Orchestrating Azure Container Service Cluster using Kubernetes

SONU SATHYADAS


sonusathyadas@hotmail.com


https://in.linkedin.com/in/sonusathyadas


@sonusathyadas

Sonu Sathyadas

Tech Lead & Training Consultant @ Synergetics India

8+ Years of corporate training experience

Microsoft CERTIFIED Trainer

Microsoft Specialist

Developing Microsoft Azure Solutions

Microsoft Specialist

Implementing Microsoft Azure Infrastructure Solutions

Agenda

- Containerization
- Docker
- Docker Hub and Azure Container Registry
- Kubernetes
- Azure Container Services
- Conclusion

Containers

OS-level virtualization method for deploying and running distributed applications.

Run on a single control host and access a single kernel.

Hardware Virtualization Vs Application Isolation

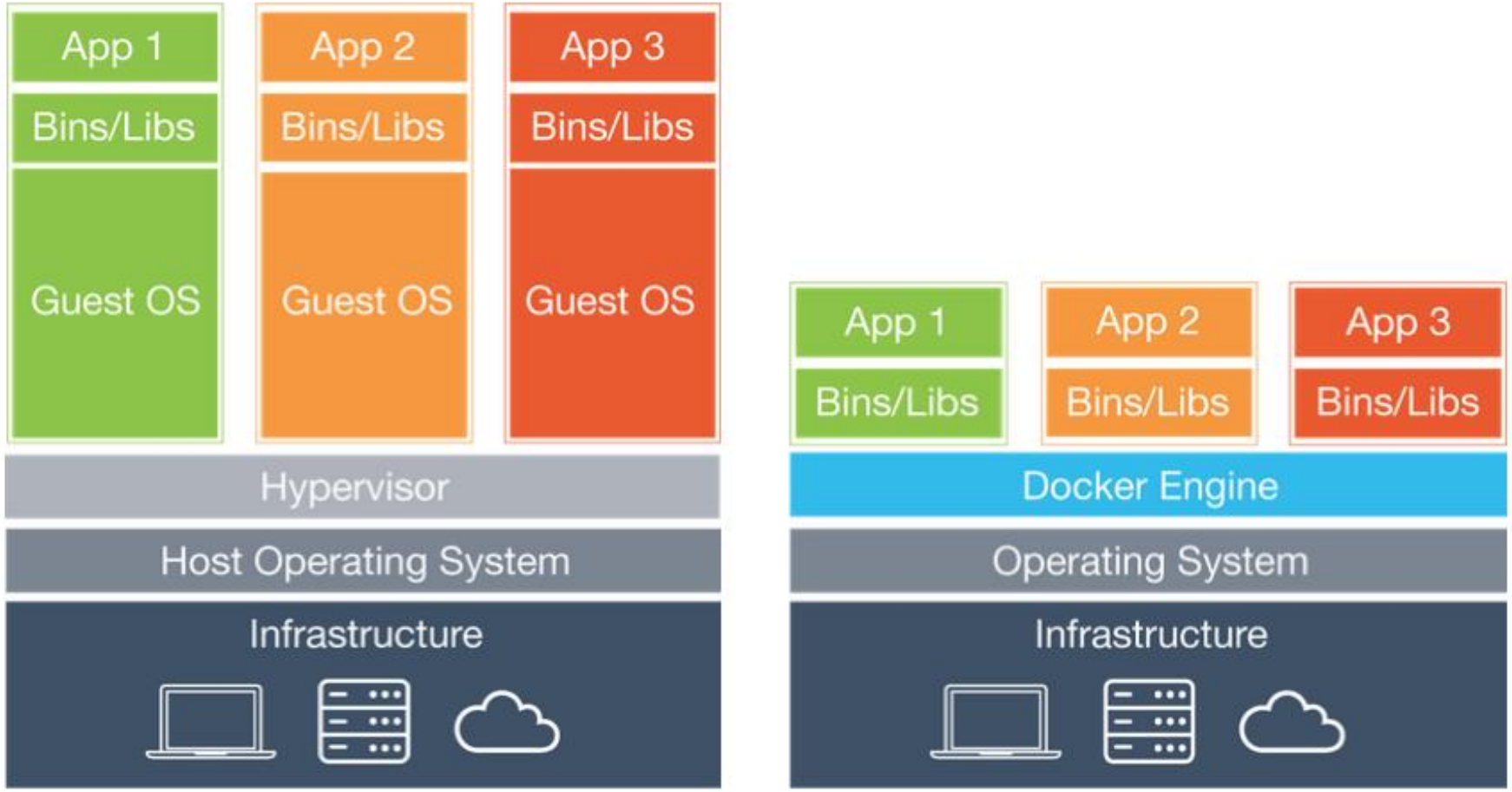
- Containers and virtual machines both allow you to abstract the workload from the underlying hardware.
- VMs mimics a complete server - OS, Applications, Network.
- VMs runs on top of Hypervisor (eg: VMWare, Hyper-V)
- Containerization allows virtual instances to share a single host operating system and relevant binaries, libraries or drivers.
- Containerization is handled by a containerization engine, like Docker.

Virtual Machines Vs Containers

- Containers are an abstraction at the app layer that packages code and dependencies together.
- Multiple containers can run on the same machine and share the OS kernel with other containers, each running as isolated processes in user space.
- Containers take up less space than VMs
- Containers size is small (MBs)
- Provisioning containers only take a few seconds or less.
- Containers are a very cost effective solution.
- Container-based virtualization are a great option for microservices, DevOps, and continuous deployment.

- Virtual machines are an abstraction of physical hardware turning one server into many servers
- VM hypervisor allows multiple VMs to run on a single machine.
- Each VM includes a full copy of an operating system, one or more apps, necessary binaries and libraries - taking up tens of GBs.
- VMs can also be slow to boot.

Virtual Machiens Vs Containers



Advantages of containers

Container is notably smaller, easier to migrate or download, faster to backup or restore and requires less memory.

No licence cost.

Optimized utilization of host resources.

Support many more containers on the same infrastructure.

Fast deployment than VMs

Better for microservices and continuous integration and delivery.

Disadvantages of Containers

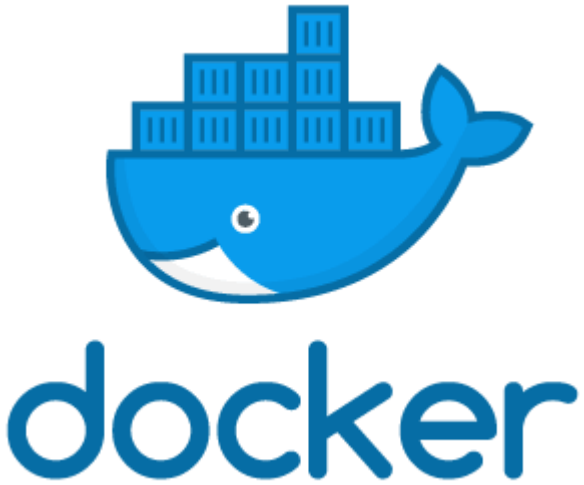
Lack of isolation from the host OS

Each container must use the same OS as the base OS.

Lack of security - Host OS attack affect containers too.

Container implementations

- Docker
- Rocket (rkt) from CoreOS
- LXD containerization engine for Ubuntu
- BSD Jails
- LXC
- Solaris Zones
- NonLinux-based Container Solutions
  - Windows Server containers and Hyper-V containers



Docker

IMPLEMENTING DOCKER CONTAINERS

Docker images

Docker is the company driving the container movement and the only container platform provider to address every application across the hybrid cloud.

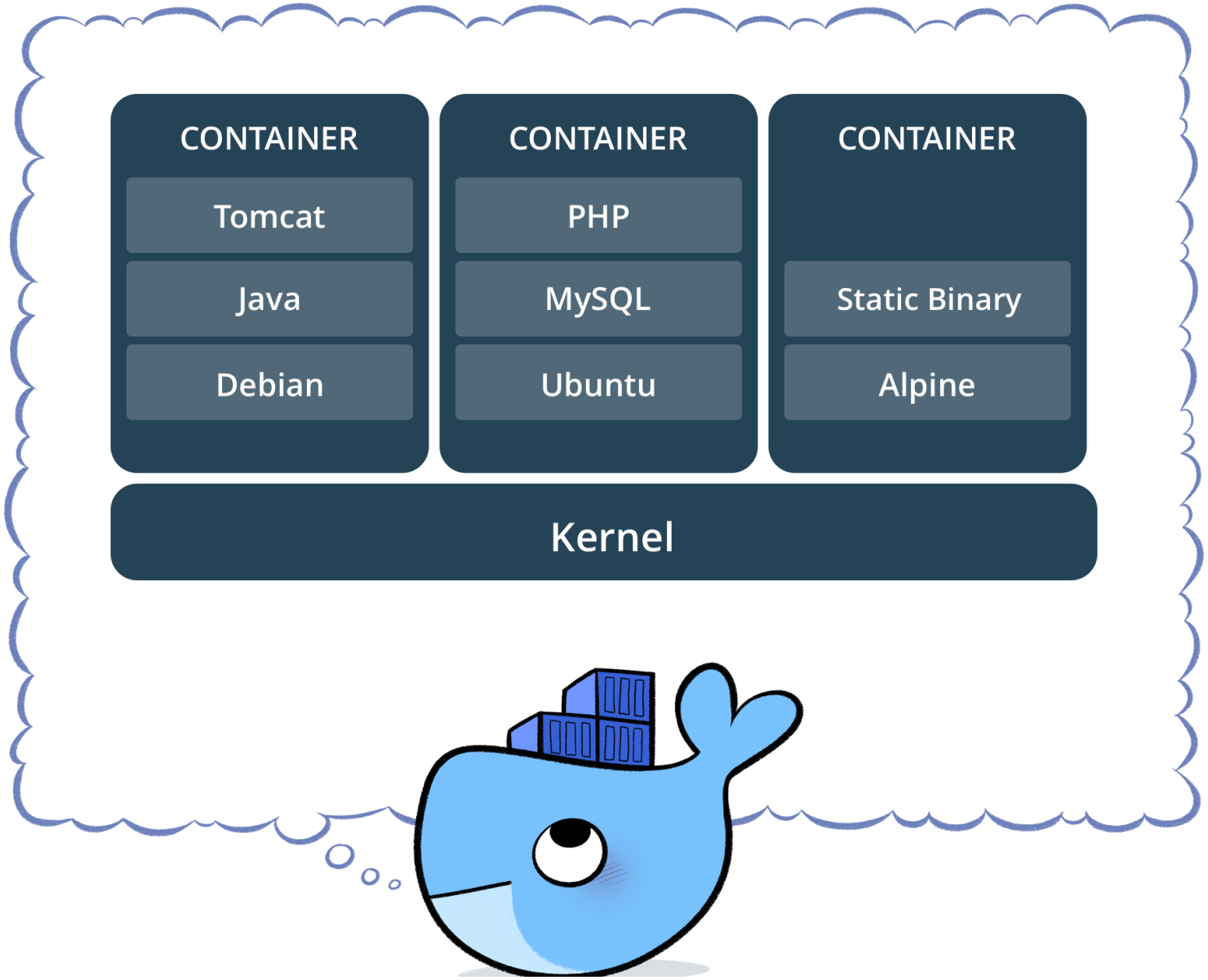
Container image is a lightweight, stand-alone, executable package of a piece of software.

Docker image includes everything needed to run it: code, runtime, system tools, system libraries, settings.

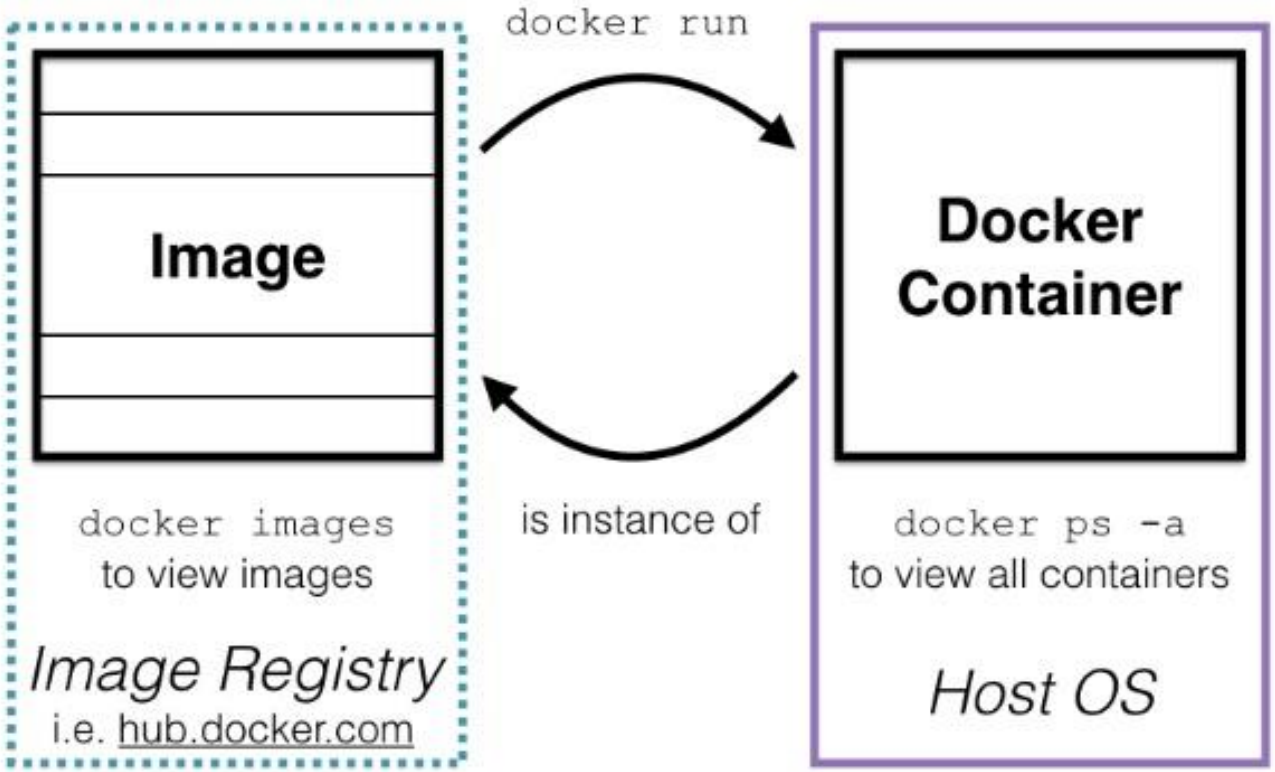
Available for both Linux and Windows based apps

Containers isolate software from its surroundings

Images are constructed from filesystem layers and share common files to minimizes disk usage and image downloads are much faster.


