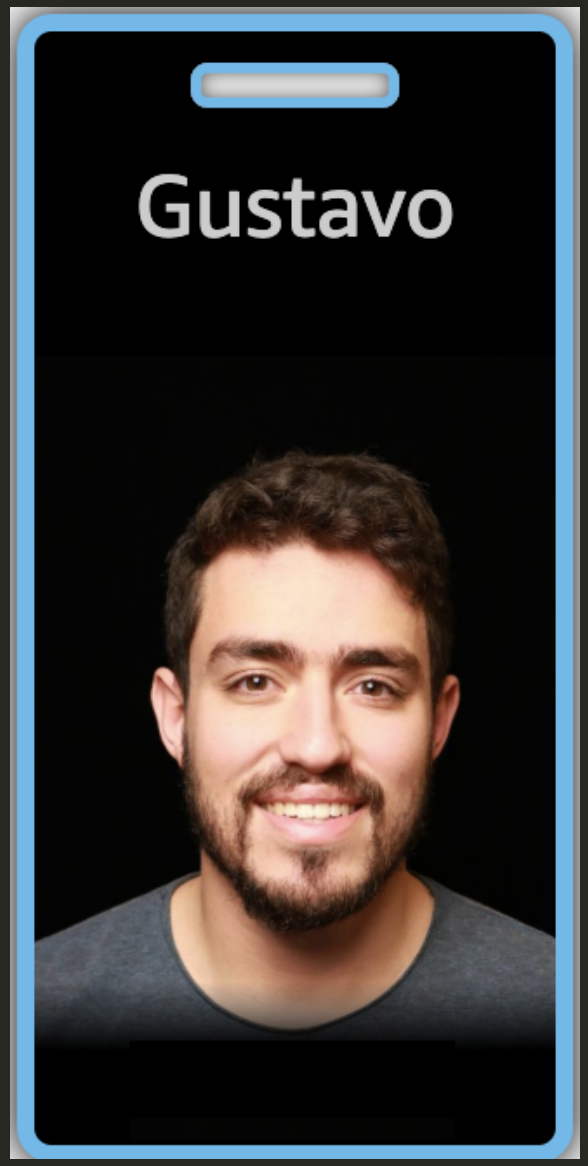
Getting Started with Kubernetes on AWS

Brought to you by the AWS Cloud Support Team

Welcome and thank you!

Instructors



About us

- Brazilian
- Football addicted and Corinthians supporter
- Quarentined for 70 days and counting :(

About us

- DevOps Cloud Support Engineer at AWS. 16+ years of work experience in IT.
- Brazilian from Brasília, one of the must beautiful cities in the world.
- A sports fan, former handball player (College athlete).
- A language enthusiast. I speak English, Portuguese, Spanish and now I'm trying to learn French.

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

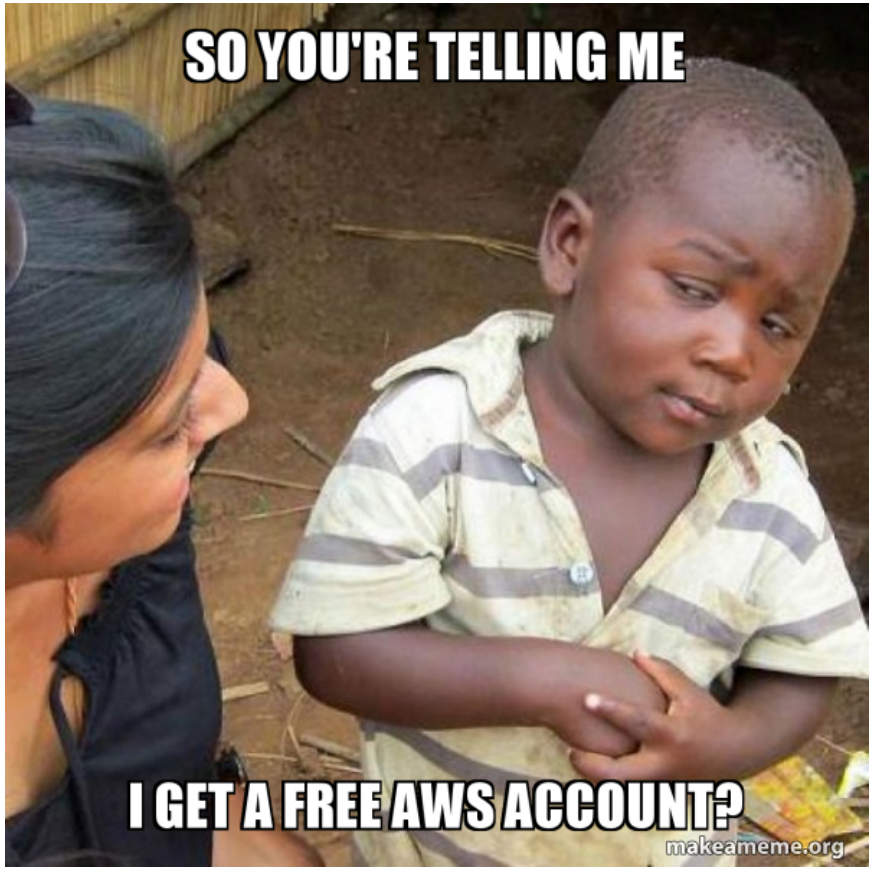
Day 1

Agenda

- Introduction to Containers and Docker
- Kubernetes and EKS
- Environment Setup
    - Exploring and Deploying
- Exploring the cluster with kubectl
- Deploying an application with Kubernetes

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-vls-dub/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

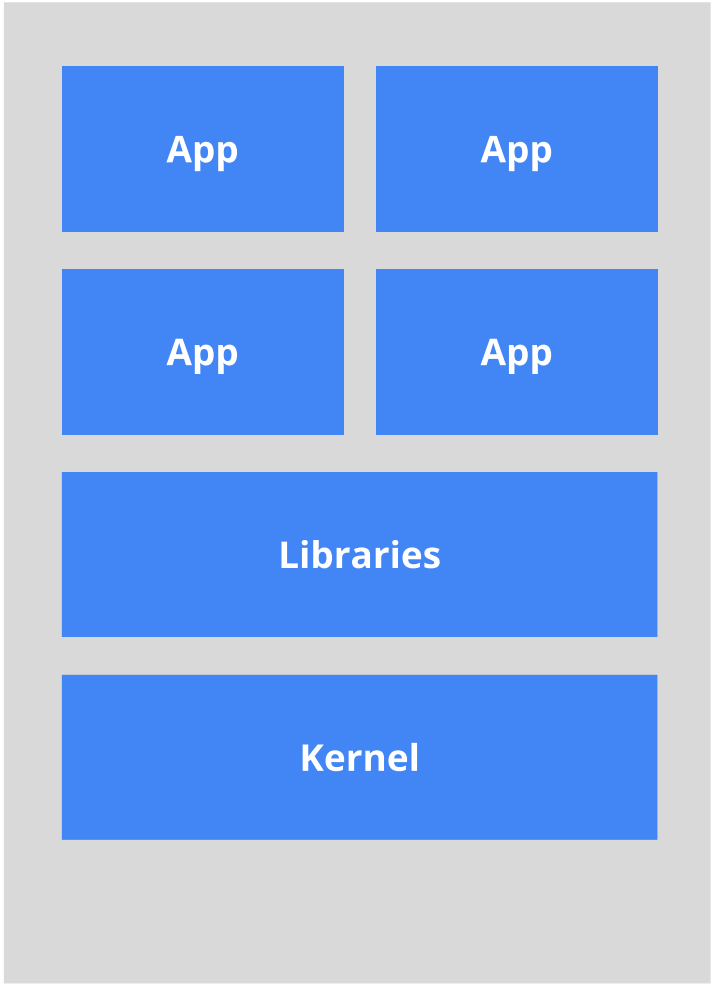
- What is a container?
- What is Docker? what benefits does it provide?
  - Consistent environment
  - Portability of code, runtime, system tools, system libraries, etc.
  - Version control
  - Lightweight

