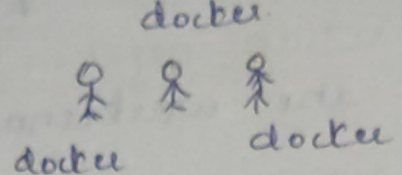
KUBERNETES

Kubernetes

- Google websites was crashing, lot of traffics to Autoscale & Autoheal kubernetes came into Existence
  ↳ First name: Borg.

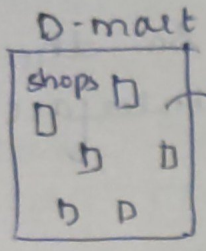
Ex: phir hera pheri.



⇒ Orchestration

* Why do we use kubernetes?

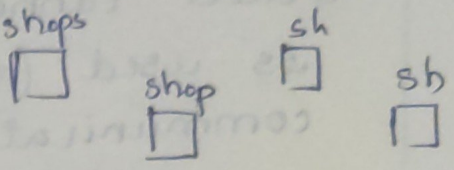
Monolithic



containers stored in one repository

- If anything crashes the entire system should be updated
- Huge cost, management problem

Microservices

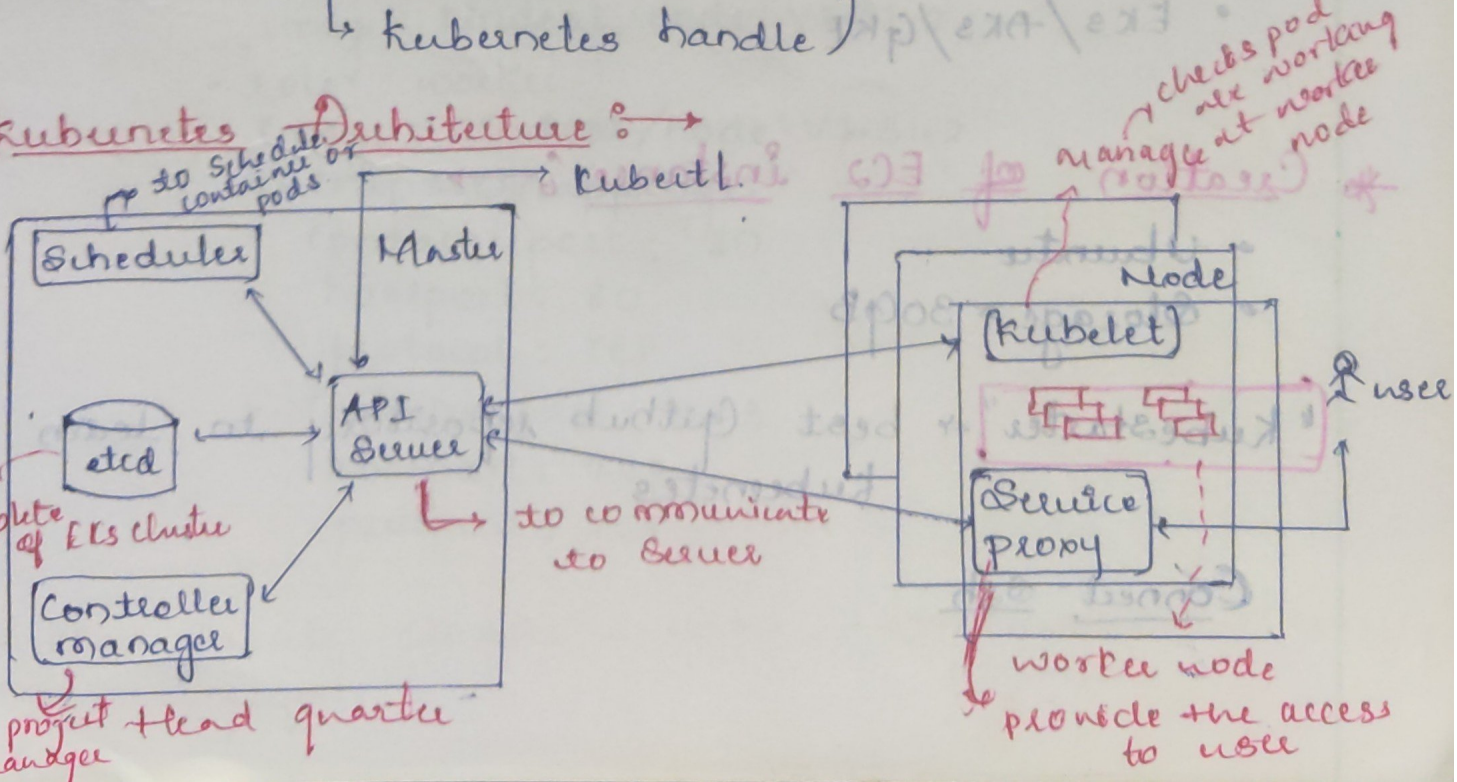


work independently
low cost
managing is easier.

