

KUBERNETES WORKSHOP

- Introduction to Kubernetes
- Pods
- Deployments
- Statefulset
- Networking and Services
- RBAC
- Config map & secrets
- Setup -> Local + Kubeadm

INTRO TO K8S →

- What is Kubernetes?
- Why Kubernetes?
- Kubernetes Architecture
- YAML file
- Kubernetes components
- Imperative vs declarative
- Namespaces
- Labels and Selectors

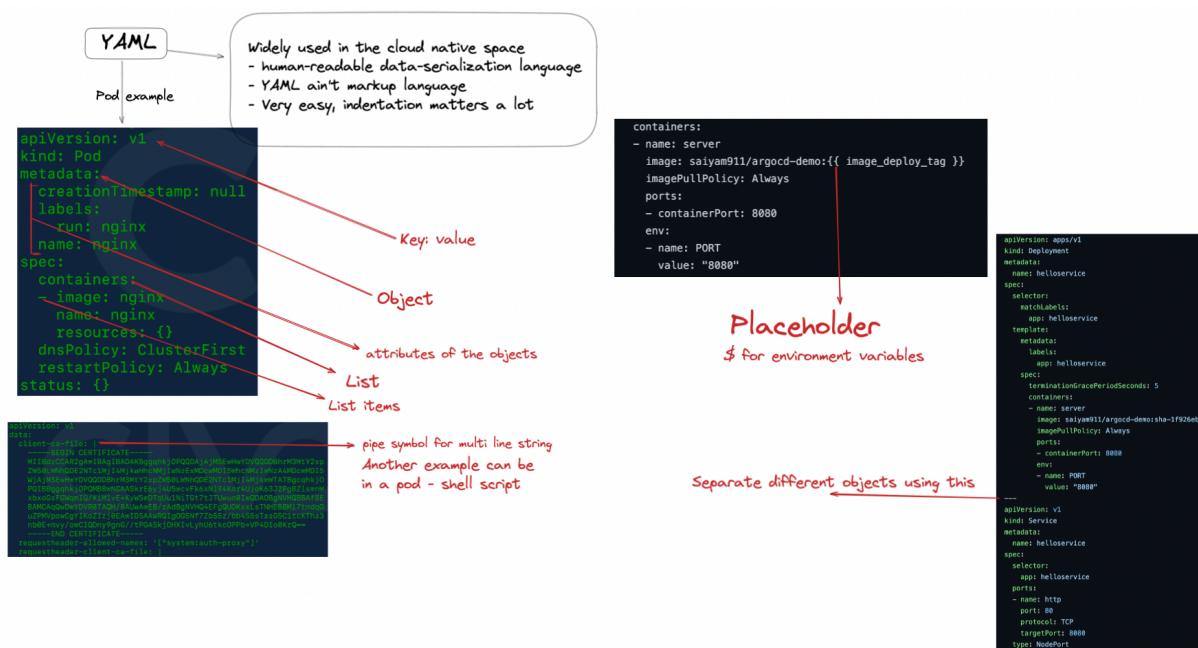
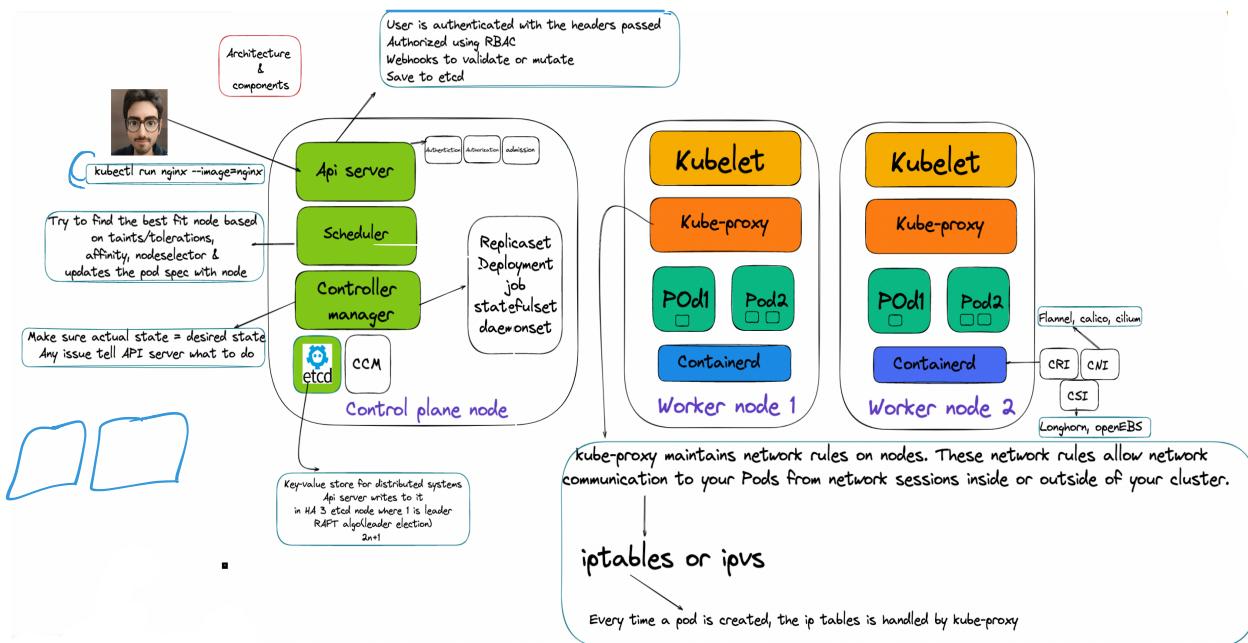
What is Kubernetes ?

