Getting Started with Kubernetes on AWS

Brought to you by the AWS Cloud Support Team

Day 2

Agenda

- What have we learnt?
- Labels
- Controllers (Deployment)
- Services
- Divide your Environment into different Namespaces
- Exposing your Application using ALB Ingress Controller
- Scale your Application with Horizontal Pod Auto Scaler

Firstly...

Do you have a cluster?

Using a terminal in Cloud9, verify there are Worker Nodes in your cluster

Do you have a cluster?

Using a terminal in Cloud9, verify there are Worker Nodes in your cluster

Ooops, no, I don't have a cluster!

https://github.com/aws-vls-dub/eks

```
eksctl create cluster --version 1.16 --node-type t3.medium --name eks
```

Review

Questions:

Review

Questions:

- What are some advantages of using containers?

Review

Questions:

- What are some advantages of using containers?
- What's the most common runtime environment for containers?

Review

Questions:

- What are some advantages of using containers?
- What's the most common runtime environment for containers?
- If we have Docker why we need something like Kubernetes?

Review

Questions:

- What are some advantages of using containers?
- What's the most common runtime environment for containers?
- If we have Docker why we need something like Kubernetes?
- True or False - Is Kubernetes open source?

Review

Questions:

- What are some advantages of using containers?
- What's the most common runtime environment for containers?
- If we have Docker why we need something like Kubernetes?
- True or False - Is Kubernetes open source?
- Multiple Choice - Which are components of a Kubernetes Master?
  - API Server
  - Scheduler
  - Kubelet
  - Cloud Controller Manager
  - Garbage Collector

Review

Questions:

Review

Questions:

- How many IP addresses can be assigned to a single pod?

Review

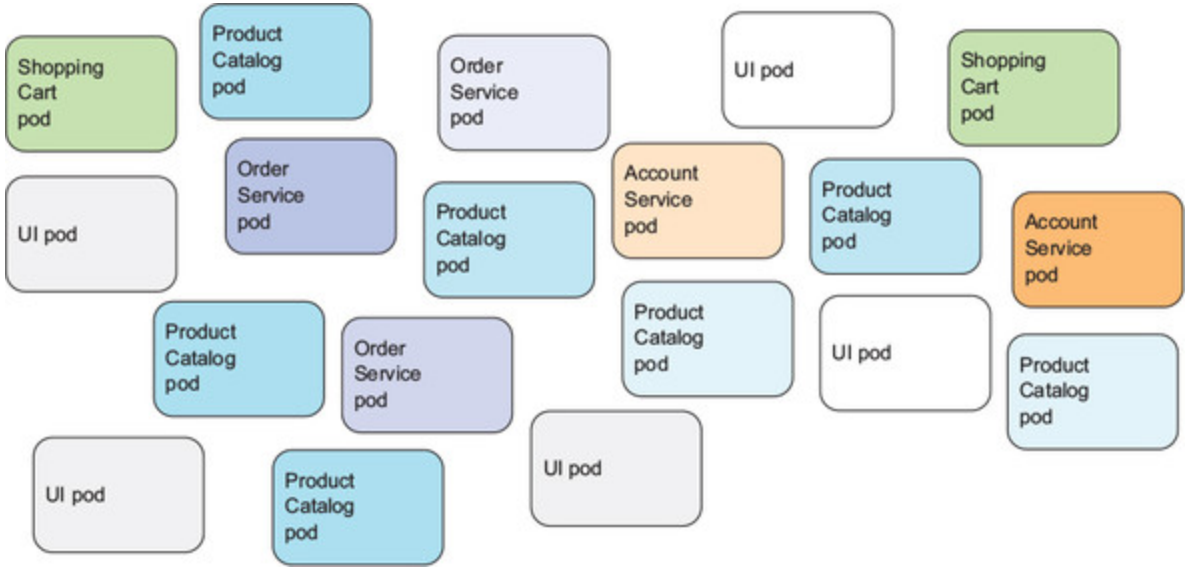
Questions:

- How many IP addresses can be assigned to a single pod?
- Who's your best buddy when you need to talk to Kubernetes?

Are you ready ?

Labels

What if we are running a lot of pods?

