Linux-Foundation

Exam Questions KCNA

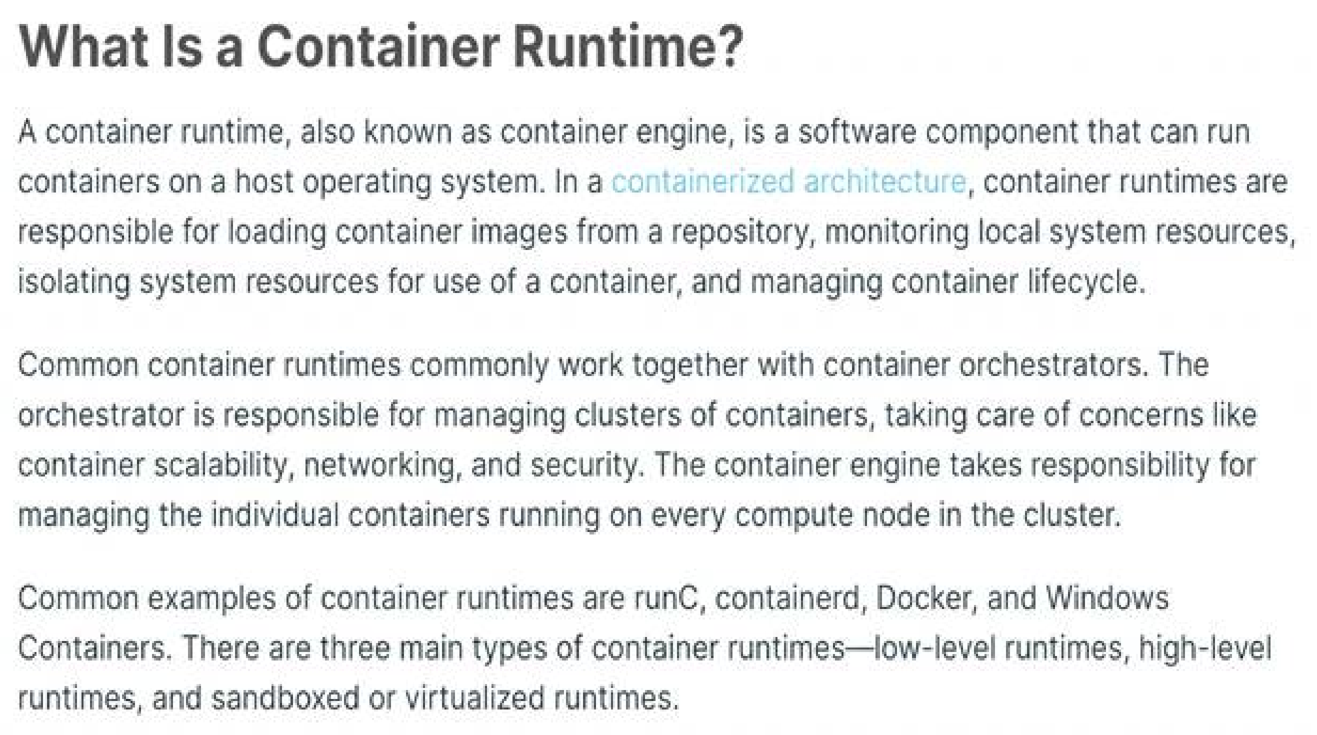
Kubernetes and Cloud Native Associate (KCNA)

NEW QUESTION 1
What is container runtime?

A. The amount of time it takes a container to execute
B. A container image format
C. Another term of kubelet or kubectl
D. Software that runs containers

Answer: D

Explanation:
https://www.aquasec.com/cloud-native-academy/container-security/container-runtime/ Text Description automatically generated


What Is a Container Runtime?

A container runtime, also known as container engine, is a software component that can run containers on a host operating system. In a containerized architecture, container runtimes are responsible for loading container images from a repository, monitoring local system resources, isolating system resources for use of a container, and managing container lifecycle.

Common container runtimes commonly work together with container orchestrators. The orchestrator is responsible for managing clusters of containers, taking care of concerns like container scalability, networking, and security. The container engine takes responsibility for managing the individual containers running on every compute node in the cluster.

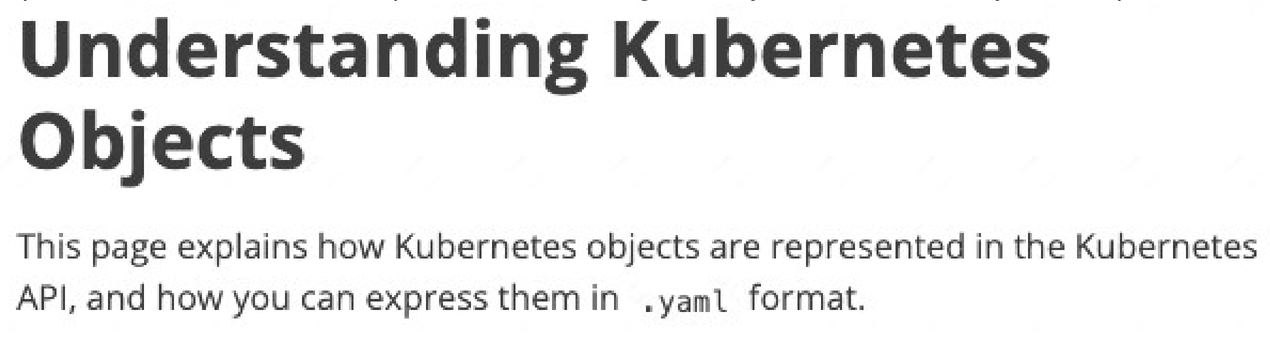
Common examples of container runtimes are runC, containerd, Docker, and Windows Containers. There are three main types of container runtimes—low-level runtimes, high-level runtimes, and sandboxed or virtualized runtimes.

NEW QUESTION 2
To specify a Kubernetes object which language is used?

A. JSON
B. Go
C. YAML
D. Node
E. Python

Answer: C

Explanation:
https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/ Graphical user interface, text Description automatically generated


Understanding Kubernetes Objects

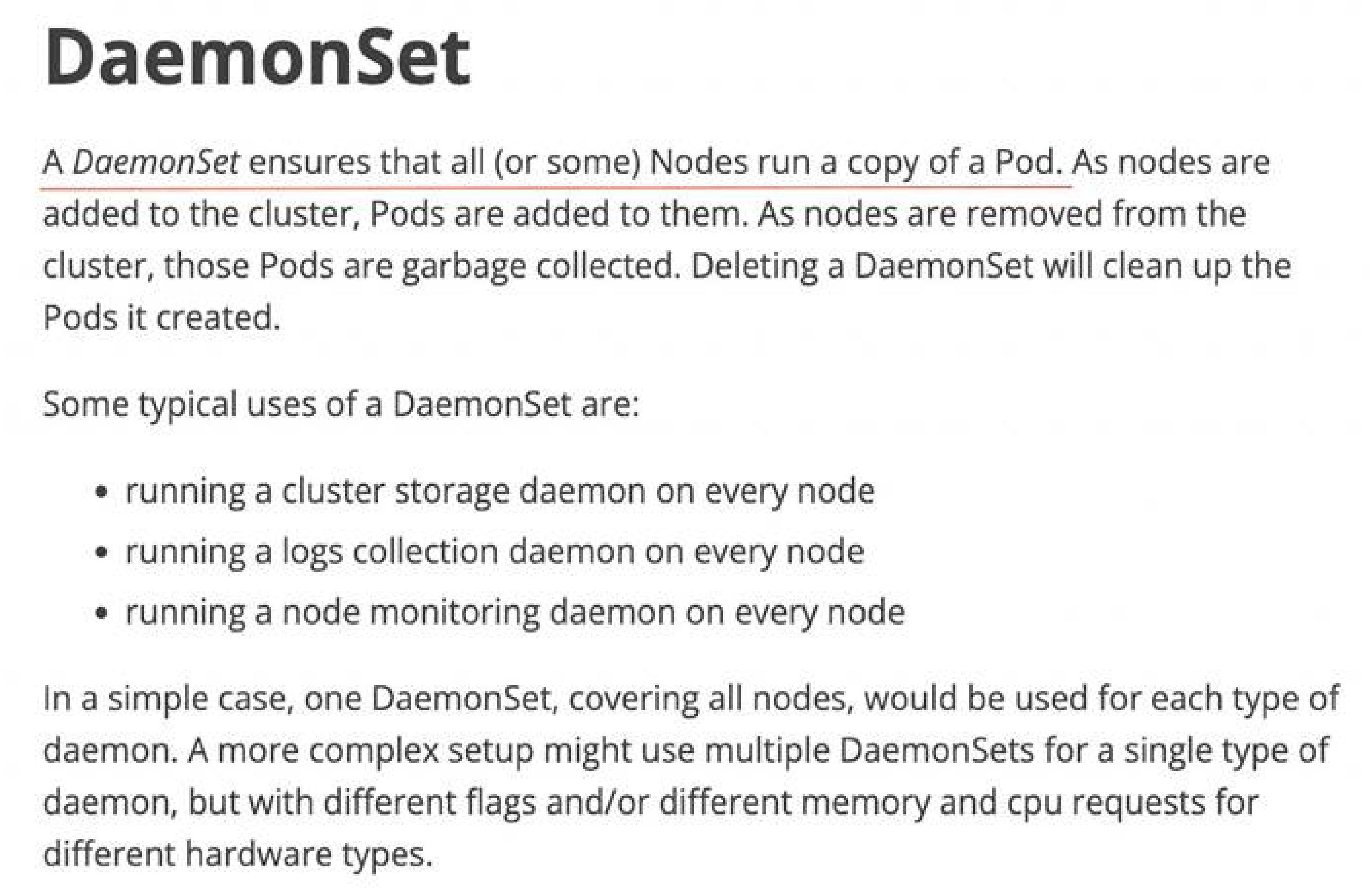
This page explains how Kubernetes objects are represented in the Kubernetes API, and how you can express them in `.yaml` format.

NEW QUESTION 3
What Kubernetes resource would allow you to run one Pod on some of your Nodes?

A. DaemonSet
B. ClusterSet
C. Deployment
D. ReplicaSet

Answer: A

Explanation:
https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/ Graphical user interface, text, application Description automatically generated



DaemonSet

A *DaemonSet* ensures that all (or some) Nodes run a copy of a Pod. As nodes are added to the cluster, Pods are added to them. As nodes are removed from the cluster, those Pods are garbage collected. Deleting a DaemonSet will clean up the Pods it created.

Some typical uses of a DaemonSet are:

- running a cluster storage daemon on every node
- running a logs collection daemon on every node
- running a node monitoring daemon on every node

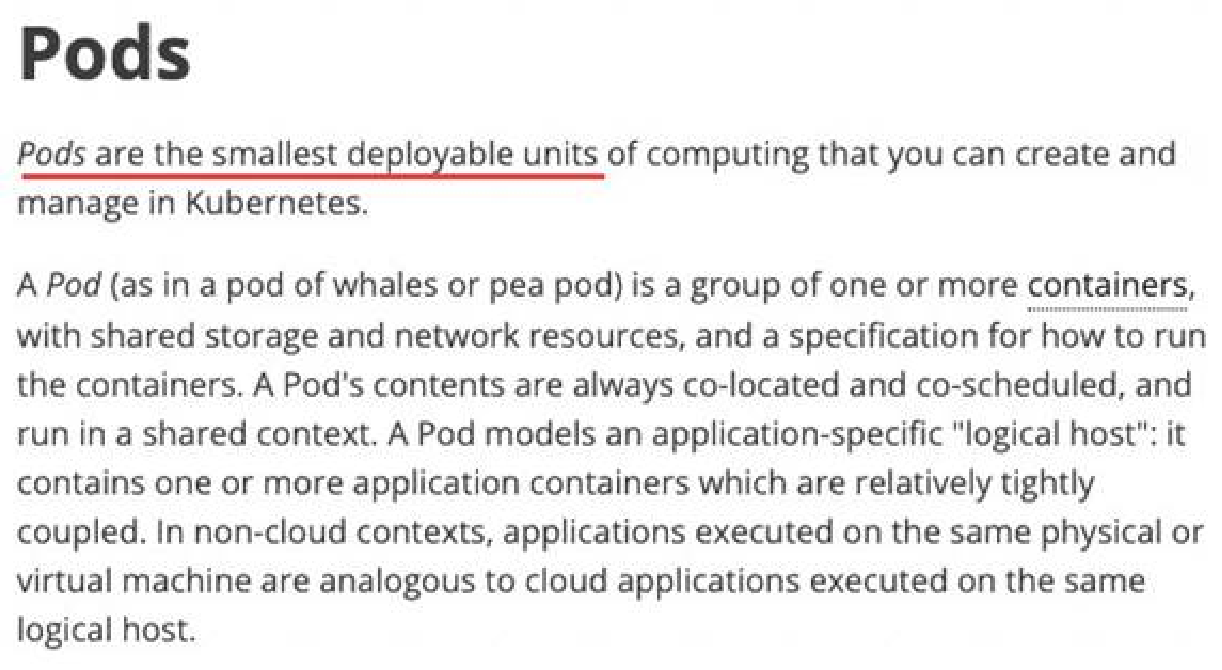
In a simple case, one DaemonSet, covering all nodes, would be used for each type of daemon. A more complex setup might use multiple DaemonSets for a single type of daemon, but with different flags and/or different memory and cpu requests for different hardware types.

NEW QUESTION 4
What is the smallest possible unit in Kubernetes to run a container?

