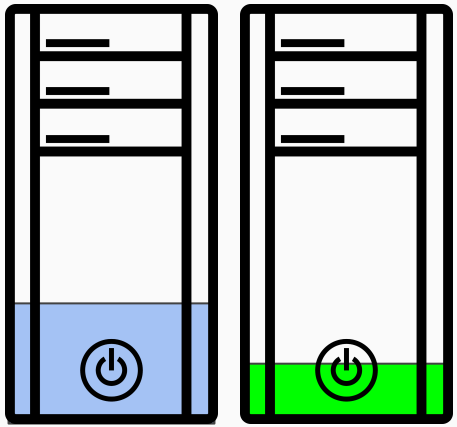
{
“oneRudeDude”: “Lucas Allgood”,
“notRudeDude”: “Baldemar Sepulveda”,
“totalNerd” : “Zachary Downs”,
“somehowStillSingle”: “Tom Bonner”
}

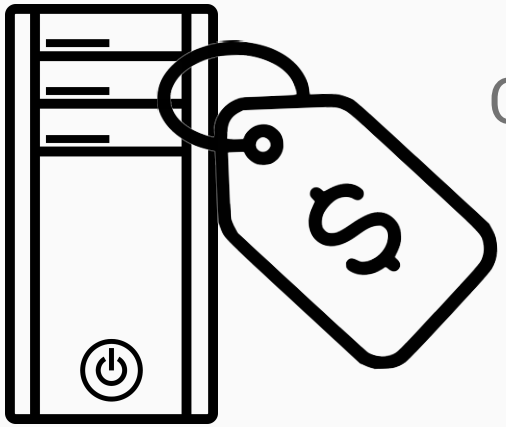
Virtualization

Problem Statement

Independent, self-contained, systems fail utilize system resources effectively.



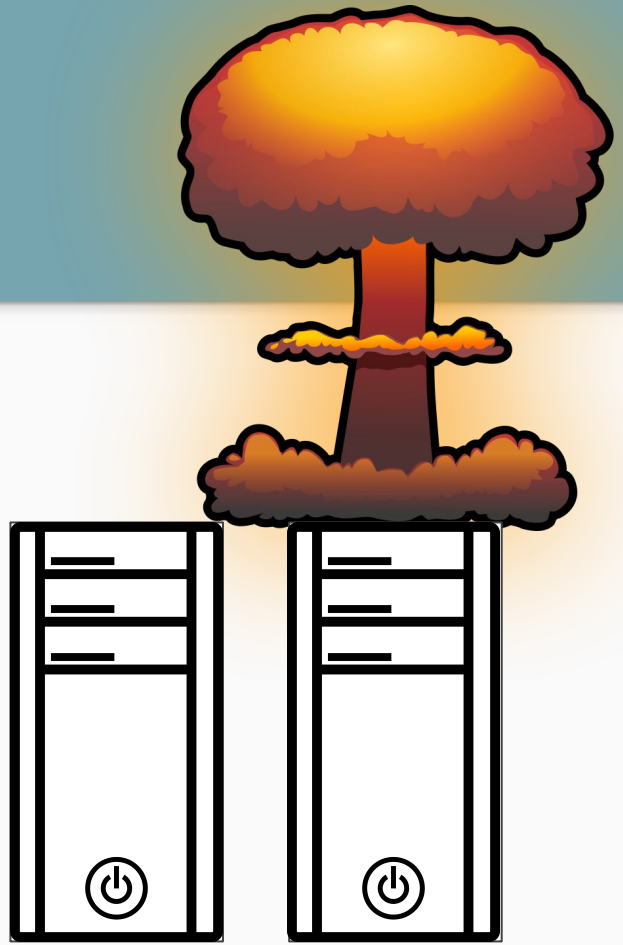
Problem Statement



Computer Hardware is expensive.

Problem Statement

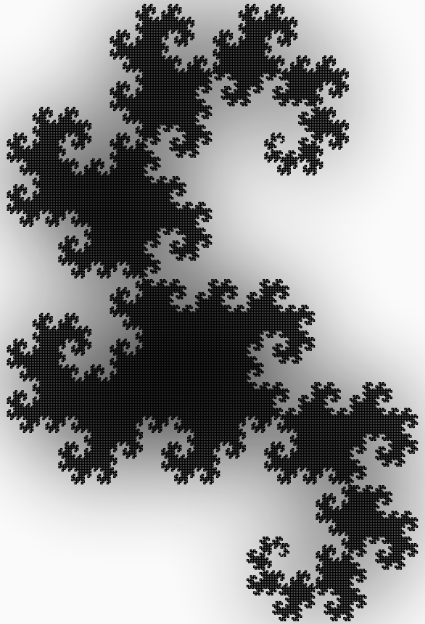
Problems in one application can affect entire system.