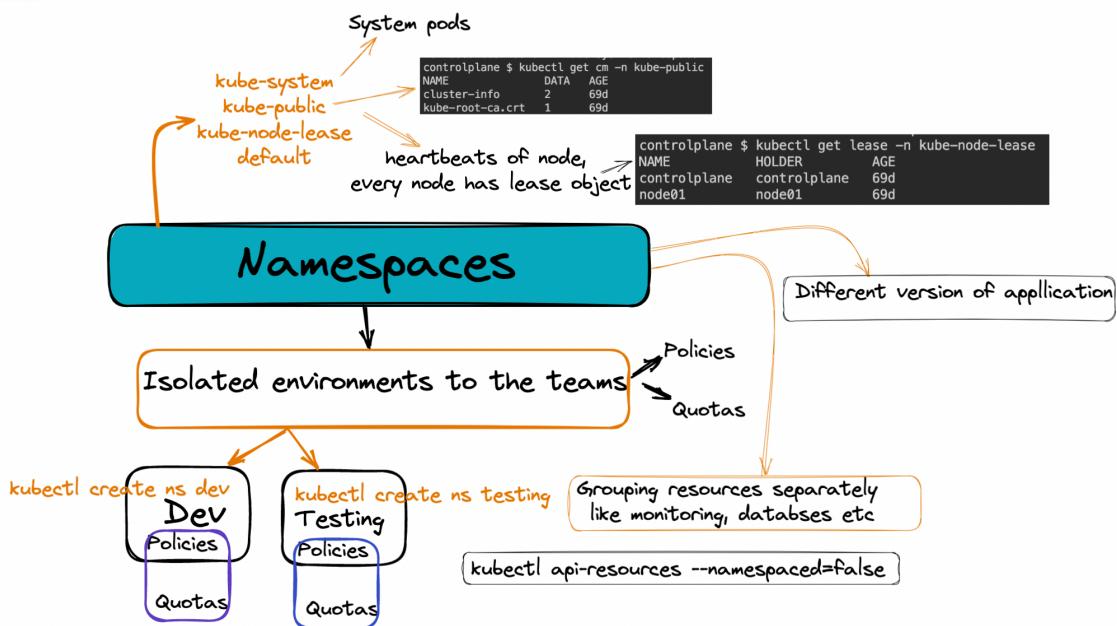
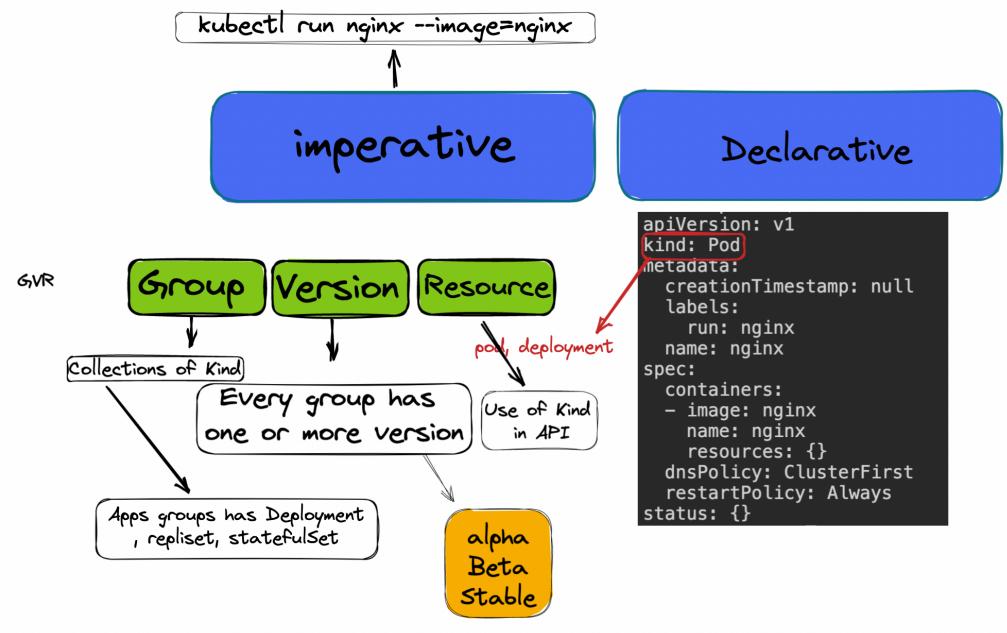
Kubernetes, also known as K8s, is an open-source system for automating deployment, scaling, and management of containerized applications.

