Getting Started with Kubernetes on AWS

Brought to you by the AWS Sydney Cloud Support Team

Welcome and thank you!

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

The Team

Day 1

Agenda

- Introduction to Containers
- Docker Overview
- Kubernetes Explained
- EKS - Exploring and Deploying
- Environment Setup
- Launching a Kubernetes Cluster

Before we get started

Adding credits to your AWS Account

Adding credits to your AWS Account

1. Open the Credits page of the Billing and Cost Management console.
2. In the Promo Code box, type the promotional code.
3. In the box labeled Please type the characters as shown above, type the code.
4. Choose Redeem.

Adding credits to your AWS Account

1. Open the Credits page of the Billing and Cost Management console.
2. In the Promo Code box, type the promotional code.
3. In the box labeled Please type the characters as shown above, type the code.
4. Choose Redeem.

Note

The credit amount will cover running the resources we need for the sessions, other resources you may wish to use in your AWS account may not be covered by this.

You are responsible for deleting the resources on Monday. We'll remind you again next week.

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

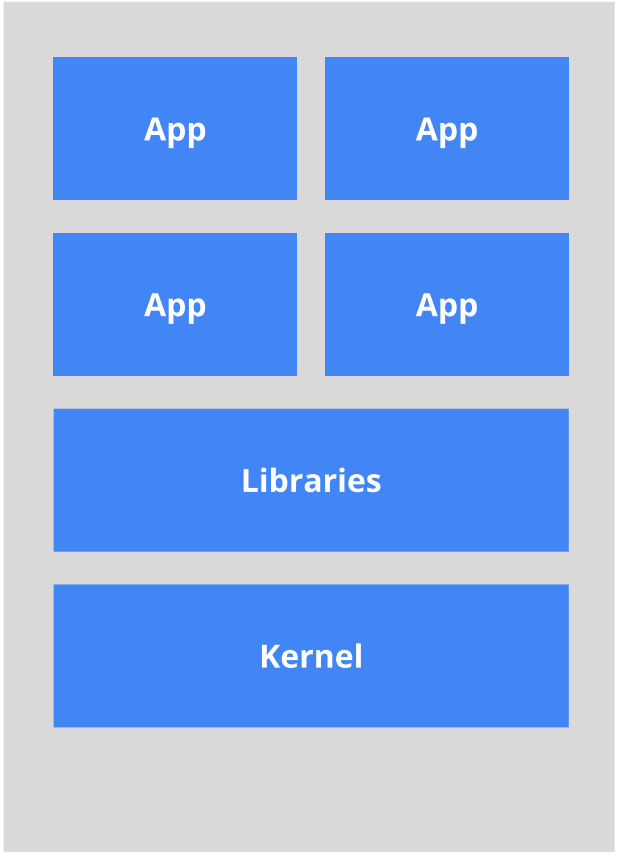
- What is a container?
- What is Docker? what benefits does it provide?
  - Consistent environment
  - Portability of code, runtime, system tools, system libraries, etc.
  - Version control
  - Lightweight

