Containes Orchestration.

→ Containes Orchestration is a process that automates the Containes management lifecycle for containerized application. Including managements, deployments, scaling, networking, and availibity.

★ Beneficial for large application becaus.

1. Seamless complexity.
2. Easily integerated with CI/CD pipelines.
3. Hands off depolyment & scaling.
4. Increases Speed agility & efficiency.
5. Developers can work togethes for efficient resource ubilization.

→ Containes Orchestration is easily implemented on any cloud including public, private & multi-cloud.

→ Helps orgs. to implement SOAR features by Respone.

features.

~~benefits~~ of Containes Orchestration.

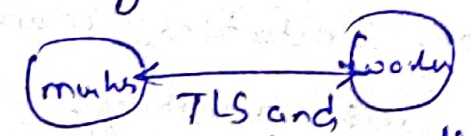
1. provisioning and dyplayment of containes.
2. Implement secured comm blw containes's
3. allocation of containes's based upon Performance.
4. per scaling of containes when required.
5. performing rolling updates & rollbacks.
6. Health monitoring of containes.

★ Benefits of Containes Orchestration

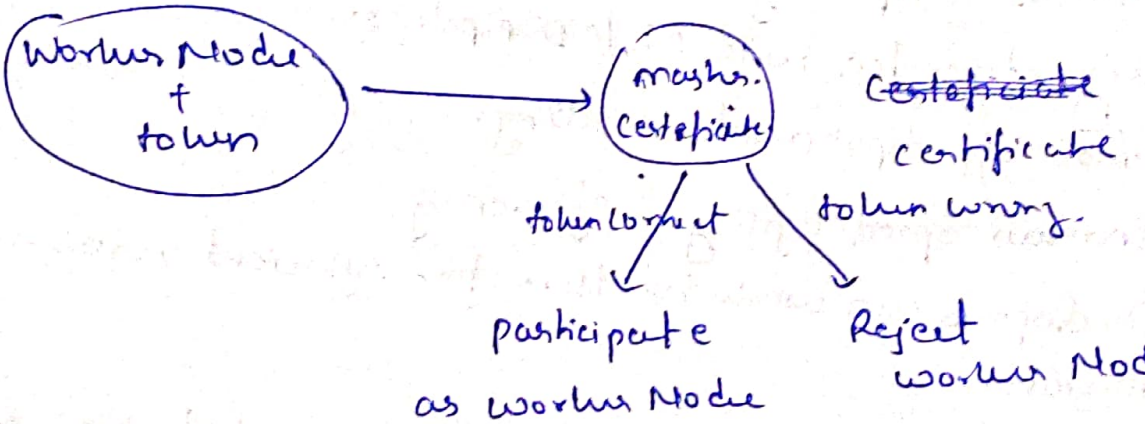
PDCSSR

TLS authentication (Transport layer security).

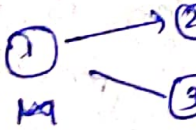
A manager Node can reject a worker Node based upon the token generated by master Node.

Secure communication blw 

A Ger certificate is already created at master while creation of so Node. (token).



How to create cluster.



1. make sure docker is already installed in all machines.
2. docker swarm init (O/P) token with join commands.
3. docker node ls.
4. docker swarm join --token <token-from-manager>
   Now paste above command in terminal of both the workers.
5. docker node ls → O/P. (3) Nodes as output. ? only work on manager Node.
6. Command to get the token after some few days.
   docker swarm join-token (worker).

(7) How to add manager node (multiple).

docker swarm join-token manager → o/p token. run in V.M will act as manager

(8) How to add worker node

docker swarm join-token worker. → o/p token run in V.M act as worker

(9) How to remove node from cluster.

docker swarm leave [on terminal of worker node after using this command node status will be down from running.]

(10) How to completely remove node from cluster.

docker node rm [on master node].

docker node rm <worker-name>

Eg worker01

(11) How to forcefully delete a node without leaving swarm.

docker node rm -f <worker-name>

Eg worker02

(12) docker node inspect <workerName> } used on managers.

(13) How to promote ~~worker~~ a Node (worker → manager)

docker node promote worker01 worker02.