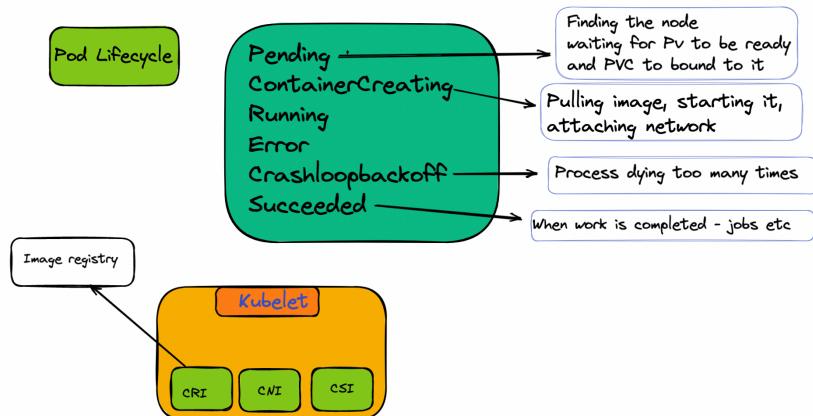
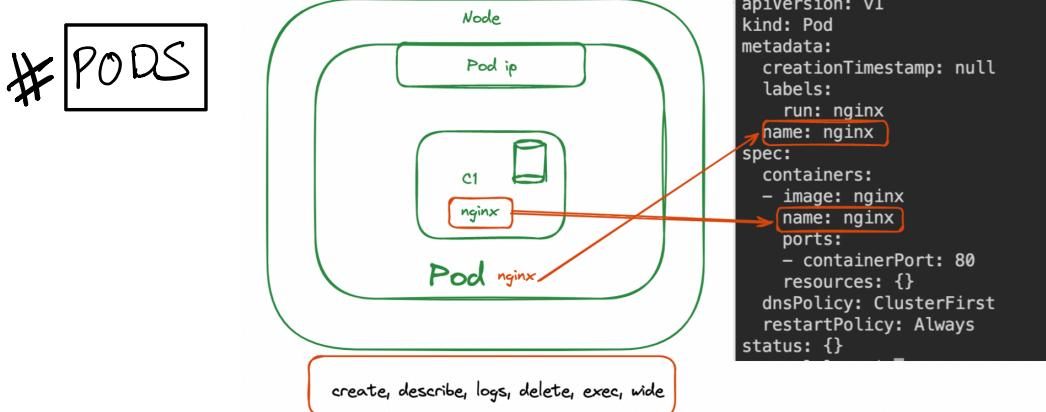
- CNCF Graduated
- Born out of Borg and Omega
- Launched 2014
- Designed for Scale
- Run anywhere

