KCNA$^{\text{Q\&As}}$

Kubernetes and Cloud Native Associate (KCNA)

Pass Linux Foundation KCNA Exam with 100% Guarantee

QUESTION 1

What feature is used for selecting the container runtime configuration?

A. RuntimeClass

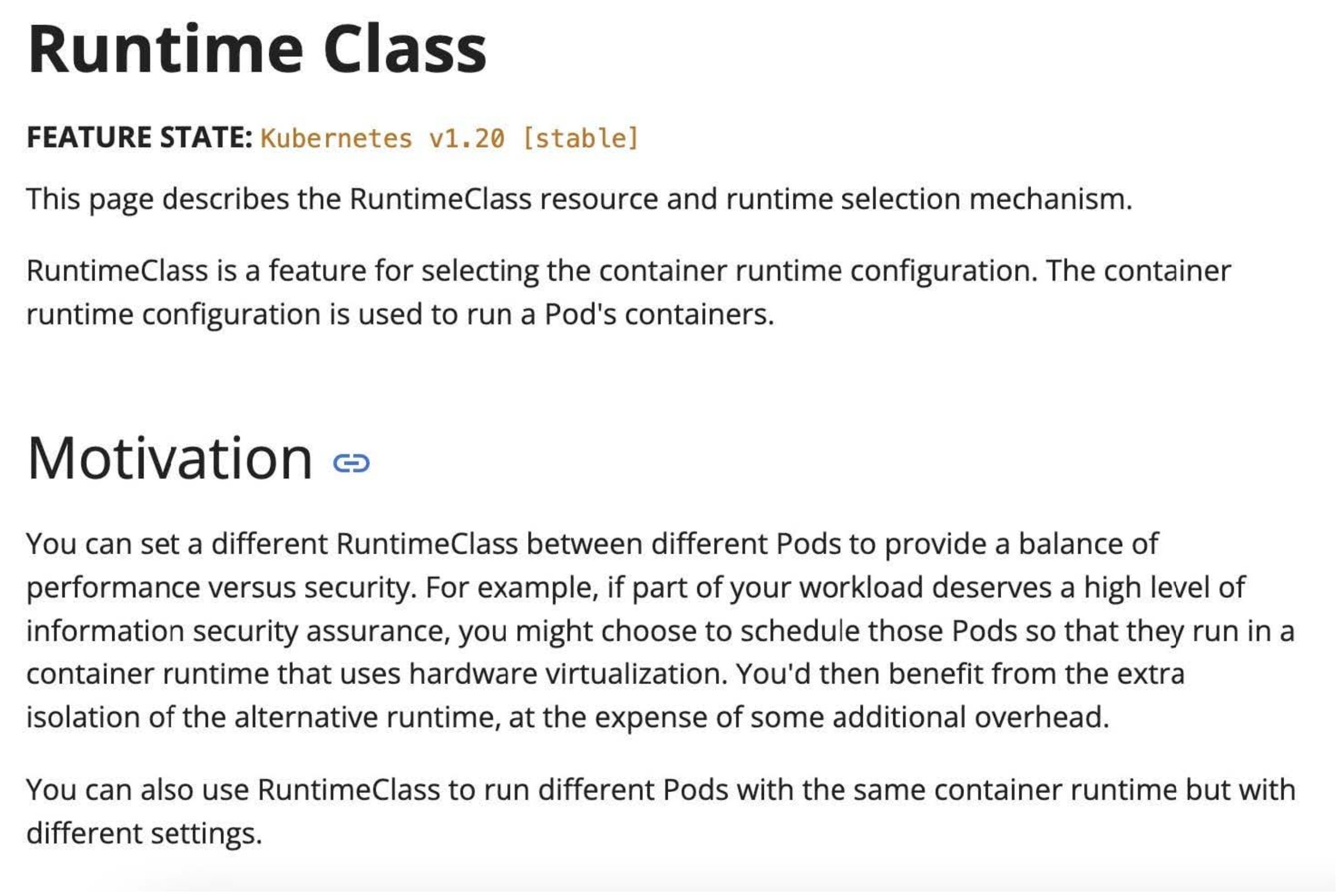
B. RuntimeContainer

C. Runtime

D. RuntimeConfig

Correct Answer: A

Explanation: https://kubernetes.io/docs/concepts/containers/runtime-class/



Runtime Class

FEATURE STATE: `Kubernetes v1.20 [stable]`

This page describes the RuntimeClass resource and runtime selection mechanism.

RuntimeClass is a feature for selecting the container runtime configuration. The container runtime configuration is used to run a Pod's containers.

Motivation

You can set a different RuntimeClass between different Pods to provide a balance of performance versus security. For example, if part of your workload deserves a high level of information security assurance, you might choose to schedule those Pods so that they run in a container runtime that uses hardware virtualization. You'd then benefit from the extra isolation of the alternative runtime, at the expense of some additional overhead.

You can also use RuntimeClass to run different Pods with the same container runtime but with different settings.

