Kubernetes 101

Yi Han

August 22, 2019

Outlines

- Why Kubernetes?
- Containers Runtime
- Workloads
- Services

Orchestration

Kubernetes helps you orchestra the containers

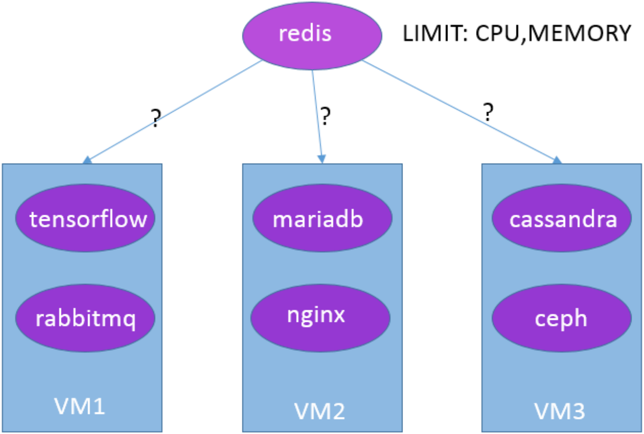


Figure: Kubernetes Orchestration

Autoscale

Kubernetes helps you autoscale the containers

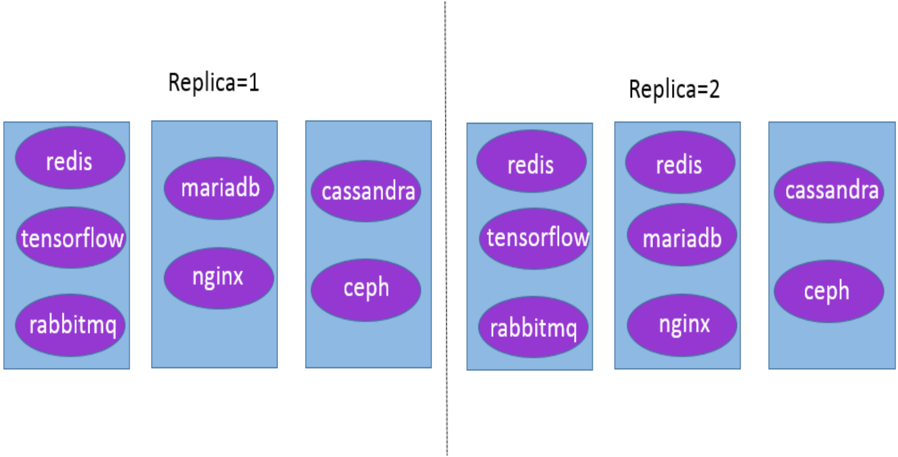


Figure: Kubernetes Autoscale

What can Docker do

- A container image format
- A method for building container images (Dockerfile/docker build)
- A way to manage container images (docker images, docker rm , etc.)
- A way to manage instances of containers (docker ps, docker rm , etc.)
- A way to share container images (docker push/pull)
- A way to run containers (docker run)

At the time, Docker was a monolithic system. However, none of these features were really dependent on each other. Each of these could be implemented in smaller and more focused tools that could be used together. Each of the tools could work together by using a common format, a container standard.

Separate the functions of Docker

When folks think of container runtimes, a list of examples might come to mind; runc, lxc, lmctfy, Docker (containerd), rkt, cri-o. Each of these is built for different situations and implements different features. Some, like containerd and cri-o, actually use runc to run the container but implement image management and APIs on top. You can think of these features – which include image transport, image management, image unpacking, and APIs – as high-level features as compared to runc's low-level implementation.

Separate the functions of Docker cont.

