

Docker Container Orchestration with Kubernetes



Who am i

Souri Abdelhalim

Security Engineer . His interests include secdevops and studies of how to apply machine learning to detect eventual security flaws.

twitter : https://twitter.com/nordo_9

github: <https://github.com/etadata>



Lab overview

- What is a container and why you may want one
- VM vs Container
- How to run and manage containers
- How to create your own containers
- How to share your containers
- Useful Commandes



What is a Container...

an isolated and secure application platform.

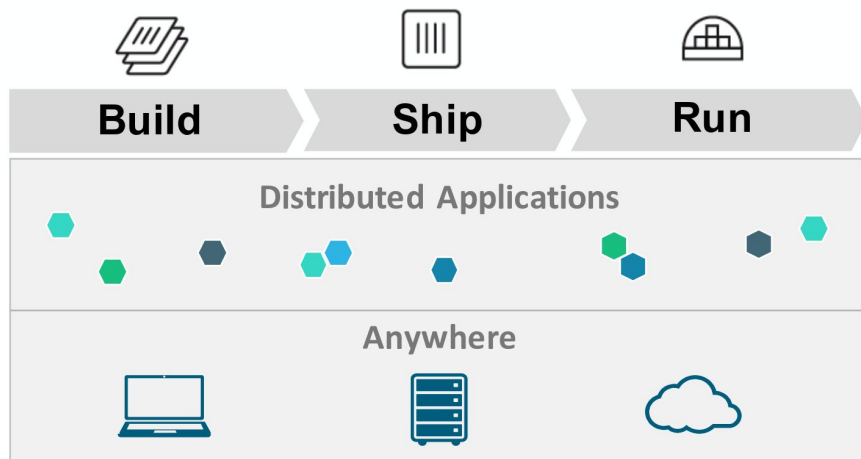
- run, started, stopped, moved, and deleted.
- created from a Docker image.



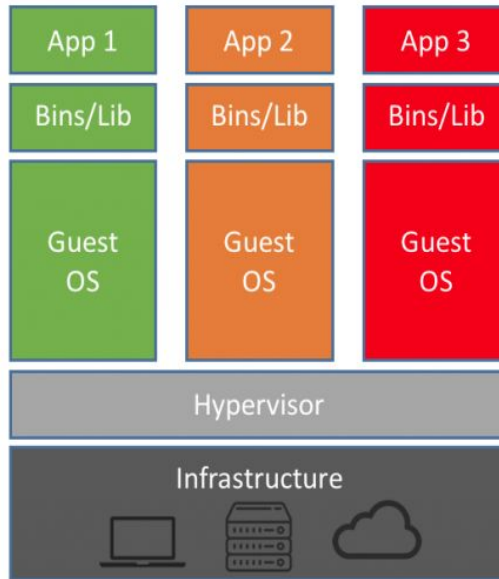
- It provides **isolation**, so applications on the same host and stack can avoid dependency conflict.
- It is **portable**, so you can be sure to have exactly the same dependencies at runtime during development, testing and in production.

Container, mission

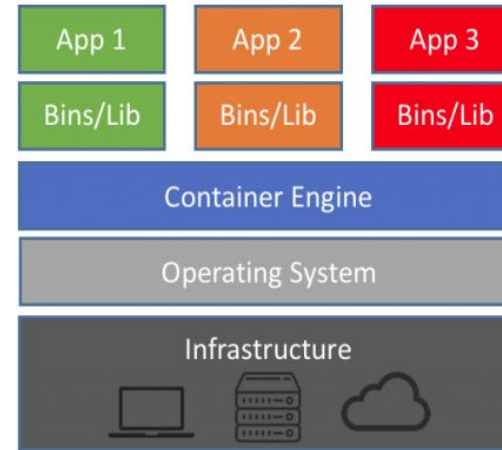
Docker Mission



VM vs Container



Machine Virtualization



Containers

So, why Container ?

- Containers are more lightweight and faster.
- No need to install guest OS.
- Less CPU, RAM, storage space required.
- More containers per machine than VMs.
- Greater portability.
- Containers are easy to manage as they share a common OS.
- Share multiple workloads on a single OS.
- Containers are a better way to develop and deploy microservices compared with VMs.



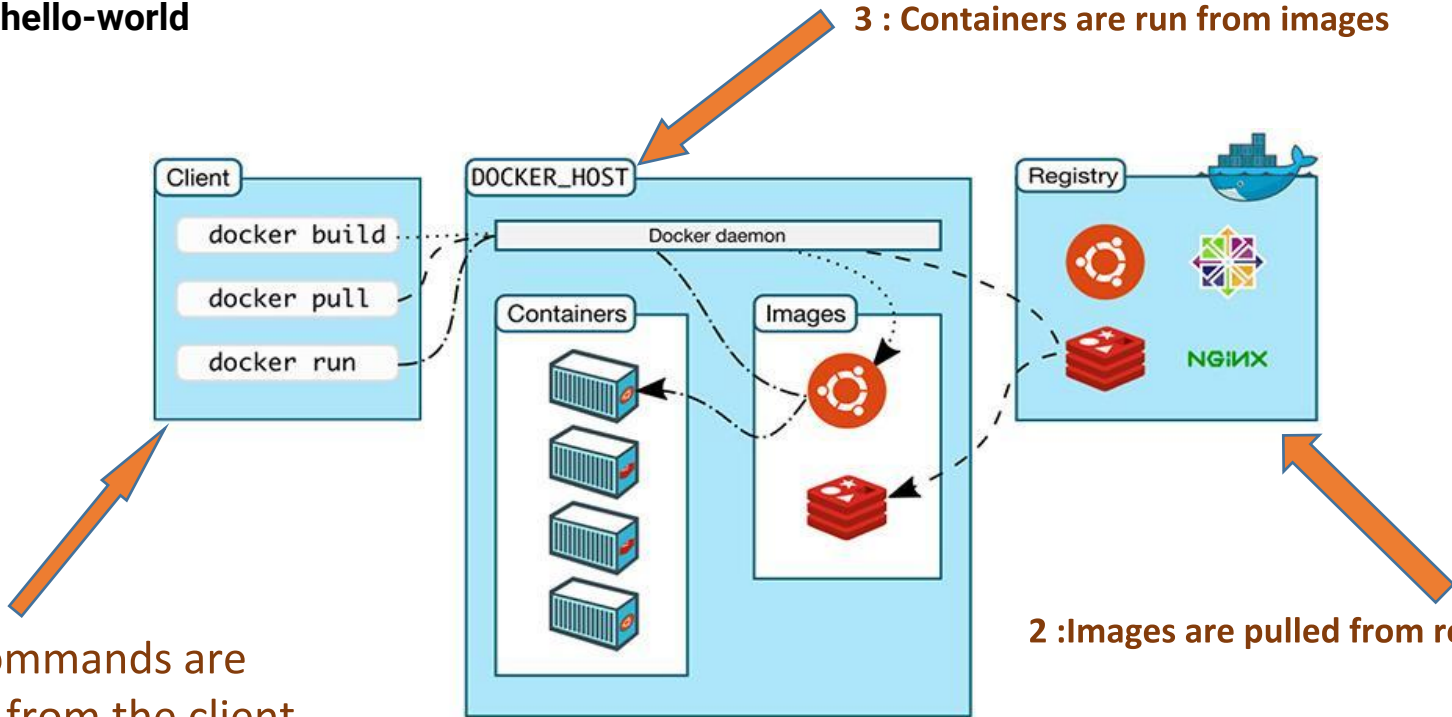
Let's create a container

\$ docker run hello-world

1 : The commands are executed from the client

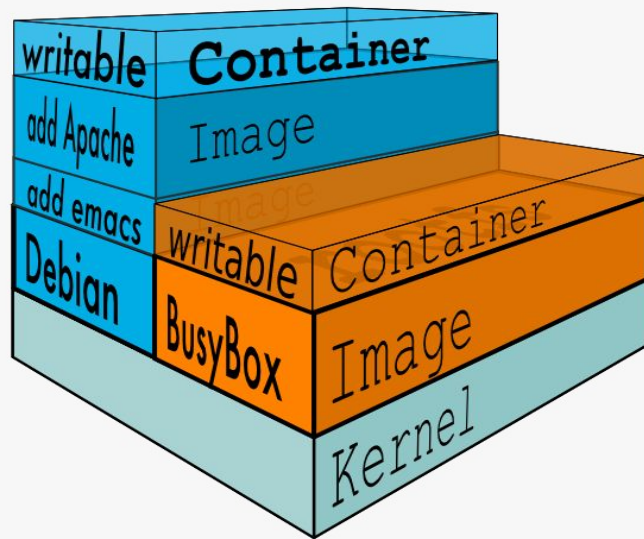
2 : Images are pulled from repositories

3 : Containers are run from images



Container, image ?