QUESTION 2

What is a benefits of Kubernetes federation?

A. Avoids scalability limits on pods and nodes

B. Creates highly available clusters in different regions

C. Low latency

Correct Answer: ABC

QUESTION 3

What is the use of labels in Kubernetes?

A. All of the options

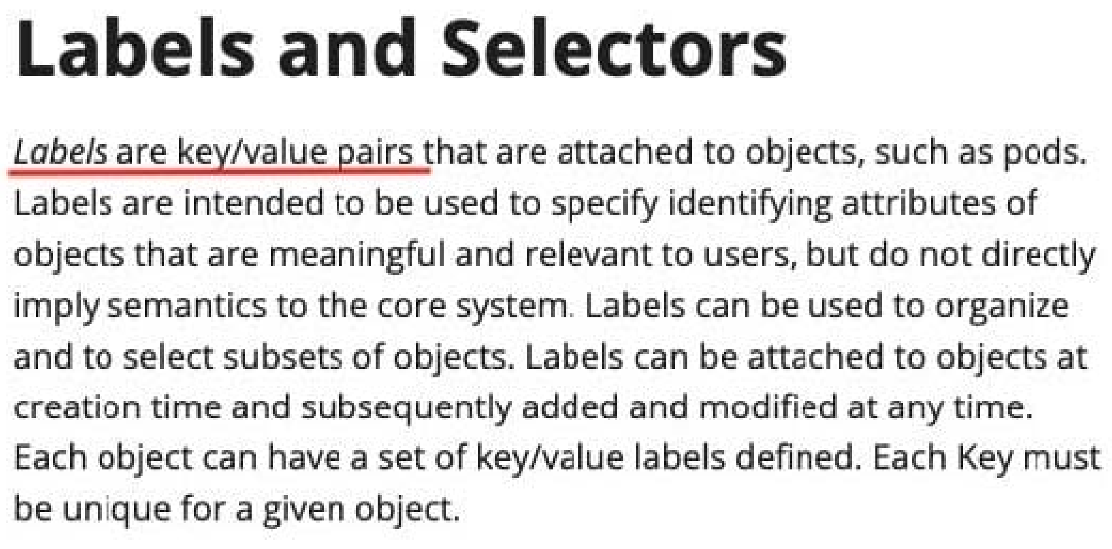
B. It is used to assign annotation to an object

C. It is used to assign key-value pair to an object

D. It is used to assign a name to an object.

Correct Answer: C

Explanation: https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/



Labels and Selectors

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. Labels can be used to organize and to select subsets of objects. Labels can be attached to objects at creation time and subsequently added and modified at any time. Each object can have a set of key/value labels defined. Each Key must be unique for a given object.

QUESTION 4

Which of the following best describes the way K8S Role-based access control (RBAC) works?

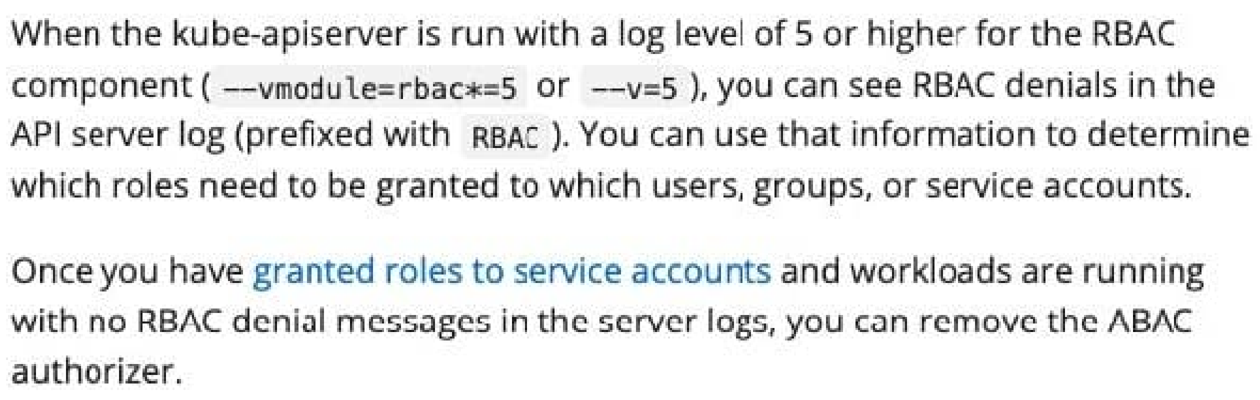
A. K8S does not do RBAC or Cluster role

B. RBAC lists which operations are denied to users

C. States which users can perform which actions against the resources.

Correct Answer: C

Explanation: https://kubernetes.io/docs/reference/access-authn-authz/rbac/


When the kube-apiserver is run with a log level of 5 or higher for the RBAC component (`--vmodule=rbac*=5` or `--v=5`), you can see RBAC denials in the API server log (prefixed with `RBAC`). You can use that information to determine which roles need to be granted to which users, groups, or service accounts.

