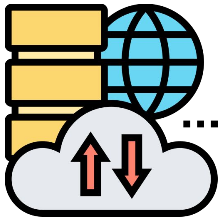
COEN 241
Introduction to Cloud Computing

Lecture 8 - Kubernetes



Lecture 7 Recap

- Kata Container
- Orchestration in Cloud
- Infrastructure as Code
- CoreOS

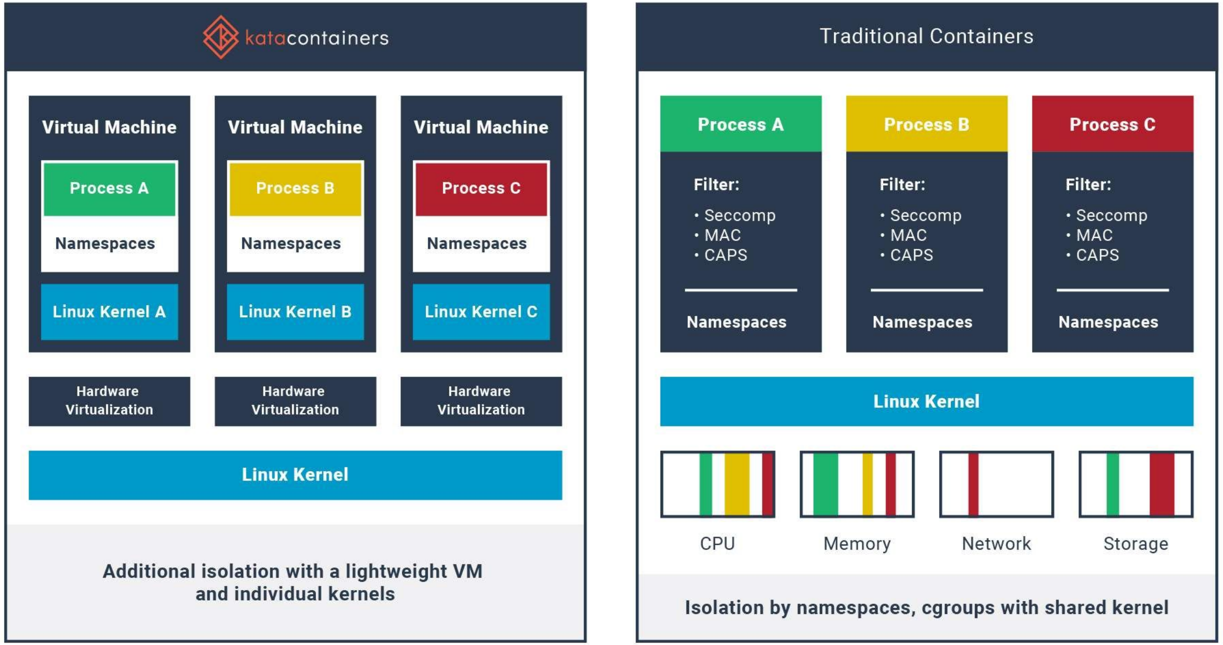
What is Kata Container?

- Released in 2017
  - From the merge of Intel's Clear Containers and Hyper's runV
  - "Wraps" containers into dedicated virtual machines
  - OCI runtime implementation can be plugged into the container engine
    - Docker
  - Can consume existing container images

