Introduction to Kubernetes

Ihor Dvoretskyi, Developer Advocate
Cloud Native Computing Foundation
@idvoretskyi

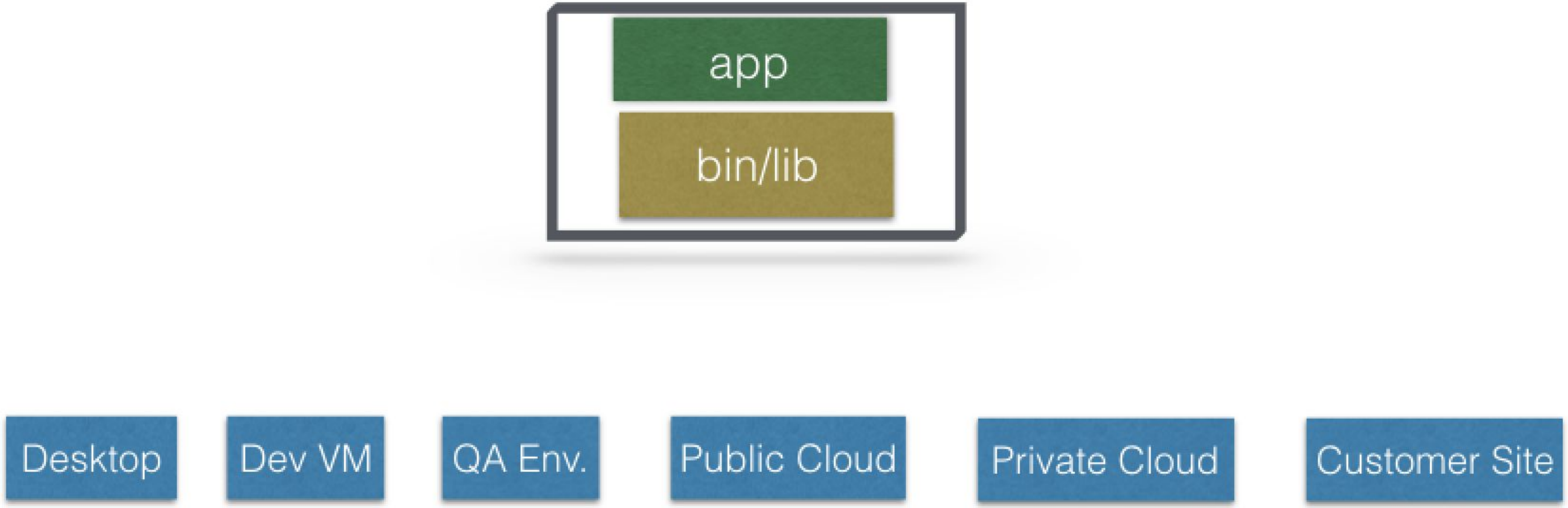
kubernetes

LinuxFoundationX: LFS158x

- This is a brief overview of Kubernetes, based on the Linux Foundation Course LFS158x
- Full course is available on [EDX for free]()

kubernetes

What Are Containers?



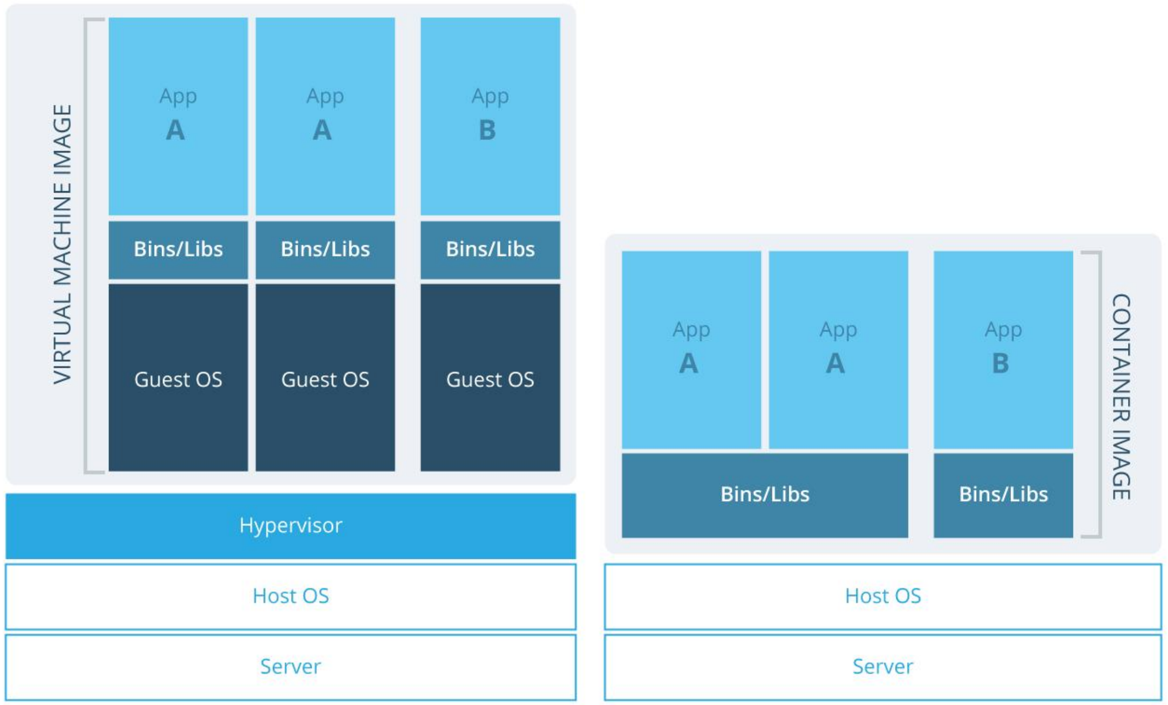
kubernetes

Why Containers?

- Portability
- Immutability
- Ease of update (layers)
- Pre-packaged containers
- Lightweight (create, destroy)
- Excellent for microservices
- Standard repositories

kubernetes

Containers vs Virtual Machines



kubernetes

Container Management

- Containers are powerful, running one container is useless
- Applications using microservices will include components, bundled into multiple containers
- To manage them, the Container scheduling & orchestration tools are required

Kubernetes

Docker Swarm/ Compose

Mesos

Kubernetes

What Is Kubernetes?

