Container Management Platform - Kubernetes

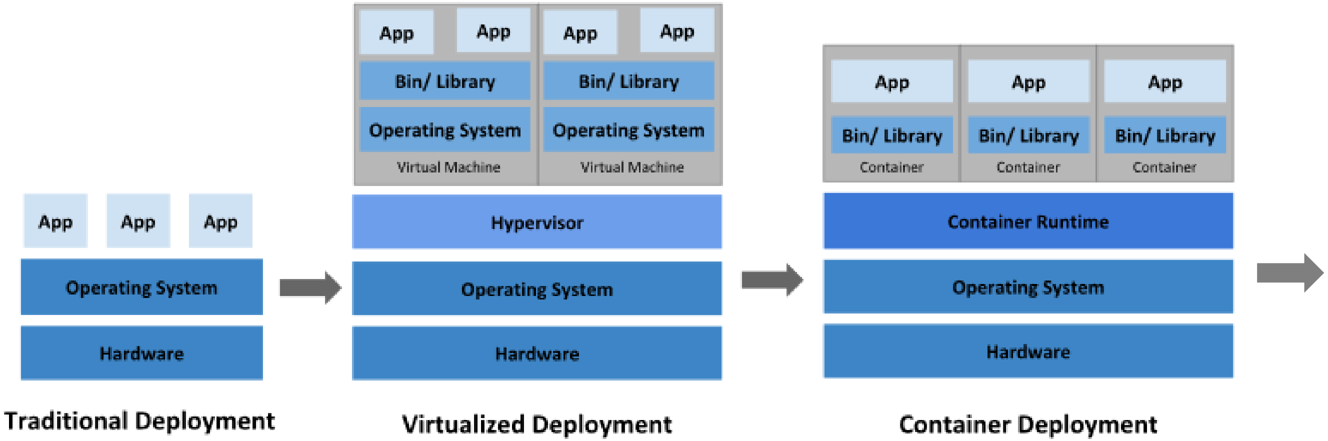
Infosys®
Navigate your next

Need for Container Management Platform

Orchestration	• Orchestration of container deployment and management
Multiple container management	• Need to create the application services that consists of multiple containers
Load Balancing	• Containers shall be scheduled across the cluster to distribute the workload
Auto Scaling	• Scale number of containers(replicas) on-demand for resiliency
Health check	• Health check of deployed containers
Self healing	• Self healing capability and high availability of hosted application services
Zero downtime deployment	• Rolling updates and Zero downtime deployment
micro service application hosting	• Enable micro service application hosting platform
Stateless and Stateful Hosting	• Provide a platform to run both state full and stateless application hosting

Evolution of Application Deployments

Comparing traditional, virtualized, containerized and Kubernetes deployment architectures.



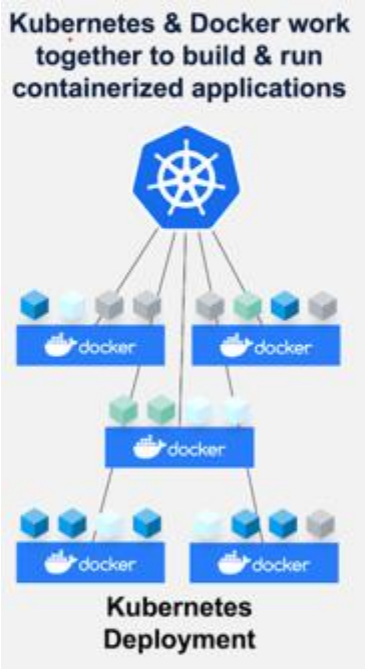

Kubernetes & Docker work together to build & run containerized applications
docker
docker
docker
docker
docker
Kubernetes Deployment

Image Source: https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Origin of Kubernetes(K8s)

History

Initially designed and developed at Google

Originators

Borg and Omega were the primary influence behind Kubernetes technology

How it got the name

The name Kubernetes originates from Greek, meaning helmsman or pilot, and is that the root of governor and cybernetic.

K8s

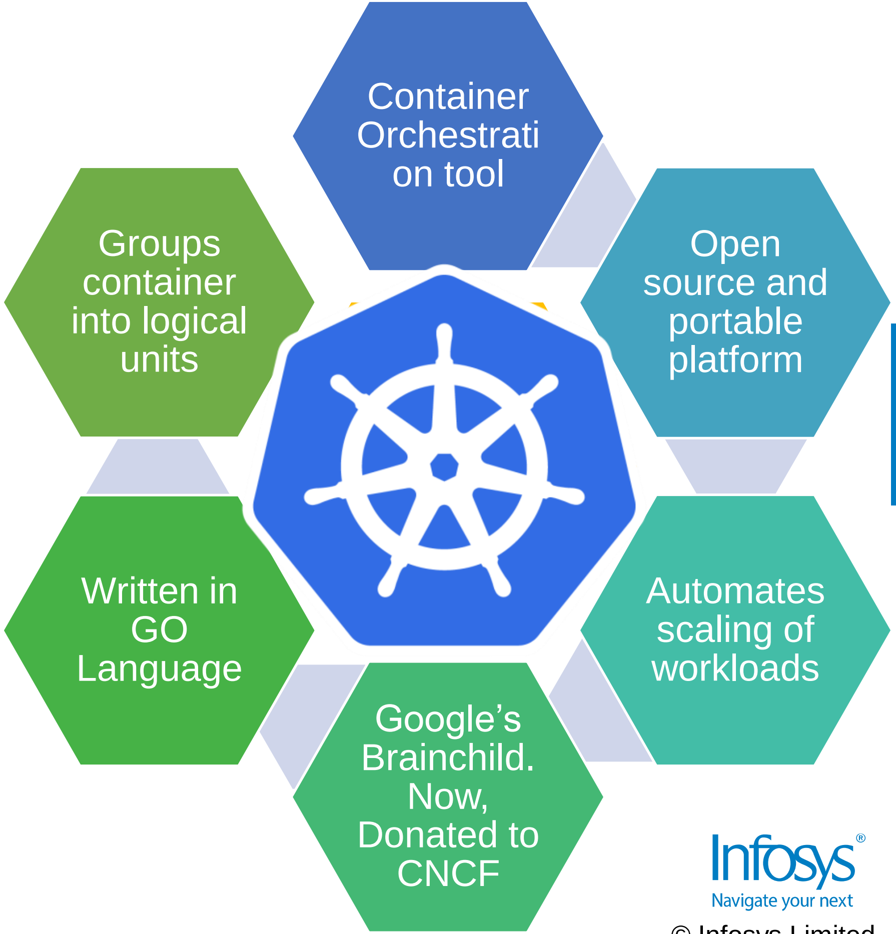
Kubernetes is also called as K8s. K8s is an abbreviation derived by replacing the 8 letters “ubernete” with “8”

Source : https://kubernetes.io/blog/2015/04/borg-predecessor-to-kubernetes/

Infosys®
Navigate your next

What is Kubernetes

- Kubernetes is an open source platform that automates Container deployment and management operations
- Eliminates manual activities in deploying and scaling containerized applications
- By clustering groups of container engines, Kubernetes deploys and manages the containers efficiently
- Clusters can be hosted in public, private or hybrid clouds and possible to run poly-cloud environment as well.



Container Orchestrati on tool

Open source and portable platform

Automates scaling of workloads

Google's Brainchild. Now, Donated to CNCF

Written in GO Language

Groups container into logical units

Infosys® Navigate your next

Kubernetes Components

Master

- The machine that controls Kubernetes nodes.
- Responsible to schedule the containers and manage them.
- High availably is achieved by hosting multiple master nodes.

