1. Verify scaling behavior:
kubectl get hpa
kubectl get pods
Explanation:
Autoscaling ensures that applications adapt dynamically to workload demands, improving resource utilization and maintaining performance.

2. Configure a Horizontal Pod Autoscaler (HPA) for an application that scales based on custom metrics using Prometheus. Validate the custom metrics scaling behavior.
Answer:
See the Solution below.
Solution:

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

4. Apply the configuration:
oc apply -f oauth-config.yaml

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

6. Restrict project creation to specific users. Validate the restricted behavior for other users.
Answer:
See the Solution below.
Solution:

7. Migrate an application from one namespace to another without downtime. Include commands to copy configurations, secrets, and deployments.
Answer:
See the Solution below.
Solution:

8. Deploy the pod: kubectl apply -f pod.yaml

9. Validate the resource limits: kubectl describe pod <pod-name>
Explanation:
Customizing Helm values allows precise control over resource allocation during deployment.

10. Apply the pod:
oc apply -f writable-pod.yaml

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

12. Update the Service to route traffic to both deployments:
apiVersion: v1
kind: Service
spec:
selector:
app: my-app
trafficPolicy:
- stable-deployment: 80%
- canary-deployment: 20%

13. Deploy a pod without limits:
oc run test-pod --image=nginx -n dev-project

14. Replace the namespace in the exported file:
sed -i 's/source-namespace/target-namespace/g' resources.yaml

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

16. Validate their access:
oc auth can-i get pods/log -n dev-namespace --as=user1 oc auth can-i delete pods -n dev-namespace --as=user1
Explanation:
Custom roles tailored to specific operations, such as viewing logs, enhance security by minimizing unnecessary permissions.

17. Update the pod spec to mount the Secret:
volumes:
- name: binary-volume secret:
secretName: binary-secret
volumeMounts:
- mountPath: /etc/binary name: binary-volume

18. Create a DeploymentConfig that integrates with a ConfigMap and updates the deployment dynamically when the ConfigMap changes. Validate the update behavior.
Answer:
See the Solution below.
Solution:

19. Define the limit range YAML:
apiVersion: v1
kind: LimitRange
metadata:
name: storage-limits
namespace: dev-project
spec:
limits:
- type: PersistentVolumeClaim default:
storage: "10Gi"

20. Verify retry attempts: kubectl describe job retry-job
Explanation:
Exponential backoff reduces load on the cluster by increasing retry intervals, preventing excessive resource consumption during failures.

21. Resolve by updating or removing problematic network policies.
Explanation:
Pod network issues can stem from DNS, service misconfigurations, or restrictive network policies. Systematic diagnosis ensures efficient resolution.

22. Create a ConfigMap:
oc create configmap app-config --from-literal=APP_ENV=production

23. Create a project using the template:
oc new-project test-project --template=project-template-configmap-service

24. Filter logs for the specific user:
grep "user=<username>" /var/log/audit.log

Explanation:
Audit logging tracks administrative actions, ensuring accountability and compliance with organizational policies.

25. Install Metrics Server:
kubectl apply -f
https://github.com/kubernetes-sigs/metrics-server/releases/latest/download/components.yaml

26. Create a project template to automatically include a ConfigMap and a Service when a project is created.
Validate the creation of resources.
Answer:
See the Solution below.
Solution:

27. Scale the ReplicaSet:
kubectl scale replicaset my-app-rs --replicas=5

28. Validate enforcement:
Deploy resources exceeding the quota and check violations:
oc describe quota cluster-quota
Explanation:
Cluster resource quotas ensure fair resource usage across projects, preventing any single project from exhausting cluster resources.

29. Add a liveness probe with a custom script:
livenessProbe:
exec:
command:
- sh
- -c
- "curl -f http://localhost/health || exit 1" initialDelaySeconds: 5 periodSeconds: 10

30. Remove the user from the HTPasswd file: htpasswd -D /etc/origin/htpasswd newuser

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

32. Assign the cluster-admin role:
oc adm policy add-cluster-role-to-user cluster-admin -z admin-sa -n app-security

33. Configure an OpenShift Service to route traffic to multiple backends based on the request path. Include steps to create the Service and validate traffic routing.
Answer:
See the Solution below.
Solution:

34. Verify the quota:

kubectl describe quota project-quota -n custom-project
Explanation:
Resource quotas prevent overconsumption of cluster resources, ensuring fair allocation and stability across multiple projects.

