KUBERNETES :- 04/04/22

Orchestration tool

Managing high availability, scalability, deployment and etc.,

Master Node :- components '4'

* API server.
* schedular
* controller
* etcd.

Worker Node :-

* kubelet
* kube-proxy
* Docker

Differences :-

Kubernets	DOcker swarm
complex installation	Simple installion : installion
comes with an inbuilt Dashboard	There's no dashboard : GUI
High scalability 5000 node clusters with 1,50,000 pods	Very high scalability : scalability 1000 node clusters with. 30,000 containers.
inbuilt tool available for logging.	lack of inbuilt tool. : logging & monitoring

KUBERNETES AWS-2 [set-up]

Master - Medium. [Ubuntu server 18.4 Version]

Worker - Micro.

Pass the script in the instance creation (Bootable script)

in step 3, User data (script) → from PPT.

Review and Launch.

Connect all the Nodes.

sudo kubeadm init --pod-network-cidr=(I.P)/16.

[Container = Pod].

Now,

set up the user from the help provided in the instance

Whenever we want to manage cluster with a user, then user

User should have a key.

Kubectl version → (To check if cluster is working)

(Weavenet Network)

command from pdf

→ create the GUI

[provide the commands from pdf].

Now,

With Ip address and port : 31000

→ Kubernetes Dashboard is ready

→ To connect to work Nodes join command is needed for that

* Kubeadm token create --print-join-command

we will get join command, run this command in worker nodes

* Kubectl get Nodes

→ TO see the Nodes

* Kubectl

→ TO see all the basic commands and etc.,

* Kubectl -help

→ TO see all options of the commands.

* Kubectl describe node ip-ipaddress.

→ To see the information of component

* Kubectl get pods

→ To see the containers.

whatever, the pods are running they are placed in Name Space

When ever we create pod on kube by default it will go to "Default"

When ever we try to fetch it will try to fetch from "Default"

Here, the pods which are running, running on kube-system.

* Kubectl get pod -n kube-system.

[-n = namespace]

How to know certain namespace is running in our cluster.

* Kubectl get namespace.

When main launches components kube-system

user " " Default.

Now, Deployment using user.

* Kubectl run nginx --image nginx.

* Kubectl get pods.

⇒ we used user. So, component is in default.

Kubectl get namespaces.

⇒ To create our namespace.

* kubectl create ns dev.

⇒ when ever you want to deploy into other namespace.

* kubectl run nginx --image nginx -n dev.

* kubectl get pods -n dev.

pod = small unit in kubernetes

⇒ To know information of pod.

* kubectl describe pod [pod name] -n [namespace name].

ex:- kubectl describe pod [pod name] -n dev

⇒ To check logs on container

* kubectl logs [pod name] -n dev. ⇒ for 1 container.

* kubectl log [pod name] -n dev -c container name

⇒ To know what all resources are available on the cluster

* kubectl get all

⇒ To clean up certain component.

*. kubectl delete deployment.apps/nginx.

⇒ When we delete the deploy component then pod and replicaset also deleted but service is present because its default

⇒ To create a pod.

* kubectl run nginx --image nginx

here, deployment is created with the pod.

⇒ To create only pod [no deployment and replicaset].

* vi pod.yml

⇒ paste the code

⇒ To ~~create~~ trigger the file

* kubectl create -f pod.yml

create first time

* kubectl apply -f (filename) second time

* kubectl get all

⇒ Here only pod and service are created no deploy and replicaset

⇒ To delete any pod

* kubectl delete pod (pod name).

⇒ To access the pod with port

* kubectl expose pod sample-pod --port=80 --type=NodePort

* kubectl get all

05/04/22

*kubectl get nodes

→ To check the Nodes present.

Service

→ To access outside service is needed

deployment and replicatis :-

⇒ high availability

deployment

api version : extentions

kind : Deployment.

Yaml file = manifest file.

kubectl create -f deloy.yml.

replicaset = creating pod when one pod deleted.

Service acts as a Load balancer

vi service.yml.

api version : v1

kind : service.

