- 46% Domain 1: Kubernetes Fundamentals
- 22% Domain 2: Container Orchestration
- 16% Domain 3: Container Native Architecture
- 8% Domain 4: Cloud Native Observability
- 8% Domain 5: Cloud Native Application Delivery
- There are 60 questions on the exam
- The exam time is 90 minutes
- The passing score is 75%

ExamPro - https://www.exampro.co/kcna

kubectl Cheat Sheet - https://kubernetes.io/docs/reference/kubectl/cheatsheet/

https://training.linuxfoundation.org/training/introduction-to-kubernetes/

https://github.com/bradmccoydev/mentoring/blob/main/programs/kcna.md

YouTube Videos:

Plural Sight - https://youtu.be/CO0e5lrPuC8

https://www.youtube.com/watch?v=iGkFHB1kFZ0

https://kubernetes.io/docs/home/

Cloud Native Computing Foundation (CNCF)

Cloud native computing uses an open source software stack to deploy applications as microservices, packaging each part into its own container, and dynamically orchestrating those containers to optimize resource utilization

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

Cloud native architecture can provide solutions for the increasing complexity of applications and the growing demand by users. The basic idea is to break down your application in smaller pieces which makes them more manageable. Instead of providing all functionality in a single application, you have multiple decoupled applications that communicate with each other in a network. If we stick to the example from before, you could have an app for your user interface, your checkout and everything else.

Characteristics of Cloud Native Architecture

- High level of automation
- Self healing
- Scalable
- (Cost-) Efficient
- Easy to maintain
- Secure by default

Autoscaling

Dynamic adjustment of resources based on the current demand

Cloud environments rely on usage based on-demand pricing models provide very effective platforms for automatic scaling, with the ability to provision a large amount of resources within seconds or even scale to zero, if resources are temporarily not needed.

Serverless

cloud vendors have one or more commercial offerings of proprietary serverless runtimes and a subset of serverless, also known as Function as a Service (FaaS). The cloud provider abstracts the underlying infrastructure (configure resources like a network, virtual machines, operating systems and load balancers to run a simple web application), so that developers can deploy software by uploading their code for example as .zip files or providing a container image.

Open Standards

Open Container Initiative provides two standards which define the way how to build and run containers. The image-spec defines how to build and package container images. While the runtime-spec specifies the configuration, execution environment and lifecycle of containers

- OCI Spec: image, runtime and distribution specification on how to run, build and distribute containers
- Container Network Interface (CNI): A specification on how to implement networking for Containers.
- Container Runtime Interface (CRI): A specification on how to implement container runtimes in container orchestration systems.
- Container Storage Interface (CSI):A specification on how to implement storage in container orchestration systems.
- Service Mesh Interface (SMI): Ap specification on how to implement Service Meshes in container orchestration systems with a focus on Kubernetes.

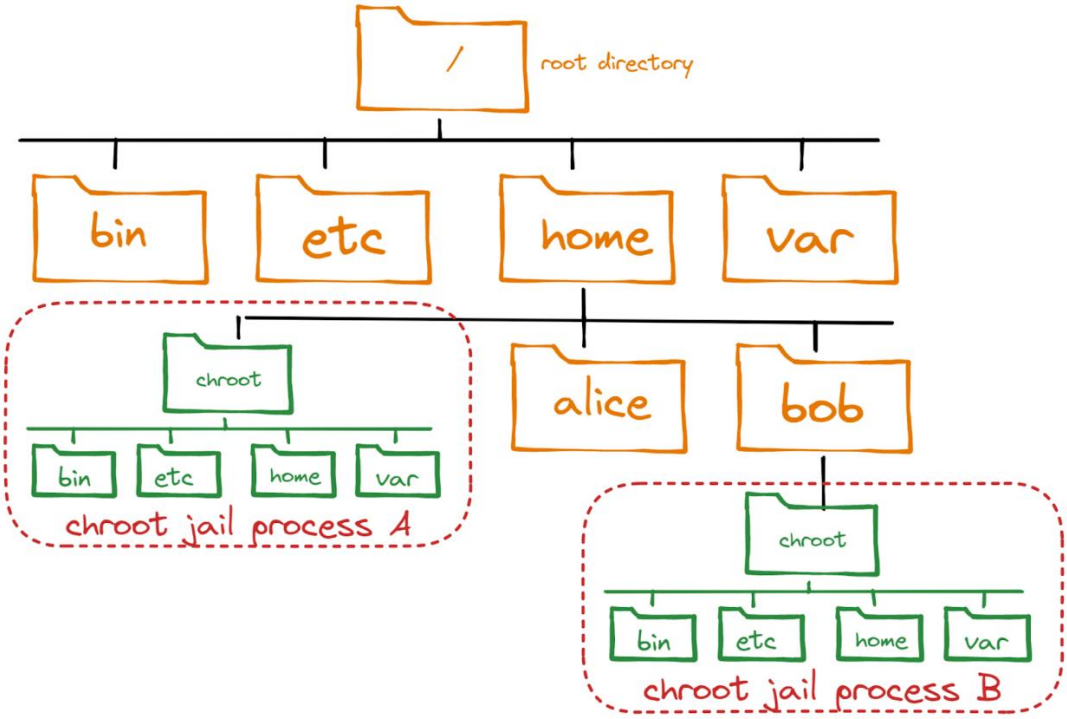
Community and Governance

The CNCF has a Technical Oversight Committee (TOC) that is responsible for defining and maintaining the technical vision, approving new projects, accepting feedback from the end-user committee, and defining common practices that should be implemented in CNCF projects.

Container Orchestration

Containers can be used to solve problems of managing the dependencies of an application and running much more efficiently than spinning up a lot of virtual machines.

The **chroot** command could be used to isolate a process from the root filesystem and basically "hide" the files from the process and simulate a new root directory. The isolated environment is a so-called *chroot jail*, where the files can't be accessed by the process, but are still present on the system.



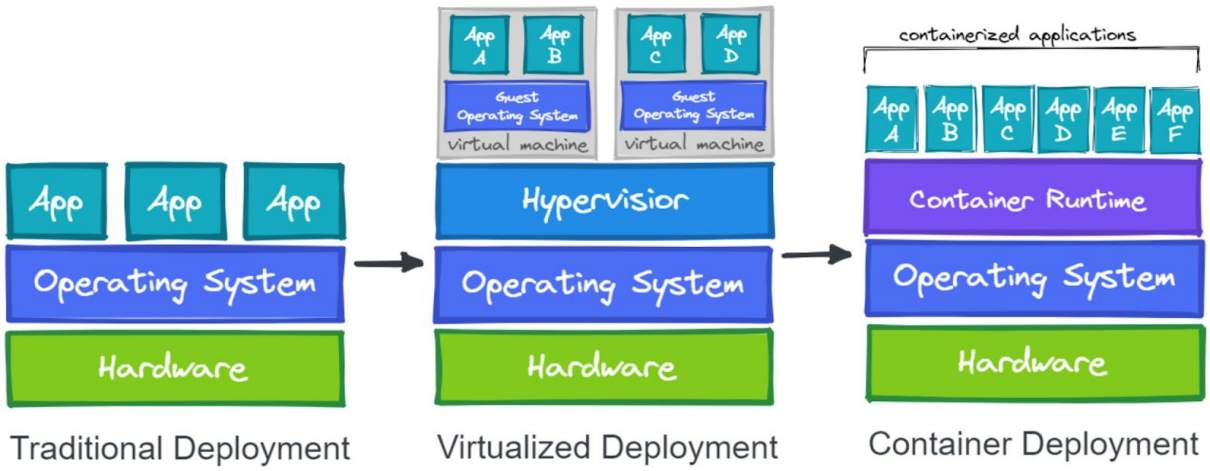
Container technologies embody this chroot concept. To isolate a process even more than chroot can do, current Linux kernels provide features like *namespaces* and cgroups.

Namespaces are used to isolate various resources, for example the network. A network namespace can be used to provide a complete abstraction of network interfaces and routing tables.

The Linux Kernel 5.6 currently provides 8 namespaces:

- pid - process ID provides a process with its own set of process IDs.
- net - network allows the processes to have their own network stack, including the IP address.
- mnt - mount abstracts the filesystem view and manages mount points.
- ipc - inter-process communication provides separation of named shared memory segments.
- user - provides process with their own set of user IDs and group IDs.
- uts - Unix time sharing allows processes to have their own hostname and domain name.
- cgroup - a newer namespace that allows a process to have its own set of cgroup root directories.
- time - the newest namespace can be used to virtualize the clock of the system.

cgroups are used to organize processes in hierarchical groups and assign them resources like memory and CPU



A Docker container image is a lightweight, standalone, executable package of software that includes everything needed to run an application: code, runtime, system tools, system libraries and settings

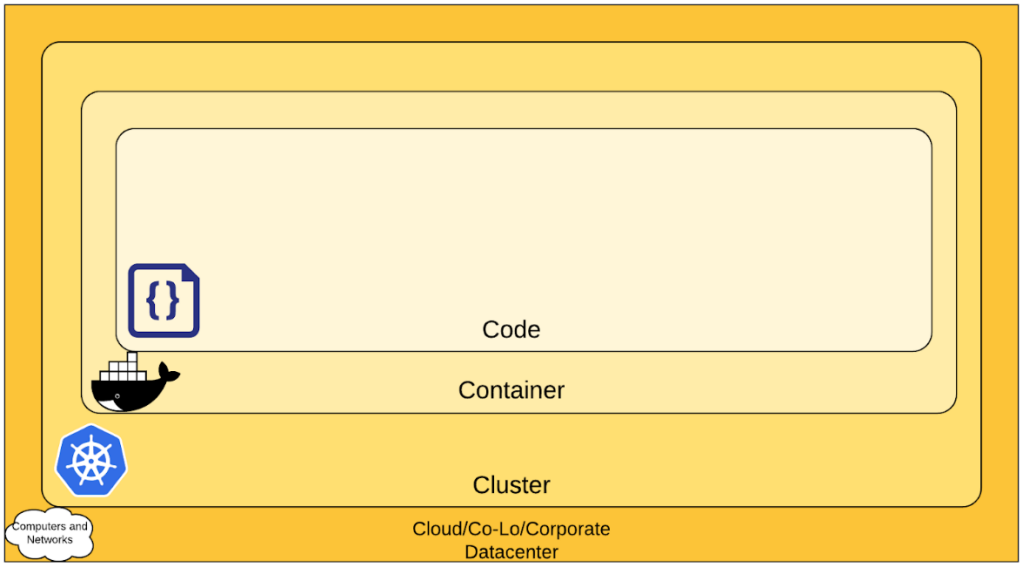
Container images are what makes containers portable and easy to reuse on a variety of systems. Images can be built by reading the instructions from a buildfile called Dockerfile.

```
docker build -t my-python-image -f Dockerfile
docker push my-registry.com/my-python-image
docker pull my-registry.com/my-python-image
```

Security

When containers are started on a machine, they always share the same kernel, which then becomes a risk for the whole system, if containers are allowed to call kernel functions like for example killing other processes or modifying the host network by creating routing rules

One of the greatest security risks, not only in the container area, is the execution of processes with too many privileges, especially starting processes as root or administrator. The 4C's of Cloud Native security can give a rough idea what layers need to be protected if you're using containers.



Networking

Most modern implementations of container networking are based on the Container Network Interface (CNI). CNI is a standard that can be used to write or configure network plugins

Network namespaces allow each container to have its own unique IP address, therefore multiple applications can open the same network port; for example, you could have multiple containerized web servers that all open port 8080.

To make the application accessible from outside the host system, containers have the ability to map a port from the container to a port from the host system.

To allow communication between containers across hosts, we can use an overlay network which puts them in a virtual network that is spanned across the host systems.

Service Discovery & DNS

Containers are deployed on varieties of different hosts, different data centers or even geolocations. Containers or Services need DNS to communicate. Using IP addresses is nearly impossible

Instead of having a manually maintained list of servers (or in this case containers), all the information is put in a *Service Registry*. Finding other services in the network and requesting information about them is called *Service Discovery*

DNS and Key value store are the approaches for Service Discovery.

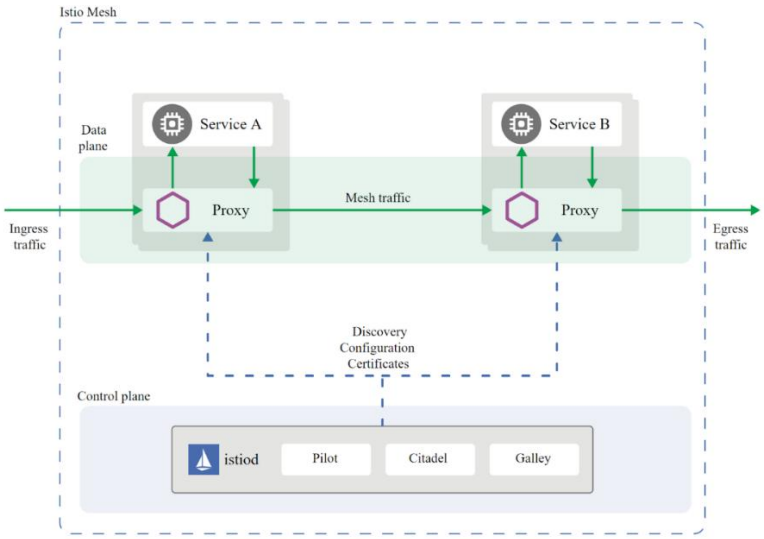
Service Mesh

A service mesh adds a proxy server to *every* container that you have in your architecture. The software you can use to manage network traffic is called a proxy. This is a server application that sits between a client and server and can modify or filter network traffic before it reaches the server.

When a service mesh is used, applications don't talk to each other directly, but the traffic is routed through the proxies instead. The most popular service meshes at the moment are istio and linkerd.

