Intro to DevOps

Women Reboot

Maha & Roxana

Cloud Support Engineers @ AWS
Premium Support

What Support Engineering does?

- Cloud Support Engineers are the front line of AWS’ technical support
- Multiple teams, each with specific focus areas.
- What do we do:
  - Customer interactions
  - Training
  - Drive projects
  - Continuous learning

What do we do as Cloud Support Engineers - Deployment

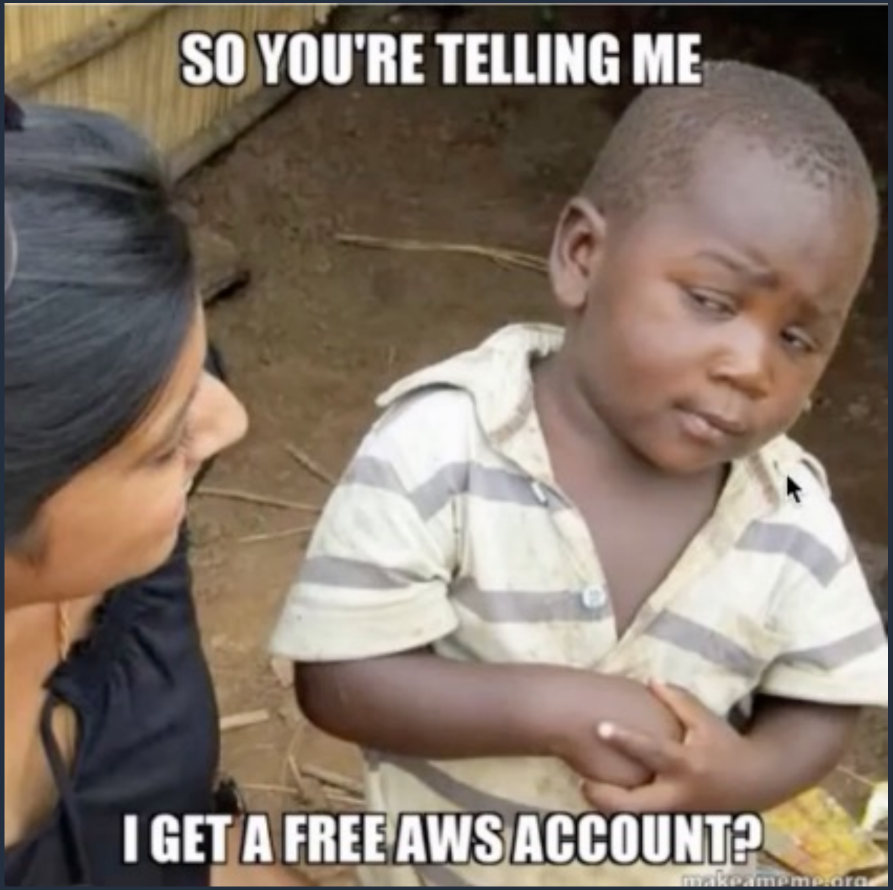
We assist customers to develop services and technologies built on top of AWS Cloud Platform:

- ECS and EKS (container orchestration)
- CloudFormation, Elastic Beanstalk, OpsWorks (Automating operations with Chef)
- CodeDeploy, CodePipeline, CodeCommit (Implementing CI/CD pipelines on AWS)

Agenda:

- Environment Setup
- Introduction to Containers and Docker
- Kubernetes and EKS
  - Architecture
  - Eks
  - Kubectl
- Exploring the cluster with kubectl

Setting up an AWS Account and Environment



Login into your AWS account

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

Note : The account will be deactivated at the end of the event

Launching your lab environment

- Cloud9 environment – this will be where you'll be performing the labs throughout the session

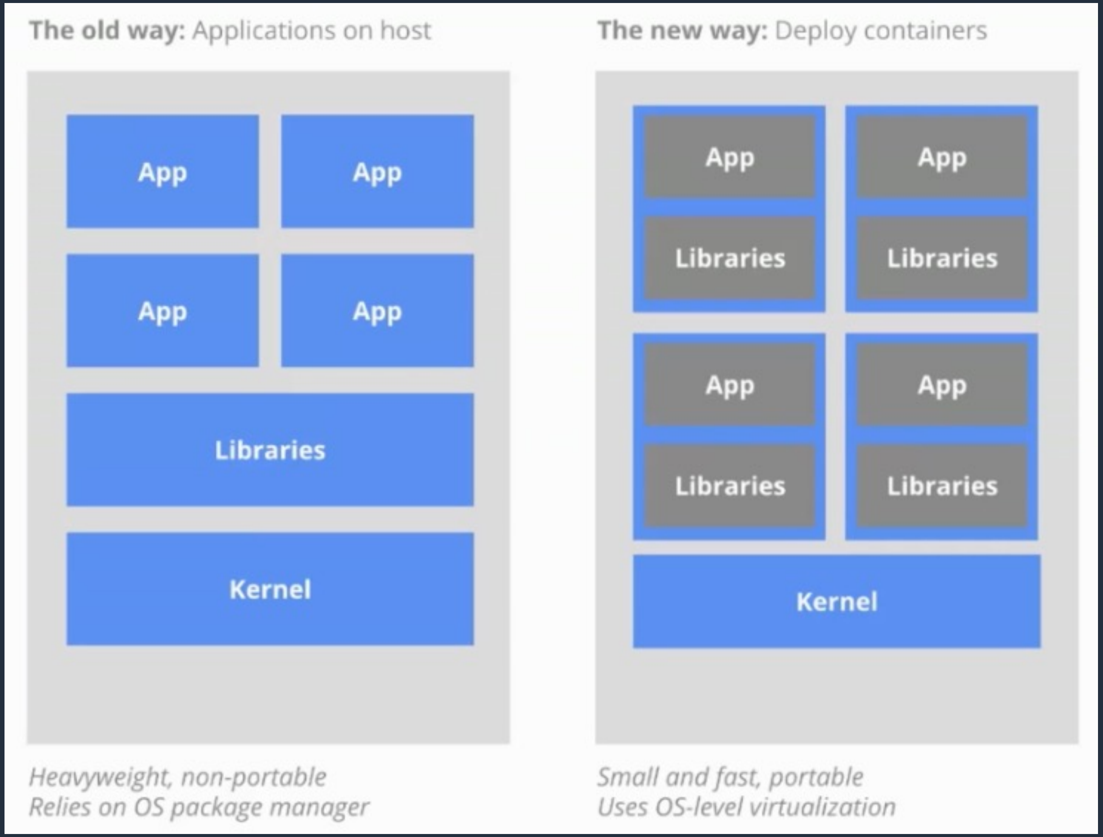
- Follow the steps in the README.md of the below repo to setup the environment

