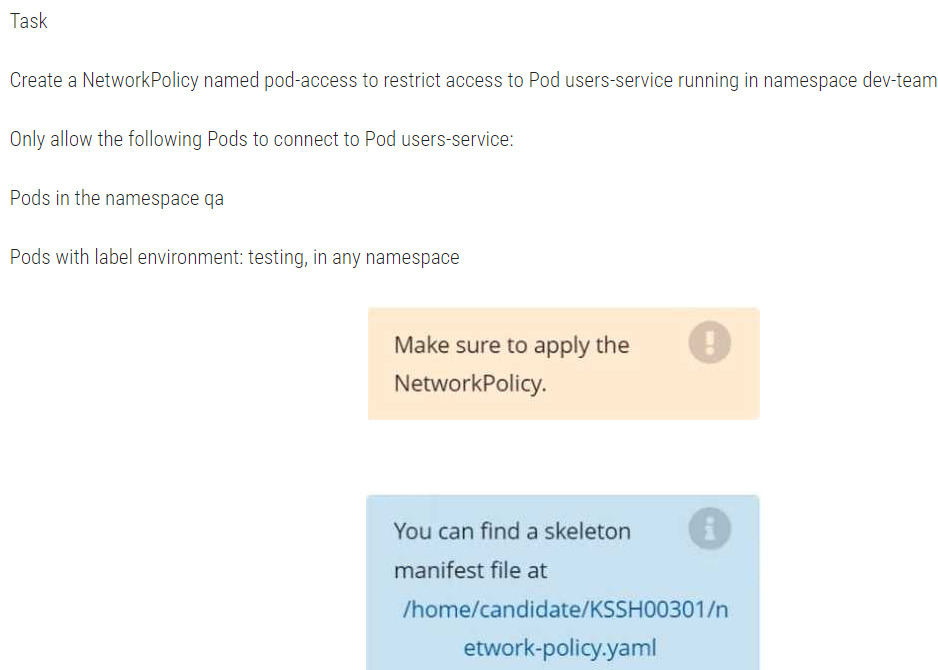
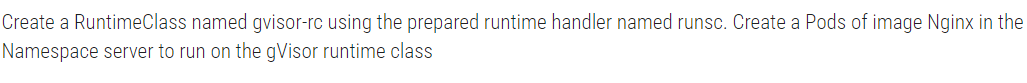
Q-

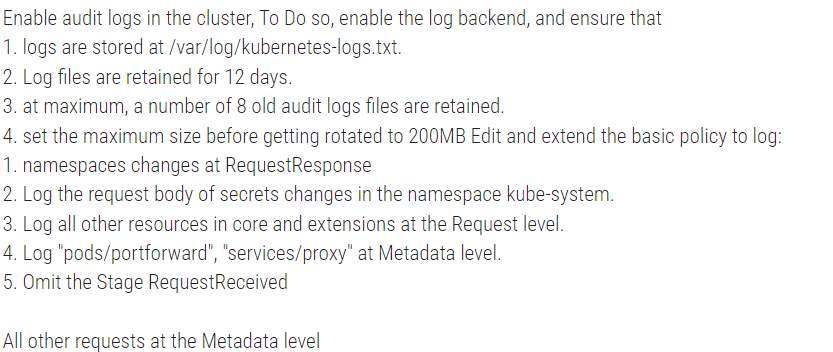


Q-

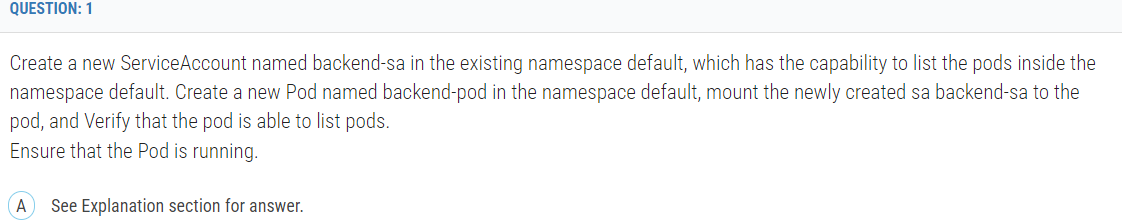


Create a RuntimeClass named untrusted using the prepared runtime handler named runsc. Create a Pods of image alpine:3.13.2 in the Namespace default to run on the gVisor runtime class.