Node

- Group of container engines which runs the containers
- These machines perform the requested, assigned tasks.
- The Kubernetes master controls them

Pod

- A gaggle of one or more containers deployed to one node.
- All containers running in a pod share the common resource such as IP address, IPC etc.
- Pods abstract network and storage faraway from the underlying container.
- Pod setup helps to move containers round the cluster more easily

Kubernetes(K8S) architecture

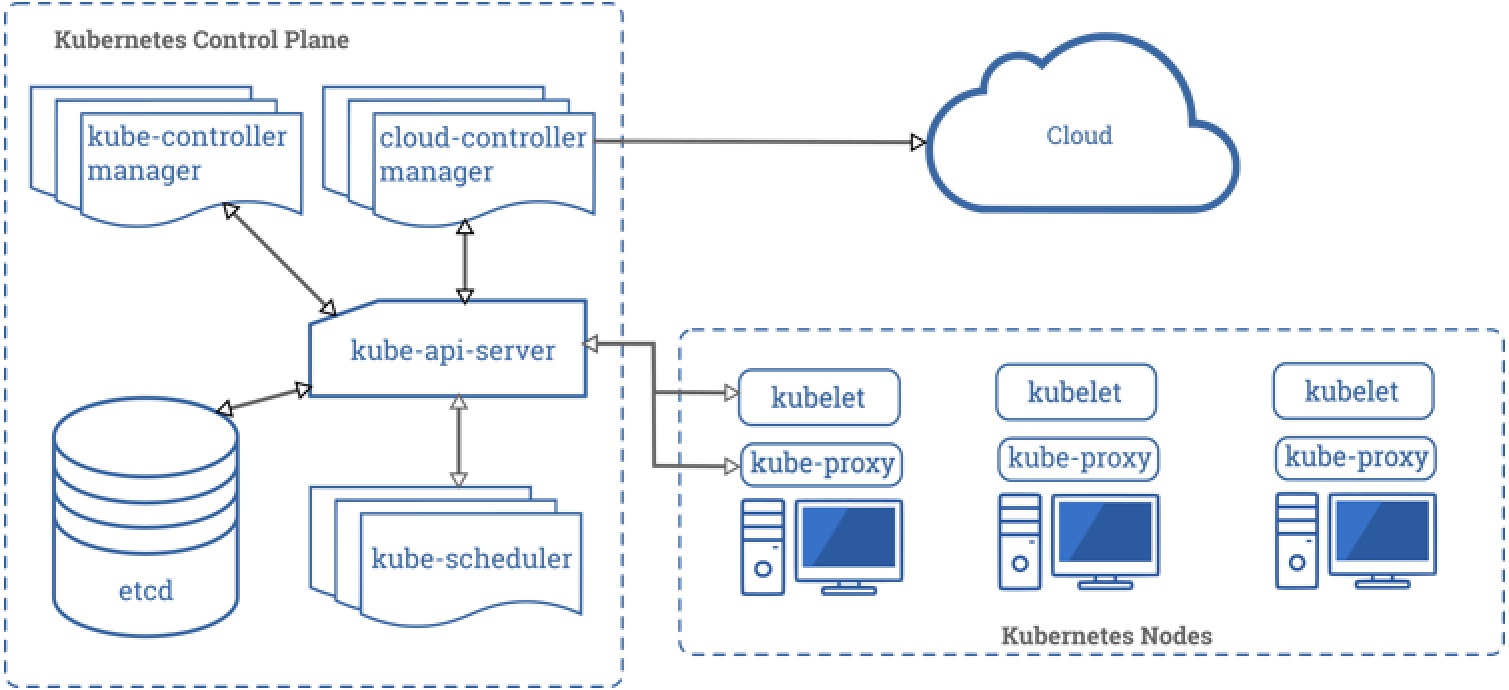


Image Source: https://kubernetes.io/docs/concepts/architecture/cloud-controller/

Master node architecture

Master components

gives full control over Kubernetes cluster and all its components

Scheduler

It decides where in the cluster the workloads are to be run

apiserver:

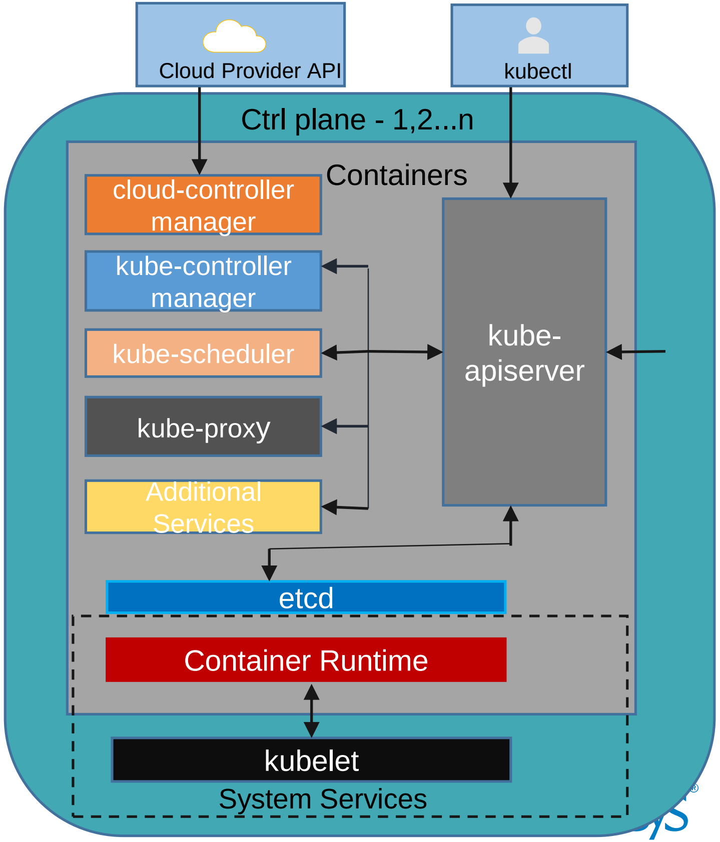
Configures and validates data for api objects like pods, services, replication controllers. Its a front-end of control plane

Etcd

Stores all cluster-related data

Controller

Daemon that embeds core control loops that regulates system state via routine tasks



Navigate your next

Worker node architecture

kubelet

- Primary node agent which performs various tasks like mounting volumes, running containers, etc. for pods assigned to the node

kube-proxy

- Provides service abstraction and connection forwarding

Docker/rkt

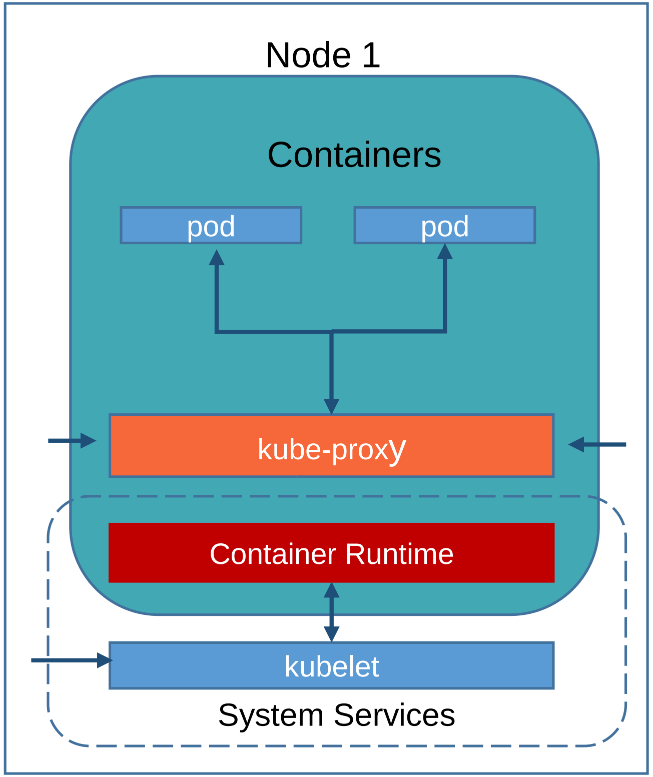
- Container engines for running respective containers

supervisord

- Lightweight process monitor and control system

fluentd

- Daemon which provides cluster-level logging



Namespace

- Namespace is logical way to divide the resources and workloads in a cluster between multiple users.
- Almost all resources like pods, deployments and services are logically grouped into a namespace.
- It provides the way limit as well as restrict access to create, view, or manage resources.

Any given Kubernetes cluster will have below namespaces.

default

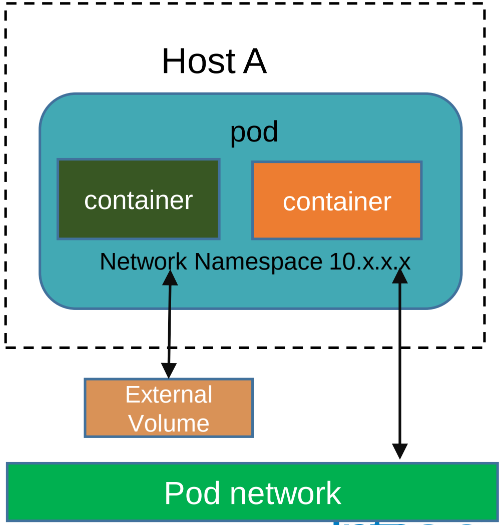
- Resources are created by default when name space is not provided in this namespace
- In smaller environments, default namespace is used to deploy applications without creating any logical separations.
- While interacting with Kubernetes API, such as with issuing kubectl commands to get pods, the default namespace is considered when none is specified.

kube-system

- System namespace is where kubernetes core resources runs
- Hosts network features like DNS and proxy and kubernetes dashboard.
- Ideally we don't deploy any other applications in this namespace.

kube-public

- This namespace is typically not used,.
- Used for run services to make it available to the entire cluster



