Welcome!

Intro and Agenda

In the previous episodes we looked at Kubernetes POD networking in great details:

- Understanding Kubernetes Networking. Part 1: Container Networking
- Understanding Kubernetes Networking. Part 2: POD Network, CNI, and Flannel CNI Plug-in
- Understanding Kubernetes Networking Part 3: Calico Kubernetes CNI Provider in depth

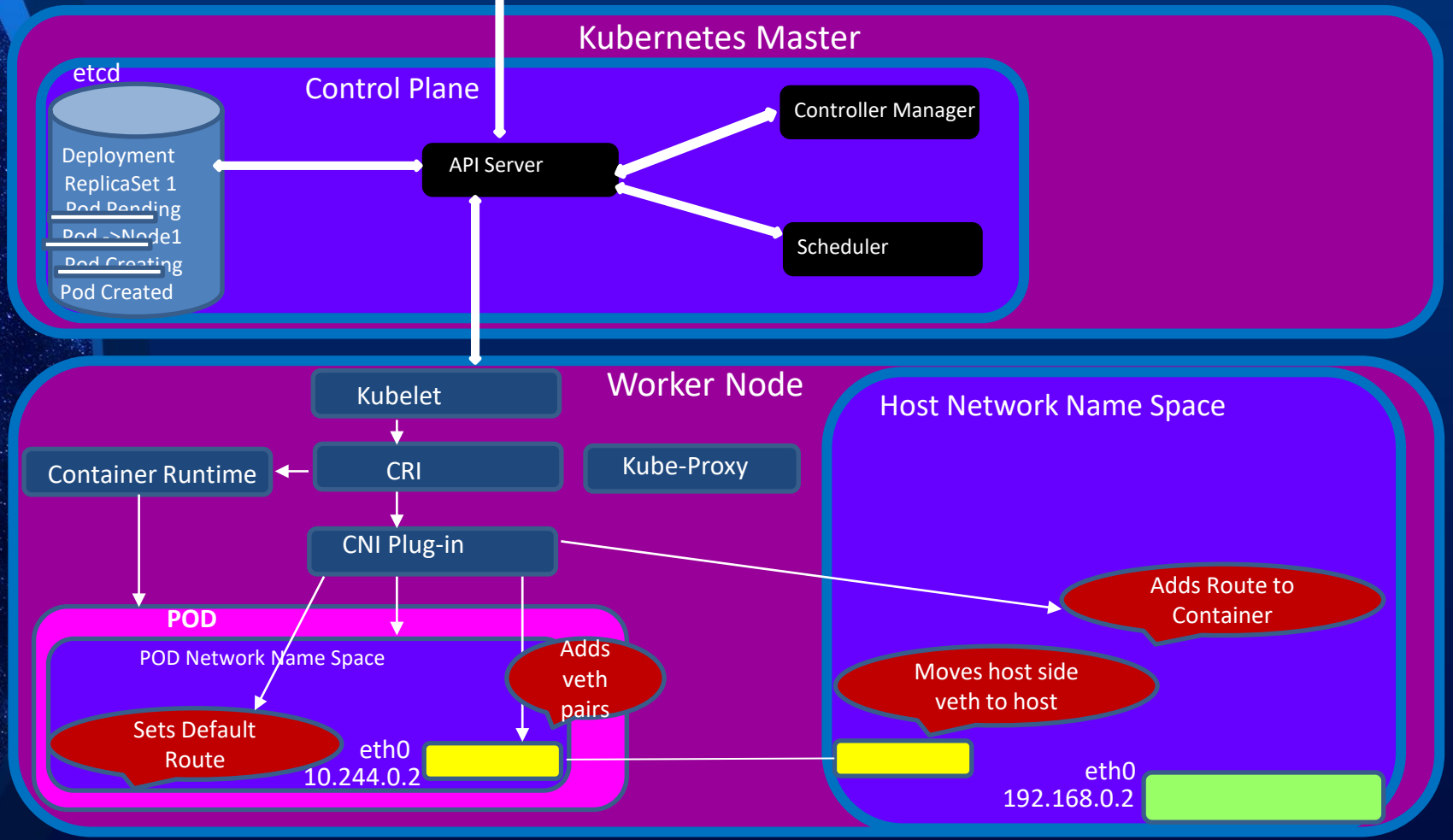
In this episode we will look at Kubernetes services. Specifically we will cover:

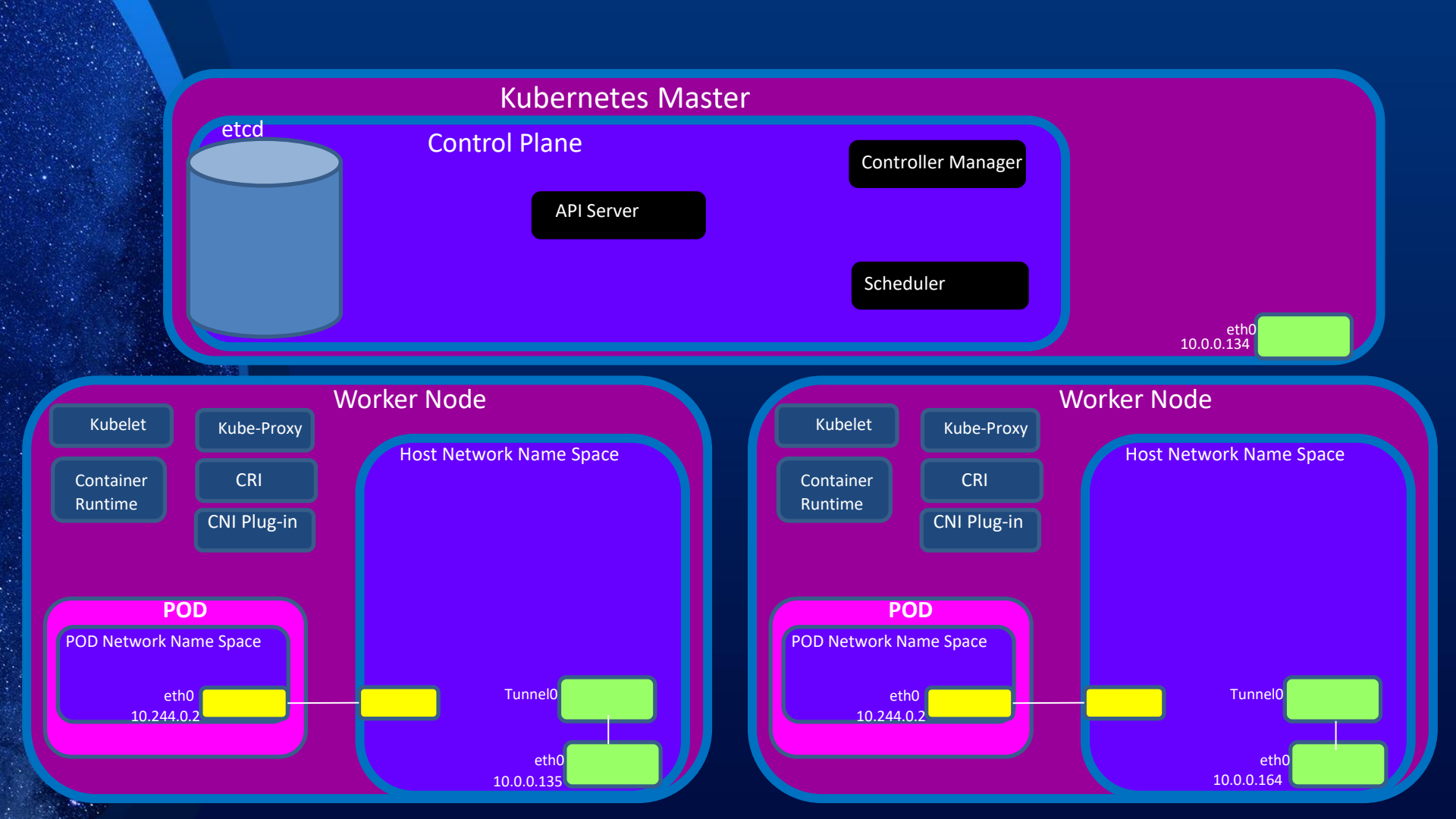
- Kubernetes control plane and walk through how PODs are created
- Learn what a Kubernetes service is
- Learn about Kube-proxy and Core-DNS
- Service types: ClusterIP, NodePort, and LoadBalancer, “Headless”
- Service discovery

Container Network Interface (CNI)

- CNI (Container Network Interface), a Cloud Native Computing Foundation project, consists of a specification and libraries for writing plugins to configure network interfaces in Linux containers, along with a number of supported plugins. CNI concerns itself only with network connectivity of containers and removing allocated resources when the container is deleted. A The specification is vendor-neutral.
- Used by Kubernetes, CloudFoundry, podman, and CRI-O.

```
gary@ubuntu-server2:~$ kubectl create deployment hello-world --image=gcr.io/google-samples/hello-app:1.0
deployment.apps/hello-world created
```