Once you have granted roles to service accounts and workloads are running with no RBAC denial messages in the server logs, you can remove the ABAC authorizer.

QUESTION 5

What is the name for the tool that manages communication between pods, injects a sidecar proxy container into each pod and directs network traffic through the proxy container?

A. namespace

B. Deployment

C. Network policy

D. Service mesh

E. Service

Correct Answer: D

QUESTION 6

Which project in this list is a leading project in the observability space?

A. Jaeger

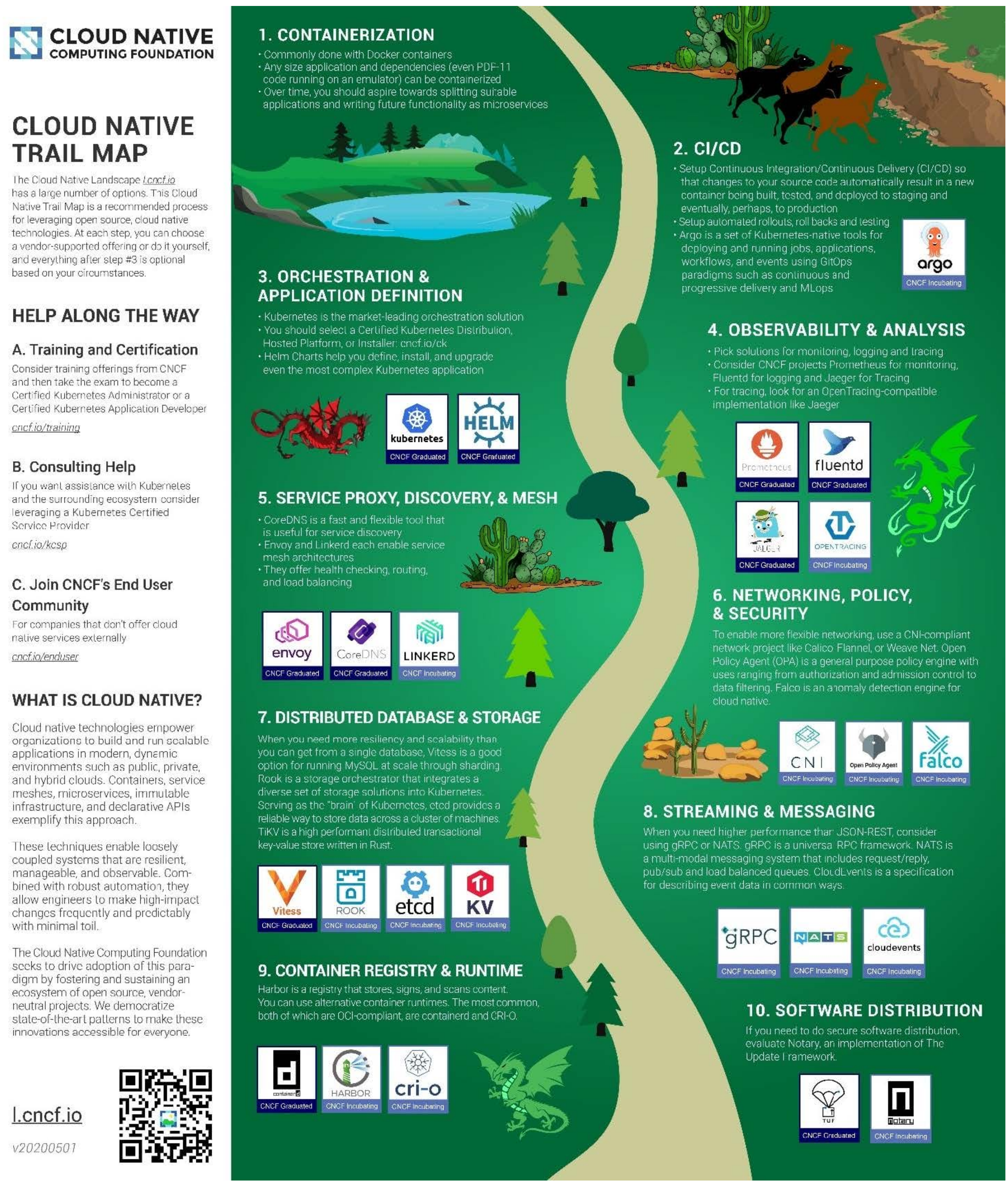
B. Vitess

C. Argo

D. Kubernetes

Correct Answer: A

Explanation: https://github.com/cncf/landscape#trail-map


CLOUD NATIVE COMPUTING FOUNDATION

CLOUD NATIVE TRAIL MAP

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

HELP ALONG THE WAY

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

B. Consulting Help
If you want assistance with Kubernetes and the surrounding ecosystem consider leveraging a Kubernetes Certified Service Provider
cncf.io/kcsp

C. Join CNCF's End User Community
For companies that don't offer cloud native services externally
cncf.io/enduser

WHAT IS CLOUD NATIVE?

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.

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 toil.

