SELA

Appendix: Kustomize

Kubernetes Workshop

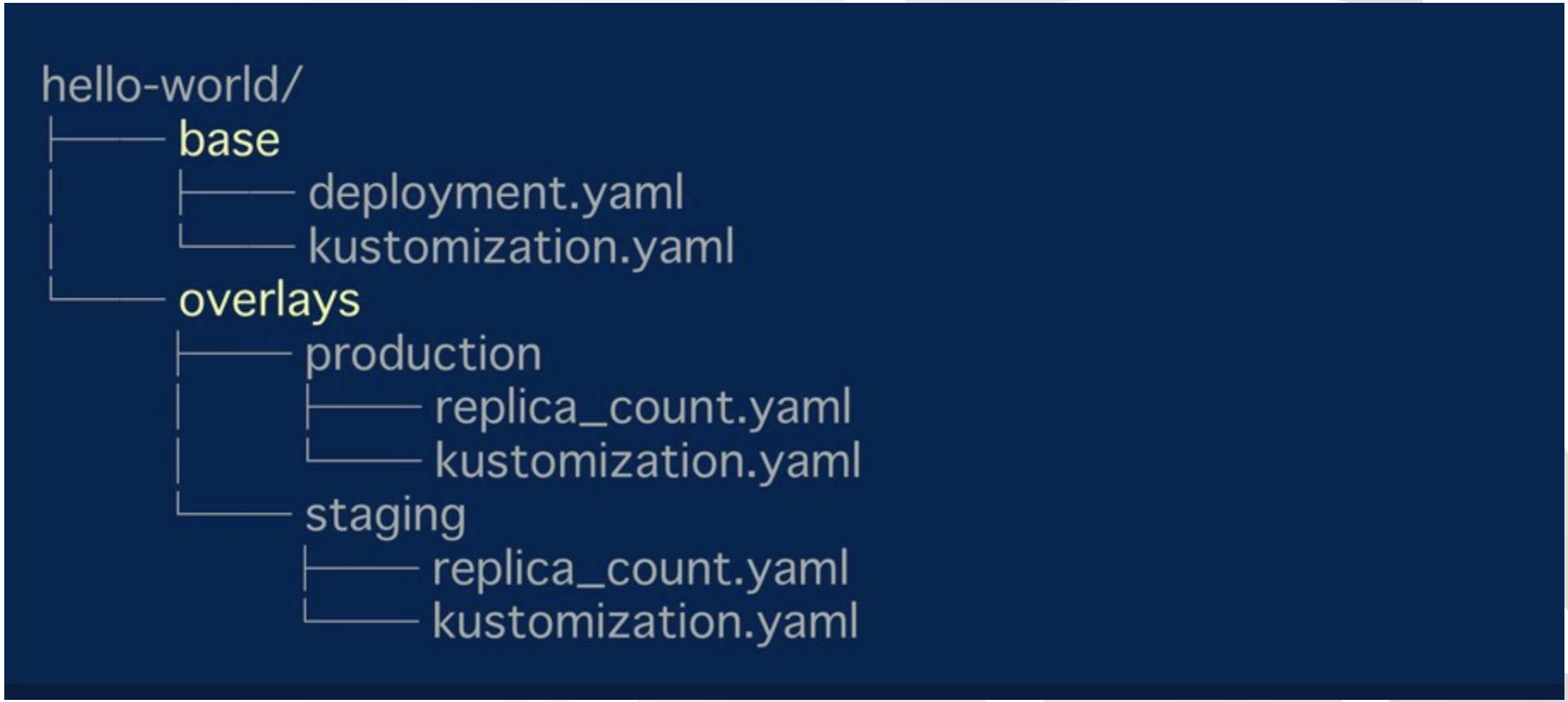
Agenda

- What is Kustomize?
- File Structure
- The Kustomization.yaml
- Overlay Customization Model
- Lab: Kustomize

What is Kustomize?

- Kustomize is a CLI for managing Kubernetes style object with declarative way

File Structure

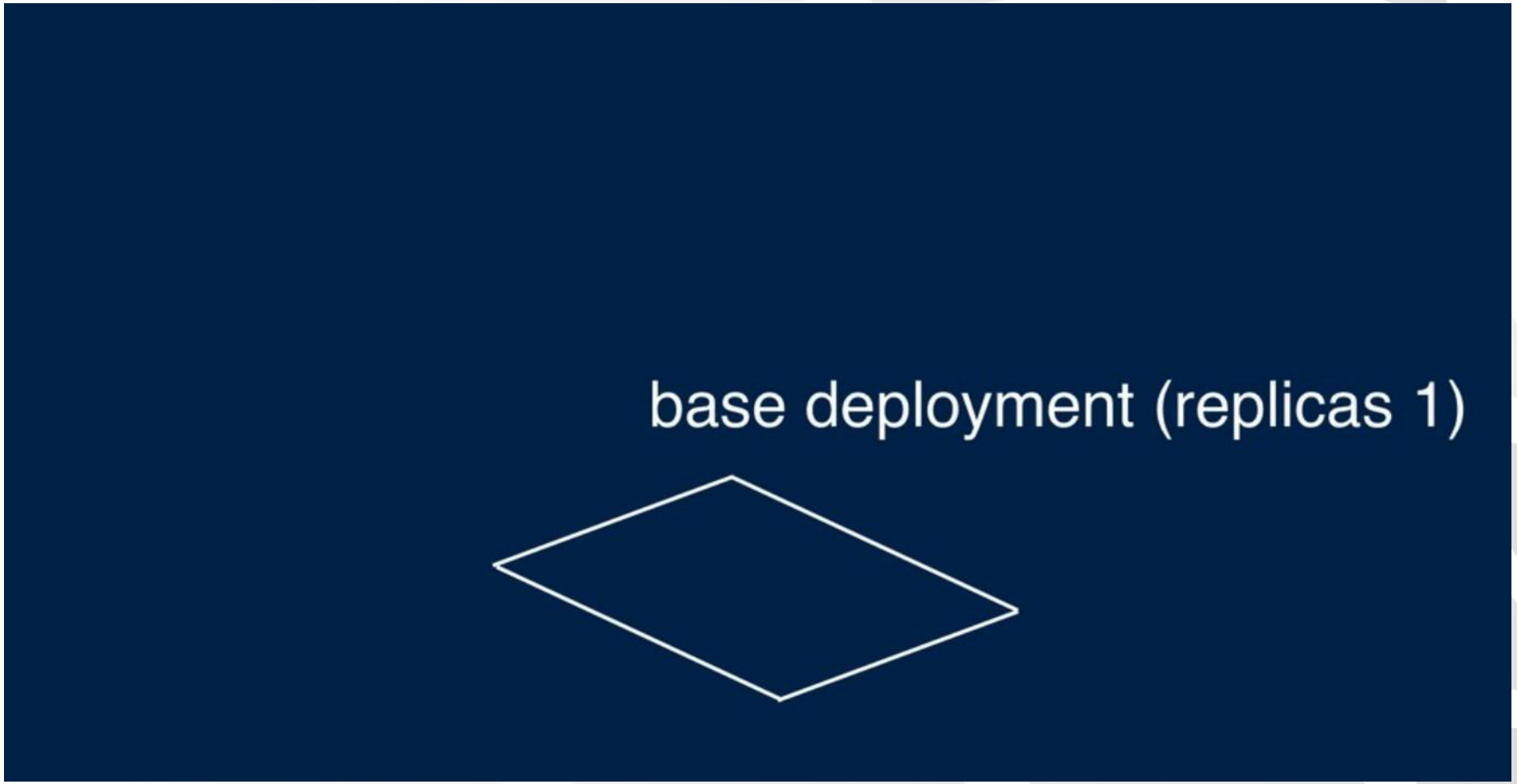


Kustomization.yaml

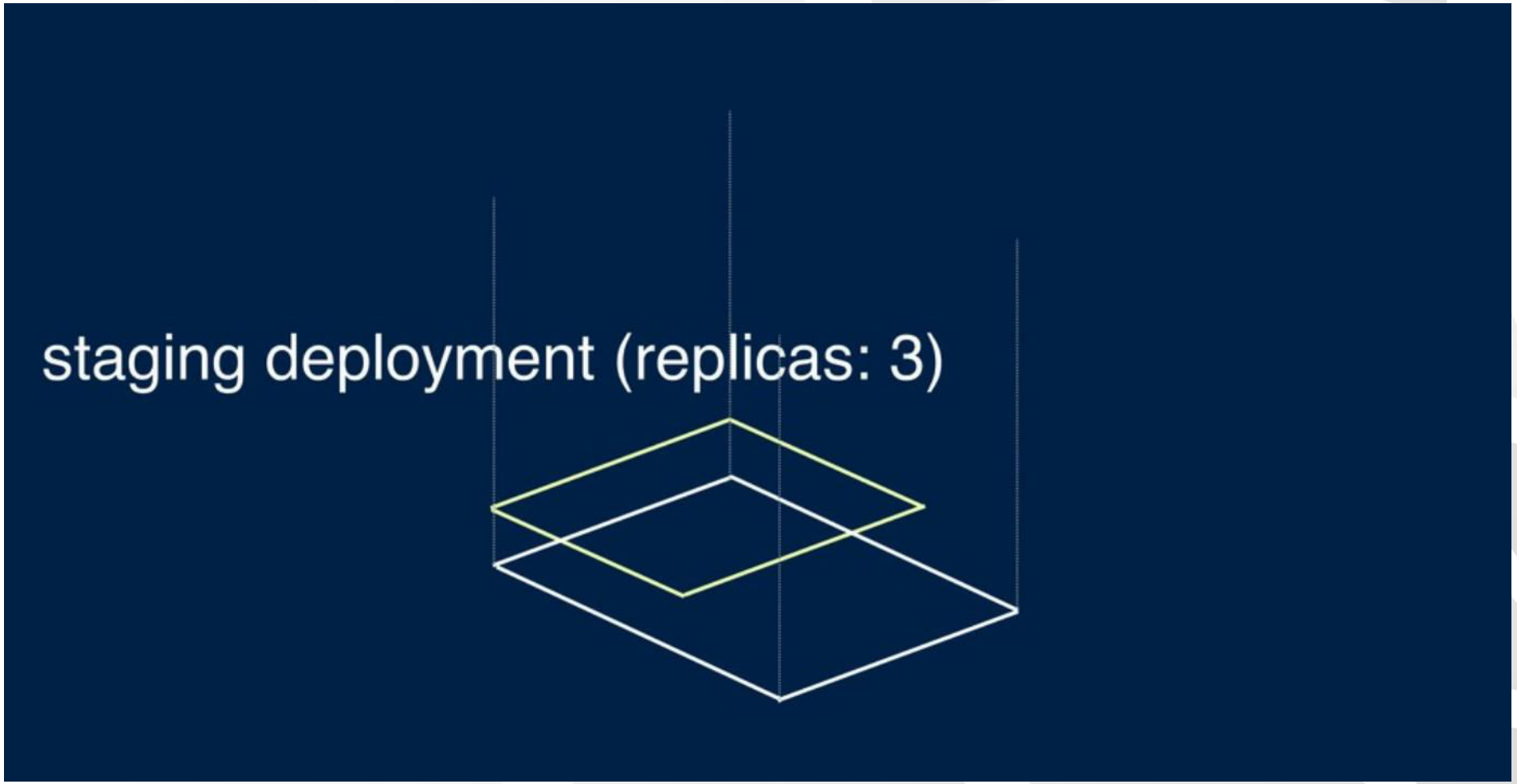
```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

resources:
  - configmap.yaml
  - service.yaml
  - deployment.yaml

commonLabels:
  deployment: kustomize

namespace: default
namePrefix: prefix-
nameSuffix: "-suffix"
```

```yaml
apiVersion: kustomize.config.k8s.io/v1beta1
kind: Kustomization

namespace: production
bases:
- ../../base

resources:
- namespace.yaml

patchesStrategicMerge:
- custom-env.yaml
```

Overlay Customization Model



Overlay Customization Model



Overlay Customization Model

Questions

Lab: Using Kustomize

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-kustomize

Noam Amrani

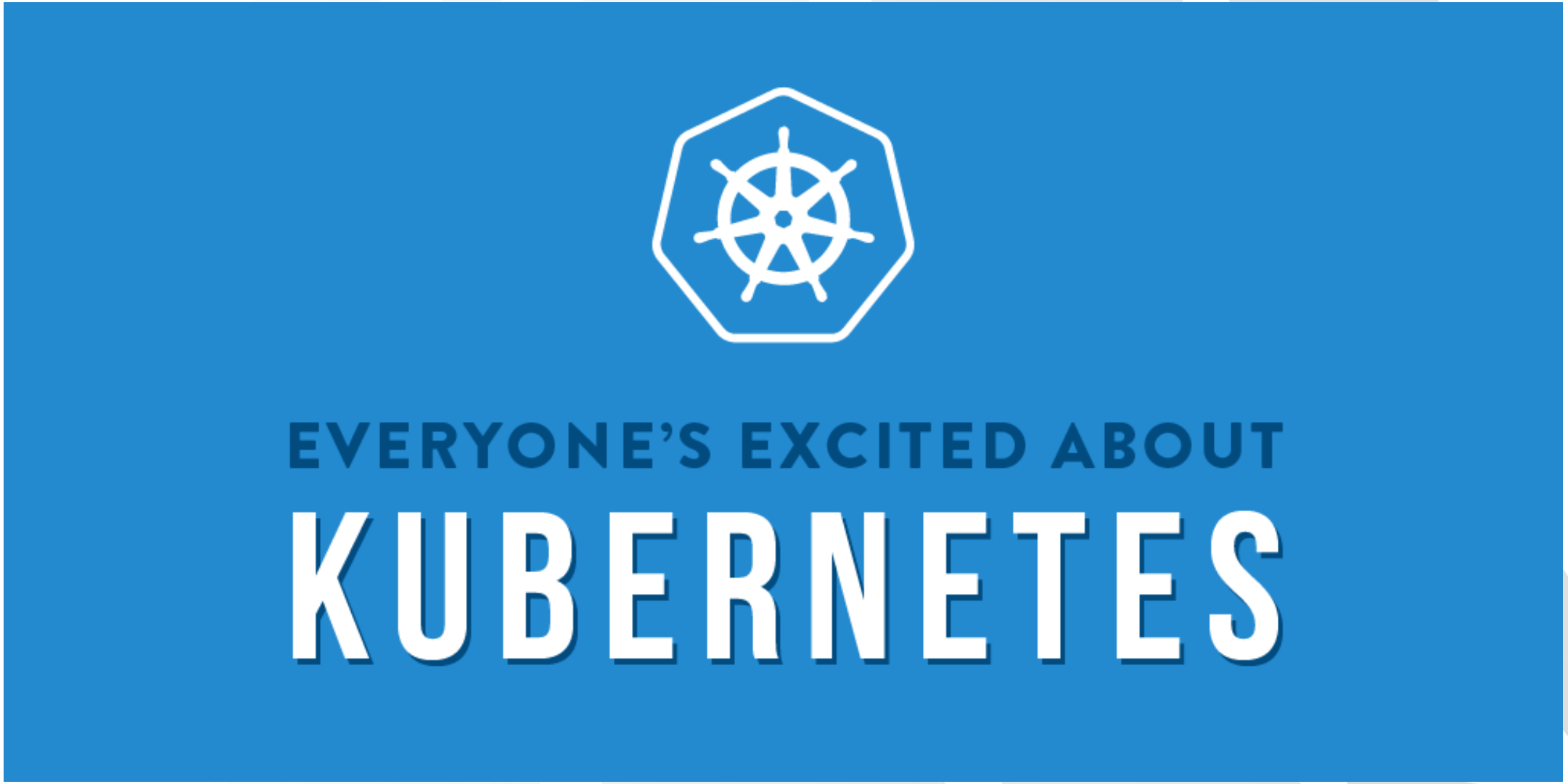
Module 01: Introduction

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

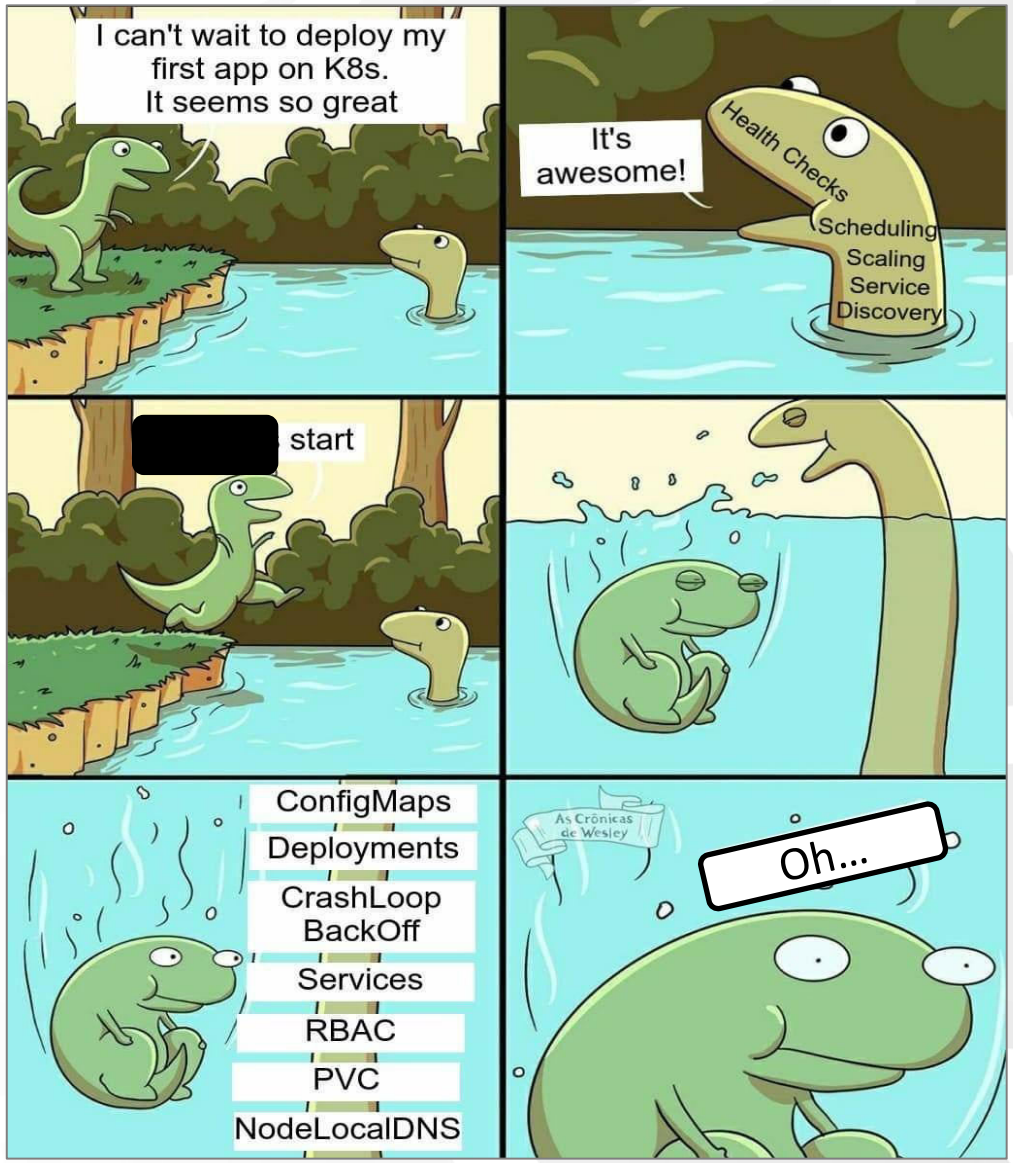
Agenda

- Workshop Objectives
- Workshop Agenda
- Kubernetes Introduction



Workshop Objectives

- Getting started with Kubernetes
- Learn Kubernetes through hands-on labs
- Understand how Kubernetes works
- And most importantly, have fun with Kubernetes!

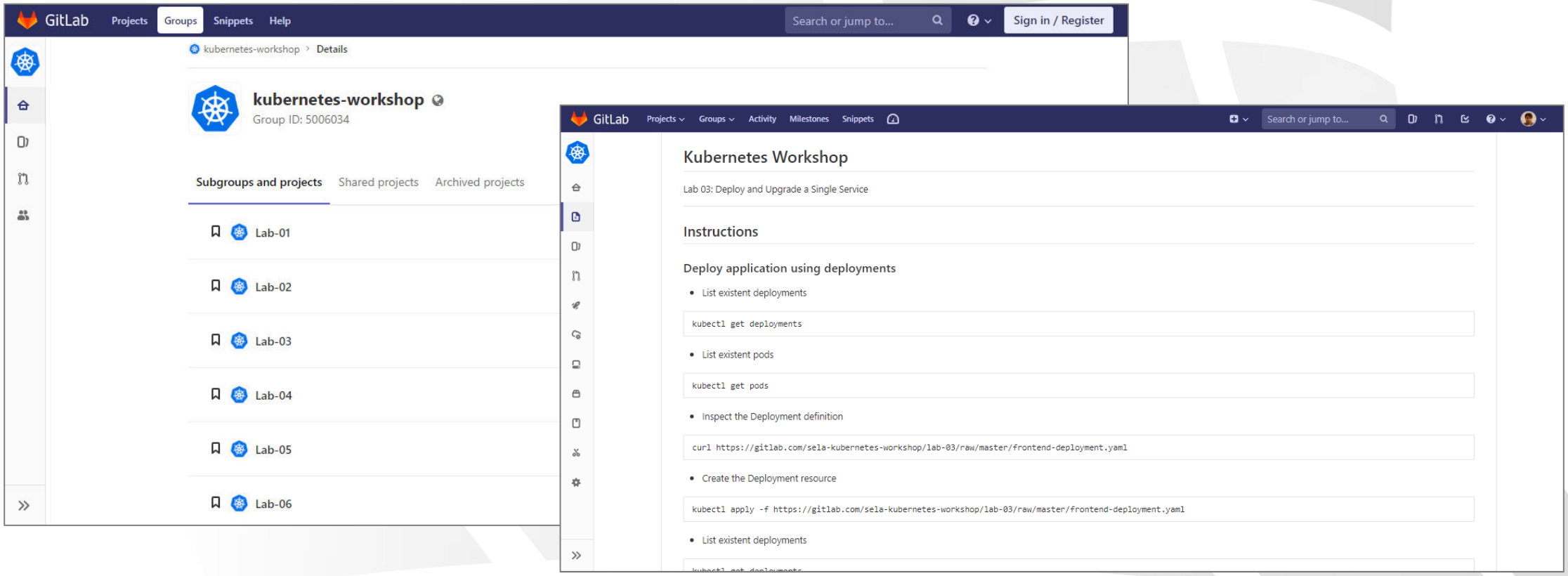


https://twitter.com/CloudRss/status/1274302509616697344/photo/1

Workshop Agenda

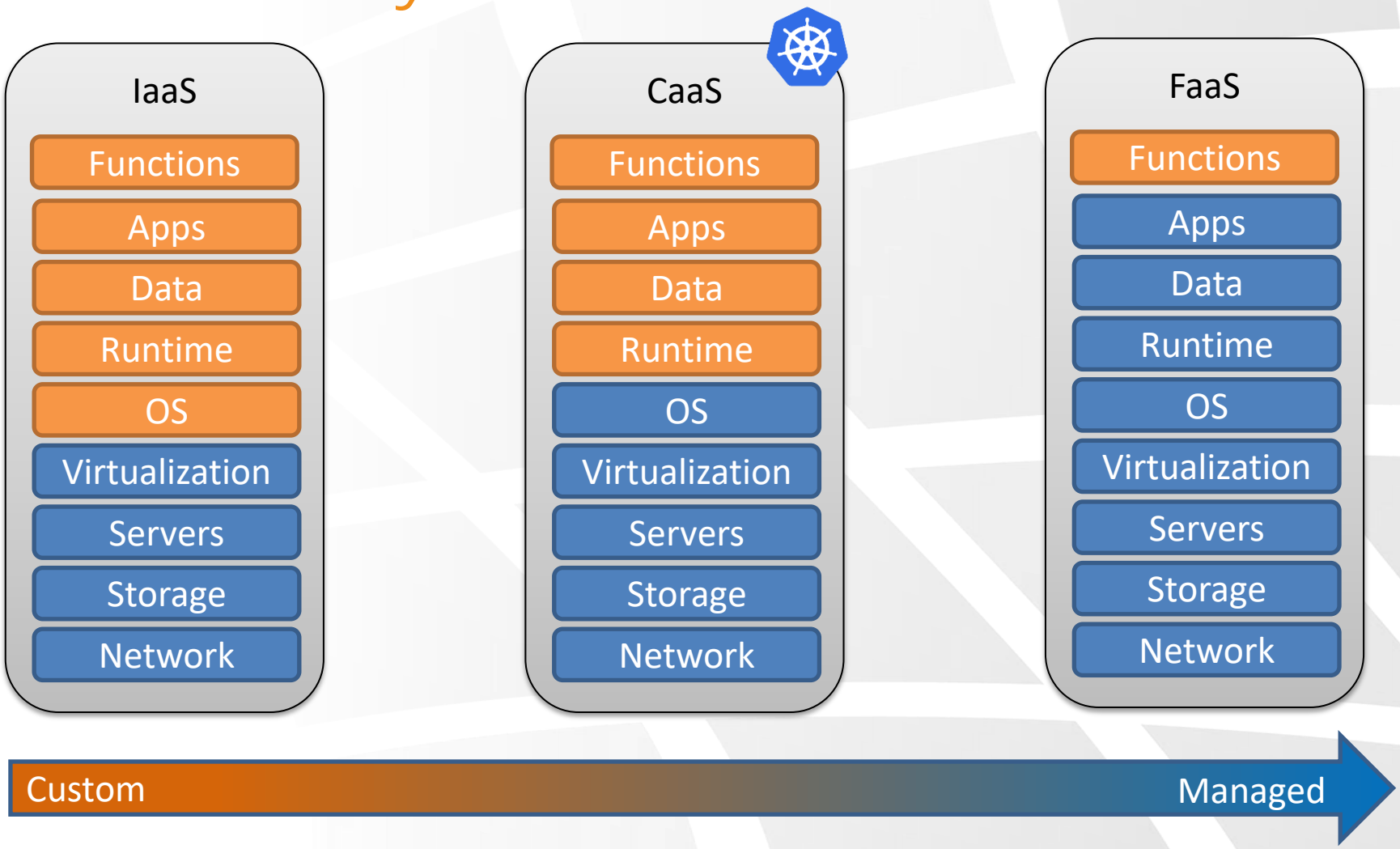
- Module 01: Introduction
- Module 02: Kubernetes Architecture
- Module 03: YAML & Kubectl
- Module 04: Kubernetes Basics
- Module 05: Deployments & Upgrades
- Module 06: Labels, selectors & Annotations
- Module 07: Kubernetes Networking
- Module 08: Services
- Module 09: Ingress
- Module 10: ConfigMaps & Secrets
- Module 11: Jobs & CronJobs
- Module 12: DaemonSets
- Module 13: Helm Package Manager
- Module 14: Managed Kubernetes
- Module 15: Advanced Scheduling
- Module 16: Autoscaling
- Module 17: Kubernetes Storage
- Module 18: StatefulSets
- Module 19: Microservices & Istio
- Module 20: Summary

Workshop Labs



https://gitlab.com/sela-kubernetes-workshop

Our ecosystem



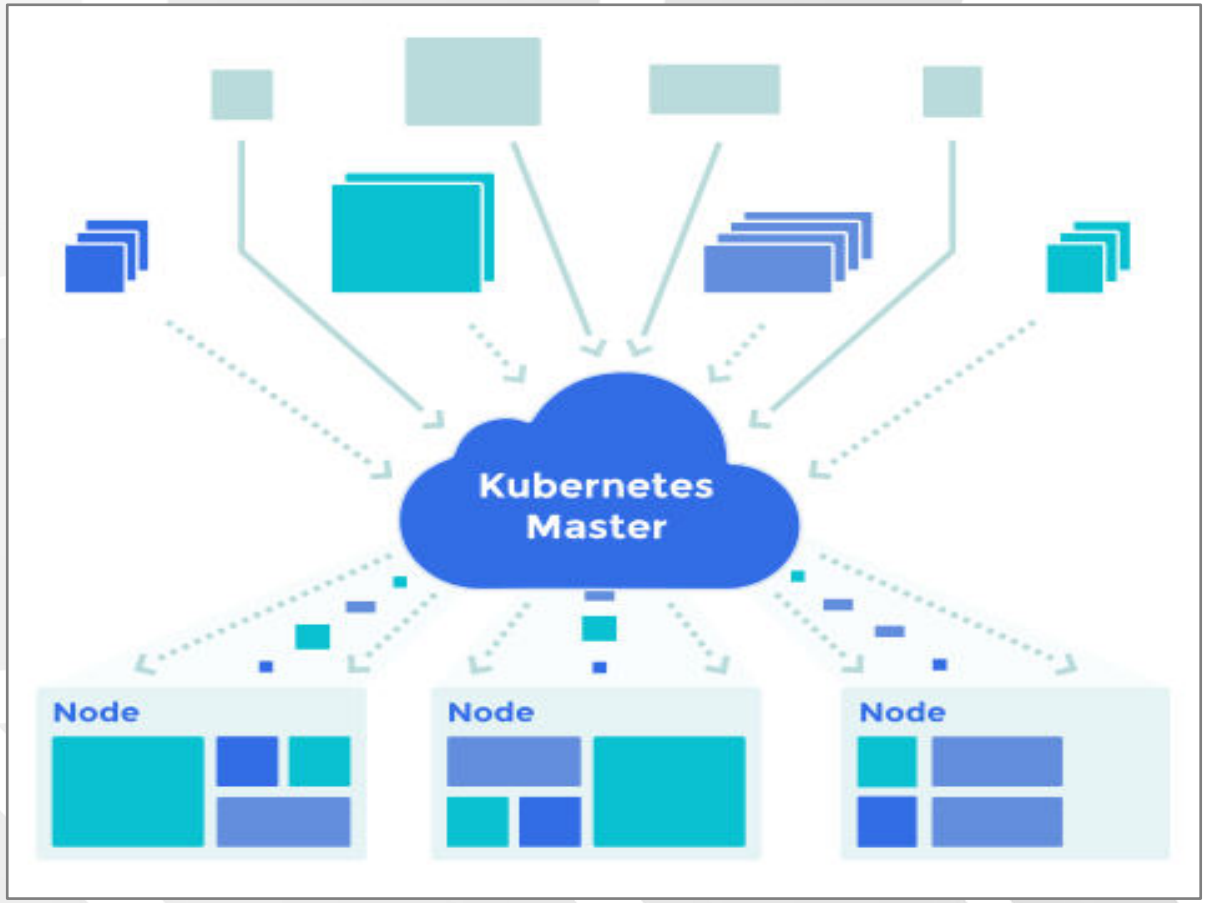
What is Kubernetes?

- Platform for container orchestration
- Kubernetes abstract away the underlying hardware (CPU, GPU, Mem, Disks)
- Scalable, Resilient, Self-Healing and Fault Tolerant
- Allow to run any kind of workload, on any cloud or on-prem setup
- Is responsible for maintaining the desired state
- Kubernetes encourages cattle; not pets (ephemeral)
- Individual machines don't matter
- Container isolates the app from the host

Kubernetes

Is used for

- Microservices
- CI/CD
- Machine Learning platforms
- Portable “cloud”



Kubernetes is the de facto platform for running containers today

How it works?

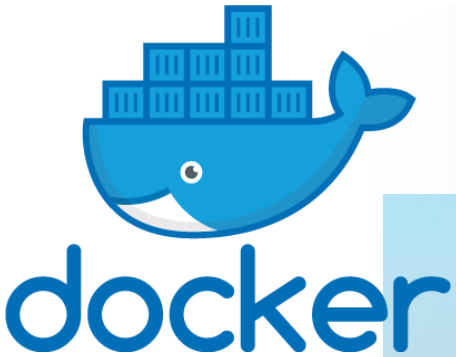
- Kubernetes has a declarative API.
- Apply the desired configuration to your cluster.
- Kubernetes will drive "current state" to the "desired state" eventually.

- Container crashed?
  - Restart it.
- Container unhealthy?
  - Remove from LB and Reschedule to another node.
- Container overloaded?
  - Add more replicas automatically.

How did it all started?

- Google spawns more than **2 billion** containers per week in their data center!
- Google have developed it's own tools – namely **Borg**, and **Omega** – systems that are running until now and are proven for production in massive scale
- Using all the experience gathered using those systems, Google started in 2014 the **Kubernetes** project (GoLang), and was developed as OSS project from the get go
- It's first stable release (v1.0.1) was at **July 2015**
- Kubernetes is developing quickly with new features every month or two
- V1.18 was released in May 2020

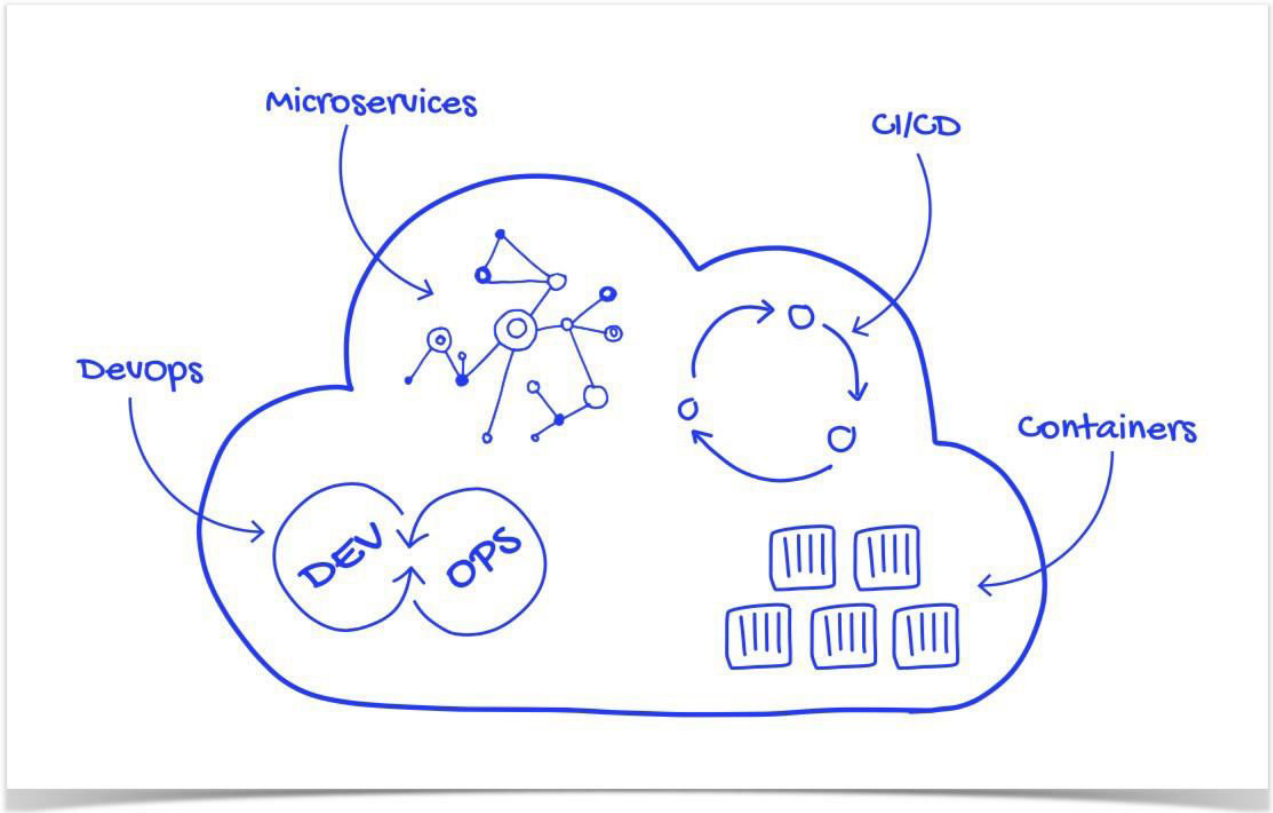
Whats behind a name and a logo?



Google Borg

A word about Cloud Native Architecture

- An **application architecture** designed to leverage the **strengths** and accommodate the **challenges** of a **standardized** cloud environment, including concepts such as **elastic scaling**, **immutable deployment**, **disposable instances**, and **less predictable infrastructure**.



