

# How This Course is Structured

Kubernetes Basics

Fargate

EKS Basics

Deploying EKS with  
DevOps

Logging And Monitoring

Real World EKS Projects

EKS Advanced Concepts

Securing EKS

# What is Docker

## Container?

# In The Beginning

Environment: Dev

Everything working great! I am genius!



Code

Environment: Test



Code



Runtime Engine: Python 3.8

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration



Runtime Engine: Python 3.6

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration

# In The Beginning

Environment: Dev

Need to  
change the  
code. I guess  
it's okay.



Code



Runtime Engine: Python 3.8

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration

Environment: Test



Code



Runtime Engine: Python 3.6

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration

Environment: Prod



Code



Runtime Engine: Python 2.7

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration

# In The Beginning

Environment: Dev

Why did I  
take this job



Code



Runtime Engine: Python 3.8

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration

Environment: Test



Code



Runtime Engine: Python 3.6

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration

Environment: Prod



Code



Runtime Engine: Python 2.7

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration

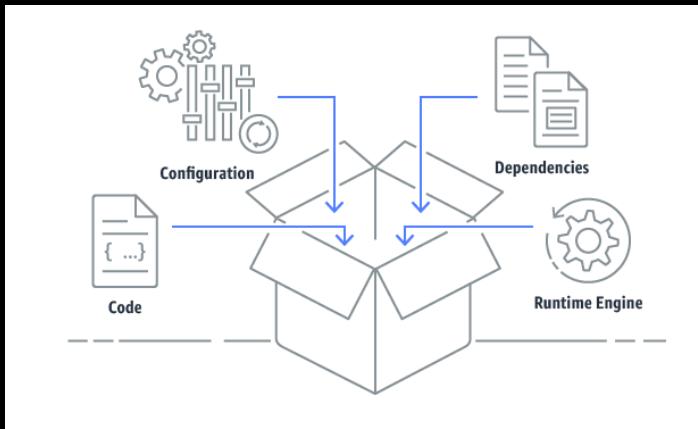
**WORKED FINE IN  
DEV**

**OPS PROBLEM NOW**

[memegenerator.net](http://memegenerator.net)

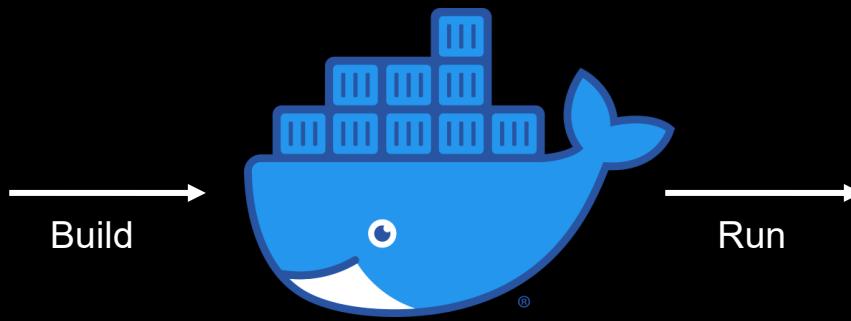
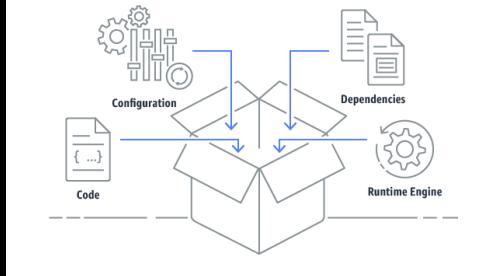


# Container

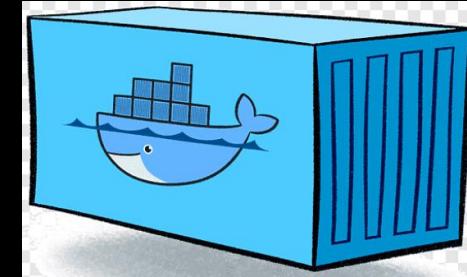


A container is an atomic, self contained package of software that includes everything it needs to run (code, runtime, libraries, packages, etc.)

# Docker Image Vs Container

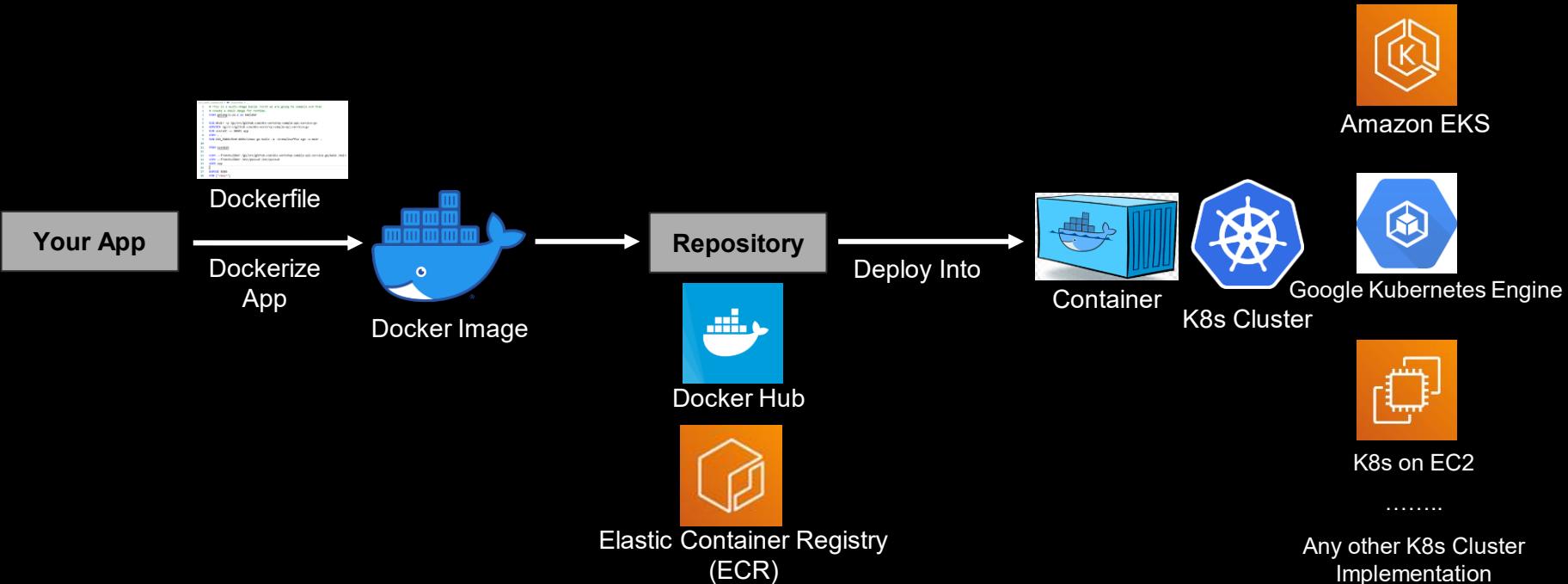


Docker Image  
Container Image

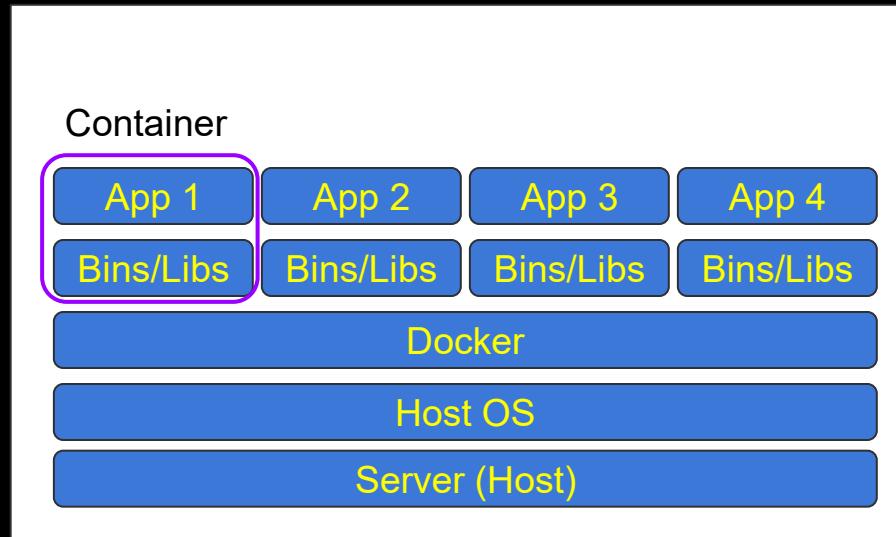
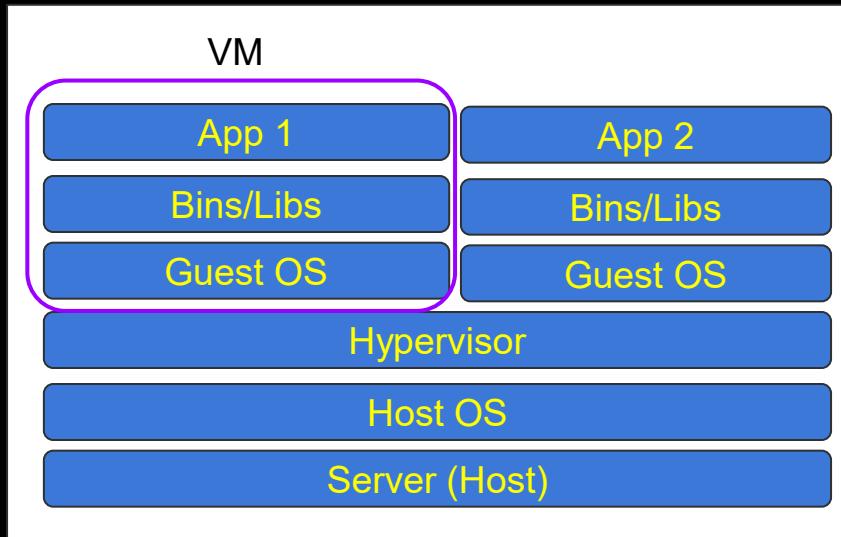


Container

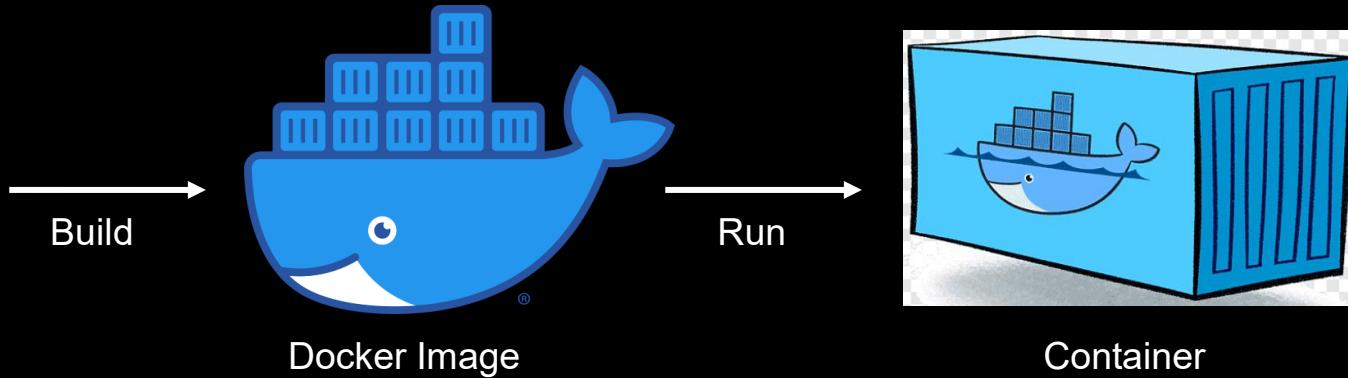
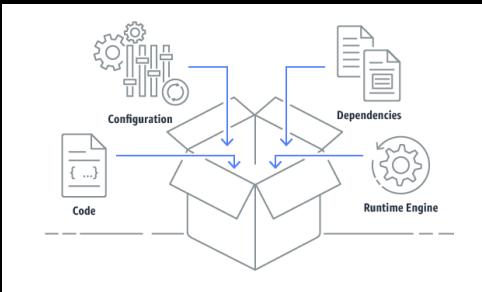
# The Big Picture



# Virtual Machine Vs Container



# Advantages



Runs reliably in any environment



# Going Back to Our Sad Developer

Environment: Dev

Environment: Test

Environment: Prod

Why did I  
take this job



Runtime Engine: Python 3.8

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration



Code



Runtime Engine: Python 3.6

```
import requests  
import kitchen-sink
```

Dependencies

DNS Service Name  
Database connection

Configuration



Code



Runtime Engine: Python 2.7

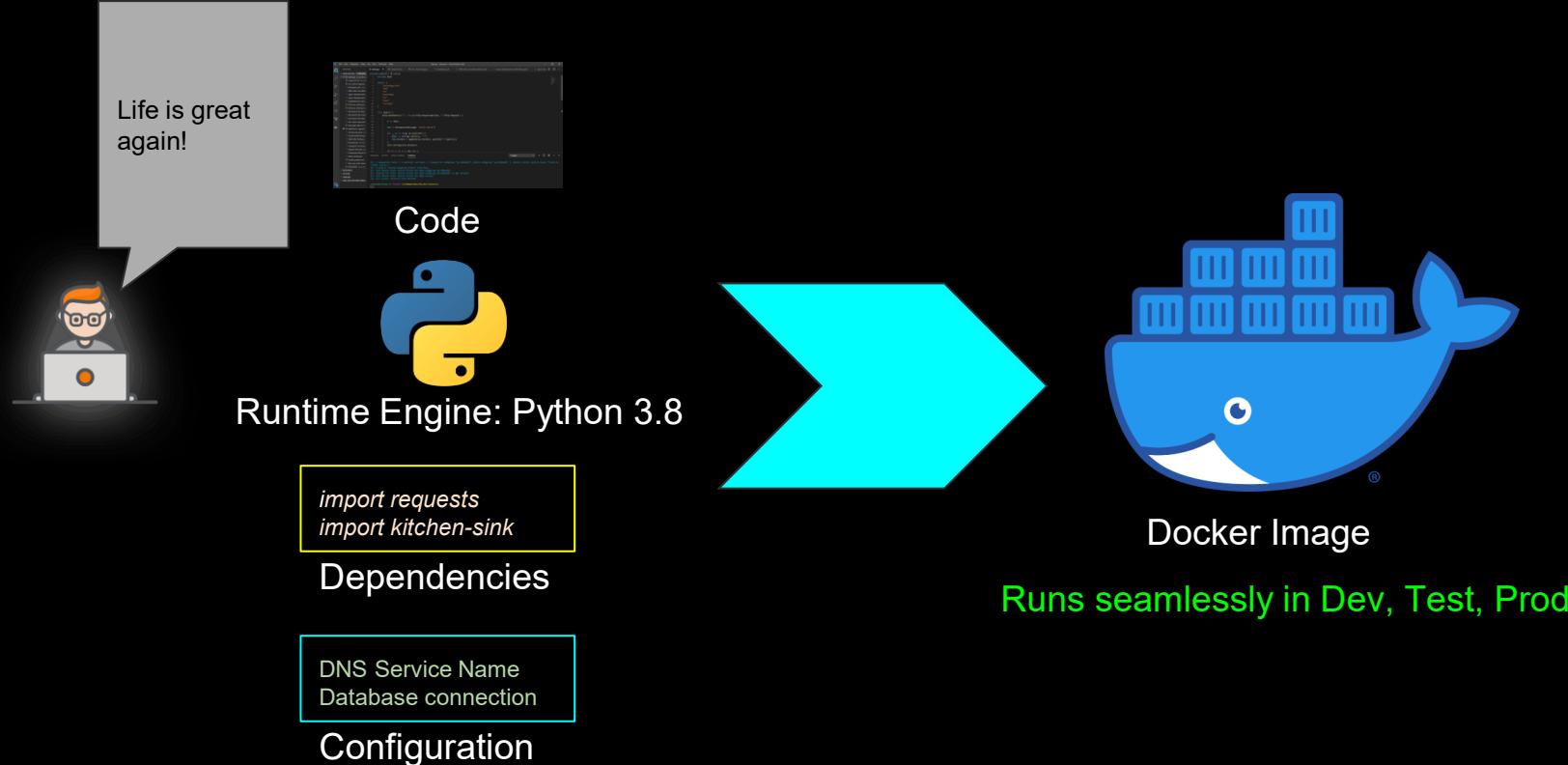
```
import requests  
import kitchen-sink
```

Dependencies

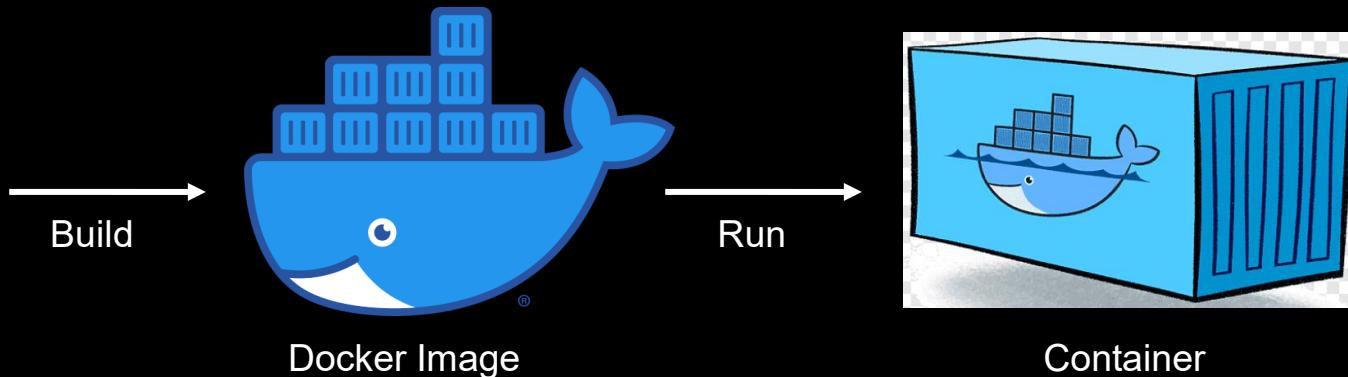
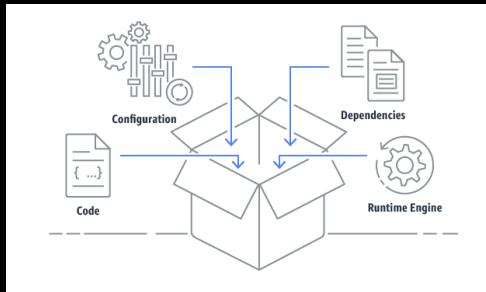
DNS Service Name  
Database connection

Configuration

# Going Back to Our Sad Developer



# Advantages



Runs reliably in any environment



Better resource utilization

App isolation

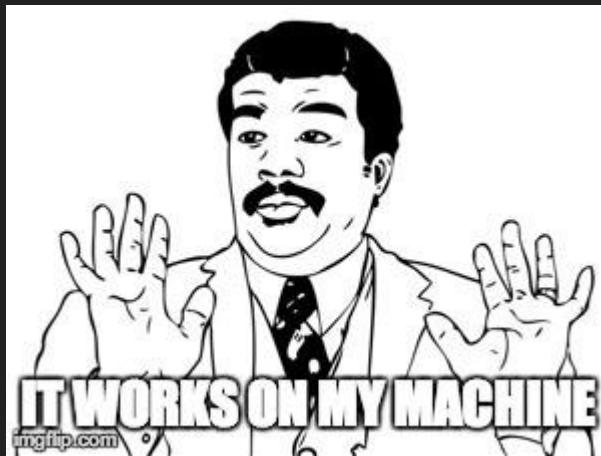
Speed

Container Orchestration is SOLVED!

What is Container  
Orchestrator??!!

# What is Docker/Container?

- Docker packages software into standardized units called containers that have everything your software needs to run including libraries, code and runtime
- Lets you quickly deploy and scale applications into any environment



# What is Container Orchestrator?



# How Does Docker Work?

Insert Video here

Draw a pentagon representations of 2 apps, color it different, then say it needs a host to run, so we spin up EC2s, move the pentagons inside EC2.

EC2s are like hyenas, if you see one, other ones are nearby.

To make it Highly Available, you need another in Az

Then comes scaling, put it in ASG, to route traffic you need Load Balancer.

If one task fails then u need to spin up

# Tasks Associated with Containers

- Deployment of Containers
- Redundancy and availability of Containers
- Scaling up or down of Containers
- Load Balancing
- Health Monitoring of Containers and Hosts
- Service Discovery
- And More...

# Container Orchestrator



# Say Hello to Container Orchestrators

- Docker Swarm



- Apache Mesos



- Cattle, Nomad, Empire

- AWS ECS (Elastic Container Service)



- Kubernetes



- EKS (Elastic Container Service for Kubernetes)



- AWS Fargate



# What is

# Kubernetes?

(The most popular Container Orchestrator)

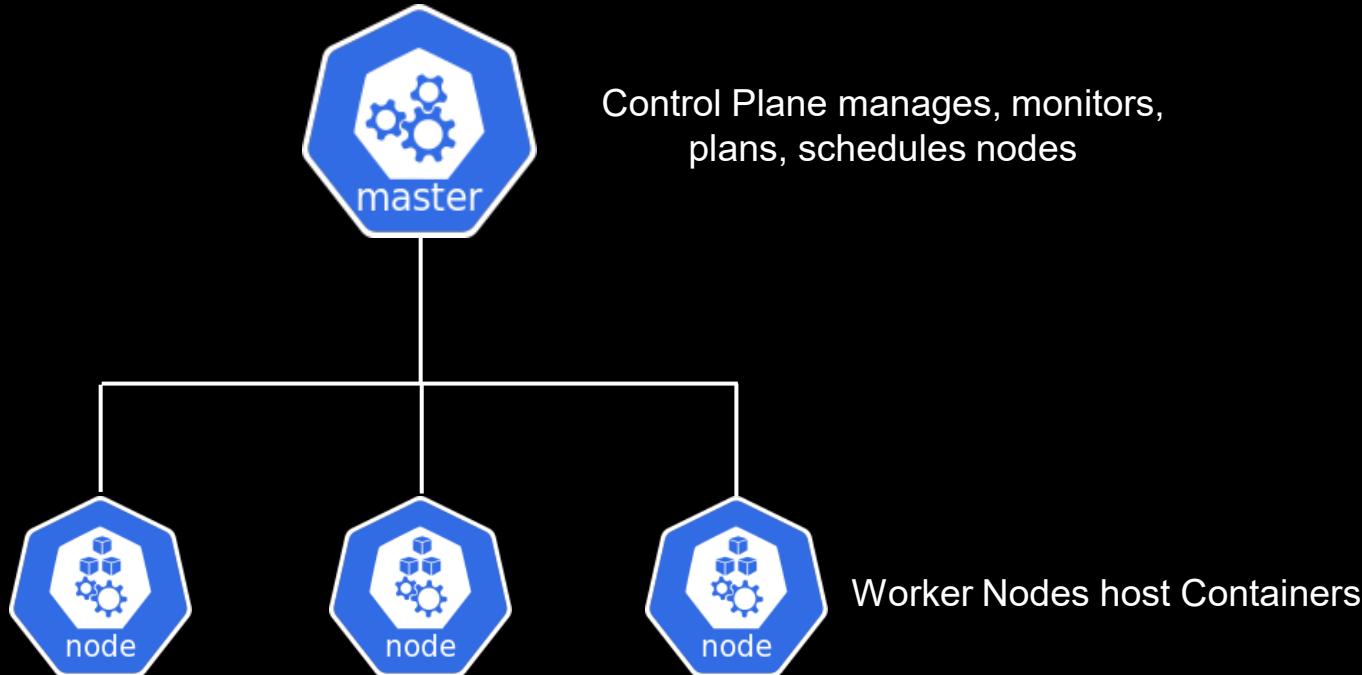
# Working Backwards



# Kubernetes Architecture



# Who Manages Nodes?



# Control Plane Components



Control Plane manages, monitors,  
plans, schedules nodes



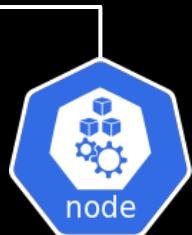
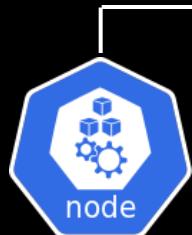
Key Value Store for critical cluster info



Ensures proper state of cluster components

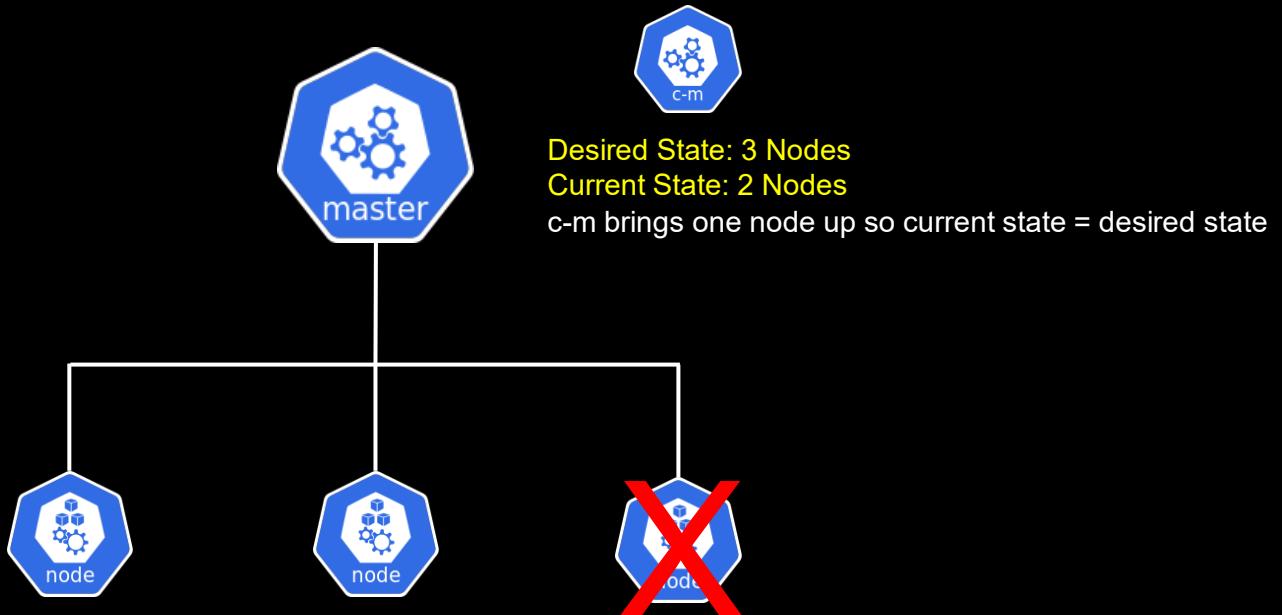


Puts containers to proper nodes

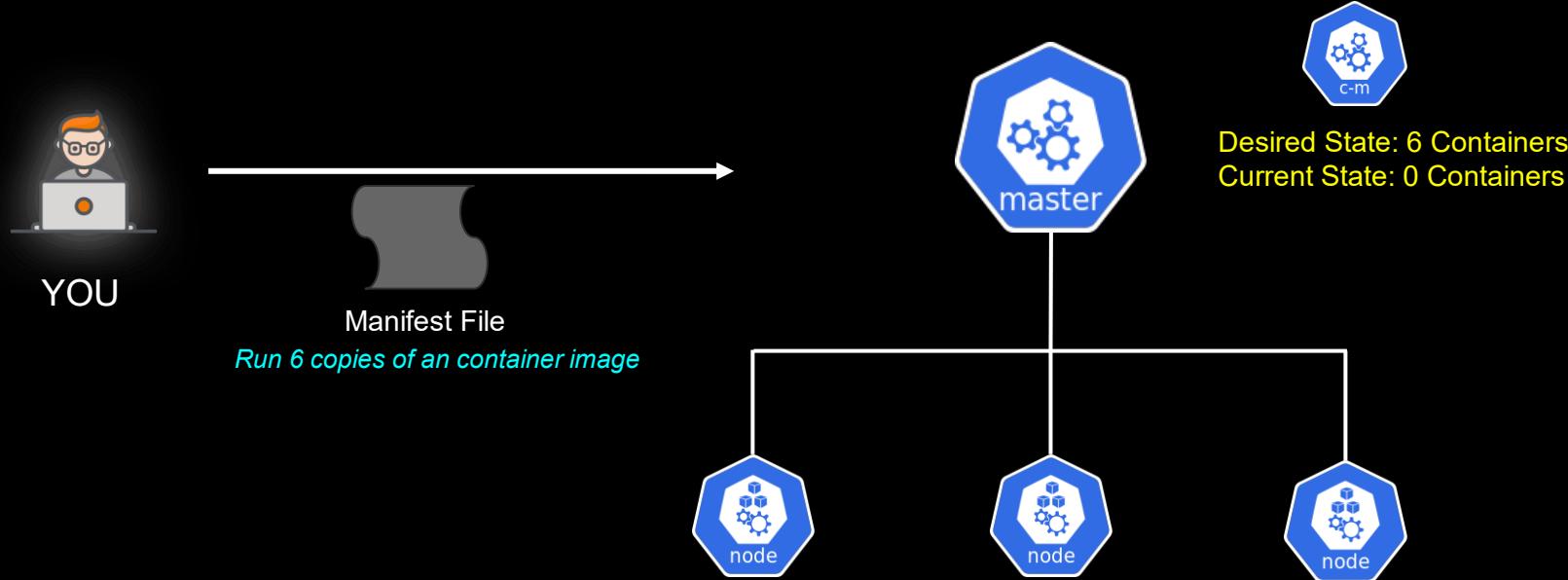


Worker Nodes host Containers

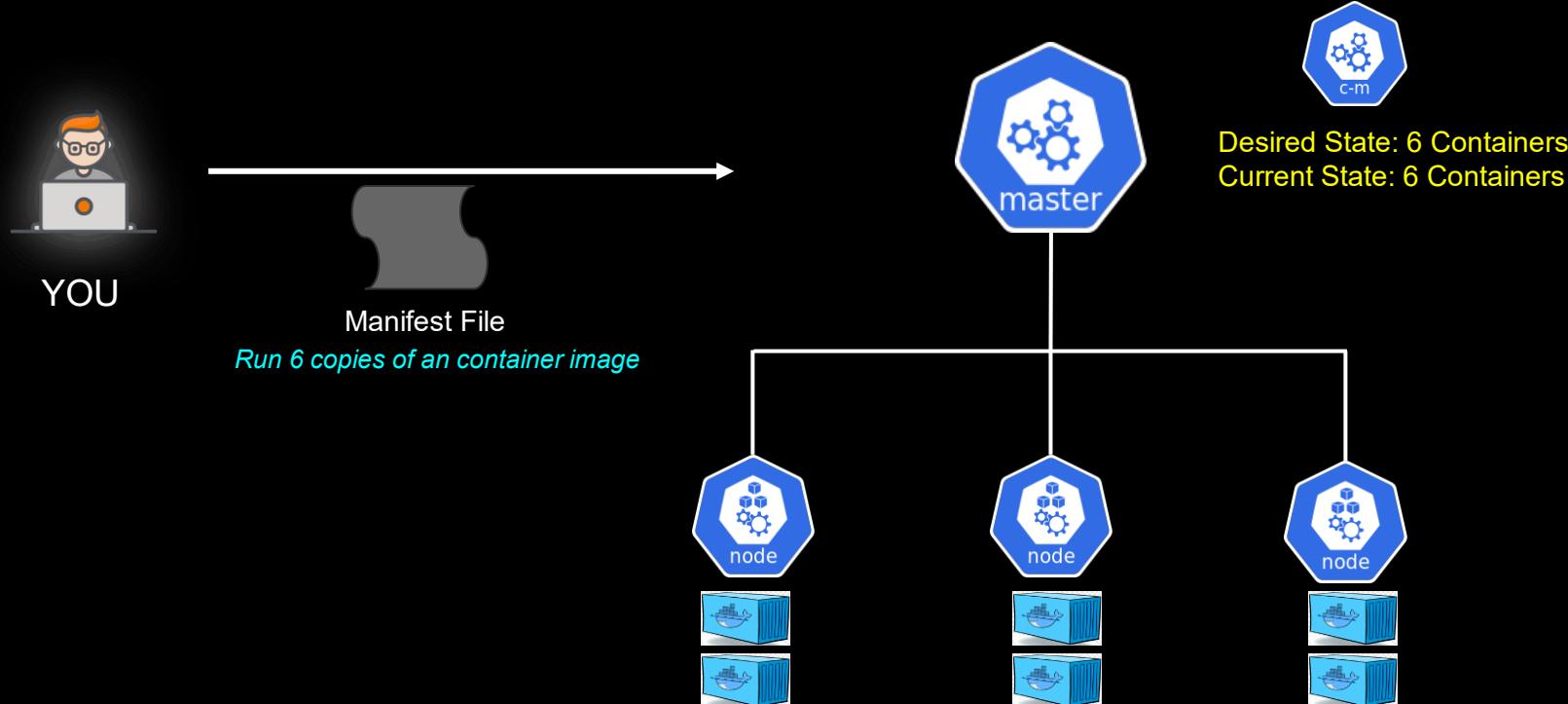
# Kubernetes Cluster State



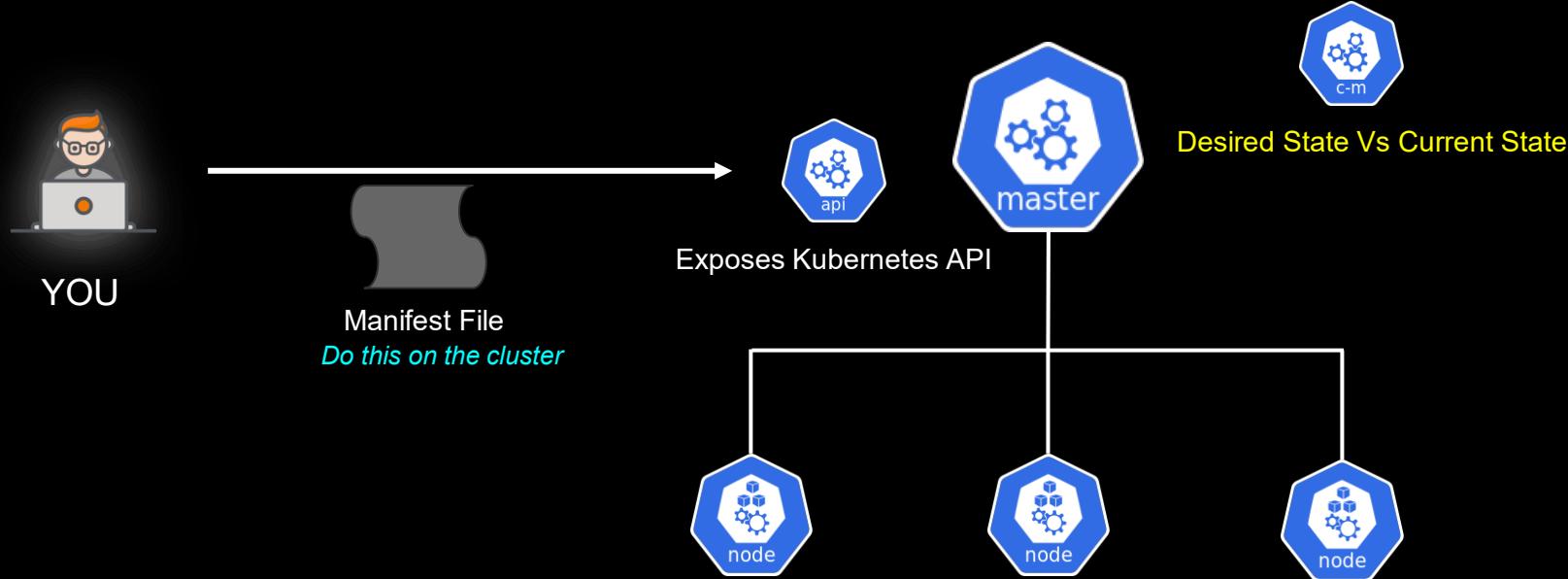
# Who Specifies State?



# Who Specifies State?



# Gateway to Control Plane



# Kubernetes Architecture



Control Plane manages, monitors,  
plans, schedules nodes



Key Value Store for critical cluster info



Ensures proper state of cluster components



Puts containers to proper nodes

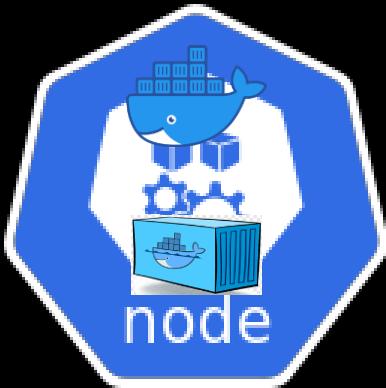


Exposes Kubernetes API



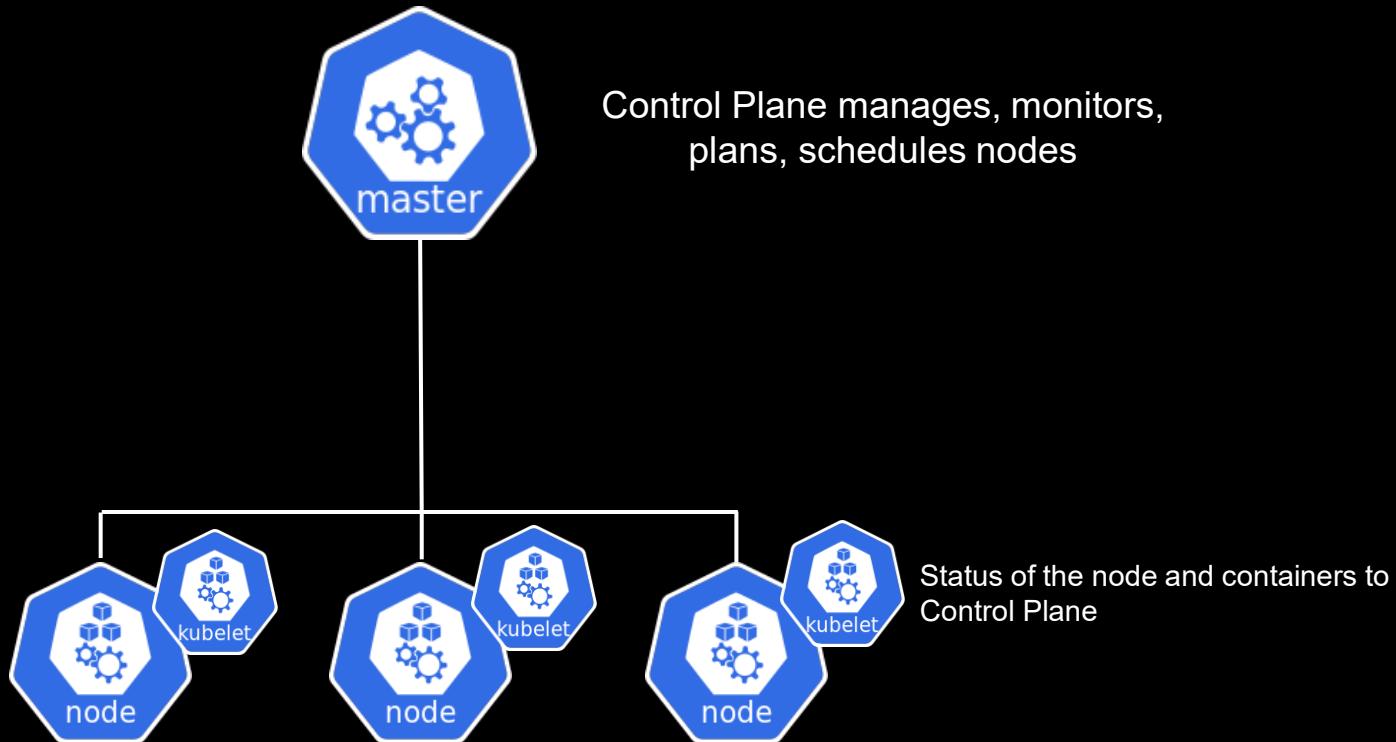
Worker Nodes host Containers

# What's In Node?

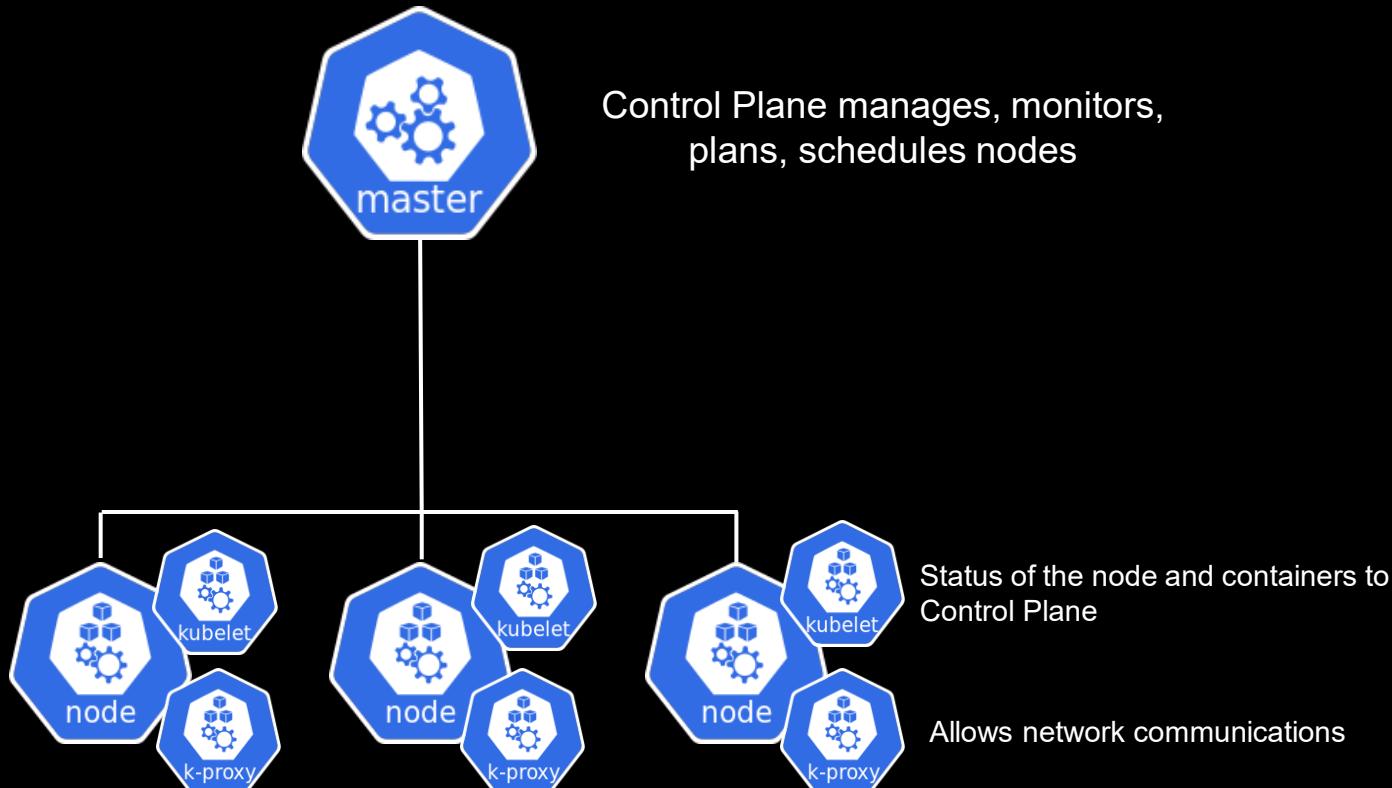


Container Runtime Engine  
Docker, Containerd, CRI-O, frakti

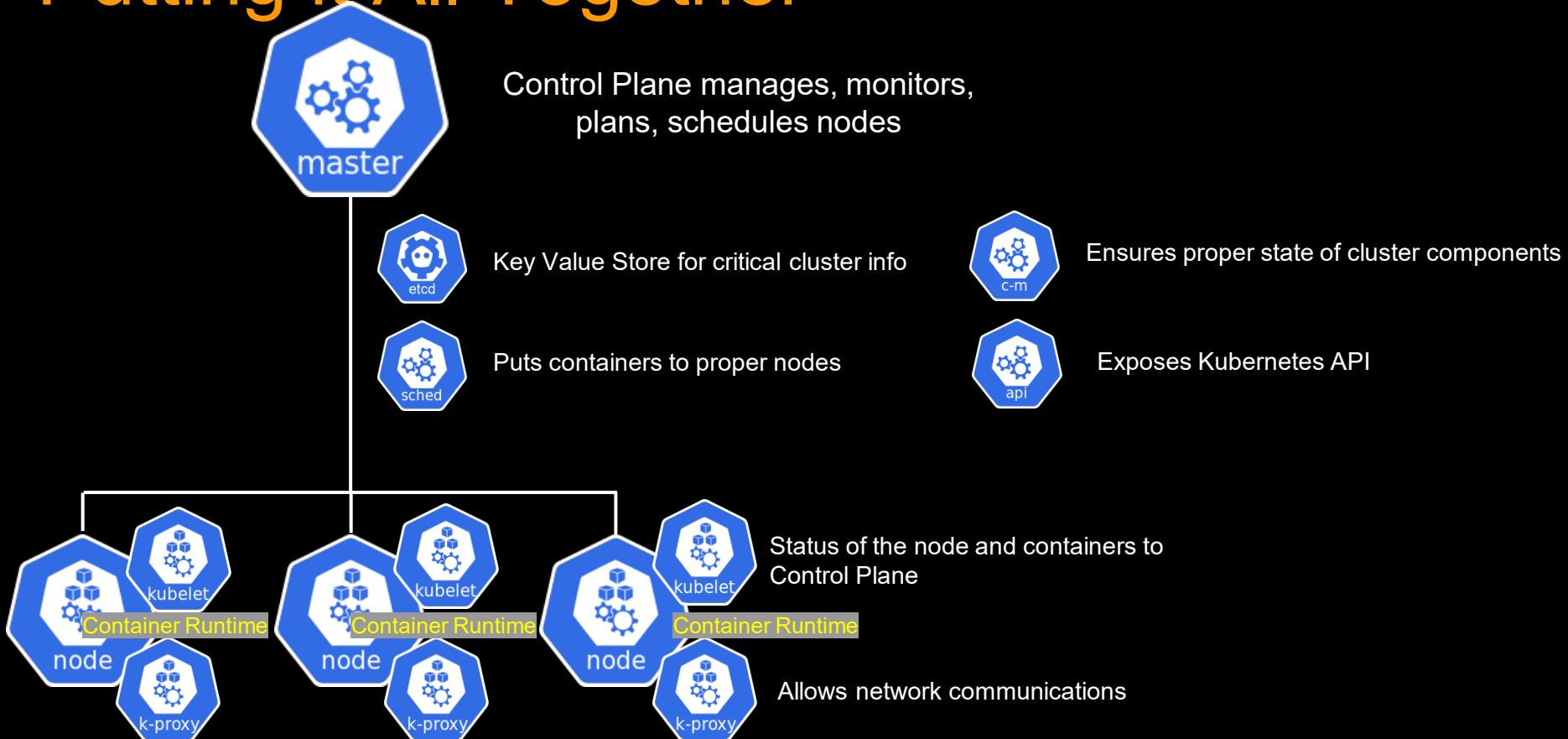
# Control Plane - Node Communication



# Container-Container Communication



# Putting it All Together



K8s?!

Kubernetes

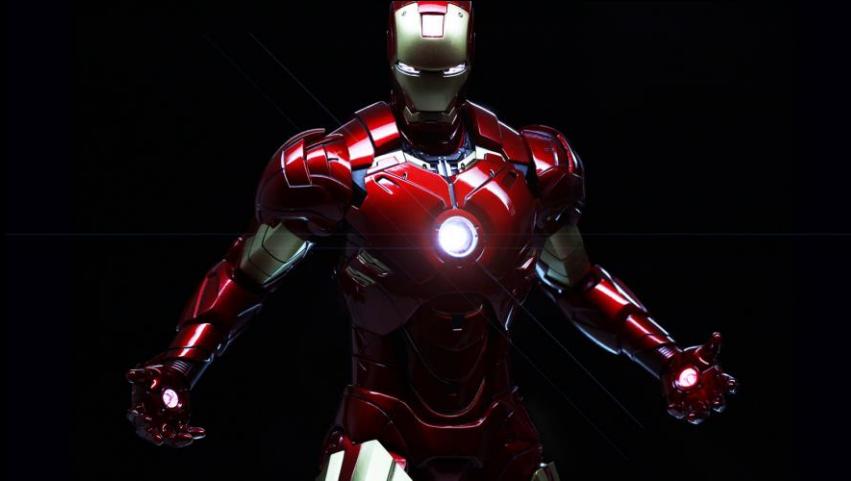
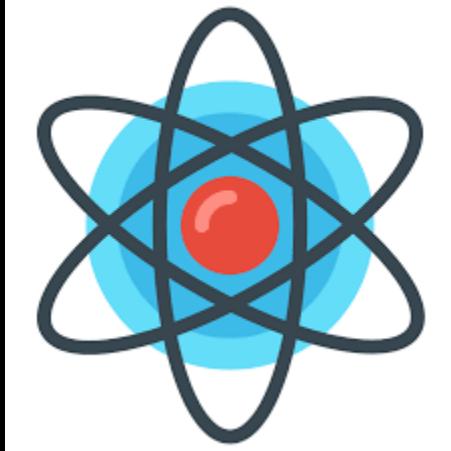
8 Letters

K8s

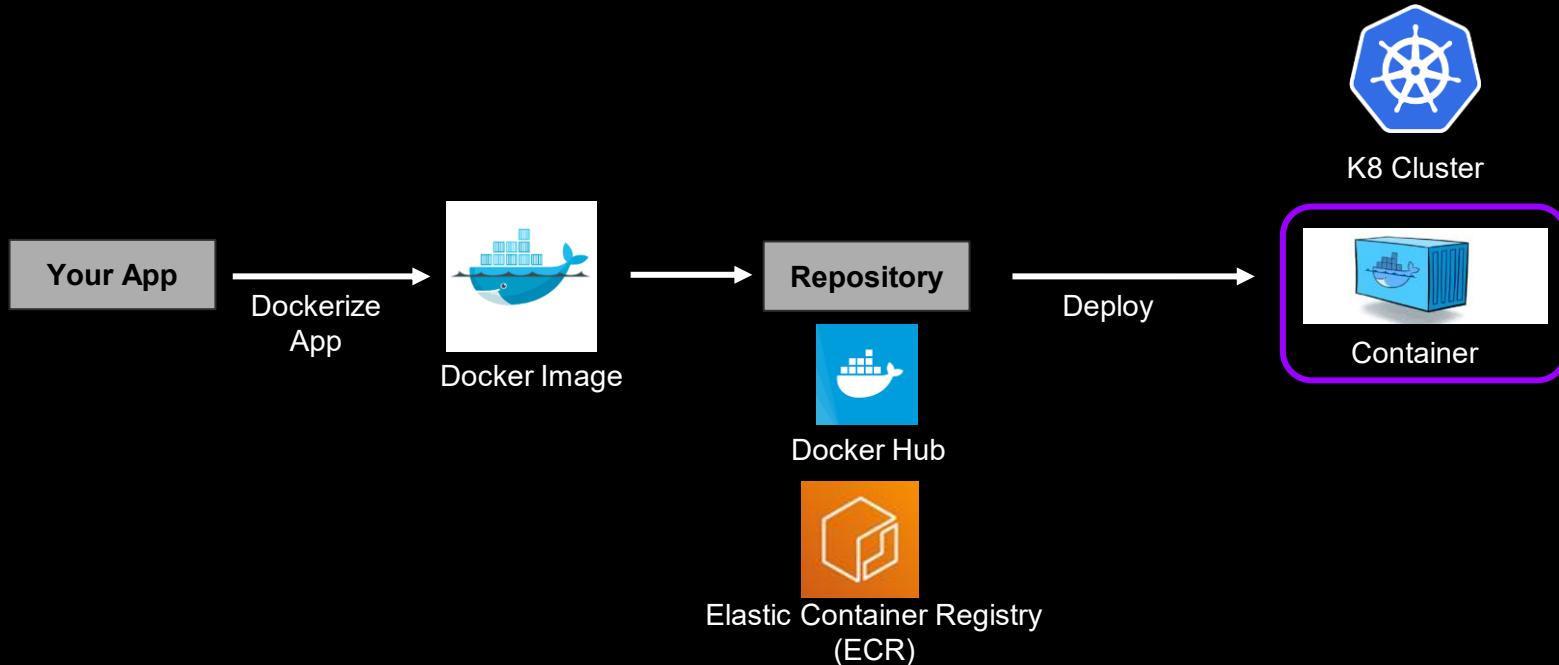
# Pods



# Pods



# The Big Picture



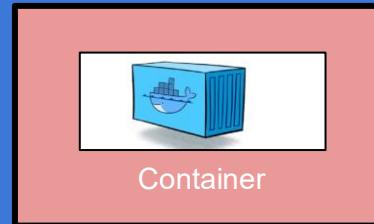
# NPC



K8 Cluster

NODE

POD

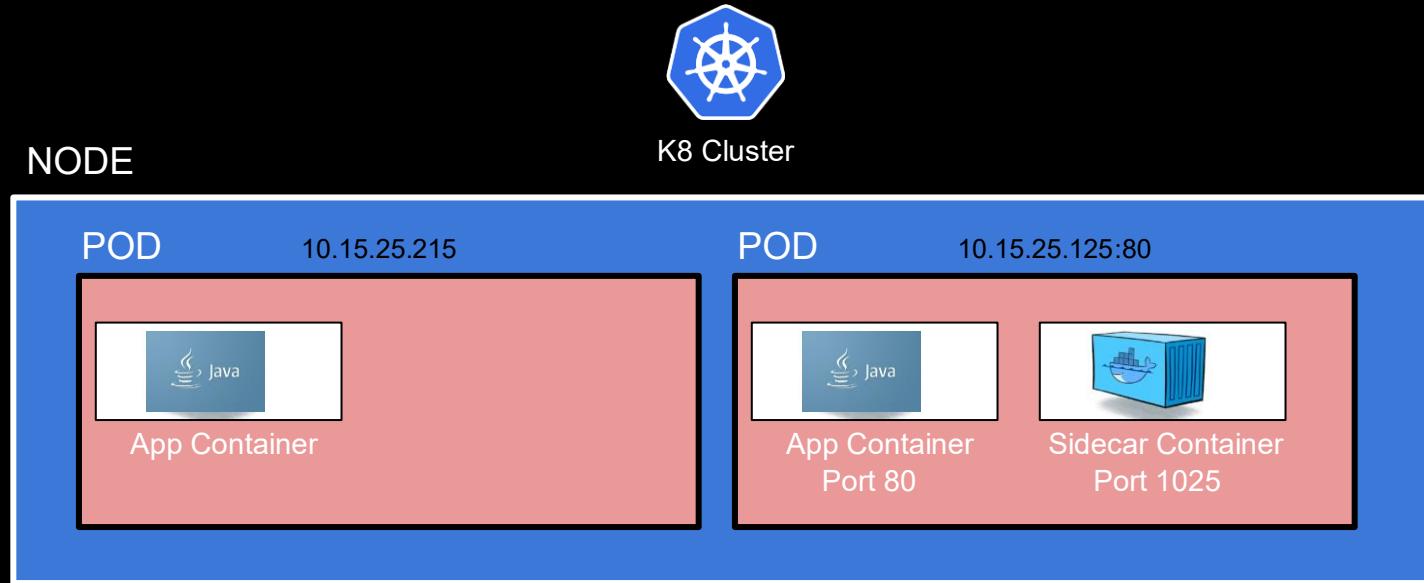


Container

- Smallest object you can create in Kubernetes
- Remember NPC (Node > Pod > Container)

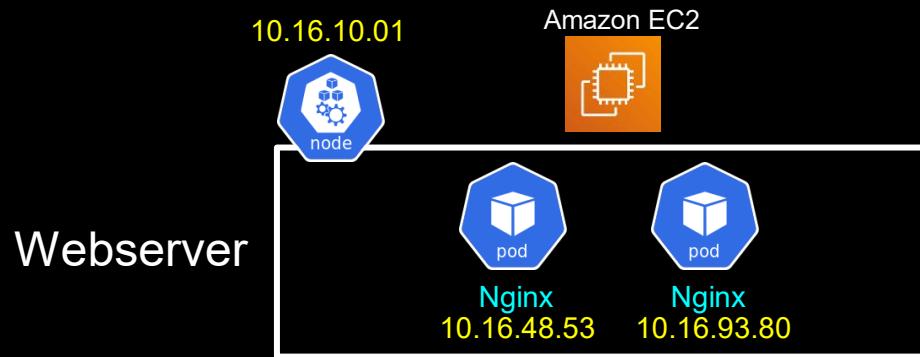


# Life of a Pod

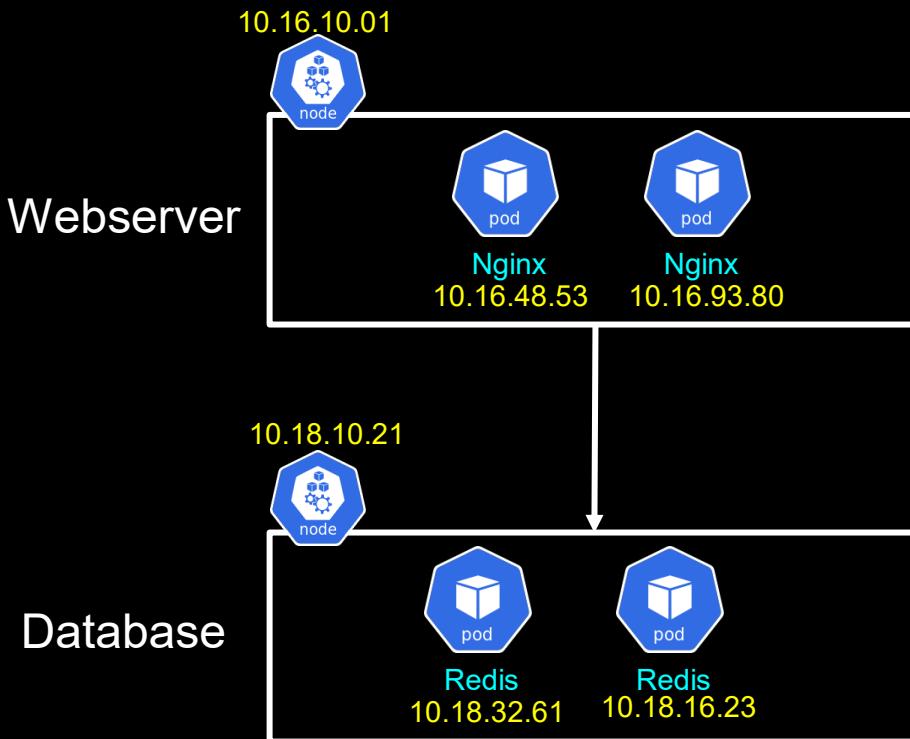


- One App Container per Pod
- Each POD has unique IP address

# Sample Pods



# Life Of A Simple Pod



# Deployment



# Deployment

+

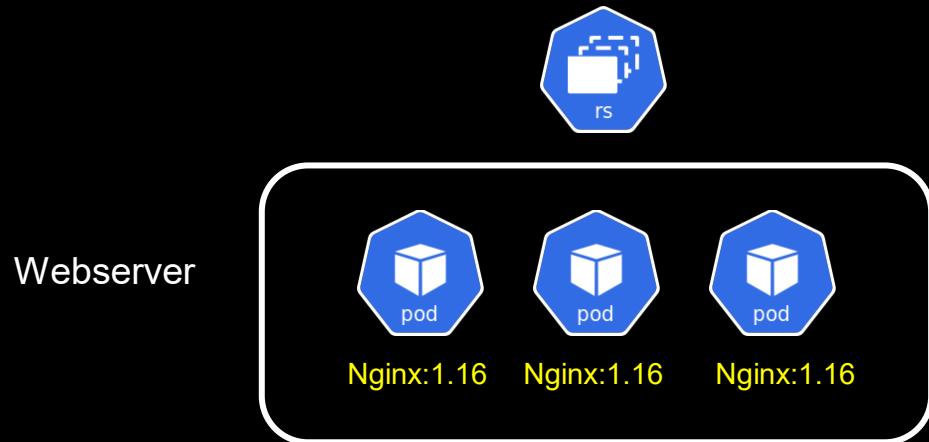
# Rolling Update

+

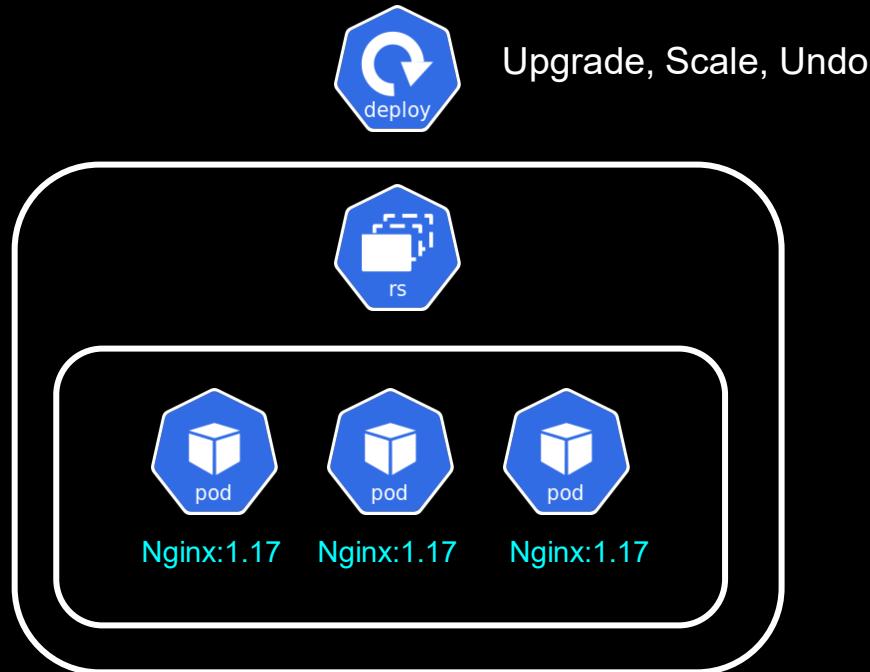
# Replicaset



# Pods In Real World



# Deployment: Update of Container



# Deployment: Manifest File

```
! nginx-deployment-withrolling.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: test
6  name: testdeploy
7
8  spec:
9    replicas: 3
10   selector:
11     matchLabels:
12       environment: test
13   minReadySeconds: 10
14   strategy:
15     rollingUpdate:
16       maxSurge: 1
17       maxUnavailable: 0
18       type: RollingUpdate
19   template:
20     metadata:
21       labels:
22         environment: test
23     spec:
24       containers:
25         - image: nginx:1.16
26           name: nginx
```