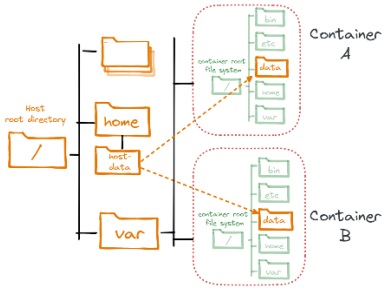
The proxies in a service mesh form the *data plane*. This is where networking rules are implemented and shape the traffic flow.

So instead of writing code and installing libraries, you just write a config file where you tell the service mesh that service A and service B should always communicate encrypted. The config is then uploaded to the control plane and distributed to the data plane to enforce the new rule.

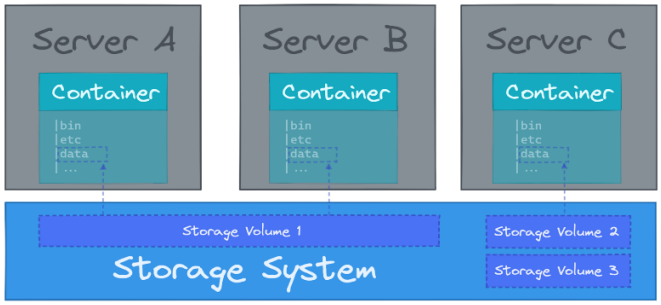


Storage

The problem here is that this read-write layer is lost when the container is stopped or deleted. Just like the memory of your computer gets erased when you shut it down. To persist data, you need to write it to your disk**.**



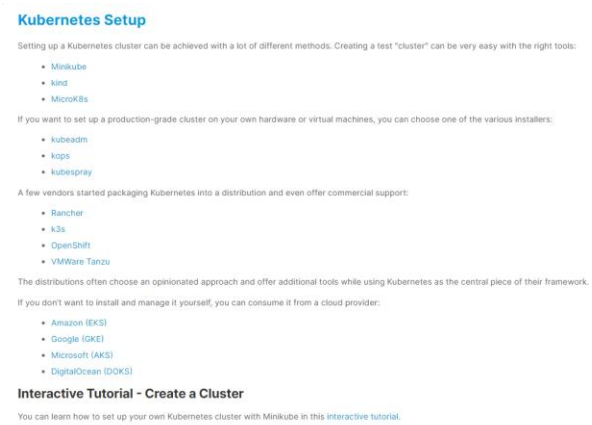
Container orchestration systems like Kubernetes can help to mitigate these problems, but always require a robust storage system that is attached to the host servers.



Storage is provisioned via a central storage system. Containers on Server A and Server B can share a volume to read and write data

Kubernetes Architecture

Kubernetes provides you with a framework to run distributed systems resiliently. It takes care of scaling and failover for your application, provides deployment patterns, and more. Kubernetes is often used as a cluster, meaning that it is spanned across multiple servers that work on different tasks and to distribute the load of a system



Kubernetes Setup

Setting up a Kubernetes cluster can be achieved with a lot of different methods. Creating a test "cluster" can be very easy with the right tools:

- Minikube
- kind
- MicroK8s

If you want to set up a production-grade cluster on your own hardware or virtual machines, you can choose one of the various installers:

- kubeadm
- kops
- kubespray

A few vendors started packaging Kubernetes into a distribution and even offer commercial support:

- Rancher
- k3s
- OpenShift
- VMWare Tanzu

The distributions often choose an opinionated approach and offer additional tools while using Kubernetes as the central piece of their framework.

If you don't want to install and manage it yourself, you can consume it from a cloud provider:

- Amazon (EKS)
- Google (GKE)
- Microsoft (AKS)
- DigitalOcean (DOKS)

Interactive Tutorial - Create a Cluster

You can learn how to set up your own Kubernetes cluster with Minikube in this interactive tutorial.

Namespaces

In Kubernetes, *namespaces* provides a mechanism for isolating groups of resources within a single cluster

Kubernetes starts with four initial namespaces: **default, kube-node-lease, kube-public, kube-system**

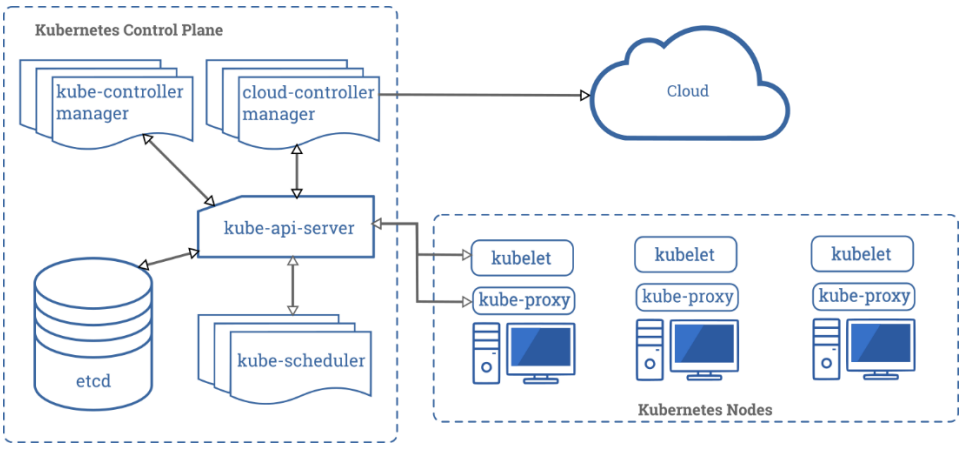
Kubernetes provides you with

Service discovery and load balancing
Storage orchestration
Automated rollouts and rollbacks
Automatic bin packing
Self-healing
Secret and configuration management

Kubernetes clusters consist of two different server node types that make up a cluster

- **Control plane node(s)**
  These are the brains of the operation. Control plane nodes contain various components which manage the cluster and control various tasks like deployment, scheduling and self-healing of containerized workloads.
- **Worker nodes**
  The worker nodes are where applications run in your cluster. This is the only job of worker nodes and they don't have any further logic implemented. Their behavior, like if they should start a container, is completely controlled by the control plane node.

  Self-registering nodes - When the kubelet flag --register-node is true (the default), the kubelet will attempt to register itself with the API server



kubectl describe node – To view status of the node

kube-apiserver

This is the centerpiece of Kubernetes. All other components interact with the api-server and this is where users would access the cluster. It scales components horizontally.

etcd

A database which holds the state of the cluster. etcd is a standalone project and not an official part of Kubernetes.

kube-scheduler

When a new workload should be scheduled, the kube-scheduler chooses a worker node that could fit, based on different properties like CPU and memory.

kube-controller-manager

Contains different non-terminating control loops that manage the state of the cluster. For example, one of these control loops can make sure that a desired number of your application is available all the time.

cloud-controller-manager (optional)

Can be used to interact with the API of cloud providers, to create external resources like load balancers, storage or security groups.

Components of worker nodes

container runtime

The container runtime is responsible for running the containers on the worker node. For a long time, Docker was the most popular choice, but is now replaced in favor of other runtimes like containerd.

kubelet

A small agent that runs on every worker node in the cluster. The kubelet talks to the api-server and the container runtime to handle the final stage of starting containers.

kube-proxy

A network proxy that handles inside and outside communication of your cluster. Instead of managing traffic flow on it's own, the kube-proxy tries to rely on the networking capabilities of the underlying operating system if possible.

Addons

Addons use Kubernetes resources (DaemonSet, Deployment, etc) to implement cluster features. Because these are providing cluster-level features, namespaced resources for addons belong within

the kube-system namespace. Few addons are DNS, Web UI, Container Resource Monitoring, Cluster Level logging rtc.

DNS
It's a service that is responsible for translating a service name to its IP address. CoreDNS is the default DNS server for Kubernetese.

While the other addons are not strictly required, all Kubernetes clusters should have cluster DNS, as many examples rely on it.

Cluster DNS is a DNS server, in addition to the other DNS server(s) in your environment, which serves DNS records for Kubernetes services.

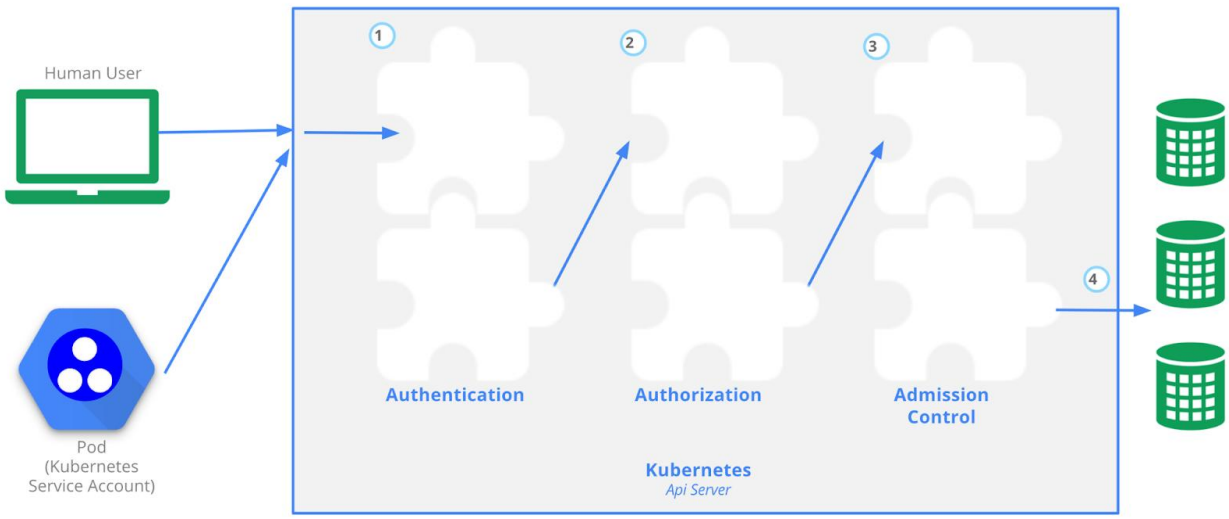
Containers started by Kubernetes automatically include this DNS server in their DNS searches.

Configuration file in Kubernetese are called manifests. They are in the form of yml or json

Kubernetes API

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).



Authentication
The requester needs to present a means of identity to authenticate against the API. Commonly done with a digital signed certificate (X.509) or with an external identity management system. Kubernetes users are *always* externally managed. Service Accounts can be used to authenticate technical users.

Authorization
It is decided what the requester is allowed to do. In Kubernetes this can be done with Role Based Access Control (RBAC).

Admission Control
In the last step, admission controllers can be used to modify or validate the request. For example, if a user tries to use a container image from an untrustworthy registry, an admission controller could block this request. Tools like the Open Policy Agent can be used to manage admission control externally

Running Containers on Kubernetes

The idea of containerd and CRI-O was very simple: provide a runtime that only contains the absolutely essentials to run containers. Nevertheless, they have additional features, like the ability to integrate with container runtime sandboxing tools. These tools try to solve the security problem that comes with sharing the kernel between multiple containers. The most common tools at the moment are:

- gvisor
  Made by Google, provides an application kernel that sits between the containerized process and the host kernel.
- Kata Containers
  A secure runtime that provides a lightweight virtual machine, but behaves like a container.

Networking

Kubernetes distinguishes between four different networking problems that need to be solved:

1. **Container-to-Container communications**
   This can be solved by the Pod concept as we'll learn later. All the containers in a pod will have same IP address
2. **Pod-to-Pod communications**
   This can be solved with an overlay network. Veth is used to communicate b/w same pods. And for pods in different nodes used for ex VPC.
3. **Pod-to-Service communications**
   It is implemented by the kube-proxy and packet filter on the node.
4. **External-to-Service communications**
   It is implemented by the kube-proxy and packet filter on the node.

There are different ways to implement networking in Kubernetes, but also three important requirements:

- All pods can communicate with each other across nodes.
- All nodes can communicate with all pods.
- No Network Address Translation (NAT).

To implement networking, you can choose from a variety of network vendors like:

- Project Calico
- Weave

- Cilium

In Kubernetes, every Pod gets its own IP address, so there is no manual configuration involved. Moreover, most Kubernetes setups include a DNS server add-on called core-dns, which can provide service discovery and name resolution inside the cluster.

A deployment is responsible for keeping a set of pods running. A service is responsible for enabling network access to a set of pods

Endpoints is a way where service links to a pod

Scheduling

In a Kubernetes cluster, the kube-scheduler is the component that makes the scheduling decision, but is not responsible for actually starting the workload. The scheduling process in Kubernetes always starts when a new Pod object is created. Remember that Kubernetes is using a declarative approach, where the Pod is only described first, then the scheduler selects a node where the Pod actually will get started by the kubelet and the container runtime

Working with Kubernetes

Kubernetes has solutions for some inherent problems with containers and orchestration, be it configuration management, cross-node networking, routing of external traffic, load balancing or scaling of the pods.

Workloads

A workload is an application running on Kubernetes. Whether your workload is a single component or several that work together on Kubernetes you run it inside a set of *pods*

Kubernetes Objects

Kubernetes objects are mostly abstract resources, they are used to describe how your workload should be handled.

A Kubernetes object is a "record of intent"--once you create the object, the Kubernetes system will constantly work to ensure that object exists. By creating an object, you're effectively telling the Kubernetes system what you want your cluster's workload to look like; this is your cluster's *desired state*

Almost every Kubernetes object includes two nested object fields that govern the object's configuration: the object spec and the object status. For objects that have a spec, you have to set this when you create the object, providing a description of the characteristics you want the resource to have: its *desired state*.

Kubernetes objects can be distinguished between workload-oriented objects that are used for handling container workloads and infrastructure-oriented objects, that for example handle configuration, networking and security.

