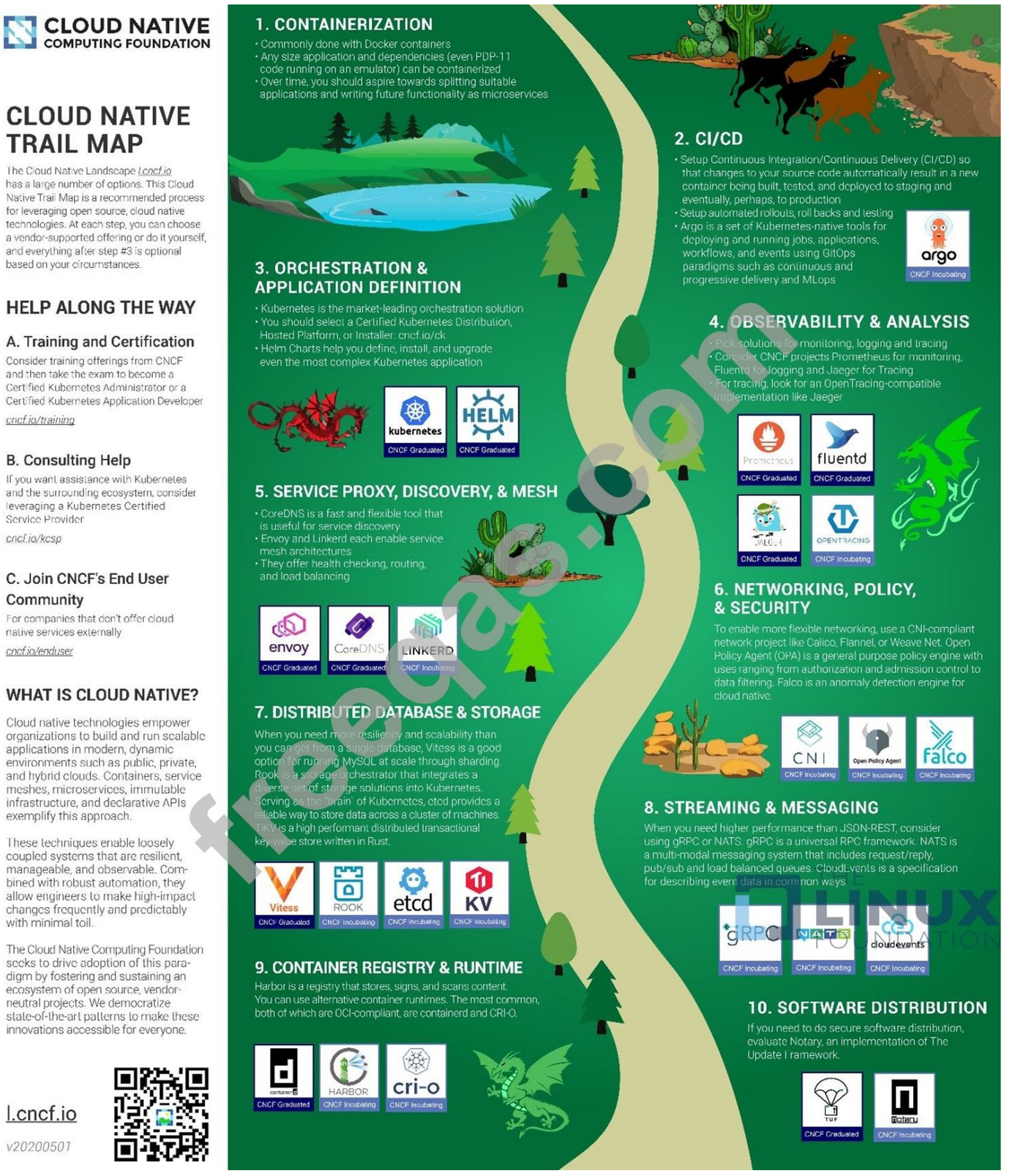
LinuxFoundation.KCNA.v2025-07-29.q103

Exam Code:	KCNA
Exam Name:	Kubernetes and Cloud Native Associate
Certification Provider:	Linux Foundation
Free Question Number:	103
Version:	v2025-07-29
# of views:	144
# of Questions views:	1030
https://www.freeqas.com/qa/Linux-Foundation/KCNA/LinuxFoundation.KCNA.v2025-07-29.q103.html	

NEW QUESTION: 1
Which project is not a dominant CNCF project in the storage landscape?
A. Envoy
B. Vitess
C. Rook
D. TiKV
Answer: A (LEAVE A REPLY)
https://github.com/cncf/landscape#trail-map



NEW QUESTION: 2

You are running a service in Kubernetes that uses a persistent volume. You need to ensure that the volume is automatically cleaned up when the service is deleted. Which of the following approaches would you use?

A. Set the 'persistentVolumeReclaimPolicy' to 'Delete' in the PersistentVolumeClaim

B. Delete the PersistentVolumeClaim manually when the service is deleted

C. Use a DaemonSet to run a cleanup script that removes the volume when the service is deleted
D. Configure a CronJob to periodically check for unused volumes and delete them
E. There is no automatic way to clean up a persistent volume when a service is deleted; it needs to be done manually.
Answer: (SHOW ANSWER)
Setting the 'persistentVolumeReclaimPolicy' to 'Delete' in the PersistentVolumeClaim ensures that the volume is automatically deleted when the PVC is deleted. This is the most straightforward and recommended approach for cleaning up persistent volumes in Kubernetes.

NEW QUESTION: 3
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 (LEAVE A REPLY)
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").

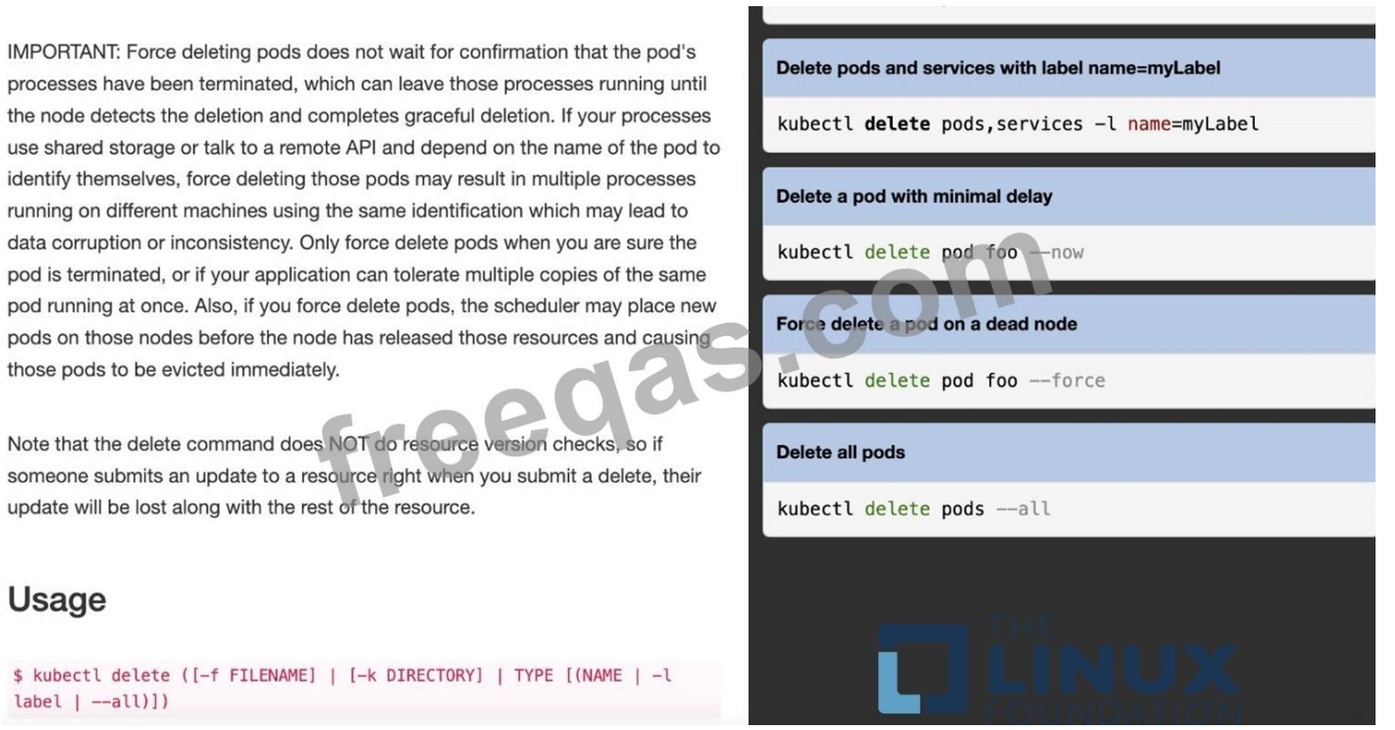
Each span is a single step on the request's journey and is encoded with important data relating to the microservice process that is performing that operation. These include:

- 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.

NEW QUESTION: 4
'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
Answer: B (LEAVE A REPLY)
https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#delete



NEW QUESTION: 5
Your application relies on a backend database service. Using Istio, you want to configure a circuit breaker pattern to prevent cascading failures if the database becomes unresponsive. How would you implement this?
A. Create a custom Istio VirtualService with a 'destinationRule' specifying a fallback service in case of failure
B. Utilize Istio's 'fault injection' feature to simulate failures and test the circuit breaker
C. Configure the 'retry' policy in the Istio configuration to automatically retry failed requests
D. Use the 'timeout' setting in the Istio configuration to limit the duration of requests to the database
E. Deploy a separate health check pod to monitor the database service and trigger the circuit breaker if needed
Answer: (SHOW ANSWER)
Istio's VirtualService and DestinationRule features allow you to configure fallback services. In the case of the database being unavailable, you can define a fallback service or mechanism to handle the request. Option B helps test circuit breaker behavior but doesn't implement it. Option C could be used for retries, but doesn't address the circuit breaker pattern. Option D is related to request timeouts, not circuit breaking. Option E is a manual approach, while Istio provides a more integrated solution.

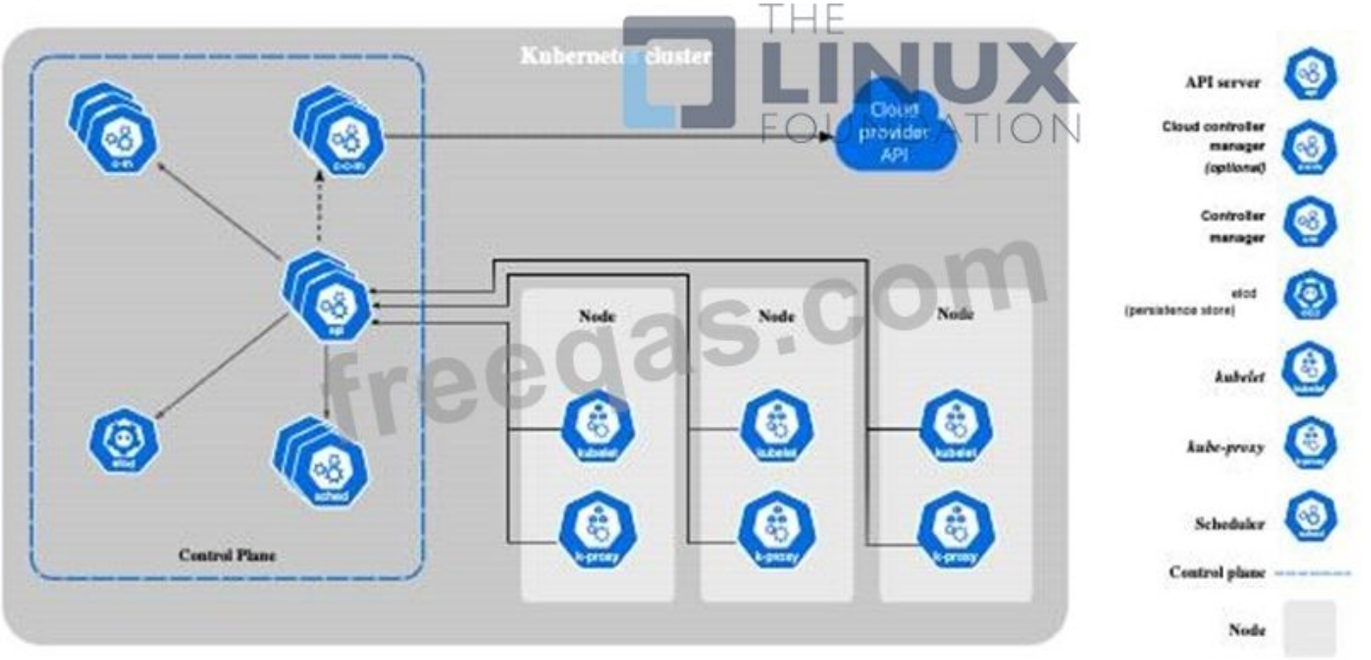
NEW QUESTION: 6
Which of the following is not the part of Kubernetes Control Plane?
A. kube scheduler
B. etcd (pronounce: esty-d)

C. kube api-server

D. kube-proxy

Answer: D (LEAVE A REPLY)

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



NEW QUESTION: 7

What feature is used for selecting the container runtime configuration?

A. RuntimeClass

B. RuntimeContainer

C. Runtime

D. RuntimeConfig

Answer: A (LEAVE A REPLY)

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.

NEW QUESTION: 8

Consider a scenario where you need to configure a Kubernetes storage class with a specific storage provisioner and access modes. How would you achieve this using a YAML configuration?

A.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-storage-class
provisioner: kubernetes.io/gce-pd
accessModes:
  - ReadWriteOnce
```

B.

```
apiVersion: storage.k8s.io/v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  persistentVolumeReclaimPolicy: Retain
```

C.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-container
        image: my-image:latest
        volumeMounts:
        - name: my-volume
          mountPath: /data
      volumes:
      - name: my-volume
        persistentVolumeClaim:
          claimName: my-pvc
```

D.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: my-storage-class
```

E.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-storage-class
provisioner: kubernetes.io/gce-pd
parameters:
  type: pd-standard
```

Answer: (SHOW ANSWER)

Option A correctly defines a StorageClass named "my-storage-class" with the provisioner "kubernetes.io/gce-pd" and an access mode "ReadWriteOnce". StorageClasses define how volumes should be provisioned, specifying the storage provisioner and other attributes. Option B defines a PersistentVolume, not a StorageClass. Option C defines a Deployment, not a StorageClass. Option D defines a PersistentVolumeClaim, not a StorageClass. Option E defines a StorageClass, but does not include the accessModes property.

