubleshoot the Docker daemon for more details.

Feedback

- For bugs, visit kubeadm Github issue tracker
- For support, visit kubeadm Slack Channel: #kubeadm
- General SIG Cluster Lifecycle Development Slack Channel: #sig-cluster-lifecycle
- SIG Cluster Lifecycle SIG information
- SIG Cluster Lifecycle Mailing List: kubernetes-sig-cluster-lifecycle

Version skew policy

The kubeadm CLI tool of version vX.Y may deploy clusters with a control plane of version vX.Y or vX.(Y-1). kubeadm CLI vX.Y can also upgrade an existing kubeadm-created cluster of version vX.(Y-1).

Due to that we can't see into the future, kubeadm CLI vX.Y may or may not be able to deploy vX.(Y+1) clusters.

Example: kubeadm v1.8 can deploy both v1.7 and v1.8 clusters and upgrade v1.7 kubeadm-created clusters to v1.8.

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

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

kubeadm works on multiple platforms

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

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

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

Limitations

Please note: kubeadm is a work in progress and these limitations will be addressed in due course.

1. The cluster created here has a single master, with a single etcd database running on it. This means that if the master fails, your cluster may lose data and may need to be recreated from scratch. Adding HA support (multiple etcd servers, multiple API servers, etc) to kubeadm is still a work-in-progress.

Workaround: regularly back up etcd. The etcd data directory configured by kubeadm is at `/var/lib/etcd` on the master.

Troubleshooting

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

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 20, 2019 at 7:05 PM PST by Fix relative links issue in English content (#13307) (Page History)

Edit This Page

Customizing control plane configuration with kubeadm

FEATURE STATE: `Kubernetes 1.12` stable

This feature is *stable*, meaning:

- The version name is vX where X is an integer.
- Stable versions of features will appear in released software for many subsequent versions.

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

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

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

1. Add the appropriate fields to your configuration.
2. Add the flags to override to the field.

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

- APIServer flags
- ControllerManager flags
- Scheduler flags

APIServer flags

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

Example usage:

```
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: v1.13.0
metadata:
  name: 1.13-sample
apiServer:
  extraArgs:
    advertise-address: 192.168.0.103
    anonymous-auth: false
    enable-admission-plugins: AlwaysPullImages,DefaultStorageClass
```

```
    audit-log-path: /home/johndoe/audit.log
```

ControllerManager flags

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

Example usage:

```
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: v1.13.0
metadata:
  name: 1.13-sample
controllerManager:
  extraArgs:
    cluster-signing-key-file: /home/johndoe/keys/ca.key
    bind-address: 0.0.0.0
    deployment-controller-sync-period: 50
```

Scheduler flags

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

Example usage:

```
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: v1.13.0
metadata:
  name: 1.13-sample
scheduler:
  extraArgs:
    address: 0.0.0.0
    config: /home/johndoe/schedconfig.yaml
    kubeconfig: /home/johndoe/kubeconfig.yaml
```

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on January 08, 2019 at 9:30 PM PST by adding feature state snip (#12134) (Page History)

Edit This Page

Options for Highly Available Topology

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

You can set up an HA cluster:

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

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

Stacked etcd topology

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

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

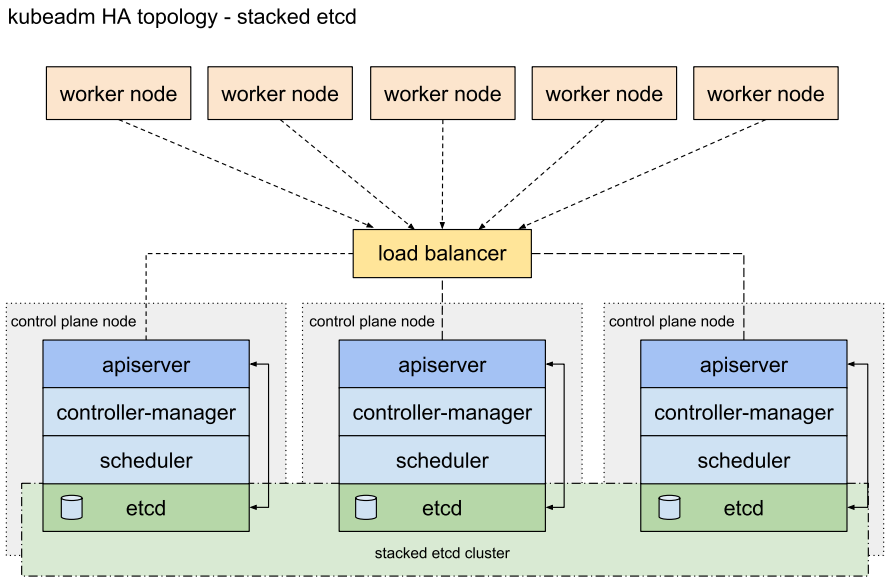
Each control plane node creates a local etcd member and this etcd member communicate only with the `kube-apiserver` of this node. The same applies to the local `kube-controller-manager` and `kube-scheduler` instances.

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

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

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

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



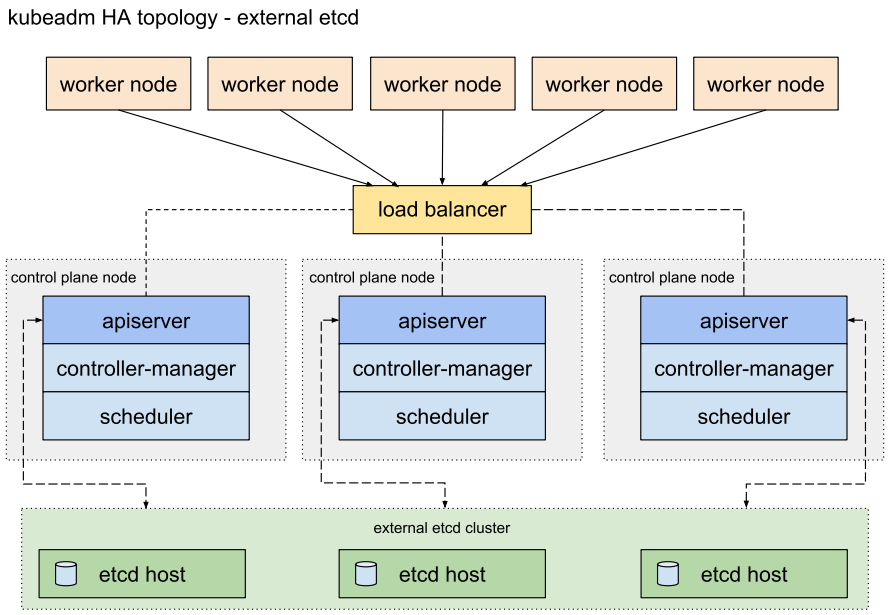
External etcd topology

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

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

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

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



What's next

- Set up a highly available cluster with kubeadm

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on December 03, 2018 at 7:21 PM PST by Official 1.13 Release Docs (#11401) (Page History)

Edit This Page

Creating Highly Available Clusters with kubeadm

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

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

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

Your clusters must run Kubernetes version 1.12 or later. You should also be aware that setting up HA clusters with kubeadm is still experimental and will be further simplified in future versions. You might encounter issues with upgrading your clusters, for example. We encourage you to try either approach, and provide us with feedback in the kubeadm issue tracker.

Note that the alpha feature gate `HighAvailability` is deprecated in v1.12 and removed in v1.13.

See also The HA upgrade documentation.

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

Before you begin

For both methods you need this infrastructure:

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

For the external etcd cluster only, you also need:

- Three additional machines for etcd members

  Note: The following examples run Calico as the Pod networking provider. If you run another networking provider, make sure to replace any default values as needed.

First steps for both methods

  Note: All commands on any control plane or etcd node should be run as root.

- Some CNI network plugins like Calico require a CIDR such as `192.168.0.0/16` and some like Weave do not. See the CNI network documentation. To add a pod CIDR set the `podSubnet: 192.168.0.0/16` field under the `networking` object of `ClusterConfiguration`.

Create load balancer for kube-apiserver

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

1. Create a kube-apiserver load balancer with a name that resolves to DNS.

   - In a cloud environment you should place your control plane nodes behind a TCP forwarding load balancer. This load balancer distributes traffic to all healthy control plane nodes in its target list. The health check for an apiserver is a TCP check on the port the kube-apiserver listens on (default value `:6443`).

   - It is not recommended to use an IP address directly in a cloud environment.

- The load balancer must be able to communicate with all control plane nodes on the apiserver port. It must also allow incoming traffic on its listening port.
- HAProxy can be used as a load balancer.
- Make sure the address of the load balancer always matches the address of kubeadm's `ControlPlaneEndpoint`.

2. Add the first control plane nodes to the load balancer and test the connection:

   ```
   nc -v LOAD_BALANCER_IP PORT
   ```

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

3. Add the remaining control plane nodes to the load balancer target group.

Configure SSH

SSH is required if you want to control all nodes from a single machine.

1. Enable ssh-agent on your main device that has access to all other nodes in the system:

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

2. Add your SSH identity to the session:

   ```
   ssh-add ~/.ssh/path_to_private_key
   ```

3. SSH between nodes to check that the connection is working correctly.

   - When you SSH to any node, make sure to add the `-A` flag:

     ```
     ssh -A 10.0.0.7
     ```

   - When using sudo on any node, make sure to preserve the environment so SSH forwarding works:

     ```
     sudo -E -s
     ```

Stacked control plane and etcd nodes

Steps for the first control plane node

1. On the first control plane node, create a configuration file called `kubeadm-config.yaml`:

```
apiVersion: kubeadm.k8s.io/v1beta1
kind: ClusterConfiguration
kubernetesVersion: stable
apiServer:
  certSANs:
  - "LOAD_BALANCER_DNS"
controlPlaneEndpoint: "LOAD_BALANCER_DNS:LOAD_BALANCER_PORT"
```

- `kubernetesVersion` should be set to the Kubernetes version to use. This example uses `stable`.
- `controlPlaneEndpoint` should match the address or DNS and port of the load balancer.
- It's recommended that the versions of kubeadm, kubelet, kubectl and Kubernetes match.

2. Make sure that the node is in a clean state:

   ```
   sudo kubeadm init --config=kubeadm-config.yaml
   ```

   You should see something like:

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

   kubeadm join 192.168.0.200:6443 --token j04n3m.octy8zely83cy2ts --discovery-token-ca-ce
   ```

3. Copy this output to a text file. You will need it later to join other control plane nodes to the cluster.

4. Apply the Weave CNI plugin:

   ```
   kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | bas
   ```

5. Type the following and watch the pods of the components get started:

   ```
   kubectl get pod -n kube-system -w
   ```

   - It's recommended that you join new control plane nodes only after the first node has finished initializing.

6. Copy the certificate files from the first control plane node to the rest:

   In the following example, replace `CONTROL_PLANE_IPS` with the IP addresses of the other control plane nodes.

   ```
   USER=ubuntu # customizable
   CONTROL_PLANE_IPS="10.0.0.7 10.0.0.8"
   for host in ${CONTROL_PLANE_IPS}; do
       scp /etc/kubernetes/pki/ca.crt "${USER}"@$host:
       scp /etc/kubernetes/pki/ca.key "${USER}"@$host:
       scp /etc/kubernetes/pki/sa.key "${USER}"@$host:
       scp /etc/kubernetes/pki/sa.pub "${USER}"@$host:
   ```

```
    scp /etc/kubernetes/pki/front-proxy-ca.crt "${USER}"@$host:
    scp /etc/kubernetes/pki/front-proxy-ca.key "${USER}"@$host:
    scp /etc/kubernetes/pki/etcd/ca.crt "${USER}"@$host:etcd-ca.crt
    scp /etc/kubernetes/pki/etcd/ca.key "${USER}"@$host:etcd-ca.key
    scp /etc/kubernetes/admin.conf "${USER}"@$host:
done
```

Caution: Copy only the certificates in the above list. kubeadm will take care of generating the rest of the certificates with the required SANs for the joining control-plane instances. If you copy all the certificates by mistake, the creation of additional nodes could fail due to a lack of required SANs.

Steps for the rest of the control plane nodes

1. Move the files created by the previous step where `scp` was used:

   ```
   USER=ubuntu # customizable
   mkdir -p /etc/kubernetes/pki/etcd
   mv /home/${USER}/ca.crt /etc/kubernetes/pki/
   mv /home/${USER}/ca.key /etc/kubernetes/pki/
   mv /home/${USER}/sa.pub /etc/kubernetes/pki/
   mv /home/${USER}/sa.key /etc/kubernetes/pki/
   mv /home/${USER}/front-proxy-ca.crt /etc/kubernetes/pki/
   mv /home/${USER}/front-proxy-ca.key /etc/kubernetes/pki/
   mv /home/${USER}/etcd-ca.crt /etc/kubernetes/pki/etcd/ca.crt
   mv /home/${USER}/etcd-ca.key /etc/kubernetes/pki/etcd/ca.key
   mv /home/${USER}/admin.conf /etc/kubernetes/admin.conf
   ```

   This process writes all the requested files in the `/etc/kubernetes` folder.

2. Start `kubeadm join` on this node using the join command that was previously given to you by `kubeadm init` on the first node. It should look something like this:

   ```
   sudo kubeadm join 192.168.0.200:6443 --token j04n3m.octy8zely83cy2ts --discovery-token-
   ```

   - Notice the addition of the `--experimental-control-plane` flag. This flag automates joining this control plane node to the cluster.

3. Type the following and watch the pods of the components get started:

   ```
   kubectl get pod -n kube-system -w
   ```

4. Repeat these steps for the rest of the control plane nodes.

External etcd nodes

Set up the etcd cluster

- Follow these instructions to set up the etcd cluster.

Set up the first control plane node

1. Copy the following files from any node from the etcd cluster to this node:

   ```
   export CONTROL_PLANE="ubuntu@10.0.0.7"
   +scp /etc/kubernetes/pki/etcd/ca.crt "${CONTROL_PLANE}":
   +scp /etc/kubernetes/pki/apiserver-etcd-client.crt "${CONTROL_PLANE}":
   +scp /etc/kubernetes/pki/apiserver-etcd-client.key "${CONTROL_PLANE}":
   ```

   - Replace the value of `CONTROL_PLANE` with the `user@host` of this machine.

2. Create a file called `kubeadm-config.yaml` with the following contents:

   ```
   apiVersion: kubeadm.k8s.io/v1beta1
   kind: ClusterConfiguration
   kubernetesVersion: stable
   apiServer:
     certSANs:
     - "LOAD_BALANCER_DNS"
   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
   ```

   - The difference between stacked etcd and external etcd here is that we are using the `external` field for `etcd` in the kubeadm config. In the case of the stacked etcd topology this is managed automatically.
   - Replace the following variables in the template with the appropriate values for your cluster:
     - `LOAD_BALANCER_DNS`
     - `LOAD_BALANCER_PORT`
     - `ETCD_0_IP`
     - `ETCD_1_IP`

- ETCD_2_IP

3. Run `kubeadm init --config kubeadm-config.yaml` on this node.
4. Write the join command that is returned to a text file for later use.
5. Apply the Weave CNI plugin:

```
kubectl apply -f "https://cloud.weave.works/k8s/net?k8s-version=$(kubectl version | bas
```

Steps for the rest of the control plane nodes

To add the rest of the control plane nodes, follow these instructions. The steps are the same as for the stacked etcd setup, with the exception that a local etcd member is not created.

To summarize:

- Make sure the first control plane node is fully initialized.
- Copy certificates between the first control plane node and the other control plane nodes.
- Join each control plane node with the join command you saved to a text file, plus add the `--experimental-control-plane` flag.

Common tasks after bootstrapping control plane

Install a pod network

Follow these instructions to install the pod network. Make sure this corresponds to whichever pod CIDR you provided in the master configuration file.

Install workers

Each worker node can now be joined to the cluster with the command returned from any of the `kubeadm init` commands. The flag `--experimental-control-plane` should not be added to worker nodes.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 26, 2019 at 12:40 AM PST by Removed duplicate 'Note' keyword (#12833) (Page History)

Edit This Page

Set up a High Availability etcd cluster with kubeadm

Kubeadm defaults to running a single member etcd cluster in a static pod managed by the kubelet on the control plane node. This is not a high availability setup as the etcd cluster contains only one member and cannot sustain any members becoming unavailable. This task walks through the process of creating a high availability etcd cluster of three members that can be used as an external etcd when using kubeadm to set up a kubernetes cluster.

- Before you begin
- Setting up the cluster
- What's next

Before you begin

- Three hosts that can talk to each other over ports 2379 and 2380. This document assumes these default ports. However, they are configurable through the kubeadm config file.
- Each host must have docker, kubelet, and kubeadm installed.
- Some infrastructure to copy files between hosts. For example `ssh` and `scp` can satisfy this requirement.

Setting up the cluster

The general approach is to generate all certs on one node and only distribute the *necessary* files to the other nodes.

Note: kubeadm contains all the necessary crytographic machinery to generate the certificates described below; no other cryptographic tooling is required for this example.

1. Configure the kubelet to be a service manager for etcd.

   Since etcd was created first, you must override the service priority by creating a new unit file that has higher precedence than the kubeadm-provided kubelet unit file.

   ```
   cat << EOF > /etc/systemd/system/kubelet.service.d/20-etcd-service-manager.conf
   [Service]
   ExecStart=
   ExecStart=/usr/bin/kubelet --address=127.0.0.1 --pod-manifest-path=/etc/kubernetes/mani
   Restart=always
   EOF

   systemctl daemon-reload
   systemctl restart kubelet
   ```

2. Create configuration files for kubeadm.

   Generate one kubeadm configuration file for each host that will have an etcd member running on it using the following script.

   ```
   # Update HOST0, HOST1, and HOST2 with the IPs or resolvable names of your hosts
   export HOST0=10.0.0.6
   export HOST1=10.0.0.7
   export HOST2=10.0.0.8

   # Create temp directories to store files that will end up on other hosts.
   mkdir -p /tmp/${HOST0}/ /tmp/${HOST1}/ /tmp/${HOST2}/

   ETCDHOSTS=(${HOST0} ${HOST1} ${HOST2})
   NAMES=("infra0" "infra1" "infra2")

   for i in "${!ETCDHOSTS[@]}"; do
   HOST=${ETCDHOSTS[$i]}
   NAME=${NAMES[$i]}
   cat << EOF > /tmp/${HOST}/kubeadmcfg.yaml
   apiVersion: "kubeadm.k8s.io/v1beta1"
   kind: ClusterConfiguration
   etcd:
       local:
           serverCertSANs:
           - "${HOST}"
           peerCertSANs:
           - "${HOST}"
           extraArgs:
   ```