CONTAINER
Tomcat
Java
Debian
CONTAINER
PHP
MySQL
Ubuntu
CONTAINER
Static Binary
Alpine
Kernel

Docker images VS Containers

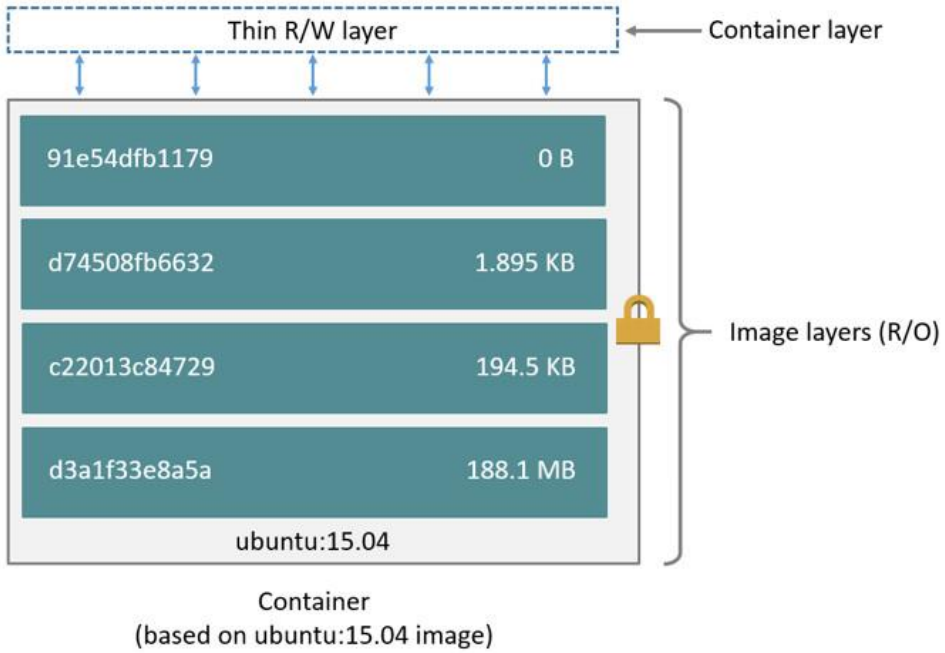


Docker image and layers

Docker image is built up from a series of layers.

Each layer represents an instruction in the image's Dockerfile.

Each layer except the very last one is read-only.



Dockerfile

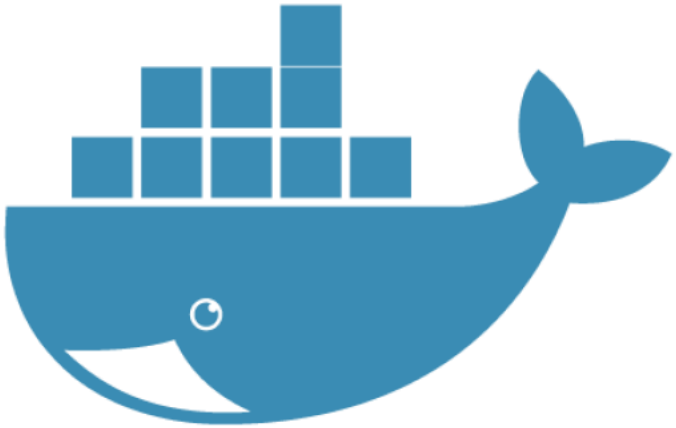
A Dockerfile is a text document that contains all the commands a user could call on the command line to assemble an image.

```
FROM ubuntu:15.04
COPY . /app
RUN make /app
CMD python /app/app.py
```

Command	Description
docker build	Build an image from a Dockerfile
docker commit	Create a new image from a container’s changes
docker cp	Copy files/folders between a container and the local filesystem
docker create	Create a new container
docker exec	Run a command in a running container
docker image	Manage images
docker images	List images
docker login	Log in to a Docker registry
docker logout	Log out from a Docker registry
docker ps	List containers
docker pull	Pull an image or a repository from a registry
docker push	Push an image or a repository to a registry
docker rename	Rename a container
docker restart	Restart one or more containers
docker rm	Remove one or more containers
docker rmi	Remove one or more images
docker run	Run a command in a new container
docker start	Start one or more stopped containers
docker stop	Stop one or more running containers

DEMO

CREATING DOCKER CONTAINERS



Docker Hub

THE REPOSITORY FOR DOCKER IMAGES

Docker Hub

Repository for docker images.

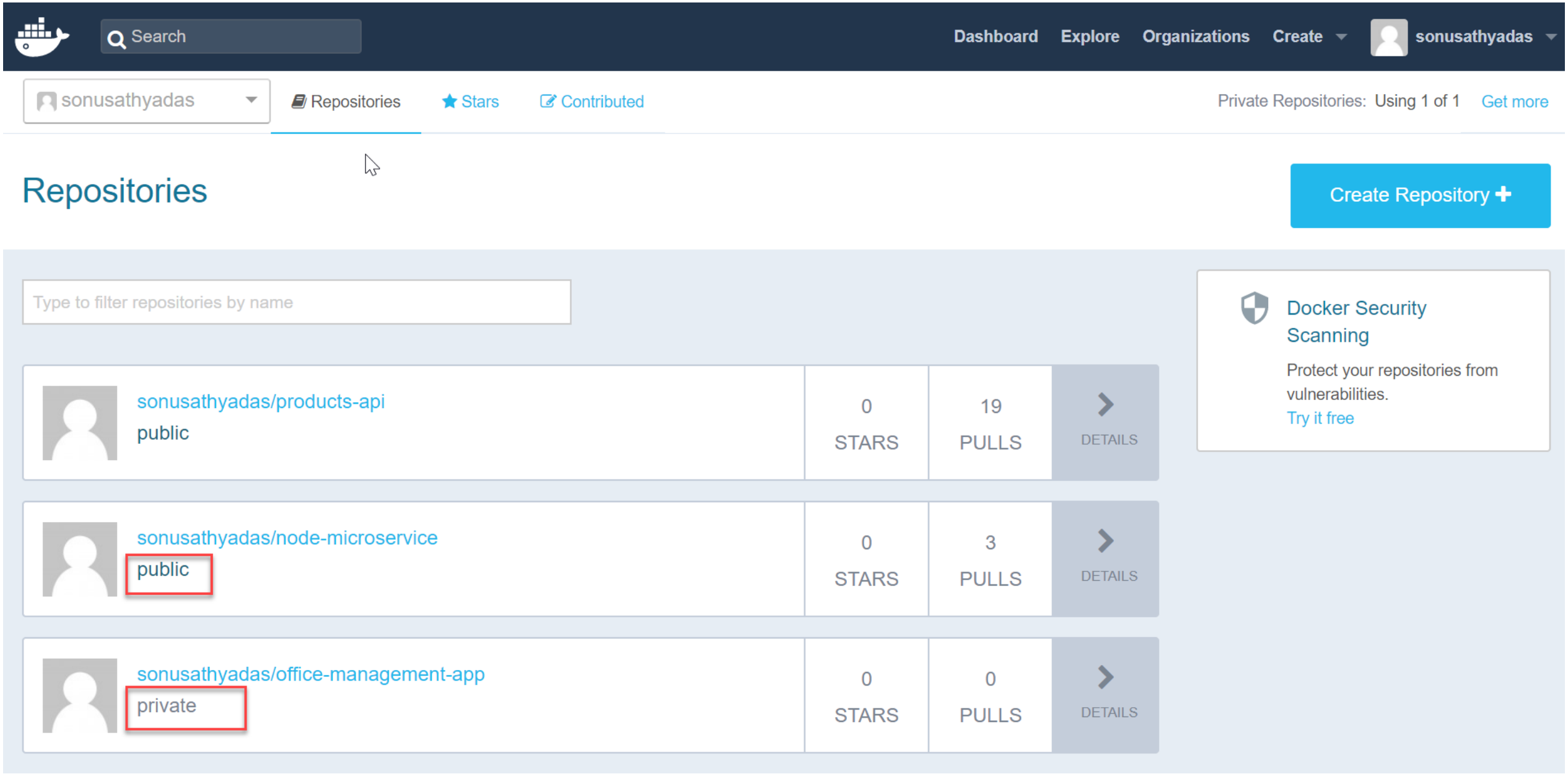
Public or private repositories.

Organization in Docker Cloud contains Teams, and each Team contains users.

Push images to Docker hub

- Login to docker hub account
- Build the image with a tag
- Push images using '***Docker push***' command

Pull the image using '***docker pull***' command


Search
Dashboard
Explore
Organizations
Create
sonusathyadas
sonusathyadas
Repositories
Stars
Contributed
Private Repositories: Using 1 of 1
Get more
Repositories
Create Repository +
Type to filter repositories by name
sonusathyadas/products-api
public
0
STARS
19
PULLS
DETAILS
sonusathyadas/node-microservice
public
0
STARS
3
PULLS
DETAILS
sonusathyadas/office-management-app
private
0
STARS
0
PULLS
DETAILS
Docker Security Scanning
Protect your repositories from vulnerabilities.
Try it free

Push images to Docker Hub

Login to Docker Hub

docker login

Build the image in local repository.

docker build –t <imagename:tag> .

Upload the docker image to Docker hub

docker push <imagename:tag>

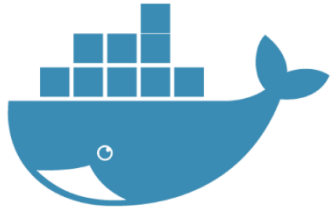
Pull image from Docker Hub

Login to Docker Hub

docker login

Pull image from Docker Hub

docker pull <imagename:tag>



Plan and Pricing

Plan	Price	Private Repositories	Parallel Builds
Free	$0/mo	1	1
Micro	$7/mo	5	5
Small	$12/mo	10	10
Medium	$22/mo	20	20
Large	$50/mo	50	50
XLarge	$100/mo	100	100
XX-Large	$250/mo	250	250
XXX-Large	$500/mo	500	500

Kubernetes

ORCHESTRATING CLUSTERS

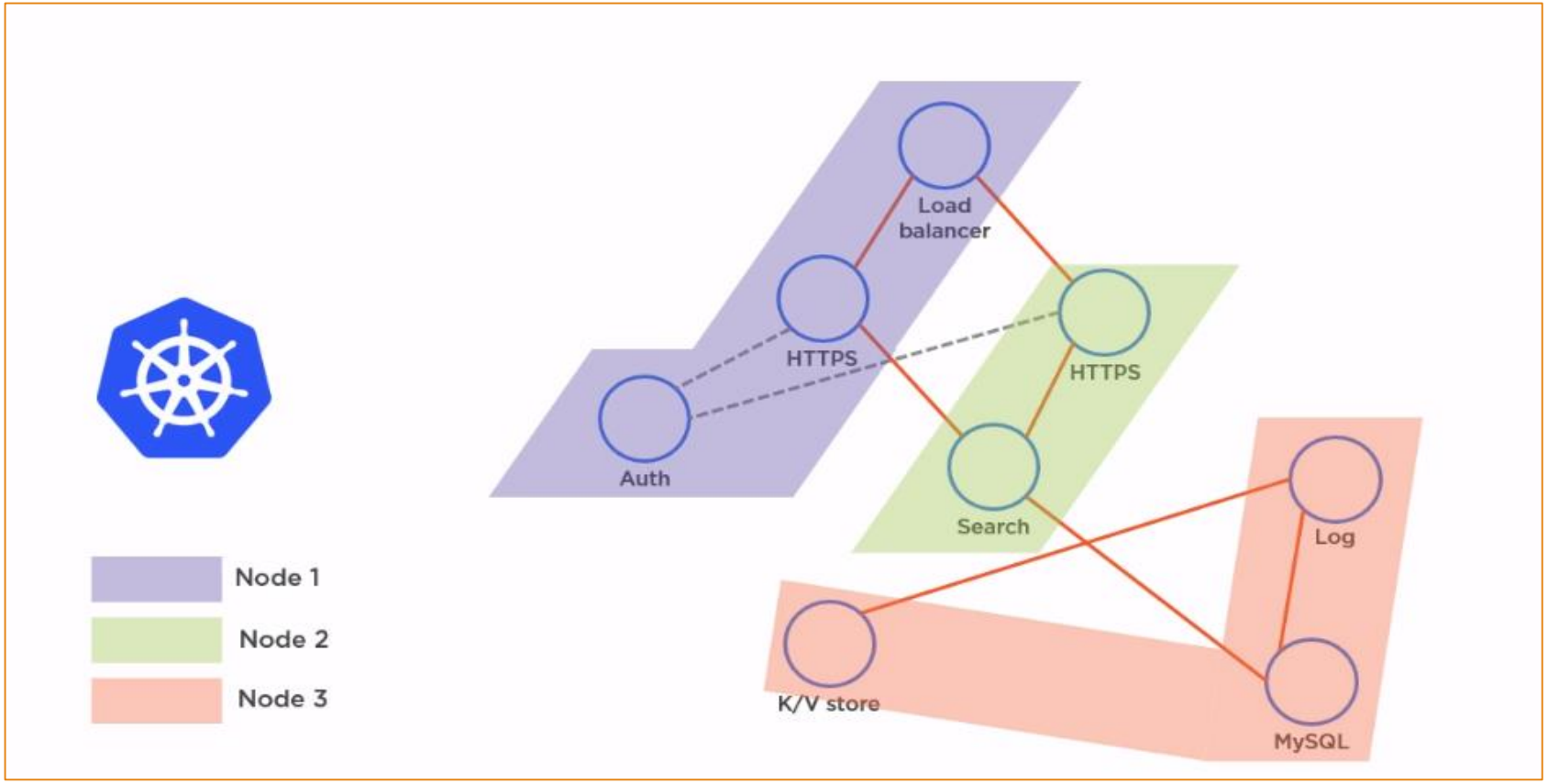
Kuberbetes

- Initialized by Google
- Open-source orchestration tool for Clusters.
- Written in Go/GoLang
- Derived from Google’s proprietary container management tools- Borg and omega
-