NEW QUESTION: 9

Consider the following Kubernetes YAML configuration for a Deployment:

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

What is the purpose of the 'replicas' field in this configuration, and how does it affect the Deployment? The 'replicas'' field specifies the number of Pods that should be created and maintained by the Deployment. It controls the desired number of running instances of the application.

A. The 'replicas' field defines the minimum number of Pods that should be available at all times. It ensures that the application remains available even if some Pods fail.

B. The 'replicas' field specifies the maximum number of Pods that can be created by the Deployment. It limits the amount of resources that the application can consume.
C. The 'replicas' field specifies the initial number of Pods that should be created when the Deployment is deployed. It does not affect the number of Pods that are running after the initial deployment.
D. The 'replicas' field specifies the number of nodes in the Kubernetes cluster. It determines the number of Pods that can be scheduled
Answer: A (LEAVE A REPLY)
The 'replicas' field in a Deployment specifies the number of Pods that should be created and maintained by the Deployment. It controls the desired number of running instances of the application. If a Pod fails, the Deployment will automatically create a new Pod to replace it, ensuring that the desired number of replicas is maintained.

NEW QUESTION: 10
Which of the following Kubernetes components is responsible for managing the lifecycle of Pods, including scheduling, creation, and deletion?
A. Kubelet
B. etcd
C. kubectl
D. amserver
E. Controller Manager
Answer: (SHOW ANSWER)
The Kubernetes Controller Manager is responsible for managing the lifecycle of Pods. It monitors the state of Pods and ensures that they are running as intended. It also handles pod creation, deletion, and scaling based on the defined Deployment or ReplicaSet.

NEW QUESTION: 11
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 (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/

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:

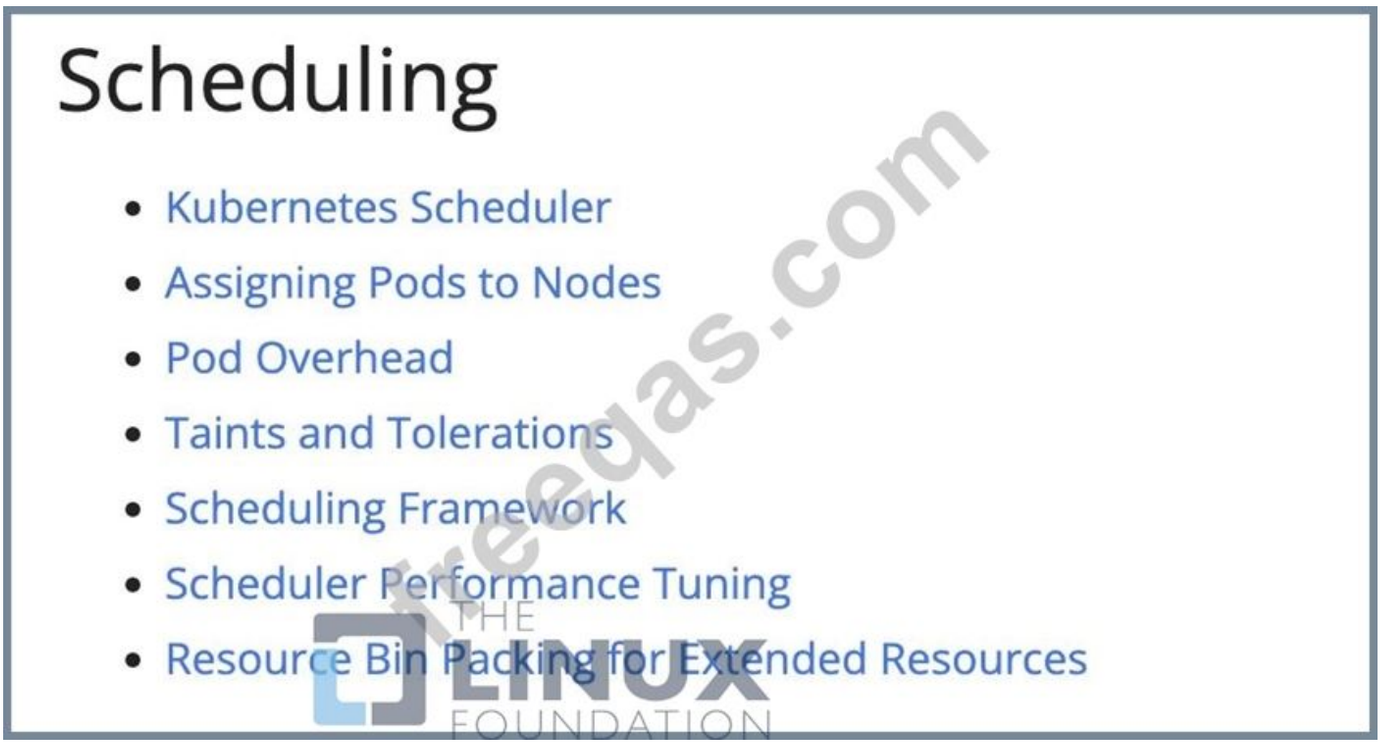
- `default` The default namespace for objects with no other namespace
- `kube-system` The namespace for objects created by the Kubernetes system
- `kube-public` This namespace is created automatically and is readable by all users (including those not authenticated). This namespace is mostly reserved 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: 12
You have a Kubernetes Deployment that defines 3 replicas of your application. During a deployment, one of the replicas fails to start. What happens to the deployment?
A. The deployment remains in a failed state.
B. The deployment continues, but with only 2 healthy replicas.
C. Kubernetes automatically restarts the failed replica.
D. The deployment scales down to 2 replicas.
E. The deployment fails and rolls back to the previous version.
Answer: (SHOW ANSWER)
Kubernetes' self-healing capabilities ensure high availability. If a pod fails, Kubernetes will automatically restart it, attempting to bring the deployment back to the desired replica count.

NEW QUESTION: 13
Which of the following are valid ways to define resource requests for a pod? (Select all that apply)
A. Using the resources field in the pod's YAML definition.
B. Using the requests' field in the container's YAML definition.
C. Using the 'limits' field in the container's YAML definition.
D. Using the •affinity' field in the pod's YAML definition.
E. Using the •tolerations' field in the pod's YAML definition.
Answer: A,B,C (LEAVE A REPLY)
Resource requests and limits are defined within the container's definition (under the resourceS field). The •requests' field specifies the minimum resources a container needs to run, while the 'limits' field defines the maximum resources the container can use. The 'affinity' and •tolerations' fields are used for controlling pod scheduling preferences and tolerating specific node conditions, but they don't directly define resource requirements.

NEW QUESTION: 14
You are deploying a web application that requires a specific version of Node.js. How would you ensure that the correct Node.js version is installed on all nodes in your Kubernetes cluster?
A. Install Node.js manually on each node before deploying the application.
B. Use a Kubernetes init container to install Node.js before the main application container starts.
C. Define a custom Kubernetes resource to manage Node.js installation across the cluster.
D. Use a Dockerfile to include the specific Node.js version in the container image for the web application.
E. Use a Helm chart to manage the installation of Node.js on all nodes in the cluster.
Answer: (SHOW ANSWER)
The most effective way to ensure the correct Node.js version is used is to include it within the container image using a Dockerfile. By defining the Node.js version in the Dockerfile, you guarantee that every container built from that image will have the required version, eliminating dependency issues across nodes in your cluster. The container image becomes a self-contained unit, ensuring consistency and simplifying deployment.

NEW QUESTION: 15
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 (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/scheduling-eviction/



NEW QUESTION: 16
You are running a Kubernetes cluster with multiple nodes. Your application's Pods need to be scheduled across different nodes for high availability. Which Kubernetes concept allows you to control the distribution of Pods across the cluster?
A. Pod Security Policy
B. Node Affinity
C. Service
D. Deployment
E. Ingress
Answer: (SHOW ANSWER)
Node Affinity lets you define rules to control where your Pods are scheduled. You can specify hard (required) or soft (preferred) preferences for nodes based on labels or other criteria. This helps to distribute Pods across the cluster for better resource utilization and fault tolerance.

NEW QUESTION: 17
Consider the following Kubernetes YAML definition for a Deployment:

Which of the following statements about this Deployment are true?

A. The Deployment will create 3 Pods with the label 'app: nginx'.

B. The Deployment will only create Pods with the label 'app: nginx' if a corresponding Service exists.

C. The Deployment will use the 'nginx:l. 14.2' image to create the Pods.

D. The Deployment will expose the Pods on port 80.

E. The Deployment will create a Service that exposes the Pods on port 80.

Answer: A,C (LEAVE A REPLY)

The Deployment will create 3 Pods with the label 'app: nginx' because the 'selector' and 'template.metadata.labels' sections specify this. The Deployment will use the 'nginx:l. 14.2' image to create the Pods as defined in the 'spec.template.spec.containers' section. The Deployment itself does not create a Service. You would need to create a separate Service resource to expose the Pods on port 80.

NEW QUESTION: 18

You are implementing a new security policy for your Kubernetes cluster. The policy requires that all pods running in the cluster must authenticate with a specific identity provider before they are allowed to access any resources. Which Kubernetes component is responsible for enforcing this authentication policy?

A. kubelet

B. etcd

C. kube-proxy

D. kube-apiserver

E. kubectl

Answer: D (LEAVE A REPLY)

The kube-apiserver component acts as the central control plane for Kubernetes. It handles all communication and requests from other components, including authentication and authorization. It enforces security policies by verifying credentials and granting access based on configured rules.

NEW QUESTION: 19

You are managing a Kubernetes cluster using GitOps principles. Which of the following tools would you use to monitor the state of your cluster and trigger actions based on changes in your Git repository?

A. kubectl

B. Flux

C. Helm

D. Prometheus

Answer: (SHOW ANSWER)

Flux is a GitOps operator that continuously monitors your Git repository for changes and applies those changes to your Kubernetes cluster. It is specifically designed for GitOps workflows, ensuring your cluster's state reflects the desired configuration in your Git repository.

NEW QUESTION: 20

You are migrating a monolithic application to a microservices architecture on Kubernetes. You choose to use Istio to manage the communication between these new services. Which of the following is NOT a benefit of adopting Istio in this scenario?

A. Istio simplifies the process of migrating existing code to a microservices architecture

B. Istio provides a consistent way to handle network calls and security across all microservices

C. Istio helps in achieving better scalability and fault tolerance for microservices

D. Istio's control plane simplifies the deployment and management of microservices

E. Istio allows for easier debugging and troubleshooting of complex microservice interactions

Answer: (SHOW ANSWER)

While Istio provides benefits like centralized traffic management, security, and observability for microservices, it doesn't automatically simplify the process of migrating existing monolithic code. The migration itself requires careful refactoring and architectural changes, which Istio

complements but doesn't replace. Options B, C, D, and E are all valid benefits of using Istio for microservices.

NEW QUESTION: 21
What are cluster-wide objects
A. Service and Pods
B. Volumes and Nodes
C. ConfigMaps and Secrets
Answer: (SHOW ANSWER)
https://kubernetes.io/docs/concepts/overview/working-with-objects/_print/

NEW QUESTION: 22
You are developing a serverless application using Azure Functions that processes real-time streaming dat
a. How would you ensure reliable and efficient data ingestion from a Kafka topic to your Azure Functions?
A. Use Azure Event Hubs to connect Kafka to Azure Functions-
B. Use Azure Service Bus to act as a message broker between Kafka and Azure Functions.
C. Configure Azure Functions to poll Kafka topics directly-
D. Use a custom connector to integrate Kafka with Azure Functions.
E. Utilize Azure Data Factory to create pipelines for data movement between Kafka and Azure Functions.
Answer: A (LEAVE A REPLY)
Azure Event Hubs is the recommended approach for connecting Kafka to Azure Functions for real-time streaming data ingestion. It offers high throughput and scalability, making it ideal for handling large volumes of streaming events. Service Bus (B) is suitable for message queuing but not primarily designed for streaming data. Directly polling Kafka topics (C) can be inefficient and might not scale well. Custom connectors (D) can be complex and might lack the required functionality for streaming data ingestion. Data Factory (E) is more focused on data movement and transformation, not real-time streaming

NEW QUESTION: 23

You have deployed an application in Kubernetes with a container image that has known vulnerabilities. Which of the following security measures is MOST effective in mitigating the risk of these vulnerabilities?

A. Deploying the application in a private Kubernetes cluster.

B. Using a container security scanner to identify and fix vulnerabilities.

C. Disabling Kubernetes RBAC and granting full access to all users.

D. Running all containers with the '--privileged' flag enabled.

E. Deploying the application in a separate namespace.

Answer: B (LEAVE A REPLY)

Using a container security scanner to identify and fix vulnerabilities is the most effective measure for mitigating the risk of known vulnerabilities in your container images. These scanners analyze your images for known security issues and provide recommendations for patching or upgrading them. This helps you proactively address potential vulnerabilities and improve the overall security of your Kubernetes applications.

NEW QUESTION: 24

What is the command used to scale the application?