(14) How to demode a Node

docker node demote. <manager-name>

Docker service commands.

↳ to create the services on whole docker. docker service command is use. Only Run on's on manager Node'].

(1) How to create service

docker service create. <image-name> <command to create inside container>

docker service create -d alpine ping 192.168.25.10

(2) How to get running service.

docker service ls

(3) docker service inspect <service-id>

(4) How to create service with 4 replicas.

docker service create -d --replicas=4 alpine ping

(5) Status about service

docker service ps <service-id>

(1) If we use (docker node rm -f) command of host/worker running containers then it will delete the containers.

If master/manager Node again create two more containers to meet desired state of cluster is 4.

Use docker service ls. command get the running status of each service.

How to scaleup & scale down replicas / service.

docker service scale <image id = No. of replicas>

ie, docker service scale id=4

How to remove service.

docker service rm <service-id>

Port Mapping

similar to containers

docker service create -d (-p 8090:80) nginx

This nginx replica service is accessible on any virtual machine with their respective ip address. Port: respective IP address : port

How to create global service.

docker service create --mode=global ~~nginx~~ topline ping 8.8.8.8

global service mode = No. of ~~pods~~ VM [each node].

How to create ~~node~~ service only on manager or worker.

--constraints = "node.role==worker"

How to apply Rolling updates on services.

~~update~~

--update-delay flag } only apply some in update.

docker service create -d --replicas 3 --name nginx-se

--update-delay 10m20s nginx ping 8.8.8.8

Eg.

Old version in service.

docker service create --name demo nginx:1.24.0

New version update

docker service update (--image nginx:1.25.7) → Latest version

(<image-name>) /id

↓

oudated image name previously created.

docker service im update --image 1.27.0 demo.

Docker Networking

① Allow communication C_1 & C_2. (default bridge) 172.17.0.1

(a) docker run -d --name login nginx. → C1

(b) docker exec -it login /bin/bash

(c) apt-get install iputils-ping -y

(d) docker run -d --name logout nginx → C2

172.17.0.2

(e) docker network ls

(f) ~~docker network rm <name>~~

② Don't allow communication C_1 & C_2 (custom bridge)

(1) docker network create <u>secure-network</u>

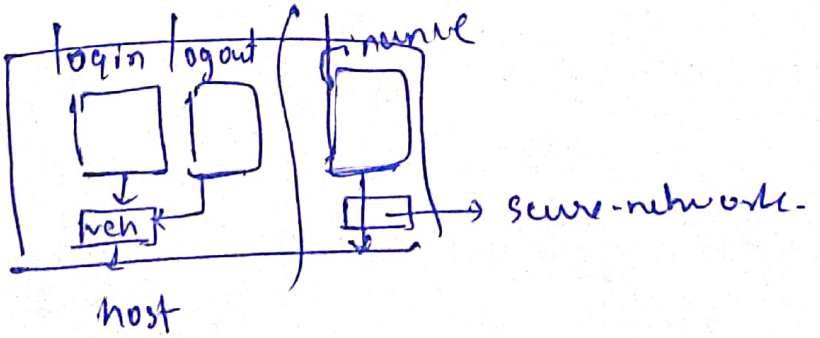
(2) docker run -d --name payments finance --network.

secure-network nginx

(3) docker ps.

(4) docker inspect finance.

<container.name>



Outer. VPC | EC2 => 192.168.0.1.

Container: 172.17.0.1

Custom bridge Network: ~~172.18.0.2~~
172.18.0.1

Container benifits.

1. Isolation & ~~Automation~~ Allocation.
2. Server Utilization
3. Provisioning
4. Portability
5. Performance
6. Reliability.
7. Scalability
8. Automation.

Kubernetes

① Kubernetes is an Open-source cloud orchestration platform that helps in management of containerized application. automating deployment, scaling and.

② Kubernetes grew as extension of container as Service (CaaS)

③ Kubernetes portable across cloud on prems is.

What Kubernetes Can't Do.

① Does not act as traditional paas.

② Does not provide (CI/CD) pipelines for build our application & deploy our source code.

