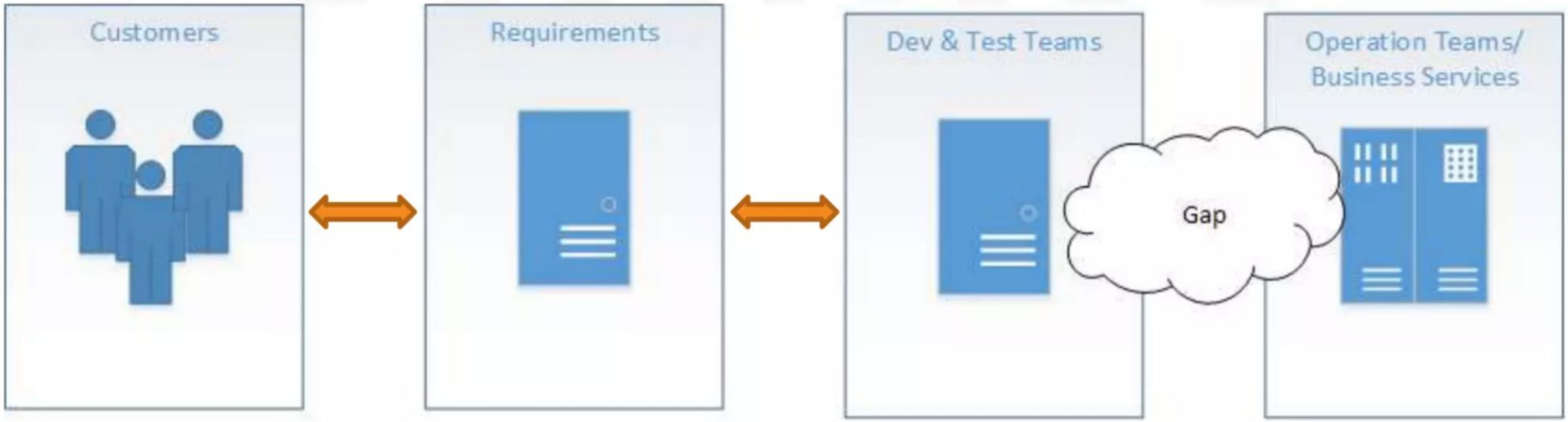
HEPAPI
DIGITAL TRANSITION PARTNER

Kubernetes

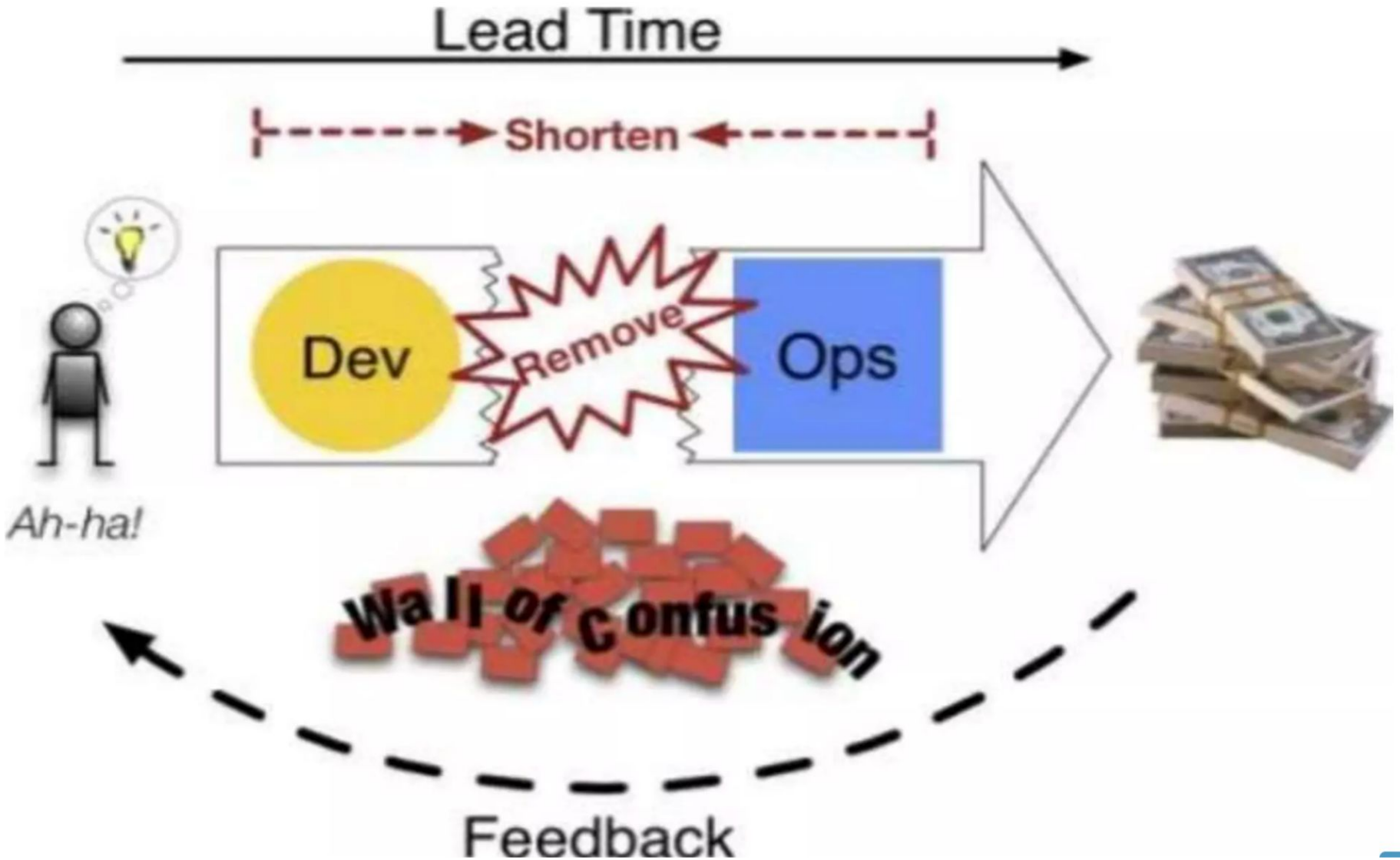
Before DevOps

Why DevOps?

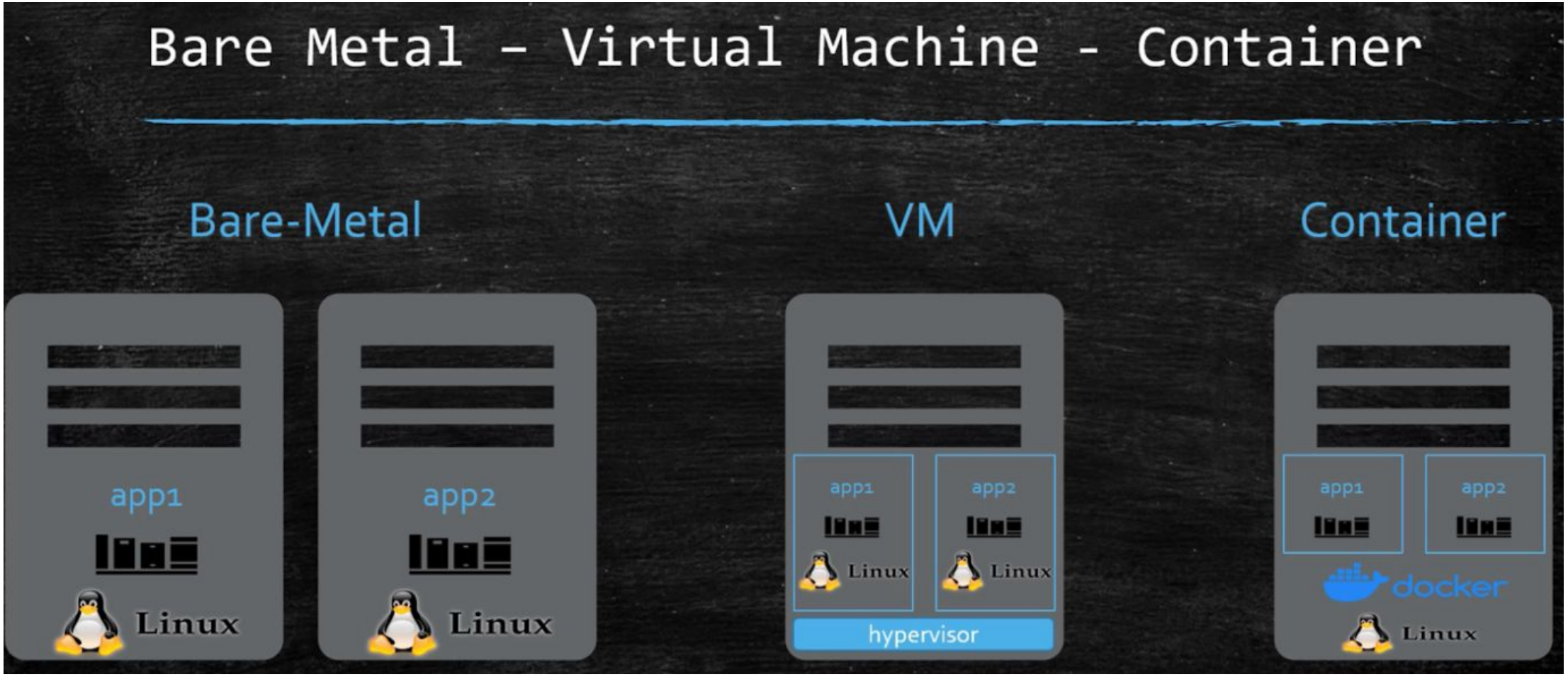
HEPAPI
DIGITAL TRANSITION PARTNER



Why DevOps?

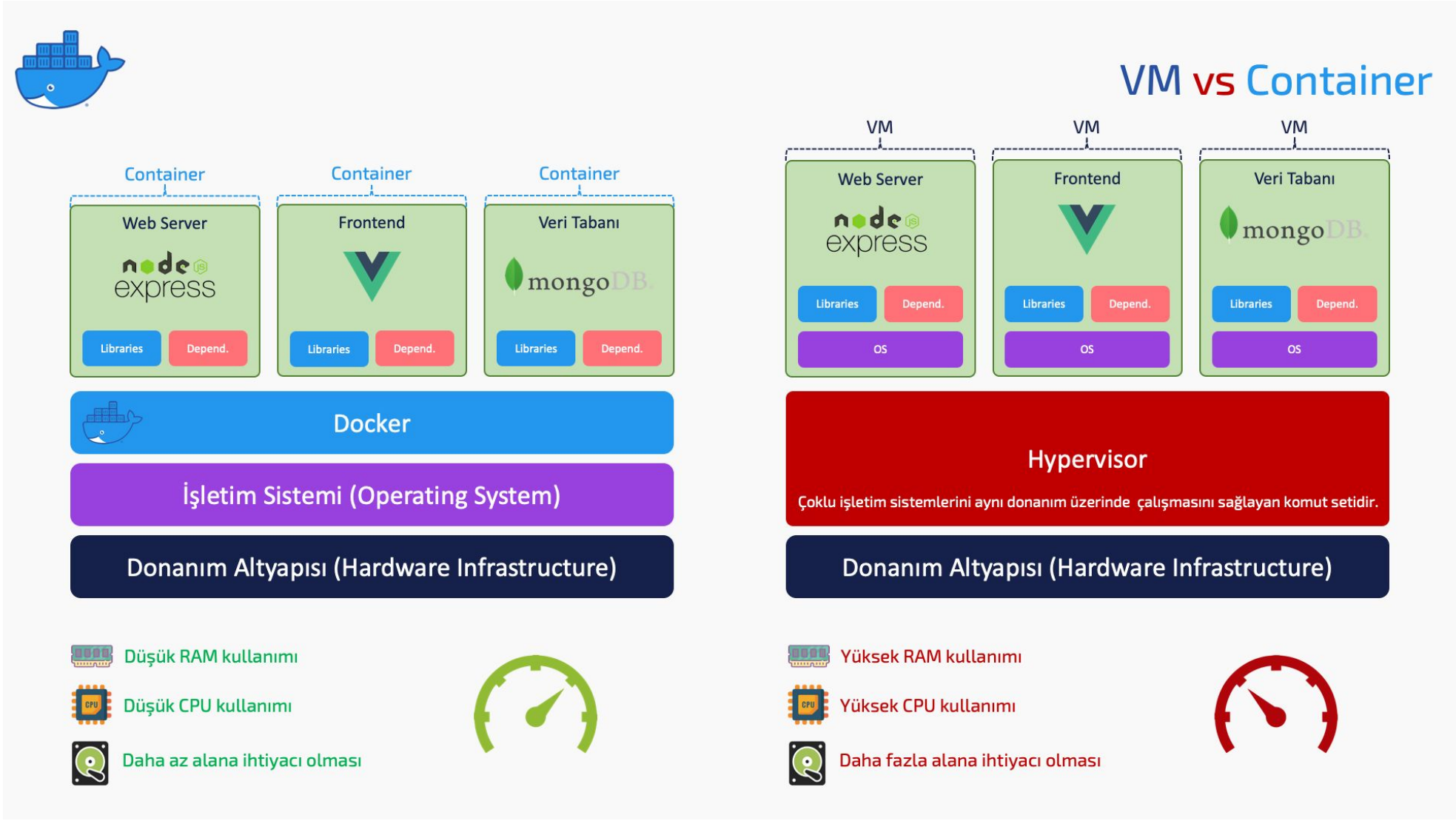

Lead Time
Shorten
Dev
Remove
Ops
Ah-ha!
Wall of Confusion
Feedback

Containerization



Containerization

HEPAPI
DIGITAL TRANSITION PARTNER

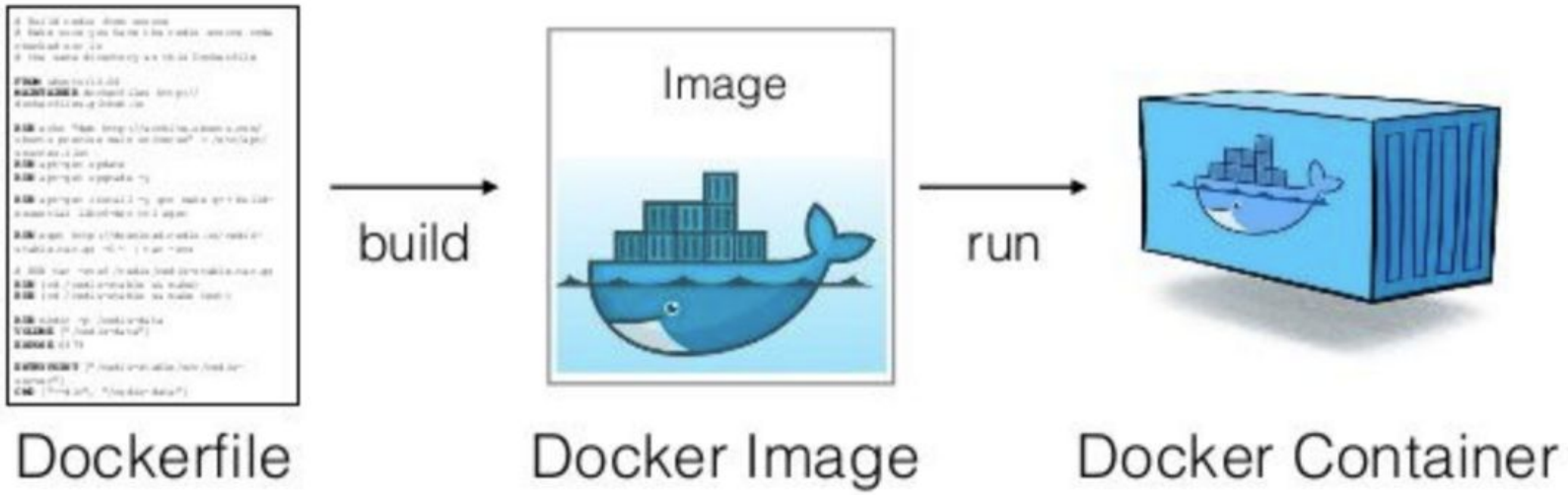


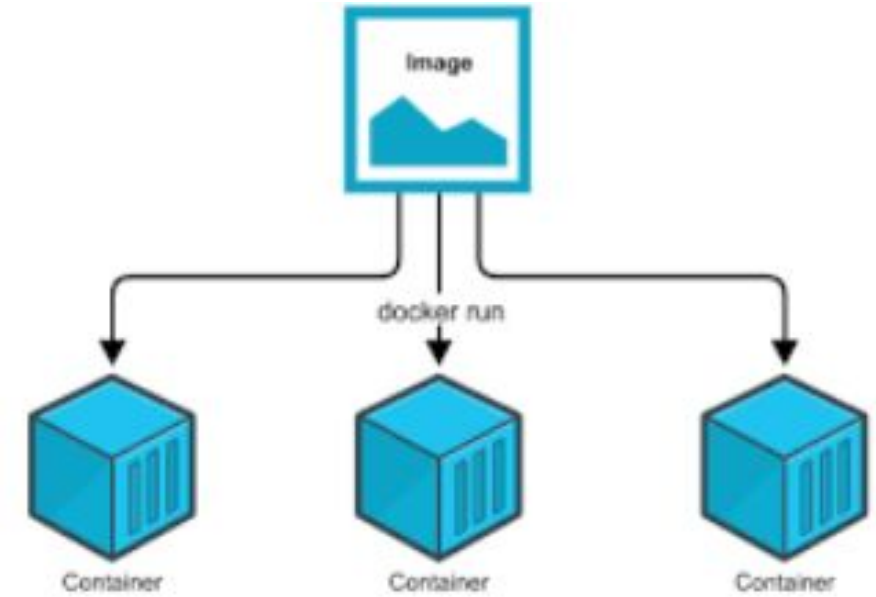
Containerization

HEPAPI
DIGITAL TRANSITION PARTNER

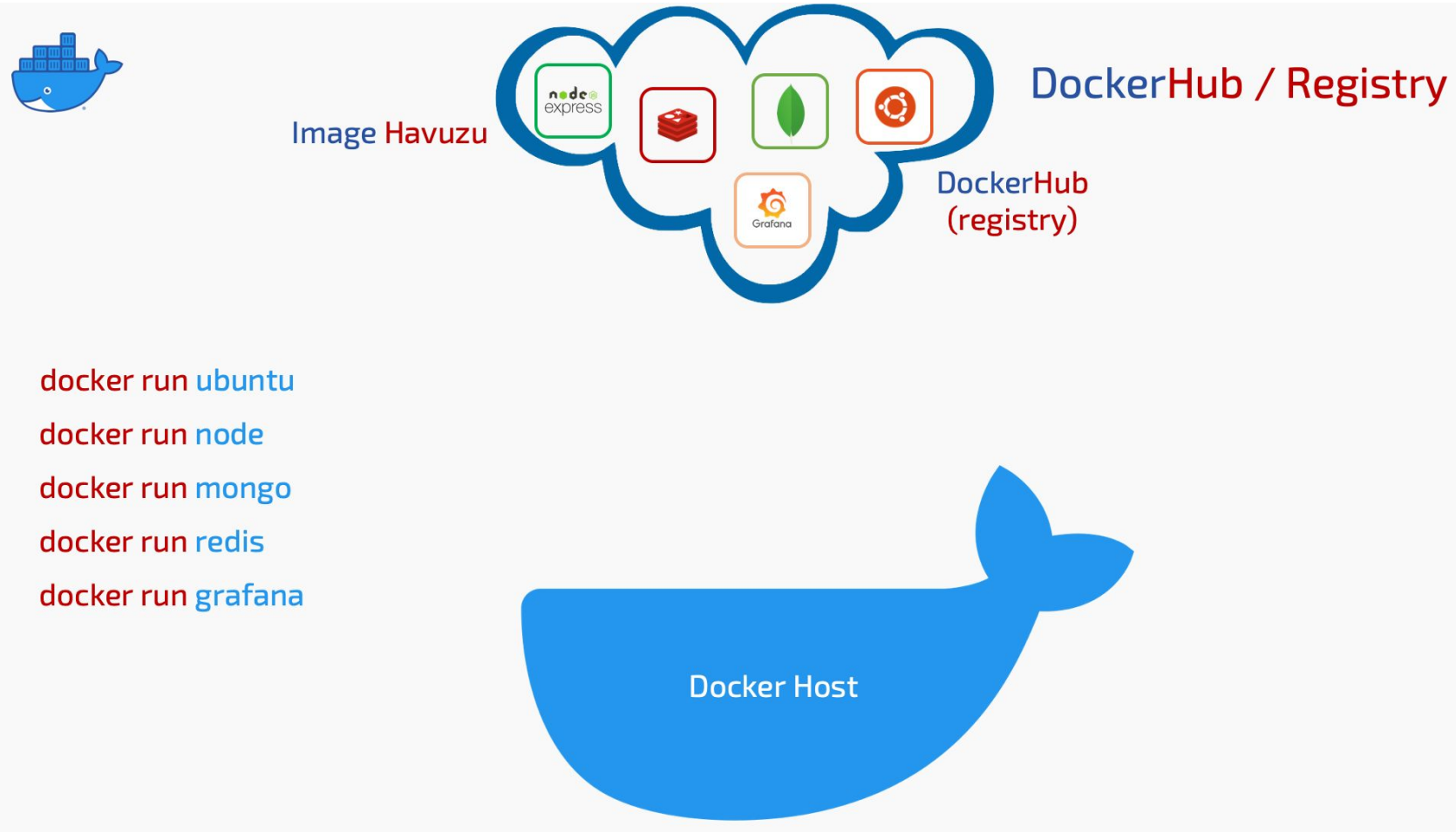
- Application Isolation
- Rapid Deployment and Scalability
- Portability
- Lightweight and Speed
- Continuous Integration and Delivery (CI/CD)
- Orchestration (Kubernetes)

Containerization





Containerization



Containerization

HEPAPI
DIGITAL TRANSITION PARTNER

HEPAPI
DIGITAL TRANSITION PARTNER

KUBERNETES

Kubernetes

HEPAPI
DIGITAL TRANSITION PARTNER

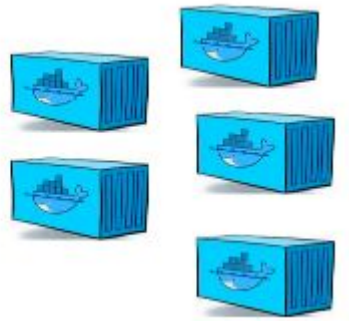
Orchestration

- 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**.

