EQUIFAX®

Apigee Micro Gateways on Kubernetes

Build once , run anywhere

apigee

SID Global Solutions
Digital Transformation Experts

Google Partner

Updated Date: 2/15/2018

Agenda

Day 1

Introduction

- ✓About the Pilot
- ✓Outcome

Kubernetes

- ✓Architecture
- ✓Basic Commands
- ✓PODS
- ✓Deployment
- ✓Services
- ✓Primitives
- ✓Networking
- ✓Ingress
- ✓Name Spaces
- •Configuration
- ✓Service Discovery

Hands On

- ✓Kubernetes

Day 2

Edge Micro Gateway

- •Architecture
- •Deployment Options
- •Gateway as an Ingress
- •Security
- •Analytics

CICD

- •Jenkins Jobs
  - •Build
  - •Multi Cluster Deploy

Pilot Knowledge Transfer

- •Micro Gateway Container
- •Micro Service
- •KOPS on AWS
- •Kubernetes on Azure

Hands On

- •Edge Micro Gateway

Pilot Details

Scope :

- Create 3 node Kubernetes cluster on AWS. Cluster should have 1 master and 2 worker nodes
- Create 3 node Kubernetes cluster on Azure. Cluster should have 1 master and 2 worker nodes
- Create and Configure ingress controllers on both AWS and Azure
- Create a total of 4 different work spaces
- Build and configure Apigee micro gateway (AMG) container capable of being initialized with environment specific information
- Create and deploy service composed of AMG and helloworld microservices as shown in the architecture to the right in parallel on AWS and Azure
- Demonstrate that the architecture is scalable, portable and cloud agnostic

Deliverables:

1. Kubernetes (k8s) Cluster on AWS with 1 master and 2 worker nodes with ingress controllers
2. Kubernetes Cluster on Azure with 1 master and 2 worker nodes with ingress controllers
3. Jenkins Server with Two (2) Jenkins Jobs to deploy Service to AWS and Azure k8s
4. Git repo to build a AMG image capable of taking environment information as a runtime variable
5. Demo showing the following
    1. Solution is cloud agnostic
    2. Solution is scalable
6. Documentation of Steps to repeat items #1, #2 and #3

Outcome

- Deploy a Micro Gateway on AWS and AZURE using common manifest
- Deploy Micro Service on AWS and AZURE using common manifest
- Observe Scaling and Load balancing on both clusters
- Create 1 proxy for both Micro gateways with a cloud and cluster agnostic target
- Observe Cloud agnostic Routing
- Provide Knowledge Transfer

Day 1

SID Global Solutions
Digital Transformation Experts

apigee
Google Partner

Part 1

- Introduction
- What is it Really?
- Installation

Introduction

- Kubernetes is an open-source software for automating deployment, scaling, and management of containerized workloads.
- Builds on 15 years of experience at Google.

Alternatives (not an exhaustive list)