CNCF – Cloud Native Computing Foundation

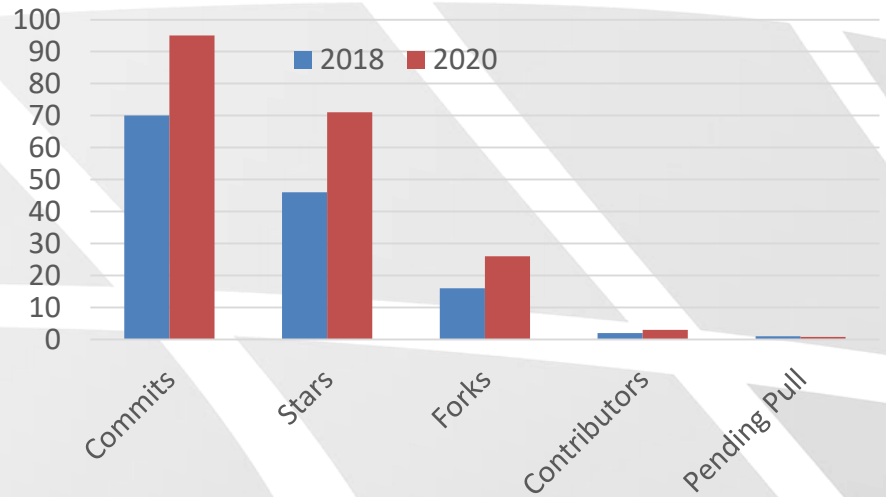
- Kubernetes is the first project that graduated CNCF
  - Later came Prometheus, Envoy proxy
- The CNCF is a child entity of the Linux Foundation and operates as a vendor neutral governance group.
- CNCF is managing Kubernetes Roadmap and certifications.

The Kubernetes Journey

- Recent surveys from from CNCF 2019 shows that over **78%** of the respondence are using using **Kubernetes** in **production**

- It has a huge community:

github	2018	2020
Commits	70K	95K
Stars	46K	71K
Forks	16K	26K
Contributors	2K	3K
Pending Pull	1000	807



The Kubernetes Journey – CNCF 2019

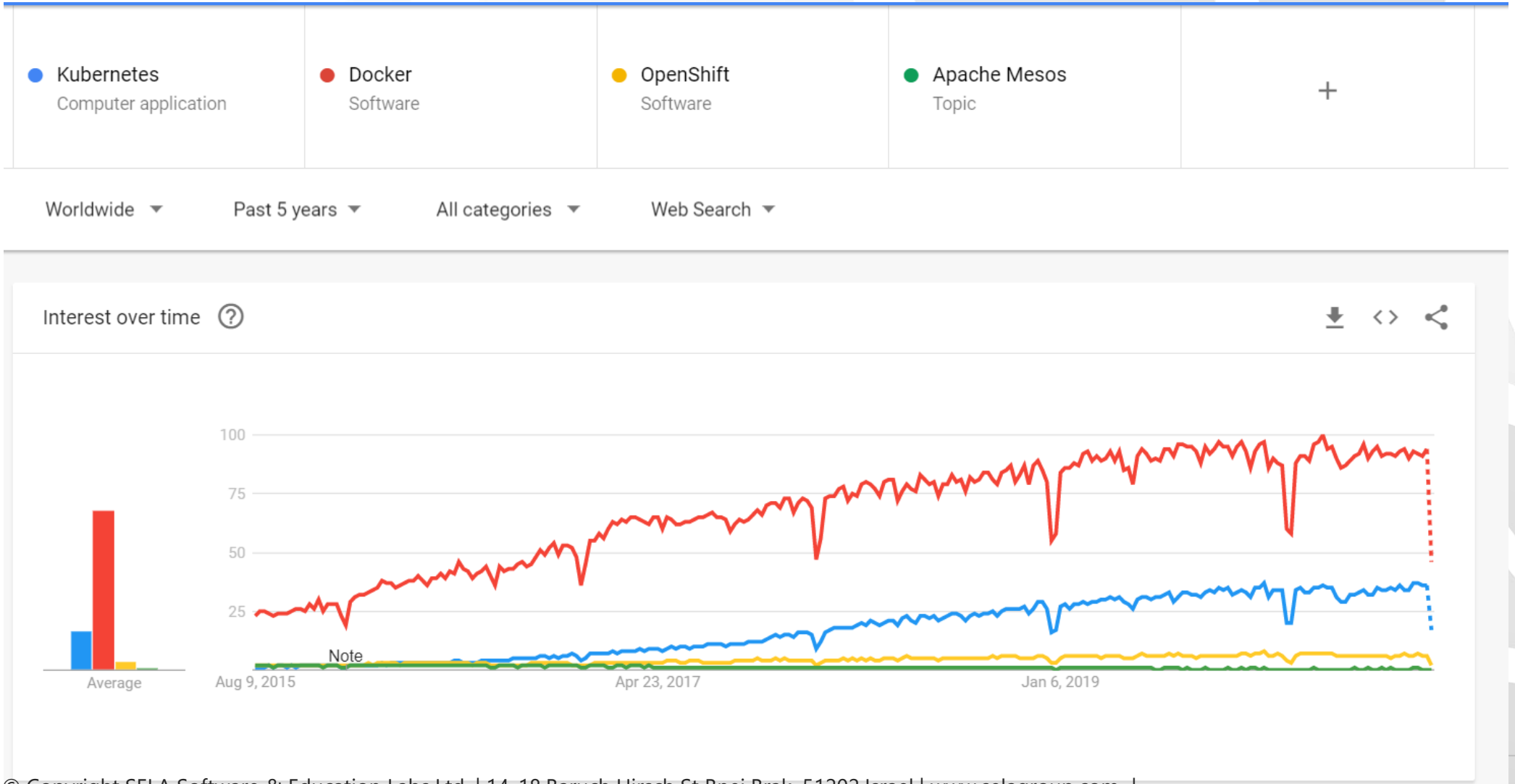
Key Takeaways:

- Usage in production has increased for almost all CNCF projects.
- 78% of respondents are using Kubernetes in production, a huge jump from 58% last year.
- 18% of respondents indicated they are using a service mesh in production, while a total of 47% are evaluating the use of service mesh in their organization.
- At least 41% of respondents are using serverless technologies.
- An increase in both the number and reliability of CI/CD tools is driving a decrease in manual release cycles and an acceleration of release cycles.
- The use of containers in production has increased significantly – 84% of respondents are using containers in production, a jump of more than 15% from 2018.

https://www.cncf.io/wp-content/uploads/2020/03/CNCF_Survey_Report.pdf

CNCF SURVEY: DEPLOYMENTS ARE GETTING LARGER AS CLOUD NATIVE ADOPTION BECOMES MAINSTREAM 1

Kubernetes in Search Trends



Built-in Features

- Monitoring Data
- Health Checking mechanisms
- Horizontal Auto Scaling
- Service Discovery
- Rolling Deployment and Rollback
- Logging
- Application Load Balancing

K8s biggest feature is it's huge ecosystem

Questions

Noam Amrani

Module 02: Kubernetes Architecture

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

- Workshop Objectives
- Workshop Agenda
- Kubernetes Introduction

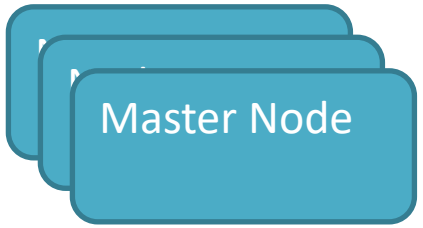
Agenda

- Core Concepts
- High-Level components Architecture
- Master Components
- Worker Node Components
- Putting All Together
- Additional Services

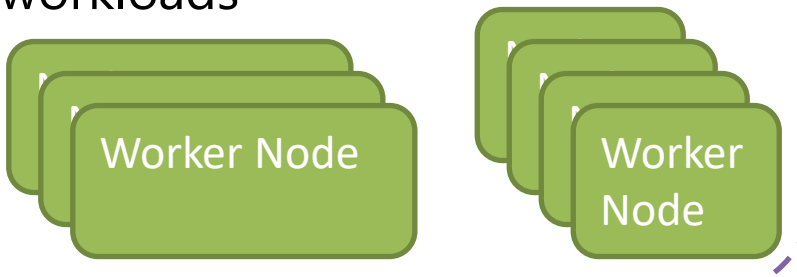
Core Concepts

Cluster: A collection of hosts that aggregate their available resources including CPU, GPU, RAM, disks, and other devices into a usable pool.

Control Plane (master)
Represents a collection of components that make up the control plane of Kubernetes.



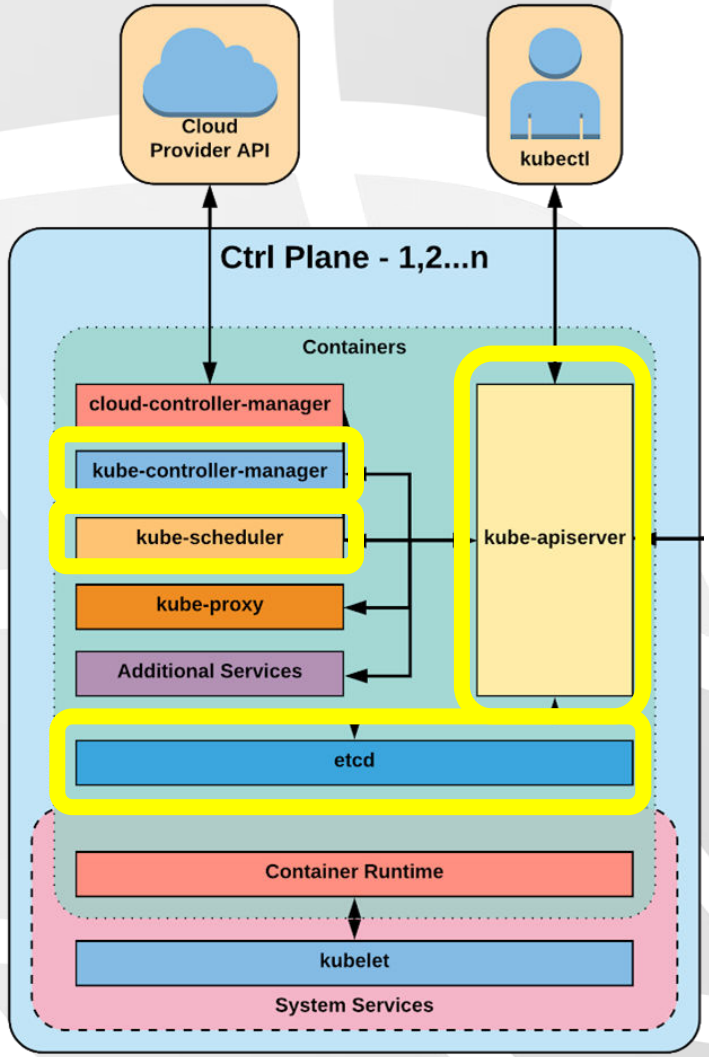
Workers
Represents a collection of components that runs the workloads



Node: A single host, physical or virtual capable of running pods

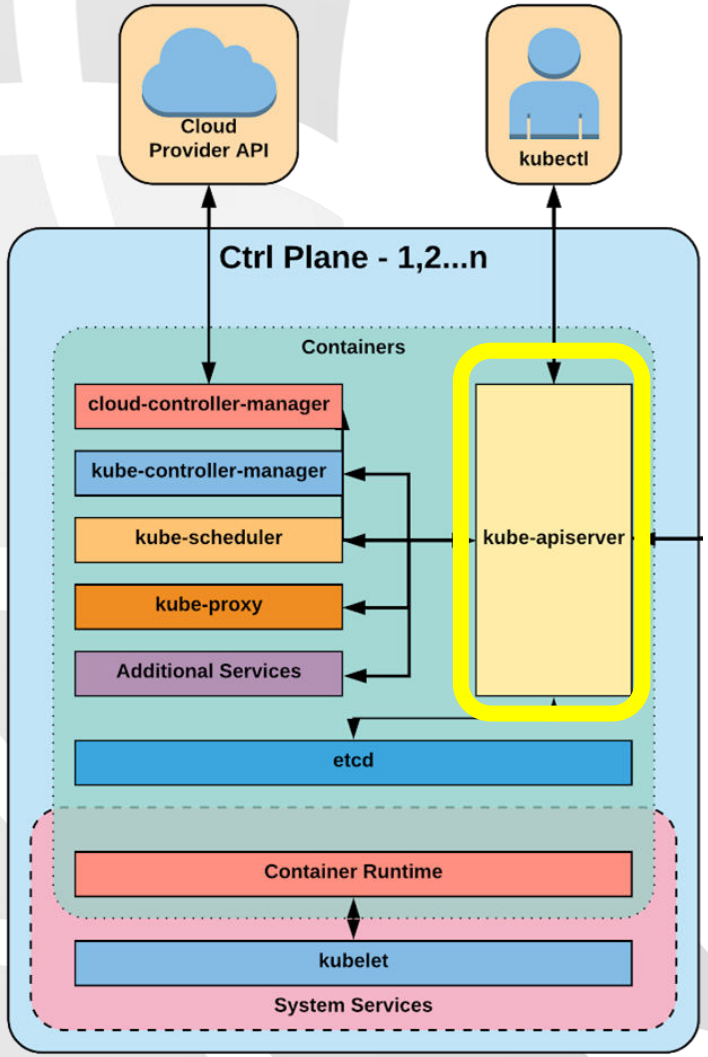
Master Components

- API Server
- etcd
- Controller Manager
- Kube-Scheduler



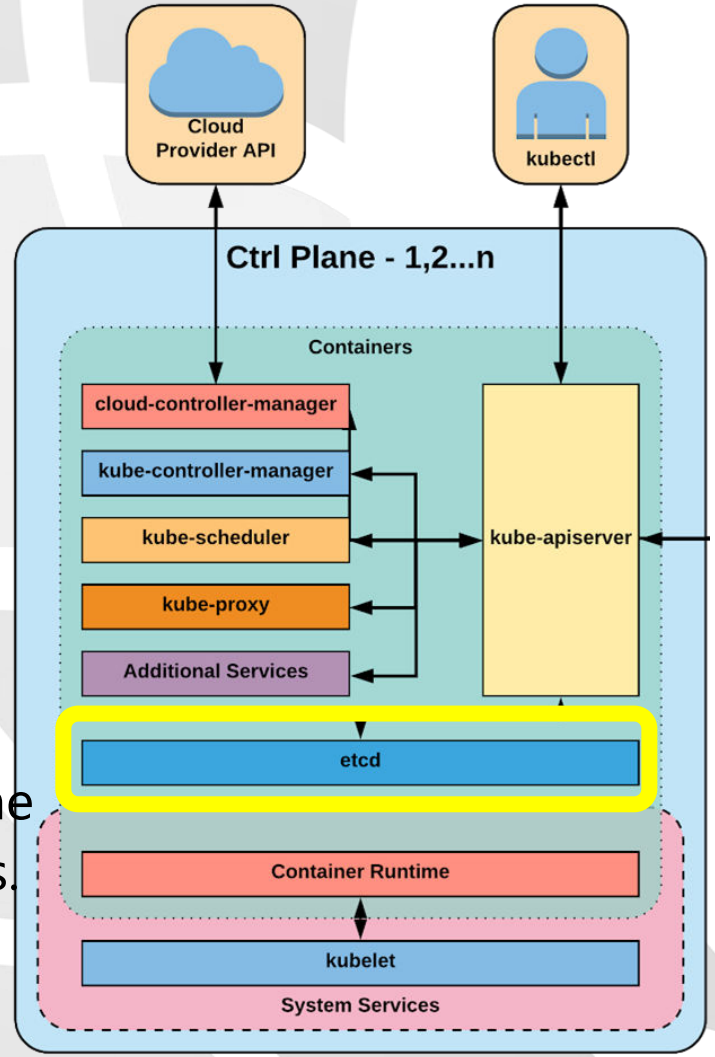
Kube-APIServer

- Provides a forward facing **REST interface** into the kubernetes control plane and datastore.
- All clients and other applications **interact** with kubernetes strictly through the API Server.
- Acts as the gatekeeper to the cluster by handling authentication and authorization, request validation, mutation, and admission control in addition to being the front-end to the backing datastore
- API server define a versioned and backward compatible interface (REST or gRPC) that enable integration with other tools



etcd

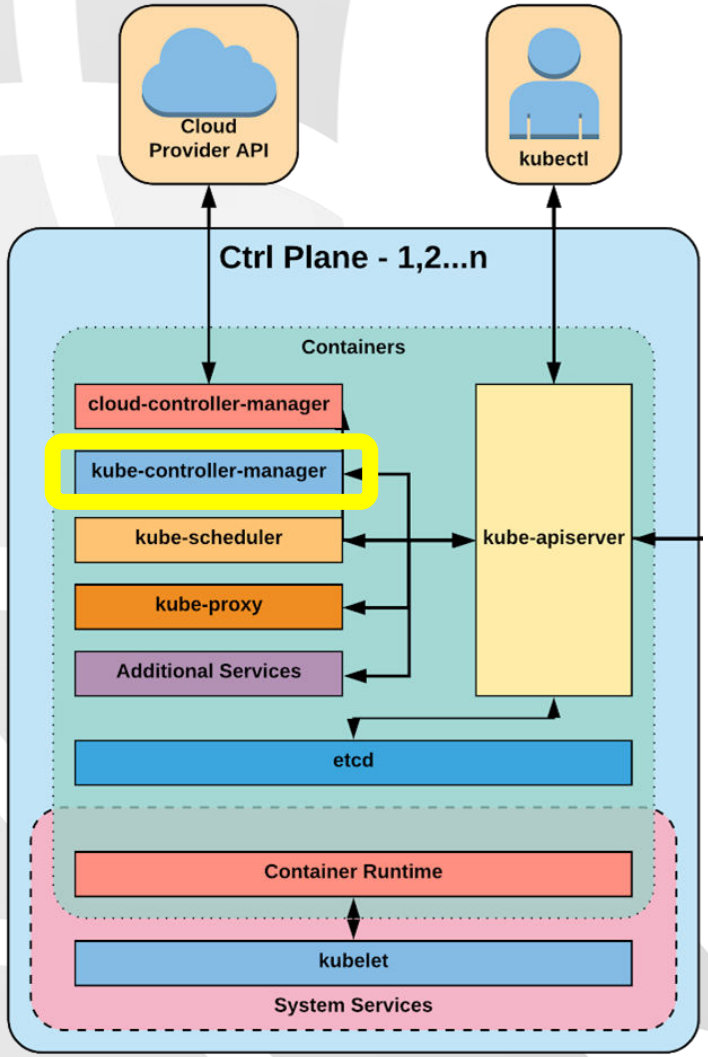
- etcd acts as the cluster datastore.
- Purpose in relation to Kubernetes is to provide a strong, consistent and highly available key-value store for persisting cluster state.
- Stores objects and config information.
- etcd is a very-fast, high available key-value store that can be distributed among multiple nodes.
- etcd is accessible only by Kubernetes API server as it may have some sensitive information, and should not be used by other applications.



etcd

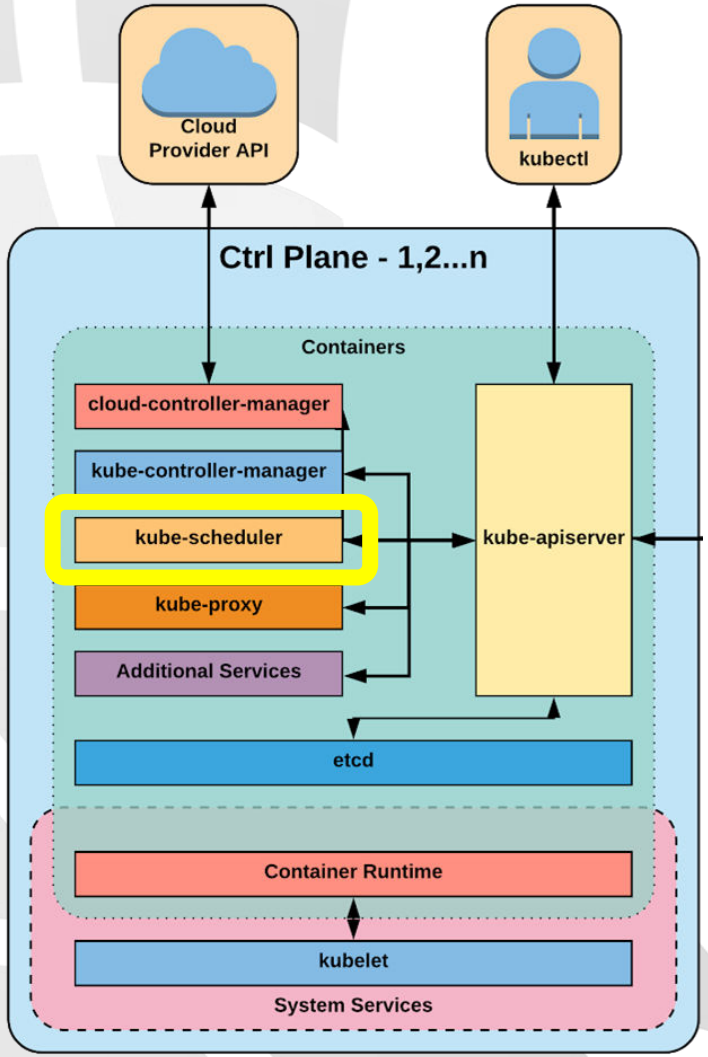
Controller Manager

- Serves as the primary daemon that manages all core component control loops.
- Monitors the cluster state via the apiserver and steers the cluster towards the desired state
- Manages most of the Core Building Blocks in Kubernetes



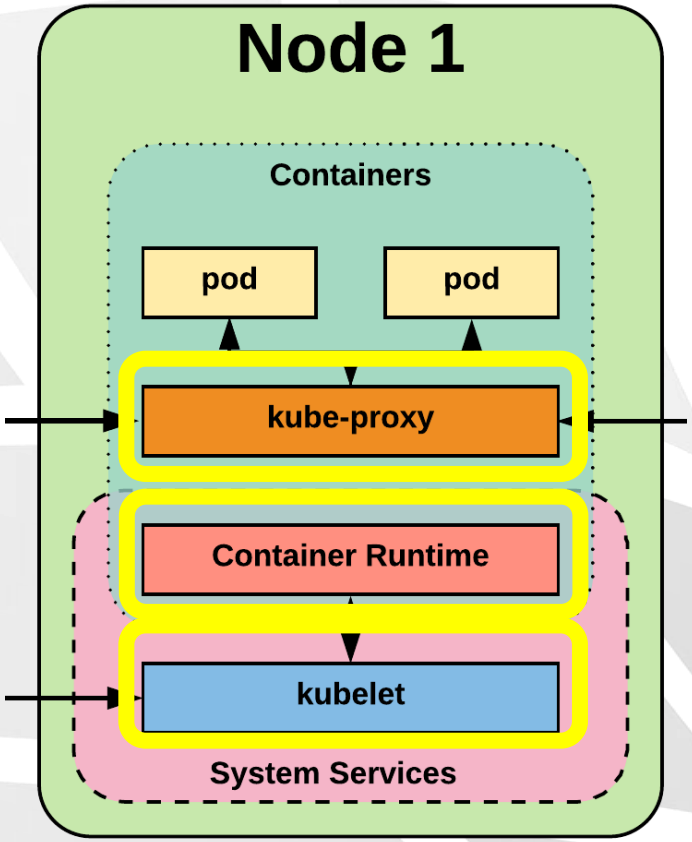
Kube-Scheduler

- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.
- Default scheduler uses bin packing.
- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements
- The Scheduler is located on the master and responsible for making scheduling and pods distribution decisions
- Most decisions are as simple as “This **pod** should run on that **node**”



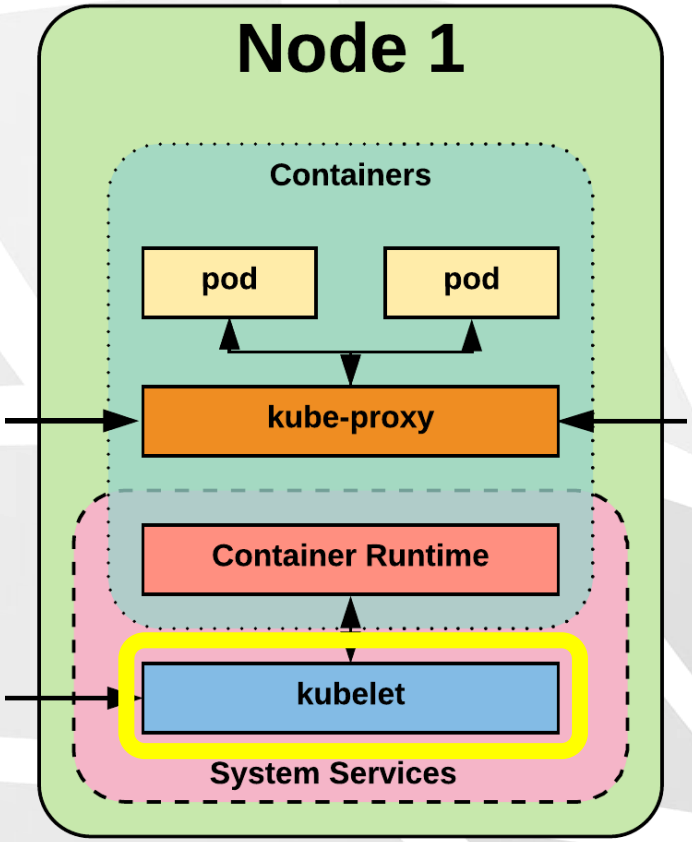
Worker Node Components

- Kubelet
- Kube-proxy
- Container Runtime Engine



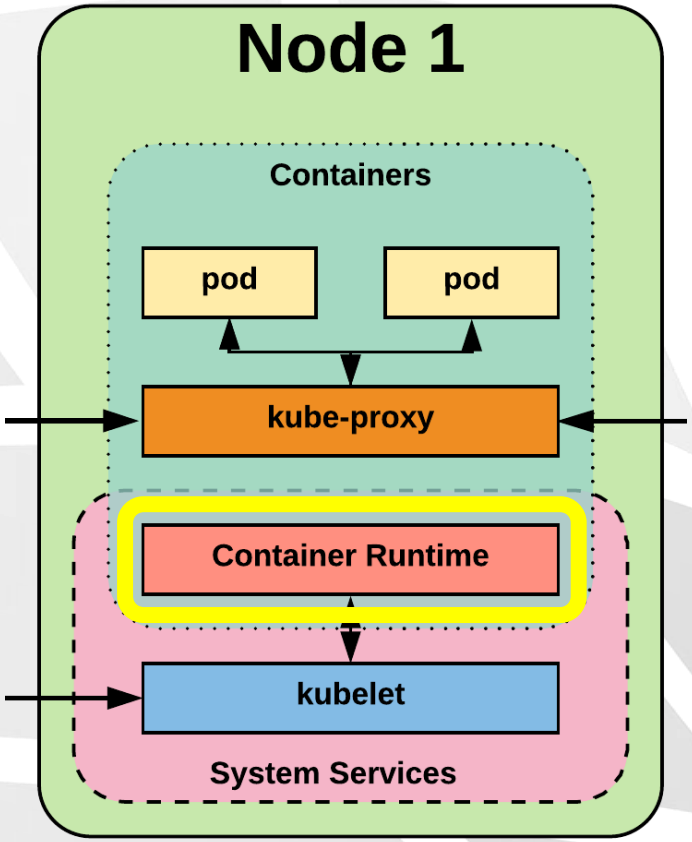
Kubelet Service

- Kubelet is the cluster agent to that is placed on every node in the cluster, and is responsible for relaying information to and from the control plane
- Kubelet communicates with the master components, and receives instructions – i.e run a pod with those containers, expose port on the node, update routing tables, etc..
- Kubelet then assumes responsibility for maintaining the state the node, and the pods that are scheduled to run on the node, as instructed by the master.



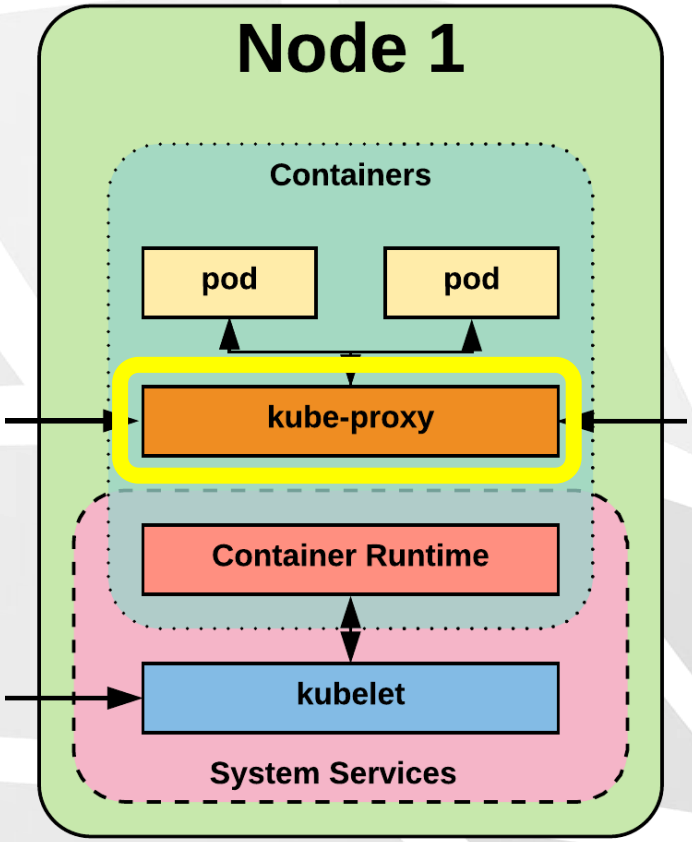
Container Runtime Engine

- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
  - Docker
  - Containerd
  - Cri-o
  - Rkt
  - Kata (formerly clear and hyper)
  - Virtlet (VM CRI compatible runtime)



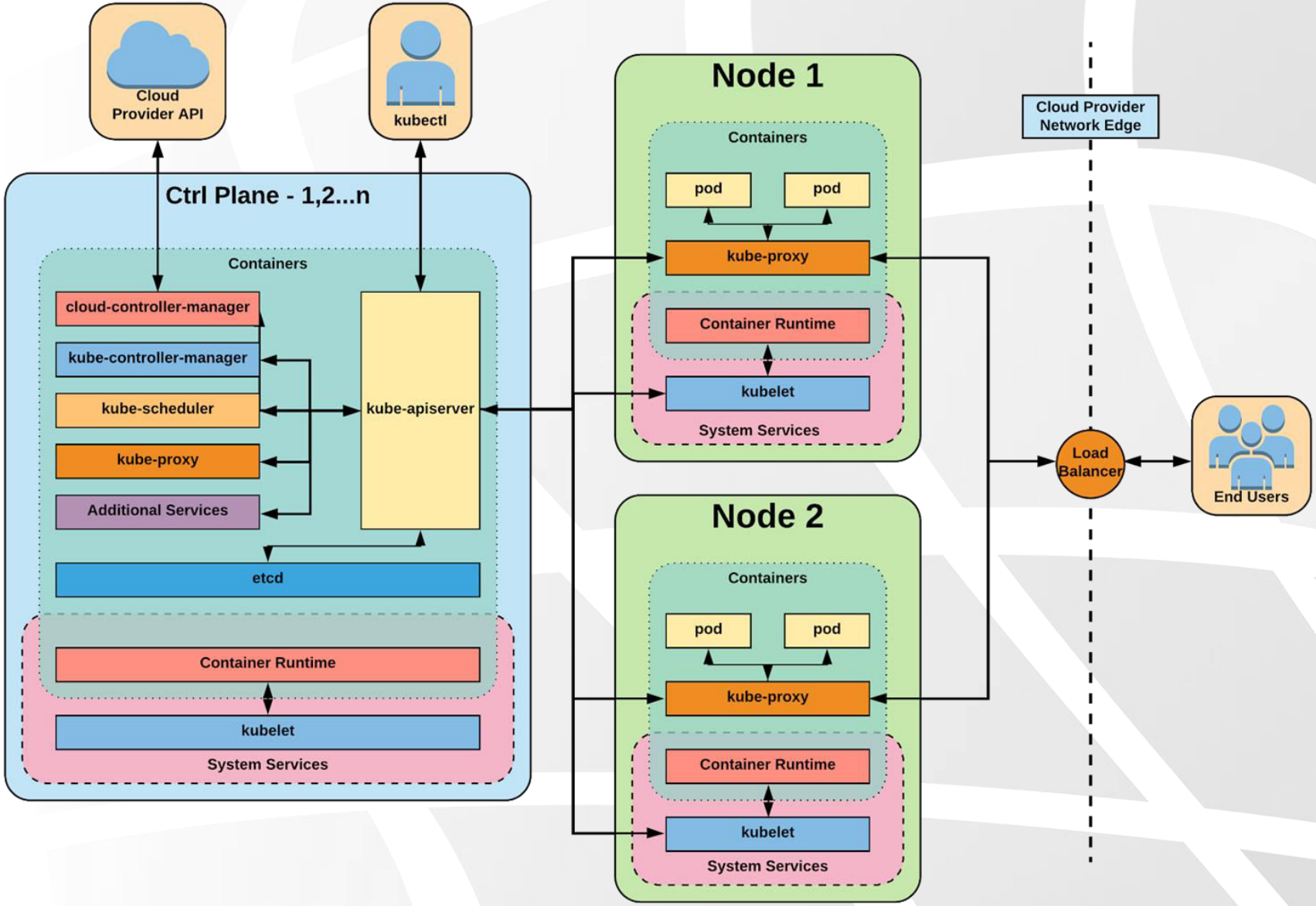
KubeProxy Service

- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.
- It makes sure that the networking environment is predictable and accessible and at the same time isolated.



Additional Services (Optional)

- **cloud-controller-manager**
  - Daemon that provides cloud-provider specific knowledge and integration capability into the core control loop of Kubernetes
- **Cluster DNS**
  - Provides Cluster Wide DNS for Kubernetes Services
- **Metrics API Server**
  - Provides metrics for use with other Kubernetes Components.