Kubernetes

HEPAPI
DIGITAL TRANSITION PARTNER

Orchestration



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

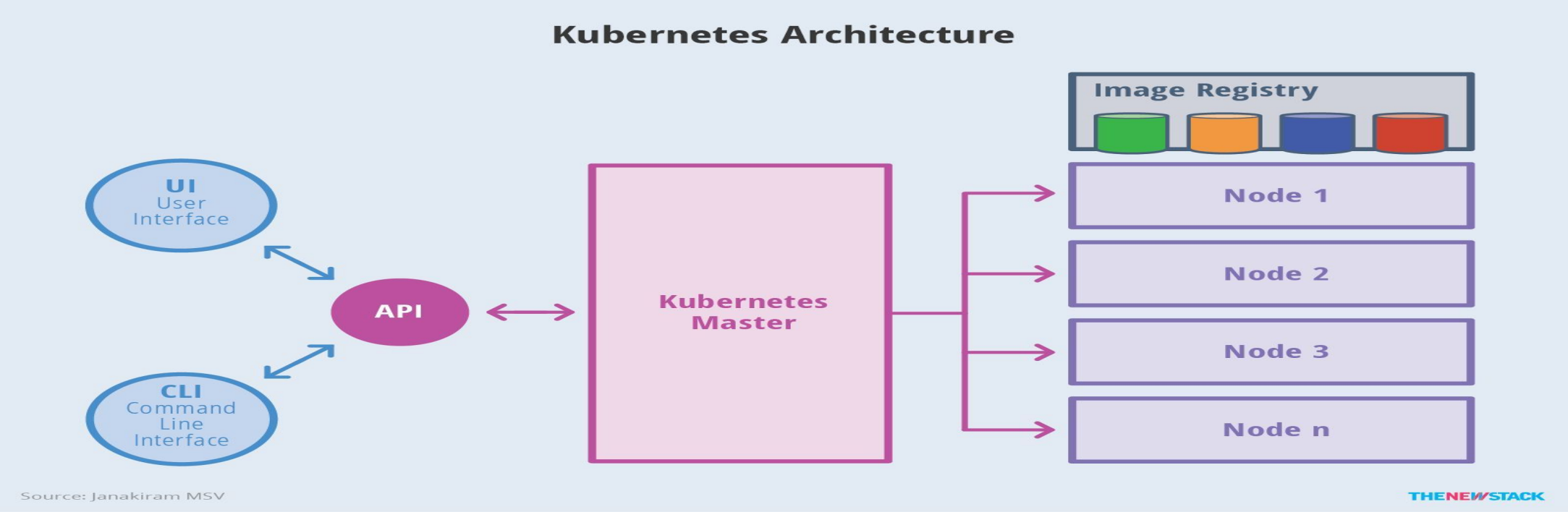
- Provisioning and deployments of containers
- Availability of containers
- Load balancing, traffic routing and service discovery of containers
- Health monitoring of containers
- Securing the interactions between containers.
- Configuring and scheduling of containers
- Allocation of resources between containers

Kubernetes

Kubernetes Components

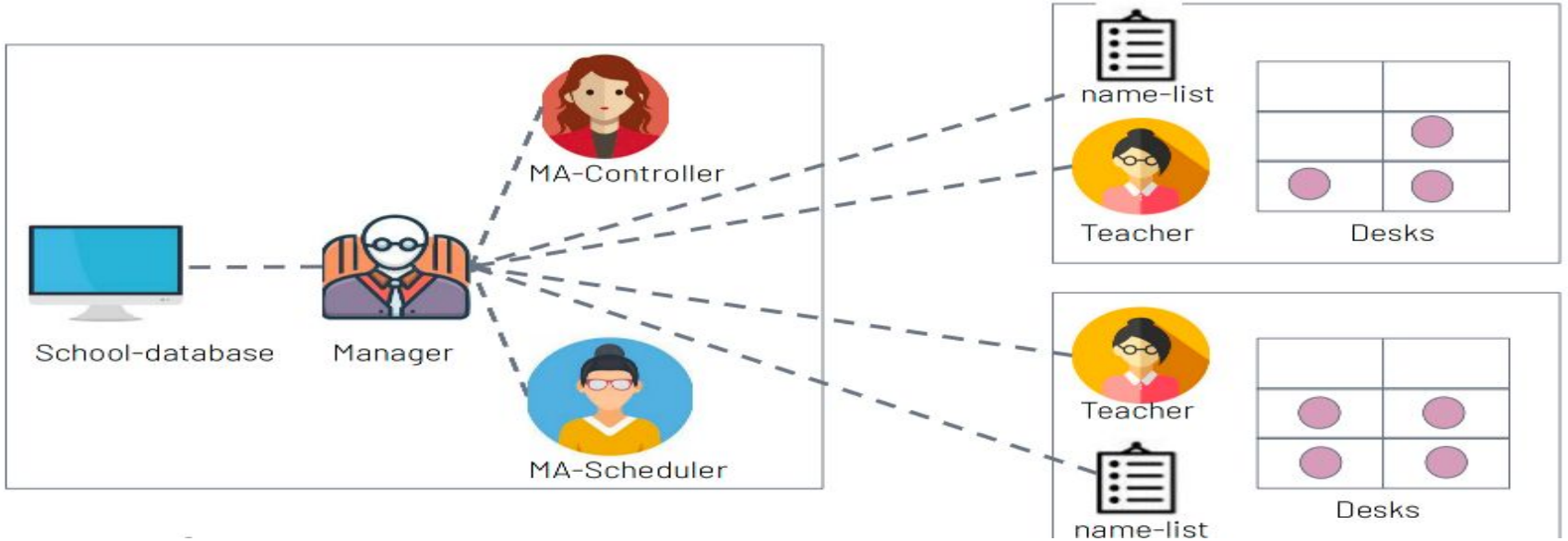
Kubernetes has the following main components:

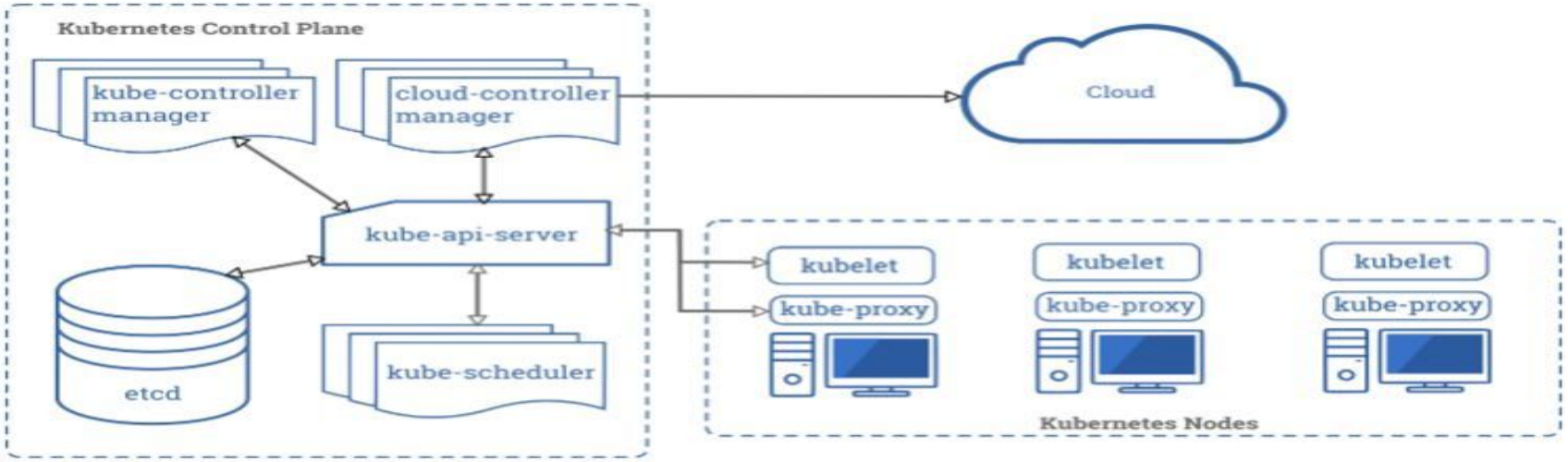
- One or more master nodes
- One or more worker nodes.



Kubernetes

Kubernetes Components





Kubernetes

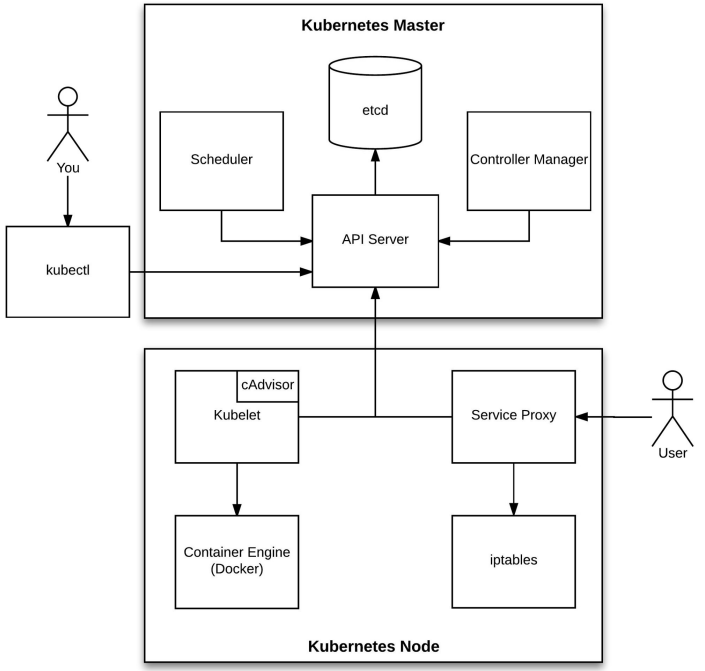
Kubernetes Components

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.

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.



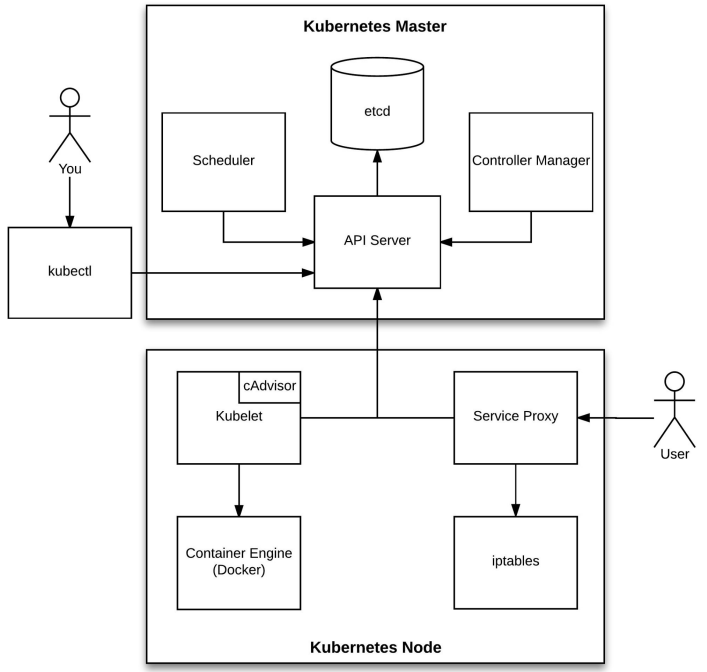
Kubernetes

Kubernetes 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



kube-scheduler:

- Verbose policy-rich engine that evaluates workload requirements and attempts to place it on a matching resource.

- Workload Requirements can include: general hardware requirements, affinity/anti-affinity, labels, and other various custom resource requirements.

Kubernetes

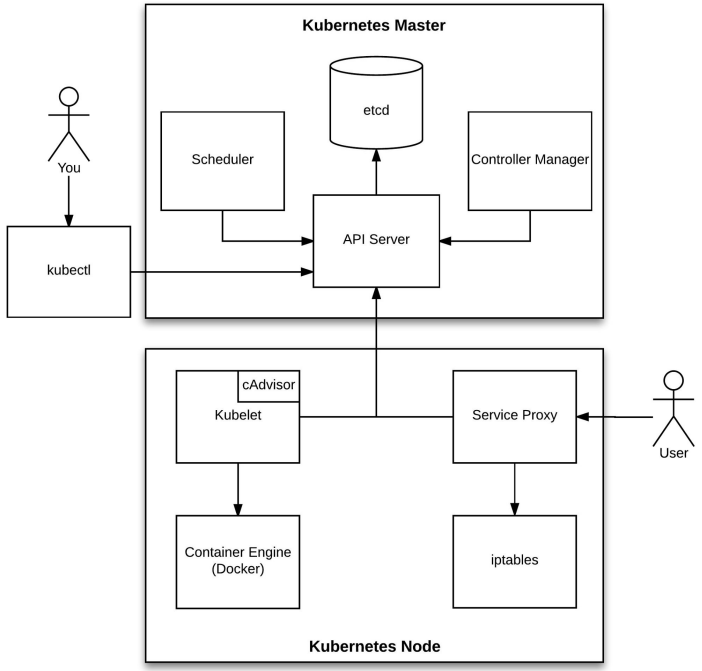
Kubernetes Components

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.

kube-proxy:

- Manages the network rules on each node.
- Performs connection forwarding or load balancing for Kubernetes cluster services.



HEPAPI
DIGITAL TRANSITION PARTNER

KUBERNETES Objects

- PODs
- ReplicaSets
- Deployment
- Namespaces
- Object Model

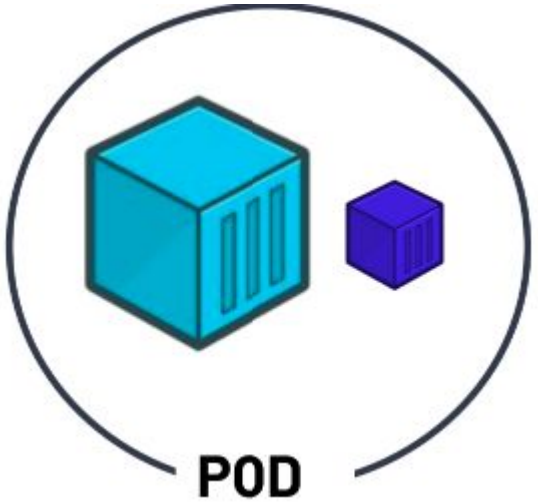
Kubernetes Objects

HEPAPI
DIGITAL TRANSITION PARTNER

Pods

- Kubernetes doesn't deal with containers directly.
- PODs are Kubernetes objects that encapsulate the containers.
- Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

- A POD can have multiple containers.
- Sometimes an application need a helper container, such as logging, monitoring, etc.
- These helper containers should coexist with your application container.



