kubernetes

Introduction to Kubernetes

Kubernetes

Course Objectives

In this module, you will learn:

- What and Why of Kubernetes
- Architecture Components
- Kubernetes deployment options
- Setting up Minikube

kubernetes

What & Why of Kubernetes?

Introduction to Kubernetes

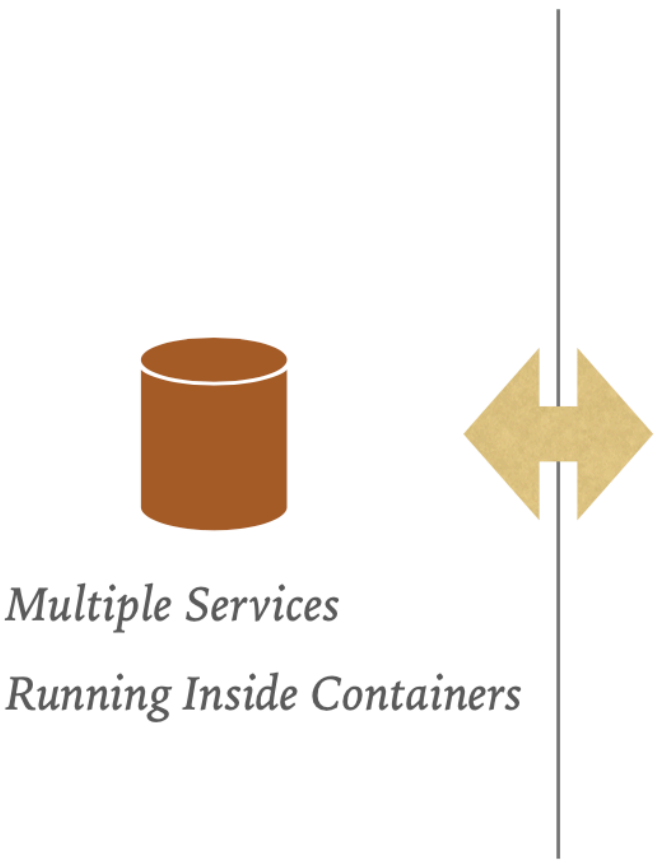
What is Kubernetes?

- Kubernetes is an open-source container orchestration system
  - Typical use case involves a containerized application running on a cluster of machines
  - Extremely popular and active open-source project
- Originally developed by Google to run Google's data centres
  - Designed to operate at Google scale
  - Proven and tested running Google's applications
  - Donated to the Cloud Native Computing Foundation in 2015

