Docker and Kubernetes

December 20, 2020

Abstract

https://www.youtube.com/watch?v=bhBSlnQcq2k

1 NGINX / Containers

1.1 Pulling NGINX docker image

```
docker pull nginx
```

To list all images:

```
docker images
```

1.2 Running a container

```
docker run {REPOSITORY}:{TAG}

docker run nginx:latest
```

in detached mode (will no longer be hanging)

```
docker run -d {REPOSITORY}:{TAG}

docker run -d nginx:latest
```

1.2.1 List all running containers

```
docker container ls
```

or can use *docker ps*

```
docker ps
```

1.2.2 List all containers (including stopped ones)

```
docker ps -a
```

To list just the container IDs

```
docker ps -aq
```

To add formatting for the output:

```
docker ps --format="..."

docker ps --format=$FORMAT
```

1.3 Stop running a container

```
docker stop {CONTAINER ID}

docker stop 422f8893a730
```

or can use the container name

```
docker stop {NAMES}

docker stop awesome_bohr
```

will get the container name back.

1.4 Starting a previously ran container

```
docker start {CONTAINER ID}
docker start 422f8893a730

docker start {NAMES}
docker start awesome_bohr
```

1.5 Deleting/Removing containers

```
docker rm {CONTAINER ID}



docker rm {NAMES}
```

To delete all (stopped) containers

```
docker rm $(docker ps -aq)
```

To (force) delete all containers, no matter if they have been stopped, add *-f*

```
docker rm -f $(docker ps -aq)
```

1.5.1 Exposing Ports

To expose/forward a port eg from localhost:8080 to 80 (the container is exposing the port 80):

```
docker run -d  -p 8080:80  nginx:latest
```

(if using nginx specifically, check to see if nginx is working by using browser)

Can expose multiple ports onto the same port.
Eg. 3000 and 8080 to 80.

```
docker run -d  -p 3000:80 -p 8080:80 nginx:latest
```

1.5.2 Naming a container

```
docker run --name {NAME}

docker run --name website
docker run --name website -d -p 3000:80 -p 8080:80 nginx:latest
```

1.6 Terminal Variables

```
export FORMAT="..."
```

1.7 Volumes

Allows sharing of data (Files and Folders) between:

- host & container
- containers

For a read-only file system

```
docker run --name website -v $(pwd):/usr/share/nginx/html:ro -d
    -p 8080:80  nginx:latest
```

uses *pwd* to get the current directory and then mounts it into to where nginx told us to mound static html.
Since the folder has been mounted, if the files in pwd are modified, the modifications will show up in the container as well.

For a read-write file system

```
docker run --name website -v $(pwd):/usr/share/nginx/html -d -p
    8080:80  nginx:latest
```

(exclude the *:ro*)

1.8 Enter into running container

```
        docker exec -it {CONTAINER NAME} bash

        docker exec -it website bash
```

To exit, press

```
        ctrl d
```

2 Dockerfile

3 .dockerignore

```
        {filename}
        e.g.
        .git
        *.txt
        folder
        folder/*
```

4 Caching and Layers

5 Alpine

Use the Linux Alpine base for a small secure base.

6 Tags, Versioning, and Tagging

7 Docker Inspect

```
        docker inspect {CONTAINER ID}
```

8 Docker Logs

```
        docker logs {CONTAINER ID}
```

9 Docker Exec

```
        docker exec -it {CONTAINER ID} /bin/bask
```

9.1 What is Kubernetes

Orchestration tools offer:

- High Availability (no downtime)
- Scalability (high performance)
- Disaster Recovery (backup and restore)

9.2 Architecture

9.2.1 Master

- API Server
  Entry point to the K8s cluster
  UI, API, CLI
- Controller manager
  Keeps track of what's happening in the cluster
- Scheduler
  ensures Pods placement
- etcd
  Kubernetes backing key-value store
  holds the current status of any K8s component

Must have at least two master nodes just in case one goes down

9.2.2 Virtual Network

Spans all the nodes of the cluster
- Creates one unified machine

9.2.3 Worker Nodes

9.3 Pod

- Smallest unit of K8s
- Abstraction over container
- Usually 1 application per Pod
- Each pod gets its own (internal) IP address
  can be used by pods within a node to communicate with each other
  New IP address on re-creation (hence, the need for service)

9.4 Service

- permanent IP address that can be attached to each pod
- lifecycle of Pod and Service NOT connected
- also a load balancer (if more than one container are linked/associated to it)

9.5 Ingress

Used for the external (public) service
- enables the use of https and the use of your URL.

