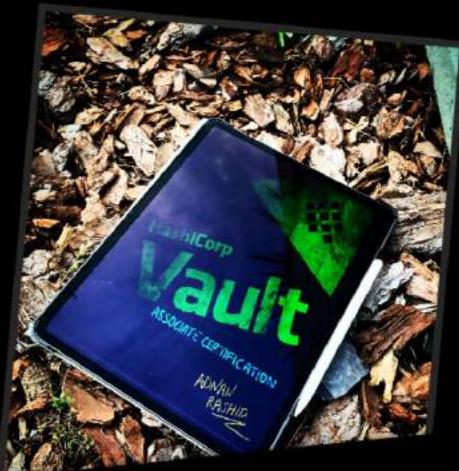


CLOUD NATIVE CERTIFIED

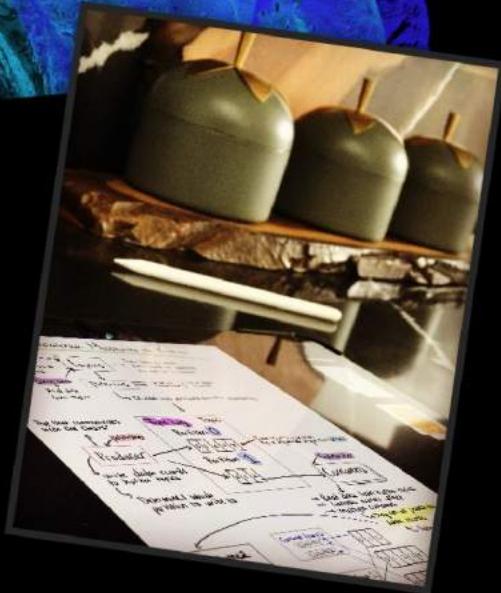


KUBERNETES ADMINISTRATOR

ADNAN RASHID

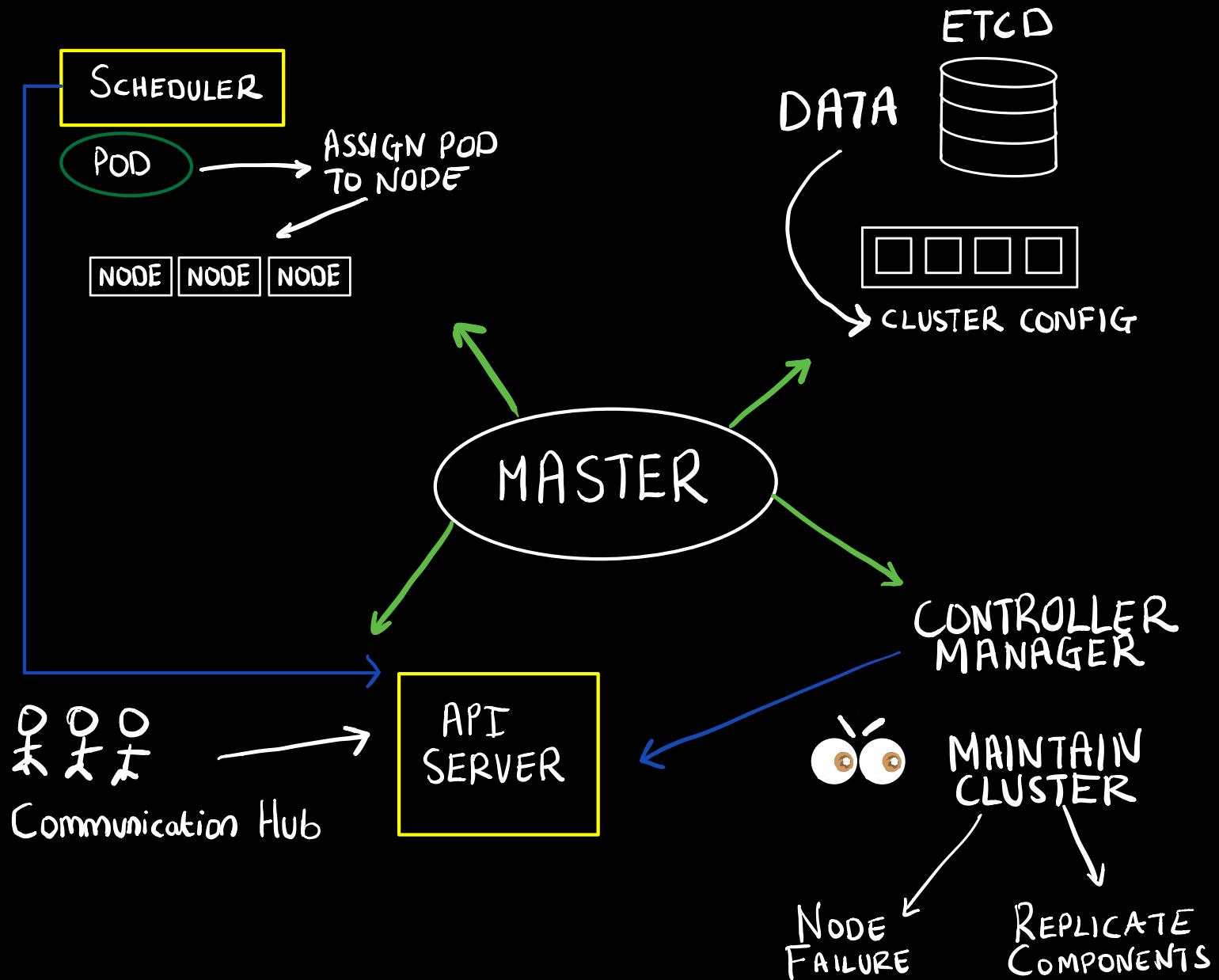


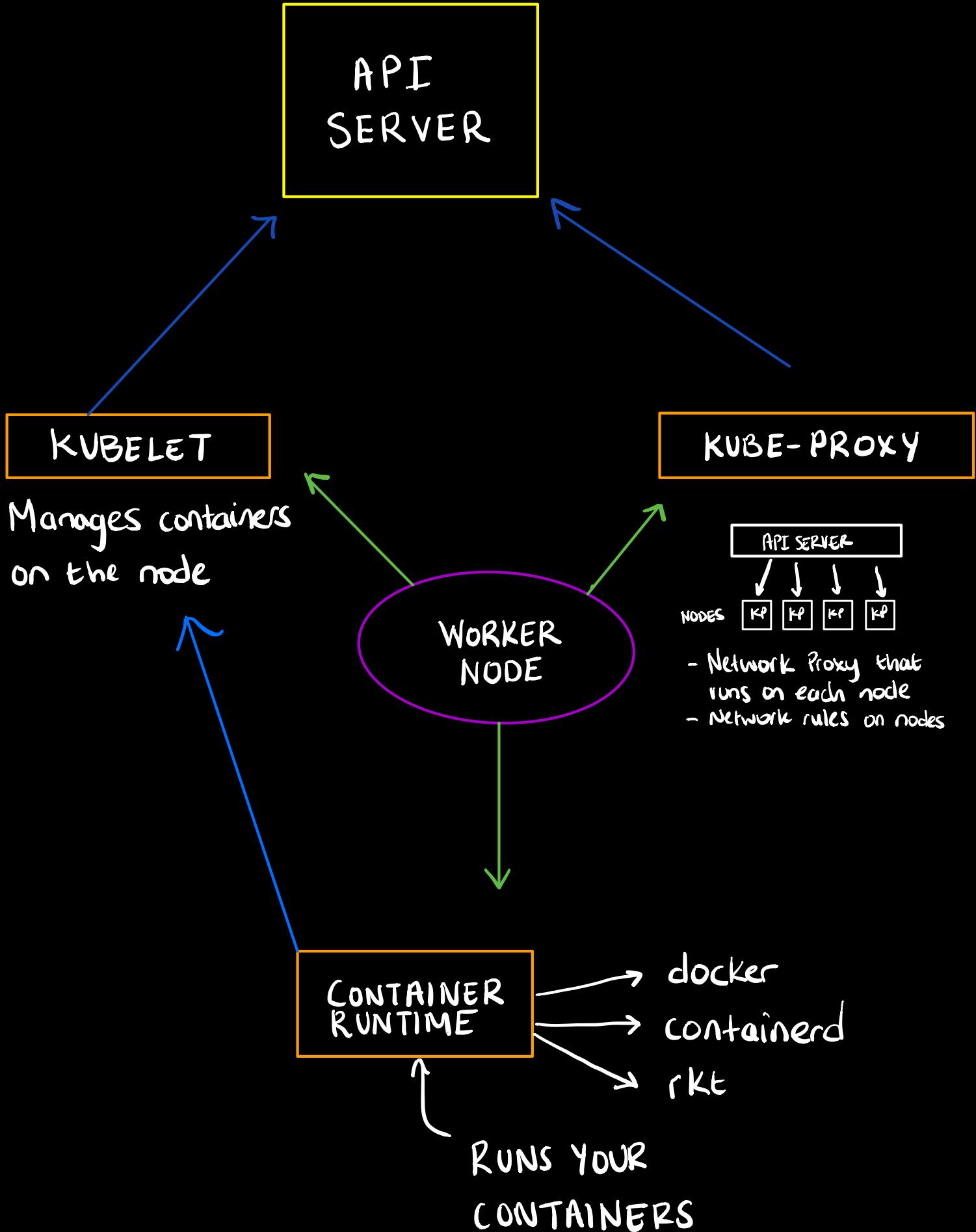
instagram.com/
adnans_techie_studies



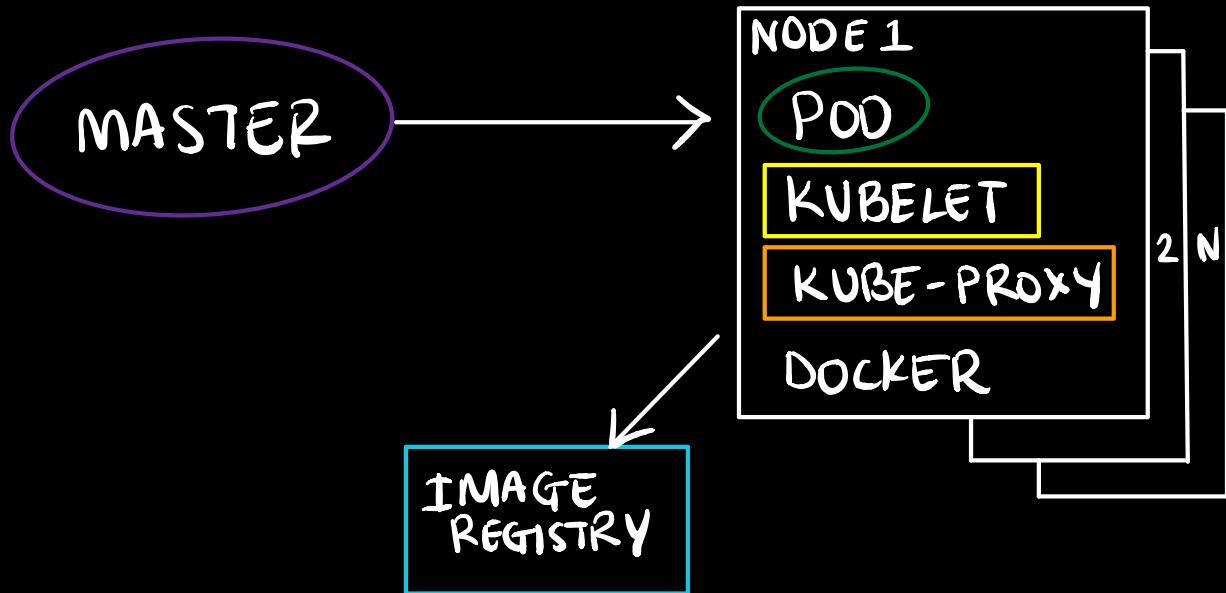
UNDERSTANDING THE KUBERNETES ARCHITECTURE

CLUSTER ARCHITECTURE



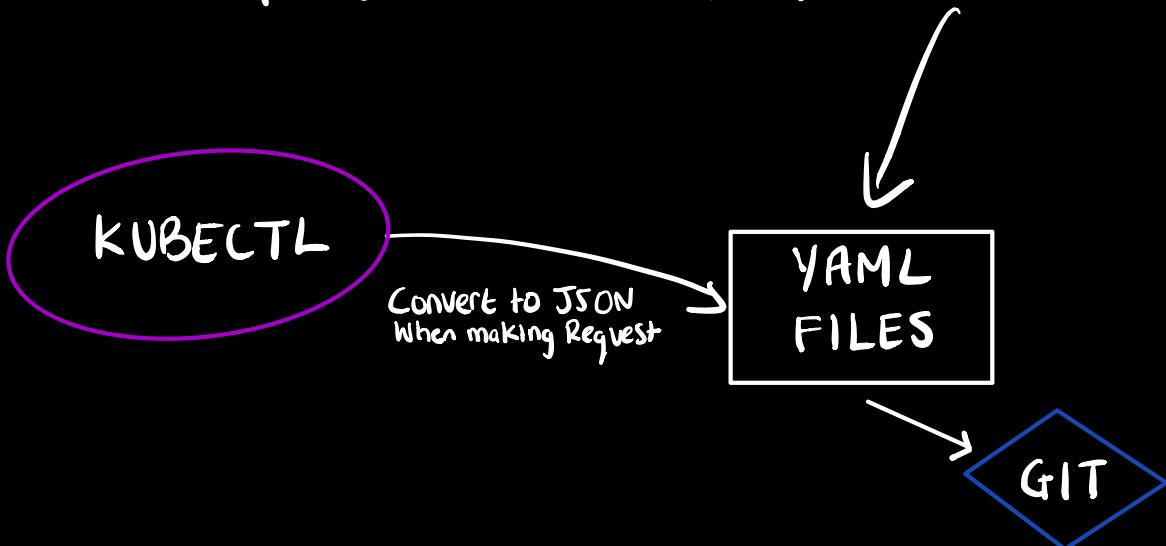


APPLICATION RUNNING ON K8S



API PRIMITIVES

API Server is the only one that communicates with etcd
Every single component speak with API server only not to each other
Objects like pods and services are declarative intents



YAML FILE COMPOSITION

API VERSION

→ clear consistent view
of resources

KIND

→ The kind of object you want to create

- Pod
- Deployment
- Job

METADATA

→ uniquely identify the object

Name
String

↓
UID

↓
namespace

SPEC

Container image
volume exposed ports

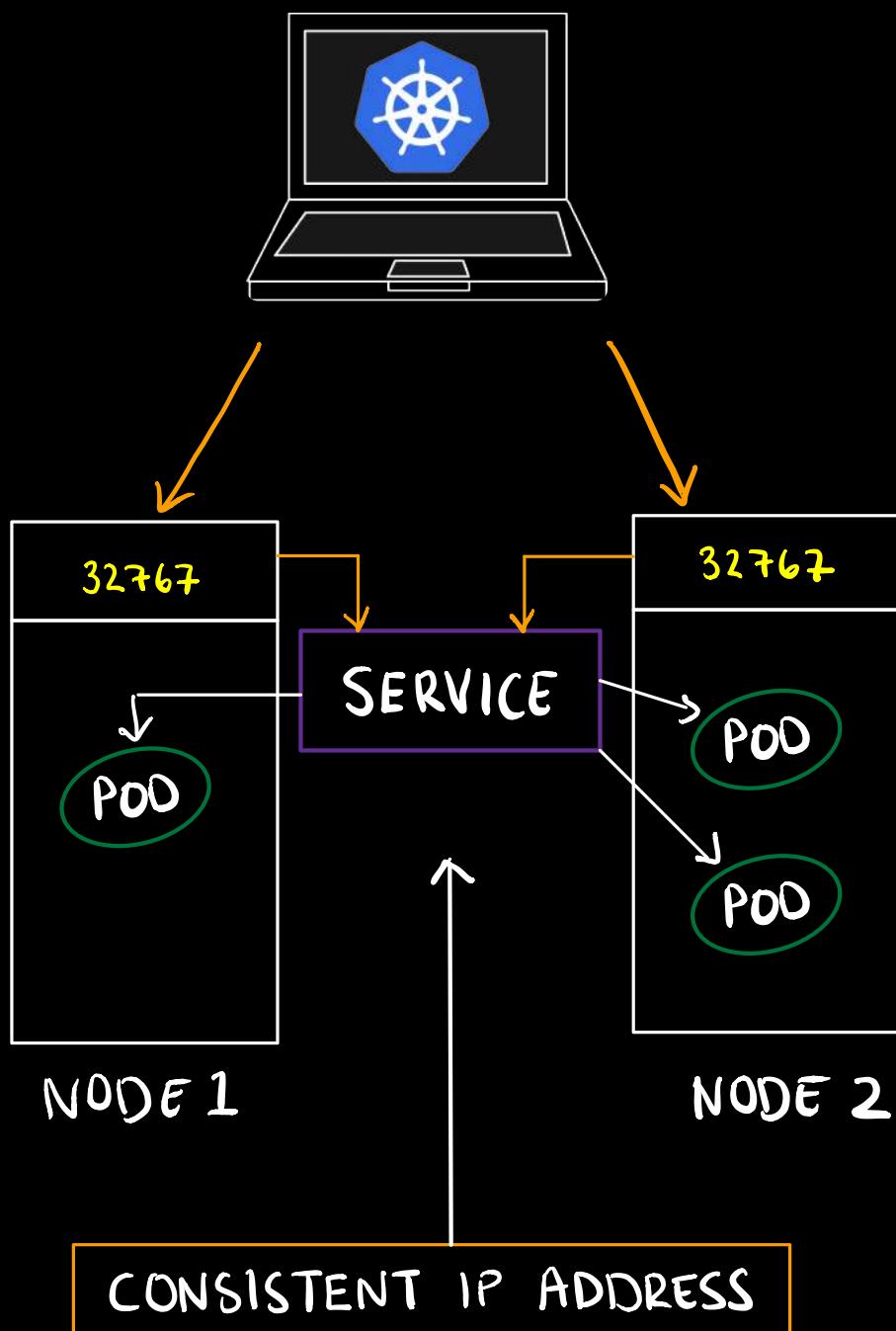
STATUS

→ State of the object

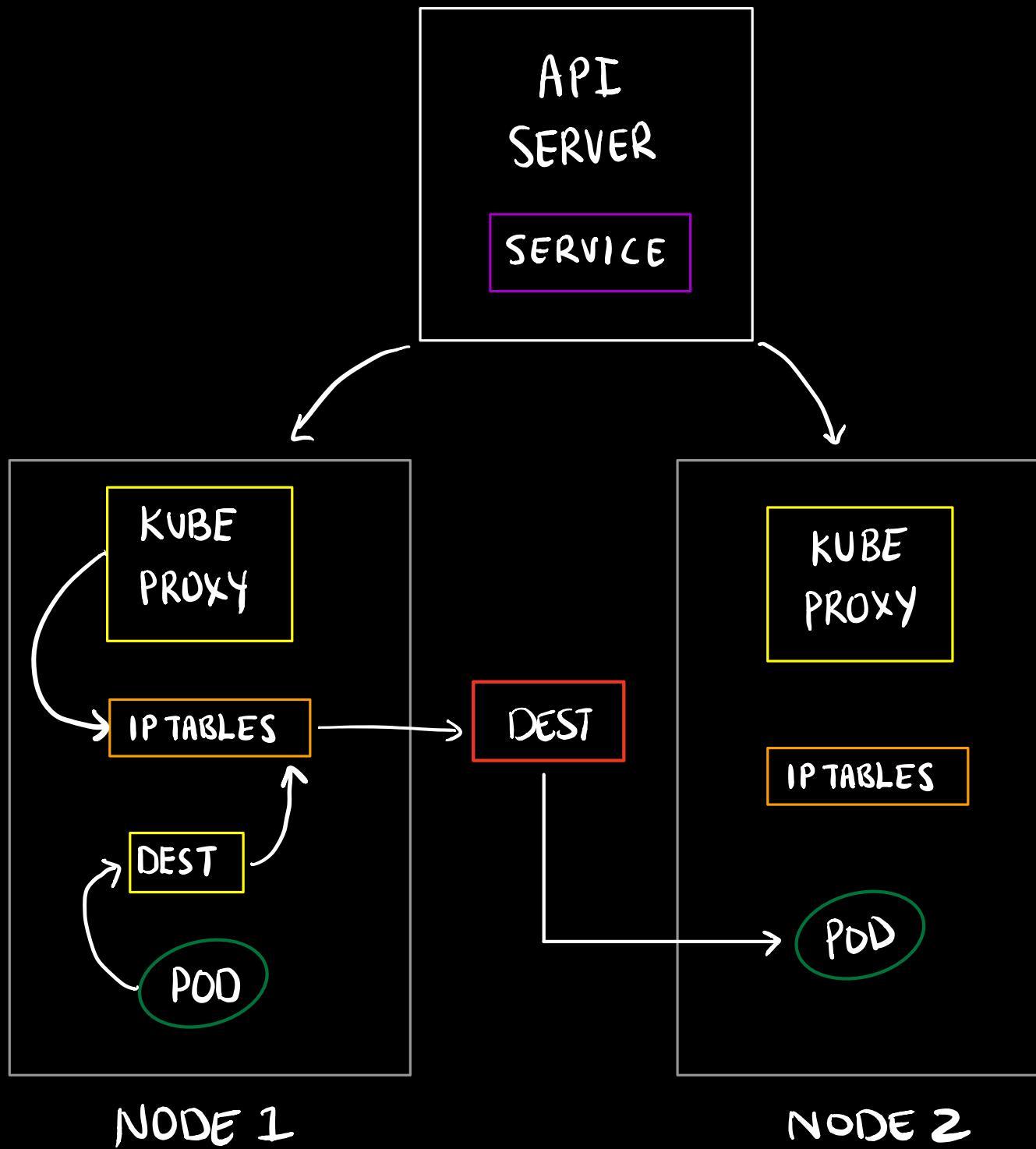
↓
match desired states

SERVICES AND NETWORK PRIMITIVES

SERVICES ALLOW YOU TO DYNAMICALLY
ACCESS A GROUP OF REPLICA PODS



KUBE-PROXY



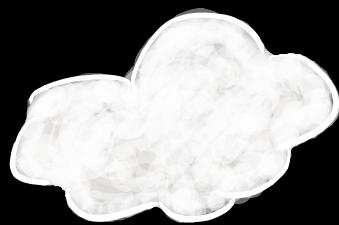
KUBE-PROXY HANDLES THE TRAFFIC ASSOCIATED WITH A SERVICE BY CREATING IP TABLE RULES