Questions

Noam Amrani

Module 03: YAML and Kubectl

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

- YAML
- Kubectl
- Lab 01: Setting Up Your Workstation

YAML

- Stands for **YAML Ain't Markup Language**
- Is a human-readable text-based format for specifying configuration-type information
- Fortunately, there are only two types of structures you need to know about in YAML:
  - Lists
  - Maps

YAML - Maps

- Maps let you associate name-value pairs

```yaml
---
apiVersion: v1
kind: Pod
```

YAML

```json
{
   "apiVersion": "v1",
   "kind": "Pod"
}
```

JSON

YAML - Lists

- Lists are literally a sequence of objects

```yaml
args:
  - sleep
  - "1000"
  - message
  - "Bring back Firefly!"
```

YAML

```json
{
   "args": ["sleep", "1000", "message", "Bring back Firefly!"]
}
```

JSON

Kubernetes YAML

```yaml
apiVersion: v1
kind: Pod
metadata:
  namespace: default
  name: sample-pod
spec:
  containers:
  - image: nginx
    name: container-name
```

```yaml
apiVersion: apps/v1beta2
kind: ReplicaSet
metadata:
  name: nginx
  annotations:
    description: "nginx frontend"
  labels:
    app: nginx
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
      tier: frontend
  template:
    metadata:
      labels:
        app: nginx
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

```yaml
apiVersion: apps/v1beta2
kind: Deployment
metadata:
  name: nginx
  annotations:
    description: "nginx frontend"
  labels:
    app: nginx
    tier: frontend
spec:
  replicas: 3
  mindReadySeconds: 10
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 5
      maxUnavailable: 2
  selector:
    matchLabels:
      app: nginx
      tier: frontend
  template:
    metadata:
      labels:
        app: nginx
        tier: frontend
    spec:
      containers:
      - name: nginx
        image: nginx:latest
        ports:
        - containerPort: 80
```

```yaml
kind: Service
apiVersion: v1
metadata:
  name: nginx
spec:
  type: ClusterIP
  selector:
    app: nginx
  ports:
  - protocol: TCP
    port: 80
    targetPort: 80
```

Kubectl

- Kubectl is a command line interface for running commands against Kubernetes clusters
- Allow for listing and querying objects, creating and modifying objects, port-forwarding, accessing logs, ssh to containers, etc…

Everything you do in Kubernetes, you do with kubectl

Kubectl

```
$ kubectl <command> <resource> [arguments] [-n <namespace>]
```

$ kubectl get pods

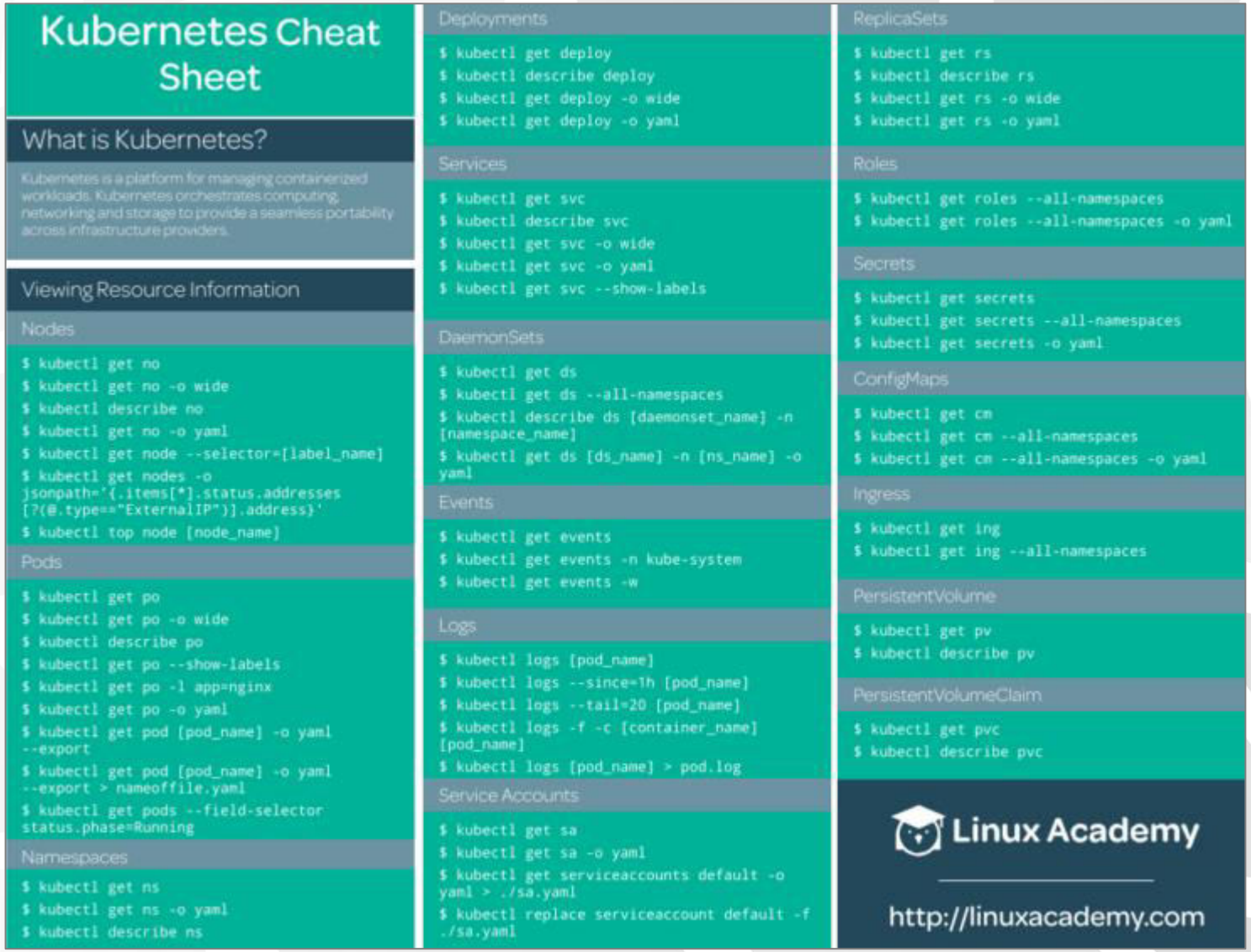
$ kubectl get svc -n dev

$ kubectl describe pod <pod-name>

$ kubectl edit svc <service-name>

$ kubectl get pods --all-namespaces

Kubectl



Kubernetes Cheat Sheet

What is Kubernetes?

Kubernetes is a platform for managing containerized workloads. Kubernetes orchestrates computing, networking and storage to provide a seamless portability across infrastructure providers.

Viewing Resource Information

Nodes

```
$ kubectl get no
$ kubectl get no -o wide
$ kubectl describe no
$ kubectl get no -o yaml
$ kubectl get node --selector=[label_name]
$ kubectl get nodes -o
jsonpath='{.items[*].status.addresses
[?(@.type=="ExternalIP")].address}'
$ kubectl top node [node_name]
```

Pods

```
$ kubectl get po
$ kubectl get po -o wide
$ kubectl describe po
$ kubectl get po --show-labels
$ kubectl get po -l app=nginx
$ kubectl get po -o yaml
$ kubectl get pod [pod_name] -o yaml
--export
$ kubectl get pod [pod_name] -o yaml
--export > nameoffile.yaml
$ kubectl get pods --field-selector
status.phase=Running
```

Namespaces

```
$ kubectl get ns
$ kubectl get ns -o yaml
$ kubectl describe ns
```

Deployments

```
$ kubectl get deploy
$ kubectl describe deploy
$ kubectl get deploy -o wide
$ kubectl get deploy -o yaml
```

Services

```
$ kubectl get svc
$ kubectl describe svc
$ kubectl get svc -o wide
$ kubectl get svc -o yaml
$ kubectl get svc --show-labels
```

DaemonSets

```
$ kubectl get ds
$ kubectl get ds --all-namespaces
$ kubectl describe ds [daemonset_name] -n
[namespace_name]
$ kubectl get ds [ds_name] -n [ns_name] -o
yaml
```

Events

```
$ kubectl get events
$ kubectl get events -n kube-system
$ kubectl get events -w
```

Logs

```
$ kubectl logs [pod_name]
$ kubectl logs --since=1h [pod_name]
$ kubectl logs --tail=20 [pod_name]
$ kubectl logs -f -c [container_name]
[pod_name]
$ kubectl logs [pod_name] > pod.log
```

Service Accounts

```
$ kubectl get sa
$ kubectl get sa -o yaml
$ kubectl get serviceaccounts default -o
yaml > ./sa.yaml
$ kubectl replace serviceaccount default -f
./sa.yaml
```

ReplicaSets

```
$ kubectl get rs
$ kubectl describe rs
$ kubectl get rs -o wide
$ kubectl get rs -o yaml
```

Roles

```
$ kubectl get roles --all-namespaces
$ kubectl get roles --all-namespaces -o yaml
```

Secrets

```
$ kubectl get secrets
$ kubectl get secrets --all-namespaces
$ kubectl get secrets -o yaml
```

ConfigMaps

```
$ kubectl get cm
$ kubectl get cm --all-namespaces
$ kubectl get cm --all-namespaces -o yaml
```

Ingress

```
$ kubectl get ing
$ kubectl get ing --all-namespaces
```

PersistentVolume

```
$ kubectl get pv
$ kubectl describe pv
```

PersistentVolumeClaim

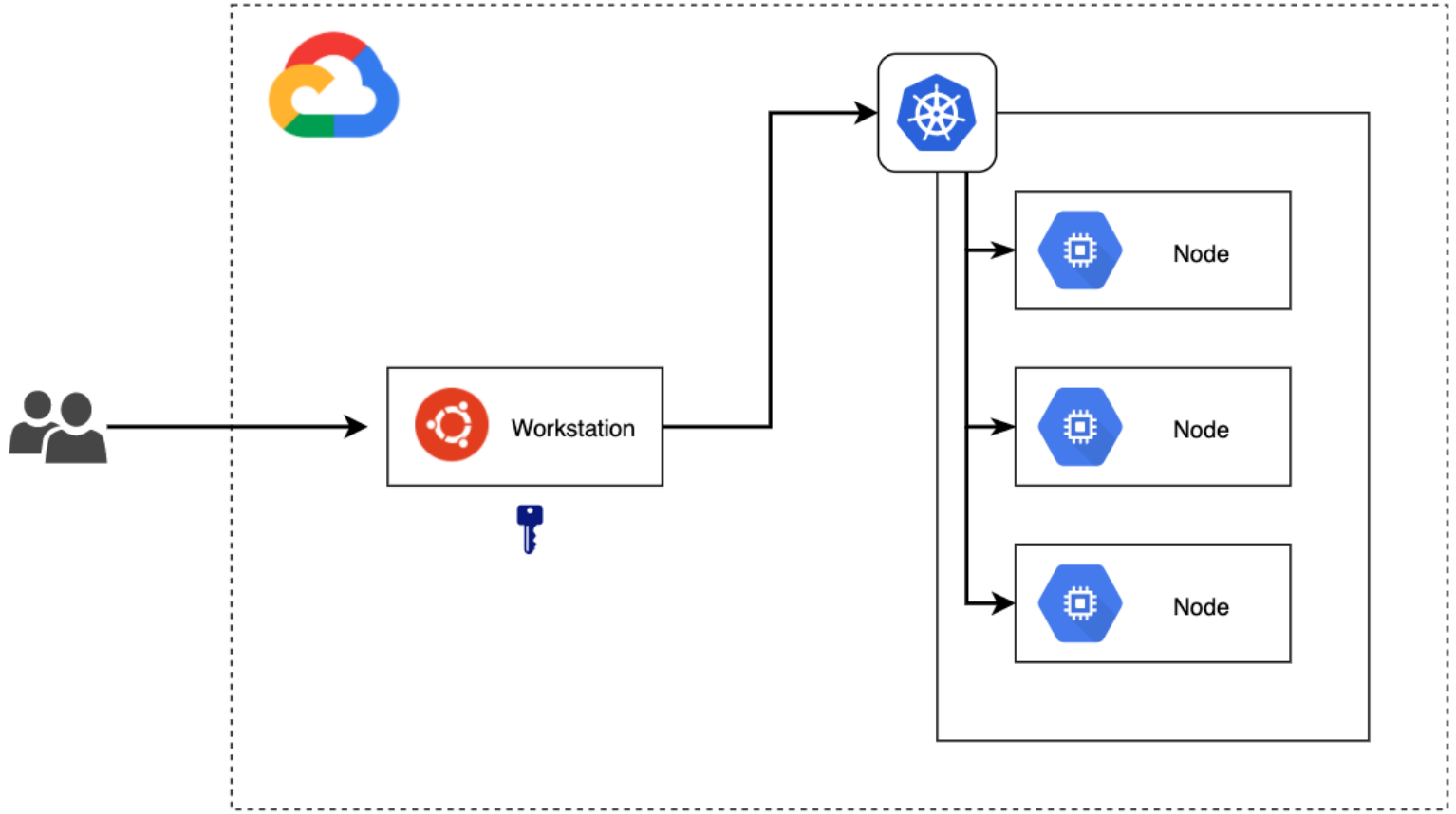
```
$ kubectl get pvc
$ kubectl describe pvc
```

Linux Academy

http://linuxacademy.com

Questions

Lab architecture



Lab 01: Setting Up Your Workstation

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-01

Noam Amrani

Module 04: Kubernetes Basics

Kubernetes Workshop

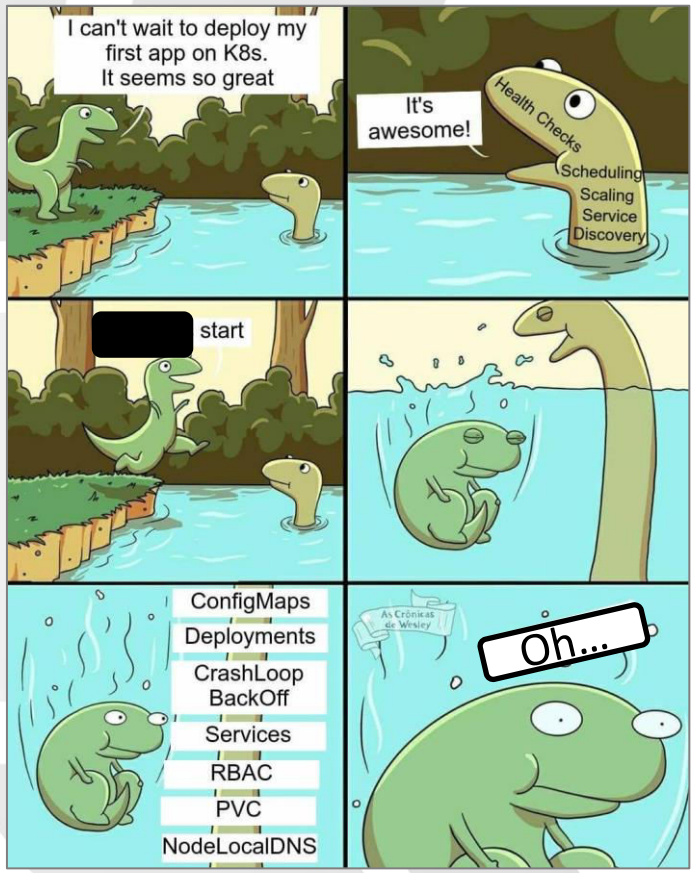
https://www.linkedin.com/in/noamamrani

Agenda

- Kubernetes Building Blocks
- Namespaces
- Pods
- Replication Sets
- Lab 02: Creating Our First Pod

Kubernetes Building Blocks

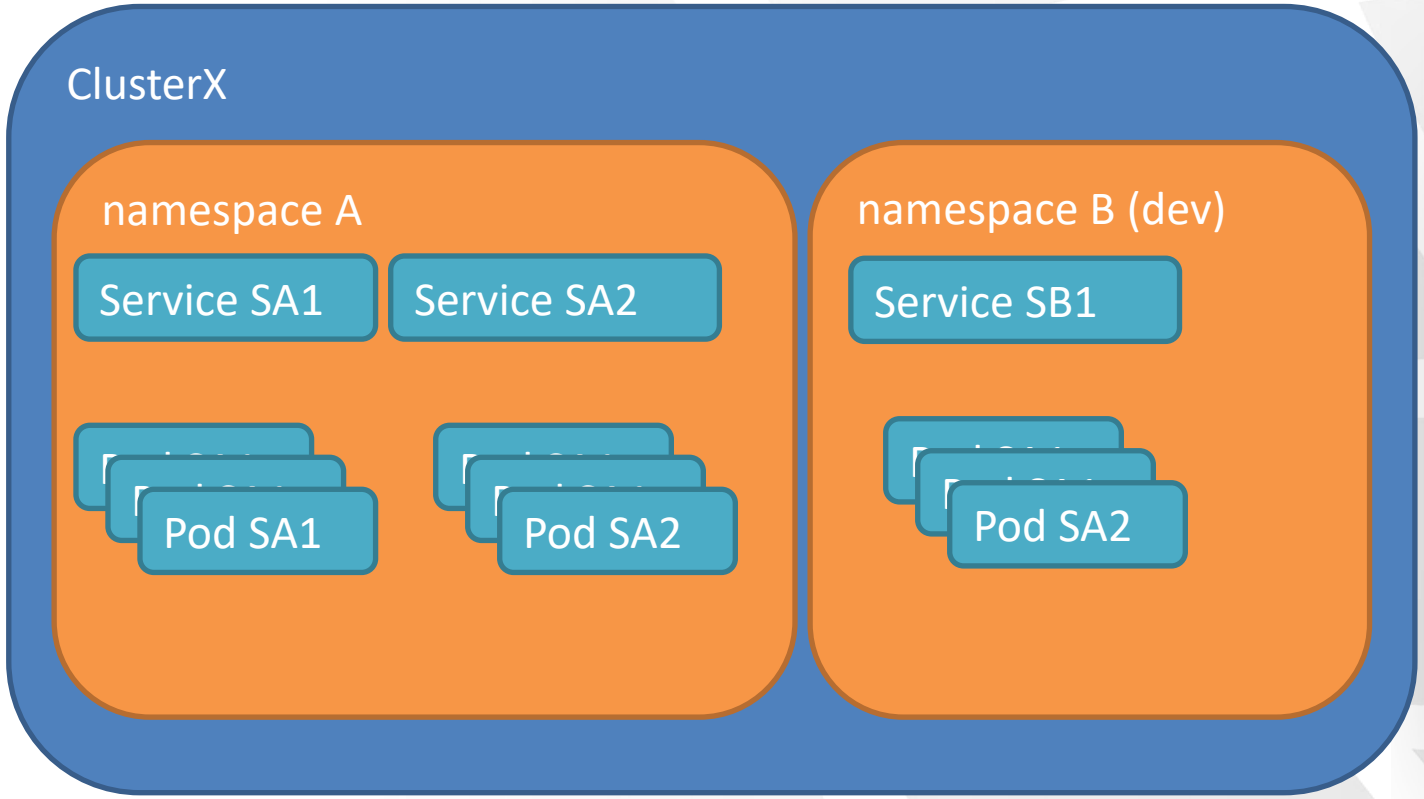
- Namespace
- Pod
- ReplicaSet
- Deployment
- Services
- Ingress
- ConfigMaps
- Secrets
- Jobs
- CronJobs
- DaemonSet
- StatefulSet
- Volumes
- PersistentVolumes
- PersistentVolumeClaims
- StorageClasses



Namespace

- A Namespace is a logical isolation method, and most Kubernetes objects are scoped in a Namespace
- A Namespace often used to group logically similar workloads, and enforce different policies
- Namespaces can also allow multi-tenancy
  - For example, Namespace per environment (dev, test | stage, prod), Namespace per Team or a developer, Namespace per PR, etc..
  - Role-Based Access Control (RBAC) **should** be used to limit specific access to resources or namespaces.

Namespace




```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: dev
  labels:
    name: development
```

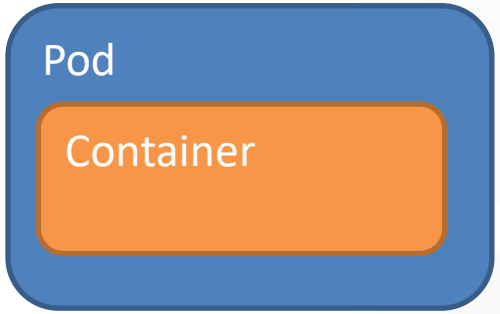
Default Namespaces

- **default:** The default namespace for any object without a namespace.
- **kube-system:** Acts as the home for objects and resources created by Kubernetes itself.
- **kube-public:** A special namespace; readable by all users that is reserved for cluster bootstrapping and configuration.

Pod

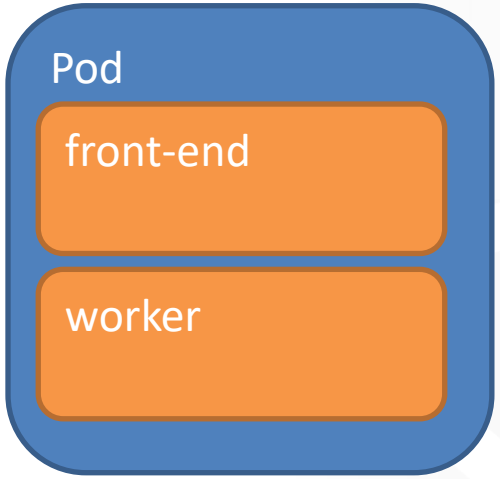
- A Pod is the most basic, atomically deployable unit in Kubernetes
- A Pod is ephemeral – once terminated, it will be reclaimed
- A Pod consists of one or many co-located containers
- A Pod represent a single instance of an application
- The Containers in a Pod share a loopback network interface, and can share mounted folders
- Containers in a Pod cannot use the same network port within the Pod
- Each Pod has it's own, uniquely assigned internal IP address

Pod




```yaml
# file: pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: time
spec:
  containers:
  - name: front-end
    image: nginx
```

Pod

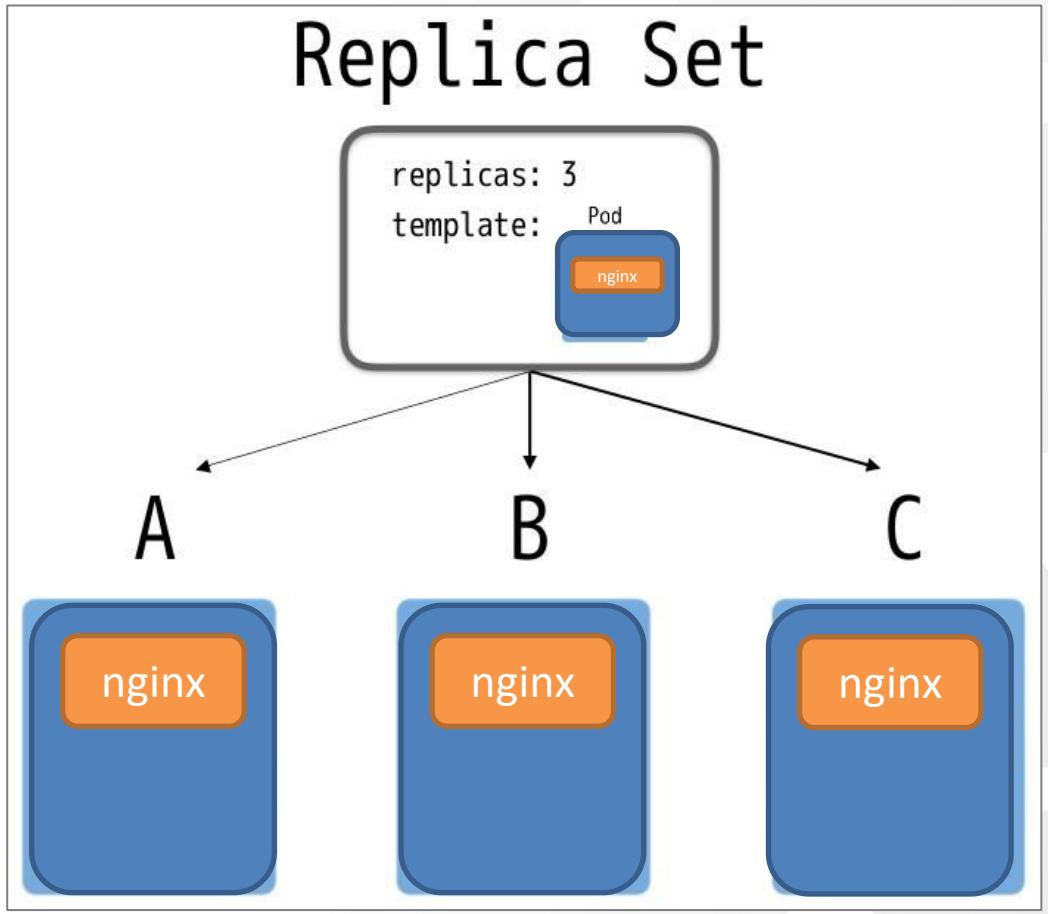



```yaml
# file: pod-with-multiple-containers.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: time
spec:
  volumes:
  - name: html
    emptyDir: {}
  containers:
  - name: front-end
    image: nginx
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
  - name: worker
    image: debian
    volumeMounts:
    - name: html
      mountPath: /html
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          date >> /html/index.html;
          sleep 1;
        done
```

ReplicaSet

- ReplicaSet ensures that a specified number of Pod replicas are running at any given time.
- If there are more Pods than defined in the ‘replicas’ field – The ReplicaSet kills them
- If there are less Pods than defined in the ‘replicas’ field – The ReplicaSet creates them

ReplicaSet




```yaml
# ReplicaSet.yaml
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

Questions

Lab 02: Creating and scaling your first pod

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-02

Noam Amrani

Module 05: Deployments and Upgrades

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

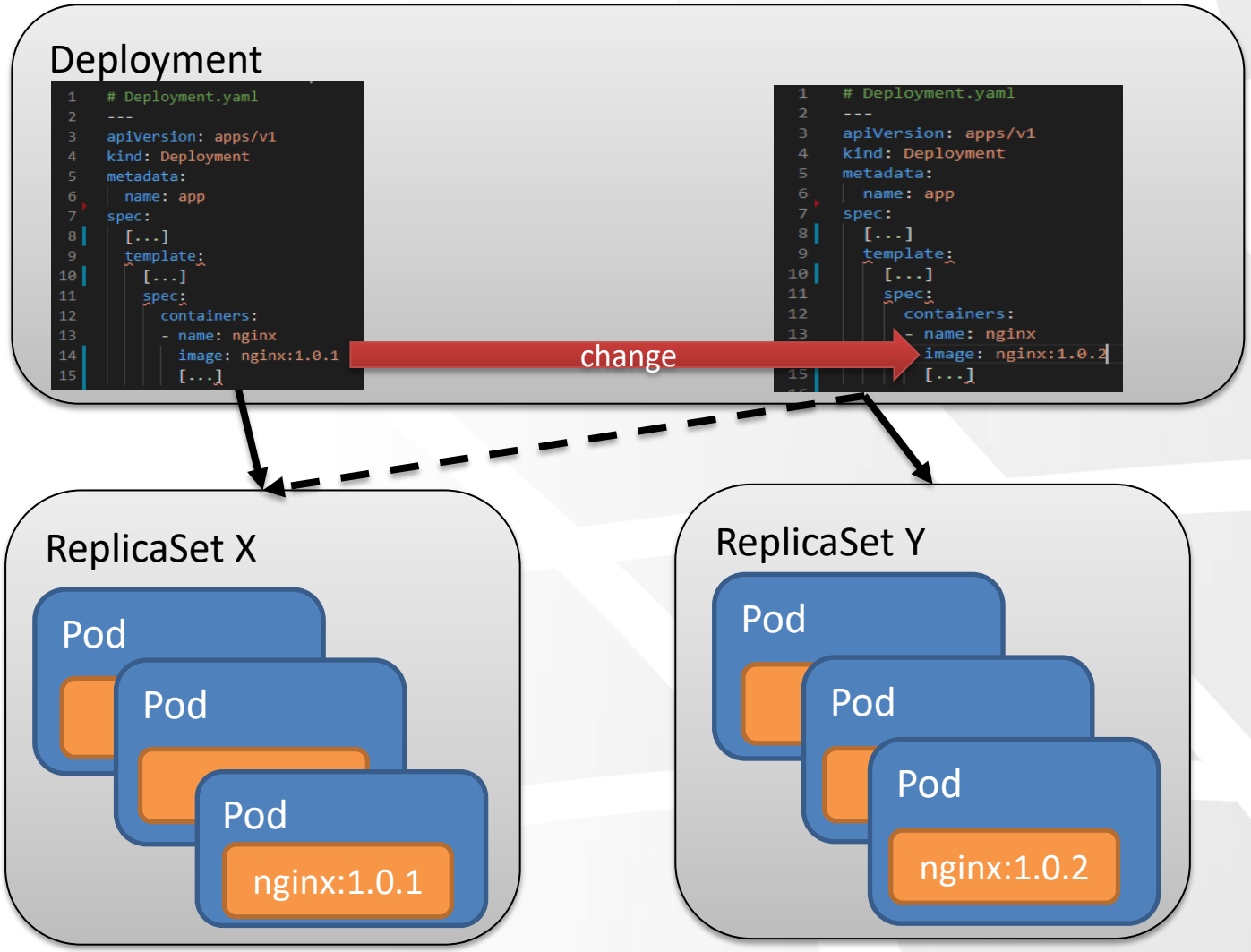
Agenda

- Deployments
- Rolling Upgrades
- Lab 03: Deploy and Upgrade a Single Service

Deployment

- Deployments hold ReplicaSets (one for each version) and adds Upgrade Strategy
- When a Deployment is updated, Kubernetes will perform a rolling upgrade of the Pods running on the cluster.
- Rolling Upgrades actually creates new ReplicaSets, with the updated container image, continually replacing replicas from the old ReplicaSet.
- After Rolling Upgrade is done, the old ReplicaSet kept alive (with zero replicas) for easy rollback.

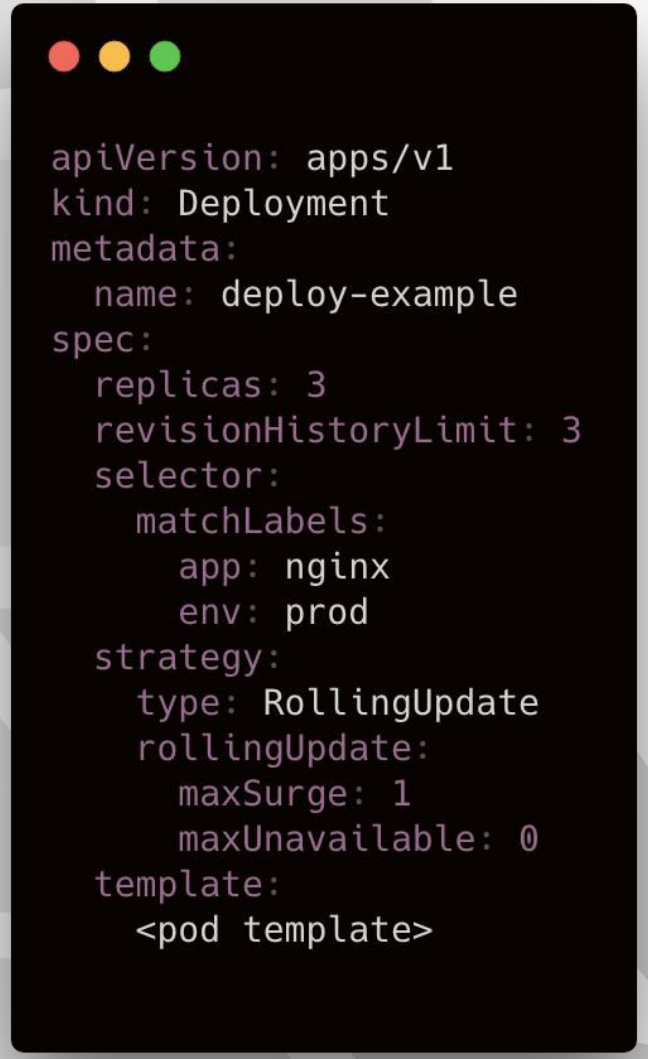
Deployment




```yaml
# Deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  annotations:
    description: "my app"
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx # Matching the labels of the
                 #  template (line 18)
  template:
    metadata:
      labels:
        app: nginx # Matching the matchLabels
                   #  of the deployment.spec.selector
                   #  (line 13)
    spec:
      containers:
      - name: nginx
        image: nginx:1.0.1
        ports:
        - containerPort: 80
```

Deployment

- **revisionHistoryLimit:** The number of previous iterations of the Deployment to retain.
- **strategy:** Describes the method of updating the Pods based on the ***type***. Valid options are **Recreate** or **RollingUpdate**.
  - **Recreate:** All existing Pods are killed before the new ones are created.
  - **RollingUpdate:** Cycles through updating the Pods according to the parameters: ***maxSurge*** and ***maxUnavailable***.

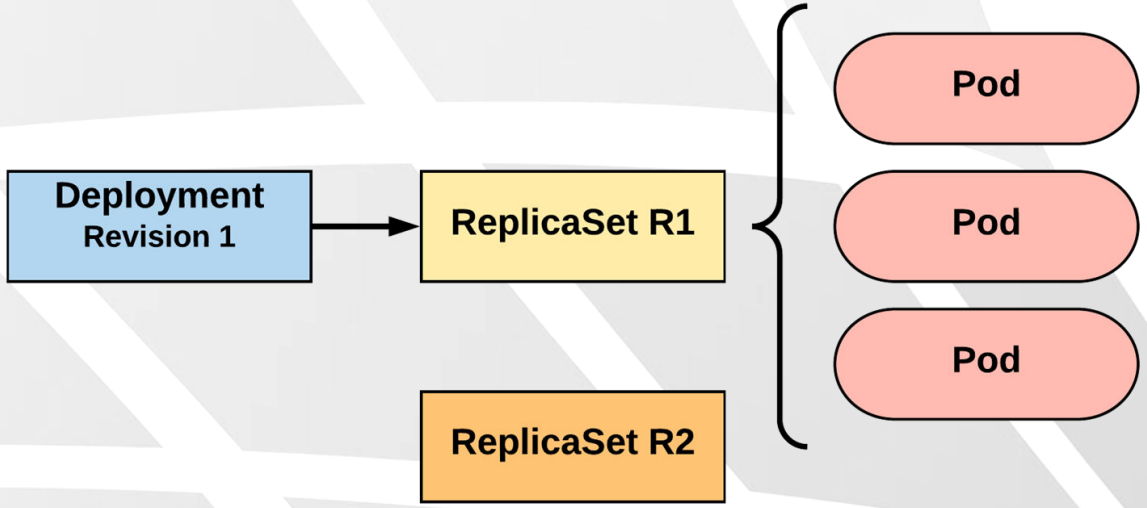


```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: deploy-example
spec:
  replicas: 3
  revisionHistoryLimit: 3
  selector:
    matchLabels:
      app: nginx
      env: prod
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    <pod template>
```

RollingUpdate Deployment

- Updating pod template generates a new ReplicaSet revision.

R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d




```
$ kubectl get replicaset
NAME                DESIRED   CURRENT   READY   AGE
mydep-6766777fff    3         3         3       5h
```

```
$ kubectl get pods
NAME                      READY   STATUS    RESTARTS   AGE
mydep-6766777fff-9r2zn    1/1     Running   0          5h
mydep-6766777fff-hsfz9    1/1     Running   0          5h
mydep-6766777fff-sjxhf    1/1     Running   0          5h
```

RollingUpdate Deployment

- New **ReplicaSet** is initially scaled up based on ***maxSurge***

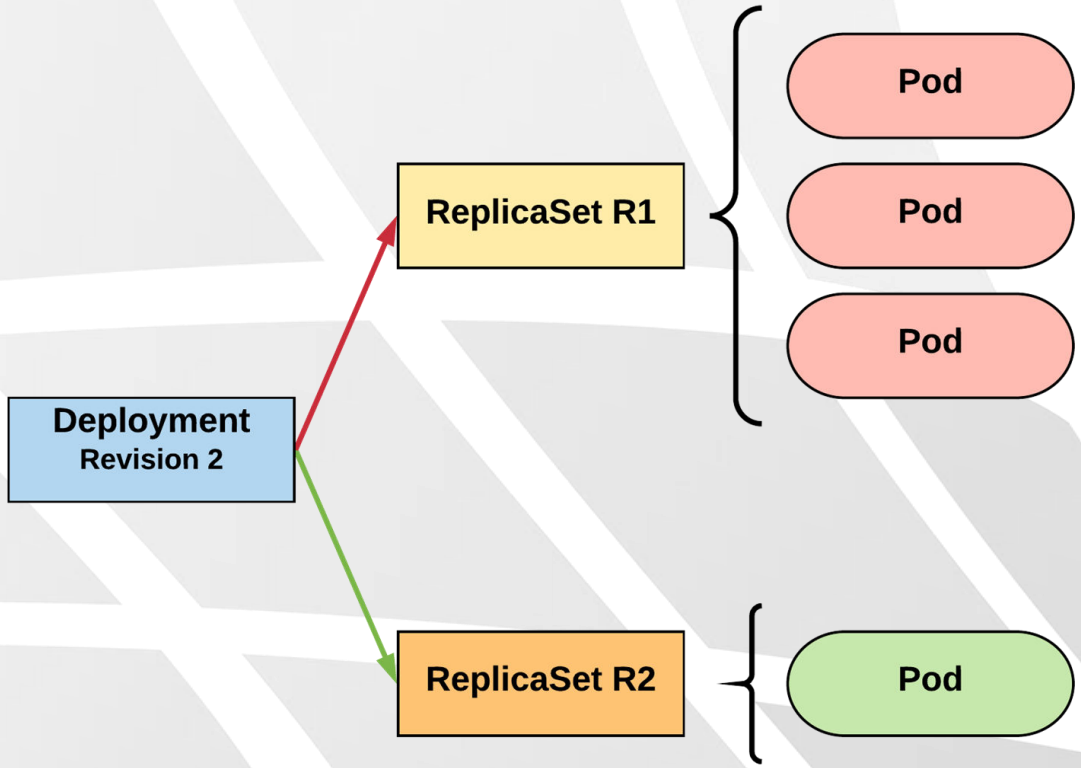
R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d

$ kubectl get replicaset

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	**1**	**1**	**1**	**5s**
mydep-6766777fff	2	3	3	5h

$ kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	**1/1**	**Running**	**0**	**2s**
mydep-6766777fff-9r2zn	1/1	Running	0	5h
mydep-6766777fff-hsfz9	1/1	Running	0	5h
mydep-6766777fff-sjxhf	1/1	Running	0	5h



RollingUpdate Deployment

- Phase out of old Pods managed by ***maxSurge*** and ***maxUnavailable***.