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

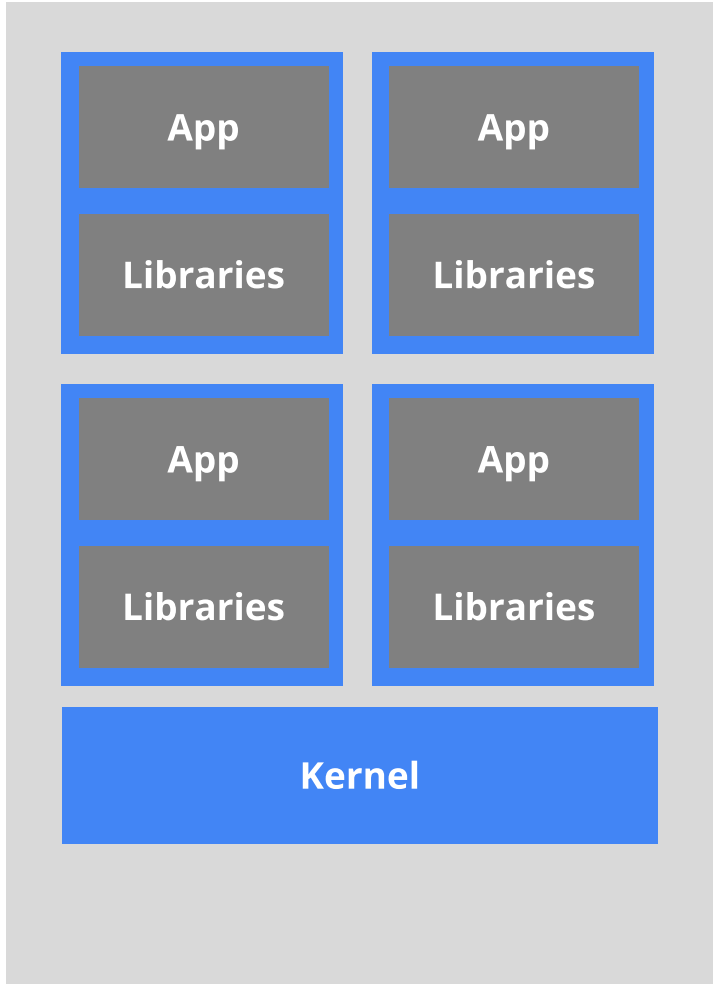
Why containers

The old way: Applications on host



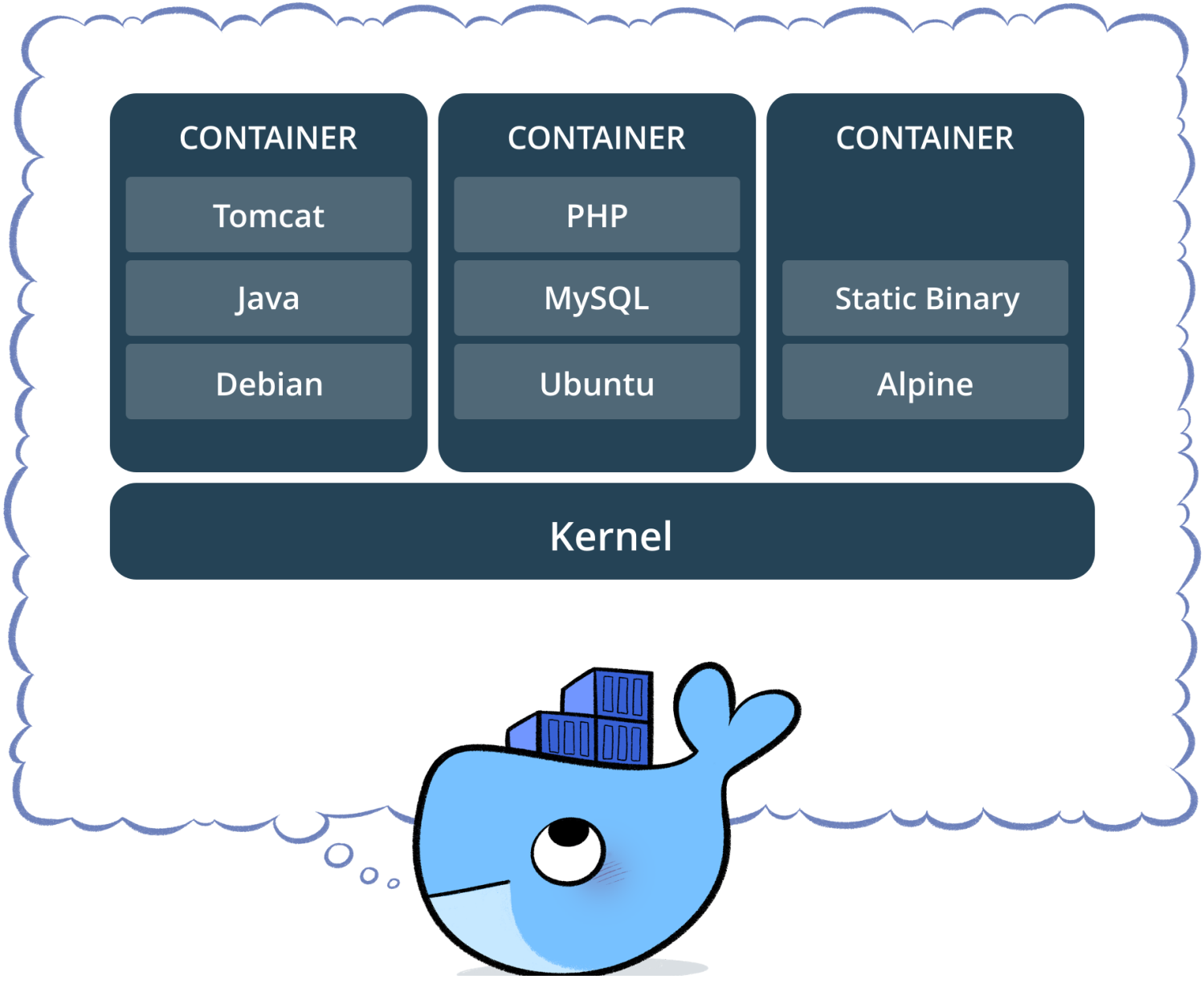
Heavyweight, non-portable
Relies on OS package manager