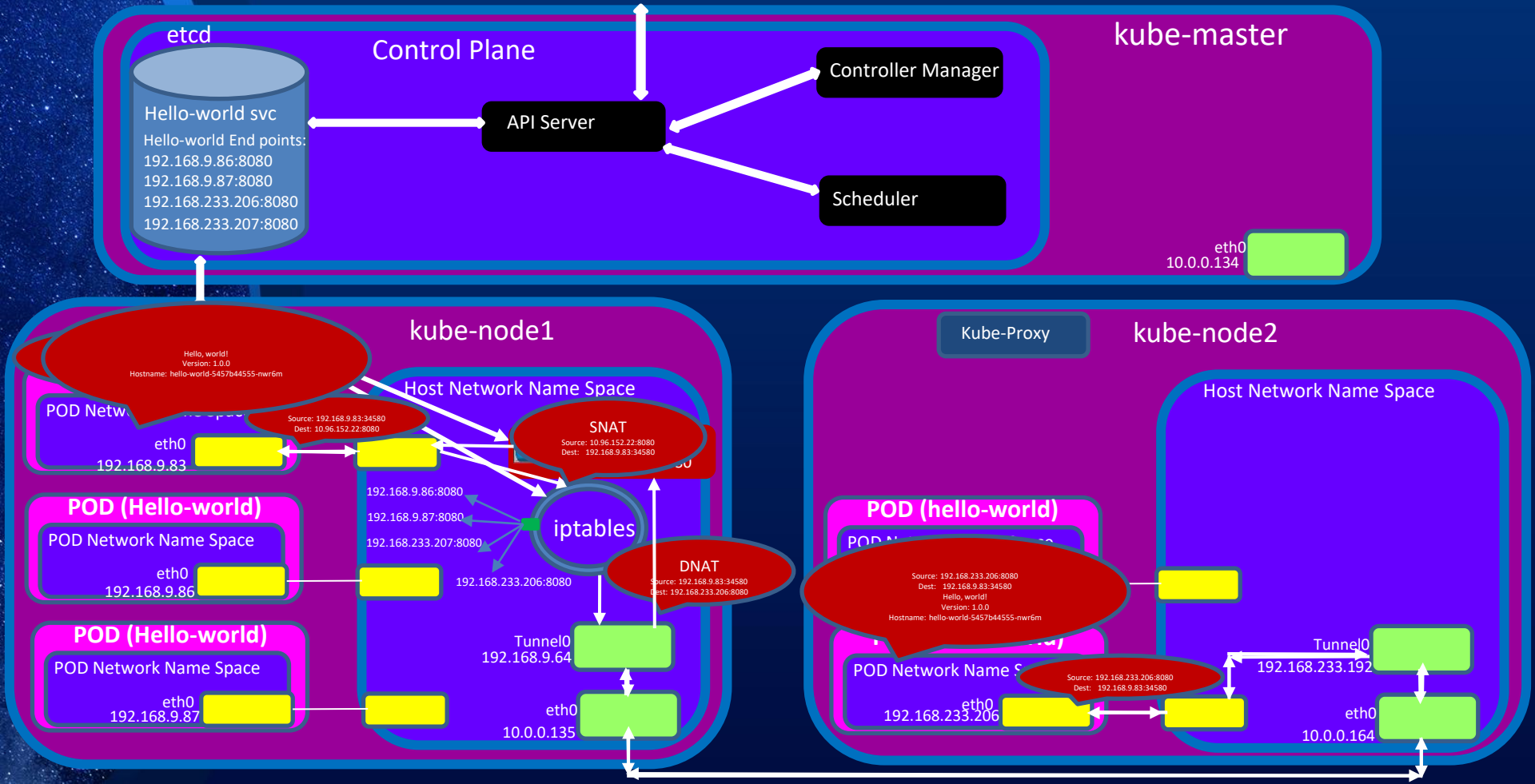

Kubernetes Master
etcd
Control Plane
Controller Manager
API Server
Scheduler
eth0
10.0.0.134
Worker Node
Kubelet
Kube-Proxy
Container Runtime
CRI
CNI Plug-in
Host Network Name Space
POD
POD Network Name Space
eth0
10.244.0.2
Tunnel0
eth0
10.0.0.135
Worker Node
Kubelet
Kube-Proxy
Container Runtime
CRI
CNI Plug-in
Host Network Name Space
POD
POD Network Name Space
eth0
10.244.0.2
Tunnel0
eth0
10.0.0.164