Kubernetes Objects

HEPAPI
DIGITAL TRANSITION PARTNER

Pods

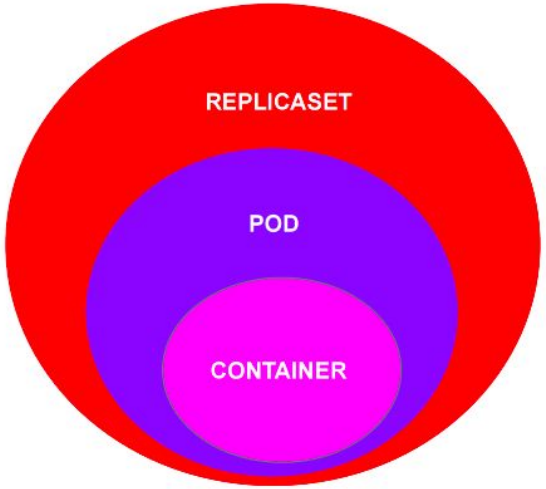
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: init-demo
spec:
  containers:
  - name: main-app
    image: nginx
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', 'echo Initializing Service; sleep 30']
```

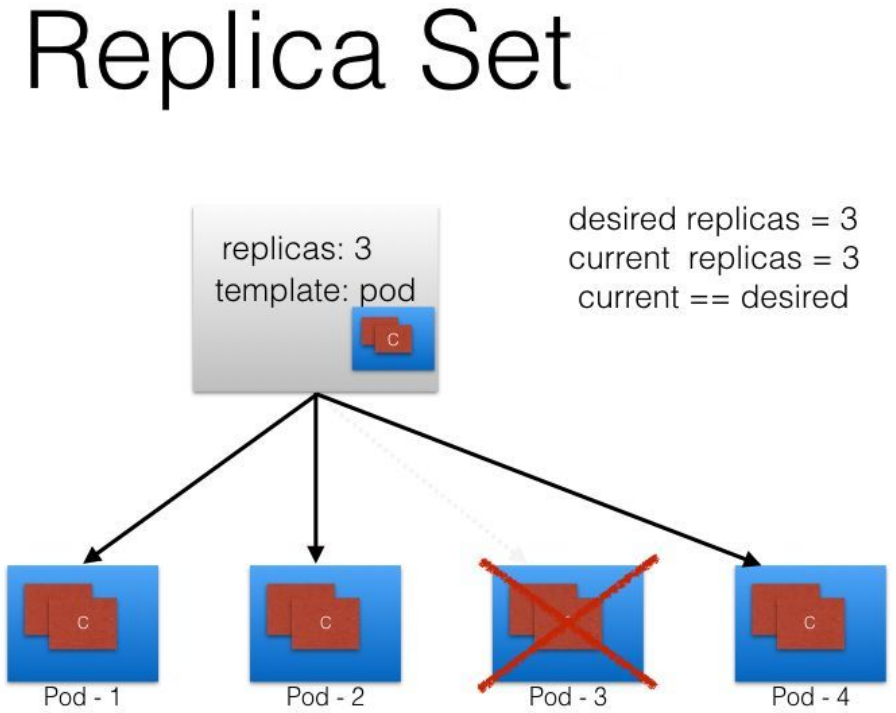
```yaml
apiVersion: v1
kind: Pod
metadata:
  name: sidecar-demo
spec:
  containers:
  - name: main-app
    image: nginx
  - name: logger-sidecar
    image: fluentd
    volumeMounts:
    - name: log-volume
      mountPath: /var/log/nginx
  volumes:
  - name: log-volume
    emptyDir: {}
```

Kubernetes Objects

ReplicaSets

- A **ReplicaSet's** purpose is to maintain a stable set of replica Pods running at any given time.
- Even if you have a single POD, the ReplicaSet will bring up a new POD when the existing one fails.





Kubernetes Objects

ReplicaSets

Pod to ReplicaSet

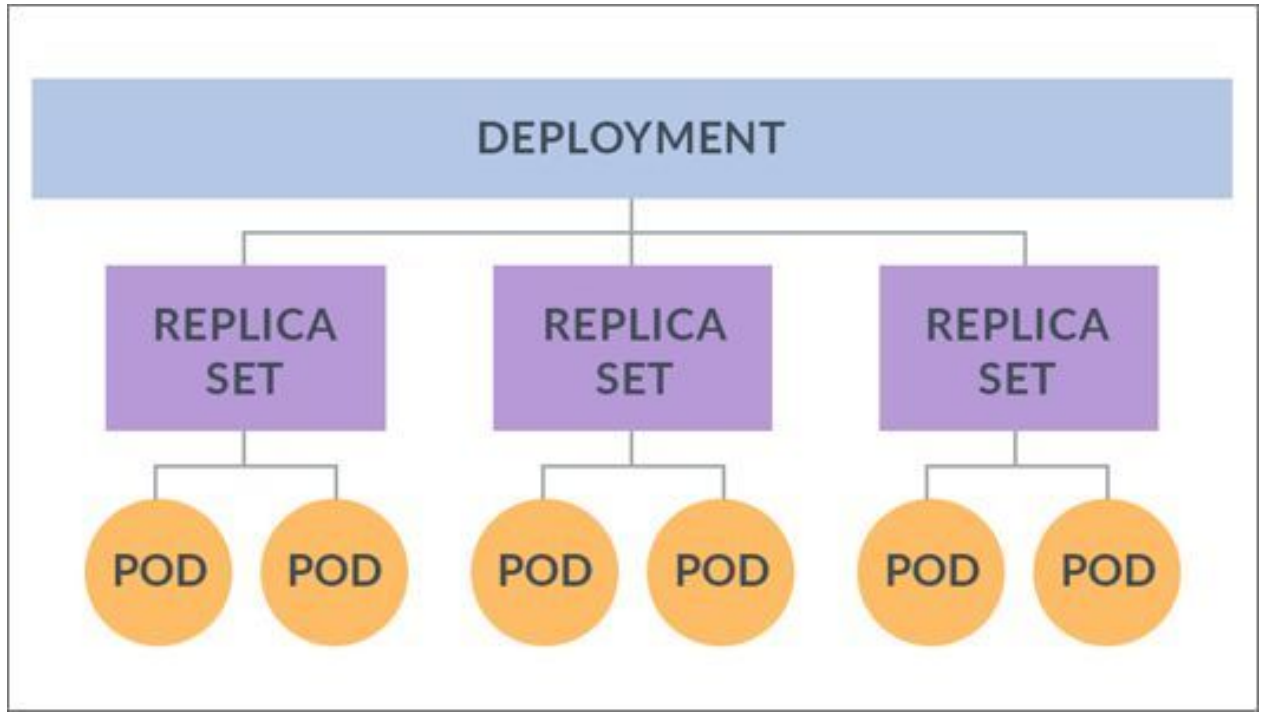
```yaml
apiVersion: v1
kind: Pod
metadata:
 name: nginx-pod
 labels:
   app: nginx
spec:
 containers:
 - name: mynginx
   image: nginx:1.19
   ports:
   - containerPort: 80
```

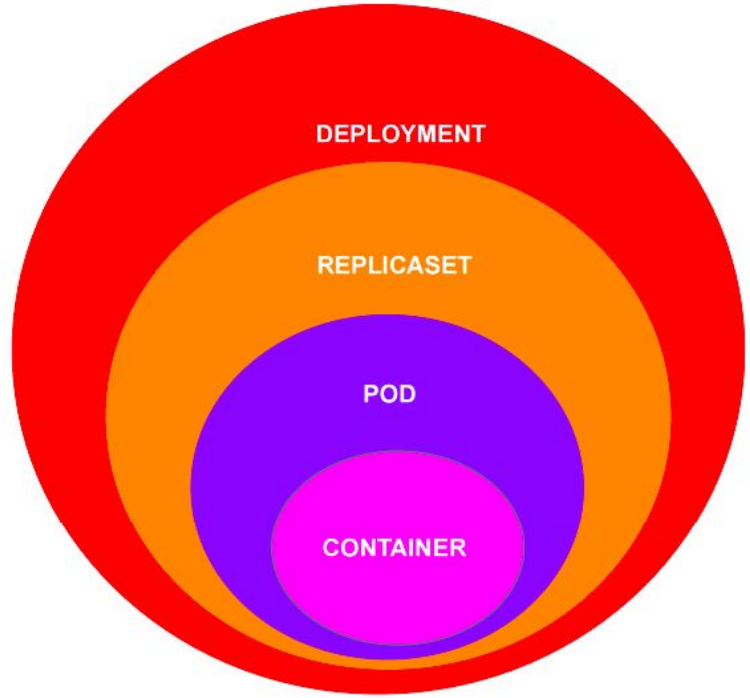
```yaml
apiVersion: apps/v1
kind: ReplicaSet
metadata:
 name: nginx-rs
 labels:
   environment: dev
spec:
 replicas: 3
 selector:
   matchLabels:
     app: nginx
 template:
   metadata:
     labels:
       app: nginx
   spec:
     containers:
     - name: mynginx
       image: nginx:1.19
       ports:
       - containerPort: 80
```

Kubernetes Objects

Deployment

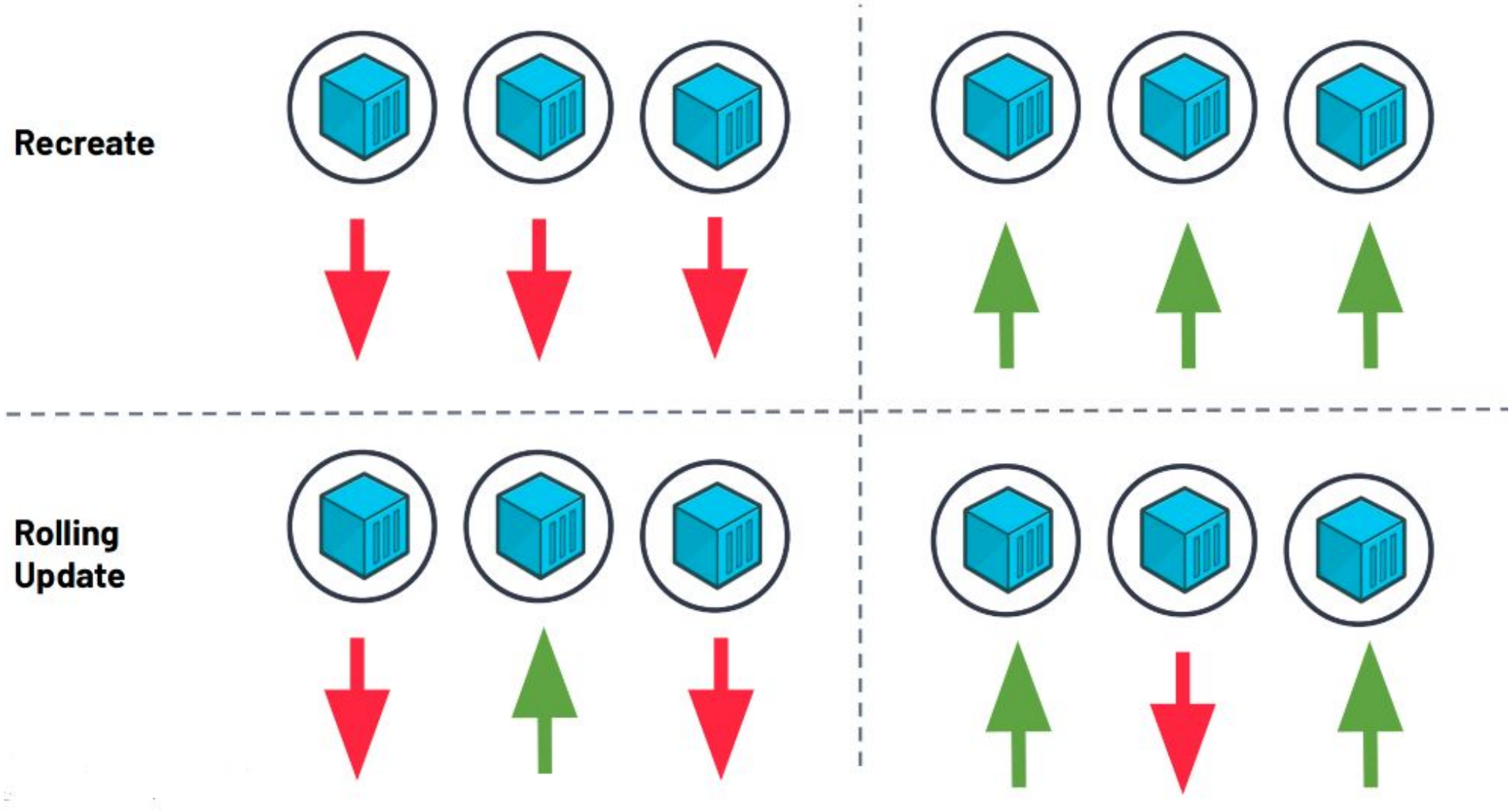
- One step higher in the hierarchy, deployments provides declarative updates for Pods and ReplicaSets.





Kubernetes Objects

Deployment Strategy

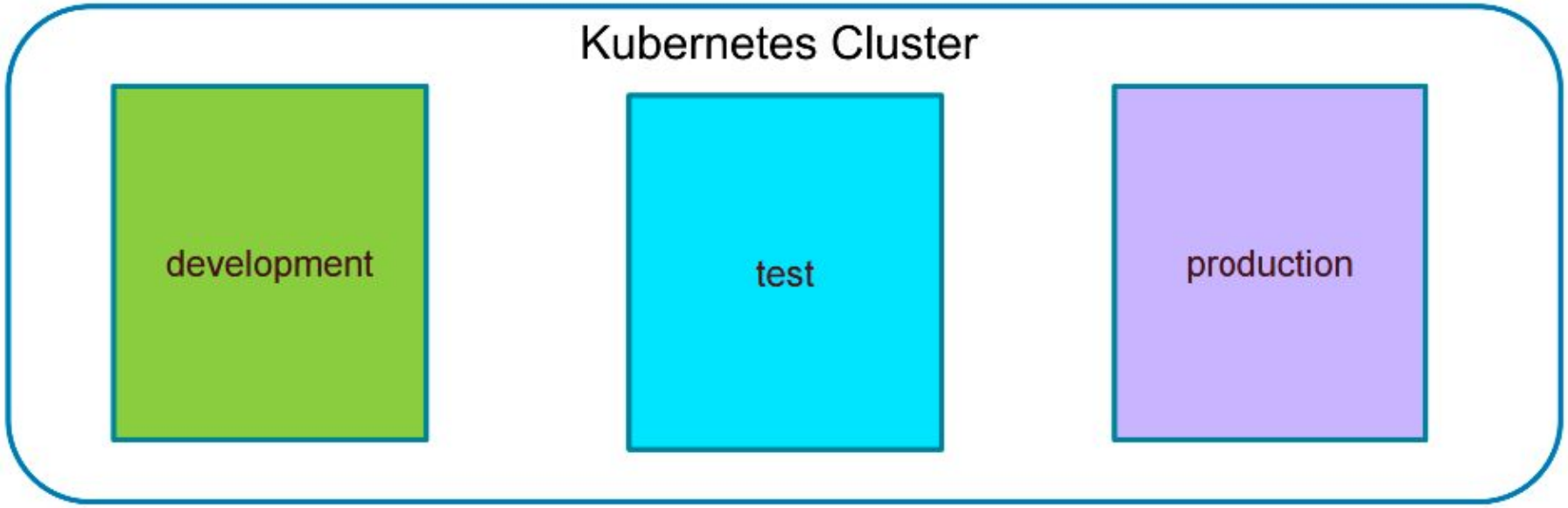


Kubernetes Objects

HEPAPI
DIGITAL TRANSITION PARTNER

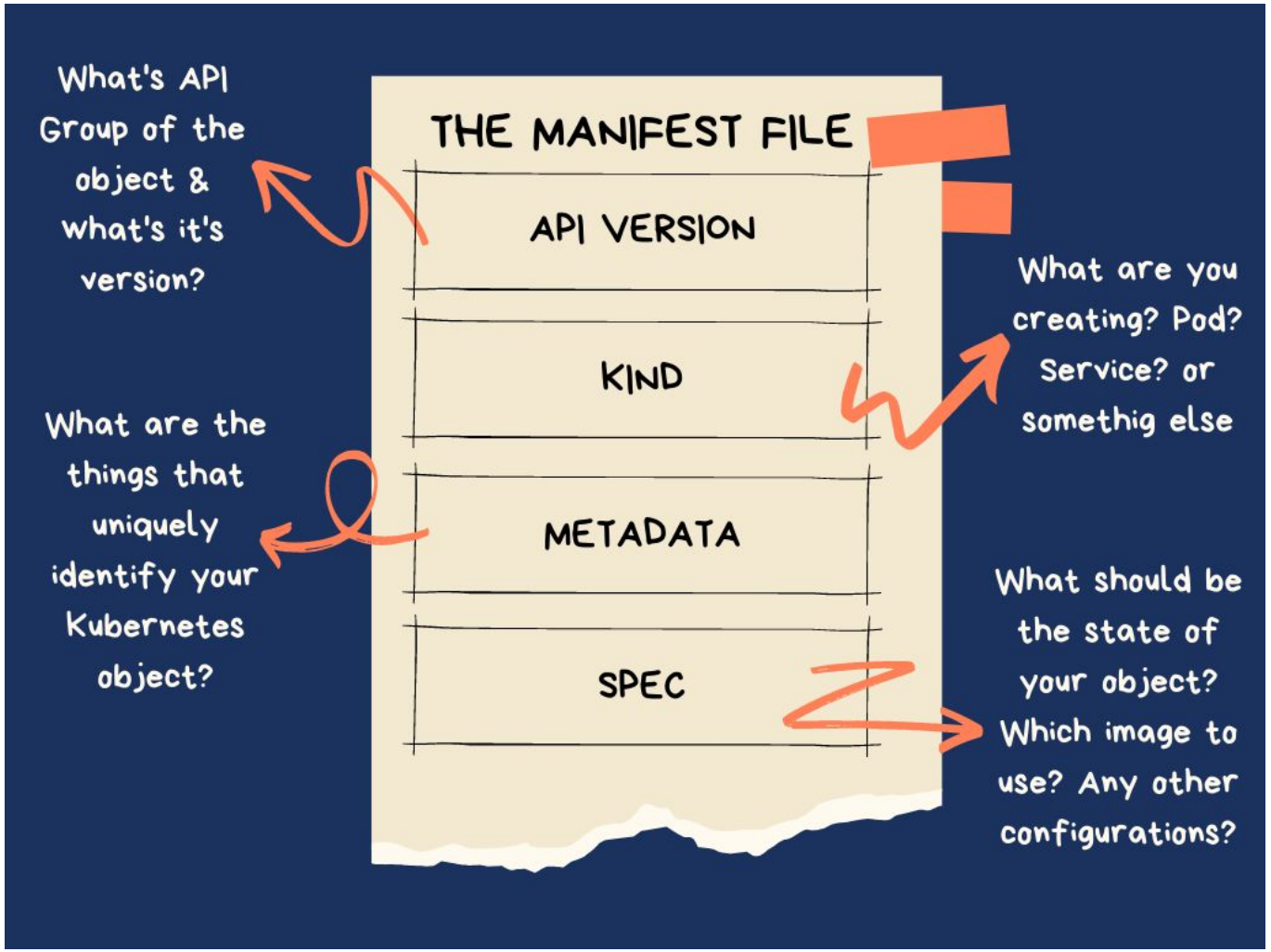
Namespaces

- Kubernetes supports multiple virtual clusters backed by the same physical cluster. These virtual clusters are called **namespaces**.
- Namespaces are intended for use in environments with many users spread across multiple teams, or projects.



Kubernetes Objects

Object Model



Kubernetes Objects

HEPAPI
DIGITAL TRANSITION PARTNER

Object Model

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  selector:
    matchLabels:
      app: nginx
  replicas: 2
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2
        ports:
        - containerPort: 80
```

- All objects must have **apiVersion, kind, metadata** and **spec** fields.
- **apiVersion:** Which version of the Kubernetes API you're using to create this object
- **kind:** What kind of object you want to create
- **metadata:** Data that helps uniquely identify the object, including a **name** string, **labels**, and optional **namespace**
- **spec:** What state you desire for the object

HEPAPI
DIGITAL TRANSITION PARTNER

KUBERNETES Networking

- Cluster Networking
- Services
- Service Types
- Ingress

Kubernetes Networking

HEPAPI
DIGITAL TRANSITION PARTNER

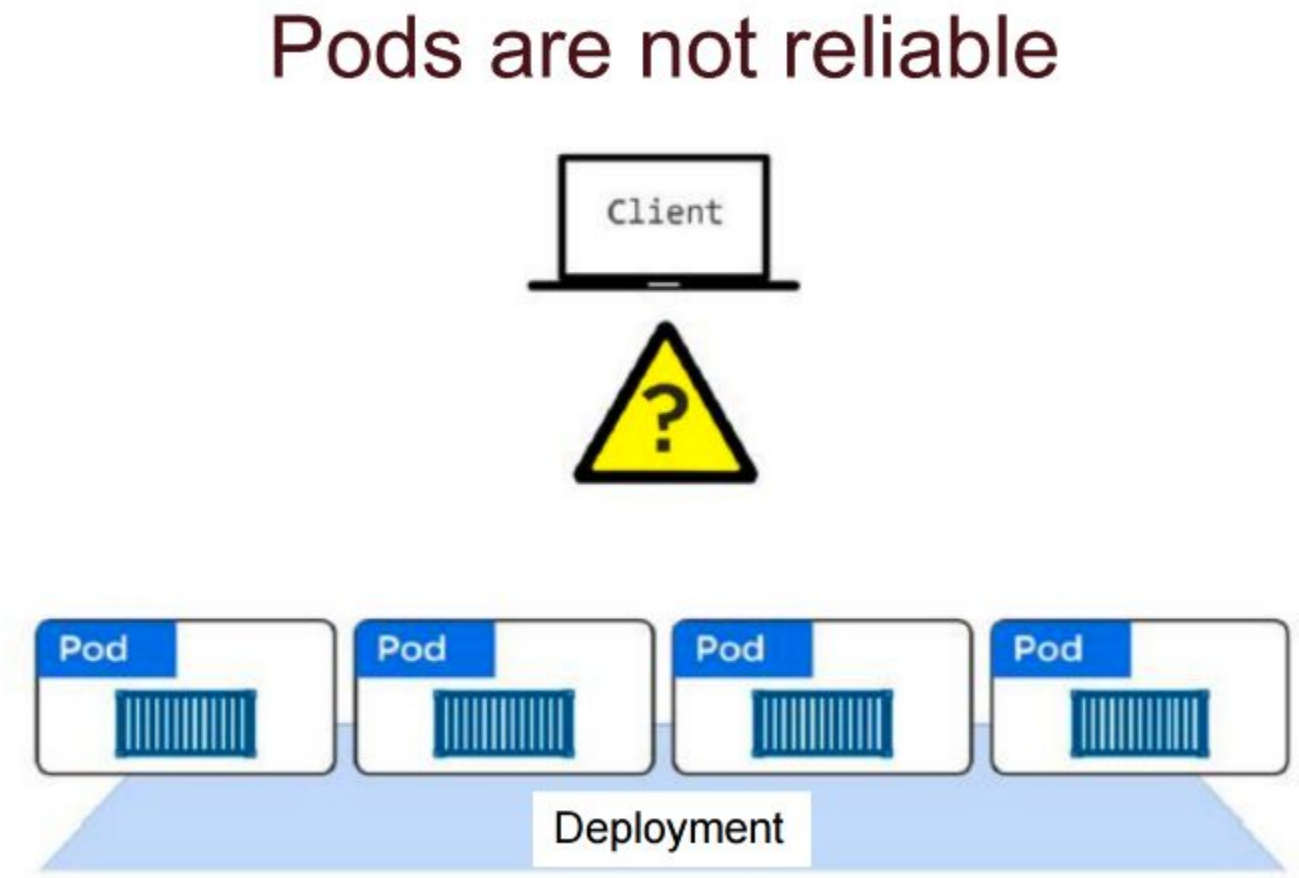
Cluster Networking

There are 4 distinct networking problems to address:

1. container-to-container communications:
2. Pod-to-Pod communications:
3. Pod-to-Service communications: this is covered by services.
4. External-to-Service communications: this is covered by services.

