

Exam Session - Knowledge Check: Kubernetes Concepts

 cloudacademy.com/quiz/exam/3761534/results

#1

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



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



Volumes must be explicitly claimed by pods



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](#)

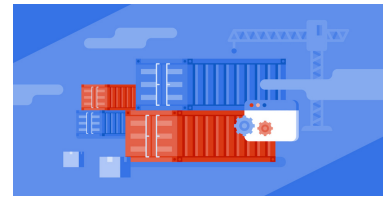
[Course:Introduction to Kubernetes](#)

13m



#2

Which fields are required when writing a Kubernetes YAML manifest file for creating any type of Kubernetes resource? (Select all that apply)



apiVersion



kind



metadata



status

Explanation

Every resource manifest must specify the version of the API (apiVersion), the kind of resource, and metadata that usually includes a name.

The status field is added to a resource after it has been created. Similarly, the metadata.uid field is added after a resource is created. Neither need to be specified when creating a resource from a manifest file.

 <https://kubernetes.io/docs/concepts/workloads/pods/>

Covered in this lecture

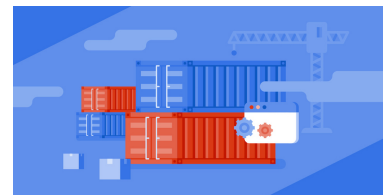
Deployments

Course: Introduction to Kubernetes

11m



#3



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



list



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

#4



A colleague of yours told you they have cordoned a node. When you check the status of the node, which of the following values do you see?



SchedulingDisabled



Cordoned




Drained



Pending

Explanation

When a node is cordoned, it becomes ineligible to schedule new pods to the node. The status of SchedulingDisabled is reported to indicate that a node has been cordoned.

 </lab/deploy-a-stateless-application-in-a-kubernetes-cluster/cleaning-up-your-kubernetes-cluster/>

#5

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




ExternalName



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/>

#6

What property of a Kubernetes service defines the set of pods that are accessed via the service?



The service's selector



The service's label



The service's endpoints list



The service's pods list

Explanation

The service's selector defines the pods that are accessed via the service. Any pod that has a label matching the service's selector will be accessed through the service.

The service's label doesn't impact the pods accessed via the service.

A service doesn't have an endpoints property. Endpoints are separate Kubernetes resources that are automatically updated based on the service's selector.

Services don't have a pods list.

 <https://kubernetes.io/docs/concepts/workloads/pods/>

Covered in this lecture

Deployments

Course: Introduction to Kubernetes

11m

#7



You have an application that uses a persistent volume in Kubernetes. If an application pod terminates, you would like to have the persistent volume remain but have its data deleted. How can you achieve this with the least amount of effort?



This is supported natively because PersistentVolumes can be scrubbed after releasing from a pod.

✗

This cannot be achieved.

✗

You can deploy a sidecar container in the pod that will delete the volume's contents when the main application container signals it is about to terminate.

✗

You can deploy an application that watches pod events to delete the contents of the persistent volume when the target pod is terminated.

Explanation

PersistentVolumes can be scrubbed to delete the data stored on the volume without deleting the volume entirely. This is accomplished by using a reclaim policy of recycle. When possible dynamic provisioning of persistent volumes is preferred over the recycle reclaim policy.

 </course/introduction-to-kubernetes/volumes/>

Covered in this lecture

Volumes

Course:Introduction to Kubernetes

13m

#8



You have a container that requires some custom code to run before the container is ready to start. In Kubernetes, what Pod field should you use to ensure that the code runs before the main application container starts?

✗

containers

✓

initContainers

✗

livenessProbe

✗

readinessProbe

Explanation

Init containers are the recommended way to support the describe use case. Init containers are like regular containers, but they must run to completion before any normal containers start. An example would be to clone a git repository to a volume to copy source code that the main application container depends on before it can be started. The `initContainer` field of a Pod spec is where you declare init containers.

The `containers` field doesn't provide an easy way to ensure that one container completes before another.

Liveness probes are used to monitor if a container should be restarted.

Readiness probes are used to test when a container is ready to be added to a Kubernetes service to start accepting traffic.

 <https://kubernetes.io/docs/concepts/workloads/pods/init-containers/>

Covered in this lecture

Init Containers

Course: Introduction to Kubernetes

5m

#9



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/>

#10

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

22m



#11



What are some reasons you would prefer to use Deployments rather than "naked" Pods (Pods that are not managed by a higher-level resource, such as a Deployment) for managing applications in Kubernetes? (Select all that apply)



Deployments are compatible with Horizontal Pod Autoscalers



Deployments support running multiple containers



Deployments can reschedule pods that fail




Deployments support rolling updates and rollbacks

Explanation

Deployments can be autoscaled using Horizontal Pod Autoscalers, reschedule pods that fail, and perform rolling updates and rollbacks. Naked pods (pods without any higher-level resource managing them) cannot do any of those.

Pods can run multiple containers and support various restart policies, so those are not reasons to prefer Deployments over naked Pods.

 </course/introduction-to-kubernetes/deployments/>

Covered in this lecture

Deployments

Course:Introduction to Kubernetes

11m

#12



What is the recommended Kubernetes resource for running applications?



Stand-alone Pod



Container



ReplicaSet




Deployment

Explanation

Deployments are a high-level concept for managing applications in Kubernetes. Pods are where applications actually run, but deployments make managing applications much simpler than working with pods directly. For example, you can easily scale and perform a rolling update to a new version using deployments.

Deployments automatically maintain a ReplicaSet for replicating pods, and are easier to work with than a ReplicaSet.

Containers aren't a type of Kubernetes resources.

 </course/introduction-to-kubernetes/deployments/>

Covered in this lecture

Probes

Course:Introduction to Kubernetes

11m