BUILDING THE KUBERNETES CLUSTER

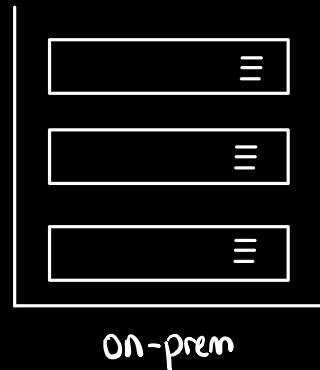
RELEASE BINARIES, PROVISIONING & TYPES OF CLUSTERS

Picking the right Solution



CLOUD

VS



ALSO

CUSTOM

- Install Manually
- Configure your own network fabric
- Locate the release binaries
- Build your own Images
- Secure cluster config

VS

Pre-Built

- Minikube
- Minishift
- Microk8s
- Ubuntu on LXD
- AWS, Azure, GCP

✓ using minikube locally on MacOS

INSTALLING KUBE MASTER AND NODES

MASTER + WORKERS

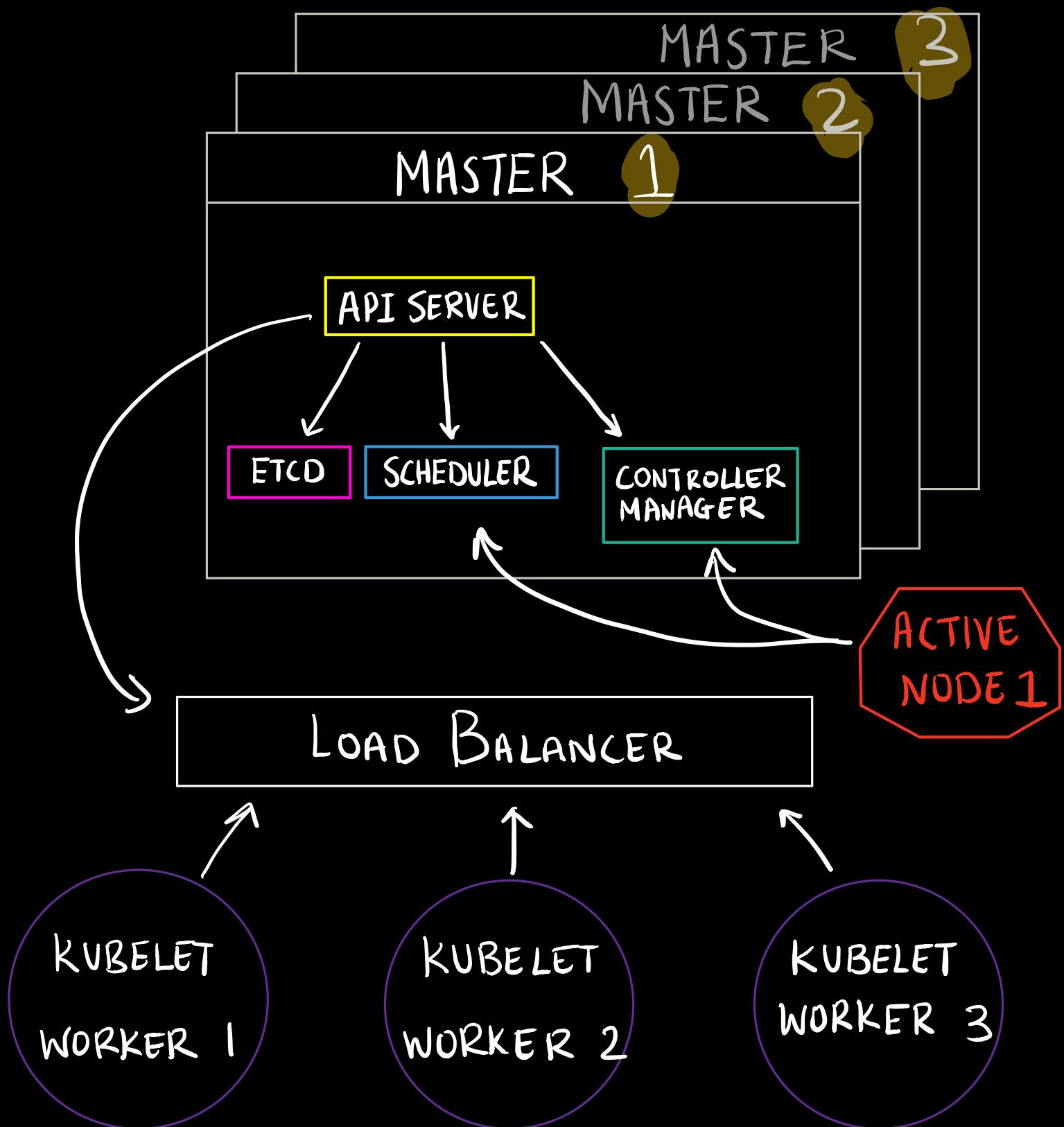
- ① DOCKER + KUBE → GPG KEY
→ ADD REPOS
- ② UPDATE PACKAGES
- ③ INSTALL Docker, kubelet, kubeadm, kubectl
- ④ Modify bridge adapter settings

MASTER ONLY

- ① Initialise Cluster
- ② Make directory for kss
- ③ Copy kube config
- ④ Change ownership of config
- ⑤ Apply Flannel CNI

BUILDING HIGHLY AVAILABLE CLUSTER

ALL components can be replicated, but only certain can operate simultaneously



The controller manager and the scheduler actively watch the cluster state and take action when it changes

SCHEDULER



CM



CLUSTER

ARE WE IN

CHARGE?

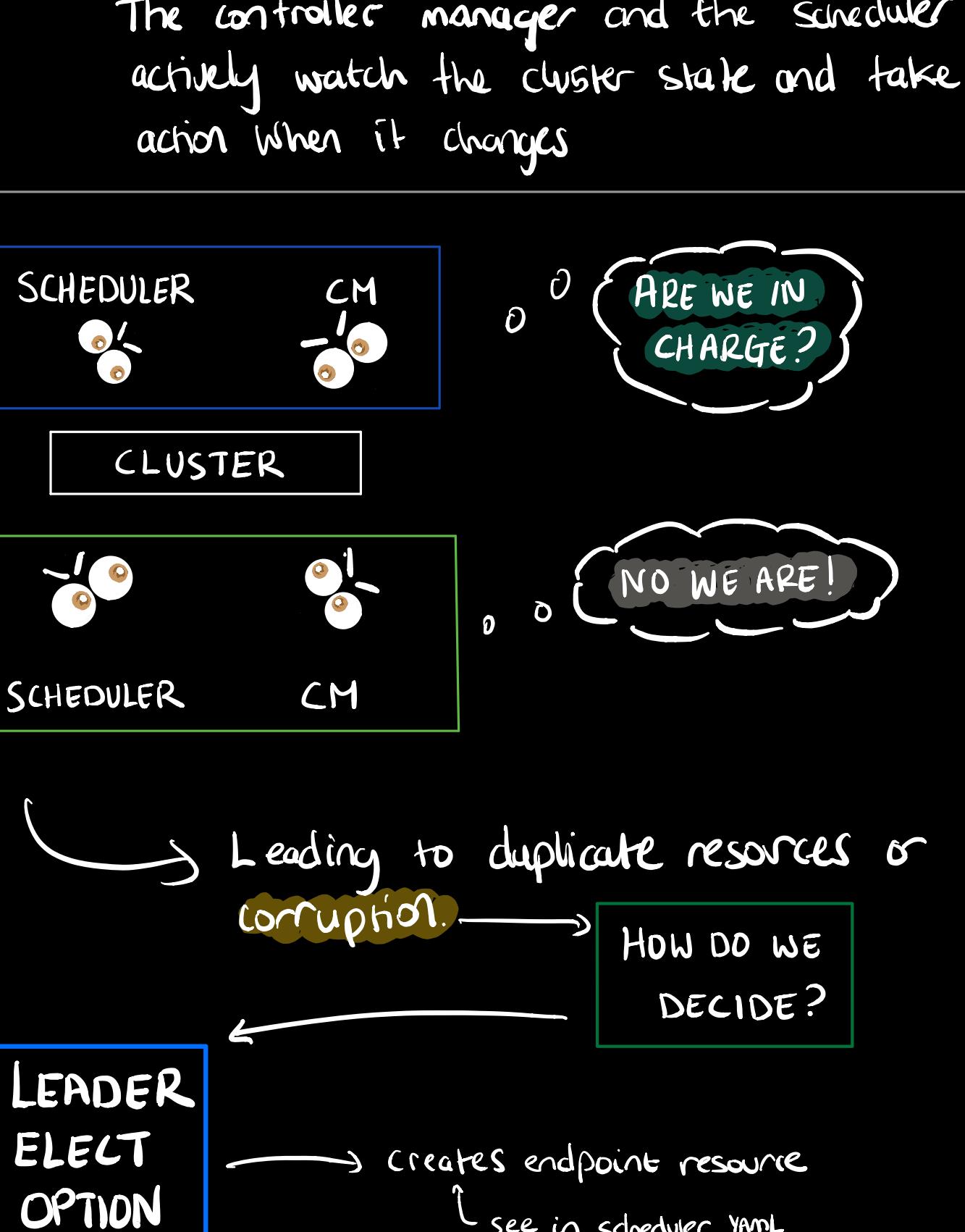
SCHEDULER



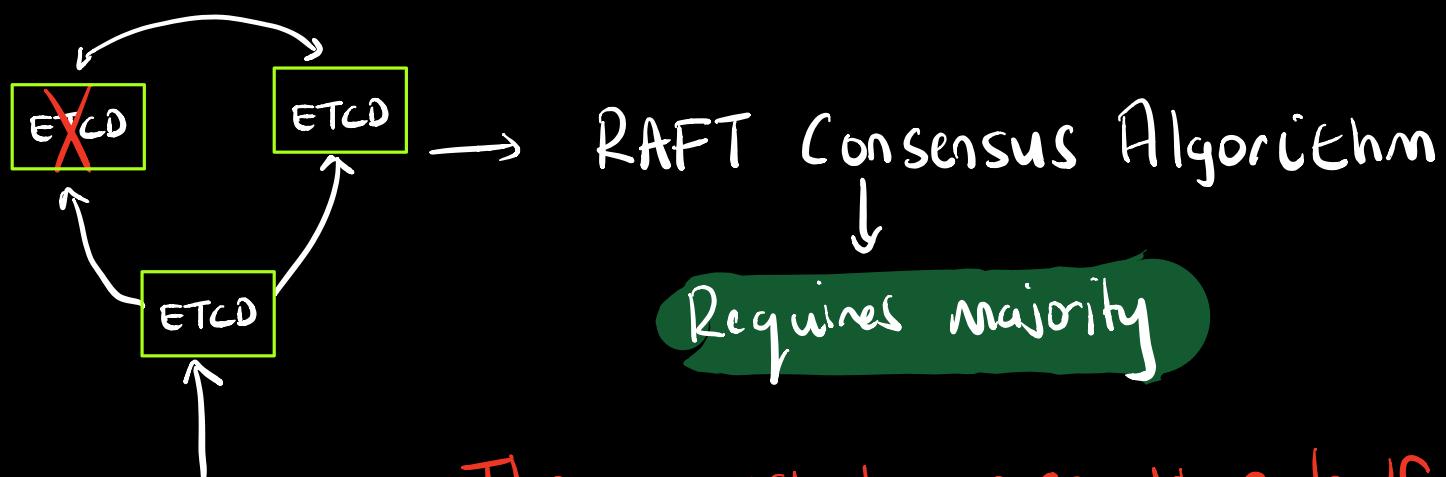
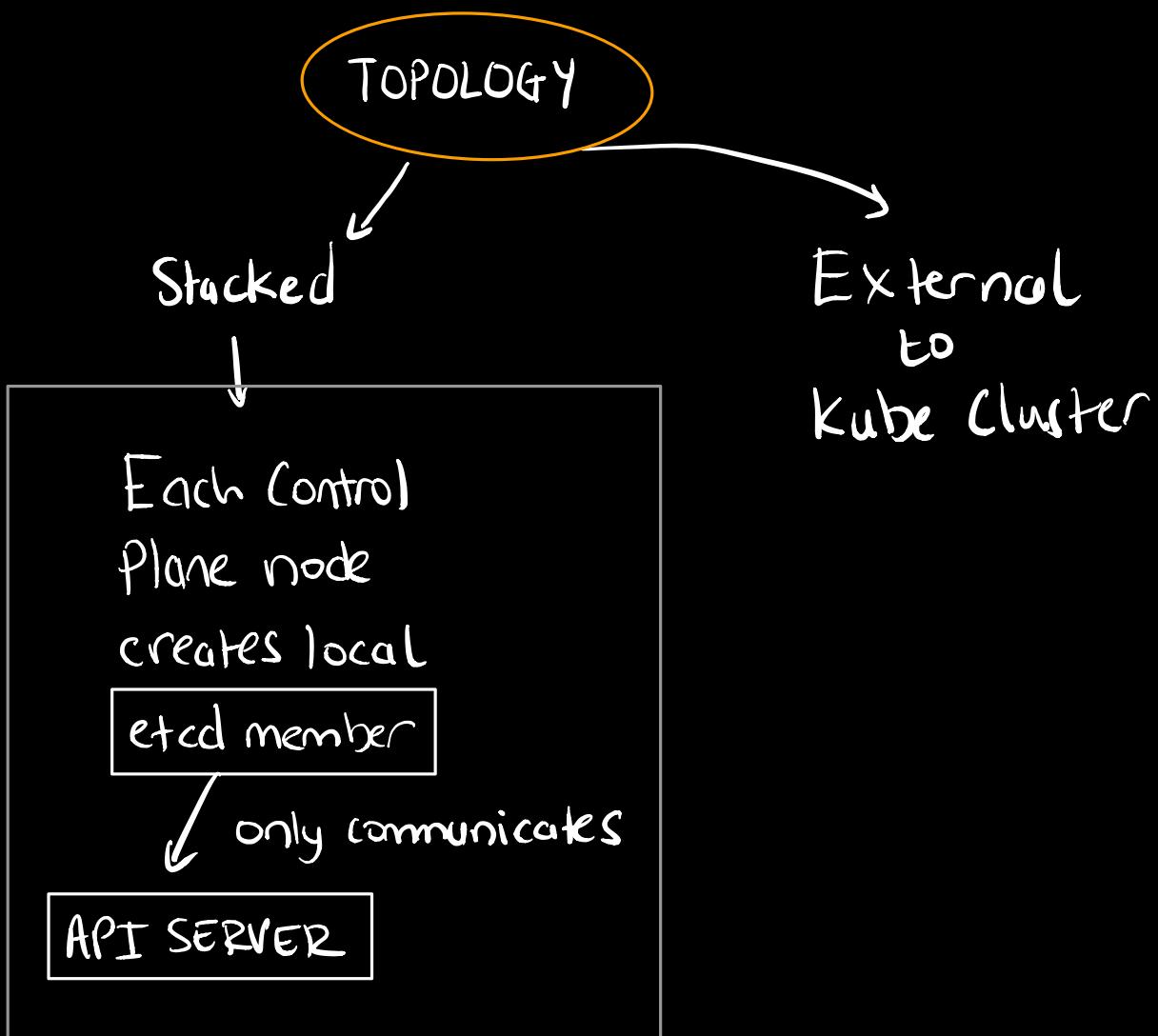
CM



NO WE ARE!



REPLICATING ETCD

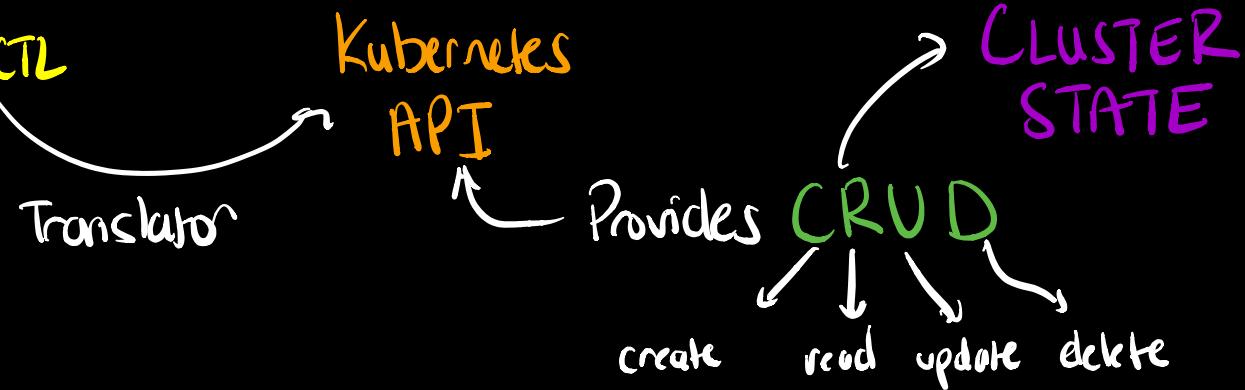


There must be more than half taking place in the state change
∴ there must be odd number of nodes