The Cloud Native Computing 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.

l.cncf.io
v20200501

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

2. CI/CD
- Setup Continuous Integration/Continuous Delivery (CI/CD) so that changes to your source code automatically result in a new container being built, tested, and deployed to staging and eventually, perhaps, to production
- Setup automated rollouts, roll backs and testing
- Argo is a set of Kubernetes-native tools for deploying and running jobs, applications, workflows, and events using GitOps paradigms such as continuous and progressive delivery and MLops

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

4. OBSERVABILITY & ANALYSIS
- Pick solutions for monitoring, logging and tracing
- Consider CNCF projects Prometheus for monitoring, Fluentd for logging and Jaeger for Tracing
- For tracing, look for an OpenTracing-compatible implementation like Jaeger

5. SERVICE PROXY, DISCOVERY, & MESH
- CoreDNS is a fast and flexible tool that is useful for service discovery
- Envoy and Linkerd each enable service mesh architectures
- They offer health checking, routing, and load balancing

6. NETWORKING, POLICY, & SECURITY
To enable more flexible networking, use a CNI-compliant network project like Calico, Flannel, or Weave Net. Open Policy Agent (OPA) is a general purpose policy engine with uses ranging from authorization and admission control to data filtering. Falco is an anomaly detection engine for cloud native.

7. DISTRIBUTED DATABASE & STORAGE
When you need more resiliency and scalability than you can get from a single database, Vitess is a good option for running MySQL at scale through sharding. Rook is a storage orchestrator that integrates a diverse set of storage solutions into Kubernetes. Serving as the "brain" of Kubernetes, etcd provides a reliable way to store data across a cluster of machines. TiKV is a high performant distributed transactional key-value store written in Rust.

8. STREAMING & MESSAGING
When you need higher performance than JSON-REST, consider using gRPC or NATS. gRPC is a universal RPC framework. NATS is a multi-modal messaging system that includes request/reply, pub/sub and load balanced queues. CloudEvents is a specification for describing event data in common ways.

9. CONTAINER REGISTRY & RUNTIME
Harbor is a registry that stores, signs, and scans content. You can use alternative container runtimes. The most common, both of which are OCI-compliant, are containerd and CRI-O.

10. SOFTWARE DISTRIBUTION
If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.

QUESTION 7

What tool allows us to build useful visual representations of prometheus data?

A. Grafana

B. kubectl

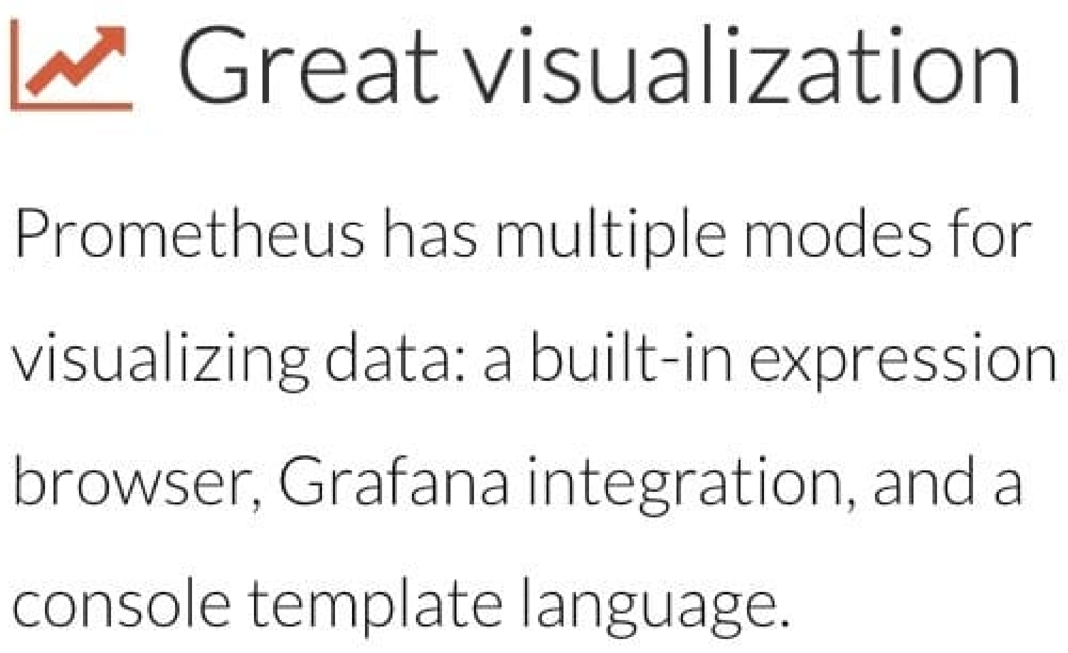
C. Distributed system tracing

D. Rook

E. Kibana

Correct Answer: A

Explanation: https://prometheus.io/


Great visualization

Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

QUESTION 8

A ________ is a ready-to-run software package, containing everything needed to run an application.