Problem Statement

Testing new systems is slow.

Virtualization



The process of creating an abstract system in an existing physical system.

The act of creating a simulated computer inside of a computer.

Solution - Virtualization

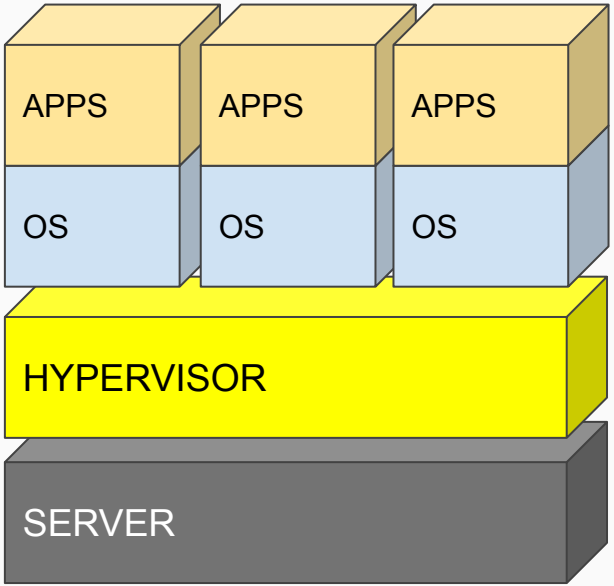
- System resources are allocated as they are needed on a consolidated machine.
- Rather than buying new hardware, just add a new simulated computer inside an existing platform.
- If any individual virtualized system fails, it will not affect other systems on the same platform… usually.
- Great for testing new platforms without having to buy new hardware.

Types of Virtualization

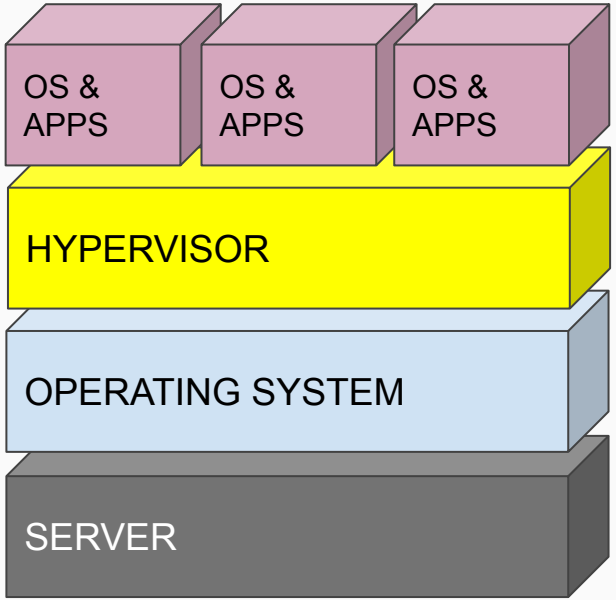
- Server Virtualization (Virtual Desktop Infrastructure or VDI, Type I Hypervisors)
- Hardware Virtualization / Desktop Virtualization (utilizes Type II Hypervisors)
- Application Virtualization (Docker)
- Storage Virtualization
- Network Virtualization

Types of Hypervisors

Type I (Bare metal hypervisor):



Type II (Hosted Hypervisor):



Well known VMs

AWS EC2: OS / Server Virtualization (Type I Hypervisor)

Oracle VirtualBox : Hardware Virtualization (Type II Hypervisor)

Docker, JVM: Application Virtualization

VMWare, HyperV, etc...