Manages pods with label `environment:test` and manages replicaset defined within



Manages pods with label `environment:test`



Nginx:1.16  
`environment:test`



Nginx:1.16  
`environment:test`



Nginx:1.16  
`environment:test`

# Kubernetes Deployment YAML + DEMO

! nginx-deployment-withrolling.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: test
6  name: testdeploy
7
8  spec:
9    replicas: 3
10   selector:
11     matchLabels:
12       environment: test
13   minReadySeconds: 10
14   strategy:
15     rollingUpdate:
16       maxSurge: 1
17       maxUnavailable: 0
18       type: RollingUpdate
19   template:
20     metadata:
21       labels:
22         environment: test
23     spec:
24       containers:
25         - image: nginx:1.16
26           name: nginx
```



Manages pods with label `environment:test` and manages replicaset defined within



Manages pods with label `environment:test`



Nginx:1.16  
`environment:test`



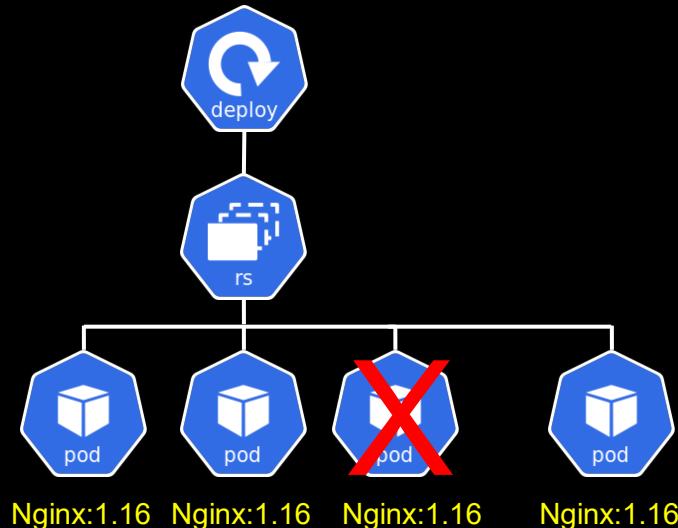
Nginx:1.16  
`environment:test`



Nginx:1.16  
`environment:test`

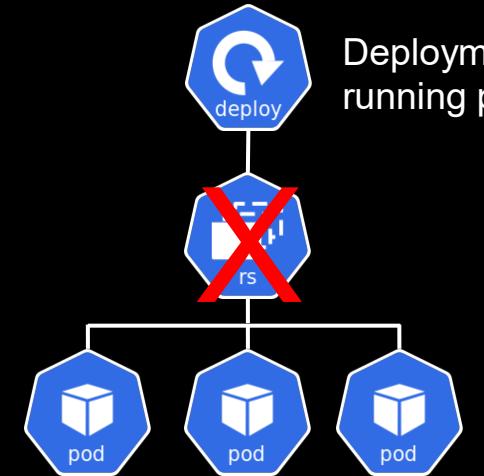
# Replicaset Restores Pods

```
! nginx-deployment-with-replicaset.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: test
6      name: testdeploy
7  spec:
8    replicas: 3
9    selector:
10   matchLabels:
11     environment: test
12   minReadySeconds: 10
13   strategy:
14     rollingUpdate:
15       maxSurge: 1
16       maxUnavailable: 0
17     type: RollingUpdate
18   template:
19     metadata:
20       labels:
21         environment: test
22     spec:
23       containers:
24         - image: nginx:1.17
25           name: nginx
```



# Deployment Restores Replicaset

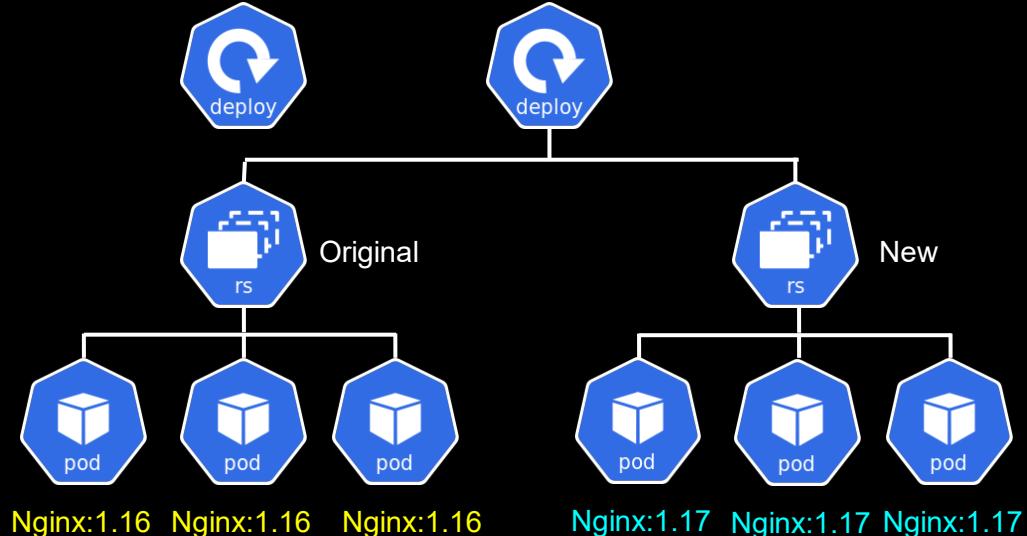
```
! nginx-deployment-with-rolling.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: test
6      name: testdeploy
7  spec:
8    replicas: 3
9    selector:
10   matchLabels:
11     environment: test
12   minReadySeconds: 10
13   strategy:
14     rollingUpdate:
15       maxSurge: 1
16       maxUnavailable: 0
17     type: RollingUpdate
18   template:
19     metadata:
20       labels:
21         environment: test
22     spec:
23       containers:
24         - image: nginx:1.17
25           name: nginx
```



Deployment will restore replicaset with running pods - super fast!

# Deployment Rolling Update

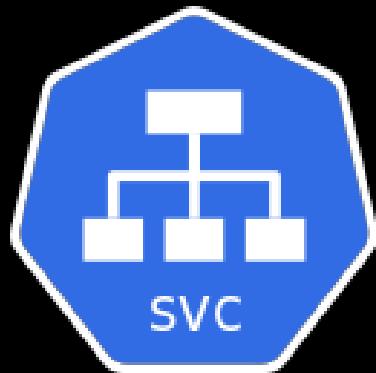
```
! nginx-deployment-with-rolling.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: test
6      name: testdeploy
7  spec:
8    replicas: 3
9    selector:
10   matchLabels:
11     environment: test
12   minReadySeconds: 10
13   strategy:
14     rollingUpdate:
15       maxSurge: 1
16       maxUnavailable: 0
17       type: RollingUpdate
18   template:
19     metadata:
20       labels:
21         environment: test
22     spec:
23       containers:
24         - image: nginx:1.17
25           name: nginx
```



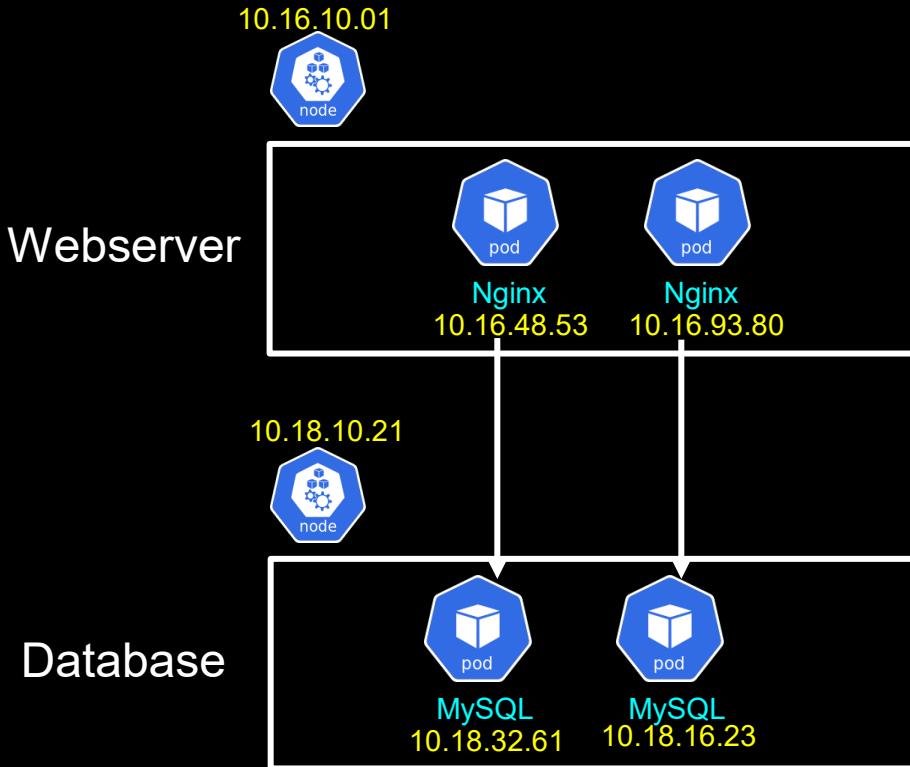
# Replicaset And Deployment Demo

- Implement Deployment
- Delete some stuff!
- Update Container Image via Deployment

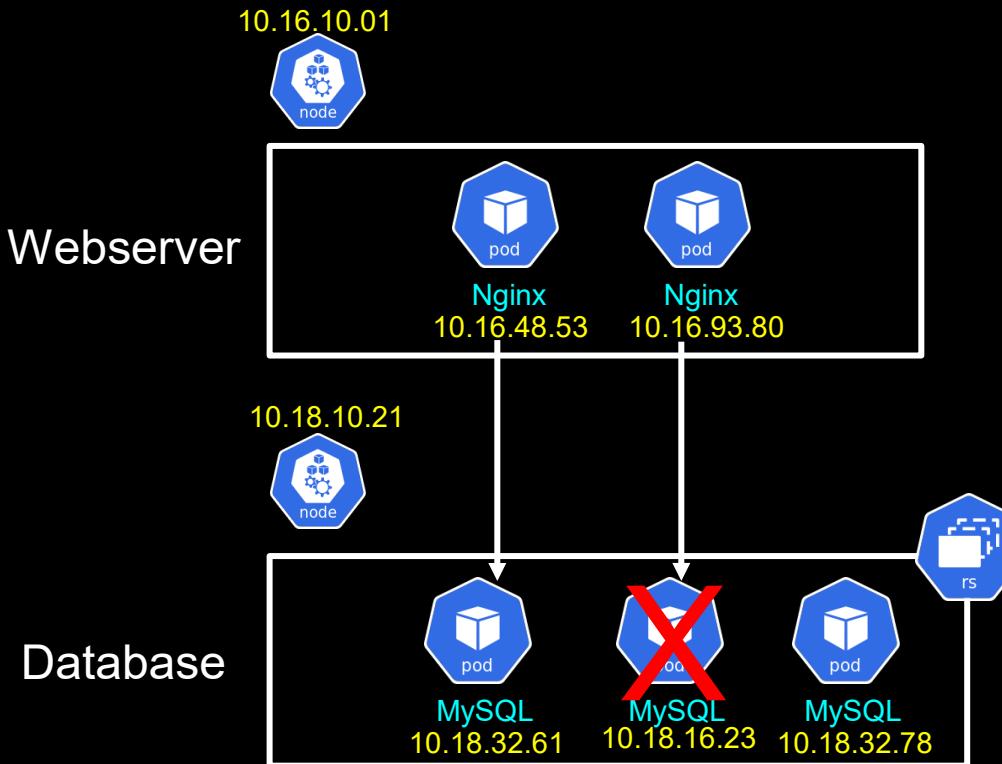
# Service



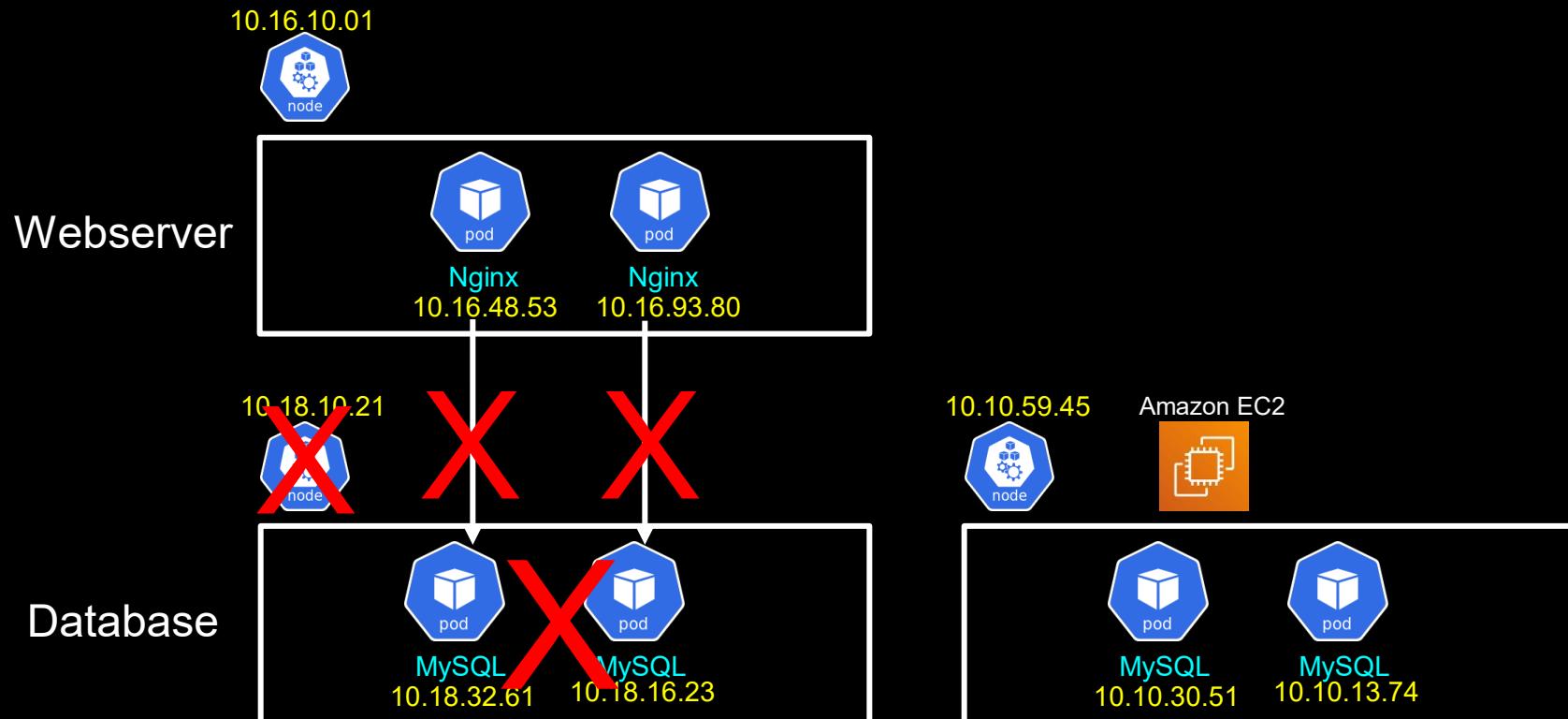
# Life Of A Simple Pod



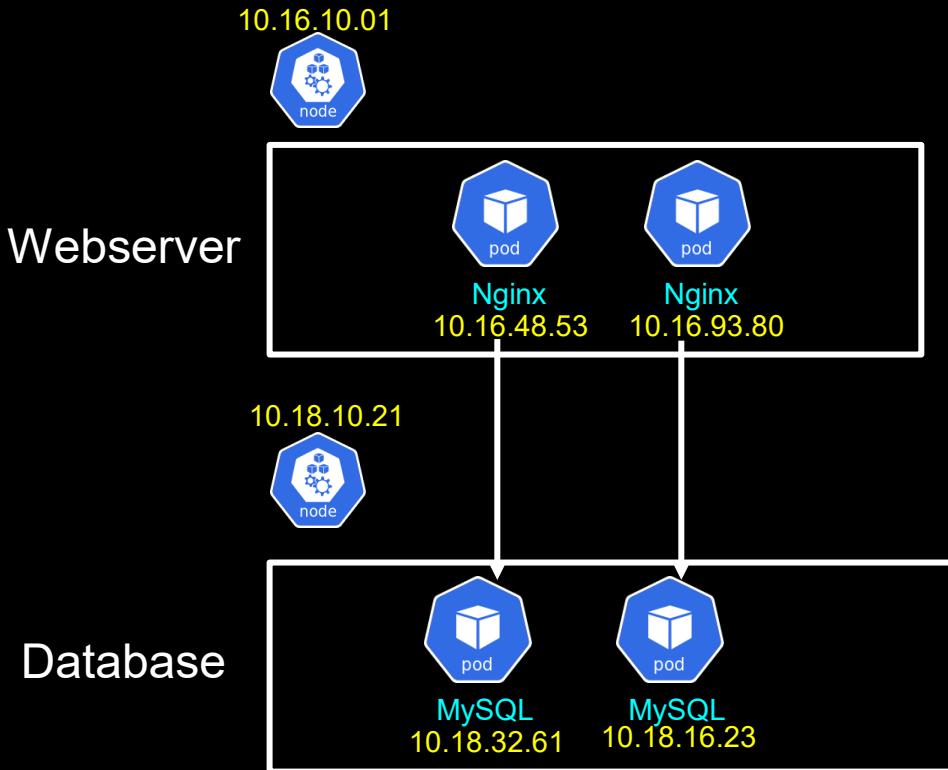
# Life Of A Simple Pod



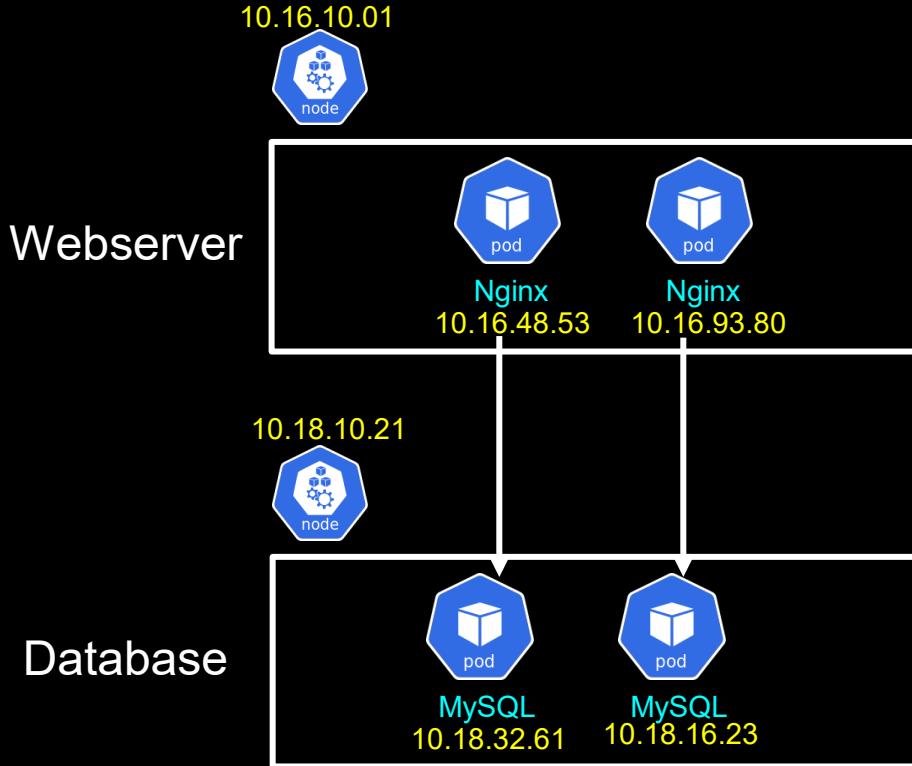
# Life Of A Simple Pod



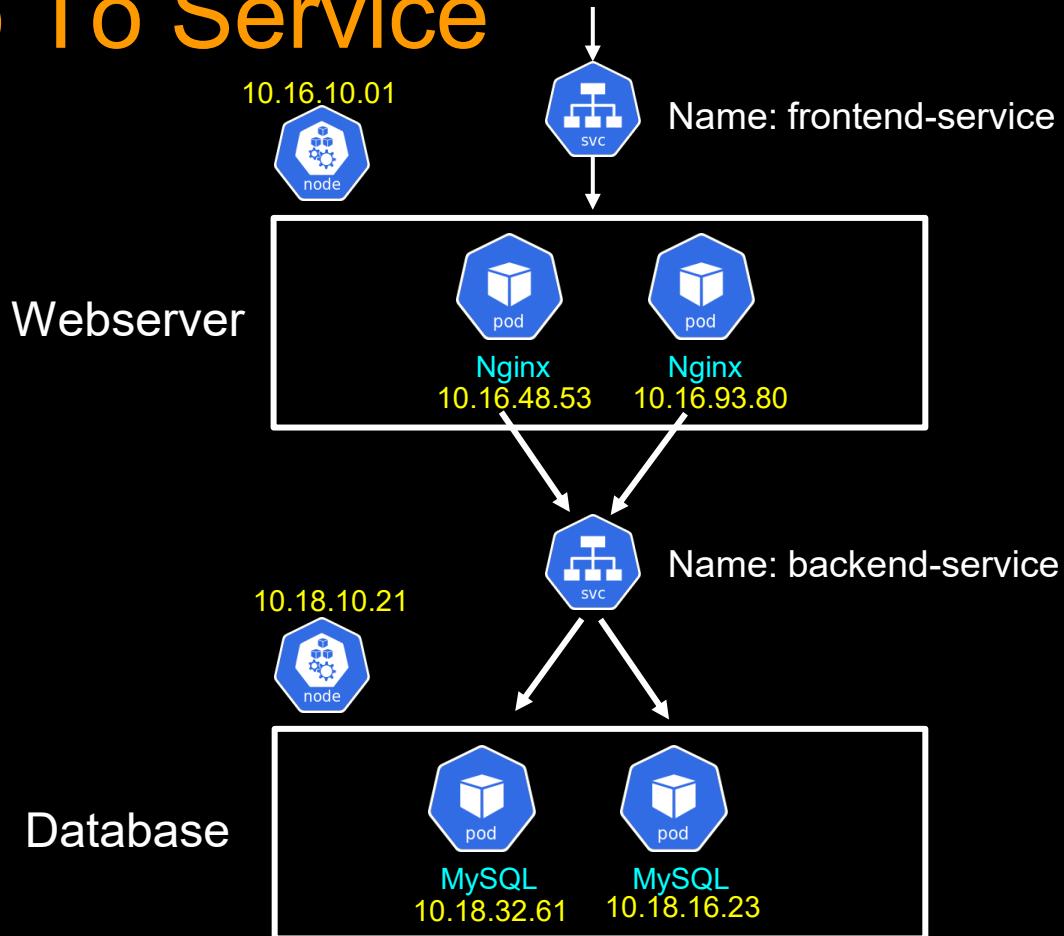
# Life Of A Simple Pod



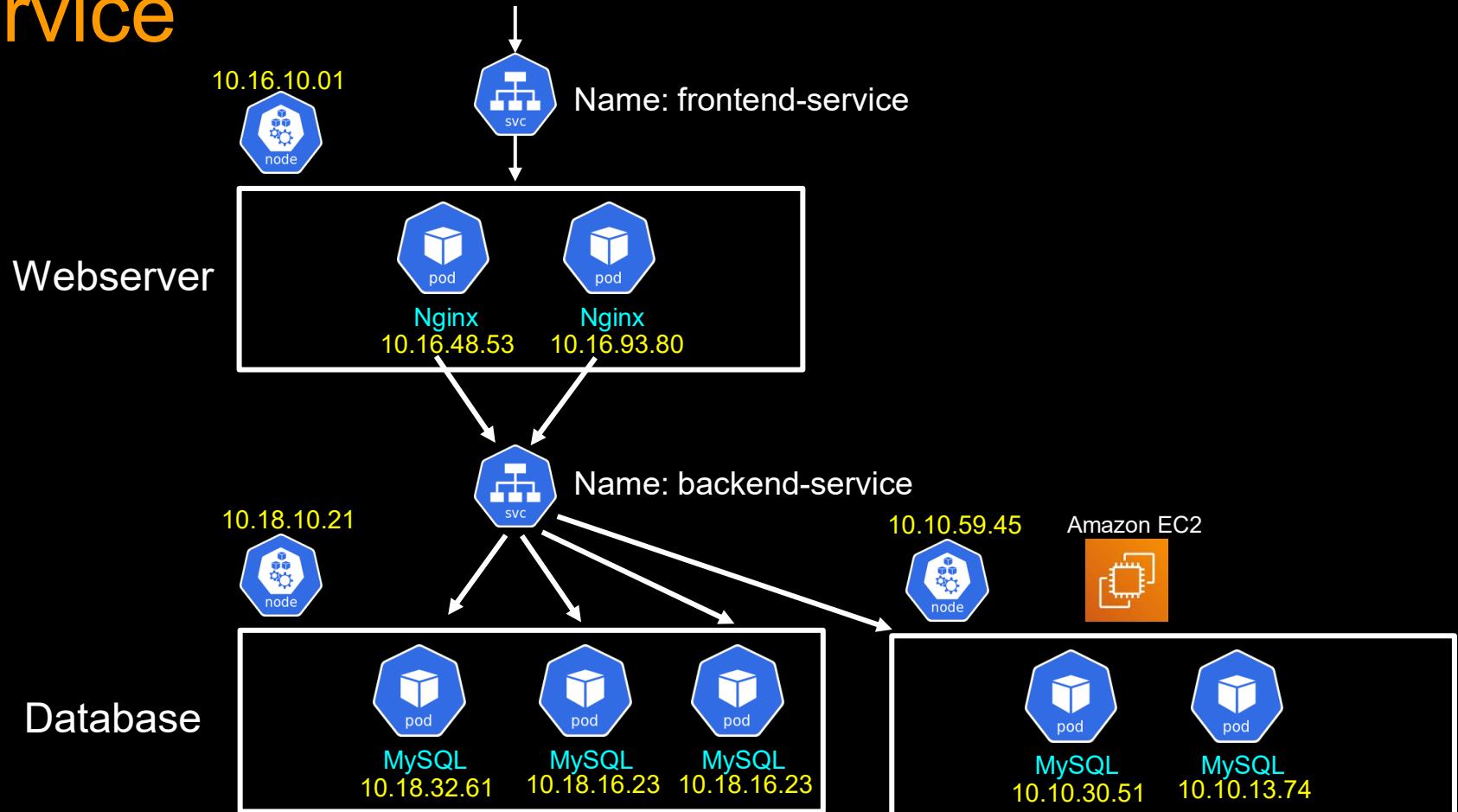
# Say Hello To Service



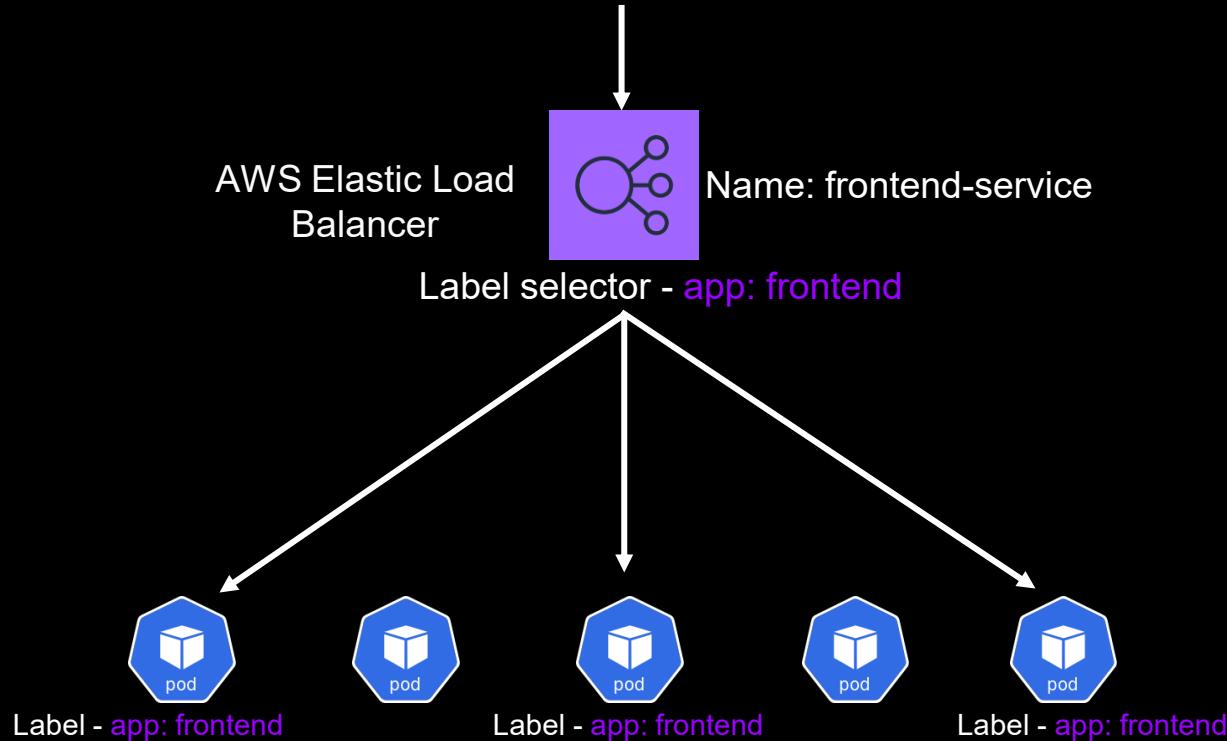
# Say Hello To Service



# Service



# Service



# Different Types of Service

- LoadBalancer
- ClusterIP
- Nodeport

# Service Manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  minReadySeconds: 30
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend-container
          image: nginx
```

! loadbalancer-service.yaml

```
1  apiVersion: v1
2  kind: Service
3  metadata:
4    name: lb-service
5    labels:
6      app: lb-service
7  spec:
8    type: LoadBalancer
9    ports:
10   - port: 80
11    selector:
12      app: frontend
```

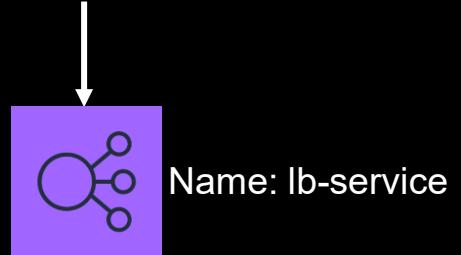
# LoadBalancer Service Manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: frontend-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: frontend
  minReadySeconds: 30
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 1
      maxUnavailable: 0
  template:
    metadata:
      labels:
        app: frontend
    spec:
      containers:
        - name: frontend-container
          image: nginx
```

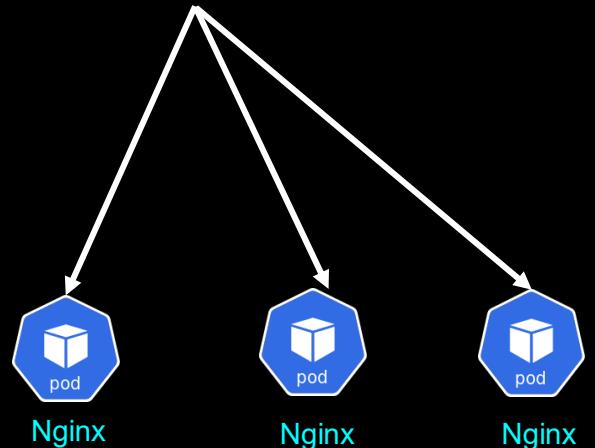
! loadbalancer-service.yaml

```
1   apiVersion: v1
2   kind: Service
3   metadata:
4     name: lb-service
5     labels:
6       app: lb-service
7   spec:
8     type: LoadBalancer
9     ports:
10    - port: 80
11      selector:
12        app: frontend
```

AWS Elastic Load  
Balancer



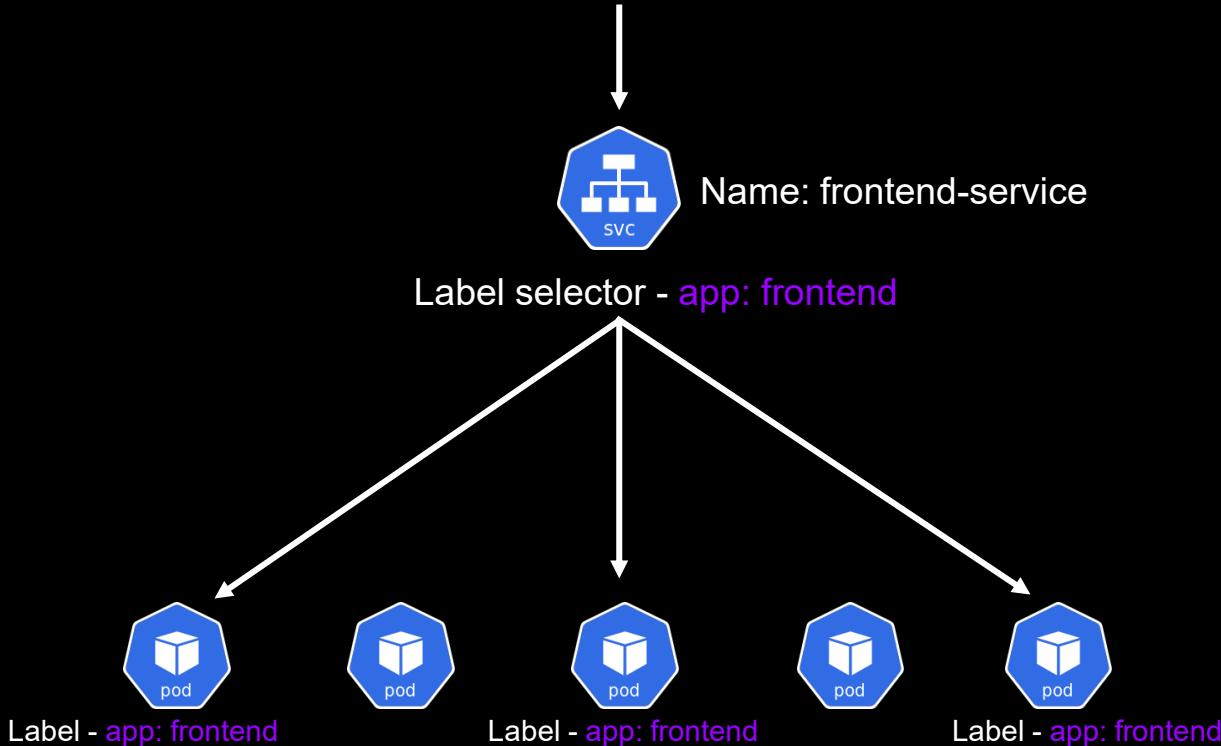
Label selector - app: frontend



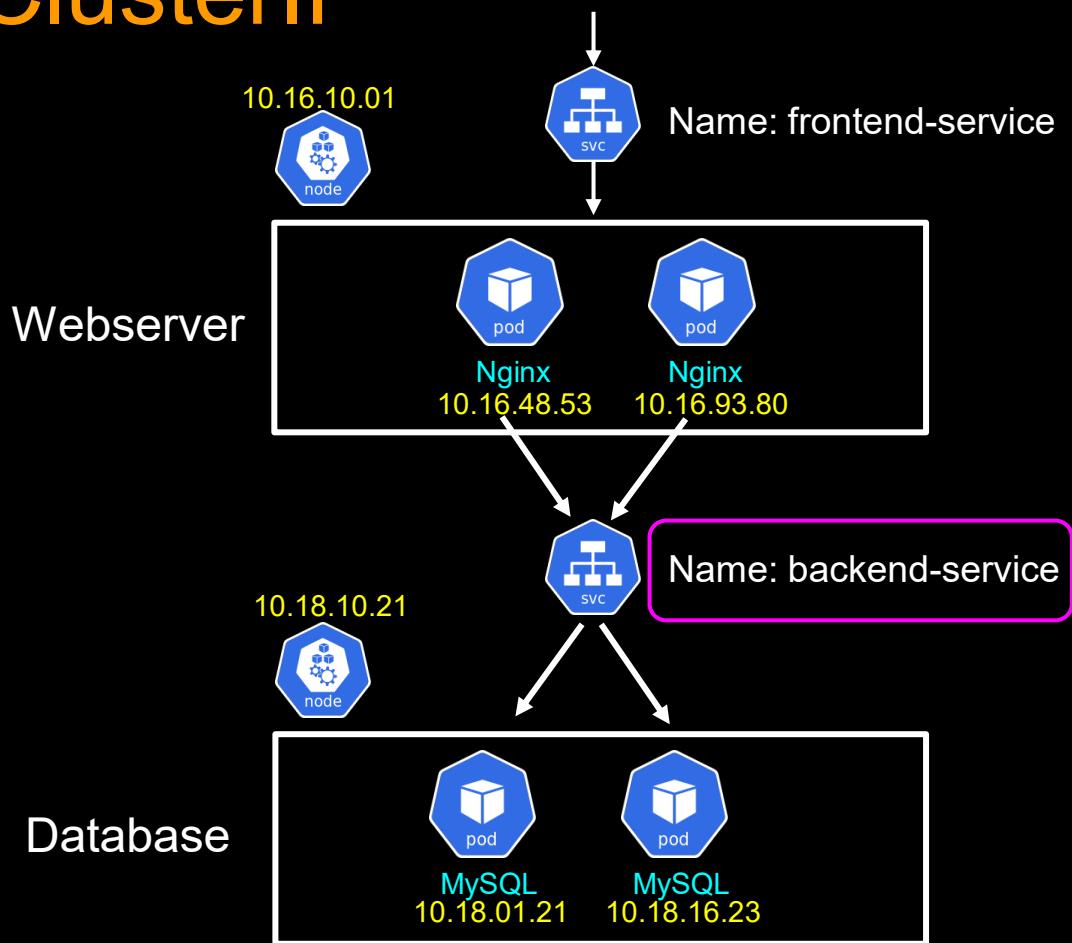
# Different Types of Service

- ClusterIP
- Nodeport
- LoadBalancer

# Motto of all Service



# ClusterIP

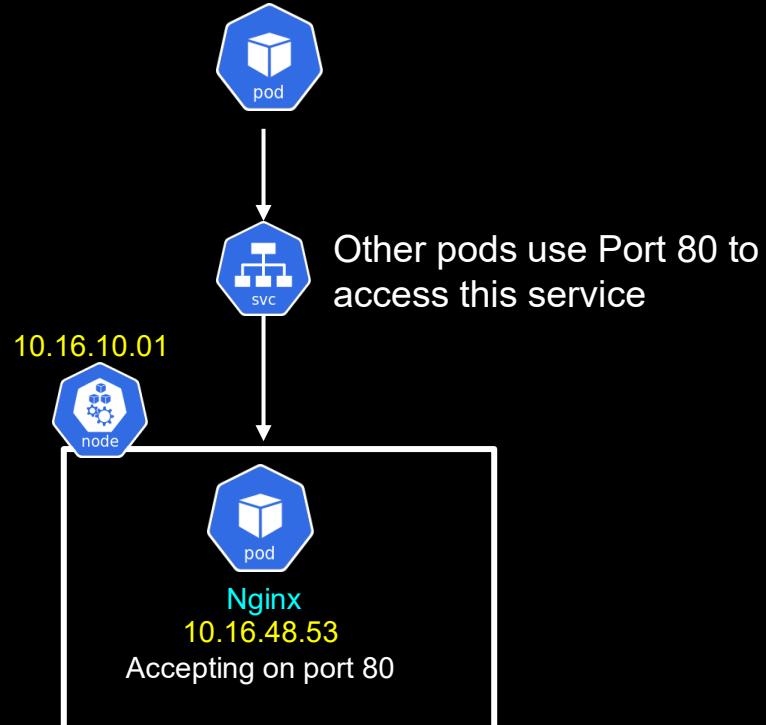


- Default kind of Service
- Only accessible from within cluster

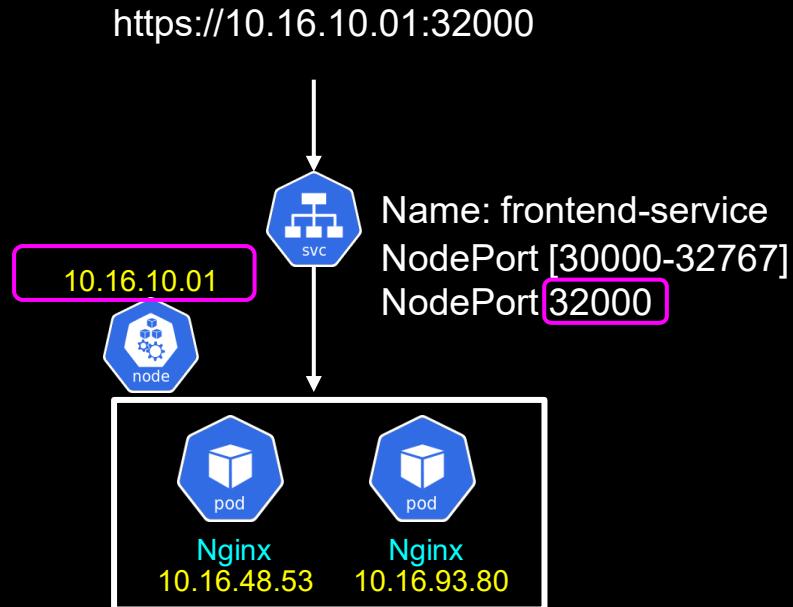
# ClusterIP Manifest File

```
! clusterip-service.yaml
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: app-service
5 spec:
6   ports:
7     - port: 80
8       protocol: TCP
9 selector:
10  app: app-server
11
```

```
16    apiVersion: apps/v1
17    kind: Deployment
18    metadata:
19      name: app-server
20      labels:
21        app: app-server
22    spec:
23      selector:
24        matchLabels:
25          app: app-server
26      template:
27        metadata:
28          labels:
29            app: app-server
30        spec:
31          containers:
32            - name: web-server
33              image: nginx
34              ports:
35                - containerPort: 80
```



# Nodeport

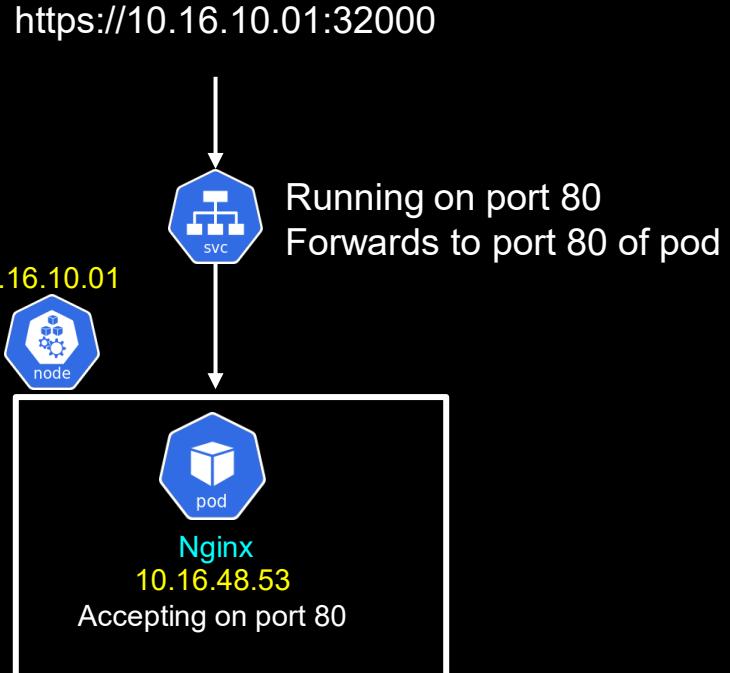


- Accessible from outside cluster
- Creates cluster wide port

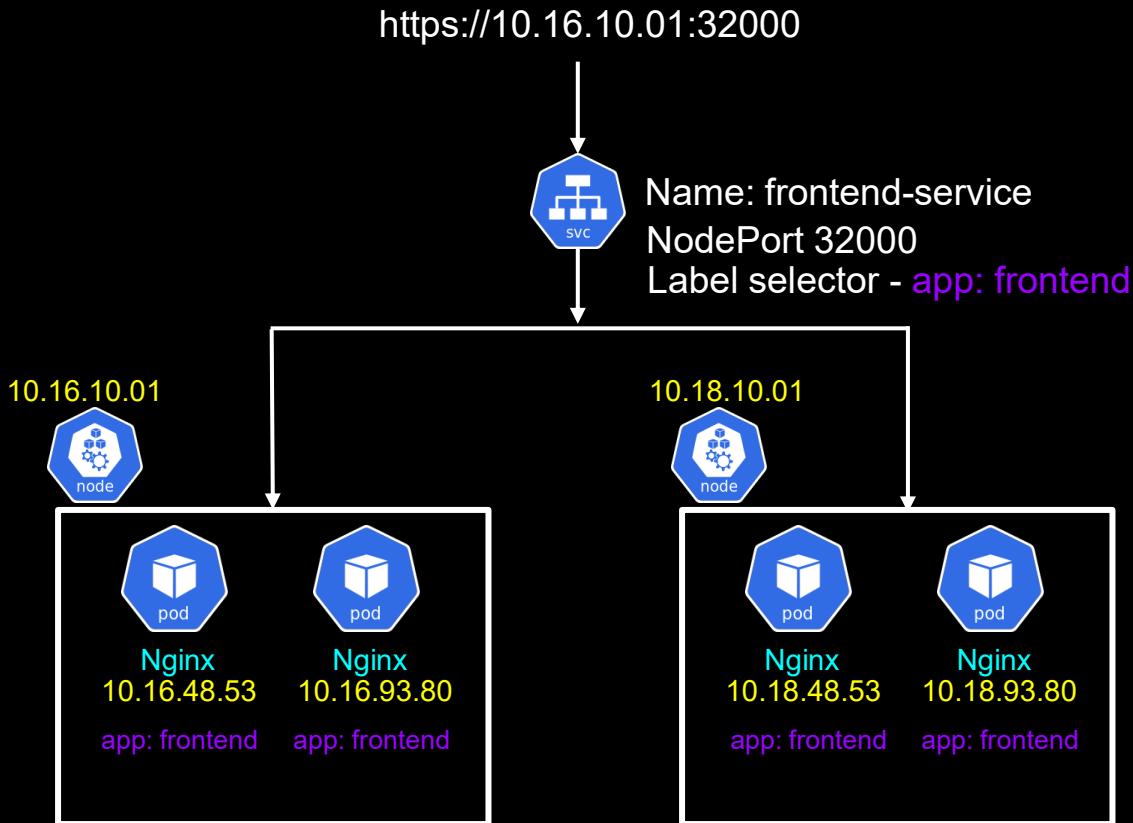
# Nodeport Manifest File

```
! nodeport-service.yaml
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: app-service
5 spec:
6   type: NodePort
7   ports:
8     - nodePort: 32000
9       port: 80
10      targetPort: 80
11    selector:
12      app: app-server
13
```