- A read-only template for creating containers.
- The **build** component of docker.
- Stored in registries.
- Can be created by yourself and distributed by others.



\$ docker run hello-world → When an image is run, a writable layer is added

Container, saving changes...

1. Docker commit

saves changes in a container
as a new image :

```
$ docker commit 234d3ea32  
mynewimage:version1
```

to run this new image :

```
$ docker run -it  
mynewimage:version1 bash
```

2. DockerFile

Run instructions are executed in the
top writable layer

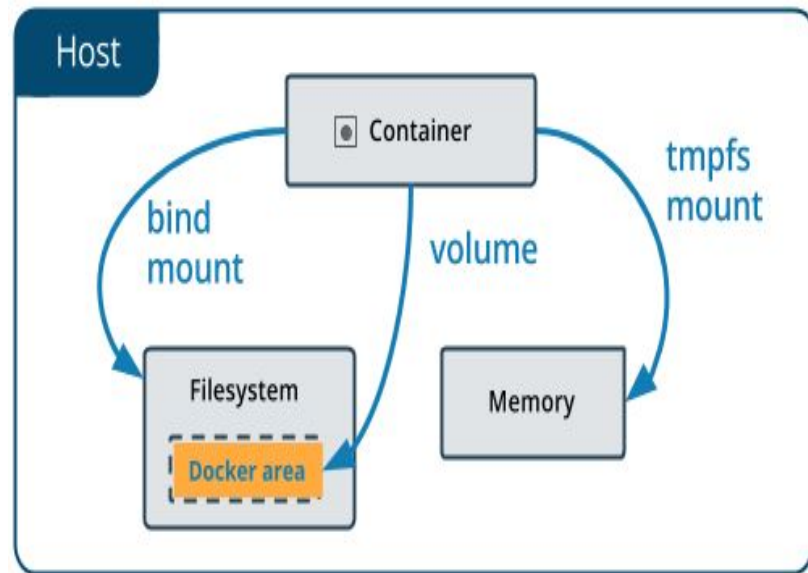
```
1 # Version: 0.0.1  
2 FROM ubuntu:16.04  
3 MAINTAINER Souri Abdelhalim "abdelhalimsouri@gmail.com"  
4 RUN apt-get update  
5 RUN apt-get install -y nginx  
6 RUN echo 'Welcome to Owasp workshop 2018, from container' > /var/www/html/index.nginx-debian.html  
7 EXPOSE 80
```

to build an image from a Dockerfile

```
$ docker build -t simple:1.1 path-to-Dockerfile
```

Docker volume

- Can be created in Dockerfile or via CLI.
- Can be used to share(or/and persist) data between host and containers.
- Volumes mount a directory on the host into the container at a specific location.



Docker volume, commandes

\$ docker volume create

\$ docker volume ls

\$ docker volume inspect

\$ docker volume rm



Docker network

Docker containers communicate with each other and the outside world via the host machine.

Docker supports different types of networks, each fits for certain use cases.

- Bridge
- Overlay
- Macvlan
- Host
- none



Docker network: bridge

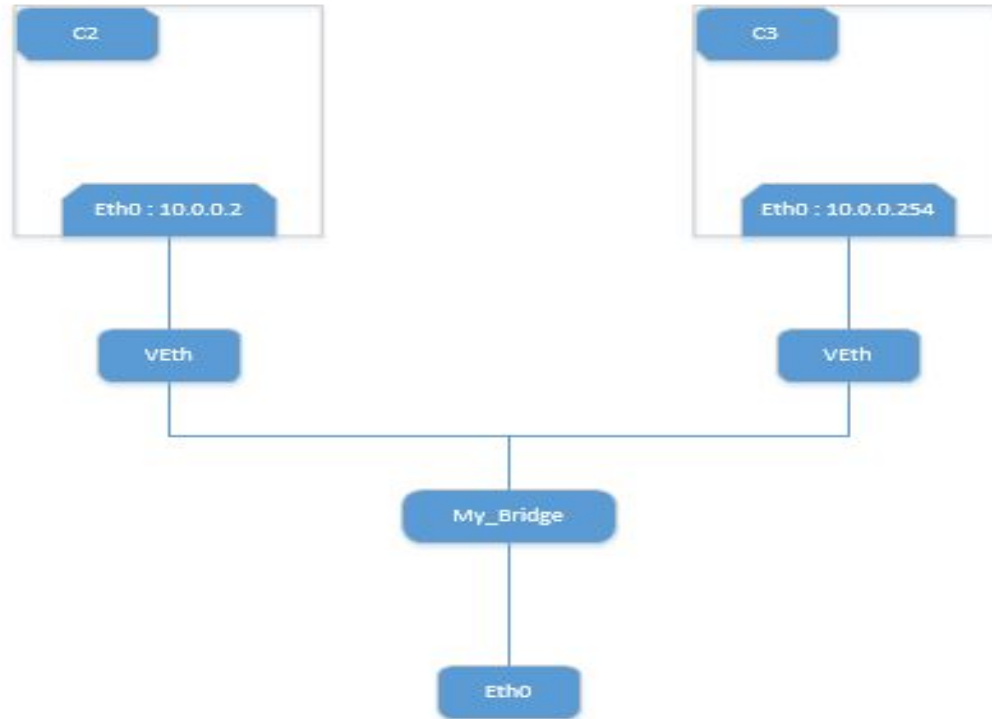
Bridge networking is the most common network type. It is limited to containers within a single host running the Docker engine.

Bridge networks are easy to :

- Create
- Manage
- Troubleshoot.



Our lab Schema



Some GUI Container managers : Portainer

The screenshot displays the Portainer web interface. On the left is a dark blue sidebar with the 'N+ONE DATACENTERS' logo and a menu including Home, LOCAL, Dashboard, App Templates, Stacks, Containers, Images, Networks, Volumes, Events, Engine, SETTINGS, Endpoints, Registries, and Settings. The main area is titled 'Dashboard Endpoint summary' and includes a 'Portainer support' link. Below the title is a table with 'Endpoint info' containing details for the 'local' endpoint: 4 CPUs, 8 GB memory, Standalone 18.03.1-ce, URL /var/run/docker.sock, and no tags. The dashboard also features five summary cards: 0 Stacks, 4 Containers (2 running, 1 stopped), 25 Images (9 GB), 43 Volumes, and 8 Networks.