The new way: Deploy containers

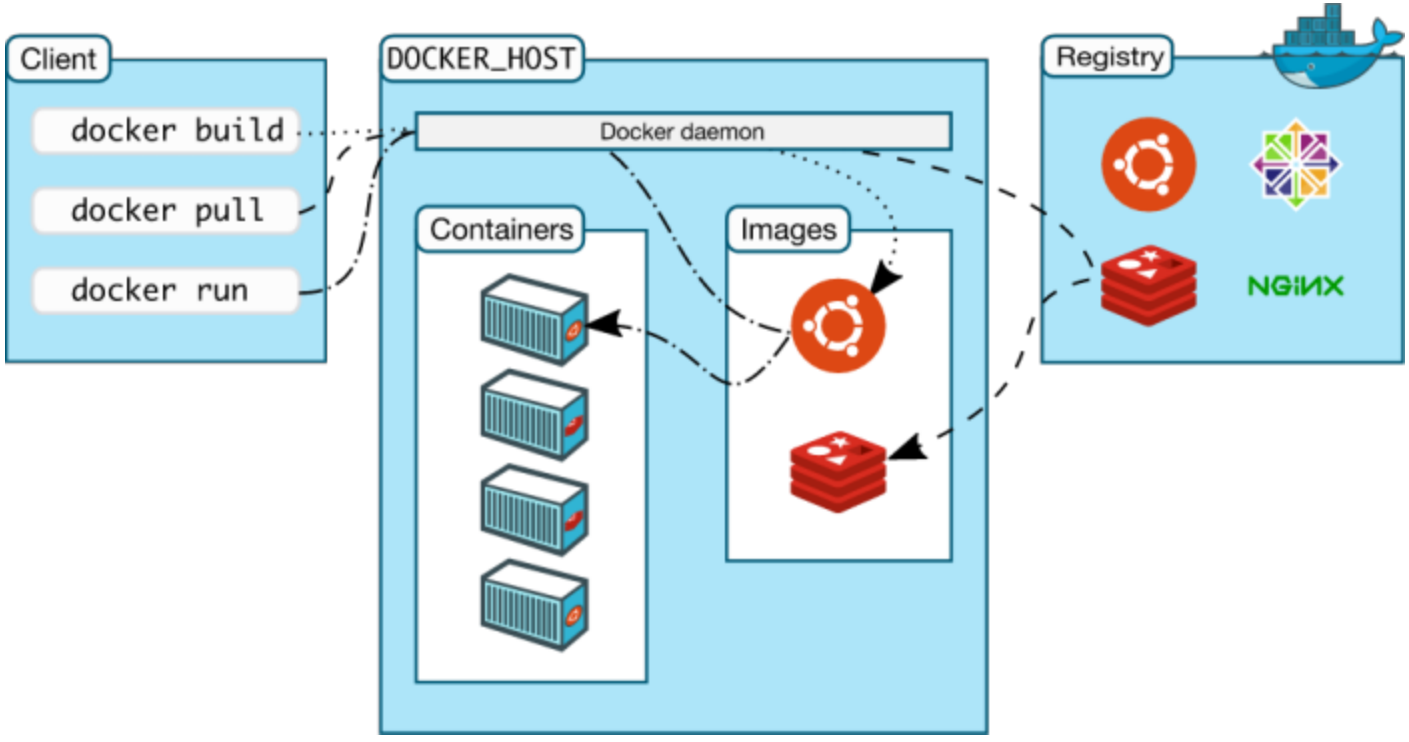


Small and fast, portable
Uses OS-level virtualization

Container Overview



Container Lifecycle

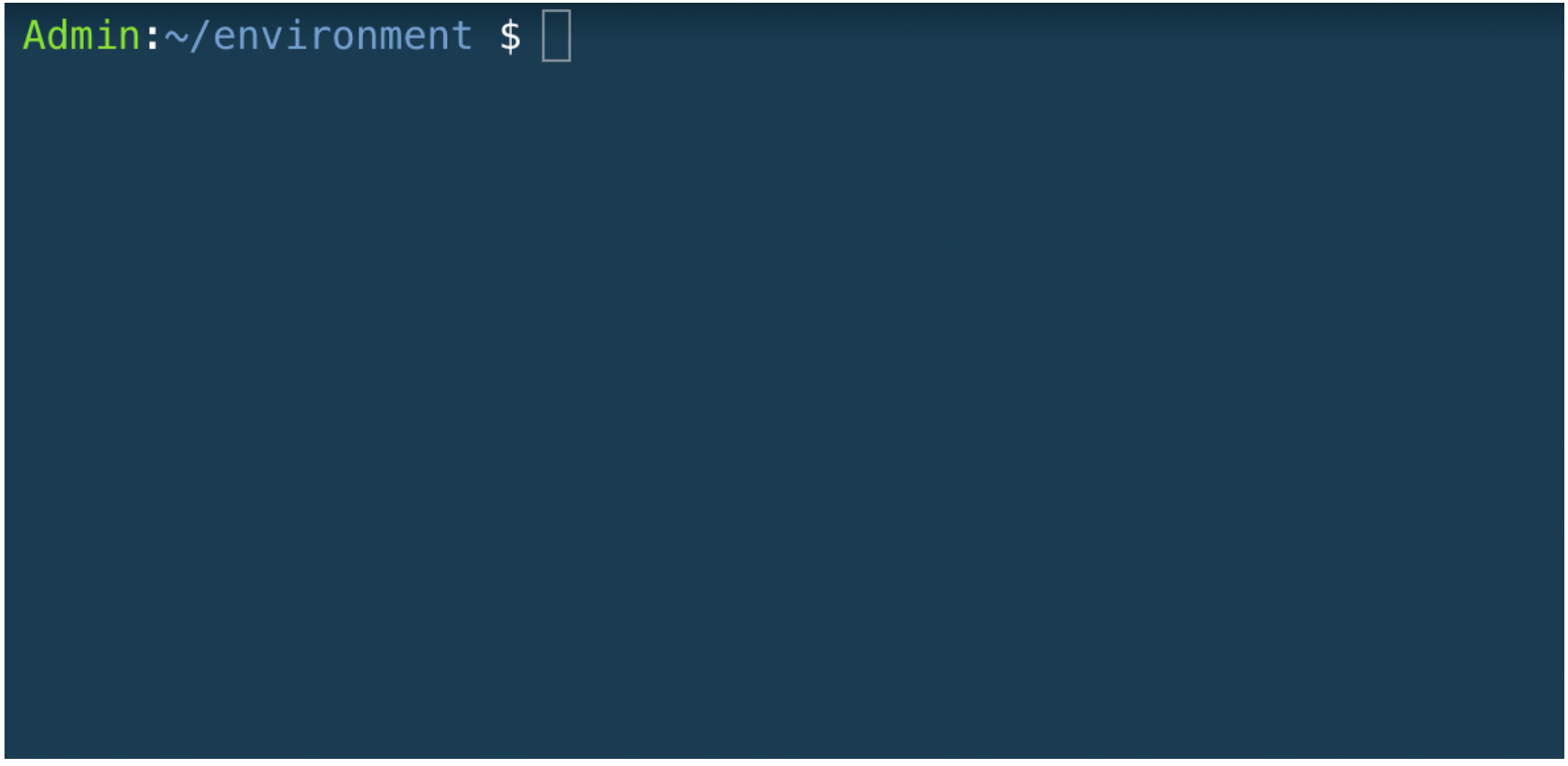


Docker commands - pull

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-vls-dub/eks/tree/master/labs/01-docker