- Docker Swarm
- Cloud Foundry
- Apache Mesos
- Rancher
- Hashicorp Nomad

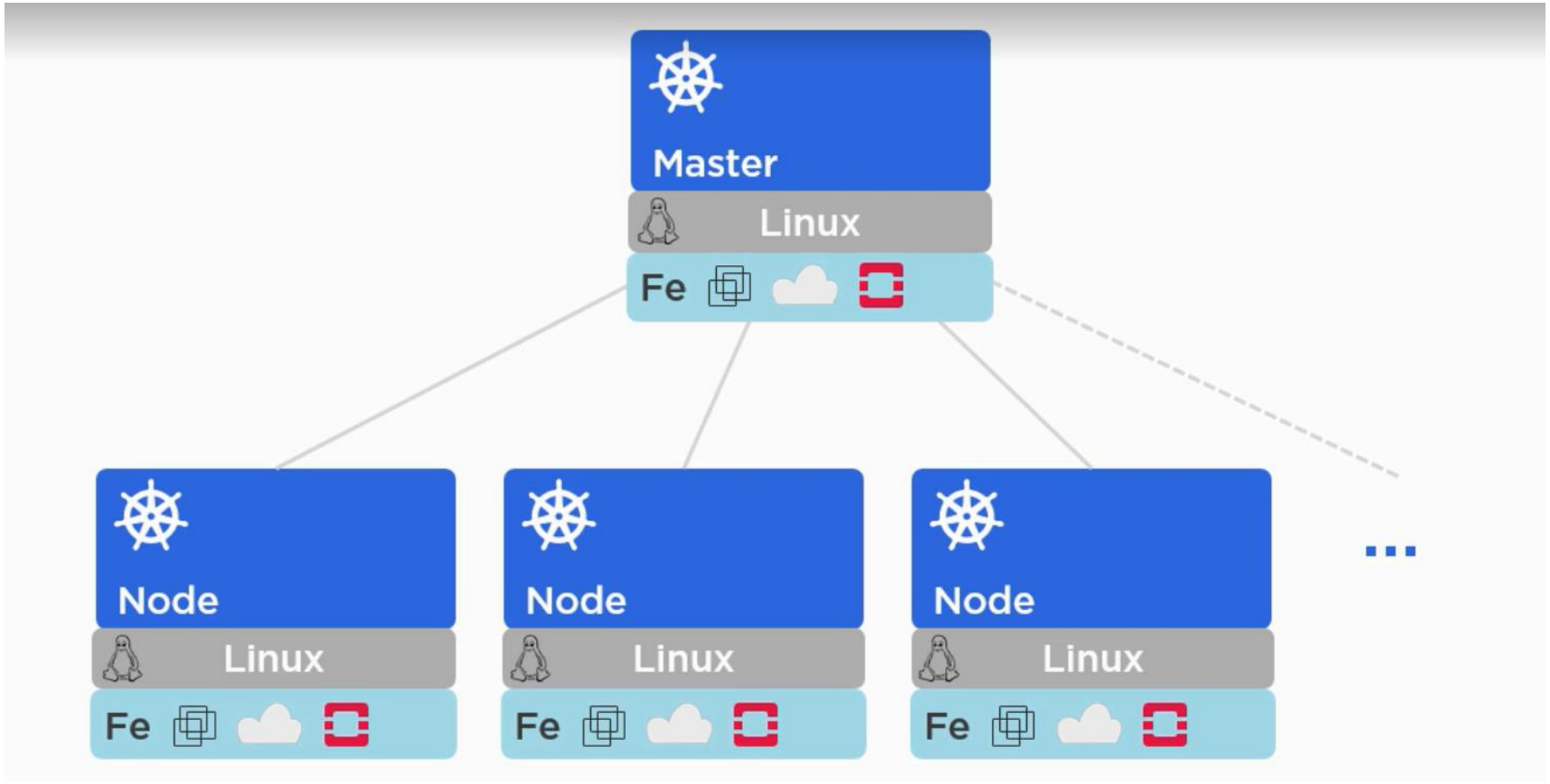
Why are we talking about it today?

Because its your target environment where a majority of your apps and integrations will run and it runs everywhere…

- Amazon Cloud
- Azure Cloud
- Runs on VMWare

Finally fulfills the promise of "Code Once, Run Anywhere.."

Why Kubernetes, what is Kubernetes?

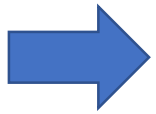


What :
Cluster Manager
Container Scheduler

Why ?
Geared towards scale
Lots of HA features
Deployed on Bare Metal or VM's

Installation Options

- Minikube (Easiest to set up)
- KOPS (AWS)
- ACS (Azure)
- kubeadm
- hyperkube
- Bootkube