Kubernetes Objects

- Objects represent the state of a cluster
- It’s used to set desired state of a cluster
- Kubernetes API is used to create, modify or delete an object
- Each object has two main fields in its configuration: spec and status
- Spec describes the desired state of the object and is set by the user
- Status describes state of the object. It is provided and updated by Kubernetes

Pods in Kubernetes

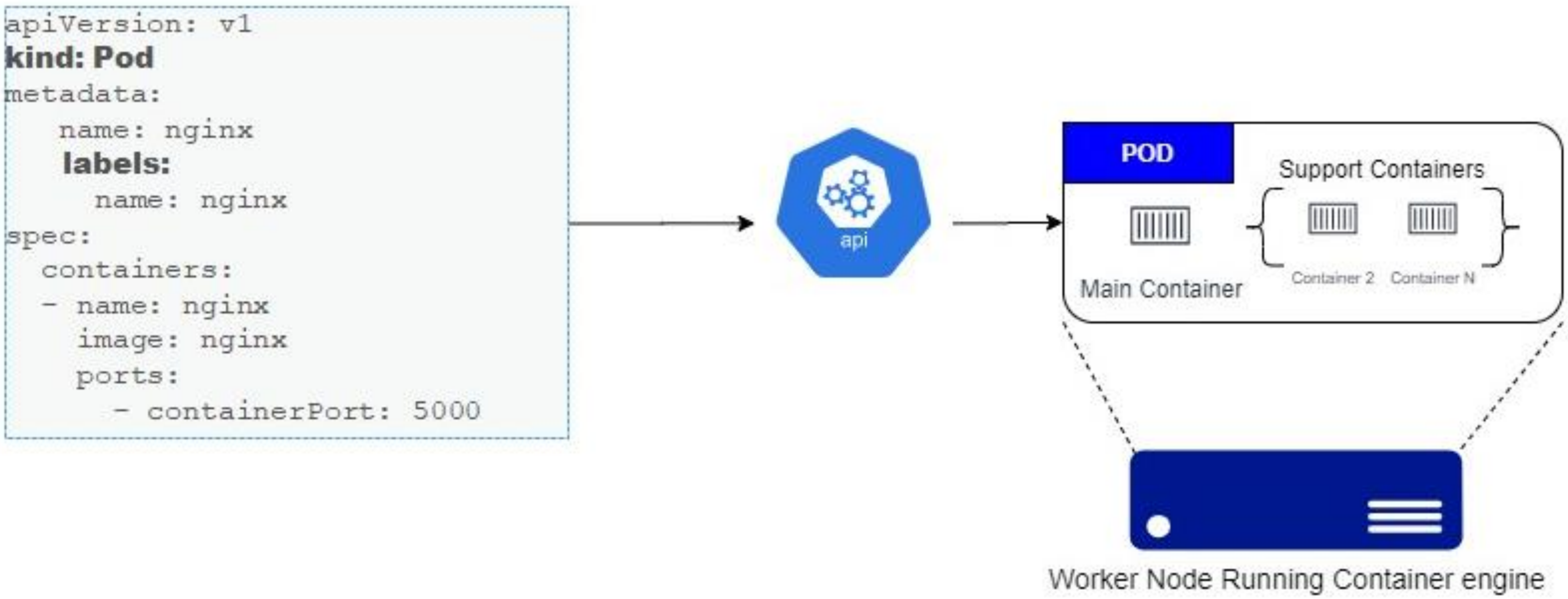
- Smallest deployable computing units that can be created and managed in Kubernetes
- It is a Kubernetes abstraction representing a group of one or more application containers that are relatively tightly coupled
- Containers share an IP address and port space
- Containers have access to shared volumes. They can be mounted on each container in the Pod
- They cannot be moved across nodes

```
apiVersion: v1
kind: Pod
metadata:
   name: nginx
   labels:
     name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 5000
```

Infosys®
Navigate your next

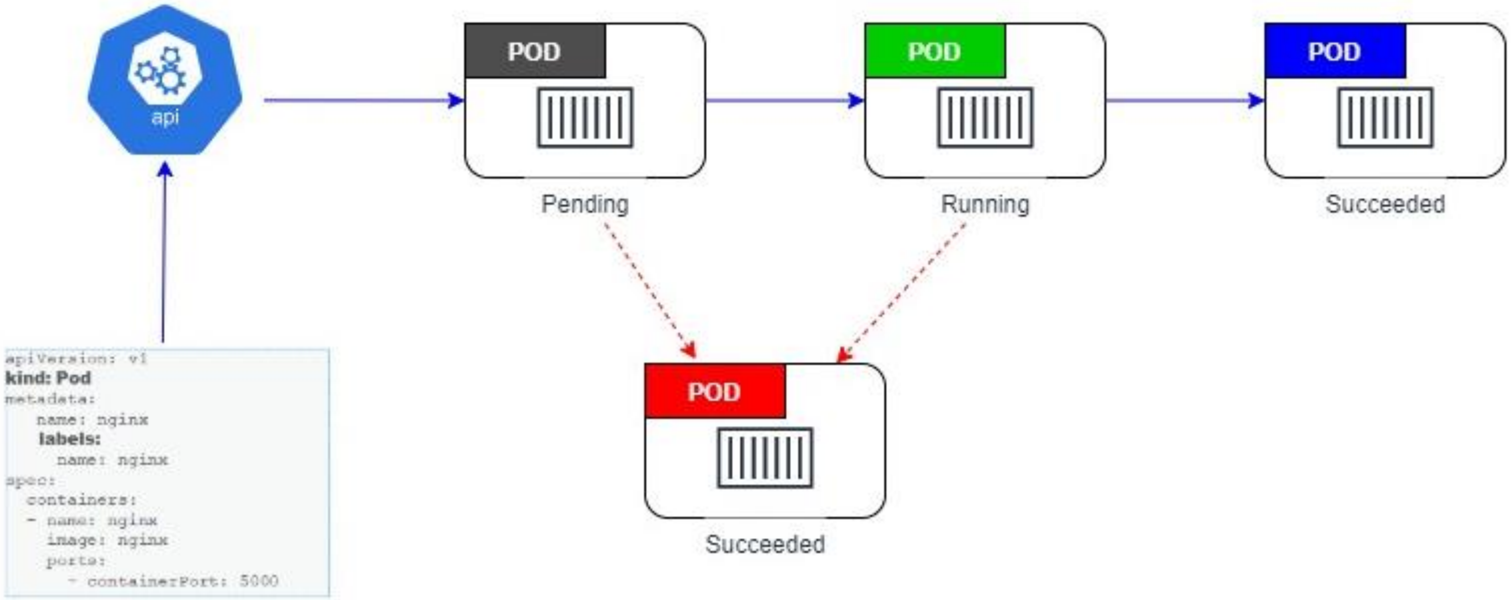
How do we deploy Pods

Define Pod in a manifest → POST manifest to API server → Schedule Pod on cluster



```
apiVersion: v1
kind: Pod
metadata:
   name: nginx
   labels:
     name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 5000
```

How do we deploy Pods – Pod States

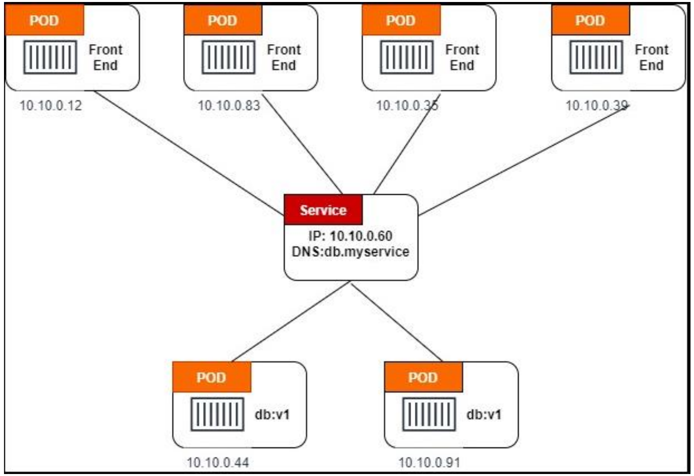


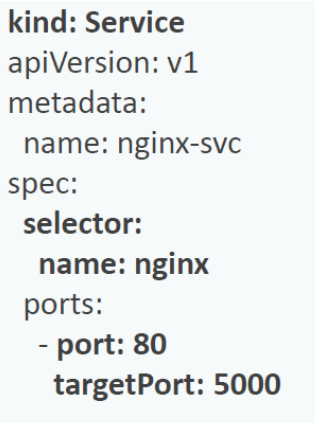
```
apiVersion: v1
kind: Pod
metadata:
   name: nginx
   labels:
     name: nginx
spec:
  containers:
  - name: nginx
    image: nginx
    ports:
      - containerPort: 5000
```

Reliable Network Endpoint - Services

Service is an abstraction which defines a logical set of Pods along with policies with which to access them

- When Pods are destroyed, cannot be brought back. Which leads to issues with dependency
- Service in Kubernetes aims to solve the dependency
- For each Service, On every node, kube-proxy configures the IPtables rules to capture the traffic for its ClusterIP and forwards it to one of the endpoints.
- When a service is removed, kube-proxy removes the IPtables rules on all nodes as well.