Kubernetes Services

- An abstract way to expose an application running on a set of Pods as a network service.
  - provides persistent end point for the clients through virtual IP.
  - Adds persistency to the ephemerality of Pods.
  - Load balances the backend Pods.
  - Automatically updated during Pod controller operations.
  - There are several different types of services: ClusterIP, NodePort, LoadBalancer.
  - How Service work:
    - Services match Pods using Labels and Selectors.
    - Creates and registers Endpoints in the Service (Pod IP and Port pair) .
    - The kube-proxy on each node creates a VIP (called ClusterIP) and programs the iptables to provide L4 load balancing.

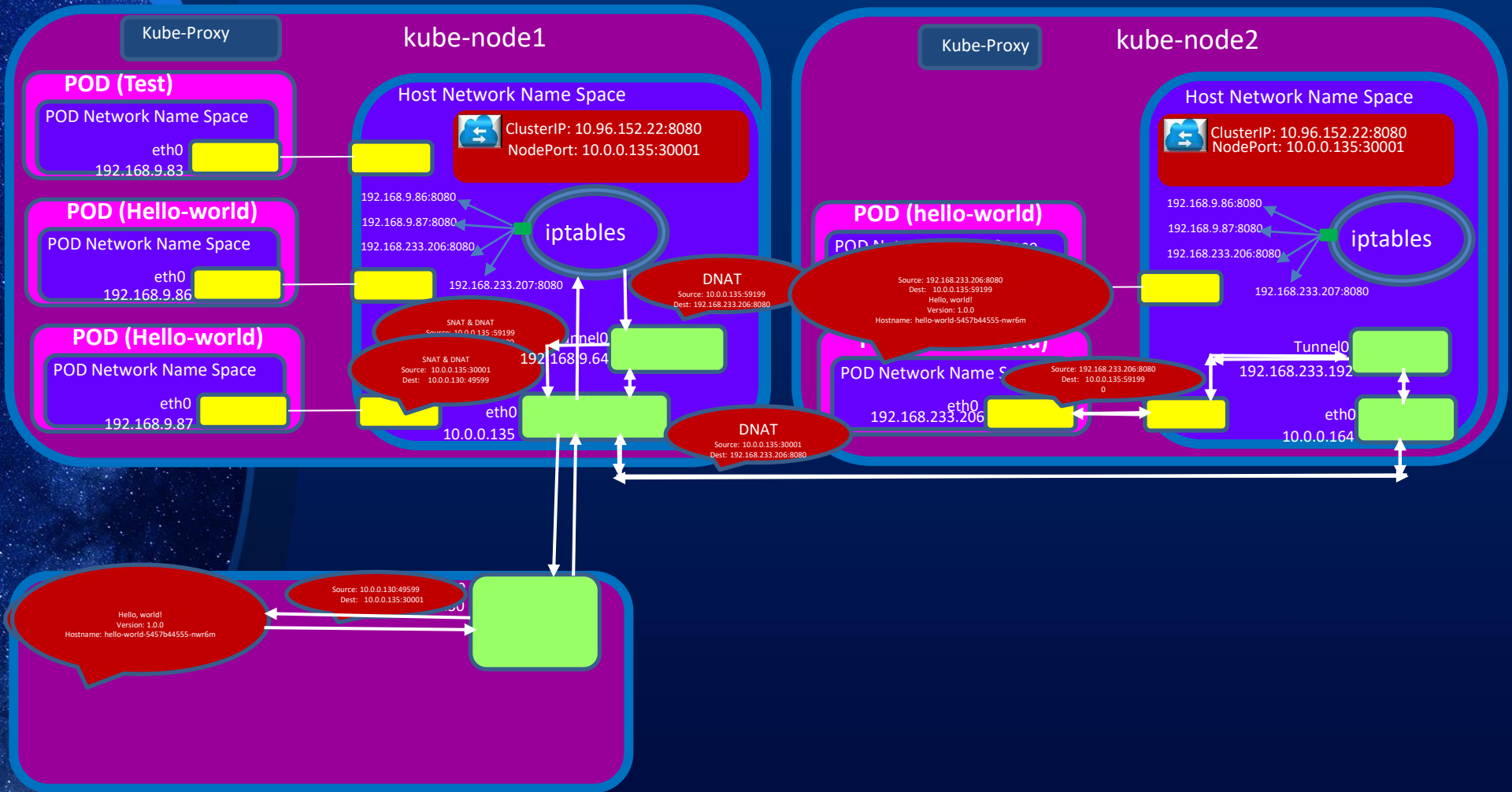
ClusterIP Service Architecture

```
gary@kube-node2:~$ kubectl expose deployment hello-world --port=8080 --target-port=8080 --type=ClusterIP
```