```
16   apiVersion: apps/v1
17   kind: Deployment
18   metadata:
19     name: app-server
20     labels:
21       app: app-server
22   spec:
23     selector:
24       matchLabels:
25         app: app-server
26     template:
27       metadata:
28         labels:
29           app: app-server
30       spec:
31         containers:
32           - name: web-server
33             image: nginx
34             ports:
35               - containerPort: 80
```

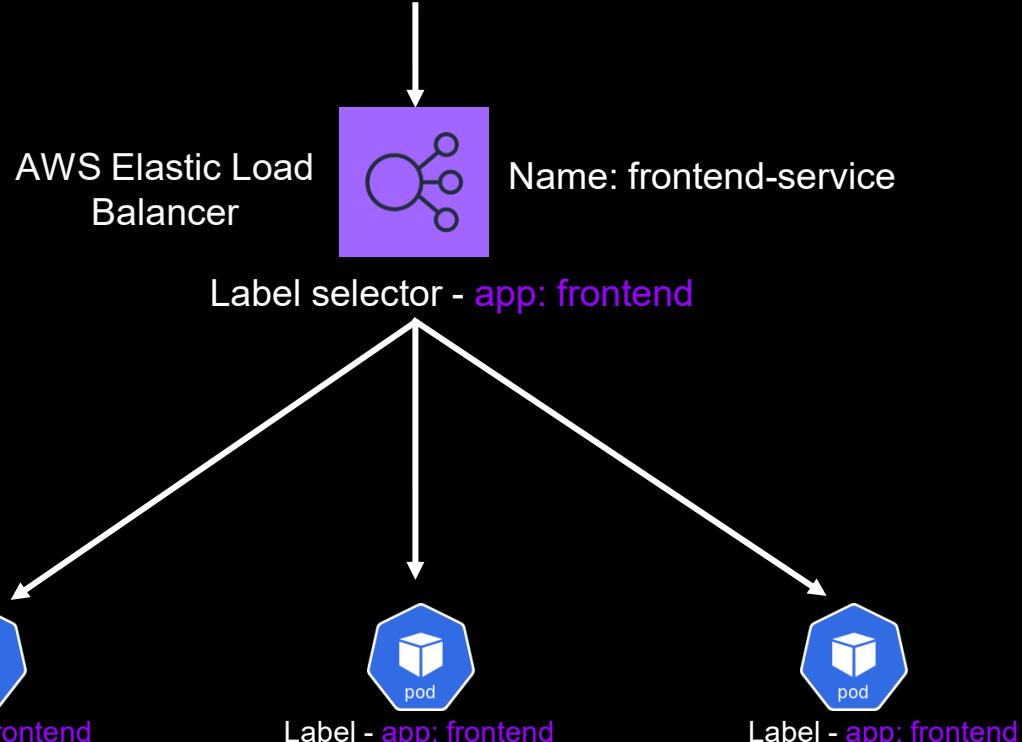


# Nodeport



- Nodeport exposed to all nodes in cluster

# LoadBalancer



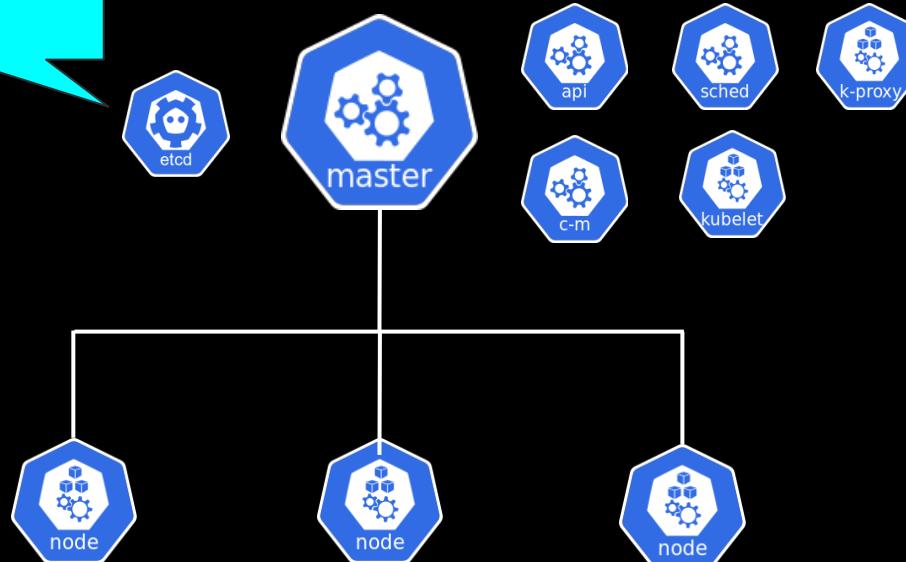
- Cloud Specific Implementation
- Accessible from outside cluster
- Has DNS name
- SSL Termination, WAF Integration, Access Logs, Health Check etc.

# What is EKS?



# Kubernetes Architecture

Better keep me  
alive!



# Kubernetes Control Plane - Self Managed



Amazon EC2

- Need to make Control Plane Highly Available
  - Maintain multiple EC2 in multiple AZ
- Scale Control Plane if needed
- Keep etcd up and running
- Overhead of managing EC2s
  - AMI Rehydration
  - Security Patching
  - Replace failed EC2s
  - Orchestration for Kubernetes Version Upgrade

# Kubernetes Control Plane - AWS Managed



Amazon EKS  
Kubernetes Service

## AWS Manages Kubernetes Control Plane

- AWS maintains High Availability - Multiple EC2s in Multiple AZs
- AWS Detects and Replaces Unhealthy Control Plane Instances
- AWS Scales Control Plane
- AWS Maintains etcd
- Provides Automated Version Upgrade and Patching
- Supports Native and Upstream Kubernetes
- Integrated with AWS Ecosystem

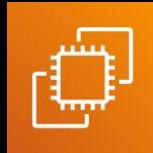
# EKS Data Plane



Amazon EC2

## Self Managed Node Groups

- You maintain worker EC2s
- You orchestrate version upgrade, security patching, AMI Rehydration, keeping pods up during upgrade
- Can use custom AMI



Amazon EC2

## Managed Node Groups

- AWS manages worker EC2s
- AWS provides AMI with security patches, version upgrade
- AWS manages pod disruption during upgrade
- Doesn't work with custom AMI



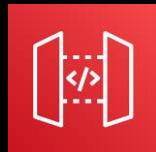
## AWS Fargate

- No worker EC2 whatsoever!
- You define and deploy pods
- Container + Serverless!

# EKS in AWS Ecosystem



AWS Secrets Manager



Amazon API Gateway



Elastic Load Balancing



Amazon EKS



AWS Identity and Access Management (IAM)



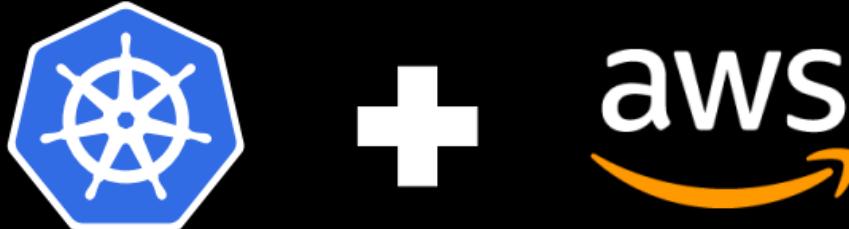
AWS CodePipeline



Amazon CloudWatch

And More....

# Kubernetes on AWS



---

51%

of Kubernetes workloads run on  
AWS today

- CNCF

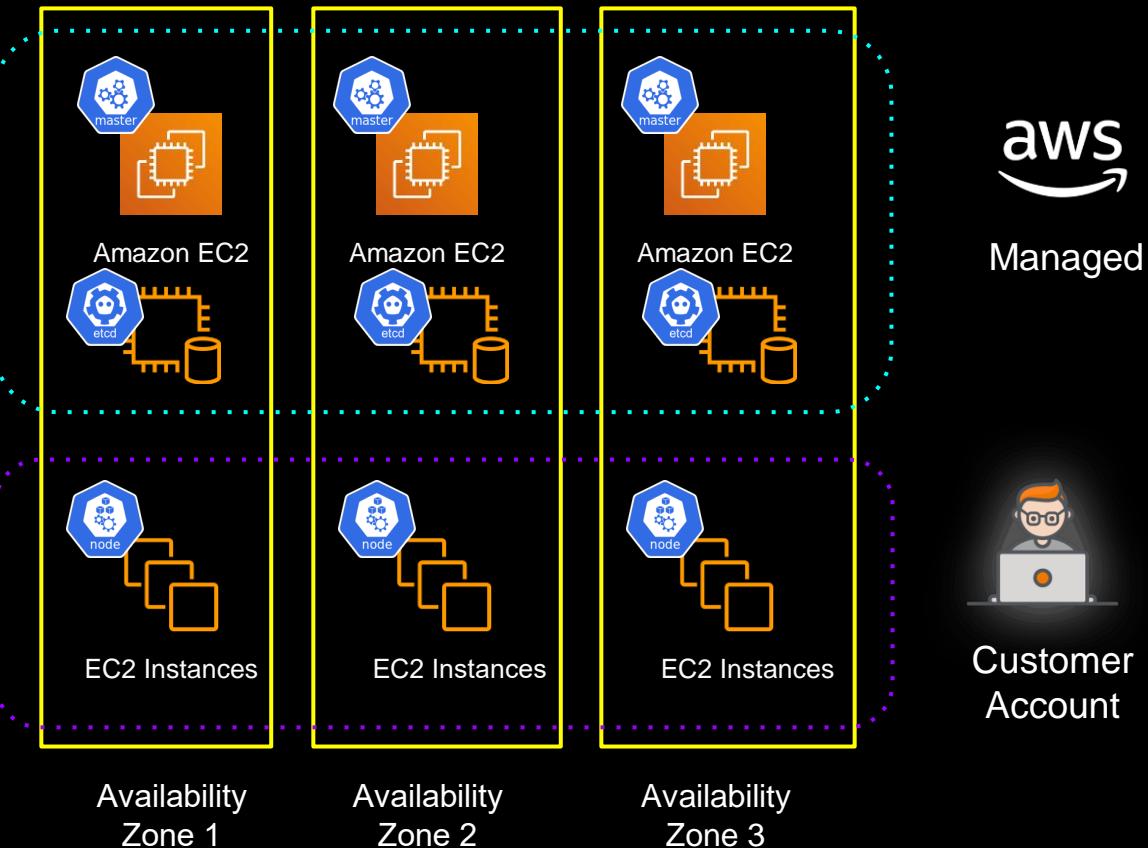
# Amazon EKS Customers



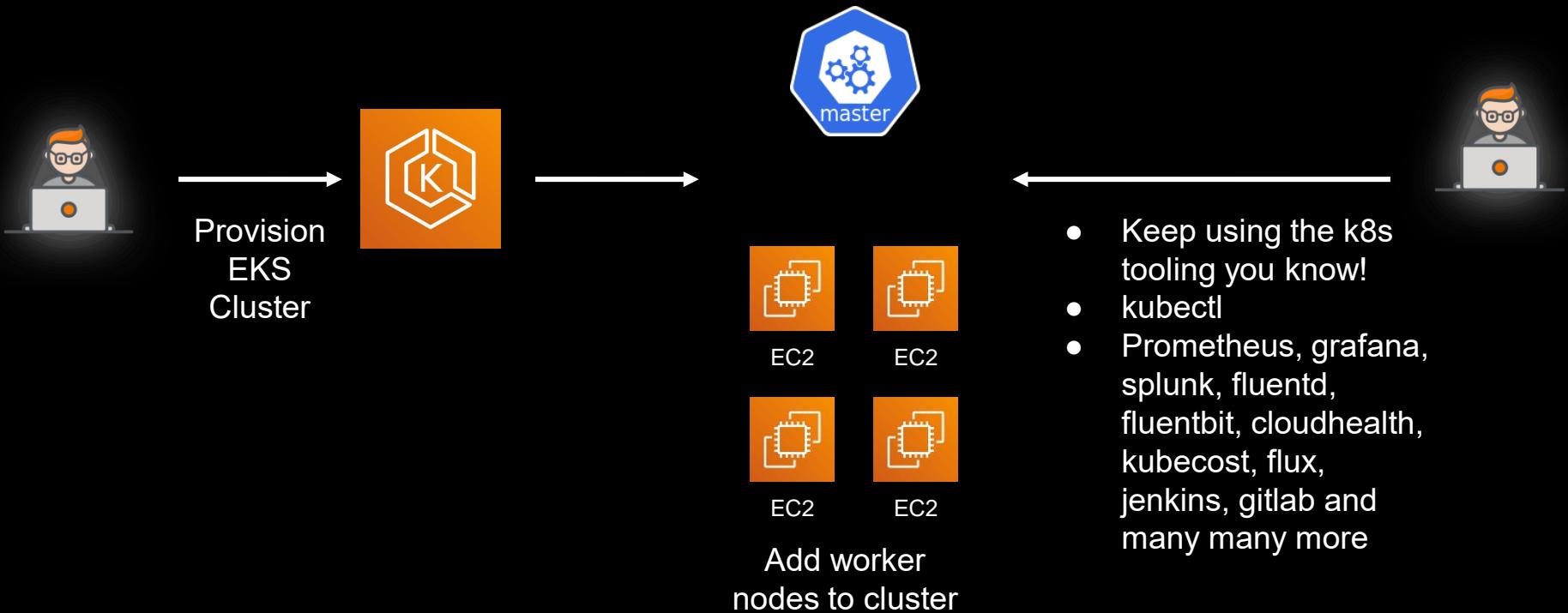
And Many More..

Source: <https://aws.amazon.com/eks/customers/>

# EKS High Level Architecture



# High Level Flow



# Cost Of EKS



# Money Matters



Amazon Elastic Container  
Service for Kubernetes



Flat Charge 10 cents/Hour = \$72/Month



EC2



EC2



EC2

Price based on EC2

$$\begin{aligned} \text{EKS with 3 m5.large in us-east-1} &= \text{EKS Control Plane + Worker Nodes (m5.large X 3)} \\ &= \$72/\text{mo} + \$219.24/\text{mo} \\ &= \$291.24/\text{mo} \end{aligned}$$

EC2 Price calculation: <https://calculator.aws/#/>

# eksctl

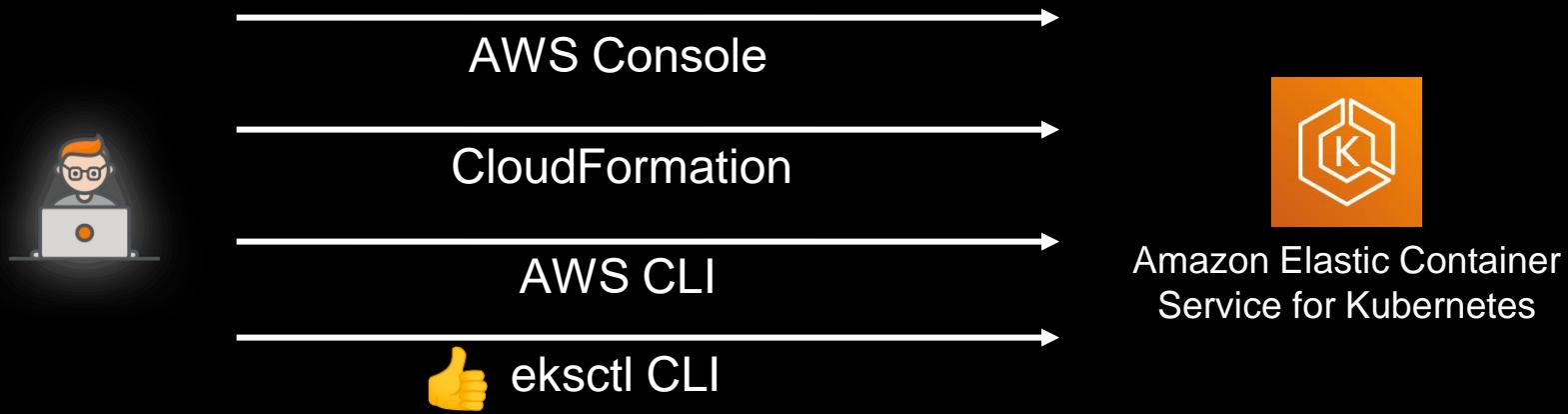


&

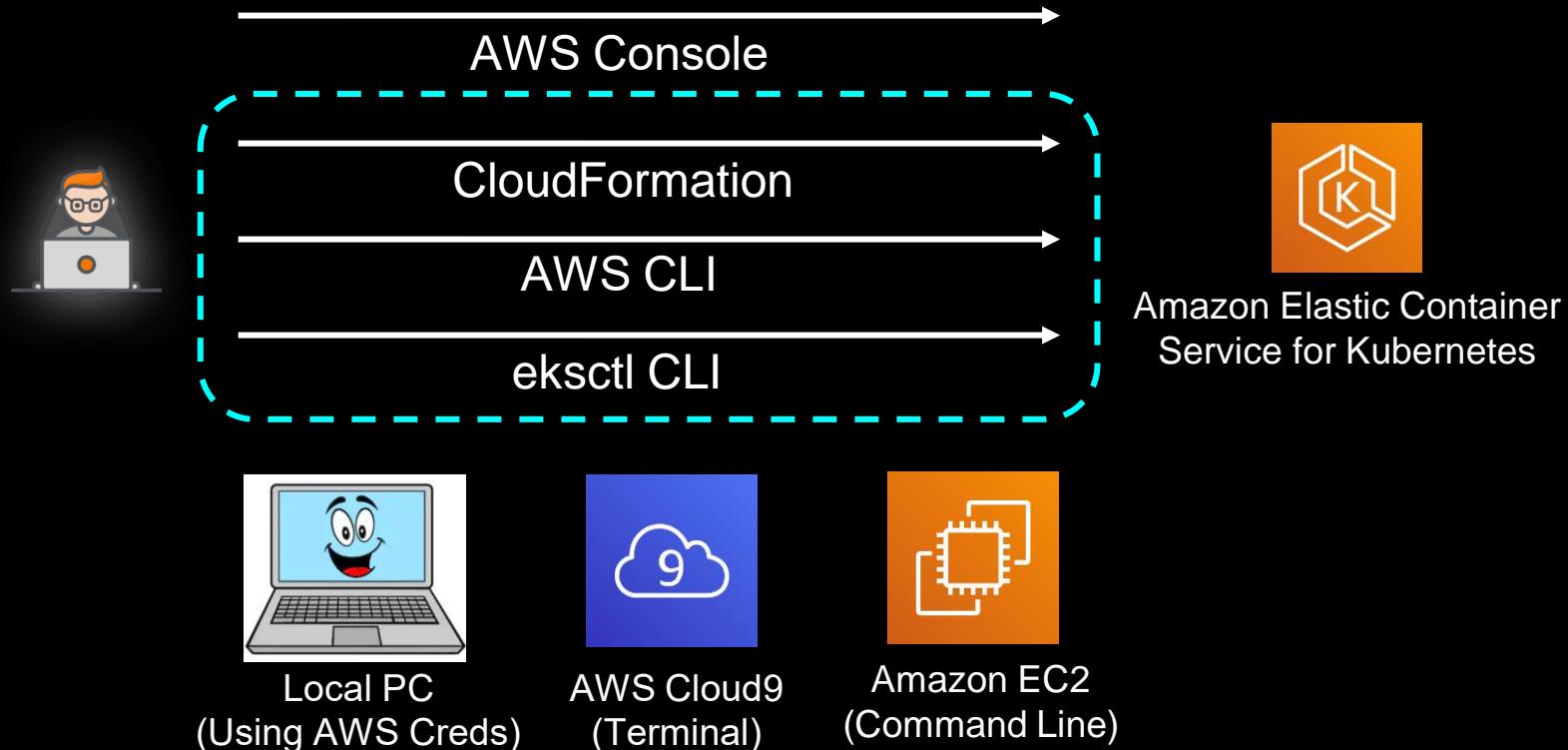
# kubectl



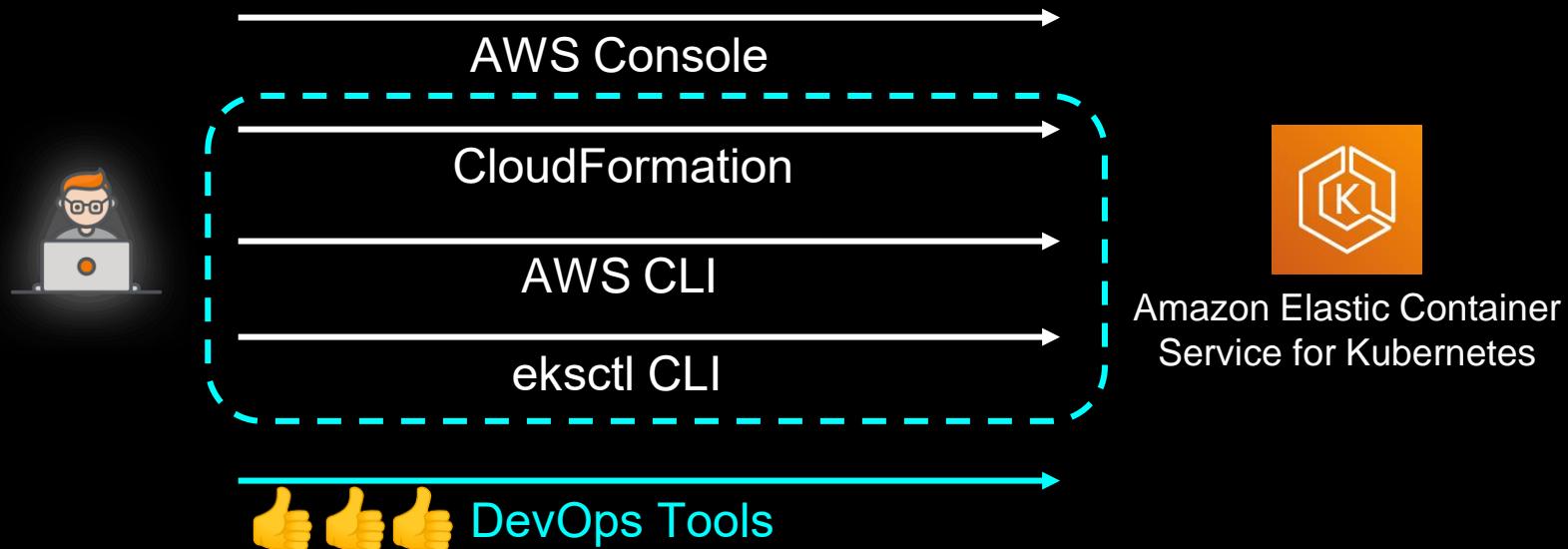
# Ways To Spin Up EKS Cluster



# Learning Medium



# Automation



# What is eksctl?

- CLI tool for creating clusters on EKS
- Easier than console, for real!
- Abstracts lots of stuff - VPC, Subnet, Sec. Group etc.  
using CloudFormation



eksctl create cluster



Amazon Elastic Container  
Service for Kubernetes



AWS Fargate (On EKS)

# Available eksctl features (Only on EKS)

- Create, get, list and delete clusters
- Create, drain and delete nodegroups
- Scale a nodegroup
- Update a cluster
- Use custom AMIs
- Configure VPC Networking
- Configure access to API endpoints
- Support for GPU nodegroups
- Spot instances and mixed instances
- IAM Management and Add-on Policies
- List cluster Cloudformation stacks
- Install coredns
- Write kubeconfig file for a cluster

# eksctl Commands

Command	Brief Description
eksctl create cluster	Create EKS Cluster with one nodegroup containing 2 m5.large nodes
eksctl create cluster --name <name> --version 1.15 --node-type t3.micro --nodes 2	Create EKS Cluster with K8 version 1.15 with 2 t3.micro nodes
eksctl create cluster --name <name> --version 1.15 --nodegroup-name <nodegrpname> --node-type t3.micro --nodes 2 --managed	Create EKS cluster with managed node group
eksctl create cluster --name <name> --fargate	EKS Cluster with Fargate Profile

# What is kubectl?

- CLI for running commands against a cluster on K8s resources
- Communicate via cluster API Server
- Works for any K8s cluster - EKS, K8s on EC2, GKE etc.



Amazon Elastic Container  
Service for Kubernetes



AWS Fargate (On EKS)



Google Kubernetes Engine



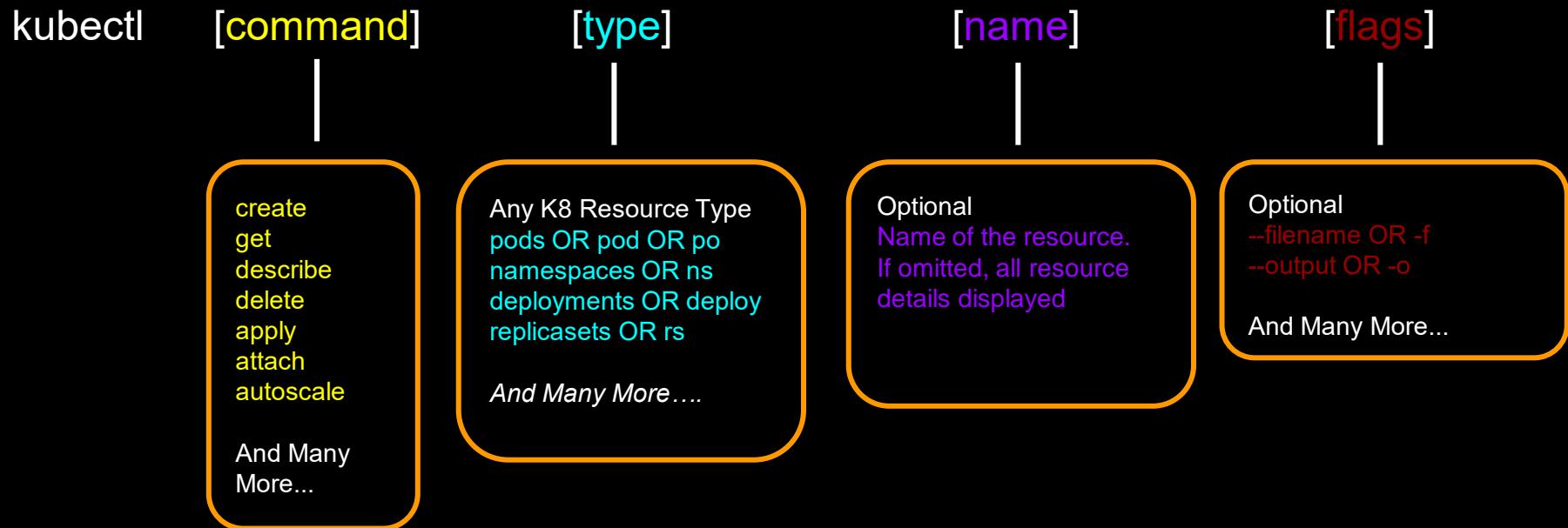
K8s on EC2



kubectl get pod



# kubectl Command Syntax



All available command and resource type : <https://kubernetes.io/docs/reference/kubectl/overview/>

# kubectl Command Syntax

kubectl [command] [TYPE] [NAME] [flags]

kubectl get pod pod1

kubectl get pod

kubectl get po

kubectl get pod pod2 -o yaml



# Most Used kubectl Commands

Command	Brief Description
<code>kubectl apply -f ./manifest-file.yaml</code>	Create resources based on manifest. Declarative Way! Best Way!
<code>kubectl get nodes</code>	List all node info
<code>kubectl get services</code>	List all services
<code>kubectl get pods -o wide</code>	List pods with more details
<code>kubectl get pod my-pod -o yaml</code>	Get a pod's YAML
<code>kubectl get deployment my-dep</code>	List a particular deployment
<code>kubectl exec -it <i>podname</i> -- /bin/bash</code>	Get a shell to the running Container

# DEMO TIME!

- Spin up EKS Cluster using eksctl
- Use kubectl
  - Deploy nginx using manifest file
  - Get resources info

# EC2 Instance Type And Pod Limit

- Max number of allowed pods depends on EC2 Instance Type
- Bigger the instance type, more pods

Note - For some demos, m5.large is used

# EKS Managed Nodegroups



EC2 AMI



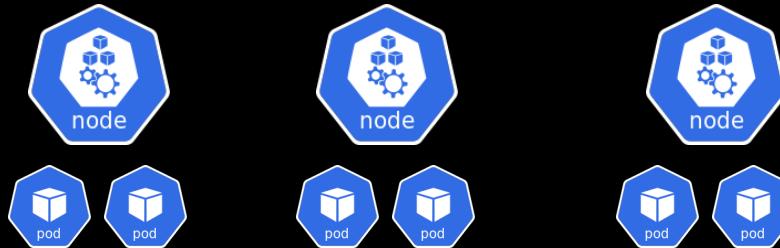
Amazon EC2

# EKS Regular Nodes and Pods



- K8 Version needs to be updated
- Deploy Security Patches
- EC2 AMI Need to Be Updated

EC2 AMI



# EKS Nodes Regular Update

- K8 Version needs to be updated
- Deploy Security Patches
- EC2 AMI Need to Be Updated
  - You need to create AMIs



EC2 AMI



- Application will be DOWN!!
- Or You have to orchestrate HA

# EKS Managed Nodegroups

- Create and Manage EC2 Workers for you
- Amazon releases AMIs with bug fix, security patches for EKS Worker Nodes
  - Custom AMIs not supported (As of April, 2020)
- Automated deployment of updated AMIs with security patches, CVEs
  - No app downtime
  - No overhead of user managed orchestration
  - Auto scaling group is used behind the scenes

# EKS Nodes: Managed Nodegroup Update

- Worker Node K8s Version updated with one click/API call
- AWS Provides AMI with security updates
  - No patching/Rehydrate effort for you



EC2 AMI for K8s

V1.14



EC2 AMI for K8s

V1.15



- Application will be up and running
- AWS orchestrate HA through Auto Scaling Group
- Respects Pod Disruption Budget

# EKS Managed Nodegroups

**DEMO**

# Helm And Charts

## Chart

From Wikipedia, the free encyclopedia

For other uses, see [Chart \(disambiguation\)](#), [Graph \(disambiguation\)](#), and [Diagram](#).

For information about charts in Wikipedia, see [Wikipedia:Graphs and charts](#).



This article has multiple issues. Please help [improve it](#) or discuss these issues on the [talk page](#). (Learn how and when to remove this message) [\[show\]](#)

A chart is a graphical representation of [data](#), in which "the data is represented by [symbols](#), such as bars in a [bar chart](#), lines in a [line chart](#), or slices in a [pie chart](#)".<sup>[1]</sup> A chart can represent [tabular numeric data](#), [functions](#) or some kinds of qualitative structure and provides different info.

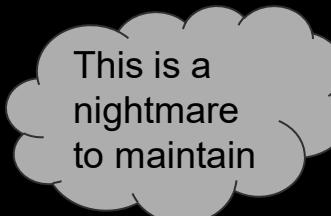
The term "chart" as a graphical representation of [data](#) has multiple meanings:

- A [data chart](#) is a type of [diagram](#) or [graph](#), that organizes and represents data using symbols such as bars, lines, slices, etc.
- [Maps](#) that are adorned with extra information ([map surround](#)) for a specific purpose are often known as charts, such as a [nautical chart](#) or [aeronautical chart](#), typically spread over several [map sheets](#).
- Other domain specific constructs are sometimes called charts, such as the [chord chart](#) in music notation or a [record chart](#) for album popularity.

## Helm Charts have nothing to do with graphical charts or dashboards!

# Helm And Charts - What Does it Solve?

## Deploying Wordpress

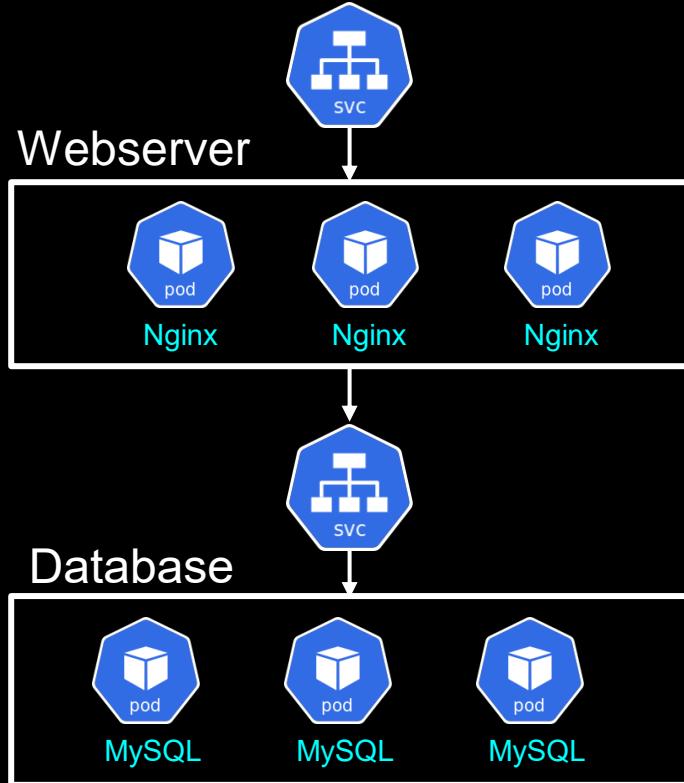


`kubectl apply -f frontservice.yml`

`kubectl apply -f websrvr.yml`

`kubectl apply -f backservice.yml`

`kubectl apply -f database.yml`

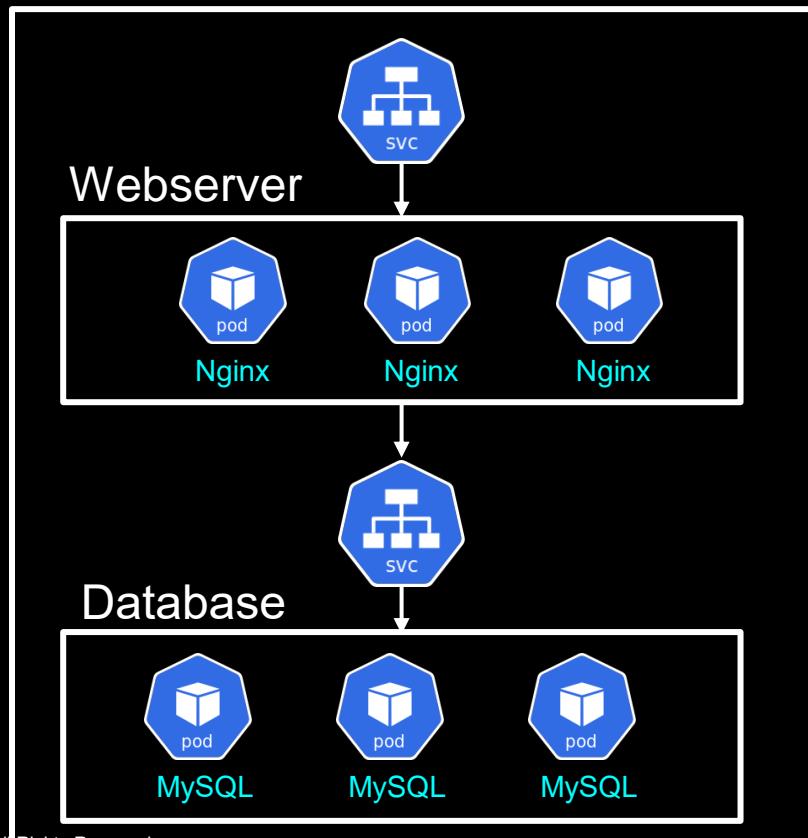


# Helm And Charts - What Does it Solve?

Deploying Wordpress



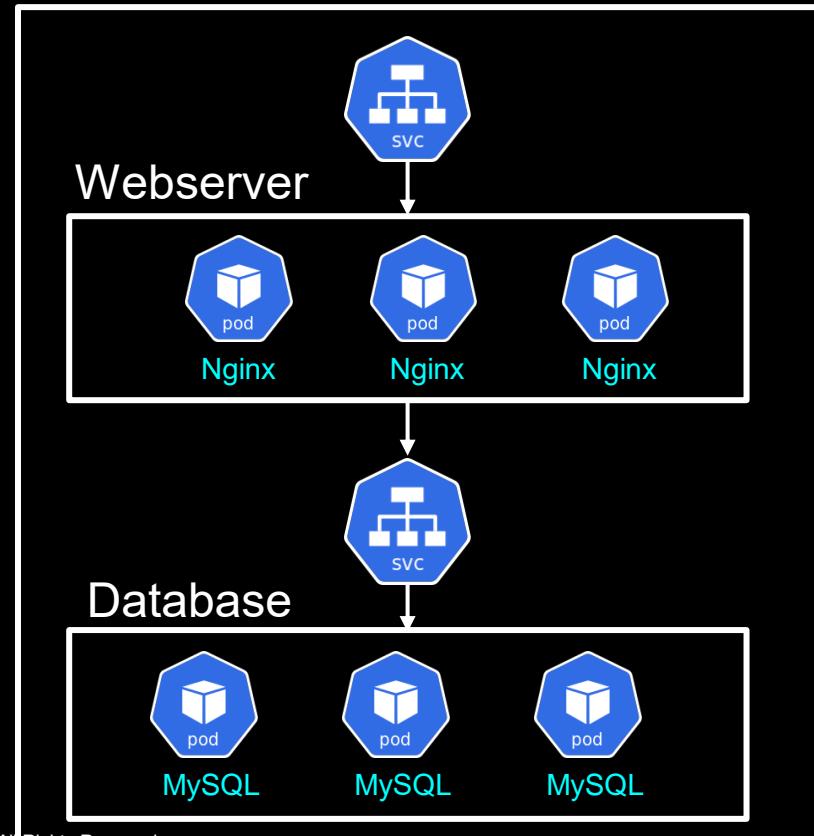
helm install x stable/wordpress



# HELM CHARTS



helm install x  
stable/wordpress

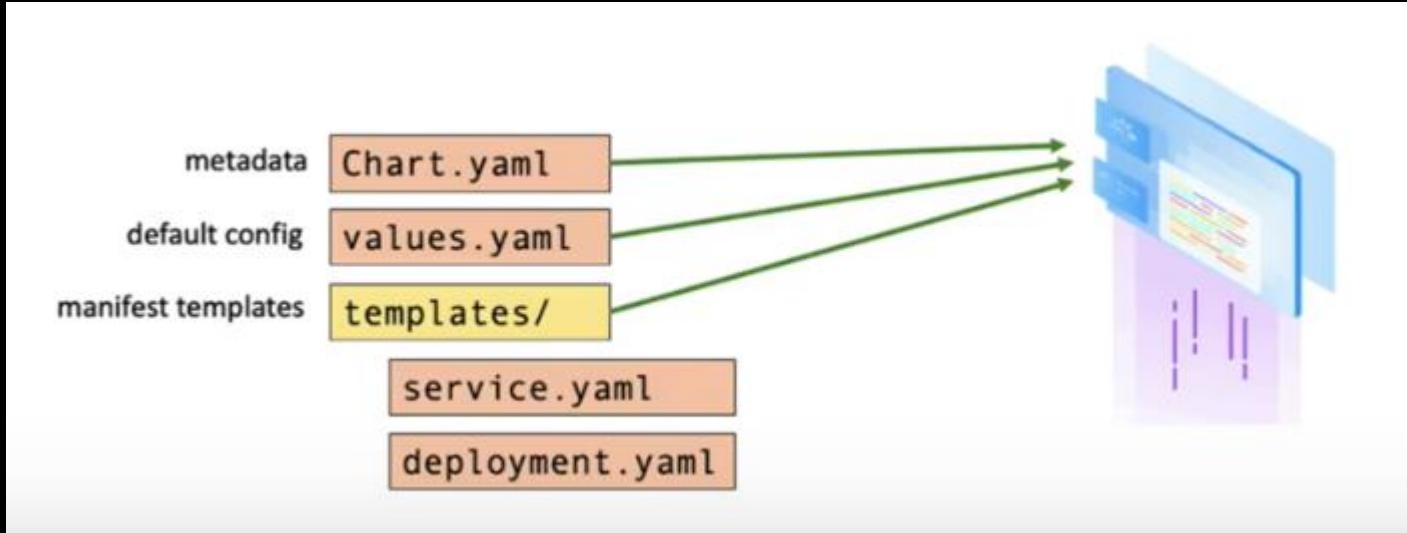


# Helm And Charts

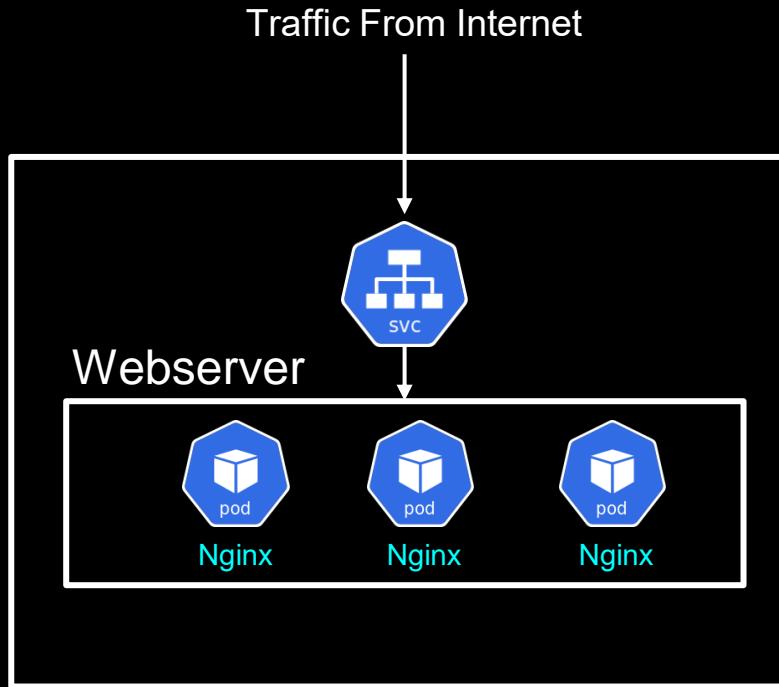


- Helm is package manager for Kubernetes
- Helm packages are called Charts
- Helm Charts help define, install, and upgrade complex Kubernetes application
- Helm Charts can be versioned, shared, and published
- Helm Charts can accept input parameter
  - kubectl need template engine to do this (Kustomize, jinja etc.)
- Popular packages already available

# Charts Files



# Helm On EKS Demo



# EKS Logging & Monitoring

# EKS Logging

- EKS Control Plane Logging
- EKS Worker Nodes Logging

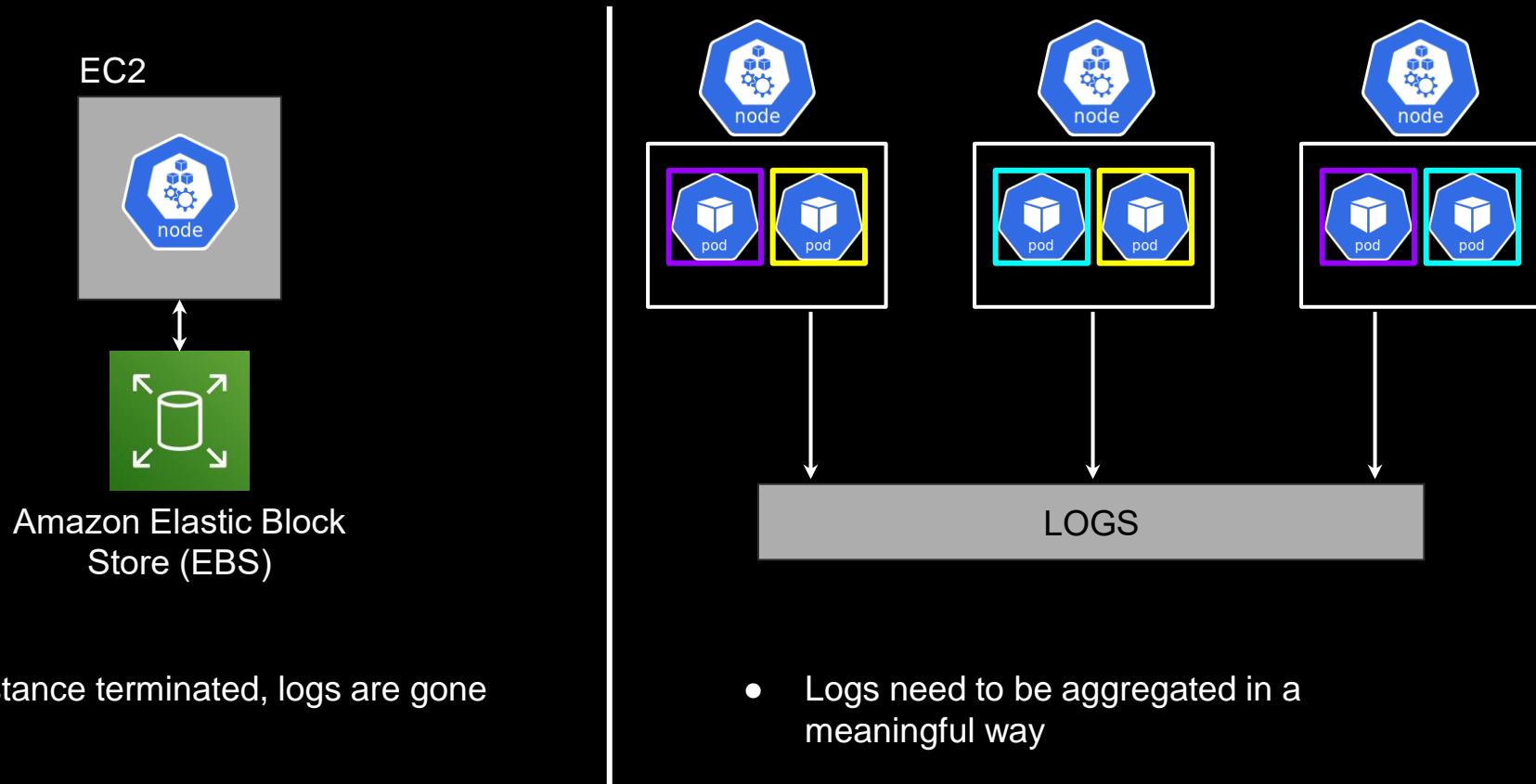
# EKS Logging

- EKS Control Plane Logging
  - K8 api
  - audit
  - authenticator
  - controllerManager
  - scheduler
- EKS Worker Nodes Logging

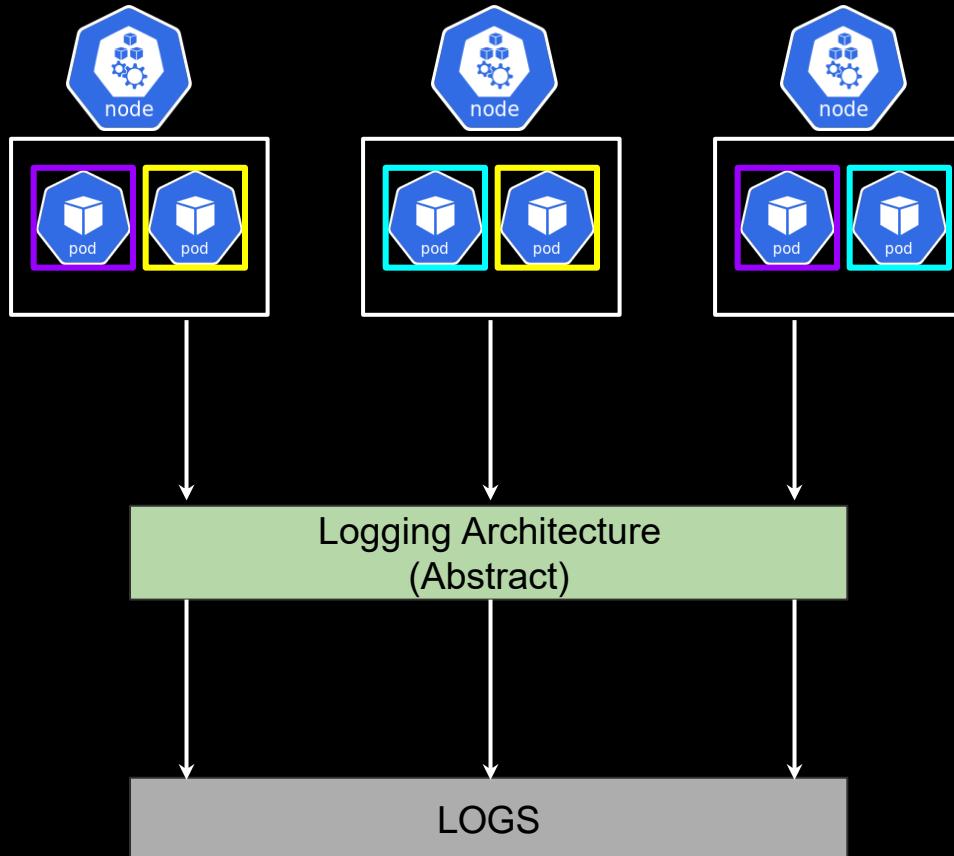
# EKS Logging

- EKS Control Plane Logging
  - K8 api
  - audit
  - authenticator
  - controllerManager
  - scheduler
- EKS Worker Nodes Logging
  - System logs from kubelet, kube-proxy, or dockerd
  - Application logs from application containers

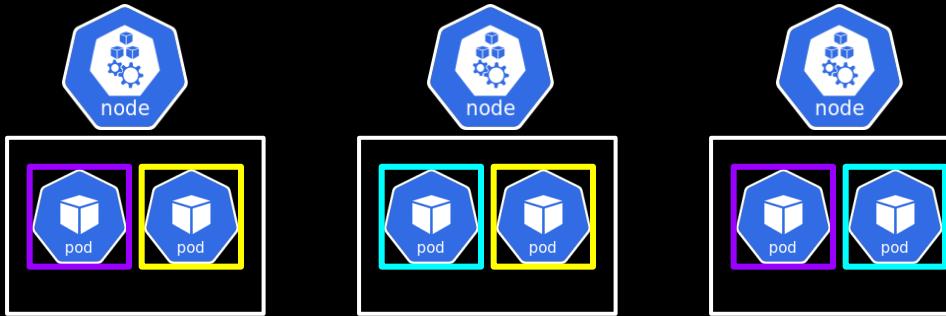
# Logging Caveat



# Logging Caveat

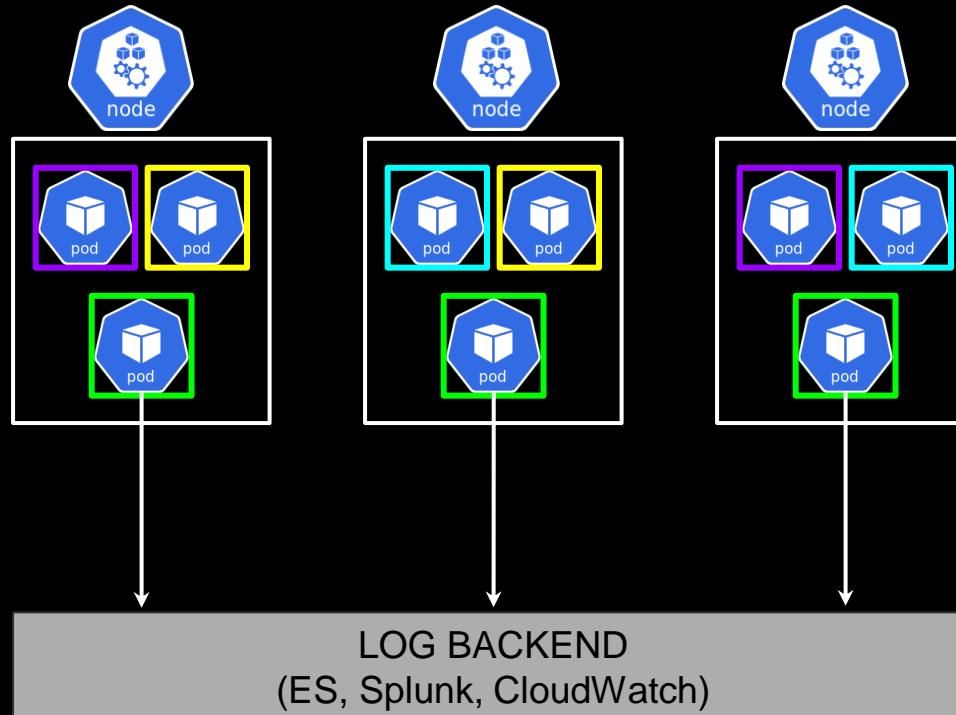


# Kubernetes Worker Nodes



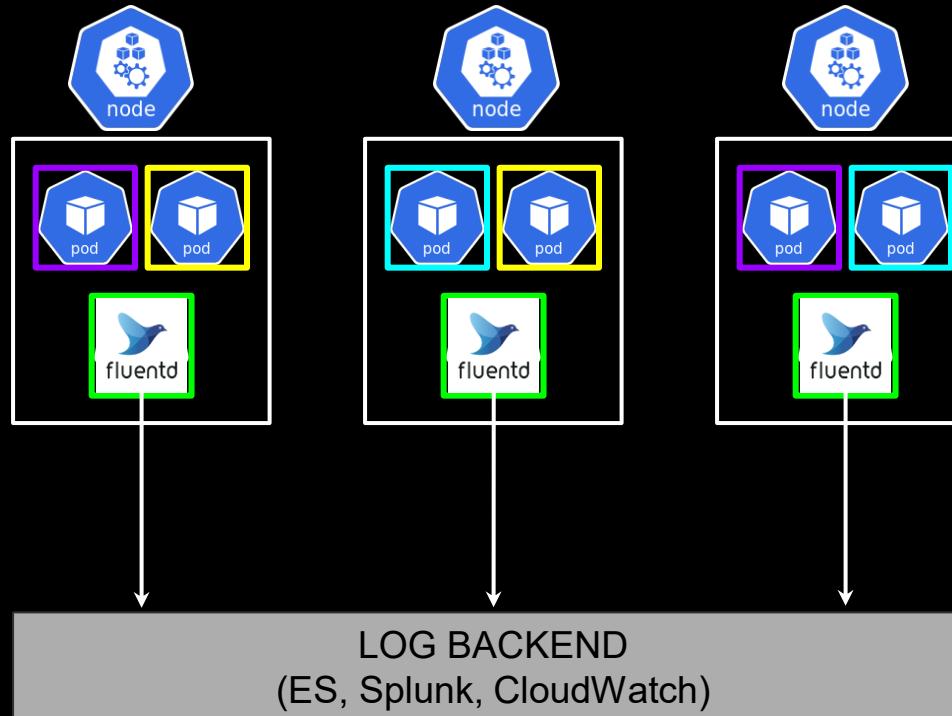
- Containerized application writes to
  - stdout and stderr
- System logs go to
  - systemd
- Container redirect logs to /var/log/containers/\*.log files
- Now we know where to extract logs from!

# Implementation



= Logging agent running as daemon, reading logs from /var/log  
and sending to logging backend

# Implementation



= Logging agent running as daemon, reading logs from /var/log  
and sending to logging backend

# Different Options

Agent



Logging Backend

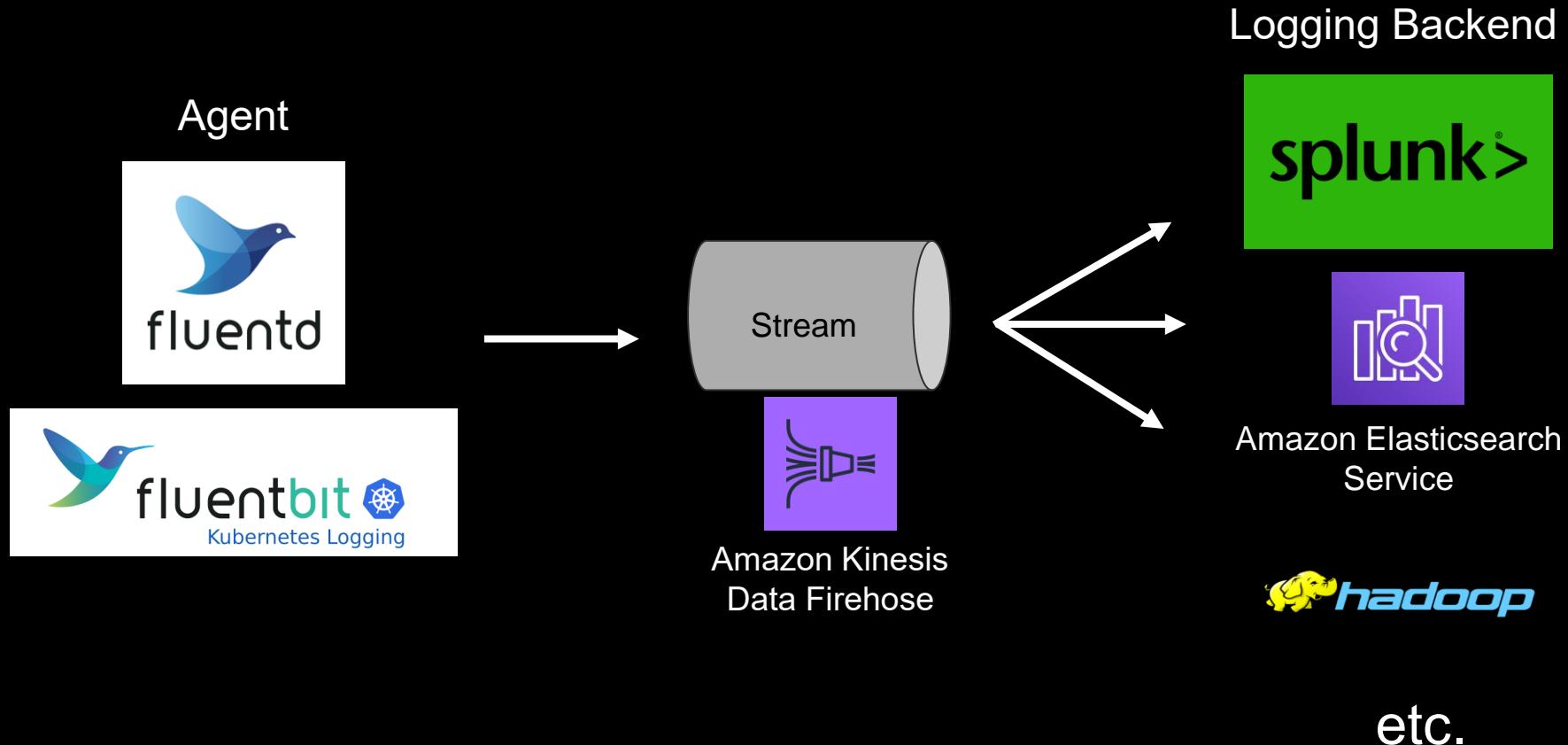


Amazon Elasticsearch  
Service



etc.

# For Streaming to Multiple Log Backends



# EFK Stack



Amazon  
Elasticsearch  
Service



# Under Pressure!



- fluentd has 100+ plugins, fluentbit has around 20 (April 2020)
- However as traffic goes up, fluentd can't keep up
  - fluentd based on Ruby and memory intensive
  - Slow propagation of logs
  - Loss of logs
  - fluentd buffer can be increased to solve this, but not dynamic

Contd on next slide..

# Under Pressure!



- fluentbit is lightweight and keeps up with higher traffic
- Ways to solve the high traffic problem
  - fluentd to Kinesis Data Firehose to Logging Backend
  - fluentbit to Logging Backend
  - Hard to replace fluentd coz of plugin support if already existing in enterprise

# Logging Demo - 1



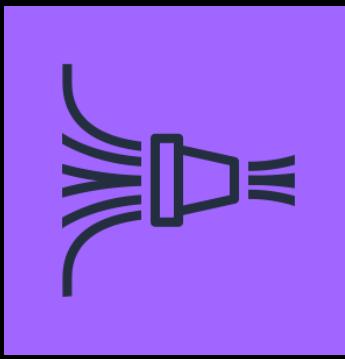
Amazon  
CloudWatch



Amazon  
Elasticsearch  
Service



# Logging Demo - 2



Amazon Kinesis  
Data Firehose



Amazon S3

<https://aws.amazon.com/blogsopensource/centralized-container-logging-fluent-bit/>

# EKS Control Plane Logging

- EKS Control Plane Logging
  - K8s api
  - audit
  - authenticator
  - controllerManager
  - scheduler

# EKS Control Plane Logging

- EKS Control Plane Logging
  - K8s api
  - audit
  - authenticator
  - controllerManager
  - scheduler



Amazon  
CloudWatch

# EKS Control Plane Logging



# EKS Control Plane Logging Demo



# Kubernetes Dashboard

- Web-based Kubernetes user interface
- Overview of applications and resources running on cluster
- Create and modify resources!
  - Pod
  - Deployments
  - Jobs
  - etc

# Kubernetes Dashboard Demo

The screenshot shows the Kubernetes Dashboard interface for the kube-system namespace. On the left, a sidebar navigation bar includes options like Nodes, Persistent Volumes, Roles, Storage Classes, Namespaces (selected), and kube-system. Under Workloads, the Pods option is selected. The main content area features two charts: 'CPU usage' and 'Memory usage'. The CPU usage chart shows usage over time from 11:10 to 11:24, with values ranging from 0.030 to 0.135 cores. The Memory usage chart shows usage over the same period, with values ranging from 143 Mi to 644 Mi. Below the charts is a table listing five pods:

Name	Node	Status	Restarts	Age	CPU (cores)	Memory (bytes)
kubernetes-dashboard-7b9c7b	minikube	Running	0	27 minutes	0	19.746 Mi
heapster-qhq6r	minikube	Running	0	27 minutes	0	18.004 Mi
influxdb-grafana-77c7p	minikube	Running	0	27 minutes	0	43.926 Mi
kube-scheduler-minikube	minikube	Running	0	20 hours	0.01	11.930 Mi
etcd-minikube	minikube	Running	0	20 hours	0.015	58.445 Mi

<https://kubernetes.io/docs/tasks/access-application-cluster/web-ui-dashboard/>

# Monitoring using Prometheus

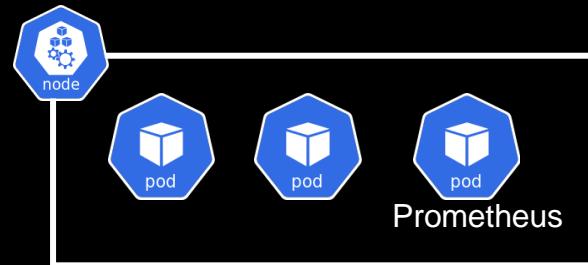
- Monitor Kubernetes cluster
- Query time series data to generate graphs, tables
- Create alerts
- Open source



# Where Does Prometheus Live?