Kubernetes Networking

Services

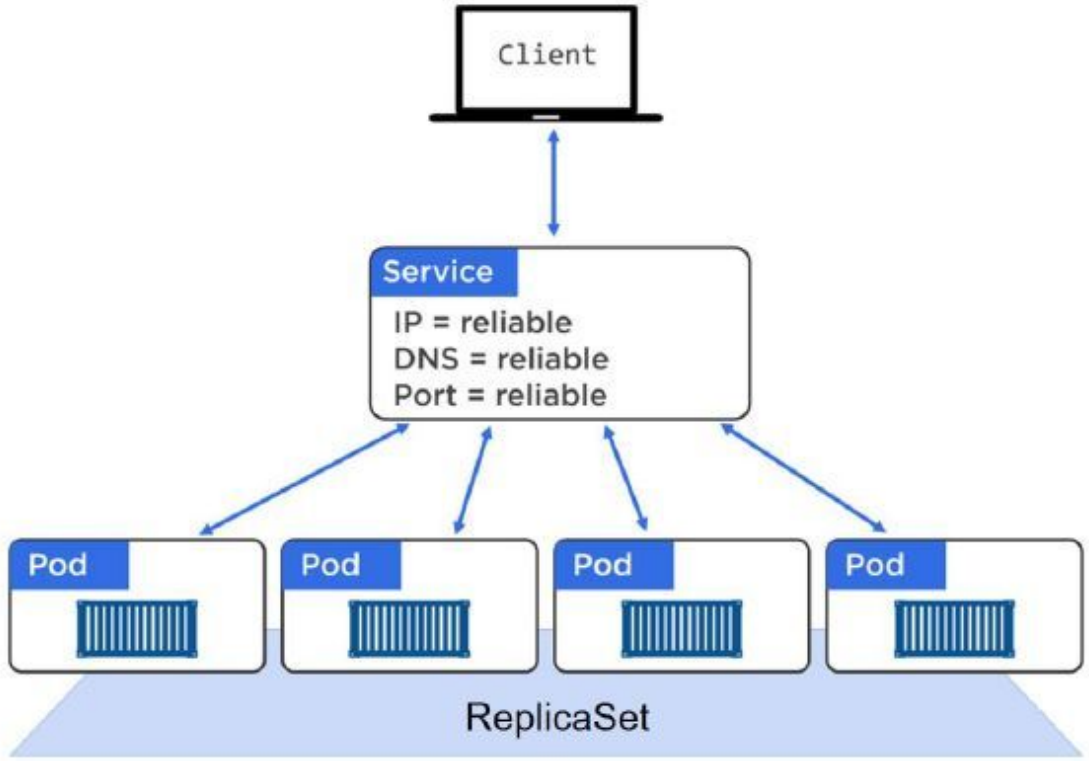


Kubernetes Networking

Services

A **Service** offers a single **DNS entry** for a containerized application managed by the Kubernetes cluster

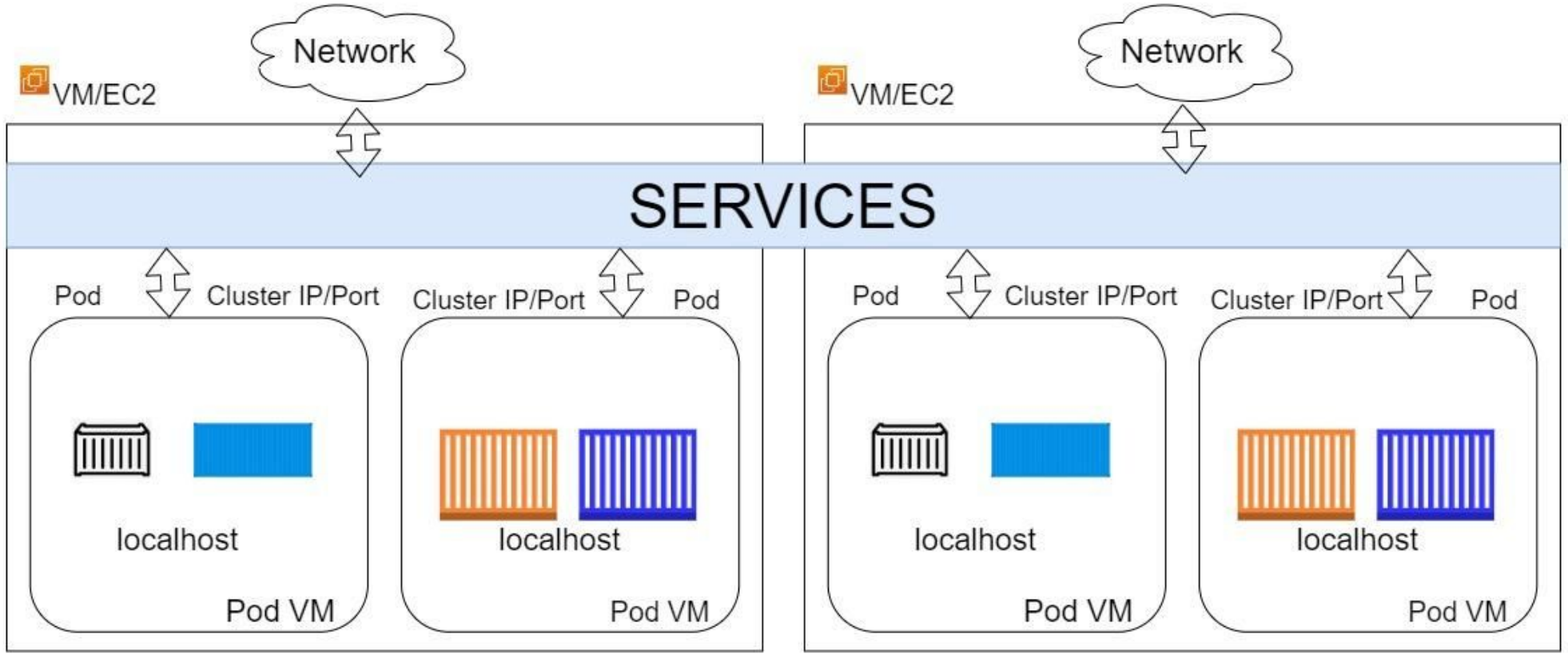
The Service is associated with the Pods, and provides them with a stable IP, DNS and port. It also loadbalances requests across the Pods.



Kubernetes Networking

HEPAPI
DIGITAL TRANSITION PARTNER

Services

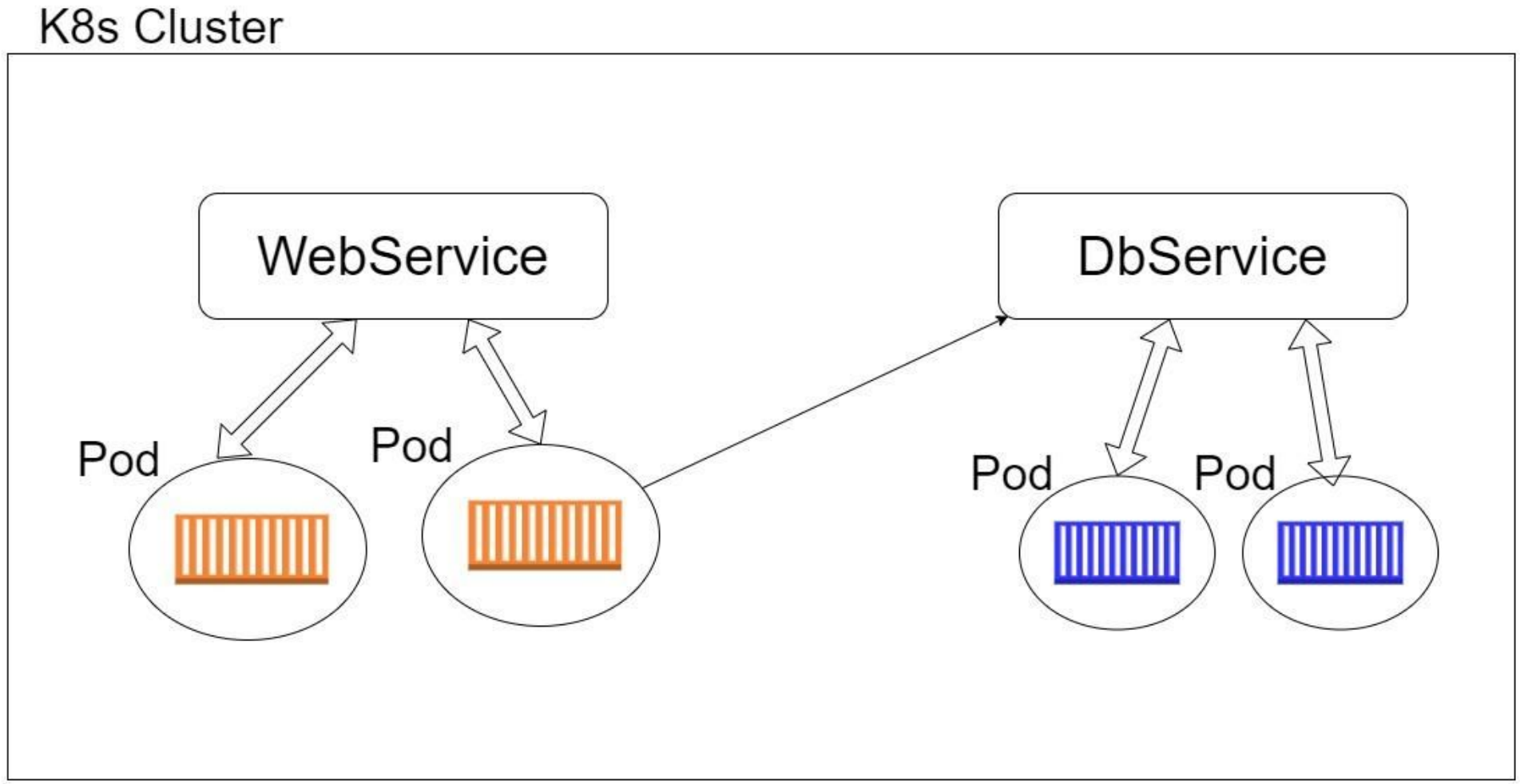


K8s Network

Kubernetes Networking

HEPAPI
DIGITAL TRANSITION PARTNER

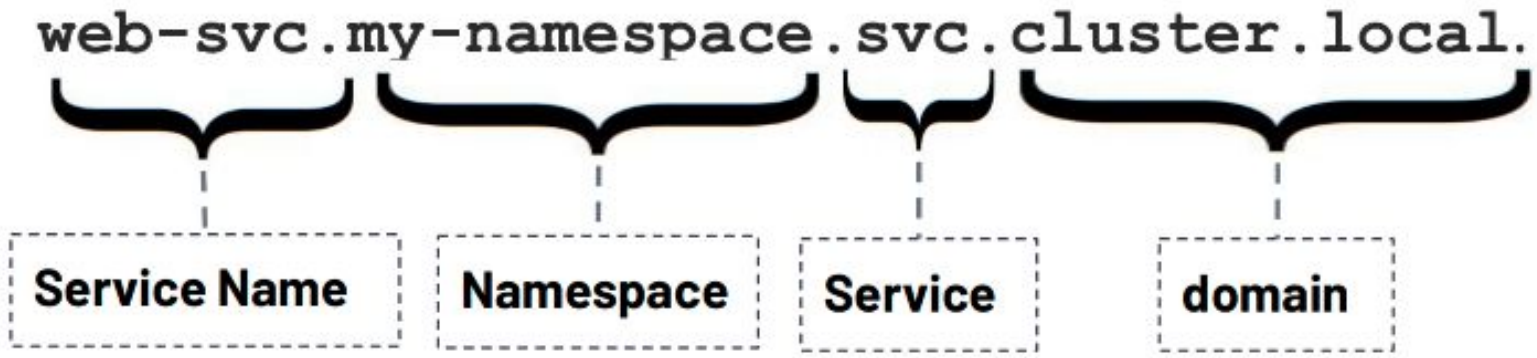
Services



Kubernetes Networking

Services Discovery

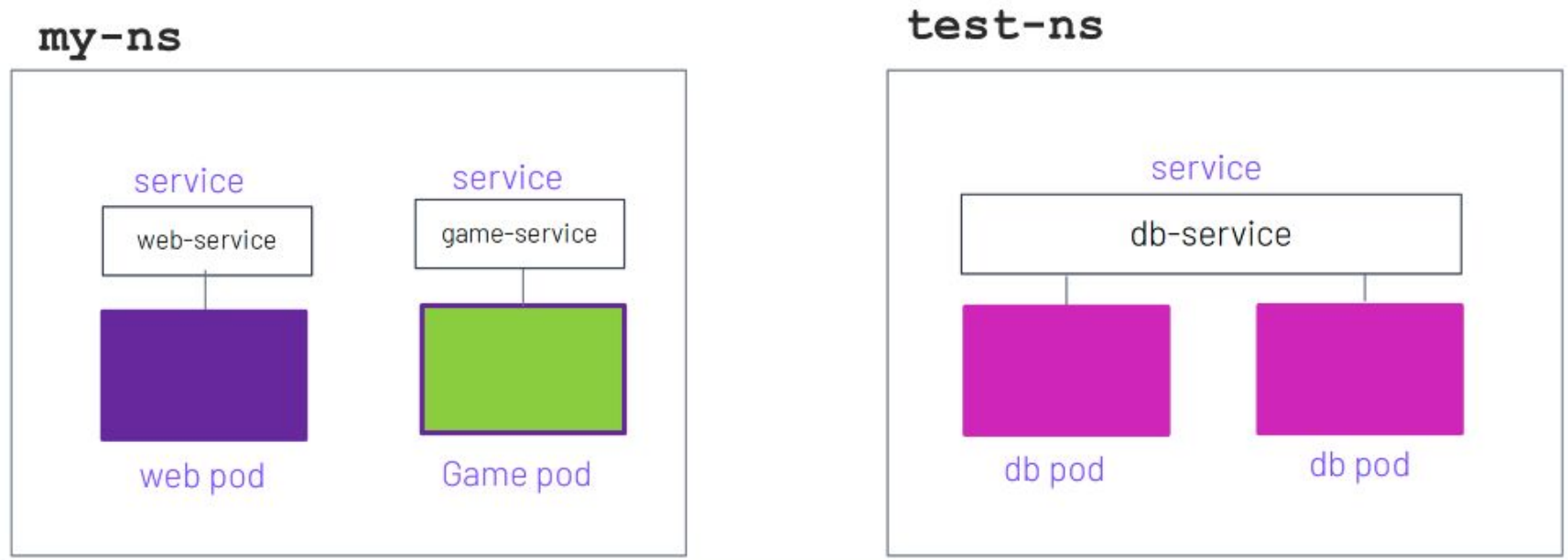
Kubernetes has an add-on for **DNS**, which creates a DNS record for each Service and its format is



Kubernetes Networking

HEPAPI
DIGITAL TRANSITION PARTNER

Services Discovery



To connect to the "Game pod" and "db pod":

From " web pod" -> " Game pod" —> hostname: game-service.my-ns:port
game-service:port

From " web pod" -> "db pod —> hostname: db-service.test-ns.svc.cluster.local:port

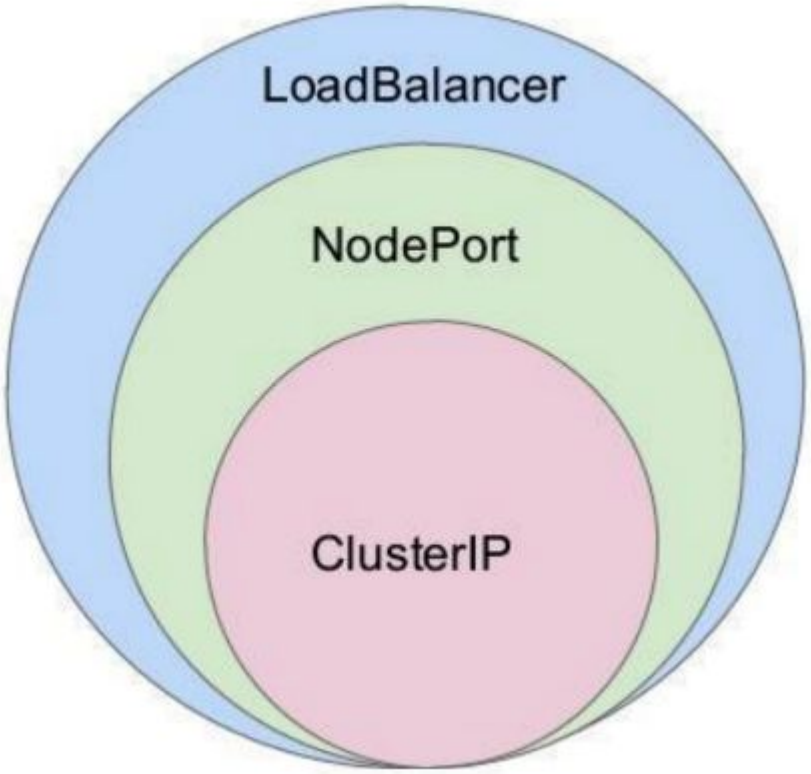
Kubernetes Networking

HEPAPI
DIGITAL TRANSITION PARTNER

Services Types

There are 3 major service types:

- ClusterIP (default)
- NodePort
- LoadBalancer



Kubernetes Networking

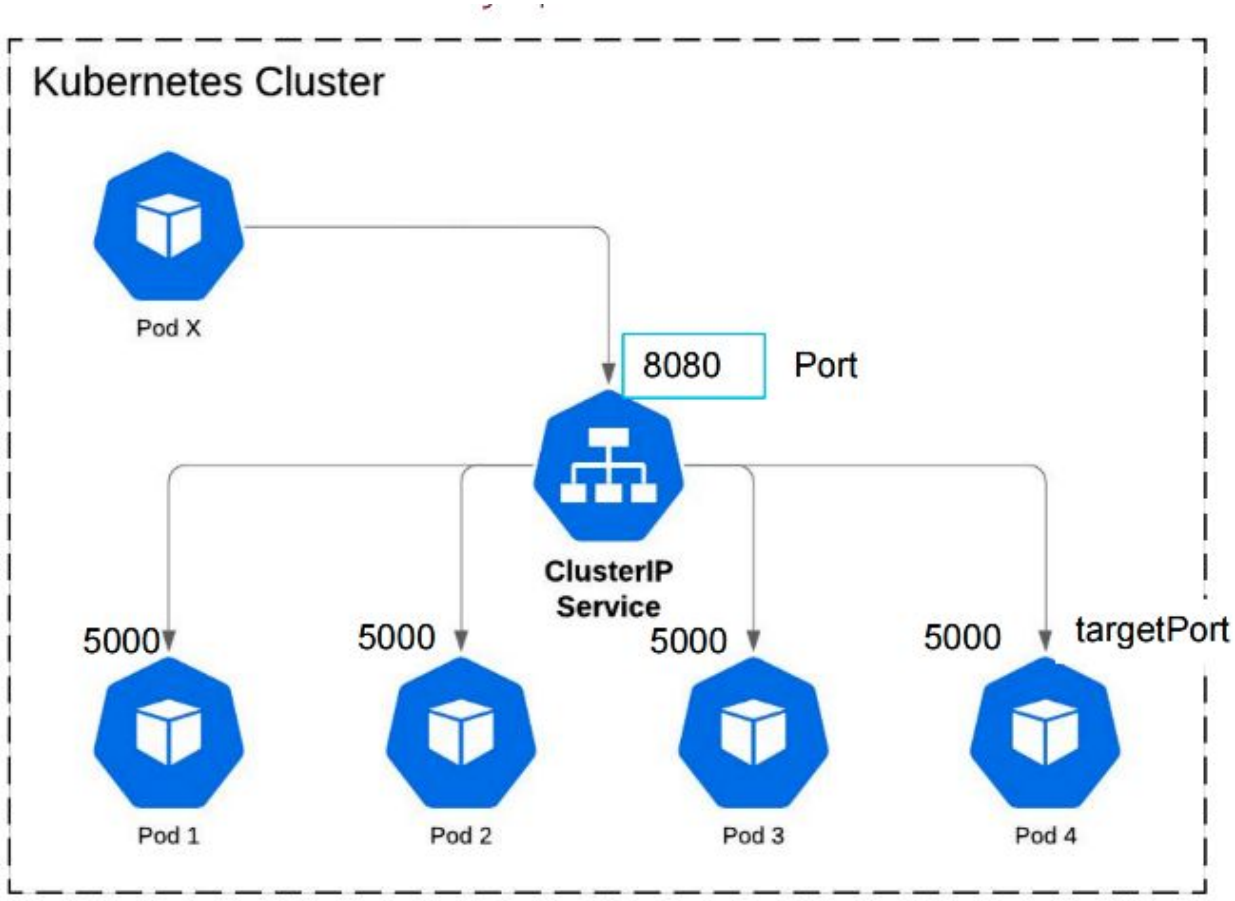
HEPAPI
DIGITAL TRANSITION PARTNER

ClusterIP

Expose traffic internally

Example Usecase:

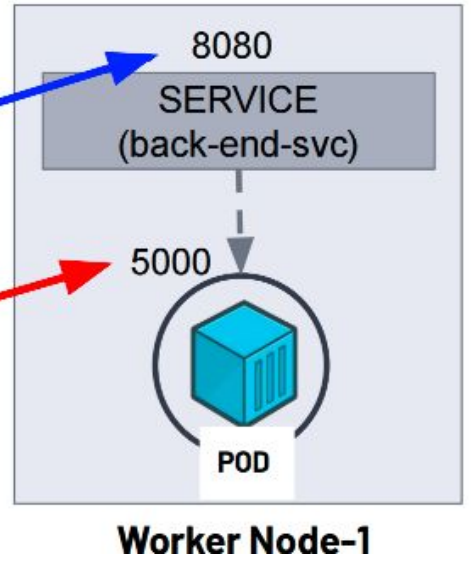
Good for service of frontend & back-end apps.