Components

- Master
- Nodes
- Pod
- Replication Controller (ReplicaSet)
- Services
- Deployments

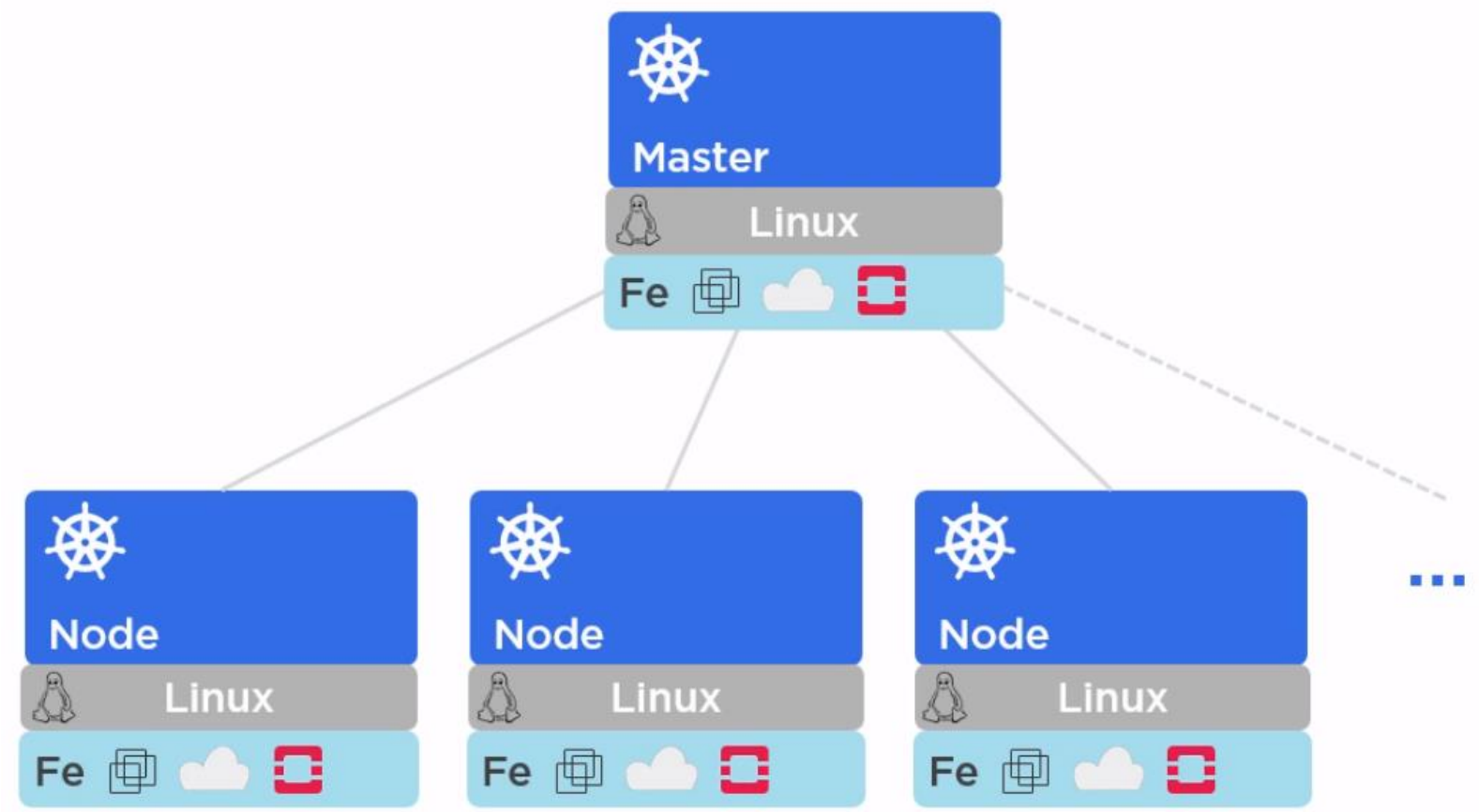
Kubernetes - The Big picture



Kubernetes Cluster

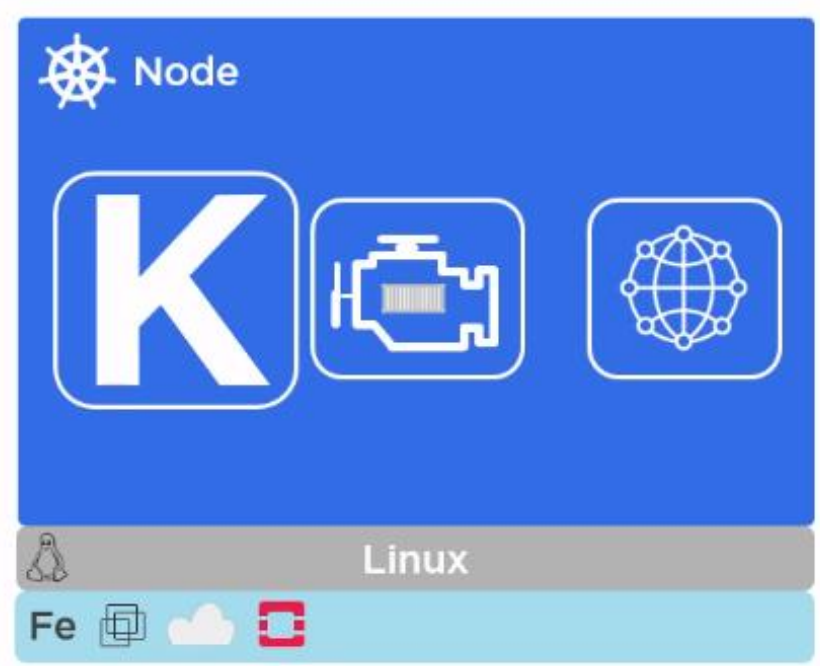
- Coordinates highly available cluster of machines that works as single unit.
- Machines – Master and Nodes
- Automates the distribution and scheduling of application containers across a cluster
- Apps need to be containerized.
- Containers runs on Pod
- Node contains multiple Pods
- Master controls Worker Nodes

Kubernetes Cluster



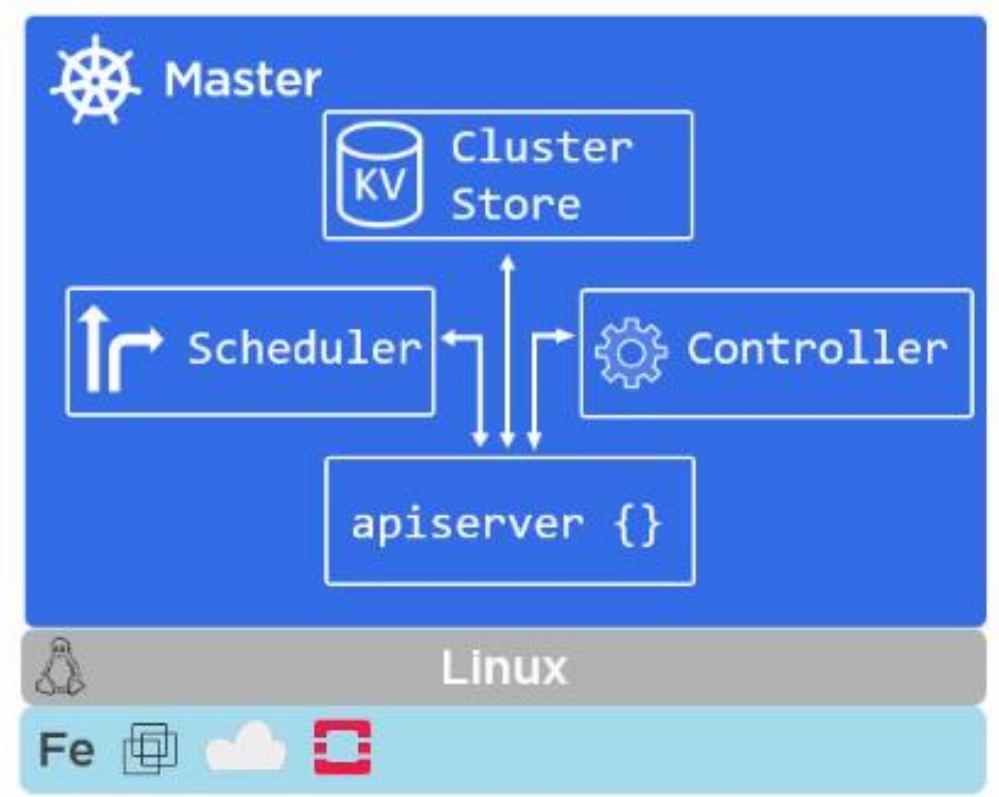
Cluster Node

- Workers in a cluster
- Apps runs in node, not in master
- VM or Physical machine
- Node components
  - Kubelet
  - Container management tool - Docker or Rkt
- 3 node recommended for production (min).
- Communicates with master using kubernets API



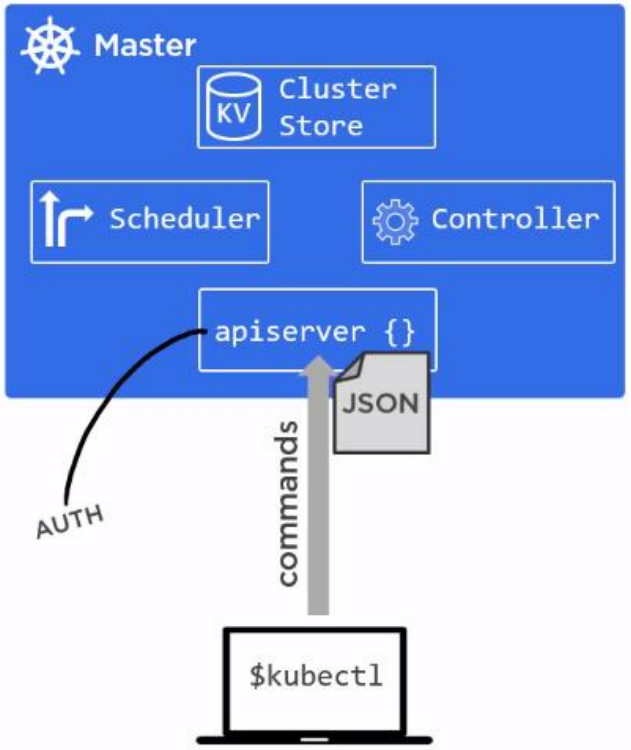
Cluster Master

- Managing the cluster
- Coordinates all activities in cluster
  - Scheduling applications
  - Maintaining applications' desired state
  - Scaling applications
  - Rolling out new updates
- One or more masters for HA.
- Master components
  - Kube-apiserver
  - Cluster store (etcd)
  - Kube-controller-manager
  - Kube-scheduler



Kube-apiserver

- Master are only exposed to public.
- Expose the REST API to connect with the cluster
- kube-apiserver exposes the Kubernetes API
- front-end for the Kubernetes control plane
- scale horizontally – scales by deploying more instances.