A. pod
B. docker
C. service
D. container

Answer: A

Explanation:
https://kubernetes.io/docs/concepts/workloads/pods/
Graphical user interface, text, application Description automatically generated



Pods

Pods are the smallest deployable units of computing that you can create and manage in Kubernetes.

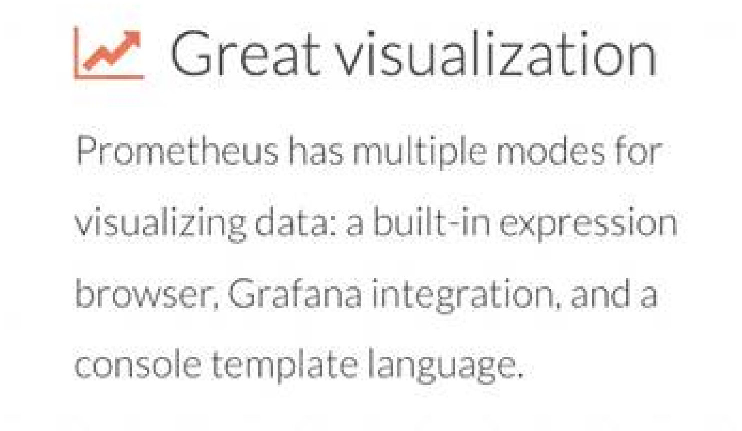
A *Pod* (as in a pod of whales or pea pod) is a group of one or more containers, with shared storage and network resources, and a specification for how to run the containers. A Pod's contents are always co-located and co-scheduled, and run in a shared context. A Pod models an application-specific "logical host": it contains one or more application containers which are relatively tightly coupled. In non-cloud contexts, applications executed on the same physical or virtual machine are analogous to cloud applications executed on the same logical host.

NEW QUESTION 5
What tool allows us to build useful visual representations of prometheus data?

A. Grafana
B. kubectl
C. Distributed system tracing
D. Rook
E. Kibana

Answer: A

Explanation:
https://prometheus.io/
Graphical user interface, text, application Description automatically generated


Great visualization

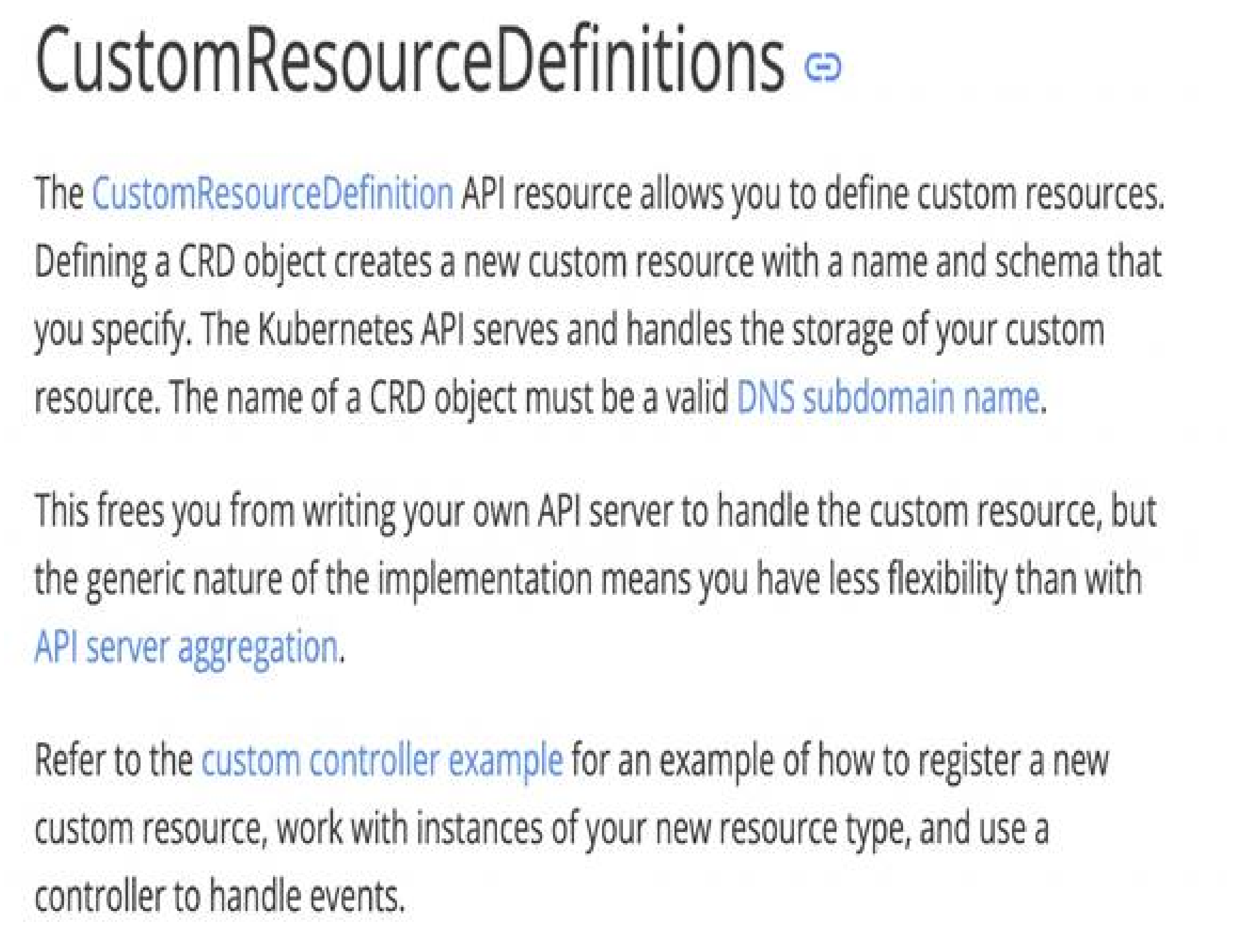
Prometheus has multiple modes for visualizing data: a built-in expression browser, Grafana integration, and a console template language.

NEW QUESTION 6
What can you use to add new resource types to your cluster?

A. start container
B. CustomResourceDefinitions
C. init container
D. Flux
E. CRI-O

Answer: B

Explanation:
https://kubernetes.io/docs/concepts/extend-kubernetes/api-extension/custom-resources/ Graphical user interface, text, application Description automatically generated


CustomResourceDefinitions

The CustomResourceDefinition API resource allows you to define custom resources. Defining a CRD object creates a new custom resource with a name and schema that you specify. The Kubernetes API serves and handles the storage of your custom resource. The name of a CRD object must be a valid DNS subdomain name.

This frees you from writing your own API server to handle the custom resource, but the generic nature of the implementation means you have less flexibility than with API server aggregation.

Refer to the custom controller example for an example of how to register a new custom resource, work with instances of your new resource type, and use a controller to handle events.

NEW QUESTION 7
What kind of limitation cgroups allows?

A. Prioritization
B. Resource limiting
C. Accounting
D. None of the options
E. Control
F. Server cpu and memory

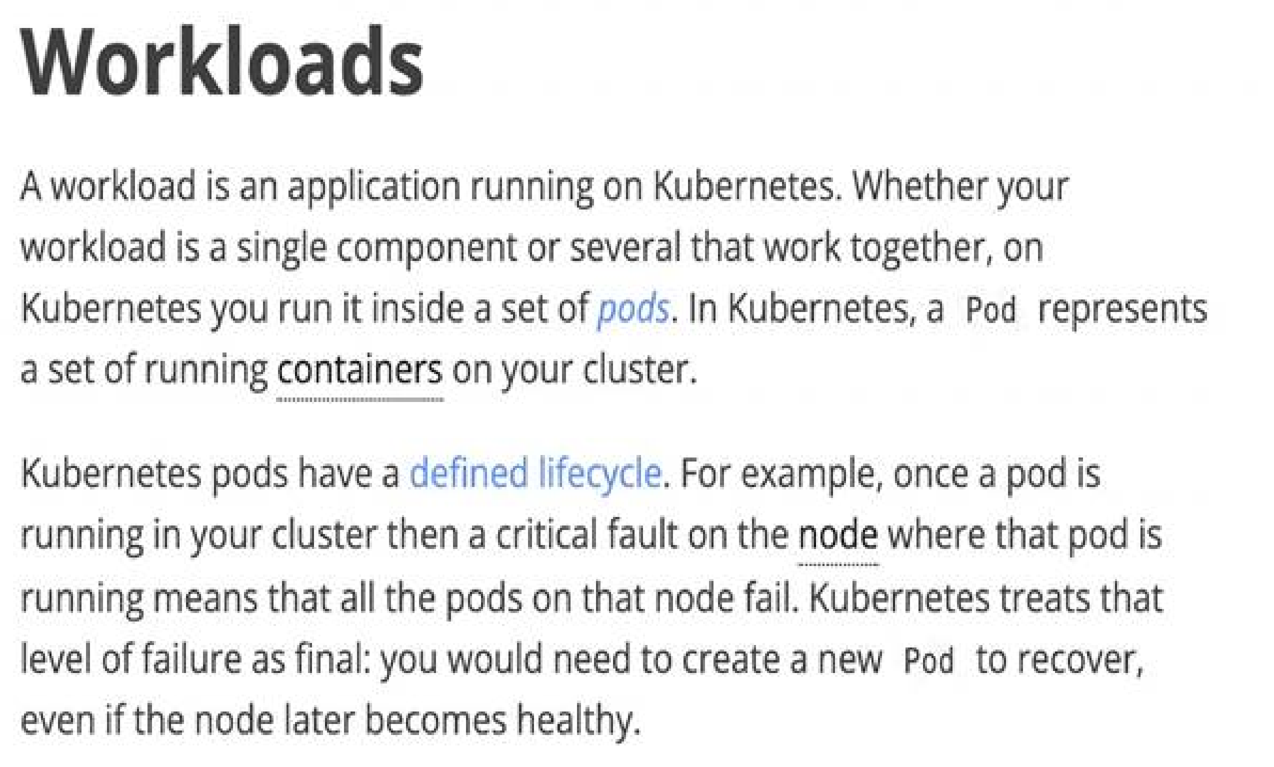
Answer: ABCE

NEW QUESTION 8
A ________ is an application running on kubernetes.

A. node
B. pod
C. workload
D. container

Answer: C

Explanation:
https://kubernetes.io/docs/concepts/workloads/ Text Description automatically generated


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*. In Kubernetes, a `Pod` represents a set of running containers on your cluster.

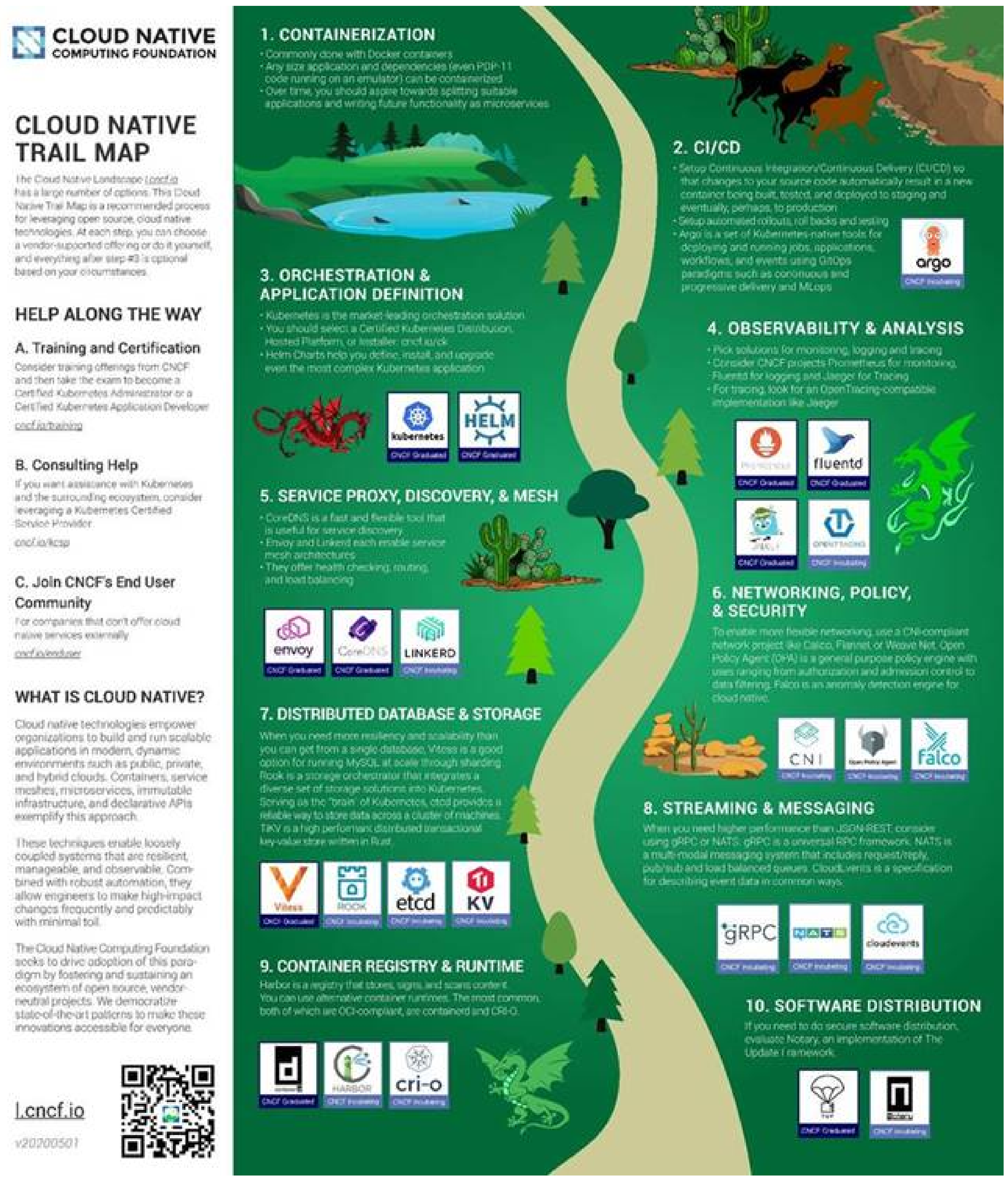
Kubernetes pods have a defined lifecycle. For example, once a pod is running in your cluster then a critical fault on the node where that pod is running means that all the pods on that node fail. Kubernetes treats that level of failure as final: you would need to create a new `Pod` to recover, even if the node later becomes healthy.

