

Container orchestration with Kubernetes

Maddie Patrichi: ioana-madalina.patrichi@oracle.com

Tutorial Overview

1. Background
2. Kubernetes primitives
3. Architecture
4. Lifecycle of a pod
5. Single-node demo (Minikube)
6. Creating your own Kubernetes cluster (Terraform)
7. Multi-node demo (OKE)

Overview: What is Kubernetes?

“Kubernetes is an open-source system for automating deployment, scaling and management of containerised applications.”



Overview: What problems does it solve?

- Users expect applications/services to be available 24/7
- Developers expect to be able to deploy updates to their code multiple times per day
- Desire to use cloud resources efficiently via container orchestration
- Fault-tolerant, self-healing
- Scalability

Overview: History

- Born from a Google internal project in mid-2014 (Google “Borg”)
- 1.0 release in July 2015
- Google partnered with the Linux Foundation to form the Cloud Native Computing Foundation (CNCF) to offer Kubernetes as an open standard
- Abbreviated as “k8s”, Greek for “helmsman” or “pilot”
- Written in Golang

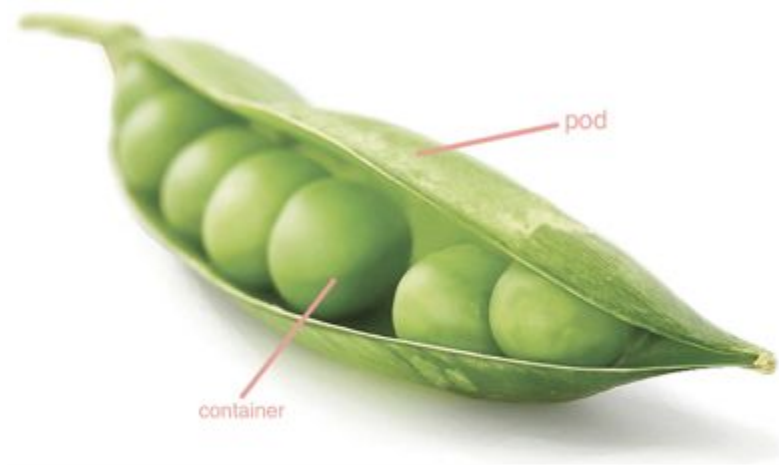
Overview: Advantages of using Kubernetes

... vs. other Container Orchestration applications

- Based on extensive experience from Google, over a long period of time
- Large open source community project, mature governing organisation (CNCF)
- Auto-scaling, cloud-agnostic-yet-integratable technologies

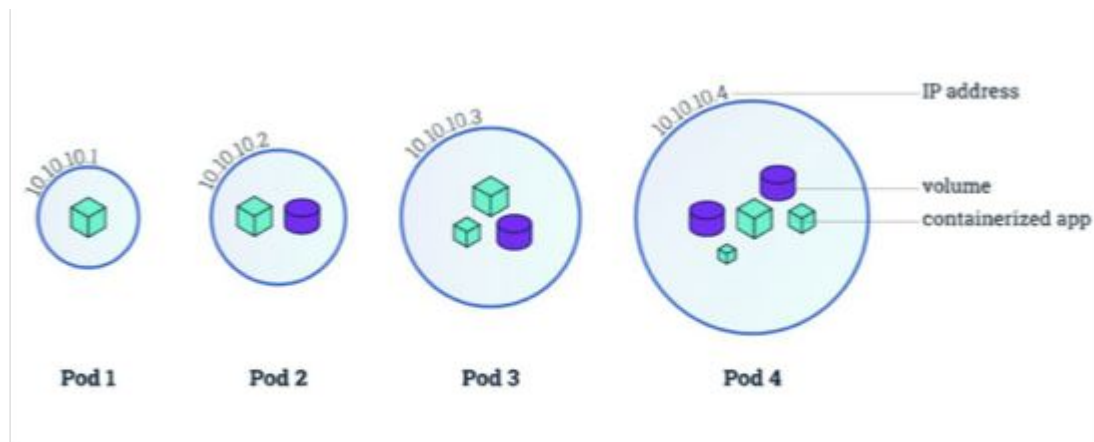
Kubernetes primitives: Pod

- Set of one or more containers that act as a unit and are scheduled onto a node together
- Share a local network and can share file-system volumes



Kubernetes primitives: Pod

- Set of one or more containers that act as a unit and are scheduled onto a node together
- Share a local network and can share file-system volumes



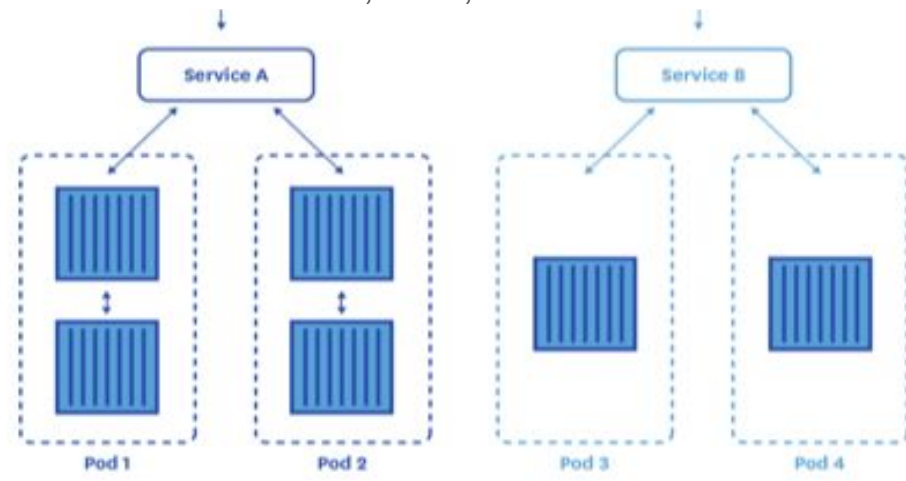
Kubernetes primitives: Deployment

- Responsible for creating and updating pods
- Kubernetes will manage state based on the definitions provided
- Can scale up/down to meet demand
- Can roll back to older version or roll forward

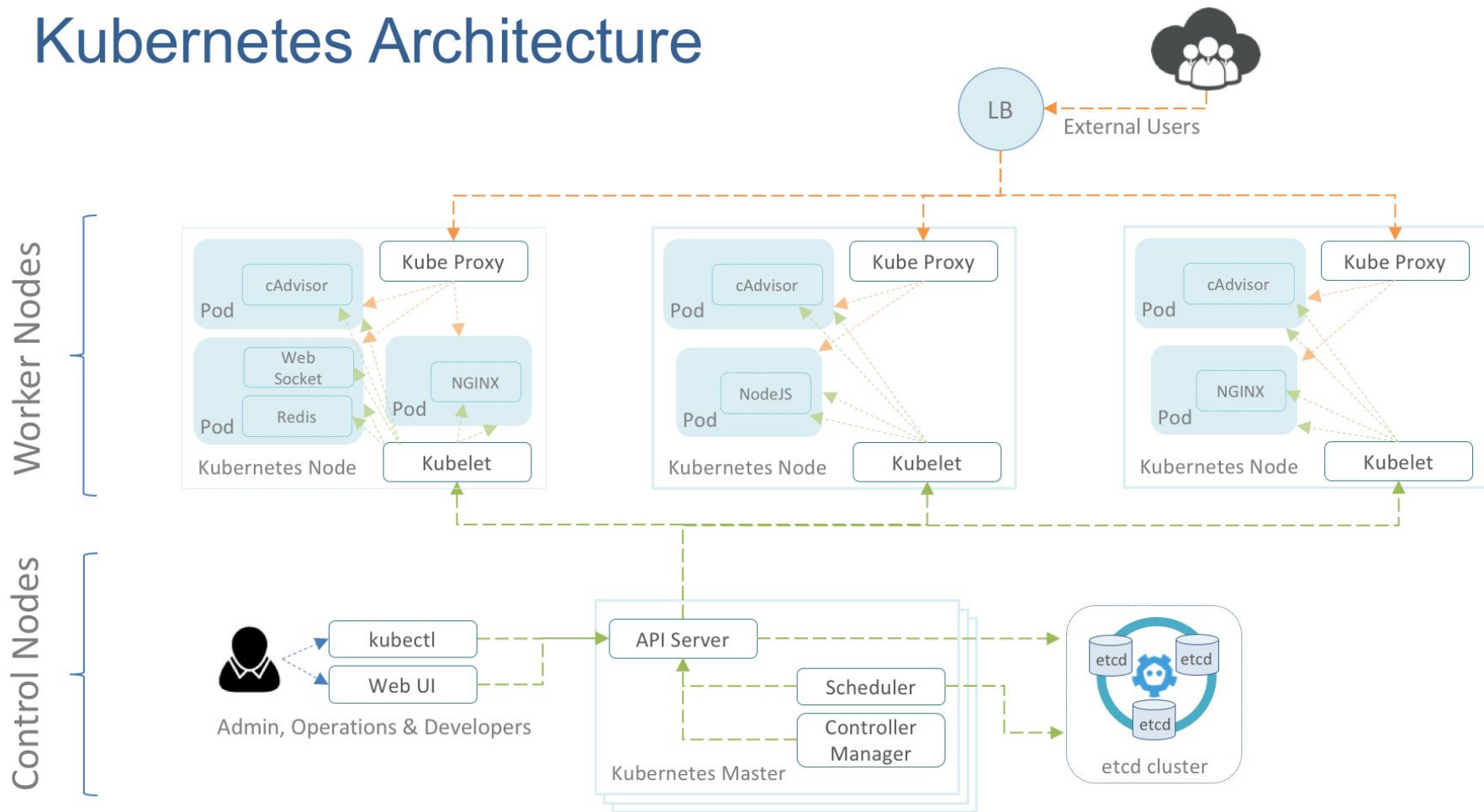
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