```
kubectl get all -n prometheus
```

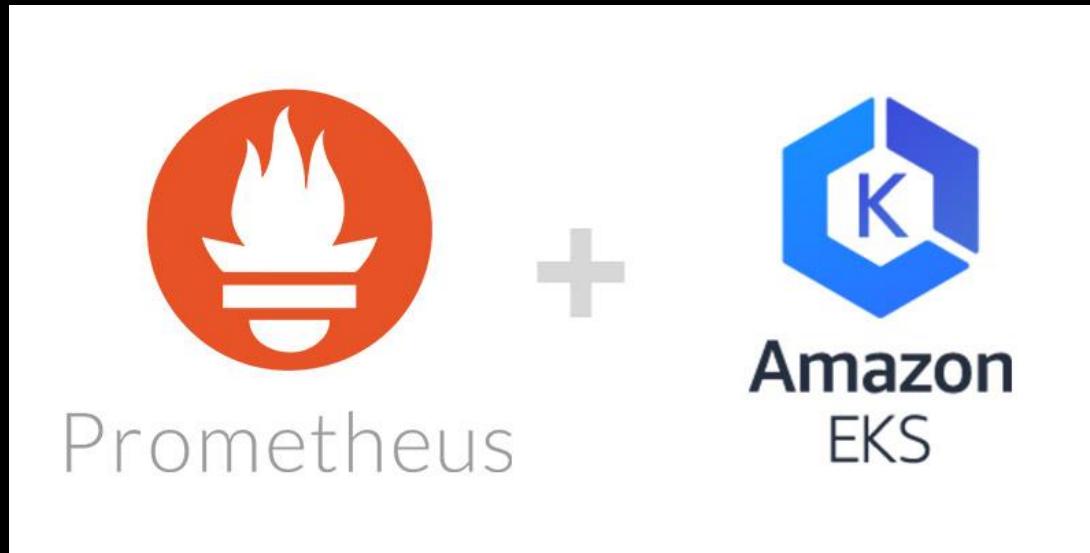
NAME	READY
pod/prometheus-alertmanager-77cfdf85db-s9p48	2/2
pod/prometheus-kube-state-metrics-74d5c694c7-vqtjd	1/1
pod/prometheus-node-exporter-6dhpw	1/1
pod/prometheus-node-exporter-nrfkn	1/1
pod/prometheus-node-exporter-rtrm8	1/1
pod/prometheus-pushgateway-d5fdc4f5b-dbmr9	1/1
pod/prometheus-server-6d665b876-dsmh9	2/2



# Prometheus Deep Dive



# Prometheus Demo



# Graphs using Grafana

- Visualize metrics
- Works out of the box with Prometheus
- Create alerts
- Open source



Grafana

# Grafana Demo



Grafana

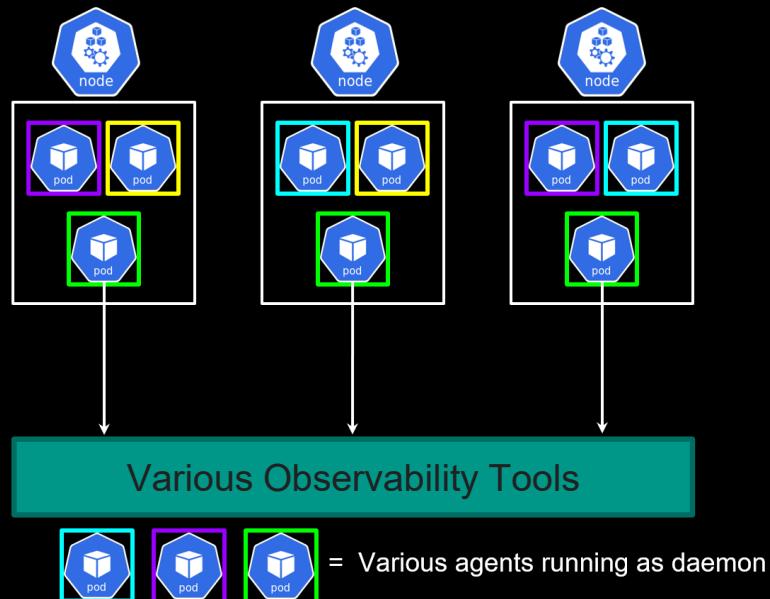


Grafana

# ADOT (AWS Distro for Open Telemetry)

# Metrics Logs General Flow & Pain Point

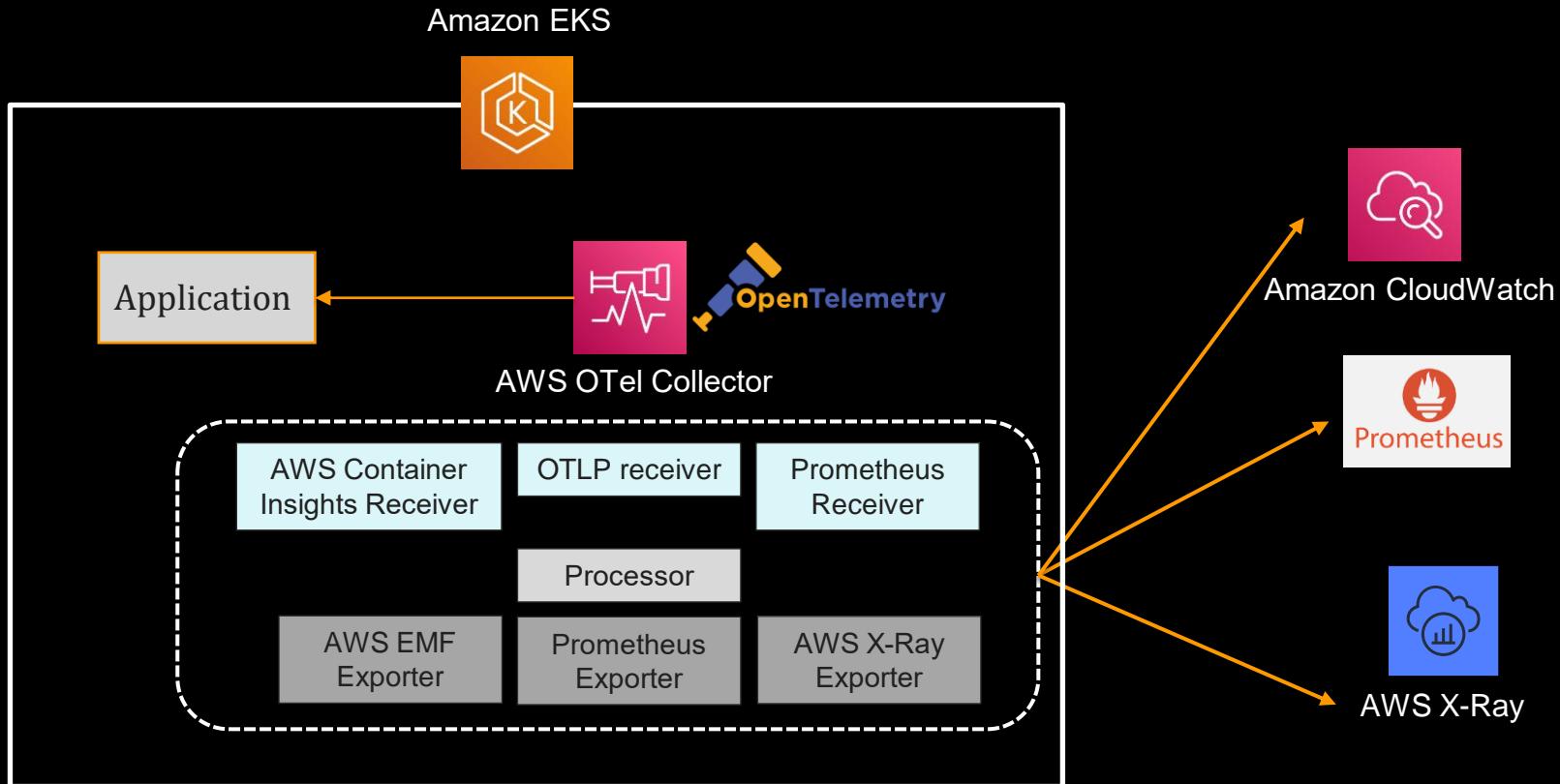
- Select the tool of choice
  - Prometheus, Grafana, Container Insights, Fluent Bit, FluentD etc.
- Install the tool
- Pain points
  - Maintain all these separate tools
  - You are in charge of security



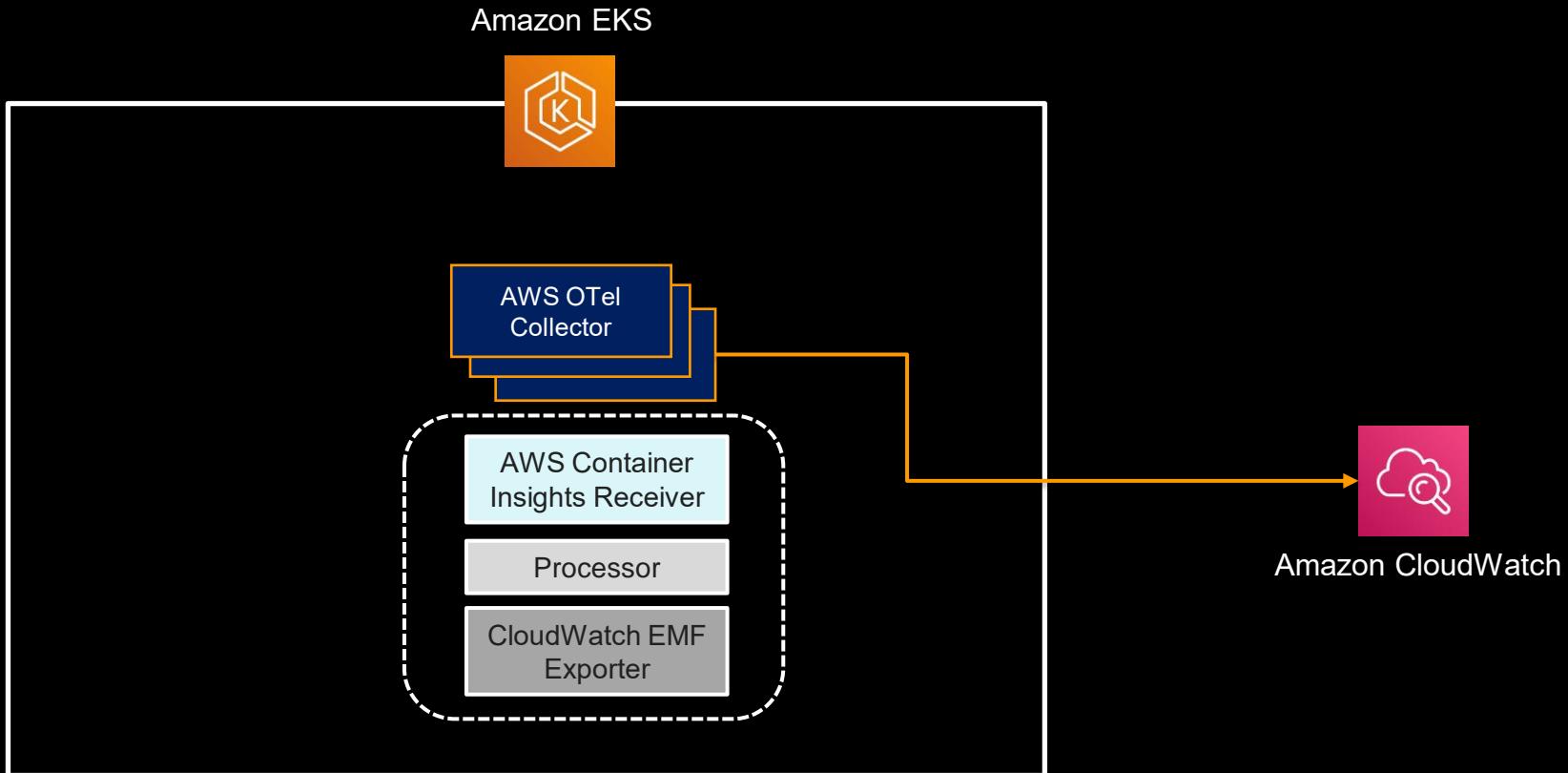
# One Tool to Rule Them All – ADOT!

- Open Telemetry – open source APIs, libraries, and agents to collect traces and metrics
- AWS-supported open telemetry project
- Install this “super agent” once and forward logs, metrics, and traces to tool of your choice
  - Vendor agnostic
- Includes latest security patches and bug fixes
  - Validated by AWS
- Currently in open preview

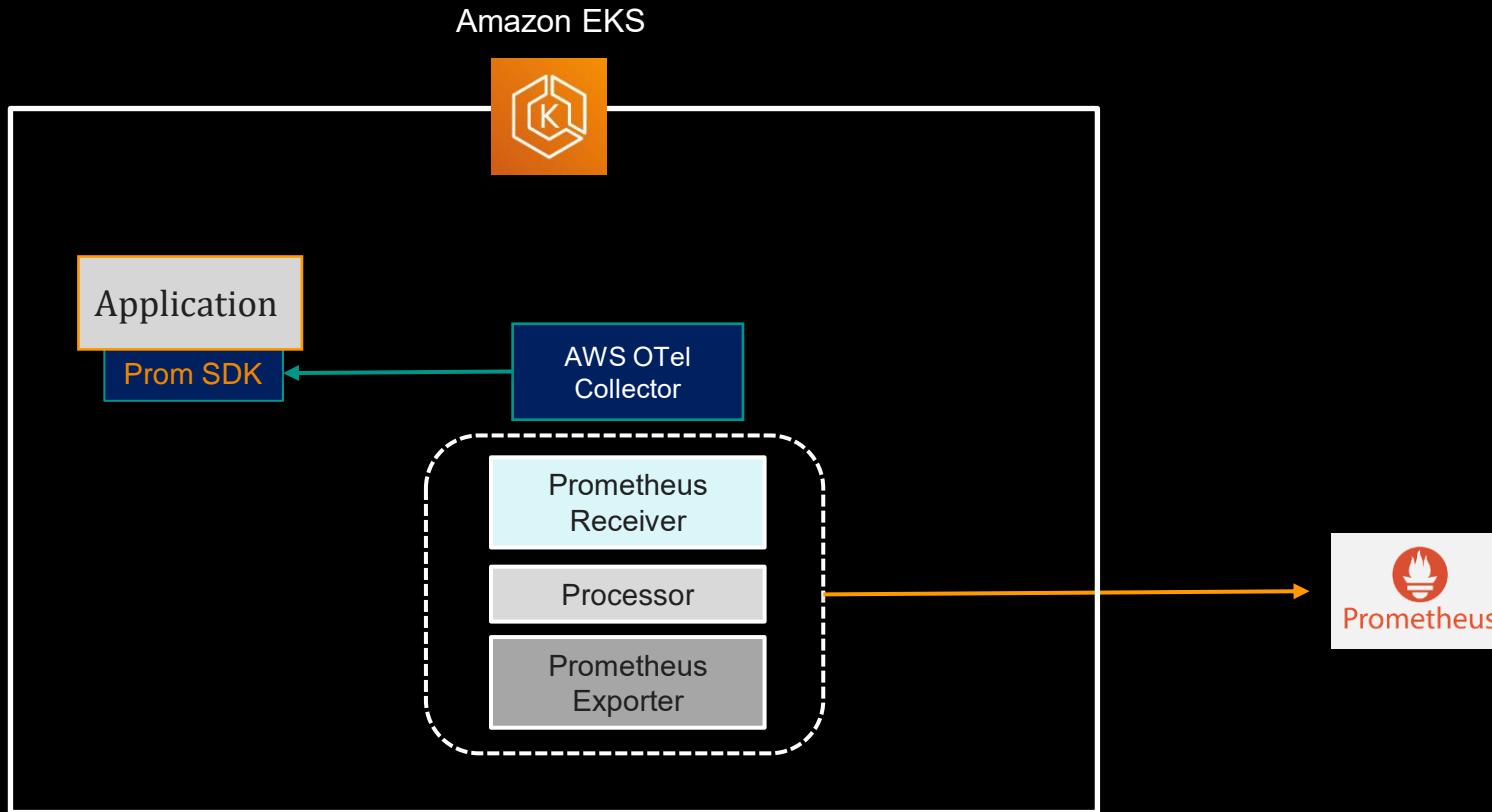
# ADOT Diagram



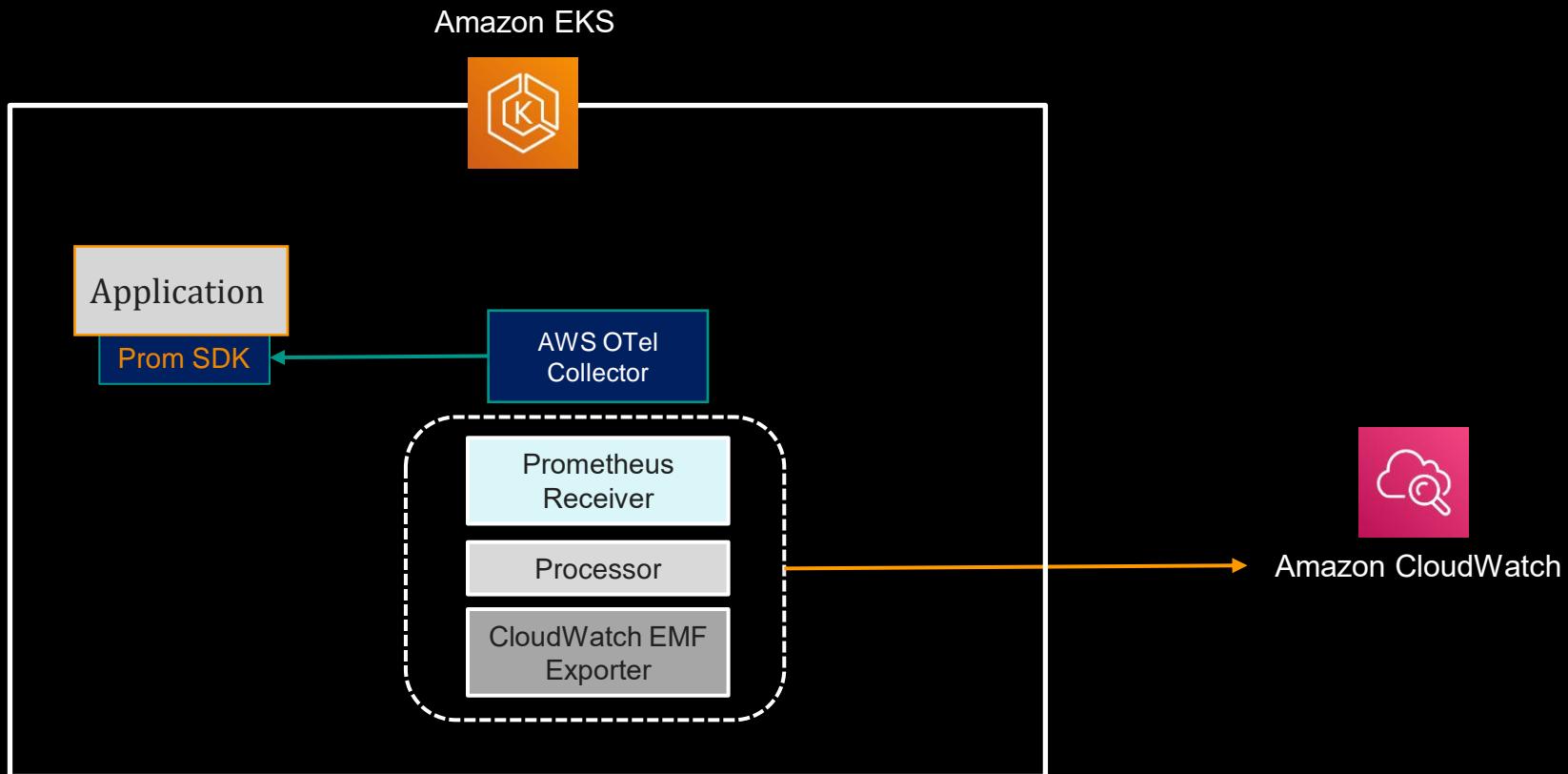
# CloudWatch Container Insights with ADOT



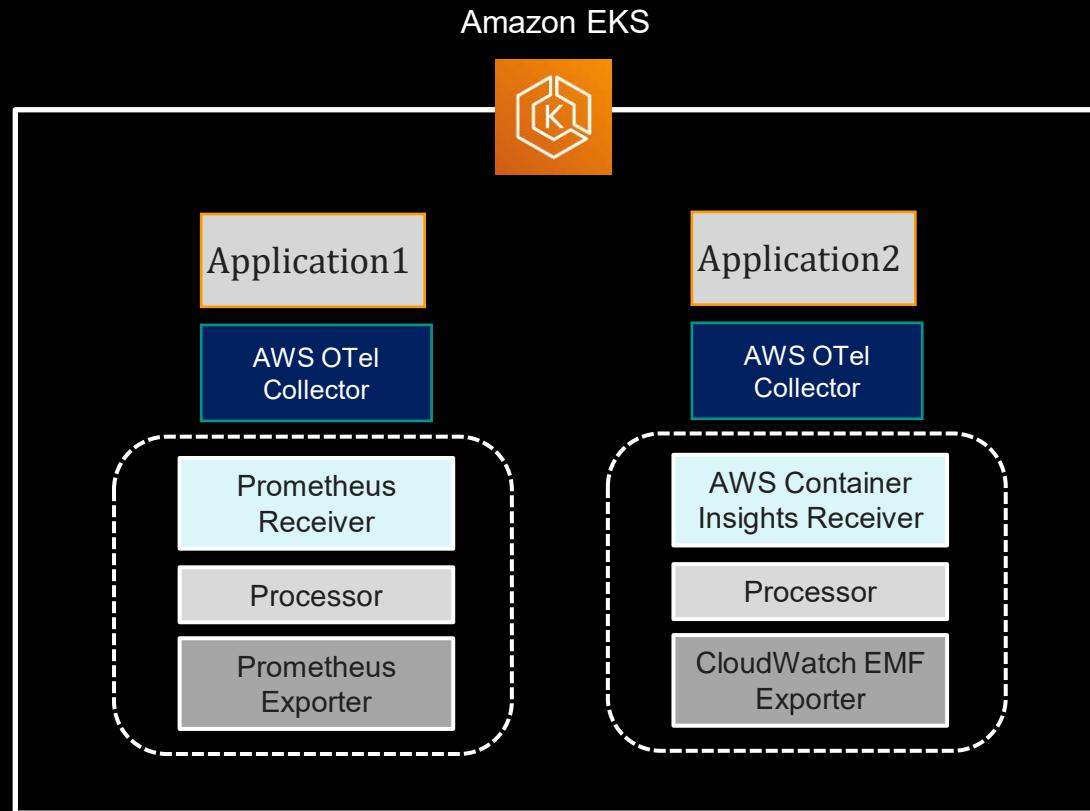
# Prometheus with ADOT



# CloudWatch Container Insights for Prometheus with ADOT



# ADOT Pipelines



# EKS ADVANCED CONCEPTS



Prometheus

# Quiz: What Resources Created?

! nginx-deployment.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: test
6    name: test
7  spec:
8    replicas: 2
9    selector:
10   matchLabels:
11     environment: test
12   template:
13     metadata:
14       labels:
15         environment: test
16     spec:
17       containers:
18         - image: nginx:1.12
19           name: nginx
```



Deployment



Replicaset



Pods

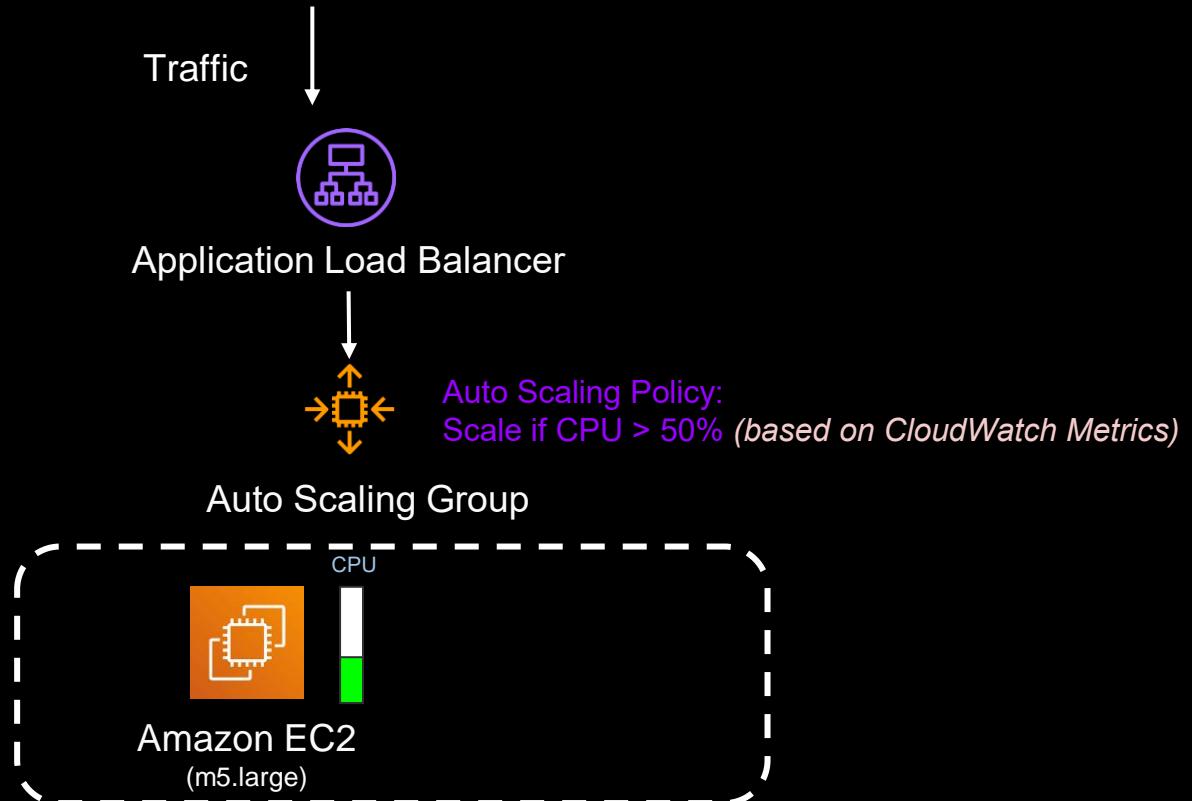
# Container Scaling

- Quick Look into EC2 Scaling
- Container Scaling
- Understand Pod Limits and Requests
- Horizontal Pod Autoscaler
- Understand Manifest File
- HPA Demo

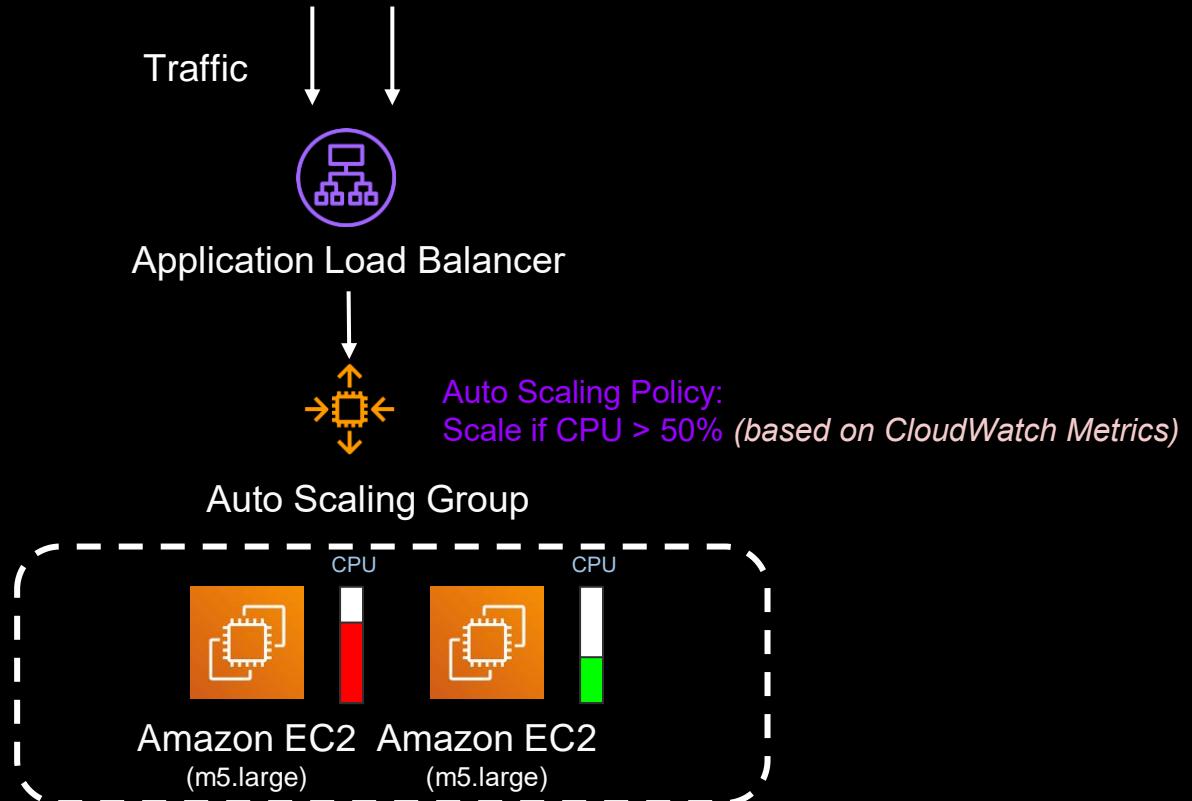
# Horizontal Pod Autoscaler



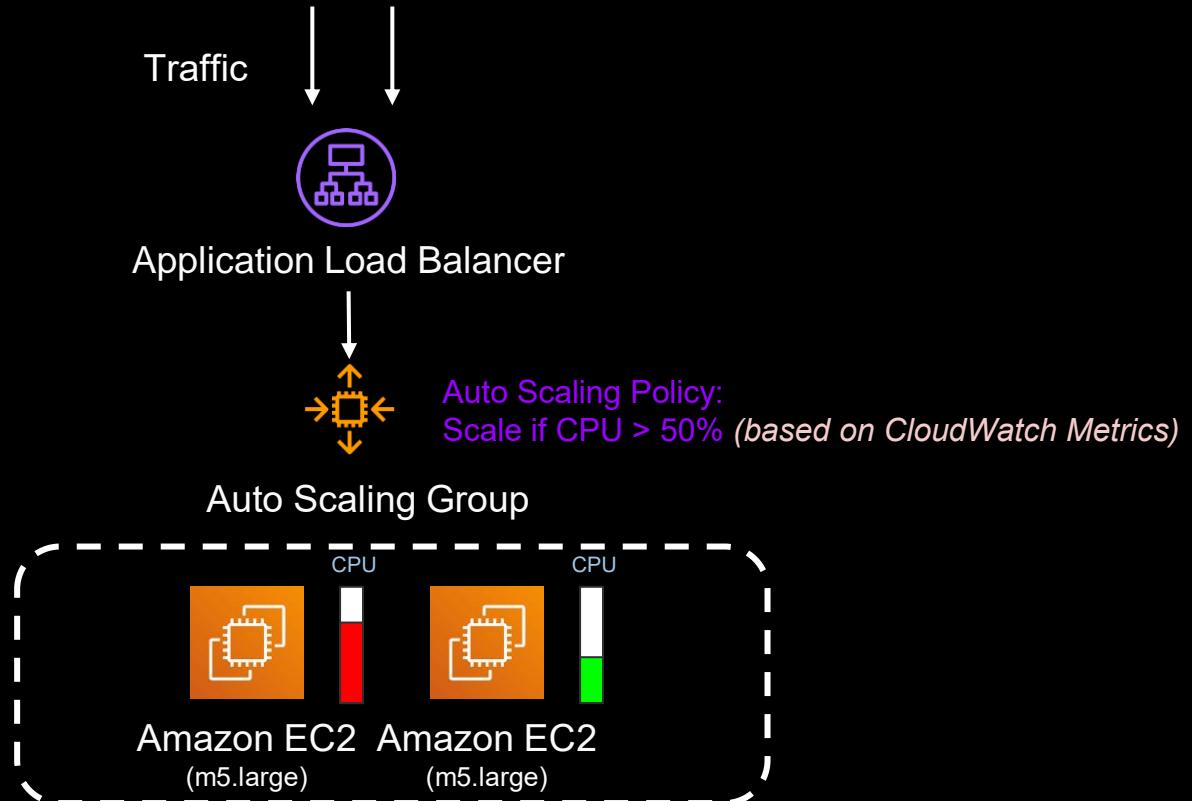
# Going Back to EC2 Scaling



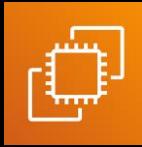
# Going Back to EC2 Scaling



# Going Back to EC2 Scaling

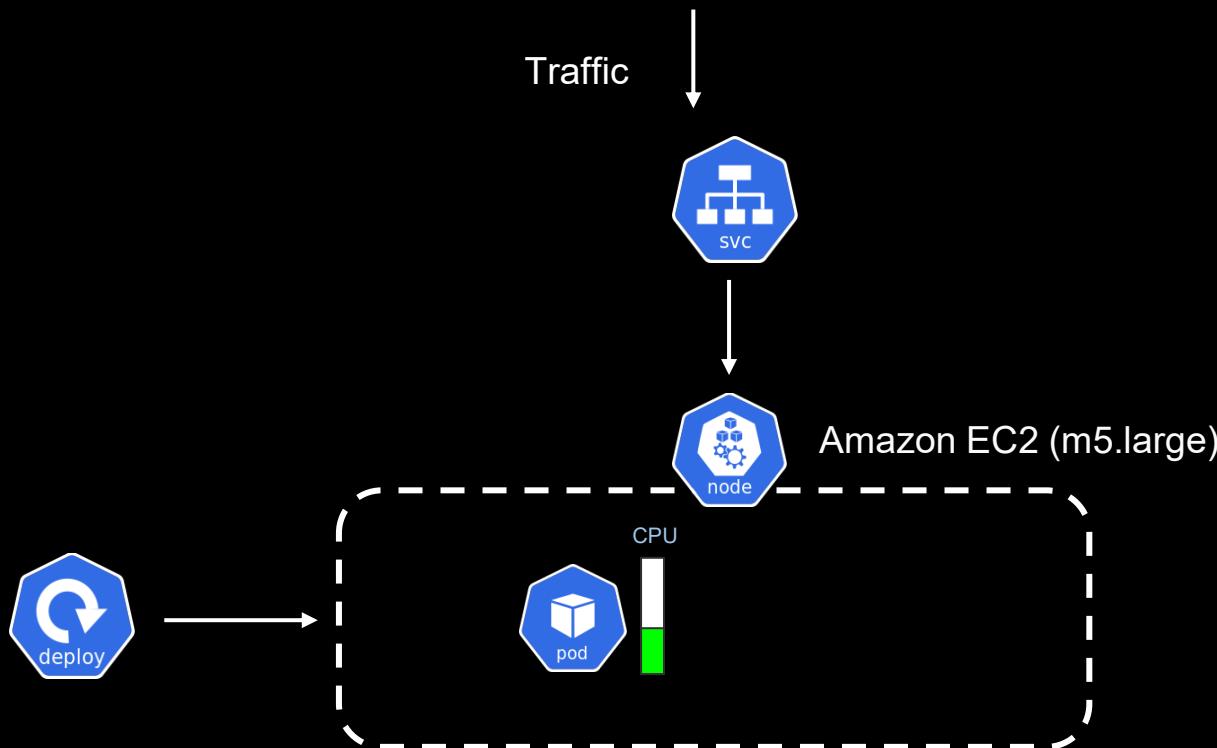


# EKS Container Scaling

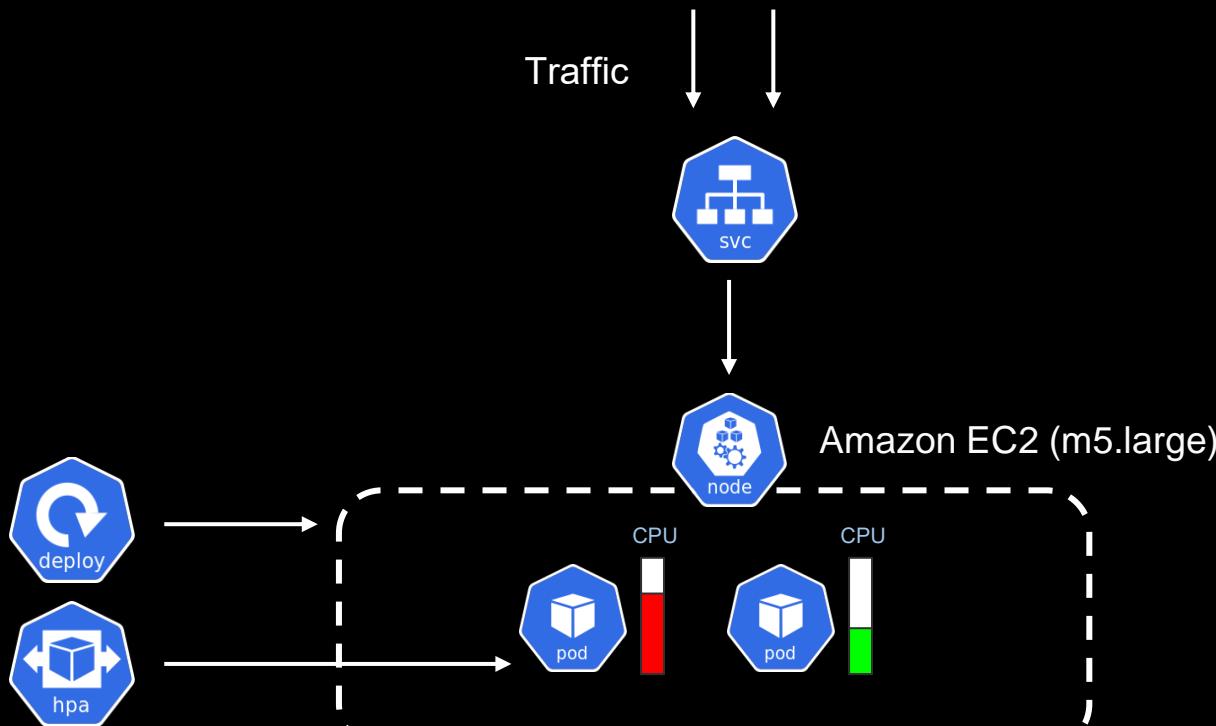


Amazon EC2  
(m5.large)

# Horizontal Pod Autoscaler (HPA)

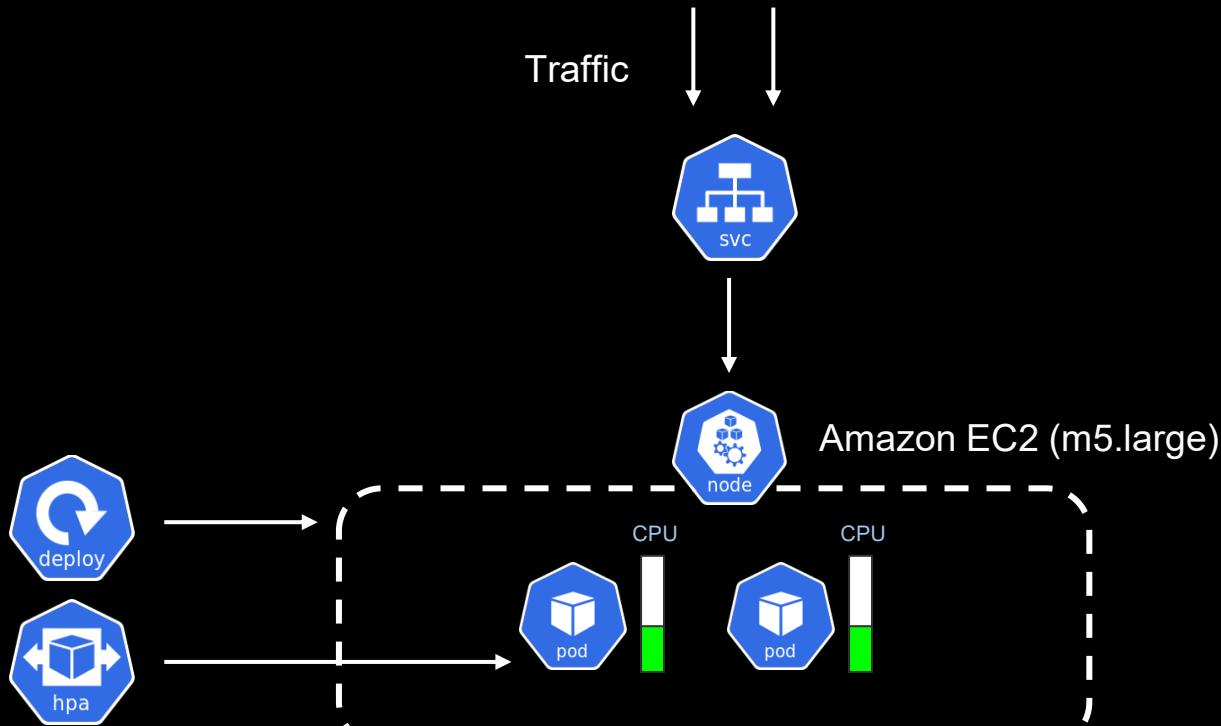


# Horizontal Pod Autoscaler (HPA)



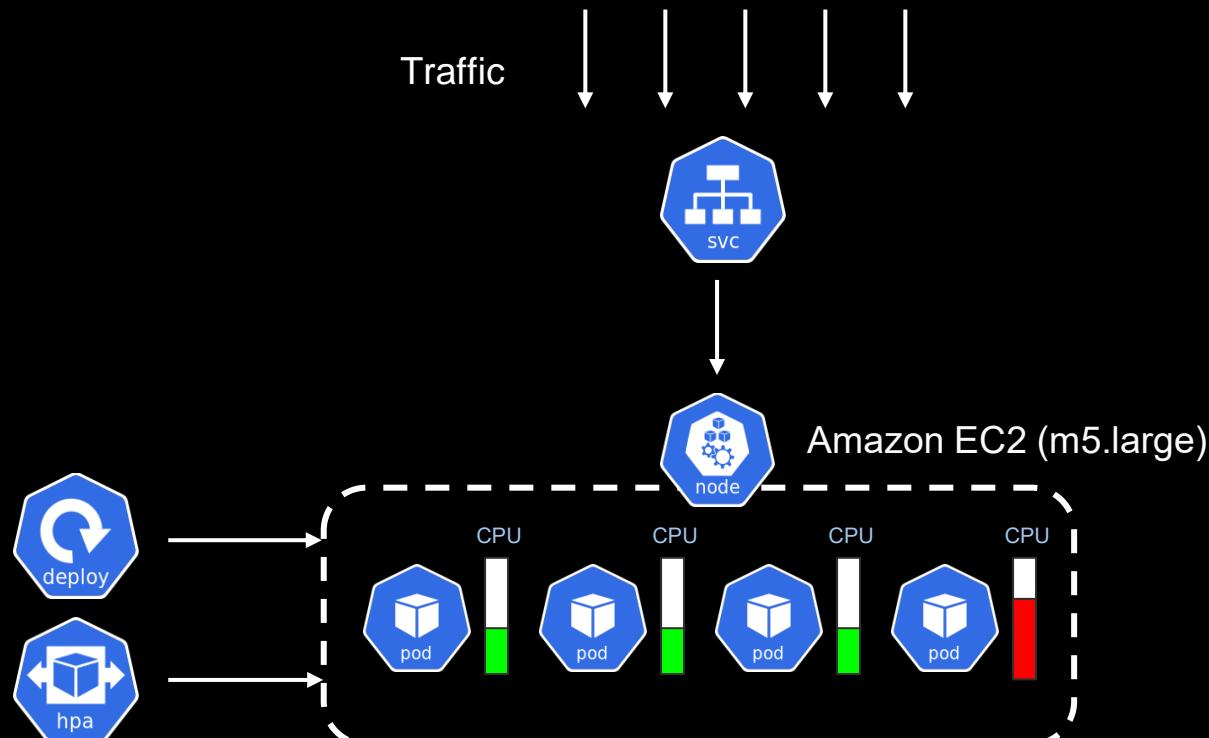
Scale pod if pod CPU >  
50%

# Horizontal Pod Autoscaler (HPA)



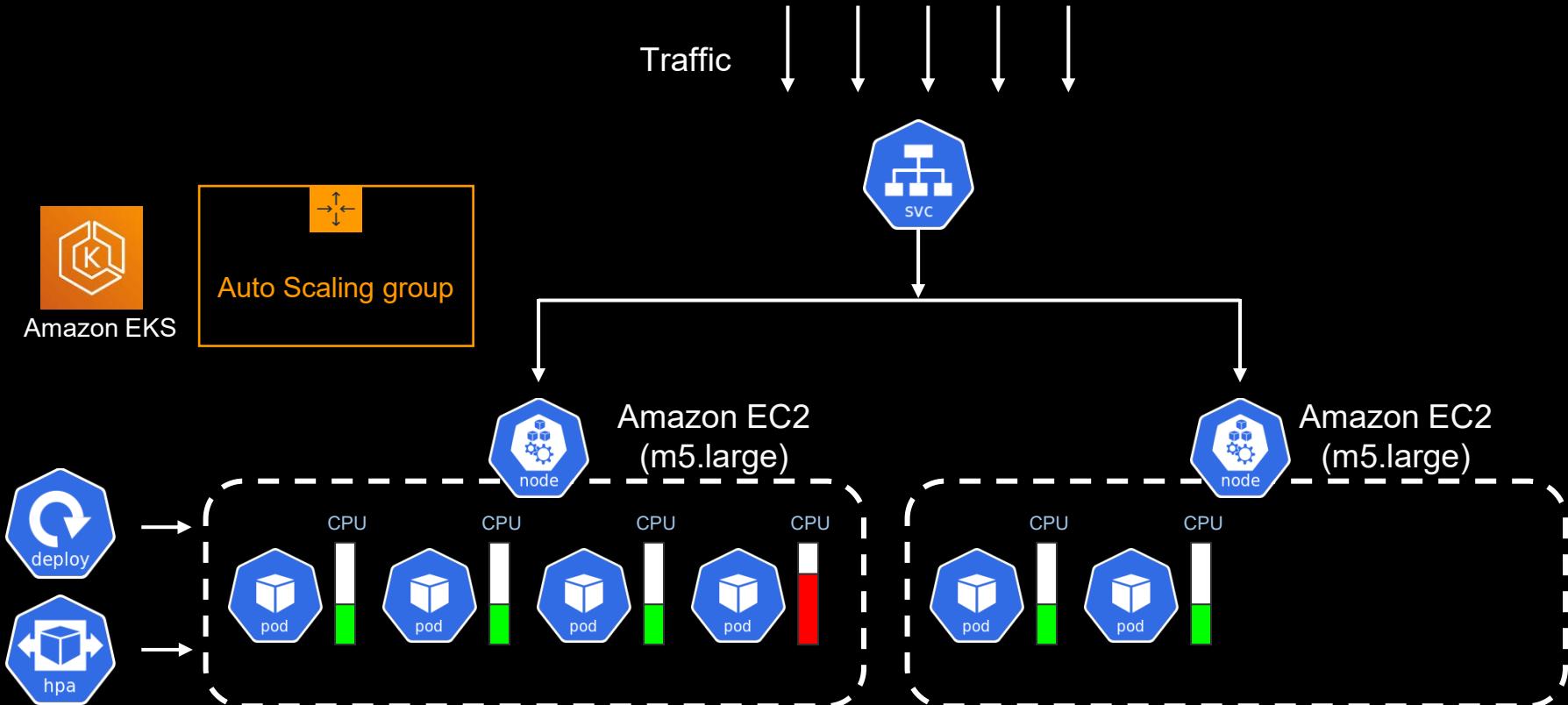
Scale pod if pod CPU > 50%

# Horizontal Pod Autoscaler (HPA)

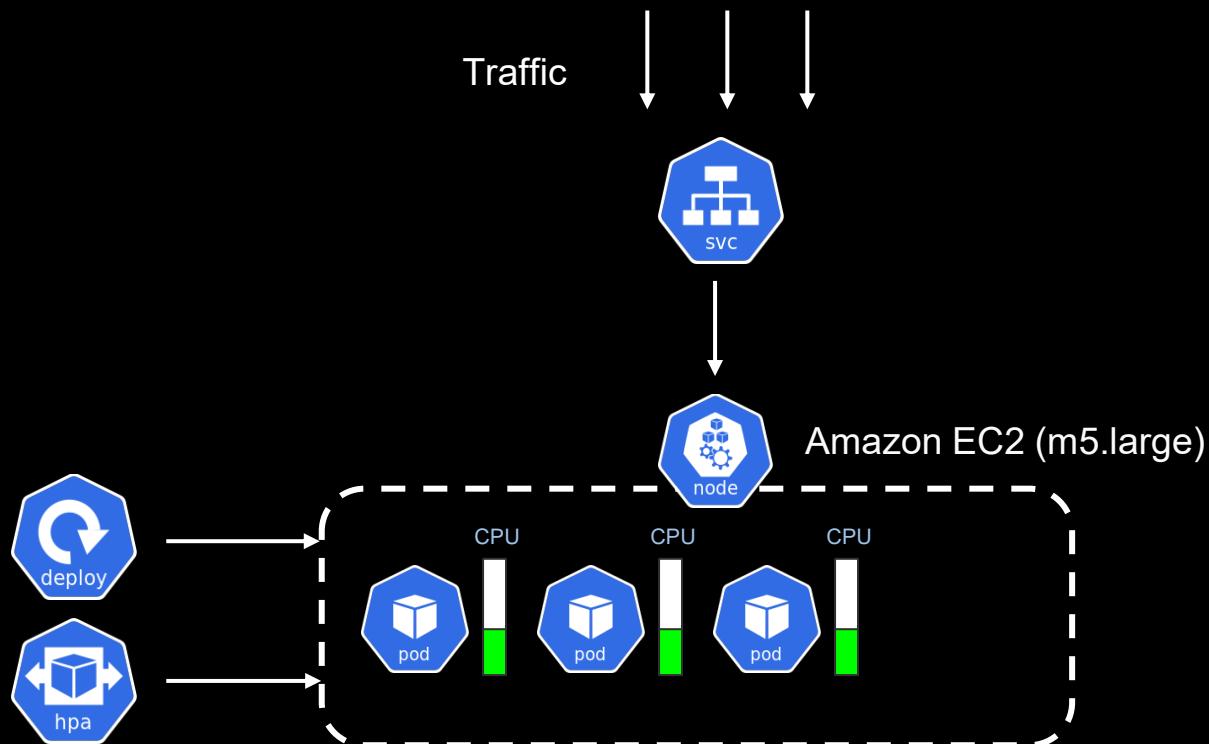


Scale pod if pod CPU > 50%

# EKS Cluster Autoscaler



# Horizontal Pod Autoscaler (HPA)



# Pod Request And Limit

! hpa-php-apache.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment ----->
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9    replicas: 1 ----->
10   template:
11     metadata:
12       labels:
13         run: php-apache
14     spec:
15       containers: ----->
16         - name: php-apache
17           image: k8s.gcr.io/hpa-example
18           ports:
19             - containerPort: 80
20               resources:
21                 requests:
22                   cpu: 500m
23                   limits:
24                     cpu: 1000m
```



# Pod Request And Limit

hpa-php-apache.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9  replicas: 1
10 template:
11   metadata:
12     labels:
13       run: php-apache
14   spec:
15     containers:
16       - name: php-apache
17         image: k8s.gcr.io/hpa-example
18         ports:
19           - containerPort: 80
20         resources:
21           requests:
22             cpu: 500m
23           limits:
24             cpu: 1000m
```



Amazon EC2  
(m5.large)



Instance Size	vCPU	Memory (GiB)
m5.large	2	8

1 vCPU = 1000m (milicore)

→ One pod requesting CPU of 500m (Half of 1 vCPU)

→ One pod CPU limit 1000m (1 vCPU)

# Pod Request And Limit

hpa-php-apache.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9  replicas: 1
10 template:
11   metadata:
12     labels:
13       run: php-apache
14   spec:
15     containers:
16       - name: php-apache
17         image: k8s.gcr.io/hpa-example
18         ports:
19           - containerPort: 80
20         resources:
21           requests:
22             cpu: 500m
23           limits:
24             cpu: 1000m
```



! hpa-php-apache.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4  name: php-apache
5  spec:
6  selector:
7  matchLabels:
8  run: php-apache
9  replicas: 1
10 template:
11   metadata:
12     labels:
13       run: php-apache
14   spec:
15     containers:
16       - name: php-apache
17         image: k8s.gcr.io/hpa-example
18         ports:
19           - containerPort: 80
20         resources:
21           requests:
22             cpu: 0.5
23           limits:
24             cpu: 1
```

# Pod Request And Limit

! hpa-cpu-and-memory-php-apache.yaml

```
1 apiVersion: apps/v1
2 kind: Deployment
3 metadata:
4   name: php-apache
5 spec:
6   selector:
7     matchLabels:
8       run: php-apache
9   replicas: 1
10  template:
11    metadata:
12      labels:
13        run: php-apache
14    spec:
15      containers:
16        - name: php-apache
17          image: k8s.gcr.io/hpa-example
18          ports:
19            - containerPort: 80
20          resources:
21            requests:
22              cpu: 500m
23              memory: 256Mi
24            limits:
25              cpu: 1000m
26              memory: 512Mi
```



Amazon EC2  
(m5.large)

Instance Size	vCPU	Memory (GiB)
m5.large	2	8

1 vCPU = 1000m (milicore)

→ One pod requesting CPU of 500m (Half of 1 vCPU) and 256 MiB of Memory

→ One pod CPU limit 1000m (1 vCPU) and 512 MiB of Memory

# Pod Request And Limit + HPA

hpa-php-apache.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9    replicas: 1
10   template:
11     metadata:
12       labels:
13         run: php-apache
14     spec:
15       containers:
16         - name: php-apache
17           image: k8s.gcr.io/hpa-example
18           ports:
19             - containerPort: 80
20           resources:
21             requests:
22               cpu: 500m
23             limits:
24               cpu: 1000m
```



Create Replica of Pod  
If pod CPU > 50% of  
request CPU  
(If pod CPU exceeds  
250m)

```
42  apiVersion: autoscaling/v1
43  kind: HorizontalPodAutoscaler
44  metadata:
45    name: php-apache
46    namespace: default
47  spec:
48    scaleTargetRef:
49      apiVersion: apps/v1
50      kind: Deployment
51      name: php-apache
52    minReplicas: 1
53    maxReplicas: 10
54    targetCPUUtilizationPercentage: 50
55
```

# K8s HPA YAML + DEMO



! hpa-php-apache.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9    replicas: 1
10   template:
11     metadata:
12       labels:
13         run: php-apache
14     spec:
15       containers:
16         - name: php-apache
17           image: k8s.gcr.io/hpa-example
18           ports:
19             - containerPort: 80
20           resources:
21             requests:
22               cpu: 500m
23             limits:
24               cpu: 1000m
```

```
42  apiVersion: autoscaling/v1
43  kind: HorizontalPodAutoscaler
44  metadata:
45    name: php-apache
46    namespace: default
47  spec:
48    scaleTargetRef:
49      apiVersion: apps/v1
50      kind: Deployment
51      name: php-apache
52    minReplicas: 1
53    maxReplicas: 10
54    targetCPUUtilizationPercentage: 50
55
```

# Pod Request And Limit + HPA

hpa-php-apache.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9    replicas: 10
10   template:
11     metadata:
12       labels:
13         run: php-apache
14     spec:
15       containers:
16         - name: php-apache
17           image: k8s.gcr.io/hpa-example
18           ports:
19             - containerPort: 80
20           resources:
21             requests:
22               cpu: 500m
23             limits:
24               cpu: 1000m
```



Create Replica of Pod  
If pod CPU > 50% of  
request CPU  
(If pod CPU exceeds  
250m)

```
42  apiVersion: autoscaling/v1
43  kind: HorizontalPodAutoscaler
44  metadata:
45    name: php-apache
46    namespace: default
47  spec:
48    scaleTargetRef:
49      apiVersion: apps/v1
50      kind: Deployment
51      name: php-apache
52    minReplicas: 1
53    maxReplicas: 10
54    targetCPUUtilizationPercentage: 50
55
```

# HPA Demo on EKS

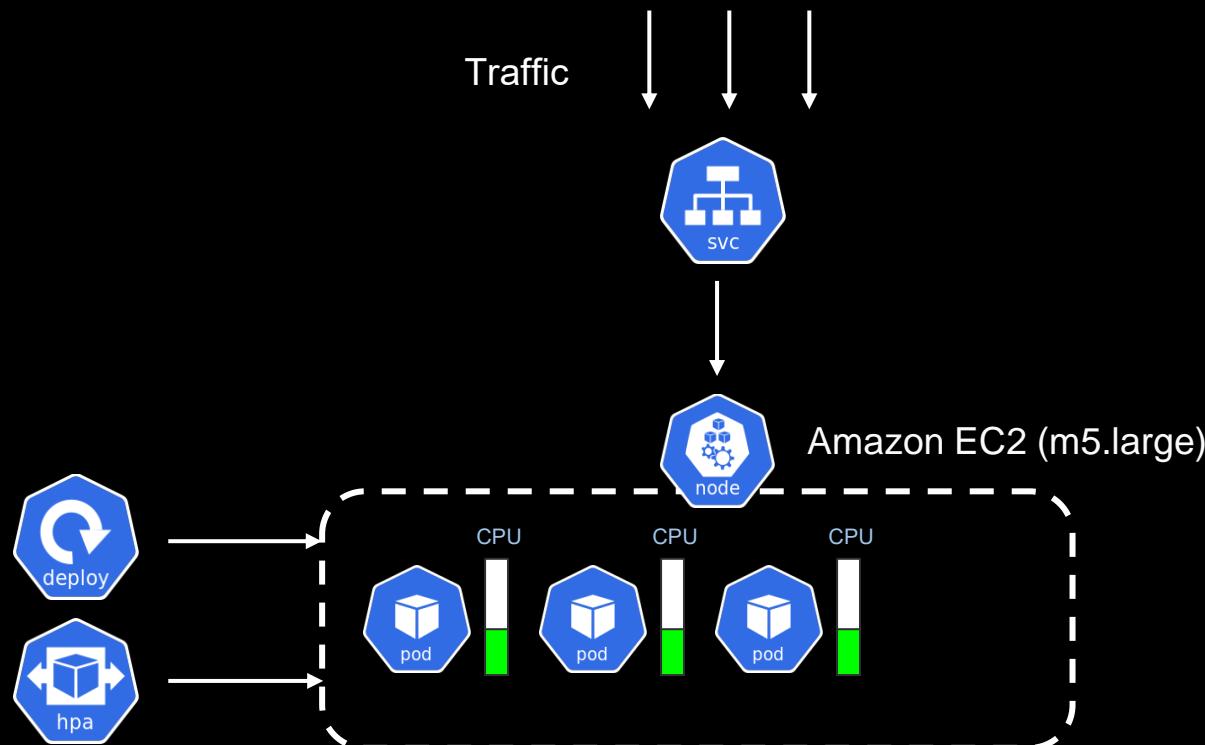
- Install Metrics Server
- Deploy Deployment, Service, HPA
- Increase Load
- Pod scale!

*Note - Need node larger than t3.micro*

<https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale-walkthrough/>

# Cluster Autoscaler

# Horizontal Pod Autoscaler (HPA)



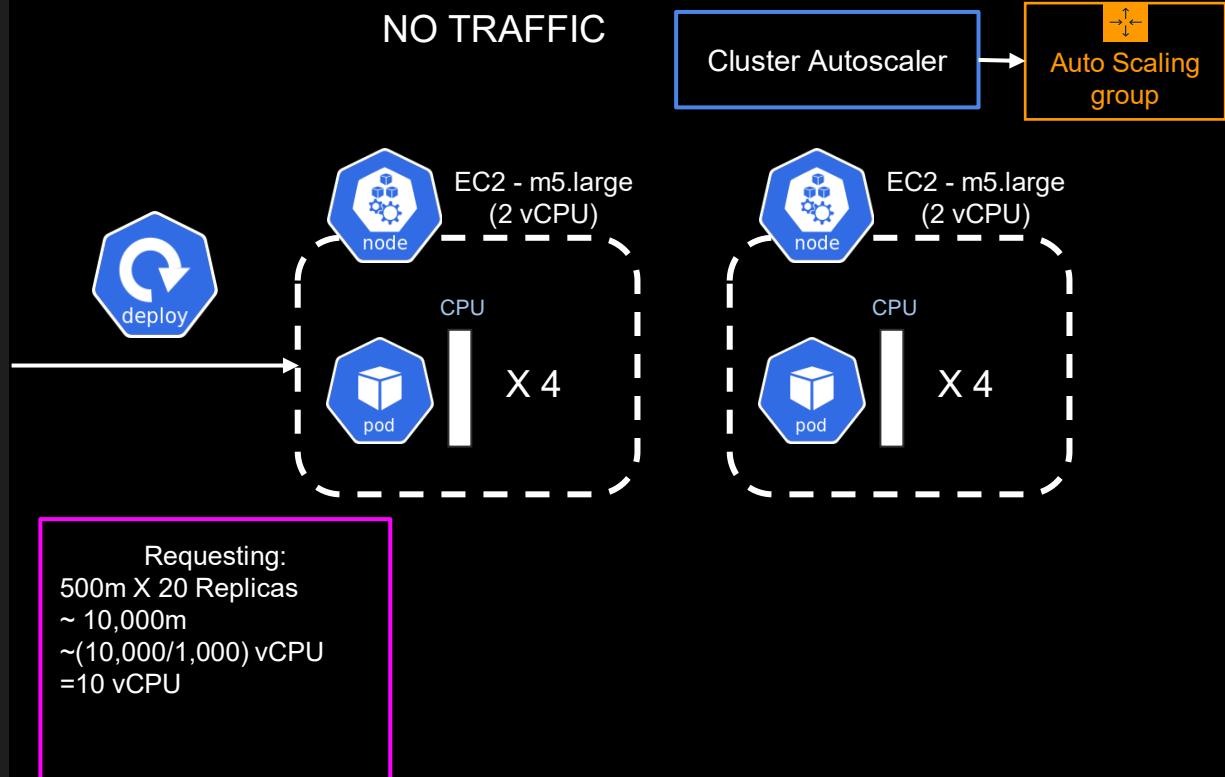
# Cluster Autoscaler



Amazon EKS

! cluster-autoscaler-deployment-1.yaml

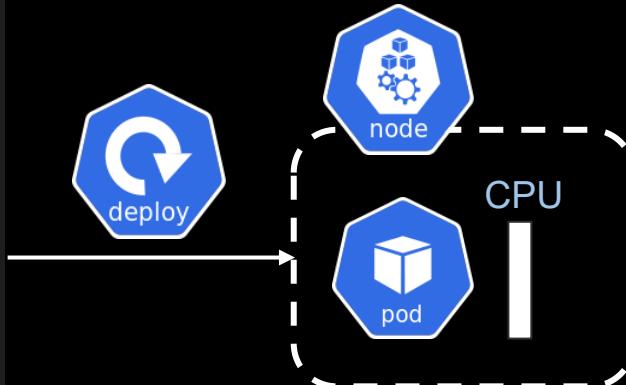
```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9    replicas: 20
10   template:
11     metadata:
12       labels:
13         run: php-apache
14     spec:
15       containers:
16         - name: php-apache
17           image: k8s.gcr.io/hpa-example
18           ports:
19             - containerPort: 80
20           resources:
21             requests:
22               cpu: 500m
23             limits:
24               cpu: 1000m
```



# CLUSTER AUTOSCALER

! cluster-autoscaler-deployment-1.yaml

```
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    name: php-apache
5  spec:
6    selector:
7      matchLabels:
8        run: php-apache
9    replicas: 20
10   template:
11     metadata:
12       labels:
13         run: php-apache
14     spec:
15       containers:
16         - name: php-apache
17           image: k8s.gcr.io/hpa-example
18           ports:
19             - containerPort: 80
20           resources:
21             requests:
22               cpu: 500m
23             limits:
24               cpu: 1000m
```



# EKS Cluster Autoscaler Demo

- Cluster Autoscaler has two components
  - Open Source Cluster Autoscaler
  - EKS Implementation (ASG, IAM etc.)
- Deploy App
- Nodes scale!

<https://docs.aws.amazon.com/eks/latest/userguide/cluster-autoscaler.html>

+

Extra Steps Required

# Vertical Pod Autoscaler



Do NOT use this in Production!

- Restarts PODs

# Vertical Pod Autoscaler

- Vertical Vs Horizontal Scaling
  - Vertical - Going up in Size
  - Horizontal - Create more of same size
- Pods will go up or down in size (after restart!)
- Used in dev to determine optimal CPU and memory for the Pod
  - VPA recommends pod request, limit
  - Use the numbers for request, limits, HPA
  - VPA should not be used with HPA
- Accept VPA recommendation with grain of salt!

# Vertical Pod Autoscaler Demo

- Practical and Useful Application of VPA



# Vertical Pod Autoscaler Demo

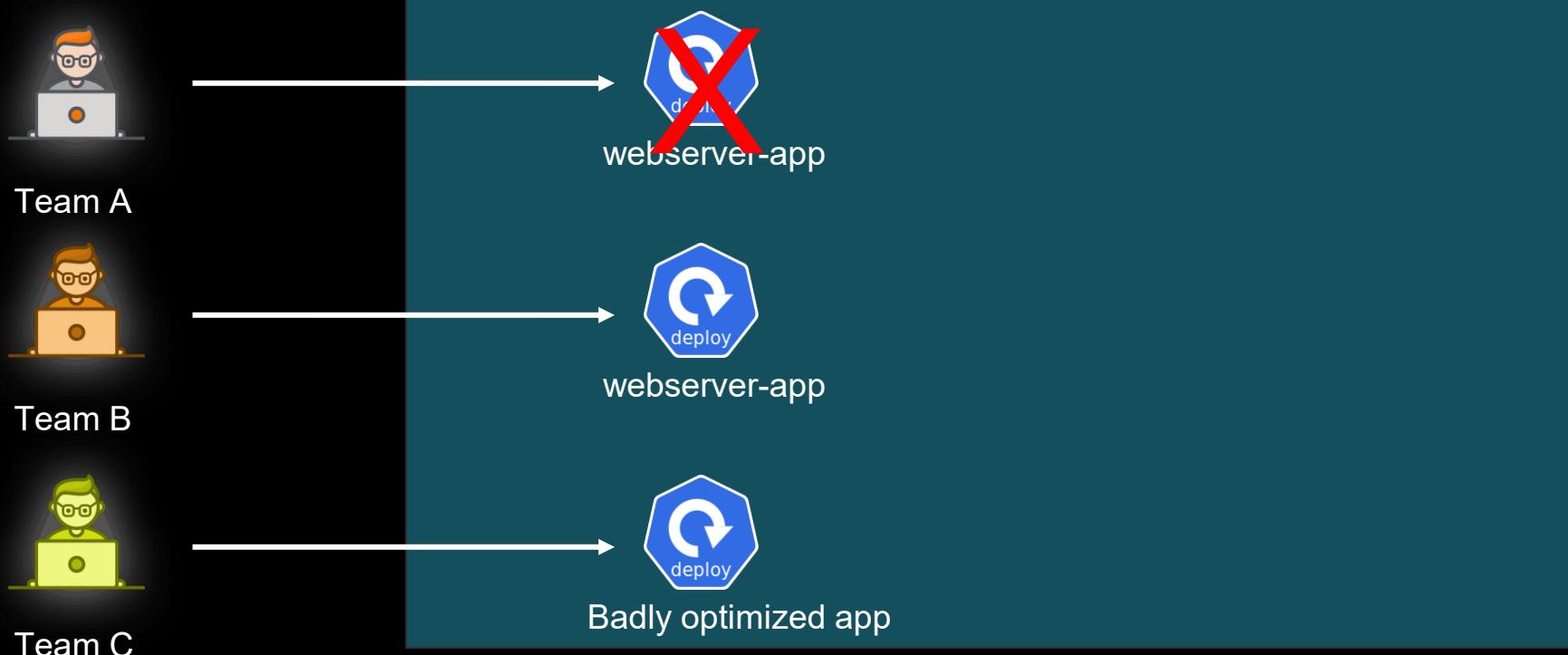
- Run VPA standalone
- Why do we need a tool?
- Demo Goldilocks

# Namespaces

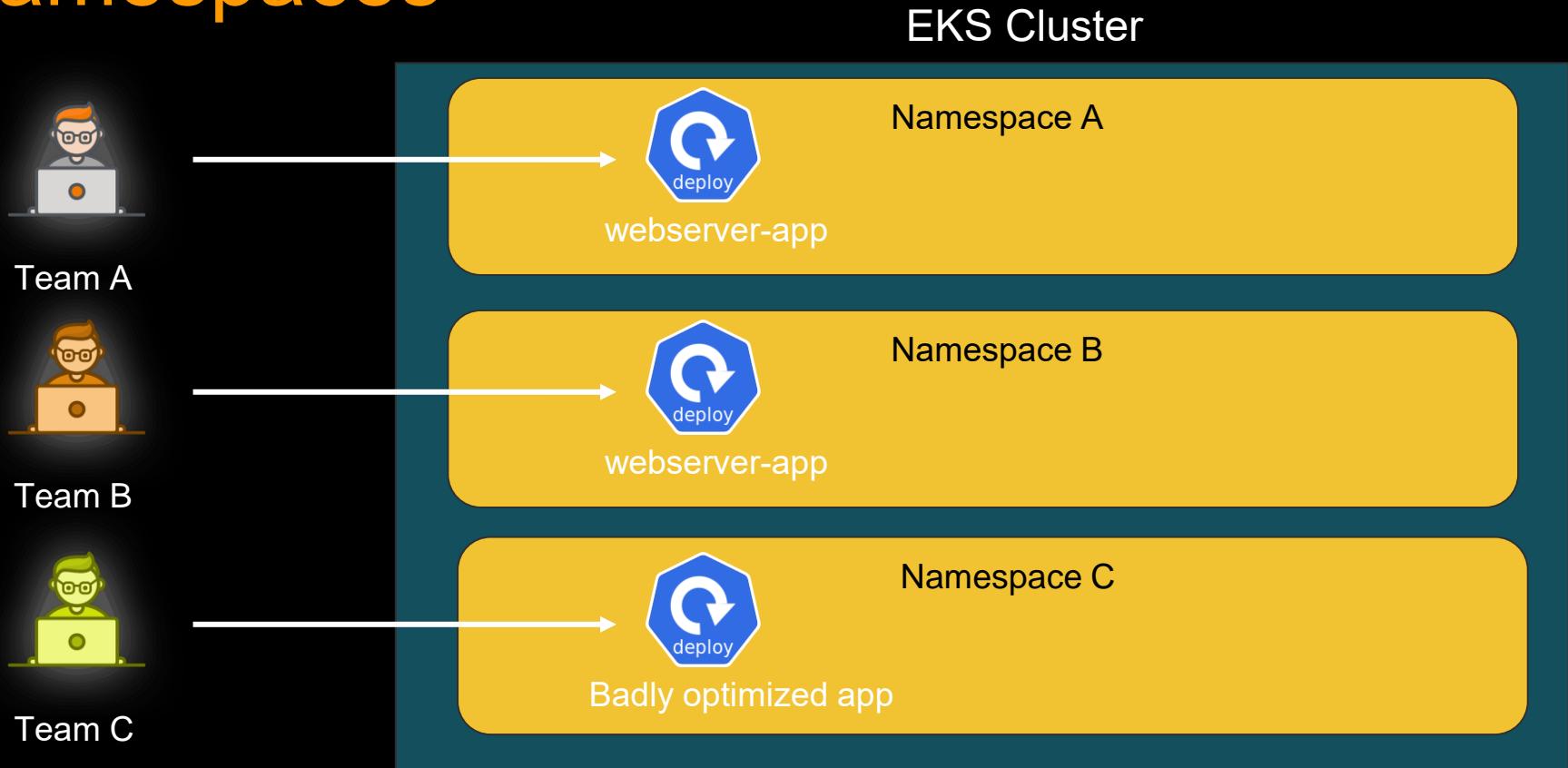


# Namespaces

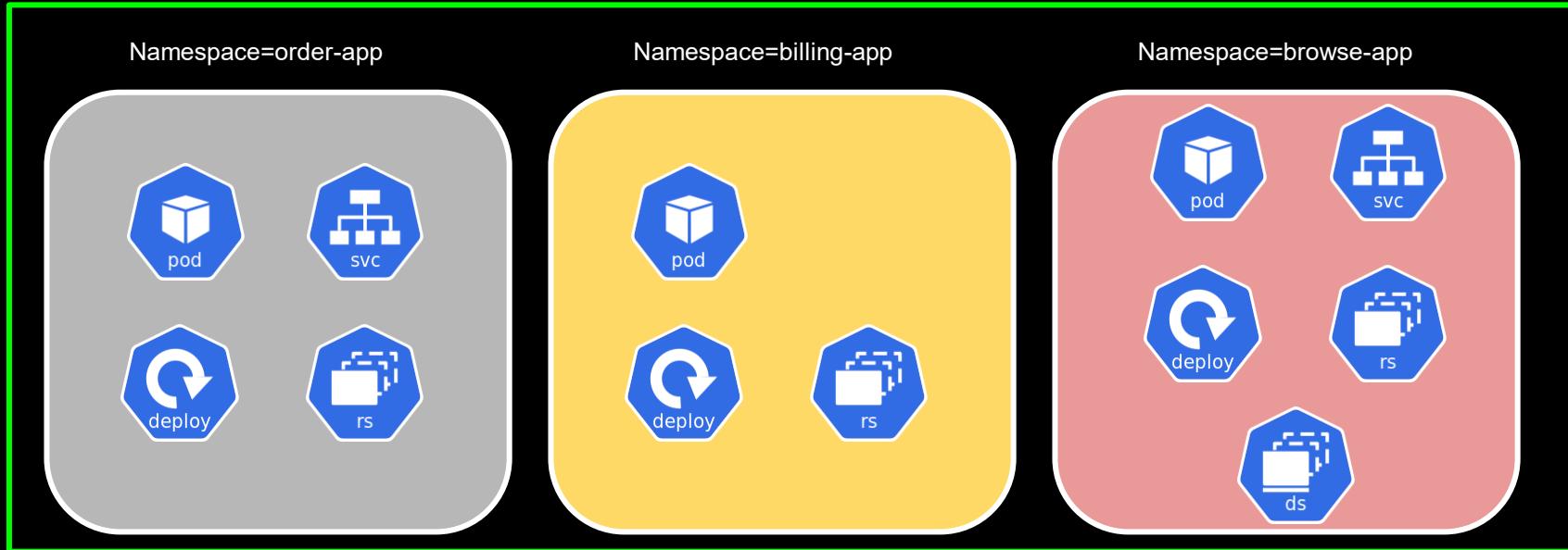
EKS Cluster



# Namespaces



# EKS Cluster Worker Nodes



- Each namespace is a virtual cluster within cluster
- Used for segregate different applications within same cluster
- Can apply separate quota, policies to each namespace
- Namespaces are NOT segregated by nodes

# Namespaces

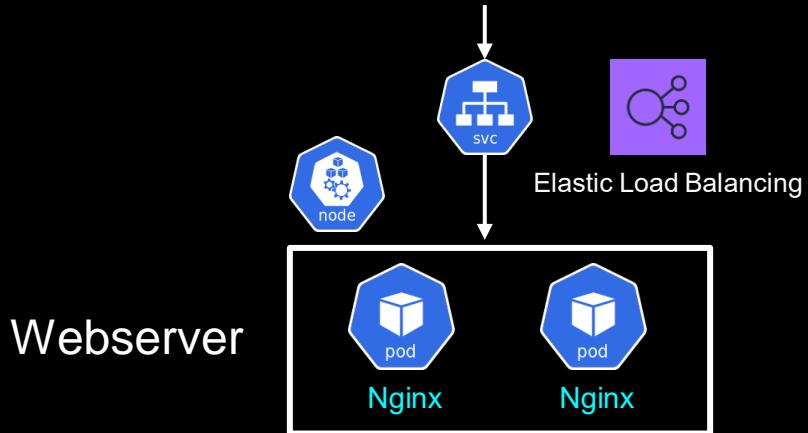
- Each K8s cluster comes with three namespaces
  - kube-public
  - kube-system
  - default (used when no namespace parameter given)
- Let's see this in action!

# Namespaces

- Multiple virtual cluster boundary within one physical cluster
- Provides scope for naming
- Divide cluster resources to namespaces
- Useful for namespace specific accesses

# EKS Ingress

# Ingress - What and Why?



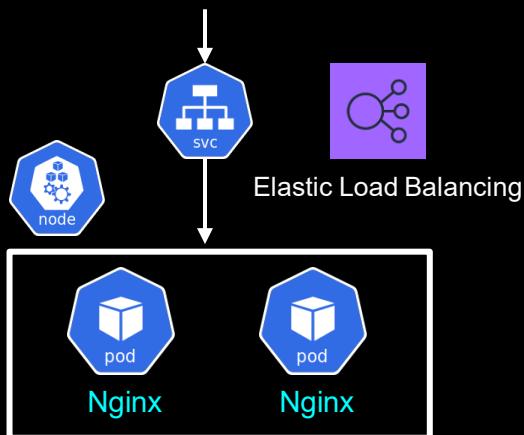
```
! loadbalancer-service.yaml
1 apiVersion: v1
2 kind: Service
3 metadata:
4   name: lb-service
5   labels:
6     app: lb-service
7 spec:
8   type: LoadBalancer
9   ports:
10    - port: 80
11   selector:
12     app: frontend
```

# Ingress - What and Why?

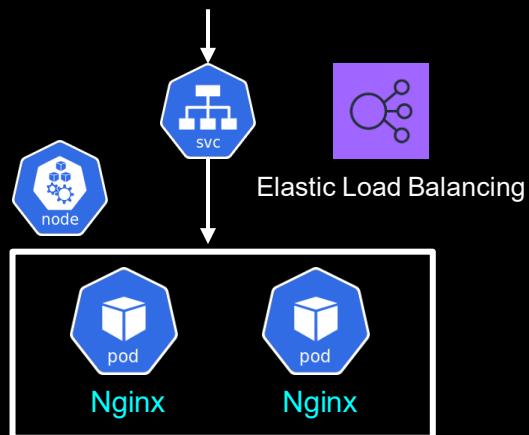


I want to run Bat Cave Systems using Kubernetes

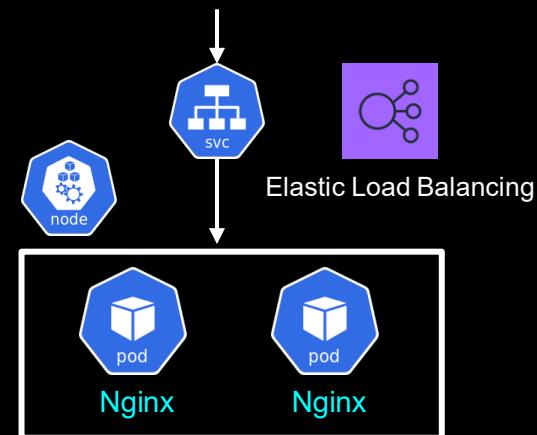
<http://track-joker.com>



<http://monitor-batcave.com>



<http://order-new-batsuit.com>

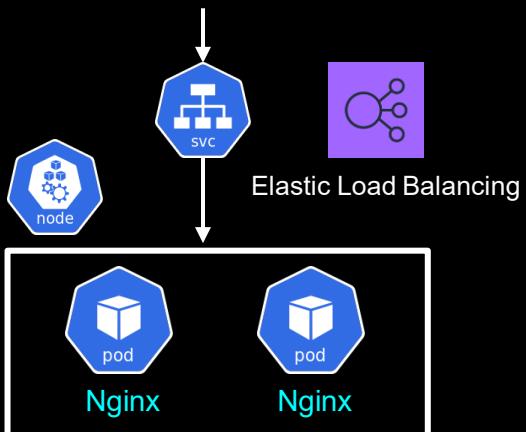


# Ingress - What and Why?

- Costly to maintain one Load Balancer for each service
  - No URL based routing
  - Maintenance overhead of managing all separate services
- “Alfred HELP ME”

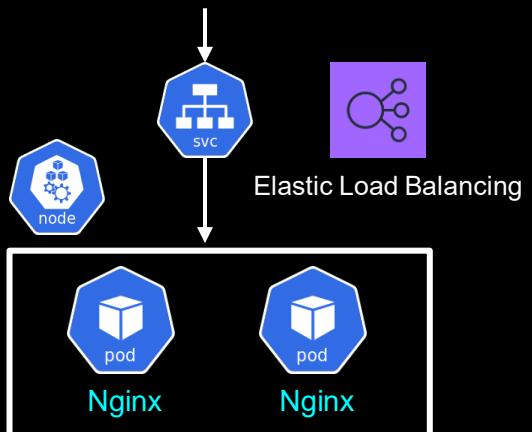


<http://track-joker.com>

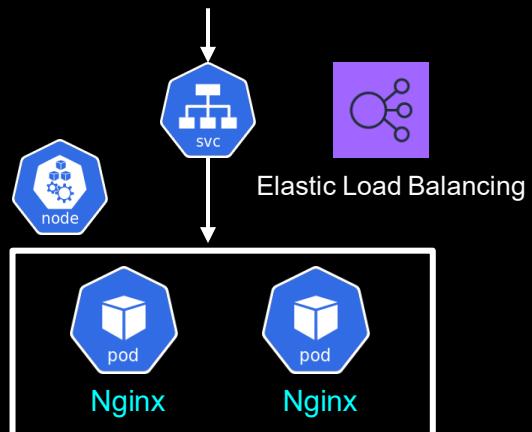


<https://batcave.com/track-joker>  
<https://batcave.com/monitor-batcave>  
<https://batcave.com/order-new-batsuit>

<http://monitor-batcave.com>



<http://order-new-batsuit.com>



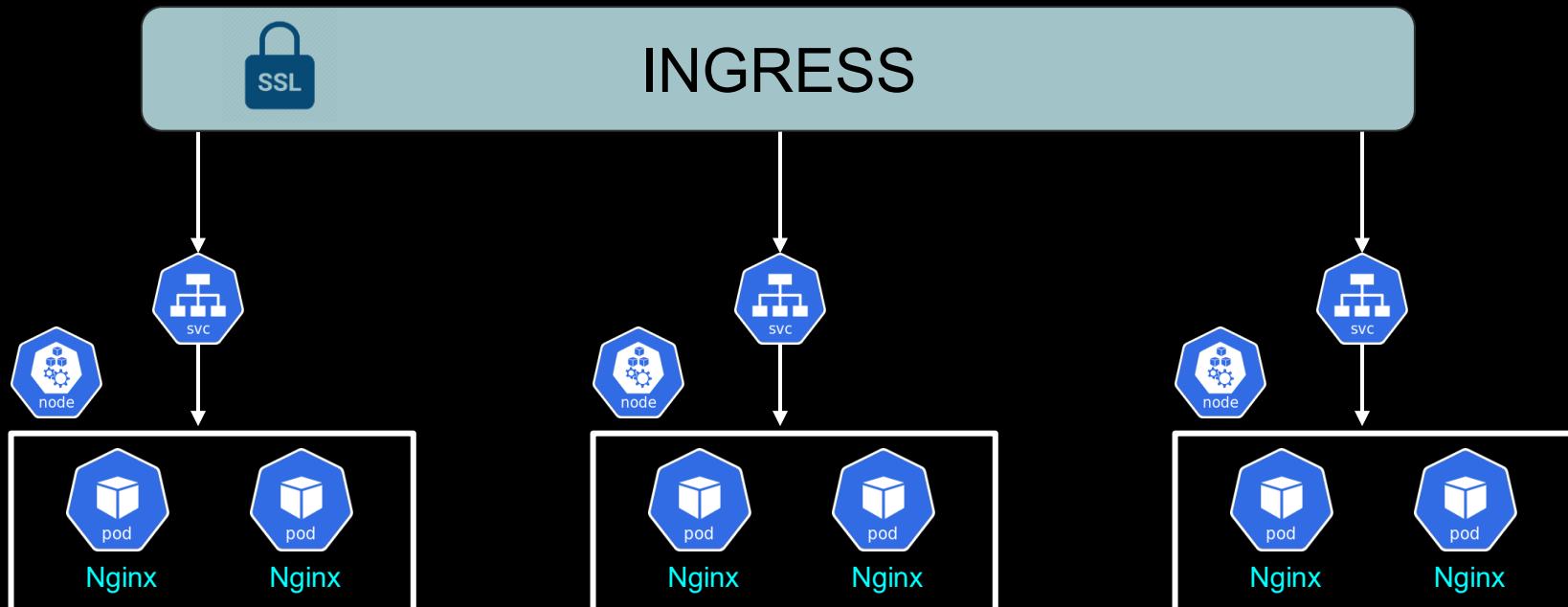
# Ingress - What and Why?

Master Bruce, you need Ingress!

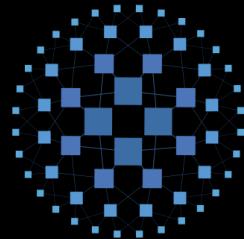


# Ingress - What and Why?

<https://batcave.com/track-joker>  
<https://batcave.com/monitor-batcave>  
<https://batcave.com/order-new-batsuit>



# Ingress Controllers

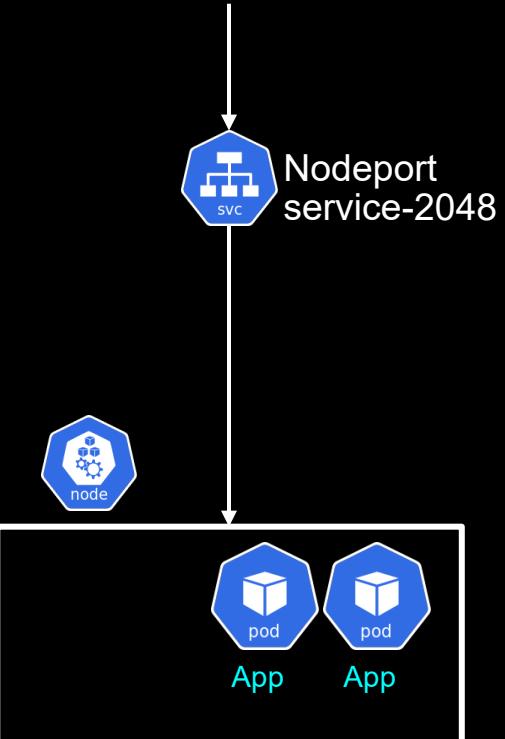


HAPROXY



alb-ingress-controller  
+  
AWS ALB

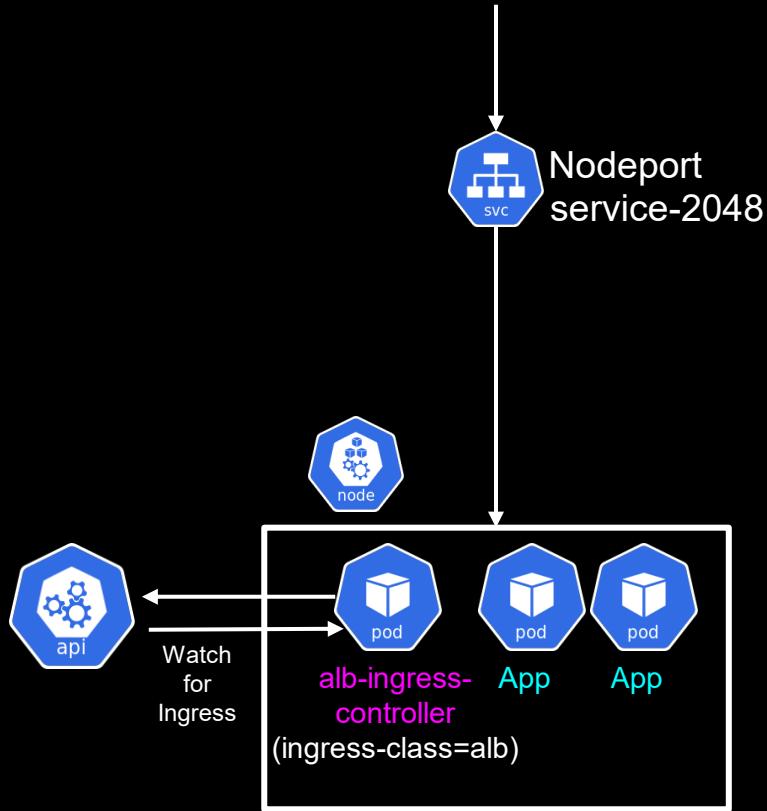
# ALB Ingress



```
apiVersion: v1
kind: Service
metadata:
  name: "service-2048"
  namespace: "2048-game"
spec:
  ports:
    - port: 80
      targetPort: 80
      protocol: TCP
  type: NodePort
  selector:
    app: "2048"
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: "2048-deployment"
  namespace: "2048-game"
spec:
  selector:
    matchLabels:
      app: "2048"
  replicas: 5
  template:
    metadata:
      labels:
        app: "2048"
    spec:
      containers:
        - image: alexwhen/docker-2048
          imagePullPolicy: Always
          name: "2048"
          ports:
            - containerPort: 80
```

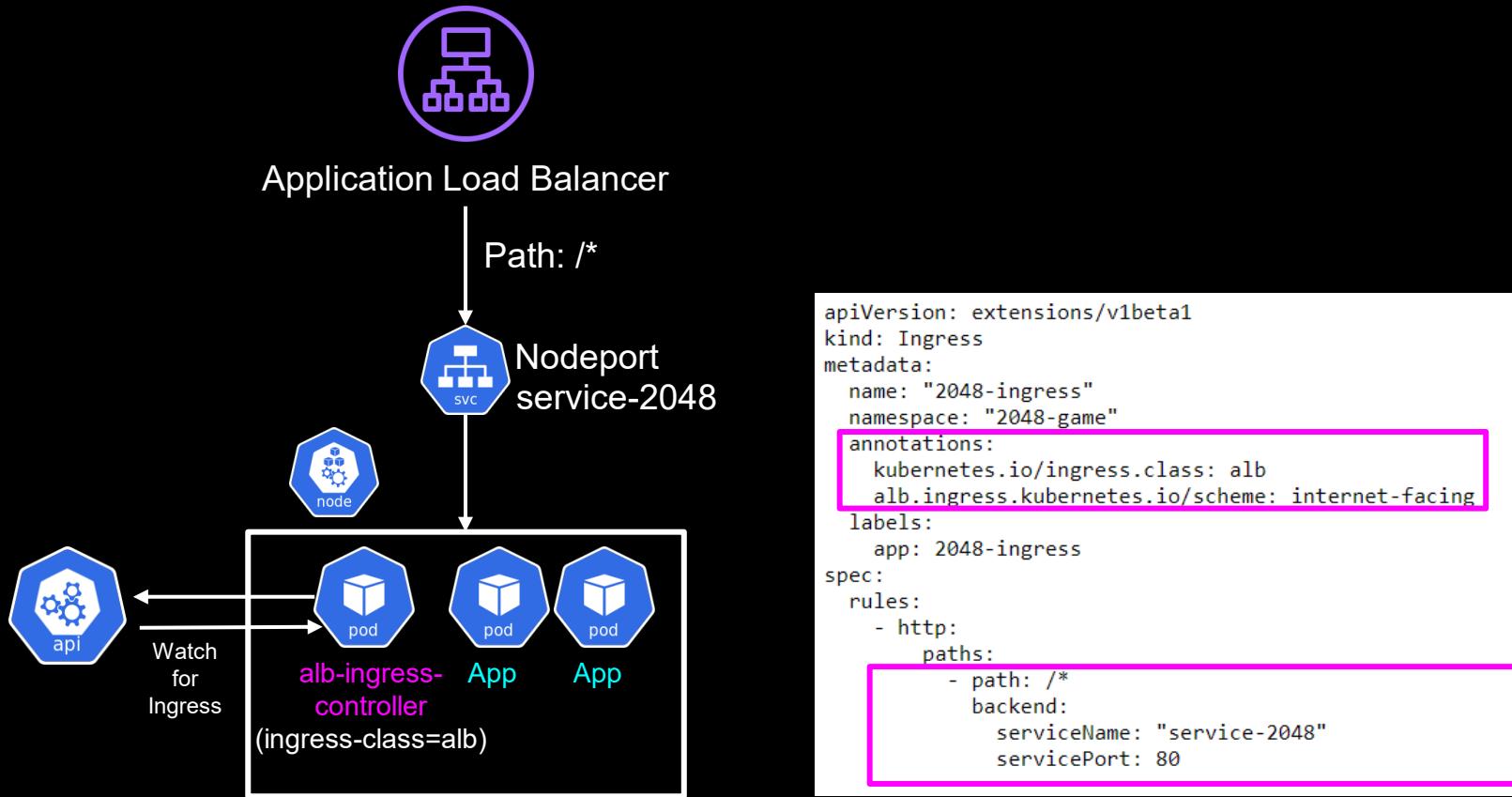
# ALB Ingress Controller



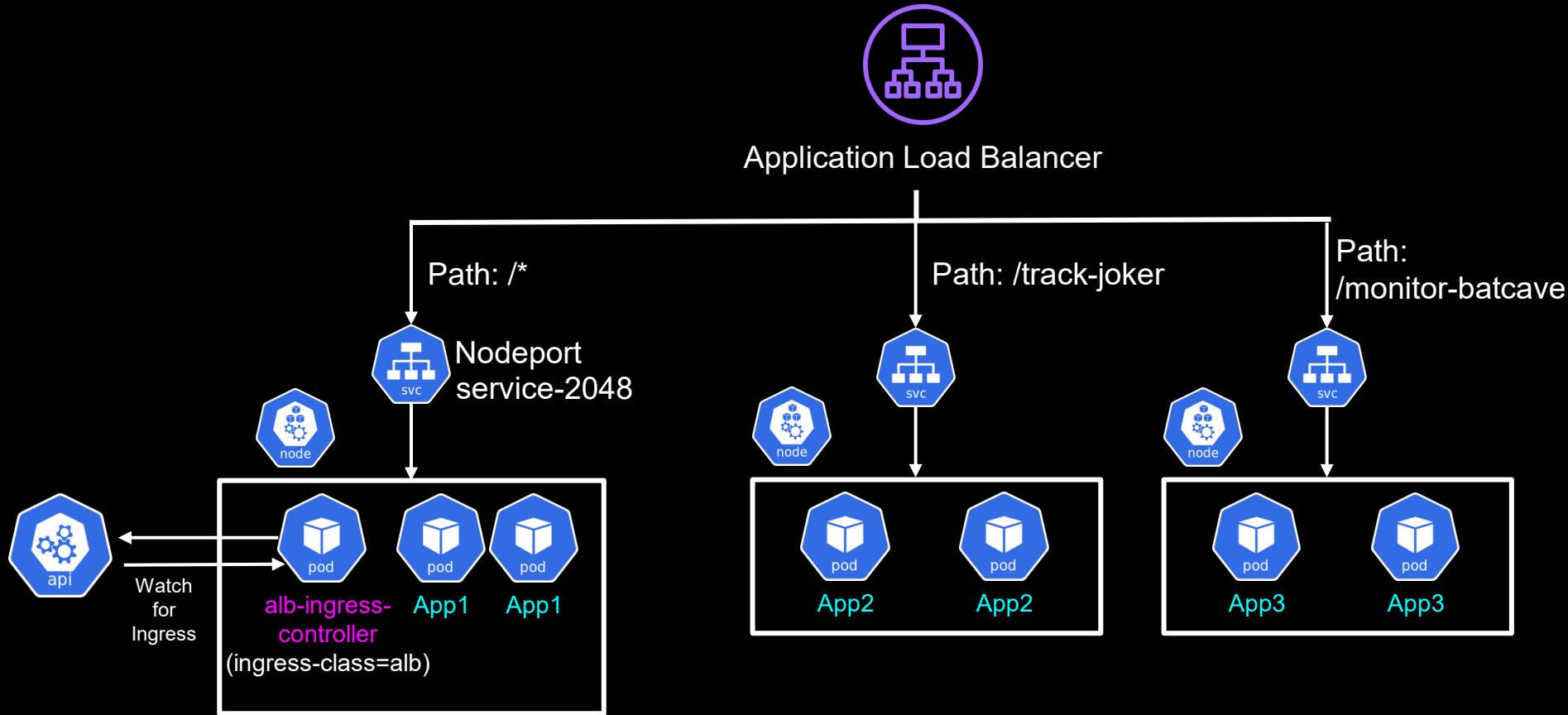
```
# Application Load Balancer (ALB) Ingress Controller Deployment Manifest.
# This manifest details sensible defaults for deploying an ALB Ingress Controller.
# GitHub: https://github.com/kubernetes-sigs/aws-alb-ingress-controller
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
  # Namespace the ALB Ingress Controller should run in. Does not impact which
  # namespaces it's able to resolve ingress resource for. For limiting ingress
  # namespace scope, see --watch-namespace.
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: alb-ingress-controller
  template:
    metadata:
      labels:
        app.kubernetes.io/name: alb-ingress-controller
    spec:
      containers:
        - name: alb-ingress-controller
      args:
        # Limit the namespace where this ALB Ingress Controller deployment will
        # resolve ingress resources. If left commented, all namespaces are used.
        # - --watch-namespace=your-k8s-namespace

