Kubernetes Introduction

DevOps Course

Agenda

- Introduction
- High Level Architecture
- Kubernetes Objects and Workloads
- Demo

What is Kubernetes?

- Is an open-source system for automating deployment, scaling, and management of containerized applications

- Kubernetes abstracts away the hardware infrastructure and exposes your whole datacenter as a single enormous computational resource.

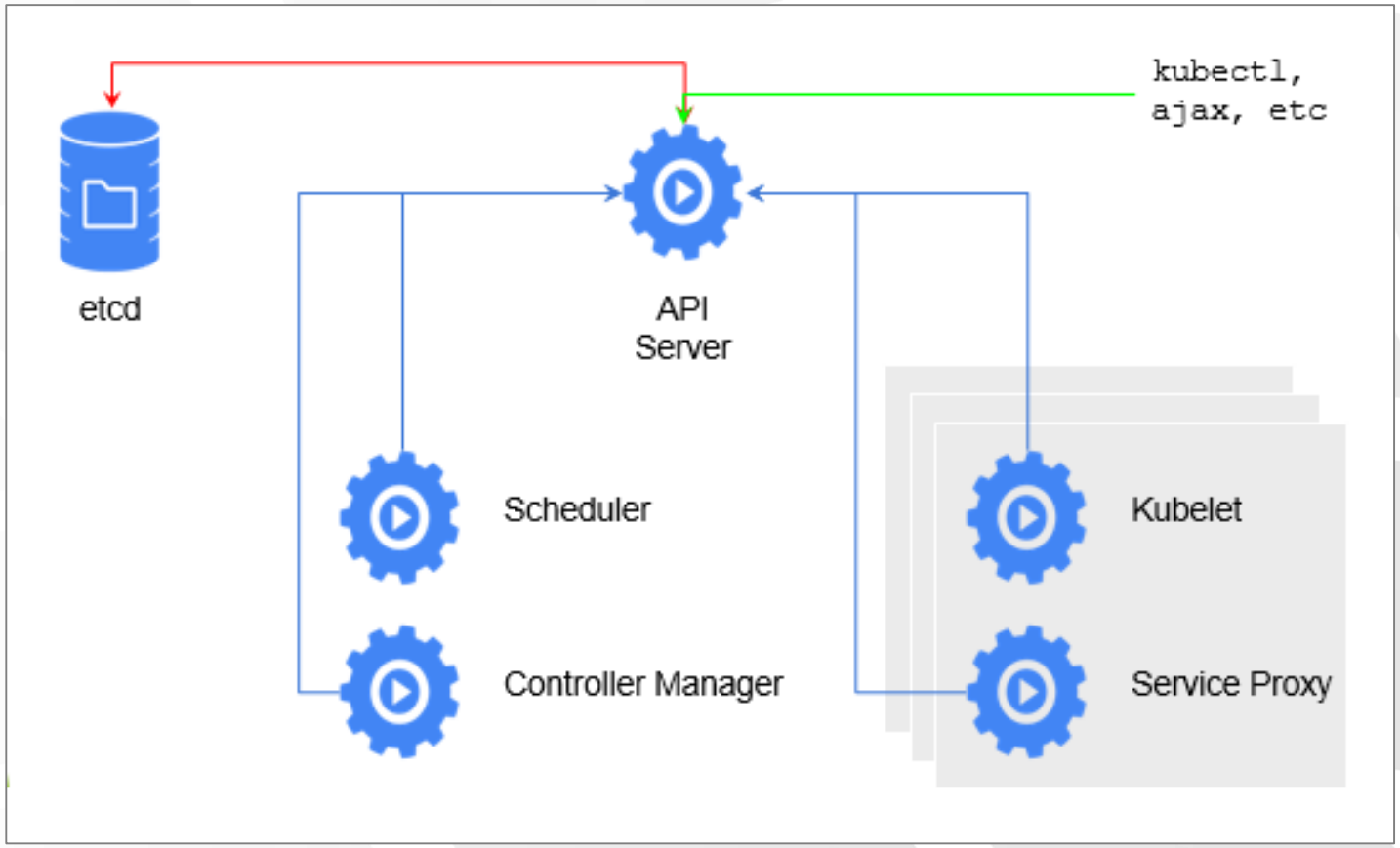
Why Kubernetes?