N+ONE DATACENTERS

Home

LOCAL

Dashboard

App Templates

Stacks

Containers

Images

Networks

Volumes

Events

Engine

SETTINGS

Endpoints

Registries

Settings

Dashboard
Endpoint summary

Portainer support

Endpoint info

Endpoint	local 4 8 GB - Standalone 18.03.1-ce
URL	/var/run/docker.sock
Tags	-

0
Stacks

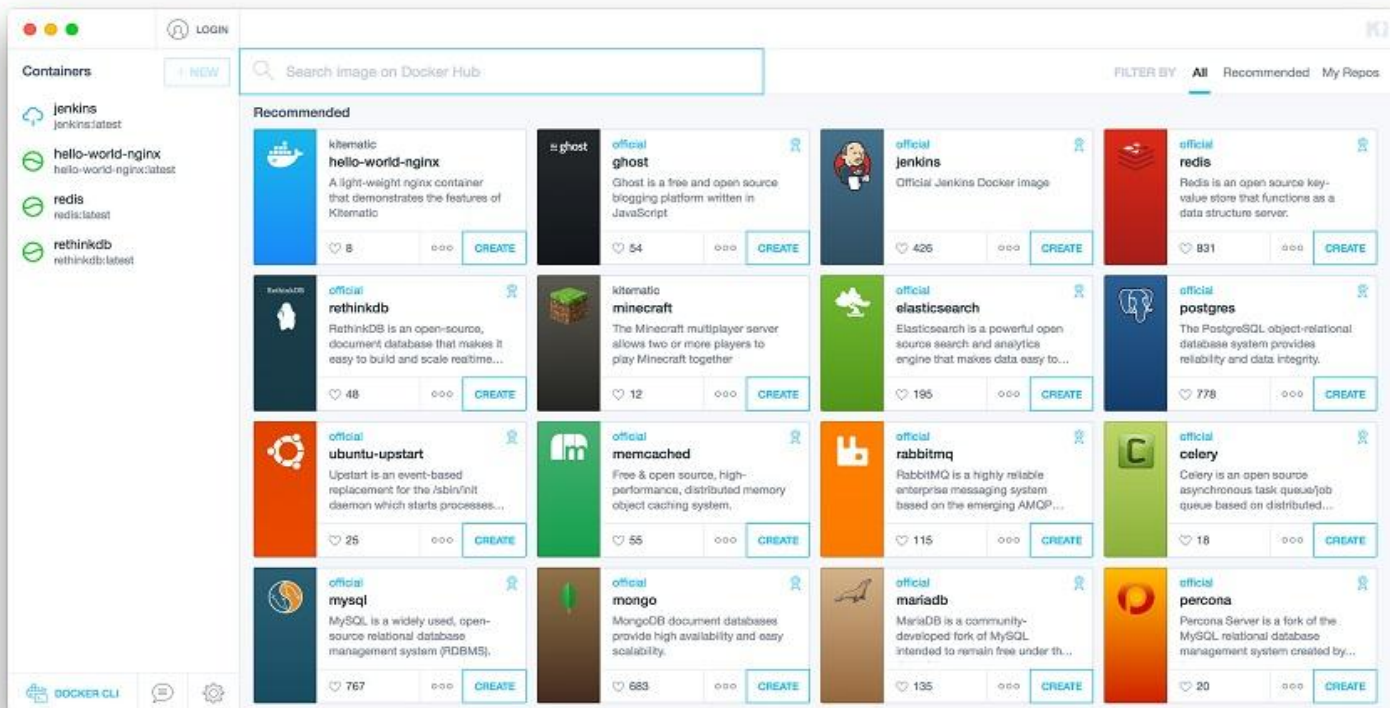
4
Containers
2 running
1 stopped

25
Images
9 GB

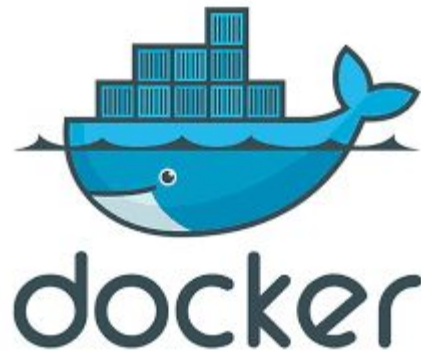
43
Volumes

8
Networks

Some GUI Container managers : Kitmatic



Sharing Containers : DockerHub



```
$ docker push abdelhalim/static_web:v1.0
```

DockerHub, what it provides

- Library of public images
- Storage for your images
- free for public images
- cost for private images
- Automated builds(link github/bitbucket repo; trigger build on commit)



useful commandes

- `docker build` # Build an image from a Dockerfile
- `docker images` # List all images on a Docker host
- `docker run` # Run an image
- `docker ps` # List all running and stopped instances
- `docker stop` # Stop a running instances
- `docker rm` # Remove an container
- `docker rmi` # Remove an image





End of the first part



kubernetes



Content

- Kubernetes definition
- Core Concepts of kubernetes
- Kubernetes Objects
- The detail of each object



What is Kubernetes ?

Kubernetes is a platform for hosting Docker containers in a clustered environment with multiple Docker hosts

It provides :

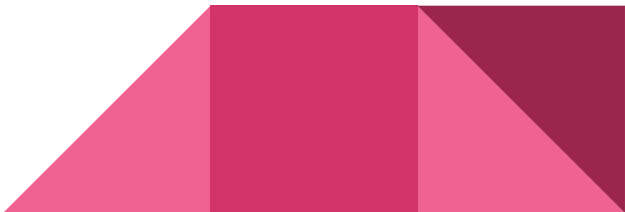
- container grouping, **automating deployment**
- loadbalancing,
- auto-healing,
- scaling

Project was started by Google

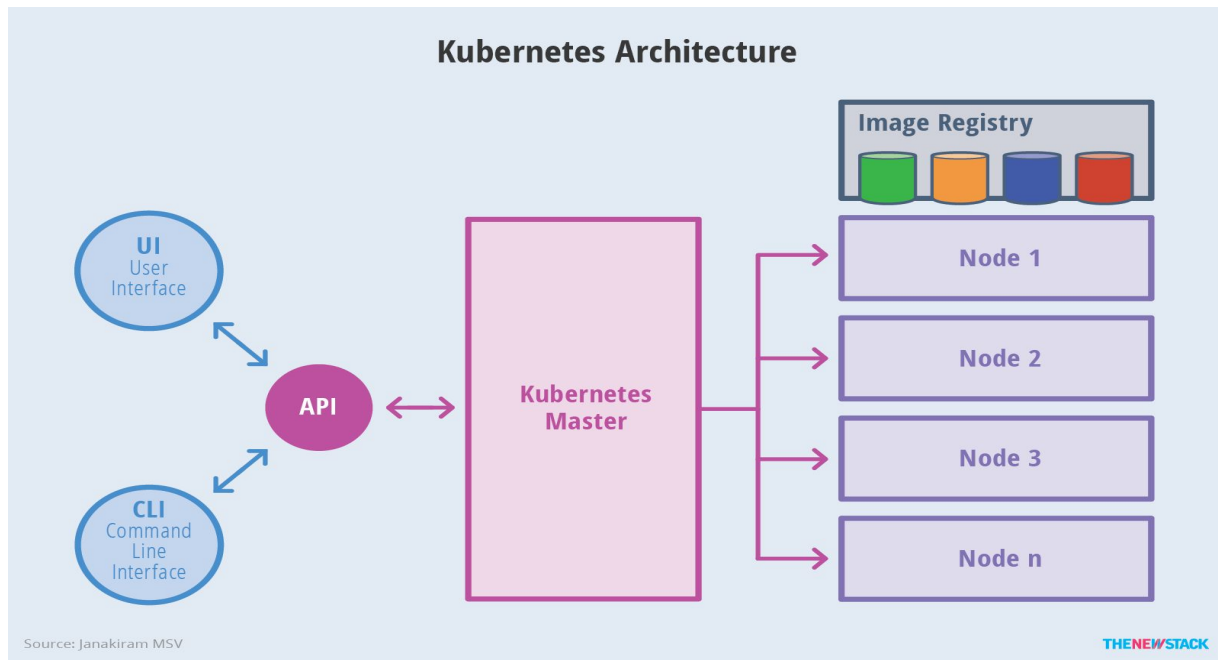
Contributors == Google, CodeOS, Redhat, Mesosphere, Microsoft, HP, IBM, VMWare, Pivotal, SaltStack, etc



