Kubernetes 101

Workshop by Renana Yacobi

What you'll need:

- **Docker Desktop** - follow the instructions to install docker desktop for your operating system at: https://www.docker.com/get-started
- **minikube** - follow the instructions to install minikube for your operating system at: https://minikube.sigs.k8s.io/docs/start/
- **Recommended**: Text editor with yaml support. We will be using IntelliJ (the community edition)

Workshop materials

- All the materials needed for the workshop can be found at: https://github.com/renana/kubernetes-101
- clone or download the repository

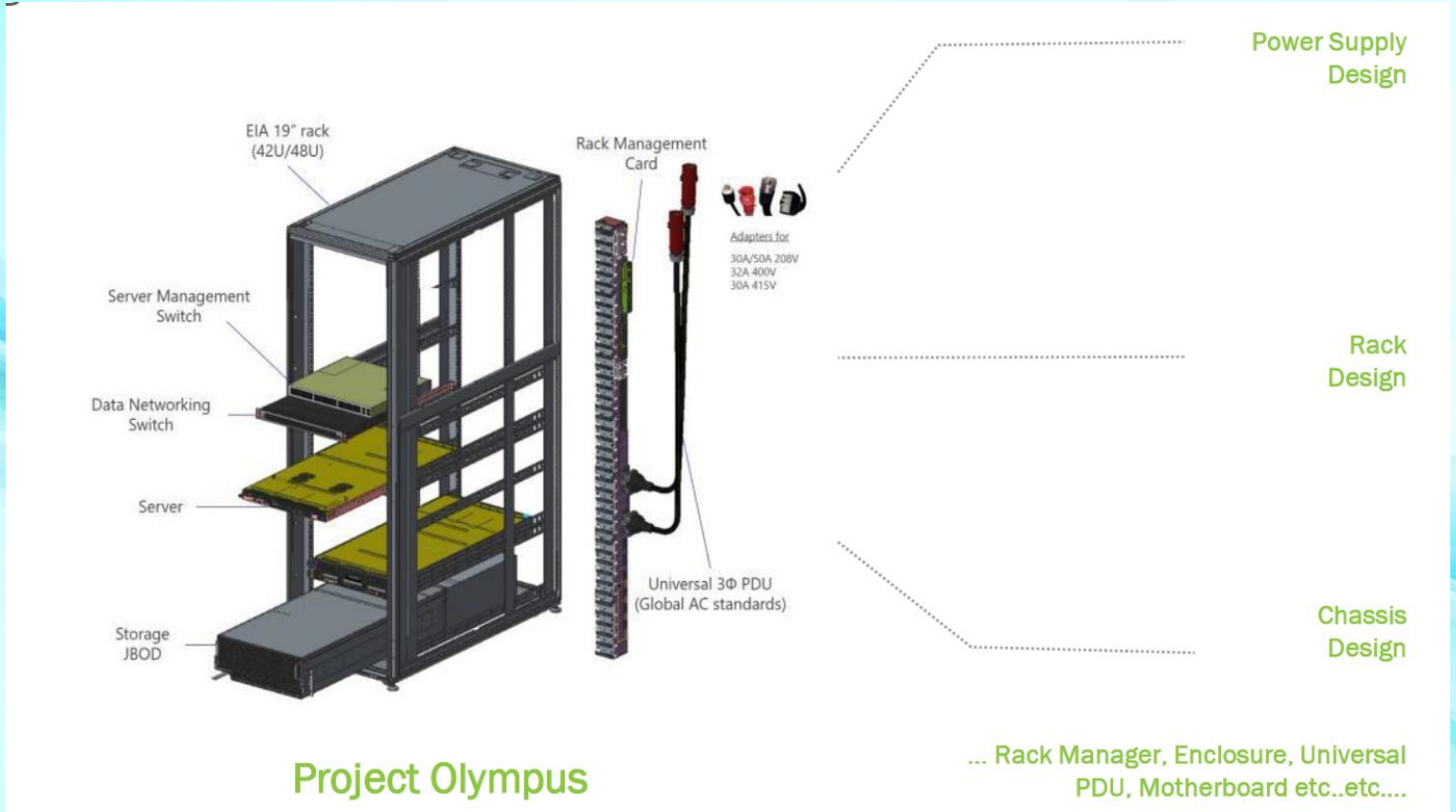
What is kubernetes and why we care

- Kubernetes (K8s) is an open-source system for automating deployment, scaling, and management of containerized applications.
- Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and services, that facilitates both declarative configuration and automation. It has a large, rapidly growing ecosystem. Kubernetes services, support, and tools are widely available.

