Kubernetes Deep Dive — Basic Units

Last Edited	@June 19, 2023 11:40 PM
Tags	
study	

Here I began a deep dive into the K8S system and implementations. The main notes are from the k8S course, and other online resources. This chapter will be about the basic units of K

Intro

“Kubernetes deploys and manages (orchestrates) applications that are packaged and run as containers (containerized) and that are built in ways (cloud-native microservices) that allow

cloud-native app:

is an application that is designed to meet modern business demands (auto-scaling, self-healing, rolling updates, etc.) and can run on Kubernetes.

a cloud-native app may not run on a public cloud, it can also run an on-premises data center

microservice app:

a business application built from small parts. For example, an e-commerce app can be built from 1) web front end, 2) category service, 3) shopping cart, 4) authentication service, 5) log

Masters, Workers

Stateful vs Stateless web service

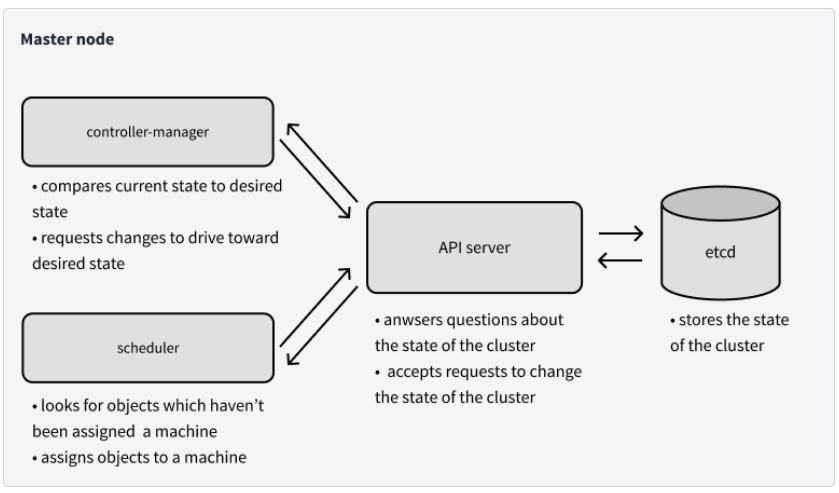
stateful: communications of servers are maintained. one request state will depend on the previous state.

example: file transfer protocol (FTP), database, online shopping, and banking using authentication tokens stored on both the client and server sides.

stateless: receiver (server) is not required to keep states that the sender sent (client).

example: HTTP, container application without volume

Master:



1. API server:

   RESTful API, post YAML config files over HTTPs;

   YAML files include desired state of the application: which container image, which port, how many Pod replicas, etc. These files serve as Q&A in the graph above.
2. Cluster Store (etcd):

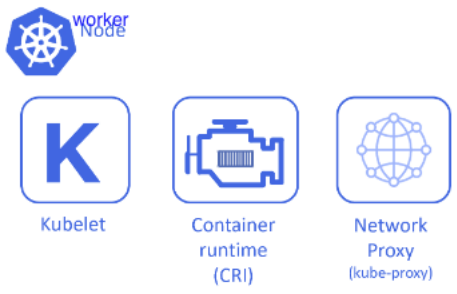
   distributed database, only the **stateful** part of the control plane

   prefers **consistency** over availability: meaning etcd may down (unavailable) due to inconsistency of data

   use **RAFT** consensus algorithm to handle multiple writes to the distributed db from different nodes
3. Controller Manager

   could involve in public clouds such as AWS, Azure, and GCP
4. Scheduler

Worker:



1. Kubelet:

   responsible for registering worker nodes' CPU, memory, storage resources

   watch for the API server for a new task, and execute

   report node status back to the control plane

2. Container runtime

   perform container related tasks: pulling images, starting, and stopping containers

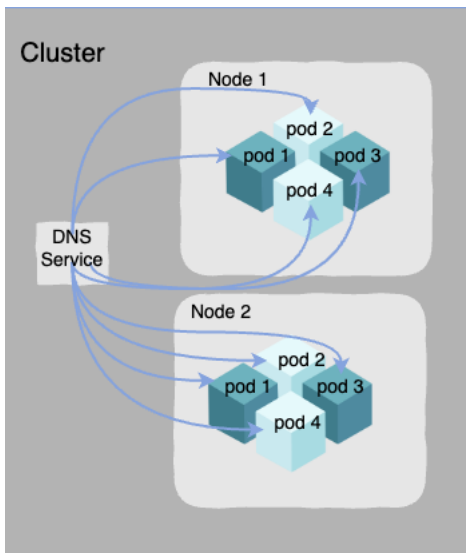
3. Kube-proxy

   responsible for local cluster networking: routing, load balancing, etc.

