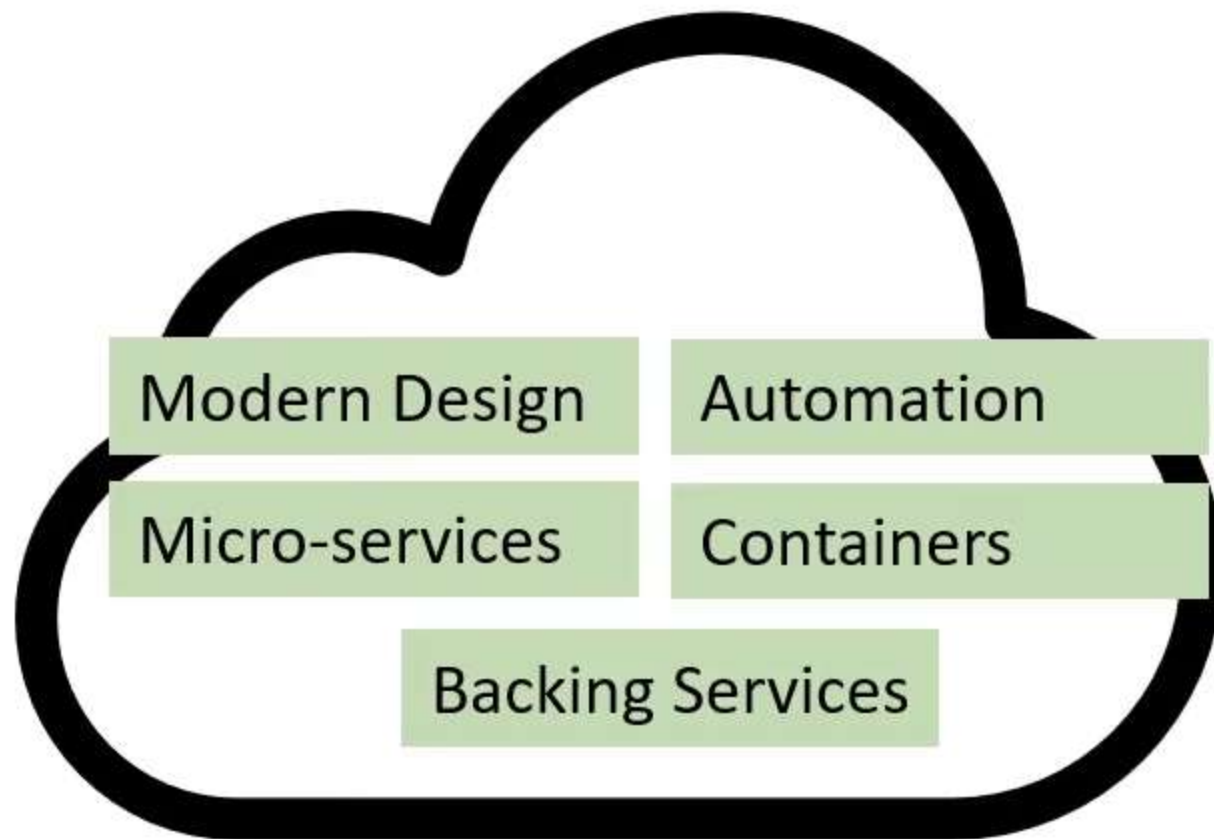


What is a Cloud-Native?

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

Cloud-native describes an architectural approach that emphasizes application workloads that are ***portable, modular and isolate**, between different cloud deployment models and Cloud Service Providers (CSP)

CSPs commonly describe Cloud-Native as a term meaning everything built on the CSP.
This is more so better described as being called “Cloud-First”



Some describe Cloud-Native being 4 key principles:

1. Microservices
2. Containerization
3. Continuous Delivery
4. DevOps

In the context of this course, Cloud-Native will mean technologies like Kubernetes and CNCF projects that are both distributed and are agnostic to any Cloud Service Provider (CSPs)

What is a Cloud-Native?

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

Cloud Native technologies empower organizations to build and run scalable applications in modern, dynamic, environments, such as public, private and hybrid clouds. Containers, services, meshes, microservices, immutable infrastructure, and declarative APIs exemplify this approach.

These techniques enable loosely coupled systems that are resilient, manageable and observable.

Combined with robust automation, they allow engineers, to make high impact changes, frequently and predictably with minimal toll

The Cloud Native Foundation seeks to drive adoption of this paradigm by fostering and sustaining an ecosystem of open source, vendor neutral projects.

We democratize state-of-the-art patterns to make these innovations accessible for everyone

“Cloud Native” definition described by the CNCF



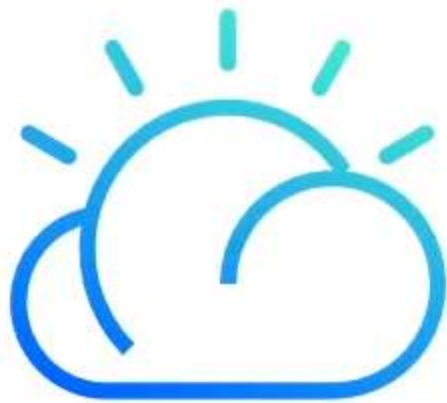
Cloud Native vs Cloud Service Providers

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

A Cloud Service Provider (CSP):

- A collection of cloud services
- Strong application integration and synergies between services
- Utilizing metered billing
- Under a single unified API

Cloud-Native is a workload, application or system, that is designed to run on cloud services, and “takes advantage of cloud offerings”



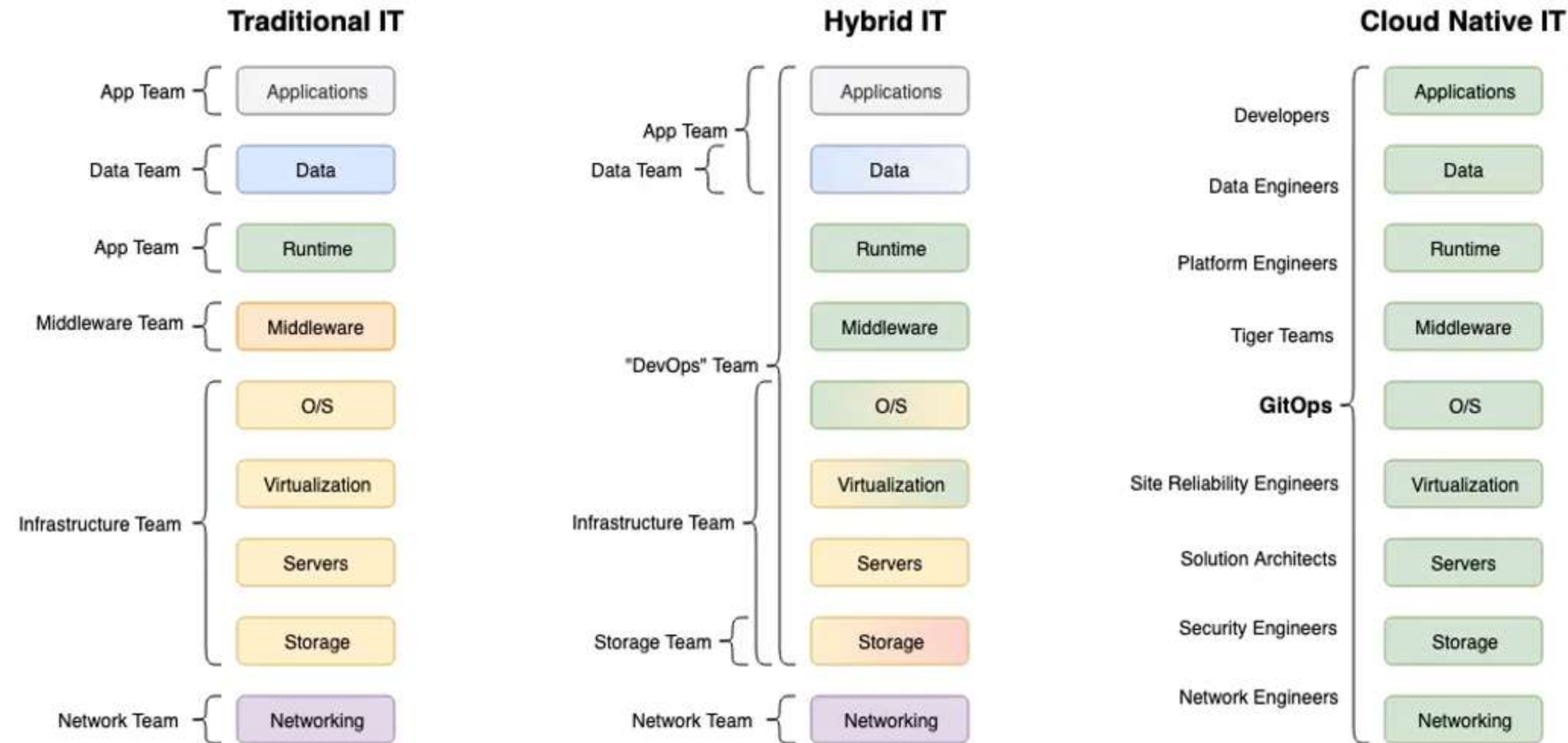
Cloud-Native workloads cannot take advantage of all or full the advantage of CSPs because CSPs intentionally have proprietary technology to encourage you to use their managed services that have exclusive integrations with other services.