**"selector" is 'app name'

kubectl create -f service.yml

i) Nodeport:- for service range is 30,000 to 31,700

ii) ClusterIp :- is for internal use not for external use.

iii) loadBalancer no port, we will get constant Ip. by default we will not get loadBalancer we have to install the loadBalancer.

iv) External Name :- Attaching the domain name to server.

Services available on the kubernetes these '4' :-

To scale up the replicas

Vi deploy.yml

⇒ replicas = 3.

Hence scaled to 3 pods.

* kubectl get pod -o wide ← kubectl describe pod podname instead of this

⇒ To get more information about pods.

kubectl get Nodes

⇒ To see the nodes.

Service Object = Creating Service + Load balancing + Grouping.

Service (or) src

To edit the deployment (or) check the manifest file.

* kubectl edit deployment.apps/nginx

scaling :- [with command] → manually

--replicas=5

* kubectl scale --current-replicas=3 deployment.apps/ sample-deploy.

but, with manifest file it's easy.

Automatic scaling :- (horizontal pod Anto scalaer) with.

HPA uses Heapster.

Heapster will not be available by default.

* kubectl top nodes

→ Top is command for metrics cpu and ram utilization on nodes

* kubectl top pods.

git clone (path from pdf)

↓

cd metric-server.

ls

cd metric-server/

ls

cd deploy/

ls

cd 1.8+/

ls

→ metric files.

* kubectl apply -f .

* kubectl get all -n kube-system

To check the logs.

* kubectl log [pod/metric server] -n kube-system.

Application deployment

* kubectl run nginx --image nginx

To access the application [expose]

* kubectl expose deploy nginx --port 80 --type Nodeport.

* kubectl get all

Now, "Auto scaling"

* kubectl autoscale deploy nginx --min 1 --max 5
--cpu-persent 20

[Real time 70-75]

* kubectl get all.

* kubectl describe hpa

→ To know hpa trouble-shooting

* kubectl delete hpa nginx.

Now update the deployment

* kubectl edit deploy nginx

Now,
add resource code

```
resources:
  limits:
    cpu: "100m"
  requests:
    cpu: "100m"
```

Now, scale again

kubectl autoscale deploy nginx --min 1 --max 5
--cpu-percent 20

* kubectl get all.

* kubectl describe hpa

* kubectl top nodes.

Tool for checking the load testing.

* Sudo apt install Siege

* Siege -q -c 5 -t 2m http://ip adress:port Node

-q : quite mode

-c : conquerent mode

-t : time

Now, the checking of load on Nodes is for 2m and after that it will scale down.

Deploying Jenkins [Jenkins]

* Vi Jenkins.yml
  ⇒ Paste the code
* Vi Jenkins-service.yml
  ⇒ Paste the code ⇒ port : 32000
* Kubectl apply -f jenkins.yml
* Kubectl apply -f Jenkins-service.yml
* Kubectl get all
* Kubectl logs [deployment_jenkins]
  ↓
  pod name.

[Yaml online validator to validate the yaml file]

Run log command to know the install password and then access the jenkins

06/4/22.

Deployment strategies :-

Rolling update :- Default from Kubernetes

Pros :-
Version is slowly released across instances.
convenient for stateful applications.

Cons :-
rollout/rollback can take time.
Supporting multiple APIs is hard
no control over traffic

Now,

```
git clone [path from pdf]
ls
cd kuber/
ls
cd deployment-strategies/
ls
cd rolling-update
ls
cat app-v1.yml
```

These are health probes

⇒ liveliness probe for recreate the pods if not running due to some issue.
Readyness probe
it will health state of container

Now, deploy

* kubectl apply -f app-v1.yaml

* kubectl get all

→ Here all are created at same point.

* kubectl get pods -w (continous monetoring)

-w = watch

maxSurge: 1
maxUnavailable: 0

Here, in Rollout after executing another pod (V2) then in the (V1) pods one after another gets deleted after another is created from (V2)

and in (V1) the service creation is done so in (V2) the service is not needed.

* kubectl apply -f app-v2.yml

→ After execution time taken from pod creation differes for (V2)

Rollback (Revert) :-

kubectl get deployments

→ status command to check successful deployment.

for my-app

* Kubectl rollout status deployment my-app

* Kubectl rollout history deployment my-app

⇒ To see the history (or) revision. (max revision=10).

