kubernetes

1. what is kubernetes?

- Kubernetes is an open source system for
    - automating deployments,
    - scaling and management of containerized applications
- it is a container orchestration system
- using Kubernetes that means your application is following
    - 1. the microservices architecture and
    - 2. is already containerized

2. what does Orchestrator do ?

- 1. cloning container / replica creation
- 2. restarting container
- 3. destroying container
- 4. replacing existing container

3. which are the Container Orchestation tools

- 1. Docker Swarm
    - used when replica < 10,000
    - simpler than kubernetes
- 2. Kubernetes
    - used when replica(above) > 10,000
- 3. Apache Mesos
- 4. Marathon

4. work on K for developers

- 1. make K , application development ready ---> developer
    - here we have certification : (CKAD= Certified K Application Developer)
    - upload app to K
- 2. cluster administeration : setting,config cluster ---> administrator
    - here we have certification : (CKA : where Administrator) + (CKS : where S: security Specialist)

5. K history?

- developed by Google , in Borg project
- taken over by Cloud Native Computing Foundation (CNFC)
- open source/community

kubernetes ppt

yaml in docker ppt

Kubernetes Cluster demo

0. install virtual box

> https://www.virtualbox.org/wiki/Linux_Downloads

OR

> https://download.virtualbox.org/virtualbox/6.1.16/virtualbox-6.1_6.1.16-140961~Ubuntu~bionic_amd64.deb

1. install vagrant

> $ curl -O https://releases.hashicorp.com/vagrant/2.2.14/vagrant_2.2.14_x86_64.deb

> $ sudo apt install ./vagrant_2.2.14_x86_64.deb

1. Install kubernates from

> https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/install-kubeadm/

- OR use

```
    sudo apt-get update && sudo apt-get install -y apt-transport-https curl
    curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo
apt-key add -
    cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
    deb https://apt.kubernetes.io/ kubernetes-xenial main
    EOF
    sudo apt-get update
    sudo apt-get install -y kubelet kubeadm kubectl
    sudo apt-mark hold kubelet kubeadm kubectl
```

2. akms in pod.yaml file

- a : apiversion
- k : <kind of object>
- m : <metadata>
- spec : <specification of object>

4. create a directory for vagrant and use

```
# creates a Vagrantfile in the directory
vagrant init hashicorp/bionic64
```

- edit the vagrantFile as

```
Vagrant.configure("2") do |config|
  config.vm.box = "ubuntu/bionic64"
  config.vm.provision "shell", path: 'init2.sh'
```

```
  config.vm.define "node1" do |app|
      app.vm.hostname = 'node1'
      app.vm.network "private_network", ip: '192.168.2.201'
      app.vm.provider "virtualbox" do |vb|
          vb.name = "node1"
      end
  end

  config.vm.define "node2" do |app|
      app.vm.hostname = 'node2'
      app.vm.network "private_network", ip: '192.168.2.202'
      app.vm.provider "virtualbox" do |vb|
          vb.name = "node2"
      end
  end
end
```

- create init.sh file

```
sudo apt-get update

sudo apt-get install -y \
  apt-transport-https \
  ca-certificates \
  curl \
  gnupg-agent \
  software-properties-common

curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add
-

sudo add-apt-repository \
  "deb [arch=amd64] https://download.docker.com/linux/ubuntu \
  $(lsb_release -cs) \
  stable"

sudo apt-get update

sudo apt-get install -y docker-ce=5:19.03.12~3-0~ubuntu-bionic

sudo apt-mark hold docker-ce

sudo swapoff -a

sudo sed -i '/ swap / s/^\(.*\)$/#\1/g' /etc/fstab

curl -s https://packages.cloud.google.com/apt/doc/apt-key.gpg | sudo apt-
key add -

cat <<EOF | sudo tee /etc/apt/sources.list.d/kubernetes.list
```

```
deb https://apt.kubernetes.io/ kubernetes-xenial main
EOF

sudo apt-get update

sudo apt-get install -y kubelet=1.19.1-00 kubeadm=1.19.1-00 kubectl=1.19.1-
00

sudo apt-mark hold kubelet kubeadm kubectl
```

