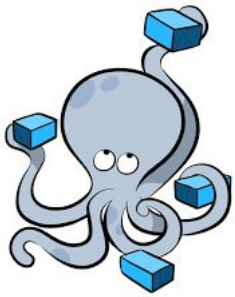
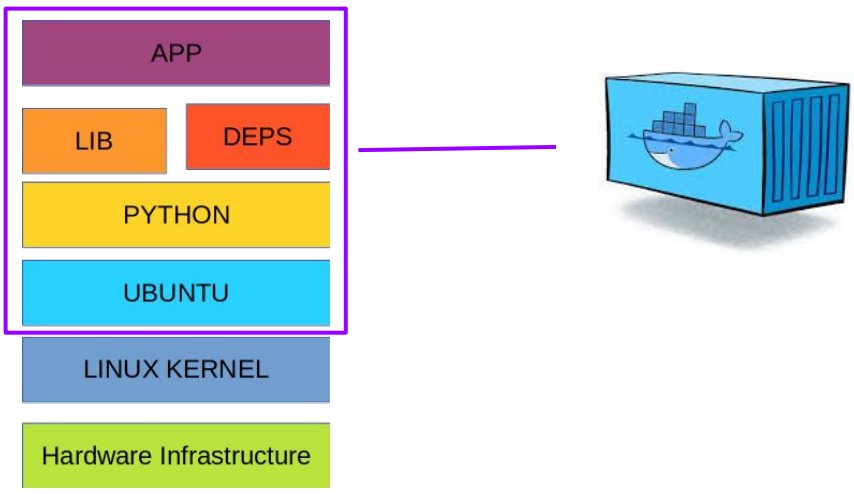
Introduction

Table of Contents

1 Orchestration

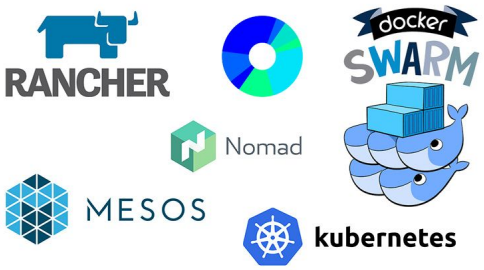


Orchestration

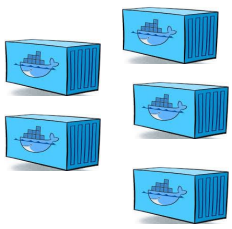


Orchestration

- Containers are great, but when you get lots of them running, at some point, you need them all working together in harmony to solve business problems.
- Tools to manage, scale, and maintain containerized applications are called orchestrators, and the most common example of this is **Kubernetes**.



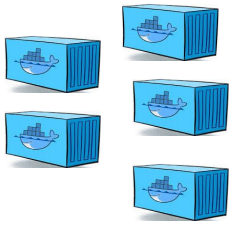
Orchestration



Container orchestration is used to automate the following tasks at scale:

- Provisioning and deployments of containers
- Availability of containers
- Load balancing, traffic routing and service discovery of containers
- Health monitoring of containers

Orchestration



- Securing the interactions between containers.
- Configuring and scheduling of containers
- Allocation of resources between containers

2 Declarative vs Imperative

Declarative vs Imperative

imperative focuses on **how** and declarative focuses on **what**.

Imperative approach:

1. Build the foundation
2. Put in the framework
3. Add the walls
4. Add the doors and windows

Declarative approach:

I want a tiny and cute house.

3

What is Kubernetes?

kubernetes

What is Kubernetes?

- Born in Google
- Donated to CNCF in 2014
- Open source (Apache 2.0)
- v1.0 July 2015
- Written in Go/Golang
- Often shortened to k8s

Kubernetes
k8 s

CNCF: Cloud Native Computing Foundation

What is Kubernetes?

➢ Kubernetes is **Open Source Orchestration** system for Containerized Applications.
➢ Kubernetes is a platform that **eliminates the manual processes** involved in **deploying** containerized applications.
➢ Kubernetes used to manage the **State of Containers**.

- Start Containers on Specific Nodes.
- Restart Containers when gets Killed.
- Move containers from one Node to Another.