- Many companies use microservices ⇒ kubernetes (demand)

1 node = Server
Multinode = Cluster (Multiple Server)
↳ kubernetes handle

* Kubernetes Architecture :→



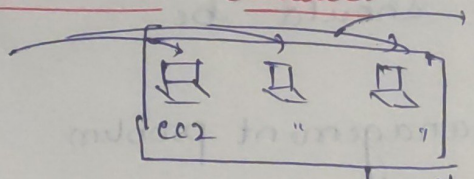
- Cognizant main branch is US - Headquarter (Master)
  Sub branches pune, bangalore, hyderabad., Noida
  Nodes

Important

- Docker runs on the workernode → Masters make them to work
- API server communicate with the workernode
  - Server cannot communicate directly so the API is used (Application programmable interface) to communicate.

* Creation of Kubernetes Cluster

- Kubeadm



- Minikube (local/ec2)
- KIND cluster
  ↳ kubernetes in docker
- EKS/AKS/GKE

* Creation of EC2 instance →

- Ubuntu
- Storage = 30GB

"kubestarter" → best Github repository to learn kubernetes

Connect ssh

* Kind cluster for Macos & Linux

After installation of docker

docker ps => user denied

To give permission

Command = `sudo usermod -aG docker $USER && newgrp docker`

docker ps

* Minikube for windows installation.

ubuntu = install kind
kubectl
docker

* mkdir kind-cluster
cd kind-cluster
vim config.yml

key value

```
kind: cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
- role: control-plane
  image: kindest/node:v1.31.2
- role: worker
  image: kindest/node:v1.31.2
- role: worker
  image: kindest/node:v1.31.2
- role: worker
  image: kindest/node:v1.31.2
  extraPortMappings:
  - containerPort: 80
    hostPort: 80
    protocol: TCP
  - containerPort: 443
    hostPort: 443
    protocol: TCP
```

list

* kind create cluster --name tws-cluster --config =config.yml.

Minikube installation Guide for Ubuntu

1. update system packages

```
Sudo apt update
```

2. Install required packages

```
Sudo apt install -y curl wget apt-transport-https
```

3. Install docker

```
Sudo apt install -y docker.io
```

Start & Enable docker

```
Sudo Systemctl enable --now docker
```

Add current user to docker group

```
Sudo usermod -aG docker $USER && newgrp docker
```

4. Install minikube

```
curl -LO minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64
```

```
chmod +x minikube
```

```
Sudo mv minikube /usr/local/bin/
```

5. Install Kubectl

```
curl -LO "https://dl.k8s.io/release/$(curl -L -s https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"
```

Check above image Executable & move it into your path

```
chmod +x kubectl
```

```
Sudo mv kubectl /usr/local/bin/
```

6. Start Minikube

```
minikube start --driver=docker --vm=true
```

7. Check Cluster Status

```
minikube Status
```

↳ uses containers in the background & create Clusters

* **Kubeadm** :→

Master → Worker

↳ connected 6443-port

1 Master → Multiple Workers also can be connected

kubeadm init

↳ it becomes master

Create EC2 instances

One Master — connect SSh — 2 - terminals master & worker — install by using the commands = kubestarter

One Worker — same

* Both uses contained it creates the pre requirements of each

- Master
  - etcd
  - Scheduler
  - API
  - Manager
- Worker
  - nodes
  - Kubelet
  - Service proxy

gets installed ← kubeadm init

- If In this whenever I do kubeadm init it becomes master

* kubeadm token create --print-join-command

↳ this command removes kubeadm, this is beneficial for worker node

↳ no need of kubeadm

- If in case I did "kubeadm init" at worker node this command helps to recover that mistake.

* kubectl get nodes

* **Pods :→**

Namespaces

→ Pod → deployment → Service →

Nginx/ and other like mysql

. This is the process in kubernetes how user access it

Name Spaces [mysql " " "]

↳ Same as groups ⇒ they are isolated the upper pod donot disturb to another pod.

* mkdir kubernetes

cd "

minikube start --cpus 2

- Minikube installation
  .vsl config

kubectl get namespace

Or

" " ns

pod = nginx

kubectl create ns nginx

kubectl get ns

kubectl run nginx --image = nginx

kubectl get pods

kubectl delete nginx

" " ns

* **Kubernetes pods:→**
pods are kubernetes objects that are the basic unit for running our containers inside our kubernetes cluster.

* **Kubernetes Namespaces:→**
namespaces provide a mechanism for isolating groups of resources within a single cluster.