- run vagrant commands on terminals

```
$  vagrant up
$  vagrant ssh node1
# initiate kube on terminal 1
sudo kubeadm init --apiserver-advertise-address=<ip-address> --pod-network-
cidr=10.244.0.0/16
# get all nodes
$ sudo kubectl get nodes

$ sudo kubectl describe node1



# initiate kube on terminal 2
$ vagrant ssh node2

$ sudo kubeadm init --apiserver-advertise-address=<ip-address> --pod-
network-cidr=10.244.0.0/16
```

on terminal 1

- create a yaml file for pod creation

> $ vim mypod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod

spec:
  containers:
  - name: mypod-container
    image: nginx
    ports:
    - containerPort: 80
    name: http-port
```

```
    protocol: TCP
```

- create pod using

```
$ kubectl create -f mypod.yml
```

- get info of pod

```
$ kubectl describe pod mypod
```

day2

1. creation Replica of POD using Replica Set and Replica Controller

- 1. demo for need for label in yml file
  - create a 3rd node and create a pod3.yaml file
  - edit pod.yaml file
  - to add replica set info in pod.yml metadata
    - like : desired pod count
  - so label is used to tag/identify the pod by replica set and added by replica set
  - ```
    $ vim pod3.yml
    ```

```
apiVersion: v1
kind: Pod
metadata:
  name: pod3
  labels:
    app: myapp
    type: frontend
    version: v1
spec:
  containers:
  - name: mypod-container
    image: nginx
    ports:
    - containerPort: 80
      protocol: TCP
      # label/tag for RS to identify it
      name: http-port
```

```
:e :qw
```

- 2. create a replica set , for that
  - create rs.yml (rs = ReplicaSet)
```

> vim rs.yml

- command in yml

```
# for replica app/? , as they are complex object
apiVersion: app/v1
# for rs
kind: ReplicaSet

metadata:
  name: myrs
spec:
#specify no of pod
  replicas: 5
  selector:
   # label/tag for RS to identify it
    matchLabels:
      type: frontend
  template:
    metadata:
      name: pod-rs
      label:
        type: frontend
    spec:
      containers:
      - name: pod-1-container
        image: nginx
        ports:
        - containerPort: 80
          protocol: TCP
          name: http-port
```

- when pod created add to replica set , based on matchLabels: type
- commands

```
# create pods based on rs1.yml
 $ kubectl create -f rs1.yml
# shows replica set
 $ kubectl get replicaset

$ kubectl describe pods
$ kubectl get pod
```

CREATE DEPLOYMENT

1. create a deployment , for that

- create deployment.yml

> vim deployment.yml :i

- command in yml

```
# for replica app/? , as they are complex object
apiVersion: app/v1
# for deployment
kind: Deployment

metadata:
  name: myapp-deployment
spec:
#specify no of pod
  replicas: 5
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      name: myapp-pod
      label:
        type: myapp
    spec:
      containers:
      - name: myapp-container
        image: nginx
        ports:
        - containerPort: 80
          protocol: TCP
          name: http-port
```

> :w :wq

- cames out of vim
- adv of deployment
  - can create different versions of deployment
  - it uses RS at backend
- commands

```
$ kubectl get rs
$ kubectl get deployments
# now create one deployment
$ kubectl create -f deployment.yml
# see the running pods
```

```
$ kubectl get pods
# it show pod get recreated as terminated,for count remains constant
$ kubectl delete pod <pod-NAME>
```

installing minikube

```
# installing
$ curl -LO
https://storage.googleapis.com/minikube/releases/latest/minikube-linux-
amd64
$ sudo install minikube-linux-amd64 /usr/local/bin/minikube

# initialize

minikube start
```

kuber for service

1. services

- create a pod

```
mkdir practice
$ vim pod.yml
```

-

```
apiVersion: v1
kind: pod
metadata:
  name: pod1
  labels:
    app: web
spec:
  containers:
  - name: pod1-container
    image: nginx
    port
    - containerPort: 80
    name: http-port
    protocol: TCP
```

-

```
# create a pod
$ kubectl create -f pod.yml
# check
$ kubectl describe pod pod1
```

2.

- service.yml

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort
  selector:
    app: web
  ports:
  - port: 80
    targetPort: 80
    Protocol: TCP
```