Dockerfile

Dockerfile contains a list of instructions to build a container image

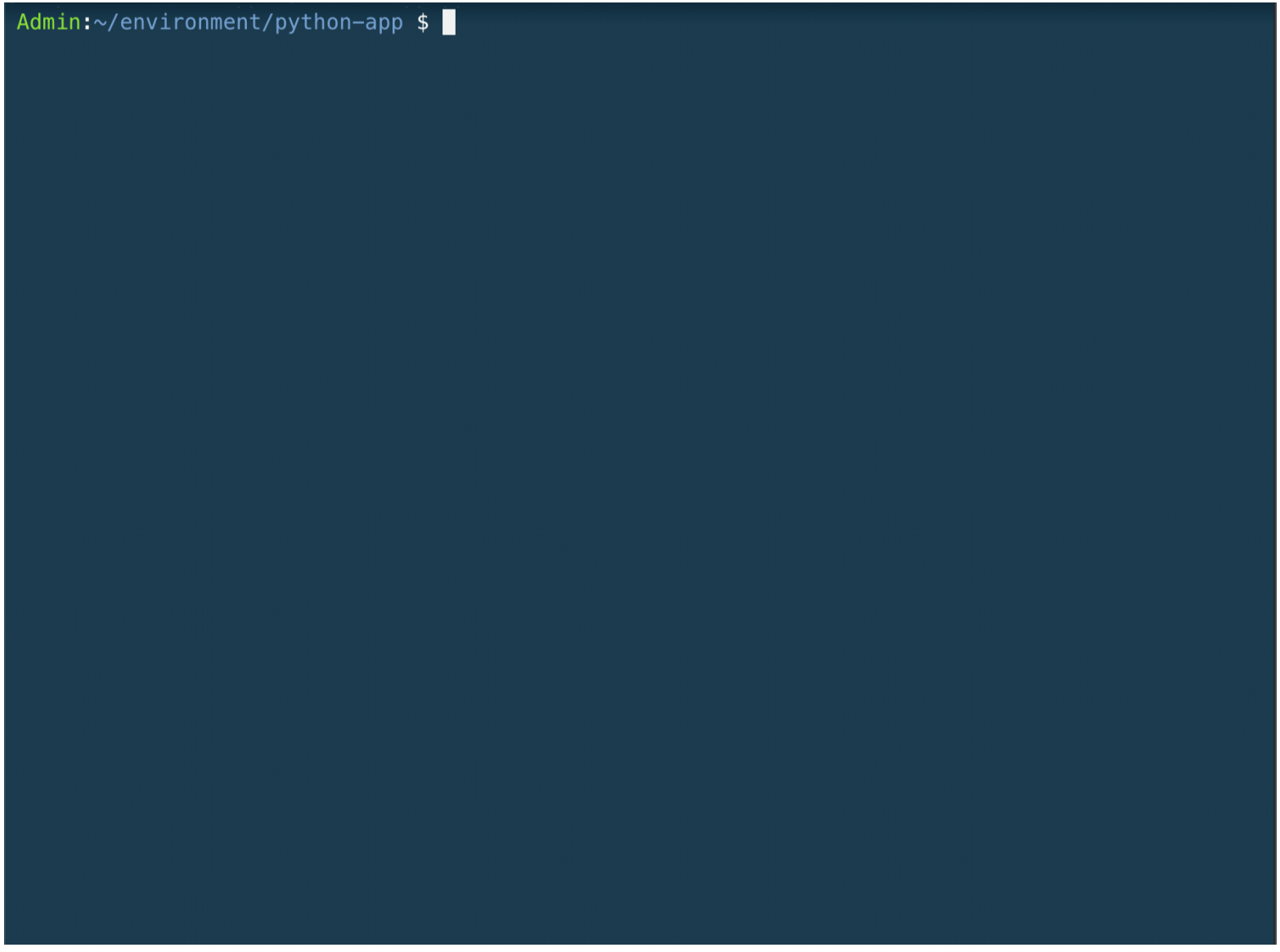
`labs/01-docker/Dockerfile`

Dockerfile ×

external > labs > 01-docker > Dockerfile

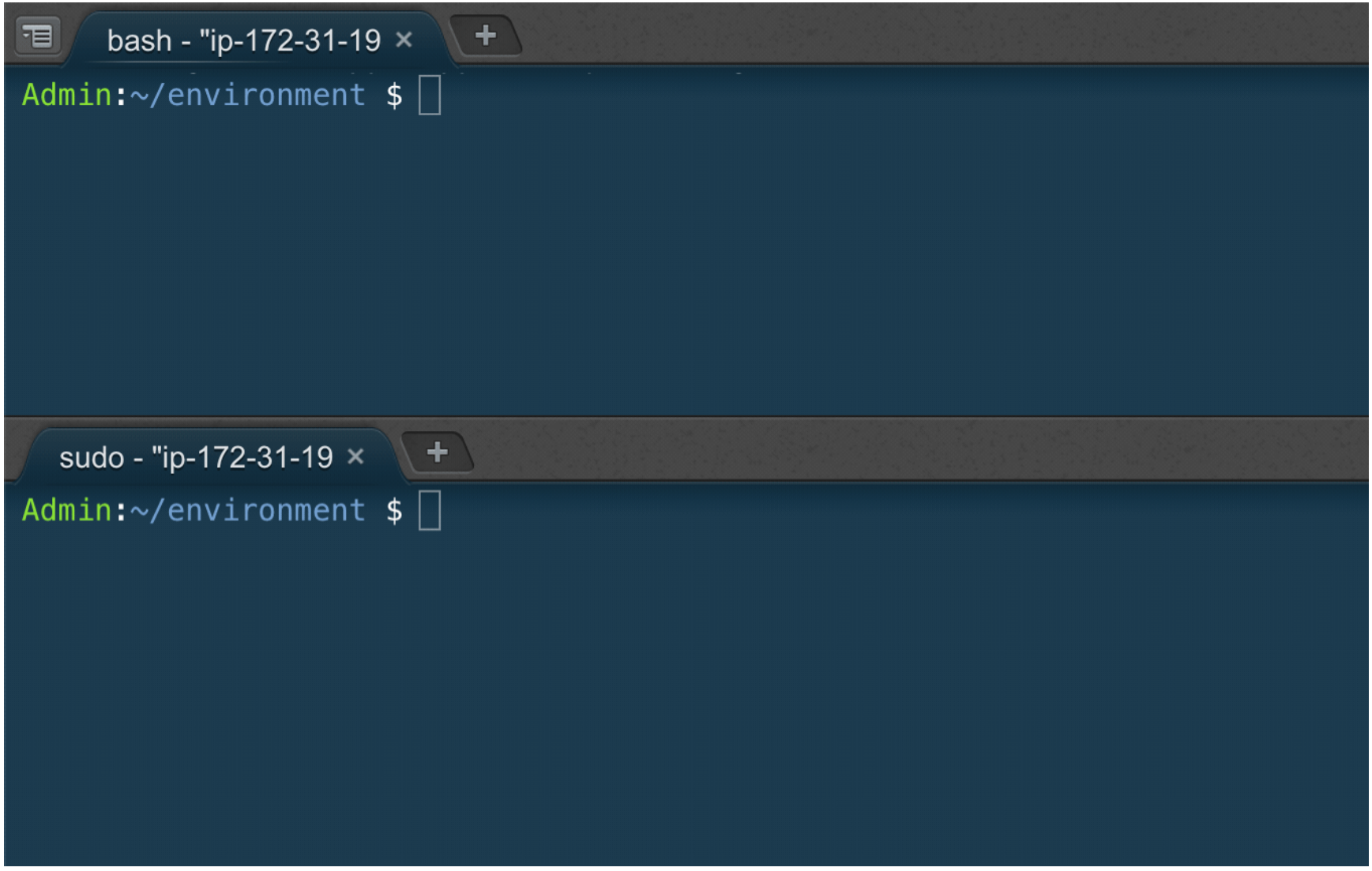
```
1   FROM python:3.6
2
3   COPY requirements.txt requirements.txt
4
5   RUN pip install --no-cache -r requirements.txt
6
7   EXPOSE 5000
8
9   CMD ["python", "server.py"]
10
11  COPY . .
12
```