CONFIGURING SECURE CLUSTER COMMUNICATIONS

→ All communication via HTTPS

→ Kube CTL



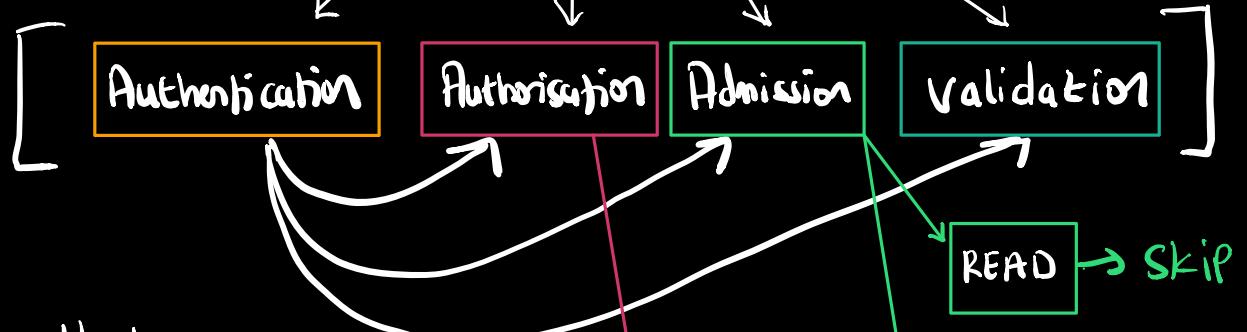
Return Response

→ Kube CTL
CREATE A POD

HTTP POST
API SERVER

State Stored in ETCD

MULTIPLE PLUGINS



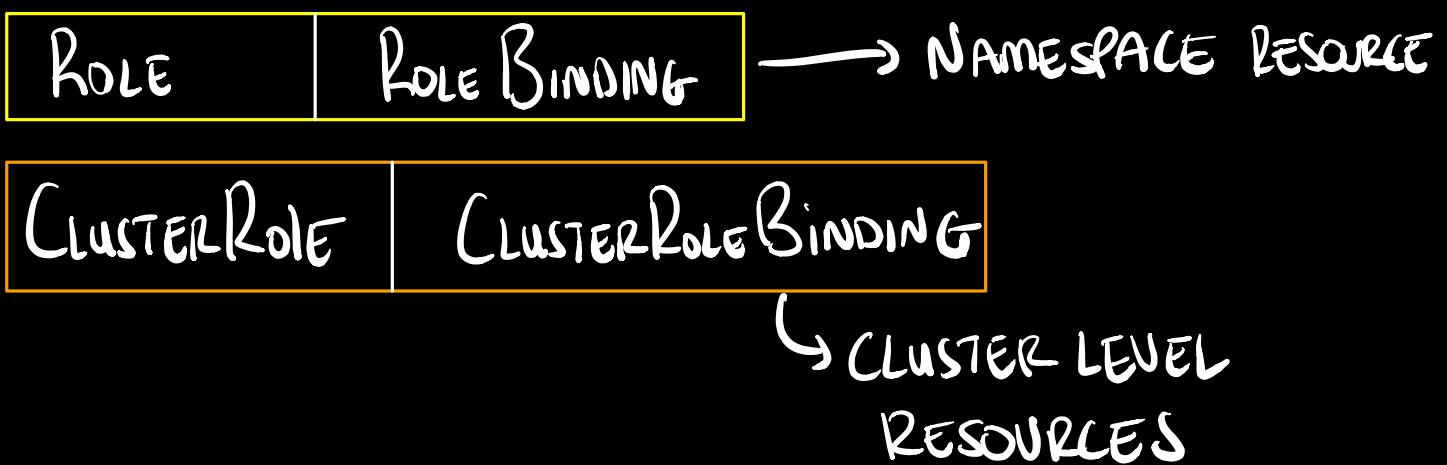
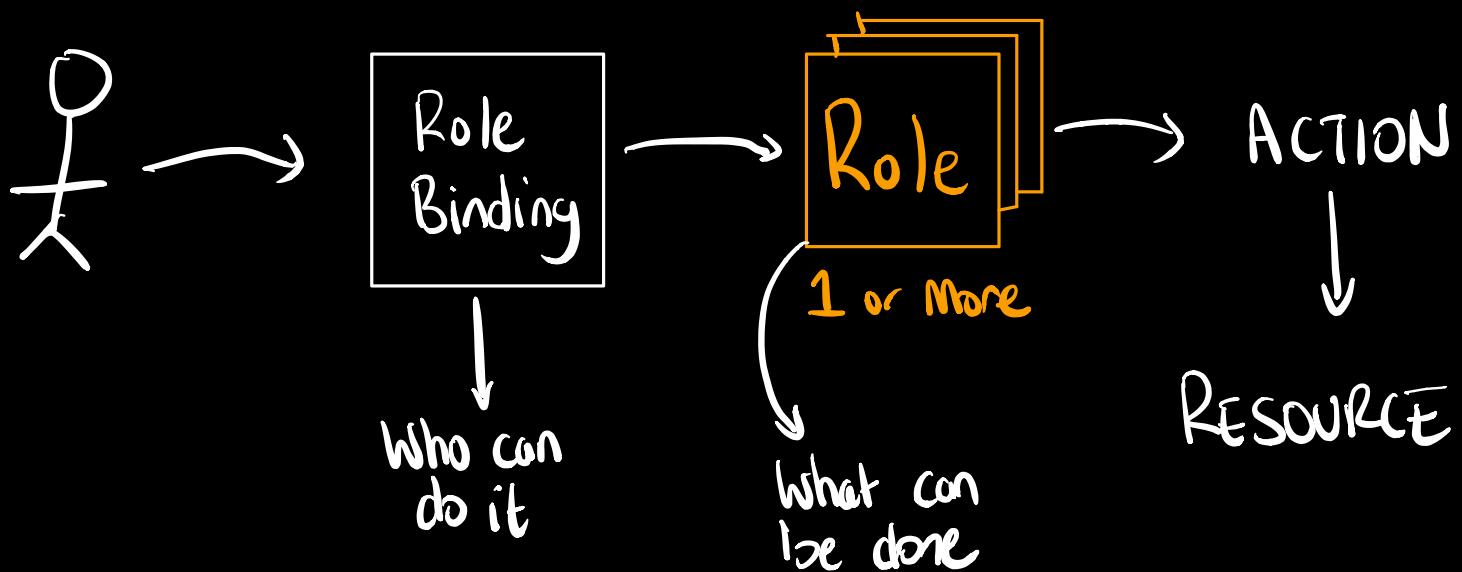
HTTP

HEADER

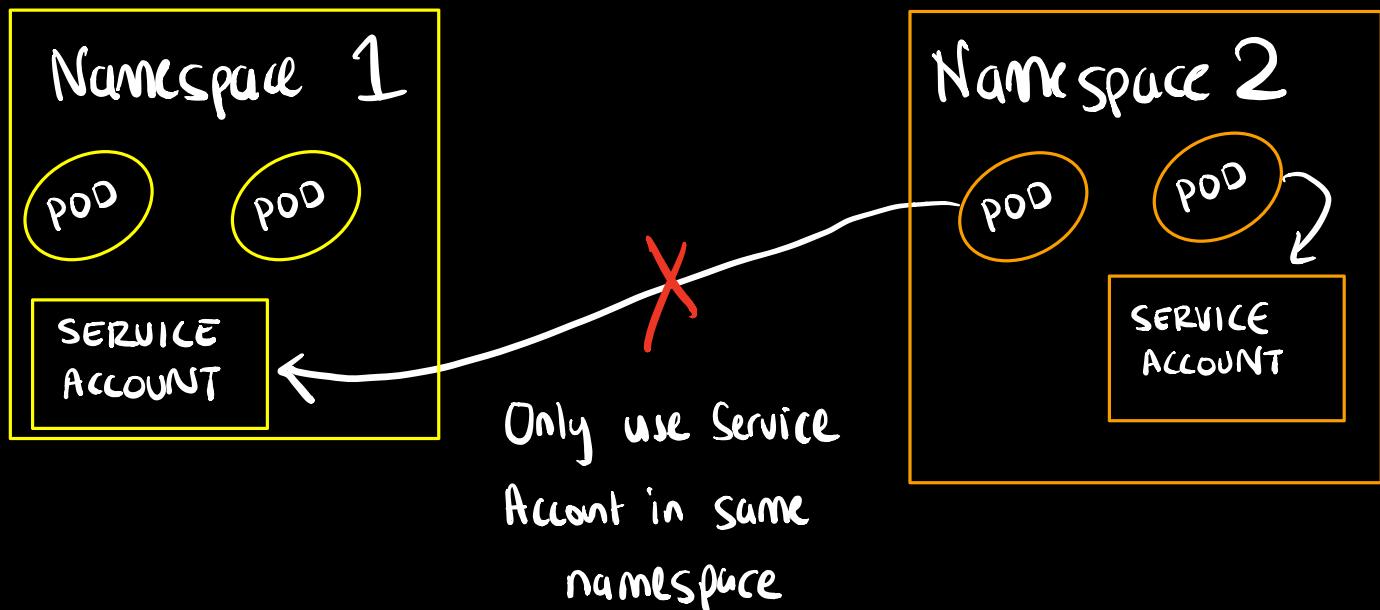
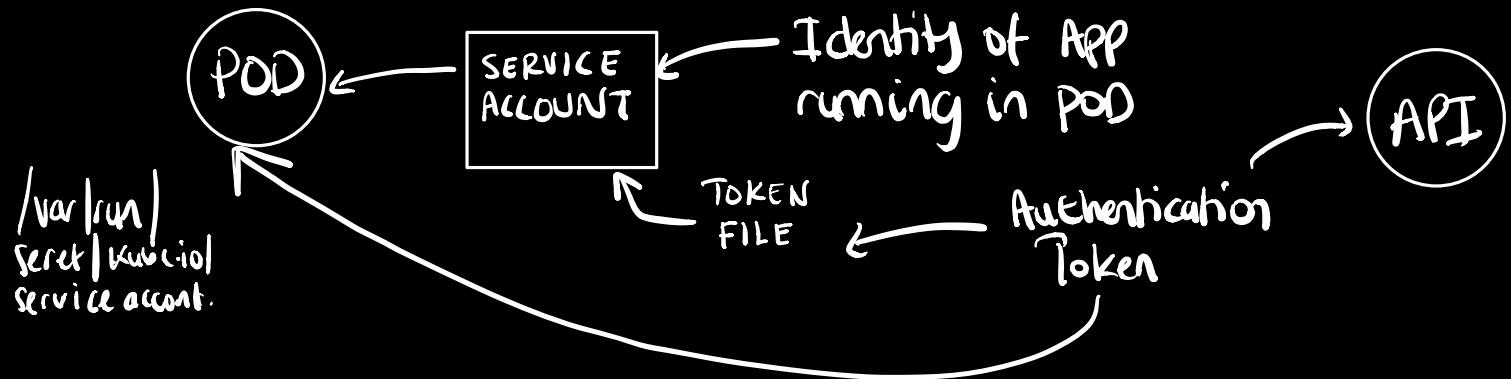
CERTIFICATE

ROLES AND ACCESS

RBAC is used to prevent unauthorised users from modifying the cluster state



SERVICE ACCOUNT



RUNNING END TO END TESTS ON CLUSTER

Performance
and
Response of
Application

Poor Cluster
Performance

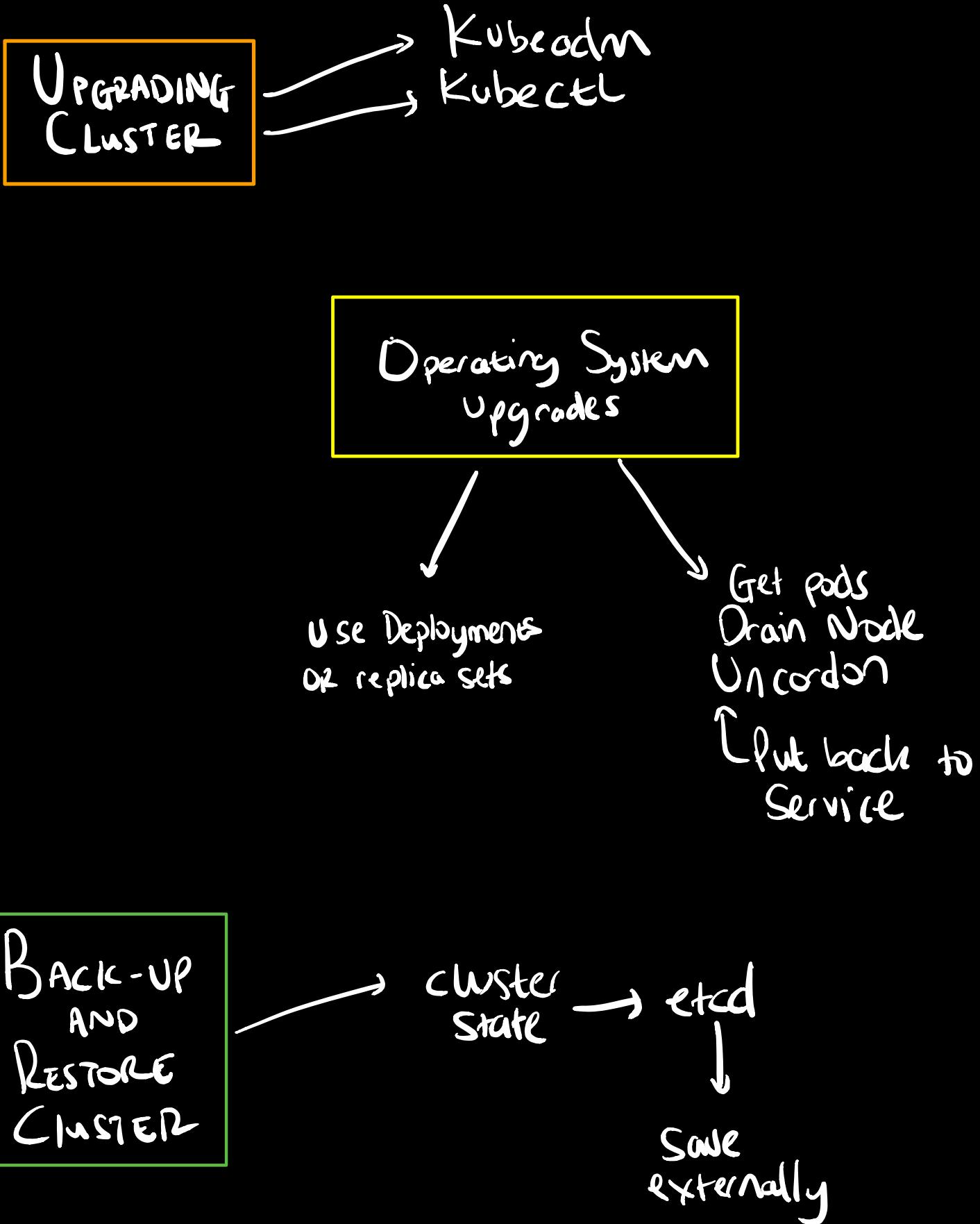


kubeTest

Example
Tests

- ✓ Deployments can run
- ✓ Pods can run
- ✓ Pods can be directly accessed
- ✓ Logs can be collected
- ✓ Commands run from pod
- ✓ Services can provide access
- ✓ Nodes are healthy
- ✓ Pods are healthy

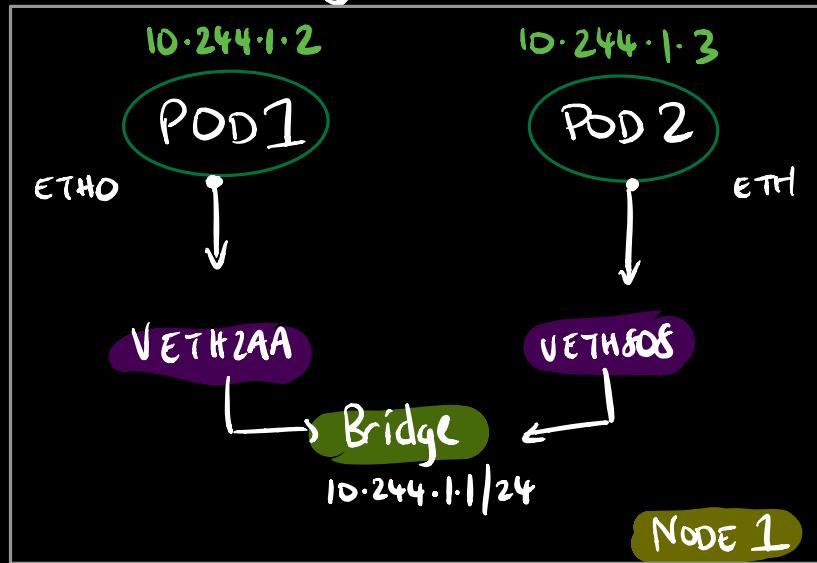
MANAGING CLUSTER



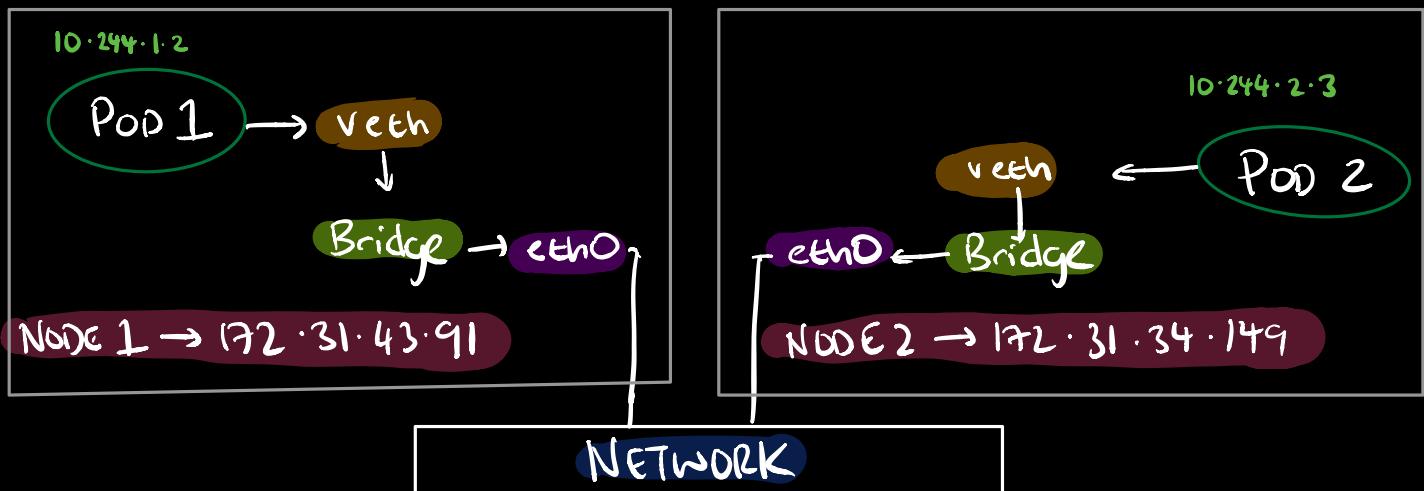
NETWORK
CLUSTER
COMMUNICATIONS

POD AND NODE NETWORKING

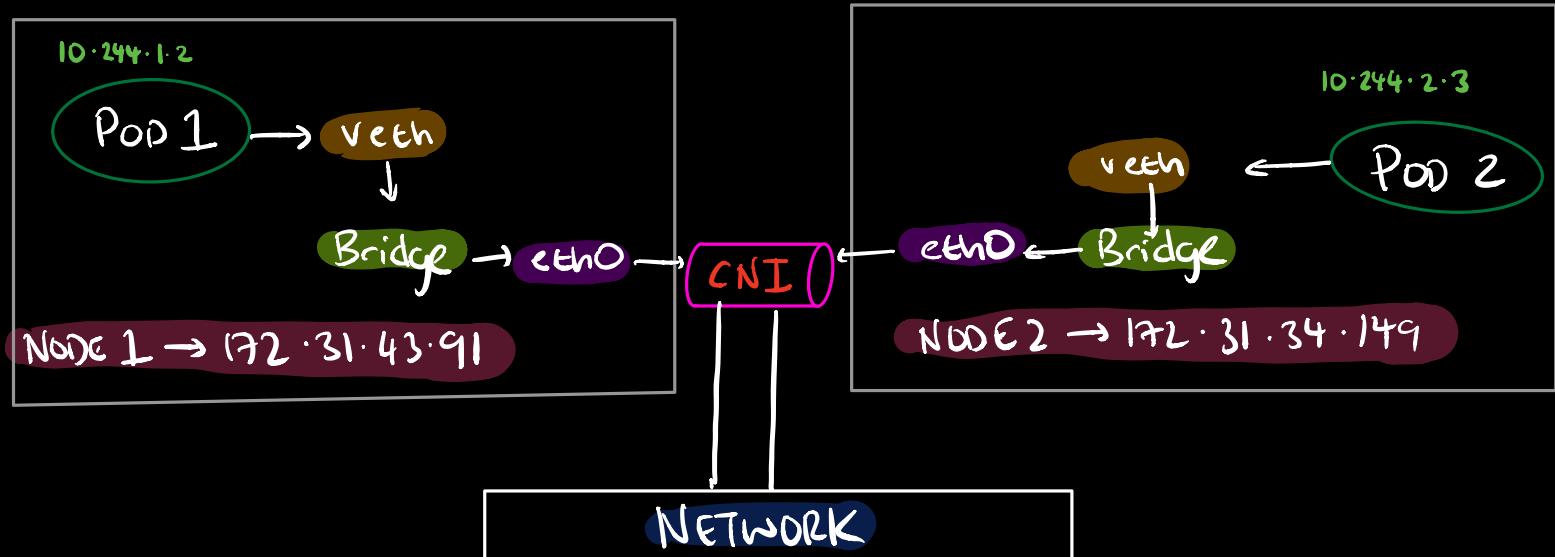
Networking within a node



Networking Outside of the Node



CONTAINER NETWORK INTERFACE



CNI is a network Overlay

- ↳ Allows building tunnel between nodes
 - ↳ Sits on top of existing networks
 - ↳ Encapsulates Packet
 - ↳ changes SRC & DST



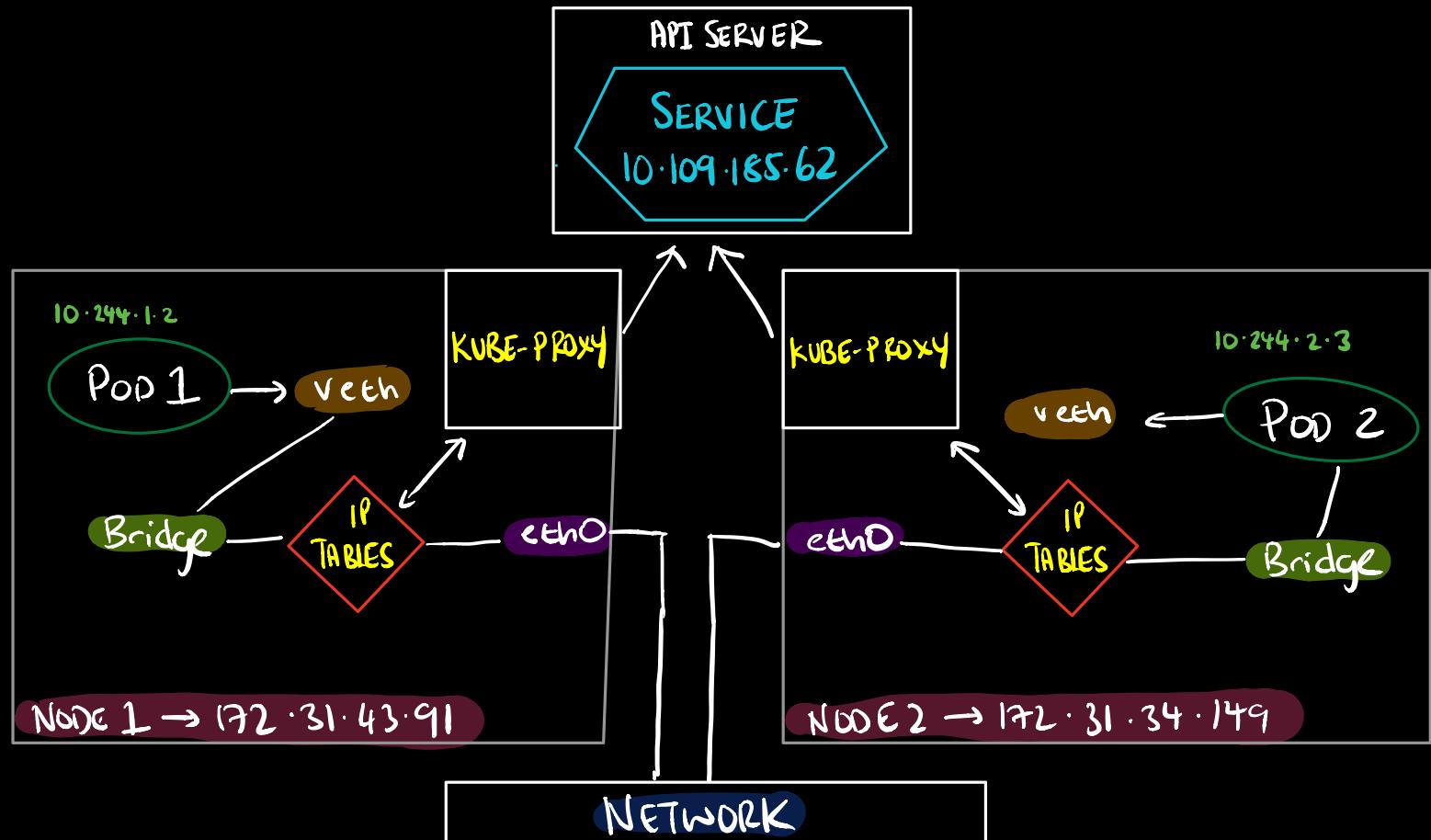
How
CNI
DO
THIS?