It is all about the cloud

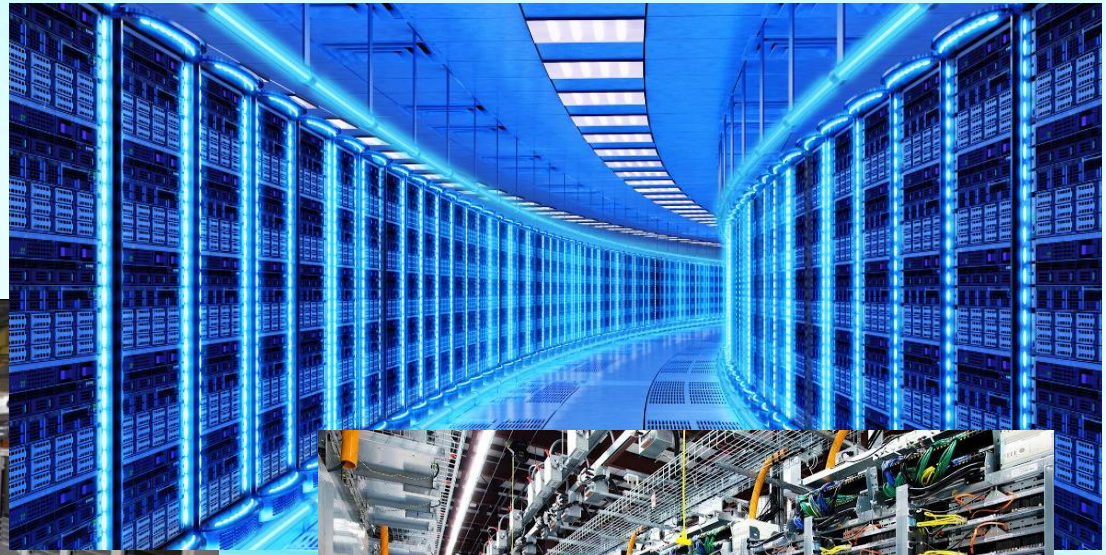
- If you want to learn more about what is this cloud all about - Sailing your fleet through the streams of the cloud is focused on cloud networks but covers related issues like….
- Compute resources - provides virtual equipment by combining CPUs, Memory, and Disks to create Virtual Machines. Compute Resources are provided by virtualizing physical servers and storage devices shared by multiple users.