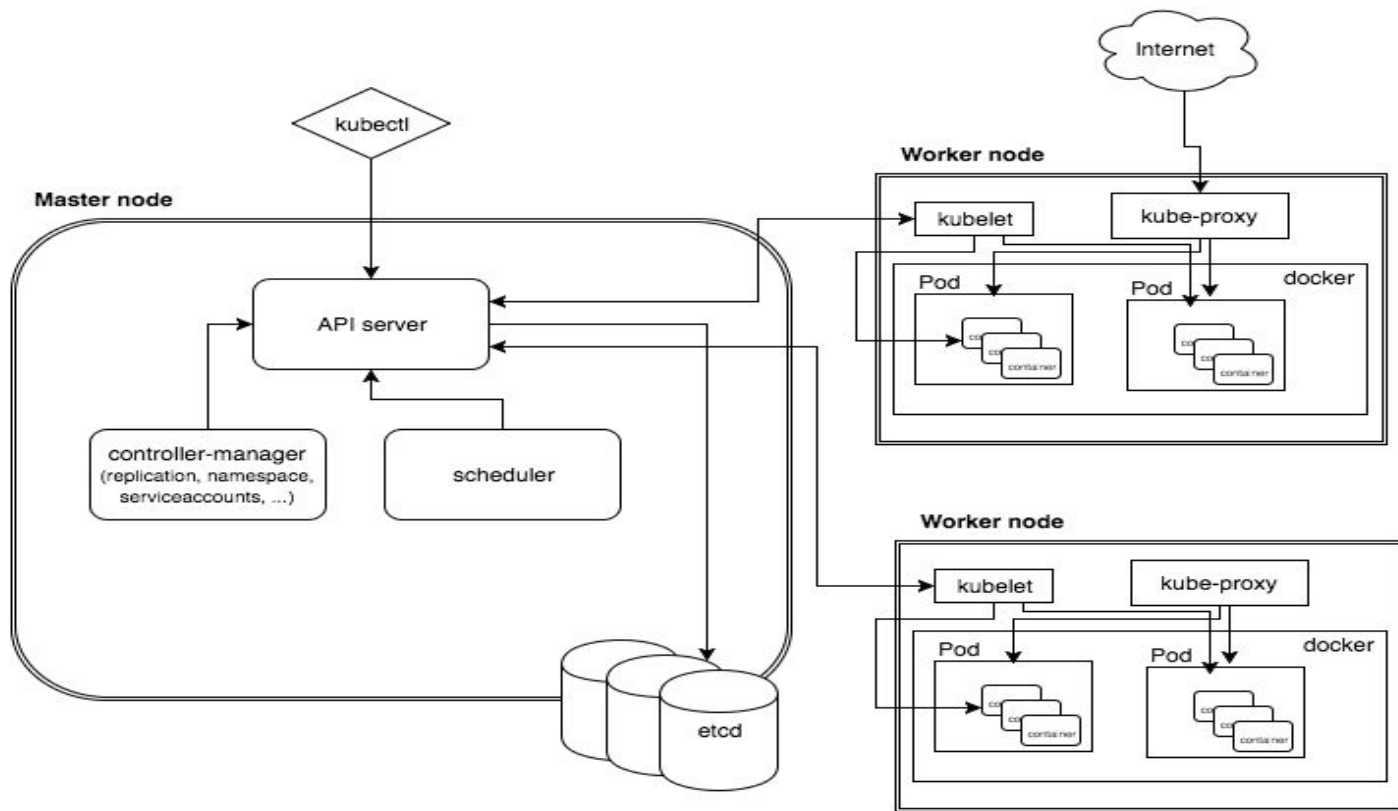
Kubernetes Objects

- **Pod** : A group of Containers
 - **Replication Controller/Replicaset** : Manages replication of pods
 - **Deployment** : A Deployment controller provides declarative update for **Pods** and **ReplicaSets**
 - **Service** : to expose your service
 - **Volume** : as a directory which is accessible to the containers in a pod.
 - **Labels** : Labels for identifying pods
 - **Kubelet** : Container Agent
 - **Proxy** : A load balancer for Pods
 - **Scheduler** : Schedules pods in worker nodes
 - **API Server** : Kubernetes API server
- 

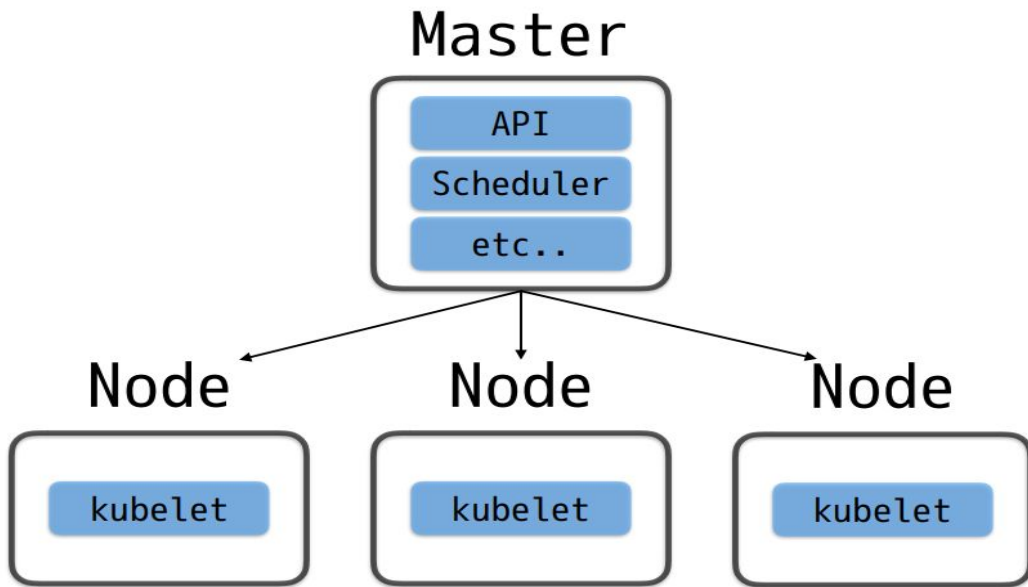
Kubernetes Architecture



Kubernetes Architecture



Basic concept : Node

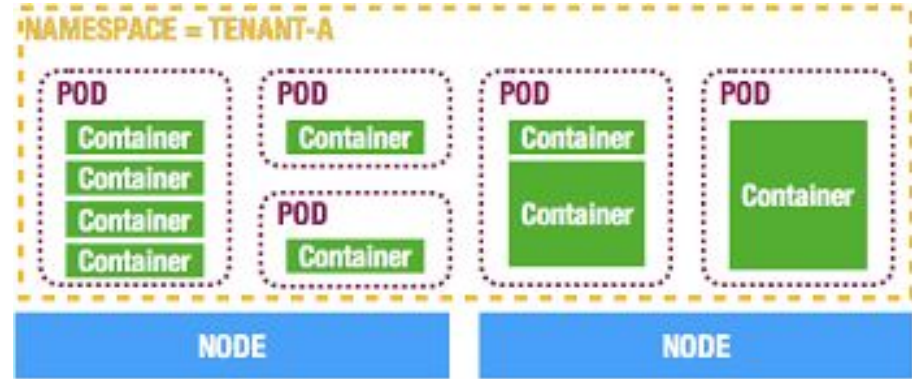


Nodes : Hosts running k8s daemons

Basic concept : POD

POD : Group of containers

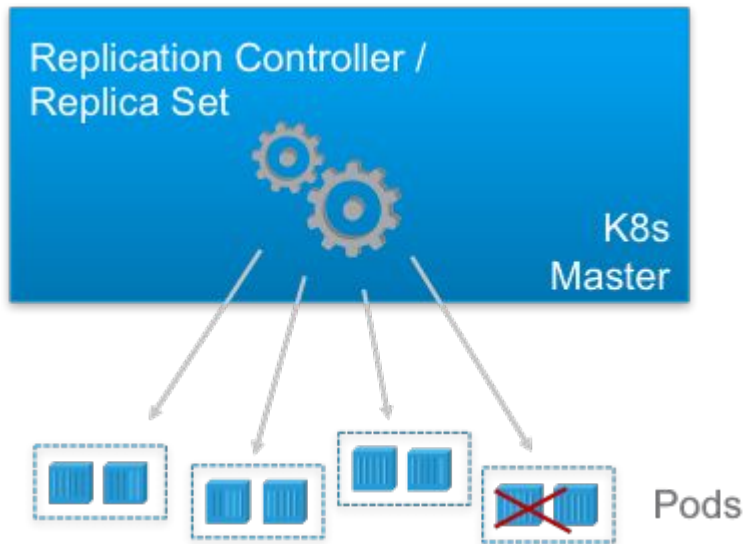
- Each pod has its own IP address
- Guaranteed to be on the same node
- Shared storage