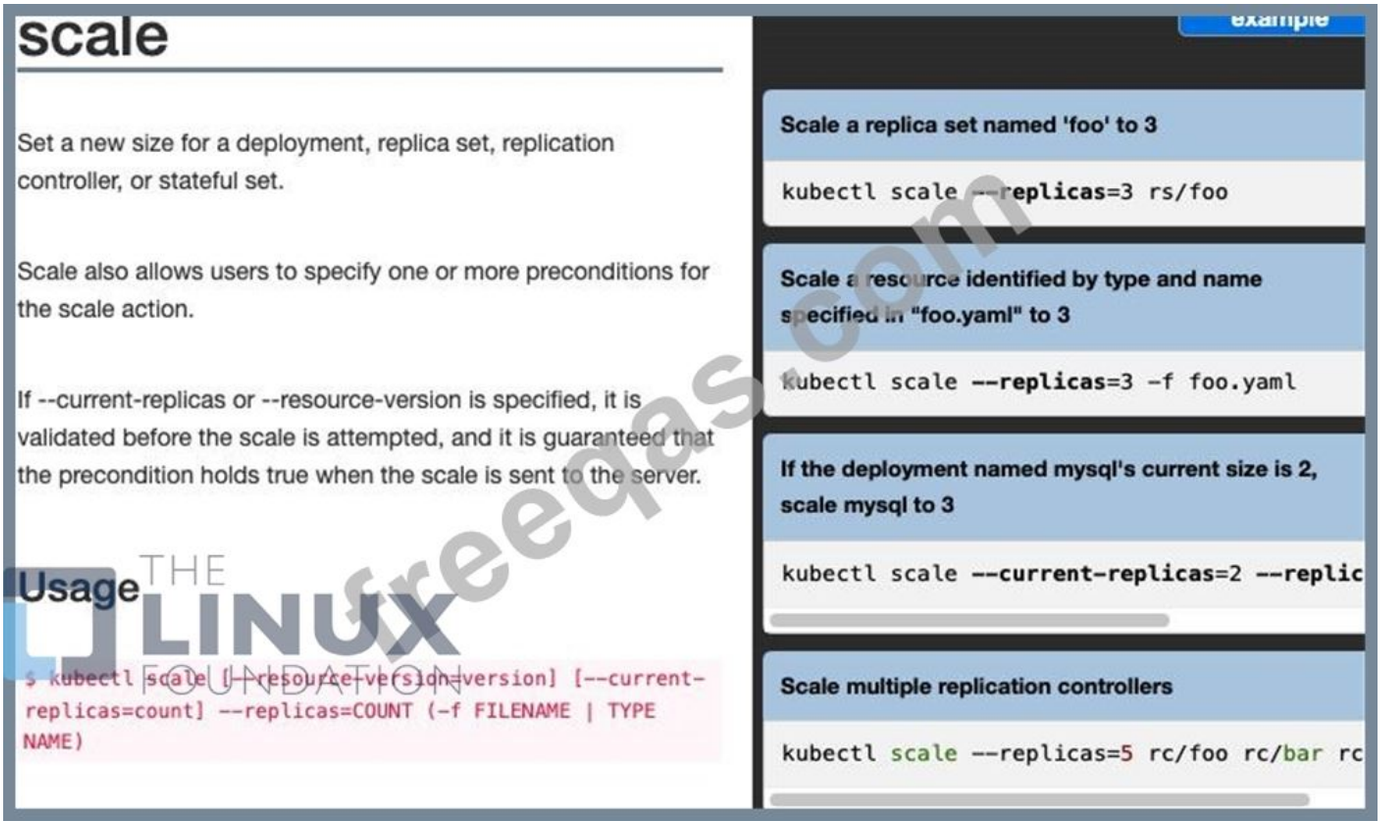
A. kubectl run

B. kubectl explain

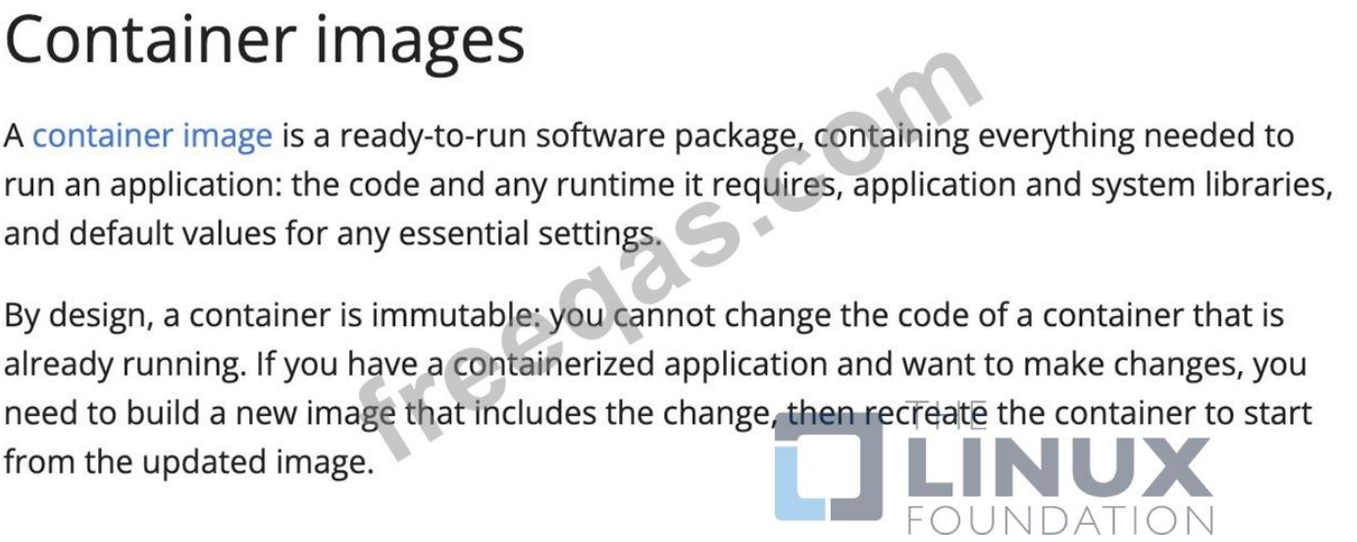
C. kubectl scale

Answer: C (LEAVE A REPLY)

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



NEW QUESTION: 25
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
Answer: D (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/containers/#container-images



NEW QUESTION: 26
Which component of the Kubernetes architecture is responsible for scheduling Pods to nodes and ensuring that they are running as intended?
A. etcd
B. kube-apiserver
C. kube-scheduler
D. kube-controller-manager
E. kubelet
Answer: (SHOW ANSWER)
The kube-scheduler is the component responsible for scheduling Pods to nodes. It takes into account resource availability node affinity, and other factors to ensure optimal resource utilization and application performance.

NEW QUESTION: 27
You need to configure a HorizontalPodAutoscaler (HPA) to scale a deployment based on custom metrics. The metric data is being published by a Prometheus server. How can you integrate Prometheus with the HPA for custom metric scaling?
A. Use the '-metrics-sources flag for the HPA command to specify the Prometheus endpoint.
B. Create a Kubernetes Custom Resource Definition (CRD) to define the Prometheus metric.

C. Use the '-metrics' flag in the Deployment YAML file to link the HPA to Prometheus metrics.
D. Configure a Kubernetes Service to expose the Prometheus metrics to the HPA.
E. Use a custom Kubernetes controller that connects the HPA to Prometheus.
Answer: (SHOW ANSWER)
You need a custom controller to bridge the gap between the HPA and Prometheus. The controller will fetch metrics from Prometheus and provide them to the HPA. You can use the '-metrics' flag in the HPA YAML file to specify the custom metrics and their sources (including the Prometheus endpoint).

NEW QUESTION: 28
Your organization is migrating a legacy web application to a serverless architecture on Azure. You want to ensure that the application's state is persisted during and after migration. Which Azure service would be most suitable for storing and managing the application's state?
A. Azure Cosmos DB
B. Azure Storage Account
C. Azure Service Bus
D. Azure Event Hubs
E. Azure SQL Database
Answer: A,E (LEAVE A REPLY)
Azure Cosmos DB and Azure SQL Database are the most suitable services for storing and managing application state during and after migration to a serverless architecture on Azure- Both offer persistent storage and support various data models. Cosmos DB is a fully managed, globally distributed, multi-model database service, while Azure SQL Database is a relational database service- Storage Accounts (B) are primarily for blob storage and may not be the best fit for managing application state. Service Bus (C) is for message queuing, and Event Hubs (D) are for real-time event ingestion- While these services might play a role in the serverless architecture, they are not the primary choice for persisting application state-

NEW QUESTION: 29
What framework allows developers to write code without worrying about the servers and operating systems they will run on?
A. Kubernetes
B. Virtualization
C. Serverless
D. Docker
Answer: C (LEAVE A REPLY)

NEW QUESTION: 30
Your Kubernetes cluster has limited resources, and you need to ensure that the pods for your application are effectively scheduled and managed. Which of the following approaches would contribute to resource optimization?

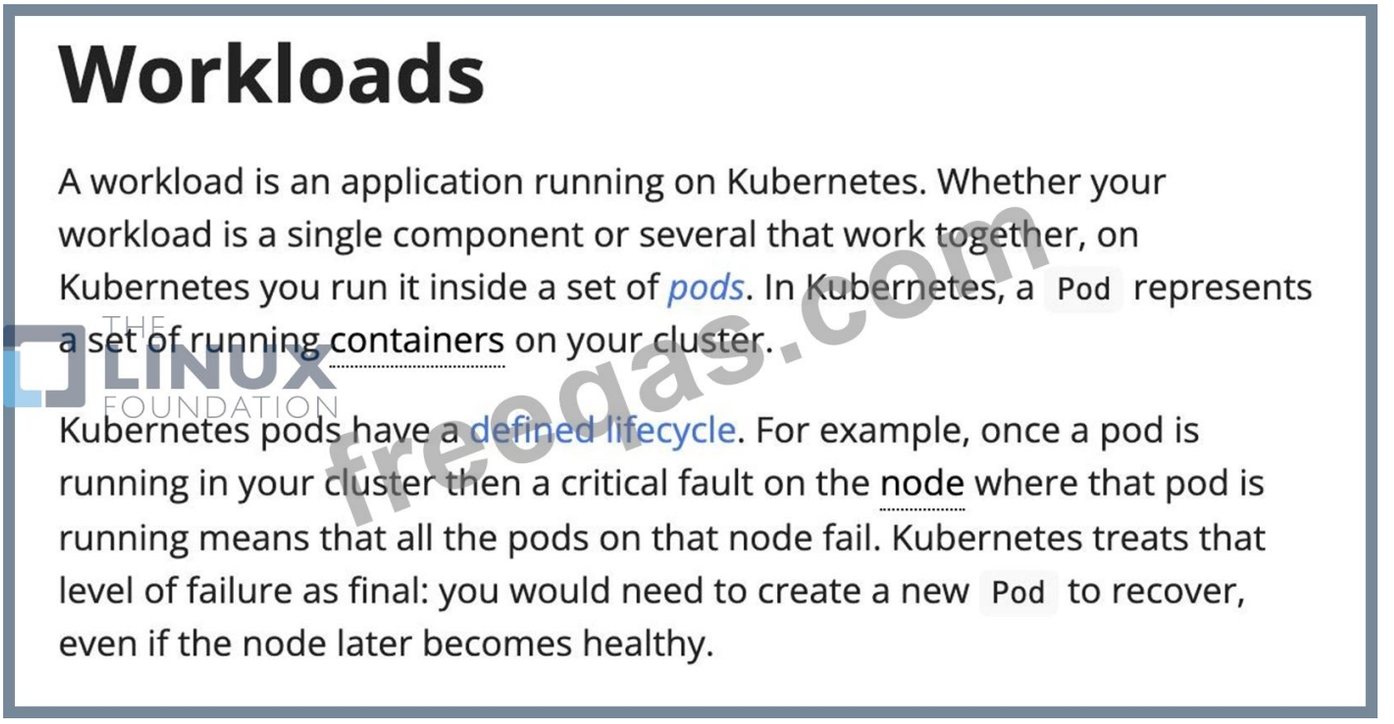
A. Deploying all pods on a single node for centralized management
B. Setting resource requests and limits for pods based on their actual resource consumption
C. Using DaemonSets to ensure that a pod runs on every node in the cluster
D. Scheduling all pods with the highest priority to guarantee their resources
E. Disabling resource quotas to allow pods to utilize all available resources
Answer: (SHOW ANSWER)
Setting resource requests and limits for pods is crucial for resource optimization. By defining these values, you provide Kubernetes with guidance on how much resources each pod needs and helps prevent resource starvation for other applications. Requests are the minimum resources a pod needs, while limits prevent a pod from consuming more than its assigned quota.

NEW QUESTION: 31
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 (LEAVE A REPLY)
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: 32
Consider a Kubernetes deployment where you have a microservice responsible for user authentication. You want to ensure that any communication to this service is encrypted and secure. What Kubernetes feature or open standard can help achieve this?
A. Pod Security Policies (PSPs)
B. NetworkPolicy
C. TLS/SSL
D. Open Policy Agent (OPA)
E. Kubernetes RBAC
Answer: C (LEAVE A REPLY)

TLS/SSL is a standard protocol for encrypting communication between applications. You can configure your authentication service and its clients to use TLS/SSL to ensure secure communication between them. While other options like PSPs, NetworkPolicy, and RBAC are related to security, they don't specifically address the encryption aspect of communication-

NEW QUESTION: 33
A _____ is an application running on kubernetes.
A. node
B. pod
C. workload
D. container
Answer: C (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/workloads/



NEW QUESTION: 34
Which of the following command is used to get detailed information about the pod?
A. kubectl info
B. kubectl get
C. kubectl describe
D. kubectl explain
Answer: C (LEAVE A REPLY)
https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#describe

NEW QUESTION: 35
You are using Flux to manage your Kubernetes cluster with GitOps. You need to ensure that all deployments are automatically rolled back to the previous version if they fail. Which configuration option in Flux would you use to achieve this?