- There is a mapping associated in user space
 - ↳ Program all pod ip address's to node IP, when reach other node, de-encapsulate packet and give to bridge

Example CNI
Calico Flannel

Appears Local To Node!

SERVICE NETWORKING



→ Pods come and go.

↳ How does cluster keep track?

↳ Services!

↳ Provides virtual interface → Auto assigned to pods behind interface.

Example

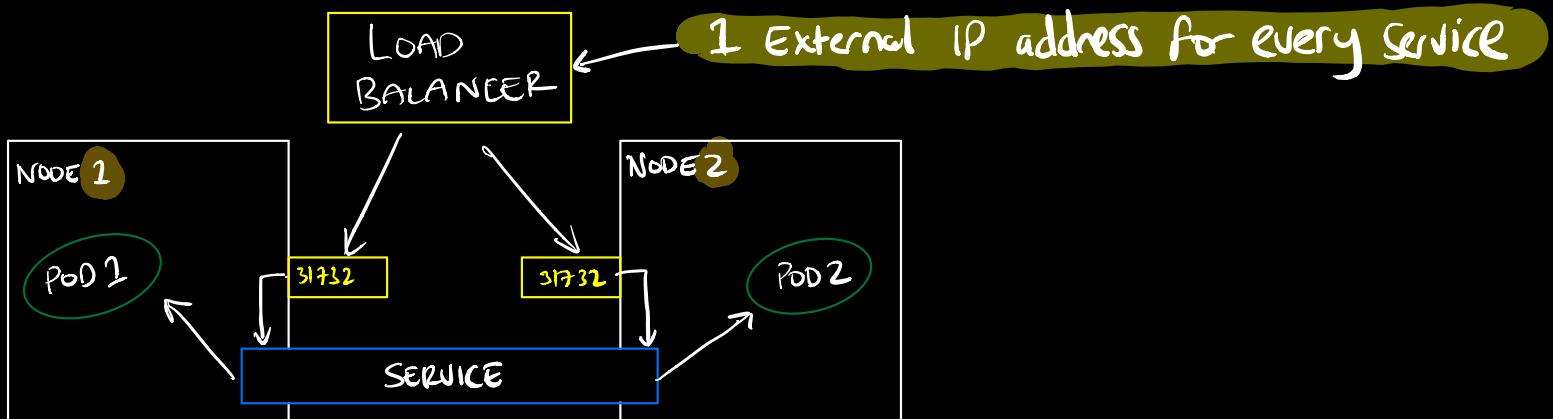
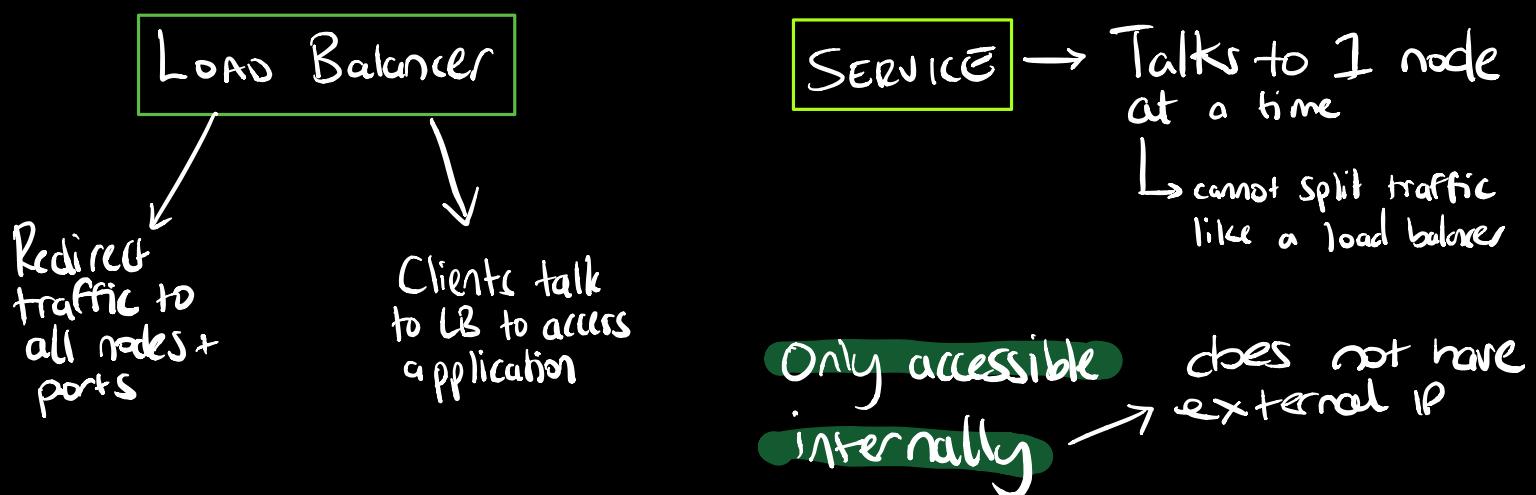
Cluster IP Services

→ Auto created on cluster creation

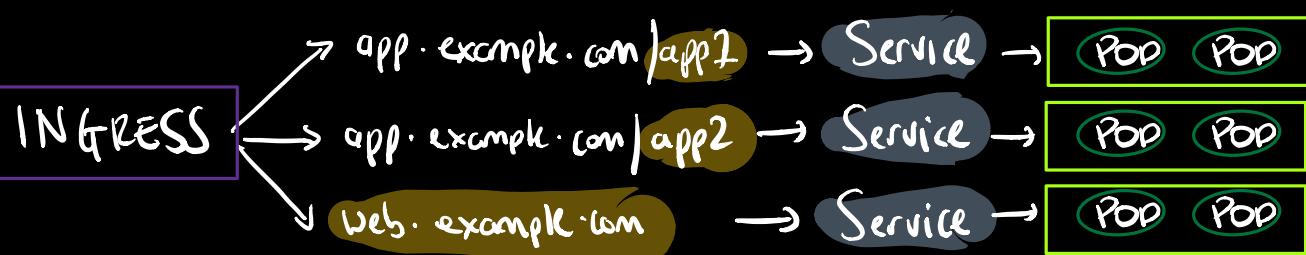
→ Takes care of internal routing

→ No matter where moves other pods know how to communicate to it

INGRESS RULES AND LOAD BALANCERS



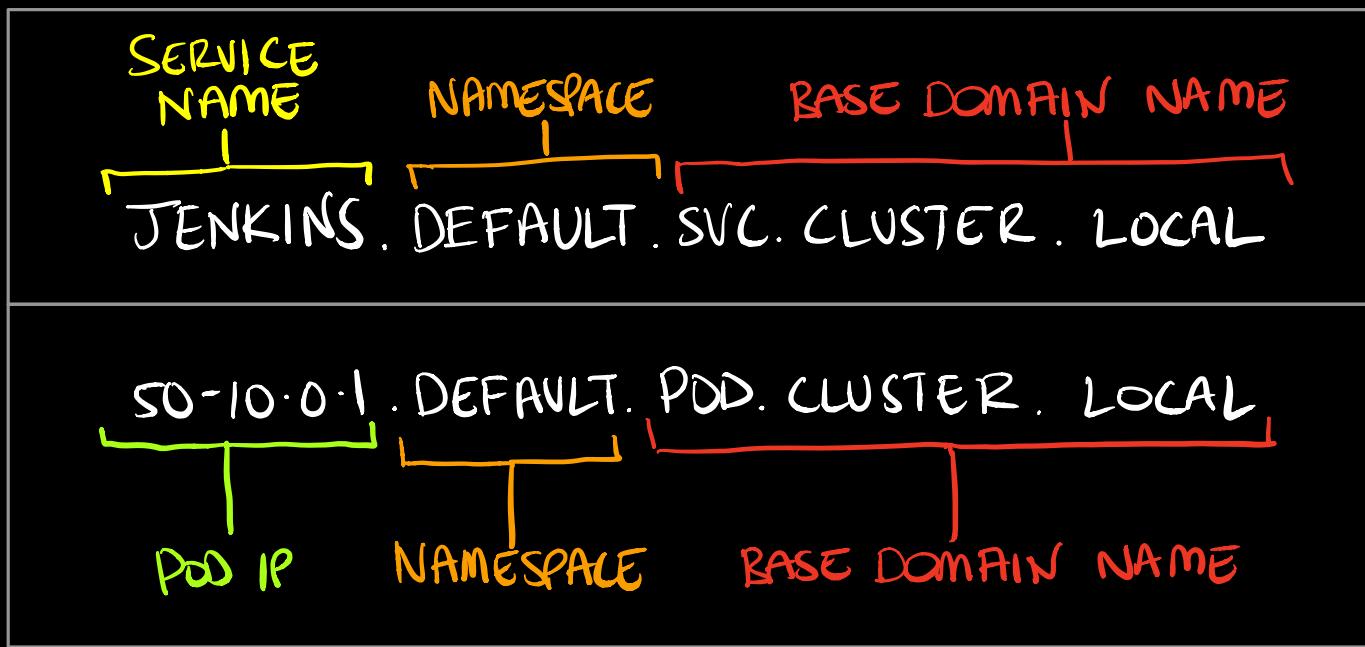
INGRESS



Access multiple Services with Single IP Address

CLUSTER DNS

EVERY SERVICE DEFINED IN THE CLUSTER IS ASSIGNED A DNS NAME



A PODS DNS SEARCH WILL INCLUDE THE PODS OWN NAMESPACE AND THE CLUSTERS DEFAULT DOMAIN

POD SCHEDULING WITHIN THE KUBERNETES CLUSTER

CONFIGURING THE KUBERNETES SCHEDULER

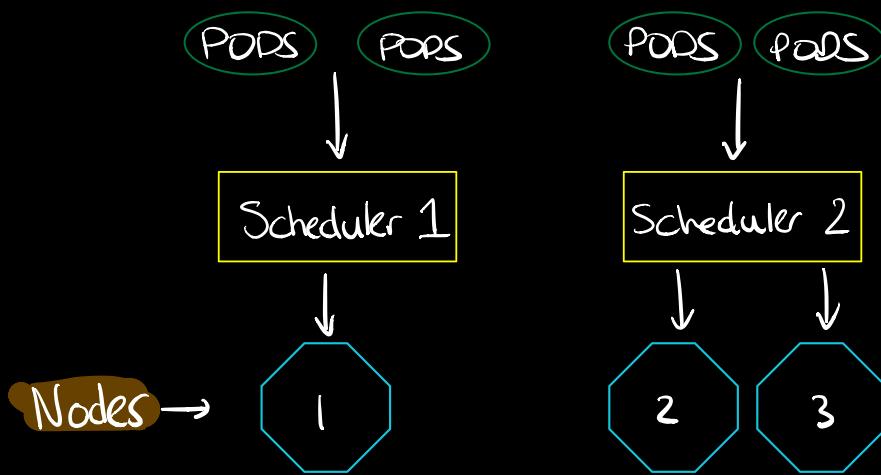
SCHEDULER RESPONSIBLE FOR ASSIGNING POD TO NODE
BASED ON RESOURCE REQUIREMENTS OF THE POD



- SCHEDULER** →
- 1 Does the node have adequate hardware resources?
 - 2 Is the node running out of resources?
 - 3 Does the pod request a specific node?
 - 4 Does the node have a matching label?
 - 5 If pod requests a port, is it available?
 - 6 If pod requests a volume, can it be mounted?
 - 7 Does the pod tolerate the taints of the node?
 - 8 Does the pod specify node or pod affinity?

RUNNING MULTIPLE SCHEDULERS FOR MULTIPLE PODS

It is possible to have 2 schedulers working alongside each other.

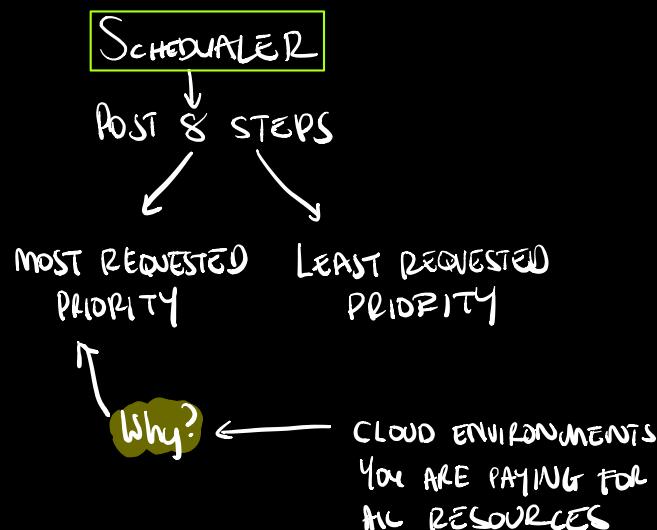


SCHEDULING PODS WITH LIMITS AND LABEL SELECTORS

TRAINTS → REPEL WORK → EXAMPLE
MASTER NODE
NO SCHEDUAL

TOLERATIONS → ALLOW YOU TO
TOLERATE A TRAINT → EXAMPLE
KUBE-PROXY ← DAEMON SET POD
MUST RUN ON ALL
NODES

CPU/MEMORY
↓
POD MAY NOT BE
USING ALL REQUESTED
RESOURCE AT A GIVEN
TIME.
SCHEDULER LOOKS
AT THE SUM OF
RESOURCES REQUESTED
BY EXISTING PODS



DAEMONSETS

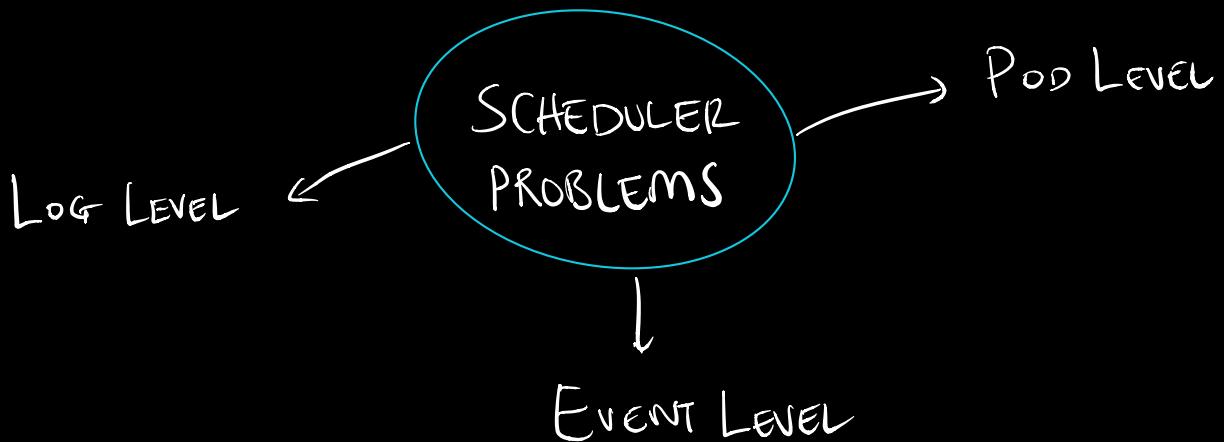
DaemonSets ensure that a single replica of a pod is running on each node at all times



POD DAEMONSET POD
POD REPLICASET POD

If you try delete a daemonset pod, it will simply recreate it.

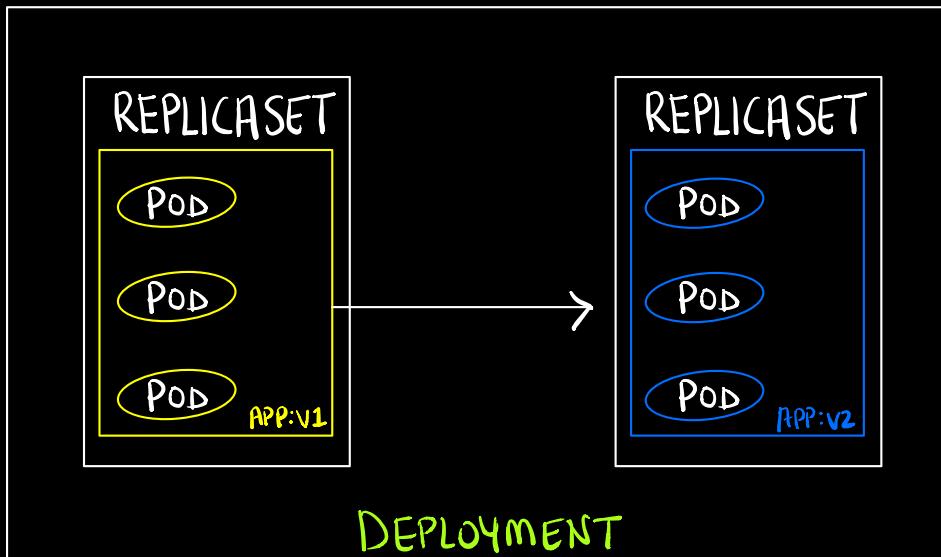
DISPLAY SCHEDULER EVENTS



DEPLOYING
APPLICATIONS
IN THE
KUBERNETES
CLUSTER

DEPLOYING AN APPLICATION, ROLLING UPDATES, AND ROLLBACKS

DEPLOYMENTS → HIGH LEVEL RESOURCE FOR
DEPLOYING AND UPDATING APPS



KUBECTL APPLY → MODIFY OBJECTS TO EXISTING YAML
IF DEPLOYMENT NOT CREATED → ALSO CREATE

KUBECTL REPLACE → REPLACES OLD WITH NEW AND OBJECT MUST
EXIST.

