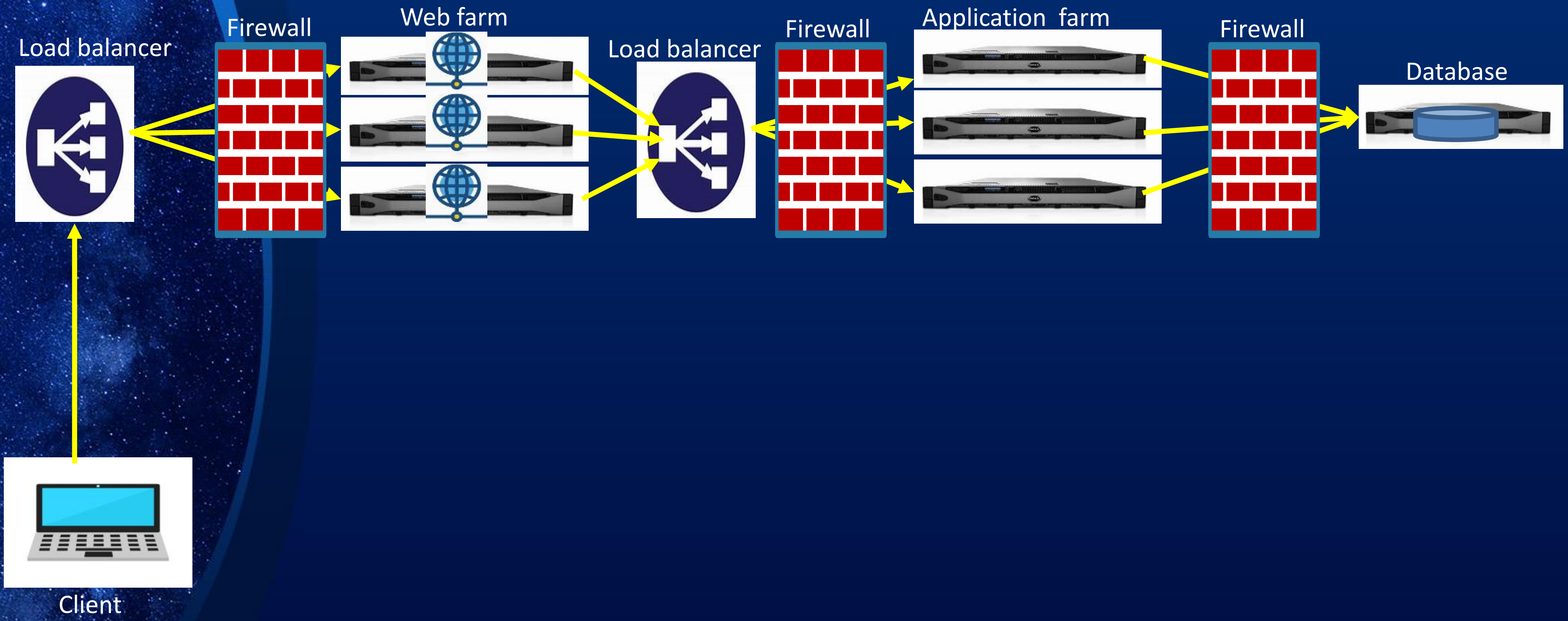
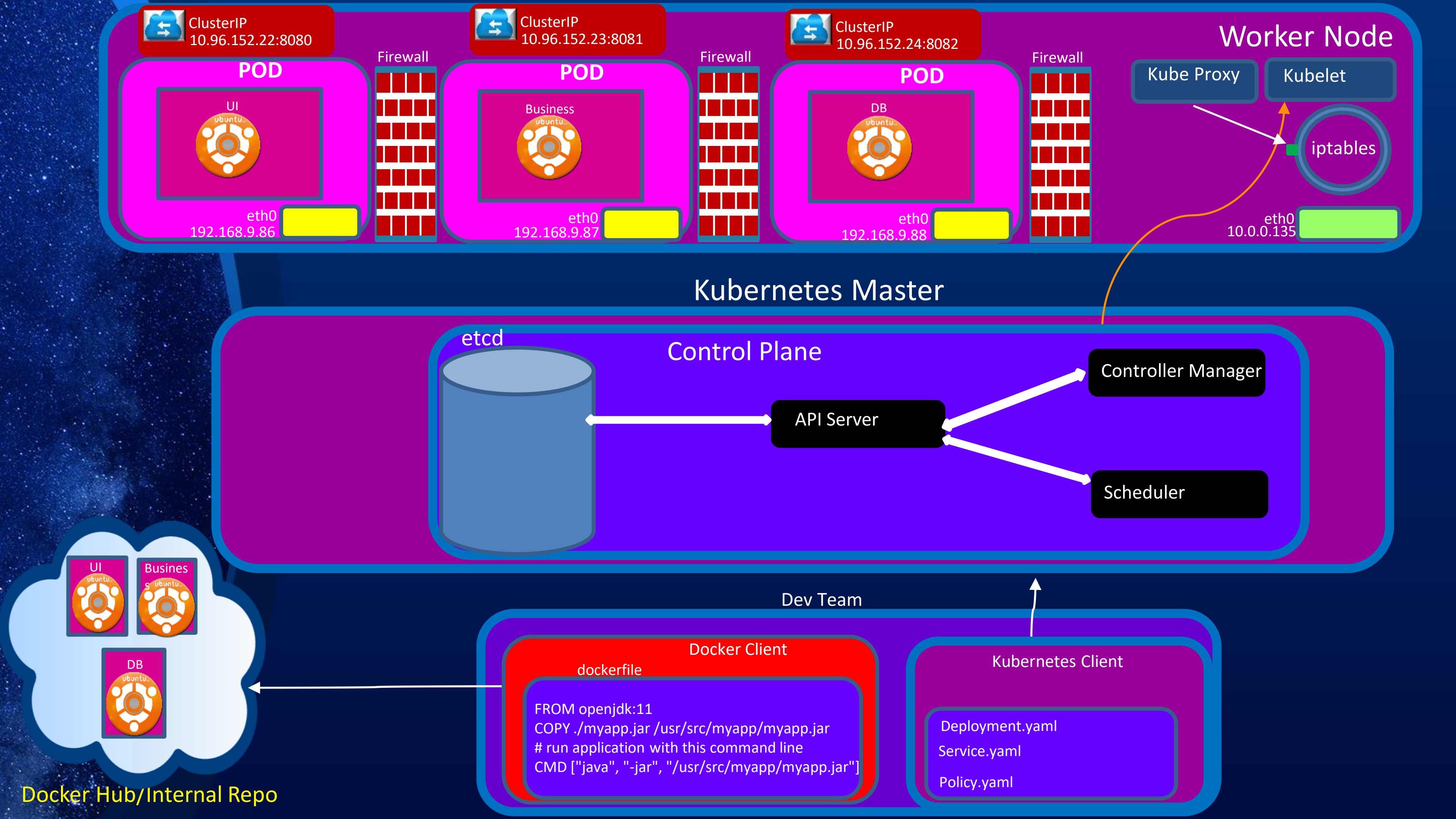
# Welcome!