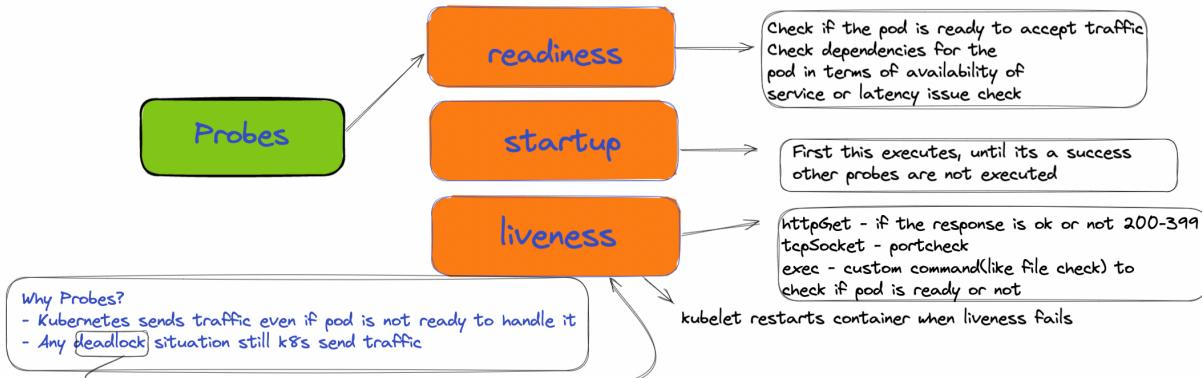
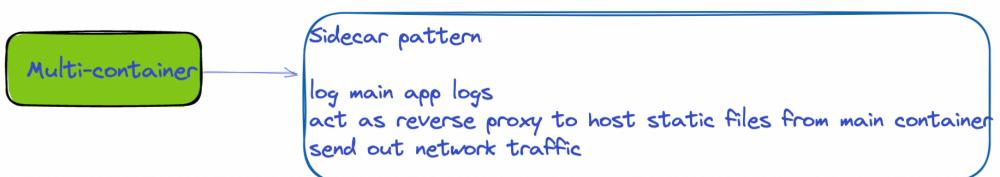
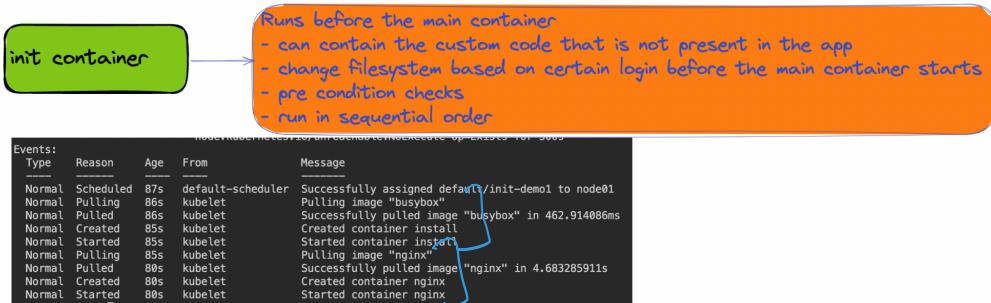
Why Kubernetes ?