Kubernetes primitives: Service

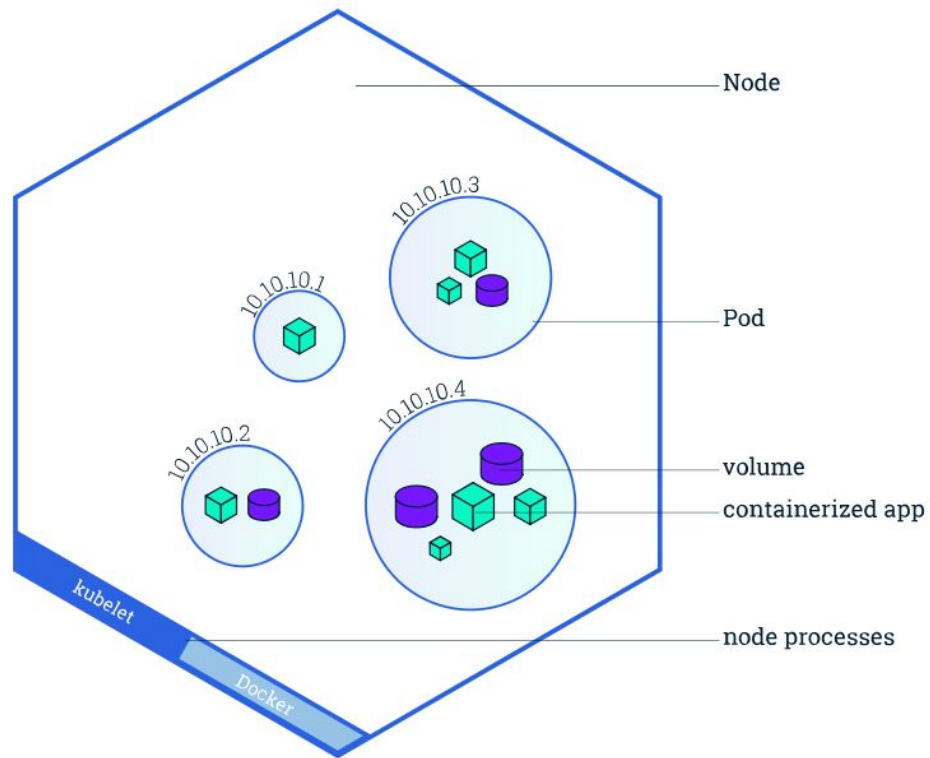
- Defines a way to access pods in a consistent way
- Services find the Pod to route traffic based on the Labels/Selectors in the manifest
 - Inside the cluster, they perform load balancing
 - They can also interact with GKE, OKE, etc to create external load balancers