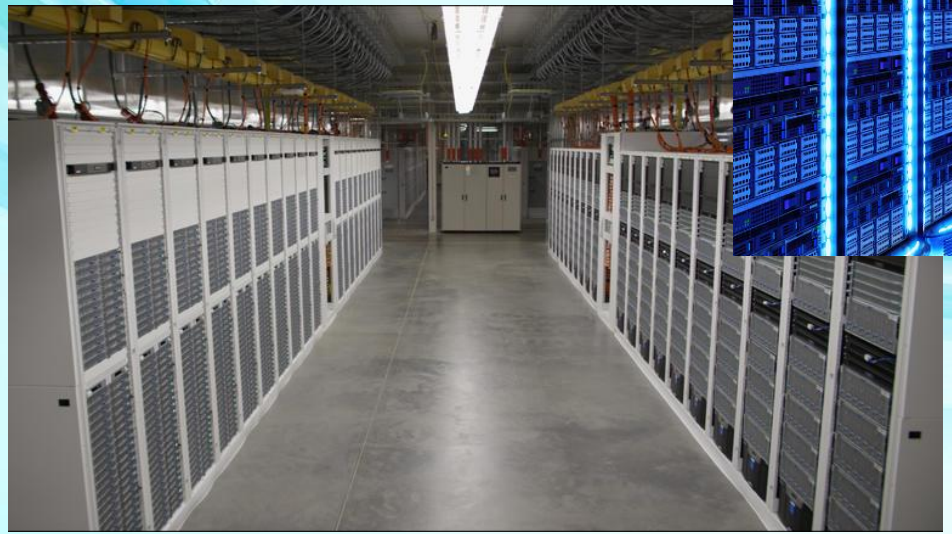
Racks

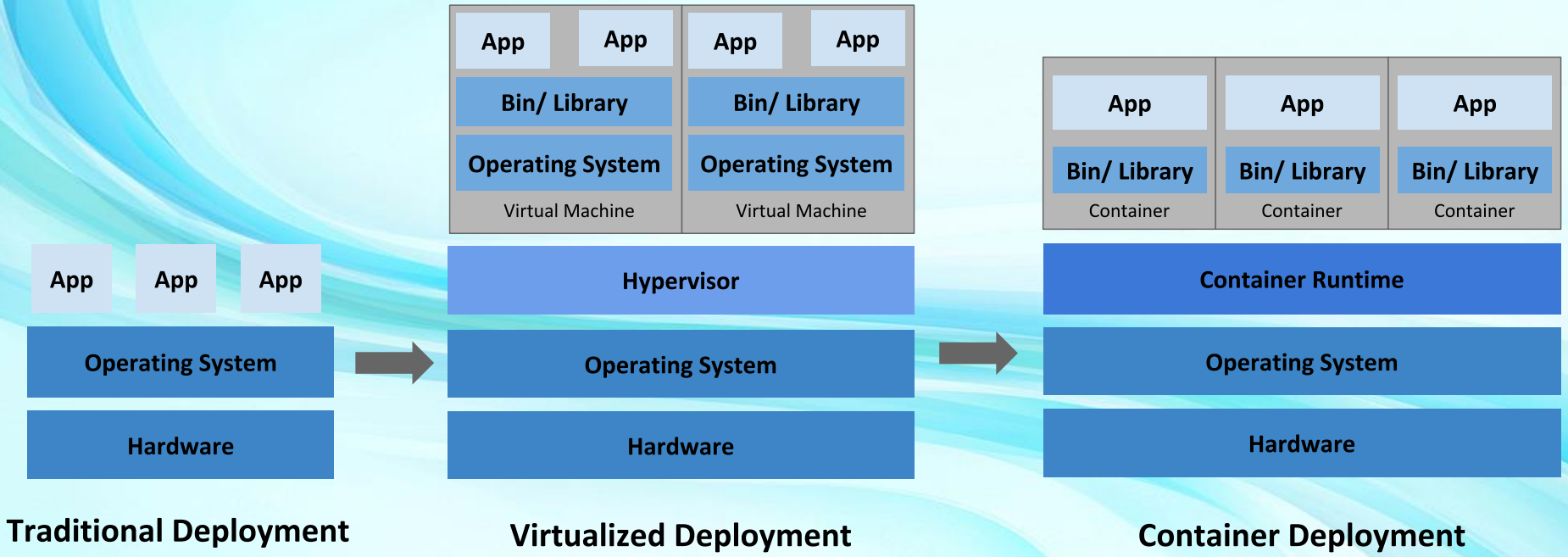


Source: http://dailychipdigest.blogspot.com/p/server-fundamentals.html

Inside a data center



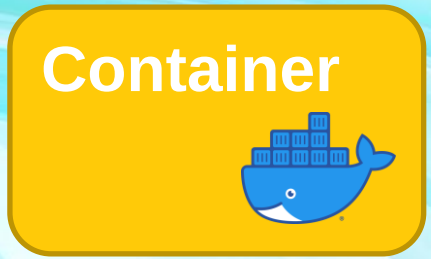



App
App
App
Operating System
Hardware
Traditional Deployment
App
App
App
App
Bin/ Library
Bin/ Library
Operating System
Operating System
Virtual Machine
Virtual Machine
Hypervisor
Operating System
Hardware
Virtualized Deployment
App
App
App
Bin/ Library
Bin/ Library
Bin/ Library
Container
Container
Container
Container Runtime
Operating System
Hardware
Container Deployment

Starting small

- Starting with the smaller units and building up:
  - Images and containers
  - Pods
  - Node
  - cluster

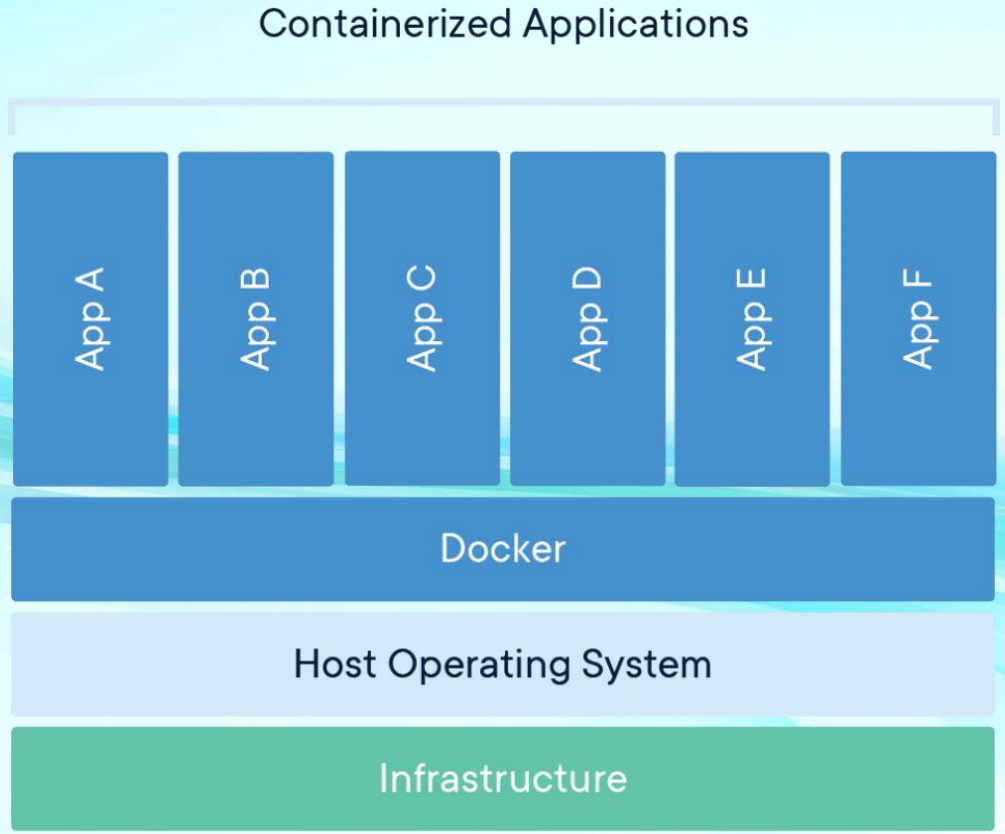
Docker



Docker

- A container is a standard unit of software that packages up code and all its dependencies, so the application runs quickly and reliably from one computing environment to another. A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings.
- Container **images** become **containers** at runtime and in the case of Docker containers - images become containers when they run on Docker Engine.

Docker

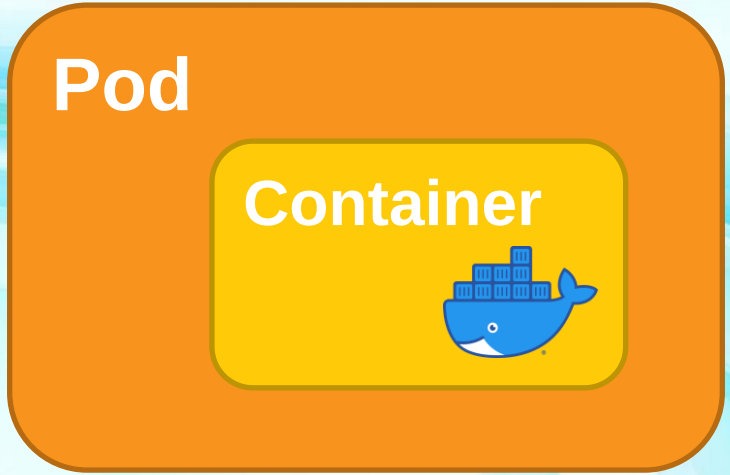

Containerized Applications
App A
App B
App C
App D
App E
App F
Docker
Host Operating System
Infrastructure

Container

We are going to use an http-echo server by Hashicorp.