- Moving from monolithic apps to microservices
- Providing a consistent environment to applications
- Moving to continuous delivery: DevOps and NoOps
- Containers need managaement
- Runs on any cloud & bare metal

Main Features

- Monitoring
- Health Checking
- Horizontal Auto Scaling
- Service Discovery
- Rolling Deployment and Rollback
- Logging
- Load Balancing

High Level Architecture



etcd

- It stores the configuration information which can be used by each of the nodes in the cluster.
- It is a high availability key value store that can be distributed among multiple nodes.
- It is accessible only by Kubernetes API server as it may have some sensitive information.
- It is a distributed key value Store which is accessible to all.

API Server

- Kubernetes is an API server which provides all the operation on cluster using the API
- API server implements an interface, which means different tools and libraries can readily communicate with it.
- Kubeconfig is a package along with the server side tools that can be used for communication. It exposes Kubernetes API.

Controller Manager

- This component is responsible for most of the collectors that regulates the state of cluster and performs a task.
- In general, it can be considered as a daemon which runs in nonterminating loop and is responsible for collecting and sending information to API server.
- It works toward getting the shared state of cluster and then make changes to bring the current status of the server to the desired state.
- The controller manager runs different kind of controllers to handle nodes, endpoints, etc.

Scheduler

- It is a service in master responsible for distributing the workload
- It is responsible for tracking utilization of working load on cluster nodes and then placing the workload on which resources are available and accept the workload.
- The scheduler is responsible for workload utilization and allocating pod to new node.