```
            initial-cluster: ${NAMES[0]}=https://${ETCDHOSTS[0]}:2380,${NAMES[1]}=https
            initial-cluster-state: new
            name: ${NAME}
            listen-peer-urls: https://${HOST}:2380
            listen-client-urls: https://${HOST}:2379
            advertise-client-urls: https://${HOST}:2379
            initial-advertise-peer-urls: https://${HOST}:2380
EOF
done
```

3. Generate the certificate authority

   If you already have a CA then the only action that is copying the CA's `crt` and `key` file to `/etc/kubernetes/pki/etcd/ca.crt` and `/etc/kubernetes/pki/etcd/ca.key`. After those files have been copied, proceed to the next step, "Create certificates for each member".

   If you do not already have a CA then run this command on `$HOST0` (where you generated the configuration files for kubeadm).

   ```
   kubeadm init phase certs etcd-ca
   ```

   This creates two files

   - `/etc/kubernetes/pki/etcd/ca.crt`
   - `/etc/kubernetes/pki/etcd/ca.key`

4. Create certificates for each member

   ```
   kubeadm init phase certs etcd-server --config=/tmp/${HOST2}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-peer --config=/tmp/${HOST2}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST2}/kubeadmcfg.yaml
   kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST2}/kubeadmcfg.yaml
   cp -R /etc/kubernetes/pki /tmp/${HOST2}/
   # cleanup non-reusable certificates
   find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete

   kubeadm init phase certs etcd-server --config=/tmp/${HOST1}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-peer --config=/tmp/${HOST1}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST1}/kubeadmcfg.yaml
   kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST1}/kubeadmcfg.yaml
   cp -R /etc/kubernetes/pki /tmp/${HOST1}/
   find /etc/kubernetes/pki -not -name ca.crt -not -name ca.key -type f -delete

   kubeadm init phase certs etcd-server --config=/tmp/${HOST0}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-peer --config=/tmp/${HOST0}/kubeadmcfg.yaml
   kubeadm init phase certs etcd-healthcheck-client --config=/tmp/${HOST0}/kubeadmcfg.yaml
   kubeadm init phase certs apiserver-etcd-client --config=/tmp/${HOST0}/kubeadmcfg.yaml
   # No need to move the certs because they are for HOST0
   ```

```
# clean up certs that should not be copied off this host
find /tmp/${HOST2} -name ca.key -type f -delete
find /tmp/${HOST1} -name ca.key -type f -delete
```

5. Copy certificates and kubeadm configs

   The certificates have been generated and now they must be moved to their respective hosts.

   ```
   USER=ubuntu
   HOST=${HOST1}
   scp -r /tmp/${HOST}/* ${USER}@${HOST}:
   ssh ${USER}@${HOST}
   USER@HOST $ sudo -Es
   root@HOST $ chown -R root:root pki
   root@HOST $ mv pki /etc/kubernetes/
   ```

6. Ensure all expected files exist

   The complete list of required files on `$HOST0` is:

   ```
   /tmp/${HOST0}
     kubeadmcfg.yaml
   ---
   /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
   ```

   On `$HOST1`:

   ```
   $HOME
     kubeadmcfg.yaml
   ---
   /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
```

On $HOST2

```
$HOME
  kubeadmcfg.yaml
---
/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
```

7. Create the static pod manifests

   Now that the certificates and configs are in place it's time to create the manifests. On each host run the `kubeadm` command to generate a static manifest for etcd.

```
root@HOST0 $ kubeadm init phase etcd local --config=/tmp/${HOST0}/kubeadmcfg.yaml
root@HOST1 $ kubeadm init phase etcd local --config=/home/ubuntu/kubeadmcfg.yaml
root@HOST2 $ kubeadm init phase etcd local --config=/home/ubuntu/kubeadmcfg.yaml
```

8. Optional: Check the cluster health

```
docker run --rm -it \
--net host \
-v /etc/kubernetes:/etc/kubernetes quay.io/coreos/etcd:${ETCD_TAG} etcdctl \
--cert-file /etc/kubernetes/pki/etcd/peer.crt \
--key-file /etc/kubernetes/pki/etcd/peer.key \
--ca-file /etc/kubernetes/pki/etcd/ca.crt \
--endpoints https://${HOST0}:2379 cluster-health
...
cluster is healthy
```

   - Set `${ETCD_TAG}` to the version tag of your etcd image. For example `v3.2.24`.
   - Set `${HOST0}`to the IP address of the host you are testing.

What's next

Once you have a working 3 member etcd cluster, you can continue setting up a highly available control plane using the external etcd method with kubeadm.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 11, 2019 at 1:35 AM PST by docs: fix typo (#12568) (Page History)

Edit This Page

Configuring each kubelet in your cluster using kubeadm

FEATURE STATE: `Kubernetes 1.11` stable

This feature is *stable*, meaning:

- The version name is vX where X is an integer.
- Stable versions of features will appear in released software for many subsequent versions.

The lifecycle of the kubeadm CLI tool is decoupled from the kubelet, which is a daemon that runs on each node within the Kubernetes cluster. The kubeadm CLI tool is executed by the user when Kubernetes is initialized or upgraded, whereas the kubelet is always running in the background.

Since the kubelet is a daemon, it needs to be maintained by some kind of a init system or service manager. When the kubelet is installed using DEBs or RPMs, systemd is configured to manage the kubelet. You can use a different service manager instead, but you need to configure it manually.

Some kubelet configuration details need to be the same across all kubelets involved in the cluster, while other configuration aspects need to be set on a per-kubelet basis, to accommodate the different characteristics of a given machine, such as OS, storage, and networking. You can manage the configuration of your kubelets manually, but kubeadm now provides a `KubeletConfiguration` API type for managing your kubelet configurations centrally.

- Kubelet configuration patterns
- Configure kubelets using kubeadm
- The kubelet drop-in file for systemd
- Kubernetes binaries and package contents

Kubelet configuration patterns

The following sections describe patterns to kubelet configuration that are simplified by using kubeadm, rather than managing the kubelet configuration for each Node manually.

Propagating cluster-level configuration to each kubelet

You can provide the kubelet with default values to be used by `kubeadm init` and `kubeadm join` commands. Interesting examples include using a different CRI runtime or setting the default subnet used by services.

If you want your services to use the subnet `10.96.0.0/12` as the default for services, you can pass the `--service-cidr` parameter to kubeadm:

```
kubeadm init --service-cidr 10.96.0.0/12
```

Virtual IPs for services are now allocated from this subnet. You also need to set the DNS address used by the kubelet, using the `--cluster-dns` flag. This setting needs to be the same for every kubelet on every manager and Node in the cluster. The kubelet provides a versioned, structured API object that can configure most parameters in the kubelet and push out this configuration to each running kubelet in the cluster. This object is called **the kubelet's ComponentConfig**. The ComponentConfig allows the user to specify flags such as the cluster DNS IP addresses expressed as a list of values to a camelCased key, illustrated by the following example:

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

For more details on the ComponentConfig have a look at this section.

Providing instance-specific configuration details

Some hosts require specific kubelet configurations, due to differences in hardware, operating system, networking, or other host-specific parameters. The following list provides a few examples.

- The path to the DNS resolution file, as specified by the `--resolv-conf` kubelet configuration flag, may differ among operating systems, or depending on whether you are using `systemd-resolved`. If this path is wrong, DNS resolution will fail on the Node whose kubelet is configured incorrectly.
- The Node API object `.metadata.name` is set to the machine's hostname by default, unless you are using a cloud provider. You can use the `--hostname-override` flag to override the default behavior if you need to specify a Node name different from the machine's hostname.
- Currently, the kubelet cannot automatically detects the cgroup driver used by the CRI runtime, but the value of `--cgroup-driver` must match the cgroup driver used by the CRI runtime to ensure the health of the kubelet.
- Depending on the CRI runtime your cluster uses, you may need to specify different flags to the kubelet. For instance, when using Docker, you need to specify flags such as `--network-plugin=cni`, but if you are using an external runtime, you need to specify `--container-runtime=remote` and specify the CRI endpoint using the `--container-runtime-path-endpoint=<path>`.

You can specify these flags by configuring an individual kubelet's configuration in your service manager, such as systemd.

Configure kubelets using kubeadm

It is possible to configure the kubelet that kubeadm will start if a custom `KubeletConfiguration` API object is passed with a configuration file like so `kubeadm ... --config some-config-file.yaml`.

By calling `kubeadm config print-default --api-objects KubeletConfiguration` you can see all the default values for this structure.

Also have a look at the API reference for the kubelet ComponentConfig for more information on the individual fields.

Workflow when using `kubeadm init`

When you call `kubeadm init`, the kubelet configuration is marshalled to disk at `/var/lib/kubelet/config.yaml`, and also uploaded to a ConfigMap in the

cluster. The ConfigMap is named `kubelet-config-1.X`, where `.X` is the minor version of the Kubernetes version you are initializing. A kubelet configuration file is also written to `/etc/kubernetes/kubelet.conf` with the baseline cluster-wide configuration for all kubelets in the cluster. This configuration file points to the client certificates that allow the kubelet to communicate with the API server. This addresses the need to propagate cluster-level configuration to each kubelet.

To address the second pattern of providing instance-specific configuration details, kubeadm writes an environment file to `/var/lib/kubelet/kubeadm-flags.env`, which contains a list of flags to pass to the kubelet when it starts. The flags are presented in the file like this:

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

In addition to the flags used when starting the kubelet, the file also contains dynamic parameters such as the cgroup driver and whether to use a different CRI runtime socket (`--cri-socket`).

After marshalling these two files to disk, kubeadm attempts to run the following two commands, if you are using systemd:

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

If the reload and restart are successful, the normal `kubeadm init` workflow continues.

Workflow when using `kubeadm join`

When you run `kubeadm join`, kubeadm uses the Bootstrap Token credential perform a TLS bootstrap, which fetches the credential needed to download the `kubelet-config-1.X` ConfigMap and writes it to `/var/lib/kubelet/config.yaml`. The dynamic environment file is generated in exactly the same way as `kubeadm init`.

Next, `kubeadm` runs the following two commands to load the new configuration into the kubelet:

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

After the kubelet loads the new configuration, kubeadm writes the `/etc/kubernetes/bootstrap-kubelet.conf` KubeConfig file, which contains a CA certificate and Bootstrap Token. These are used by the kubelet to perform the TLS Bootstrap and obtain a unique credential, which is stored in `/etc/kubernetes/kubelet.conf`. When this file is written, the kubelet has finished performing the TLS Bootstrap.

The kubelet drop-in file for systemd

The configuration file installed by the kubeadm DEB or RPM package is written to `/etc/systemd/system/kubelet.service.d/10-kubeadm.conf` and is used by systemd.

```
[Service]
Environment="KUBELET_KUBECONFIG_ARGS=--bootstrap-kubeconfig=/etc/kubernetes/bootstrap-kubele
--kubeconfig=/etc/kubernetes/kubelet.conf"
Environment="KUBELET_CONFIG_ARGS=--config=/var/lib/kubelet/config.yaml"
# This is a file that "kubeadm init" and "kubeadm join" generates at runtime, populating
the KUBELET_KUBEADM_ARGS variable dynamically
EnvironmentFile=-/var/lib/kubelet/kubeadm-flags.env
# This is a file that the user can use for overrides of the kubelet args as a last resort. P
#the user should use the .NodeRegistration.KubeletExtraArgs object in the configuration file
# KUBELET_EXTRA_ARGS should be sourced from this file.
EnvironmentFile=-/etc/default/kubelet
ExecStart=
ExecStart=/usr/bin/kubelet $KUBELET_KUBECONFIG_ARGS $KUBELET_CONFIG_ARGS $KUBELET_KUBEADM_AR
```

This file specifies the default locations for all of the files managed by kubeadm for the kubelet.

- The KubeConfig file to use for the TLS Bootstrap is `/etc/kubernetes/bootstrap-kubelet.conf`, but it is only used if `/etc/kubernetes/kubelet.conf` does not exist.
- The KubeConfig file with the unique kubelet identity is `/etc/kubernetes/kubelet.conf`.
- The file containing the kubelet's ComponentConfig is `/var/lib/kubelet/config.yaml`.
- The dynamic environment file that contains `KUBELET_KUBEADM_ARGS` is sourced from `/var/lib/kubelet/kubeadm-flags.env`.
- The file that can contain user-specified flag overrides with `KUBELET_EXTRA_ARGS` is sourced from `/etc/default/kubelet` (for DEBs), or `/etc/sysconfig/kubelet` (for RPMs). `KUBELET_EXTRA_ARGS` is last in the flag chain and has the highest priority in the event of conflicting settings.

Kubernetes binaries and package contents

The DEB and RPM packages shipped with the Kubernetes releases are:

Package name	Description
`kubeadm`	Installs the `/usr/bin/kubeadm` CLI tool and the kubelet drop-in file for the kubelet.
`kubelet`	Installs the `/usr/bin/kubelet` binary.
`kubectl`	Installs the `/usr/bin/kubectl` binary.
`kubernetes-cni`	Installs the official CNI binaries into the `/opt/cni/bin` directory.
`cri-tools`	Installs the `/usr/bin/crictl` binary from the cri-tools git repository.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 27, 2019 at 7:49 PM PST by Fix the borken link in kubelet-integration.md page (#12864) (Page History)

Edit This Page

Troubleshooting kubeadm

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

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

- If you think your problem is a bug with kubeadm:
  - Go to github.com/kubernetes/kubeadm and search for existing issues.
  - If no issue exists, please open one and follow the issue template.
- If you are unsure about how kubeadm works, you can ask on Slack in #kubeadm, or open a question on StackOverflow. Please include relevant tags like `#kubernetes` and `#kubeadm` so folks can help you.
- `ebtables` or some similar executable not found during installation
- kubeadm blocks waiting for control plane during installation
- kubeadm blocks when removing managed containers
- Pods in `RunContainerError`, `CrashLoopBackOff` or `Error` state
- `coredns` (or `kube-dns`) is stuck in the `Pending` state
- `HostPort` services do not work
- Pods are not accessible via their Service IP
- TLS certificate errors
- Default NIC When using flannel as the pod network in Vagrant

- Non-public IP used for containers
- `coredns` pods have `CrashLoopBackOff` or `Error` state
- etcd pods restart continually
- Not possible to pass a comma separated list of values to arguments inside a `--component-extra-args` flag

`ebtables` or some similar executable not found during installation

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

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

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

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

kubeadm blocks waiting for control plane during installation

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

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

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

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

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

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

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

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

kubeadm blocks when removing managed containers

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

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

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

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

Inspecting the logs for docker may also be useful:

```
journalctl -ul docker
```

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

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

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

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

This is **expected** and part of the design. kubeadm is network provider-agnostic, so the admin should install the pod network solution of choice. You have to

install a Pod Network before CoreDNS may deployed fully. Hence the `Pending` state before the network is set up.

`HostPort` services do not work

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

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

For more information, see the CNI portmap documentation.

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

Pods are not accessible via their Service IP

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

TLS certificate errors

The following error indicates a possible certificate mismatch.

```
# kubectl get pods
Unable to connect to the server: x509: certificate signed by unknown authority (possibly bec
```

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

```
unset KUBECONFIG
```

Or set it to the default `KUBECONFIG` location:

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

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

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

Default NIC When using flannel as the pod network in Vagrant

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

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

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

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

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

Non-public IP used for containers

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

```
Error from server: Get https://10.19.0.41:10250/containerLogs/default/mysql-ddc65b868-glc5m/
```

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

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

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

The workaround is to tell `kubelet` which IP to use using `--node-ip`. When using Digital Ocean, it can be the public one (assigned to `eth0`) or the private one

(assigned to `eth1`) should you want to use the optional private network. The `KubeletExtraArgs` section of the kubeadm `NodeRegistrationOptions` structure can be used for this.

Then restart `kubelet`:

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

coredns pods have CrashLoopBackOff or Error state

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

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

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

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

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

etcd pods restart continually

If you encounter the following error:

```
rpc error: code = 2 desc = oci runtime error: exec failed: container_linux.go:247: starting
```

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

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

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

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

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

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

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

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

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

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

A known workaround is to use the kubeadm configuration file.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 20, 2019 at 7:05 PM PST by Fix relative links issue in English content (#13307) (Page History)

Edit This Page

Running Kubernetes on AWS EC2

This page describes how to install a Kubernetes cluster on AWS.

Before you begin

To create a Kubernetes cluster on AWS, you will need an Access Key ID and a Secret Access Key from AWS.

Supported Production Grade Tools

- conjure-up is an open-source installer for Kubernetes that creates Kubernetes clusters with native AWS integrations on Ubuntu.
- Kubernetes Operations - Production Grade K8s Installation, Upgrades, and Management. Supports running Debian, Ubuntu, CentOS, and RHEL in AWS.
- CoreOS Tectonic includes the open-source Tectonic Installer that creates Kubernetes clusters with Container Linux nodes on AWS.
- CoreOS originated and the Kubernetes Incubator maintains a CLI tool, kube-aws, that creates and manages Kubernetes clusters with Container Linux nodes, using AWS tools: EC2, CloudFormation and Autoscaling.

Getting started with your cluster

Command line administration tool: kubectl

The cluster startup script will leave you with a `kubernetes` directory on your workstation. Alternately, you can download the latest Kubernetes release from this page.

Next, add the appropriate binary folder to your `PATH` to access kubectl:

```
# macOS
export PATH=<path/to/kubernetes-directory>/platforms/darwin/amd64:$PATH

# Linux
```

```
export PATH=<path/to/kubernetes-directory>/platforms/linux/amd64:$PATH
```

An up-to-date documentation page for this tool is available here: kubectl manual

By default, `kubectl` will use the `kubeconfig` file generated during the cluster startup for authenticating against the API. For more information, please read kubeconfig files

Examples

See a simple nginx example to try out your new cluster.

The "Guestbook" application is another popular example to get started with Kubernetes: guestbook example

For more complete applications, please look in the examples directory

Scaling the cluster

Adding and removing nodes through `kubectl` is not supported. You can still scale the amount of nodes manually through adjustments of the 'Desired' and 'Max' properties within the Auto Scaling Group, which was created during the installation.

Tearing down the cluster

Make sure the environment variables you used to provision your cluster are still exported, then call the following script inside the `kubernetes` directory:

```
cluster/kube-down.sh
```

Support Level

IaaS Provider	Config. Mgmt	OS	Networking	Docs	Conforms	Support Level
AWS	kops	Debian	k8s (VPC)	docs		Community (@justinsb
AWS	CoreOS	CoreOS	flannel	docs		Community
AWS	Juju	Ubuntu	flannel, calico, canal	docs	100%	Commercial, Communi

For support level information on all solutions, see the Table of solutions chart.

Further reading

Please see the Kubernetes docs for more details on administering and using a Kubernetes cluster.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on August 21, 2018 at 11:37 PM PST by apply template (#9961) (Page History)

Edit This Page

Running Kubernetes on AWS EC2

This page describes how to install a Kubernetes cluster on AWS.

- Before you begin
- Getting started with your cluster
- Scaling the cluster
- Tearing down the cluster
- Support Level
- Further reading

Before you begin

To create a Kubernetes cluster on AWS, you will need an Access Key ID and a Secret Access Key from AWS.

Supported Production Grade Tools

- conjure-up is an open-source installer for Kubernetes that creates Kubernetes clusters with native AWS integrations on Ubuntu.
- Kubernetes Operations - Production Grade K8s Installation, Upgrades, and Management. Supports running Debian, Ubuntu, CentOS, and RHEL in AWS.
- CoreOS Tectonic includes the open-source Tectonic Installer that creates Kubernetes clusters with Container Linux nodes on AWS.
- CoreOS originated and the Kubernetes Incubator maintains a CLI tool, kube-aws, that creates and manages Kubernetes clusters with Container Linux nodes, using AWS tools: EC2, CloudFormation and Autoscaling.

Getting started with your cluster

Command line administration tool: kubectl

The cluster startup script will leave you with a `kubernetes` directory on your workstation. Alternately, you can download the latest Kubernetes release from this page.

Next, add the appropriate binary folder to your `PATH` to access kubectl:

```
# macOS
export PATH=<path/to/kubernetes-directory>/platforms/darwin/amd64:$PATH

# Linux
export PATH=<path/to/kubernetes-directory>/platforms/linux/amd64:$PATH
```

An up-to-date documentation page for this tool is available here: kubectl manual

By default, `kubectl` will use the `kubeconfig` file generated during the cluster startup for authenticating against the API. For more information, please read kubeconfig files

Examples

See a simple nginx example to try out your new cluster.

The "Guestbook" application is another popular example to get started with Kubernetes: guestbook example

For more complete applications, please look in the examples directory

Scaling the cluster

Adding and removing nodes through `kubectl` is not supported. You can still scale the amount of nodes manually through adjustments of the 'Desired' and 'Max' properties within the Auto Scaling Group, which was created during the installation.

Tearing down the cluster

Make sure the environment variables you used to provision your cluster are still exported, then call the following script inside the `kubernetes` directory:

```
cluster/kube-down.sh
```

Support Level

IaaS Provider	Config. Mgmt	OS	Networking	Docs	Conforms	Support Level
AWS	kops	Debian	k8s (VPC)	docs		Community (@justinsb
AWS	CoreOS	CoreOS	flannel	docs		Community
AWS	Juju	Ubuntu	flannel, calico, canal	docs	100%	Commercial, Communi

For support level information on all solutions, see the Table of solutions chart.

Further reading

Please see the Kubernetes docs for more details on administering and using a Kubernetes cluster.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on August 21, 2018 at 11:37 PM PST by apply template (#9961) (Page History)

Edit This Page

Running Kubernetes on Alibaba Cloud

- – Alibaba Cloud Container Service
  – Custom Deployments

Alibaba Cloud Container Service

The Alibaba Cloud Container Service lets you run and manage Docker applications on a cluster of Alibaba Cloud ECS instances. It supports the popular open source container orchestrators: Docker Swarm and Kubernetes.

To simplify cluster deployment and management, use Kubernetes Support for Alibaba Cloud Container Service. You can get started quickly by following the Kubernetes walk-through, and there are some tutorials for Kubernetes Support on Alibaba Cloud in Chinese.

To use custom binaries or open source Kubernetes, follow the instructions below.

Custom Deployments

The source code for Kubernetes with Alibaba Cloud provider implementation is open source and available on GitHub.

For more information, see “Quick deployment of Kubernetes - VPC environment on Alibaba Cloud” in English and Chinese.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on January 24, 2019 at 2:03 AM PST by update alibaba-cloud.md (#11940) (Page History)

Edit This Page

Running Kubernetes on Azure

- – Azure Kubernetes Service (AKS)
  – Custom Deployments: AKS-Engine
  – CoreOS Tectonic for Azure

Azure Kubernetes Service (AKS)

The Azure Kubernetes Service offers simple deployments for Kubernetes clusters.

For an example of deploying a Kubernetes cluster onto Azure via the Azure Kubernetes Service:

Microsoft Azure Kubernetes Service

Custom Deployments: AKS-Engine

The core of the Azure Kubernetes Service is **open source** and available on GitHub for the community to use and contribute to: **AKS-Engine**. The legacy ACS-Engine codebase has been deprecated in favor of AKS-engine.

AKS-Engine is a good choice if you need to make customizations to the deployment beyond what the Azure Kubernetes Service officially supports. These customizations include deploying into existing virtual networks, utilizing multiple agent pools, and more. Some community contributions to AKS-Engine may even become features of the Azure Kubernetes Service.

The input to AKS-Engine is an apimodel JSON file describing the Kubernetes cluster. It is similar to the Azure Resource Manager (ARM) template syntax used to deploy a cluster directly with the Azure Kubernetes Service. The resulting output is an ARM template that can be checked into source control and used to deploy Kubernetes clusters to Azure.

You can get started by following the **AKS-Engine Kubernetes Tutorial**.

CoreOS Tectonic for Azure

The CoreOS Tectonic Installer for Azure is **open source** and available on GitHub for the community to use and contribute to: **Tectonic Installer**.

Tectonic Installer is a good choice when you need to make cluster customizations as it is built on Hashicorp's Terraform Azure Resource Manager (ARM) provider. This enables users to customize or integrate using familiar Terraform tooling.

You can get started using the Tectonic Installer for Azure Guide.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 28, 2019 at 5:04 AM PST by docs: update acs-engine -> aks-engine (#12804) (Page History)

Edit This Page

Running Kubernetes on CenturyLink Cloud

- – Find Help
  – Clusters of VMs or Physical Servers, your choice.
  – Requirements
  – Script Installation
    * · Script Installation Example: Ubuntu 14 Walkthrough
  – Cluster Creation
    * · Cluster Creation: Script Options
  – Cluster Expansion
    * · Cluster Expansion: Script Options
  – Cluster Deletion
  – Examples
  – Cluster Features and Architecture

  - Optional add-ons
  - Cluster management
    * Accessing the cluster programmatically
    * Accessing the cluster with a browser
  - What Kubernetes features do not work on CenturyLink Cloud
  - Ansible Files
  - Further reading

{: toc}

These scripts handle the creation, deletion and expansion of Kubernetes clusters on CenturyLink Cloud.

You can accomplish all these tasks with a single command. We have made the Ansible playbooks used to perform these tasks available here.

Find Help

If you run into any problems or want help with anything, we are here to help. Reach out to use via any of the following ways:

- Submit a github issue
- Send an email to Kubernetes AT ctl DOT io
- Visit http://info.ctl.io/kubernetes

Clusters of VMs or Physical Servers, your choice.

- We support Kubernetes clusters on both Virtual Machines or Physical Servers. If you want to use physical servers for the worker nodes (minions), simple use the –minion_type=bareMetal flag.
- For more information on physical servers, visit: https://www.ctl.io/bare-metal/
- Physical serves are only available in the VA1 and GB3 data centers.
- VMs are available in all 13 of our public cloud locations

Requirements

The requirements to run this script are:

- A linux administrative host (tested on ubuntu and macOS)
- python 2 (tested on 2.7.11)
  - pip (installed with python as of 2.7.9)
- git
- A CenturyLink Cloud account with rights to create new hosts
- An active VPN connection to the CenturyLink Cloud from your linux host

Script Installation

After you have all the requirements met, please follow these instructions to install this script.

1) Clone this repository and cd into it.

```
git clone https://github.com/CenturyLinkCloud/adm-kubernetes-on-clc
```

2) Install all requirements, including

- Ansible
- CenturyLink Cloud SDK
- Ansible Modules

```
sudo pip install -r ansible/requirements.txt
```

3) Create the credentials file from the template and use it to set your ENV variables

```
cp ansible/credentials.sh.template ansible/credentials.sh
vi ansible/credentials.sh
source ansible/credentials.sh
```

4) Grant your machine access to the CenturyLink Cloud network by using a VM inside the network or configuring a VPN connection to the CenturyLink Cloud network.

Script Installation Example: Ubuntu 14 Walkthrough

If you use an ubuntu 14, for your convenience we have provided a step by step guide to install the requirements and install the script.

```
# system
apt-get update
apt-get install -y git python python-crypto
curl -O https://bootstrap.pypa.io/get-pip.py
python get-pip.py

# installing this repository
mkdir -p ~home/k8s-on-clc
cd ~home/k8s-on-clc
git clone https://github.com/CenturyLinkCloud/adm-kubernetes-on-clc.git
cd adm-kubernetes-on-clc/
pip install -r requirements.txt

# getting started
cd ansible
cp credentials.sh.template credentials.sh; vi credentials.sh
source credentials.sh
```

Cluster Creation

To create a new Kubernetes cluster, simply run the `kube-up.sh` script. A complete list of script options and some examples are listed below.

```
CLC_CLUSTER_NAME=[name of kubernetes cluster]
cd ./adm-kubernetes-on-clc
bash kube-up.sh -c="$CLC_CLUSTER_NAME"
```

It takes about 15 minutes to create the cluster. Once the script completes, it will output some commands that will help you setup kubectl on your machine to point to the new cluster.

When the cluster creation is complete, the configuration files for it are stored locally on your administrative host, in the following directory

```
> CLC_CLUSTER_HOME=$HOME/.clc_kube/$CLC_CLUSTER_NAME/
```

Cluster Creation: Script Options

```
Usage: kube-up.sh [OPTIONS]
Create servers in the CenturyLinkCloud environment and initialize a Kubernetes cluster
Environment variables CLC_V2_API_USERNAME and CLC_V2_API_PASSWD must be set in
order to access the CenturyLinkCloud API

All options (both short and long form) require arguments, and must include "="
between option name and option value.

     -h (--help)                   display this help and exit
     -c= (--clc_cluster_name=)     set the name of the cluster, as used in CLC group names
     -t= (--minion_type=)          standard -> VM (default), bareMetal -> physical]
     -d= (--datacenter=)           VA1 (default)
     -m= (--minion_count=)         number of kubernetes minion nodes
     -mem= (--vm_memory=)          number of GB ram for each minion
     -cpu= (--vm_cpu=)             number of virtual cps for each minion node
     -phyid= (--server_conf_id=)   physical server configuration id, one of
                                      physical_server_20_core_conf_id
                                      physical_server_12_core_conf_id
                                      physical_server_4_core_conf_id (default)
     -etcd_separate_cluster=yes    create a separate cluster of three etcd nodes,
                                   otherwise run etcd on the master node
```

Cluster Expansion

To expand an existing Kubernetes cluster, run the `add-kube-node.sh` script. A complete list of script options and some examples are listed below. This script

must be run from the same host that created the cluster (or a host that has the cluster artifact files stored in `~/.clc_kube/$cluster_name`).

```
cd ./adm-kubernetes-on-clc
bash add-kube-node.sh -c="name_of_kubernetes_cluster" -m=2
```

Cluster Expansion: Script Options

```
Usage: add-kube-node.sh [OPTIONS]
Create servers in the CenturyLinkCloud environment and add to an
existing CLC kubernetes cluster

Environment variables CLC_V2_API_USERNAME and CLC_V2_API_PASSWD must be set in
order to access the CenturyLinkCloud API

     -h (--help)                   display this help and exit
     -c= (--clc_cluster_name=)     set the name of the cluster, as used in CLC group names
     -m= (--minion_count=)         number of kubernetes minion nodes to add
```

Cluster Deletion

There are two ways to delete an existing cluster:

1) Use our python script:

```
python delete_cluster.py --cluster=clc_cluster_name --datacenter=DC1
```

2) Use the CenturyLink Cloud UI. To delete a cluster, log into the CenturyLink Cloud control portal and delete the parent server group that contains the Kubernetes Cluster. We hope to add a scripted option to do this soon.

Examples

Create a cluster with name of k8s_1, 1 master node and 3 worker minions (on physical machines), in VA1

```
bash kube-up.sh --clc_cluster_name=k8s_1 --minion_type=bareMetal --minion_count=3 --datacent
```

Create a cluster with name of k8s_2, an ha etcd cluster on 3 VMs and 6 worker minions (on VMs), in VA1

```
bash kube-up.sh --clc_cluster_name=k8s_2 --minion_type=standard --minion_count=6 --datacente
```

Create a cluster with name of k8s_3, 1 master node, and 10 worker minions (on VMs) with higher mem/cpu, in UC1:

```
bash kube-up.sh --clc_cluster_name=k8s_3 --minion_type=standard --minion_count=10 --datacent
```

Cluster Features and Architecture

We configure the Kubernetes cluster with the following features:

- KubeDNS: DNS resolution and service discovery
- Heapster/InfluxDB: For metric collection. Needed for Grafana and auto-scaling.
- Grafana: Kubernetes/Docker metric dashboard
- KubeUI: Simple web interface to view Kubernetes state
- Kube Dashboard: New web interface to interact with your cluster

We use the following to create the Kubernetes cluster:

- Kubernetes 1.1.7
- Ubuntu 14.04
- Flannel 0.5.4
- Docker 1.9.1-0~trusty
- Etcd 2.2.2

Optional add-ons

- Logging: We offer an integrated centralized logging ELK platform so that all Kubernetes and docker logs get sent to the ELK stack. To install the ELK stack and configure Kubernetes to send logs to it, follow the log aggregation documentation. Note: We don't install this by default as the footprint isn't trivial.

Cluster management

The most widely used tool for managing a Kubernetes cluster is the command-line utility `kubectl`. If you do not already have a copy of this binary on your administrative machine, you may run the script `install_kubectl.sh` which will download it and install it in `/usr/bin/local`.

The script requires that the environment variable `CLC_CLUSTER_NAME` be defined

```
authentication certificates for the particular cluster.  The configuration file is
written to the  ```${CLC_CLUSTER_HOME}/kube``` directory

```shell
export KUBECONFIG=${CLC_CLUSTER_HOME}/kube/config
kubectl version
kubectl cluster-info
```
```

Accessing the cluster programmatically

It's possible to use the locally stored client certificates to access the apiserver. For example, you may want to use any of the Kubernetes API client libraries to program against your Kubernetes cluster in the programming language of your choice.

To demonstrate how to use these locally stored certificates, we provide the following example of using `curl` to communicate to the master apiserver via https:

```
curl \
   --cacert ${CLC_CLUSTER_HOME}/pki/ca.crt  \
   --key ${CLC_CLUSTER_HOME}/pki/kubecfg.key \
   --cert ${CLC_CLUSTER_HOME}/pki/kubecfg.crt  https://${MASTER_IP}:6443
```

But please note, this *does not* work out of the box with the `curl` binary distributed with macOS.

Accessing the cluster with a browser

We install the kubernetes dashboard. When you create a cluster, the script should output URLs for these interfaces like this:

kubernetes-dashboard is running at `https://${MASTER_IP}:6443/api/v1/namespaces/kube-system/servic`

Note on Authentication to the UIs: The cluster is set up to use basic authentication for the user *admin*. Hitting the url at

```
from the apiserver, and then presenting the admin password written to file at:

```> _${CLC_CLUSTER_HOME}/kube/admin_password.txt_```


Configuration files

Various configuration files are written into the home directory *CLC_CLUSTER_HOME* under
```.clc_kube/${CLC_CLUSTER_NAME}``` in several subdirectories. You can use these files
to access the cluster from machines other than where you created the cluster from.

* ```config/```: Ansible variable files containing parameters describing the master and mini
* ```hosts/```: hosts files listing access information for the Ansible playbooks
* ```kube/```: ```kubectl``` configuration files, and the basic-authentication password for
* ```pki/```: public key infrastructure files enabling TLS communication in the cluster
* ```ssh/```: SSH keys for root access to the hosts

```kubectl``` usage examples
```

```
There are a great many features of _kubectl_.  Here are a few examples

List existing nodes, pods, services and more, in all namespaces, or in just one:

```shell
kubectl get nodes
kubectl get --all-namespaces pods
kubectl get --all-namespaces services
kubectl get --namespace=kube-system replicationcontrollers
```

The Kubernetes API server exposes services on web URLs, which are protected by requiring client certificates. If you run a kubectl proxy locally, `kubectl` will provide the necessary certificates and serve locally over http.

```
kubectl proxy -p 8001
```

Then, you can access urls like `http://127.0.0.1:8001/api/v1/namespaces/kube-system/services/kuberne`
without the need for client certificates in your browser.

What Kubernetes features do not work on CenturyLink Cloud

These are the known items that don't work on CenturyLink cloud but do work on other cloud providers:

- At this time, there is no support services of the type LoadBalancer. We are actively working on this and hope to publish the changes sometime around April 2016.
- At this time, there is no support for persistent storage volumes provided by CenturyLink Cloud. However, customers can bring their own persistent storage offering. We ourselves use Gluster.