- mkdir nginx
- cd "
- vim ~~nginx.yml~~ namespace.

yml = manifest file

```
kind : Namespace
apiVersion: v1
metadata:
    name: nginx
```

kubectl apply -f namespace.yml.

vim pod.yml

```
kind: pod
apiVersion: v1
metadata:
    name: nginx-pod
    namespace: nginx
Spec:
  Containers:
  - name: nginx
    image: nginx:latest
    ports:
    - containerPort: 80
```

Kubectl apply -f pod.yml

Kubectl get pods -n nginx

Kubectl exec -it pod/nginx-pod -n nginx --(space) bash

+ To ensure that the pod is running correctly or not

```
Kubectl describe pod/nginx-pod -n nginx
```

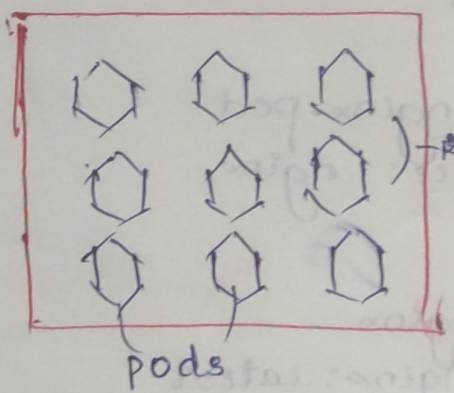
↳ provides complete information

```
Kubectl exec -it nginx-pod -n nginx -- bash
exit
```

- Vim deployment.yml ⇒ search on chrome deployment kubernetes

```
kind: Deployment
apiVersion: apps/v1
metadata:
    name: nginx-deployment
    namespace: nginx
spec:
    replica: 2
    selector:
        matchLabels:
            app: nginx
    template:
        metadata:
            labels:
                app: nginx
```

Replicaset / Statefulset / Deployment



↳ It maintains replica

Ex: On Bigbillion days Amazon gets lot of traffic & the amazon has only 3-pods now. it will create the replicas to store control traffic

Only 1 or 2 pods cannot handle the traffic so the replicas are created

- pod contains replication controller - which controls the replicas of pods

* **Replica Set** = kubernetes resource that ensures a specified number of identical pod replicas are running at any gave time.

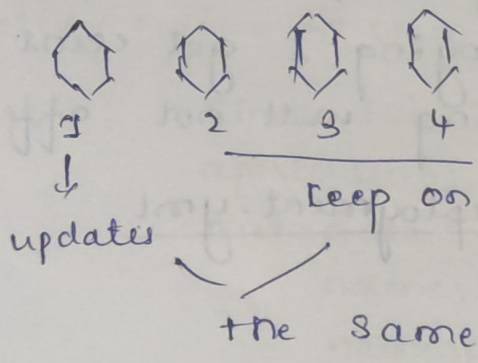
  Ex: I created a template
  Specified nginx = 4 => it create 4 replicas

* **Statefulset** = It mainly keeps the record of pods like running or deleted
  It provides the numbers to each pod to maintain the sequence

* **Deployment** = It mainly handles the rollback or rolling updates

  Ex: The image present in pods, I changed them it gets changed to all other pods by due to replica

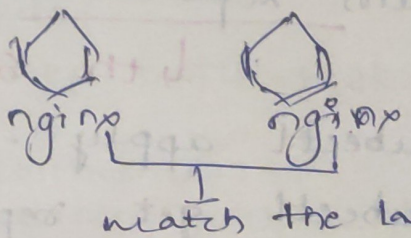
  - In this process we get downtime, System doesn't work



* **Labels & Selectors** :→

  Labels → Pod → I named it as nginx Label.

  Selectors → match & manage specific pods based on their labels



↳ no need bz I have mentioned its works in deployment.yml

- kubectl get pods -n nginx
- kubectl delete -f pod.yml

- kubectl apply -f deployment.yml
- kubectl get deployment -n nginx
  " " pods -n "
- kubectl scale deployment/nginx-deployment -n nginx --replicas=5
  ↓ 5 will be created in 'n' number "
- kubectl get pods -n nginx
- kubectl set image deployment/nginx-deployment -n nginx=nginx:1.26.3
- "

Deployment (rollingupdates) =>
Eg: while deploying I get error, the pods which r running will not affect

- kubectl delete -f deployment.yml

cp deployment.yml replicasets.yml
↳ copy the same ymlfile to replica file.

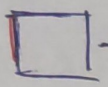
ls
vim replicasets.yml
↳ the same file is created as deployment.
kubectl apply -f replicasets.yml
kubectl get replicasets -n nginx

Daemonsets :→

Ensures that ~~atleast~~ 1 replica or copy of pod should be running on ~~node atleast~~ each node

- vim daemonsets.yml
- cp replicaset.yml daemonsets.yml
- kubectl apply -f daemonsets.yml
- kubectl get pods -n nginx
- kubectl delete -f daemonsets.yml.