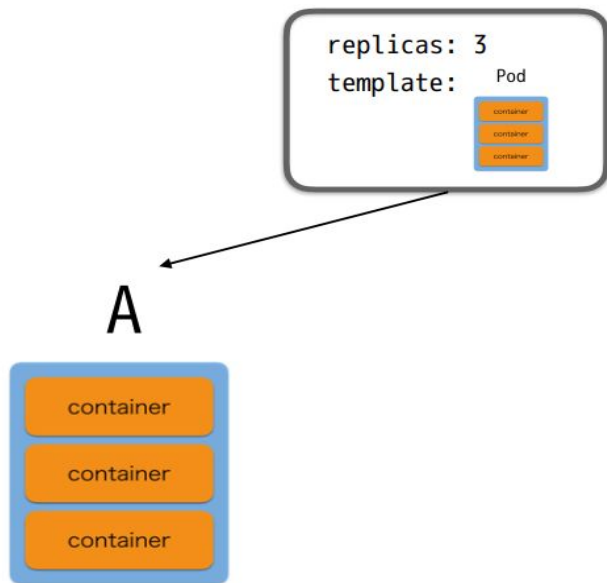
Basic concept : replication controller/replica Set

Have one job : ensure N copies of a pod

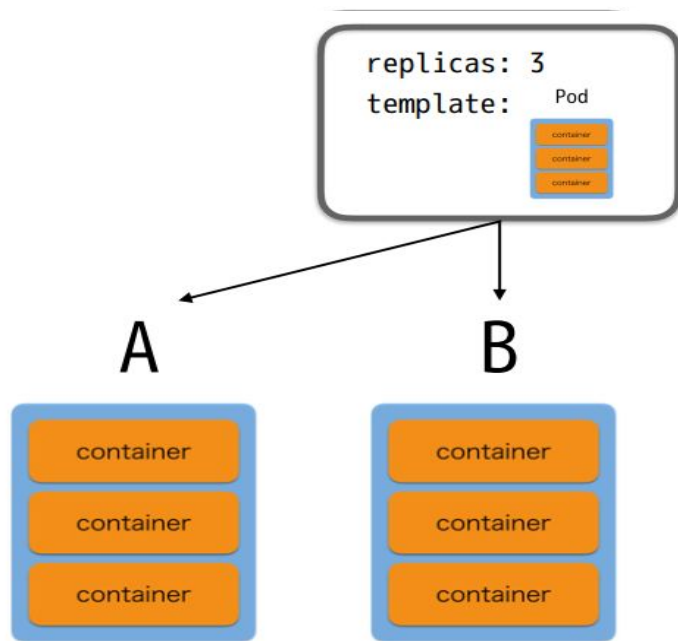
- Replicated pods are fungible
- Keeps track of Pod replicas



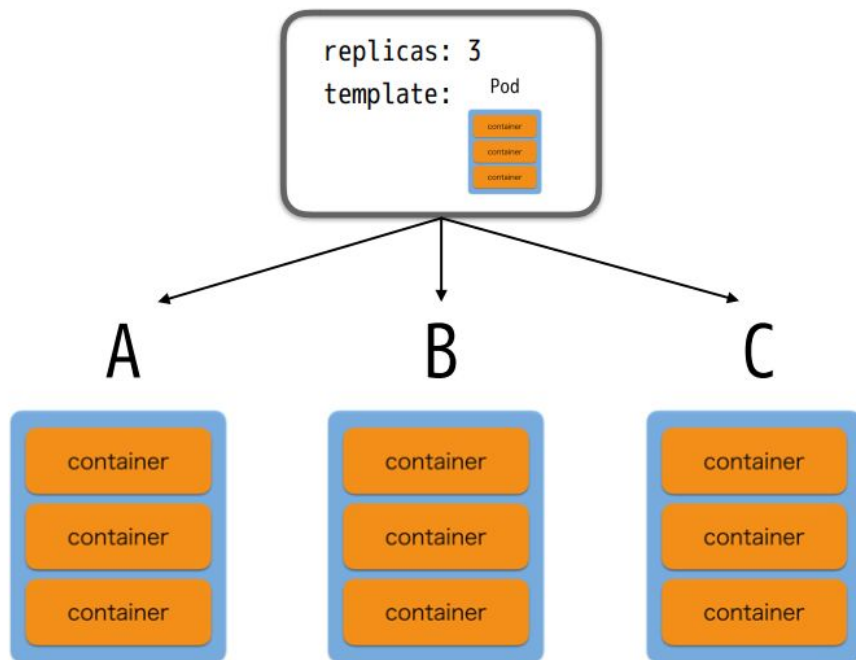
Basic Concepts : Replication Controller / Replica Set...



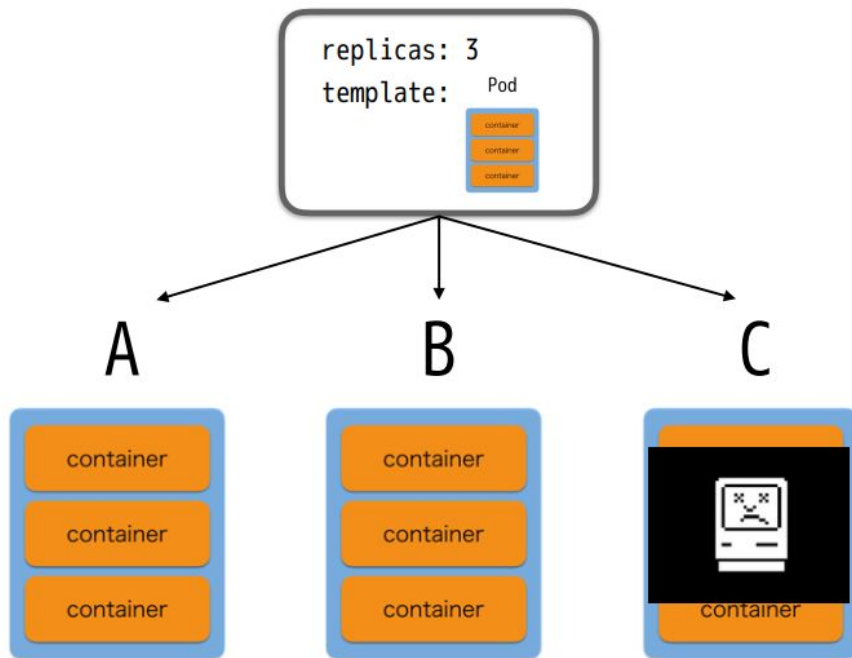
Basic Concepts : Replication Controller / Replica Set...



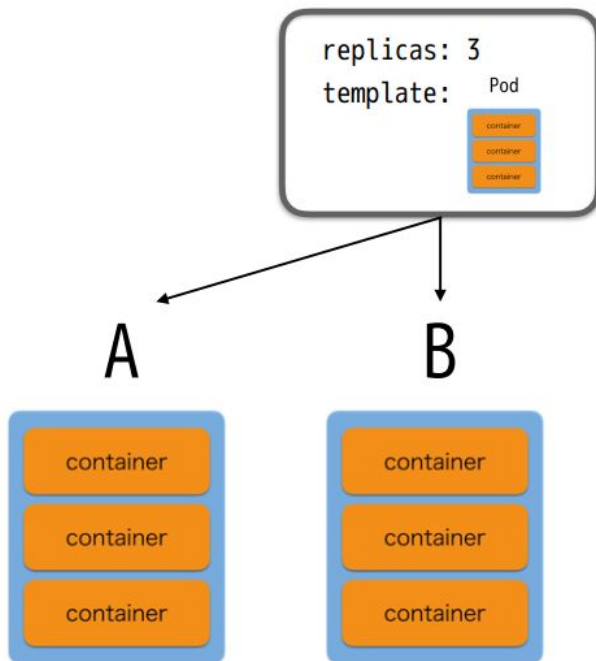
Basic Concepts : Replication Controller / Replica Set...