Kubernetes supports Windows nodes(with some limitations). In the previous episode we set up a hybrid Kubernetes cluster using Flannel. In this tutorial,  I'll walk through the process of setting up a regular Kubernetes cluster with Calico as the CNI, and then will join Windows nodes to the cluster. This will enable us to schedule Windows containers  containers on the cluster.

# Kubernetes Network Policies

Kubernetes Network Policies = Network Security.





```
FROM openjdk:11
COPY ./myapp.jar /usr/src/myapp/myapp.jar
# run application with this command line
CMD ["java", "-jar", "/usr/src/myapp/myapp.jar"]
```

## What is network policy?

- Historically in enterprise networks, network security was provided by designing a physical topology of network devices (switches, routers, firewalls) and their associated configuration. The physical topology defined the security boundaries of the network. In the first phase of virtualization, the same network and network device constructs were virtualized in the cloud, and the same techniques for creating specific network topologies of (virtual) network devices were used to provide network security. Adding new applications or services often required additional network design to update the network topology and network device configuration to provide the desired security.

- In contrast, the Kubernetes network model defines a “flat” network in which every pod can communicate with all other pods in the cluster using pod IP addresses. This approach massively simplifies network design and allows new workloads to be scheduled dynamically anywhere in the cluster with no dependencies on the network design.

  In this model, rather than network security being defined by network topology boundaries, it is defined using network policies that are independent of the network topology. Network policies are further abstracted from the network by using label selectors as their primary mechanism for defining which workloads can talk to which workloads, rather than IP addresses or IP address ranges.

## Why is network policy important?

- In an age where attackers are becoming more and more sophisticated, network security as a line of defense is more important than ever.

- While you can (and should) use firewalls to restrict traffic at the perimeters of your network (commonly referred to as north-south traffic), their ability to police Kubernetes traffic is often limited to a granularity of the cluster as a whole, rather than to specific groups of pods, due to the dynamic nature of pod scheduling and pod IP addresses. In addition, the goal of most attackers once they gain a small foothold inside the perimeter is to move laterally (commonly referred to as east-west) to gain access to higher value targets, which perimeter based firewalls can't police against.