Managed Kubernetes Service

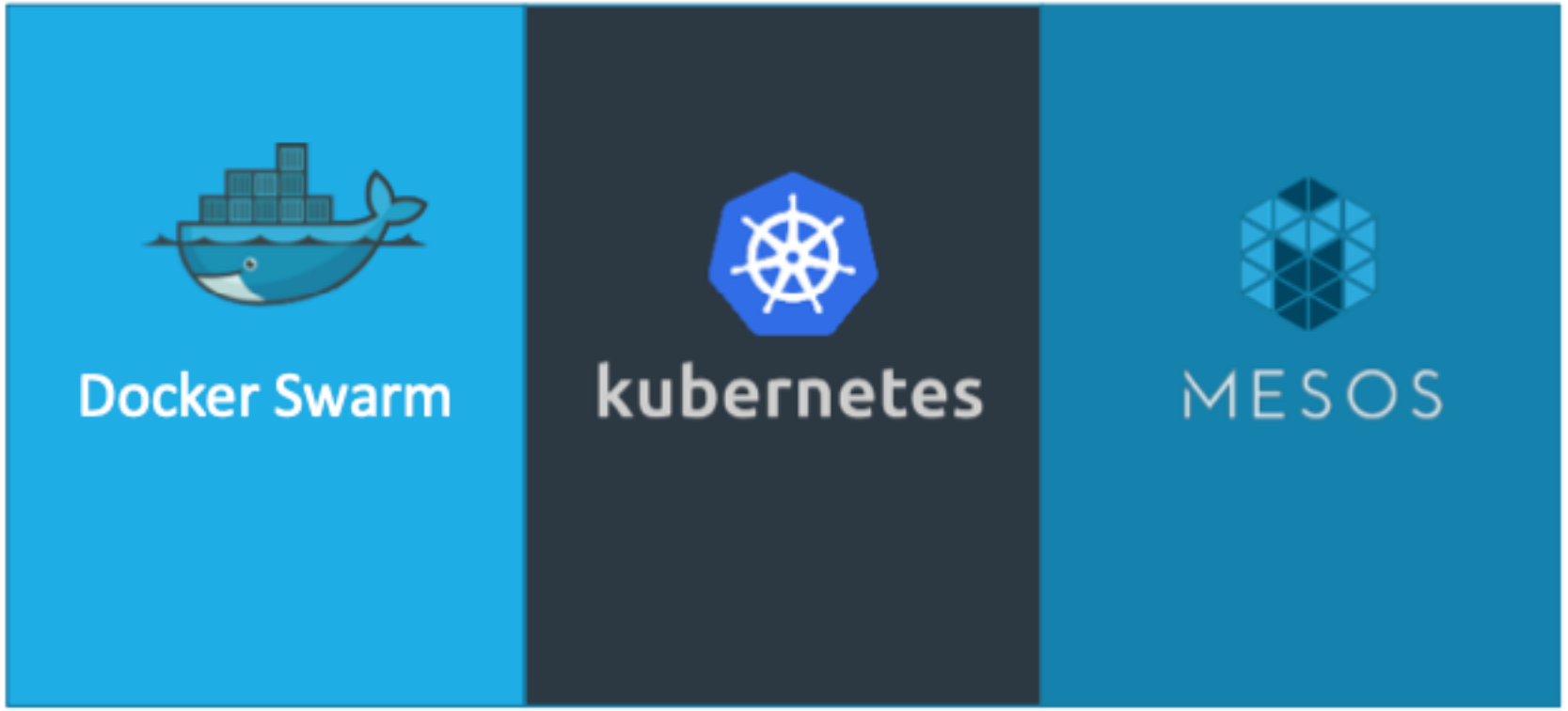
- Kubernetes has wide support from cloud vendors including:
  - Google Kubernetes Engine (GKE)
  - AWS Elastic Kubernetes Service (EKS)
  - Microsoft Azure Container Service (AKS)
  - Red Hat OpenShift
- Managed Kubernetes services typically provide:
  - Automated cluster creation and maintenance
  - Clusters implemented as a collection of VMs
  - Very simple creation using a web console or command line

Challenges without Orchestration



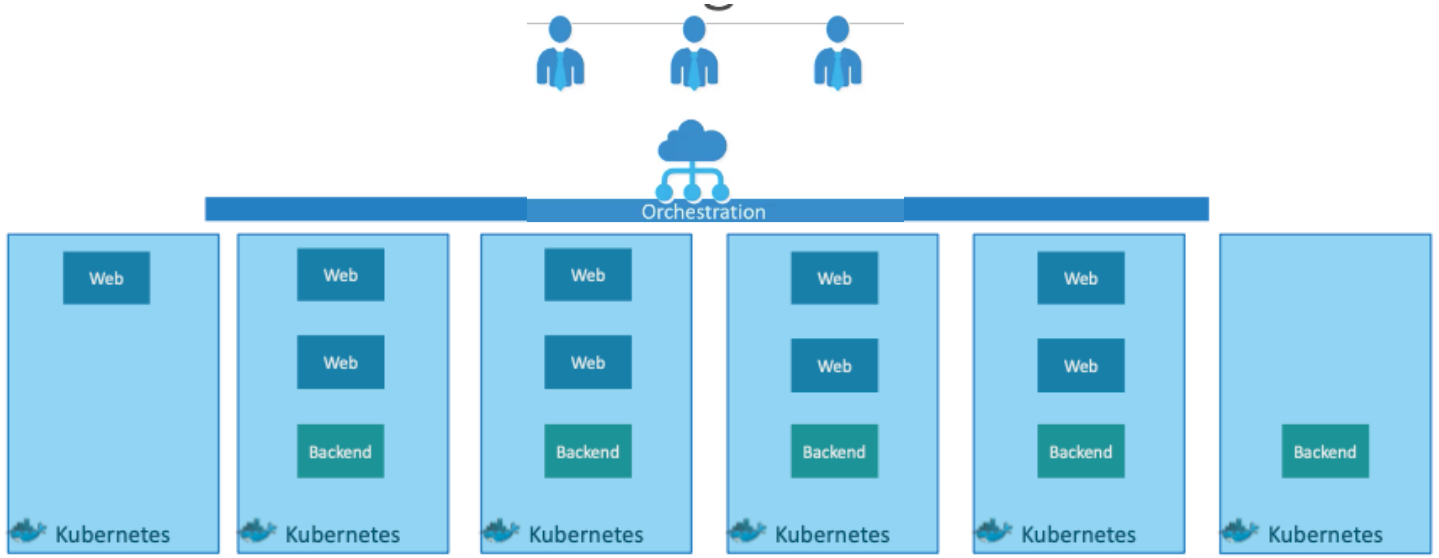
- Increase the Human Cost of Running Service
- Increase the Bills from Cloud Service Provider
- Increase Complexity of Infrastructure
- Scaling was very Difficult
- Setting Up Services Manually
- Manual Fix if any Node is Crashed