Cloud Native Shared Responsibility Model

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampopro.co/kcna

Organisational Cloud Native Shared Responsibility Model

Copyright © 2020 Lachlan White



Lachlan White Cloud-Native Model
@ LachieWhite7



On Premise → Infrastructure as a Service → Platform as a Service → Containers as a Service → Functions As a Service

Reduced Risk of Vendor and Technology Lock-In
Reduce Cost of Innovation / Development
Reduce Time to Market

High Barrier To Entry	→	Large Overlaps	→	Low Barrier of Entry
Costly to Change		Battles for Control		Inexpensive to Change
Scaling Limitations		Hard to Communicate		Exponential Scaling
Closed to Innovation		Deep Coupling		Open to Innovation

The Linux Foundation

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampro.co/kcna



The Linux Foundation (LF) is a non-profit technology consortium founded in 2000 as a merger between Open Source Development Labs and the Free Standards Group to standardize Linux, support its growth, and promote its commercial adoption.

The Linux Foundation is supported by a variety of technology companies:



Cloud Native Computing Foundation

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



Cloud Native Computing Foundation (CNCf) is a **Linux Foundation project** that was founded in 2015 to help advance container technology.

CNCf operates as an **independent** organization from its parent organization



CNCf has its own board members



CNCf has its own global tech conference called: CloudNativeCon + KubeCon



CNCf has its own cloud native certifications



CNCf has its own **collection of projects**:

- **Kubernetes**
- Prometheus
- Etcd
- ContainerD
- And more.....

Cloud Native Trail Map

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampopro.co/kcna

The Cloud Native Trail Map is a recommended **path to adopting Cloud-Native** architecture

1. Containerization
2. Continuous Integration and Deployment (CI/CD)
3. Orchestration and Application Definition
4. Observability and Analysis
5. Service Proxy, Discovery and Mesh
6. Networking Policy, and Security
7. Distributed Databases and Storage
8. Streaming and Messaging
9. Container Registry and Runtime
10. Software Distribution

- A. Training and Certification
- B. Consulting Help
- C. Join CNCF End User Community



The path order will vary based on your use-case.
All parts of the path will be trekked during adoption.



CLOUD NATIVE TRAIL MAP

The Cloud Native Landscape [/landscape.cncf.io](https://landscape.cncf.io) has a large number of options. This Cloud Native Trail Map is a recommended process for leveraging open source, cloud native technologies. At each step, you can choose a vendor-supported offering or do it yourself, and everything after step #3 is optional based on your circumstances.

HELP ALONG THE WAY

A. Training and Certification

Consider training offerings from CNCF and then take the exam to become a Certified Kubernetes Administrator or a Certified Kubernetes Application Developer cncf.io/training

B. Consulting Help

If you want assistance with Kubernetes and the surrounding ecosystem, consider leveraging a Kubernetes Certified

Community

For companies that don't offer cloud native services externally

1. CONTAINERIZATION

- Commonly done with Docker containers
- Any size application and dependencies (even PDP-11 code running on an emulator) can be containerized
- Over time, you should aspire towards splitting suitable applications and writing future functionality as microservices

3. ORCHESTRATION & APPLICATION DEFINITION

- Kubernetes is the market-leading orchestration solution
- You should select a Certified Kubernetes Distribution, Hosted Platform, or Installer: cncf.io/ck
- Helm Charts help you define, install, and upgrade even the most complex Kubernetes application



5. SERVICE PROXY, DISCOVERY, & MESH

- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing



2. CI/CD

- Setup Continuous Integration/Continuous Deployment (CI/CD) that changes to your source code, container being built, tested, and eventually, perhaps, to production
- Setup automated rollouts, roll back
- Argo is a set of Kubernetes-native tools for deploying and running jobs, applications, workflows, and events using Git paradigms such as continuous delivery and MLOps

4. OBSERVABILITY

- Pick solutions for monitoring, logging, and tracing
- Consider CNCF projects like Prometheus, Fluentd for logging and Jaeger for tracing
- For tracing, look for an OpenTelemetry implementation like Jaeger