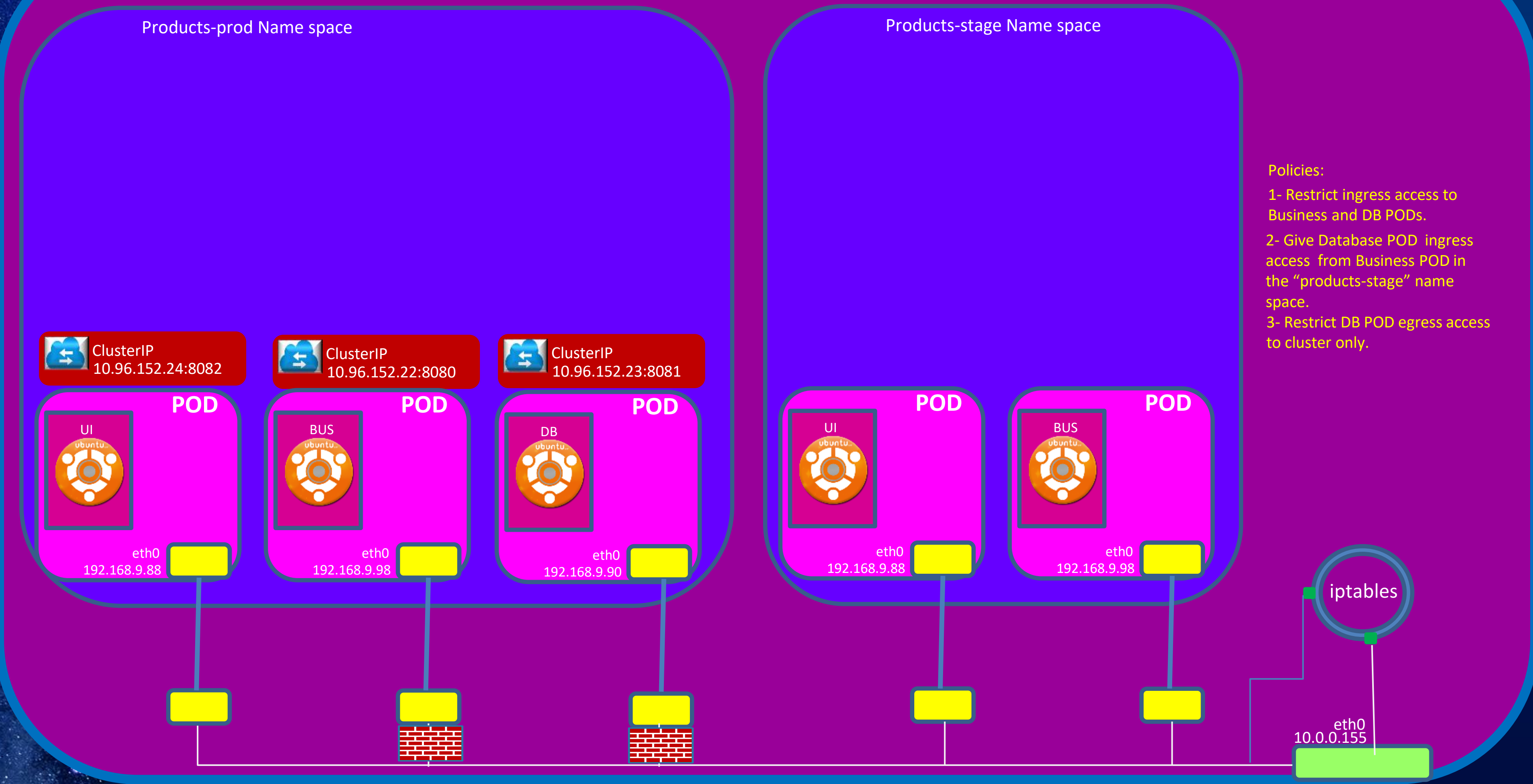
- Network policy on the other hand is designed for the dynamic nature of Kubernetes by following the standard Kubernetes paradigm of using label selectors to define groups of pods, rather than IP addresses. And because network policy is enforced within the cluster itself it can police both north-south and east-west traffic.

- Network policy represents an important evolution of network security, not just because it handles the dynamic nature of modern microservices, but because it empowers dev and devops engineers to easily define network security themselves, rather than needing to learn low-level networking details or raise tickets with a separate team responsible for managing firewalls. Network policy makes it easy to define intent, such as "only this microservice gets to connect to the database", write that intent as code (typically in YAML files), and integrate authoring of network policies into git workflows and CI/CD processes.

## Kubernetes network policy

- The main features of Kubernetes policies are:
  - Policies are namespace scoped (i.e. you create them within the context of a specific namespace just like, for example, pods)
  - Policies are applied to pods using label selectors
  - Policy rules can specify the traffic that is allowed to/from other pods, namespaces, or CIDRs
  - Policy rules can specify protocols (TCP, UDP, SCTP), named ports or port numbers
- *Note: Kubernetes itself does not enforce network policies, and instead delegates their enforcement to network plugins. Most network plugins implement the mainline elements of Kubernetes network policies, though not all implement every feature of the specification.