- Few containers are fine but what about scale ?
- How do monitor them ?
- What about running pods on multiple nodes?
- What about flexibility ?
- Autoscaling?
- Scheduling
- Self healing capabilities.



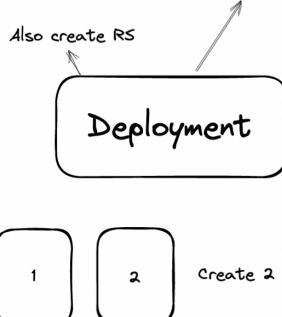






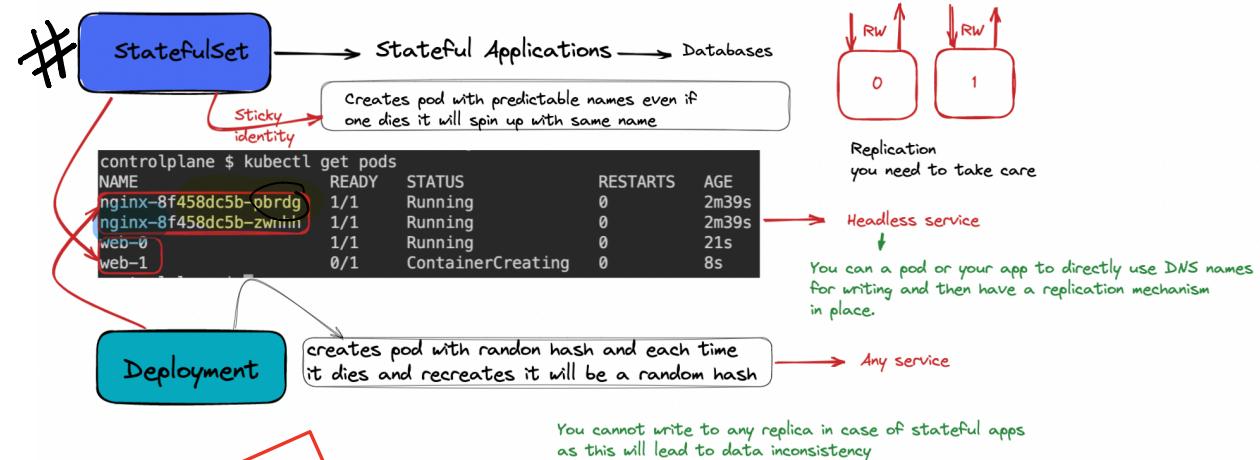
#Deployment

controlplane \$ kubectl get pods
 NAME READY STATUS RESTARTS AGE
 nginx-8f458dc5b-8fcf 1/1 Running 0 8s
 nginx-8f458dc5b-dzphw 1/1 Running 0 8s
 StrategyType: RollingUpdate
 MinReadySeconds: 0
 RollingUpdateStrategy: 25% max unavailable, 25% max surge



```
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: nginx
    name: nginx
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: nginx
    spec:
      containers:
        - image: nginx
          name: nginx
          resources: {}
status: {}
```

- When you update the deploy image
 - New pod gets created, old pod serves traffic until it finishes its terminationGracePeriod (30 by default)

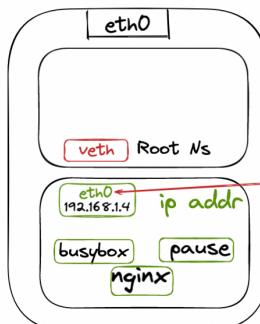


NETWORKING

What happens when you run the pod?

```
controlplane $ kubectl get pods
NAME      STATUS   ROLES   AGE   VERSION
controlplane   Ready   control-plane   6hd   v1.24.0
node01     Ready   <none>    6hd   v1.24.0
controlplane $ 
controlplane $ cat pod.yaml
apiVersion: v1
kind: Pod
metadata:
  name: shared-namespace
spec:
  containers:
    - name: p1
      image: busybox
      command: ['"/bin/sh', '-c', 'sleep 10000']
    - name: p2
      image: nginx
controlplane $ kubectl apply -f pod.yaml
pod/shared-namespace created
```