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

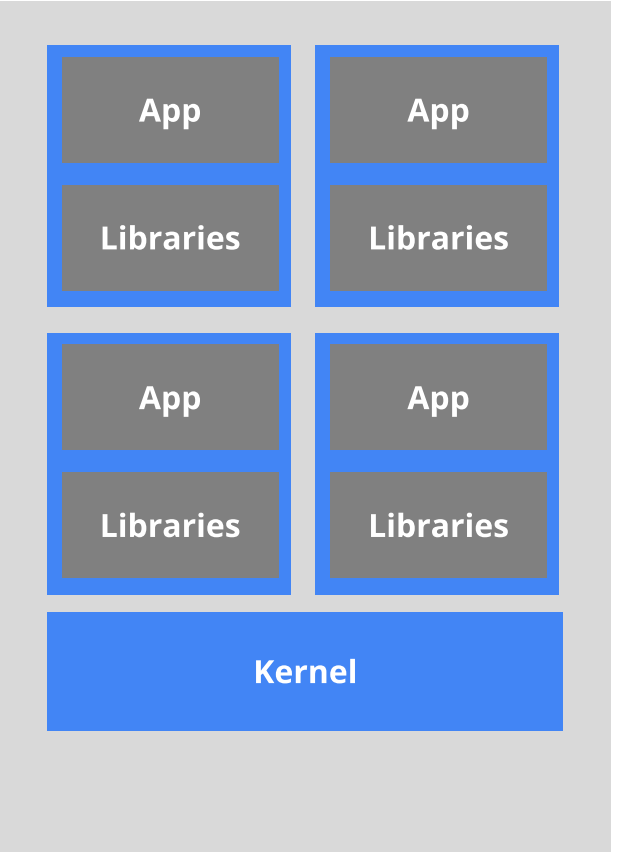
Why containers

The old way: Applications on host



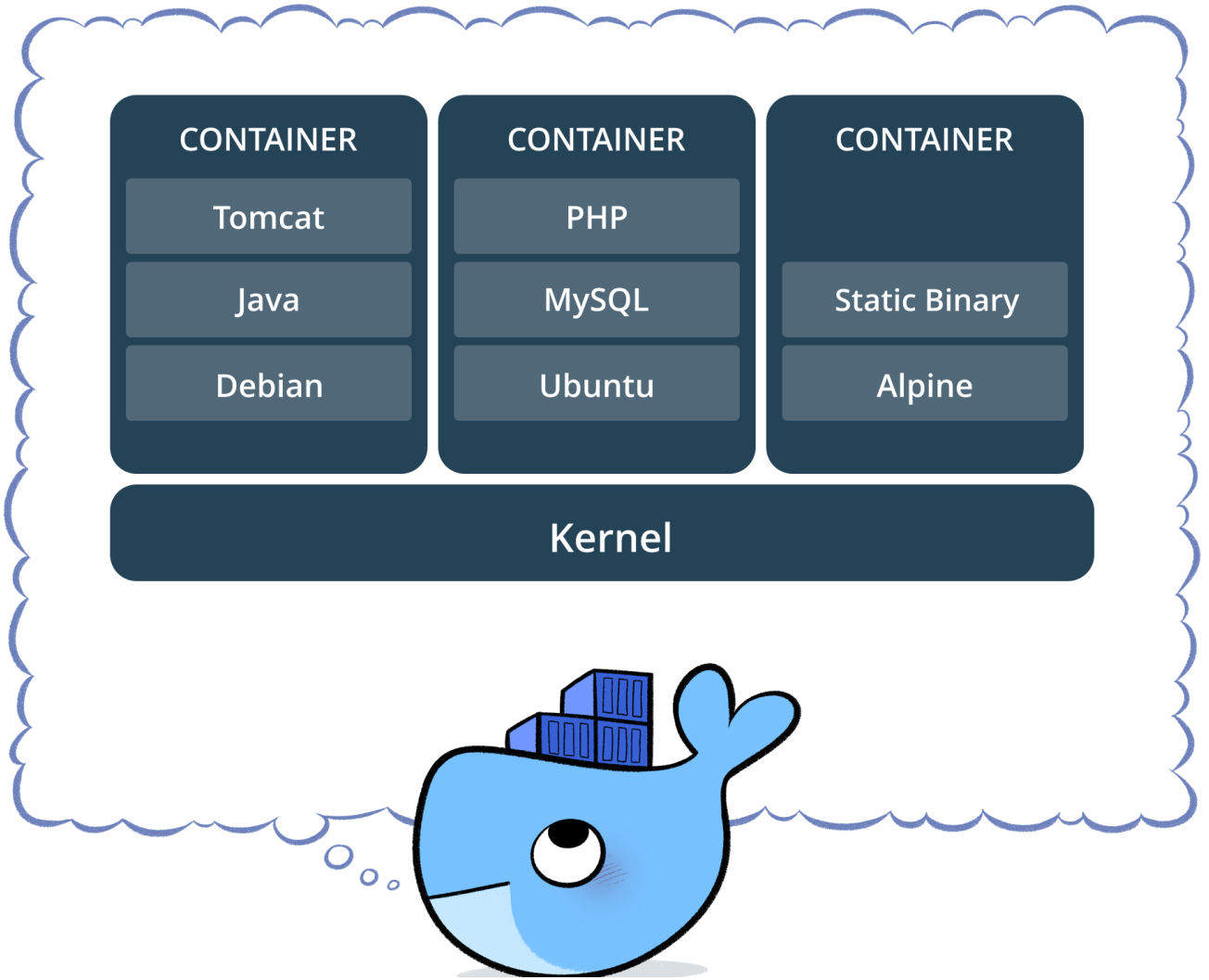
Heavyweight, non-portable
Relies on OS package manager