A. Container Repository

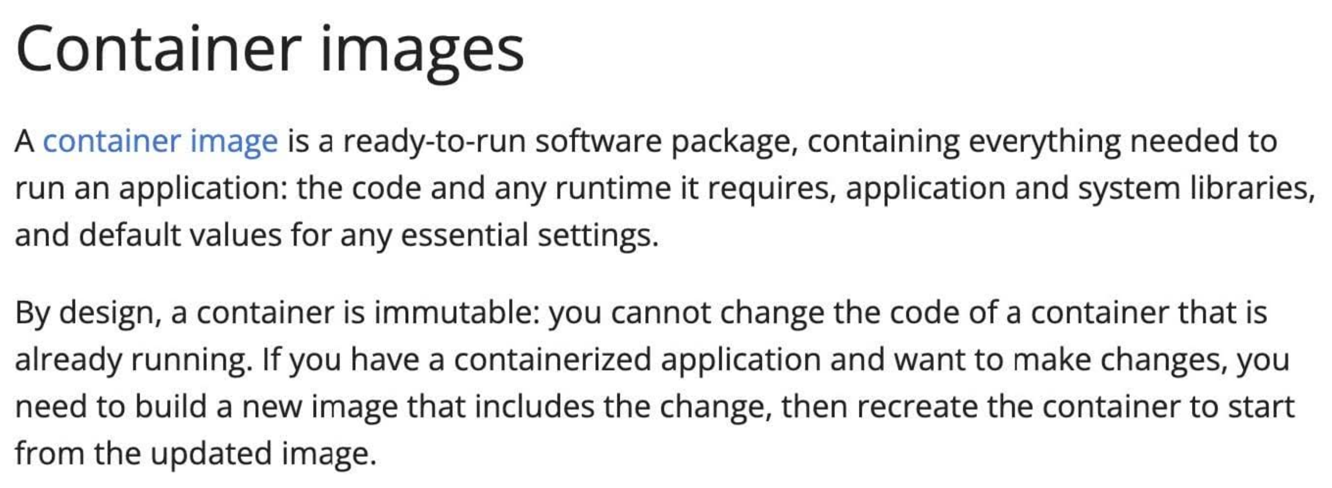
B. Container Runtime

C. Docker

D. Container Image

Correct Answer: D

Explanation: https://kubernetes.io/docs/concepts/containers/#container-images


Container images

A container image is a ready-to-run software package, containing everything needed to run an application: the code and any runtime it requires, application and system libraries, and default values for any essential settings.

By design, a container is immutable: you cannot change the code of a container that is already running. If you have a containerized application and want to make changes, you need to build a new image that includes the change, then recreate the container to start from the updated image.

QUESTION 9

What is the command to list all the available objects in your Kubernetes cluster?

A. kubectl get all

B. kubectl get api-resources

C. kubectl api-resources

D. kubectl get pods

Correct Answer: C

Explanation: https://kubernetes.io/docs/reference/kubectl/cheatsheet/

Resource types

List all supported resource types along with their shortnames, API group, whether they are namespaced, and Kind:

```
kubectl api-resources
```

QUESTION 10

Which control plane component is responsible for scheduling pods?

A. kube-proxy

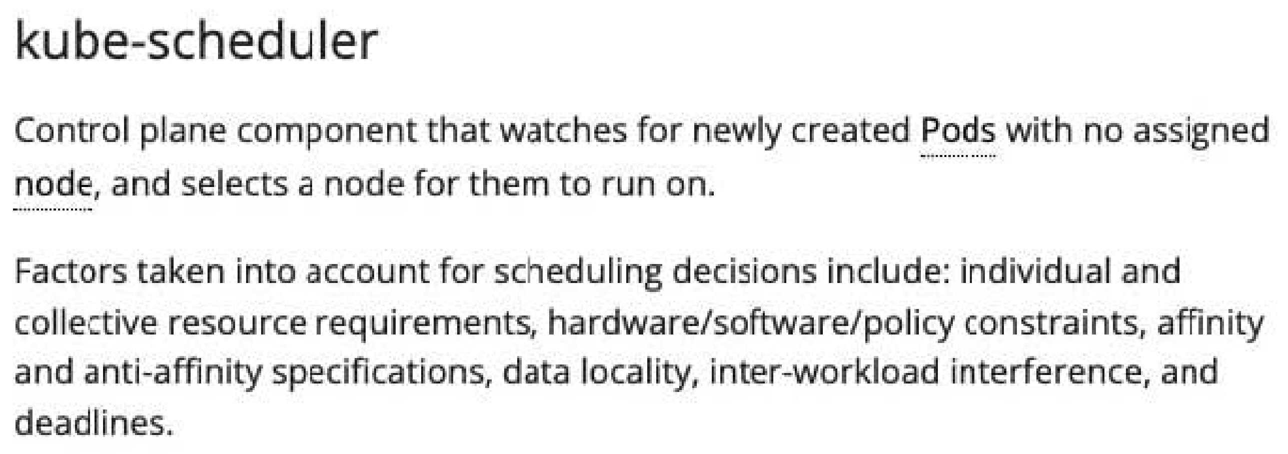
B. kube scheduler

C. kubelet

D. kube api-server

Correct Answer: B

Explanation: https://kubernetes.io/docs/concepts/overview/components/



kube-scheduler

Control plane component that watches for newly created Pods with no assigned node, and selects a node for them to run on.

Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

QUESTION 11

Which of the following components is part of the Kubernetes control panel

A. kubectl

B. kube-proxy

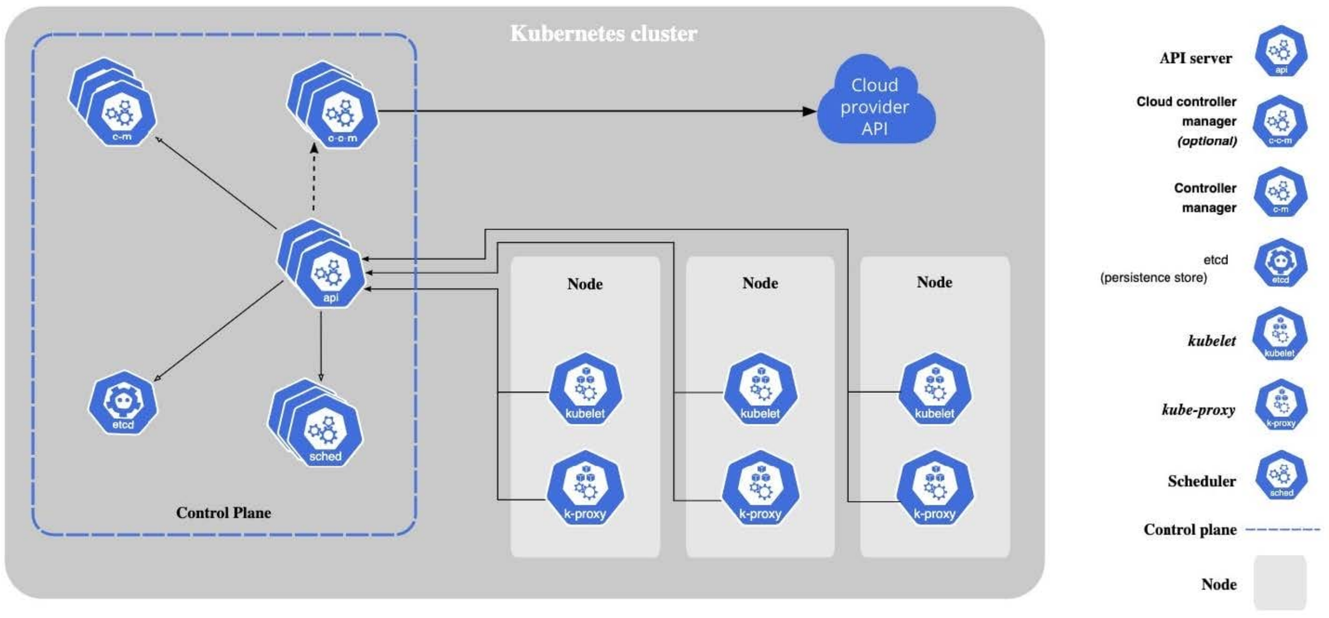
C. Service Mesh

D. kubelet

E. Cloud control manager

Correct Answer: E

Explanation: https://kubernetes.io/docs/concepts/overview/components/



QUESTION 12

Which of the following is NOT a Kubernetes component?

A. Scheduler

B. Docker

C. Cloud Controller manager

D. Kube-proxy

Correct Answer: B

Explanation: Docker is not a Kubernetes component.

QUESTION 13

How to create deployment name app-dep, image=nginx, and replicas 5 using imperative command?

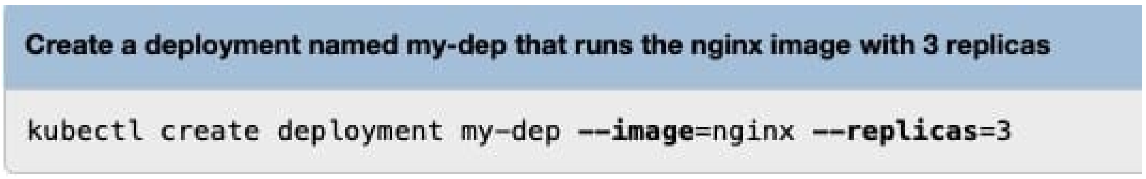
A. kubectl create app-dep deployment --image=nginx --replicas=5

B. kubectl create deployment app-dep --image=nginx --replicas=5

C. kubectl create app-dep deployment --replicas=5 --image=nginx

Correct Answer: B

Explanation: https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#- em-deploymentem-



Create a deployment named my-dep that runs the nginx image with 3 replicas

```
kubectl create deployment my-dep --image=nginx --replicas=3
```

