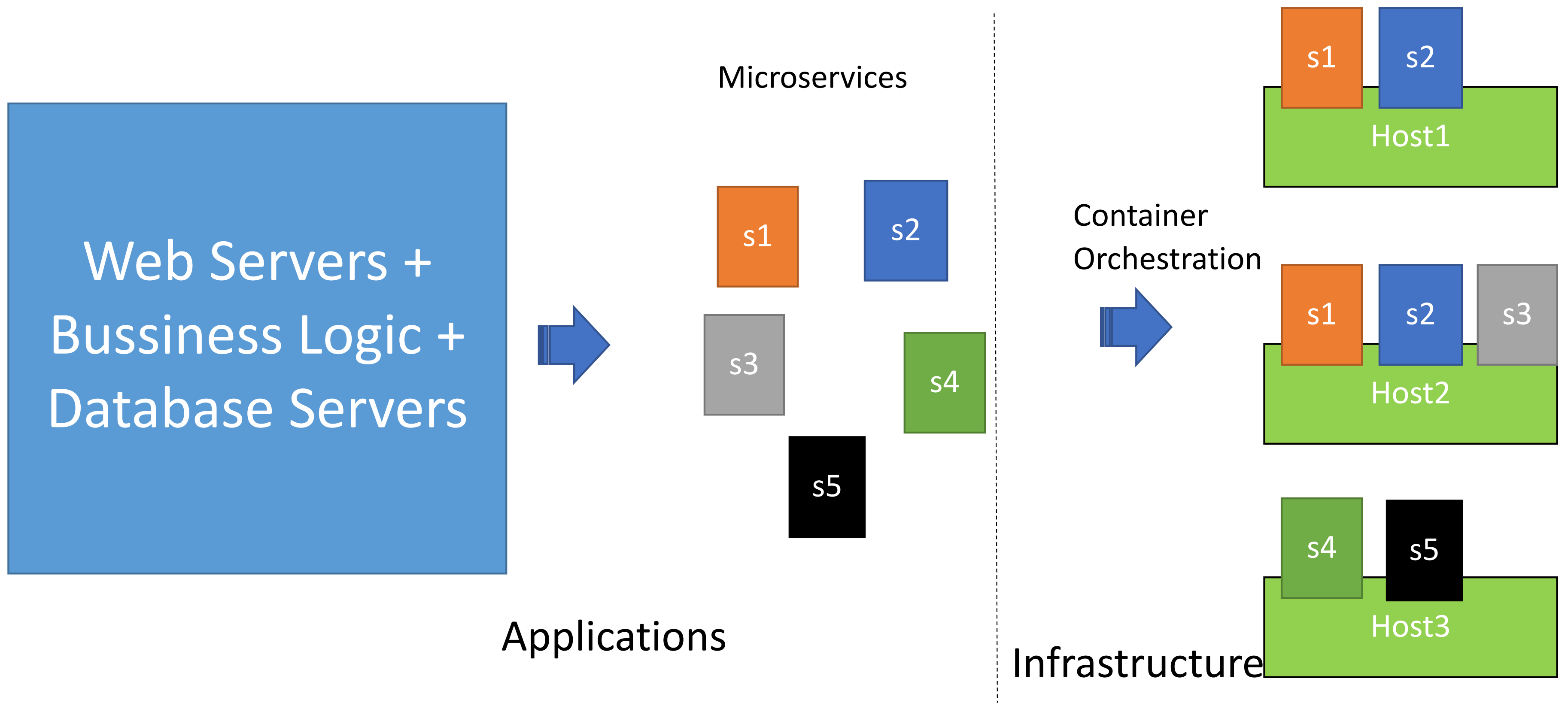
CS-552/452 Introduction to Cloud Computing

Orchestration (Part I)

Overview of Kubernetes

Micro-services Deployment and Management Challenge



Orchestration

- Common in deploying and managing a large-scale, distributed application in a distributed environment.
- Developer/Administrator gets an interface for specifying the application deployment setup.
- Orchestration platform automatically handles the actual deployment and runtime management.
- The configuration setup is very application specific such as
    - number of services
    - how they communicate
    - failover plans, etc.

Orchestration Tools/Platforms

- Kubernetes:
  - container orchestration tool maintained by the cloud-native foundation
- Openshift:
  - built on top of Kubernetes by RedHat
- Nomad:
  - supports container and non-container workloads.
- Docker Swarm:
  - one component in Docker ecosystem
- Mesos:
  - cluster management tool created by Twitter first and later open sourced
- Google Container Engine (GKE), WS Elastic Kubernetes Service (EKS), and Azure AKS Service
  - At their core is Kubernetes

kubernetes

What is Kubernetes?

- Open source container orchestration tool developed by Google and donated to the cloud-native foundation

- A platform for hosting containers in a clustered environment with multiple hosts (or nodes)

- Helps manage containerized applications that are made up of hundreds, if not thousands, components (microservices)
  - On a cluster of VMs, physical machines, or hybrid

What problems does Kubernetes solve?

- Driven by the trend from monolithic to microservices
- Plus, the usage of container technology for hosting small, independent services
- However, managing hundreds of containerized microservices across multiple compute nodes (VMs or physical machines) is challenging
- Need for a uniform way of managing those hundreds of containers

What does Kubernetes offer?