Download

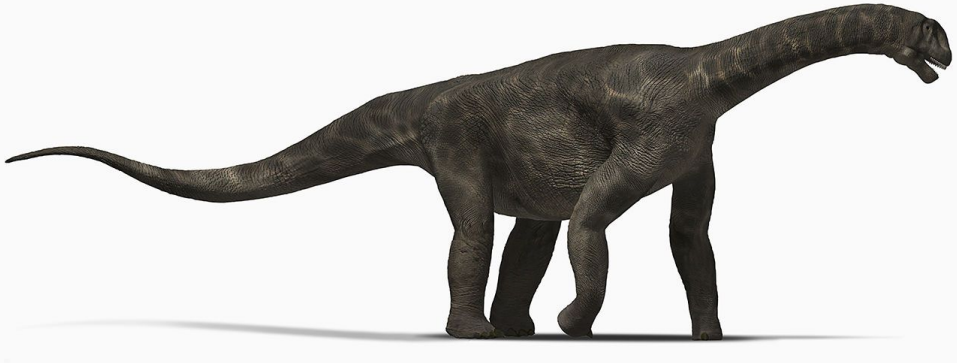
Oracle VirtualBox: https://www.virtualbox.org/wiki/Downloads

Linux Mint: https://linuxmint.com/download.php

Ubuntu: https://ubuntu.com/download/desktop

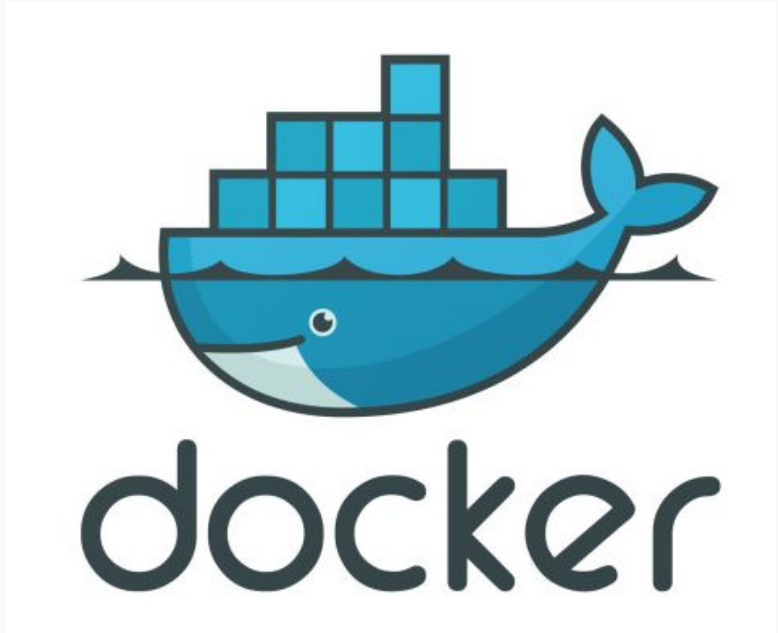
Virtual Machines at a Glance

- Abstraction of Physical Hardware
- Bulky
- Slow to boot



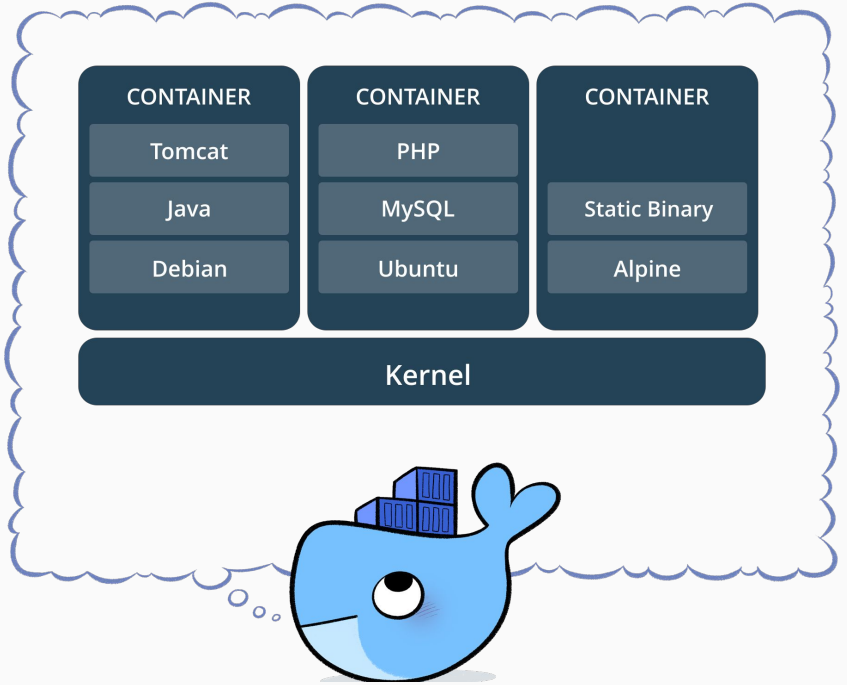
The Solution

- Container Toolkit
- Open Sourced March 2013



Docker at a Glance

- Abstraction at the Application Layer
- Runs on the Host OS kernel
  - do not need specific drivers (except for nVidia CUDA functionality)
- Light Weight