Jobs :→



Task → Job → Done

Container

Ensures that a pod runs to completion successfully

Ex: update Shutdown =
Only once run & stop

- Its a parallel process
  3-pods running parallely

/Kubernetes/nginx # vim job.yml

```
kind: Job
apiVersion: batch/v1
metadata:
    name: nginx-job
    namespace: nginx
Spec:
    completions: 1
    parallelism: 1
    template:
        metadata:
         name: demo-job-pod
         labels:
            app: batch-task
        Spec:
          containers:
          - name: batch-container
          Image: busybox: latest
```

command: ["sh", "-c", "echo Hello Dosto! && sleep 10"]

restartpolicy: Never

* kubectl apply -f job.yml

* kubectl get job -n nginx

" " pods " "

running → Created → Stop

If you want back running → delete & start

* Cron job :→

Following particular Schedule

- Is used to run jobs on a Scheduled basis

vim cronjob.yml

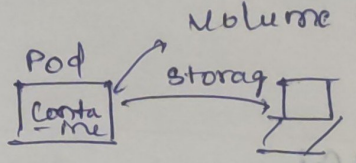
```
kind: CronJob
apiVersion: batch/v1
metadata:
    name: minute-backup
    namespace: nginx
spec:
    schedule: "*/2 * * * *"
    jobTemplate:
        spec:
            template:
                metadata:
                    name: minute-backup
                labels:
                    app: minute-backup
```

* * * * *
min hr day day month
week

```
Spec:
  containers:
  - name: backup-container
    image: busybox
    command:
    - sh
    - -c
    - >
      echo "Backup started";
      mkdir -p /backups &&
      mkdir -p /data &&
      cp -r /demo-data /backups
```

* Volumes :→

provide persistant storage to containers inside pods



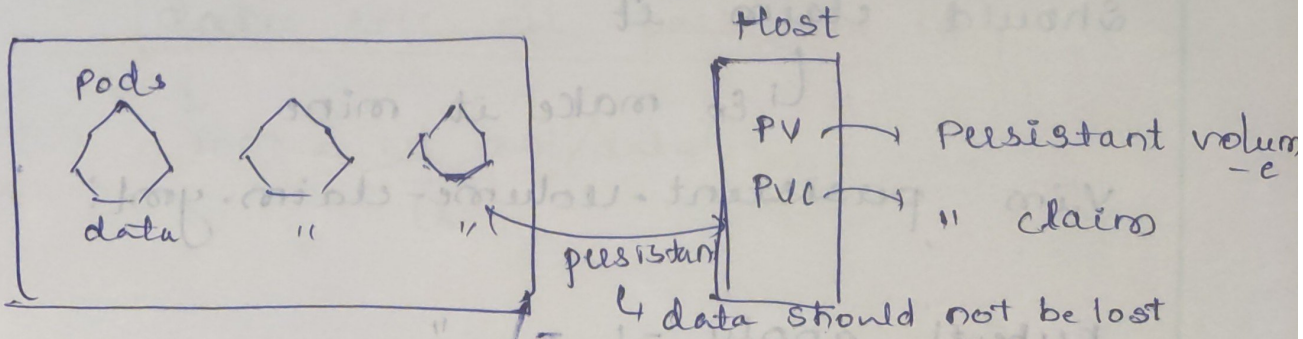
for Cronjob

```
kubectl get pods -n nginx -w   → Every sec
```

" Enter

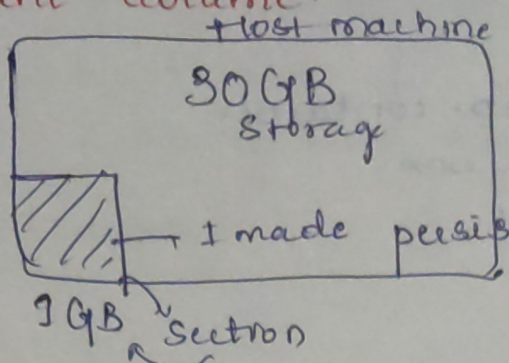
```
kubectl delete -f cron-job
```

* Storage :→



- The pods running on nodes if it gets crashed or deleted so there should be persistant storage =>

persistant volume



Vim persistent-volume.yml

```
kind: persistentVolume
apiVersion: v1
metadata:
    name: local-pv
    labels:
        app: local
spec:
    capacity:
      Storage: 1Gi
    accessModes:
      - ReadWriteOnce
    persistentVolumeReclaimPolicy: Retain
      StorageClassname
```