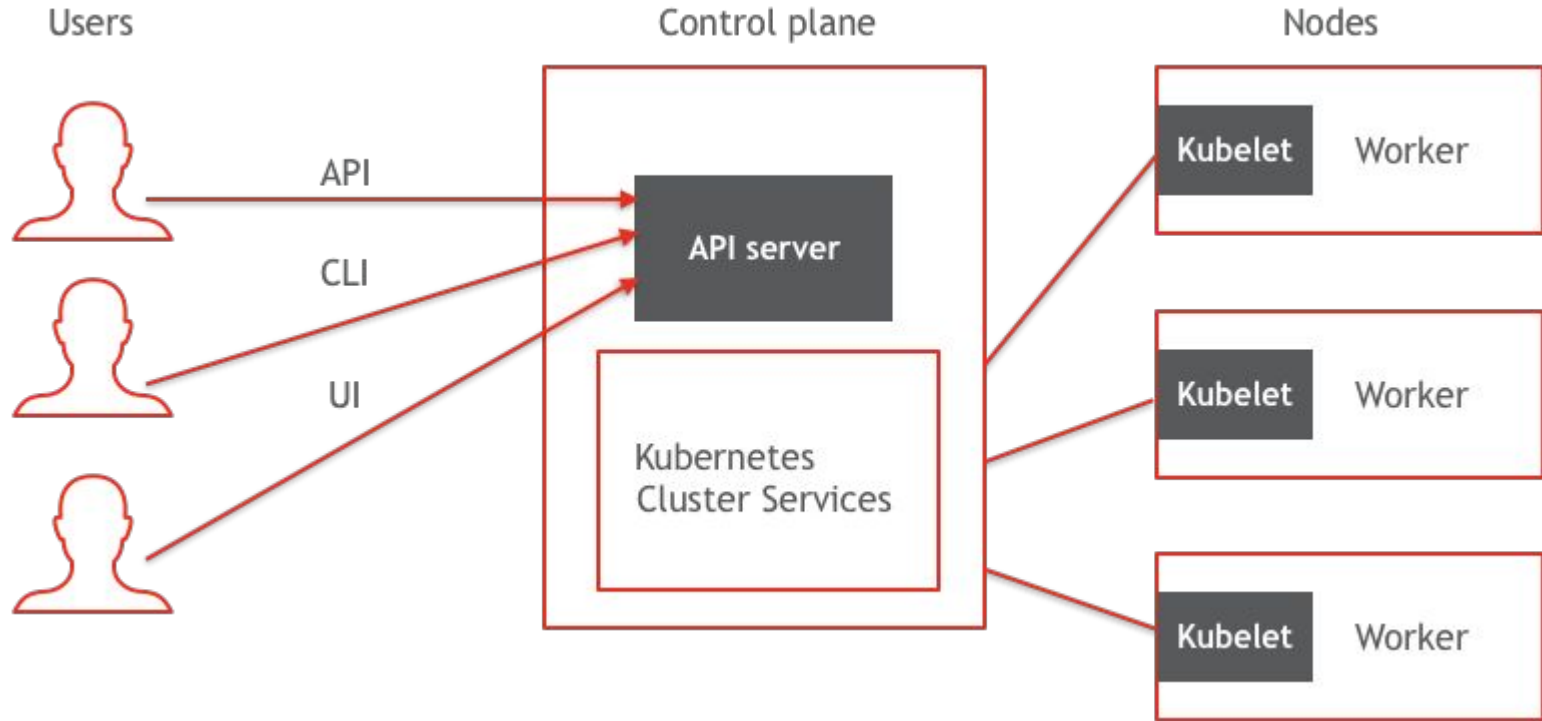
Kubernetes Architecture



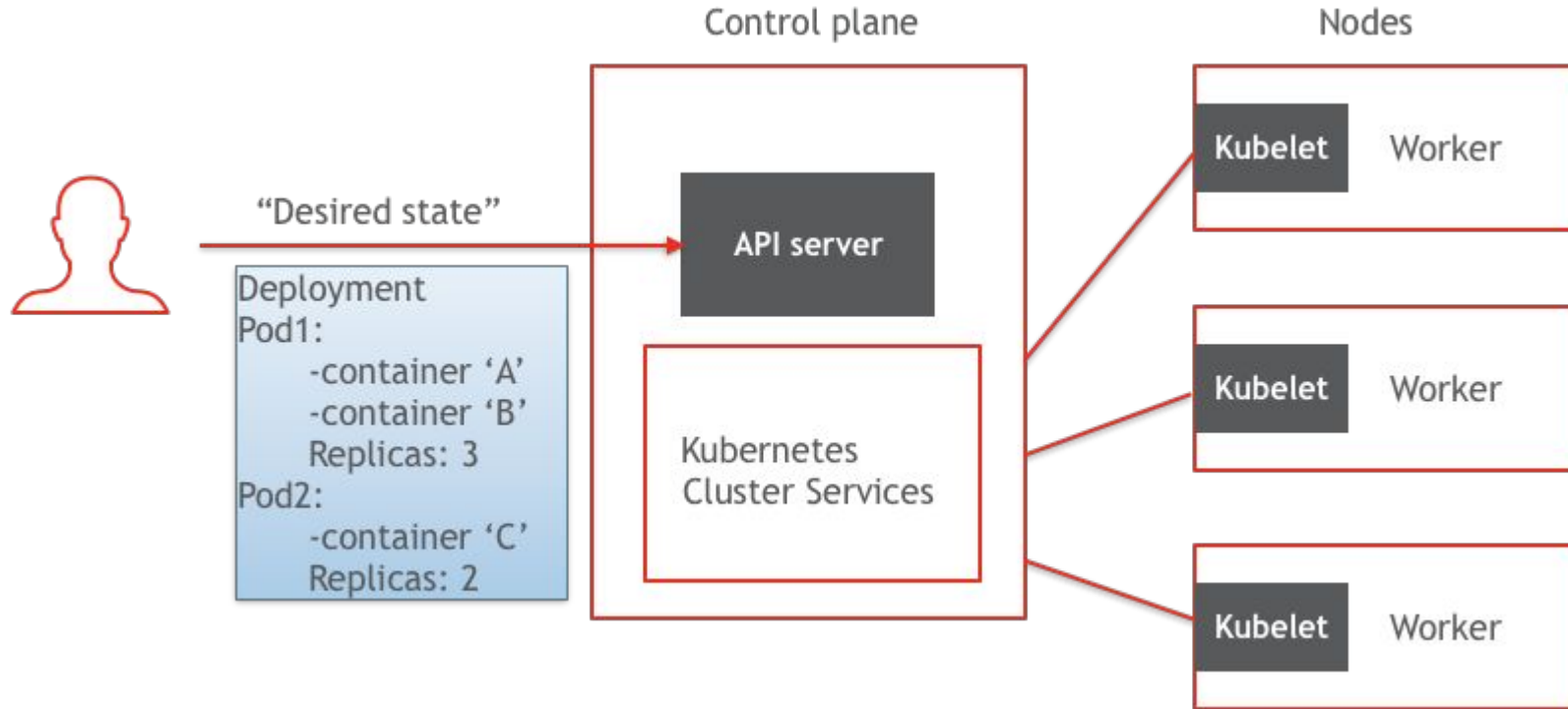
Kubernetes: Worker Nodes



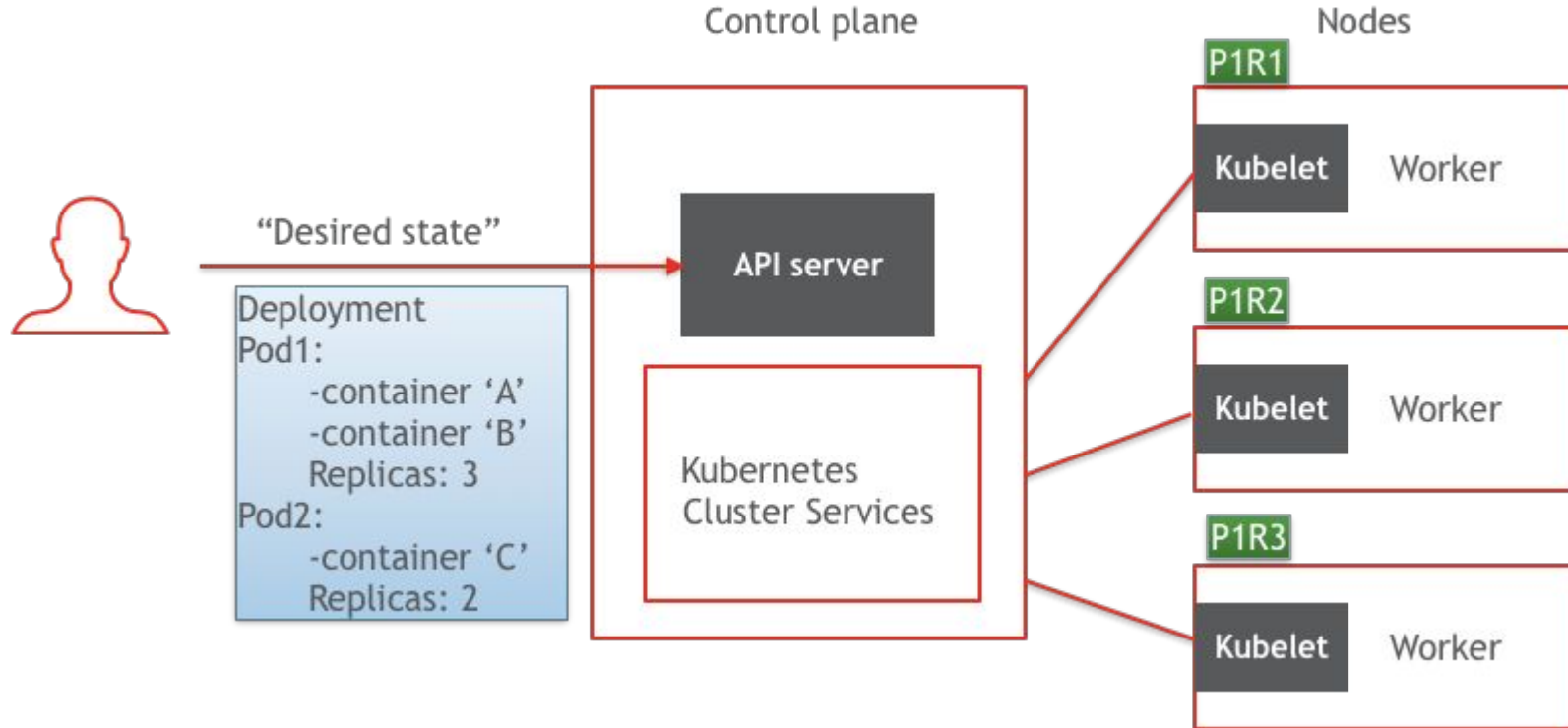
Lifecycle of a pod



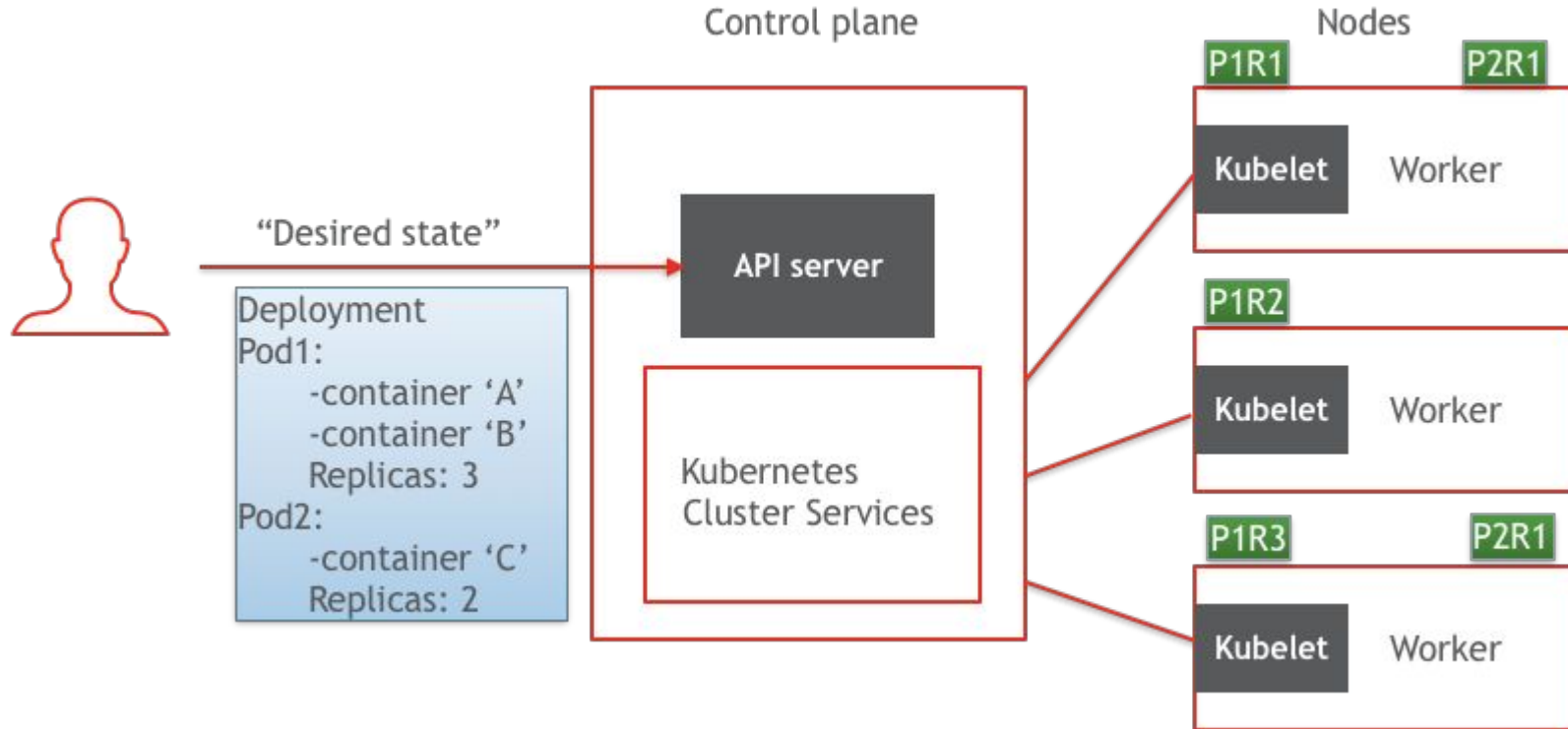
Lifecycle of a pod



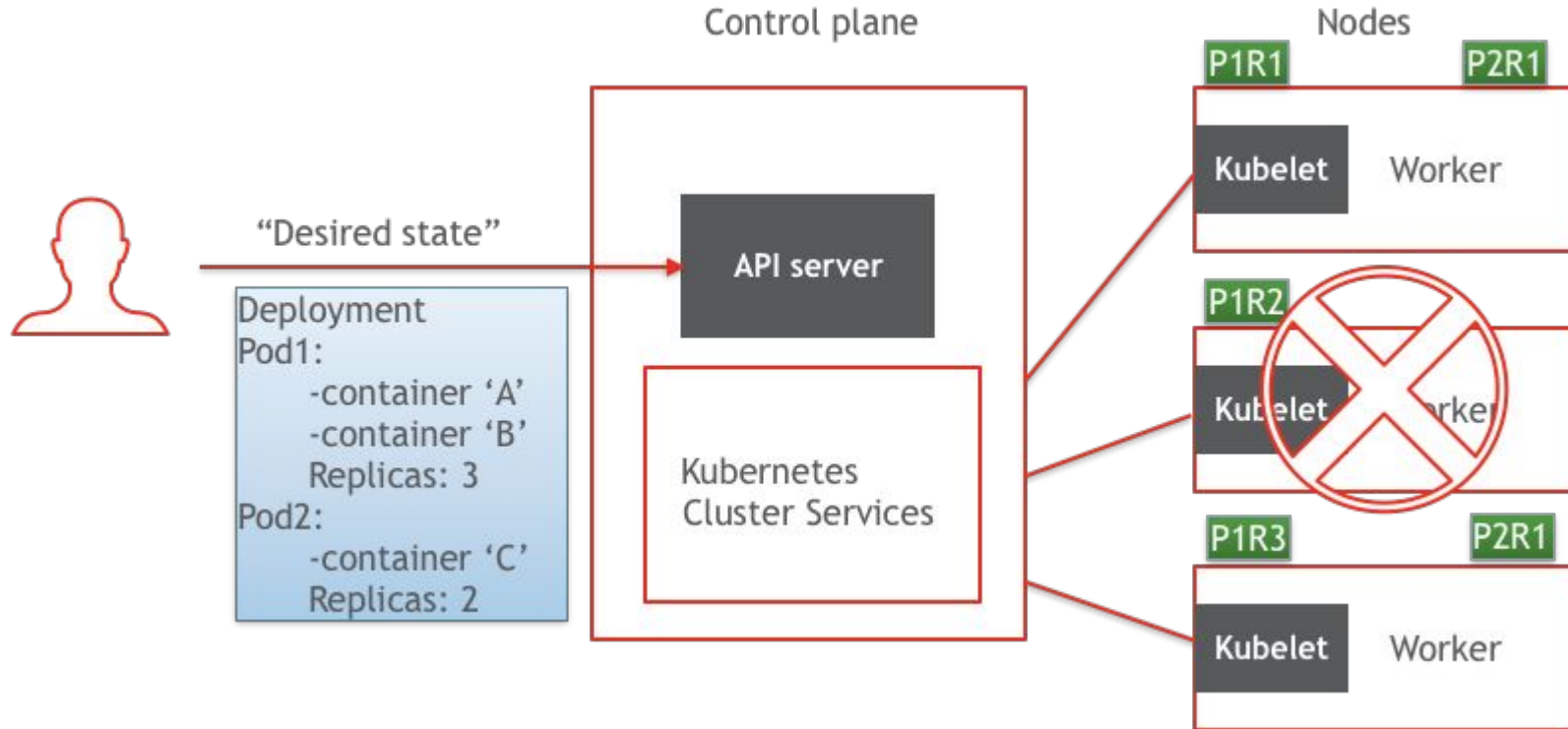
Lifecycle of a pod



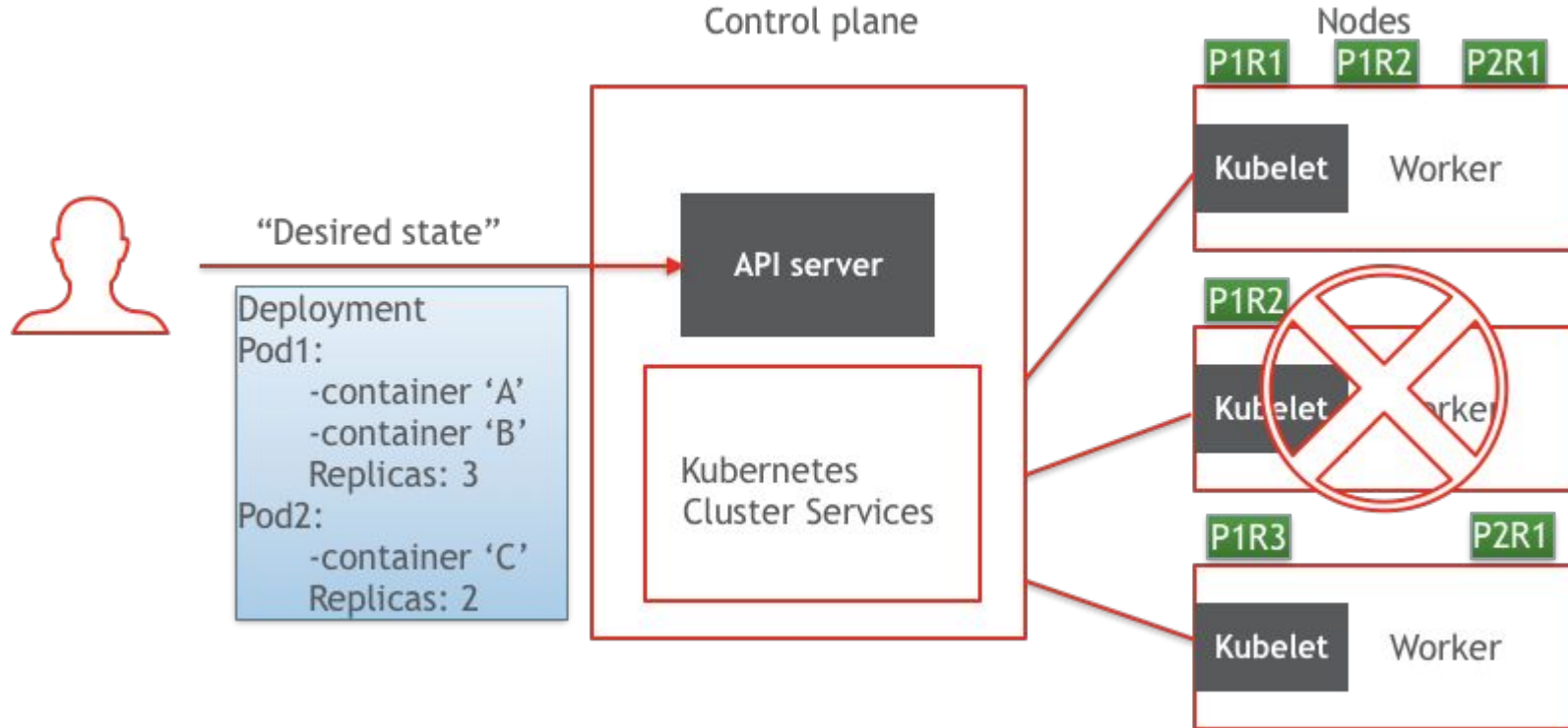
Lifecycle of a pod



Lifecycle of a pod



Lifecycle of a pod



Minikube

<https://kubernetes.io/docs/tasks/tools/install-minikube/>

<https://github.com/kubernetes/minikube>

- Tool which runs Kubernetes locally
- Great for testing out Kubernetes components
- Not so great in production

Minikube

<https://kubernetes.io/docs/tasks/tools/install-minikube/>

1. Start minikube


```
mpatrich-Mac:bristoldojo mpatrich$ minikube start
There is a newer version of minikube available (v0.30.0).  Download it here:
https://github.com/kubernetes/minikube/releases/tag/v0.30.0

To disable this notification, run the following:
minikube config set WantUpdateNotification false
Starting local Kubernetes v1.9.4 cluster...
Starting VM...
Getting VM IP address...
Moving files into cluster...
Setting up certs...
Connecting to cluster...
Setting up kubeconfig...
Starting cluster components...
Kubectl is now configured to use the cluster.
```

Minikube

2. Check minikube status

```
mpatrich-Mac:~ mpatrich$ minikube status
minikube: Running
cluster: Running
kubectl: Correctly Configured: pointing to minikube-vm at 192.168.99.100
```



Minikube-vm which contains both K8S Master and Worker components

Note: Commands which you would typically run on your Kubernetes nodes will be prefixed by *minikube* if run from your local machine.