Pods: Pod is a collection of containers that can run on a host. This resource is created by clients and scheduled onto hosts

Common and mandatory objects in yaml are apiVersion, Kind, metadata (name) and spec

You can list the available objects in your cluster with the following command:

$ kubectl api-resources

$ kubectl explain pod

$ kubectl explain pod.spec

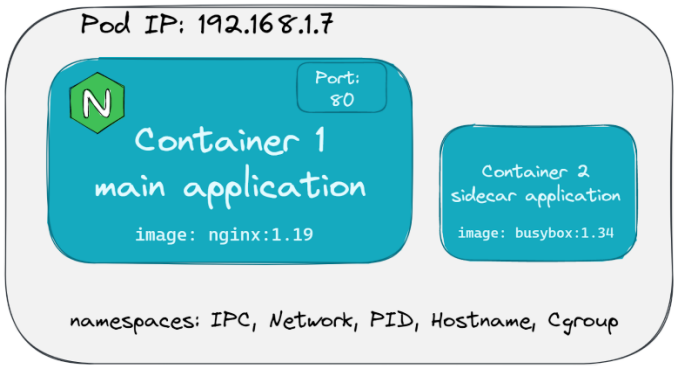
Interacting with Kubernetes

To access the API, users can use the official command line interface client called **kubectl.** Kubectl uses a config file called **kubeconfig.** Kubeconfig can be used to manage multiple clusters users.

Helm: Helm is a package manager for Kubernetes, which allows easier updates and interaction with objects. Helm packages Kubernetes objects in so-called Charts, which can be shared with others via a registry.

Pod Concept

The most important object in Kubernetes is a Pod. A pod describes a unit of one or more containers that share an isolation layer of namespaces and cgroups. It is the smallest deployable unit in Kubernetes, which also means that Kubernetes is not interacting with containers directly. The pod concept was introduced to allow running a combination of multiple processes that are interdependent. All containers inside a pod share an IP address and can share via the filesystem.



Multiple containers share namespaces to form a pod

Here is an example of a simple Pod object with **two containers**:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-with-sidecar
spec:
  containers:
  - name: nginx
    image: nginx:1.19
    ports:
    - containerPort: 80
  - name: count
    image: busybox:1.34
    args: [/bin/sh, -c,
            'i=0; while true; do echo "$i: $(date)"; i=$((i+1));
sleep 1; done']
```

You could add as many containers to your main application as you want. But be careful since you lose the ability to scale them individually! Using a second container that supports your main application is called a sidecar container.

All containers defined are started at the same time with no ordering, but you also have the ability to use initContainers to start containers before your main application starts. In this example, the init container init-myservice tries to reach another service. Once it completes, the main container is started.

```
apiVersion: v1
kind: Pod
metadata:
  name: myapp-pod
  labels:
    app: myapp
spec:
```

```
  containers:
  - name: myapp-container
    image: busybox
    command: ['sh', '-c', 'echo The app is running! && sleep
3600']
  initContainers:
  - name: init-myservice
    image: busybox
    command: ['sh', '-c', 'until nslookup myservice; do echo
waiting for myservice; sleep 2; done;']
```
Make sure to explore the documentation about Pods, since there are many more settings to discover.

Probes are used to detect the state of the containers.
Some examples probes of important settings that can be set for every container in a Pod are:

resources: Set a resource request and a maximum limit for CPU and Memory.
livenessProbe: Configure a health check that periodically checks if your application is still alive. Containers can be restarted if the check fails.
securityContext: Set user & group settings, as well as kernel capabilities.
Readiness probe: When a container is ready to accept traffic
Startup probe: when application has started

You can use workload resources to create and manage multiple Pods for you. A controller for the resource handles replication and rollout and automatic healing in case of Pod failure.

PodTemplates are specifications for creating Pods, and are included in workload resources such as Deployments, Jobs, and DaemonSets.

when the Pod template for a workload resource is changed, the controller creates new Pods based on the updated template instead of updating or patching the existing Pods.

- Most of the metadata about a Pod is immutable. For example, you cannot change the namespace, name, uid, or creationTimestamp fields; the generation field is unique. It only accepts updates that increment the field's current value.
- Pod updates may not change fields than spec.containers[*].image, spec.initContainers[*].image, spec.activeDeadlineSeconds or spec.tolerations. For spec.tolerations, you can only add new entries.
- A Pod can specify a set of shared storage volumes. All containers in the Pod can access the shared volumes, allowing those containers to share data. Volumes also allow persistent data in a Pod to survive in case one of the containers within needs to be restarted

The name of a Pod must be a valid DNS subdomain value. Every container in a Pod shares the network namespace, including the IP address and network ports.

Static Pods are managed directly by the kubelet daemon on a specific node, without the API server observing them

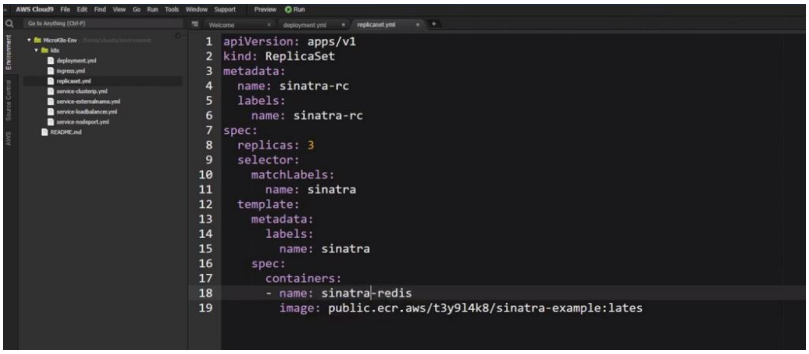
A probe is a diagnostic performed periodically by the kubelet on a container. To perform a diagnostic, the kubelet can invoke different actions:

- ExecAction (performed with the help of the container runtime)
- TCPSocketAction (checked directly by the kubelet)
- HTTPGetAction (checked directly by the kubelet)

Workload Objects

Working just with Pods would not be flexible enough in a container orchestration platform. For example, if a Pod is lost because a node failed, it is gone forever. To make sure that a defined number of Pod copies runs all the time, we can use controller objects that manage the pod for us.

ReplicaSet: A controller object that ensures a desired number of pods is running at any given time. ReplicaSets can be used to scale out applications and improve their availability. They do this by starting multiple copies of a pod definition.



Deployment: The most feature-rich object in Kubernetes. A Deployment can be used to describe the complete application lifecycle, they do this by managing multiple ReplicaSets that get updated when the application is changed by providing a new container image, for example. Deployments are perfect to run stateless applications in Kubernetes. (A stateless app is an application program that does not save client data generated in one session for use in the next session with that client). A *Deployment* provides declarative updates for Pods and ReplicaSets.

Create the Deployment by running the following command:
kubectl apply -f https://k8s.io/examples/controllers/nginx-deployment.yaml

Run kubectl get deployments to check if the Deployment was created.

To update the nginx Pods to use the nginx:1.16.1 image instead of the nginx:1.14.2 image.
kubectl set image deployment/nginx-deployment nginx=nginx:1.16.1

Get details of your Deployment: kubectl describe deployments

StatefulSet: Considered a bad practice for a long time, StatefulSets can be used to run stateful applications like databases on Kubernetes. Stateful applications have special requirements that don't fit

the ephemeral nature of pods and containers. In contrast to Deployments, StatefulSets try to retain IP addresses of pods and give them a stable name, persistent storage and more graceful handling of scaling and updates.
Unlike a Deployment, a StatefulSet maintains a sticky identity for each of their Pods.

Stateful set will always have unique and predictable names.A persistent volume will be attached. Will use headless services.

Ephemeral - loading for a very short time

DaemonSet: Ensures that a copy of a Pod runs on all (or some) nodes of your cluster. DaemonSets are perfect to run infrastructure-related workload, for example monitoring or logging tools.

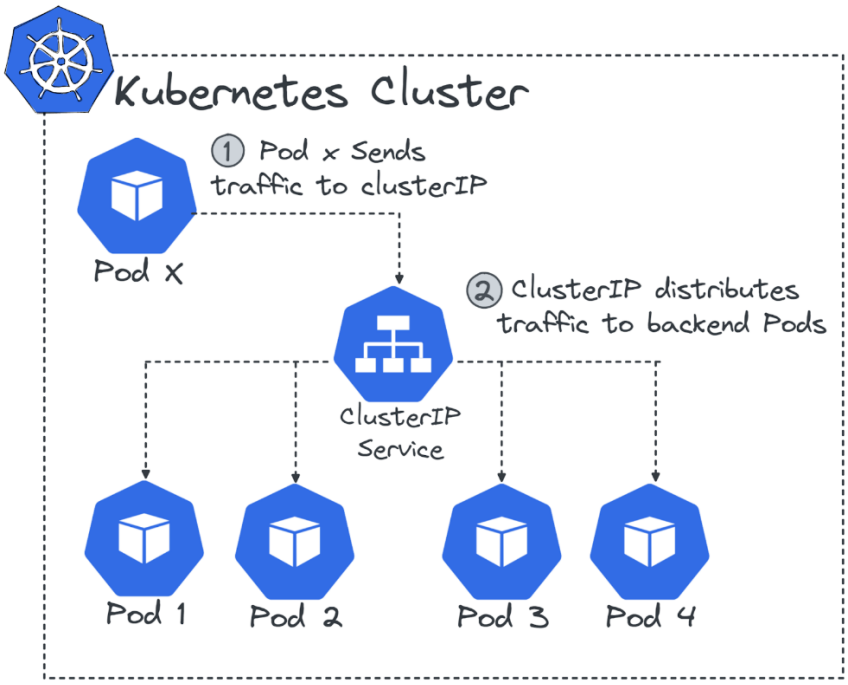
Job: Creates one or more Pods that execute a task and terminate afterwards. Job objects are perfect to run one-shot scripts like database migrations or administrative tasks.

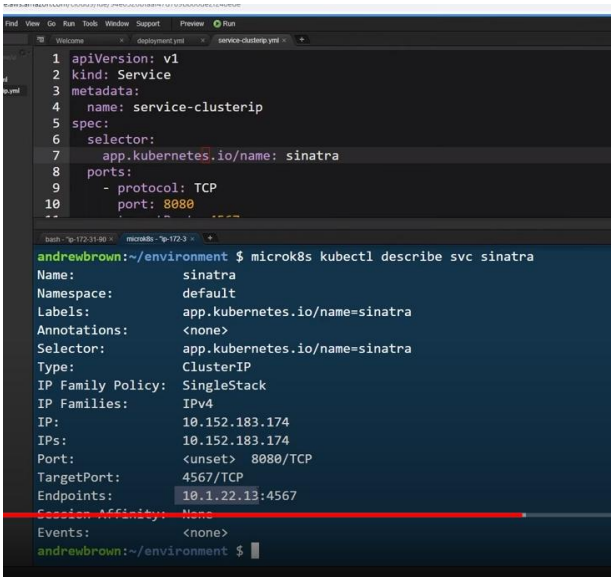
CronJob: CronJobs add a time-based configuration to jobs. This allows running Jobs periodically, for example doing a backup job every night at 4am.
S

Networking Objects

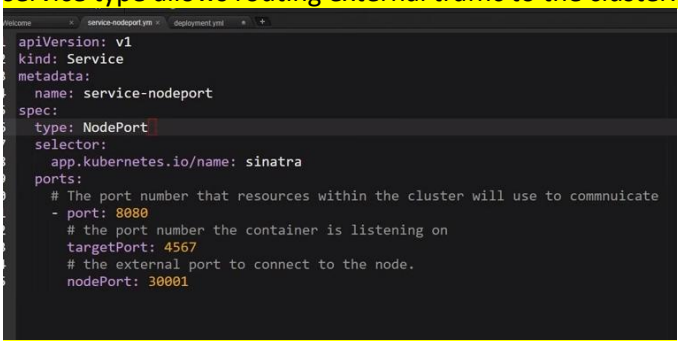
Since a lot of Pods would require a lot of manual network configuration, we can use *Service* and *Ingress objects* to define and abstract networking. Services can be used to expose a set of pods as a network service.

ClusterIP: The most common service type. A ClusterIP is a virtual IP inside Kubernetes that can be used as a single endpoint for a set of pods. This service type can be used as a round-robin load balancer.





NodePort: The NodePort service type extends the ClusterIP by adding simple routing rules. It opens a port (default between 30000-32767) on every node in the cluster and maps it to the ClusterIP. This service type allows routing external traffic to the cluster.

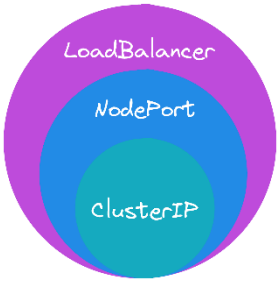


LoadBalancer: The LoadBalancer service type extends the NodePort by deploying an external LoadBalancer instance. This will only work if you're in an environment that has an API to configure a LoadBalancer instance, like GCP, AWS, Azure or even OpenStack. (AWS has – Application, Network and gateway load balances)

