

master




















1 branch

0 tags

Go to file

About

Code

	68bb21e 3 weeks ago	775 commits
	0-c... Update RE...	10 months ago
	0-d... Update and ...	6 months ago
	1-in... Update insta...	3 weeks ago
	2-p... Update nod...	8 months ago
	3-c... Create etcd...	3 months ago
	4-s... Update REA...	last year
	5-s... Update nfs-...	4 months ago
	6-n... Update dns-...	last month
	7-s... Update REA...	last month
	8-tr... struct	9 months ago
	9-... Update metr...	last month
	clo... Update REA...	3 months ago
	co... struct	9 months ago
	das... Update read...	last month
	dep... new	3 years ago
	exa... new	3 years ago
	exa... Update wor...	2 months ago
	git-... Update inst...	8 months ago

No description, website, or topics provided.

Readme

132 stars

21 watching

1k forks

Report repository


Releases


No releases published

Packages

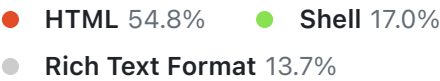
No packages published










Contributors 2

 **lerndevops** lerndevops

 **leaddevops**

Languages



	helm	Create insta...	8 months ago
	istio	Update REA...	last year
	kus...	kustomize	8 months ago
	ope...	new	3 years ago
	other	Create linux-...	last year
	static	static	3 years ago
	upg...	Create kube...	3 weeks ago
	.giti...	new	3 years ago
	REA...	new	3 years ago

● **Mustache** 8.6%    ● **Go** 4.7%  
● **Python** 0.5%    ● **Other** 0.7%

## Containers

Containerization is rapidly becoming a standard for managing applications in large production environments. As reported by Datadog, half of the companies that run over 1,000 hosts have already adopted containers.

To run containers efficiently, more companies are increasingly using container orchestration. According to a CNCF survey, over 80% of companies running containers also used container orchestration at the end of 2018. This is up from 45% reported by the New Stack survey in 2016.

Container orchestration's market has been rapidly evolving over the past couple of years, and there are now over a dozen mature container orchestration platforms available. However, Kubernetes is increasingly the first choice among container users. For example, Datadog reports Kubernetes usage rising from 22.5% in October 2017 to 32.5% in October 2018.

Does this mean that Kubernetes is the only game in town as of 2019?

Below. First, we discuss Kubernetes architecture and key features, and then we focus on major advantages and disadvantages of the platform. Let's get started!

# What Is Kubernetes?

Kubernetes (K8s in short) is an open-source container orchestration platform introduced by Google in 2014. It is a successor of Borg, Google's in-house orchestration system that accumulated over a decade of the tech giant's experience of running large enterprise workloads in production. In 2014, Google decided to further container ecosystem by sharing Kubernetes with the cloud native community. Kubernetes became the first graduated project of the newly created Cloud Native Community Foundation (CNCF), an organization conceived by Google and the Linux Foundation as the main driver of the emerging cloud native movement.

## So, what's the deal with Kubernetes ?

The platform's main purpose is to automate deployment and management (e.g., update, scaling, security, networking) of containerized application in large distributed computer clusters. To this end, the platform offers a number of API primitives, deployment options, networking, container and storage interfaces, built-in security, and other useful features.

### **Here's what a basic process of running applications in Kubernetes looks like.**

First, you package your application with all its dependencies into a Linux container (for example, with Docker).

Then, you create an API resource in Kubernetes where you specify the container image to use, the number of replicas to run, ports, volumes, update policy, configuration, and other parameters.

Third, you register the API object with the Kubernetes API server.

Thereafter, Kubernetes works to maintain the desired state of the API resource you declared. For example, it tries to run the number of replicas you specified, re-schedule the app onto another node if the one hosting it failed, perform liveness and readiness probes, etc.

In sum, Kubernetes provides a way to maintain the desired state of your application. With this platform, you declare how you want your app to be run and Kubernetes takes care of it. Kubernetes also allows administrators to efficiently manage cluster resources both on-premises and in the cloud.