QUESTION 14

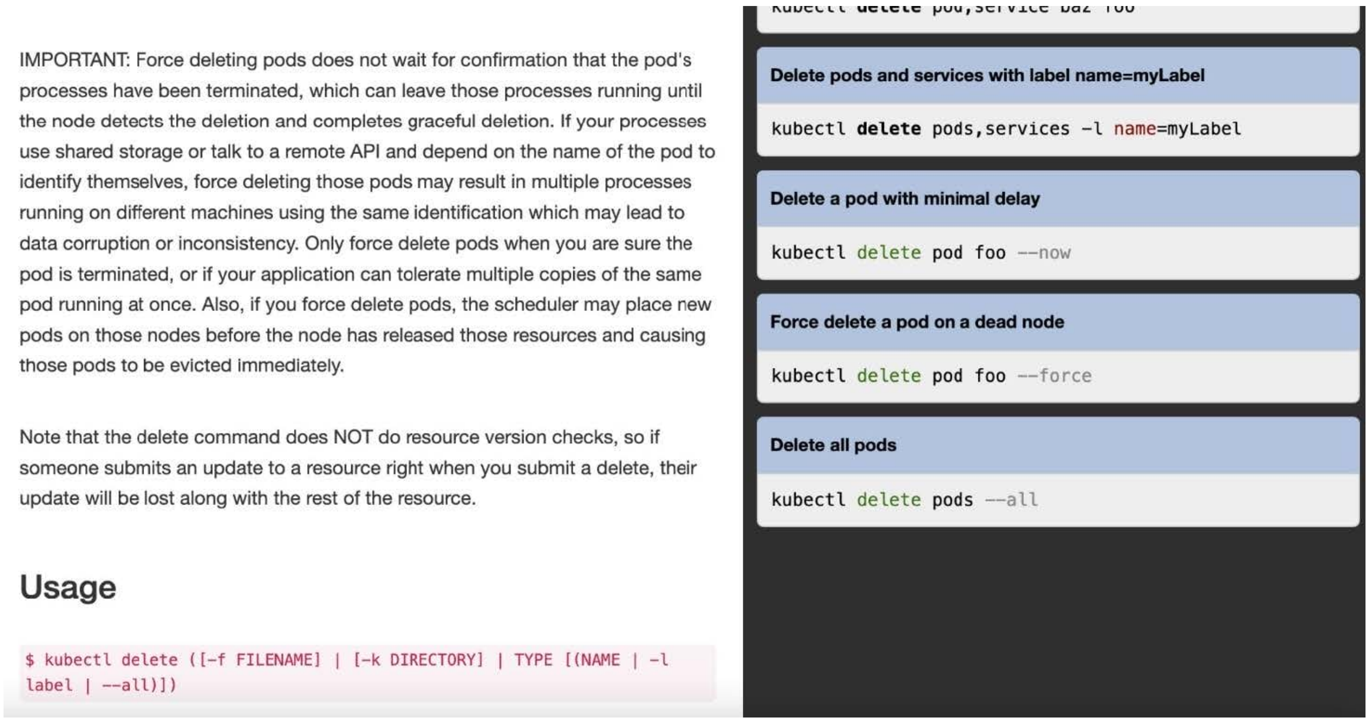
\\'kubectl delete -n my-ns po,svc --all\\' will delete pods and services including uninitialized ones in the namespace \\'my-ns\\'

A. FALSE

B. TRUE

Correct Answer: B

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



IMPORTANT: Force deleting pods does not wait for confirmation that the pod's processes have been terminated, which can leave those processes running until the node detects the deletion and completes graceful deletion. If your processes use shared storage or talk to a remote API and depend on the name of the pod to identify themselves, force deleting those pods may result in multiple processes running on different machines using the same identification which may lead to data corruption or inconsistency. Only force delete pods when you are sure the pod is terminated, or if your application can tolerate multiple copies of the same pod running at once. Also, if you force delete pods, the scheduler may place new pods on those nodes before the node has released those resources and causing those pods to be evicted immediately.

Note that the delete command does NOT do resource version checks, so if someone submits an update to a resource right when you submit a delete, their update will be lost along with the rest of the resource.

Usage

```
$ kubectl delete ([-f FILENAME] | [-k DIRECTORY] | TYPE [(NAME | -l
label | --all)])
```

```
kubectl delete pod,service baz foo
```

Delete pods and services with label name=myLabel

```
kubectl delete pods,services -l name=myLabel
```