A.
```
imagePullPolicy: Always
```

B.
```
rollbackConfig: { failed: true }
```

C.
```
strategy: { type: Recreate }
```

D.
```
updateStrategy: { type: RollingUpdate }
```

E.
```
autoscaling: { minReplicas: 1 }
```

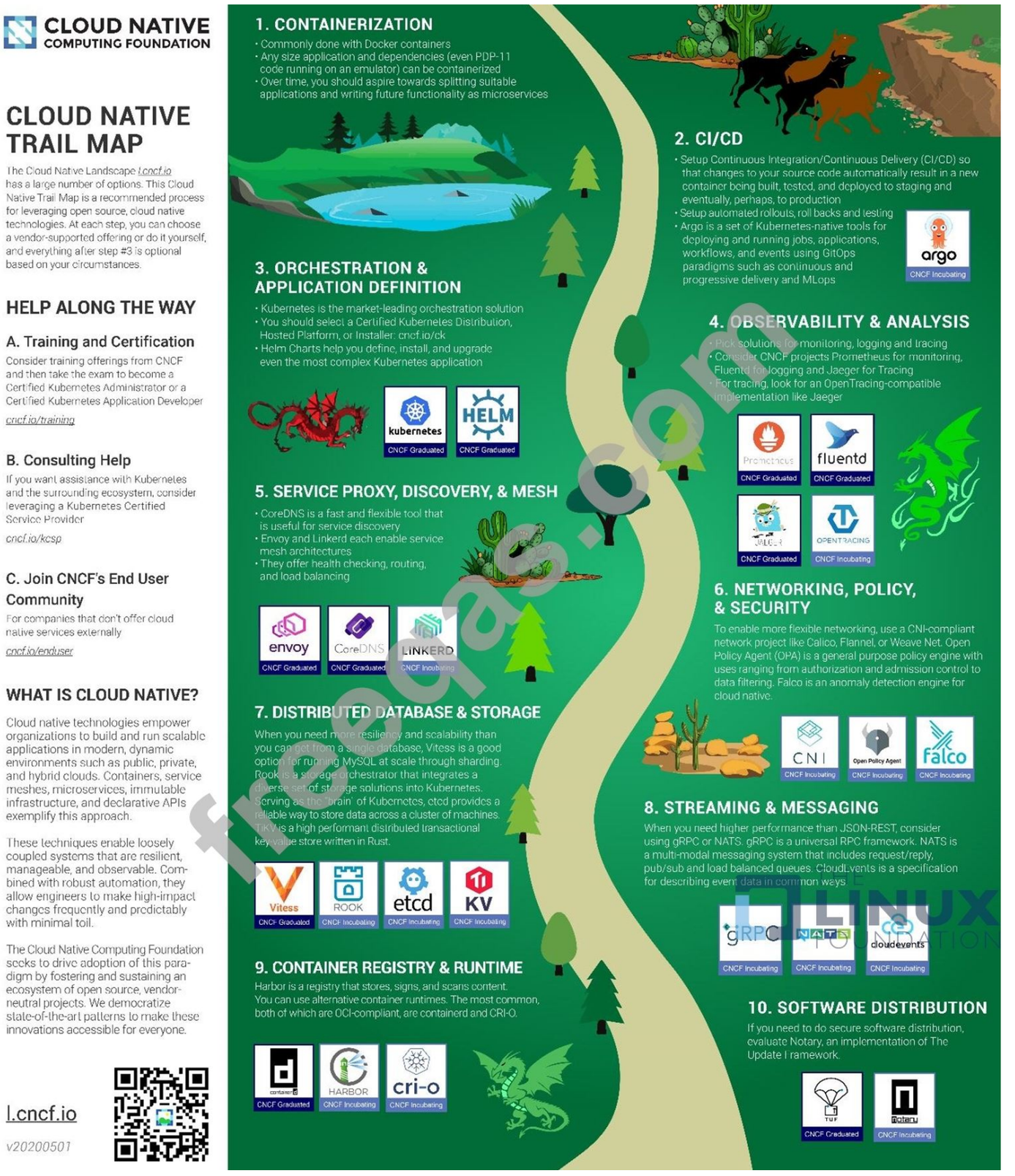
Answer: B (LEAVE A REPLY)
The •rollbackConfig• option in Flux allows you to define rollback strategies for deployments. Setting 'failed: true' ensures that if a deployment fails, it will be automatically rolled back to the previous working version, minimizing downtime and ensuring stability.

NEW QUESTION: 36
You are building a web application that needs to serve traffic through a load balancer. Which Kubernetes resource should you use to expose the application service externally?
A. Deployments
B. Services
C. Ingress
D. ConfigMaps
E. StatefulSets
Answer: (SHOW ANSWER)

Ingress is a Kubernetes resource used to define rules for how external traffic is routed to different services within your cluster. It allows you to configure load balancers, SSL termination, and other advanced routing features.

NEW QUESTION: 37
What is a commonly used package manager for kubernetes applications?
A. npm
B. apt
C. helm
D. kubernetes manifest
Answer: C (LEAVE A REPLY)
https://helm.sh/

NEW QUESTION: 38
Fluentd is the only way to export logs from Kubernetes cluster or applications running in cluster
A. True
B. False
Answer: B (LEAVE A REPLY)
https://github.com/cncf/landscape#trail-map



NEW QUESTION: 39

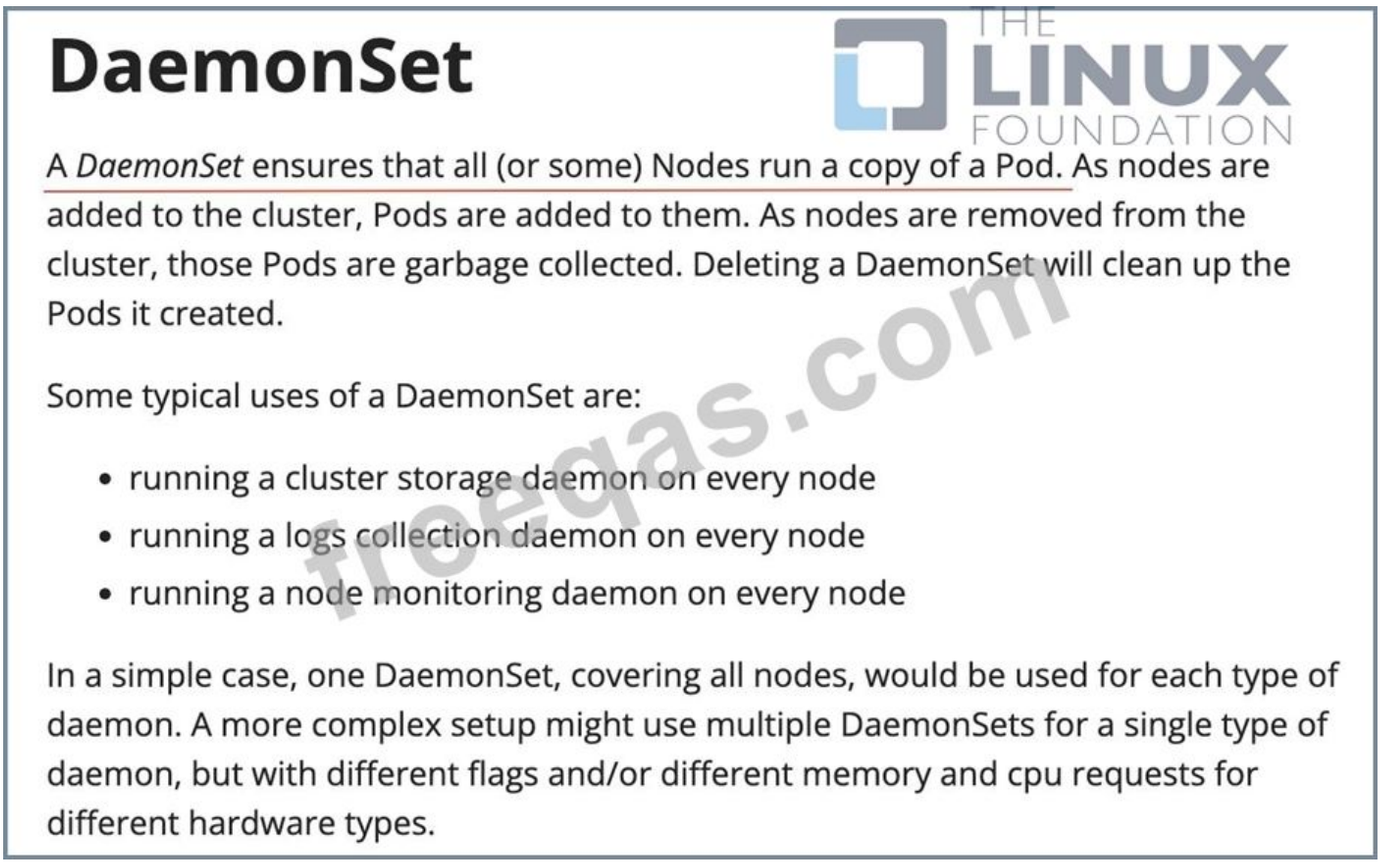
You are implementing a GitOps workflow for a complex application with multiple microservices. To manage dependencies between these microservices and ensure their correct deployment order, which approach would be most suitable?

A. Use a separate Git repository for each microservice and manually coordinate their deployments.

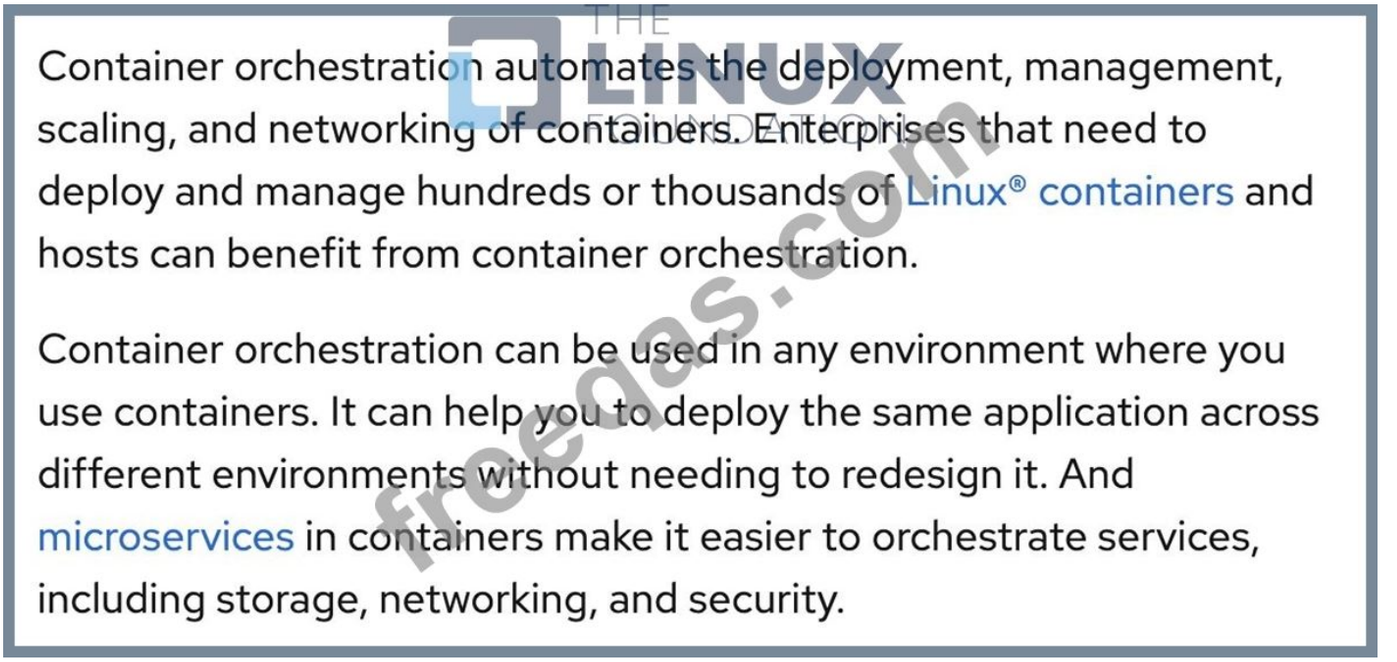
B. Use a single Git repository for all microservices and define deployment dependencies using Helm charts.
C. Use a single Git repository for all microservices and leverage a GitOps tool like ArgoCD or Flux to manage dependencies with features like resource dependencies and deployment order.
D. Use a Kubernetes Operator to automate the deployment and management of the microservices, ensuring their dependencies are met.
E. Use a separate Git repository for each microservice and configure a CIICD pipeline to manage their dependencies.
Answer: C (LEAVE A REPLY)
Option C provides the most suitable approach for managing dependencies between multiple microservices in a GitOps workflow. Using a single repository for all microservices and leveraging a GitOps tool like ArgoCD or Flux allows you to define resource dependencies and specify the deployment order for microservices, ensuring a consistent and predictable deployment process.

NEW QUESTION: 40
You need to create a Kubernetes service that exposes a TCP-based application on port 8080. You want the service to be accessible from external clients. Which type of service should you create?
A. ClusteriP
B. LoadBalancer
C. NodePort
D. ExternalName
E. Headless
Answer: B (LEAVE A REPLY)
The *LoadBalancer• service type is the most suitable for exposing your TCP-based application on port 8080 to external clients. It will automatically create a load balancer in the cloud provider's infrastructure, allowing external access to your application. Option 'A' (ClusterlP) only allows access from within the cluster. Option 'C' (NodePort) exposes the service on a specific port on each node, making it accessible via the node's IP address. Option 'D' (ExternalName) is for exposing services that are already externally accessible using a DNS name. Option 'E' (Headless) is for services where you want to access Pods directly by their names, which is not the case here.

NEW QUESTION: 41
What Kubernetes resource would allow you to run one Pod on some of your Nodes?
A. DaemonSet
B. ClusterSet
C. Deployment
D. ReplicaSet
Answer: (SHOW ANSWER)
https://kubernetes.io/docs/concepts/workloads/controllers/daemonset/



NEW QUESTION: 42
What is container orchestration?
A. Packaging code and all of its dependencies into a single executable
B. Adding code to a container image so it can run as a container
C. Using automation to manage containers
D. Spinning a new containers to replace old ones
Answer: (SHOW ANSWER)
https://www.redhat.com/en/topics/containers/what-is-container-orchestration



NEW QUESTION: 43
You are running a resource-intensive application in Kubernetes. You want to reduce costs without impacting performance. Which of the following approaches would be most effective?
A. Decrease the number of replicas for the application's deployment.
B. Reduce the resource requests and limits for the application's pods.
C. Implement vertical pod autoscaling to adjust the resource allocation based on actual usage.
D. Optimize the application code to reduce resource consumption.
E. Migrate the application to a different cloud provider with lower pricing.
Answer: C,D (LEAVE A REPLY)
For resource-intensive applications, the most effective cost reduction strategies focus on optimizing resource usage and application efficiency. Vertical pod autoscaling (VPA) dynamically adjusts resource allocation based on actual usage, ensuring pods have sufficient resources without overprovisioning. Optimizing the application code itself to reduce resource consumption is crucial, as it can significantly impact overall cost. While decreasing replicas or reducing requests/limits might seem tempting, they can negatively impact performance. Migrating to a different cloud provider might offer cost advantages, but it's a complex and potentially disruptive process.

NEW QUESTION: 44
Which Kubernetes resource creates Kubernetes Jobs?
A. JobFactory
B. CronJob
C. Task
D. JobDeployment
Answer: (SHOW ANSWER)
https://kubernetes.io/docs/concepts/workloads/controllers/cron-jobs/

CronJob

FEATURE STATE: Kubernetes v1.21 [stable]

A *CronJob* creates Jobs on a repeating schedule.

One CronJob object is like one line of a *crontab* (cron table) file. It runs a job periodically on a given schedule, written in Cron format.

