Exam Session - Cert Prep: Certified Kubernetes Application Developer (CKAD)

cloudacademy.com/quiz/exam/3761622/results

#1

What would be the recommended first step when debugging a pod?

✗
Make sure insufficient available resources are not preventing scheduling the pod.

✓
Use the " `kubectl describe pods` " and " `kubectl logs` " commands to check the current state of the pod and recent events.

✗
Use the " `kubectl get nodes -o` " command to see if any pod reports an error status.

✗
Verify if there was a failure to pull the image.

Explanation

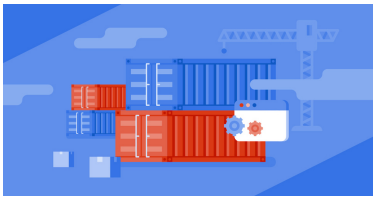
The first step in debugging a pod is taking a look at it. Check the current state of the pod and recent events with the following command:

```
$ kubectl describe pods ${POD_NAME}
```

Look at the state of the containers in the pod. Are they all Running? Have there been recent restarts?

Continue debugging depending on the state of the pods.

https://kubernetes.io/docs/user-guide/debugging-pods-and-replication-controllers/
Covered in this lecture
Services
Course:Introduction to Kubernetes



7m

#2

In Kubernetes, a(n) _____ is a key/value pair, intended to be used to specify identifying attributes of objects that are meaningful and relevant to users, but which do not directly imply semantics to the core system.

✗
pod

✗
annotation

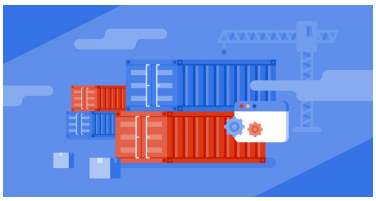
✓
label

✗
selector

Explanation

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 which 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.

https://kubernetes.io/docs/user-guide/labels/
Covered in this lecture
Pods
Course:Introduction to Kubernetes



15m

#3

You have deployed an application in Kubernetes. The application container exposes port 80. You need to be able to access the application from outside of the cluster. What Kubernetes resource should you use to meet this requirement?

✓
A service

✗

A binding

✕
A deployment

✕
An endpoint

Explanation

A service provides a mechanism for accessing a logical set of pods. You can use a service of type NodePort or LoadBalancer to allow external access to an application running in Kubernetes.

A binding is used for associating a role with a user to authorize actions the user is allowed to perform.

A deployment can deploy an application in the cluster, but it can't grant external access without a service.

An endpoint is a resource that a service automatically manages to keep track of the pods that are accessible via the service.

/lab/deploy-a-stateless-application-in-a-kubernetes-cluster/deploying-a-stateless-application-in-the-kubernetes-cluster/
Covered in this lecture
Deploying a Microservices Application into EKS
Course:Introduction to AWS EKS

22m

#4

How would you segment resources in Kubernetes when you want to prevent users from easily affecting resources in other projects, teams, or environments?

✓
Create a Namespace for each segment

✕
Use resource naming conventions

✕

Use a rigorous label schema

✕
Use Kubernetes Scopes

Explanation

You can use labels to distinguish resources within the same Namespace. However, to avoid unintentionally modifying other teams' resources, you are better off using Namespaces. Namespaces provide a scope for names. Names of resources need to be unique within a Namespace, but not across Namespaces. This reduces the chance of unitentionally affecting other teams' resources.

https://kubernetes.io/docs/concepts/overview/working-with-objects/namespaces/
Covered in this lecture
Scheduling Pods
Course:Administering Kubernetes Clusters

20m

#5

You've been asked to debug a pod that is crashing. What command would be particularly helpful during this process?

✕
kubectl create -f

✕
kubectl get nodes -o

✓
kubectl logs

✕
docker pull

Explanation

What to do when a pod is crashing or otherwise unhealthy?