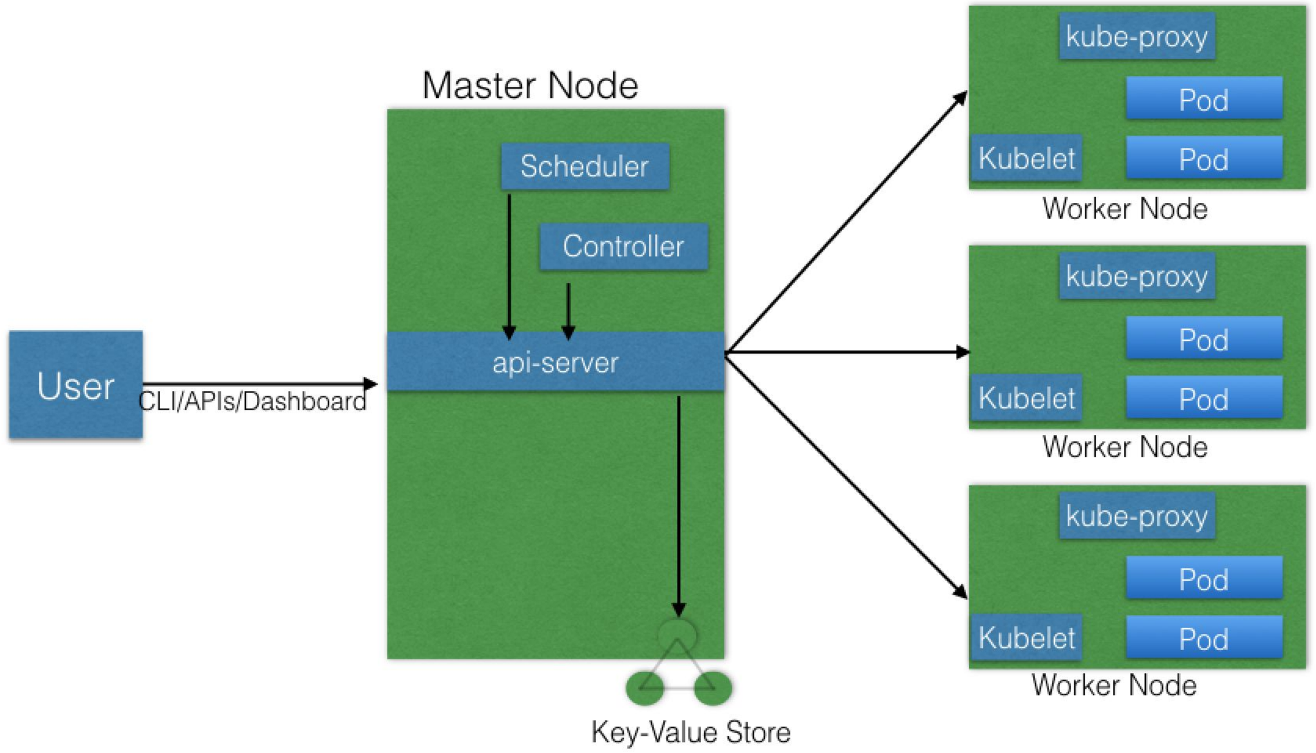
- Kubernetes is an open-source system for automating deployment, scaling, and management of containerized applications.
- Originally developed by Google (inspired by the internal system called Borg)

Key Kubernetes Features

- Self-healing
- Horizontal scaling
- Service discovery and Load balancing
- Automated rollouts and rollbacks
- Secrets and configuration management
- Storage orchestration

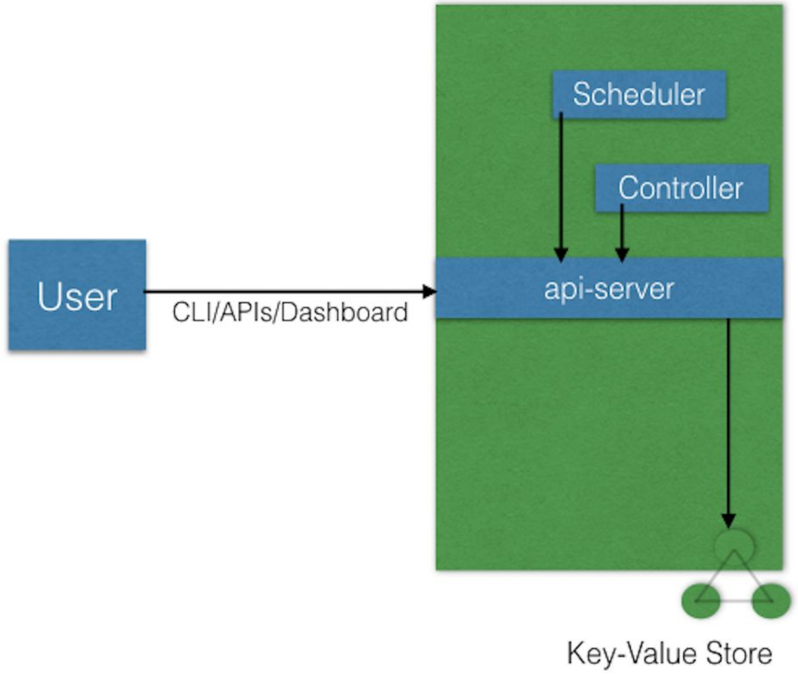
Kubernetes Architecture

Kubernetes Architecture



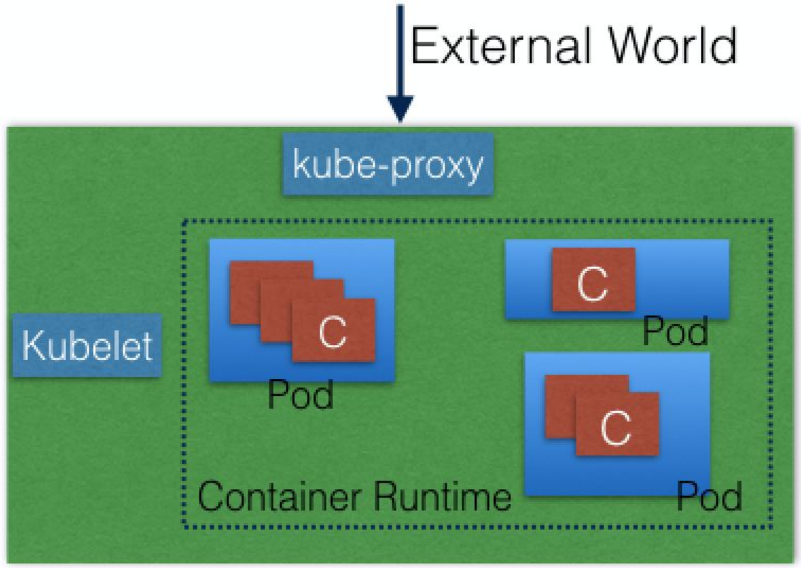
Master Node aka Control Plane

- The master node is responsible for managing the Kubernetes cluster, and it is the entry point for all administrative tasks
- User can communicate to the Master Node via the CLI, the GUI (Dashboard), or via APIs.
- For fault tolerance purposes, there can be more than one master node in the cluster (Multi-Master).

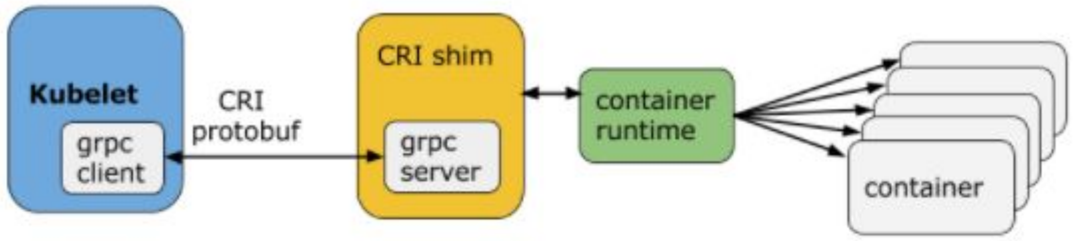


Worker Node

- A worker node is a machine (VM, physical server, etc.) which runs the applications using Pods and is controlled by the master node
- Pods are scheduled on the worker nodes, which have the necessary tools to run and connect them
- A Pod is the atomic of the Kubernetes workloads



Container Runtime Interface



Kubernetes Configuration

Kubernetes Configurations

- All-in-One Single-Node Installation
- Single-Node etcd, Single-Master, and Multi-Worker Installation
- Single-Node etcd, Multi-Master, and Multi-Worker Installation
- Multi-Node etcd, Multi-Master, and Multi-Worker Installation