ROLLING UPDATE → PREFERRED WAY → SERVICE NOT INTERRUPTED
→ FASTEST WAY

KUBECTL ROLLOUT → ROLL BACK PREVIOUS VERSION

CONFIGURING AN APP FOR HA AND SCALE

AVOIDING
BAD
VERSIONS

→ BLOCK BAD VERSION
RELEASE

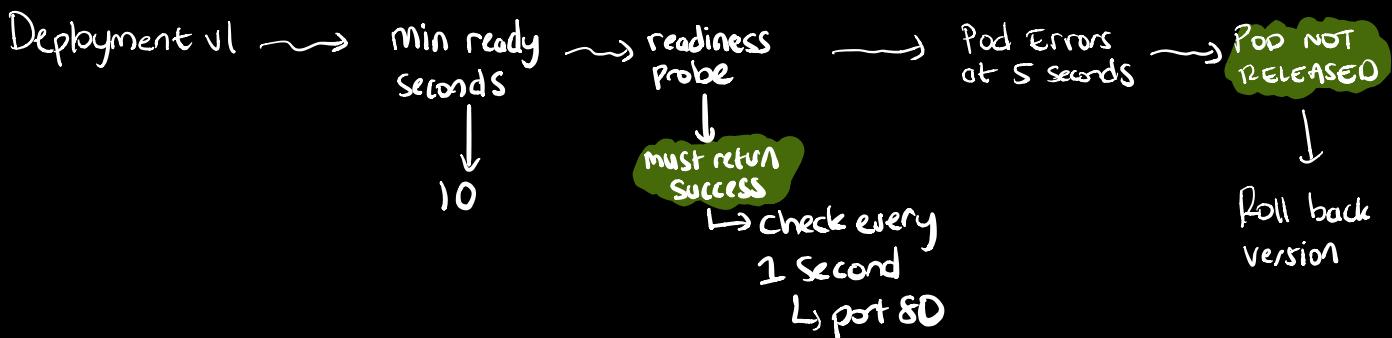
MIN READY
SECONDS

How long a
newly created
pod should be
ready before
considered
available



READINESS
PROBE

Determines
if a specific
pod should
receive
client request
or not.



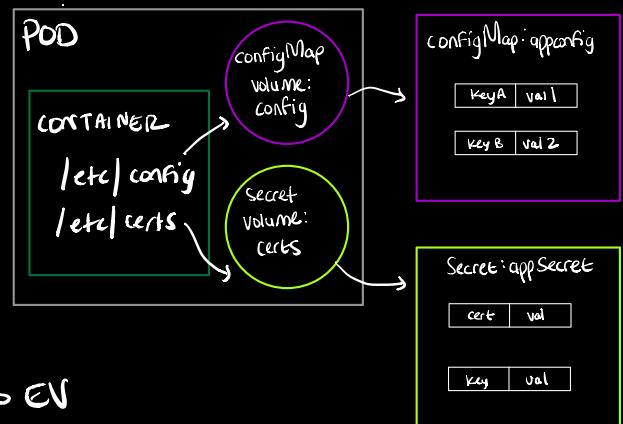
PASSING CONFIGURATION OPTIONS TO APP

ENVIRONMENT VARIABLES

STORE IN CONFIG MAP
CREATE SECRET & PASS TO EV

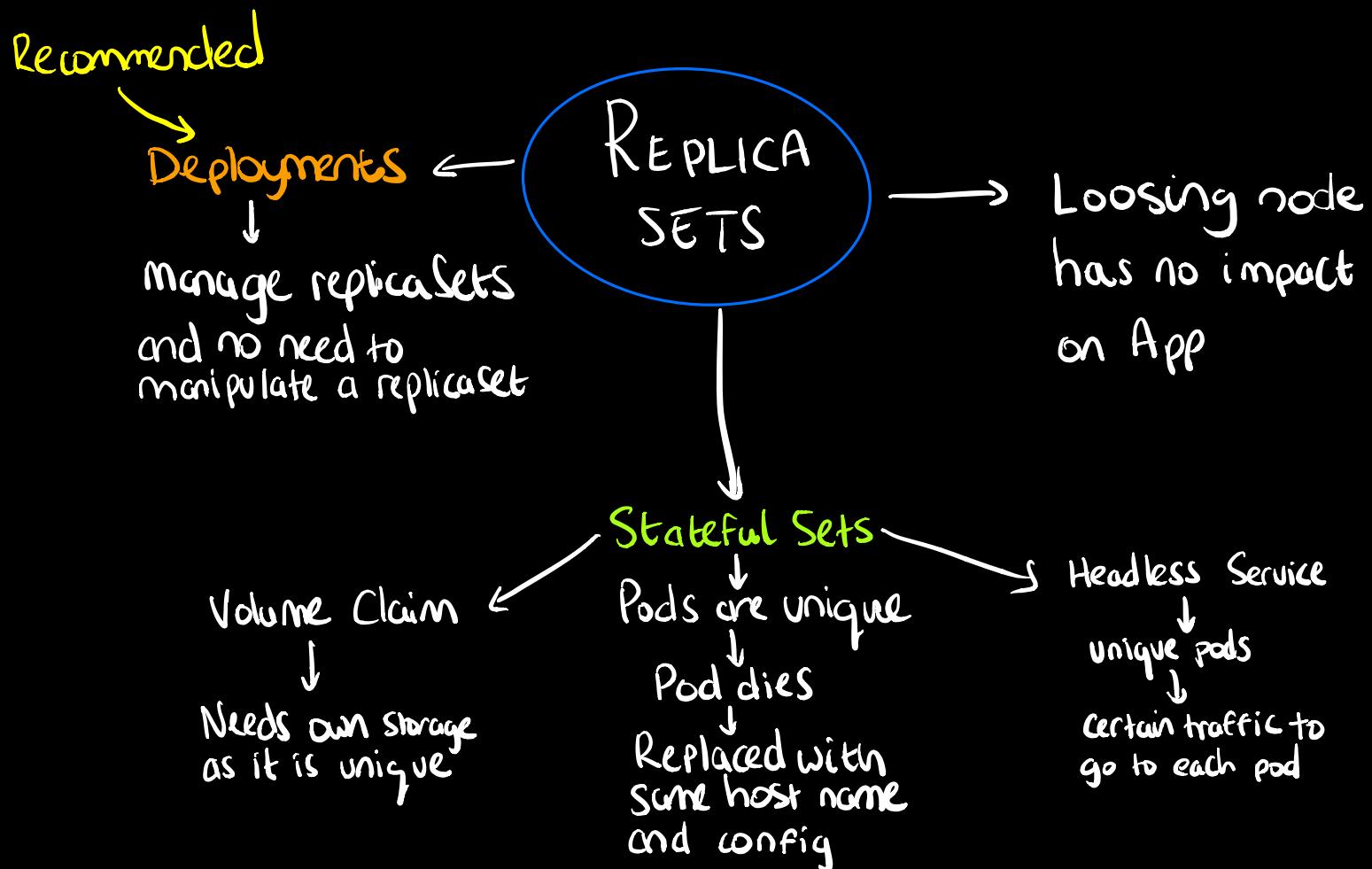
JUST UPDATE → NO NEED TO REBUILD
IMAGE

MULTIPLE
CONTAINERS
CAN USE
SAME



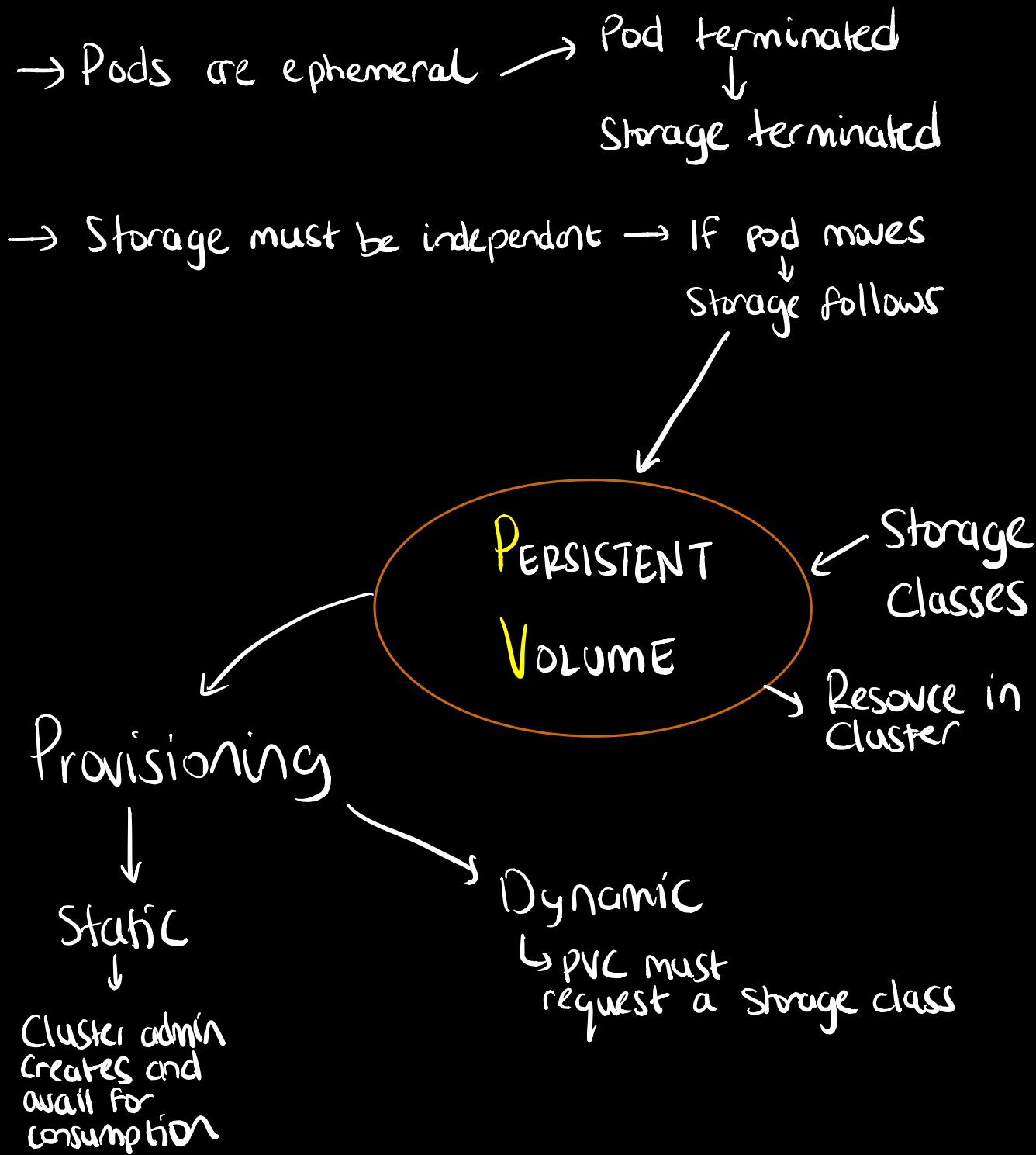
CREATING A SELF-HEALING APP

ReplicaSets ensure that a identically configured pods are running at the desired replica count



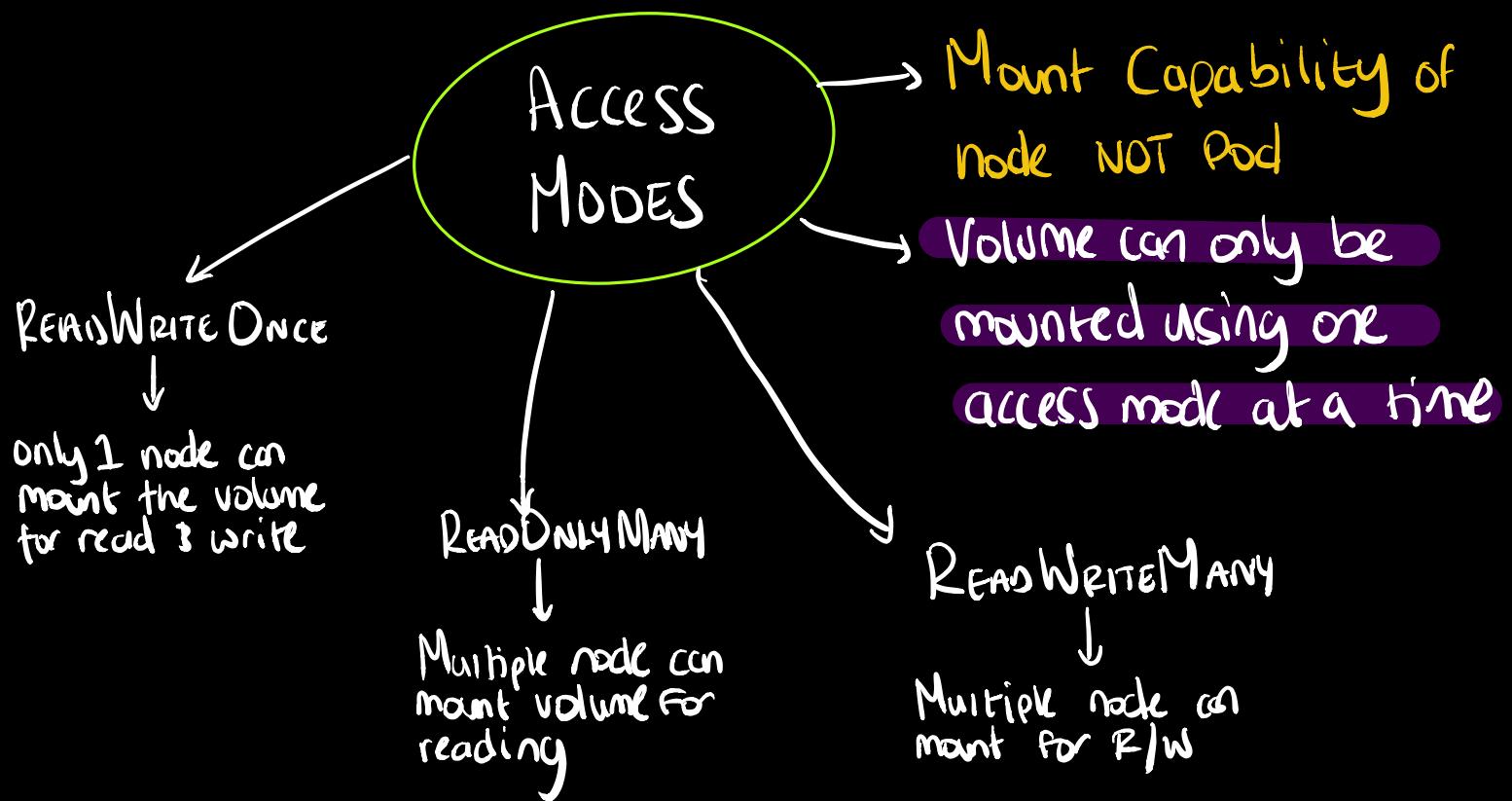
MANAGING DATA IN KUBERNETES CLUSTER

PERSISTENT VOLUMES



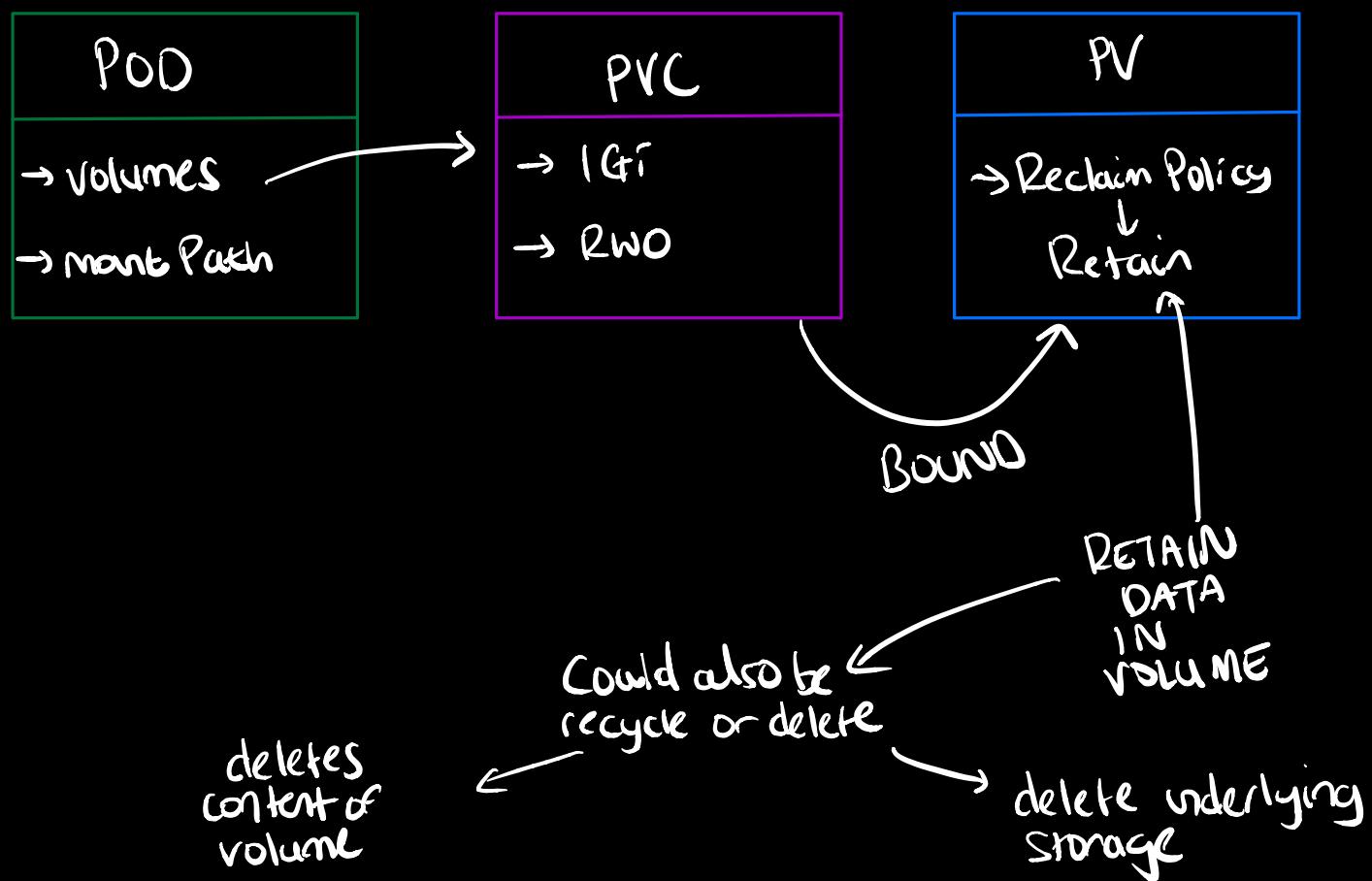
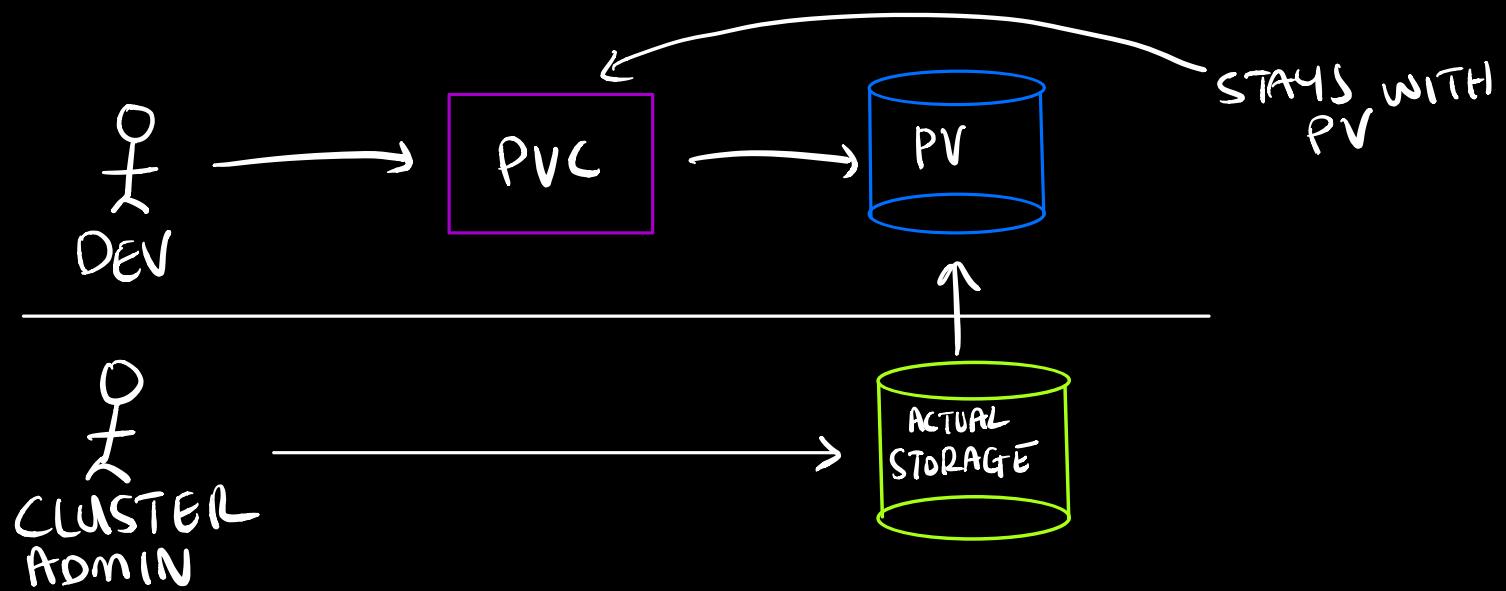
VOLUME ACCESS MODES