# Worker Node

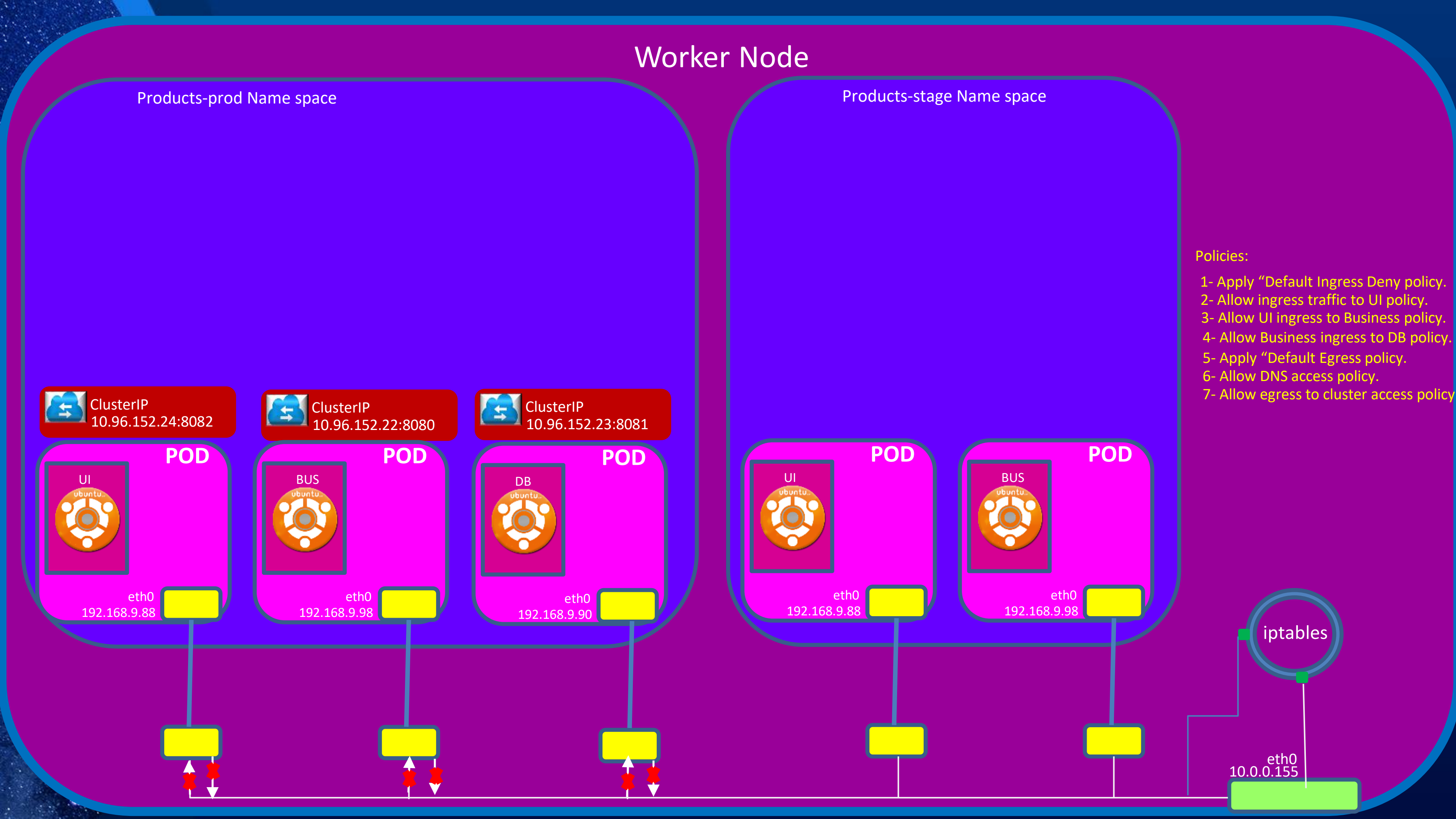


Policies:

1- Restrict ingress access to Business and DB PODs.

2- Give Database POD ingress access from Business POD in the “products-stage” name space.

3- Restrict DB POD egress access to cluster only.



Policies:

1- Apply “Default Ingress Deny policy.
2- Allow ingress traffic to UI policy.
3- Allow UI ingress to Business policy.
4- Allow Business ingress to DB policy.
5- Apply “Default Egress policy.
6- Allow DNS access policy.
7- Allow egress to cluster access policy

## Thank You!