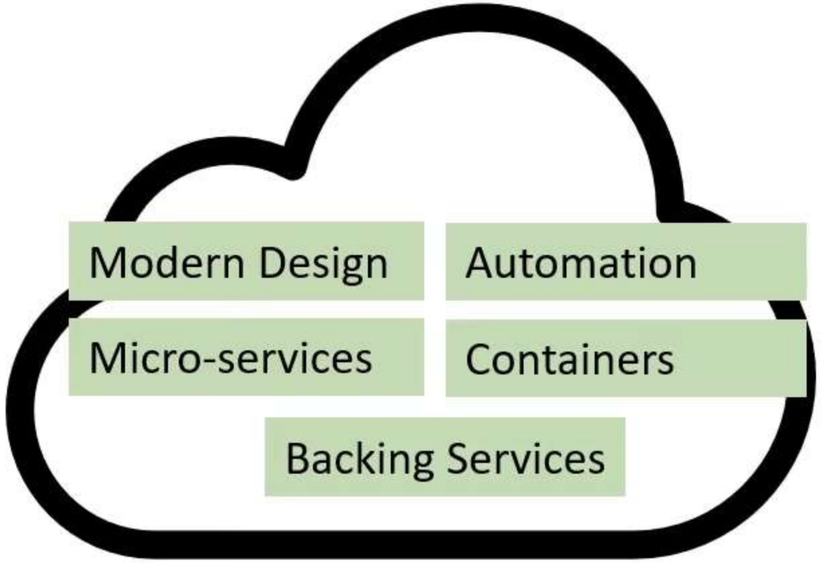
What is a Cloud-Native?

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.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.exampro.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
COMPUTING FOUNDATION

Cloud Native vs Cloud Service Providers

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

A Cloud Service Provider (CSP):

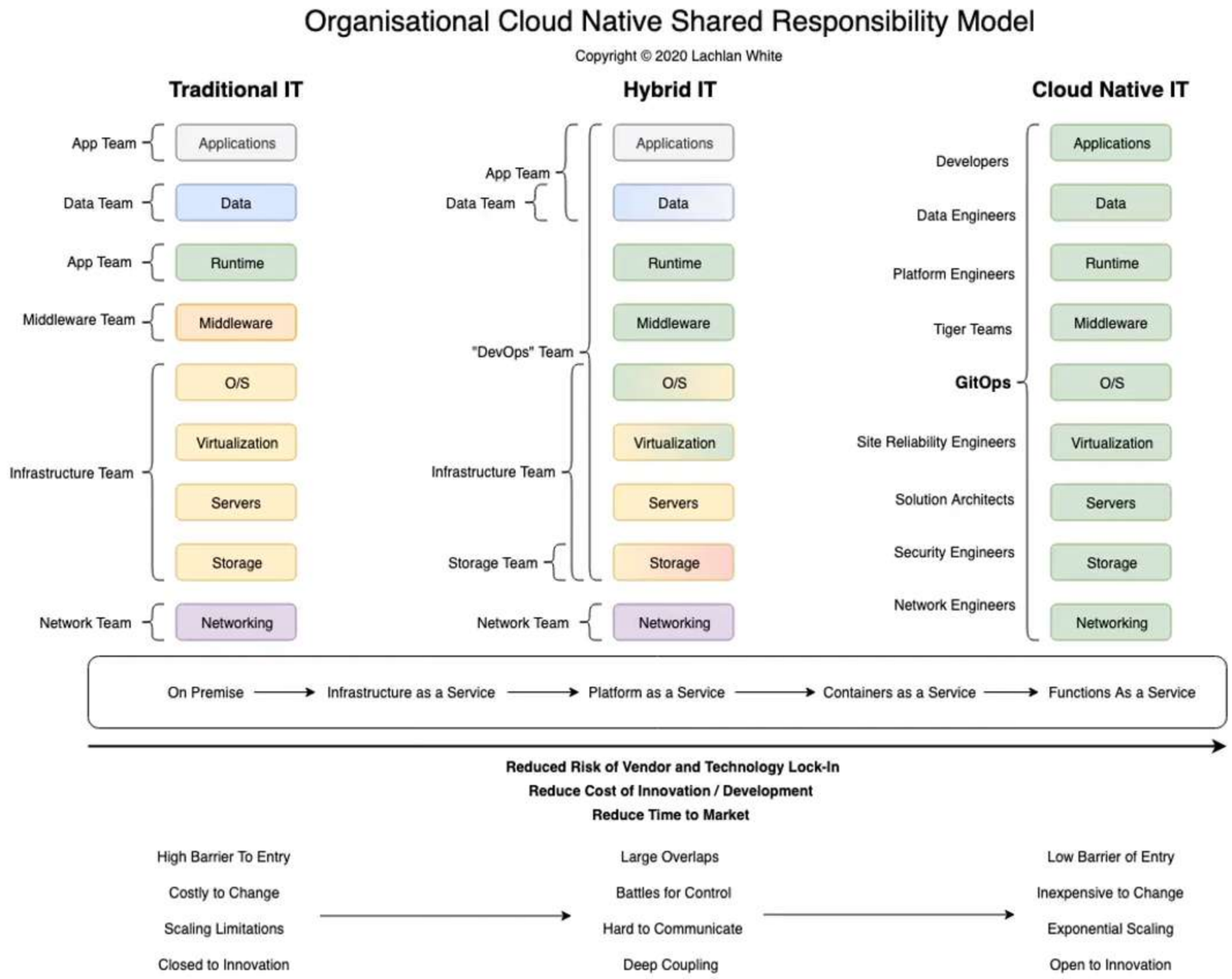
- A collection of cloud services
- Strong application integration and synergies between services
- Utilizing metered billing
- Under a single unified API

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.exampro.co/kcna



Lachlan White Cloud-Native Model

@ LachieWhite7

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:

AT&T
SAMSUNG
CISCO
FUJITSU
Google
Tencent
orange
ORACLE
HUAWEI
Qualcomm
HITACHI
Microsoft
IBM
NEC
vmware
intel

Cloud Native Computing Foundation

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

CLOUD NATIVE COMPUTING FOUNDATION

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.exampro.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

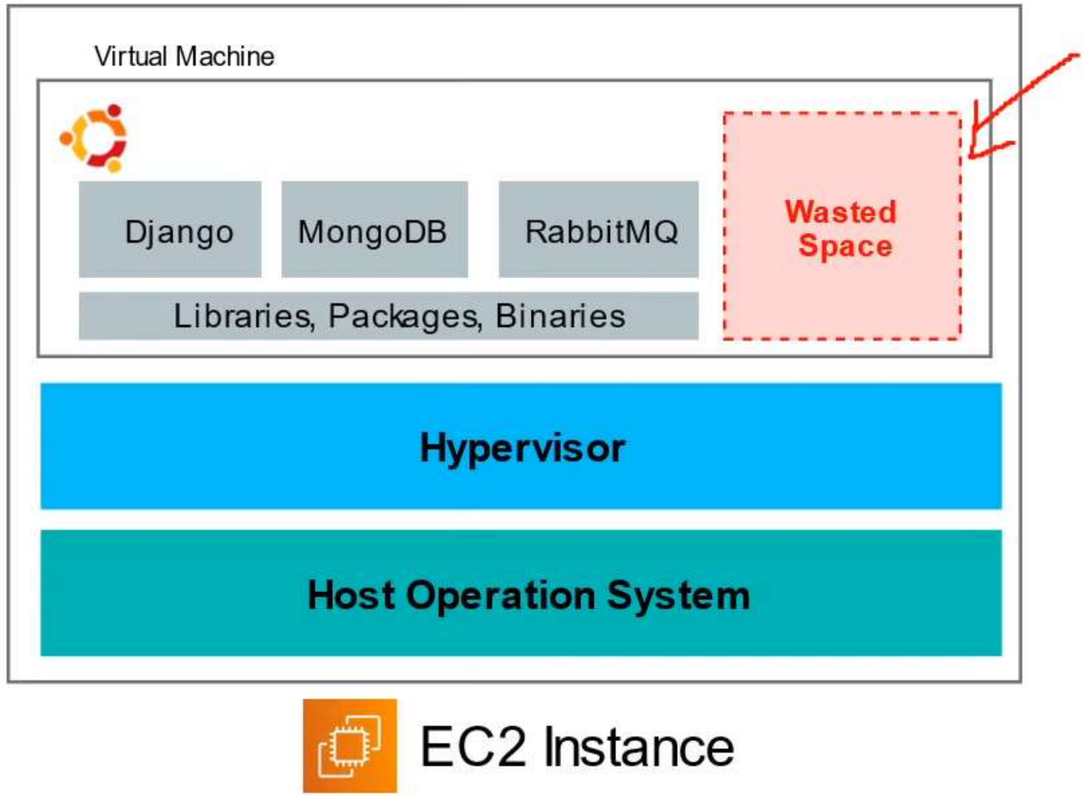
A. Training and Certification
B. Consulting Help
C. Join CNCF End User Community

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

VMs vs Containers

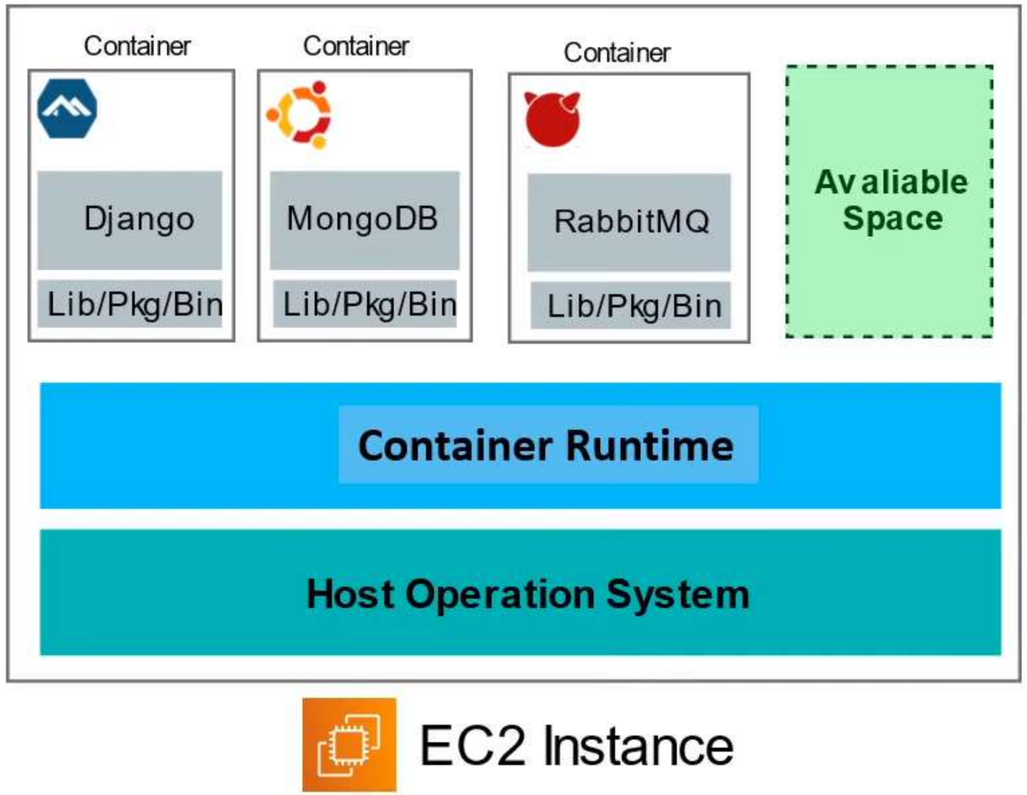
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

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.



What are Microservices

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

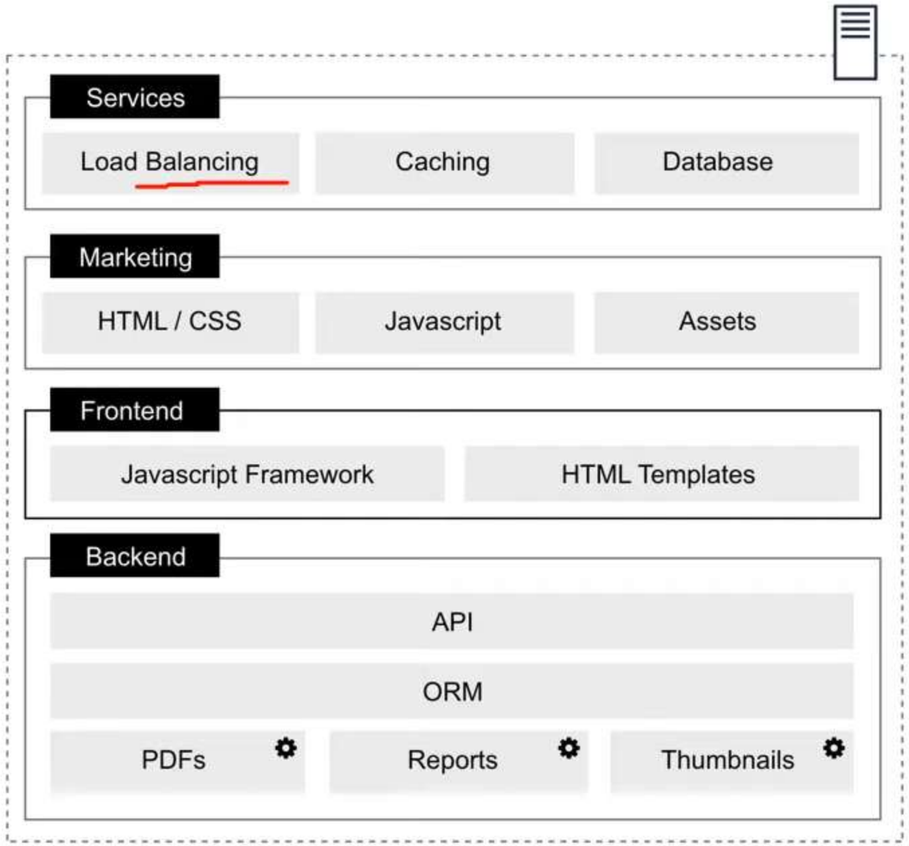
Monolithic Architecture

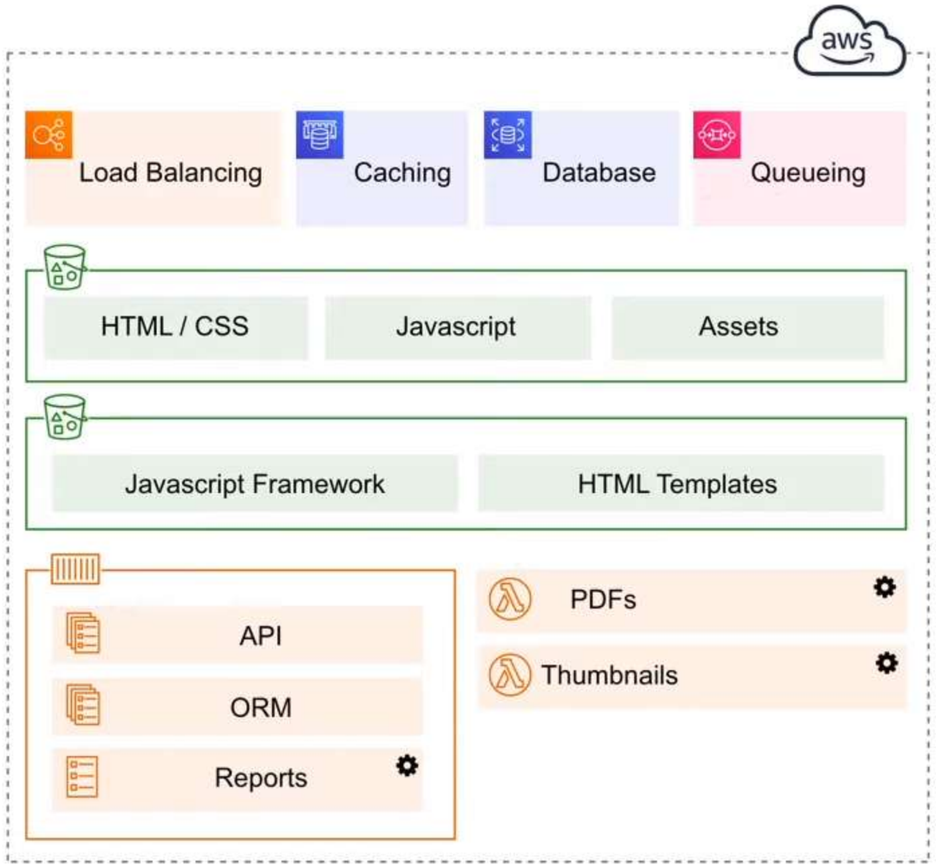
One app which is responsible for everything
Functionality is tightly coupled

VS

Microservices Architecture

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





Kubernetes

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

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

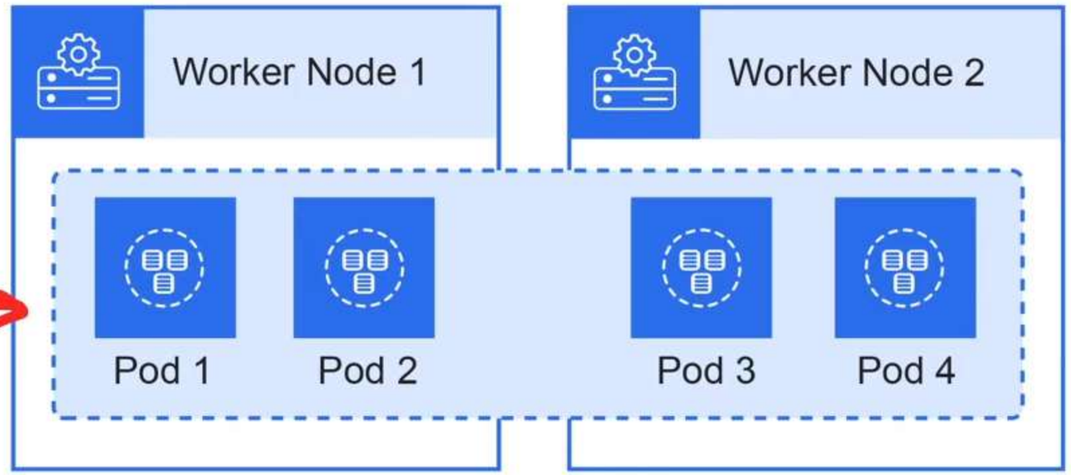
Originally created by Google and now maintained by the **Cloud Native Computing Foundation (CNCF) as a CNCF Project**

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.exampro.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.exampro.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. Kublelet 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 a 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.exampro.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.exampro.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

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

YAML

```json
{
  "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.exampro.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.

```yaml
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
```

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.

Nodes – Control Plane and Worker Nodes

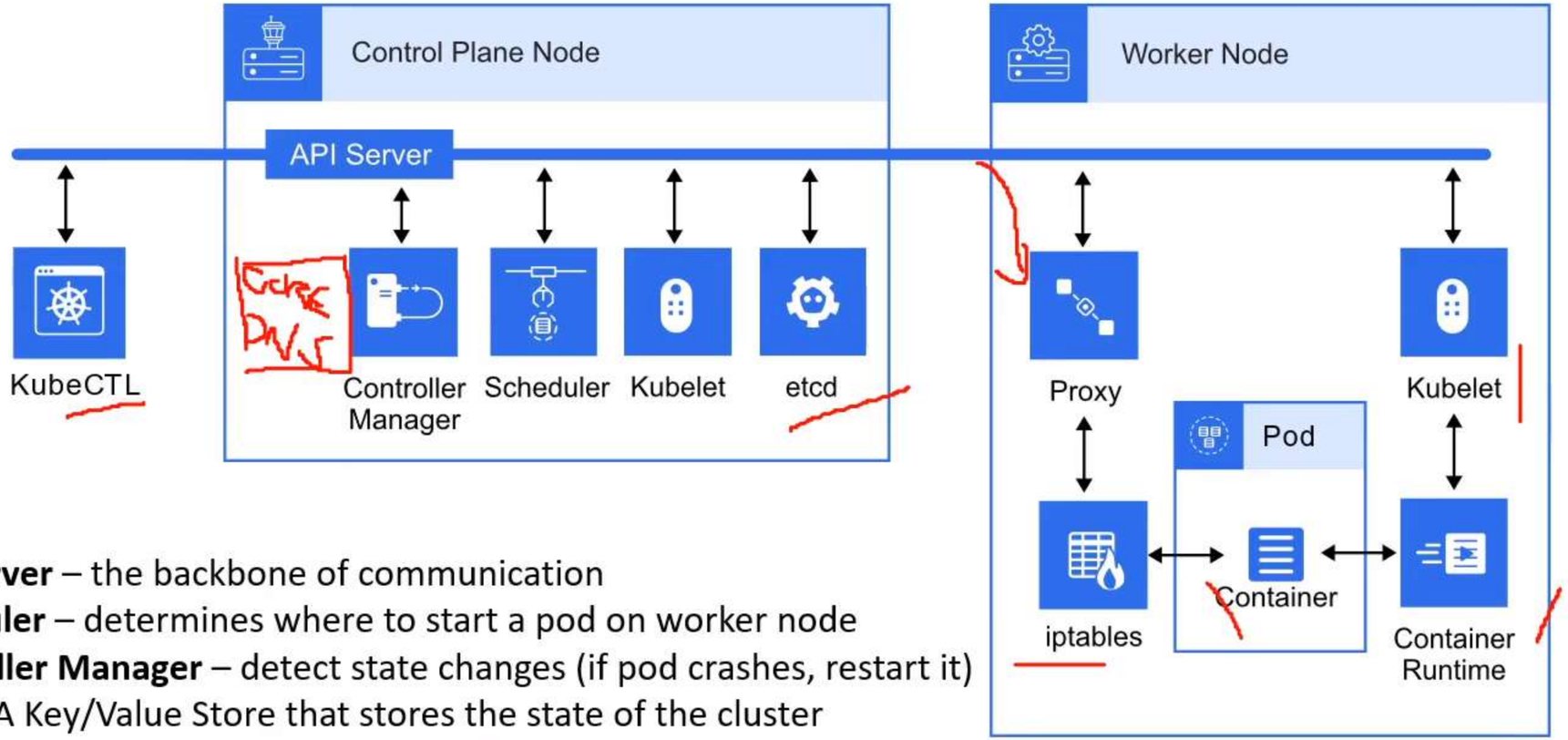
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

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

Manages processes like scheduling, restarting nodes…

Worker Node

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



The worker node runs:

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

- **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

Pods

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.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 it own private IP address

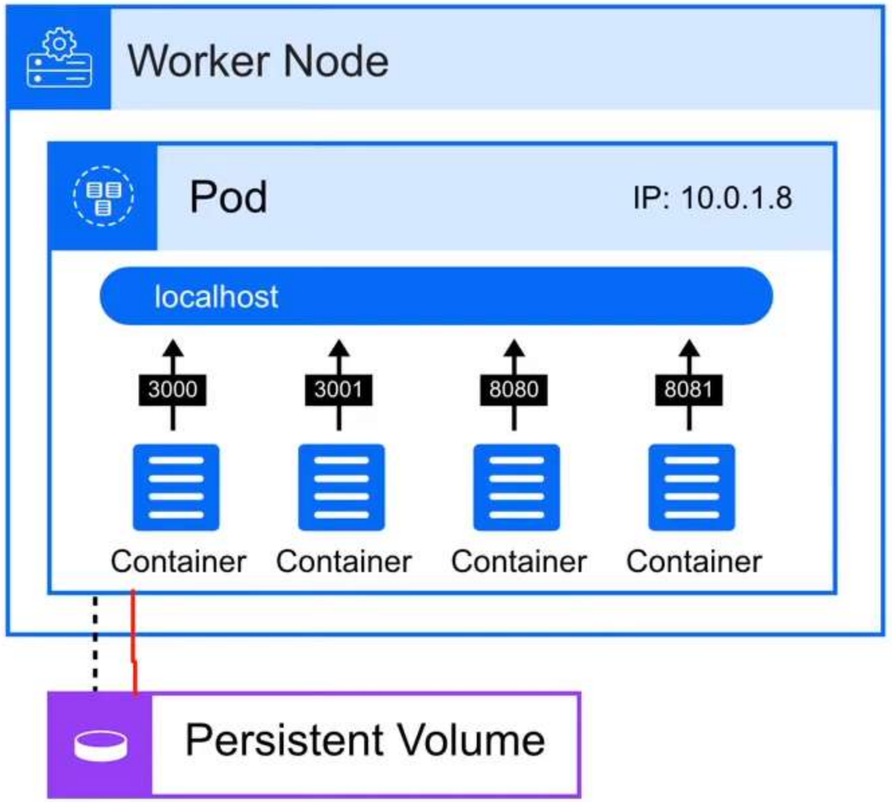
- Containers will run on different ports
- Containers can talk to each other via localhost

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.exampro.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

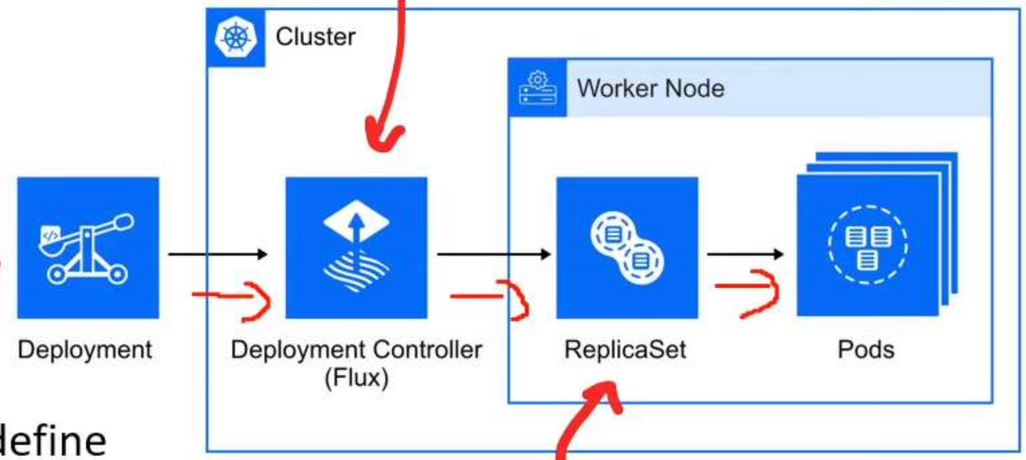
Deployment

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.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.

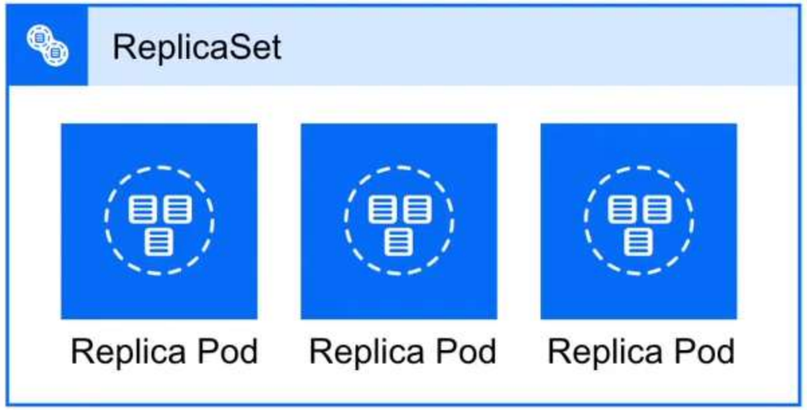
```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.14.2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

Replica Sets

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.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.




```yaml
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
```

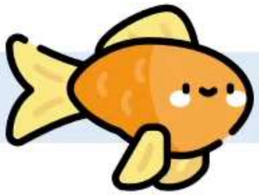
It is not recommended to directly create ReplicaSets
Instead a Deployment can create and manage a ReplicaSet for you.

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

Stateless vs Stateful

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

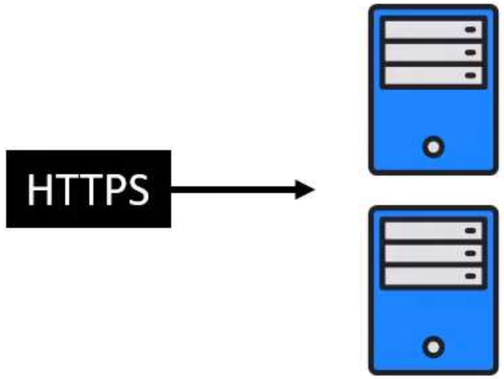
Stateless



ReplicaSets

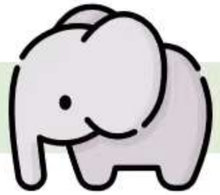
Every request does not care (forgets) about previous or current state

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

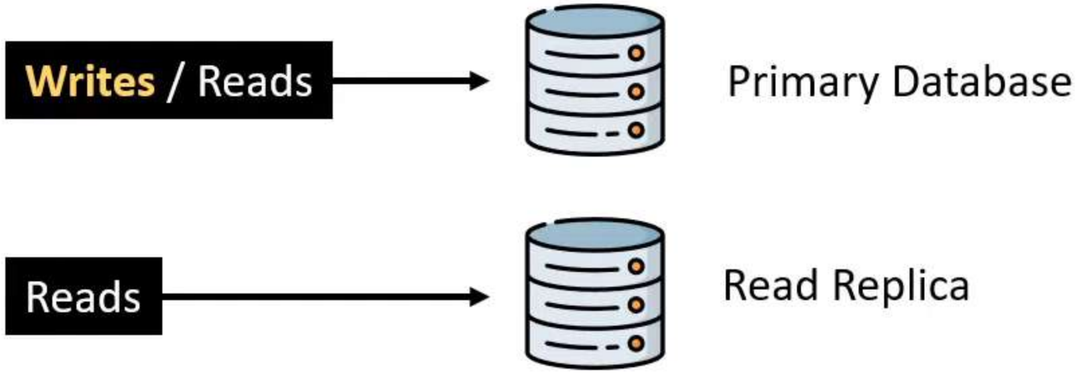
Stateful



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.exampro.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

```yaml
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

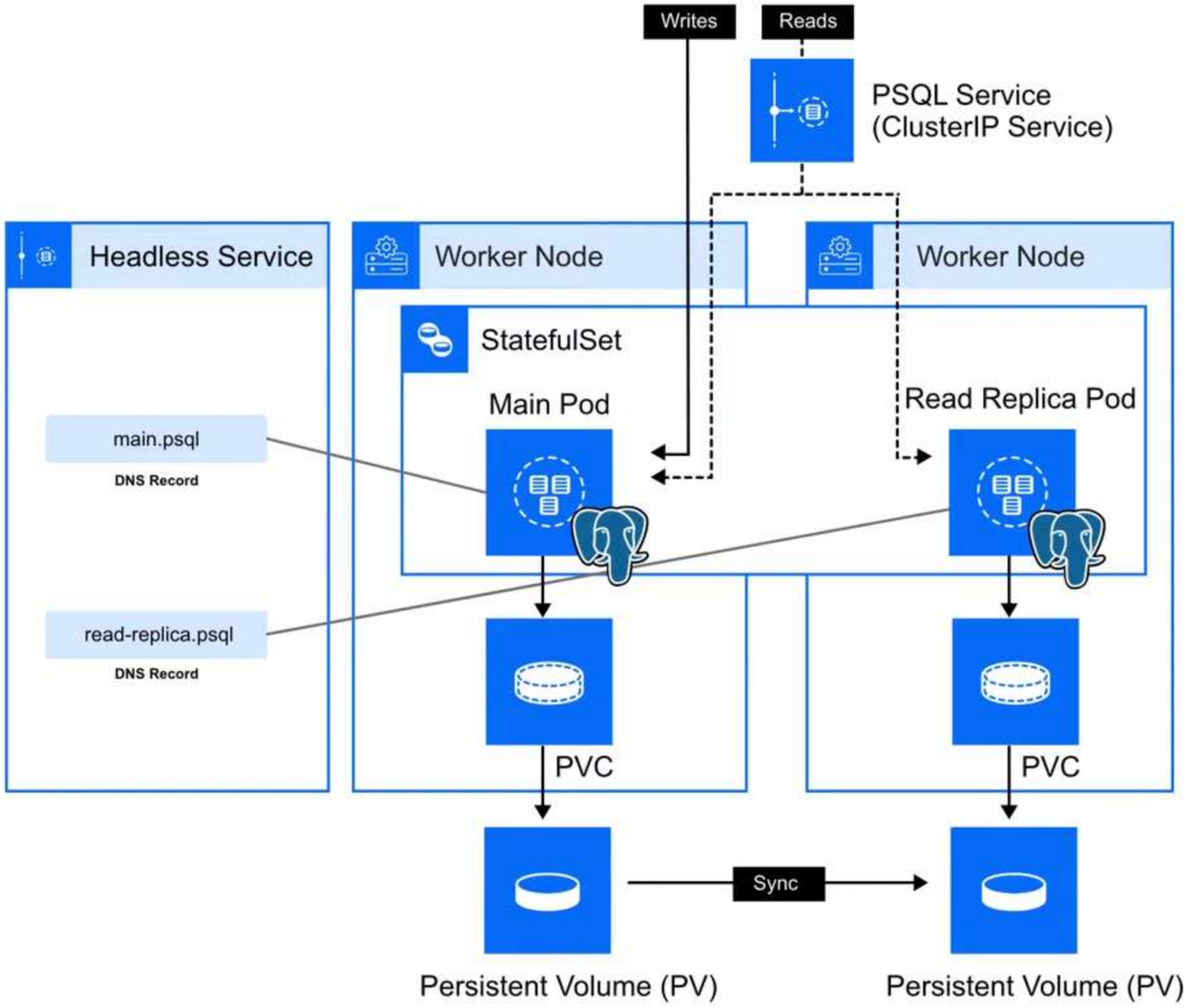
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

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 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.exampro.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.exampro.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

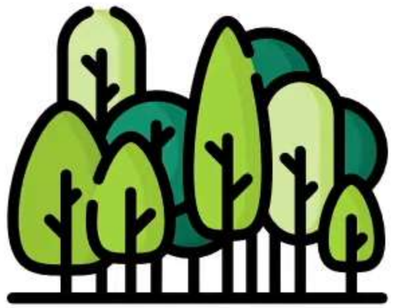
In-Tree vs Out-of-Tree

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.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***

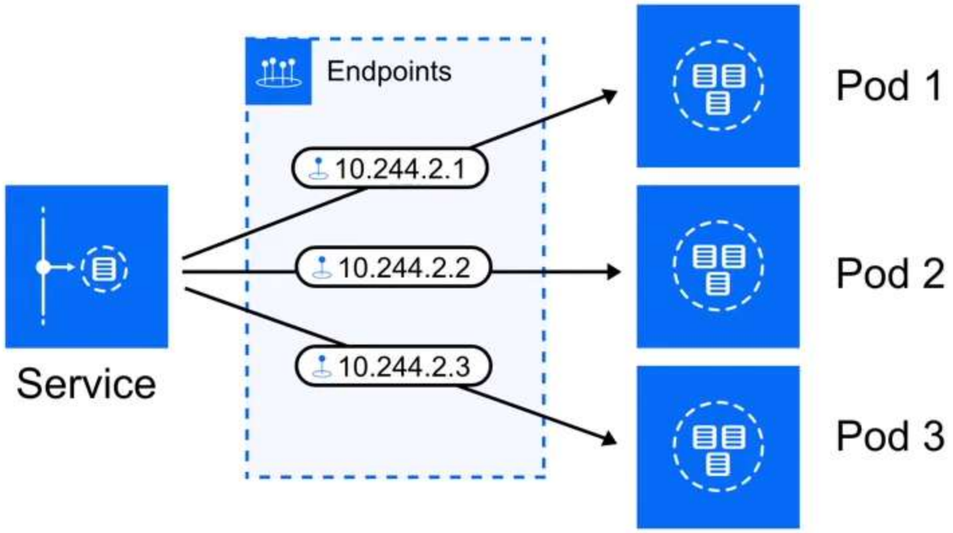


This is Andrew's own definition of In-Tree vs Out-of-Tree as the term does not appeared to be standardized.

Endpoints and Endpoint Slices

Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.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.exampro.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"
```