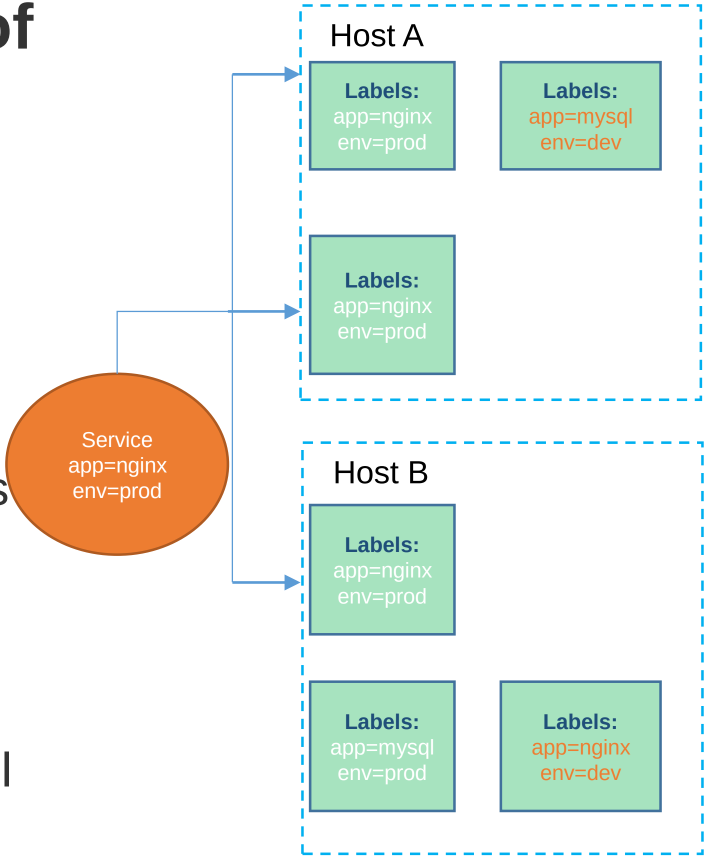


```
kind: Service
apiVersion: v1
metadata:
  name: nginx-svc
spec:
  selector:
    name: nginx
  ports:
    - port: 80
      targetPort: 5000
```

Publishing Services and discovery of associated pods

- Discovering services relies on its integrated DNS service (i.e. CoreDNS or Kube-DNS)
- Label plays a crucial role in discovering the associated pods.
- CoreDNS or Kube-DNS keep track of the DNS records of services as well as its associated pods
- DNS record allows applications to access other pods and services in the cluster via simple consistent naming scheme.

E.g : service with label app=nginx and env=prod will route all traffics to pods which has same labels



Service Discovery and Types

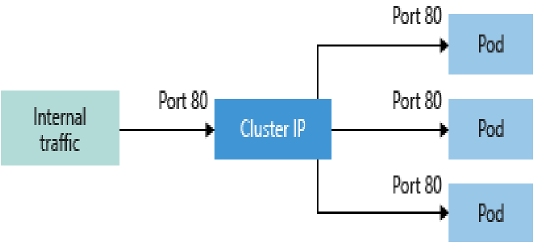
Service Discovery - Two primary modes of finding a Service:

- Environment variables - Every active Service will have a set of environment variables
- DNS - DNS server creates a set of DNS records for each new Service

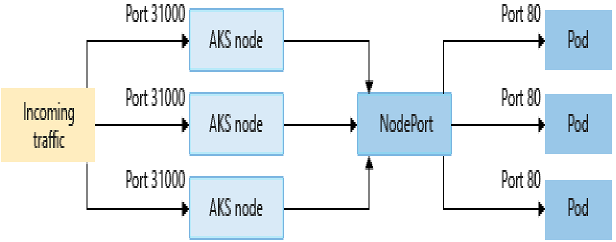
Service Types - When an application needs to be exposed as Service there are three different service type

1. ClusterIP: Exposes Service on cluster’s IP address. It will be reachable from within the cluster
2. NodePort: Exposes Service on each Node’s IP on a static port. It is accessible from outside the cluster
3. LoadBalancer: Exposes Service externally using cloud provider’s load balancer

Cluster IP



NodePort



LoadBalancer

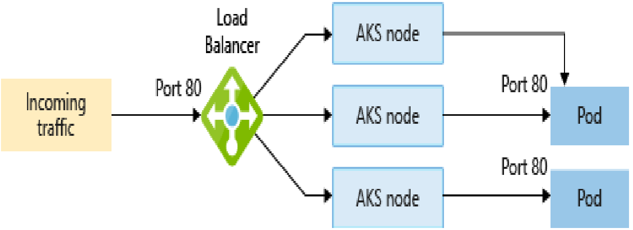
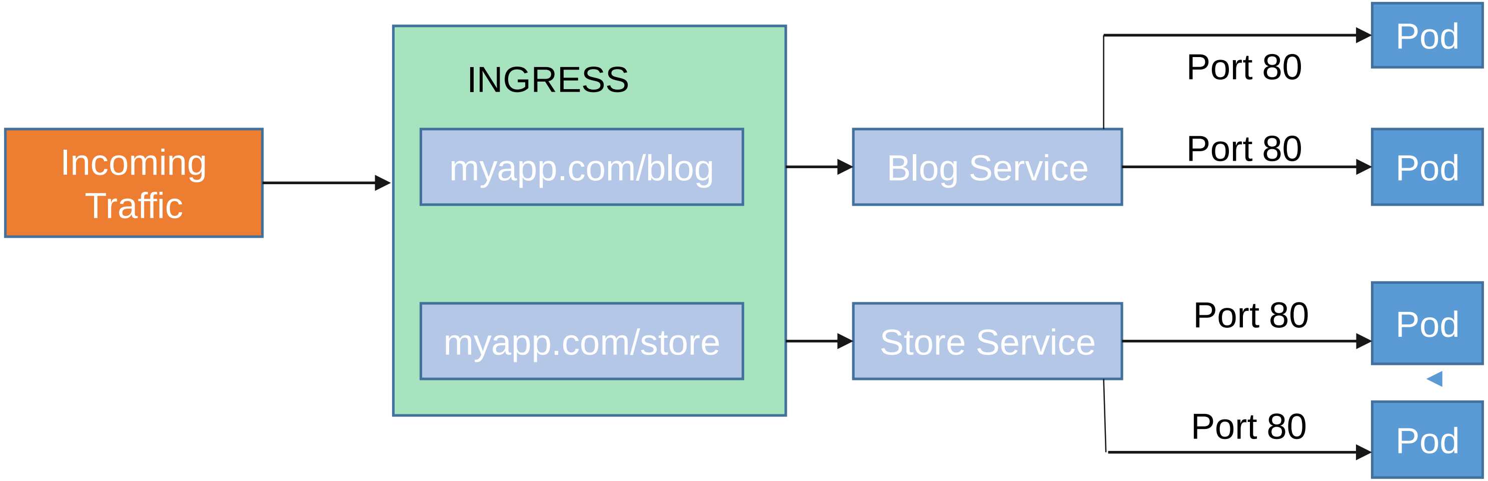


Image Source: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/microservices/aks

Need of Ingress Controller

- Service Type LoadBalancer utilizes Cloud service provider load balancer resource.
  e.g : AWS ELB will be created when service type is configured as LoadBalancer.
- Load balancer is configured to distribute traffic to the pods in your Service on a given port.
- LoadBalancer service only works at layer 4
  - Service is unaware of the actual applications
  - Can't make any additional routing considerations.

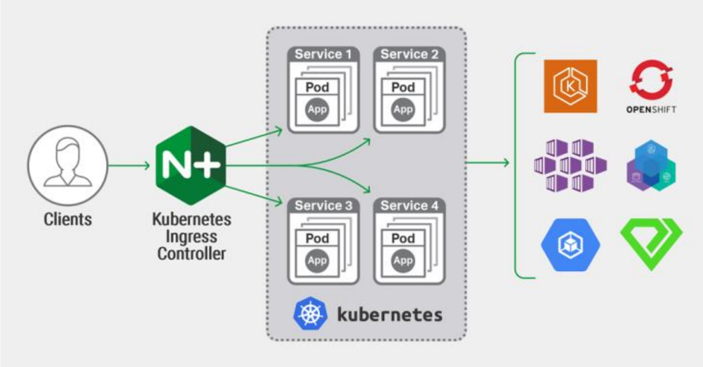
Ingress controllers works on layer 7, and may use more intelligent rules to distribute application traffic.