6. NETWORKING & SECURITY

To enable more flexible network

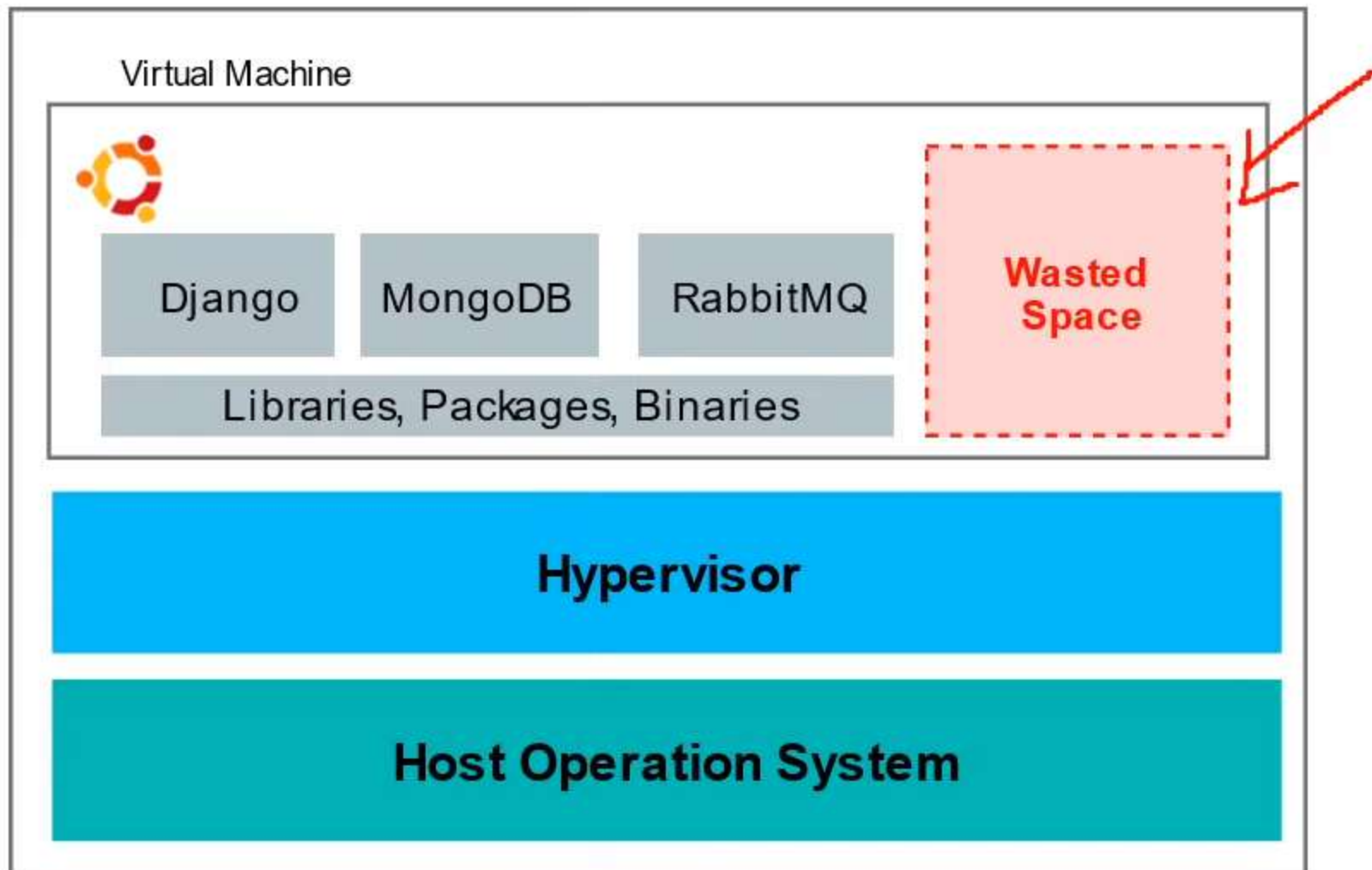
VMs vs Containers

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

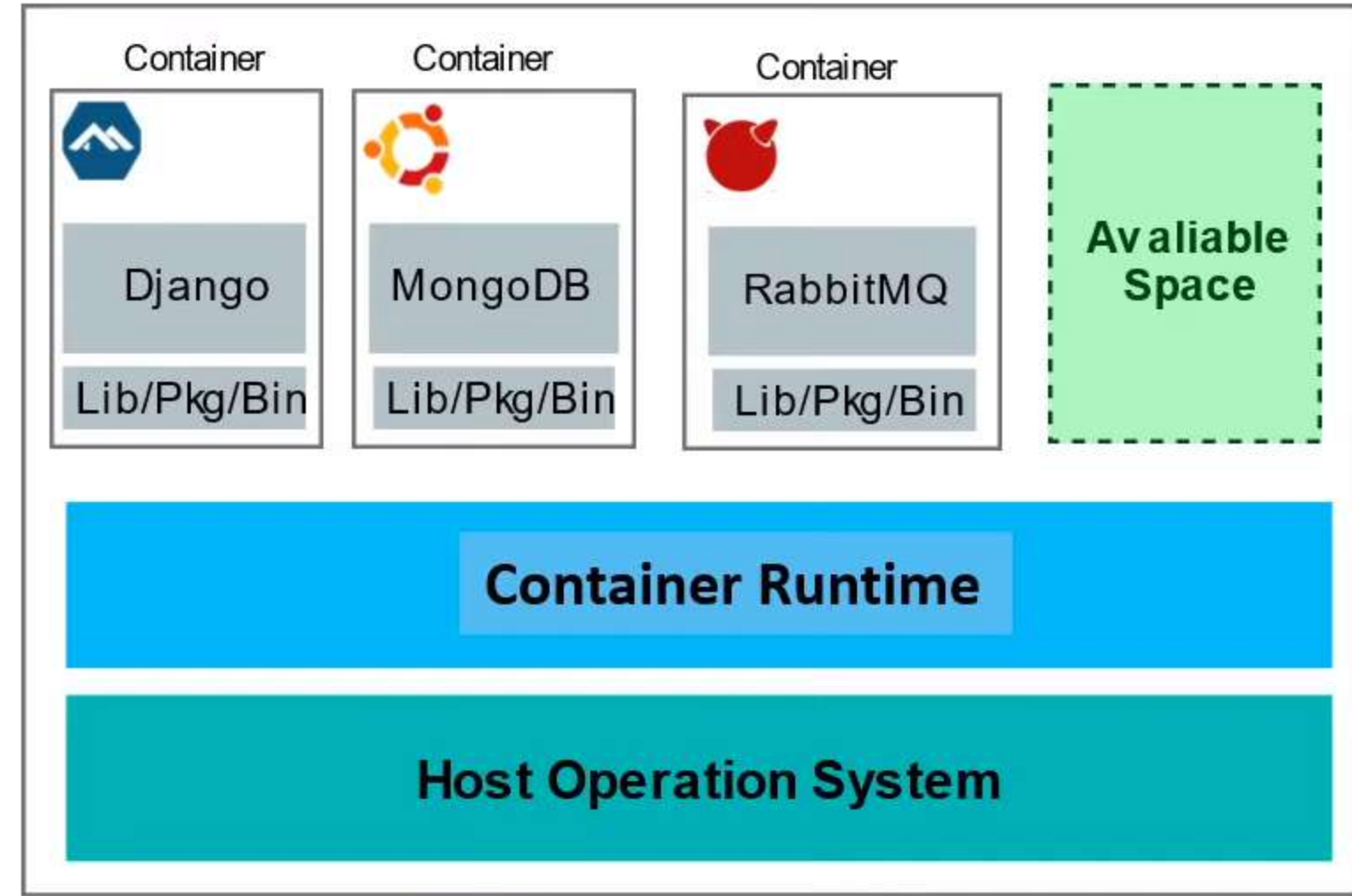
Virtual Machines (VMs) **do not** make the best use of space. Apps are not isolated which could cause **config conflicts**, **security problems** or **resource hogging**.

Containers allow you to run multiple apps which are virtually isolated from each other.

Launch new containers and configure OS Dependencies per container.



 EC2 Instance



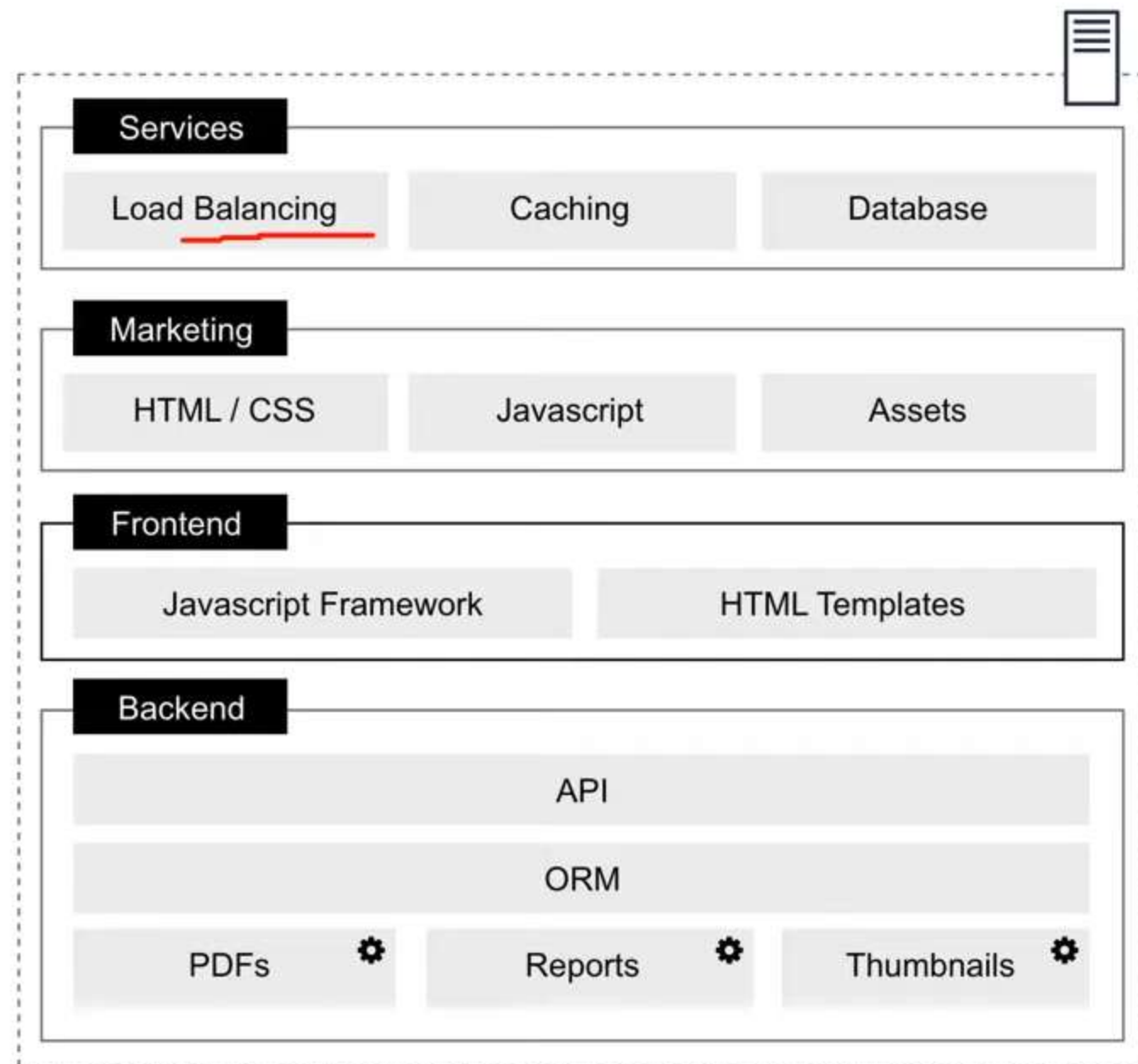
 EC2 Instance

What are Microservices

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

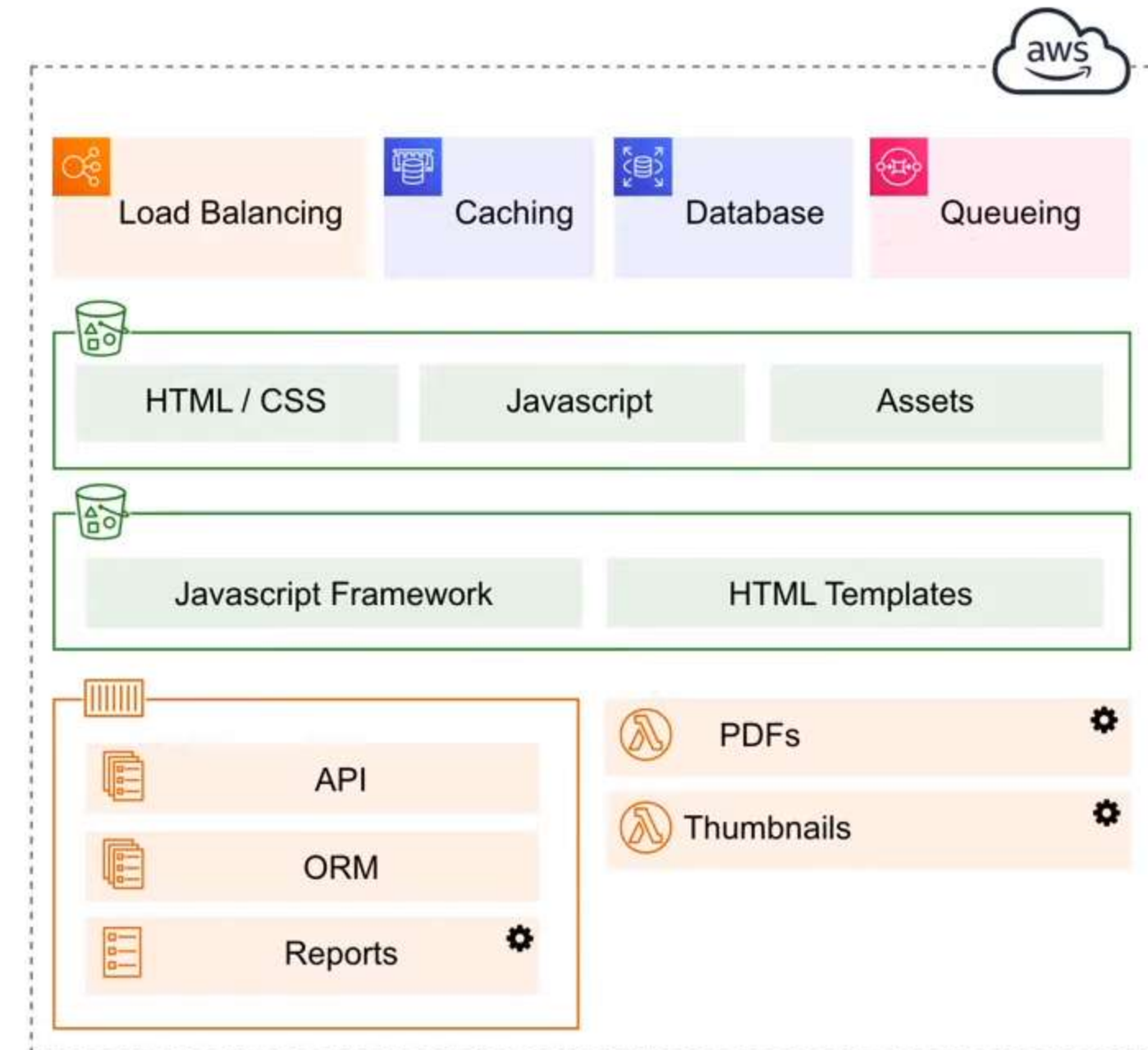
Monolithic Architecture

One app which is responsible for everything
Functionality is tightly coupled



Microservices Architecture

VS Multiple apps which are each responsible for one thing
Functionality is isolate and stateless



Kubernetes

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampopro.co/kcna



Kubernetes is an **open-source container orchestration system** for automating **deployment, scaling and management** of containers.