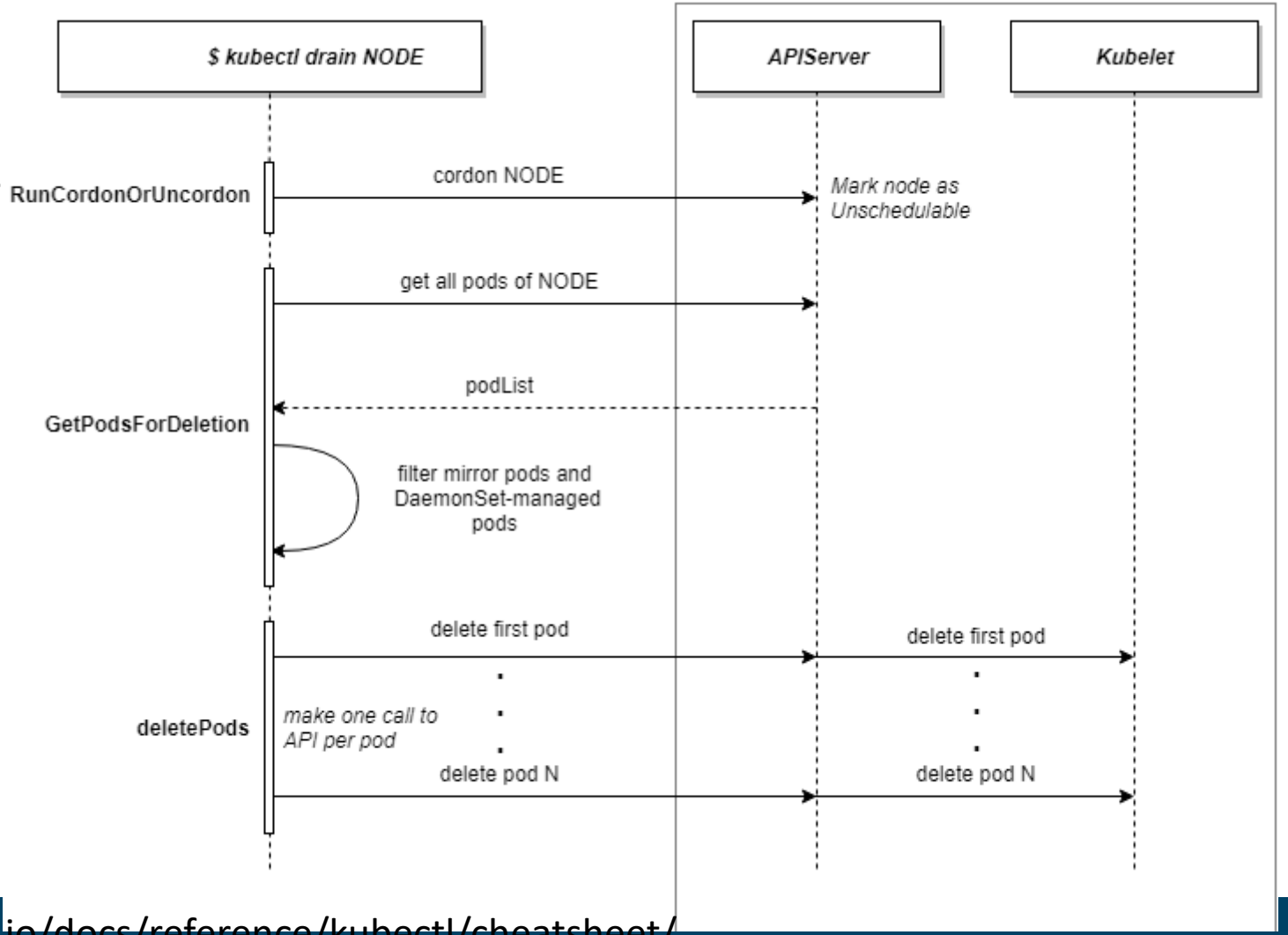
Used for Pilot

Part 2

- Kubectl
  - The life of an API
- Kubeconfig
- Kubernetes Primitives

Role of Kubectl

- Command-line tool configured to communicate with your k8S cluster
- kubectl
  - Basic commands
  - Exec [For debugging]
- How does kubectl work?
  - Ex : Drain a NODE



https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Life of an API request

- The request is received on a TLS secured API Server
- The API Server
  - Authenticates the request using one of
    - Certificates
    - Tokens
    - Basic Auth
    - Open Id
    - Web Hooks
  - Authorizes the request (RBAC based on group memberships)
  - Runs admission control policies
- The actual work is done

Role of Kubeconfig

- The file used to configure access to a K8S cluster
- Provides information about
  - Cluster Location
  - Users
  - Context - A combination of cluster + user [+ namespace]