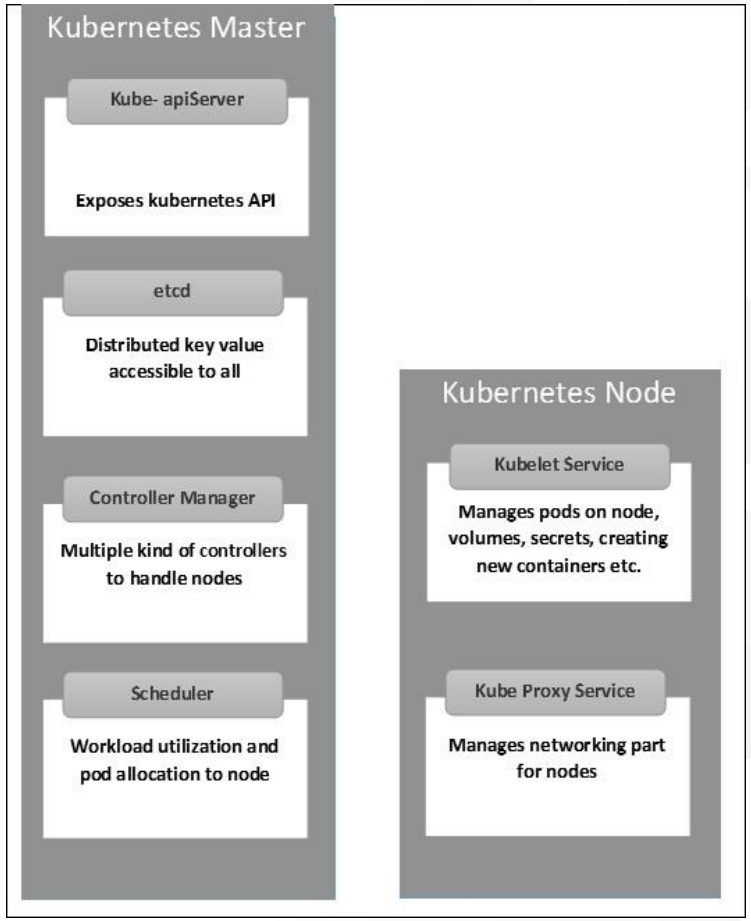
Kubelet Service

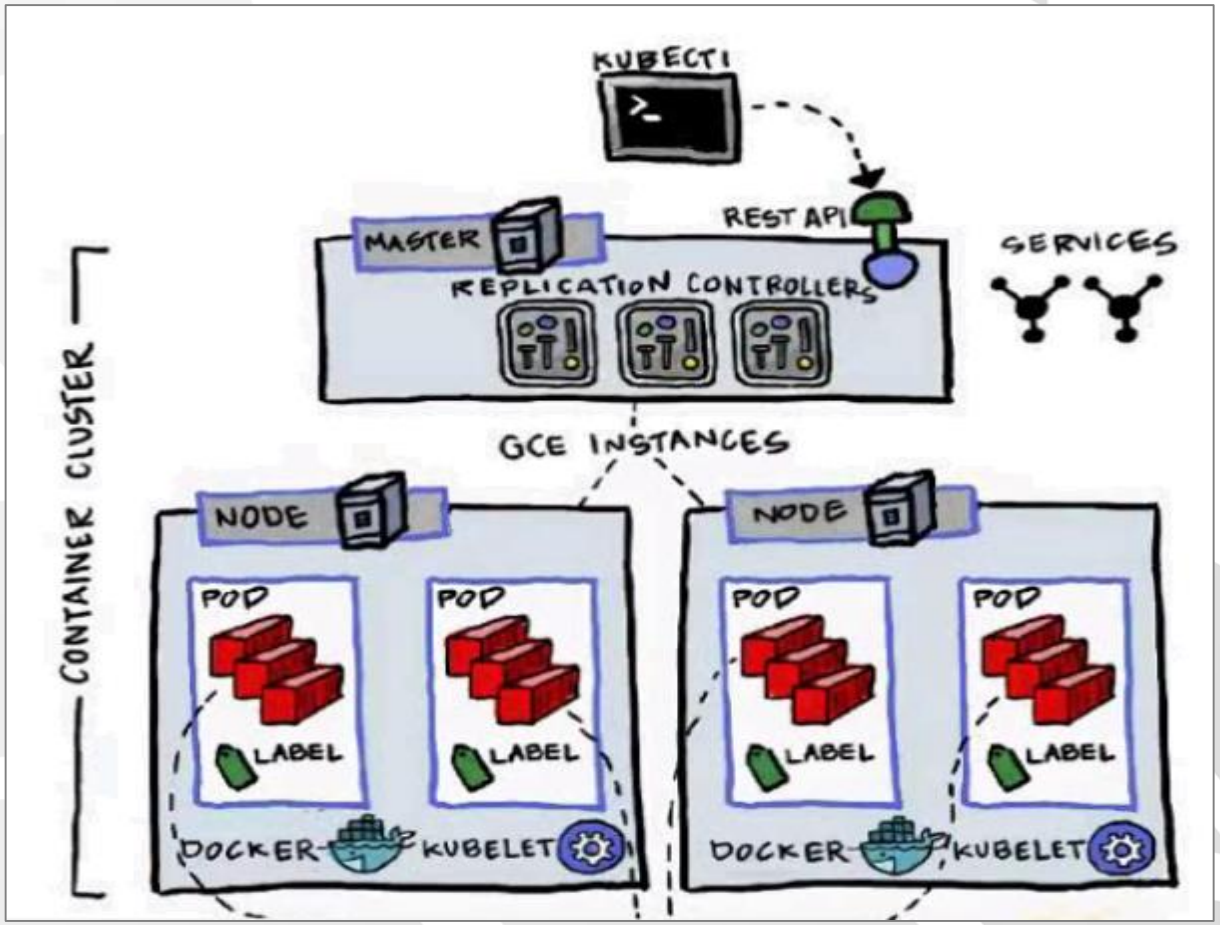
- This is a small service in each node responsible for relaying information to and from control plane service
- It interacts with etcd store to read configuration details and wright values.
- This communicates with the master component to receive commands and work.
- The kubelet process then assumes responsibility for maintaining the state of work and the node server.