Orchestration Technologies



Kubernetes Advantage

- Automated Scheduling
- Healing Capabilities
- Rolling Upgrades & Rollback
- Horizontal Scaling
- Run Anywhere

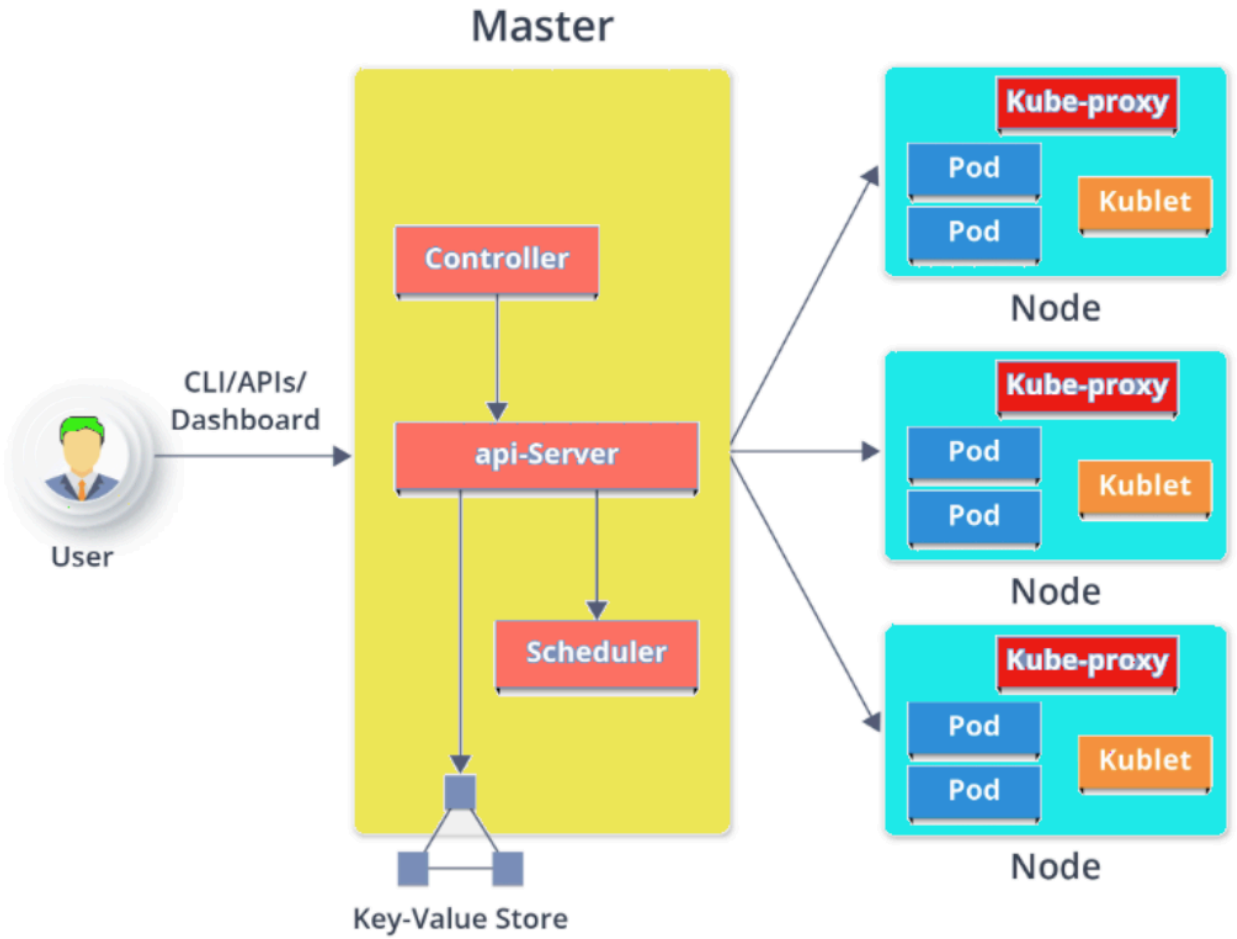


kubernetes

Architecture

Introduction to Kubernetes

Architecture Components



Master vs. Worker Node

kubectl

```
kubectl run hello-minikube
```

```
kubectl cluster-info
```

```
kubectl get nodes
```

Worker Node

- **Worker Node :** It's a physical server or you can say a VM which runs the applications using Pods.
- Worker nodes contain all the necessary **services to manage** the **networking** between the containers, communicate with the master node, and assign resources to the scheduled containers.

Worker Node - Kubelet

- **Kubelet :** It is an agent which communicates with the Master node and executes on worker nodes.
- Kubelet gets the PodSpec configuration of a Pod from the API server and ensures that the described containers are up and running and are healthy.
- It talks to the Docker daemon using the API over the Docker socket to manipulate containers lifecycle
- It then reports back to the API Server the status of changes regarding those pods.
- The Kubelet ships with built-in support for cAdvisor, which collects, aggregates, processes and exports metrics (such as CPU, memory, file and network usage) about running containers on a given node.
  - It role is to Retrieve container metrics from cAdvisor, aggregate and expose them through the Kubelet Summary API
