4/3/25

Kubernetes story

home → Restaurant

chef:

① you cook each & everything.

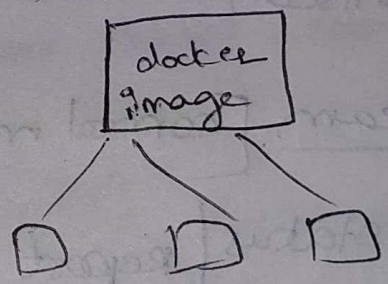
the other ingredients are required. — PASTA

Dependency Resolution

* There's dependency b/w each and every ingredient
* Also I have limited space in my kitchen.
* Food Truck
  - has all the ingredients
  - specific dishes
  - exact cooking setup
  - from one food truck you can × 100 food trucks

Dependencies — Application.

(container) docker

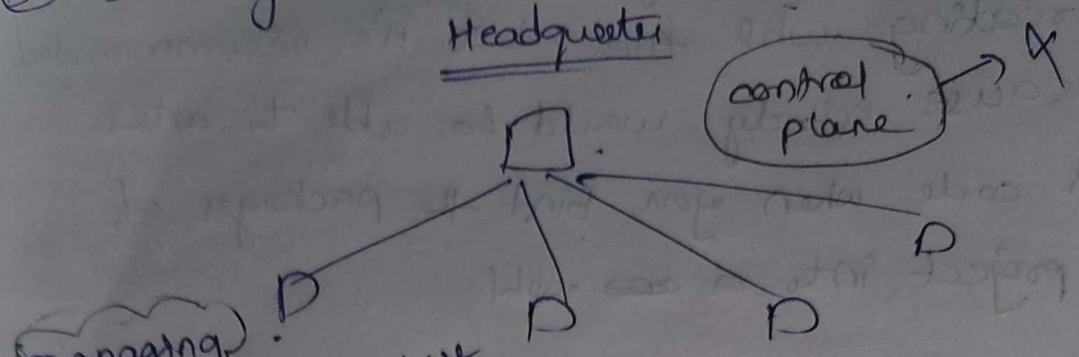


chapter 2 → business growth x100 food trucks → self healing

- each. Truck parked. schedule parking
- every truck operating: scaling — exo cylinder gone out. add'nal shld be there
- Customer finds you (service discovery)

Chapter

③ Building Restarant management system.



* request goes to API server

① * planning department [API server] → It receives orders, validates, processes and sends to right dept.

② * Record keeping [etcd]

details about each truck license, location, menu, staff, inventory.

* If control plane crashes it allows quick recovery (etcd, scheduler)

③ * Scheduling [scheduler]

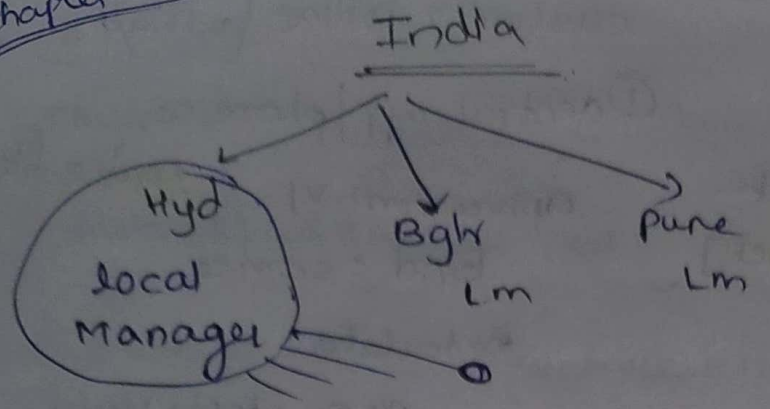
kharadi → where to park based on schedule location. (where do yo host)

Trucks organised / scheduled properly

④ Operations team [control management]

* monitor status / Report order / Replace
* continuously check.