        # Setting the ingress-class flag below ensures that only ingress resources with the
        # annotation kubernetes.io/ingress.class: "alb" are respected by the controller. You may
        # choose any class you'd like for this controller to respect.
        - --ingress-class=alb
```

# Ingress resource



# ALB Ingress



# Ingress - What and Why?

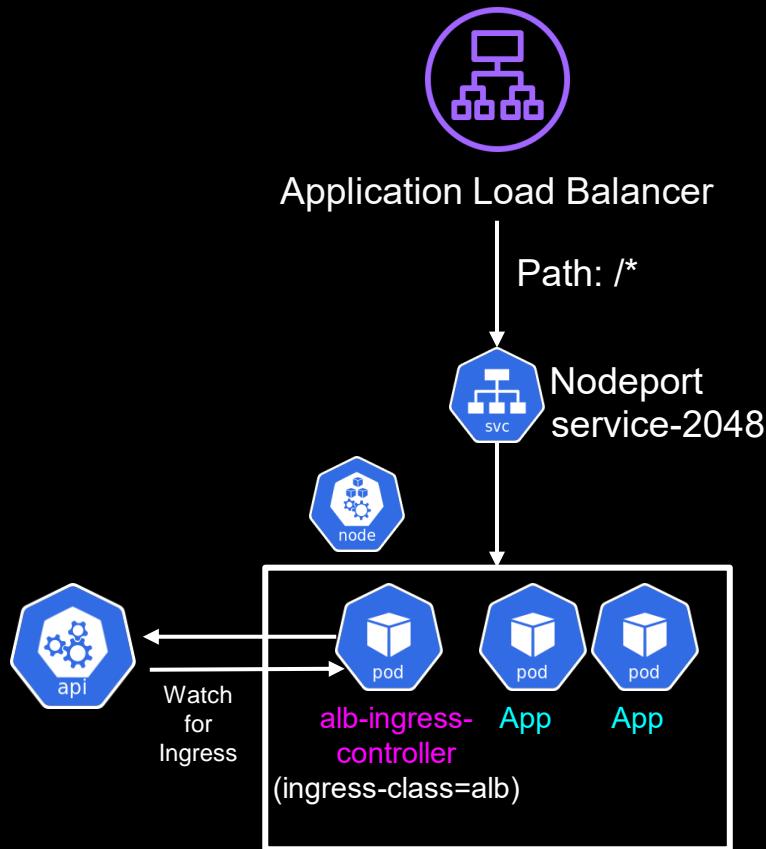
## INGRESS CONTROLLER

- Monitors Ingress resources
- Creates necessary AWS resources for Ingress
  - Such as ALB for ALB Ingress Controller
- One Cluster can have more than one Ingress Controller!
  - Ingress Resource defines which Ingress Controller to use

## INGRESS RESOURCE

- Selects which Ingress Controller to use
- Defines the URL Path and corresponding backend Service

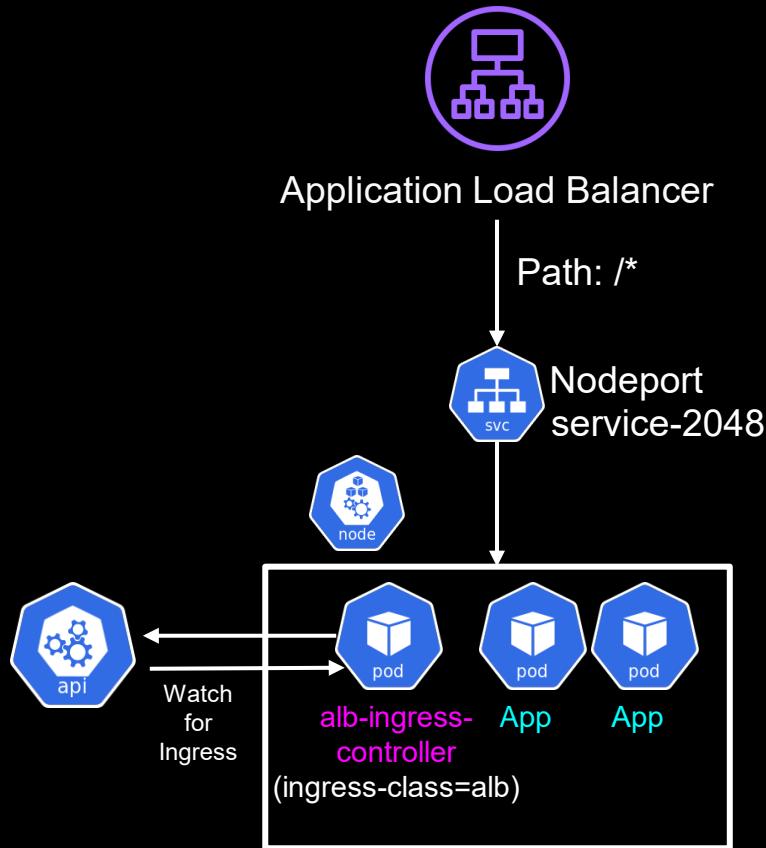
# ALB Ingress



## INGRESS CONTROLLER

- Creates ALB and AWS Resources
  - Directed by Ingress Resource
- Require proper IAM Policy
- Require Service Account and IAM Role for the alb-ingress-controller pod

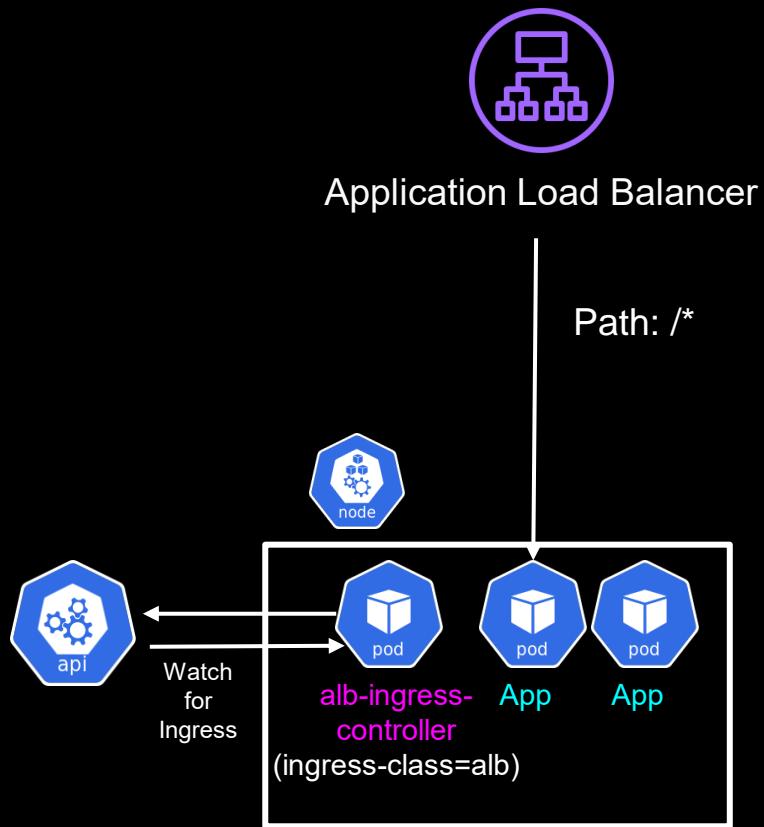
# ALB Ingress



## INGRESS TRAFFIC

- **instance mode**
  - ALB to Nodeport to pods
- **ip mode**
  - ALB to pods directly
  - Require secondary IP address on ENI as pod IP for networking plugin (AWS CNI plugin for kubernetes)
  - Saves one hop
  - alb-ingress-controller specific feature

# ALB Ingress

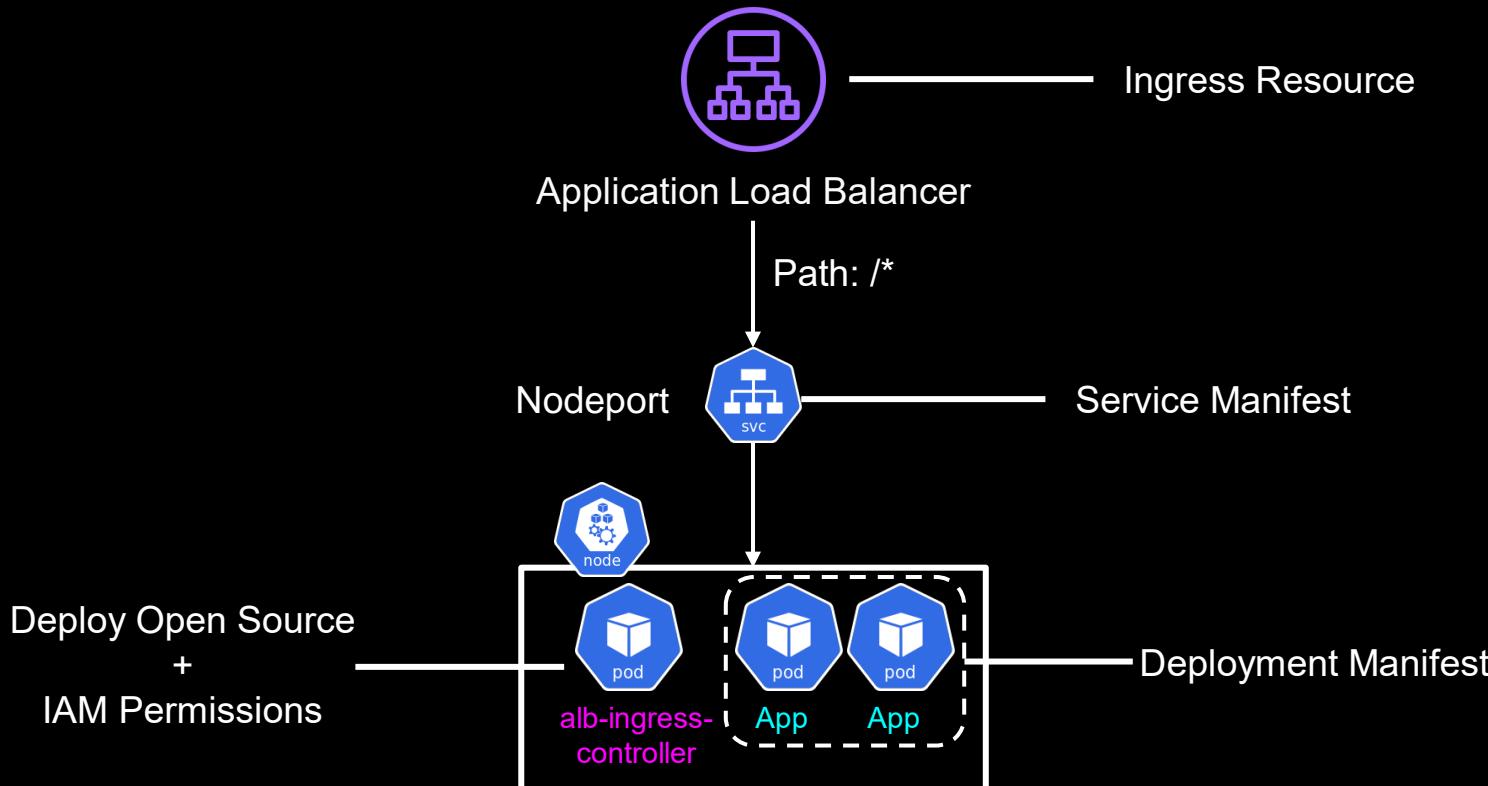


## INGRESS TRAFFIC

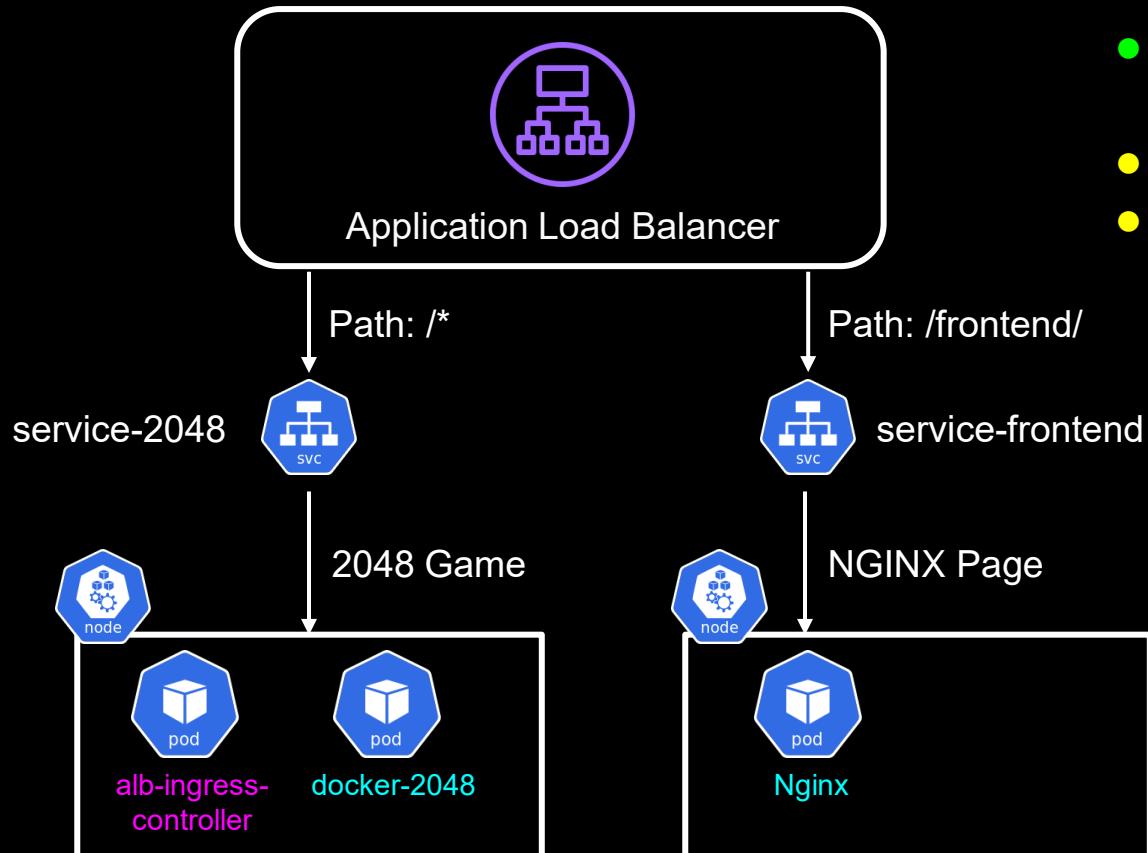
- **instance mode**
  - ALB to Nodeport to pods
- **ip mode**
  - ALB to pods directly
  - Require secondary IP address on ENI as pod IP for networking plugin (AWS CNI plugin for kubernetes)
  - Saves one hop
  - `alb-ingress-controller` specific feature

# ALB Ingress Demo

# ALB Ingress Demo-1

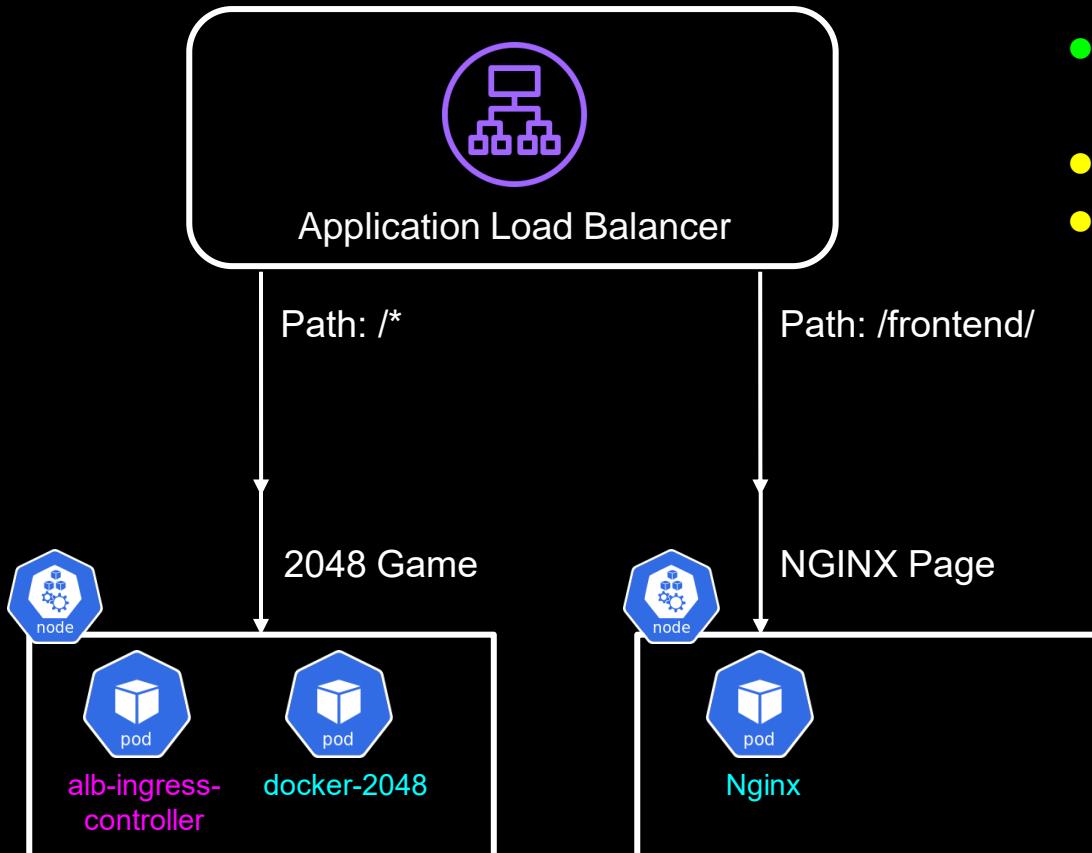


# ALB Ingress Demo-2



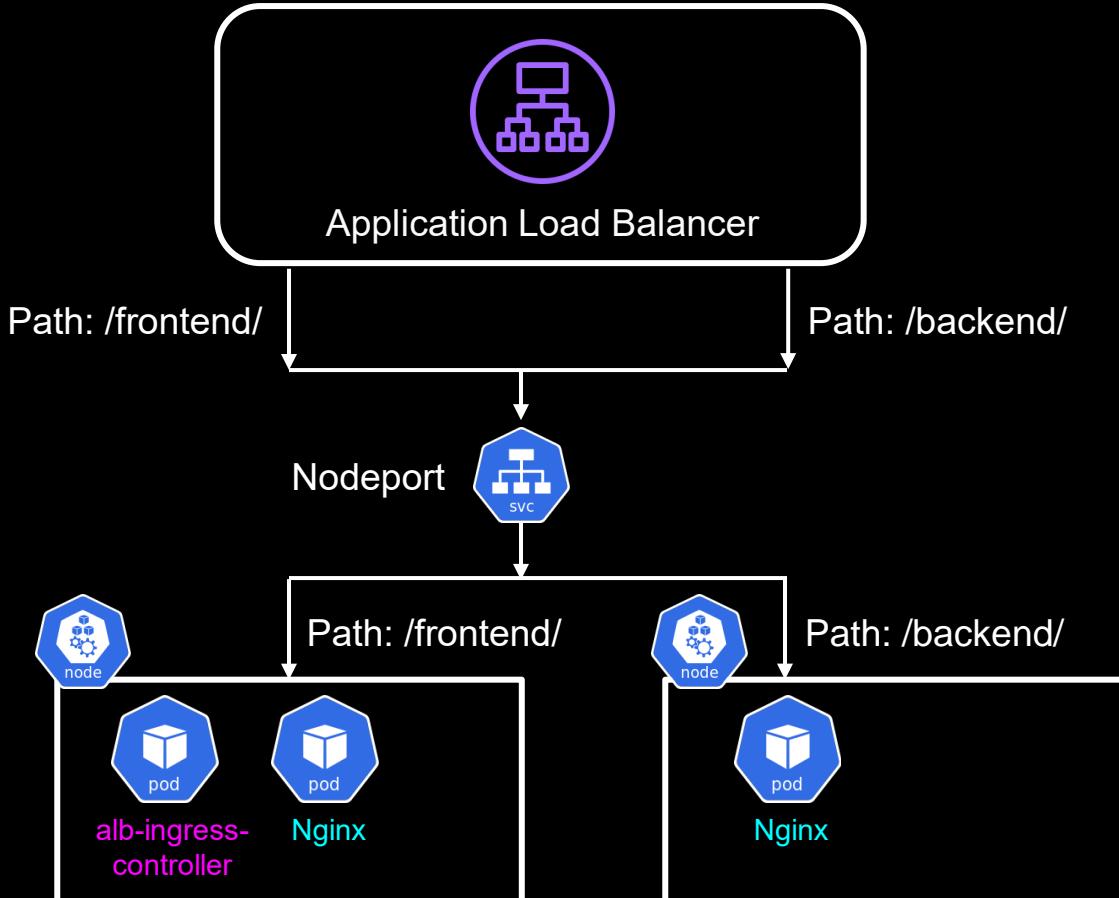
- NOTE - Ingress path ordering matters
- Tweaking in Container!
- IP Mode - bypass one hop!

# ALB Ingress Demo-2



- NOTE - Ingress path ordering matters
- Tweaking in Container!
- IP Mode - bypass one hop!

# ALB Ingress Demo-3



# Basic HTML

www.example.com  
www.example.com/



/var/www/html/index.html

www.example.com/frontend



/var/www/html/frontend/index.html

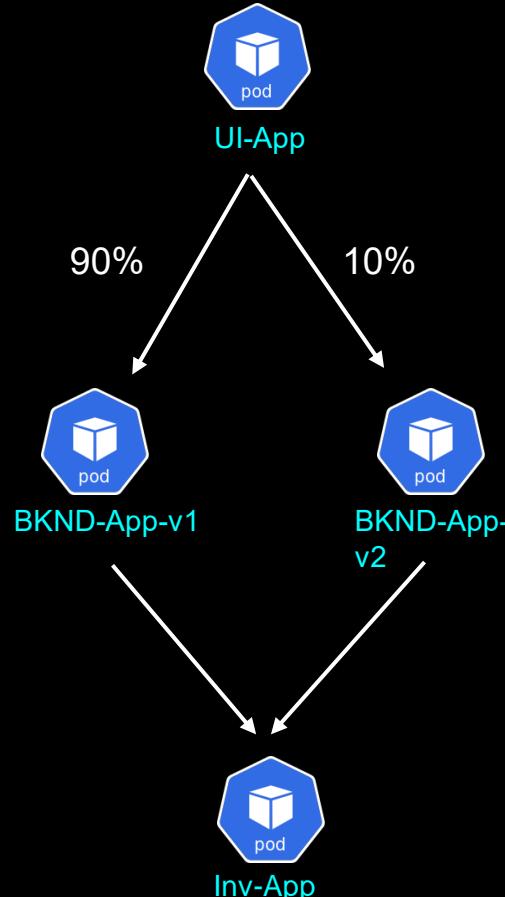
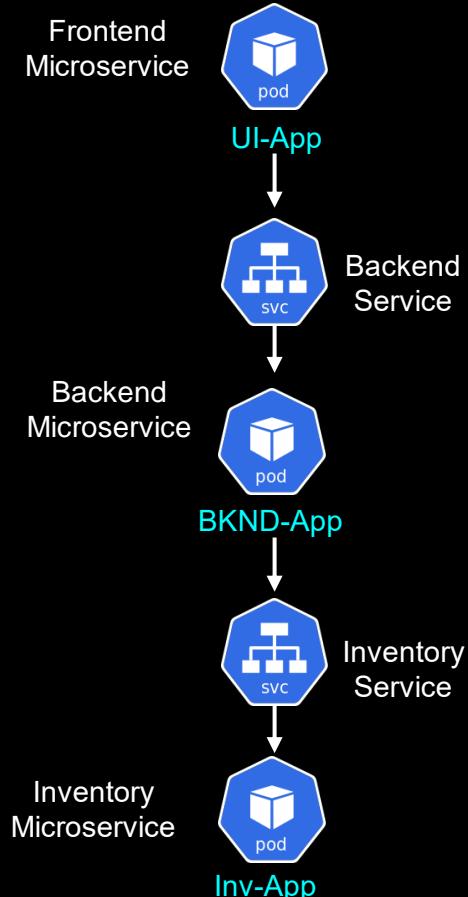
www.example.com/backend



/var/www/html/backend/index.html

# Service Mesh

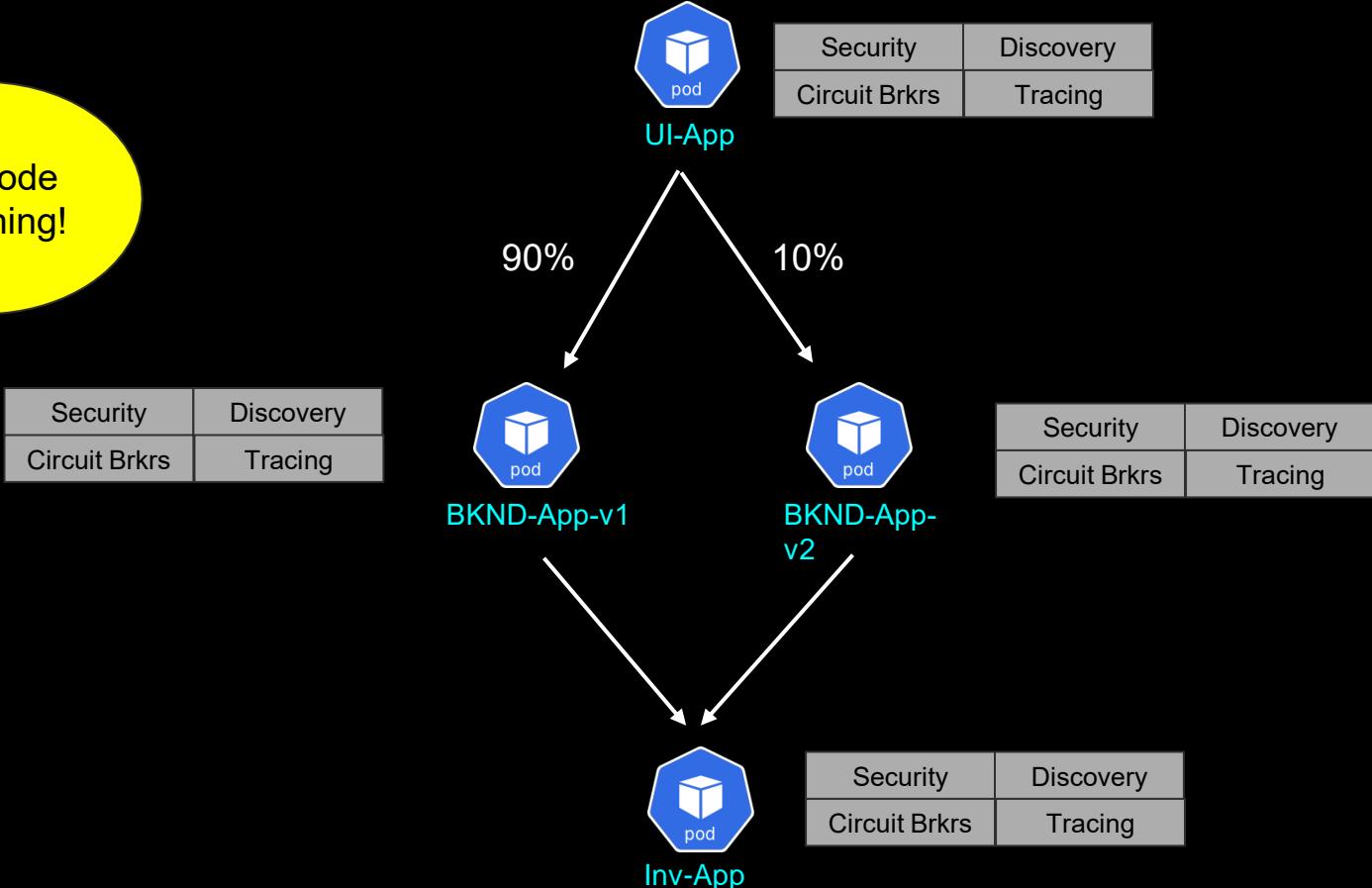
# Regular K8s Application



# Regular K8s Application



I can code  
everything!

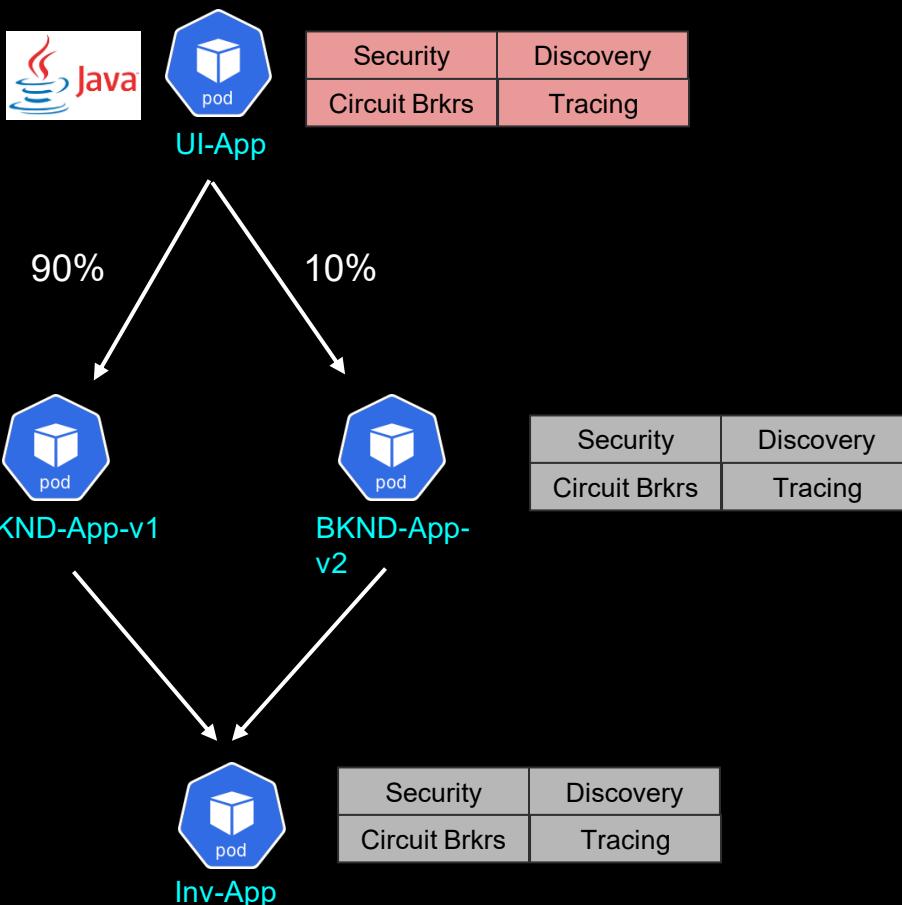


# Regular K8s Application



I can code  
everything!

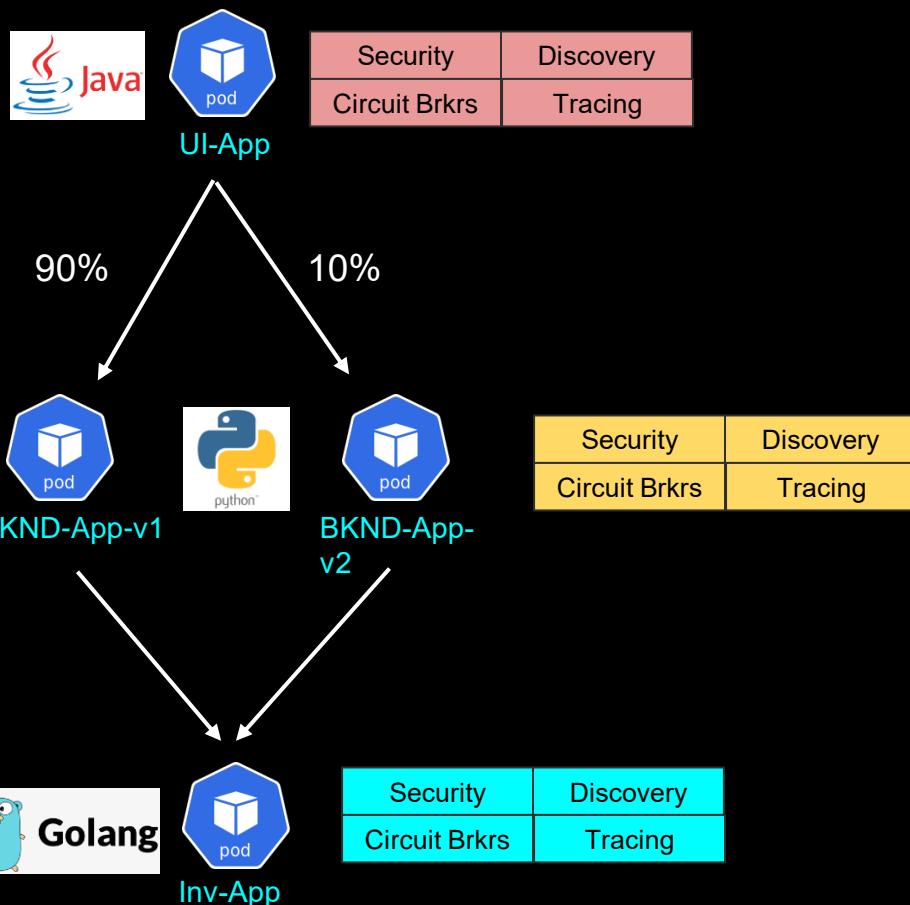
Security	Discovery
Circuit Brkrs	Tracing



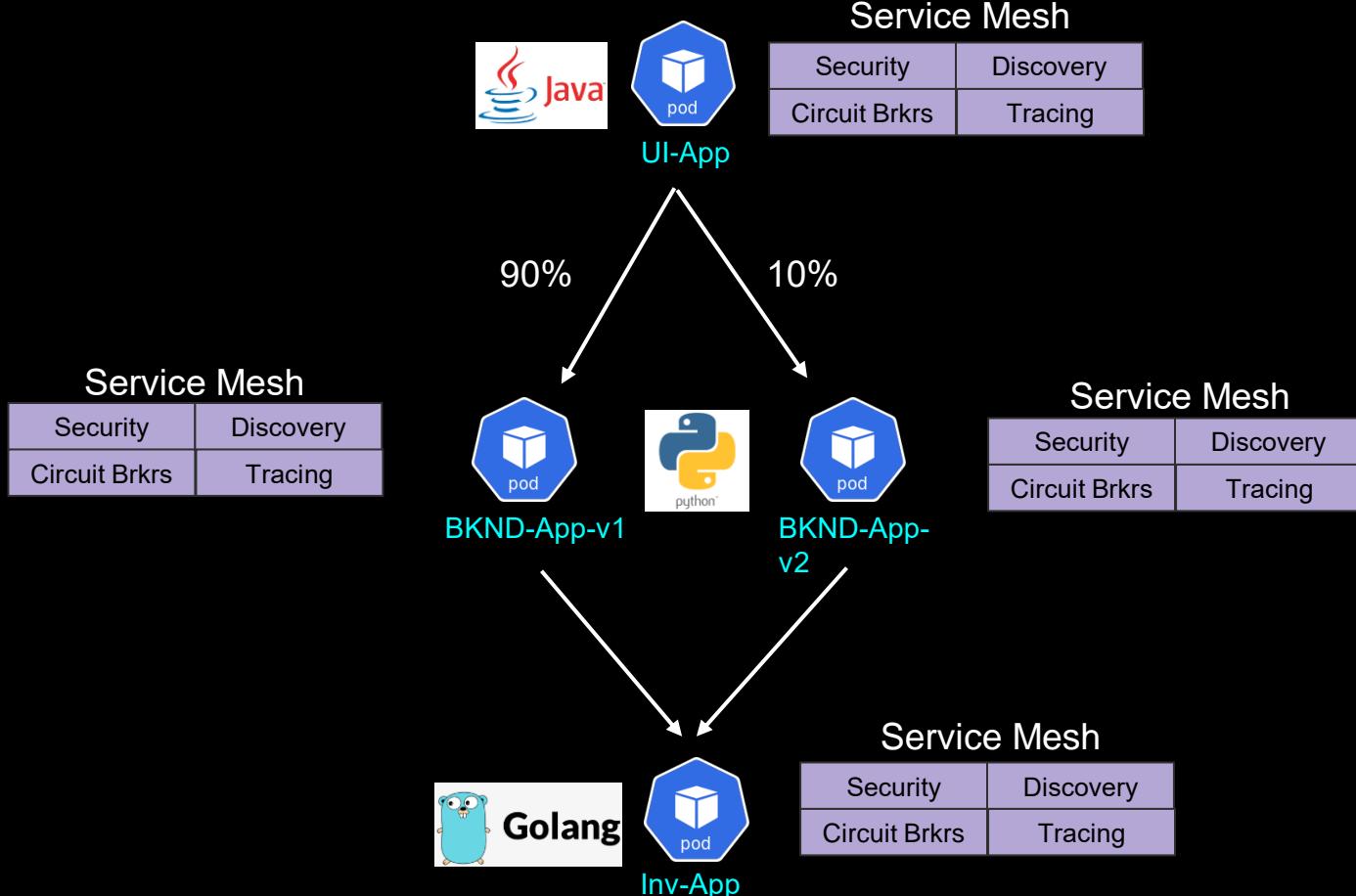
# Polyglot



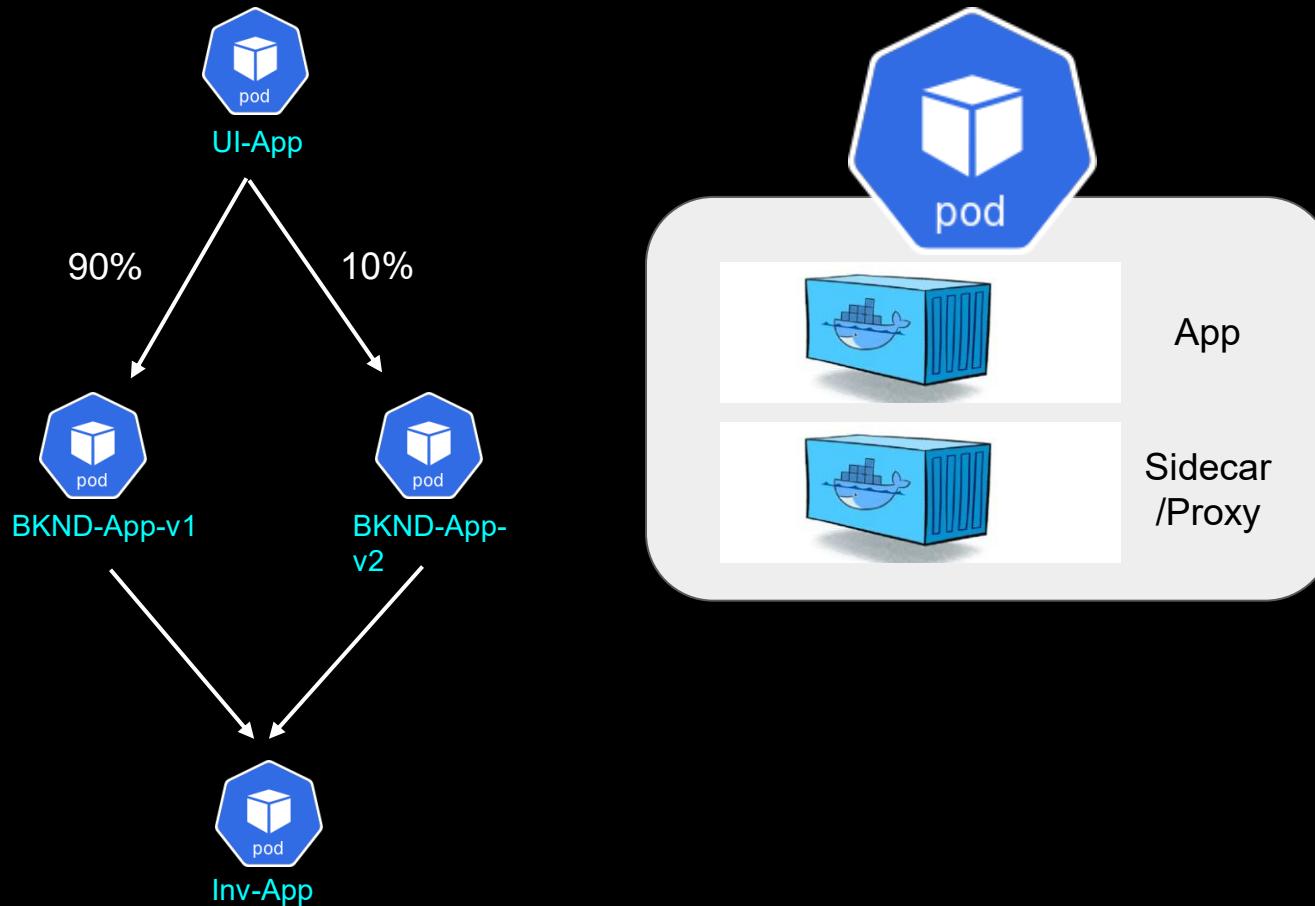
Security	Discovery
Circuit Brkrs	Tracing



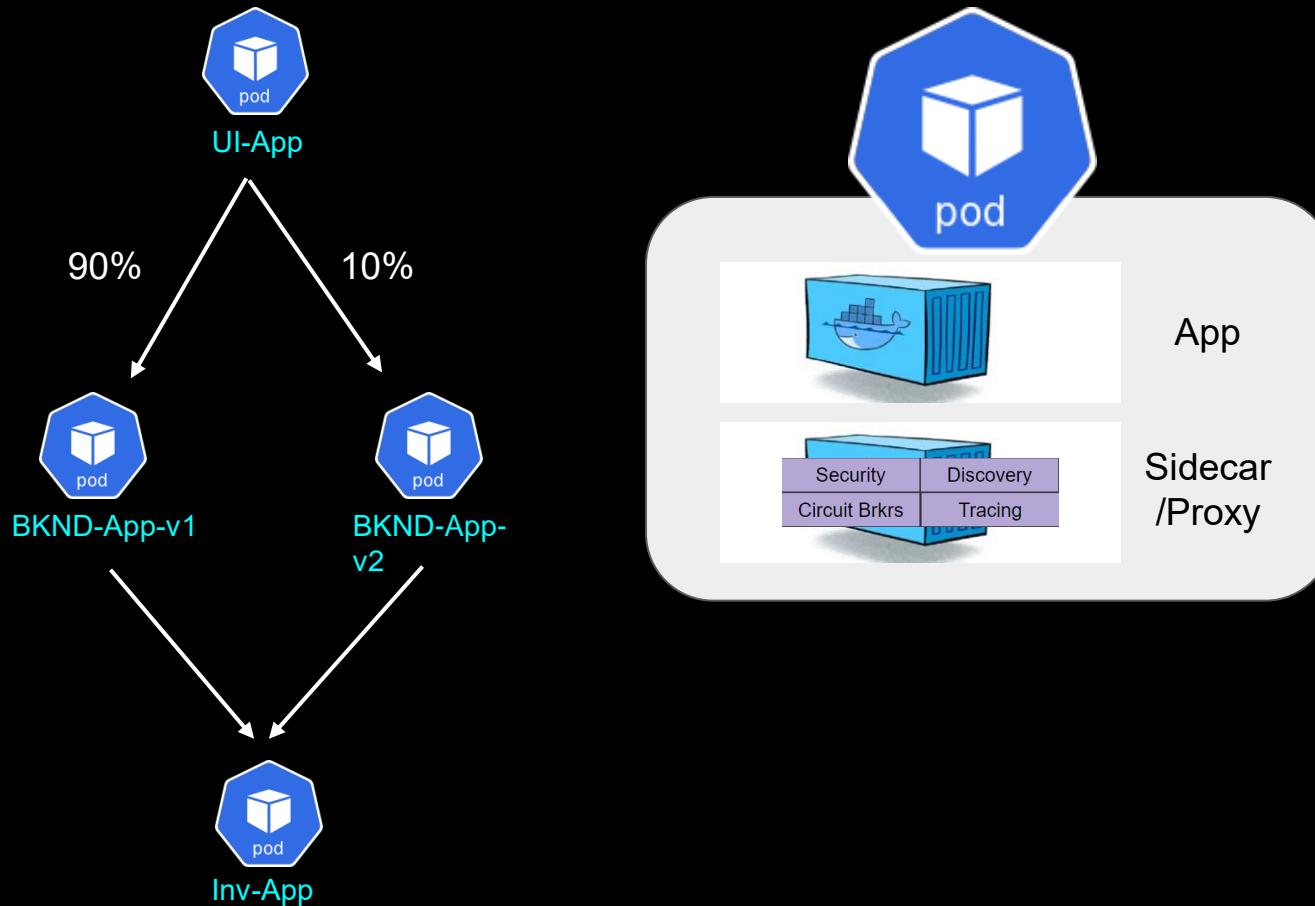
# Say Hello to Service Mesh!



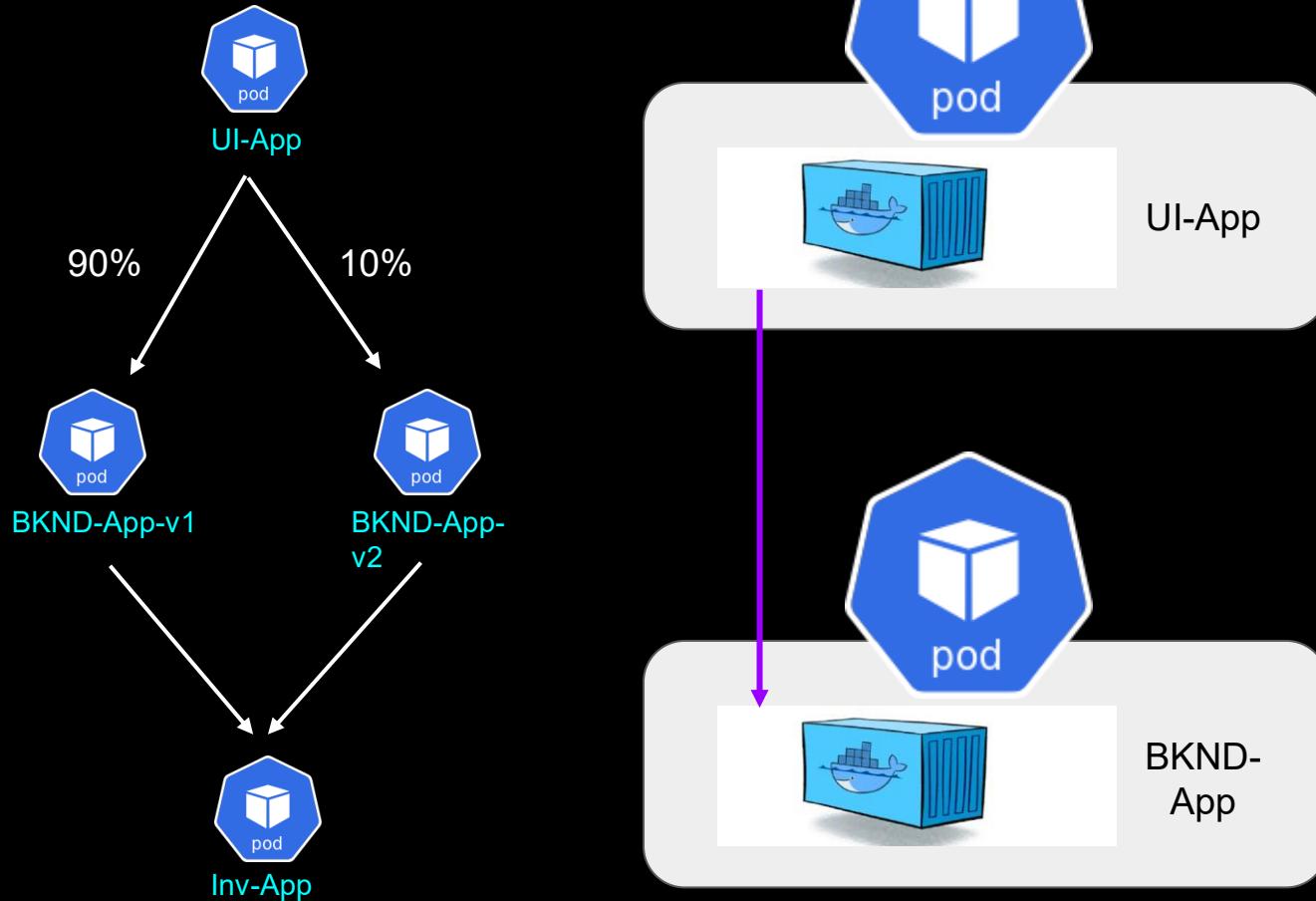
# Anatomy of a Pod



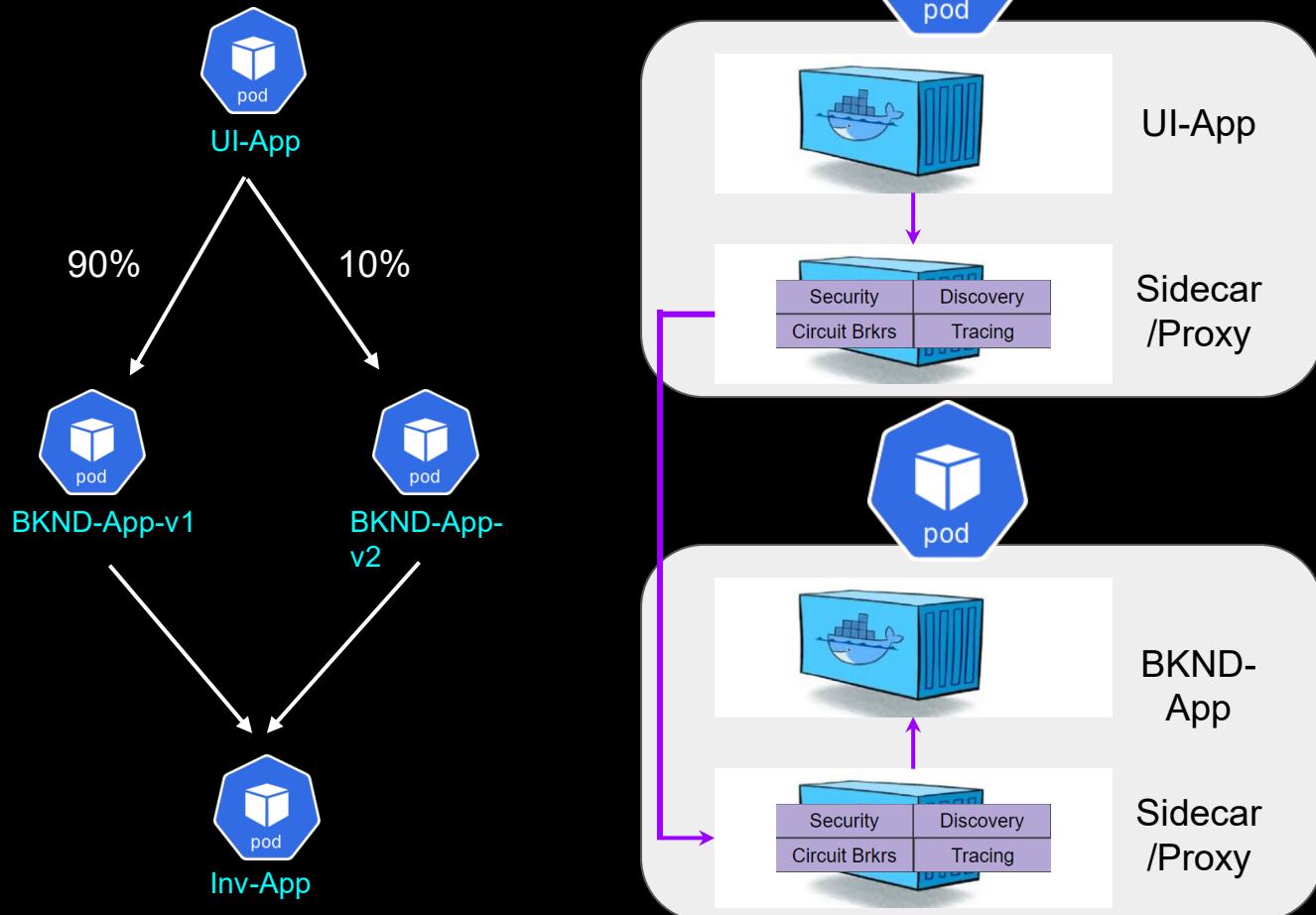
# Anatomy of a Pod



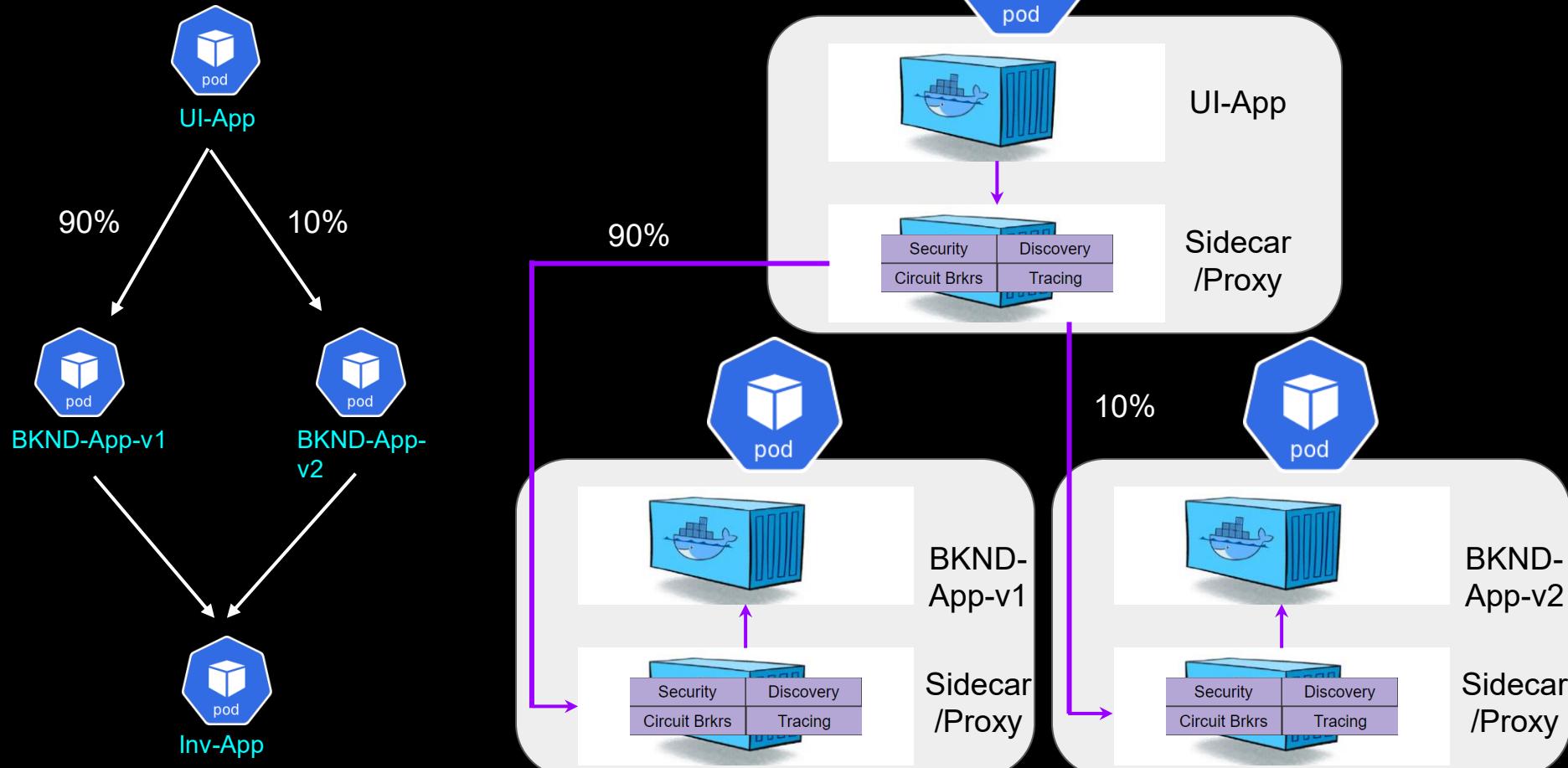
# Without Service Mesh



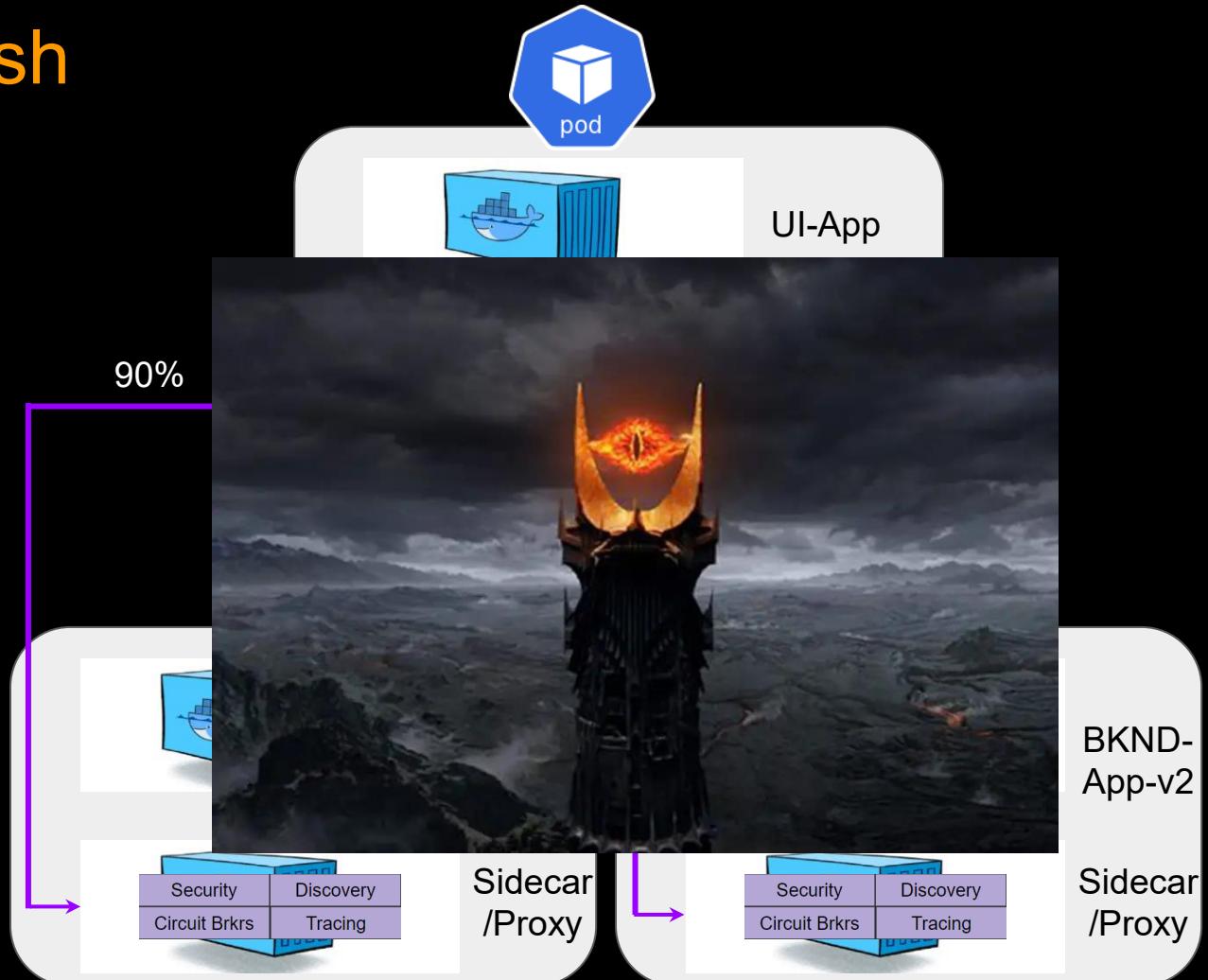
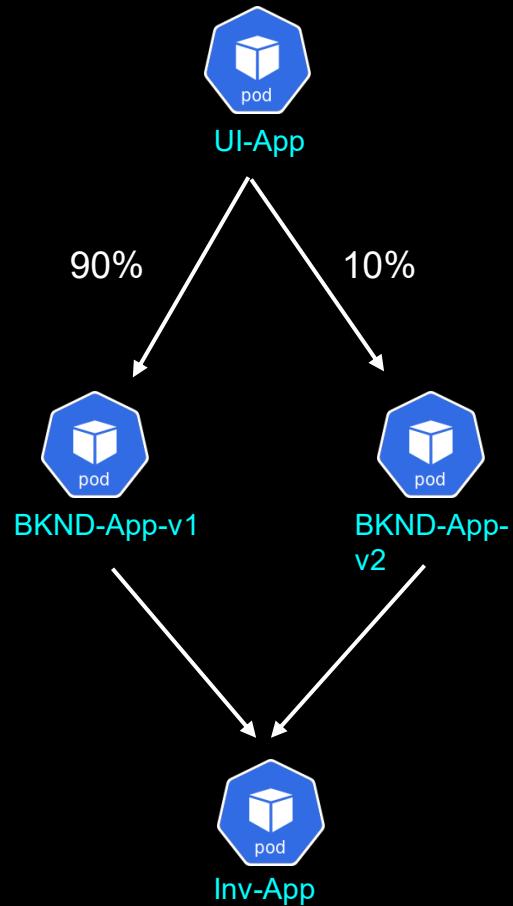
# With Service Mesh



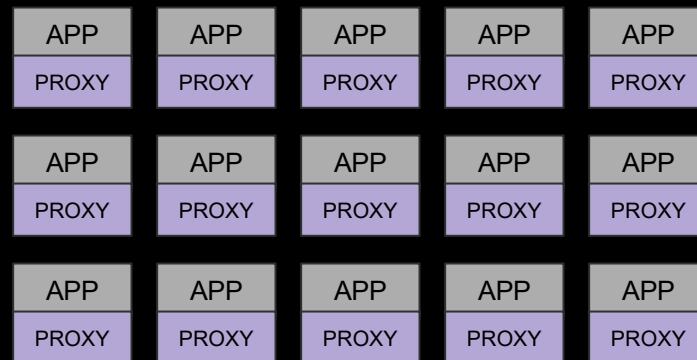
# With Service Mesh



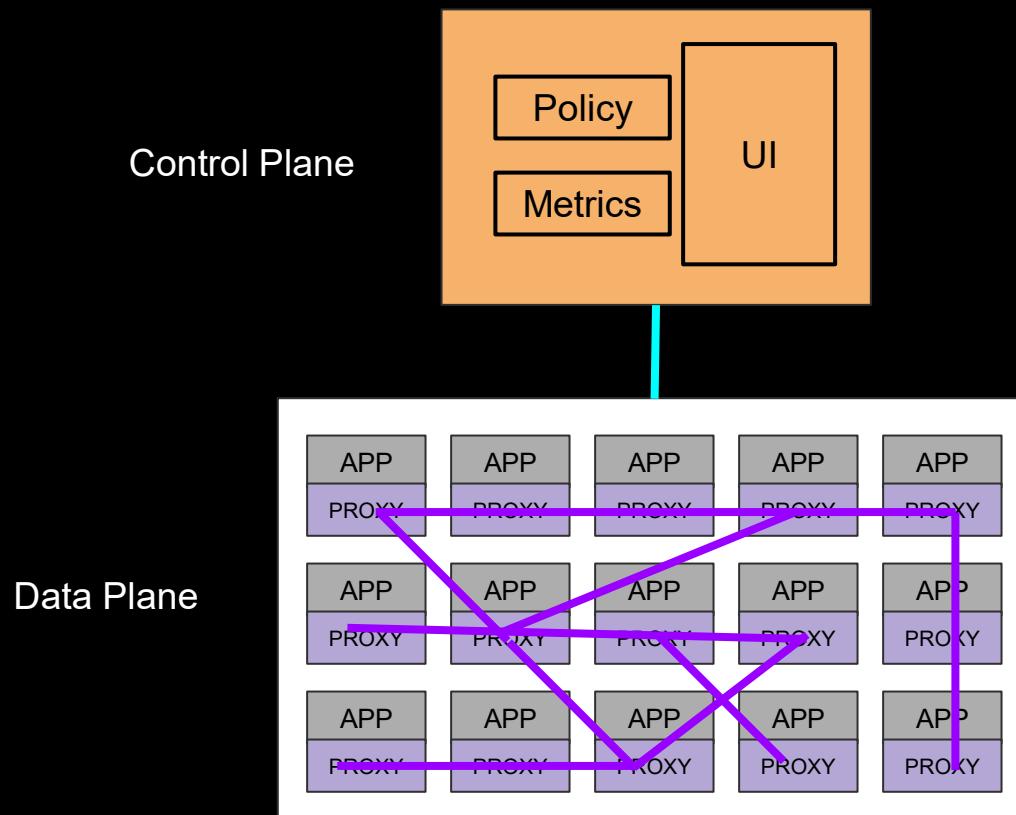
# With Service Mesh



# Control Plane - Data Plane



# Control Plane - Data Plane



# Popular Service Meshes



AWS App Mesh

# CNI (Container Network Interface ) (Moderate to Advanced)

# Agenda

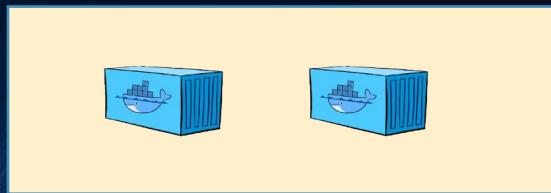
- What is CNI ?
- Kubernetes Networking – Pod and Host Networking
  - Amazon VPC CNI
- What do you do with CNI?
  - Kubernetes Networking Policy
- CNI Evolution - Pod Security Group
- Kubernetes Networking Policy vs Service Mesh

# Big Picture

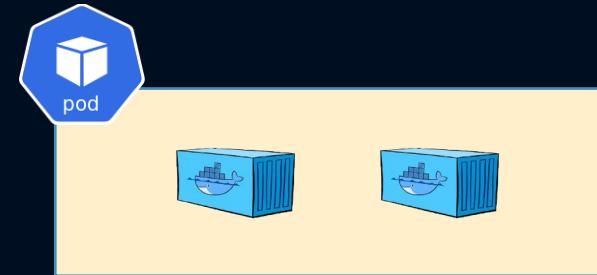


# CRI (Container Runtime Interface)

- Pulls images and runs container
- Docker, RKT, CRI-O etc.



Task

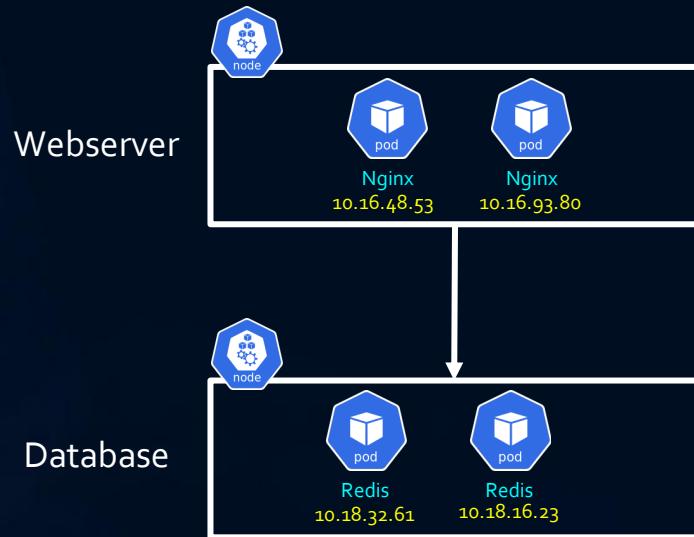


Pod



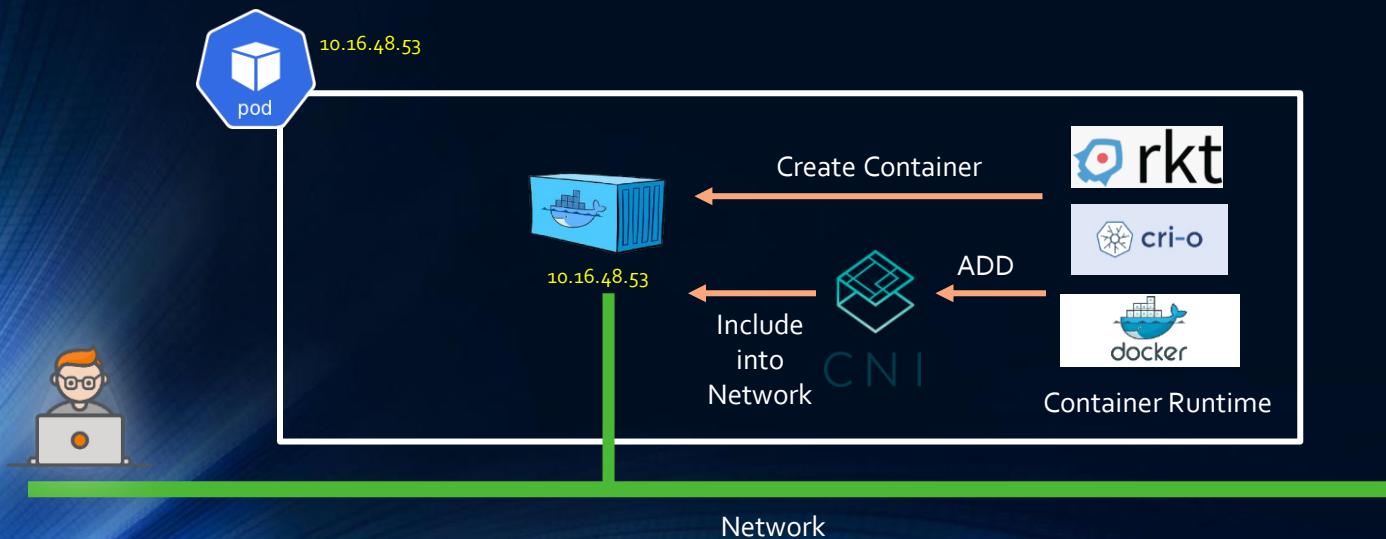
Amazon Elastic Container  
Service

# How do you access Pod?

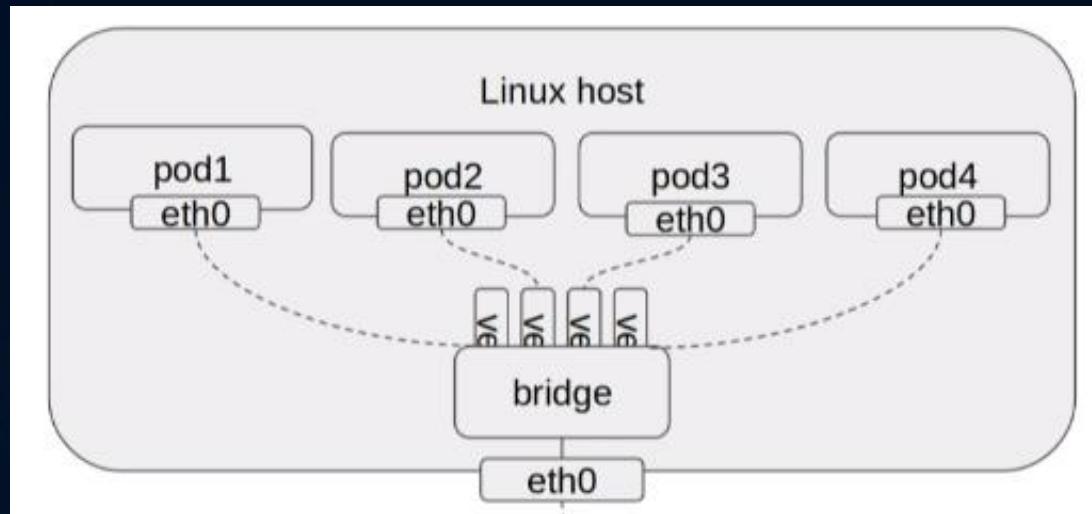


# CNI (Container Network Interface)

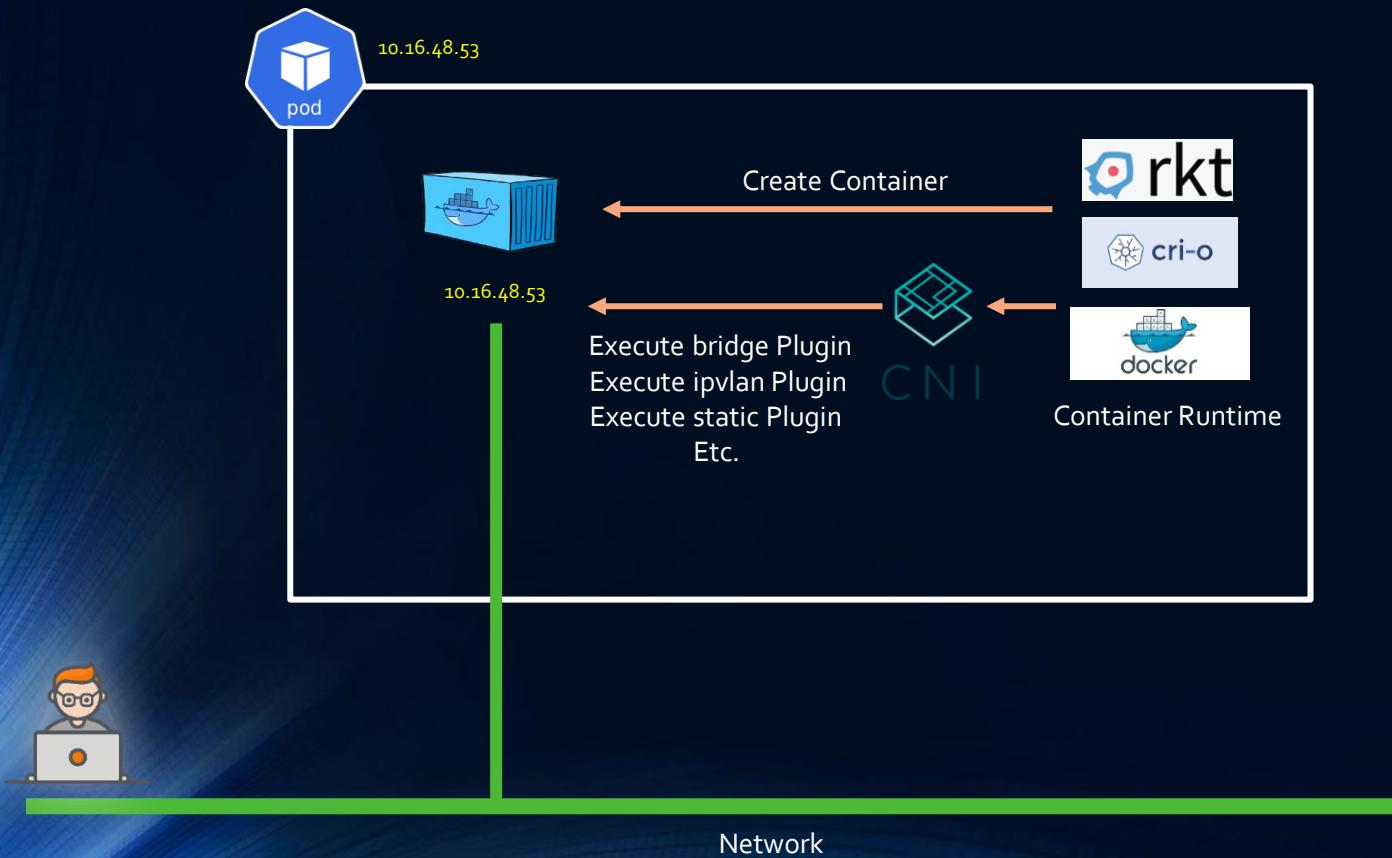
- Interface between container runtime interface and network
- Configures the network routes



# Kubernetes Network



# CNI Plugin



## Plugins supplied:

### Main: interface-creating

- `bridge` : Creates a bridge, adds the host and the container to it.
- `ipvlan` : Adds an `ipvlan` interface in the container.
- `loopback` : Set the state of loopback interface to up.
- `macvlan` : Creates a new MAC address, forwards all traffic to that to the container.
- `ptp` : Creates a veth pair.
- `vlan` : Allocates a vlan device.
- `host-device` : Move an already-existing device into a container.

### Windows: windows specific

- `win-bridge` : Creates a bridge, adds the host and the container to it.
- `win-overlay` : Creates an overlay interface to the container.

### IPAM: IP address allocation

- `dhcp` : Runs a daemon on the host to make DHCP requests on behalf of the container
- `host-local` : Maintains a local database of allocated IPs
- `static` : Allocate a static IPv4/IPv6 addresses to container and it's useful in debugging purpose.

### Meta: other plugins

- `flannel` : Generates an interface corresponding to a flannel config file
- `tuning` : Tweaks sysctl parameters of an existing interface
- `portmap` : An iptables-based portmapping plugin. Maps ports from the host's address space to the container.
- `bandwidth` : Allows bandwidth-limiting through use of traffic control tbf (ingress/egress).
- `sbr` : A plugin that configures source based routing for an interface (from which it is chained).
- `firewall` : A firewall plugin which uses iptables or firewalld to add rules to allow traffic to/from the container.

# CNI is NOT just for Kubernetes

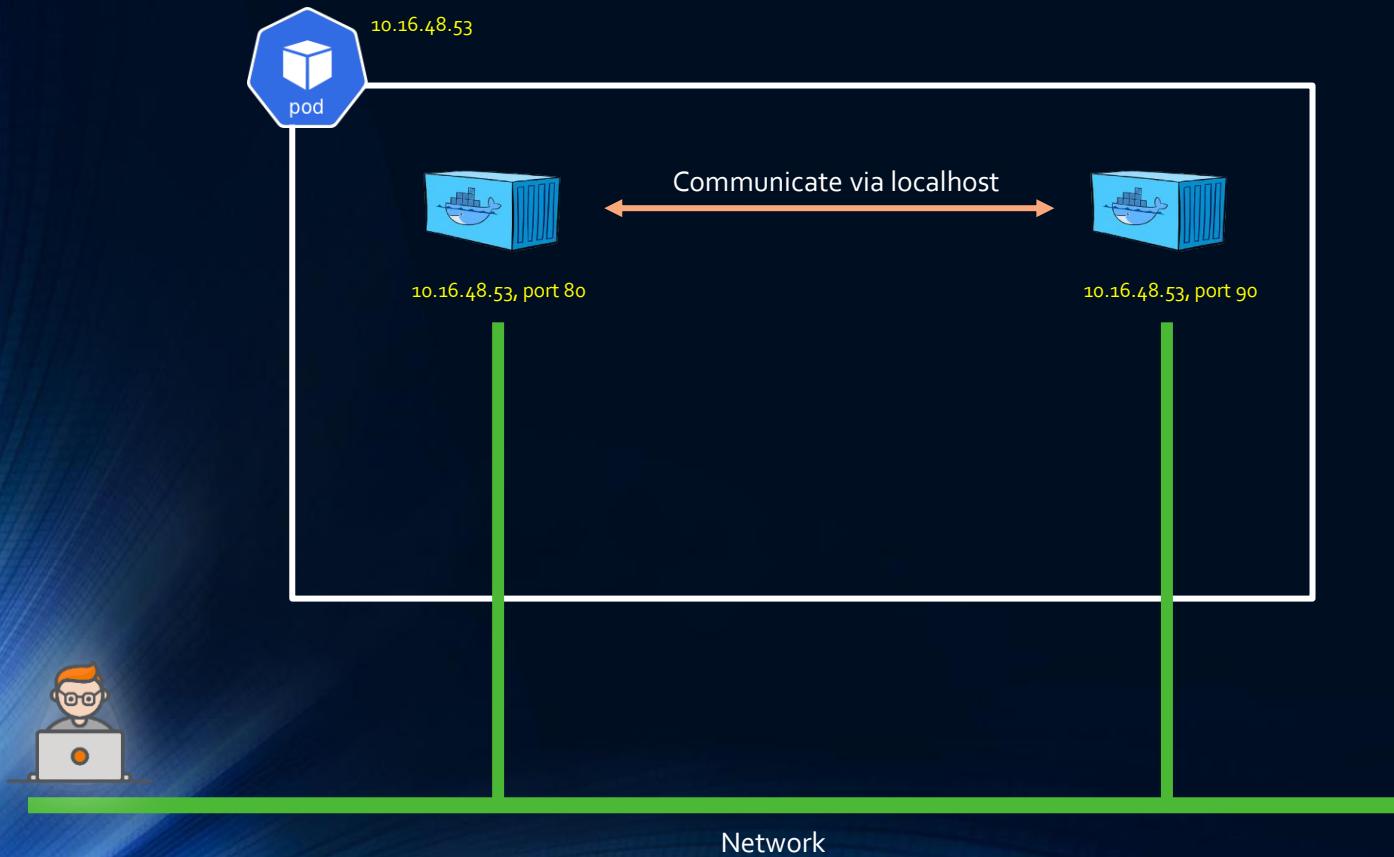
- Any CRI (Container Runtime Interface) can call CNI Plugins
- CNI Plugins need to follow certain specifications
  - Specific parameters
  - Need to support specific operations (ADD, DEL, CHECK, VERSION)

<https://github.com/containernetworking/cni/blob/master/SPEC.md>

# Kubernetes Networking Requirements

- Each pod gets its own IP
  - Containers within a pod share network namespaces
- All pods can communicate with all other pods without NAT (Network Address Translation)
- All nodes can communicate with all pods without NAT
- The IP of the pod is same throughout the cluster

# Containers in same Pod



# Available CNIs



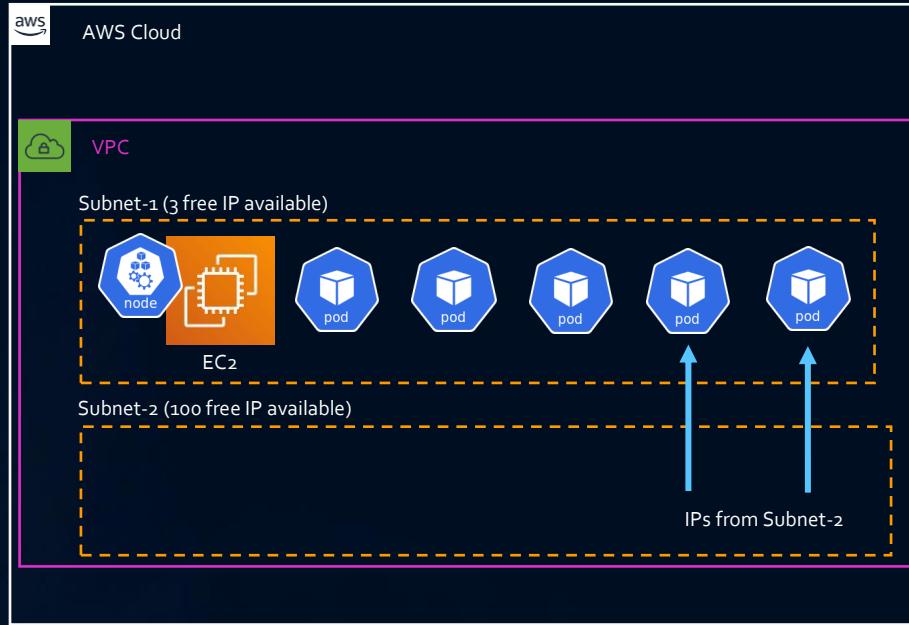
Amazon  
VPC  
CNI Plugin





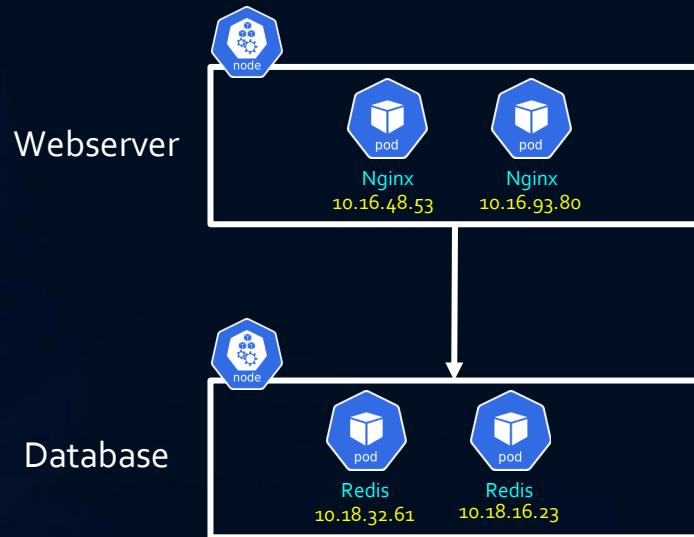
- Pod IP is same throughout the VPC
- Native VPC networking fast performance, scalable
- VPC features – flow logs, direct connect etc.
- Support pod security group (more on this later!)
- Open source, supported by AWS

# Amazon VPC CNI Custom Networking for Pod

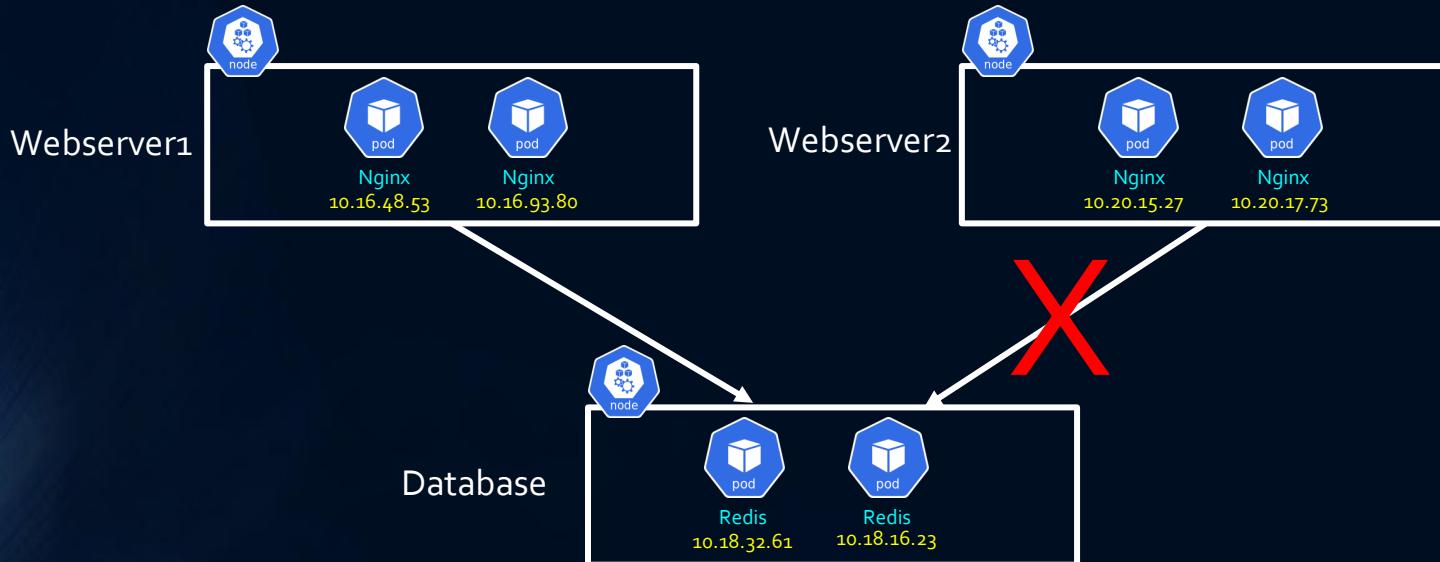


- Using secondary network interface attached to EC2
- Expand EKS cluster by adding additional CIDR to VPC

# What can you do with ALL this?



# Kubernetes Network Policies

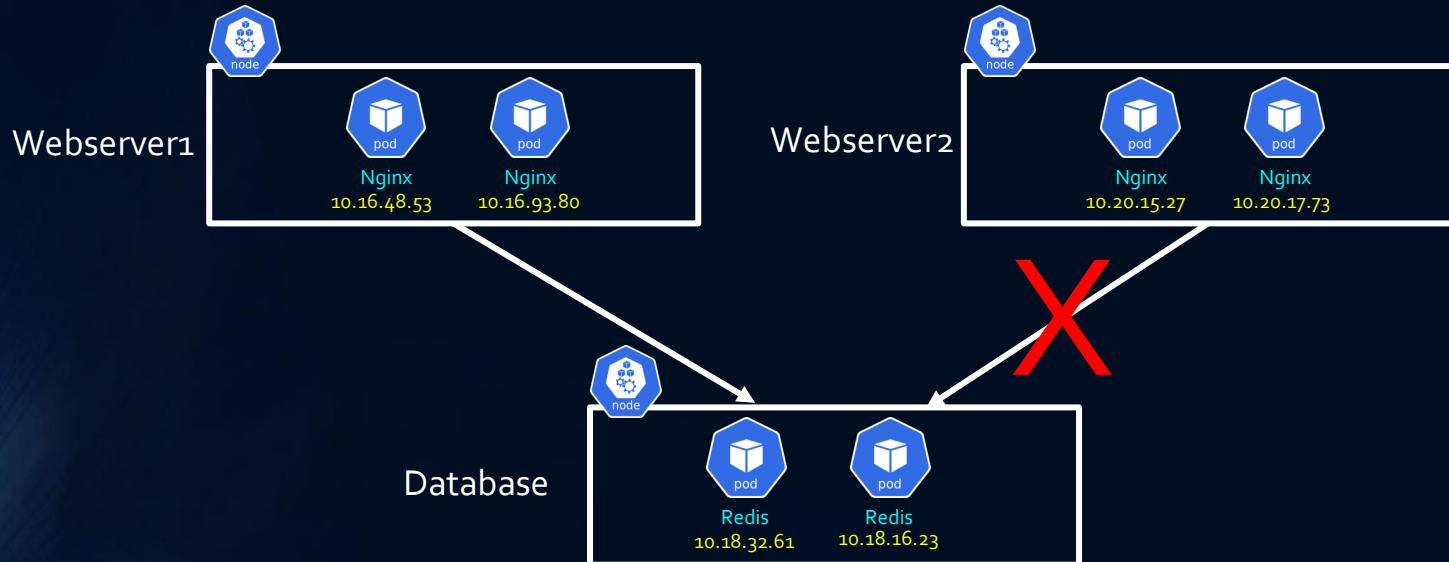


- Great for multi-tenant cluster

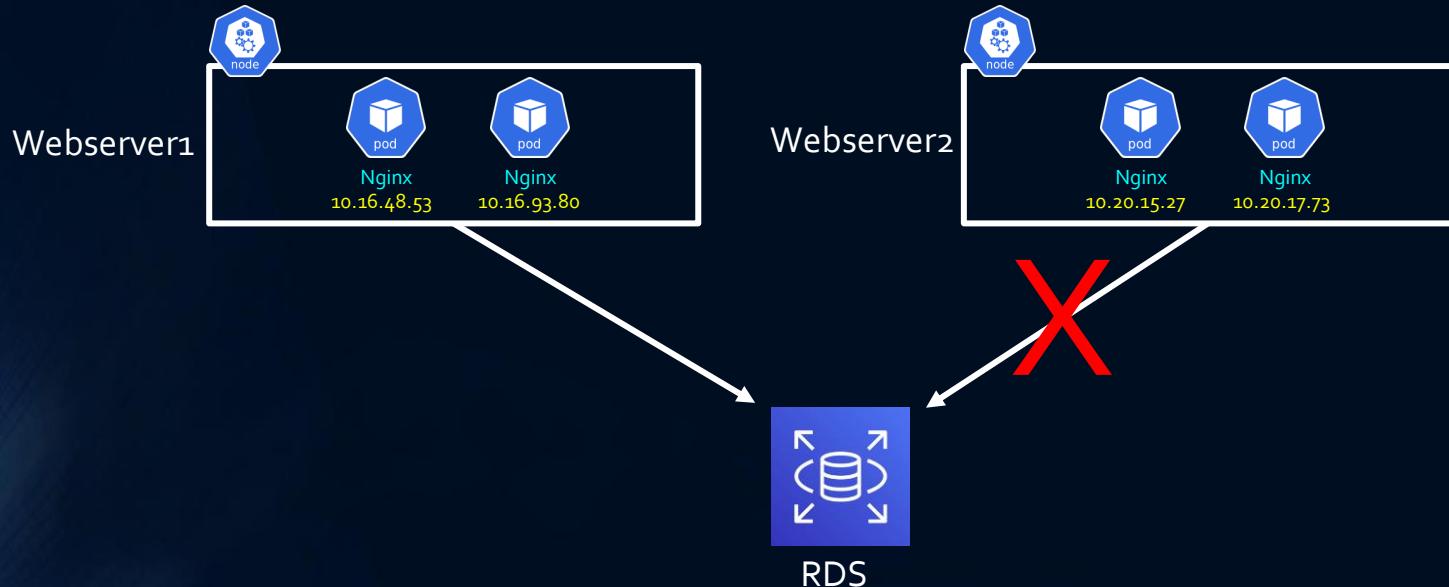
# Kubernetes Network Policies

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
    - Ingress
    - Egress
  ingress:
    - from:
        - ipBlock:
            cidr: 172.17.0.0/16
            except:
              - 172.17.1.0/24
        - namespaceSelector:
            matchLabels:
              project: myproject
        - podSelector:
            matchLabels:
              role: frontend
      ports:
        - protocol: TCP
          port: 6379
  egress:
    - to:
        - ipBlock:
            cidr: 10.0.0.0/24
      ports:
        - protocol: TCP
          port: 5978
```

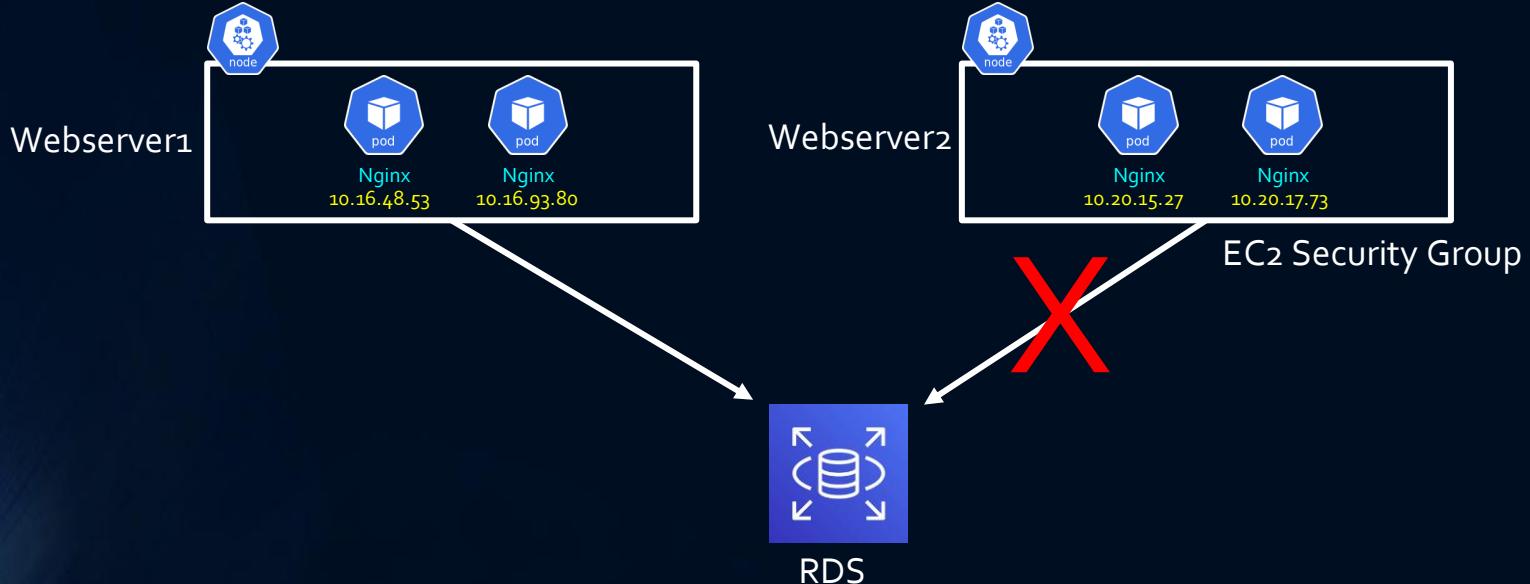
# What if a resource doesn't expose IP?