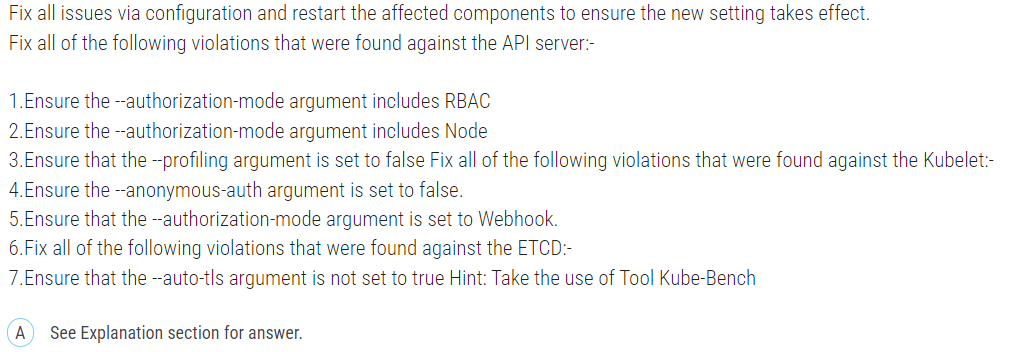
Q-



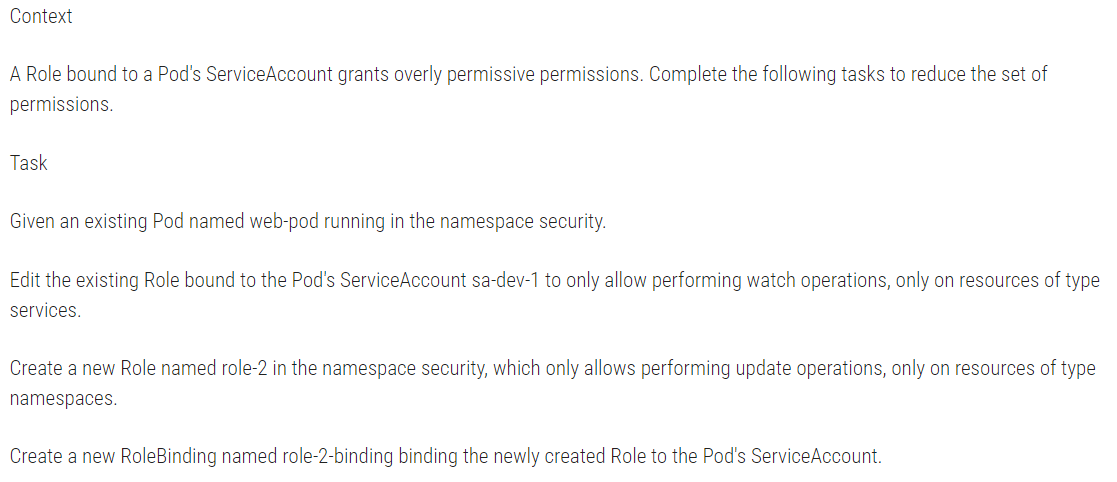
Q-

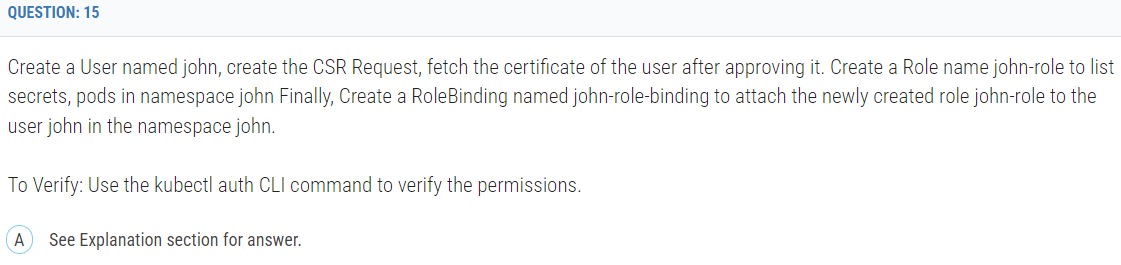