Localhost Installation

- There are a few localhost installation options available to deploy single- or multi-node Kubernetes clusters on our workstation/laptop:
  - Minikube
  - Ubuntu on LXD
- Minikube is the preferred and community-recommended way to create an all-in-one Kubernetes setup

kubernetes

Non-Local Installations

- On-Premise Installation
  - On-Premise VMs
  - On-Premise Bare Metal
- Public Cloud Installation
  - Managed Solutions
    - Google Kubernetes Engine (GKE)
    - Azure Container Service (AKS)
    - Amazon Elastic Container Service for Kubernetes (EKS)
  - Self-managed Solutions
    - Kubeadm
    - Kops
    - Kubespray

kubernetes

Minikube

Minikube

- The community-recommended way to deploy Kubernetes locally
- Minikube runs inside a VM on:
  - Windows
  - Mac
  - Linux
    - *Also available without VM, being run directly on host (Docker is required)*

Installing Minikube (Requirements)

- kubectl
- macOS
  - Hyperkit driver, xhyve driver, VirtualBox, or VMware Fusion
- Linux
  - VirtualBox or KVM
  - NOTE: Minikube also supports a --vm-driver=none option that runs the Kubernetes components on the host and not in a VM
- Windows
  - VirtualBox or Hyper-V
- VT-x/AMD-v virtualization must be enabled in BIOS
- Internet connection on first run

kubernetes

Installing Minikube

- Use the link to find installation instructions for your OS: https://github.com/kubernetes/minikube#installation
- Linux:
  - *curl -Lo minikube https://storage.googleapis.com/minikube/releases/latest/minikube-linux-amd64 && chmod +x minikube && sudo cp minikube /usr/local/bin/ && rm minikube*
- macOS (with *brew*):
  - *brew cask install minikube*
- Windows (with *chocolatey*):
  - *choco install minikube*

kubernetes

Accessing Minikube

Any healthy running Kubernetes cluster can be accessed via one of the following methods:

- Command Line Interface (CLI)
- Graphical User Interface (GUI)
- APIs

kubernetes

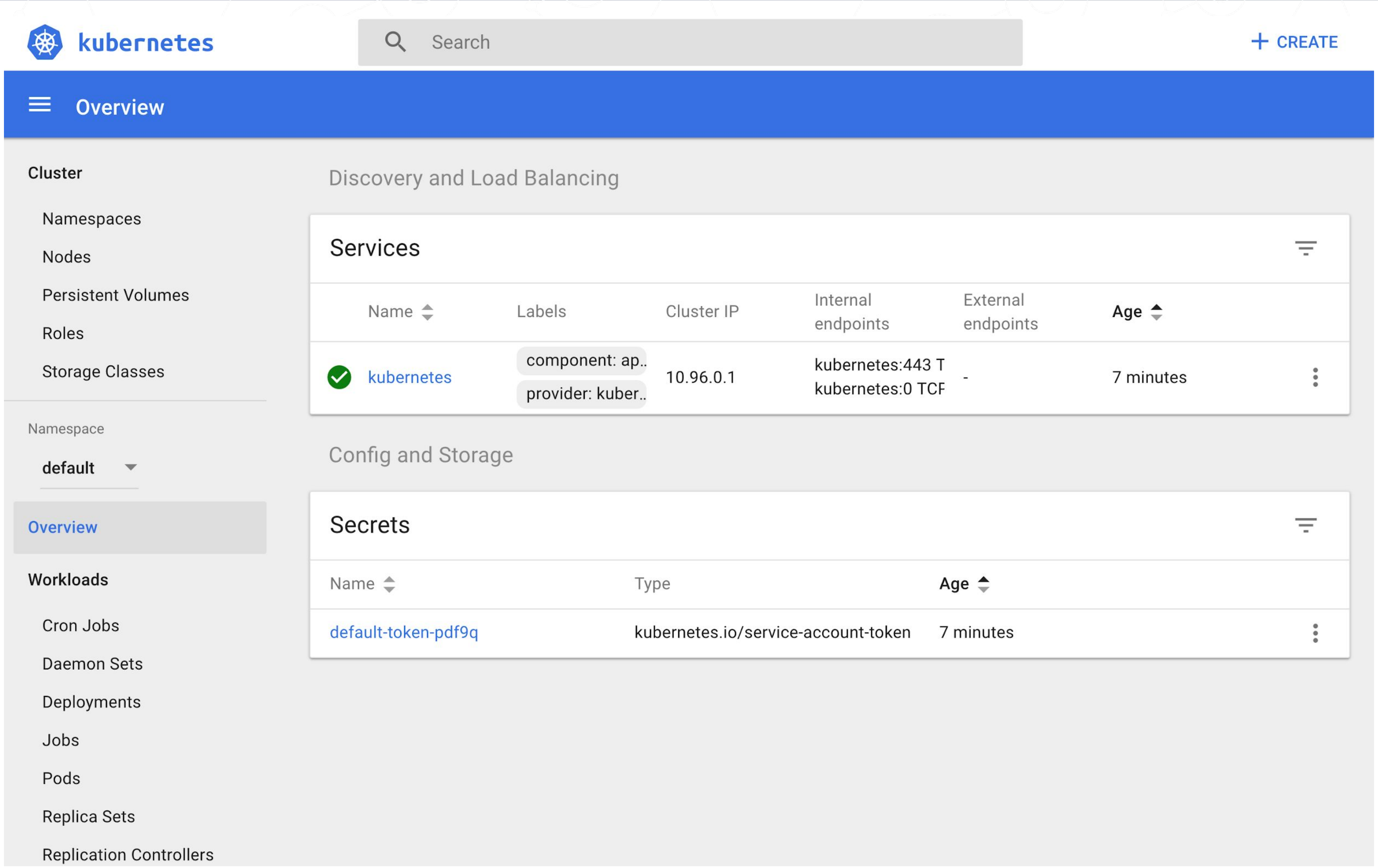
CLI: kubectl

- Use the link to find installation instructions for your OS:
  - https://kubernetes.io/docs/tasks/tools/install-kubectl/
- Check if it is working:
  - *kubectl help*

UI: dashboard

- Enabled by default in Minikube, accessible via:
    - *minikube dashboard*

kubernetes



kubernetes

Search

+ CREATE

Overview

Cluster

- Namespaces
- Nodes
- Persistent Volumes
- Roles
- Storage Classes

Namespace

default

Overview

Workloads

- Cron Jobs
- Daemon Sets
- Deployments
- Jobs
- Pods
- Replica Sets
- Replication Controllers

Discovery and Load Balancing

Services

	Name	Labels	Cluster IP	Internal endpoints	External endpoints	Age
✓	kubernetes	component: ap.. provider: kuber..	10.96.0.1	kubernetes:443 T kubernetes:0 TCF	-	7 minutes

Config and Storage

Secrets

Name	Type	Age
default-token-pdf9q	kubernetes.io/service-account-token	7 minutes

Using the 'kubectl proxy' Command

```
$ kubectl proxy

Starting to serve on 127.0.0.1:8001

$ curl http://localhost:8001/
```

Kubernetes Objects

Kubernetes Objects

Kubernetes has a very rich object model, with which it represents different persistent entities in the Kubernetes cluster. Those entities describe:

- What containerized applications we are running and on which node
- Application resource consumption
- Different policies attached to applications, like restart/upgrade policies, fault tolerance, etc.