Kubernetes Networking

ClusterIP

```
apiVersion: v1
kind: Service
metadata:
 name: back-end-svc
 labels:
   app: back-end
spec:
 type: ClusterIP (default)
 selector:
   app: back-end
 ports:
 - port: 8080
   protocol: TCP
   targetPort: 5000
```



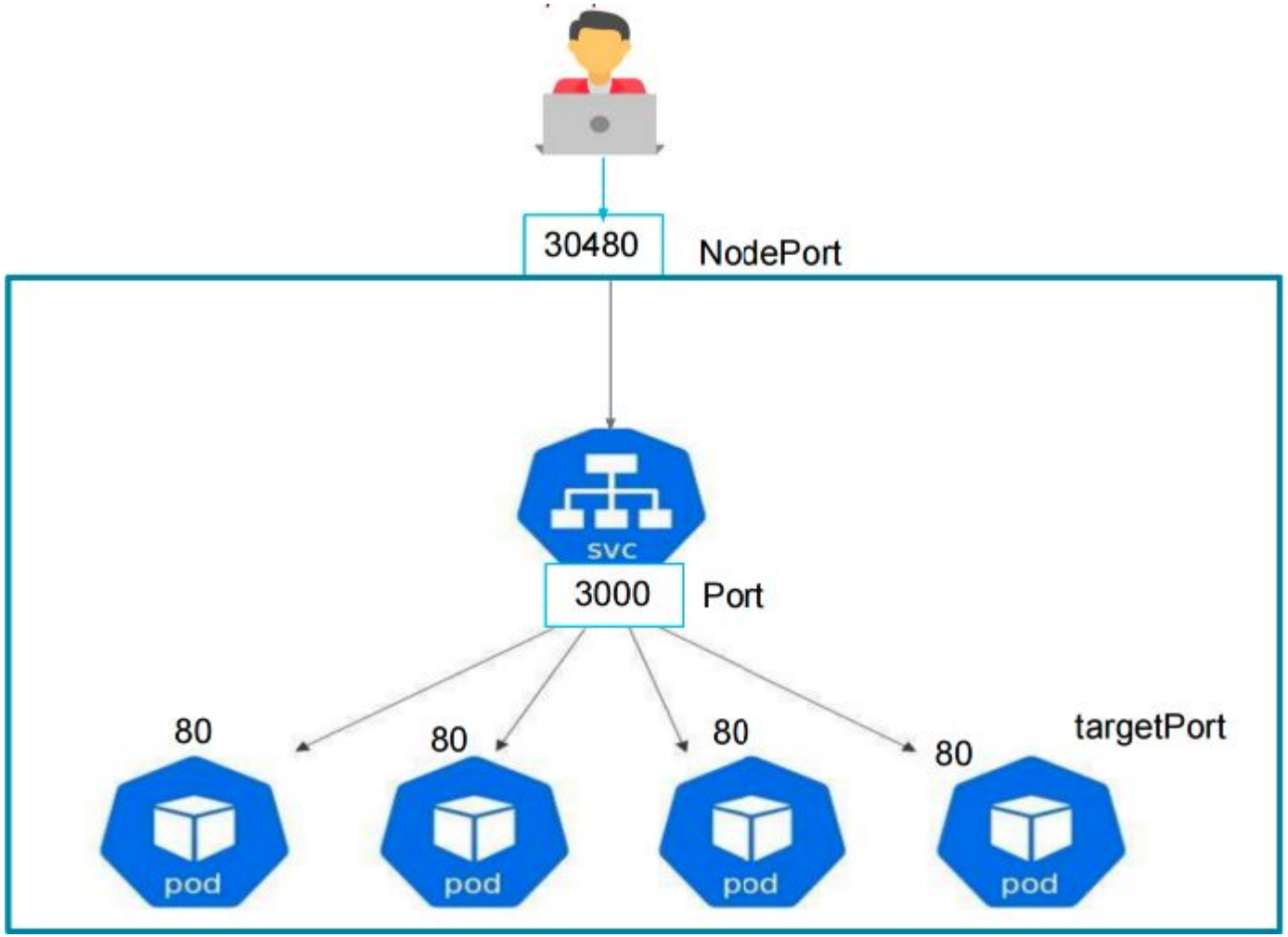
Kubernetes Networking

HEPAPI
DIGITAL TRANSITION PARTNER

NodePort

Exposes traffic to the outside.

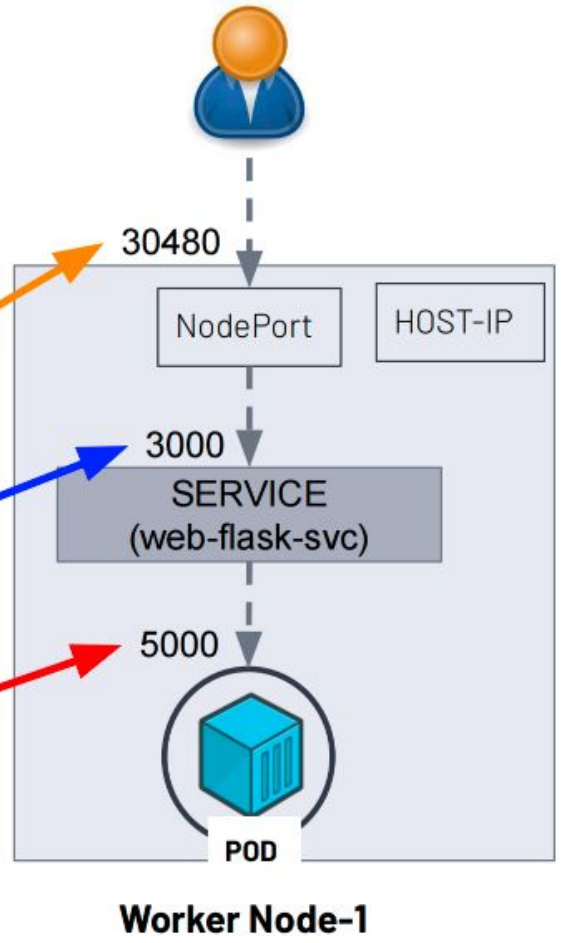
Example Use Case:
when we want to make our Services which has our app or website accessible from the external world.



Kubernetes Networking

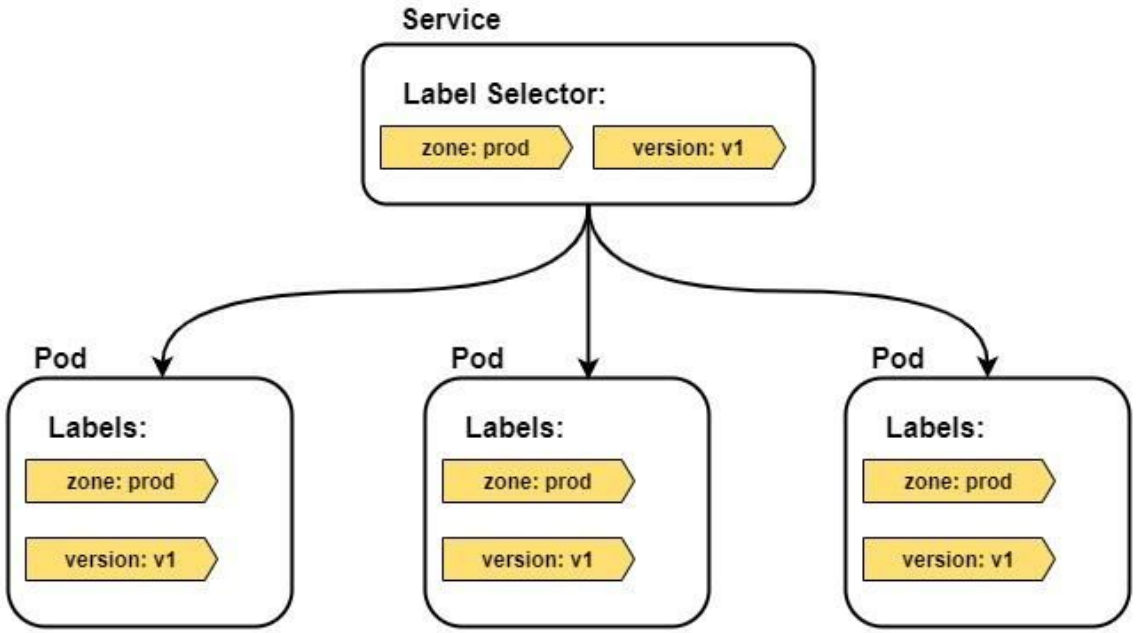
NodePort

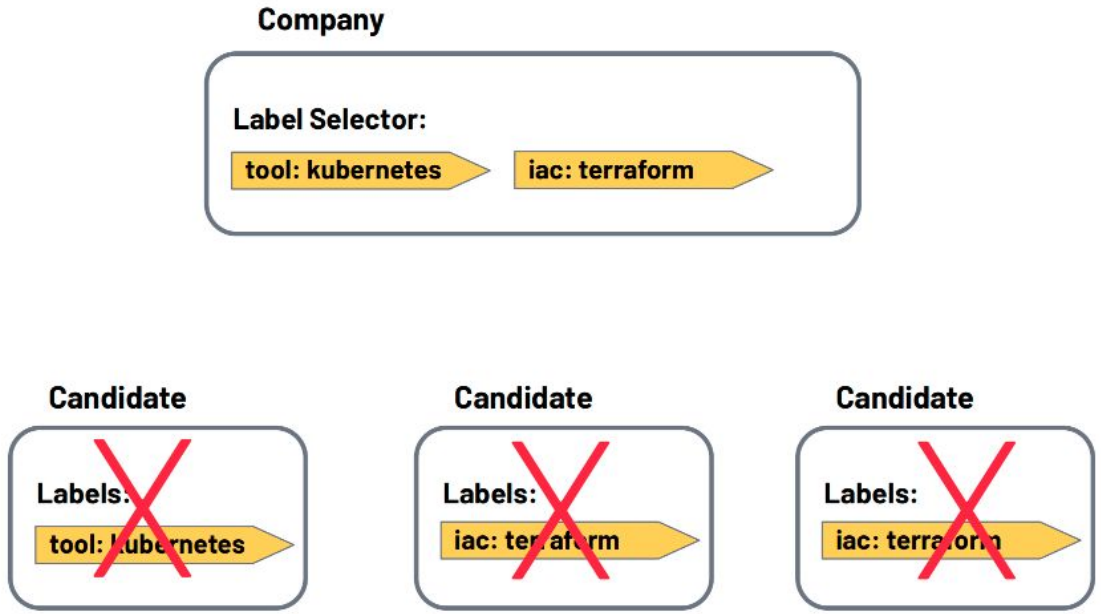
```
apiVersion: v1
kind: Service
metadata:
 name: web-flask-svc
 labels:
   app: web-flask
spec:
 type: NodePort
 selector:
   app: web-flask
 ports:
 - nodePort: 30480
   port: 3000
   protocol: TCP
   targetPort: 5000
```



Kubernetes Networking

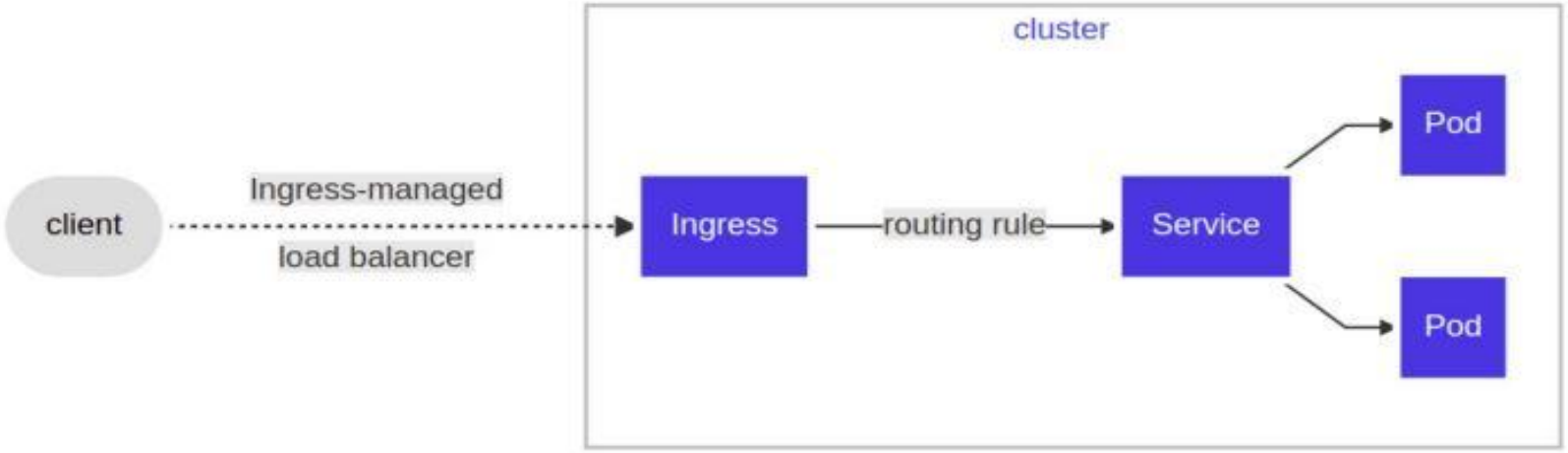
Labels and loose coupling





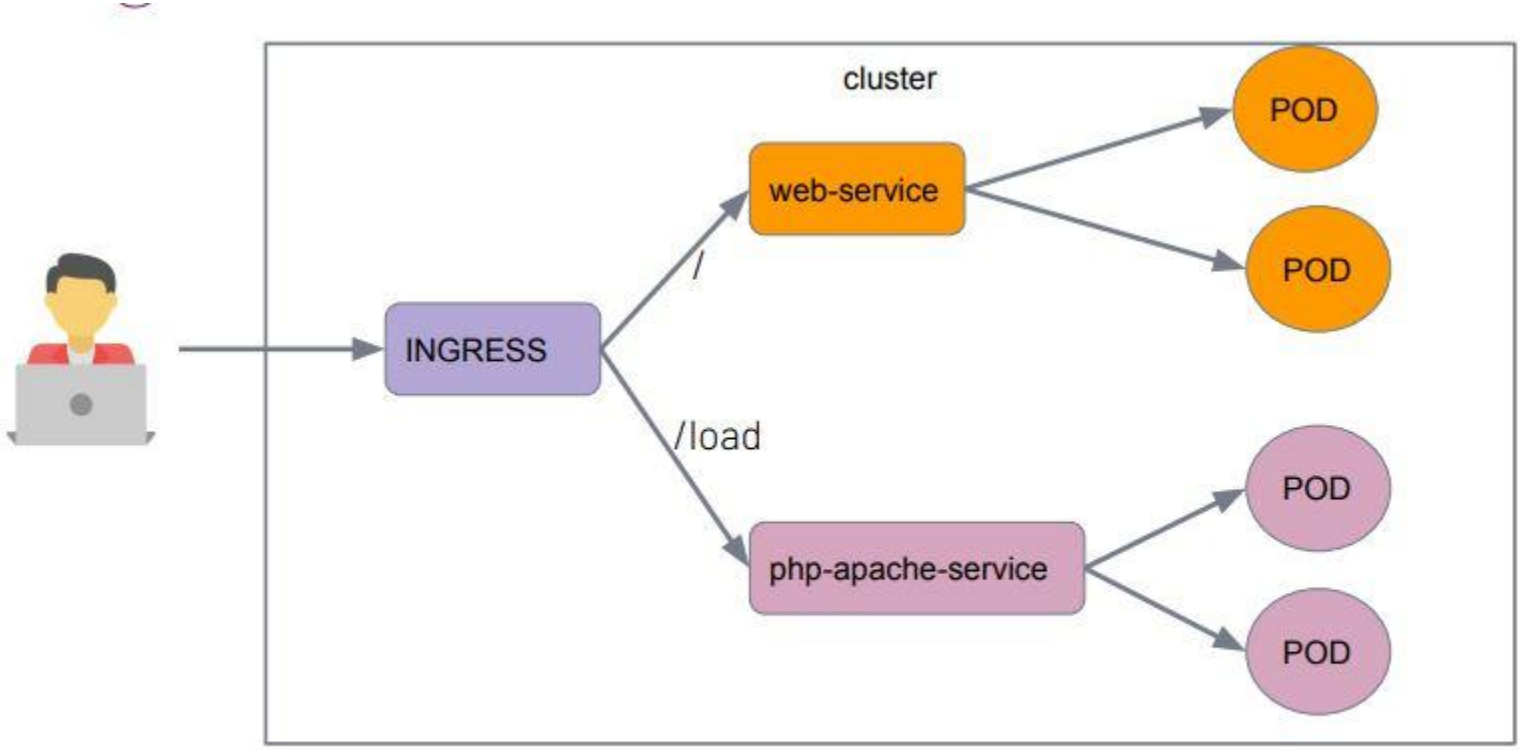
Kubernetes Networking

Ingress: With Ingress, users do not connect directly to a Service. Users reach the Ingress endpoint, and, from there, the request is forwarded to the desired Service.



Kubernetes Networking

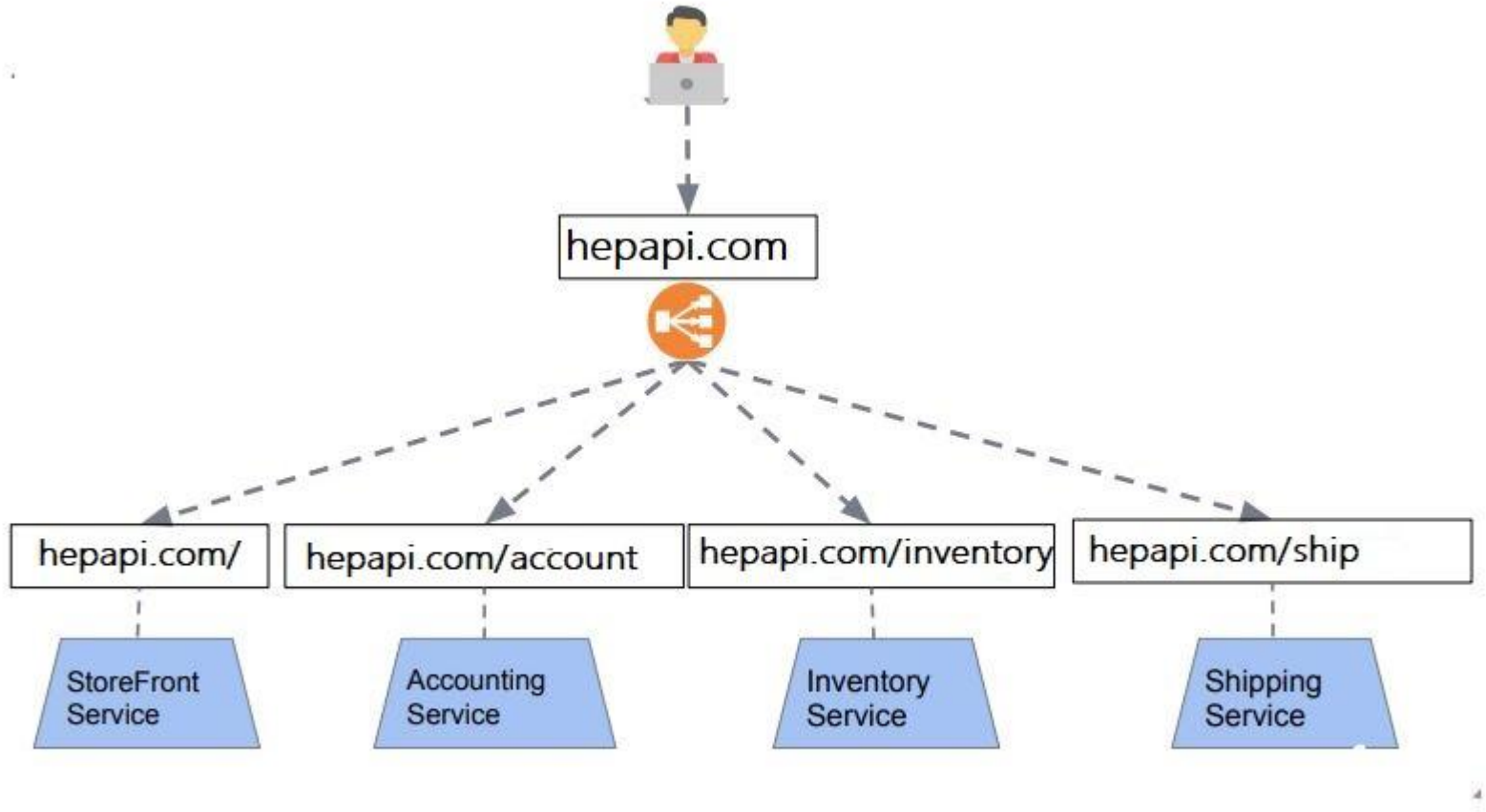
Ingress



Kubernetes Networking

HEPAPI
DIGITAL TRANSITION PARTNER

Ingress: exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.



