1. Analyze audit logs:

```
tail -f /var/log/audit/audit.log
```

Explanation:
Audit logs provide detailed records of user actions, essential for detecting and responding to security incidents.

2. Create a project template YAML:

```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
name: project-template
objects:
- apiVersion: v1 kind: Deployment metadata:
name: nginx-deployment
spec:
replicas: 2
template:
spec:
containers:
- name: nginx
image: nginx
- apiVersion: v1 kind: Service metadata:
name: nginx-service spec:
ports:
- protocol: TCP port: 80 selector: app: nginx
```

3. Test the HTTPS connection: `curl -k https://secure.example.com`
Explanation:
Re-encrypt termination provides end-to-end encryption, ensuring traffic remains encrypted from the client to the backend service.

4. Create a LoadBalancer service YAML file:

```
apiVersion: v1
kind: Service
metadata:
name: external-lb-service
spec:
type: LoadBalancer
selector:
app: my-app
ports:
- protocol: TCP
port: 80
targetPort: 8080
```

5. Deploy the LoadBalancer service:

```
kubectl expose deployment nginx-deployment --type=LoadBalancer --port=80
```

6. Add node affinity rules:

```
affinity:
nodeAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
```

nodeSelectorTerms:
- matchExpressions:
- key: node-role.kubernetes.io/worker
operator: In values:
- true

7. Diagnose network policies: kubectl get networkpolicy

8. Create a TLS secret:
oc create secret tls my-ingress-tls --cert=cert.pem --key=key.pem -n app-security

9. Monitor canary performance: kubectl logs <canary-pod-name>
Explanation:
Canary deployments reduce risks by testing updates on a small subset of traffic, allowing issues to be identified before a full rollout.

10. List all users and groups:
oc get users
oc get groups

11. Verify the user's permissions:
kubectl auth can-i list pods --as dev-user -n test-project
Explanation:
RBAC provides fine-grained access control, enabling administrators to assign specific permissions to users or groups. Verification ensures the intended access level is configured.

12. Apply the quota:
oc apply -f cluster-quota.yaml

13. Apply all YAML files:
kubectl apply -f mysql-pv.yaml
kubectl apply -f mysql-pvc.yaml
kubectl apply -f mysql-statefulset.yaml

14. Deploy a Kubernetes Deployment that uses a Secret to store sensitive information like database credentials. Validate that the application uses the Secret.
Answer:
See the Solution below.
Solution:

15. Deploy a StatefulSet with persistent storage to run a MySQL database. Validate that data persists after pod restarts.
Answer:
See the Solution below.
Solution:

16. Create a TLS secret:
oc create secret tls my-tls-secret --cert=cert.pem --key=key.pem -n app-security

17. Manually update an application by tagging a new image in an ImageStream. Validate the deployment.
Answer:
See the Solution below.

Solution:

18. Apply the DeploymentConfig: oc apply -f deploymentconfig.yaml

19. Describe the pod to identify configuration issues: kubectl describe pod <pod-name>

20. Restart the pod and verify data persistence:
kubectl delete pod <redis-pod>
redis-cli get key
Explanation:
StatefulSets with persistent storage ensure data durability, even when individual pods are restarted.

21. Define the pod YAML:

```
apiVersion: v1
kind: Pod
metadata:
name: raw-socket-pod
namespace: app-security
spec:
containers:
- name: busybox image: busybox securityContext: capabilities:
add: ["NET_RAW"]
command: ["sh", "-c", "sleep 1000"]
```

22. Apply the HPA:
kubectl apply -f hpa.yaml

23. Create an HPA configuration:

```
apiVersion: autoscaling/v2beta2
kind: HorizontalPodAutoscaler
metadata:
name: custom-hpa
spec:
scaleTargetRef:
apiVersion: apps/v1
kind: Deployment
name: my-app
minReplicas: 2
maxReplicas: 10
metrics:
- type: Pods pods:
metricName: custom_metric target:
type: AverageValue
averageValue: 500m
```

24. Deploy a multi-container pod where one container runs the main application, and the other container logs application data to an external service. Validate the sidecar functionality.
Answer:
See the Solution below.
Solution:

25. Check for remaining resources: oc get pods -n monitoring
Explanation:
Removing operators and verifying the cleanup ensures no orphaned resources remain, preventing potential issues with the cluster's state.