Kubernetes Proxy Service

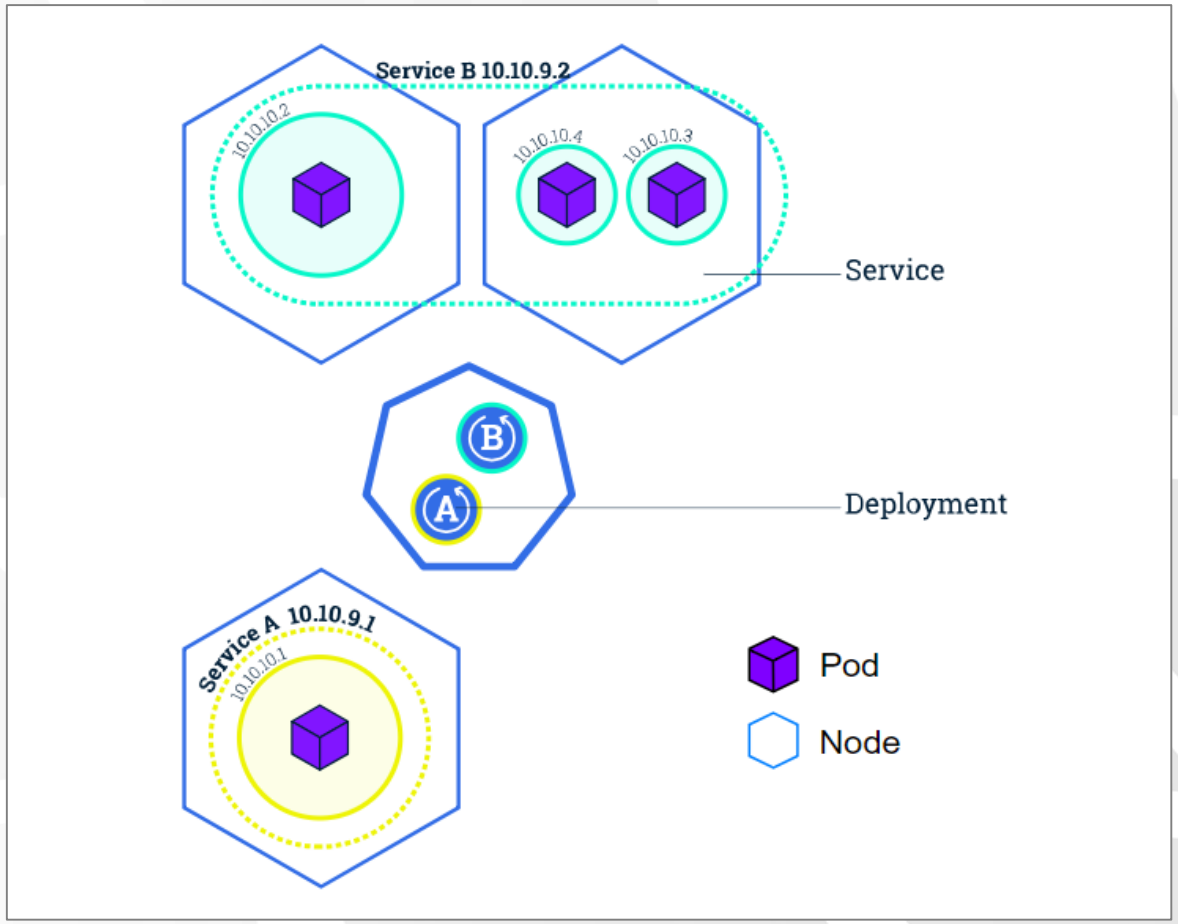
- This is a proxy service which runs on each node and helps in making services available to the external host.
- It helps in forwarding the request to correct containers and is capable of performing primitive load balancing.
- It makes sure that the networking environment is predictable and accessible and at the same time it is isolated as well.
- It manages pods on node, volumes, secrets, creating new containers' health checkup, etc.

Master and Node Structure





Kubernetes Objects and Workloads



Pods

- A pod is the most basic unit that Kubernetes deals with.
- Containers themselves are not assigned to hosts. Instead, one or more tightly coupled containers are encapsulated in an object called a pod.
- A pod generally represents one or more containers that should be controlled as a single application.
- users are encouraged to work with higher level objects that use pods or pod templates as base components.

Replication Controllers

- Is an object that defines a pod template and control parameters to scale identical replicas of a pod horizontally by increasing or decreasing the number of running copies.
- The replication controller is responsible for ensuring that the number of pods deployed in the cluster matches the number of pods in its configuration.
- If the number of replicas in a controller's configuration changes, the controller either starts up or kills containers to match the desired number.

Replication Sets

- are an iteration on the replication controller design with greater flexibility in how the controller identifies the pods it is meant to manage.
- Replication sets are meant to be used inside of additional, higher level units that provide that functionality.
- Replication sets are not able to do rolling updates to cycle backends to a new version like replication controllers can.

Deployments

- Deployments are upgraded and higher version of replication controller. They manage the deployment of replica sets which is also an upgraded version of the replication controller.
- They have the capability to update the replica set and are also capable of rolling back to the previous version.
- Deployments are a high level object designed to ease the life cycle management of replicated pods.

Jobs and Cron Jobs

- Provide a more task-based workflow where the running containers are expected to exit successfully after some time once they have completed their work.
- Jobs are useful if you need to perform one-off or batch processing instead of running a continuous service.
- Cron jobs can be used to schedule a job to execute in the future or on a regular, reoccurring basis.