Docker build



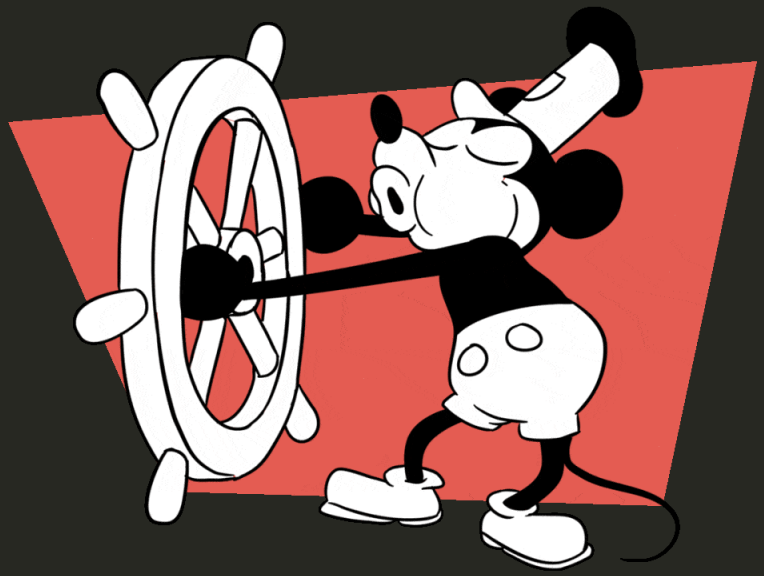
Docker run

Finally we can run a container from the image



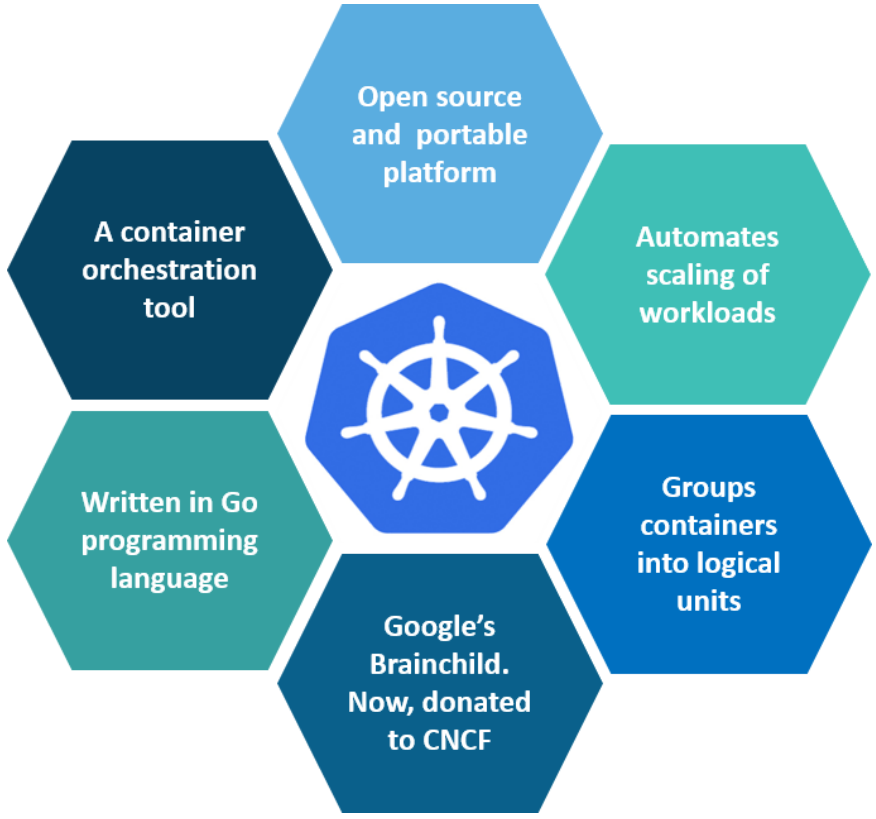
Before we move on, any questions?

What is



Kubernetes?

What is Kubernetes?



Why Kubernetes?

Why Kubernetes?

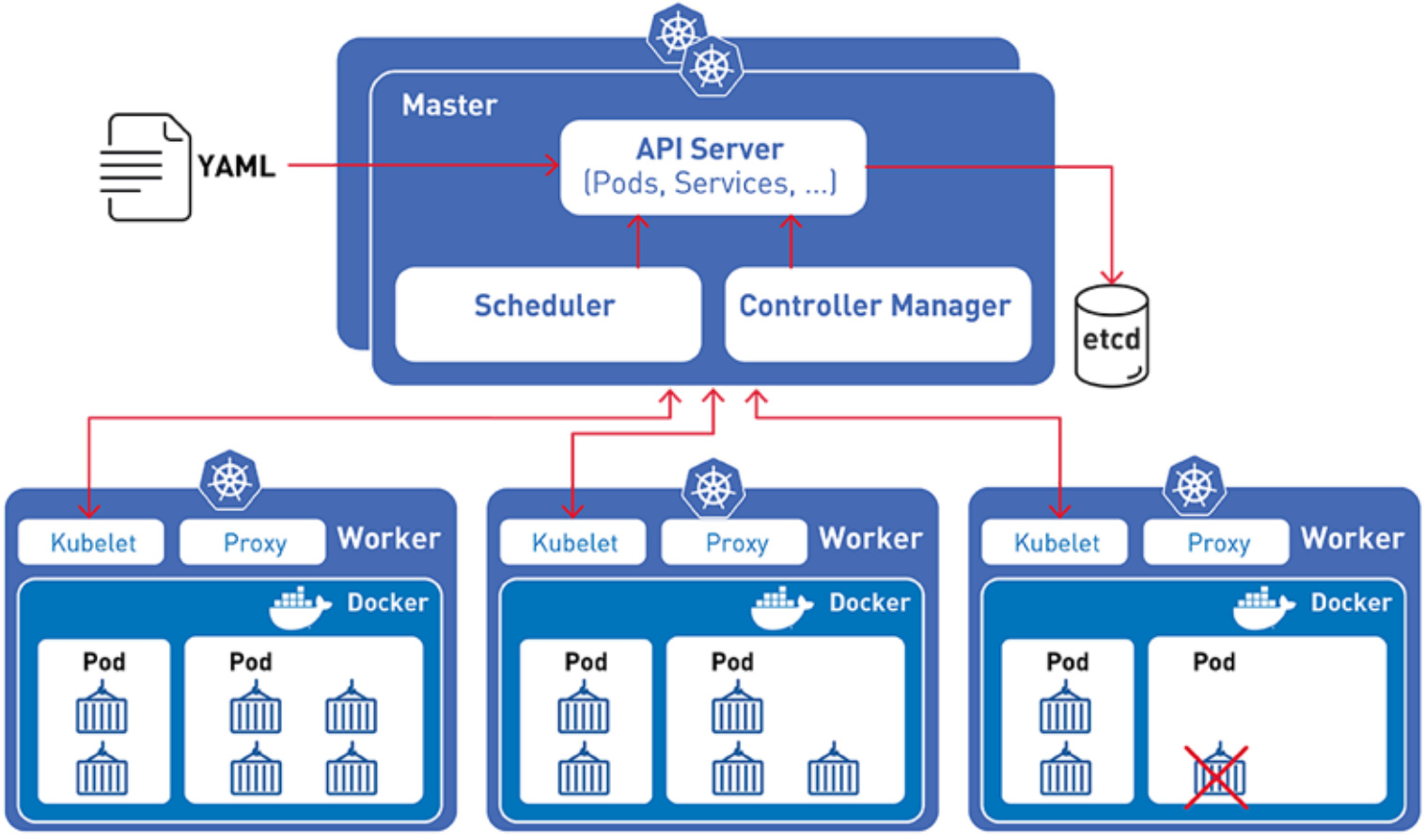
Why Kubernetes?

- 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.
- It manages thousands of containers that run the applications and ensure that there is no downtime.
- Kubernetes is an online, self-healing system. Self-Healing Systems