* kubectl apply -f persistent-volume.yml

* persistant Volume claim :→

Now I have persistant volume but I should claim it

↳ Ex: make it mine

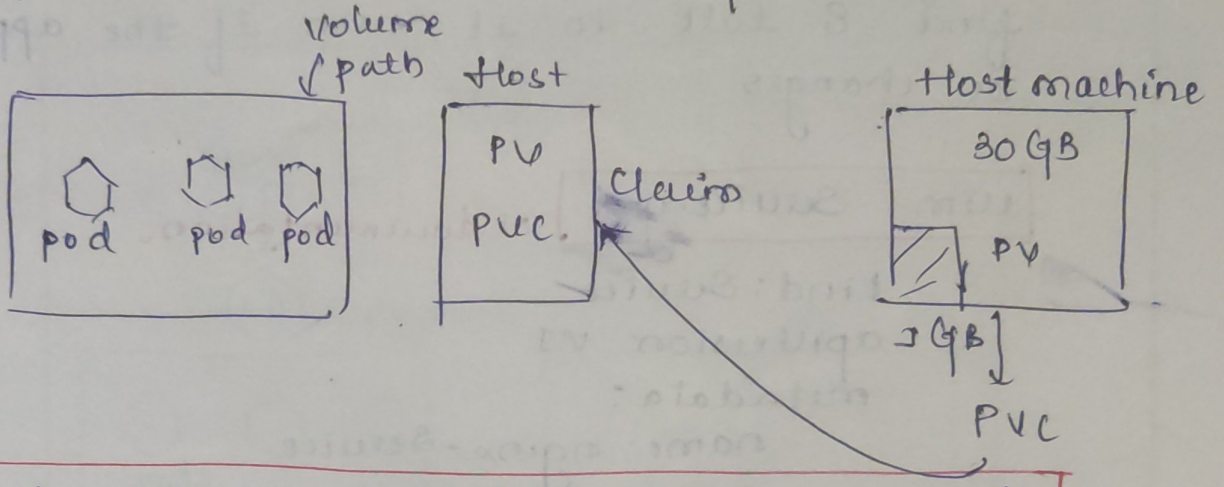
Vim persistent-volume-claim.yml

kubectl apply -f "

kubectl get pvc

```
volumeMounts:
  - mountPath: /var/www/html
    name: local-pvc
volumes:
  - name: local-pv
    persistentVolumeClaim:
      ClaimName: local-pvc
```

- local-path is bound with persistent-volume



- `kubectl get -f persistent-nginx -o wide`

↓

by this we can check the pod is working on which node.

- As I'm connected to EC2 the worker node is inside the docker

`docker ps`

↳ search for worker node

`docker exec -it` paste the id.

↳

root:~ cd mnt/data/

ls

helps to connect pods reliably
Even if pods come & go, the Service always
* **Service** :→ gives you a fixed way to reach them.

redirect through deployments

A service is a method for exposing a network application that is running as one or more pods in your cluster

Eg. It is like a helper that gives your app(pod) a fixed name & address so others can always find & talk to it, even if the app restarts or changes.

vim service.yaml → documentation.

```
kind: Service
apiVersion: v1
metadata:
    name: nginx-service
    namespace: nginx
Spec:
    selector:
      app: nginx
    port:
      - protocol: TCP
        port: 80
        targetport: 80
```

kubectl apply -f service.yaml

kubectl get all.

[bz it is in docker
it can't access by EC2]

kubectl port-forward service/nginx-service -n nginx 80:80 --address=0.0.0.0

sudo -E "

↳ if this command doesn't work write sudo -E & it will create environment

cd ..

Access the IP address on EC2

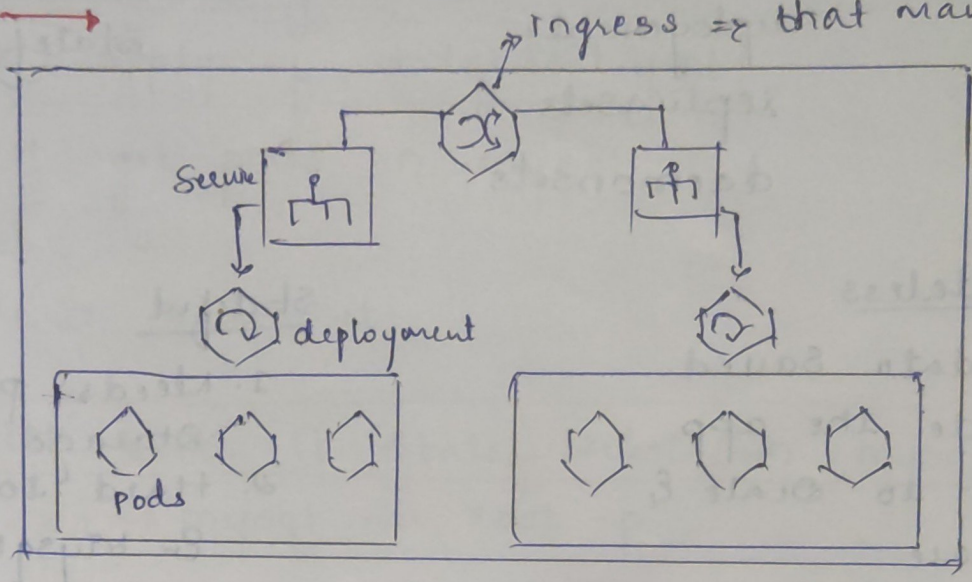
[git clone "paste"]