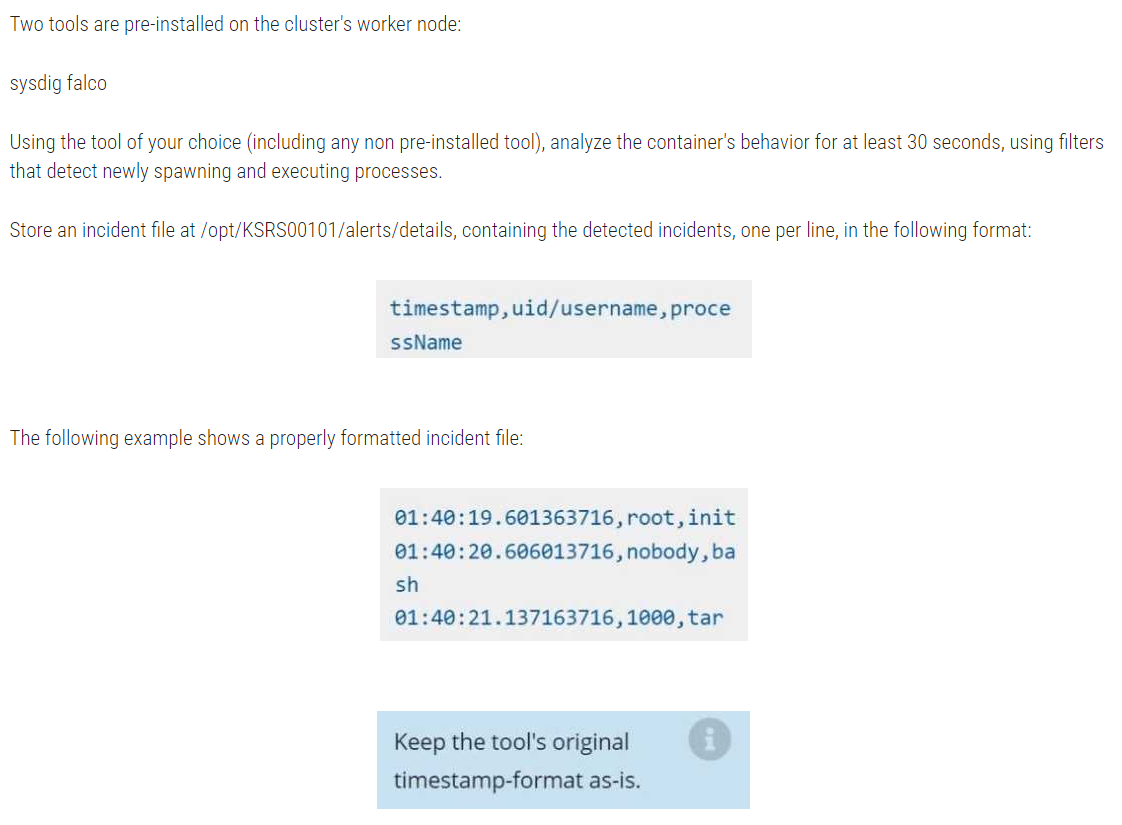
Q-

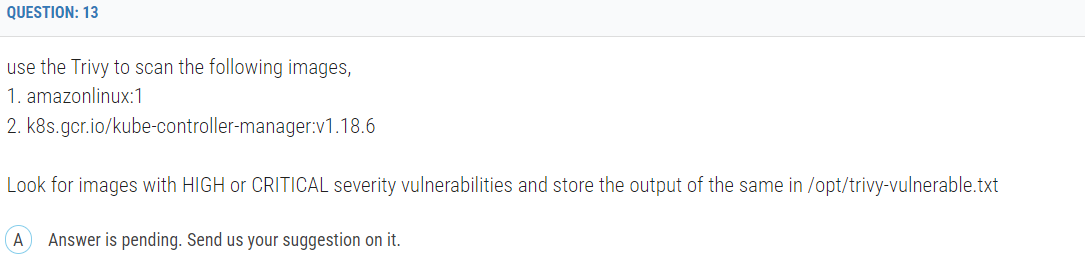


Q-

