ROITRAINING
MAXIMIZE YOUR TRAINING INVESTMENT

Kubernetes Essentials

ROITRAINING

MAXIMIZE YOUR TRAINING INVESTMENT

The following course materials are copyright protected materials. They may not be reproduced or distributed and may only be used by students attending the *Kubernetes Essentials* course.

Welcome!

- ROI leads the industry in designing and delivering customized technology and management training solutions
  - Our expertise is in supporting large scale training programs for the global banking community; e.g., cloud and new hire training
- Meet your instructor
  - Name
  - Background

Sample Public Cloud Provider Training Offerings

AWS Technical Essentials (1 day)

Deploying Infrastructure on AWS (3 days)

Developing for AWS (2 days)

AWS Security (3 days)

DevOps Engineering (3 days)

Big Data on AWS (3 days)

Architecting with Google Compute Engine (3 days)

Architecting with Google Kubernetes Engine (3 days)

Developing Applications w/ Google Cloud (3 day)

Data Engineering on GCP (4 days)

Security in Google Cloud (3 days)

Machine Learning on Google Cloud (5 days)

Microsoft Azure Administrator (4 days)

Microsoft Azure Security Technologies (4 days)

Developing Solutions for Microsoft Azure (5 days)

Designing and Implementing a Data Science Solution on Azure (3 days)

Course Objectives

- Introduction to running Kubernetes clusters in the cloud
- Quickly deploying autoscaled, load-balanced applications to Kubernetes
- Implement a DevOps automated deployment with a CI/CD pipeline
- Gain hands-on experience with Kubernetes

Qwiklabs

- This course leverages Qwiklabs to provide hands-on tutorials
  - You will be provided with Qwiklabs credits
  - We will perform one lab during the course
  - These credits will be available for approximately 2 weeks to perform additional labs after the course
- You will need to create a free Qwiklabs account:
  - Go to https://www.qwiklabs.com/
  - Click the Join button
  - Follow the prompts to create new account
  - You may receive an email to confirm your account
  - Once logged in, click on your profile (top-right corner), and verify you have credits

Agenda

Microservice Architecture

Kubernetes

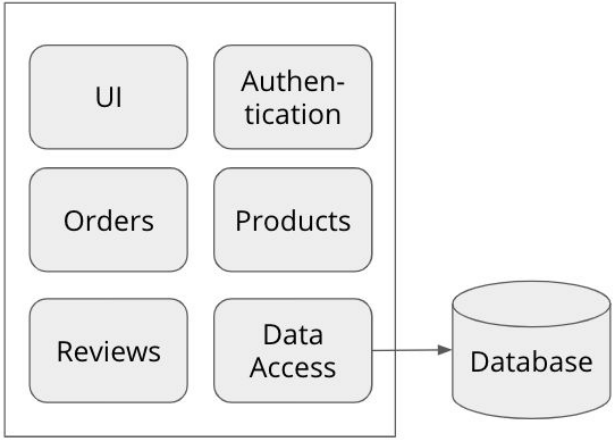
DevOps Automation (CI/CD)

Microservice Architecture

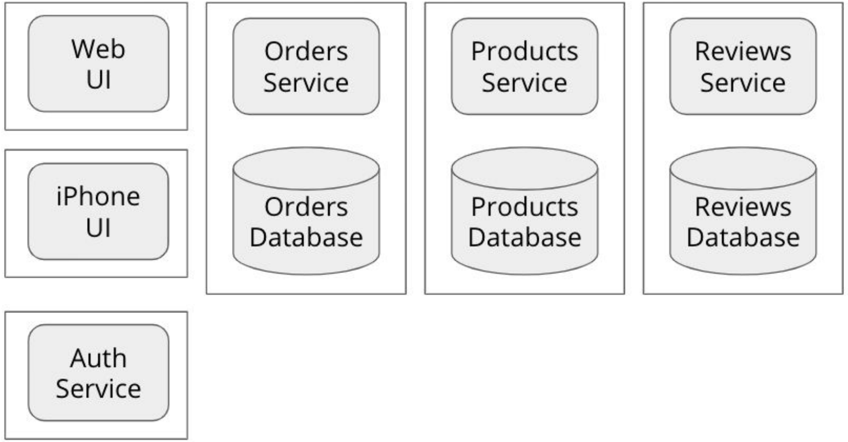
- Divide a large program into a number of smaller, independent services
  - Each service is programmed, deployed, and run separately
- Advantages of microservices include:
  - Reduced risk when deploying new versions
    - Software can be changed and re-deployed without fear of compromising the core application
  - Services scale independently to optimize use of infrastructure
  - Easier to innovate and add new features
  - Can use different languages and frameworks for different services

Monolithic vs. Microservice Architecture

- Monolithic applications implement all features in a single code base
  - Single database for all data



- Microservices break large programs into a number of smaller services
  - Each service manages its own data



Recognizing Microservice Boundaries

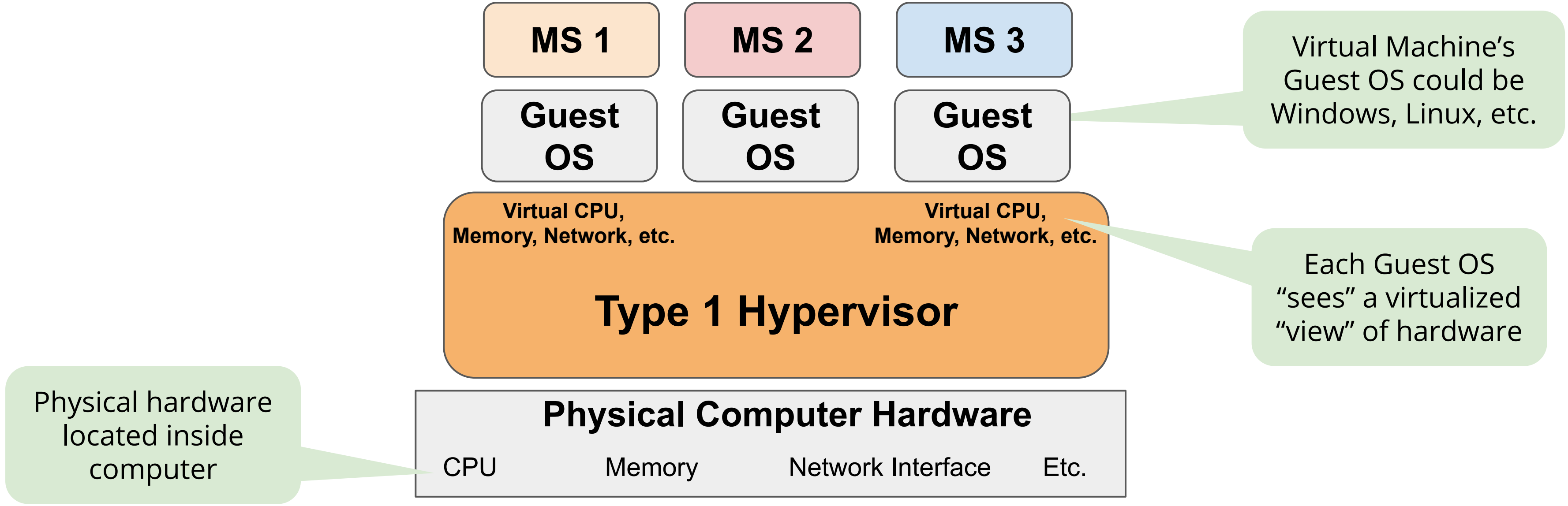
- Defining service boundaries is a matter of decomposition
    - Deciding what pieces of the larger application should be separated
- Typically done by features to minimize dependencies across services
    - Balance service, interest rate service, amortization service, underwriting service, etc.
    - Reviews service, Order Processing service, Product Catalog service, etc.
- Sometimes services are organized by architectural layer
    - Web, iPhone, and Android user interfaces
    - Services that provide access to shared data
- Some services provide shared behavior to the others
    - For example, authentication service

Virtual Machine-Based Virtualization

- Virtual machine-based virtualization is the ability to virtualize hardware
  - Allows resources of a single physical computer to be shared among more than one virtual machine
  - Each virtual machine (VM) has its own virtual set of resources
  - Each VM provides functionality to execute entire operating systems (OS)
- Software that manages this virtualization is known as a hypervisor
  - Type 1 or native hypervisor
  - Type 2 or hosted hypervisor (not discussed here)

Type 1 (Native) Hypervisor

- A type 1 hypervisor runs directly on the machine hardware
  - Each VM runs in complete isolation from other virtual machines
  - Hypervisor exposes a subset of the actual hardware



MS 1
MS 2
MS 3
Guest OS
Guest OS
Guest OS
Virtual CPU, Memory, Network, etc.
Virtual CPU, Memory, Network, etc.
Type 1 Hypervisor
Physical Computer Hardware
CPU
Memory
Network Interface
Etc.

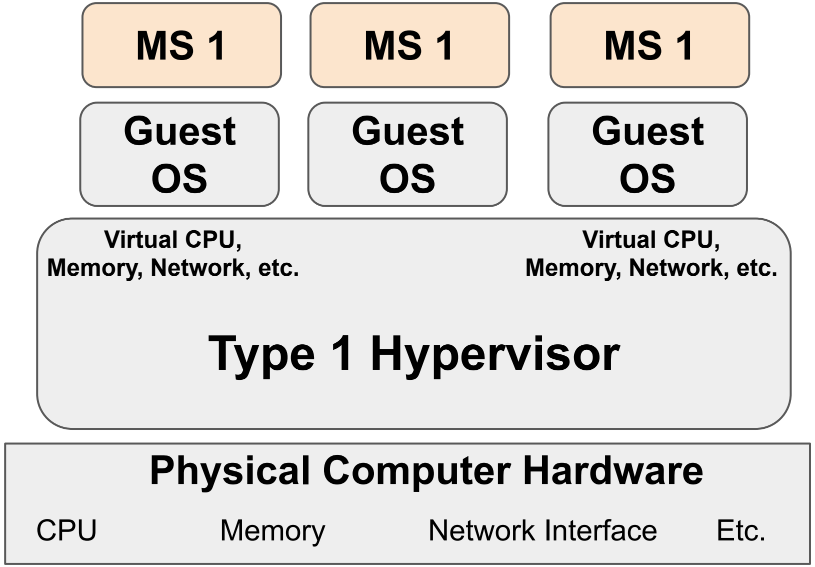
Virtual Machine's Guest OS could be Windows, Linux, etc.

Each Guest OS "sees" a virtualized "view" of hardware

Physical hardware located inside computer

Handling Demand with VMs

- It is standard practice to isolate services on their own virtual machines
    - And run multiple virtual machines for application availability or load
    - Has costs in boot time (minutes) and resources (Gigabytes)

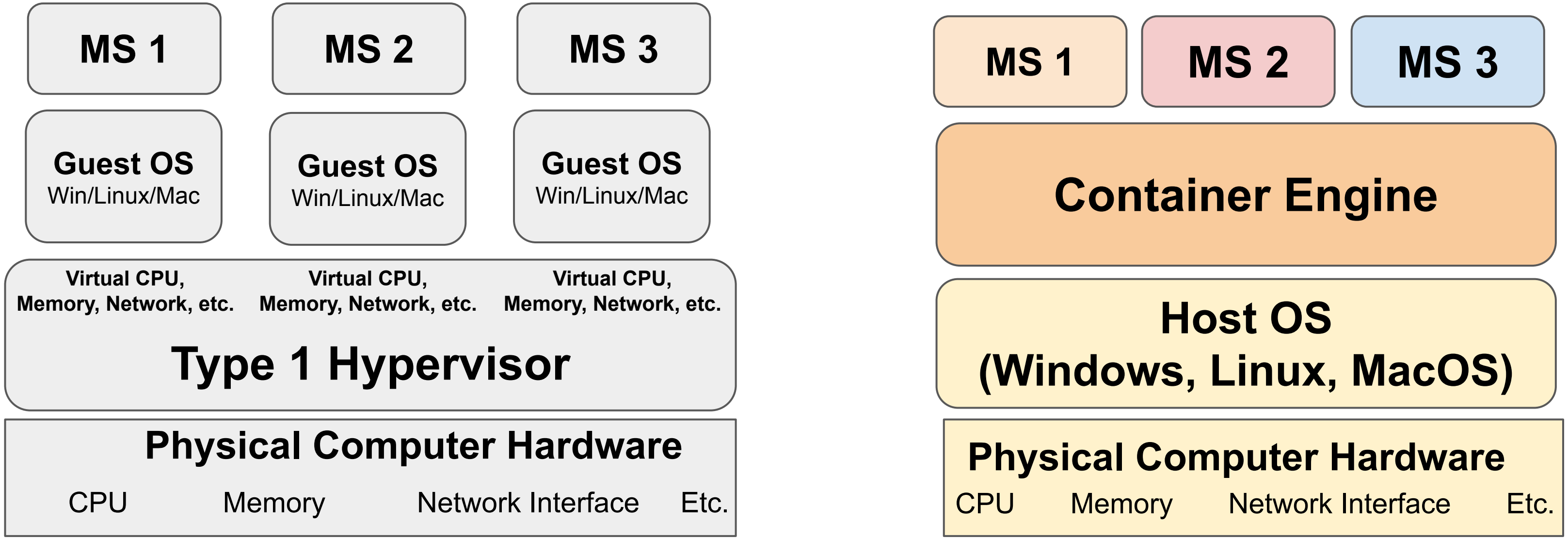

MS 1
MS 1
MS 1
Guest OS
Guest OS
Guest OS
Virtual CPU, Memory, Network, etc.
Virtual CPU, Memory, Network, etc.
Type 1 Hypervisor
Physical Computer Hardware
CPU
Memory
Network Interface
Etc.

Container-Based Virtualization

- Container-based virtualization is the ability to virtualize the OS kernel
  - Multiple isolated systems, called containers, access a single OS kernel
- Everything required for a piece of software to run is packaged into an isolated container
  - An abstraction layer of the hardware and OS
  - Like an invisible box with configurable access to isolated partitions of file system, RAM, and networking
  - Contain software dependencies, libraries, and settings required to make the software work
- Can deploy and run service without launching an entire VM for each service

Virtual Machines vs. Containers

- Applications hosted on virtual machines require significantly more resources
    - Each application on a VM requires its own operating system
    - In essence, container engines are virtualizing the OS

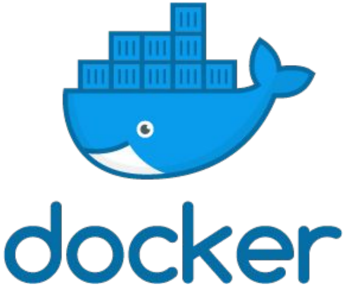


Advantages of Containers

- More efficient
  - Require less resources, less overhead
- Better performance
  - Containers can be very fast to start
    - No VM to boot, which takes tens of seconds or even minutes
- Provides an agile environment
  - Can improve portability across systems
- Encapsulates application dependencies
  - Facilitates a microservices/DevOps approach
  - Works well with a continuous integration strategy

Docker

- Docker is one of the most popular container environments
  - www.docker.com
  - Docker Engine client runs natively on Linux, MacOS, and Windows



Agenda

Microservice Architecture

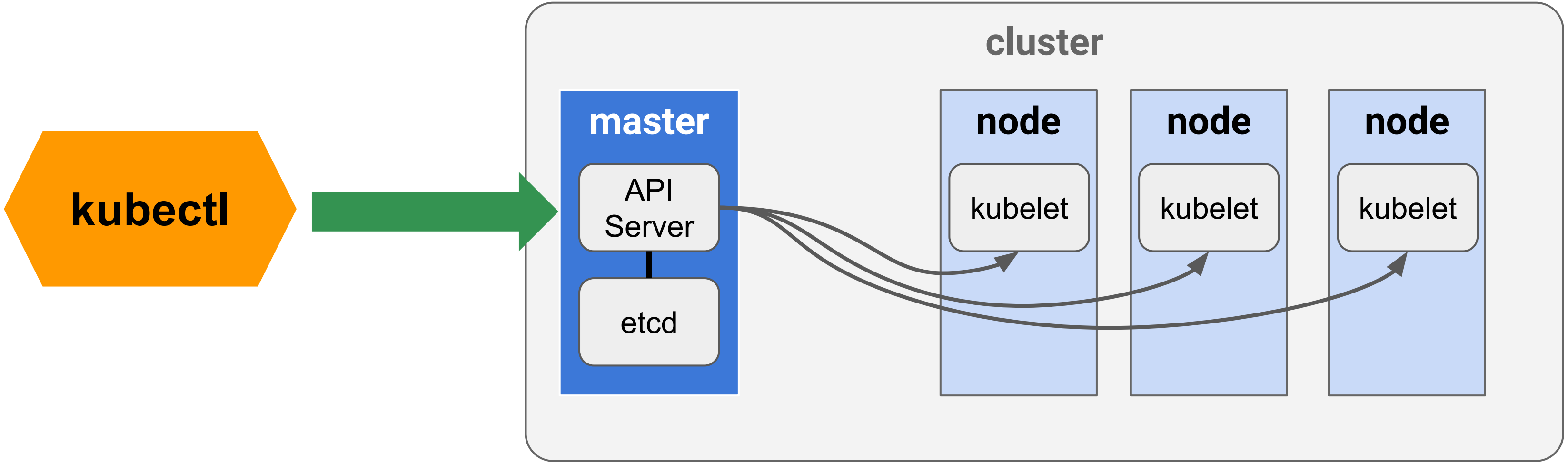
Kubernetes

DevOps Automation (CI/CD)

Kubernetes

- Open-source container orchestration system
  - Extremely popular and active open-source project
  - See: https://kubernetes.io/
- Originally developed by Google to run Google's data centers
  - Designed to operate at Google scale
  - Proven and tested running Google's applications
- Wide support
  - Used on Google Cloud when using Google Kubernetes Engine (GKE)
  - Supported by Microsoft in Azure Container Service (AKS)
  - Supported by AWS with Elastic Kubernetes Service (EKS)
  - Red Hat OpenShift
  - And more ...

Kubernetes Clusters

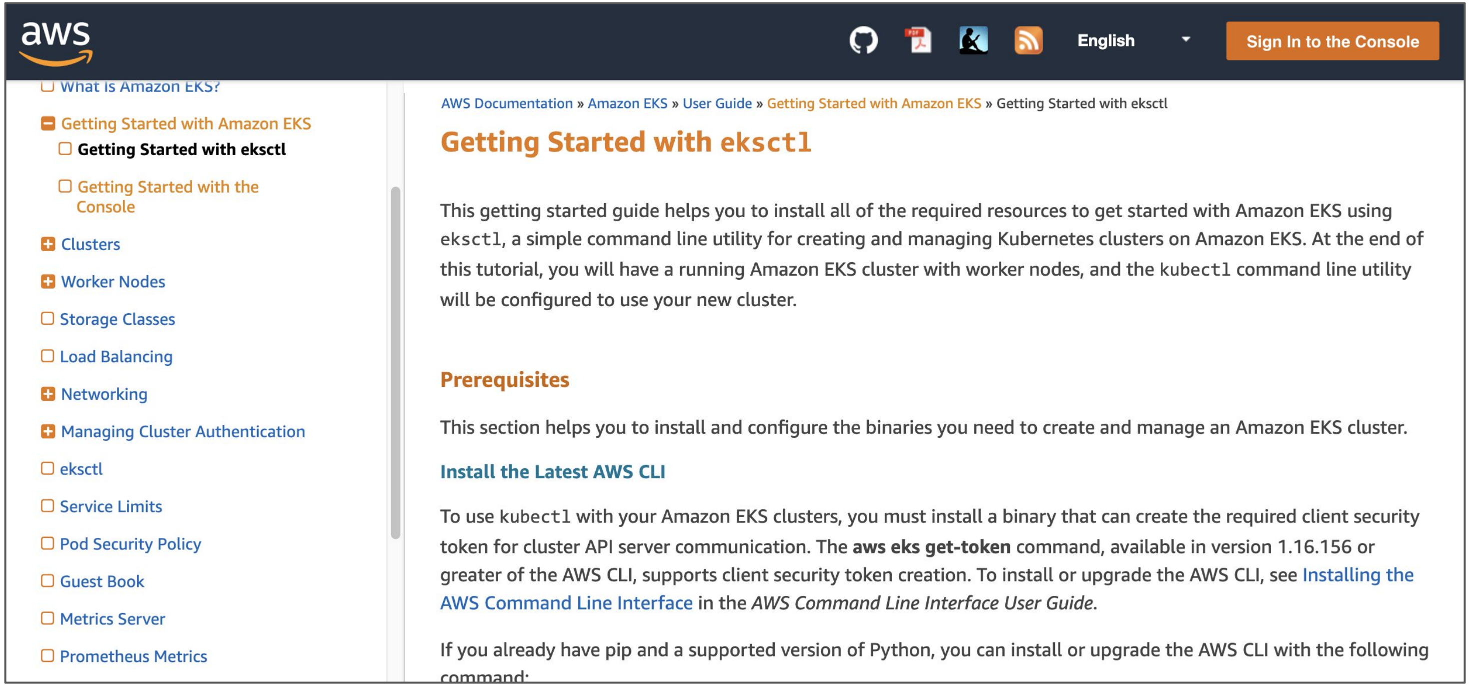


Managed Kubernetes Services

- Many cloud platforms provide a managed Kubernetes service
  - Automates cluster creation and maintenance
  - Clusters are implemented as a collection of cloud VMs
  - Very simple creation using the web console or command line

Tutorial: Getting Started with AWS Elastic Kubernetes Service (EKS)

- https://docs.aws.amazon.com/eks/latest/userguide/getting-started-eksctl.html



- ☐ What is Amazon EKS?
- ⊟ Getting Started with Amazon EKS
  - ☐ **Getting Started with eksctl**
  - ☐ Getting Started with the Console
- ⊞ Clusters
- ⊞ Worker Nodes
- ☐ Storage Classes
- ☐ Load Balancing
- ⊞ Networking
- ⊞ Managing Cluster Authentication
- ☐ eksctl
- ☐ Service Limits
- ☐ Pod Security Policy
- ☐ Guest Book
- ☐ Metrics Server
- ☐ Prometheus Metrics

AWS Documentation » Amazon EKS » User Guide » Getting Started with Amazon EKS » Getting Started with eksctl

Getting Started with eksctl

This getting started guide helps you to install all of the required resources to get started with Amazon EKS using `eksctl`, a simple command line utility for creating and managing Kubernetes clusters on Amazon EKS. At the end of this tutorial, you will have a running Amazon EKS cluster with worker nodes, and the `kubectl` command line utility will be configured to use your new cluster.

Prerequisites

This section helps you to install and configure the binaries you need to create and manage an Amazon EKS cluster.

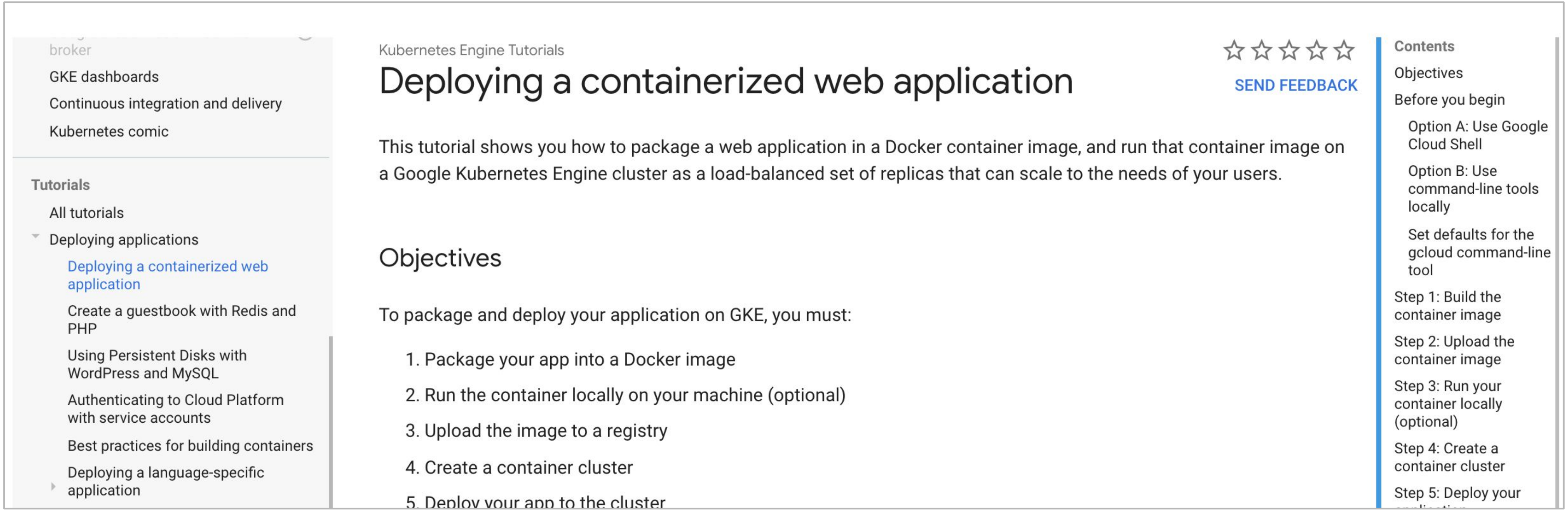
Install the Latest AWS CLI

To use `kubectl` with your Amazon EKS clusters, you must install a binary that can create the required client security token for cluster API server communication. The **aws eks get-token** command, available in version 1.16.156 or greater of the AWS CLI, supports client security token creation. To install or upgrade the AWS CLI, see Installing the AWS Command Line Interface in the *AWS Command Line Interface User Guide*.

If you already have pip and a supported version of Python, you can install or upgrade the AWS CLI with the following command:

Tutorial: Getting Started with Google Kubernetes Engine (GKE)

- https://cloud.google.com/kubernetes-engine/docs/tutorials/hello-app



broker

GKE dashboards

Continuous integration and delivery

Kubernetes comic

Tutorials

All tutorials

Deploying applications

Deploying a containerized web application

Create a guestbook with Redis and PHP

Using Persistent Disks with WordPress and MySQL

Authenticating to Cloud Platform with service accounts

Best practices for building containers

Deploying a language-specific application

Kubernetes Engine Tutorials

☆☆☆☆☆

SEND FEEDBACK

Deploying a containerized web application

This tutorial shows you how to package a web application in a Docker container image, and run that container image on a Google Kubernetes Engine cluster as a load-balanced set of replicas that can scale to the needs of your users.

Objectives

To package and deploy your application on GKE, you must:

1. Package your app into a Docker image
2. Run the container locally on your machine (optional)
3. Upload the image to a registry
4. Create a container cluster
5. Deploy your app to the cluster

Contents

Objectives

Before you begin

Option A: Use Google Cloud Shell

Option B: Use command-line tools locally

Set defaults for the gcloud command-line tool

Step 1: Build the container image

Step 2: Upload the container image

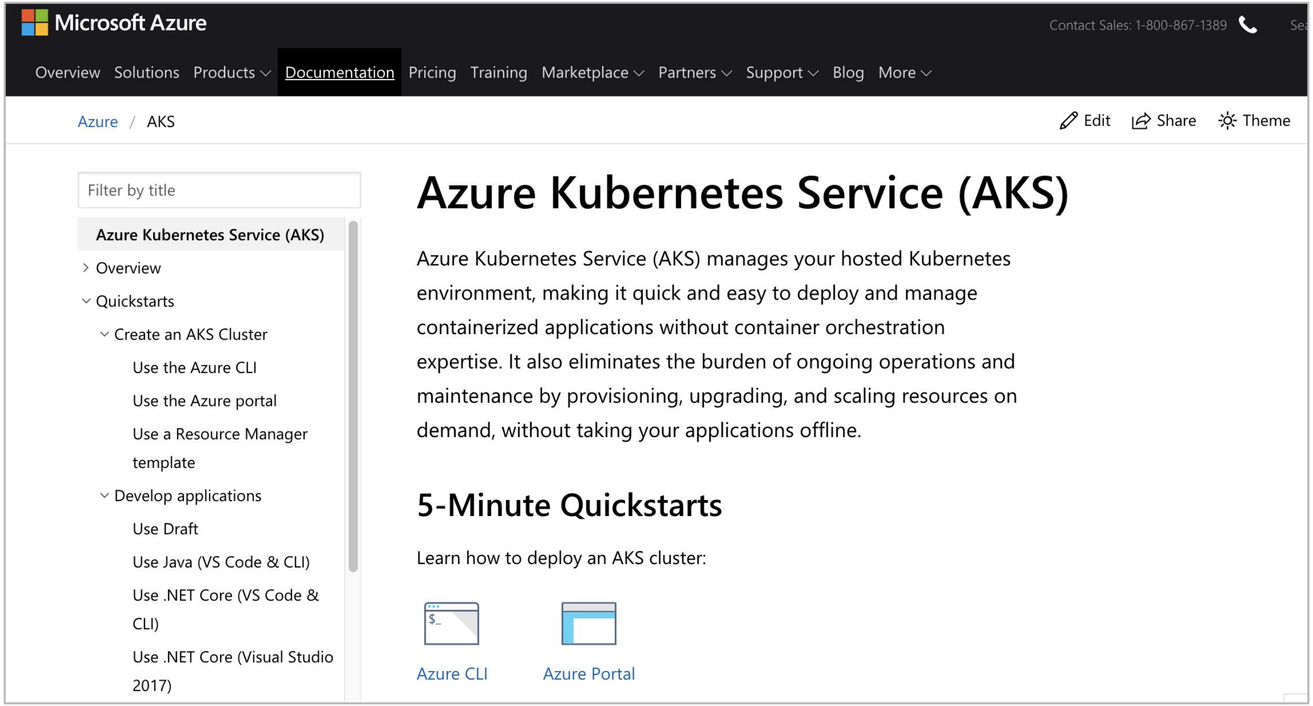
Step 3: Run your container locally (optional)

Step 4: Create a container cluster

Step 5: Deploy your

Tutorial: Getting Started with Azure Kubernetes Service (AKS)

- https://docs.microsoft.com/en-us/azure/aks/



Azure Kubernetes Service (AKS)

Azure Kubernetes Service (AKS) manages your hosted Kubernetes environment, making it quick and easy to deploy and manage containerized applications without container orchestration expertise. It also eliminates the burden of ongoing operations and maintenance by provisioning, upgrading, and scaling resources on demand, without taking your applications offline.

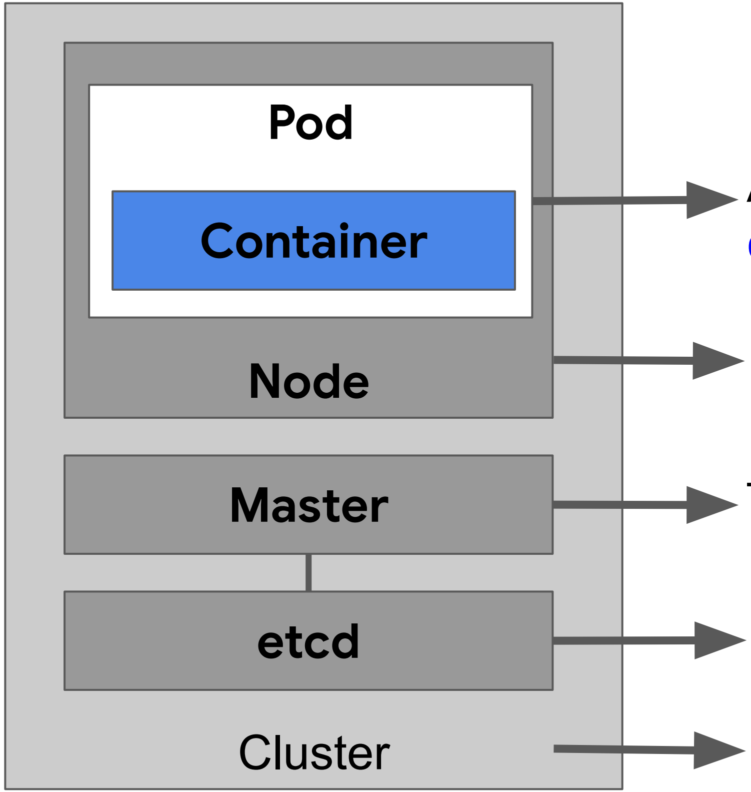
5-Minute Quickstarts

Learn how to deploy an AKS cluster:

Kubernetes Terms

- Clusters are collections of machines that will run containers
  - Each machine is a node in the cluster
- Pods are the smallest unit of deployment
  - Usually pods represent a single container
- Deployments are configurations that define service resources
- Replication controllers are used to create multiple instances of a pod
  - Guarantee pods are healthy and the right number exist
- Load balancers route requests to pods
- Autoscalers monitor load and turn pods on or off

Kubernetes Components



A **pod** is a collection of one or more **containers**, deployed on nodes

Nodes are machines that run the containers, talk to the master

The **master** controls the cluster

etcd is for storing cluster state

A **cluster** is a set of nodes

Kubernetes CLI

- Kubernetes is automated with a combination of CLI commands and configuration code
  - `kubectl` is the Kubernetes CLI

kubectl Command	**Description**
kubectl get nodes	Show information about the cluster nodes.
kubectl apply -f <file-name>	Apply a configuration file
kubectl get pods	Show the running pods.
kubectl get services	Show running services along with addresses and ports.
kubectl delete [name] kubectl delete deployments/spaceinvaders	Delete specified resources.
kubectl exec [options] [pod-name] kubectl exec -it spaceinvaders-55c88695bb-bcxb2 -- /bin/bash	Execute a command in the container (the example runs a bash shell).

Deployments Combine Pods with Replica Sets

```
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: devops-deployment
  labels:
    <Some code omitted to save space>
spec:
  replicas: 3
  selector:
    <Some code omitted to save space>
  template:
    <Some code omitted to save space>
    spec:
      containers:
      - name: devops-demo
        image: drehnstrom/devops-demo:latest
        ports:
        - containerPort: 8080
```

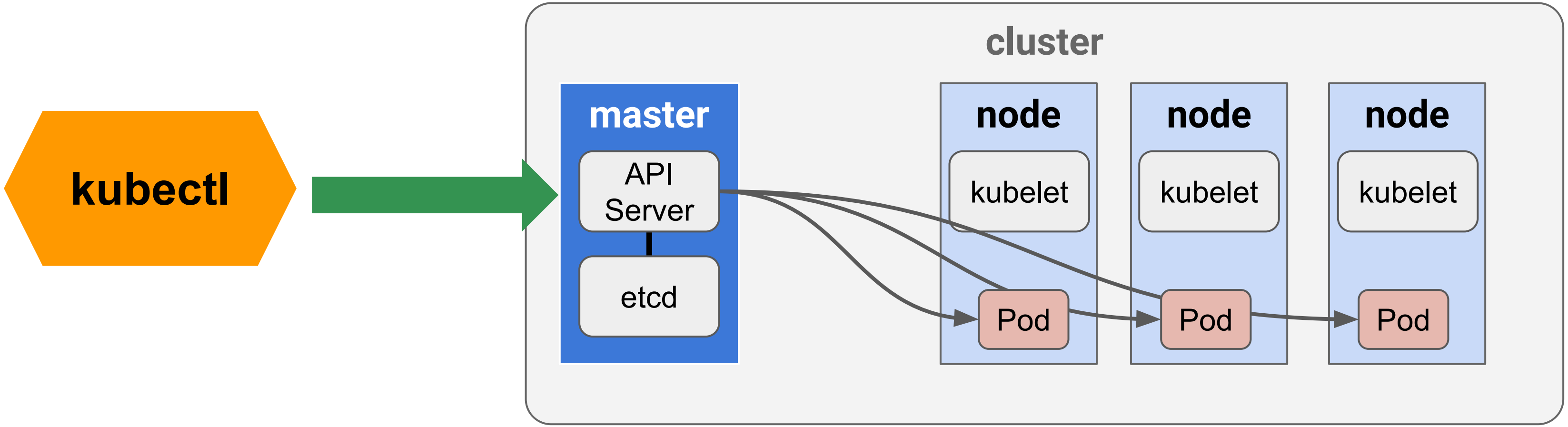
Kind of resource: Deployment

We want 3 replicas of the pod

This spec defines the pod. The same as the Pod configuration.

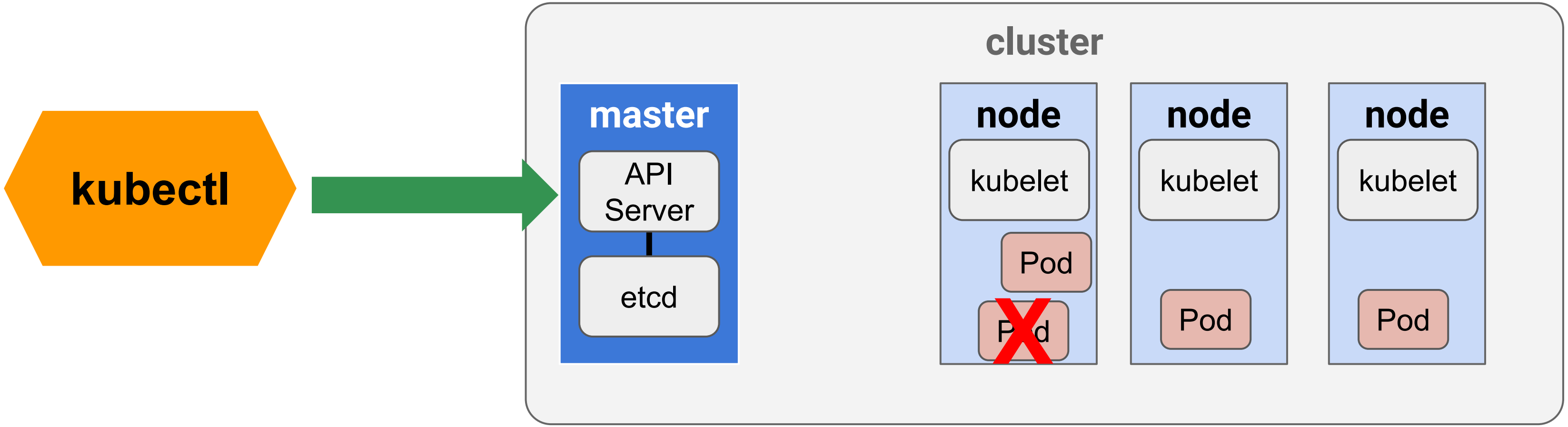
Kubernetes Deployments

- Use kubectl to send the config file to the master
  - The master then decides how to deploy the pods



Kubernetes Replica Set

- A replica set ensures the correct number of pods is running
  - And replaces any pod that fails



Creating a Deployment from a Configuration File

- Deploy a service based on a configuration file

```
kubectl apply -f kubernetes-config.yaml
```

- Show the running pods

```
kubectl get pods
```

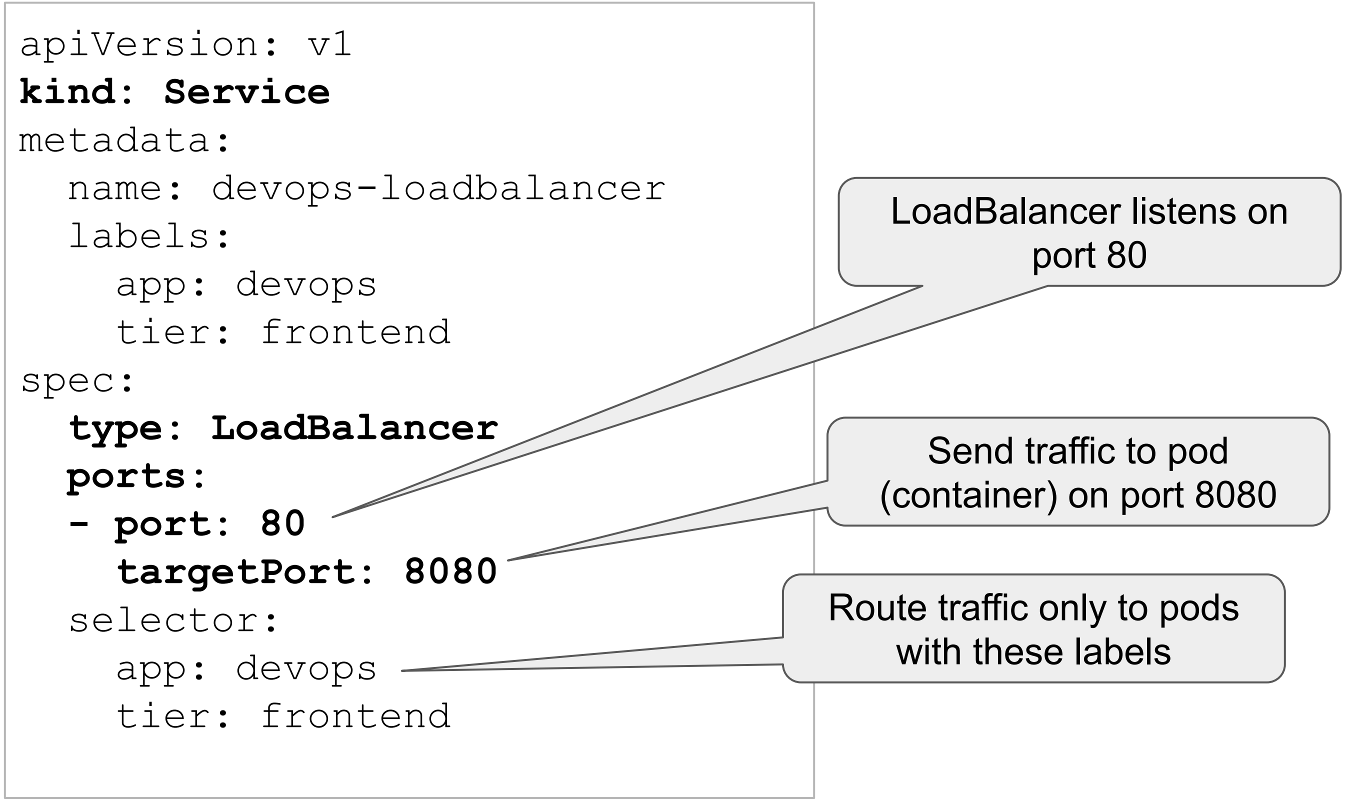
- Show all the deployments

```
kubectl get deployments
```

- Show details of a deployment

```
kubectl describe deployments devops-deployment
```

Adding the Load Balancer to Configuration



```
apiVersion: v1
kind: Service
metadata:
  name: devops-loadbalancer
  labels:
    app: devops
    tier: frontend
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: devops
    tier: frontend
```

LoadBalancer listens on port 80

Send traffic to pod (container) on port 8080

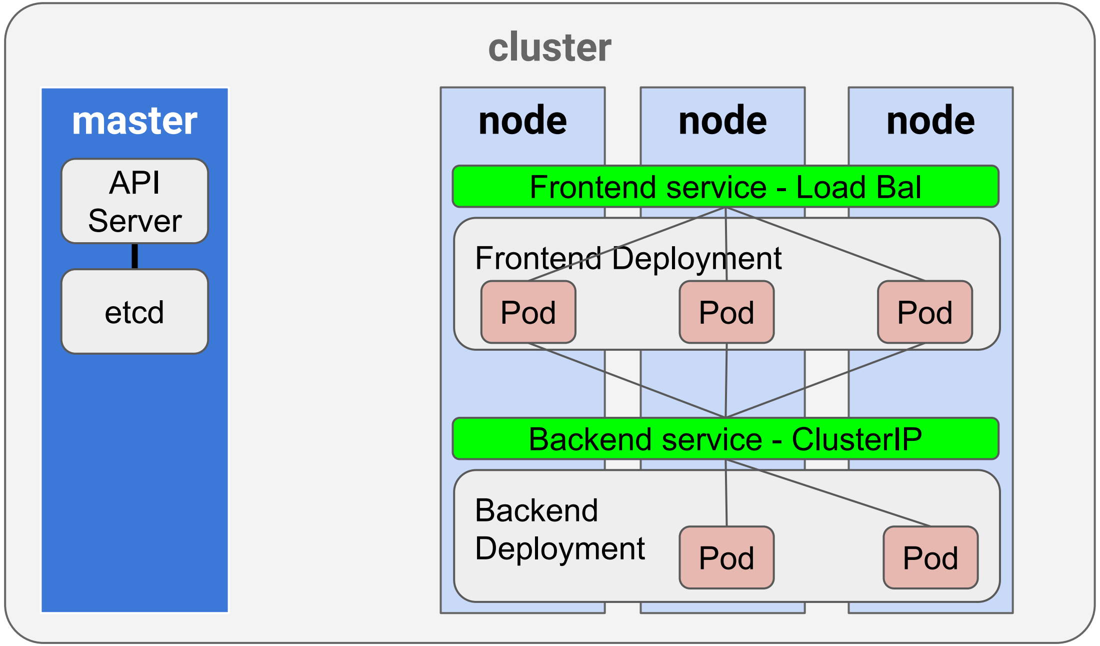
Route traffic only to pods with these labels

Types of Services

ClusterIP	The default service type. Has only an internal IP address that is only accessible by other services running inside the cluster.
LoadBalancer	A service that provides an external IP address. In Google Cloud, this is implemented as a TCP load balancer. In AWS, this is implemented as an Elastic Load balancer. Not all Kubernetes deployments would support this type of service. Can be expensive if you have lots of services, which means lots of load balancers.
NodePort	Assigns a port between 30000 and 32767 to nodes in your cluster. When a node is accessed at that port, it routes to your service.

Types of Services

- A single application can have multiple services



Adding the Autoscaler to Configuration

- To dynamically scale up and down, create an autoscaler
  - Specify min and max number of pods and some metric to monitor

```
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: devops-autoscaler
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: devops-deployment
  minReplicas: 3
  maxReplicas: 10
  metrics:
  - type: Resource
    resource:
      name: cpu
      targetAverageUtilization: 60
```

Demo: Kubernetes in Action

- Your instructor will demo Kubernetes in action

Kubernetes Engine: Qwik Start

HANDS-ON EXERCISE

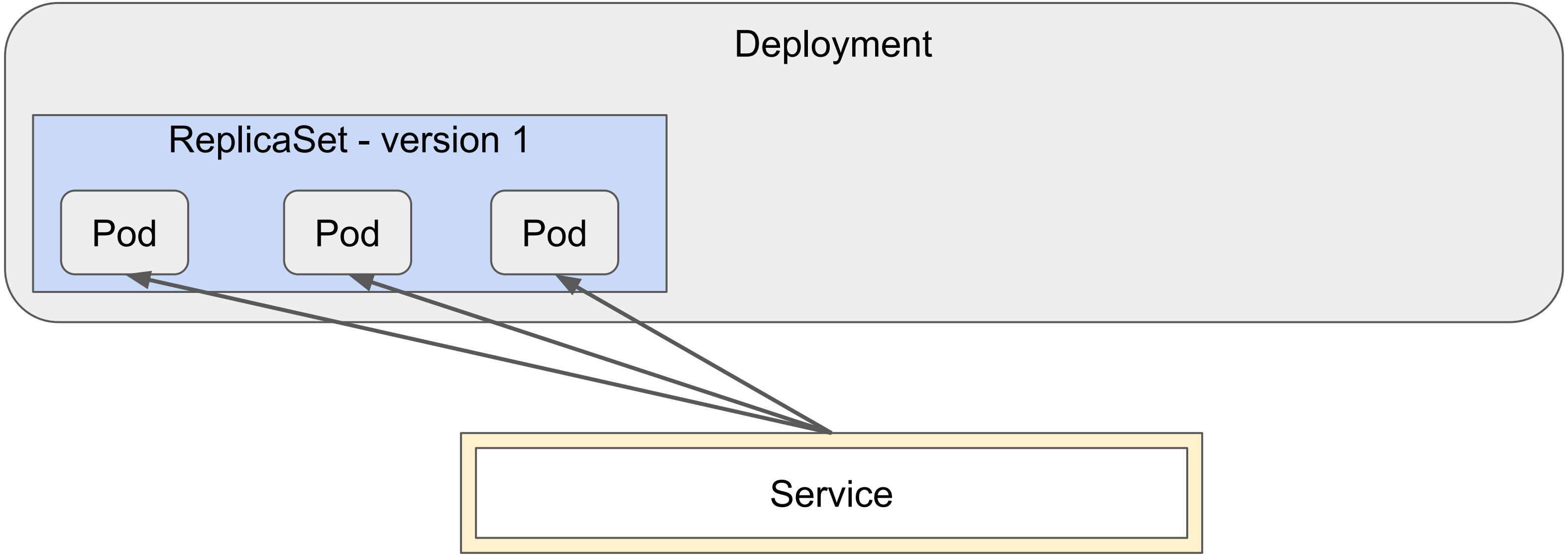
- If you have not already, create a free Qwiklabs account:
  - Go to https://www.qwiklabs.com/
  - Click the Join button
  - Follow the prompts to create new account
  - You may receive an email to confirm your account
  - Once logged in, click on your profile (top-right corner), and verify you have credits
- Perform the Kubernetes Engine: Qwik Start lab:
  - https://www.qwiklabs.com/focuses/878?parent=catalog

Rolling Updates

- Cloud services typically have multiple instances behind a load balancer
  - Multiple replicas of a pod in a Kubernetes cluster
- Rolling deployments update instances incrementally
  - One at a time, 10% at a time, etc.
  - Allows services to be updated with no downtime
- Supported by managed instance groups
- Supported by Kubernetes using the apply command
  - Simply change the container image and re-apply the configuration
  - Can also roll back the update

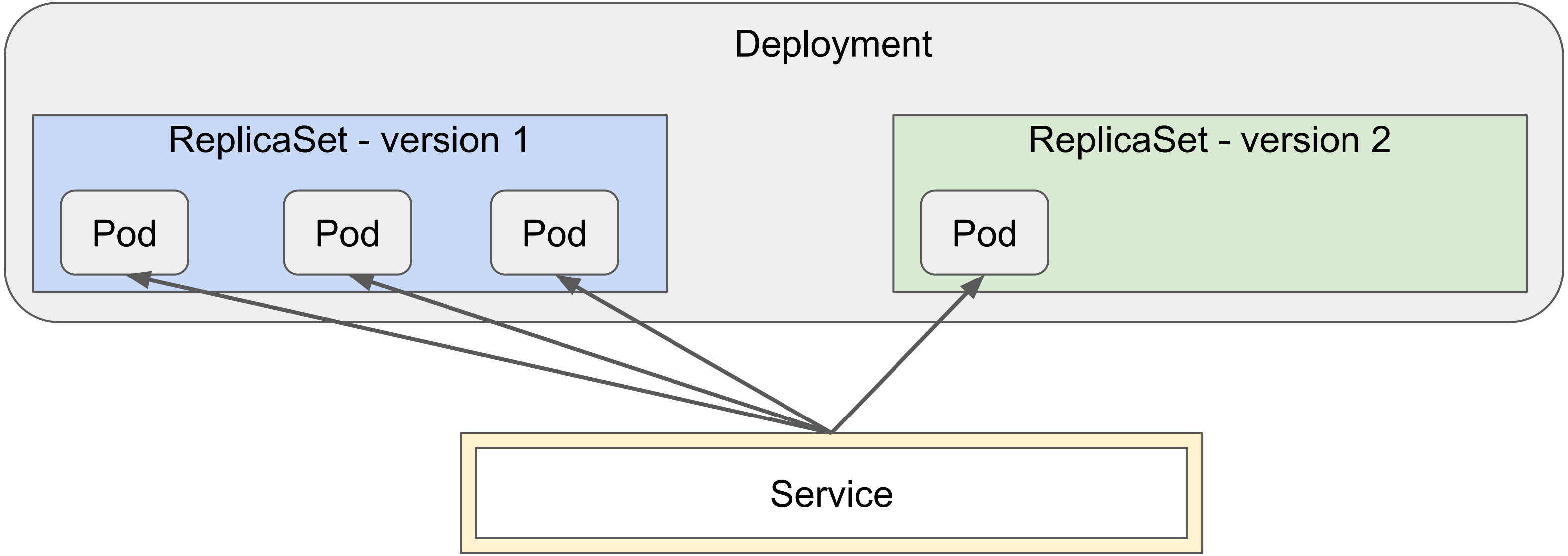
Rolling Update with Kubernetes

- Initially a deployment creates a single ReplicaSet



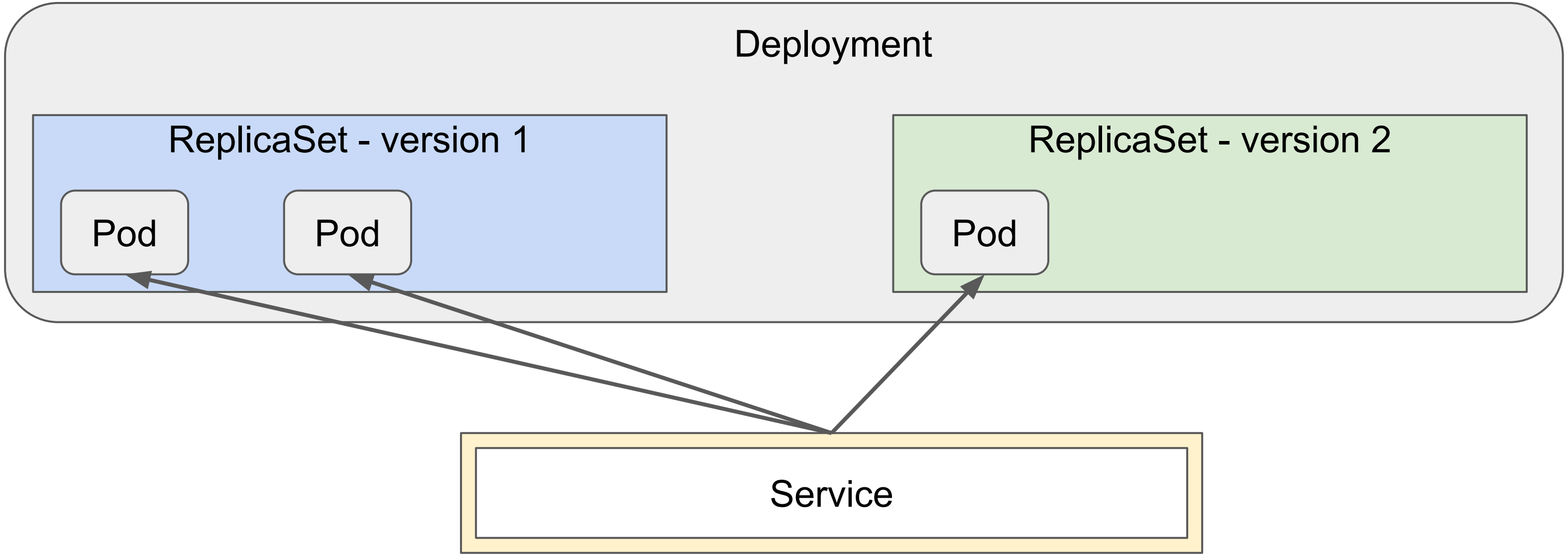
Rolling Update with Kubernetes

- When a container update is applied
  - A new ReplicaSet is created in the same deployment



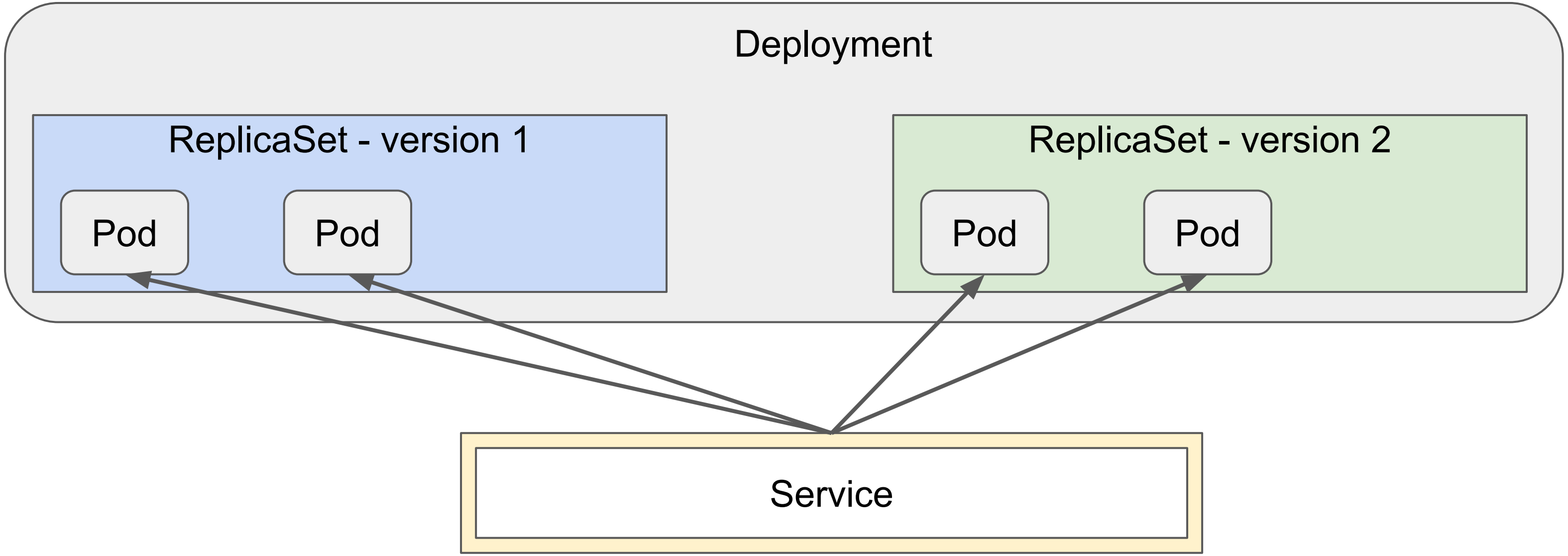
Rolling Update with Kubernetes

- A pod in the old ReplicaSet is deleted



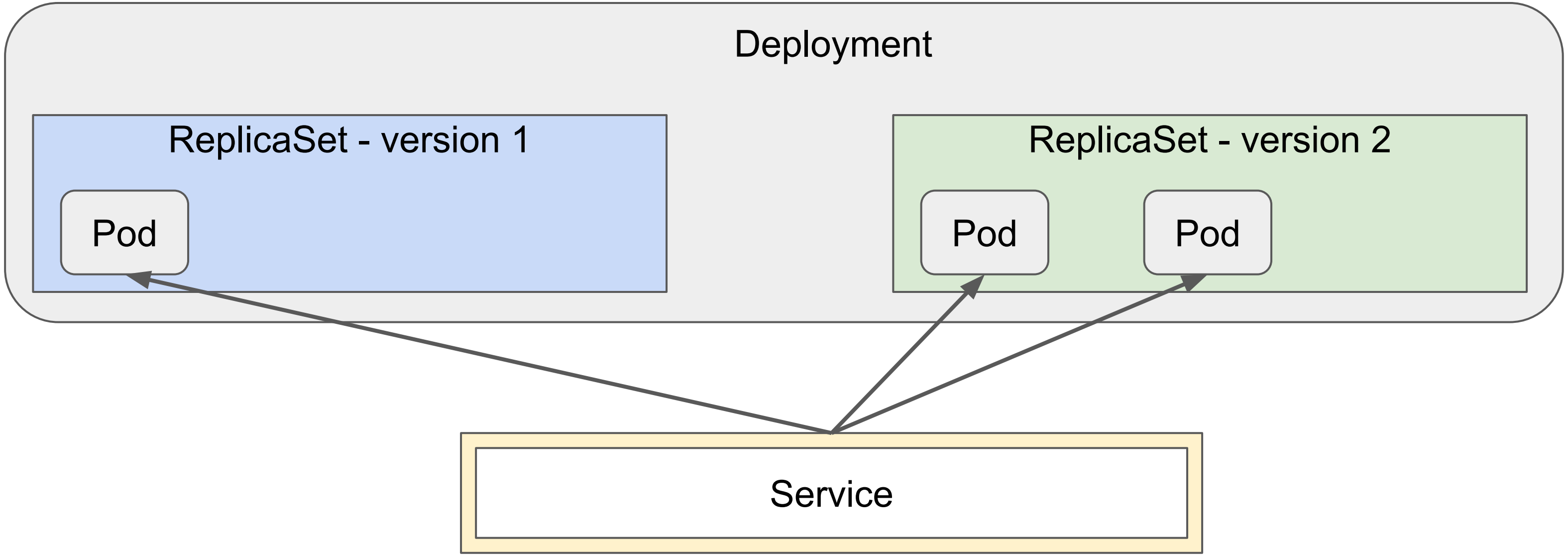
Rolling Update with Kubernetes

- This is repeated



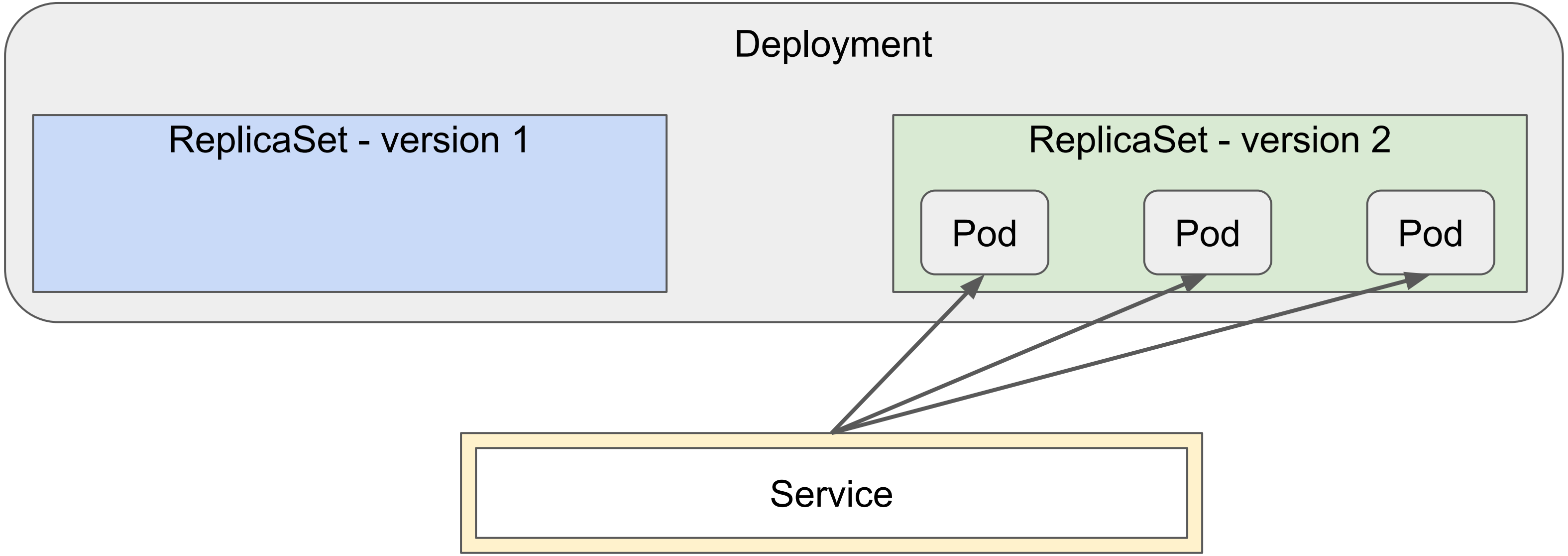
Rolling Update with Kubernetes

- This is repeated



Rolling Update with Kubernetes

- Until all pods have been updated

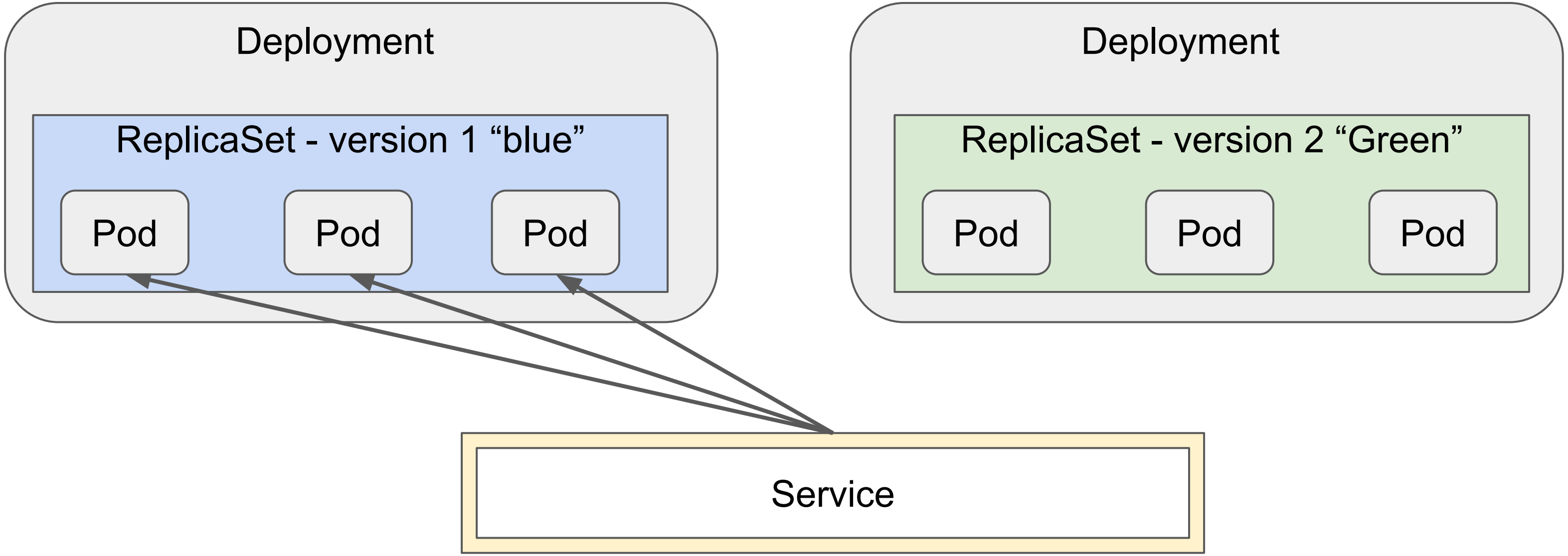


Blue/Green Deployment

- Allows new revisions to be deployed with less risk and no downtime
- There are two copies of the production environment
  - Blue environment is taking requests
  - Green environment is idle
- When deploying a new version:
  - Update new version to green environment leaving the blue environment in place
  - Test the green environment
  - When testing is complete, move the workload to the green environment
  - Green is now blue; can turn the old environment off
- Blue/green deployments also make rolling back to old versions easy

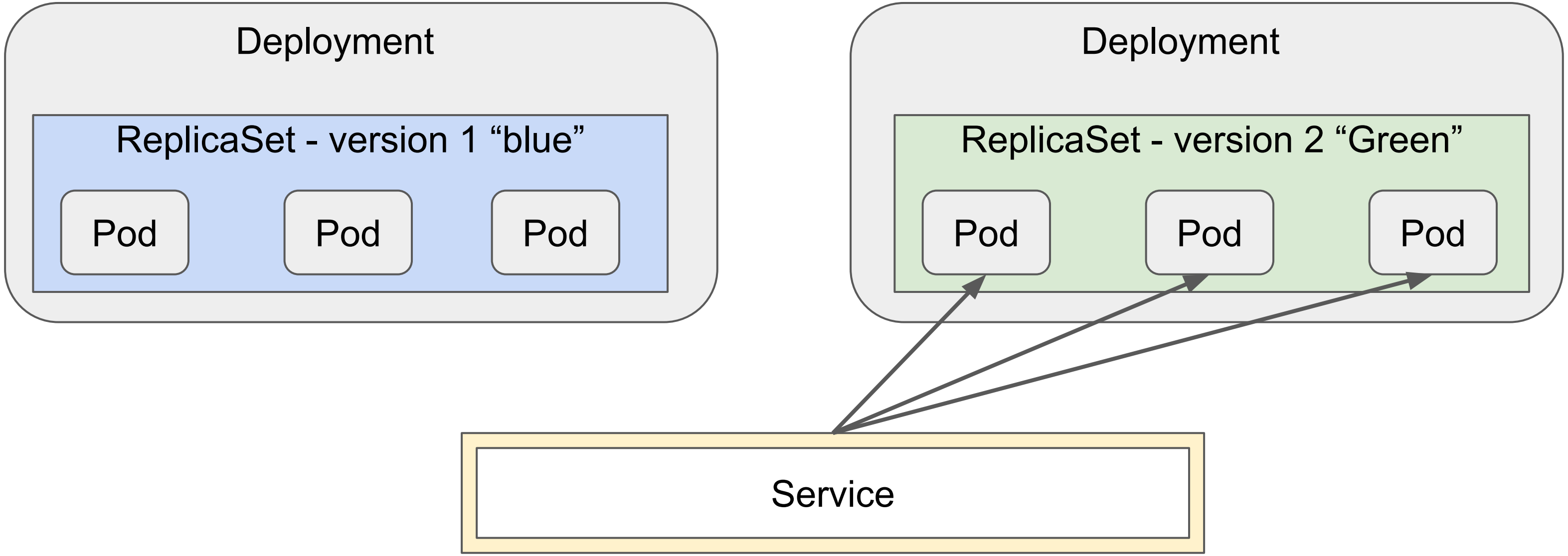
Blue/Green Deployments in Kubernetes

- Service routes all traffic to one version



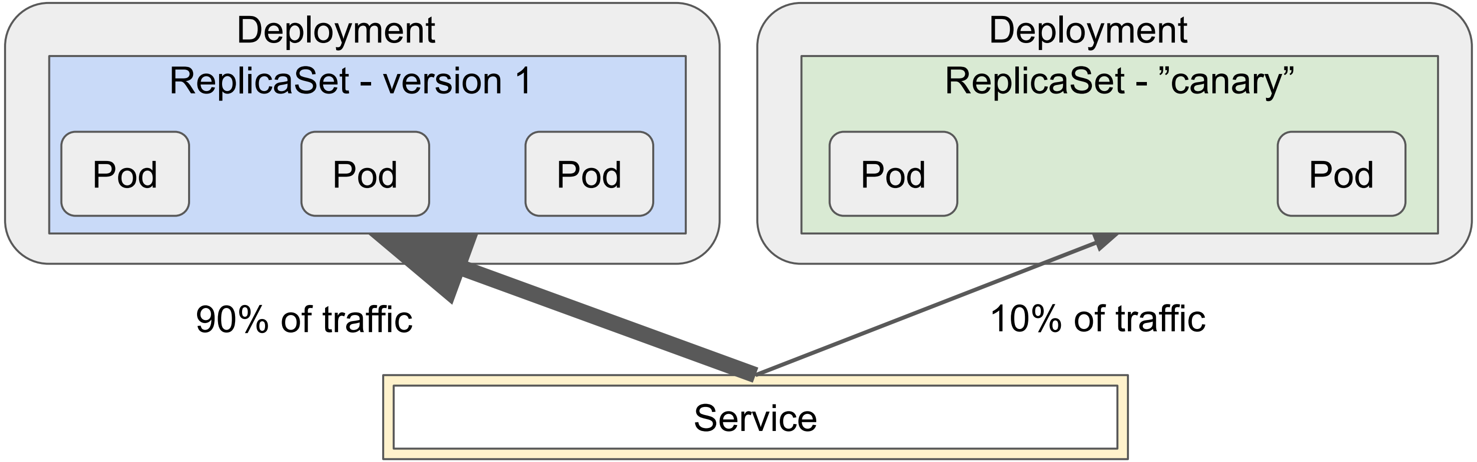
Blue/Green Deployments in Kubernetes

- Can switch the version quickly



Canary Releases

- A new version of a service is put into production alongside old versions
    - A small subset of select traffic is routed to the canary release
- Canary releases help developers know how a new version will perform
- Canary releases are easy to pull back if they fail their testing



Canary Release in Kubernetes

```
apiVersion: v1
kind: Service
metadata:
  name: devops-loadbalancer
  labels:
    app: devops
    tier: frontend
spec:
  type: LoadBalancer
  ports:
  - port: 80
    targetPort: 8080
  selector:
    app: devops
    tier: frontend
```

1. Services use selectors to determine what pods to route traffic to.
2. Create a new deployment with a new container, but use the same labels as the current deployment.
3. If the current deployment has 3 replicas and the new deployment has 1, then the new deployment gets about 25% of the traffic.

More advanced control possible with an Ingress controller or a service mesh (Istio)

- Outside the scope of this discussion

Demo: Blue/Green Deployments

- Your instructor will demo Blue/Green deployments

Agenda

Microservice Architecture

Kubernetes

DevOps Automation (CI/CD)

Continuous Integration/Continuous Deployment

- CI/CD bridges the gaps between development and operation activities
  - CI = Automated code build and test sequence
  - CD = Automating the entire software release process
  - Enforced automation in building, testing and deployment of services
- Pipelines are used to manage the deployment process
  - Steps are defined within the pipeline
  - Some quality metric is used to trigger the next step
  - The final step is deploying a new release into production
- CI/CD pipelines form the backbone of modern day DevOps

Demo: Automated Build and Deployment

- Your instructor will demo automated builds and deployment

Additional Labs for Homework

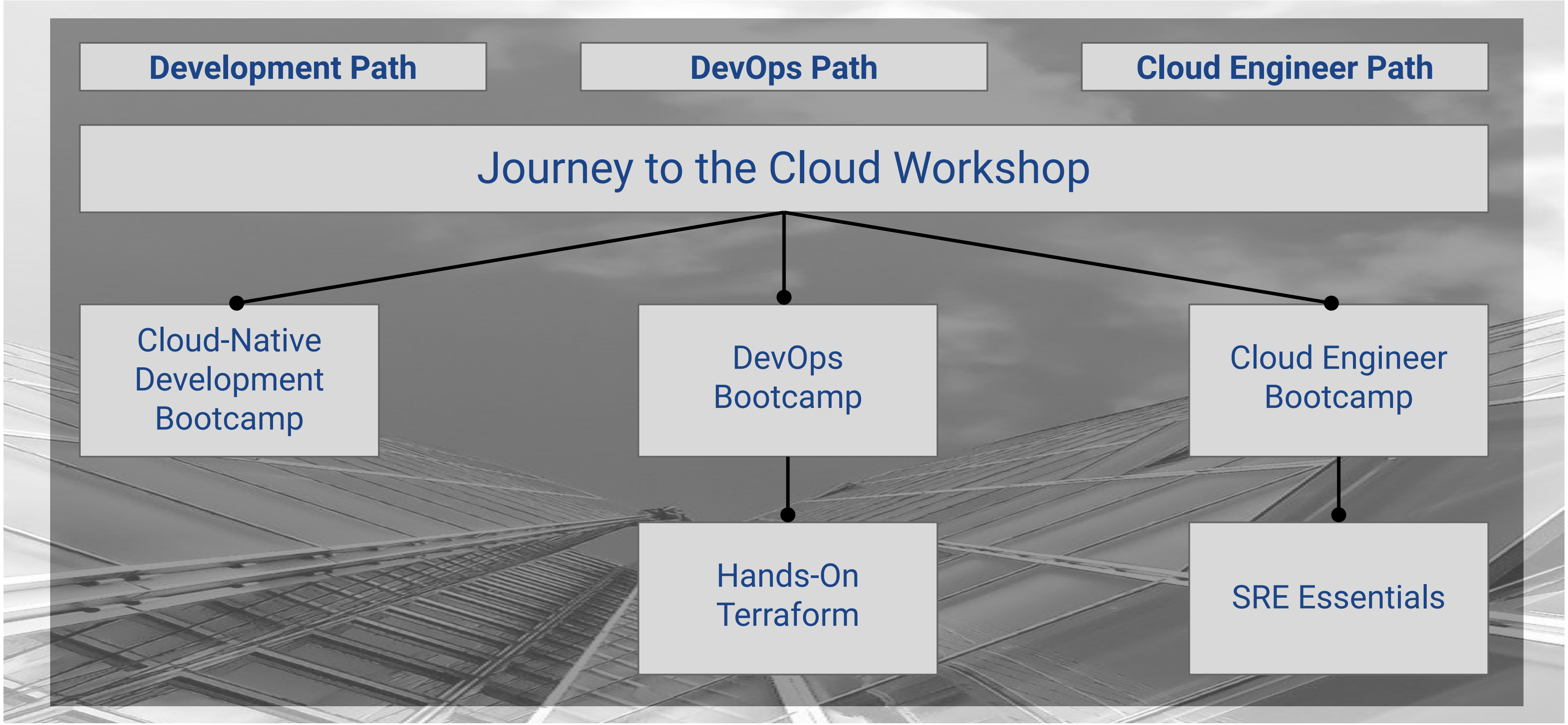
- You have been provided with Qwiklab credits
    - Credits will be available for approximately 2 weeks
- Perform one or both of the following labs for more hands on:
    - Introduction to Docker (to gain experience with containers)
        - https://www.qwiklabs.com/focuses/1029?parent=catalog
    - Orchestrating the Cloud with Kubernetes
        - https://www.qwiklabs.com/focuses/557?parent=catalog

Course Summary

In this course, we have:

- Created a Kubernetes cluster in the cloud and deployed a simple workload
- Quickly deployed autoscaled, load-balanced applications to Kubernetes
- Implemented a DevOps automated deployment with a CI/CD pipeline
- Gained hands-on experience with Kubernetes

Cloud Adoption—What's Next?



Development Path

DevOps Path

Cloud Engineer Path

Journey to the Cloud Workshop

Cloud-Native Development Bootcamp

DevOps Bootcamp

Cloud Engineer Bootcamp

Hands-On Terraform

SRE Essentials

Banking Customer Case Study: Cloud Migration

2,200
Learners trained in 2020

533
Learners trained in APAC

159
Learners trained in EMEA

111
Learners trained in US

752
Cloud architect learners

28
Data engineer learners

23
App developer learners

DESIGN PROJECT REVIEWS

83 design review sessions delivered

27 review groups ran concurrently in 1 week and in 4 time zones

5:1 student to instructor ratio

SCALABILITY

2 ROI trainers dedicated to program

4 bootcamps ran concurrently in 1 week

125 trained in 2 time zones in 1 week

BOOTCAMPS BY PATHWAY

70 days of bootcamps

30 Cloud Architect bootcamps

1 Cloud Developer bootcamp

1 Cloud Data Engineer bootcamp

WHAT LEARNERS ARE SAYING

4.66
Overall, this training was worth my time

4.66
I would recommend this training to others

4.46
I will be able to apply what I've learned to my work

4.73
The technical knowledge of the trainer was satisfactory

Our Snapshot

130,000
Trained on cloud technologies and best practices

130+
Certifications held across major public cloud providers

100+
Cloud instructors across NORTHAM, EMEA, and APAC

50+
Google Cloud Authorized trainers (largest team globally)

Author of AWS, Google Cloud, and Azure courseware

In-person and virtual delivery options

3X Google Cloud Training Partner of the Year (2017, 2018, 2019)

Global delivery capabilities

ROI Contacts

- **Steve Lockwood**
  steve@lochnetsystems.com
  https://www.linkedin.com/in/steve-lockwood/

- **Mike Prensky**
  Director, Business Development
  mike.prensky@roitraining.com
  781-254-7844