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

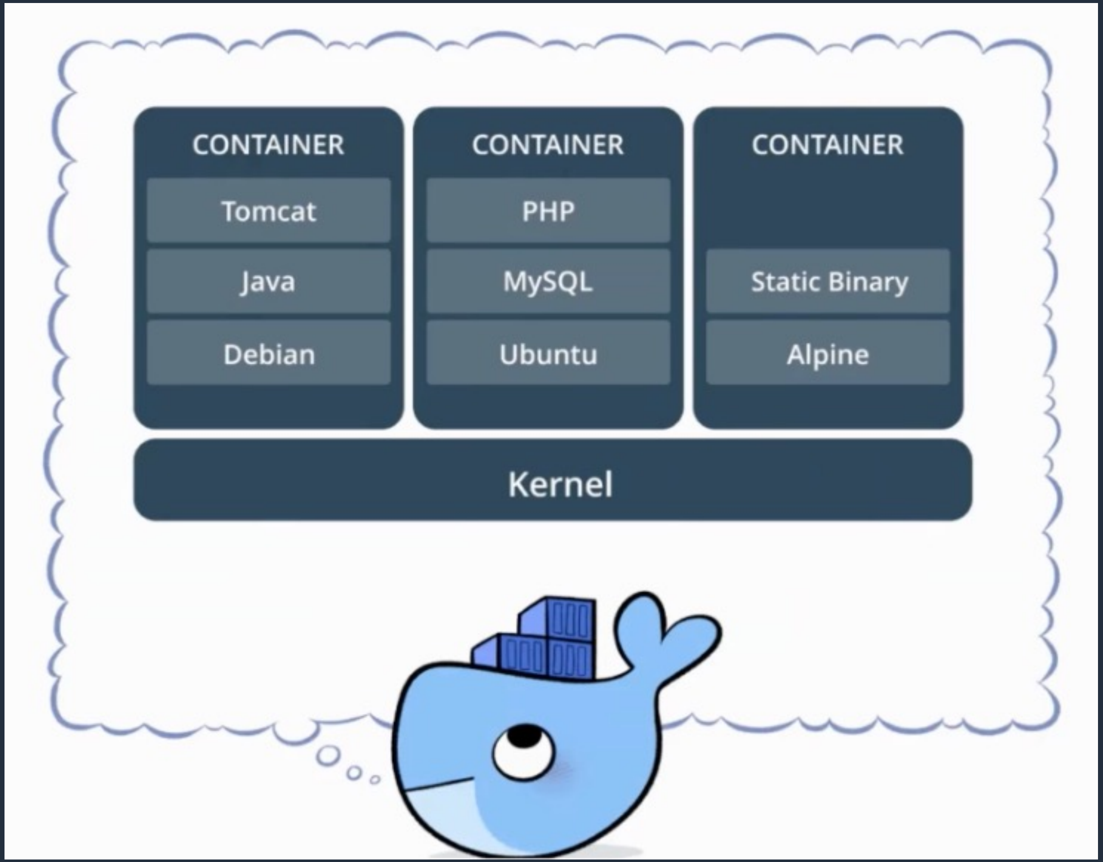
Containers and Docker

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

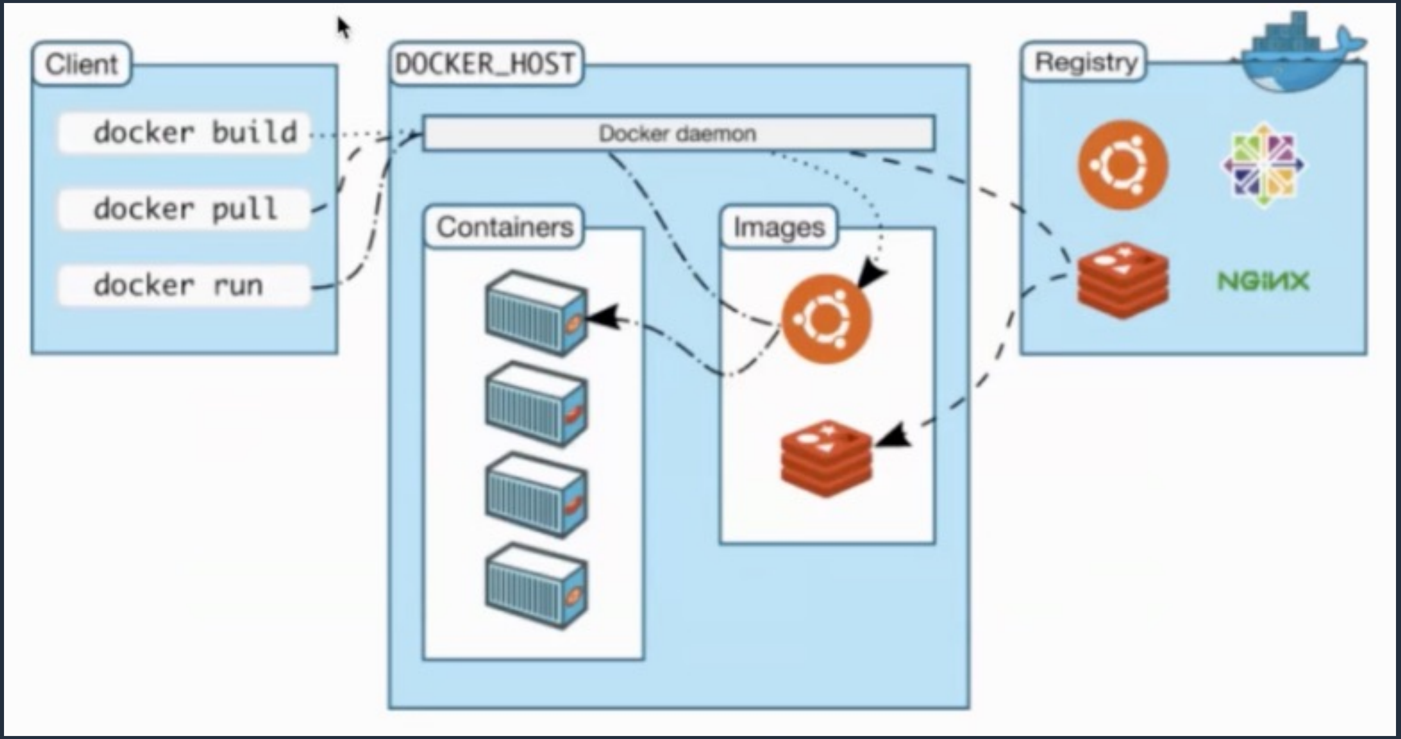
Why containers?


The old way: Applications on host
App
App
App
App
Libraries
Kernel
Heavyweight, non-portable
Relies on OS package manager
The new way: Deploy containers
App
Libraries
App
Libraries
App
Libraries
App
Libraries
Kernel
Small and fast, portable
Uses OS-level virtualization