35. Apply the BuildConfig:
oc apply -f buildconfig.yaml

36. Use the ConfigMap in a pod:
apiVersion: v1
kind: Pod
metadata:
name: configmap-pod
namespace: app-security
spec:
containers:
- name: busybox
image: busybox volumeMounts:
- mountPath: /etc/config name: config-volume
readOnly: true
volumes:
- name: config-volume configMap:
name: app-config

37. Configure a StatefulSet to deploy a MySQL database with persistent storage. Include steps to define the StatefulSet, create a PersistentVolume (PV) and PersistentVolumeClaim (PVC), and verify the database is running correctly.
Answer:
See the Solution below.
Solution:

38. Create a new project using the template:
oc new-app --template=project-template -n test-project
Explanation:
Project templates automate the creation of default resources, simplifying developer workflows and ensuring consistency.

39. Deploy an application using Kustomize to overlay environment-specific configurations, including labels and resource limits. Validate the deployment.
Answer:
See the Solution below.
Solution:

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

41. Assign self-provisioner to a specific user:
oc adm policy add-cluster-role-to-user self-provisioner user1

42. Add the service account to the privileged SCC:
oc adm policy add-scc-to-user privileged -z custom-sa -n app-security

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

44. Create a PVC without storage size: oc create pvc test-pvc -n dev-project

45. Validate the failure:
oc describe is nginx-stream

46. Create a global deny-all egress policy for a namespace and validate the behavior.
Answer:
See the Solution below.
Solution:

47. Create a Secret to store the database credentials:
kubectl create secret generic db-credentials --from-literal=username=admin --from-literal=password=securepass

48. 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.

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

50. Create a limit range YAML file:
apiVersion: v1
kind: LimitRange
metadata:
name: cpu-memory-limits
namespace: dev-project
spec:
limits:
- type: Pod default: cpu: "500m" memory: "512Mi" max:
cpu: "2"
memory: "2Gi"

51. Write data to Redis: redis-cli set key value

52. Verify that the route exists: oc get routes

53. Tag a new image version:
oc tag nginx:1.22 nginx-stream:stable

54. Use OpenShift CLI to configure and test a Service Account with limited permissions. Create a Role for accessing ConfigMaps, bind it to the Service Account, and validate the access.
Answer:
See the Solution below.
Solution:

55. Provision a PersistentVolume (PV) and PersistentVolumeClaim (PVC) for block-based storage. Mount the PVC to a pod and validate access to the storage.
Answer:
See the Solution below.
Solution:

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

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

58. Define the CronJob YAML:
apiVersion: batch/v1
kind: CronJob
metadata:
name: log-cleanup
namespace: app-security
spec:
schedule: "0 0 * * *"
jobTemplate:
spec:
template:
spec:
containers:
- name: cleanup
image: busybox
command: ["sh", "-c", "rm -rf /var/logs/*"]
restartPolicy: OnFailure

59. Validate HPA behavior: kubectl get hpa
Explanation:
Metrics Server provides real-time resource usage data, enabling HPA to scale applications based on actual workload demands.

60. Monitor the deployment: oc get pods

61. Verify pod placement:
kubectl get pods -o wide
Explanation:
Taints and tolerations provide fine-grained scheduling control, ensuring critical workloads are assigned to reserved nodes.

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

63. Create a PersistentVolume YAML file:
apiVersion: v1
kind: PersistentVolume

metadata:
name: shared-pv
spec:
capacity:
storage: 1Gi
accessModes:
- ReadWriteOnce hostPath:
path: /data/shared-pv

64. Configure a readiness probe for an application to ensure that traffic is sent only to pods that are fully ready. Include steps to create the probe in the Deployment YAML and test its functionality.
Answer:
See the Solution below.
Solution:

65. Simulate an ImageStream update failure due to an unavailable remote registry. Validate the failure and recovery process.
Answer:
See the Solution below.
Solution:

66. Configure an ingress resource with wildcard hostname support to handle multiple subdomains. Validate traffic routing.
Answer:
See the Solution below.
Solution:

67. Deploy the StatefulSet:
kubectl apply -f elasticsearch-statefulset.yaml

68. Update the pod definition with tolerations:
tolerations:
- key: "key" operator: "Equal" value: "value" effect: "NoSchedule"

69. Create a Secret:
kubectl create secret generic api-keys --from-literal=API_KEY=abc123

70. Attempt to exceed the quota:
oc create deployment deploy-{1..6} --image=nginx -n dev-project
Explanation:
Deployment quotas prevent a namespace from creating excessive deployment resources, ensuring balanced cluster resource usage.