By specifying an access mode with your PV,
you allow the volume to be mounted to one
or many nodes, as well as read by one or many



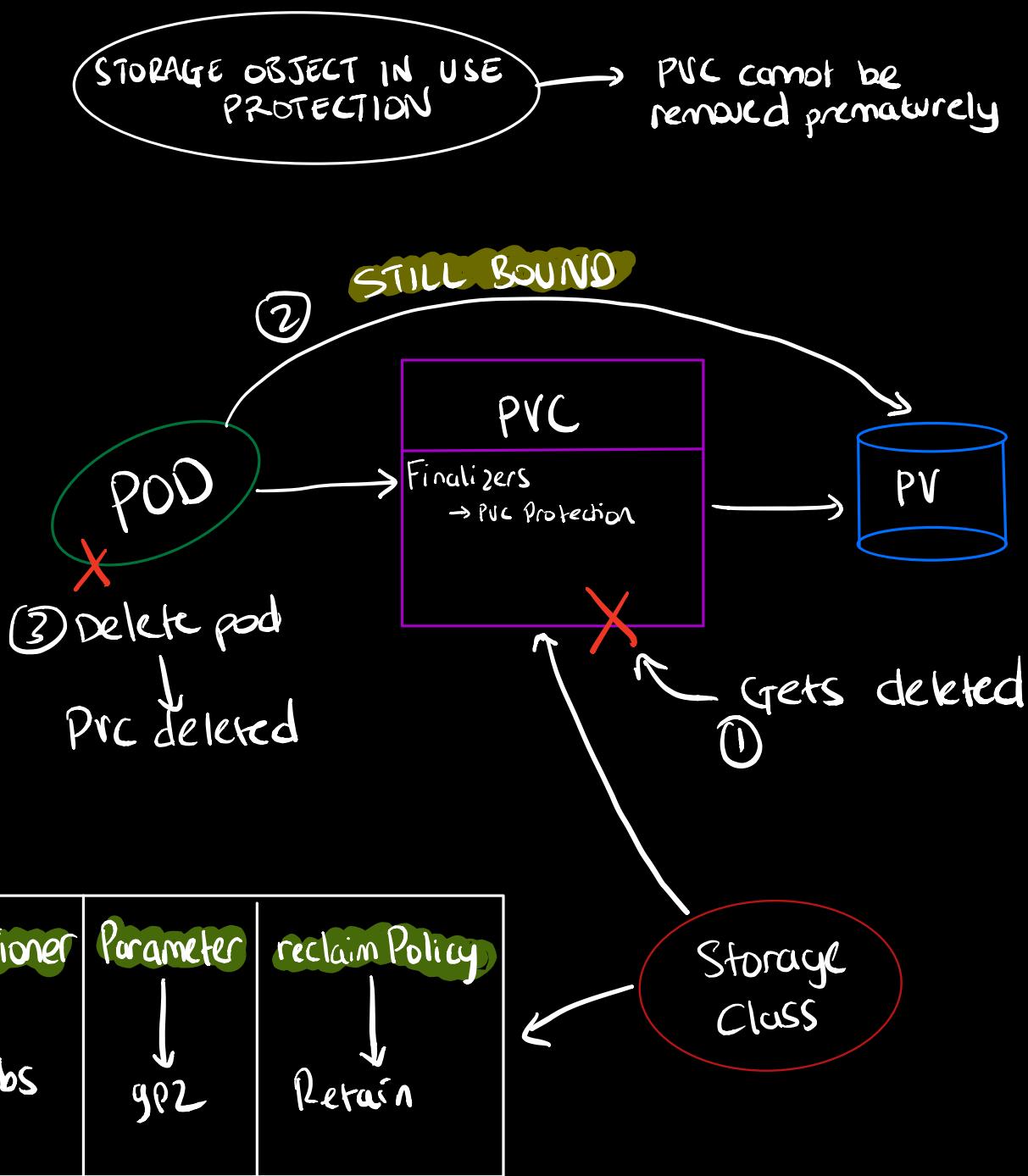
PERSISTENT VOLUME CLAIMS (PVC)

PVC allows the application developer to request storage for the application, without having to know underlying infra.



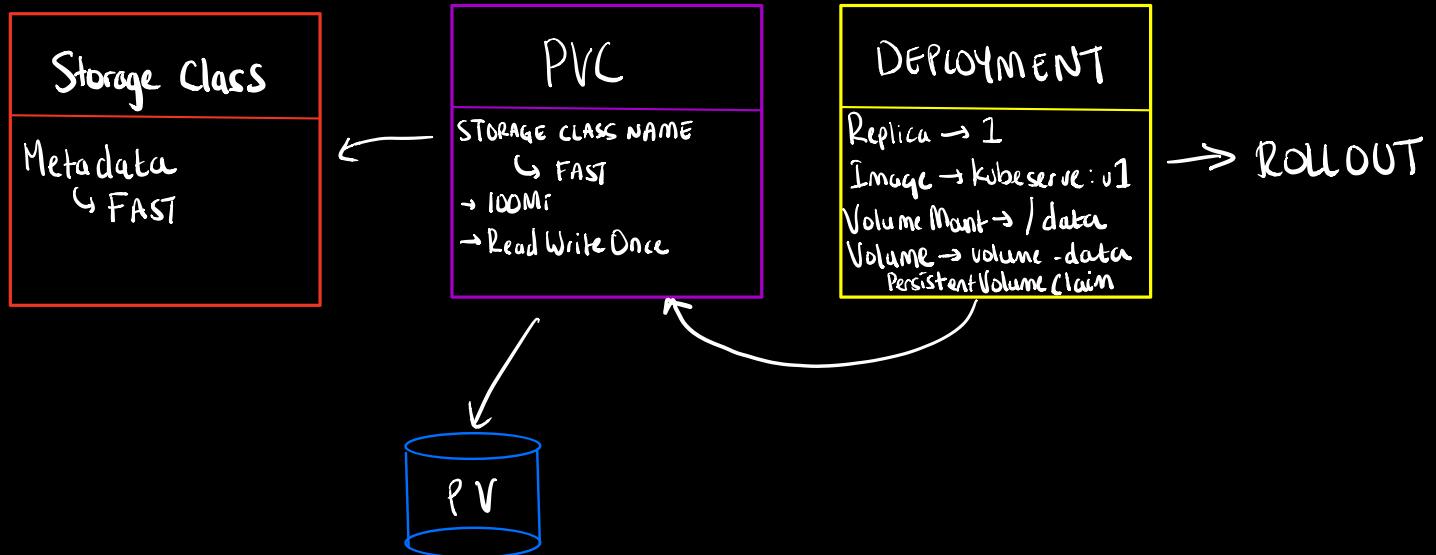
STORAGE OBJECTS

VOLUMES that are already in use by a pod are protected against data loss. This means even if you delete a PVC, you can still access volume from pod.



APPLICATIONS WITH PERSISTENT STORAGE

Example →

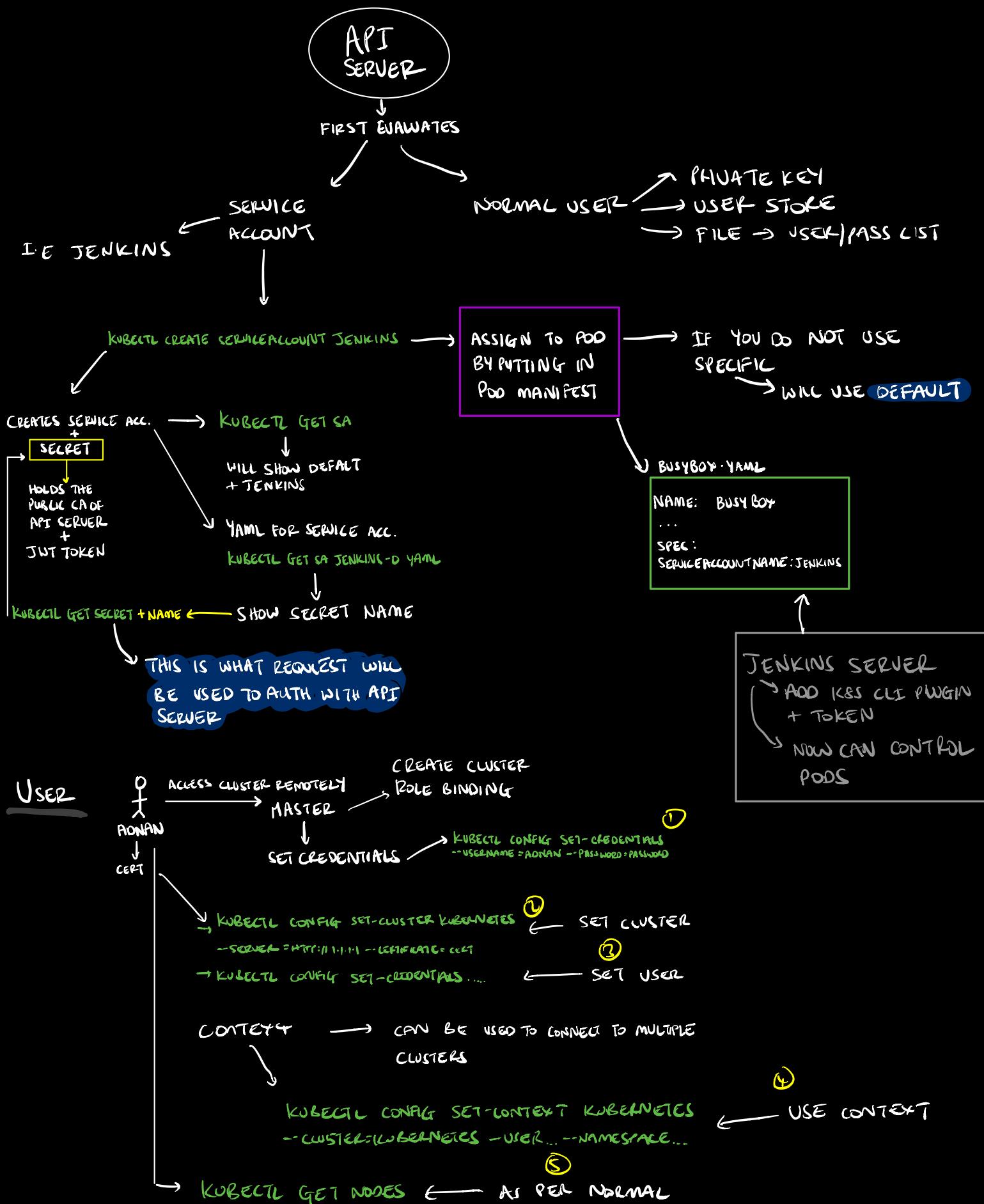


- ① create storage class object
- ② create PVC object
- ③ create deployment
- ④ rollout deployment
- ⑤ check pods
- ⑥ create file on mount
- ⑦ List contents.

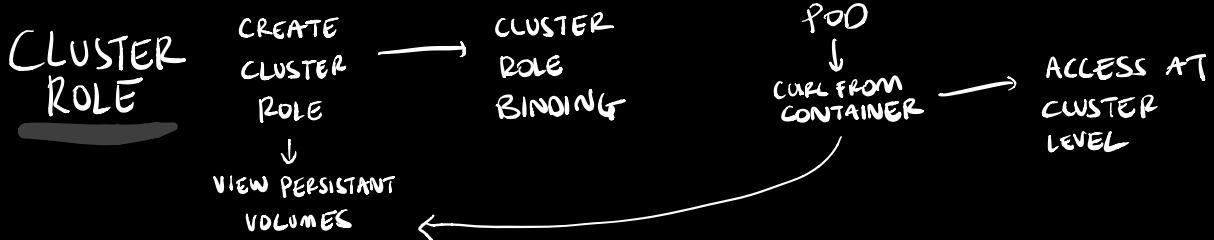
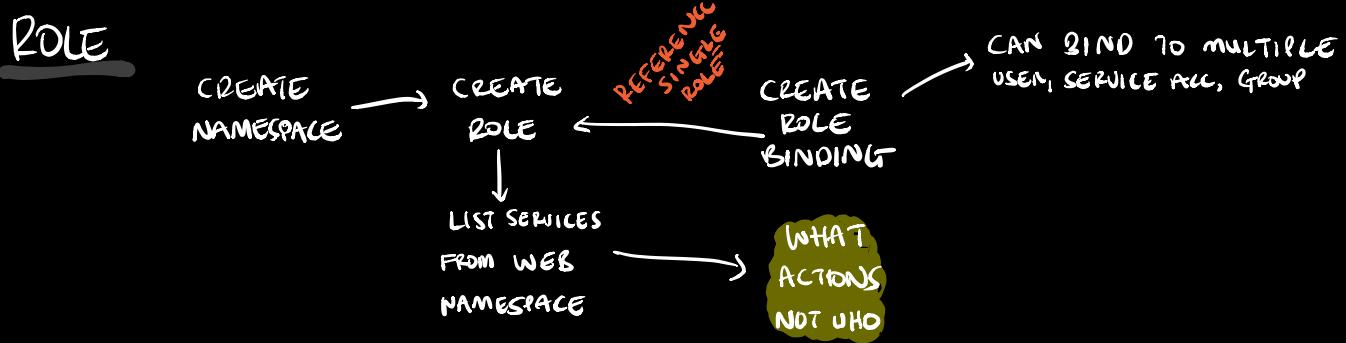
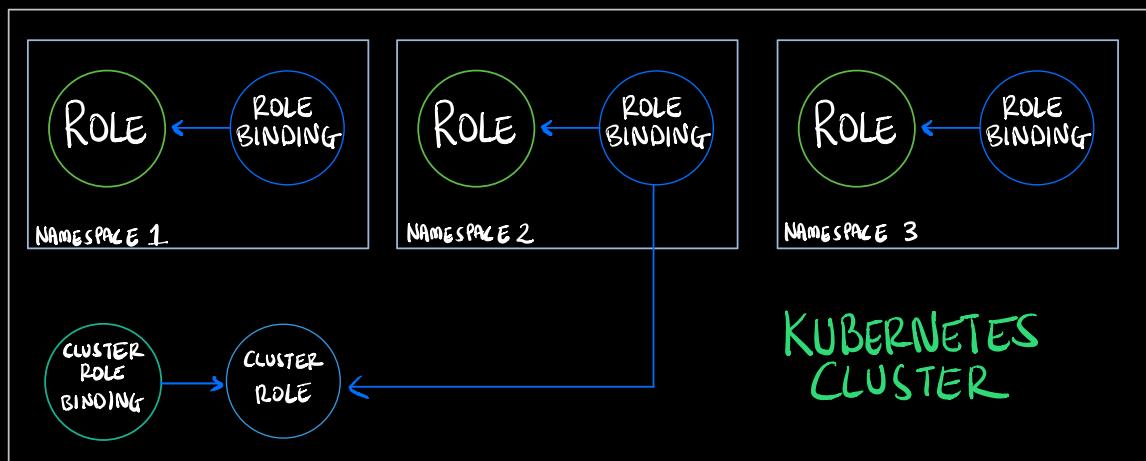
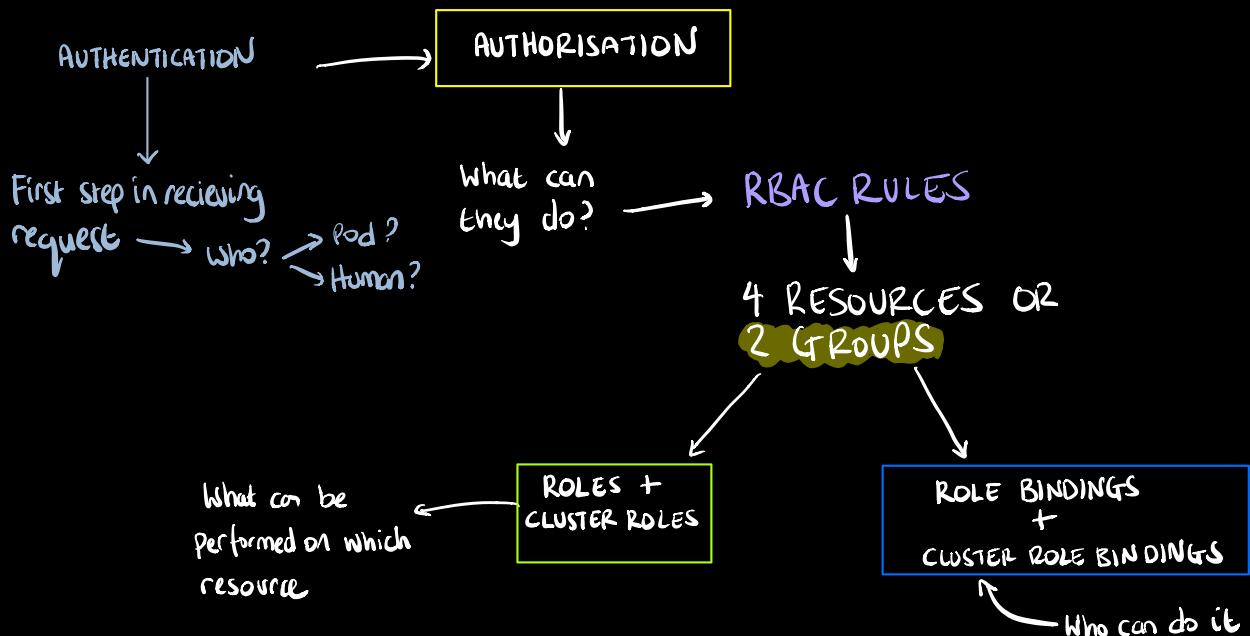


SECURING THE KUBERNETES CLUSTER

SERVICE ACCOUNTS AND USERS

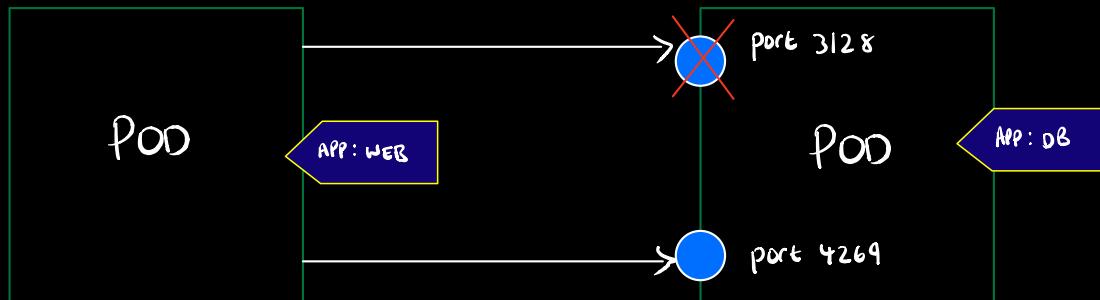


CLUSTER AUTHENTICATION AND AUTHORISATION

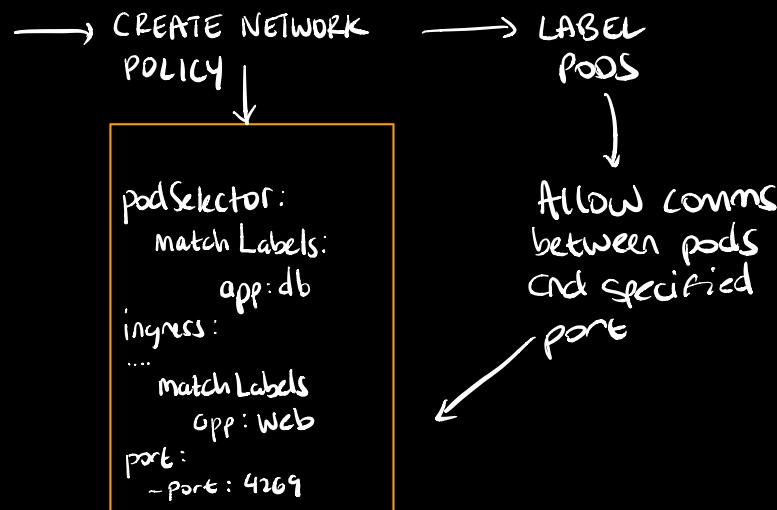
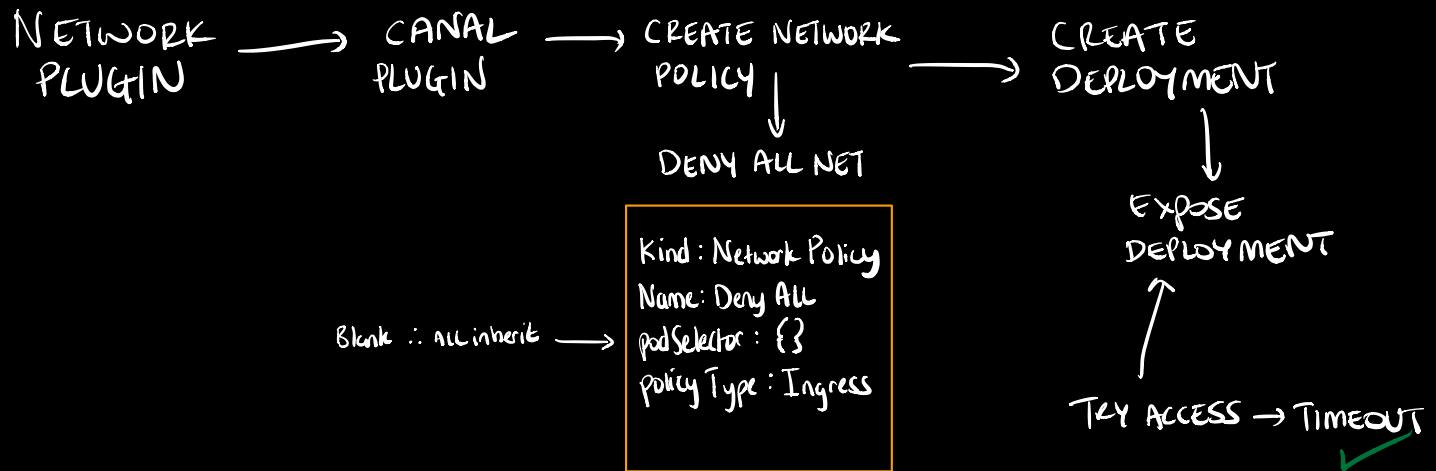