- The kubelet doesn't manage containers which were not created by Kubernetes.

Worker Node - Pods

- **Pods :** Is a group of one or more containers with shared storage/network, and a specification for how to run the containers.
- The smallest and simplest unit in the Kubernetes object model that you create or deploy.
- Pods run on nodes and run together as a logical unit.
- Single Pod can Run on Multiple Machines and Single Machine can Run Multiple Pods.
- Each pod has a unique IP address assigned to it.
- If a pod is running multiple containers, then the containers can communicate with each other using localhost. When they have to communicate outside the Pod, they expose a port.

Worker Node – Kube-Proxy

- **Kube-Proxy :** Kube-proxy runs on each node to deal with individual host sub-netting and ensure that the services are available to external parties.
- Kube-proxy acts as a network proxy and a load balancer for a service on a single worker node.
- It is the network proxy which runs on each worker node and listens to the API server for each Service endpoint creation/ deletion.

Master Node

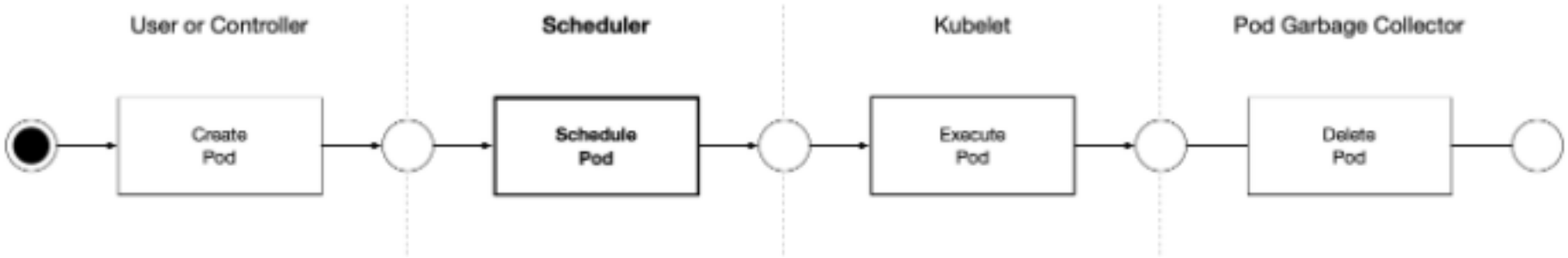
- Kubernetes follow the Master - Slave(Worker) Node Architecture.
- **Master Node :** Responsible for the management of Kubernetes cluster. It is mainly the entry point for all administrative tasks.
- More than One Master Nodes be the there in Kubernetes Cluster.
- In **Multi Master Node System**, Single Master node will be commanding Node for own workers and other Masters too.