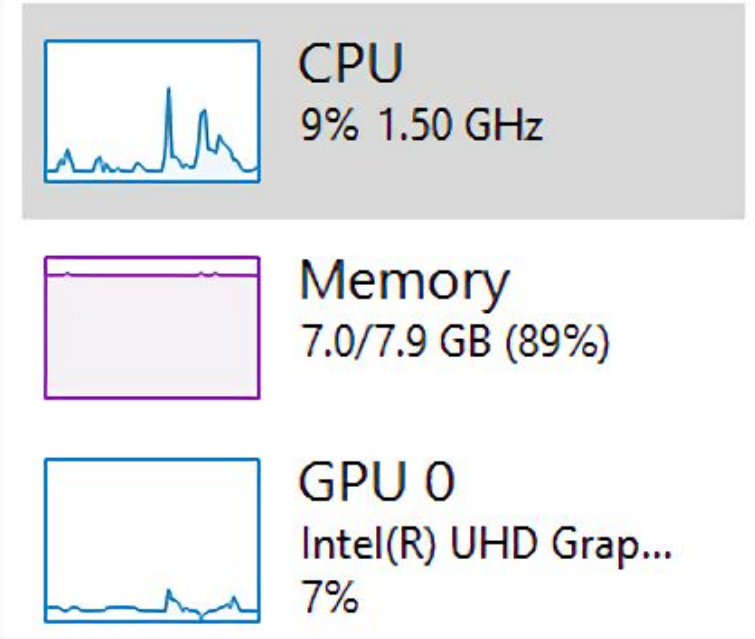


Docker Images

- System Libraries
- Tools
- Files/Dependencies
- Docker CLI

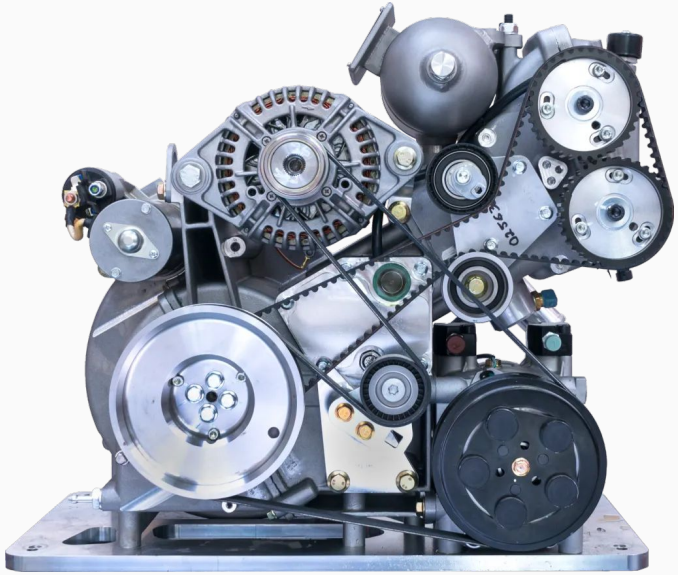
Image Resources

- CPUs/Cores
- Memory
  - Swapping
- GPUs/Cores (nVidia)



Docker Engine

- Containerd
- Supports any type of application
- Works across hybrid/multi-cloud
- Implements Kubernetes CRI



Security

- FIPS 140-2 Encryption
  - Government Ready
- Docker Content Trust
  - Digital Signing