Kubernetes objects include:

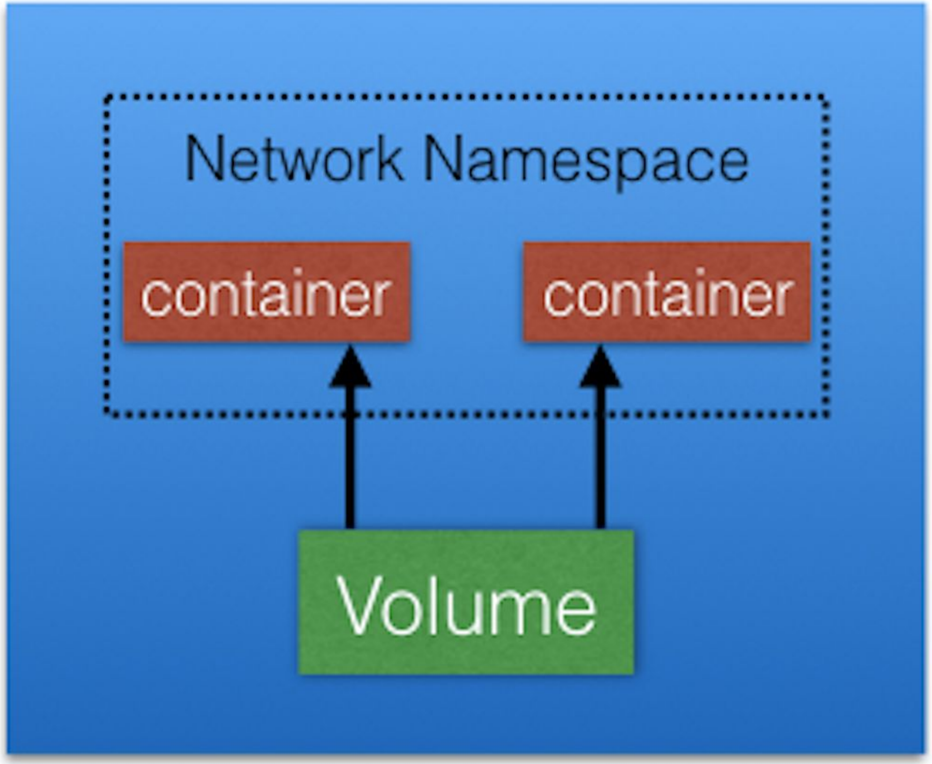
- Pods
- ReplicaSets
- Deployments
- Namespaces

kubernetes

Pods

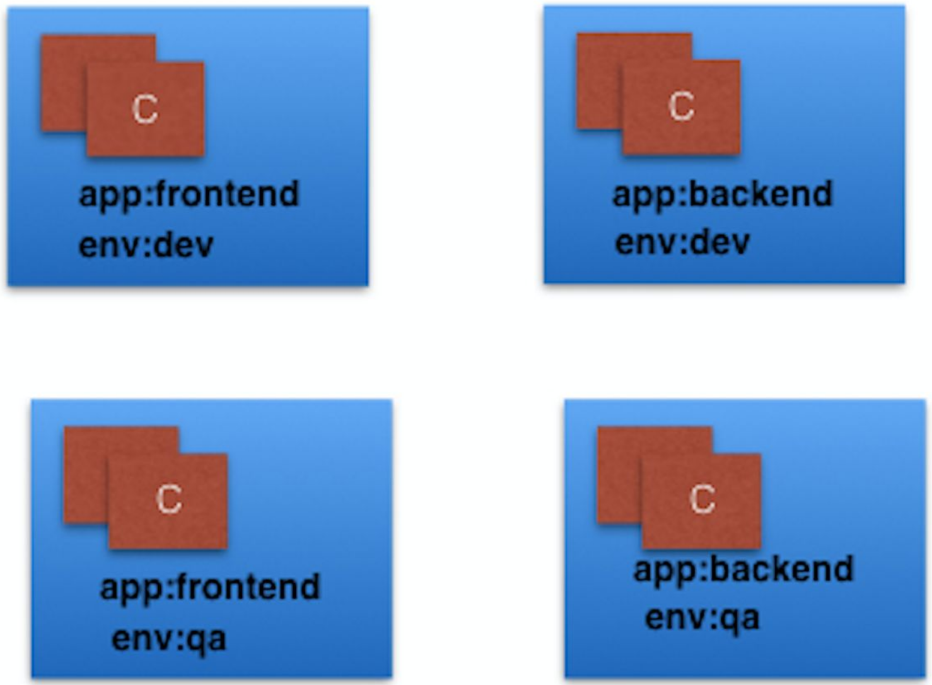
A Pod is the smallest and simplest Kubernetes object. It is the unit of deployment in Kubernetes, which represents a single instance of the application. A Pod is a logical collection of one or more containers, which:

- Are scheduled together on the same host
- Share the same network namespace
- Mount the same external storage (volumes).



Labels

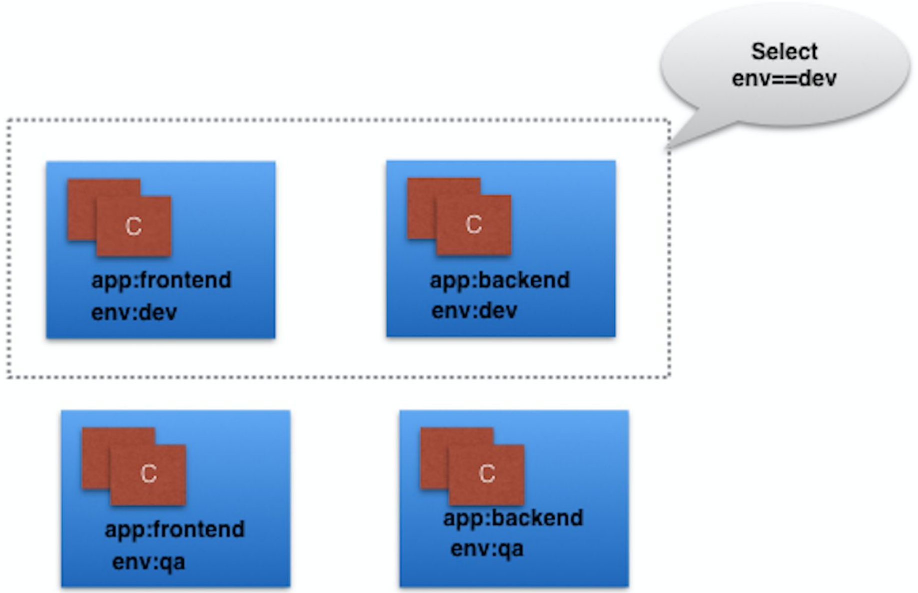
- Labels are key-value pairs that can be attached to any Kubernetes objects (e.g. Pods).
- Labels are used to organize and select a subset of objects, based on the requirements in place.
- Many objects can have the same Label(s).
- Labels do not provide uniqueness to objects.



