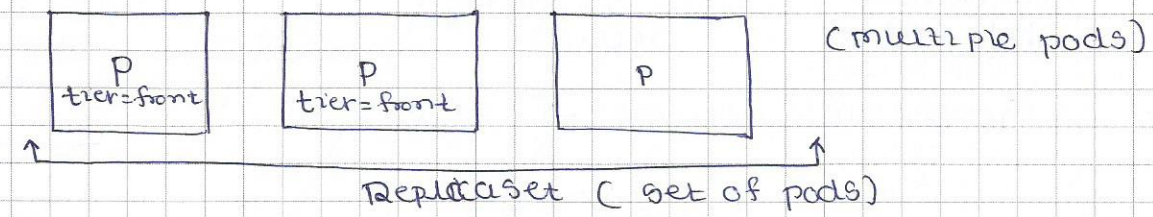
* Replicaset - multiple pods.



→ YML → 3 - Replicas of ignix (6 matching same condition)

I ignix ↑ always Run 3. (2

→ ~~cluster~~ one existing +2 more added.

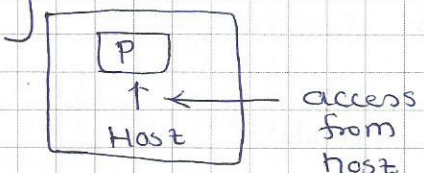
→ rs.yml (Replica set yml)

apiVersion: apps/v1 { kubectl explain rs (replicaset)
Kind: Replicaset
metadata:
Name: rs
spec: → {desired states}
replicas: (3) ← ~~mess~~ metrons.
Selector:
matchLabels: what pods. ← must match else.
tier: frontend (criterial)
template:
metadata:
Labels:
tier: frontend } (matching skills)
Pod
→ so and so label matches, join the team.
spec:
containers:
- name: apache
image: httpd

→ Services configuration

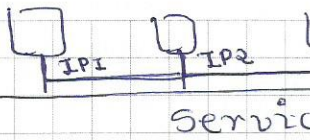
> kubectl create -f rs.yml
> kubectl get rs.
> kubectl get pods.
Name
my-rs-p7w9
RS-name Pode-id

{30-seconds to delete pods}



> kubectl get rs -o code
Name Desired current
3
> kubectl describe rs

* Publish Port outside (Kubernetes @ Services)
'Service



(2)

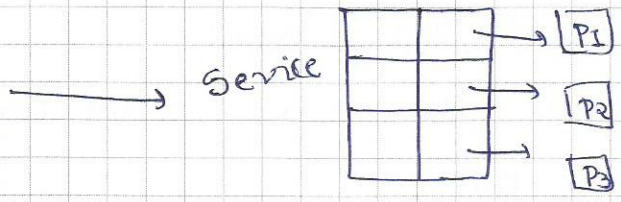
[Virtual Layers across your pods]

Service (Load Balancing access) — ServiceIP address — Load Balancing (internal)

(ClusterIP) → Request goes to all pods

Nodeport - external access (like -P) [external]

> Service IP address → (expose Port)



{expose port out-side}

apiversion: app/v1
Kind: Service
metadata
name:- → (3000 → start taking ports)
Spec:
type: NodePort (ext access to outside world)
selector:
tier: → where it should forward Request
↑ frontend → [selecting all the pods where service could make a request]

{target Port}
Ports:
- protocol: TCP
port: 80 → {where this service to connect. means which Port}

> kubectl explain Service
version: v1

> Service is created.

kubectl get svc.

Name	type	Cluster-IP	ExternalIP	Port	Age
f-svc	Nord	10.20.30.40 ↑ [Service-IP]	X (For Cloud)	80 : 3756 ↑ Port No. of machine. (machine IP addres)	For external coord

→ Create Service of type Nord port

target Port: 80 (where it sends Request)
External Port: 32765 (

* Deployments: (Next level of RS)

— tradition — nginx:v1 → 3 Pods

↓ update with → Remove Image
↳ Remove old Pods creating new Pods.

nginx:v2 → (3) v1

→ Remove existing deploy: v1
new deploy: v2] → [maintance time Frame] down-time window.

Q [Replace one by Replace: v2 — 10 seconds. ↳ slow slow

↳ Rolling & Rollback (possible) (
↳ more Feature to [RS]

* Example:
> vi deployment.yml

Deployment = Rolling update + RS

apiVersion: app
kind: Deployment
metadata:
name: kubeserve
Spec:
replicas: 3 } (internally use - RS)
Selector:
matchLabels:
app: ~~kubeservice~~ kubeserve (Label must be
template: ← For container part of pod)
name: kuber
labels:
app: ~~kuberservice~~, kubeserve

--record (why Recording)

Name Ready Up-to-date

> ↑ Annotation: _____ : (1) → {update will be version according Rolling}

Strategy Type: Rolling Update.
Rolling Update Type: 25% max unavail,

Image: _____ ← deployment

> kubectl rollout history deployment kubeserve.
Revision
1 — First deployment.

* kubectl Scale deployment kubeserve --replicas = (5)
→ Scaled. no. of Rep.

> kubectl expose deployment kubeserve --port 80 (forward it to)
public outside
--target-port (80) Service --type NodePort

80: 31623 | TCP
[externally accessible]

Pods (Ports) [Service (Ports)]

* kubectl patch deployment -p '{"spec":{"..."}}

* change or update version of Image.
→ Apply new version.

>
Rollout → Rollback

Apply Running Bus.

* always update .yml & apply. !(_)

> while true; do curl http://x47 echo "test"; done

> Set image (update application)
↳ deployment app = <image --v 6
{6 lines verbose}

> Rolling Deployment - {Slowing deployed}

→ History of deployment

kubectly rollout undo deployment kubeserve.
{Roll back to previous version}

> Rollout history of a deployment.

— to go back to specific version.

{you can go to any version}

> kubectly - - - - - - - -to-revision=1 { version no. 1}

> kubectl edit ← ↑ {our data in File}

↑ on the Fly editing deployment

* kubernetes Dashboard: