SSID: Guest

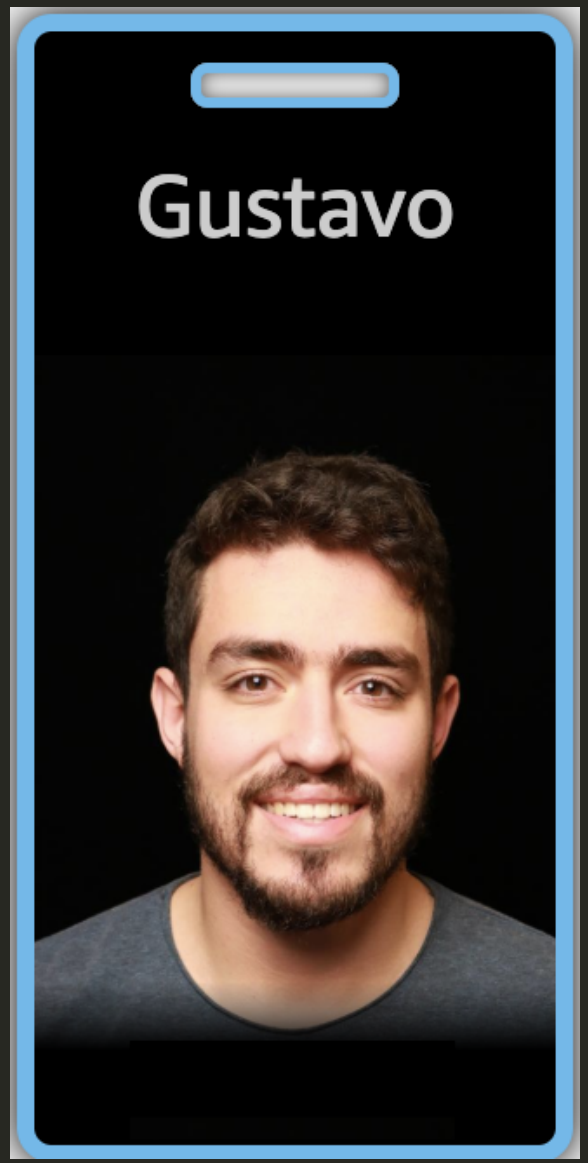
Password: BrokenWires@@2019

Getting Started with Kubernetes on AWS

Brought to you by the AWS Cloud Support Team

Welcome and thank you!

Instructors



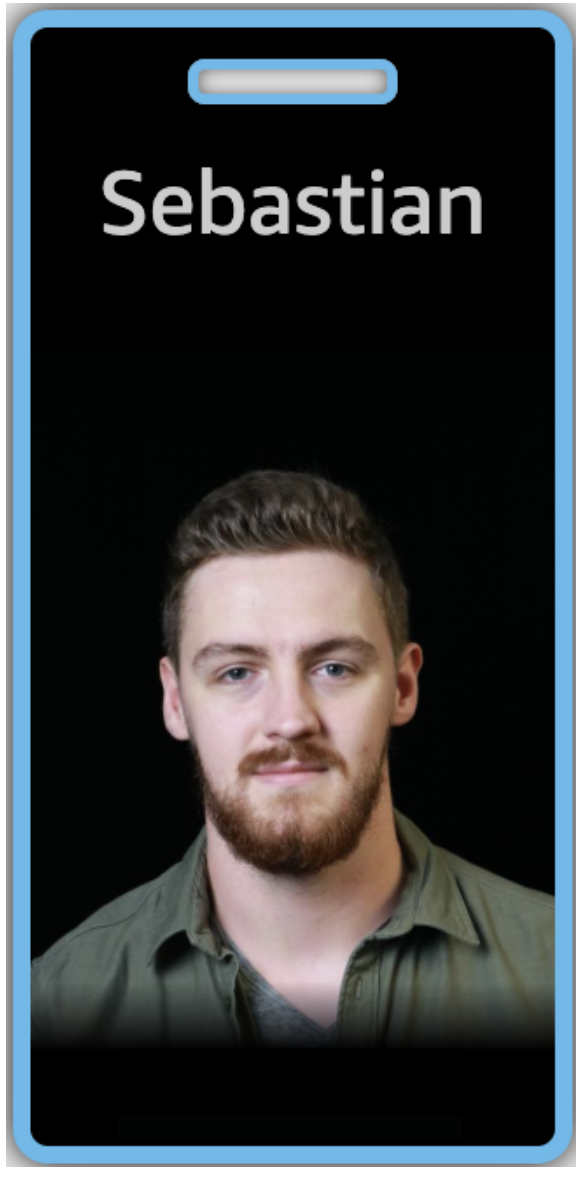
About us

- Brazilian
- Football addicted and Corinthians supporter
- Love Italian Food :D

About us

- Brazilian from Brasília, the capital and one of the must beautiful cities in the world.
- A sports fan, former handball player, and Corinthians supporter.
- A language enthusiast. I speak English, Portuguese, Spanish and a bit of French. It would be great if you guys could teach me some Italian during the intervals.

Helpers around the room



What is Cloud Support Engineering?

- Cloud Support Engineers are the front line of AWS' technical support
- Multiple teams, each with specific focus areas.
- What do we do? A surprising amount:
  - Customer interactions
  - Training
  - Learning is part of the job
  - Hiring

What do we do as Cloud Support Engineers - Deployment

- We assist customers to develop services and technologies built on top of AWS Cloud Platform.
    - ECS and EKS (container orchestration)
    - CloudFormation, Elastic Beanstalk, OpsWorks (Automating operations with Chef)
    - CodeDeploy, CodePipeline, CodeCommit (Implementing CI/CD pipelines on AWS)
- Apart from working on a broad spectrum of technical issues, we also work actively:
    - Coaching and mentoring new hires.
    - Developing and delivering trainings
    - Recruiting, interviewing and participating on the hiring process.

The Team

Day 1

Agenda Day1

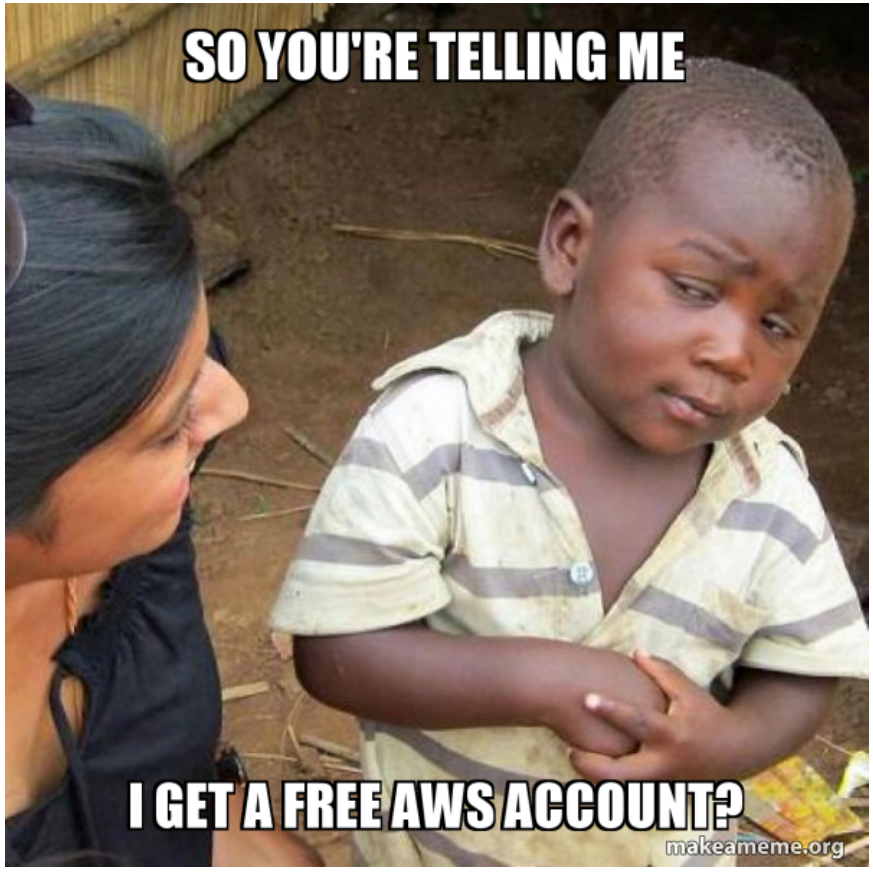
- Introduction to Containers
- Docker Overview
- Kubernetes Explained
- EKS - Exploring and Deploying
- Environment Setup
    - Break (Networking and Food)
- Exploring the cluster with kubectl
- Deploying an application with Kubernetes
- Kubernetes Services

Agenda Day2

- What have we learnt?
- Review questions.
- Challenge

Before we get started

Setting up AWS Account and Environment



Login into your AWS Account

1. Navigate to: https://dashboard.eventengine.run/login
2. Enter the team hash provided to you on arrival.
3. Click the AWS Console button.
4. Click the Open AWS Console button to log in to the account.
5. Optionally, credentials are provided for CLI access.

Note

The account will be deactivated at the end of the Event.

Setting up the environment

Launching your Lab Environment

1. Cloud9 - this will be where you'll be performing the labs throughout the sessions.

Follow the steps provided in the README.md to setup your environment

https://github.com/aws-els-lin/eks

Containers and Docker

Container Overview

- What is a container?

Container Overview

- What is a container?
- What is Docker? what benefits does it provide?

Container Overview

- What is a container?
- What is Docker? what benefits does it provide?
    - Consistent environment

Container Overview

- What is a container?
- What is Docker? what benefits does it provide?
  - Consistent environment
  - Portability of code, runtime, system tools, system libraries, etc.

Container Overview

- What is a container?
- What is Docker? what benefits does it provide?
  - Consistent environment
  - Portability of code, runtime, system tools, system libraries, etc.
  - Version control

Container Overview

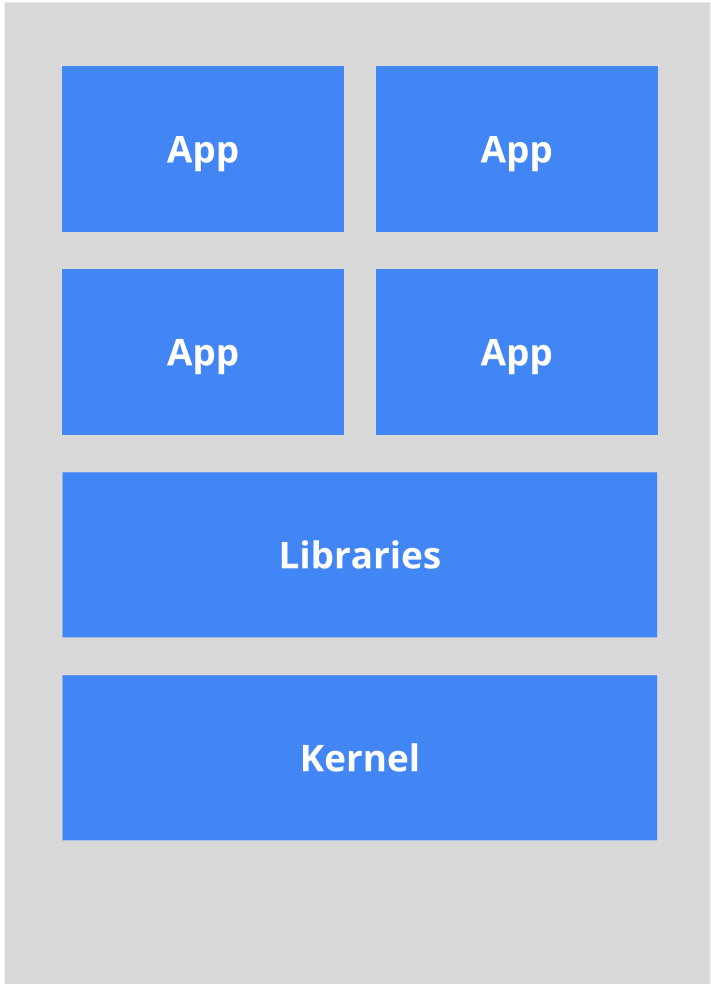
- What is a container?
- What is Docker? what benefits does it provide?
  - Consistent environment
  - Portability of code, runtime, system tools, system libraries, etc.
  - Version control
  - Lightweight

Container Overview

- What is a container?
- What is Docker? what benefits does it provide?
  - Consistent environment
  - Portability of code, runtime, system tools, system libraries, etc.
  - Version control
  - Lightweight
  - Microservices

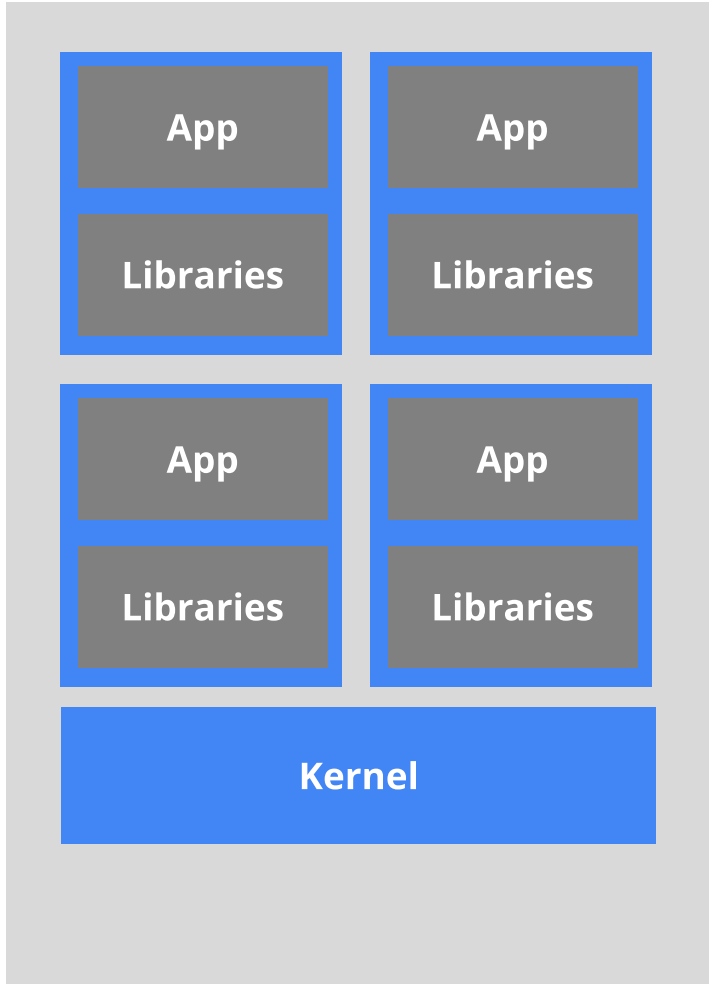
Why containers

The old way: Applications on host



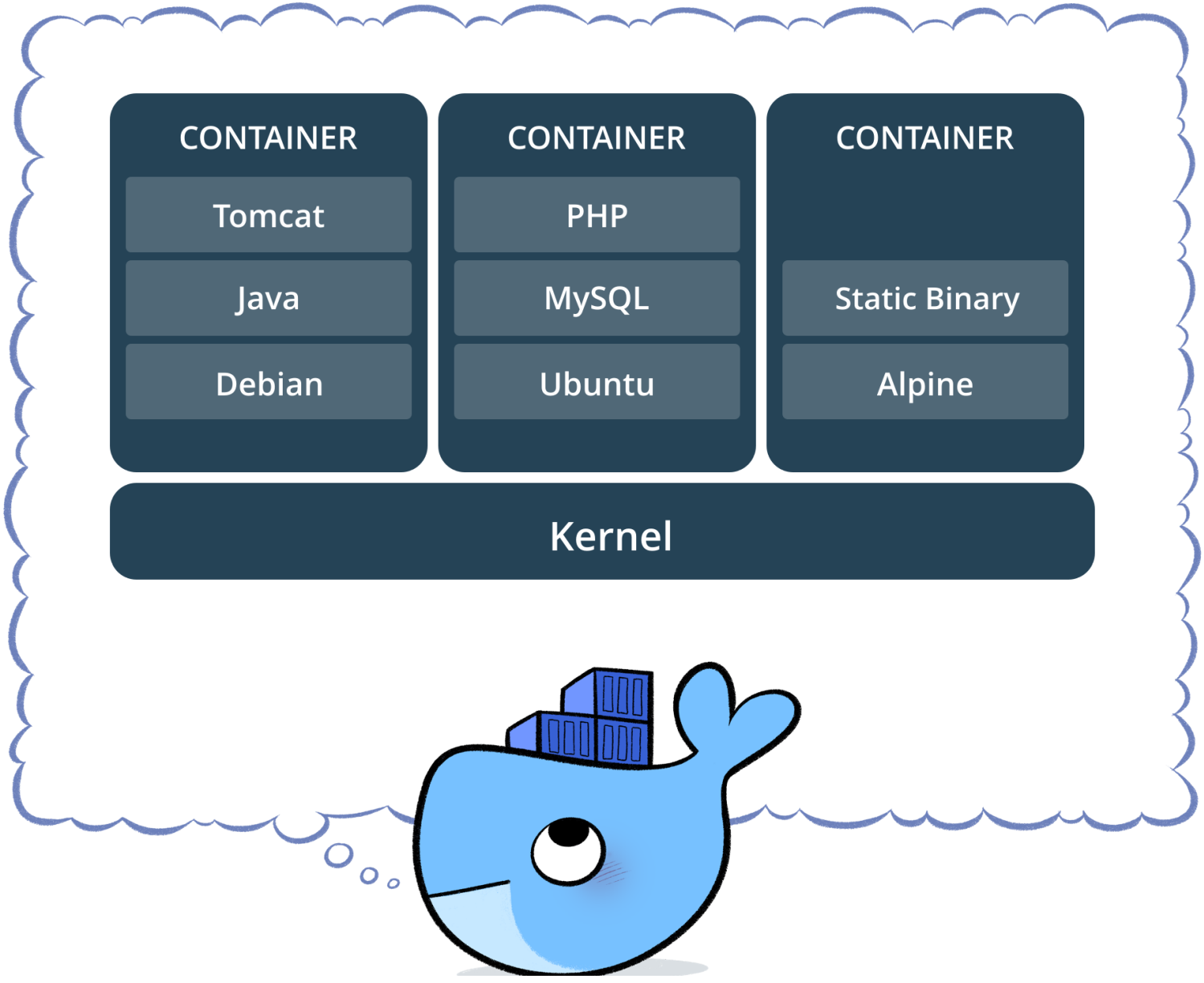
Heavyweight, non-portable
Relies on OS package manager

The new way: Deploy containers

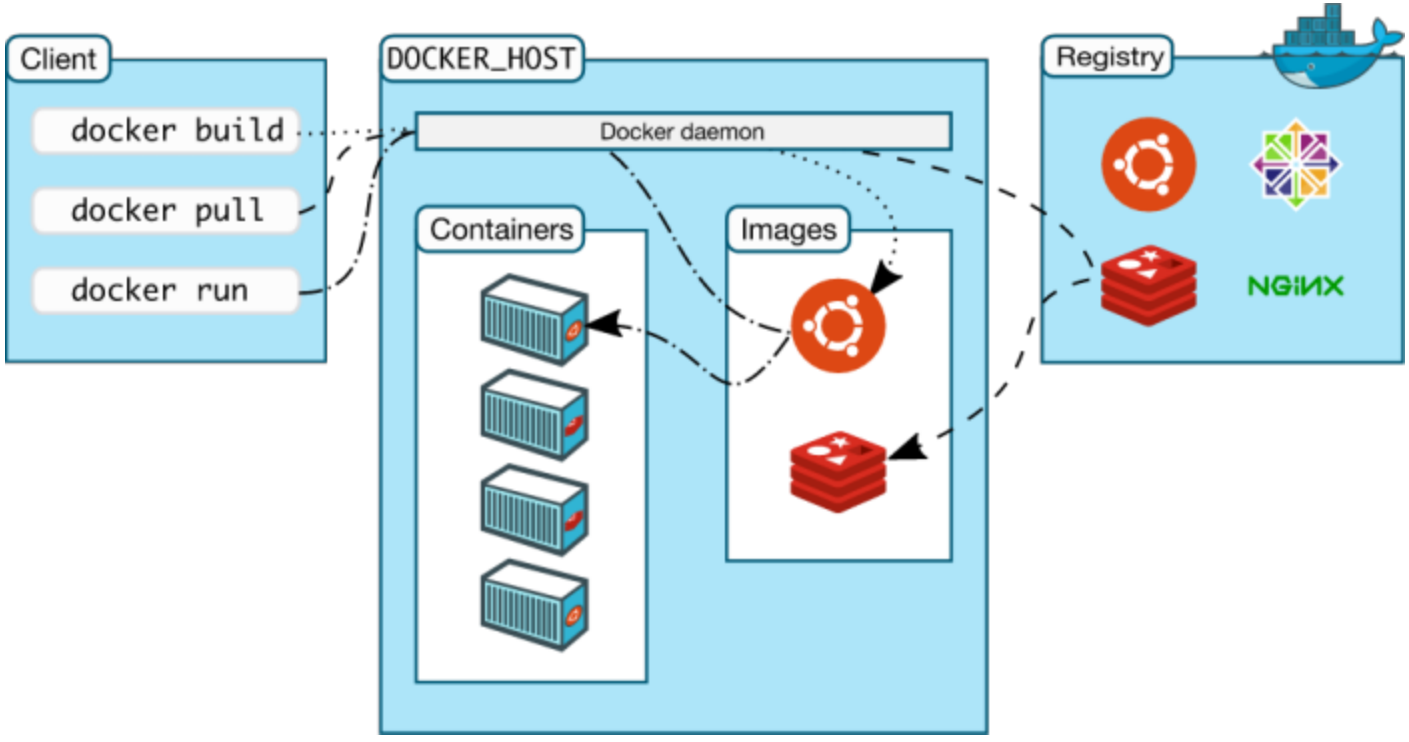


Small and fast, portable
Uses OS-level virtualization

Container Overview



Container Lifecycle

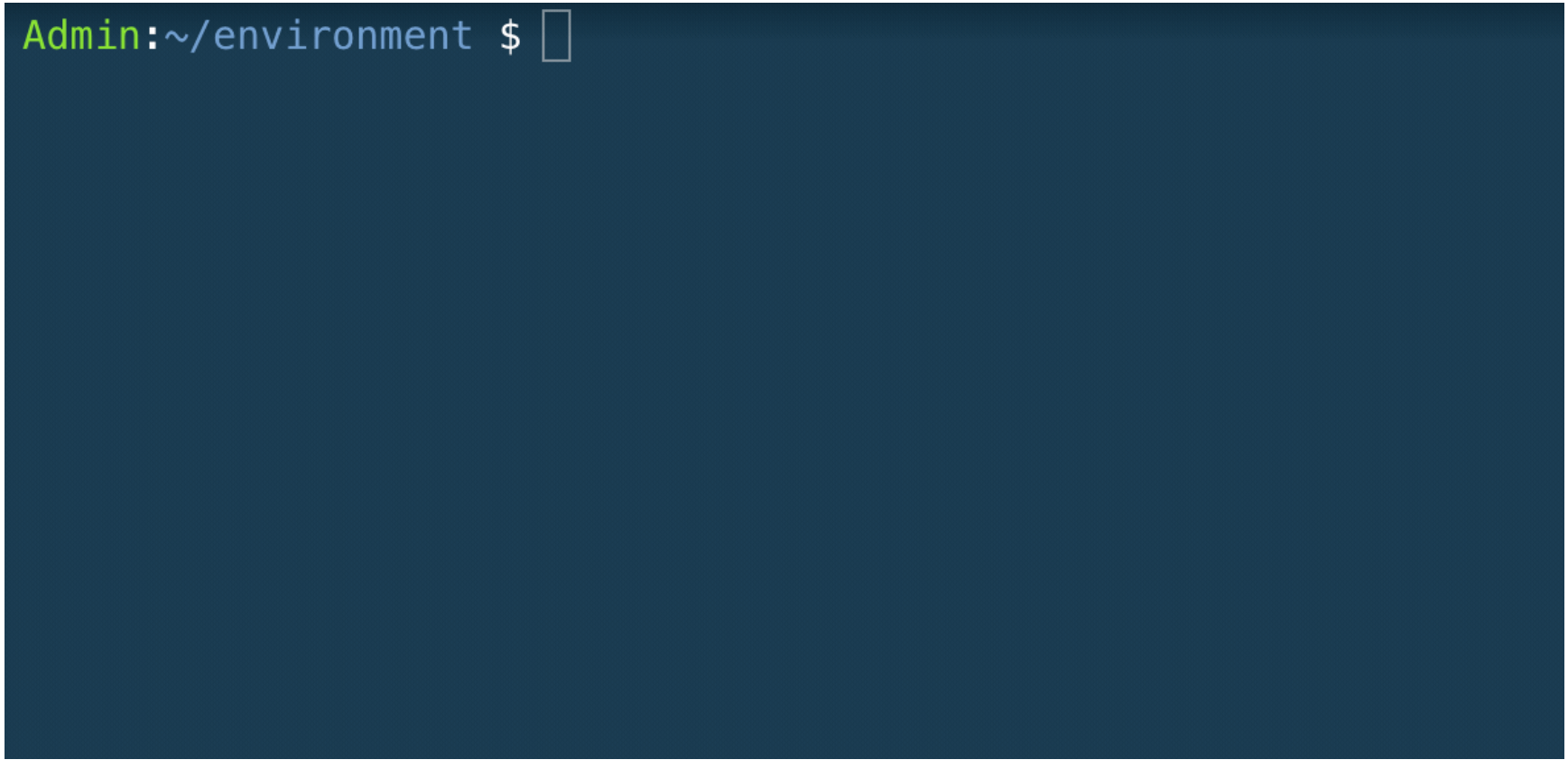


Docker commands - pull

Pull an image from an image registry

Docker commands - run

Running a container from an image



The `-p` flag maps a port from the host to the container

The `-d` flag runs the container in the background - detached

Let's docker

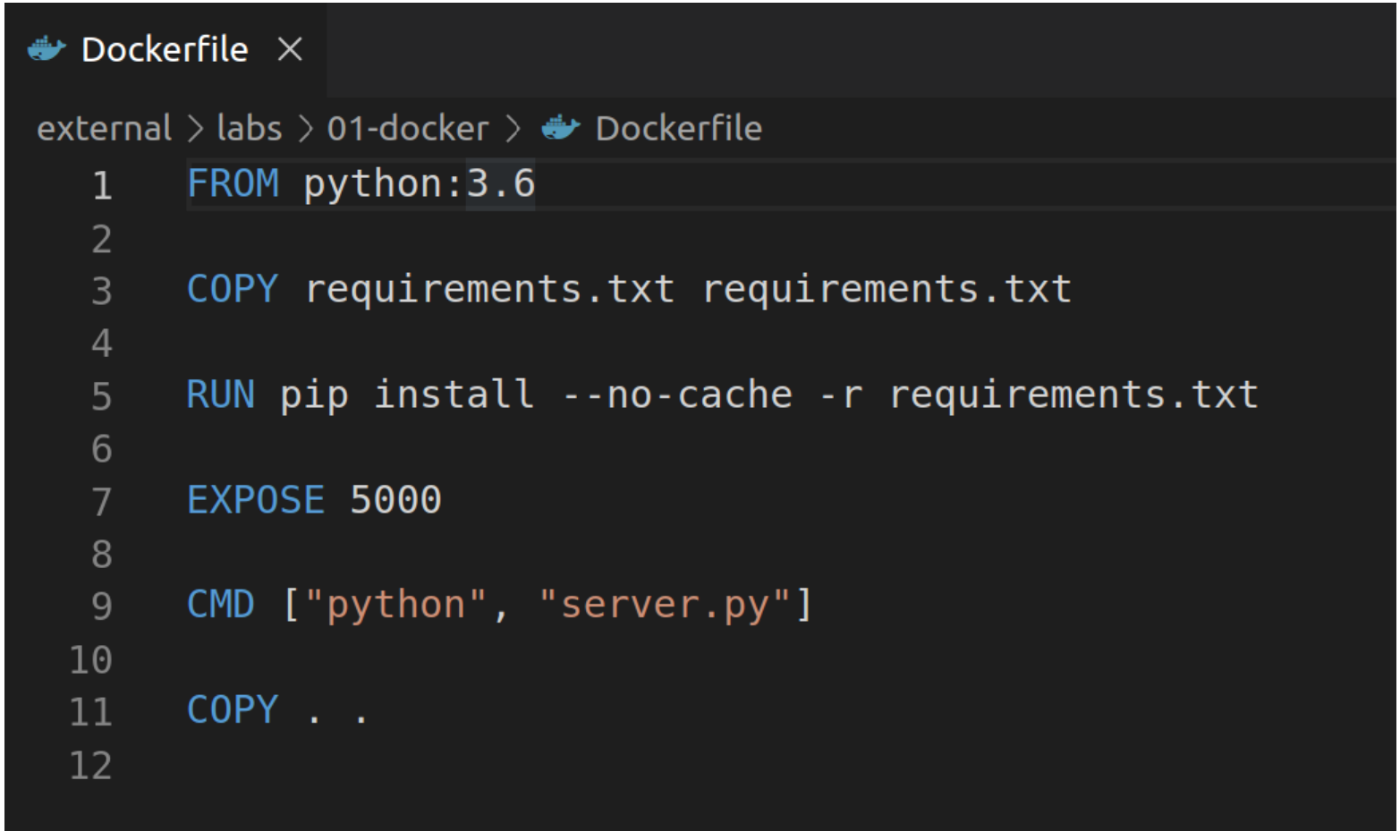
Lab 1: Building & Running a Docker Image

https://github.com/aws-els-lin/eks/tree/master/labs/01-docker

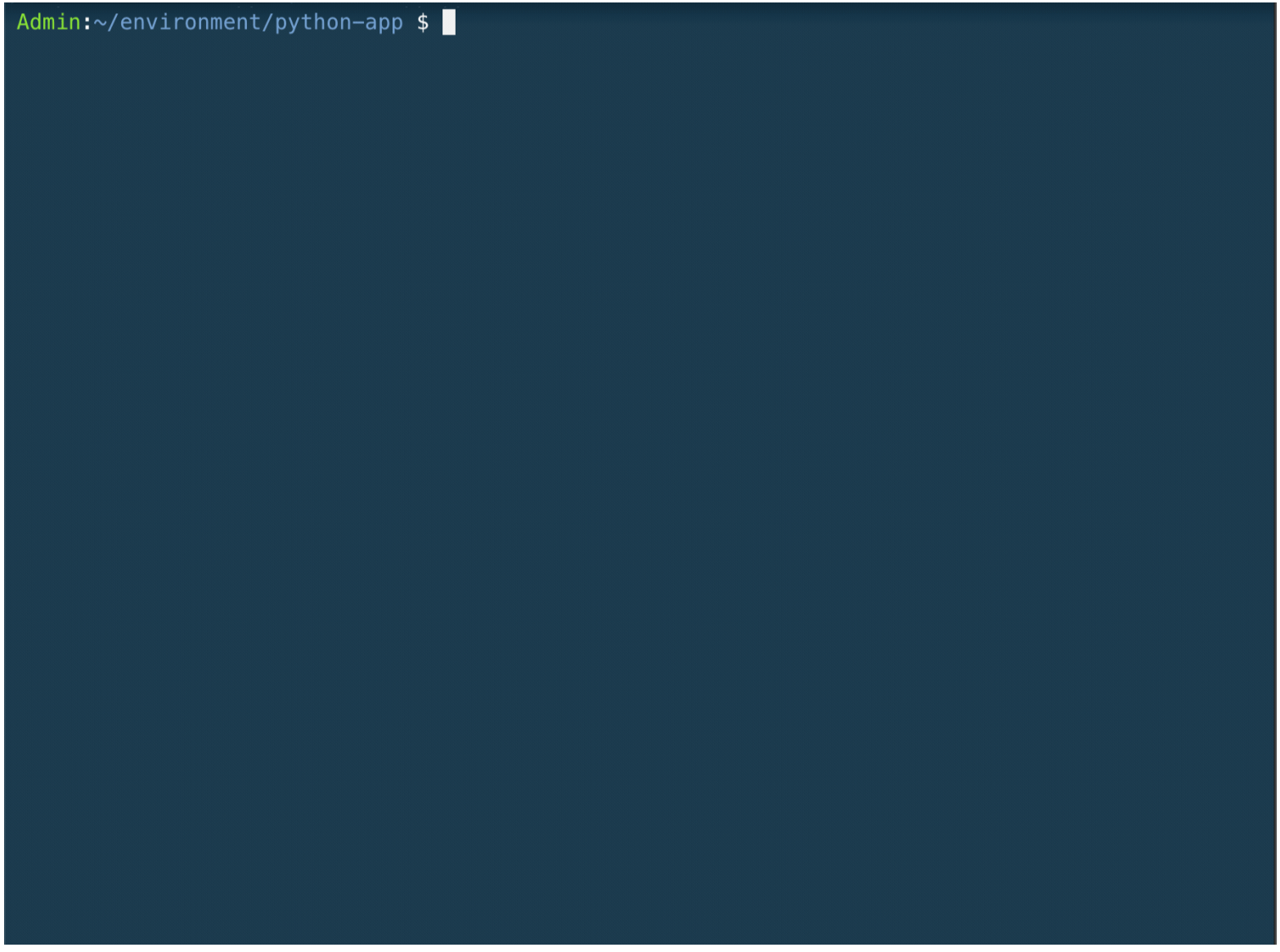
Dockerfile

Dockerfile contains a list of instructions to build a container image

`labs/01-docker/Dockerfile`

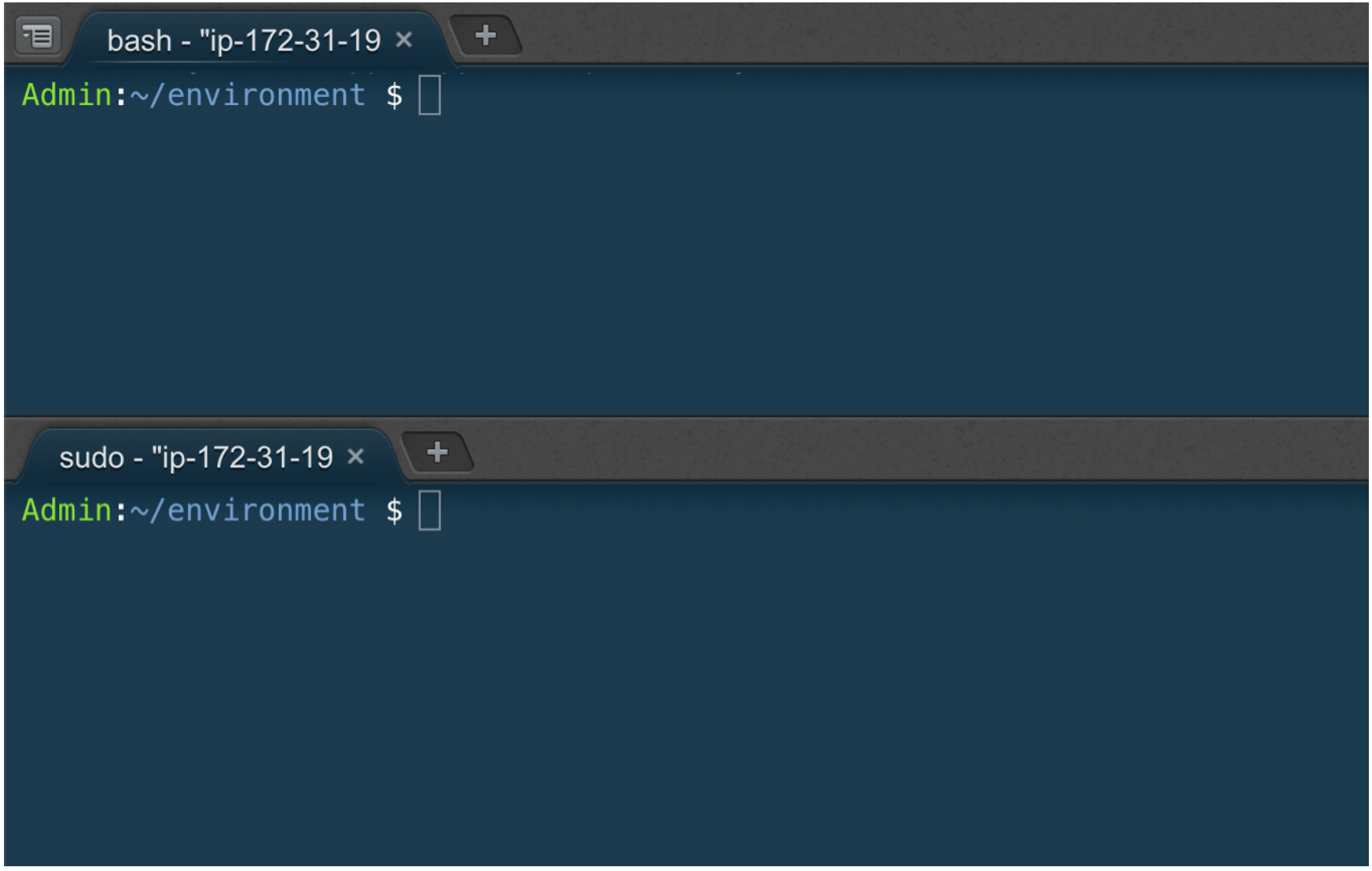


Docker build



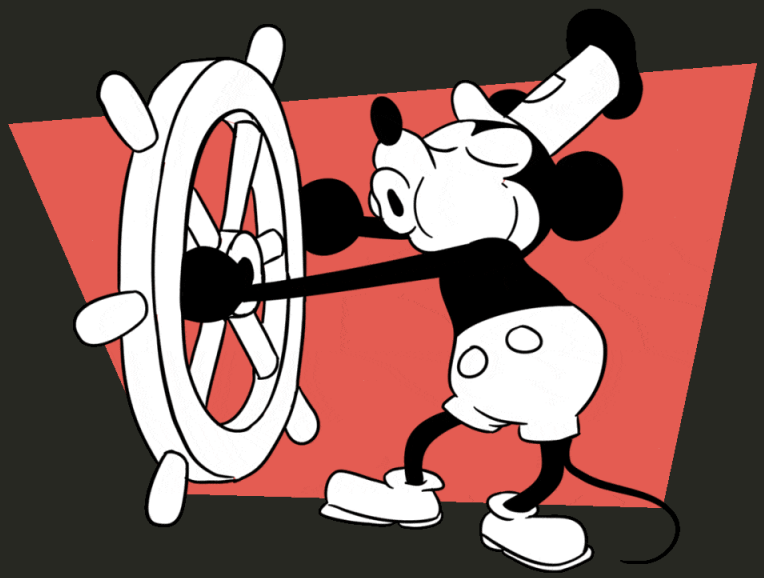
Docker run

Finally we can run a container from the image



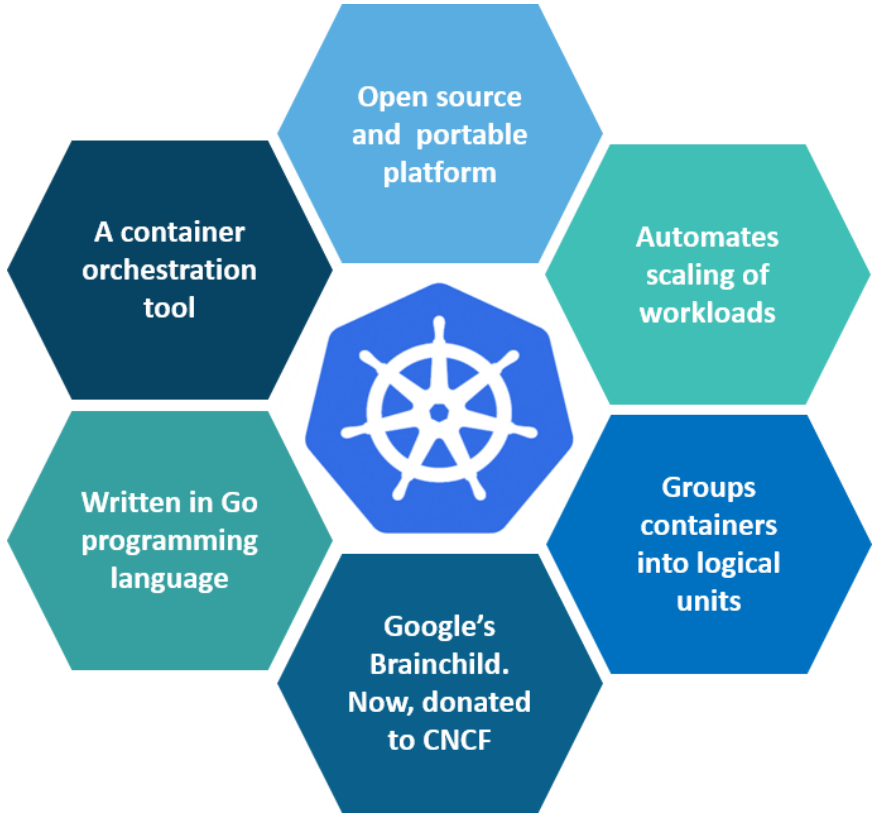
Before we move on, any questions?

What is



Kubernetes?

What is Kubernetes?



Why Kubernetes?

Why Kubernetes?

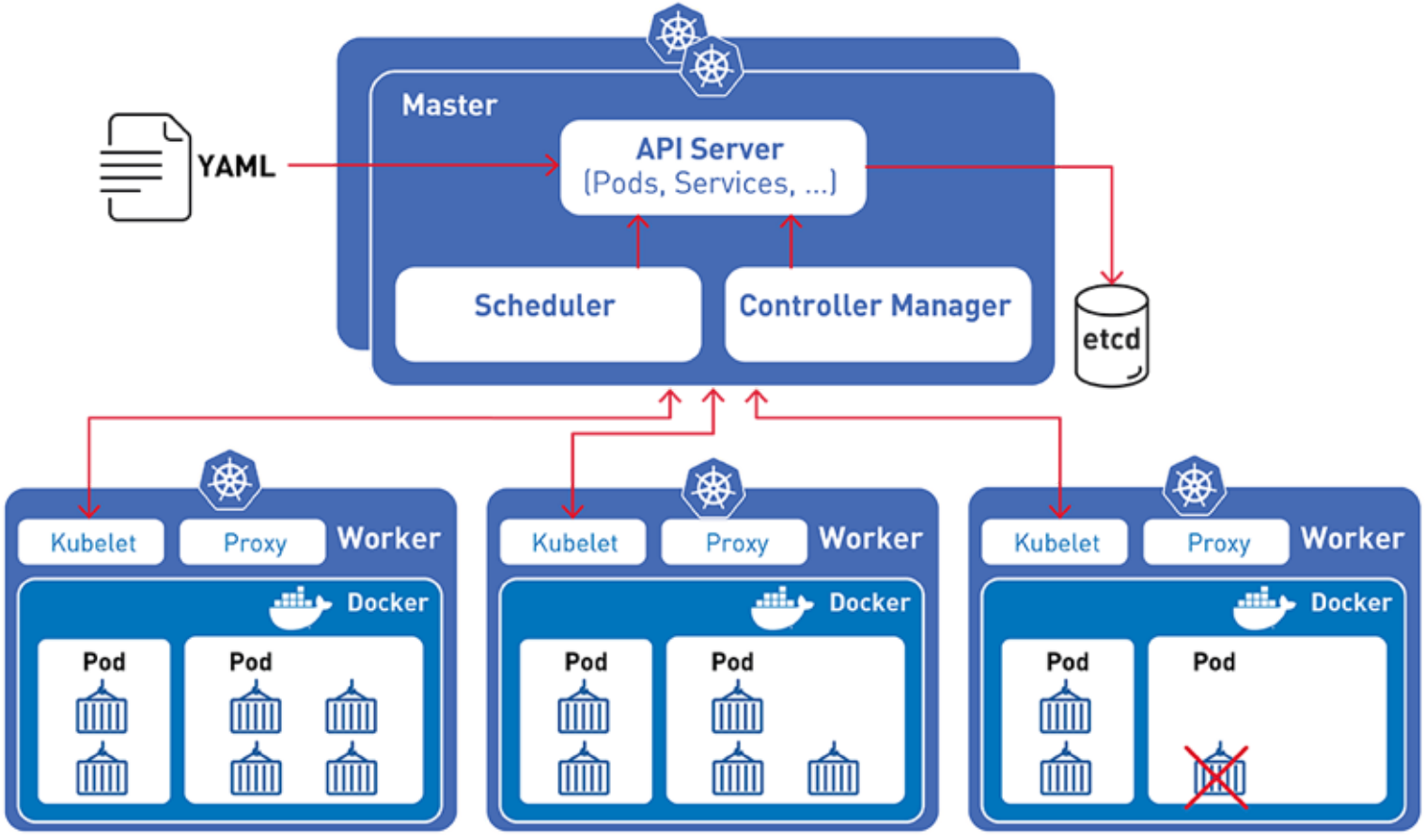
Why Kubernetes?

- Containers are a good way to bundle and run your applications.
- Containers paved the way to build cloud native systems, in which services are implemented using small clouds of containers.
- However, in a production environment, you need to manage the thousands of containers that run the applications and ensure that there is no downtime.

Key capabilities were missing !

- Resource Utilization
- Using multiple containers with shared resources
- Monitoring running containers
- Handling dead containers
- Autoscaling container instances to handle load
- Making the container services easily accessible Connecting containers to a variety of external data sources

Kubernetes Architecture



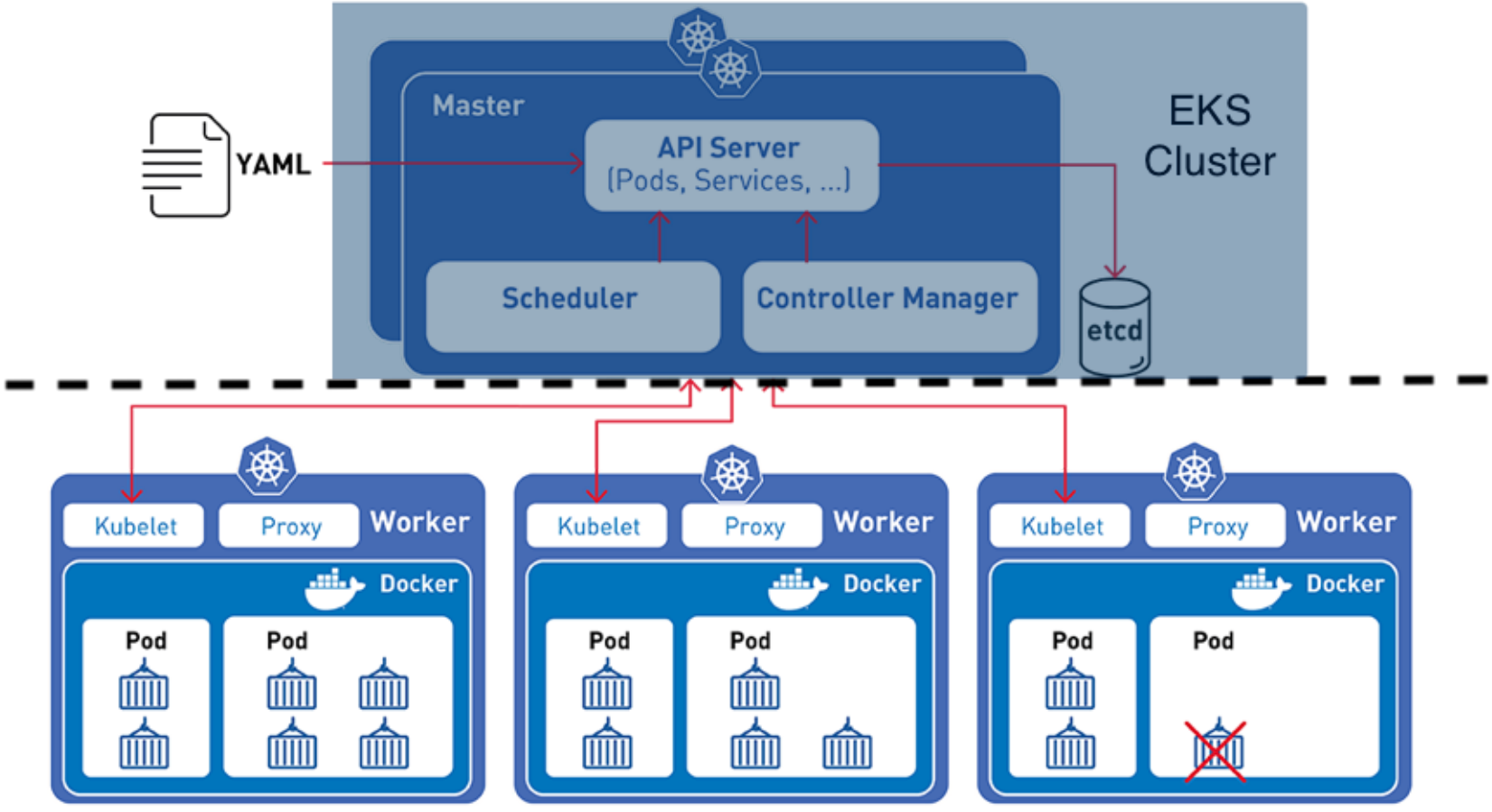
EKS

What is EKS?

What is EKS?

- EKS is the **managed** version of Kubernetes offered by AWS
- EKS launches and maintains the Control Plane for you with **high available** components
- EKS offers integration with other AWS services such as VPC and IAM. These integrations are **open source** projects built with the community
- EKS takes care of upgrades and patching
- EKS is based on vanilla Kubernetes

What is EKS?



Connecting to your K8s/EKS Cluster

What we'll need

- A client, which is a single binary called `kubectl`

What we'll need

- A client, which is a single binary called `kubectl`
- A configuration file for kubectl to be stored under: ~/.kube/config

What we'll need

- A client, which is a single binary called `kubectl`
- A configuration file for kubectl to be stored under: ~/.kube/config

The kubectl configuration can be generated automatically.

Launching a Kubernetes Cluster / EKS

Creating a Cluster

For this we'll use `eksctl` to create an EKS cluster and Worker Nodes:

```
eksctl create cluster --version 1.14 --node-type t3.medium --name eks
```

This command is also available on the GitHub page

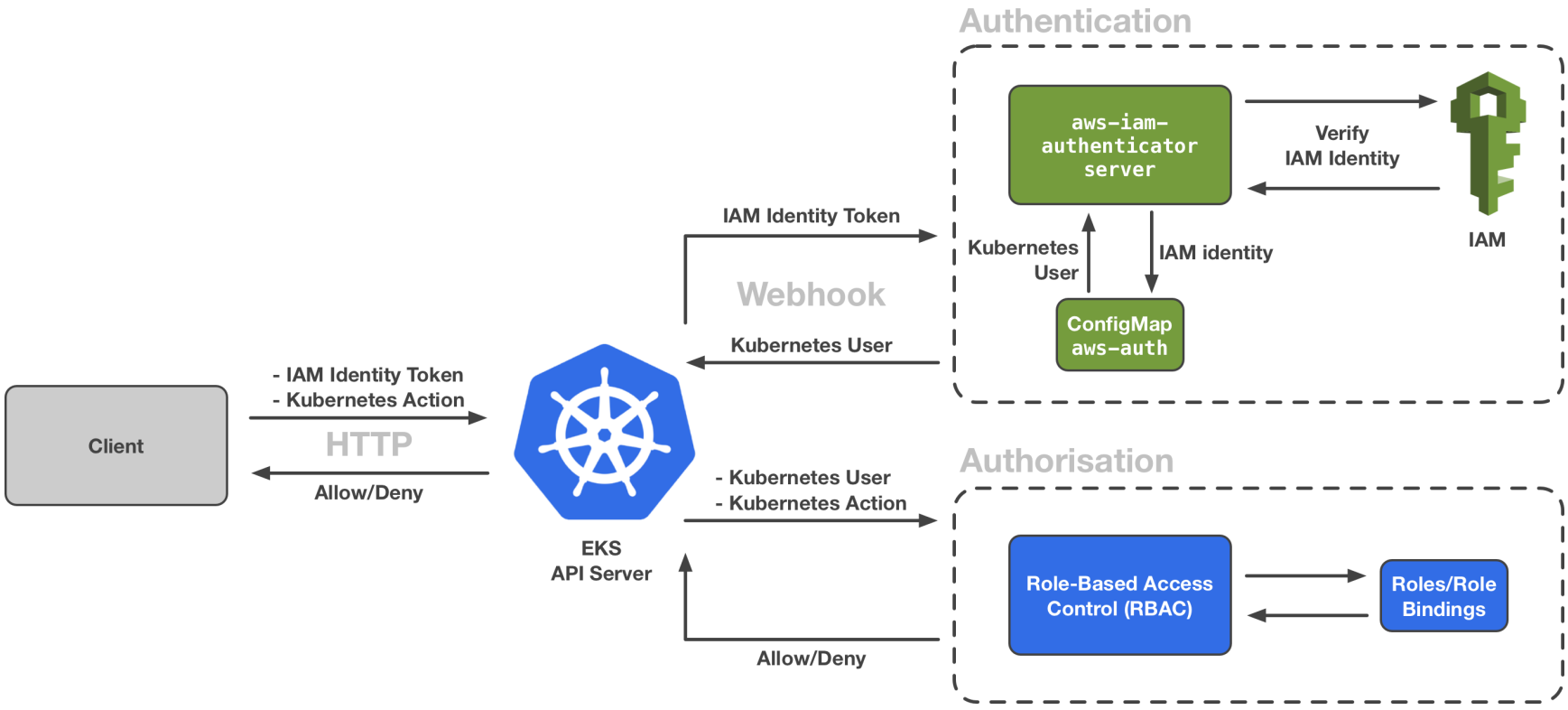
The EKS Cluster will take a approx 15 minutes to create

While that runs...

kubectl configuration - eksctl

```
Admin:~/environment $ kubectl config view
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://07413FD8A52FEC4D8C73360C03988541.yl4.ap-southeast-1.eks.amazonaws.com
  name: eks.ap-southeast-1.eksctl.io
contexts:
- context:
    cluster: eks.ap-southeast-1.eksctl.io
    user: i-0d8228356c7f829a6@eks.ap-southeast-1.eksctl.io
  name: i-0d8228356c7f829a6@eks.ap-southeast-1.eksctl.io
current-context: i-0d8228356c7f829a6@eks.ap-southeast-1.eksctl.io
kind: Config
preferences: {}
users:
- name: i-0d8228356c7f829a6@eks.ap-southeast-1.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      args:
      - token
      - -i
      - eks
      command: aws-iam-authenticator
      env: null
```

kubectl - Authentication



configmap - Authentication

Check to see if you have already applied the `aws-auth` ConfigMap.

```
kubectl describe configmap -n kube-system aws-auth
```

configmap - Authentication

Example ConfigMap:

```
# Please edit the object below. Lines beginning with a '#' will be ignored,
# and an empty file will abort the edit. If an error occurs while saving this file will be
# reopened with the relevant failures.
#
apiVersion: v1
data:
  mapRoles: |
    - rolearn: arn:aws:iam::555555555555:role/devel-worker-nodes-NodeInstanceRole-74RF4UBDUKL6
      username: system:node:{{EC2PrivateDNSName}}
      groups:
        - system:bootstrappers
        - system:nodes
  mapUsers: |
    - userarn: arn:aws:iam::555555555555:user/admin
      username: admin
      groups:
        - system:masters
    - userarn: arn:aws:iam::111122223333:user/ops-user
      username: ops-user
      groups:
        - system:masters
```

Let's check our new Cluster

Let's check our new Cluster

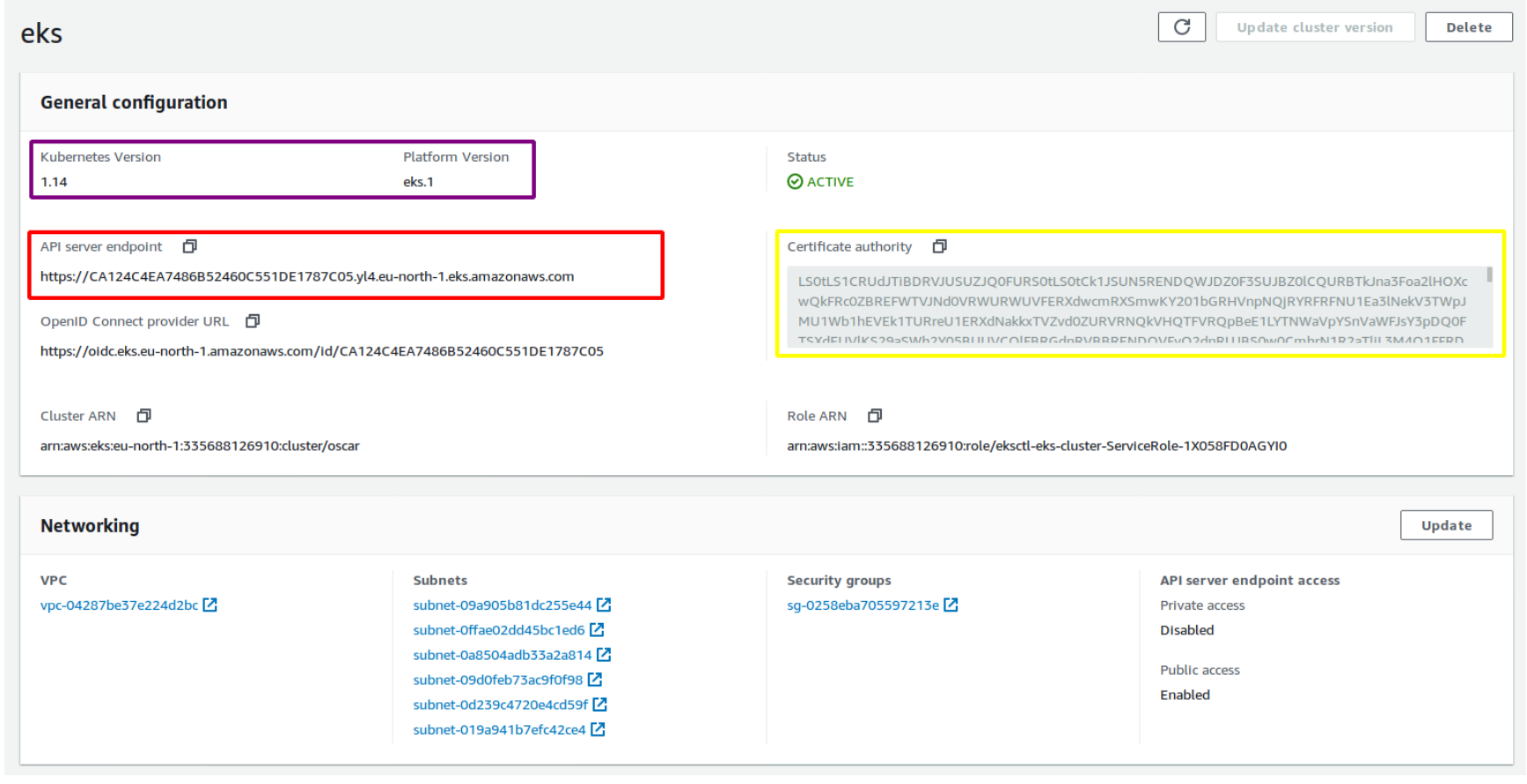
1 - Head to the EKS service in the AWS Web Console

Let's check our new Cluster

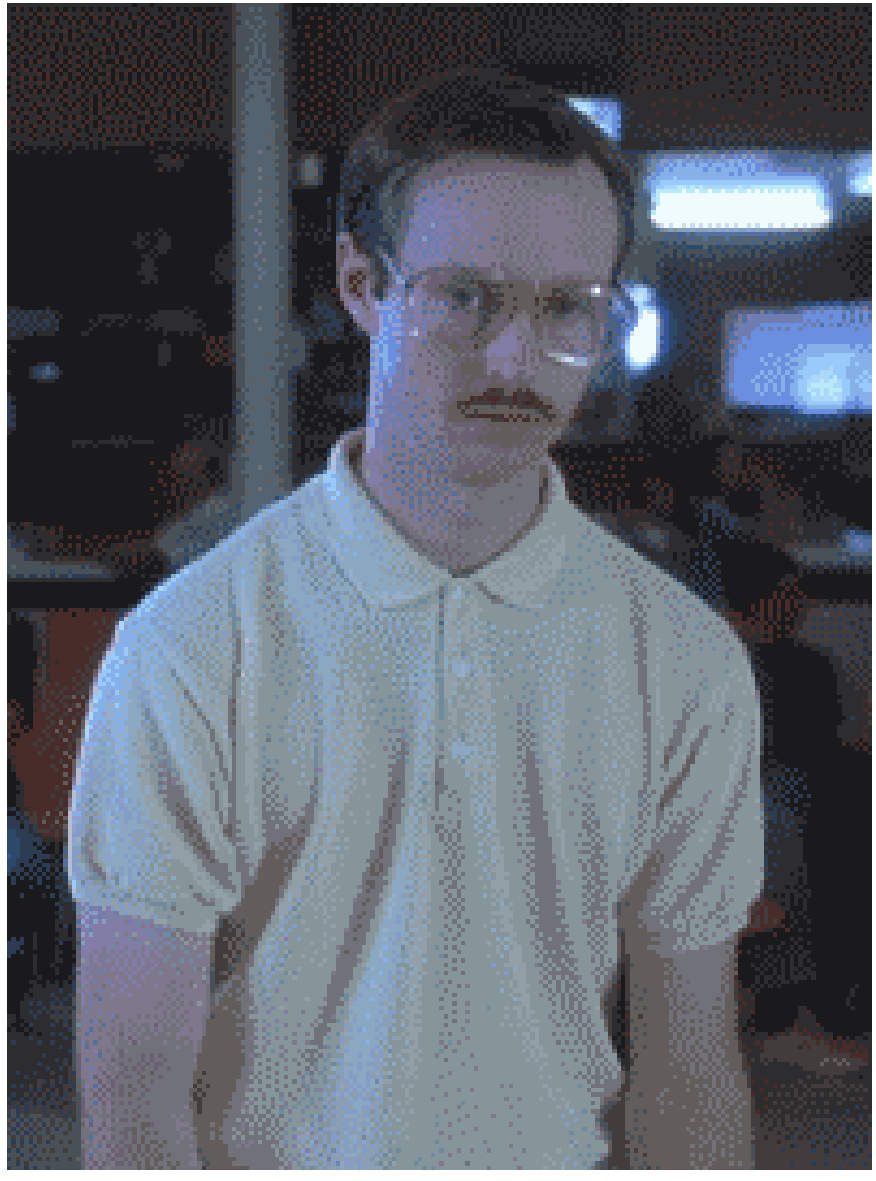
1 - Head to the EKS service in the AWS Web Console

2 - Select the 'eks' cluster and check what options are available

Checking your EKS cluster



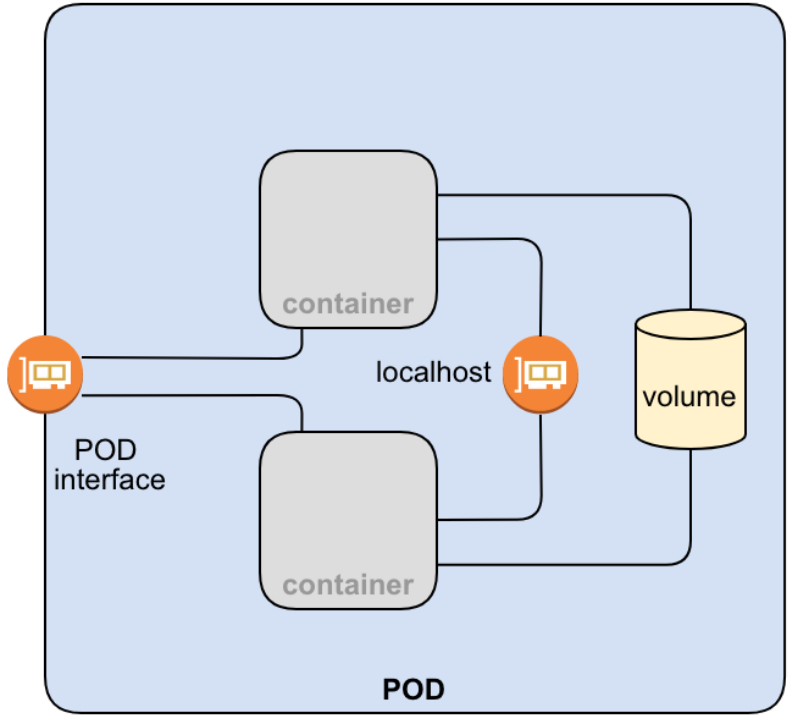
Successfully Create and Connect to EKS Cluster



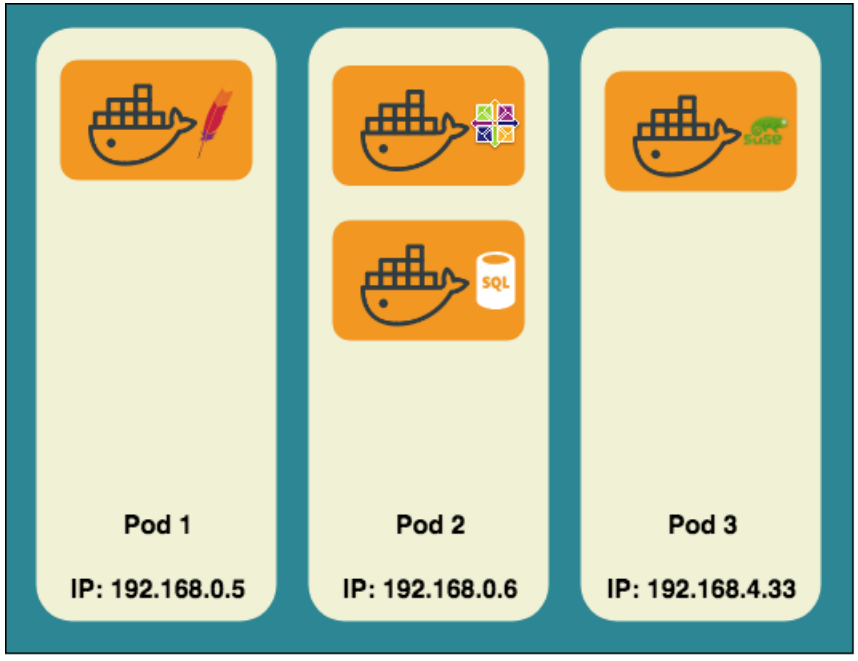
Pod

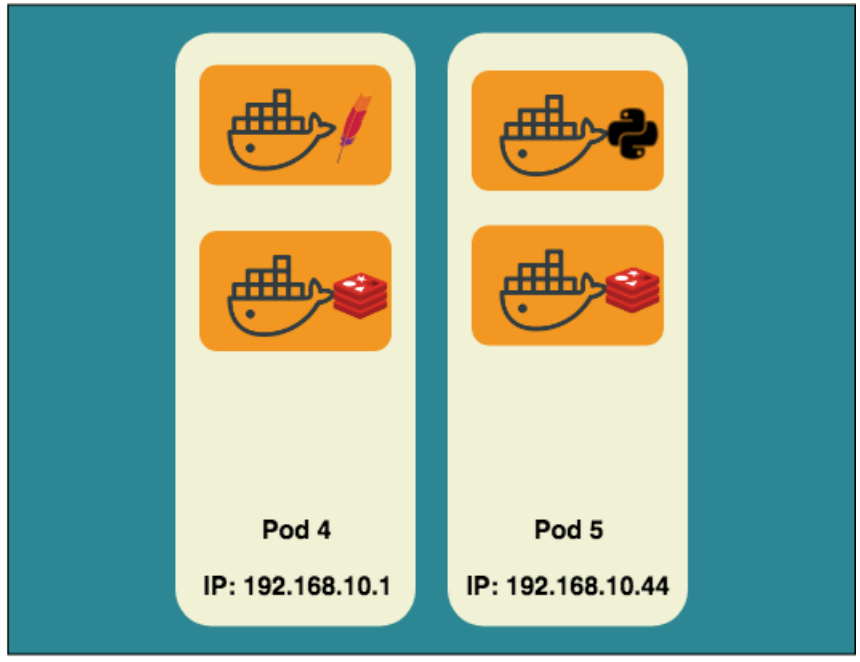
What is a Pod?

- The smallest building block of Kubernetes
- A Pod encapsulates the container(s) and resources needed to run the application
- A unit of deployment



What is a Pod?





Creating Pods in Kubernetes

Let's kubectl

Lab 2: Introduction to Pods

https://github.com/aws-els-lin/eks/tree/master/labs/02-pods

Lab 2: Define a Pod

Pod definition

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - name: container1
    image: nginx
```

Lab 2: Creating a Pod

Send the definition to the cluster:

```
$ kubectl apply -f pod.yaml
pod/web-server created
```

Lab 2: Check the Pod

List and describe the Pod

```
# View the deployed pod
$ kubectl get pods
NAME                              READY   STATUS    RESTARTS   AGE
web-server                        1/1     Running   0          10s

# See details of the pod
$ kubectl describe pod web-server
Name:               web-server
Namespace:          default
Priority:           0
PriorityClassName:  <none>
Node:               ip-10-0-100-30.eu-north-1.compute.internal/10.0.100.30
Start Time:         Fri, 27 Sep 2019 15:55:35 +0200
Labels:             <none>
[...]
```

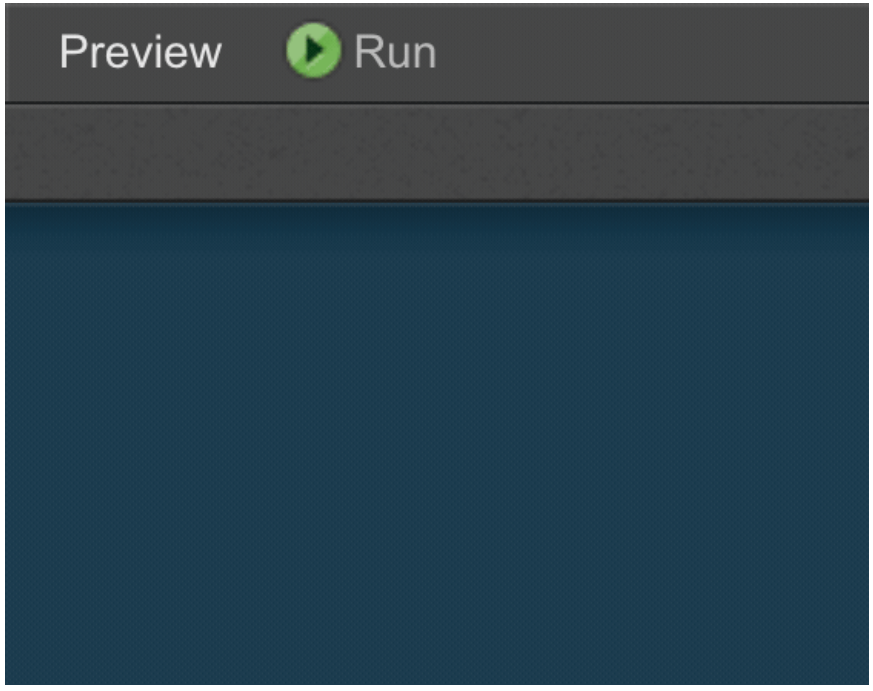
Lab 2: Check the Pod

Create a tunnel and connect to your Pod

```
kubectl port-forward pod/web-server 8080:80 &

curl localhost:8080
```

Or, connect using a browser and Cloud9



Lab 2: Clean up

We ran the port-forward in the background, lets clean it up before we move on

In the Cloud9 terminal to bring it back to the foreground:

```
fg

# Hit Ctrl + C to kill the port-forward
```

Working with Pods

Lab 3: Working with Pods

Check the logs

```
$ kubectl logs web-server
```

Lab 3: Working with Pods

Check the logs

```
$ kubectl logs web-server
```

Connect

```
# run one command
kubectl exec web-server cat /etc/hostname

# run with console connected to pod
kubectl exec -it web-server -- bash
```

Lab 3: Working with Pods

Check the logs

```
$ kubectl logs web-server
```

Connect

```
# run one command
kubectl exec web-server cat /etc/hostname

# run with console connected to pod
kubectl exec -it web-server -- bash
```

You can delete it: *But keep it for now!*

```
kubectl delete pod web-server

# OR

kubectl delete -f labs/01_pod.yaml
```

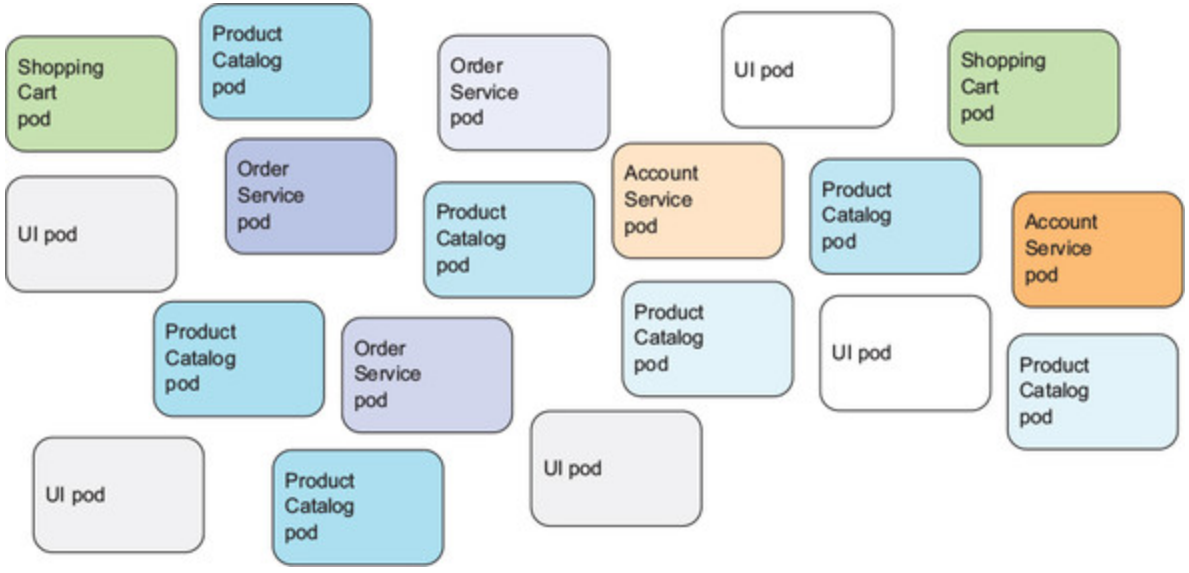
Let's kubectl

Lab 3: Playing around with our Pod

https://github.com/aws-els-lin/eks/tree/master/labs/03-more-pods

Labels

What if we are running a lot of pods?

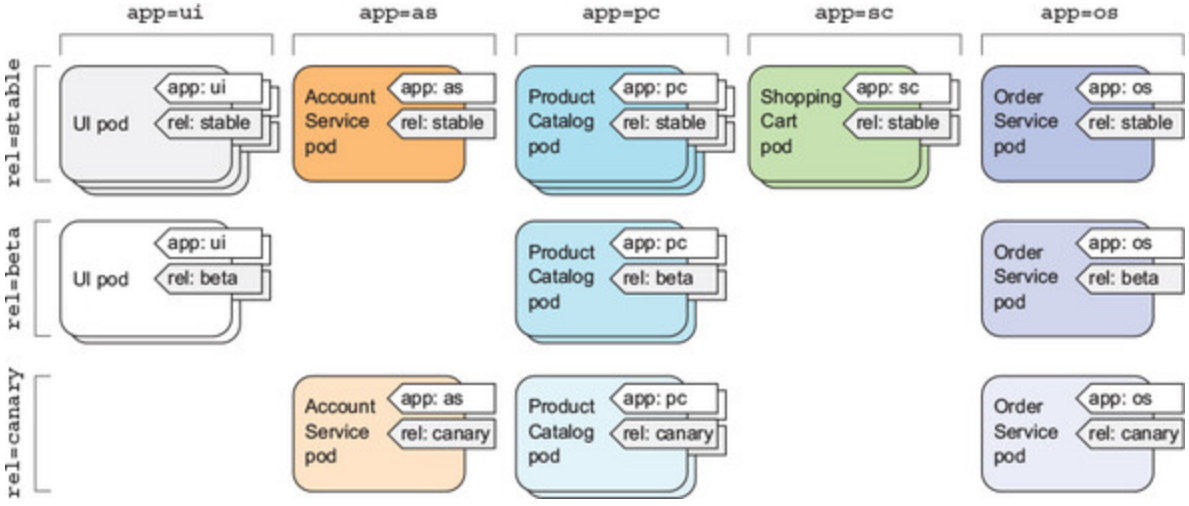


Picture from Kubernetes in action

Labels

Labels are key/value pair tags (like tags in AWS)

Can be used to query and organize resources



Picture from Kubernetes in action

Lab 4: Applying labels

Labels in the object definition

```
apiVersion: v1
kind: Pod
metadata:
  name: echo-server
  labels:
   env: training
   type: single_pod
spec:
  containers:
  - name: echo
    image: k8s.gcr.io/echoserver:1.4
```

Checking the labels