Ingress Controller in Kuberntes

- In Kubernetes, We cab deploy 3rd party Ingress controllers like NGINX, Traefik, HA-Proxy etc.
- We can leverage Cloud service provider Layer 7 Load balancer as ingress controller i.e AWS and Azure Application Load balancers.
- By enabling HTTP application routing in a Kubernetes cluster, we can utilize the Ingress controller and an External- DNS controller.
- When an ingress resources are created , DNS A records are created in a cluster-specific DNS zone.

E.g. : In our hands on lab setup, we have deployed a Traefik ingress controller to route the traffic and Leveraged the Route 53 for external DNS A record



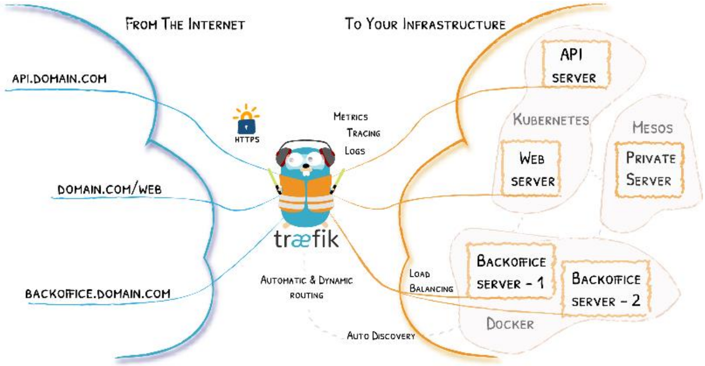


Image Source: https://www.nginx.com/blog/announcing-nginx-ingress-controller-for-kubernetes-release-1-5-0/ and https://docs.traefik.io/v1.0/basics/

Storage options for Stateful Applications in K8s

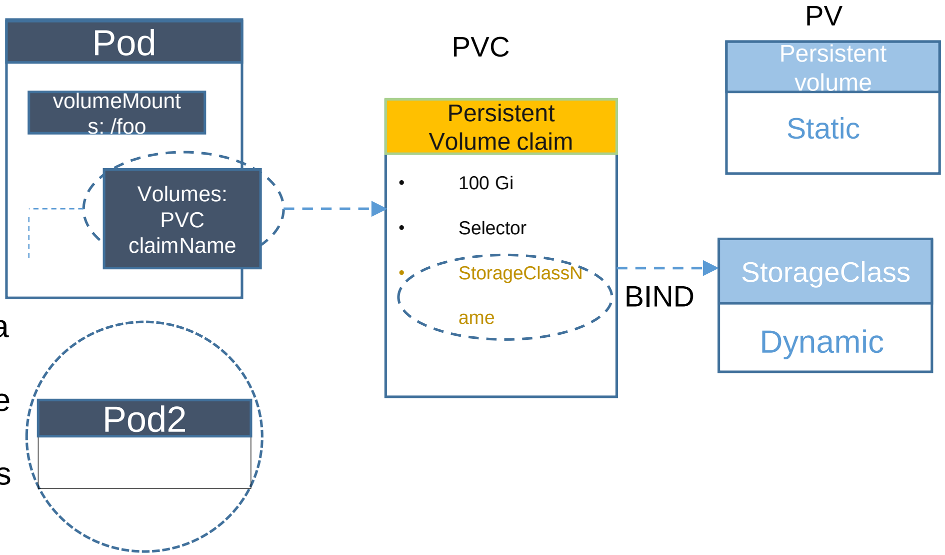
Persistence volume(PV) and Persistence Volume Claim(PVC) is the mechanism to provision external storage like cloud storage, SAN or NAS on kubernetes clusters to run stateful applications.

Here are the few scenarios where persistence storages required.

- Applications which run on Kubernetes cluster may need to store and retrieve data.

  e.g : container hosting database for an application in kubernetes cluster
- Workloads require storage that persists on more regular data volumes within the Hosted platform.

  e.g. : Jenkins running on kubernetes shall have persistence storage to store configuration, jobs, logs etc.
- Multiple pods may need to share the same data and volumes
- Need to reattach the data volumes if the pod is rescheduled on a different node.



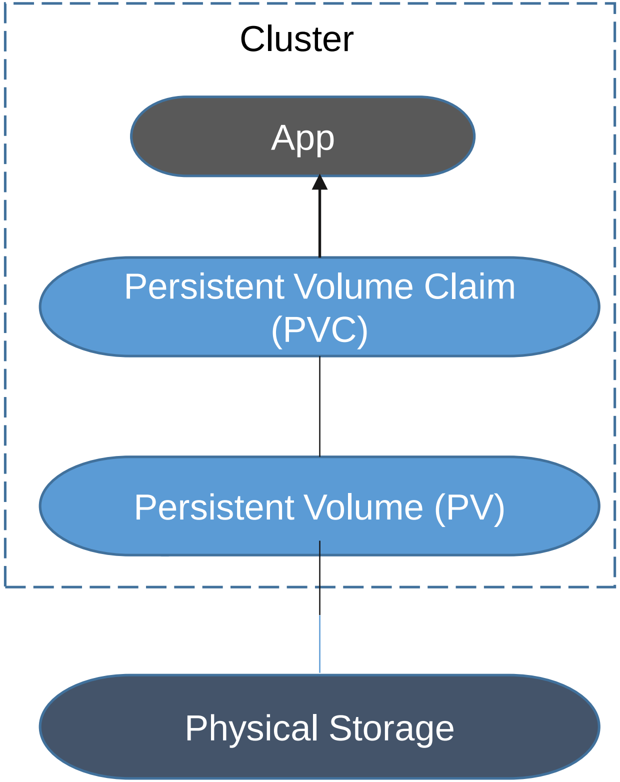
Persistent Volume and Persistent Volume Claim

Persistent Volume (PV) is a storage resource which is created and managed by the Kubernetes API using external storages and it retained beyond the lifetime of a pod.

- We can create PV based on the available storage of the host, remote SAN.
- If the cluster is hosted in Cloud platform, we can leverage the Using Disks or Files services offered can be used to create the PV

PersistentVolumeClaim(PVC) requests either Disk or File storage of a particular StorageClass, access mode, and size.

- A PersistentVolume is bound to a PersistentVolumeClaim once an available storage resource has been assigned to the pod requesting it.
- The pod definition includes the quantity mount once the quantity has been connected to the pod.



Storage Classes

StorageClass provides how for administrators to explain the “classes” of storage they provide . Different classes might map to quality-of-service levels, or to backup policies, or to arbitrary policies determined by the cluster administrators.

- A Persistent Volume are often statically created by a cluster administrator, or dynamically created by the Kubernetes API server.
- Storage class helps to determine the type of backend storage to be selected when cluster is hosted in Cloud platform.
- Popular Cloud service platforms and corresponding class names
  - Azure – Managed and Un managed, Premium , standard and Slow disks
  - AWS – Too many EBS offerings . Default is gp2 and others are io1, gp2, sc1, st1
  - GCP - pd-standard or pd-ssd (Standard and SSD disks)
- StorageClass also defines the reclaim Policy which helps to retain the data despite of deleting/ destroying the application.

K8s Deployment concepts

Deployment:

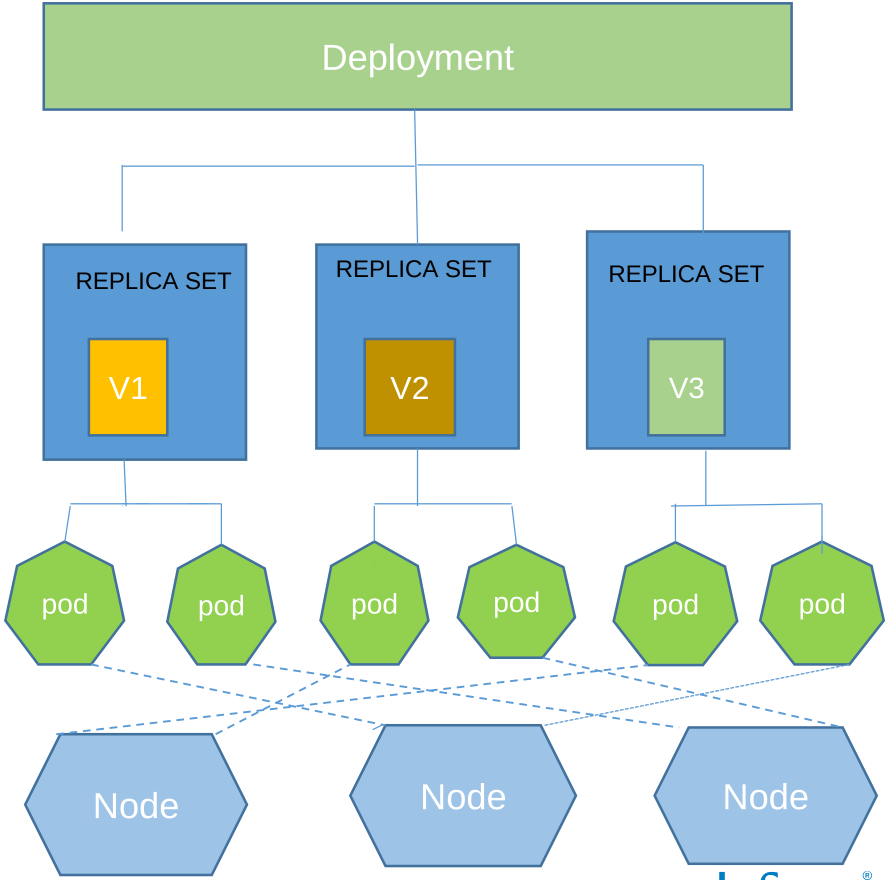
- Deployment controller changes the particular state to the specified state. Deployments manage your updates.