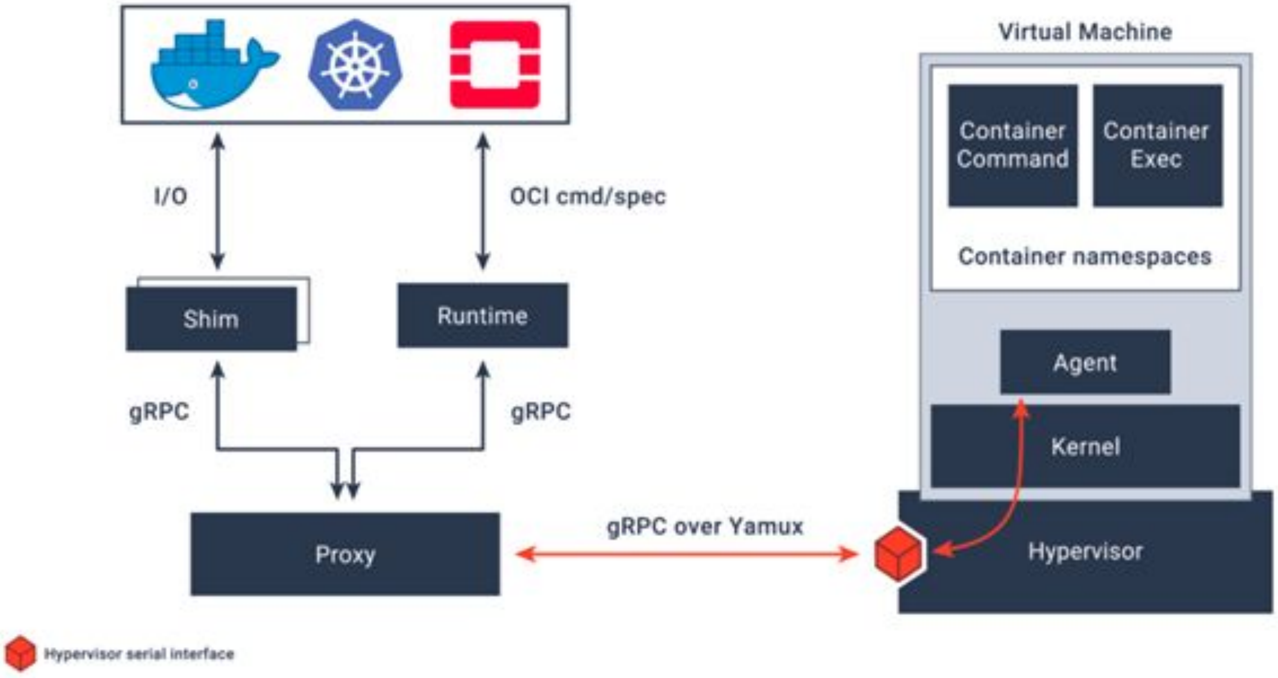
- Kata is a container runtime
  - Can still be coupled with other Docker platforms

https://katacontainers.io/

Kata Architecture



Kata Workflow



Orchestration

- Definition: Automatic management of computer systems
  - Deployment and configuration
  - Interconnection and coordination
  - Monitoring
    - Can also configure to hook up monitoring for management
    - E.g., Deploy more resources as load increases

- Growing set of very good (open-source) solutions:
  - **Machine focused**: e.g., Puppet, Terraform, Ansible, Salt, Chef…
  - **Cloud-based:** e.g., AWS CloudFormation, Terraform
  - **Container clusters:** e.g., Kubernetes

Automation vs Orchestration

- **Automation:** completing a **single task** or function without human intervention

- **Orchestration:** Managing a large-scale virtual environment or network by orchestrating the scheduling and integration of automated tasks between complex distributed systems and services
  - Simplifies interconnected workloads, repeatable processes, and operations.

- To simplify, **Automation** refers to a single task vs **Orchestration** arranges multiple tasks to optimize a workflow