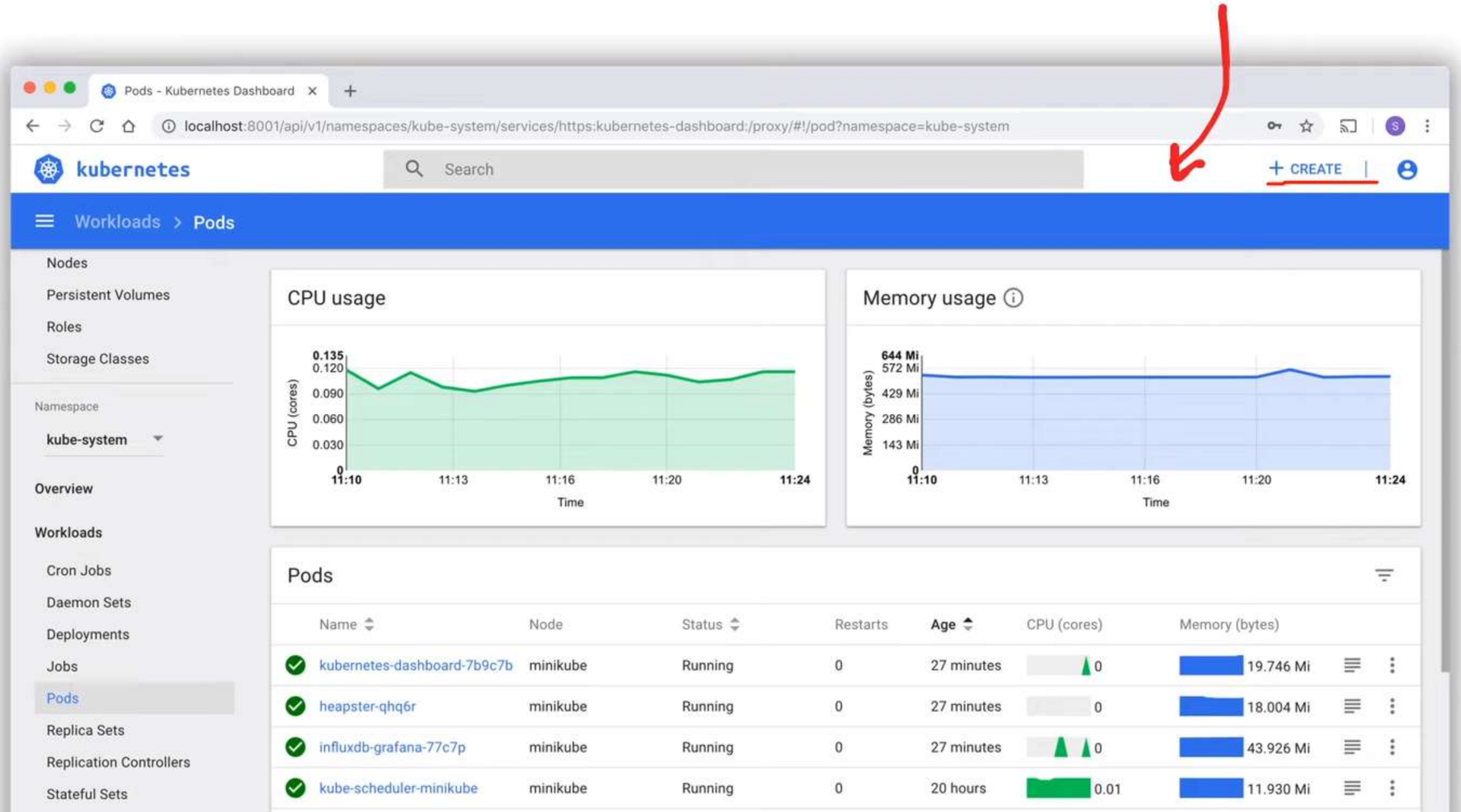
A CronJob is a job that execute based on a repeating schedule.

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

Kubernetes Dashboard

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

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

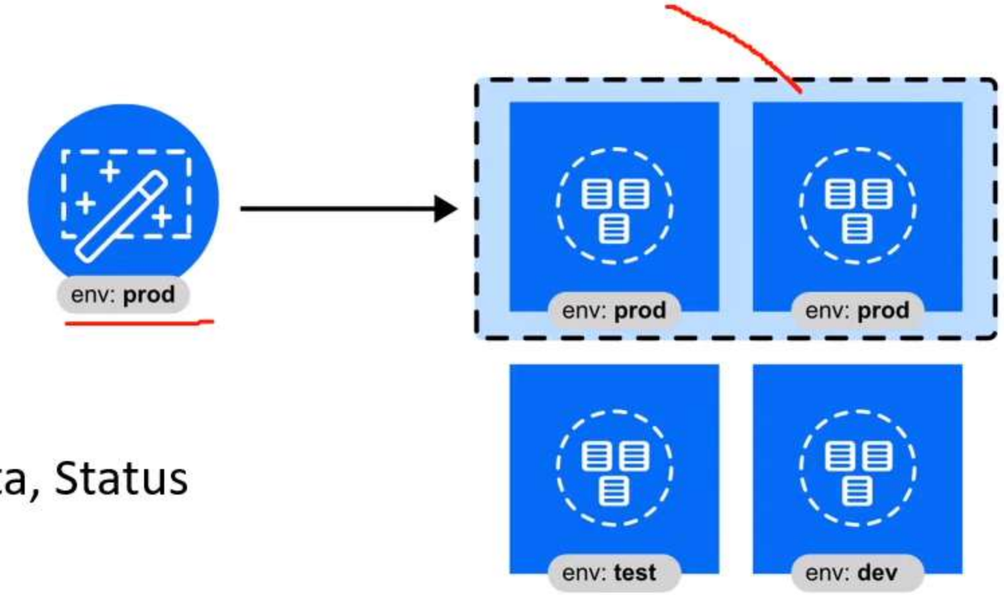


Selectors

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

Selectors are a way of selecting one or more Kubernetes Objects.

In Kubernetes there are **3 types** of selectors:



1. Label Selector
Select K8s objects based on the applied label

2. Field Selector
Select K8s objects based on object data eg. Metadata, Status

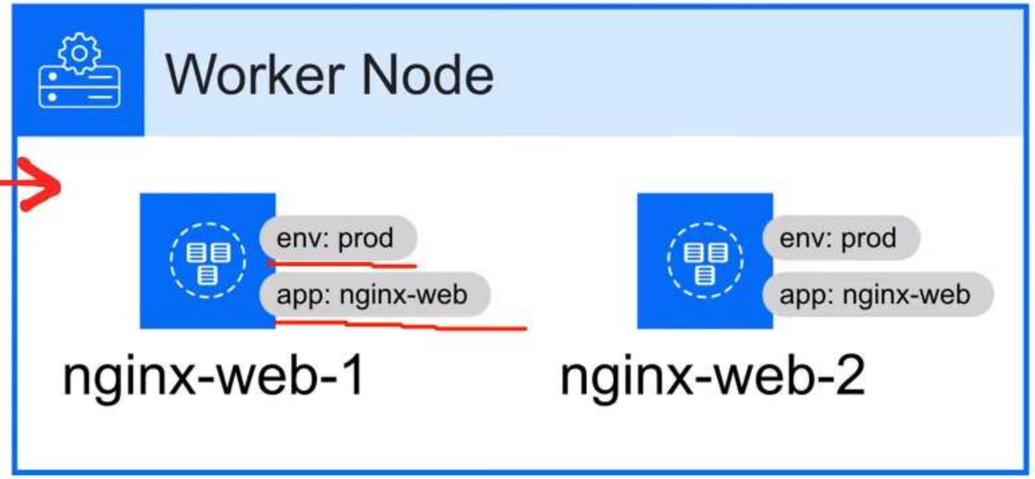
3. Node Selector
Select nodes for very specific pod placement

Label Selector – Applying Labels

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

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-web-1
  labels:
    env: prod
    app: nginx-web
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
---
apiVersion: v1
kind: Pod
metadata:
  name: nginx-web-2
  labels:
    env: prod
    app: nginx-web
spec:
  containers:
    - name: nginx
      image: nginx
      ports:
        - containerPort: 80
```

Label Selectors define labels as a **key/value pair** under metadata in a Manifest file




```
kubectl get pods --show-labels
```

You can use the **–show-labels** flag to see labels.

```
kubectl label pods apache-web owner=devops
```

You can apply labels with the **label** command

Recommended Labels

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

These are recommended labels that "should" **apply to every resource object**:

app.kubernetes.io/name
The name of the application

app.kubernetes.io/instance
A unique name identifying the instance of an application

app.kubernetes.io/version
The current version of the application (e.g., a semantic version, revision hash, etc.)

app.kubernetes.io/component
The component within the architecture

app.kubernetes.io/part-of
The name of a higher level application this one is part of

app.kubernetes.io/managed-by
The tool being used to manage the operation of an application

app.kubernetes.io/created-by
The controller/user who created this resource

Shared labels and annotations share a common prefix: app.kubernetes.io. Labels without a prefix are private to users.

```yaml
# This is an excerpt
apiVersion: apps/v1
kind: StatefulSet
metadata:
  labels:
    app.kubernetes.io/name: mysql
    app.kubernetes.io/instance: mysql-abcxzy
    app.kubernetes.io/version: "5.7.21"
    app.kubernetes.io/component: database
    app.kubernetes.io/part-of: wordpress
    app.kubernetes.io/managed-by: helm
    app.kubernetes.io/created-by: controller-manager
```

Label Selector – Selecting Labels

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

K8 objects like **Services** and **ReplicaSets** will target pods based on label selectors

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
```

```yaml
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
```

Notice the selector syntax varies for different templates

You can use selector in the KubeCTL command line with **--selector** or it alias **-l**

```
kubectl get pods --selector env=development
kubectl get pods -l env=development
```

Annotations

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

Kubernetes annotations allows you to attach **arbitrary non-identifying metadata to objects**. Clients (eg. tools and libraries) can and my require this annotation in order to work.

Ingress often use annotation to communicate to Ingress Controllers.

To use the Nginx Ingress Controller you need To specifc a rewrite-target path.

```yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: minimal-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
  - http:
      paths:
      - path: /testpath
        pathType: Prefix
        backend:
          service:
            name: test
            port:
              number: 80
```

PodSpec File

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

PodSpec is a configuration file that describes a pod.

Notice kind is **pod**.

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

The **spec** section will define

- multiple containers
  - The name of the container
  - The image
  - The command to run on startup
  - The port the container will operate on
  - The restart policy
  - …

You can directly deploy a pod with the **kubectl apply** command

```
kubectl apply -f ngnix.yaml
```

In practice you wont directly deploy pods, instead you'd use Deployment or Job as the **kind**

gRPC

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

What is Remote Procedure Call (RPC)?

RPC is a framework of communication in distributed systems. It allows one program on machine to communication with another program on remote machine without knowing its remote. The concept of RPC has been around since 1970s.

gRPC is a modern open source high performance Remote Procedure Call (RPC) framework that can run in any environment. gRPC was initially created by Google.

You can think of gRPC as an **alternate method of communication** instead of REST or GraphQL

gRPC can connect services in and across data centers with pluggable support for:

- load balancing
- tracing
- health checking
- authentication

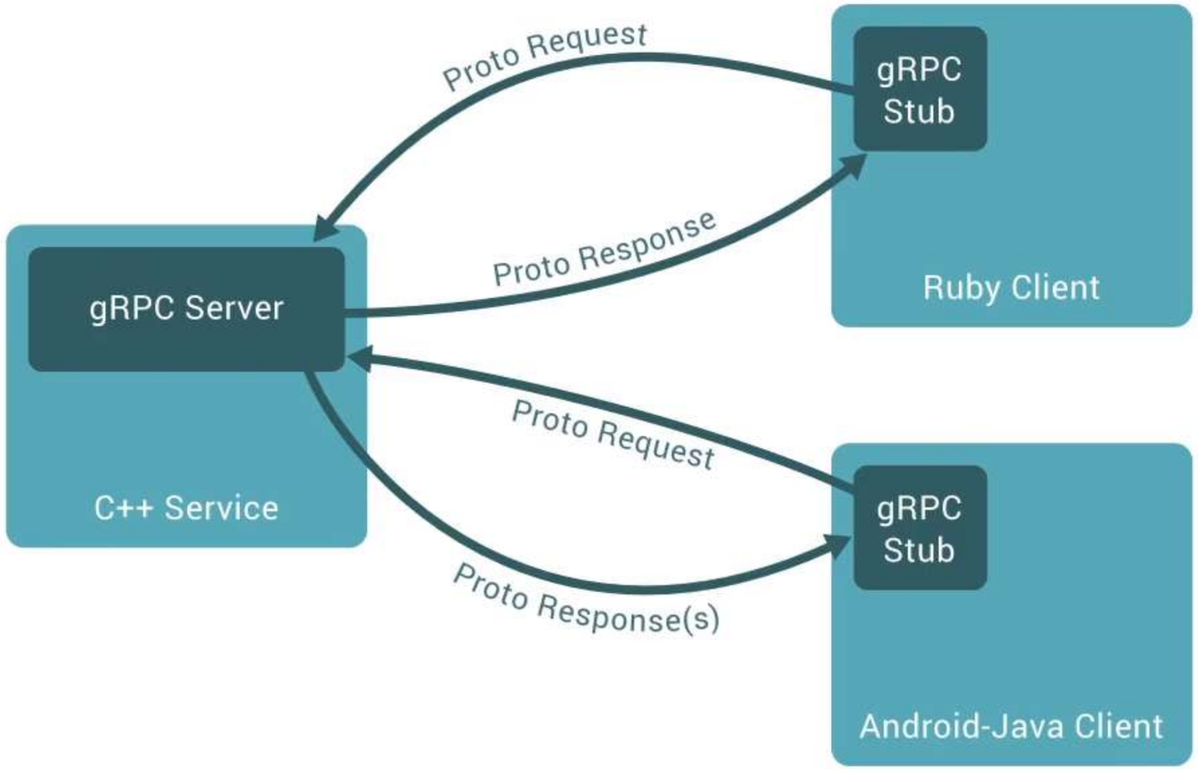
gRPC

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

gRPC works by:

- Installing a gRPC library for your program.
- Defining a Protobuf file that describe the data structure.
  - Profobuf files have **.proto** exentsion
- Writing code that works with gRPC

Distributed systems like Kubernetes uses gRPC for pod communication.



Kubelet

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

Kubelet is responsible for pod internal API communication via the API Server.

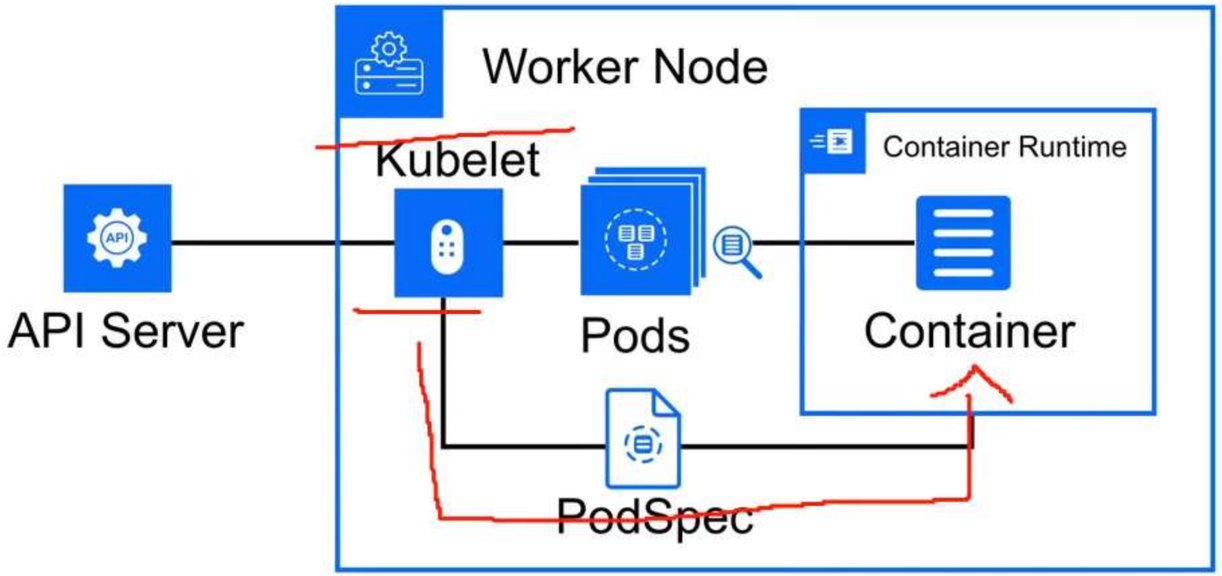
Kubelet is a **node agent** that runs on every single node (control plane and workers).

What is an agent?
An agent is a program installed on a server to observe what occurs for specific programs, and to communicate information externally or trigger actions to be performed.

The Kubelet perform the following tasks:

- watches for pod changes
- Configures the Container Runtime to
  - Pull images
  - Create namespaces
  - Run containers

Kubelet uses **PodSpec files** to determine what images to pull and containers to run.



Kubelet

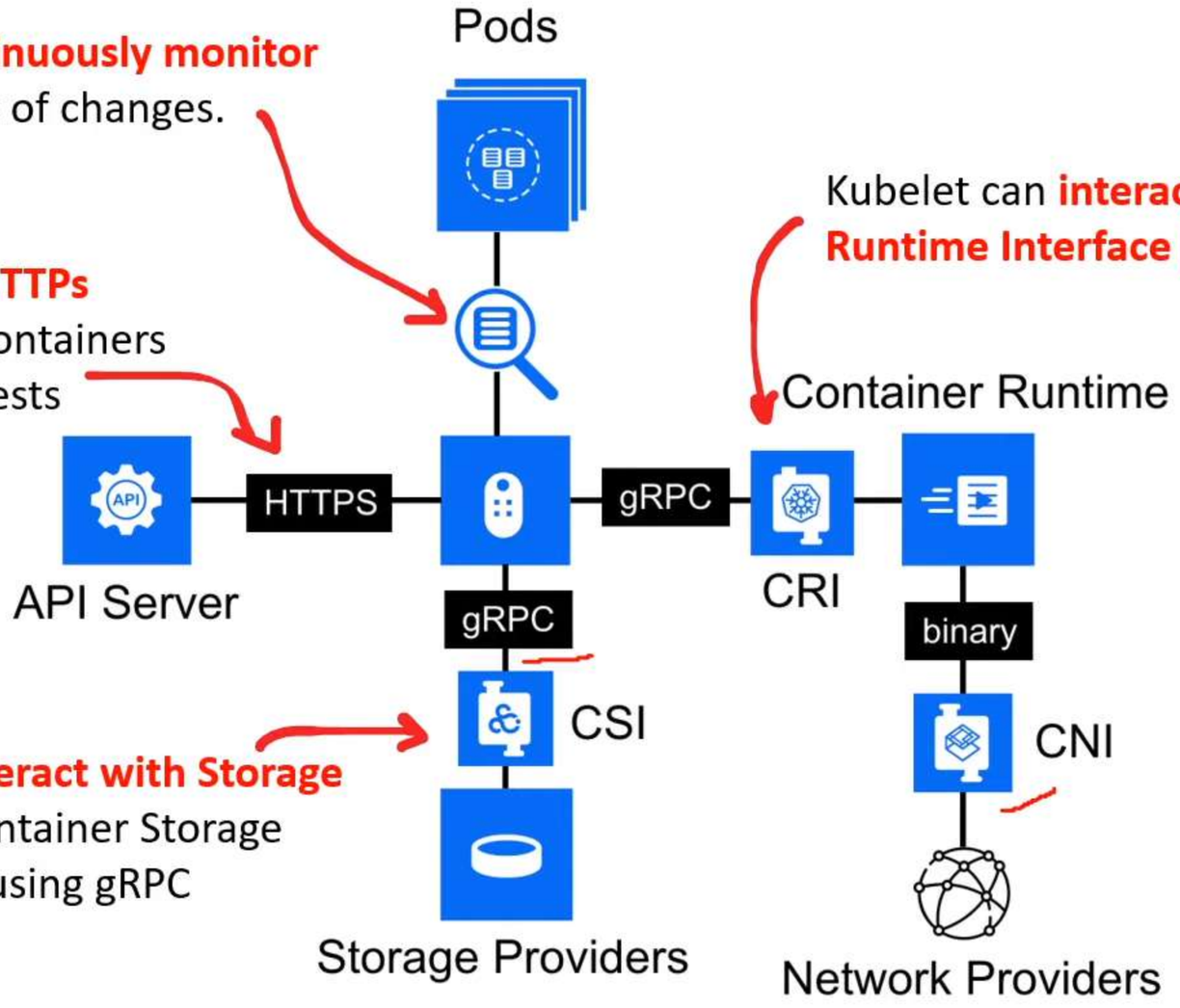
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

Kubelet will **continuously monitor** Pods of any kinds of changes.

Kubelet will **send back HTTPs requests to API server** containers logs and execution requests

Kubelet can **interact with a Container Runtime Interface (CRI)** also using gPRC

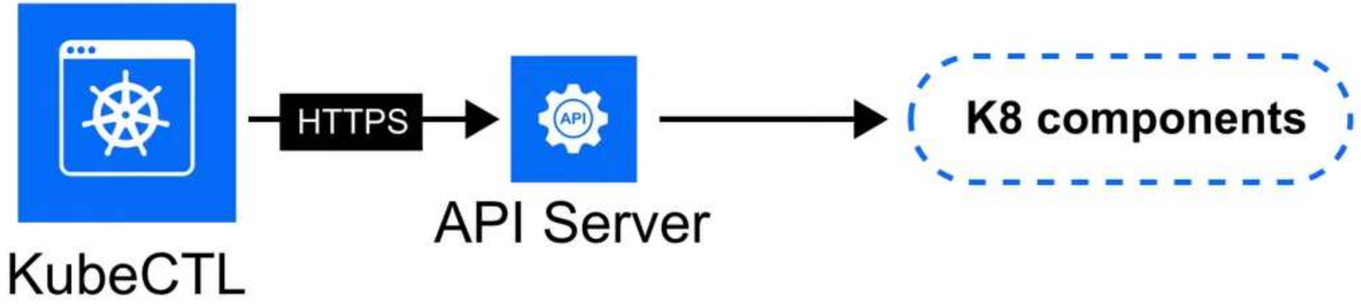
Kubelet can **interact with Storage** through the Container Storage Interface (CSI) using gRPC



KubeCTL

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

KubeCTL is a command line tool lets you control Kubernetes clusters



kubectl looks for a file named config in the $HOME/.kube directory

KubeCTL has the following syntax:

- kubectl **[COMMAND] [TYPE] [NAME] [FLAGS]**

KubeCTL – Command

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

kubectl **[COMMAND]** **[TYPE] [NAME] [FLAGS]**

COMMAND: The **operation** you want to perform

```
kubectl cp simple.txt nginx-6db489d4b7-qkd5d:.
```

Available commands:

annotation - key value data that can be applied to resources

apply - Executes manifest files to create, modify K8 resources

auth - Inspect if you are authorized to perform an action

autoscale - Create an autoscaler that automatically chooses and sets the number of pods that run in a kubernetes cluster

cp - Copy files and directories to and from containers

create — Create specific K8 cluster-level resources: ConfigMap, Deployment, Job, Namespace, Role, Secret

delete — Delete resources filenames, stdin, resources and names, or by resources and label select

describe — Show details of a specific resource or group of resources

diff — diffs the online configuration with local config

KubeCTL – Command

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

- **edit** — Edit a resource from the default editor. Edit a deployed manifest file and apply the changes.
- **exec** — Execute a command within a container
- **expose** — Expose as resource as Kubernetes service
- **get** — generally used to get the status of an existing Kubernetes resource
- **kustomize** — Print a set of API resources generated from instructions in a kustomization.yaml file
- **label** — update labels on a resource
- **logs** — print the logs for a container in a pod or specific resource
- **patch** — Update fields of a resource using strategic merge patch, a JSON merge patch, or a JSON patch
- **port-forward** — Forward one or more local ports to a pod.
- **proxy** — Creates a proxy server between localhost and the Kubernetes API Server
- **run** — Create and run a container image in a pod
- **scale** — Set a new size for a Deployment, ReplicaSet, Replication Controller, or StatefulSet

KubeCTL – Type

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

kubectl [COMMAND] **[TYPE]** [NAME] [FLAGS]

TYPE: The **resource type** you want to command

```
kubectl describe deployments
```

Resource types can have abbreviations:

- deployments → deploy
- persistentVolumes → pv
- pods → po

```
kubectl describe deploy
```

There are over **50+ resource types**

- Bindings
- Componentstatuses (cs)
- Configmaps (cm)
- Endpoints (ep)
- Events (ev)
- Limitranges (limits)
- Namespaces (ns)
- Nodes (no)
- Persistentvolumeclaims (pvc)
- Persistentvolumes (pv)
- Pods (po)
- Podtemplates
- Replicationcontrollers (rc)
- Resourcequotas (quota)
- Secrets
- Serviceaccounts (sa)
- Services (svc)
- Mutatingwebhookconfigurations
- Validatingwebhookconfigurations
- Customresourcedefinitions (crd)
- Apiservices
- Controllerrevisions
- Daemonsets (ds)
- Deployments (deploy)
- Replicasets (rs)
- Statefulsets (sts)
- Tokenreviews
- Localsubjectaccessreviews
- Selfsubjectaccessreviews
- Selfsubjectrulesreviews
- Subjectaccessreviews
- Horizontalpodautoscalers
- Cronjobs (cj)
- Jobs
- Certificatesigningrequests (csr)
- Leases
- Endpointslices
- Events (ev)
- Ingresses (ing)
- Flowschemas
- Prioritylevelconfigurations
- Ingressclasses
- Networkpolicies (netpol)
- Runtimeclasses
- Poddisruptionbudgets (pdb)
- Podsecuritypolicies (psp)
- Clusterrolebindings
- Clusterroles
- Rolebindings
- Roles
- Priorityclasses (pc)
- Csidrivers
- Csinodes
- Storageclasses (sc)
- volumeattachments

KubeCTL – Name

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

kubectl [COMMAND] [TYPE] [NAME] [FLAGS]

NAME: Specifies the **name of the resource**

```
kubectl get pod example-pod1 example-pod2
```

Names are case-sensitive

If the name is **omitted**, details for all resources are displayed

```
kubectl get pods
```

KubeCTL – Flags

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

kubectl [COMMAND] [TYPE] [NAME] **[FLAGS]**

FLAGS: Specifies optional flags

```
kubectl create role my-role --verb=get,list,watch --resource=rs.extensions
```

- Flags generally start with two hyphens. eg. –server
- Sometimes flags have abbreviations with single hypen eg. -s
- Available flags will vary based on command
- Sometimes flags can be assigned values or do not expect a value.

KubeCTL Documentation

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

KubeCTL documentation shows example of how to use all the KubeCTL commands.

https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands

Kubectl Reference Docs

kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#run

run
expose
delete

APP MANAGEMENT
apply
annotate
autoscale
debug
diff
edit
kustomize
label
patch
replace
rollout
scale
set
wait

WORKING WITH APPS
attach
auth
cp
describe
exec
logs

run

Create and run a particular image in a pod.

Usage

```
$ kubectl run NAME --image=image [--env="key=value"] [--port=port] [--dry-run=server|client] [--overrides=inline-json] [--command] -- [COMMAND] [args...]
```

Flags

Name	Shorthand	Default	Usage
allow-missing-template-keys		true	If true, ignore any errors in templates when a field or map key is missing in the template. Only applies to golang and jsonpath output formats.
annotations		[]	Annotations to apply to the pod.
attach		false	If true, wait for the Pod to start running, and then attach to the Pod as if 'kubectl attach ...' were called. Default false, unless '-i/--stdin' is set, in which case the default is true. With '--restart=Never' the exit code of the container process is returned.
cascade		backgroun d	Must be "background", "orphan", or "foreground". Selects the deletion cascading strategy for the dependents

example

Start a nginx pod

```
kubectl run nginx --image=nginx
```

Start a hazelcast pod and let the container expose port 5701

```
kubectl run hazelcast --image=hazelcast/hazelcast --port=5701
```

Start a hazelcast pod and set environment variables "DNS_DOMAIN=cluster" and "POD_NAMESPACE=default" in the container

```
kubectl run hazelcast --image=hazelcast/hazelcast --env="DNS_DOMA
```

Start a hazelcast pod and set labels "app=hazelcast" and "env=prod" in the container

```
kubectl run hazelcast --image=hazelcast/hazelcast --labels="app=h
```

Dry run; print the corresponding API objects without creating them

```
kubectl run nginx --image=nginx --dry-run=client
```

Start a nginx pod, but overload the spec with a partial set of values parsed from JSON

```
kubectl run nginx --image=nginx --overrides='{ "apiVersion": "v1",
```

Start a busybox pod and keep it in the foreground, don't restart it if it exits

```
kubectl run -i -t busybox --image=busybox --restart=Never
```

MiniKube

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

Minikube sets up a local single-node Kubernetes cluster on macOS, Linux, and Windows for learning purposes.

Minikube is running a Virtual Machine running a control-place and work processes with Docker as the container layer

- Supports the latest Kubernetes release
- Cross-platform
- Deploy as a VM, a container, or on bare-metal
- Multiple container runtimes
- Docker API endpoint for blazing fast image pushes
- Advanced features such as LoadBalancer, filesystem mounts, and FeatureGates
- Addons for easily installed Kubernetes applications
- Supports common CI environments

```
~ time minikube start
😄 minikube v1.13.0 on Darwin 10.15.6
✨ Using the docker driver based on user configuration
👍 Starting control plane node minikube in cluster minikube
🔥 Creating docker container (CPUs=2, Memory=3892MB) ...
🐳 Preparing Kubernetes v1.19.0 on Docker 19.03.8 ...
🔎 Verifying Kubernetes components...
🌟 Enabled addons: default-storageclass, storage-provisioner
💡 kubectl not found. If you need it, try: 'minikube kubectl -- get pods -A'
🏄 Done! kubectl is now configured to use "minikube" by default

_____________________________________________________________
Executed in   23.96 secs   fish           external
   usr time    1.66 secs  237.00 micros    1.66 secs
   sys time    0.78 secs  943.00 micros    0.78 secs

~
```

K3s and K3d

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

K3s is a lightweight tool designed to run production-level Kubernetes workloads for **low-resourced and remotely located IoT and Edge devices and Bare metal**.
Originally Created by **Rancher**, a Sandbox CNCF Project

RANCHER

K3s does not use kubelet, but it runs kubelet on the host machine and uses the host's scheduling mechanism to run containers

K3s uses kube-proxy to proxy the network connections of the nodes.
K8s uses kube-proxy to proxy the network connections of an individual container.

k3s can have tighter security deployment than k8s because of their small attack surface area.

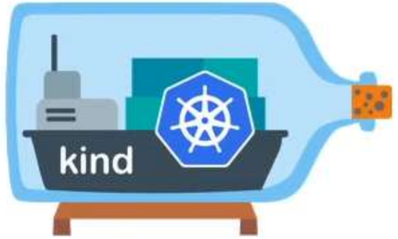
K3s has some advantage but comes with limitations and you'll need to investigate for yourself whether it makes sense to use K3s for your use case

K3D

K3d is a platform-agnostic, **lightweight wrapper** that **runs K3s in a docker container.**
It helps run and scale single or multi-node K3S clusters quickly without further setup while maintaining a high availability mode.