ReplicationController

- It ensures a specified number of Pod replicas are running at any one time

ReplicaSet

- Same as ReplicationController except that it has selector support. A replica set manages a group of different pods selected based on a common label.



Deployment concepts

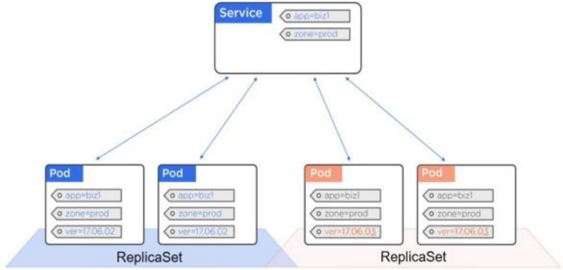
Deployment consists of one or more identical pods running across nodes.

Deployment defines the amount of replicas (pods)

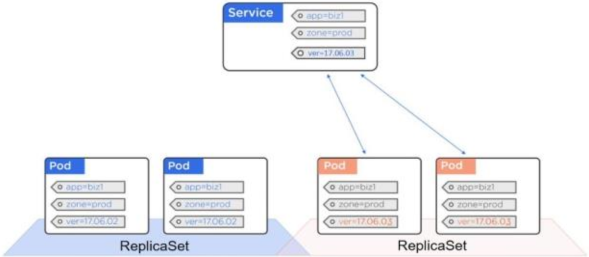
Kubernetes Deployment Controller manages the pod deployment based on the manifest and replicas defined.

If pods or nodes encounter problems, Kubernetes Scheduler deploys additional pods on healthy nodes to replace the un-healthy pods.

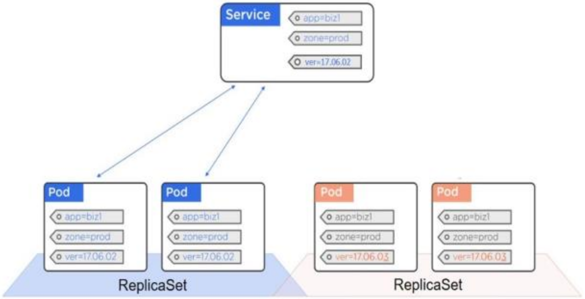
Stateless applications shall use deployment model instead of scheduling individual pods.



Deployed a new version of application



Switching the service to new version



Rollback the service to Old version of application

Use of Deployment

To rollout, ReplicaSet to create Pods and monitor its status

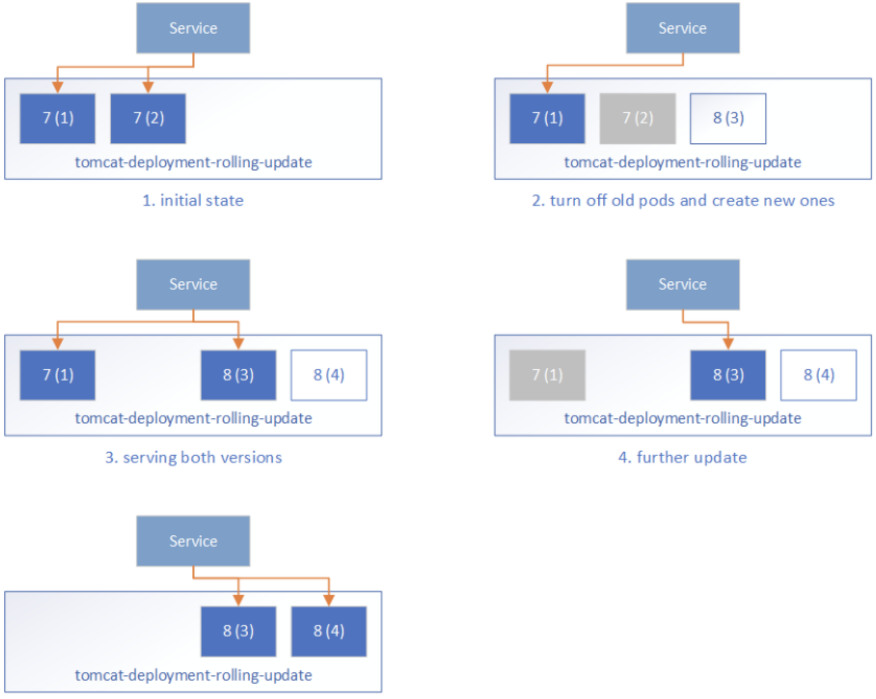
Changing or updating Pod state

Scaling up of the deployment

Roll back to a previous Deployment

Pause Deployment to fix template

Cleaning old ReplicaSets


Service
7 (1)
7 (2)
tomcat-deployment-rolling-update
Service
7 (1)
7 (2)
8 (3)
tomcat-deployment-rolling-update
Service
7 (1)
8 (3)
8 (4)
tomcat-deployment-rolling-update
Service
7 (1)
8 (3)
8 (4)
tomcat-deployment-rolling-update
Service
8 (3)
8 (4)
tomcat-deployment-rolling-update

1. initial state

2. turn off old pods and create new ones

3. serving both versions

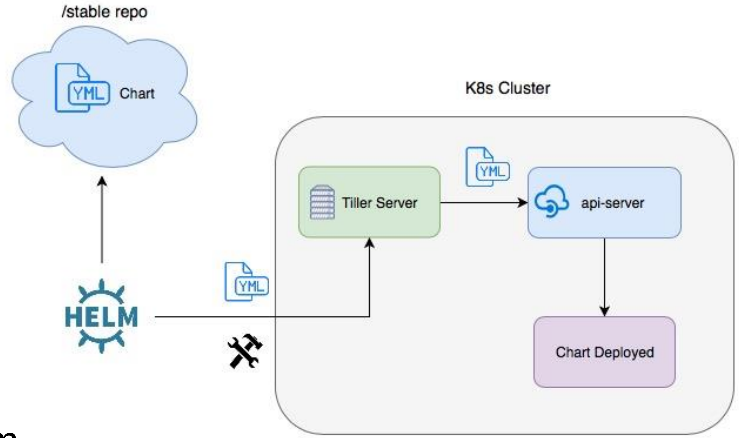
4. further update

5. update complete

Source: https://kubernetes.io/blog/2018/04/30/zero-downtime-deployment-kubernetes-jenkins/

Package management with Helm

- Helm is used as common approach in packaging and managing application deployment.
- Chart is the packaged version which contains application code and manifests to deploy resources.
- We can build custom helm charts to deploy and update the application
- Many open source applications and tools offers helm charts to deploy them in Kubernetes.
- Helm charts can be stored locally, or in remote repositories like Harbor, Nexus, Artifactory etc.
- Cloud providers also provide repo's like Azure's ACR Helm Chart repo.
- In Helm V2, a server component called Tiller is deployed in kubernetes to manage the installation of charts. Where Helm client is installed on local workstation.
- Helm V3 onwards, tiller less deployment is possible which leverages kube config.



Note : Jenkins, Monitoring stack, Kubernetes Dashboard of our hands on lab was created using helm chart.