26. Deploy a Redis custom resource: apiVersion: redis.redis.opstreelabs.in/v1beta1

```
kind: Redis
metadata:
name: redis-cluster
spec:
size: 3
```

27. Create a Job with a TTL configuration:

```
apiVersion: batch/v1
kind: Job
metadata:
name: cleanup-job
spec:
ttlSecondsAfterFinished: 3600
template:
spec:
containers:
- name: cleanup-container image: busybox
command: ["sh", "-c", "echo Job complete!"]
restartPolicy: Never
```

28. Configure a re-encrypt route for an application to ensure traffic is encrypted from the client to the router and from the router to the backend service.
Answer:
See the Solution below.
Solution:

29. Label the namespaces and pods:

```
kubectl label namespace allowed-namespace name=allowed-namespace kubectl label pod allowed-
pod app=allowed-pod -n allowed-namespace
```

30. Create a StatefulSet YAML for Redis:

```
volumeClaimTemplates:
- metadata:
name: redis-data
spec:
accessModes: ["ReadWriteOnce"]
resources:
requests:
storage: 5Gi
```

31. Review node resource usage:

```
kubectl top nodes
```

32. Set up a network policy to allow ingress traffic from pods in a specific namespace only. Validate the restricted access.

Answer:
See the Solution below.
Solution:

33. Configure a resource quota to limit the total number of CPU cores used by a namespace. Validate by exceeding the CPU limit.
Answer:
See the Solution below.
Solution:

34. Validate backups:

```
kubectl get jobs
ls  /data/backup
```

Explanation:
CronJobs automate recurring tasks like backups, ensuring critical processes are performed on schedule without manual intervention.

35. Check service logs:

```
kubectl get events --field-selector involvedObject.name=<service-name>
```

36. Update or delete conflicting policies:

```
oc delete networkpolicy <conflicting-policy-name>
```

Explanation:
Network policy conflicts often arise from overlapping or incorrect selectors. Resolving these ensures the intended communication paths are restored.

37. Install the Vertical Pod Autoscaler:

```
kubectl apply -f
https://github.com/kubernetes/autoscaler/releases/latest/download/vertical-pod-autoscaler.yaml
```

38. Test the header injection: `curl -I http://example.com`
Explanation:
Custom headers are used for traffic redirection or additional metadata, ensuring applications receive context-specific requests.

39. Apply the CronJob:

```
oc apply -f db-backup.yaml
```

40. Install the Cert Manager Operator from the OperatorHub using the OpenShift web console. Validate the installation.
Answer:
See the Solution below.
Solution:

41. Deploy the application: `kubectl apply -f deployment.yaml`

42. Apply the quota:

```
oc apply -f pvc-quota.yaml
```

43. Update DNS or /etc/hosts to point custom-domain.example.com to the cluster IP.

44. Create the secret:
oc create secret generic db-secret --from-literal=username=admin --from-literal=password=secret123 -n app-security

45. Identify the image digest:
docker inspect nginx:latest | grep Digest

46. Bind the role to the user:
oc create rolebinding restricted-binding --role=restricted-role --user=user1 -n dev-namespace

47. Insert data into PostgreSQL and restart a pod:
psql -h <pg-host> -U user -d dbname -c "INSERT INTO test VALUES (1, 'data');"
kubectl delete pod <pg-pod>

48. Configure OpenShift logging to send application logs to an external Elasticsearch cluster. Include steps for Fluentd configuration and validation.
Answer:
See the Solution below.
Solution:

49. Configure a pod with a ConfigMap-mounted volume for application configuration. Validate that the pod accesses the ConfigMap data.
Answer:
See the Solution below.
Solution:

50. Create a namespace:
oc create namespace knative-serving

51. Simulate high memory usage:
kubectl exec <pod-name> -- stress --vm 1 --vm-bytes 100M --timeout 60s

52. Apply the quota:
oc apply -f deployment-quota.yaml

53. List pods with the label: kubectl get pods -l app=frontend
Explanation:
Labels and selectors enable efficient filtering and management of resources in OpenShift.

54.1.You are tasked with deploying a highly available application in OpenShift. Create a Deployment using YAML to deploy the nginx container with three replicas, ensuring that it runs successfully. Verify that the Deployment is active, all replicas are running, and the application can serve requests properly. Provide a complete walkthrough of the process, including necessary commands to check deployment status.
Answer:
See the Solution below.
Solution:

55. Validate by logging in with the new password: oc login -u admin -p <new-password>

Explanation:
Password modifications using the HTPasswd file ensure credentials are updated promptly, enhancing security.

56. Define the project template YAML:
```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
name: project-template-pvc
objects:
- apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: default-pvc
spec:
accessModes:
- ReadWriteOnce resources: requests:
storage: 10Gi
```

57. Validate the deployment: kubectl get pods
Explanation:
Operators automate the deployment and management of complex applications, such as databases, by abstracting operational logic.

58. Validate read access:
```
oc auth can-i get pods -n dev-namespace --as=user1
```

59. Modify the password policy for HTPasswd users to enforce complexity requirements.
Answer:
See the Solution below.
Solution:

60. Simulate and troubleshoot a pod network issue where a pod cannot communicate with a service in the same namespace. Provide steps to identify and resolve the problem.
Answer:
See the Solution below.
Solution:

61. Apply the overlay:
```
kubectl apply -k overlays/prod
```

62. Apply the quota:
```
oc apply -f replica-quota.yaml
```

63. Attach the PVC to a pod and validate: kubectl exec <pod-name> -- ls /mnt
Explanation:
Binding PVCs to PVs enables applications to use persistent storage dynamically and securely.

