TESTPASSPORT

練 習 問 題



高 品 質 な 学 習 リ ソ ー ス

https://www.testpassport.jp

1年間無料アップデート

Exam : **3V0-24.25**

Title : Advanced VMware Cloud Foundation 9.0 vSphere Kubernetes Service

Version : DEMO

1.A cloud platform team is deploying multiple Supervisors in a VMware Cloud Foundation (VCF) environment to support a multi-tenant VKS architecture. Each Supervisor will host a distinct set of namespaces mapped to different organizational tenants such as Finance, Public Services, and R&D, where each tenant consumes resources from dedicated workload domains.
What is the purpose of the vcf context create command?
A. Provides authentication to the Supervisor cluster.
B. Creates a connection to VCF Operations for the current session.
C. Creates a named set of access parameters for a VKS cluster.
D. Creates a new storage profile for a cluster.
Answer: C
Explanation:
CF 9.0 defines a VCF CLI context as a configuration set for a specific endpoint (for example, a Kubernetes cluster, vSphere Supervisor, or VCF Automation) and explains that a context stores the key connection parameters such as endpoint, context type, path to kubeconfig, and credentials. This aligns directly with option C, because vcf context create is used to create a named context that encapsulates the access parameters needed to work with a specific Supervisor or VKS cluster and to switch between multiple environments without rewriting configuration each time.
The documentation further shows vcf context create being used to connect either to a Supervisor endpoint or to a workload (VKS) cluster (by providing workload cluster name/namespace or by using a generated kubeconfig), which is exactly the multi-tenant operational pattern in the question. While authentication is part of what happens during context creation (for example, creating a configuration that includes a JWT for Kubernetes API access), the command's purpose is broader: it creates the reusable context object that defines where and how the CLI connects.

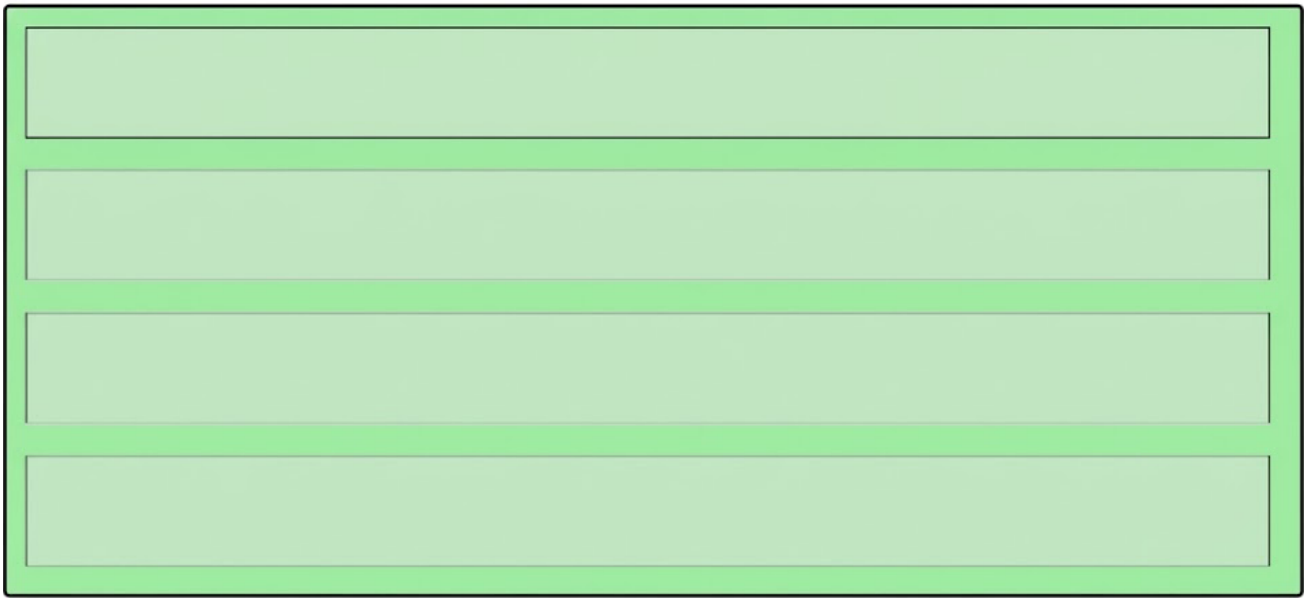
2.DRAG DROP
An administrator attempts to upgrade an existing Tanzu Kubernetes Cluster (TKC).
The update fails and results in a cluster status of upgrade failed. Sequence the actions to troubleshoot the failure.

Run `kubectl proxy` so that `curl` can be used to issue requests.

Log in to the Supervisor as an administrator.

Use `curl` to issue a patch command to clear the .status.conditions on the Job controller

Run `kubectl` to look up the update_job_name.



Answer:

Run `kubectl proxy` so that `curl` can be used to issue requests.

Log in to the Supervisor as an administrator.

Use `curl` to issue a patch command to clear the .status.conditions on the Job controller

Run `kubectl` to look up the update_job_name.

Log in to the Supervisor as an administrator.

Run `kubectl` to look up the update_job_name.

Run `kubectl proxy` so that `curl` can be used to issue requests.

Use `curl` to issue a patch command to clear the .status.conditions on the Job controller

Explanation:

Log in to the Supervisor as an administrator.

Run kubectl to look up the updatejob_name.

Run kubectl proxy so that curl can be used to issue requests.

Use curl to issue a patch command to clear the .status.conditions on the Job controller.