NEW QUESTION: 45
Which project in this list is a leading project in the observability space?

A. Jaeger

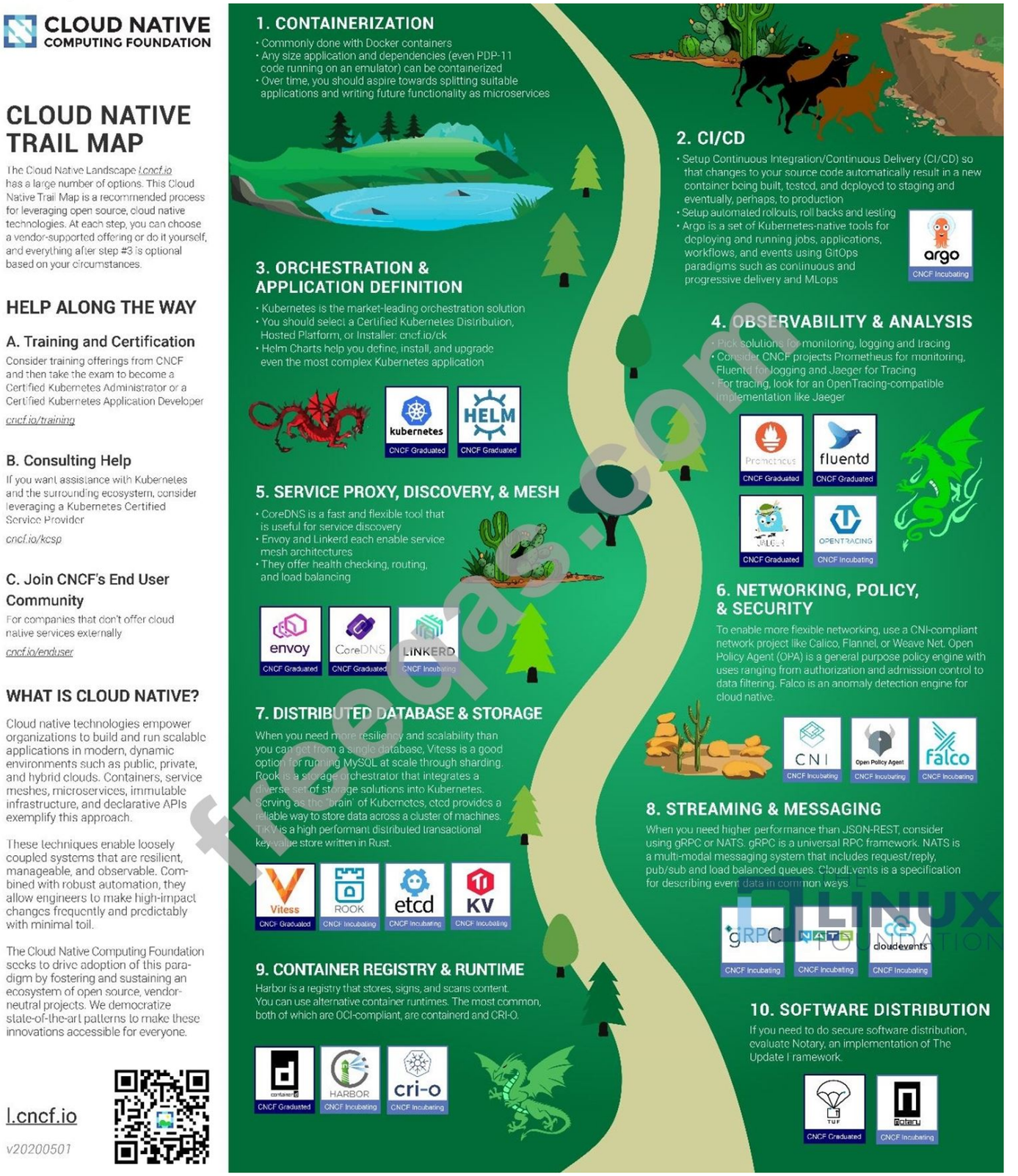
B. Vitess

C. Argo

D. Kubernetes

Answer: (SHOW ANSWER)

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



NEW QUESTION: 46
You are deploying a microservices application in Kubernetes. Each microservice has its own deployment and service. How can you enforce network policy rules to restrict communication between these microservices, allowing only authorized communication?
A. Use Kubernetes namespaces to isolate the microservices and restrict communication between them.
B. Configure network policies at the cluster level to define rules for all pods in the cluster.
C. Use network namespaces to isolate the microservices and restrict communication between them.
D. Use network policies to define rules for specific pods or groups of pods, allowing communication only between authorized microservices.
E. Configure the deployments to use specific ports and restrict access to those ports through the services.
Answer: D (LEAVE A REPLY)
The most effective way to enforce network policy rules for microservices in Kubernetes is to use network policies. Network policies allow you to define specific rules for communication between pods or groups of pods. By creating network policies, you can specify which pods are allowed to communicate with each other, based on their labels, namespaces, or other criteria. This allows you to restrict communication between microservices, ensuring that only authorized communication is allowed. While namespaces can help to isolate microservices logically, they do not provide granular network control. Configuring ports and services can restrict access to specific services, but it does not provide the level of control needed to enforce communication rules between microservices.

NEW QUESTION: 47
Which of the following is used to request storage in Kubernetes?
A. PersistentVolume 'PV'
B. PersistentVolumeClaim 'PVC'
C. Container Storage Interface 'CSI'
D. StorageClasses
Answer: B (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/storage/persistent-volumes/

NEW QUESTION: 48
You have a microservice that relies on a separate authentication service. You want to implement fine-grained authorization using Istio based on request attributes like user roles or specific headers. What Istio feature would you use?
A. Istio's 'RequestAuthentication' feature
B. Istio's 'AuthorizationPolicy' feature
C. Istio's 'TrafficManagement' feature
D. Istio's 'Telemetry' feature
E. Istio's 'ServiceEntry' feature
Answer: B (LEAVE A REPLY)
Istio's AuthonzationPolicy feature allows you to define granular authorization rules based on request attributes, including headers, user identities, and other context. This enables you to control access to microservices based on specific conditions. Option A is for authentication, not authorization. Option C is for traffic management. Option D is for telemetry. Option E is for service discovery and configuration.

NEW QUESTION: 49
You're building a new cloud-native application using gRPC for communication between microservices. You want to use Jaeger for distributed tracing to track the flow of requests through your microservices. What should you do to integrate Jaeger with your gRPC application?
A. Configure a custom gRPC interceptor to intercept gRPC calls and emit tracing data to Jaeger.
B. Use Jaeger's built-in support for gRPC tracing, which automatically intercepts gRPC calls and sends data to Jaeger.
C. Modify your gRPC client and server code to explicitly log trace information using Jaeger's logging libraries.
D. Configure Istio to automatically inject tracing into gRPC calls and send the data to Jaeger.
E. Use a dedicated tracing proxy that sits between your gRPC services and intercepts traffic to send tracing data to Jaeger.
Answer: B,D (LEAVE A REPLY)
The correct answers are B and D . Jaeger provides built-in support for tracing gRPC applications, and Istio can also be used to automatically inject tracing into gRPC calls. B: Use Jaeger's built-in support for gRPC tracing, which automatically intercepts gRPC calls and sends data to Jaeger. Jaeger offers libraries and integration points for popular frameworks like gRPC. Using Jaeger's gRPC tracing capabilities, you can automatically instrument your gRPC applications and capture tracing data without needing to manually modify your code extensively. D: Configure Istio to automatically inject tracing into gRPC calls and send the data to Jaeger. Istio, a service mesh, offers automatic tracing capabilities for gRPC applications. By configuring Istio, you can enable tracing for gRPC calls and send tracing data to Jae er streamlinin the tracin rocess and reducin code modifications within our services. The other options are incorrect or less efficient: A: Configure a custom gRPC interceptor to intercept gRPC calls and emit tracing data to Jaeger. While you can create a custom interceptor, Jaeger's built-in support and Istio's automatic tracing

mechanisms are more convenient and often preferred for gRPC tracing. C: Modify your gRPC client and server code to explicitly log trace information using Jaeger's logging libraries. Manual code modifications are less efficient compared to using Jaeger's built-in support or Istio's automatic tracing. E: Use a dedicated tracing proxy that sits between your gRPC services and intercepts traffic to send tracing data to Jaeger. While a tracing proxy could work, it adds an extra layer of complexity and potential performance overhead compared to built-in support or Istio's integration.

NEW QUESTION: 50
Which kubernetes object do deployments use behind the scenes when they need to scale pods?
A. Horizontal pod autoscaler
B. ReplicaSets
C. kubectl
D. Replication controller
Answer: (SHOW ANSWER)
https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

NEW QUESTION: 51
You need to deploy a new version of your application to Kubernetes without causing any downtime or disruption to users. What open standard would you use to facilitate this process?
A. Open Service Mesh (OSM)
B. Containerd
C. Open Policy Agent (OPA)
D. Kubernetes Rolling Update
E. CloudEvents
Answer: D (LEAVE A REPLY)
Kubernetes Rolling Update allows you to deploy new versions of your application gradually- It replaces existing pods with the new version one by one, ensuring that there is always at least one pod running the previous version, minimizing downtime. This approach helps maintain the application's availability during updates-

NEW QUESTION: 52
You have a Kubernetes cluster with multiple applications deployed. Each application is instrumented to emit logs, metrics, and traces. You want to use a single dashboard to visualize

the performance of all applications in a unified view. What are the possible approaches to achieve this?

A. Use Prometheus to aggregate metrics from all applications and create a single dashboard.

B. Use Grafana to create a single dashboard that queries data from Prometheus and Jaeger.

C. Use Loki for log aggregation and create a single dashboard in Grafana to visualize logs, metrics, and traces.

D. Use a custom application to collect and aggregate data from Prometheus, Jaeger, and Loki, and then visualize the data on a custom dashboard.

E. Configure Kubernetes to forward logs, metrics, and traces to a centralized observability platform like CloudWatch or Stackdriver.

Answer: (SHOW ANSWER)

All of the provided options can contribute to achieving a unified dashboard for visualizing the performance of multiple applications. Each option has its strengths and weaknesses: A: Use Prometheus to aggregate metrics from all applications and create a single dashboard. Prometheus is a powerful tool for collecting and aggregating metrics. You can use Prometheus's query language to fetch data from multiple applications and create a centralized dashboard in Grafana or a custom application. B: Use Grafana to create a single dashboard that queries data from Prometheus and Jaeger. Grafana is a popular dashboarding tool that can visualize data from multiple sources. It can query metrics from Prometheus and tracing data from Jaeger to create a unified view. C: Use Loki for log aggregation and create a single dashboard in Grafana to visualize logs, metrics, and traces. Loki is a log aggregation system that can collect logs from various sources, including Kubernetes. By integrating Loki with Grafana, you can visualize logs, metrics, and traces on a single dashboard. D: Use a custom application to collect and aggregate data from Prometheus, Jaeger, and Loki, and then visualize the data on a custom dashboard. You can build a custom application to collect data from Prometheus, Jaeger, and Loki and then create a custom dashboard using a framework like React or Vue.js. This allows you to have full control over the data aggregation and visualization process. E: Configure Kubernetes to forward logs, metrics, and traces to a centralized observability platform like CloudWatch or Stackdriver. Cloud-based observability platforms like Amazon CloudWatch or Google Stackdriver provide a centralized platform for collecting, aggregating, and visualizing data from multiple applications. These platforms often have pre- built dashboards and alerting capabilities, making it easy to monitor and analyze data from different applications in a unified view. The best approach depends on your specific needs, resources, and preferred tools. You can choose a combination of these options to meet your requirements.

NEW QUESTION: 53

How should folks new to the cloud native ecosystem, go about learning the different aspects of the ecosystem?

A. by signing up the CNCF slack

B. by reading the Kubernetes documentation

C. by looking at the cloud native landscape

D. by looking at the cloud native trail-map
Answer: D (LEAVE A REPLY)
https://github.com/cncf/landscape#trail-map

NEW QUESTION: 54
What cloud-native construct does a kubernetes pod wrap?
A. Container
B. Virtual Machine (VM)
C. side car process
D. Docker image
Answer: A (LEAVE A REPLY)
Kubernetes is an orchestrator of containerized apps. However, containers must be wrapped in pods before they can be deployed on kubernetes.

NEW QUESTION: 55
What is the difference between a Service and an Ingress in Kubernetes?
A. Services provide external access to your application, while Ingress provides internal access.
B. Services manage the lifecycle of Pods, while Ingress manages the lifecycle of Deployments.
C. Services expose applications running within the cluster, while Ingress provides routing and load balancing for external traffic.
D. Services are used for scheduling Pods to nodes, while Ingress is used for managing the communication between Pods and services.
E. Services are used for managing the state of Pods, while Ingress is used for managing the health of Pods.
Answer: (SHOW ANSWER)
In Kubernetes Services provide a way to expose applications running within the cluster, while Ingress provides a mechanism for routing and load balancing external traffic to your applications. Services are used to make Pods accessible within the cluster, while Ingress enables users outside the cluster to access your applications.

NEW QUESTION: 56
What is OPA?
A. Open Permission Agent
B. Online Policy Audit
C. Open Policy Agent
D. Offline Policy Accessor
Answer: C (LEAVE A REPLY)
https://www.cncf.io/projects/open-policy-agent-opa/

NEW QUESTION: 57
Various Container Orchestrator Systems (COS)?
A. Docker Swarm
B. Apache Mesos
C. Kubernetes
D. None of the options
Answer: (SHOW ANSWER)

NEW QUESTION: 58
You might need to run a stateless application in kubernetes, and you want to be able to scale easily and perform rolling updates. What kubernetes resource type can you use to do this
A. Dameon set
B. Replica set
C. Deployment
D. pod
E. service
F. Stateful set
Answer: (SHOW ANSWER)
https://kubernetes.io/docs/concepts/workloads/controllers/deployment/

Deployments

A *Deployment* provides declarative updates for Pods and ReplicaSets.

You describe a *desired state* in a Deployment, and the Deployment Controller changes the actual state to the desired state at a controlled rate. You can define Deployments to create new ReplicaSets, or to remove existing Deployments and adopt all their resources with new Deployments.

> **Note:** Do not manage ReplicaSets owned by a Deployment. Consider opening an issue in the main Kubernetes repository if your use case is not covered below.