Docker Hub

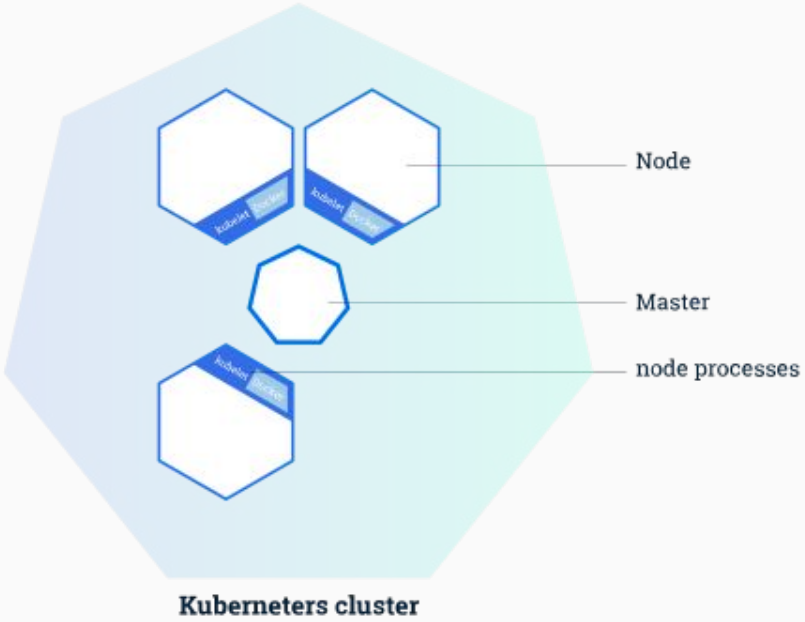
- Offered by Docker Inc.
- Cloud-based registry service

Kubernetes

Background

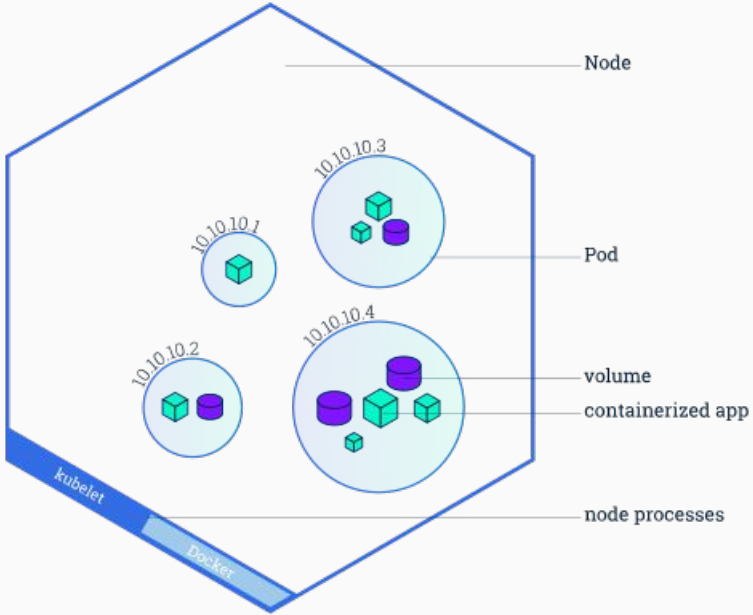
- Kubernetes Documentation: https://kubernetes.io/docs/home/
- Released by Google as an open source project in 2014
- Created to solve the problem of container orchestration

Concepts



- Cluster - A collection of nodes and a master node
- Node - A worker machine that runs your containers
- Master Node - A node that manages the cluster

