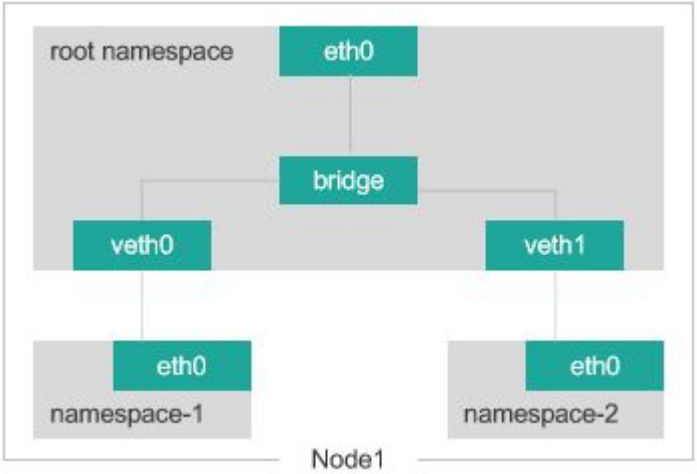
Networking with Kubernetes!

Part 1

Intro: K8S Networking

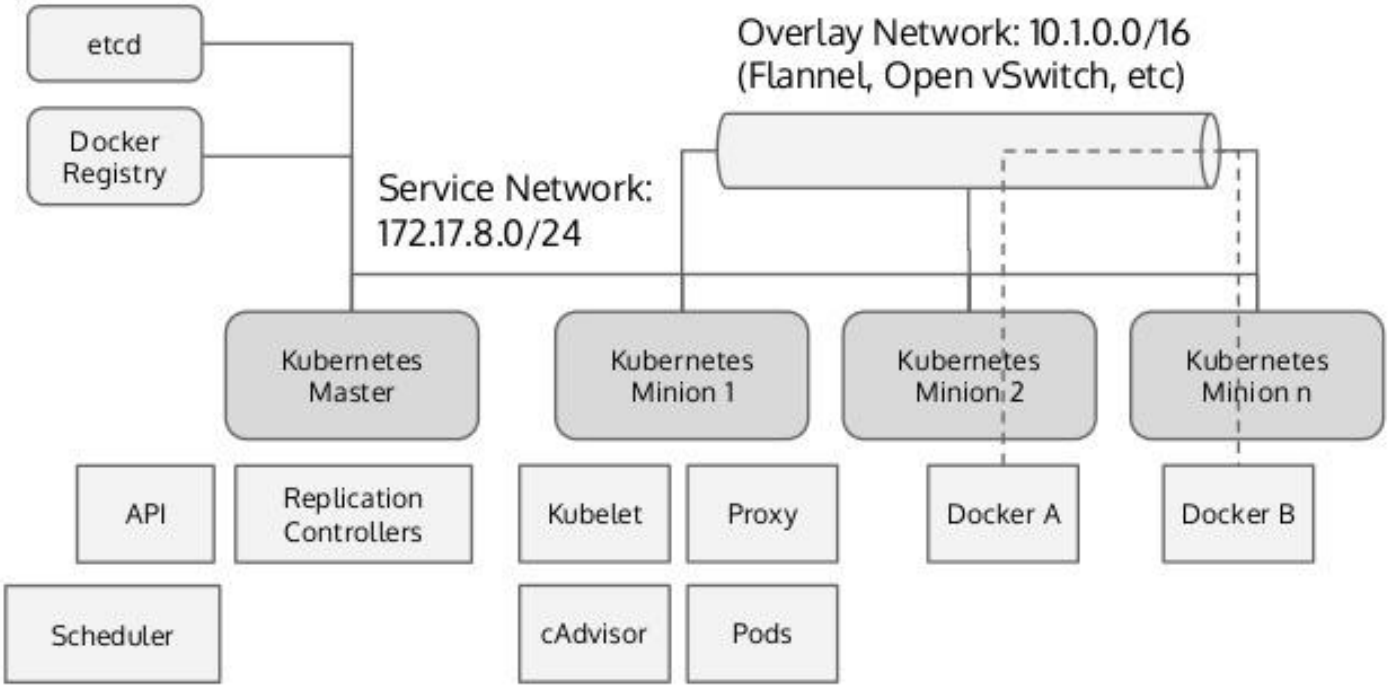
Namespace

- Linux kernel has 6 types of namespaces: *pid,net,mnt,uts,ipc,user*
- Network namespaces provide a brand-new network stack for all the processes within the namespace.
- That includes network interfaces,routing tables and iptables rules

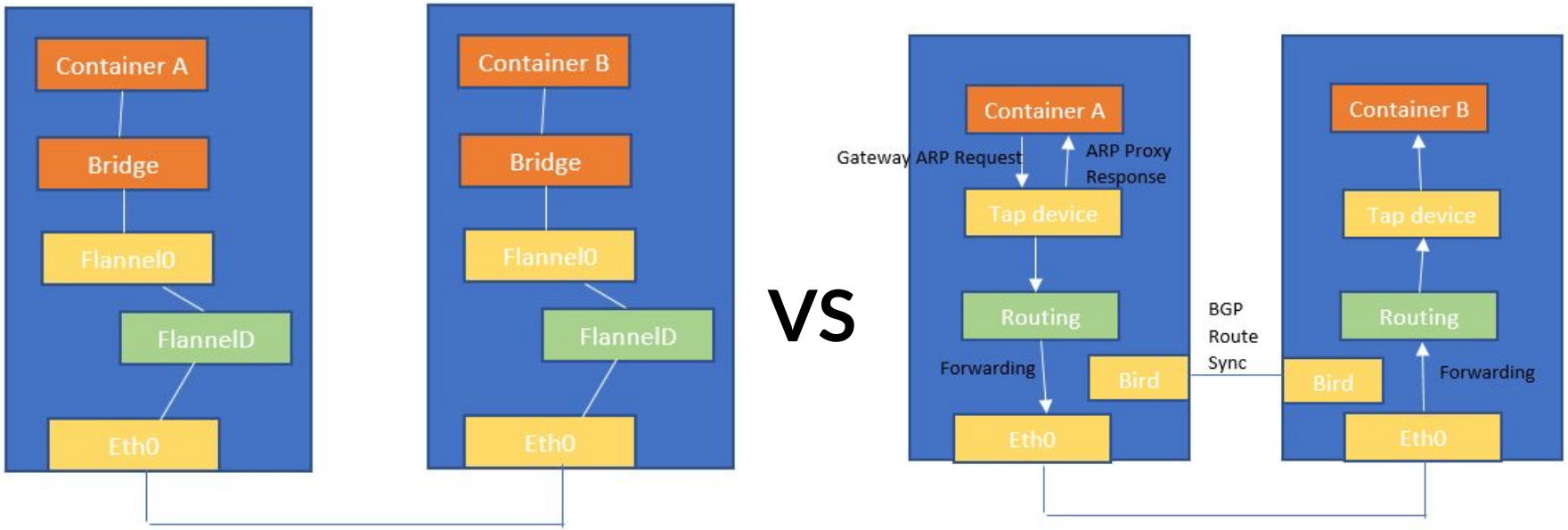


Live demo: Play with namespaces and cgroups.