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

72. Use Kubernetes Metrics Server to monitor resource usage of pods and validate that HPA reacts to increased CPU usage.
Answer:
See the Solution below.
Solution:

73. Deploy the stable version:
apiVersion: apps/v1
kind: Deployment
metadata:
name: app-stable
spec:
replicas: 3
selector:
matchLabels:
app: my-app
version: stable
template:
metadata:
labels:
app: my-app
version: stable
spec:
containers:
- name: app-container image: my-app: v1

74. Create a security context to allow a pod to write to a mounted volume. Validate the configuration.
Answer:
See the Solution below.
Solution:

75. Create a Service Account:
kubectl create serviceaccount limited-access

76. Validate the installation:
oc get csv -n cert-manager
Explanation:
Installing operators via the OperatorHub allows administrators to extend cluster functionality easily. Validating the ClusterServiceVersion (CSV) ensures that the operator is ready for use.

77. Verify the pod restart: kubectl get pods
Explanation:
CrashLoopBackOff indicates repeated failures. Investigating logs and descriptions often reveals misconfigurations. Updating the Deployment resolves the issue without redeploying the entire application.

78. Implement an ImageStream policy that disables automatic image updates. Validate the manual update process.
Answer:
See the Solution below.
Solution:

79. Apply the Service and verify DNS resolution:
kubectl apply -f headless-service.yaml
kubectl exec <pod-name> -- nslookup headless-service
Explanation:

Headless Services provide direct access to individual pod IPs, which is essential for StatefulSet workloads.

80. Check the rollout status:
kubectl rollout status deployment <deployment-name>

81. Create a Role YAML file configmap-role.yaml:
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
namespace: default
name: configmap-reader
rules:
- apiGroups: [""]
resources: ["configmaps"]
verbs: ["get", "list"]

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

83. Apply the CronJob: kubectl apply -f cronjob.yaml

84. Create a role:
oc create role configmap-manager --verb=get,create,update,delete --resource=configmaps -n dev-namespace

85. 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

86. Troubleshoot and resolve an issue where an ingress rule is not directing traffic to the correct backend service. Include validation steps after fixing the issue.
Answer:
See the Solution below.
Solution:

87. Enable audit logging in the cluster: oc edit apiserver

88. Create a Role YAML file:
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
namespace: test-project
name: pod-reader
rules:
- apiGroups: [""]
resources: ["pods"]
verbs: ["get", "list"]

89. Configure the route to use the TLS secret:
oc expose svc my-app --name=my-app-route --hostname=example.com --tls-secret=my-tls-secret

90. 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.

91. Trigger a deployment with a broken image: oc tag nginx:broken nginx-stream:stable

92. Deploy a StatefulSet for a MySQL database with non-shared storage. Validate data persistence after restarting pods.
Answer:
See the Solution below.
Solution:

93. 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.

94. Validate the restriction:
oc describe pod exceeding-pod -n app-security
Explanation:
LimitRange ensures pods adhere to resource usage policies, preventing overcommitment of cluster resources.

95. Audit and review the cluster's role bindings to identify unnecessary permissions. Remove any unnecessary bindings.
Answer:
See the Solution below.
Solution:

96. Apply the updated pod configuration: kubectl apply -f pod.yaml

97. Create a Secret to store a database connection string. Inject it as an environment variable into a pod.
Validate that the application can use it.
Answer:
See the Solution below.
Solution:

98. Validate the route:
oc get route app-route -n test-project
Explanation:
Including default Routes in templates automates external access setup for applications in new projects.

99. Create a CronJob to clean up old logs in a specific directory every day. Validate the CronJob.
Answer:
See the Solution below.
Solution:

100. Use an ImageStream to switch traffic between two application versions using traffic splitting. Validate the behavior.
Answer:
See the Solution below.
Solution:

101. 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:

102. Cordon the node to prevent new pods: kubectl cordon <node-name>

103. Validate automatic updates:
oc get is app-stream
Explanation:
Automatic tracking via ImageStreams ensures that the latest image from a registry is consistently available for deployments.

104. Validate email logs:
oc logs <pod-name> -n app-security
Explanation:
Automating email notifications with CronJobs enhances operational efficiency by providing timely updates.