Kind

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



Primarily **designed to test Kubernetes**, Kind (Kubernetes in Docker) helps you run Kubernetes clusters locally and in CI pipelines using Docker containers as “nodes”.

```
$ time kind create cluster
Creating cluster "kind" ...
 ✓ Ensuring node image (kindest/node:v1.16.3) 🖼
 ✓ Preparing nodes 📦
 ✓ Writing configuration 📜
 ✓ Starting control-plane 🕹️
 ✓ Installing CNI 🔌
 ✓ Installing StorageClass 💾
Set kubectl context to "kind-kind"
You can now use your cluster with:

kubectl cluster-info --context kind-kind

Not sure what to do next? 😅 Check out https://kind.sigs.k8s.io/docs/user/quick-start/

real    0m21.890s
user    0m1.278s
sys     0m0.790s
```

It is an open source CNCF certified Kubernetes installer that supports highly available multi-node clusters and builds Kubernetes release builds from its source.

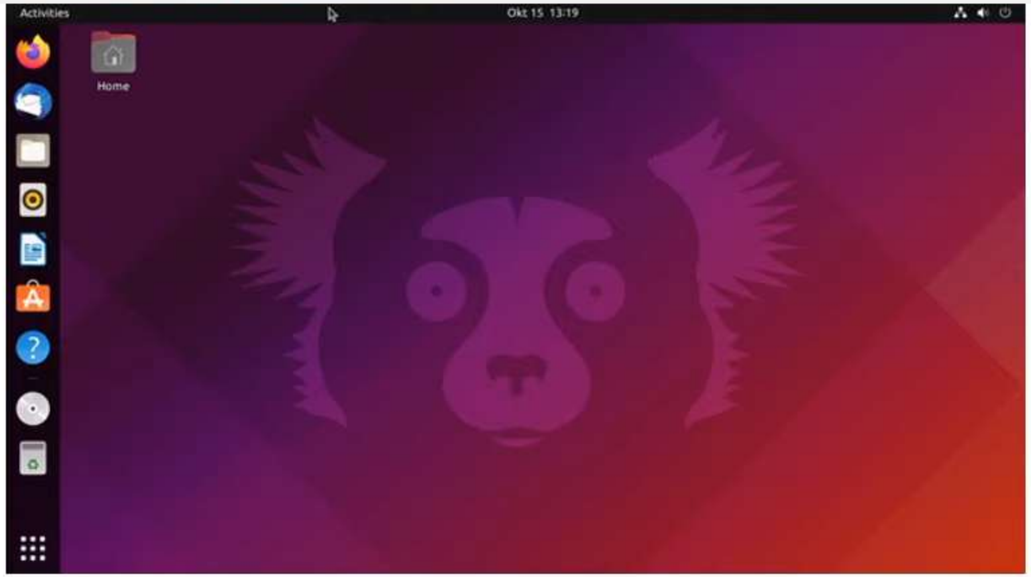
MicroK8S

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

Ubuntu is a Linux distribution based on Debian.
Ubuntu is known for:

- lots of Linux programs pre-installed
- the of the easiest Linux distributions to use
- More frequent updates
- More progressive on new linux programs and systems

It comes in many editions eg. **Desktop**, Server, Core



Canonical is the company that publishers of Ubuntu

Snap is a package manager by Canonical that can be installed on many different distributions of Linux.

```
sudo snap install ruby --classic
```

Classic Flag, allows access to your system's resources in much the same way traditional package. Without the flag snaps run in complete isolation.

MicroK8S

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

MicroK8s is created by Canonical and is installed using **Snap**.

MicroK8s

```
sudo snap install microk8s --classic
```

It is a a Kubernetes distribution designed to run fast, self-healing, and highly available Kubernetes clusters.

It is optimized for quick and easy installation of single and multi-node clusters on multiple operating systems, including macOS, Linux, and Windows *(as long as you have snap).*

It is ideal for running Kubernetes in the cloud, local development environments, and Edge and IoT devices

Microk8s is modular in design, you start with nothing and can **enable addons** to quickly use exactly what you need and nothing more:

```
brown-laptop:~/environment $ microk8s status
microk8s is running
high-availability: no
  datastore master nodes: 127.0.0.1:19001
  datastore standby nodes: none
addons:
  enabled:
    cilium               # SDN, fast with full network policy
    dashboard            # The Kubernetes dashboard
    dns                  # CoreDNS
    ha-cluster           # Configure high availability on the cu
    helm                 # Helm 2 - the package manager for Kube
    helm3                # Helm 3 - Kubernetes package manager
    ingress              # Ingress controller for external acces
    istio                # Core Istio service mesh services
    metrics-server       # K8s Metrics Server for API access to
    rbac                 # Role-Based Access Control for authori
    storage              # Storage class; allocates storage from
  disabled:
    ambassador           # Ambassador API Gateway and Ingress
    dashboard-ingress    # Ingress definition for Kubernetes das
    fluentd              # Elasticsearch-Fluentd-Kibana logging
    gpu                  # Automatic enablement of Nvidia CUDA
    host-access          # Allow Pods connecting to Host se
    inaccel              # Simplifying FPGA management in Kuberne
```

Lightweight K8s Distribution Comparison

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

Minikube runs in a Virtual Machine
Intended just for development purposes.
Very easy to use, very popular

Kind is designed to run anywhere container runs.
Intended just for development purposes
Faster startup time than a Minikube since its not spinning up a VM

K3s and K3D is a K8s distribution by rancher
Can be used for production use-cases
Designed for embedded, edge devices, or limited resources.

MicoK8s is created by Canonical, Need Snap to install
Modular, starts with nothing installed.
Restarts everything if there is a crash.
We'll suited for self-hosted production use-cases

Managed Kubernetes Providers

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

Managed Kubernetes providers are Cloud Service Providers (CSPs) or platforms that **abstracts away** the effort of setting up, maintaining (updating and patching) a cluster. They can easy autoscaling as well.

Google Kubernetes Engine (GKE)
The easiest to use with the richest amount of features

Amazon Elastic Kubernetes Service (EKS)
Difficult to use via the UI, powerful CLI tool
Can be worth it for integrations with other AWS services

Azure Kubernetes Service (AKS)
Fairly easy to use. Unique service offers for debugging live containers.

IBM Cloud Kubernetes Service
Easy to use, not feature rich. More expensive than any other cloud service provider

Oracle Container Engine for Kubernetes
Cost effective for a cloud service provider, worst UI, limited options

DigitalOcean Kubernetes (DOKS)
Very easy to use, predictable spend.
Beautiful UI

CIVO Kubernetes
Most cost effective. Simple UI
A cloud platform specifically focused on just Kubernetes.

Management Layers

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

A management layer for running Kubernetes on other platforms or allows you to extend your control plane to multiple platform

Weave Kubernetes Platform (WKP)
All of Weaves open-sources tools packaged as a platform so you can build out a GitOps enabled cluster.

Azure Arc *multi-cluster-management*
Governing compute such as K8s from other CSPs or on-premise or the edge.

Rafay
Similar to OpenShift with a larger focus on governance and GitOps-based management for any K8s clusters running on anything (including OpenShift)

Google Anthos *multi-cluster-management*
Is GKE being extended to mange clusters deployed to VMs on other cloud's or on-premise. Its focused on managing K8.

VMWare Tanzu
Wherever vSphere runs, you can mange you can deploy and monitor Kubernetes clusters.

PLATFORM9

Platform9
Similar to RayFay but relies more on third-party tooling instead of trying to leverage native functionality from public cloud service providers.

Management Layers

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

Red Hat OpenShift *Platform as a Service for K8s*

- Openshift is Kubernetes with a commercial platform by RedHat built on-top.
- Kubectl is extended with additional functionality with the oc cli
- quickly deploy local code to a remote OpenShift cluster via odo
- A quality assurance pipeline built into the platform
- Fixing critical bugs earlier instead of waiting for next K8s release
- Using Redhat CoreOS (an operating system optimized for running containers)
- OperatorsHub, an automated installation tool (one click marketplace)
- Graphical UI developer console
- CodeReady workspaces, Cloud Developer Environment for Kubernetes

Rancher Kubernetes Engine (RKE)

- Runs entirely within Docker containers.
- It works on bare-metal and virtualized servers.
- RKE solves the problem of installation complexity, a common issue in the Kubernetes community.
- Installation and operation of Kubernetes is both simplified and easily automated,
- It’s entirely independent of the operating system and platform you’re running.
- As long as you can run a supported version of Docker, you can deploy and run Kubernetes with RKE.

Container Runtime Interfaces

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

Orchestration

Docker

Kubernetes

Orchestration systems will use a container runtime interface.

Container Runtime Interface (CRI)

Containerd

CRI-O

Container Runtime Interface allows you to run a variety of different container runtimes.
Eg. Push and pull images, supervise containers

Open Container Initiative (OCI) Runtimes

Native Runtimes	Sandboxed / Virtualized Runtimes
• runC • Crun	• gviso • nabla-containers • Kata-containers

Runtimes create and run containers.
The major difference between native and virtual is ***isolation***.

Virtualized can provide security benefits through isolation.

ContainerD

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

ContainerD is an industry-standard container runtime with an emphasis on simplicity, robustness and portability.

Docker extracted their container runtime in the project ContainerD and then donated it to CNCF

This includes Docker's functionality for executing containers, handling low-level storage and managing image transfers.

containerd makes it easier for projects like Kubernetes to access the low-level "Docker" elements they need. Instead of actually using Docker

Images you build with Docker aren't really "Docker images".

- Docker now uses the containerd runtime
  - your images are built in the standardized Open Container Initiative (OCI) format.

CRI-O

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

CRI-O is an implementation of the Kubernetes CRI (Container Runtime Interface) to enable using OCI (Open Container Initiative) compatible runtimes.

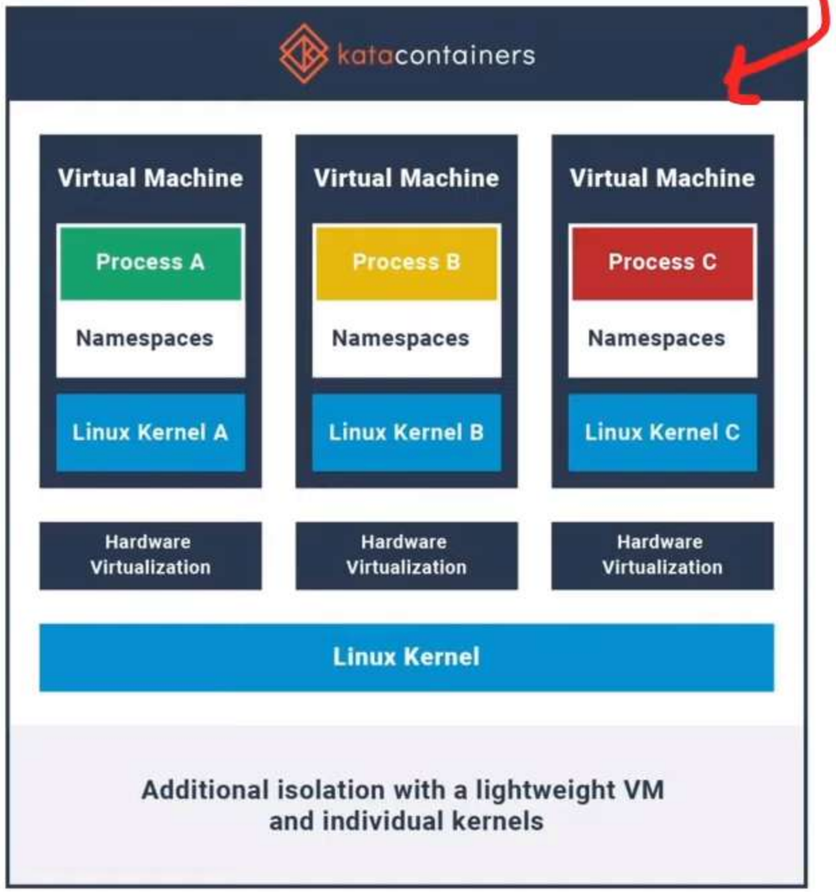
It is a **lightweight alternative to using Docker (containerd)** as the runtime for Kubernetes.

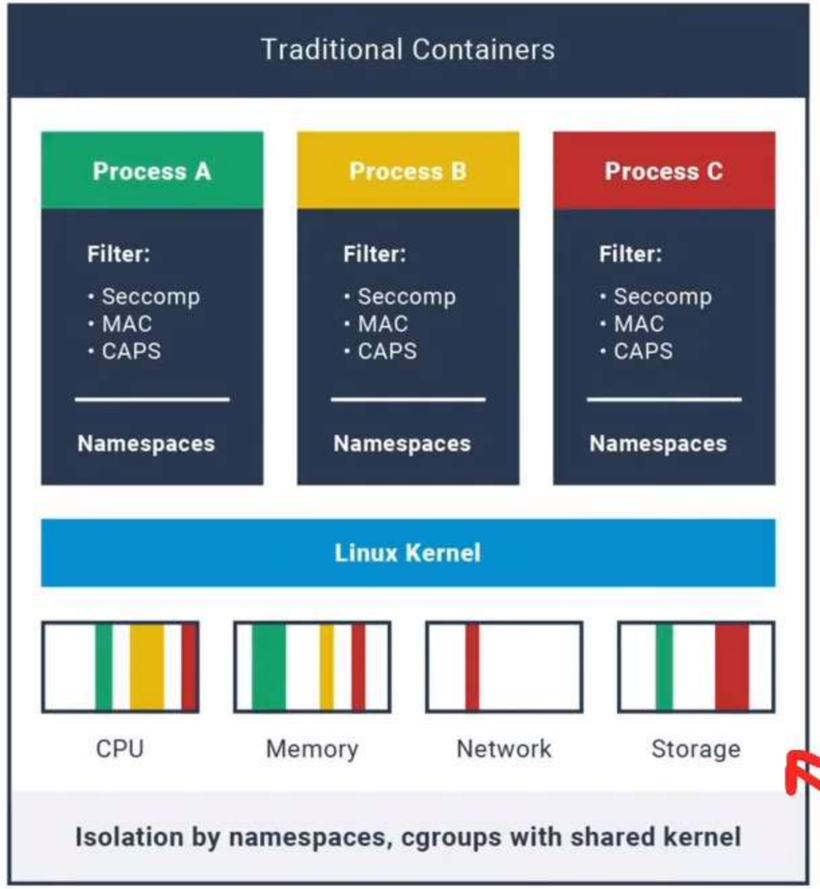
It allows Kubernetes to use any OCI-compliant runtime as the container runtime for running pods. Today it supports **runc** and **Kata Containers** as the container runtimes but any OCI-conformant runtime can be plugged in principle.

Container Runtimes

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

Virtualized Runtimes use **lightweight Vms** for isolation





Native Runtimes with use **cgroups** for isolation

Container Runtimes

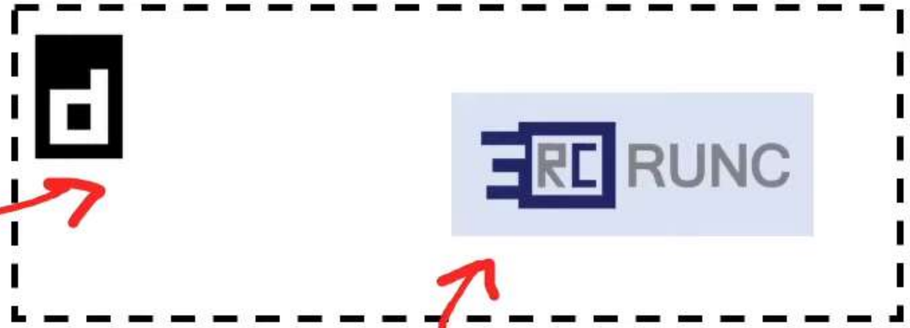
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

Runc is a low-level container runtime that **creates and runs containers**.

Runc would be used alongside a ContainerD or CRI-O

daemon process that manages and runs containers.

- pushes and pulls images manages storage and networking
- supervises the running of containers



creates and runs containers

CGroups

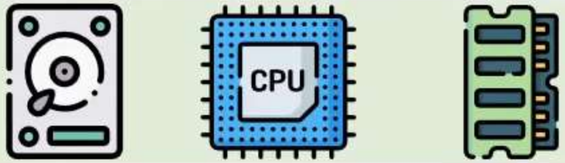
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

What is a process?
A process is an instance of a running program on Linux

What is a CGroup?
Control groups (cgroups) allows you to group processes to apply different kinds of limitations:

- **Resource limiting —** groups can be set to not exceed a configured memory limit, which also includes the file system cache
- **Prioritization —** some groups may get a larger share of CPU utilization or disk I/O throughput
- **Accounting —** measures a group's resource usage, which may be used, for example, for billing purposes
- **Control —** freezing groups of processes, their checkpointing and restarting

Think of CGroups as a way to limit programs on linux
From overusing CPU, Memory or Storage.



The primary design goal for cgroups was to provide a unified interface to manage processes or whole operating-system-level virtualization, including **Linux Containers (LXC)**

Linux Containers

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

LXC (LinuX Containers) is a OS-level virtualization technology that allows creation and running of multiple isolated Linux virtual environments (VE) on a single control host.

VEs vs VMs

VE there is no preloaded emulation manager software as in a VM

In a VE, the application (or OS) is spawned in a container and runs with no added overhead, except for a usually minuscule VE initialization process

There is no hardware emulation, which means that aside from the small memory software penalty

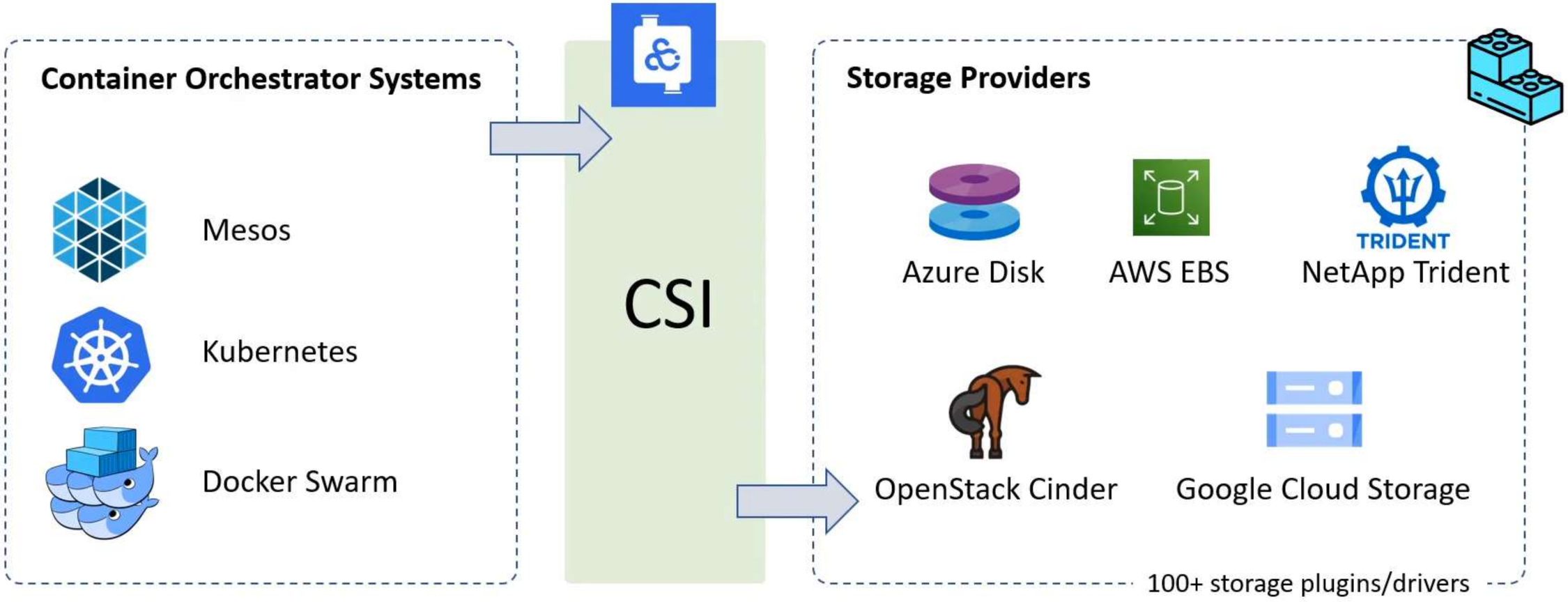
LXC will boast bare metal performance characteristics because it only packages the needed application

Ves cannot be easily managed via neat GUI management consoles and they don't offer some other neat features of VM's such as IaaS setups and live migration.

Container Storage Interface (CSI)

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

Container Storage Interface (CSI) standardizes how
Container Orchestrator Systems (COS) access to various **storage providers**



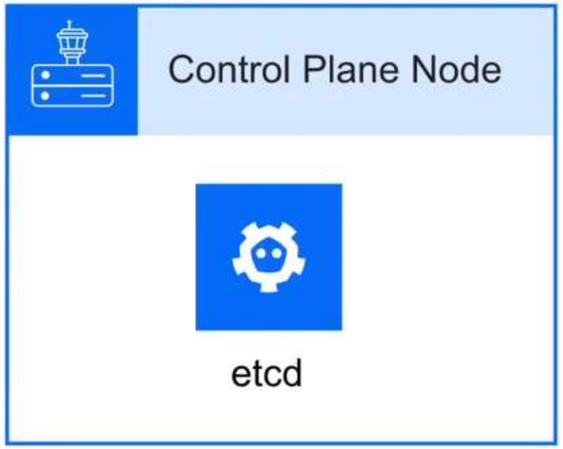
Kubernetes Backing Store and etcd

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

Components of a Kubernetes cluster include: pods, nodes, control plane, and volumes, and each of these needs a level of protection in case of disaster

Kubernetes resources are stored in an etcd (but could be backed by MariaDB).
Application data is stored in persistent volumes for applications running on the cluster.

etcd is a strongly consistent, distributed key-value store that provides a reliable way to store data that needs to be accessed by a distributed system or cluster of machines.



Things that use etcd:

- Kubernetes Cluster
- CoreDNS
- Rook

MinIO and Rook

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

Rook turns distributed storage systems into self-managing, self-scaling, self-healing storage services.

It automates the tasks of a storage administrator: deployment, bootstrapping, configuration, provisioning, scaling, upgrading, migration, disaster recovery, monitoring, and resource management.

MinIO offers high-performance, **S3 compatible object storage**.

Native to Kubernetes, MinIO is the only object storage suite available on every public cloud, every Kubernetes distribution, the private cloud and the edge.

MinIO is software-defined and is 100% open source under GNU AGPL v3.

Volumes

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

Kubernetes supports many types of volumes.

A Pod can use any number of volume types simultaneously

Persistent Volumes
Attaching an external storage to a pod.
The data will persist even if the pod is terminated.

Ephemeral Volumes
A volume that's only exists as long as the pod exists.
Intended for temporary data storage.

Projected Volumes
maps several existing volume sources into the same directory.

Volume Snapshot
Archiving a volume configuration and its data for rollbacks or backup

Types of **Volumes Supported**:

- awsElasticBlockStore
- azureDisk
- azureFile
- cephfs
- cinder
- configMap
- downwardAPI
- emptyDir
- fc (fibre channel)
- gcePersistentDisk
- glusterfs
- hostPath
- iscsi
- local
- nfs
- persistentVolumeClaim
- portworxVolume
- projected
- rbd
- secret
- vsphereVolume

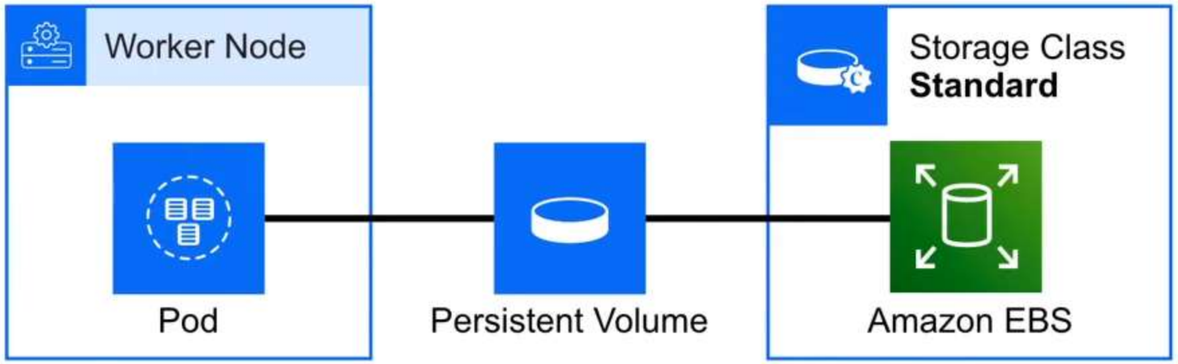
Persistent Volume (PV)

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

A piece of storage in the cluster that has been provisioned by an administrator or dynamically provisioned using Storage Classes

PV is similar to node in that is a cluster resource.

PV does not reside within a node or pod.



This diagram is **technically not accurate*

PVs are volume plugins like Volumes, but have a lifecycle independent of any individual Pod that uses the PV

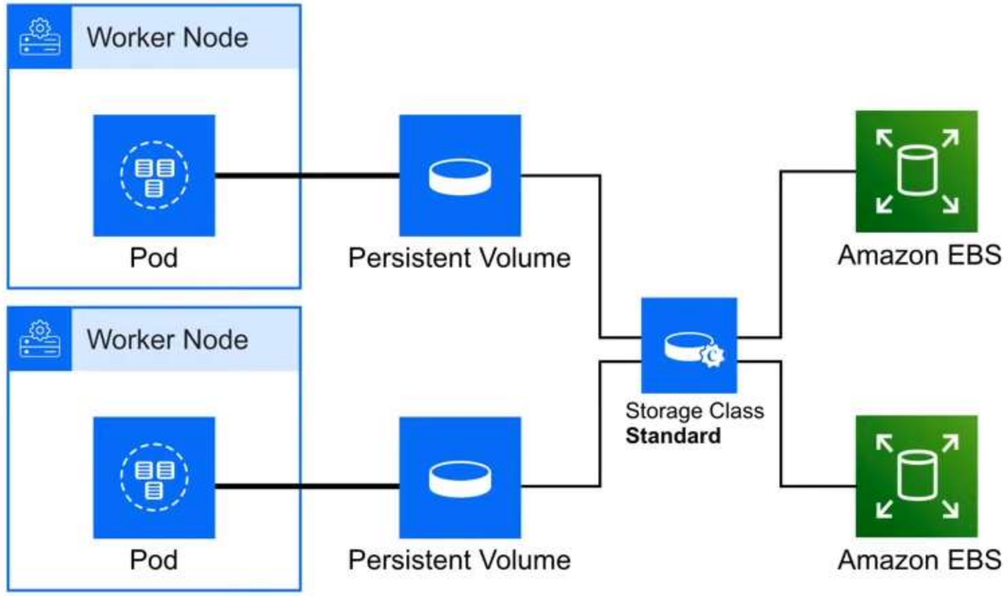
This API object captures the details of the implementation of the storage, be that NFS, iSCSI, or a cloud-provider-specific storage system

mounting pv directly to a pod is not allowed and is against the Kubernetes design principles. It would cause tight coupling below the pod volume and the underlying storage. **Persistent Volume Claims** ensures decoupling

Storage Classes

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

A storage class is way of defining a class of storage that is backed by a provisioner.




```yaml
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard
provisioner: kubernetes.io/aws-ebs
parameters:
  type: gp2
reclaimPolicy: Retain
allowVolumeExpansion: true
mountOptions:
  - debug
volumeBindingMode: Immediate
```

Storage Class can be used by many Persistent Volumes.

Storage class contains:

- Provisioner – who is storage? Eg. Amazon EBS
- Parameters – what type of storage of Amazon EBS to use
- reclaimPolicy — If the pod is gone does the volume remain?

Persistent Volume Claim (PVC)

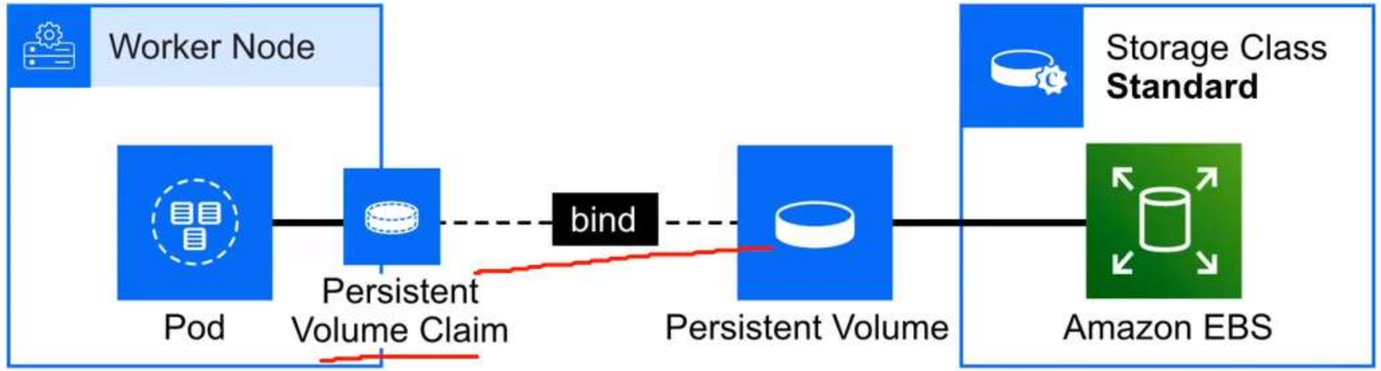
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

A Persistent Volume Claim (PVC) is used to decouple Persistent Volumes from pods. A PVC asked for a type of storage, and if a PV that meets the criteria is matched, the a PV is Claimed and Bound.

PVCs similar to a Pod requesting resources from a Node

- Pods consume node resources
  - PVCs consume PV resources
- Pods can request specific levels of resources (CPU and Memory)
  - PVCs can request specific size and access modes (e.g., they can be mounted ReadWriteOnce, ReadOnlyMany or ReadWriteMany, see AccessModes).

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: myclaim
spec:
  accessModes:
    - ReadWriteOnce
  volumeMode: Filesystem
  resources:
    requests:
      storage: 8Gi
  storageClassName: slow
  selector:
    matchLabels:
      release: "stable"
    matchExpressions:
      - {key: environment, operator: In, values: [dev]}
```



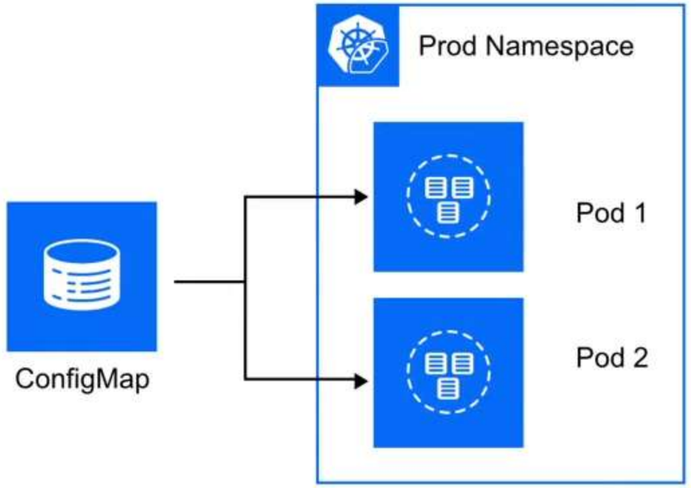
ConfigMaps

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

A ConfigMap is an API object used to store **non-confidential data** in key-value pairs for pods

Pods can consume ConfigMaps as:

- environment variables
- command-line arguments
- configuration files in a volume



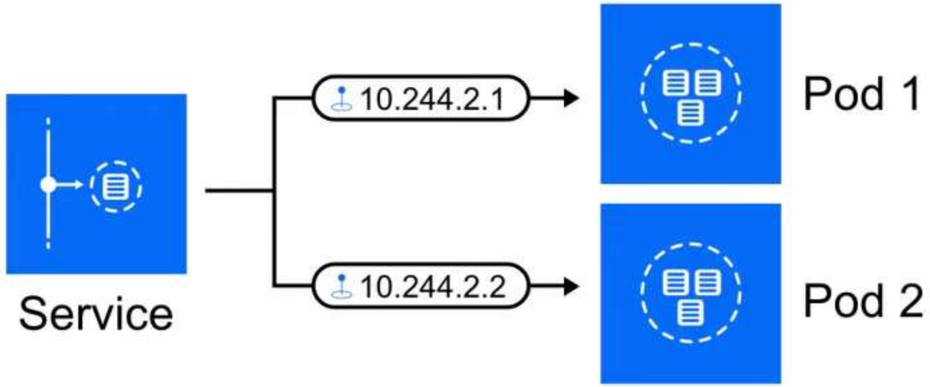

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: game-demo
data:
  # property-like keys; each key maps to a simple value
  player_initial_lives: "3"
  ui_properties_file_name: "user-interface.properties"
  # file-like keys
  game.properties: |
    enemy.types=aliens,monsters
    player.maximum-lives=5
  user-interface.properties: |
    color.good=purple
    color.bad=yellow
    allow.textmode=true
```

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

Kubernetes Services

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

Kubernetes Service allows you to attach a **static IP address** and DNS name for a set of pods
A K8s Service **allows you to persist an address** for a pod even if it dies.
K8s Service also acts a load balancer



A Pod without a service will have a dynamic IP address.
So when the pod dies so does the IP address.

Kubernetes Services have the following **service types**:

- **ClusterIP** – default, randomly forward traffic to any pod set with target port
- **Headless** – send traffic to very specific pod, when you have stateful pods eg. Database
- **NodePort** – external service, allows you to use worker node IP address
- **LoadBalancer** – similar to nodeport except leverages Cloud Service Provider's (CSPs) load balancer
- **ExternalName** — a special service that does not have selectors and uses DNS names instead

Kubernetes Services – Traffic Policies

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

K8s Service allows you set a Traffic policies to determine how ingress traffic is routed.