```yaml
metadata:
  name: service-loadbalancer
spec:
  selector:
    app.kubernetes.io/name: sinatra
  ports:
    - protocol: TCP
      port: 8080
      targetPort: 4567
  clusterIP: 10.0.171.239
  type: LoadBalancer
status:
  loadBalancer:
    ingress:
    - ip: 192.0.2.127
```

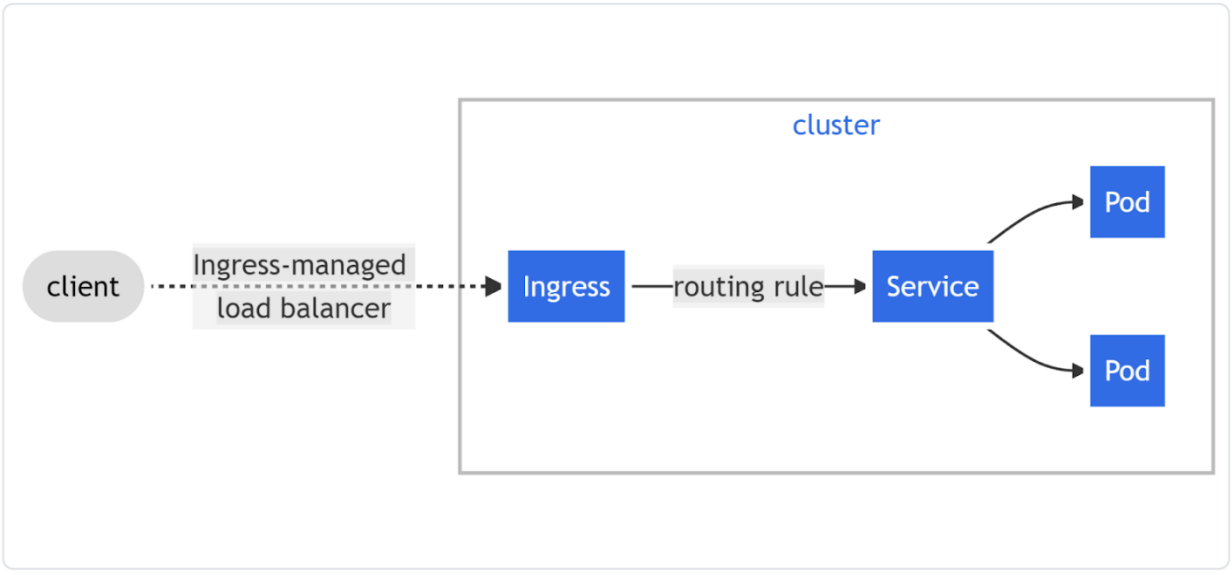
s

ExternalName: A special service type that has no routing whatsoever. ExternalName is using the Kubernetes internal DNS server to create a DNS alias. You can use this to create a simple alias to resolve a rather complicated hostname like: my-cool-database-az1-uid123.cloud-provider-i-like.com. This is especially useful if you want to reach external resources from your Kubernetes cluster.



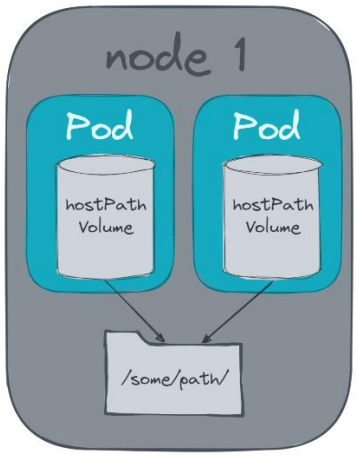
Ingress

If you need even more flexibility to expose applications, you can use an *Ingress object*. Ingress provides a means to expose HTTP and HTTPS routes from outside of the cluster for a service within the cluster. It does this by configuring routing rules that a user can set and implement with an *ingress controller*.



Volume & Storage Objects

```
apiVersion: v1
kind: Pod
metadata:
  name: test-pd
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-pd
      name: test-volume
  volumes:
  - name: test-volume
    hostPath:
      # directory location on host
      path: /data
      # this field is optional
      type: Directory
```



Volumes allow sharing data between multiple containers within the same Pod.

- **PersistentVolumes (PV)**
  An abstract description for a slice of storage. The object configuration holds information like type of volume, volume size, access mode and unique identifiers and information how to mount it.
- **PersistentVolumeClaims (PVC)**
  A request for storage by a user. If the cluster has multiple persistent volumes, the user can create a PVC which will reserve a persistent volume according to the user's needs.

Configuration Objects

It is considered bad practice to incorporate the configuration directly into the container build. Any configuration change would require the entire image to be rebuilt and the entire container or pod to be redeployed. This problem gets only worse when multiple environments (development, staging, production) are used and images are being built for each and every environment. The twelve factor app explains this problem in more detail: Dev/prod parity.

In Kubernetes, this problem is solved by decoupling the configuration from the Pods with a ConfigMap. ConfigMaps can be used to store whole configuration files or variables as key-value pairs. There are two possible ways to use a ConfigMap:

- Mount a ConfigMap as a volume in Pod
- Map variables from a ConfigMap to environment variables of a Pod.

Here is an example of a ConfigMap that contains a nginx configuration:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: nginx-conf
data:
  nginx.conf: |
    user nginx;
    worker_processes 3;
    error_log /var/log/nginx/error.log;
...
      server {
          listen     80;
          server_name _;
          location / {
              root   html;
              index  index.html index.htm; } } }
```

Once the ConfigMap is created you can use it in a Pod:

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx
spec:
  containers:
  - name: nginx
    image: nginx:1.19
    ports:
    - containerPort: 80
    volumeMounts:
```

```
    - mountPath: /etc/nginx
      name: nginx-conf
  volumes:
  - name: nginx-conf
    configMap:
      name: nginx-conf
```

Right from the beginning Kubernetes also provided an object to store sensitive information like passwords, keys or other credentials. These objects are called *Secrets*. Secrets are very much related to ConfigMaps and basically their only difference is that secrets are base64 encoded.

There is an on-going debate about the risk of using Secrets, since their - in contrast to their name - not considered secure. In cloud-native environments purpose-built secret management tools have emerged that integrate very well with Kubernetes.

Labels are key/value pairs that are attached to objects, such as pods. Labels are intended to be used to specify identifying attributes of objects that are meaningful and relevant to users, but do not directly imply semantics to the core system. Cluster wide objects are Nodes, persistent volumes, storage class.

Autoscaling Objects

To scale the workload in a Kubernetes cluster, we can use three different Autoscaling mechanisms.

Horizontal Pod Autoscaler (HPA) is the most used autoscaler in Kubernetes. The HPA can watch Deployments or ReplicaSets and increase the number of Replicas if a certain threshold is reached. Imaging your Pod can use 500MiB of memory and you configured a threshold of 80%. If the usage is over 400MiB (80%), a second Pod will get scheduled. Now you have a capacity of 1000MiB. If 800MiB is used, a third Pod will get scheduled and so on.

Cluster Autoscaler : Of course, there is no point in starting more and more Replicas of Pods, if the Cluster capacity is fixed. The Cluster Autoscaler can add new worker nodes to the cluster if the demand increases. The Cluster Autoscaler works great in tandem with the Horizontal Autoscaler.

Vertical Pod Autoscaler: The Vertical Pod Autoscaler is relatively new and allows Pods to increase the resource requests and limits dynamically. As we discussed earlier, vertical scaling is limited by the node capacity.

autoscaling in Kubernetes is NOT available out of the box and requires installing an add-on called metrics-server.

It is possible though to replace the metrics-server with Prometheus Adapter for Kubernetes Metrics APIs. The prometheus-adapter allows you to use custom metrics in Kubernetes and scale up or down based on things like requests or number of users on your system.

KEDA can be used to scale the Kubernetes workload based on events triggered by external systems. KEDA stands for Kubernetes-based Event Driven Autoscaler and was started in 2019 as a partnership between Microsoft and Red Hat. Similar to the HPA, KEDA can scale deployments, ReplicaSets, pods, etc., but also other objects such as Kubernetes jobs. With a large selection of out-of-the-box scalers,

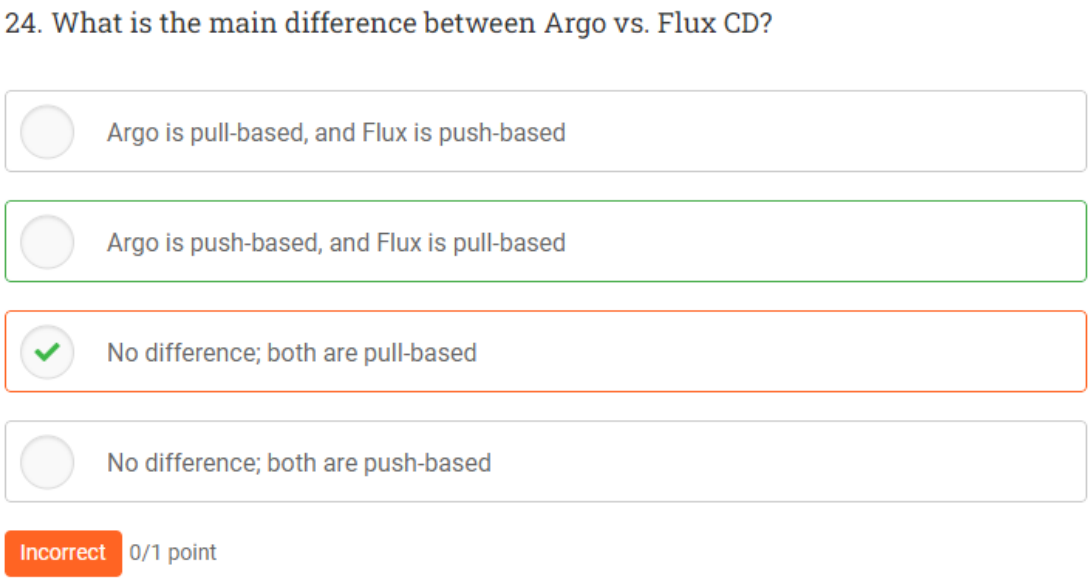
KEDA can scale to special triggers such as a database query or even the number of pods in a Kubernetes cluster.

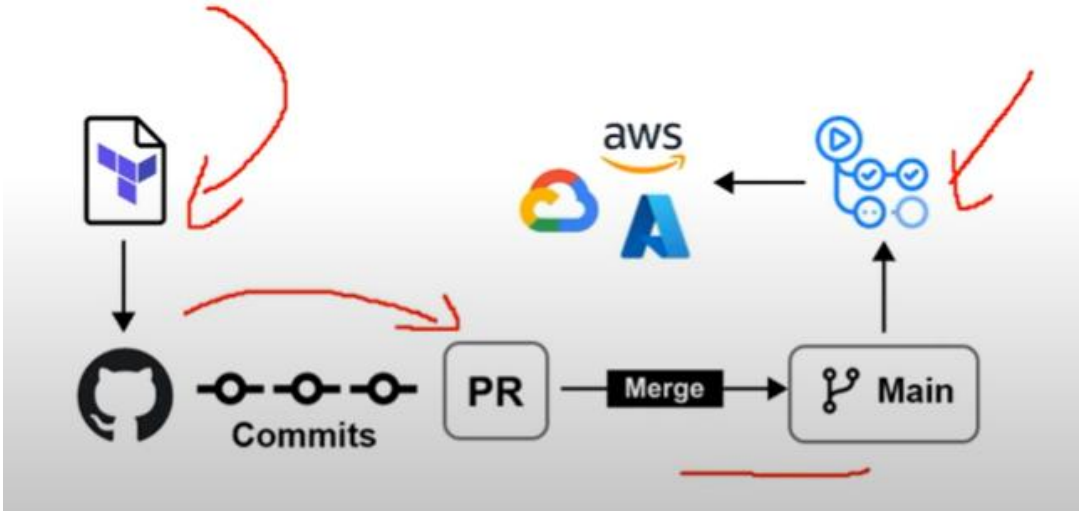
GitOps

GitOps used to manage infrastructure changes.

There are two different approaches how a CI/CD pipeline can implement the changes you want to make:

- **Push-based**
  The pipeline is started and runs tools that make the changes in the platform. Changes can be triggered by a commit or merge request.
- **Pull-based**
  An agent watches the git repository for changes and compares the definition in the repository with the actual running state. If changes are detected, the agent applies the changes to the infrastructure.





Observability

Observability should give answers to questions like:

- Is the system stable or does it change its state when manipulated?
- Is the system sensitive to change, e.g. if some services have high latency?
- Do certain metrics in the system exceed their limits?
- Why does a request to the system fail?
- Are there any bottlenecks in the system?

Telemetry

In container systems, each and every application should have tools built in that generate information data, which is then collected and transferred in a centralized system. The data can be divided into three categories, logs, metrics and tracks.

Logging

Command line tools like **docker**, **kubectl** or **podman** provide a command to show the logs of containerized processes if you let them log directly to the console or to **/dev/stdout** and **/dev/stderr**. Ex. **docker logs nginx, kubectl logs nginx.**

- **Node-level logging**
  The most efficient way to collect logs. An administrator configures a log shipping tool that collects logs and ships them to a central store.
- **Logging via sidecar container**
  The application has a sidecar container that collects the logs and ships them to a central store.
- **Application-level logging**
  The application pushes the logs directly to the central store. While this seems very convenient at first, it requires configuring the logging adapter in every application that runs in a cluster.

There are several tools to choose from to ship and store the logs. The first two methods can be done by tools like fluentd or filebeat. Popular choices to store logs are OpenSearch or Grafana Loki.

Prometheus

Prometheus is an open source monitoring system. To query data that is stored in the time series database, Prometheus provides a querying language called PromQL (Prometheus Query Language). Graphana is data visualization tool used with Prometheus.

Tracing

To understand how a request is processed in a microservice architecture, traces can be of good use. A trace describes the tracking of a request while it passes through the services. A trace consists of multiple units of work which represent the different events that occur while the request is passing the system. Each application can contribute a span to the trace, which can include information like start and finish time, name, tags or a log message. These traces can be stored and analyzed in a tracing system like Jaeger.

Cost Management

Identify wasted and unused resources
Right-Sizing
Reserved Instances
Reserved Instances

Authentication of kubectl with cluster

By default kubectl will first determine if it is running within a pod, and thus in a cluster. It starts by checking for the KUBERNETES_SERVICE_HOST and KUBERNETES_SERVICE_PORT environment variables and the existence of a service account token file at /var/run/secrets/kubernetes.io/serviceaccount/token. If all three are found in-cluster authentication is assumed.

Kubectl Common Operations:

Kubectl has lot of operations like get, expose, config, auth, attach, certificate, cp, diff etc.
Kubectl supports resources like nodes, events, pods, services, secrets, deployments etc.
We can format the output using kubectl using option -o

Kubectl create -f filename --- create a resource form file or stdin
There are many sub actions for create like clusterrole, clusterrolebinding, configmap, cronjob, ingress, deployment, job, namespace, priorityclass, quota, role, secrets, service, token etc.
Examples of important operations are as below

kubectl create deployment NAME --image=image -- [COMMAND] [args...]
kubectl create clusterip NAME [--tcp=<port>:<targetPort>] [--dry-run=server|client|none]

kubectl get -o json pod web-pod-13je7 --- Display one or many resources. Prints a table of the most important information about the specified resources. You can filter the list using a label selector and the --selector flag

Kubectl run nginx –image=nginx --- Create and run a particular image in a pod.

kubectl expose rc nginx --port=80 --target-port=8000 - Expose a resource as a new Kubernetes service. Looks up a deployment, service, replica set, replication controller or pod by name and uses the selector for that resource as the selector for a new service on the specified port

kubectl delete -f ./pod.json - Delete resources by file names, stdin, resources and names, or by resources and label selector.

kubectl apply -f ./pod.json - Apply a configuration to a resource by file name or stdin
Kubectl annotate - Update the annotations (descriptions or explanations) on one or more resources.

kubectl autoscale deployment foo --min=2 --max=10 --- Creates an autoscaler that automatically chooses and sets the number of pods that run in a Kubernetes cluster.

kubectl debug mypod -it --image=busybox - Debug cluster resources using interactive debuggin g containers.

kubectl diff -f pod.json --- Diff configurations specified by file name or stdin between the curren t online configuration, and the configuration as it would be if applied.

kubectl edit svc/registry -- Edit a resource from the default editor.

kubectl label --overwrite pods foo status=unhealthy --- *A label key and value must begin with a l etter or number, and may contain letters, numbers, hyphens, dots, and underscores, up to 63 ch aracters each.*

kubectl patch node k8s-node-1 -p '{"spec":{"unschedulable":true}}' - Update fields of a resource u sing strategic merge patch, a JSON merge patch, or a JSON patch.JSON and YAML formats are ac cepted.

kubectl replace -f ./pod.json -- Replace a resource by file name or stdin.JSON and YAML formats a re accepted.

Kubectl rollout history/pause/restart/status/resume/undo Manage the rollout of one or many resource s.Valid resource types include:*deployments,* daemonsets, statefulsets

kubectl scale **--replicas**=3 rs/foo ---- Set a new size for a deployment, replica set, replication controller, or stateful set.

kubectl set env/image/resources/selector/serviceaccount/subject ---- Configure application resources.These commands help you make changes to existing application resources.

Kubectl attach --- Attach to a process that is already running inside an existing container.

Kubectl auth can-i(Check whether an action is allowed)/reconcile(Reconciles rules for RBAC role, role binding, cluster role, and cluster role binding objects) ---- Copy files and directories to and from containers.

Kubectl cp src dest -- Copy files and directories to and from containers.

Kubectl describe - Show details of a specific resource or group of resources

Kubectl exec – execute a command in container

Kubectl logs --- Print the logs for a container in a pod or specified resource. If the pod has only one container, the container name is optional.

Kubectl port-forward --- Forward one or more local ports to a pod

Kubectl proxy --- Creates a proxy server or application-level gateway between localhost and the Kubernetes API server

Kubectl top node/pod -- The top command allows you to see the resource consumption for nodes or pods

Kubectl api-versions ---- Print the supported API versions on the server, in the form of "group/version".

Kubectl certificate approve/deny ---- Modify certificate resources.

Kubectl cluste-info ---- Display addresses of the control plane and services with label kubernetes.io/cluster-service=true.

Kubectl cordon --- Mark node as unschedulable.

Kubectl drain --- Drain node in preparation for maintenance.The given node will be marked unschedulable to prevent new pods from arriving

Kubectl config set/set-cluster/get-user/view/delete-user/delete-cluster/current-context etc ---- Modify kubeconfig files using subcommands like "kubectl config set current-context my-context"

Kubectl explain - List the fields for supported resources.This command describes the fields associated with each supported API resource. Fields are identified via a simple JSONPath identifier: <type>.<fieldName>[.<fieldName>]

kubectl api-resources **----** Print the supported API resources on the server.

RollingUpdate Deployments support running multiple versions of an application at the same time.

Kubernetes objects are RESTful objects

This volume type can be used to share contents within containers in a pod, but will not persist b eyond the life of a pod is called EmptyDir

An abstraction in kubernetes which defines a logical set of pods and a policy to access them is c alled Service

Kube-apiserver on kubernetes master is designed to scale Horizontally

Cronjobs in kubernetes run in UTC only

PersistentVolumes and Nodes does not associate with any namespace.

Which 2 steps does the scheduler perform to select a node for a pod – Scoring and Filtering

Which authentication method allows JWTs to authenticate? --- OpenId connect

In distributed system tracing, is the term used to refer to a request as it passes through a single com-ponent of the distributed system? ---- Span

Cgroups allows limitations like Prioritization, Resource Limiting, Accounting, and Control

Which of the following provides cloud-native storage orchestration?

- **Cloud Provider Specific storage (EBS, EFS, Cloud Storage)** correct
- Cloud Storage
- Storage IO

Which is not a service type in Kubernetes?

- ClusterIP
- NodePort
- **Ingress** correct
- LoadBalancer
- ExternalName

Containerization	Docker, gvisor, kata containers
CI/CD	Agro (Kunernetese-native tool)
Packaging	Helm
Monitoring	Prometheus

Logging	Fluentd, Graphana loki, opensearch, file beat
Tracing	Jaeger
Service Discovery	CoreDNS
Service mesh	Envoy, Linkered, Istio
Database and storage	Vitess, Rook, TiKV
Streaming and messaging	gRPC, NATS, Cloudevents
Container Registry	Cri-o, containered-d, Harbor
Container Runtime reference Implementation (by OCI)	runC
Podman	Similar
Public Imahe registry	Docker hub, Quay
Networking for Kubernetese	Cilium, Weave, project calico
GitOps pull based approach	Flux and ArgoCD
Data Visualization	Graphana

SRE uses 3 main metrics

Service Level Objectives (SLO)

Service Level Indicators (SLI)

Service Level Agreements (SLA)

Kops – Kubernetese operations

Namespace-based scoping is applicable only for namespaced objects *(e.g. Deployments, Service s, etc)* and not for cluster-wide objects *(e.g. StorageClass, Nodes, PersistentVolumes, etc)*

Management of multiple resources can be simplified by grouping them together in the same fil e (separated by --- in YAML).

assume there is a directory project/k8s/development that holds all of the manifests needed for the development environment, organized by resource type:

```
project/k8s/development
├── configmap
│   └── my-configmap.yaml
├── deployment
│   └── my-deployment.yaml
└── pvc
    └── my-pvc.yaml
```

performing a bulk operation on project/k8s/development can be done by specifying the --recursive or -R flag with the --filename,-f by giving all the file names

Canary Deployment

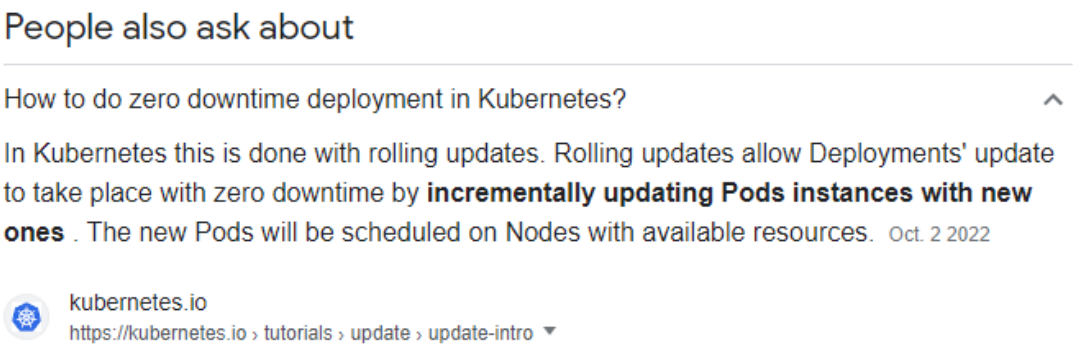
multiple labels are needed is to distinguish deployments of different releases or configurations of the same component. It is common practice to deploy a *canary* of a new application release (specified via image tag in the pod template) side by side with the previous release so that the new release can receive live production traffic before fully rolling it out.

you can use kubectl apply to push your configuration changes to the cluster.

Blue Green deployment

Blue green deployment is an application release model that gradually transfers user traffic from a previous version of an app or microservice to a nearly identical new release—both of which are running in production

Zero Downtime deployments



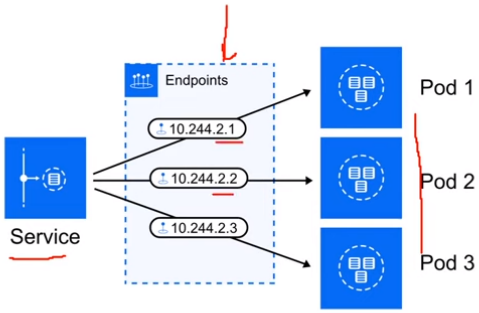
Custom Resources

Kubernetes API has a few built-in objects that it understands like pods, namespaces, ConfigMaps, services or nodes are all API objects. *Custom resources* are extensions of the Kubernetes API. We can create one or use existing many Kubernetes tools will install their own CRDs, so even if you don't create any yourself, you'll probably still end up having some on your cluster.One example is cert-manager, a very popular Kubernetes tool for managing certificates. It installs a few CRDs on your cluster in order to do its job. If you execute kubectl get clusterissuers you get all the certs

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.



Eg. Back up the database every 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"
```

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.

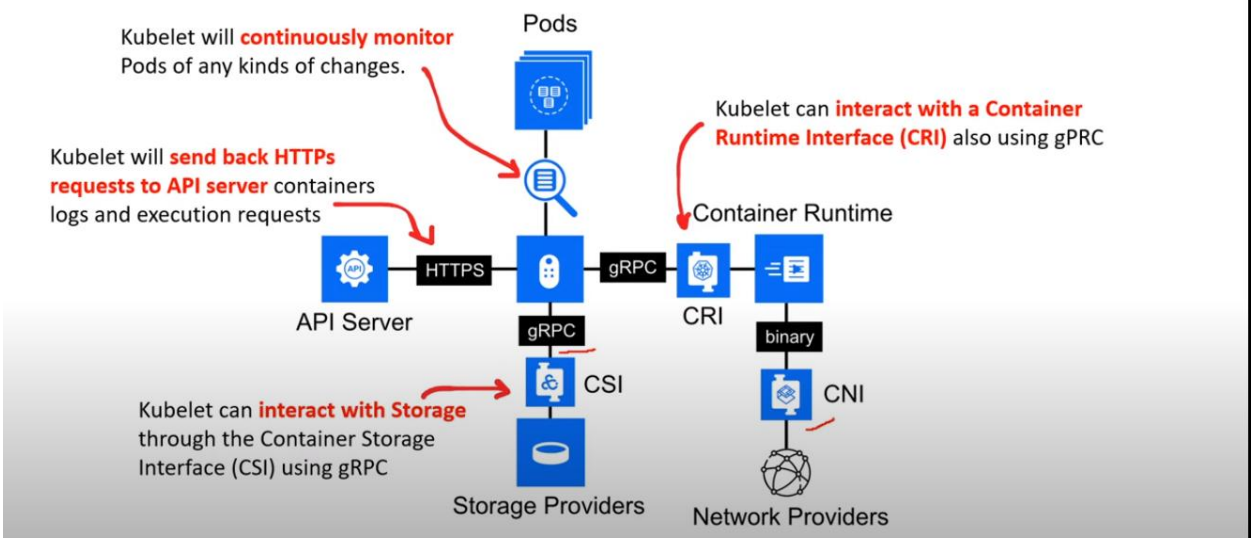
Ingress often use annotation to communicate to Ingress Controllers.

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
```

Kubelet

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



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.

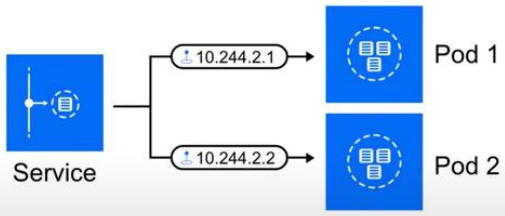
Volume Snapshot
Archiving a volume configuration and its data for rollbacks or backup

Kubernetes Services

Search

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

05:37