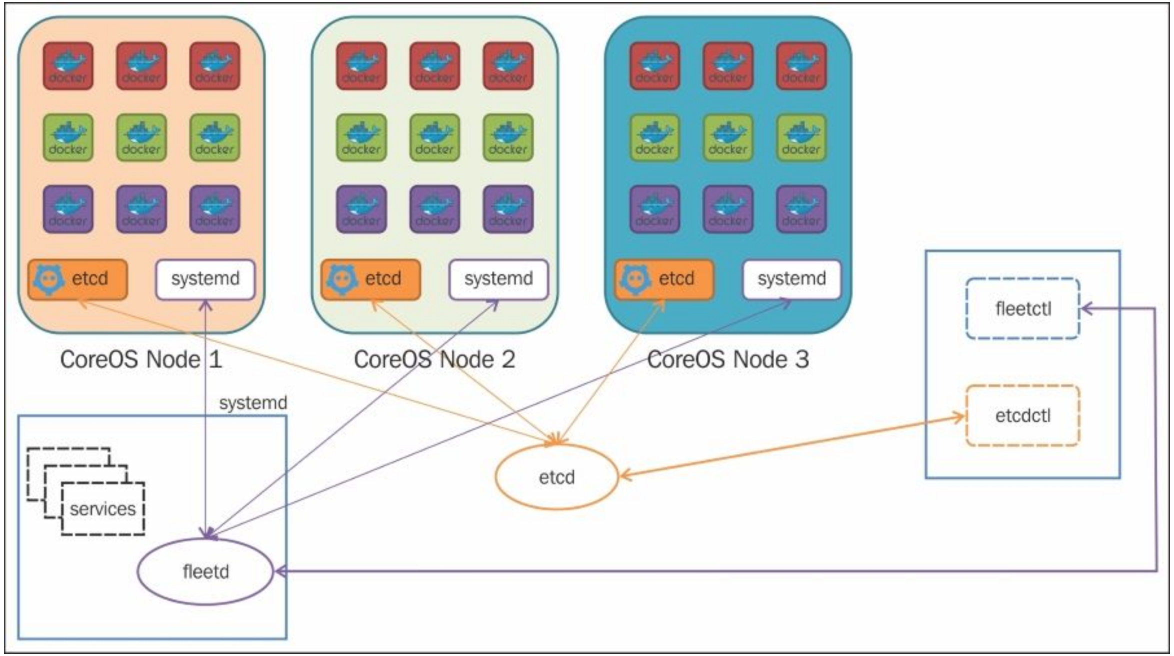
Declarative Configuration Management

- Declarative tools specify the desired target state
  - E.g., Can I have a coffee on my desk at 9AM on Monday morning?
- The means to reach target state is up to the configurations
  - Can take corrective action to react to drift in machine’s state
- State specification will be a domain specific language (DSL)
- Some example FOSS systems with large user communities
  - Puppet, Terraform, SaltStack

Imperative Configuration Management

- Also called ‘procedural’, i.e., specifying steps to run:
  - Usually written in chunks of code in configuration system authors’ favorite PL
- Some example systems:
  - Ansible (Py), Chef (Ruby), Saltstack
- Can write imperative code to have declarative effect

Core OS Architecture



Agenda for Today

- Kubernetes
  - Concepts
  - Architecture
- Minikube
- Readings
  - Recommended: None
  - Optional:
    - https://kubernetes.io/docs/tutorials/kubernetes-basics/
    - https://labs.play-with-k8s.com/
    - https://minikube.sigs.k8s.io/docs/handbook/
    - https://circleci.com/blog/what-is-yaml-a-beginner-s-guide/

Kubernetes Overview

What is Kubernetes?

- An open-source container-orchestration system for automating computer application deployment, scaling, and management.
- It is a Greek word meaning helmsman or pilot.
- Often called K8s, which is derived by replacing the eight letters of “ubernete” with the digit 8.
- Project emerged from Google in 2014; v1.0 released 2015
  - Related to Borg—Google’s (secret) internal container scheduler
  - K8s is implemented in Go, in contrast to Borg’s C++

Why is Kubernetes Useful?

- Provides an easy way to scale your containers
- Avoids cloud vendor lock-in
- Allows easy multi-cloud adoption
- Better management of application
- Works with many different container technologies
- It is a BIG piece of software with lot of moving parts
  - We will give a short, high-level overview of it
  - There are numerous tutorials and classes dedicated to Kubernetes

Interacting with Kubernetes

- kubectl
- APIs

Some Kubernetes Terms

- **Pods:** tightly coupled **set of containers**; smallest unit of scheduling
- **Node:** A Pod always runs on a Node. A Node is **a worker machine** in Kubernetes and may be either **a virtual or a physical machine**.
- **Services:** An abstraction which defines a logical set of Pods and a policy by which to access them
  - Sometimes this pattern is called a micro-service architecture
- **Volumes:** Persistent storage that can be shared between containers
- **Namespaces:** Multiple virtual cluster in a same physical cluster