Image Source: https://supergiant.io/blog/simplifying-app-deployment-in-kubernetes-with-helm-charts/

Commercial Distribution of K8s

RANCHER®

RED HAT® OPENSHIFT Container Platform

Pivotal Container Service™

TECTONIC by CoreOS

PLATFORM9

SUSE CaaS Platform

Azure Kubernetes Service

Amazon EKS

DevOps in OpenShift K8s platform

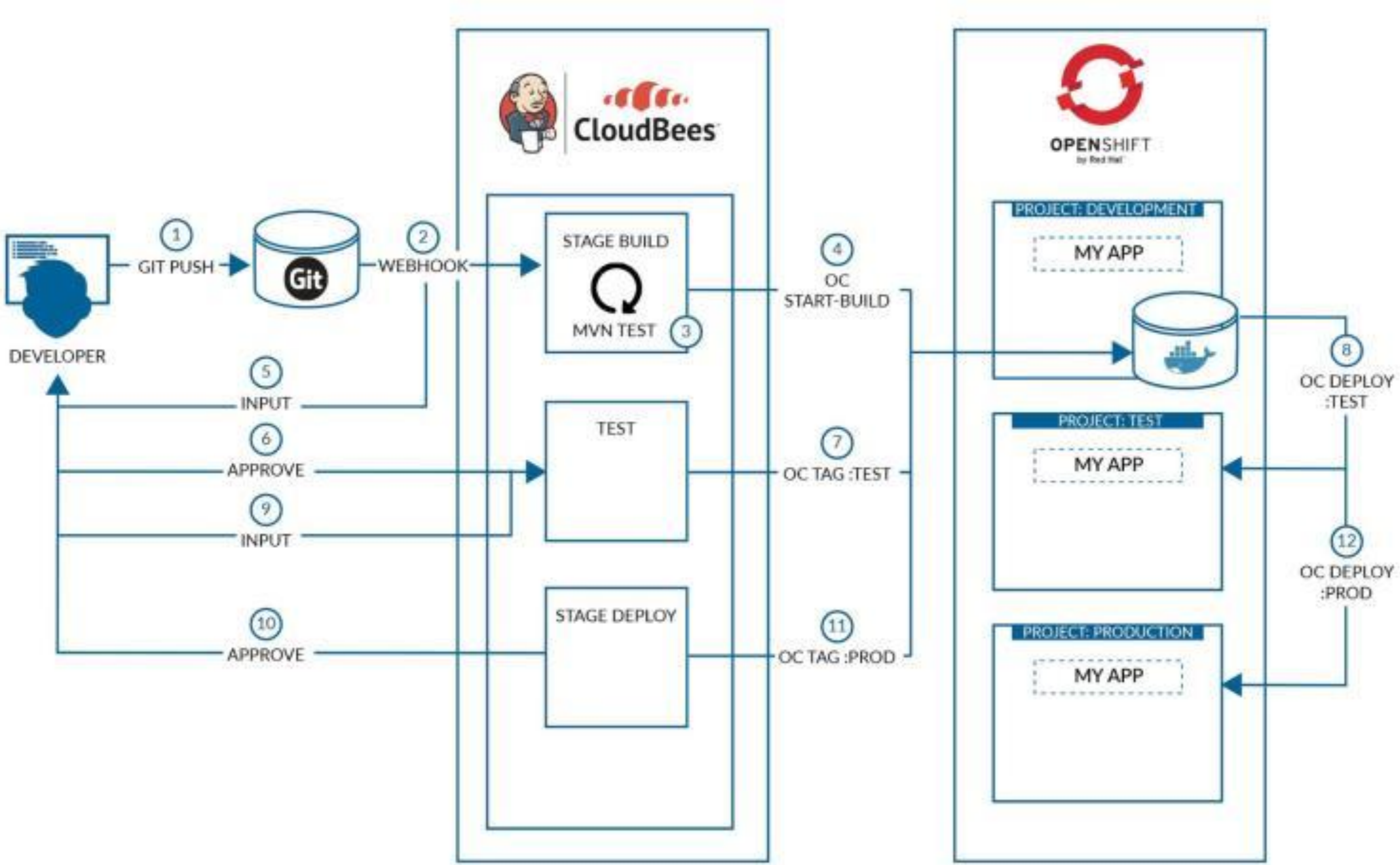


Image Source: https://www.openshift.com/blog/openshift-cloudbees-jenkins-enterprise-devops

Infosys®
Navigate your next

Microservices on Azure Kubernetes Services(AKS)

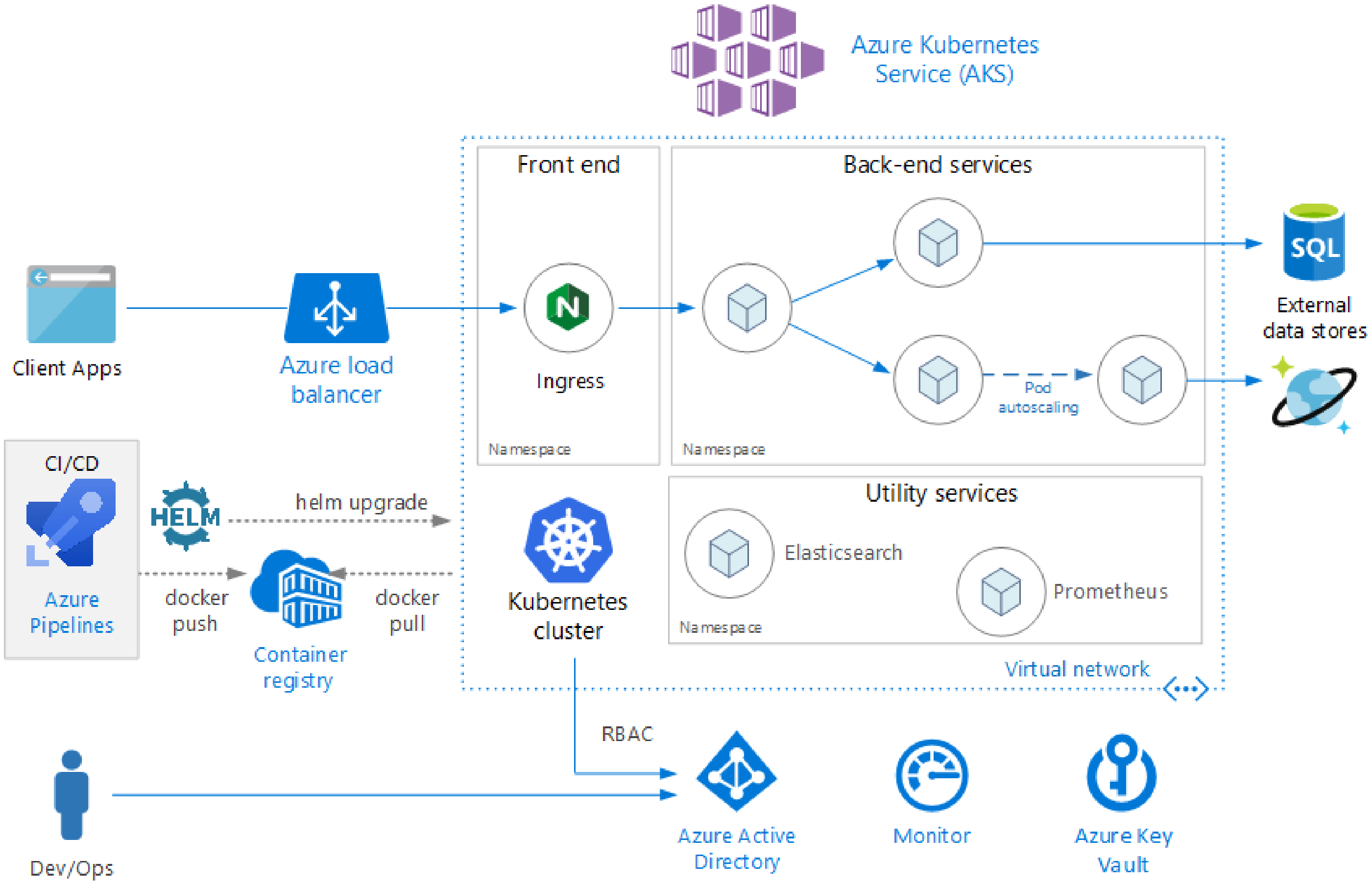
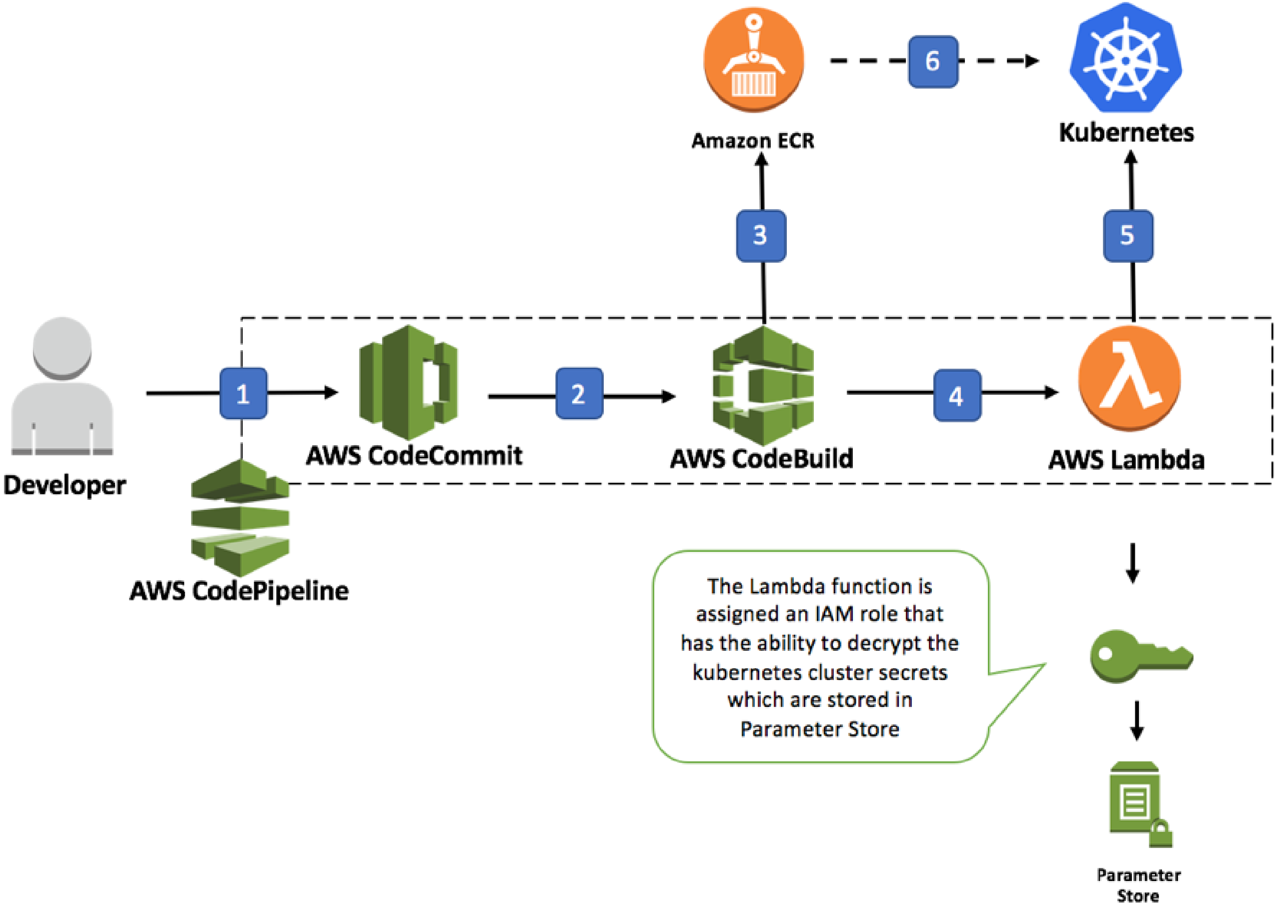