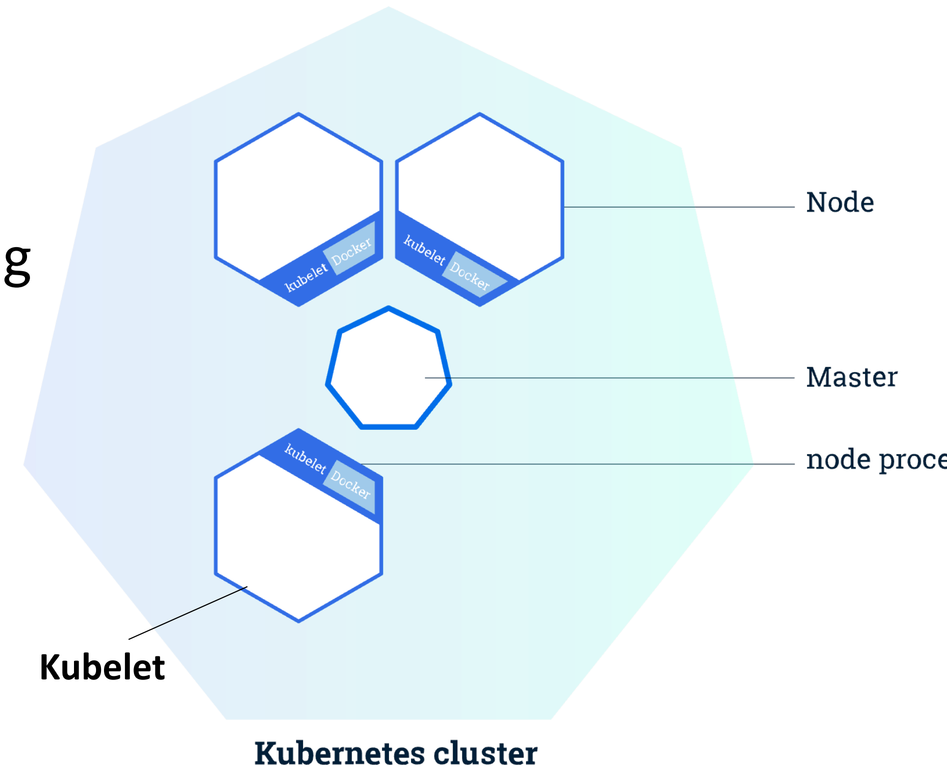
- High availability or no downtime
  - The “5 9s” rule – 99.999% percent of time service is online (i.e., 6 minutes down each year)
  - In practice, most major companies have stopped advertising 5-9s due to the difficulty of guaranteeing it.
- Scalability or high performance
  - Adjust to varying loads
- Disaster recovery – backup and restore

Architecture of Kubernetes: Clusters

- Coordinate a cluster of computers (nodes) – VMs or physical machines
- Abstract such a cluster of computers as a single unit.
- Allow users to deploy containerized applications to the cluster without tying them to individual machines.
- Automate the distribution and scheduling of application containers across a cluster in a more efficient way.

Architecture of Kubernetes: Nodes

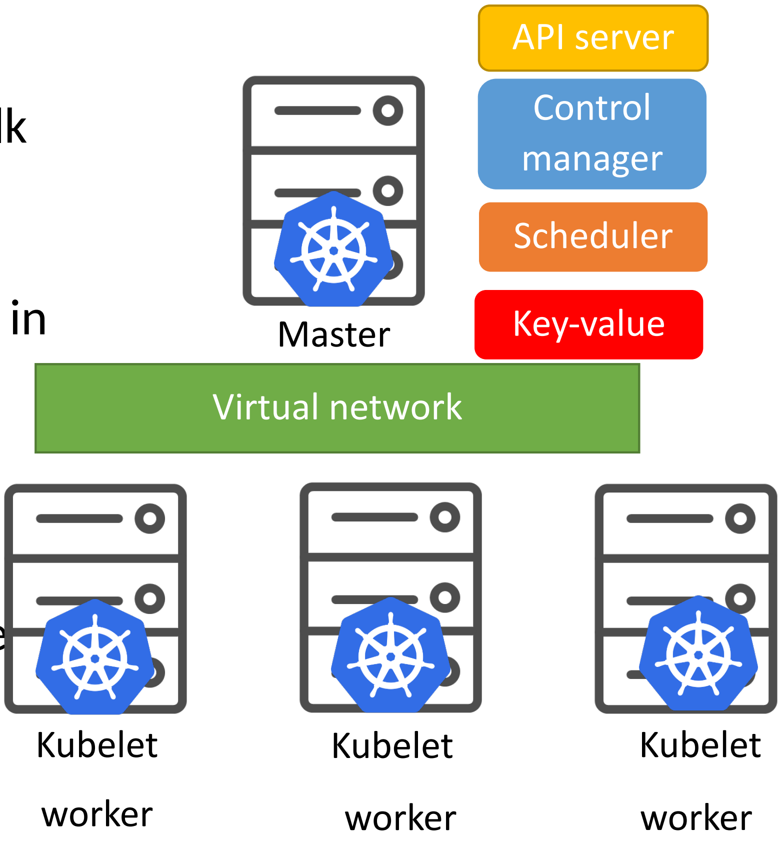
- A Kubernetes cluster consists of two types of nodes:
  - The Master coordinates the cluster
    - Make global decisions and respond to cluster events
    - Scheduling, maintaining, scaling, rolling out new updates
  - Nodes are the workers that run containerized applications
    - Runs an agent in the worker nodes
    - Carry out tasks guided by the master
    - Rely on container tools (e.g., Docker) for container management