NEW QUESTION 9
Which project in this list is a leading project in the observability space?

A. Jaeger
B. Vitess
C. Argo
D. Kubernetes

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.

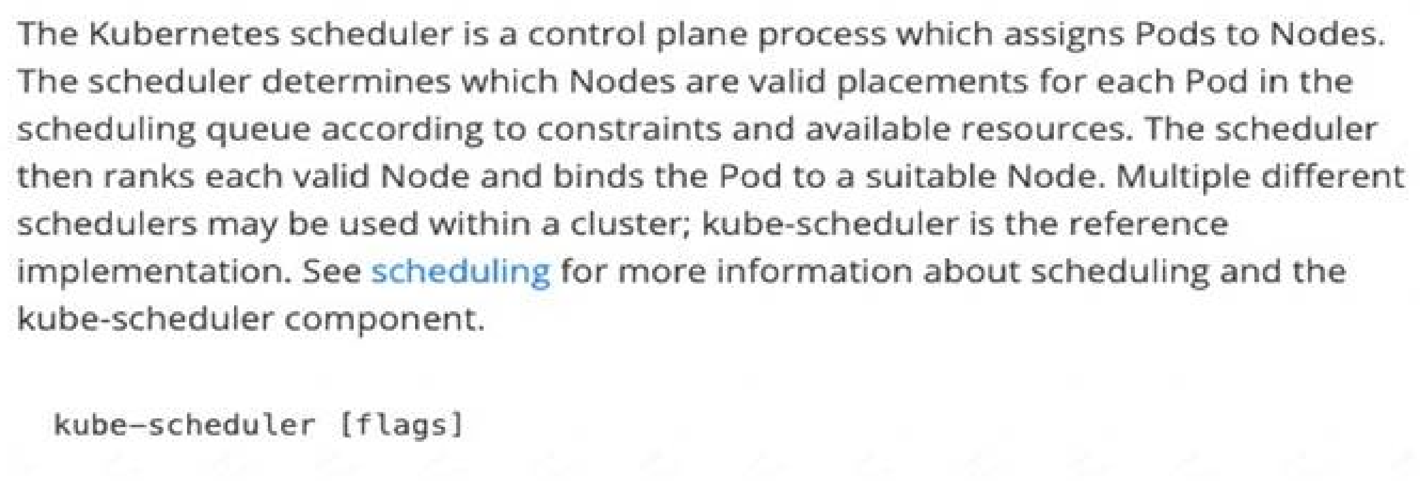
10. SOFTWARE DISTRIBUTION
If you need to do secure software distribution, evaluate Notary, an implementation of The Update Framework.

NEW QUESTION 10
A new Pod is created. Then, the Pod is assigned to a Node. Which Kubernetes component was re-sponsible for determining which Node to assign the Pod to?

A. kubelet
B. Scheduler
C. API Server
D. Controller manager

Answer: B

Explanation:
https://kubernetes.io/docs/reference/command-line-tools-reference/kube-scheduler/ Graphical user interface, text, application Description automatically generated



The Kubernetes scheduler is a control plane process which assigns Pods to Nodes. The scheduler determines which Nodes are valid placements for each Pod in the scheduling queue according to constraints and available resources. The scheduler then ranks each valid Node and binds the Pod to a suitable Node. Multiple different schedulers may be used within a cluster; kube-scheduler is the reference implementation. See scheduling for more information about scheduling and the kube-scheduler component.

```
kube-scheduler [flags]
```

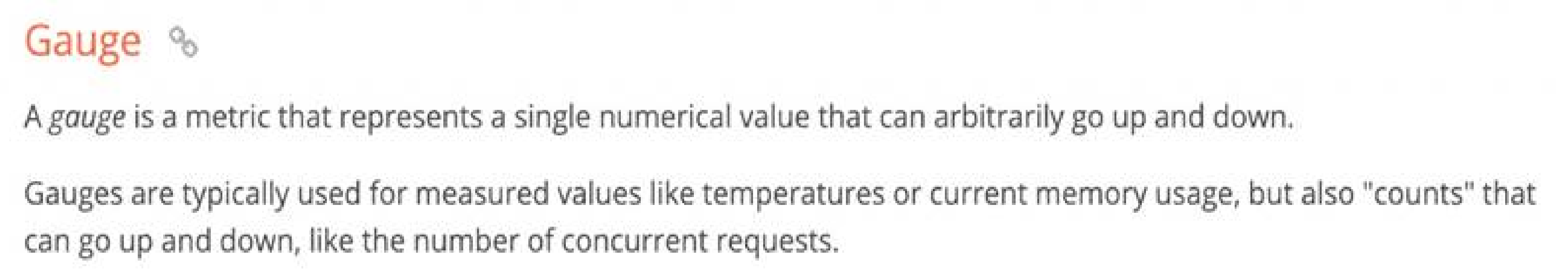
NEW QUESTION 10
Which prometheus metric type represents a single number value that can increase and decrease over time?

A. Gauge

B. Histogram
C. Summary
D. Counter

Answer: A

Explanation:
https://prometheus.io/docs/concepts/metric_types/#gauge Graphical user interface, text Description automatically generated


Gauge

A *gauge* is a metric that represents a single numerical value that can arbitrarily go up and down.

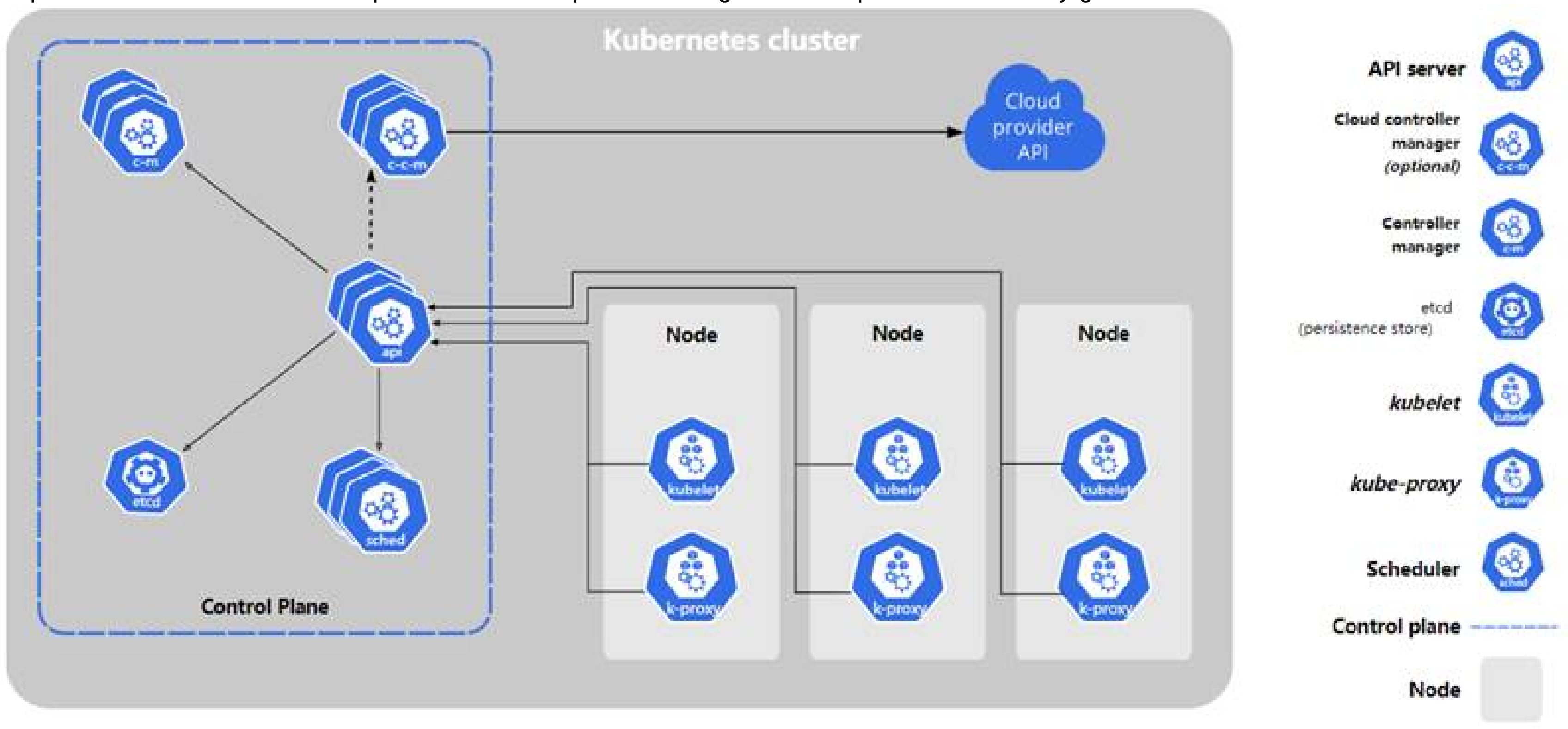
Gauges are typically used for measured values like temperatures or current memory usage, but also "counts" that can go up and down, like the number of concurrent requests.

NEW QUESTION 12
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

Answer: E

Explanation:
https://kubernetes.io/docs/concepts/overview/components/ Diagram Description automatically generated

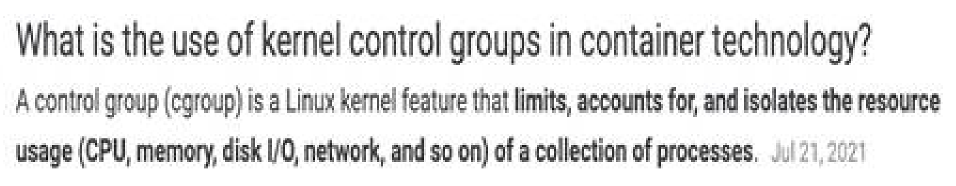


NEW QUESTION 16
What do control groups provide when it come to containers

A. Permission
B. Image Storage
C. Isolation
D. Logging

Answer: C

Explanation:
Text Description automatically generated


What is the use of kernel control groups in container technology?

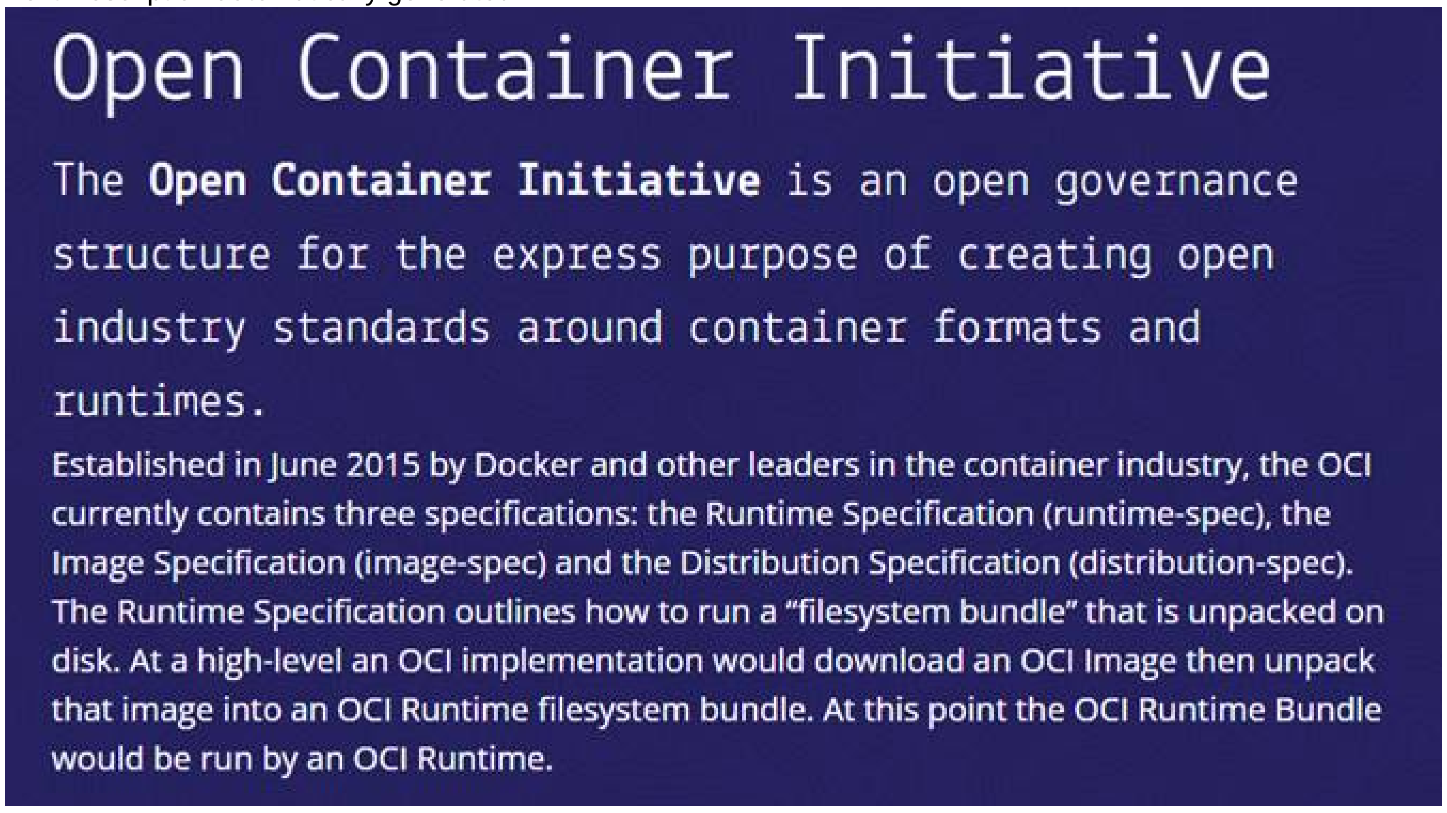
A control group (cgroup) is a Linux kernel feature that **limits, accounts for, and isolates the resource usage (CPU, memory, disk I/O, network, and so on) of a collection of processes.** Jul 21, 2021

NEW QUESTION 20
What is Open Container Initiative 'OCI'?