4 Why you need Kubernetes?

Why you need Kubernetes?

Containers are a perfect way to get the applications packaged and run. In a production environment, you should manage the containers that run the applications and ensure no downtime.

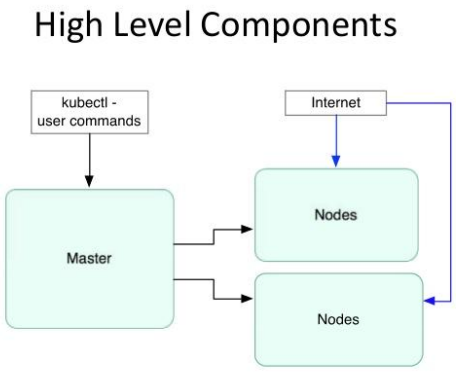
Why you need Kubernetes?

Kubernetes supplies you with:

- Service discovery and load balancing
- Storage orchestration
- Automated rollouts and rollbacks
- Automatic bin packing
- Self-healing
- Secret and configuration management

5

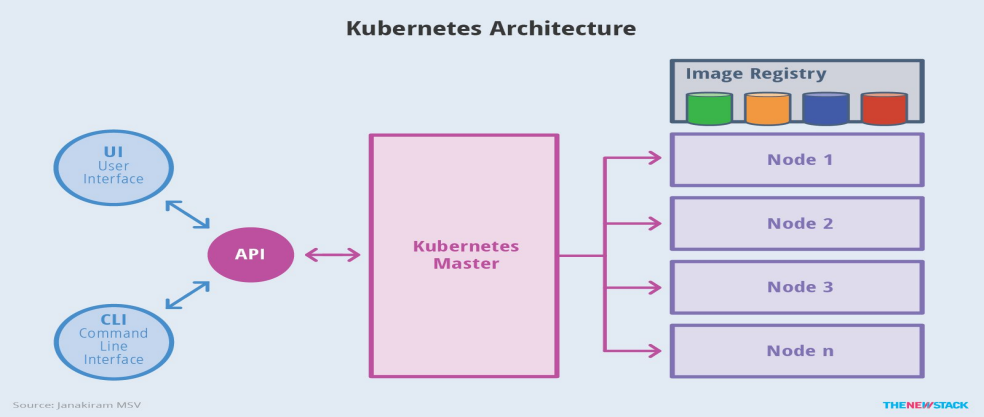
Kubernetes Components



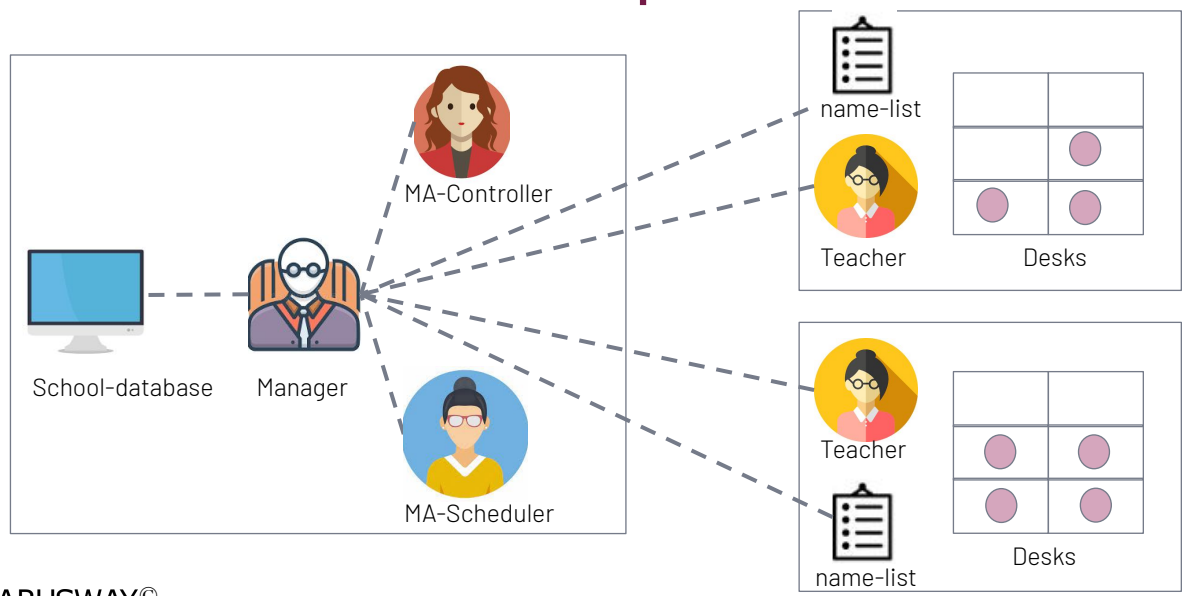
Kubernetes Components

Kubernetes has the following main components:

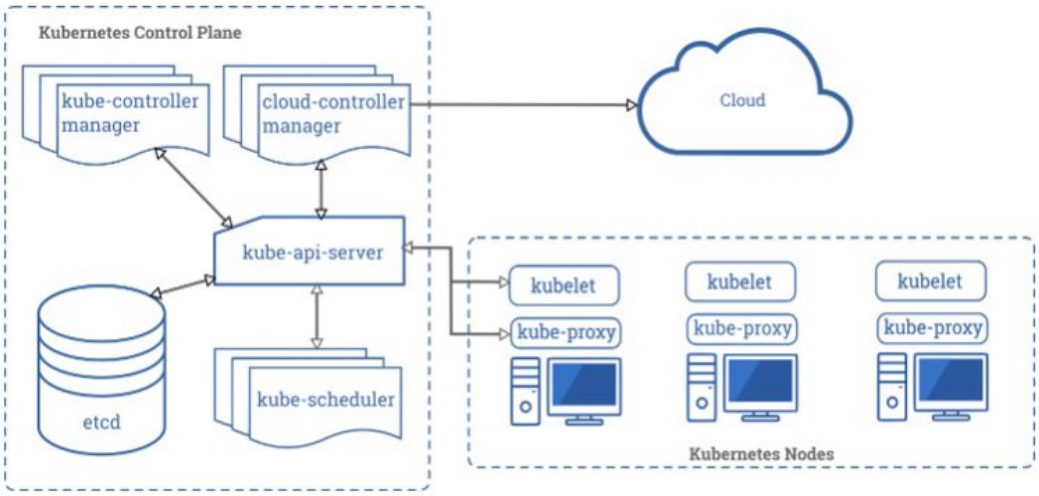
- One or more master nodes
- One or more worker nodes.