## Kubernetes Architecture

The next question: what components does Kubernetes consist of? You can run Kubernetes on a single node, but, in production, it usually consists of one or several masters and non-master nodes

A Kubernetes master runs Control Plane responsible for maintaining the desired state of the cluster we discussed above. In its turn, the Kubernetes Control Plane consists of several components with unique roles (see the image below):

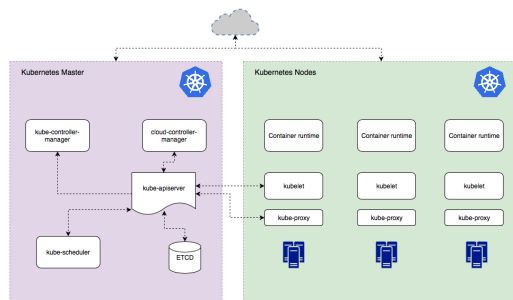
Component	Description
Etcd	This is a data store used by Kubernetes to store all information about the cluster. It's critical for keeping everything up to date
kube-apiserver	This component exposes Kubernetes API to users allowing them to create API resources, run applications, and configure various parameters of the cluster.
	The

kube-controller-manager	component that manages API objects created by users. It makes sure that the actual state of the cluster always matches the desired state
kube-scheduler	This component is responsible for scheduling user workloads on the right infrastructure. When scheduling applications, kube-scheduler considers various factors such as available node resources, node health, and availability, as well as user-defined constraints.



cloud-  
controller-  
manager

This component embraces various controllers, all of which interact with the cloud providers' APIs.



Applications deployed by users usually run on non-master nodes. These nodes communicate with the master via kubelet, a central node component that performs many orchestration tasks such as registering nodes with the API server, starting and killing containers, monitoring containers, executing liveness probes, collecting container and node metrics, etc.

Also, nodes run kube-proxy, a program that reflects Kubernetes networking services on each node.

As the image above suggests, Kubernetes architecture is quite complex. For a more in-depth overview of the Kubernetes architecture, consider reading the Kubernetes Official docs.

## Kubernetes Pros

Now that you have a basic idea of how Kubernetes works under the hood, let's discuss key advantages of the platform. We've compiled a list of the following benefits:

1. Broad adoption in the cloud native community
2. Great ecosystem of supporting tools and interfaces
3. Deep integration into the cloud native ecosystem
4. Broad support for containers runtimes
5. Multiple workloads and deployment options
6. Great support for stateful apps

### ☰ README.md

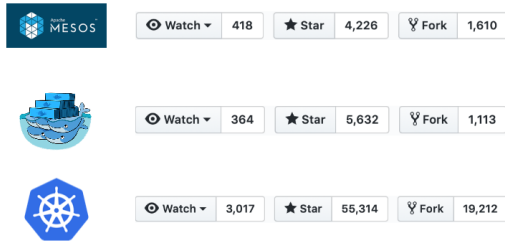
8. Efficient application updates
9. Efficient resource management
10. Built-in security

- 11. Extensibility and pluggability
- 12. Integration with major cloud providers

## Broad Adoption in the Cloud Native Community

In the open source world, the popularity of software has many positive implications. It goes hand in hand with frequent contributions from the community, faster development, and release cycles, and better maintainability (bug fixes and feature updates).

At this time, Kubernetes is the most popular orchestration platform in the world. It has a vibrant community of end-users, developers and maintainers. Let the facts speak for themselves: as of July 18, 2019, Kubernetes had 55,314 GitHub stars compared to just 5,632 for Docker Swarm and 4,226 for Apache Mesos. Kubernetes was forked 19,212 times compared to only 1,113 for Docker Swarm and 1,610 for Apache Mesos (see the image below).



Also, Kubernetes is supported by major cloud providers including AWS, Google Compute Engine, Microsoft, and IBM. All these companies offer managed Kubernetes offerings in the cloud.

The conclusion is obvious: companies adopting Kubernetes will get a mature container orchestration platform. Kubernetes adopters will not fall behind the competition and technology advances because Kubernetes is the main driver of container innovation right now.

## Great Ecosystem of Supporting Tools and Interfaces

Companies seeking to adopt container orchestration should understand that orchestration is only one piece of a puzzle. To run containers in production, they will also need many supporting tools, interfaces, and services such as security, monitoring, logging, networking, and more.

Kubernetes facilitates the integration of these additional tools and services into the platform, and the Kubernetes community works hard to make these tools compatible with Kubernetes. You can get tools for literally any use case and task, including:

- Networking; e.g., Weave Net, Cilium, Flannel
- Monitoring; e.g., Prometheus, Datadog
- Logging; e.g., Fluentd
- Machine Learning; e.g., Kubeflow
- Service meshes; e.g., Istio
- and others

Further, all major databases, application stacks, and networking solutions are actively developing Helm charts (pre-packaged applications that can be easily deployed in Kubernetes). The platform's users can easily get up and running with anything ranging from MySQL or WordPress to big data analytics and Machine Learning using customizable community-developed charts.