Kubernetes Master

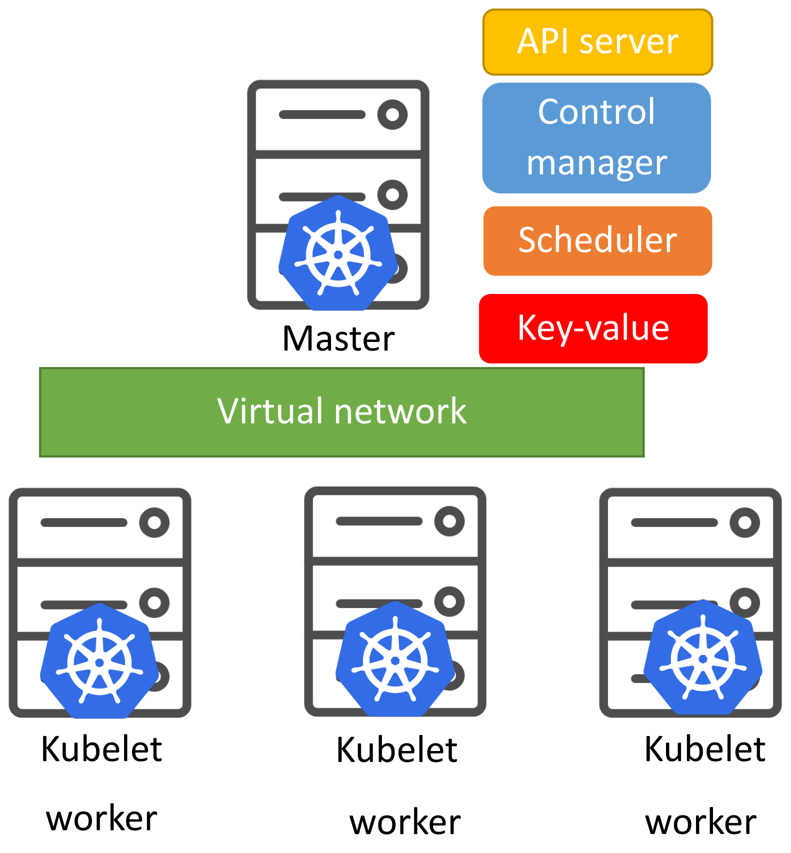
- At least one master, serving as the control plane

- Run multiple processes for the management of the cluster
  - One API server: the entry point (for platform users) to talk to the Kubernetes clusters (e.g., UI or CMD)
  - One controller manager: Keep track of what's happening in the cluster and maintain the desired state for each container instance
    - e.g., a container died and needs to be restarted
  - One scheduler: for scheduling containers across available (worker) nodes based on workload and available server resources
  - One key-value store: to store the state of the cluster.