The new way: Deploy containers

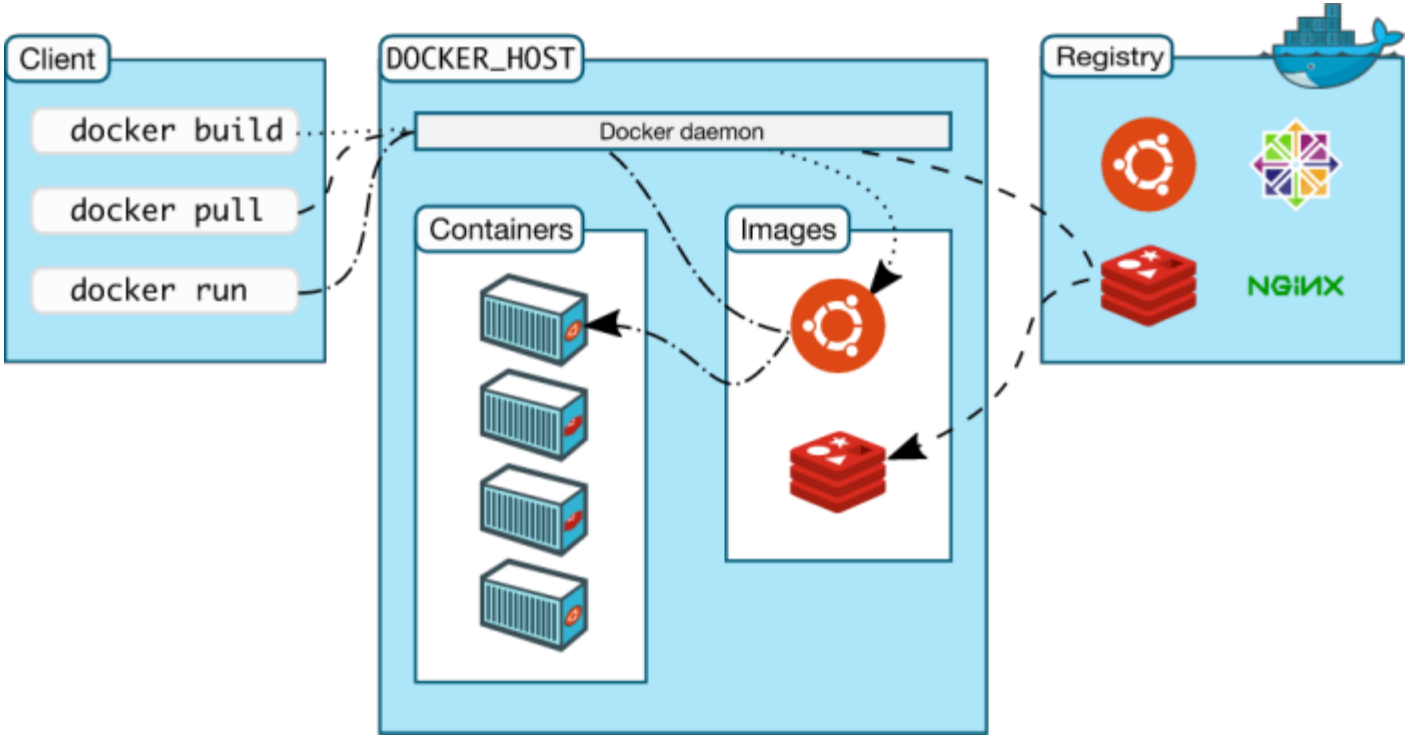


Small and fast, portable
Uses OS-level virtualization

Container Overview



Container Overview



Docker commands - pull

Pull an image from an image registry

```
$ docker pull nginx
  Using default tag: latest
  latest: Pulling from library/nginx
  fc7181108d40: Already exists
  d2e987ca2267: Already exists
  0b760b431b11: Already exists
  Digest: sha256:48cbeee0cb0a3b5e885e36222f969e0a2f41819a68e07aeb6631ca7cb356fed1
  Status: Downloaded newer image for nginx:latest
```

Link to GIF

Docker commands - run

Running a container from an image

```
$ docker run -d -p 80:80 nginx
  38d7b167668b0d036b3b1a5dee959b1207f8f7ad41f664bd974743a85a6fa5af
$ curl -I localhost
  HTTP/1.1 200 OK
  Server: nginx/1.17.1
  Date: Thu, 11 Jul 2019 02:00:16 GMT
  Content-Type: text/html
  Content-Length: 612
  Last-Modified: Tue, 25 Jun 2019 12:19:45 GMT
  Connection: keep-alive
  ETag: "5d121161-264"
  Accept-Ranges: bytes
```

Link to GIF

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

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

Dockerfile

A Dockerfile contains a list of instructions to build a container image

Dockerfile

```
1   # Our base image
2   FROM python:alpine