First, take a look at the logs of the current container:

```
$ kubectl logs ${POD_NAME} ${CONTAINER_NAME}
```

If your container has previously crashed, you can access the previous container's crash log with:

```
$ kubectl logs --previous ${POD_NAME} ${CONTAINER_NAME}
```

Alternately, you can run commands inside that container with exec:

```
$ kubectl exec ${POD_NAME} -c ${CONTAINER_NAME} -- ${CMD} ${ARG1} ${ARG2} ... ${ARGN}
```

https://kubernetes.io/docs/user-guide/debugging-pods-and-replication-controllers/

Covered in this lecture

Networking

Course:Kubernetes Patterns for Application Developers

11m

#6

You need to utilize a PersistentVolume for application storage in a Kubernetes cluster. What field of a PersistentVolume can you use to control the number of nodes that can mount the PersistentVolume for reading and writing?

✓ accessMode

✗ mountOptions

✗ sharingOptions

✗ nodeSelector

Explanation

A PersistentVolumes accessMode field controls how many nodes can mount it for reading and writing. The supported values are ReadWriteOnce, ReadOnlyMany, and ReadWriteMany.

/lab/deploy-a-stateful-application-in-a-kubernetes-cluster/deploying-stateful-application-kubernetes-cluster/

#7

Your DevOps manager has asked you to manually create a service account named jenkins. Which command could you use?

✗
kubectl config set-credentials -name jenkins -create

✗
kubelet config set-credentials -name jenkins -create -i yaml

✓
kubectl create serviceaccount jenkins

✗
kubelet create serviceaccount jenkins -i yaml

Explanation

Service account bearer tokens are perfectly valid to use outside the cluster and can be used to create identities for long standing jobs that wish to talk to the Kubernetes API. To manually create a service account, simply use the kubectl create serviceaccount (NAME)command. This creates a service account in the current namespace and an associated secret. For example:

```
$ kubectl create serviceaccount jenkins
```

https://kubernetes.io/docs/admin/authentication/
#8

What happens when you apply both container and pod level security context settings that overlap?

✗
The assignment that causes the overlap fails

✓
The container level security context settings override settings made at the pod level

✗
The pod level security context settings override settings made at the container level

✗
The settings applied first are overridden by the ones applied last

Explanation

Container level security context settings are applied to the specific container and override settings made at the pod level where there is overlap. Container level settings however do not affect the pod's volumes.

https://kubernetes.io/docs/concepts/policy/security-context/
#9

You are modernizing a suite of legacy applications that work together and deploying the applications as Pods in Kubernetes as part of the process. Currently the applications use several different monitoring tools and you would like to use a single monitoring tool for all the applications. You will be able to modify the code of some pieces but not others. Which multi-container Pod design pattern can you use to accomplish the goal?

✗
Leader Election

✓
Adapter

✗
Sidecar

✗
Ambassador

Explanation

The adapter pattern is ideally suited for this scenario. Each application can be left unmodified and the adapter can present a consistent interface to the monitoring tool. The adapter container internalizes the complexity of converting between the format of metrics each legacy application uses and single consistent interface presented to the selected monitoring tool.

https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/45406.pdf
Covered in this lecture

Multi-container Patterns
Course:Kubernetes Patterns for Application Developers

8m

#10

Your DevOps manager wants to use a monitoring tool that supports Kubernetes natively. Which tool would you suggest?

✗ Ganglia

✗ Nagios

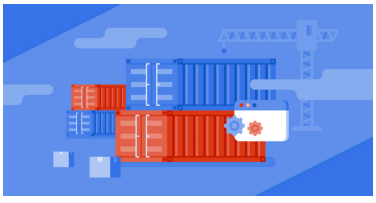
✗ Splunk

✓ Metrics Server/Heapster

Explanation

Metrics Server, and its predecessor Heapster, are cluster-wide aggregator of monitoring and event data. It currently supports Kubernetes natively and works on all Kubernetes setups. Heapster and Metrics Server run as a pod in the cluster, similar to how any Kubernetes application would run. The Heapster or Metrics Server pod discovers all nodes in the cluster and queries usage information from the nodes' Kubelet, the on-machine Kubernetes agent. The Kubelet itself fetches the data from cAdvisor. Heapster or Metrics Server groups the information by pod along with the relevant labels. This data is then pushed to a configurable backend for storage and visualization.

https://kubernetes.io/docs/user-guide/monitoring/
Covered in this lecture
Kubernetes Ecosystem
Course:Introduction to Kubernetes



8m

#11

What is the core grouping primitive in Kubernetes?