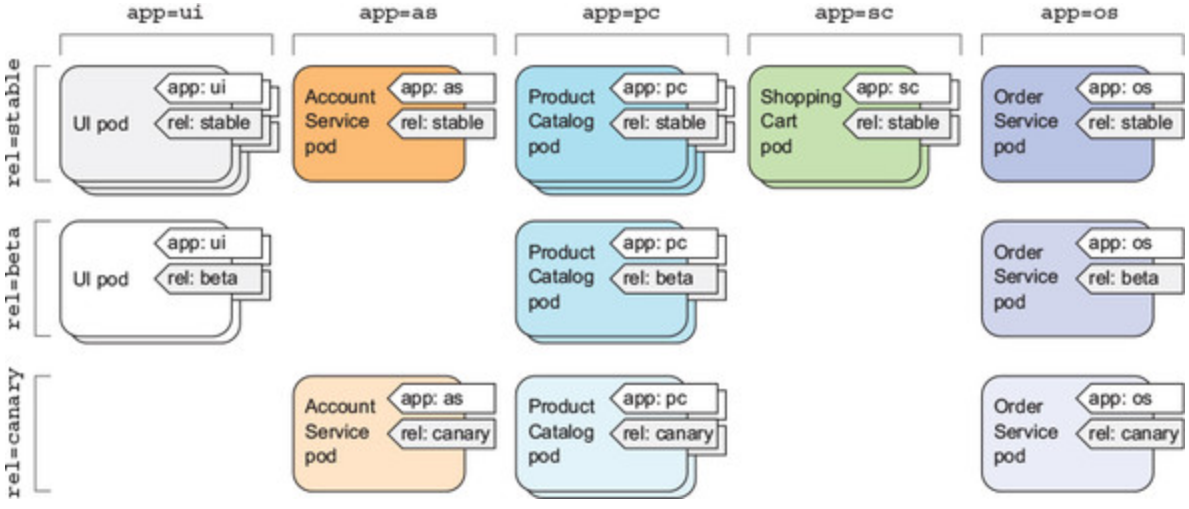


Picture from Kubernetes in action

Labels

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

Can be used to query and organize resources



Picture from Kubernetes in action

Lab 4: Applying labels

Labels in the object definition

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

Checking the labels

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

kubectl get pods --show-labels
```

File: labs/03_labels.yaml

Let's kubectl

Lab 4: Labeling our Pods

https://github.com/aws-vls-dub/eks/tree/master/labs/04-labels

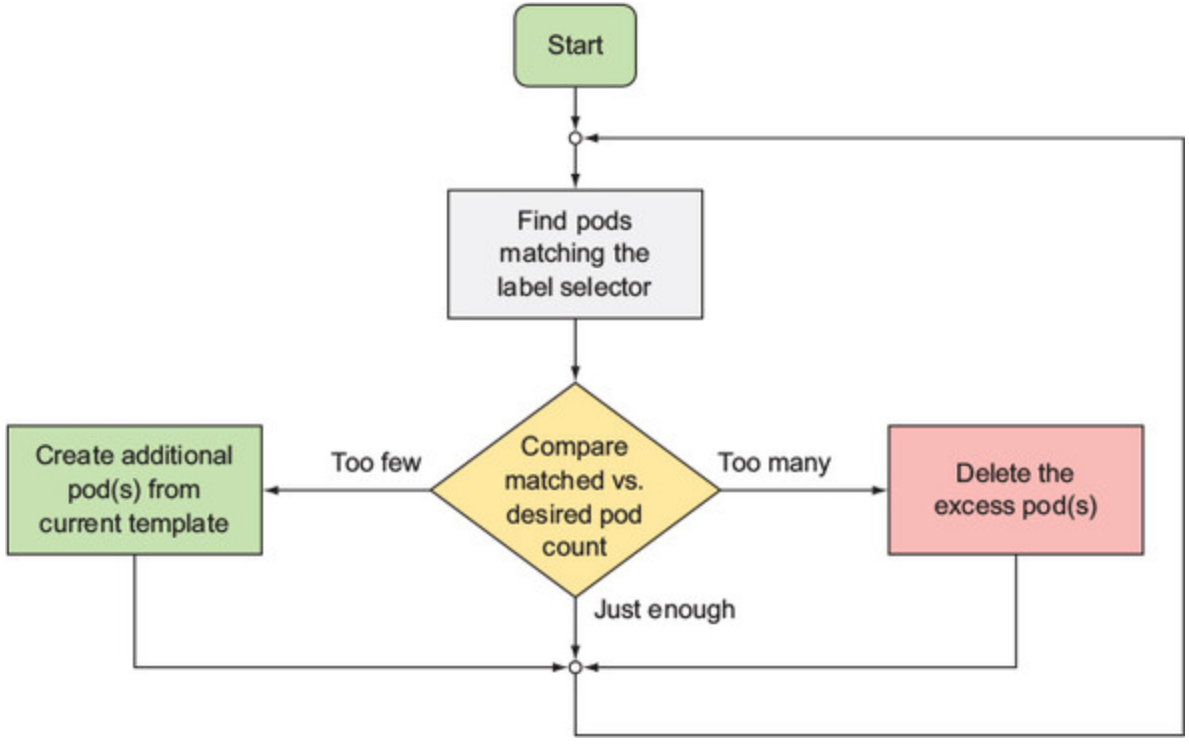
Questions?

Controllers

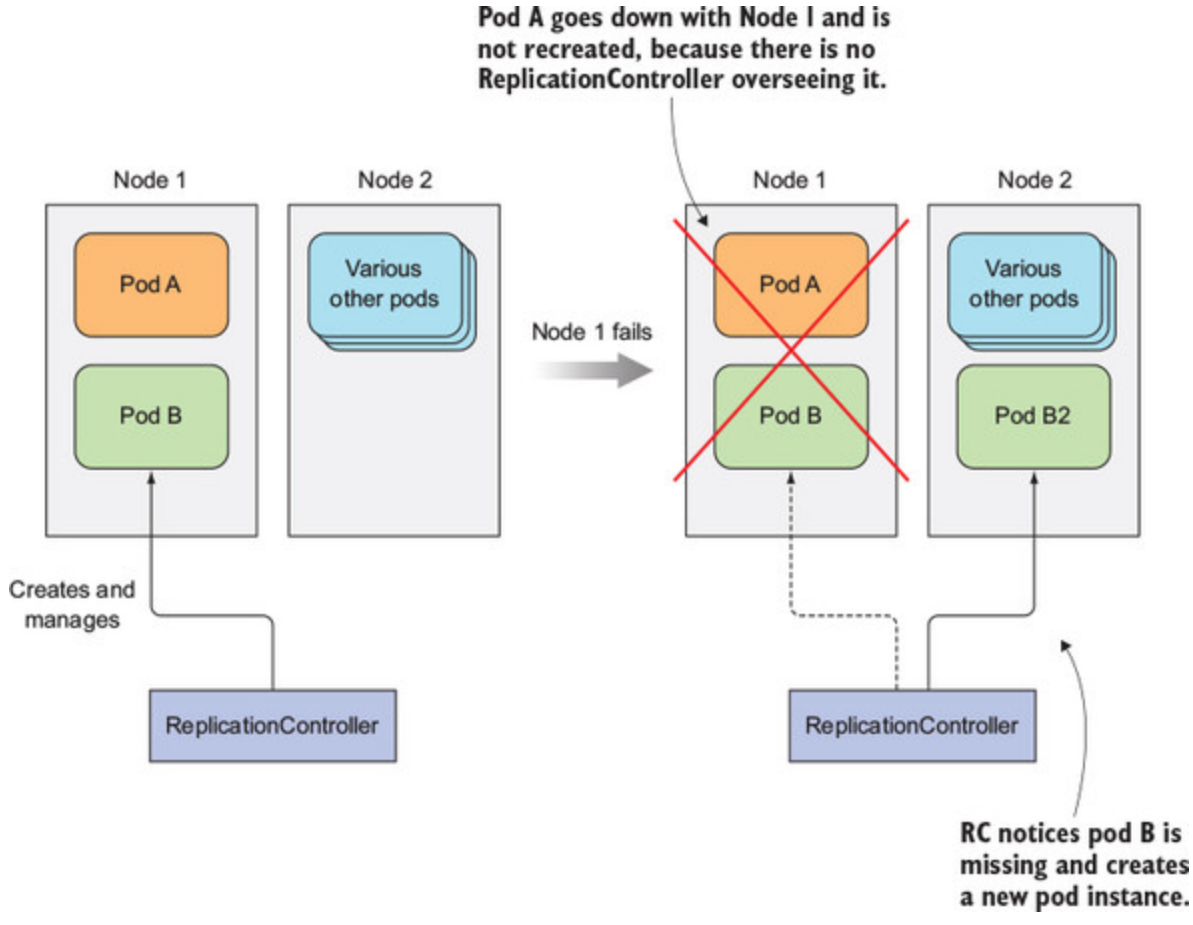
What controllers are and what they do?

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

What controllers are and what they do?



What controllers are and what they do?



Controllers

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

Deployments

Deployments

What is it?

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

Deployments

Comparing: Pods vs Deployment:

Single Pod:

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

Deployment:

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

File: labs/04_deployment.yaml

Deployments

Lab 5: Create a Deployment

- Creating a Deployment

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

- Checking the results

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

- Scale out the Deployment

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

- Check the nginx server version

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

Deployments

Lab 5: Scale the Deployment

- Scale in the deployment. Let's be frugal