chapter - 4



* Storage Volumes for ingredients

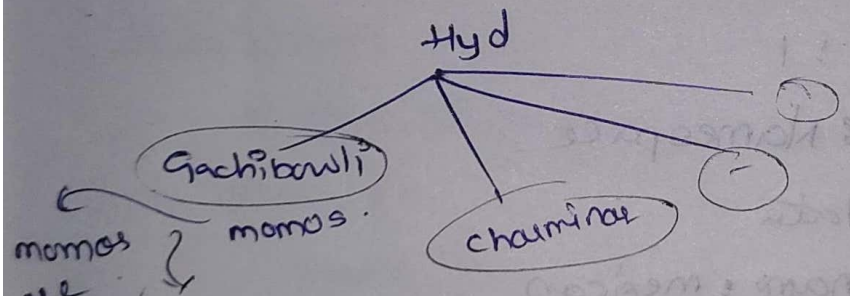
* All the locations are Nodes & Local manager is kubelet (It ensures right no. of pods are running) → request replacement in breaks down

* 10 centers each city Traffic manger b/w office and truck
↓
[Kube proxy] → route req automatic

* container Runtime → person who looks after running each food truck.

* kubectl is command line tool, smart phone based (used to communicate with control panel)

① Zomato [kubernetes Resource].



momos are all being made here

these districts are called namespace

② food Truck [Pods]. (no. of pods).

District Manager (max no. of trucks at money making area)

Deployments file (containers) → if anything goes down manage

apiversion : apps/v1

kind : deployment.

metadata name : taco-truck.

config for pods

Recipie Book

common → [config maps] file

secret → [secret]

apiversion: v

kind: configmaps.

metadata:

: mexican-recipie

data:

salsa spice level: medium

corn-taquela: extra chilli

customer hotline [service/app]

① central app | phone no.

service file

apiversion: v1

kind: service

metadata

name: taco-service

Spec:

Selector:

cusion: mexican

port

- port 80

- target 8080.

Daily operations

Name space.

apiversion: 1

kind: Namespace

metadata

name: mexican.

Launch New food Truck

① Build a truck Image (Docker Image)

② Deployment yaml (

③ submit (kubectl apply) → to CP [deploy]

④ Headquarter dispatch the patch.

1) kubectl apply -f taco_deployment.yaml.

2) kubectl get pods.

3) kubectl describe pod $\xrightarrow{\text{name}}$

4) kubectl logs taco-truck-123 y. (view logs)

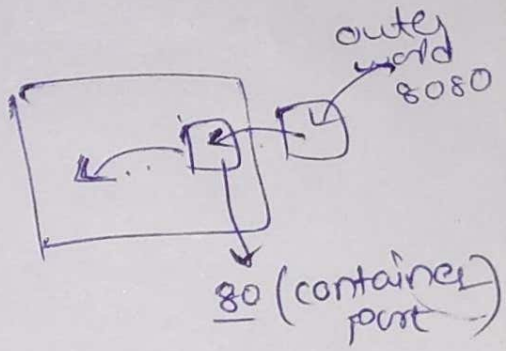
5) kubectl scale deployment taco-truck --replica =5 → need more nodes

6) kubectl set image deployment /taco-truck : v3.
(deploy new truck)

