PassQueen
a.k.a dumpsqueen.com

NetApp Certified Hybrid Cloud Implementation Engineer

Netapp NS0-403

Version Demo

Total Demo Questions: 8

Total Premium Questions: 88

Buy Premium PDF

https://passqueen.com

support@passqueen.com

Topic Break Down

Topic	No. of Questions
Topic 1, Cloud Manager	30
Topic 2, Cloud Volumes ONTAP	24
Topic 3, StorageGRID	7
Topic 4, NetApp HCI	9
Topic 5, Mix Questions	18
Total	88

QUESTION NO: 1

You are designing a StorageGRID ILM rule to protect objects by using erasure coding across geographically separate sites.

Which two statements are correct about this design? (Choose two.)

A. An erasure coding profile must be created before it can be used in an ILM rule.

B. The selected sites must support the site count required by the erasure coding scheme.

C. Erasure coding can be configured only for objects stored in a Cloud Storage Pool.

D. An erasure coding profile removes the need for Storage Nodes at the selected sites.

ANSWER: A B

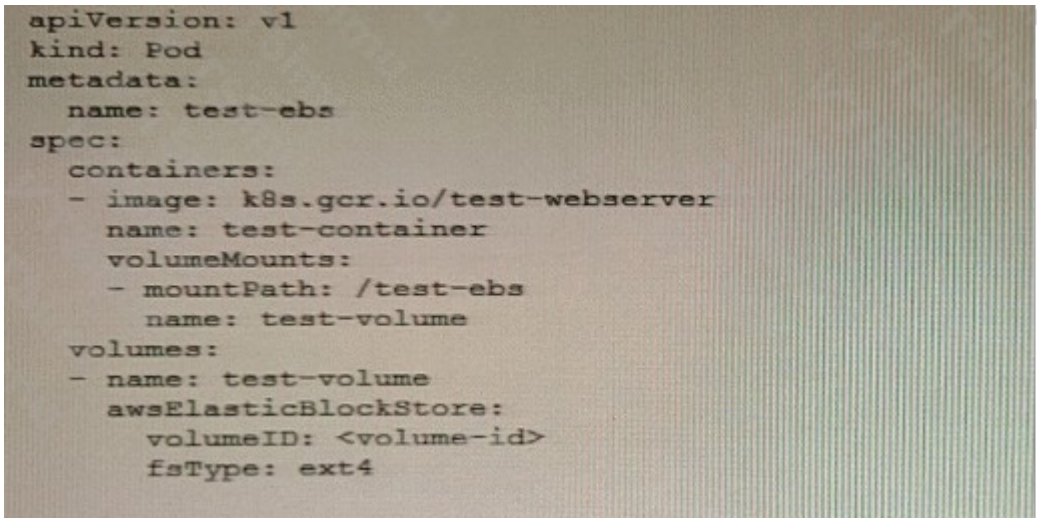
Explanation:

An erasure coding profile specifies the erasure coding scheme and identifies the StorageGRID sites used for the encoded object fragments. The profile must be created before it can be selected in an ILM rule. This enables StorageGRID to distribute fragments according to the selected scheme and placement topology.

The number of sites selected in the erasure coding profile must support the chosen scheme. For example, a scheme that distributes fragments across three sites requires the profile to include the required number of storage sites. StorageGRID must also have sufficient Storage Nodes and usable capacity at each selected site to store the fragments. See the StorageGRID erasure coding documentation and the StorageGRID erasure coding profile documentation.

QUESTION NO: 2

Click the Exhibit button.



```
apiVersion: v1
kind: Pod
metadata:
  name: test-ebs
spec:
  containers:
  - image: k8s.gcr.io/test-webserver
    name: test-container
    volumeMounts:
    - mountPath: /test-ebs
      name: test-volume
  volumes:
  - name: test-volume
    awsElasticBlockStore:
      volumeID: <volume-id>
      fsType: ext4
```

You are reviewing a Kubernetes manifest to deploy an Amazon Elastic Block Store (AWS EBS).

Which two Kubernetes API object types are shown in the exhibit?