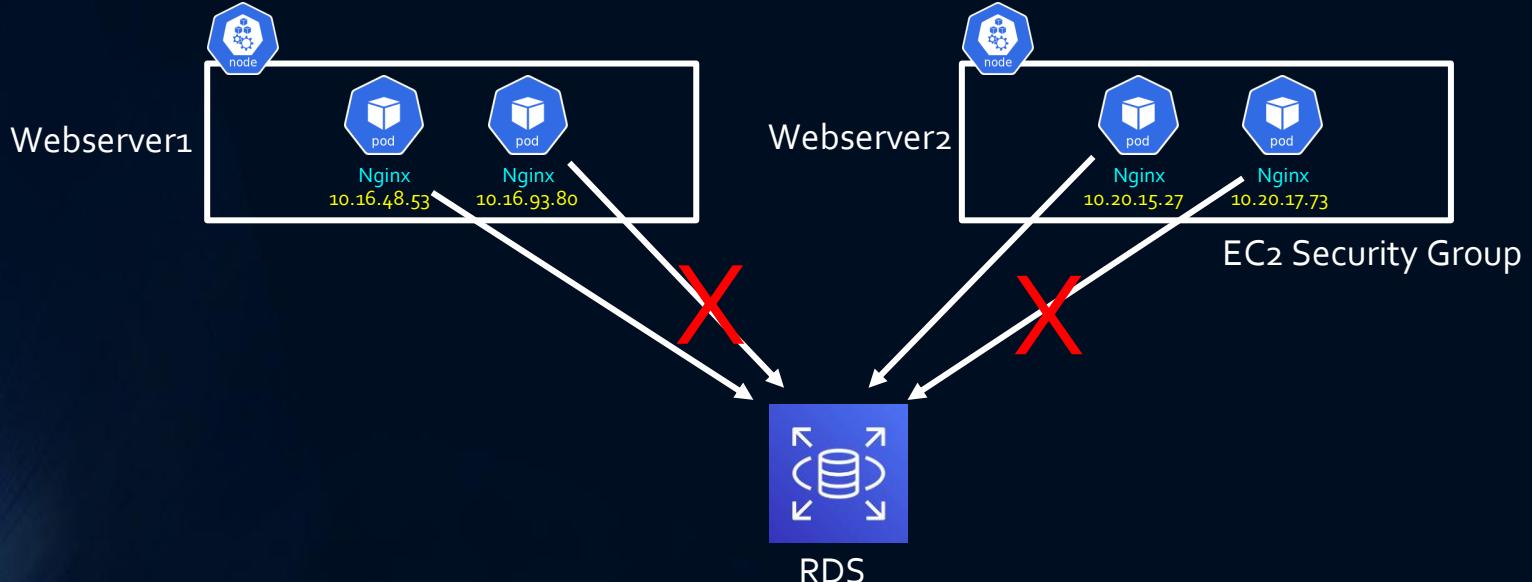
# What if a resource doesn't expose IP?



# Issue with EC2 Security Group

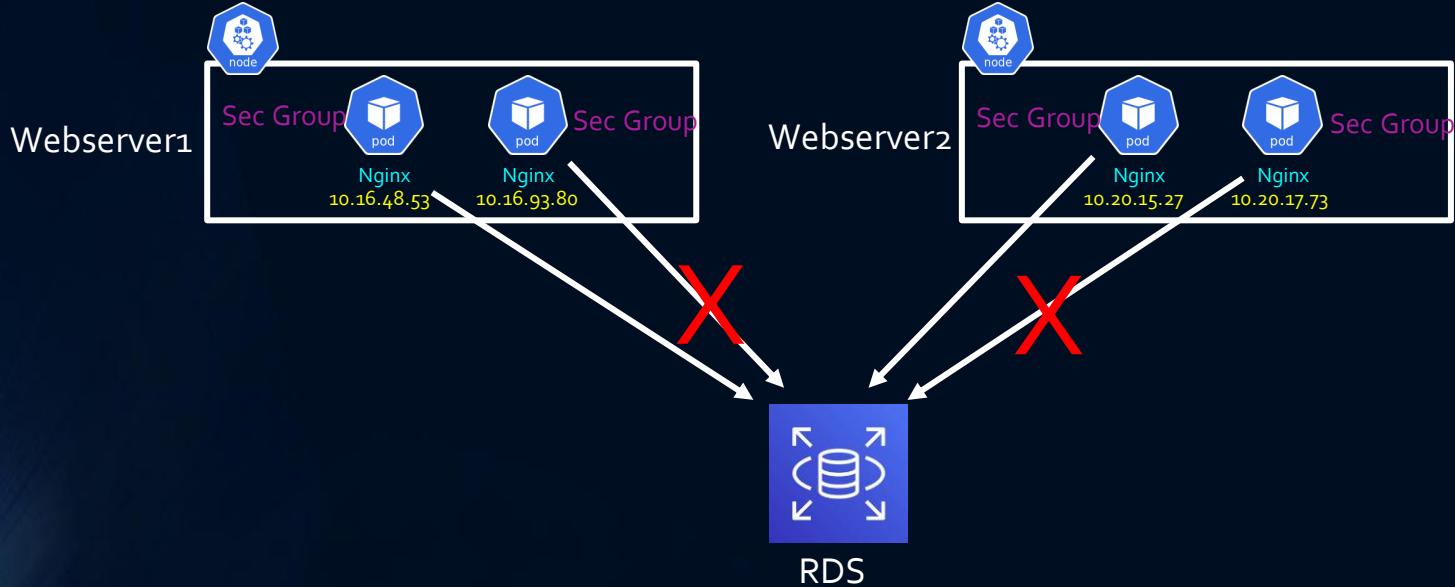


# Issue with EC2 Security Group



- Requires using taint to schedule specific pods to specific nodes
- Gets complex as cluster scales

# Pod Security Group EKS



- Security control in multi tenant cluster
- Control network access from pods to outside cluster AWS services
- Easier to migrate from EC2 to EKS

# Service Mesh vs. Network Policy Enforcer

# EKS Cost Optimization



# 4 Commandments of Cost Optimization

## Right Sizing

- Utilize pod requests, limits, resource quotas
- Use open source, third-party tools to tune pod requests, limits  
(More on this later)

## Auto Scaling

- Once pods are optimized, enable auto scaling
- Utilize HPA, Cluster Autoscaling, Proportional Autoscaling

## Down Scaling

- Terminate pods unnecessary during nights, weekends
- Utilize DevOps

## EC2 Purchase Options

- Use RI, Spot, Savings Plan

# Right Sizing Tools

Open Source



right size guide



Kubernetes Resource Report



And More...

Third Party



And More...

# EKS Tools Ecosystem

What  
does this  
button  
do?



# EKS Tools Ecosystem

DevOps



Security



Ingress



ALB Ingress Controller

# EKS Tools Ecosystem

Logging



Monitoring



Cost  
Optimization



Kubernetes Resource Report

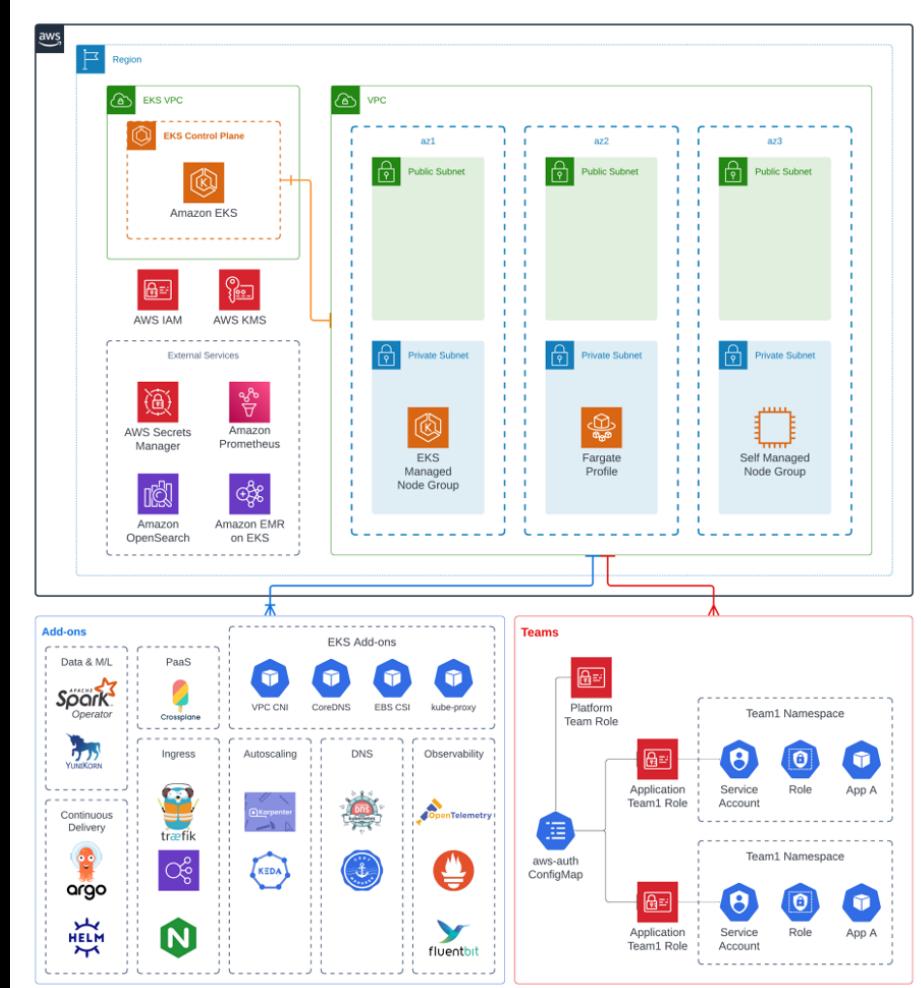
# EKS Blueprints

# CNCF Landscape is MASSIVE!

# EKS Blueprints

- An open source framework that configure and deploy EKS clusters
- AWS best practices and recommendations built in
- Multiple patterns with popular addons
- Available in Terraform and CDK

# Sample Blueprint Architecture



# Sample EKS Blueprints

Complete examples with addons  
EKS cluster with external DNS  
EKS Fargate cluster  
Fully private EKS clusters  
Gitops with ArgoCD  
Gitlab CI/CD  
EFS shared storage  
EMR on EKS  
Cert-manager examples

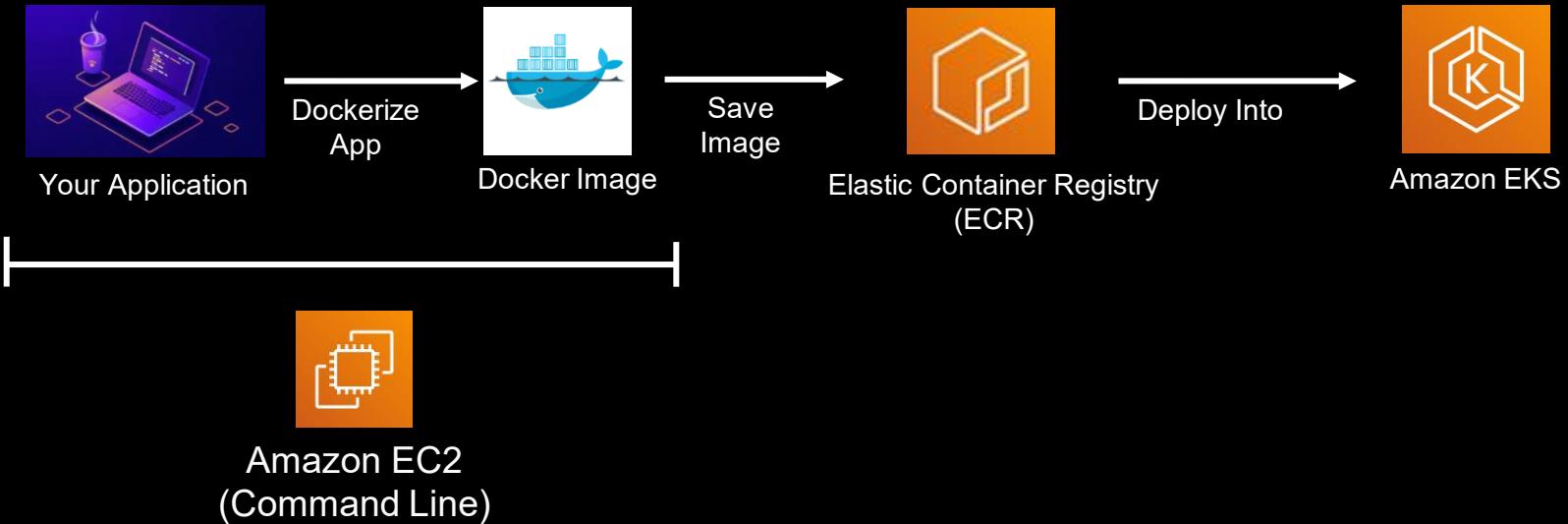
IPV6 cluster  
IPV4 Prefix delegation  
Karpenter  
Multi-tenancy with teams  
Nodes groups advanced configurations  
Apache Airflow  
Stateful cluster  
Observability (Adot, opensearch, managed prometheus..)

# Demo

# EKS Roadmap



# ECR Demo Setup



# EKS Security

# Kubernetes Security

RBAC

IRSA

RoleBinding

ClusterRole

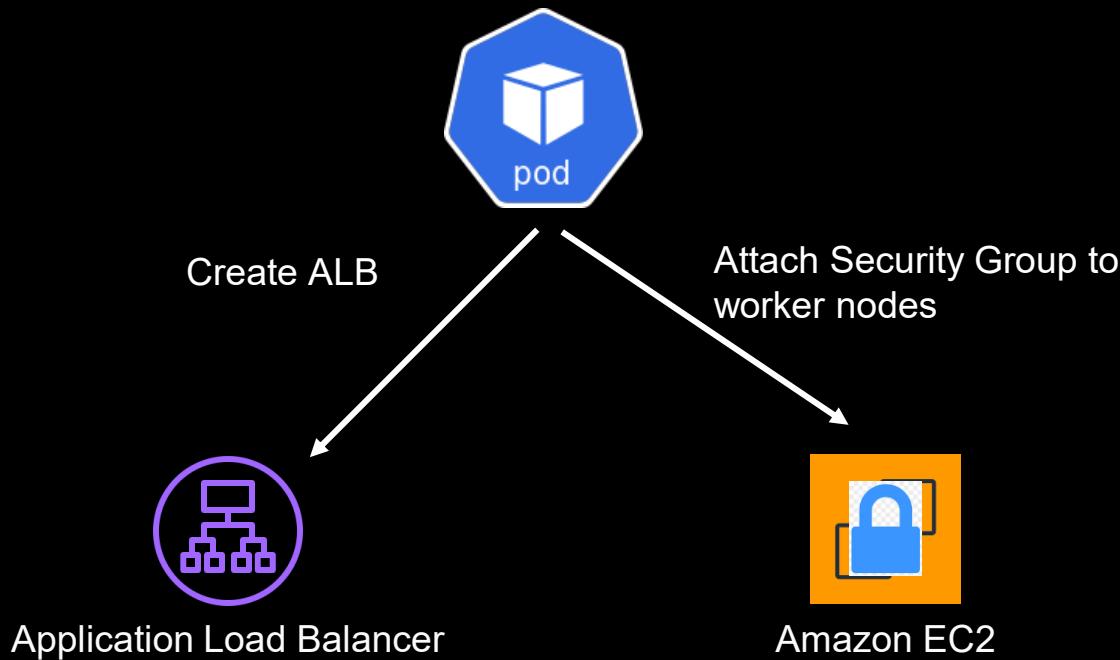
ClusterRoleBinding



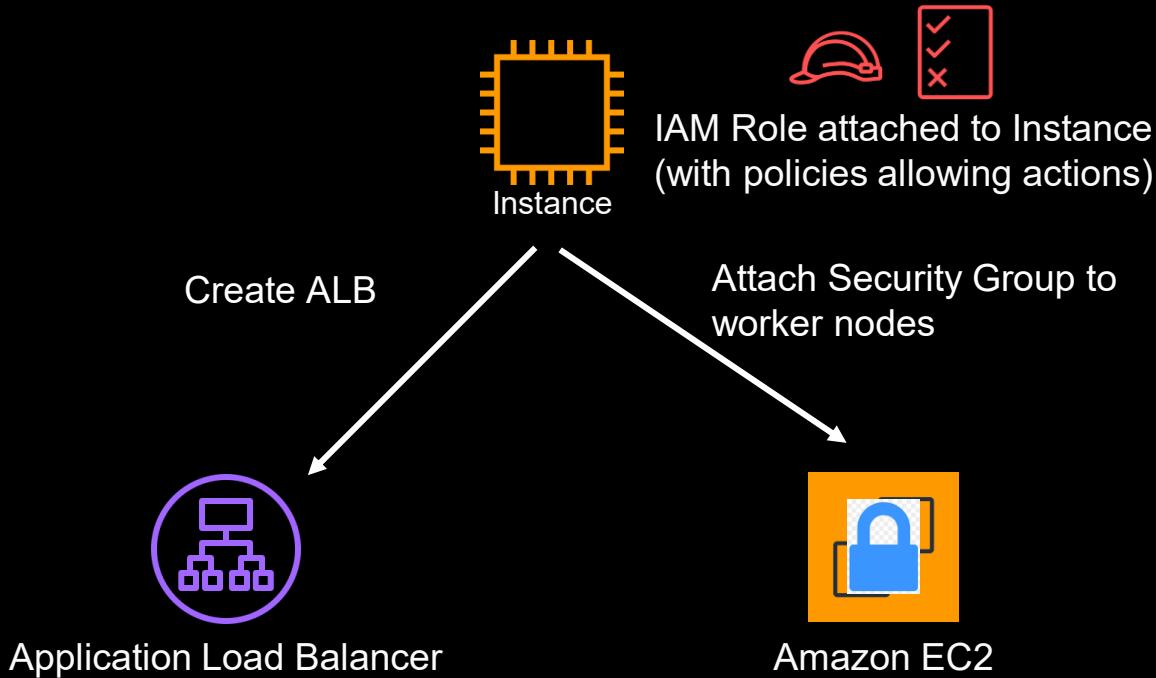
# Kubernetes Security

- Your Application
- You! (As in human users)

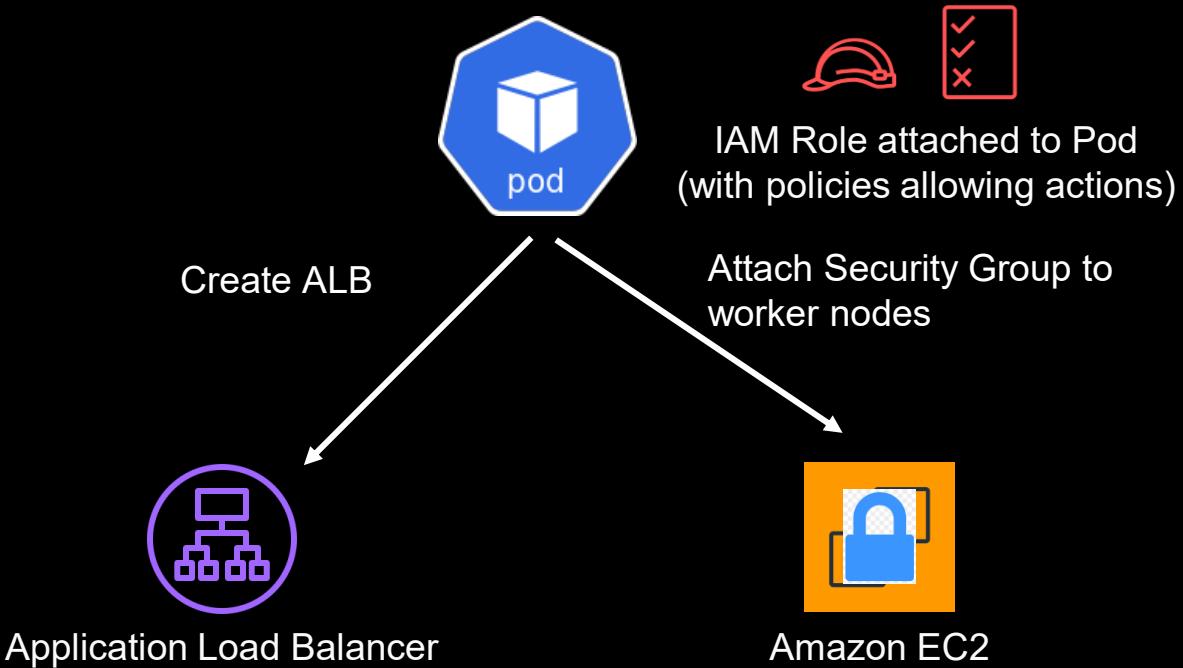
# Working Backwards



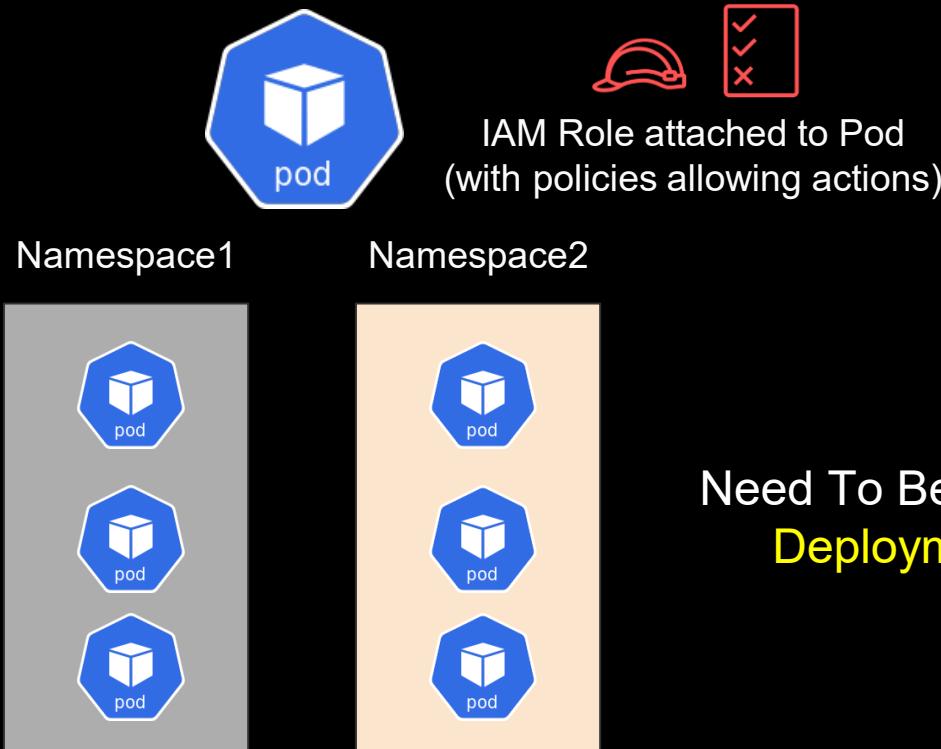
# Back to EC2 Days



# Back to Pod



# Not so simple life of pod

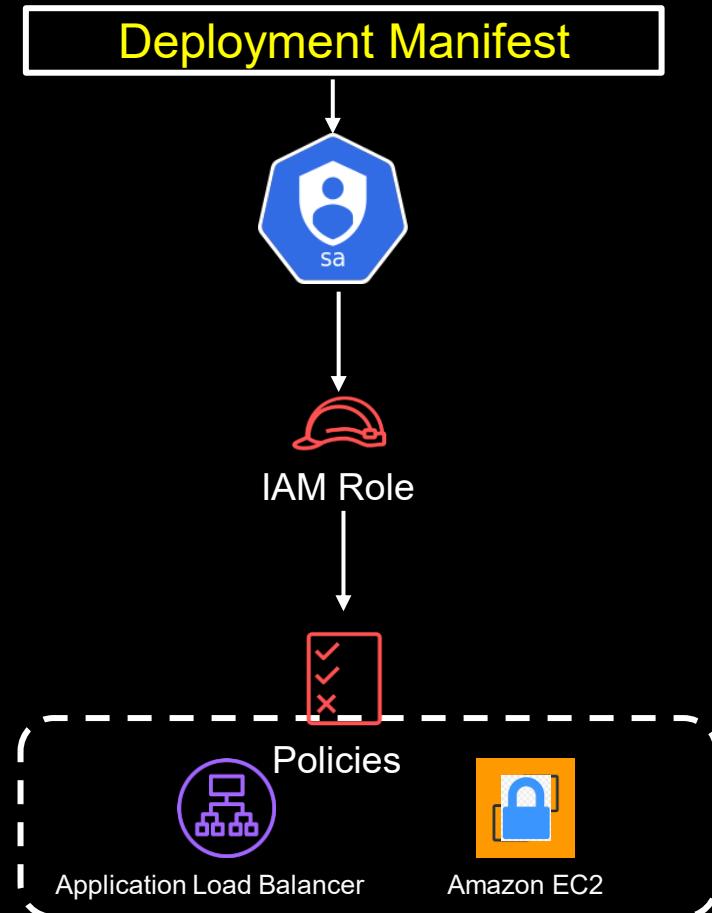


# Deployment Manifest

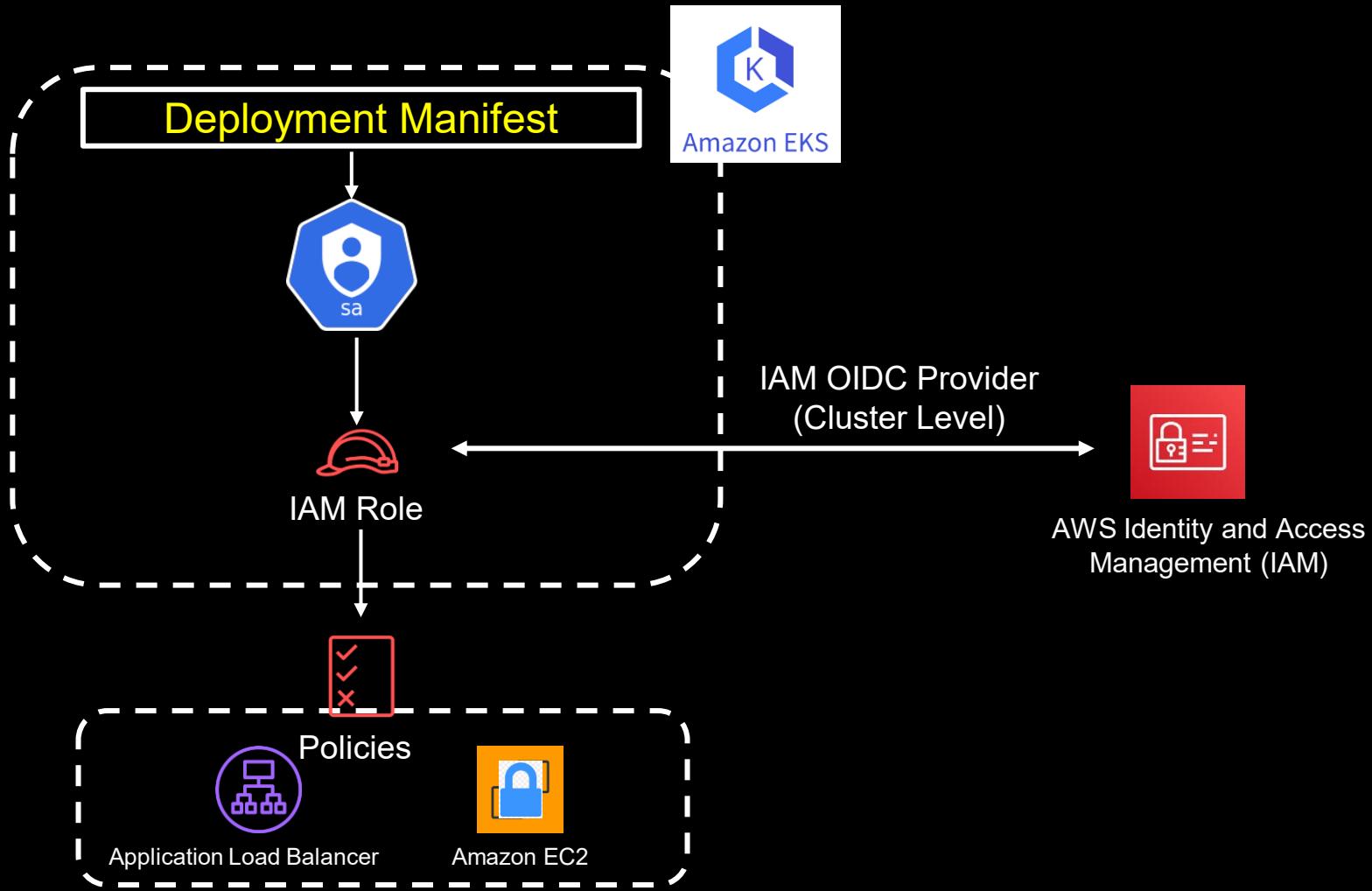
```
! nginx-deployment-withrolling.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      environment: test
6    name: testdeploy
7  spec:
8    replicas: 3
9    selector:
10      matchLabels:
11        environment: test
12    minReadySeconds: 10
13    strategy:
14      rollingUpdate:
15        maxSurge: 1
16        maxUnavailable: 0
17      type: RollingUpdate
18    template:
19      metadata:
20        labels:
21          environment: test
22      spec:
23        containers:
24          - image: nginx:1.16
25            name: nginx
```

- Deployment is a Kubernetes construct
- Need to abstract cloud specific construct in K8s construct
  - Enables K8s to run in multiple platform
- Say hello to Service Account

# Service Account



# OIDC



# Deployment Manifest

```
! nginx-deployment-withrolling.yaml
1  apiVersion: apps/v1
2  kind: Deployment
3  metadata:
4    labels:
5      | environment: test
6    name: testdeploy
7  spec:
8    replicas: 3
9    selector:
10   | matchLabels:
11     | environment: test
12   minReadySeconds: 10
13   strategy:
14     rollingUpdate:
15       maxSurge: 1
16       maxUnavailable: 0
17     type: RollingUpdate
18   template:
19     metadata:
20       labels:
21         | environment: test
22     spec:
23       containers:
24         - image: nginx:1.16
25           name: nginx
```

- When Service Account is not specified, default is used

```
AdminRole:~/environment $ kubectl get sa
NAME      SECRETS   AGE
default   1          17h
```

```
AdminRole:~/environment $ kubectl get sa -A
NAMESPACE   NAME      SECRETS   AGE
2048-game  alb-ingress-controller  1          15h
2048-game  default    1          15h
default    default    1          17h
kube-node-lease  default    1          17h
kube-public   default    1          17h
kube-system   alb-ingress-controller  1          16h
kube-system   attachdetach-controller 1          17h
kube-system   aws-cloud-provider    1          17h
kube-system   aws-node     1          17h
kube-system   certificate-controller 1          17h
kube-system   clusterrole-aggregation-controller 1          17h
kube-system   coredns     1          17h
kube-system   cronjob-controller  1          17h
kube-system   daemon-set-controller 1          17h
kube-system   default     1          17h
kube-system   deployment-controller 1          17h
kube-system   disruption-controller 1          17h
kube-system   endpoint-controller  1          17h
kube-system   expand-controller   1          17h
kube-system   generic-garbage-collector 1          17h
kube-system   horizontal-pod-autoscaler 1          17h
kube-system   job-controller    1          17h
kube-system   kube-proxy     1          17h
kube-system   namespace-controller 1          17h
kube-system   node-controller   1          17h
kube-system   persistent-volume-binder 1          17h
kube-system   pod-garbage-collector 1          17h
```

# Deployment Manifest

```
pods/pod-projected-svc-token.yaml

apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - image: nginx
    name: nginx
    volumeMounts:
    - mountPath: /var/run/secrets/tokens
      name: vault-token
  serviceAccountName: build-robot
```

# Deployment Manifest

```
apiVersion: apps/v1
kind: Deployment
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
  namespace: kube-system
spec:
  selector:
    matchLabels:
      app.kubernetes.io/name: alb-ingress-controller
  template:
    metadata:
      labels:
        app.kubernetes.io/name: alb-ingress-controller
    spec:
      containers:
        - name: alb-ingress-controller
          args:
            image: docker.io/amazon/aws-alb-ingress-controller:v1.1.4
          serviceAccountName: alb-ingress-controller
```

# Service Account IAM Mapping

```
AdminRole:~/environment $ kubectl describe sa alb-ingress-controller -n 2048-game
Name:           alb-ingress-controller
Namespace:      2048-game
Labels:         <none>
Annotations:   eks.amazonaws.com/role-arn: arn:aws:iam::708523690135:role/eksctl-fargate-cluster-addon-iamserviceac
cou-Role1-KYVVDL01VZSIM
Image pull secrets: <none>
Mountable secrets:  alb-ingress-controller-token-v4x29
Tokens:          alb-ingress-controller-token-v4x29
Events:          <none>
```

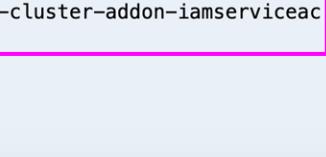


IAM Role

Visual editor    JSON

Import managed policy

```
1 {
2     "Version": "2012-10-17",
3     "Statement": [
4         {
5             "Effect": "Allow",
6             "Action": [
7                 "acm:DescribeCertificate",
8                 "acm>ListCertificates",
9                 "acm:GetCertificate"
10            ],
11            "Resource": "*"
12        },
13        {
14            "Effect": "Allow",
15            "Action": [
16                "ec2:AuthorizeSecurityGroupIngress",
17                "ec2>CreateSecurityGroup",
18                "ec2>CreateTags",
19                "ec2>DeleteTags",
20                "ec2>DeleteSecurityGroup",
21                "ec2:DescribeSecurityGroups",
22                "ec2:DescribeTags"
23            ],
24            "Resource": "*"
25        }
26    ]
27}
```



Policies attached to the Role

# Service Account and Cluster

- IAM Role is all about specific implementation (AWS in this case)
- Pods need Kubernetes Cluster access as well
- ClusterRoles grant access to cluster-specific resources
  - Create/Delete/List/etc. Nodes
  - Create/Delete/List/etc. Pods
  - Create/Delete/List/etc. Namespaces
  - etc.

# Service Account and Cluster

NAME	AGE
admin	17h
alb-ingress-controller	16h
aws-node	17h
cluster-admin	17h
edit	17h
eks:fargate-manager	17h
eks:node-bootstrapper	17h
eks:node-manager	17h
eks:podsecuritypolicy:privileged	17h
system:aggregate-to-admin	17h
system:aggregate-to-edit	17h
system:aggregate-to-view	17h
system:auth-delegator	17h
system:basic-user	17h
system:certificates.k8s.io:certificatesigningrequests:nodeclient	17h
system:certificates.k8s.io:certificatesigningrequests:selfnodeclient	17h
system:controller:attachdetach-controller	17h
system:controller:certificate-controller	17h
system:controller:clusterrole-aggregation-controller	17h
system:controller:cronjob-controller	17h
system:controller:daemon-set-controller	17h
system:controller:deployment-controller	17h
system:controller:disruption-controller	17h
system:controller:endpoint-controller	17h
system:controller:expand-controller	17h
system:controller:generic-garbage-collector	17h
system:controller:horizontal-pod-autoscaler	17h

# ClusterRole admin

```
AdminRole:~/environment $ kubectl describe clusterrole admin
```

Name:	admin	Non-Resource URLs	Resource Names	Verbs
Labels:	kubernetes.io/bootstrapping=rbac-defaults			
Annotations:	rbac.authorization.kubernetes.io/autoupdate: true			
PolicyRule:				
Resources				
rolebindings.rbac.authorization.k8s.io	[]	[]		[create delete deletecollection get list patch update watch]
roles.rbac.authorization.k8s.io	[]	[]		[create delete deletecollection get list patch update watch]
configmaps	[]	[]		[create delete deletecollection patch update get list watch]
endpoints	[]	[]		[create delete deletecollection patch update get list watch]
persistentvolumeclaims	[]	[]		[create delete deletecollection patch update get list watch]
pods	[]	[]		[create delete deletecollection patch update get list watch]
replicationcontrollers/scale	[]	[]		[create delete deletecollection patch update get list watch]
replicationcontrollers	[]	[]		[create delete deletecollection patch update get list watch]
services	[]	[]		[create delete deletecollection patch update get list watch]
daemonsets.apps	[]	[]		[create delete deletecollection patch update get list watch]
deployments.apps/scale	[]	[]		[create delete deletecollection patch update get list watch]
deployments.apps	[]	[]		[create delete deletecollection patch update get list watch]
replicasetss.apps/scale	[]	[]		[create delete deletecollection patch update get list watch]
replicasetss.apps	[]	[]		[create delete deletecollection patch update get list watch]
statefulsets.apps/scale	[]	[]		[create delete deletecollection patch update get list watch]
statefulsets.apps	[]	[]		[create delete deletecollection patch update get list watch]
horizontalpodautoscalers.autoscaling	[]	[]		[create delete deletecollection patch update get list watch]
cronjobs.batch	[]	[]		[create delete deletecollection patch update get list watch]
jobs.batch	[]	[]		[create delete deletecollection patch update get list watch]
daemonsets.extensions	[]	[]		[create delete deletecollection patch update get list watch]
deployments.extensions/scale	[]	[]		[create delete deletecollection patch update get list watch]
deployments.extensions	[]	[]		[create delete deletecollection patch update get list watch]

# ClusterRole admin contd..

daemonsets.extensions	[]	[]	[create delete deletecollection patch update get list watch]
deployments.extensions/scale	[]	[]	[create delete deletecollection patch update get list watch]
deployments.extensions	[]	[]	[create delete deletecollection patch update get list watch]
ingresses.extensions	[]	[]	[create delete deletecollection patch update get list watch]
networkpolicies.extensions	[]	[]	[create delete deletecollection patch update get list watch]
replicasets.extensions/scale	[]	[]	[create delete deletecollection patch update get list watch]
replicasets.extensions	[]	[]	[create delete deletecollection patch update get list watch]
replicationcontrollers.extensions/scale	[]	[]	[create delete deletecollection patch update get list watch]
ingresses.networking.k8s.io	[]	[]	[create delete deletecollection patch update get list watch]
networkpolicies.networking.k8s.io	[]	[]	[create delete deletecollection patch update get list watch]
poddisruptionbudgets.policy	[]	[]	[create delete deletecollection patch update get list watch]
deployments.apps/rollback	[]	[]	[create delete deletecollection patch update]
deployments.extensions/rollback	[]	[]	[create delete deletecollection patch update]
localsubjectaccessreviews.authorization.k8s.io	[]	[]	[create]
pods/attach	[]	[]	[get list watch create delete deletecollection patch update]
pods/exec	[]	[]	[get list watch create delete deletecollection patch update]
pods/portforward	[]	[]	[get list watch create delete deletecollection patch update]
pods/proxy	[]	[]	[get list watch create delete deletecollection patch update]
secrets	[]	[]	[get list watch create delete deletecollection patch update]
services/proxy	[]	[]	[get list watch create delete deletecollection patch update]
bindings	[]	[]	[get list watch]
events	[]	[]	[get list watch]
limitranges	[]	[]	[get list watch]
namespaces/status	[]	[]	[get list watch]
namespaces	[]	[]	[get list watch]
pods/log	[]	[]	[get list watch]
pods/status	[]	[]	[get list watch]
replicationcontrollers/status	[]	[]	[get list watch]
resourcequotas/status	[]	[]	[get list watch]
resourcequotas	[]	[]	[get list watch]

# User Defined ClusterRole

NAME	AGE
admin	17h
alb-ingress-controller	16h
aws-node	17h
cluster-admin	17h
edit	17h
eks:fargate-manager	17h
eks:node-bootstrapper	17h
eks:node-manager	17h
eks:podsecuritypolicy:privileged	17h
system:aggregate-to-admin	17h
system:aggregate-to-edit	17h
system:aggregate-to-view	17h
system:auth-delegator	17h
system:basic-user	17h
system:certificates.k8s.io:certificatesigningrequests:nodeclient	17h
system:certificates.k8s.io:certificatesigningrequests:selfnodeclient	17h
system:controller:attachdetach-controller	17h
system:controller:certificate-controller	17h
system:controller:clusterrole-aggregation-controller	17h
system:controller:cronjob-controller	17h
system:controller:daemon-set-controller	17h
system:controller:deployment-controller	17h
system:controller:disruption-controller	17h
system:controller:endpoint-controller	17h
system:controller:expand-controller	17h
system:controller:generic-garbage-collector	17h
system:controller:horizontal-pod-autoscaler	17h

# User Defined ClusterRole

```
AdminRole:~/environment $ kubectl describe clusterrole alb-ingress-controller
Name:      alb-ingress-controller
Labels:    app.kubernetes.io/name=alb-ingress-controller
Annotations: kubectl.kubernetes.io/last-applied-configuration:
            {"apiVersion":"rbac.authorization.k8s.io/v1","kind":"ClusterRole","metadata":{"annotations":{},"labels":{"app.kubernetes.io/name":"a
lb-ing...
PolicyRule:
  Resources          Non-Resource URLs  Resource Names  Verbs
  -----            -----           -----           -----
  services           []               []             [create get list update watch patch get list watch]
  services.extensions []              []             [create get list update watch patch get list watch]
  configmaps         []               []             [create get list update watch patch]
  endpoints          []               []             [create get list update watch patch]
  events             []               []             [create get list update watch patch]
  ingresses/status   []               []             [create get list update watch patch]
  ingresses          []               []             [create get list update watch patch]
  configmaps.extensions []            []             [create get list update watch patch]
  endpoints.extensions []            []             [create get list update watch patch]
  events.extensions  []            []             [create get list update watch patch]
  ingresses.extensions/status []        []             [create get list update watch patch]
  ingresses.extensions []            []             [create get list update watch patch]
  namespaces          []               []             [get list watch]
  nodes              []               []             [get list watch]
  pods               []               []             [get list watch]
  secrets             []               []             [get list watch]
  namespaces.extensions []            []             [get list watch]
  nodes.extensions   []            []             [get list watch]
  pods.extensions    []            []             [get list watch]
  secrets.extensions []            []             [get list watch]
```

# ClusterRole Manifest & ClusterRoleBinding

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
rules:
  - apiGroups:
    - ""
      - extensions
    resources:
      - configmaps
      - endpoints
      - events
      - ingresses
      - ingresses/status
      - services
    verbs:
      - create
      - get
      - list
      - update
      - watch
      - patch
  - apiGroups:
    - ""
      - extensions
    resources:
      - nodes
      - pods
      - secrets
      - services
      - namespaces
    verbs:
      - get
      - list
      - watch
  ---
```

Create ClusterRole

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: alb-ingress-controller
subjects:
  - kind: ServiceAccount
    name: alb-ingress-controller
    namespace: kube-system
  ---
```

Tie ClusterRole And  
Service Account  
ClusterRoleBinding

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
  namespace: kube-system
  ...
```

Create Service Account

# Bringing Them All Together!

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
rules:
  - apiGroups:
      - ""
      - extensions
      resources:
        - configmaps
        - endpoints
        - events
        - ingresses
        - ingresses/status
        - services
      verbs:
        - create
        - get
        - list
        - update
        - watch
        - patch
    - apiGroups:
        - ""
        - extensions
      resources:
        - nodes
        - pods
        - secrets
        - services
        - namespaces
      verbs:
        - get
        - list
        - watch
```

Create ClusterRole

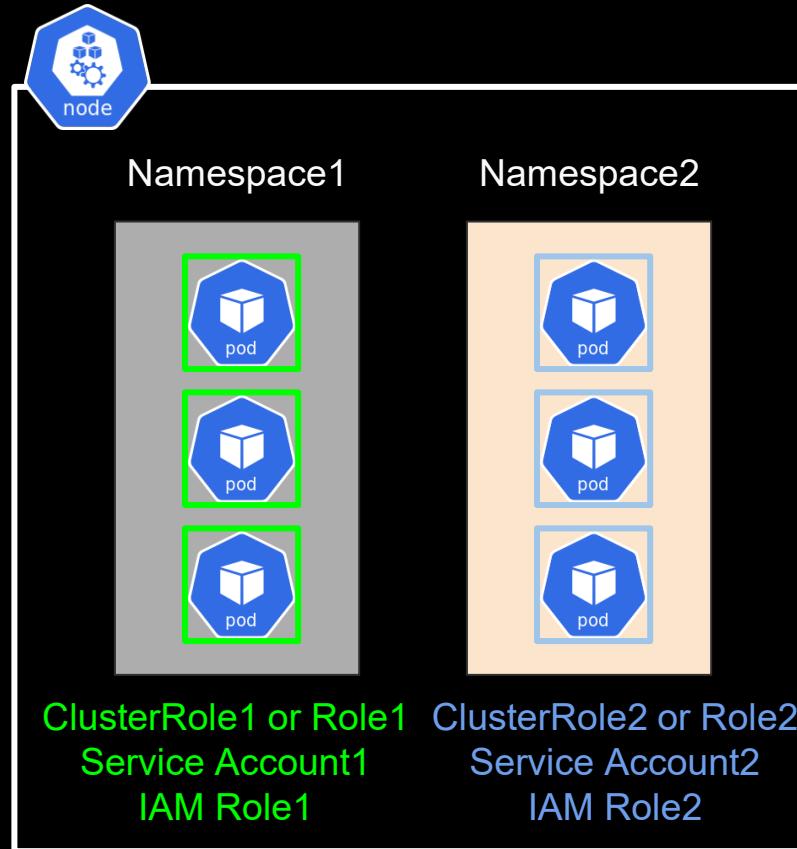
```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: alb-ingress-controller
subjects:
  - kind: ServiceAccount
    name: alb-ingress-controller
    namespace: 2048-game
```

Tie ClusterRole And Service Account  
ClusterRoleBinding

```
apiVersion: v1
kind: ServiceAccount
metadata:
  labels:
    app.kubernetes.io/name: alb-ingress-controller
  name: alb-ingress-controller
  namespace: 2048-game
```

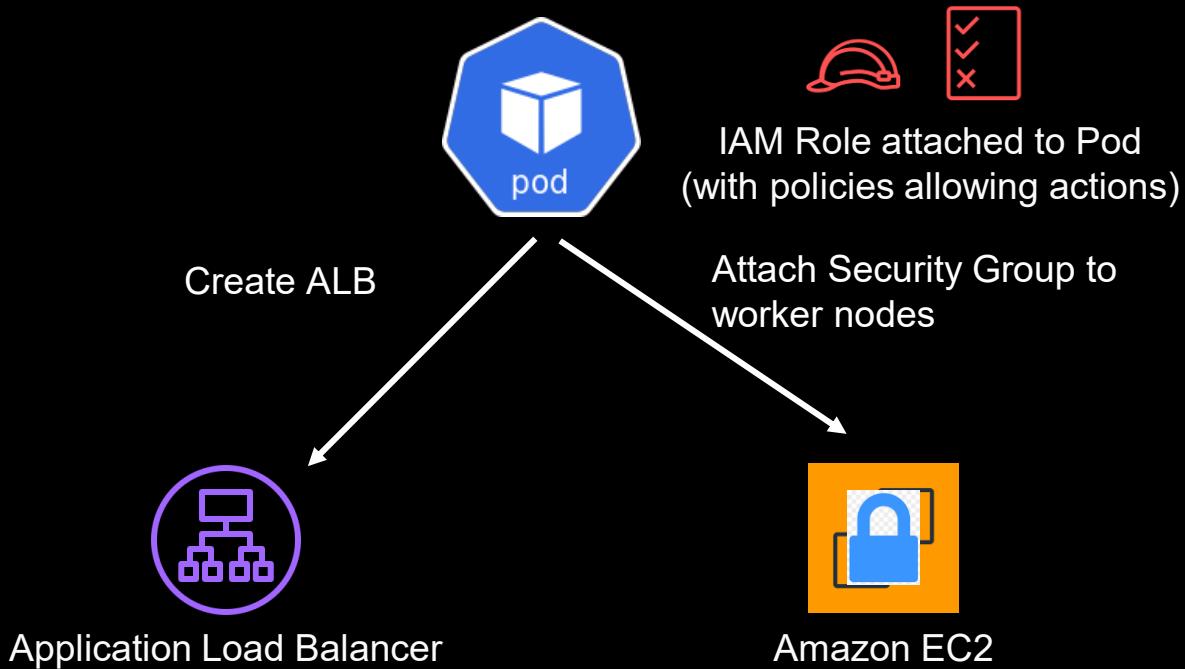
Create Service Account

# What does this buy us?

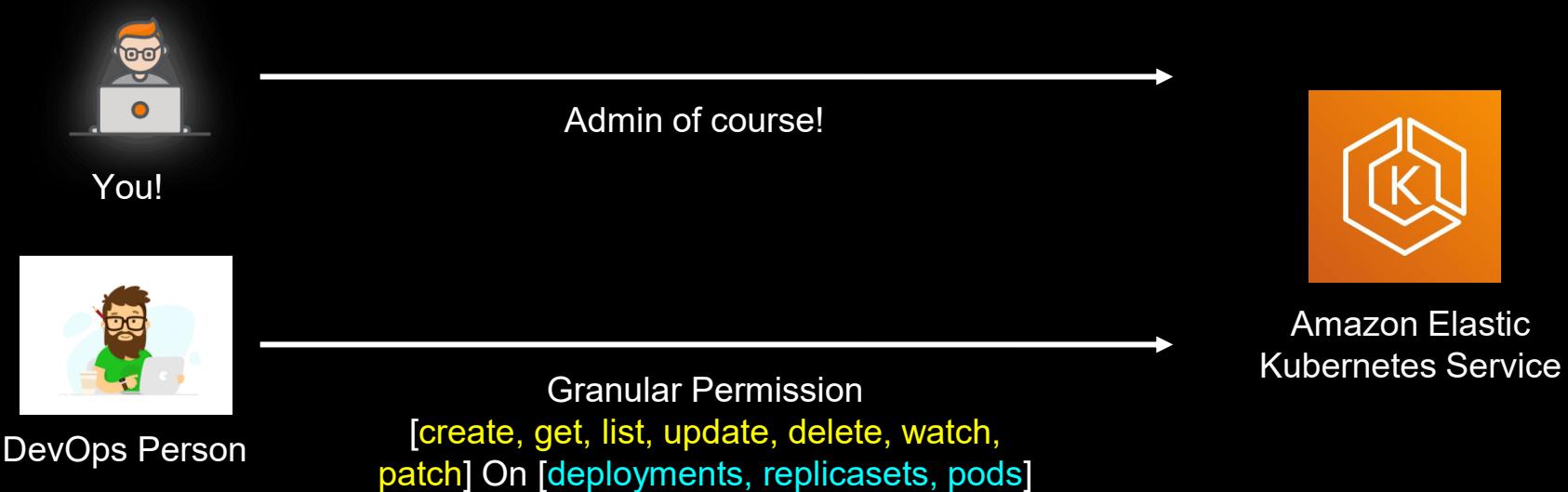


- Each application can have different access
- Node IAM Role doesn't have to have ALL access
- Secure and Granular Design
- Also known as IRSA (IAM Roles for Service Accounts) - replaces kube2iam

# Your App Running in Pod



# What About YOU?



# What About YOU?



You!



Admin of course!

- K8s has a predefined Role/Group for Admin (system:masters)
- Map AWS IAM username to K8s username and Group (Configmap)



Amazon Elastic  
Kubernetes Service

# ConfigMap/aws-auth

- Grant other AWS users permission to your cluster
- Grant AWS roles access to your cluster
- Works with RBAC
- Easier to understand with Demo!

# What About YOU?



You!

```
apiVersion: v1
data:
  mapUsers: |
    - userarn: arn:aws:iam::719217631821:user/developertina
      username: developertina
      groups:
        - system:masters
kind: ConfigMap
```

# What About DevOps Person?



DevOps Person



Granular Permission in Namespace  
[create, get, list, update, delete, watch,  
patch] On [deployments, replicaset, pods]

- Create K8s Role defining access to resources in namespace (Role)
- Map K8s username to Role (RoleBinding)
- Map AWS IAM username to K8s username and Group (Configmap/aws-auth)



Amazon Elastic  
Kubernetes Service

# Role Vs ClusterRole

- Both Role and ClusterRole represent set of permissions
- A Role sets permission within a particular namespace - have to specify namespace
- ClusterRole is non-namespaced

# What About DevOps Person?



DevOps Person

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: deployment-role
  namespace: frontend
rules:
  - apiGroups:
    - ""
    - extensions
    - apps
  resources:
    - deployments
    - replicasesets
    - pods
  verbs:
    - create
    - get
    - list
    - update
    - delete
    - watch
    - patch
```

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: deployment-rolebinding
  namespace: frontend
roleRef:
  apiGroup: ""
  kind: Role
  name: deployment-role
subjects:
  - kind: User
    name: developerbob
    apiGroup: ""
```

```
mapUsers: |
  - userarn: arn:aws:iam::719217631821:user/developerbob
    username: developerbob
    groups:
      - deployment-role
kind: ConfigMap
```

# Kubernetes Security

RBAC

IRSA

RoleBinding

ClusterRole

ClusterRoleBinding



# Kubernetes Security

RBAC

IRSA

RoleBinding



I can secure my EKS  
App Now!