64. Validate by logging in:

```
oc login -u admin -p <complex-password>
```
Explanation:
Enforcing complex passwords improves account security, reducing the risk of unauthorized access.

65. Define the project template YAML:
```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
name: project-template-route
objects:
- apiVersion: route.openshift.io/v1 kind: Route
metadata: name: app-route
spec:
host: app.example.com to:
kind: Service
name: app-service
```

66. Create a Service to route traffic:
```
apiVersion: v1
kind: Service
metadata:
name: app-service
spec:
selector:
app: blue-green
version: v1
ports:
- protocol: TCP
port: 80
```

67. Define the project template YAML:
```
apiVersion: template.openshift.io/v1
kind: Template
metadata:
name: project-template-configmap
objects:
- apiVersion: v1 kind: ConfigMap metadata:
name: app-config data:
key: value
```

68. Attempt to exceed the limit:
```
oc create svc loadbalancer lb-svc-{1..3} --tcp=80:8080 -n dev-project
```
Explanation:
Limiting LoadBalancer services prevents overuse of external resources like cloud load balancers, controlling associated costs.

69. Bind the role to a user:
```
oc create rolebinding view-only-binding --role=view-only --user=user1 -n dev-namespace
```

70. Validate write restrictions:

oc auth can-i create pods -n dev-namespace --as=user1
Explanation:
The view role allows read-only operations, ensuring users can monitor resources without modifying or deleting them.

71. Set up an OAuth identity provider to integrate with an external authentication service. Validate user login through the external provider.
Answer:
See the Solution below.
Solution:

72. Create a new project using the template:
oc new-project test-project --template=project-template-secrets

73. Update the pod spec to mount the ConfigMap as a volume:
volumes:
- name: config-volume configMap:
name: app-config
containers:
- name: app-container image: nginx volumeMounts:
- mountPath: /etc/config
name: config-volume

74. Configure Mutual TLS (mTLS) for secure communication between two services using custom certificates. Validate mTLS.
Answer:
See the Solution below.
Solution:

75. Configure a Role-based Access Control (RBAC) setup to allow a user named dev-user to list all pods in the test-project namespace. Provide YAML definitions for the Role and RoleBinding, and demonstrate how to verify the permissions.
Answer:
See the Solution below.
Solution:

76. Apply the updated Deployment and verify:
kubectl exec -it <pod-name> -- ls /usr/share/nginx/html
Explanation:
Persistent volumes and claims abstract storage allocation in Kubernetes. By binding PVs to PVCs, applications can use persistent storage seamlessly across deployments, ensuring data persistence beyond pod lifecycles.

77. Adjust the pod’s resource requests and limits in the Deployment:
resources:
requests:
cpu: "200m"
memory: "256Mi"
limits:
cpu: "500m"

memory: "512Mi"

78. Simulate load on the application:
kubectl run load-generator --image=busybox -- /bin/sh -c "while true; do wget -q -O- http://<service-IP>; done"

79. Create a secret for an SSH key pair and use it in a pod. Validate the SSH setup.
Answer:
See the Solution below.
Solution:

80. Deploy an application across multiple namespaces using a common Deployment YAML file. Include steps to create the namespaces, apply the deployment, and verify that the pods are running in each namespace.
Answer:
See the Solution below.
Solution:

81. Attempt to drain a node:
kubectl drain <node-name> --ignore-daemonsets

82. Apply the template:
oc create -f project-with-sa.yaml

83. Create a Secret to store SSL certificates. Mount it in a pod and validate the certificates' presence.
Answer:
See the Solution below.
Solution:

84. Apply the pod YAML and validate:
kubectl apply -f pod.yaml
kubectl exec <pod-name> -- env | grep APP_MODE
Explanation:
ConfigMaps decouple configuration from code, allowing flexible and centralized management of application settings.

85. Create a PV and PVC YAML:
apiVersion: v1
kind: PersistentVolume
metadata:
name: file-pv
spec:
capacity:
storage: 10Gi
accessModes:
- ReadWriteMany hostPath:
path: "/data/file-storage"
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:

name: file-pvc
spec:
accessModes:
- ReadWriteMany resources: requests:
storage: 10Gi

86. Define a network attachment:
apiVersion: "k8s.cni.cncf.io/v1"
kind: NetworkAttachmentDefinition
metadata:
name: additional-network
spec:
config: '{
"cniVersion": "0.3.1",
"type": "macvlan",
"master": "eth0",
"ipam": {
"type": "host-local",
"subnet": "192.168.1.0/24"
}
}'

87. Deploy an application using a DeploymentConfig in OpenShift to support rollback mechanisms. Validate rollback functionality.
Answer:
See the Solution below.
Solution:

88. Configure a security context to allow a container to write to a volume using a specific GID. Validate the configuration.
Answer:
See the Solution below.
Solution:

89. Create a RoleBinding YAML file:
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
name: read-pods-binding
namespace: test-project
subjects:
- kind: User
name: dev-user
roleRef:
kind: Role
name: pod-reader
apiGroup: rbac.authorization.k8s.io

90. Deploy a pod without CPU requests:
oc run test-pod --image=nginx -n dev-project

91. Apply the ingress resource: oc apply -f ingress.yaml

Explanation:
TLS certificates provide secure communication for external traffic. Configuring ingress with custom certificates ensures encrypted communication with external users.