③ Does not provide logging, monitoring & altering solutions.

④ Does not provide build in middle ware & database.

⑤ D Not rigid nor opiniated but ~~support ware~~ provide flexible model that support varity of workloads including stateful, stateless & data processing workloads.

Kubernetes Capabilities

① Automated rollbacks & rollouts

② Storage orchestration

③ horizontal scaling

④ Automated bin packing

⑤ Secrets & config management

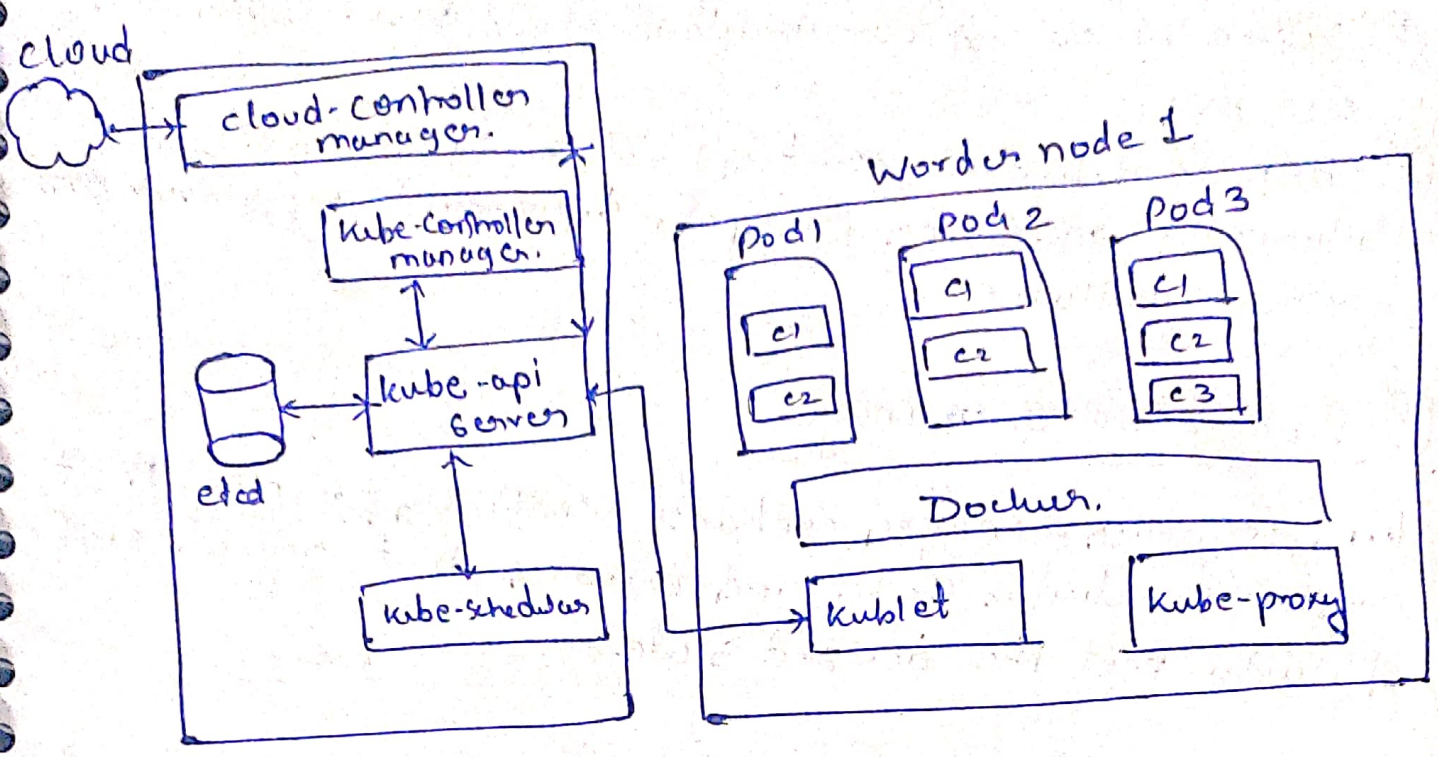
⑥ IPV4/IPV6 dual stack

⑦ Batch Execution

⑧ Self Healing

⑨ Self Discovery & load balancing

Kubernetes Architecture



Control plane.