Kubernetes Architecture



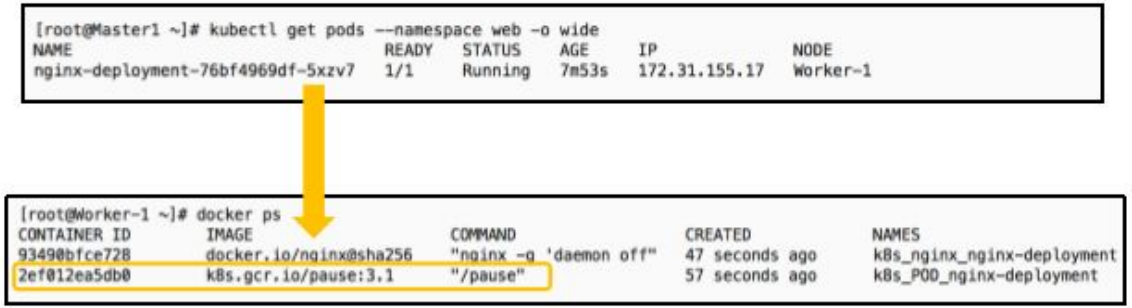
Flannel VS Calico



Pods

Pods

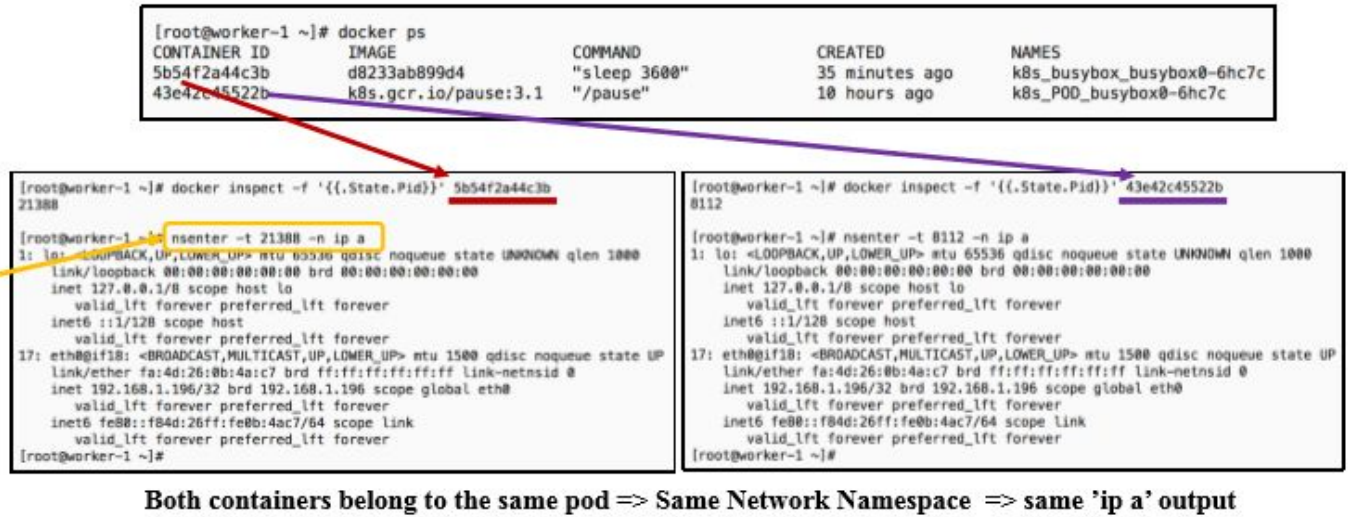
- Lowest common denominator in K8S. Pod is comprised of one or more containers along with a “pause” container
- Pause container act as the “parent” container for other containers inside the pod. One of it’s primary responsibilities is to bring up the network namespace
- Great for the redundancy: Termination of other containers do not result in termination of the network namespace



Accessing Pod Namespaces

Accessing Pod Namespaces

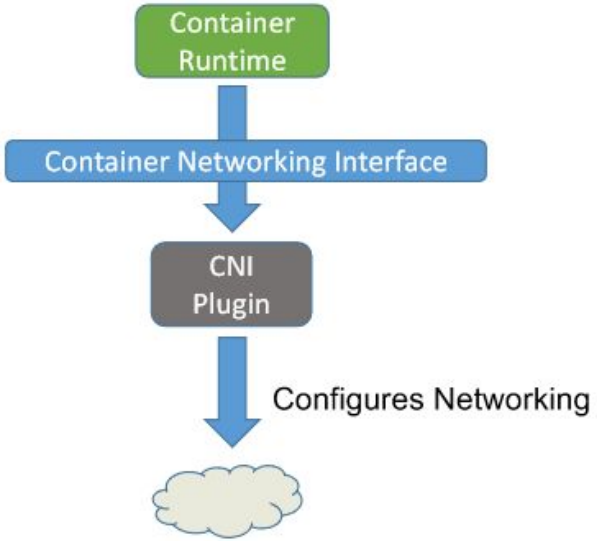
- Multiple ways to access pod namespaces
- ‘kubectl exec --it’
- ‘docker exec --it’
- nsenter (“**namespace enter**”, let you run commands that are installed on the host but not on the container)



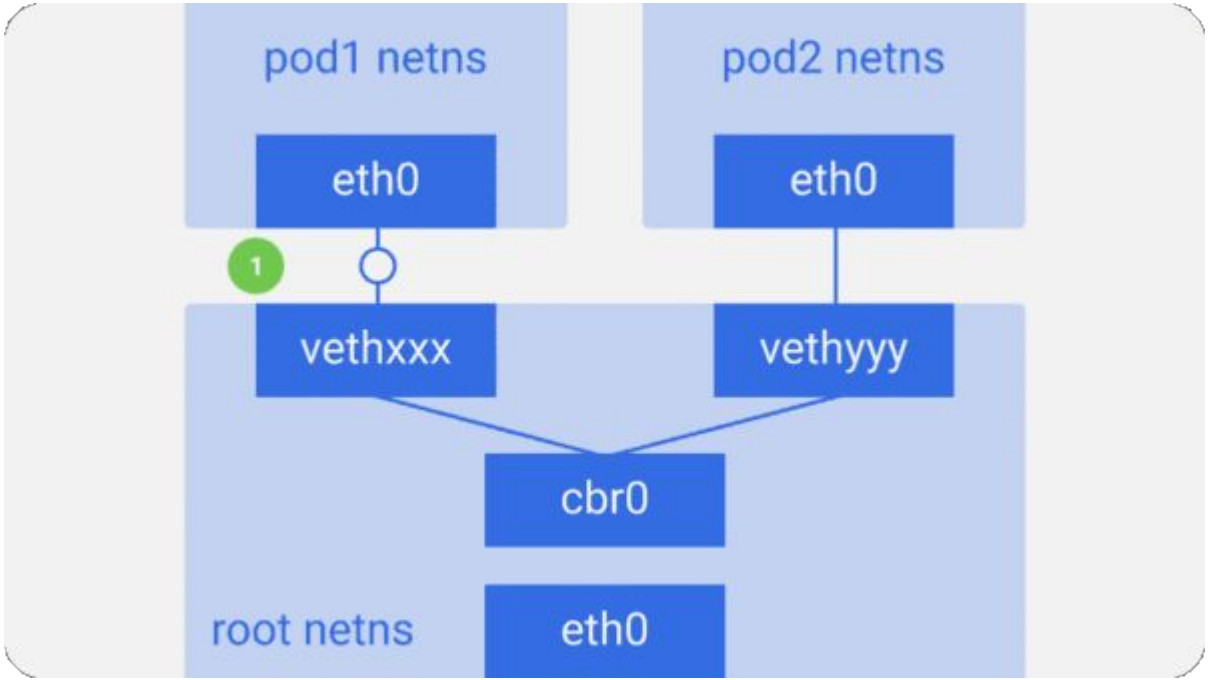
Both containers belong to the same pod => Same Network Namespace => same ’ip a’ output