Container overview

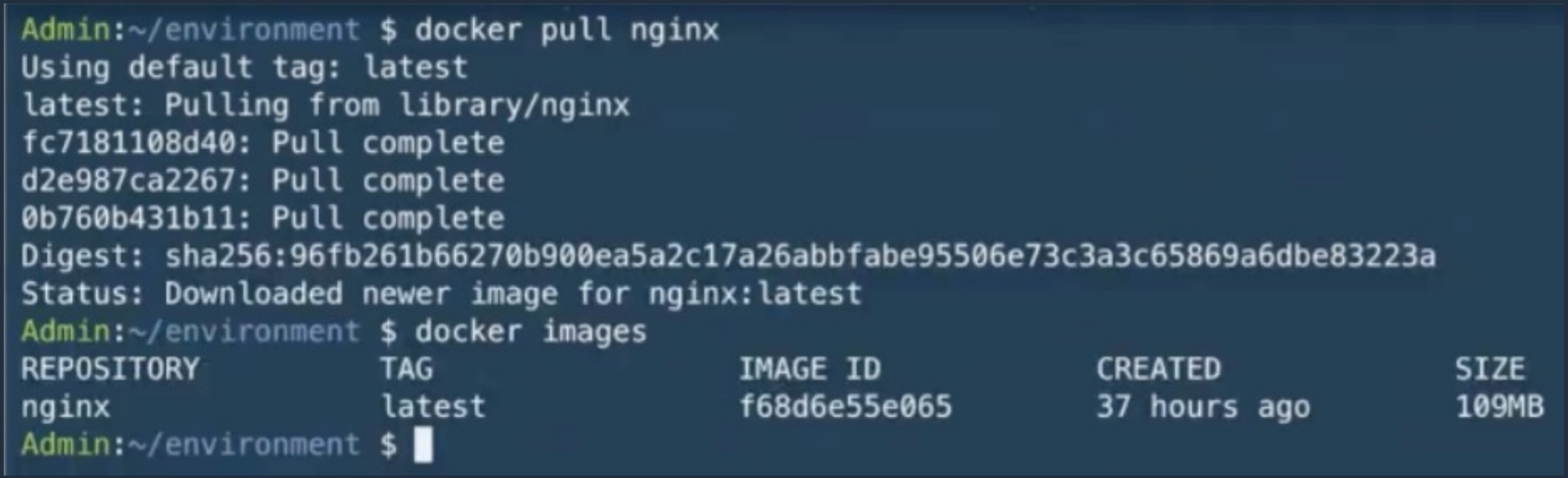


Container lifecycle



Docker commands - pull

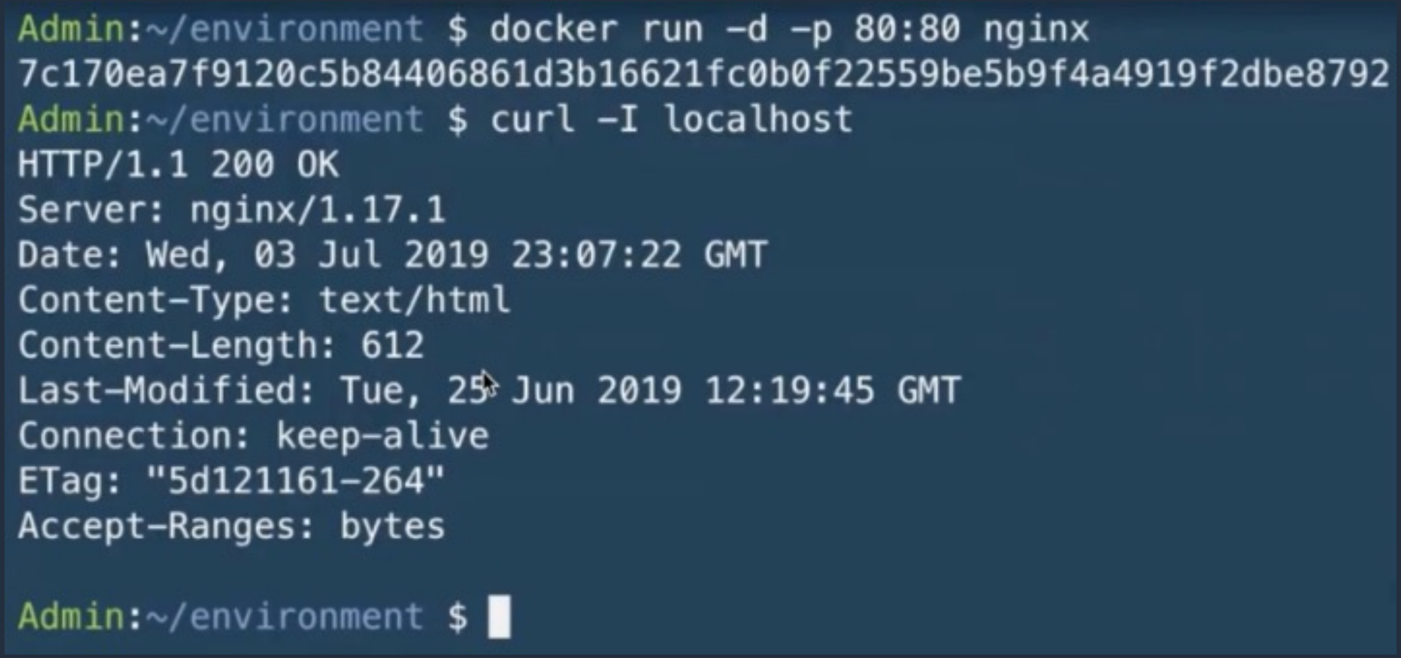
- Pull an image from an image registry



```
Admin:~/environment $ docker pull nginx
Using default tag: latest
latest: Pulling from library/nginx
fc7181108d40: Pull complete
d2e987ca2267: Pull complete
0b760b431b11: Pull complete
Digest: sha256:96fb261b66270b900ea5a2c17a26abbfabe95506e73c3a3c65869a6dbe83223a
Status: Downloaded newer image for nginx:latest
Admin:~/environment $ docker images
REPOSITORY          TAG                 IMAGE ID            CREATED             SIZE
nginx               latest              f68d6e55e065        37 hours ago        109MB
Admin:~/environment $ 
```

Docker commands - run

- Running a container from an image



```
Admin:~/environment $ docker run -d -p 80:80 nginx
7c170ea7f9120c5b84406861d3b16621fc0b0f22559be5b9f4a4919f2dbe8792
Admin:~/environment $ curl -I localhost
HTTP/1.1 200 OK
Server: nginx/1.17.1
Date: Wed, 03 Jul 2019 23:07:22 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

Admin:~/environment $ 
```

- The `-p` flag maps a port from the host to the container
- The `-d` flag runs the container in the background - detached

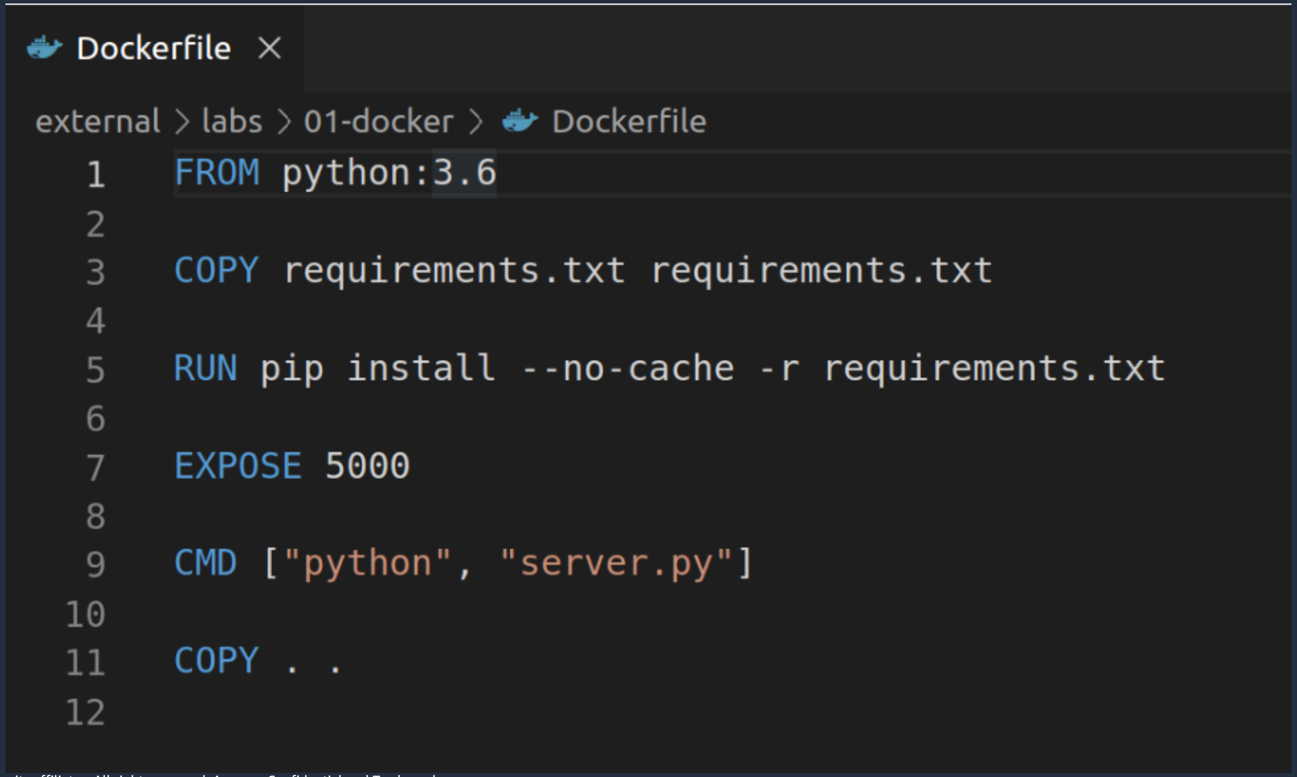
Let’s docker!

Lab 1: Building and Running a Docker Image

https://github.com/aws-ReBOOT/eks/tree/main/labs/01-docker

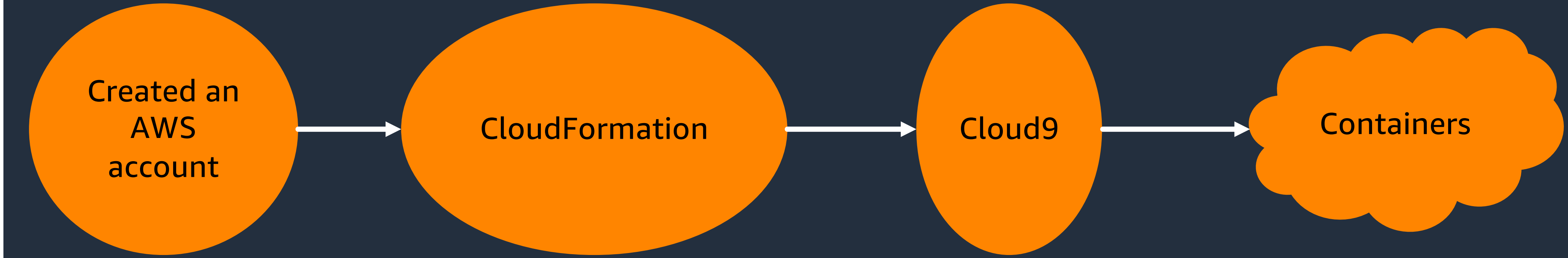
Dockerfile

Dockerfile contains a list of instructions to build a container image
labs/01-docker/Dockerfile