92. Apply the Deployment and validate:
kubectl apply -f deployment.yaml
kubectl exec <pod-name> -- printenv | grep DB_
Explanation:
Using Secrets decouples sensitive information from code, improving security and simplifying management of credentials.

93. Apply the modified backup file:
kubectl apply -f backup.yaml -n new-namespace
Explanation:
Renaming namespaces ensures compatibility when restoring configurations to a different namespace. This approach minimizes resource conflicts.

94. Apply the pod configuration: kubectl apply -f latest-image-pod.yaml

95. Configure OpenShift to automatically clean up completed Jobs after a specific time. Include steps to modify the TTL for finished Jobs and verify its functionality.
Answer:
See the Solution below.
Solution:

96. Validate resources:
oc get configmap app-config -n test-project
oc get service app-service -n test-project
Explanation:
Combining ConfigMaps and Services in templates automates default configurations and service exposure for new projects.

97. List previous revisions:
kubectl rollout history deployment/my-app

98. Create a ResourceQuota YAML file resource-quota.yaml:
```
apiVersion: v1
kind: ResourceQuota
metadata:
name: namespace-quota
namespace: resource-restricted
spec:
hard:
pods: "10"
requests.cpu: "2"
requests.memory: 2Gi
limits.cpu: "4"
limits.memory: 4Gi
```

99. Apply the Deployment:

```
kubectl apply -f deployment.yaml
```

100. Create a Secret:

```
kubectl create secret generic db-secret --from-literal=DB_CONN_STRING=jdbc:mysql://db:3306
```

101. Deploy the application: kubectl apply -f deployment.yaml

102. Create a PVC YAML file mysql-pvc.yaml:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: mysql-pvc
spec:
accessModes:
- ReadWriteOnce resources:
requests:
storage: 5Gi
```

103. Validate the restriction:

```
oc describe pod root-pod -n app-security
```

Explanation:
Custom SCCs ensure security compliance by restricting pods from running with elevated privileges.

104. Create a new role with custom permissions for managing ConfigMaps. Validate the role's functionality.
Answer:
See the Solution below.
Solution:

105. Check the pod's DNS resolution:

```
kubectl exec <pod-name> -- nslookup <service-name>
```

106. Define the CronJob YAML:

```
apiVersion: batch/v1
kind: CronJob
metadata:
name: db-backup
namespace: app-security
spec:
schedule: "0 2 * * *"
jobTemplate:
spec:
template:
spec:
containers:
- name: db-backup
image: postgres
command: ["sh", "-c", "pg_dump -U admin > /backup/db_backup.sql"] volumeMounts:
- mountPath: /backup
```

name: backup-volume
restartPolicy: OnFailure
volumes:
- name: backup-volume persistentVolumeClaim:
claimName: db-backup-pvc

107. Create a namespace for the operator: oc create namespace elasticsearch

108. Deploy an application using a Helm chart. Use the nginx chart from the stable repository and provide instructions to customize the deployment.
Answer:
See the Solution below.
Solution:

109. Deploy a PVC and pod using the StorageClass:
kubectl apply -f pvc-and-pod.yaml
kubectl exec <pod-name> -- ls /mnt
Explanation:
CSI drivers provide a standardized way to integrate external storage solutions with Kubernetes, enabling dynamic provisioning.

110. Create a custom SCC:
oc adm policy add-scc-to-group restricted non-root-scc

111. Verify the StatefulSet and pod: kubectl get statefulsets
kubectl get pods
Explanation:
StatefulSets ensure stable identities for applications requiring persistent data, like databases. Coupling them with PVs and PVCs ensures data persistence across restarts.

112. Check network policies:
oc get networkpolicy -n default

113. Validate the read-only filesystem:
oc exec readonly-pod -n app-security -- touch /root/test
Explanation:
Enabling read-only root filesystems reduces the attack surface by preventing unauthorized modifications.

114. Validate the configuration:
oc exec configmap-pod -n app-security -- cat /etc/config/key1
Explanation:
Using ConfigMaps as read-only volumes ensures the application configuration remains immutable, improving security.

115. Apply the DeploymentConfig: oc apply -f deploymentconfig.yaml

116. Apply the Deployment: kubectl apply -f deployment.yaml

117. Create a multi-container pod YAML file:

```
apiVersion: v1
kind: Pod
metadata:
name: app-with-logger
spec:
containers:
- name: app-container
image: nginx
- name: logging-sidecar
image: fluentd
args: ["-c", "/fluentd/etc/fluent.conf"]
```

118. Install the CSI driver for your environment (e.g., AWS EBS, Ceph): kubectl apply -f csi-driver.yaml

119. Expose the exporter service:

```
apiVersion: v1
kind: Service
metadata:
name: metrics-service
spec:
selector:
app: my-app
ports:
- protocol: TCP
port: 8080
```

120. Create an HPA:

```
kubectl autoscale deployment hpa-demo --cpu-percent=50 --min=1 --max=5
```

121. Validate DNS resolution:

```
kubectl exec <pod-name> -- nslookup <headless-service>
```

Explanation:
Headless Services allow direct communication between pods in a StatefulSet, essential for clustered applications.

122. Create a Secret with binary data and mount it as a file in a pod. Validate the file contents.
Answer:
See the Solution below.
Solution:

123. Create a security context constraint (SCC) to restrict pods from running as root. Validate by deploying a pod that violates the rule.
Answer:
See the Solution below.
Solution:

124. Verify PV availability and matching storage class: kubectl get pv

125. Create a ConfigMap:

```
kubectl create configmap app-config --from-literal=APP_ENV=production
```

126. Update the Deployment with a correct or accessible image: kubectl edit deployment <deployment-name>

127. Create an egress policy for the domain:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
name: allow-egress-domain
namespace: dev-namespace
spec:
podSelector:
matchLabels:
app: my-app
egress:
- to:
- dnsName: api.example.com
```

128. Expose the Deployment:

```
kubectl expose deployment hpa-demo --port=80
```

129. Define the pod YAML:

```
apiVersion: v1
kind: Pod
metadata:
name: multi-container-pod
namespace: app-security
spec:
containers:
- name: writer
image: busybox
command: ["sh", "-c", "echo 'Hello from writer' > /data/message.txt; sleep 1000"]
volumeMounts:
- name: shared-volume mountPath: /data
- name: reader
image: busybox
command: ["sh", "-c", "cat /data/message.txt; sleep 1000"] volumeMounts:
- name: shared-volume
mountPath: /data
volumes:
- name: shared-volume
emptyDir: {}
```

130. Verify both images are identical: docker images --digests | grep nginx

Explanation:

Tags are mutable and can change, but digests uniquely identify a specific image version, ensuring consistency across environments.

131. Verify the label:

```
kubectl get pods --show-labels
```
Explanation:
Kustomize simplifies managing configurations across environments by applying overlays without modifying the base configuration.

132. Audit user access logs to identify any unauthorized actions.
Answer:
See the Solution below.
Solution:

133. Create a PersistentVolume YAML file:
```
apiVersion: v1
kind: PersistentVolume
metadata:
name: shared-pv
spec:
capacity:
storage: 1Gi
accessModes:
- ReadWriteOnce hostPath:
path: /data/shared-pv
```

134. Set up OpenShift monitoring to send email alerts when CPU usage exceeds 80% for any node. Include steps to configure an alert in Prometheus.
Answer:
See the Solution below.
Solution:

135. Deploy the pod and validate:
```
oc apply -f app-pod.yaml
oc exec app-pod -n app-security -- env | grep DB
```
Explanation:
Secrets securely store sensitive information like credentials, ensuring they are not hardcoded in configurations.

136. Deploy the default ingress controller:
```
oc create -f /etc/origin/master/manifests/ingress-controller.yaml
```

137. Configure a project template to include a default Route for application exposure. Validate by creating a project and checking the route.
Answer:
See the Solution below.
Solution:

138. Implement a Horizontal Pod Autoscaler (HPA) to scale an application based on CPU usage. Validate scaling by simulating CPU load.
Answer:
See the Solution below.
Solution:

139. Create a shared volume PVC:
```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: shared-pvc
spec:
accessModes:
- ReadWriteMany
resources:
requests:
storage: 1Gi
```

140. Attach the PVC to a pod and validate:
```
kubectl apply -f pvc-and-pod.yaml
kubectl exec <pod-name> -- ls /mnt
```
Explanation:
StorageClasses automate persistent volume provisioning, reducing manual intervention for file-based storage.

141. Deploy the prod overlay: kubectl apply -k overlays/prod

142. Add a liveness probe to the DeploymentConfig:
```
spec:
containers:
- name: app-container
livenessProbe:
httpGet:
path: /
port: 80
initialDelaySeconds: 5
periodSeconds: 10
```

143. Create a custom resource for the Operator:
```
apiVersion: apps.example.com/v1
kind: MyApp
metadata:
name: my-app
spec:
replicas: 3
```

144. Investigate network-related issues: kubectl describe svc <service-name>
Explanation:
Examining logs and events identifies potential service or pod issues, such as resource limitations or misconfigurations, enabling targeted fixes.