```
kubectl edit deployment web-server-deployment

# set spec.replicas to 3
```

- Checking the results

```
kubectl get pod

# OR

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

Deployments

Lab 5: Update the Deployment

- Update the deployment

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

# OR

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

- Checking the results

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

- Check the Nginx server version now

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

Deployments

Let's kubectl

Lab 5: Deployment

https://github.com/aws-vls-dub/eks/tree/master/labs/05-deployments

Services

Services

What's a service?

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

Services

What's a service?

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

But why?

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

Services

What's a service?

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

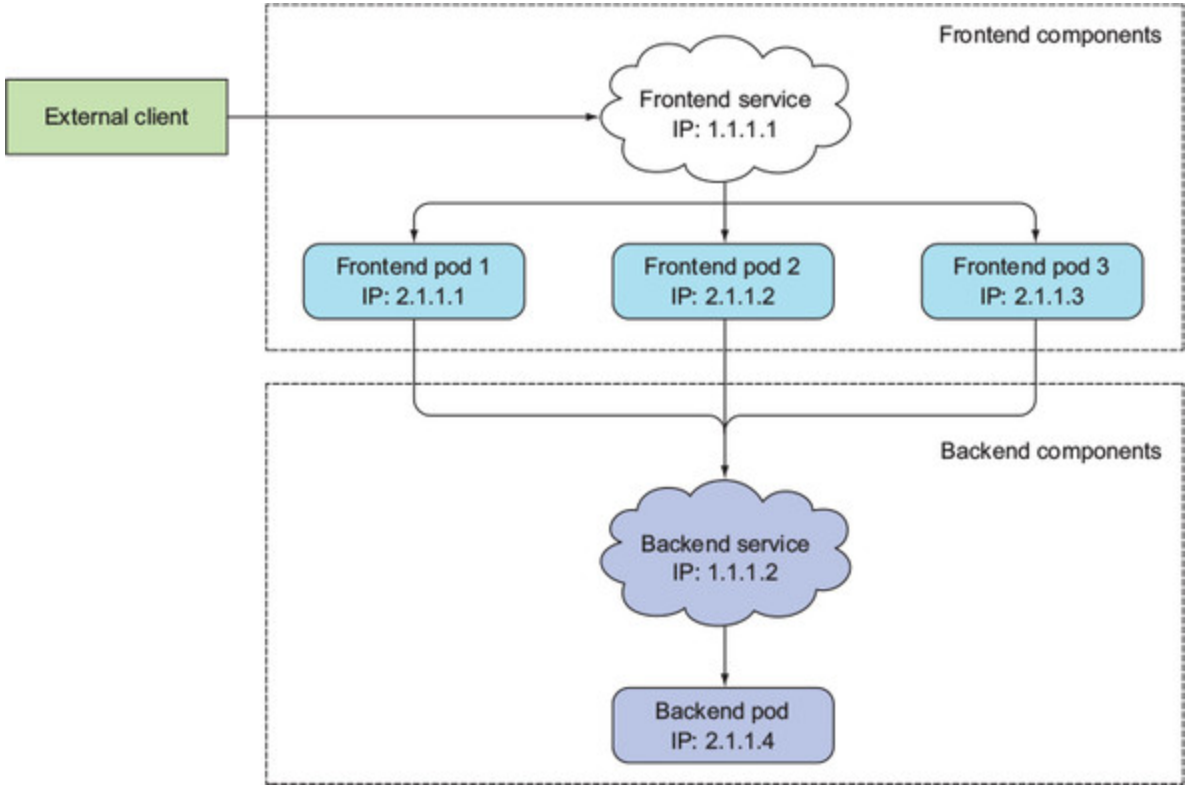
But why?