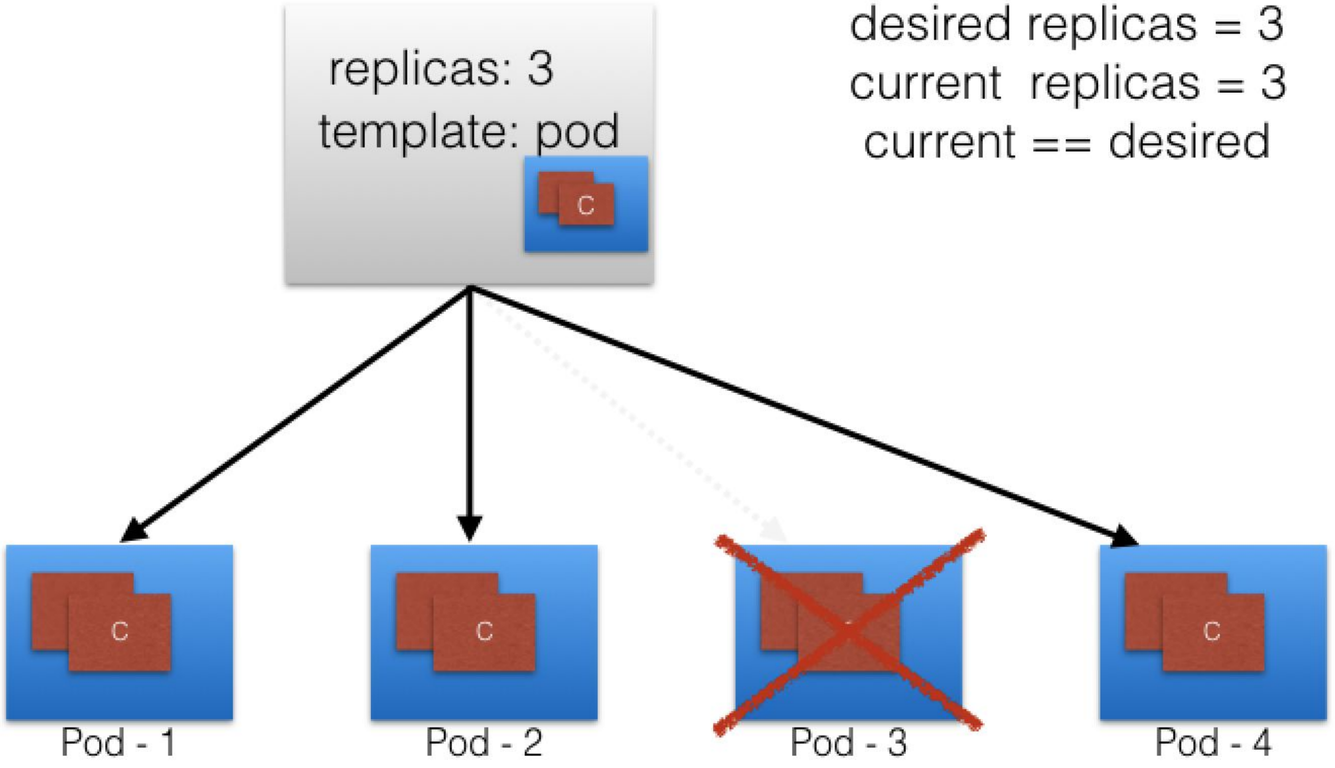
Label Selectors

With Label Selectors, we can select a subset of objects. Kubernetes supports two types of Selectors:

- Equality-Based Selectors
  - Equality-Based Selectors allow filtering of objects based on Label keys and values
    - *env==dev*
- Set-Based Selectors
  - Set-Based Selectors allow filtering of objects based on a set of values
    - *env in (dev,qa)*



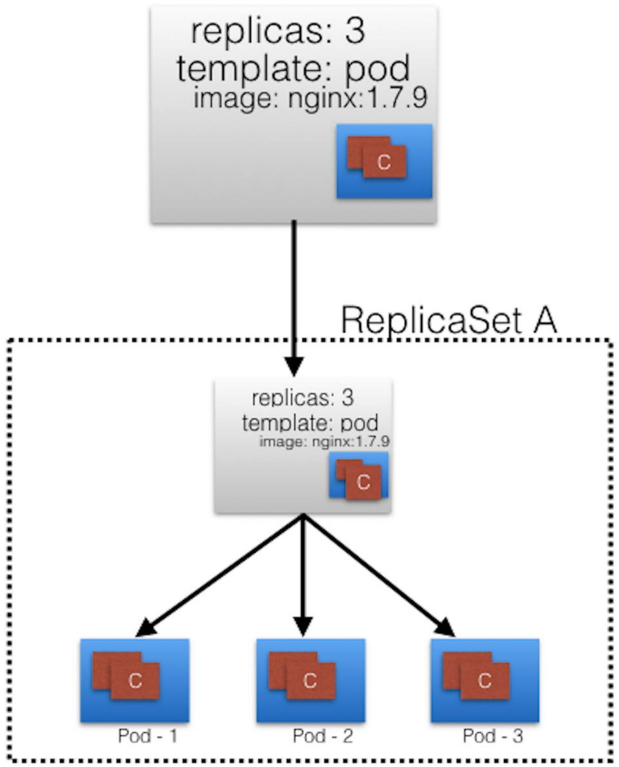
ReplicaSets



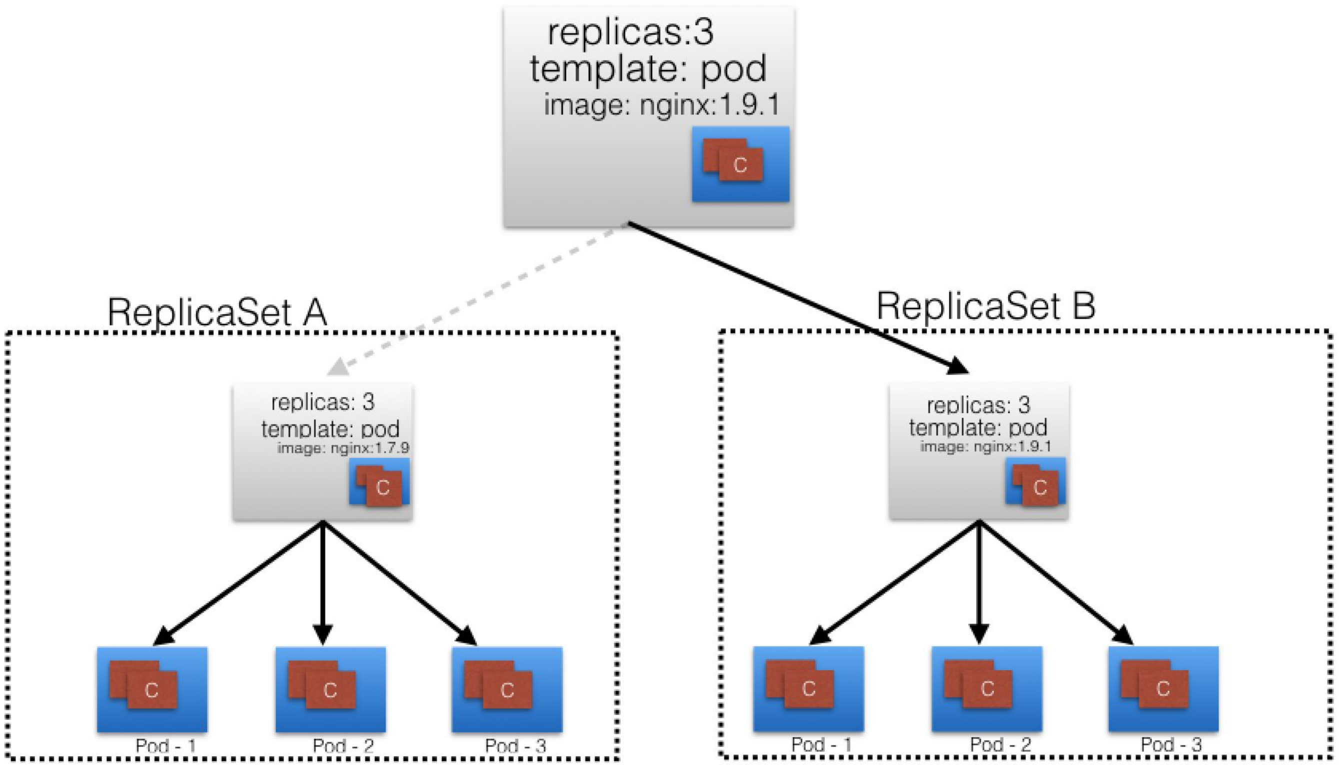
desired replicas = 3
current replicas = 3
current == desired

Deployments

- Deployment objects provide declarative updates to Pods and ReplicaSets
- The DeploymentController is part of the master node's controller manager, and it makes sure that the current state always matches the desired state.