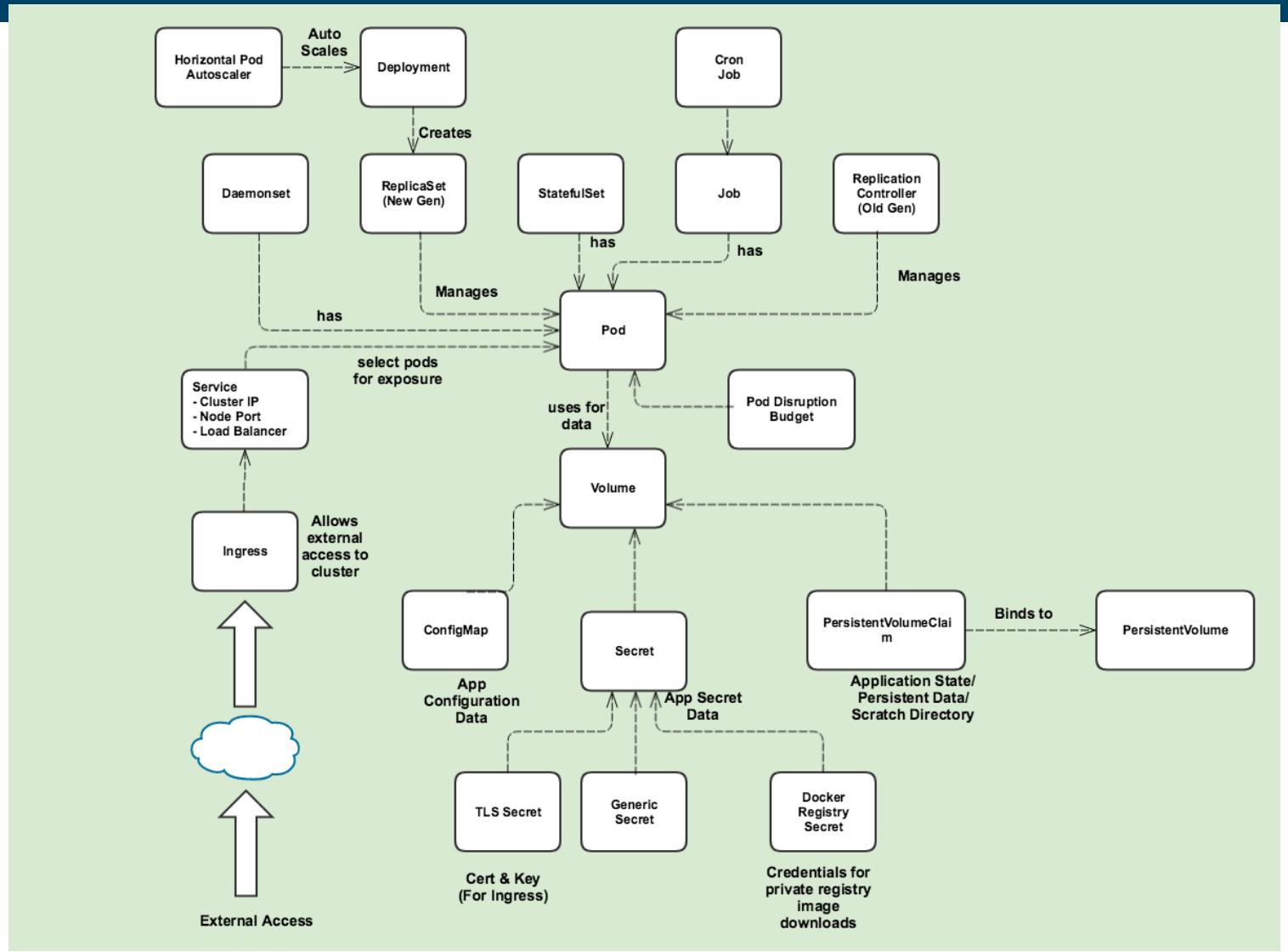
```
apiVersion: v1
kind: Config
clusters:
- cluster:
    certificate-authority: ssl/ca.pem
    server: https://172.17.4.101:443
  name: vagrant-multi-cluster
contexts:
- context:
    cluster: vagrant-multi-cluster
    namespace: default
    user: vagrant-multi-admin
  name: vagrant-multi
users:
- name: vagrant-multi-admin
  user:
    client-certificate: ssl/admin.pem
    client-key: ssl/admin-key.pem
current-context: vagrant-multi
~
```

Kubernetes Primitives – Developers Perspecitive

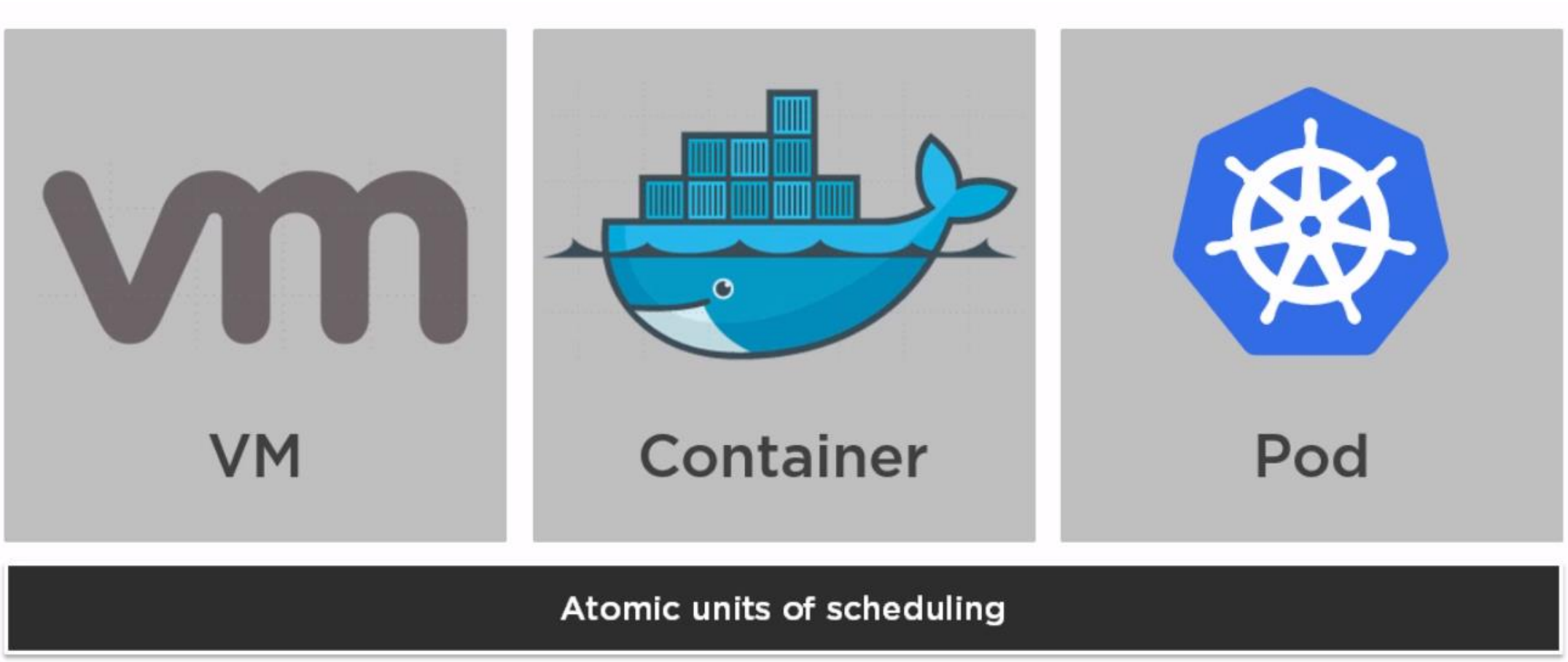
Pod Autoscaler
Deployment
Demonset
ReplicaSet
Stateful Set
Job
Cron Job
Replication Controller