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

So, how to reach an application?

Services

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



* from https://kubernetes.io/

Lab 6: Create a Service

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

File: labs/06-services

Lab 6: Create a Service

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

- Creating a service

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

File: labs/06-services

Lab 6: Create a Service

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

- Creating a service

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

- Checking the results

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

File: labs/06-services

Services

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

Connecting to your service:

```
kubectl edit svc web-app

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

- Checking the results

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

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

Let's kubectl

Lab 6: Services

https://github.com/aws-vls-dub/eks/tree/master/labs/06-services

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

Questions?

Namespaces

Namespaces

What is a Namespace?

- You can think of each namespace as a folder that holds a set of objects.
- Virtual clusters backed by the same physical cluster.

Namespaces

What is a Namespace?

- You can think of each namespace as a folder that holds a set of objects.
- Virtual clusters backed by the same physical cluster.

But why?

- Namespaces are a way to divide cluster resources between multiple users.
- Limit and group resources
- Isolate networking

Namespaces

What is a Namespace?

- You can think of each namespace as a folder that holds a set of objects.
- Virtual clusters backed by the same physical cluster.

But why?

- Namespaces are a way to divide cluster resources between multiple users.
- Limit and group resources
- Isolate networking

Very important!

- Don't confuse Kubernetes namespaces with Linux namespaces.

Namespaces

Namespaces enable you to separate resources that don’t belong together into non-overlapping groups.

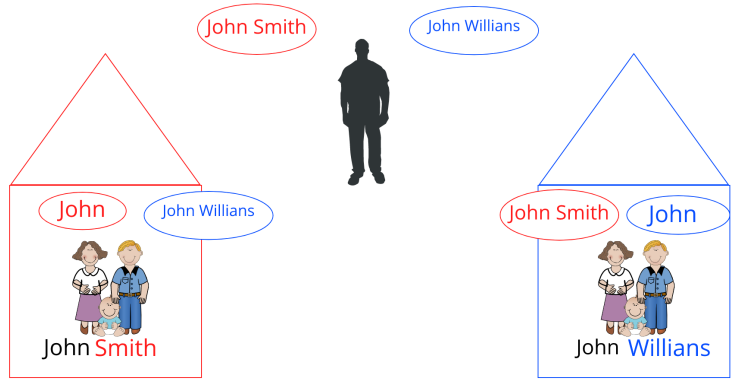
Namespaces don’t provide any kind of isolation of running objects.

Namespaces

Namespaces enable you to separate resources that don’t belong together into non-overlapping groups.

Namespaces don’t provide any kind of isolation of running objects.

Let's imagine namespaces as if it was a family houses.



Namespaces

Lab 8: Create namespaces

- Creating a namespace using the declarative method

```
apiVersion: v1
kind: Namespace
metadata:
  name: mynamespaced
```

File: labs/08-namespaces

Namespaces

Lab 8: Create namespaces

- Creating a namespace using the declarative method

```
apiVersion: v1
kind: Namespace
metadata:
  name: mynamespaced
```

- Creating a namespace using the imperative method

```
kubectl create namespace mynamespace
```

File: labs/08-namespaces

Namespaces

Lab 8: Create namespaces

- Creating a namespace using the declarative method

```
apiVersion: v1
kind: Namespace
metadata:
  name: mynamespaced
```

- Creating a namespace using the imperative method