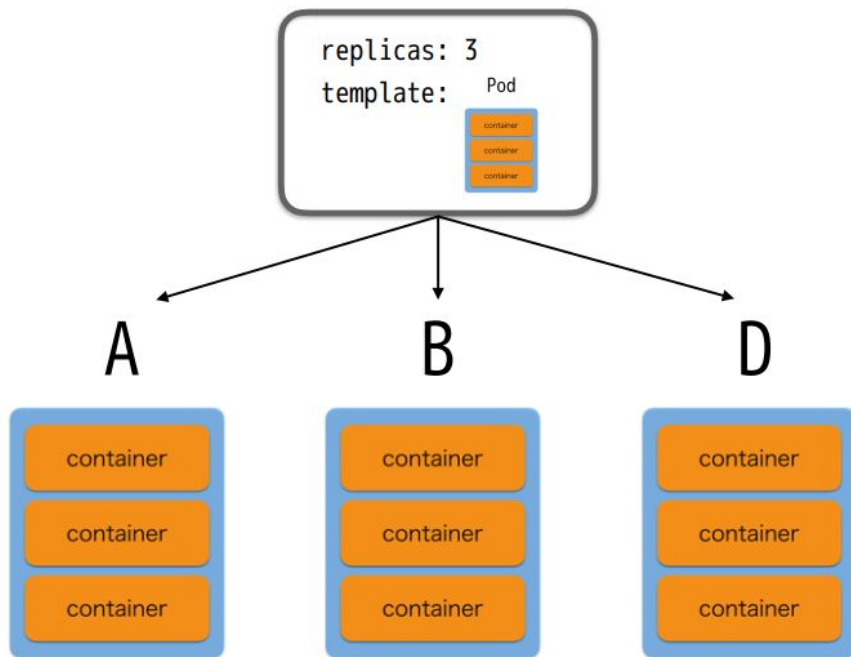
Basic Concepts : Replication Controller / Replica Set...



Basic Concepts : Replication Controller / Replica Set...

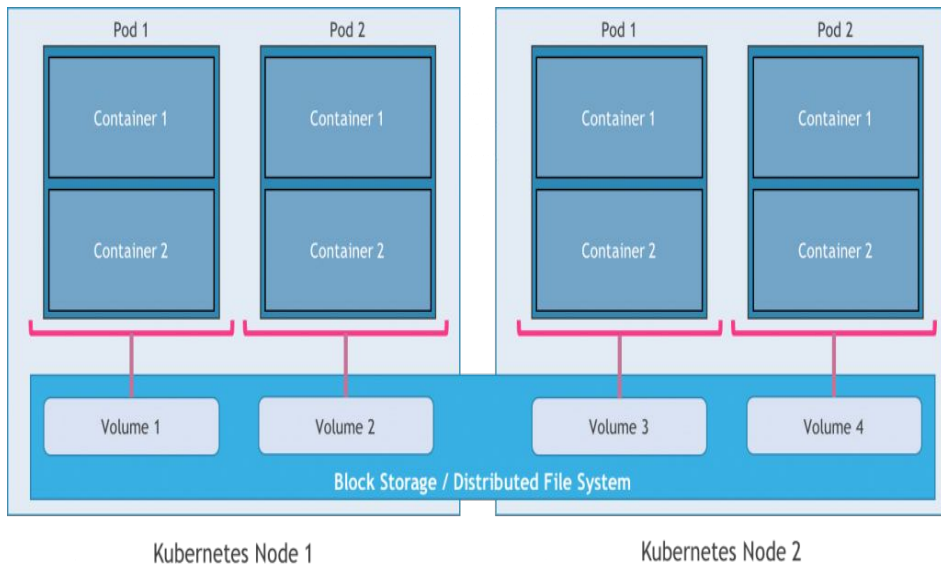


Basic Concepts : Replication Controller / Replica Set...



Basic Concepts : VOLUME

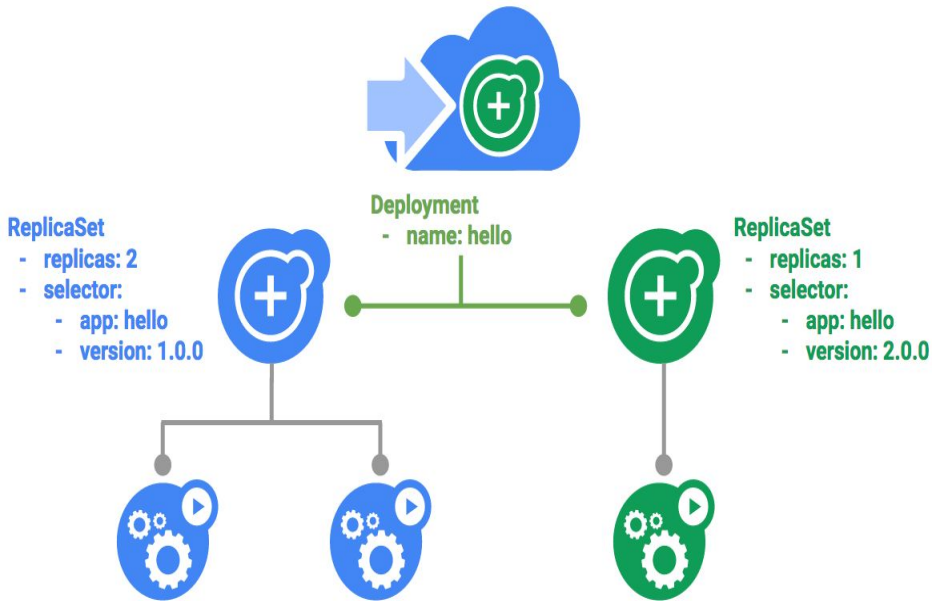
Similar to a container volume in Docker, but a Kubernetes volume applies to a whole pod and is mounted on all containers in the pod.