Deployment rollout



Namespaces

- If we have numerous users whom we would like to organize into teams/projects, we can partition the Kubernetes cluster into sub-clusters using Namespaces.
- Kubernetes creates two default Namespaces:
  - kube-system
  - default

```
$ kubectl get namespaces

NAME         STATUS    AGE

default      Active    11h

kube-public   Active    11h

kube-system   Active    11h
```

Authentication, Authorization, and Admission Control

Overview

- Authentication
  - Logs in a user.
- Authorization
  - Authorizes the API requests added by the logged-in user.
- Admission Control
  - Software modules that can modify or reject the requests based on some additional checks, like Quota.

Authentication

Kubernetes has two kinds of users (while they are not *users* in a common meaning):

- Normal Users
  - They are managed outside of the Kubernetes cluster via independent services like User/Client Certificates, a file listing usernames/passwords, Google accounts, etc.
- Service Accounts
  - With Service Account users, in-cluster processes communicate with the API server to perform different operations
  - Most of the Service Account users are created automatically via the API server, but they can also be created manually.

Authorization

- After a successful authentication, users can send the API requests to perform different operations
- Then, those API requests get authorized by Kubernetes using various authorization modules.

Authorization modules

- Node Authorizer
  - Node authorization is a special-purpose authorization mode which specifically authorizes API requests made by kubelets.
- Attribute-Based Access Control (ABAC) Authorizer
  - With the ABAC authorizer, Kubernetes grants access to API requests, which combine policies with attributes.
- Webhook Authorizer
  - With the Webhook authorizer, Kubernetes can offer authorization decisions to some third-party services, which would return true for successful authorization, and false for failure.
- Role-Based Access Control (RBAC) Authorizer
  - In Kubernetes, we can have different roles that can be attached to subjects like users, service accounts, etc.

Admission Control

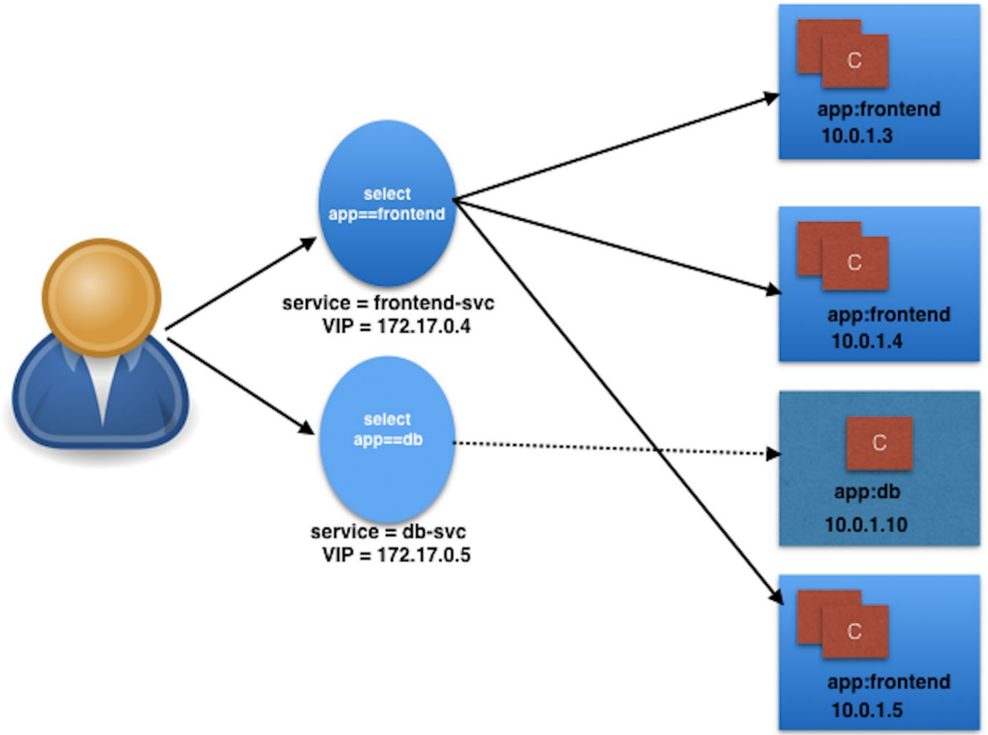
- Admission control is used to specify granular access control policies, which include allowing privileged containers, checking on resource quota, etc.
- We force these policies using different admission controllers, like ResourceQuota, AlwaysAdmit, DefaultStorageClass, etc.
- They come into effect only after API requests are authenticated and authorized.

Services

Overview

- To access the application, a user/client needs to connect to the Pods
- As Pods are ephemeral in nature, resources like IP addresses allocated to it cannot be static
- Pods could die abruptly or be rescheduled based on existing requirements
- Unexpectedly, the Pod to which the user/client is connected dies, and a new Pod is created by the controller
- The new Pod will have a new IP address, which will not be known automatically to the user/client of the earlier Pod
- Kubernetes provides a higher-level abstraction called **Service**, which logically groups Pods and a policy to access them.

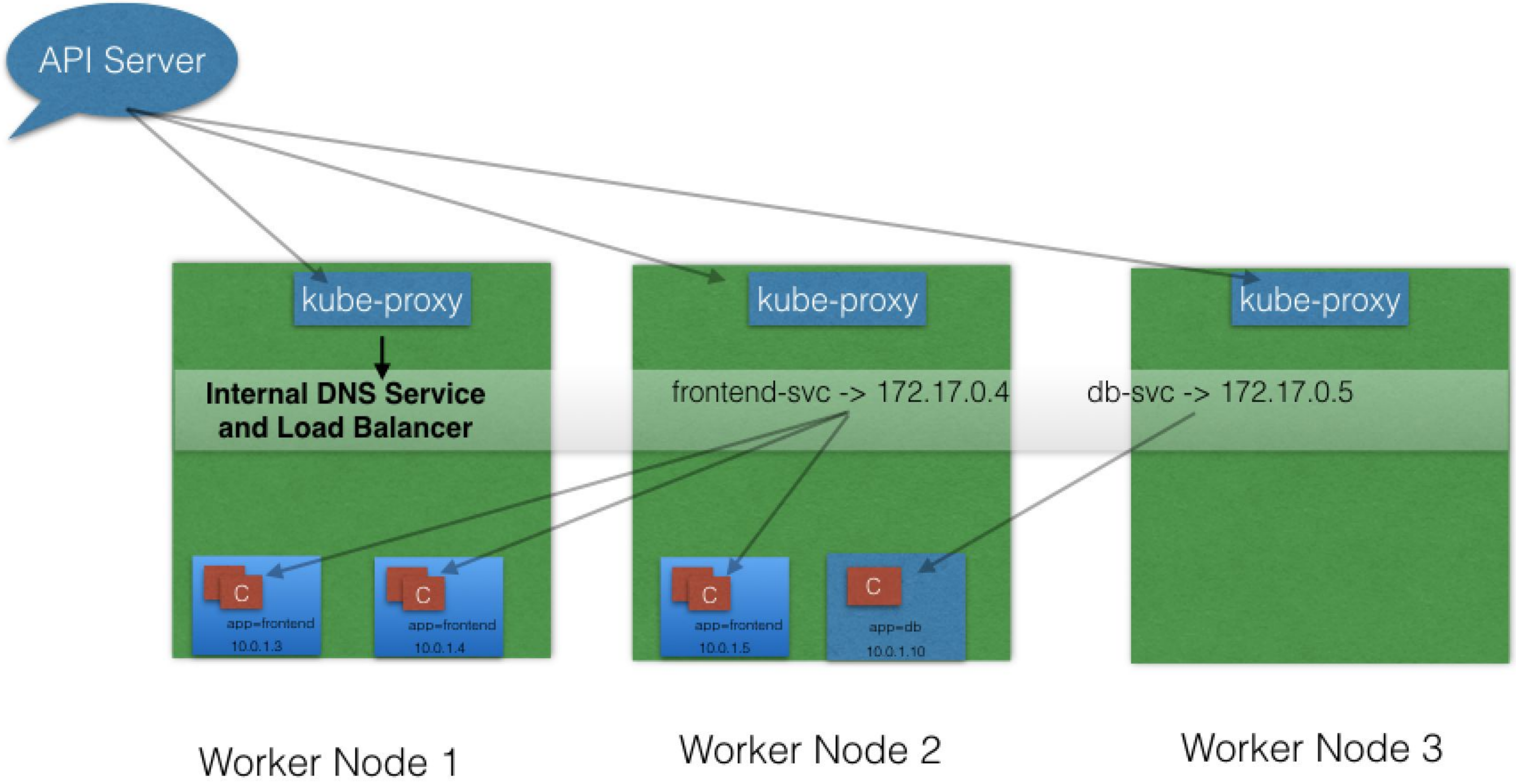
Grouping of Pods using the Service object