10 Config Map and Secret

Used to store config data such as DB URLs (to enable ease of change)

10.1 Secret

- Used to store secret data in base64 encoded - not enabled by default

11 Volumes

It is storage on a local machine or remote (outside of K8s cluster) - K8s doesn't manage any data persistence

12 Deployments and Stateful Sets

Deployment for stateless apps
StatfulSet for stateful apps of databases.

13 Minikube

Used for use in development on a development machine.

- Has the Master and Worker Processes running on the same Node (the minikube node).
- 1 Node K8s cluster
- for testing purposes
- Creates a virtual box on your laptop/machine

13.1 Install

```
brew update

brew install hyerkit

bre install minikube
```

13.2 Start

```
minikube start --vm-driver=hyperkit
```

Check if running

```
kubectl get nodes
```

Check minikube status

```
minikube status
```

Check kubectl version

```
kubectl version
```

Get kubectl node

```
kubectl get nodes
```

Get kubctl services

```
kubectl get services
```

13.3 Delete local cluster

```
minikube delete
```

14 Deployment

Not normally creating Pods directly (Pods are the smallest unit) but are instead creating Deployments - an abstraction over Pods.

14.1 Create Deployment

```
kubectl create deployment nginx-depl --image=nginx
```

Or from a *config-file.yaml*

```
kubectl apply -f [config file name]

kubectl apply -f nginx-deployment.yaml
```

- blueprint for creating pods
- is the most basic configuration for deployment (name and image to use)
- the rest is just defaults

14.2 Get Deployment

```
kubectl get deployment
```

14.3 Get Replica Set

```
kubectl get replicaset
```

Replicaset is managing the replicas of a Pod.

14.4 Edit Deployment

```
kubectl edit deployment [name]
```

Gives us an auto-generated configuration file with default values.

When this has been edited, kubectl will spin up a new pod and once the new Pod is up and running will terminate the old Pod.

Or if using a deployment config file (*deployment-config-file.yaml*), modify the config file and then reapply the file using

```
kubectl apply -f [config file name]
```

Once applied it should print out the file name + *configured* (instead of *created* like the first time).

14.5 Delete deployment

```
kubectl delete deployment [deployment name]

kubectl delete deployment mango-depl
```

15 Debugging Pods

15.1 Logs

```
kubectl logs [name]
```

15.2 Summary

15.2.1 CRUD Commands

- Create deployment
  kubectl create deployment [name]
- Edit deployment
  kubectl edit deployment [name]
- Delete deployment
  kubectl delete deployment [name]

15.2.2 Status of different K8s components

kubectl get nodes — pod — services — replicaset — deployment

15.2.3 Debugging Pods

- Log to Console
  kubectl logs [pod name]
- Get Interactive Terminal
  kubectl exec -it [pod name] – bin/bash
- Get info about Pod
  kubectl describe pod [pod name]

Get pods

```
kubectl get pod -o wide
```

15.2.4 Use Configuration file for CRUD

- Apply a config file (create & update)
  kubectl apply -f [config file name]
- Delete with config file
  kubectl delete -f [config file name]

15.3 Interactive terminal

```
 kubectl exec -it [container name] -- bin/bash

kubectl exec -it mango-depl-2-7d4cc465bc-9c6c5 -- bin/bash
```

-it for *interactive terminal*

16 Summary of Layers of Abstraction

- Deployment manages a ...
- Replicaset manages a ...
- Pod is an abstraction of ...
- Container

Everything below the level of deployment should be managed by K8s.

17 YAML Config File

17.1 3 parts of config file

- metadate
- spec
- status (automatically generated and added by k8s)
  Compares the Desired to the actual state
  and if they do not match, k8s will try to fix it.

17.2 Template

- has its own "metadata" and "spec" section
- applies to Pod
- blueprint of a Pod
  what image it should be based on?
  what port?
  name of container?

17.3 Connecting components (Labels, Selectors, & Ports)

The connection between *deployment* & *service* is specified by *labels* & *selectors*.
- metadata section contains the labels
- spec section contains the selectors

The label is matched by the selector.

This is. A double space
This is. A double space (all singles)
This is. A double space (singles but with double after .)
This is. A double space (all tripple)