There are 2 types of Traffic policies

1. External Traffic Policy
how traffic from external sources is routed and has two valid values:

- Cluster – route external traffic to all ready endpoints
- Local - only route to ready node-local endpoints

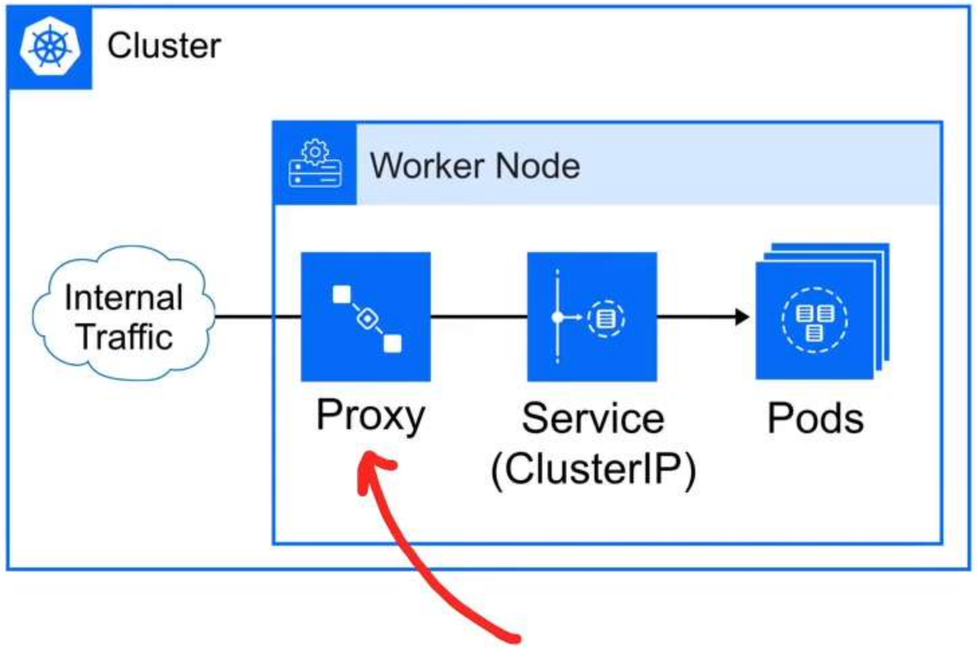
2. Internal Traffic Policy
how traffic from internal sources is routed (has the same two values as External)

🚦 If the traffic policy is Local and there are are no node-local endpoints, then kube-proxy does not forward any traffic for the relevant Service

ClusterIP

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

ClusterIP is the default service type for a K8 service.
It is used for internal traffic. External traffic will not reach the service.
Traffic will be randomly distributed to any targeted pods.



If you **omit type** it will default to ClusterIP

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-internal-service
spec:
  selector:
    app: my-app
  type: ClusterIP
  ports:
  - name: http
    port: 80
    targetPort: 80
    protocol: TCP
```

Traffic originating from within the cluster will pass through the Node's **Kubernetes Proxy** and then onto to Kubernetes Service

A service can span multiple worker nodes for cross-node pods.

When to use ClusterIP:

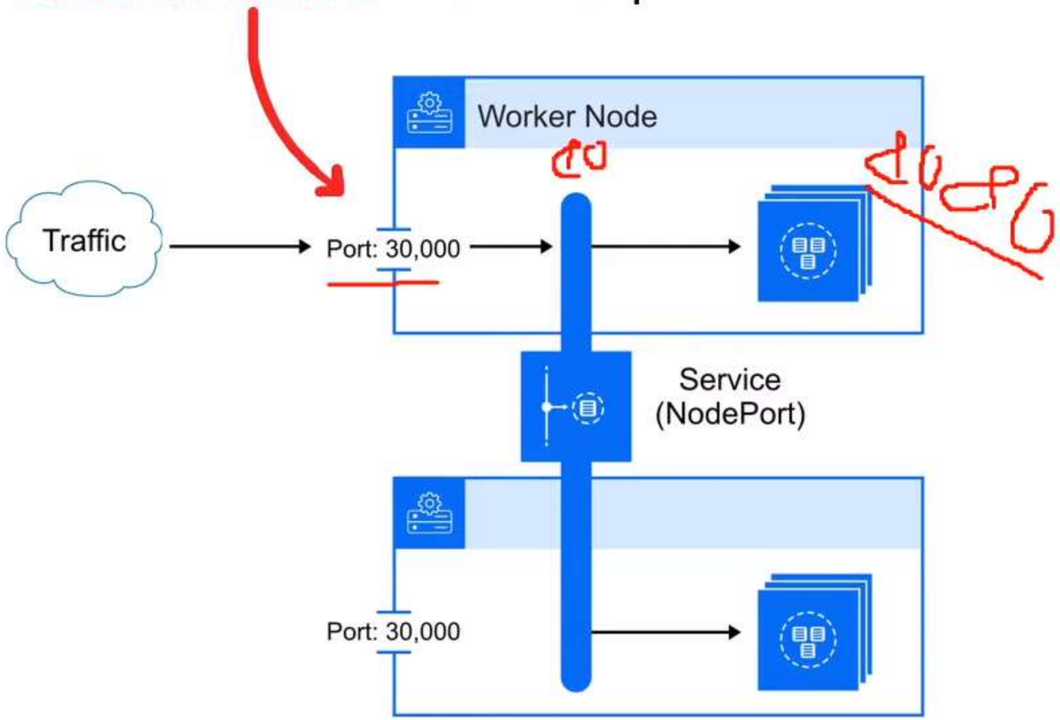
- Debugging
- Testing
- Internal traffic
- Internal Dashboards

NodePort

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

NodePort allows you to expose a port for Virtual Machines running pods that the Service is managing.

External traffic can reach pods



Set **type** to NodePort

Set **nodePort**

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
spec:
  selector:
    app: my-app
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30036
    protocol: TCP
```

There is no external load balancer so NodePort is intended for a single Kubernetes Service and for non production workloads.

NodePort

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

Port
Port exposes the Kubernetes service on the specified port within the cluster.
Other pods within the cluster can communicate with this server on the specified port.

TargetPort
Target Port is the port on which the service will send requests to, that your pod will be listening on. Your application in the container will need to be listening on this port also.

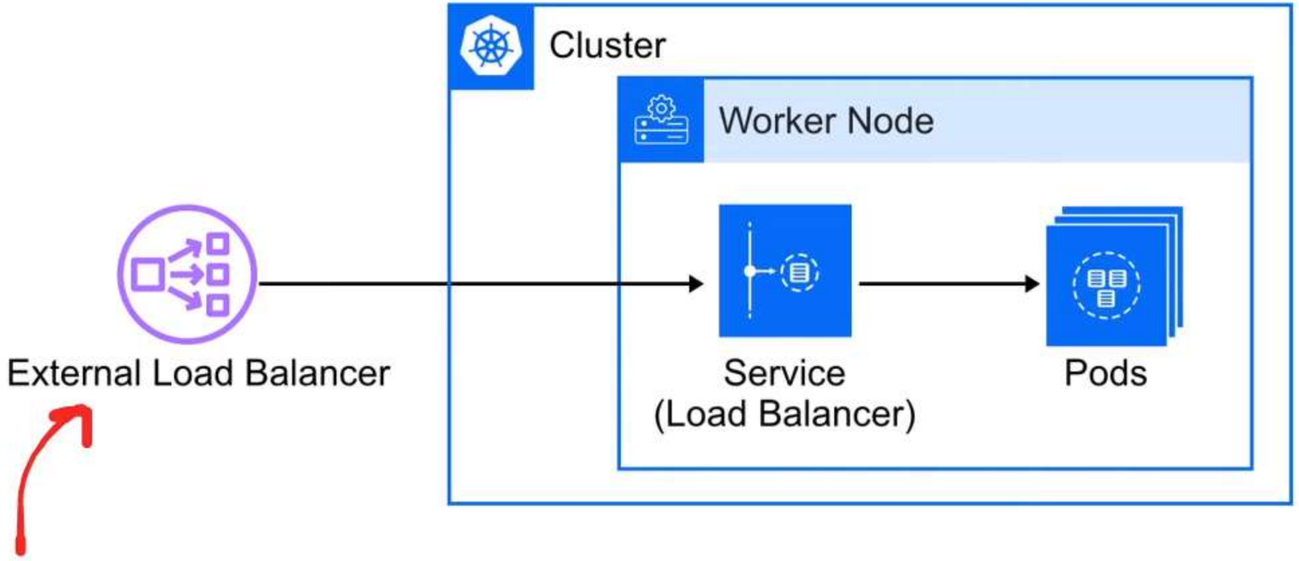
NodePort
NodePort exposes a service externally to the cluster by means of the target nodes IP address and the NodePort. NodePort is the default setting if the port field is not specified.

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-nodeport-service
spec:
  selector:
    app: my-app
  type: NodePort
  ports:
  - name: http
    port: 80
    targetPort: 80
    nodePort: 30036
    protocol: TCP
```

Load Balancer

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

A Load Balancer service type allows you to use an External Load Balancer.
The External Load Balancer handles the routings and traffic distribution logic



An external load balancer would be from a **managed third-party cloud service**.
Eg. AWS Network Load Balancer (NLB)

Load Balancer type is well suited for production workloads.
Generally its recommended to use Kubernetes Ingress

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: MyApp
  ports:
    - protocol: TCP
      port: 80
      targetPort: 9376
  clusterIP: 10.0.171.239
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 192.0.2.127
```

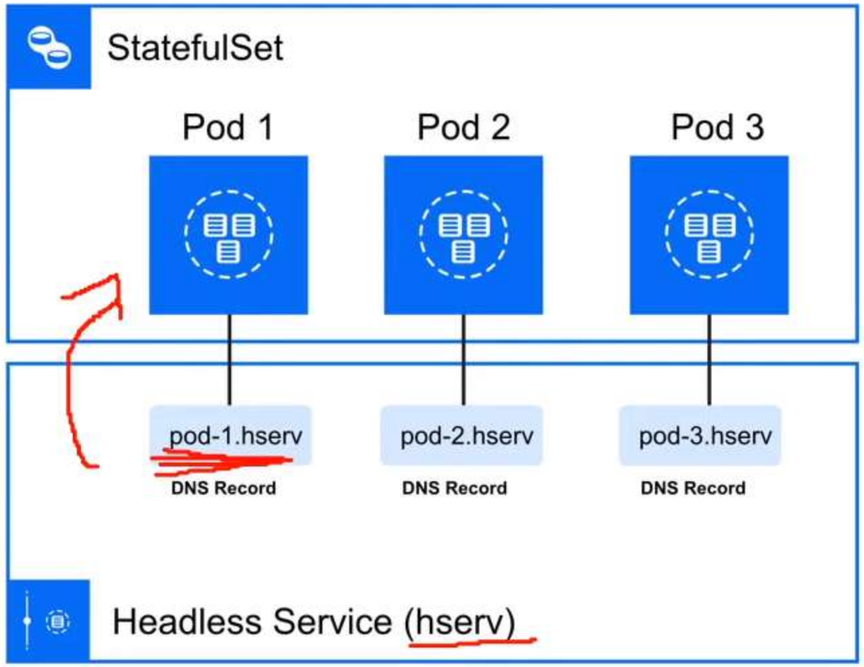
Headless

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

- A headless service is a service with no ClusterIP address
- A headless service does not provide load balancing or proxying

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  ports:
  - port: 80
    name: web
  clusterIP: None
  selector:
    app: nginx
```

Headless are useful when you are dealing with a Stateful application (reads and writes) and you need writes to go to a specific pod.



Headless Service is needed to manage network identity of the stateful pods by assigning a DNS Record to each pod so you can route traffic to a DNS Hostname.

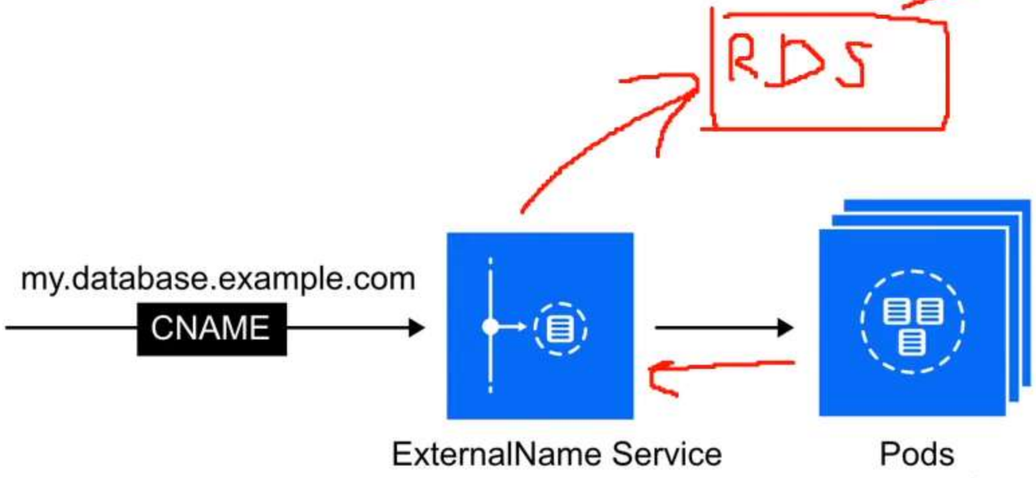
ExternalName

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

ExternalName Services as the same as ClusterIP Service with the exception of instead returning a StaticIP it returns a CNAME record.

What is a CNAME record?

Canonical Name (CName) record is a DNS record that maps one doman name (an alias) to another name (canonical name)




```yaml
kind: Service
apiVersion: v1
metadata:
  name: my-service
spec:
  type: ExternalName
  externalName: my.database.example.com
```

KubeCTL Expose Command

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

Expose is used to quickly create Kubernetes Services for a **deployment**

```
kubectl expose deployment my-app --type=NodePort \
--name=my-service --port=80 --targetport=8080 --nodeport=3000
```

BusyBox

BusyBox combines tiny versions of many common UNIX utilities into a single small executable

It's The Swiss Army knife of Embedded Linux, as the single executable replaces basic functions of more than **300 common commands**

BusyBox can be used to **interactively debug Services** to ensure they are working.

```
kubectl run -it --rm --restart=Never busybox --image=gcr.io/google-containers/busybox sh
```

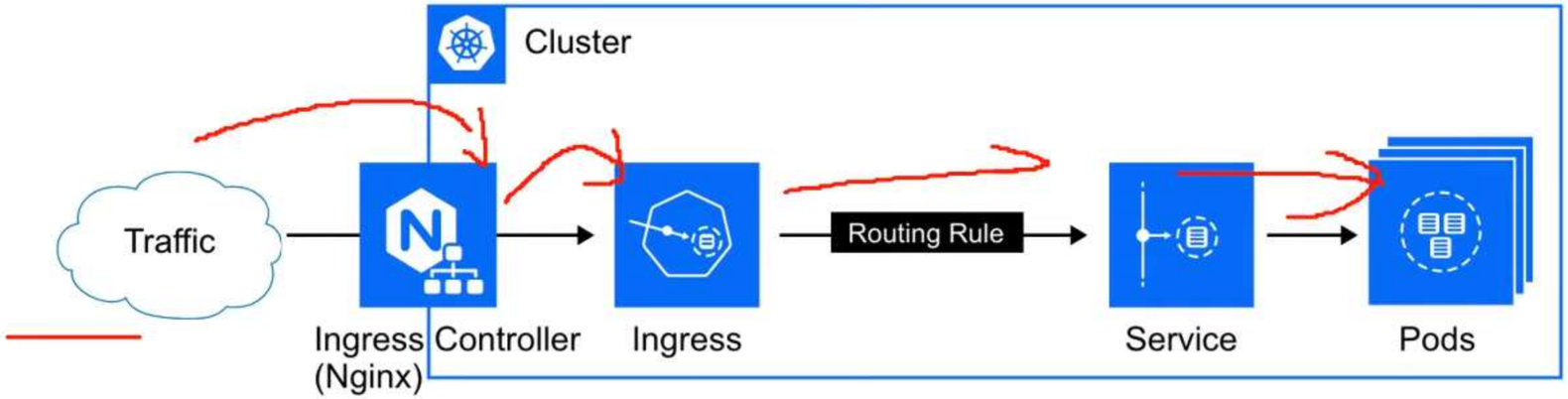
Kubernetes Ingress

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

Ingress exposes HTTP and HTTPS routes from outside the cluster to services within the cluster.
Traffic routing is controlled by rules defined on the Ingress resource.

The reason we use a K8s Ingress is so we can translate a custom domain on SSL to a service running within our K8s cluster.

In order for Ingress to work you need to use an Ingress Controller eg. AWS, GCE, Nginx



Ingress enables you to consolidate the traffic-routing rules into a single resource and runs as part of a Kubernetes cluster.

DNS

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

What is a Domain Name System (DNS)?

It is a service that is responsible for **translating (or resolving) a service name to its IP address**.

Service Registry	
frontend	10.90.142.14
backend	10.90.52.46
admin	10.90.41.101
...	...

CoreDNS is the default DNS server for Kuberentes and **ensures pods and services** haves Fully Qualified Domain Name (FQDN). Without CoreDNS the cluster communication would cease to work.

What is a FQDN?

a domain name that specifies its exact location in the tree hierarchy, also known as an absolute domain.

DNS

Lets take a look at the functionality provided via CoreDNS plugins

In-Tree Plugins (Internal Plugins)

- acl - enforces access control policies on source ip and prevents unauthorized access to DNS servers
- any - any gives a minimal response to ANY
- azure - enables serving zone data from Microsoft Azure DNS service
- cache - enables a frontend cache
- health - enables a health check endpoint
- log - enables query logging to standard output
- and many more…

Out-of-Tree Plugins (External Plugins)

- git – pull git repositories
- alias – replaces zones apex CNAMEs
- redisc – enables a networked cache using Redis
- Kubernetai – serve multiple Kubernetes within a Server
- and many more …..

DNS

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

CoreDNS pods are abstracted by a service object called kube-dns.

```
kubectl get service kube-dns -n kube-system
```

Each pod (any pod, not just CoreDNS pods) has a **resolv.conf** file to help with DNS resolving.

```
kubectl exec -it my-pod -- sh
cat /etc/resolv.conf
# nameserver 10.100.0.10
# search default.svc.cluster.local svc.cluster.local cluster.local ec2.internal
# options ndots:5
```

nslookup can be used to see what resolves Stored with CoreDNS

```
nslookup my-service-name
# Server:     10.100.0.10
# Address:    10.100.0.10#53
# Name:       kubernetes.default.svc.cluster.local
# Address:    10.100.0.1
```

Load Balancing

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

What is Load Balancing?
Load Balancing is a networking component, where traffic flows through the load balancer, and the load balancer decides how to distribute the traffic to multiple targets eg. (compute nodes) based on a set of rules.

Ingress and **Service** K8s both have load balancing.

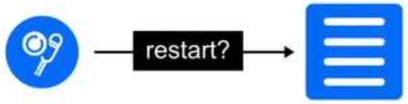
External Load Balancer	Ingress	Service	Internal Load Balancing (iptables / ipvs)
Load balancing control By third-party service	load balancing algorithm backend weight scheme	persistent sessions dynamic weights	Load balancing to containers within pods. Randomly distributed

Probes

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

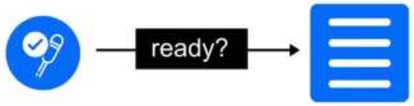
Probes are used to detect the state of a container

Liveness Probe



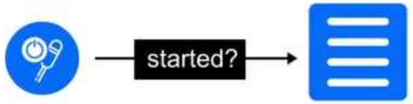
The kubelet uses liveness probes to know **when to restart a container.**
For example, liveness probes could catch a deadlock, where an application is running, but unable to make progress. Restarting a container in such a state can help to make the application more available despite bugs.

Readiness Probe



The kubelet uses readiness probes to know **when a container is ready to start accepting traffic.**
A Pod is considered ready when all of its containers are ready.
One use of this signal is to control which Pods are used as backends for Services.
When a Pod is not ready, it is removed from Service load balancers.

Startup Probe



The kubelet uses startup probes to know **when a container application has started.**
If such a probe is configured, it disables liveness and readiness checks until it succeeds, making sure those probes don't interfere with the application startup.
This can be used to adopt liveness checks on slow starting containers, avoiding them getting killed by the kubelet before they are up and running.

Netfilter

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

The netfilter project enables:

- packet filtering
- network address and port translation (NAPT)
- Translation packet logging
- Userspace packet queueing
- other packet mangling

The netfilter hooks are a framework inside the Linux kernel that allows kernel modules to register callback functions at different locations of the Linux network stack.

The registered callback function is then called back for every packet that traverses the respective hook within the Linux network stack.

Projects build ontop of Netfilter:

- Iptables — generic firewalling software that allows you to define rulesets
- Nftables – successor to iptables, more flexible, scalable and performance packet classification
- IPVS — specifically designed for load balancing, uses hash mapping

IP Tables

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

What is a Userspace?

Modern computer operating system segregates virtual memory into kernel space and user space

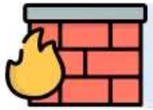
Kernel space is reserved for running a privileged operating system kernel, kernel extensions, and device drivers.

User space is the memory area where application software and some drivers execute.

What is IP Tables?

iptables is a user-space utility program that allows a system administrator to configure the IP packet filter rules of the Linux kernel firewall

- *iptables* applies to IPv4
- ip6tables to IPv6



Iptables are simply virtual firewalls on Linux

It is common to and modify iptables to restrict access based on ports and protocols

```
iptables -I INPUT 1 -p tcp --dport 80 -j ACCEPT
```

```
iptables -L
Chain INPUT (policy DROP)
target     prot opt source               destination
ACCEPT     tcp  --  anywhere             anywhere             tcp dpt:http
ACCEPT     all  --  anywhere             anywhere             state RELATED,ESTABLISHED
ACCEPT     icmp --  anywhere             anywhere
ACCEPT     all  --  anywhere             anywhere
ACCEPT     tcp  --  anywhere             anywhere             state NEW tcp dpt:ssh

Chain FORWARD (policy ACCEPT)
target     prot opt source               destination

Chain OUTPUT (policy ACCEPT)
target     prot opt source               destination
```

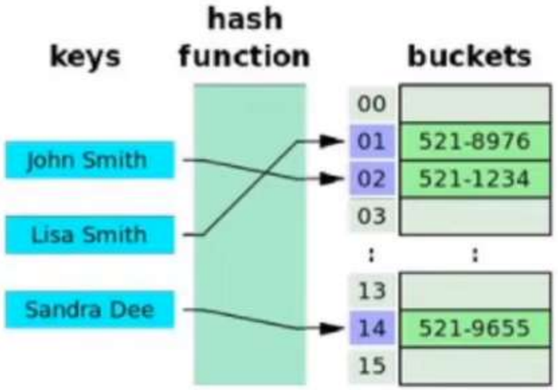
IPVS

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

IP Virtual Server (IPVS) uses the NetFilter framework.
IPVS also incorporates LVS (Virtual Linux Server)

Iptables struggles to scale to tens of thousands of services as Iptables is bottlenecked at 5000 nodes per cluster

IPVS is specifically designed for load balancing and uses more efficient data structures (hash tables) allowing for almost unlimited scale under the hood

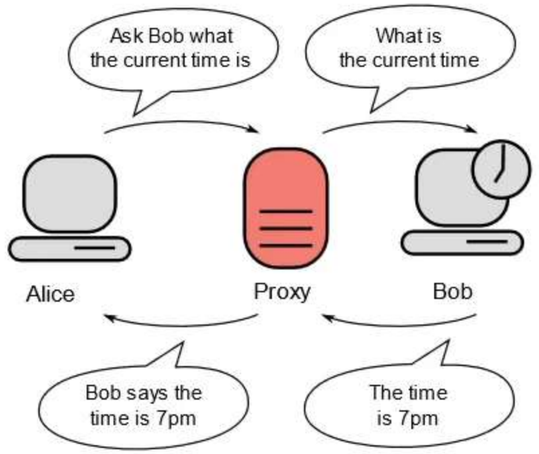


In the future the Kube Proxy will default to using IPVS.

Various Proxies

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

What is a proxy?
a server application that acts as an intermediary between a client requesting a resource and the server providing that resource



There are **many kinds of proxies you will encounter** in Kubernetes:

- **Kubectl proxy** — proxies from a localhost address to the Kubernetes apiserver
- **Apiserver proxy** — a bastion built into the apiserver, connects a user outside of the cluster to cluster IPs which otherwise might not be reachable
- **Kube proxy** — runs on each node and used to reach services
- **Proxy/Load balancer in front of API servers** — acts as load balancer if there are several apiserver
- **Cloud Load Balancers** — for external cluster traffic to reach pods

- **Forward Proxy:** A bunch of servers egressing traffic have to pass through the proxy first
- **Reverse Proxy:** Ingress traffic trying to reach a collection of servers

Kubernetes Proxy

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

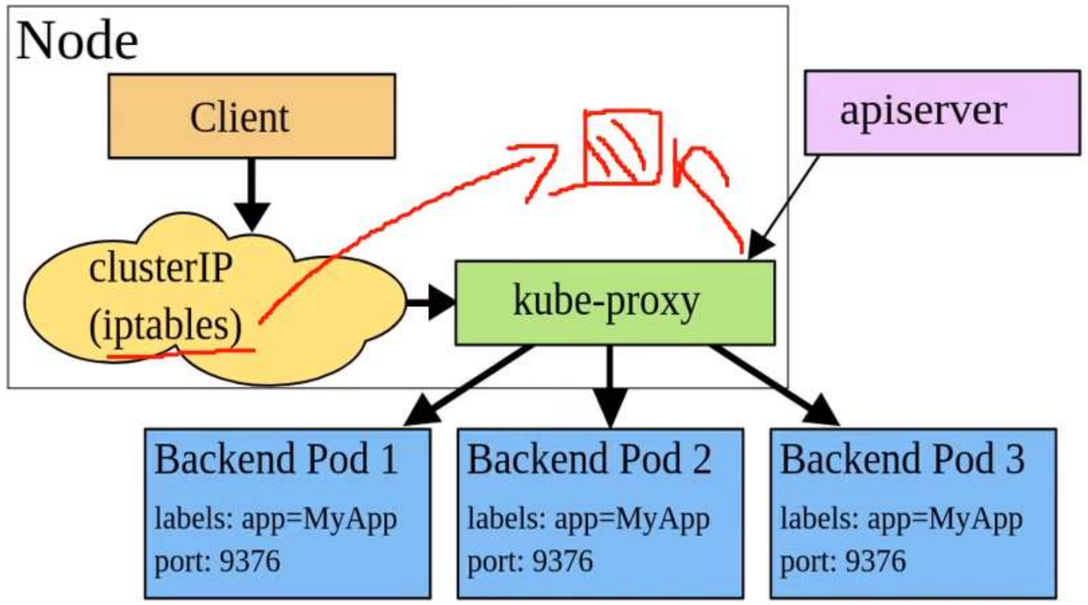
kube-proxy is a network proxy that **runs on each node** in your cluster.
It is designed to load **balance traffic to pods**.

kube-proxy maintains network rules on nodes. These network rules allow network communication to your Pods from network sessions inside or outside of your cluster.

kube-proxy uses the operating system packet filtering layer (if there is one and it's available). Otherwise, kube-proxy forwards the traffic itself.

Kube-proxy can run in three modes:

1. **iptables** (default). — Suited for simple use cases
2. **Ipvs** — Suites for 1000+ services.
3. **Userspace** (legacy) — Not recommended for use

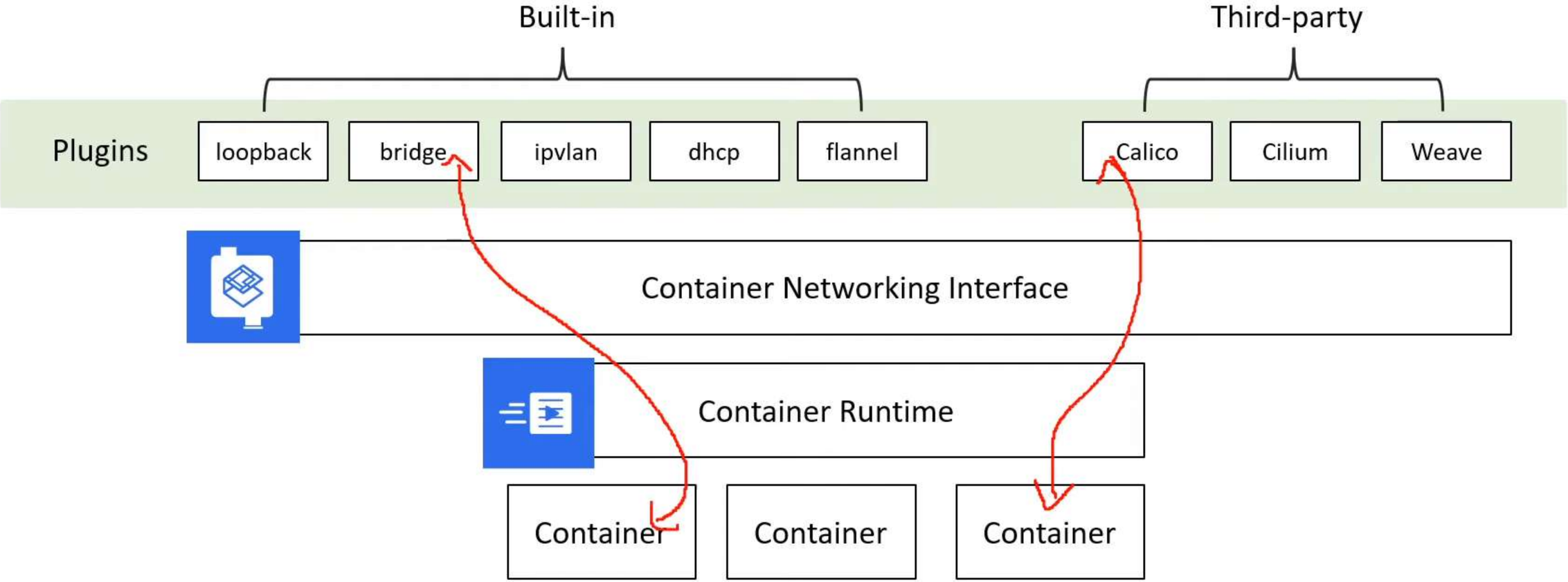


Container Networking Interface (CNI)

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

CNI

The **Container Networking Interface (CNI) is** a specification (open standard) for writing plugins to configure networking interfaces for linux containers



Service Mesh

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

You don't have to run a service mesh, but in most production use-cases it is recommended to do so.

Available Service Meshes for Kubernetes

Istio is the currently most popular service mesh for Kubernetes due to its highly configurable nature.
Istio uses Envoy as its proxy. Istio is a not a CNCF project.

Envoy is an open-source edge and service proxy. Multiple Service Meshes uses Envoy as its proxy.
Envoy is a graduated CNCF project.

Kuma is a CNCF sandbox project that uses Envoy as its proxy.

Linkerd is a CNCF graduated project is known for having strong security and "it just works".
Linkerd does not use Envoy, instead it uses a simple and ultralight "micro-proxy" called Linkerd2-proxy

Consul is an open-source service mesh by Hashicorp. Its not a CNCF project.
Consul is offered by Hashicorp as managed cloud Service Mesh.

Envoy

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

Envoy is a self contained process that is designed to run alongside every application server.
Envoy can be installed on a Virtual Machine or as a Container.

Envoy supports a wide range of functionality

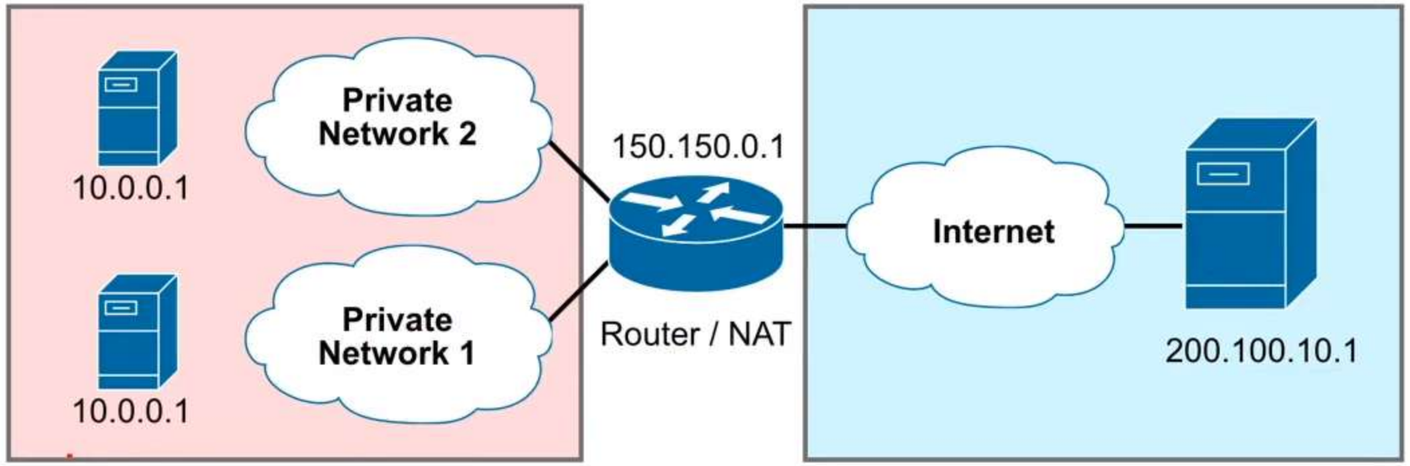
- L3/L4 filter architecture
- HTTP L7 filter architecture:
- First class HTTP/2 support
- HTTP/3 support
- HTTP L7 routing
- gRPC support
- Service discovery and dynamic configuration
- Health checking
- Advanced load balancing
- Front/edge proxy support
- Best in class observability

In practice you likely will not install and manually configure Envoy,
You would allow a Service Mesh control plane to install into your pods
A Service Mesh will may come with a UI or configuration files to configure your envoy

Network Address Translation (NAT)

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

What is Network Address Translation (NAT)?
a method of mapping an IP address space into another by modifying network address information in the IP header of packets while they are in transit across a traffic routing device



Eth0 and Network Namespace

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

What is an Ethernet Device?

An Ethernet Device is a software and/or hardware technology that allows a server to communicate on a computer network. A **Network Interface Card (NIC)** are commonly used to establish a wired connection to a network.

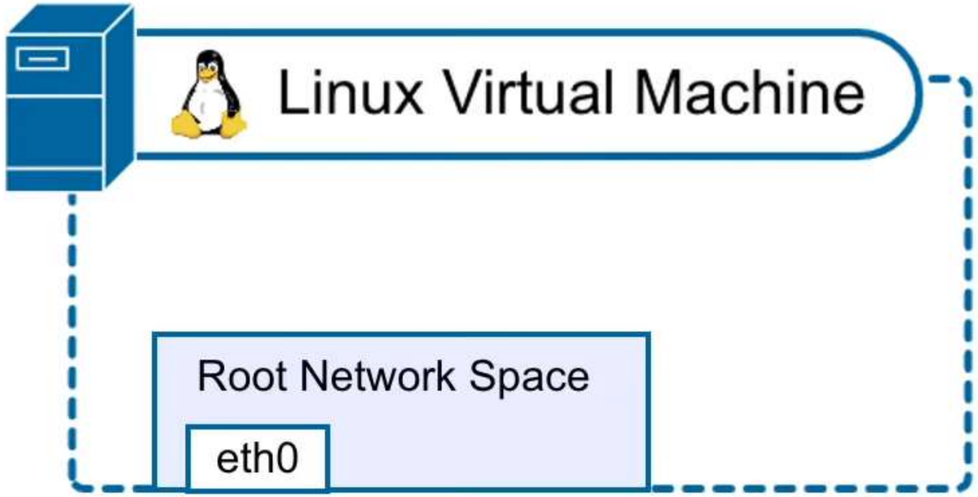
Cloud Service Providers (CSPs), have Virtual NICs for your Virtual Machines (VMs) to connect to the Virtual Network

What is a Network Namespace?

eth0 represents the first Ethernet interface attached to your Virtual Machine.

A network interface is an abstraction on top of the ethernet interface to provide a logical networking stack with its own routes, firewall rules, and network devices

Linux by default has one Network Namespace called the Root Network Space and this is what programs will use by default.



Eth0 and Network Namespace

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

To observe network devices attached to a linux virtual machine you use the **ifconfig** command.

Session ID: brown-laptop-0587bdce3842e9e35 Instance ID: i-07eba66923d9fa554