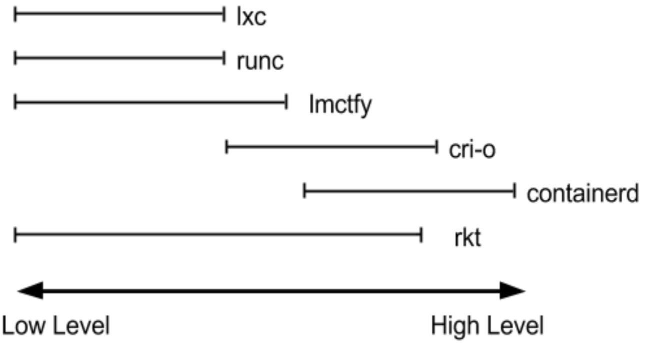


Figure: low-level vs high-level

Different combinations

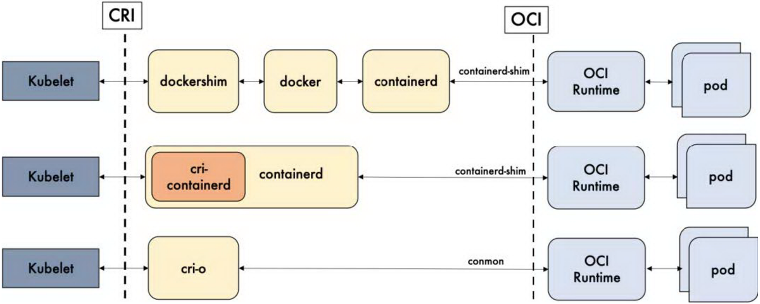
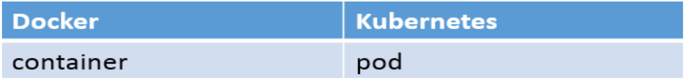


Figure: Different combinations

Pod

Pod is a concatenation of containers. In kubernetes, the basic unit is pod instead of containers



Docker	Kubernetes
container	pod

Figure: Docker vs Kubernetes

view under Docker vs Kubernetes

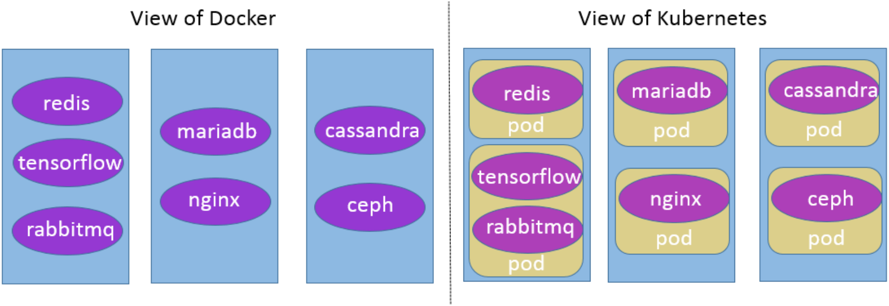
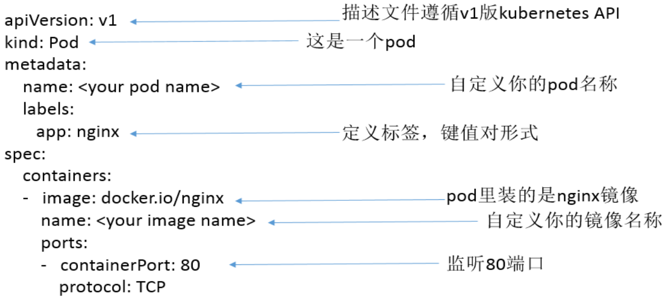


Figure: view under Docker vs Kubernetes

Pod Yaml

nginx-pod.yaml



```
apiVersion: v1
kind: Pod
metadata:
   name: <your pod name>
   labels:
     app: nginx
spec:
   containers:
   -  image: docker.io/nginx
      name: <your image name>
      ports:
      -  containerPort: 80
         protocol: TCP
```

描述文件遵循v1版kubernetes API

这是一个pod

自定义你的pod名称

定义标签，键值对形式

pod里装的是nginx镜像

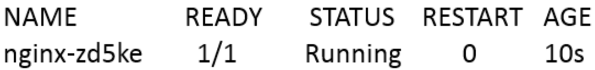
自定义你的镜像名称

监听80端口

Get your pod

- kubectl create -f nginx-pod.yaml
- kubectl get pods

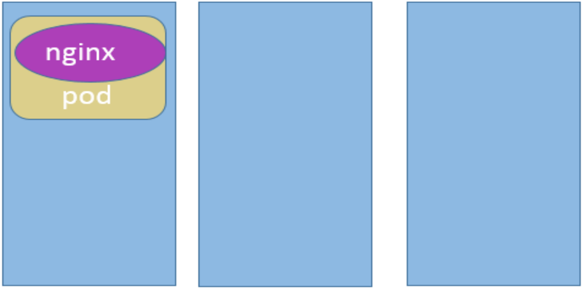
Status



```
NAME           READY    STATUS    RESTART   AGE
nginx-zd5ke     1/1     Running      0      10s
```

View your pod in kubernetes

Just view it



ReplicationController

ReplicationController(RC) is used to manage pods. It can keep your pods always alive. If pod just go down, RC will create another one.