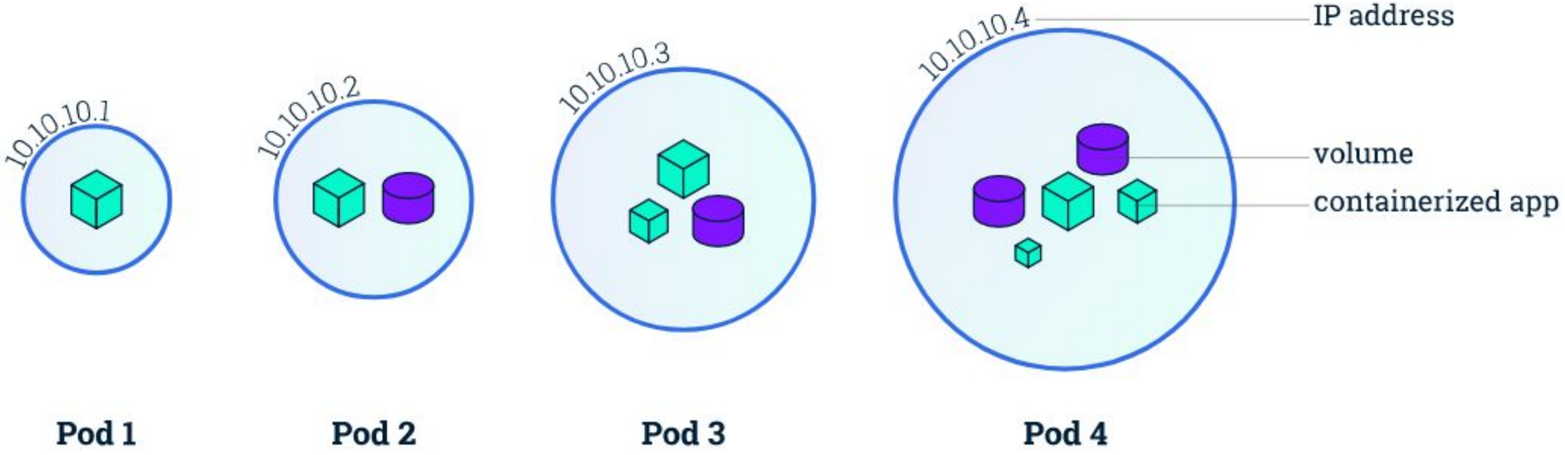
Kubernetes Pods

- Pods are the smallest deployable units of computing that you can create and manage in Kubernetes
- Group of one or more containers with **shared storage** and **network resources**, and a specification for how to run the containers
  - A group of Docker containers with shared namespaces and file system volumes
- A Pod's contents are always co-located and co-scheduled, and run in a shared context

Kubernetes Pods

- Pods are assigned an IP address, for networking
  - All containers within the pod share that address and its ports
- Pods provide app. storage (volumes) to containers
- Pods usually created by controllers, and not directly
- List and show detailed information of the Pods by running
  - `kubectl get pods`
  - `kubectl describe pods`

Kubernetes Pod Overview

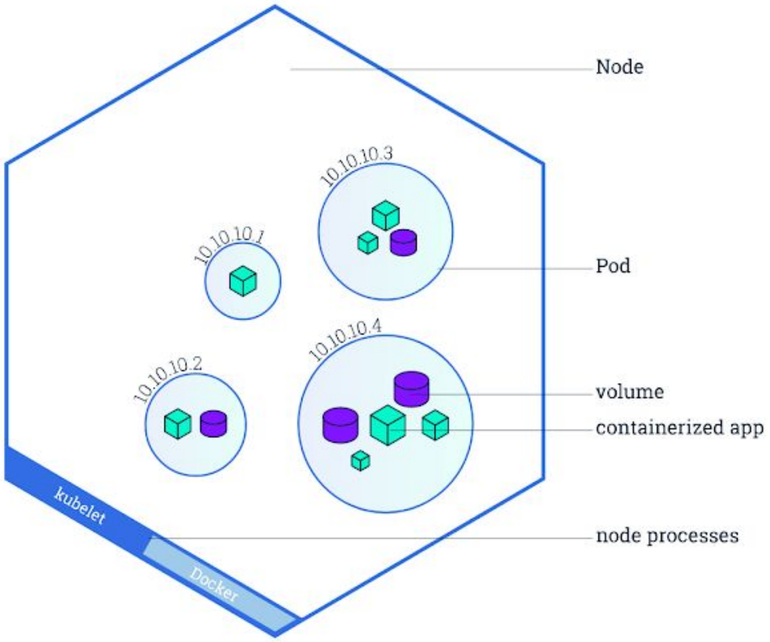


Example Pod Definition (PodSpec)