Pod
Pod Disruption Budget
Volume
ConfigMap
Secret (* Secret)
Persistent Volume Claim

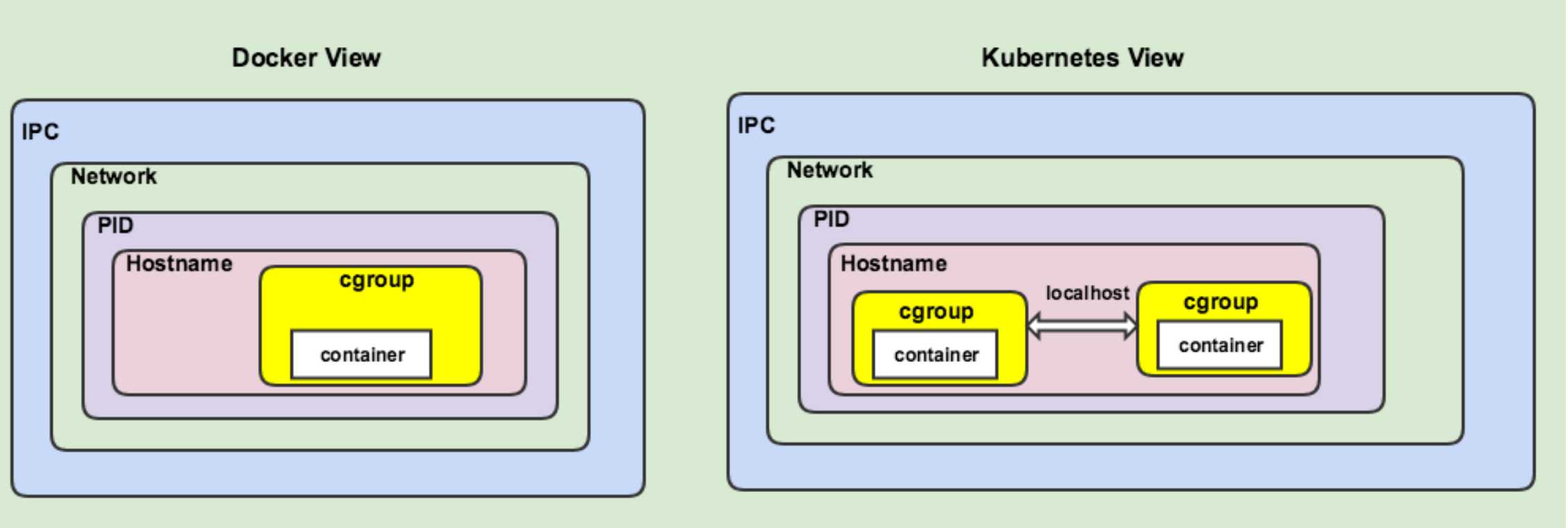
Ingress
Service
- Cluster IP
- Node Port
- Load Balancer



What's is a Pod ?