```
kubectl create namespace mynamespace
```

- Checking the results

```
kubectl get namespaces
kubectl describe ns mynamespace
```

File: labs/08-namespaces

Let's kubectl

Lab 8: Namespaces

https://github.com/aws-vls-dub/eks/tree/master/labs/08-namespaces

Follow the steps in the `labs/08-namespaces/README.md` file.

Questions?

Ingress

What's a Ingress?

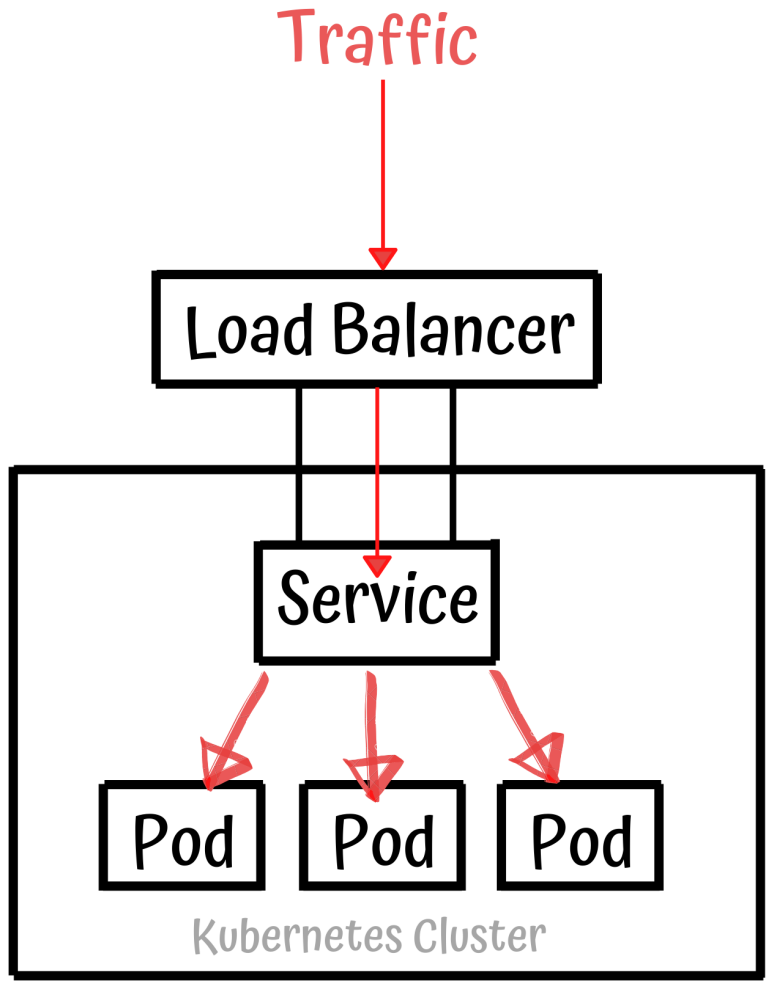
- Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster
- Traffic routing is controlled by rules defined on the Ingress resource.

Wait, so what's the difference between Ingress and Service (Load Balancer)?

Ingress vs Service

- LoadBalancer
  - Service is the standard way to expose a service to the internet.
  - All traffic on the port you specify will be forwarded to the service
  - There is no filtering, no routing, etc.
- Ingress
  - Ingress is actually NOT a type of service.
  - Instead, it sits in front of multiple services and act as a “smart router” or entrypoint into your cluster.

LoadBalancer



Ingress

How to apply it?

- In order for the Ingress resource to work, the cluster must have an **Ingress Controller** running.
- Unlike other types of controllers which run as part of the kube-controller-manager binary, Ingress controllers are not started automatically with a cluster.
- Kubernetes as a project currently supports and maintains GCE and nginx controllers.
- AWS Supports ALB Ingress Controller which enables ingress using the AWS Application Load Balancer.

ALB Ingress Controller

- The AWS ALB Ingress Controller for Kubernetes is a controller that triggers the creation of an Application Load Balancer (ALB) whenever an Ingress resource is created
- The trigger will happen once you add the kubernetes.io/ingress.class: alb annotation on your Ingress Object.
- ALB Ingress repository: https://github.com/kubernetes-sigs/aws-alb-ingress-controller

Create an ALB Ingress

- AWS Docs: https://docs.aws.amazon.com/eks/latest/userguide/alb-ingress.html

Ingress definition

```
apiVersion: extensions/v1beta1
kind: Ingress
metadata:
  name: "2048-ingress"
  namespace: "2048-game"
  annotations:
     kubernetes.io/ingress.class: alb
     alb.ingress.kubernetes.io/scheme: internet-facing
  labels:
    app: 2048-ingress
spec:
  rules:
    - http:
        paths:
          - path: /*
            backend:
              serviceName: "service-2048"
              servicePort: 80
```

Lab 7: Expose an Application using ALB Ingress

- Executing the installation script: `eks/labs/07-Ingress/configure-ingress.sh`

Exposing an Application