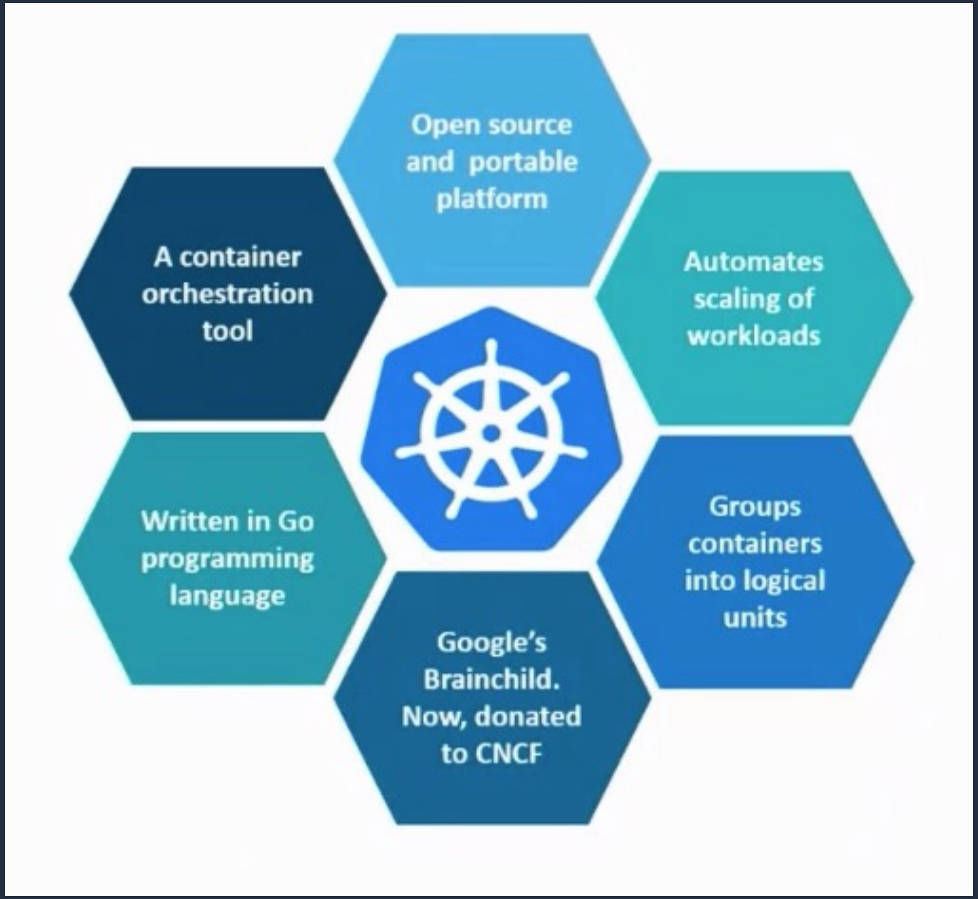
```
FROM python:3.6

COPY requirements.txt requirements.txt

RUN pip install --no-cache -r requirements.txt

EXPOSE 5000

CMD ["python", "server.py"]

COPY . .

```

Summary



What is Kubernetes?

What is Kubernetes?



A container orchestration tool

Open source and portable platform

Automates scaling of workloads

Written in Go programming language

Groups containers into logical units

Google's Brainchild. Now, donated to CNCF

Why Kubernetes?

- Containers
  - Good to bundle the application
  - Paved the way to build cloud native applications

- Self healing system
- Manages thousands of containers that run apps and ensure no downtime

Key capabilities!

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

EKS

What is EKS?

- EKS stands for Elastic Kubernetes Service
- Managed version of Kubernetes offered by AWS
- Launches and maintains the Control Plane for you with high available components
- Offers integration with other AWS services such as VPC and IAM. These integrations are open source projects built with the community
- Takes care of upgrades and patching
- Based on vanilla Kubernetes

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 --node-type t3.medium --name eks
```

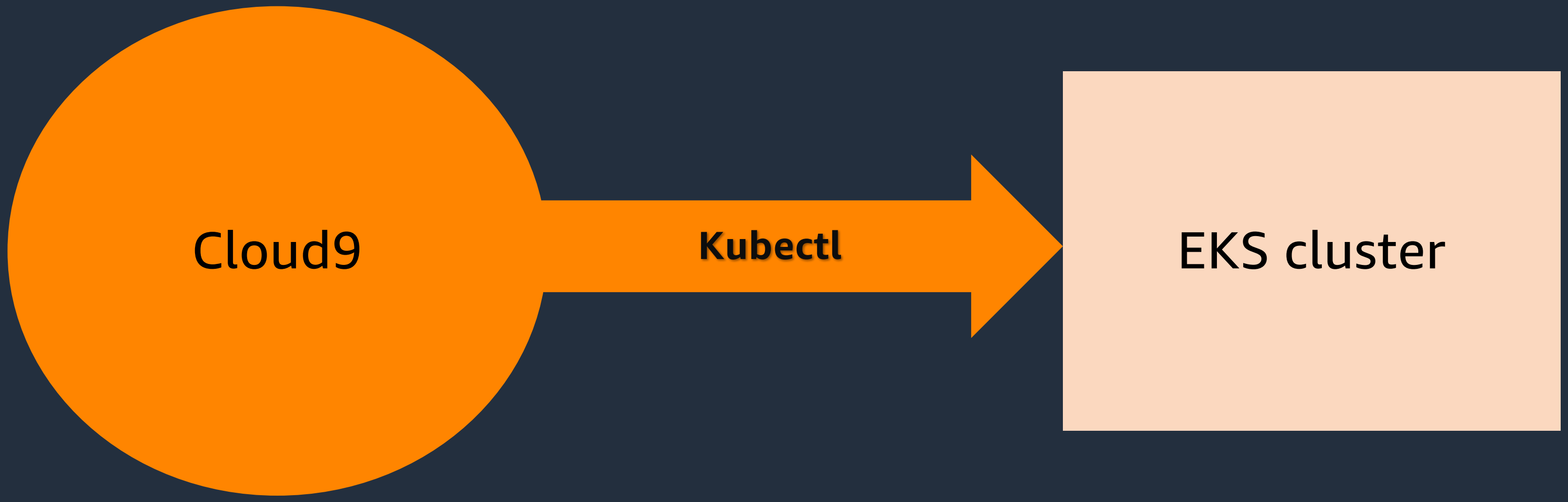
The EKS cluster will take approx. 15 minutes to create

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.

Kubectl



Let's check our new Cluster

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

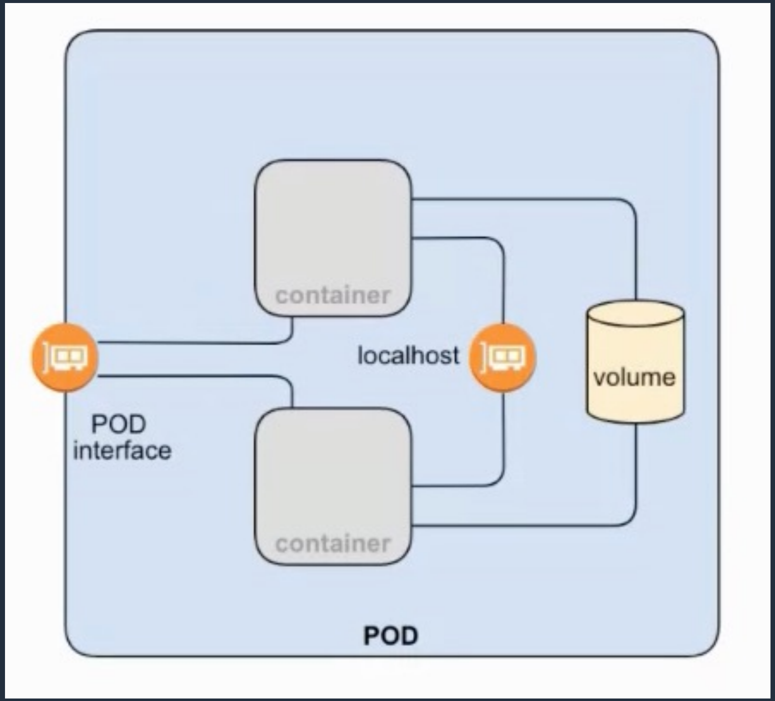
2- Listing your cluster with eksctl:

```
$ eksctl get clusters
```

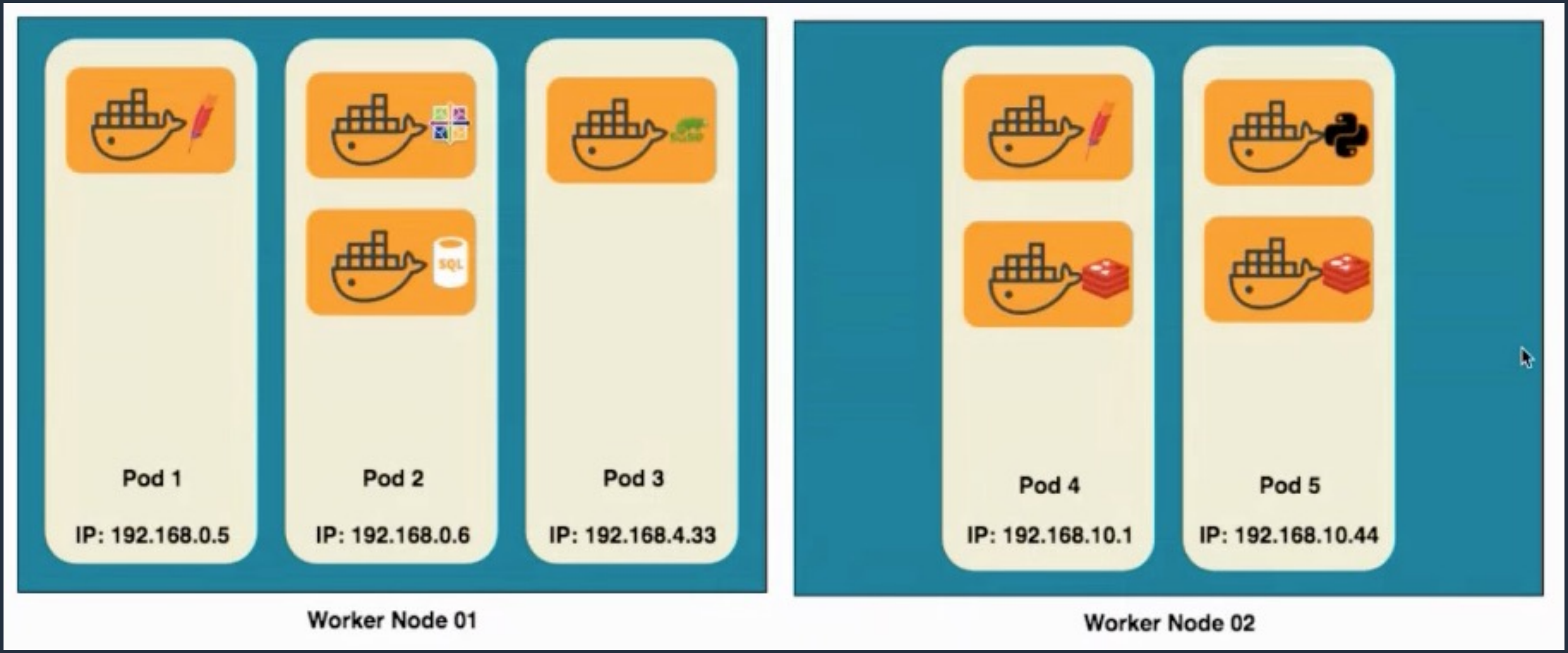
Pod

What is a pod?

- Smallest building block of Kubernetes
- Encapsulates the container and resources needed to run the application
- A unit of deployment



What is a Pod?


Pod 1
IP: 192.168.0.5
Pod 2
IP: 192.168.0.6
SQL
Pod 3
IP: 192.168.4.33
Pod 4
IP: 192.168.10.1
Pod 5
IP: 192.168.10.44
Worker Node 01
Worker Node 02

Creating pods in Kubernetes

Let’s kubectl!

Lab2: Introduction to Pods

https://github.com/aws-ReBOOT/eks/tree/main/labs/02-pods

- Define a Pod
- Creating a Pod
- Check the Pod: List and describe the Pod
- Clean Up

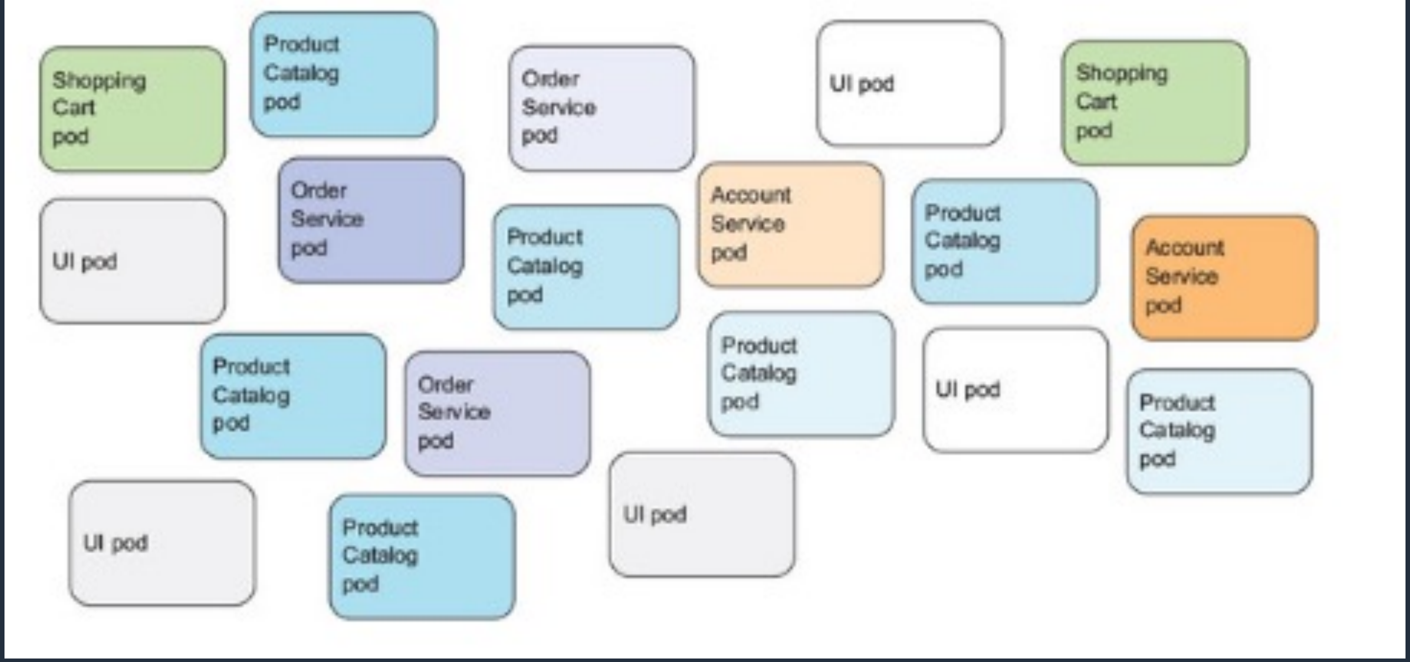
Environment recap

- CloudFormation service
- Cloud9 service
- EKS cluster
- Kubectl
- .
- .
- .
- Objects: Pod

Labels

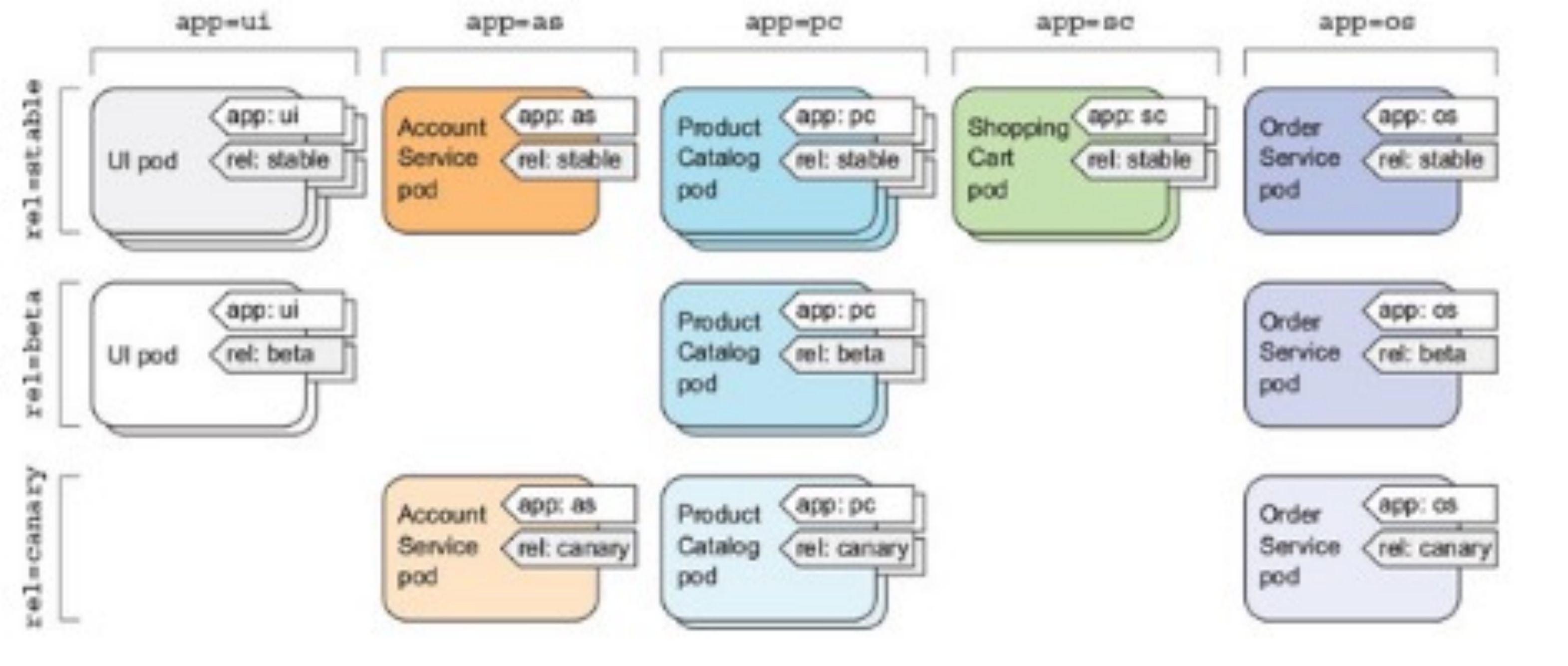
Labels

What if we are running a lot of pods?



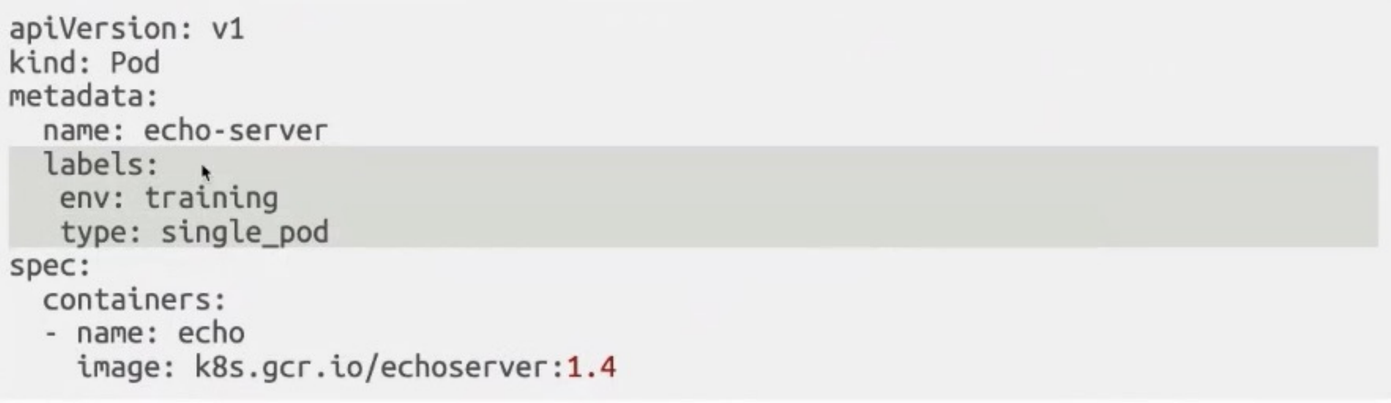
Labels

- Labels are key/value pair tags (like tags in AWS)
- Can be used to query and organize resources



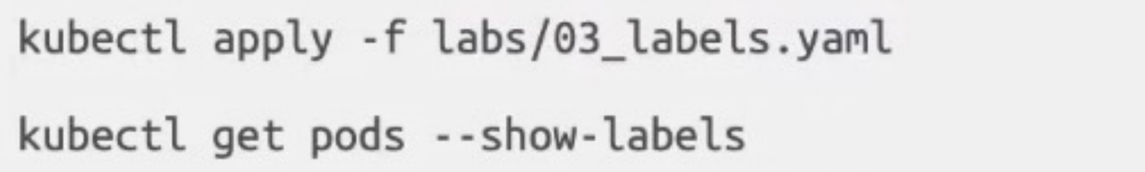
LAB: Labels

- 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
```