Services

- Is a component that acts as a basic internal load balancer and ambassador for pods.
- A service groups together logical collections of pods that perform the same function to present them as a single entity.
- Any time you need to provide access to one or more pods to another application or to external consumers, you should to configure a service.
- Although services, by default, are only available using an internally routable IP address, they can be made available outside of the cluster by choosing one of several strategies.

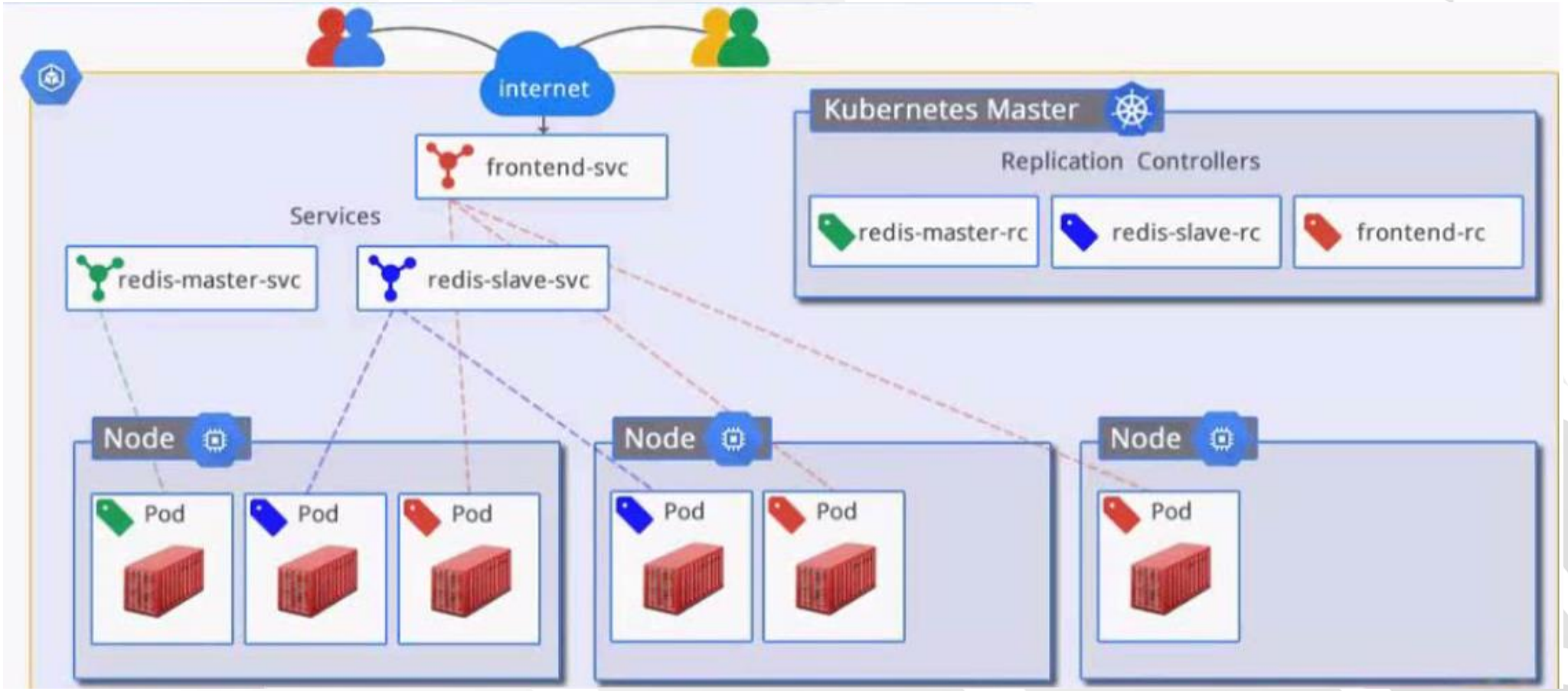
Volumes

- Reliably sharing data and guaranteeing its availability between container restarts is a challenge in many containerized environments.
- Kubernetes uses its own volumes abstraction that allows data to be shared by all containers within a pod and remain available until the pod is terminated.
- Persistent volumes are a mechanism for abstracting more robust storage that is not tied to the pod life cycle.

Labels and Annotations

- A label is a semantic tag that can be attached to Kubernetes objects to mark them as a part of a group.
- Services use labels to understand the backend pods they should route requests to.
- Annotations are a similar mechanism that allows you to attach arbitrary key-value information to an object.
- Annotations are a way of adding rich metadata to an object that is not helpful for selection purposes.

Kubernetes Architecture



Questions

Demo