Cluster Store (etcd)

- Used as Kubernetes' backing store.
- Always have a backup plan for etcd's data
- Stores Cluster state and configuration

Kube-controller-manager

- Controller of controllers
- Runs background controller threads that handle routine tasks in the cluster
- Each controller is a separate process, but all are compiled into a single binary and run in a single process
- Controllers are
  - Node Controller
  - Replication Controller
  - Endpoints Controller
  - Service Account & Token Controllers

Kube-scheduler

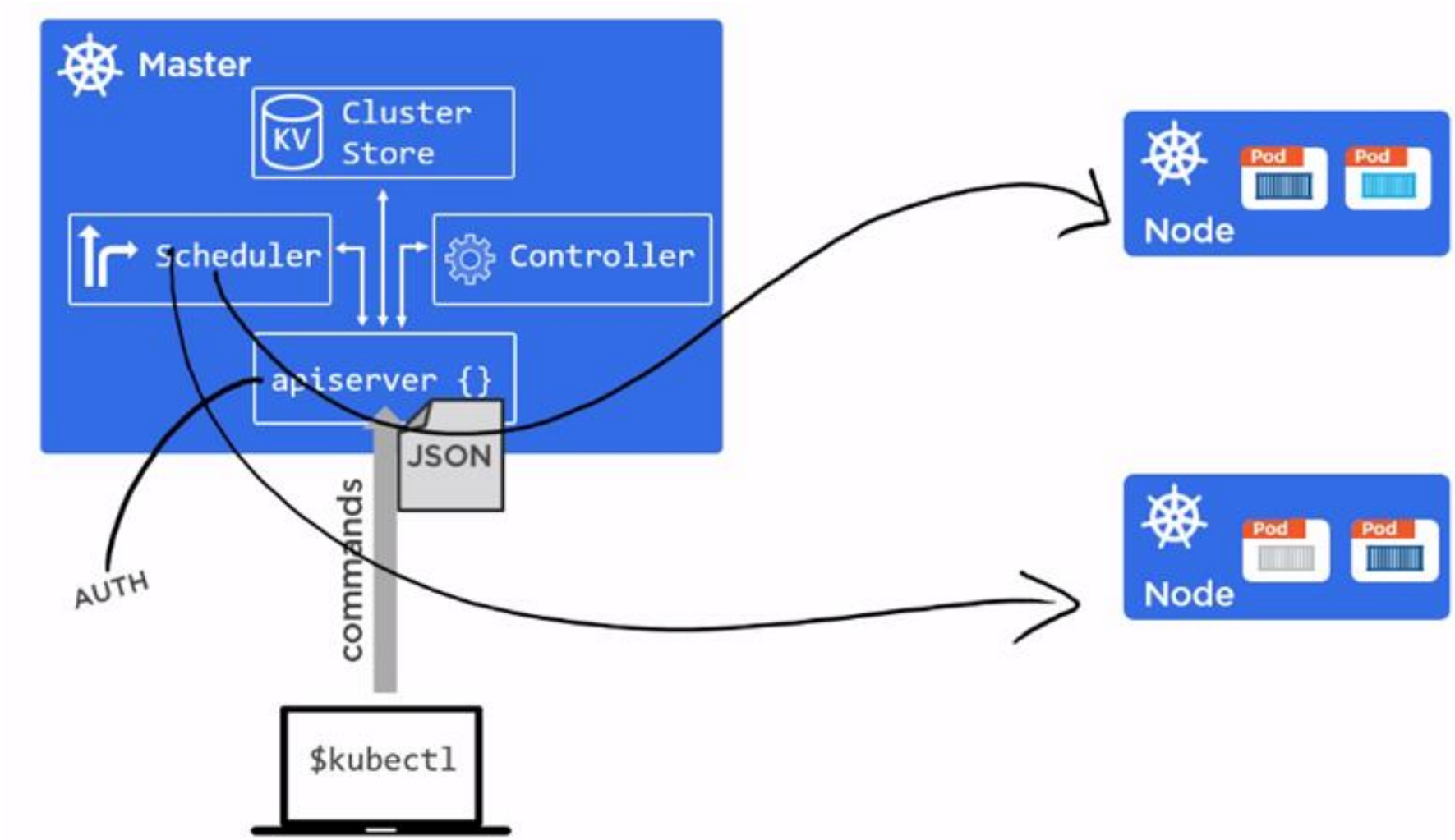
- Watches apiserver for new pods
- Assign works to nodes – affinity/anti-affinity, constraints, resources etc.
- Watches newly created pods that have no node assigned, and selects a node for them to run on