Container Networking Interface : CNI

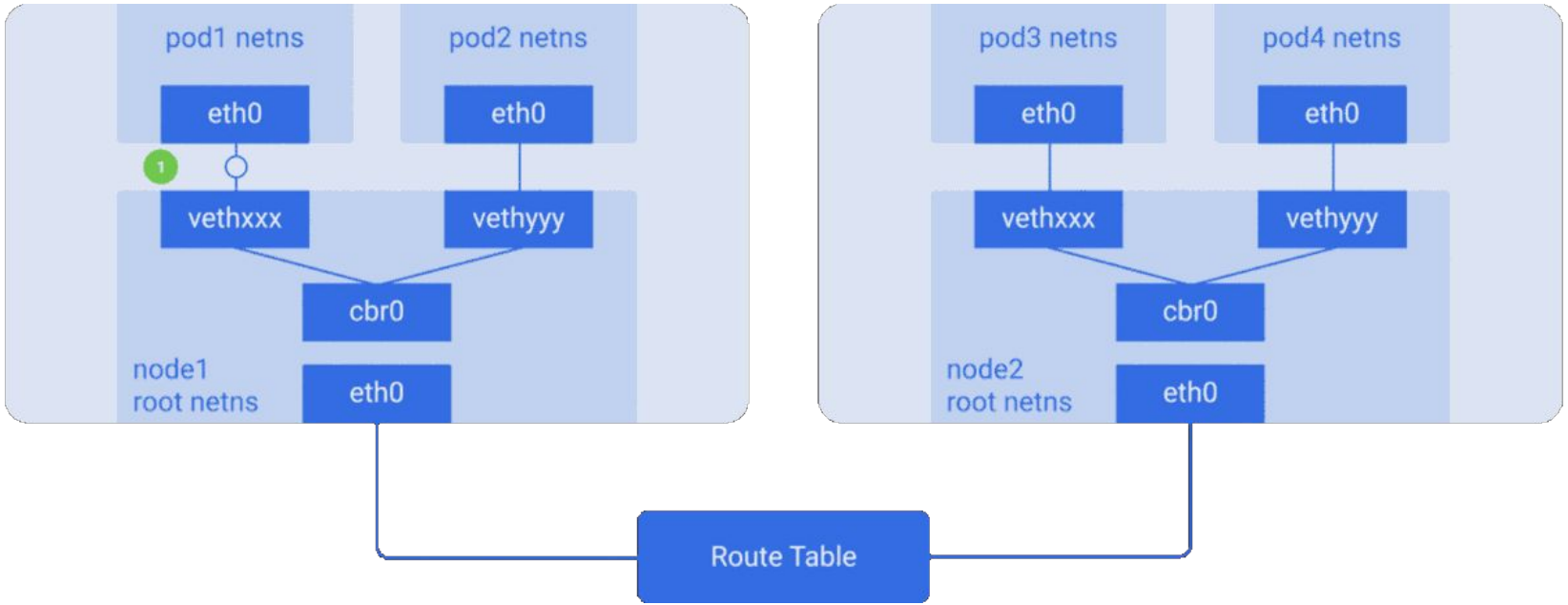
- Interface between container runtime and network implementation
- Network plugin implements the CNI spec. It takes a container runtime and configure (attach/detach) it to the network
- CNI plugin is an executable (in: /opt/cni/bin)
- When invoked it reads in a JSON config & Environment Variables to get all the required parameters to configure the container with the network