3   # Copy requirements.txt first
4   COPY requirements.txt .
5   # Install Python modules needed by the Python app
6   RUN pip install --no-cache-dir -r requirements.txt
7   # The port number the container should expose
8   EXPOSE 5000
9   # Run the application
10  CMD ["python", "./app.py"]
11  # Copy remaining files required for the app to run
12  COPY . .
```

Docker build

```
$ docker build -t python-app .
  Sending build context to Docker daemon  9.216kB
  Step 1/7 : FROM python:alpine
  ---> 2caaa0e9feab
  Step 2/7 : WORKDIR /usr/src/app
  ---> Running in a37b192f1cc7
  Removing intermediate container a37b192f1cc7
  ---> d39e5d8c764b
  Step 3/7 : COPY requirements.txt .
  [...]
  Step 7/7 : COPY . .
  ---> c292b2b548b4
  Successfully built c292b2b548b4
  Successfully tagged python-app:latest
```

Link to GIF

Docker run

Finally we can run a container from the image

```
$ docker run -p 5000:5000 python-app
  * Serving Flask app "app" (lazy loading)
  * Environment: production
    WARNING: This is a development server. Do not use it in a production deploymen
    Use a production WSGI server instead.
  * Debug mode: off
  * Running on http://0.0.0.0:5000/ (Press CTRL+C to quit)
```

Link to GIF

What is

Kubernetes?

What is Kubernetes?

What is Kubernetes?