105. Test with a pod running as root:

apiVersion: v1
kind: Pod
metadata:
name: root-pod
namespace: app-security
spec:
containers:
- name: nginx
image: nginx securityContext: runAsUser: 0

106. Create a PeerAuthentication policy:
apiVersion: security.istio.io/v1beta1
kind: PeerAuthentication
metadata:
name: default
namespace: dev-namespace
spec:
mtls:
mode: STRICT

107. List groups: oc get groups
Explanation:
Listing users and groups provides an overview of cluster access, aiding in auditing and management.

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

109. Validate execution:
oc get jobs -n app-security
Explanation:
CronJobs automate repetitive tasks like log compression, reducing manual intervention and improving efficiency.

110. Create a ResourceQuota:
apiVersion: v1
kind: ResourceQuota
metadata:
name: build-quota
spec:
hard:
requests.cpu: "2"
requests.memory: "2Gi"

111. Create a Secret to store Docker credentials for pulling private images. Validate the pod uses the Secret successfully.

Answer:
See the Solution below.
Solution:

112. Simulate a failure by breaking the endpoint:
kubectl exec <pod-name> -- mv /usr/share/nginx/html/index.html /tmp/

113. Update the pods to use the certificates:
volumeMounts:
- mountPath: /etc/tls name: tls-secret readOnly: true

114. List users: oc get users

115. Configure Horizontal Pod Autoscaling (HPA) based on CPU usage. Validate the autoscaler by generating load.
Answer:
See the Solution below.
Solution:

116. Apply the Ingress resource: kubectl apply -f ingress.yaml

117. Ensure the service uses port 443:
spec:
ports:
- name: https
port: 443
targetPort: 443

118. Tag a new image:
oc tag nginx:1.22 nginx-stream:stable

119. Verify cleanup:
oc get pods -n knative-serving
oc get csv -n knative-serving
Explanation:
Uninstalling operators involves a systematic cleanup to ensure no residual resources remain, maintaining cluster efficiency.

120. Control cluster network ingress by configuring an ingress controller. Validate traffic flow to services.
Answer:
See the Solution below.
Solution:

121. 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:

122. Validate interaction:
oc logs multi-container-pod -c reader -n app-security
Explanation:
Shared volumes enable communication between containers within the same pod, fostering modular application designs.

123. Test the application by exposing the Deployment:
kubectl expose deployment nginx-deployment --type=NodePort --port=80 kubectl get svc

124. 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.

125. Update the HTPasswd secret:
oc create secret generic htpasswd-secret --from-file=htpasswd=/etc/origin/htpasswd -n openshift-config

126. Deploy the StatefulSet with volume templates:
volumeClaimTemplates:
- metadata:
name: pg-data
spec:
accessModes: ["ReadWriteOnce"]
resources:
requests:
storage: 5Gi

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

128. Define the project template YAML:
apiVersion: template.openshift.io/v1
kind: Template
metadata:
name: project-template-configmap-service
objects:
- apiVersion: v1 kind: ConfigMap metadata:
name: app-config data:
key: value
- apiVersion: v1 kind: Service metadata:
name: app-service spec:
selector: app: my-app ports:
- protocol: TCP port: 80 targetPort: 8080

129. Attach the PVC to multiple pods and validate shared access:
kubectl apply -f pvc-and-pods.yaml
kubectl exec <pod-name-1> -- touch /mnt/test
kubectl exec <pod-name-2> -- ls /mnt

Explanation:
NFS provides shared storage across pods, ideal for collaborative applications like content management systems.

130. Monitor the rollout status:
kubectl rollout status deployment <deployment-name>

131. Validate load balancing:
for i in {1..5}; do curl http://loadbalancer.example.com; done
Explanation:
Load balancer routes distribute incoming traffic across multiple backend instances, ensuring better availability and performance.

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

133. Install the Knative Serving Operator to manage serverless applications. Validate its installation.
Answer:
See the Solution below.
Solution:

134. Create a secret to store sensitive application credentials and validate its usage in a pod.
Answer:
See the Solution below.
Solution:

135. Create a service account for a specific namespace and validate its creation.
Answer:
See the Solution below.
Solution:

136. Create a ReplicaSet YAML file:
apiVersion: apps/v1
kind: ReplicaSet
metadata:
name: nginx-replicaset
spec:
replicas: 3
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: nginx image: nginx:latest ports:
- containerPort: 80

137. 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.