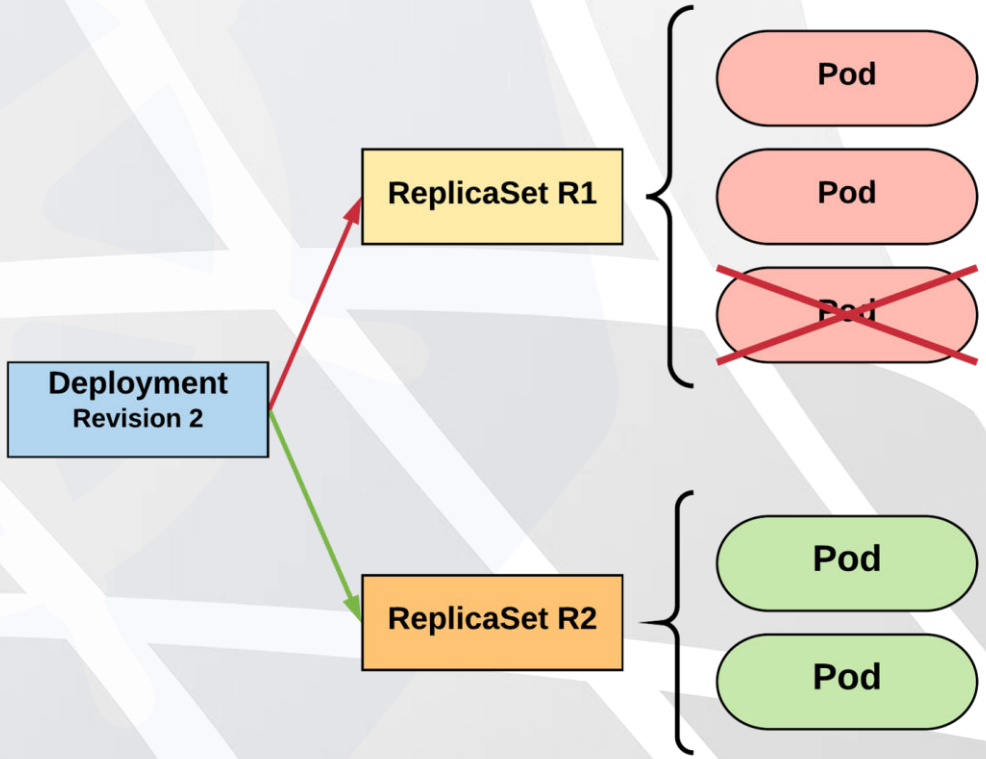
R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d

$ kubectl get replicaset

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	**2**	**2**	**2**	**8s**
mydep-6766777fff	2	2	2	5h

$ kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	**1/1**	**Running**	**0**	**5s**
mydep-54f7ff7d6d-cqvlq	**1/1**	**Running**	**0**	**2s**
mydep-6766777fff-9r2zn	1/1	Running	0	5h
mydep-6766777fff-hsfz9	1/1	Running	0	5h



RollingUpdate Deployment

- Phase out of old Pods managed by ***maxSurge*** and ***maxUnavailable***

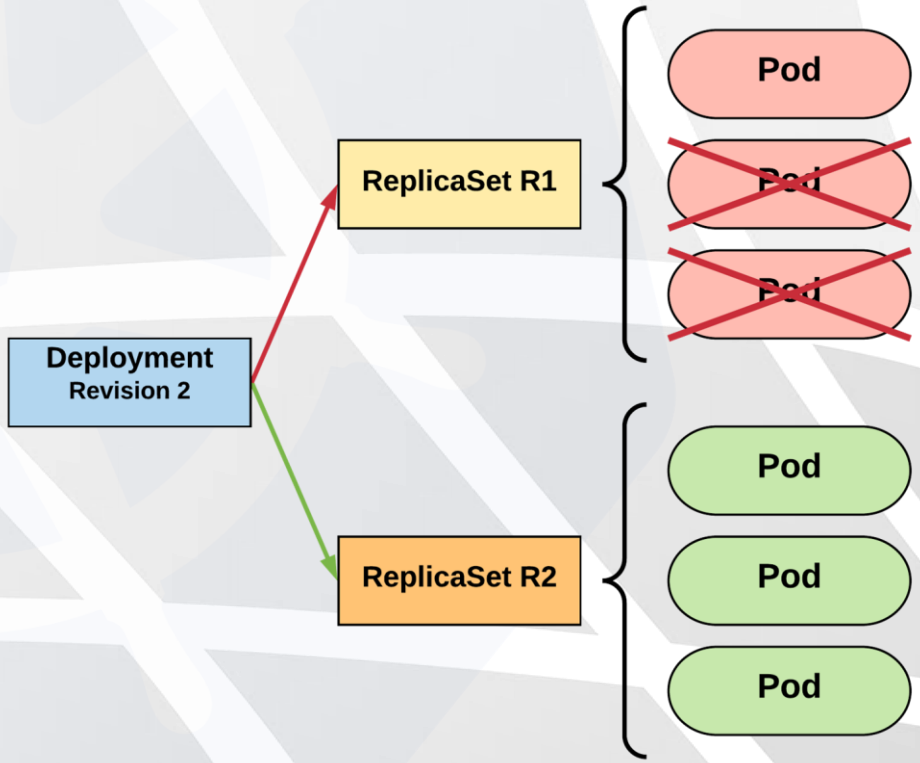
R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d

$ kubectl get replicaset

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	**3**	**3**	**3**	**10s**
mydep-6766777fff	0	1	1	5h

$ kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	**1/1**	**Running**	**0**	**7s**
mydep-54f7ff7d6d-cqvlq	**1/1**	**Running**	**0**	**5s**
mydep-54f7ff7d6d-gccr6	**1/1**	**Running**	**0**	**2s**
mydep-6766777fff-9r2zn	1/1	Running	0	5h



RollingUpdate Deployment

- Phase out of old Pods managed by ***maxSurge*** and ***maxUnavailable***

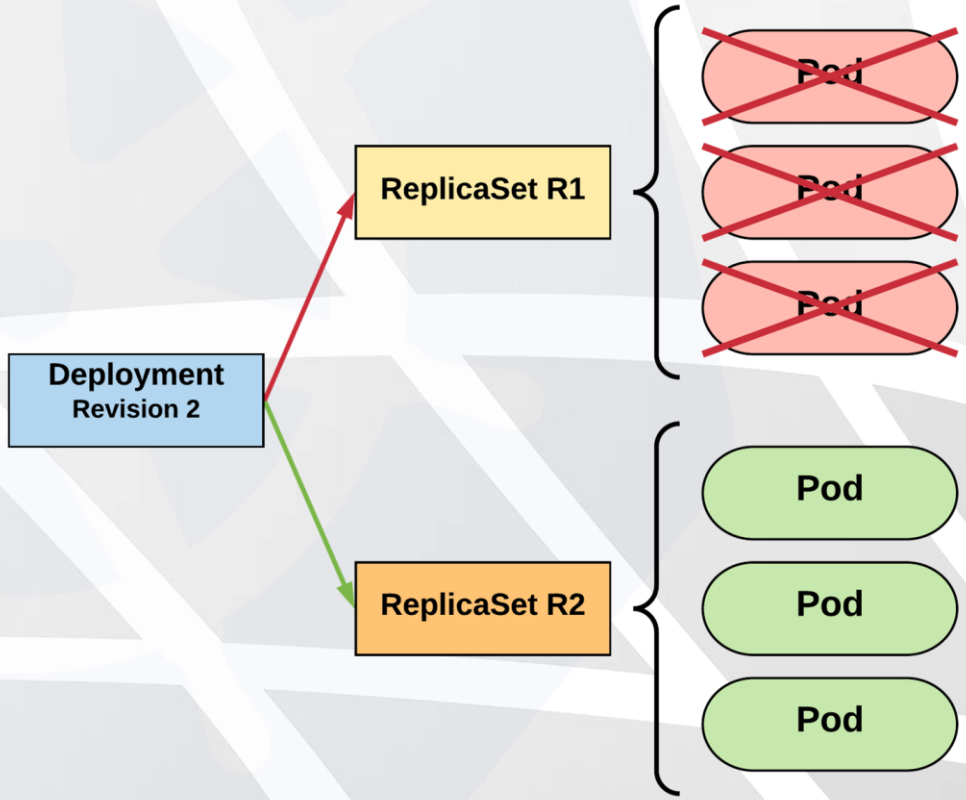
R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d

$ kubectl get replicaset

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	**3**	**3**	**3**	**13s**
mydep-6766777fff	0	0	0	5h

$ kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	**1/1**	**Running**	**0**	**10s**
mydep-54f7ff7d6d-cqvlq	**1/1**	**Running**	**0**	**8s**
mydep-54f7ff7d6d-gccr6	**1/1**	**Running**	**0**	**5s**



RollingUpdate Deployment

- Updated to new deployment revision completed.

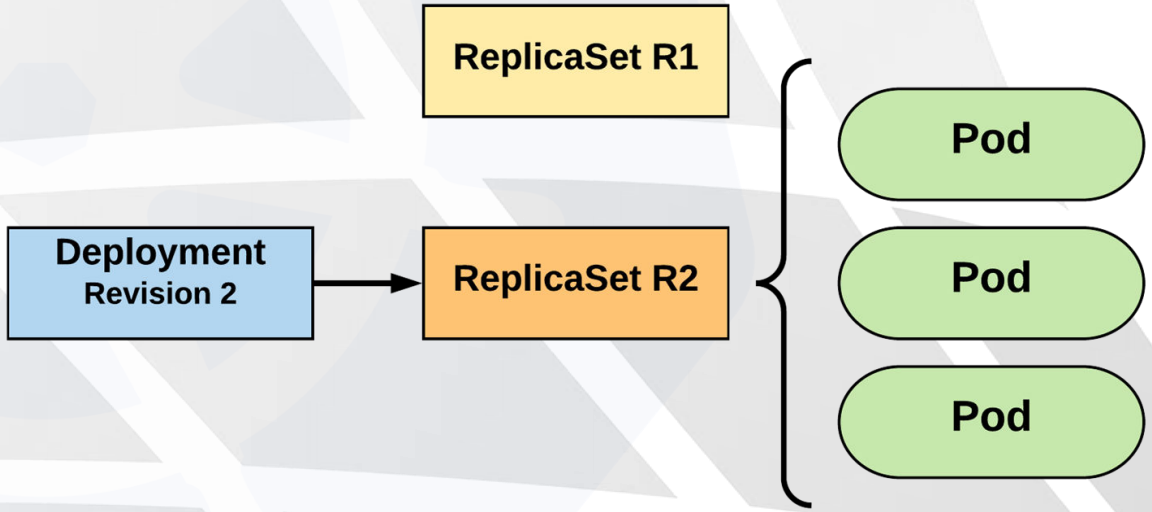
R1 pod-template-hash:
676677fff
R2 pod-template-hash:
54f7ff7d6d

$ kubectl get replicaset

NAME	DESIRED	CURRENT	READY	AGE
mydep-54f7ff7d6d	**3**	**3**	**3**	**15s**
mydep-6766777fff	0	0	0	5h

$ kubectl get pods

NAME	READY	STATUS	RESTARTS	AGE
mydep-54f7ff7d6d-9gvll	**1/1**	**Running**	**0**	**12s**
mydep-54f7ff7d6d-cqvlq	**1/1**	**Running**	**0**	**10s**
mydep-54f7ff7d6d-gccr6	**1/1**	**Running**	**0**	**7s**



Questions

Lab 03: Deploy and Upgrade a Single Service

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-03

Noam Amrani

Module 06: Labels, Selectors & Annotations

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

- Labels & Selectors
- Annotations

Labels

- A label is a semantic tag that can be attached to Kubernetes objects to mark them as a part of a group.

- Services use labels to understand the backend pods they should route requests to.

Label Example

```yaml
# Deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  labels:
    app: nginx
```

```yaml
# ReplicaSet.yaml
---
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  # modify replicas according to your case
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
```

Selectors

- ***Selectors*** use labels to filter or select objects, and are used throughout Kubernetes.

```
selector:
  matchLabels:
    gpu: nvidia
```

- ***Equality based*** selectors allow for simple filtering (=,==, or !=)

```
selector:
  matchExpressions:
    - key: gpu
      operator: in
      values: [“nvidia”]
```

- ***Set-based*** selectors are supported on a limited subset of objects. However, they provide a method of filtering on a set of values, and supports multiple operators including: in, notin, and exist.

Selector Example

```yaml
# Deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx # Matching the labels of the
                 #  template (line 18)
  template:
    metadata:
      labels:
        app: nginx # Matching the matchLabels
                   #  of the deployment.spec.selector
                   #  (line 13)
    spec:
      containers:
      - name: nginx
        image: nginx
        ports:
        - containerPort: 80
```

```yaml
# nodeSelector.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx
  labels:
    env: test
spec:
  containers:
  - name: nginx
    image: nginx
    imagePullPoli     IfNotPresent
  nodeSelector:
    disktype: ssd
    # The pod will be scheduled only on
    #  nodes with label 'disktype: ssd'
```

Annotations

- Annotations are a similar mechanism that allows you to attach arbitrary key-value information to an object.

- Annotations are a way of adding rich metadata to an object that enables extending the object functionality (as oppose to selection purposes)

Annotation Example

```yaml
# Deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: app
  annotations:
    description: "my app"
  labels:
    app: nginx
[...]
```

```yaml
# ingress.yaml
---
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
[...]
```

```yaml
# Service.yaml
---
apiVersion: v1
kind: Service
metadata:
    name: my-service
    annotations:
        cloud.google.com/load-balancer-type: "Internal"
        prometheus.io/path: /healthz
[...]
```

Questions

Noam Amrani

Module 07: Kubernetes Networking

Kubernetes Workshop

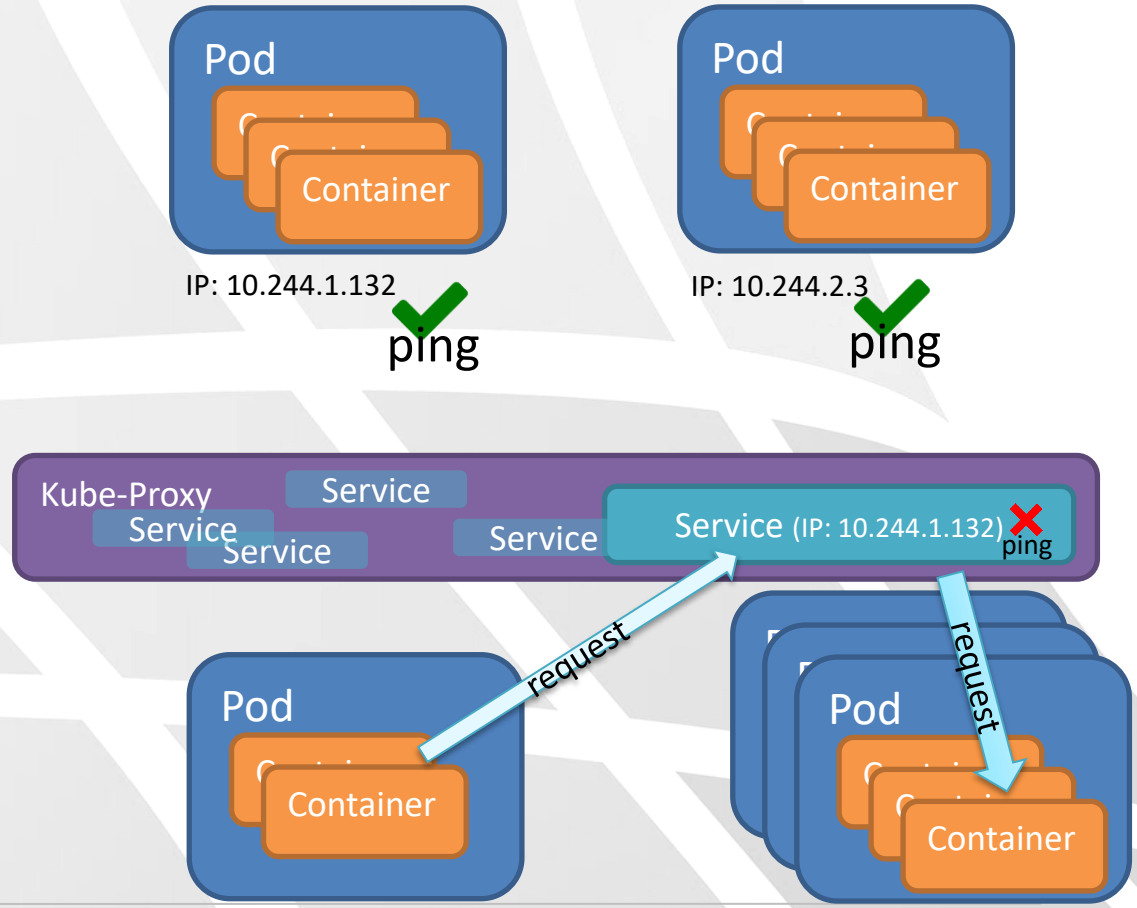
https://www.linkedin.com/in/noamamrani

Agenda

- Kubernetes Networking Introduction
- Container Network Interface (CNI)
- CNI Plugins
- Fundamental Networking Rules
- Networking patterns

Kubernetes Networking

- **Pod Network**
  - Cluster-wide network used for pod-to-pod communication managed by a CNI (Container Network Interface) plugin.
- **Service Network**
  - Cluster-wide range of Virtual IPs managed by kube-proxy for service discovery.

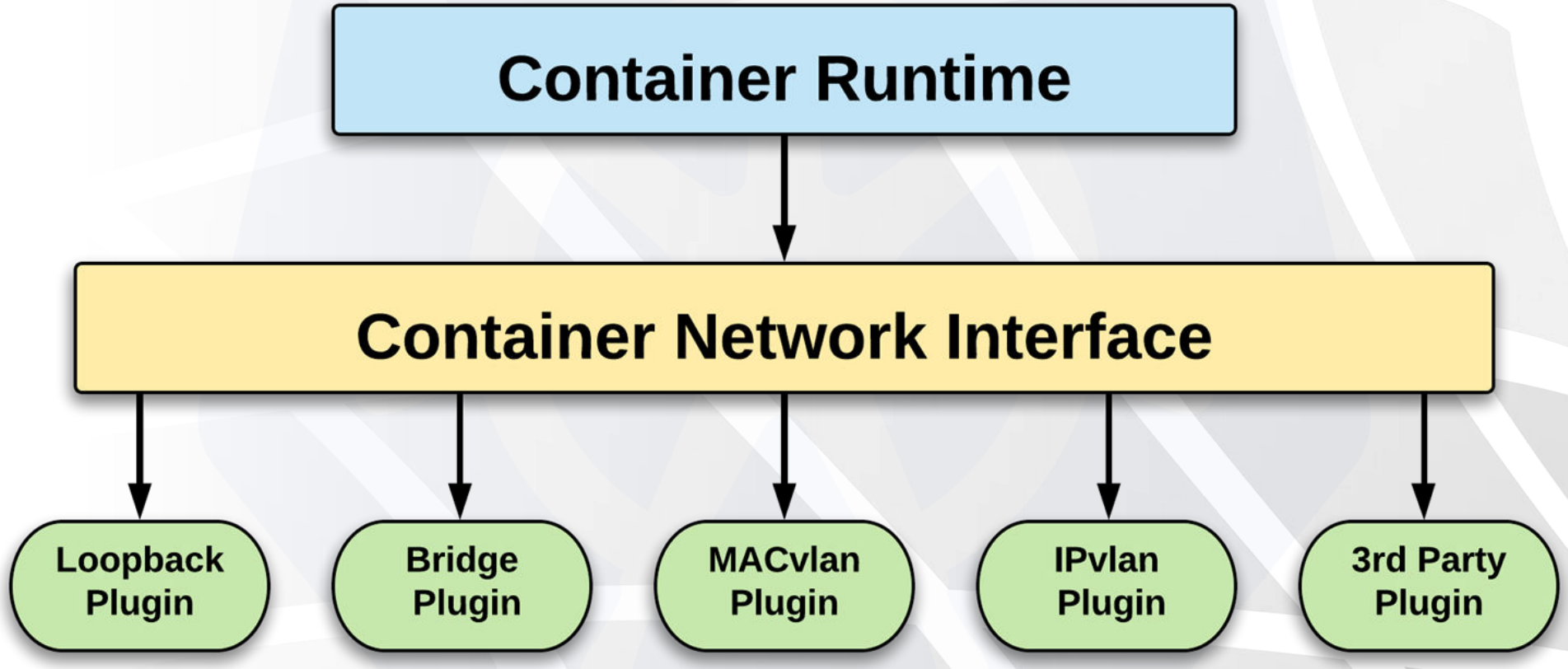


Container Network Interface (CNI)

- Pod networking within Kubernetes is plumbed via the Container Network Interface (CNI).
- Functions as an interface between the container runtime and a network implementation plugin.
- CNCF Project
- Uses a simple JSON Schema.

CNI

CNI Overview



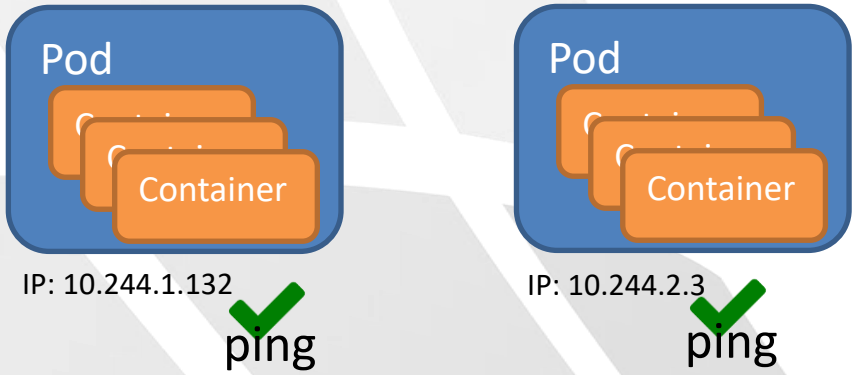
CNI Plugins

- Kubenet
- Flannel
- Calico
- Canal

- Weave
- Amazon ECS
- GCE

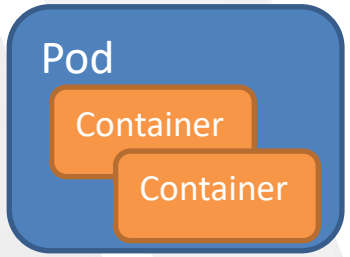
Fundamental Networking Rules

- All containers within a pod can communicate with each other.
- All Pods can communicate with all other Pods without NAT.
- All nodes can communicate with all Pods (and vice-versa) without NAT.
- The IP that a Pod sees itself as is the same IP that others see it as.

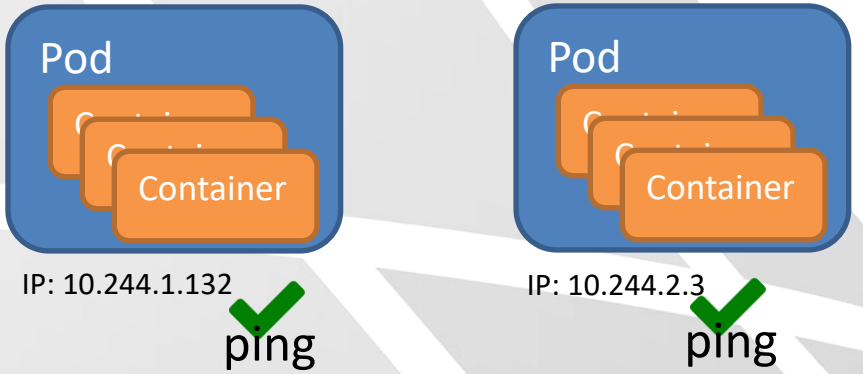


Networking patterns

- **Container-to-Container**
  - Containers within a pod exist within the same network namespace and share an IP.
  - Enables intrapod communication over localhost.

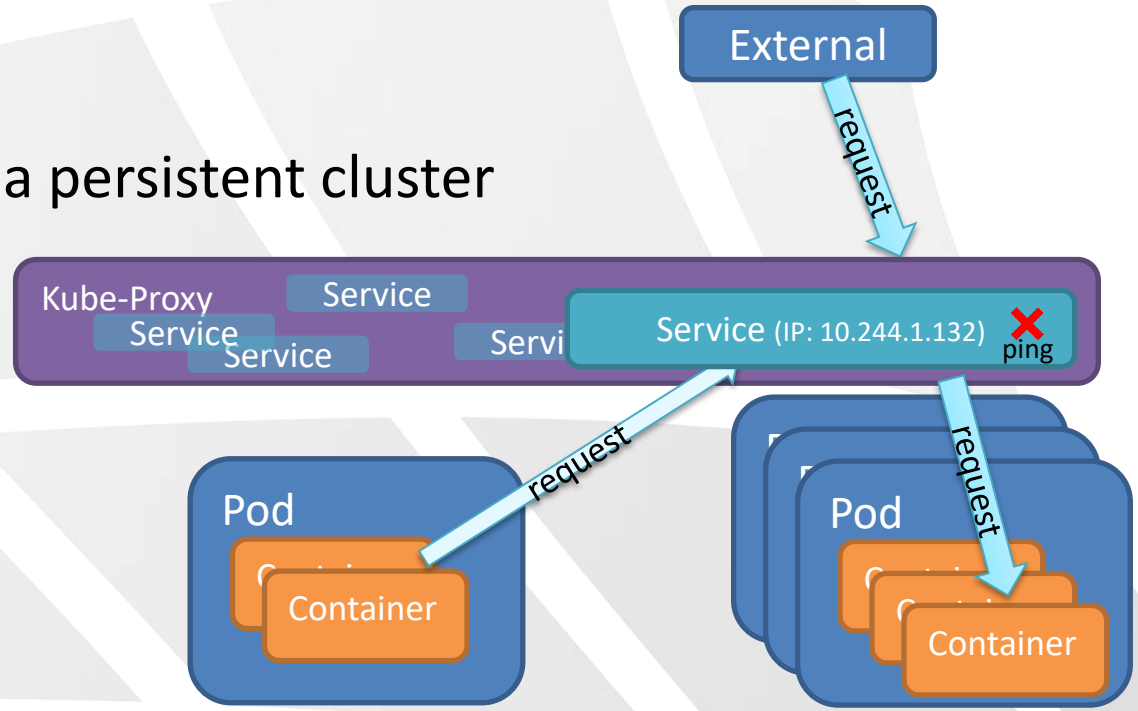


- **Pod-to-Pod**
  - Allocated cluster unique IP for the duration of its life cycle.
  - Pods themselves are fundamentally ephemeral.



Networking patterns

- **Pod-to-Service**
  - Managed by kube-proxy and given a persistent cluster unique IP
  - Exists beyond a Pod's lifecycle.



- **External-to-Service**
  - Handled by kube-proxy.
  - Works in cooperation with a cloud provider or other external entity (LB)

Questions

Noam Amrani

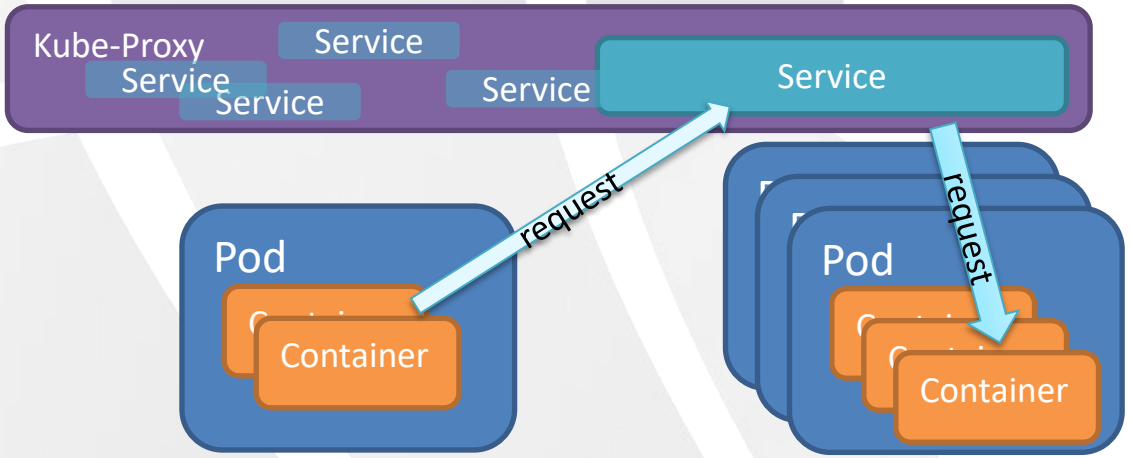
Module 08: Services

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

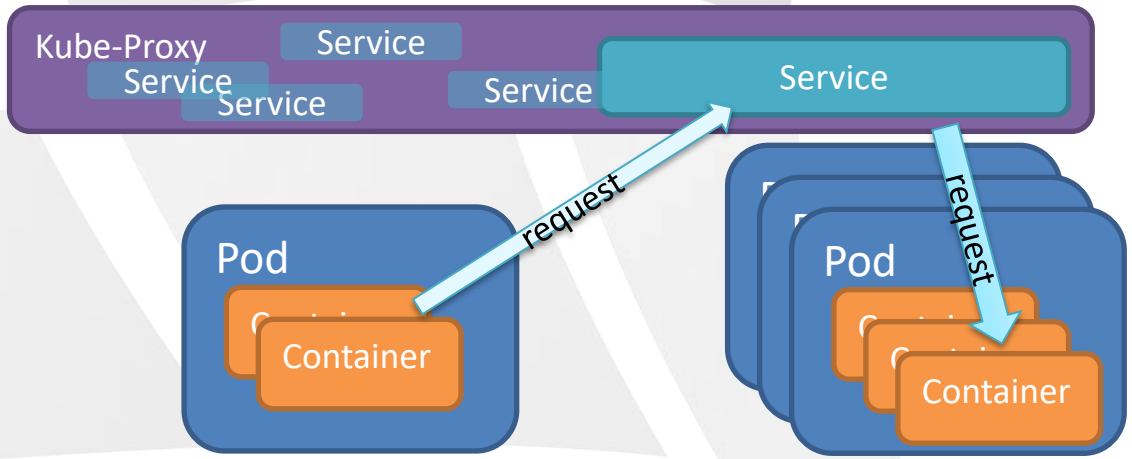
Agenda

- Services Introduction
- Service Types – ClusterIP
- Service Types – NodePort
- Service Types – LoadBalancer
- Service Types – ExternalName
- Lab 04: Creating a Load Balancer Service



Services

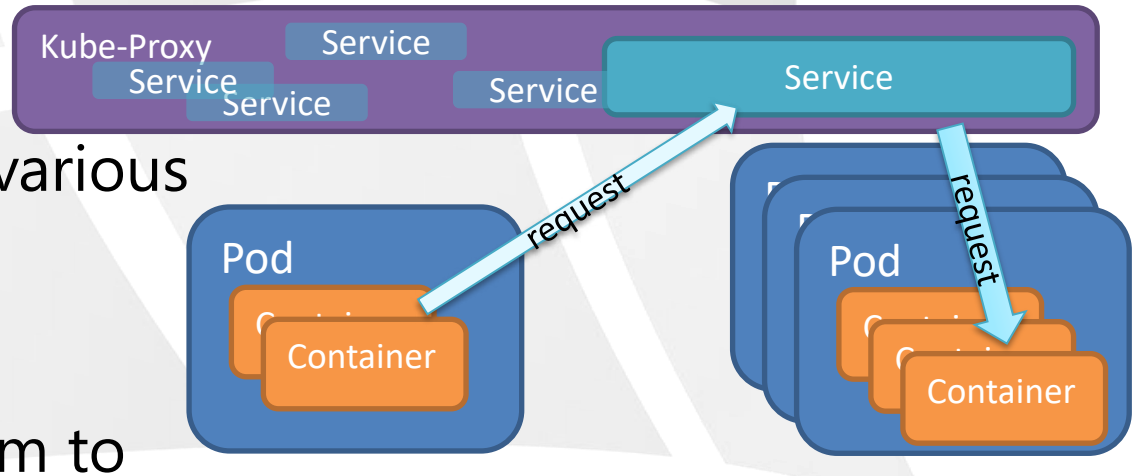
- A Service exposes one or more Pods via a stable, consistent, internal IP address
- It also exposes a hierarchical DNS name for cluster-internal communication
- The Service selects Pods based on a label, key-value selector




```
{service}.{namespace}.svc.cluster.local (i.e nginx.dev.svc.cluster.local)
```

Services

- A Service may expose multiple ports in various protocols – UDP, TCP, HTTP or HTTPS
- A Service is using Round-Robin algorithm to Load Balance traffic to the Pods
- There are several Service Types, depending on the environment we are running the cluster on



Service Types - ClusterIP

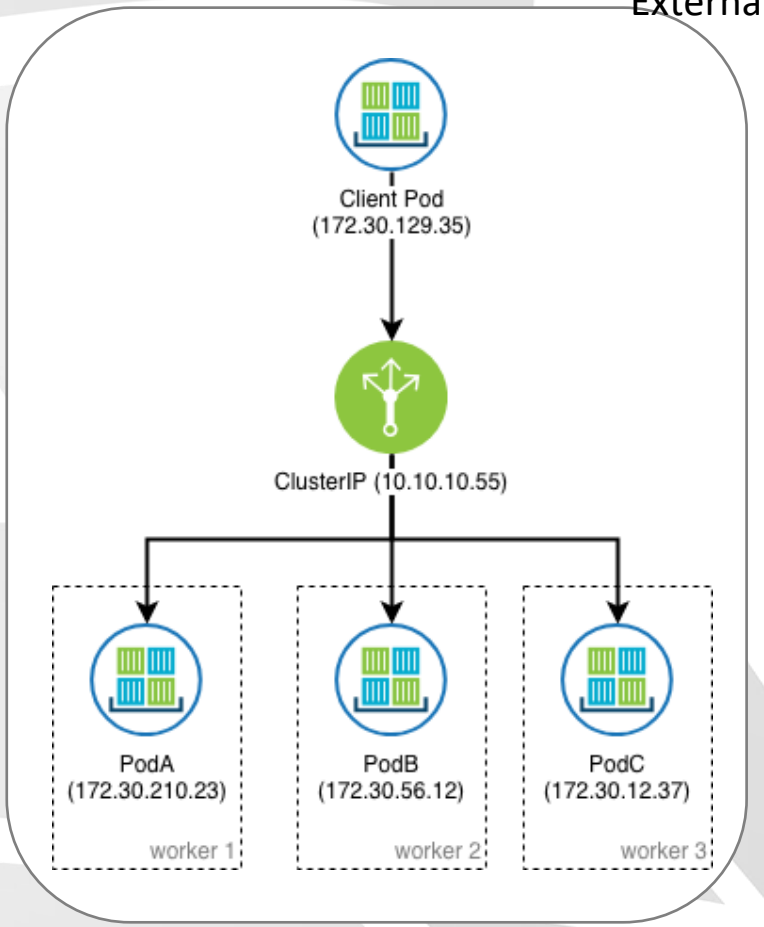
ClusterIP

NodePort

LoadBalancer

ExternalName

- Exposes the service on a cluster-internal IP
- Choosing this value makes the service only reachable from within the cluster
- This is the default ServiceType



ClusterIP

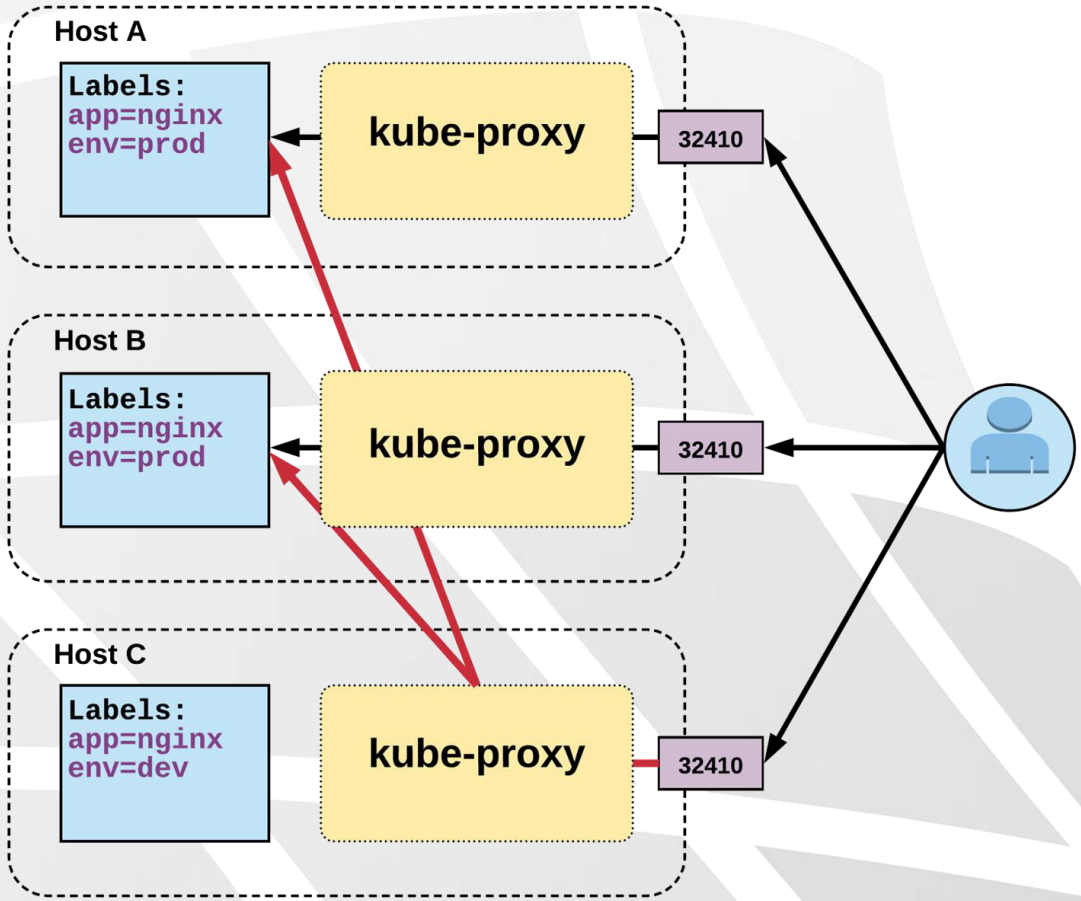
NodePort

LoadBalancer

ExternalName

Service Types - NodePort

- Exposes the service on each Node's IP at a static port (the NodePort)
- A ClusterIP service, to which the NodePort service will route, is automatically created
- Ports being used are between 30000-32767



Service Types - LoadBalancer

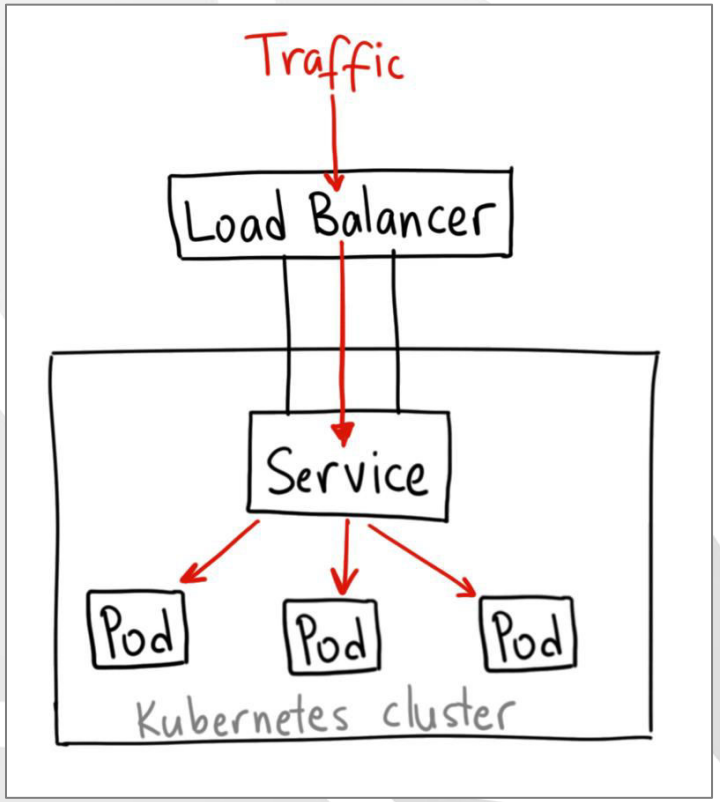
ClusterIP

NodePort

LoadBalancer

ExternalName

- Exposes the service externally using a cloud provider's load balancer
- NodePort and ClusterIP services, to which the external load balancer will route, are automatically created.