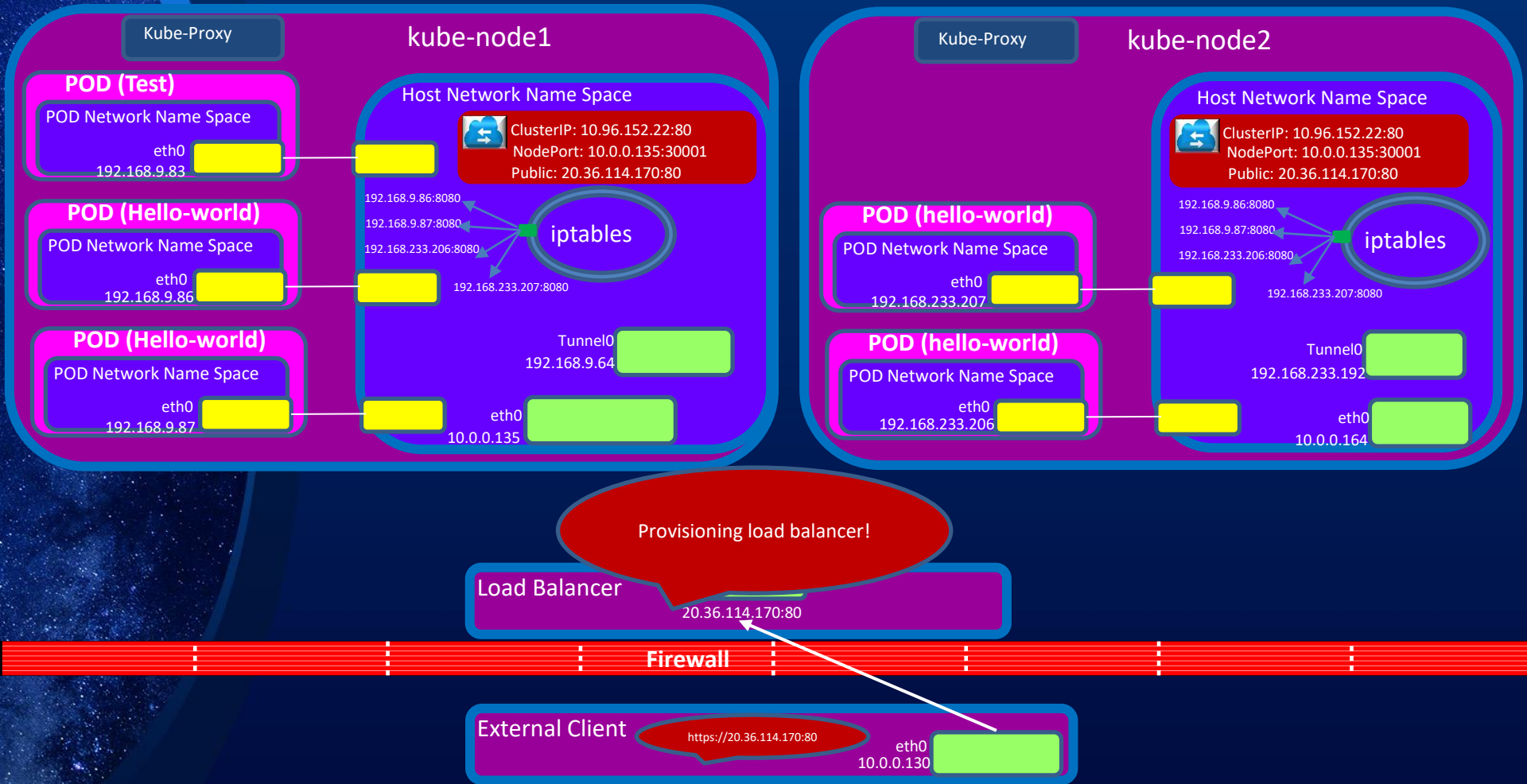
NodePort Service

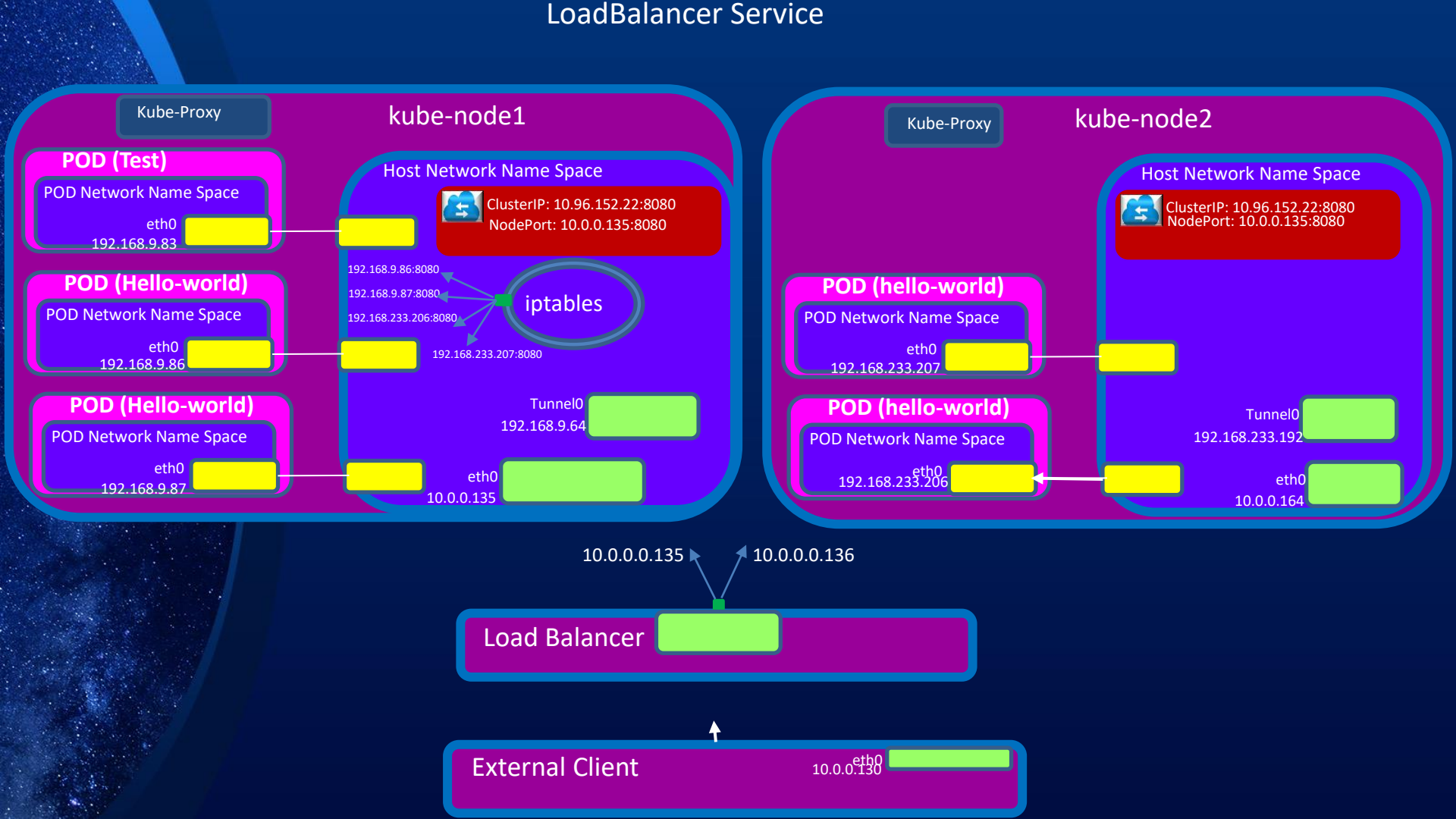
```
gary@kube-node2:~$ kubectl expose deployment hello-world --port=8080 --target-port=8080 --type=NodePort
```



LoadBalancer Service

```
gary@kube-node2:~$ kubectl expose deployment hello-world --port=80 --target-port=8080 --type=LoadBalancer
```