To ssh into the minikube vm:


```
mpatrich-Mac:~ mpatrich$ minikube ssh
```

```
minikube
$
```

Minikube: Kubectl

- CLI tool used to deploy and manage applications on Kubernetes
- Inspect cluster resources
- CRUD operations on components, etc.
- Interacts with the API server
- Configuration file: `~/.kube/config`

Minikube master node
API server



```
apiVersion: v1
clusters:
- cluster:
    insecure-skip-tls-verify: true
    server: https://129.213.108.39:443
  name: bmcs-k8s-cluster
- cluster:
    certificate-authority: /Users/mpatrich/.minikube/ca.crt
    server: https://192.168.99.100:8443
  name: minikube
contexts:
- context:
    cluster: bmcs-k8s-cluster
    user: default-admin
  name: bmcs-k8s-context
- context:
    cluster: minikube
    user: minikube
  name: minikube
current-context: minikube
kind: Config
preferences: {}
users:
- name: default-admin
  user:
```

Minikube: Kubectl

- View physical nodes in K8S cluster:

```
mpatrich-Mac:~ mpatrich$ kubectl get nodes
```

NAME	STATUS	ROLES	AGE	VERSION
minikube	Ready	<none>	159d	v1.9.4

- Get cluster info:


```
mpatrich-Mac:~ mpatrich$ kubectl cluster-info
```

Kubernetes master is running at <https://192.168.99.100:8443>

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.

Minikube: Dashboard

```
mpatrigh-Mac:~ mpatrich$ minikube dashboard
Opening kubernetes dashboard in default browser...
```

 **kubernetes**

Search

+ CREATE

Overview

Cluster

Namespaces

Nodes

Persistent Volumes

Roles

Storage Classes

Namespace

default

Overview

Workloads

Cron Jobs

Daemon Sets

Deployments

Jobs

Pods

Replica Sets

Replication Controllers

Stateful Sets

Discovery and Load Balancing