145. Check the operator's deployment status: oc get pods -n openshift-gitops

146. Update the HTPasswd secret:
```
oc create secret generic htpasswd-secret --from-file=htpasswd=/etc/origin/htpasswd -n openshift-
config - -dry-run=client -o yaml | oc apply -f -
```

147. Apply the DeploymentConfig: oc apply -f deploymentconfig.yaml

148. Create a secret:
oc create secret generic sensitive-data --from-literal=password=mysecretpassword -n app-security

149. Create the route with no hostname (internal only): oc expose svc my-app --name=my-app-route

150. Create namespaces:
kubectl create namespace ns1
kubectl create namespace ns2

151. Export resources:
kubectl get all -n test-project -o yaml > test-project-backup.yaml

152. Edit the OAuth configuration to include the external identity provider:
```
apiVersion: config.openshift.io/v1
kind: OAuth
metadata:
name: cluster
spec:
identityProviders:
- name: external-idp
type: OpenID
mappingMethod: claim openID:
clientID: my-client-id clientSecret:
name: my-client-secret claims: preferredUsername:
- email
name:
- name email:
- email urls:
authorize: https://idp.example.com/authorize
token: https://idp.example.com/token
userInfo: https://idp.example.com/userinfo
```

153. Create a ConfigMap:
kubectl create configmap app-config --from-literal=config-key=value

154. Roll back to the previous version:
oc rollout undo dc/my-app

155. Validate write permissions:
oc exec gid-volume-pod -n app-security -- touch /data/test-file
Explanation:
Setting fsGroup ensures shared write permissions for a specific group on mounted volumes, enhancing access control.

156. Bind the role to the user:
oc create rolebinding sa-manager-binding --role=sa-manager --user=user1 -n dev-namespace

157. Create two DeploymentConfigs (blue and green):

```
oc apply -f blue-deploymentconfig.yaml
oc apply -f green-deploymentconfig.yaml
```

158. Verify the backup:

```
ETCDCTL_API=3 etcdctl snapshot status /backup/etcd-backup.db
```

159. Create a role with deployment management permissions:

```
oc create role deployment-manager-role --verb=get,list,create,delete --resource=deployments -n app-security
```

160. Install MetalLB:

```
kubectl apply -f https://raw.githubusercontent.com/metallb/metallb/v0.9.3/manifests/metallb.yaml
```

161. Troubleshoot an issue where pods are not starting due to missing NodePorts in a Service configuration.
Fix the Service and validate pod communication.
Answer:
See the Solution below.
Solution:

162. Deploy a ConfigMap with custom configuration files and mount it into a pod. Validate the file structure inside the pod.
Answer:
See the Solution below.
Solution:

163. Troubleshoot a pod stuck in the CrashLoopBackOff state. Investigate and resolve an issue where the application fails due to a missing environment variable. Provide commands to identify the problem and fix it.
Answer:
See the Solution below.
Solution:

164. Define the CronJob YAML:

```
apiVersion: batch/v1
kind: CronJob
metadata:
name: cleanup-tmp
namespace: app-security
spec:
schedule: "0 3 * * *"
jobTemplate:
spec:
template:
spec:
containers:
- name: cleaner
image: busybox
command: ["sh", "-c", "find /tmp -type f -mtime +7 -delete"]
restartPolicy: OnFailure
```

165. Verify the rolling update:

kubectl rollout status deployment/<deployment-name>
Explanation:
Annotations in the Deployment template allow OpenShift to detect changes in the ConfigMap and trigger a rolling restart of affected pods, ensuring updated configurations are applied.

166. Update the pod spec to use the ConfigMap:
env:
- name: APP_MODE valueFrom:
configMapKeyRef: name: app-config key: APP_MODE

167. Secure external and internal traffic using TLS certificates for encrypted communication.
Answer:
See the Solution below.
Solution:

168. Apply the template:
oc create -f project-template.yaml

169. Create a template YAML file app-template.yaml:
apiVersion: v1
kind: Template
metadata:
name: sample-app
objects:
- apiVersion: apps.openshift.io/v1 kind: DeploymentConfig metadata:
name: sample-app spec:
replicas: 2 selector:
app: sample-app template: metadata:
labels:
app: sample-app spec: containers:
- name: sample-container
image: nginx:latest
- apiVersion: v1
kind: Service
metadata:
name: sample-service
spec:
selector:
app: sample-app
ports:
- port: 80 targetPort: 80
- apiVersion: route.openshift.io/v1 kind: Route
metadata:
name: sample-route spec:
to:
kind: Service
name: sample-service