A. Secret

B. Pod

C. Volume

D. Config

ANSWER: B

Explanation:

Pod is correct because an Amazon EBS volume is configured as storage within a Pod specification. A Kubernetes Pod is an API object represented by a manifest containing fields such as `apiVersion`, `kind: Pod`, `metadata`, and `spec`. Within the Pod's `spec`, the manifest can define a `volumes` section and mount that volume into a container through `volumeMounts`. The EBS-specific settings identify the EBS volume to attach and the filesystem parameters used by the workload. This structure describes a workload unit that Kubernetes schedules and runs, with the EBS storage made available to its container. Kubernetes documentation describes Pods as the smallest deployable computing objects and documents volumes as storage declared in a Pod specification and mounted by containers. See the Kubernetes Pods documentation and the Kubernetes Volumes documentation.

QUESTION NO: 3

Which DevOps principle implies that your documentation and your configuration are the same?

A. infrastructure as code

B. continuous testing

C. continuous deployment

D. continuous monitoring

ANSWER: A

Explanation:

Infrastructure as code is the DevOps principle in which infrastructure definitions, desired configuration, and deployment procedures are expressed in version-controlled, machine-readable code. The configuration files themselves become the authoritative operational documentation: they specify what resources should exist, how they are configured, and how the environment can be recreated consistently. Because changes are reviewed, committed, tested, and tracked through source control, the documented state remains aligned with the deployed configuration rather than relying on manually maintained documents that can become outdated. This approach also supports repeatability across development, test, and production environments, enables peer review and auditing of changes, and reduces configuration drift. For NetApp environments, automation tooling and APIs can be used with infrastructure-as-code workflows to define and manage storage resources consistently. HashiCorp describes infrastructure as code as using configuration files to define and provision infrastructure, while NetApp's ONTAP automation documentation provides the REST API foundation commonly used by such automated workflows. See HashiCorp: What is infrastructure as code? and NetApp ONTAP REST API automation documentation.

QUESTION NO: 4

Your customer wants to stream large amounts of real-time data from multiple sources at the application level into their Apache Spark platform for transformation before it goes into their machine learning platform.

In this scenario, which technology should be used for this task?

A. NetApp SnapMirror

B. Apache Kafka

C. NetApp Global File Cache

D. NetApp Cloud Sync service

ANSWER: B

Explanation:

Apache Kafka is the appropriate technology because it is a distributed event-streaming platform designed to ingest, durably store, and deliver high volumes of real-time data from many producing applications. Kafka organizes incoming events into

topics and partitions, enabling scalable, parallel consumption by downstream processing systems. This makes it well suited to decouple multiple data sources from an Apache Spark transformation pipeline while maintaining reliable, ordered streams within partitions.

Apache Spark Structured Streaming has native Kafka integration: Spark can subscribe to Kafka topics, continuously process records as they arrive, apply transformations, and write the resulting data to storage or machine-learning data preparation pipelines. Kafka also supports consumer groups, retention, replay of historical events, and fault-tolerant processing patterns, which are valuable when transformations must be restarted or when downstream consumers need to independently process the same source stream. This architecture provides a scalable application-level ingestion layer before Spark processing rather than merely copying files or replicating storage data. See the Apache Spark Structured Streaming + Kafka Integration Guide and the Apache Kafka documentation.

QUESTION NO: 5

You are developing microservices on a Kubernetes system. When developers push new code to a branch, you want to have your application deployed on Kubernetes using the CI/CD pipeline that automatically picks up new changes in the main branch-In this scenario, what are three steps to accomplish this deployment workflow? (Choose three.)

A. Issue a pull request.

B. Reset the Git branches.

C. Commit changes to a Git branch

D. Configure Git hooks.

E. Merge code changes to the main branch.

ANSWER: A C E

Explanation:

A branch-based CI/CD workflow starts when developers *Commit changes to a Git branch*. The branch provides an isolated place to develop and validate a change before it affects the deployable source of truth. Developers then *Issue a pull request* to propose integrating that branch into the shared mainline. Pull requests provide the review, approval, and automated validation checkpoint that teams commonly use before production-oriented code is accepted.

After the required checks and reviews are complete, the workflow must *Merge code changes to the main branch*. This is the event that makes the new application or deployment definition available on the branch watched by the CI/CD or GitOps process. A pipeline trigger configured for the main branch can then build the updated container image, publish it to its registry, and apply or reconcile the Kubernetes manifests so the desired workload revision is deployed. This sequence maintains a clear, auditable promotion path: development work occurs in a feature branch, review occurs through the pull request, and deployment automation responds to the accepted main-branch revision. GitHub documents the standard pull-request merge process at About pull requests; Kubernetes documents declarative workload deployment concepts at Deployments.