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



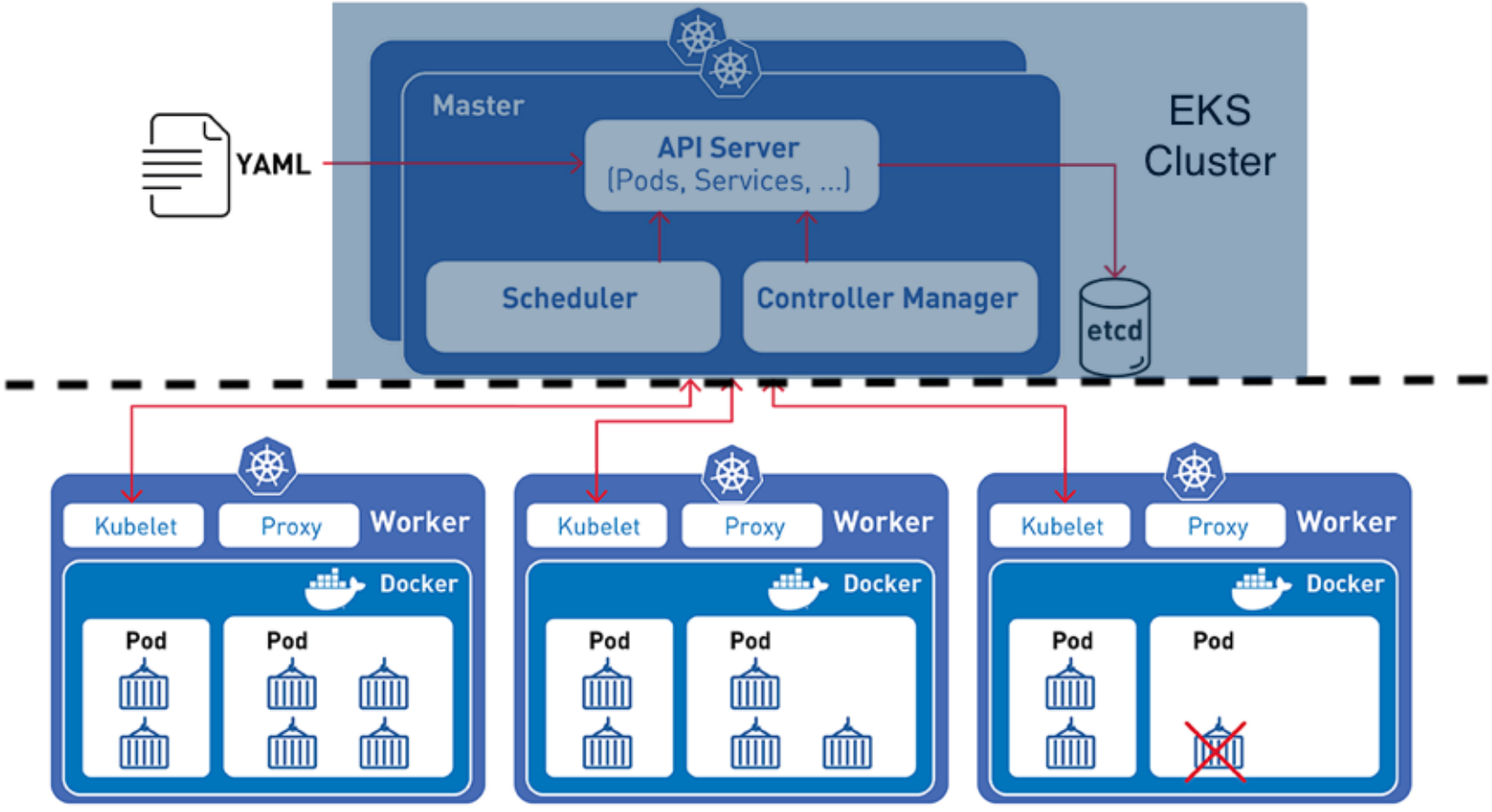
EKS

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?



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 1.16 --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

Let's take a break while that runs, see you in 10 minutes...

Let's talk about kubectl

- Is a powerful tool used to create objects and interact with the Kubernetes API.
- Everything contained in Kubernetes is represented by a RESTful resource. (Kubernetes objects)
- Each Kubernetes object exists at a unique HTTP path; for example, https://your-k8s.com/api/v1/namespaces/default/pods/my-pod
- The kubectl command makes HTTP requests to these URLs to access the Kubernetes objects that reside at these paths.

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

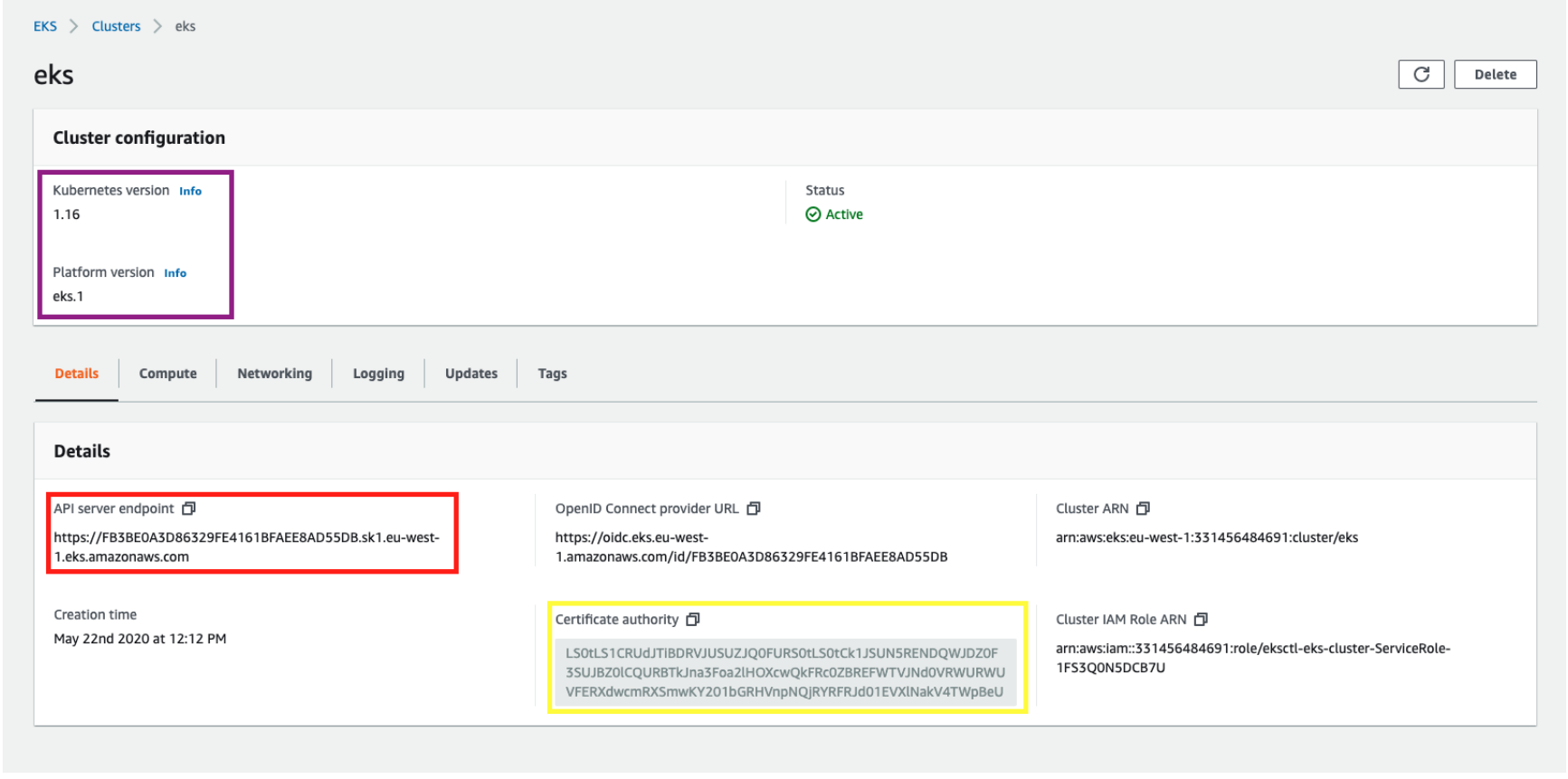
1 - Head to the EKS service in the AWS Web Console

