Getting Started with Kubernetes on AWS

Brought to you by the AWS Sydney Cloud Support Team

Day 2

Agenda

- Kubernetes basic resources
- Exploring the cluster with kubectl
- Designing your application for high availability
- Architecting your application using services
- Exposing your application to the world

Firstly...

Do you have a cluster?

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

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

Link to GIF

Do you have a cluster?

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

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

Link to GIF

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

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

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

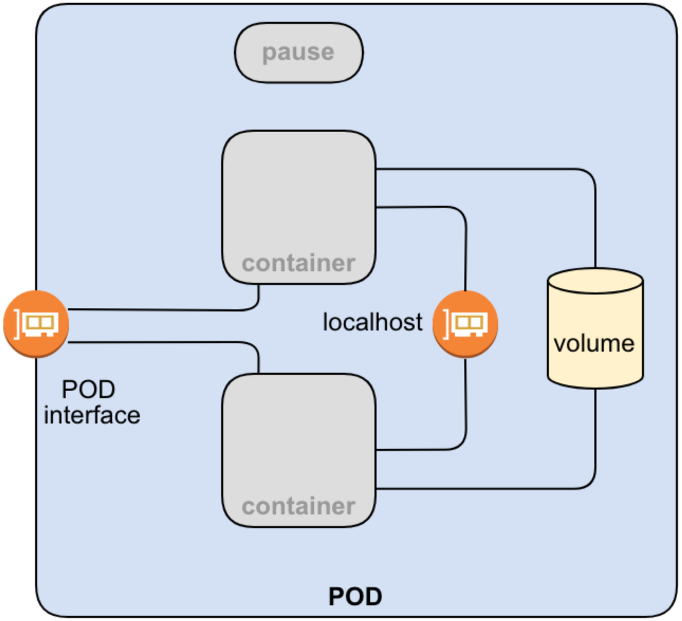
Lets pull in the latest changes

```
$ cd eks
$ git pull
```

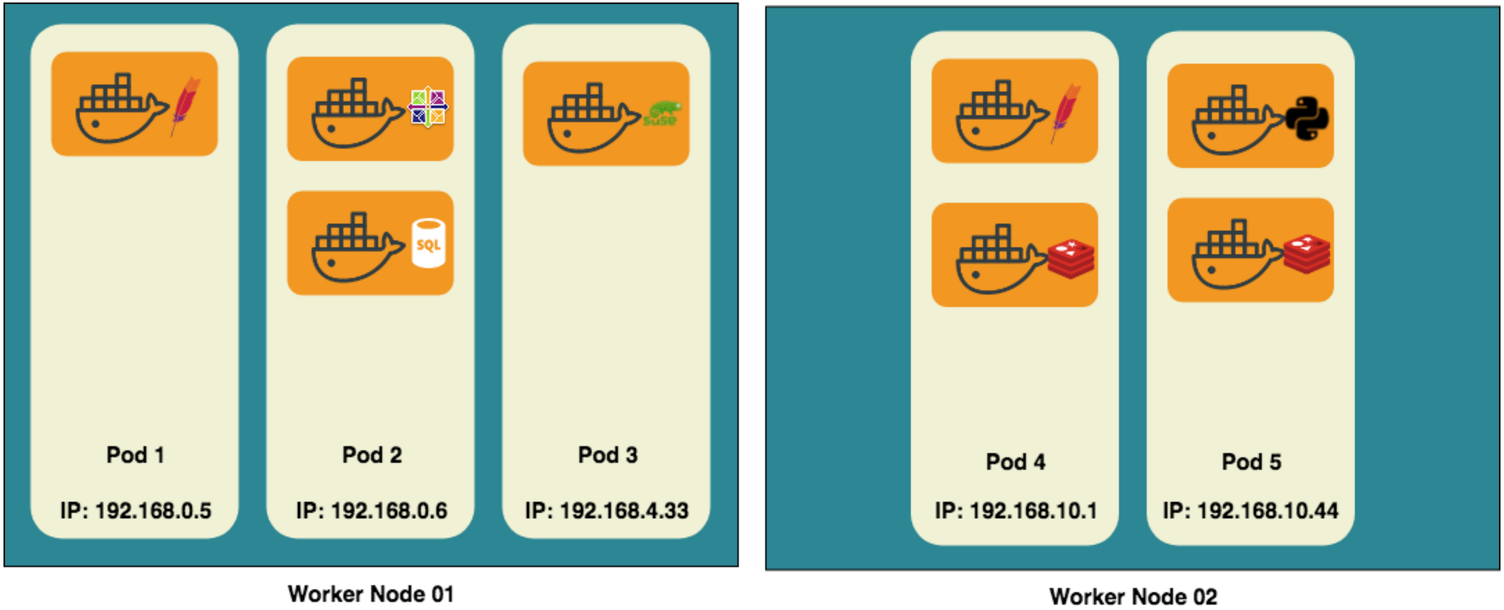
Pod

What is a Pod?

- The smallest building block of Kubernetes
- A Pod encapsulates the container(s) and resources needed to run the application
- A unit of deployment



What is a Pod?


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

Creating Pods in Kubernetes

Lab 1: Define a Pod

Pod definition

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - name: container1
    image: nginx
```

Lab 1: Define a Pod

Pod definition

```
apiVersion: v1
kind: Pod
metadata:
  name: web-server
spec:
  containers:
  - name: container1
    image: nginx
```

File: labs/01_pod.yaml

Lab 1: Creating a Pod

Send the definition to the cluster:

```
kubectl create -f labs/01_pod.yaml
```

Lab 1: Check the Pod

List and describe the Pod

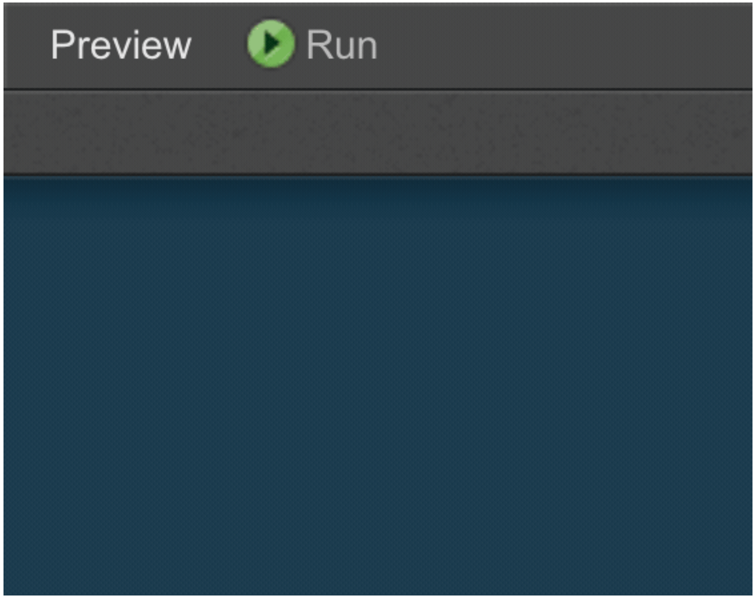
```
kubectl get pod

kubectl describe pod web-server
```

Lab 1: Check the Pod

Create a tunnel and connect to your Pod

```
kubectl port-forward pod/web-server 8080:80 &

curl localhost:8080
```

Or, connect using a browser and Cloud9



Lab 1: Clean up

We ran the port-forward in the background, lets clean it up before we move on

In the Cloud9 terminal to bring it back to the foreground:

```
fg

# Hit Ctrl + C to kill the port-forward
```

Lab 2: Working with Pods

Check the logs

```
kubectl logs web-server
```

Lab 2: Working with Pods

Check the logs

```
kubectl logs web-server
```

Connect

```
# run one command
kubectl exec web-server cat /etc/hostname

# run with console connected to pod
kubectl exec -it web-server -- bash
```

Lab 2: Working with Pods

Check the logs

```
kubectl logs web-server
```

Connect

```
# run one command
kubectl exec web-server cat /etc/hostname

# run with console connected to pod
kubectl exec -it web-server -- bash
```

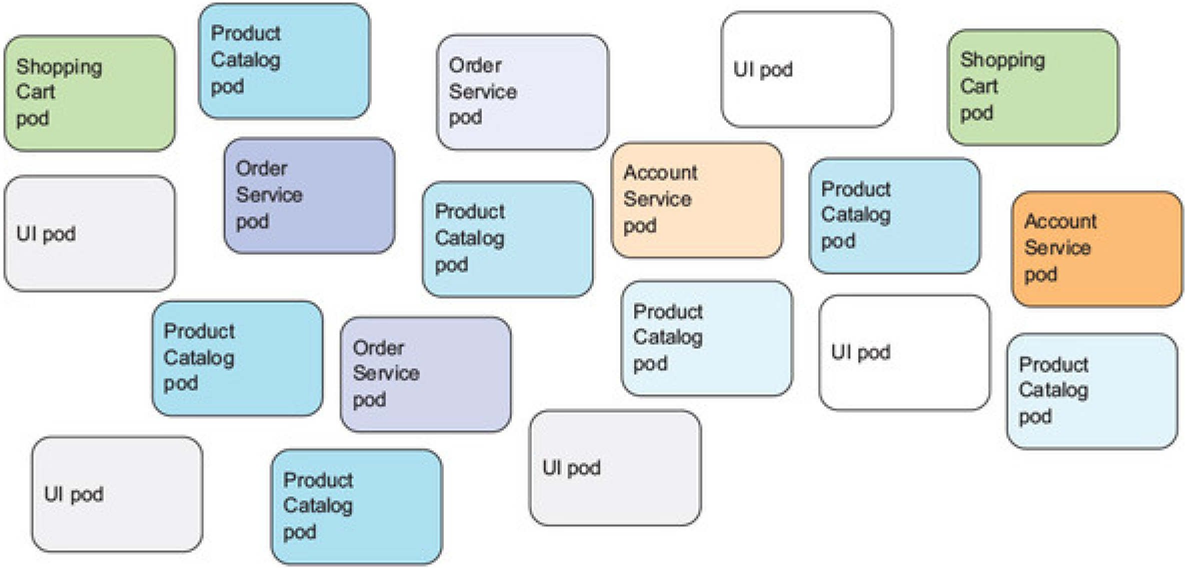
You can delete it: *But keep it for now!*