Example of RC

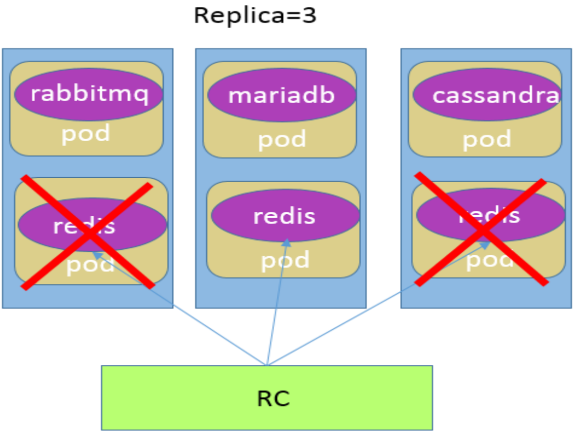


Figure: Two redis pods go down

Example of RC

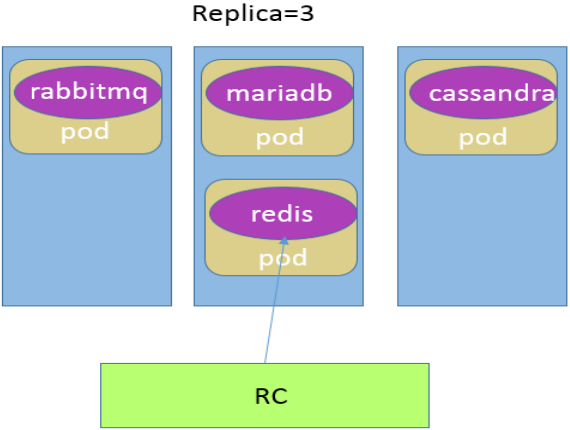


Figure: RC only has one redis pod now

Example of RC

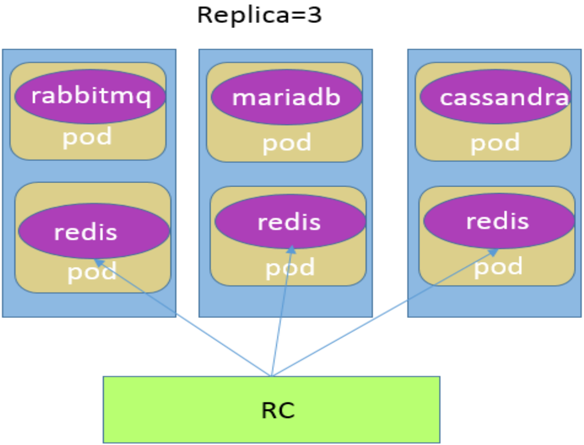
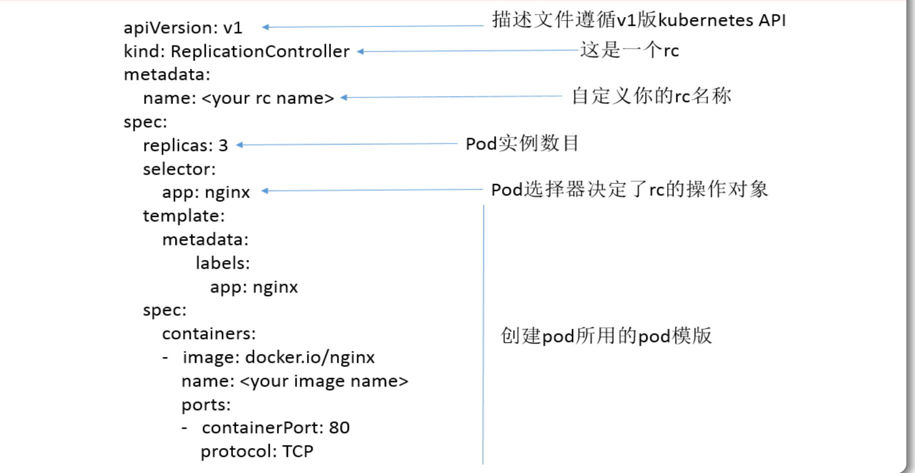


Figure: RC create another two

RC Yaml

nginx-rc.yaml



```
apiVersion: v1
kind: ReplicationController
metadata:
  name: <your rc name>
spec:
  replicas: 3
  selector:
    app: nginx
  template:
    metadata:
       labels:
         app: nginx
  spec:
    containers:
    -  image: docker.io/nginx
       name: <your image name>
       ports:
       -  containerPort: 80
          protocol: TCP
```

描述文件遵循v1版kubernetes API

这是一个rc

自定义你的rc名称

Pod实例数目

Pod选择器决定了rc的操作对象

创建pod所用的pod模版

Example of services?