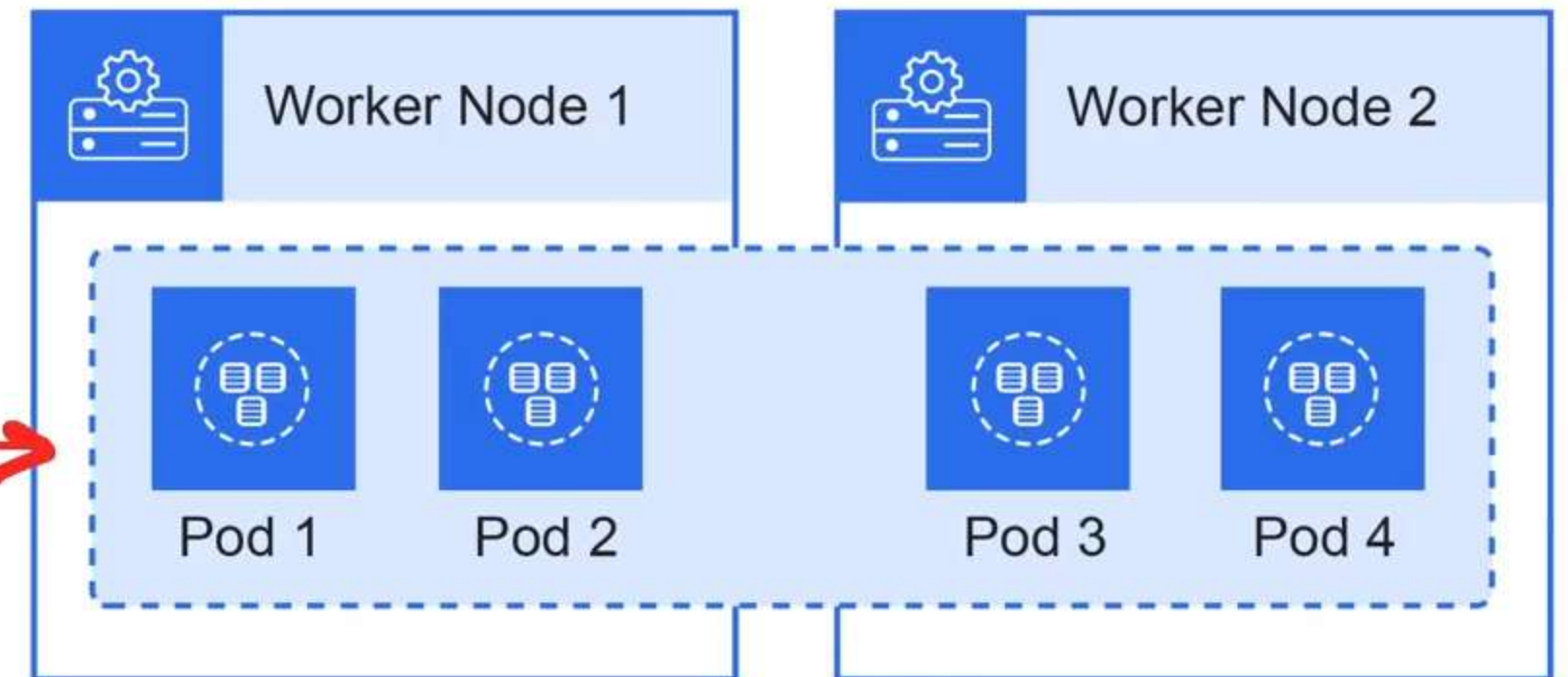
Originally created by Google and now maintained by the **Cloud Native Computing Foundation (CNCF)** as a **CNCF Project**

Kubernetes is commonly called **K8s**

- The 8 represent the remaining letters “ubernete”

****The advantage of Kubernetes over Docker is the ability to run “container apps” distributed across multiple VMs**

A unique component of Kubernetes are **Pods**.
A pod is a group of one more containers with shared storage, network resources and other shared settings.



Kubernetes is ideally for micro-service architectures where a company has tens to **hundreds of services** they need to manage

Kubernetes Components Overview

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



Cluster

A logical grouping of all components within a cluster.



Namespace

A named logical grouping of Kubernetes components within a cluster.
Used to Isolate different workloads on the same cluster



Node

A virtual machine or underlying server. There are two types of nodes: Control Plane and Worker nodes.
Worker nodes is where your application or workloads run. Control Plane nodes manage worker nodes.



Pod

The smallest unit in K8s. It is an **abstraction over a container.**
Generally defines an application workload



Service

A static IP address and DNS name for a set of pods (persists an address even if a pod dies) and a load balancer



A "service" can also mean a container that continuously runs.



Ingress

Translates HTTP/S rules to point to services

Kubernetes Components Overview

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



API Server

The API Server allows users to interact with K8s components using the KubeCTL or by sending HTTP requests.



Kubelet

Kubelet is an agent installed on all nodes. Kubelet allows users to interact with node via the API Server and KubeCTL



KubeCTL

A command line interface (CLI) that allows users to interact with the cluster and components via the API Server



Cloud Controller Manager

Allows you to link a Cloud Service Provider (CSP) eg. AWS, Azure GCP to leverage cloud services.



Controller Manager

A control loop that watches the state of the cluster and will change the current state back to desired state.



Scheduler

Determines where to place pods on nodes. Places them in a scheduling queue



Kube Proxy

An application on worker nodes that provides routing and filtering rules for ingress (incoming) traffic to pods.



Network Policy

Acts as a virtual firewall at the namespace-level or pod-level



Kubernetes Components

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



ConfigMap

allows you to decouple environment-specific configuration from your container images, so that your applications are easily portable. Used to store non-confidential data in key-value pair



Secret

small amount of sensitive data such as a password, a token, or a key



Volumes

mounting storage eg. locally on the node, or remote to cloud storage



StatefulSet

provides guarantees about the ordering and uniqueness of these Pods

- Think of databases where you have to determine read and write order or limit the amount of containers
- StatefulSets are hard, when you can host your db externally from K8s cluster



ReplicaSets

Maintain a stable set of replica pods running at a given time. Can provide a guarantee of availability.



Deployment

Is a blueprint for a pod (think Launch Template)

Manifest Files in Kubernetes

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampopro.co/kcna



Manifest (not technical description)

A Manifest file is a document that is commonly used for customs to list the contents of cargo, or passengers. Its an itemized list of things.

Manifest File (in the context of Kubernetes)

A Manifest file is a generalized name **for any Kubernetes Configuration File** that define the configuration of various K8s components.

These are all Manifest files with specific purposes:

- Deployment File
- PodSpec File
- Network Policy File

Manifest Files can be written in either:

- YAML
- JSON

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    ports:
    - containerPort: 80
```

YAML

```
{
  "apiVersion": "v1",
  "kind": "Pod",
  "metadata": {
    "name": "nginx"
  },
  "spec": {
    "containers": [
      {
        "name": "nginx",
        "image": "nginx:1.14.2",
        "ports": [
          {
            "containerPort": 80
          }
        ]
      }
    ]
  }
}
```

JSON

Manifest Files in Kubernetes

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampopro.co/kcna

A manifest can contain **multiple** K8s component definitions/configurations.

In YAML you can see the **three hyphens ---** is used to defined multiple components.

kubectl apply command is generally used to deploy manifest files