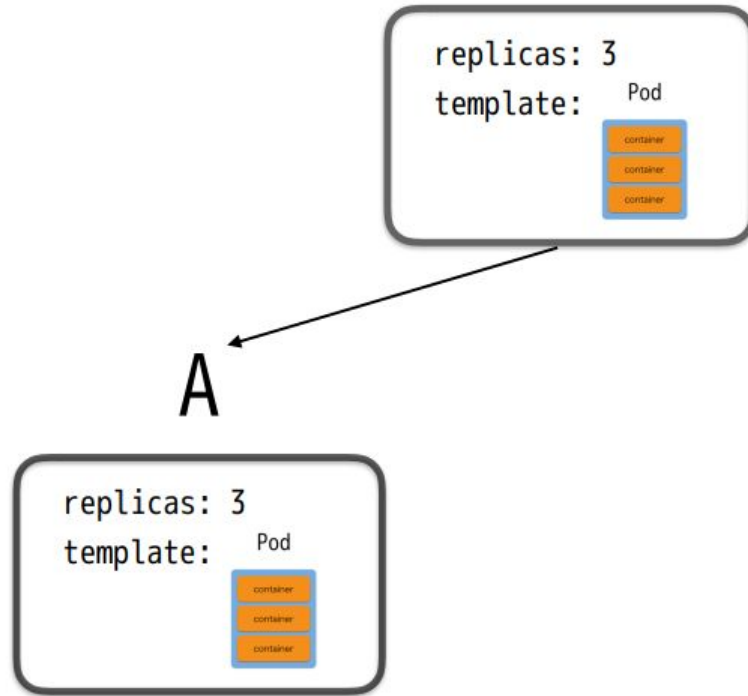
Basic Concepts : Deployment

Reliable mechanism for creating, updating and managing Pods

Deployment manages replica changes, including rolling updates and scaling

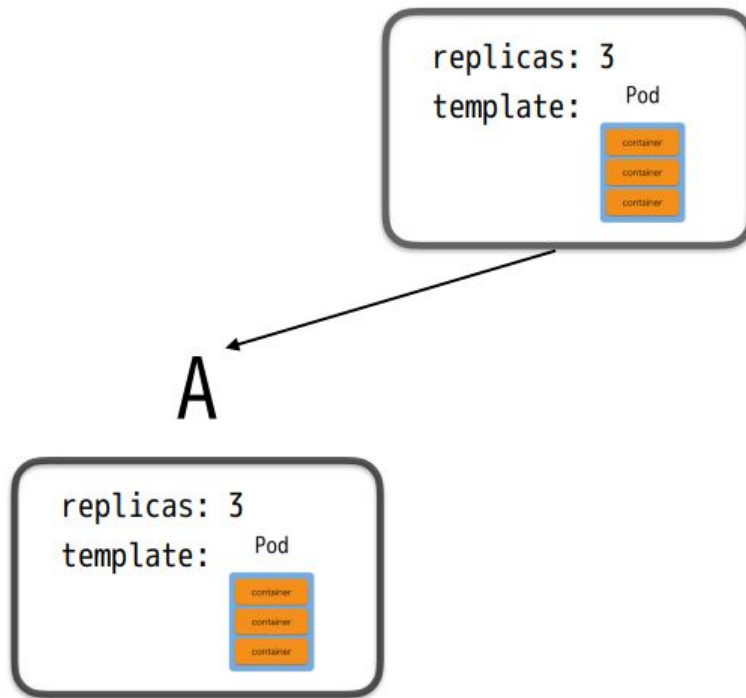


Basic Concepts : Deployment

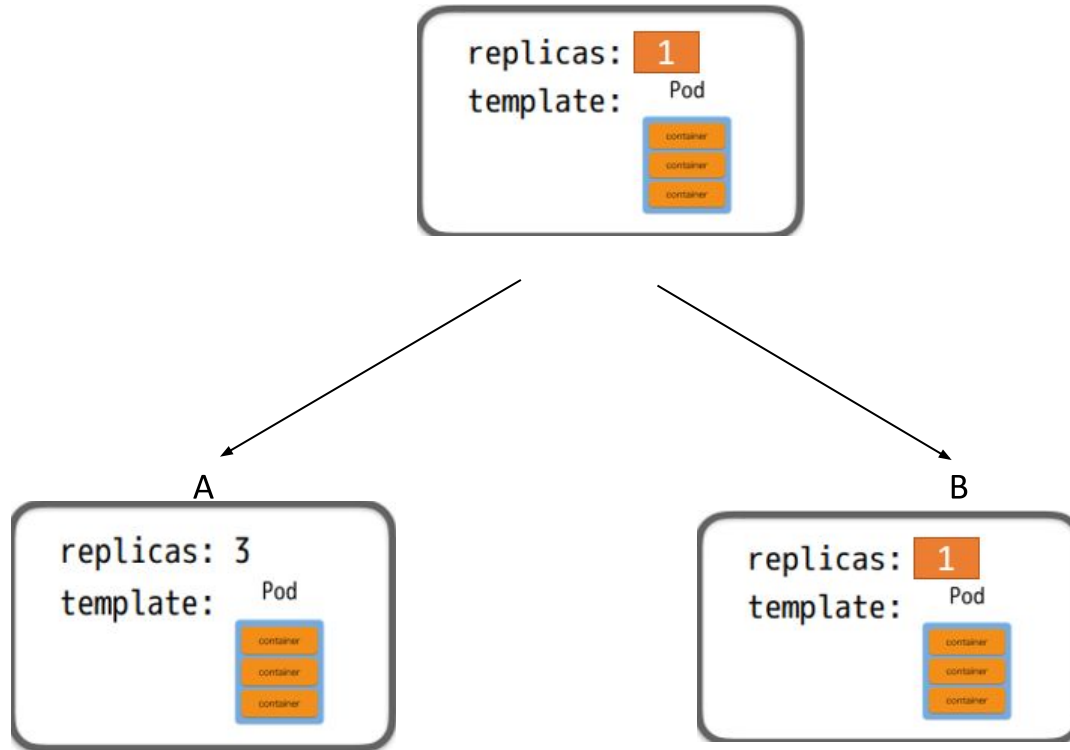


Basic Concepts : Deployment

let's make some
change



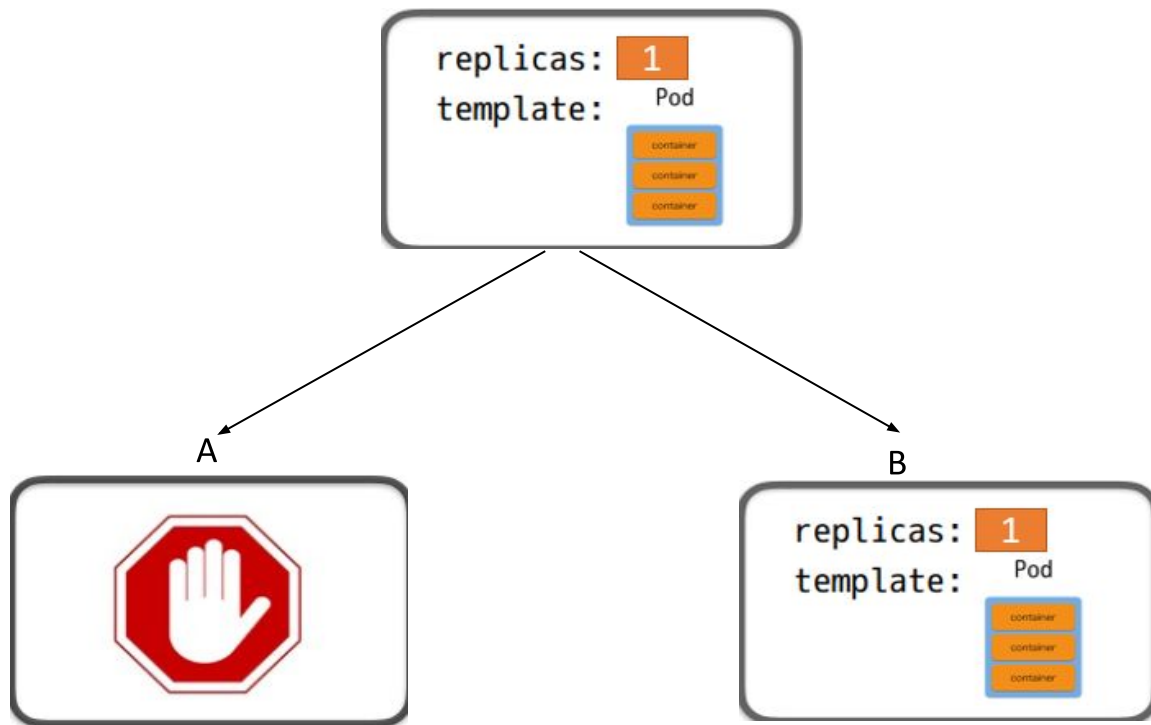
Basic Concepts : Deployment



let's make some change

Basic Concepts : Deployment

let's make some change





let's make some
change

B



Oh no,
wait wait
please

let's make some
change



B



Oh no,
wait wait
please



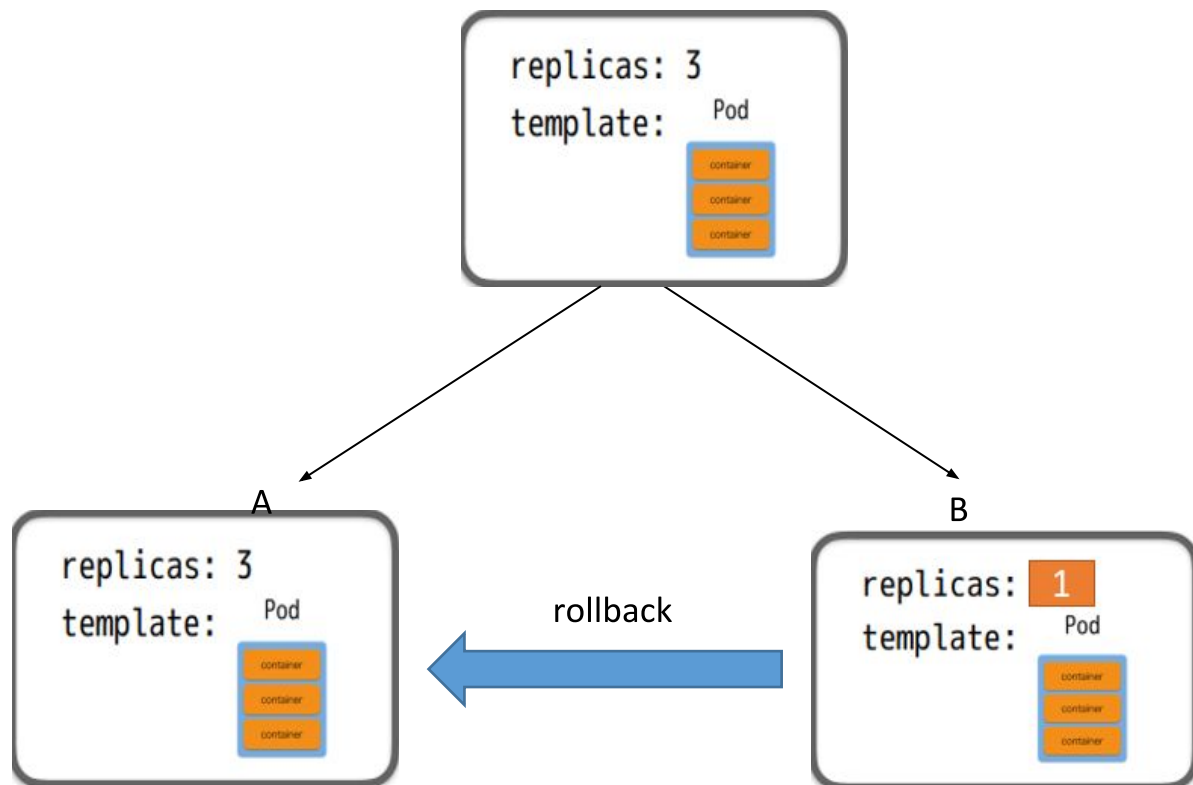
B



let's make some
change

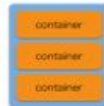
Don't worry, you
can do a rollback

Replication Controller



oh, yes i did it

```
replicas: 3  
template: Pod
```



A

```
replicas: 3  
template: Pod
```



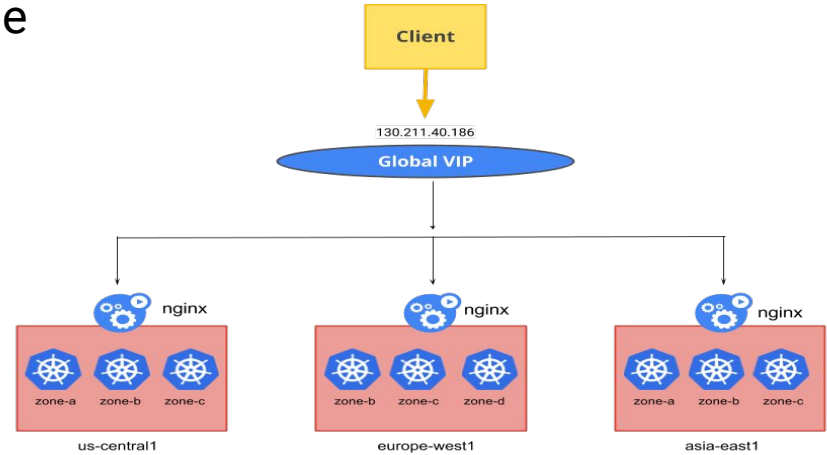