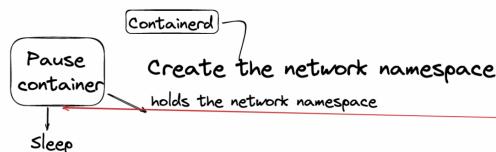
After pod creation CNI attach IP address and attach it to network



```
controlplane $ kubectl exec -it shared-namespace -- sh
Defaulted container "p1" out of: p1, p2
# ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue qlen 1000
  Link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
  inet 127.0.0.1/8 brd 127.255.255.255 scope host lo
    valid_lft forever preferred_lft forever
  inet6 ::1/128 brd :: scope host
    valid_lft forever preferred_lft forever
3: eth0@if19: <NOFORWARD,BROADCAST,MULTICAST,UP,LOWER_UP,M-DOWN> mtu 1460 qdisc noqueue
  Link/ether be:26:a7:95:24:17 brd ff:ff:ff:ff:ff:ff
  inet 192.168.1.4/32 brd 192.168.1.4 scope global eth0
    valid_lft forever preferred_lft forever
  inet6 fe80::bc26:a7ff:fe95:2417/64 scope link
    valid_lft forever preferred_lft forever
/ # route
Kernel IP routing table
Destination     Gateway         Genmask        Flags Metric Ref    Use Iface
default         169.254.1.1   0.0.0.0       UG        0      0      0 eth0
169.254.1.1   *             255.255.255.255  UH        0      0      0 eth0
```

ip netns list
List of network namespaces

There are veth pairs created or depending on CNI maybe different



To see that network namespace is held by pause container

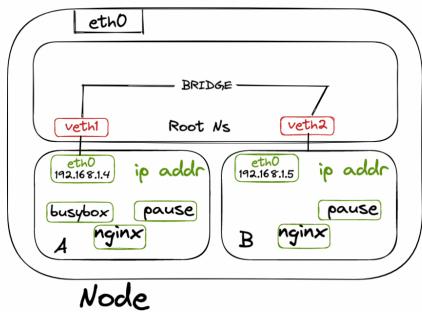
```
node01 $ lsns | grep nginx
4026532987 mnt          2 32059 root           nginx: master process nginx -g daemon off;
4026532988 net          2 32059 root           nginx: master process nginx -g daemon off;

node01 $ lsns -p 32059
NS TYPE    NRPROC PID USER COMMAND
4026531835 cgroup    157   1 root   /sbin/init
4026531837 user     157   1 root   /sbin/init
4026532518 net       3 31073 65535 /pause
4026532584 uts      3 31973 65535 /pause
4026532587 mnt      2 32059 root           nginx: master process nginx -g daemon off;
4026532588 pid      2 32059 root           nginx: master process nginx -g daemon off;
```