✗
Key-value pair

✗
Namespace

✗
Annotation

✓
Label selector

Explanation

Unlike names and UIDs, labels do not provide uniqueness, so in general, we expect many objects to carry the same label(s). The label selector is the core grouping primitive in Kubernetes, and allows the the client/user to identify a set of objects.

https://kubernetes.io/docs/user-guide/labels/

#12

Every namespace has a default service account resource called _____.

✗
root

✗
admin

✗
sysadmin

✓
default

Explanation

Every namespace has a default service account resource called default. You can list this and any other serviceAccount resources in the namespace with this command:

$ kubectl get serviceAccounts

NAME SECRETS

default 1

https://kubernetes.io/docs/user-guide/service-accounts/
Covered in this lecture
Service Accounts
Course:Kubernetes Patterns for Application Developers

5m

#13

Service accounts are tied to a set of credentials stored as _____, which are mounted into pods allowing in cluster processes to talk to the Kubernetes API.

✕
Crumbs

✕
Nodes

✓
Secrets

✕
Tokens

Explanation

Service accounts are users managed by the Kubernetes API and are bound to specific namespaces. The accounts are created automatically by the API server or manually through API calls, and tied to a set of credentials stored as Secrets. These Secrets are then mounted into pods allowing in cluster processes to talk to the Kubernetes API.

https://kubernetes.io/docs/admin/authentication/
Covered in this lecture
Service Accounts
Course:Kubernetes Patterns for Application Developers

5m

#14

Which levels of security context are supported by Kubernetes?

✗ Resource level security context only

✗ Pod level security context only

✓ Pod level and Container level security

✗ Namespace level and Resource level security context

Explanation

A security context defines the operating system security settings (uid, gid, capabilities, SELinux role, etc..) applied to a container. There are two levels of security context: pod level security context, and container level security context.

https://kubernetes.io/docs/concepts/policy/security-context/

Covered in this lecture

Multi-container Patterns

Course:Kubernetes Patterns for Application Developers

8m

#15

You have written a manifest file for a service in Kubernetes. You did not include the type field in the service's specification. What is the type of the service that will be created?

✓ ClusterIP

✗ LoadBalancer

✗
None of these options -- the service must declare a type

✗
NodePort

Explanation

If you don't specify a type for a service, it will use the default value of ClusterIp. ClusterIp services are accessible only within the cluster. Other types of services can be used to access a service from outside the cluster (NodePort and LoadBalancer) or access services outside the cluster (ExternalName).

https://kubernetes.io/docs/tasks/access-application-cluster/communicate-containers-same-pod-shared-volume/
#16

What is one way through which users can access the Kubernetes API?

✓
kubectl

✗
Java API

✗
ssh

✗
gRPC requests

Explanation

Users access the API using kubectl, client libraries, or by making REST requests. Both human users and Kubernetes service accounts can be authorized for API access.

https://kubernetes.io/docs/admin/accessing-the-api/
#17

Which of the following statements related to Kubernetes storage are true? (Select all that apply)

✓
A Volume's lifetime is connected to the lifetime of a pod

✗
A PersistentVolume's lifetime is connected to the lifetime of a pod

✗
Volumes can be shared by multiple containers in a pod, while PersistentVolumes cannot be shared by containers

✓
PersistentVolumes must be explicitly claimed by pods

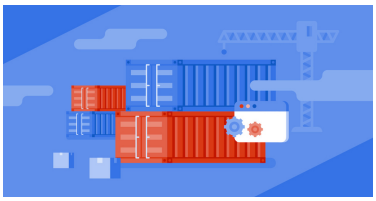
Explanation

A Volume's lifetime is the same as the lifetime of the pod that encloses it. PersistentVolumes have a lifetime independent of any pod allowing the data on a PersistentVolume to be reused by other pods.