Kube-proxy

- All of the worker nodes run a daemon called kube-proxy, which watches the API server on the master node for the addition and removal of Services and endpoints.
- For each new Service, on each node, kube-proxy configures the iptables rules to capture the traffic for its ClusterIP and forwards it to one of the endpoints.
- When the service is removed, kube-proxy removes the iptables rules on all nodes as well.

kube-proxy, Services, and Endpoints



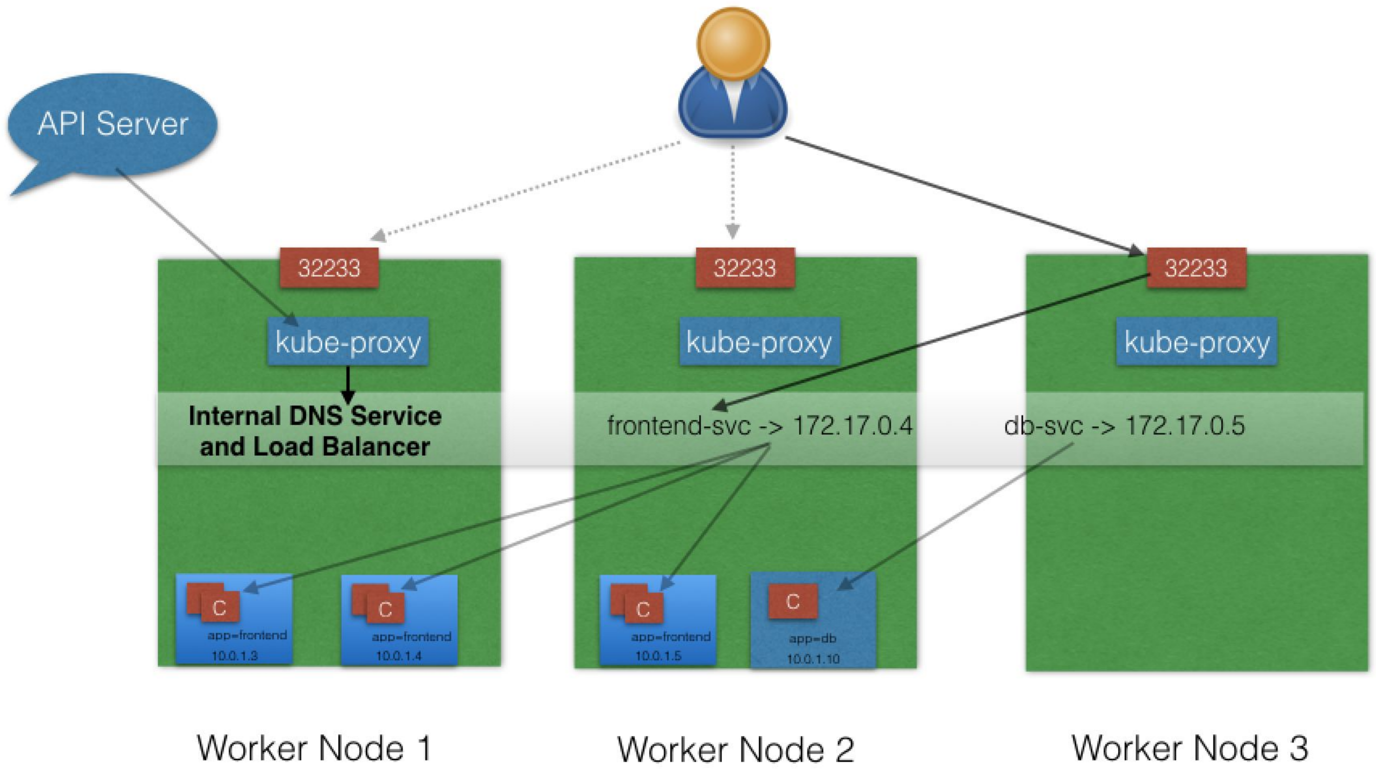
ServiceType

- While defining a Service, we can also choose its access scope. We can decide whether the Service:
    - Is only accessible within the cluster
    - Is accessible from within the cluster and the external world
    - Maps to an external entity which resides outside the cluster.
- Access scope is decided by ServiceType, which can be mentioned when creating the Service.

ServiceType: ClusterIP and NodePort

- ClusterIP is the default ServiceType.
  - A Service gets its Virtual IP address using the ClusterIP
  - That IP address is used for communicating with the Service and is accessible only within the cluster.
- The NodePort ServiceType is useful when we want to make our Services accessible from the external world.
  - The end-user connects to the worker nodes on the specified port, which forwards the traffic to the applications running inside the cluster.
  - To access the application from the external world, administrators can configure a reverse proxy outside the Kubernetes cluster and map the specific endpoint to the respective port on the worker nodes.