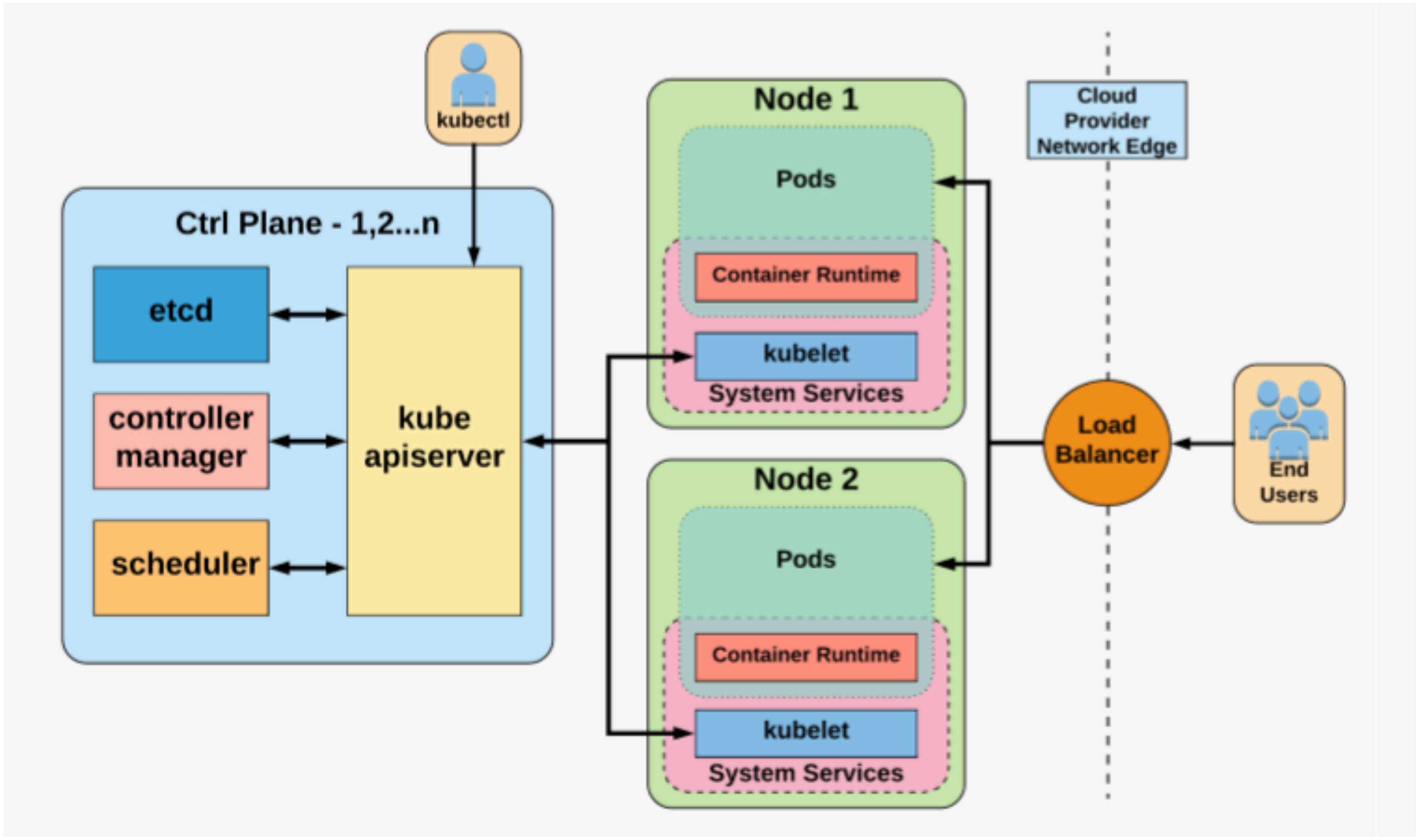


Kubernetes Nodes (Worker)

- Runs the actual workloads (in containers)
- Consumes more resources than master
- Should be bigger and faster
- It’s okay to lose some of these worker nodes – services of those nodes can be recovered by the master node
- But what if the master node fails?
  - Have at least a backup of the master node
  - Or run multiple master nodes – in the High Availability (HA) cluster



Summary: Kubernetes Architecture



Main Kubernetes Objects

- Kubernetes objects are entities that describe
  - What containerized applications are running
  - The resources available to those applications
  - The policies around how those applications behave, such as restart policies, upgrades, and fault-tolerance
- Creation, modification, and deletion of Kubernetes objects are via Kubernetes API

Pod

Service

Volumes

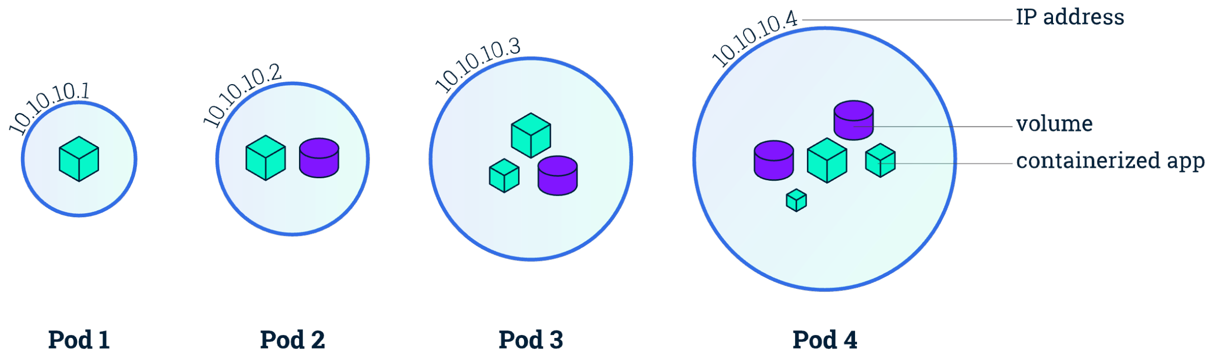
Deployment

Replicas

Main Kubernetes Objects : Pods

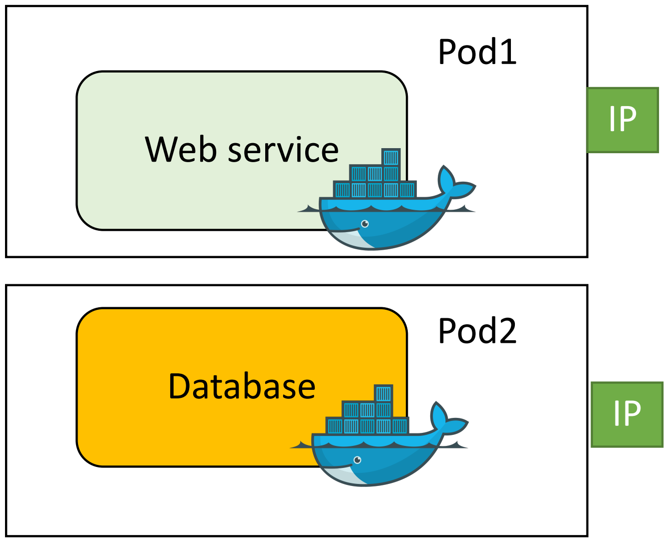
- A Pod is a Kubernetes abstraction over containers
  - Thus, later you can use other technologies for hosting your applications