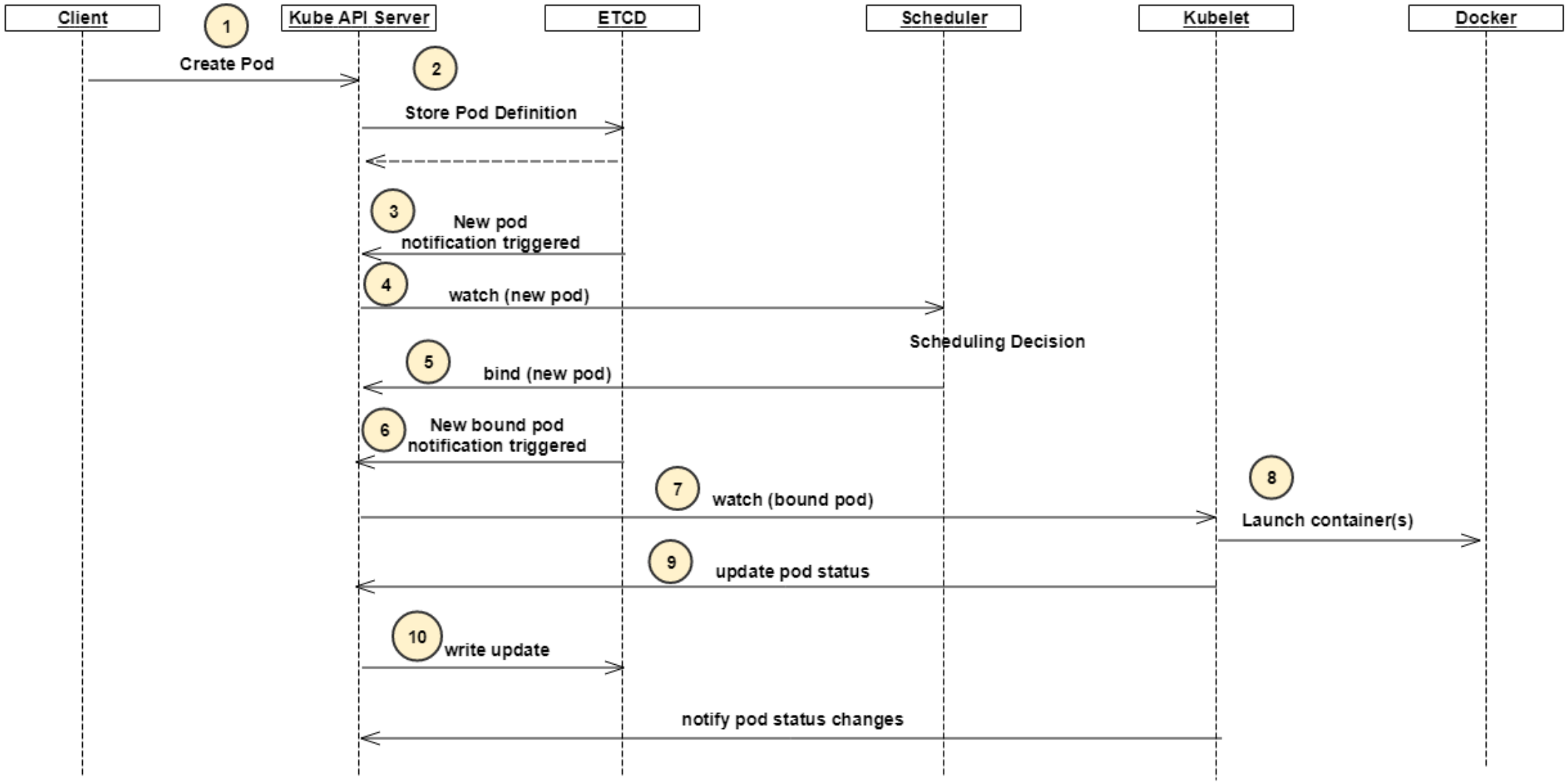


What's in a Pod ?



cgroups (abbreviated from control groups) is a Linux kernel feature that limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network, etc.) of a collection of processes

Scheduling a Pod



Kubernetes – Key takeaways (Recap)

- Actual State vs Desired State
- Based on the concept of Resources
  - Each Resource is defined as an object in a YAML file
  - A Controller inspects each resource and manages its lifecycle
  - Roles and permissions are managed around CRUD permissions on resources
- Imperative vs Declarative commands

- 15 Minute Break

Part 3

- Namespaces
- Replica Sets
- Deployments
- Labels and Services
- Accessing applications
  - Via services
  - Via Ingress

Namespaces

Namespaces are intended to isolate multiple groups / teams and give the access to a set of resources. Each namespace can have quotas

Think: Virtual Clusters

Note:

- Namespaces are flat and cannot be nested
- Namespaces DO NOT provide network isolation

Hands On:

- Creating a namespace
- Using a namespace

Replica Sets (Replication Controller)

A replica set ensures that a specified number of pod “replicas” are running at any one time.