```
[ec2-user@ip-172-31-82-19 ~]$ ifconfig
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 9001
        inet 172.31.82.19  netmask 255.255.240.0  broadcast 172.31.95.255
        inet6 fe80::106f:4aff:fe1b:9b13  prefixlen 64  scopeid 0x20<link>
        ether 12:6f:4a:1b:9b:13  txqueuelen 1000  (Ethernet)
        RX packets 53021  bytes 72653845 (69.2 MiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 23977  bytes 2018753 (1.9 MiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1  netmask 255.0.0.0
        inet6 ::1  prefixlen 128  scopeid 0x10<host>
        loop  txqueuelen 1000  (Local Loopback)
        RX packets 24  bytes 1944 (1.8 KiB)
        RX errors 0  dropped 0  overruns 0  frame 0
        TX packets 24  bytes 1944 (1.8 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0

[ec2-user@ip-172-31-82-19 ~]$
```

To create modify or view Network Namespaces the **ip netns** command cand be used.

Session ID: brown-laptop-06df3855e35907f62 Instance ID: i-07e

```
[ec2-user@ip-172-31-82-19 ~]$ sudo ip netns add ns1
[ec2-user@ip-172-31-82-19 ~]$ ip netns
ns1
[ec2-user@ip-172-31-82-19 ~]$
```

Cluster Networking

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

Kubernetes has the following opinions about cluster networking:

- all Pods can communicate with all other Pods without using NAT
- all Nodes can communicate with all Pods without using NAT.
- the IP that a Pod sees itself as, is the same IP that others see it as

NATs are and can be used in Kubernetes, even though of the above contradiction

There are 4 broad types of network communication for clusters:

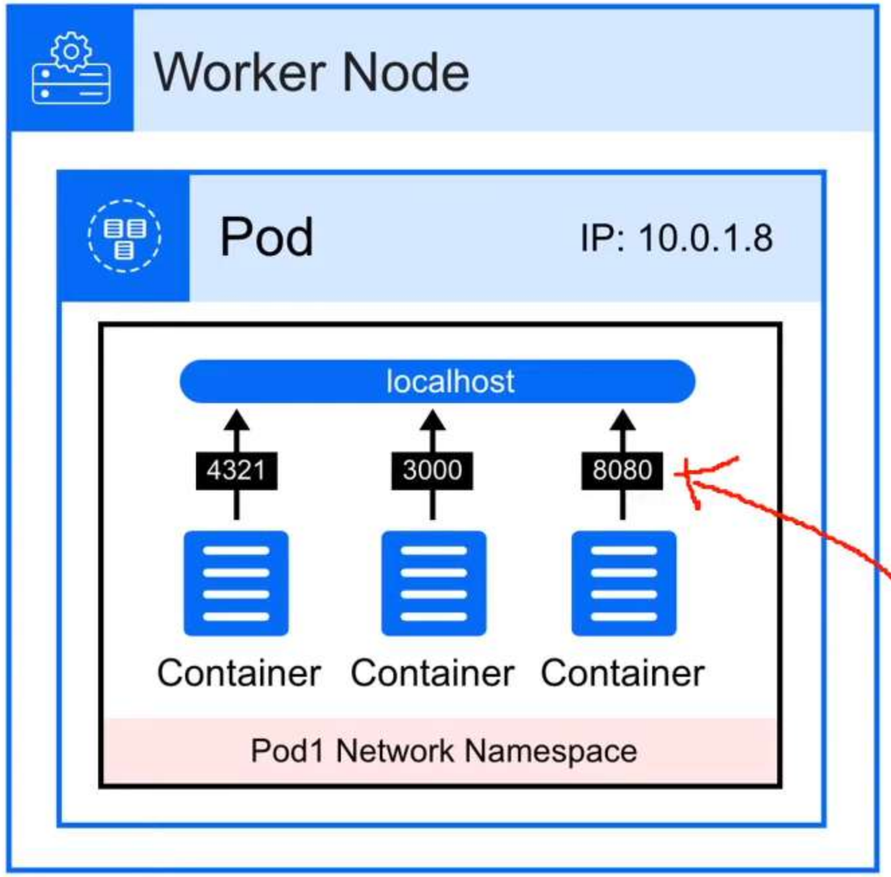
1. Container-to-Container
2. Pod-to-Pod
3. Pod-to-Service
4. External-to-Service

Watch the CNCF tech talk **Life of a Packet** by Michael Rubin for detailed technical information about Cluster Networking

Container to Container Networking

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

Containers all in the same pod have the same IP address and port space.
Containers can communicate with each other via **localhost** via different ports




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

Within the manifest file **containerPort** is used to define the local port for the container

Virtual Ethernet Devices (Veths)

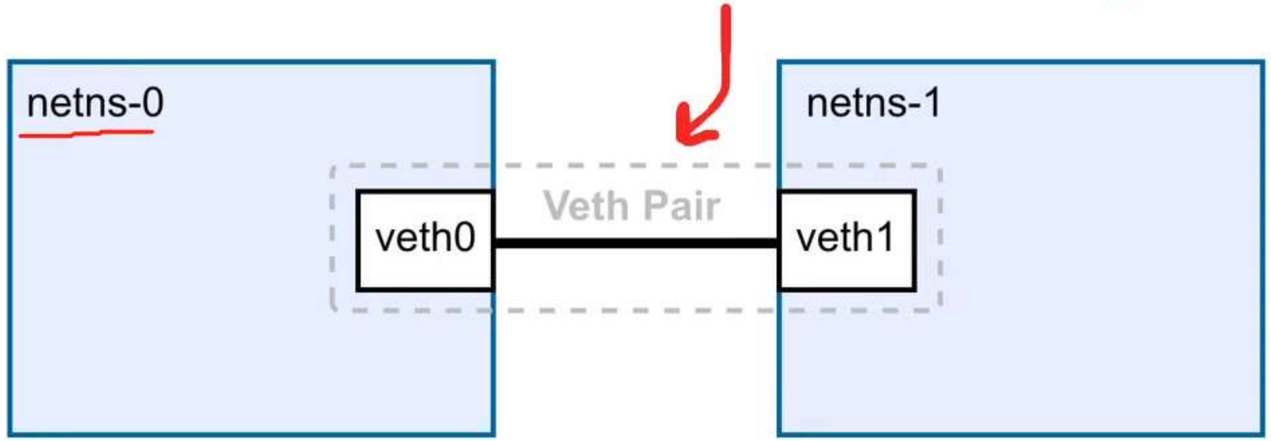
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

veth devices are Virtual Ethernet devices.

They can act as tunnels between network namespaces to create a bridge to a physical network device in another namespace, but can also be used as standalone network devices.

Packets on one device in the pair are immediately received on the other device

veth devices are always created in **interconnected pairs**



You use the **ip link** command to create veth pairs

```
ip link add <p1-name> netns <p1-ns> type veth peer <p2-name> netns <p2-ns>
```

Pod to Pod Same Node Networking

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

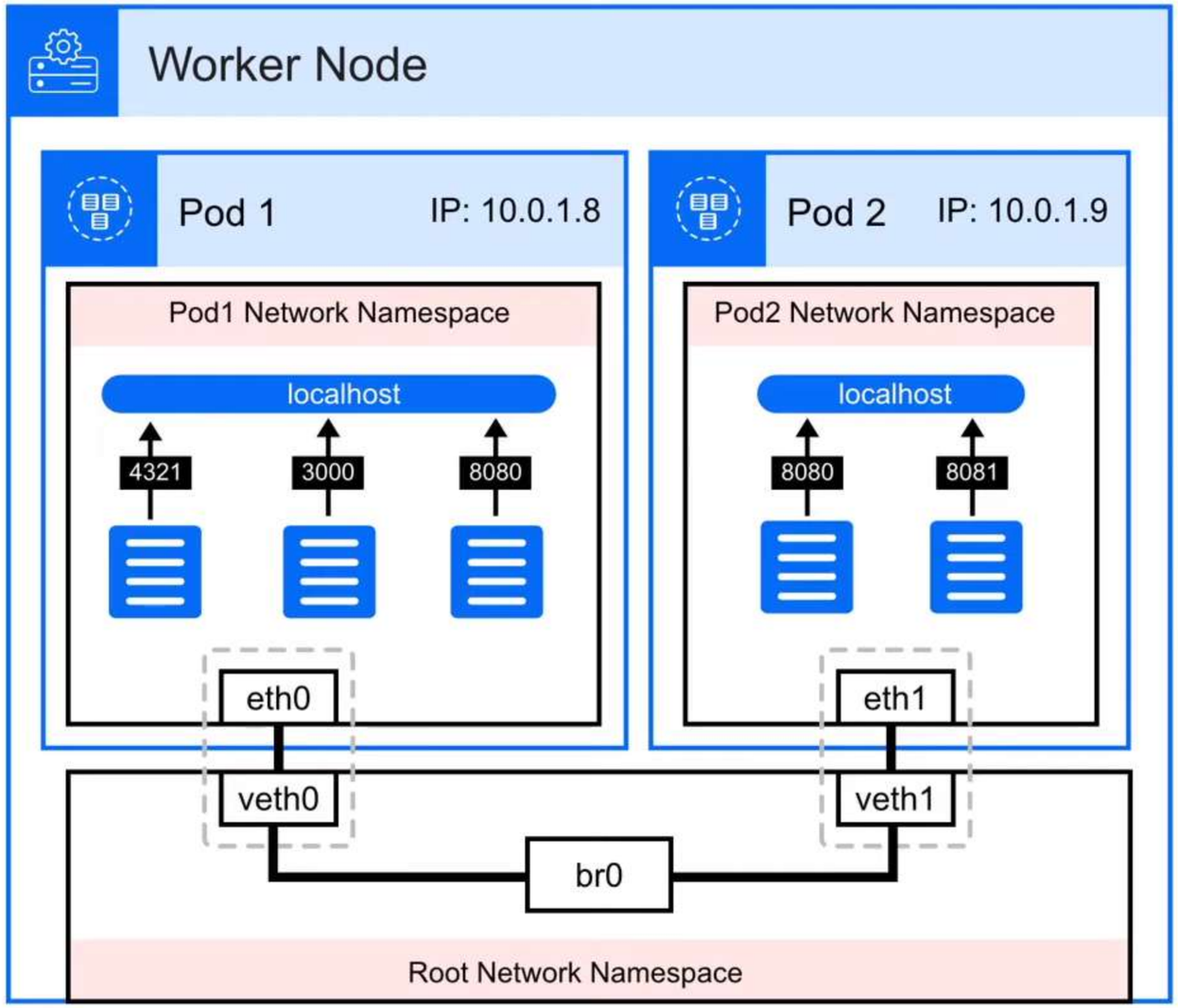
For Pod to Pod communication on the same on the node Veth is used to communication from the Pod Network Namespace to the Root Network Namespace.

In the Root Network Namespace a **bridge** is used allow all Pod Network Namespaces to talk to other pods.

Pods can see all other pods, and communicate using their IP addresses.

Routing allows multiple networks to communicate independently and yet remain separate using a Router

Bridging connects two separate networks as if they were a single network using a Bridge



Pod to Pod Across Node Networking

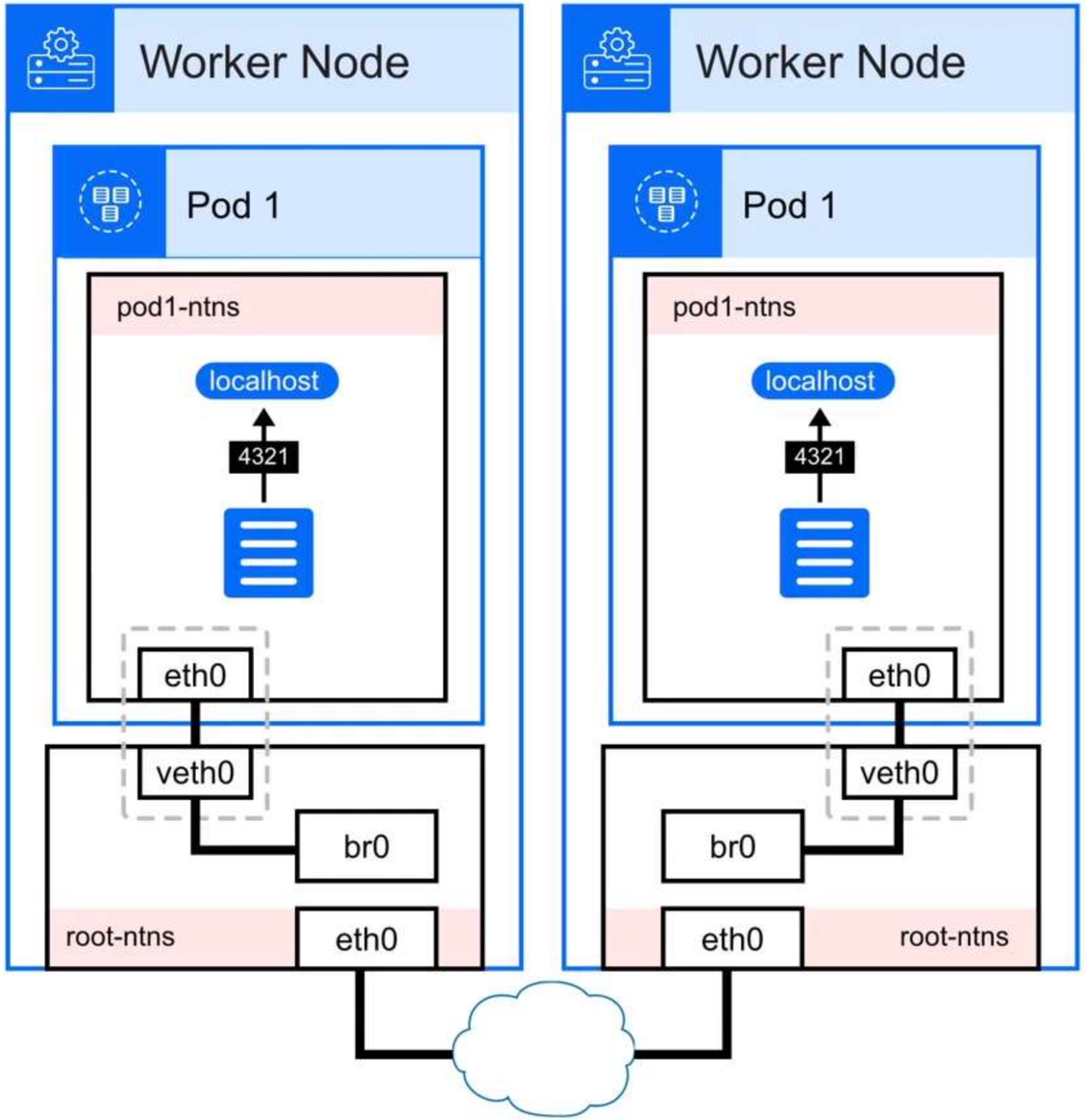
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

Pods can communicate to other Pods running on other worker nodes

How pods can communicate pods on other nodes is network specific to the scenario and will vary.

In the case of AWS, they have their own implementation of the Container Networking Interface (CNI)

Amazon VPC Container Network Interface (CNI) plugin for Kubernetes managed node to node communicate leverage AWS Virtual Private Cloud (VPC)

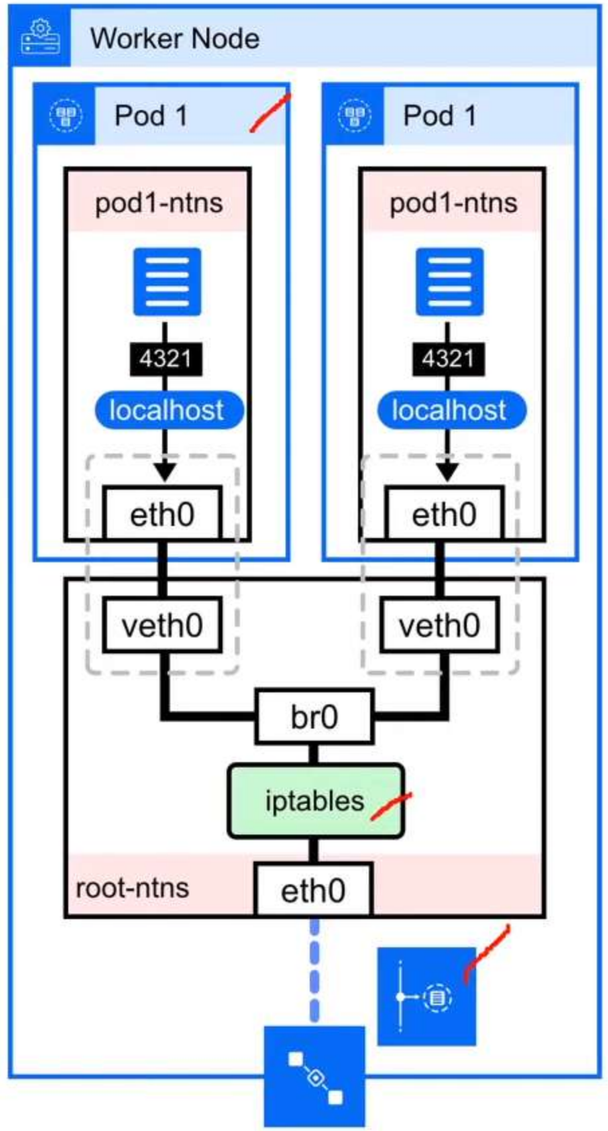


Pods to Service Networking

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

When a pod dies its IP Address changes and this can make communication hard if you are relying on the IP address for communication.

A Service creates a virtualized IP (static IP) and then uses iptables which is installed on the Node to Network Address Translation (NAT) and Load Balancing to other pods.



Ingress/Egress From Internet to Cluster

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

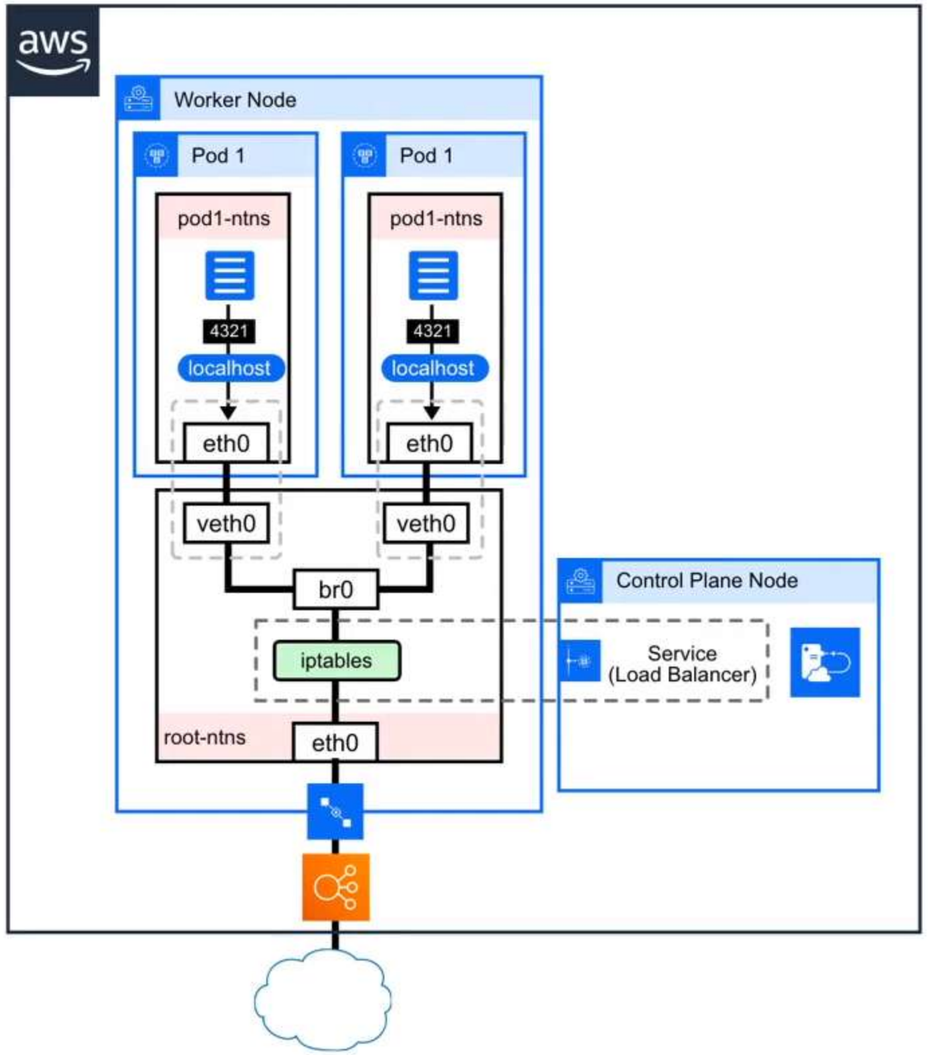
Egress

How pod traffic exits to the internet will be network specific. So in the case of AWS pods use the Amazon VPC Container Network Interface (CNI) plugin to be able to talk to your Virtual Private Cloud (VPC) and then egress out to the Internet Gateway (IGW) via route tables.

Ingres

For traffic to reach a pod, it travels to a service.
From there a Service could be using:

- K8s Service with Type Load Balancer
  - This will work with Cloud Controller Manager to implement a solution that works with a T4 (UDP/TCP) Load Balancer
- K8s Ingress
  - It will use a Ingress Controller to work with a Cloud Service Provider load balancer eg. T4 or T7 (application load balancer)

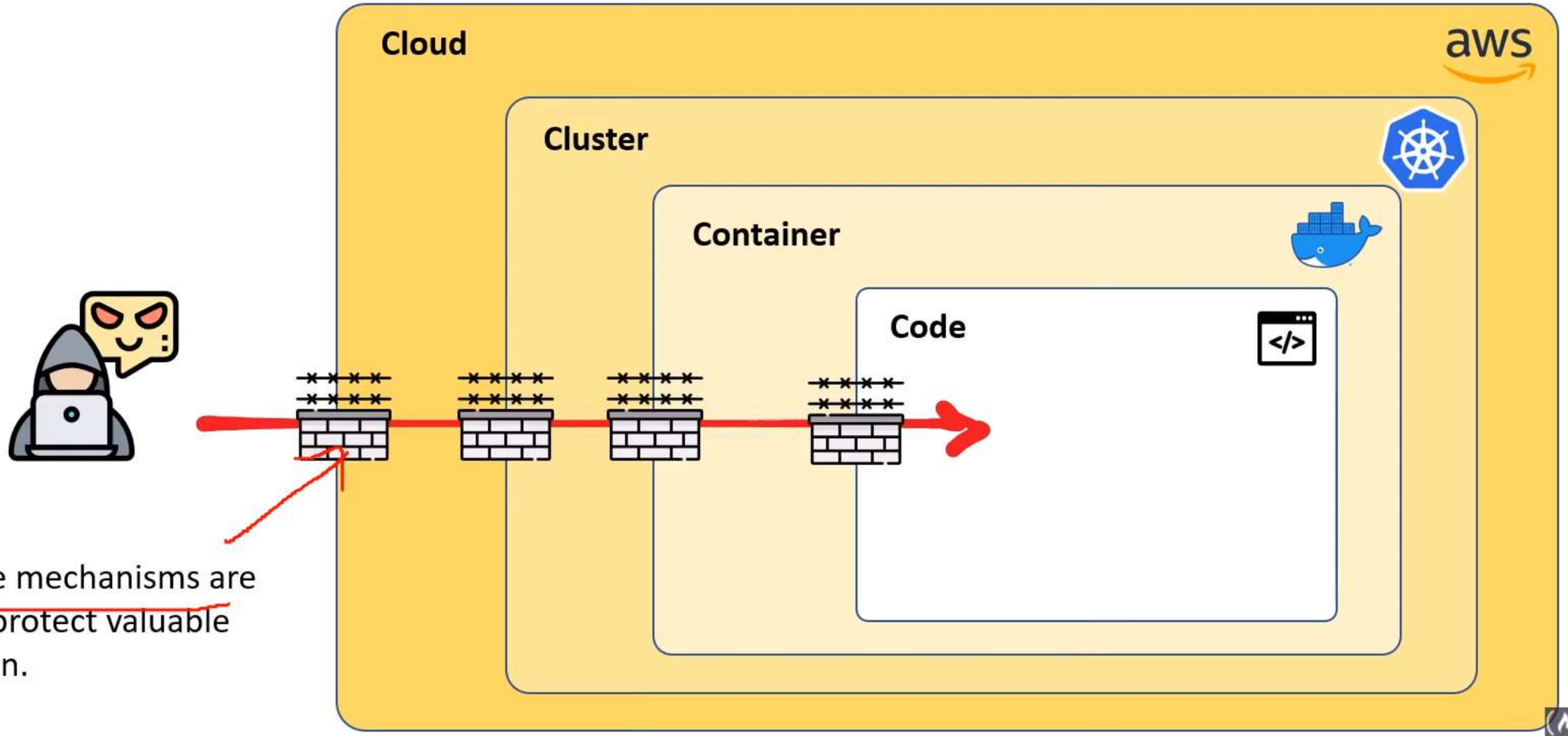


4Cs of Cloud Native security

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

The 4C's of Cloud Native security are **Cloud, Clusters, Containers, and Code**

Each layer of the Cloud Native security model **builds upon the next outermost layer**



Depth in Defense

a series of defensive mechanisms are layered in order to protect valuable data and information.

4Cs of Cloud Native security – Cloud Layer

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

The Cloud Layer is also known as the **base layer.**

Security at this layer is going to vary based on **managed** or **self-managed infrastructure**

Managed Infrastructure (IaaS)

- Cloud Service Provider (CSP)
  - AWS, Azure, GCP
- Cloud Platform
  - CIVO

Security is going to greatly vary based on provider, and independent research will need be undertaken

Eg. AWS's K8s managed offering is called EKS. Research EKS security would need to performed

Self-Managed

- Co-Located Servers
  - Equinix
- Corporate Datacenters

Security is going to be based on <u>Infrastructure Security</u>

- Network access to API Server (Control plane)
- Network access to Nodes (nodes)
- Kubernetes access to Cloud Provider API
- Access to etcd
- etcd Encryption

4Cs of Cloud Native security – Cluster Layer

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

There are two parts to Cluster Layer security

1. Components of the cluster

Securing configurable cluster components

- Controlling access to the Kubernetes API
  - Use Transport Layer Security (TLS) for all API traffic
  - API Authentication
  - API Authorization
- Controlling access to the Kubelet
- Controlling the capabilities of a workload or user at runtime
  - Limiting resource usage on a cluster
  - Controlling what privileges containers run with
  - Preventing containers from loading unwanted kernel modules
  - Restricting network access
  - Restricting cloud metadata API access
  - Controlling which nodes pods may access
- Protecting cluster components from compromise
  - Restrict access to etcd
  - Enable audit logging
  - Restrict access to alpha or beta features
  - Rotate infrastructure credentials frequently
  - Review third party integrations before enabling them
  - Encrypt secrets at rest
  - Receiving alerts for security updates and reporting vulnerabilities

2. Components in the cluster

Securing the applications running within the cluster

- RBAC Authorization (Access to the Kubernetes API)
- Authentication
- Application secrets management (and encrypting them in etcd at rest)
- Ensuring that pods meet defined Pod Security Standards
- Quality of Service (and Cluster resource management)
- Network Policies
- TLS for Kubernetes Ingress

4Cs of Cloud Native security – Container and Code Layer

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

Container Layer

- Container Vulnerability Scanning and OS Dependency Security
- Image Signing and Enforcement
- Disallow privileged users
- Use container runtime with stronger isolation

Code Layer

Application code is one of the primary attack surfaces over which you have the most control

- Access over TLS only
- Limiting port ranges of communication
- 3rd Party Dependency Security
- Static Code Analysis
- Dynamic probing attacks

Infrastructure Security

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

Suggestions for securing your infrastructure in a Kubernetes cluster

Network access to API Server (Control plane)

All access to the Kubernetes control plane is not allowed publicly on the internet and is controlled by network access control lists restricted to the set of IP addresses needed to administer the cluster.

Network access to Nodes (nodes)

Nodes should be configured to *only* accept connections from the control plane on the specified ports, and accept connections for services in Kubernetes of type NodePort and LoadBalancer.

Kubernetes access to Cloud Provider API

Each cloud provider grant a different set of permissions to the Kubernetes control plane and nodes. It is best to provide the cluster with cloud provider access that follows the principle of least privilege for the resources it needs to administer.

Access to etcd

Access to etcd (the datastore of Kubernetes) should be limited to the control plane only. Depending on your configuration, you should attempt to use etcd over TLS. More information can be found in the etcd documentation.

etcd Encryption

Wherever possible it's a good practice to encrypt all storage at rest, and since etcd holds the state of the entire cluster (including Secrets) its disk should especially be encrypted at rest

AAA

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

Authentication, Authorization and Accounting (AAA) framework for Identity management systems.

Authentication — to identify

- Static passwords
- One-time password (OTP) — MFA/UFA
- Digital certificates (x.509)
- Basic Auth

Authorization — to get permission

- Role Based Access Controls (RBAC)

Accounting (auditing) —to log and audit trail

- Audit Policies
- Audit Backends (where the logs will be stored)

Role-based Access Controls

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

Role-based Access Controls (RBAC) is a way of defining permissions for identities based on a organizational role.

RBAC authorization uses the **rbac.authorization.k8s.io** API group to drive authorization decisions, allowing you to dynamically configure policies through the Kubernetes API.

To enable RBAC, start the API server with the **--authorization-mode** flag