ClusterRole

ClusterRoleBinding

# Cluster Authentication

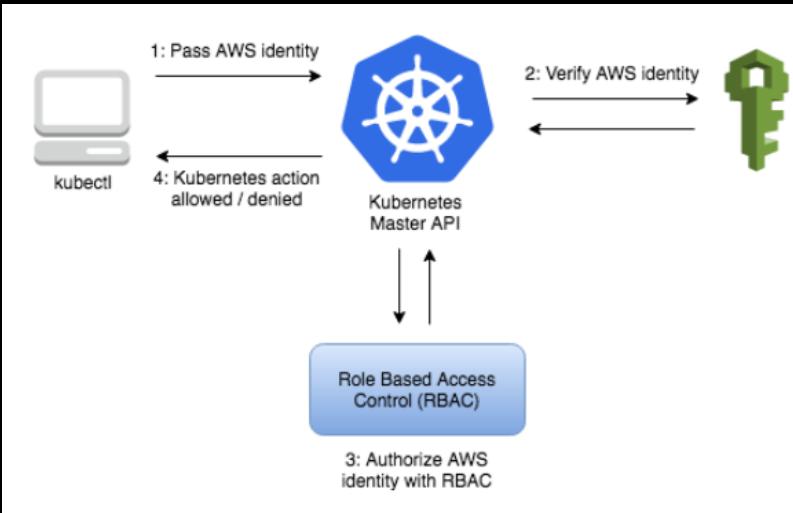


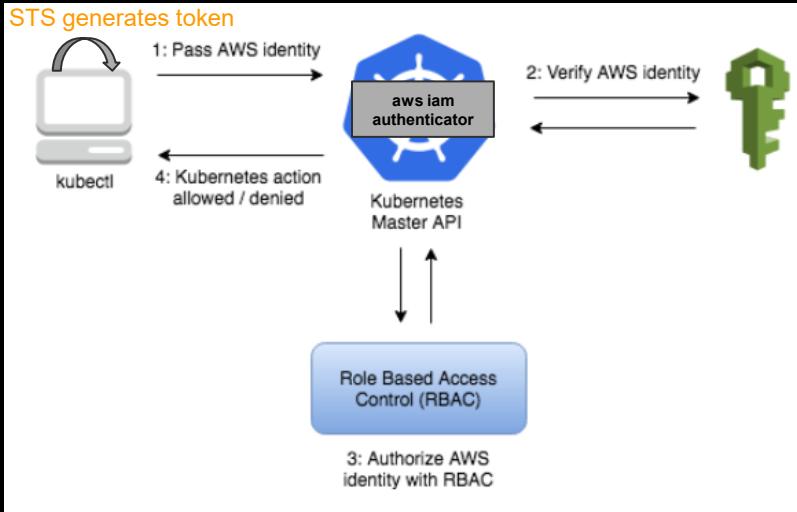
Image: <https://docs.aws.amazon.com/eks/latest/userguide/managing-auth.html>

# In The Early Days



- If Private Key is intercepted, cluster is vulnerable
- For each user, need to generate and maintain cert
- Management overhead

# AWS IAM Authenticator



- Works with AWS IAM to authenticate to K8s cluster
- No separate credential provider required
- Removes overhead of creating certs for new users
- Works well with IAM Roles and RBAC

# AWS IAM Authenticator - Kubeconfig

```
[+] creating EKS cluster "exciting-mongoose-1587093191" in "us-west-2" region with un-managed nodes
[+] will create 2 separate CloudFormation stacks for cluster itself and the initial nodegroup
[+] if you encounter any issues, check CloudFormation console or try 'eksctl utils describe-stacks --region=us-west-2 --cluster=exciting-mongoose-1587093191'
[+] CloudWatch logging will not be enabled for cluster "exciting-mongoose-1587093191" in "us-west-2"
[+] you can enable it with 'eksctl utils update-cluster-logging --region=us-west-2 --cluster=exciting-mongoose-1587093191'
[+] Kubernetes API endpoint access will use default of {publicAccess=true, privateAccess=false} for cluster "exciting-mongoose-1587093191" in "us-west-2"
[+] 2 sequential tasks: { create cluster control plane "exciting-mongoose-1587093191", create nodegroup "ng-ed09bddb" }
[+] building cluster stack "eksctl-exciting-mongoose-1587093191-cluster"
[+] deploying stack "eksctl-exciting-mongoose-1587093191-cluster"
[+] building nodegroup stack "eksctl-exciting-mongoose-1587093191-nodegroup-ng-ed09bddb"
[+] --nodes-min=2 was set automatically for nodegroup ng-ed09bddb
[+] --nodes-max=2 was set automatically for nodegroup ng-ed09bddb
[+] deploying stack "eksctl-exciting-mongoose-1587093191-nodegroup-ng-ed09bddb"
[+] all EKS cluster resources for "exciting-mongoose-1587093191" have been created
[+] saved kubeconfig as "C:\\\\Users\\\\Ashdeep/.kube/config"
[+] adding identity "arn:aws:iam::719217631821:role/eksctl-exciting-mongoose-15870931-NodeInstanceRole-A9PRSS2R2HWG" to auth ConfigMap
[+] nodegroup "ng-ed09bddb" has 0 node(s)
[+] waiting for at least 2 node(s) to become ready in "ng-ed09bddb"
[+] nodegroup "ng-ed09bddb" has 2 node(s)
[+] node "ip-192-168-38-172.us-west-2.compute.internal" is ready
[+] node "ip-192-168-92-221.us-west-2.compute.internal" is ready
[+] kubectl command should work with "C:\\\\Users\\\\Ashdeep/.kube/config", try 'kubectl get nodes'
[+] EKS cluster "exciting-mongoose-1587093191" in "us-west-2" region is ready
```

# Giving Access To Your EKS Cluster

- Give Admin Access to Other IAM Users for Your Cluster
- Give Role Based Granular Access to Other IAM Users for Your Cluster

# Giving Admin Access

```
kubectl edit -n kube-system configmap/aws-auth
```

```
# 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: |
    - groups:
      - system:bootstrappers
      - system:nodes
        rolearn: arn:aws:iam::719217631821:role/eksctl-eks-access-test-nodegroup-NodeInstanceRole-1U4X1KCH9BEMJ
        username: system:node:{{EC2PrivateDNSName}}
  mapUsers: |
    []
kind: ConfigMap
metadata:
  creationTimestamp: "2020-06-01T18:56:49Z"
  name: aws-auth
  namespace: kube-system
  resourceVersion: "753"
  selfLink: /api/v1/namespaces/kube-system/configmaps/aws-auth
  uid: 9b0d3aef-efe6-4325-bb59-5c64ea5bff24
```

# Giving Access To Your EKS Cluster

- Give Admin Access to Other IAM Users for Your Cluster
- Give Role Based Granular Access to Other IAM Users for Your Cluster

# PREREQUISITE - WATCH K8S SECURITY FIRST

RBAC

IRSA



I can secure my EKS  
App Now!

ClusterRole

ClusterRoleBinding

# KubeConfig

```
PS C:\UdemyVideos\k8_eks\resources> kubectl get pods
NAME             READY   STATUS    RESTARTS   AGE
test-845595ffc-9cztc   1/1     Running   0          94s
test-845595ffc-jslmw   1/1     Running   0          94s
test-845595ffc-k8cml   1/1     Running   0          94s
PS C:\UdemyVideos\k8_eks\resources> █
```

# KubeConfig

```
PS C:\UdemyVideos\k8_eks\resources\security> eksctl create cluster --name=udemy-ekscourse
[i]  eksctl version 0.20.0
[i]  using region us-west-2
[i]  setting availability zones to [us-west-2b us-west-2a us-west-2c]
```

```
[i]  deploying stack "eksctl-udemy-ekscourse-nodegroup-ng-81e0059a"
[i]  waiting for the control plane availability...
[✓]  saved kubeconfig as "C:\\Users\\Ashdeep\\.kube/config"
[i]  no tasks
[✓]  all EKS cluster resources for "udemy-ekscourse" have been created
```

\$HOME/.kube/config

# Life Without KubeConfig

```
kubectl get pods  
  --server my-eks-cluster:6443  
  --client-key admin.key  
  --client-certificate admin.crt  
  --certificate-authority-data <base64 encoded long string>
```

# KubeConfig File

## Clusters

dev-ekscluster  
stage-ekscluster  
prod-ekscluster

## Contexts

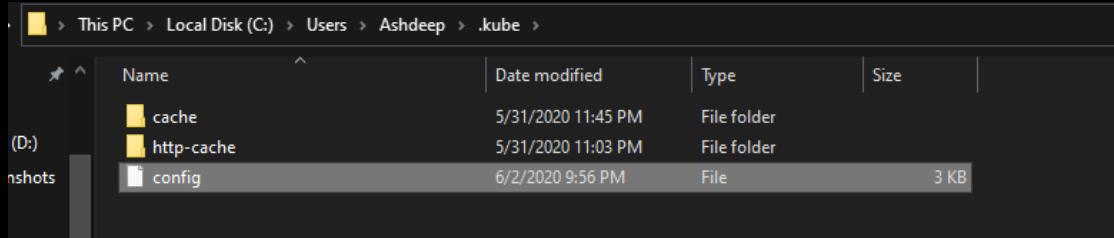
MegaAdmin@dev-ekscluster  
MegaAdmin@prod-ekscluster  
DeveloperGal@stage-ekscluster  
DevopsGuy@dev-ekscluster

## Users

MegaAdmin  
DevopsGuy  
DeveloperGal

current-context: MegaAdmin@dev-ekscluster

# A Real KubeConfig File



# A Real KubeConfig File

# AWS Shared Responsibility Model

Security of the cloud - Responsibility of AWS

- Kubernetes Control Plane
  - Control Plane Nodes
  - etcd Database

Security in the cloud - Responsibility of You

- Security configuration of Data Plane
  - Security groups for traffic between EKS control plane and VPC
- Configuration of worker nodes and containers
- Worker node operating system (updates, security patches)
- Network controls - firewall rules
- Platform level identity and access
- Data in worker nodes

# IAM for EKS

Security of the cloud - Responsibility of AWS

- Kubernetes Control Plane
  - Control Plane Nodes
  - etcd Database

Security in the cloud - Responsibility of You

- Security configuration of Data Plane
  - Security groups for traffic between EKS control plane and VPC
- Configuration of worker nodes and containers
- Worker node operating system (updates, security patches)
- Network controls - firewall rules
- Platform level identity and access
- Data in worker nodes

# EKS Fargate

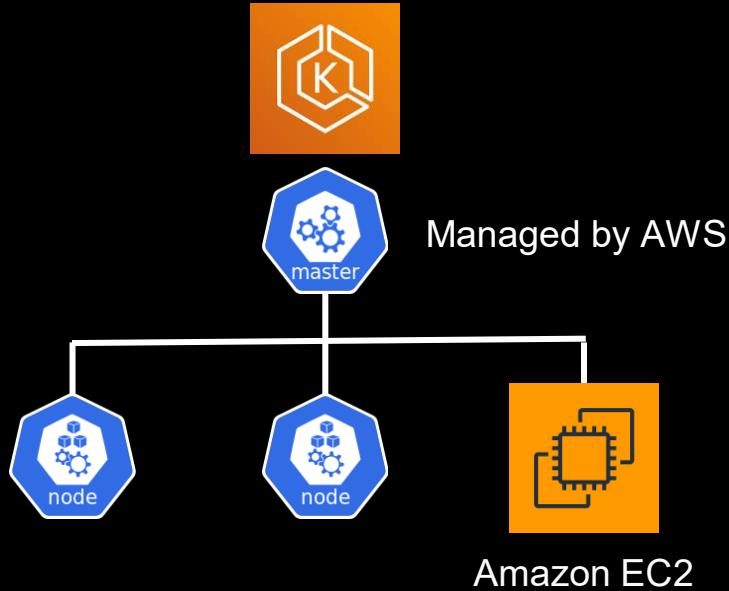


Amazon Elastic Container  
Service for Kubernetes



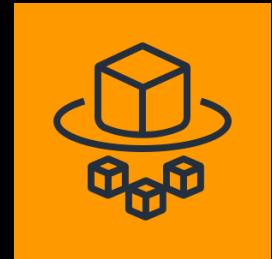
AWS Fargate

# Why?



- At the end of the day, still have overhead of EC2
  - AMI Rehydration
  - Patching, Scaling, Securing
  - Focus on cost optimization

# EKS FARGATE



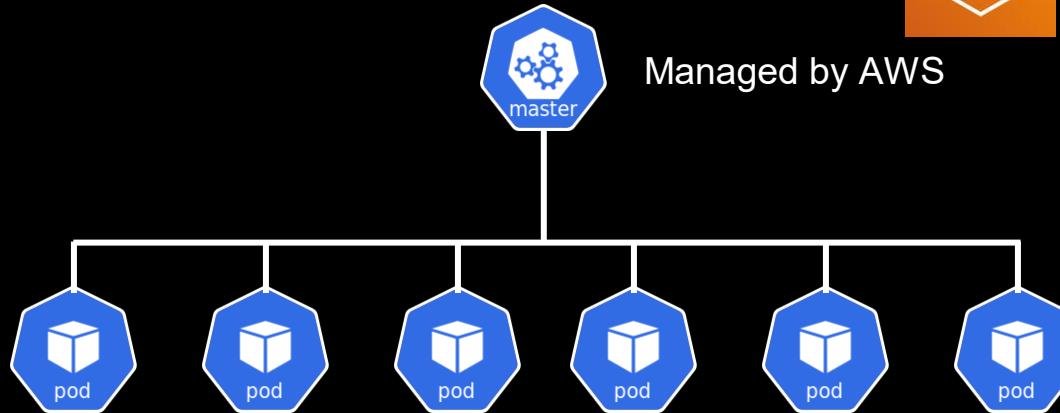
DEMO

VS

# EKS



# What is fargate



- Fargate
  - NO Worker Nodes to manage (Still require Control Plane)
  - Define and deploy your pods on Fargate
  - Possible to mix Fargate and regular EC2 based EKS cluster
  - HPA possible
  - Pay Control Plane Cost + (Pay per pod based on vCPU, Memory, Time)

## EKS Worker



Control Plane runs on EKS

Worker Nodes runs on EC2, user need to manage Nodes

Pods can be exposed using Services (Load Balancer) And Ingress

Daemonsets are supported and used heavily

Able to run stateful apps (using EFS)

Wide range of workload dependant EC2 selection, e.g GPU

Can work in public and private subnet

## EKS Fargate



Control Plane runs on EKS

No Worker Nodes required. Much less management overhead

Classic and NLBs not supported. Pods can be exposed using Ingress

Daemonsets are not supported. Need to run as Sidecar

Stateful apps not recommended

Can't select workload specific underlying hardware, no GPU. Max Pod size 4 vCPU and 30 Gb memory per pod

Only works in private subnet

Contd..

May 2020

## EKS Worker



HostPort or HostNetwork supported for Pods

CNI Custom Networking possible

Cost?

Control Plane + Worker EC2 Cost  
(Possibility to have more idle cost)

EKS Cluster with 2 m5.large EC2

= Control Plane Cost + Worker Nodes Cost

Control Plane Cost

= \$0.10/hour X 24 Hours X 30 Days

= \$72/Month

Worker Nodes Cost

= (\$0.096/hour X 24 Hours X 30 Days) X 2

= \$138.24/Month

Total Cost/Month

= \$72 + \$138.24 = **\$210.24/Month**

## EKS Fargate



HostPort or HostNetwork not supported for Pods

CNI Custom Networking not possible

Cost?

Control Plane + Cost to run Pods  
(Leaning towards pay per use model)

# EKS Fargate



Pod cost per vCPU per hour	= \$0.04048
Pod cost per GB memory per hour	= \$0.004445

CPU	Memory Values
0.25 vCPU	0.5GB, 1GB, and 2GB
0.5 vCPU	Min. 1GB and Max. 4GB, in 1GB increments
1 vCPU	Min. 2GB and Max. 8GB, in 1GB increments
2 vCPU	Min. 4GB and Max. 16GB, in 1GB increments
4 vCPU	Min. 8GB and Max. 30GB, in 1GB increments

Pricing is per second with 1 minute minimum

## Pricing Case 1

20 Pods, using 1vCPU, 2GB memory each, ran for 10 minutes, every day of month

= Control Plane Cost + Fargate Pods Cost

### Control Plane Cost

= \$0.10/hour X 24 Hours X 30 Days

= **\$72/Month**

### Fargate Pod vCPU Cost

= # of pods X # vCPUs X price per CPU-second X CPU duration per day in seconds X # of days

= 20 pods X (1 vCPU X (\$0.04048/(60 X 60) seconds) X 600 seconds X 30 days

= **\$4.04/Month**

### Fargate Pod memory Cost

= # of pods X # memory in GB X price per GB-second X memory duration per day in seconds X # of days

= 20 X 2 X (\$0.004445/(60X60)) X 600 X 30 days

= **\$0.889**

Total Cost/Month = \$72 + \$4.04 + \$0.889

= **\$76.92/Month**

# EKS Fargate



Pod cost per vCPU per hour	= \$0.04048
Pod cost per GB memory per hour	= \$0.004445

CPU	Memory Values
0.25 vCPU	0.5GB, 1GB, and 2GB
0.5 vCPU	Min. 1GB and Max. 4GB, in 1GB increments
1 vCPU	Min. 2GB and Max. 8GB, in 1GB increments
2 vCPU	Min. 4GB and Max. 16GB, in 1GB increments
4 vCPU	Min. 8GB and Max. 30GB, in 1GB increments

Pricing is per second with 1 minute minimum

## Pricing Case 2

20 Pods, using 1vCPU, 2GB memory each, ran for 12 hours, **every day of month**

= Control Plane Cost + Fargate Pods Cost

### Control Plane Cost

= \$0.10/hour X 24 Hours X 30 Days

= **\$72/Month**

### Fargate Pod vCPU Cost

= # of pods X # vCPUs X price per CPU-second X CPU duration per day in seconds X # of days

= 20 pods X (1 vCPU X (\$0.04048/(60 X 60) seconds) X 43200 seconds X 30 days

= **\$291.45/Month**

### Fargate Pod memory Cost

= # of pods X # memory in GB X price per GB-second X memory duration per day in seconds X # of days

= 20 X 2 X (\$0.004445/(60X60)) X 43200 X 30 days

= **\$64.00**

Total Cost/Month = \$72 + \$291.45 + \$64.00

= **\$427.45/Month**



**“We don’t believe in one tool to rule the world. We want you to use the right tool for the right job.”—Andy Jassy, CEO of AWS**

# EKS Fargate Scaling

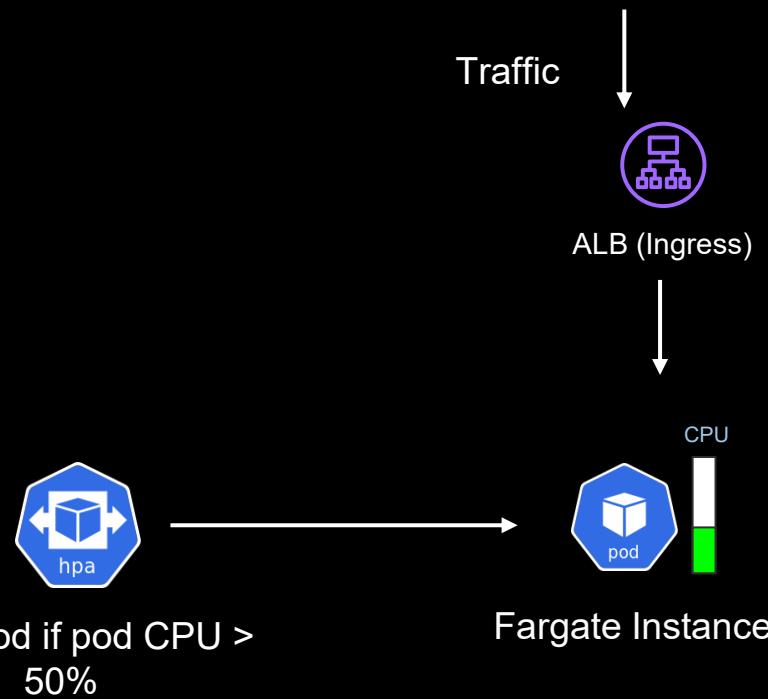


Amazon Elastic Container  
Service for Kubernetes

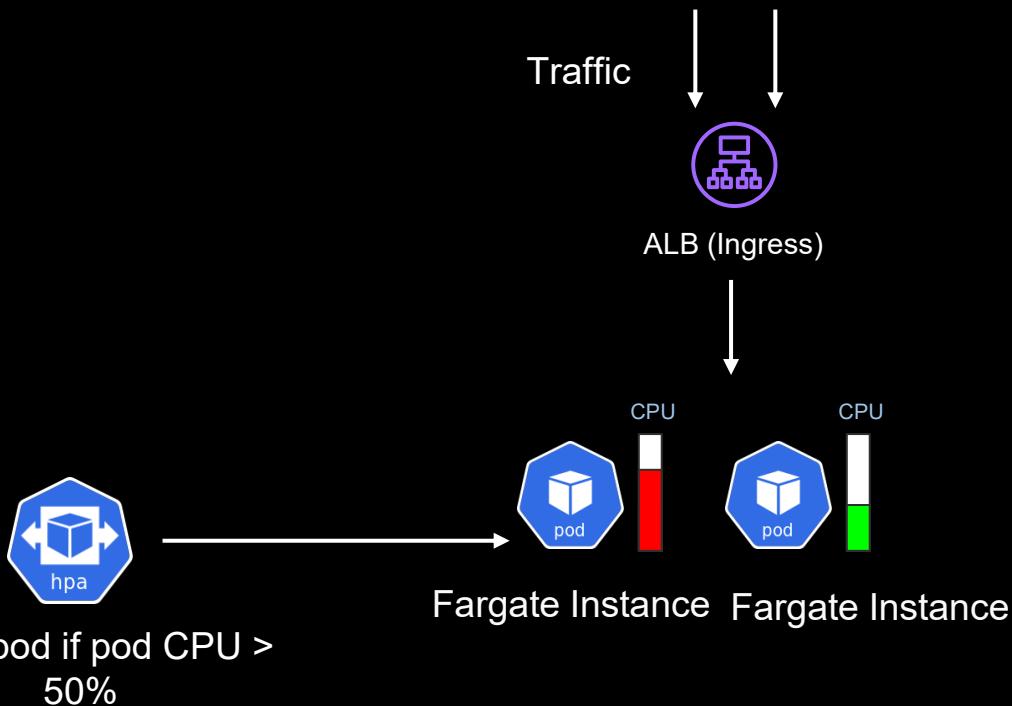


AWS Fargate

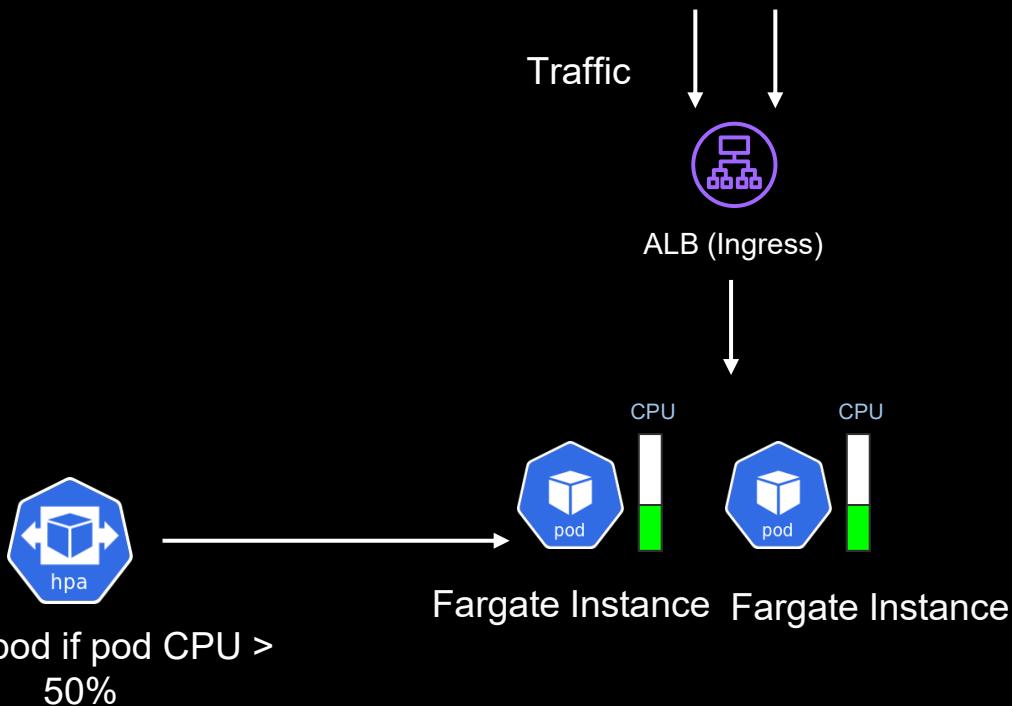
# Horizontal Pod Autoscaler (HPA)



# Horizontal Pod Autoscaler (HPA)

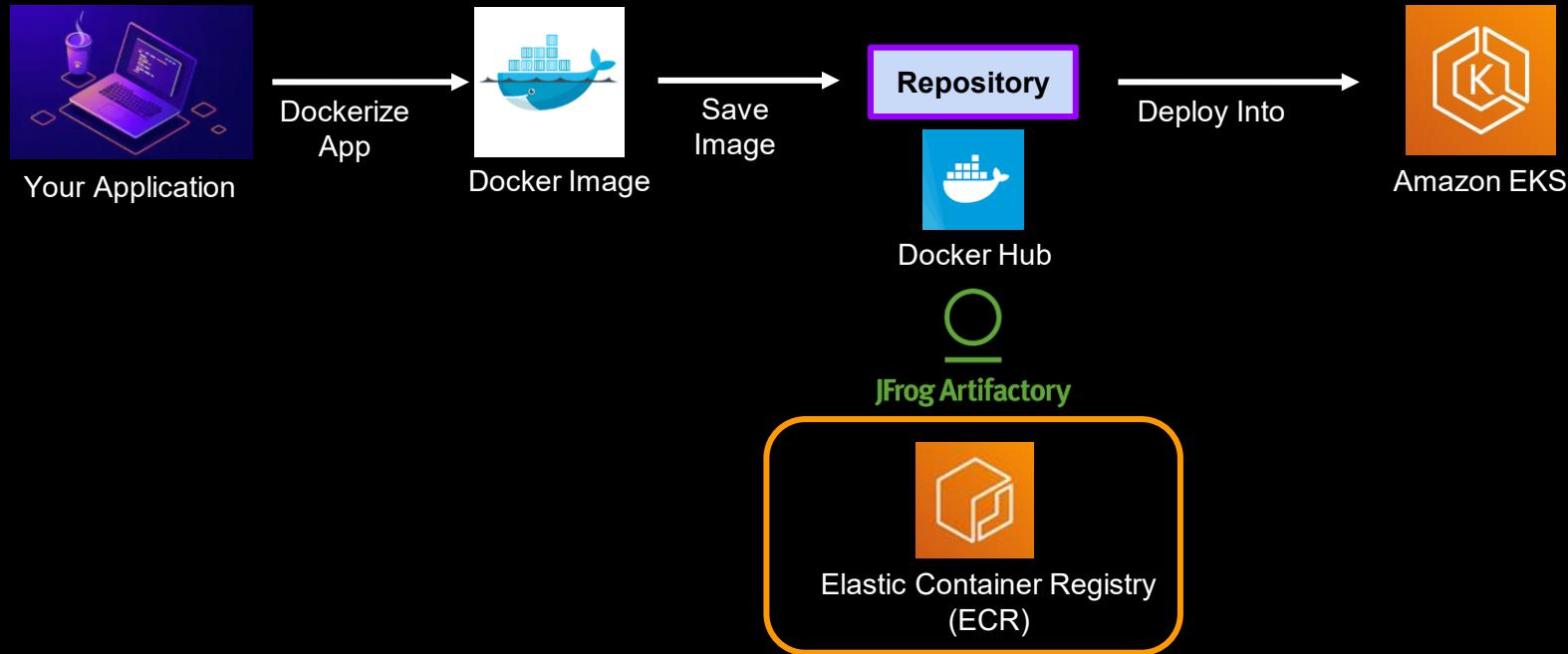


# Horizontal Pod Autoscaler (HPA)



# EKS DevOps

# ECR - In Big Picture



# ECR - What And Why

- Docker Container Registry to store, manage, and deploy container images
- Fully Managed And Scalable
- Secure
- Highly Available
- Simplified Workflow
- No Upfront Fee Or Commitments

# How ECR Works

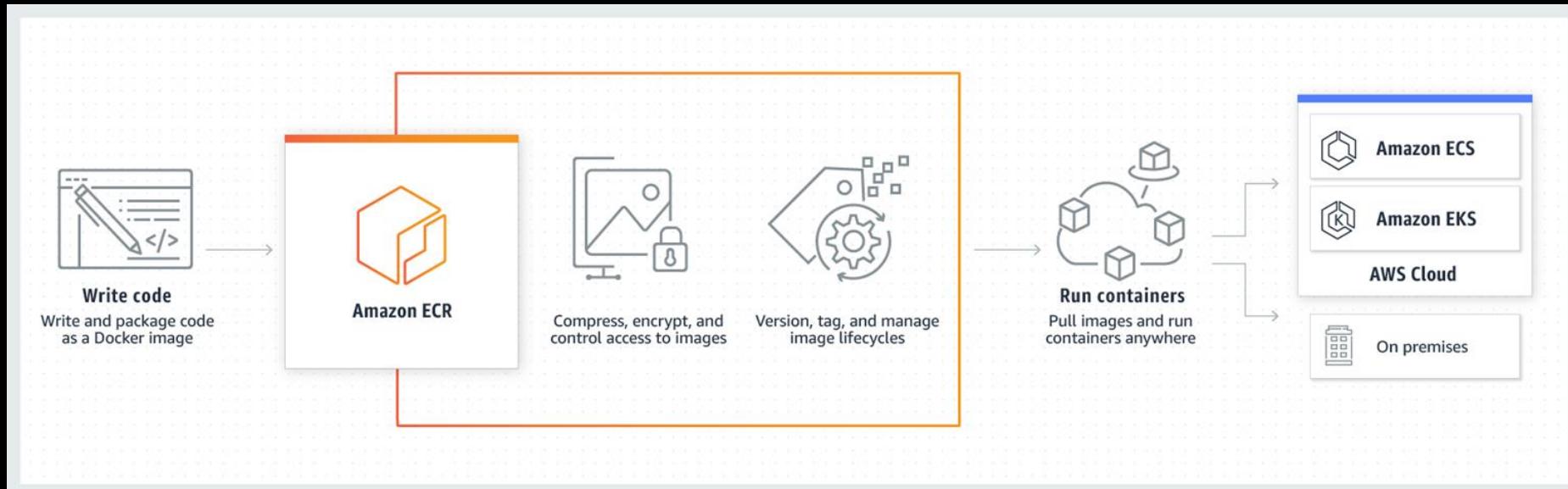
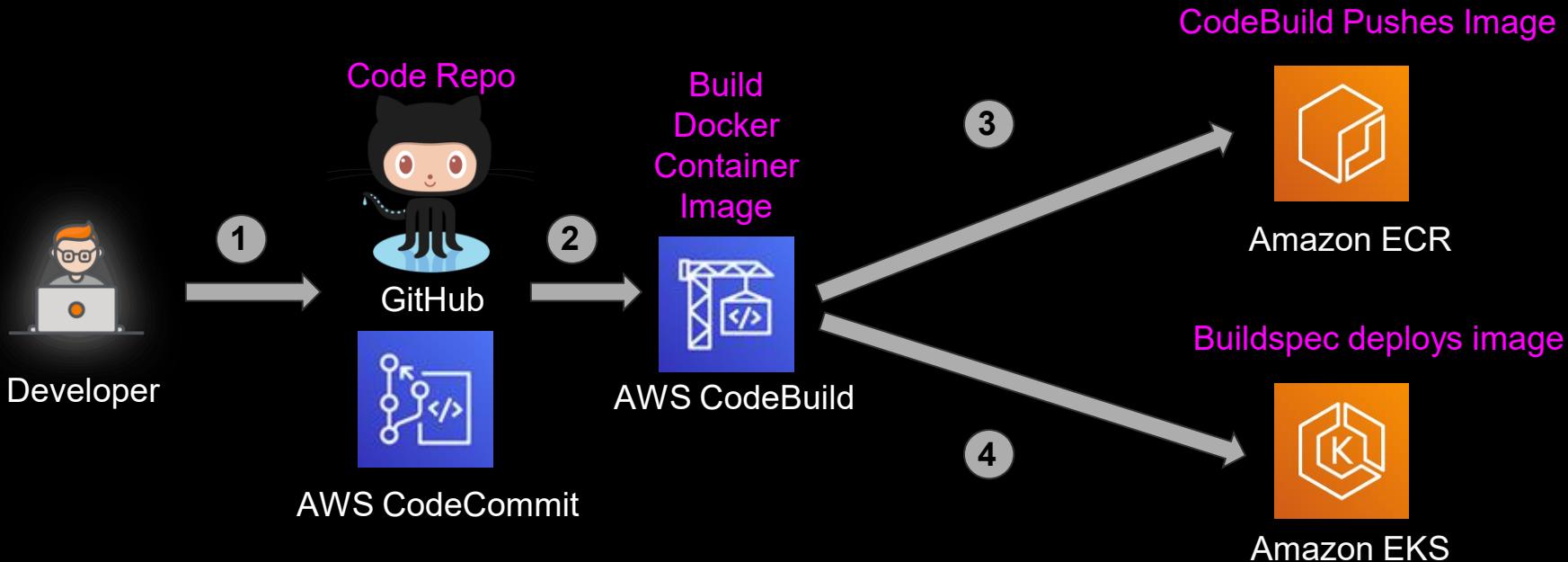


Image: <https://aws.amazon.com/ecr/>

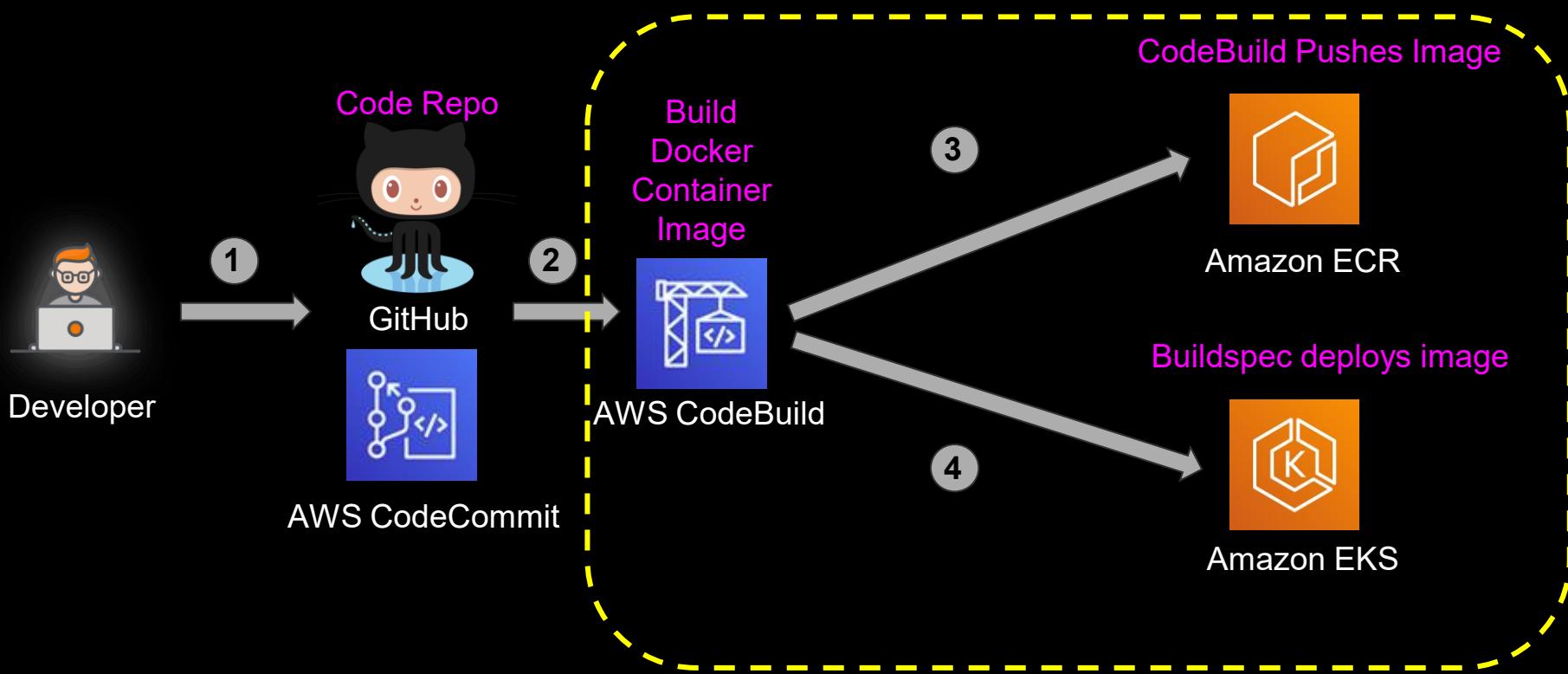
# DEMO TIME!

- Dockerize Sample App
- Save image in ECR
- Deploy in EKS

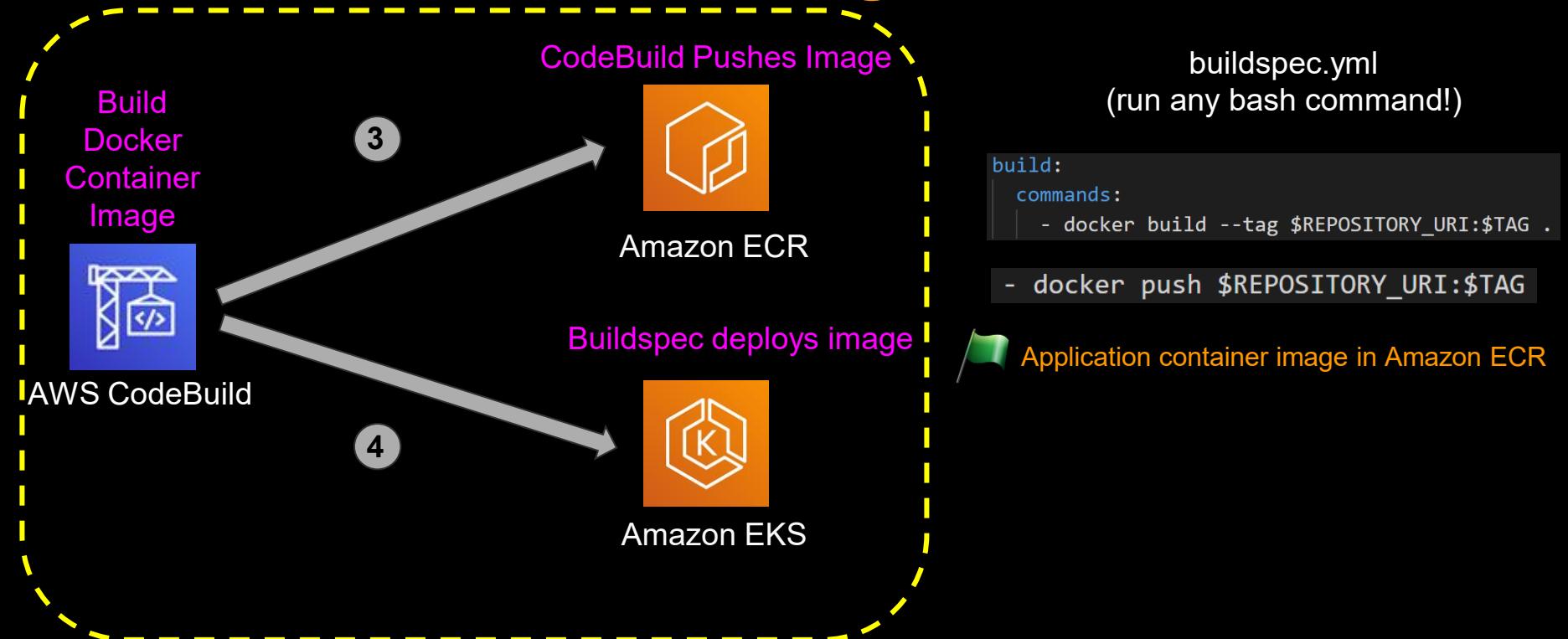
# CI/CD Flow-1 for EKS



# CI/CD Flow-1 for EKS



# Understanding Buildspec



# Understanding Buildspec Flow

hello-k8s.yml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: hello-k8s
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxUnavailable: 2
      maxSurge: 2
  selector:
    matchLabels:
      app: hello-k8s
  template:
    metadata:
      labels:
        app: hello-k8s
    spec:
      containers:
        - name: hello-k8s
          image: CONTAINER_IMAGE
          securityContext:
            privileged: false
            readOnlyRootFilesystem: true
            allowPrivilegeEscalation: false
          ports:
            - containerPort: 8080
```

buildspec.yml  
(run any bash command!)

```
build:
  commands:
    - docker build --tag $REPOSITORY_URI:$TAG .
    - docker push $REPOSITORY_URI:$TAG
```



Application container image in Amazon ECR

```
- sed -i 's@CONTAINER_IMAGE@'$REPOSITORY_URI:$TAG'@' hello-k8s.yml
```

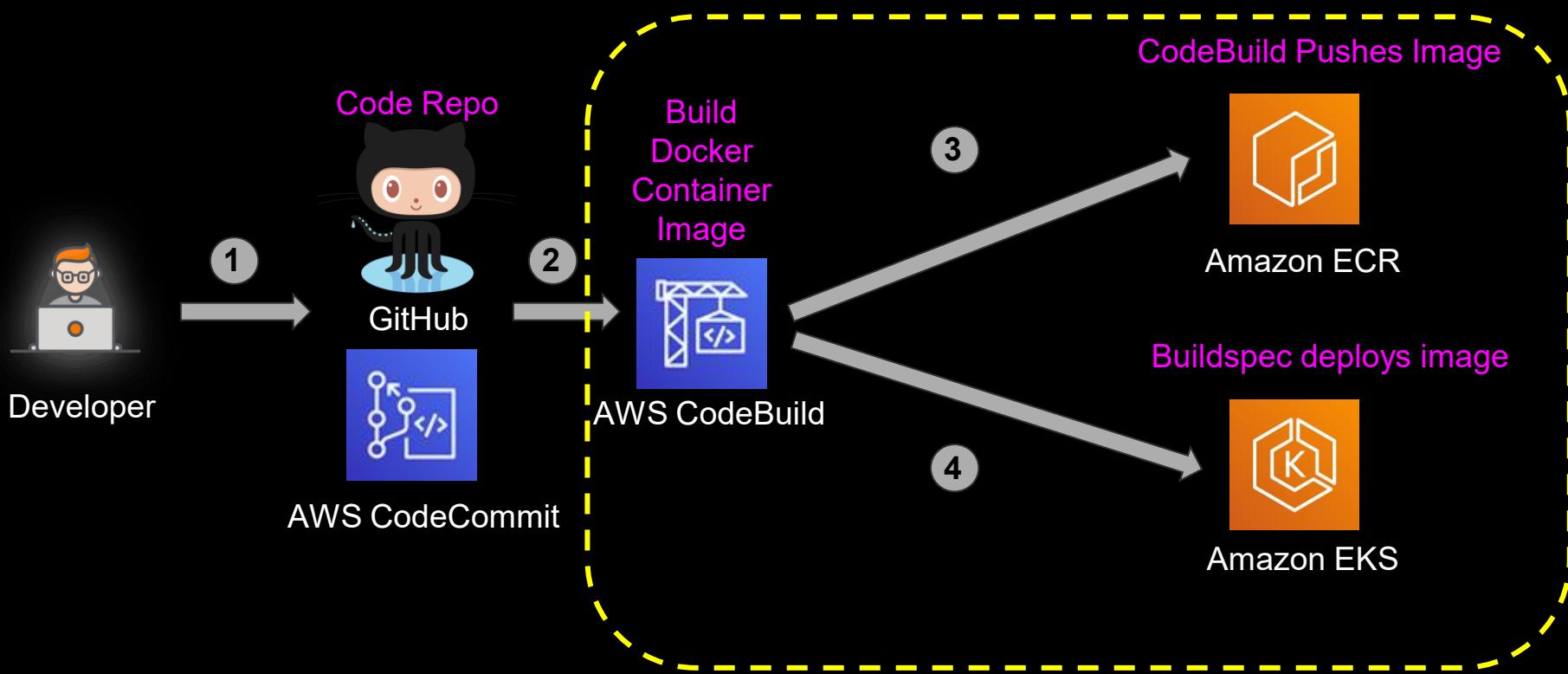
(replace CONTAINER\_IMAGE with actual image with tags)

```
- kubectl apply -f hello-k8s.yml
```

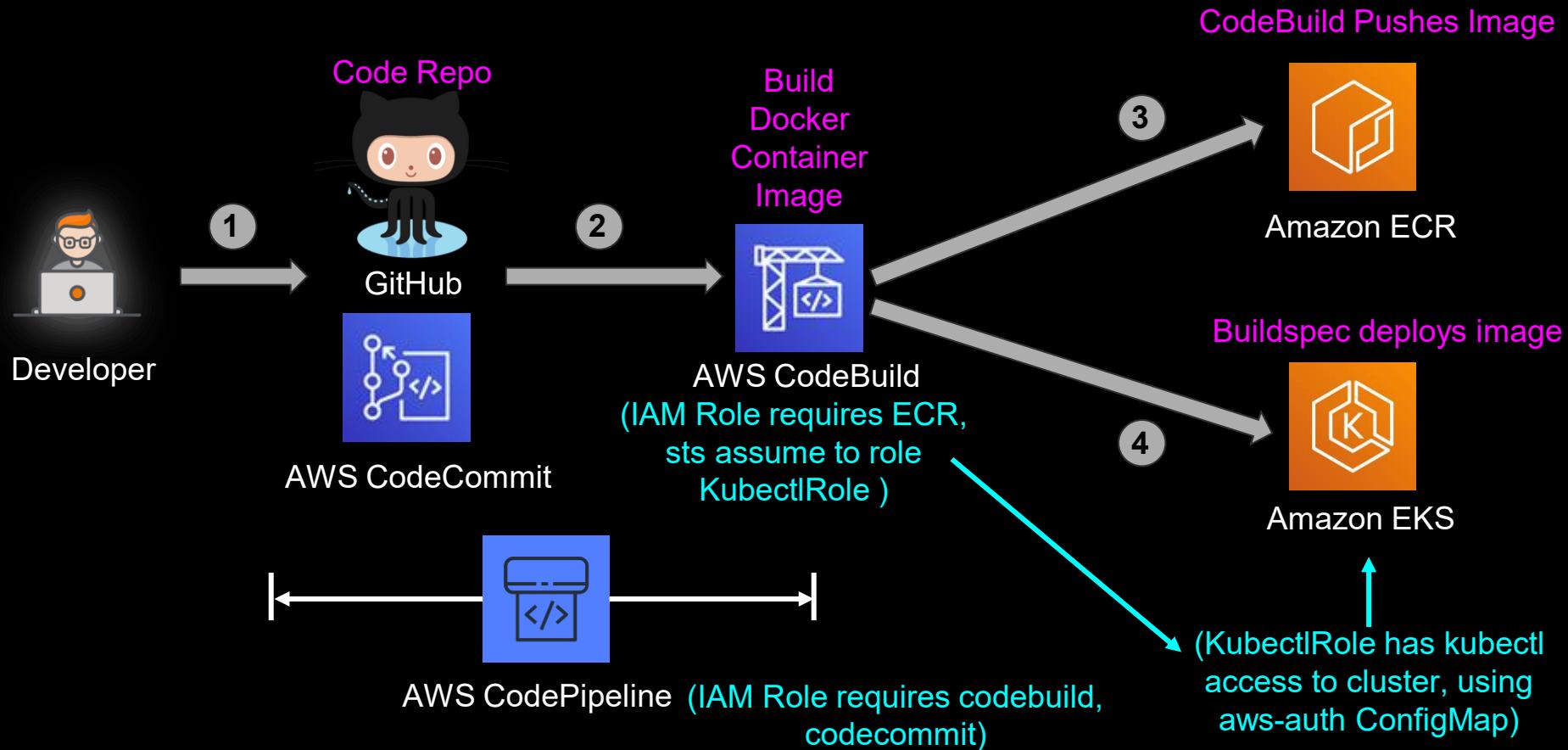


New Image deployed into EKS

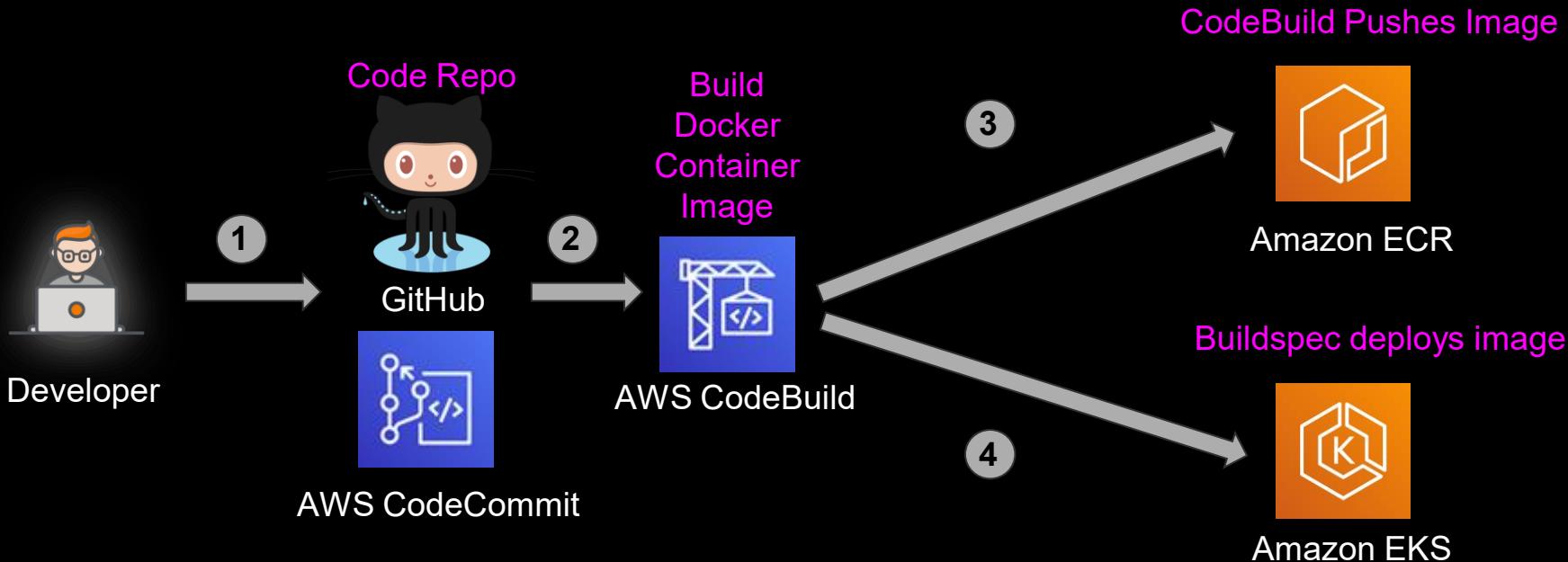
# CI/CD Flow-1 for EKS



# The Mundane (Admin) Part!



# CI/CD Flow-1 for EKS DEMO



# CI/CD Flow-1 for EKS DEMO

## Part 1 - IAM and Setup

- Create IAM Roles for
  - CodePipeline
  - CodeBuild
  - Kubectl role
- Edit configmap to give access to Kubectl role

## Part 2 - Pipeline

- Create CodePipeline pipeline
- Discuss relevant files
  - Dockerfile
  - Buildspec
- Run the pipeline

# CI/CD Flow-1 for EKS DEMO

## Part 1 - IAM and Setup

- Create IAM Roles for
  - CodePipeline
  - CodeBuild
  - Kubectl role
- Edit configmap to give access to Kubectl role

## Part 2 - Pipeline

- Create CodePipeline pipeline
- Discuss relevant files
  - Dockerfile
  - Buildspec
- Run the pipeline

# CI/CD Flow-1 for EKS DEMO

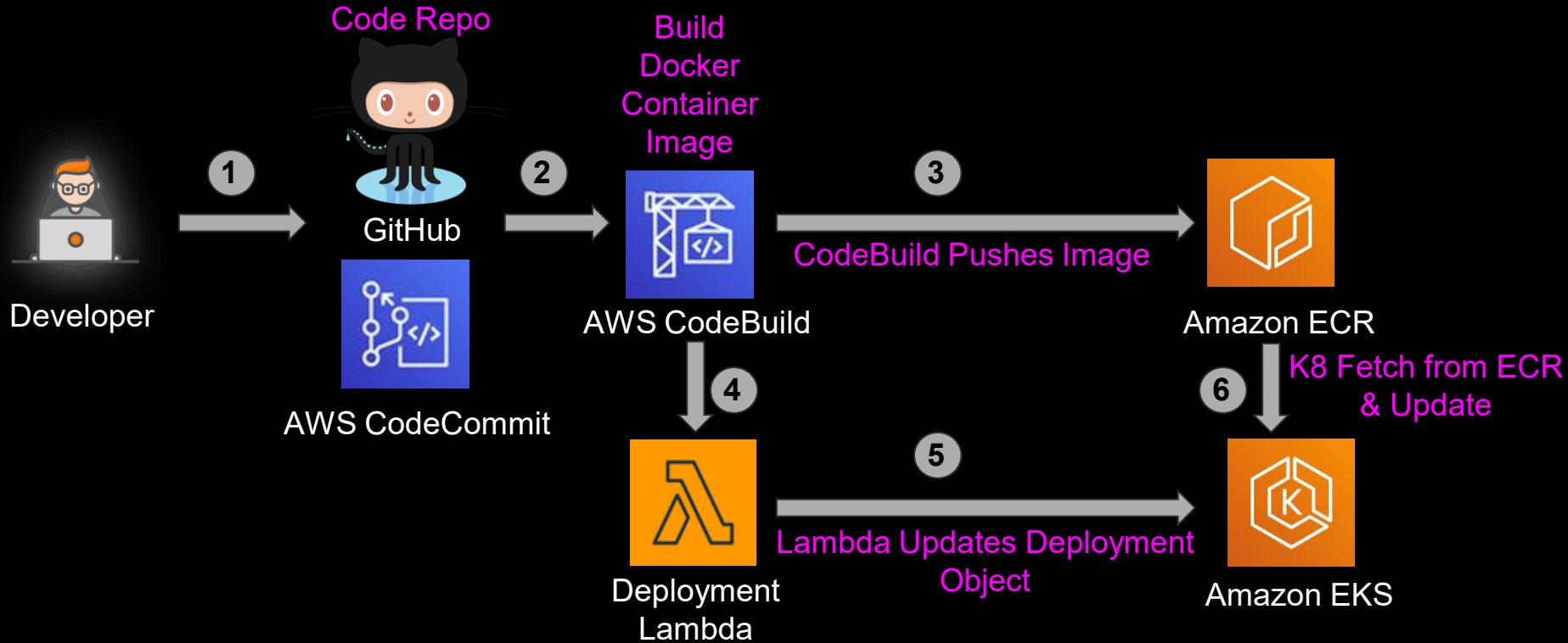
## Part 1 - IAM and Setup

- Create IAM Roles for
  - CodePipeline
  - CodeBuild
  - Kubectl role
- Edit configmap to give access to Kubectl role

## Part 2 - Pipeline

- Create CodePipeline pipeline
- Discuss relevant files
  - Dockerfile
  - Buildspec
- Run the pipeline

# CI/CD Flow-2 for EKS



# Real World Stateless App

# Guest Book Application

# The Big Picture

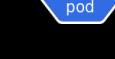
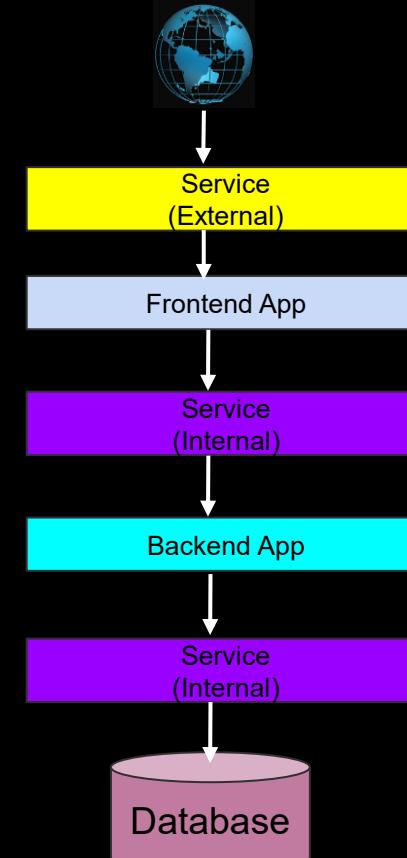
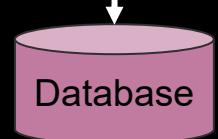
Think about the concepts and not a specific solution!

Traffic From Internet



Frontend App

Backend App



LoadBalancer

Container for Frontend App

ClusterIP

Container for backend App

ClusterIP

Database Containers

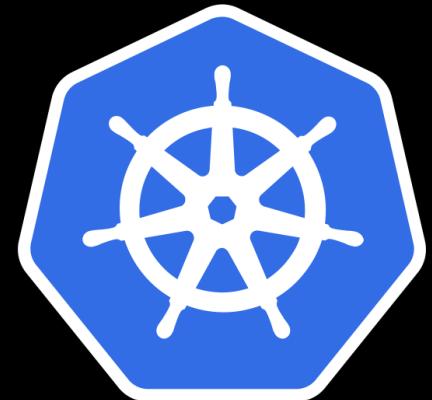
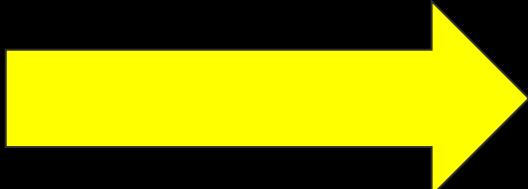


**FRONTEND**

**BACKEND**

**DATABASE**

# HOW ??!!



# The Big Picture

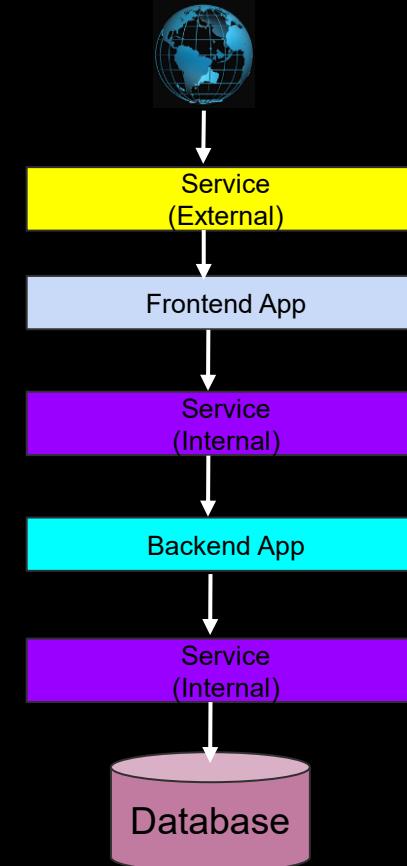
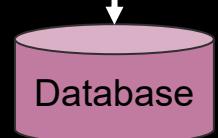
Think about the concepts and not a specific solution!

Traffic From Internet



Frontend App

Backend App



LoadBalancer



Container for Frontend App



ClusterIP



Container for backend App



ClusterIP



Database Containers