Delete a pod with minimal delay

```
kubectl delete pod foo --now
```

Force delete a pod on a dead node

```
kubectl delete pod foo --force
```

Delete all pods

```
kubectl delete pods --all
```

QUESTION 15

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

A. Log

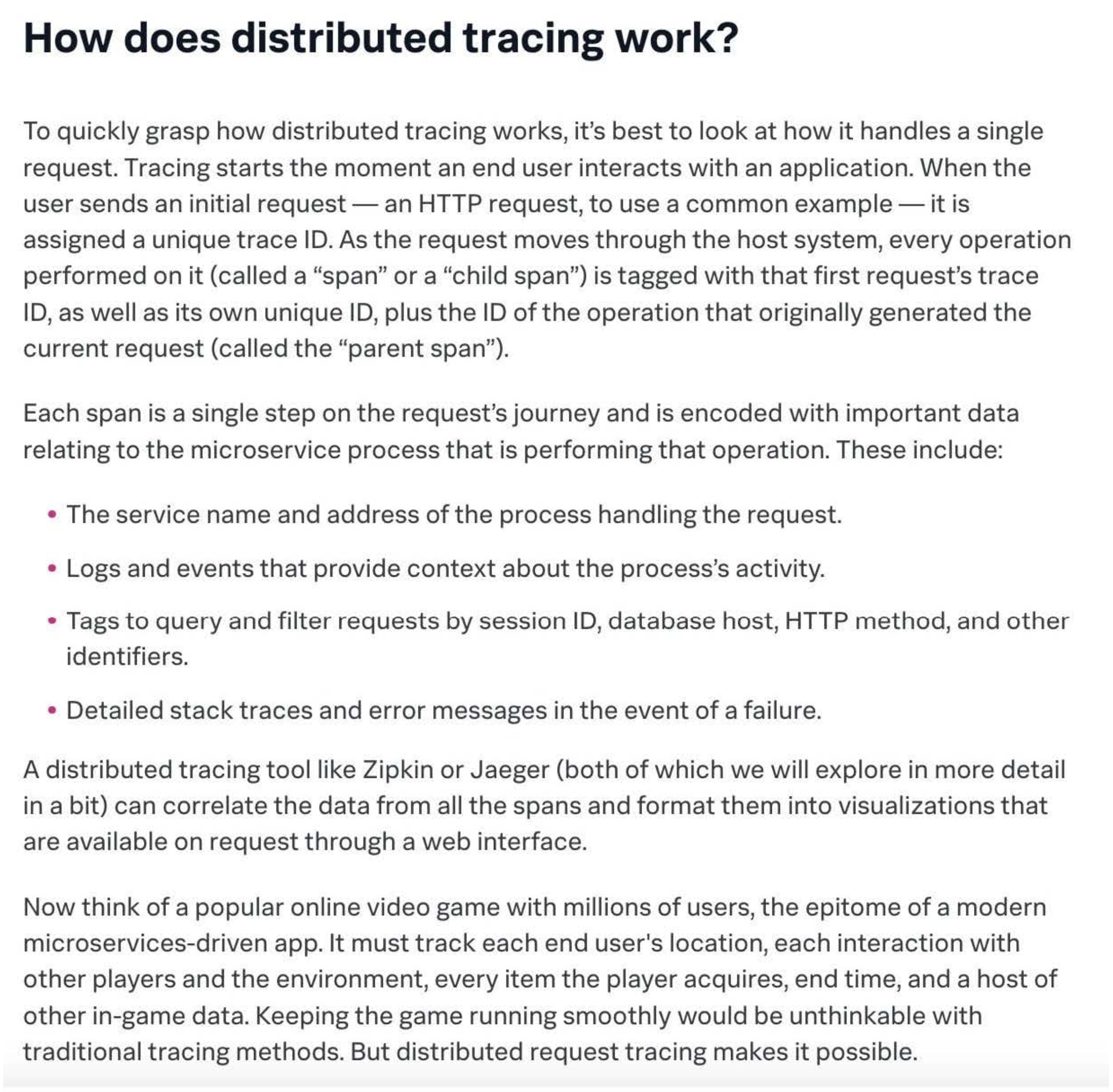
B. Span

C. Trace

D. Bucket

Correct Answer: B

Explanation: https://www.splunk.com/en_us/data-insider/what-is-distributed-tracing.html



How does distributed tracing work?

To quickly grasp how distributed tracing works, it's best to look at how it handles a single request. Tracing starts the moment an end user interacts with an application. When the user sends an initial request — an HTTP request, to use a common example — it is assigned a unique trace ID. As the request moves through the host system, every operation performed on it (called a "span" or a "child span") is tagged with that first request's trace ID, as well as its own unique ID, plus the ID of the operation that originally generated the current request (called the "parent span").

Each span is a single step on the request's journey and is encoded with important data relating to the microservice process that is performing that operation. These include:

- The service name and address of the process handling the request.
- Logs and events that provide context about the process's activity.
- Tags to query and filter requests by session ID, database host, HTTP method, and other identifiers.
- Detailed stack traces and error messages in the event of a failure.

A distributed tracing tool like Zipkin or Jaeger (both of which we will explore in more detail in a bit) can correlate the data from all the spans and format them into visualizations that are available on request through a web interface.

Now think of a popular online video game with millions of users, the epitome of a modern microservices-driven app. It must track each end user's location, each interaction with other players and the environment, every item the player acquires, end time, and a host of other in-game data. Keeping the game running smoothly would be unthinkable with traditional tracing methods. But distributed request tracing makes it possible.

Text, letter