Ansible Files

If you want more information about our Ansible files, please read this file

Further reading

Please see the Kubernetes docs for more details on administering and using a Kubernetes cluster.
```

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on January 14, 2019 at 3:26 AM PST by Improved openshift, docker, ansible references (#12201) (Page History)

Edit This Page

Running Kubernetes on Google Compute Engine

The example below creates a Kubernetes cluster with 3 worker node Virtual Machines and a master Virtual Machine (i.e. 4 VMs in your cluster). This cluster is set up and controlled from your workstation (or wherever you find convenient).

- Before you begin
- Starting a cluster
- Installing the Kubernetes command line tools on your workstation
- Getting started with your cluster
- Tearing down the cluster
- Customizing
- Troubleshooting
- Support Level
- Further reading

Before you begin

If you want a simplified getting started experience and GUI for managing clusters, please consider trying Google Kubernetes Engine for hosted cluster installation and management.

For an easy way to experiment with the Kubernetes development environment, click the button below to open a Google Cloud Shell with an auto-cloned copy of the Kubernetes source repo.

If you want to use custom binaries or pure open source Kubernetes, please continue with the instructions below.

Prerequisites

1. You need a Google Cloud Platform account with billing enabled. Visit the Google Developers Console for more details.
2. Install `gcloud` as necessary. `gcloud` can be installed as a part of the Google Cloud SDK.
3. Enable the Compute Engine Instance Group Manager API in the Google Cloud developers console.
4. Make sure that gcloud is set to use the Google Cloud Platform project you want. You can check the current project using `gcloud config list project` and change it via `gcloud config set project <project-id>`.
5. Make sure you have credentials for GCloud by running `gcloud auth login`.
6. (Optional) In order to make API calls against GCE, you must also run `gcloud auth application-default login`.
7. Make sure you can start up a GCE VM from the command line. At least make sure you can do the Create an instance part of the GCE Quickstart.
8. Make sure you can SSH into the VM without interactive prompts. See the Log in to the instance part of the GCE Quickstart.

Starting a cluster

You can install a client and start a cluster with either one of these commands (we list both in case only one is installed on your machine):

```
curl -sS https://get.k8s.io | bash
```

or

```
wget -q -O - https://get.k8s.io | bash
```

Once this command completes, you will have a master VM and four worker VMs, running as a Kubernetes cluster.

By default, some containers will already be running on your cluster. Containers like `fluentd` provide logging, while `heapster` provides monitoring services.

The script run by the commands above creates a cluster with the name/prefix "kubernetes". It defines one specific cluster config, so you can't run it more than once.

Alternately, you can download and install the latest Kubernetes release from this page, then run the `<kubernetes>/cluster/kube-up.sh` script to start the cluster:

```
cd kubernetes
cluster/kube-up.sh
```

If you want more than one cluster running in your project, want to use a different name, or want a different number of worker nodes, see the `<kubernetes>/cluster/gce/config-default.sh` file for more fine-grained configuration before you start up your cluster.

If you run into trouble, please see the section on troubleshooting, post to the Kubernetes Forum, or come ask questions on Slack.

The next few steps will show you:

1. How to set up the command line client on your workstation to manage the cluster
2. Examples of how to use the cluster
3. How to delete the cluster
4. How to start clusters with non-default options (like larger clusters)

Installing the Kubernetes command line tools on your workstation

The cluster startup script will leave you with a running cluster and a `kubernetes` directory on your workstation.

The kubectl tool controls the Kubernetes cluster manager. It lets you inspect your cluster resources, create, delete, and update components, and much more. You will use it to look at your new cluster and bring up example apps.

You can use `gcloud` to install the `kubectl` command-line tool on your workstation:

```
gcloud components install kubectl
```

> **Note:** The kubectl version bundled with `gcloud` may be older than the one downloaded by the get.k8s.io install script. See Installing kubectl document to see how you can set up the latest `kubectl` on your workstation.

Getting started with your cluster

Inspect your cluster

Once `kubectl` is in your path, you can use it to look at your cluster. E.g., running:

```
kubectl get --all-namespaces services
```

should show a set of services that look something like this:

```
NAMESPACE     NAME          TYPE            CLUSTER_IP       EXTERNAL_IP       PORT(S)
default       kubernetes    ClusterIP       10.0.0.1         <none>            443/TCP
kube-system   kube-dns      ClusterIP       10.0.0.2         <none>            53/TCP,53/UI
kube-system   kube-ui       ClusterIP       10.0.0.3         <none>            80/TCP
...
```

Similarly, you can take a look at the set of pods that were created during cluster startup. You can do this via the

```
kubectl get --all-namespaces pods
```

command.

You'll see a list of pods that looks something like this (the name specifics will be different):

```
NAMESPACE     NAME                                          READY     STATUS    RESTARTS
kube-system   coredns-5f4fbb68df-mc8z8                      1/1       Running   0
kube-system   fluentd-cloud-logging-kubernetes-minion-63uo   1/1       Running   0
kube-system   fluentd-cloud-logging-kubernetes-minion-c1n9   1/1       Running   0
kube-system   fluentd-cloud-logging-kubernetes-minion-c4og   1/1       Running   0
kube-system   fluentd-cloud-logging-kubernetes-minion-ngua   1/1       Running   0
kube-system   kube-ui-v1-curt1                              1/1       Running   0
kube-system   monitoring-heapster-v5-ex4u3                  1/1       Running   1
kube-system   monitoring-influx-grafana-v1-piled            2/2       Running   0
```

Some of the pods may take a few seconds to start up (during this time they'll show `Pending`), but check that they all show as `Running` after a short period.

Run some examples

Then, see a simple nginx example to try out your new cluster.

For more complete applications, please look in the examples directory. The guestbook example is a good "getting started" walkthrough.

Tearing down the cluster

To remove/delete/teardown the cluster, use the `kube-down.sh` script.

```
cd kubernetes
cluster/kube-down.sh
```

Likewise, the `kube-up.sh` in the same directory will bring it back up. You do not need to rerun the `curl` or `wget` command: everything needed to setup the Kubernetes cluster is now on your workstation.

Customizing

The script above relies on Google Storage to stage the Kubernetes release. It then will start (by default) a single master VM along with 3 worker VMs. You can tweak some of these parameters by editing `kubernetes/cluster/gce/config-default.sh` You can view a transcript of a successful cluster creation here.

Troubleshooting

Project settings

You need to have the Google Cloud Storage API, and the Google Cloud Storage JSON API enabled. It is activated by default for new projects. Otherwise, it can be done in the Google Cloud Console. See the Google Cloud Storage JSON API Overview for more details.

Also ensure that– as listed in the Prerequisites section– you've enabled the `Compute Engine Instance Group Manager API`, and can start up a GCE VM from the command line as in the GCE Quickstart instructions.

Cluster initialization hang

If the Kubernetes startup script hangs waiting for the API to be reachable, you can troubleshoot by SSHing into the master and node VMs and looking at logs such as `/var/log/startupscript.log`.

Once you fix the issue, you should run `kube-down.sh` to cleanup after the partial cluster creation, before running `kube-up.sh` to try again.

SSH

If you're having trouble SSHing into your instances, ensure the GCE firewall isn't blocking port 22 to your VMs. By default, this should work

but if you have edited firewall rules or created a new non-default network, you'll need to expose it: `gcloud compute firewall-rules create default-ssh --network=<network-name> --description "SSH allowed from anywhere" --allow tcp:22`

Additionally, your GCE SSH key must either have no passcode or you need to be using `ssh-agent`.

Networking

The instances must be able to connect to each other using their private IP. The script uses the "default" network which should have a firewall rule called "default-allow-internal" which allows traffic on any port on the private IPs. If this rule is missing from the default network or if you change the network being used in `cluster/config-default.sh` create a new rule with the following field values:

- Source Ranges: `10.0.0.0/8`
- Allowed Protocols and Port: `tcp:1-65535;udp:1-65535;icmp`

Support Level

IaaS Provider	Config. Mgmt	OS	Networking	Docs	Conforms	Support Level
GCE	Saltstack	Debian	GCE	docs		Project

For support level information on all solutions, see the Table of solutions chart.

Further reading

Please see the Kubernetes docs for more details on administering and using a Kubernetes cluster.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on December 17, 2018 at 12:02 PM PST by GCE: cluster/kube-up.sh creates 3 nodes by default (#11694) (Page History)

Edit This Page

Running Kubernetes on Multiple Clouds with IBM Cloud Private

IBM® Cloud Private is a turnkey cloud solution and an on-premises turnkey cloud solution. IBM Cloud Private delivers pure upstream Kubernetes with the typical management components that are required to run real enterprise workloads. These workloads include health management, log management, audit trails, and metering for tracking usage of workloads on the platform.

IBM Cloud Private is available in a community edition and a fully supported enterprise edition. The community edition is available at no charge from Docker Hub. The enterprise edition supports high availability topologies and includes commercial support from IBM for Kubernetes and the IBM Cloud Private management platform. If you want to try IBM Cloud Private, you can use either the hosted trial, the tutorial, or the self-guided demo. You can also try the free community edition. For details, see Get started with IBM Cloud Private.

For more information, explore the following resources:

- IBM Cloud Private
- Reference architecture for IBM Cloud Private
- IBM Cloud Private documentation

IBM Cloud Private and Terraform

The following modules are available where you can deploy IBM Cloud Private by using Terraform:

- AWS: Deploy IBM Cloud Private to AWS

- Azure: Deploy IBM Cloud Private to Azure
- IBM Cloud: Deploy IBM Cloud Private cluster to IBM Cloud
- OpenStack: Deploy IBM Cloud Private to OpenStack
- Terraform module: Deploy IBM Cloud Private on any supported infrastructure vendor
- VMware: Deploy IBM Cloud Private to VMware

IBM Cloud Private on AWS

You can deploy an IBM Cloud Private cluster on Amazon Web Services (AWS) by using either AWS CloudFormation or Terraform.

IBM Cloud Private has a Quick Start that automatically deploys IBM Cloud Private into a new virtual private cloud (VPC) on the AWS Cloud. A regular deployment takes about 60 minutes, and a high availability (HA) deployment takes about 75 minutes to complete. The Quick Start includes AWS CloudFormation templates and a deployment guide.

This Quick Start is for users who want to explore application modernization and want to accelerate meeting their digital transformation goals, by using IBM Cloud Private and IBM tooling. The Quick Start helps users rapidly deploy a high availability (HA), production-grade, IBM Cloud Private reference architecture on AWS. For all of the details and the deployment guide, see the IBM Cloud Private on AWS Quick Start.

IBM Cloud Private can also run on the AWS cloud platform by using Terraform. To deploy IBM Cloud Private in an AWS EC2 environment, see Installing IBM Cloud Private on AWS.

IBM Cloud Private on Azure

You can enable Microsoft Azure as a cloud provider for IBM Cloud Private deployment and take advantage of all the IBM Cloud Private features on the Azure public cloud. For more information, see IBM Cloud Private on Azure.

IBM Cloud Private on Red Hat OpenShift

You can deploy IBM certified software containers that are running on IBM Cloud Private onto Red Hat OpenShift.

Integration capabilities:

- Supports Linux® 64-bit platform in offline-only installation mode
- Single-master configuration
- Integrated IBM Cloud Private cluster management console and catalog

- Integrated core platform services, such as monitoring, metering, and logging
- IBM Cloud Private uses the OpenShift image registry

For more information see, IBM Cloud Private on OpenShift.

IBM Cloud Private on VirtualBox

To install IBM Cloud Private to a VirtualBox environment, see Installing IBM Cloud Private on VirtualBox.

IBM Cloud Private on VMware

You can install IBM Cloud Private on VMware with either Ubuntu or RHEL images. For details, see the following projects:

- Installing IBM Cloud Private with Ubuntu
- Installing IBM Cloud Private with Red Hat Enterprise

The IBM Cloud Private Hosted service automatically deploys IBM Cloud Private Hosted on your VMware vCenter Server instances. This service brings the power of microservices and containers to your VMware environment on IBM Cloud. With this service, you can extend the same familiar VMware and IBM Cloud Private operational model and tools from on-premises into the IBM Cloud.

For more information, see IBM Cloud Private Hosted service.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 24, 2019 at 8:28 AM PST by Revising bluemix.net links to cloud.ibm.com (#12792) (Page History)

Edit This Page

Running Kubernetes on Multiple Clouds with Stackpoint.io

StackPointCloud is the universal control plane for Kubernetes Anywhere. StackPointCloud allows you to deploy and manage a Kubernetes cluster to the cloud provider of your choice in 3 steps using a web-based interface.

- AWS
- GCE
- Google Kubernetes Engine
- DigitalOcean
- Microsoft Azure
- Packet

AWS

To create a Kubernetes cluster on AWS, you will need an Access Key ID and a Secret Access Key from AWS.

1. Choose a Provider

   a. Log in to stackpoint.io with a GitHub, Google, or Twitter account.

   b. Click **+ADD A CLUSTER NOW**.

   c. Click to select Amazon Web Services (AWS).

2. Configure Your Provider

   a. Add your Access Key ID and a Secret Access Key from AWS. Select your default StackPointCloud SSH keypair, or click **ADD SSH KEY** to add a new keypair.

   b. Click **SUBMIT** to submit the authorization information.

3. Configure Your Cluster

   Choose any extra options you may want to include with your cluster, then click **SUBMIT** to create the cluster.

4. Run the Cluster

   You can monitor the status of your cluster and suspend or delete it from your stackpoint.io dashboard.

   For information on using and managing a Kubernetes cluster on AWS, consult the Kubernetes documentation.

GCE

To create a Kubernetes cluster on GCE, you will need the Service Account JSON Data from Google.

1. Choose a Provider

   a. Log in to stackpoint.io with a GitHub, Google, or Twitter account.

   b. Click **+ADD A CLUSTER NOW**.

   c. Click to select Google Compute Engine (GCE).

2. Configure Your Provider

   a. Add your Service Account JSON Data from Google. Select your default StackPointCloud SSH keypair, or click **ADD SSH KEY** to add a new keypair.

   b. Click **SUBMIT** to submit the authorization information.

3. Configure Your Cluster

   Choose any extra options you may want to include with your cluster, then click **SUBMIT** to create the cluster.

4. Run the Cluster

   You can monitor the status of your cluster and suspend or delete it from your stackpoint.io dashboard.

   For information on using and managing a Kubernetes cluster on GCE, consult the Kubernetes documentation.

Google Kubernetes Engine

To create a Kubernetes cluster on Google Kubernetes Engine, you will need the Service Account JSON Data from Google.

1. Choose a Provider

   a. Log in to stackpoint.io with a GitHub, Google, or Twitter account.

   b. Click **+ADD A CLUSTER NOW**.

   c. Click to select Google Kubernetes Engine.

2. Configure Your Provider

   a. Add your Service Account JSON Data from Google. Select your default StackPointCloud SSH keypair, or click **ADD SSH KEY** to add a new keypair.

   b. Click **SUBMIT** to submit the authorization information.

3. Configure Your Cluster

   Choose any extra options you may want to include with your cluster, then click **SUBMIT** to create the cluster.

4. Run the Cluster

   You can monitor the status of your cluster and suspend or delete it from your stackpoint.io dashboard.

   For information on using and managing a Kubernetes cluster on Google Kubernetes Engine, consult the official documentation.

DigitalOcean

To create a Kubernetes cluster on DigitalOcean, you will need a DigitalOcean API Token.

1. Choose a Provider

   a. Log in to stackpoint.io with a GitHub, Google, or Twitter account.

   b. Click **+ADD A CLUSTER NOW**.

   c. Click to select DigitalOcean.

2. Configure Your Provider

   a. Add your DigitalOcean API Token. Select your default StackPoint-Cloud SSH keypair, or click **ADD SSH KEY** to add a new keypair.

   b. Click **SUBMIT** to submit the authorization information.

3. Configure Your Cluster

   Choose any extra options you may want to include with your cluster, then click **SUBMIT** to create the cluster.

4. Run the Cluster

   You can monitor the status of your cluster and suspend or delete it from your stackpoint.io dashboard.

   For information on using and managing a Kubernetes cluster on DigitalOcean, consult the official documentation.

Microsoft Azure

To create a Kubernetes cluster on Microsoft Azure, you will need an Azure Subscription ID, Username/Email, and Password.

1. Choose a Provider

   a. Log in to stackpoint.io with a GitHub, Google, or Twitter account.

   b. Click **+ADD A CLUSTER NOW**.

   c. Click to select Microsoft Azure.

2. Configure Your Provider

   a. Add your Azure Subscription ID, Username/Email, and Password. Select your default StackPointCloud SSH keypair, or click **ADD SSH KEY** to add a new keypair.

   b. Click **SUBMIT** to submit the authorization information.

3. Configure Your Cluster

   Choose any extra options you may want to include with your cluster, then click **SUBMIT** to create the cluster.

4. Run the Cluster

   You can monitor the status of your cluster and suspend or delete it from your stackpoint.io dashboard.

   For information on using and managing a Kubernetes cluster on Azure, consult the Kubernetes documentation.

Packet

To create a Kubernetes cluster on Packet, you will need a Packet API Key.

1. Choose a Provider

   a. Log in to stackpoint.io with a GitHub, Google, or Twitter account.

   b. Click **+ADD A CLUSTER NOW**.

   c. Click to select Packet.

2. Configure Your Provider

   a. Add your Packet API Key. Select your default StackPointCloud SSH keypair, or click **ADD SSH KEY** to add a new keypair.

   b. Click **SUBMIT** to submit the authorization information.

3. Configure Your Cluster

   Choose any extra options you may want to include with your cluster, then click **SUBMIT** to create the cluster.

4. Run the Cluster

   You can monitor the status of your cluster and suspend or delete it from your stackpoint.io dashboard.

   For information on using and managing a Kubernetes cluster on Packet, consult the official documentation.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on July 18, 2018 at 4:26 PM PST by Change formatting of Stackpoint turnkey doc (#9493) (Page History)

Edit This Page

Installing Kubernetes On-premises/Cloud Providers with Kubespray

This quickstart helps to install a Kubernetes cluster hosted on GCE, Azure, OpenStack, AWS, vSphere, Oracle Cloud Infrastructure (Experimental) or Baremetal with Kubespray.

Kubespray is a composition of Ansible playbooks, inventory, provisioning tools, and domain knowledge for generic OS/Kubernetes clusters configuration management tasks. Kubespray provides:

- a highly available cluster
- composable attributes
- support for most popular Linux distributions
  - Container Linux by CoreOS
  - Debian Jessie, Stretch, Wheezy
  - Ubuntu 16.04, 18.04
  - CentOS/RHEL 7

    - Fedora/CentOS Atomic
    - openSUSE Leap 42.3/Tumbleweed
- continuous integration tests

To choose a tool which best fits your use case, read this comparison to kubeadm and kops.

- Creating a cluster
- Cluster operations
- Cleanup
- Feedback
- What's next

Creating a cluster

(1/5) Meet the underlay requirements

Provision servers with the following requirements:

- **Ansible v2.5 (or newer) and python-netaddr is installed on the machine that will run Ansible commands**
- **Jinja 2.9 (or newer) is required to run the Ansible Playbooks**
- The target servers must have **access to the Internet** in order to pull docker images
- The target servers are configured to allow **IPv4 forwarding**
- **Your ssh key must be copied** to all the servers part of your inventory
- The **firewalls are not managed**, you'll need to implement your own rules the way you used to. in order to avoid any issue during deployment you should disable your firewall
- If kubespray is ran from non-root user account, correct privilege escalation method should be configured in the target servers. Then the `ansible_become` flag or command parameters `--become` or `-b` should be specified

Kubespray provides the following utilities to help provision your environment:

- Terraform scripts for the following cloud providers:
    - AWS
    - OpenStack

(2/5) Compose an inventory file

After you provision your servers, create an inventory file for Ansible. You can do this manually or via a dynamic inventory script. For more information, see "Building your own inventory".

(3/5) Plan your cluster deployment

Kubespray provides the ability to customize many aspects of the deployment:

- Choice deployment mode: kubeadm or non-kubeadm
- CNI (networking) plugins
- DNS configuration
- Choice of control plane: native/binary or containerized with docker or rkt
- Component versions
- Calico route reflectors
- Component runtime options
  - docker
  - rkt
  - cri-o
- Certificate generation methods (**Vault being discontinued**)

Kubespray customizations can be made to a variable file. If you are just getting started with Kubespray, consider using the Kubespray defaults to deploy your cluster and explore Kubernetes.

(4/5) Deploy a Cluster

Next, deploy your cluster:

Cluster deployment using ansible-playbook.