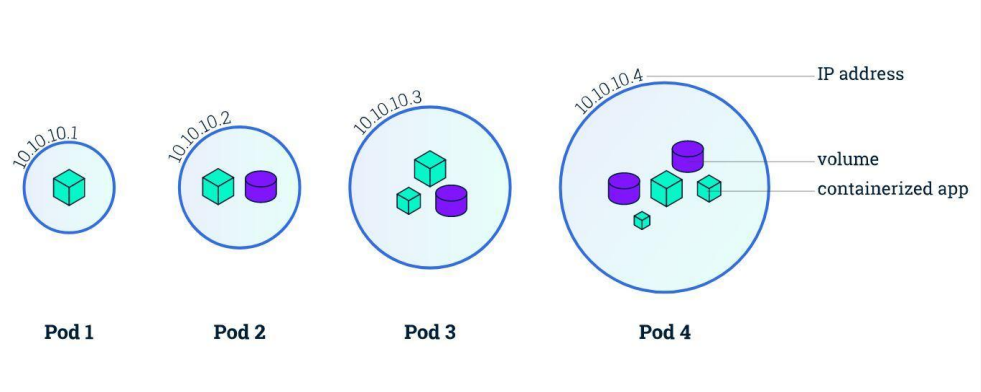
Concepts



- Pod
- Service
- Volume
- Namespace
- Deployment
- ReplicaSet

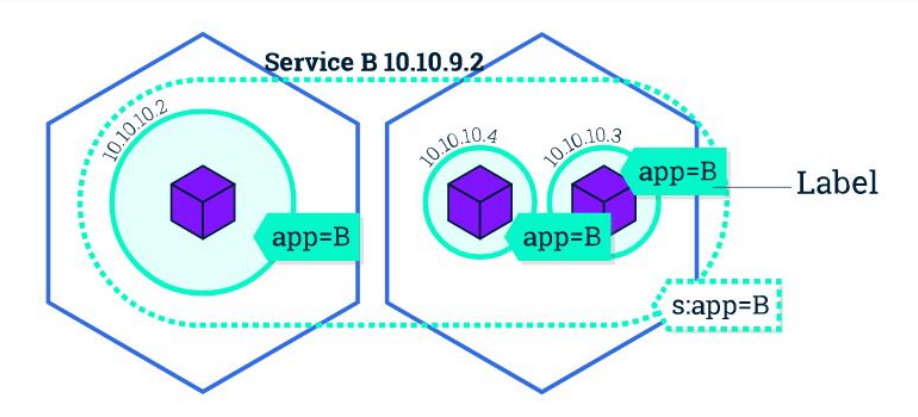
Pods

- The smallest deployable unit of Kubernetes
- Contains a container or a group of tightly coupled containers

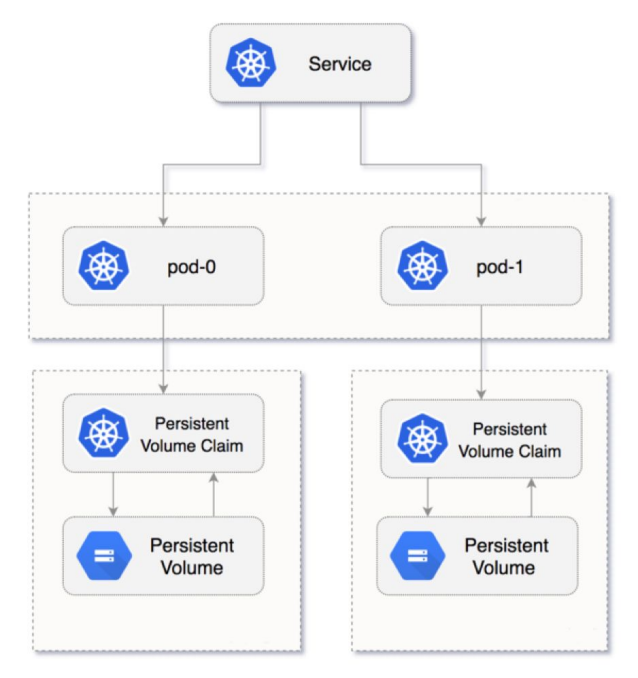


Services

- An abstraction that defines how to access a set of pods
- Allows for communication between pods and outside the cluster