```
kubectl get services                    # List all services in the namespace
kubectl get pods --all-namespaces       # List all pods in all namespaces
kubectl get pods -o wide                # List all pods in the current namespace, with more details
kubectl get deployment my-dep           # List a particular deployment
kubectl get pods                        # List all pods in the namespace
kubectl get pod my-pod -o yaml          # Get a pod's YAML
```

Netfilter

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

netfilter
firewalling, NAT, and packet mangling for linux

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

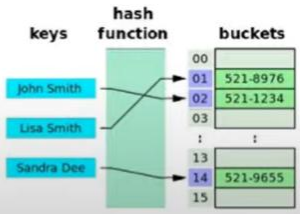
IPVS

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.

Kubernetes Proxy

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

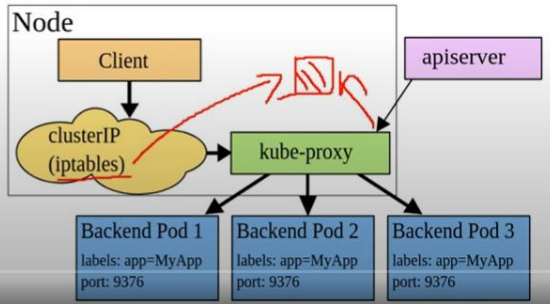
kube-proxy is a network proxy that **runs on each node** in your cluster.
It is designed to load **balance traffic to pods**.

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



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

Ingress/Egress From Internet to Cluster

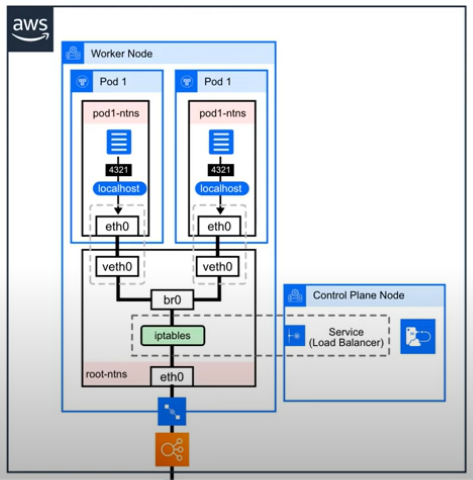
Egress

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)



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

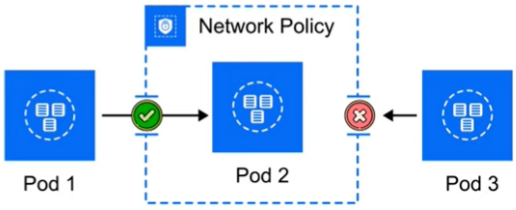
```
kube-apiserver --authorization-mode=RBAC
```

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.**

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…

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

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

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

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.

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

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

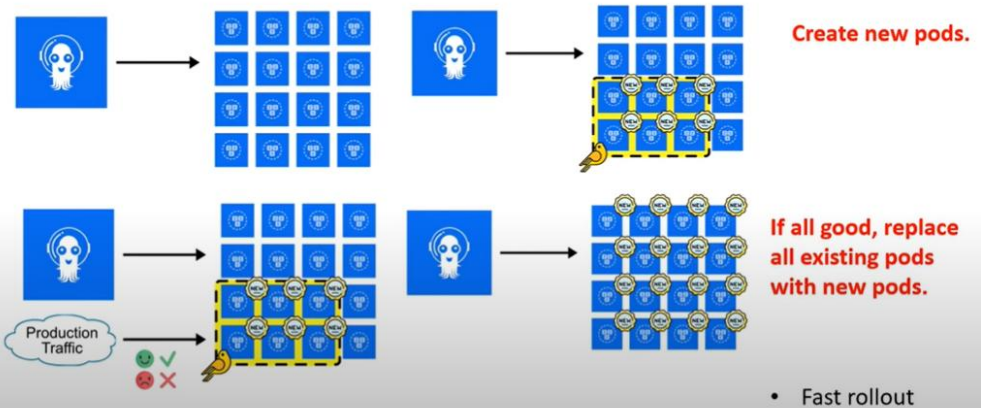
maxUnavailable :
- The amount of pods that can be unavailable

```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
```

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



- Fast rollout

Kubectl Scale vs AutoScale

Scale in deployment tells how many pods should be always running to ensure proper working of the application. You have to specify it manually. In YAMLs you have to define it in spec.replicas like in example below:

```
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:1.7.9
        ports:
        - containerPort: 80
```

Second way to specify scale (replicas) of deployment is use command.

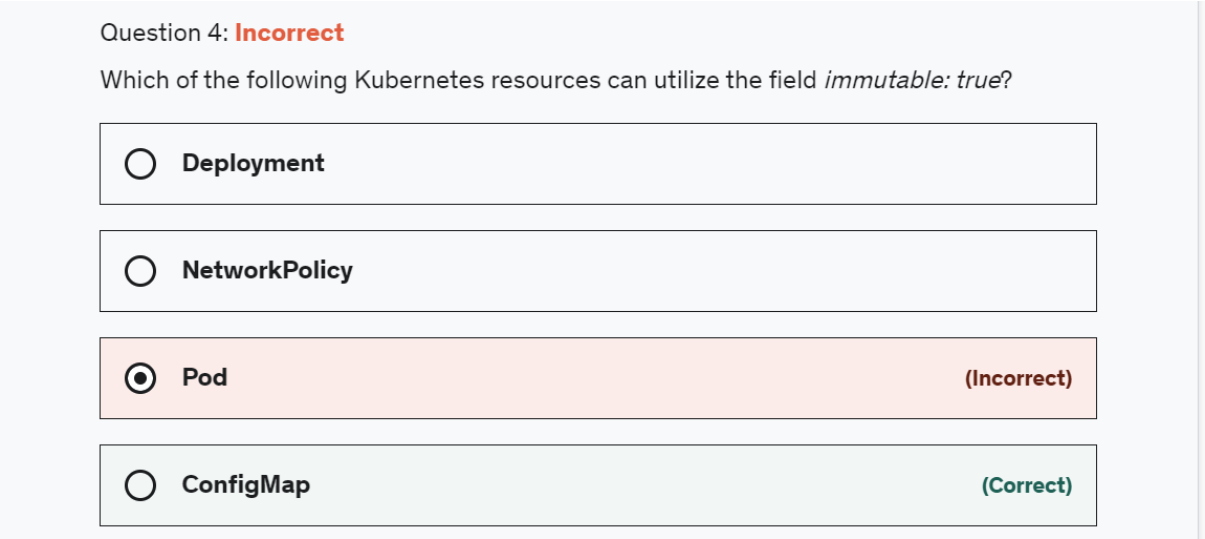
```
$ kubectl run nginx --image=nginx --replicas=3
deployment.apps/nginx created
```

```
$ kubectl get deployment
NAME   DESIRED  CURRENT  UP-TO-DATE  AVAILABLE  AGE
nginx  3        3        3           3          11s
```

Horizontal Pod Autoscaling (HPA) was invented to scale deployment based on metrics produced by pods. For example, if your application have about 300 HTTP request per minute and each your pod allows to 100 HTTP requests for minute it will be ok. However if you will receive a huge amount of HTTP request ~ 1000, 3 pods will not be enough and 70% of request will fail. When you will use HPA, deployment will autoscale to run 10 pods to handle all requests. After some time, when number of request will drop to 500/minute it will scale down to 5 pods. Later depends on request number it might go up or down depends on your HPA configuration.
Easiest way to apply autoscale is:

```
$ kubectl autoscale deployment <your-deployment> --<metrics>=value --min=3 --max=10
```

It means that autoscale will automatically scale based on metrics to maximum 10 pods and later it will downscale minimum to 3. Very good example is shown at [HPA documentation] with CPU usage.



Which Prometheus metric type represents a single numerical value that can increase and decrease over time?

- (●) **Gauge** (Correct)
- ( ) **Summary**
- ( ) **Histogram**
- ( ) **Counter**

What is the name of a service that has no *clusterIp* address?

- ( ) **ClusterIP**
- (●) **Headless** (Correct)
- ( ) **LoadBalancer**
- ( ) **NodePort**

kubectl create deploy nginx-dep --image=nginx:stable --replicas=1

How many Pods will be created?

- ( ) **1** (Correct)
- ( ) **3**
- ( ) **none**
- (●) **2** (Incorrect)

Which deployment strategy can you use with Kubernetes Deployments to deploy new code with zero downtime?

- (●) **BlueGreen** (Incorrect)
- ( ) **RollingUpdate** (Correct)
- ( ) **Recreate**
- ( ) **ZeroDowntime**

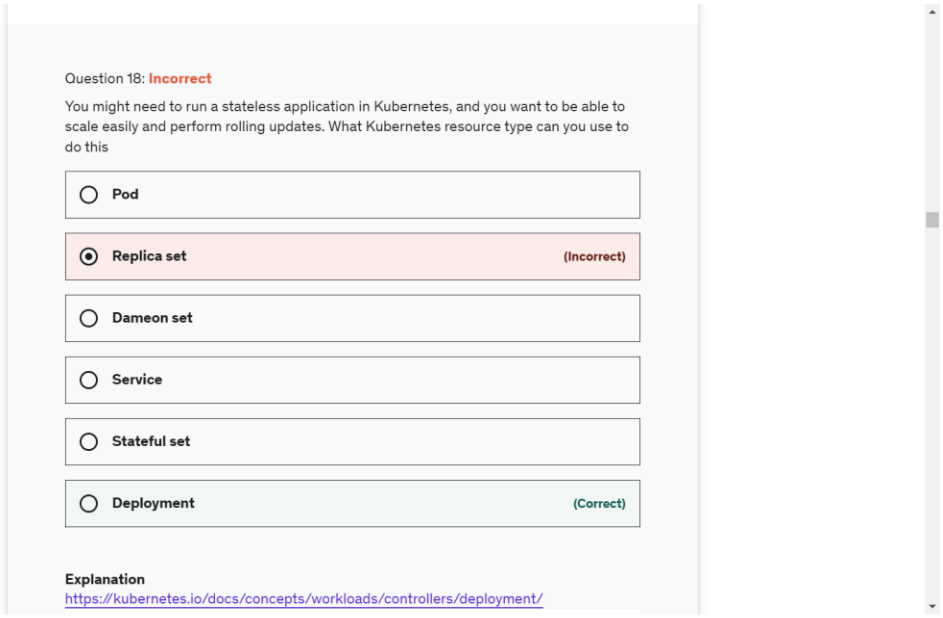
Which authentication method allows JWTs to authenticate?

- ( ) **Anonymous**
- ( ) **OPA gatekeeper**
- ( ) **Client 'TLS' certificates**
- (●) **OpenId connect** (Correct)

Which resource would allow you to manage several replica Pods for a stateless application?