In VMware Cloud Foundation (VCF) 9.0, managing Tanzu Kubernetes Clusters (TKC) through the vSphere Kubernetes Service (VKS) involves a declarative lifecycle managed by the vSphere Supervisor. When a TKC upgrade enters an upgrade failed state, it is typically because the underlying cluster reconciliation logic has hit a terminal error or a timeout within the Cluster API (CAPI) controllers. To resolve this, an administrator must interact directly with the Supervisor's API server to reset the state of the upgrade job.

The troubleshooting workflow begins with authenticating to the Supervisor cluster using the kubectl vsphere login command (Step B). Once authenticated and the context is set to the correct namespace, the administrator must identify the specific job responsible for the upgrade failure by querying the namespace for job resources (Step D). Because the Supervisor API requires specific authentication tokens that are complex to manage via standard CLI tools for manual JSON patching, running kubectl proxy (Step A) is the standard procedure to create a local gateway. This allows the administrator to use a

simple curl command (Step C) to send a JSON Merge Patch to the API server. The critical step is clearing the .status.conditions field. By removing the "Failed" condition, the Supervisor's controller-manager is triggered to re-evaluate the desired state and attempt the upgrade process again. This sequence ensures the declarative engine can resume operations without requiring a full cluster redeployment.

3.HOTSPOT
The vSphere Admin creates a vSphere Namespace harbor-project and grants the DevOps Engineer edit permissions. The DevOps Engineer deploys the Harbor service in this namespace with the command: kubectl apply -f harbor-svc.yml
The Harbor service deploys successfully, but the database data harbor database 0 pods does not come into a Running state.
Click the two locations where the administrator should verify the Access Modes for this pod's Perslstent Volume Claim (PVC). (Choose two.)

Answer Area



Answer:

Answer Area



Explanation:
Storage
Resources
When a stateful pod (like Harbor's database) is stuck Pending/Container Creating or repeatedly failing to attach/mount storage, one of the first checks is whether the PVC's Access Modesmatch what the workload and backing storage support (for example, ReadWriteOnce vs ReadWriteMany). In Kubernetes, the PVC and PV explicitly show Access Modes(for example Access Modes: RWO) when you inspect them.
In the vSphere Namespace UI, the Storagetab is where namespace storage is managed and reviewed (including storage policy assignments that back storage classes/PVCs), making it the natural place to validate the storage configuration being consumed by the Harbor database PVC. VMware's VCF documentation directs administrators to navigate to a namespace and use theStoragetab for storage-related settings.
The Resource stab is the Kubernetes-object oriented view for the namespace, where you can locate the PVC object associated with the failing pod and inspect its properties/YAML (including the requested storage class and access mode intent). Namespace permissions explicitly include the ability to view Kubernetes resources within the namespace, which aligns with checking PVC details there.

4.DRAG DROP

Drag and drop the three features into the correct order from Possible Features list on the left and place them into the Provided by Service Mesh on the right side. (Choose three.)

Possible Features	Provided by Service Mesh
Federation	
Autoscaling	
Graphical User Interface	
Application backup	
Database connection management	
Observability	

Answer:

Possible Features	Provided by Service Mesh
Federation	Federation
Autoscaling	Database connection management
Graphical User Interface	Observability
Application backup	
Database connection management	
Observability	

Explanation:

Provided by Service Mesh (choose three, in order):

Federation

Graphical User Interface

Observability

A service mesh is an application networking layer that manages service-to-service communication across Kubernetes clusters, providing consistent connectivity, policy enforcement, and visibility without requiring application code changes. Federation is a service-mesh capability because modern meshes (especially multi-cluster/enterprise implementations) can connect services across multiple clusters and environments, enabling shared identity, cross-cluster service discovery, and uniform policy application (often described as multi-cluster or federated service connectivity). A Graphical User Interface is commonly provided alongside the service mesh platform to centrally configure policies (traffic routing, access controls, security settings) and to visualize service topology and health. Observability is a core service-mesh outcome: by inserting sidecar proxies (or equivalent data plane components) into the data path, the mesh can generate consistent metrics, logs, and distributed traces for service traffic, enabling latency/error monitoring and dependency mapping.

The other options are not service-mesh features: Autoscaling is handled by Kubernetes/HPA and metrics pipelines, application backup is typically provided by backup tools (e.g., Velero-like solutions), and database connection management is handled by application frameworks or database proxies rather than the service mesh itself.

5.An administrator is tasked to install Istio service mesh for the vSphere Kubernetes Service (VKS).

What command installs Istio?
A. kubectl istioctl install
B. curl -L https://istio.io/downloadIstio
C. VCF package install istioctl
D. docker build istioctl
Answer: A
Explanation:
For administrators managing modern application workloads within VMware Cloud Foundation (VCF) 9.0, the vSphere Kubernetes Service (VKS) provides the infrastructure layer required for advanced networking via service meshes. While VKS offers various integrated services, Istio is typically deployed as a manual add-on to the workload clusters to provide advanced traffic management, observability, and security.
The official method for deploying Istio into a VKS-managed cluster is via the istioctl command-line utility. While curl (Option B) is frequently used to download the installation script and binary to the administrator's workstation, it does not perform the installation itself.
The command that actually executes the logic to deploy the Istio control plane (istiod), configures the necessary Custom Resource Definitions (CRDs), and sets up the required namespaces is istioctl install. In many enterprise environments and documentation contexts, this utility is integrated or utilized as a plugin, often referred to in the context of the kubectl toolset (Option A). This command applies the selected configuration profile (such as 'default' or 'demo') to the cluster, enabling features like mutual TLS (mTLS) and fine-grained routing policies. In VCF 9.0, ensuring Istio is correctly installed is a prerequisite for implementing Zero Trust security architectures across the SDDC, as it allows for policy-driven communication between microservices running on Supervisor-managed Kubernetes clusters.