170. Create a StatefulSet YAML file mysql-statefulset.yaml:
apiVersion: apps/v1
kind: StatefulSet

```
metadata:
name: mysql
spec:
serviceName: "mysql-service"
replicas: 1
selector:
matchLabels:
app: mysql
template:
metadata:
labels:
app: mysql
spec:
containers:
- name: mysql image: mysql:5.7 env:
- name: MYSQL_ROOT_PASSWORD
value: rootpassword
volumeMounts:
- name: mysql-data
mountPath: /var/lib/mysql
volumeClaimTemplates:
- metadata:
name: mysql-data
spec:
accessModes: ["ReadWriteOnce"]
resources:
requests:
storage: 5Gi
```

171. Apply the quota:
```
oc apply -f secrets-quota.yaml
```

172. Create a passthrough route:
```
oc expose svc my-app --name=passthrough-route --hostname=secure.example.com --tls-termination=passthrough
```

173. Validate by logging in through the external provider: oc login --token=<external-idp-token>
Explanation:
Integrating external identity providers centralizes authentication management and supports single sign-on (SSO) capabilities.

174. Remove a user from a group and validate their removal.
Answer:
See the Solution below.
Solution:

175. Create a CronJob to run a database backup task every hour. Validate the CronJob execution.
Answer:
See the Solution below.
Solution:

176. Verify replicas: kubectl get pods
Explanation:
ReplicaSets ensure high availability by maintaining a specified number of pod replicas.

177. Validate data persistence:
curl -X GET http://<es-ip>:9200/my-index/_doc/1
Explanation:
StatefulSets ensure persistent volumes for each replica, critical for stateful distributed systems like Elasticsearch.

178. Apply the quota:
oc apply -f configmap-quota.yaml

179. Start the upgrade:
oc adm upgrade --to-latest

180. Create a namespace:
oc create namespace app-security

181. Add the Helm repository:
helm repo add stable https://charts.helm.sh/stable

182. Create a role for PVC management:
oc create role pvc-manager --verb=create,get,list --resource=persistentvolumeclaims -n dev-namespace

183. Modify the namespace in the backup file:
sed -i 's/source-namespace/target-namespace/g' backup.yaml

184. Create and manage an external route for an application in OpenShift. Validate the route is accessible externally.
Answer:
See the Solution below.
Solution:

185. Create a pod YAML file node-selector-pod.yaml:
apiVersion: v1
kind: Pod
metadata:
name: nginx-pod
spec:
containers:
- name: nginx
image: nginx:latest nodeSelector:
kubernetes.io/hostname: <node-name>

186. Validate DNS resolution:
kubectl exec <pod-name> -- nslookup headless-service
Explanation:
Headless Services enable direct communication between StatefulSet pods, which is essential for

clustered applications like databases.

187. Implement OpenShift Autoscaler to scale pods dynamically based on CPU usage. Validate scaling behavior under simulated CPU load.
Answer:
See the Solution below.
Solution:

188. Apply the Job:
kubectl apply -f exponential-retry-job.yaml

189. Apply the DeploymentConfig: oc apply -f deploymentconfig.yaml

190. Validate the readiness probe:
kubectl get pods
kubectl describe pod <pod-name>
Explanation:
Readiness probes ensure only healthy pods receive traffic, reducing downtime and user-facing errors during deployments.

191. Integrate an external LDAP server as an identity provider for authentication. Validate the LDAP login.
Answer:
See the Solution below.
Solution:

192. Configure the networking components in OpenShift by setting up DNS for internal services and external routing. Validate connectivity between internal services and external users.
Answer:
See the Solution below.
Solution:

193. Apply the updated Deployment and verify rollout completion:
kubectl apply -f deployment.yaml
kubectl rollout status deployment <deployment-name>
Explanation:
Readiness probes ensure that only functional pods receive traffic. Correcting probe configurations resolves rollout issues.

194. Update the Deployment YAML with anti-affinity rules:
```
affinity:
podAntiAffinity:
requiredDuringSchedulingIgnoredDuringExecution:
- labelSelector:
matchExpressions:
- key: app
operator: In values:
- my-app
topologyKey: "kubernetes.io/hostname"
```

195. Remove the self-provisioner role from all authenticated users:
oc adm policy remove-cluster-role-from-group self-provisioner system:authenticated

196. Validate deployment:
oc get pods -n openshift-operators
oc get csv -n openshift-operators
Explanation:
The Service Mesh Operator enables service-to-service communication with features like traffic control, observability, and security.

197. Set up a resource quota to limit the total ephemeral storage usage for all pods in a project. Validate the enforcement.
Answer:
See the Solution below.
Solution:

198. Configure a load balancer route to distribute traffic among multiple backend services. Validate load balancing behavior.
Answer:
See the Solution below.
Solution:

199. Use an ImageStream to manage different versions of an application and configure a policy for automated updates. Validate the behavior with a new image version.
Answer:
See the Solution below.
Solution:

200. Validate the SCC assignment: oc describe scc privileged
Explanation:
Privileged SCCs are required for applications needing access to host-level resources, such as network interfaces or devices.

