**KUBERNETES USEFUL COMMANDS**

* **Installation Method In Linux** (Two Easy Method)

1. Install Kubectl To interact with Kubernetes
2. curl -LO https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl
3. kubectl version
4. Install minikube
   1. curl -LO https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
   2. sudo install minikube-linux-amd64 /usr/local/bin/minikube
   3. minikube or minikube version

* minikube start or minikube start --memory=4096 --driver=hyperkit
* kubectl get nodes
* vi pod.yml

apiVersion: v1

kind: Pod

metadata:

name: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

* kubectl create -f pod.yml
* kubectl get pods
* kubectl get pods -o wide
* curl 172.17.0.3 (Image IP address)
* minikube ssh or ssh -I <identity file> sfgg
* curl 172.17.0.3
* kubectl delete pod nginx
* vim pod.yaml (To Open file)
* kubectl log
* kubectl apply -f pod.yml
* kubectl logs nginx
* kubectl logs pod nginx
* kubectl describe pod nginx
* kubectl get deploy
* kubectl delete deploy nginx-deployment.app
* kubectl get all
* kubectl get all -A
* vi.deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

ports:

- containerPort: 80

* kubectl apply -f deployment.yml
* kubectl get deploy
* kubectl get pods (Deploy -> ReplicaSet -> pod)
* kubectl get rs
* kubectl get pods -w
* minikube status
* kubectl get all
* kubectl delete deploy sample-app
* kubectl delete svc demo-service
* cd ~/go/src/github.com/iam-DataEngineer/Docker-Zero-to-Hero
* cd example
* cd python-web-app
* ls
* rm service.yml
* vim Dockerfile
* docker
* docker build -t amit/python-sample-app-demo:v1 .
* vim deployment.yml
* kubectl apply -f deployment.yml
* kubectl get deploy
* kubectl get pods -o wide
* kubectl get pods -v=7
* minikube ssh
* curl -L <http://172.17.0.5>
* curl -L <https://172.17.0.5:8000/demo>
* vim service.yml

apiVersion: v1

kind: Service

metadata:

name: load-balancer-mode-example

spec:

type: NodePort

selector:

app:sample-python-app

ports:

- port: 80

# By default and for convenience, the `targetPort` is set to

# the same value as the `port` field.

targetPort: 8000

# Optional field

# By default and for convenience, the Kubernetes control plane

# will allocate a port from a range (default: 30000-32767)

nodePort: 30007

* kubectl apply -f service.yml
* kubectl get svc -v=9
* kubectl get svc
* minikube ip
* curl -L <http://192.168.64.10>:30007/demo (node IP address)
* kubectl edit svc sample-app
* kubectl get svc
* vim ingress.yml

apiVersion: networking.k8s.io/v1

kind: Ingress

metadata:

name: ingress-example-host

spec:

rules:

- host: "foo.bar.com"

http:

paths:

- pathType: Prefix

path: "/bar"

backend:

service:

name: sample-app

port:

number: 80

* kubectl apply -f ingress.yml
* kubectl get ingress
* curl -L <http://foo.bar.com/bar-v>
* minikube addons enable ingress
* kubectl get pods -A | grep nginx
* kubectl logs ingress-nginx-controller-sample -n
* nginx loadblancer
* nginx.conf -> ingress con
* kubectl get ingress
* minikube IP
* kubectl get cm
* kubectl delete cm Test-cm
* vim cm.yml

apiVersion: v1

kind: ConfigMap

metadata:

name: test-cm

data:

db-port: “3306”

* kubectl apply -f cm.yml
* kubectl get cm
* kubectl describe cm test-cm
* git remote -v
* vim deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

env:

-name: DB-PORT

valueFrom:

configMapKeyRef:

name:test-cm

key: db-port

ports:

- containerPort: 8000

* kubectl get pods -w
* kubectl exec -it sample-python-application -- /bin/bash
* env | grep db
* kubectl apply -f deployment.yml
* kubectl get pods -w
* kubectl exec -it podsample --bin/bash
* vim deployment.yml

apiVersion: apps/v1

kind: Deployment

metadata:

name: nginx-deployment

labels:

app: nginx

spec:

replicas: 3

selector:

matchLabels:

app: nginx

template:

metadata:

labels:

app: nginx

spec:

containers:

- name: nginx

image: nginx:1.14.2

volumeMounts:

* + - * + name: db-connection

mountPath: /opt

ports:

- containerPort: 8000

Volumes:

-name: db-connection

configMap:

name: test-cm

* kubectl apply -f deployment.yml
* kubectl get pods -w
* kubectl exec -it sample-python-app --/bin/bash
* env | grep DB
* ls /opt
* cat /opt/db-port | more
* kubectl create secret generic test-secret --from—literal=db-port=”3306”
* kubectl describe secret test-secret
* minikube start --memory=4098 --driver=hyperkit
* kubectl get pods -A
* helm repo add prometheus-community <https://prometheus-community.github.io/helm-charts>
* helm repo update
* helm install prometheus prometheus-community/prometheus
* kubectl get pods
* kubectl get svc
* kubectl expose service prometheus-server --type=NodePort --target-port=9090 --name=prometheus-server-ext
* minikube ip
* helm repo add grafana <https://grafana.github.io/helm-charts>
* helm repo update
* helm install grafana grafana/grafana
* kubectl get svc
* kubectl expose service grafana --type=NodePort --target-port=3000 --name=grafana-ext
* kubectl get secret –namespace default grafana -o jsonpath=”{.data.admin-password}” | base64 –decode : echo (Provide us login password for grafana)
* kubectl expose service prometheus-kube-state-metrics --type=NodePort --target-port=8080 --name=prometheus-kube-state-metrics-ext
* helm repo add istio <https://istio-release.storage.googleapis.com/charts>
* helm repo update
* kubectl create namespace istio-system
* helm install istio-base istio/base -n istio-system
* helm ls -n istio-system
* helm install istiod istio/istiod -n istio-system --wait