- You can place one or more containers inside a Pod
  - As well as some other resources like network (IP addresses), storage (volumes), container images, etc.
  - Typically, one container per Pod
  - If you need to place more, usually, they are closely coupled



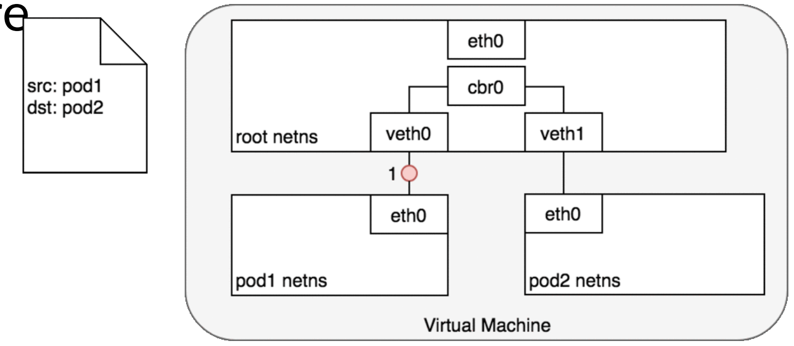
Main Kubernetes Objects : Pods

- A Pod represents an application-specific "logical host" and can contain different application containers which are relatively tightly coupled.



- Pods are the atomic unit on the Kubernetes platform
  - Smallest unit in Kubernetes
  - Usually, one application per Pod
  - If many, they are deployed and scheduled as a whole

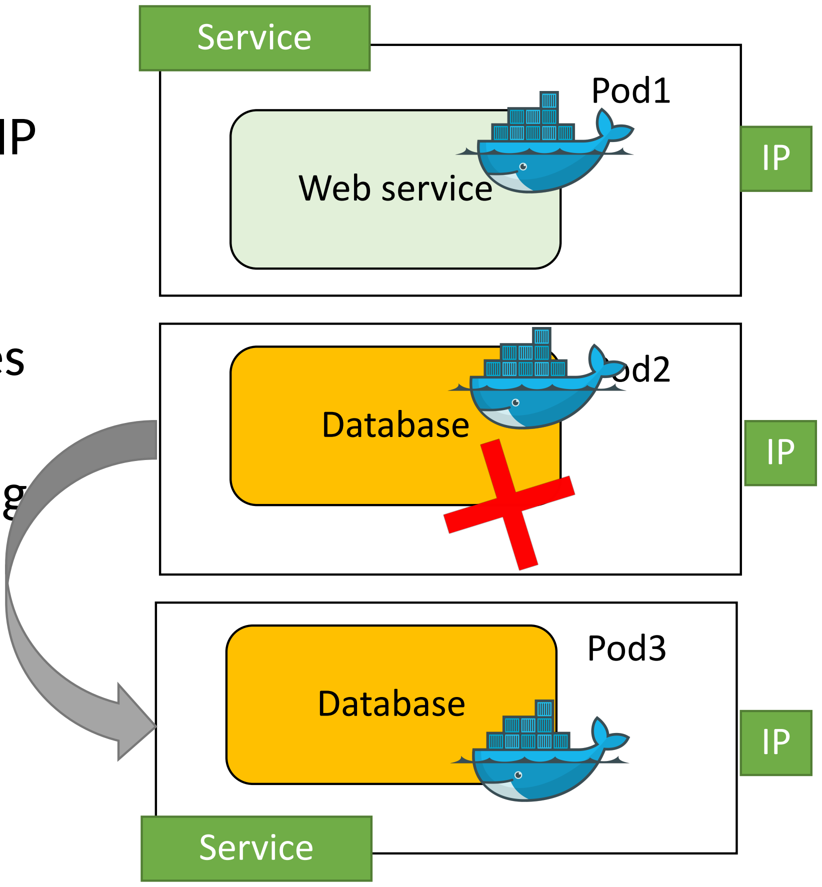
- To allow Pods to communicate, internal IP addresses are assigned
  - To pods, but not containers
  - All containers of the same Pod share the <u>same network namespace</u>)



Main Kubernetes Objects : Services

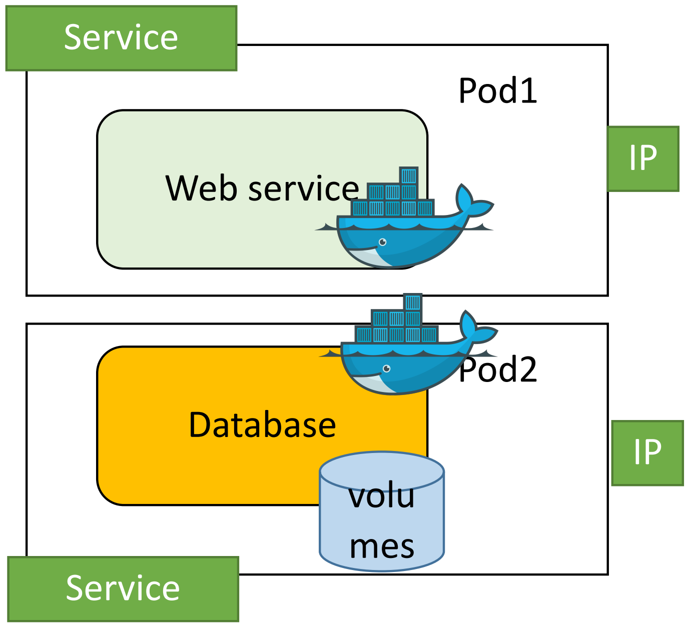
- Pods are ephemeral
  - Once a pod dies, a new pod will be created
  - The IP of the new pod will be changed
  - Problematic for services that need to have a fixed IP addresses