138. Apply the deployment:
oc apply -f db-app.yaml

139. Assign a role with API access:
oc create role pod-reader --verb=get,list --resource=pods -n app-security
oc create rolebinding pod-reader-binding --role=pod-reader --serviceaccount=app-security:api-access-sa -n app-security

140. Add a readiness probe to the DeploymentConfig:
spec:
containers:
- name: app-container readinessProbe: httpGet:
path: /
port: 80

141. Deploy an application that dynamically provisions storage using a CSI driver. Validate the storage binding.
Answer:
See the Solution below.
Solution:

142. Roll back to the previous version: kubectl rollout undo deployment/my-app

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

144. Deploy a LoadBalancer service:
spec:
type: LoadBalancer

145. Configure and validate a cluster-wide limit range to enforce CPU and memory limits for all newly created pods. Include the YAML definition and demonstrate how to test the limit enforcement.
Answer:
See the Solution below.
Solution:

146. Enable access logs by updating ingress annotations:
oc annotate ingresscontroller default --overwrite --type=merge haproxy.router.openshift.io/log-format='%ci:%cp [%t] %ft %b/%s %Tt %B %Tw %Tr %ST %U %r'

147. Apply the CronJob:
oc apply -f db-backup.yaml

148. Apply the pod configuration:
oc apply -f raw-socket-pod.yaml

149. Validate by exceeding memory requests:
oc run test-pod --image=nginx --requests='memory=3Gi'
Explanation:
Memory-specific quotas prevent resource exhaustion by restricting memory allocation for pods within a namespace.

150. Configure a service mesh with Istio to encrypt all traffic between services using mTLS. Validate secure communication.
Answer:
See the Solution below.
Solution:

151. Remove unused users and groups:
oc delete user <user-name>
oc delete group <group-name>

152. Set up a quota template to enforce resource limits for all projects created by a specific group. Validate by creating a project under the group.
Answer:
See the Solution below.
Solution:

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

154. Create a base directory base/ with deployment.yaml:
apiVersion: apps/v1
kind: Deployment
metadata:
name: nginx-deployment
spec:
replicas: 3
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
spec:
containers:
- name: nginx image: nginx:latest

155. Create a project template to include a default service account with additional permissions. Validate by creating a project.
Answer:
See the Solution below.
Solution:

156. Apply the Job:
kubectl apply -f migration-job.yaml

157. 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:

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

159. Create a PVC YAML file using the StorageClass:
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: dynamic-pvc
spec:
accessModes:
- ReadWriteOnce resources:
requests: storage: 5Gi
storageClassName: dynamic-storage

160. Create a ConfigMap and Secret:
kubectl create configmap app-config --from-literal=APP_MODE=production kubectl create secret generic app-secret --from-literal=APP_KEY=12345

161. Update the image in the Deployment:
kubectl set image deployment/<deployment-name> <container-name>=<new-image> -n <namespace>

162. Check the Ingress configuration: kubectl describe ingress <ingress-name>

163. Create an ImageStream tracking a remote registry:
spec:
tags:
- name: stable from:
kind: DockerImage
name: registry.example.com/nginx:stable

164. Validate restricted access:
oc auth can-i delete pods -n dev-namespace --as=user1
Explanation:
Custom roles provide granular control over user permissions, allowing precise enforcement of security policies.

165. Apply the CronJob:
oc apply -f email-notifier.yaml

166. Generate traffic to validate logs: curl http://example.com
Explanation:
Access logs provide detailed insights into incoming traffic, essential for monitoring, debugging, and compliance requirements.

167. Push a change to the repository and validate: oc get builds
Explanation:
BuildConfigs with Git triggers automate builds upon code changes, enabling seamless CI/CD workflows.