CONFIGURING NETWORK POLICIES

Network policies use selectors to apply rules to pods for communication throughout the cluster

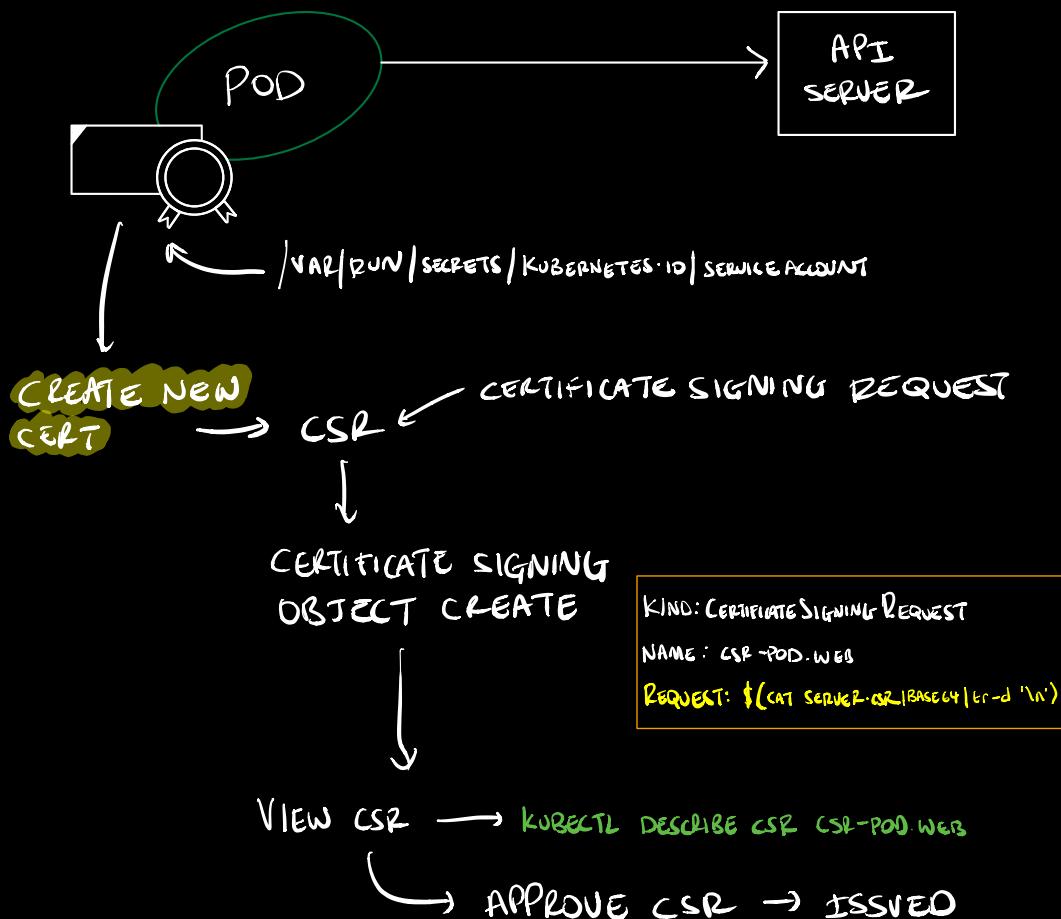


How?



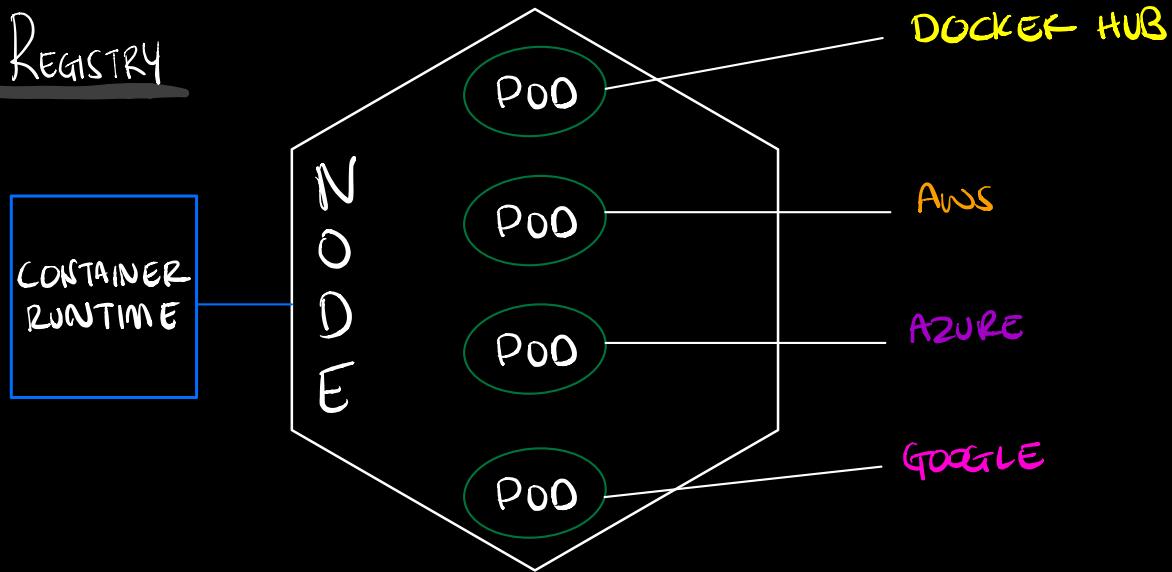
CREATING TLS CERTIFICATES

The CA is used to generate a TLS certificate and authenticate with the API Server.



SECURE IMAGES

PRIVATE REGISTRY



CONTROLLING
IMAGES THAT
GO INTO
PRODUCTION

VULNERABILITIES → CLAIR (SCANNING)

SOMETHING ON IT
CAUSE NODE CRASH

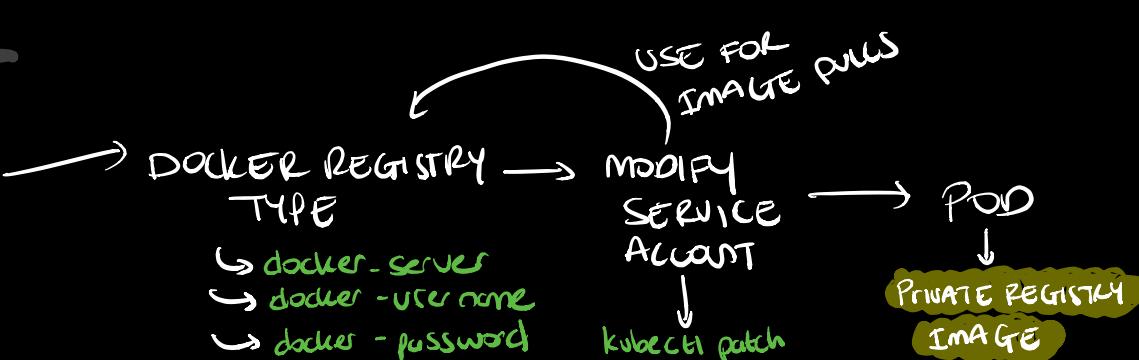
LOGIN TO PRIVATE REGISTRY

↳ TAG DOCKER IMAGE

↳ PUSH TO PRIVATE REGISTRY

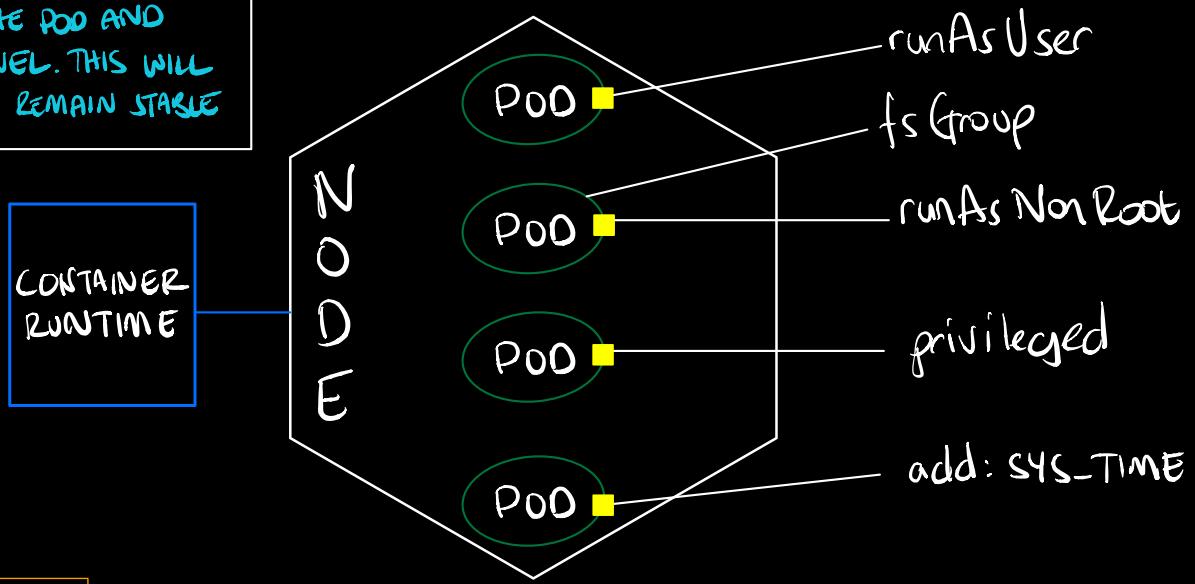
How?

KUBERNETES
CREATE
SECRET



DEFINING SECURITY CONTEXT

LIMIT ACCESS TO CERTAIN OBJECTS AT THE POD AND CONTAINER LEVEL. THIS WILL ALLOW IMAGES TO REMAIN STABLE



Kind: Pod
image: alpine
Security context:
runAsUser: 405

Run Pod as 405

Can also put 'runAsNonRoot'

Ability to run as privileged → 'privileged: true'

CONTAINER LEVEL

ABILITY TO LOCK DOWN KERNEL LEVEL FEATURES ON CONTAINER

SETTING CAPABILITIES ON POD LEVEL

ADD

Security Context:
add:
- SYS_TIME
- NET_ADMIN

REMOVE

Security Context:
drop:
CHOWN

SECURING PERSISTENT KEY/VALUE STORE

SECRETS ALLOW YOU TO EXPOSE ENTRIES AS FILES
IN A VOLUME. KEEPING THIS DATA
SECURE IS CRITICAL TO CLUSTER SECURITY

DATA MUST
LIVE BEYOND
LIFE OF POD → SECRETS → KEY/VALUE PAIR

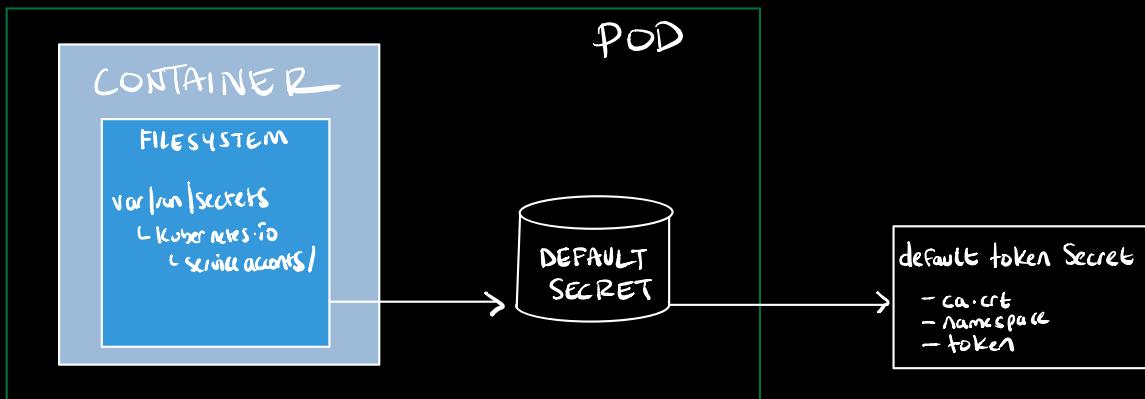
↓
PASS AS ENV VAR

OR

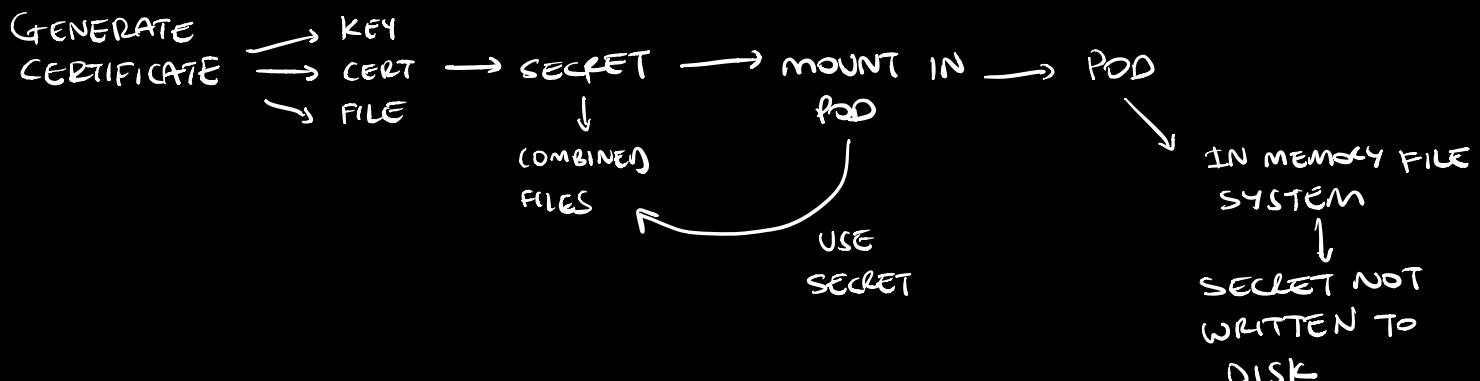
EXPOSE AS FILES
IN VOLUME

NOT BEST
PRACTICE

↓
MAY BE
OUTPUT TO
LOG FILES



HTTPS TO WEBSITE

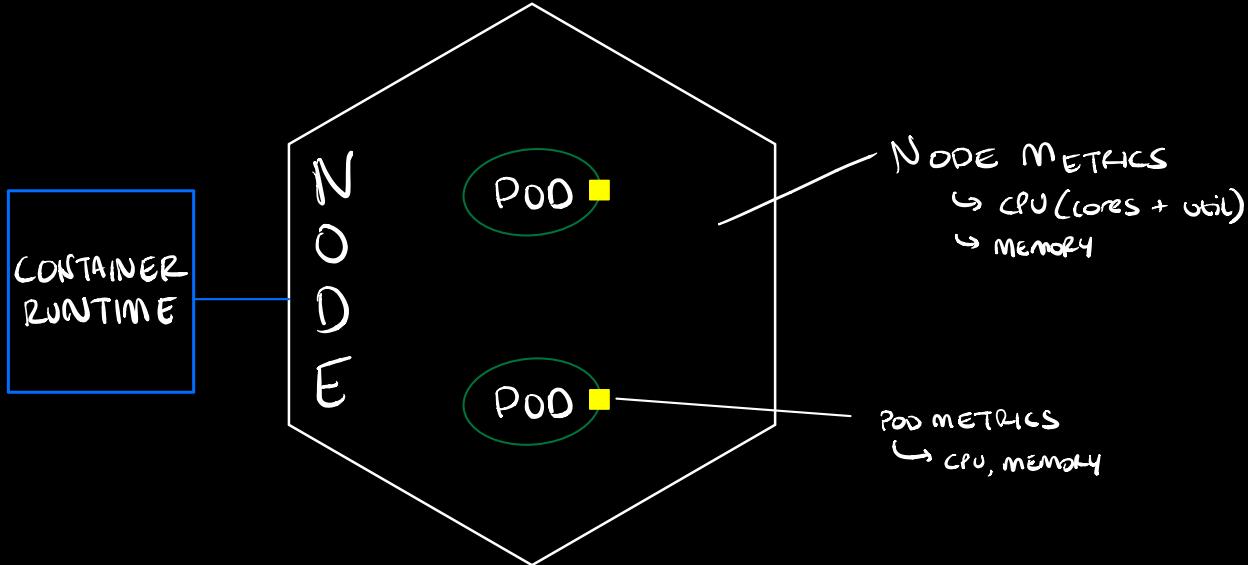




MONITORING CLUSTER COMPONENTS

MONITORING THE CLUSTER COMPONENTS

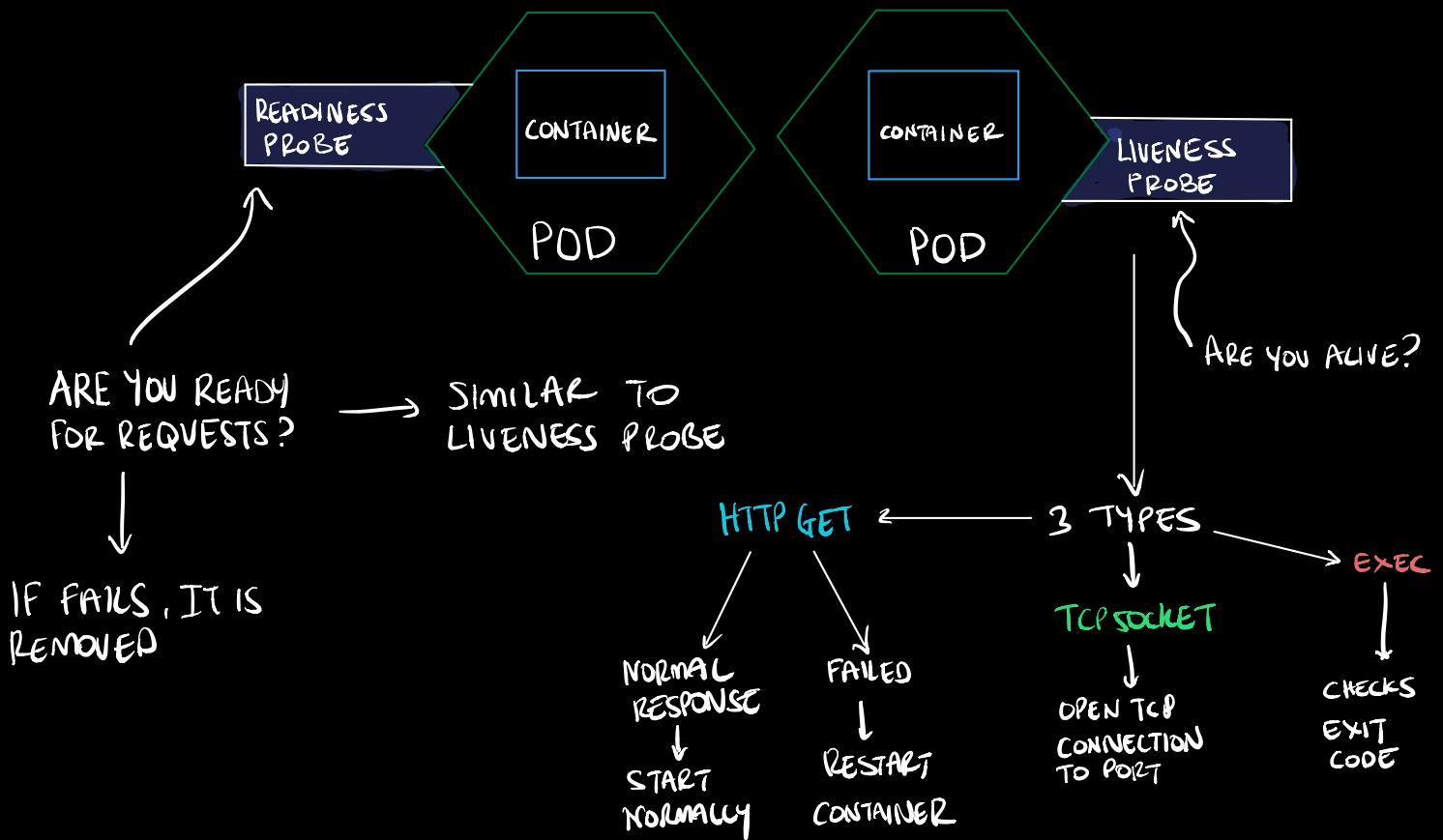
THE METRIC SERVER ALLOWS YOU TO COLLECT CPU AND MEMORY DATA FROM THE NODES AND PODS IN YOUR CLUSTER