Deployment of K8S is known as Deployment cluster. Deployment cluster consist of cluster of nodes that run containerized application.

① kube-api-server

ⓐ front end of kubernetes control plane.
ⓑ All components of kubernetes cluster interact with api exposed by kube-api-server.
ⓒ It helps to scale Horizontally by adding more no. of instances.
ⓓ We can create several instance of kube-api-server and manange the workload on those instances. distribute.

② etcd.

① etcd is highly available key-value store that contains cluster data.
② etcd stores desired state of our cluster.
③ when we deploy our app. all deployment config are stored in etcd
④ etcd stores the cluster state. And system works to bring actual to desired state.
to match.

(3) kube-scheduler

(a) assigns to newly created pods to containers.
(b) Determine where our workload shoud b run within the cluster.
(c) Helps to determine most optimal node according kubernetes scheduling principal, deployment configuration & resources available.

(4) kube-controller-manager.

kube-controller manager contains controller process that monitor the cluster state and ensures that actual state of cluster met the desired state.

(5) cloud-controllermanager.

(1) run's controller's that helps to interact with underlying CSP.
(2) cloud-controller manger allows both CSP, kubernetes to evolve freely without introducing dependency on each other.

Worker plane/Node.

(A) kubelet.

(1) kube-controller manager interact with kube api-server to get new & modified pod specification and ensures that pod and their associated container's are running in desired state.
(2) kublet sends health and status report to Control plane
(3) To start a pod, kubelet uses container runtime. Container runtime is responsible for downloading images & starting container.
(4) Instead of using single container runtime, container runtime interface is implemented.

kube proxy

① kube-proxy is network proxy that run on each node of cluster.

② kube-proxy defines network rules that allow communication b/w pods ~~running~~ within a node.

③ ~~Network~~, communication can be within or outside the cluster. kubelet allows is

mono, micro & SOA architecture

① monolithic architecture

In software industry monolithic simply means individual software unit that can be ~~break~~ into multiple components. The main concept of monlithic architecture lies in multiple components of an application are combined together to perform some specific functions. And monolithic ~~Monolithic~~ architecture consist of database, user-interface, server-backend.

Good cases

① ~~Monolith~~ Monolithic architecture is good for if company is at founding stage, less experience developer & doesn't have knowledge about microservice.

② When we are building unproven product & launch into the market as soon as possible.

③ Good for startup.

Advantages

① Simple & easy development as well as deployment

② less cost-cutting.

③ Better performance

Dissavantages.

① Complex codebase become complex over time.

② less Agility.

③ Difficult to ~~move~~ shift new on another architecture / technology.

SOA architecture.

SOA refers software development architectural style in which discrete components & loosly coupled (components) are combine together to perform some specific function. software agents
These software agents are of two type. service provides & service consumer.

The main concept of SOA architecture lies that build and design our application in such a way that different ~~Components~~ modules are integrated seamlessly and ~~code~~ can be easily resused.

use case

① best suitated for complex enterprise system for bank.
② banking system are complex that can't break into microservices & monolithic is also not good as single point failure.

Advantages.

① Reusibility of service.
② Better manageability.
③ Better / high reliability.
④ Parallel development.

Disadvantages.

① Complex schp. management
② high investment cost.
③ extra overload.

Kubectl

kubectl (command)
C

① CLI / Command line tool line
② command line interface
③ Deploy application, management of kubernetes clusters.

Command structure

3 Types. Command
① Imperative ② Imperative object config.
③ Declarative config.

Imperative

① In Imperative, we can use commands to create, update and delete live objects directly.
② In Imperative commands operation to be performed as specific in command itself ~~with~~ as parameter & flags.
③ Imperative command docs not provide audit trails :., it is very difficult to track the changes.
④ Imperative commands are not flexible & doesn't use any template.
⑤ Best suitable for development & testing environment.

Imperative limitation.