- A Service in Kubernetes is an abstraction which defines a set of Pods and a policy by which to access them
  - A static, permanent IP (public IP) used for accessing Pods running inside
  - Now you can expose your services publicly
  - You can call each a micro-service



Main Kubernetes Objects : Volumes

- Data stored in container will be lost once the container is shutdown
- To persist data, Kubernetes uses Volumes
  - Attaches a physical store to a pod
  - Could be local, remote, or cloud storage services



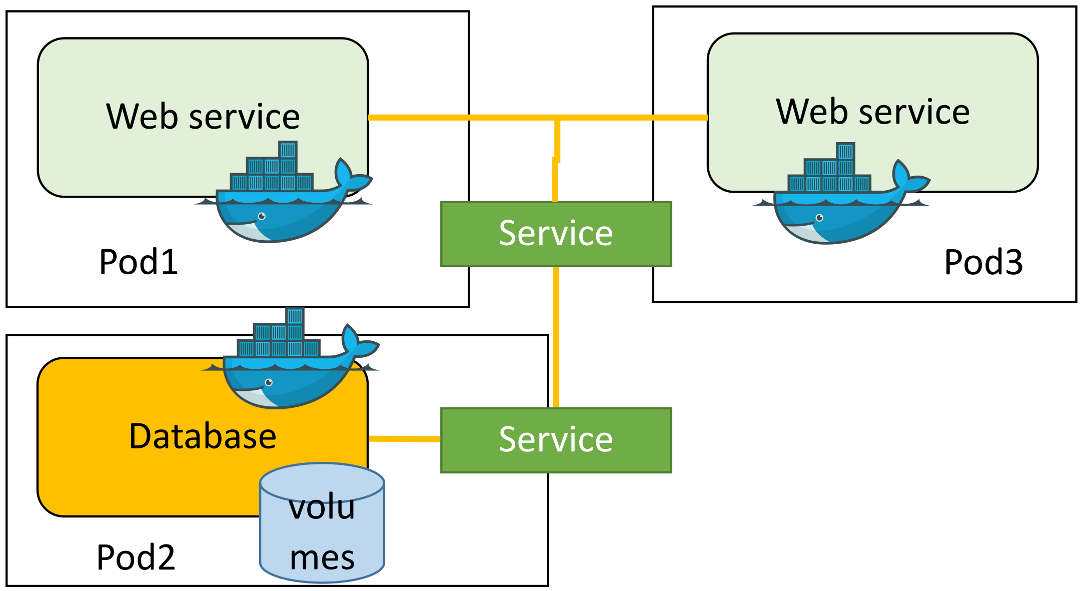
Main Kubernetes Objects : Replicas

- Users can specify multiple replicas of the same pod
  - For availability and performance

- Multiple instances of that pod will be deployed during runtime

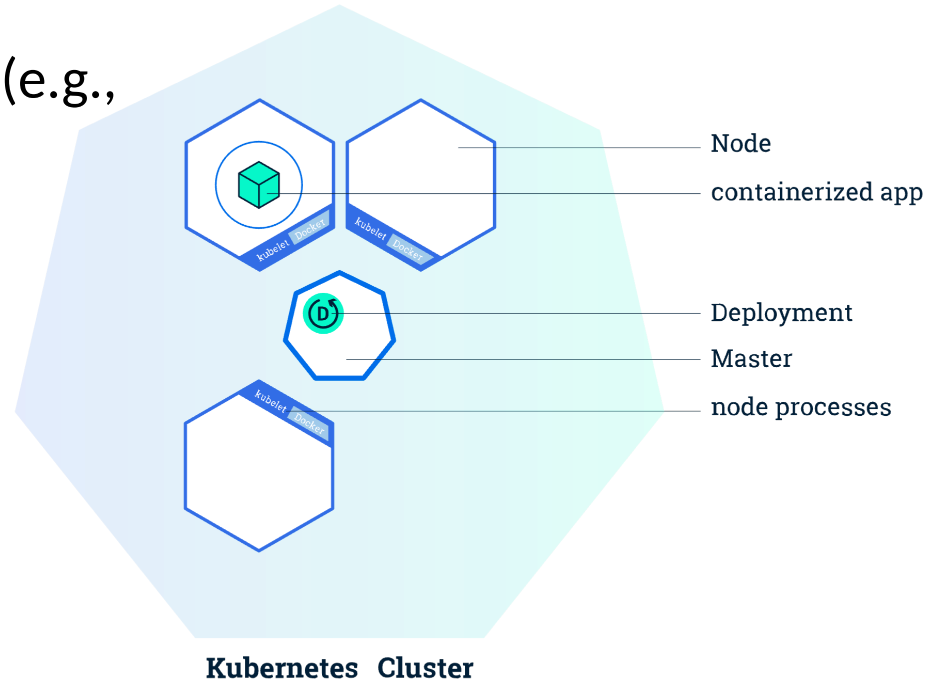
- A load balancer (part of the service) routes the incoming requests to a replica automatically

- How to specify these configurations?