Let’s label!

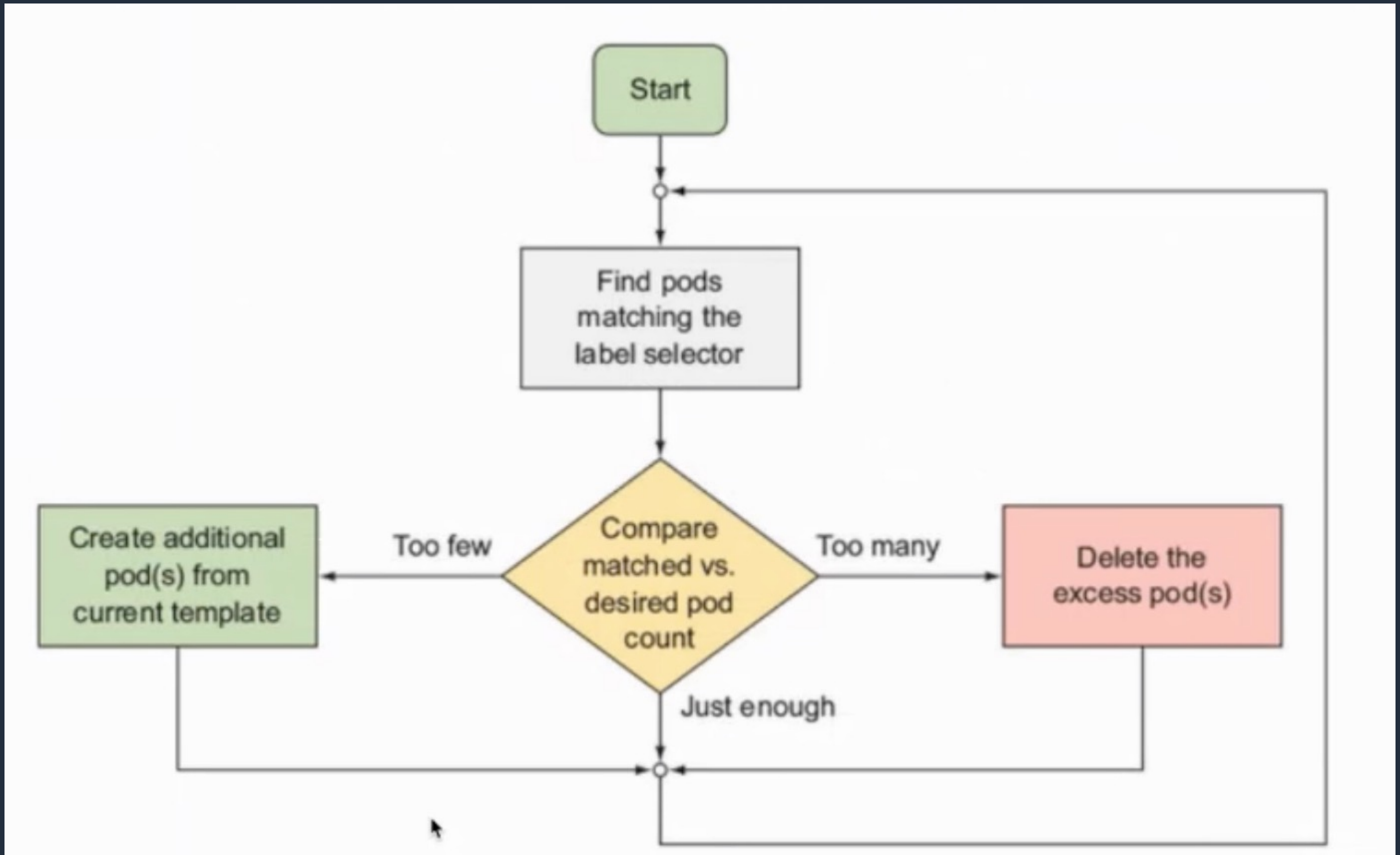
What happens if we delete a pod?