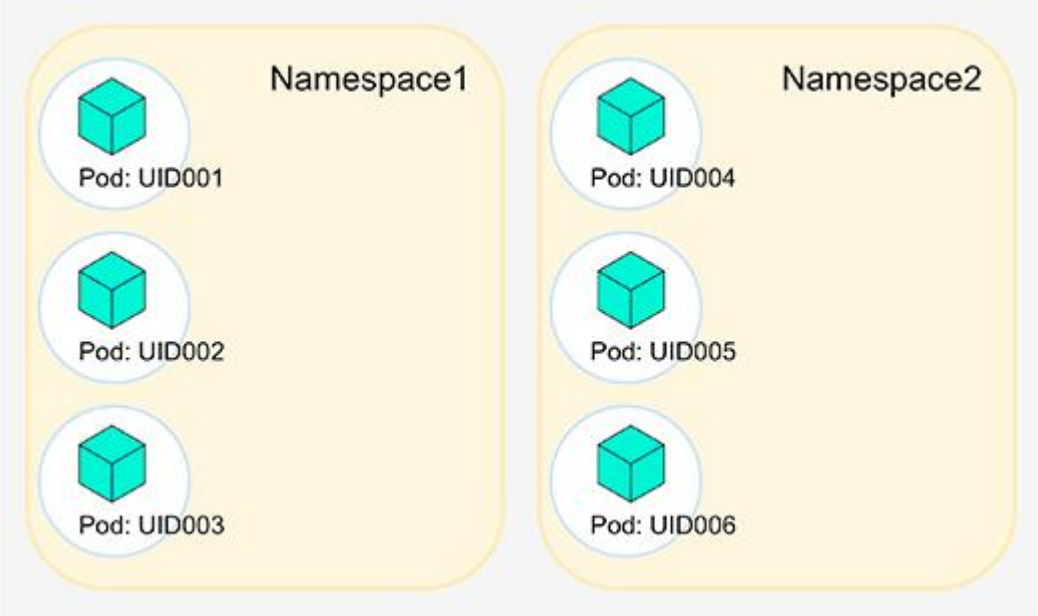


Volumes



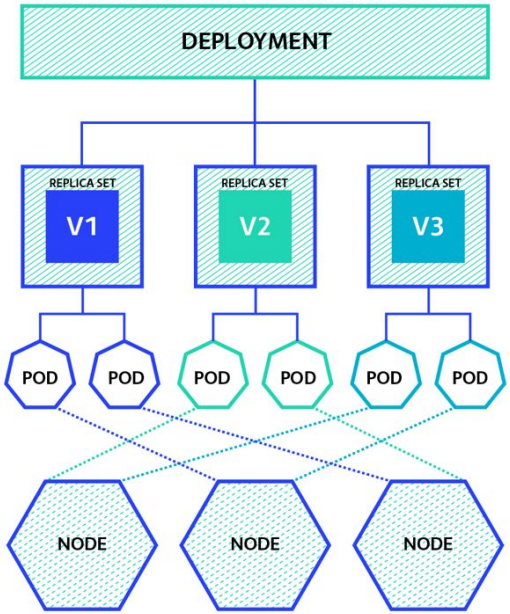
- Allow pods to have persistent storage
- Will be either static or dynamic
- Can be a variety of storage types (EBS)
- Usually provisioned with a persistentVolumeClaim

Namespaces



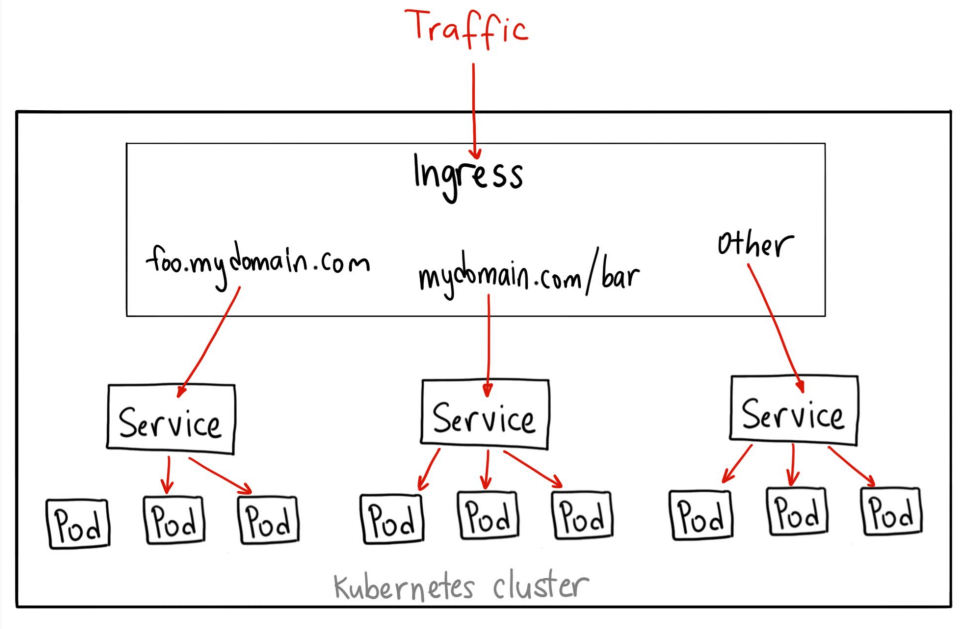
- Allows for the separation of projects for easier cluster management
- Intended for clusters spread across users and teams

Deployment / ReplicaSet



- Deployments specify how kubernetes runs a pod
- They allow you to create ReplicaSets which makes the scaling of pods incredibly simple