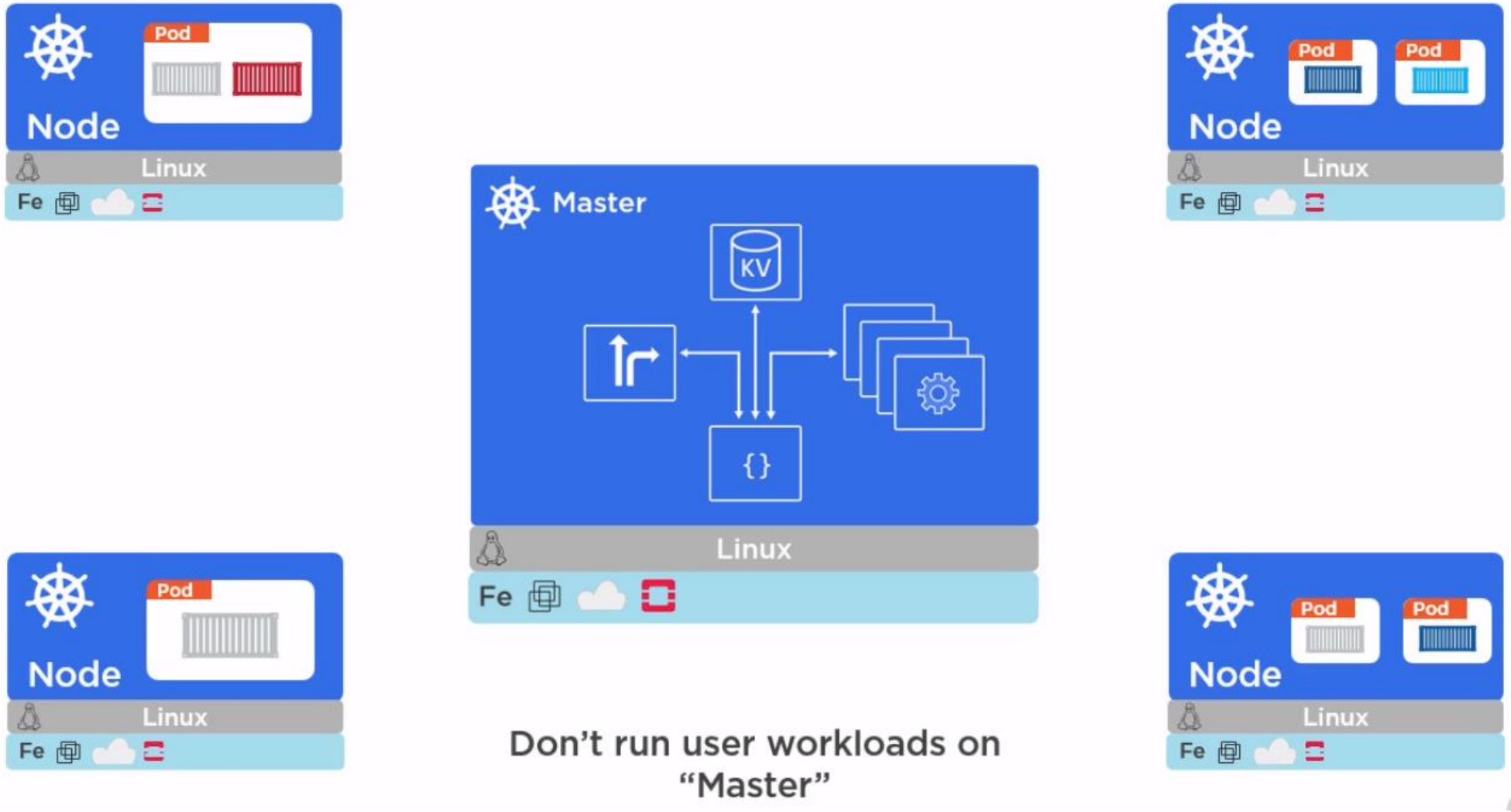
Manifest file

- Declaration of deployments, Services, pods …
- YAML or JSON

nginx-app.yaml

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nginx-svc
  labels:
    app: nginx
spec:
  type: LoadBalancer
  ports:
  - port: 80
  selector:
    app: nginx
---
apiVersion: apps/v1beta1
kind: Deployment
metadata:
  name: my-nginx
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.7.9
        ports:
        - containerPort: 80
```

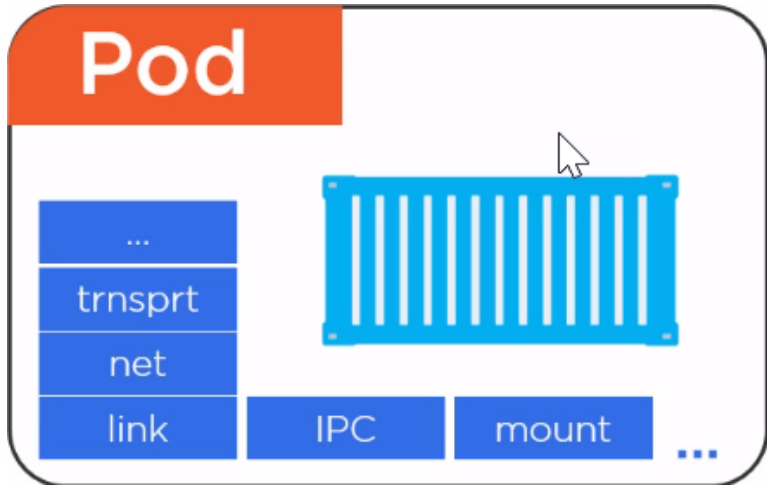

Master
KV
Cluster
Store
Scheduler
Controller
apiserver {}
JSON
commands
AUTH
$kubectl
Pod
Pod
Node
Pod
Pod
Node



Don't run user workloads on "Master"

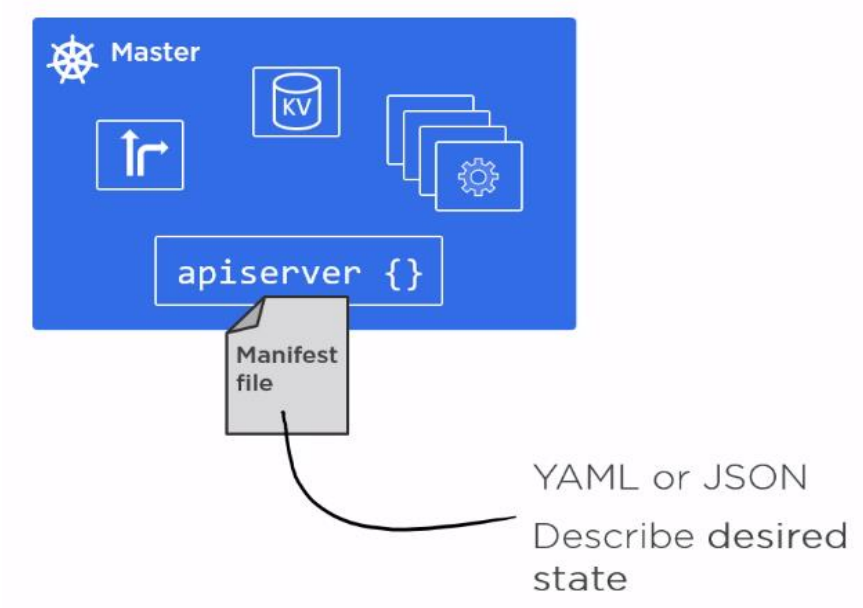
Pod

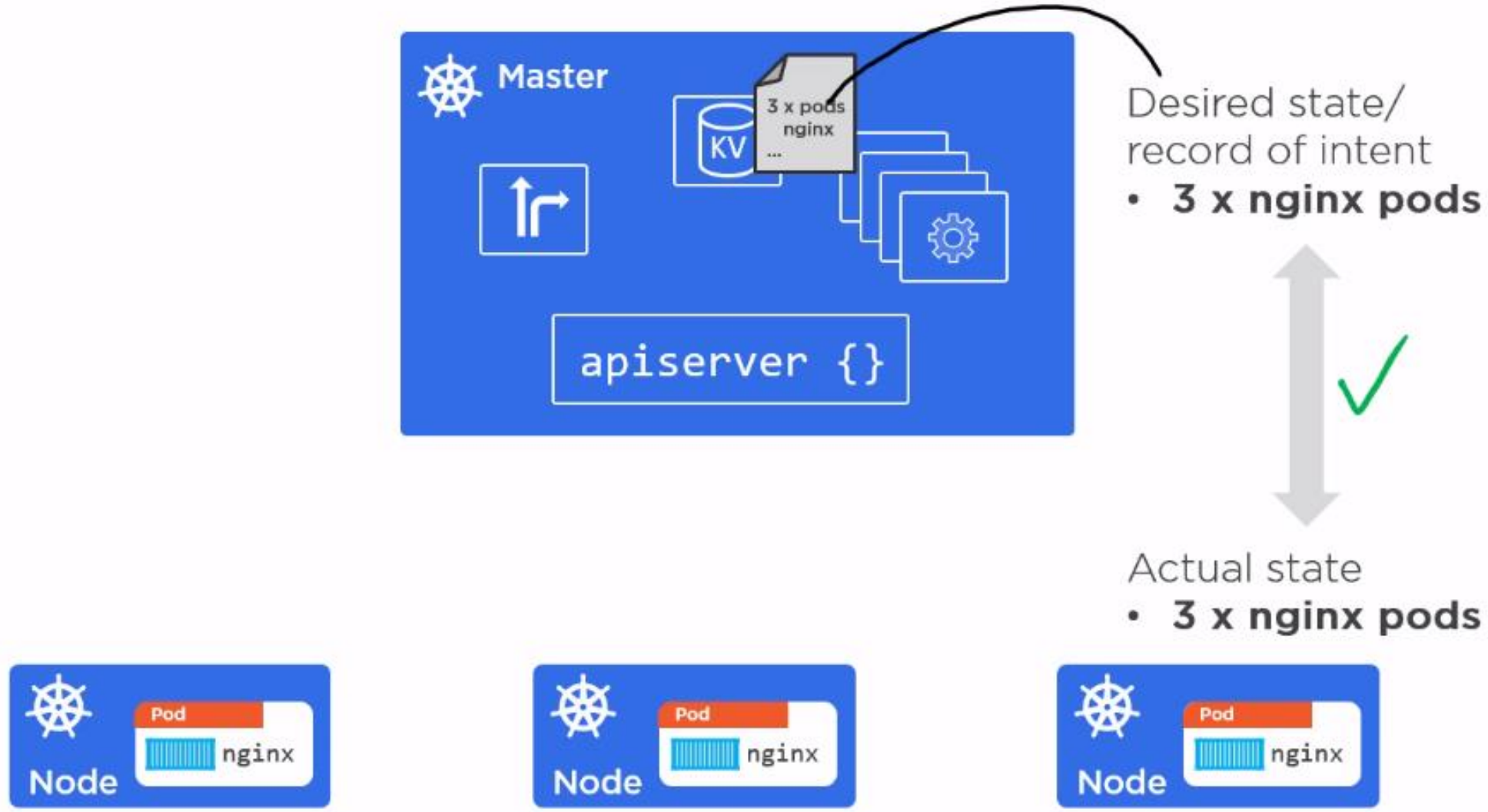
- Basic building block of Kubernetes
- Smallest and simplest unit to create or deploy
- Pod represents a running process on your cluster
- Pod encapsulates
  - Application container (in some cases, multiple containers)
  - Storage resources
  - A unique network IP
  - Options that govern how the container(s) should run

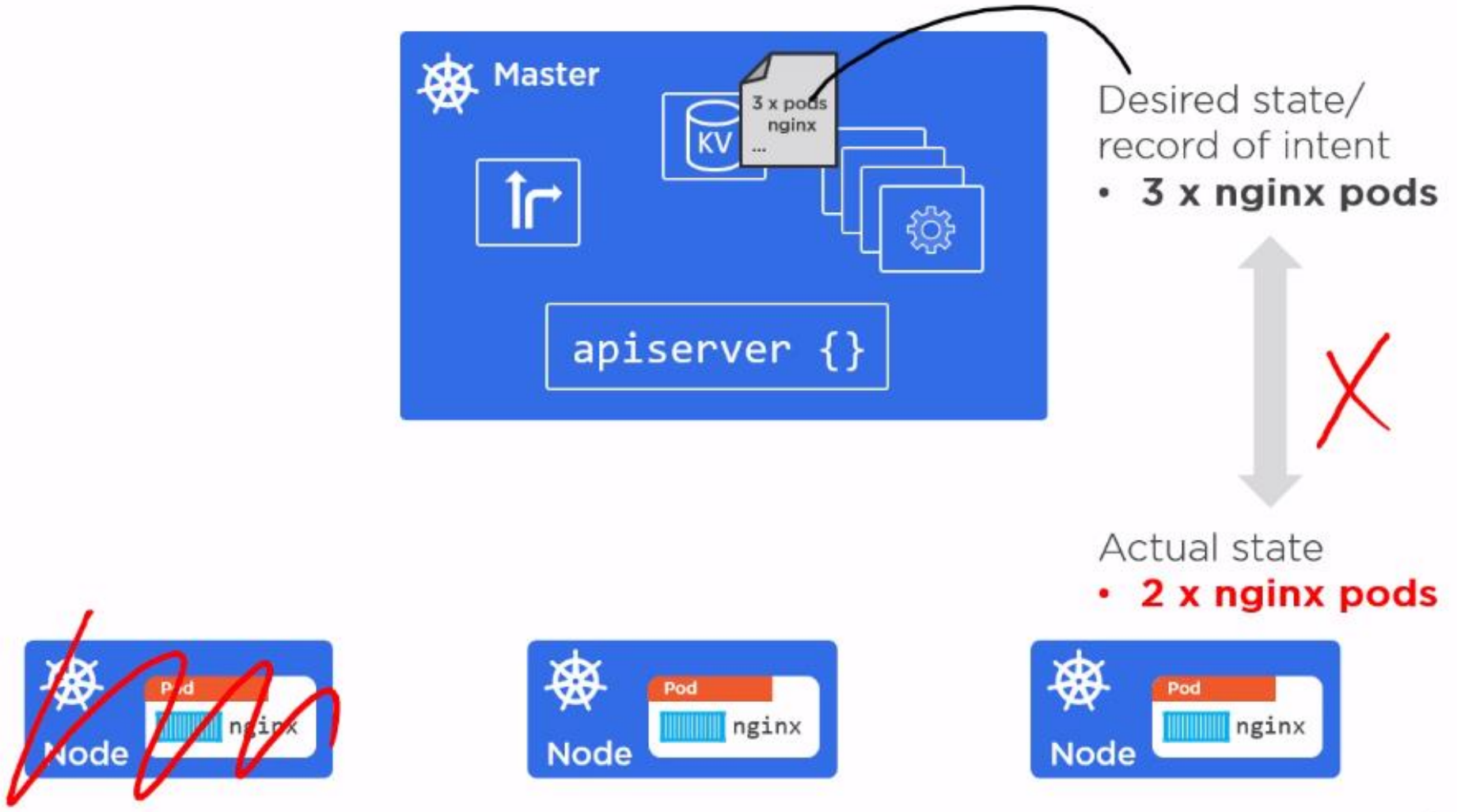


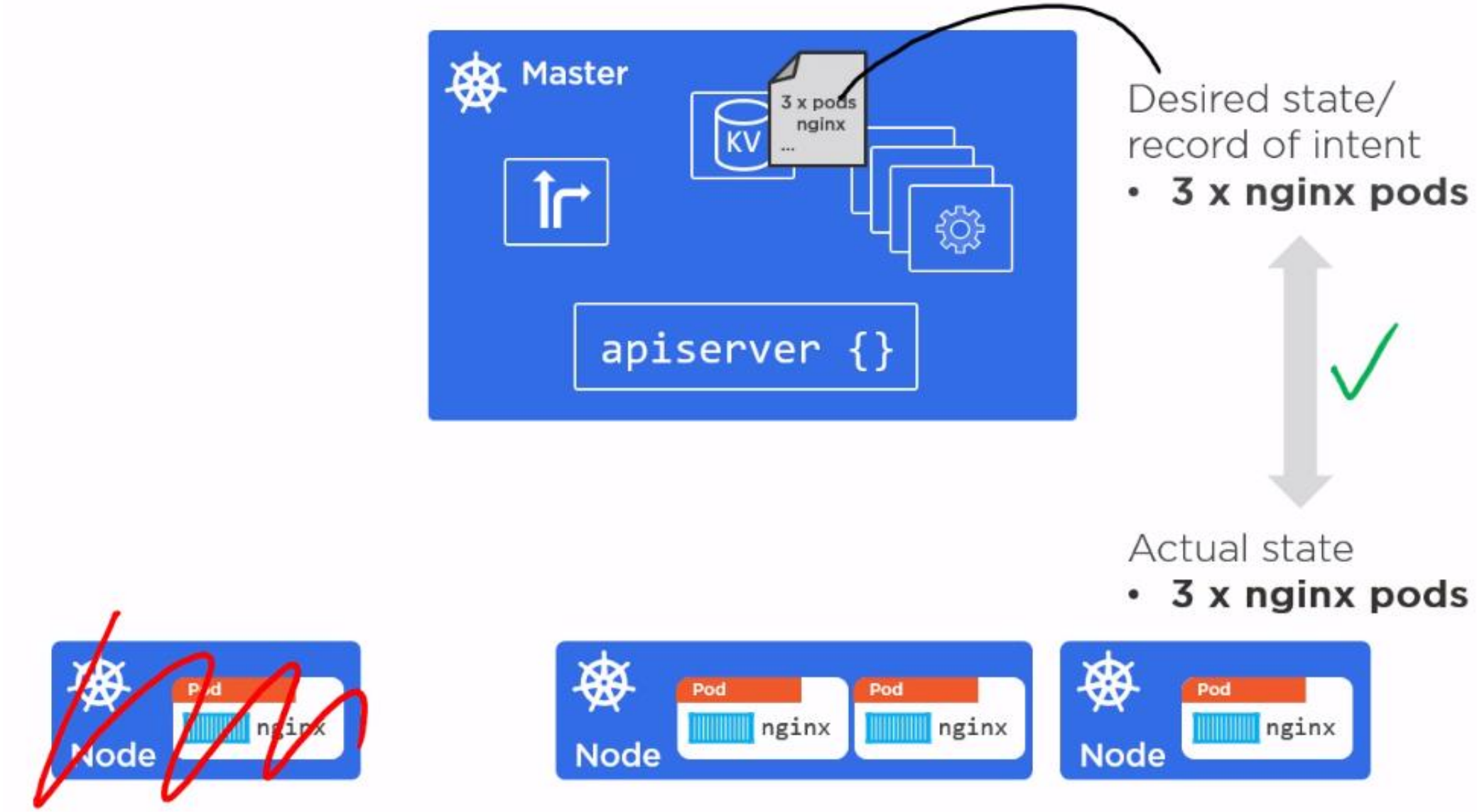
Declarative Model and Desired State

- Kubernetes objects can be created, updated, and deleted using configuration files.
- Configuration files are stored in a directory
- kubectl recursively create and update those objects as needed.



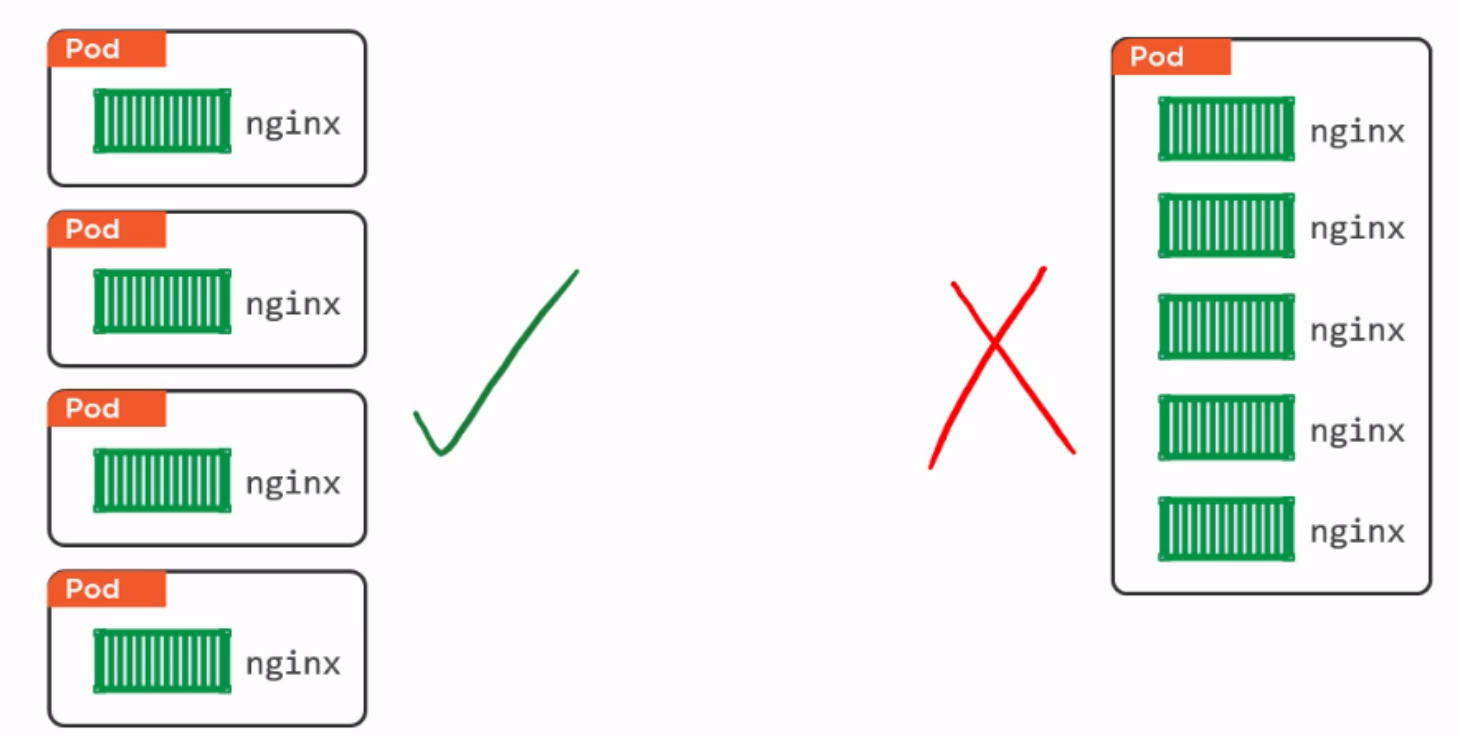

Master
KV
3 x pods
nginx
...
apiserver {}
Desired state/
record of intent
• 3 x nginx pods
Actual state
• 3 x nginx pods
Node
Pod
nginx
Node
Pod
nginx
Node
Pod
nginx


Master
KV
3 x pods
nginx
...
apiserver {}
Desired state/
record of intent
• 3 x nginx pods
Actual state
• 2 x nginx pods
Node
Pod
nginx
Node
Pod
nginx
Node
Pod
nginx


Master
KV
3 x pods
nginx
...
apiserver {}
Desired state/
record of intent
• 3 x nginx pods
Actual state
• 3 x nginx pods
Node
Pod
nginx
Node
Pod
nginx
Pod
nginx
Node
Pod
nginx

Pod scaling

Scale out is recommended



Replication Controller (ReplicaSet)

Ensures a specified number of pod replicas are always up and available

ReplicaSet is the next-generation Replication Controller.

Only difference between a ReplicaSet and a Replication Controller is the selector support.

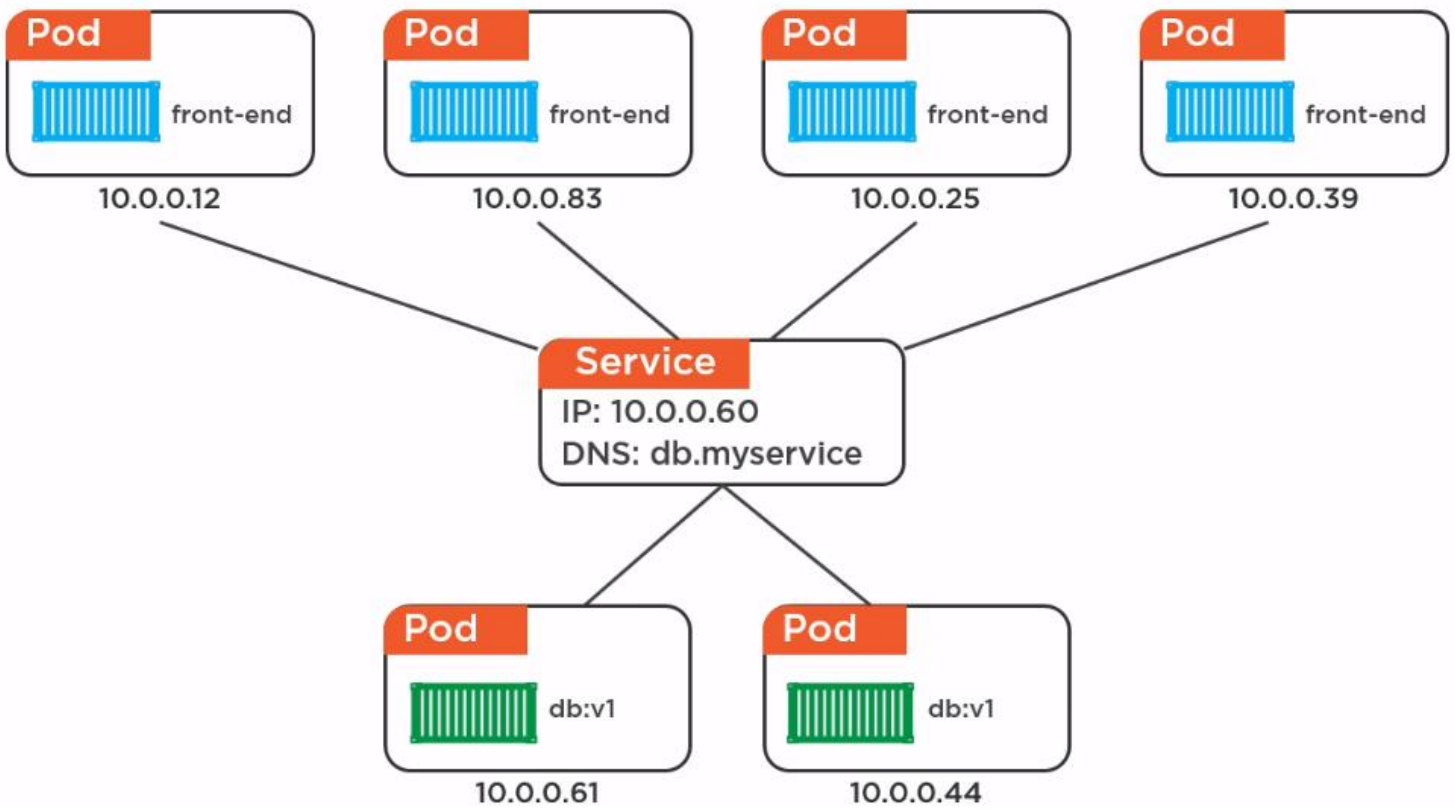
ReplicaSet supports the new set-based selector requirements