- docker pull hashicorp/http-echo’
- docker run -p 8080:8080 hashicorp/http-echo -listen=:8080 -text="hello world“
- docker ps
- docker stop <container id>
- docker rm <container id>

Kubernetes Pods

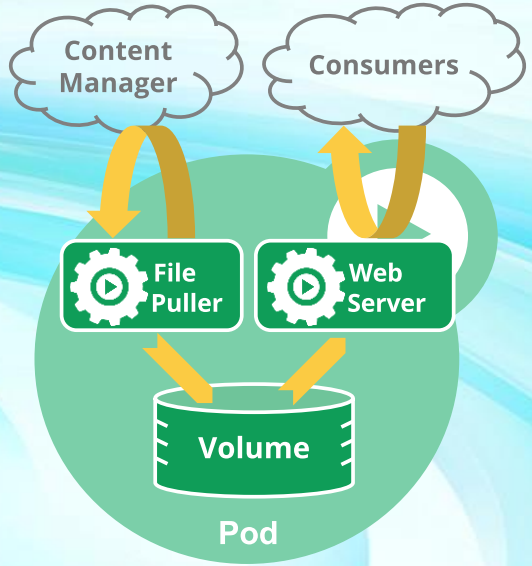


Pods

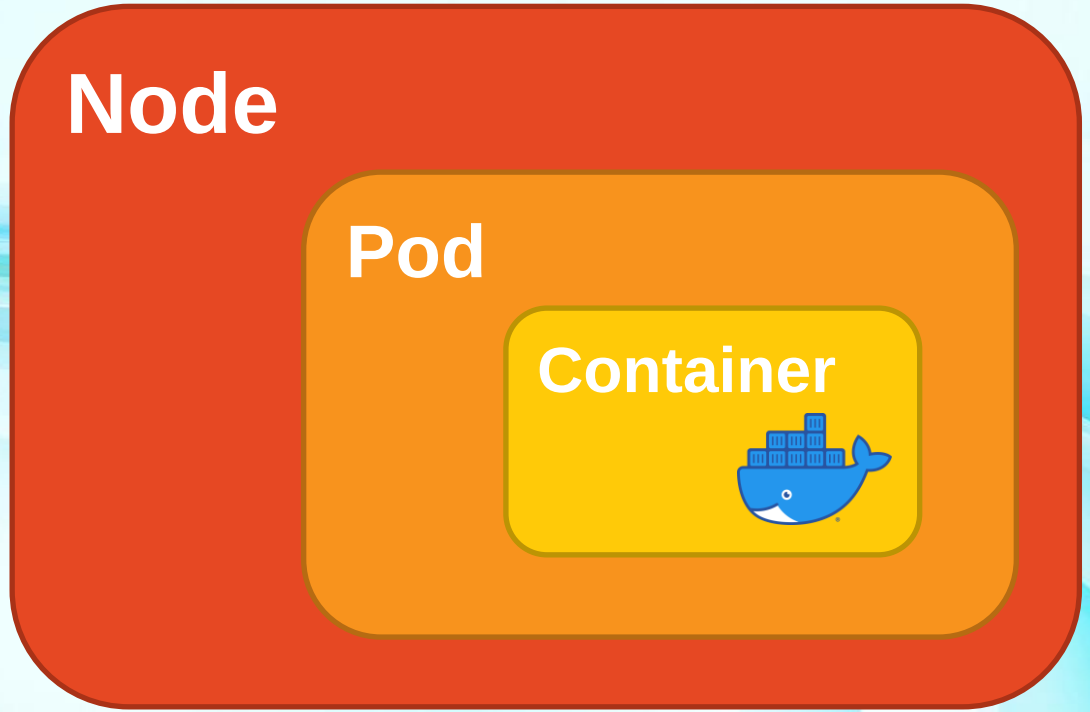
- The smallest deployable unit in Kubernetes
- Includes:
  - A group of one or more (docker) containers
  - Shared storage/network resources
  - Specification for how to run the containers

Things you should know

- Pod's contents are always co-located and co-scheduled, and run in a shared context
- Each pod can contain a single container (the common case) or several containers that are tightly coupled
- You usually don’t create pods directly but by using “workload resources”