Kubernetes is an extensible open-source platform for managing containerized workloads

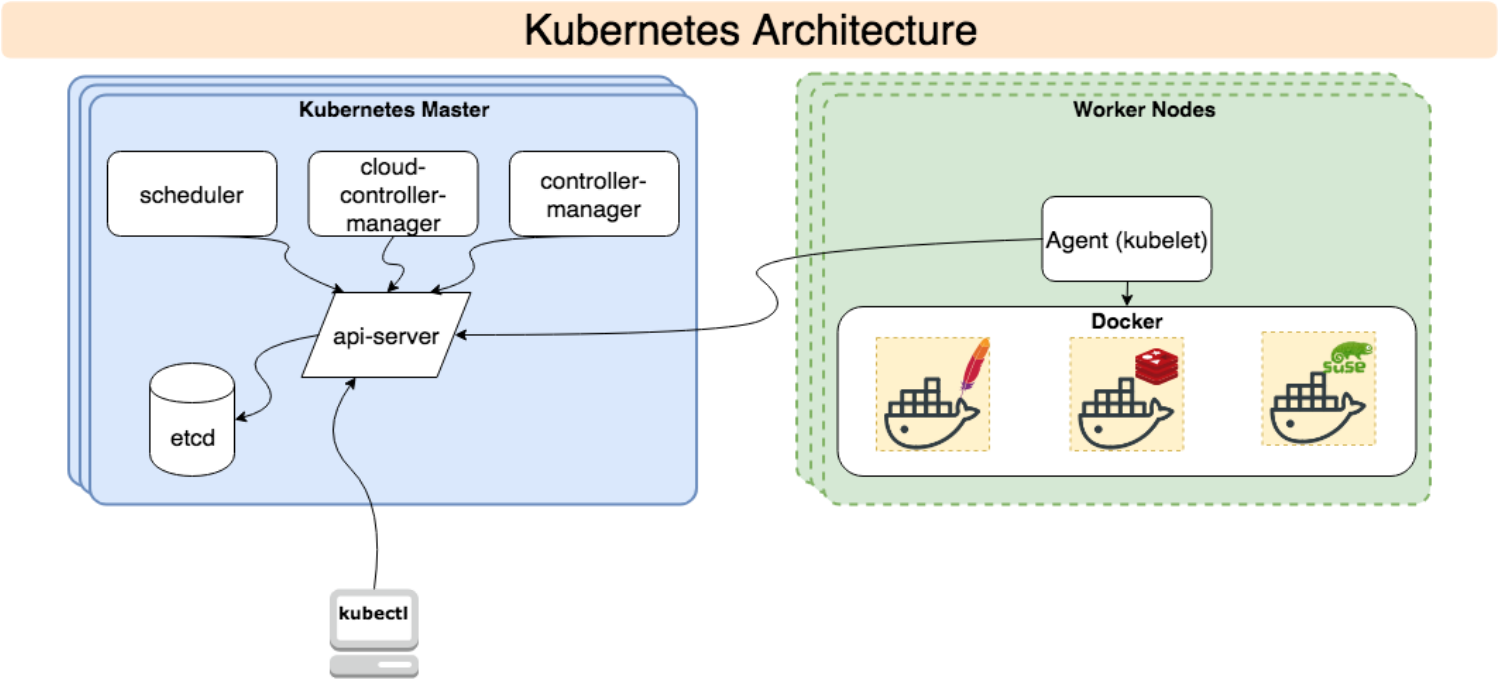
Kubernetes Finally… A True Cloud Platform

Kubernetes is Greek for "helmsman"

Kubernetes a.k.a K8s (pronounced: Kates)

From https://kubernetes.io/

What is Kubernetes?



