PassQueen
a.k.a dumpsqueen.com

Associate VMware Application Modernization

VMware 1V0-71.21

Version Demo

Total Demo Questions: 8

Total Premium Questions: 80

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Kubernetes Fundamentals	36
Topic 2, VMware Tanzu Fundamentals	25
Topic 3, Application Modernization Fundamentals	12
Topic 4, Cloud Native Architecture and Design	1
Topic 5, Cloud Native Operations	6
Total	80

QUESTION NO: 1

What are two ways a container in a Pod can consume ConfigMap data? (Choose two.)

A. Mounted as a Persistent Volume Claim (PVC)

B. From Read-only files

C. From Read-write files

D. From Environment variables

E. Containers do not consume ConfigMap data

ANSWER: B D

Explanation:

ConfigMap data can be consumed by a container as environment variables or by mounting the ConfigMap as a volume, which exposes its keys as files in the container filesystem. Therefore, *From Environment variables* and *From Read-only files* are the applicable methods. Environment-variable consumption is useful when an application reads configuration during startup; individual keys can be mapped to named variables, or multiple keys can be imported together. With the volume approach, each ConfigMap key is represented as a file whose content is the corresponding value, allowing applications to read configuration from a known directory path. Kubernetes documents ConfigMap volumes as read-only mounts, which protects the projected configuration from being modified by the container. These mechanisms let configuration be kept separate from a container image, supporting reuse of the same image across environments with environment-specific settings supplied at deployment time. See the Kubernetes ConfigMap documentation and the guide to configuring Pods with ConfigMaps.

QUESTION NO: 2

What are three features of Harbor? (Choose three.)

A. Security and Vulnerability Analysis

B. Container Build Service

C. Role-Based Access Control

D. Replication across multiple registries

E. Secure Container Runtime

F. SQL storage

ANSWER: A C D

Explanation:

Harbor is a cloud-native registry designed to manage, secure, and distribute container images and related artifacts. **Security and Vulnerability Analysis** is a core Harbor capability: Harbor can scan images for known vulnerabilities using an integrated scanner, apply vulnerability severity policies, and expose scan results to registry users and automation. **Role-Based Access Control** is also fundamental to Harbor projects, enabling administrators to assign project-scoped roles and control who can pull, push, manage members, or administer repositories. This supports secure separation of teams, applications, and environments within a shared registry platform.

Replication across multiple registries is another Harbor feature. Replication policies can synchronize images and artifacts between Harbor instances and supported external registries, helping organizations distribute content across sites, support regional deployments, and maintain registry availability. Harbor's project-based repository management, security scanning, access controls, and replication capabilities are specifically documented as key registry functions. See the Harbor vulnerability scanning documentation and the Harbor replication documentation.

QUESTION NO: 3

What is the result of Continuous Delivery?

A. An application build has been produced and ready for deployment to production

B. A broken application build is able to be fixed rapidly

C. An application build artifact has been produced

D. An application build has been produced and deployed to production

ANSWER: A

Explanation:

Continuous Delivery results in an application build that has passed the automated stages required to be considered deployable to production. The delivery pipeline compiles, tests, validates, and packages changes so that the software is always in a release-ready state. A production deployment can therefore be performed when the business or release owner chooses, typically through an explicit approval or release decision. This distinction is central to Continuous Delivery: frequent, reliable production-ready builds are created, but releasing them to users is not necessarily automatic.

In a VMware application-modernization environment, this outcome supports repeatable delivery of containerized applications through pipelines and consistent promotion across environments. Teams gain confidence that each validated build can be released without a lengthy, risky stabilization phase. The build is not merely an artifact; it has completed the quality and deployment-readiness activities defined by the delivery process. For a concise definition of Continuous Delivery and its distinction from Continuous Deployment, see Atlassian's CI, Continuous Delivery, and Continuous Deployment guide. The related deployment-pipeline practices are also described in Martin Fowler's Continuous Delivery overview.

QUESTION NO: 4

What does it mean by a "lift and shift" approach regarding moving applications to the cloud?

A. Retiring the application and replacing it with a different application

B. Code changes and user facing changes

C. No code changes and no user facing changes

D. Code changes and no user facing changes

ANSWER: C

Explanation:

"No code changes and no user facing changes" correctly describes a lift-and-shift migration. This approach moves an existing application from its current hosting environment to cloud infrastructure substantially as it is, rather than redesigning its architecture or changing its business functionality. The application's runtime, dependencies, configuration patterns, and operating model may require environment-specific adjustments to enable deployment in the destination cloud, but the application itself is not refactored into new cloud-native components as part of the migration.

Because the application behavior and user experience are preserved, lift and shift can be a practical first step for organizations seeking to exit a data center, reduce infrastructure-management responsibilities, or establish an initial cloud presence with limited disruption. It is distinct from application modernization activities such as replatforming, refactoring, or rebuilding, which intentionally make deeper changes to improve cloud-native scalability, resiliency, delivery speed, or operational automation. VMware Tanzu describes modernization as a progression that can include moving applications before subsequently improving them for modern platforms and practices. See VMware Tanzu: What Is Application Modernization? and AWS: What Is Lift and Shift?.

QUESTION NO: 5

Which two statements describe a GitOps operating model? (Choose two.)

A. Git is used as the source of truth for desired application and infrastructure configuration

B. An automated agent reconciles the deployed environment with the configuration stored in Git

C. Cluster changes are made only by directly editing running Pods

D. Container images must be built manually on each worker node

E. Application configuration cannot be version controlled

ANSWER: A B

Explanation:

Git is used as the source of truth for desired application and infrastructure configuration is correct. Declarative manifests and configuration changes are stored, reviewed, and versioned in Git repositories. **An automated agent reconciles the deployed environment with the configuration stored in Git** is also correct. The agent continuously compares the actual environment with the desired state and applies changes to bring the environment into alignment.

GitOps supports auditable, repeatable delivery because changes are proposed and tracked through version control rather than being made only through direct manual modification of a running cluster. See the OpenGitOps principles and the Kubernetes documentation on declarative configuration.

QUESTION NO: 6

Which are two cluster management capabilities of Tanzu Mission Control? (Choose two.)

A. Data protection

B. Multi-cloud distributed clusters

C. Cluster migration

D. Data encryption

E. Cluster observability and diagnostic

ANSWER: A E

Explanation:

Tanzu Mission Control provides centralized management for Kubernetes clusters across environments, including capabilities that help operators protect workloads and maintain operational visibility. **Data protection** is a cluster-management capability because Tanzu Mission Control integrates backup and restore workflows for Kubernetes resources and persistent application data. Administrators can define protection policies and manage backup-related operations consistently across managed clusters, supporting recovery requirements for modern applications.

Cluster observability and diagnostic is also a cluster-management capability. Tanzu Mission Control provides cluster health and inspection capabilities that help administrators assess cluster configuration, identify operational issues, and collect diagnostic information. This centralized insight is especially valuable when clusters are distributed among multiple Kubernetes platforms and cloud environments, because it gives teams a common operational view and a consistent process for investigating cluster state.

These capabilities align with Tanzu Mission Control's role as a centralized control plane for Kubernetes fleet operations: it applies governance while also helping teams protect application data and observe the health and configuration of their clusters. See the VMware Tanzu Mission Control concepts documentation and the Tanzu Mission Control data protection documentation.

QUESTION NO: 7

Which state is invalid in a Pod's lifecycle?

A. Failed

B. Running

C. Succeeded

D. Setup

E. Pending

ANSWER: D

Explanation:

Setup is not a valid Kubernetes Pod phase. Kubernetes defines the Pod `status.phase` field as a concise, high-level summary of where a Pod is in its lifecycle. The defined phases include `Pending`, when the Pod has been accepted by the cluster but one or more containers have not yet been created and started; `Running`, when the Pod has been bound to a node and at least one container is running or starting; `Succeeded`, when all containers have terminated successfully and will not be restarted; and `Failed`, when all containers have terminated and at least one ended in failure. Kubernetes also documents `Unknown` for cases where the Pod state cannot be obtained, commonly because of an issue communicating with the node. Although setup activities, such as image pulling, volume mounting, scheduling, and container creation, can occur before a workload becomes operational, Kubernetes does not expose a Pod lifecycle phase named `Setup`. Therefore, it is the invalid state among the listed values. See the Kubernetes documentation on Pod lifecycle and the API reference for PodStatus and phase.

QUESTION NO: 8

What is the purpose of a Kubernetes Ingress object?

A. To specify HTTP and HTTPS traffic routing rules from outside the cluster to Services within the cluster

B. To load balance across a set of Pods

C. To manage the number of instances of a container image running within the cluster

D. To manage traffic between Services within the cluster

ANSWER: A

Explanation:

The purpose of a Kubernetes Ingress object is to define HTTP and HTTPS routing rules that direct traffic originating outside a Kubernetes cluster to the appropriate Services inside that cluster. An Ingress can map requests according to host names and URL paths, such as routing `app.example.com/api` to one Service and `app.example.com/web` to another. It can also define TLS settings so that HTTPS connections are terminated and managed consistently at the cluster entry point. Ingress is an API resource, so its rules are implemented by an Ingress controller, such as NGINX or another controller installed for the cluster. This provides a central, declarative way to expose multiple web applications without requiring a separate external load balancer configuration for every Service. Kubernetes documents Ingress as an API object for managing external access to Services, typically HTTP, and notes that routing is provided through an Ingress controller. VMware vSphere with Tanzu similarly describes an Ingress resource as providing external HTTP/HTTPS access to Services through routing rules. See the Kubernetes Ingress documentation and VMware vSphere with Tanzu documentation.