cd django-notes ↳ mini-project

- ls ↳ git check-out dev
- docker build -t notes-app-k8s

* docker-hub

* Ingress :→



- vim ingress.yml

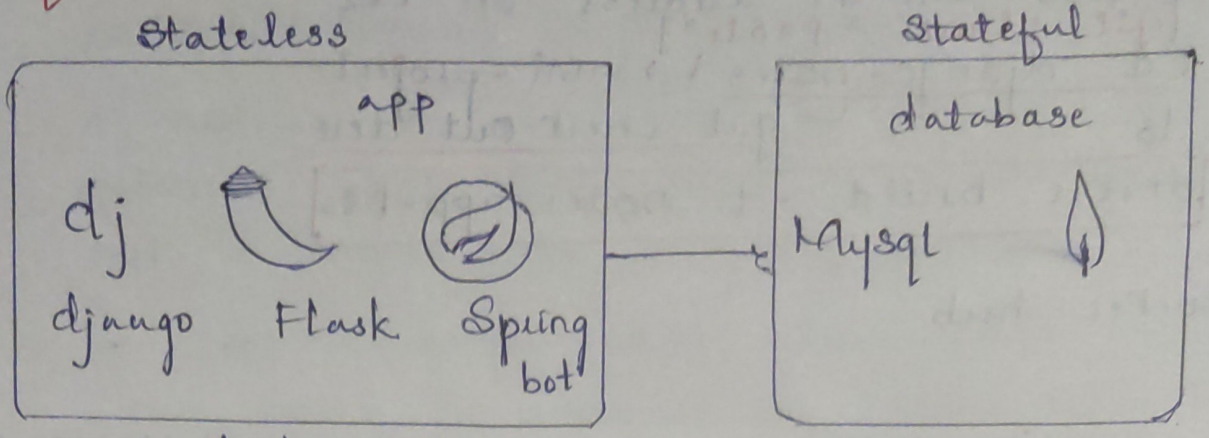
* Annotations :→

- They r like sticky notes attached to your kubernetes objects
- They store extra information that tools or humans need, but kubernetes itself doesn't use them to make decisions

* Cluster Networking :→

All the parts of your kubernetes System talk to each other - like pods, Services & nodes

* Statefulsets :



deployments
replicasets
daemonsets

Statefulsets

Stateless

1. No data saved inside the app
2. Easy to scale & replace
   Ex: web server, Api gateway

Stateful

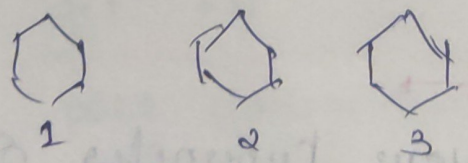
1. Needs persistant Storage
2. Hard to scale
   Ex: Mysql, MangoDB

* Statefulsets :⟶

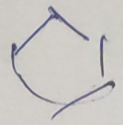
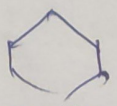
Controllers used to manage applicatio-ns that require a stable, unique identity for each pod & pusistent storage

For I have 3-pods

Benefit



4 if this deletes it will create replica which will be the same as before

 = data deleted :  the same data will get replicated.

replica

Mysql port = 3306
Nginx = 80

cd Kubernetes / cd Mysql =

Kubernetes / Mysql = vim statefulsets.yaml (documentation)

```
Kind: Statefulset
apiVersion: apps/v1
metadata:
    name: mysql-statefulset
    namespace: mysql
```

vim namespace.yaml

vim service.yaml

- kubectl apply -f statefulset.yaml
- kubectl get pods -n mysql
- watch "
  - ↳ 0 Statefulset
  - 1
  - 2 "
- kubectl exec -it statefulset-0 -n mysql -- bash
  - @root => mysql -u root -p
  - > mysql = show database
- kubectl delete pod mysql-statefulset-0 -n mysql
  - ↳ now if I g delete
- kubectl get pods -n mysql
  - ↳ deleted by now it creates the same nothing change
    then Statefulset comes into picture

* Configmaps :→

An API Object that lets you store configuration for other Objects to use.

- It stores configuration data (like urls, names, Environment Settings) Separately from the App code.

Ex: I have created Statefulset.yaml.
here I have declared configuration.
database: devops

If I want to change I should do it manually.

So, to not do it manually we use configmaps

Vim Configmap.yaml

kind: ConfigMap

kubectl apply -f configMap.yaml

Note: I have created statefulset.yaml file make it & dont change.

If any changes are required configmap through can be done.

Ex: • Keep your app & its settings separate
• change settings without touching the app code
• Useful for non-sensitive info (secrets / passwords)

• kubectl get configmap -n mysql

↳ to use secretfiles in binary

Secrets:→

A secret is an object that contains a small amount of sensitive data such as a password, a token, or a key.