201. Reference the Secret in the Deployment YAML:
env:
- name: DB_USER valueFrom:
secretKeyRef: name: db-secret key: username
- name: DB_PASS valueFrom:
secretKeyRef: name: db-secret key: password
Explanation:
Secrets securely store sensitive data like credentials. Referencing them in Deployments ensures that applications receive the data without exposing it in configurations.

202. Set up a DeploymentConfig to include a liveness probe and validate that failed containers are restarted automatically.
Answer:
See the Solution below.
Solution:

203. Deploy the stable version:

```
kubectl apply -f stable-deployment.yaml
```

204. Scale the backend service:

```
oc scale deployment my-app --replicas=3
```

205. Validate group membership: oc get groups developers -o yaml

Explanation:

Groups simplify user management by enabling bulk operations and role assignments for multiple users.

206. Create a quota YAML file:

```
apiVersion: v1
kind: ResourceQuota
metadata:
name:  configmap-quota
namespace:  dev-project
spec:
hard:
configmaps: "3"
```

207. Validate routing behavior:

```
curl  http://example.com/service1
curl  http://example.com/service2
```

Explanation:

Path-based routing allows directing traffic to different services based on URL paths, improving flexibility in handling requests.

208. Create a custom health check for an application and configure a network policy to block traffic to unhealthy pods.

Answer:

See the Solution below.

Solution:

209. Install the Prometheus Operator using the OpenShift CLI. Validate the operator's status.

Answer:

See the Solution below.

Solution:

210. Deploy the StatefulSet with volume templates:

```
volumeClaimTemplates:
- metadata:
name: pg-data
spec:
accessModes: ["ReadWriteOnce"]
resources:
requests:
storage: 5Gi
```

211. Validate the deployment:
oc get pods -n app-security
Explanation:
Docker registry secrets enable secure authentication to private image repositories, ensuring only authorized access.

212. Apply the PDB: kubectl apply -f pdb.yaml

213. View the pod logs:
kubectl logs <pod-name>

214. Verify the metrics endpoint:
curl http://<service-IP>:8080/metrics
Explanation:
Custom metrics enhance application monitoring by exposing key performance indicators, integrating seamlessly with Prometheus.

215. Create the service account:
oc create serviceaccount configmap-sa -n app-security

216. Update the Deployment YAML to include a readiness probe:
```
readinessProbe:
httpGet:
path: /
port: 80
initialDelaySeconds: 5
periodSeconds: 10
```

217. Apply the deployment in each namespace:
```
kubectl apply -f app-deployment.yaml -n ns1
kubectl apply -f app-deployment.yaml -n ns2
```

218. Create a Job YAML file batch-job.yaml:
```
apiVersion: batch/v1
kind: Job
metadata:
name: simple-job
spec:
template:
spec:
containers:
- name: batch-container image: busybox
command: ["sh", "-c", "echo Hello, Kubernetes! && sleep 30"]
restartPolicy: Never
```

219. Simulate a readiness failure:
kubectl exec <pod-name> -- mv /usr/share/nginx/html/index.html /tmp/

220. Validate job execution: oc get jobs -n app-security
Explanation:

CronJobs schedule periodic tasks, such as backups or cleanups, ensuring automation of repetitive operations.

221. Attempt to exceed the limit:

```
oc create svc clusterip svc-{1..4} --tcp=8080 -n dev-project
```

Explanation:
Quotas on services ensure that namespaces do not exceed their allocation, maintaining cluster balance.

222. Define a liveness probe in the pod spec:

```
livenessProbe:
httpGet:
path: /healthz
port: 8080
initialDelaySeconds: 3
periodSeconds: 5
```

223. Deploy a pod using the default service account:

```
apiVersion: v1
kind: Pod
metadata:
name: default-sa-pod
namespace: app-security
spec:
containers:
- name: nginx
image: nginx
```

224. Deploy an application using an OpenShift template that includes a DeploymentConfig, Service, and Route. Validate that the application is accessible via the Route.
Answer:
See the Solution below.
Solution:

225. Create a CPU quota YAML file:

```
apiVersion: v1
kind: ResourceQuota
metadata:
name: cpu-quota
namespace: dev-project
spec:
hard:
requests.cpu: "4"
limits.cpu: "8"
```

226. Ensure the service uses port 443:

```
spec:
ports:
- name: https
port: 443
```

targetPort: 443

227. Retrieve the NodePort: kubectl get svc

228. Create a service account:
oc create serviceaccount readonly-sa -n app-security

229. List all users and groups in the cluster.
Answer:
See the Solution below.
Solution:

230. Use OpenShift to deploy a blue-green application update strategy to minimize downtime during a version upgrade. Include steps to configure and validate the deployment.
Answer:
See the Solution below.
Solution:

231. Expose the Deployment as a ClusterIP service:
kubectl expose deployment my-app --port=80 --target-port=80

232. Update the DeploymentConfig to include a ConfigMap volume:
spec:
triggers:
- type: ConfigChange