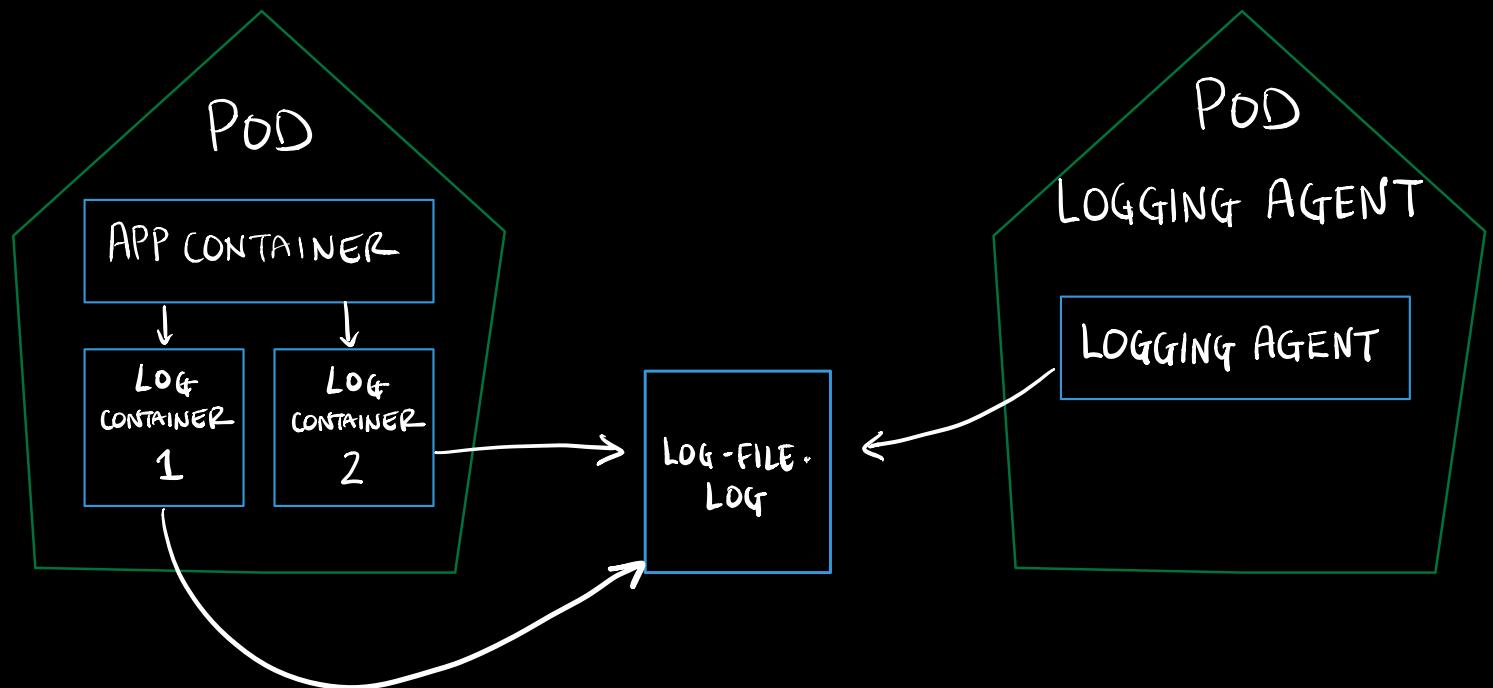
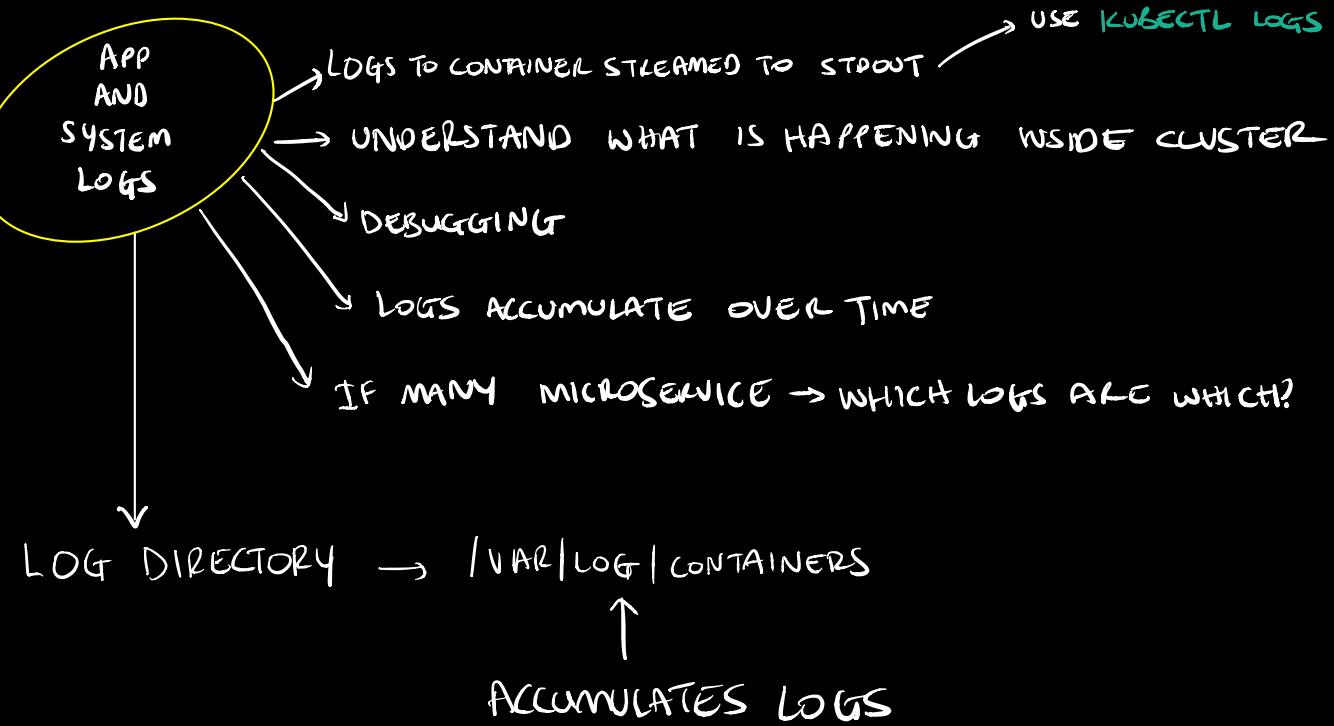
INSTALL METRIC SERVER → KUBECTL TOP NODE → CPU/MEMORY FOR ALL THE NODES
KUBECTL TOP POD → CPU/MEMORY FOR ALL THE PODS
KUBECTL TOP POD --ALL-NAMESPACES → ALL NAMESPACES
KUBECTL TOP POD -N KUBE-SYSTEM → KUBE-SYSTEM NAMESPACE
KUBECTL TOP GROUP-CONTEXT --CONTAINERS → PODS CONTAINERS

MONITORING THE APPS RUNNING WITHIN A CLUSTER

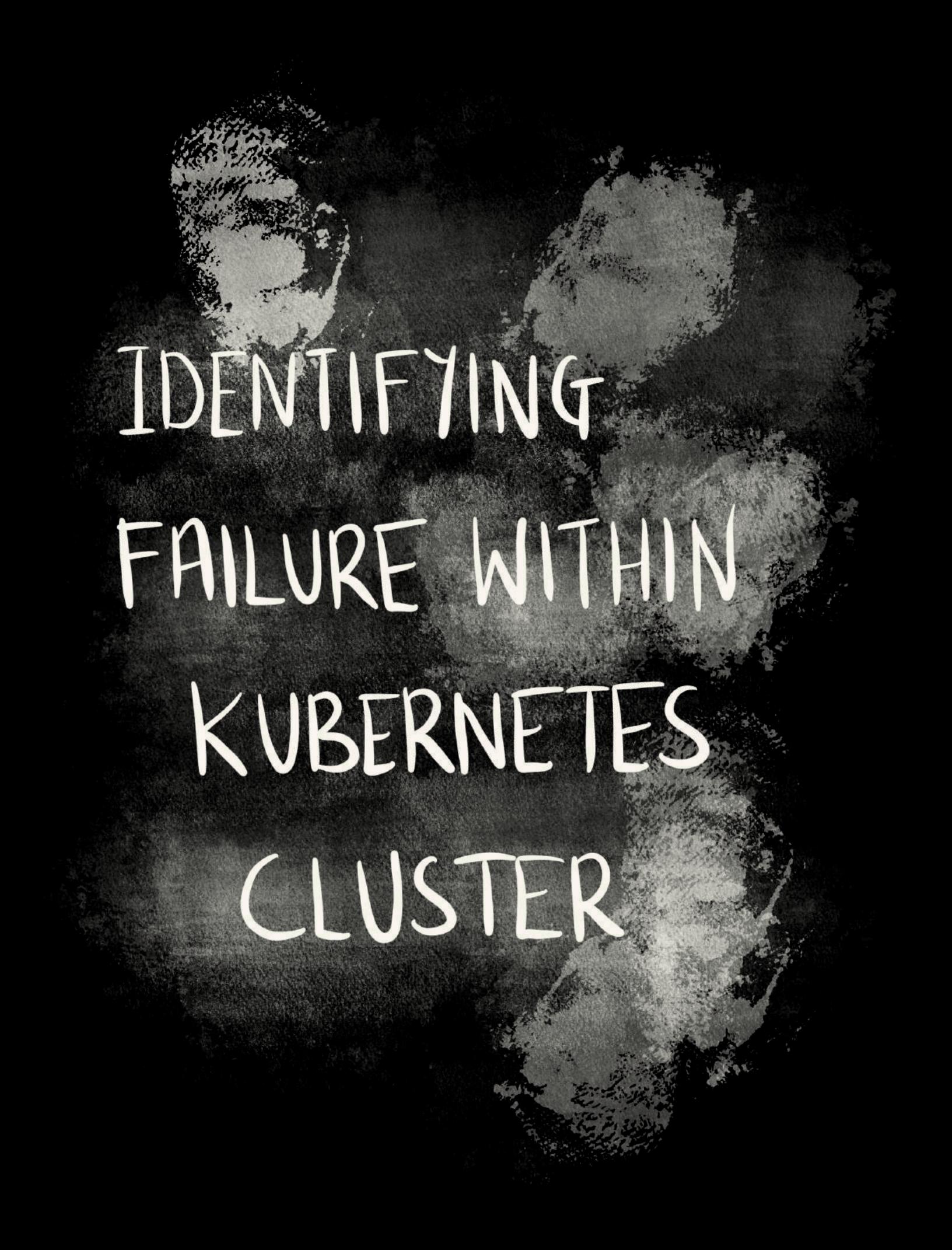
LIVENESS AND READINESS PROBE



MANAGING CLUSTER COMPONENT LOGS



- HAVE SIDEKAR CONTAINER TO DO LOGGING SO YOU CAN ACCESS SPECIFIC LOGS
- ABLE TO ROTATE LOGS USING OTHER TOOLS → NO NATIVE



IDENTIFYING FAILURE WITHIN KUBERNETES CLUSTER

TROUBLESHOOTING APPLICATION FAILURE

ABILITY TO WRITE
TERMINATION MESSAGE
TO SPECIFIC FILE ON
CONTAINER



POD1.yaml

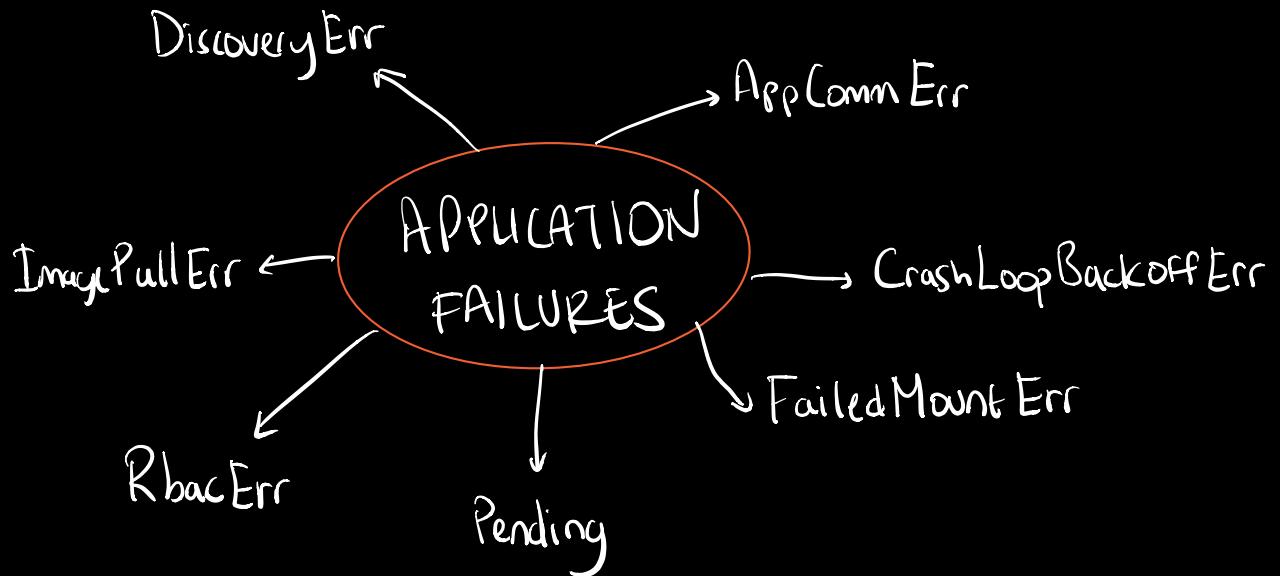
```
KIND: POD
NAME: POD1
IMAGE: busybox
COMMAND:
...
TERMINATION MESSAGE PATH
```

→ KUBECTL
DESCRIBE
↓
WILL SHOW
ERROR MESSAGE

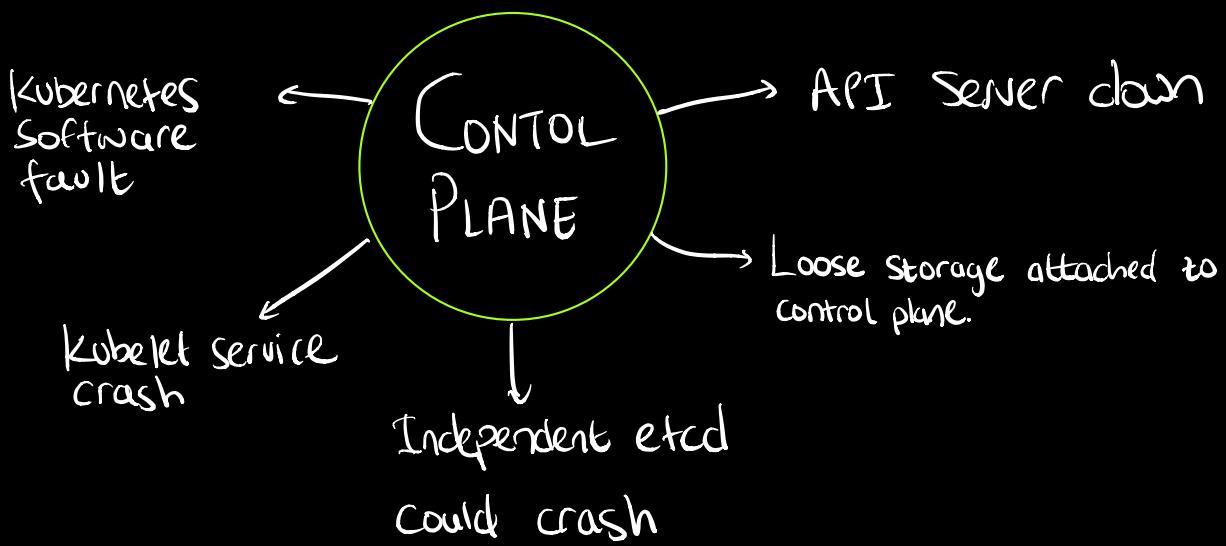
→ Only particular fields can be changed i.e Image

→ To change other fields of failed pod

↓
export configuration → -o yaml --export
modify yaml to change memory request

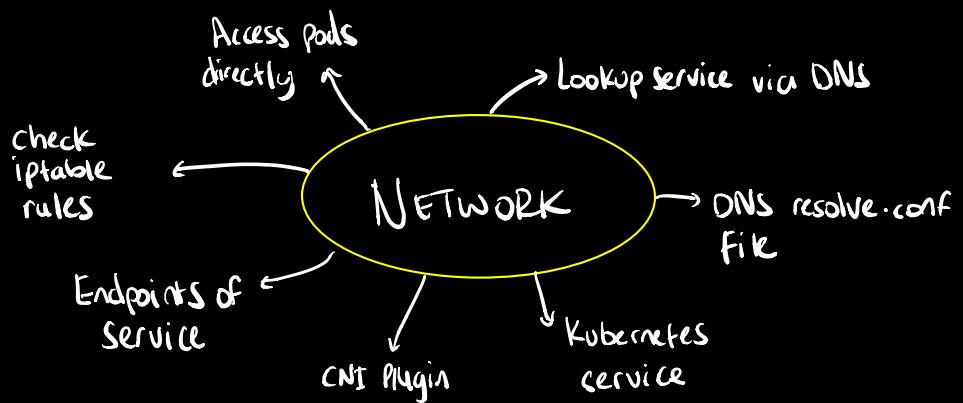
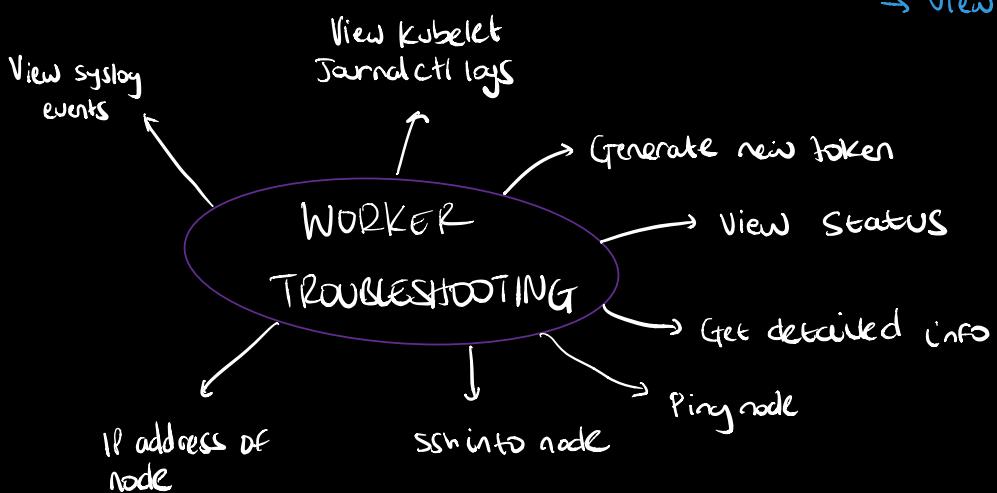


TROUBLESHOOT FAILURES



- View the events from control plane components
- View logs for control plane pods
- check status of docker service

- check status of kubelet service
- Disable swap
- check firewalld service
- View kube config



THANK You FOR READING

PLEASE LIKE AND SHARE FOR MORE

LATEST NOTES AVAILABLE ON

INSTAGRAM → adnans_techie_studies

ADNAN
RASHID
Z