QUESTION NO: 6

You used a Terraform configuration to create a number of resources in Google Cloud for testing your applications. You have completed the tests and you no longer need the infrastructure. You want to delete all of the resources and save costs.

In this scenario, which command would you use to satisfy the requirements?

A. terraform untaint

B. terraform apply

C. terraform force-unlock

D. terraform destroy

ANSWER: D

Explanation:

`terraform destroy` is the appropriate command because it creates and executes a destruction plan for all infrastructure resources tracked in the current Terraform state. In this case, the Google Cloud resources were provisioned through the Terraform configuration for temporary application testing and are no longer required. Running `terraform destroy` instructs Terraform to identify the managed resources, show the planned deletions for confirmation, and then remove them from Google Cloud. This ensures that resources such as compute instances, networks, storage, and other billable services managed by the state are deprovisioned, helping prevent unnecessary ongoing charges. Terraform uses its state and provider configuration to delete the resources in the proper dependency order, so dependent resources are handled safely. The command can be run with an auto-approval flag in an appropriately controlled automated workflow, but interactive plan review is generally preferred before deleting infrastructure. HashiCorp documents `terraform destroy` as the command for destroying Terraform-managed infrastructure: Terraform destroy command reference. For the standard workflow and state-based resource lifecycle behavior, see the Terraform CLI workflow documentation.

QUESTION NO: 7

An organization is implementing infrastructure as code to deploy NetApp Cloud Volumes ONTAP instances.

In this scenario, what are two ways in which this implementation improves operation efficiencies? (Choose two.)

A. Defining the system as code makes it more secure.

B. Best practices can be implemented directly into configurations.

C. Deployment failures are automatically resolved by configuration management.

D. System configurations can be version controlled.

ANSWER: B D

Explanation:

Infrastructure as code improves operational efficiency by expressing the desired Cloud Volumes ONTAP environment in reusable, declarative configuration files. “Best practices can be implemented directly into configurations” is correct because standardized templates can embed approved settings for instance sizing, storage configuration, networking, security controls, tagging, and other deployment requirements. Each deployment can then apply the same validated standards without requiring administrators to manually repeat configuration decisions. “System configurations can be version controlled” is also correct because IaC files can be stored in a source-control repository. Version control provides a history of changes, supports peer review and approval workflows, enables rollback to a known-good configuration, and allows teams to use the same tested configuration across environments. Together, these practices make Cloud Volumes ONTAP deployments more repeatable, auditable, and consistent while reducing manual effort and configuration drift. NetApp supports infrastructure automation for BlueXP and Cloud Volumes ONTAP through Terraform-based workflows and provider resources. See the NetApp BlueXP automation documentation and the NetApp Cloud Manager Terraform provider documentation.

QUESTION NO: 8

Your organization is adopting DevOps principles to deliver innovation rapidly. Which three methods support this approach? (Choose three.)

A. Automate processes as much as possible.

B. Plan to reduce the blast radius of failures.

C. Implement version control for code changes.

D. Implement daily stand-up meetings for all team members.

E. Allow more people within the organization to approve changes.

ANSWER: A B C

Explanation:

DevOps emphasizes shortening the path from an idea to a reliable production release by combining collaborative practices with repeatable technical controls. **Automate processes as much as possible.** is fundamental because automation makes builds, tests, deployments, infrastructure provisioning, and operational checks consistent and repeatable. It also reduces manual handoffs, allowing teams to release changes more frequently while maintaining quality.

Plan to reduce the blast radius of failures. supports rapid delivery by ensuring that an unexpected issue affects the smallest possible scope. Practices such as incremental releases, canary deployments, feature flags, isolation boundaries, monitoring, and quick rollback mechanisms enable teams to experiment and deploy with controlled risk rather than delaying all change.

Implement version control for code changes. provides the auditable, collaborative source of truth required for DevOps workflows. Source control enables peer review, branching and merging, traceability between a change and a deployment, and automated continuous integration pipelines that validate each update. Together, automation, limited failure impact, and version-controlled changes create a delivery process that is both fast and dependable. See Microsoft Learn: What is DevOps? and Pro Git: About Version Control.