* Kubectl rollout history deployment my-app --revision=2

⇒ To check to the revision (or) version we want to track deployments

Revert Changes:-

* Kubectl rollout undo deployment my-app.

*Kubectl rollout undo deployment my-app --to-revision= revision number

⇒ If we want to revert (or) rollout to particular version (or) revision

Now, delete the

* Kubectl delete (deployment) (service).

* Kubectl get all

Recreate :-

Pros :- Application state entirely renewed

Cons :- downtime that depends on both shutdown & boot duration of the app

cd ..

cd recreate/

cat app-v1.yml

→ Strategy:
    type: Recreate.

Now, deploy.

* kubectl apply -f app-v1.yaml
* kubectl get all
* kubectl get svc

TO know the service (or) Node port.

* kubectl get deploy

TO know the deployment app name.

* kubectl describe deploy my-app
* kubectl apply -f app-v2.yaml

→ When deploying the new version all will be created at a time and all will be deleted at a time. So, the here downtime there (very very downtime) but downtime is there

kubectl delete (deploy) (svc)

Blue/Green :- Two clusters.

Routing the traffic b/w the clusters.

Expensive method

* cd..
* cd blue/green/
* cd single-service/

~~kubectl apply -f app-v1.yaml~~

* Kubectl apply -f .
* Kubectl get pods -o wide

⇒ To checked which pods are grouped

* Kubectl patch service my-app -p '{"spec": {selector": {version": "v2.0.0"}}}'

⇒ ~~patch~~ (patch) we can update the svc object to point to v2 pods.

* Kubectl get pods -o wide ⇒ to see traffic request is running on which pod.

kubectl describe ~~svc~~ my-app

Canary :-

We will release to set-of users but not all.

Easy to maintain

If any issue impact will be minimum.

cd canary/

ls

cd native/

ls.

cat app-v1.yaml

cat app-v2-yaml.

kubectl apply -f app-v1.yaml

kubectl get all

Now, scale down the pods.

* kubectl scale --replicas=4 deployment.app/my-app-v1
* kubectl get all
* kubectl apply -f app-v2.yaml
* kubectl get all

Now, 4 will be with v1 and 1 will be with v2

If that (1) is working fine then. {scale v1 to 3 and v2 to 2}

* kubectl scale --replicas=3 deployment.app/my-app-v1

* kubectl scale --replicas=2 deployment.apps/my-app-v2

* kubectl get all.

* What are the strategies available?
* How are we are deploying code into cluster?
* Which strategies you followed! Rolling update.

In project used Rolling-update.

07/04/22

Monitoring :-

```
git clone (repo)
ls
cd kubernetes 1/
ls
cd monitoring/
kubectl get pods -all-namespaces
kubectl get nodes
ls
cd kubernetes-prometheus/
ls
```

⇒ config-map.yaml } is to create env-variable in pods.
Secrets } doesn't belong to any name space

kind : configmap

⇒ ClusterRole.yaml

* Rbac :- role base asses creating
* creating users and providing the access

Role sets permissions within a name space.
clusterrole is a non-name space source.

kind : Role

rules : creating groups & giving permissions

Role binding

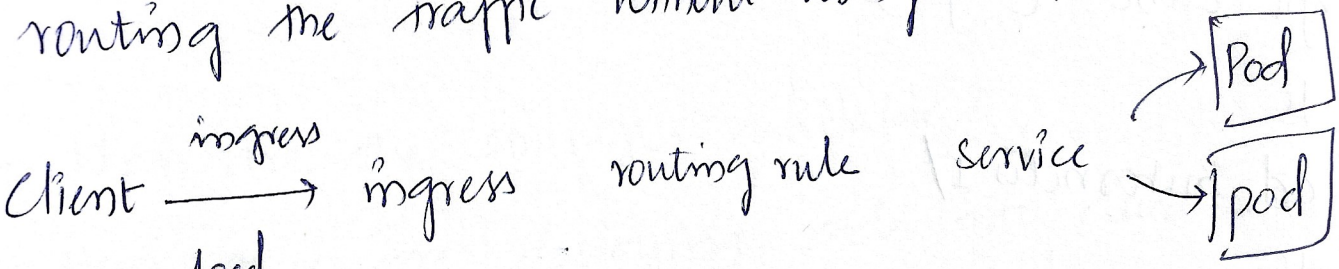
kind : role binding

giving authentication to users (or) groups on cluster level

Prometheus-ingres:- kind: ingres

kind: secrets.