NEW QUESTION: 59
What is the primary function of the Kubernetes control plane?
A. Schedules Pods to nodes based on resource availability and node affinity
B. Manages the lifecycle of Pods, ensuring that they are running as intended.
C. Provides a secure and reliable connection between the Kubernetes control plane and nodes.
D. Provides a central point of control and management for the Kubernetes cluster.
E. Enables communication between Pods and services within a Kubernetes cluster.
Answer: D (LEAVE A REPLY)
The Kubernetes control plane is responsible for managing the overall state and behavior of the cluster- It provides a central point of control, ensuring that all components are running as expected and that the cluster is operating efficiently.

NEW QUESTION: 60
What is a benefits of Kubernetes federation?
A. Low latency
B. Avoids scalability limits on pods and nodes
C. Creates highly available clusters in different regions
Answer: A,B,C (LEAVE A REPLY)

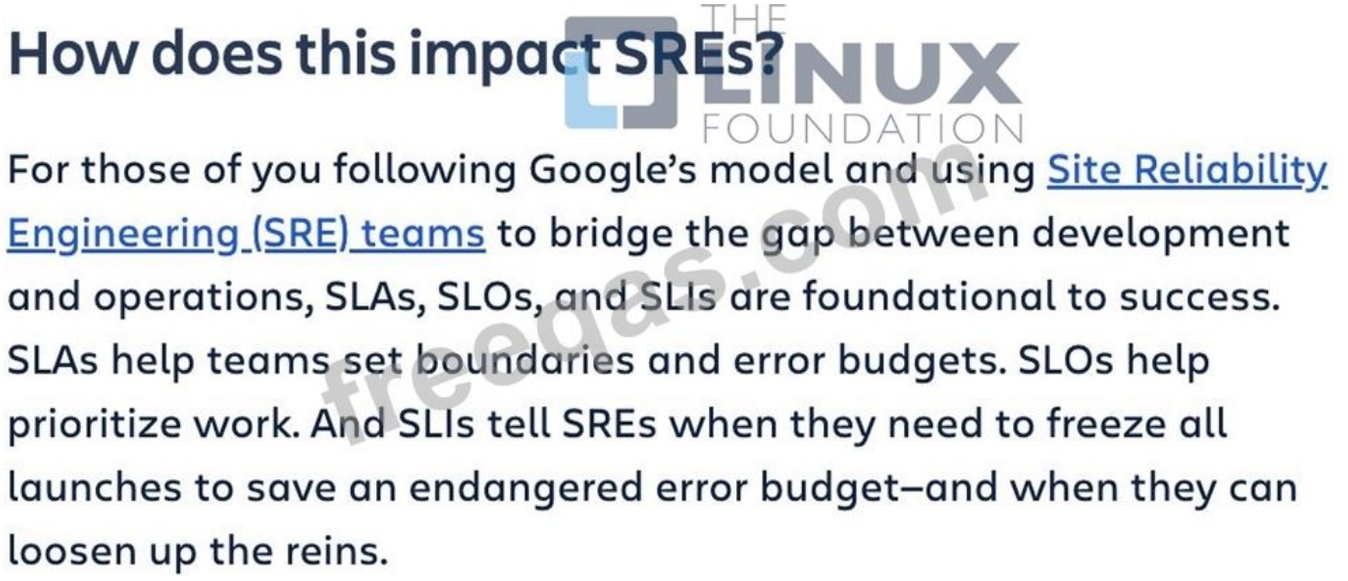
NEW QUESTION: 61
You are using Istio to manage traffic flow in a microservices architecture. Your application consists of three services: 'service-A', 'service-B', and 'service-C', where 'service-A' depends on 'service-B' and 'service-B' depends on 'service-C'. You want to implement a canary rollout for

'service-B' to test a new version before rolling it out to all users. How would you configure Istio to achieve this?
A. Use the 'istio.io/canary' annotation on 'service-B' and configure the rollout percentage
B. Create a new Istio VirtualService for 'service-B' and route a percentage of traffic to the new version
C. Deploy the new version of 'service-B' with a different Kubernetes namespace and use Istio to route traffic to the new namespace
D. Configure 'service-A' to automatically retry requests if they fail to 'service-B' and route to the new version
E. Use the 'istio.io/route' annotation on 'service-B' to specify a weighted distribution of traffic to different versions
Answer: B (LEAVE A REPLY)
Istio's VirtualService allows you to define routing rules that can direct a percentage of traffic to a specific version of a service- This enables you to perform canary rollouts by gradually introducing the new version of 'service-B' to a subset of users- Option A is not a valid Istio configuration option- Option C creates unnecessary complexity Option D is related to retries, not canary rollouts. Option E is closer but doesn't specify the percentage split.

NEW QUESTION: 62
You are deploying a pod that requires access to a specific storage volume attached to a particular node. Which Kubernetes feature can you utilize to guarantee the pod is scheduled only on that specific node?
A. Node affinity
B. Node anti-affinity
C. Pod affinity
D. Pod anti-affinity
E. Taints and tolerations
Answer: A,E (LEAVE A REPLY)
You can achieve this by using either •nodeAffinity• or •taints and tolerationS: 'nodeAffinity•• Similar to the previous question, define 'requiredDuringSchedulingIgnoredDuringExecution• to enforce scheduling on the specific node. 'Taints and TolerationS: Apply a taint on the specific node that reflects the volume availability. Then, configure the pod to tolerate that specific taint, ensuring it can only be scheduled on the node with the matching taint. While •podAffinity• and

•podAntiAffinity' are useful for grouping or distributing pods, they do not directly guarantee scheduling on a specific node based on volume availability.

NEW QUESTION: 63
Which role is responsible of creating service level indicator 'SLI', service level objective 'SLO', & Service Level Agreements 'SLA'
A. Site reliability engineer 'SRE'
B. DevOps
C. GitOps
D. Security and compliance engineer
E. Developer
Answer: A (LEAVE A REPLY)
https://www.atlassian.com/incident-management/kpis/sla-vs-slo-vs-sli



NEW QUESTION: 64
You are implementing a GitOps workflow for your Kubernetes cluster. Which of the following best practices should you consider?
A. Store all your Kubernetes configurations in a single Git repository, regardless of their purpose or scope.
B. IJse a separate Git repository for each Kubernetes resource, such as deployments, services, and ingress.
C. Organize your Git repository with a clear directory structure for different namespaces and applications.
D. IJse Git tags to mark specific releases of your Kubernetes configurations.
E. Ensure that all changes to your Kubernetes configurations are reviewed and approved before being deployed.
Answer: C,D,E (LEAVE A REPLY)
Option C promotes organization and maintainability by structuring your repository, while option D enables tracking specific releases through Git tags. Option E emphasizes the importance of code

reviews and approvals to ensure the integrity of your deployments. These practices collectively contribute to a robust and well-managed GitOps workflow.

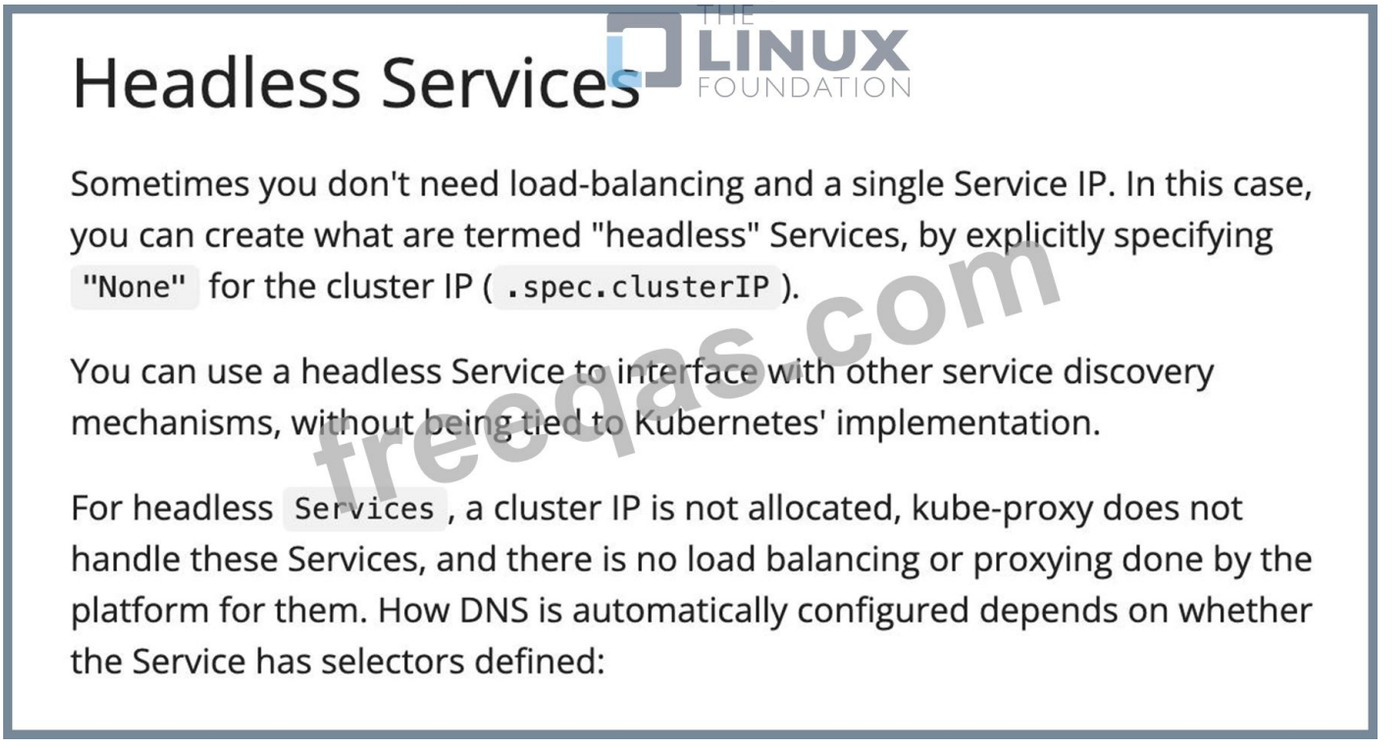
NEW QUESTION: 65
Which control plane component is responsible for scheduling pods?
A. kube-proxy
B. kube scheduler
C. kubelet
D. kube api-server
Answer: B (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/overview/components/

NEW QUESTION: 66
You're developing a new microservice for an application running on Kubernetes. The service requires access to a database hosted in a separate Kubernetes cluster. How would you ensure secure communication between your microservice and the database?
A. Use a shared secret stored in an environment variable within the microservice pod
B. Configure a service mesh to handle encryption and authentication between the services.
C. Set up a VPN connection between the two Kubernetes clusters-
D. Use an API gateway to proxy requests between the services.
E. Configure a Kubernetes Ingress resource to route traffic to the database
Answer: B (LEAVE A REPLY)
A service mesh like Istio or Linkerd provides a layer of abstraction that can handle encryption, authentication, and authorization bewveen microservices within and across Kubernetes clusters. It allows for secure communication without exposing sensitive credentials directly within the microservice code.

NEW QUESTION: 67
You are running a stateless application on Kubernetes and want to avoid persistent storage. Which of the following options is suitable?
A. Use a PersistentVolumeClaim with a storage class that provides persistent storage.
B. Mount a HostPath volume to access the host machine's filesystem.
C. Use an emptyDir volume to create a temporary directory within the pod.

D. Utilize a ConfigMap to store application configuration data.
E. Create a Local Persistent Volume for each pod.
Answer: C (LEAVE A REPLY)
For stateless applications, an emptyDir volume is the most suitable choice as it provides a temporary directory within the pod, which is discarded when the pod is terminated. PersistentVolumeClaim, HostPath volumes, and Local Persistent Volumes are designed for persistent storage, unsuitable for stateless applications. ConfigMaps store application configuration data, not application data itself.

NEW QUESTION: 68
What is the name for a service that has no clusterIp address?
A. Headless
B. NodePort
C. ClusterIP
D. LoadBalancer
Answer: A (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/services-networking/service/#headless-services



NEW QUESTION: 69
Your application requires specific network configurations for its pods, including custom DNS settings and network namespaces. How can you achieve this in Kubernetes?
A. Use a NetworkPolicy to define custom network rules for the pods.
B. Configure a Pod's security context with specific network settings.
C. Use a DaemonSet to deploy a network configuration agent on each node.
D. Create a custom network plugin and integrate it with Kubernetes-

E. Modify the Kubernetes API server's network configuration settings.

Answer: D (LEAVE A REPLY)

The correct answer is 'Create a custom network plugin and integrate it with Kubernetes'. Kubernetes allows you to extend its networking functionality by developing and integrating custom network plugins. These plugins can provide advanced network configurations, including custom DNS settings, network namespaces, and other specific network requirements. Options A, B, C, and E are not suitable for this scenario. 'NetworkPolicy' is used for network access control, 'Pod security context' defines security settings for a pod, 'DaemonSet' is used for deploying agents on nodes, and modifying the API server's network settings can affect the entire cluster's network configuration.

NEW QUESTION: 70

You want to deploy a new microservice to your Kubernetes cluster using GitOps principles. Which of
the following approaches would you use to manage the deployment process?

A. Manually create and apply Kubernetes YAML files using 'kubectl apply'.

B. Use a CI/CD pipeline to build and deploy the microservice directly to the cluster.

C. Store the desired state of your microservice (deployment configuration, service definition, etc.) in a Git repository and use a GitOps tool like Flux or ArgoCD to manage deployments.

D. Use a Helm chart to package and manage the deployment of your microservice.

E. Use a Kubernetes Operator to automate the deployment and management of your microservice.

Answer: C (LEAVE A REPLY)

GitOps emphasizes the use of Git as the single source of truth for managing your cluster's desired state. Option C correctly describes the GitOps approach by storing the configuration in a Git repository and using a GitOps tool to manage deployments, ensuring consistency and traceability.

NEW QUESTION: 71

You're deploying a microservices application in a Kubernetes cluster. How can you use Jaeger for distributed tracing to troubleshoot a performance issue within the application?