```
ansible-playbook -i your/inventory/hosts.ini cluster.yml -b -v \
  --private-key=~/.ssh/private_key
```

Large deployments (100+ nodes) may require specific adjustments for best results.

(5/5) Verify the deployment

Kubespray provides a way to verify inter-pod connectivity and DNS resolve with Netchecker. Netchecker ensures the netchecker-agents pods can resolve DNS requests and ping each over within the default namespace. Those pods mimic similar behavior of the rest of the workloads and serve as cluster health indicators.

Cluster operations

Kubespray provides additional playbooks to manage your cluster: *scale* and *upgrade*.

Scale your cluster

You can add worker nodes from your cluster by running the scale playbook. For more information, see "Adding nodes". You can remove worker nodes from your cluster by running the remove-node playbook. For more information, see "Remove nodes".

Upgrade your cluster

You can upgrade your cluster by running the upgrade-cluster playbook. For more information, see "Upgrades".

Cleanup

You can reset your nodes and wipe out all components installed with Kubespray via the reset playbook.

> **Caution:** When running the reset playbook, be sure not to accidentally target your production cluster!

Feedback

- Slack Channel: #kubespray
- GitHub Issues

What's next

Check out planned work on Kubespray's roadmap.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on November 16, 2018 at 5:05 PM PST by Update kubespray.md (#10992) (Page History)

Edit This Page

Installing Kubernetes On-premises/Cloud Providers with Kubespray

This quickstart helps to install a Kubernetes cluster hosted on GCE, Azure, OpenStack, AWS, vSphere, Oracle Cloud Infrastructure (Experimental) or Baremetal with Kubespray.

Kubespray is a composition of Ansible playbooks, inventory, provisioning tools, and domain knowledge for generic OS/Kubernetes clusters configuration management tasks. Kubespray provides:

- a highly available cluster
- composable attributes
- support for most popular Linux distributions
  - Container Linux by CoreOS
  - Debian Jessie, Stretch, Wheezy
  - Ubuntu 16.04, 18.04
  - CentOS/RHEL 7
  - Fedora/CentOS Atomic
  - openSUSE Leap 42.3/Tumbleweed
- continuous integration tests

To choose a tool which best fits your use case, read this comparison to kubeadm and kops.

- Creating a cluster
- Cluster operations
- Cleanup
- Feedback
- What's next

Creating a cluster

(1/5) Meet the underlay requirements

Provision servers with the following requirements:

- **Ansible v2.5 (or newer) and python-netaddr is installed on the machine that will run Ansible commands**
- **Jinja 2.9 (or newer) is required to run the Ansible Playbooks**
- The target servers must have **access to the Internet** in order to pull docker images
- The target servers are configured to allow **IPv4 forwarding**
- **Your ssh key must be copied** to all the servers part of your inventory
- The **firewalls are not managed**, you'll need to implement your own rules the way you used to. in order to avoid any issue during deployment you should disable your firewall
- If kubespray is ran from non-root user account, correct privilege escalation method should be configured in the target servers. Then the `ansible_become` flag or command parameters `--become` or `-b` should be specified

Kubespray provides the following utilities to help provision your environment:

- Terraform scripts for the following cloud providers:
  - AWS
  - OpenStack

(2/5) Compose an inventory file

After you provision your servers, create an inventory file for Ansible. You can do this manually or via a dynamic inventory script. For more information, see "Building your own inventory".

(3/5) Plan your cluster deployment

Kubespray provides the ability to customize many aspects of the deployment:

- Choice deployment mode: kubeadm or non-kubeadm
- CNI (networking) plugins
- DNS configuration
- Choice of control plane: native/binary or containerized with docker or rkt
- Component versions
- Calico route reflectors
- Component runtime options
  - docker
  - rkt
  - cri-o
- Certificate generation methods (**Vault being discontinued**)

Kubespray customizations can be made to a variable file. If you are just getting started with Kubespray, consider using the Kubespray defaults to deploy your cluster and explore Kubernetes.

(4/5) Deploy a Cluster

Next, deploy your cluster:

Cluster deployment using ansible-playbook.

```
ansible-playbook -i your/inventory/hosts.ini cluster.yml -b -v \
  --private-key=~/.ssh/private_key
```

Large deployments (100+ nodes) may require specific adjustments for best results.

(5/5) Verify the deployment

Kubespray provides a way to verify inter-pod connectivity and DNS resolve with Netchecker. Netchecker ensures the netchecker-agents pods can resolve DNS requests and ping each over within the default namespace. Those pods mimic similar behavior of the rest of the workloads and serve as cluster health indicators.

Cluster operations

Kubespray provides additional playbooks to manage your cluster: *scale* and *upgrade*.

Scale your cluster

You can add worker nodes from your cluster by running the scale playbook. For more information, see "Adding nodes". You can remove worker nodes from your cluster by running the remove-node playbook. For more information, see "Remove nodes".

Upgrade your cluster

You can upgrade your cluster by running the upgrade-cluster playbook. For more information, see "Upgrades".

Cleanup

You can reset your nodes and wipe out all components installed with Kubespray via the reset playbook.

> **Caution:** When running the reset playbook, be sure not to accidentally target your production cluster!

Feedback

- Slack Channel: #kubespray
- GitHub Issues

What's next

Check out planned work on Kubespray's roadmap.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on November 16, 2018 at 5:05 PM PST by Update kubespray.md (#10992) (Page History)

Edit This Page

Installing Kubernetes on AWS with kops

This quickstart shows you how to easily install a Kubernetes cluster on AWS. It uses a tool called `kops`.

kops is an opinionated provisioning system:

- Fully automated installation
- Uses DNS to identify clusters
- Self-healing: everything runs in Auto-Scaling Groups
- Multiple OS support (Debian, Ubuntu 16.04 supported, CentOS & RHEL, Amazon Linux and CoreOS) - see the images.md
- High-Availability support - see the high_availability.md
- Can directly provision, or generate terraform manifests - see the terraform.md

If your opinions differ from these you may prefer to build your own cluster using kubeadm as a building block. kops builds on the kubeadm work.

- Creating a cluster
- Cleanup
- Feedback
- What's next

Creating a cluster

(1/5) Install kops

Requirements

You must have kubectl installed in order for kops to work.

Installation

Download kops from the releases page (it is also easy to build from source):

On macOS:

```
curl -OL https://github.com/kubernetes/kops/releases/download/1.10.0/kops-darwin-amd64
chmod +x kops-darwin-amd64
mv kops-darwin-amd64 /usr/local/bin/kops
# you can also install using Homebrew
brew update && brew install kops
```

On Linux:

```
wget https://github.com/kubernetes/kops/releases/download/1.10.0/kops-linux-amd64
chmod +x kops-linux-amd64
mv kops-linux-amd64 /usr/local/bin/kops
```

(2/5) Create a route53 domain for your cluster

kops uses DNS for discovery, both inside the cluster and so that you can reach the kubernetes API server from clients.

kops has a strong opinion on the cluster name: it should be a valid DNS name. By doing so you will no longer get your clusters confused, you can share clusters with your colleagues unambiguously, and you can reach them without relying on remembering an IP address.

You can, and probably should, use subdomains to divide your clusters. As our example we will use `useast1.dev.example.com`. The API server endpoint will then be `api.useast1.dev.example.com`.

A Route53 hosted zone can serve subdomains. Your hosted zone could be `useast1.dev.example.com`, but also `dev.example.com` or even `example.com`. kops works with any of these, so typically you choose for organization reasons (e.g. you are allowed to create records under `dev.example.com`, but not under `example.com`).

Let's assume you're using `dev.example.com` as your hosted zone. You create that hosted zone using the normal process, or with a command such as `aws route53 create-hosted-zone --name dev.example.com --caller-reference 1`.

You must then set up your NS records in the parent domain, so that records in the domain will resolve. Here, you would create NS records in `example.com` for `dev`. If it is a root domain name you would configure the NS records at your domain registrar (e.g. `example.com` would need to be configured where you bought `example.com`).

This step is easy to mess up (it is the #1 cause of problems!) You can double-check that your cluster is configured correctly if you have the dig tool by running:

```
dig NS dev.example.com
```

You should see the 4 NS records that Route53 assigned your hosted zone.

(3/5) Create an S3 bucket to store your clusters state

kops lets you manage your clusters even after installation. To do this, it must keep track of the clusters that you have created, along with their configuration, the keys they are using etc. This information is stored in an S3 bucket. S3 permissions are used to control access to the bucket.

Multiple clusters can use the same S3 bucket, and you can share an S3 bucket between your colleagues that administer the same clusters - this is much easier than passing around kubecfg files. But anyone with access to the S3 bucket will have administrative access to all your clusters, so you don't want to share it beyond the operations team.

So typically you have one S3 bucket for each ops team (and often the name will correspond to the name of the hosted zone above!)

In our example, we chose `dev.example.com` as our hosted zone, so let's pick `clusters.dev.example.com` as the S3 bucket name.

- Export `AWS_PROFILE` (if you need to select a profile for the AWS CLI to work)
- Create the S3 bucket using `aws s3 mb s3://clusters.dev.example.com`
- You can `export KOPS_STATE_STORE=s3://clusters.dev.example.com` and then kops will use this location by default. We suggest putting this

in your bash profile or similar.

(4/5) Build your cluster configuration

Run "kops create cluster" to create your cluster configuration:

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

kops will create the configuration for your cluster. Note that it *only* creates the configuration, it does not actually create the cloud resources - you'll do that in the next step with a `kops update cluster`. This give you an opportunity to review the configuration or change it.

It prints commands you can use to explore further:

- List your clusters with: `kops get cluster`
- Edit this cluster with: `kops edit cluster useast1.dev.example.com`
- Edit your node instance group: `kops edit ig --name=useast1.dev.example.com nodes`
- Edit your master instance group: `kops edit ig --name=useast1.dev.example.com master-us-east-1c`

If this is your first time using kops, do spend a few minutes to try those out! An instance group is a set of instances, which will be registered as kubernetes nodes. On AWS this is implemented via auto-scaling-groups. You can have several instance groups, for example if you wanted nodes that are a mix of spot and on-demand instances, or GPU and non-GPU instances.

(5/5) Create the cluster in AWS

Run "kops update cluster" to create your cluster in AWS:

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

That takes a few seconds to run, but then your cluster will likely take a few minutes to actually be ready. `kops update cluster` will be the tool you'll use whenever you change the configuration of your cluster; it applies the changes you have made to the configuration to your cluster - reconfiguring AWS or kubernetes as needed.

For example, after you `kops edit ig nodes`, then `kops update cluster --yes` to apply your configuration, and sometimes you will also have to `kops rolling-update cluster` to roll out the configuration immediately.

Without `--yes`, `kops update cluster` will show you a preview of what it is going to do. This is handy for production clusters!

Explore other add-ons

See the list of add-ons to explore other add-ons, including tools for logging, monitoring, network policy, visualization & control of your Kubernetes cluster.

Cleanup

- To delete your cluster: `kops delete cluster useast1.dev.example.com --yes`

Feedback

- Slack Channel: #kops-users
- GitHub Issues

What's next

- Learn more about Kubernetes concepts and `kubectl`.
- Learn about `kops` advanced usage
- See the `kops` docs section for tutorials, best practices and advanced configuration options.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on August 21, 2018 at 11:44 PM PST by apply template (#9964) (Page History)

Edit This Page

Cloudstack

CloudStack is a software to build public and private clouds based on hardware virtualization principles (traditional IaaS). To deploy Kubernetes on CloudStack there are several possibilities depending on the Cloud being used and what images are made available. CloudStack also has a vagrant plugin available, hence Vagrant could be used to deploy Kubernetes either using the existing shell provisioner or using new Salt based recipes.

CoreOS templates for CloudStack are built nightly. CloudStack operators need to register this template in their cloud before proceeding with these Kubernetes deployment instructions.

This guide uses a single Ansible playbook, which is completely automated and can deploy Kubernetes on a CloudStack based Cloud using CoreOS images. The playbook, creates an ssh key pair, creates a security group and associated rules and finally starts coreOS instances configured via cloud-init.

- Prerequisites
- Support Level

Prerequisites

```
sudo apt-get install -y python-pip libssl-dev
sudo pip install cs
sudo pip install sshpubkeys
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

On CloudStack server you also have to install libselinux-python :

```
yum install libselinux-python
```

cs is a python module for the CloudStack API.

Set your CloudStack endpoint, API keys and HTTP method used.

You can define them as environment variables: `CLOUDSTACK_ENDPOINT`, `CLOUDSTACK_KEY`, `CLOUDSTACK_SECRET` and `CLOUDSTACK_METHOD`.

Or create a `~/.cloudstack.ini` file:

```
[cloudstack]
endpoint = <your cloudstack api endpoint>
key = <your api access key>
secret = <your api secret key>
method = post
```

We need to use the http POST method to pass the *large* userdata to the coreOS instances.

Clone the playbook

```
git clone https://github.com/apachecloudstack/k8s
cd kubernetes-cloudstack
```

Create a Kubernetes cluster

You simply need to run the playbook.

```
ansible-playbook k8s.yml
```

Some variables can be edited in the `k8s.yml` file.

```
vars:
  ssh_key: k8s
  k8s_num_nodes: 2
  k8s_security_group_name: k8s
  k8s_node_prefix: k8s2
  k8s_template: <templatename>
  k8s_instance_type: <serviceofferingname>
```

This will start a Kubernetes master node and a number of compute nodes (by default 2). The `instance_type` and `template` are specific, edit them to specify your CloudStack cloud specific template and instance type (i.e. service offering).

Check the tasks and templates in `roles/k8s` if you want to modify anything.

Once the playbook as finished, it will print out the IP of the Kubernetes master:

```
TASK: [k8s | debug msg='k8s master IP is {{ k8s_master.default_ip }}'] ********
```

SSH to it using the key that was created and using the *core* user.

```
ssh -i ~/.ssh/id_rsa_k8s core@<master IP>
```

And you can list the machines in your cluster:

```
fleetctl list-machines
```

```
MACHINE         IP              METADATA
a017c422...     <node #1 IP>    role=node
ad13bf84...     <master IP>     role=master
e9af8293...     <node #2 IP>    role=node
```

Support Level

IaaS Provider	Config. Mgmt	OS	Networking	Docs	Conforms	Support Level
CloudStack	Ansible	CoreOS	flannel	docs		Community (@Guiques)

For support level information on all solutions, see the Table of solutions chart.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on July 19, 2018 at 6:04 AM PST by apply templates/concept and fix code snippets (#9540) (Page History)

Edit This Page

Cloudstack

CloudStack is a software to build public and private clouds based on hardware virtualization principles (traditional IaaS). To deploy Kubernetes on CloudStack there are several possibilities depending on the Cloud being used and what images are made available. CloudStack also has a vagrant plugin available, hence Vagrant could be used to deploy Kubernetes either using the existing shell provisioner or using new Salt based recipes.

CoreOS templates for CloudStack are built nightly. CloudStack operators need to register this template in their cloud before proceeding with these Kubernetes deployment instructions.

This guide uses a single Ansible playbook, which is completely automated and can deploy Kubernetes on a CloudStack based Cloud using CoreOS images. The playbook, creates an ssh key pair, creates a security group and associated rules and finally starts coreOS instances configured via cloud-init.

- Prerequisites

- Support Level

Prerequisites

```
sudo apt-get install -y python-pip libssl-dev
sudo pip install cs
sudo pip install sshpubkeys
sudo apt-get install software-properties-common
sudo apt-add-repository ppa:ansible/ansible
sudo apt-get update
sudo apt-get install ansible
```

On CloudStack server you also have to install libselinux-python :

```
yum install libselinux-python
```

cs is a python module for the CloudStack API.

Set your CloudStack endpoint, API keys and HTTP method used.

You can define them as environment variables: `CLOUDSTACK_ENDPOINT`, `CLOUDSTACK_KEY`, `CLOUDSTACK_SECRET` and `CLOUDSTACK_METHOD`.

Or create a `~/.cloudstack.ini` file:

```
[cloudstack]
endpoint = <your cloudstack api endpoint>
key = <your api access key>
secret = <your api secret key>
method = post
```

We need to use the http POST method to pass the *large* userdata to the coreOS instances.

Clone the playbook

```
git clone https://github.com/apachecloudstack/k8s
cd kubernetes-cloudstack
```

Create a Kubernetes cluster

You simply need to run the playbook.

```
ansible-playbook k8s.yml
```

Some variables can be edited in the `k8s.yml` file.

```
vars:
  ssh_key: k8s
  k8s_num_nodes: 2
  k8s_security_group_name: k8s
  k8s_node_prefix: k8s2
  k8s_template: <templatename>
  k8s_instance_type: <serviceofferingname>
```

This will start a Kubernetes master node and a number of compute nodes (by default 2). The `instance_type` and `template` are specific, edit them to specify your CloudStack cloud specific template and instance type (i.e. service offering).

Check the tasks and templates in `roles/k8s` if you want to modify anything.

Once the playbook as finished, it will print out the IP of the Kubernetes master:

```
TASK: [k8s | debug msg='k8s master IP is {{ k8s_master.default_ip }}'] ********
```

SSH to it using the key that was created and using the *core* user.

```
ssh -i ~/.ssh/id_rsa_k8s core@<master IP>
```

And you can list the machines in your cluster:

```
fleetctl list-machines
```

```
MACHINE         IP              METADATA
a017c422...     <node #1 IP>    role=node
ad13bf84...     <master IP>     role=master
e9af8293...     <node #2 IP>    role=node
```

Support Level

IaaS Provider	Config. Mgmt	OS	Networking	Docs	Conforms	Support Level
CloudStack	Ansible	CoreOS	flannel	docs		Community (@Guiques)

For support level information on all solutions, see the Table of solutions chart.

Feedback

Was this page helpful?

Yes

No

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

if you want to report a problem or suggest an improvement.

Create an Issue Edit This Page

Page last modified on July 19, 2018 at 6:04 AM PST by apply templates/concept and fix code snippets (#9540) (Page History)

Edit This Page

Kubernetes on DC/OS

Mesosphere provides an easy option to provision Kubernetes onto DC/OS, offering:

- Pure upstream Kubernetes
- Single-click cluster provisioning
- Highly available and secure by default
- Kubernetes running alongside fast-data platforms (e.g. Akka, Cassandra, Kafka, Spark)
- Official Mesosphere Guide

Official Mesosphere Guide

The canonical source of getting started on DC/OS is located in the quickstart repo.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on July 19, 2018 at 7:25 AM PST by apply templates/concept (#9539) (Page History)

Edit This Page

oVirt

oVirt is a virtual datacenter manager that delivers powerful management of multiple virtual machines on multiple hosts. Using KVM and libvirt, oVirt can be installed on Fedora, CentOS, or Red Hat Enterprise Linux hosts to set up and manage your virtual data center.