routing the traffic without using Load balancer.



ls

cat prometheus-deployment:-

emptyDir: {} Temporary storage once pod deleted then the storage gone

ls

know, execute

8:11

kubectl apply -f.

kubectl delete -f.

kubectl get ns

kubectl create ns monitoring

kubectl get ns

→ monitoring name space is created

kubectl apply -f.

kubectl get all -n monitoring.

Prometheus is ready "it is a Query based tool."

cd ..

~~Grafana~~ ls

kubernetes - grafana/

ls

kubectl apply -f .

kubectl get all -n monitoring.

Grafana monitoring.

User : Admin

pass : Admin

Change password: Admin

+ create

import

8588	Load

prometheus.

import

To create dash board.

+ create

Add a new panel.

Title : . m:- podcount

visualization : m:- Guage.

Data source prometheus

metric browser ex:- kubelet_running_pod_count.

⇒ Weave Scope :-

hit on browser and take instruction (installion) and run on Master Node.

kubectl apply -f (instruction).

kubectl get all -n weave

Now, update service from cluster ip to Node port.

kubectl get svc (past path).

kubectl get all -n weave

⇒ port is created (hit on browser).

Kubernetes - Elk :- Central log streaming

cd
ls
cd Kubernetes-1/
ls
cd monitoring/
ls.
cd Kubernetes-elk/
ls
Kubectl apply -f .
Kubectl get all
Kubectl get svc --all-namespaces.

⇒ Elasti search
⇒ log stash
⇒ Kibana

⇒ Kibana (visualisation tool).
⇒ Elastic search - logging. (data base)

→ Kibana

set up index pattern.
index pattern.
logstash *
next step.
@ time stamp.
Create index pattern

create apache and check

vi apache.yml

→ past the code

kubectl apply -f achache.yml

kubectl get all

→ kibana

Discover

add a filter

| kubernetes.label.app | is | web |

Stateful Set :-

→ to manage the sets

It works in order wise

08/04/2022

daemonset:-

Node are present on the cluster, so, it will make sure one copy is going to run on every node.

3 copies - Deamonset = 3 copies on 3 nodes

3 copies - Deployment = 3 copies on 2 nodes (or) 1 nodes.

Deamonset makesure one pod is running on each node

Deployment not sure about how many pods created on each node.

only kind is different

kind = Deamonset.

Persistant volume and persistant volume claim:-

This volume are created seperatly and mounted to any node of cluster and now pods can connect to volume.

Appnities and Anti-Appnities:-

node appnity :- Node level

pod appnity :- pod level

Kubectl get nods

Kubectl describe node [any ip]

at top Labels:

key: kubernetes.io/os

"If Label is match of the node then it will deploy on the node.

Affinities :- it must match the condition then it will create [Node/pod].

Anti :- if condition is false then it will create [Node/pod]

I want to deploy pod in particular node !

Node selector (or) when Node selector not working.

Affinity is used.

Taints : NoShedule [usually on master].

No pods are created on Taints.

But to create pods on Taints we will "Tolerations":

k8's helm :- For files need to execute in k8's.

- deploy
- service
- routing - ingress.

→ deploying service in to cluster without using the manifest files is below.

it's a package

Interview Questions:-

Deployments
Service
Deamonsets
Service accounts
configmaps
Secrets
Pv
Pvc
statefulsets
replicates
Rok & Rolebinding
Cluster & Cluster role
binding
Node selectors
Affinity & Antiaffinity
Taints & Tolerations

resource type
Deployment
daemonset
statefulsets

	Monolithic Service	micro service
Taken the code from SCM	GIT	GIT
Run the unit test cases	maven	maven
code Quality	SonarQube	SonarQube
Create Artifacts	maven	DOCKER images
push the artifacts to repo	Nexus	
Deployment	Ansible	kubernetes
publish reports	Junit, Cobertura	Junit
Email communication	Email	Email