Main Kubernetes Objects : Deployment

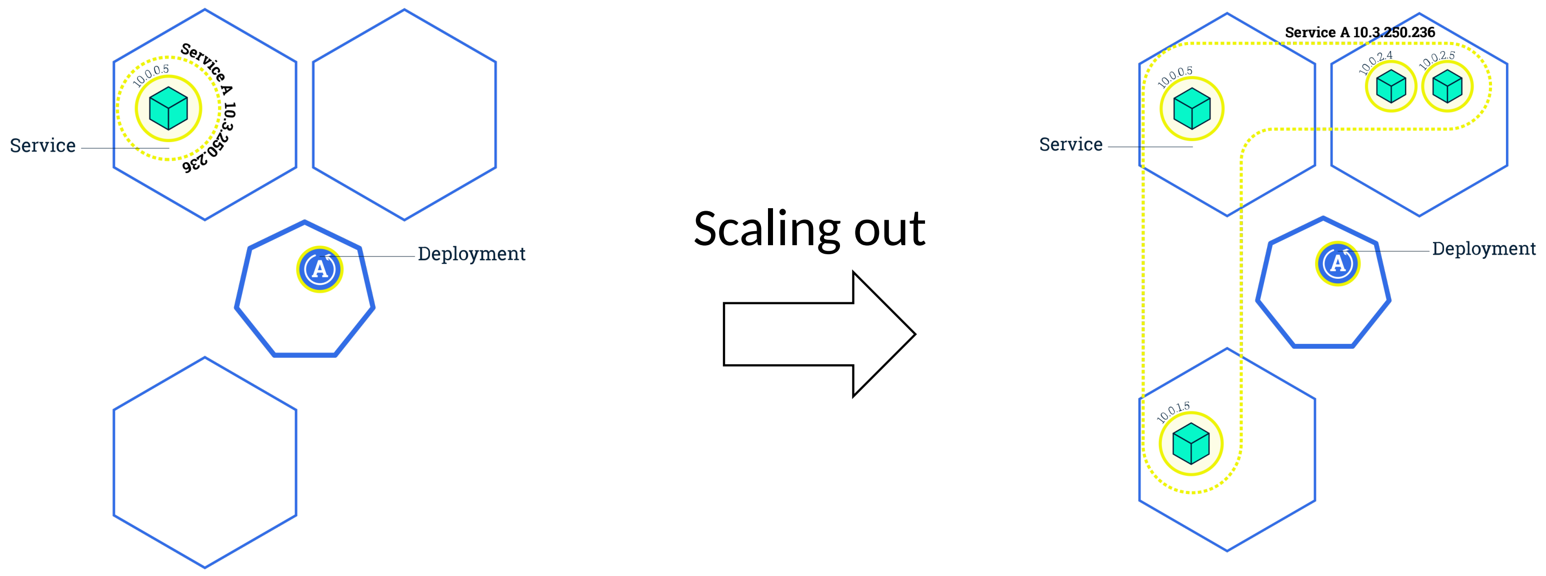
- You can create a Kubernetes Deployment object to specify a desired state of a pod
  - E.g., Number of replicas, volumes, resources (e.g., CPU and memory)
- You can deploy such a Pod by submitting the deployment object to the Kubernetes API server
- A Kubernetes Deployment Controller
  - Continuously monitors pods
  - Ensures that a specified number of pod replicas are running at any given time.



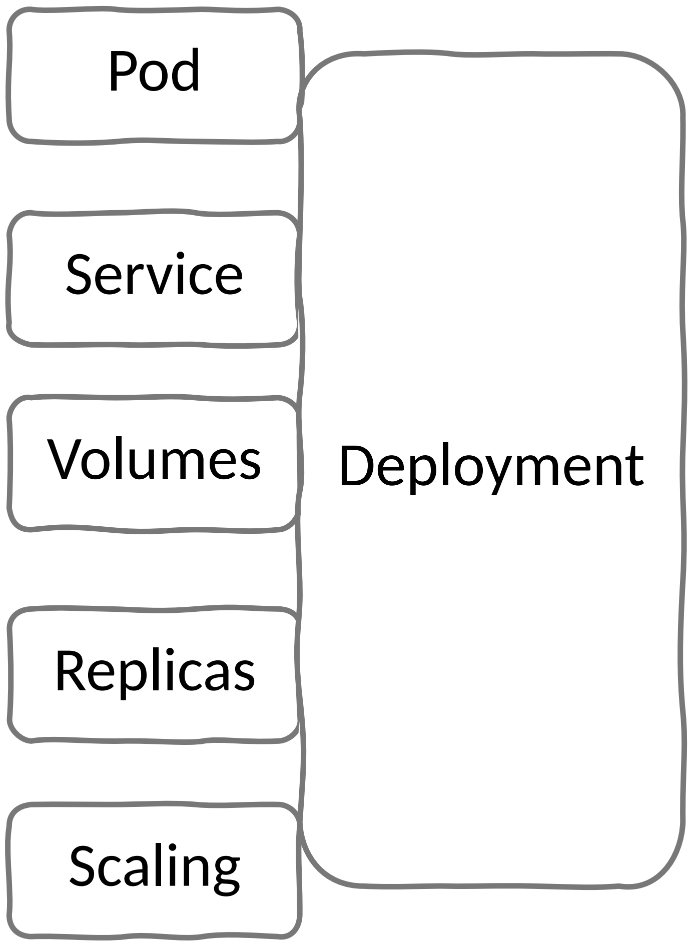
Main Kubernetes Objects : Scaling

- Scaling (out/in) is accomplished by changing the number of replicas in a Deployment