A. A protocol for communicating with the kubernetes api
B. The governing body of the Cloud Native Computing Foundation 'CNCF'
C. An open standard for managing service mesh in kubernetes
D. An organization that creates open standards for containers

Answer: D

Explanation:
https://opencontainers.org/
Text Description automatically generated



Open Container Initiative

The **Open Container Initiative** is an open governance structure for the express purpose of creating open industry standards around container formats and runtimes.

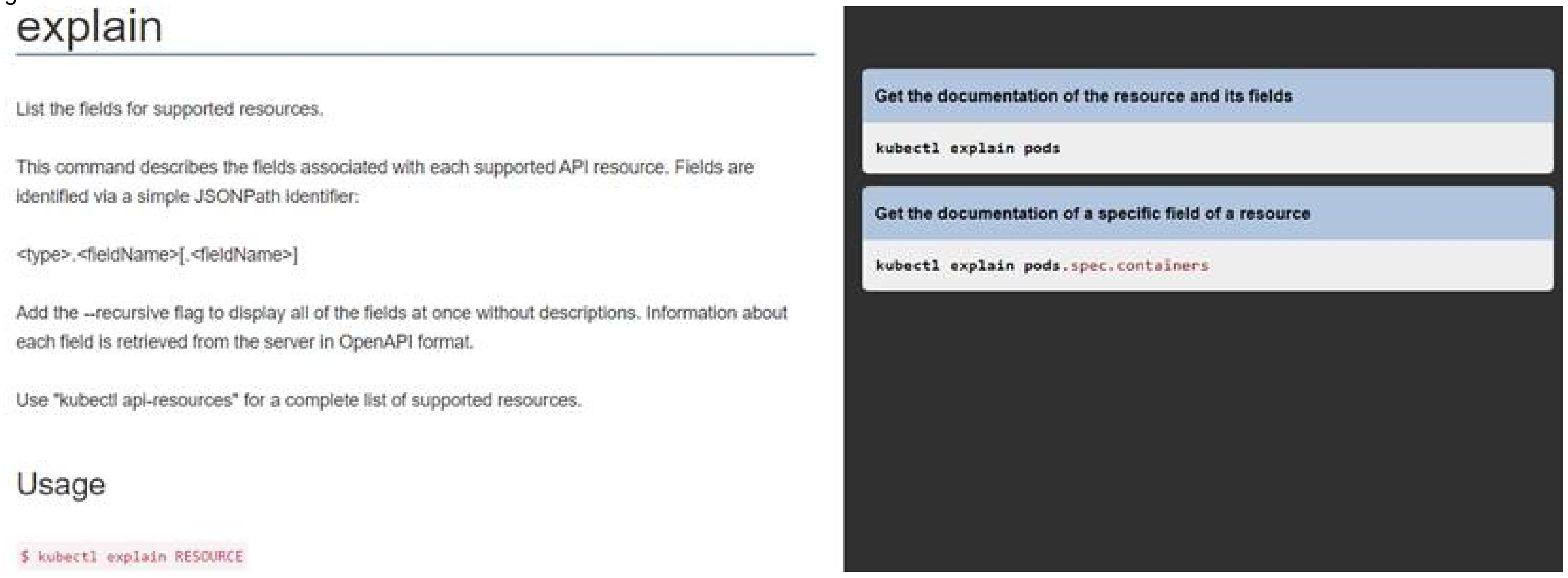
Established in June 2015 by Docker and other leaders in the container industry, the OCI currently contains three specifications: the Runtime Specification (runtime-spec), the Image Specification (image-spec) and the Distribution Specification (distribution-spec). The Runtime Specification outlines how to run a "filesystem bundle" that is unpacked on disk. At a high-level an OCI implementation would download an OCI Image then unpack that image into an OCI Runtime filesystem bundle. At this point the OCI Runtime Bundle would be run by an OCI Runtime.

NEW QUESTION 24
What command can you use to get documentation about a resource type from the command line?

A. kubectl api-resources
B. kubectl explain
C. kubectl get
D. kubeadm get-resource

Answer: B

Explanation:
https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#explain Graphical user interface, text, application, email Description automatically generated



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>]

Add the --recursive flag to display all of the fields at once without descriptions. Information about each field is retrieved from the server in OpenAPI format.

Use "kubectl api-resources" for a complete list of supported resources.

Usage

```
$ kubectl explain RESOURCE
```

Get the documentation of the resource and its fields

```
kubectl explain pods
```

Get the documentation of a specific field of a resource

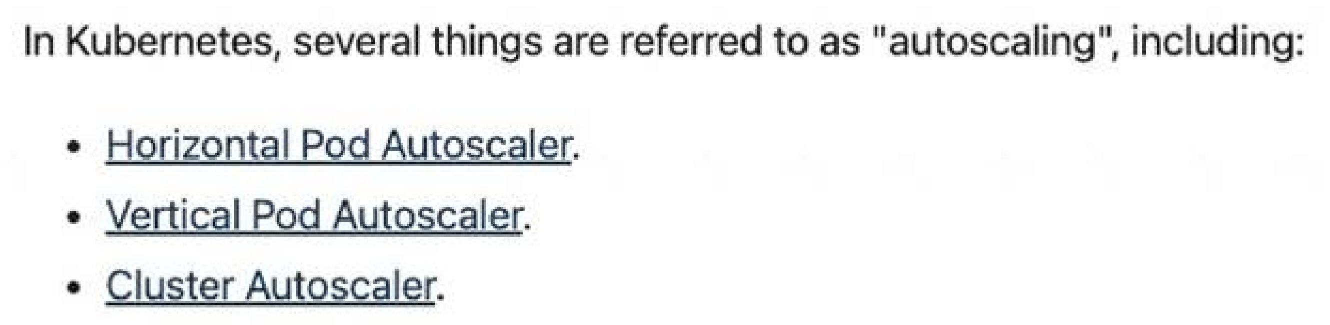
```
kubectl explain pods.spec.containers
```

NEW QUESTION 29
Which of the following is not the Kubernetes AutoScaling Strategy?

A. Horizontal Pod Autoscaler
B. Cluster Autoscaler
C. Vertical Pod Autoscaler
D. Load Balancing AutoScaler

Answer: D

Explanation:
https://learnk8s.io/kubernetes-autoscaling-strategies
Graphical user interface, text Description automatically generated with medium confidence


In Kubernetes, several things are referred to as "autoscaling", including:

- Horizontal Pod Autoscaler.
- Vertical Pod Autoscaler.
- Cluster Autoscaler.

NEW QUESTION 34
What framework allows developers to write code without worrying about the servers and operating systems they will run on?

A. Virtualization
B. Docker
C. Serverless
D. Kubernetes

Answer: C

NEW QUESTION 35
How would you return all the pod data in the json format using kubectl command?

A. kubectl get pods -o json
B. kubectl get pods --all-namspaces
C. kubectl get pods -o wide
D. kubectl get pods -o jsonpath

Answer: A

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

NEW QUESTION 38
What Linux feature is used to provide isolation for containers?

A. Processes
B. Services
C. NetworkPolicy
D. Control groups

Answer: D

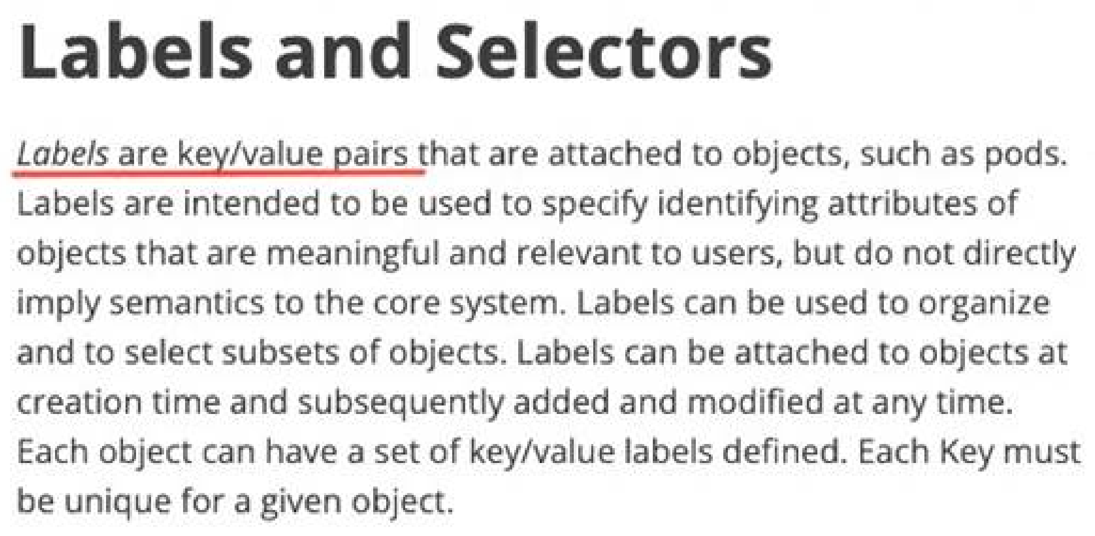
Explanation:
Control groups provide isolation for container processes, keeping them separate from other process-es on the host.

NEW QUESTION 41
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.

Answer: C

Explanation:
https://kubernetes.io/docs/concepts/overview/working-with-objects/labels/ Text Description automatically generated


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.

NEW QUESTION 46

Which part of a Kubernetes cluster is responsible for running container workloads?

A. Worker Node
B. kube-proxy
C. Control plane
D. etcd

Answer: A

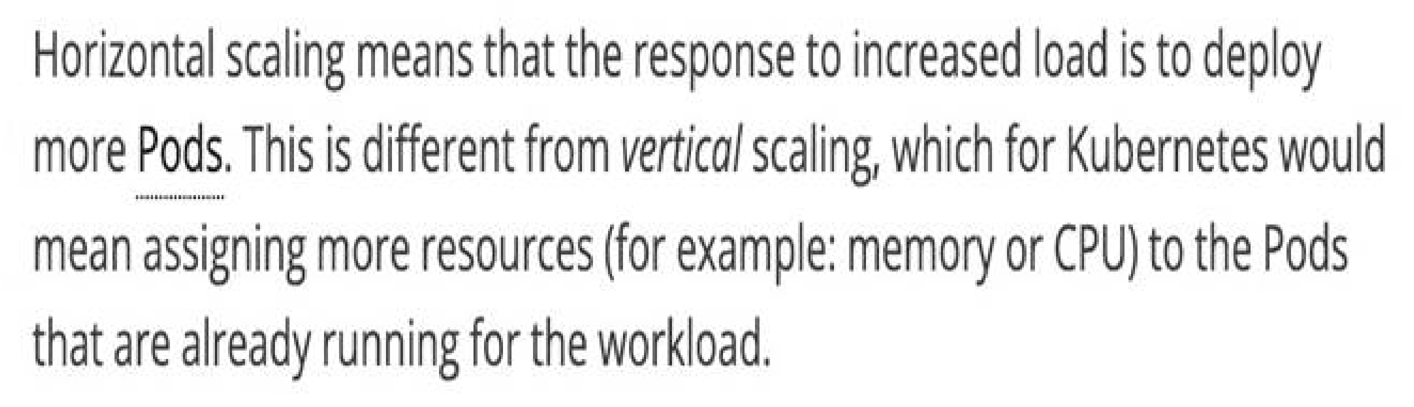
Explanation:
Worker Nodes are responsible for executing containerized workloads.

NEW QUESTION 48
Which of the following is an example of vertical scaling?