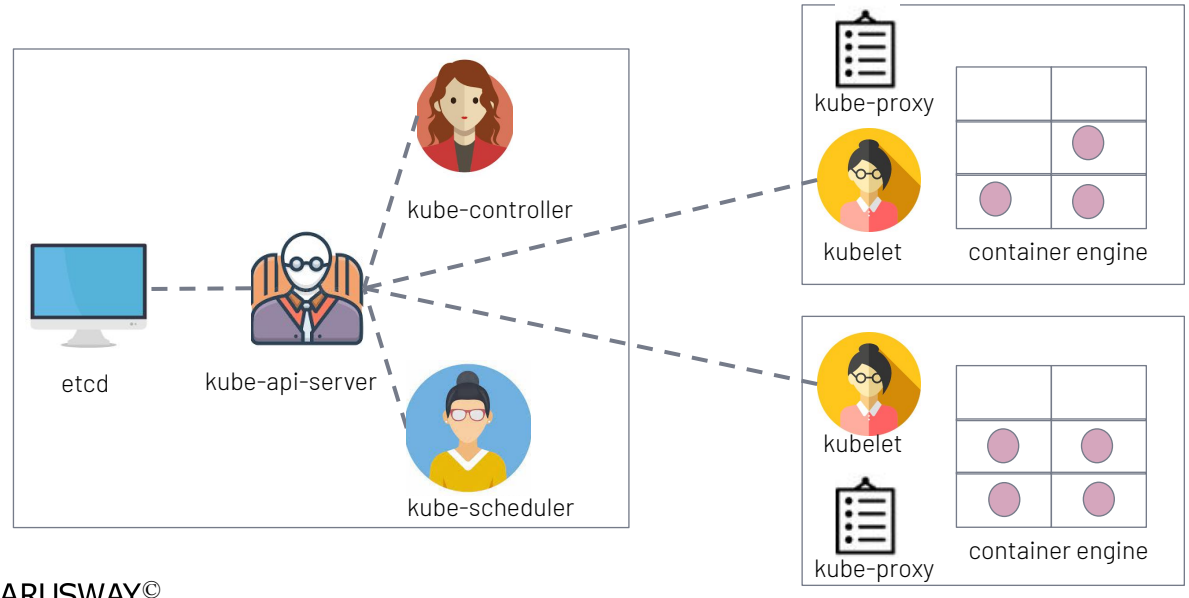
Control Plane Components



Control Plane Components



Control Plane Components



Control Plane Components

kube-apiserver:

- Provides a forward facing REST interface into the kubernetes control plane and datastore.
- All clients and other applications interact with kubernetes strictly through the API Server.
- Acts as the gatekeeper to the cluster by handling authentication and authorization, request validation, mutation, and admission control in addition to being the front-end to the backing datastore.

Control Plane Components

etcd:

- etcd acts as the cluster datastore.
- Purpose in relation to Kubernetes is to provide a strong, consistent and highly available key-value store for persisting cluster state.
- Stores objects and config information.

Control Plane Components

kube-controller-manager:

- Serves as the primary daemon that manages all core component control loops.
- Monitors the cluster state via the apiserver and steers the cluster towards the desired state

Control Plane Components

kube-scheduler:

- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.
- Default scheduler uses bin packing.
- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements.

Node Components

kubelet:

- Acts as the node agent responsible for managing the lifecycle of every pod on its host.
- Kubelet understands YAML container manifests that it can read from several sources:
  - file path
  - HTTP Endpoint
  - etcd watch acting on any changes
  - HTTP Server mode accepting container manifests over a simple API.

Node Components

kube-proxy:

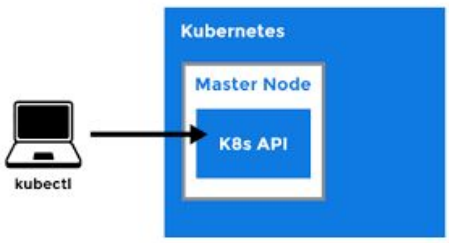
- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.
- Available Proxy Modes:
  - Userspace
  - iptables
  - ipvs (default if supported)

Node Components

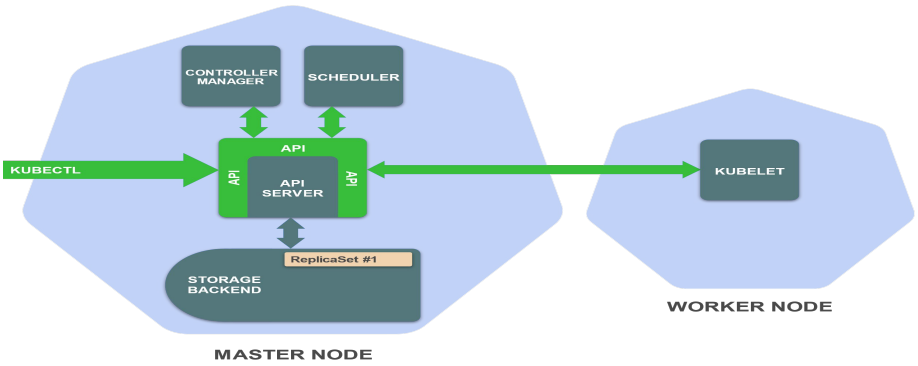
Container Runtime Engine:

- A container runtime is a CRI (Container Runtime Interface) compatible application that executes and manages containers.
  - Containerd (docker)
  - Cri-o
  - Rkt
  - Kata (formerly clear and hyper)
  - Virtlet (VM CRI compatible runtime)

7 kubectl



kubectl



- **kubectl** is (almost) the only tool we'll need to talk to Kubernetes
- It is a rich CLI tool around the Kubernetes API
- Everything you can do with kubectl, you can do directly with the API
- kubectl can be pronounced "Cube C T L", "Cube cuttle", "Cube cuddle".

THANKS!

Any questions?