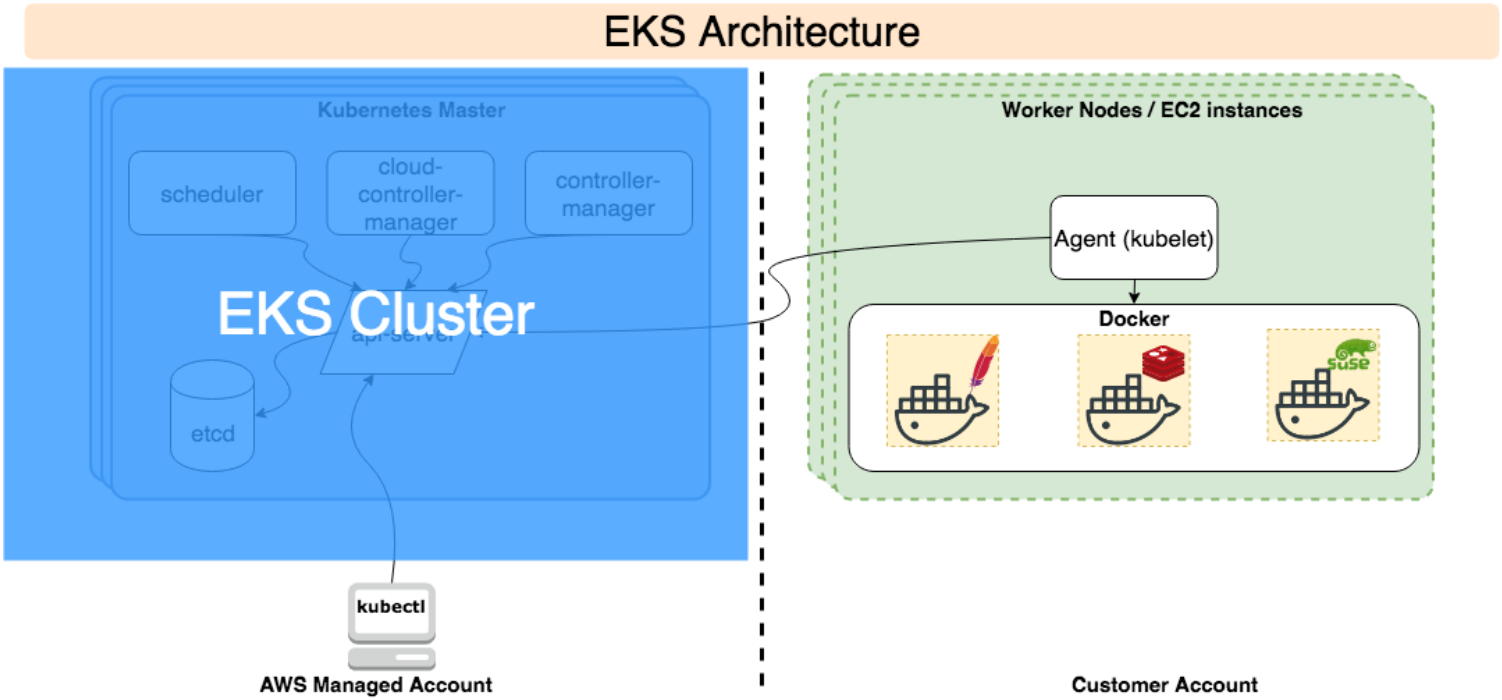
EKS

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

Launching a Kubernetes Cluster / EKS

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

Regions we'll use:

- US East (N. Virginia)
- US West (Oregon)
- Asia Pacific (Singapore)

Creating a Cluster (Step 5)

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

```
eksctl create cluster --ssh-access --version 1.13 --node-type t3.medium --name eks
```