Service Types - ExternalName

ClusterIP

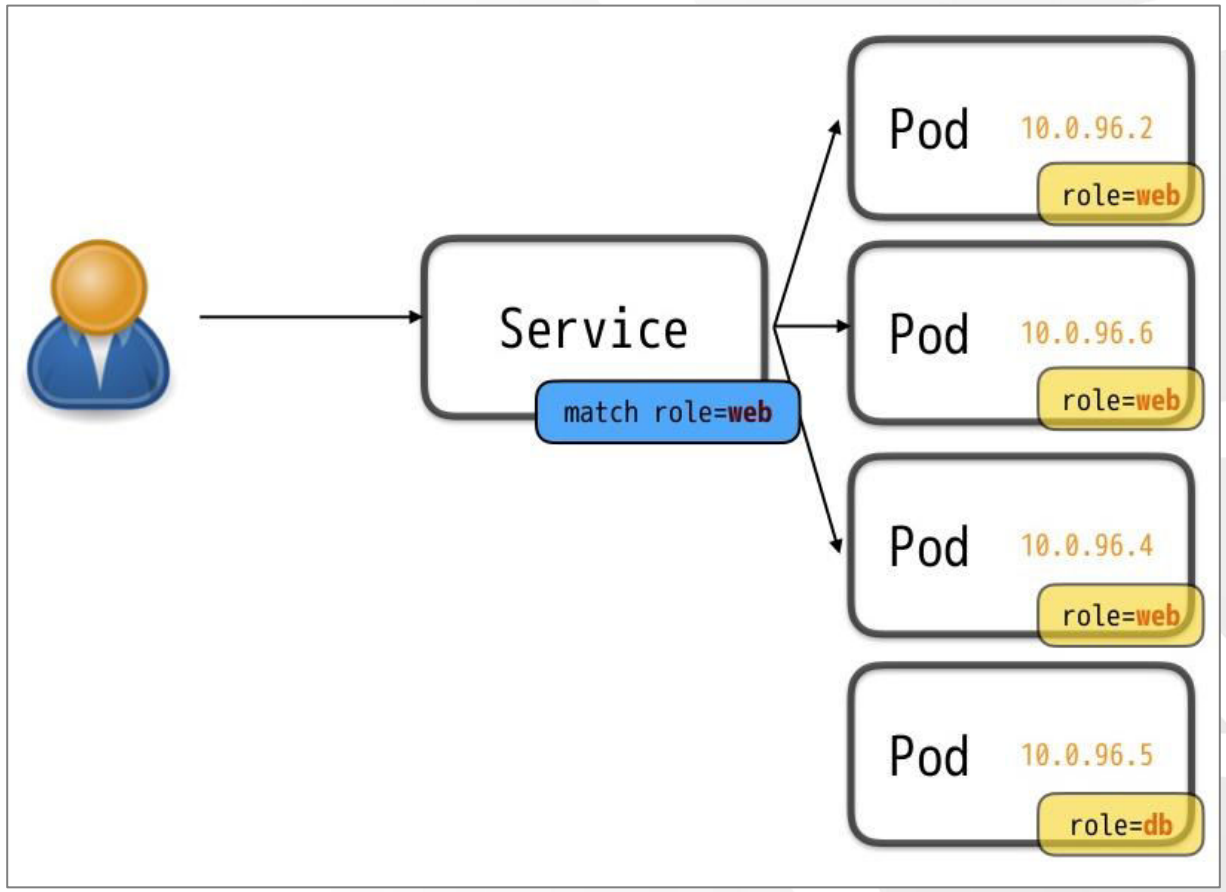
NodePort

LoadBalancer

ExternalName

- Is used to reference endpoints OUTSIDE the cluster
- It creates an internal CNAME DNS entry that aliases another
- Maps the service to the contents of the externalName field (e.g. foo.bar.example.com), by returning a CNAME record with its value

Simple Service



service.yaml

```yaml
# Service.yaml
---
apiVersion: v1
kind: Service
metadata:
    name: my-service
    labels:
        app: nginx-service
spec:
    ports:
    - port: 80
      targetport: 4000
    type: ClusterIP
    selector:
        role: web
```

pod.yaml

```yaml
# file: pod.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: time
  labels:
    role: web
spec:
  containers:
  - name: front-end
    image: nginx
```

Syntax

```yaml
apiVersion: apps/v1
kind:  Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
```

```yaml
apiVersion: v1
kind:  Service
apiVersion: v1
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
  - protocol: TCP
    port: 80
    targetPort: 9376
  loadBalancerIP: 78.11.24.19
  type: LoadBalancer
```

```yaml
apiVersion: apps/v1
kind:  Service
apiVersion: v1
metadata:
  name: my-service
  namespace: prod
spec:
  type: ExternalName
  externalName: my.database.example.com
```

Questions

Lab 04: Creating a Load Balancer Service

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-04

Noam Amrani

Module 09: Ingress

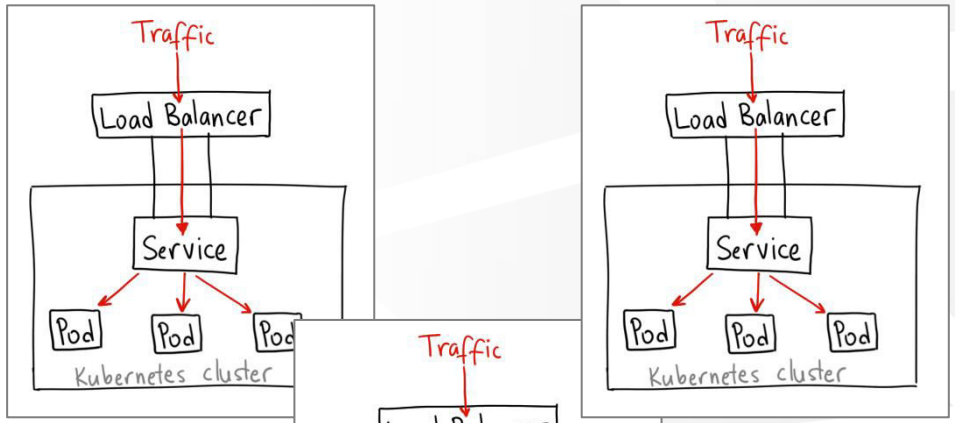
Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

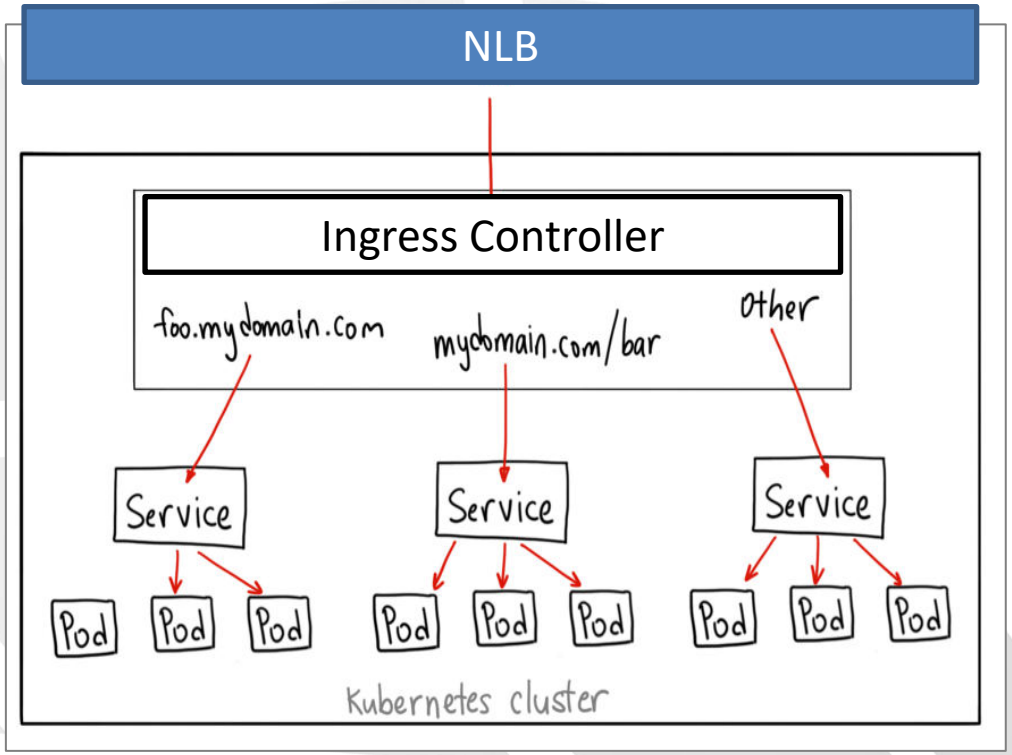
Agenda

- Introduction to Ingress
- Ingress Controllers
- Lab 05: Deploying applications using Ingress

Why Ingress?



VS



Load Balancer Services

Ingress

Ingress

- An API object that manages external access to the services in a cluster, typically HTTP/S
- Ingress can provide L7 load balancing, SSL termination and name-based virtual hosting
- Support installing a Certificate for HTTPS workloads
- Can use ‘Host’ header, or route (/service) for smart routing
- Combined with Namespaces, can be leveraged to create a smart, multi environment, multi-purpose cluster

Ingress Example

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: dev-dispatcher-ingress
  namespace: dev
  annotations:
    # Annotations for Nginx ingress controller:
    ingress.kubernetes.io/ssl-redirect: "false"
    ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - host: devcluster
    http:
      paths:
      - path: /dispatcher
        backend:
          serviceName: dispatcher-service
          servicePort: 80
```

```yaml
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: tls-example-ingress
spec:
  tls:
  - hosts:
    - sslexample.foo.com
    secretName: testsecret-tls
  rules:
    - host: sslexample.foo.com
      http:
        paths:
        - path: /
          backend:
            serviceName: service1
            servicePort: 80
```

Ingress Controller

- In order for the ingress resource to work, the cluster must have an ingress controller running.
- There are many to choose from, and they are mostly compatible with other
- Multiple Ingress Controllers can be deployed on a single cluster
- Ingress Controllers opens ports 80 and 443 (HTTP/S)
- Rules are applied to incoming requests, and if a request is not matched, return 404 (Not Found) using a customizable default http service.

Various Ingress Controllers Available

- Ambassador
- HAProxy
- Kong
- Traefik
- Gloo
- Contour (based on Envoy)
- Nginx (most popular and maintained by Kubernetes Community)
  - Although most popular and wide-support, there are better for specific scenarios
  - A commercial offering also available - Nginx+

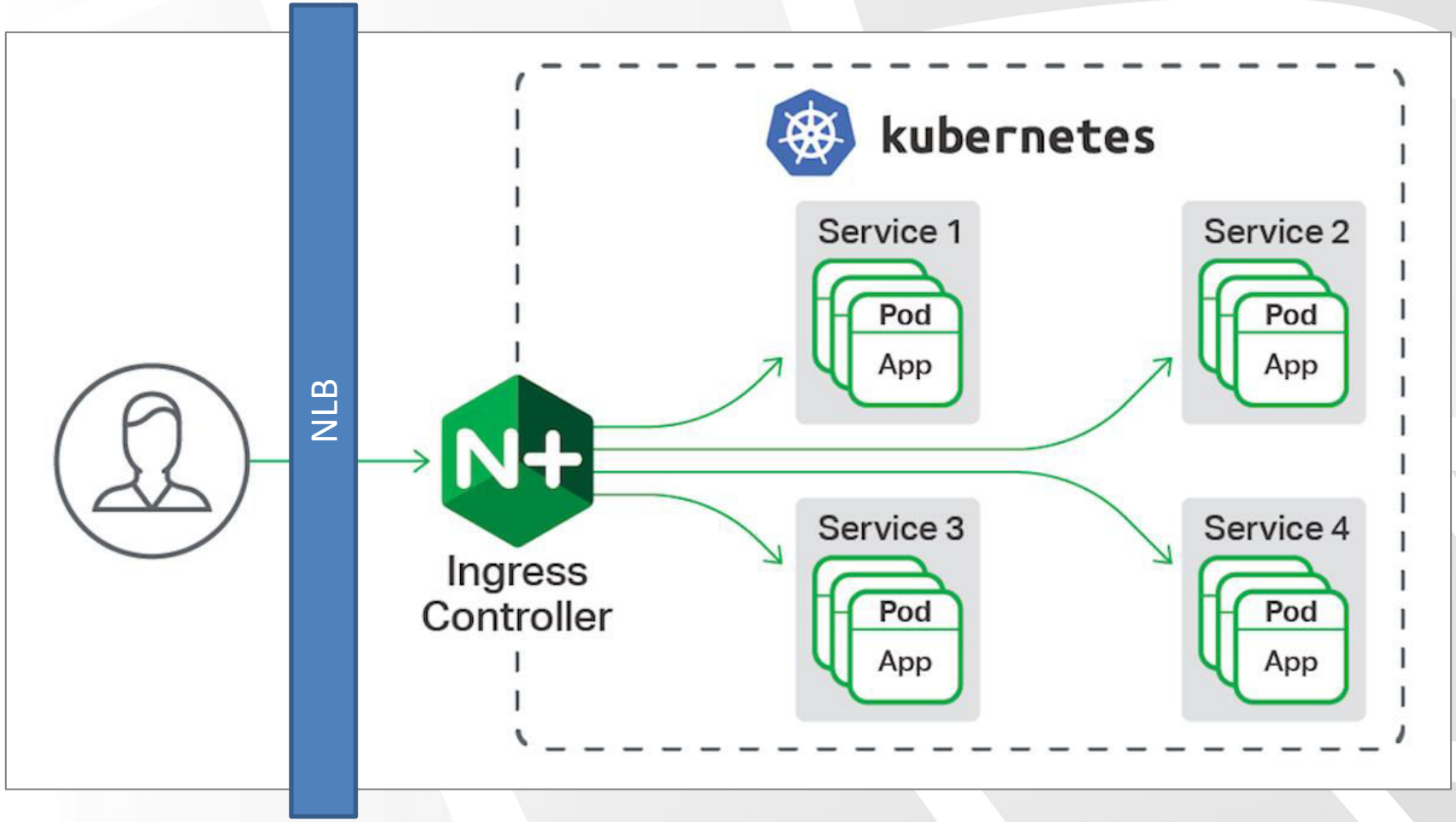
	Kubernetes Ingress	NGINX Ingress	Kong Ingress	Traefik	HAproxy	Voyager	Contour	Istio Ingress	Ambassador	Gloo	Skipper
Protocols	http/https, http2, grpc, tcp/udp (partial)	http/https, http2, grpc, tcp/udp	http/https, http2, grpc, tcp (l4)	http/https, http2 (h2c), grpc, tcp, tcp+tls	http/https, http2, grpc, tcp, tcp+tls	http/https, http2, grpc, tcp, tcp+tls	http/https, http2, grpc, tcp/udp, tcp+tls	http/https, http2, grpc, tcp/udp, tcp+tls, mongo, mysql, redis	http/https, http2, grpc, tcp/udp, tcp+tls	http/https, http2, grpc, tcp, tcp+tls	http/https
Based on	nginx	nginx/nginx plus	nginx	traefik	haproxy	haproxy	envoy	envoy	envoy	envoy	—
Traffic routing	host, path (with regex)	host, path	host, path, method, header*	host (regex), path (regex), headers (regex), query, path prefix, method	host, path	host, path	host, path	host, path, method, header (all with regex)	host, path, method, header (all with regex)	host, path, method, header, query param (all with regex)	host, path, method, header (all with regex)
Namespace limitations	All cluster or specified namespaces	All cluster or specified namespaces	Specified namespace	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces	All cluster or specified namespaces
Traffic distribution	canary, a/b (cookie balancing)	-	canary, acl, blue-green, proxy caching*	canary, blue-green, shadowing	blue-green, shadowing	canary, blue-green, acl	canary, blue-green	canary, a/b, shadowing, http headers, acl, whitelist	canary, a/b, shadowing, http headers, acl, whitelist	canary, shadowing	canary, a/b, blue-green, shadowing, whitelist
Upstream probes	retry, timeouts	retry, active health checks (based on http probe for pod)*	active, circuit breaker	retry, timeouts, active, circuit breaker	check-uri, check-address , check-port	haproxy healthchecks	timeouts, active	retry, timeouts, active checks, circuit breakers	retry, timeouts, active checks, circuit breakers	retry, timeouts, circuit breakers	retry, timeouts, circuit breaker
Load balancing	round-robin, sticky sessions, least-conn, ip-hash, ewma	round-robin, least-conn, least-time, random, sticky sessions*	weighted-round-robin, sticky sessions	weighted-round-robin, dynamic-round-robin, sticky sessions	round-robin, static-rr, leastconn, first, source, uri, url_param, header, sticky sessions	round-robin, static-rr, leastconn, first, source, uri, url_param, header, sticky sessions	round-robin, sticky sessions, weighted-least-request, ring hash, maglev, random	round-robin, sticky sessions, weighted-least-request, ring hash, maglev, random, limit conn, limit req	round-robin, sticky sessions, weighted-least-request, ring hash, maglev, random	round-robin, sticky sessions, least request, random	round-robin, sticky sessions, random
Authentication	Basic, Client cert, external Basic, external OAuth	Basic	Basic, HMAC, Key, LDAP, OAuth 2.0, PASETO, OpenID Connect**	Basic, auth-url, auth-tls, external auth	Basic, OAuth, Auth TLS	Basic, OAuth, auth-tls, OAuth Google, OAuth GitHub	-	mutual tls, OpenID, custom auth	Basic, external auth, OAuth, OpenID	Basic*, external auth*, OAuth*, OpenID*, LDAP*	Basic, OAuth, OpenID
Paid subscription	-	+	+	+	+	+	-	-	+	+	-
GUI	-	+ * **	+ * **	+	-	-	-	-	-	+ *	-
JWT validation	-	+ *	+ **	-	+ **	-	-	+	+ *	+ *	+
Basic DDoS protection	rate limit, limit conn, liimt rps, limit rpm, limit-rate-after, limit-whitelist	rate limit, rate-limit-burst	advanced rate limit*, rate limit, request size limit, request termination, response rate limit	max-conn, rate limit, ip whitelist	limit-rps, limit-connections, limit-whitelist	max-conn, rate limit, whitelist	max-conn, max-request	acl, whitelist, rate limit	rate limit, load shedding	rate limit*	rate limit
Requests tracing	+	-	+	+	-	-	-	+	+	+	+
Config customization	+	+	+	+	+	+	-	+	-	-	+
WAF	lua-resty-waf, ModSecurity	+ *	Wallarm	-	ModSecurity	-	-	ModSecurity	-	ModSecurity*	-
GitHub: stars/commits/releases	5799 4472 54	1950 582 30	667 393 16	25104 3190 264	501 658 63	1109 1219 80	1905 2095 34	19843 8917 88	2246 8355 357	1729 614 171	1870 1582 505
	** In paid version only.*										
	*** Module is available.*										

https://medium.com/flant-com/comparing-ingress-controllers-for-kubernetes-9b397483b46b

https://hsto.org/webt/nj/pq/ea/njpqeaqfbddxy9zhvyidqwazgua.png

Questions

Lab Overview



Lab 05: Deploying applications using Ingress

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-05

Noam Amrani

Module 10: ConfigMaps And Secrets

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

- ConfigMaps
- ConfigMaps – As Environment Variables
- ConfigMaps – As Volumes
- Secrets
- Lab 06: Using ConfigMaps and Secrets

ConfigMaps

- ConfigMaps allow you to decouple configuration artifacts from image content to keep containerized applications portable
- Frequently used to inject configuration files and environment variables into containers
- ConfigMaps can be created by using YAML files, or load it from files

```
kubectl create configmap app-config --from-file=test.config
```

ConfigMaps – Mapped as Environment Variable

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: env-config
  namespace: default
data:
  log_level: INFO
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: dapi-test-pod
spec:
  containers:
    - name: test-container
      image: k8s.gcr.io/busybox
      command: [ "/bin/sh", "-c", "env" ]
      env:
        - name: LOG_LEVEL
          valueFrom:
            configMapKeyRef:
              name: env-config
              key: log_level
```

ConfigMaps – Mapped as a Volume

```yaml
# configmap.yaml
---
kind: ConfigMap
apiVersion: v1
metadata:
  name: demo-app-content
  labels:
    app: demo-app
data:
  index.html: |
    <!doctype html>
    <html>
      <head>
        <meta charset="utf-8">
        <title>Demo App</title>
        <link rel="stylesheet" href="main.css">
      </head>
      <body>
        <h1>Hello from my ConfigMap!</h1>
      </body>
    </html>
  main.css: |
    body {
      background-color: rgb(224,224,224);
      font-family: Verdana, Arial, Helvetica, sans-serif;
      font-size: 100%;
    }
```

```yaml
# file-deployment.yaml
---
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app: demo-app
  name: demo-app
spec:
  selector:
    matchLabels:
      app: demo-app
  replicas: 1
  template:
    metadata:
      labels:
        app: demo-app
      name: demo-app
    spec:
      containers:
        - name: demo-app
          image: selaworkshops/nginx:alpine
          volumeMounts:
            - readOnly: true
              mountPath: /usr/share/nginx/html
              name: html-files
      volumes:
        - name: html-files
          configMap:
            name: demo-app-content
```

Secrets

- Very similar to ConfigMaps primitive, meant to deal with sensitive data
- There are three common types of secrets:
  - TLS - a TLS secret (certificates)
  - Docker-registry - a secret for use with a Docker registry
  - Generic - a secret from a local file, directory or literal value
- As ConfigMaps, can be created from the CLI, or YAML files.
- And can be mounted as Environment Variables or Volumes.

Secrets

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: test-secret
data:
  username: bXktYXBw
  password: Mzk1MjgkdmRnN0pi
```

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: secret-test-pod
spec:
  containers:
    - name: test-container
      image: nginx
      volumeMounts:
          - name: secret-volume
            mountPath: /etc/secret-volume
  volumes:
    - name: secret-volume
      secret:
        secretName: test-secret
```

Secrets

Docker secret

```yaml
# secret.yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: secret-dockercfg
type: kubernetes.io/dockercfg
data:
  .dockercfg: |
    "<base64 encoded ~/.dockercfg file>"
```

TLS secret

```yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: secret-tls
type: kubernetes.io/tls
data:
  # the data is abbreviated in this example
  tls.crt: |
    MIIC2DCCAcCgAwIBAgIBATANBgkqh ...
  tls.key: |
    MIIEpgIBAAKCAQEA7yn3bRHQ5FHMQ ...
```

Generic

```yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: my-secret
type: Opaque
data:
  key: dmFsdWUK
```

ssh auth secret

```yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: secret-ssh-auth
type: kubernetes.io/ssh-auth
data:
  # the data is abbreviated in this example
  ssh-privatekey: |
    MIIEpQIBAAKCAQEAulqb/Y ...
```

Basic auth secret

```yaml
---
apiVersion: v1
kind: Secret
metadata:
  name: secret-basic-auth
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: t0p-Secret
```

Questions

Lab 06: Using ConfigMaps and Secrets

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-06

Noam Amrani

Module 11 - Jobs and CronJobs

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

- Jobs
- CronJobs
- Lab 07: Running Jobs and CronJobs

Jobs and Cron Jobs

- A job creates one or more pods and ensures that a specified number of them successfully terminate
- Jobs are useful if you need to perform one-off or batch processing instead of running a continuous service.
- Queue Processing, and Build Jobs are first class candidate for being a Job
- CronJobs can be used to schedule a job to execute in the future or on a regular, recurring basis
- CronJobs mostly being used for maintenance tasks (re-indexing, generate a nightly report, Cleanups, etc..)

Jobs and Cron Jobs

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: pi
spec:
  template:
    spec:
      containers:
      - name: pi
        image: perl
        command: ["perl",  "-Mbignum=bpi",
                  "-wle", "print bpi(2000)"]
      restartPolicy: Never
  backoffLimit: 4
```

```yaml
apiVersion: batch/v1beta1
kind: CronJob
metadata:
  name: hello
spec:
  schedule: "*/1 * * * *"
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from the Kubernetes cluster
          restartPolicy: OnFailure
```

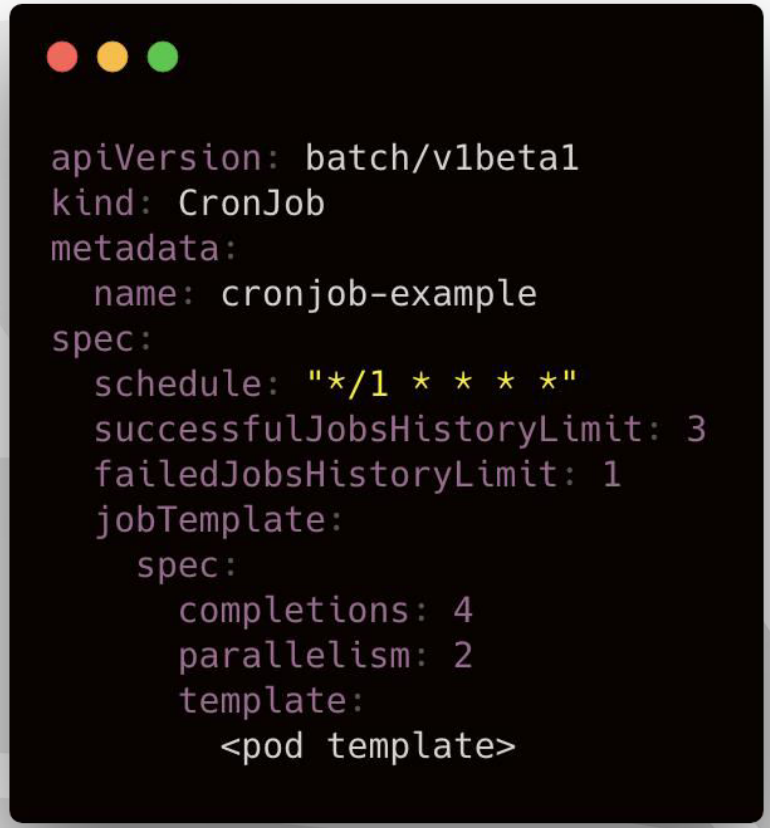
Job - Settings

- **backoffLimit**: The number of failures before the job itself is considered failed.
- **completions**: The total number of successful completions desired.
- **parallelism**: How many instances of the pod can be run concurrently.
- **restartPolicy**:Jobs only support a container's restartPolicy of type Never or OnFailure.

```yaml
apiVersion: batch/v1
kind: Job
metadata:
  name: job-example
spec:
  backoffLimit: 4
  completions: 4
  parallelism: 2
  template:
    spec:
      restartPolicy: Never
    <pod-template>
```

CronJob - Settings

- **schedule**: The cron schedule for the job.
- **successfulJobHistoryLimit**: The number of successful jobs to retain.
- **failedJobHistoryLimit**: The number of failed jobs to retain.



Questions

Lab 07: Running Jobs and CronJobs

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-07

Noam Amrani

Module 12 - Daemonsets

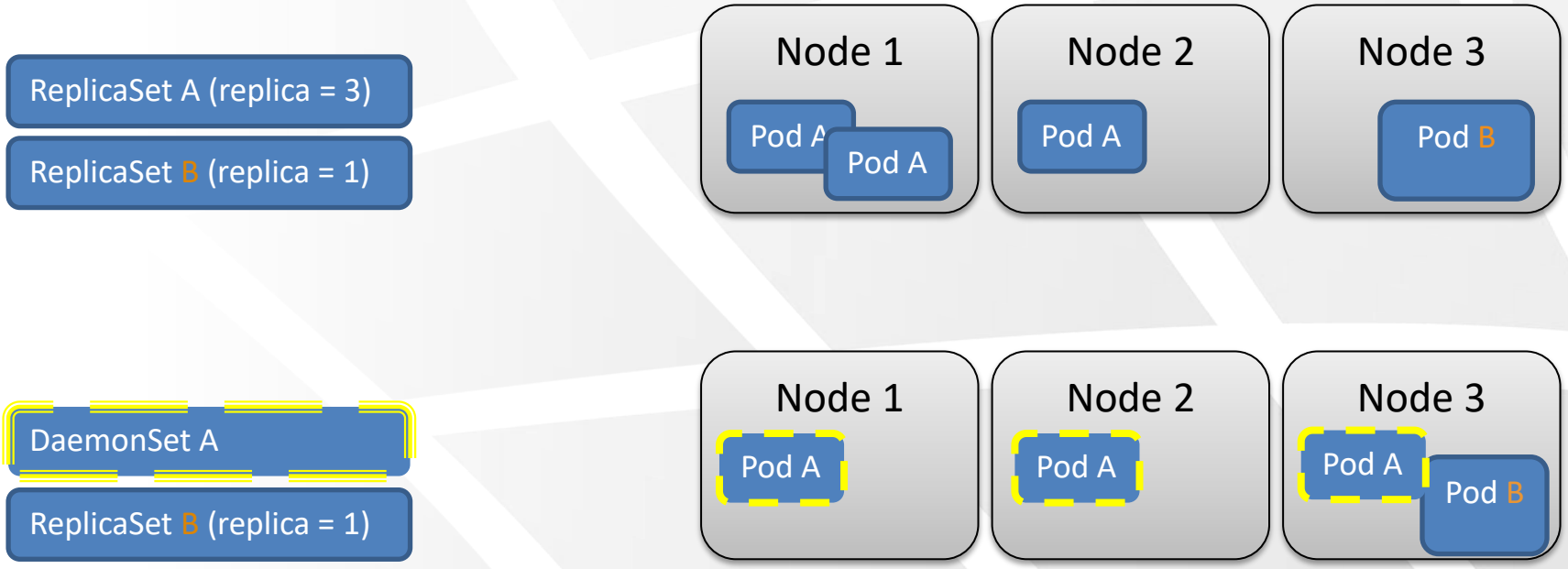
Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

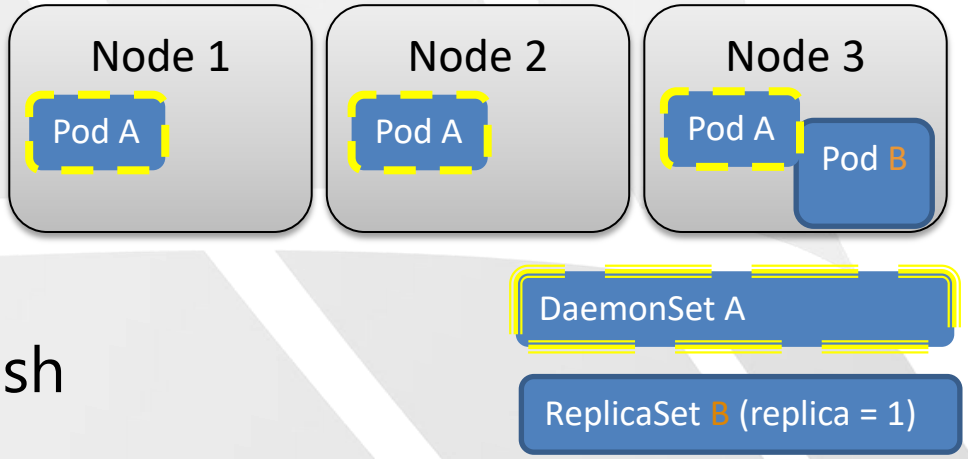
- DaemonSets
- Lab 08: Running Pods as DaemonSets

ReplicaSet vs DaemonSets

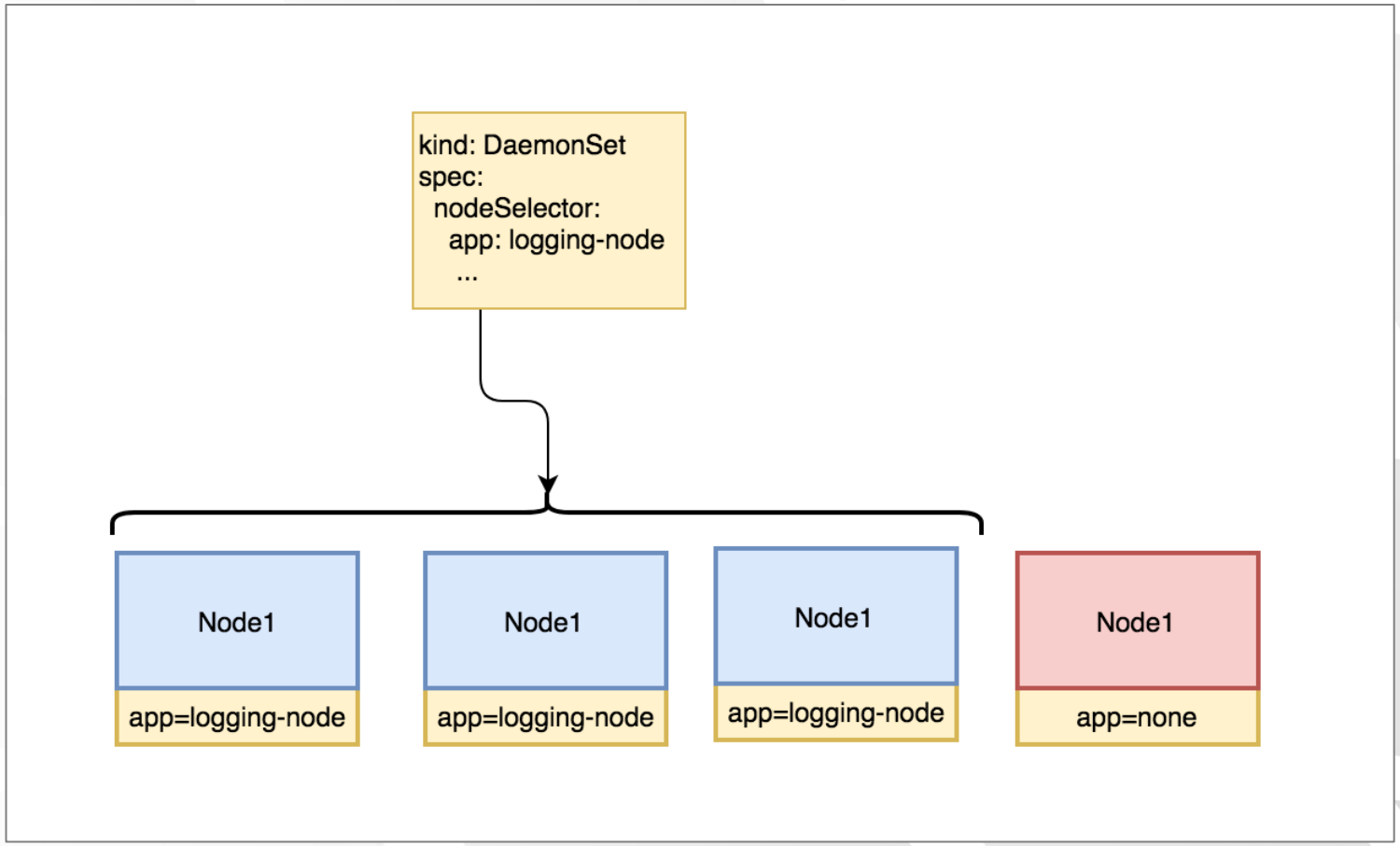


DaemonSets

- A DaemonSet ensures that all (or some) Nodes run a copy of a Pod
- No more than one instance per node
- Very useful for daemon tools
  - Cluster storage daemon
  - Log collection daemon – fluentd, logstash
  - Monitoring daemon
- Kube-proxy is a DaemonSet



DaemonSets



DaemonSets Example