In other words, A replica set makes sure that a pod or homogeneous set of pods are always up and available.

Hands On:

Replica Sets

Deployments

- Allow Server side update of pods at a specified rate
- Generate Replica Sets
- Allow for Deployment Patterns
  - Blue Green
  - Canary

Hands On:

- Scale up and Down
- Blue Green
- Canary
- Canary (Dedicated Access)

Labels and Services

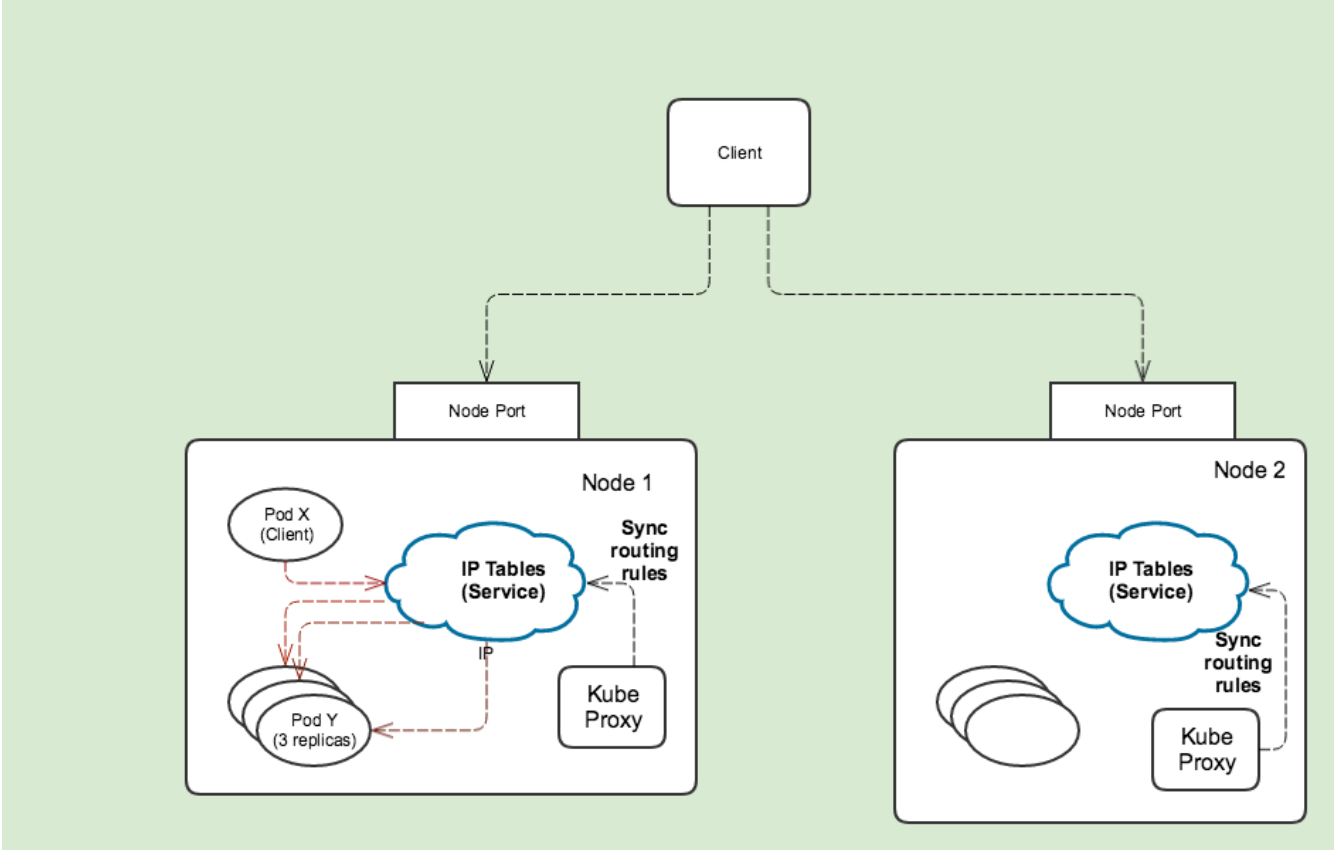
- Labels select objects
- Services expose applications

Hands On:

- Labels
- Services
- Load Balancing

Services – Accessing your applications

- Provide a stable virtual endpoint to ephemeral pods
- Not tied to any network interface
- Based purely on IP Tables (as of v 1.9)



Service Types

- **Cluster IP** – Default, Provides internal access to the endpoint (unless manually created to point to an external endpoint)
- **Node Port** – Exposes the same high range port for every host
- **Load Balancer** – Native solution to expose services on cloud providers