A. Using cluster autoscaler
B. Adding more resources (memory and/or cpu) to a kubernetes node
C. Adding more nodes to kubernetes cluster
D. Adding more replica pods to a deployment

Answer: B

Explanation:
https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ Text Description automatically generated

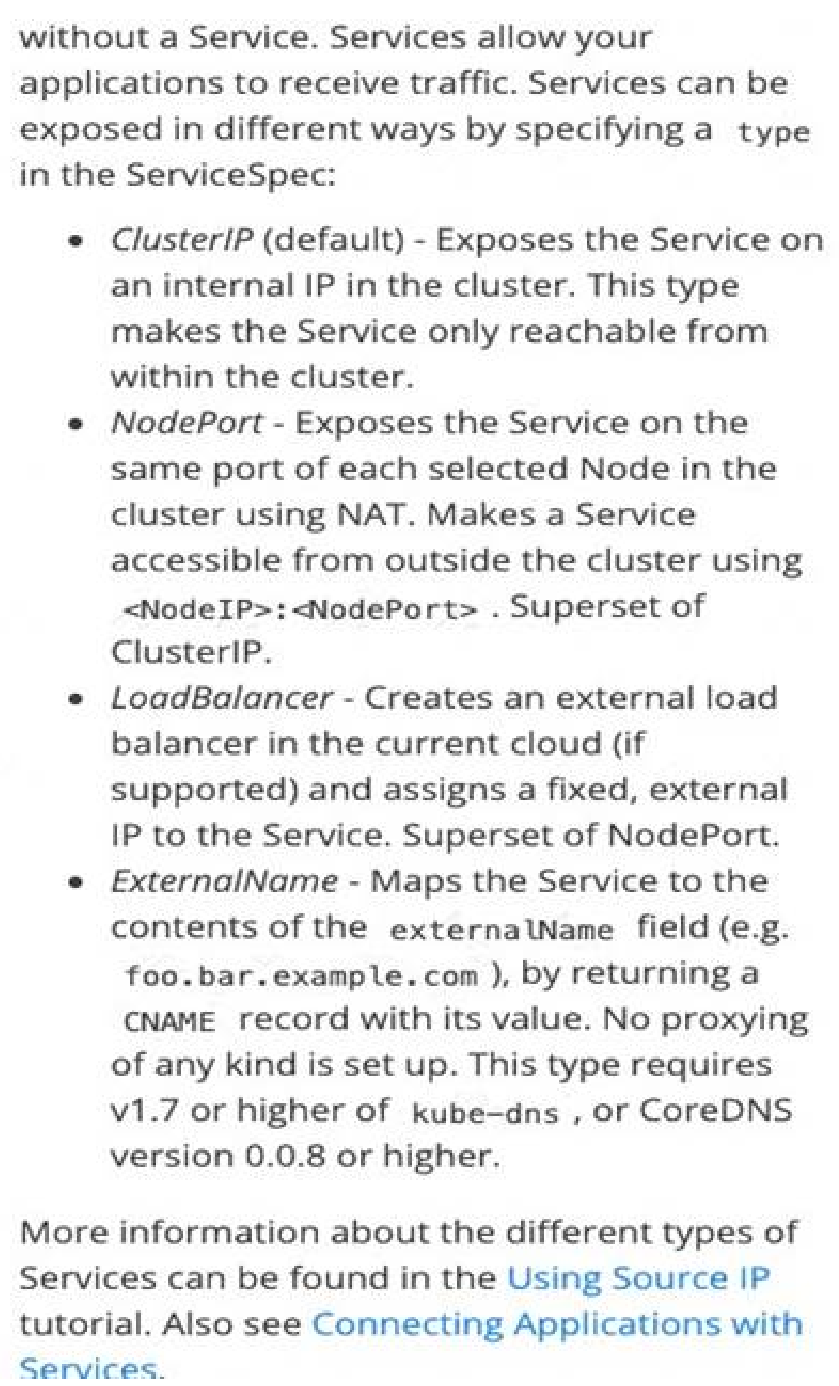

Horizontal scaling means that the response to increased load is to deploy more Pods. This is different from *vertical* scaling, which for Kubernetes would mean assigning more resources (for example: memory or CPU) to the Pods that are already running for the workload.

NEW QUESTION 51
Which is not a service type in Kubernetes?

A. ClusterIP
B. NodePort
C. Ingress
D. LoadBalancer
E. ExternalName

Answer: C

Explanation:
https://kubernetes.io/docs/tutorials/kubernetes-basics/expose/expose-intro/


without a Service. Services allow your applications to receive traffic. Services can be exposed in different ways by specifying a `type` in the ServiceSpec:

- *ClusterIP* (default) - Exposes the Service on an internal IP in the cluster. This type makes the Service only reachable from within the cluster.
- *NodePort* - Exposes the Service on the same port of each selected Node in the cluster using NAT. Makes a Service accessible from outside the cluster using `<NodeIP>:<NodePort>`. Superset of ClusterIP.
- *LoadBalancer* - Creates an external load balancer in the current cloud (if supported) and assigns a fixed, external IP to the Service. Superset of NodePort.
- *ExternalName* - Maps the Service to the contents of the `externalName` field (e.g. `foo.bar.example.com`), by returning a `CNAME` record with its value. No proxying of any kind is set up. This type requires v1.7 or higher of `kube-dns`, or CoreDNS version 0.0.8 or higher.

More information about the different types of Services can be found in the Using Source IP tutorial. Also see Connecting Applications with Services.

Text Description automatically generated

NEW QUESTION 56
Which of the following is an advantage a cloud-native microservices application has over monolithic applications?

A. Cloud-native microservices applications tend to be faster and more responsive than monolithic applications.
B. Cloud-native microservice applications tend to be easier to troubleshoot.
C. Cloud-native microservice applications tend to be easier to scale and perform updates on.

Answer: C

Explanation:
Cloud-native applications tend to be microservice base, they have individual services that can be independently scaled, updated and rolled back. This makes scaling and update operations simpler and less risky.

NEW QUESTION 59
Which of the following is not a stop on the cloud native trailmap?

A. Microservices
B. CI/CD
C. Containerization
D. Software distribution

Answer: A

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

NEW QUESTION 62
Which of the following factors does scheduling take into account when selecting a Node?

A. How many replicas there are in a Deployment
B. Services
C. Resource requirements
D. The number of existing Pods on a Node

Answer: C

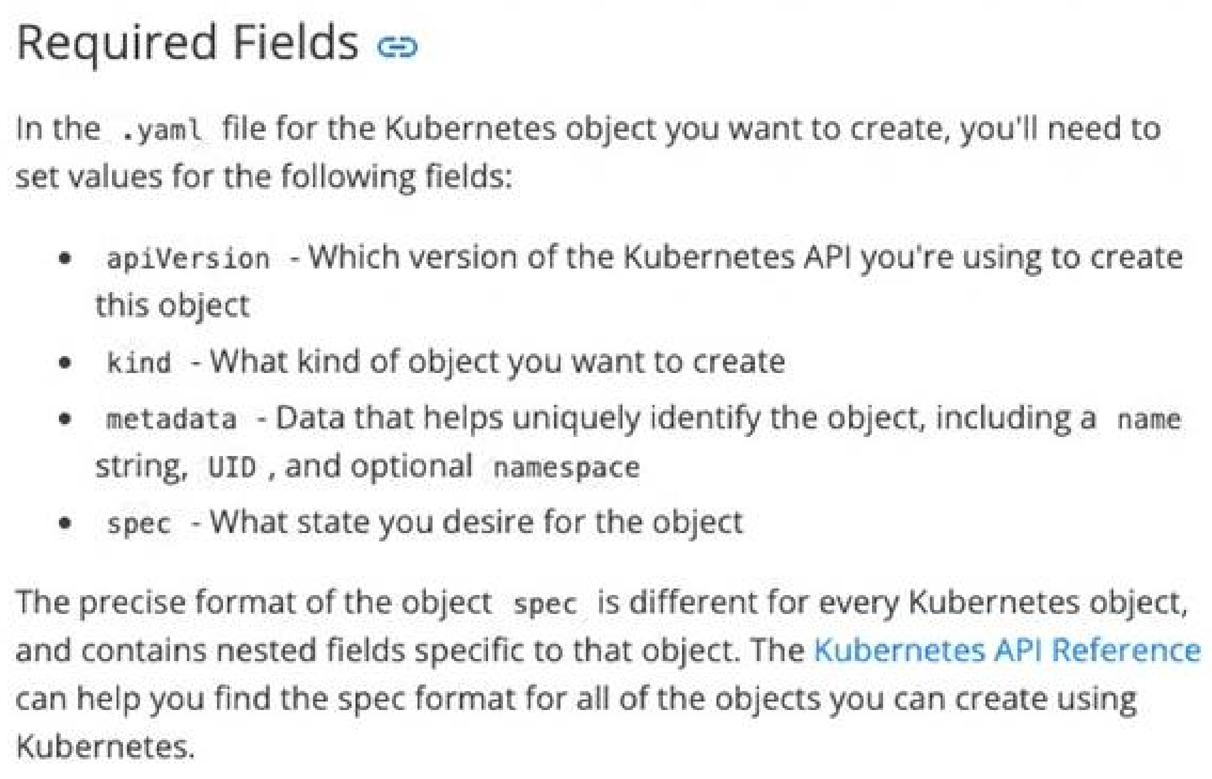
Explanation:
Scheduling takes resource requirements into account in the form of resource requests.

NEW QUESTION 66
Which of the following is not the required field to describe Kubernetes objects?

A. metadata
B. apiVersion
C. Kind
D. Container
E. spec

Answer: D

Explanation:
https://kubernetes.io/docs/concepts/overview/working-with-objects/kubernetes-objects/ Graphical user interface, text, application Description automatically generated


Required Fields

In the `.yaml` file for the Kubernetes object you want to create, you'll need to set values for the following fields:

- `apiVersion` - Which version of the Kubernetes API you're using to create this object
- `kind` - What kind of object you want to create
- `metadata` - Data that helps uniquely identify the object, including a `name` string, `UID`, and optional `namespace`
- `spec` - What state you desire for the object

The precise format of the object `spec` is different for every Kubernetes object, and contains nested fields specific to that object. The Kubernetes API Reference can help you find the spec format for all of the objects you can create using Kubernetes.

NEW QUESTION 68
What is etcd used for in Kubernetes?

A. Integration with cloud platforms
B. Network routing for the cluster
C. Kubernetes API security
D. Backend object storage for the Kubernetes API

Answer: D

Explanation:
etcd serves as a distributed object store that backs the Kubernetes API.

NEW QUESTION 69
What is not semantic versioning?

A. 1.0.0
B. 2022-05-04
C. 1.0.0-alpha
D. 1.0.0-beta.2

Answer: B

Explanation:
https://semver.org/
RegEx SemVer at https://regex101.com/r/vkijKf/1/

NEW QUESTION 72
Observability and monitoring are not the same?

A. True
B. False

Answer: A

NEW QUESTION 75
What is autoscaling?

A. Automatically measuring resource usage

B. Automatically assigning workloads to nodes in a cluster
C. Automatically repairing broken application instances
D. Automatically adding or removing compute resources as needed

Answer: D

Explanation:
https://kubernetes.io/blog/2016/07/autoscaling-in-kubernetes/
Autoscaling means automatically scaling up or down in response to real-time usage data.

NEW QUESTION 80
Continuous delivery is .

A. Manually deploying the code
B. Coding, Building and Testing the code
C. Automatically deploying code to [container or server] environment

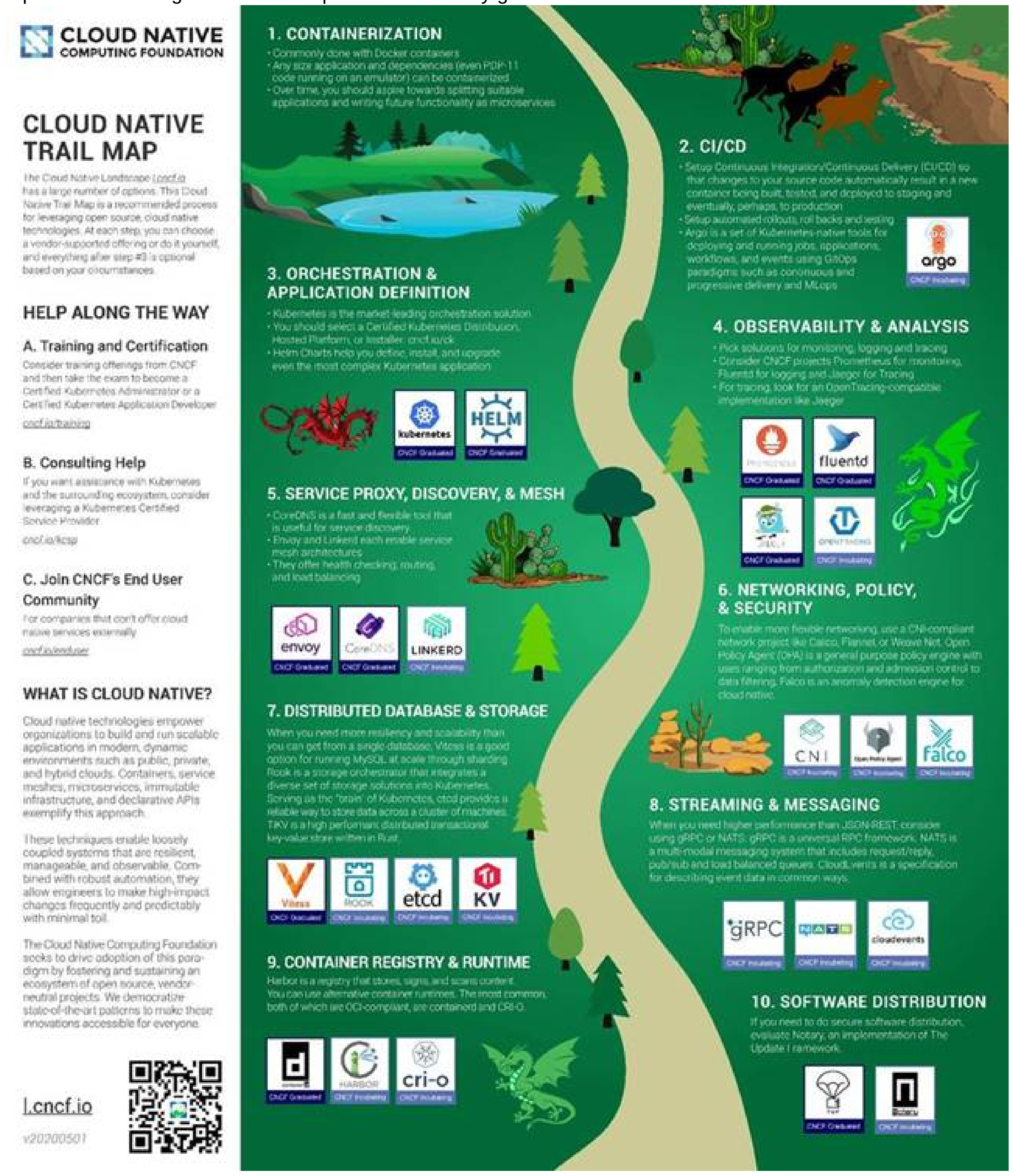
Answer: C

NEW QUESTION 85
What CNCF project is the leading DNS project in the CNCF landscape?