```
kube-apiserver --authorization-mode=RBAC
```

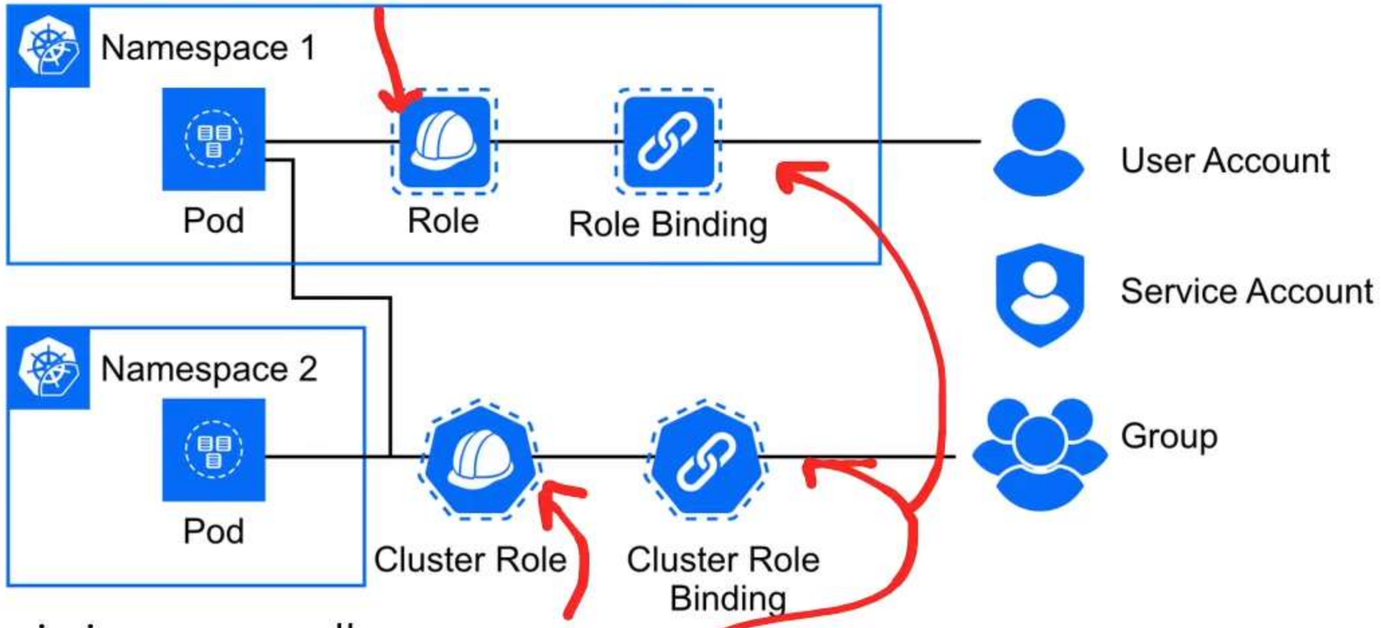
With Kubernetes RBAC there are only Allow Rules, Everything is Deny by default.

Role-based Access Controls Types

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

The RBAC API declares four kinds of Kubernetes object: Role and RoleBinding, ClusterRole and ClusterRoleBinding.

A **Role** is a set of permissions for a particular namespace



A **ClusterRole** is a set of permissions across all namespace

Role Binding and **Cluster Role Binding**, link Permissions to to **Subjects** (an Identity).

- User Account – A single user
- Service Account – Represents a machine user, to be used by an application service
- Group – A group of User Accounts and/or Service Accounts

Role Configuration Example

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

Example of a **Role configuration** in the default namespace

Kind:

- Role
- ClusterRole

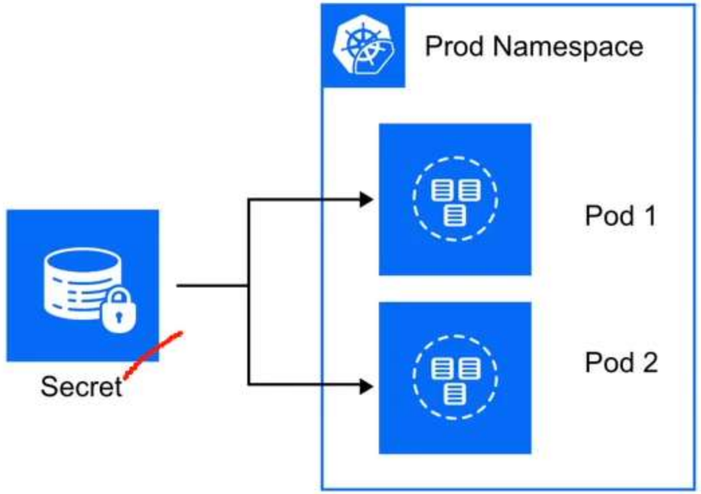
```yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
- apiGroups: [""] # "" indicates the core API group
  resources: ["pods"]
  verbs: ["get", "watch", "list"]
```

The rules (permissions) of **actions** this role is allowed to perform

Secrets Management

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

A Secret is similar to a ConfigMap with the exception that they can be encrypted




```yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-basic-auth
type: kubernetes.io/basic-auth
stringData:
  username: admin
  password: t0p-Secret
```

By default Secrets are unencrypted in etcd store.

- Anyone with access to the etcd store has access to the secrets
- Anyone who has access to a Pod within Namespace with have access to the Secrets used by that pod

How to keep Secrets safe:

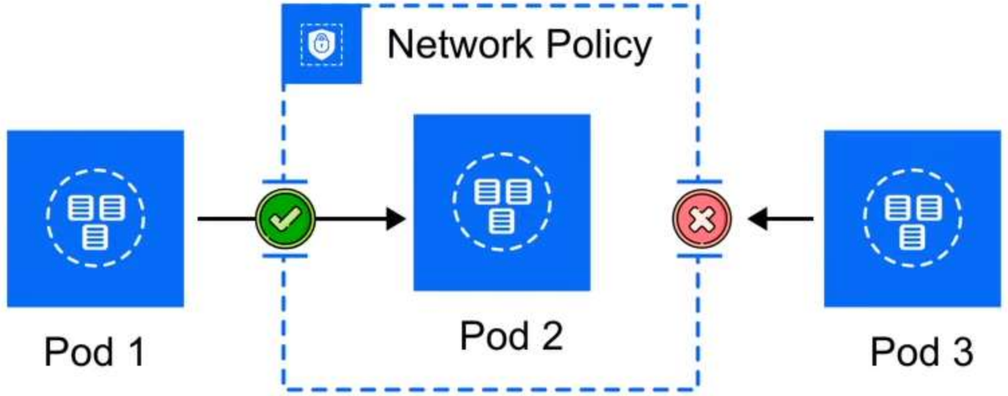
- Enable Encryption at Rest for Secrets
- Enable or configure RBAC rules that restrict reading data in Secrets
- Use mechanisms such as RBAC to limit which principals are allowed to create new Secrets or replace existing ones

Network Policy

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

Network Policies act as virtual firewalls for pod communication.
Pod communication can be restricted based on the follow scopes:

- Pod-to-pod
- Namespaces
- Specific IPs



Selectors are used to determine to select resources with matching labels for the Network Policy to be applied to

The Network Plugin you are using must support Network Policies
Eg. **Calico**, **Weave Net** , Cilium, Kube-router, Romana.

Calico

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

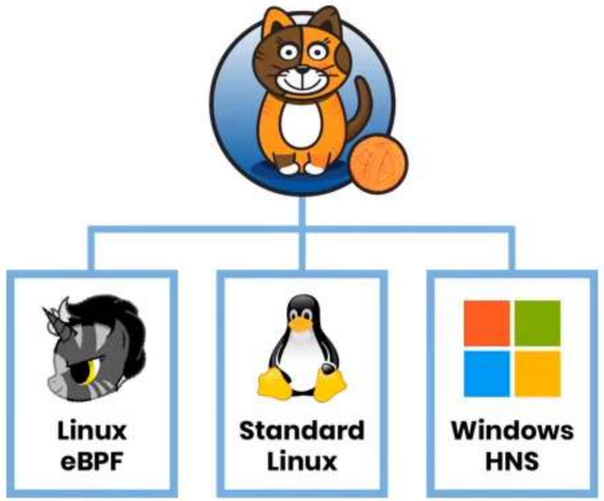
Calico is an **open-source network and network security solution** for **containers, VMs, native host-based workloads.**

Calico supports a broad range of platforms including:

- Kubernetes
- OpenShift
- Mirantis Kubernetes Engine (MKE)
- OpenStack
- Bare Metal Services

Calico gives you a choice of dataplanes:

- Linux eBPF
- Standard Linux
- Windows HNS



Calico Network Policies extends the base functionality of Network Policies:

- policies can be applied to any object
- the rules can contain the specific action
- you can use ports, port ranges, protocols, HTTP/ICMP attributes, IPs or subnets, any selectors as a source/target of the rules
- you can control traffic flows via DNAT settings and policies for traffic forwarding

Calico **can perform better** than alternative eg. Flannel, Cilium, WeaveNet

Anatomy of a Network Policy File

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

For a network policy you'll specify either a **Pod Selector** or **Namespace Selector**

You can define **Ingress rules** (permitted traffic entering a pod)

You can define **Egress rules** (permitted traffic to exit a pod)

```yaml
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: test-network-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      role: db
  policyTypes:
  - Ingress
  - Egress
  ingress:
  - from:
    - ipBlock:
        cidr: 172.17.0.0/16
        except:
        - 172.17.1.0/24
    - namespaceSelector:
        matchLabels:
          project: myproject
    - podSelector:
        matchLabels:
          role: frontend
    ports:
    - protocol: TCP
      port: 6379
  egress:
  - to:
    - ipBlock:
        cidr: 10.0.0.0/24
    ports:
    - protocol: TCP
      port: 5978
```

In-Transit vs At-Rest Encryption

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

Encryption In-Transit
Data that is secure when moving between locations
Algorithms: **TLS, SSL**

Encryption At-Rest
Data that is secure when residing on storage or within a database
Algorithms: **AES, RSA**

Transport Layer Security (TLS)
An encryption protocol for data integrity between two or more communicating computer application.
TLS 1.0, 1.1 are deprecated. **TLS 1.2** and **1.3** is the current best practice

Secure Sockets Layers (SSL)
An encryption protocol for data integrity between two or more communicating computer application
SSL 1.0, 2.0 and 3.0 *are deprecated*

Certificates and TLS

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

Kubernetes provides a **certificates.k8s.io** API, which lets you provision TLS certificates signed by a Certificate Authority (CA) that you control. These CA and certificates can be used by your workloads to establish trust.

What is Public key infrastructure (PKI)?
PKI is a set of roles, policies, hardware, software and procedures needed to create, manage, distribute, use, store and revoke digital certificates and manage public-key encryption.

What is a x.509 certificate?
A standard defined by the International Telecommunication Union (ITU) for **public key certifications.**
X.509 certificates are used in many Internet protocol:

- SSL/TLS and HTTPS
- Signed and encrypted email
- Code Signing and Document Signing

A certificate contains

- An identity — hostname, organization or individual
- A public key — RSA, DSA, ECDA etc…

Certificates and TLS

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

Kubernetes requires PKI for the following operations:

- Client certificates for the kubelet to authenticate to the API server
- Server certificate for the API server endpoint
- Client certificates for administrators of the cluster to authenticate to the API server
- Client certificates for the API server to talk to the kubelets
- Client certificate for the API server to talk to etcd
- Client certificate/kubeconfig for the controller manager to talk to the API server
- Client certificate/kubeconfig for the scheduler to talk to the API server.
- Client and server certificates for the front-proxy

most certificates are stored in **/etc/kubernetes/pki**

- Lightweight distributions store in different locations

K8 Security Best Practices

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

Best Practices of securing Kubernetes according to Aqua Security

1. Enable Kubernetes Role-Based Access Control (RBAC)
2. Use Third-Party Authentication for API Server
3. Protect etcd with TLS, Firewall and Encryption
4. Isolate Kubernetes Nodes
5. Monitor Network Traffic to Limit Communications
6. Use Process Whitelisting
7. Turn on Audit Logging
8. Keep Kubernetes Version Up to Date
9. Lock Down Kubelet

Autoscaling

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

What is Autoscaling?

In computing; autoscaling is when systems without manual intervention adjust capacity (eg. amount of CPU, Ram) to meet the demand (traffic from users) by adding or removing resources commonly triggered by events.

Pod-Based Scaling

- **Horizontal Pod Scaling** (*HorizontalPodAutoscaler*)
  Add more pods to meet the demand
- **Vertical Pod Scaling** (*VerticalPodAutoscaler*)
  Right-size pods for the optimal CPU and memory resources

Node-based scaling

Cluster Auto Scaling *(Cluster Autoscaler or Karpenter)*
Add or remove Nodes (compute servers) based on demand

Cluster API

Declarative APIs and tooling to simplify provisioning, upgrading, and operating multiple Kubernetes clusters. Cluster API can be extended to support any infrastructure (AWS, Azure, vSphere, etc.), bootstrap or control plane (kubeadm is built-in) provider.

KubeCTL Scale vs Autoscale

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

The **scale** command it used to:

- update the amount of replicas in the state of deployment object
- perform a deploy

```
kubectl scale --replicas=3 deploy/my-app
```

The **autoscale** command is used to create a **HorizontalPodAutoscaler**

```
kubectl autoscale rc foo --min=1 --max=5 --cpu-percent=80
```

KEDA

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

Kubernetes Event-driven Autoscaling (KEDA) allows you scale based on event data.

KEDA has a wide range of scalers

- ActiveMQ
- Apache Kafka
- AWS CloudWatch
- AWS Kinesis Stream
- AWS SQS Queue
- Azure Application Insights
- Azure Blob Storage
- Azure Event Hubs
- Azure Log Analytics
- Azure Monitor
- Azure Pipelines
- Azure Service Bus
- Azure Storage Queue
- Cassandra
- CPU
- Cron
- Datadog
- Elasticsearch
- External
- External Push
- GCP Pub/Sub
- Graphite
- Hauwei Cloudeye
- IBM MQ
- InfluxDB
- Kubernetes Workload
- Liklus Topic
- Memory
- Metrics API
- MongoDB
- MSSQL
- MySQL
- NATS Streaming

New Relic
OpenStack Metric
OpenStack Swift
PostgreSQL
Predictkube
Prometheus
RabbitMQ Queue
Redis Lists
Redis Streams
Selenium Grid Scalars
Solace PubSub + Event Broker

Open Standards

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

What is an Open Standard?
When multiple organizations or technologies adopt a specific technical standard. A standard is considered open when a public-facing community can participate on the development, maintenance and future changes to a standard.

Open Container Initiative
define industry standards around container image formats and runtime to make sure that all container runtimes could run images produced by any build tool,

CNI (Container Networking Interface)
a specification and libraries for writing plugins to configure network interfaces in Linux containers

CRI (Container Runtime Interface)
plugin interface which enables kubelet to use a wide variety of container runtimes, without the need to recompile.
CRI-O is an implementation of the Kubernetes CRI to enable using OCI compatible runtimes.

CSI (Container Storage Interface)
a standard for exposing arbitrary block and file storage storage systems to containerized workloads on Container Orchestration Systems (COs) like Kubernetes

Open Telemetry
A collection of tools, APIs, and SDK, to instrument, generate, collect, and export telemetry data (metrics, logs, and traces)

CNCF Governance Structure

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

The CNCF is composed of three main bodies:

Governing Board (GB)

- responsible for marketing
- other business oversight
- budget decisions

Marketing Committee

Technical Oversight Committee (TOC)

- Defining and maintaining technical vision

Special Interest Groups (SIGs)

End User Community (EUC)

- Providing feedback from companies and startups to improve cloud-native ecosystem

End User SIGs

User Groups

Memberships are composed of: Platinum, Gold, Silver, End User, and Academic / Non-Profit members

CNCF Memberships

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

Platinum members:

- Appoint one (1) representative to the CNCF Governing Board.
- Appoint one (1) representative as a voting member in any subcommittees or activities of the Governing Board.
- Enjoy most prominent placement in displays of membership including on the website.

Gold members:

- Appoint one (1) representative to the CNCF Governing Board per every five (5) Gold members, up to three (3) maximum Gold representatives

Silver members:

- Appoint one (1) representative to the CNCF Governing Board per every ten (10) Silver members, up to three (3) maximum Silver representatives.

End User members:

- Participate in the End User Advisory Community as described below
- Nominate one (1) representative to the End User Technical Advisory Board

Academic and Non-Profit members:

- participation is limited to academic and non-profit institutions respectively and requires approval by the Governing Board
- entitled to identify their organization as members supporting the mission of CNCF and any other rights or benefits as determined by the Governing Board

CNCF Memberships

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

Platinum members:
$370,000 USD at 3-Year minimum contract

Gold members:
Up to $120,000 USD annually

Silver members:

- $7,000 - Under 50 employees
- $10,000 - 50 – 99 employees
- $15,000 - 100 – 499 employees
- $25,000 - 500 – 999 employees
- $35,000 - 1,000 – 2,999 employees
- $45,000 - 3,000 – 4,999 employees
- $50,000 - 5,000 employees+

Academic and Non-Profit members:

- $1000 / $500 annually

End User members:

- No Cost

CNCF Governing Board (GB) - Responsibilities

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

- approve a budget directing the use of funds raised from all sources of revenue to be used for technical, marketing or community investments that advance the mission of the CNCF
- elect a Chair of the Governing Board to preside over meetings, authorize expenditures approved by the budget and manage any day-to-day operations
- vote on decisions or matters before the Governing Board
- define and enforce policy regarding intellectual property (copyright, patent or trademark) of the foundation
- direct marketing and evangelism efforts through events, press and analyst outreach, web, social and other marketing efforts
- oversee operations and qualification efforts
- establish and oversee any committees created to drive the mission of CNCF
- establish and execute a brand compliance program, if any, based on the CNCF requirements, which may include a certification test, to use the brand marks established by the TOC
- adopt guidelines or a policy for use of the trademark
- provide financial governance overall

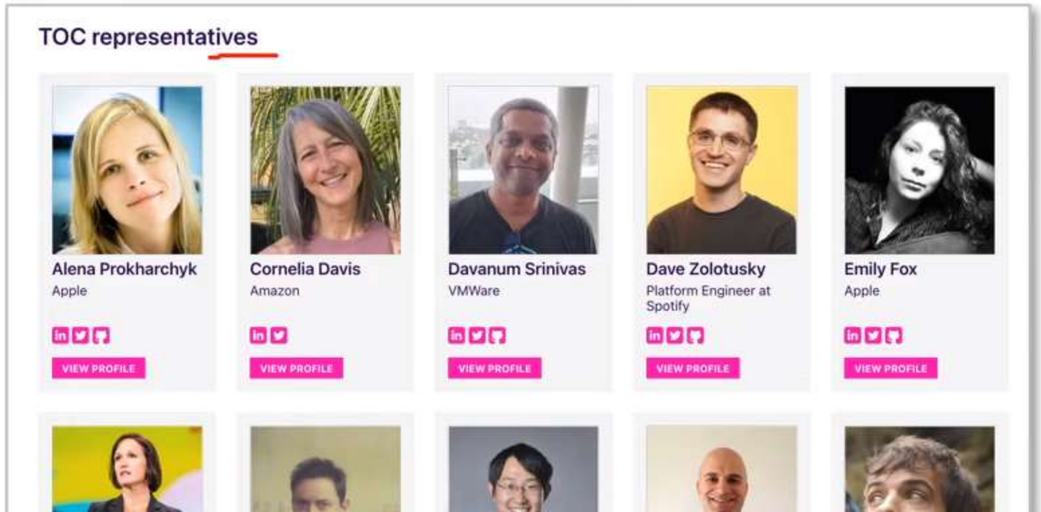
CNCF Technical Oversight Committee (TOC)

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

Technical Oversight Committee (TOC) provides technical leadership to the cloud native community

There responsibilities include:

- defining and maintaining the technical vision for the Cloud Native Computing Foundation,
- approving new projects and creating a conceptual architecture for the projects,
- aligning projects and removing or archiving projects,
- accepting feedback from end user committee and mapping to projects,
- aligning interfaces to components under management (code reference implementations before standardizing), and



Special Interest Groups (SIGs)

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

Special Interest Groups (SIGs) are **specialized committees** that work under (report to) the TOC.

SIGs include around:

- Traffic
- Observability
- Governance
- App Delivery
- Core and Applied Architectures
- Security

SIGs are long-lived groups.

led by recognised and relevant experts

- Strengthen the project ecosystem to meet the needs of end users and project contributors.
- Identify gaps in the CNCF project portfolio. Find and attract projects to fill these gaps.
- Educate and inform users with unbiased, effective, and practically useful information.
- Focus attention & resources on helping foster project maturity, systematically across CNCF projects.
- Clarify relationship between projects, CNCF project staff, and community volunteers.
- Engage more communities and create an on-ramp to effective TOC contribution & recognition.
- Reduce some project workload on TOC while retaining executive control & tonal integrity with this elected body.
- Avoid creating a platform for politics between vendors.

CNCF End User Community (EUC)

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

CNCF End Users are:

- Individual or organizations **that use cloud-native technologies**
- But they **do not sell** cloud-native services

Examples of NOT End Users

- Vendors
- Consultancies
- training partners
- telcos

The EUC acts as a feedback loop between the those using cloud-native and those maintaining and building cloud-native solutions.

End User Community is an eco-system of:

- Finding cloud-native talent
- Finding local User Groups
- Meet project maintainers
- Contribute to the CNCF Technology Radar

End User Technology Radar

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

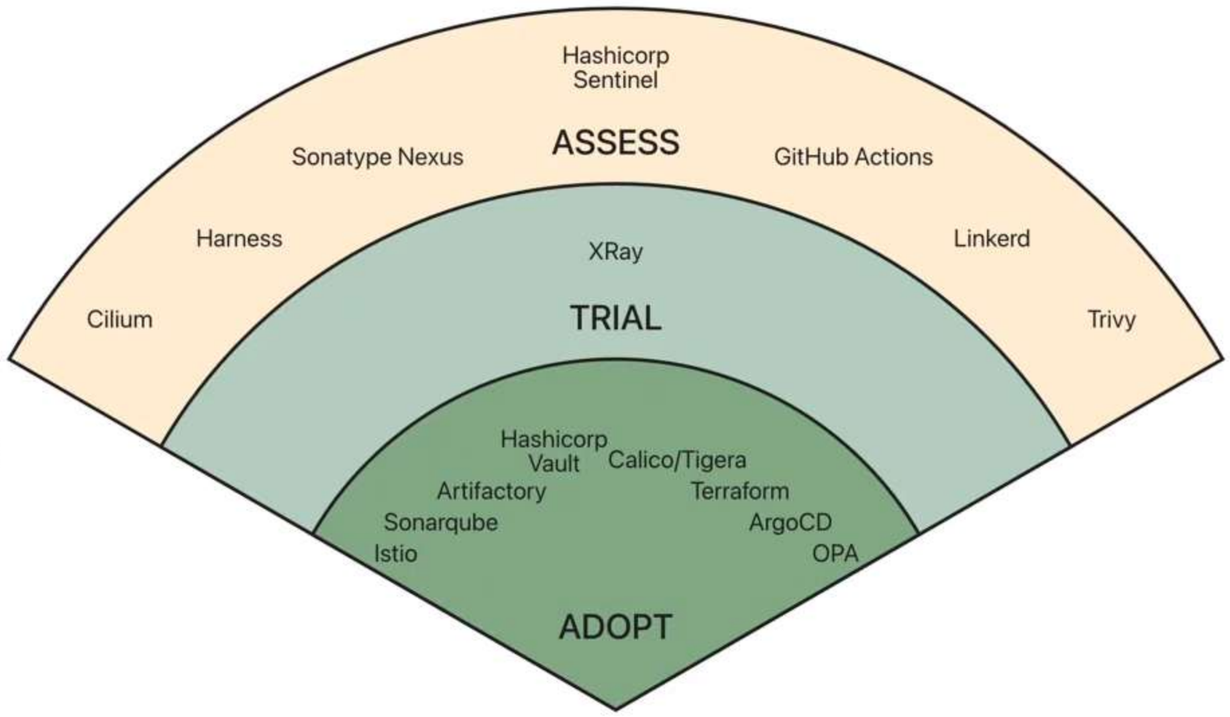
A technology radar is an **opinionated guide to a set of emerging technologies**.

The CNCF End User Technology Radar is **intended for a technical audience** who want to understand what solutions end users use in cloud native, and which they recommend

Assess
The CNCF End User Community has tried it out, and we find it promising. We recommend having a look at these items when you face a specific need for the technology in your project.

Trial
The CNCF End User Community has used it with success, and we recommend you have a closer look at the technology.

Adopt
The CNCF End User Community can clearly recommend this technology. We have used it for long periods of time in many teams, and it has proven to be stable and useful.



CNCF Charter

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

The CNCF Charter is **a public document** that contains:

- The organization structure of the CNCF
- The mission and values of the CNCF
- The description of the membership tiers
- The definition of CNCF projects
- CNCF governing Policies
- Reference to Codes of Conduct
- General definitions in relation to the CNCF
- General Rules

It’s a big document to help define and guide the CNCF.

CNCF Values

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

The CNCF will strive to adhere to the following principles

Fast is better than slow	enables projects to progress at high velocity to support aggressive adoption by users.
Open	Open and accessible, and operates independently of specific partisan interests. Accepts all contributors based on the merit of their contributions. Technology must be available to all according to open source values. The technical community and its decisions shall be transparent.
Fair	avoid undue influence, bad behavior or “pay-to-play” decision-making.
Strong technical identity	achieve and maintain a high degree of its own technical identify shared across the projects
Clear boundaries	establish clear goals and what the non-goals to allow projects to effectively co-exist. help the ecosystem understand where to focus for new innovation.
Scalable	support all scales of deployment, from startups, to service providers to enterprises.
Platform Agnostic	The specifications developed will not be platform specific such that they can be implemented on a variety of architectures and operating systems.

KubeCon + CloudNativeCon

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

KubeCon

KubeCon is technology conference for **Kubernetes**

CloudNativeCon is a technology conference for **Cloud Native**

CloudNativeCon

KubeCon and CloudNativeCon are essentially a single conference now commonly referred to as:

- KubeCon + CloudNativeCon

The conference happens twice a year:

- KubeCon + CloudNativeCon (North America)
- KubeCon + CloudNativeCon (Europe)

2018: 8,000 attendees (in-person)
2019: 12,000 attendees (in-person)
2020: 22,816+ attendees (virtual)
2021: 17,000 attendees (4K in-person)

IstioCon
ServiceMeshCon
GitOpsCon

Adjacent conferences at North America

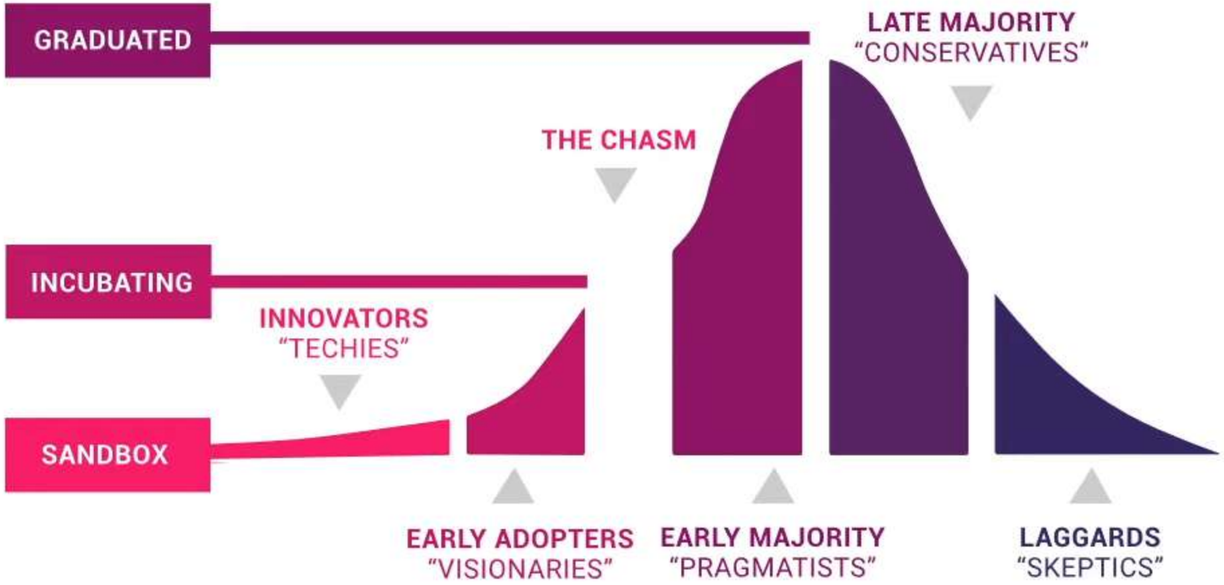
CNCF Projects

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

CNCF projects are technologies and managed by the CNCF.
Many CNCF projects are developed by external tech companies and then are handed over to the CNCF for long-term support

https://www.cncf.io/projects/

CNCF projects are categorized based on their "Maturity level" :



Graduated
Its production ready for enterprises

Incubating
The API might be rapidly changing. May have incomplete to be adopted by large enterprises.

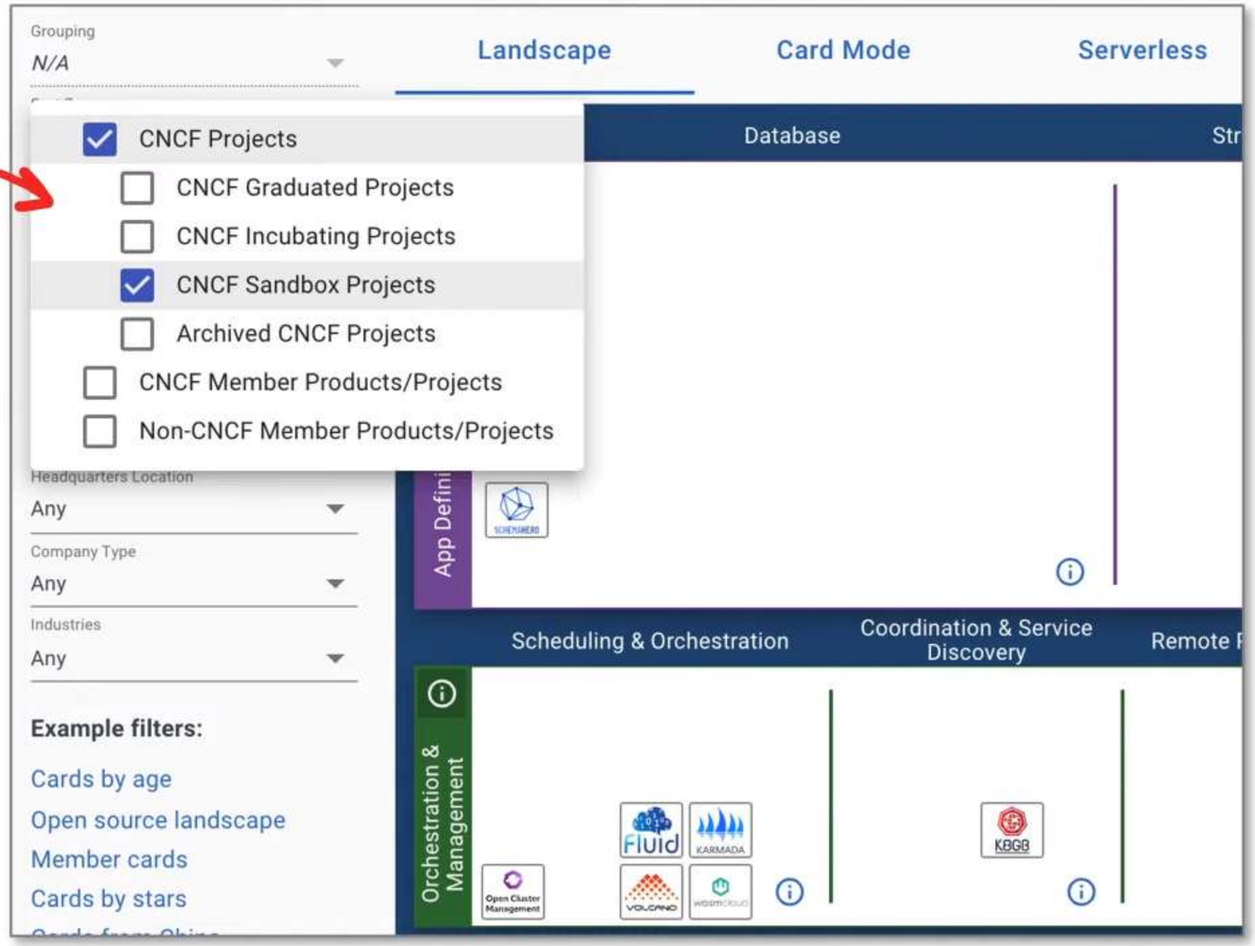
Sandbox
Experimental, prototypes, don't bet your security review on these projects. There are so many they are not listed on the project page.

The Maturity level is based on Crossing the Chasm diagram from the 1991 book **Crossing the Chasm**.

CNCF Projects - Sandboxes

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

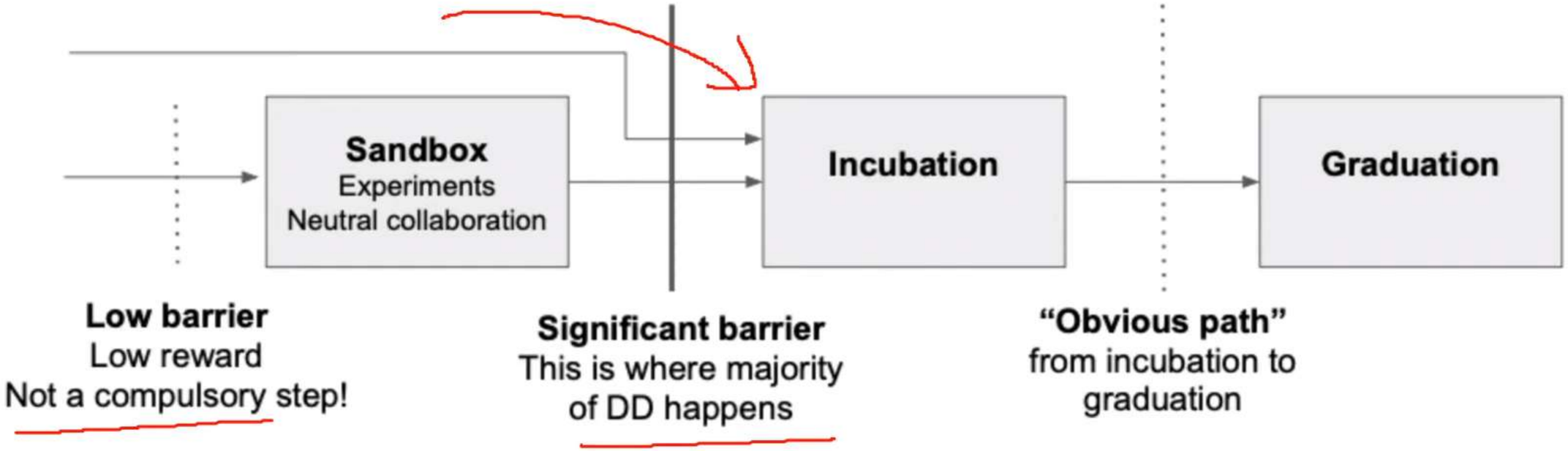
One way to locate Sandbox CNCF Projects is to filter based on **CNCF relation**



CNCF Projects

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

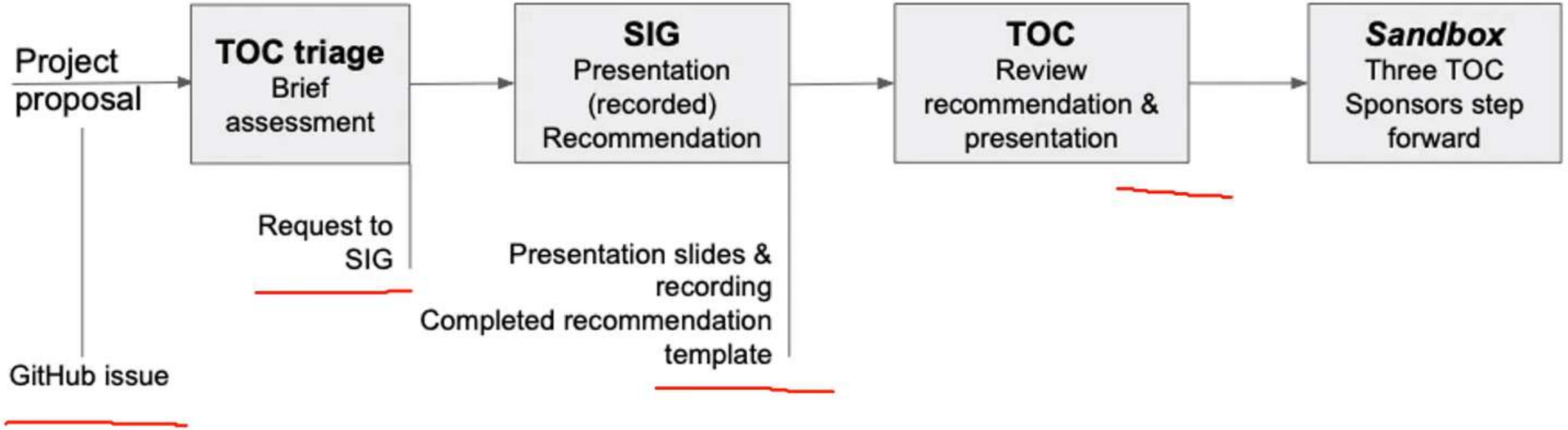
Project Stages



CNCF Projects

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

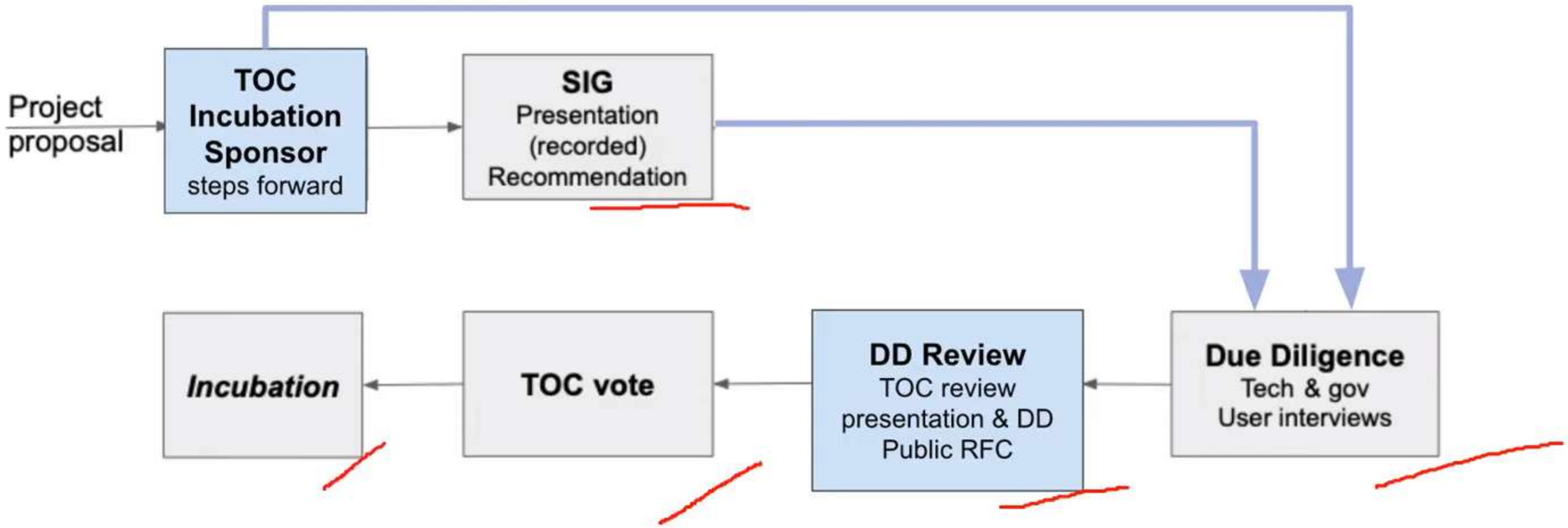
Sandbox Process



CNCF Projects

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

Incubation Process



CNCF Projects

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

There is at least one Graduated CNCF project generally
for each broad Cloud-Native category

Container Runtime
ContainerD

Coordination & Service Discovery
CoreDNS

Service Proxy
Envoy

Coordination & Service Discovery
etcd

Logging
Fluentd

Container Registry
Harbor

Application Definition & Image Build
Helm

Tracing
Jaeger

Scheduling & Orchestration
Kubernetes

Service Mesh
Linkerd

Security & Compliance
Open Policy Agent (OPA)

Monitoring
Prometheus

Security & Compliance
The Update Framework (TUF)

Cloud Native Storage
Rook

Database
TiKV

Database
Vitess

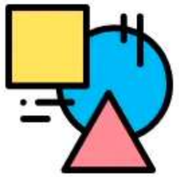
What is Serverless?

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

What is Serverless?
Serverless architecture generally describes fully managed cloud services.
The classification of a cloud service being serverless is not a Boolean answer (yes or no), but an answer on a scale where a cloud service has a degree of serverless.

A serverless service could have all or most of the following characteristics:

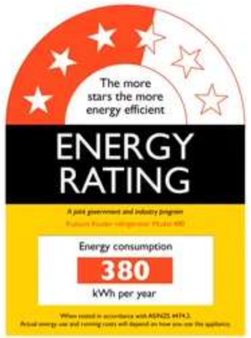
- Highly elastic and scalable
- highly available
- Highly durable
- Secure by default



Abstracts away the underlying infrastructure and are billed based on the execution of your business task.

Serverless can **Scale-to-Zero** meaning when not in use the serverless resources cost nothing.

Pay-for-Value (you don't pay for idle servers).

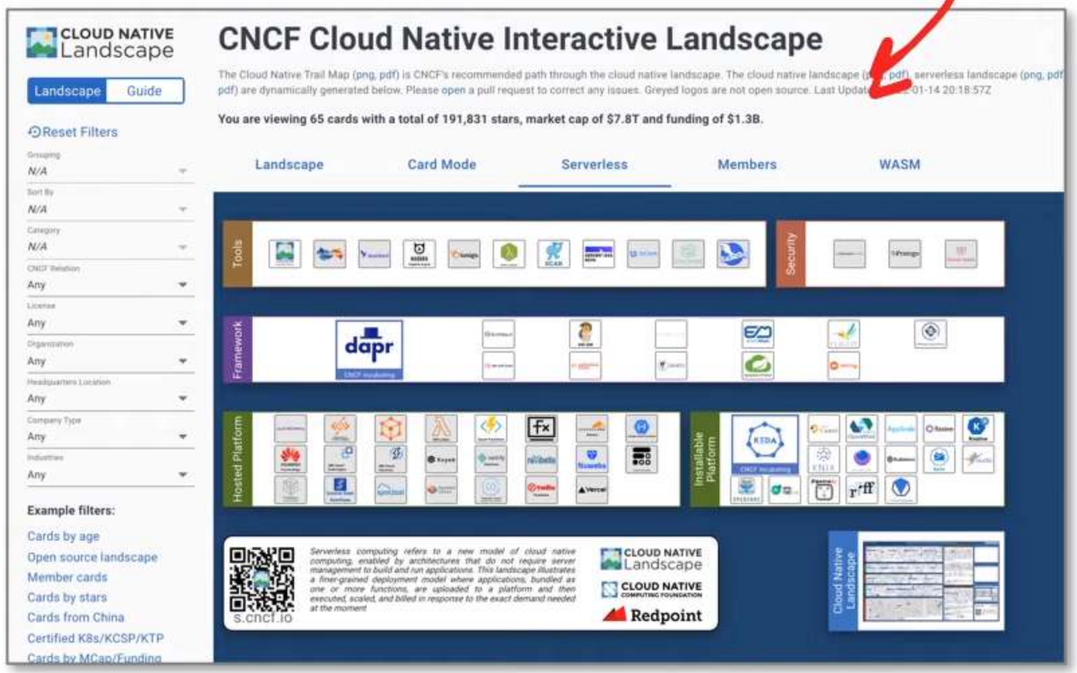


An analogy of serverless could be similar to an energy rating labels which allows consumers to compare the energy efficiency of a product. Some services are more serverless than others.

Cloud Native and K8s Serverless

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

CNCF has a landscape just for **serverless**:



What CNCF classifies as *"serverless"* are:

Function as a Service: eg.

- AWS Lambda
- Azure Serverless Functions
- Google Cloud Functions

Serverless Frameworks: eg.

- Dapr
- AWS SAM
- Chalice

Installable Platforms: eg

- Kubernetes-based Event Driven Autoscaling (KEDA)
- Apache OpenWisk
- OpenFaaS
- Knative
- Fission
- Kubeless

Tools: eg

- Lumigo
- Dashbird

Knative

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

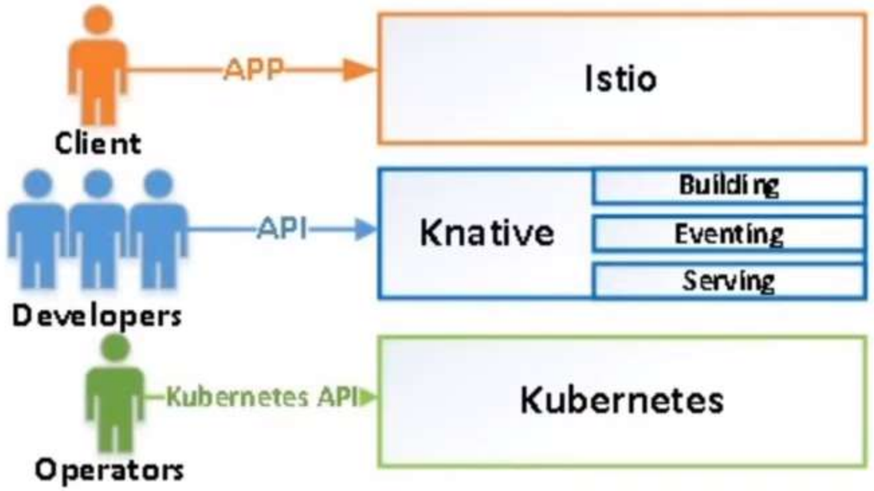
Knative
Kubernetes-based platform to **deploy and manage modern serverless workloads**

Knative is a project to create a standard set of building blocks for Kubernetes to enable serverless development patterns.

Knative generally is composed of two parts:

Knative Serving
- take containerized code and deploy it with relative ease.
- Scale to zero costs

Knative Eventing
- Trigger serverless functions based on Kubernetes API events
- Loop in other event sources to trigger serverless functions



"Considerations"
- It's not a complete serverless framework
- It does not offer a Function as Service (FaaS) offering

Knative

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

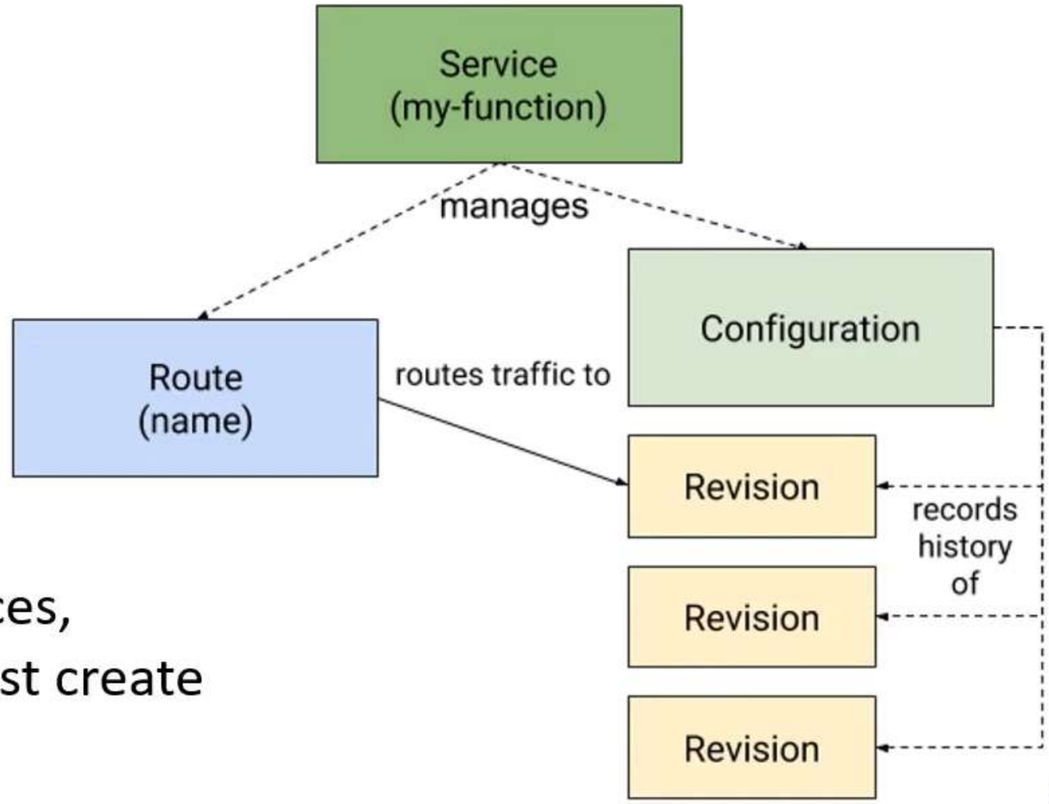
Knative defines its own set of Kubernetes Objects as **Kubernetes Custom Resource Definitions (CRDs).**