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-pod
  label: myApp
spec:
  volumes:
  - name: html
    emptyDir: {}
  containers:
  - name: 1st
    image: nginx
    volumeMounts:
    - name: html
      mountPath: /usr/share/nginx/html
  - name: 2nd
    image: debian
    volumeMounts:
    - name: html
      mountPath: /html
    command: ["/bin/sh", "-c"]
    args:
      - while true; do
          date >> /html/index.html;
          sleep 1;
        Done
```

Kubernetes Nodes

- A set of worker machines that run containerized applications
- Every cluster has at least one worker node
- When you deploy K8s, you get a **cluster**
- `kubectl get nodes`



Kubernetes Services

- A **logical abstraction** for a deployed group of pods in a cluster
  - which all perform the same function
- Kubernetes way of configuring a proxy to forward traffic to a set of pods
- Since pods are ephemeral, a service enables a group of pods to be assigned a name and unique IP address (clusterIP)
- As long as the service is running on that IP address, it will not change.
- Services also define policies for the access to Pods

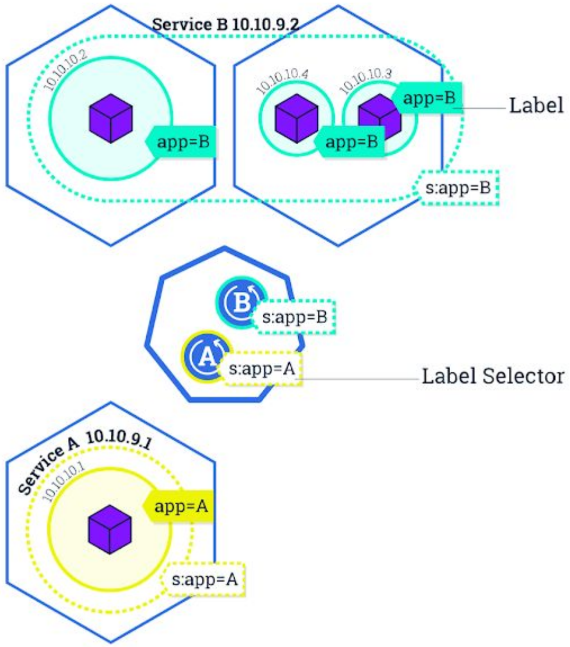
Types of Kubernetes Services

- **ClusterIP** (default): Exposes a service which is only accessible from within the cluster.
- **NodePort**: Exposes a service via a static port on each node's IP.
- **LoadBalancer**: Exposes the service via the cloud provider's load balancer.
- **ExternalName**: Maps a service to a predefined externalName field by returning a value for the CNAME record.

Example Service Definition

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  type: ClusterIP
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

Service Overview



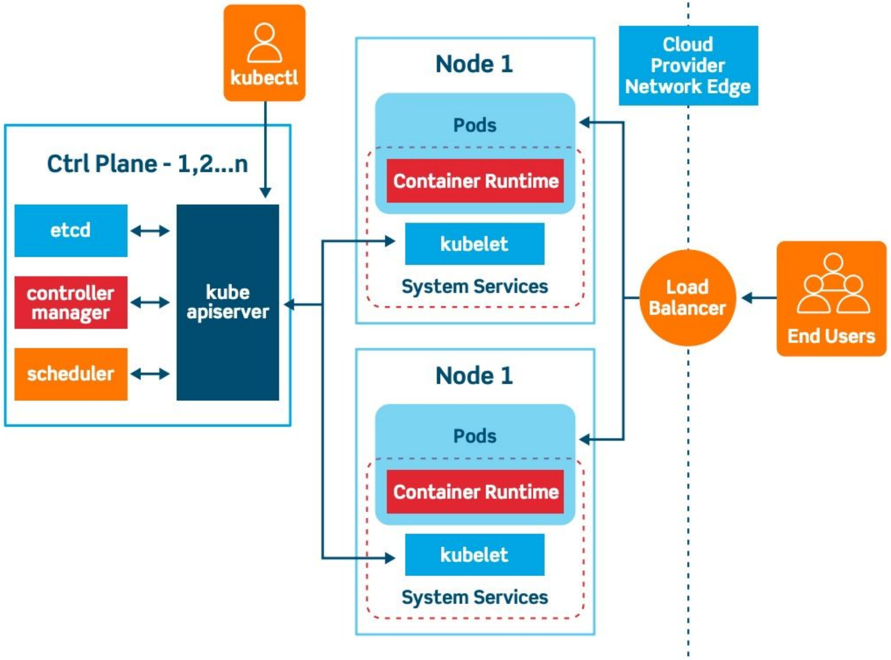
Stateless vs Stateful Applications

- Stateless applications scale easily: just start more pods
    - E.g., web servers presenting read-only workloads
- Stateful apps are more difficult, e.g.:
    - Databases having primary and secondary instances
    - Machine Learning Applications
- Kubernetes StatefulSet controllers enables stateful applications
    - Provides durable storage
    - Persistent ID for each Pod
    - https://kubernetes.io/docs/tutorials/stateful-application/basic-stateful-set/

Kubernetes Architecture

Kubernetes Architecture Overview

- Kubernetes cluster consists of:
- Master Node(s) (control plane)
    - With a distributed storage (etcd)
- A number of cluster nodes

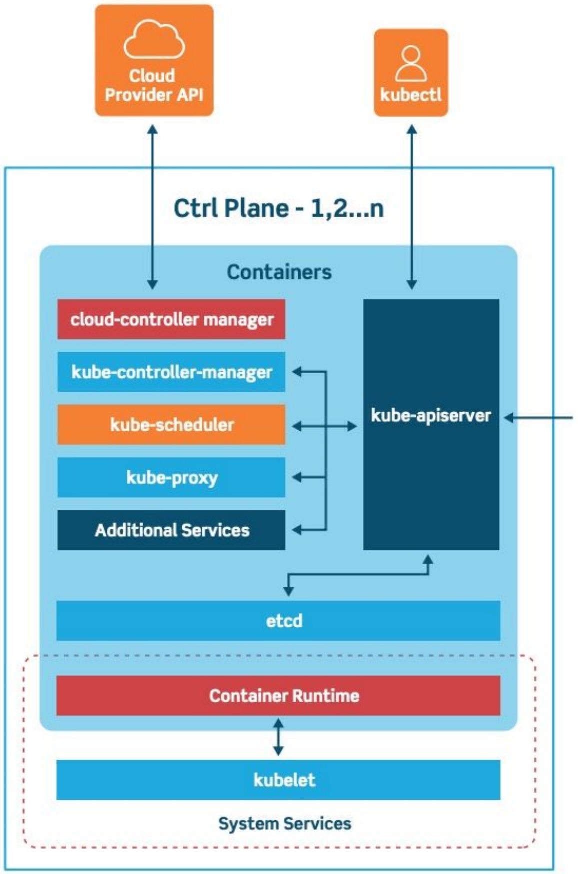


Kubernetes Master Node

- A logically centralized control point
- Maintains a record of all Kubernetes objects
  - Manages object states, responding to changes in the cluster
  - Works to make the actual state of system objects match the desired state
- Consists of the following components
  - **API server**: Allows Kubernetes cluster to be controlled from a user
  - **Controller manager**: Checks replication, status of nodes and state of cluster
  - **Scheduler**: Allocates pods waiting to run nodes
  - **etcd**: Consistent repository of configuration information

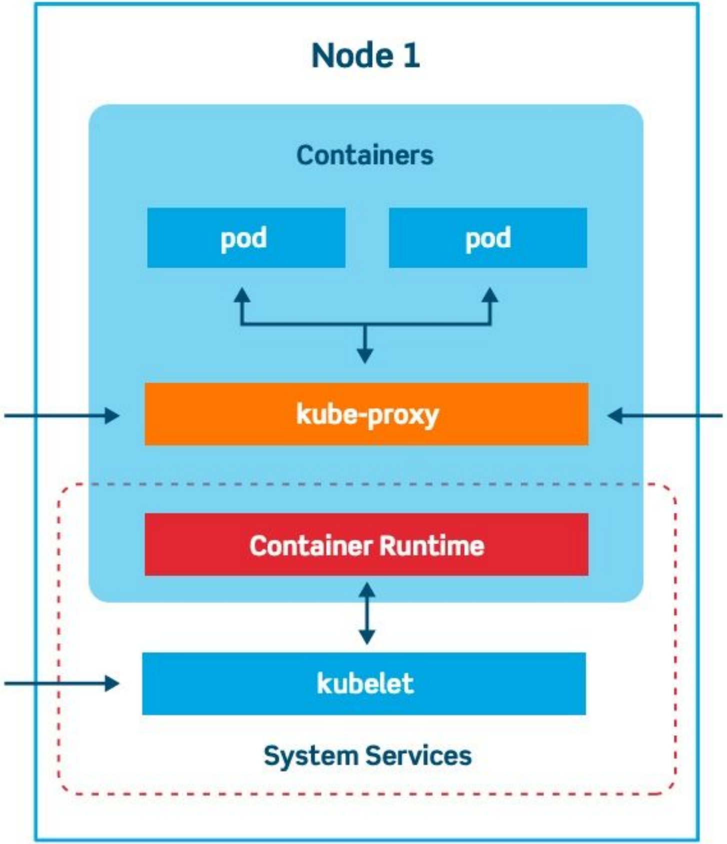
Kubernetes Master Node

- Can be one or more master nodes
- Can accept APIs from cloud providers
- Also provides proxy for network access
- **etcd** can be configured for multi-node distributed configuration manager



Kubernetes Cluster Node

- Nodes can run pods but also,
  - kube-proxy: provides network services; leveraging OS facilities
  - **kubelet**: checks on health of containers within a pod
  - cAdvisor: provides statistics about container resource use
- **Kubelet** is the primary and most important controller
  - Drives the container execution layer



Kubelet

- The kubelet is the primary "node agent" that runs on each node
- It can register the node with the kube apiserver using one of:
  - The hostname
  - A flag to override the hostname