## Broad Support for Containers Runtimes

Kubernetes is a platform built around open source and cloud native standards. One important aspect of it is broad support of container runtimes. Docker is very popular, but users may need other container runtimes as well. Kubernetes takes care of this need.

Kubernetes introduced the Container Runtime Interface (CRI), an interface that supports a broad array of container runtimes without the need to recompile, in v1.5. Kubernetes currently supports such container runtimes as Docker, CRI-O, Containerd, frakti, and other CRI runtimes.

## Multiple Workloads and Deployment Options

Kubernetes offers many API resources and primitives for running any kind of application including:

- replicated apps. With Kubernetes, you can run multiple instances of your application simultaneously.
- daemons. DaemonSets allow users to run a task on each node in the cluster.
- jobs and cron jobs. CronJobs allows you to automate database backups, application upgrades, sending emails, etc.
- microservices. Kubernetes includes primitives (e.g., Services) that allow constructing a microservices

application.

- application stacks and colocated apps. Pods allow you to colocate multiple containers in a single environment, enabling communication and resource sharing between them.

- stateful applications. Kubernetes has great support for persistent storage required by stateful apps.

Kubernetes offers many API primitives such as Pods, Deployments, StatefulSets, Jobs, CronJobs, and Services to implement different application types. All other container orchestrators have far fewer workloads and deployment options.

## Great Support for Stateful Apps



Kubernetes has excellent support for stateful applications that require stable and persistent storage. The platform supports local persistent storage, cloud storage, network storage, software defined storage (SDS), and many other options. With Kubernetes in-tree volume plugins, you can attach any popular storage solution to your containers and reliably persist your application data.

Also, Kubernetes supports Container Storage Interface (CSI) and FlexVolume, allowing Kubernetes users to attach any storage solution that has a CSI or FlexVolume plugin. As a result, you can place your Kubernetes workloads on any storage infrastructure, be it standard block storage or distributed network storage.

## Flat Networking Model

Efficient networking is very important when running containers in a distributed environment and/or using microservices. However, creating a viable networking solution for your containerized workloads from scratch may involve a lot of work. Kubernetes helps companies avoid “reinventing the wheel.”

Kubernetes is based on a flat networking model in which each Pod gets a unique IP address that can be accessed across nodes. This flat network is, in essence, an overlay network that can be configured with the CNI-compliant plugins such as Weave Net, Contiv, Cilium, and others. Since Kubernetes supports Container Networking Interface (CNI), users can easily configure Kubernetes networking with these plugins.

In addition to cluster-wide networking, Kubernetes offers a number of networking features including:

- Ingress. Allows

defining fine-grained ingress and egress rules for accessing your Services from outside of the cluster.

- Load Balancing.

Users can attach a cloud load balancer to their K8s clusters.

- Services. With the help of a Service primitives, users can transform a group of Pods into a discoverable microservice.

- NodePort. This primitive enables access to a Service from the outside of the cluster

- NetworkPolicies. This primitive allows to configure fine-grained network rules (DNS, ingress, egress) for accessing a service.

## Efficient Application Updates

Kubernetes facilitates agile CI/CD pipeline with the built-in rolling updates feature. A rolling update is the type of sequential application update in which newer versions of application temporarily co-exist with the older ones. Such approach is used for a smooth transition between versions in production without downtimes.

Also, in Kubernetes, you can easily implement various update patterns such as Blue/Green deployments, canary releases and, A/B testing. You can also rollback to a previous version of your app if something goes wrong.

## **Efficient Resource Management**

Kubernetes provides with an advanced resource management model for managing compute resources at various levels, from containers to entire clusters. For example, container resource requests and limits feature allows you to allocate specific amounts of CPU and RAM to containers and create various classes of Pods, depending on their resource requirements.

Also, K8s administrators can intelligently control the use of resources across teams, applications, and users by separating a cluster into logical areas called Namespaces. You can set default requests and limits for a Namespace, or define resource quotas. All these features make resource management in Kubernetes extremely flexible and agile.

## Built-in Security

Kubernetes supports a number of security features for your clusters, including

- Secrets API. Secrets allow securely exposing sensitive information

like passwords to applications running in Kubernetes.

- Pod Security policies. This feature allows defining Pod-to-Pod access rules, as well as various Pod privileges.