A. Deploy Jaeger as a sidecar container within each pod.

B. Instrument your application code to emit tracing data.

C. Create a custom Prometheus exporter that integrates with Jaeger.

D. Configure a dedicated Jaeger deployment and configure your application to send tracing data to it.

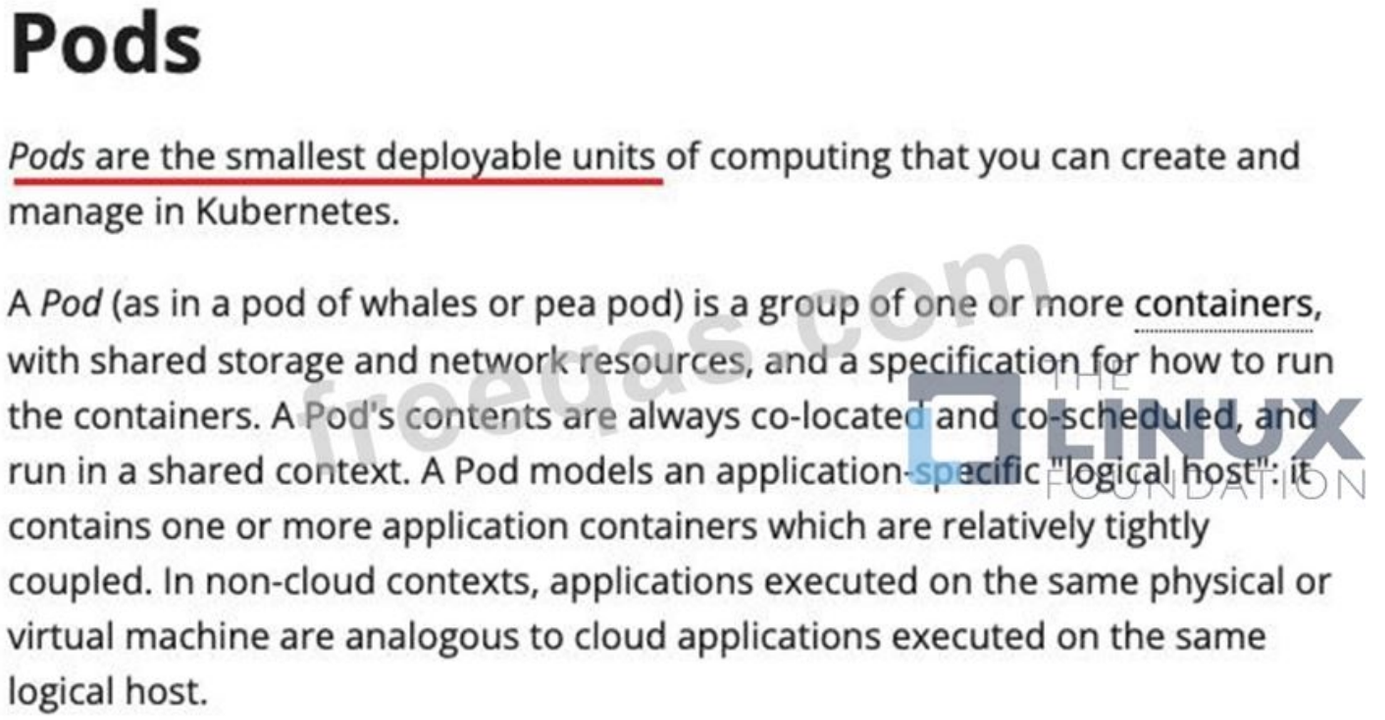
E. Use Jaeger's IJI to visualize the tracing data and analyze request flow across different services.

Answer: B,D,E (LEAVE A REPLY)

The correct answers are B, D, and E . To effectively use Jaeger for distributed tracing, you need to perform these steps: B: Instrument your application code to emit tracing data. Jaeger integrates

with application code to track requests across services. It's common to use libraries like OpenTelemetry or Jaeger's client libraries to inject tracing data into your application's code. D: Configure a dedicated Jaeger deployment and configure your application to send tracing data to it. Jaeger needs its own deployment to collect and process the tracing data. You need to set up a Jaeger instance within your Kubernetes cluster and configure your application to send its tracing data to this instance. E: Use Jaeger's UI to visualize the tracing data and analyze request flow across different services. Once the tracing data is collected, Jaeger provides a user interface to visualize the flow of requests across your microservices, identify bottlenecks, and analyze performance issues. The Jaeger UI allows you to drill down into individual traces, view the spans within a trace, and gain insights into latency, errors, and dependencies between services.

NEW QUESTION: 72
What is the smallest possible unit in Kubernetes to run a container?
A. pod
B. docker
C. service
D. container
Answer: A (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/workloads/pods/



NEW QUESTION: 73
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.

Answer: (SHOW ANSWER)

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

NEW QUESTION: 74

An application that is nearing its usage limit. To increase the amount of users it can handle, you allo-cate additional memory resources to each instance of the application. What type of scaling is this?

A. Horizontal Scaling

B. Cluster Autoscaling

C. Recursive Scaling

D. Vertical Scaling

Answer: D (LEAVE A REPLY)

NEW QUESTION: 75

You are building a CI/CD pipeline to deploy a Java application to Kubernetes. The application requires a specific Java version and some libraries to run correctly. How would you ensure that the correct runtime environment is available in the Kubernetes cluster?

A. Include the necessary Java version and libraries in the Dockerfile for the application image.

B. Pre-install the Java version and libraries on the Kubernetes nodes.

C. Use Kubernetes Init Containers to install the required dependencies before starting the main container.

D. Use Kubernetes ConfigMaps to define the environment variables for Java and libraries.

E. Use Kubernetes Secrets to store the Java version and library information securely.
Answer: (SHOW ANSWER)
The most common approaches are to include the required Java and libraries within the Dockerfile to create a self-contained image, or to use Kubernetes Init Containers. Init containers run before the main application container, installing the dependencies, ensuring the proper runtime environment is available.

NEW QUESTION: 76
What does the 'kops' acronym means?
A. Kubernetes Open Platform Specification
B. Kubernetes Operations
C. Kubernetes Operators
D. Kubernetes Operation Policy Specification
Answer: B (LEAVE A REPLY)
https://github.com/kubernetes/kops

NEW QUESTION: 77

You are deploying a new microservice on Kubernetes. The microservice needs access to a shared configuration file stored in a ConfigMap. How would you access the configuration file from within your microservice container?
A. Mount the ConfigMap as a volume into the container.
B. Use the *kubectl get' command to retrieve the configuration file from the ConfigMap.
C. Define an environment variable within the pod spec and map it to the ConfigMap data.
D. Use a Kubernetes Secret to store the configuration file securely.
E. Access the ConfigMap data directly using the Kubernetes API.
Answer: A,C (LEAVE A REPLY)
Both options A and C are valid ways to access configuration data from a ConfigMap within a container Option A (Mount as a volume): You can mount the ConfigMap as a volume into the container, allowing the microservice to access the configuration file directly from the mounted directory This is useful if the configuration file is large or has complex structure. Option C (Environment variable): You can define an environment variable in the pod spec and map it to the ConfigMap data This is simpler for smaller configuration values and can be easily accessed within the microservice code- The other options are not suitable for accessing ConfigMap data within a container.

NEW QUESTION: 78
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 (LEAVE A REPLY)
Worker Nodes are responsible for executing containerized workloads.

NEW QUESTION: 79
Your Kubernetes cluster is running a batch processing workload that only needs to run during specific time windows. How can you effectively manage the cost associated with this workload?
A. Use a CronJob to schedule the batch processing workload to run only during the required time window.
B. Configure a horizontal pod autoscaler (HPA) to scale the workload up during the time window and down afterward.
C. Create a separate namespace for the batch processing workload and apply resource quotas to limit its resource consumption.
D. Utilize spot instances or preemptible nodes for the batch processing workload, leveraging their discounted pricing.
E. Manually scale the workload up and down during the required time window.
Answer: A,D (LEAVE A REPLY)

For batch processing workloads with defined time windows, scheduling and leveraging cost-effective resources are key. CronJobs can be used to schedule the workload to run only during the required time window, eliminating unnecessary resource consumption. Utilizing spot instances or preemptible nodes provides discounted pricing for workloads that can tolerate interruptions, further reducing costs. While HPA can scale workloads, it's not the most efficient approach for time-bound batch processing. Creating a separate namespace and applying resource quotas can limit resource usage, but doesn't directly address the intermittent nature of the workload. Manually scaling up and down is inefficient and prone to errors.

NEW QUESTION: 80
What is the purpose of the "securityContext" field in a pod specification? Choose all that apply.
A. To define the user ID and group ID for the containers within the pod.
B. To specify the network namespace that the pod should be created in.
C. To configure the resource limits for the containers within the pod.
D. To define the security context of the pods, such as SELinux and AppArmor.
E. To control access to the Kubernetes API server.
Answer: A,D (LEAVE A REPLY)
The "securityContext" field in a pod specification is used to define the security context of the containers within the pod. This includes settings related to user ID and group ID, SELinux and AppArmor profiles, and other security-related options. It does not control access to the Kubernetes API server, configure resource limits, or specify the network namespace.

NEW QUESTION: 81
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
Answer: (SHOW ANSWER)
https://kubernetes.io/docs/reference/generated/kubectl/kubectl-commands#-em-deployment-em-

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

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

NEW QUESTION: 82
You have a containerized application that needs to access a specific environment variable. Which of the following methods would you typically use to provide this environment variable within a Kubernetes Pod definition?
A. Environment variable within the container image
B. ConfigMap

C. Secret
D. VolumeMounts
E. Pod Security Policy
Answer: B,C (LEAVE A REPLY)
Both ConfigMaps and Secrets are Kubernetes resources that allow you to pass configuration data to containers. ConfigMaps store simple key-value pairs, suitable for environment variables, while Secrets are used to store sensitive information like passwords or API keys. In this case, both are valid options to provide environment variables within a Pod.

NEW QUESTION: 83
Consider the following Prometheus query:
kube_pod_container_resource_requests_cpu_cores{pod="my-app-pod", container="my-app-container"} > 0.5 What does this query identify? Select all that apply.
A. Pods named "my-app-pod" with a container named "my-app-container" that are requesting more than 0.5 CPU cores.
B. Pods named "my-app-pod" with a container named "my-app-container" that are currently using more than 0.5 CPIJ cores.
C. Pods named "my-app-pod" with a container named "my-app-container" that have a CPIJ limit set to 0.5 cores or higher.
D. Pods named "my-app-pod" with a container named "my-app-container" that are experiencing high CPU utilization and may be resource-constrained.
E. Pods named "my-app-pod" with a container named "my-app-container" that are running on a Kubernetes node with a CPU capacity of 0.5 cores or higher.
Answer: A (LEAVE A REPLY)
The correct answer is A . The Prometheus query 'kube_pod_container_resource_requests_cpu_cores{pod="my-app-pod", container="my-app-container"} > 0.5' specifically targets the 'kube_pod_container_resource_requests_cpu_cores• metric, which represents the amount of CPU resources requested by a container in a pod. The query filters for pods with the name "my-app-pod" and containers with the name "my-app-container." The 0.5 condition indicates that the query only returns results where the requested CPU cores are greater than 0.5. The other options are incorrect: B : The query does not monitor the actual CPU usage. It focuses on the requested CPU resources, not the currently consumed CPIJ. C : This query does not relate to CPU limits. It focuses on the requested CPU resources, not the defined limits. D : While a pod exceeding its requested resources might indicate resource constraints, this query doesn't provide information about the actual CPU usage, only the requested resources. E : The query does not contain information about the node's CPU capacity. It focuses solely on the resources requested by a specific container within a pod.