```
kubectl apply -f resources.yml
```

Resource Configuration file is sometimes used to describe multiple resources in a manifest.

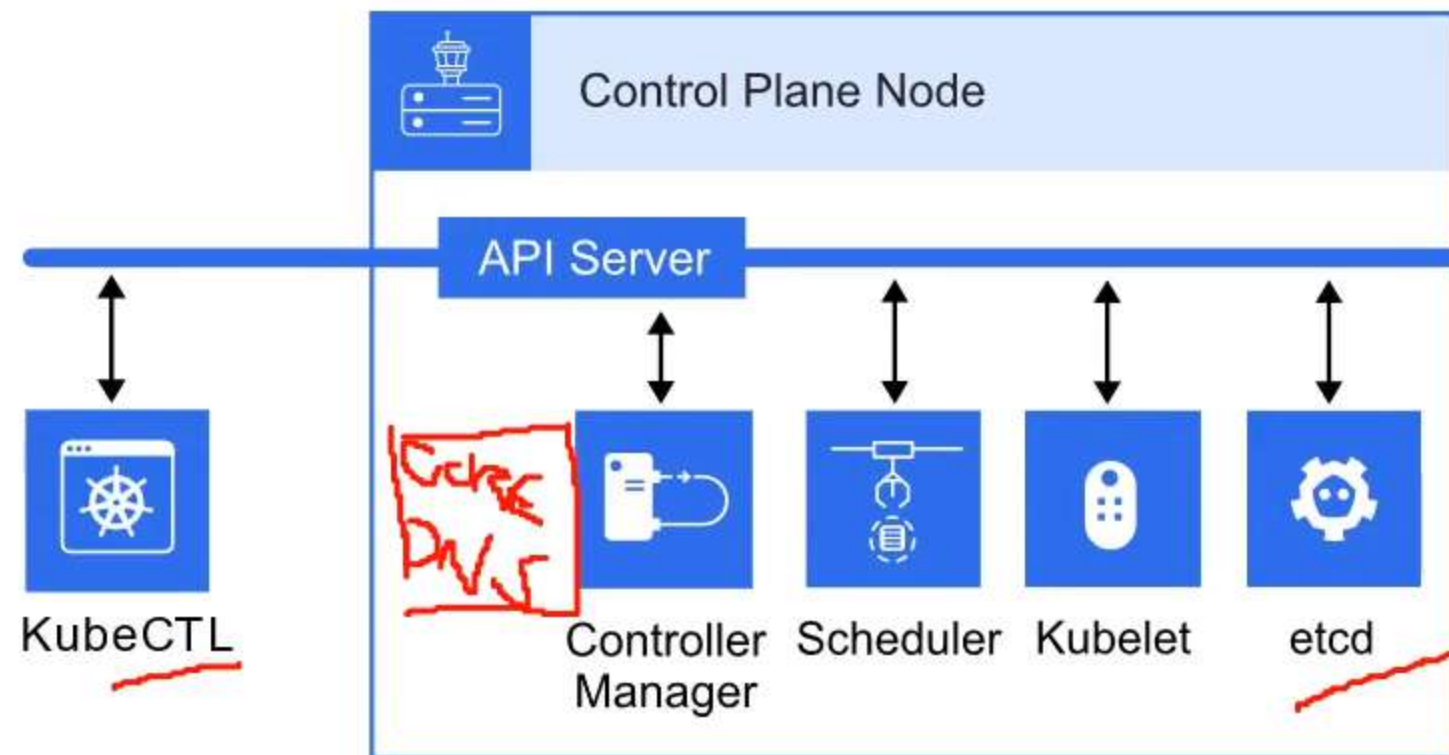
```
apiVersion: v1
kind: Service
metadata:
  name: my-nginx-svc
  labels:
    app: nginx
spec:
  type: LoadBalancer
  ports:
    - port: 80
  selector:
    app: nginx
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-nginx
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```


Nodes – Control Plane and Worker Nodes

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

Control Plane Node (Formally known as ~~Master Node~~)

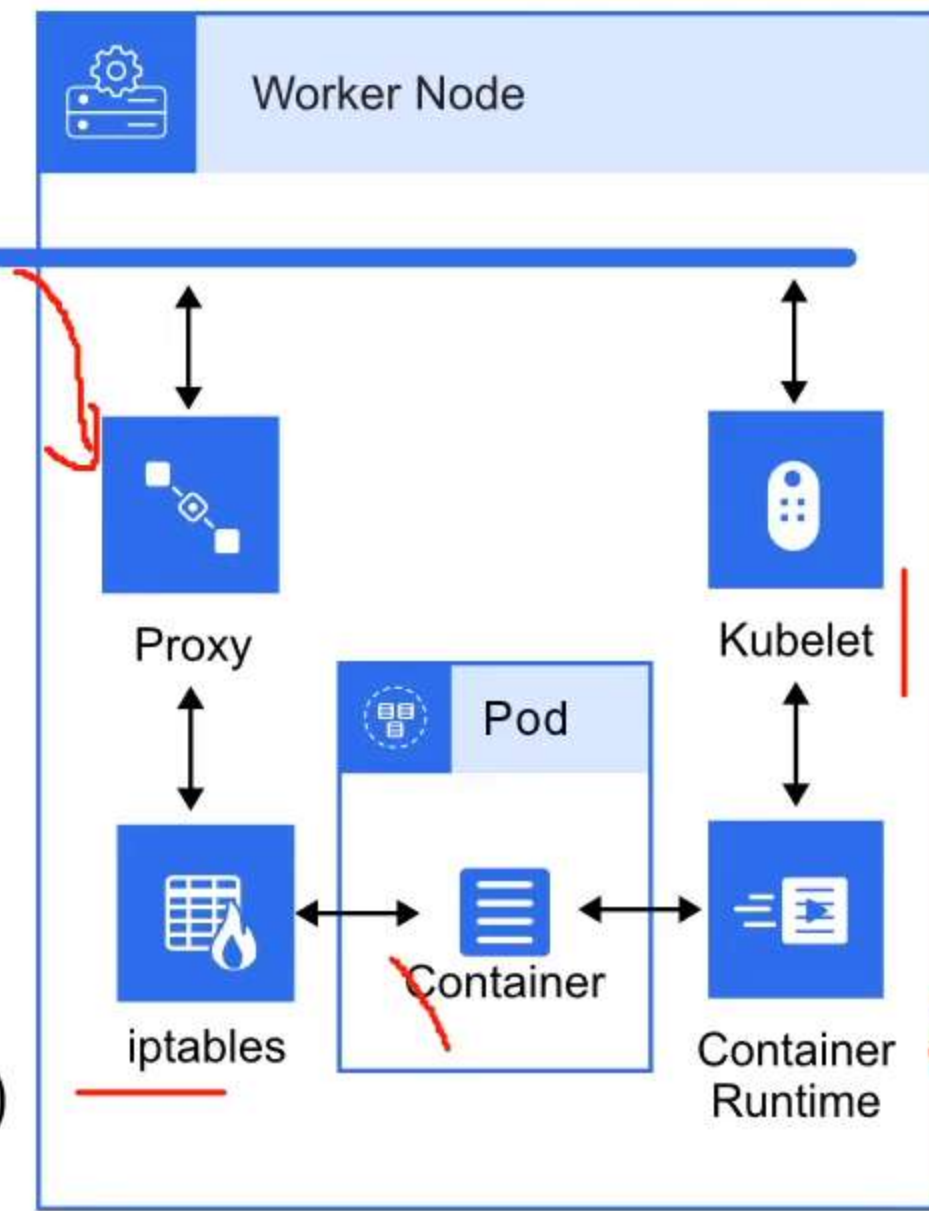
Manages processes like scheduling, restarting nodes...



- **API Server** – the backbone of communication
- **Scheduler** – determines where to start a pod on worker node
- **Controller Manager** – detect state changes (if pod crashes, restart it)
- **etcd** – A Key/Value Store that stores the state of the cluster
- **Kubelet** – Allows user to interact with the node via KubeCTL

Worker Node

Does the work, running your app in pods and containers...



The worker node runs:

- Kubelet
- Kube Proxy
- Container Runtime
- Pods and Containers

Pods

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampopro.co/kcna



Pods are the smallest unit in Kubernetes.
Pods abstract away the container layer so you can directly interact with the Kubernetes Layer.

A Pod is intended to run one application in multiple containers

- Database Pod, Job Pod, Frontend App Pod, Backend App Pod
- You can run multiple apps in a pod but those containers will tightly dependent.