  - Specific logic for a cloud provider
- The kubelet ingests a PodSpec and runs each pod

Extension/Alternatives to Kubernetes

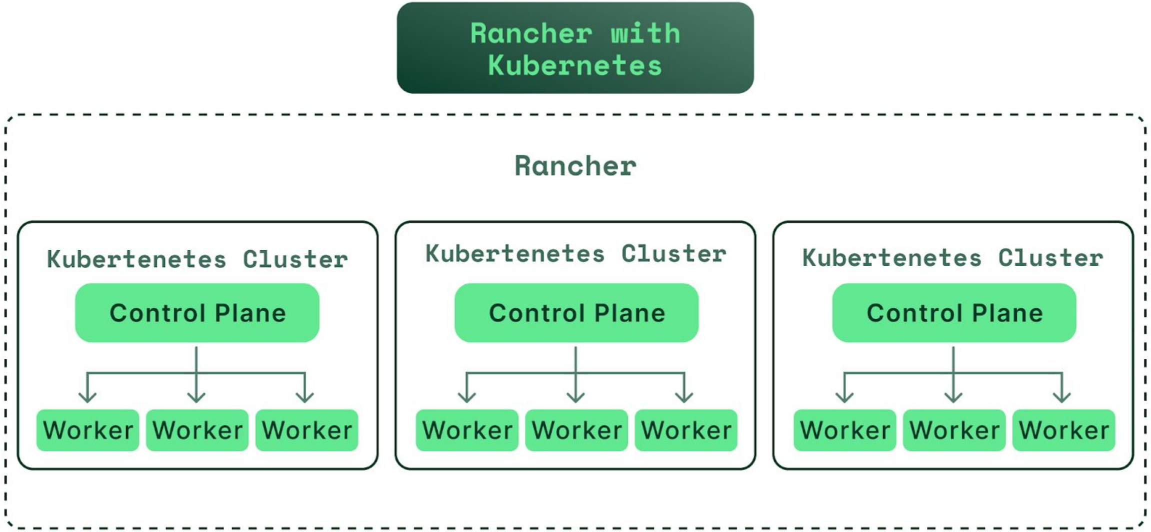
K8s as a Service

- K8s can manage your containers, but how to set it up given a IaaS?
  - Need **VMs** running master and the nodes
- Cloud providers can provide a range of proprietary options
  - AWS
    - AWS Fargate provides a complete container service
    - AWS EKS provides control plane; you set up K8s nodes on EC2
    - Use EC2 to deploy all the components if you want full control
  - Azure and GCP has a similar services

K8s in Multi-Cloud Application

- Quite difficult!
  - Very Manual
  - Or you can use a different service
- Tools like Rancher provide for **multi-K8s** cluster management
  - Can even import existing K8s cluster!
  - https://rancher.com/
  - https://www.youtube.com/watch?v=LX0zVh-AYm4
- Works because cloud providers’ container services are very similar!
  - Benefit of learning the basics!

Rancher versus Kubernetes



Rancher versus Kubernetes

- Kubernetes Features
  - Kubernetes based platform is easily migratable across cloud providers
  - Easier to scale as compared to traditional applications hosted in virtual machines (VM)
  - Easy to control cluster density and autoscaling
  - In node failures, pods are automatically rescheduled to other nodes
  - Can be hard to manage multiple clusters
- Rancher Features
  - Easy visualization to manage multiple clusters
  - Can easily create new clusters or add existing ones to it
  - Add a concept of projects for better grouping of namespaces
  - Permissions can be configured per project across clusters

Terraform versus Rancher, Kubernetes, etc.

- Terraform is a stack below tools like Rancher, it:
  - Allows previous of the plans using a form of IAC
  - Can easily provision at level of particular instance type
  - Can work with Rancher and Kubernetes