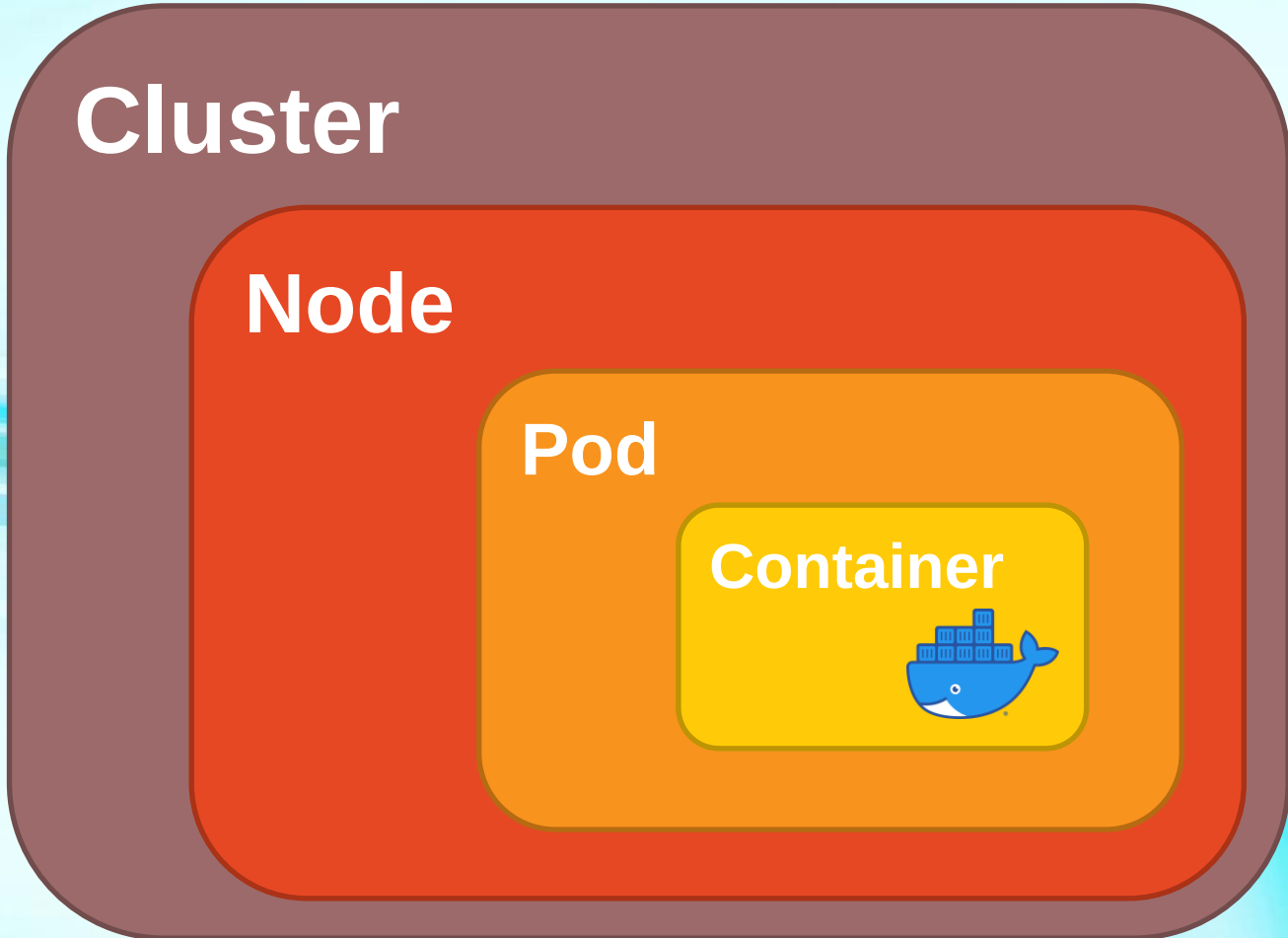
Kubernetes Nodes

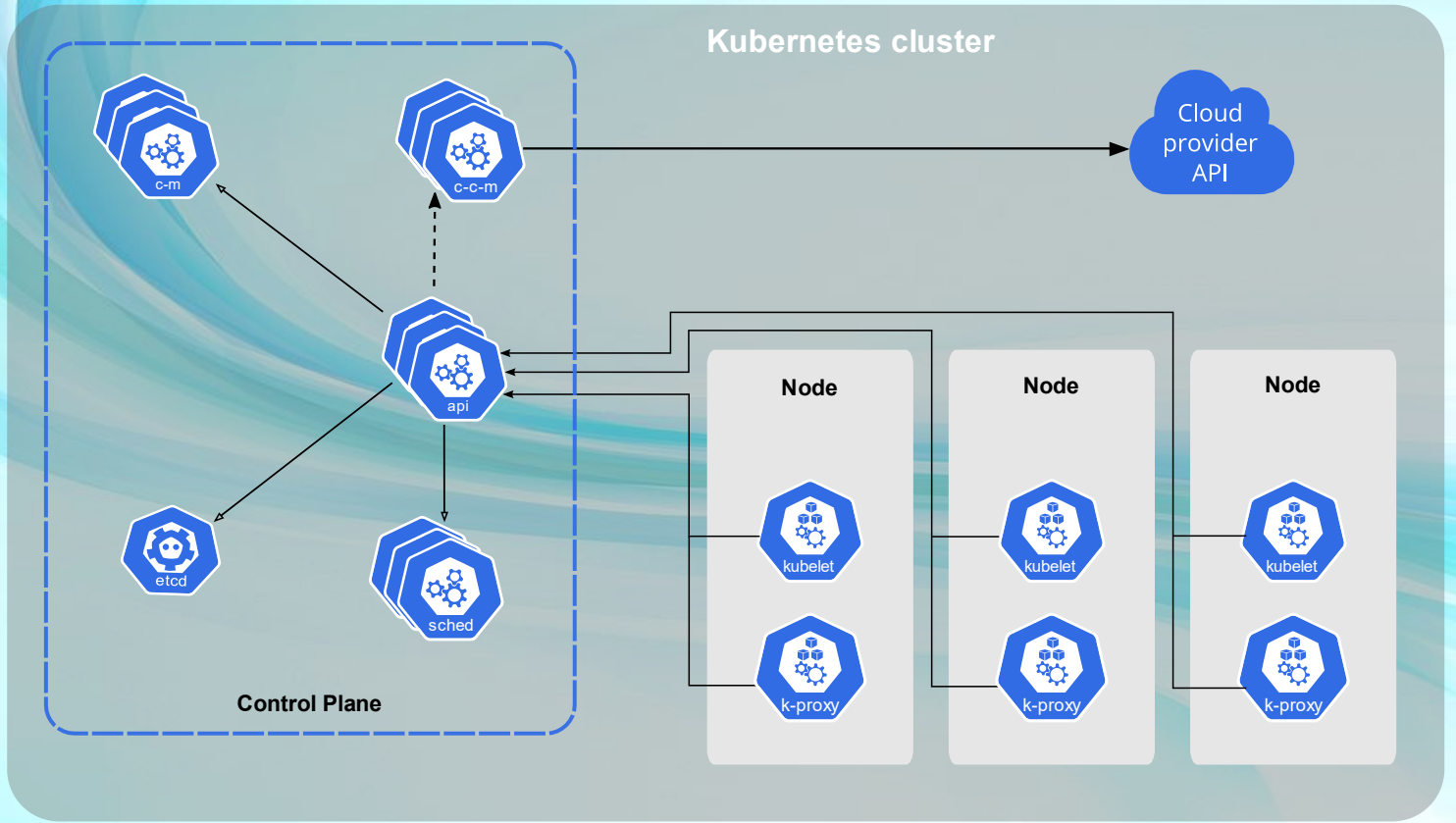


Nodes

- Kubernetes runs your workload by placing containers into pods to run on nodes.
- A node may be a virtual or physical machine, depending on the cluster.
- Each node contains the services necessary to run pods, managed by the control plane.