NEW QUESTION: 84
You have a Deployment with the following YAML definition:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app-container
        image: my-app:latest
        ports:
        - containerPort: 8080
```


You want to update the image to 'my-app:v2' without deleting and recreating the Deployment. Which Kubernetes command can you use to achieve this update?

A. kubectl apply -f deployment.yaml

B. kubectl delete deployment my-app

C. kubectl rollout restart deployment my-app

D. kubectl set image deployment/my-app my-app-container=my-app:v2

E. kubectl scale deployment my-app -replicas=0

Answer: D (LEAVE A REPLY)

The 'kubectl set image' command is specifically designed to update the container image within a Deployment without needing to delete and recreate it. It allows you to target a specific container within the Deployment and specify the new image.

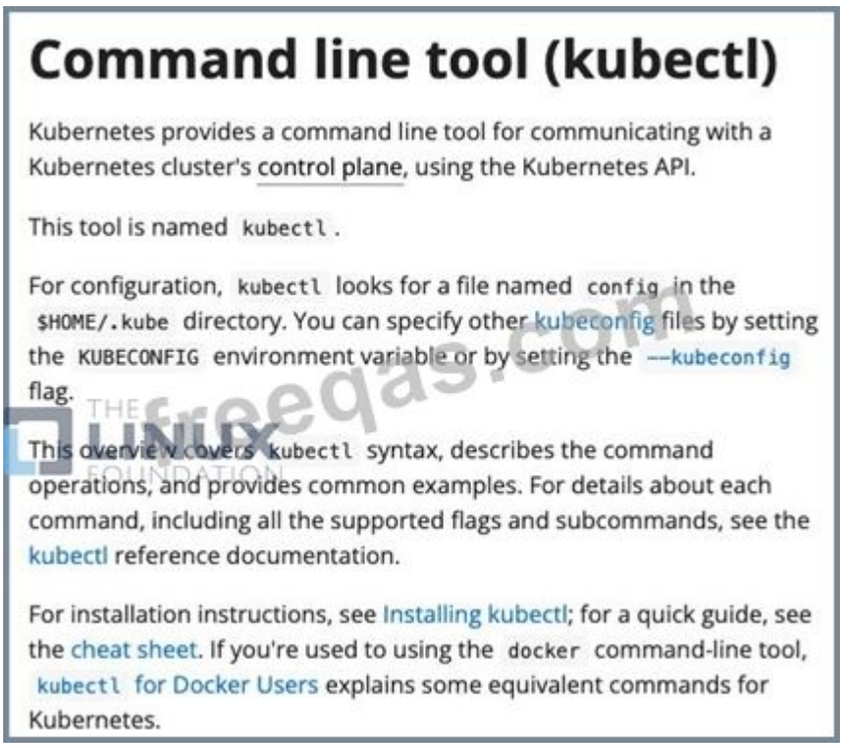
NEW QUESTION: 85

You are running a database application in a Kubernetes cluster. The database requires persistent storage that survives pod restarts. Which Kubernetes feature would you use to achieve this?

A. ConfigMap
B. Secret
C. PersistentVolumeClaim
D. Deployment
E. Service
Answer: (SHOW ANSWER)
A PersistentVolumeClaim (PVC) requests a volume from the cluster. This volume is then attached to a Pod and persists even if the Pod is deleted or restarted. This ensures that the database data is preserved even if a pod restarts or is recreated.

NEW QUESTION: 86
You are running a highly sensitive application in Kubernetes. Which of the following security measures is MOST effective in preventing unauthorized access to your application's secrets?
A. Using the '-privileged' flag for your application's container.
B. Deploying your application in a private Kubernetes cluster.
C. Configuring strong passwords for all Kubernetes users.
D. Employing a secrets management solution like HashiCorp Vault or AWS Secrets Manager.
E. Disabling Kubernetes RBAC and granting full access to all users.
Answer: D (LEAVE A REPLY)
Using a secrets management solution like HashiCorp Vault or AWS Secrets Manager is the most effective approach to securely storing and managing secrets in a Kubernetes environment. These solutions offer strong encryption, access control, and audit logging, providing comprehensive protection for your application's sensitive data.

NEW QUESTION: 87
Which command-line tool is used to interact with the Kubernetes cluster?
A. kube-api
B. kubectl
C. kube-scheduler
Answer: B (LEAVE A REPLY)
https://kubernetes.io/docs/reference/kubectl/



NEW QUESTION: 88
Which kubernetes object do deployments use behind the scenes when they need to scale pods?
A. POD
B. Deployment
C. Horizontal pod autoscaler
D. Api Scheduler
E. Replicasets
Answer: E (LEAVE A REPLY)
https://kubernetes.io/docs/concepts/workloads/controllers/replicaset/

NEW QUESTION: 89
Which style of operations are preferred for K8S and cloud native applications?
A. JSON
B. Declarative
C. Imperative
Answer: B (LEAVE A REPLY)
https://kubernetes.io/docs/tasks/manage-kubernetes-objects/declarative-config/#trade-offs

NEW QUESTION: 90
Explain the difference between a Docker image and a Docker container. Provide practical scenarios where they are used.
A. A Docker image is a blueprint for creating a container, while a container is a running instance of that image.
B. A Docker container is a blueprint for creating an image, while an image is a running instance of that container.
C. Docker images and containers are the same thing, just different names.
D. Docker images are used for storing data, while containers are used for running applications.
E. Docker images are used for running applications, while containers are used for storing data.
Answer: (SHOW ANSWER)
A Docker image is a static, immutable template that contains all the necessary components (files, libraries, dependencies, etc.) to run a specific application. Think of it as a blueprint. A Docker container is a running instance of that image, meaning it's an actual process running on your system. You can create multiple containers from the same image. Practical Scenarios: Image: You create a Docker image for a web server application, including the web server software, configuration files, and application code. This image can be shared with others or deployed to different environments. Container: You run multiple instances of this web server image as containers on your server. Each container gets its own isolated environment, allowing you to scale your web application easily.

NEW QUESTION: 91
You are running a stateless application in a Kubernetes cluster. The application has multiple instances deployed as Pods. You need to ensure that all incoming traffic to the application is distributed evenly across these Pods. Which Kubernetes resource should you use to achieve this?
A. Deployment
B. Service
C. Ingress
D. Namespace
E. ConfigMap
Answer: (SHOW ANSWER)
A Service is a Kubernetes resource that provides a stable endpoint for a group of Pods. It acts as a load balancer, distributing traffic across the Pods in a round-robin manner by default. This ensures that all Pods receive an equal share of the incoming traffic.

NEW QUESTION: 92
You are deploying a stateful application with persistent storage using PersistentVolumeClaims (PVCs). What are the possible ways to ensure that the PVCs are bound to the correct PersistentVolumes (PVs)?
A. Use the 'storageClassName' field in the PVC to specify a storage class that matches the PV
B. Use the 'accessModes' field in the PVC to specify the access mode (ReadWriteOnce, ReadOnlyMany, ReadWriteMany) that matches the PV
C. Manually bind the PVC to a specific PV by specifying the PV name in the PVC's 'specvolumeName' field-
D. Use the 'capacity' field in the PVC to specify the storage capacity that matches the PV
E. None of the above
Answer: A,B,C (LEAVE A REPLY)
You can ensure PVCs are bound correctly by specifying the 'storageClassName' field to match the PV's storage class, using 'accessModes' to match access types, or by manually binding the PVC to a specific PV The 'capacity' field can be used to specify the storage capacity needed, but doesn't directly control the binding process.

NEW QUESTION: 93
Consider the following Kubernetes pod YAML definition:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
spec:
  containers:
  - name: nginx
    image: nginx:1.14.2
    resources:
      requests:
        cpu: 100m
        memory: 200Mi
      limits:
        cpu: 200m
        memory: 400Mi
  nodeSelector:
    kubernetes.io/hostname: node1
  tolerations:
  - key: "key1"
    operator: "Equal"
    value: "value1"
    effect: "NoSchedule"
```

Which of the following statements is TRUE about this pod's scheduling behavior?

A. The pod will only be scheduled on a node with the label 'kubernetes.io/hostname: nodel , regardless of available resources.
B. The pod will only be scheduled on a node with the label 'keyl: valuel s, regardless of available resources.
C. The pod will only be scheduled on a node that has at least 100m CPU and 200Mi memory available.
D. The pod will not be scheduled on any node that has the taint *keyl: valuel: NoSchedule• , as long as the •nodeselector• condition is met.
E. The pod will be scheduled on any node that meets the •nodeSelector• and •tolerations* conditions, but it may be evicted if resources become scarce.
Answer: D (LEAVE A REPLY)
The 'nodeselector• field instructs Kubernetes to schedule the pod ONLY on a node labeled with 'kubernetes.io/hostname: nodel*. However, the •tolerations' field specifies that the pod can tolerate a taint with the key 'keyl • , the value 'valuel • , and the effect 'NoSchedule'. This means that the pod will NOT be scheduled on a node that has that taint applied, even if the node meets the •nodeSelector• condition. The •requests' and 'limits' fields specify resource requirements for the pod, but they are not the primary factor determining the pod's scheduling in this case. The •tolerations* field takes precedence due to its 'NoSchedule• effect.

NEW QUESTION: 94
Your application requires a specific storage class for its persistent dat
a. How do you configure this storage class within your deployment YAML?
A. Specify the storage class name directly within the 'spec.template.spec.containers[0].volumeMounts[0].name' field of the deployment.
B. Create a separate PersistentVolumeClaim (PVC) with the desired storage class and reference the PVC in the deployment's name' field.
C. Specify the storage class name within the field of the deployment.
D. Specify the storage class name within the 'spec.template.spec.containers[01.volumeMounts[0].storageClassName' field of the deployment.
E. None of the above
Answer: B (LEAVE A REPLY)
The correct approach is to create a separate PersistentVolumeClaim (PVC) that specifies the desired storage class and reference the PVC in the deployments 'spec_template.spec_containers[0]_volumeMounts[0]_name' field. This ensures the PVC is automatically bound to a PV with the correct storage class. Specifying the storage class name directly within the deployment or the volumeMounts section is not the standard practice for defining storage requirements.

NEW QUESTION: 95

You are implementing a continuous integration and continuous deployment (CI/CD) pipeline for a cloud native application running in Kubernetes. Which of the following tools can be used for building, testing, and deploying the application to the cluster?

A. Jenkins

B. CircleCI

C. Travis CI

D. GitHub Actions

E. GitLab CI/CD

Answer: A,B,C,D,E (LEAVE A REPLY)

All of the listed tools are widely used for CI/CD pipelines and can be integrated with Kubernetes. Each tool has its own strengths and weaknesses depending on your specific needs. Here's a brief overview: Jenkins: Open-source, highly customizable, and supports a wide range of plugins. CircleCI: Cloud-based, user-friendly, and well-suited for smaller teams and projects. Travis CI: Also cloud-based, popular for open-source projects and known for its simple setup. GitHub Actions: Native to GitHub, allows you to build, test, and deploy directly within your repository. GitLab CI/CD: Integrated with GitLab, provides a complete CI/CD solution with features for building, testing, and deploying applications.

NEW QUESTION: 96

You are running a Kubernetes cluster with three nodes. One node experiences a hardware failure. Which of the following Kubernetes features ensures that your application remains available and running?

A. Horizontal Pod Autoscaling

B. Node affinity

C. Pod Disruption Budgets

D. Self-healing

E. Rolling updates

Answer: (SHOW ANSWER)

Kubernetes' self-healing capabilities ensure that your application remains available even when a node fails. It automatically detects and restarts failed Pods on healthy nodes, maintaining the desired number of replicas.

NEW QUESTION: 97

You are running a web application in Kubernetes. The application uses a database that is accessed by multiple Pods. You want to ensure that the database connection details are securely stored and accessed by the Pods. Which Kubernetes feature would you use for this purpose?

A. ConfigMap

B. Secret

C. PersistentVolumeClaim

D. Service

E. Namespace

Answer: B (LEAVE A REPLY)
Secrets are Kubernetes resources that allow you to store sensitive information, such as passwords, API keys, and certificates. You can then mount Secrets into Pods, allowing them to securely access the information they need without exposing it in plain text.

NEW QUESTION: 98
You are running a web application with a high demand for CPU resources. Which Kubernetes scheduling strategy could help you ensure pods are scheduled on nodes with the most available CPU capacity?
A. Node affinity
B. Pod affinity
C. Node anti-affinity
D. Pod anti-affinity
E. Taints and tolerations
Answer: A (LEAVE A REPLY)
Node affinity allows you to define preferences for where pods should be scheduled based on node labels. You can use node affinity to prioritize scheduling on nodes with high CPU capacity. While the other options can influence scheduling, they are not directly focused on CPU availability.

NEW QUESTION: 99
What are the key differences between a Kubernetes Pod and a Docker container? Provide examples of how they are used in a Kubernetes deployment.
A. A Pod is a single running container, while a Docker container can run multiple applications.
B. A Pod is a logical grouping of one or more containers, while a Docker container is a single running instance of an image.
C. A Docker container is managed by Kubernetes, while a Pod is managed by Docker.
D. A Pod is a physical machine, while a Docker container is a virtual machine.
E. A Pod is a cloud-native application, while a Docker container is a traditional application.
Answer: B (LEAVE A REPLY)
A Pod in Kubernetes is the smallest deployable unit. It represents a group of one or more containers that share resources and networking. Docker containers, on the other hand, are individual running instances of Docker images. Examples: Pod: You could have a Pod that runs a web server container, a database container, and a logging container, all working together as a single unit. Kubernetes manages the scheduling, networking, and resource allocation for the entire Pod. Docker Container: You could run a single Docker container for a web server application on a single machine.

NEW QUESTION: 100

You have a Kubernetes cluster running on AWS. You need to ensure that only approved container images are used in your cluster. Which Kubernetes feature can you use to enforce this policy?

A. Network Policies

B. Pod Security Policies

C. Admission Controllers

D. Resource Quotas

E. Service Accounts

Answer: (SHOW ANSWER)

Admission Controllers in Kubernetes can be used to enforce policies for container images. You can configure an Admission Controller to check if the image is present in an approved image registry or if it meets certain security criteria. This helps prevent unauthorized or insecure images from being deployed to your cluster.

NEW QUESTION: 101

You have a Kubernetes cluster running on AWS. You want to configure a persistent volume claim (PVC) that uses an AWS EBS volume for storage. Which annotation can be used to specify the EBS volume type?

A. volume.beta.kubernetes.io/storage-class

B. volume.beta.kubernetes.io/storage-provisioner

C. volume.beta.kubernetes.io/aws-ebs-volume-type

D. volume.beta.kubernetes.io/aws-ebs-volume-size

E. volume.beta.kubernetes.io/aws-ebs-volume-encrypted

Answer: C (LEAVE A REPLY)

The annotation •volume.beta.kubernetes.io/aws-ebs-volume-type• is used to specify the EBS volume type (e.g., 'gp2 , 701', 'standard') when using an AWS EBS volume for persistent storage. Option 'A' is used to specify the storage class for the PVC. Option 'B' specifies the storage provisioner, which is responsible for creating the volume. Option 'D' is used to specify the size of the EBS volume. Option 'E' is for specifying whether the EBS volume should be encrypted.

NEW QUESTION: 102

You're running a Kubernetes cluster with several applications, and you want to analyze the performance of your applications at the Kubernetes cluster level. What are the key metrics you should monitor to gain insights into cluster-wide performance?

A. Number of pods running in each namespace.

B. Total CPU and memory usage across all nodes in the cluster.

C. Number of failed deployments and pod restarts.

D. Latency and throughput of requests to the Kubernetes API server.

E. Number of active Kubernetes controllers (e.g., Deployment, ReplicaSet).

Answer: (SHOW ANSWER)

The correct answers are B, C, D, and E . These metrics are crucial for understanding the overall performance of your Kubernetes cluster. B: Total CPU and memory usage across all nodes in the cluster. This metric helps you assess the overall resource consumption of your cluster. If the CPU or memory utilization consistently reaches high levels, it might indicate resource constraints, performance bottlenecks, or potential capacity planning issues. C: Number of failed deployments and pod restarts. This metric provides insights into the stability and reliability of your applications. Frequent deployment failures or pod restarts might indicate issues with application code, deployment configurations, or underlying infrastructure. D: Latency and throughput of requests to the Kubernetes API server. The Kubernetes API server handles all requests to manage the cluster. Monitoring API server latency and throughput can help identify performance bottlenecks and potential issues with cluster communication. E: Number of active Kubernetes controllers (e.g., Deployment, ReplicaSet). Kubernetes controllers are responsible for managing and maintaining the state of your applications. Tracking the number of active controllers can help you identify potential issues with controller activity, such as resource exhaustion or unexpected controller behavior. Option A is not as crucial for cluster-wide performance analysis. While knowing the number of pods running in each namespace might be useful for resource allocation, it doesn't directly reflect the overall performance of the cluster.

NEW QUESTION: 103
You are running a containerized application on Kubernetes. You want to monitor the performance and health of your application and its containers effectively. Which of these open standards would you leverage?
A. Kubernetes Resource Quotas
B. open Policy Agent (OPA)
C. OpenTelemetry
D. Kubernetes Ingress
E. Service Mesh
Answer: (SHOW ANSWER)
OpenTelemetry is a standard for collecting, processing, and exporting telemetry data (metrics, logs, and traces). It enables you to gain insights into the performance and health of your application and its containers within a Kubernetes environment, providing valuable information for troubleshooting and optimization.