```
$ kubectl apply -f 2048-namespace.yaml
$ kubectl apply -f 2048-deployment.yaml
$ kubectl apply -f 2048-service.yaml
$ kubectl apply -f 2048-ingress.yaml
```

Let's kubectl

Lab 7: Ingresss

https://github.com/aws-vls-dub/eks/tree/master/labs/07-Ingress

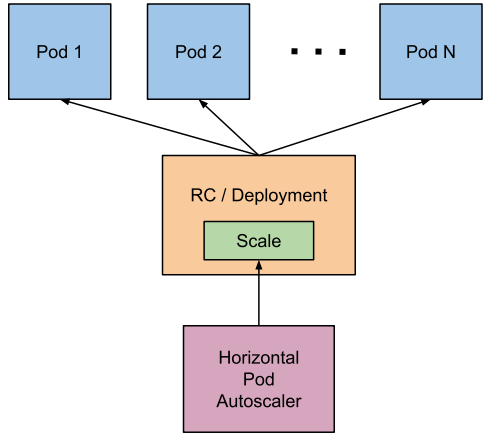
Follow the steps in the `labs/07-Ingress/README.md` file.

Before we move on, any questions?

Horizontal Pod Autoscaler (HPA)

What's HPA?

- The Horizontal Pod Autoscaler automatically scales the number of pods in a replication controller, deployment, replica set or stateful set based on observed CPU utilization(or, with custom metrics support, on some other application-provided metrics).
- The controller periodically adjusts the number of replicas in a replication controller or deployment to match the observed average CPU utilization to the target specified by user.



How does the Horizontal Pod Autoscaler work?

- During each period, the controller manager queries the resource utilization against the metrics specified in each HorizontalPodAutoscaler definition.
- The controller manager obtains the metrics from either the resource metrics API (for per-pod resource metrics), or the custom metrics API (for all other metrics).
- This can help your applications scale out to meet increased demand or scale in when resources are not needed, thus freeing up your worker nodes for other applications.

Lab 9: Scale your Pods Automatically with HPA

Install Metric Server

- Deploy the Metrics Server with the following command:

```
$ kubectl apply -f https://github.com/kubernetes-sigs/metrics-server/releases/download/v0.3.6/c
```

- Verify that the metrics-server deployment is running the desired number of pods with the following command.

```
$ kubectl get deployment metrics-server -n kube-system
```

Output:

```
NAME             READY   UP-TO-DATE   AVAILABLE   AGE
metrics-server   1/1     1            1           6m
```

Lab 9: Scale your Pods Automatically with HPA

1. After Installing the the metric-server, let's update the deployment from the previous lab, it was added the following lines:

```
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
         resources:
           limits:
             cpu: 300m
           requests:
             cpu: 200m
```

Lab 9: Scale your Pods Automatically with HPA

1. Apply the new version:

```
$ kubectl apply -f 2048-deployment-hpa.yaml
```

2. Create a Horizontal Pod Autoscaler resource for the 2048 deployment.

```
kubectl autoscale deployment 2048-deployment -n 2048-game --cpu-percent=50 --min=1 --max=10
```

1. Create a load for the web server. The following command uses the Apache Bench program to send hundreds of thousands of requests. This should significantly increase the load and cause the autoscaler to scale out the deployment.

```
kubectl run apache-bench -i --tty --rm --image=httpd -- ab -n 500000 -c 1000 http://service
```

1. Open a new tab on your Cloud9 Environment and check the pod scaling with the command:

```
kubectl get horizontalpodautoscaler.autoscaling/2048-deployment -n 2048-game -w
```

Output:

```
NAME    REFERENCE          TARGETS   MINPODS   MAXPODS   REPLICAS   AGE
2048-deployment   Deployment/httpd   76%/50%   1         10        10         4m50s
```

Let's kubectl

Lab 9: HPA

https://github.com/aws-vls-dub/eks/tree/master/labs/09-hpa

Follow the steps in the `labs/09-hpa/README.md` file.

Questions?

Thank you!