```
kubectl apply -f labs/03_labels.yaml

kubectl get pods --show-labels
```

File: labs/03_labels.yaml

Let's kubectl

Lab 4: Labeling our Pods

https://github.com/aws-els-lin/eks/tree/master/labs/04-labels

Links

Pods

- https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/
- https://kubernetes.io/docs/concepts/workloads/pods/pod/

Kubernetes in Action

- https://www.safaribooksonline.com/library/view/kubernetes-in-action/9781617293726/

Demos

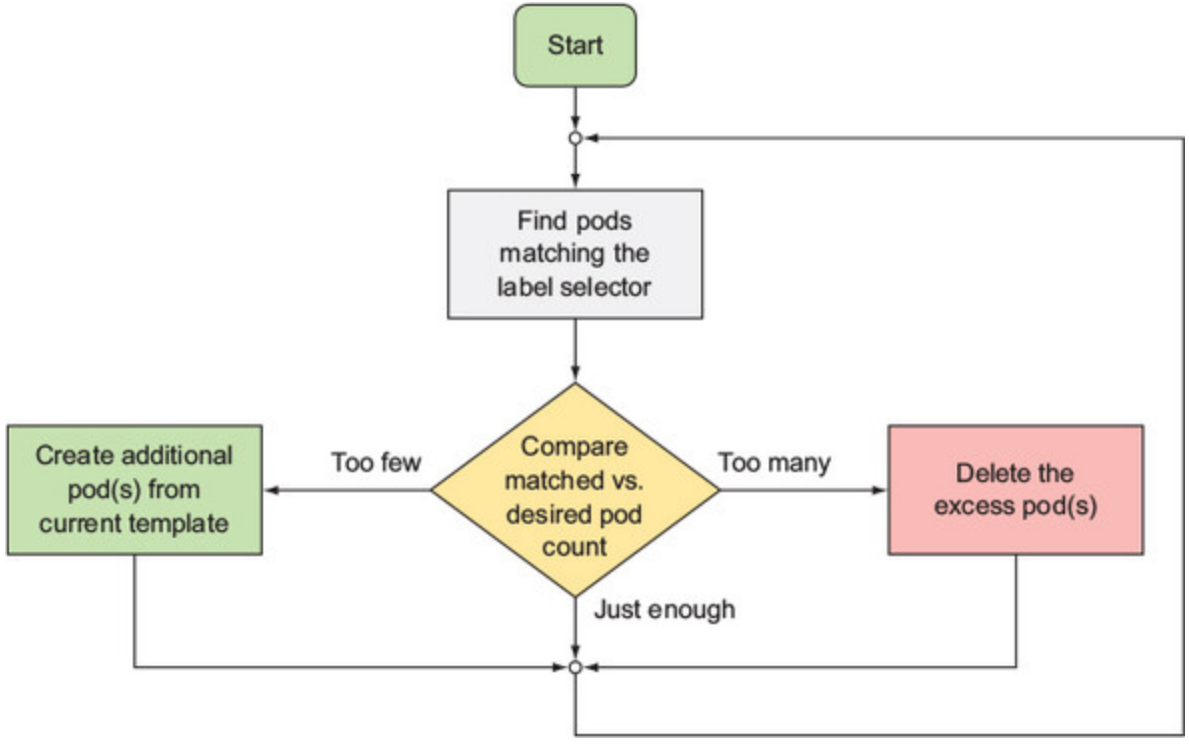
- https://eksworkshop.com/
- https://github.com/kubernetes/contrib/blob/master/micro-demos/
- https://www.katacoda.com/

Controllers

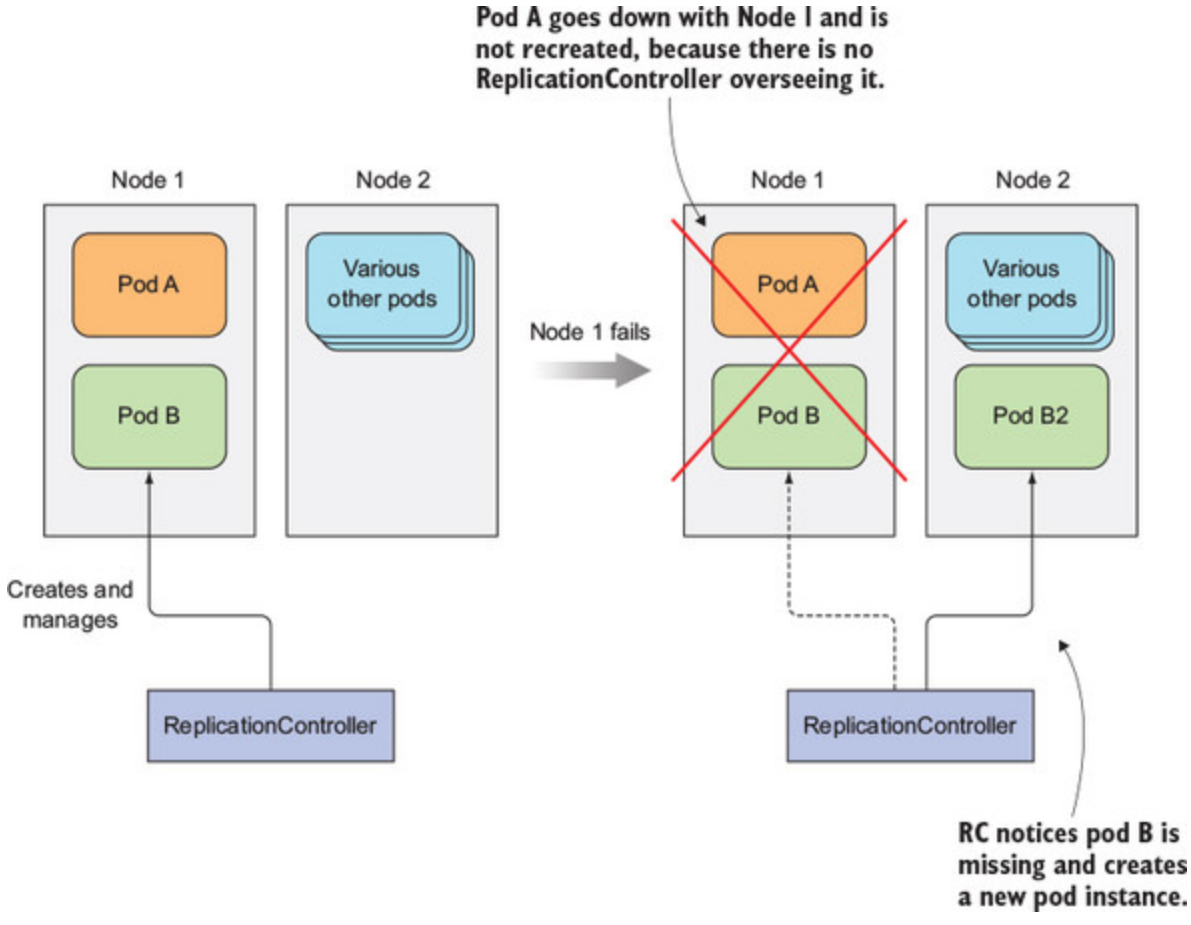
What controllers are and what they do?

- Resource responsible for managing pods
- Ensure pods are always running
- Replace missing and unhealthy pods
- Delete 'extra' pods
- Provide an easy way to scale the application
- Rely on Labels to account for the pods

What controllers are and what they do?



What controllers are and what they do?



Controllers

- Most commonly used controllers in Kubernetes:
  - ReplicationController
  - ReplicaSet the next generation of Replication Controllers
  - Deployments - preferred way to manage Replica Sets
  - DaemonSet
  - Jobs
  - CronJobs
  - StatefulSets

Deployments

Deployments

What is it?

- A Deployment controller provides declarative updates at controlled rate for Pods
- An easy way to deploy updates for existing applications
- Allows you to pause/resume deployments

Deployments

Comparing: Pods vs Deployment:

Single Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.7.9
```

Deployment:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web-server-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
     app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
```

File: labs/04_deployment.yaml

Deployments

Lab 5: Create a Deployment

- Creating a Deployment

```
kubectl apply -f labs/04_deployment.yaml
```

- Checking the results

```
kubectl get deployments
kubectl get pod
kubectl get pod -l app=nginx
```

- Scale out the Deployment

```
kubectl scale deployment web-server-deployment --replicas=5
```

- Check the nginx server version

```
kubectl port-forward web-server-deployment-XXXXXX-YYYYY 8080:80
```

Deployments

Lab 5: Scale the Deployment

- Scale in the deployment. Let's be frugal

```
kubectl edit deployment web-server-deployment

# set spec.replicas to 3
```

- Checking the results

```
kubectl get pod

# OR

kubectl get pod -l app=nginx -L app
```

Deployments

Lab 5: Update the Deployment

- Update the deployment

```
kubectl set image deployment web-server-deployment nginx=httpd

# OR

kubectl edit deployment web-server-deployment
# change spec.template.spec.containers.image to httpd
```

- Checking the results

```
kubectl rollout status deployment web-server-deployment
# OR
kubectl get pod
# OR
kubectl get pod -l app=nginx -L app
```

- Check the Nginx server version now

```
kubectl port-forward web-server-deployment-XXXXXX-YYYYY 8080:80
```

Deployments

Let's kubectl

Lab 5: Deployment

https://github.com/aws-els-lin/tree/master/labs/05-Deployments

Services

Services

What's a service?

- Service is another layer on top of the pods
- Instead of connecting to the pods directly we connect to the service instead
- Very similar to a load balancer

Services

What's a service?

- Service is another layer on top of the pods
- Instead of connecting to the pods directly we connect to the service instead
- Very similar to a load balancer

But why?

- Pods are ephemeral
- Pods' IPs are dynamic
- A single application might contain several Pods

Services

What's a service?

- Service is another layer on top of the pods
- Instead of connecting to the pods directly we connect to the service instead
- Very similar to a load balancer

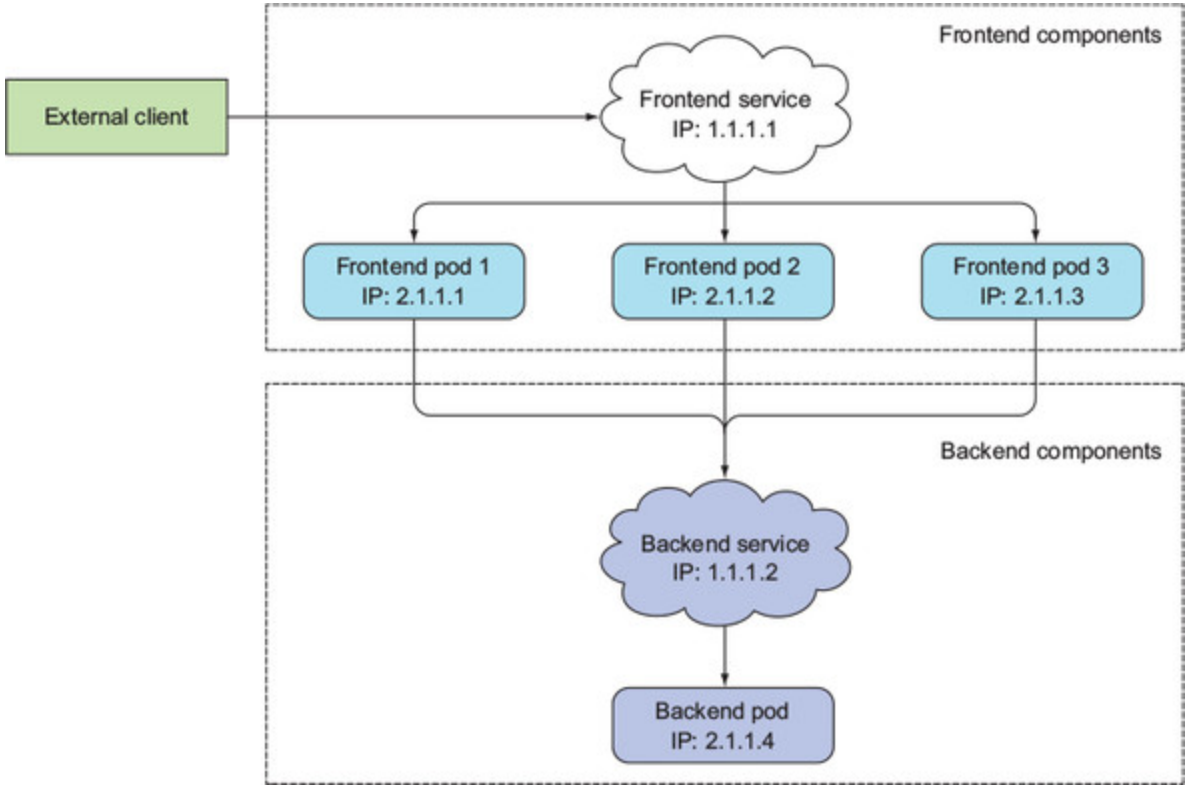
But why?

- Pods are ephemeral
- Pods' IPs are dynamic
- A single application might contain several Pods

So, how to reach an application?

Services

A Service is an abstraction layer which enables external traffic exposure, load balancing and service discovery



* from https://kubernetes.io/

Lab 6: Create a Service

```
apiVersion: v1
kind: Service
metadata:
  name: web-app
spec:
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: web-server
```

File: labs/06-services

Lab 6: Create a Service

```
apiVersion: v1
kind: Service
metadata:
  name: web-app
spec:
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: web-server
```

- Creating a service

```
kubectl apply -f service.yaml
```

File: labs/06-services

Lab 6: Create a Service

```
apiVersion: v1
kind: Service
metadata:
  name: web-app
spec:
  ports:
  - port: 80
    targetPort: 80
  selector:
    app: web-server
```

- Creating a service

```
kubectl apply -f service.yaml
```

- Checking the results

```
kubectl get services
kubectl describe svc web-app
```

File: labs/06-services

Services

- There are different types of services:
  - ClusterIP is the default. Used for intra-cluster communication
  - LoadBalancer provisions a Load Balancer for you.

Connecting to your service:

```
kubectl edit svc web-app

# change spec.type from 'ClusterIP' to 'LoadBalancer'
```

- Checking the results

```
kubectl get svc web-app
```

- Now you should get the LB URL from the EC2 console and open it in your browser.

Let's kubectl

Lab 6: Services

https://github.com/aws-els-lin/eks/tree/master/labs/06-services

Follow the steps in the `labs/06-services/README.md` file.

End of Day 1

Questions?

Thank you!