Replication Controller only supports equality-based selector requirements

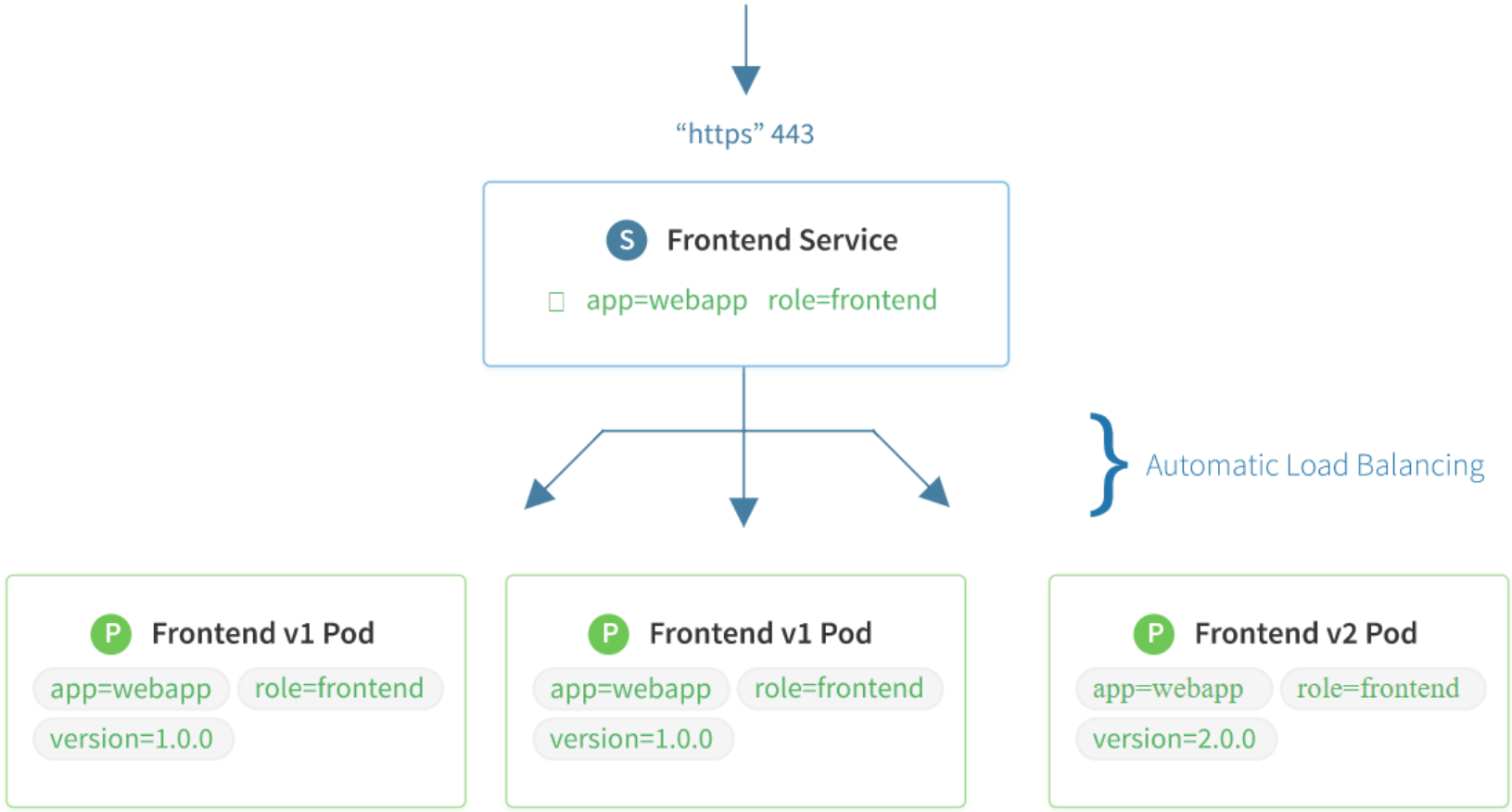
Services

- An abstraction which defines a logical set of Pods and a policy by which to access them.
- Set of Pods targeted by a Service is usually determined by a Label Selector.
- Pods internal IP’s are not reliable for communication.
- IP may change during scaling rolling deployment.
- Services acts as intermediary/Load balancer to access pods.

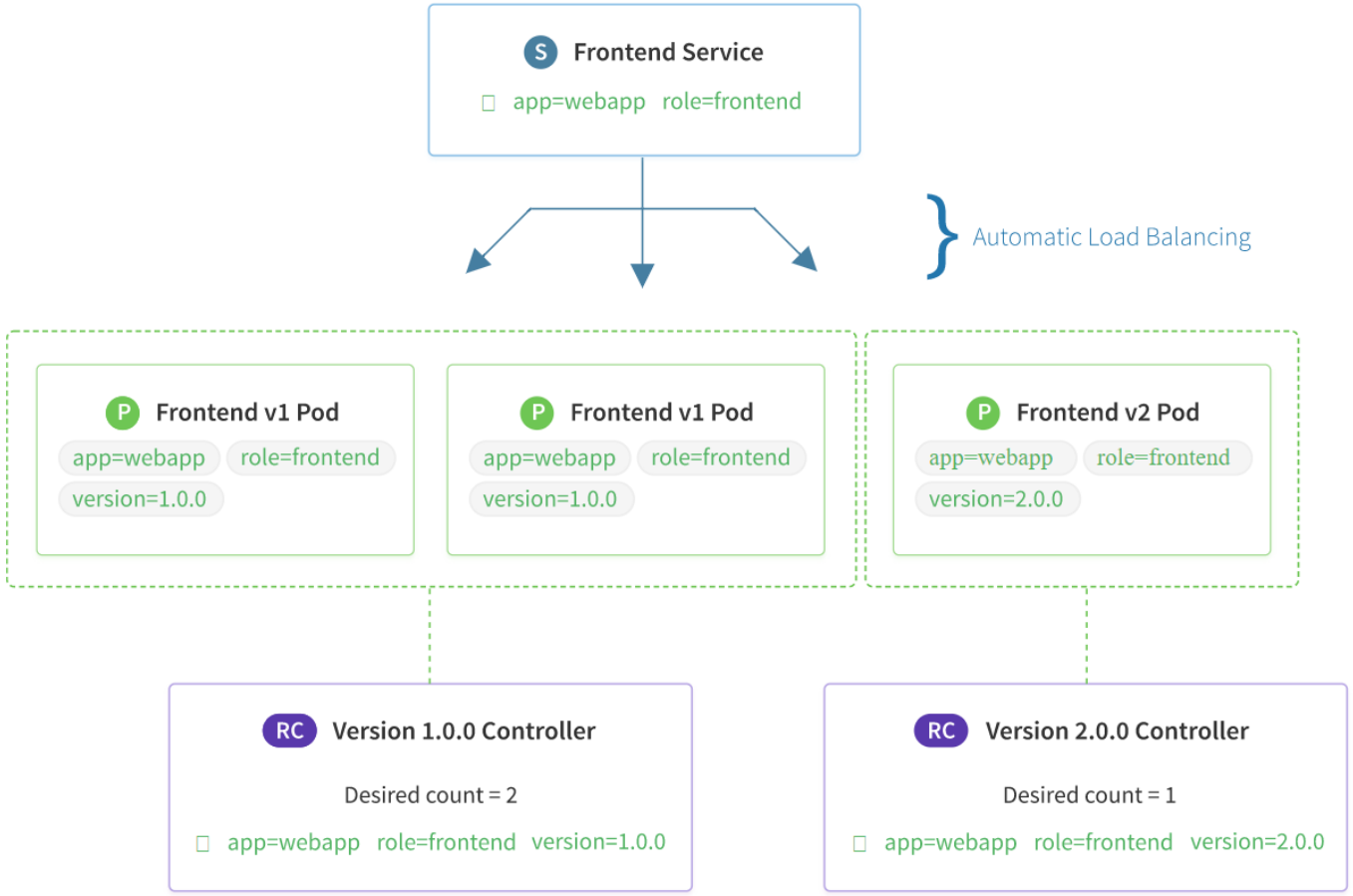
Services


Pod
front-end
10.0.0.12
Pod
front-end
10.0.0.83
Pod
front-end
10.0.0.25
Pod
front-end
10.0.0.39
Service
IP: 10.0.0.60
DNS: db.myservice
Pod
db:v1
10.0.0.61
Pod
db:v1
10.0.0.44

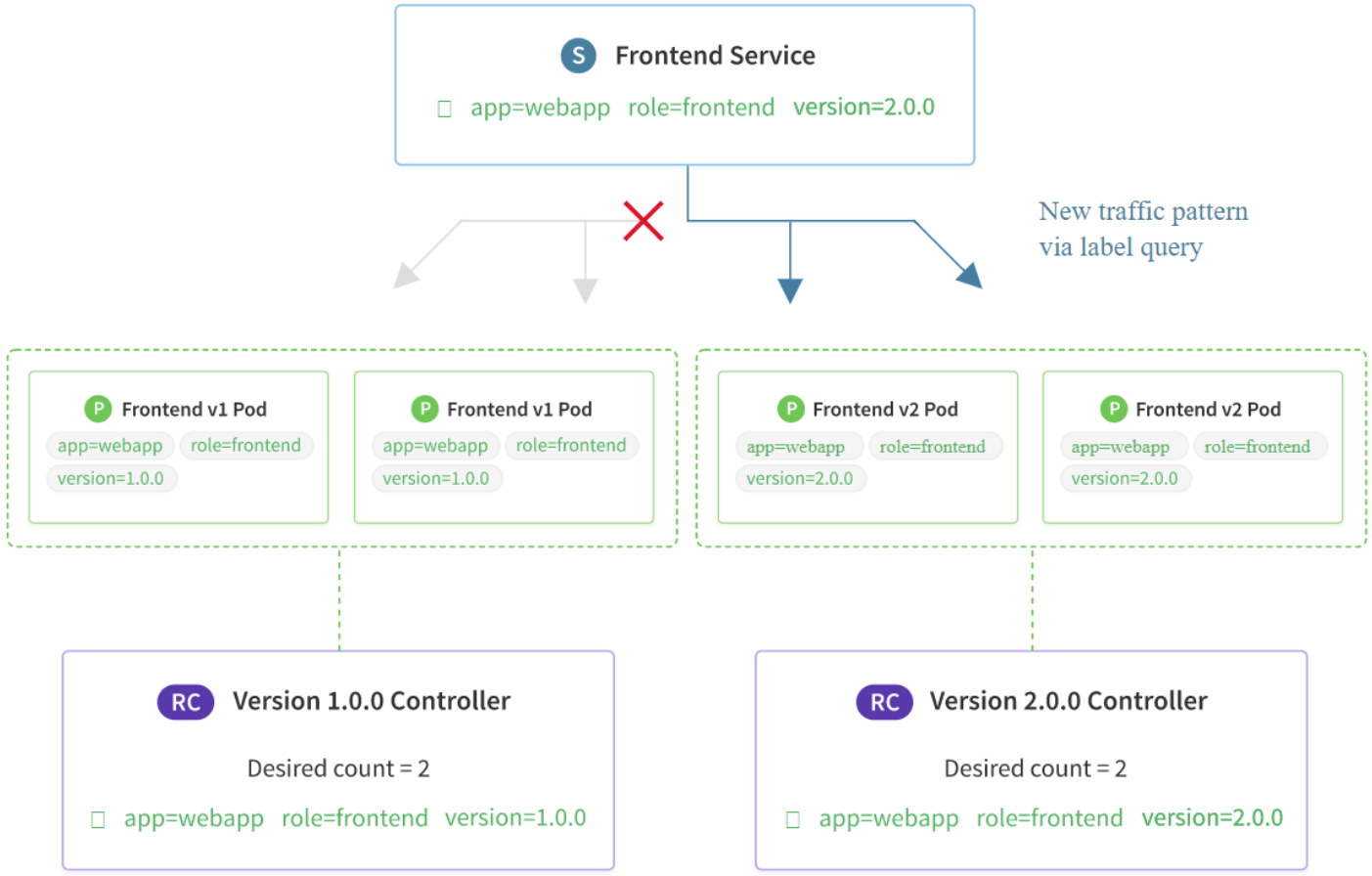
Services – Load balancing



Services - Rolling Deployment



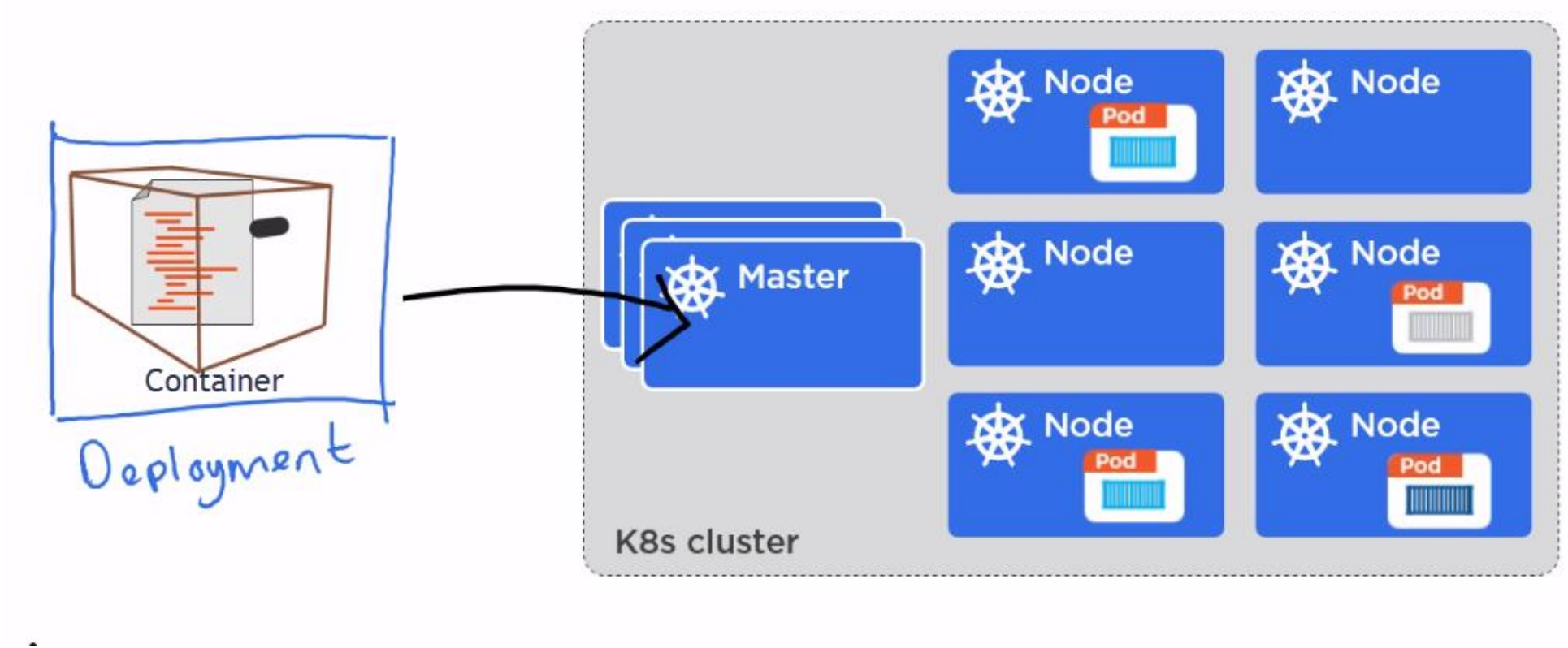
Services - Traffic Shift Deployment



Deployments

- Deployment controller provides declarative updates for Pods and ReplicaSets
- Describe the desired state in deployment configuration.
- Deployment controller changes the actual state to the desired state at a controlled rate.

Deployments



Secrets

- Secrets are intended to hold sensitive information, such as passwords, OAuth tokens, and ssh keys.
- Putting this information in a secret is safer and more flexible than putting it verbatim in a pod definition or in a docker image.
- Suitable while accessing private repositories – store repository login credentials.

Create secrets

kubectl create secret docker-registry NAME --docker-username=username --docker-password=password --docker-email=email [--docker-server=string]

Example

kubectl create secret docker-registry *mysamplekey* **--docker-server**=synacr.azurecr.io

--docker-username=acruuser **--docker-password** = 5//b=aG3Oo+/+ZBQhQSXCc=FEHA/Fv9w

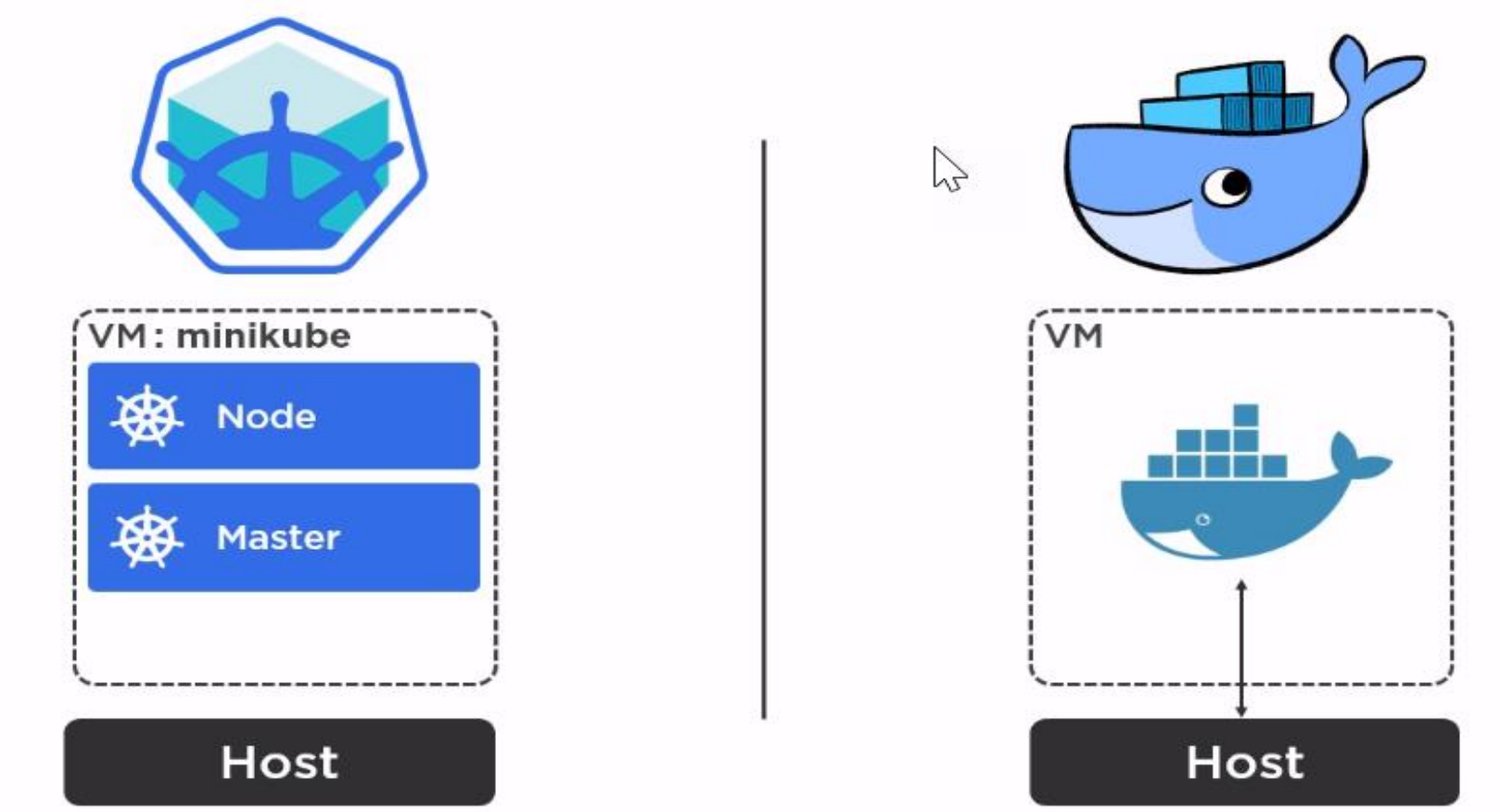
--docker-email = username@mail.com

Install, Configure Kubernetes

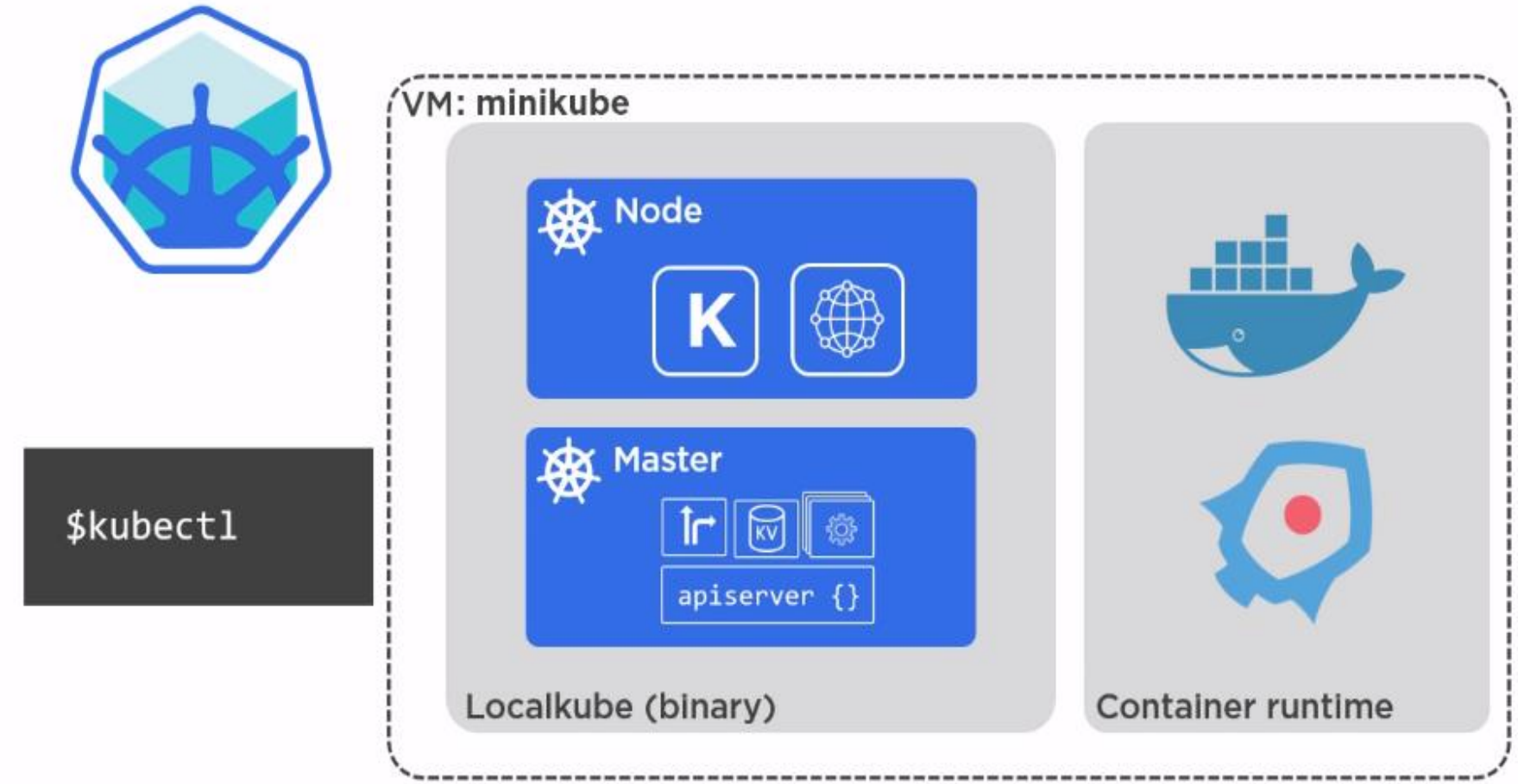
Kubernetes

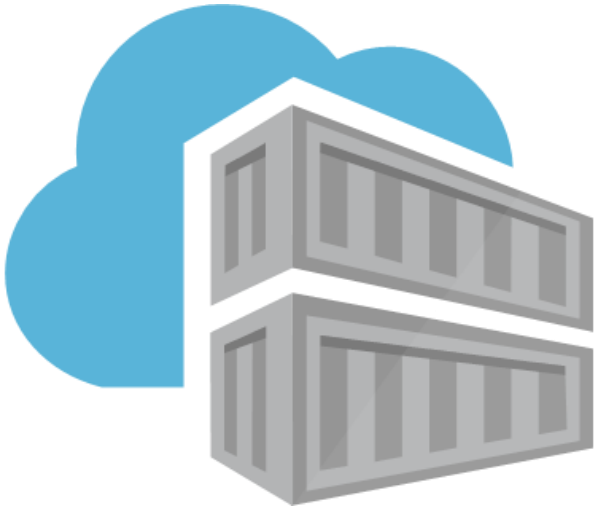
- Kubernetes can be installed and configured to manage clusters that runs Local machine, Cloud or on-premise.
- Local machine
  - Minikube - Tool that makes it to run Kubernetes locally.
- Cloud
  - Google Container Engine (GKE)
  - Azure Container Services (ACS), Azure Container Services – AKS