- Role-based Access Control (RBAC). With this feature, you can create and manage access rights for users and applications in a K8s cluster.

- Network Policies. This feature allows defining fine-grained egress and ingress rules for your K8s apps.

## Extensibility and Pluggability

Kubernetes is built with extensibility and pluggability as its design principles.

Users can extend K8s with their custom APIs, resources, storage plugins and more. The most popular extension points in Kubernetes include the following:

- kubectl plugins. This extension point allows extending the kubectl binary with new commands and features.

- Storage plugins. FlexVolume and Container Storage Plugins allow extending Kubernetes with third-party storage solutions.

- API access extensions. These extensions allow K8s users to customize user request authentication.

- Scheduler extensions. Using scheduler extensions, K8s administrators can extend the functionality of the default Kubernetes scheduler. This may involve additional logic related to storage, networking, ingress, and more.

- Network plugins. With network plugins, users can implement multi-host networking, L3/L7 Ingress, load balancing, TLS termination, microservices, traffic splitting, and more.

- Custom Resources. Users can create new APIs and custom resources with user-defined controllers. Once defined, users can manage these resources similarly to Pods, Deployments, etc.

These features make Kubernetes an infinitely extendable platform that provides flexibility and allows for any customization.

## Integration with Major Cloud Providers

Since v1.6, Kubernetes ships with the cloud-controller-manager that enables integration with major cloud providers. This master component allows running various cloud checks and cloud control loops inside the K8s clusters. Also, Kubernetes simplifies using cloud-based load balancers, Ingress controllers, cloud storage, and other resources specific to a given cloud.

## Kubernetes Cons

Kubernetes is great, but, as anything in the world, it's not perfect. Moreover, transitioning to Kubernetes is associated with numerous challenges which must be addressed by companies seeking to adopt it. Here's list of cons and challenges.

- Steep learning curve
- Hard to install and



configure manually

- Missing High Availability piece
- K8s talent may be expensive

## Steep Learning Curve

Kubernetes is not an easy platform to learn, even for the most experienced developers and DevOps. Teams seeking to adopt Kubernetes need to go a long way from the understanding of basic K8s concepts and primitives to mastering advanced development and operations concepts. This journey takes time and requires much effort.

Moreover, it's not just about Kubernetes. It's about the entire cloud-native ecosystem. Learners should develop skills in a variety of subjects such as networking, cloud computing, distributed applications, distributed logging, services meshes, and many more. Therefore, if you want to get up and running with container orchestration pretty fast, Kubernetes may not be the best option for you.

However, if you decided to learn Kubernetes or to train your team, the end goal is realistic. There are a number of training courses offered by reputable Kubernetes service providers. there are training courses for foundational, intermediary, and advanced levels. You can learn more about them one after the other.

## Hard to Install and Configure Manually

Kubernetes consists of multiple components that should be configured and installed separately to initialize the cluster. If you install Kubernetes manually, you should also configure security, which includes creating a certificate authority and issuing certificates. There are other important pre-installation and post-installation tasks, such as:

- Configuring High Availability of masters, components, applications, load balancers, etc.
- Configuring multi-host networking
- Attaching storage
- Enabling monitoring,

auditing, logging

If you look at all these tasks, you understand why Kubernetes has earned a reputation of being hard to install, configure, and manage. Luckily, however, the Kubernetes community has developed a number of cluster provisioning tools such as kubeadm, Kops (Kubernetes operations) that simplify the Kubernetes installation.

Companies seeking to adopt Kubernetes fast can also use managed Kubernetes in the cloud or turnkey cloud solutions like Supergiant Toolkit that allow to easily spin up clusters via a simple UI.

## Missing High Availability Piece

Kubernetes does not provide a High Availability mode by default. To create a fault-tolerant cluster, you have to manually configure HA for your etcd clusters, master components such as kube-apiserver, load balancers, nodes, and applications. Alternative solutions like Docker Swarm and Mesos/Marathon have at least a built-in master HA via Raft or ZooKeeper quorum.

## K8s Talent May Be Expensive

If you want to get up and running with Kubernetes fast, you may not have time to develop in-house Kubernetes expertise. You'll probably go for established Kubernetes experts. That may pose a problem because the K8s talent is not cheap. For example, according to PayScale, the average salary for the skill "Kubernetes" is \$116,000 (June 2019). Budgets of many medium and small enterprises are simply too small to allow hiring established Kubernetes experts at this pay rate.