```yaml
# daemonset.yaml
---
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd-elasticsearch
  namespace: kube-system
  labels:
    k8s-app: fluentd-logging
spec:
  selector:
    matchLabels:
      name: fluentd-elasticsearch
  template:
    metadata:
      labels:
        name: fluentd-elasticsearch
    spec:
      containers:
      - name: fluentd-elasticsearch
        image: quay.io/fluentd_elasticsearch/fluentd:v2.5.2
        resources:
          limits:
            memory: 200Mi
          requests:
            cpu: 100m
            memory: 200Mi
```

Questions

Lab 08: Running Pods as DaemonSets

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-08

SELA

Module 13: Helm Package Manager

Kubernetes Workshop

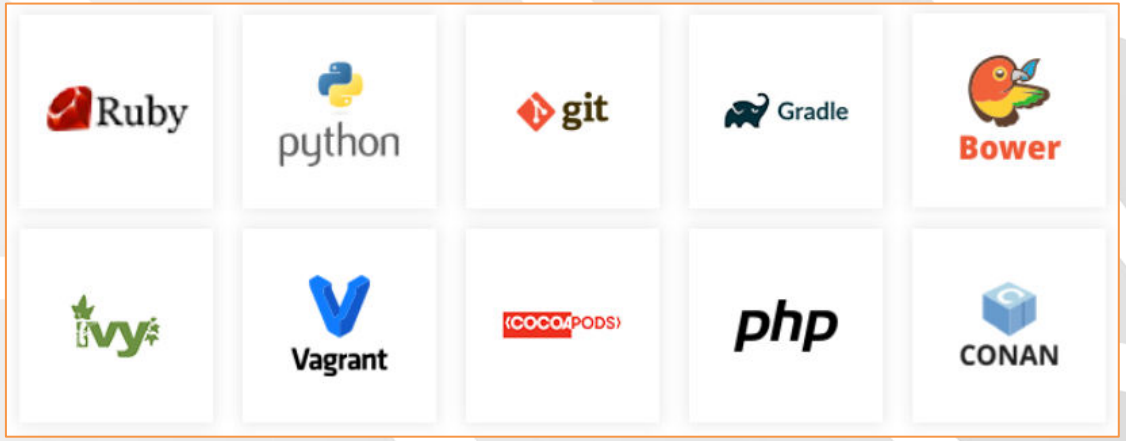
Agenda

- What is a package manager?
- What is Helm?
- Helm Concepts
- Helm Architecture
- Helm CLI
- Using Helm Charts
- Lab 09: Deploying Applications using Helm

What is Package Management?

- Is a collection of software tools that automates the process of installing, upgrading, configuring, and removing computer programs for a computer's operating system in a consistent manner.

Package Management Systems



Artifacts managers

Repository manager is a third party tools that hosting packages of many types in order to make them:

- Maintainable (version, search, tag, etc)
- Highly Available
- Secure
- Accessible (CI/CD, developers, other)

Artifacts managers

The best artifact managers are

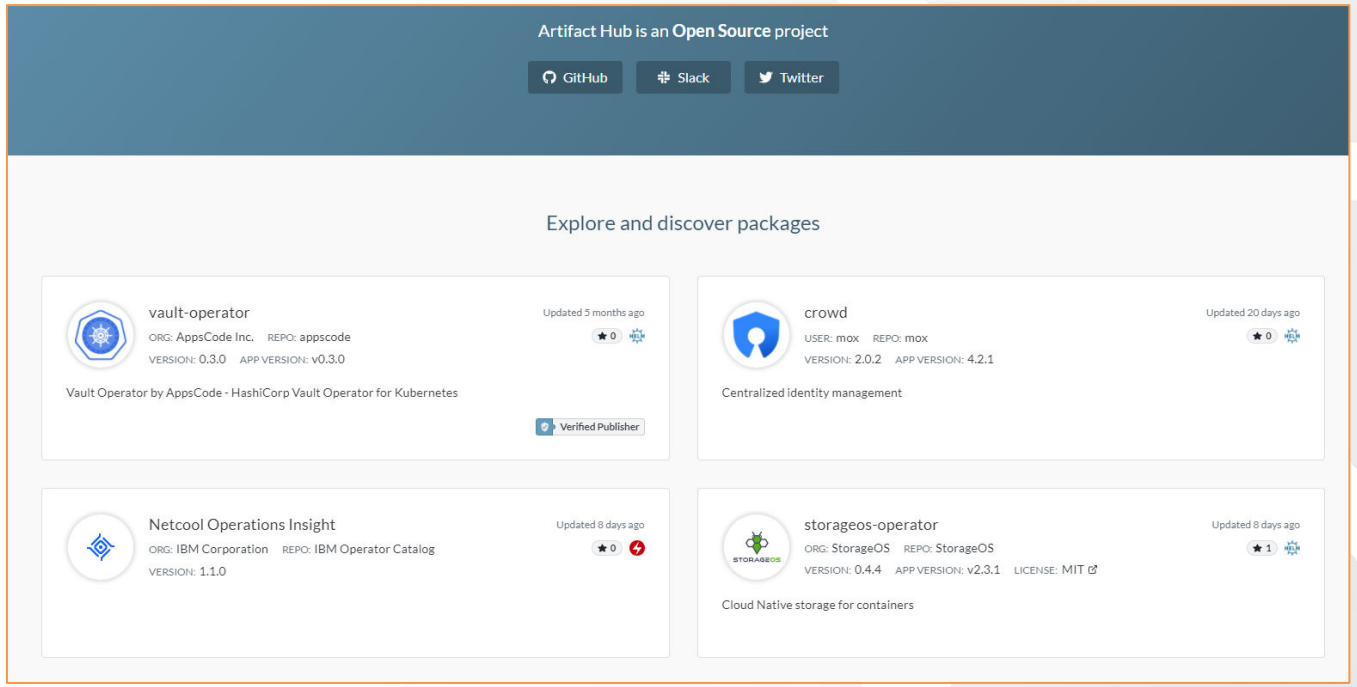
- Jfrog's Artifactory
- Nexus
- Azure artifacts

What is Helm?

- Helm is a **Package Manager** for Kubernetes
  - Package multiple K8s resources into a single logical deployment unit
  - But it's not just a Package Manager
- Helm is also a **Deployment Management** for Kubernetes
  - Do a repeatable deployment
  - Manage dependencies and multiple configurations
  - Update, rollback and test application deployments

Artifact Hub – Helm Artifact Warehouse

Artifact Hub is a SAAS based service provided by the CNCF Organization designated for finding, organizing and sharing Helm Charts from multiple sources easily.



Helm Concepts

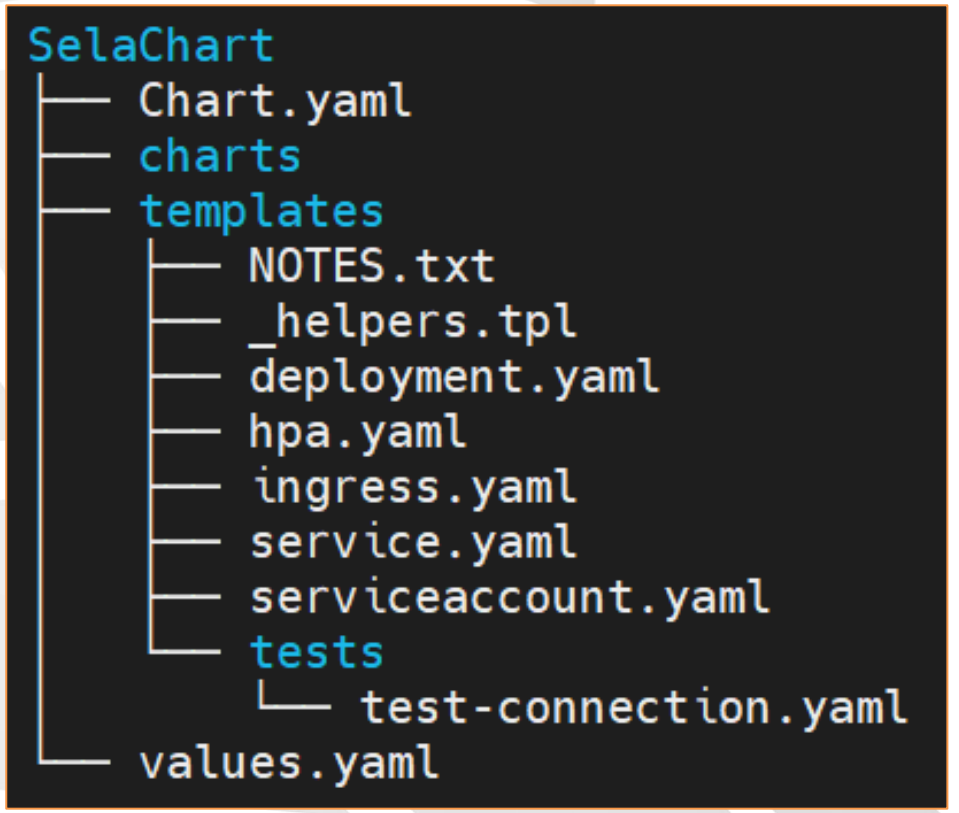
- **Chart:** a package; bundle of Kubernetes resources
- **Release:** a chart instance is loaded into Kubernetes
  - Same chart can be installed several times into the same cluster; each will have it's own Release
- **Repository:** a repository of published charts
- **Template:** a K8s configuration file mixed with go/Sprig template

Helm Repo's CLI related command

Command	Description
Helm Repo Add	add a chart repository
Helm Repo List	list chart repositories
Helm Repo Update	update information of available charts locally from chart repositories
Helm Repo Remove	remove one or more chart repositories
Helm Search Hub	search for charts in the Helm Hub or an instance of Monocular
Helm Search Repo	search repositories for a keyword in charts on local repositories

Cardinal Helm Chart Components

- **Chart.yaml:** A YAML file containing information about the main chart
- **Values.yaml:** A configuration file of values which will be used in the chart.yaml
- **Charts/:** A directory containing charts upon which main chart depends.
- **Templates/:** A directory of templates that, when combined with values, will generate valid Kubernetes manifest files.



Helm General Commands

Command	Description
Helm Install	This command installs a chart archive
Helm List	This command lists all of the releases for a specified namespace
Helm Upgrade	This command upgrades a release to a new version of a chart
Helm Rollback	roll back a release to a previous revision
Helm Show Chart	This command inspects a chart (directory, file, or URL) and displays the contents of the Charts.yaml file
Helm Show Values	This command inspects a chart (directory, file, or URL) and displays the contents of the values.yaml file

Questions

Lab 09: Deploying Prometheus using Helm

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-09

SELA

Module 14: Managed Offerings

Kubernetes Workshop

Agenda

- Local Development Kubernetes
- Managed Kubernetes providers
- Managed Cloud Kubernetes Differences
- Managed Cloud Kubernetes Advantages
- Managed Cloud Kubernetes Drawbacks
- AKS Deep-Dive
- Kubernetes On-Premise using Kubeadm and Kubespray

Local Development Kubernetes

- **Docker for Desktop (with Kubernetes Inside)**
  - Easiest method to create local Kubernetes cluster
  - Just check the box next to “Enable Kubernetes” and you’ll have a single node cluster
- **Kind (cri-o)**
  - Kind is a tool for running local Kubernetes clusters using Docker container “nodes”.
- **Minikube (VM)**
  - Minikube is a tool that makes it easy to run Kubernetes locally.
  - Minikube runs a single-node Kubernetes cluster inside a VM on your laptop for users looking to try out Kubernetes or develop with it day-to-day.

Managed Cloud Kubernetes Services

AKS

EKS

GKE

Managed Cloud Kubernetes Differences

- As of Dec 2020 the three providers are doing virtually the same things and have very similar capabilities with minor twitches.
- The main differences between them comes in the form of:
  - Underlying software components compatibilities
  - Pricing modules
  - Kubernetes cluster version adoption time
  - Limits and capacities of Nodes & Pods

Google Cloud Platform

Managed Cloud Kubernetes Advantages

- **No** need for **maintenances** of the Kubernetes cluster control plane components.
- It is **easy and fast** to create a managed cluster.
- Easy and transparent Kubernetes cluster upgrade.
- **Integrations** with other cloud services.
- **Preconfigured** crucial **components** such as Load Balancer are easy to deploy and create.
- Flexibility of automation within the cloud realm
- **Built-in authentication** and access controls for centralized management.
- Easy to manage and designated CLI for each provider

Google Cloud Platform

Managed Cloud Kubernetes Drawbacks

- Bounded to a **specific cloud provider** and his offerings
- The version of the Kubernetes cluster offerings will always be lately adopted.
- Limited underlying nodes OS options.
- Hard or impossible to customize crucial Kubernetes components such as networking.
- The usage of the services can come out pricy.
- Limited Container runtime offerings.

Google Cloud Platform

Kubernetes On Premise

- On-prem is much harder:
  - Consider Provisioning, Load balancing, upgrades, etc
- Two leading approaches:
  - Kubeadm
    - The best way to install and operate vanilla Kubernetes On-Prem
    - HA is not trivial
  - Kubespray
    - Ansible and ‘old’ provisioning approach. still very popular.

Identity And Security Management - AKS

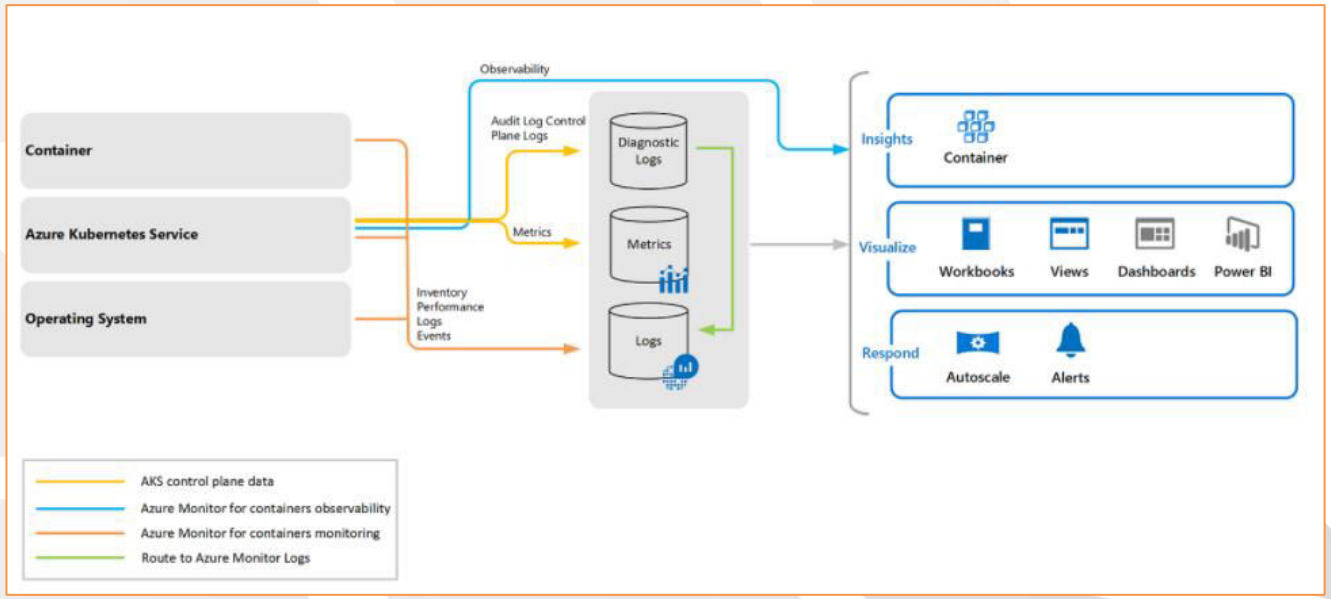
To limit access to cluster resources, AKS supports Kubernetes role-based access control (Kubernetes RBAC).
Kubernetes **RBAC** lets you control access to Kubernetes resources and namespaces, and permissions to those resources.
You can also configure an AKS cluster to integrate with **Azure Active Directory** (AD).
With Azure AD integration, Kubernetes access can be configured based on existing identity and group membership. Your existing Azure AD users and groups can be provided access to AKS resources and with an integrated sign-on experience.

Integrated Logging And Monitoring - AKS

To understand how your AKS cluster and deployed applications are performing, **Azure Monitor for container health** collects memory and processor metrics from containers, nodes, and controllers.

Container logs are available, and you can also review the Kubernetes master logs.

This monitoring data is stored in an **Azure Log Analytics** workspace, and is available through the Azure portal, Azure CLI, or a REST endpoint.



Clusters And Nodes - AKS

AKS nodes run on Azure virtual machines. You can connect storage to nodes and pods, upgrade cluster components, and use GPUs.

AKS supports Kubernetes clusters that run **multiple node pools** to support **mixed operating systems** and Windows Server containers.

Linux nodes run a customized Ubuntu OS image, and Windows Server nodes run a customized Windows Server 2019 OS image.

Virtual Networks And Ingress - AKS

AKS cluster can be deployed into an **existing virtual network.**

In this configuration, every pod in the cluster is assigned an IP address in the virtual network, and can directly communicate with other pods in the cluster, and other nodes in the virtual network.

Pods can connect also to other services in a peered virtual network, and to on-premises networks over ExpressRoute or site-to-site (S2S) VPN connections.

Development Tooling Integration - AKS

Kubernetes has a rich ecosystem of development and management tools such as Helm and the Kubernetes extension for **Visual Studio Code**. These tools work seamlessly with AKS.

Additionally, **Azure Dev Spaces** provides a rapid, iterative Kubernetes development experience for teams. With minimal configuration, you can run and debug containers directly in AKS.

The **Azure DevOps** project provides a simple solution for bringing existing code and Git repository into Azure.

The DevOps project automatically creates Azure resources such as AKS, a release pipeline in Azure DevOps Services that includes a build pipeline for CI, sets up a release pipeline for CD, and then creates an Azure Application Insights resource for monitoring.

AKS location Consideration

- Not all regions have an AKS service fully operational
- One should always check whether the region of the current and future business activity has a fully operational AKS service.
- The updated list can be found here: https://azure.microsoft.com/en-us/global-infrastructure/services/?products=kubernetes-service

		AZURE STACK HUB	AFRICA		ASIA PACIFIC		AUSTRALIA		
Products	Non-regional	Azure Stack Hub	South Africa North	South Africa West	East Asia	Southeast Asia	Australia Central	Australia Central 2	Australia East
Azure Kubernetes Service (AKS)			✓		✓	✓	✓		✓

Questions

SELA | DEVELOPER | PRACTICE
July 5-7, 2021

Kubernetes Advanced Scheduling

Noam Amrani

July 7, 2021

https://www.linkedin.com/in/noamamrani/

Requirements

- 4 -5 services
- Scale from 1 to 2000
- Predictable (mostly)

Agenda

- Requests & Limits
- Liveness, Readiness and Startup

Must

- Taints and Tolerations
- Affinity & AntiAffinity

Optional

Agenda

- Requests & Limits
- Liveness, Readiness and Startup
- Taints and Tolerations
- Affinity & AntiAffinity

Requests and Limits

CPU

resources are measured in cpu units.
One cpu (**"1000m"**), in Kubernetes, is equivalent to:

- 1 AWS/GCP/Azure/IBM vCPU
- 1 Hyperthread on a bare-metal Intel processor with Hyperthreading

Memory

resources are measured in bytes: E, P, T, G, M, K or Ei, Pi, Ti, Gi, Mi, Ki

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
  - name: wp
    image: wordpress
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

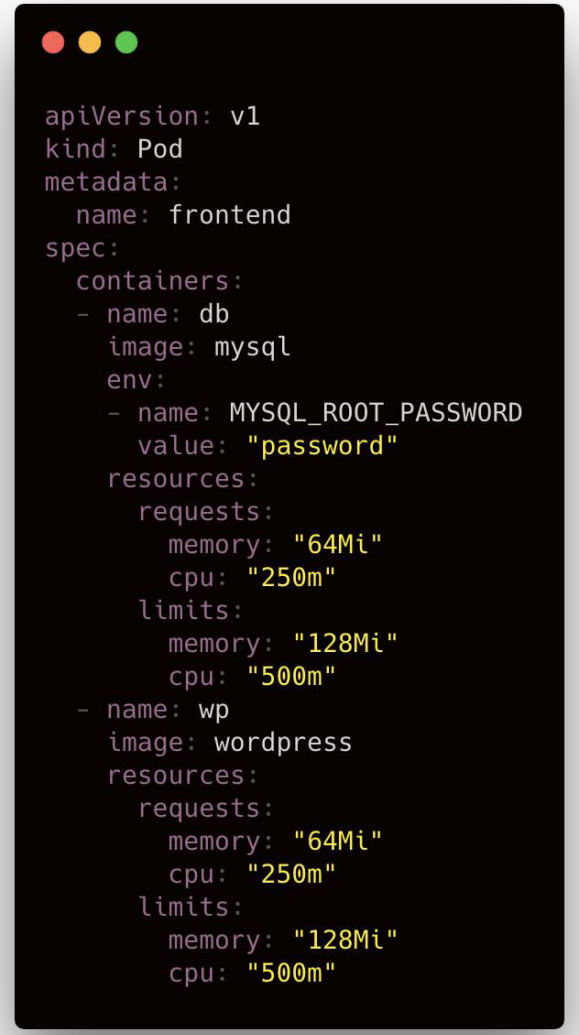
Requests and Limits

requests

minimum amount of compute resources required

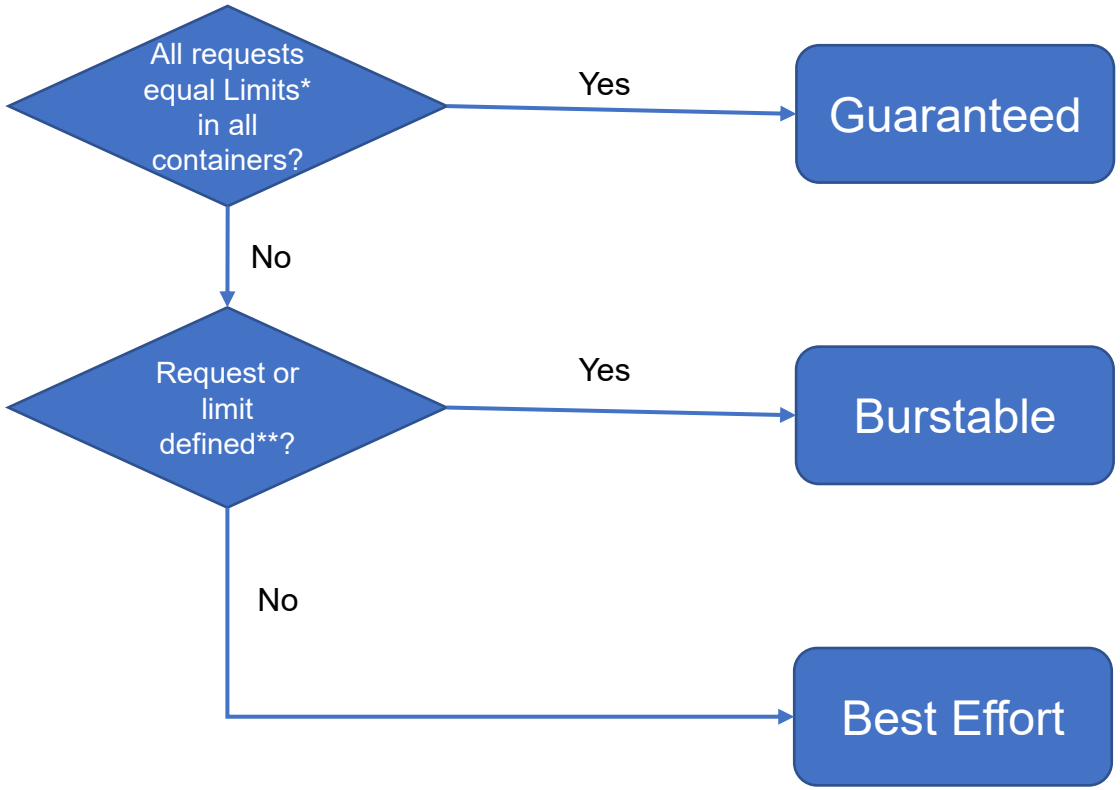
limits

maximum amount of compute resources allowed



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
  - name: wp
    image: wordpress
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

Requests and Limits: QoS

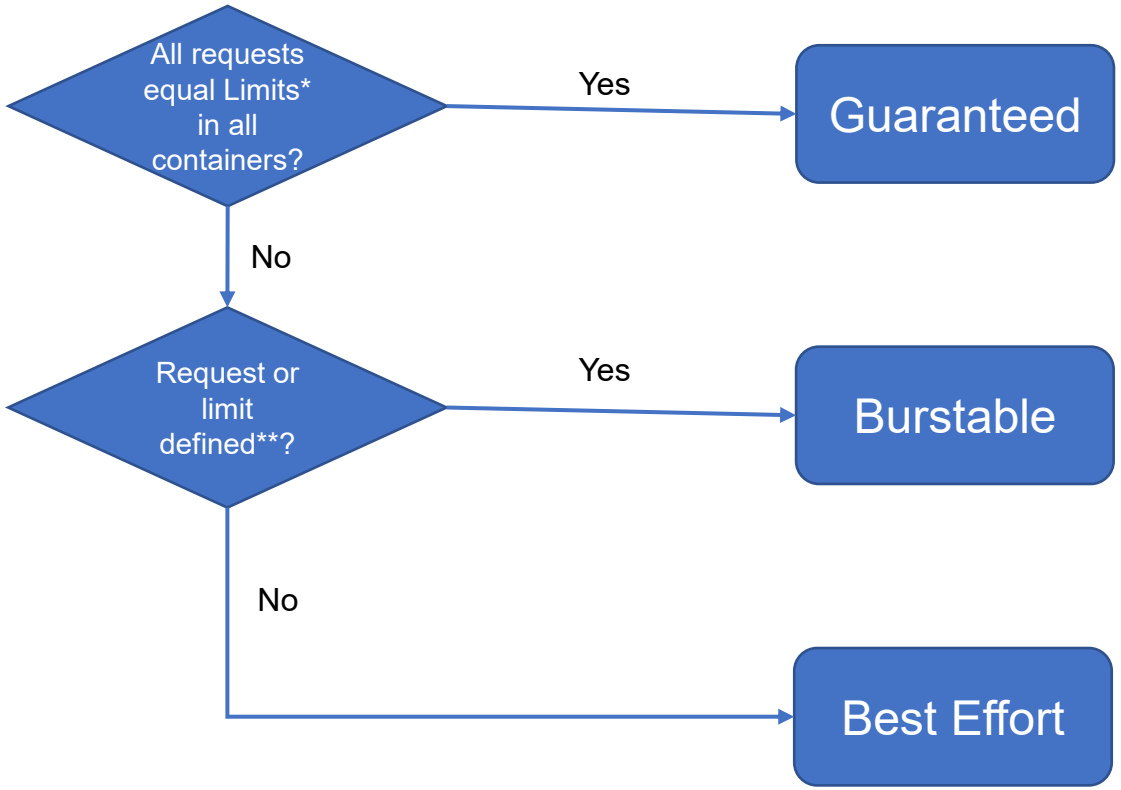



```yaml
apiVersion: v1
kind: Pod
metadata:
  name: frontend
spec:
  containers:
  - name: db
    image: mysql
    env:
    - name: MYSQL_ROOT_PASSWORD
      value: "password"
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
  - name: wp
    image: wordpress
    resources:
      requests:
        memory: "64Mi"
        cpu: "250m"
      limits:
        memory: "128Mi"
        cpu: "500m"
```

*If a Container specifies its own CPU/memory limit, but does not specify a CPU/memory request, Kubernetes automatically assigns a CPU/memory request that matches the limit.

**at least one container

Requests and Limits: QoS



Quality of Service	oom_score_adj	Range
Guaranteed	-997	-997
Burstable	min(max(2, 1000 - (1000 * memoryRequestBytes) / machineMemoryCapacityBytes), 999)	2-999
BestEffort	1000	1000

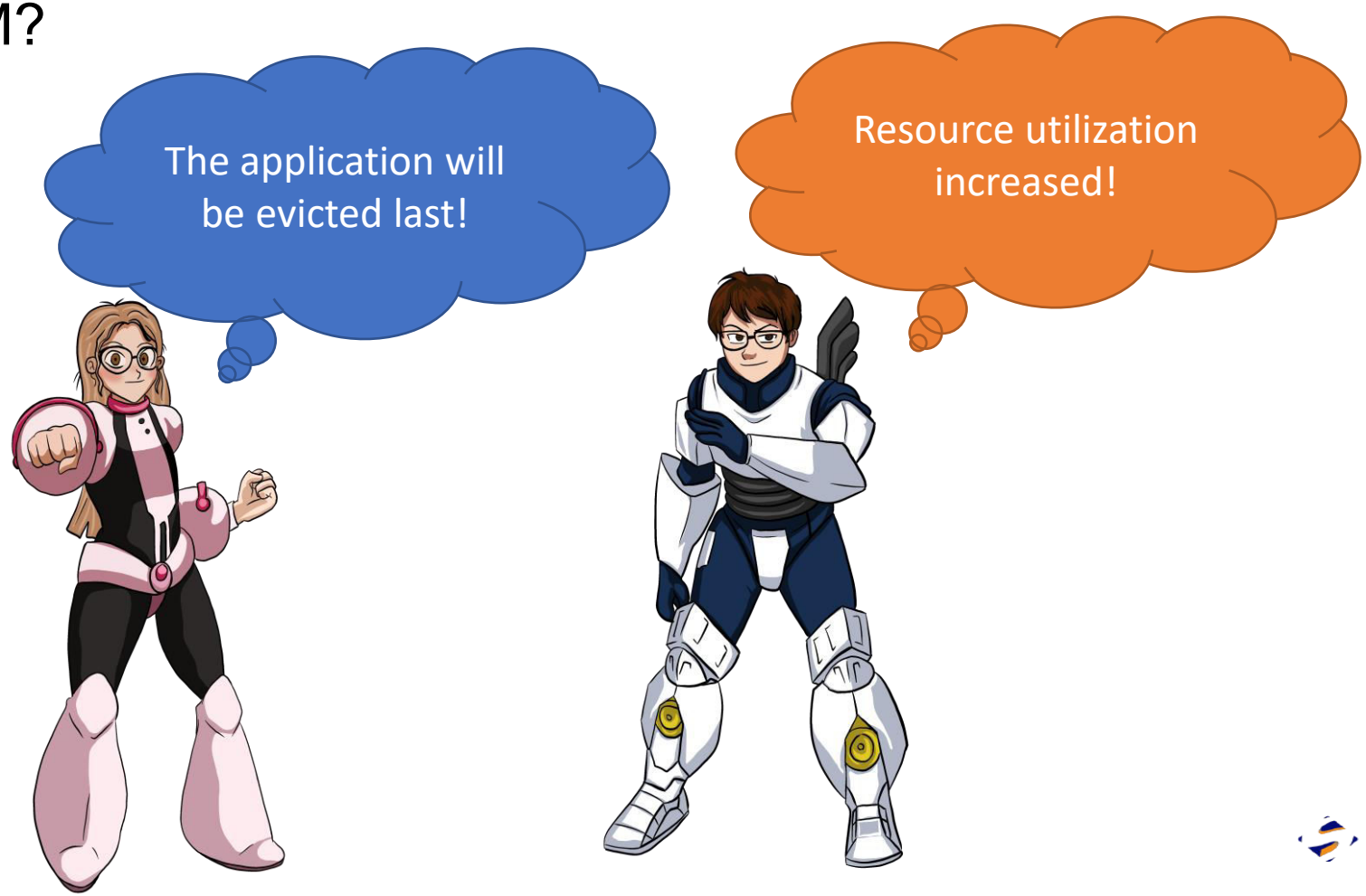
https://kubernetes.io/docs/concepts/scheduling-eviction/node-pressure-eviction/#node-conditions

*If a Container specifies its own CPU/memory limit, but does not specify a CPU/memory request, Kubernetes automatically assigns a CPU/memory request that matches the limit.

**at least one container

Requests and Limits: QoS

- WIIFM?



Agenda

- Requests & Limits
- **Liveness, Readiness and Startup**
- Taints and Tolerations
- Affinity & AntiAffinity

Readiness, Liveness & Startup

Types

Readiness

Kubernetes uses readiness probes to know when a container is ready to start accepting traffic

Liveness

Kubernetes uses liveness probes to know when to restart a container.

Startup

Kubernetes uses startup probes to know when a container application has started

Readiness, Liveness & Startup

Configuration

initialDelaySeconds

Number of seconds after the container has started before probes are initiated. Defaults to 0 seconds. Minimum value is 0.

periodSeconds

How often to perform the probe Default to 10 seconds. Minimum value is 1.

timeoutSeconds

Probe timeout Defaults to 1 second. Minimum value is 1.

failureThreshold

When a probe fails, Kubernetes will try *failureThreshold* times before giving up. Defaults to 3. Minimum value is 1.

successThreshold

How many successful probe checks are considered success Defaults to 1. Must be 1 for liveness and startup Probes. Minimum value is 1.

Readiness, Liveness & Startup

Actions

ExecAction:

Executes a specified command inside the container. The diagnostic is considered successful if the command exits with an exit code of **0**.

TCPSocketAction

Performs a TCP check against the Pod's IP address on a specified port. The diagnostic is considered successful if the port is **open**.

HTTPGetAction

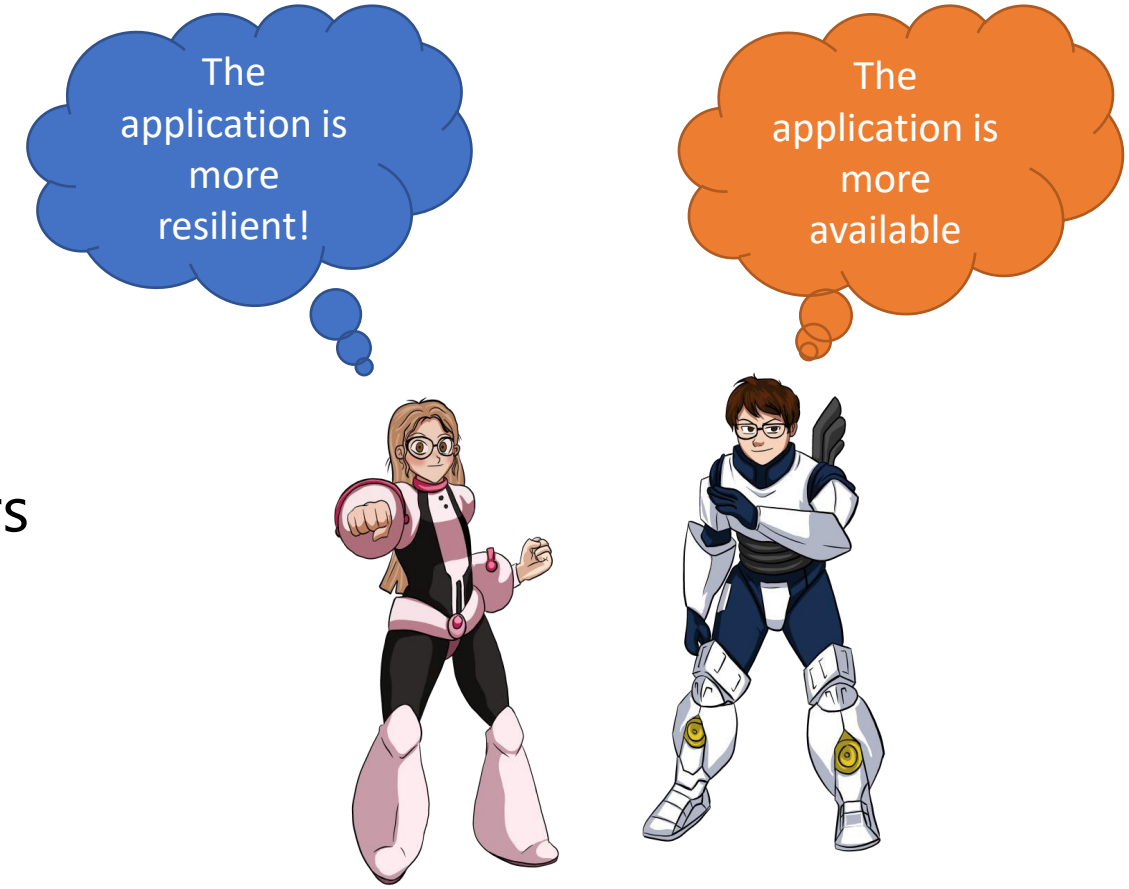
Performs an HTTP GET request against the Pod's IP address on a specified port and path. The diagnostic is considered successful if the response has a status code greater than or equal to **200** and less than **400**.