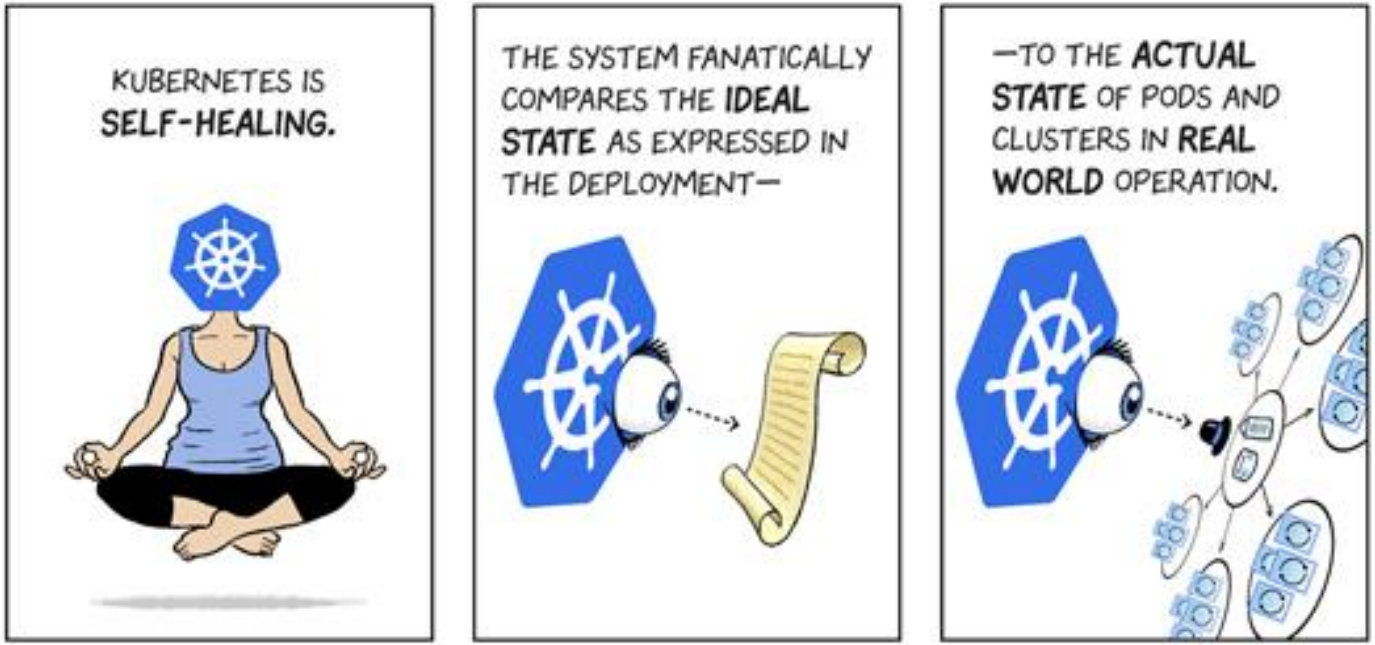
Ingress



- Exposes nodes outside the cluster
- Can map url paths to nodes
- Allows for url rewriting
- CORS configuration

Ingress

```
apiVersion: networking.k8s.io/v1beta1
kind: Ingress
metadata:
  name: test-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        backend:
          serviceName: test
          servicePort: 80
```


KUBERNETES IS
SELF-HEALING.
THE SYSTEM FANATICALLY COMPARES THE IDEAL STATE AS EXPRESSED IN THE DEPLOYMENT—
—TO THE ACTUAL STATE OF PODS AND CLUSTERS IN REAL WORLD OPERATION.

Pod Management

- Management of your pods is done by the master node
  - kube-apiserver
  - kube-controller-manager
  - Kube-scheduler
- The cluster state is stored in etcd, a highly available key store
- Kubernetes supports rolling updates to prevent down time

Pod Health

- Each node has an agent called kubelet
- Kubelet verifies that pods containers are running and healthy
- It can restart your pod based on the set restartPolicy
- Different ways of determining pod health (HTTP)
- HTTP responses outside of 200-300 are considered unhealthy

Kube-DNS

- Handles resolution of service names to their cluster ip address
- Resolution only possible from within the cluster
- Client side front ends need server side proxies for DNS resolution

proxy.conf.json 106 Bytes

```
{
    "/animal": {
        "target": "http://animals-spring-service:9000",
        "secure": false
    }
}
```

References

- Kubernetes Documentation
  - https://kubernetes.io/docs/home/
- Kubernetes Comic
  - https://cloud.google.com/kubernetes-engine/kubernetes-comic/
- Hello World Image
  - https://github.com/paulbouwer/hello-kubernetes
- Angular Proxies
  - https://github.com/angular/angular-cli/blob/master/docs/documentation/stories/proxy.md

Fin