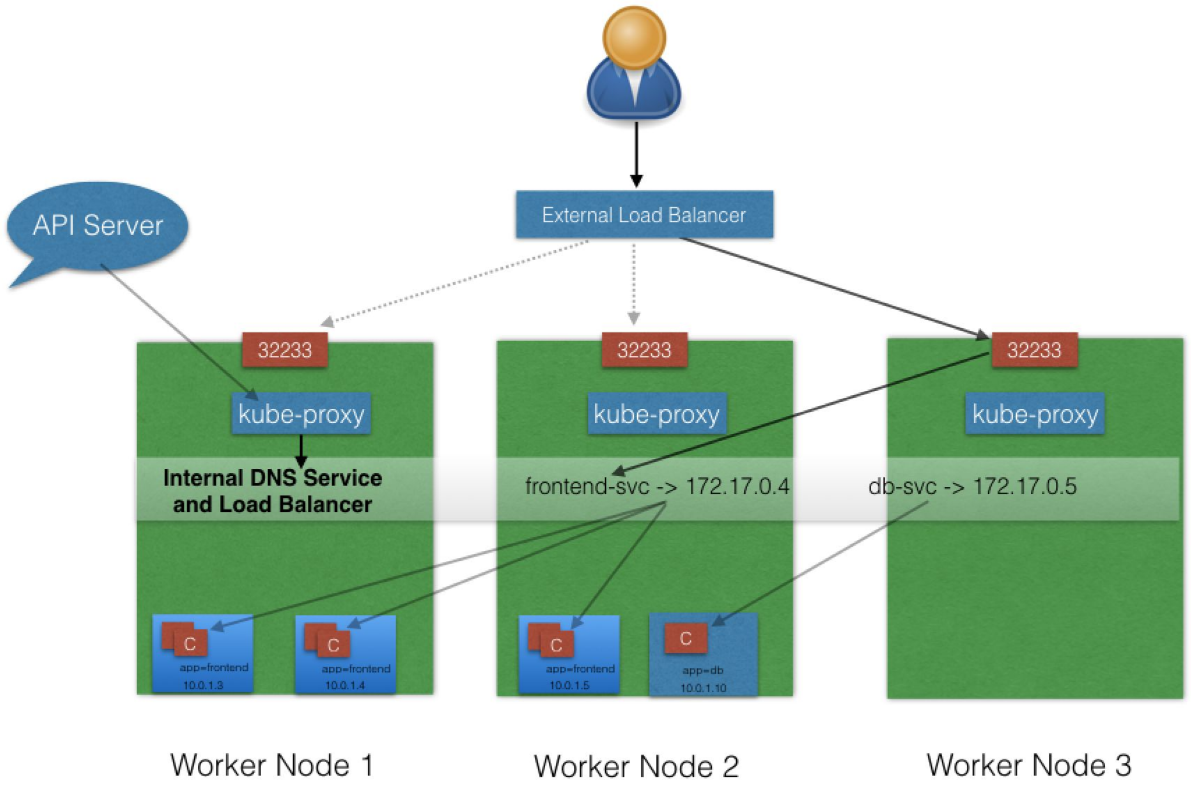
ServiceType: ClusterIP and NodePort



ServiceType: LoadBalancer

- With the **LoadBalancer** ServiceType:
    - NodePort and ClusterIP Services are automatically created, and the external load balancer will route to them
    - The Services are exposed at a static port on each worker node
    - The Service is exposed externally using the underlying cloud provider's load balancer feature.
- The **LoadBalancer** ServiceType will only work if the underlying infrastructure supports the automatic creation of Load Balancers and have the respective support in Kubernetes. Examples:
    - Google Cloud Platform
    - AWS

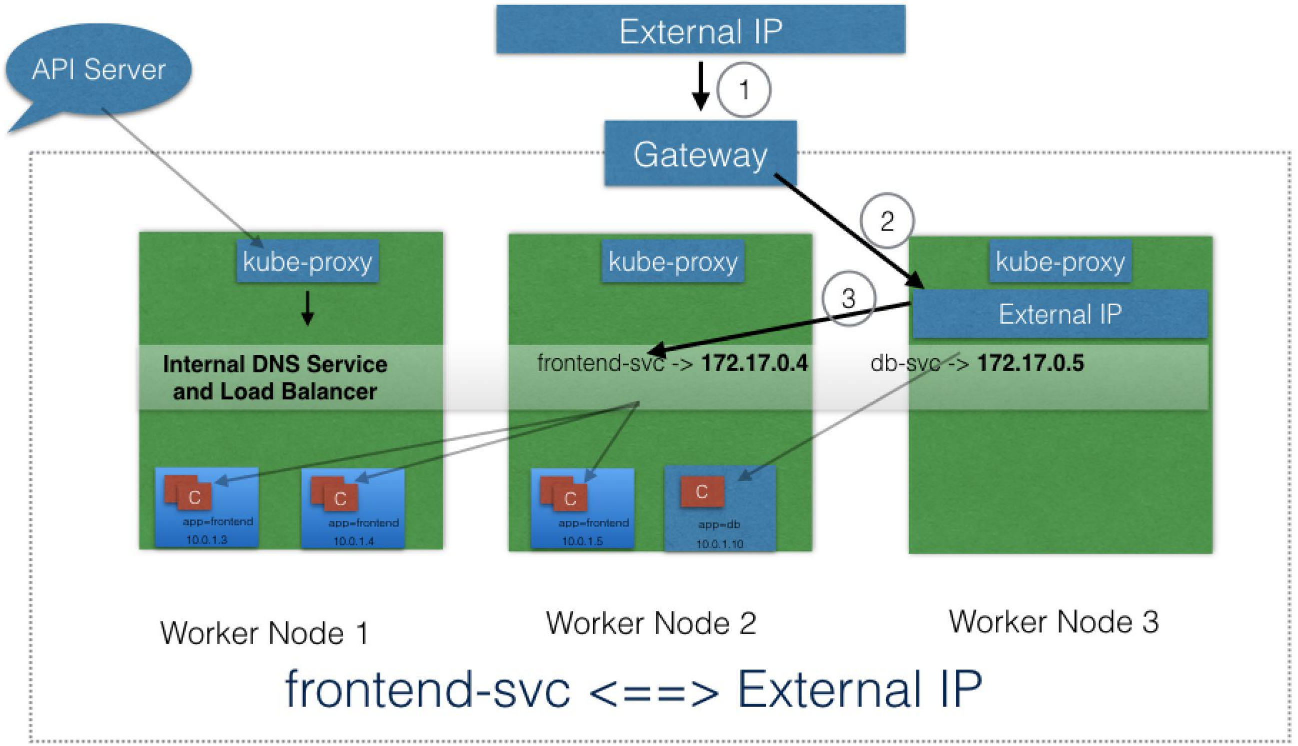
ServiceType: LoadBalancer



ServiceType: ExternalIP

- A Service can be mapped to an ExternalIP address if it can route to one or more of the worker nodes.
- Traffic that is ingressed into the cluster with the ExternalIP (as destination IP) on the Service port, gets routed to one of the the Service endpoints.

ServiceType: ExternalIP



Deploying a Standalone Application

Deploying the Application Using the CLI

- Use the sample YAML file available at:
  - https://github.com/idvoretskyi/intro-to-kubernetes/blob/master/webserver.yaml
- Creating a webserver Deployment:
  - *kubectl create -f webserver.yaml*
- This will also create a ReplicaSet and Pods, as defined:
  - *kubectl get replicasets*

Creating a Service and Exposing It

- Use the sample YAML file available at:
  - https://github.com/idvoretskyi/intro-to-kubernetes/blob/master/webserver-svc.yaml
- Using kubectl, create the Service:
  - *kubectl create -f webserver-svc.yaml*
- List the Services:
  - *kubectl get svc*
- To get more details about the service:
  - *kubectl describe svc web-service*
- Test the app:
  - *minikube service web-service --url #to get an IP address of the service*
  - *curl http://192.168.99.100:32742*

Kubernetes Community

Getting Started with the Community

- Weekly Meetings
- Meetup Groups
- Slack Channels
- Mailing Lists
- Special Interest Groups (SIGs)
- Stack Overflow
- CNCF Events
    - KubeCon + CloudNativeCon Europe
    - KubeCon + CloudNativeCon China
    - KubeCon + CloudNativeCon North America

kubernetes

Next

- Expand your Kubernetes knowledge and skills by enrolling in paid courses offered by The Linux Foundation:
    - self-paced LFS258 - Kubernetes Fundamentals
    - instructor-led LFS458 - Kubernetes Administration
- Prepare for:
    - Certified Kubernetes Administrator exam (CKA)
    - Certified Kubernetes Application Developer Program (CKAD), offered by the Cloud Native Computing Foundation

kubernetes

Questions?

@idvoretskyi
kubernetes.io
cncf.io