Master Node – API Server, etcd

- **API Server :** API server is the entry point for all the REST commands used to control the cluster.
- Main Master Node uses **etcd** to manage the Workers and Other Master Nodes.
- Stores the state of the cluster in the **distributed key-value store**.
- Can be a part of Kubernetes Master or can be configured externally
- It is mainly used for shared configuration and Service Discovery

Master Node – Scheduler

- **Scheduler :** Schedules the work in the form of Pods and Services
- The Kubernetes Scheduler is a core component of Kubernetes: After a user or a controller creates a Pod, the Kubernetes Scheduler, monitoring the Object Store for unassigned Pods, will assign the Pod to a Node.
- Then, the Kubelet, monitoring the Object Store for assigned Pods, will execute the Pod.



Master Node - Controller

- **Controller :** Regulates the Kubernetes cluster which manages the different non-terminating control loops.
- Controller watches the desired state of the objects it manages and watches their current state through the API server.
- If the current state of the objects it manages does not meet the desired state, then the control loop takes corrective steps to make sure that the current state is the same as the desired state.
- Performs lifecycle functions such as namespace creation, event garbage collection, node garbage collection, etc.

Setting up Kubernetes

Setting up Kubernetes

Setup Kuberetes

Minikube

MicroK8s MicroK8s

Kubeadm

Google Cloud Platform

AWS Amazon Web Services