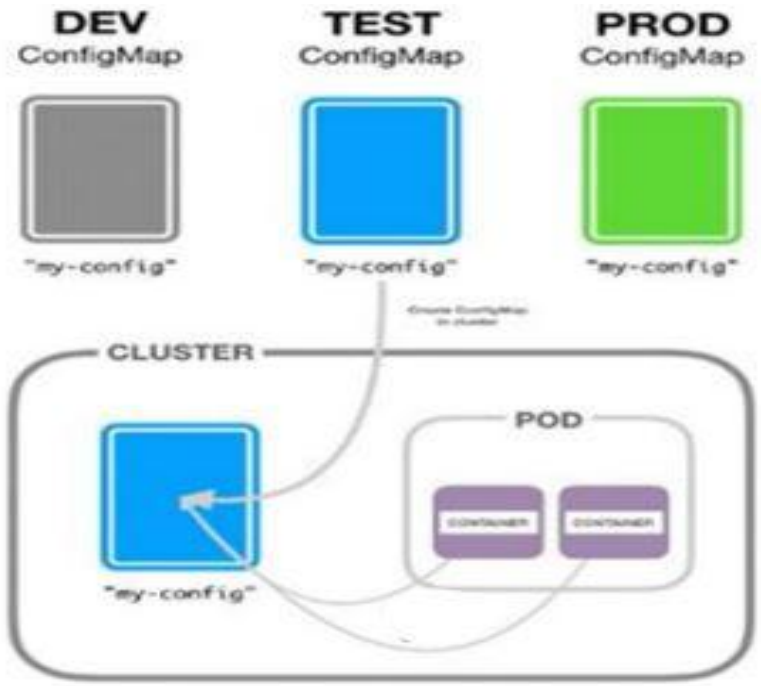
HEPAPI
DIGITAL TRANSITION PARTNER

Secrets and Configmaps

ConfigMap

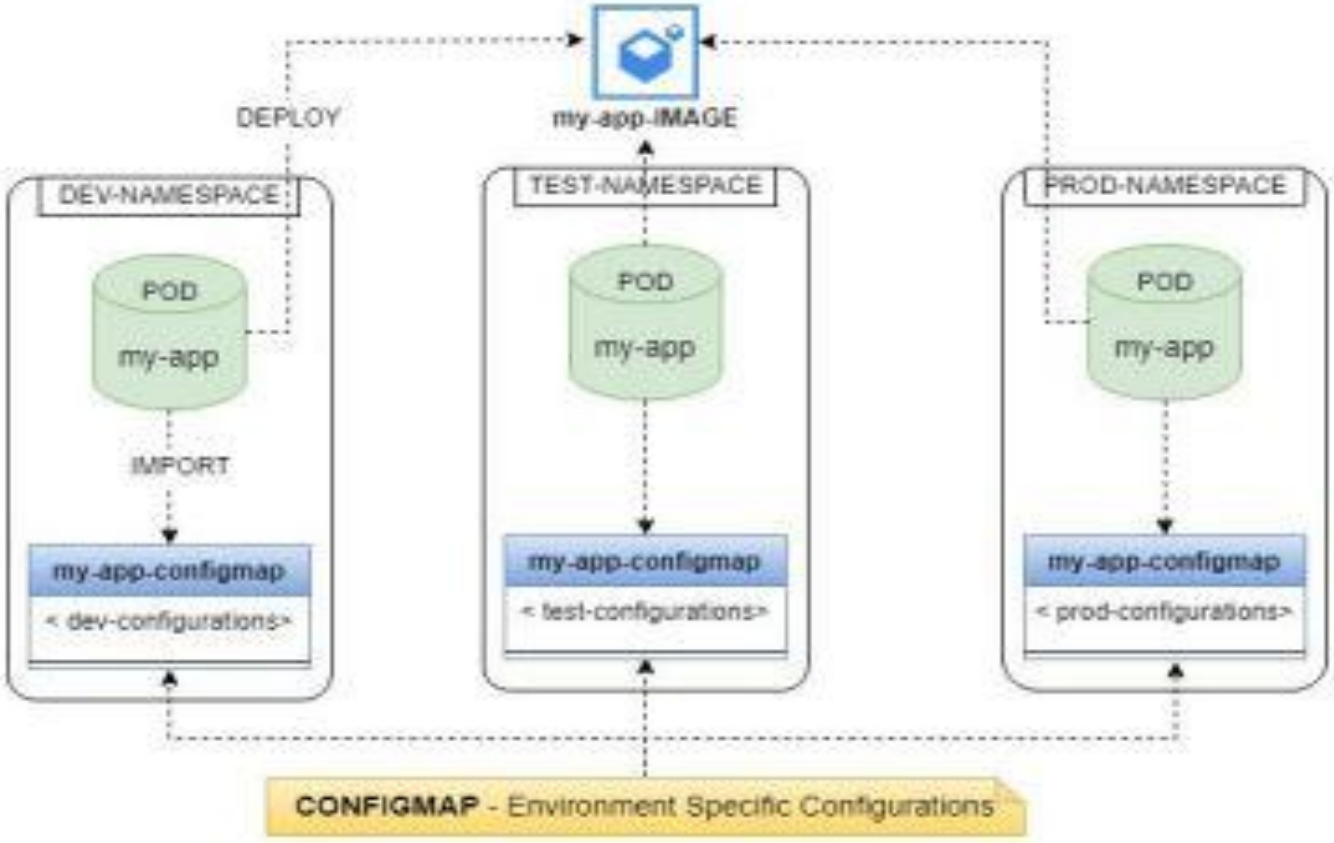
HEPAPI
DIGITAL TRANSITION PARTNER

ConfigMap is an API object used to store non-confidential data in key-value pairs. Pods can consume ConfigMaps as environment variables, command-line arguments, or as configuration files in a volume.

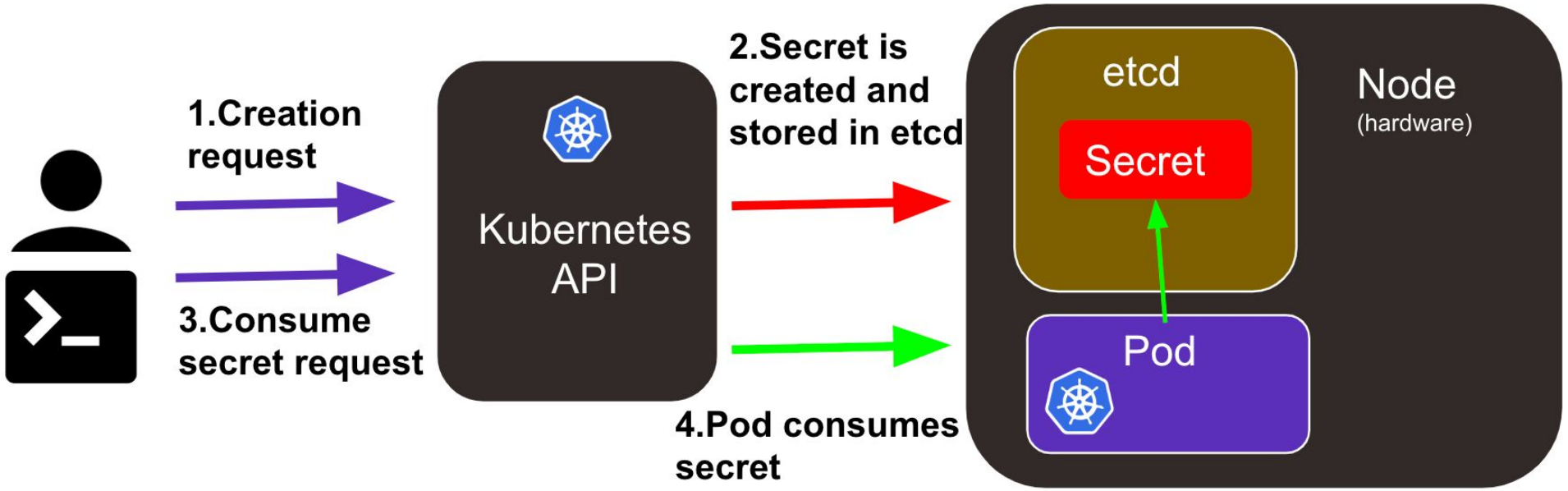


ConfigMap

A ConfigMap allows you to decouple environment-specific configuration from your container images, so that your applications are easily portable.



Secrets



A Secret is an object that contains a small amount of sensitive data such as a password, a token, or a key. Such information might otherwise be put in a Pod specification or in an image.

HEPAPI
DIGITAL TRANSITION PARTNER

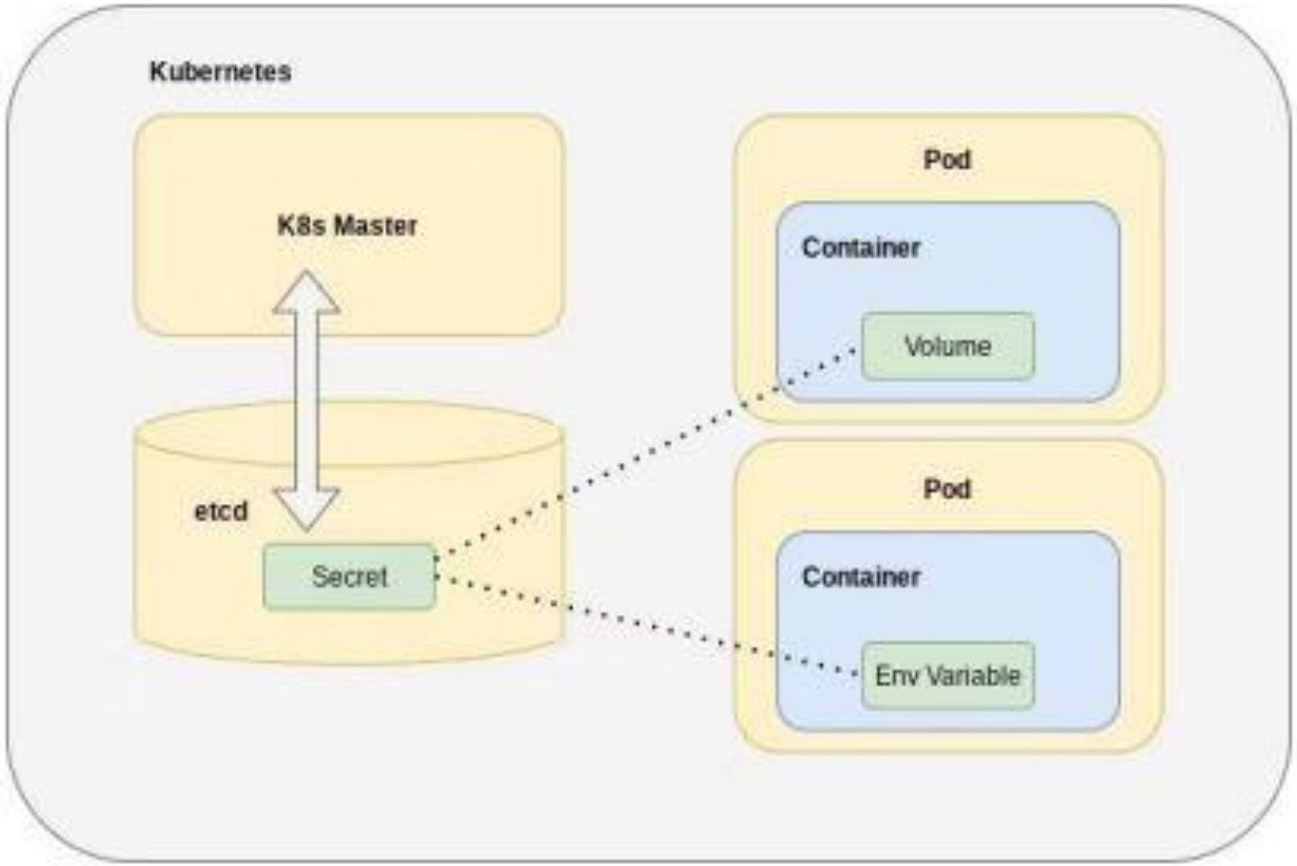
Secrets

Opaque is the default Secret type if omitted from a Secret configuration file. When you create a Secret using kubectl, you will use the generic subcommand to indicate an Opaque Secret type.

Built-in Type	Usage
Opaque	arbitrary user-defined data
kubernetes.io/service-account-token	ServiceAccount token
kubernetes.io/dockercfg	serialized ~/.dockercfg file
kubernetes.io/dockerconfigjson	serialized ~/.docker/config.json file
kubernetes.io/basic-auth	credentials for basic authentication
kubernetes.io/ssh-auth	credentials for SSH authentication
kubernetes.io/tls	data for a TLS client or server
bootstrap.kubernetes.io/token	bootstrap token data

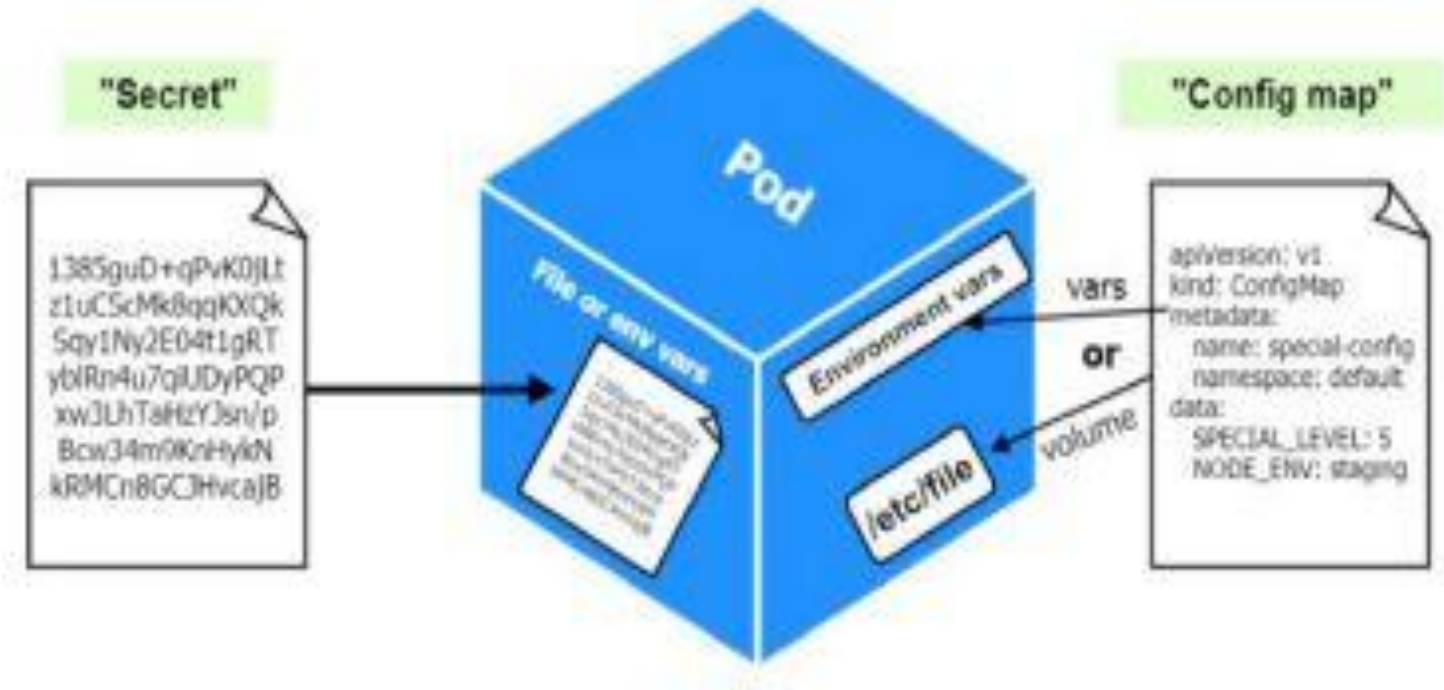
Secrets

Secrets can be mounted as data volumes or exposed as environment variables to be used by a container in a Pod.



HEPAPI
DIGITAL TRANSITION PARTNER

ConfigMap and Secret



HEPAPI
DIGITAL TRANSITION PARTNER

ConfigMap and Secret

```
env:
  - name: APP_COLOR
    value: pink
```

```
env:
  - name: APP_COLOR
    valueFrom:
      configMapKeyRef:
```

```
env:
  - name: APP_COLOR
    valueFrom:
      secretKeyRef:
```

HEPAPI
DIGITAL TRANSITION PARTNER

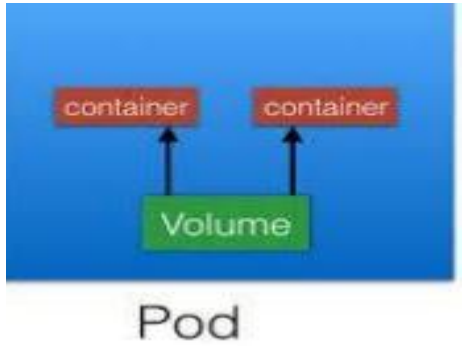
KUBERNETES Storage

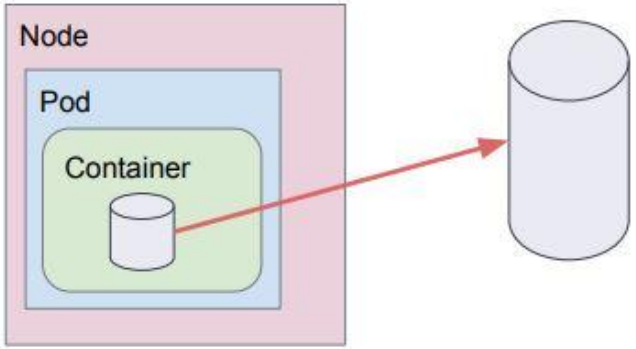
- Volumes
- Volume Types
- PersistentVolume & PersistentVolumeClaim
- StorageClass

Volumes

HEPAPI
DIGITAL TRANSITION PARTNER

- All data stored inside a container is deleted if the container crashes
- When a Container crashes, kubelet will restart it, but the files will be lost which means that it will not have any of the old data.
- To overcome this problem, Kubernetes uses Volumes. A Volume is essentially a directory backed by a storage medium. The storage medium, content and access mode are determined by the Volume Type.





HEPAPI
DIGITAL TRANSITION PARTNER

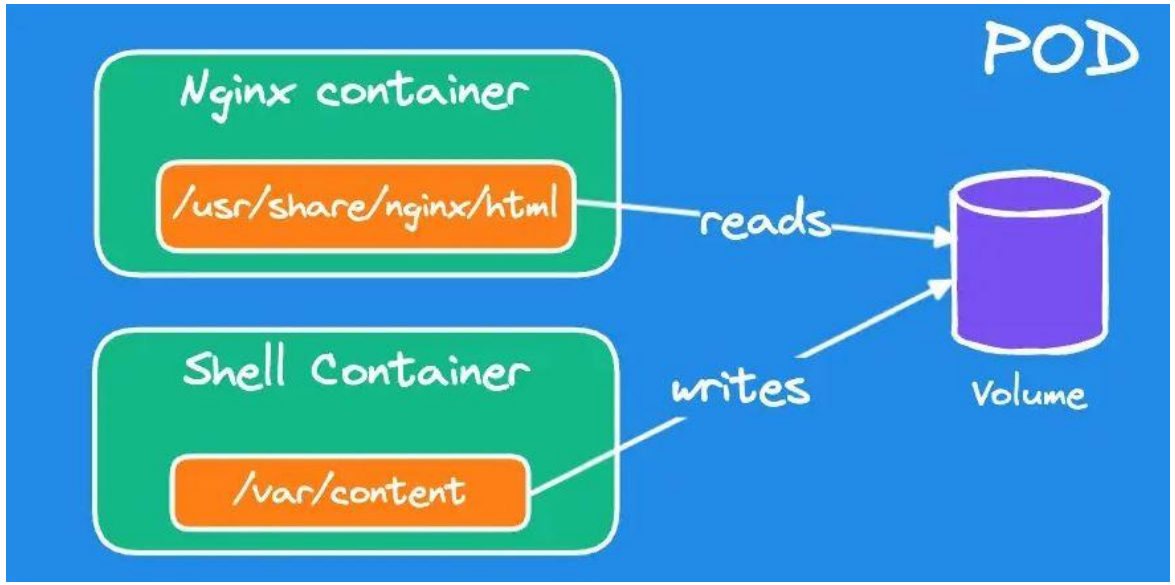
Volume Types

hostPath: A hostPath volume mounts a file or directory from the host node's file system into your Pod.