```
kn service create hello \
--image gcr.io/knative-samples/helloworld-go \
--port 8080 \
--env TARGET=World \
--revision-name=world
```

Knative has its own CLI called **kn**, used alongside Kubectl.

Knative components:

- **Service** — Manage lifecycle of a workload
- **Route** — Mapping network endpoints
- **Configuration** — Maintains desired state
- **Revision** — Point-in-time snapshots of code



Think of Knative of abstracting away deployments, services, autoscaling and many more K8s components, and you just create and deploy Kn Services.

Knative vs OpenFaaS

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

Unlike OpenFaas, Knative is not a full-fledged serverless platform, but it is better positioned as a platform for creating, deploying, and managing serverless workloads.

However, from the point of view of configuration and maintenance, OpenFaas is simpler. With OpenFaas, there is no need to install all components separately as with Knative, and you don't have to clear previous settings and resources for new developments if the required components have already been installed.

Still, as mentioned above, a significant drawback of OpenFaaS is that the container launch time depends on the provider, while Knative is not tied to any single cloud solution provider.

Based on the pros and cons of both, organizations may also choose to use Knative and OpenFaaS together to effectively achieve different goals.

The Pillars of Observability

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

What is Observability?
The ability to measure and understand how internal systems work in order to to answer questions regarding performance, tolerance, security and faults with a system / application.

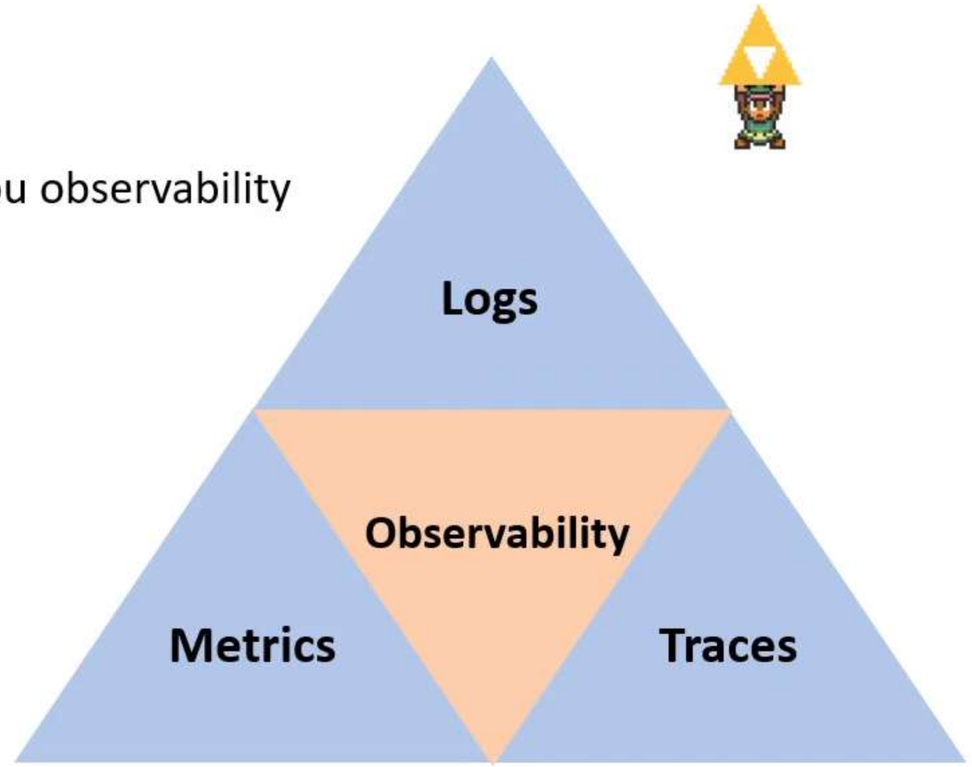
To obtain observability you need to use **Metrics**, **Logs** and **Traces**.
You have to use them together, using them in isolate does not gain you observability

Metrics
A number that is measured over period of time
eg. If we measured the CPU usage and aggerated it over an a period of time we could have an **Average CPU metric**

Logs
A text file where each line contains event data about what happened at a certain time.

Traces
A history of request that is travels through multiple Apps/services so we can pinpoint performance or failure.

Looks like they should have called it the Triforce of Observability



Open Telemetry

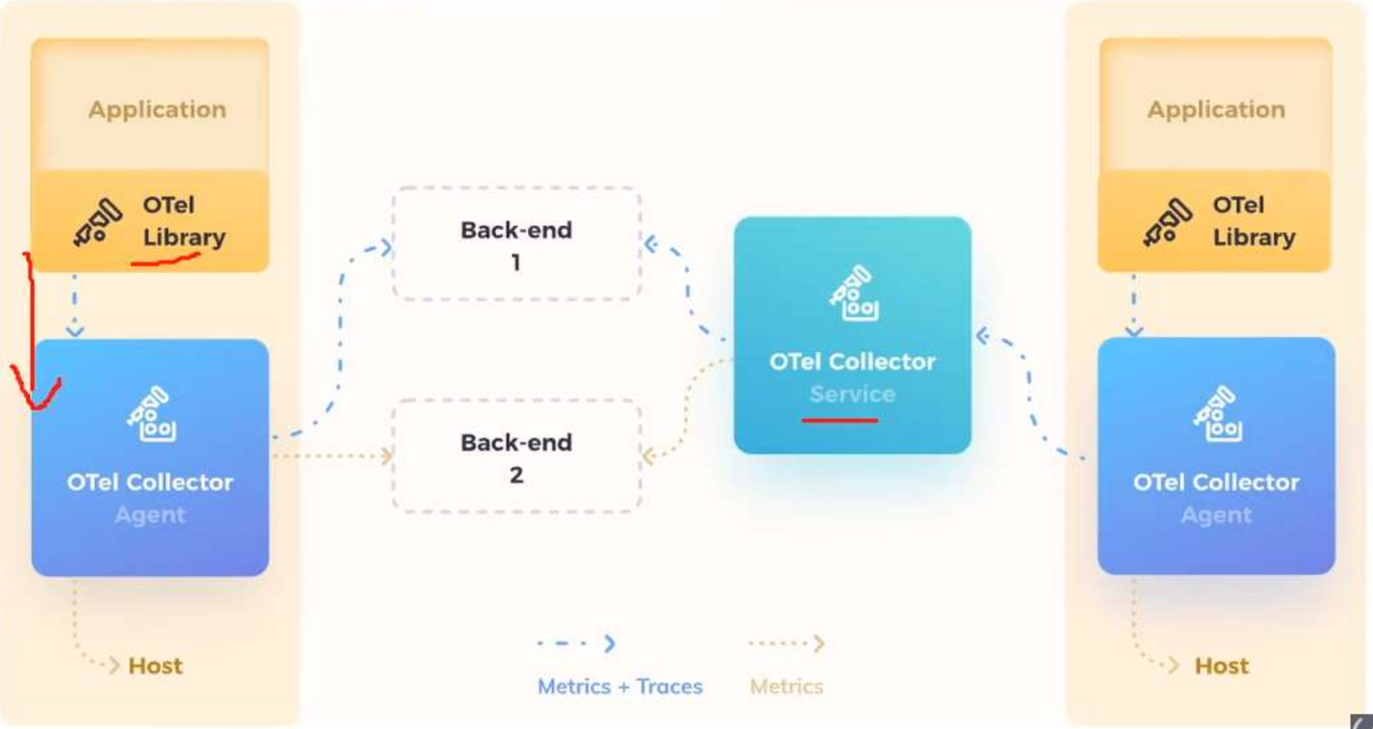
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

Open Telemetry (OTEL) is a collection of open-source tools, APIs and SDKs to instrument, generate, collect, and export telemetry data

Open Telemetry is **standardizes** the way telemetry data (metrics, logs and traces) are generated and collected.

Wire protocol

a wire protocol refers to a way of getting data from point to point. Eg. SOAP, AMQP



Open Telemetry - Instrumentation

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

Instrumentation is the act of embedding a monitoring library into your existing application in order to capture monitoring data such as: metrics, traces or logging

Open Telemetry supports a variety of languages:

- C++
- .NET
- Erlang / Elixir
- Go
- Java
- Javascript
- Php
- Python
- Ruby
- Rust
- Swift

For certain frameworks there plug-and-play libraries to quickly instrument your apps:

- Spring
- ASP.NET Core
- Express
- Quarkus

```ruby
# Require otel-ruby
require 'opentelemetry/sdk'

# Export traces to console by default
ENV['OTEL_TRACES_EXPORTER'] ||= 'console'

# configure SDK with defaults
OpenTelemetry::SDK.configure

@tracer = OpenTelemetry.tracer_provider.tracer('sinatra', '1.0')

OpenTelemetry::Context.with_current(context) do
  # Span kind MUST be `:server` for a HTTP server span
  @tracer.in_span(
    span_name,
    attributes: {
      'component' => 'http',
      'http.method' => env['REQUEST_METHOD'],
      'http.route' => env['PATH_INFO'],
      'http.url' => env['REQUEST_URI'],
    },
    kind: :server
  ) do |span|
    # Run application stack
    status, headers, response_body = @app.call(env)

    span.set_attribute('http.status_code', status)
  end
end
```

Open Telemetry - Collector

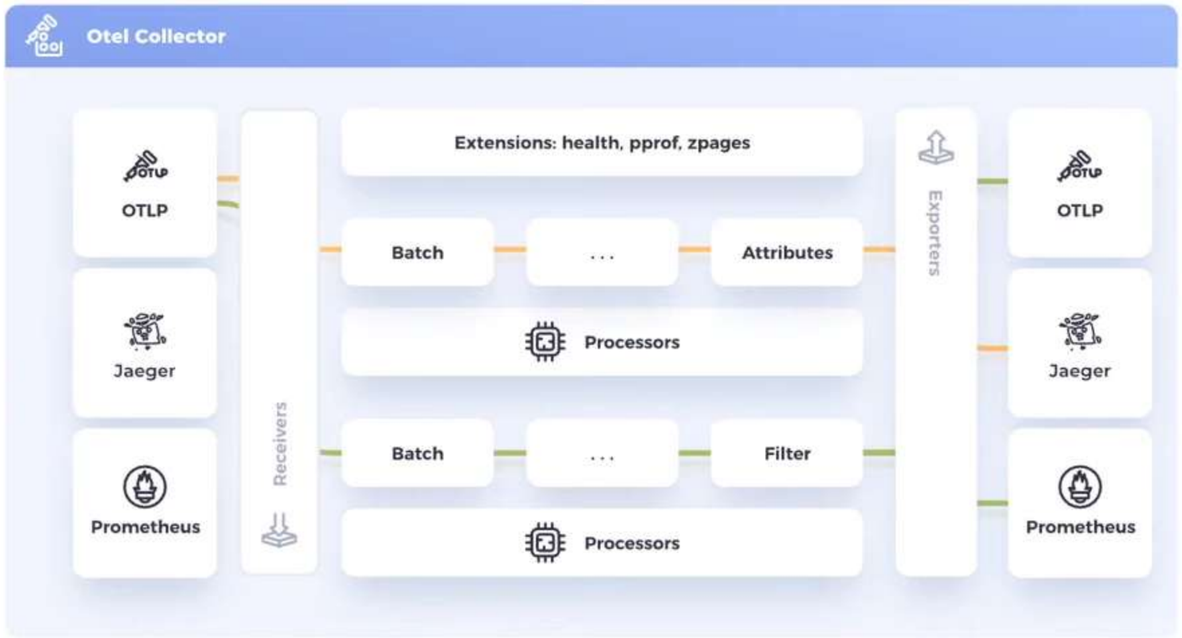
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

The Open Telemetry collector is an agent installed on the target machine, or as a dedicate server and is Vendor-agnostic way to receive, process and export telemetry data

It removes the need to run, operate, and maintain multiple agents/collectors.

This works with improved scalability and supports open-source observability data formats (e.g. Jaeger, Prometheus, Fluent Bit, etc.) sending to one or more open-source or commercial back-ends.

The local Collector agent is the default location to which instrumentation libraries export their telemetry data.



Prometheus

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

Prometheus is **an open-source systems monitoring and alerting toolkit** originally built at SoundCloud.
Prometheus collects and stores its metrics as time series data.
Prometheus is a **timeseries database**

Prometheus's main features are:

- a multi-dimensional data model with time series data identified by metric name and key/value pairs
- PromQL, a flexible query language to leverage this dimensionality
- no reliance on distributed storage; single server nodes are autonomous
- time series collection happens via a pull model over HTTP
- pushing time series is supported via an intermediary gateway
- targets are discovered via service discovery or static configuration
- multiple modes of graphing and dashboarding support

Prometheus values reliability.
You can always view what statistics are available about your system, even under failure conditions.

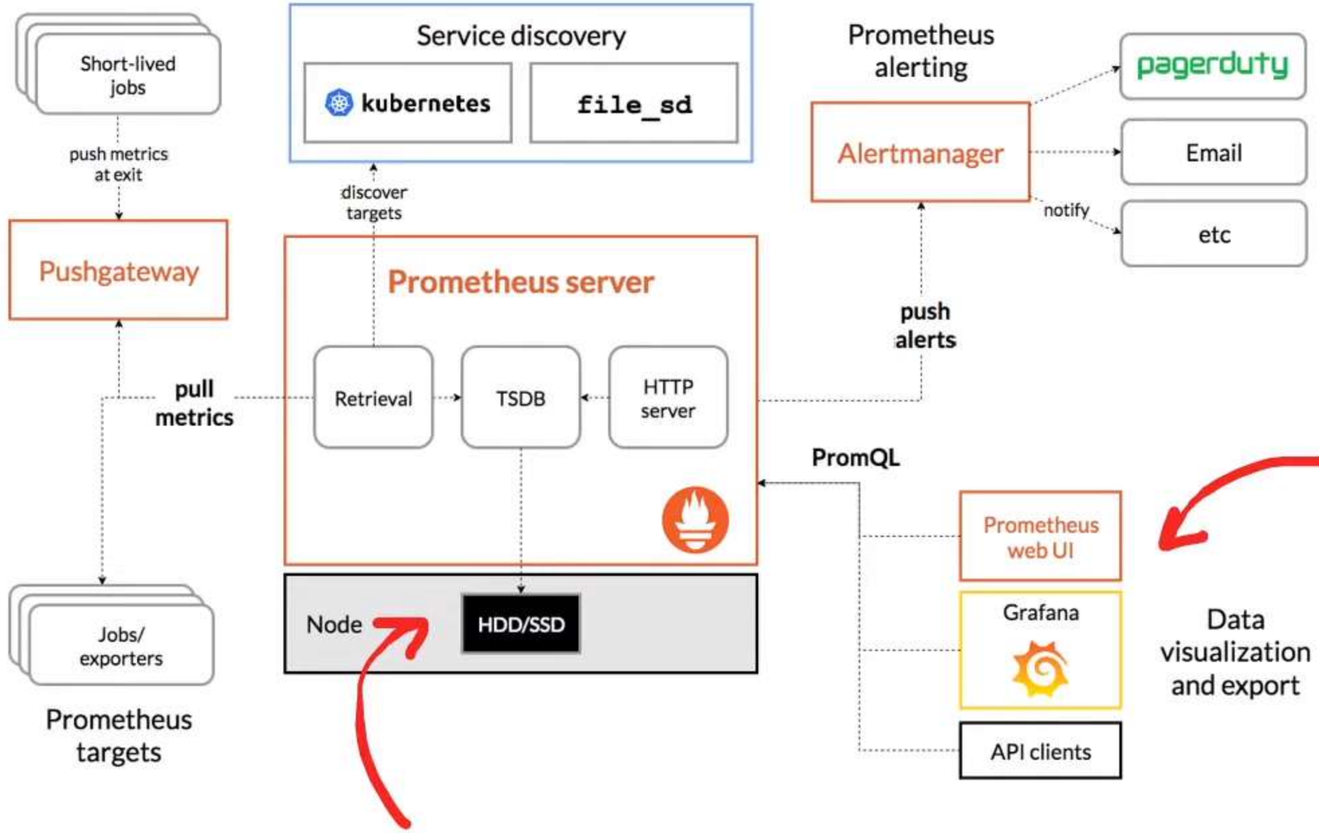
If you need 100% accuracy, such as for per-request billing, Prometheus is not a good choice as the collected data will likely not be detailed and complete enough.

In such a case you would be best off using some other system to collect and analyze the data for billing, and Prometheus for the rest of your monitoring.

Prometheus

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

Prometheus scrapes metrics from **instrumented jobs**, either directly or via an intermediary push gateway for short-lived jobs.



Grafana or other API consumers can be used to visualize

It stores all scraped samples locally and runs rules over this data to either aggregate and record new time series from existing data or generate alerts

Grafana

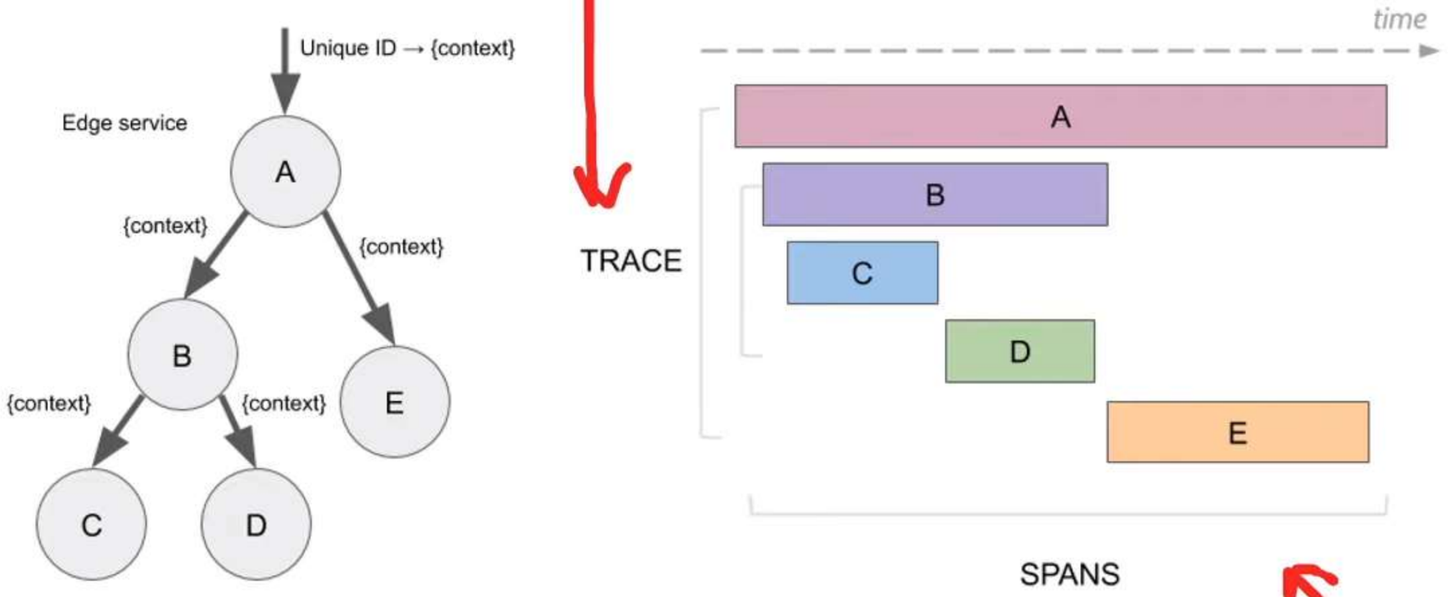
Cheat sheets, Practice Exams and Flash cards 👉 www.exampro.co/kcna

Grafana is an open source **analytics and interactive visualization** web application
Grafana is commonly used along with a timeseries database like: InfluxDB, Prometheus or Graphite

Traces and Spans

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

A **trace** is a data/execution path through the system,
and can be thought of as a **directed acyclic graph (DAG)** of spans



A **span** represents a logical unit of work in Jaeger that has an operation name,
the start time of the operation, and the duration.
Spans may be nested and ordered to model causal relationships.

Cost Management

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

- Label Resources
  - Visualize Costs Prometheus and Grafana
- Finding Idle And Unallocated Resources
  - Visualize Idle CPU, Memory and Storage Prometheus and Grafana
- Workload Rightsizing
  - VerticalPodAutoscaler — adjusting CPU, Mem of Pods
  - HorizontalPodAutoscaler — add remove pods to meet the demand
- Cluster Downsizing Opportunities
  - ClusterAutosscaler Add or remove Nodes to meet the demand
- Using Free Trials Of Kubernetes Cost Tools eg. Kubecost
- Estimate future costs, use a Load-Testing — eg. SpeedScale, K6, JMeter, Gatling
  - What does a 15 node EKS cluster cost for the year?
- Utilization waste (running but not using or not live)
  - Serverless Architect that Scales to Zero when no traffic for a period of time.
- Technical debt
  - Evaluate architecture to reduce amount of pods
  - Evaluate cloud-native technologies

Kubernetes System Logs and Klogs

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

System Logs

- System component logs record events happening in cluster, which can be very useful for debugging.
- You can configure log verbosity to see more or less detail.
- Logs can be as coarse-grained as showing errors within a component, or as fine-grained as showing step-by-step traces of events (like HTTP access logs, pod state changes, controller actions, or scheduler decisions).

In KubeCTL you can view logs from pods with the **logs** commands →

```
kubectl logs nginx --all-containers=true
```

Klog

- Klog is the Kubernetes logging library.
- klog generates log messages for the Kubernetes system components.

Chaos Testing and Testing

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

What is Testing?

Asserting the expectation of the input and outputs of functions.