Intra-node communication



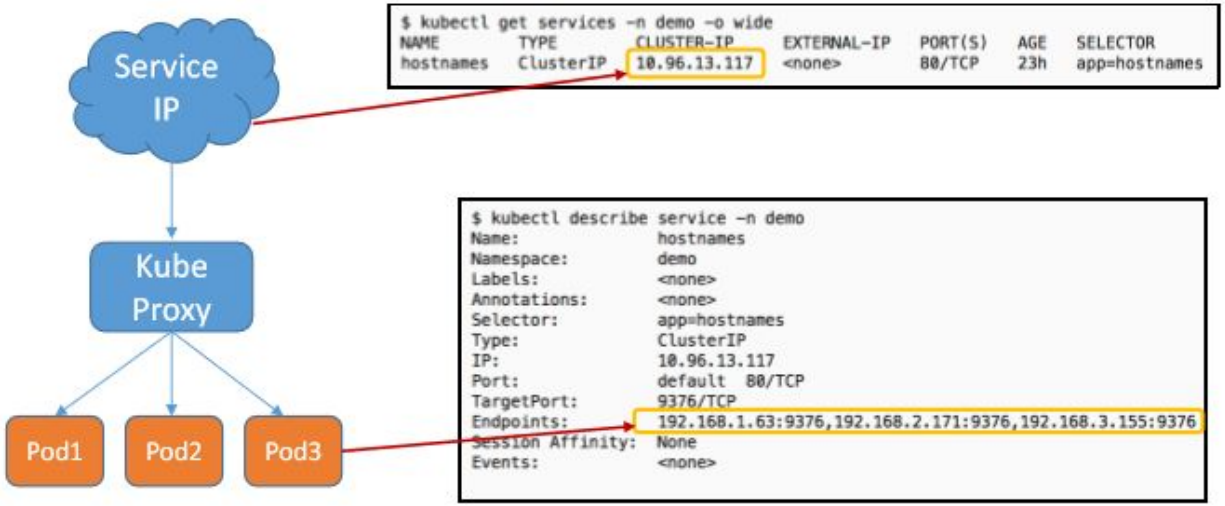
Inter-node communication



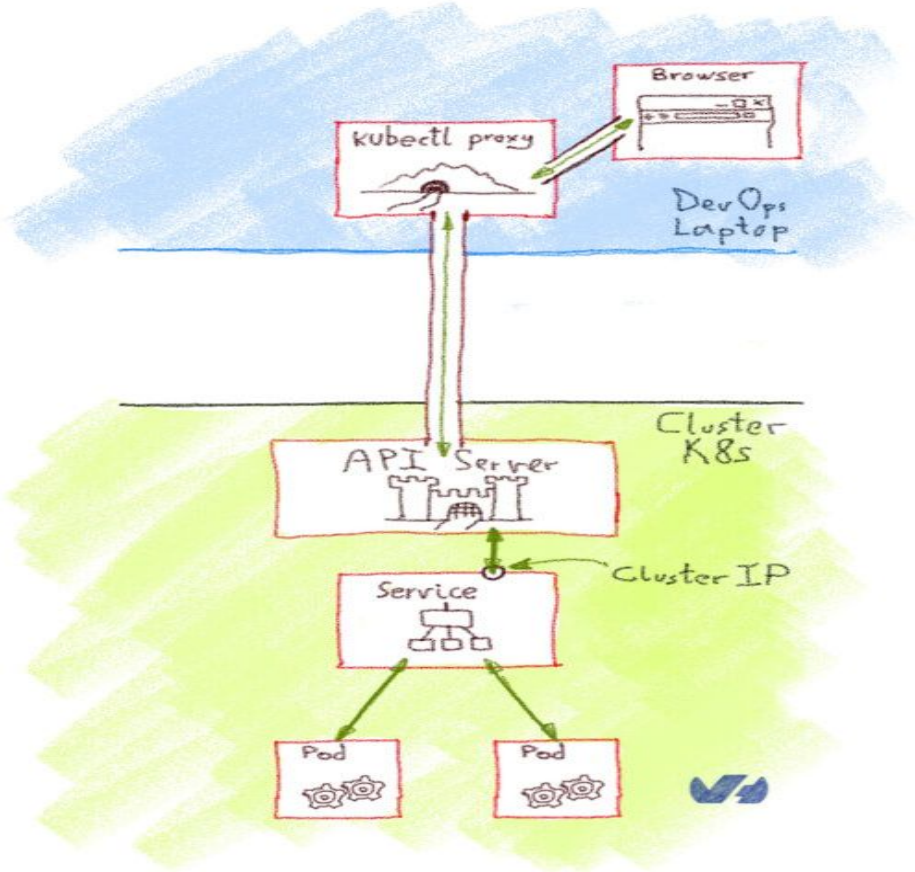
Back to the Basics

Services

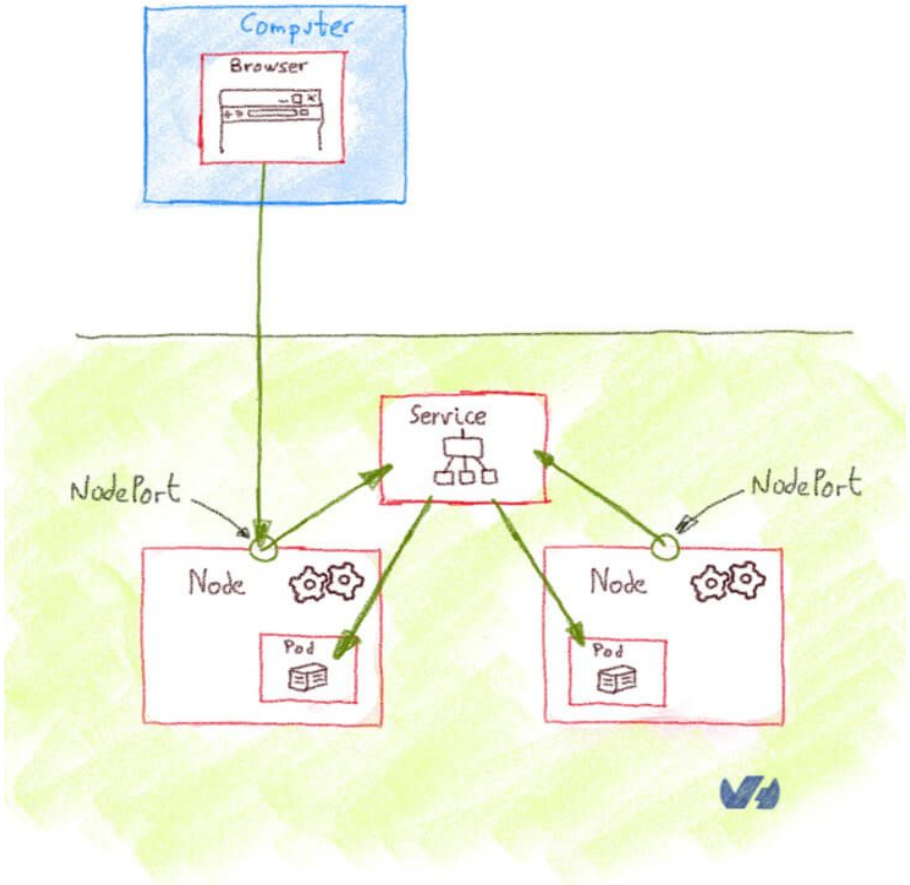
- Pods are mortal
- Need a higher level abstractions: Services
- “Service” in Kubernetes is a conceptual concept. Service is not a process/daemon. Outside networks doesn’t learn Service IP addresses
- Implemented through Kube Proxy with IPTables rules



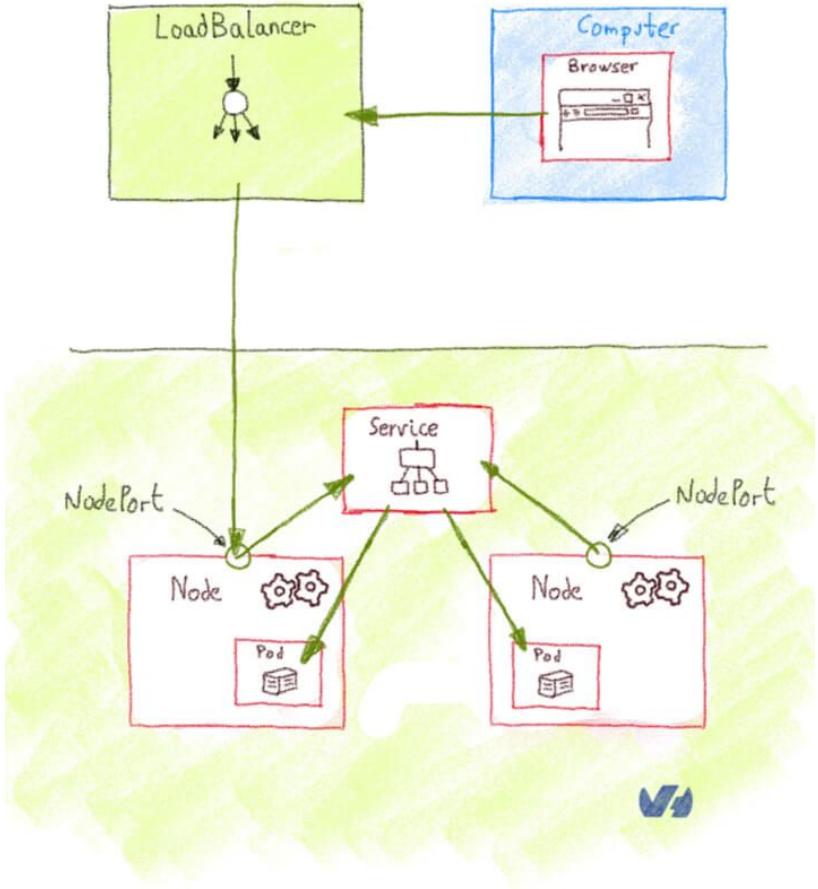
ClusterIP: Service is accessed via 'ClusterIP'



NodePort: Service is accessed via ‘NodeIP:port’

