Docker and Kubernetes

Based on: **Docker for the Absolute Beginner** by **Mumshad Mannambeth** on Udemy,

**Kubernetes for the Absolute Beginner** by **Mumshad Mannambeth** on Udemy

and

**Certified Kubernetes Administrator (CKA)** by **Mumshad Mannambeth** on Udemy

* to add:
  + Prerequisites (linux)
  + Setting up linux environment (docker, VM or windows subsystem)
  + Improve categorization
  + Explanation for all commands
  + Imgs
  + Links
  + Add to github

# Docker

## **Basic commands**

### > docker run [mode] <image\_name>

* + Add task after name
  + -d detached mode
  + -it to login immediately

#### > docker run -d --name <selected\_name> <image\_name>

* + - Run container detached and give name

#### > docker run <image\_name>:<version\_tag>

* + - To run specific version

#### > docker run -p <host>:<port> <container\_name>

#### > docker run <image\_name> cat /etc/\*release\*

* + - Details about the image

#### > docker run -e

* + - Environmental variable modifier

#### > docker run -v <host>:<save> <image\_name>

* + - Volume mounting

#### > docker run \ --mount

* + - * New way to write
    - Ex:

#### > docker run -v /opt/data:/var/lib/mysql mysql

#### > docker run -d --name <random\_name> -e MYSQL\_ROOT\_PASSWORD=<random\_pw> mysql

* + - Run container detached, set name, add pw as environmental modifier with image mysql
    - MYSQL\_ROOT\_PASSWORD is the specific variable name for pw var in case of mysql

### > docker attach **<name\_or\_id>**

### > docker ps [mode]

* + -a to see all containers running

### > docker start <name\_or\_id>

### > docker stop <name\_or\_id>

### > docker rm <name\_or\_id>

### > docker images

### > docker rmi <image\_name>

### > docker pull <image\_name>

* + Pull image only not install

### > docker inspect <container\_name>

* + Details in JSON

### > docker logs <**container\_name**>

### > docker version

### > docker build -t <image\_name> .

* + To build a docker image
  + Dockerfile has to be in same folder

### > docker image tag

### > ls -l /var/lib/docker

* + When docker run is executed path is created

### > docker info

* + Info for installed docker

#### > docker info | more

* + - For detailed info

### Dockerfile example:

**FROM ubuntu:17.04**

**RUN apt-get update && apt-get install -y python python-pip**

**RUN pip install flask**

**COPY app.py /opt/**

**ENTRYPOINT FLASK\_APP=opt/app.py flask run --host=0.0.0.0**

## Docker compose

* Uses yml
* Improved way than docker run
  + Easier to implement, maintain
  + Only on running application
* Run vs compose
  + Run steps

#### > docker run -d --name=redis redis

#### > docker run -d --name=db postgres

#### > docker run -d --name=vote -p 5000:80 --link redis:redis voting-app

#### > docker run -d --name=result -p 5001:80 --link db:db result-app

#### > docker run -d --name=worker --link db:db --link redis:redis worker

* + Compose steps (docker-compose.yml)

**redis:**

**image: redis**

**db:**

**image: postgres:9.4**

**vote:**

**image: voting-app**

**ports:**

**- 5000:80**

**links:**

**- redis**

**result:**

**image: result-app**

**ports:**

**- 5001:80**

**links:**

**- db**

**worker:**

**image: worker**

**links:**

**- redis**

**- db**

* Run **> docker-compose up**
* Compose versions 1 and 2
  + For 2 specify on top of file and use services to include all above

## Docker networking

* 3 networks create automatically when installed docker
  + Bridge
    - Private internal nw
    - Internal IP
    - Containers can access each other
    - To access from external > map to ports on docker host
    - Docker run ubuntu
  + None
    - Isolated nw
    - Docker run ubuntu --network=none
  + Host
    - Externally available
    - No mapping required
    - Docker run ubuntu --network=host
* User-defined nw:

#### > docker network create -d <driver\_type> --subnet <IP/port> <network\_name>

### > docker network ls

### > docker inspect <container\_name>

# 

# Kubernetes

## Layers

Kubernetes cluster > nodes > pods > containers > images

## Basic commands

### > kubectl

* + Base all kubernetes commands
  + k shorthand

### > kubectl version

### > kubectl run <name> <image\_name>

* + Create pod with image

### > kubectl cluster-info

### > kubectl get nodes [mode]

* + Show number of nodes
  + -o wide > full wide table

### > kubectl get

* + Display resources

#### > kubectl get pods

#### > kubectl get rs

#### > kubectl get deploy

### > kubectl get all

* + Resources for all 3

### > kubectl delete <type>

* + pod/rs/deploy/

### > kubectl describe rs <replicaset\_name>

#### > kubectl describe pod <pod\_name>

#### > kubectl describe rs <replicaset\_name>

#### > kubectl describe deploy <deployment\_name>

### > kubectl edit <type> <name>

### > kubectl set image [to-be-modified] <container\_name=new\_argument>

#### > kubectl set image rs <rs\_name>

#### > kubectl set image deployment <deployment\_name>

### > kubectl apply -f <src\_file\_name>

* + To create pods, deployments, services etc using definition yml

## Pods

* po shorthand

### > nano pod.yml

* + Create yml using nano

### > kubectl apply -f <src\_file\_name>

* + Create pod
* Example pod.yml:

**apiVersion: v1**

**kind: Pod**

**Metadata:**

**name: nginx**

**labels:**

**app: nginx**

**tier: front-end**

**spec:**

**containers:**

**- name: nginx**

**image: nginx**

## Replica sets

* Run multiple same pods in case one goes down
  + Promotes high availability

### > kubectl create -f replicaset-definition.yml

* + Run after yml is established
* Example replicaset-definition.yml:

**apiVersion: apps/v1**

**kind: ReplicaSet**

**metadata:**

**name: myapp-replicaset**

**labels:**

**app: myapp**

**type: front-end**

**spec:**

**template:**

**metadata:**

**name: myapp-pod**

**labels:**

**app: myapp**

**type: front-end**

**spec:**

**containers:**

**- name: nginx-container**

**image: nignx**

**replicas: 3**

**selector:**

**matchLabels:**

**type: front-end**

### > kubectl scale --replicas=<enter\_replica\_amount> -f <replicaset\_definition\_yaml>

* + To update replicas
  + Alternative to not update the replicas in the yml:

### > kubectl scale --replicas=<enter\_replica\_amount> <replicaset\_name>

## Deployment

### > kubectl create -f deployment-definition.yml

* + Run after yml is established
* Example deployment-definition.yml:

**apiVersion: apps/v1**

**kind: Deployment**

**metadata:**

**name: myapp-replicaset**

**labels:**

**app: myapp**

**type: front-end**

**spec:**

**template:**

**metadata:**

**name: myapp-pod**

**labels:**

**app: myapp**

**type: front-end**

**spec:**

**containers:**

**- name: nginx-container**

**image: nignx**

**replicas: 3**

**selector:**

**matchLabels:**

**type: front-end**

##### Updates and rollback

### > kubectl rollout status <deployment\_name>

### > kubectl rollout history <deployment\_name>

## Networking

* IP assigned to pod
* Internal IP is created
* With multiple nodes:
  + Nodes have different IP
  + Pods have same IP
  + This is a problem within the same cluster (holds nodes)
  + Networking requirements are not setup in kubernetes and needs to be done manually
    - All containers/pods can communicate to one another without NAT(network address translation)
    - All nodes can communicate with all containers and vica-versa without NAT
  + To set it up use tools
    - Cisco, cilium, vmware nsx etc.

## Services

* svc shorthand
* 3 types
  + NodePort
    - 30000-32767 set port range
    - service-definition.yml:

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: myapp-service**

**spec:**

**type: NodePort**

**ports:**

**- targetPort: 80**

**port: 80**

**nodePort: 30008**

**selector:**

**app: myapp**

**type: front-end**

* Selector copied from pod definition file (labels)
  + ClusterIP
    - Creates one static IP to connect to instead of multiple dynamic
    - service-definition.yml:

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: myapp-service**

**spec:**

**type: ClusterIP**

**ports:**

**- targetPort: 80**

**port: 80**

**nodePort: 30008**

**selector:**

**app: myapp**

**type: front-end**

* Selector copied from pod definition file (labels)
  + LoadBalancer
    - Create single URL instead of the multiple URLs
    - Can integrate cloud load balancer
    - service-definition.yml:

**apiVersion: v1**

**kind: Service**

**metadata:**

**name: myapp-service**

**spec:**

**type: LoadBalancer**

**ports:**

**- targetPort: 80**

**port: 80**

**nodePort: 30008**

### > k expose pod <pod\_name> --port=<port> --name=<service\_name>

* + Create service with name to expose an app(pod) on a certain port

# CKA exam preparation

## ETCD

* key-value store
* Listens at port 2379
* Basic commands (v3):

### etcdctl snapshot save

### etcdctl endpoint health

### etcdctl get

### etcdctl put

### export ETCDCTL\_API=3

* + - To set api version to 3
* Role in kubernetes:
  + Changes happen in ETCD

## Namespaces

* ns shorthand

### > k get pods --namespaces=<ns\_name>

### k get pods -A

* + - Get all namespaces

### > k describe ns <ns\_name>

## Scheduling

* Scheduler decides which pod goes on which node
* Filters nodes by pod requirements
  + Pod uses 10GB memory > any nodes below 10GB will be skipped
* Ranks nodes
  + Calculates how much additional space will be on the node if pod is connected > selects the node with the most extra space left
* If no scheduler > pod stays in pending state > manual scheduling to fix
* Manual scheduling
  + Specify nodeName in pod-definition yaml
    - Only when creating pod
  + Create binding object
    - Specify nodeName
    - Send post request to binding API

### > curl --header “Content-Type:application/json” --request POST --data ‘{“apiVersion”:”v1”, “kind”:”Binding”......}’ <api\_url>

## Labels and Selectors

* To group object like pods
* Specified in definition files under labels within spec not metadata

### > k get po --selector app=App1

* + Show pods where app is App1