Ingresses

Services

Config and Storage

Workloads

Workloads Statuses

100.00%

Deployments

100.00%

Pods

100.00%

Replica Sets






Deployments

Name	Labels	Pods	Age	Images
✓ nginx-deployment	app: nginx	4 / 4	6 days	nginx:1.7.9

Pods

Name	Node	Status	Restarts	Age
✓ nginx-deployment-6c54bd5869-fsggq	minikube	Running	2	6 days
✓ nginx-deployment-6c54bd5869-6ntwk	minikube	Running	2	6 days
✓ nginx-deployment-6c54bd5869-7l2gc	minikube	Running	2	6 days
✓ nginx-deployment-6c54bd5869-x928v	minikube	Running	3	6 days
✓ nginx-whc	minikube	Running	2	6 days

Minikube: Dashboard - Namespaces

Namespaces 			
Name 	Labels	Status	Age 
 default	-	Active	5 months
 kube-public	-	Active	5 months
 kube-system	-	Active	5 months

Minikube: Creating your first pod

- Create pod: `kubectl create -f <pod_config_file.yaml>`
- List pods: `kubectl get pods`
- Describe pod: `kubectl describe <name_of_pod>`

```
Name:          nginx-uob-installed-reqs
Namespace:     default
Node:          minikube/192.168.99.100
Start Time:    Thu, 01 Nov 2018 08:50:13 +0000
Labels:        name=nginx
Annotations:   <none>
Status:        Running
IP:            172.17.0.9
Containers:
```

- Exec into pod: `kubectl exec -it <name_of_pod> -- /bin/bash`

LIVE DEMO

Minikube: Accessing your first pod

- Ping pod from our local machine:

```
Normal Started 33m kubelet, minikube Started
mpatrich-Mac:firstpod mpatrich$ ping 172.17.0.9
PING 172.17.0.9 (172.17.0.9): 56 data bytes
Request timeout for icmp_seq 0
Request timeout for icmp_seq 1
Request timeout for icmp_seq 2
^C
--- 172.17.0.9 ping statistics ---
4 packets transmitted, 0 packets received, 100.0% packet loss
```

Will not work! Why?