Basic Concepts : Service

A logical grouping of pods that perform the same function

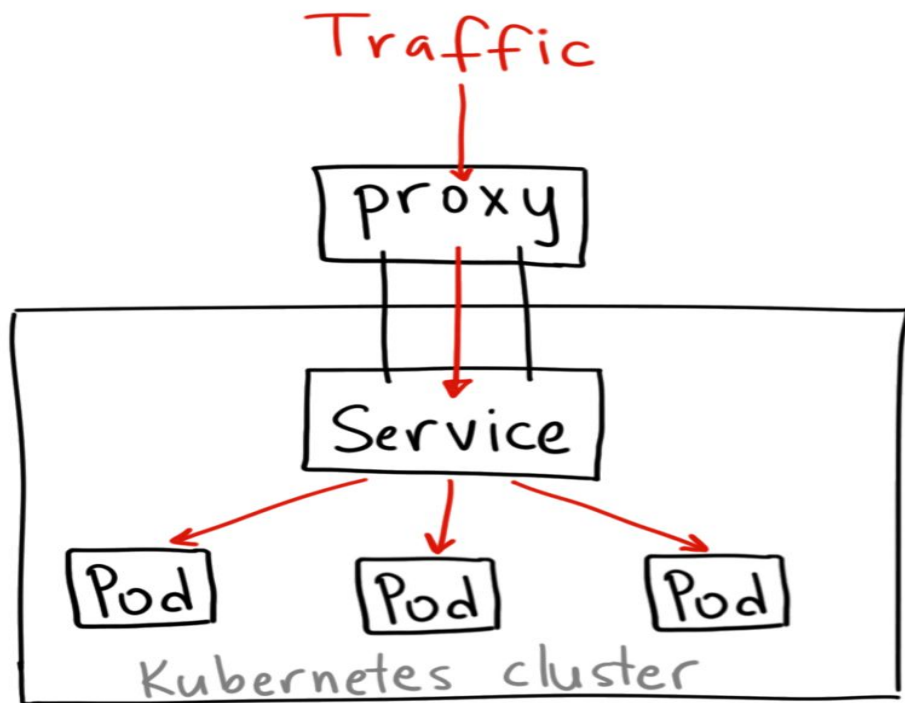
- grouped by label selector

Load balances incoming requests across constituent pods

1. CluserIP
2. NodePort
3. LoadBalancer

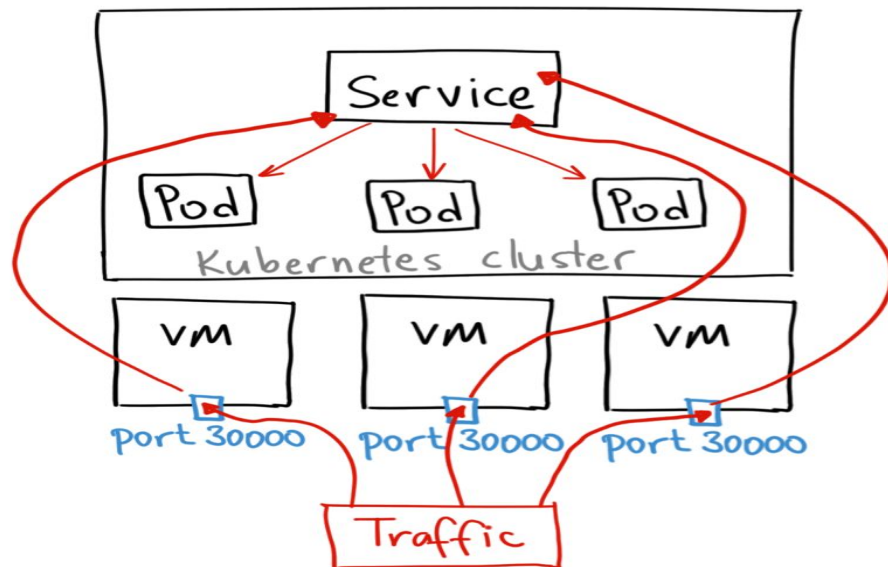


Service : ClusterIP



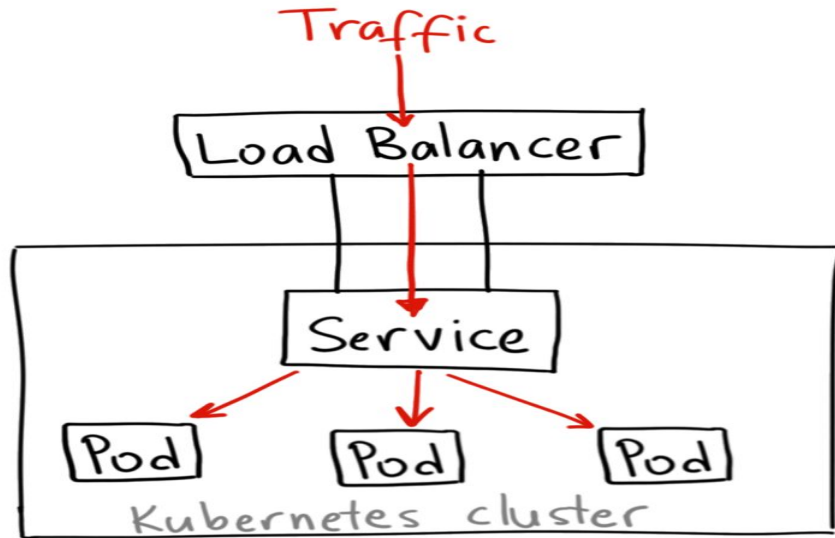
It gives you a service inside your cluster that other apps inside your cluster can access. There is no external access.

Service : NodePort



A NodePort service is the most primitive way to get external traffic directly to your service. NodePort, as the name implies, opens a specific port on all the Nodes, and any traffic that is sent to this port is forwarded to the service.

Service : LoadBalancer



A LoadBalancer service is the standard way to expose a service to the internet.

end

The lab is available on

<https://github.com/etadata/owasp-workshop>