Each Pod gets its own private IP address

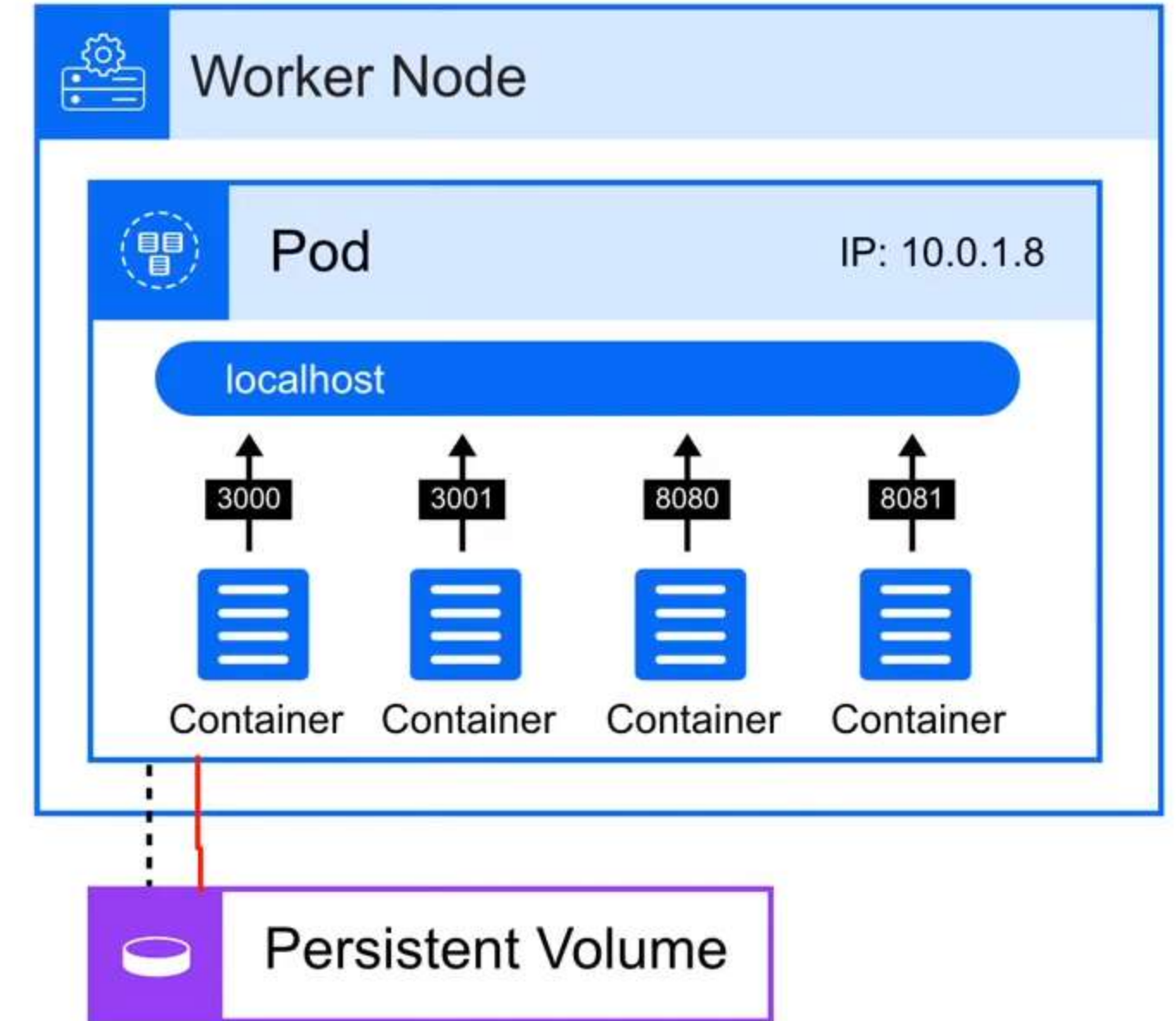
- Containers will run on different ports
- Containers can talk to each other via localhost

Each Pod can have a shared storage volume attached.

- All containers will share the same volume

When the last remaining container dies (maybe crashes) in a pod so does the pod

- When a replacement pod is created, the pod will have an IP address will be assigned.
 - IP addresses are Ephemeral, (temporary) for pods, they don't by default persist.



`kubectl get pod -o wide`

Get pods and **show their IP addresses**



API Server

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



The core of Kubernetes control plane is the API server

The API server exposes an HTTP API that lets end users, different parts of your cluster, and external components communicate with one another.

The Kubernetes API lets you query and manipulate the state of API objects in Kubernetes (for example: Pods, Namespaces, ConfigMaps, and Events).

The API server is a component of the Kubernetes control plane that exposes the Kubernetes API. The API server is the front end for the Kubernetes control plane.

The main implementation of a Kubernetes API server is kube-apiserver.

kube-apiserver is designed to scale horizontally—that is, it scales by deploying more instances.

You can run several instances of kube-apiserver and balance traffic between those instances.

Everything has to **go through the API Server**.

You can interact with the API Server in three ways:

- UI*
- API
- CLI KubeCTL

Deployment

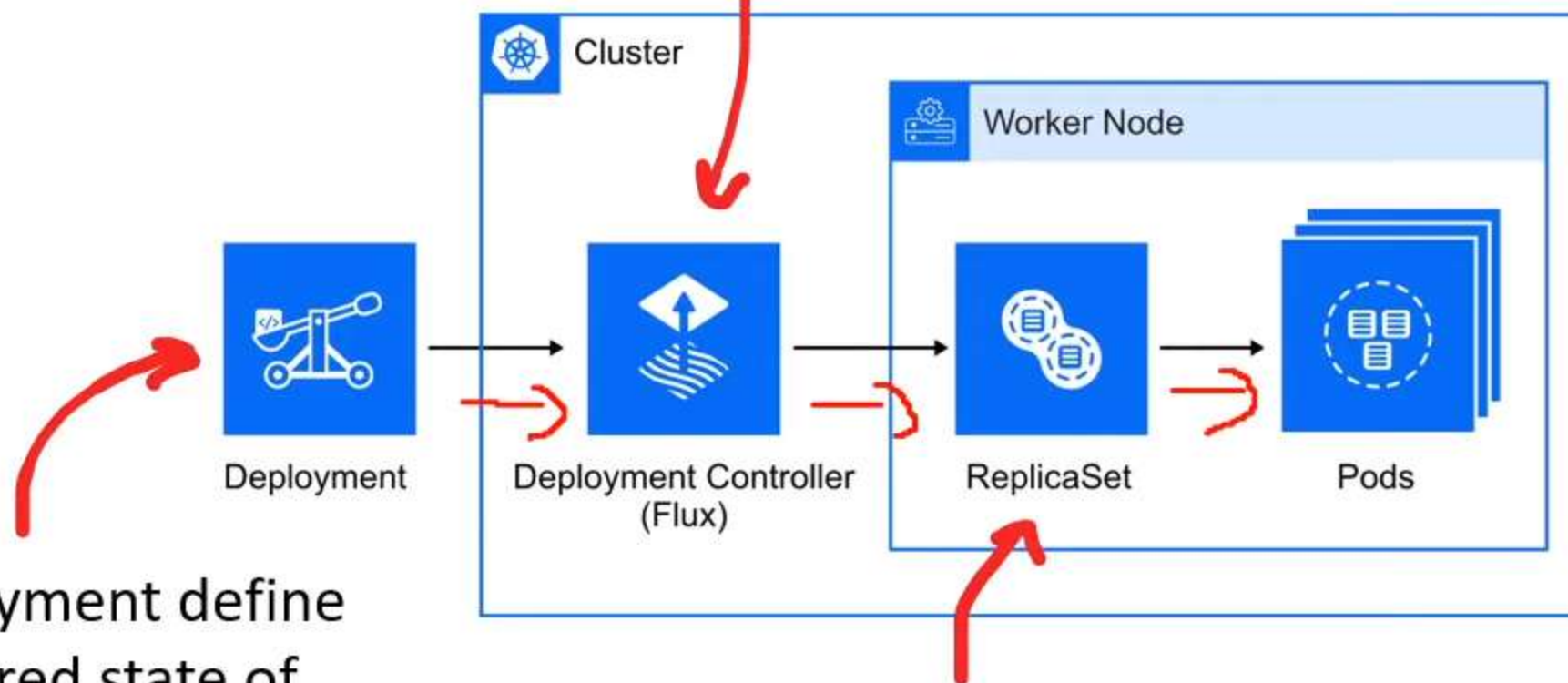
Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



A Deployment provides declarative updates for Pods and ReplicaSets.

A Deployment Controller changes the actual state to the desired state at a controlled rate.

- The default Deployment Controller can be swapped out for other deployments tools eg:
 - Argo CD, Flux, Jenkin X.....



A Deployment define the desired state of ReplicaSets and Pods.

A deployment will create and manage a ReplicaSet.
A ReplicaSet will manage replicas of pod.

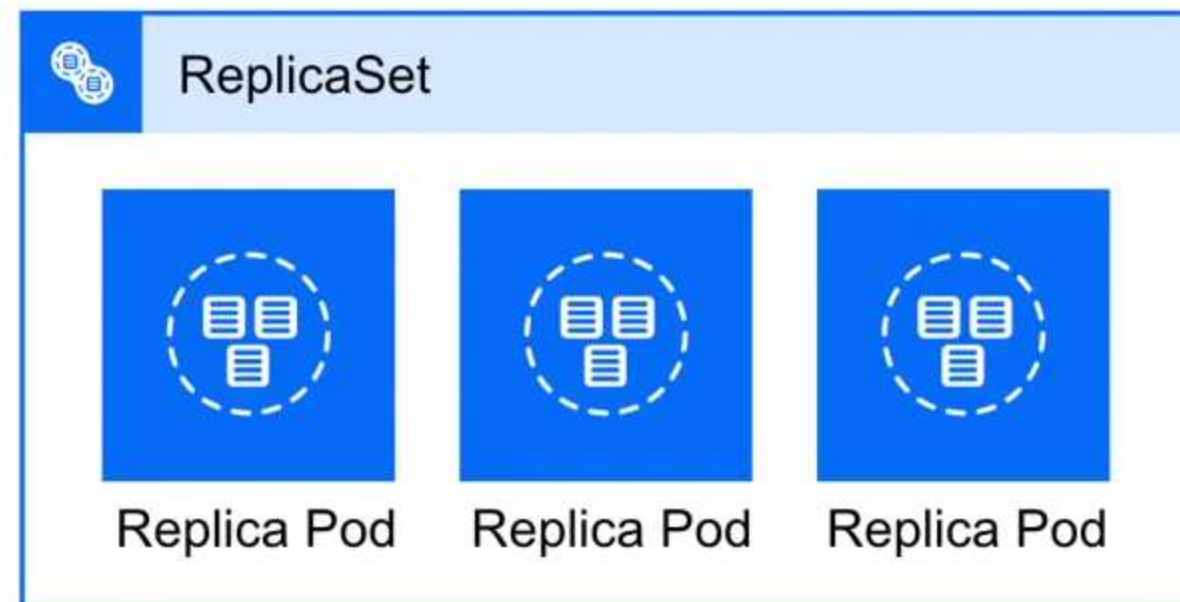
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```


Replica Sets

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

ReplicaSet is a way to **maintain a desired amount** of redundant pods (replicas) to provide a guarantee of availability.