A. Kubernetes
B. gRPC
C. KubeDNS
D. CoreDNS

Answer: D

Explanation:
https://github.com/cncf/landscape#trail-map
A picture containing timeline Description automatically generated

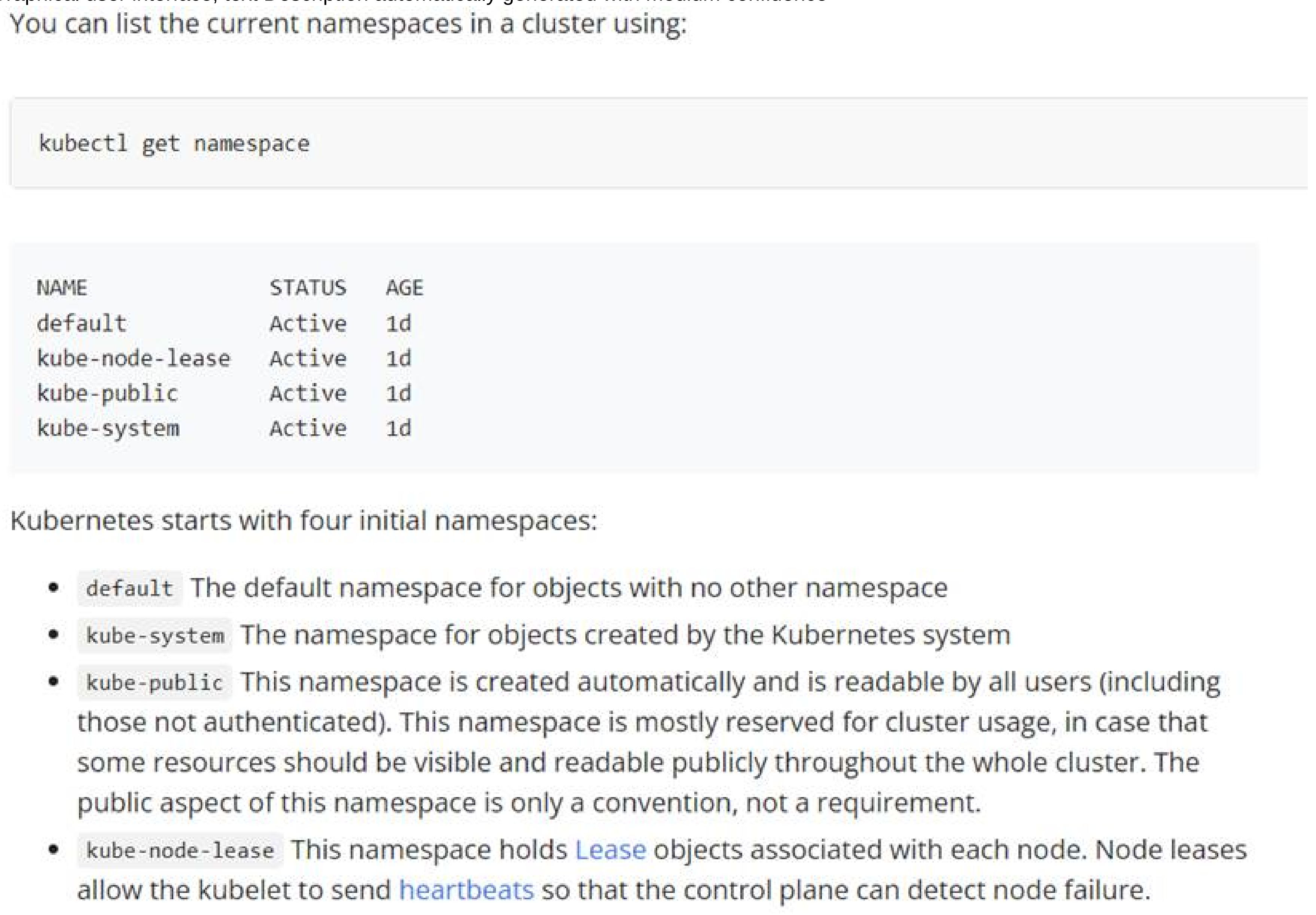

CLOUD NATIVE COMPUTING FOUNDATION
CLOUD NATIVE TRAIL MAP
HELP ALONG THE WAY
A. Training and Certification
B. Consulting Help
C. Join CNCF's End User Community
WHAT IS CLOUD NATIVE?
l.cncf.io
v20200501
1. CONTAINERIZATION
2. CI/CD
argo
3. ORCHESTRATION & APPLICATION DEFINITION
kubernetes
HELM
4. OBSERVABILITY & ANALYSIS
fluentd
5. SERVICE PROXY, DISCOVERY, & MESH
envoy
CoreDNS
LINKERD
6. NETWORKING, POLICY, & SECURITY
CNI
falco
7. DISTRIBUTED DATABASE & STORAGE
Vitess
ROOK
etcd
KV
8. STREAMING & MESSAGING
gRPC
NATS
cloudevents
9. CONTAINER REGISTRY & RUNTIME
HARBOR
cri-o
10. SOFTWARE DISTRIBUTION

NEW QUESTION 89
What are default kubernetes namespaces?

A. default, kube-public, kube-system, kube-node-lease
B. kube-default, kube-public, kube-system, kube-node-lease
C. default, kube-public, kube-systems, kube-node-lease
D. default, kube-public, kube-system, kube-node-leases

Answer: A

Explanation:
https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
Graphical user interface, text Description automatically generated with medium confidence


You can list the current namespaces in a cluster using:

```
kubectl get namespace
```

```
NAME              STATUS   AGE
default           Active   1d
kube-node-lease   Active   1d
kube-public       Active   1d
kube-system       Active   1d
```

Kubernetes starts with four initial namespaces:

- `default` The default namespace for objects with no other namespace
- `kube-system` The namespace for objects created by the Kubernetes system
- `kube-public` This namespace is created automatically and is readable by all users (including those not authenticated). This namespace is mostly reserved for cluster usage, in case that some resources should be visible and readable publicly throughout the whole cluster. The public aspect of this namespace is only a convention, not a requirement.
- `kube-node-lease` This namespace holds Lease objects associated with each node. Node leases allow the kubelet to send heartbeats so that the control plane can detect node failure.

NEW QUESTION 93
What cloud-native construct does a kubernetes pod wrap?

A. Container
B. Virtual Machine (VM)
C. side car process
D. Docker image

Answer: A

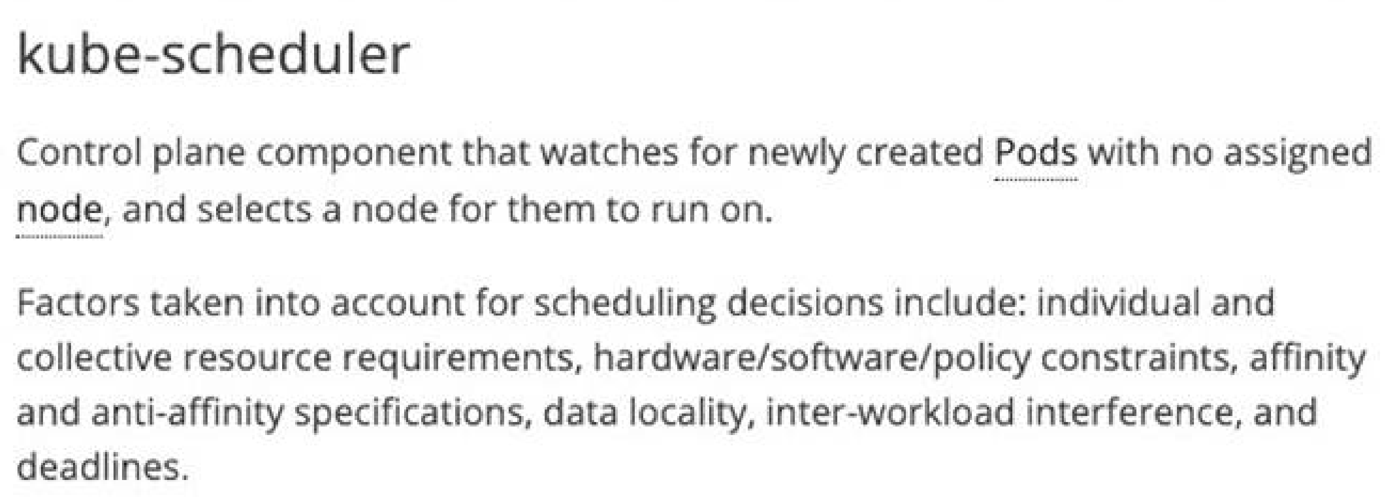
Explanation:
Kubernetes is an orchestrator of containerized apps. However, containers must be wrapped in pods before they can be deployed on kubernetes.

NEW QUESTION 98
Which control plane component is responsible for scheduling pods?

A. kube-proxy
B. kube scheduler
C. kubelet
D. kube api-server

Answer: B

Explanation:
https://kubernetes.io/docs/concepts/overview/components/
Graphical user interface, text, application Description automatically generated



kube-scheduler

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

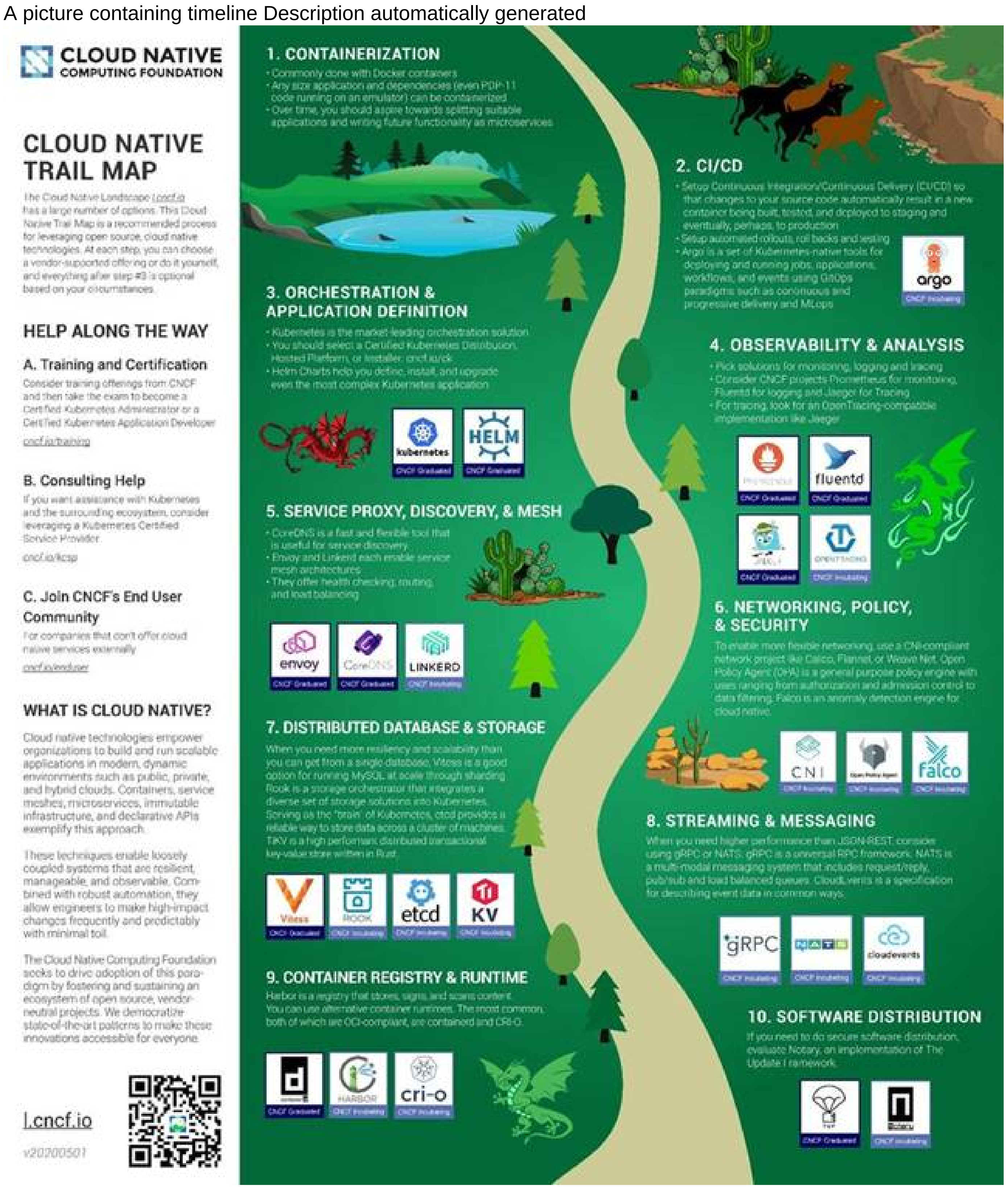
Factors taken into account for scheduling decisions include: individual and collective resource requirements, hardware/software/policy constraints, affinity and anti-affinity specifications, data locality, inter-workload interference, and deadlines.