Kubernetes Cluster





API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

kube-proxy

Scheduler

Control plane

Node

Kubernetes can be deployed to:

- Cloud providers such as Google, Microsoft, AWS but many others.
- On prem – to your own servers
- On your personal computer

minikube

- “minikube is local Kubernetes, focusing on making it easy to learn and develop for Kubernetes.”

- Run the command “**minikube start**”
- Use **“minikube status”** to check all is working well.

- **After** the workshop will end remember to run **“minikube stop”**

kubectl – Kubernetes CLI

kubectl

- “A command line tool for communicating with a Kubernetes API server.”
- Kubernetes CLI
- Operates on context.
- Let’s explore some of kubectl commands…

kubectl get

- Your can ‘get’ almost anything you want through this command:
  - pods -> ‘kubectl get pods –A’
  - Services -> ‘kubectl get services –A’
  - Secrets -> ‘kubectl get secrets –A’
- Run ‘kubectl help get’
- Or just ‘kubectl api-resources’ to find all the things you can ‘get’.

kubectl get

- What are you getting?
  - Existence confirmation
  - Status of resource
  - Configuration of resource

Namespaces

- Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called namespaces.
- Namespaces provide a scope for names. Names of resources need to be unique within a namespace, but not across namespaces. Namespaces cannot be nested inside one another and each Kubernetes resource can only be in one namespace.
- Because of the above it is mainly used to separate between different management units.

Namespaces

- Like folders in your cluster
- Pods in a namespace are isolated from pods in other namespaces

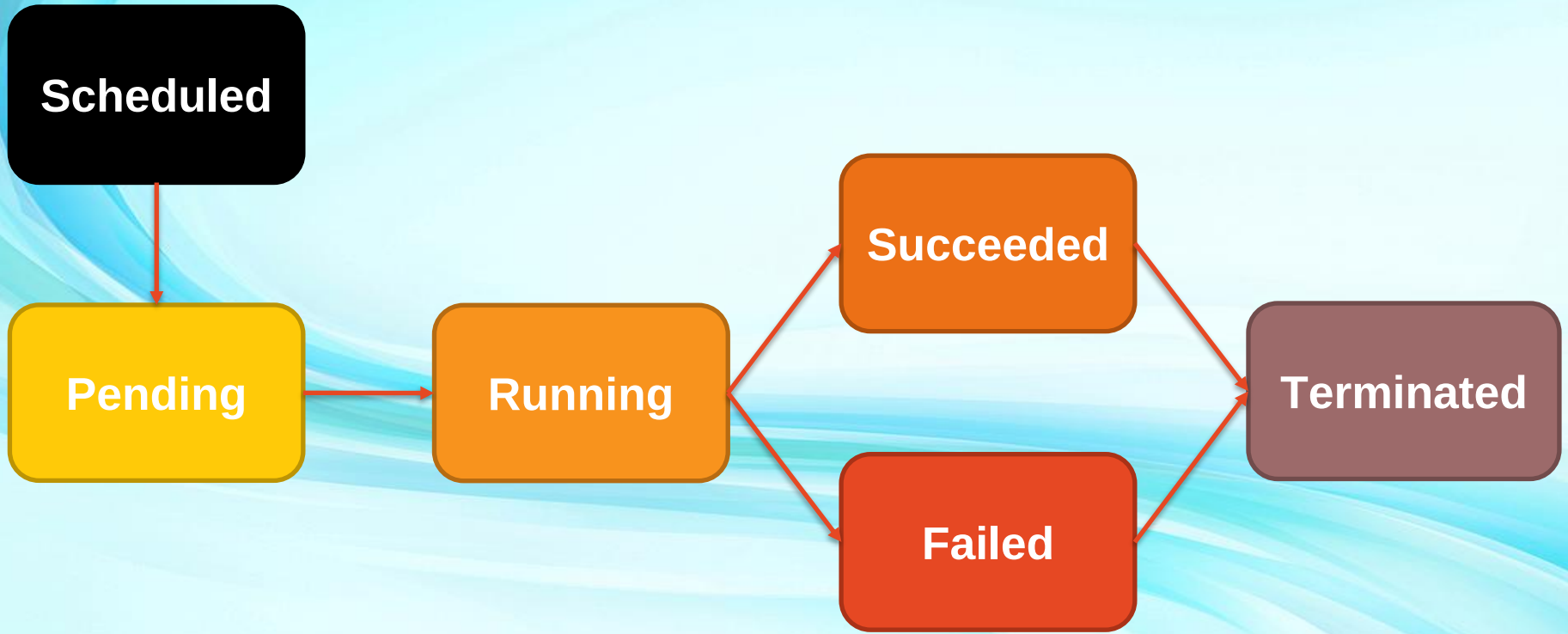
- To specify a namespace use the flag –n with kubectl as follow:
  - kubectl get pods –n kube-system

Other important kubectl commands

- kubectl describe … - similar to ‘get’ command but provides mainly the state of the object
- kubectl apply –f <object-definition.yaml> - change the state of your cluster to match the state defined in your yaml file
- kubectl delete –f <object-definition.yaml> - delete the object and resources defined in the yaml file

- https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Pod’s Lifecycle


Scheduled
Pending
Running
Succeeded
Failed
Terminated

Kubernetes Deployments

Deployment

- A deployment allows you to describe an application’s life cycle, such as which images to use for the app, the number of pods there should be, and the way in which they should be updated.
- After an object has been created, the cluster works to ensure that the object exists, maintaining the desired state of your Kubernetes cluster.