The pod field **metadata.ownerReferences** determines
The link from a pod to a ReplicaSet.



```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: frontend
  labels:
    app: guestbook
    tier: frontend
spec:
  replicas: 3
  selector:
    matchLabels:
      tier: frontend
  template:
    metadata:
      labels:
        tier: frontend
    spec:
      containers:
        - name: php-redis
          image: gcr.io/google_samples/gb-frontend:v3
```



It is not recommended to directly create ReplicaSets
Instead a Deployment can create and manage a ReplicaSet for you.

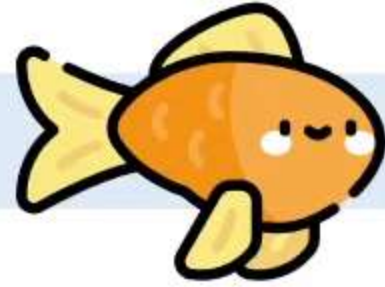


Horizontal Pod Autoscaler (HPA) can be used to autoscale a ReplicaSet

Stateless vs Stateful

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

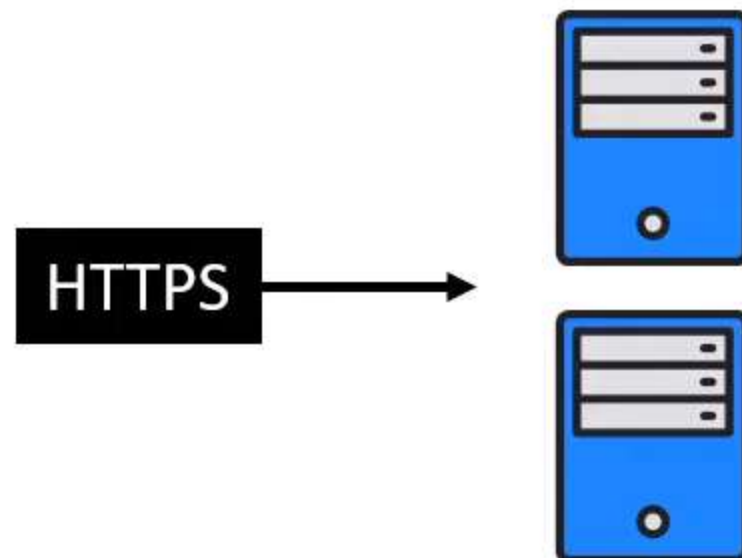
Stateless



ReplicaSets

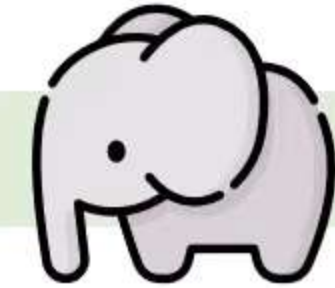
Every request does not care (forgets) about previous or current state

Web-apps that store sessions outside of the web-apps virtual machine (database or cookies) will be stateless web-apps



Send Request to any server, it doesn't matter

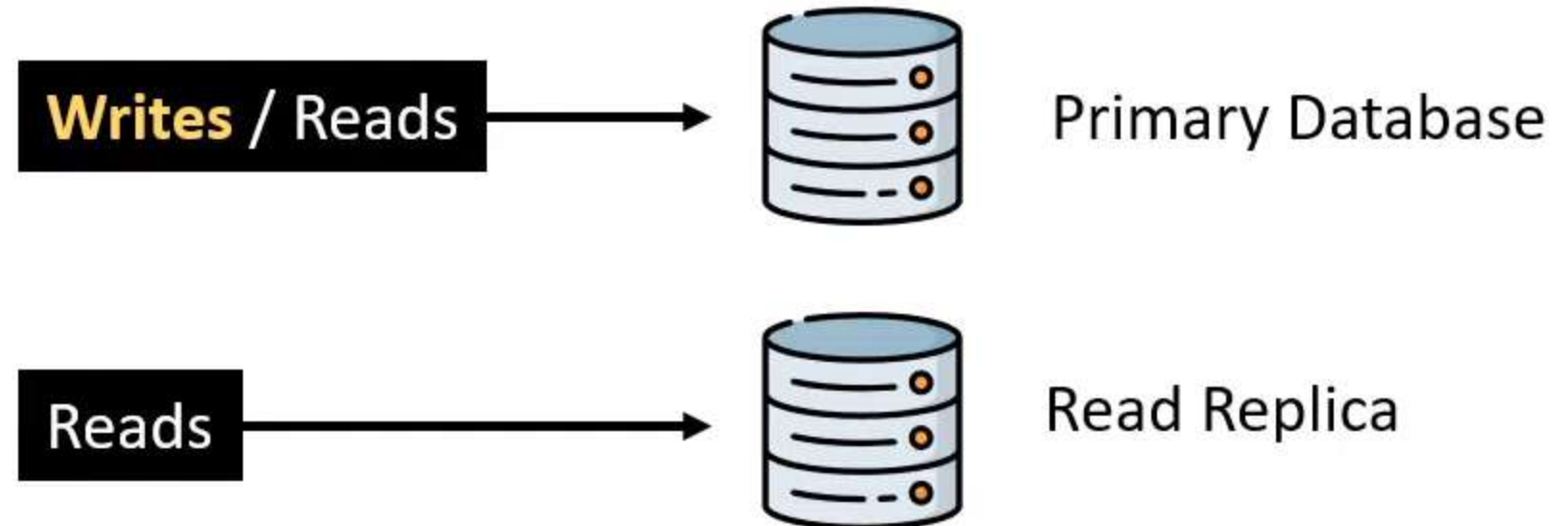
Stateful



StatefulSets

Every request relies on state data (remembers)
All databases are stateful

Monolithic web-apps that store session data in memory on a Virtual Machine are Stateful



Remember to send writes only to primary

Stateful Sets

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



Stateful Sets are used when you need traffic to be sent to a specific pods.

Stateful Set will always have:

- a unique and predictable name and address
- ordinal index number assigned each pod
- a persistent volume attached, with a persistent link from pod to the storage
 - If a pod is rescheduled the original Persistent Volume (PV) will be mounted to ensure data integrity and consistency.
- Stateful Set pods will always start in the same order, and terminate in reverse order



StatefulSets currently require a “headless” service to manage the identities

There is a headless service used to maintain the network identity of the pods, and another service that provides read access to the pods

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: web
spec:
  selector:
    matchLabels:
      app: nginx # has to match .spec.template.metadata.labels
  serviceName: "nginx"
  replicas: 3 # by default is 1
  minReadySeconds: 10 # by default is 0
  template:
    metadata:
      labels:
        app: nginx # has to match .spec.selector.matchLabels
    spec:
      terminationGracePeriodSeconds: 10
      containers:
        - name: nginx
          image: k8s.gcr.io/nginx-slim:0.8
          ports:
            - containerPort: 80
              name: web
          volumeMounts:
            - name: www
              mountPath: /usr/share/nginx/html
  volumeClaimTemplates:
    - metadata:
        name: www
      spec:
        accessModes: [ "ReadWriteOnce" ]
        storageClassName: "my-storage-class"
        resources:
          requests:
            storage: 1Gi
```


Stateful Sets

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

DNS Hostname for Writes

Writes will be directed to the Main pod by its DNS Hostname which is identified by a Headless Service

ClusterIP for Reads

For read traffic it can be distributed to all reading pods using a ClusterIP Service

Headless Service

The headless service is a Service with ClusterIP set to none. It does not provide load balancing

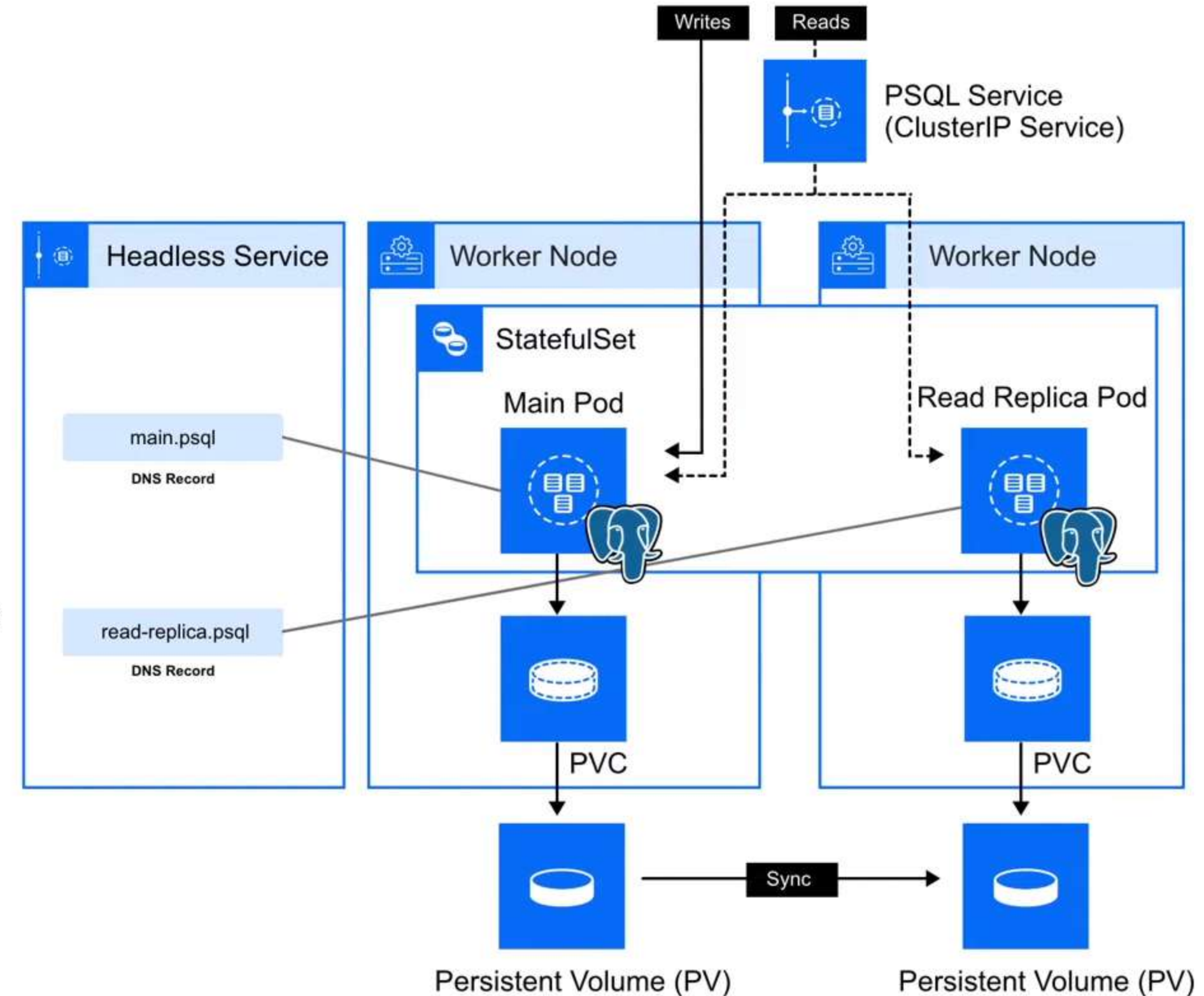
It does not provide a static IP address

A Headless Service is used to identify specific pods by assigning them DNS record.

A Headless Service is required in order for a StatefulSet to work.

PVC and PV

Each pod has its own volume. A Persistent Volume Claim can dynamically reference a Persistent Volume.



Namespaces

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



A Namespace is a way to logically isolate resources within a Kubernetes Cluster. Namespaces can be organized based on project, department or any user-defined grouping.

Kubernetes starts 4 initial namespaces

To **view all** namespaces