pods → smallest deployable unit in kubernetes

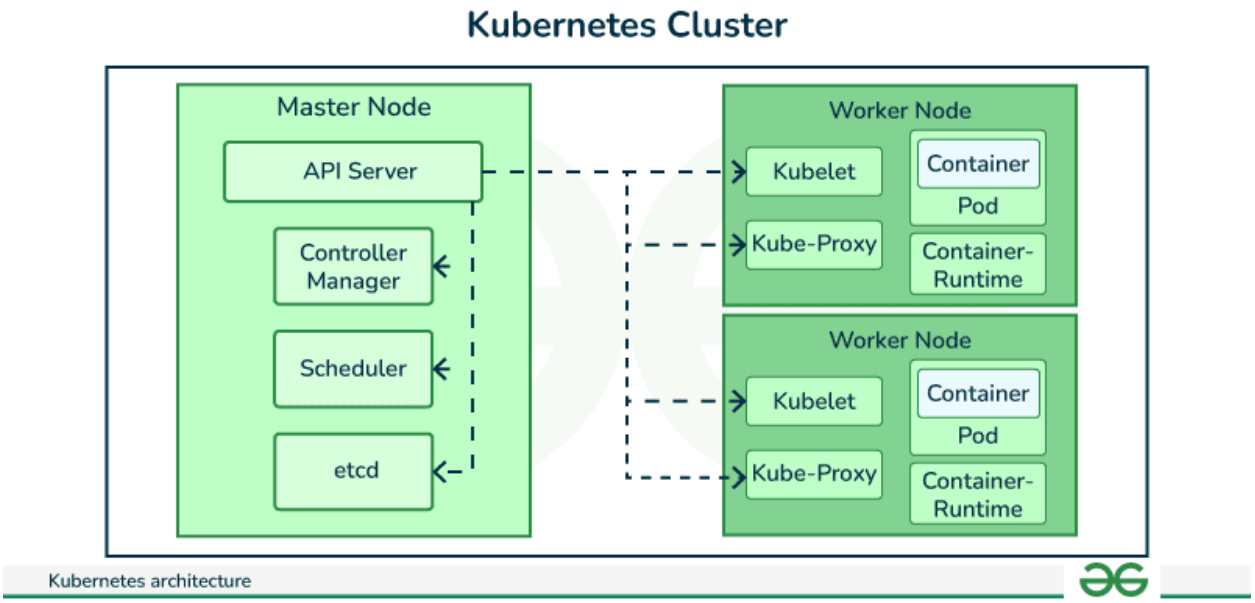
Ingress

* Directing customer from main street to your food truck.



container is not going to expose to outer world..

```
apiversion:
    kind : Ingress.
metadata:
    name:
spec:
    rules:
        host: food.example.com
        http:
            port: ———— /. com.
```



>Namespace(virtual space) is used to separate the deployment from other resources in the Kubernetes cluster

>Namespaces help organize Kubernetes resources.

- `v1` refers to **the first stable version** of the Kubernetes API.
- The **Namespace** resource is part of the **core API group**, which is stable and available under `v1`.

Example of Different API Versions

Different Kubernetes objects use different API versions. For example:

- **Pods, Services, ConfigMaps, and Namespaces** → `apiVersion: v1` (Core API)
- **Deployments** → `apiVersion: apps/v1`

service.yaml- exposes the Flask application to the outside world using a **NodePort**.

Minikube & Docker Setup Commands:-

minikube start
Starts the Minikube cluster, which is a local Kubernetes cluster for development and testing.

docker --version
Checks the installed Docker version to ensure Docker is available (Docker is often required by Minikube).

kubectl version --client
Displays the kubectl (Kubernetes CLI) version installed on your system.

Deployment Script Execution:-

chmod +x deploy.sh
Grants execute permissions to the deploy.sh script so it can be run as a program.

./deploy.sh
Runs the deploy.sh script, which contains commands to deploy your Flask application on Minikube.

Checking the Deployed Resources:-

kubectl get pods -n mini-demo
Lists all pods in the mini-demo namespace to check if the Flask app is running.

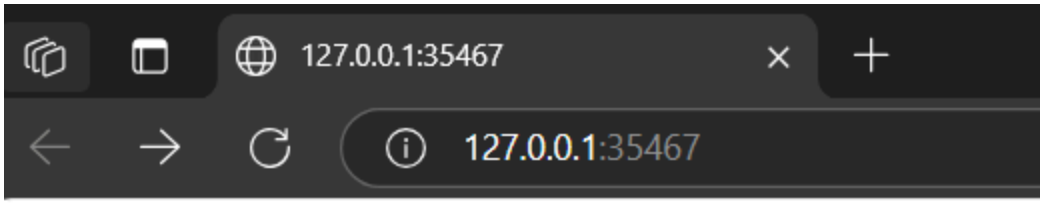
kubectl get svc -n mini-demo
Lists all services in the mini-demo namespace to find how the Flask app is exposed.

minikube service flask-app -n mini-demo --url
Retrieves the URL for accessing the flask-app service running inside Minikube.

kubectl -n mini-demo logs -l app=flask-app
Fetches logs of all pods labeled as app=flask-app in the mini-demo namespace to debug issues.

kubectl delete namespace mini-demo
Deletes the entire mini-demo namespace and all its resources.

kubectl get namespaces
Lists all available namespaces in the Kubernetes cluster



Kubernetes Mini Demo

App: Kubernetes Mini Demo v1.0.0

Hostname (Pod name): flask-app-5d5d6b89c4-svflk

Pod IP: 10.244.0.23

Request count: 1

Time: 2025-03-04 16:10:41

View API Info

Health Check