Vim Secrets.yaml

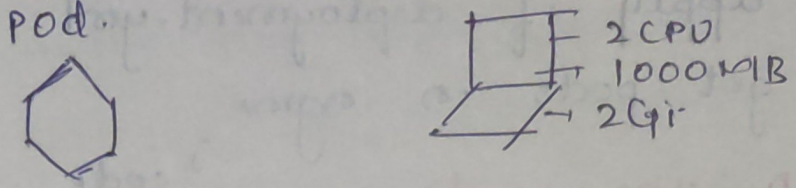
kind: Secrets

kubectl apply -f Secrets.yaml

kubectl get secret -n mysql

Scaling & Scheduling :→

Resource Quotas & limits :→



My laptop limit is 2 CPC, 2 GB, but this only 1-pod takes everything there will be nothing kept for another pod.

- In such cases we create the limits

```
vim deployment.yml
resources:
    request:
        cpu: 100m
        memory: 128MB
    limits:
        cpu: 200m
        memory: 256Mi
```

(request)

Probes :→

Determine if a container is functioning correctly, whether it should be restarted or if its ready to serve traffic.

Types :→

- liveness probe = container is live/not
- readiness " = ready to receive traffic
- Startup " = is it startingup correctly

vim deployment.yml

```
- containerPort: 8000
  livenessProbe:
    httpGet:
      path: /
      port: 8000
```

Same readiness Startup

- Kubectl apply -f service.yml
- Kubectl apply -f deployment.yml
- Kubectl get pods -n nginx

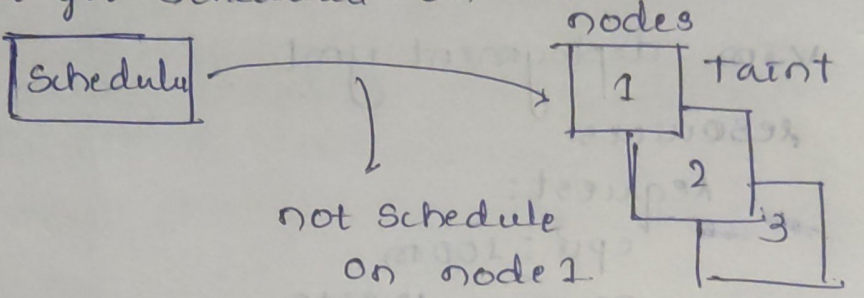
↳ code

To check the readiness, liveliness

Kubectl describe paste

Taints & Tolerations :⟶

↳ pod not get Scheduled on the node we select



- Taints are applied to nodes in a kubernetes cluster to repel from being Scheduled on those nodes.

- Tolerations :⟶

It allow a pod to run on a node with a taint

Kubernetes /nginx :- ls

↳ pod.yml

Vim pod.yml

Kubectl get pods -n nginx

Kubectl delete ns nginx

- Kubectl get nodes

↳ it displays the nodes

- Kubectl taint node paste prod=true:NoSche-dule

worker node

" = " 2

3

paste

- kubectl taint node tws-cluster-worker2 prod-true: NoSchedule-

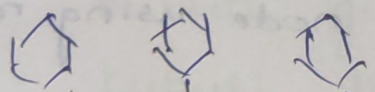
- kubectl describe node -n nginx → untainted

* How to make run tainted pods.

```
Vim pod.yml
Spec:
  tolerations:
    - key: "prod"
      Operator: "Equal"
      value: "true"
      effect: "NoSchedule"
kubectl apply -f pod.yml
"  get pods -n nginx
     to
   running
```

* HPA (Horizontal pod Autoscaling) & VPA (Vertical pod Autoscaling) — Kubernetes Event driven Autoscaling → KEDA

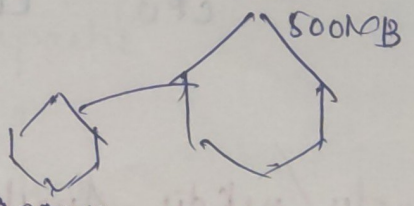
HPA



create replicas

- Automatically Scales the no. of pod replicas based on CPU/memory utilization

VPA



If the pod has 100MB limits upto 200MB if it Exceeds make the Size of the pod larger.

- Automatically adjusts the CPU & memory requests / limits for containers in a pod.

KEDA

Metrics

CPU, Storage using

Open-source component that allows you to Scale Kubernetes workloads based on event-driven metrics

- minikube addons enable metrics-server
- kubectl apply -f https://....
  - copy from kubestarter
  - install metrics server
- Edit the metrics server deployment
  kubectl -n kube-system edit deployment metrics-server
- Add the security bypass to deployment under containers.args
  - --kubelet-insecure-tls
  - --kubelet-preferred-address-types=InternalIP, Hostname, ExternalIP
- Restart the deployment
  kubectl -n kube-system rollout restart deployment metrics-server
- Verify if the metrics server is running
  kubectl get pods -n kube-system
  kubectl top nodes
  - CPU
  - CPU%
  - displays node using more CPU

Kubuntu/mkdir Apache

cd ..