```
require "test/unit/assertions"
include Test::Unit::Assertions

hello = 'world'

assert_equal 'world', hello, "hello function should return 'world'"
```

What is Chaos Testing?

Building a system to withstand and tolerant any kind of failure by **purposely introducing random failures** in a **production system**.

chaoskube

ChaosKube

TestKube

ChaosMonkey

Helm Chart Directory Structure

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

A helm chart is a collection files within a **directory**

```
wordpress/
  Chart.yaml          # A YAML file containing information about the chart
  LICENSE             # OPTIONAL: A plain text file containing the license for the chart
  README.md           # OPTIONAL: A human-readable README file
  values.yaml         # The default configuration values for this chart
  values.schema.json  # OPTIONAL: A JSON Schema for imposing a structure on the values.yaml file
  charts/             # A directory containing any charts upon which this chart depends.
  crds/               # Custom Resource Definitions
  templates/          # A directory of templates that, when combined with values
                      # will generate valid Kubernetes manifest files.
  templates/NOTES.txt # OPTIONAL: A plain text file containing short usage notes
```

Helm reserves use of the charts/, crds/, and templates/ directories, and of the listed file names. Other files will be left as they are.

Helm Chart File Structure

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

The Helm Chart describes the contents of the package such as the type of chart, dependencies

```yaml
annotations:
  category: CMS
apiVersion: v2
appVersion: 5.8.3
dependencies:
  - condition: mariadb.enabled
    name: mariadb
    repository: https://charts.bitnami.com/bitnami
    version: 9.x.x
  - condition: memcached.enabled
    name: memcached
    repository: https://charts.bitnami.com/bitnami
    version: 5.x.x
  - name: common
    repository: https://charts.bitnami.com/bitnami
    tags:
      - bitnami-common
    version: 1.x.x
description: Web publishing platform for building blogs and websites.
engine: gotpl
home: https://github.com/bitnami/charts/tree/master/bitnami/wordpress
icon: https://bitnami.com/assets/stacks/wordpress/img/wordpress-stack-220x234.png
keywords:
  - blog
  - cms
  - php
  - wordpress
maintainers:
  - email: containers@bitnami.com
    name: Bitnami
name: wordpress
sources:
  - https://github.com/bitnami/bitnami-docker-wordpress
  - https://wordpress.org/
version: 12.3.3
```

Helm Packaging and Installing

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

To package a chart directory in a versioned chart archive the “helm package” command is used.

```
helm package --sign ./mychart --key mykey --keyring ~/.gnupg/secring.gpg
```

Versioned chart archives are used by Helm package repositories

Artifact Hub is where we can find published Helm charts

https://artifacthub.io/

```
helm repo add nicholaswilde https://nicholaswilde.github.io/helm-charts/
helm repo update
helm install postgres nicholaswilde/postgres
```

Installing a helm chart

Kustomize

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

Kustomize provides more flexibility when writing Kubernetes configuration files by allowing you to overlay (override) to “patch” configurations.

Kustomize is built into KubeCTL

```
├── base
│   ├── kustomization.yaml
│   ├── deployment.yaml
│   └── service.yaml
└── patches
    ├── dev
    │   ├── kustomization.yaml
    │   ├── deployment.yaml
    │   └── namespace.yaml
    └── prod
        ├── kustomization.yaml
        ├── deployment.yaml
        └── namespace.yaml
```

Kustomization file defined what will be overwritten in the base components.

What is Infrastructure as Code?

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

The Problem with Manual Configuration

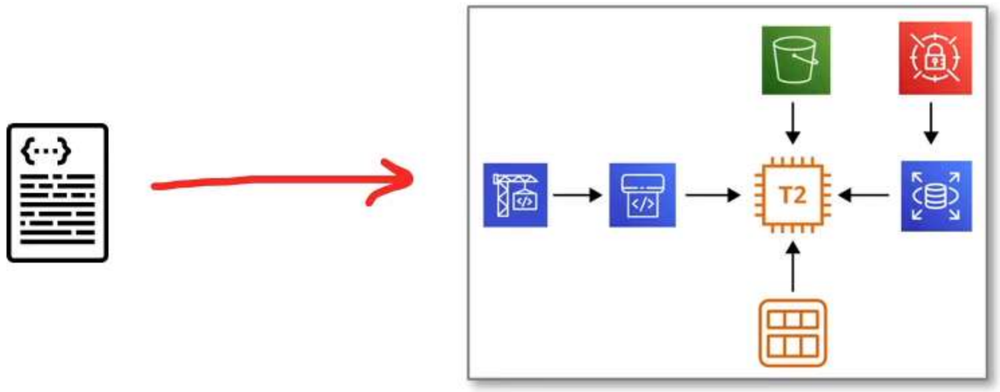
Manually configuring your cloud infrastructure allows you easily start using new service offerings to quickly prototype architectures however it comes with many downsides:

- Its easy to mis-configure a service though human error
- Its hard to manage the expected state of configuration for compliance
- Its hard to transfer configuration knowledge to other team members

Infrastructure as Code (IaC)

You write a configuration script to **automate** **creating, updating or destroying** cloud infrastructure.

- IaC is a **blueprint** of your infrastructure.
- IaC allows you to easily **share, version or inventory** your cloud infrastructure.



Popular Infrastructure as Code tools (IaC)

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

Declarative

- What you see is what you get. ***Explicit***
- More verbose, but zero chance of mis-configuration
- Uses scripting languages eg. JSON, YAML, XML

ARM Templates
Supports only Azure

Azure Blueprints
Supports only Azure
Manages relationship between services

CloudFormation
Only for AWS

Cloud Deployment Manager
Supports on Google Cloud

Terraform
Supports many cloud service providers (CSPs) and cloud services.

Imperative

- You say what you want, and the rest is filled in. ***Implict***
- Less verbose, you could end up with misconfiguration
- Does more than Declarative
- Uses programming languages eg. Python, Ruby, JavaScript

AWS Cloud Development Kit (CDK)
Supports only AWS
Many built-in templates for opinionated best practices

Pulumi
Supports AWS, Azure, GCP, K8

IaC for Kubernetes

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

Managing Infrastructure **of the Cluster**

Terraform (or any IaC) tool can be used to provision the cluster and managed services to be used alongside the cluster eg. Manage Database.

Terraform can technically manage cluster components via their Manifest module
So you can *benefit from the state management provided by Terraform.
Kubernetes already manages state with the Controller Manager and etcd

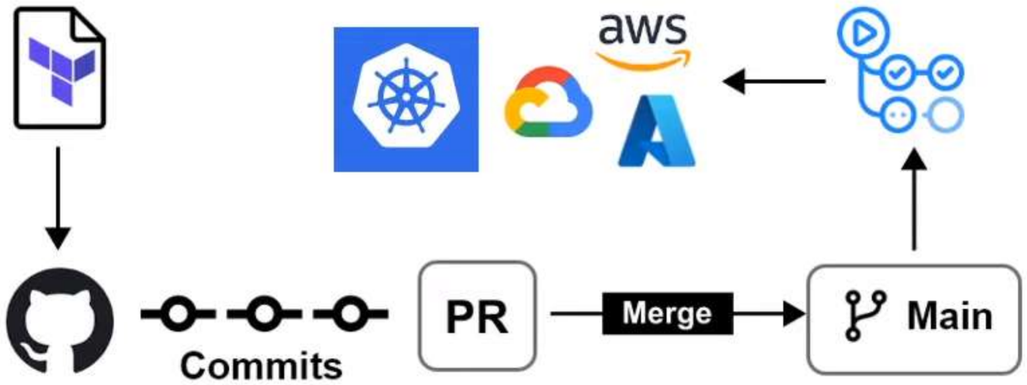
Managing Infrastructure **in the Cluster**

For managing (application) infrastructure within the cluster eg. Pods, Services, Ingres
It is recommended to package as Helm charts and use that in your CI/CD.

What is GitOps?

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

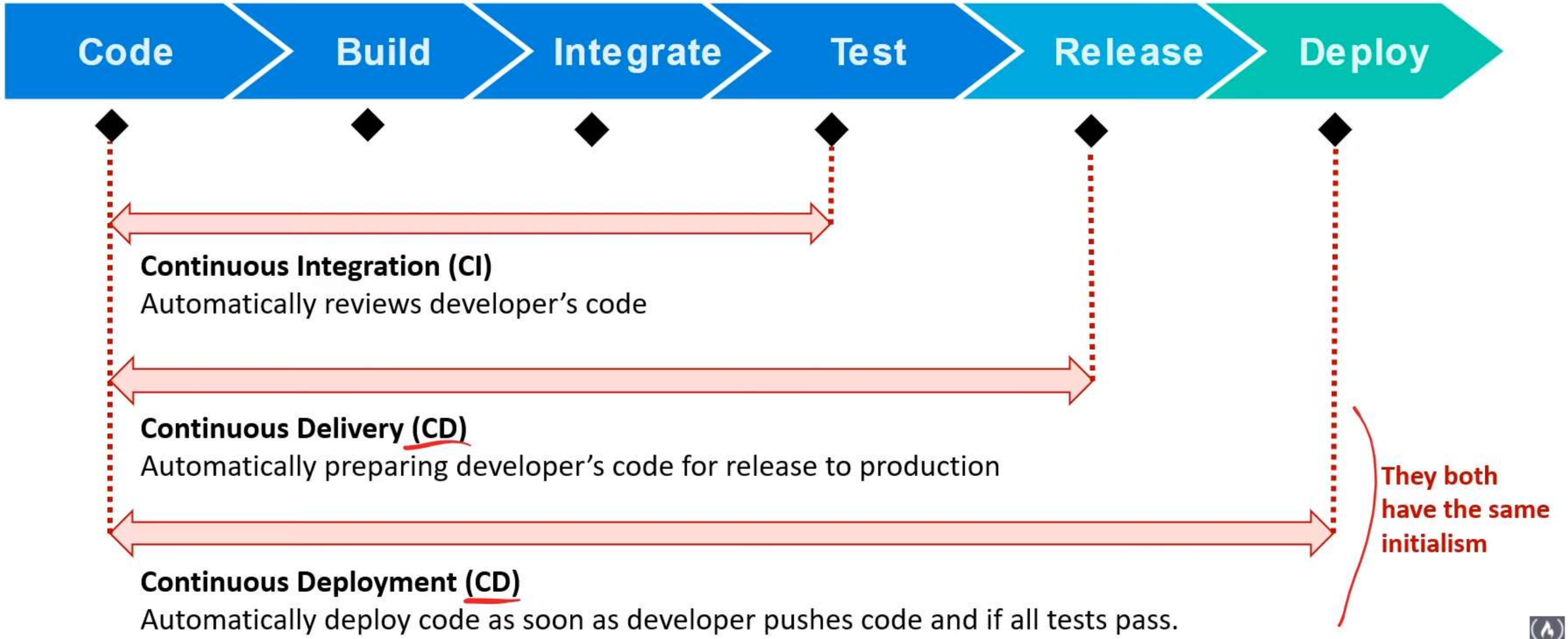
GitOps is when you take Infrastructure as Code (IaC) and you use a git repository to **introduce a formal process to review and accept changes to infrastructure code**, once that code is accepted, it automatically triggers a deploy



CI/CD Models

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

Production (prod) is the live server where real paying users are using the platform
Staging is a private server developers do a final manual tests as a customer (QA) before deploying to prod



Argo vs Flux

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

CNCF have two CI/CD projects that serve the same purpose but take a different approach.

flux

- Originally developed by Weaveworks
- CLI-first approach
- ExperimentalWeb UI as a plugin
- Supports Role-based access controls (RBAC)
- Supports multitenancy (Flux 2)
- Supports Helm and Kustomization
- Automation of Container Updates

argo

- Both a CLI and Web UI
- Supports Role-based access controls (RBAC)
- Supports Single Sign On (SSO)
- Supports multitenancy
- Helm, Kustomization, ksonnet and jsonnet
- Manual commit and sync to update containers

Generally Flux is simpler in design and CLI focused

Jenkins and Jenkins X and CloudBees

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

Jenkins is an open-source popular and mature CI/CD tool **for any kind of workload**.
Jenkins can be used to deploy applications on to Kubernetes.
Jenkins is written in Java, and have many plugins for any use case.

Jenkins X is an open-source CI/CD tool for modern cloud applications on Kubernetes.
Compared to Jenkins its suppose to be much easier to use.

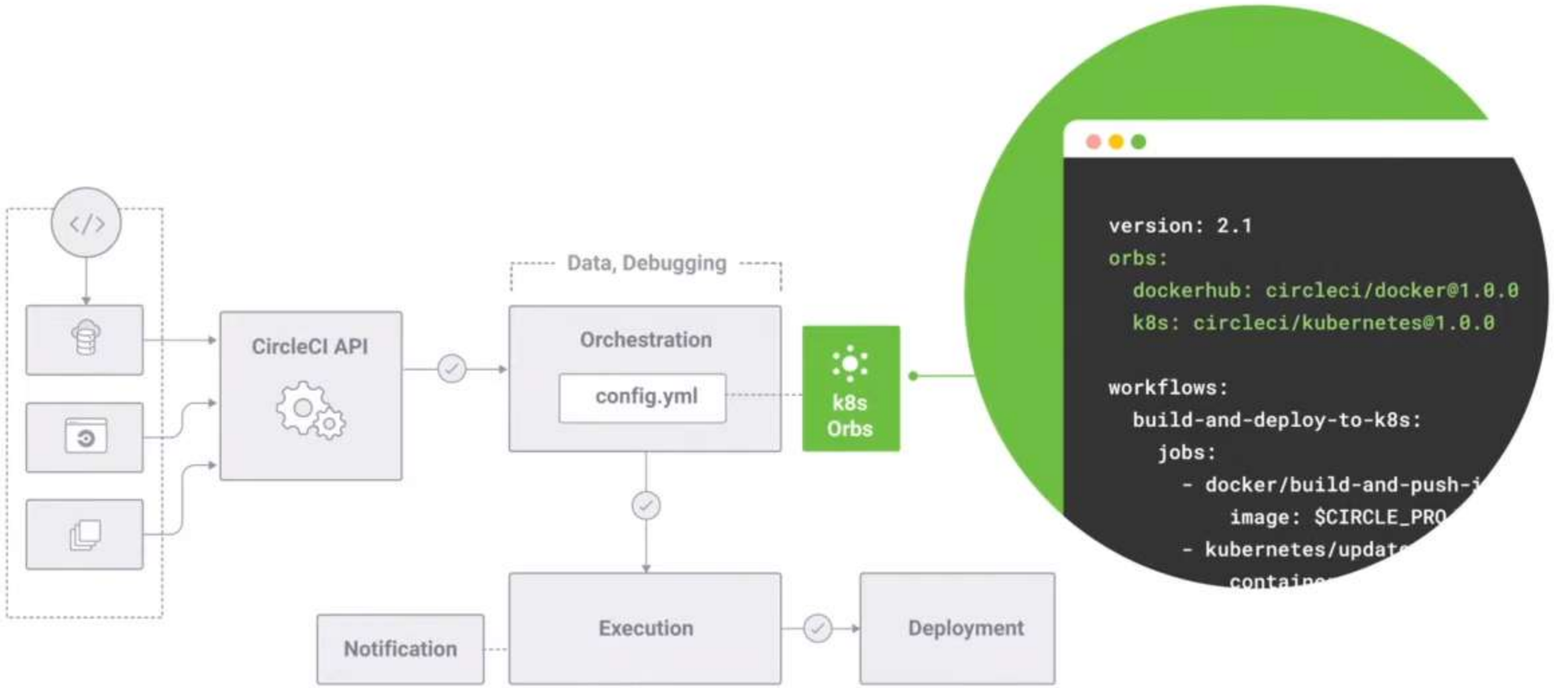
Jenkins X may replace or merge with Jenkins one day to only have a single CI/CD tool for all use cases.

CloudBees is the commercial distribution of Jenkins and Jenkins X for large and compliance first organizations
CloudBees acquired InfraDNA, InfraDNA organization created Jenkins.

CircleCI

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

CircleCI is a propriety fully managed CI/CD service to make deployments easy and seamless
CircleCI can support deploying applications to Kubernetes



Deployment Strategies

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

A deployment strategy is **a way to change or upgrade an application**.

- **Rollout** is when you replace or update servers with new version of an application
- **Rollback** is when you replace or revert recently updated servers back to the previous version

There are several different deployment strategies that can be utilized with Kubernetes:

- **Recreate** – Terminate the current pods, create new pods all at once
- **RollingUpdate** – Replaces one or multiple pods at a time
- **Canary** – Add new pods and route a subset of your users to the new server, if no bugs or errors occur, roll out changes to all pods
- **Blue / Green** – Deploy an exact copy of your entire infrastructure, swap the traffic and then terminate the old environment or rollback to old environment
- **A/B Testing, Red / Black** – Similar to Canary and Blue/Green but continuously keep the new environments running to test features on a subset of users
- **Dark Launches** – Similar to A/B Testing, except you use a feature flag to rollout new features, and rollback by turning off the feature in the software instead of reverting infrastructure changes

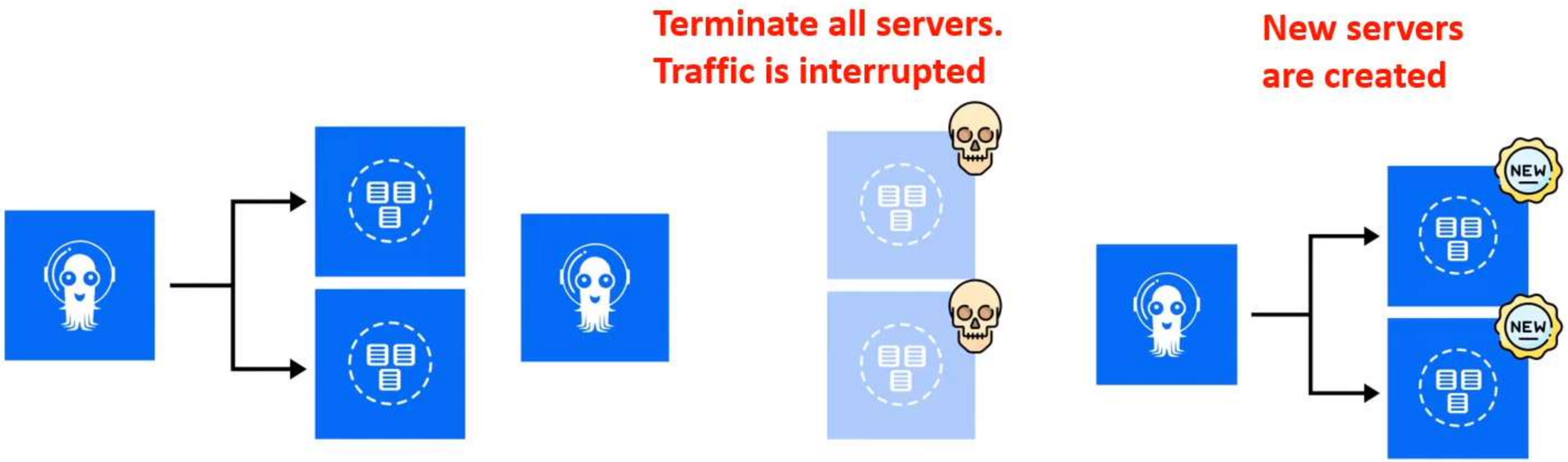
Recreate

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

Terminate all running instances and recreate with a new version.
Also known as an In-Place deploy

```yaml
spec:
  replicas: 3
  strategy:
    type: Recreate
```

This is a **snippet**

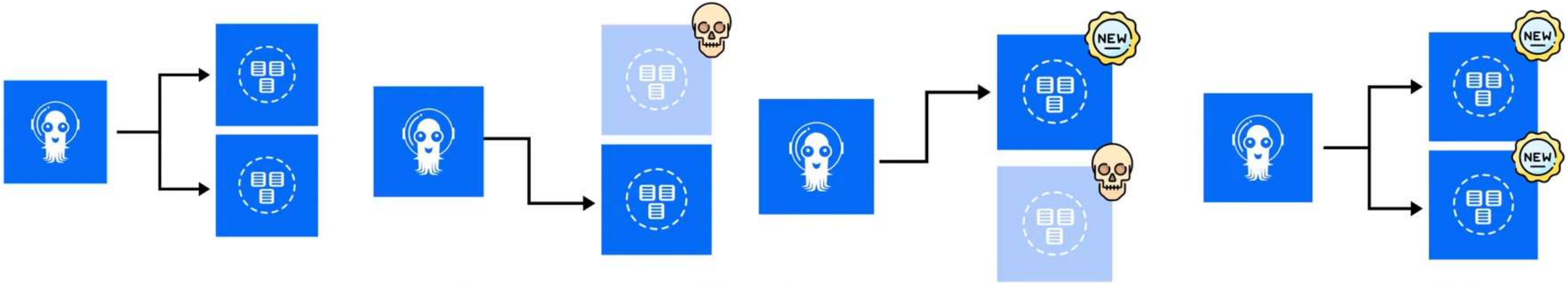


- Users will experience downtime.
- Can be very fast
- Very simple
- Rollback is not possible
- Ideal for non-production workloads or where interruptions can be tolerated

Rolling Updates

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

Rolling Updates **slowly replaces pods one by one**.
This is the **default strategy** of Kubernetes



Terminate an amount of pods

Create new Pods, and Terminate other pods

Repeat until done

- *Reduced availability might happen while each set of pods is taken terminated as new are created
- Rollbacks can be slow and hard
- Deploys will be slow

Rolling Updates

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

What is availability?
The quality of being able to be used or obtained.
If there is not enough capacity (memory, CPU, bandwidth) to meet the demand of traffic then users can experience degraded, delayed experience or no access to services at all.

Two important values:
maxSurge:
- The amount of pods that can be added

maxUnavailable :
- The amount of pods that can be unavailable

```yaml
spec:
  replicas: 3
  strategy:
    type: RollingUpdate
    rollingUpdate:
      maxSurge: 2          # how many pods we can add at a time
      maxUnavailable: 0    # how many pods can be unavailable
```

A maxSurge of 2 and a maxUnavailable of 2 would ensure no drop in availability.
The deployment would have to first create the new pods before tearing down the old.

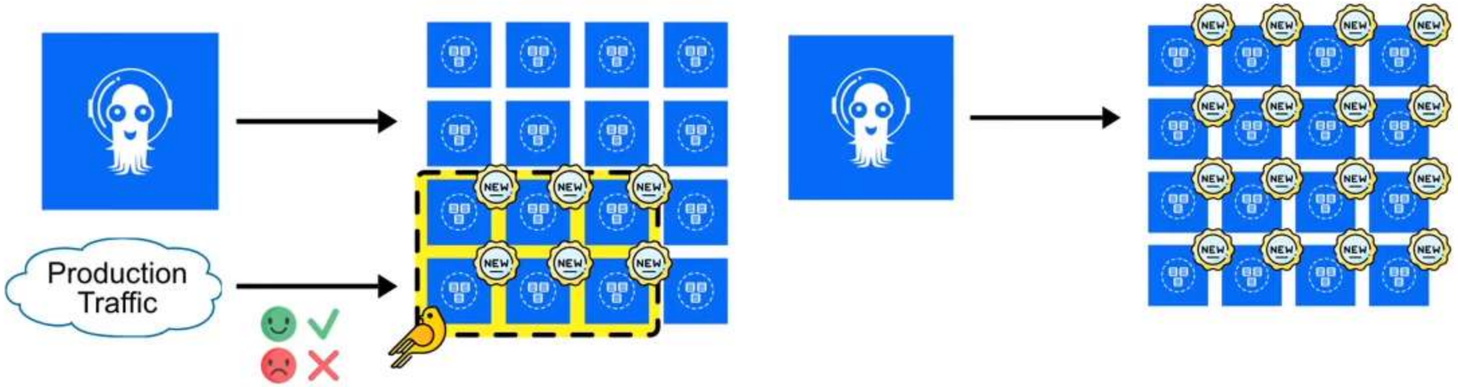
Canaries

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

Deploy a new version of the app into a new pod and serve it to a subset of existing users.
If no error or bug has occurred then rollout changes to all users by replacing old pods with new pods

Create new pods.

Send a portion of production traffic to new pods



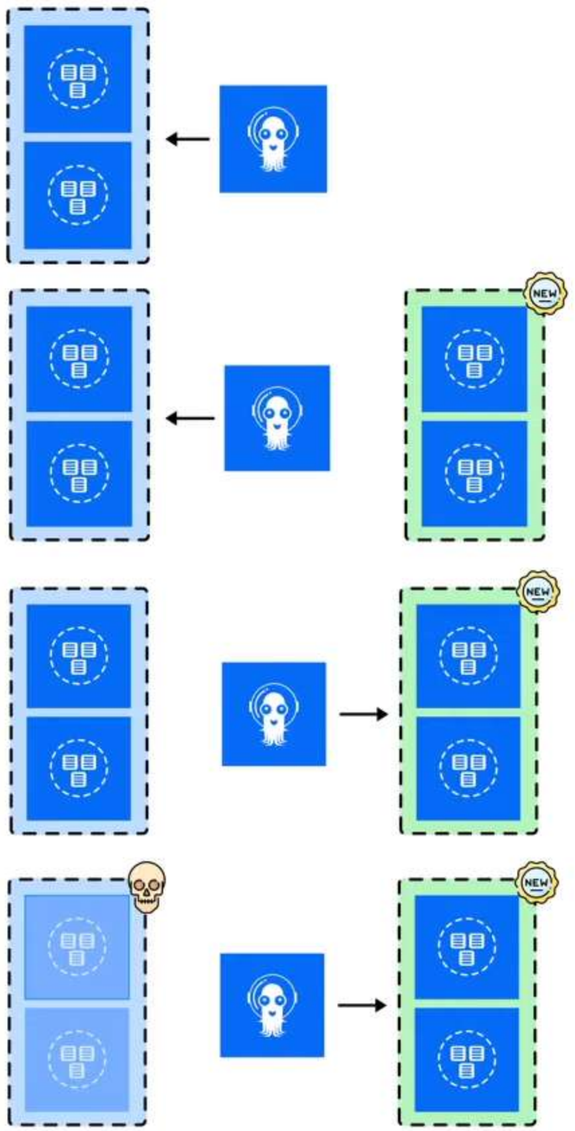
If all good, replace all existing pods with new pods.

Canary will use the load balancer weighted rules to only send an amount of traffic to the the canary pods and original pods.

- Fast rollout
- Slow rollback
- No drop in availability

Blue/Greens

Blue / Green Deployment Is when you **completely create a new environment of all components**, and you send all traffic to the new (Green) environment, if its all okay, you then terminate the old (blue) environment. If anything goes wrong you rollback to blue and teardown green.

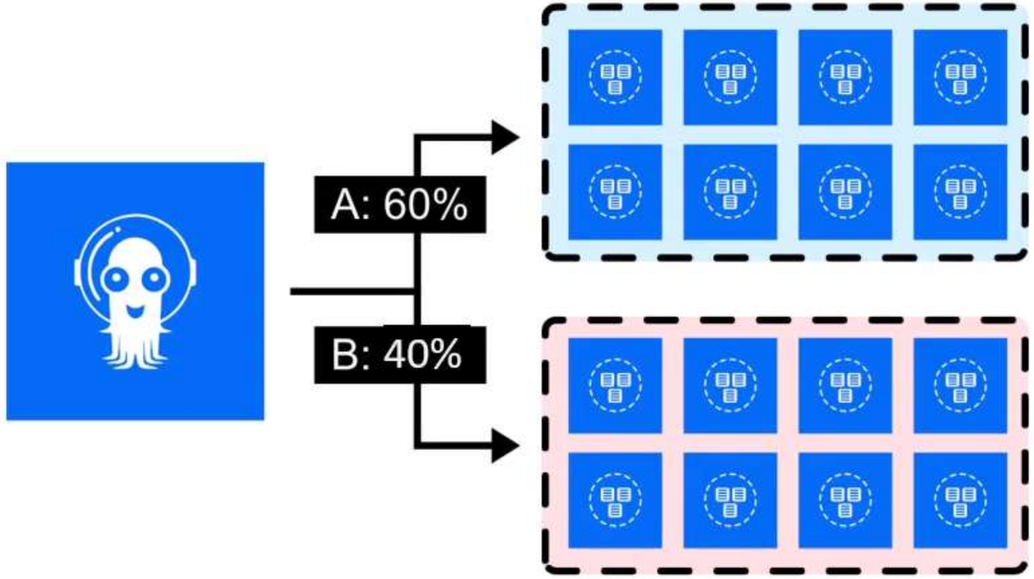


- Zero downtime
- No reduce availability
- Slow to deploy but faster than Canary
- If something goes wrong larger impact to users immediately
- Instantly rollback to previous infrastructure

A to B Testing or Red/Black

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

Uses a Canary or Blue Green method of deployment but serves the new app (experimental features) to subset of users based on a set of load balancing rules.



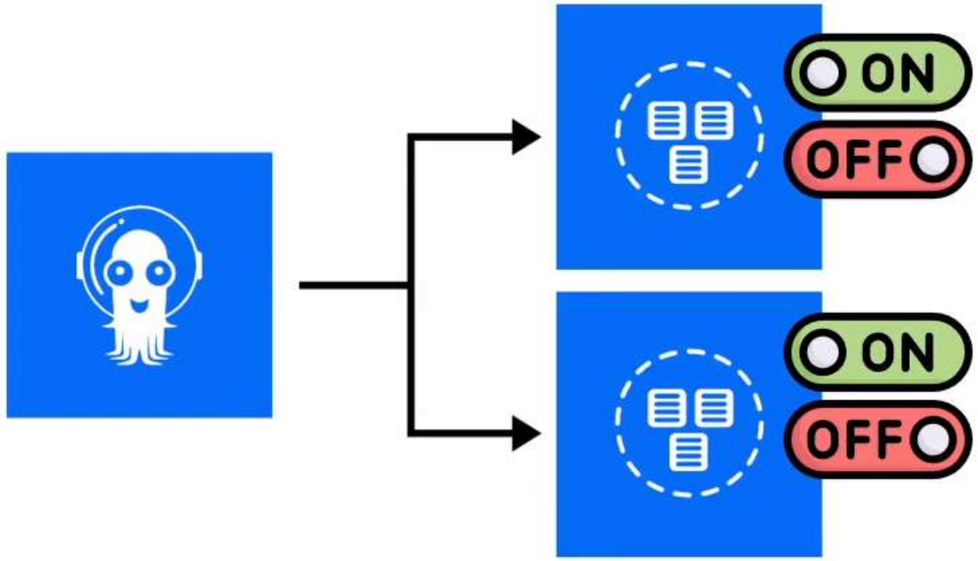
≠ The key difference is Canary and Blue/Green deployment is temporary as you intend to rollout changes to all pods. Where A-to-B Testing and Red/Black happen for a long period of time.

Dark Launches

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

Similar to A/B Testing, except A to B happens **at the application layer** (within the app code)

- You want to test a feature on a subset of users,
- You code a feature flag into your app to turn the new feature on or off
- If the users like the feature you leave it switched on, if they don't you turn it off.
- Doesn't require you to rollback at the infrastructure level. (fast rollback)



Deployment History Command

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

You can check the history of previous deploys with the following command:

```
kubectl rollout history deploy/<deployment name>
```

```
andrewbrown:~/environment $ kubectl rollout history deploy/my-app
deployment.apps/my-app
REVISION  CHANGE-CAUSE
1         kubectl apply --filename=k8s/my-app.yml --record=true
```

Deployment Rollout Status

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

You can the status of you deploy with the rollout status command:

```
kubectl rollout status deployment/<deployment name>
```

```
andrewbrown:~/environment $ kubectl rollout status deploy/my-app
Waiting for deployment "my-app" rollout to finish: 1 of 2 updated replicas
are available...
deployment "my-app" successfully rolled out
```

Deployment Rollback

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

You can rollback to the previous deploy (shown in the rollout history) with rollout undo

```
kubectl rollout undo deployment/<deployment name>
```

```
andrewbrown:~/environment $ kubectl rollout undo deploy/my-app
deployment.apps/my-app rolled back
andrewbrown:~/environment $ kubectl rollout status deploy/my-app
Waiting for deployment "my-app" rollout to finish: 1 out of 2 new
.
```