RedHat

EX380

Red Hat Certified Specialist in OpenShift Automation and Integration

Questions & Answers (Demo)

Version: 4.0

Question: 1

SIMULATION

Task SIMULATION 1
Node Management – Remove Taint on Worker Node

Answer: See the solution below in Explanation:

Explanation:

Step 1: Log in to the OpenShift web console with an account that has sufficient cluster administrative privileges.
This Task is performed from the GUI, not the CLI. The lab hint explicitly places this under the worker node details page in the console.

Step 2: Navigate to Compute.
This area contains node-level resources, including control plane and worker nodes.

Step 3: Open Nodes.
Here you can view all nodes currently registered in the cluster.

Step 4: Select the required worker node.

Choose the exact worker node referenced by the lab Task SIMULATION.

Step 5: Open the Details tab.
The taint configuration is managed from the selected node's details view.

Step 6: Locate the Taints section and click Edit.
A taint is used to control pod scheduling. If a worker has a taint, pods without matching tolerations may not schedule there.

Step 7: Remove the unwanted taint entry.
Removing the taint makes the worker eligible again for normal scheduling behavior, depending on the rest of the cluster policy.

Step 8: Click Save.
This commits the change so the node is updated and the scheduler can evaluate it without that taint.

==========

Question: 2

SIMULATION

Task SIMULATION 2
Identity Management – Create HTPasswd Secret

Answer: See the solution below in Explanation:

Explanation:

Step 1: Open a terminal with oc access to the cluster.
This Task is CLI-driven and targets the openshift-config namespace.

Step 2: Run the command:
oc create secret generic rhds-ldap-secret --from-literal bindPassword=redhatocp -n openshift-config

Step 3: Verify that the secret is created successfully.
The lab output shows:

secret/rhds-ldap-secret created

Detailed explanation:
This step creates a generic secret named rhds-ldap-secret in the openshift-config namespace. The secret stores a key called bindPassword with the value redhatocp. In an identity-provider or LDAP integration workflow, the bind password is used by OpenShift when connecting to the external directory service. Storing this value in a secret is the correct operational pattern because authentication material should not be embedded directly into configuration objects. The openshift-config namespace is specifically important because cluster authentication configuration commonly references secrets and configmaps from that namespace. If the secret name or key is wrong, the authentication configuration that depends on it may fail to validate or connect properly.

==========

Question: 3

SIMULATION

Task SIMULATION 3
Identity Management – Create CA ConfigMap

Answer: See the solution below in Explanation:

Explanation:

Step 1: Ensure the certificate file rhds_ca.crt is available in your current working directory or use the correct path.
The Task requires creating a configmap from this CA certificate file.

Step 2: Run the command:
oc create configmap rhds-ca-config-map --from-file ca.crt=rhds_ca.crt -n openshift-config

Step 3: Confirm the configmap is created.
The lab output shows:
configmap/rhds-ca-config-map created

Detailed explanation:

This creates a configmap named rhds-ca-config-map in the openshift-config namespace and maps the local file rhds_ca.crt to the key name ca.crt inside the configmap. This is important in external identity integration because OpenShift may need to trust a custom certificate authority when communicating with LDAP or another secured external service. By placing the certificate in a configmap, the authentication operator or related cluster configuration can reference it cleanly. The key name matters because many OpenShift resources expect a CA bundle key with a specific filename-like convention. If the file path is wrong, the command fails immediately. If the configmap name or key mapping is wrong, the authentication provider referencing it may not trust the external endpoint.

==========

Question: 4

SIMULATION

Task SIMULATION 4
Backup and Restore – Restore Application from Existing Backup

Answer: See the solution below in Explanation:

Explanation:

Step 1: Make sure Velero is installed and configured in the environment.
The Task SIMULATION assumes an existing backup named backup-app-daily is already present.

Step 2: Run the restore command:
velero restore create --from-backup backup-app-daily

Step 3: Confirm the restore request is submitted.
The lab output shows:
Restore request "backup-app-daily-2024" submitted successfully.

Detailed explanation:
This command instructs Velero to create a restore operation using the existing backup called backup-app-daily. Velero is commonly used to protect Kubernetes and OpenShift resources by backing up

object definitions and, when configured, persistent data integrations. The command does not manually recreate resources one by one; instead, it leverages the metadata captured during backup. A successful restore submission means the request has been accepted, not necessarily that every object has already been fully restored. In practical administration, you would often follow this by checking restore status and validating the application namespace, pods, services, routes, and storage bindings. This lab Task SIMULATION focuses specifically on initiating the restore from the named backup source.

==========

Question: 5

SIMULATION

Task SIMULATION 5
Backup and Restore – Fix SCC for Restored Application

Answer: See the solution below in Explanation:

Explanation:

Step 1: Identify the application namespace after restore.
The lab shows the namespace as my-app-namespace.

Step 2: Run the SCC assignment command:
oc adm policy add-scc-to-user anyuid -z default -n my-app-namespace

Step 3: Confirm the role binding is applied.
The lab output shows:
clusterrole.rbac.authorization.k8s.io/system:openshift:scc:anyuid added: "default"

Detailed explanation:
After a restore, the application may fail if its pods require a security context not permitted by the default SCC allocation. This command grants the anyuid SCC to the default service account in the my-app-namespace project. The -z default syntax targets the default service account, which many restored workloads use if no custom service account is defined. The anyuid SCC allows containers to

run with arbitrary user IDs, which some legacy or prebuilt images require. In OpenShift, SCC mismatches commonly cause pods to remain in pending or crash-related states. Assigning the proper SCC resolves those admission issues so workloads can start successfully. This step is therefore a post-restore operational fix to align security policy with application requirements.