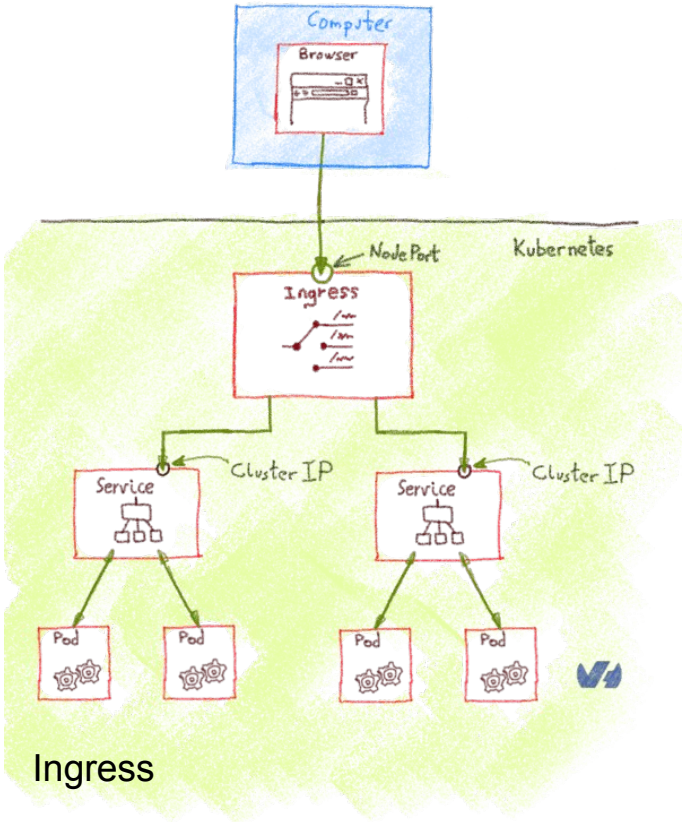


LoadBalancer: Service is accessed via Loadbalancer

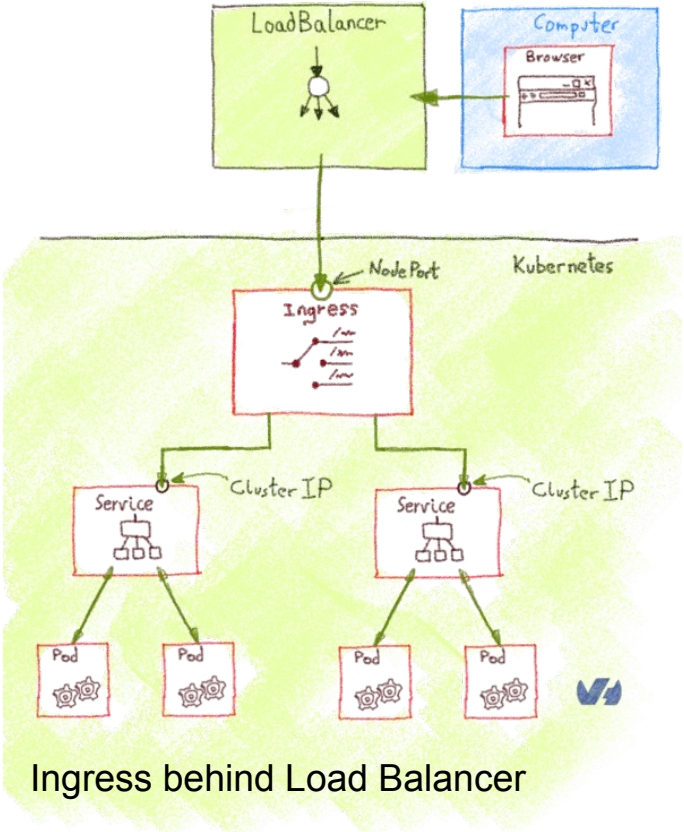


Ingress

Ingress is an API object that manages external access to the services in a cluster.



Ingress



Ingress behind Load Balancer

Part 2:

Setting up K8S The Hard Way

On top of Amazon Web Services (AWS)

1- Provisioning Compute Resources

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/03-compute-resources.md

➔ **Networking:**
- VPC
- Subnet
- Internet Gateway
- Route Tables
- Security Groups (aka Firewall Rules)

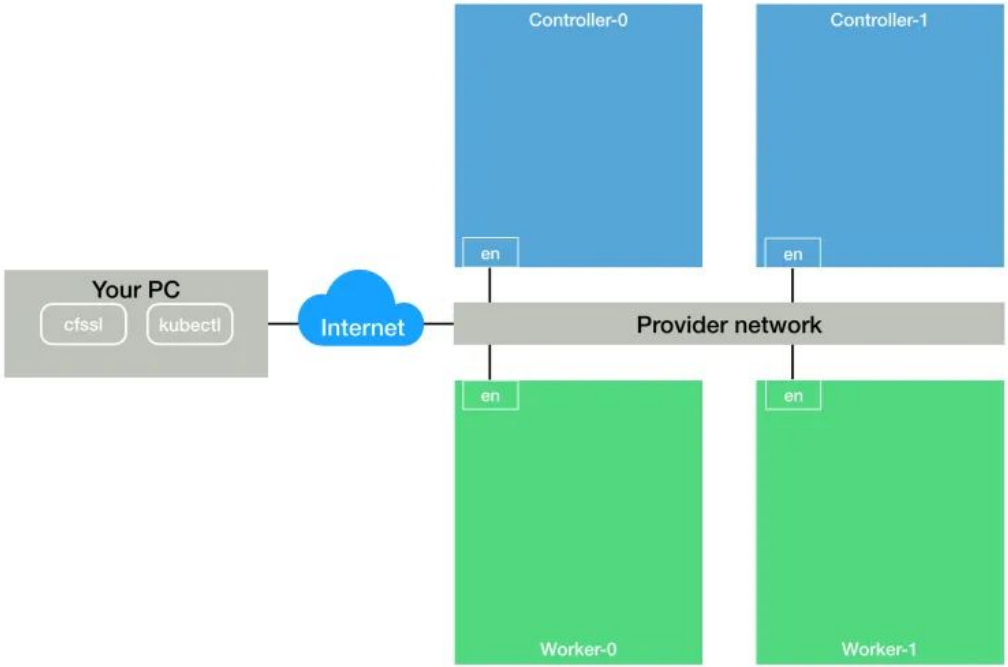
➔ Create a Network Load Balancer

➔ **Compute Instances:**

- Instance Image + SSH Key Pair
- Kubernetes Controllers
- Kubernetes Workers

1- Provisioning Compute Resources

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/03-compute-resources.md



2- Provisioning a CA and Generating TLS Certificates

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/04-certificate-authority.md

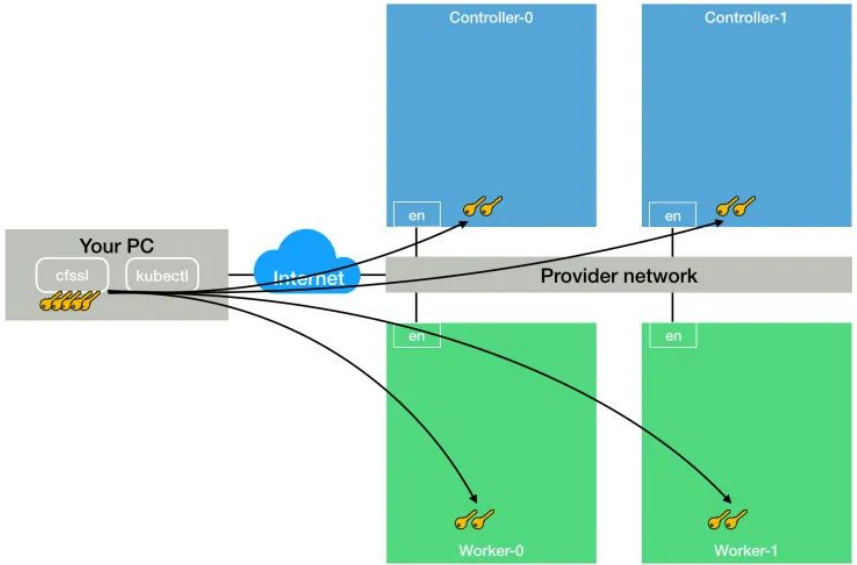
- **Certificate Authority**
- **Client and Server Certificates:**

➔ The Admin Client Certificate
➔ The Kubelet Client Certificates
➔ The Controller Manager Client Certificate
➔ The Kube Proxy Client Certificate
➔ The Scheduler Client Certificate
➔ The Kubernetes API Server Certificate