Minikube cluster Vs Docker



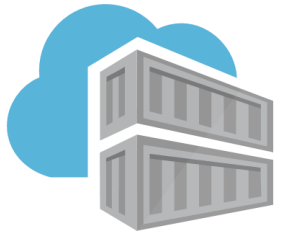
Minikube Architecture





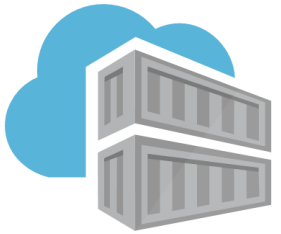
Azure Container Registry

CONTAINER REPOSITORY SERVICE ON AZURE



Azure Container Registry

- Private registry for hosting container images on Azure.
- Used for all container deployment Services
  - Azure WebApp for Containers
  - Azure Container Instance
  - Azure Container Services
- Supported by orchestrators – Kubernetes, Swarm, DCOS
- Push image using Docker Push
- Maintain windows and Linux container images
- Access management with AAD – secrets
- SKU – Basic, Standard, Premium



Push Docker images to ACR

Login to Azure using Azure CLI

az login

Create ACR in resource group

az acr create --resource-group “ResGroup” --name “ACRName” -- sku basic

Enable admin login

az acr update -n “ACRName” --admin-enabled true

Login to ACR to run commands

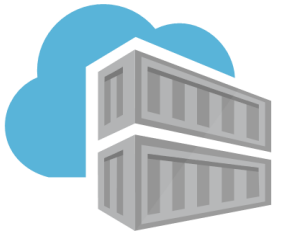
az acr login --name “ACRName”

Tag Docker image with ACR login server name

docker tag <dockernamespace/imagename:tag> <ACR-loginname/imagename:tag>

Upload image to ACR

docker push ACR-loginname/imagename:tag



Working with ACR

Login to ACR to execute Azure CLI commands

List repositories in ACR

az acr repository list -n <acrname> -o table

Display the login server name

az acr show --name <acrname> --query loginServer --output table

Display the ACR credentials

az acr credential show --name <acrname> --query "passwords[0].value"

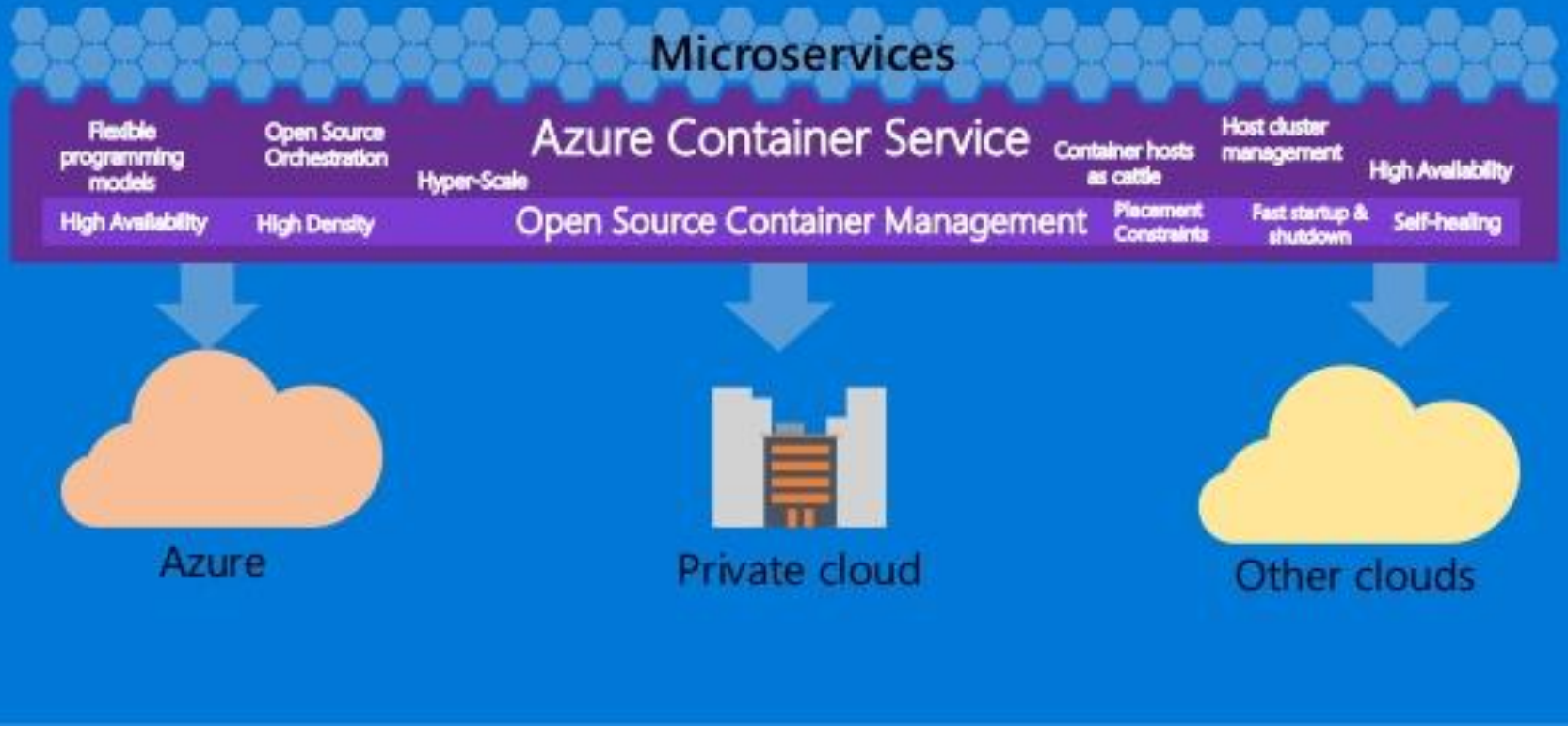
DEMO - ACR

DEPLOYING CONTAINERS USING WEB APP FOR CONTAINERS AND AZURE CONTAINER INSTANCE

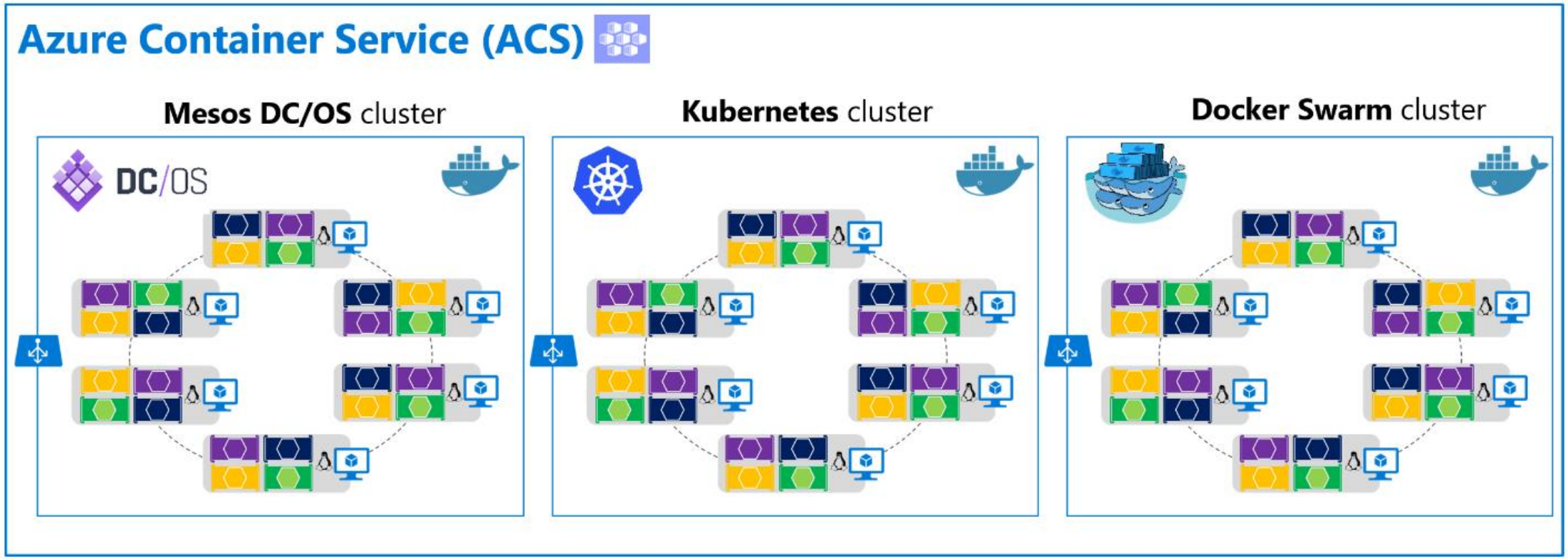
Azure Container Services (AKS)

Azure Container Service

A platform for reliable, hyperscale, container-based applications



Orchestrators for ACS



Azure Container Services (AKS)

- Managed hosted environment for containers
- Kubernetes clusters
- On demand provisioning, upgrading and scaling
- Health monitoring and maintenance by Azure
- Pay only for Agent nodes not for Masters

AKS

AKS Offers

- Automated Kubernetes version upgrades and patching
- Easy cluster scaling
- Self-healing hosted control plane (masters)
- Cost savings - pay only for running agent pool nodes
- Automatic cluster upgrades

AKS using Azure CLI

Enabling AKS preview for your Azure subscription