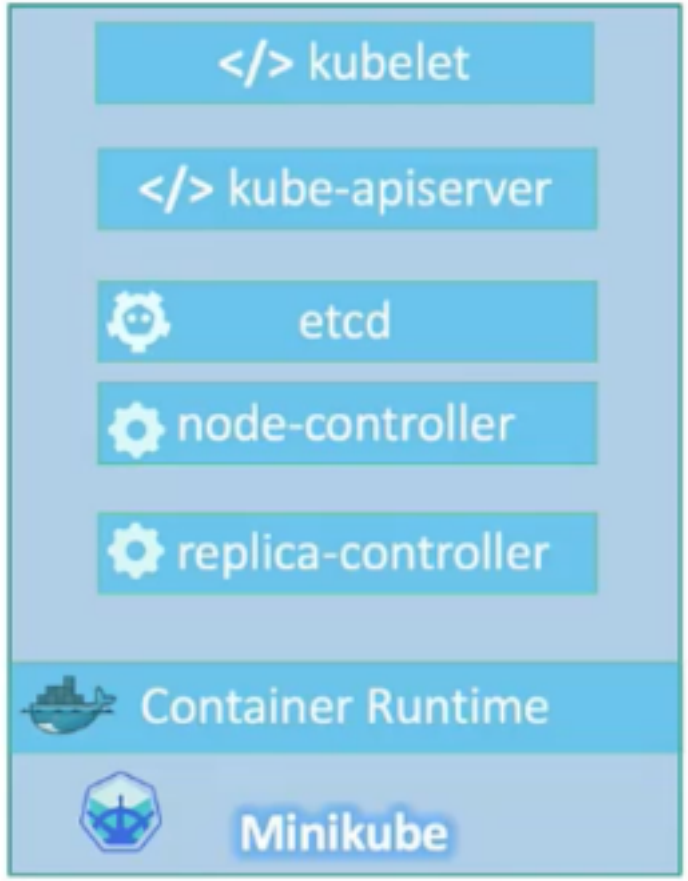
Microsoft Azure

play-with-k8s.com

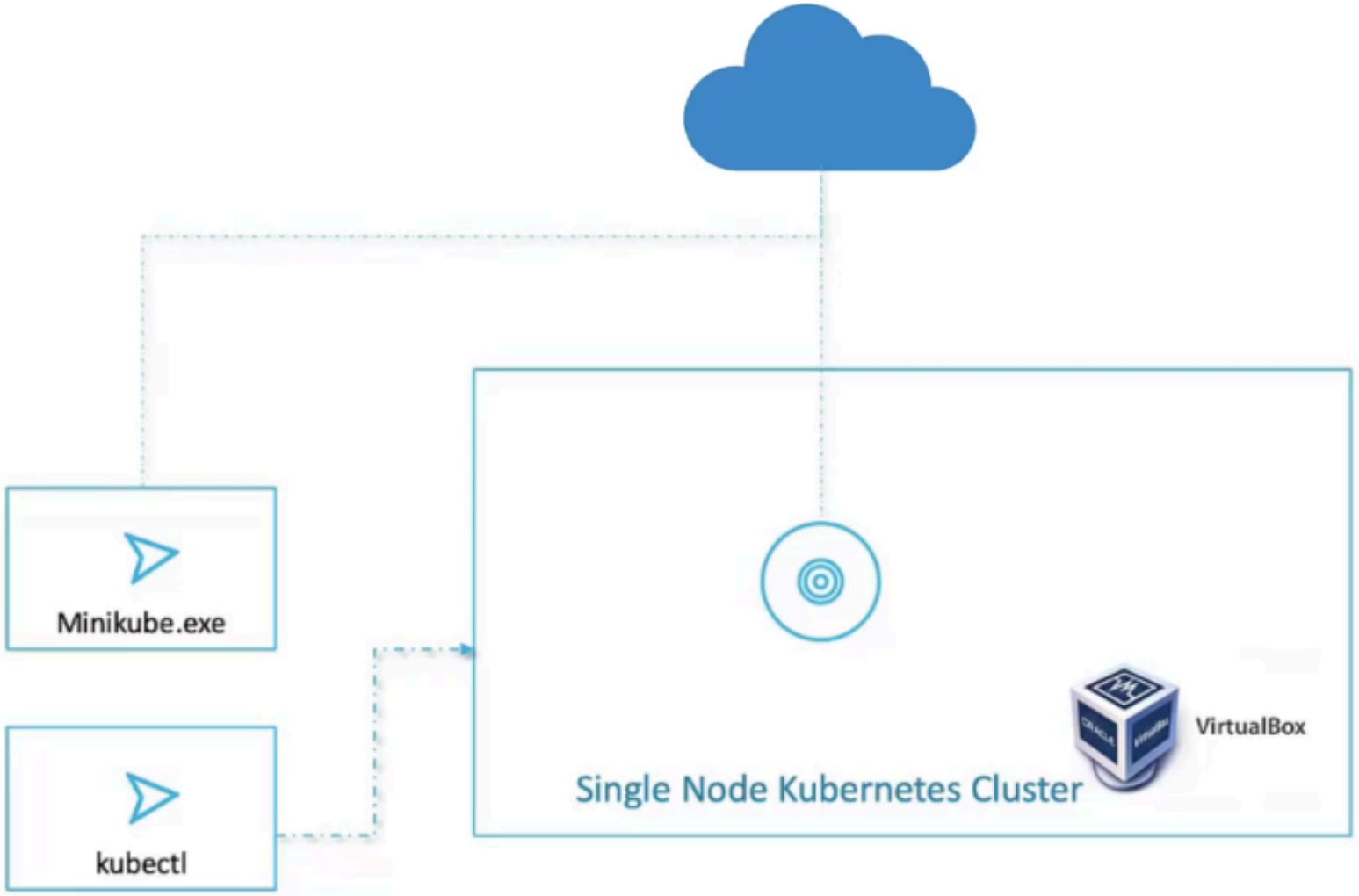
Minikube


kube-apiserver
etcd
node-controller
replica-controller
Master
Minikube
kubelet
Container Runtime
Worker Node

Minikube


kubelet
kube-apiserver
etcd
node-controller
replica-controller
Container Runtime
Minikube

Minikube


Minikube.exe
kubectl
Single Node Kubernetes Cluster
VirtualBox

Minikube - I

- Minikube is a free, open-source tool for running a Kubernetes cluster locally
  - Runs on Linux, Mac, and Windows
  - Useful for proof of concept, development, and testing
- Minikube is installed on the classroom VMs
  - Execute the following command to start a Minikube cluster:

```
$ minikube start
```

- Be sure to stop the Minikube cluster when it is no longer needed
  - It takes up resources and will cause other programs to run slower

```
$ minikube stop //next start will begin where you left off
$ minikube delete –all //next start will be totally clean
```

Minikube - II

- To get the Minikube pod status across all namespaces, execute:

```
$ kubectl get pods -A
```

- Once all the Minikube services are running, we can prove that it works
  - Execute the following commands to run a standard hello world example:

```
$ kubectl create deployment hello-minikube \
          --image=k8s.gcr.io/echoserver:1.4
$ kubectl expose deployment hello-minikube \
          --type=NodePort --port=8080
$ kubectl get services hello-minikube
$ minikube service hello-minikube
$ minikube delete --all
```