- **The Service Account Key Pair**
- **Distribute the Client and Server Certificates**

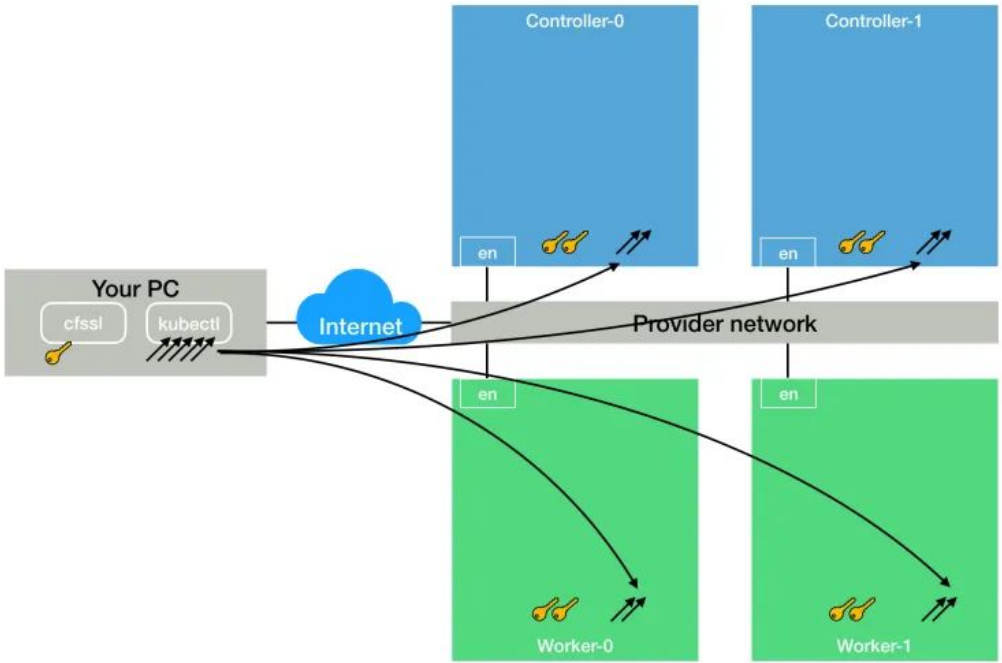
2- Provisioning a CA and Generating TLS Certificates

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/04-certificate-authority.md



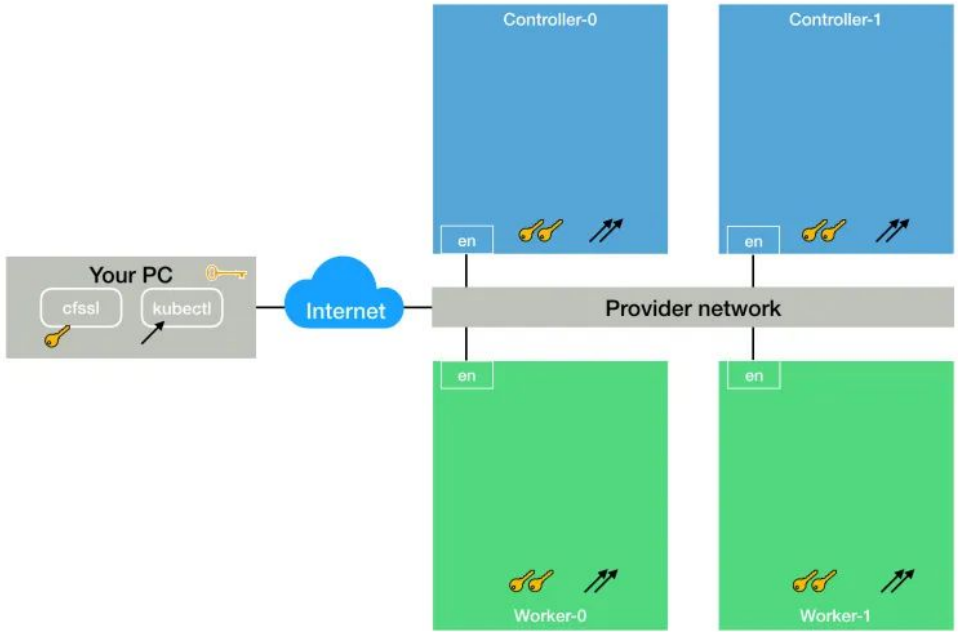
3- Generating Kubernetes Configuration Files for Authentication

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/05-kubernetes-configuration-files.md



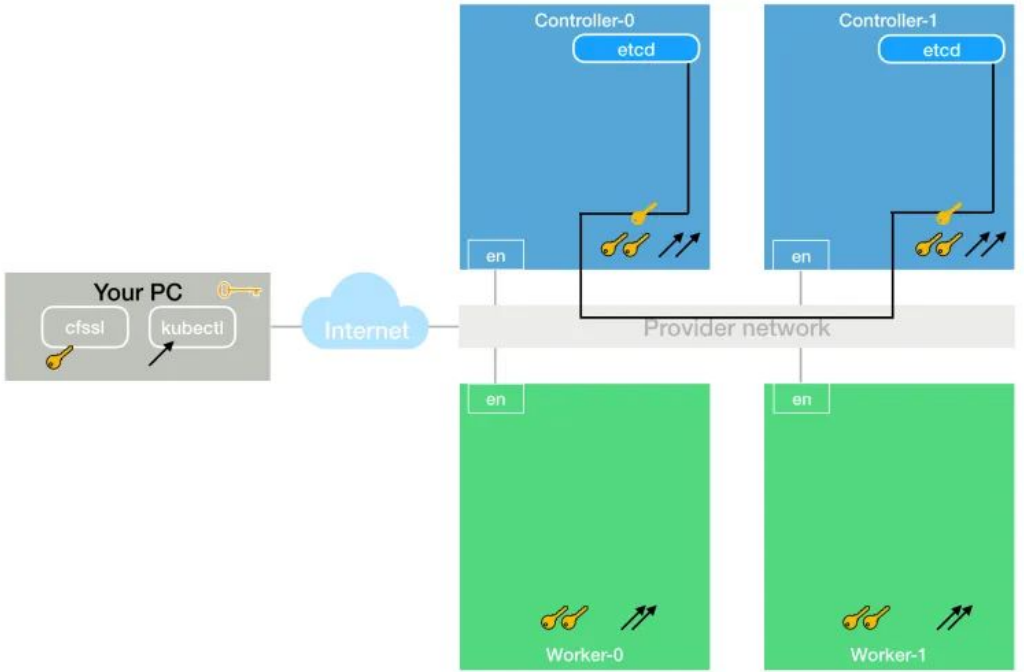
4- Generating the Data Encryption Config and Key