Controllers

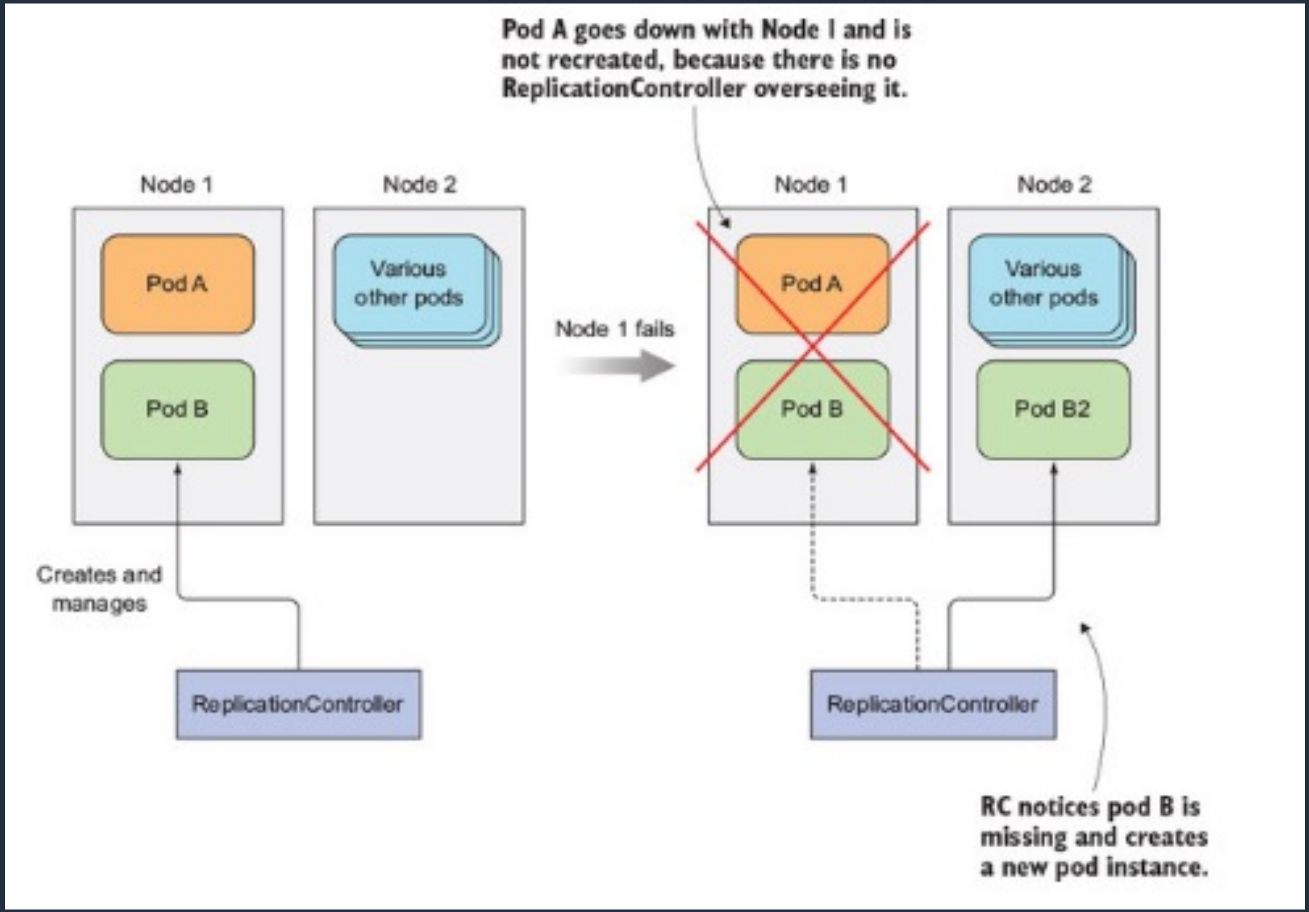
What controllers are and what they do?

- 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?


Start
Find pods matching the label selector
Compare matched vs. desired pod count
Too few
Create additional pod(s) from current template
Too many
Delete the excess pod(s)
Just enough

What controllers are and what they do?


Pod A goes down with Node I and is not recreated, because there is no ReplicationController overseeing it.
Node 1
Node 2
Pod A
Various other pods
Pod B
Node 1 fails
Node 1
Node 2
Pod A
Various other pods
Pod B
Pod B2
Creates and manages
ReplicationController
ReplicationController
RC notices pod B is missing and creates a new pod instance.

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

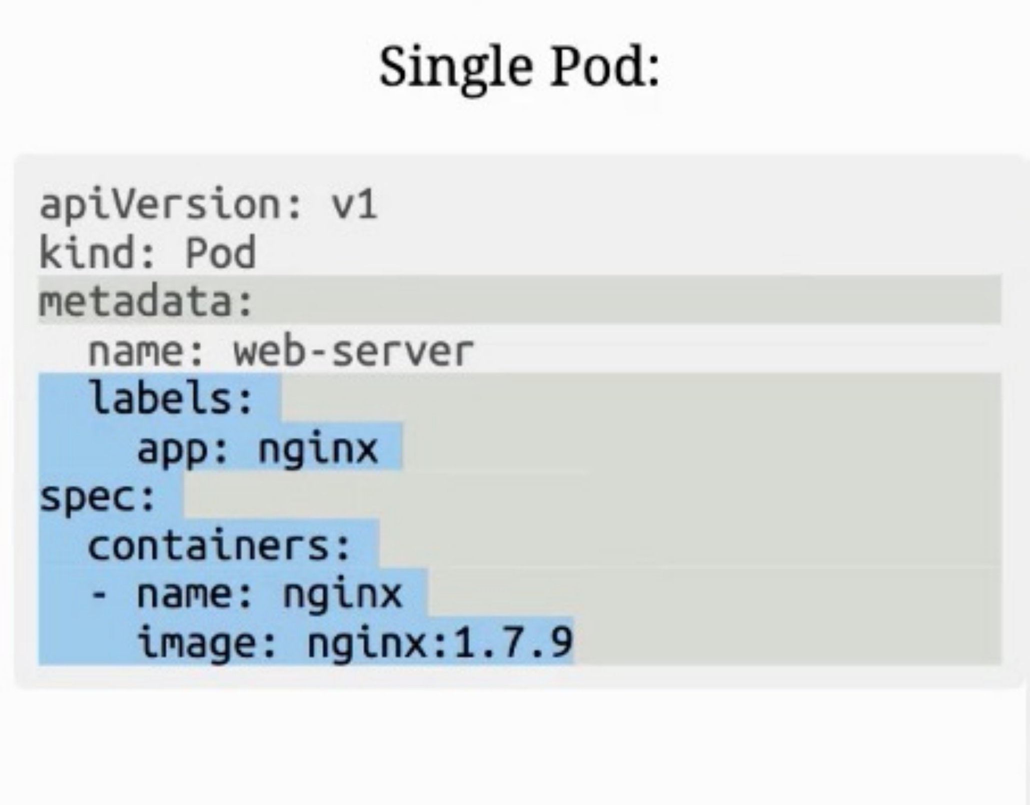
Deployments

Deployment

What is it?