- create service

```
# create a service
$ kubectl create -f service.yml
# check the service
$ kubectl get service

$ kubectl describe service nginx-service
```

final app

1. creating a pod using a docker custom image

- pod.yaml

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-myapp
  label:
```

```
    app: myapp
spec:
  containers:
  - name: container-myapp
    image: surajporje/myapp
    ports:
     - port: 80
       targetPort: 80
       Protocol: TCP
```

2. application running in pod with docker image, and expose to outside

```
apiVersion: v1
kind: Service
metadata:
  name: service-myapp
spec:
  type: NodePort
  selector:
    app: myapp
  ports:
  - port: 80
    targetPort: 80
    Protocol: TCP
```

```
kubectl create -f service.yml

kubectl describe service
kubectl get pod

kubectl delete pod  <metadata:name>
```

3. to run a deployment with docker image

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: myapp-d
spec:
  replicas : 5
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
```

```
      name: pod1
      labels:
        app: myapp
    spec:
      containers:
      - name: myapp-container
      image: surajporje/myapp
      ports:
      - containerPort: 80
        name: http-port
        protocol: TCP
```

-

```
# create a deployment
$ kubectl create -f deployment.yml

$ kubectl get pods
$ kubectl get service
```

Kubernetes Interview Questions and Answers

1. What is Kubernetes?

- Kubernetes is an open-source container orchestration tool or system that is used to automate tasks such as the management, monitoring, scaling, and deployment of containerized applications.
- It is used to easily manage several containers (since it can handle grouping of containers), which provides for logical units that can be discovered and managed.

2. What are K8s?

- K8s is another term for Kubernetes.

3. What is orchestration when it comes to software and DevOps?

- Orchestration refers to the integration of multiple services that allows them to automate processes or synchronize information in a timely fashion.
- Say, for example, you have six or seven microservices for an application to run.
- If you place them in separate containers, this would inevitably create obstacles for communication

- Orchestration would help in such a situation by enabling all services in individual containers to work seamlessly to accomplish a single goal.

4. How are Kubernetes and Docker related?

- 1. Docker is an open-source platform used to handle software development.
- Its main benefit is that it packages the settings and dependencies that the software/application needs to run into a container,

- which allows for portability and several other advantages.
- 2. Kubernetes allows for the manual linking and orchestration of several containers, running on multiple hosts that have been created using Docker.

5. What are the main differences between the Docker Swarm and Kubernetes?

- ◦ Docker Swarm is Docker's native, open-source container orchestration platform that is used to cluster and schedule Docker containers.
- Swarm differs from Kubernetes in the following ways:
- 1. Docker Swarm is more convenient to set up but doesn't have a robust cluster,
- while Kubernetes is more complicated to set up but the benefit of having the assurance of a robust cluster
- 2. Docker Swarm can't do auto-scaling (as can Kubernetes); however, Docker scaling is five times faster than Kubernetes
- 3. Docker Swarm doesn't have a GUI;
- Kubernetes has a GUI in the form of a dashboard
- 4. Docker Swarm does automatic load balancing of traffic between containers in a cluster,
- while Kubernetes requires manual intervention for load balancing such traffic
- 5. Docker can deploy rolling updates but can't deploy automatic rollbacks;
- Kubernetes can deploy rolling updates as well as automatic rollbacks

6. What are the main components of Kubernetes architecture?

- There are two primary components:
- the master node and
- the worker node.
- Each of these components has individual components in them.

7. What is a node in Kubernetes?

- A node is the smallest fundamental unit of computing hardware.
- It represents a single machine in a cluster, which could be a physical machine in a data center or a virtual machine from a cloud provider.
- Each machine can substitute any other machine in a Kubernetes cluster.
- The master in Kubernetes controls the nodes that have containers.

8. What does the node status contain?

- The main components of a node status are Address, Condition, Capacity, and Info.

9. What process runs on Kubernetes Master Node?

- The Kube-api server process runs on the master node and serves to scale the deployment of more instances.

10. What is a pod in Kubernetes?

- Pods are high-level structures that wrap one or more containers.
- This is because containers are not run directly in Kubernetes.
- Containers in the same pod share a local network and the same resources, allowing them to easily communicate with other containers in the same pod as if they were on the same machine while at

the same time maintaining a degree of isolation.

11. What is the job of the kube-scheduler?

- The kube-scheduler assigns nodes to newly created pods.