PersistentVolumes must be claimed by pods using PersistentVolumeClaims. Volumes do not require the use of claims. Simply including a volume in a Pod spec is enough to create and access the volume in the pod.

Both Volumes and PersistentVolumes can be accessed by all of the containers in the pod enclosing them.

/course/introduction-to-kubernetes/volumes/
Covered in this lecture
Volumes
Course:Introduction to Kubernetes



13m
#18

An intern in your team has asked for your help in writing a set-based label selector that filters resources with a partition key (no matter the value) and with environment different than qa. Write a set-based label selector that accomplishes this task.

✗
exists(partition) && environment isnot (qa)

✗
partition == *, environment notin (qa)

✓
partition, environment notin (qa)

✗
partition, environment != qa

Explanation

Set-based label requirements allow filtering keys according to a set of values. Three kinds of operators are supported: in,notin and exists (only the key identifier). The comma separator acts as an AND operator. So filtering resources with a partition key (no matter the value) and with environment different than qa can be achieved using partition,environment notin (qa). The set-based label selector is a general form of equality since environment=production is equivalent to environment in (production); similarly for != and notin

https://kubernetes.io/docs/user-guide/labels/

#19

A colleague has asked for your recommendation on how/where to record or store client library information that is useful for debugging purposes, like name, version, and build info. What would you recommend?

✗
Store it in a namespace.

✗
Store it the pod logs.

✓
Assign adequate annotations.

✗
Define adequate labels.

Explanation

Annotations are useful to be able to attach arbitrary non-identifying metadata, for retrieval by API clients such as tools, libraries, etc. This information may be large, may be structured or unstructured, may include characters not permitted by labels, etc. Such information would

not be used for object selection and therefore doesn't belong in labels.

Examples of such information include:

- fields managed by a declarative configuration layer, to distinguish them from client- and/or server-set default values and other auto-generated fields, fields set by auto-sizing/auto-scaling systems, etc., in order to facilitate merging
- build/release/image information (timestamps, release ids, git branch, PR numbers, image hashes, registry address, etc.)
- pointers to logging/monitoring/analytics/audit repos
- client library/tool information (e.g. for debugging purposes – name, version, build info)
- other user and/or tool/system provenance info, such as URLs of related objects from other ecosystem components
- lightweight rollout tool metadata (config and/or checkpoints)
- phone/pager number(s) of person(s) responsible, or directory entry where that info could be found, such as a team website

https://kubernetes.io/docs/user-guide/annotations/

#20

A Kubernetes _____ is an abstraction that defines a logical set of Pods and a policy by which to access them.


Job


Controller


Service


Selector

Explanation

A Kubernetes Service is an abstraction that defines a logical set of Pods and a policy by which to access them - sometimes called a micro-service. The set of Pods targeted by a Service is (usually) determined by a Label Selector (see below for why you might want a Service without a selector).

https://kubernetes.io/docs/concepts/services-networking/service/#headless-services
Covered in this lecture
Networking
Course:Administering Kubernetes Clusters

7m

#21

What is one scenario in which the use of Kubernetes pods would be recommended?

✗
To host vertically integrated stateful applications

✓
To support co-located, co-managed helper programs

✗
To run multiple instances of the same application

✗
To provide independent storage volumes across platforms

Explanation

Pods can be used to host vertically integrated application stacks (e.g., LAMP), but their primary motivation is to support co-located, co-managed helper programs, such as:

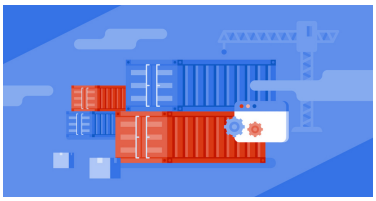
- content management systems, file and data loaders, local cache managers, etc.
- log and checkpoint backup, compression, rotation, snapshotting, etc.
- data change watchers, log tailers, logging and monitoring adapters, event publishers, etc.
- proxies, bridges, and adapters
- controllers, managers, configurators, and updaters

Individual pods are not intended to run multiple instances of the same application, in general.