```yaml
# liveness.yaml
---
apiVersion: v1
kind: Pod
metadata:
  labels:
    test: liveness
  name: liveness-exec
spec:
  containers:
  - name: liveness
    image: k8s.gcr.io/busybox
    args:
    - /bin/sh
    - -c
    - touch /tmp/healthy; sleep 30; rm -rf /tmp/healthy; sleep 600
    livenessProbe:
      exec:
        command:
        - cat
        - /tmp/healthy
      initialDelaySeconds: 5
      periodSeconds: 5
```

Readiness, Liveness & Startup

Tips

- Check your containers liveness
- Check your containers readiness for servicing applications
- Use startup for long starting containers



Agenda

- Requests & Limits
- Liveness, Readiness and Startup
- **Taints and Tolerations**
- Affinity & AntiAffinity

Taints & Tolerations

Taints

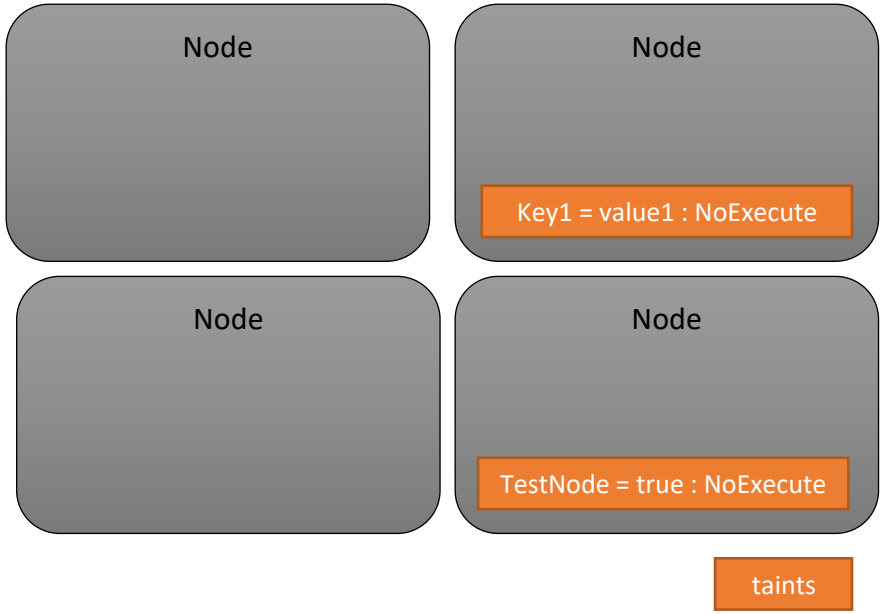
Allows users to mark a node (taint the node) so that no pods can be scheduled to it, unless a pod explicitly tolerates the taint.

Format

<key>=<value>:<effect>

Effects

NoSchedule / NoExcecute



Taints & Tolerations

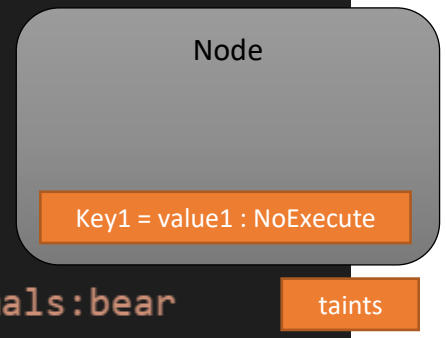
Tolerations
Allows pod to be scheduled / executed on a tainted node

operator
Equal / Exists

effect
NoSchedule / NoExcecute

tolerationSeconds
time to wait before evicting the pod

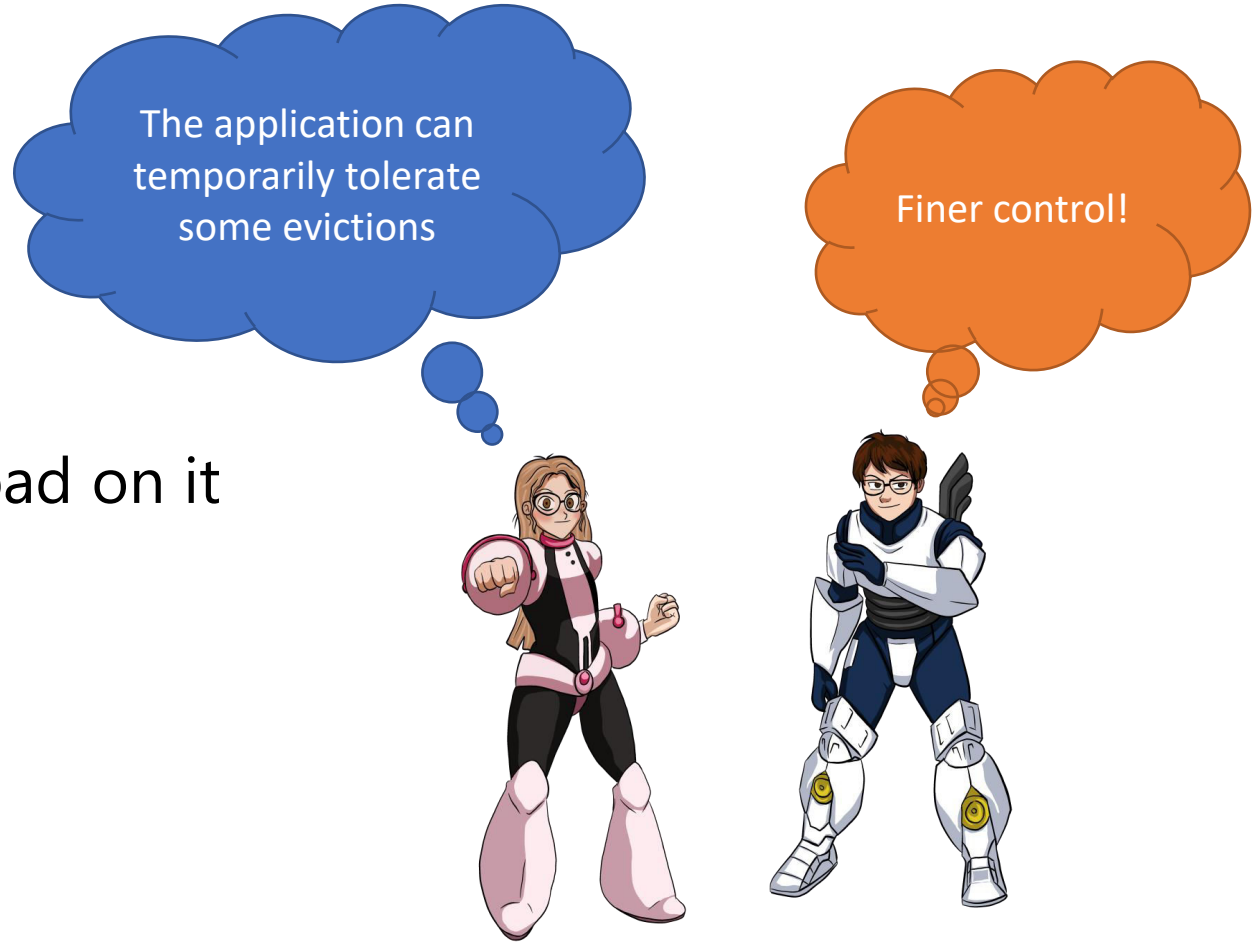
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: pod-2
  labels:
    security: s1
spec:
  containers:
  - name: bear
    image: supergiantkir/animals:bear
    resources:
  tolerations:
    - key: “key1"
      operator: “Equal“
      value: “value1"
      effect: "NoExecute"
    - key: "node.kubernetes.io/unreachable"
      operator: "Exists"
      effect: "NoExecute"
      tolerationSeconds: 6000
```



Taints and Tolerations

Use cases

- Dedicated Nodes
- Special Hardware nodes
- Taint a node to control the workload on it
- Dealing with taint based eviction



Agenda

- Requests & Limits
- Liveness, Readiness and Startup
- Taints and Tolerations
- Affinity & AntiAffinity

Node & inter-pod Affinity & Anti-Affinity

***Node** affinity*

constrains pods vs nodes' labels

***Inter-pod** affinity/anti-affinity*

constrains pods vs pods' labels on nodes with topologyKey

```yaml
# pod-with-affinity.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
            - e2e-az2
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
  containers:
  - name: with-node-affinity
    image: k8s.gcr.io/pause:2.0
```

Node Affinity & Anti-Affinity

requiredDuringSchedulingIgnoredDuringExecution
Hard

preferredDuringSchedulingIgnoredDuringExecution
Soft

operator
In, NotIn, Exists, DoesNotExist ,Gt ,Lt

*No requiredDuringSchedulingRequiredDuringExecution yet…

```yaml
# pod-with-affinity.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
            - e2e-az2
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
  containers:
  - name: with-node-affinity
    image: k8s.gcr.io/pause:2.0
```

Node Affinity & Anti-Affinity

nodeSelectorTerms

If you specify multiple nodeSelectorTerms, then the pod can be scheduled onto a node **if one** of the nodeSelectorTerms can be satisfied.

matchExpressions

If you specify multiple matchExpressions, then the pod can be scheduled onto a node only **if all** matchExpressions is satisfied.

weight (1-100)

For each node that meets all of the scheduling requirements (resource request, RequiredDuringScheduling affinity expressions, etc.), the scheduler will compute a sum by iterating through the elements of this field and adding "weight" to the sum if the node matches the corresponding MatchExpressions. This score is then combined with the scores of other priority functions for the node. The node(s) with the highest total score are the most preferred

```yaml
# pod-with-affinity.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: with-node-affinity
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: kubernetes.io/e2e-az-name
            operator: In
            values:
            - e2e-az1
            - e2e-az2
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: another-node-label-key
            operator: In
            values:
            - another-node-label-value
  containers:
  - name: with-node-affinity
    image: k8s.gcr.io/pause:2.0
```

Inter-Pod
Affinity & Anti-Affinity

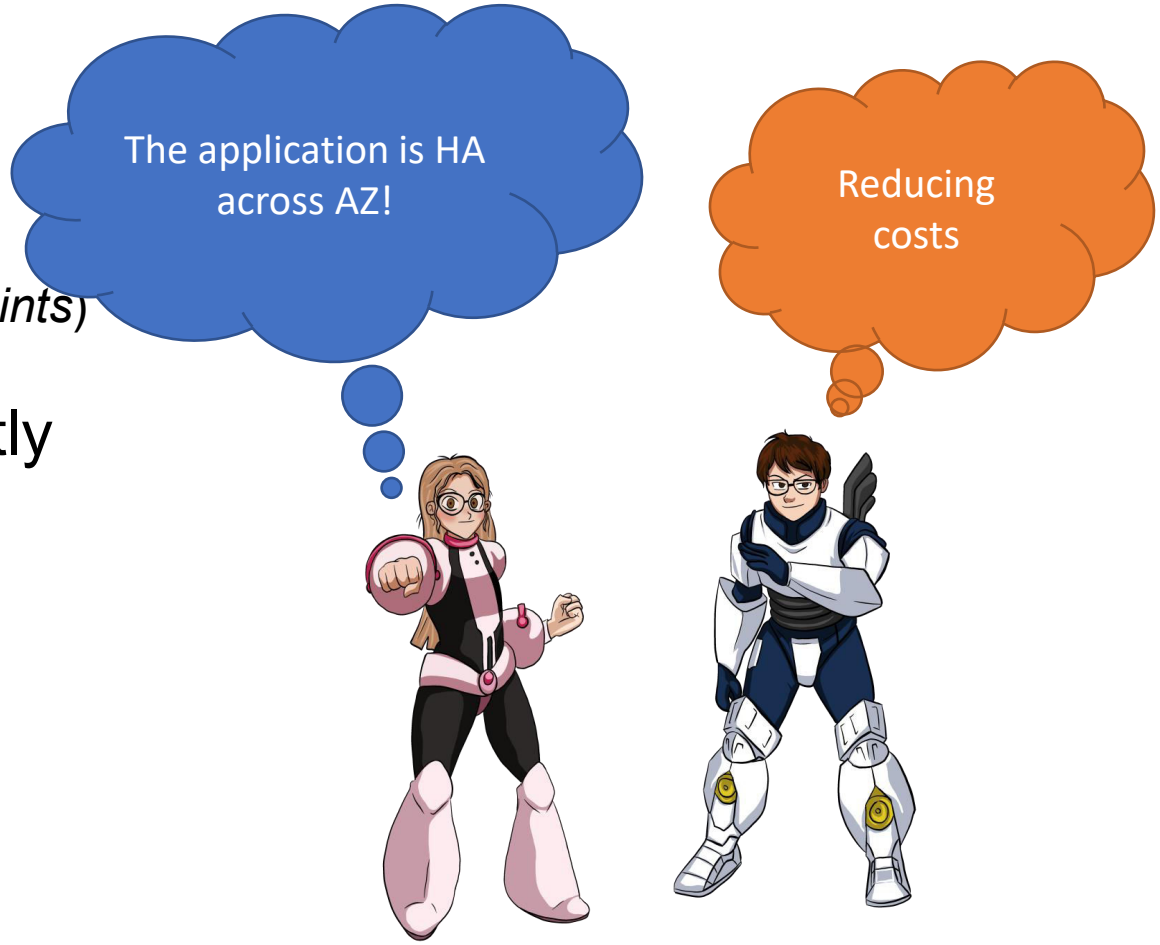
topologyKey:
a label with corresponding key must exist on the node

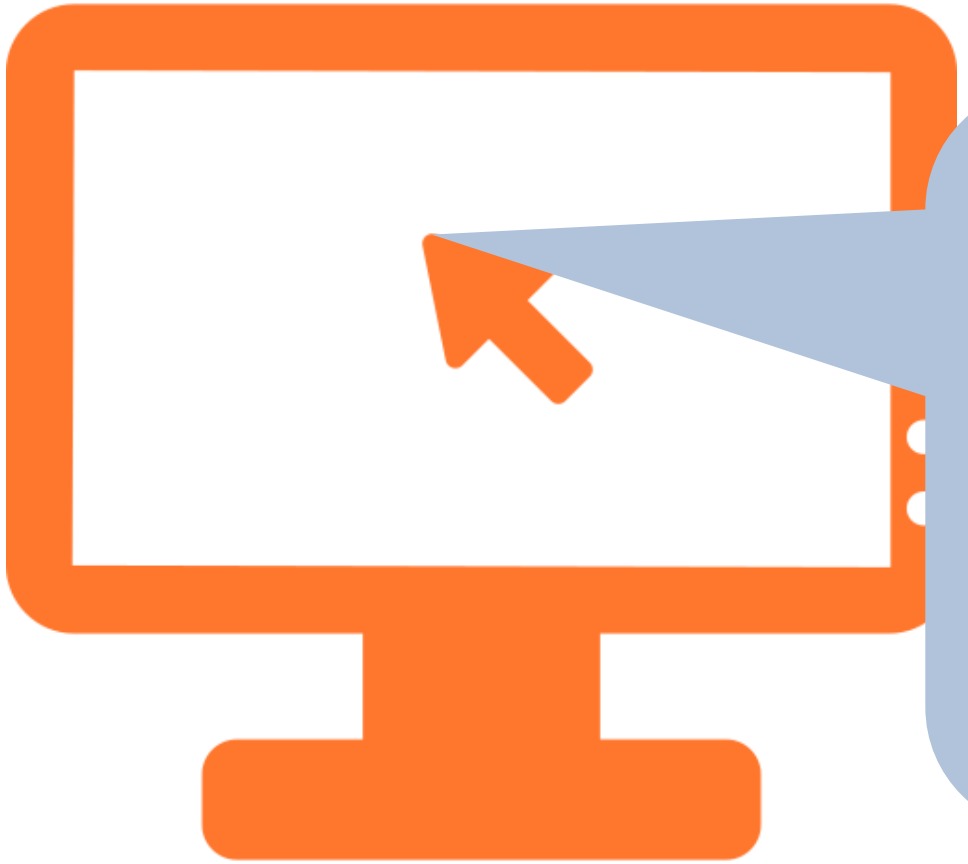
```yaml
# pod-with-interpod-affinity.yaml
---
apiVersion: v1
kind: Pod
metadata:
  name: with-pod-affinity
spec:
  affinity:
    podAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
      - labelSelector:
          matchExpressions:
          - key: security
            operator: In
            values:
            - S1
        topologyKey: topology.kubernetes.io/zone
    podAntiAffinity:
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 100
        podAffinityTerm:
          labelSelector:
            matchExpressions:
            - key: security
              operator: In
              values:
              - S2
          topologyKey: topology.kubernetes.io/zone
  containers:
  - name: with-pod-affinity
    image: k8s.gcr.io/pause:2.0
```

Affinity & Anti-Affinity

Use cases

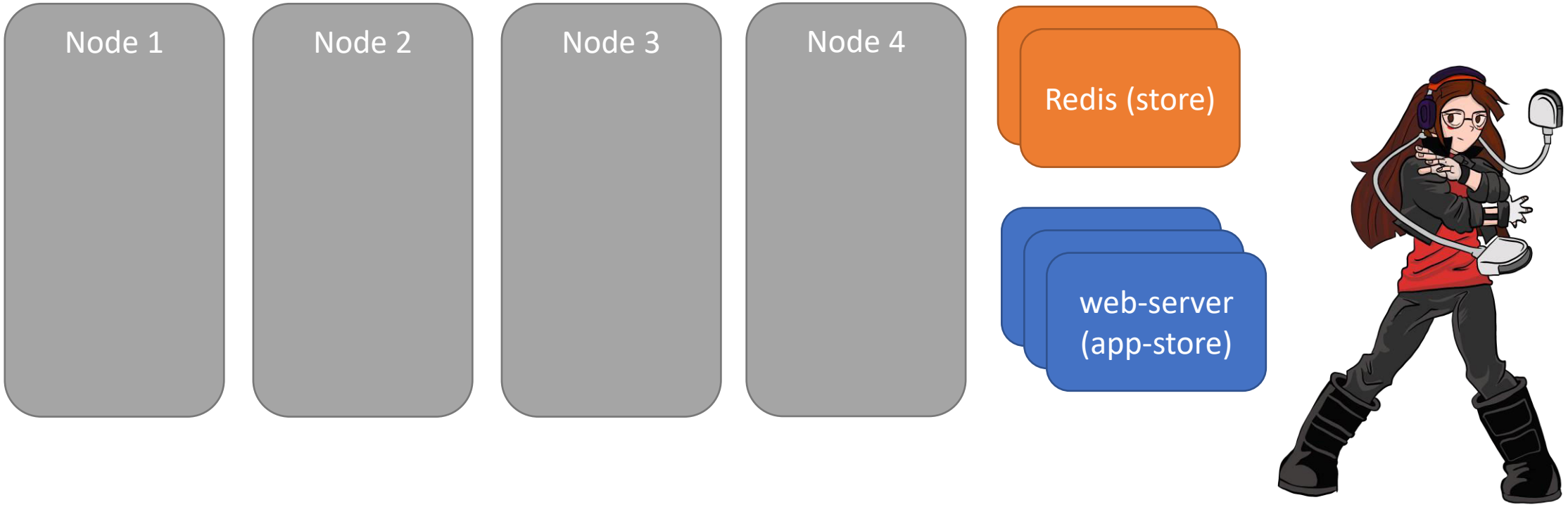
- Spread mission critical applications across AZ / nodes (*topologySpreadConstraints*)
- Prefer cheaper nodes over more costly (spot)



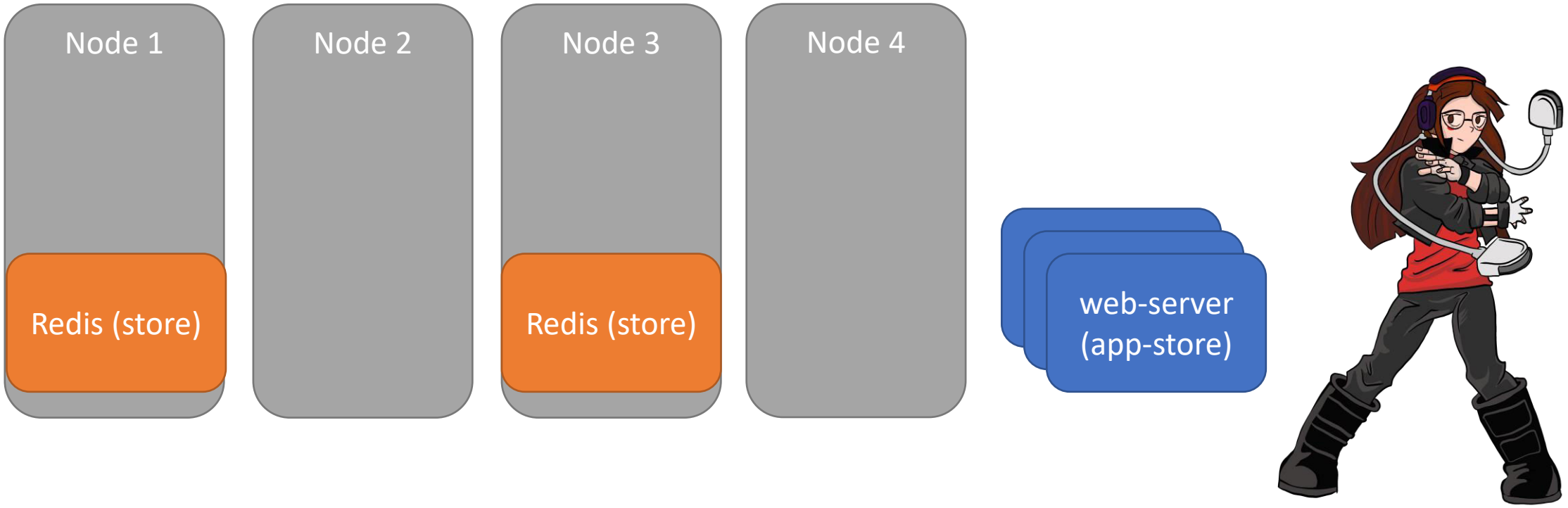


https://github.com/amraninoam/kuberenetes-advanced-scheduling

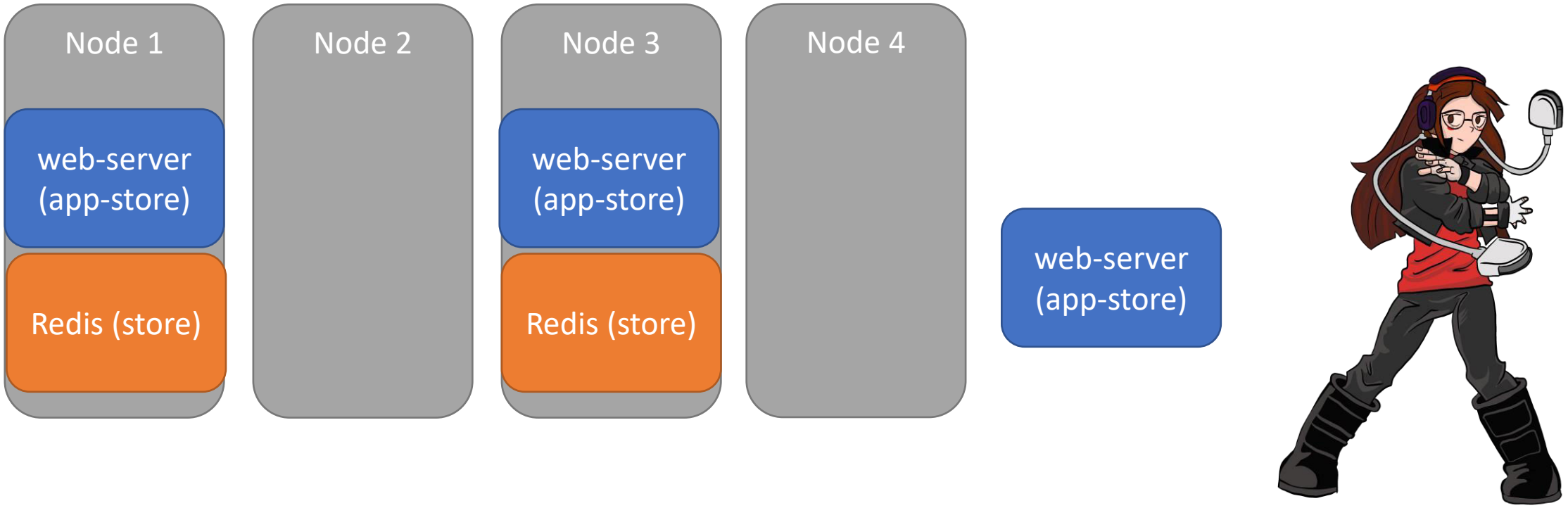
Demo

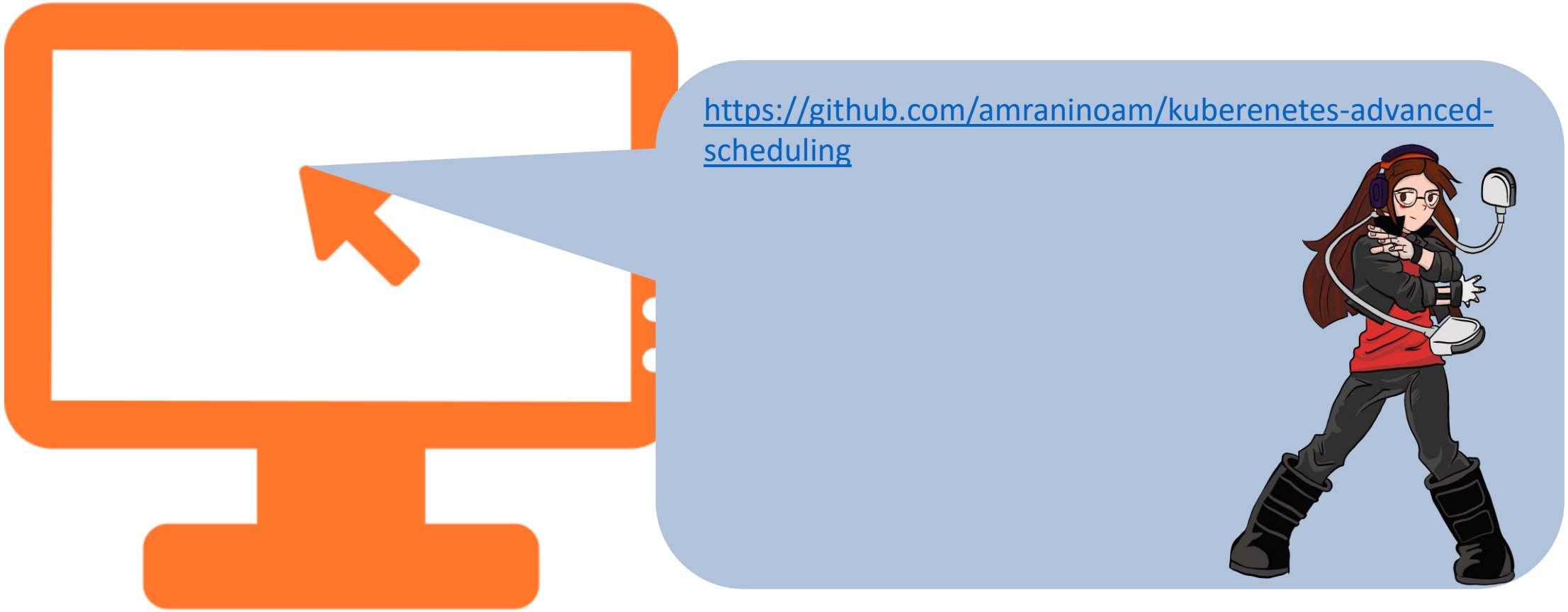


Demo


Node 1
Node 2
Node 3
Node 4
Redis (store)
Redis (store)
web-server
(app-store)

Demo


Node 1
web-server (app-store)
Redis (store)
Node 2
Node 3
web-server (app-store)
Redis (store)
Node 4
web-server (app-store)


https://github.com/amraninoam/kuberenetes-advanced-scheduling

What's next?

- Pod Topology Spread Constraints (1.19)

topologySpreadConstraints

maxSkew
The degree to which Pods may be unevenly distributed. It must be greater than zero. Its semantics differs according to the value of *whenUnsatisfiable*

topologyKey
The key of node labels.

whenUnsatisfiable
indicates how to deal with a Pod if it doesn't satisfy the spread constraint:
DoNotSchedule / ScheduleAnyway

labelSelector
used to find matching Pods

```yaml
kind: Pod
apiVersion: v1
metadata:
  name: mypod
  labels:
    foo: bar
spec:
  topologySpreadConstraints:
  - maxSkew: 1
    topologyKey: zone
    whenUnsatisfiable: DoNotSchedule
    labelSelector:
      matchLabels:
        foo: bar
  containers:
  - name: pause
    image: k8s.gcr.io/pause:3.1
```

*As of Kubernetes 1.19, previously known as EvenPodsSpread

Questions?

Noam Amrani

Module 16 – Autoscaling

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

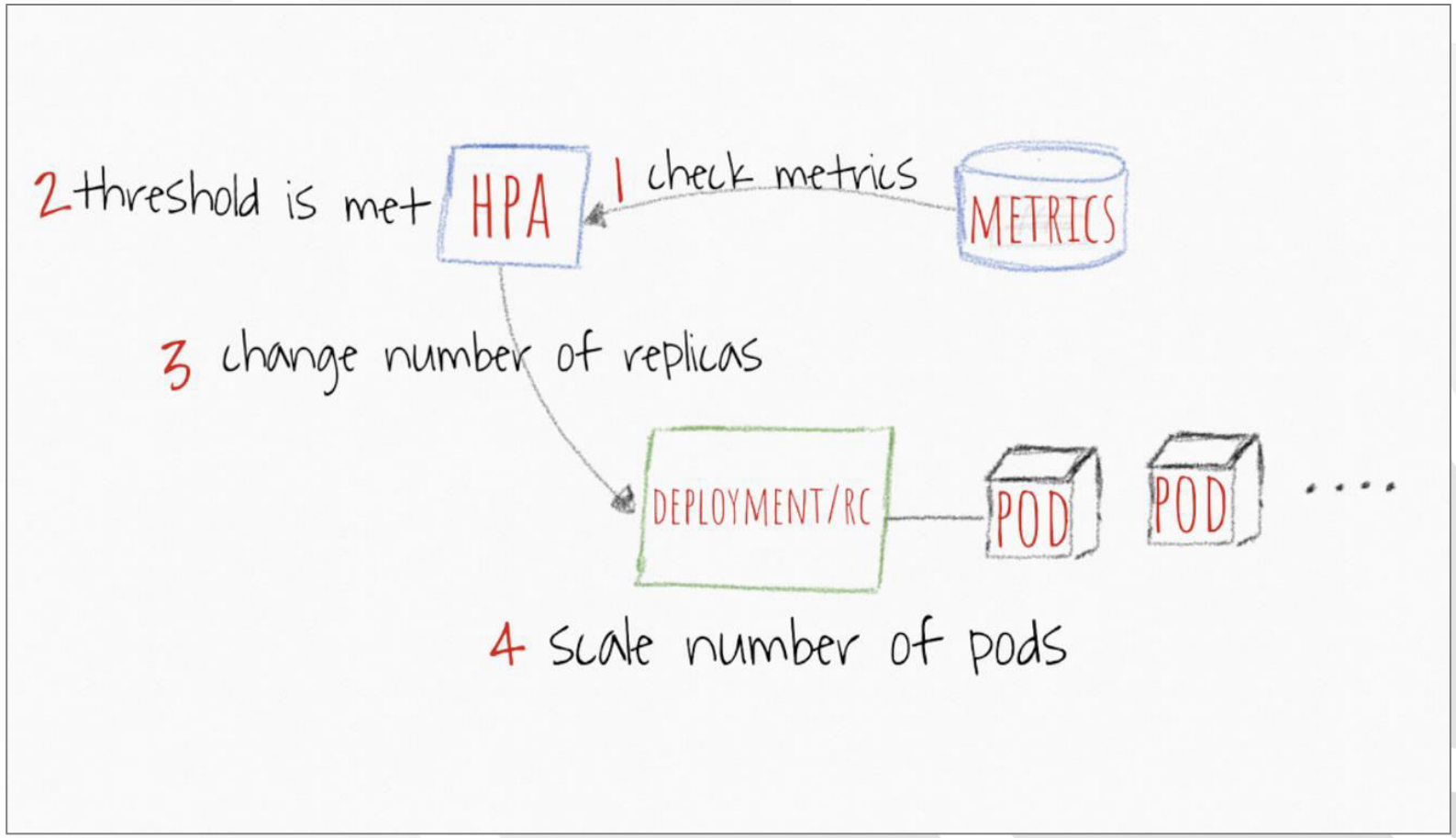
- Scaling pods horizontally
- Scaling pods vertically
- Scaling cluster horizontally
- Lab 10: Configuring Autoscaling

Horizontal Pod Autoscaler

- Horizontal Pod Autoscaler automatically scales the number of pods in a replication controller, deployment or replica set based on observed CPU utilization, or, with beta support, on some other, application-provided metrics
- Uses Kubernetes Metrics Server to gather stats

```
kubectl autoscale deployment nginx --cpu-percent=50 --min=1 --max=10
```

Horizontal Pod Autoscaler



Horizontal Pod Autoscaler

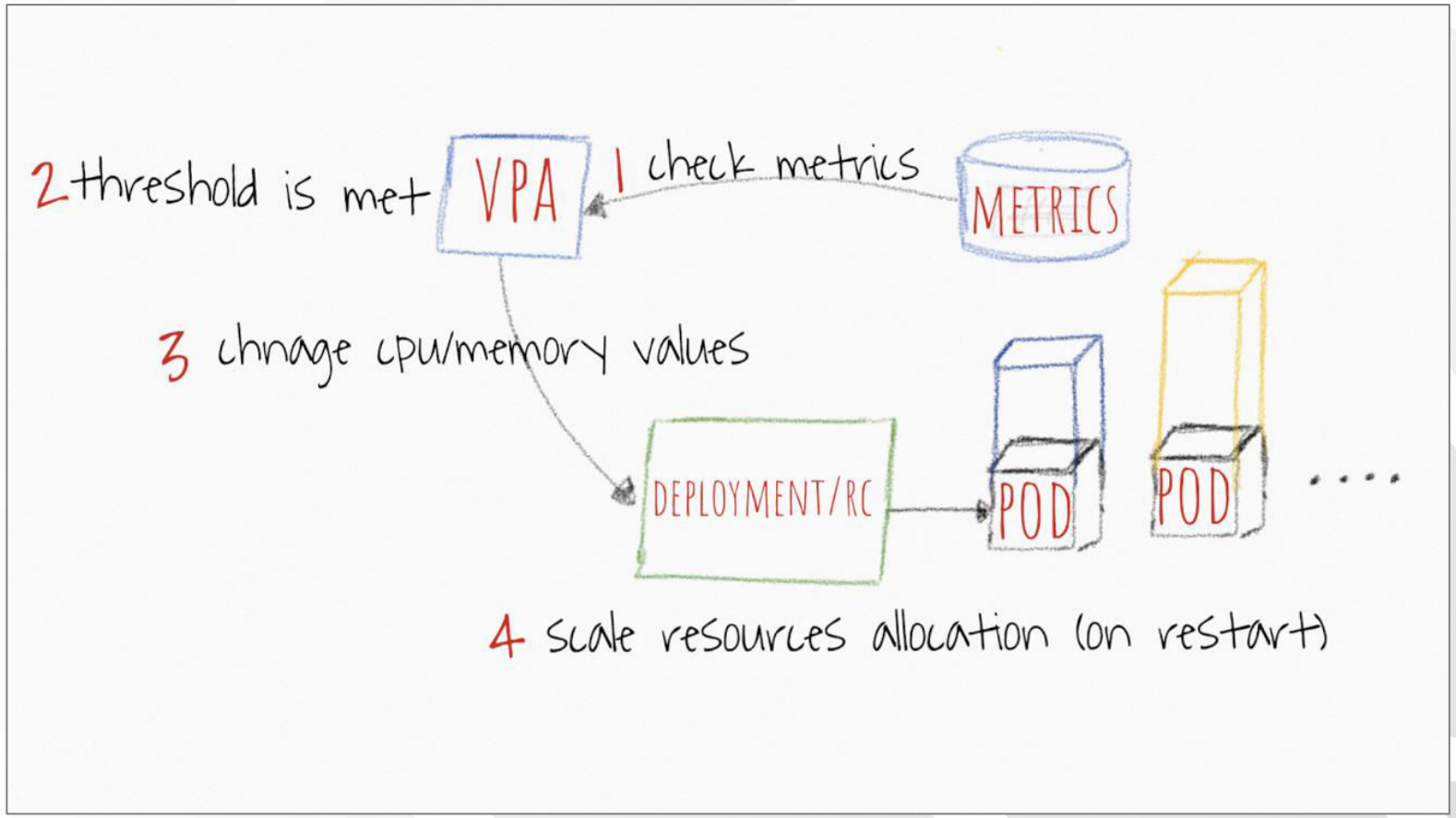
```yaml
# hpa-autoscaling.yaml
---
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
  name: php-apache
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: php-apache
  minReplicas: 1
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      target:
        type: Utilization
        averageUtilization: 50
  - type: Pods
    pods:
      metric:
        name: packets-per-second
      target:
        type: AverageValue
        averageValue: 1k
```

```yaml
# as of 1.18
[...]
behavior:
  scaleDown:
    policies:
    - type: Pods
      value: 4
      periodSeconds: 60
    - type: Percent
      value: 10
      periodSeconds: 60
    selectPolicy: Max
```

Vertical Pod Autoscaler

- Vertical Pod Autoscaler (VPA) frees the users from necessity of setting up-to-date resource requests for the containers in their pods.
- When configured, it will set the requests automatically based on usage and thus allow proper scheduling onto nodes so that appropriate resource amount is available for each pod.
- It can both down-scale pods that are over-requesting resources, and also up-scale pods that are under-requesting resources based on their usage over time.