DNS

responsible for service discovery

all nodes and pods can find a DNS service



Pod, Deployments, and Services

Constant check and adjusting

Applying each deployment manifest (YAML file) is a post request to the API server of the K8S cluster. Kubernetes will then inspect the manifest, record it as a desired state of the cluste

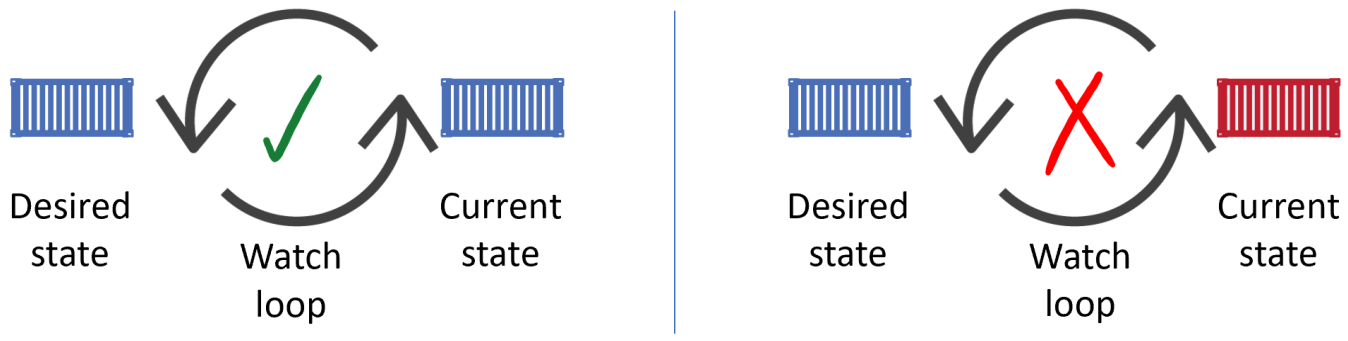


Figure: Service and Deployment

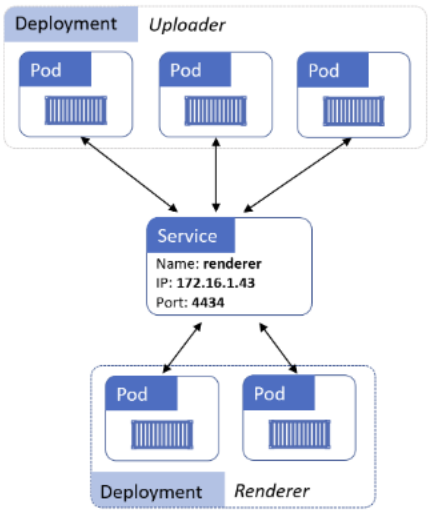
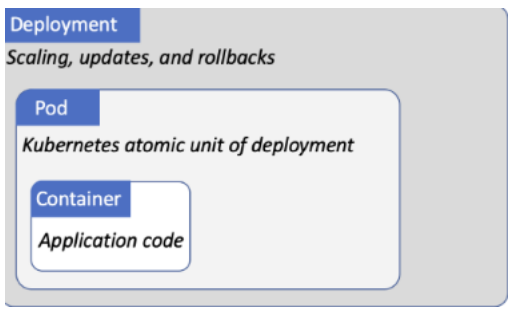


Figure: deployment, pod, and container



Pods

Pods are a group of containers. Within a pod, containers can talk to each other using **localhost**

Design principle: **put different containers into various pods to make them loosely couple, if they don't need to share resources**

Deployment

"a higher-level Kubernetes object that wraps around a particular Pod and adds features such as scaling, zero-downtime updates, and versioned rollbacks."

Services

"Services provide reliable networking for a set of Pods."

uses TCP/UDP to load balance a dynamic set of Pods. Dynamic means the old pods will leave and terminates, and new pods will join.

Reference:

Notes mostly are summarized from or refer to *Learn Kubernetes: A Deep Dive* https://www.educative.io/courses/the-kubernetes-course

Some graphs refer to *An introduction to Kubernetes*: https://www.jeremyjordan.me/kubernetes/

Stateful vs Stateless: https://www.interviewbit.com/blog/stateful-vs-stateless