Volume Types

emptyDir: An emptyDir volume is first created when a Pod is assigned to a Node and exists as long as that Pod is running on that node.
Some uses for an emptyDir are:

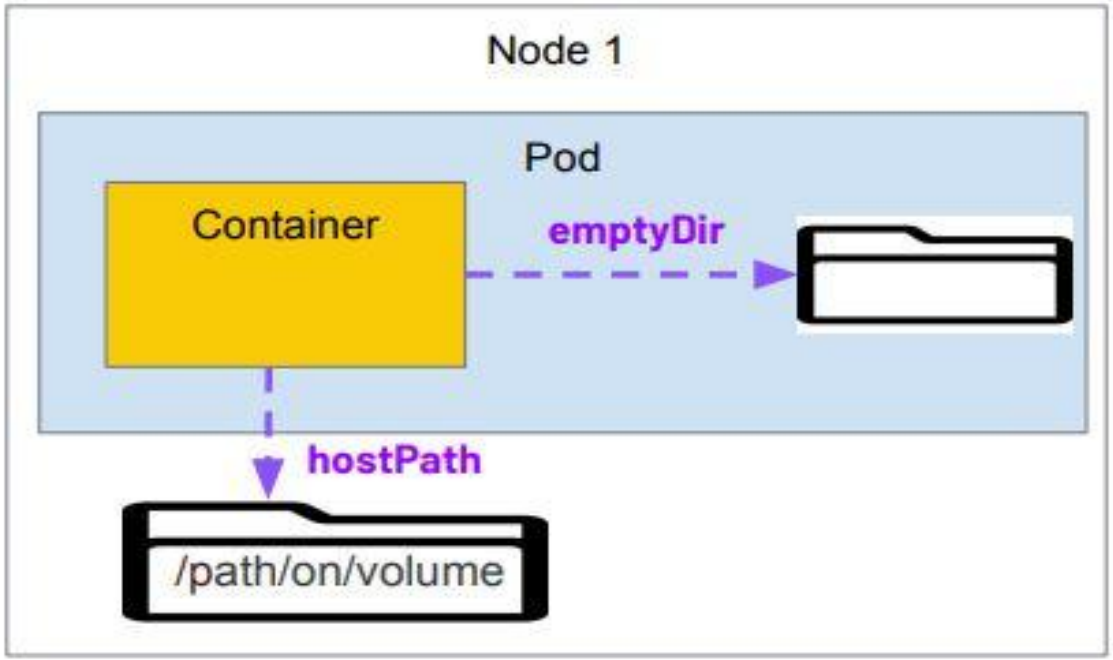
- checkpointing a long computation for recovery from crashes
- as a cache



HEPAPI
DIGITAL TRANSITION PARTNER

Volume Types

hostPath vs emptyDir



Volume Types

HEPAPI
DIGITAL TRANSITION PARTNER

Kubernetes provides several storage plugins that provide access to storage devices deployed in a Kubernetes cluster. These are implemented using the StorageClass object.

Some of the main plugins currently supported by Kubernetes are GCEPersistentDisk, AWSElasticBlockStore, AzureDisk, Glusterfs, NFS, and iSCSI.

AWS Elastic Block Store: An awsElasticBlockStore volume mounts an Amazon Web Services (AWS) EBS Volume into your Pod.

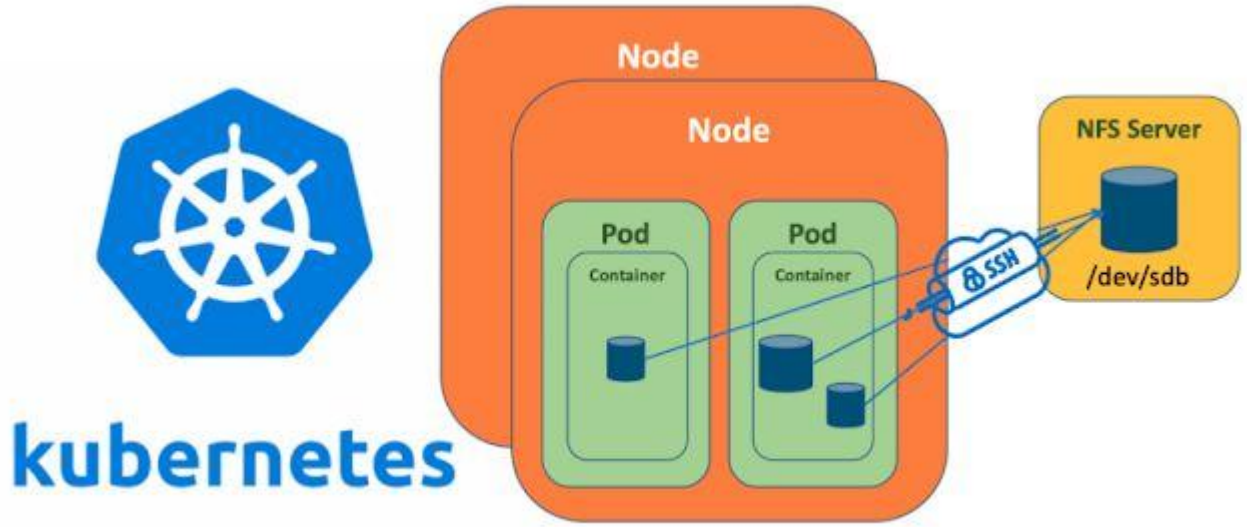
Azure Disk: An azureDisk is used to mount a Microsoft Azure Data Disk into a Pod.

GCE Persistent Disk: A gcePersistentDisk volume mounts a Google Compute Engine (GCE) persistent disk (PD) into your Pod

Volume Types

HEPAPI
DIGITAL TRANSITION PARTNER

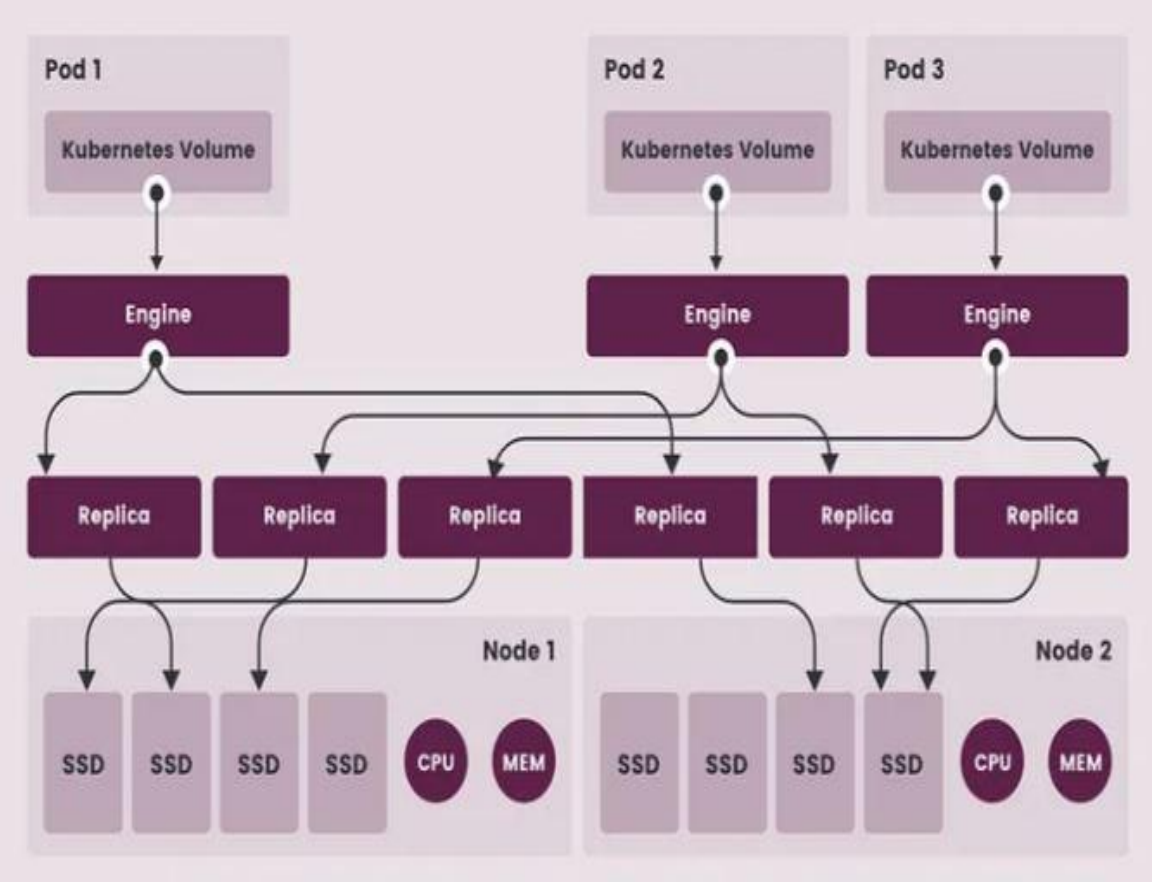
NFS: Network File System (NFS) is a standard protocol for mounting storage devices as local drives. Kubernetes lets you mount an NFS volume as a local drive in a container. Because legacy code often accesses data via NFS, this plugin is very useful for migrating legacy workloads to Kubernetes.



Volume Types

HEPAPI
DIGITAL TRANSITION PARTNER

Longhorn is an open-source distributed block storage system designed for Kubernetes. It provides persistent storage for containerized applications running on Kubernetes clusters..Longhorn supports volume snapshots, allowing you to capture the state of a volume at a specific point in time. It also provides backup functionality for disaster recovery purposes.Longhorn uses synchronous replication to ensure that data is stored on multiple nodes, providing high availability. If one node fails, the data is still accessible from replicas on other nodes.



HEPAPI
DIGITAL TRANSITION PARTNER

PersistentVolume

A PersistentVolume (PV) is a piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes

HEPAPI
DIGITAL TRANSITION PARTNER

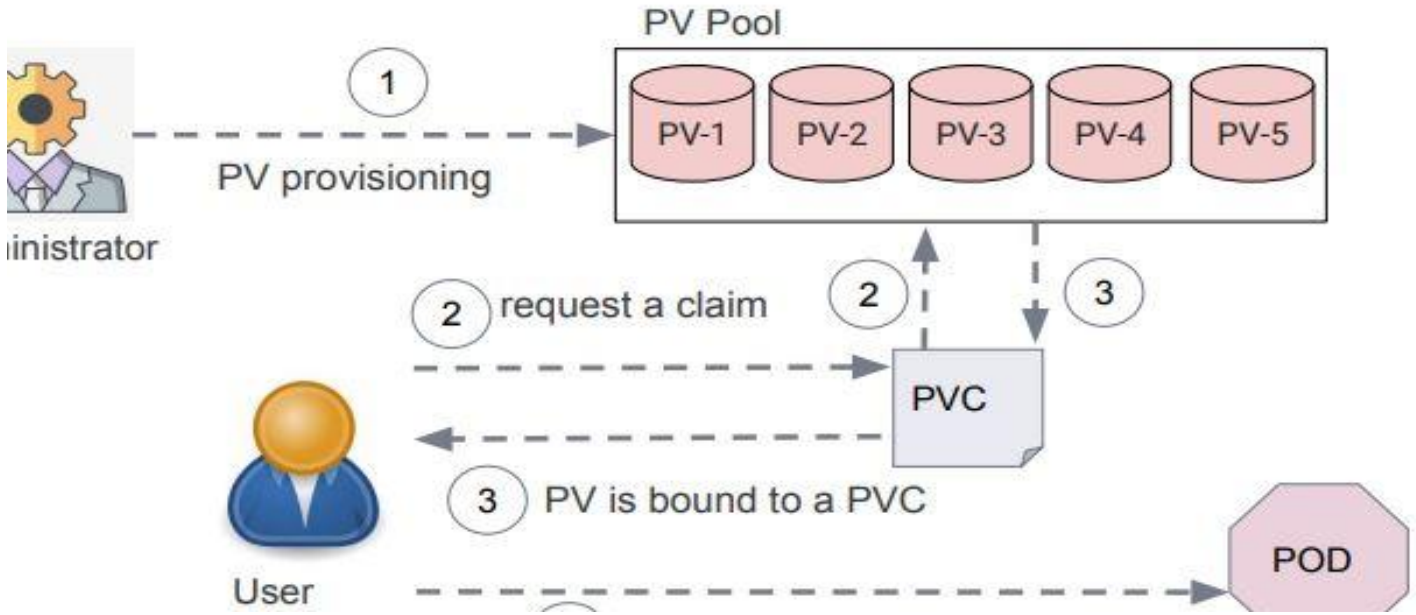
PersistentVolumeClaim

A PersistentVolumeClaim (PVC) is a request for storage by a user. Users request for PersistentVolume resources based on type, access mode, and size. There are four access modes:

- ReadWriteOnce (read-write by a single node)
- ReadOnlyMany (read-only by many nodes)
- ReadWriteMany (read-write by many nodes).
- ReadWriteOncePod (read-write only one pod in the cluster)

Once a suitable **PersistentVolume** is found, it is bound to a **PersistentVolumeClaim.**

PersistentVolumeClaim



HEPAPI
DIGITAL TRANSITION PARTNER

StorageClass

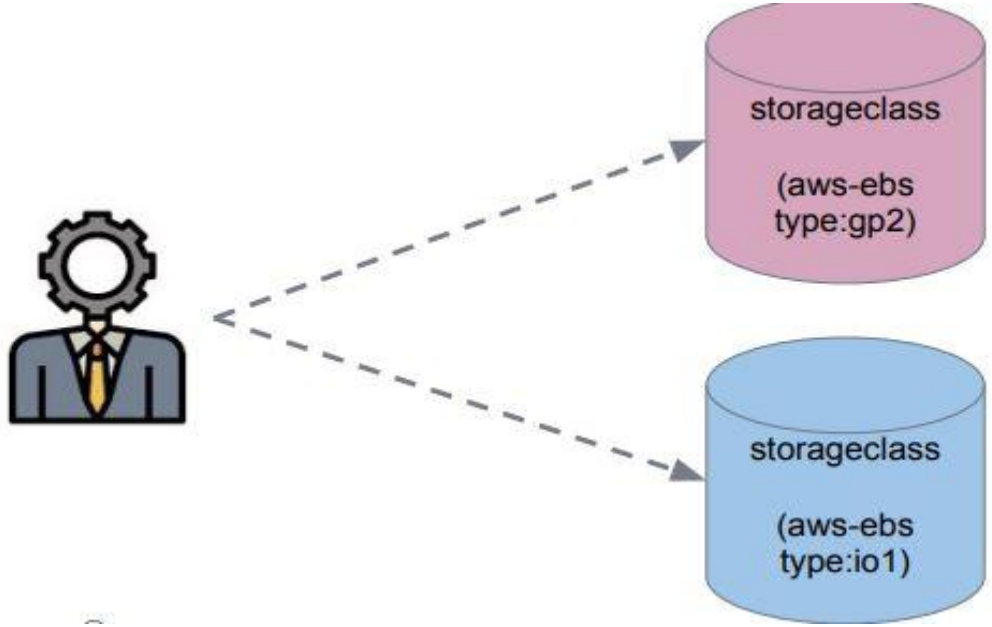
Provisioning: There are two ways PVs may be provisioned; statically or dynamically.

Static: A cluster administrator creates a number of PVs. They carry the details of the real storage, which is available for use by cluster users. They exist in the Kubernetes API and are available for consumption.

Dynamic: When none of the static PVs the administrator created match a user's PersistentVolumeClaim, the cluster may try to dynamically provision a volume specially for the PVC. This provisioning is based on **StorageClasses.**

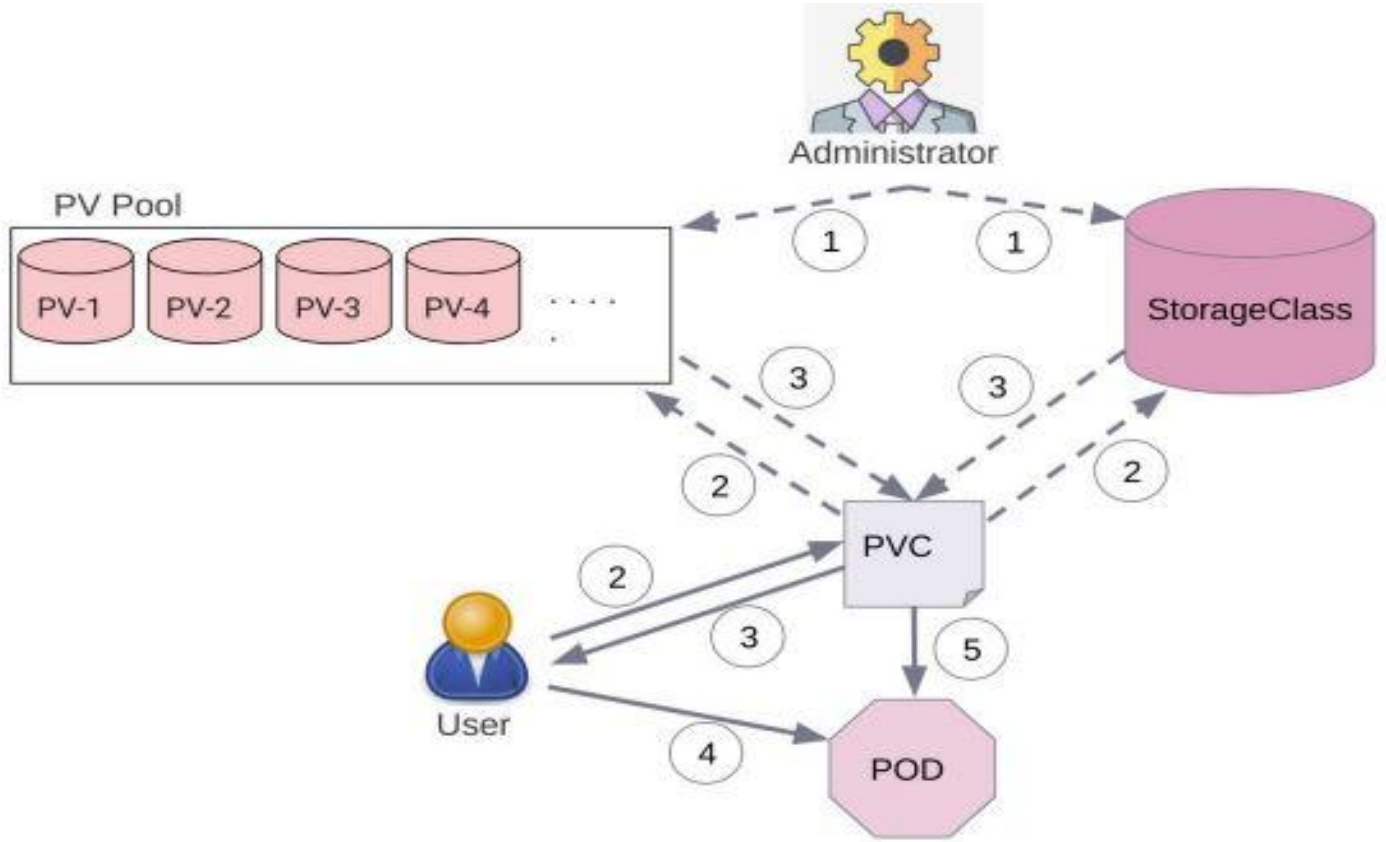
StorageClass

StorageClass: provides a way for administrators to describe the "classes" of storage they offer.