Vim namespace.yaml

kubectl apply -f namespace.yaml

Vim deployment.yaml

Kubectl apply -f .

* To give public access
  Vim service.yaml

vim hpa.yaml

```
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
    name: apache-hpa
    namespace: apache
```

* Vertical pod Autoscaler :→

Chrome = Kubernetes Autoscaler Github

/Kubernetes/apache = git clone paste

cd autoscaler/vertical-pod-Autosc
-aler

For installation

Github
↓
VPA
↓
Installation

Command c/p

vim vpa.yaml

```
kind: VerticalPodAutoscaler
apiVersion: autoscaling.k8s.io/v1
```

kubectl get pods -n apache

kubectl get hpa -n "

kubectl delete -f hpa.yaml

* Node Affinity :→

It is a feature in kubernetes that facilita-tes us to specify the rules for scheduling the pod based on the node labels.

Ex:→ If I'm thirsty, I need water, I atleast the water I needle the characteristics to drink water

* Cluster Administration :→

* RBAC (Role Base Access Control) :→

Role = bring Vegetables

role binding

User Vaishnav to bring

User Vaibhav to bring

* RBAC in kubernetes allows you to define roles & permissions for users, groups or service accounts.

RBAC → Access

Service account (Namespace level) Roles rolebinding

User account (cluster level) cluster role cluster role binding

```
cd kubernetes
cd apache
kubectl get ns
    kubectl auth whoami
        "    "  can-i get pods
    kubectl apply -f namespace.yml
    kubectl auth can-i get pods -n apache.
```

* If I write kubectl auth delete pods -n apache it can be deleted, the same can be done by other user. To access to the resources what I alot is RBAC

```
vim role.yml
    kind: Role
    apiVersion: rbac.authorization.k8s.io/v1
    metadata:
        name: apache-manager
        namespace: apache
    rules:
      - apiGroups: ["", "apps", "rbac.authorization.k8s
                                          .io"]
        resources: ["deployment", "pod", "Service"]
```

↳ I'm giving access to these all resources.

```
Kubectl apply -f role.yml
Kubectl get role -n apache
```

```
vim Service-account.yml
    kind: ServiceAccount
    apiVersion: v1
    metadata:
        name: apache-user
```

```
Kubectl apply -f service-account.yml
Kubectl get Serviceaccount -n apache
```

↳ apache user
default

```
Kubectl auth can-i get pods -n apache
                                    --as=apache-user
```

↳ No

Bz there is no binding we created Service role

↳ not bind

Vim role-bind.yml

```
Kind: Rolebinding
apiVersion: rbac.authorization.k8s
```

- Kubectl apply -f role-bind.yml
- Kubectl get rolebinding -n apache
- Kubectl auth can-i get pods -n apache --as=apache-user
  No
- Kubectl auth can-i get pods --as=apache-user -n apache
  → yes

- Vim role.yml
  ↳ make some changes = Don't allow deployment
- Kubectl apply -f role.yml
- Kubectl auth can-i get pods -- as=apache-user -n apache
- Kubectl auth can-i get deployment --as=apache-user -n apache
  → No

Vim role.yml

```
Kind: Role
apiVersion: rbac.authorization.k8s.io/v1
metadata:
    name: apache-manager
    namespace: apache
rules:
  - apiGroups: ["", "apps", rbac.authorization.k8s.io", "batch"]
    resources: ["pods", "deployments", "Services",]
    Verbs: ["get"] #get, apply, delete
```

It is the extension of kubernetes API

Custom Resource Definitions (CRDs)

Resources = pods, namespace, deployment

pod → resource

apiVersion = v1

↳ version

deployment = resource

apiVersion = app/v1

↳ group name

Vim customresourcedefinitions :→ crd.yml

```
kind: CustomResourceDefinitions
apiVersion: apiextensions.k8s.io/v1
metadata:
    name: devopsbatches
```

kubectl apply -f crd.yml

kubectl top pod -n apache

↳ displays which pod is taking more space

Helm :→

packages

Nginx	Mysql	Helm help to download these packages
deployments	"	
pods	"	
Services	"	
ingress		
configmap		

Helm: package manager for kubernetes, with your Amazon EKS cluster to manage & deploy applications seamlessly.

Document ⇒ download

Helm is in Go language

- helm create apache-helm
- ls
- cd apache-helm/
- ls
- tree
  - shows the tree what it has created

```
.
├── Chart.yaml
├── charts
└── templates
    ├── NOTES.txt
    ├── _helpers.tpl
    ├── deployment.yaml
    ├── hpa.yaml
    ├── ingress.yaml
    └── service.yaml
```