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

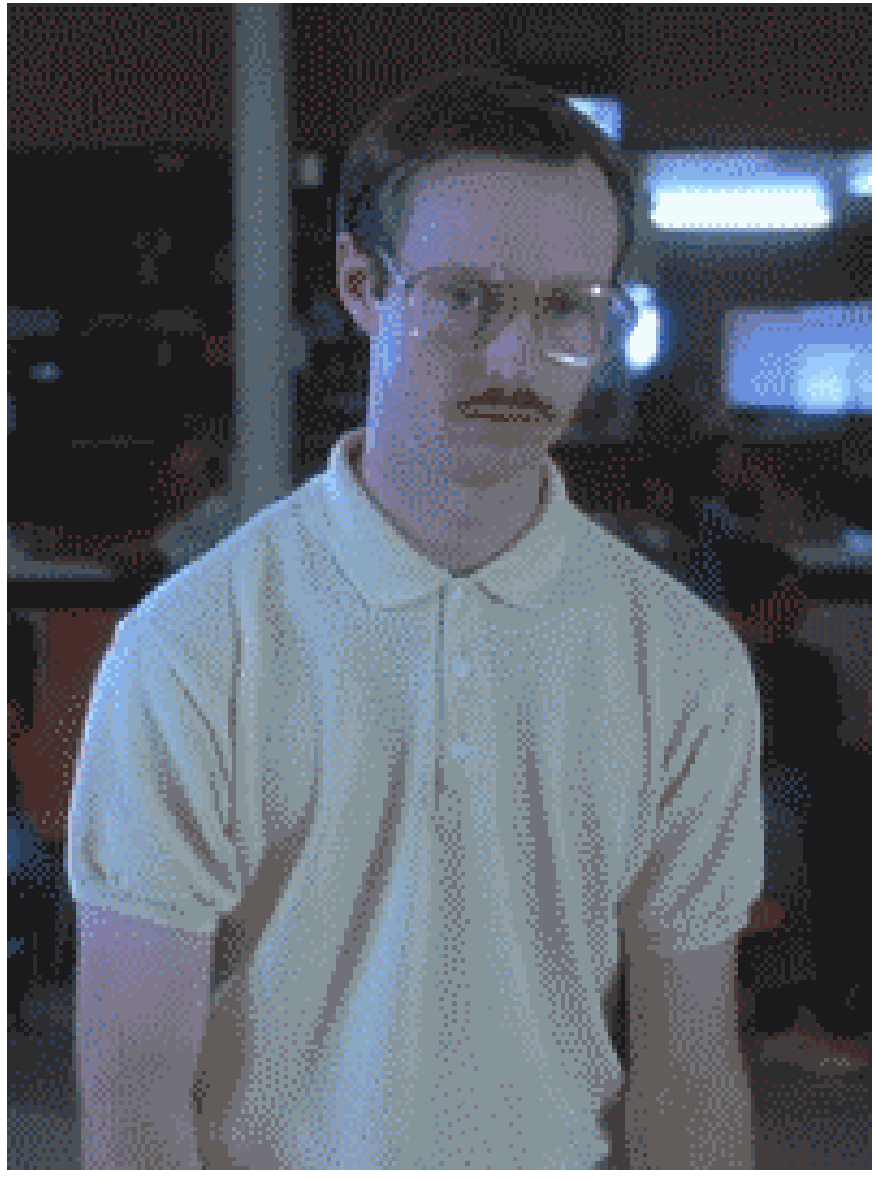
Checking your EKS cluster

- Listing your cluster with eksctl

```
eksctl get clusters
```

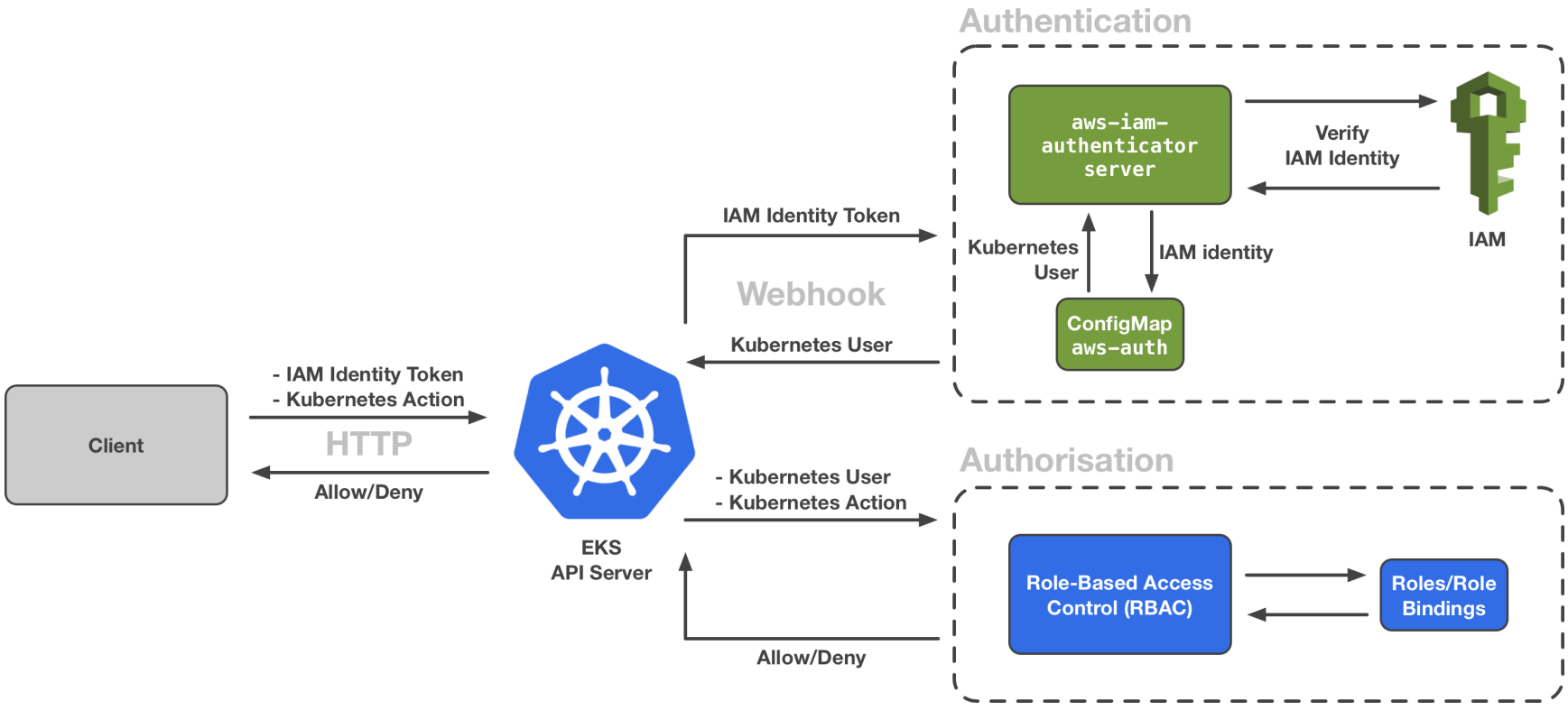


Successfully Create and Connect to EKS Cluster



Connecting to your K8s/EKS Cluster

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
```

kubectl configuration - eksctl

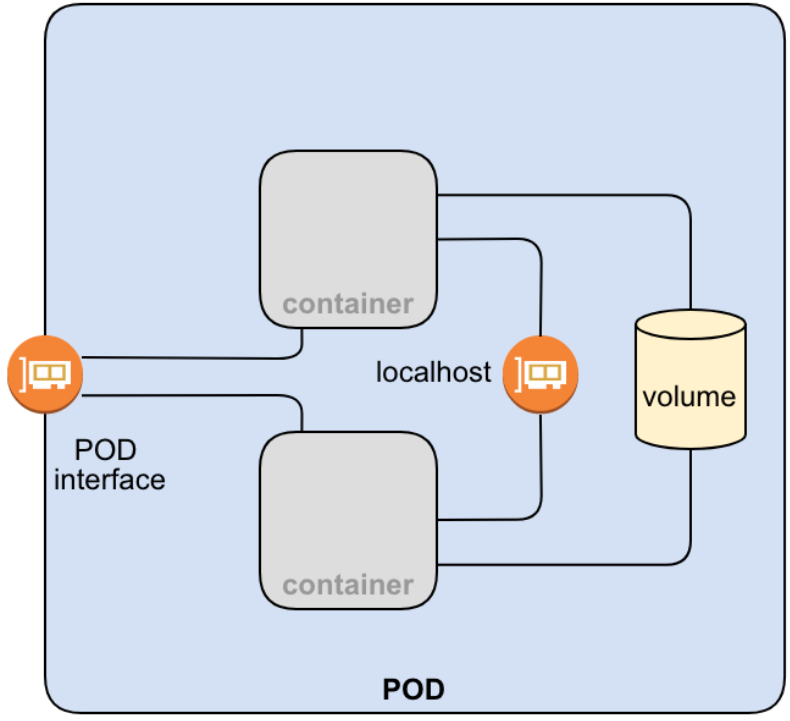
```
sampaig@88e9fe4f024f ~ kubectl config view --minify
apiVersion: v1
clusters:
- cluster:
    certificate-authority-data: DATA+OMITTED
    server: https://FB3BE0A3D86329FE4161BFAEE8AD55DB.sk1.eu-west-1.eks.amazonaws.com
  name: eks.eu-west-1.eksctl.io
contexts:
- context:
    cluster: eks.eu-west-1.eksctl.io
    user: gustavo.sampaio@eks.eu-west-1.eksctl.io
  name: gustavo.sampaio@eks.eu-west-1.eksctl.io