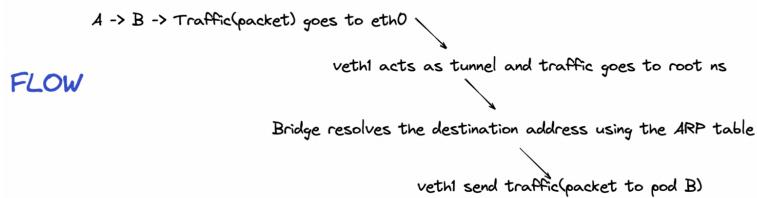
ls -lt /var/run/netns

List of namespaces created

```
node01 $ ls -lt /var/run/netns
total 0
-r--r--r-- 1 root root 0 Jul 16 14:25 cni-3cf655f5-9131-e23d-0756-b7ab6556ef8f
-r--r--r-- 1 root root 0 May 8 19:46 cni-06382f28-d5db-63f3-00ee-40e904df
-r--r--r-- 1 root root 0 May 8 19:46 cni-592cc23b-1d2b-6054-bf32-437fd23a04df
node01 $ ip netns exec cni-3cf655f5-9131-e23d-0756-b7ab6556ef8f ip link
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
  link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
3: eth0@if9: <NOFORWARD,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc noqueue state UP mode DEFAULT group default
  link/ether 9e:bd:c7:d4:e5:99 brd ff:ff:ff:ff:ff:ff link-netnsid 0
node01 $ ip link | grep -A1 9
9: calic40f455693@if3: <NOFORWARD,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1460 qdisc noqueue state UP mode DEFAULT group default
  link/ether ee:ee:ee:ee:ee:ee brd ff:ff:ff:ff:ff:ff link-netnsid 1
node01 $
```