```
az provider register -n Microsoft.ContainerService
```

Creates Kubernetes cluster

```
az aks create --resource-group myResourceGroup --name myK8sCluster \
    --node-count 1 --generate-ssh-keys
```

Connect to the cluster

```
az aks install-cli
az aks get-credentials --resource-group myResourceGroup \
    --name myK8sCluster
kubectl get nodes
```

Scaling Nodes

```
az aks scale --resource-group=<grp-name> \
        --name=<AKS name> \
        --node-count 5
```

Scaling pods

Manually scale pods

kubectl scale --replicas=<pod-count> deployment/<deployment-name>

Auto-scaling pods

**kubectl autoscale deployment <deployment-name> **

--cpu-percent=50 --min=3 --max=10

List autoscaler status

kubectl get hpa

Update deployments

Update deployments with new version of application image

**kubectl set image deployment <deployment-name> **

<container-name>=<updated-image-name>

Upgrade Kubernetes in Azure Container Service (AKS)

Get cluster versions

```
az aks get-versions --name <aks-name> \
    --resource-group <group-name> --output table
```

```
Name     ResourceGroup    MasterVersion    MasterUpgrades        NodePoolVersion    NodePoolUpgrades
-------  ---------------  ---------------  --------------------  -----------------  --------------------
default  DemoGroup        1.7.7            1.7.9, 1.8.1, 1.8.2   1.7.7              1.7.9, 1.8.1, 1.8.2
```

Upgrade cluster

az aks upgrade --name <aks-name> --resource-group <group-name> --kubernetes-version <desired version>

Eg: az aks upgrade --name bst-cluster --resource-group DemoGroup --kubernetes-version 1.8.2

Thank You


sonusathyadas@hotmail.com


https://in.linkedin.com/in/sonusathyadas


@sonusathyadas