```
kubectl delete pod web-server

# OR

kubectl delete -f labs/01_pod.yaml
```

Labels

Labels

What if we are running a lot of pods?

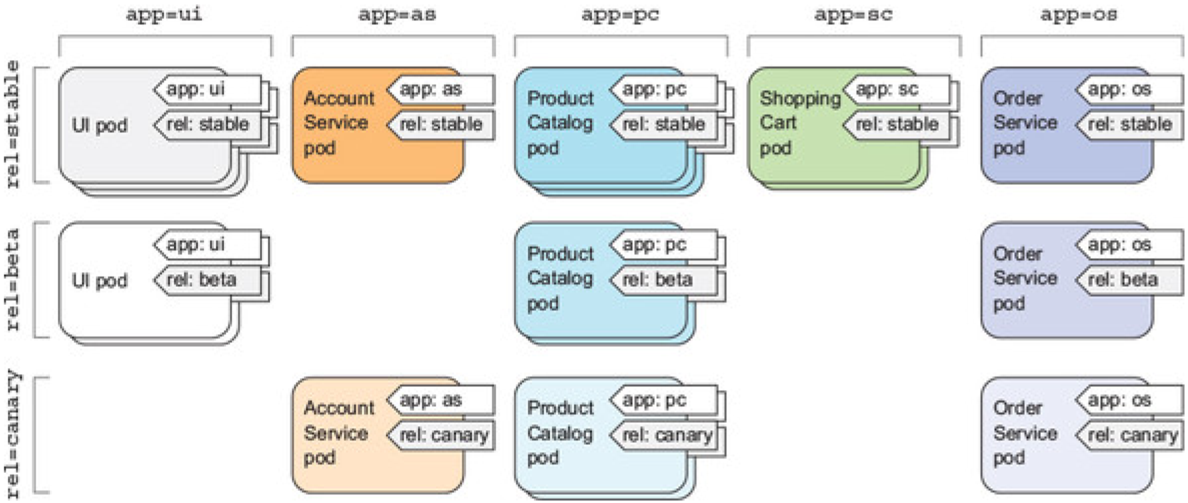


Picture from Kubernetes in action

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 3: 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

Links

Pods

- https://kubernetes.io/docs/concepts/workloads/pods/pod-overview/
- https://kubernetes.io/docs/concepts/workloads/pods/pod/

Kubernetes in Action

- https://www.safaribooksonline.com/library/view/kubernetes-in-action/9781617293726/

Demos

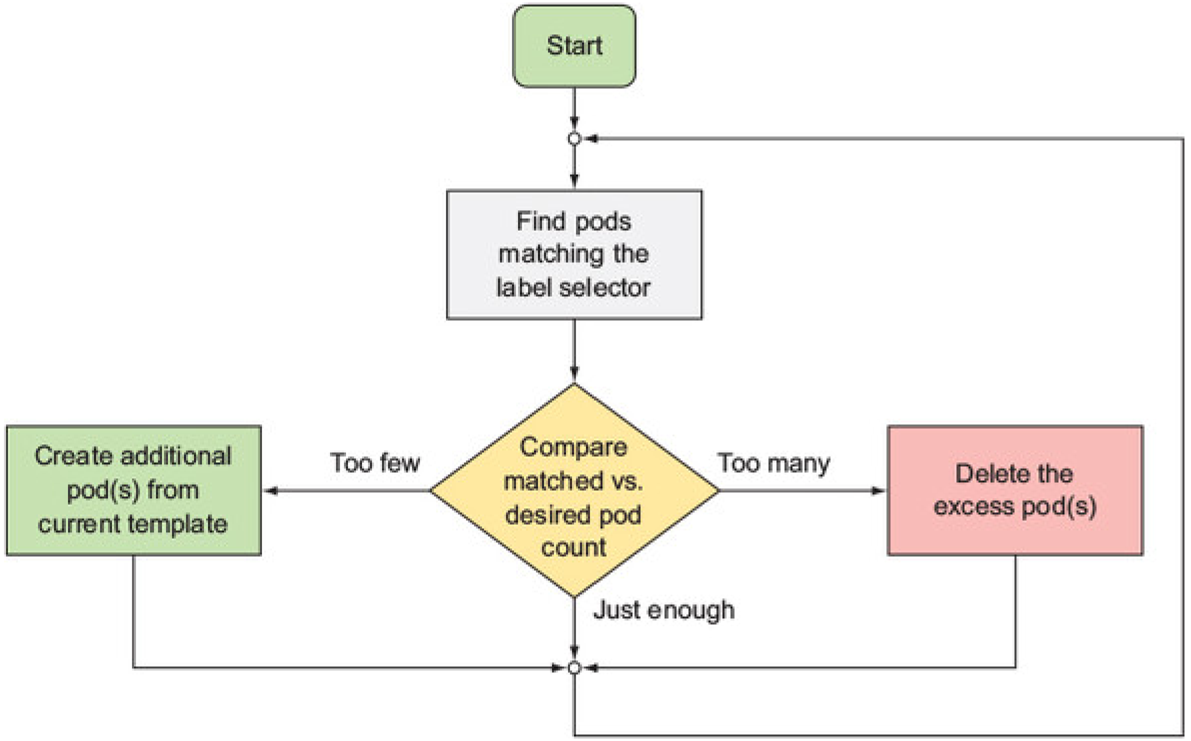
- https://eksworkshop.com/
- https://github.com/kubernetes/contrib/blob/master/micro-demos/