Minikube: Accessing your first pod

- Ping pod from K8S node:

[illegible]

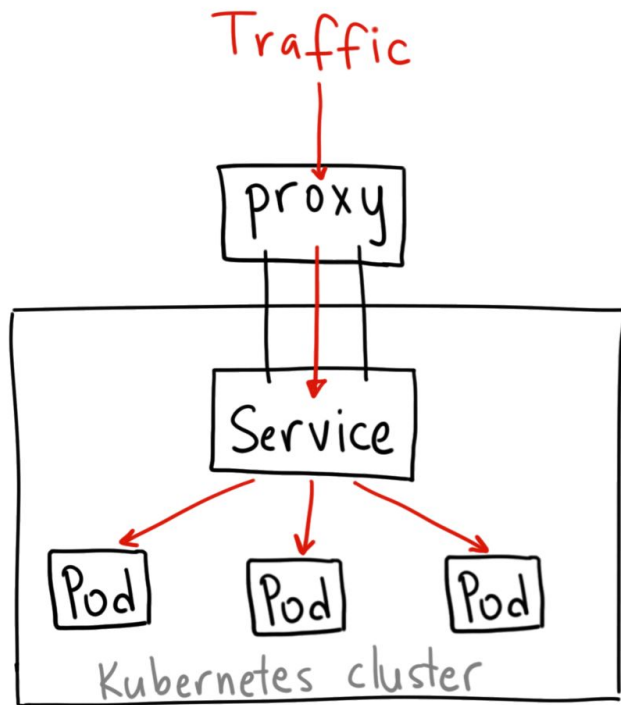
Minikube: Accessing your first pod

Solution:

1. hostPort
2. NodePort *Service*
3. LoadBalancer *Service* (Not available in Minikube)
4. ExternalName *Service* (Not available in Minikube) (Homework)

Pod Access: hostPort

- Makes use of the ClusterIP built-in Kubernetes service



Pod Access: hostPort

IP address of the K8S node where the container is running

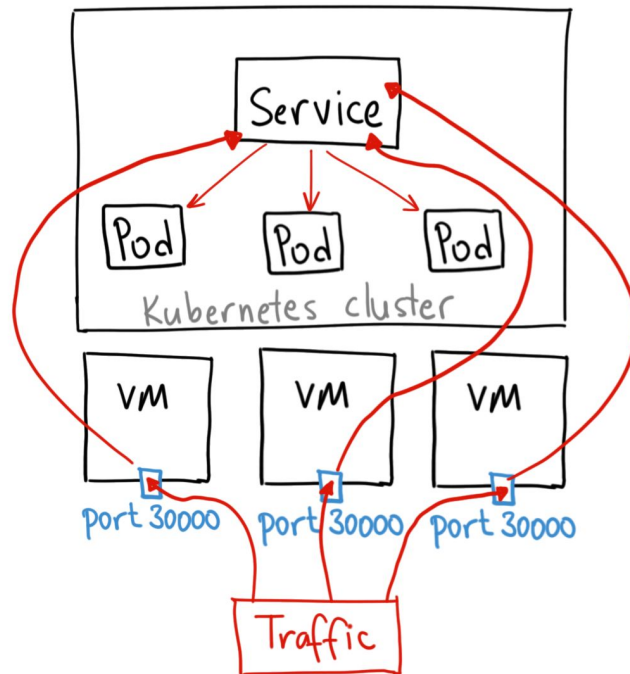
- Container port will be exposed to `<hostIP>` `<hostPort>`

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-hostport
  labels:
    name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.7.9
      ports:
        - containerPort: 80
          hostPort: 9001
```

- Drawback: hostIP can change when the container is restarted (if running in multi-node environment)

Pod Access: NodePort


- Is a type of K8S service that maps an internal port to an external one
- Port from range 30000-32767



Pod Access: NodePort

- Is a type of K8S service that maps an internal port to an external one
- Port from range 30000-32767

```
kind: Service
apiVersion: v1
metadata:
  name: nginx
spec:
  type: NodePort
  ports:
    - port: 80
      nodePort: 30023
  selector:
    name: nginx
```