Deployment configuration

- We will use:
  - yaml files to write, store and maintain the configuration
  - “kubectl apply –f” command to apply the configuration in the cluster

Let’s practice

Kubernetes Labels & Selectors

Associating objects

- We use labels to store identifying information
- Each label has a key and value, for example:
  - “app: http-echo” – app is the key and http-echo is the value.
- Using label selector we can now select the objects we want
  - Try to find our newly created deployment and pods
  - Use “-l <key>=<value>” in your commands

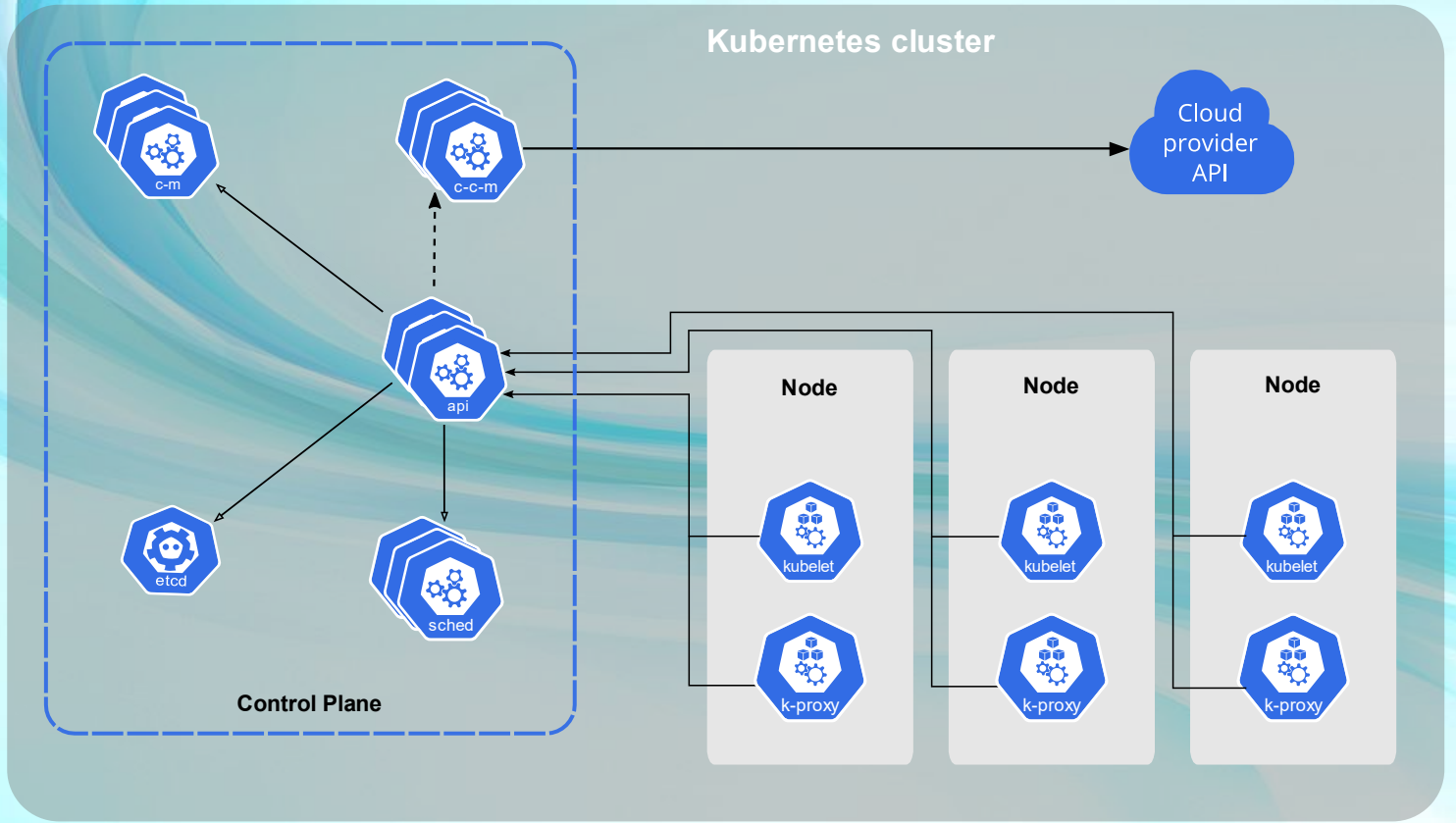
Kubernetes Services

Why do we need services

- Kubernetes Pods are created and destroyed to match the state of your cluster. Pods are nonpermanent resources. If you use a Deployment to run your app, it can create and destroy Pods dynamically.
- Each Pod gets its own IP address, however in a Deployment, the set of Pods running in one moment in time could be different from the set of Pods running that application a moment later.
- This leads to a problem: if some set of Pods (call them "backends") provides functionality to other Pods (call them "frontends") inside your cluster, how do the frontends find out and keep track of which IP address to connect to, so that the frontend can use the backend part of the workload?