Scaling and Rolling updates - Deployments

- Demos
  - Scale up
  - Guarding against bad deployments
  - Rolling back deployments

Ingress

- A reverse proxy like functionality giving you access to your applications
- Collection of rules that allow inbound access to the cluster services

Think : Nginx (But less configuration and maintenance)

Anatomy of an Ingress

- Ingress Resources (Defined by Developers per access point)
- Ingress Controller (Created by Cluster/Namespace Administrator once)

Demos:

- Ingress Controller
- Ingress Resources (Vanilla)
- Ingress Resources with TLS Termination

Part 4

- Volumes
  - Secrets
  - Config Maps
  - Persistent Volume and Persistent Volume Claim

Config Maps

- Stores non sensitive configuration data
- Demo

Secrets

- Store Secret Data
  - Generic Secret
  - TLS Secret
  - Docker Registry Secret

Volumes

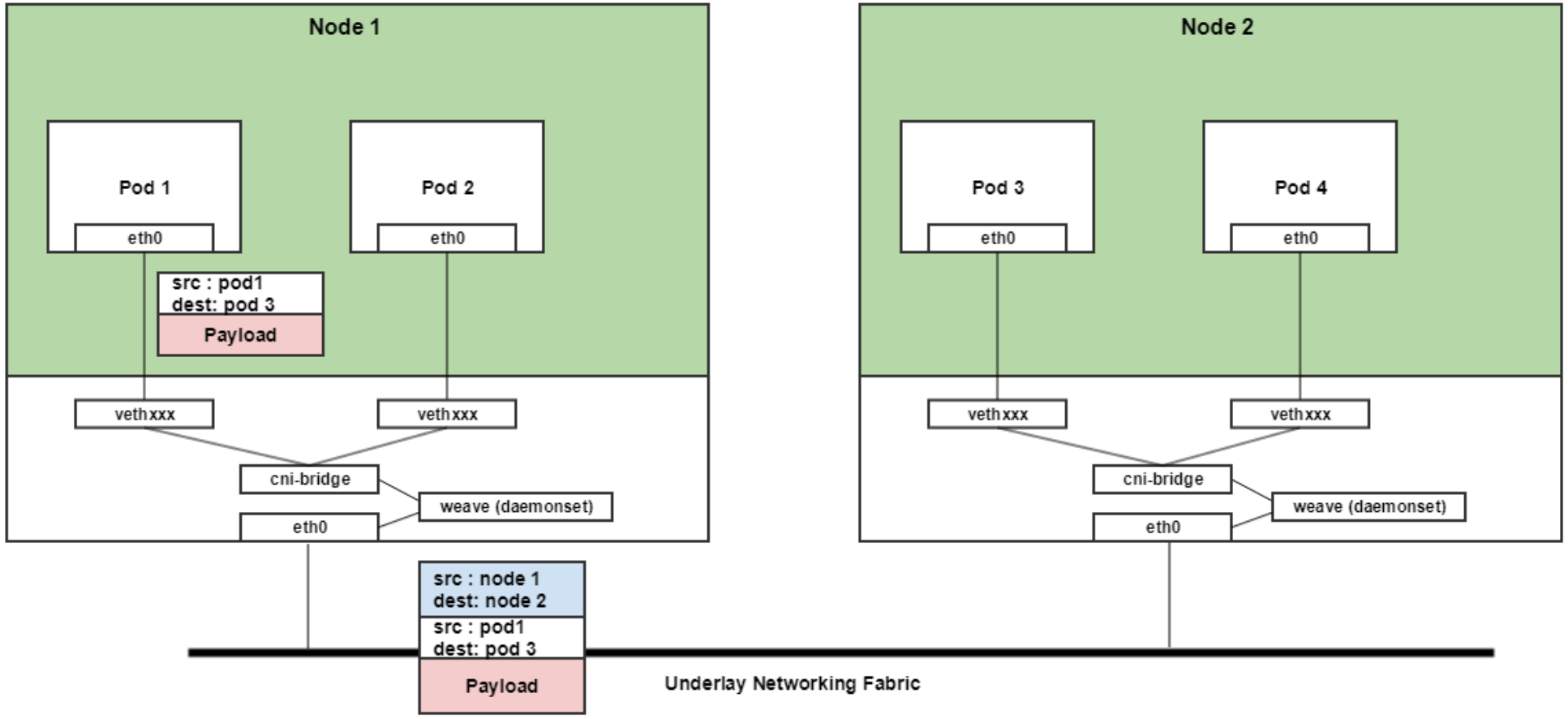
- Persistent Volumes (Storage Abstraction)
- Persistent Volume Claim (Pod's claim to a volume)

- 15 Minute Break

Part 5

- Networking Primer

Kubernetes Networking Primer



Thank you