Vertical Pod Autoscaler



Vertical Pod Autoscaler

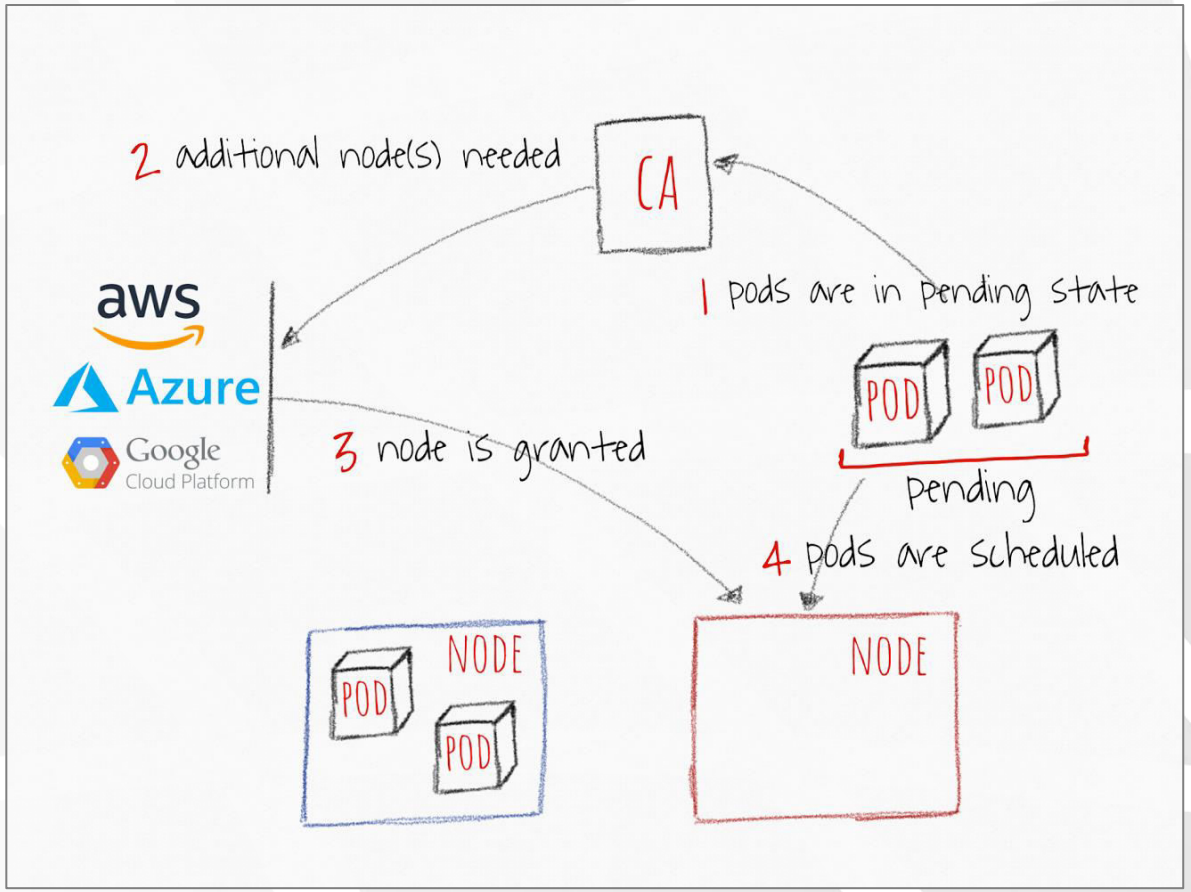
- Can be limited with Limit Range
- updatePolicy: Auto, Recreate, Initial, Off
- Should not be used with HPA on CPU or memory

```yaml
# vpa-autoscaling.yaml
---
apiVersion: autoscaling.k8s.io/v1
kind: VerticalPodAutoscaler
metadata:
  name: my-app-vpa
spec:
  targetRef:
    apiVersion: "apps/v1"
    kind:       Deployment
    name:       my-app
  updatePolicy:
    updateMode: "Auto"
```

Cluster Auto-Scaler

- Cluster Autoscaler is a tool that automatically adjusts the size of the Kubernetes cluster when one of the following conditions is true:
  - there are pods that failed to run in the cluster due to insufficient resources
  - there are nodes in the cluster that have been underutilized for an extended period of time and their pods can be placed on other existing nodes.
- Supported on GCP, AWS, Azure, Alibaba Cloud

Cluster Auto-Scaler



Questions

Lab 10: Configuring Autoscaling

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-10

Noam Amrani

Module 17 - Storage

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

- Introduction
- Volumes
- PersistentVolumes
- PersistentVolumeClaims
- StorageClasses

Storage

- Pods by themselves are useful, but many workloads require exchanging data between containers, or persisting some form of data.

- For this we have:

  - Volumes
  - PersistentVolumes
  - PersistentVolumeClaims
  - StorageClasses

Volumes

- Storage that is tied to the Pod's Lifecycle
- A pod can have one or more types of volumes attached to it.
- Can be consumed by any of the containers within the pod.
- Survive Pod restarts; however their durability beyond that is dependent on the Volume Type.

Volumes

- volumes: A list of volume objects to be attached to the Pod. Every object within the list must have it's own unique name.
- volumeMounts: A container specific list referencing the Pod volumes by name, along with their desired mountPath.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: volume-example
spec:
  containers:
  - name: nginx
    image: nginx:stable-alpine
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
      ReadOnly: true
  - name: content
    image: alpine:latest
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          date >> /html/index.html;
          sleep 5;
        done
    volumeMounts:
    - name: html
      mountPath: /html
  volumes:
  - name: html
    emptyDir: {}
```

Volumes

- volumes: A list of volume objects to be attached to the Pod. Every object within the list must have it's own unique name.

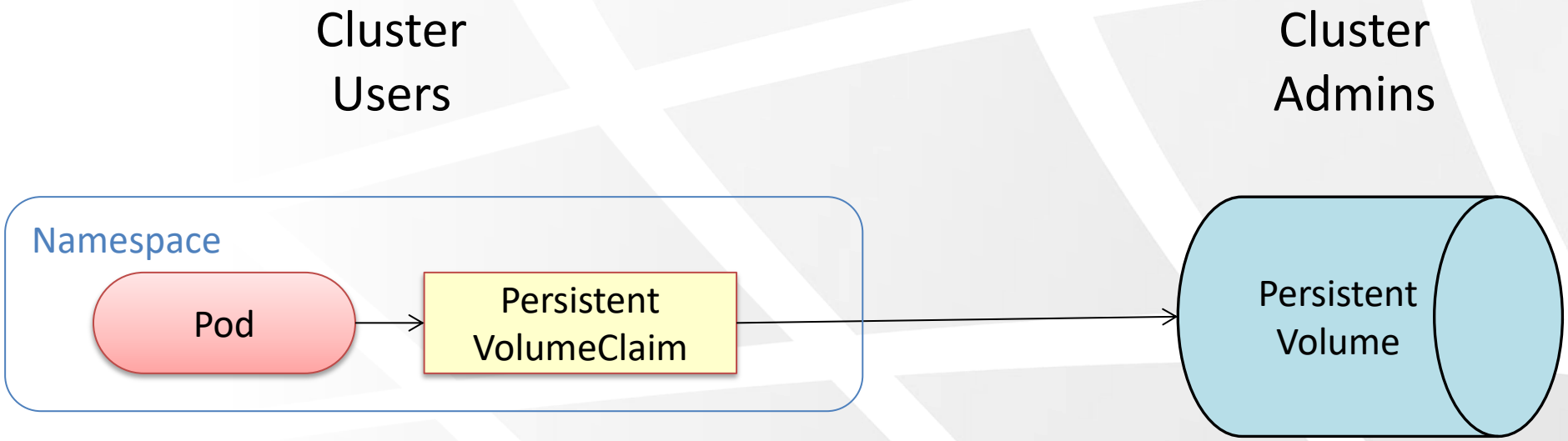
- volumeMounts: A container specific list referencing the Pod volumes by name, along with their desired mountPath.

Volume Types

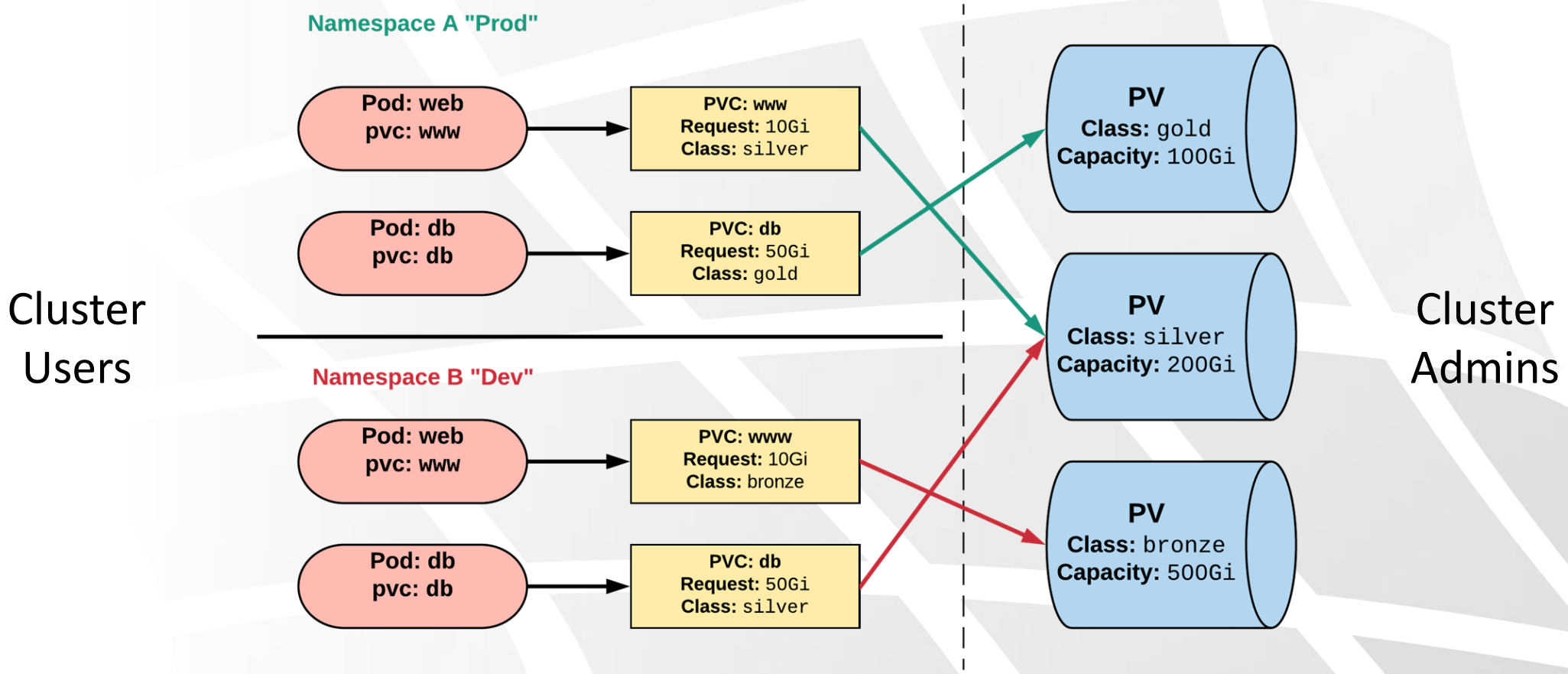
- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- configMap
- csi
- emptyDir
- fc (fibre channel)
- flocker
- gcePersistentDisk
- gitRepo
- glusterfs
- hostPath
- iscsi
- local
- nfs
- portworxVolume
- quobyte
- rbd
- scaleIO
- secret
- storageos
- vsphereVolume

Persistent Volume Supported

Persistent Volumes and Claims



Persistent Volumes and Claims



Persistent Volumes

- A **PersistentVolume** (PV) represents a storage resource.
- PVs are a **cluster wide resource** linked to a backing storage provider: NFS, GCEPersistentDisk, EBS etc.
- Generally provisioned by an administrator
- Their lifecycle is handled independently from a pod
- **CANNOT** be attached to a Pod directly. Relies on a **PersistentVolumeClaim**

PersistentVolume

- capacity.storage: The total amount of available storage.
- volumeMode: The type of volume, this can be either *Filesystem* or *Block*.
- accessModes: A list of the supported methods of accessing the volume. Options include:
  - *ReadWriteOnce*
  - *ReadOnlyMany*
  - *ReadWriteMany*

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

PersistentVolume

- capacity.storage: The total amount of available storage.
- volumeMode: The type of volume, this can be either *Filesystem* or *Block*.
- accessModes: A list of the supported methods of accessing the volume. Options include:
  - *ReadWriteOnce*
  - *ReadOnlyMany*
  - *ReadWriteMany*

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

PersistentVolume

- capacity.storage: The total amount of available storage.
- volumeMode: The type of volume, this can be either *Filesystem* or *Block*.
- accessModes: A list of the supported methods of accessing the volume. Options include:
  - *ReadWriteOnce*
  - *ReadOnlyMany*
  - *ReadWriteMany*

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

PersistentVolume

- persistentVolumeReclaimPolicy: The behaviour for PVC's that have been deleted. Options include:
  - *Retain* - manual clean-up
  - *Delete* - storage asset deleted by provider.
- storageClassName: Optional name of the storage class that PVC's can reference. If provided, **ONLY** PVC's referencing the name can consume it.
- mountOptions: Optional mount options for the PV.

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

PersistentVolume

- persistentVolumeReclaimPolicy: The behaviour for PVC's that have been deleted. Options include:
  - *Retain* - manual clean-up
  - *Delete* - storage asset deleted by provider.
- storageClassName: Optional name of the storage class that PVC's can reference. If provided, **ONLY** PVC's referencing the name can consume it.
- mountOptions: Optional mount options for the PV.

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

PersistentVolume

- persistentVolumeReclaimPolicy: The behaviour for PVC's that have been deleted. Options include:
  - *Retain* - manual clean-up
  - *Delete* - storage asset deleted by provider.
- storageClassName: Optional name of the storage class that PVC's can reference. If provided, **ONLY** PVC's referencing the name can consume it.
- mountOptions: Optional mount options for the PV.

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: nfsserver
spec:
  capacity:
    storage: 50Gi
  volumeMode: Filesystem
  accessModes:
    - ReadWriteOnce
    - ReadWriteMany
  persistentVolumeReclaimPolicy: Delete
  storageClassName: slow
  mountOptions:
    - hard
    - nfsvers=4.1
  nfs:
    path: /exports
    server: 172.22.0.42
```

PersistentVolumeClaims

- A **PersistentVolumeClaim** (PVC) is a **namespaced** request for storage.
- Satisfies a set of requirements instead of mapping to a storage resource directly.
- Ensures that an application's 'claim' for storage is portable across numerous backends or providers.

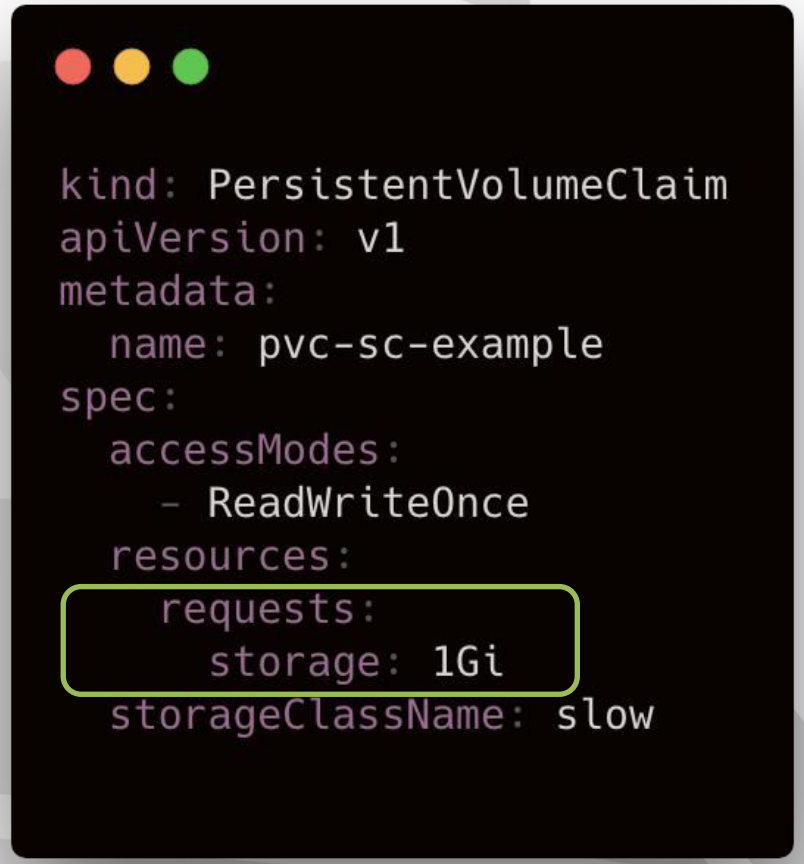
PersistentVolumeClaim

- accessModes: The selected method of accessing the storage. This **MUST** be a subset of what is defined on the target PV or Storage Class.
  - *ReadWriteOnce*
  - *ReadOnlyMany*
  - *ReadWriteMany*
- resources.requests.storage: The desired amount of storage for the claim
- storageClassName: The name of the desired Storage Class

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-sc-example
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: slow
```

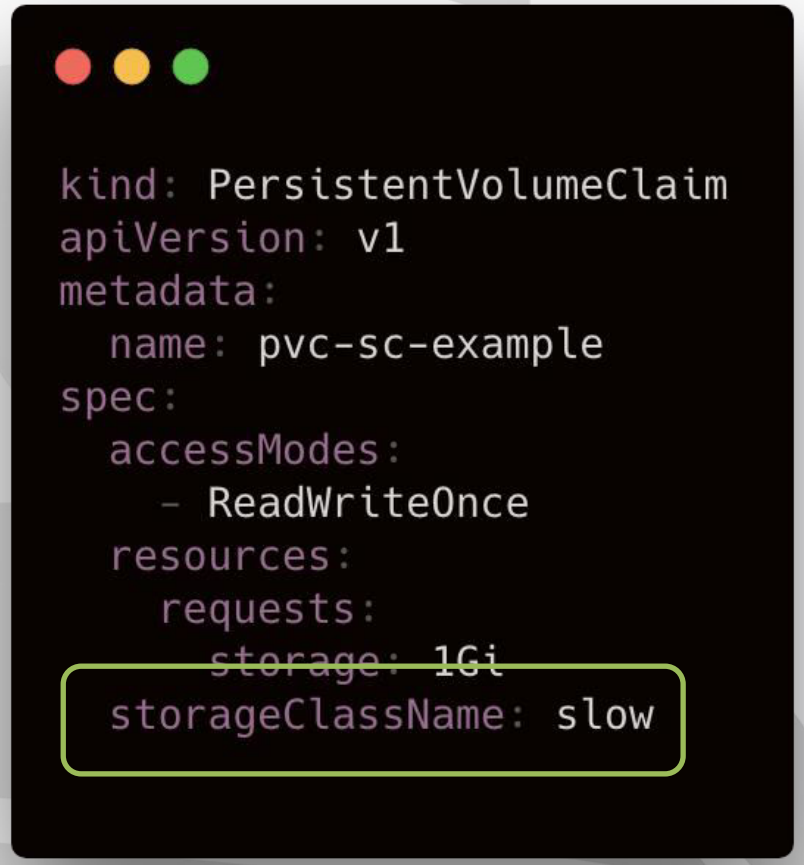
PersistentVolumeClaim

- accessModes: The selected method of accessing the storage. This **MUST** be a subset of what is defined on the target PV or Storage Class.
  - *ReadWriteOnce*
  - *ReadOnlyMany*
  - *ReadWriteMany*
- resources.requests.storage: The desired amount of storage for the claim
- storageClassName: The name of the desired Storage Class



PersistentVolumeClaim

- accessModes: The selected method of accessing the storage. This **MUST** be a subset of what is defined on the target PV or Storage Class.
  - *ReadWriteOnce*
  - *ReadOnlyMany*
  - *ReadWriteMany*
- resources.requests.storage: The desired amount of storage for the claim
- storageClassName: The name of the desired Storage Class

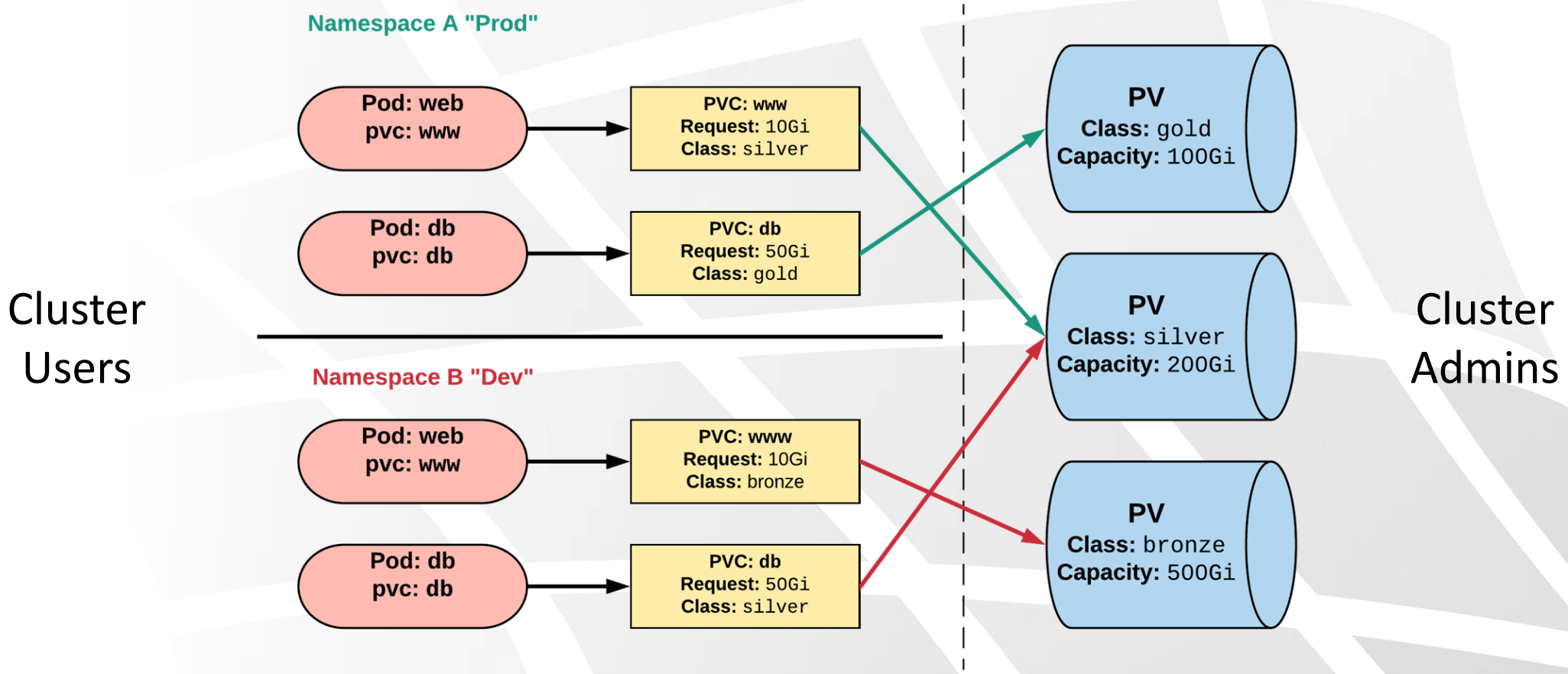


PVs and PVCs with Selectors

```yaml
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-selector-example
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  selector:
    matchLabels:
      type: hostpath
```

```yaml
kind: PersistentVolume
apiVersion: v1
metadata:
  name: pv-selector-example
  labels:
    type: hostpath
spec:
  capacity:
    storage: 2Gi
  accessModes:
    - ReadWriteMany
  hostPath:
    path: "/mnt/data"
```

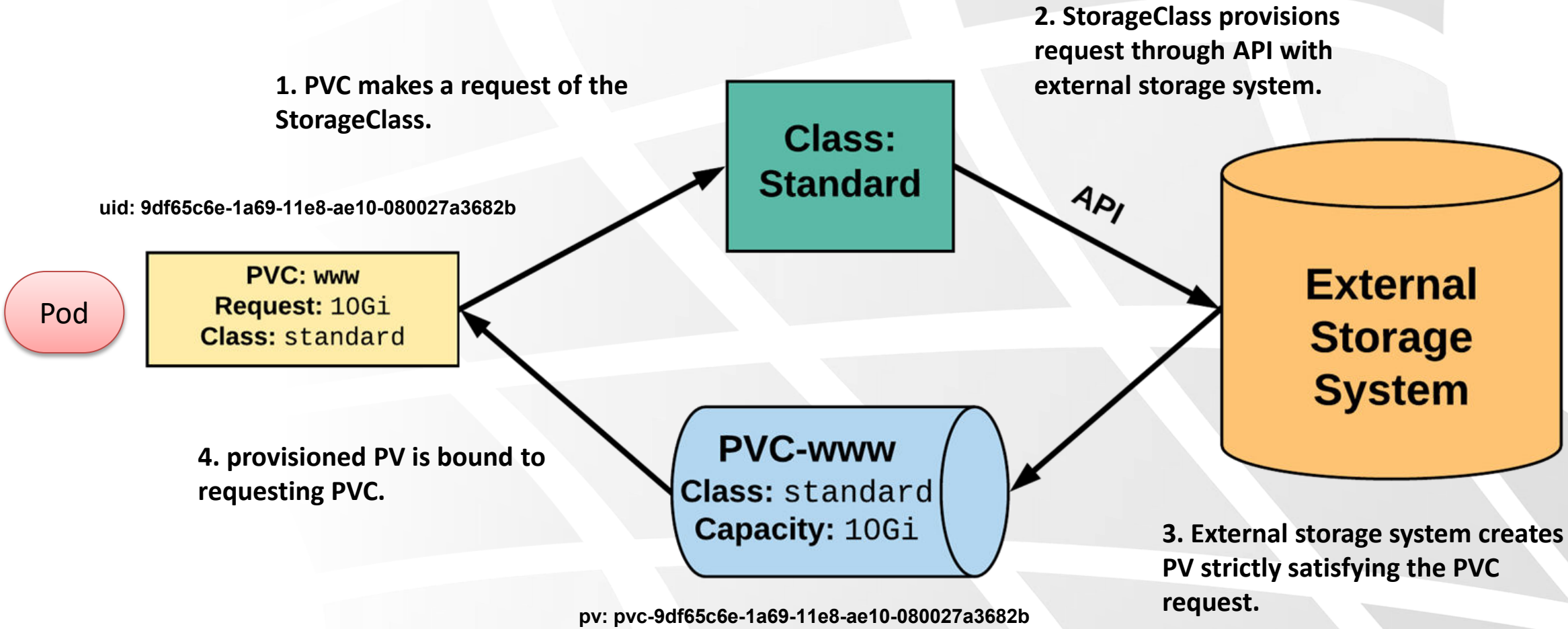
Persistent Volumes and Claims



StorageClass

- Storage classes are an abstraction on top of an external storage resource (PV)
- Work hand-in-hand with the external storage system to enable **dynamic provisioning** of storage
- Eliminates the need for the cluster admin to pre-provision a PV

StorageClass



StorageClass

- provisioner: Defines the '*driver*' to be used for provisioning of the external storage.
- parameters: A hash of the various configuration parameters for the provisioner.
- reclaimPolicy: The behaviour for the backing storage when the PVC is deleted.
  - *Retain* - manual clean-up
  - *Delete* - storage asset deleted by provider

```
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
  zones: us-central1-a, us-central1-b
reclaimPolicy: Delete
```

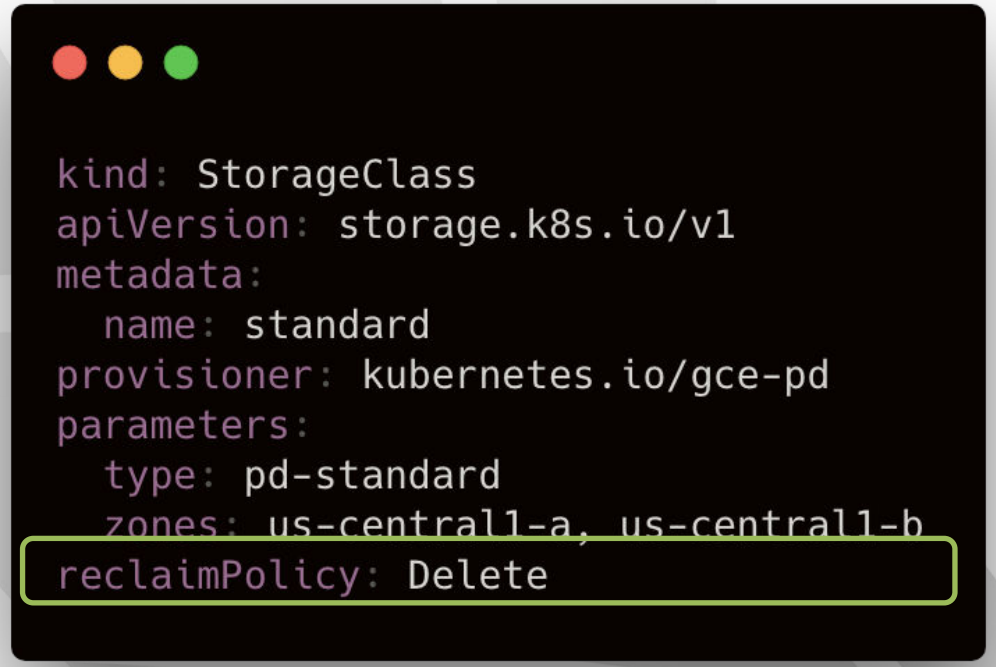
StorageClass

- provisioner: Defines the '*driver*' to be used for provisioning of the external storage.
- parameters: A hash of the various configuration parameters for the provisioner.
- reclaimPolicy: The behaviour for the backing storage when the PVC is deleted.
  - *Retain* - manual clean-up
  - *Delete* - storage asset deleted by provider

```yaml
kind: StorageClass
apiVersion: storage.k8s.io/v1
metadata:
  name: standard
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
  zones: us-central1-a, us-central1-b
reclaimPolicy: Delete
```

StorageClass

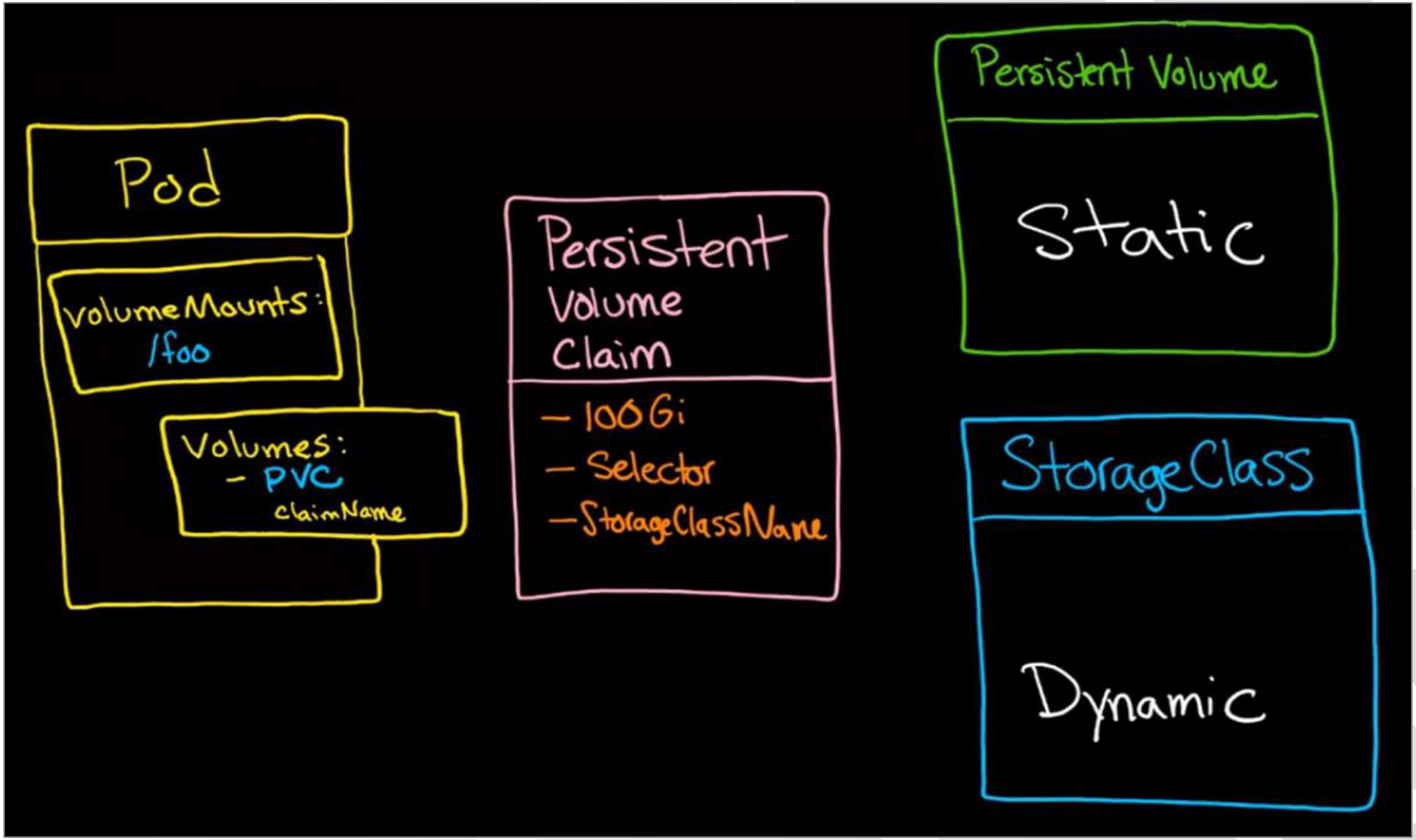
- provisioner: Defines the '*driver*' to be used for provisioning of the external storage.
- parameters: A hash of the various configuration parameters for the provisioner.
- reclaimPolicy: The behaviour for the backing storage when the PVC is deleted.
  - *Retain* - manual clean-up
  - *Delete* - storage asset deleted by provider



Available StorageClasses

- AWSElasticBlockStore
- AzureFile
- AzureDisk
- CephFS
- Cinder
- FC
- Flocker
- GCEPersistentDisk
- Glusterfs
- iSCSI
- Quobyte
- NFS
- RBD
- VsphereVolume
- PortworxVolume
- ScaleIO
- StorageOS
- Local

Internal Provisioner



Questions

Noam Amrani

Module 18: StatefulSets

Kubernetes Workshop

https://www.linkedin.com/in/noamamrani

Agenda

- StatefulSets
- Lab 11: Working with StatefulSets

StatefulSets

- StatefulSet is the workload API object used to manage stateful applications
- Manages the deployment and scaling of a set of Pods , and provides guarantees about the ordering and uniqueness of these Pods.
- StatefulSets are valuable for applications that require one or more of the following.
  - Stable, unique network identifiers.
  - Stable, persistent storage.
  - Ordered, graceful deployment and scaling.
  - Ordered, automated rolling updates.

StatefulSets

```yaml
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  template:
    metadata:
      labels:
        app: nginx # has to match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
      - name: nginx
        image: k8s.gcr.io/nginx-slim:0.8
        ports:
        - containerPort: 80
          name: web
        volumeMounts:
        - name: www
          mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
  - metadata:
      name: www
    spec:
      accessModes: [ "ReadWriteOnce" ]
      storageClassName: "my-storage-class"
      resources:
        requests:
          storage: 1Gi
```

Deployment and Scaling Guaranties

- For a StatefulSet with N replicas, when Pods are being deployed, they are created sequentially, in order from {0..N-1}.
- When Pods are being deleted, they are terminated in reverse order, from {N-1..0}.
- Before a scaling operation is applied to a Pod, all of its predecessors must be Running and Ready.
- Before a Pod is terminated, all of its successors must be completely shutdown.

Updating Strategies

- **OnDelete:** does not automatically delete and recreate Pods when the object's configuration is changed. Instead, you must manually delete the old Pods to cause the controller to create updated Pods.

- **RollingUpdate:** automatically deletes and recreates Pods when the object's configuration is changed. New Pods must be in Running and Ready states before their predecessors are deleted. With this strategy, changing the Pod specification automatically triggers a rollout. This is the **default update strategy** for StatefulSets.

```yaml
# statefulset.yaml
---
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  serviceName: "nginx"
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  updateStrategy:
    type: RollingUpdate
  template:
```

Questions

Lab 11: Working with StatefulSets

Lab

https://gitlab.com/sela-kubernetes-workshop/lab-11