```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-uob-installed-reqs
  labels:
    name: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.7.9
      ports:
        - containerPort: 80
```

LIVE DEMO

- This is the most that Minikube can do with networking
 - But what happens when a Kubernetes node goes down?
-
- Let's create our own cluster....

Creating your own Kubernetes cluster: Terraform

- It's kind of painful, needs some knowledge about interacting with the Cloud Provider and K8S architecture
- Terraform simplifies things for us

<https://www.terraform.io/>

- Set up your own Kubernetes cluster on OCI with Terraform:

<https://github.com/oracle/terraform-kubernetes-installer>



Creating your own Kubernetes cluster: OKE

- OKE (Oracle Cloud Infrastructure Container Engine for Kubernetes)
- Fully-managed, scalable, highly available service
- Used to deploy containerized application to the cloud

Creating your own Kubernetes cluster: OKE

- OKE (Oracle Cloud Infrastructure Container Engine for Kubernetes)
- Fully-managed, scalable, highly available service
- Used to deploy containerized application to the cloud

.... What it actually does

- Hosts the Control Plane on its own infrastructure
- Builds Node Pools for you to run your containers on

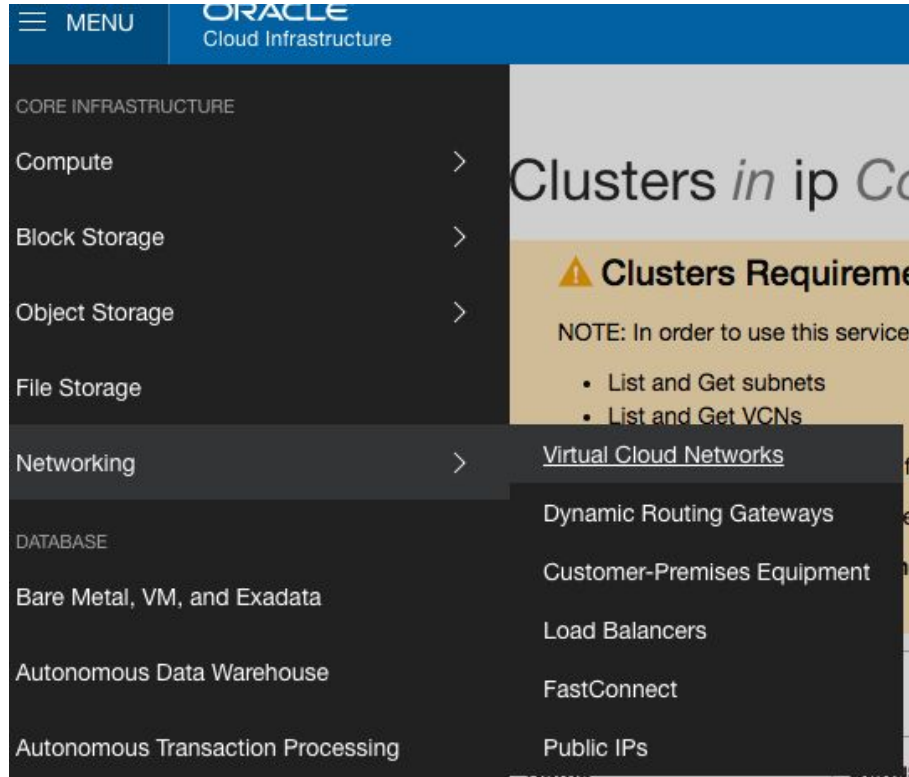
Creating your own Kubernetes cluster: OKE

1. Create your own Virtual Cloud Network (VCN)
2. Add LoadBalancer subnets
3. Create Kubernetes cluster using OKE
4. Create NodePool (worker nodes) on the generated cluster

Chocolate time



Create VCN



Create VCN

Create Virtual Cloud Network

[help](#) [cancel](#)

CREATE IN COMPARTMENT

ip

odx-mockcustomer (root)/ip

NAME OPTIONAL

uob_vcn

☐ CREATE VIRTUAL CLOUD NETWORK ONLY

☒ CREATE VIRTUAL CLOUD NETWORK PLUS RELATED RESOURCES

Automatically sets up a Virtual Cloud Network with access to the internet. You can set up firewall rules and Security Lists to control ingress and egress traffic to your Instances. All related resources will be created in the same Compartment as the VCN. These actions will occur:

Create Virtual Cloud Network

DNS RESOLUTION

☒ USE DNS HOSTNAMES IN THIS VCN

Allows assignment of DNS hostname when launching an Instance

Name: uob_vcn

CIDR: 10.0.0.0/16

DNS Label: uobvcn

DNS Domain Name: uobvcn.oraclevcn.com

Create Internet Gateway

Name: Internet Gateway

Update Default Route Table

Add Route Rule: 0.0.0.0/0 - Internet Gateway

Create Subnet

Create VCN

Create Virtual Cloud Network

Create Virtual Cloud Network

The Virtual Cloud Network was created: [uob_vcn](#)

Create Internet Gateway

The Internet Gateway "Internet Gateway uob_vcn" was created

Update Default Route Table

The Route Table was updated: [Default Route Table for uob_vcn](#)

Create Subnet

Public Subnet zkJl:US-ASHBURN-AD-1 was created

Create Subnet

Public Subnet zkJl:US-ASHBURN-AD-2 was created

Create Subnet

Public Subnet zkJl:US-ASHBURN-AD-3 was created

Close

Create Load Balancer subnets

- Ideally created in different availability domains

Create Subnet

[help](#) [cancel](#)

If the Route Table, DHCP Options, or Security Lists are in a different Compartment than the Subnet, enable Compartment selection for those resources: [Click here](#)

NAME OPTIONAL

lb1

AVAILABILITY DOMAIN OPTIONAL

zkjl:US-ASHBURN-AD-1

If you don't specify an availability domain, the subnet is regional and can contain resources in any of the availability domains.

CIDR BLOCK

10.0.5.0/24

Specified IP addresses: 10.0.5.0-10.0.5.255 (256 IP addresses)

ROUTE TABLE

Default Route Table for uob_vcn

SUBNET ACCESS

☐ PRIVATE SUBNET

Prohibit public IP addresses for Instances in this Subnet

☒ PUBLIC SUBNET

Allow public IP addresses for Instances in this Subnet

DNS RESOLUTION

☒ USE DNS HOSTNAMES IN THIS SUBNET

Allows assignment of DNS hostname when launching an Instance

DNS LABEL

lb1

Only letters and numbers, starting with a letter. 15 characters max.

DNS DOMAIN NAME (READ-ONLY)

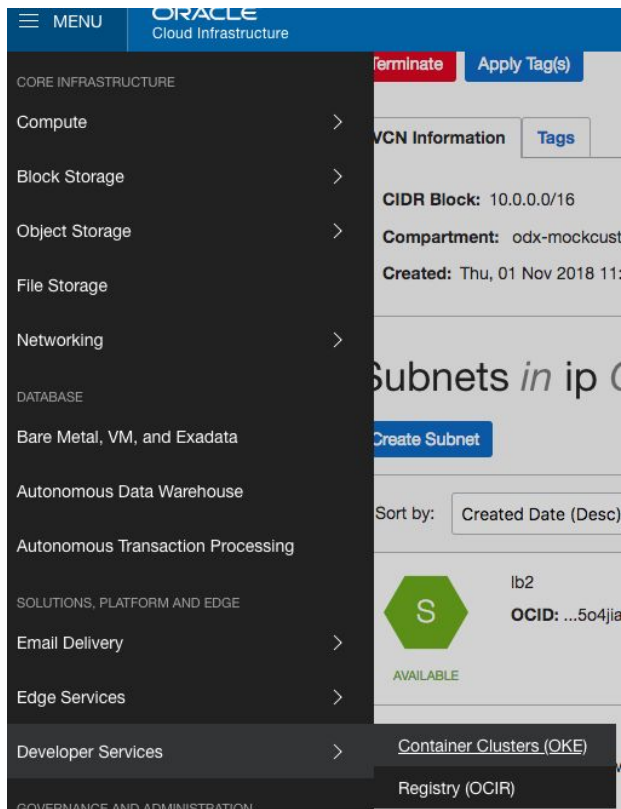
lb1.uobvcn.oraclevcn.com

DHCP OPTIONS

Select DHCP Options

Security Lists

Create OKE Cluster



Create OKE Cluster

Cluster Creation

help close

CLUSTER COMPARTMENT

ip

NAME

uob

KUBERNETES VERSION

v1.11.1

Kubernetes version install on your master nodes

☒ QUICK CREATE

Quickly create a cluster with default settings, also creates a dedicated network

☐ CUSTOM CREATE

Create a cluster with custom settings, assumes an existing network

Create Virtual Cloud Network

A new VCN network will be created for you in order to have a functioning cluster

COMPARTMENT: ip RESOURCE CREATION: 1 VCN, 2 service lb subnets and 3 worker node subnets

Create Node Pool

NAME: pool1 COMPARTMENT: ip KUBERNETES VERSION: v1.11.1 IMAGE: Oracle-Linux-7.5

SHAPE

VM.Standard1.2

The shape of all nodes in the pool

QUANTITY PER SUBNET

1

The number of nodes per subnet.

PUBLIC SSH KEY OPTIONAL

Input SSH public key

The SSH public key to access your nodes.

Don't forget to add an ssh key if you want to access your nodes

Create OKE Cluster

Creating cluster and associated network resources [close](#)

Create Virtual Cloud Network

The Virtual Cloud Network was created: [oke-vcn-quick-uob-20181101112814](#)

Create Internet Gateway

The Internet Gateway was created: oke-igw-quick-uob-20181101112814

Update Route Tables

The Route Table was updated: Default Route Table for oke-vcn-quick-uob-20181101112814

Create Security Lists

The Security List was created: oke-svclbseclist-quick-uob-20181101112814
The Security List was created: oke-seclist-quick-uob-20181101112814

Create Subnets


Public Subnet was created: oke-svclbsubnet-quick-uob-20181101112814-zkjl:US-ASHBURN-AD-1
Public Subnet was created: oke-svclbsubnet-quick-uob-20181101112814-zkjl:US-ASHBURN-AD-2
Public Subnet was created: oke-subnet-quick-uob-20181101112814-zkjl:US-ASHBURN-AD-1
Public Subnet was created: oke-subnet-quick-uob-20181101112814-zkjl:US-ASHBURN-AD-2
Public Subnet was created: oke-subnet-quick-uob-20181101112814-zkjl:US-ASHBURN-AD-3

Create Cluster

Requesting Cluster: [uob](#)

Create Node Pool


Requesting Node Pool: pool1

 Cluster and associated network resources created.

Close

Create OKE Cluster - Add NodePool

Create OKE Cluster - Retrieve kubernetes config



ACTIVE

uob

[Access Kubeconfig](#)[Delete Cluster](#)

Cluster Details

Cluster Status: ✓ Active

Node Pools: 1

Cluster Id: ...cywgmdcg5st [Show](#) [Copy](#)

Launched: Thu, 01 Nov 2018 11:31:31 GMT

Services CIDR: 10.96.0.0/16

Kubernetes Version: v1.11.1

Kubernetes Address: ...com:6443 [Show](#) [Copy](#)

VCN Name: [oke-vcn-quick-uob-20181101112814](#)

VCN Id: ...7q4tgjgq [Show](#) [Copy](#)

Pods CIDR: 10.244.0.0/16

Create OKE Cluster - Retrieve kubernetes config

How to Access Kubeconfig

[help](#) [close](#)

You must have [downloaded and installed](#) the OCI CLI and [configured](#) it for use.

To access the kubeconfig for your cluster, run the following commands:

```
1. mkdir -p $HOME/.kube
2. oci ce cluster create-kubeconfig --cluster-id
   ocid1.cluster.oc1.iad.aaaaaaaaaftdmylcmrsgkzjsmvsteobqmvsweyrxgezdozjwhcywgmdcg5st --file $HOME/.kube/config
```

More information on the available commands for OCI's Container Engine for Kubernetes CLI can be found [here](#).

Close

Create OKE Cluster - Retrieve kubernetes config

How to Access Kubeconfig

[help](#) [close](#)

You must have [downloaded and installed](#) the OCI CLI and [configured](#) it for use.

To access the kubeconfig for your cluster, run the following commands:

```
1. mkdir -p $HOME/.kube
2. oci ce cluster create-kubeconfig --cluster-id
   ocid1.cluster.oc1.iad.aaaaaaaaaftdmylcmrsgkzjsmvsteobqmvsweyrxgezdozjwhcywgmdcg5st --file $HOME/.kube/config
```

More information on the available commands for OCI's Container Engine for Kubernetes CLI can be found [here](#).

Close

Note: Will need to install and configure the oci CLI tool to communicate with OCI.

Create OKE Cluster

Done!

Let's use our cluster

Using your OKE Cluster: Kubectl

- Modify your ./kube/config with the one obtained from the Create OKE Cluster Step
- Use kubectl commands to check on the status of the cluster

```
mpatrich-Mac:~$ kubectl get nodes
NAME                                STATUS    ROLES    AGE   VERSION
129.213.53.71                       Ready     node     8m    v1.11.1
129.213.88.24                       Ready     node     8m    v1.11.1
132.145.154.41                     Ready     node     8m    v1.11.1
```

Not hosted in the same tenancy