This command is also available on the GitHub page

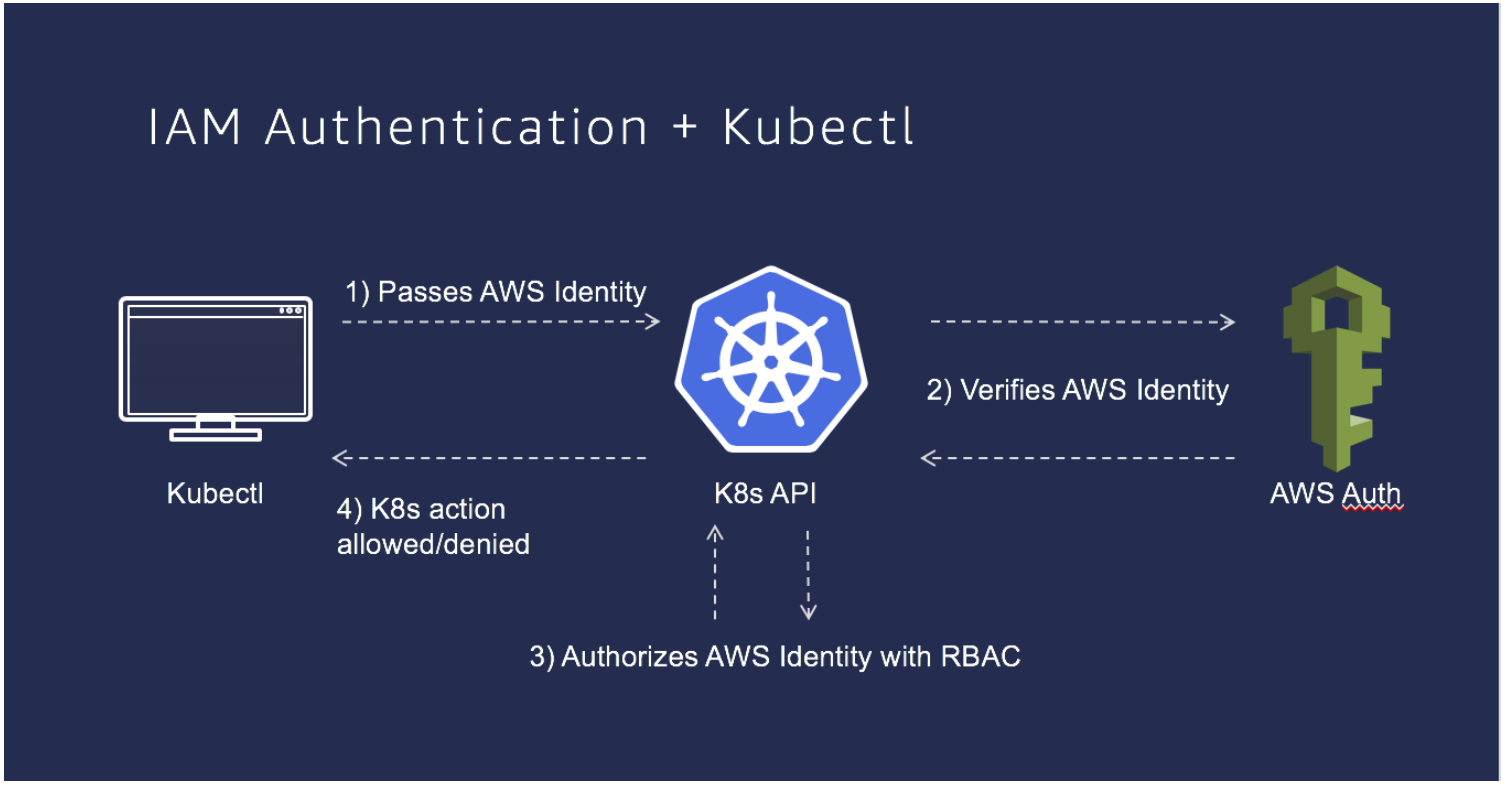
The EKS Cluster will take a approx 15 minutes to create