- The controller periodically adjusts the number of replicas to match the observed metrics such as average CPU utilization, average memory utilization or any other custom metric to the target specified by the user.



Summary

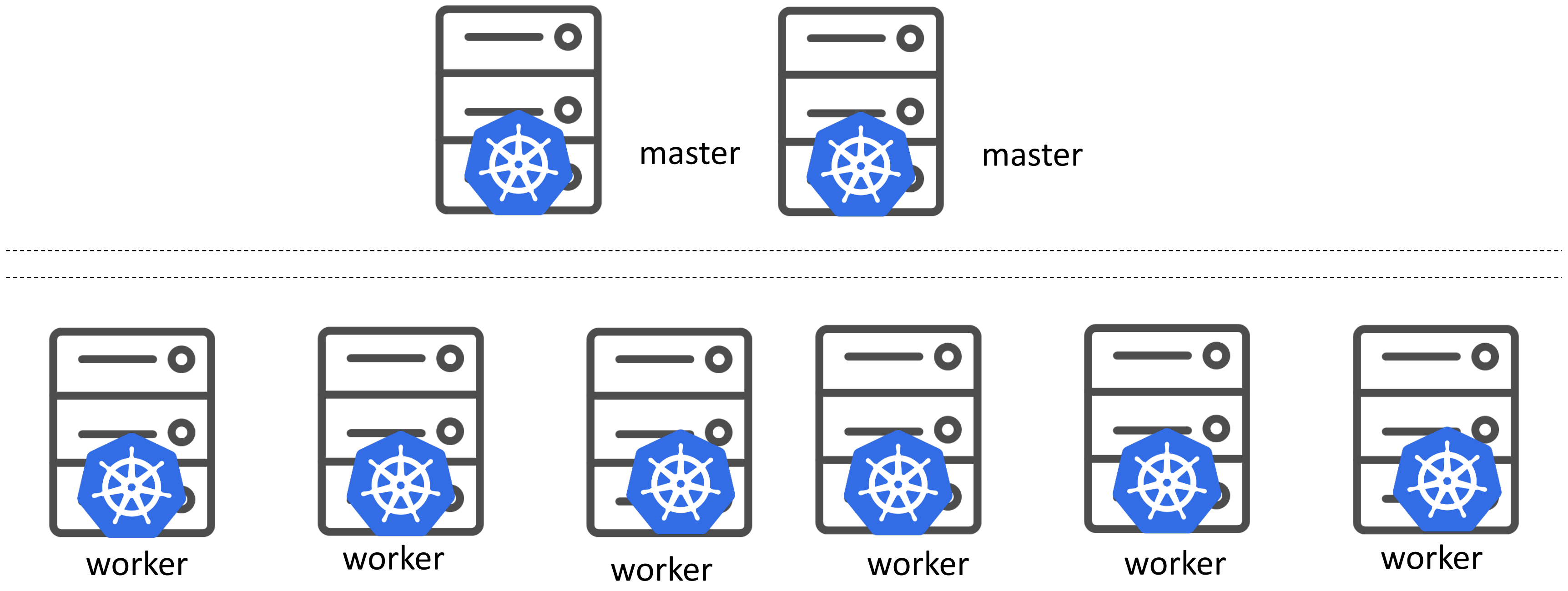


- Abstractions over containers
- Network
- Persisted storage
- Availability
- Scalability

[Kubernetes: https://kubernetes.io/docs/home/](https://kubernetes.io/docs/home/)

A production setup

- Multiple masters (at least two) for high availability
- Multiple worker nodes for actual workloads



A local/test setup -- Minikube

- https://minikube.sigs.k8s.io/docs/start/

- Running both master and worker processes on the same node with a minimum requirement:
  - 2 CPUs or more, 2GB of free memory, 20GB of free disk space

- Use kubectl to interact with the Minikube cluster
  - A command line tool for Kubernetes cluster
  - Talk to API Server of the master
  - Not only Minikube, but any type of Kubernetes clusters

Sources

- Kubernetes: https://kubernetes.io/docs/home/
- Kubernetes Basics: https://kubernetes.io/docs/tutorials/kubernetes-basics/
- Minikube Handbook: https://minikube.sigs.k8s.io/docs/handbook/
- A Guide to the Kubernetes Networking Model: https://sookocheff.com/post/kubernetes/understanding-kubernetes-networking-model/
- A multi-node playground: https://labs.play-with-k8s.com/
- What is YAML? A Beginner's Guide: https://circleci.com/blog/what-is-yaml-a-beginner-s-guide/