```
mpatrich-Mac:~$ kubectl cluster-info
Kubernetes master is running at https://cywgmdcg5st.us-ashburn-1.clusters.oci.oraclecloud.com:6443
KubeDNS is running at https://cywgmdcg5st.us-ashburn-1.clusters.oci.oraclecloud.com:6443/api/v1/namespaces/kube-system/services/kube-dns:dns/proxy

To further debug and diagnose cluster problems, use 'kubectl cluster-info dump'.
```

Using your OKE Cluster: Dashboard

Resources

[Node Pools](#)

[Work Requests](#)

[Getting Started](#)

Getting Started

Kubernetes Dashboard

You can use the Kubernetes Dashboard to get an overview of applications running in your cluster. It also provides information on the state of Kubernetes resources in your clusters, and on any errors that may have occurred.

1. `kubectl proxy`
2. Dashboard will be available at:
<http://localhost:8001/api/v1/namespaces/kube-system/services/https:kubernetes-dashboard:/proxy/>

Quick Start: Deploy Sample App

1 Access Kubeconfig File

To get started, learn how to download the `kubeconfig` file for this cluster by clicking below. This file will contain a series of authentication mechanisms and cluster connection information.

[Access Kubeconfig](#)

2 Check Version

Verify that kubernetes is available by entering the following command in your terminal

1. `kubectl version`




3 Deploy Application

Deploy a sample hello world application by running the following command in your terminal.

1. `kubectl create -f https://k8s.io/docs/tasks/run-application/deployment.yaml`

Using your OKE Cluster: Dashboard

Nodes

Name	Labels	Ready	CPU requests (cores)	CPU limits (cores)	Memory requests (bytes)	Memory limits (bytes)	Age
 129.213.88.24	<div><div>beta.kubernetes.io/arch: amd64</div><div>beta.kubernetes.io/instance-type: V...</div><div>beta.kubernetes.io/os: linux</div><div>displayName: oke-cywgmddcg5st-nzt...</div><div>failure-domain.beta.kubernetes.io/re...</div><div>show all</div></div>	True	0.1 (2.50%)	1 (25.00%)	50 Mi (0.36%)	500 Mi (3.64%)	20 minutes
 129.213.53.71	<div><div>beta.kubernetes.io/arch: amd64</div><div>beta.kubernetes.io/instance-type: V...</div><div>beta.kubernetes.io/os: linux</div><div>displayName: oke-cywgmddcg5st-nzt...</div><div>failure-domain.beta.kubernetes.io/re...</div><div>show all</div></div>	True	0.38 (9.50%)	1 (25.00%)	170 Mi (1.24%)	670 Mi (4.87%)	20 minutes
 132.145.154.41	<div><div>beta.kubernetes.io/arch: amd64</div><div>beta.kubernetes.io/instance-type: V...</div><div>beta.kubernetes.io/os: linux</div><div>displayName: oke-cywgmddcg5st-nzt...</div><div>failure-domain.beta.kubernetes.io/re...</div><div>show all</div></div>	True	0.62 (15.50%)	1 (25.00%)	270 Mi (1.96%)	840 Mi (6.11%)	20 minutes

Going back to our nginx app

- Quickly recreate what we've done on the Minikube cluster
 - Create pod
 - Create nodeport
 - Check connection (use any IP address of the NodePool nodes):

```
mpatrich-Mac:firstpod mpatrich$ curl http://132.145.154.41:30023
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

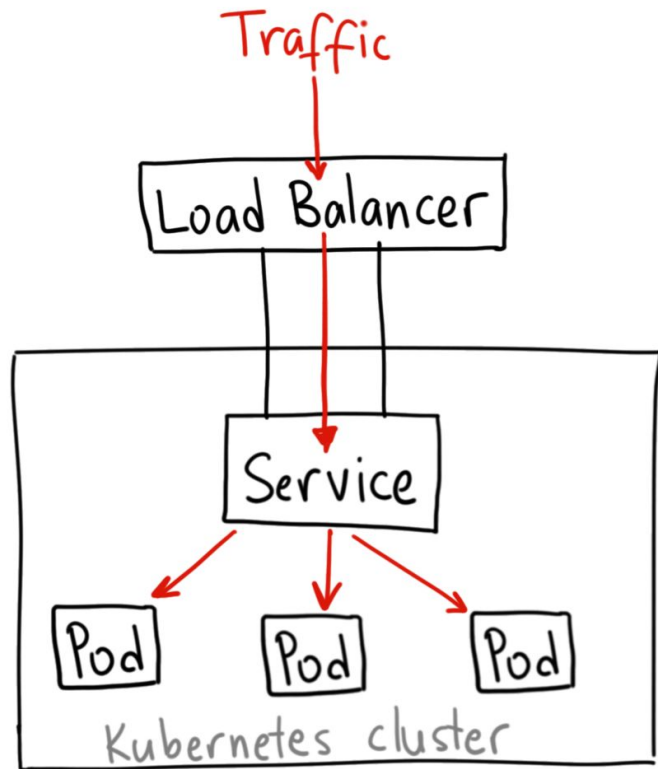
<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Nginx app: Add Load Balancer

- But what if one of our NodePool nodes goes down?

We need a reliable way of accessing our app with a
LoadBalancer Service



Nginx app: Add LoadBalancer

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-lb
spec:
  ports:
    - port: 8000 # the port that this service should serve on
      # the container on each pod to connect to, can be a name
      # (e.g. 'www') or a number (e.g. 80)
      targetPort: 80 # Pod port to target
      protocol: TCP
  # just like the selector in the deployment,
  # but this time it identifies the set of pods to load balance
  # traffic to.
  selector:
    app: nginx # Matches the pod app label
  type: LoadBalancer
```

Port exposed to the outside world

Pod port to target

Matches the pod app label

Nginx app: Add LoadBalancer

```
mpatrich-Mac:okecluster mpatrich$ kubectl get service
```

NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
kubernetes	ClusterIP	10.96.0.1	<none>	443/TCP	1h
nginx	NodePort	10.96.101.85	<none>	80:30023/TCP	54m
nginx-lb	LoadBalancer	10.96.120.175	129.213.195.69	8000:32121/TCP	28m

Nginx app: Add LoadBalancer

Let's test it!

```
mpatrich-Mac:okecluster mpatrich$ curl 129.213.195.69:8000
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
  body {
    width: 35em;
    margin: 0 auto;
    font-family: Tahoma, Verdana, Arial, sans-serif;
  }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
<p>If you see this page, the nginx web server is successfully installed and
working. Further configuration is required.</p>

<p>For online documentation and support please refer to
<a href="http://nginx.org/">nginx.org</a>.<br/>
Commercial support is available at
<a href="http://nginx.com/">nginx.com</a>.</p>

<p><em>Thank you for using nginx.</em></p>
</body>
</html>
```

Nginx app: Adding more pods

- But what if the pod goes down?
- Ideally we would want to have multiple replicas of the same pod

Nginx app: Adding more pods

- But what if the pod goes down?
- Ideally we would want to have multiple replicas of the same pod

We need a way of managing the multiple replicas of our app
with a *Deployment*

Nginx app: Adding more pods

- But what if the pod goes down?
- Ideally we would want to have multiple replicas of the same pod

We need a way of managing the multiple replicas of our app
with a *Deployment*

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx-dep
spec:
  replicas: 4
  selector:
    matchLabels:
      app: nginx-dep
  template:
    metadata:
      labels:
        app: nginx-dep
    spec:
      containers:
        - name: nginx-dep
          image: nginx:1.7.9
          ports:
            - containerPort: 80
```

Homework

https://github.com/MadalinaPatrichi/uob-cloud-computing/blob/master/slides/week_4/part_2/Deploying_TodoApp_in_Kubernetes.pdf

Useful links

- Katacoda: <https://www.katacoda.com/courses/kubernetes>
- Minikube: <https://kubernetes.io/docs/tutorials/hello-minikube/>
- Oracle Cloud Infrastructure: https://cloud.oracle.com/en_US/tryit
- Terraform: <https://www.terraform.io/>
- Create your own Kubernetes cluster on OCI:
<https://github.com/oracle/terraform-kubernetes-installer>