Image Source: https://docs.microsoft.com/en-us/azure/architecture/reference-architectures/microservices/aks

Infosys®
Navigate your next

Microservices on K8s - AWS Platform

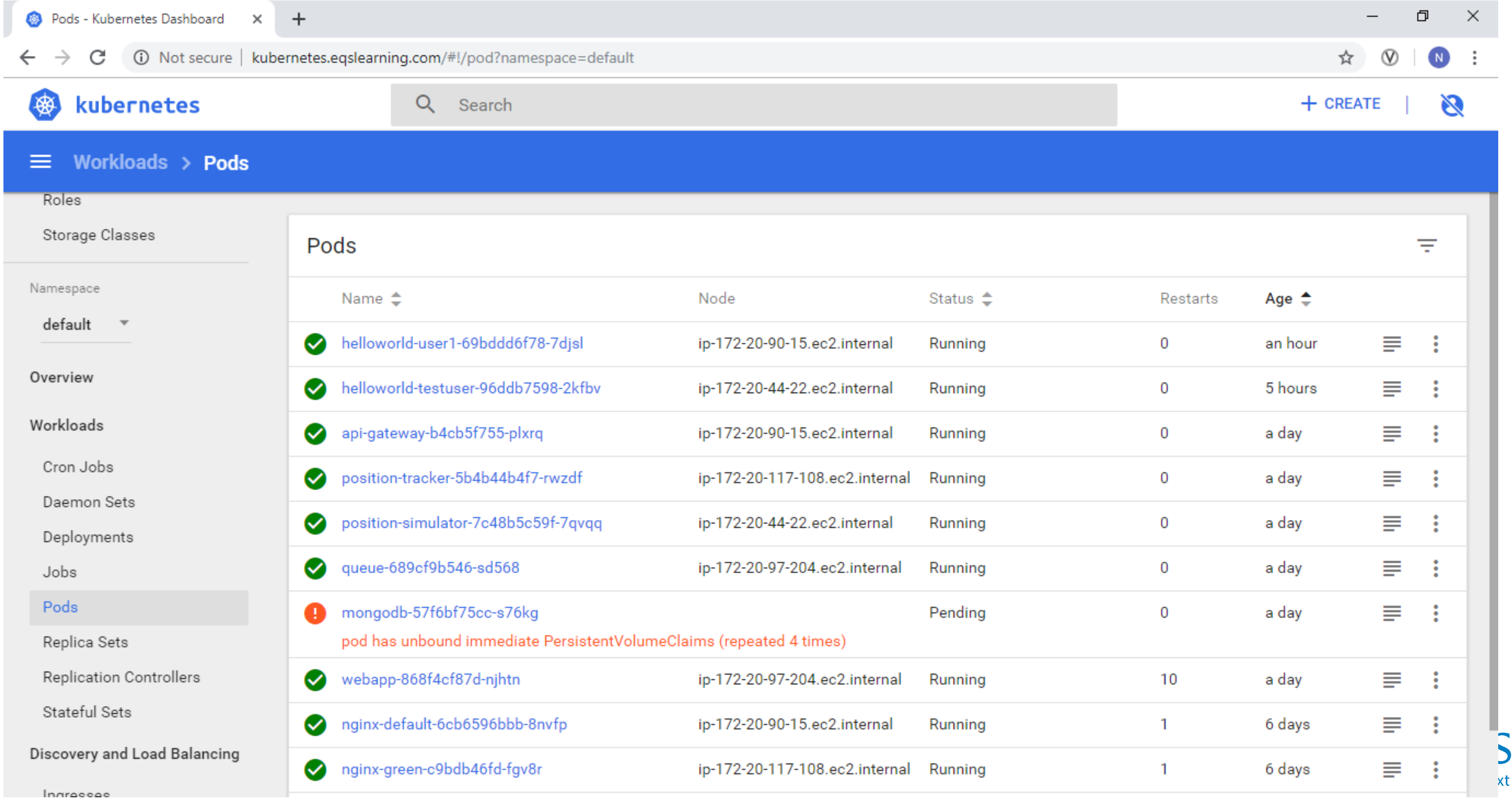


The Lambda function is assigned an IAM role that has the ability to decrypt the kubernetes cluster secrets which are stored in Parameter Store

Image Source: https://aws.amazon.com/blogs/devops/continuous-deployment-to-kubernetes-using-aws-codepipeline-aws-codecommit-aws-codebuild-amazon-ecr-and-aws-lambda/

Navigate your next

Training Lab - kubernetes.eqslearning.com - Demo



Namespace: default

Pods

	Name	Node	Status	Restarts	Age
✓	helloworld-user1-69bddd6f78-7djsl	ip-172-20-90-15.ec2.internal	Running	0	an hour
✓	helloworld-testuser-96ddb7598-2kfbv	ip-172-20-44-22.ec2.internal	Running	0	5 hours
✓	api-gateway-b4cb5f755-plxrq	ip-172-20-90-15.ec2.internal	Running	0	a day
✓	position-tracker-5b4b44b4f7-rwzdf	ip-172-20-117-108.ec2.internal	Running	0	a day
✓	position-simulator-7c48b5c59f-7qvqq	ip-172-20-44-22.ec2.internal	Running	0	a day
✓	queue-689cf9b546-sd568	ip-172-20-97-204.ec2.internal	Running	0	a day
!	mongodb-57f6bf75cc-s76kg pod has unbound immediate PersistentVolumeClaims (repeated 4 times)		Pending	0	a day
✓	webapp-868f4cf87d-njhtn	ip-172-20-97-204.ec2.internal	Running	10	a day
✓	nginx-default-6cb6596bbb-8nvfp	ip-172-20-90-15.ec2.internal	Running	1	6 days
✓	nginx-green-c9bdb46fd-fgv8r	ip-172-20-117-108.ec2.internal	Running	1	6 days

THANK YOU

Call us: 000 - 12345678/ 87/89/09
Email : mail@Infosys.com
Website: www.Infosys.com