12. What is a cluster of containers in Kubernetes?

- A cluster of containers is a set of machine elements that are nodes.
- Clusters initiate specific routes so that the containers running on the nodes can communicate with each other.
- In Kubernetes, the container engine (not the server of the Kubernetes API) provides hosting for the API server.

13. What is the Google Container Engine?

- The Google Container Engine is an open-source management platform tailor-made for Docker containers and clusters to provide support for the clusters that run in Google public cloud services.

14. What are Daemon sets?

- A Daemon set is a set of pods that runs only once on a host.
- They are used for host layer attributes like a network or for monitoring a network, which you may not need to run on a host more than once.

15. What is 'Heapster' in Kubernetes?

- A Heapster is a performance monitoring and metrics collection system for data collected by the Kublet.
- This aggregator is natively supported and runs like any other pod within a Kubernetes cluster, which allows it to discover and query usage data from all nodes within the cluster.

16. What is a Namespace in Kubernetes?

- Namespaces are used for dividing cluster resources between multiple users.
- They are meant for environments where there are many users spread across projects or teams and provide a scope of resources.

17. Name the initial namespaces from which Kubernetes starts?

- Default
    - Kube – system
    - Kube – public

18. What is the Kubernetes controller manager?

- The controller manager is a daemon that is used for embedding core control loops, garbage collection, and Namespace creation.
- It enables the running of multiple processes on the master node even though they are compiled to run as a single process.

19. What are the types of controller managers?

- The primary controller managers that can run on the master node are the
    - endpoints controller,
    - service accounts controller,
    - namespace controller,
    - node controller,
    - token controller, and
    - replication controller.

20. What is etcd?

- Kubernetes uses etcd as a distributed key-value store for all of its data,
- including metadata and configuration data, and allows nodes in Kubernetes clusters to read and write data.
- Although etcd was purposely built for CoreOS, it also works on a variety of operating systems (
- e.g., Linux, BSB, and OS X, because it is open-source.
- Etcd represents the state of a cluster at a specific moment in time and is a canonical hub for state management and cluster coordination of a Kubernetes cluster.

21. What are the different services within Kubernetes?

- Different types of Kubernetes services include: + Cluster IP service + Node Port service + External Name Creation service and + Load Balancer service

22. What is ClusterIP?

- The ClusterIP is the default Kubernetes service
- that provides a service inside a cluster (with no external access) that other apps inside your cluster can access.

23. What is NodePort?

- The NodePort service is the most fundamental way to get external traffic directly to your service.
- It opens a specific port on all Nodes and forwards any traffic sent to this port to the service.

24. What is the LoadBalancer in Kubernetes?

- The LoadBalancer service is used to expose services to the internet.
- A Network load balancer, for example, creates a single IP address that forwards all traffic to your service.

25. What is a headless service?

- A headless service is used to interface with service discovery mechanisms without being tied to a ClusterIP, therefore allowing you to directly reach pods without having to access them through a proxy.
- It is useful when neither load balancing nor a single Service IP is required.

26. What is Kubelet?

- The kubelet is a service agent that controls and maintains a set of pods by watching for pod specs through the Kubernetes API server.

- It preserves the pod lifecycle by ensuring that a given set of containers are all running as they should.
- The kubelet runs on each node and enables the communication between the master and slave nodes.

27. What is Kubectl?

- Kubectl is a CLI (command-line interface)
- that is used to run commands against Kubernetes clusters.
- As such, it controls the Kubernetes cluster manager through different create and manage commands on the Kubernetes component

28. Give examples of recommended security measures for Kubernetes.

- Examples of standard Kubernetes security measures
- include defining resource quotas, support for auditing, restriction of etcd access, regular security updates to the environment, network segmentation, definition of strict resource policies, continuous scanning for security vulnerabilities, and using images from authorized repositories.

29. What is Kube-proxy?

- Kube-proxy is an implementation of a load balancer and network proxy used to support service abstraction with other networking operations.
- Kube-proxy is responsible for directing traffic to the right container based on IP and the port number of incoming requests.

30. How can you get a static IP for a Kubernetes load balancer?

- A static IP for the Kubernetes load balancer can be achieved by changing DNS records since the Kubernetes Master can assign a new static IP address.