NEW QUESTION 101
Notary and the update framework leading security projects in CNCF

A. TRUE
B. FALSE

Answer: A

Explanation:
https://github.com/cncf/landscape#trail-map
A picture containing timeline Description automatically generated



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.

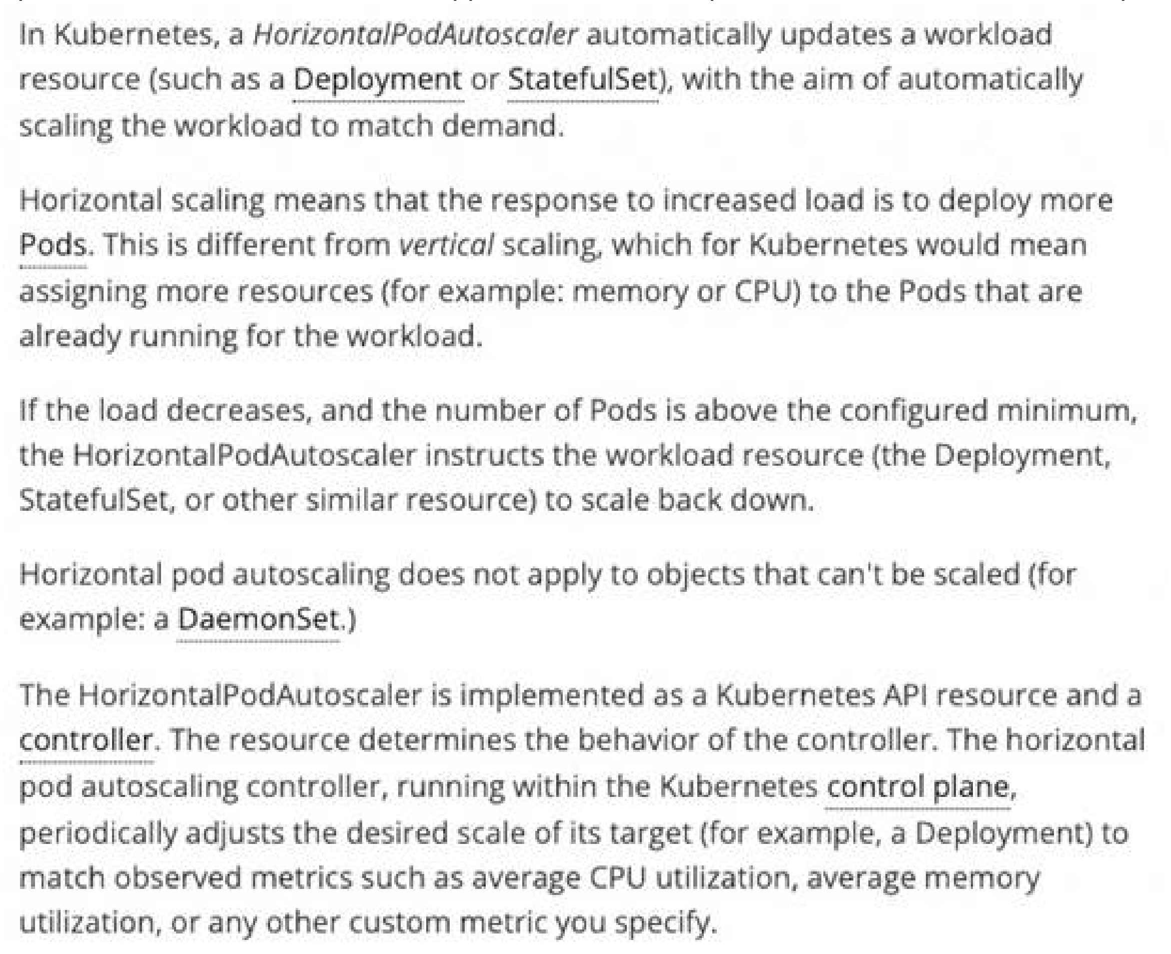
NEW QUESTION 105
What is horizontal scaling?

A. Creating a Deployment
B. Adding resources to existing apps and servers

C. Moving workloads from one server to another
D. Adding additional replicas of apps and servers

Answer: D

Explanation:
https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/ Text, letter Description automatically generated


In Kubernetes, a *HorizontalPodAutoscaler* automatically updates a workload resource (such as a Deployment or StatefulSet), with the aim of automatically scaling the workload to match demand.

Horizontal scaling means that the response to increased load is to deploy more Pods. This is different from *vertical* scaling, which for Kubernetes would mean assigning more resources (for example: memory or CPU) to the Pods that are already running for the workload.

If the load decreases, and the number of Pods is above the configured minimum, the HorizontalPodAutoscaler instructs the workload resource (the Deployment, StatefulSet, or other similar resource) to scale back down.

Horizontal pod autoscaling does not apply to objects that can't be scaled (for example: a DaemonSet.)

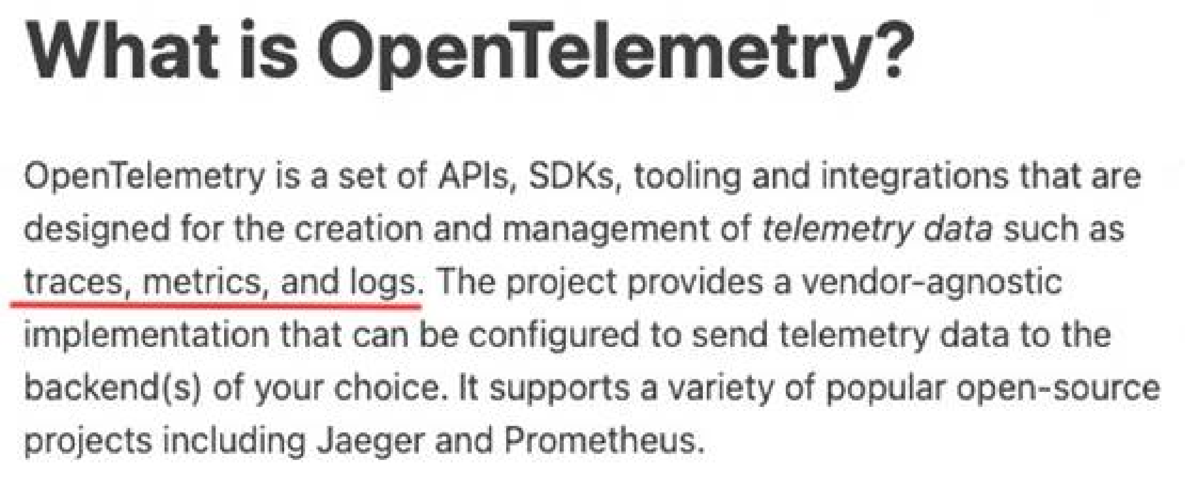
The HorizontalPodAutoscaler is implemented as a Kubernetes API resource and a controller. The resource determines the behavior of the controller. The horizontal pod autoscaling controller, running within the Kubernetes control plane, periodically adjusts the desired scale of its target (for example, a Deployment) to match observed metrics such as average CPU utilization, average memory utilization, or any other custom metric you specify.

NEW QUESTION 110
The three typical opentelemetry data is?

A. Metrics
B. Traces
C. Logs
D. All of the options

Answer: D

Explanation:
https://opentelemetry.io/docs/concepts/data-sources/ Text Description automatically generated


What is OpenTelemetry?

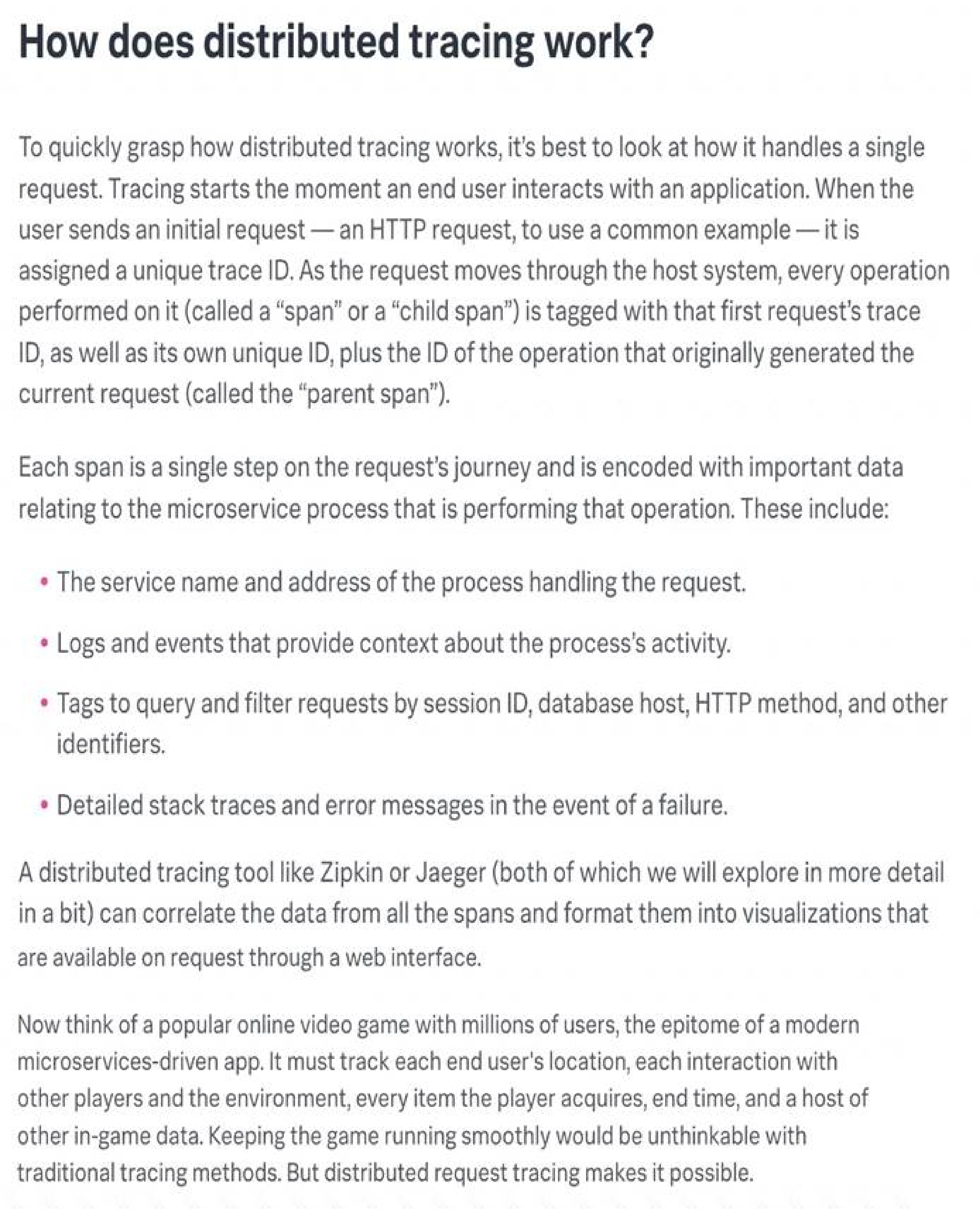
OpenTelemetry is a set of APIs, SDKs, tooling and integrations that are designed for the creation and management of *telemetry data* such as traces, metrics, and logs. The project provides a vendor-agnostic implementation that can be configured to send telemetry data to the backend(s) of your choice. It supports a variety of popular open-source projects including Jaeger and Prometheus.

NEW QUESTION 114
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?

A. Log
B. Span
C. Trace
D. Bucket

Answer: B

Explanation:
https://www.splunk.com/en_us/data-insider/what-is-distributed-tracing.html Text, letter Description automatically generated



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.