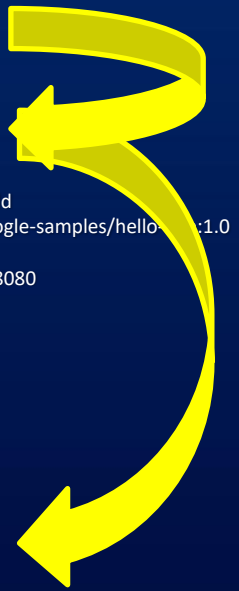



LoadBalancer Service
Kube-Proxy
kube-node1
POD (Test)
POD Network Name Space
eth0
192.168.9.83
Host Network Name Space
ClusterIP: 10.96.152.22:8080
NodePort: 10.0.0.135:8080
192.168.9.86:8080
192.168.9.87:8080
192.168.233.206:8080
192.168.233.207:8080
iptables
POD (Hello-world)
POD Network Name Space
eth0
192.168.9.86
POD (Hello-world)
POD Network Name Space
eth0
192.168.9.87
Tunnel0
192.168.9.64
eth0
10.0.0.135
Kube-Proxy
kube-node2
Host Network Name Space
ClusterIP: 10.96.152.22:8080
NodePort: 10.0.0.135:8080
POD (hello-world)
POD Network Name Space
eth0
192.168.233.207
POD (hello-world)
POD Network Name Space
eth0
192.168.233.206
Tunnel0
192.168.233.192
eth0
10.0.0.164
10.0.0.0.135
10.0.0.0.136
Load Balancer
External Client
eth0
10.0.0.130

Special Service Types

"Normal Service"

```
#Deployment  definition
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-world
spec:
  replicas: 2
  selector:
    matchLabels:
      app: hello-world
  template:
    metadata:
      labels:
        app: hello-world
    spec:
      containers:
      - name: hello-world
        image: gcr.io/google-samples/hello-[illegible]:1.0
        ports:
        - containerPort: 8080

#Service  definition
apiVersion: v1
kind: Service
metadata:
  name: hello-world
spec:
  type: ClusterIP
  selector:
    app: hello-world
  ports:
  - port: 80
    protocol: TCP
    targetPort: 8080
```



Special Service Types: Services without selectors

- Services without selectors are used abstract other kinds of backends. Examples include:
  - You want to use a database during development/testing and a prod database in production.
  - You want to point your Service to a Service in a different Namespace or on another cluster.
  - You want to gradually migrate your workload to Kubernetes

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080
```

- Because this Service has no selector, the corresponding Endpoint object is not created automatically. You can manually map the Service to the network address and port where it's running, by adding an Endpoint object manually:

```
apiVersion: v1
kind: Endpoints
metadata:
  name: my-service
subsets:
  - addresses:
      - ip: 10.0.0.135
    ports:
      - port: 30536
```

Note: This cannot be a ClusterIP!

Special Service Types: External Service

- An ExternalName Service is a special case of Service that does not have selectors and uses DNS names instead.
- Services of type “ExternalName” map a Service to a DNS name, not to a typical selector such as my-service.

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: ExternalName
  externalName: hr.db.mycompany.com
```

Special Service Types: Headless Service

- Sometimes you don't need load-balancing and a single Service IP or we need to interact with individual PODs. In this case, you can create what are termed "headless" Services, by explicitly specifying "None" for the cluster IP.:
- For headless Services, a cluster IP is not allocated, kube-proxy does not handle these Services, and there is no load balancing or proxying done by the platform for them (Note: although PODs are not load balalced through Kube-poxy(iptables), you get load balancing through DNS round robin mechanism).