- oVirt Cloud Provider Deployment
- Using the oVirt Cloud Provider
- oVirt Cloud Provider Screencast
- Support Level

oVirt Cloud Provider Deployment

The oVirt cloud provider allows to easily discover and automatically add new VM instances as nodes to your Kubernetes cluster. At the moment there are no community-supported or pre-loaded VM images including Kubernetes but it is possible to import or install Project Atomic (or Fedora) in a VM to generate a template. Any other distribution that includes Kubernetes may work as well.

It is mandatory to install the ovirt-guest-agent in the guests for the VM ip address and hostname to be reported to ovirt-engine and ultimately to Kubernetes.

Once the Kubernetes template is available it is possible to start instantiating VMs that can be discovered by the cloud provider.

Using the oVirt Cloud Provider

The oVirt Cloud Provider requires access to the oVirt REST-API to gather the proper information, the required credential should be specified in the `ovirt-cloud.conf` file:

```
[connection]
uri = https://localhost:8443/ovirt-engine/api
username = admin@internal
password = admin
```

In the same file it is possible to specify (using the `filters` section) what search query to use to identify the VMs to be reported to Kubernetes:

```
[filters]
# Search query used to find nodes
vms = tag=kubernetes
```

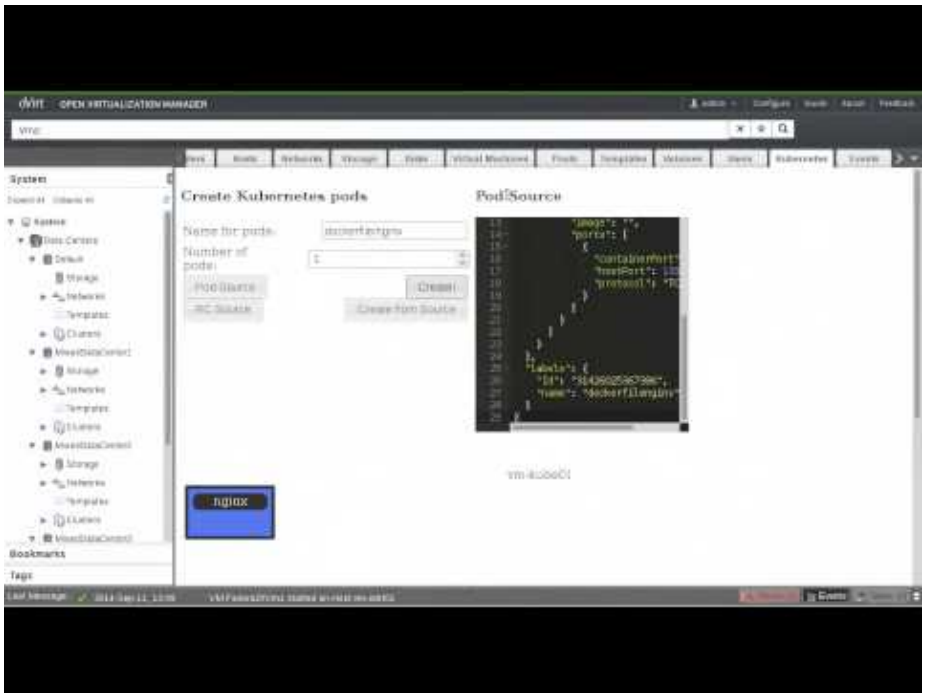
In the above example all the VMs tagged with the `kubernetes` label will be reported as nodes to Kubernetes.

The `ovirt-cloud.conf` file then must be specified in kube-controller-manager:

```
kube-controller-manager ... --cloud-provider=ovirt --cloud-config=/path/to/ovirt-cloud.conf
```

oVirt Cloud Provider Screencast

This short screencast demonstrates how the oVirt Cloud Provider can be used to dynamically add VMs to your Kubernetes cluster.



Support Level

IaaS Provider	Config. Mgmt	OS	Networking	Docs	Conforms	Support Level

IaaS Provider	Config. Mgmt	OS	Networking	Docs	Conforms	Support Level
oVirt				docs		Community (@simon3z)

For support level information on all solutions, see the Table of solutions chart.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on August 08, 2018 at 4:22 PM PST by Fix netlify "mixed content detected" warning (#9683) (Page History)

Edit This Page

Kubernetes Version and Version Skew Support Policy

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

- Supported versions
- Supported version skew
- Supported component upgrade order

Supported versions

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

The Kubernetes project maintains release branches for the most recent three minor releases.

Applicable fixes, including security fixes, may be backported to those three release branches, depending on severity and feasibility. Patch releases are cut from those branches at a regular cadence, or as needed. This decision is owned by the patch release manager. The patch release manager is a member of the release team for each release.

Minor releases occur approximately every 3 months, so each minor release branch is maintained for approximately 9 months.

Supported version skew

kube-apiserver

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

Example:

- newest `kube-apiserver` is at **1.13**
- other `kube-apiserver` instances are supported at **1.13** and **1.12**

kubelet

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

Example:

- `kube-apiserver` is at **1.13**
- `kubelet` is supported at **1.13**, **1.12**, and **1.11**

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

Example:

- `kube-apiserver` instances are at **1.13** and **1.12**
- `kubelet` is supported at **1.12**, and **1.11** (**1.13** is not supported because that would be newer than the `kube-apiserver` instance at version **1.12**)

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

`kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` must not be newer than the `kube-apiserver` instances they communicate with. They are expected to match the `kube-apiserver` minor version, but may be up to one minor version older (to allow live upgrades).

Example:

- `kube-apiserver` is at **1.13**
- `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` are supported at **1.13** and **1.12**

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

Example:

- `kube-apiserver` instances are at **1.13** and **1.12**
- `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` communicate with a load balancer that can route to any `kube-apiserver` instance
- `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` are supported at **1.12** (**1.13** is not supported because that would be newer than the `kube-apiserver` instance at version **1.12**)

kubectl

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

Example:

- `kube-apiserver` is at **1.13**
- `kubectl` is supported at **1.14**, **1.13**, and **1.12**

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

Example:

- `kube-apiserver` instances are at **1.13** and **1.12**
- `kubectl` is supported at **1.13** and **1.12** (other versions would be more than one minor version skewed from one of the `kube-apiserver` components)

Supported component upgrade order

The supported version skew between components has implications on the order in which components must be upgraded. This section describes the order in which components must be upgraded to transition an existing cluster from version **1.n** to version **1.(n+1)**.

kube-apiserver

Pre-requisites:

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

Upgrade `kube-apiserver` to **1.(n+1)**

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

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

Pre-requisites:

- The `kube-apiserver` instances these components communicate with are at **1.(n+1)** (in HA clusters in which these control plane components can communicate with any `kube-apiserver` instance in the cluster, all

`kube-apiserver` instances must be upgraded before upgrading these components)

Upgrade `kube-controller-manager`, `kube-scheduler`, and `cloud-controller-manager` to **1.(n+1)**

kubelet

Pre-requisites:

- The `kube-apiserver` instances the `kubelet` communicates with are at **1.(n+1)**

Optionally upgrade `kubelet` instances to **1.(n+1)** (or they can be left at **1.n** or **1.(n-1)**)

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

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 20, 2019 at 7:05 PM PST by Fix relative links issue in English content (#13307) (Page History)

Edit This Page

Building Large Clusters

Support

At v1.13, Kubernetes supports clusters with up to 5000 nodes. More specifically, we support configurations that meet *all* of the following criteria:

- No more than 5000 nodes
- No more than 150000 total pods
- No more than 300000 total containers
- No more than 100 pods per node

- – Support
  – Setup
    * Quota Issues
    * Etcd storage
    * Size of master and master components
    * Addon Resources
    * Allowing minor node failure at startup

Setup

A cluster is a set of nodes (physical or virtual machines) running Kubernetes agents, managed by a "master" (the cluster-level control plane).

Normally the number of nodes in a cluster is controlled by the value `NUM_NODES` in the platform-specific `config-default.sh` file (for example, see GCE's `config-default.sh`).

Simply changing that value to something very large, however, may cause the setup script to fail for many cloud providers. A GCE deployment, for example, will run in to quota issues and fail to bring the cluster up.

When setting up a large Kubernetes cluster, the following issues must be considered.

Quota Issues

To avoid running into cloud provider quota issues, when creating a cluster with many nodes, consider:

- Increase the quota for things like CPU, IPs, etc.
  – In GCE, for example, you'll want to increase the quota for:
  – CPUs
  – VM instances
  – Total persistent disk reserved

  - In-use IP addresses
  - Firewall Rules
  - Forwarding rules
  - Routes
  - Target pools
- Gating the setup script so that it brings up new node VMs in smaller batches with waits in between, because some cloud providers rate limit the creation of VMs.

Etcd storage

To improve performance of large clusters, we store events in a separate dedicated etcd instance.

When creating a cluster, existing salt scripts:

- start and configure additional etcd instance
- configure api-server to use it for storing events

Size of master and master components

On GCE/Google Kubernetes Engine, and AWS, `kube-up` automatically configures the proper VM size for your master depending on the number of nodes in your cluster. On other providers, you will need to configure it manually. For reference, the sizes we use on GCE are

- 1-5 nodes: n1-standard-1
- 6-10 nodes: n1-standard-2
- 11-100 nodes: n1-standard-4
- 101-250 nodes: n1-standard-8
- 251-500 nodes: n1-standard-16
- more than 500 nodes: n1-standard-32

And the sizes we use on AWS are

- 1-5 nodes: m3.medium
- 6-10 nodes: m3.large
- 11-100 nodes: m3.xlarge
- 101-250 nodes: m3.2xlarge
- 251-500 nodes: c4.4xlarge
- more than 500 nodes: c4.8xlarge

  Note:

  On Google Kubernetes Engine, the size of the master node adjusts automatically based on the size of your cluster. For more information, see this blog post.

On AWS, master node sizes are currently set at cluster startup time and do not change, even if you later scale your cluster up or down by manually removing or adding nodes or using a cluster autoscaler.

Addon Resources

To prevent memory leaks or other resource issues in cluster addons from consuming all the resources available on a node, Kubernetes sets resource limits on addon containers to limit the CPU and Memory resources they can consume (See PR #10653 and #10778).

For example:

```
containers:
- name: fluentd-cloud-logging
  image: k8s.gcr.io/fluentd-gcp:1.16
  resources:
    limits:
      cpu: 100m
      memory: 200Mi
```

Except for Heapster, these limits are static and are based on data we collected from addons running on 4-node clusters (see #10335). The addons consume a lot more resources when running on large deployment clusters (see #5880). So, if a large cluster is deployed without adjusting these values, the addons may continuously get killed because they keep hitting the limits.

To avoid running into cluster addon resource issues, when creating a cluster with many nodes, consider the following:

- Scale memory and CPU limits for each of the following addons, if used, as you scale up the size of cluster (there is one replica of each handling the entire cluster so memory and CPU usage tends to grow proportionally with size/load on cluster):
  - InfluxDB and Grafana
  - kubedns, dnsmasq, and sidecar
  - Kibana
- Scale number of replicas for the following addons, if used, along with the size of cluster (there are multiple replicas of each so increasing replicas should help handle increased load, but, since load per replica also increases slightly, also consider increasing CPU/memory limits):
  - elasticsearch
- Increase memory and CPU limits slightly for each of the following addons, if used, along with the size of cluster (there is one replica per node but CPU/memory usage increases slightly along with cluster load/size as well):
  - FluentD with ElasticSearch Plugin
  - FluentD with GCP Plugin

Heapster's resource limits are set dynamically based on the initial size of your cluster (see #16185 and #22940). If you find that Heapster is running out of resources, you should adjust the formulas that compute heapster memory request (see those PRs for details).

For directions on how to detect if addon containers are hitting resource limits, see the Troubleshooting section of Compute Resources.

In the future, we anticipate to set all cluster addon resource limits based on cluster size, and to dynamically adjust them if you grow or shrink your cluster. We welcome PRs that implement those features.

Allowing minor node failure at startup

For various reasons (see #18969 for more details) running `kube-up.sh` with a very large `NUM_NODES` may fail due to a very small number of nodes not coming up properly. Currently you have two choices: restart the cluster (`kube-down.sh` and then `kube-up.sh` again), or before running `kube-up.sh` set the environment variable `ALLOWED_NOTREADY_NODES` to whatever value you feel comfortable with. This will allow `kube-up.sh` to succeed with fewer than `NUM_NODES` coming up. Depending on the reason for the failure, those additional nodes may join later or the cluster may remain at a size of `NUM_NODES - ALLOWED_NOTREADY_NODES`.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on August 20, 2018 at 11:31 AM PST by Refactor Setup and Imported sections (#9916) (Page History)

Edit This Page

Running in Multiple Zones

This page describes how to run a cluster in multiple zones.

- Introduction
- Functionality
- Limitations
- Walkthrough

Introduction

Kubernetes 1.2 adds support for running a single cluster in multiple failure zones (GCE calls them simply "zones", AWS calls them "availability zones", here we'll refer to them as "zones"). This is a lightweight version of a broader Cluster Federation feature (previously referred to by the affectionate nickname "Ubernetes"). Full Cluster Federation allows combining separate Kubernetes clusters running in different regions or cloud providers (or on-premises data centers). However, many users simply want to run a more available Kubernetes cluster in multiple zones of their single cloud provider, and this is what the multizone support in 1.2 allows (this previously went by the nickname "Ubernetes Lite").

Multizone support is deliberately limited: a single Kubernetes cluster can run in multiple zones, but only within the same region (and cloud provider). Only GCE and AWS are currently supported automatically (though it is easy to add similar support for other clouds or even bare metal, by simply arranging for the appropriate labels to be added to nodes and volumes).

Functionality

When nodes are started, the kubelet automatically adds labels to them with zone information.

Kubernetes will automatically spread the pods in a replication controller or service across nodes in a single-zone cluster (to reduce the impact of failures.) With multiple-zone clusters, this spreading behavior is extended across zones (to reduce the impact of zone failures.) (This is achieved via `SelectorSpreadPriority`). This is a best-effort placement, and so if the zones in your cluster are heterogeneous (e.g. different numbers of nodes, different types of nodes, or different pod resource requirements), this might prevent perfectly even spreading of your pods across zones. If desired, you can use homogeneous zones (same number and types of nodes) to reduce the probability of unequal spreading.

When persistent volumes are created, the `PersistentVolumeLabel` admission controller automatically adds zone labels to them. The scheduler (via the

`VolumeZonePredicate` predicate) will then ensure that pods that claim a given volume are only placed into the same zone as that volume, as volumes cannot be attached across zones.

Limitations

There are some important limitations of the multizone support:

- We assume that the different zones are located close to each other in the network, so we don't perform any zone-aware routing. In particular, traffic that goes via services might cross zones (even if some pods backing that service exist in the same zone as the client), and this may incur additional latency and cost.
- Volume zone-affinity will only work with a `PersistentVolume`, and will not work if you directly specify an EBS volume in the pod spec (for example).
- Clusters cannot span clouds or regions (this functionality will require full federation support).
- Although your nodes are in multiple zones, kube-up currently builds a single master node by default. While services are highly available and can tolerate the loss of a zone, the control plane is located in a single zone. Users that want a highly available control plane should follow the high availability instructions.

Volume limitations

The following limitations are addressed with topology-aware volume binding.

- StatefulSet volume zone spreading when using dynamic provisioning is currently not compatible with pod affinity or anti-affinity policies.
- If the name of the StatefulSet contains dashes ("-"), volume zone spreading may not provide a uniform distribution of storage across zones.
- When specifying multiple PVCs in a Deployment or Pod spec, the StorageClass needs to be configured for a specific single zone, or the PVs need to be statically provisioned in a specific zone. Another workaround is to use a StatefulSet, which will ensure that all the volumes for a replica are provisioned in the same zone.

Walkthrough

We're now going to walk through setting up and using a multi-zone cluster on both GCE & AWS. To do so, you bring up a full cluster (specifying

`MULTIZONE=true`), and then you add nodes in additional zones by running `kube-up` again (specifying `KUBE_USE_EXISTING_MASTER=true`).

Bringing up your cluster

Create the cluster as normal, but pass MULTIZONE to tell the cluster to manage multiple zones; creating nodes in us-central1-a.

GCE:

```
curl -sS https://get.k8s.io | MULTIZONE=true KUBERNETES_PROVIDER=gce KUBE_GCE_ZONE=us-centra
```

AWS:

```
curl -sS https://get.k8s.io | MULTIZONE=true KUBERNETES_PROVIDER=aws KUBE_AWS_ZONE=us-west-2
```

This step brings up a cluster as normal, still running in a single zone (but `MULTIZONE=true` has enabled multi-zone capabilities).

Nodes are labeled

View the nodes; you can see that they are labeled with zone information. They are all in `us-central1-a` (GCE) or `us-west-2a` (AWS) so far. The labels are `failure-domain.beta.kubernetes.io/region` for the region, and `failure-domain.beta.kubernetes.io/zone` for the zone:

```
kubectl get nodes --show-labels
```

The output is similar to this:

```
NAME                     STATUS                     ROLES    AGE   VERSION          LABELS
kubernetes-master        Ready,SchedulingDisabled   <none>   6m    v1.13.0          beta.kub
kubernetes-minion-87j9   Ready                      <none>   6m    v1.13.0          beta.kub
kubernetes-minion-9vlv   Ready                      <none>   6m    v1.13.0          beta.kub
kubernetes-minion-a12q   Ready                      <none>   6m    v1.13.0          beta.kub
```

Add more nodes in a second zone

Let's add another set of nodes to the existing cluster, reusing the existing master, running in a different zone (us-central1-b or us-west-2b). We run kube-up again, but by specifying `KUBE_USE_EXISTING_MASTER=true` kube-up will not create a new master, but will reuse one that was previously created instead.