HEPAPI
DIGITAL TRANSITION PARTNER

StorageClass



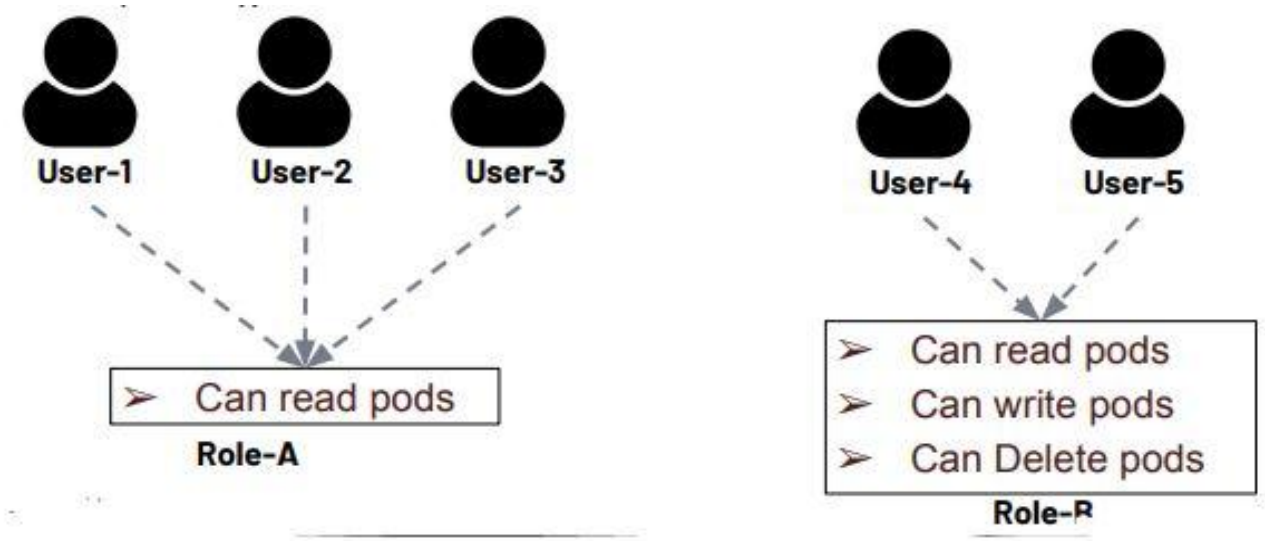
HEPAPI
DIGITAL TRANSITION PARTNER

KUBERNETES RBAC

- Role and ClusterRole
- RoleBindings and ClusterRoleBindings
- ServiceAccount

KUBERNETES RBAC

Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within your organization.



HEPAPI
DIGITAL TRANSITION PARTNER

KUBERNETES RBAC

Role and ClusterRole

RBAC Role or ClusterRole contains rules that represent a set of permissions

- **A Role** always sets permissions within a particular namespace; when you create a Role, you have to specify the namespace it belongs in

- **ClusterRole,** by contrast, is a non-namespaced resource

HEPAPI
DIGITAL TRANSITION PARTNER

KUBERNETES RBAC

RoleBinding and ClusterRoleBinding

- **A role binding** grants the permissions defined in a role to a user or set of users.

- **A RoleBinding** grants permissions within a specific namespace whereas a ClusterRoleBinding grants that access cluster-wide

HEPAPI
DIGITAL TRANSITION PARTNER

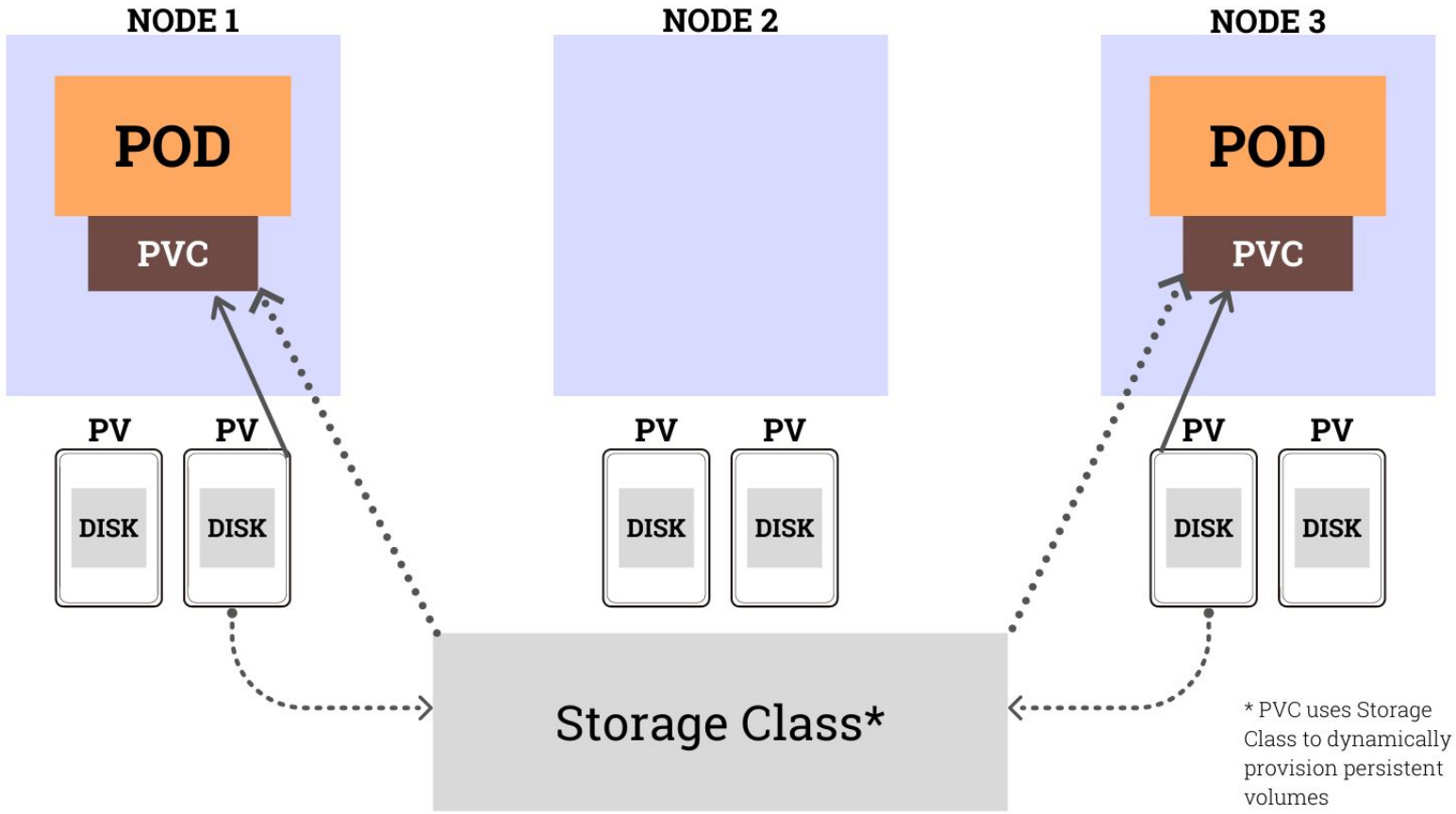
KUBERNETES RBAC

ServiceAccount

A ***service account*** provides an identity for processes that run in a Pod, and maps to a ServiceAccount object. When you authenticate to the API server, you identify yourself as a particular *user*.

Volumelar ve Depolama

- Kubernetes'te Kalıcı Depolama
- PersistentVolumes (PV): Bağımsız depolama birimleri
- PersistentVolumeClaims (PVC): Depolama kaynakları talep etmek için kullanılır
- StorageClasses: Sunulan depolama türleri



Helm ve Chartlar

HEPAPI
DIGITAL TRANSITION PARTNER

- Helm: Kubernetes Paket Yöneticisi
- Chartlar: Kubernetes uygulamaları için gerekli tüm kaynakları içeren Helm paketleri
- Avantajlar: Kubernetes uygulamalarının dağıtımını ve yönetimini basitleştirir

Best Practice'ler

HEPAPI
DIGITAL TRANSITION PARTNER

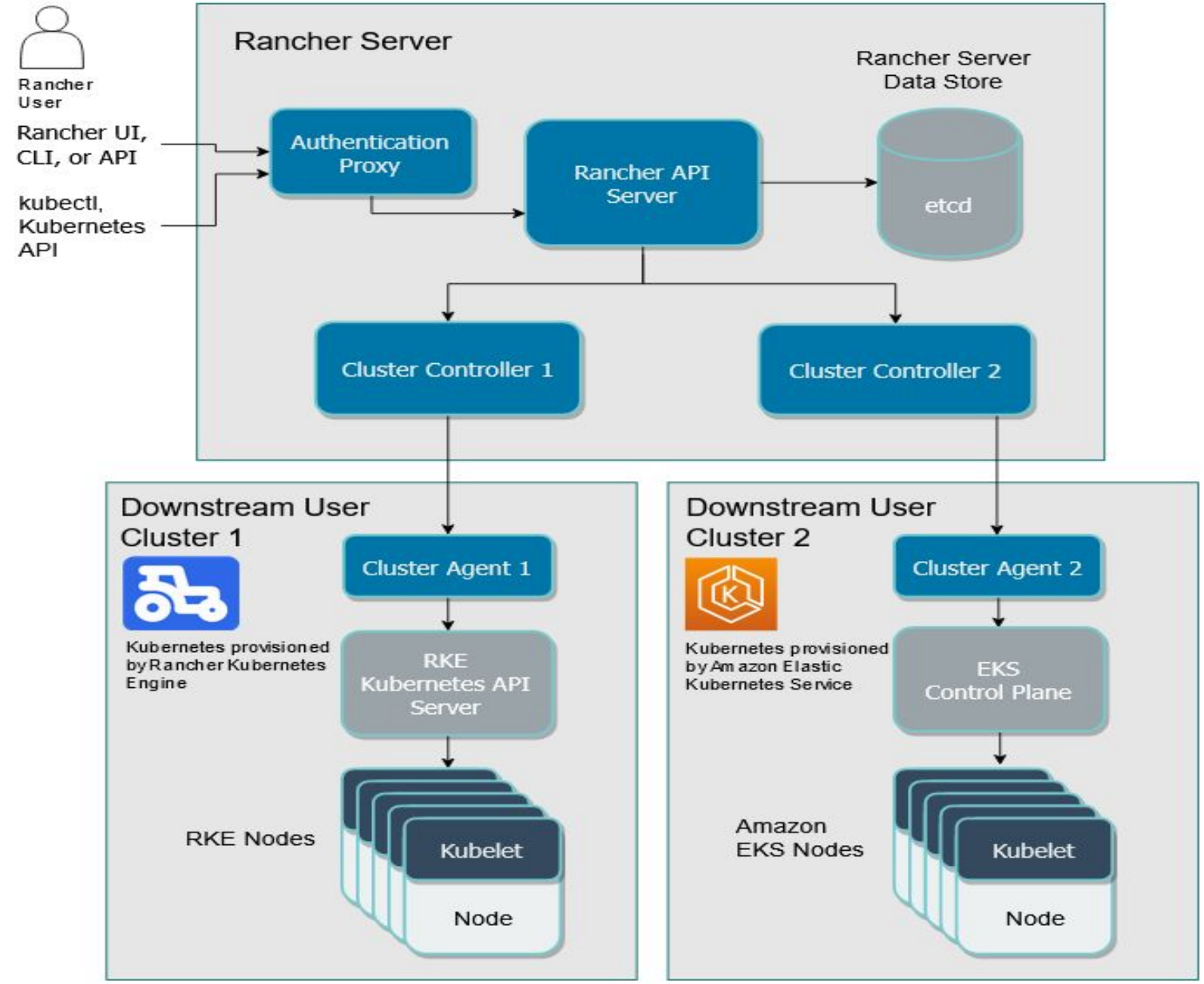
- Cluster Güvenliği: Rol Tabanlı Erişim Kontrolü (RBAC), Gizli Bilgilerin Yönetimi
- Etkili Kaynak Kullanımı: Podların doğru boyutlandırılması, Otomatik ölçeklendirme
- Düzenli Güncellemeler ve İzleme

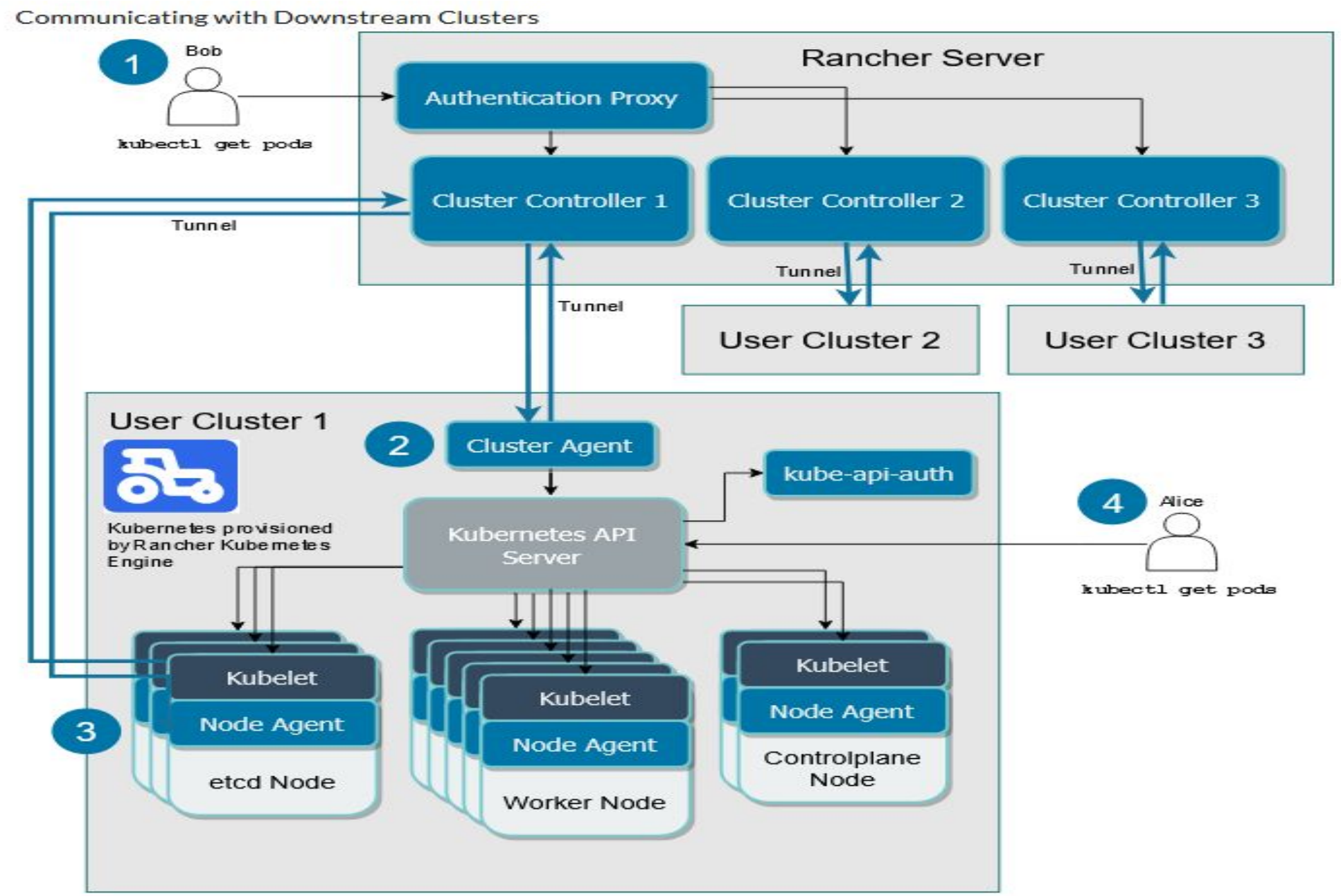
HEPAPI
DIGITAL TRANSITION PARTNER

RANCHER MANAGER MİMARİSİ

Rancher's authentication proxy:

Managing Kubernetes Clusters through Rancher's Authentication Proxy




Communicating with Downstream Clusters
1
Bob
kubectl get pods
Rancher Server
Authentication Proxy
Cluster Controller 1
Cluster Controller 2
Cluster Controller 3
Tunnel
Tunnel
Tunnel
Tunnel
User Cluster 2
User Cluster 3
User Cluster 1
2
Cluster Agent
kube-api-auth
Kubernetes provisioned by Rancher Kubernetes Engine
Kubernetes API Server
4
Alice
kubectl get pods
Kubelet
Node Agent
etcd Node
3
Kubelet
Node Agent
Worker Node
Kubelet
Node Agent
Controlplane Node

HEPAPI
DIGITAL TRANSITION PARTNER

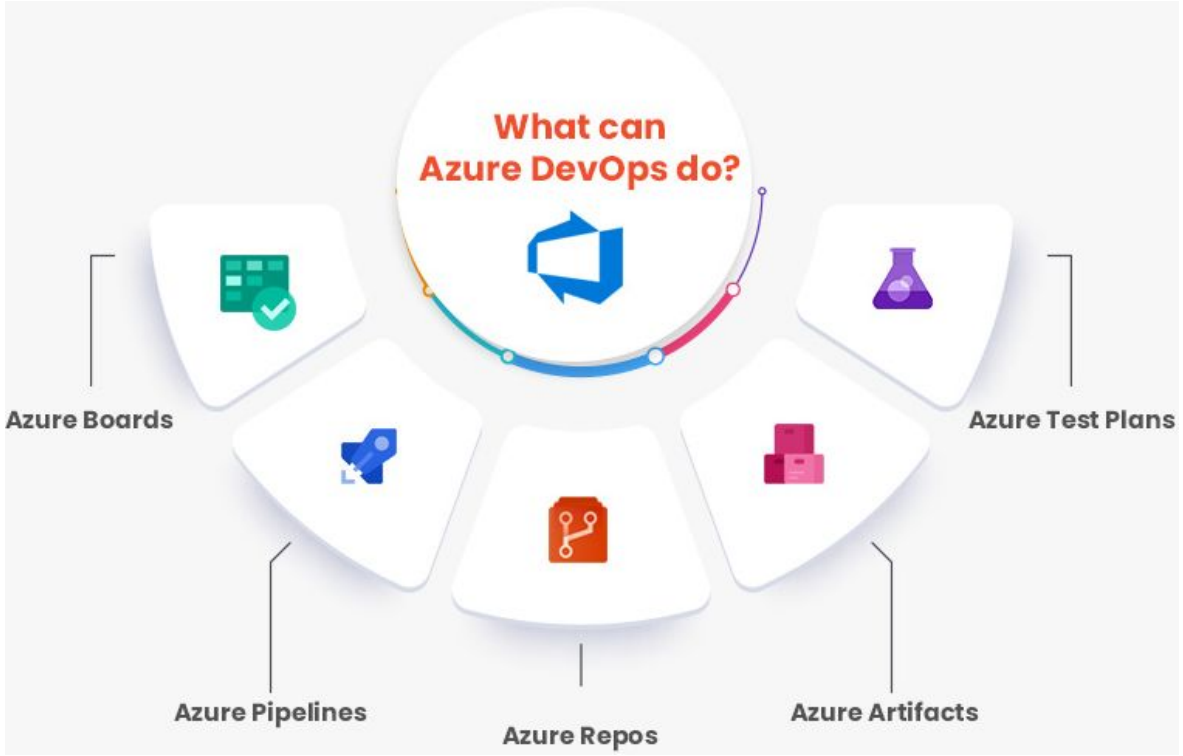
RANCHER MANAGER DEMO

HEPAPI
DIGITAL TRANSITION PARTNER

Azure DevOps Server

Azure DevOps'a Giriş

- Azure DevOps Server: CI/CD Süreçlerini Kolaylaştırma
- Bileşenler: Boards, Repos, Pipelines, Test Plans, Artifacts
- Faydaları: İş birliğini artırır, Deployment sıklığını artırır, Ürün kalitesini yükseltir



Azure Repos ve Git Entegrasyonu

HEPAPI
DIGITAL TRANSITION PARTNER

- Azure Repos: Sağlam Sürüm Kontrolü
- Git ile Sorunsuz Entegrasyon: Ortak kodlama kolaylaştırır
- Özellikler: Pull Requests, Branch Policies, Repositories Yönetimi

Azure Pipelines: CI/CD Temelleri

HEPAPI
DIGITAL TRANSITION PARTNER

- Azure Pipelines: Otomatik Yapı, Test ve Deployment
- Ana Özellikler: Cross-platform, Popüler diller ve platformlarla entegrasyon
- Baştan sona otomasyon: Koddan buluta

Azure Test Planları ve Test

HEPAPI
DIGITAL TRANSITION PARTNER

- Azure Test Planları: Test Etkinliğini Artırma
- Manuel ve Otomatik Testler: Test planlama, yürütme ve takip
- CI/CD ile Entegrasyon: DevOps döngüsünde Sürekli Test

Azure Artifacts ve Paket Yönetimi

HEPAPI
DIGITAL TRANSITION PARTNER

- Azure Artifacts: Paket Yönetimini Basitleştirme
- Bağımlılıkları Yönetme: Paketleri barındırma ve paylaşma
- NuGet, npm, Maven ve daha fazlasını destekler

HEPAPI
DIGITAL TRANSITION PARTNER

DİNLEDİĞİNİZ İÇİN TEŞEKKÜR EDERİZ

www.hepapi.com

info@hepapi.com

284 Chase Road,
London, N14 6HF,
United Kingdom

Büyükdere Caddesi
No:199, Levent 199,
34330 İstanbul, Turkey