current-context: gustavo.sampai
kind: Config
preferences: {}
users:
- name: gustavo.sampaio@eks.eu-west-1.eksctl.io
  user:
    exec:
      apiVersion: client.authentication.k8s.io/v1alpha1
      args:
      - token
      - -i
      - eks
      command: aws-iam-authenticator
      env:
      - name: AWS_STS_REGIONAL_ENDPOINTS
        value: regional
      - name: AWS_DEFAULT_REGION
        value: eu-west-1
```

Before we move on, any questions?

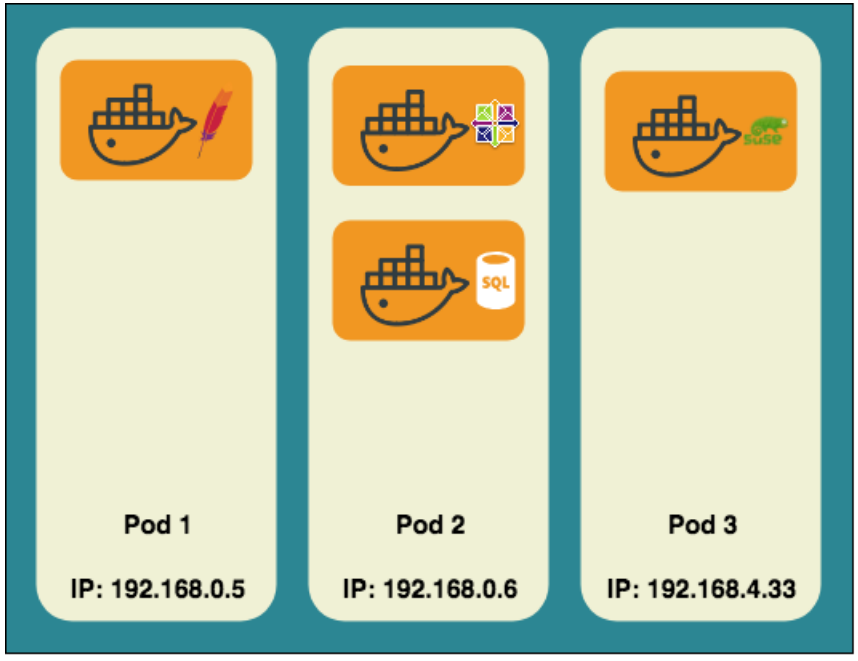
Pod

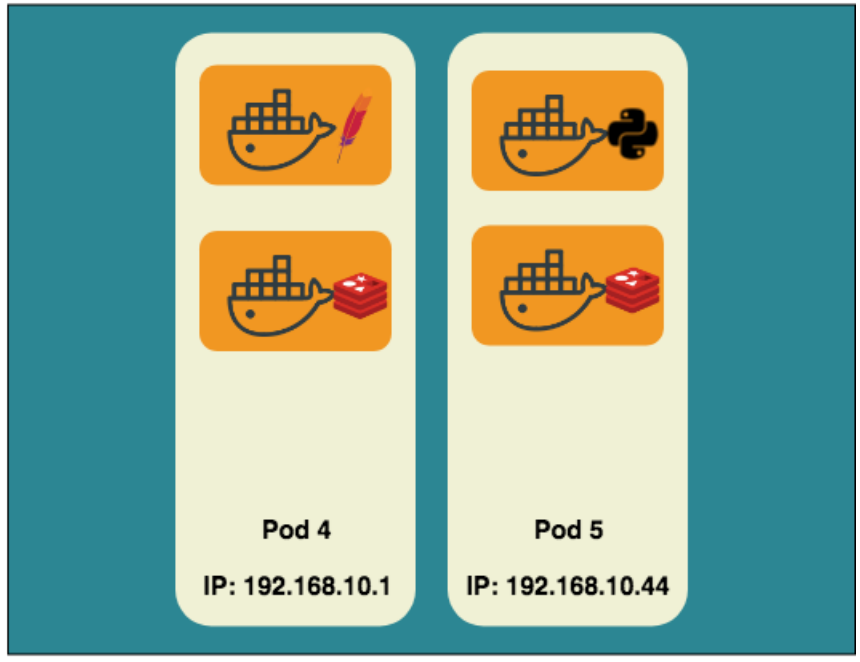
What is a Pod?

- 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-vls-dub/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>
[...]
```

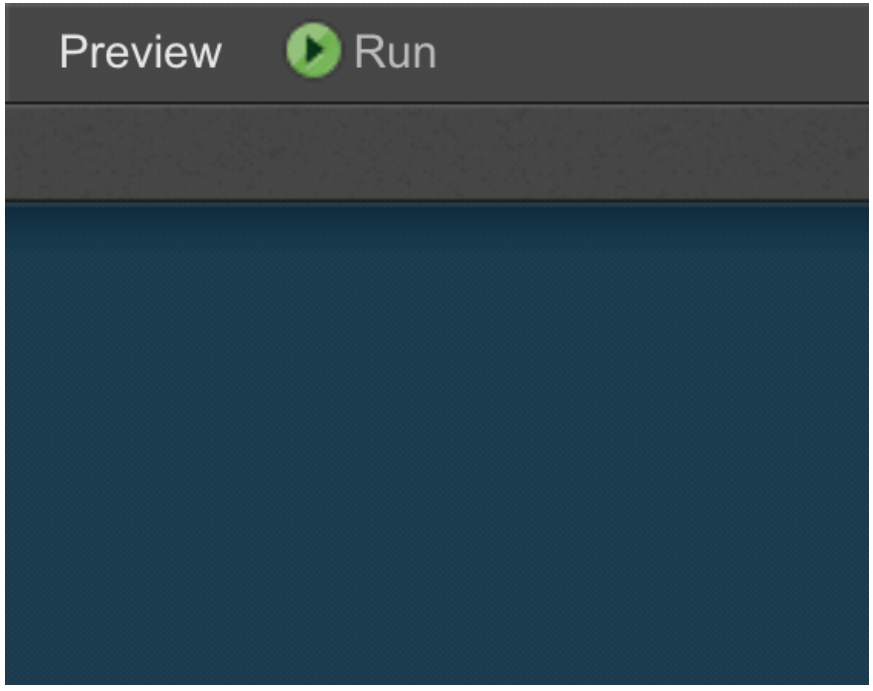
Lab 2: Check the Pod

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-vls-dub/eks/tree/master/labs/03-more-pods

End of Day 1

Questions?

Thank you!