GCE:

```
KUBE_USE_EXISTING_MASTER=true MULTIZONE=true KUBERNETES_PROVIDER=gce KUBE_GCE_ZONE=us-centra
```

On AWS we also need to specify the network CIDR for the additional subnet, along with the master internal IP address:

```
KUBE_USE_EXISTING_MASTER=true MULTIZONE=true KUBERNETES_PROVIDER=aws KUBE_AWS_ZONE=us-west-2
```

View the nodes again; 3 more nodes should have launched and be tagged in us-central1-b:

```
kubectl get nodes --show-labels
```

The output is similar to this:

```
NAME                     STATUS                     ROLES    AGE   VERSION           LABELS
kubernetes-master        Ready,SchedulingDisabled   <none>   16m   v1.13.0           beta.ku
kubernetes-minion-281d   Ready                      <none>   2m    v1.13.0           beta.ku
kubernetes-minion-87j9   Ready                      <none>   16m   v1.13.0           beta.ku
kubernetes-minion-9vlv   Ready                      <none>   16m   v1.13.0           beta.ku
kubernetes-minion-a12q   Ready                      <none>   17m   v1.13.0           beta.ku
kubernetes-minion-pp2f   Ready                      <none>   2m    v1.13.0           beta.ku
kubernetes-minion-wf8i   Ready                      <none>   2m    v1.13.0           beta.ku
```

Volume affinity

Create a volume using the dynamic volume creation (only PersistentVolumes are supported for zone affinity):

```
kubectl create -f - <<EOF
{
  "kind": "PersistentVolumeClaim",
  "apiVersion": "v1",
  "metadata": {
    "name": "claim1",
    "annotations": {
        "volume.alpha.kubernetes.io/storage-class": "foo"
    }
  },
  "spec": {
    "accessModes": [
      "ReadWriteOnce"
    ],
    "resources": {
      "requests": {
        "storage": "5Gi"
      }
    }
  }
}
EOF
```

Note: For version 1.3+ Kubernetes will distribute dynamic PV claims across the configured zones. For version 1.2, dynamic persis-

tent volumes were always created in the zone of the cluster master (here us-central1-a / us-west-2a); that issue (#23330) was addressed in 1.3+.

Now let's validate that Kubernetes automatically labeled the zone & region the PV was created in.

```
kubectl get pv --show-labels
```

The output is similar to this:

```
NAME           CAPACITY   ACCESSMODES   RECLAIM POLICY   STATUS    CLAIM            STORAGEC
pv-gce-mj4gm   5Gi        RWO           Retain           Bound     default/claim1   manual
```

So now we will create a pod that uses the persistent volume claim. Because GCE PDs / AWS EBS volumes cannot be attached across zones, this means that this pod can only be created in the same zone as the volume:

```
kubectl create -f - <<EOF
kind: Pod
apiVersion: v1
metadata:
  name: mypod
spec:
  containers:
    - name: myfrontend
      image: nginx
      volumeMounts:
      - mountPath: "/var/www/html"
        name: mypd
  volumes:
    - name: mypd
      persistentVolumeClaim:
        claimName: claim1
EOF
```

Note that the pod was automatically created in the same zone as the volume, as cross-zone attachments are not generally permitted by cloud providers:

```
kubectl describe pod mypod | grep Node
```

```
Node:        kubernetes-minion-9vlv/10.240.0.5
```

And check node labels:

```
kubectl get node kubernetes-minion-9vlv --show-labels
```

```
NAME                     STATUS    AGE    VERSION          LABELS
kubernetes-minion-9vlv   Ready     22m    v1.6.0+fff5156   beta.kubernetes.io/instance-type=
```

Pods are spread across zones

Pods in a replication controller or service are automatically spread across zones. First, let's launch more nodes in a third zone:

GCE:

```
KUBE_USE_EXISTING_MASTER=true MULTIZONE=true KUBERNETES_PROVIDER=gce KUBE_GCE_ZONE=us-centra
```

AWS:

```
KUBE_USE_EXISTING_MASTER=true MULTIZONE=true KUBERNETES_PROVIDER=aws KUBE_AWS_ZONE=us-west-2
```

Verify that you now have nodes in 3 zones:

```
kubectl get nodes --show-labels
```

Create the guestbook-go example, which includes an RC of size 3, running a simple web app:

```
find kubernetes/examples/guestbook-go/ -name '*.json' | xargs -I {} kubectl create -f {}
```

The pods should be spread across all 3 zones:

```
kubectl describe pod -l app=guestbook | grep Node
```

```
Node:        kubernetes-minion-9vlv/10.240.0.5
Node:        kubernetes-minion-281d/10.240.0.8
Node:        kubernetes-minion-olsh/10.240.0.11
```

```
kubectl get node kubernetes-minion-9vlv kubernetes-minion-281d kubernetes-minion-olsh --show
```

```
NAME                     STATUS    ROLES    AGE    VERSION          LABELS
kubernetes-minion-9vlv   Ready     <none>   34m    v1.13.0          beta.kubernetes.io/insta
kubernetes-minion-281d   Ready     <none>   20m    v1.13.0          beta.kubernetes.io/insta
kubernetes-minion-olsh   Ready     <none>   3m     v1.13.0          beta.kubernetes.io/insta
```

Load-balancers span all zones in a cluster; the guestbook-go example includes an example load-balanced service:

```
kubectl describe service guestbook | grep LoadBalancer.Ingress
```

The output is similar to this:

```
LoadBalancer Ingress:   130.211.126.21
```

Set the above IP:

```
export IP=130.211.126.21
```

Explore with curl via IP:

```
curl -s http://${IP}:3000/env | grep HOSTNAME
```

The output is similar to this:

```
  "HOSTNAME": "guestbook-44sep",
```

Again, explore multiple times:

```
(for i in `seq 20`; do curl -s http://${IP}:3000/env | grep HOSTNAME; done)  | sort | uniq
```

The output is similar to this:

```
  "HOSTNAME": "guestbook-44sep",
  "HOSTNAME": "guestbook-hum5n",
  "HOSTNAME": "guestbook-ppm40",
```

The load balancer correctly targets all the pods, even though they are in multiple zones.

Shutting down the cluster

When you're done, clean up:

GCE:

```
KUBERNETES_PROVIDER=gce KUBE_USE_EXISTING_MASTER=true KUBE_GCE_ZONE=us-central1-f kubernetes
KUBERNETES_PROVIDER=gce KUBE_USE_EXISTING_MASTER=true KUBE_GCE_ZONE=us-central1-b kubernetes
KUBERNETES_PROVIDER=gce KUBE_GCE_ZONE=us-central1-a kubernetes/cluster/kube-down.sh
```

AWS:

```
KUBERNETES_PROVIDER=aws KUBE_USE_EXISTING_MASTER=true KUBE_AWS_ZONE=us-west-2c kubernetes/cl
KUBERNETES_PROVIDER=aws KUBE_USE_EXISTING_MASTER=true KUBE_AWS_ZONE=us-west-2b kubernetes/cl
KUBERNETES_PROVIDER=aws KUBE_AWS_ZONE=us-west-2a kubernetes/cluster/kube-down.sh
```

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 28, 2019 at 1:42 PM PST by apply content_template and follow the style guide (#12759) (Page History)

Edit This Page

CRI installation

FEATURE STATE: `Kubernetes v1.6` stable

This feature is *stable*, meaning:

- The version name is vX where X is an integer.
- Stable versions of features will appear in released software for many subsequent versions.

To run containers in Pods, Kubernetes uses a container runtime. Here are the installation instruction for various runtimes.

- Docker
- CRI-O
- Containerd
- Other CRI runtimes: frakti

  Caution:

  A flaw was found in the way runc handled system file descriptors when running containers. A malicious container could use this flaw to overwrite contents of the runc binary and consequently run arbitrary commands on the container host system.

  Please refer to this link for more information about this issue cve-2019-5736 : runc vulnerability

Applicability

> **Note:** This document is written for users installing CRI onto Linux. For other operating systems, look for documentation specific to your platform.

You should execute all the commands in this guide as `root`. For example, prefix commands with `sudo`, or become `root` and run the commands as that user.

Cgroup drivers

When systemd is chosen as the init system for a Linux distribution, the init process generates and consumes a root control group (`cgroup`) and acts as a cgroup manager. Systemd has a tight integration with cgroups and will allocate cgroups per process. It's possible to configure your container runtime and the kubelet to use `cgroupfs`. Using `cgroupfs` alongside systemd means that there will then be two different cgroup managers.

Control groups are used to constrain resources that are allocated to processes. A single cgroup manager will simplify the view of what resources are being

allocated and will by default have a more consistent view of the available and in-use resources. When we have two managers we end up with two views of those resources. We have seen cases in the field where nodes that are configured to use `cgroupfs` for the kubelet and Docker, and `systemd` for the rest of the processes running on the node becomes unstable under resource pressure.

Changing the settings such that your container runtime and kubelet use `systemd` as the cgroup driver stabilized the system. Please note the `native.cgroupdriver=systemd` option in the Docker setup below.

Docker

On each of your machines, install Docker. Version 18.06.2 is recommended, but 1.11, 1.12, 1.13, 17.03 and 18.09 are known to work as well. Keep track of the latest verified Docker version in the Kubernetes release notes.

Use the following commands to install Docker on your system:

- Ubuntu 16.04
- CentOS/RHEL 7.4+

```
# Install Docker CE
## Set up the repository:
### Update the apt package index
    apt-get update

### Install packages to allow apt to use a repository over HTTPS
    apt-get update && apt-get install apt-transport-https ca-certificates curl software-prop

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

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

## Install docker ce.
apt-get update && apt-get install docker-ce=18.06.2~ce~3-0~ubuntu

# Setup daemon.
cat > /etc/docker/daemon.json <<EOF
{
  "exec-opts": ["native.cgroupdriver=systemd"],
  "log-driver": "json-file",
  "log-opts": {
```

```
    "max-size": "100m"
  },
  "storage-driver": "overlay2"
}
EOF

mkdir -p /etc/systemd/system/docker.service.d

# Restart docker.
systemctl daemon-reload
systemctl restart docker


# Install Docker CE
## Set up the repository
### Install required packages.
    yum install yum-utils device-mapper-persistent-data lvm2

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

## Install docker ce.
yum update && yum install docker-ce-18.06.2.ce

## Create /etc/docker directory.
mkdir /etc/docker

# Setup daemon.
cat > /etc/docker/daemon.json <<EOF
{
  "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

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

```
# Restart docker.
systemctl daemon-reload
systemctl restart docker
```

Refer to the official Docker installation guides for more information.

CRI-O

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

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

Prerequisites

```
modprobe overlay
modprobe br_netfilter

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

sysctl --system
```

- Ubuntu 16.04
- CentOS/RHEL 7.4+

```
# Install prerequisites
apt-get update
apt-get install software-properties-common

add-apt-repository ppa:projectatomic/ppa
apt-get update

# Install CRI-O
apt-get install cri-o-1.11
```

```
# Install prerequisites
yum-config-manager --add-repo=https://cbs.centos.org/repos/paas7-crio-311-candidate/x86_64/c

# Install CRI-O
yum install --nogpgcheck cri-o
```

Start CRI-O

```
systemctl start crio
```

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

Containerd

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

Use the following commands to install Containerd on your system:

Prerequisites

```
modprobe overlay
modprobe br_netfilter

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

sysctl --system
```

- Ubuntu 16.04+
- CentOS/RHEL 7.4+

```
apt-get install -y libseccomp2
```

```
yum install -y libseccomp
```

Install containerd

Containerd releases are published regularly, the values below are hardcoded to the latest version available at the time of writing. Please check for newer versions and hashes here.

```
# Export required environment variables.
export CONTAINERD_VERSION="1.1.2"
export CONTAINERD_SHA256="d4ed54891e90a5d1a45e3e96464e2e8a4770cd380c21285ef5c9895c40549218"

# Download containerd tar.
wget https://storage.googleapis.com/cri-containerd-release/cri-containerd-${CONTAINERD_VERSI
```

```
# Check hash.
echo "${CONTAINERD_SHA256} cri-containerd-${CONTAINERD_VERSION}.linux-amd64.tar.gz" | sha256

# Unpack.
tar --no-overwrite-dir -C / -xzf cri-containerd-${CONTAINERD_VERSION}.linux-amd64.tar.gz

# Start containerd.
systemctl start containerd
```

Other CRI runtimes: frakti

Refer to the Frakti QuickStart guide for more information.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 25, 2019 at 4:37 PM PST by Tidy install guide for container runtimes (#12704) (Page History)

Edit This Page

Installing Kubernetes with Digital Rebar Provision (DRP) via KRIB

- – Overview
  - – Creating a cluster
    - * (1/5) Discover servers
    - * (2/5) Install KRIB Content and Certificate Plugin
    - * (3/5) Start your cluster deployment
    - * (4/5) Monitor your cluster deployment
    - * (5/5) Access your cluster

- Cluster operations
  - Scale your cluster
  - Cleanup your cluster (for developers)
- Feedback

Overview

This guide helps to install a Kubernetes cluster hosted on bare metal with Digital Rebar Provision using only its Content packages and *kubeadm*.

Digital Rebar Provision (DRP) is an integrated Golang DHCP, bare metal provisioning (PXE/iPXE) and workflow automation platform. While DRP can be used to invoke kubespray, it also offers a self-contained Kubernetes installation known as KRIB (Kubernetes Rebar Integrated Bootstrap).

> **Note:** KRIB is not a *stand-alone* installer: Digital Rebar templates drive a standard *kubeadm* configuration that manages the Kubernetes installation with the Digital Rebar cluster pattern to elect leaders *without external supervision*.

KRIB features:

- zero-touch, self-configuring cluster without pre-configuration or inventory
- very fast, no-ssh required automation
- bare metal, on-premises focused platform
- highly available cluster options (including splitting etcd from the controllers)
- dynamic generation of a TLS infrastructure
- composable attributes and automatic detection of hardware by profile
- options for persistent, immutable and image-based deployments
- support for Ubuntu 18.04, CentOS/RHEL 7 and others

Creating a cluster

Review Digital Rebar documentation for details about installing the platform.

The Digital Rebar Provision Golang binary should be installed on a Linux-like system with 16 GB of RAM or larger (Packet.net Tiny and Rasberry Pi are also acceptable).

(1/5) Discover servers

Following the Digital Rebar installation, allow one or more servers to boot through the *Sledgehammer* discovery process to register with the API. This will automatically install the Digital Rebar runner and to allow for next steps.

(2/5) Install KRIB Content and Certificate Plugin

Upload the KRIB Content bundle (or build from source) and the Cert Plugin for your DRP platform (e.g.: amd64 Linux v2.4.0). Both are freely available via the RackN UX.

(3/5) Start your cluster deployment

> **Note:** KRIB documentation is dynamically generated from the source and will be more up to date than this guide.

Following the KRIB documentation, create a Profile for your cluster and assign your target servers into the cluster Profile. The Profile must set `krib\cluster-name` and `etcd\cluster-name` Params to be the name of the Profile. Cluster configuration choices can be made by adding additional Params to the Profile; however, safe defaults are provided for all Params.

Once all target servers are assigned to the cluster Profile, start a KRIB installation Workflow by assigning one of the included Workflows to all cluster servers. For example, selecting `krib-live-cluster` will perform an immutable deployment into the Sledgehammer discovery operating system. You may use one of the pre-created read-only Workflows or choose to build your own custom variation.

For basic installs, no further action is required. Advanced users may choose to assign the controllers, etcd servers or other configuration values in the relevant Params.

(4/5) Monitor your cluster deployment

Digital Rebar Provision provides detailed logging and live updates during the installation process. Workflow events are available via a websocket connection or monitoring the Jobs list.

During the installation, KRIB writes cluster configuration data back into the cluster Profile.

(5/5) Access your cluster

The cluster is available for access via *kubectl* once the `krib/cluster-admin-conf` Param has been set. This Param contains the `kubeconfig` information necessary to access the cluster.

For example, if you named the cluster Profile `krib` then the following commands would allow you to connect to the installed cluster from your local terminal.

::

```
drpcli profiles get krib params krib/cluster-admin-conf > admin.conf
export KUBECONFIG=admin.conf
kubectl get nodes
```

The installation continues after the `krib/cluster-admin-conf` is set to install the Kubernetes UI and Helm. You may interact with the cluster as soon as the `admin.conf` file is available.

Cluster operations

KRIB provides additional Workflows to manage your cluster. Please see the KRIB documentation for an updated list of advanced cluster operations.

Scale your cluster

You can add servers into your cluster by adding the cluster Profile to the server and running the appropriate Workflow.

Cleanup your cluster (for developers)

You can reset your cluster and wipe out all configuration and TLS certificates using the `krib-reset-cluster` Workflow on any of the servers in the cluster.

> **Caution:** When running the reset Workflow, be sure not to accidentally target your production cluster!

Feedback

- Slack Channel: #community
- GitHub Issues

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on February 12, 2019 at 1:45 PM PST by Update krib.md (#12580) (Page History)

Edit This Page

PKI Certificates and Requirements

Kubernetes requires PKI certificates for authentication over TLS. If you install Kubernetes with kubeadm, the certificates that your cluster requires are automatically generated. You can also generate your own certificates – for example, to keep your private keys more secure by not storing them on the API server. This page explains the certificates that your cluster requires.

- How certificates are used by your cluster
- Where certificates are stored
- Configure certificates manually
- Configure certificates for user accounts

How certificates are used by your cluster

Kubernetes requires PKI for the following operations:

- Client certificates for the kubelet to authenticate to the API server
- Server certificate for the API server endpoint
- Client certificates for administrators of the cluster to authenticate to the API server
- Client certificates for the API server to talk to the kubelets
- Client certificate for the API server to talk to etcd
- Client certificate/kubeconfig for the controller manager to talk to the API server
- Client certificate/kubeconfig for the scheduler to talk to the API server.
- Client and server certificates for the front-proxy

  Note: `front-proxy` certificates are required only if you run kube-proxy to support an extension API server.

etcd also implements mutual TLS to authenticate clients and peers.

Where certificates are stored

If you install Kubernetes with kubeadm, certificates are stored in `/etc/kubernetes/pki`. All paths in this documentation are relative to that directory.

Configure certificates manually

If you don't want kubeadm to generate the required certificates, you can create them in either of the following ways.

Single root CA

You can create a single root CA, controlled by an administrator. This root CA can then create multiple intermediate CAs, and delegate all further creation to Kubernetes itself.

Required CAs:

path	Default CN	description
ca.crt,key	kubernetes-ca	Kubernetes general CA
etcd/ca.crt,key	etcd-ca	For all etcd-related functions
front-proxy-ca.crt,key	kubernetes-front-proxy-ca	For the front-end proxy

All certificates

If you don't wish to copy these private keys to your API servers, you can generate all certificates yourself.

Required certificates:

Default CN	Parent CA	O (in Subject)	kind	hosts (SAN)
kube-etcd	etcd-ca		server, client	`localhost`, `127.`
kube-etcd-peer	etcd-ca		server, client	`<hostname>`, `<Ho`
kube-etcd-healthcheck-client	etcd-ca		client	
kube-apiserver-etcd-client	etcd-ca	system:masters	client	
kube-apiserver	kubernetes-ca		server	`<hostname>`, `<Ho`
kube-apiserver-kubelet-client	kubernetes-ca	system:masters	client	
front-proxy-client	kubernetes-front-proxy-ca		client	

[1]: `kubernetes`, `kubernetes.default`, `kubernetes.default.svc`, `kubernetes.default.svc.cluster`, `kubernetes.default.svc.cluster.local`

where `kind` maps to one or more of the x509 key usage types:

kind	Key usage
server	digital signature, key encipherment, server auth
client	digital signature, key encipherment, client auth

Certificate paths

Certificates should be placed in a recommended path (as used by kubeadm).
Paths should be specified using the given argument regardless of location.

Default CN	recommend key path	recommended cert path	command	ke
etcd-ca		etcd/ca.crt	kube-apiserver	
etcd-client	apiserver-etcd-client.key	apiserver-etcd-client.crt	kube-apiserver	–e
kubernetes-ca		ca.crt	kube-apiserver	
kube-apiserver	apiserver.key	apiserver.crt	kube-apiserver	–t
apiserver-kubelet-client		apiserver-kubelet-client.crt	kube-apiserver	
front-proxy-ca		front-proxy-ca.crt	kube-apiserver	
front-proxy-client	front-proxy-client.key	front-proxy-client.crt	kube-apiserver	–p
etcd-ca		etcd/ca.crt	etcd	
kube-etcd	etcd/server.key	etcd/server.crt	etcd	–k
kube-etcd-peer	etcd/peer.key	etcd/peer.crt	etcd	–p
etcd-ca		etcd/ca.crt	etcdctl[2]	
kube-etcd-healthcheck-client	etcd/healthcheck-client.key	etcd/healthcheck-client.crt	etcdctl[2]	–k

[2]: For a liveness probe, if self-hosted

Configure certificates for user accounts

You must manually configure these administrator account and service accounts:

filename	credential name	Default CN	O (in Subject)
admin.conf	default-admin	kubernetes-admin	system:masters
kubelet.conf	default-auth	system:node:`<nodeName>` (see note)	system:nodes
controller-manager.conf	default-controller-manager	system:kube-controller-manager	
scheduler.conf	default-manager	system:kube-scheduler	

> **Note:** The value of `<nodeName>` for `kubelet.conf` **must** match precisely the value of the node name provided by the kubelet as it registers with the apiserver. For further details, read the Node Authorization.

1. For each config, generate an x509 cert/key pair with the given CN and O.
2. Run `kubectl` as follows for each config:

```
KUBECONFIG=<filename> kubectl config set-cluster default-cluster --server=https://<host ip>
KUBECONFIG=<filename> kubectl config set-credentials <credential-name> --client-key <path-to
KUBECONFIG=<filename> kubectl config set-context default-system --cluster default-cluster --
```

```
KUBECONFIG=<filename> kubectl config use-context default-system
```

These files are used as follows:

filename	command	comment
admin.conf	kubectl	Configures administrator user for the cluster
kubelet.conf	kubelet	One required for each node in the cluster.
controller-manager.conf	kube-controller-manager	Must be added to manifest in `manifests/kube-control`
scheduler.conf	kube-scheduler	Must be added to manifest in `manifests/kube-schedul`

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 20, 2019 at 6:14 AM PST by Remove mysterious "[1][etcdbug]" (#13266) (Page History)

Edit This Page

Running Kubernetes Locally via Minikube

Minikube is a tool that makes it easy to run Kubernetes locally. Minikube runs a single-node Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

- Minikube Features
- Installation
- Quickstart
- Managing your Cluster
- Interacting with Your Cluster
- Networking
- Persistent Volumes
- Mounted Host Folders

- Private Container Registries
- Add-ons
- Using Minikube with an HTTP Proxy
- Known Issues
- Design
- Additional Links
- Community

Minikube Features

- Minikube supports Kubernetes features such as:
  - DNS
  - NodePorts
  - ConfigMaps and Secrets
  - Dashboards
  - Container Runtime: Docker, rkt, CRI-O and containerd
  - Enabling CNI (Container Network Interface)
  - Ingress

Installation

See Installing Minikube.

Quickstart

Here's a brief demo of Minikube usage. If you want to change the VM driver add the appropriate `--vm-driver=xxx` flag to `minikube start`. Minikube supports the following drivers:

- virtualbox
- vmwarefusion
- kvm2 (driver installation)
- kvm (driver installation)
- hyperkit (driver installation)
- xhyve (driver installation) (deprecated)
- hyperv (driver installation) Note that the IP below is dynamic and can change. It can be retrieved with `minikube ip`.
- none (Runs the Kubernetes components on the host and not in a VM. Using this driver requires Docker (docker install) and a Linux environment)

```
minikube start
```

```
Starting local Kubernetes cluster...
Running pre-create checks...
```

```
Creating machine...
Starting local Kubernetes cluster...

kubectl run hello-minikube --image=k8s.gcr.io/echoserver:1.10 --port=8080

deployment.apps/hello-minikube created

kubectl expose deployment hello-minikube --type=NodePort

service/hello-minikube exposed

# We have now launched an echoserver pod but we have to wait until the pod is up before curl
# via the exposed service.
# To check whether the pod is up and running we can use the following:
kubectl get pod

NAME                              READY     STATUS              RESTARTS   AGE
hello-minikube-3383150820-vctvh   0/1       ContainerCreating   0          3s

# We can see that the pod is still being created from the ContainerCreating status
kubectl get pod

NAME                              READY     STATUS    RESTARTS   AGE
hello-minikube-3383150820-vctvh   1/1       Running   0          13s

# We can see that the pod is now Running and we will now be able to curl it:
curl $(minikube service hello-minikube --url)

Hostname: hello-minikube-7c77b68cff-8wdzq

Pod Information:
    -no pod information available-

Server values:
    server_version=nginx: 1.13.3 - lua: 10008

Request Information:
    client_address=172.17.0.1
    method=GET
    real path=/
    query=
    request_version=1.1
    request_scheme=http
    request_uri=http://192.168.99.100:8080/

Request Headers:
    accept=*/*
    host=192.168.99.100:30674
    user-agent=curl/7.47.0

Request Body:
```

```
    -no body in request-
kubectl delete services hello-minikube
service "hello-minikube" deleted
kubectl delete deployment hello-minikube
deployment.extensions "hello-minikube" deleted
minikube stop
Stopping local Kubernetes cluster...
Stopping "minikube"...
```

Alternative Container Runtimes

containerd

To use containerd as the container runtime, run:

```
minikube start \
    --network-plugin=cni \
    --enable-default-cni \
    --container-runtime=containerd \
    --bootstrapper=kubeadm
```

Or you can use the extended version:

```
minikube start \
    --network-plugin=cni \
    --enable-default-cni \
    --extra-config=kubelet.container-runtime=remote \
    --extra-config=kubelet.container-runtime-endpoint=unix:///run/containerd/containerd.sock
    --extra-config=kubelet.image-service-endpoint=unix:///run/containerd/containerd.sock \
    --bootstrapper=kubeadm
```

CRI-O

To use CRI-O as the container runtime, run:

```
minikube start \
    --network-plugin=cni \
    --enable-default-cni \
    --container-runtime=cri-o \
    --bootstrapper=kubeadm
```

Or you can use the extended version:

```
minikube start \
    --network-plugin=cni \
```

```
    --enable-default-cni \
    --extra-config=kubelet.container-runtime=remote \
    --extra-config=kubelet.container-runtime-endpoint=/var/run/crio.sock \
    --extra-config=kubelet.image-service-endpoint=/var/run/crio.sock \
    --bootstrapper=kubeadm
```

rkt container engine

To use rkt as the container runtime run:

```
minikube start \
    --network-plugin=cni \
    --enable-default-cni \
    --container-runtime=rkt
```

This will use an alternative minikube ISO image containing both rkt, and Docker, and enable CNI networking.

Driver plugins

See DRIVERS for details on supported drivers and how to install plugins, if required.

Use local images by re-using the Docker daemon

When using a single VM of Kubernetes, it's really handy to reuse the Minikube's built-in Docker daemon; as this means you don't have to build a docker registry on your host machine and push the image into it - you can just build inside the same docker daemon as minikube which speeds up local experiments. Just make sure you tag your Docker image with something other than 'latest' and use that tag while you pull the image. Otherwise, if you do not specify version of your image, it will be assumed as `:latest`, with pull image policy of `Always` correspondingly, which may eventually result in `ErrImagePull` as you may not have any versions of your Docker image out there in the default docker registry (usually DockerHub) yet.

To be able to work with the docker daemon on your mac/linux host use the `docker-env command` in your shell:

```
eval $(minikube docker-env)
```

You should now be able to use docker on the command line on your host mac/linux machine talking to the docker daemon inside the minikube VM:

```
docker ps
```

On Centos 7, docker may report the following error:

```
Could not read CA certificate "/etc/docker/ca.pem": open /etc/docker/ca.pem: no such file or
```

The fix is to update /etc/sysconfig/docker to ensure that Minikube's environment changes are respected:

```
< DOCKER_CERT_PATH=/etc/docker
---
> if [ -z "${DOCKER_CERT_PATH}" ]; then
>   DOCKER_CERT_PATH=/etc/docker
> fi
```

Remember to turn off the imagePullPolicy:Always, otherwise Kubernetes won't use images you built locally.

Managing your Cluster

Starting a Cluster

The `minikube start` command can be used to start your cluster. This command creates and configures a Virtual Machine that runs a single-node Kubernetes cluster. This command also configures your kubectl installation to communicate with this cluster.

If you are behind a web proxy, you will need to pass this information to the `minikube start` command:

```
https_proxy=<my proxy> minikube start --docker-env http_proxy=<my proxy> --docker-env https_
```

Unfortunately just setting the environment variables will not work.

Minikube will also create a "minikube" context, and set it to default in kubectl. To switch back to this context later, run this command: `kubectl config use-context minikube`.

Specifying the Kubernetes version

You can specify the specific version of Kubernetes for Minikube to use by adding the `--kubernetes-version` string to the `minikube start` command. For example, to run version `v1.7.3`, you would run the following:

```
minikube start --kubernetes-version v1.7.3
```

Configuring Kubernetes

Minikube has a "configurator" feature that allows users to configure the Kubernetes components with arbitrary values. To use this feature, you can use the `--extra-config` flag on the `minikube start` command.

This flag is repeated, so you can pass it several times with several different values to set multiple options.

This flag takes a string of the form `component.key=value`, where `component` is one of the strings from the below list, `key` is a value on the configuration struct and `value` is the value to set.

Valid keys can be found by examining the documentation for the Kubernetes `componentconfigs` for each component. Here is the documentation for each supported configuration:

- kubelet
- apiserver
- proxy
- controller-manager
- etcd
- scheduler

Examples

To change the `MaxPods` setting to 5 on the Kubelet, pass this flag: `--extra-config=kubelet.MaxPods=5`.

This feature also supports nested structs. To change the `LeaderElection.LeaderElect` setting to `true` on the scheduler, pass this flag: `--extra-config=scheduler.LeaderElection.LeaderElect=t`

To set the `AuthorizationMode` on the `apiserver` to `RBAC`, you can use: `--extra-config=apiserver.authorization-mode=RBAC`.

Stopping a Cluster

The `minikube stop` command can be used to stop your cluster. This command shuts down the Minikube Virtual Machine, but preserves all cluster state and data. Starting the cluster again will restore it to its previous state.

Deleting a Cluster

The `minikube delete` command can be used to delete your cluster. This command shuts down and deletes the Minikube Virtual Machine. No data or state is preserved.

Interacting with Your Cluster

Kubectl

The `minikube start` command creates a kubectl context called "minikube". This context contains the configuration to communicate with your Minikube cluster.

Minikube sets this context to default automatically, but if you need to switch back to it in the future, run:

`kubectl config use-context minikube`,

Or pass the context on each command like this: `kubectl get pods --context=minikube`.

Dashboard

To access the Kubernetes Dashboard, run this command in a shell after starting Minikube to get the address:

```
minikube dashboard
```

Services

To access a service exposed via a node port, run this command in a shell after starting Minikube to get the address:

```
minikube service [-n NAMESPACE] [--url] NAME
```

Networking

The Minikube VM is exposed to the host system via a host-only IP address, that can be obtained with the `minikube ip` command. Any services of type `NodePort` can be accessed over that IP address, on the NodePort.

To determine the NodePort for your service, you can use a `kubectl` command like this:

```
kubectl get service $SERVICE --output='jsonpath="{.spec.ports[0].nodePort}"'
```

Persistent Volumes

Minikube supports PersistentVolumes of type `hostPath`. These PersistentVolumes are mapped to a directory inside the Minikube VM.

The Minikube VM boots into a tmpfs, so most directories will not be persisted across reboots (`minikube stop`). However, Minikube is configured to persist files stored under the following host directories:

- `/data`
- `/var/lib/minikube`
- `/var/lib/docker`

Here is an example PersistentVolume config to persist data in the `/data` directory:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv0001
spec:
  accessModes:
    - ReadWriteOnce
  capacity:
    storage: 5Gi
  hostPath:
    path: /data/pv0001/
```

Mounted Host Folders

Some drivers will mount a host folder within the VM so that you can easily share files between the VM and host. These are not configurable at the moment and different for the driver and OS you are using.

> **Note:** Host folder sharing is not implemented in the KVM driver yet.

Driver	OS	HostFolder	VM
VirtualBox	Linux	/home	/hosthome
VirtualBox	macOS	/Users	/Users
VirtualBox	Windows	C://Users	/c/Users
VMware Fusion	macOS	/Users	/Users
Xhyve	macOS	/Users	/Users

Private Container Registries

To access a private container registry, follow the steps on this page.

We recommend you use `ImagePullSecrets`, but if you would like to configure access on the Minikube VM you can place the `.dockercfg` in the `/home/docker`

directory or the `config.json` in the `/home/docker/.docker` directory.

Add-ons

In order to have Minikube properly start or restart custom addons, place the addons you wish to be launched with Minikube in the `~/.minikube/addons` directory. Addons in this folder will be moved to the Minikube VM and launched each time Minikube is started or restarted.

Using Minikube with an HTTP Proxy

Minikube creates a Virtual Machine that includes Kubernetes and a Docker daemon. When Kubernetes attempts to schedule containers using Docker, the Docker daemon may require external network access to pull containers.

If you are behind an HTTP proxy, you may need to supply Docker with the proxy settings. To do this, pass the required environment variables as flags during `minikube start`.

For example:

```
minikube start --docker-env http_proxy=http://$YOURPROXY:PORT \
                --docker-env https_proxy=https://$YOURPROXY:PORT
```

If your Virtual Machine address is 192.168.99.100, then chances are your proxy settings will prevent `kubectl` from directly reaching it. To by-pass proxy configuration for this IP address, you should modify your no_proxy settings. You can do so with:

```
export no_proxy=$no_proxy,$(minikube ip)
```

Known Issues

- Features that require a Cloud Provider will not work in Minikube. These include:
  - LoadBalancers
- Features that require multiple nodes. These include:
  - Advanced scheduling policies

Design

Minikube uses libmachine for provisioning VMs, and kubeadm to provision a Kubernetes cluster.

For more information about Minikube, see the proposal.

Additional Links

- **Goals and Non-Goals**: For the goals and non-goals of the Minikube project, please see our roadmap.
- **Development Guide**: See CONTRIBUTING.md for an overview of how to send pull requests.
- **Building Minikube**: For instructions on how to build/test Minikube from source, see the build guide.
- **Adding a New Dependency**: For instructions on how to add a new dependency to Minikube, see the adding dependencies guide.
- **Adding a New Addon**: For instructions on how to add a new addon for Minikube, see the adding an addon guide.
- **MicroK8s**: Linux users wishing to avoid running a virtual machine may consider MicroK8s as an alternative.

Community

Contributions, questions, and comments are all welcomed and encouraged! Minikube developers hang out on Slack in the #minikube channel (get an invitation here). We also have the kubernetes-dev Google Groups mailing list. If you are posting to the list please prefix your subject with "minikube: ".

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on March 13, 2019 at 11:35 PM PST by Fix typos (#13155) (Page History)

Edit This Page

Validate Node Setup

- – Node Conformance Test
  – Limitations
  – Node Prerequisite
  – Running Node Conformance Test
  – Running Node Conformance Test for Other Architectures
  – Running Selected Test
  – Caveats

Node Conformance Test

Node conformance test is a containerized test framework that provides a system verification and functionality test for a node. The test validates whether the node meets the minimum requirements for Kubernetes; a node that passes the test is qualified to join a Kubernetes cluster.

Limitations

In Kubernetes version 1.5, node conformance test has the following limitations:

- Node conformance test only supports Docker as the container runtime.

Node Prerequisite

To run node conformance test, a node must satisfy the same prerequisites as a standard Kubernetes node. At a minimum, the node should have the following daemons installed:

- Container Runtime (Docker)
- Kubelet

Running Node Conformance Test

To run the node conformance test, perform the following steps:

1. Point your Kubelet to localhost `--api-servers="http://localhost:8080"`, because the test framework starts a local master to test Kubelet. There are some other Kubelet flags you may care:
   - `--pod-cidr`: If you are using `kubenet`, you should specify an arbitrary CIDR to Kubelet, for example `--pod-cidr=10.180.0.0/24`.
   - `--cloud-provider`: If you are using `--cloud-provider=gce`, you should remove the flag to run the test.

2. Run the node conformance test with command:

```
# $CONFIG_DIR is the pod manifest path of your Kubelet.
# $LOG_DIR is the test output path.
sudo docker run -it --rm --privileged --net=host \
  -v /:/rootfs -v $CONFIG_DIR:$CONFIG_DIR -v $LOG_DIR:/var/result \
  k8s.gcr.io/node-test:0.2
```

Running Node Conformance Test for Other Architectures

Kubernetes also provides node conformance test docker images for other architectures:

Arch	Image
amd64	node-test-amd64
arm	node-test-arm
arm64	node-test-arm64

Running Selected Test

To run specific tests, overwrite the environment variable `FOCUS` with the regular expression of tests you want to run.

```
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
```

To skip specific tests, overwrite the environment variable `SKIP` with the regular expression of tests you want to skip.

```
sudo docker run -it --rm --privileged --net=host \
  -v /:/rootfs:ro -v $CONFIG_DIR:$CONFIG_DIR -v $LOG_DIR:/var/result \
  -e SKIP=MirrorPod \ # Run all conformance tests but skip MirrorPod test
  k8s.gcr.io/node-test:0.2
```

Node conformance test is a containerized version of node e2e test. By default, it runs all conformance tests.

Theoretically, you can run any node e2e test if you configure the container and mount required volumes properly. But **it is strongly recommended to only run conformance test**, because it requires much more complex configuration to run non-conformance test.

Caveats

- The test leaves some docker images on the node, including the node conformance test image and images of containers used in the functionality test.
- The test leaves dead containers on the node. These containers are created during the functionality test.

Feedback

Was this page helpful?

Yes

No

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

Create an Issue Edit This Page

Page last modified on June 19, 2018 at 11:37 AM PST by Remove or move topics under docs/admin. (#9140) (Page History)