https://kubernetes.io/docs/user-guide/pods/
Covered in this lecture
Deployments
Course:Introduction to Kubernetes

11m

#22



Which two kubectl commands are useful for collecting information about any type of resource that is active in a Kubernetes cluster? (Choose 2 answers)

✓ get

✓ describe

✗ logs

✗ expose

Explanation

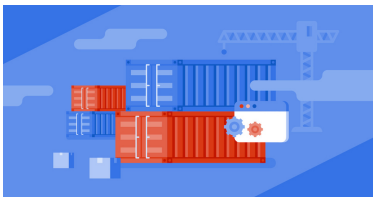
The get and describe commands are useful for reporting information about active resources in a cluster.

list and watch are not kubectl commands. They are two examples of read verbs in the Kubernetes API. For example, kubectl sends list requests to the Kubernetes API to create the output of kubectl get commands.

The explain, expose, and logs commands are kubectl commands but are not useful for gathering information about Kubernetes resources. The explain command provides information for understanding the fields of resources, but doesn't get information about resources running in the cluster. The expose command creates a service resource. The logs command get container logs. The logs may be useful for understanding the status of a Pod resources, but logs is not useful for all types of resources.

/lab/deploy-a-stateless-application-in-a-kubernetes-cluster/deploying-a-stateless-application-in-the-kubernetes-cluster/
Covered in this lecture
Multi-Container Pods
Course:Introduction to Kubernetes

10m



#23

You would like to start sending traffic to a pod only when a probe succeeds. What is the best type of probe in this situation?

✕ Use a LivenessProbe.

✕ Specify a LivenessProbe and a RestartPolicy of Always or OnFailure.

✓ Specify a ReadinessProbe.

✕ Use a ReadinessProbe and a RestartPolicy of Always or OnFailure.

Explanation

If the process in your container is able to crash on its own whenever it encounters an issue or becomes unhealthy, you do not necessarily need a liveness probe; the kubelet will automatically perform the correct action in accordance with the RestartPolicy when the process crashes.

If you'd like your container to be killed and restarted if a probe fails, then specify a LivenessProbe and a RestartPolicy of Always or OnFailure.

If you'd like to start sending traffic to a pod only when a probe succeeds, specify a ReadinessProbe. In this case, the ReadinessProbe may be the same as the LivenessProbe, but the existence of the ReadinessProbe in the spec means that the pod will start without receiving any traffic and only start receiving traffic once the probe starts succeeding.

If a container wants the ability to take itself down for maintenance, you can specify a ReadinessProbe that checks an endpoint specific to readiness, which is different than the LivenessProbe.

https://kubernetes.io/docs/user-guide/pod-states/

#24

_____ within a pod share an IP address and port space, and can find each other via localhost.

✓
Containers

✗
Instances

✗
Images

✗
Contexts

Explanation

Containers within a pod share an IP address and port space, and can find each other via the localhost. They can also communicate with each other using standard inter-process communications like SystemV semaphores or POSIX shared memory. Containers in different pods have distinct IP addresses and cannot communicate by IPC.

https://kubernetes.io/docs/user-guide/pods/
Covered in this lecture
Summary
Course:Kubernetes Patterns for Application Developers

3m

#25

When using Kubernetes, why is it recommended that you use labels?

✗
To attach arbitrary non-identifying metadata, for retrieval by API clients such as tools, libraries, etc.

✗
To use in environments where many users are spread across multiple teams, or projects.

✓
To enable users to map their own organizational structures onto system objects in a loosely coupled fashion, without requiring clients to store these mappings.

To model an application-specific “logical host”.

Explanation

Labels enable users to map their own organizational structures onto system objects in a loosely coupled fashion, without requiring clients to store these mappings.

Service deployments and batch processing pipelines are often multi-dimensional entities (e.g., multiple partitions or deployments, multiple release tracks, multiple tiers, multiple micro-services per tier). Management often requires cross-cutting operations, which breaks encapsulation of strictly hierarchical representations, especially rigid hierarchies determined by the infrastructure rather than by users.

https://kubernetes.io/docs/user-guide/labels/