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/06-data-encryption-keys.md



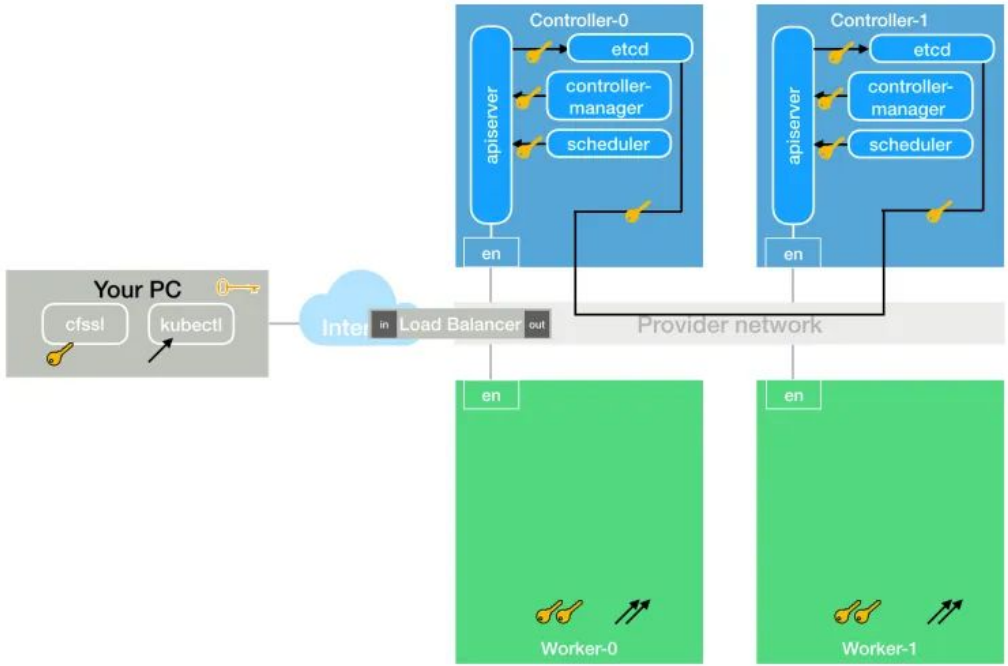
5- Bootstrapping the etcd Cluster

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/07-bootstrapping-etcd.md



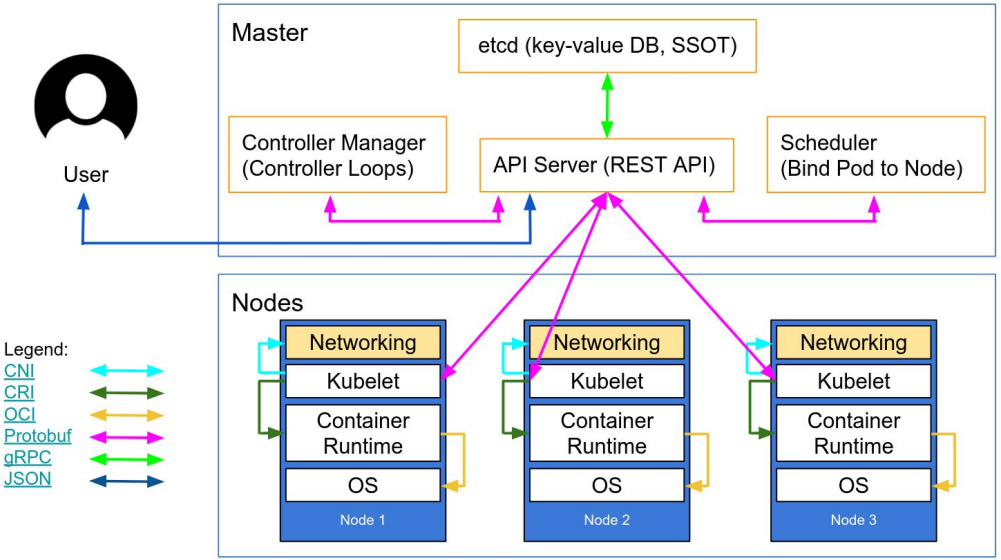
6- Bootstrapping the Kubernetes Control Plane

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/08-bootstrapping-kubernetes-controllers.md



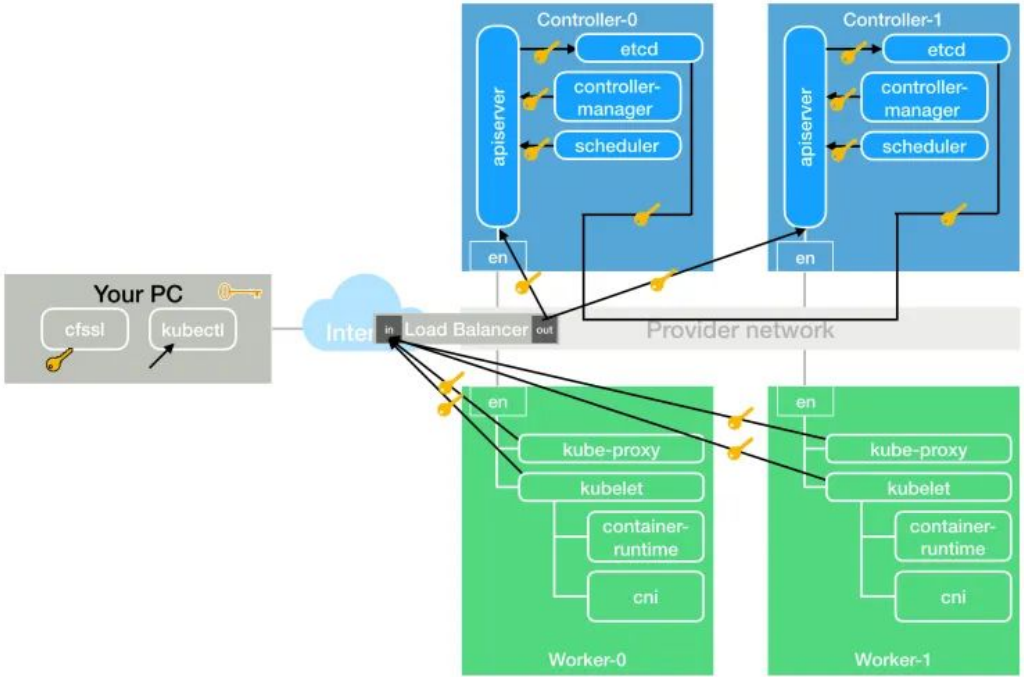
6- Bootstrapping the Kubernetes Control Plane

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/08-bootstrapping-kubernetes-controllers.md



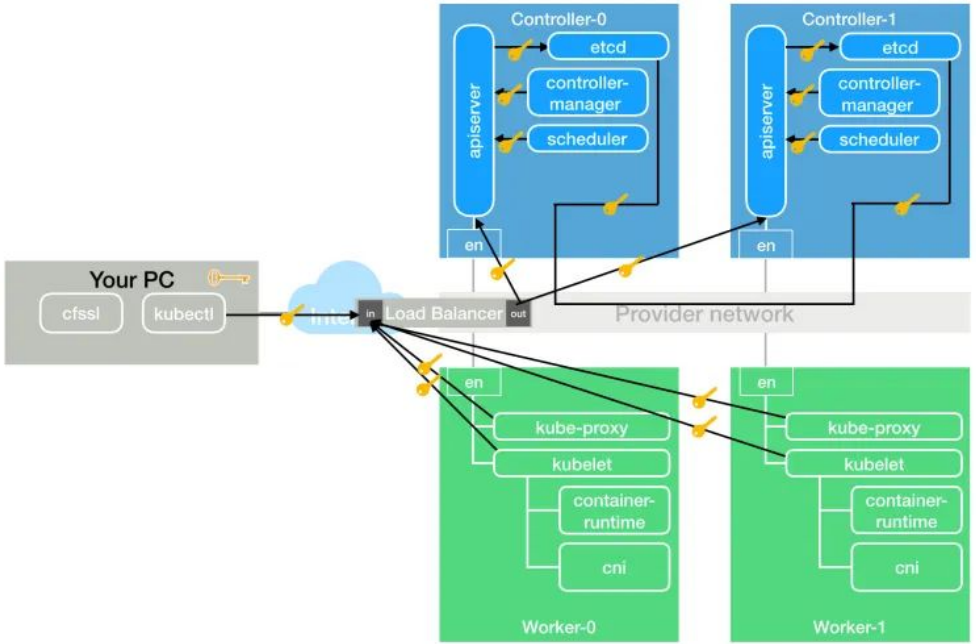
7- Bootstrapping the Kubernetes Worker Nodes