Controllers

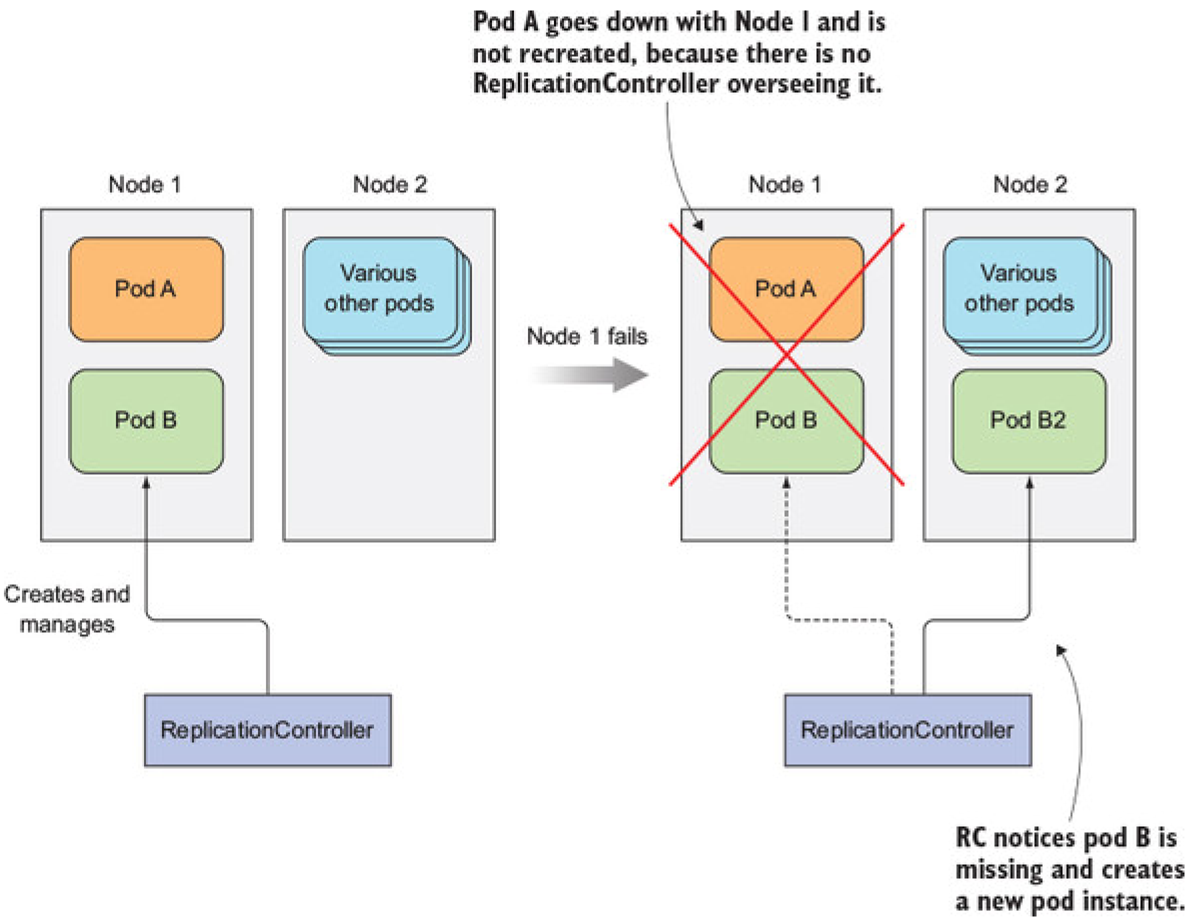
What controllers are and what they do?

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

What controllers are and what they do?


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

What controllers are and what they do?



Pod A goes down with Node I and is not recreated, because there is no ReplicationController overseeing it.

RC notices pod B is missing and creates a new pod instance.

Controllers

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

Deployments

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:

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

Deployment:

```yaml
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 4: 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 4: 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 4: 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
```

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

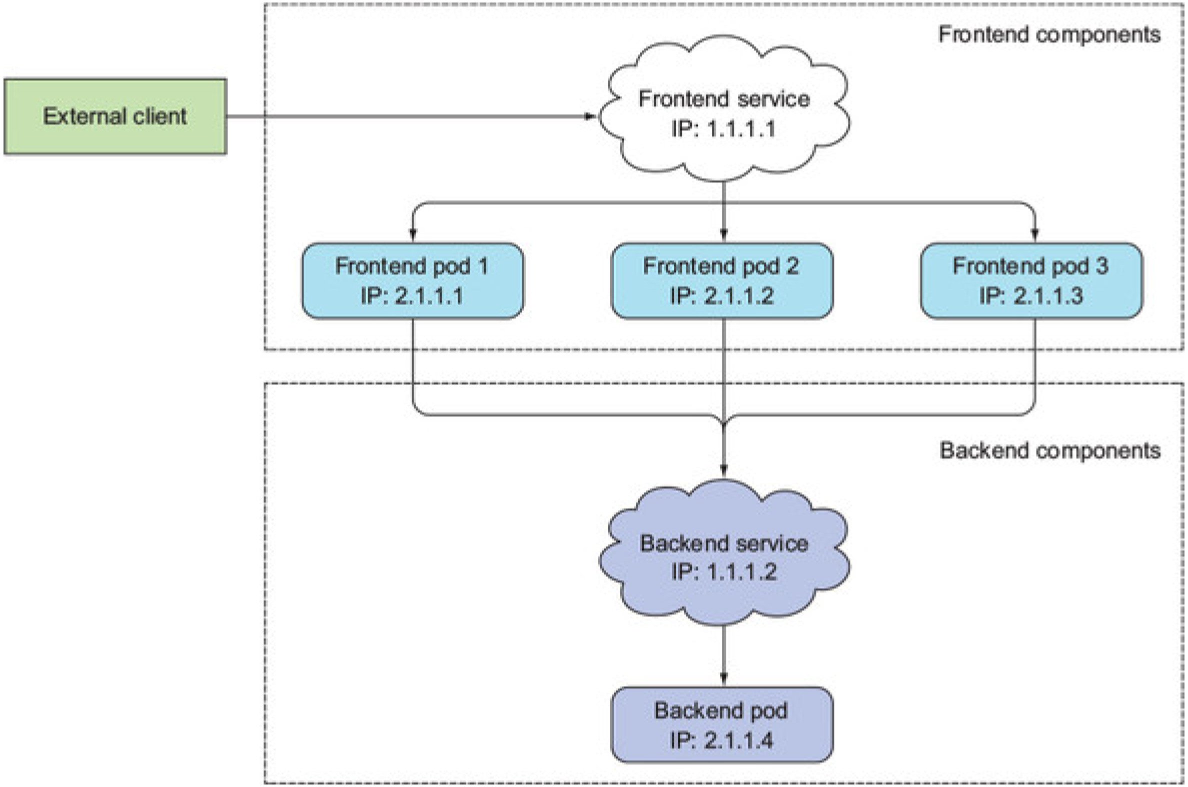
But why?

- 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 5: Create a Service

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

File: labs/05_services.yaml

Lab 5: Create a Service

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

- Creating a service

```
kubectl apply -f labs/05_services.yaml
```

File: labs/05_services.yaml

Lab 5: Create a Service

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

- Creating a service

```
kubectl apply -f labs/05_services.yaml
```

- Checking the results

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

File: labs/05_services.yaml

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 nginx-app

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

- Checking the results

```
kubectl get svc nginx-app
```

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

End of Day 2

Questions?

Thank you!