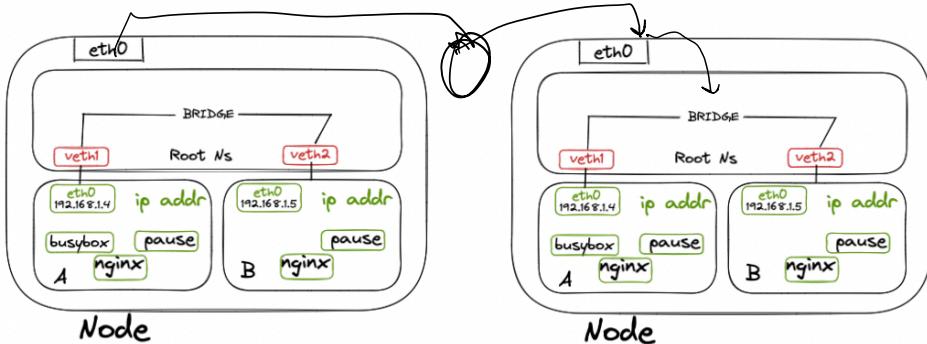


Inter node pod communication



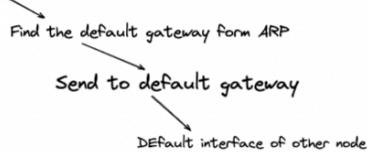
Node to node communication

Cannot use ARP as destination is not on same node



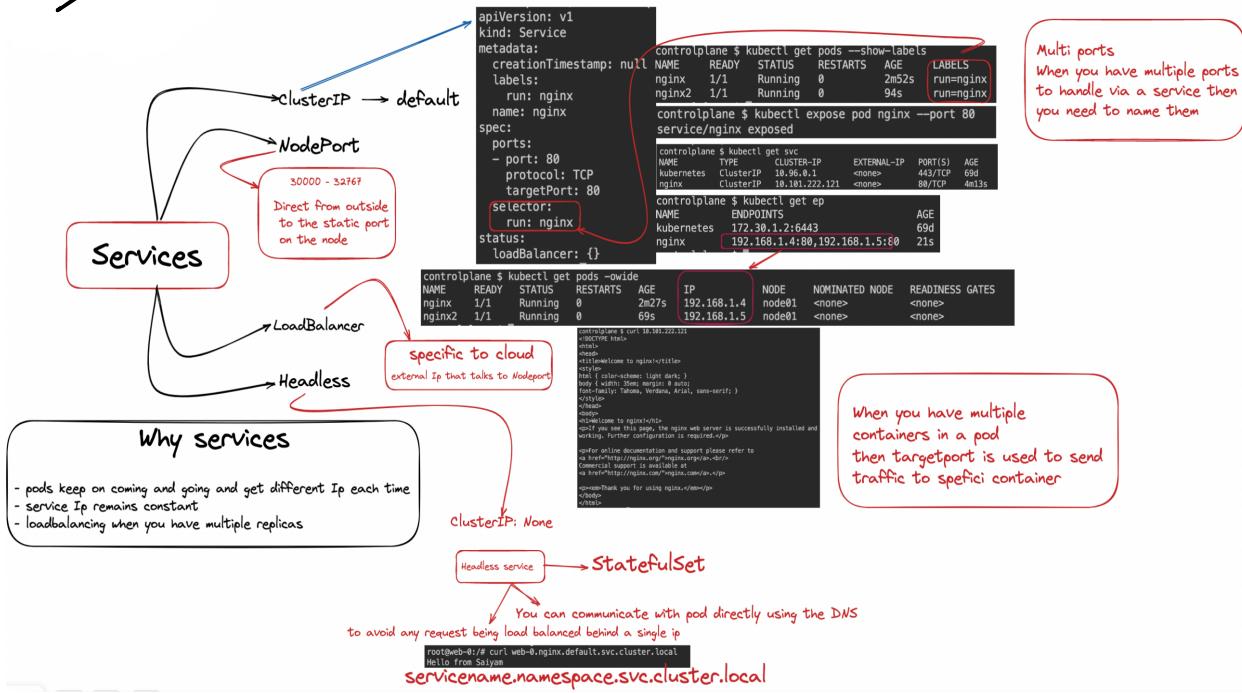
FLOW

To check if the destination is not on the same network bitwise or Adding operation is performed

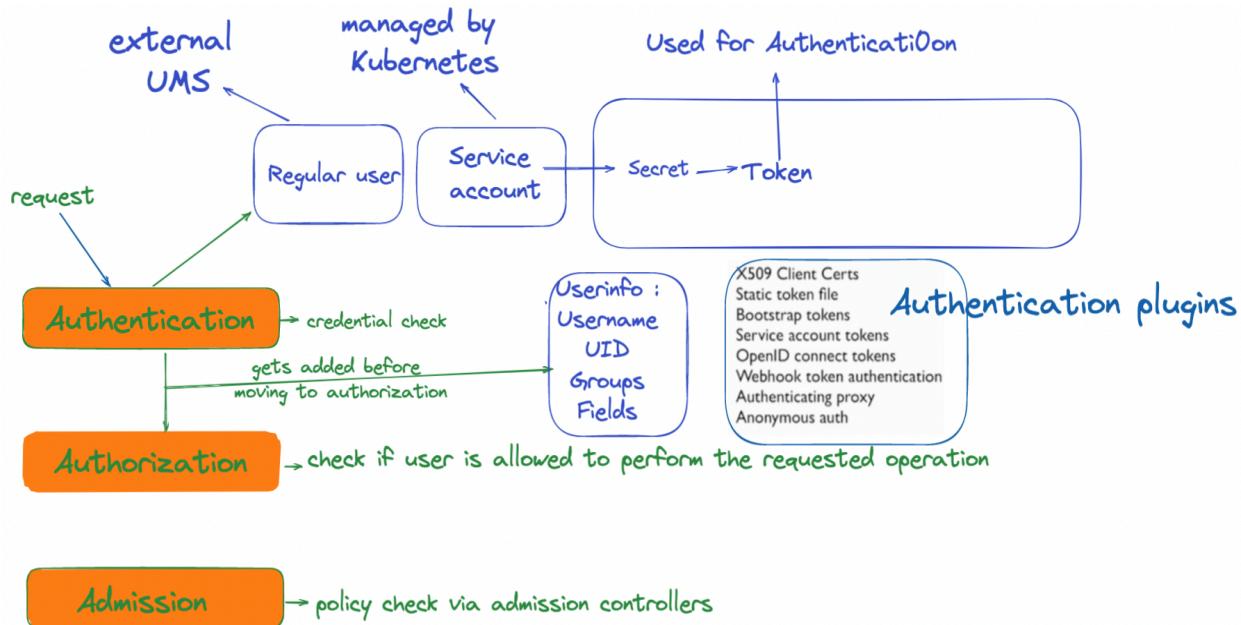


Services — iptables and netfilter

SERVICES



AAA



*Config map
secrets*