168. Create a BuildConfig:
apiVersion: build.openshift.io/v1
kind: BuildConfig
metadata:
name: app-build
spec:
source:
git:
uri: "https://github.com/example/repo.git"
strategy:
type: Docker
triggers:
- type: GitHub github:
secret: webhook-secret

169. Apply the quota:
oc apply -f memory-quota.yaml

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

171. Insert data into the cluster:
cqlsh <cassandra-pod-ip> -e "INSERT INTO keyspace.table (id, value) VALUES (1, 'data');"

172. Apply the updated Deployment: kubectl apply -f app-deployment.yaml

173. Apply the route:
oc apply -f path-based-route.yaml

174. Create a BuildConfig for the deployment stage:
spec:
input:
from:
kind: ImageStreamTag
name: app-base:latest

175. Create a StatefulSet YAML:
volumeClaimTemplates:
- metadata:
name: elasticsearch-data
spec:
accessModes: ["ReadWriteOnce"]
resources:
requests:

storage: 10Gi

176. Add users to the group:
oc adm groups add-users developers user1 user2

177. Create a route with re-encrypt termination:
oc expose svc my-app --name=re-encrypt-route --hostname=secure.example.com --tls-termination=reencrypt --dest-ca-cert=backend-ca.pem

178. 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

179. Define the pod YAML:
apiVersion: v1
kind: Pod
metadata:
name: init-container-pod
namespace: app-security
spec:
containers:
- name: main-container
image: busybox
command: ["sh", "-c", "cat /data/config.txt"]
volumeMounts:
- mountPath: /data name: shared-volume
initContainers:
- name: init-container
image: busybox
command: ["sh", "-c", "echo Pre-configured > /data/config.txt"]
volumeMounts:
- mountPath: /data name: shared-volume
volumes:
- name: shared-volume
emptyDir: {}

180. Use a TLS secret in a Kubernetes ingress resource. Validate secure communication.
Answer:
See the Solution below.
Solution:

181. Restart the API server and analyze logs:

tail -n 100 /var/log/audit.log | grep <namespace>
Explanation:
Audit logs provide a comprehensive record of cluster activity, enabling administrators to trace and analyze changes for security and compliance purposes.

182. Use the template for a new project:
oc new-project test-project --template=default-project-template

183. Add a rolling update strategy to the DeploymentConfig:
spec:
strategy:
type: Rolling
rollingParams:
maxUnavailable: 1
maxSurge: 2

184. Create a Secret:
kubectl create secret generic db-secret --from-literal=DB_CONN_STRING=jdbc:mysql://db:3306

185. Configure a role that allows a group to manage ConfigMaps and Secrets in a namespace. Validate their permissions.
Answer:
See the Solution below.
Solution:

186. Add a taint to a node:
kubectl taint nodes <node-name> key=value:NoSchedule

187. Configure a pod with read-only root filesystem for added security. Validate the configuration.
Answer:
See the Solution below.
Solution:

188. Verify the pod environment variables: kubectl exec <pod-name> -- env | grep DB
Explanation:
Secrets protect sensitive information, like credentials, by decoupling it from application code and configuration files, enhancing security and maintainability.

189. Validate the pipeline output: oc get pods
Explanation:
Multi-stage pipelines using BuildConfigs and ImageStreams enable modular and efficient application build and deployment workflows.

190. Create a CPU quota YAML file:
apiVersion: v1
kind: ResourceQuota
metadata:
name: cpu-quota
namespace: dev-project

spec:
hard:
limits.cpu: "16"

191. Trigger a pod termination and validate:
kubectl delete pod <pod-name>
kubectl get events
Explanation:
Grace periods allow applications to complete ongoing requests or cleanup tasks before termination, ensuring a smooth shutdown process.

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

193. Define a Service with multiple backends:
apiVersion: v1
kind: Service
metadata:
name: multi-backend-service
spec:
selector:
app: backend
ports:
- protocol: TCP
port: 80
targetPort: 8080

194. Remove the CSV:
oc delete csv <csv-name> -n elasticsearch

195. Test external restriction:
curl http://<cluster-external-ip>:80
Explanation:
Internal-only services improve security by restricting external access, suitable for backend services that don't require public exposure.

196. Create a ClusterIP service:
kubectl expose deployment nginx-deployment --type=ClusterIP --port=80

197. List all role bindings in a namespace: oc get rolebindings -n dev-namespace

198. Simulate a failed ImageStream-based deployment and debug the issue using logs.
Answer:
See the Solution below.
Solution:

199. Deploy a broken application version:

oc set image dc/my-app app-container=nginx:broken

200. Add annotations to the route to configure rate limiting:
apiVersion: route.openshift.io/v1
kind: Route
metadata:
name: rate-limit-route
annotations:
haproxy.router.openshift.io/rate-limit-connections: "10"
haproxy.router.openshift.io/rate-limit-connections-burst: "5"
spec:
host: ratelimit.example.com
to:
kind: Service
name: my-app

201. Apply the configuration and validate:
kubectl apply -f pod.yaml
kubectl exec <pod-name> -- ls /etc/config
Explanation:
ConfigMaps allow applications to read configuration data from files or environment variables, making deployments more flexible.