- ( ) **Deployment** (Correct)
- ( ) **StatefulSet**
- (●) **DaemonSet** (Incorrect)
- ( ) **StatelessSet**

Fu



Which of the following statements about Network Policy is true?

- ◯ **When a Pod is isolated, only traffic specifically blocked by the NetworkPolicy will be blocked.**
- ◉ **If a Pod is not selected by any NetworkPolicy, it is isolated by default.** **(Incorrect)**
- ◯ **You can only have one NetworkPolicy per Pod.**
- ◯ **A Pod can be isolated when it comes to incoming traffic and non-isolated for outgoing traffic.** **(Correct)**

Which Kubernetes resource type allows defining which pods are isolated when it comes to networking?

- (●) **Service** (Incorrect)
- ( ) **Role Binding**
- ( ) **Network policy** (Correct)
- ( ) **Domain Name System 'DNS'**

What tool automatically adds and removes Nodes from the cluster as needed?

- ( ) **Cluster Autoscaler** (Correct)
- (●) **Horizontal Pod Autoscaler** (Incorrect)
- ( ) **Kubernetes API Server**
- ( ) **Dynamic Cluster**

What are the two goals of Cloud-Native?

- ○ **Slow innovation and stable applications**
- ◉ **Frequent deployments and well-defined organizational silos** (Incorrect)
- ○ **Rapid innovation and reliability** (Correct)
- ○ **Rapid innovation and automation**

What command is used to get documentation about Kubernetes resource type

- ○ *alias k='kubectl'*
  k api-list
- ◉ *alias k='kubectl'*
  k api-resources (Incorrect)
- ○ *alias k='kubectl'*
  k get resource
- ○ *alias k='kubectl'*
  k explain (Correct)

Which Kubernetes authentication method involves using x509 certificates to authenticate a user?

- ○ **Anonymous**
- ◉ **OpenID Connect** **(Incorrect)**
- ○ **Client Certificates** **(Correct)**
- ○ **Basic**

Have a pod '*hello*' and a container in that pod '*green*'. Which of the following commands would get the logs for that container?

- ○ ***alias k='kubectl'***
  k logs -p hello green
- ○ ***alias k='kubectl'***
  k get logs -p hello -c green
- ○ ***alias k='kubectl'***
  k logs hello -c green **(Correct)**
- ◉ ***alias k='kubectl'***

Which Kubernetes command views all the currently deployed Network Policies?

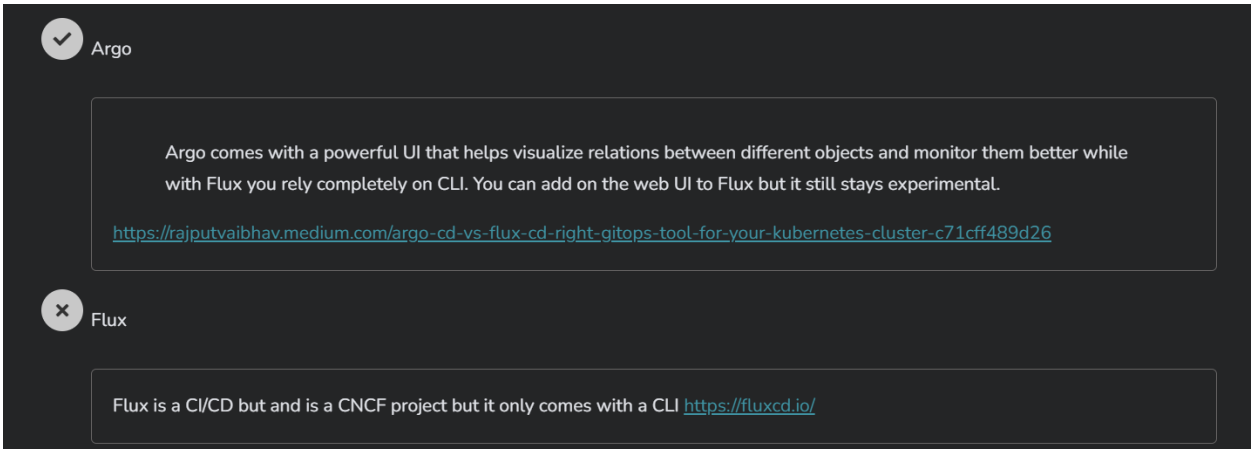
- ○ *kubectl get netpol* (Correct)
- ○ *kubectl view all*
- ○ *kubectl show netpol*
- ◉ *kubectl view netpol* (Incorrect)

Ways to package deploy and manage kubernetese applications – Helm charts and Kubernetese operators

Scheduling refers to making sure that Pods are matched to Nodes so that the kubelet can run them

A to B deployment is when you want to deploy to only a subset of users.

https://www.weave.works/blog/kubernetes-deployment-strategies - read important

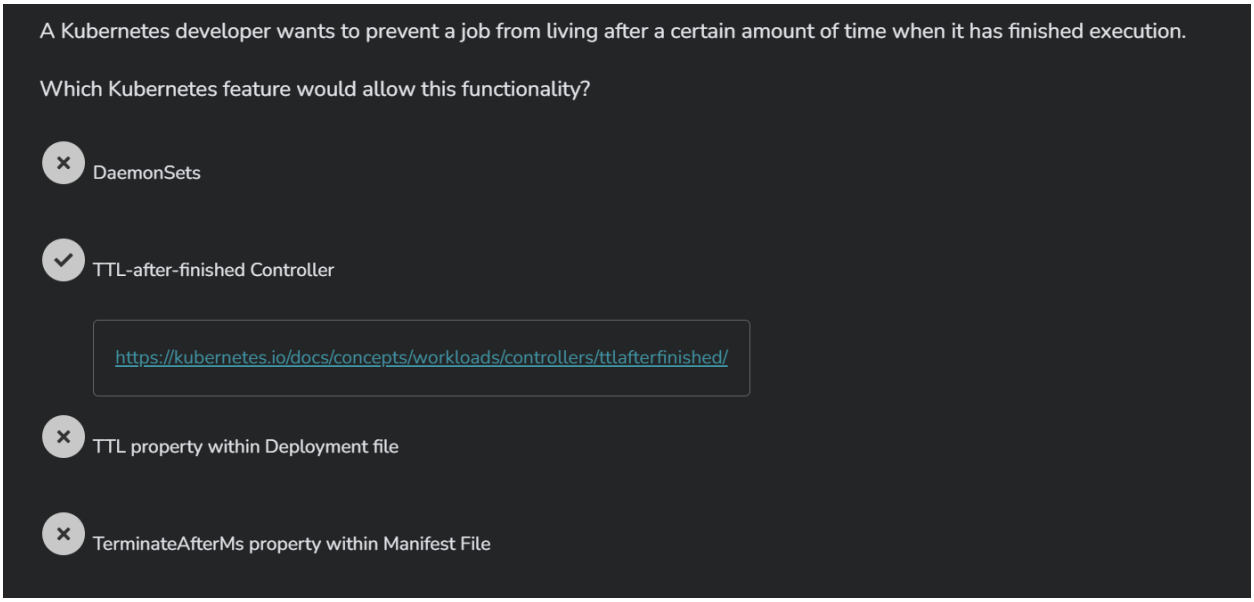


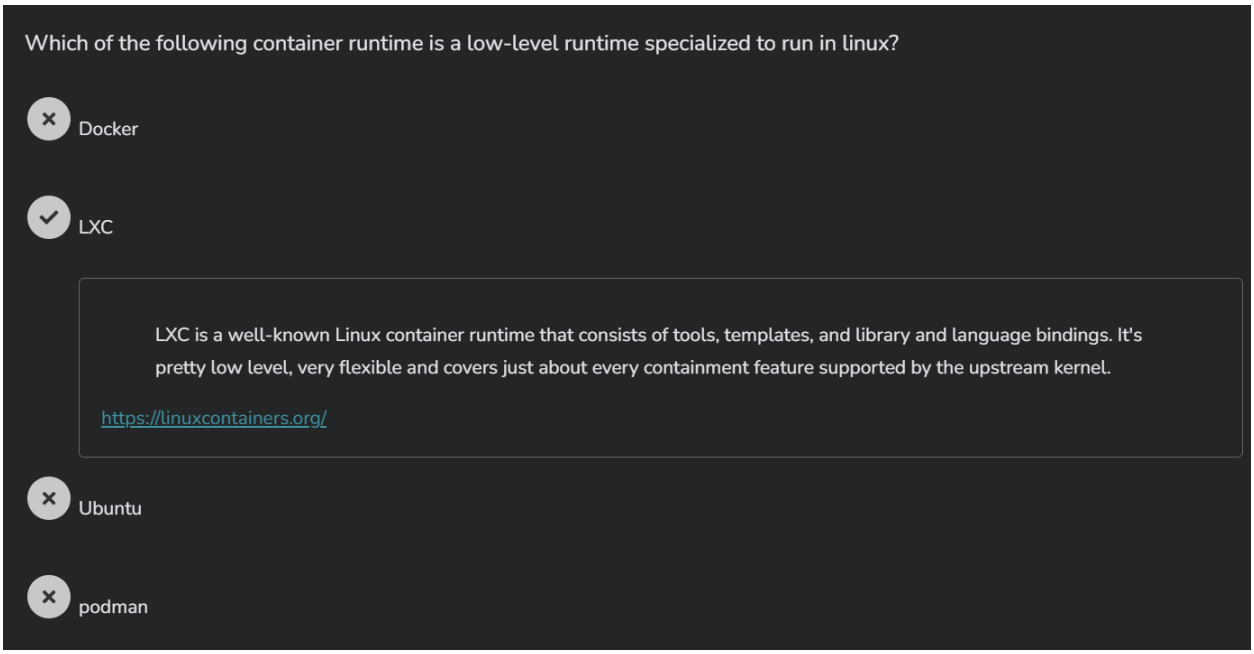
EndpointSlices provide a simple way to track network endpoints within a Kubernetes cluster. They offer a more scalable and extensible alternative to Endpoints.

Fission - fast serverless framework for Kubernetes with a focus on developer productivity and high performance

Many objects in Kubernetes link to each other through owner references. Owner references tell the control plane which objects are dependent on others. Kubernetes uses owner references to give the control plane, and other API clients, the opportunity to clean up related resources before deleting an object. In most cases, Kubernetes manages owner references automatically.

Deployments were designed to replace the legacy ReplicationControllers





When managing the scale of a group of replicas using the HorizontalPodAutoscaler, it is possible that the number of replicas keeps fluctuating frequently due to the dynamic nature of the metrics evaluated. This is sometimes referred to as _____

✗ drifting

✗ slipping

✓ thrashing

> When managing the scale of a group of replicas using the HorizontalPodAutoscaler, it is possible that the number of replicas keep fluctuating frequently due to the dynamic nature of the metrics evaluated. This is sometimes referred to as thrashing, or flapping. It's similar to the concept of hysteresis in cybernetics.
>
> https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/#flapping

What mechanism is used to fetch a group of objects based on a key/value pair in Kubernetes?

✗ Clusters

✓ Selectors

> https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors

✗ Labels

✗ Resource Groups

What authorization modules/modes does Kubernetes support? (Select 3)

- ✗ OpenID
- ✗ OAuth
- ✗ Risked-based policy roles (RBRC)
- ✓ Attribute-based access control (ABAC)
- ✓ Role-based access controls (RBAC)
- ✓ Webhooks

Which lightweight Kubernetes distribution is built for IoT and Edge Computing?

- ✗ K3d
- ✓ K3s
- ✗ Minikube
- ✗ Kind

To list the available API groups and their versions you can run kubectl with the “api-versions” option:

```
kubectl api-versions |more
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
...
```

Which deployment strategy offers zero-downtime and instant rollback?

✕ A/B Testing

rlease a new version to a subset of users in a precise way (HTTP headers, cookie, weight, etc.). This doesn’t come out of the box with Kubernetes, it imply extra work to setup a smarter loadbalancing system

✓ Blue/Green

Blue Green is when you deploy an exact duplicate of your workload. You can easily switch back and forth between the Blue and Green environment and terminate the old one when you are happy with the state of the new environment.

✕ Canary

Canary is when you deploy your changes and only roll it out for a subset of your users. If there are no reported problems then all users are shifted over to the new environment and the old environment is terminated.

In most cases metrics are available on _____ endpoint of the HTTP server

✗ /apiserver

This is made up

✗ /events

✗ /apiserver/metrics

This is made up

✗ /kubernetes/events

✓ /metrics

Which mechanism in Kubernetes allows you to select information for fields such as name, namespace or status of Kubernetes object?

✗ MetaData Selectors

✓ Field Selectors

Field selectors let you select Kubernetes resources based on the value of one or more resource fields.

https://kubernetes.io/docs/concepts/overview/working-with-objects/field-selectors/

✗ Label Selectors

✗ Resource Selectors

What is the successor to PodSecurityPolicies?

- ✗ Kubernetes IAM
- ✗ Policies Controller
- ✓ Admission Controller

Kubernetes offers a built-in Pod Security admission controller, the successor to PodSecurityPolicies. Pod security restrictions are applied at the namespace level when pods are created.

https://kubernetes.io/docs/concepts/security/pod-security-admission/

A Site Reliability Engineer has the following responsibilities:

- ✓ Create runbooks to automate responses to incidents, and apply critical updates to production environments without impacting users.
- ✗ Use open source technology and frameworks to package and deploy applications into clusters. SELECTED

  Developer responsibility.
- ✗ Manage logging, keys, and other Kubernetes configurations. SELECTED

  DevOps or Administrator responsibility.
- ✓ Provide developers access to relevant logs, events, and performance metrics necessary to troubleshoot and quickly resolve problems.
- ✓ Maintain application availability, reliability, and performance from the product user's perspective SELECTED
- ✓ Respond to and recover from various outages in underlying IaaS or CaaS services and prevent security breaches SELECTED

Which of the following container runtime is a low-level runtime specialized to run in linux?

- Docker
- Ubuntu
- LXC — SELECTED

  LXC is a well-known Linux container runtime that consists of tools, templates, and library and language bindings. It's pretty low level, very flexible and covers just about every containment feature supported by the upstream kernel.

  https://linuxcontainers.org/
- podman

The RBAC API declares what four kinds of Kubernetes objects?

- ClusterRole — SELECTED
- User
- Policy
- Permission — SELECTED
- ClusterRoleBinding — SELECTED
- Role
- RoleBinding — SELECTED

Which kubectl command can be used to list all available API groups?

✗ There is no such kubectl command

This is not true.

✗ kubectl api-groups

This is not a real command.

✓ kubectl api-versions SELECTED

To list the available API groups and their versions you can run kubectl with the “api-versions” option:

kubectl api-versions |more
admissionregistration.k8s.io/v1beta1
apiextensions.k8s.io/v1beta1
apiregistration.k8s.io/v1
apiregistration.k8s.io/v1beta1
...

✗ kubectl api-resources

The kubectl “api-resources” is a command to view the available resource types as well as the API group they are associated with.

Which mechanism in Kubernetes cleans up related resources before deleting an object?

- ✗ Garbage Collection
- ✗ Labels
- ✓ Owners SELECTED
- ✗ Tags
- ✗ Selectors

EXPLANATION

Many objects in Kubernetes link to each other through owner references. Owner references tell the control plane which objects are dependent on others. Kubernetes uses owner references to give the control plane, and other API clients, the opportunity to clean up related resources before deleting an object. In most cases, Kubernetes manages owner references automatically.

https://kubernetes.io/docs/concepts/architecture/garbage-collection/#owners-dependents

What are the names of three CNCF landscapes?

- ✓ Serverless landscape
  - https://landscape.cncf.io/images/serverless.png
- ✗ Storage landscape
- ✗ Compute landscape — SELECTED
- ✗ Networking landscape
- ✗ Container landscape
- ✓ Cloud-native landscape — SELECTED
  - https://landscape.cncf.io/images/landscape.png
- ✓ Member landscape — SELECTED
  - https://landscape.cncf.io/images/members.png

EXPLANATION

All three are listed on the landscape page. https://landscape.cncf.io/

Which deployment method rolls out front-end features to a consistent, small subset of users initially to test the effectiveness of a feature?

- red / black deployments
- canary deployments — SELECTED

  Canary will deploy a few pods, and will route some traffic to these pods. If there are no issues than updates are rolled out to the rest of the pods.

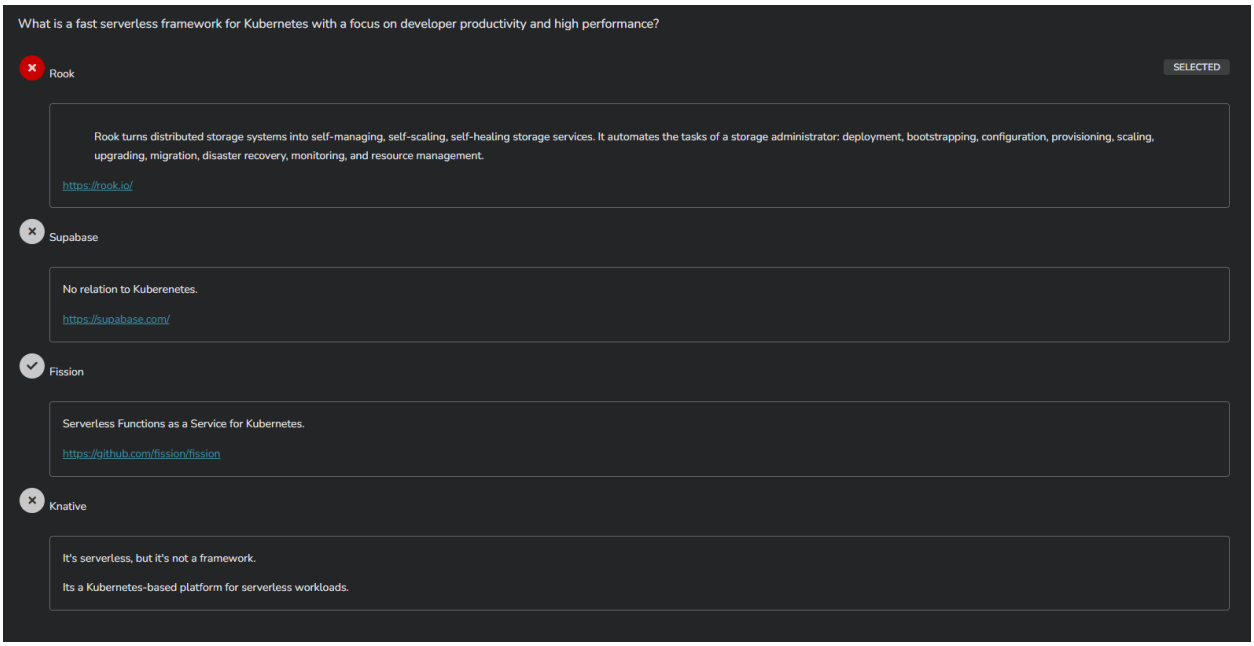
  The reason this is not the answer is the question focuses on front-end features with the purpose of testing out those features.

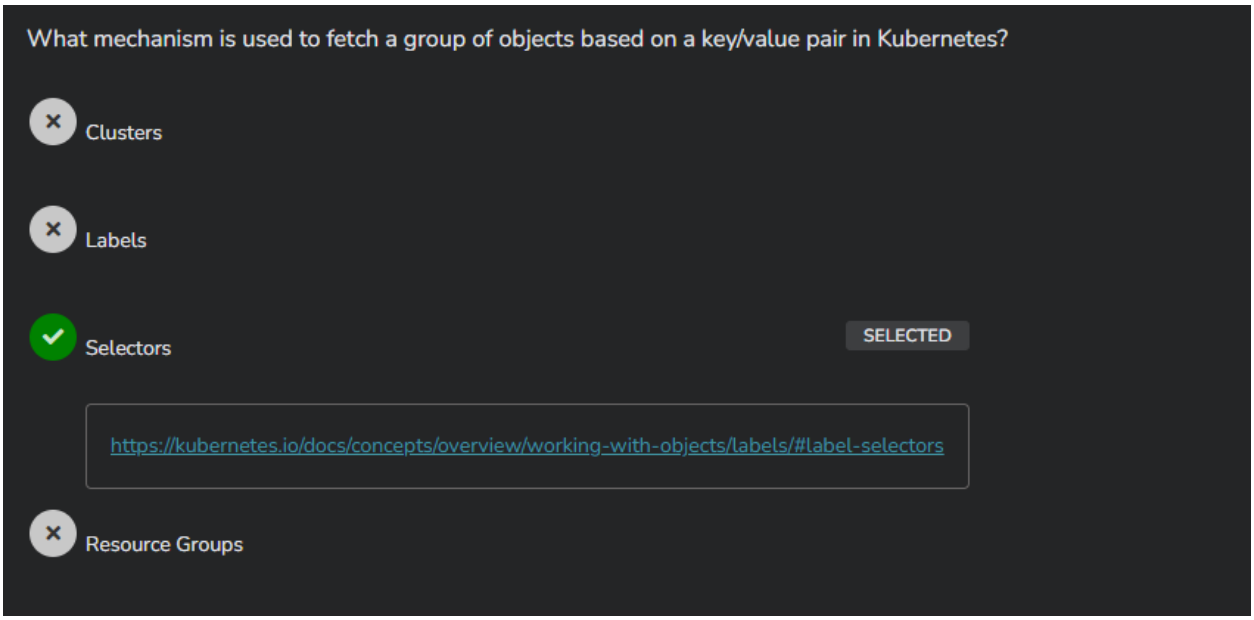
  Canary is a deployment method to mitigate risk not to evaluate features.
- A to B deployments

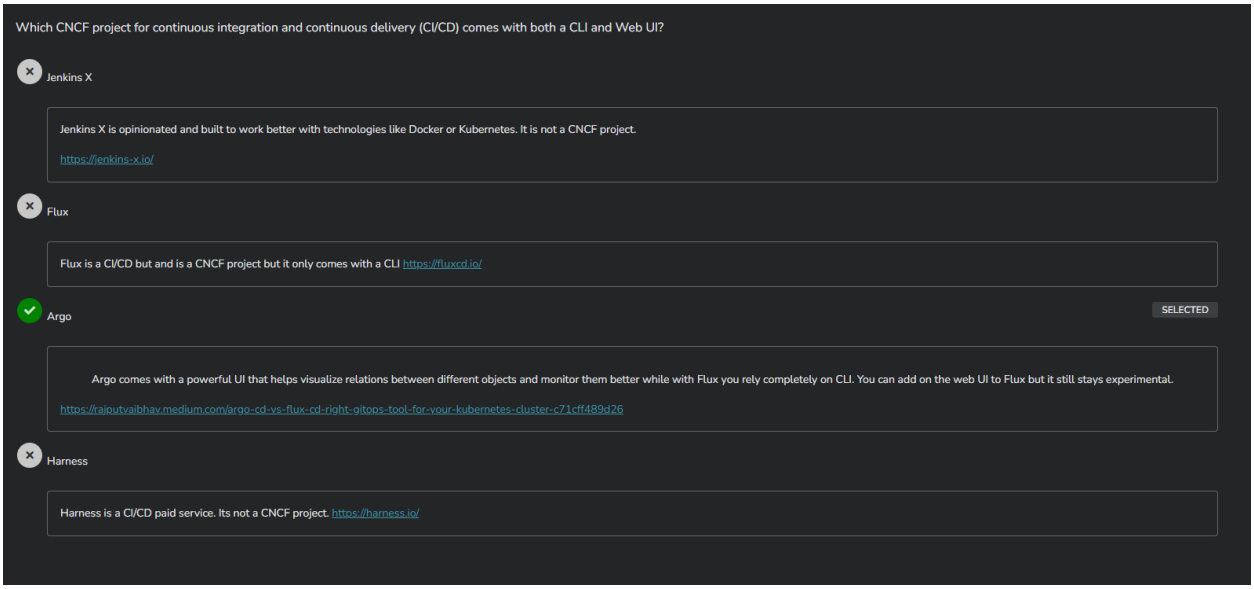
  A to B deployment is when you want to deploy to only a subset of users.
- recreate deployments
- blue / green deployments

Which of the following is NOT a value of Kubernetes Community?

- ✗ Distribution is better than centralization
- ✗ Inclusive is better than exclusive — SELECTED
- ✗ Evolution is better than stagnation
- ✗ Automation over process
- ✓ Rapid-paced Innovation is better than slow and careful
- ✗ Community over product or company


What is a fast serverless framework for Kubernetes with a focus on developer productivity and high performance?
Rook
SELECTED
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.
https://rook.io/
Supabase
No relation to Kuberenetes.
https://supabase.com/
Fission
Serverless Functions as a Service for Kubernetes.
https://github.com/fission/fission
Knative
It's serverless, but it's not a framework.
Its a Kubernetes-based platform for serverless workloads.


What mechanism is used to fetch a group of objects based on a key/value pair in Kubernetes?
Clusters
Labels
Selectors
SELECTED
https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/#label-selectors
Resource Groups



Close

Site Reliability Engineer (SRE)

A role with a stronger definition is the Site Reliability Engineer (SRE). SRE was founded around 2003 at Google and became an important job for many organizations. The overarching goal of SRE is to create and maintain software that is reliable and scalable. To achieve this, software engineering approaches are used to solve operational problems and automate operation tasks.

To measure performance and reliability, SREs use three main metrics:

- **Service Level Objectives (SLO)**: *"Specify a target level for the reliability of your service."* - A goal that is set, for example reaching a service latency of less that 100ms.
- **Service Level Indicators (SLI)**: *"A carefully defined quantitative measure of some aspect of the level of service that is provided"* - For example how long a request actually needs to be answered.
- **Service Level Agreements (SLA)**: *"An explicit or implicit contract with your users that includes consequences of meeting (or missing) the SLOs they contain. The consequences are most easily recognized when they are financial – a rebate or a penalty – but they can take other forms."* - Answers the question what happens if SLOs are not met.

Around these metrics, SREs might define an error budget. An error budget defines the amount (or time) of errors your application can have, before actions are taken, like stopping deployments to production.

The CNCF has a Technical Oversight Committee (TOC) that is responsible for defining and maintaining the technical vision, approving new projects, accepting feedback from the end-user committee, and defining common practices that should be implemented in CNCF projects.

Our graduated projects

VIEW ON CNCF LANDSCAPE

Continuous Integration & Delivery

Container Runtime

Coordination & Service Discovery

Service Proxy

Coordination & Service Discovery

Logging

Continuous Integration & Delivery

Container Registry

Application Definition & Image Build

Tracing

Scheduling & Orchestration

Service Mesh

Security & Compliance

Monitoring

Cloud Native Storage

Key Management

Key Management

Security & Compliance

Database

Database

Cloud Native Architecture Characteristics

- High level of automation — Expand
- Self healing — Expand
- Scalable — Expand
- (Cost-) Efficient — Expand
- Easy to maintain — Expand
- Secure by default — Expand

A good baseline and starting point for your cloud native journey is the twelve-factor app. The twelve factor app is a guideline for developing cloud native applications, which starts with simple things like version control of your codebase, environment-aware configuration and more sophisticated patterns like statelessness and concurrency of your application.

Approaches to Service Discovery

DNS — Close

Modern DNS servers that have a service API can be used to register new services as they are created. This approach is pretty straightforward, as most organizations already have DNS servers with the appropriate capabilities.

Key-Value-Store — Close

Using a strongly consistent datastore especially to store information about services. A lot of systems are able to operate highly available with strong failover mechanisms. Popular choices, especially for clustering, are etcd, Consul or Apache Zookeeper.

How can you show the logs of a previously terminated container named ruby in the web-1 pod?

- kubectl logs web-1 ruby
- kubectl logs -p web-1 ruby
- kubectl web-1 ruby -p
- kubectl logs -p -c ruby web-1

What is a good format for structured logging?

- YAML ✗
- JSON ✓
- XML
- HTML

9 / 10

What kind of software is Prometheus?

- Software to manage containers
- Software to collect and store metrics ✓
- Software to manage virtual machines
- Software to collect and store logs ✗

10 / 10

33. Which of the following is considered as the Kubernetes controller?

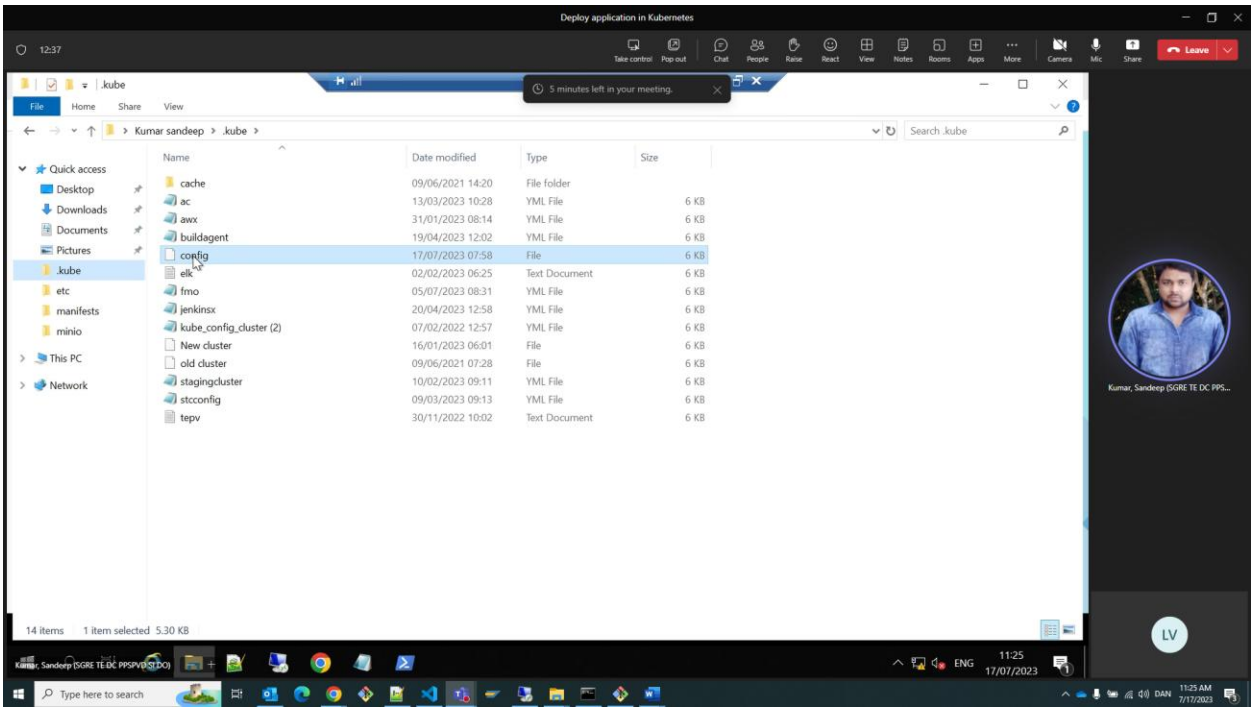
- [x] Namespace
- [ ] Node
- [ ] Replicaset
- [ ] Pod

Incorrect 0/1 point

45. What is the command used to login to the pod?

- [ ] kubectl exec
- [ ] kubectl list
- [x] kubectl login
- [ ] kubectl get

Incorrect 0/1 point


Deploy application in Kubernetes
12:37
Take control
Pop out
Chat
People
Raise
React
View
Notes
Rooms
Apps
More
Camera
Mic
Share
Leave
.kube
File
Home
Share
View
5 minutes left in your meeting.
Kumar sandeep › .kube ›
Search .kube
Quick access
Desktop
Downloads
Documents
Pictures
.kube
etc
manifests
minio
This PC
Network
Name
Date modified
Type
Size
cache 09/06/2021 14:20 File folder
ac 13/03/2023 10:28 YML File 6 KB
awx 31/01/2023 08:14 YML File 6 KB
buildagent 19/04/2023 12:02 YML File 6 KB
config 17/07/2023 07:58 File 6 KB
elk 02/02/2023 06:25 Text Document 6 KB
fmo 05/07/2023 08:31 YML File 6 KB
jenkinsx 20/04/2023 12:58 YML File 6 KB
kube_config_cluster (2) 07/02/2022 12:57 YML File 6 KB
New cluster 16/01/2023 06:01 File 6 KB
old cluster 09/06/2021 07:28 File 6 KB
stagingcluster 10/02/2023 09:11 YML File 6 KB
stcconfig 09/03/2023 09:13 YML File 6 KB
tepv 30/11/2022 10:02 Text Document 6 KB
14 items
1 item selected 5.30 KB
Kumar, Sandeep (SGRE TE DC PPS...
LV
ENG
11:25
17/07/2023
Type here to search
DAN
11:25 AM
7/17/2023