```
kubectl get namespace
```

1. default

- The default namespace where all our pods and services run unless a namespace is specified.

2. kube-public

- For resources that are publicly visible and readable.

3. kube-system

- The system namespace that stores objects created by the Kubernetes System.
- Engineers deploying applications are not supposed to touch this namespace.

4. kube-node-lease

- Holds lease objects associated with each node. Used to detect node failures by sending heartbeats

You can **create** your own namespaces

```
kubectl create namespace production
```



In clusters with a small amount of resources, Namespaces aren't necessary.



Namespaces

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



Names of resources need to be unique within a namespace, but not across namespaces.
Namespace-based scoping is applicable only for namespaced objects eg. *Deployments, Services*
But not for cluster-wide objects eg. *StorageClass, Nodes, PersistentVolumes*

Single-Namespace Objects

ConfigMaps and **Secrets** can't be shared across namespaces



Multi-Namespace Objects

A **Service** and **Pods** can belong to multiple namespaces



Cluster-Wide Objects

Volumes and **Nodes** cannot be bound to namespace since they are global



You can apply **system quota restrictions** on a namespace to avoid overuse eg Mem, Compute

If you don't provide a namespace for a component it will end up in the default namespace

In-Tree vs Out-of-Tree

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna

In Cloud-Native Projects you'll hear the term "In-Tree" and "Out-of-Tree"

In-Tree:

Plugins, components or functionality that are provided by default and/or reside in the main repository

Think of In-Tree as **Internal plugins**

Out-of-Tree:

Plugins, components or functionality that must be installed manually, and extends or replaces the default behaviour.

Think of Out-of-Tree as **External plugins**



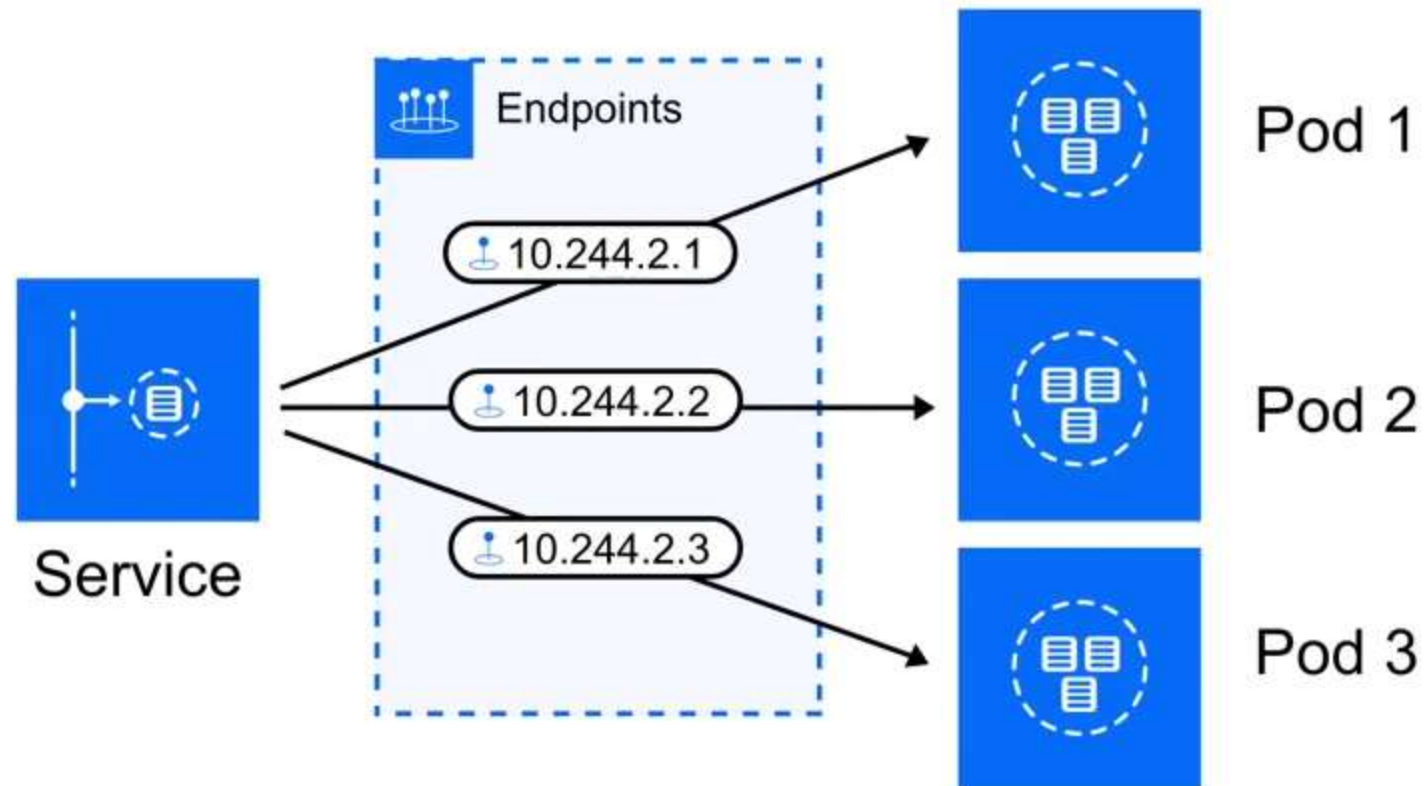
This is Andrew's own definition of In-Tree vs Out-of-Tree as the term does not appeared to be standardized.

Endpoints and Endpoint Slices

Cheat sheets, Practice Exams and Flash cards 🖱️ www.examprompro.co/kcna



Endpoints track the IP Addresses of the pods assigned to a Kubernetes Service. When a service selector matches a pod label, The pod IP Address is added to the pool of endpoints. Pods expose themselves to services via endpoints.



```
kubectl get endpoints
```

→ To see a list of endpoints



Endpoint Slices break up Endpoints into smaller manageable segments. Each Endpoint Slice has a limit of 100 pods.

Jobs and CronJobs

Cheat sheets, Practice Exams and Flash cards  www.exampopro.co/kcna

What is a Background Job?

A background job is a one-off task that is used to run a piece of code.

Commonly used to perform maintenance or to trigger a computational workload.

Eg. Back up the database ever x minutes, Delete all users who have not confirmed their email



A Job creates one or more Pods and will continue to retry execution of the Pods until a specified number of them successfully terminate.

```
kubectl create job hello --image=busybox -- echo "Hello World"
```



A CronJob is a job that execute based on a repeating schedule.

```
kubectl create cronjob hello --image=busybox --schedule="*/1 * * * *" -- echo "Hello World"
```


Kubernetes Dashboard

Cheat sheets, Practice Exams and Flash cards 🖱️ www.exampopro.co/kcna

Kubernetes Dashboard is an open-source application you can deploy to your cluster to provide a UI to view K8s components