For example a Developer1 Deploy an application on k8's cluster & Developer2 also want to deploy same application. Then Developer2 requires the exact command that used by Developer1 to Deploy & run it.

Solution: Both the developer's must follow the common ~~shared~~ template. That used in Imperative object configuration.

② Imperative object configuration.

① In Imperative object configuration operation to be be performed must be specified in the instruction.
② along with at least one configuration file
③ Configuration file contains the inform full definitation of the live objects.

Advantages

① configuration file can easily stores in version control system.
② integreate with change process.
③ provide avid trails & template for creating objects.

Disadvantages.

① requries understanding of object schemes.
② written json & or yaml format.

limitations.

If Developer performs some update operation that are not mereged with configuration file. Then at another developer still uses outdated configuration file for for future development.

Solutions

Developer must uses shared configuration file that contains desired state of kubernetes cluster. That used in Dedcrative object configuration.

Declerative object configuration.

(1) In Declarative object config. operation to be performed are identified by kubectl itself and user doesn't provide command is explicitly.

(2) Declarative object configuration are best for production enviroment.

(3) Configuration contains information of desired & system works on to bring actual to desired state automatically.

(4) User's doesn't requrie to perform any operation.

Benifits

If A developer's perform some update operation in running application. since config. data are stored in shared tempelearte. If another developer misses the served updates Then, Another Developer have to only apply that same shared configuration tempelate. kubernetes automatically perform's the neccessary changes using that shared tempelate to met actual state with desired state.

① kubectl get pods
② kubectl get deployment
③ kubectl get services
④ kubectl get replicaset
⑤ kubectl get deploy
↓
get the deployment details

Imperative command without config files.

How to create deployment.

kubectl create deployment nginx-server --image=nginx --port=80

① deployment created → ② Replica set are defined. → ③ based upon replica set pods are created.

How to get information about deployment.

① kubectl describe deployment <deployment-name>
② kubectl describe pod <pod-name/Id>
③ kubectl describe replicaset

How to edit replicaset.

① kubectl edit deployment

How to show logs.

① kubectl ~~get~~ logs

How to delete deployment.

① kubectl delete deployment <name>
② kubectl delete pods <pod-name>
↓ fullname: <deployment-name-id>

(or)

kubectl delete -l app=nginx-server

How to create configuration file.

```
apiVersion: apps/v1
kind: Deployment
metadata:
    name: myapp-deploy   # name of the deployment
    labels:
      app: myapp   # nginx
spec:
    replicaset
    replicas: 1
    selectors:
      matchlables:
        app: myapp
        :
```

How to apply configuration file.

```
kubectl apply -f ./deploy.yaml
```

★ How to scale No. of Replicas without using Config file and with using Config:

① ~~kubectl scale --replicas=3 -f foo.yaml~~

Eg ② Deployment file name: nginx-deployment

How to scale replica set

→ Declarative: update configuration file & apply it again

→ Imperative:

```
kubectl scale deployment nginix-deployment
-replicas=2
```

How to create Replicaset from Scratch.

① First of update kind label for configuration file using and replace with kind: Replicaset

② As per our requirements we or can also update the No. of replicas required for our deployments by updating ~~kind~~: ~~replicas~~: replicas: 3 label.

③ Now by using create replicaset command apply to create replicw set i.e,

First time creation of kubectl create -f replicaset.yaml.

To override of kubectl apply -f replicaset.yaml previous resource config.

④ ~~Observe~~ observe the status of pods using

kubectl get pods

⑤ Now use get replicaset to get the info about newly created pods. Replicaset

kubectl get replicaset

Qn why it is always prefered to create a deployment that includes replicaset rather than creating a single replica set?

Ans. chatgpt.

Volume Commands

1. docker volume create my-vol
2. docker volume ls
3. docker volume inspect my-vol
4. docker volume rm

Features of docker compose.

1. multiple isolated environment on a single host.
2. Persist data when containers are created.
3. Only create changed files in containers.
4. Support environment variables.

kubectl commands

1. kubectl create -f
2. kubectl apply -f
3. kubectl get
4. kubectl describe
5. kubectl edit
6. kubectl scale