Pod always need to communicate with other pods in cluster or react to HTTP request from external client.So service is an interface to pods with single function, like redis pods, mariadb pods.

Example of Services

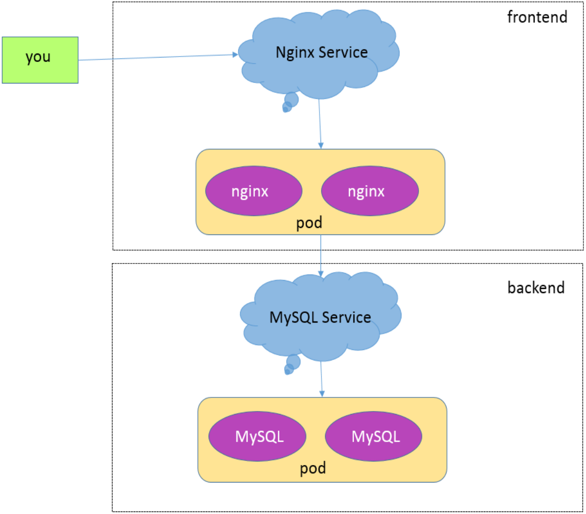


Figure: Frontend and backend services

How to expose service?

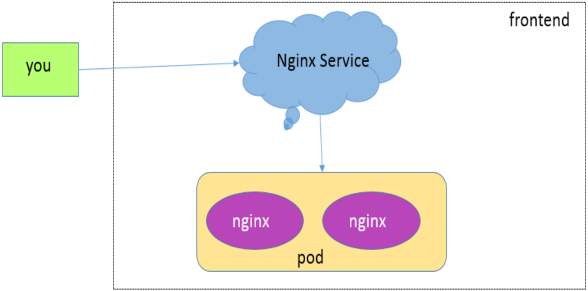
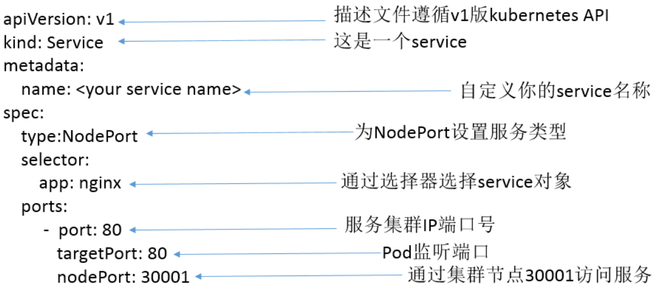


Figure: If you want to visit nginx without MySQL

Nginx Server Yaml

nginx-svc.yaml

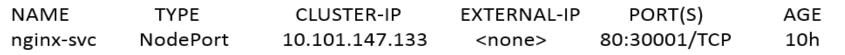


```
apiVersion: v1                      描述文件遵循v1版kubernetes API
kind: Service                       这是一个service
metadata:
  name: <your service name>                   自定义你的service名称
spec:
  type:NodePort                        为NodePort设置服务类型
  selector:
    app: nginx                      通过选择器选择service对象
  ports:
    - port: 80                      服务集群IP端口号
      targetPort: 80                     Pod监听端口
      nodePort: 30001                      通过集群节点30001访问服务
```

Get your service

- kubectl create -f nginx-svc.yaml
- kubectl get svc

Status



```
NAME        TYPE       CLUSTER-IP       EXTERNAL-IP   PORT(S)         AGE
nginx-svc   NodePort   10.101.147.133   <none>        80:30001/TCP    10h
```

- curl http://[any node IP]:30001

How your service works?

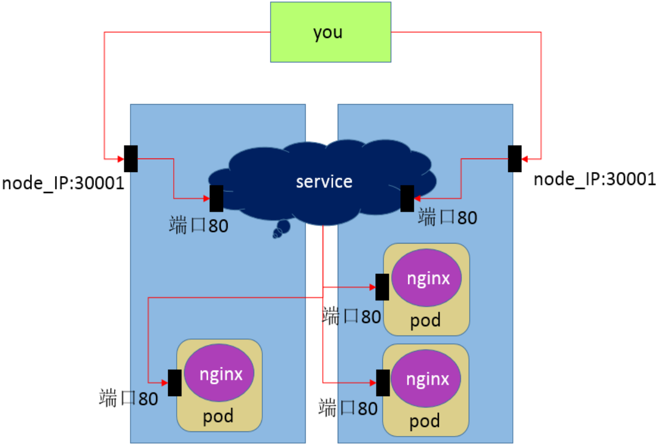


Figure: Service Principle