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/09-bootstrapping-kubernetes-workers.md



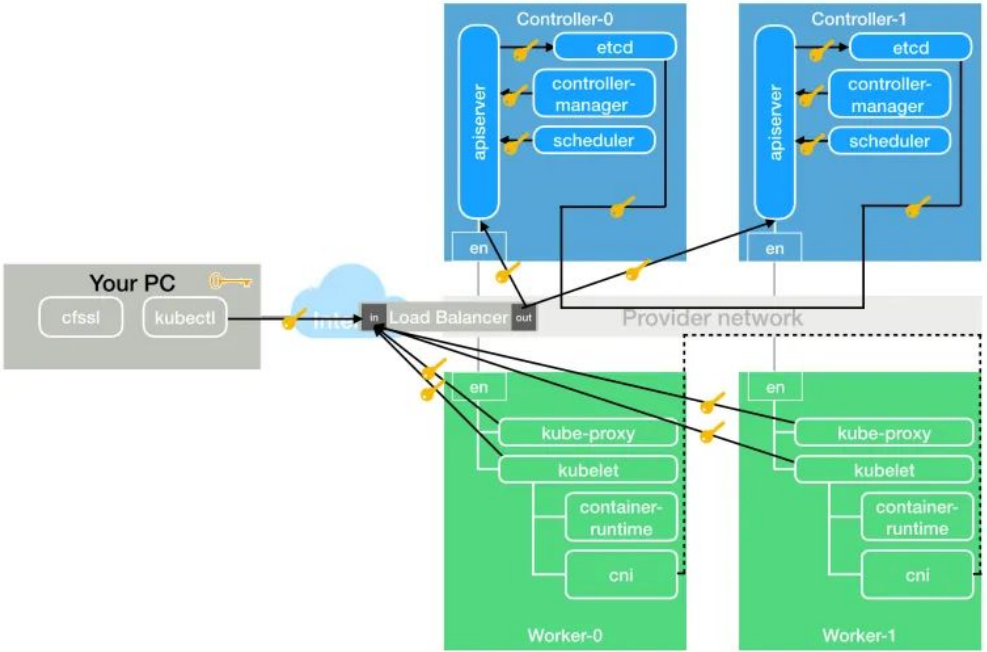
8- Configuring kubectl for Remote Access

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/10-configuring-kubectl.md



9- Provisioning Pod Network Routes

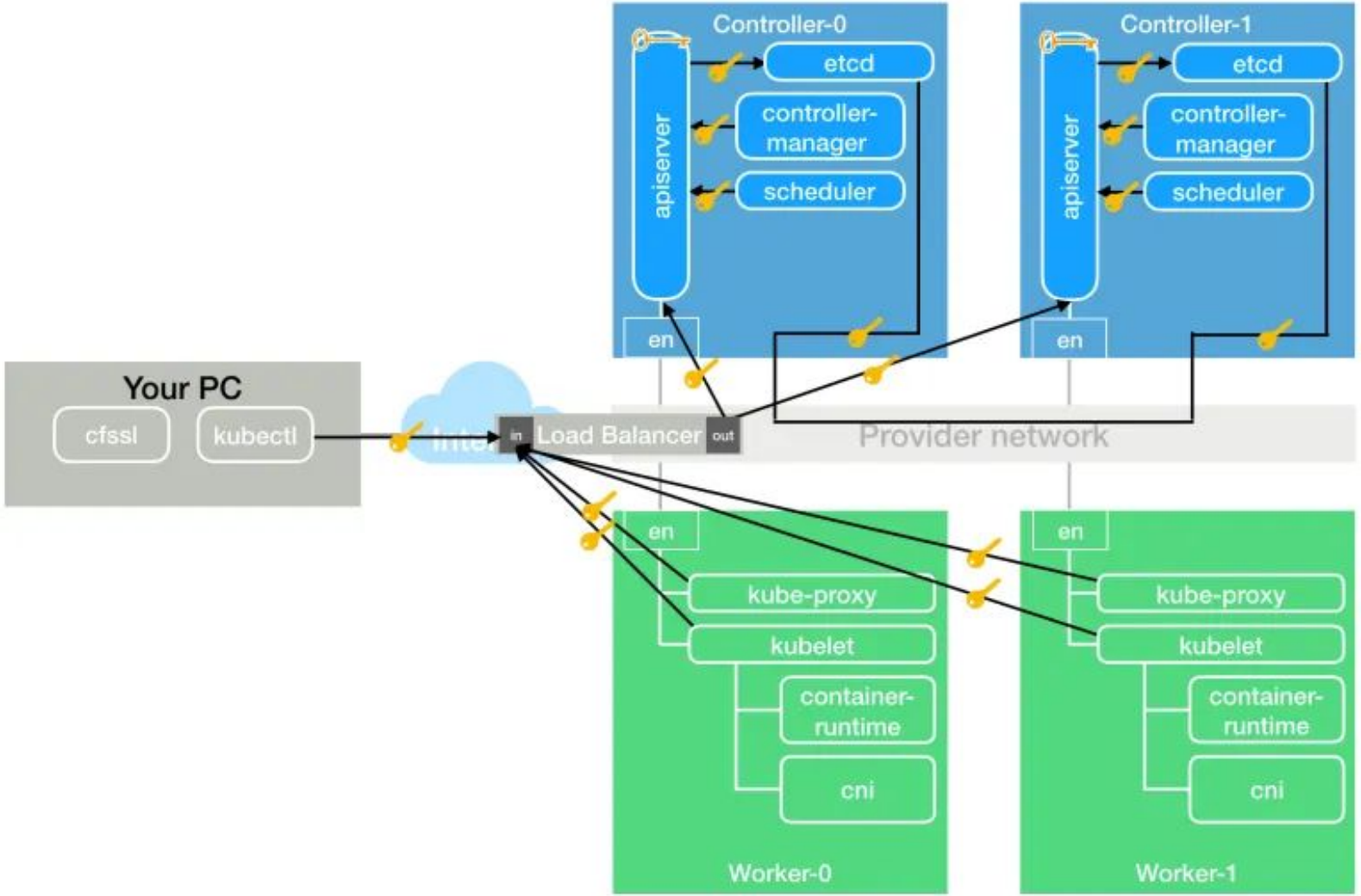
https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/11-pod-network-routes.md



10- Deploying the DNS Cluster Add-on

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/12-dns-addon.md

1


Controller-0
apiserver
etcd
controller-manager
scheduler
en
Controller-1
apiserver
etcd
controller-manager
scheduler
en
Your PC
cfssl
kubectl
in
Load Balancer
out
Provider network
en
kube-proxy
kubelet
container-runtime
cni
Worker-0
en
kube-proxy
kubelet
container-runtime
cni
Worker-1

12- Cleaning Up

https://github.com/TunisJAM/kubernetes-the-hard-way-aws/blob/master/docs/14-cleanup.md