202. 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

203. Configure the DNS for internal services:
Edit the dns.config in the OpenShift configuration.
apiVersion: operator.openshift.io/v1
kind: DNS
metadata:
name: cluster
spec:
servers:
- 172.30.0.10

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

205. Apply the template:
oc create -f project-template-route.yaml

206. Add a pre-hook to the DeploymentConfig:
spec:
strategy:
type: Rolling
rollingParams:
pre:
execNewPod:
containerName: app-container
command:
- sh
- -c
- "curl http://localhost/validate"

207. Define the limit range YAML file:
apiVersion: v1
kind: LimitRange
metadata:
name: cpu-limit
namespace: dev-project
spec:
limits:
- type: Container defaultRequest: cpu: "500m"

208. Create a PVC using the StorageClass:
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
name: dynamic-pvc
spec:
storageClassName: dynamic-storage-class
accessModes:
- ReadWriteOnce resources:
requests: storage: 10Gi

209. Restart a pod and validate data persistence:
kubectl delete pod cassandra-0
cqlsh <cassandra-pod-ip> -e "SELECT * FROM keyspace.table;"
Explanation:
StatefulSets ensure each Cassandra node has its own persistent storage, allowing data retention even if pods are restarted or moved.

210. Validate by creating a secret:
oc exec secret-manager-pod -n app-security -- oc create secret generic test-secret --from-literal=key=value
Explanation:
Granting granular permissions to service accounts ensures secure management of critical resources like secrets.

211. Access the OpenShift documentation: oc adm upgrade

212. Set up a DeploymentConfig with both manual and image-based triggers. Validate both trigger mechanisms.
Answer:
See the Solution below.
Solution:

213. Create a StorageClass YAML file:
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
name: file-storage-class
provisioner: kubernetes.io/aws-ebs
parameters:
type: gp2

214. Verify the Route and test external access:
o c get routes
curl https://<route-hostname>
Explanation:
Custom annotations on Routes configure advanced features like SSL termination, enhancing security for external traffic.

215. Configure a Deployment to use a custom health check script in a liveness probe. Validate pod behavior.
Answer:
See the Solution below.
Solution:

216. Verify the route has been created: oc get routes

217. Troubleshoot an ingress controller that is not processing external traffic.
Answer:
See the Solution below.
Solution:

218. Implement a canary deployment for an application to test a new version with a subset of users. Validate traffic splitting.
Answer:
See the Solution below.
Solution:

219. Delete the subscription:
oc delete subscription knative-serving -n knative-serving

220. Provision block-based storage with a specific StorageClass. Validate the PVC is dynamically provisioned and attached to a pod.
Answer:
See the Solution below.

Solution:

221. Modify a user’s password and validate the update.
Answer:
See the Solution below.
Solution:

222. Create a route to distribute traffic:
oc expose svc my-app --hostname=loadbalancer.example.com --name=loadbalancer-route

223. Create a StorageClass that uses the CSI driver:
provisioner: ebs.csi.aws.com

224. Apply the CronJob:
oc apply -f db-backup.yaml

225. Validate that a minimum of 2 pods remain running: kubectl get pods
Explanation:
PDBs ensure a minimum number of pods remain available during disruptions, enhancing application availability during maintenance.

226. Back up etcd data from the OpenShift cluster control plane and explain how to restore it in case of a
failure.
Answer:
See the Solution below.
Solution:

227. Create a group and add users:
oc adm groups new namespace-admins
oc adm groups add-users namespace-admins user1 user2

228. Deploy a ConfigMap and inject its data as environment variables into a pod. Validate that the pod uses the environment variables.
Answer:
See the Solution below.
Solution:

229. Deploy a broken image:
kubectl set image deployment/my-app app-container=nginx:broken

230. Create a PVC and attach it to the pod:
volumes:
- name: nfs-volume
persistentVolumeClaim:
claimName: nfs-pvc
volumeMounts:
- mountPath: /mnt/nfs name: nfs-volume

231. Add a taint to the node:
kubectl taint nodes <node-name> reserved=true:NoSchedule

232. Restart Prometheus:
kubectl rollout restart deployment prometheus -n openshift-monitoring