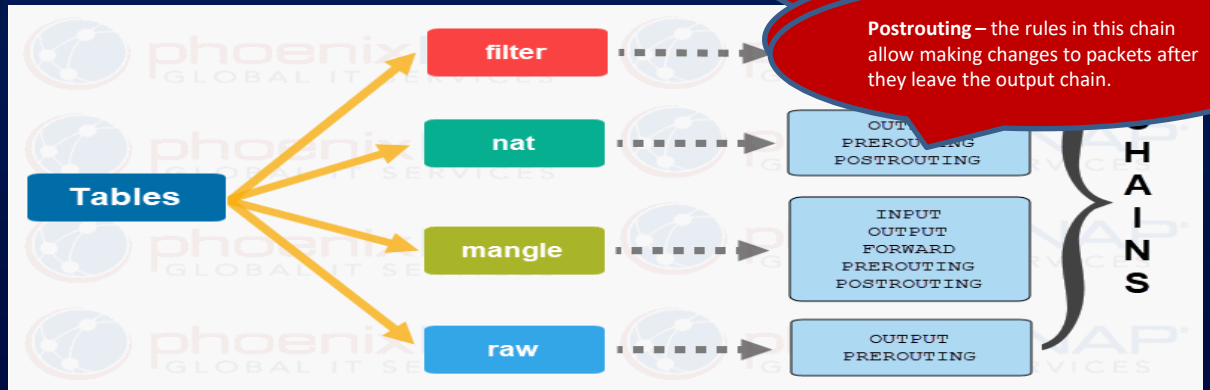
```
apiVersion: v1
kind: Service
metadata:
  name: headless-hello-world
spec:
  clusterIP: None
  selector:
    app: hello-world
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 8080

  #From inside a POD

  nslookup headless-hello-world.default.svc.cluster.local

  Name:      headless-hello-world.default.svc.cluster.local
  Address 1: 192.168.9.87 192-168-9-87.headless-hello-world.default.svc.cluster.local
  Address 2: 192.168.9.86 192-168-9-86.headless-hello-world.default.svc.cluster.local
  Address 3: 192.168.233.207 192-168-233-207.headless-hello-world.default.svc.cluster.local
  Address 4: 192.168.233.206 192-168-233-206.headless-hello-world.default.svc.cluster.local

 #From inside a POD

 nslookup hello-world

 #From inside a POD
 Name:      hello-world
 Address 1: 10.102.1.248 hello-world.default.svc.cluster.local
```

Quick intro to “iptables”

- The basic firewall software most commonly used in Linux is called iptables. The iptables firewall works by interacting with the packet filtering hooks in the Linux kernel’s networking stack. These kernel hooks are known as the netfilter framework.
- The iptables firewall uses **tables** to organize its rules. These tables classify rules according to the type of decisions they are used to make. For instance, if a rule deals with network address translation, it will be put into the nat table. If the rule is used to decide whether to allow the packet to continue to its destination, it will be added to the filter table.
- Within each iptables table, rules are further organized within separate “**chains**”. While tables are defined by the general aim of the rules they hold, the built-in chains represent the netfilter hooks which trigger them. ***Chains basically determine when rules will be evaluated***.
- A rule is a statement that tells the system what to do with a packet. Rules can block one type of packet, or forward another type of packet. The outcome, where a packet is sent, is called a **target**. A target is a decision of what to do with a packet. Typically, this is to accept it, drop it, or reject it.

The Mangle Table

The iptables [illegible] value of [illegible] unrelated packets. The connection tracking [illegible] hits the network interface. The raw table is used to exempt packets from the connection tr[illegible] orking tools. This mark does not touch the actual packet, but adds the mark to the kernel [illegible]



Service Discovery

- Kubernetes clusters automatically configure an internal DNS service (“CoreDNS”) to provide a lightweight mechanism for service discovery. Built-in service discovery makes it easier for applications to find and communicate with each other on Kubernetes clusters, even when pods and services are being created, deleted, and shifted between nodes.
- The CoreDNS Deployment is exposed as a Kubernetes Service (“kube-dns”) with a static IP. It provides these services:
  - The kube-dns listens for service and endpoint events and creates DNS as services are created:
    - Services get A (IPV4)/AAAA(IPV6) (<svcname>.<ns>.svc.<clusterdomain> for example: “hello-world.default.svc.cluster.local”)
  - On each POD, sets “ /etc/resolv.conf” nameserver to the ClusterIP address of of the kube-dns .

Thank You!

Please check out my other videos:

- Understanding Kubernetes Networking. Part 1: Container Networking.
- Understanding Kubernetes Networking. Part 2: POD Network, CNI, and Flannel CNI Plug-in.
- Understanding Kubernetes Networking Part 3: Calico Kubernetes CNI Provider in depth.
- Step by Step Instructions on Setting up a Multi Node Kubernetes Cluster on CentOS 8.
- Setup a "Docker-less" Multi-node Kubernetes Cluster On Ubuntu Server.
- Setup and Configure CentOS Linux Server on A Windows 10 Hypervisor.
- Setup NAT (Network Address Translation) on Hyper-V.
- Enable Nested Virtualization on Windows to run WSL 2 (Linux) and Hyper-V on a VM.
- Setup a Multi Node MicroK8S Cluster on Windows 10.
- Detail Windows Terminal, (WSL 2), Linux, Docker, and Kubernetes Install Guide on Windows 10.