NEW QUESTION 115
The 4C's of Cloud Native security

A. Chroot, Compute, Cluster and Container
B. Cluster, Cloud, Compute, and Containers
C. Code, Containers, Compute, and Cloud
D. Cloud, Clusters, Containers, and Code

Answer: D

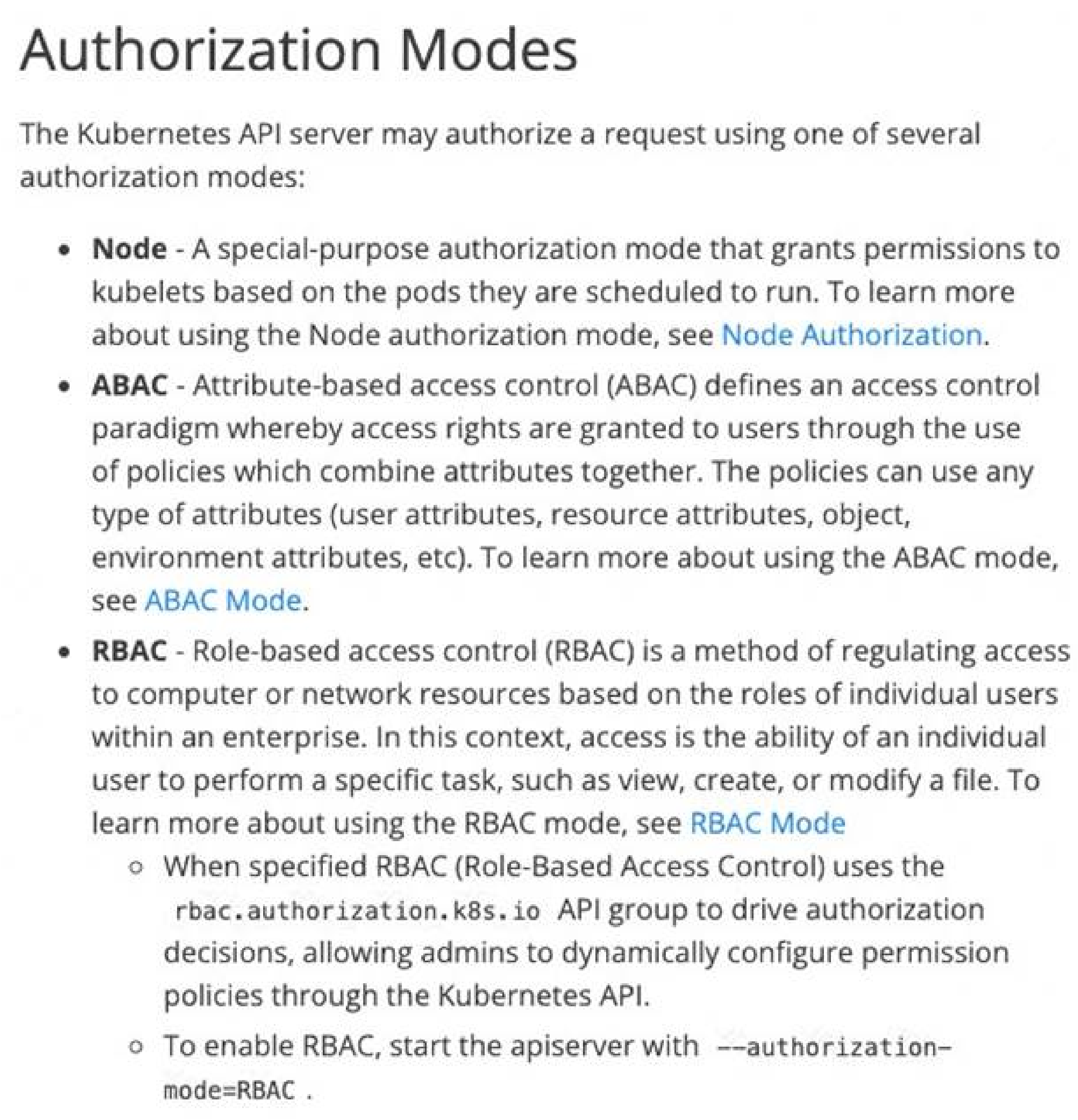
Explanation:
https://kubernetes.io/docs/concepts/security/overview/

NEW QUESTION 116
Which access control component of Kubernetes is responsible for authorization and decides what requestor is allowed to do?

A. Service Account
B. Role-based access control 'RBAC'
C. Deployment

Answer: B

Explanation:
https://kubernetes.io/docs/reference/access-authn-authz/authorization/ Text, letter Description automatically generated



Authorization Modes

The Kubernetes API server may authorize a request using one of several authorization modes:

- **Node** - A special-purpose authorization mode that grants permissions to kubelets based on the pods they are scheduled to run. To learn more about using the Node authorization mode, see Node Authorization.
- **ABAC** - Attribute-based access control (ABAC) defines an access control paradigm whereby access rights are granted to users through the use of policies which combine attributes together. The policies can use any type of attributes (user attributes, resource attributes, object, environment attributes, etc). To learn more about using the ABAC mode, see ABAC Mode.
- **RBAC** - Role-based access control (RBAC) is a method of regulating access to computer or network resources based on the roles of individual users within an enterprise. In this context, access is the ability of an individual user to perform a specific task, such as view, create, or modify a file. To learn more about using the RBAC mode, see RBAC Mode
  - When specified RBAC (Role-Based Access Control) uses the `rbac.authorization.k8s.io` API group to drive authorization decisions, allowing admins to dynamically configure permission policies through the Kubernetes API.
  - To enable RBAC, start the apiserver with `--authorization-mode=RBAC` .

NEW QUESTION 119
Various Container Orchestrator Systems (COS)?

A. Apache Mesos
B. None of the options
C. Docker Swarm
D. Kubernetes

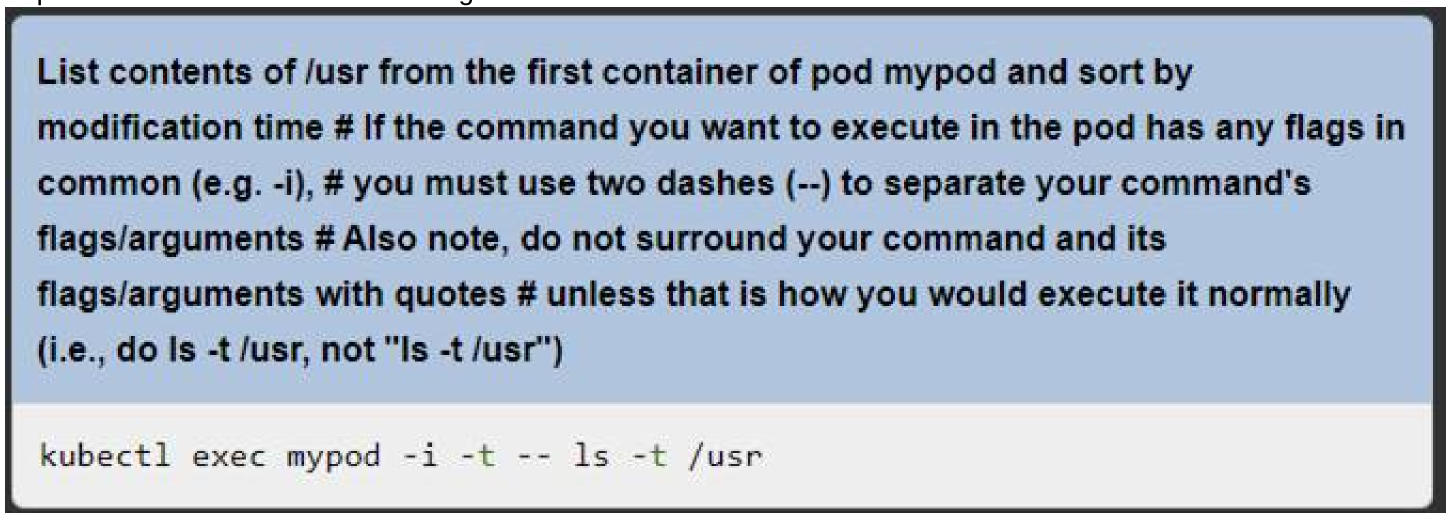
Answer: ACD

NEW QUESTION 121
What is the command used to login to the pod?

A. kubectl login
B. kubectl list
C. kubectl exec
D. kubectl get

Answer: C

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



List contents of /usr from the first container of pod mypod and sort by modification time # If the command you want to execute in the pod has any flags in common (e.g. -i), # you must use two dashes (--) to separate your command's flags/arguments # Also note, do not surround your command and its flags/arguments with quotes # unless that is how you would execute it normally (i.e., do ls -t /usr, not "ls -t /usr")

```
kubectl exec mypod -i -t -- ls -t /usr
```

Text Description automatically generated

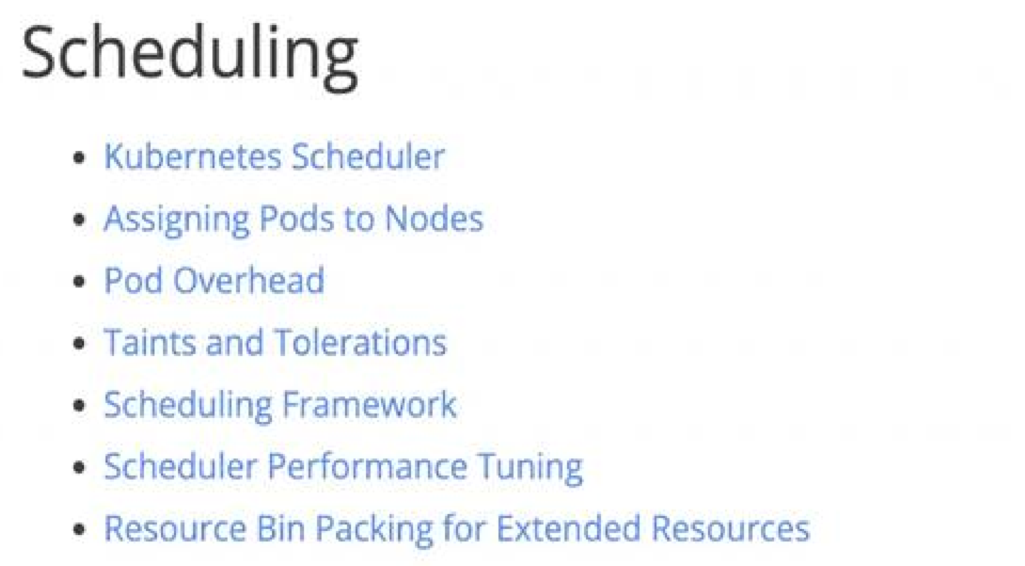
NEW QUESTION 123

What is scheduling in Kubernetes

A. Determining when to execute a cron-job
B. Assigning pods to nodes
C. Joining a new nodes to the clusters
D. Setting a time for automated tasks

Answer: B

Explanation:
https://kubernetes.io/docs/concepts/scheduling-eviction/
Graphical user interface, application Description automatically generated



Scheduling

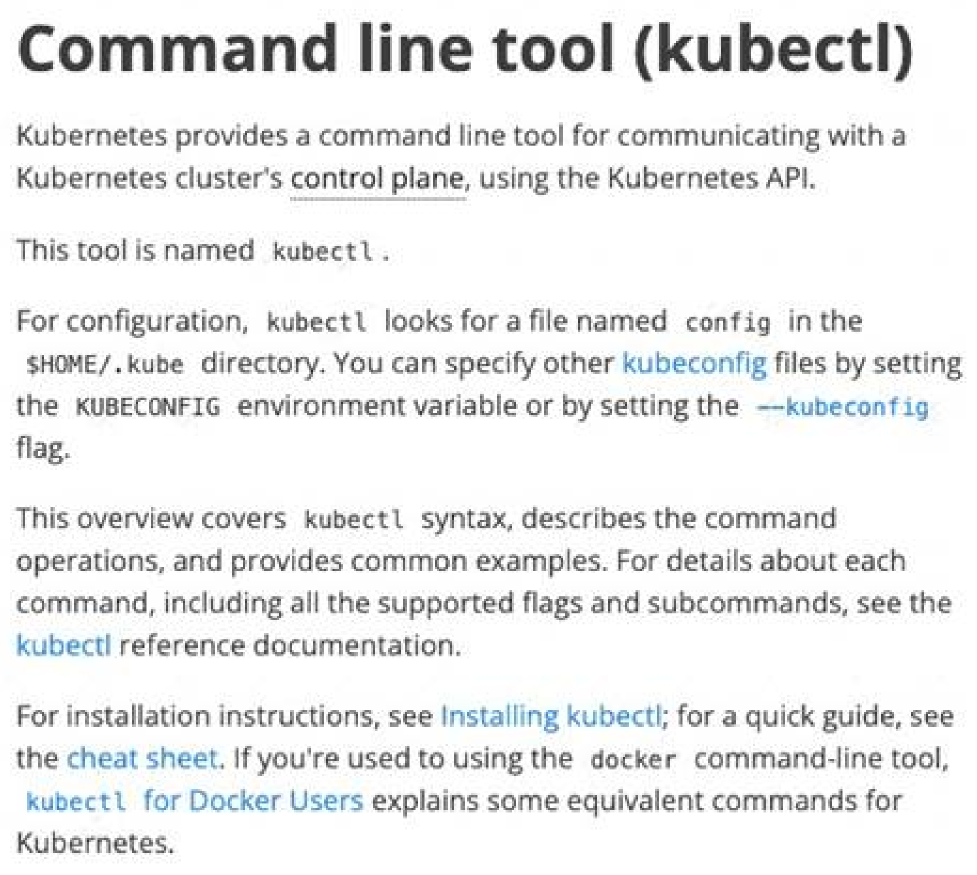
- Kubernetes Scheduler
- Assigning Pods to Nodes
- Pod Overhead
- Taints and Tolerations
- Scheduling Framework
- Scheduler Performance Tuning
- Resource Bin Packing for Extended Resources

NEW QUESTION 124
Which command-line tool is used to interact with the Kubernetes cluster?

A. kube-api
B. kubectl
C. kube-scheduler

Answer: B

Explanation:
https://kubernetes.io/docs/reference/kubectl/
Graphical user interface, text, application, email Description automatically generated



Command line tool (kubectl)

Kubernetes provides a command line tool for communicating with a Kubernetes cluster's control plane, using the Kubernetes API.

This tool is named `kubectl`.

For configuration, `kubectl` looks for a file named `config` in the `$HOME/.kube` directory. You can specify other kubeconfig files by setting the `KUBECONFIG` environment variable or by setting the `--kubeconfig` flag.

This overview covers `kubectl` syntax, describes the command operations, and provides common examples. For details about each command, including all the supported flags and subcommands, see the kubectl reference documentation.

For installation instructions, see Installing kubectl; for a quick guide, see the cheat sheet. If you're used to using the `docker` command-line tool, `kubectl` for Docker Users explains some equivalent commands for Kubernetes.

NEW QUESTION 126
......