Services

- 3 different types for different purposes:
  - ClusterIP - Exposes the Service on a cluster-internal IP. Choosing this value makes the Service only reachable from within the cluster. This is the default ServiceType.
  - NodePort - Exposes the Service on each Node's IP at a static port (the NodePort). A ClusterIP Service, to which the NodePort Service routes, is automatically created.
  - LoadBalancer - Exposes the Service externally using a cloud provider's load balancer. NodePort and ClusterIP Services, to which the external load balancer routes, are automatically created.



API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

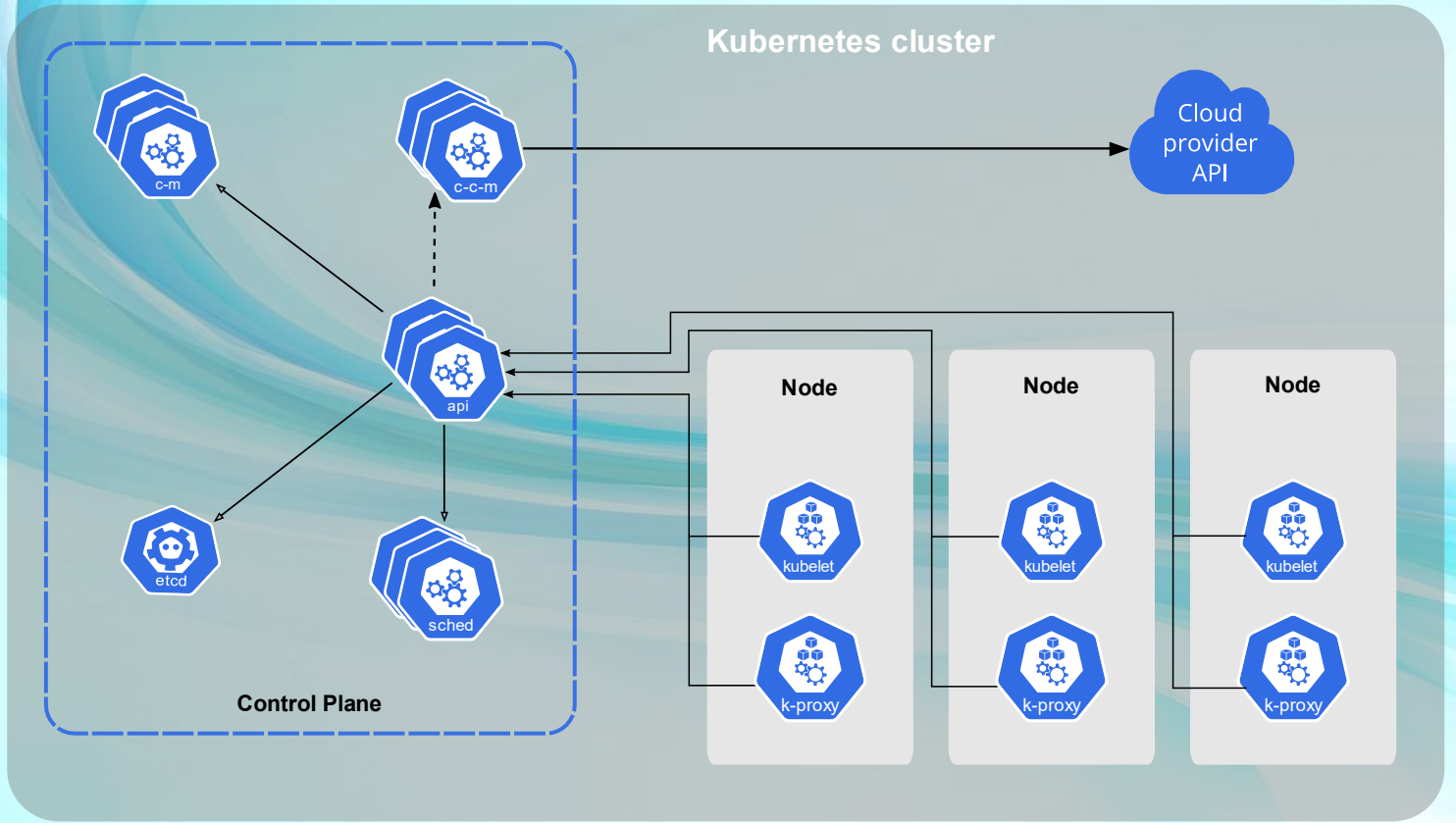
kube-proxy

Scheduler

Control plane

Node

Kubernetes Control Plane



API server

Cloud controller manager *(optional)*

Controller manager

etcd (persistence store)

kubelet

kube-proxy

Scheduler

Control plane

Node

Kubernetes control plane

- The API server - “The API server exposes an HTTP API that lets end users, different parts of your cluster, and external components communicate with one another.” -> the cluster’s communication operator.
    - Current implantation - kube-apiserver
- Key value store – “used as Kubernetes' backing store for all cluster data”
    - Current implementation – etcd
- kube-scheduler – assigns pods to nodes
- kube-controller-manager – responsible to run the different controllers of the system
- cloud-controller-manager – responsible to run only the cloud provider’s controllers

Deploying to the cloud

gcloud

- Establish an account
- Install Google Cloud SDK - https://cloud.google.com/sdk/docs/install
- Create Google Cloud Project (gcp)
- Enable billing
- Enable Google Kubernetes Engine (gke)
- Create a new cluster

Set kubectl to the right context

- Run the following to configure kubectl to connect to your newly created cluster:
  - gcloud config set project <gcp-id>
  - gcloud container clusters get-credentials <cluster-name>
- Verify the context is set correctly by running:
  - kubectl config current-context

Deploying our cluster

- kubectl apply –f echo-deployment.yaml
- kubectl apply –f echo-service.yaml

- Wait for service to be assigned an ip and then go to:
- http://<new-ip>:8080