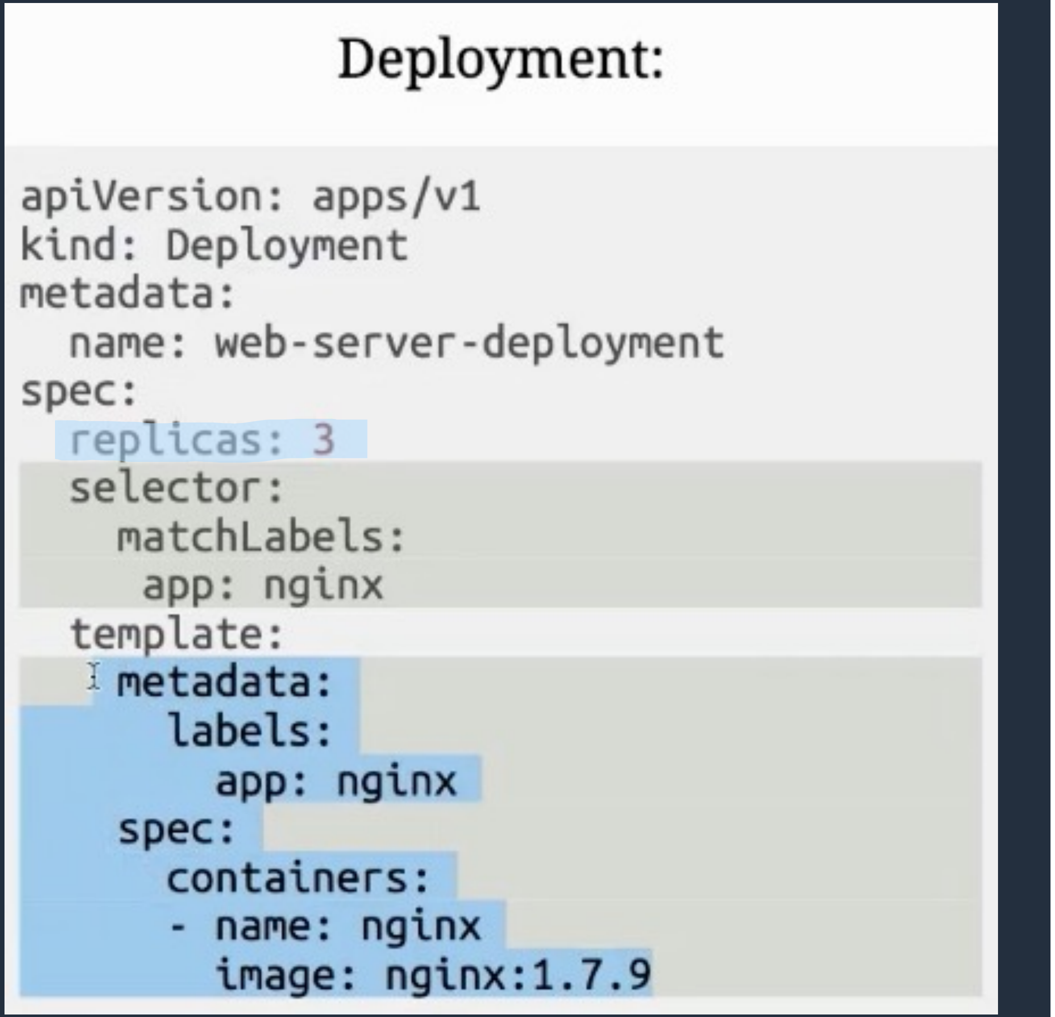
- A Deployment controller provides declarative updates at controlled rate for Pods
- An easy way to deploy updates for existing applications
- Allows you to pause/resume/rollback deployments

Deployments:



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
```

Let’s kubectl!

Lab: 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
- Check the Nginx server version
  - Kubectl port-forward web-server-deployment-XXXXXX-YYYYY 8080:80

Deployments:

Lab: Update the Deployment

- Update the deployment
  - Kubectl set image deployment web-server-deployment nginx=httpd

- Checking the results
  - Kubectl get pod –l app=nginx –L app

- Check the Nginx server version
  - Kubectl port-forward web-server-deployment-XXXXXX-YYYYY 8080:80

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

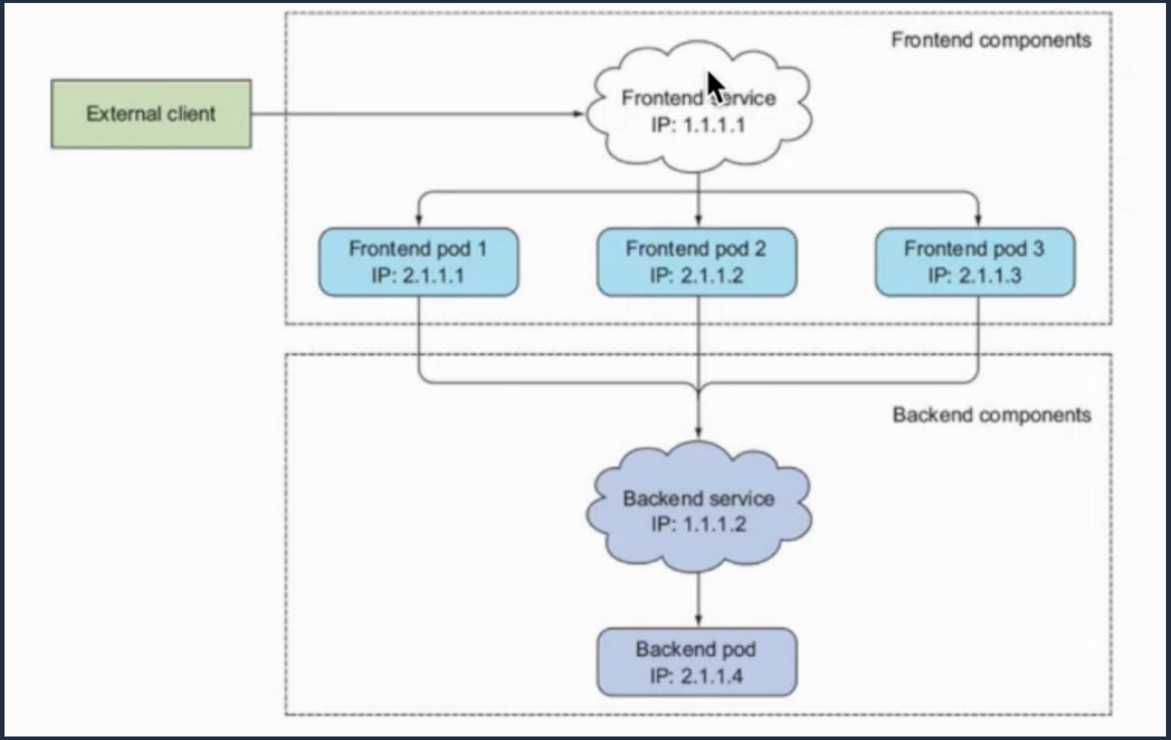
Services

But why?

- Pods are ephemeral
- Pods’IP are dynamic
- A single application might contain several Pods

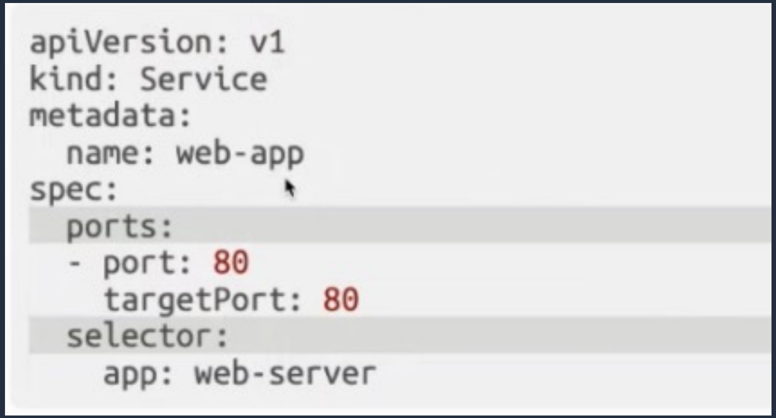
Services

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



Services

Lab: 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

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 sc web-app

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

Recap

- Containers
- Kubernetes
- Cloud9
- EKS
- Kubectl
- Pods
- Deployments
- Services

Thank you!

Maha El Ouni

Roxana Ciochina

Cloud Support Engineers @ AWS