- Rancher can also deploying Kubernetes to bare metal servers
  - Including monitoring & security management tools
- If you need specific type of infrastructure nodes it can get complicated
  - E.g., configuring a GPU node on Amazon for deep learning
  - Containers can use this hardware, but what about container managers

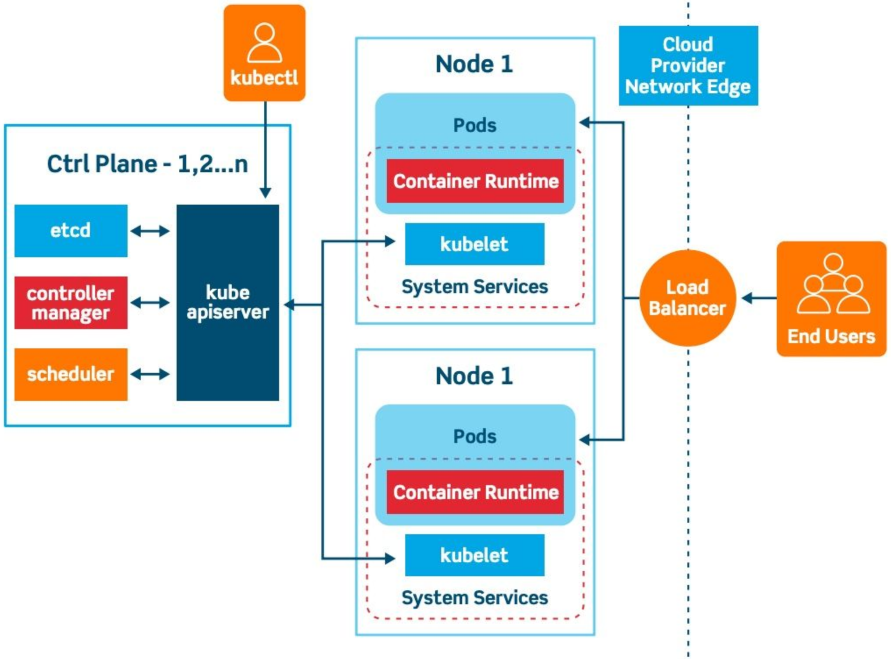
Why it's so hard to pick a "winner"?

- All are changing rapidly!
  - CoreOS Linux's original fleet functionality replaced by K8s
  - Rancher's original functionality are being replaced by Docker Swarm
- Then what to use? Consider your and **the developers' time**
  - Look to see whether a new tool can optimise your processes
  - when taking into account the cost of transition
- Must have **IAC** and continuous integration pipelines
  - IAC is a must for quickly adapting it!

Minikube

Production Kubernetes Cluster

- Requires
  - Multiple Master Nodes
  - Multiple Worker Nodes
- Hard to try Kubernetes locally!
- Minikube to the rescue



Minikube: A local Kubernetes setup

- https://minikube.sigs.k8s.io/docs/start/
- Running both master and worker processes **on the same node**
- Minimum requirement: 2 CPUs, 2GB of RAM, 20GB of free disk space
- Use **kubectl** to interact with the Minikube cluster
- **Kubectl:** A CLI for Kubenetes cluster to talk to the master
    - Not only Minikube, but any type of Kubernete cluster
    - https://kubernetes.io/docs/reference/kubectl/overview/

TODOs!

- Kubernetes Quiz!
  - https://kubernetes.io/docs/tutorials/kubernetes-basics/
  - Up to Module 4 Only!
- HW 2 will be out next class
- Study for Midterms!

Agenda for Today

- Kubernetes
  - Concepts
  - Architecture
- Minikube
- Readings
  - Recommended: None
  - Optional:
    - https://kubernetes.io/docs/tutorials/kubernetes-basics/
    - https://labs.play-with-k8s.com/
    - https://minikube.sigs.k8s.io/docs/handbook/
    - https://circleci.com/blog/what-is-yaml-a-beginner-s-guide/

Questions?