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



Let's check our new Cluster

Let's check our new Cluster

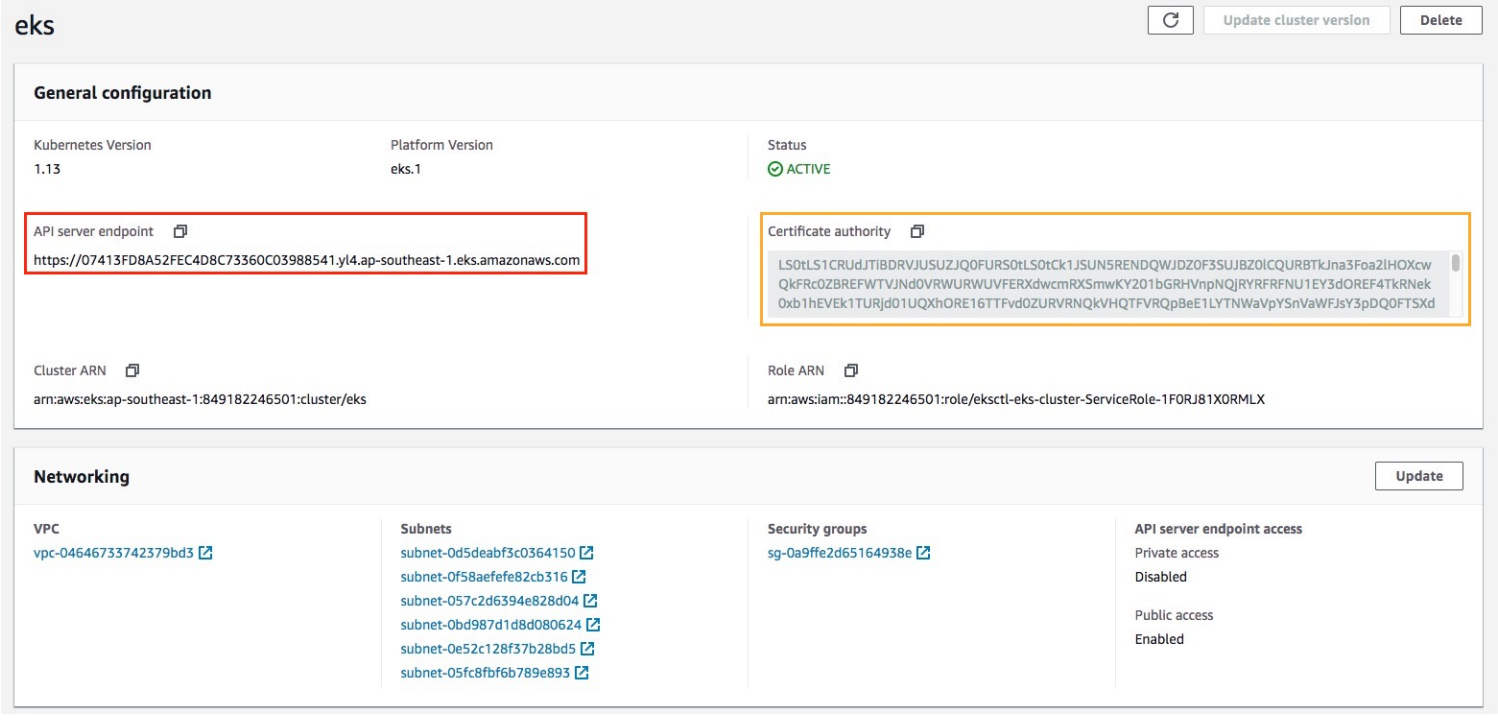
1 - Head to the EKS service in the AWS Web Console

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



Let's check our new Cluster

Let's check our new Cluster

...using kubectl

Do we have Nodes?

```
$ kubectl get nodes
NAME                                                STATUS   ROLES    AGE   VERSION
ip-192-168-29-29.ap-southeast-1.compute.internal   Ready    <none>   7d    v1.13.7
ip-192-168-88-24.ap-southeast-1.compute.internal   Ready    <none>   7d    v1.13.7
```

Link to GIF

Do we have Nodes?

```
$ kubectl get nodes
NAME                                                STATUS   ROLES    AGE   VERSION
ip-192-168-29-29.ap-southeast-1.compute.internal   Ready    <none>   7d    v1.13.7
ip-192-168-88-24.ap-southeast-1.compute.internal   Ready    <none>   7d    v1.13.7
```

Link to GIF

No, I don't have a cluster! Run:

```
eksctl create cluster --ssh-access --version 1.13 --node-type t3.medium --name eks
```

End of Day 1

Questions?

Thank you!