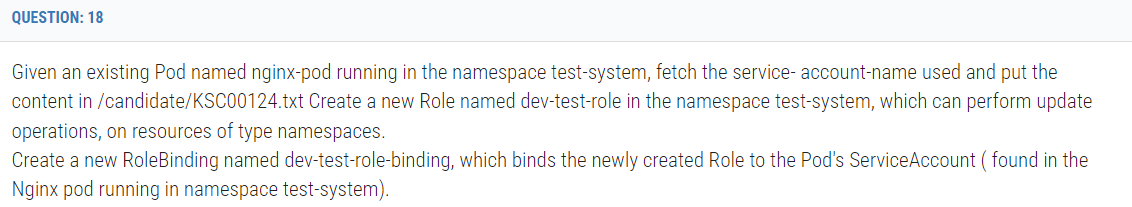


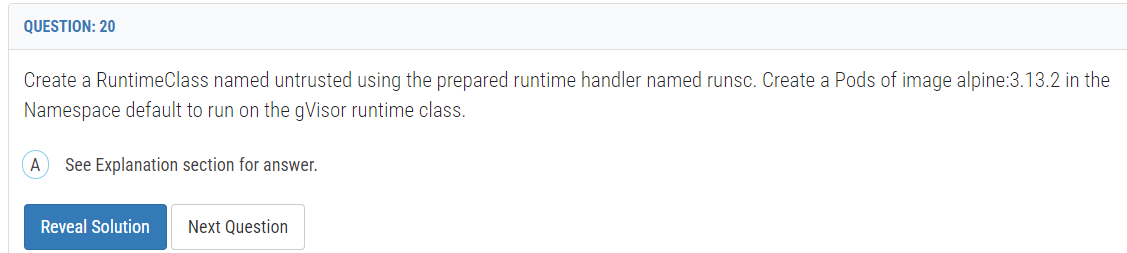


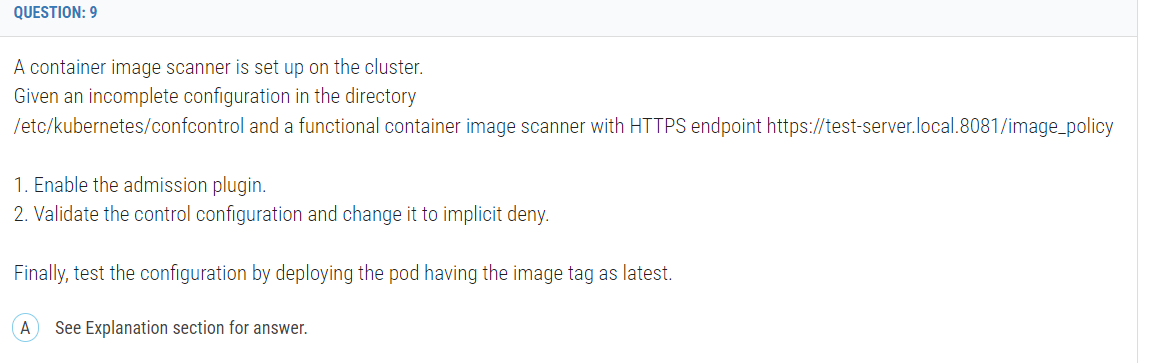




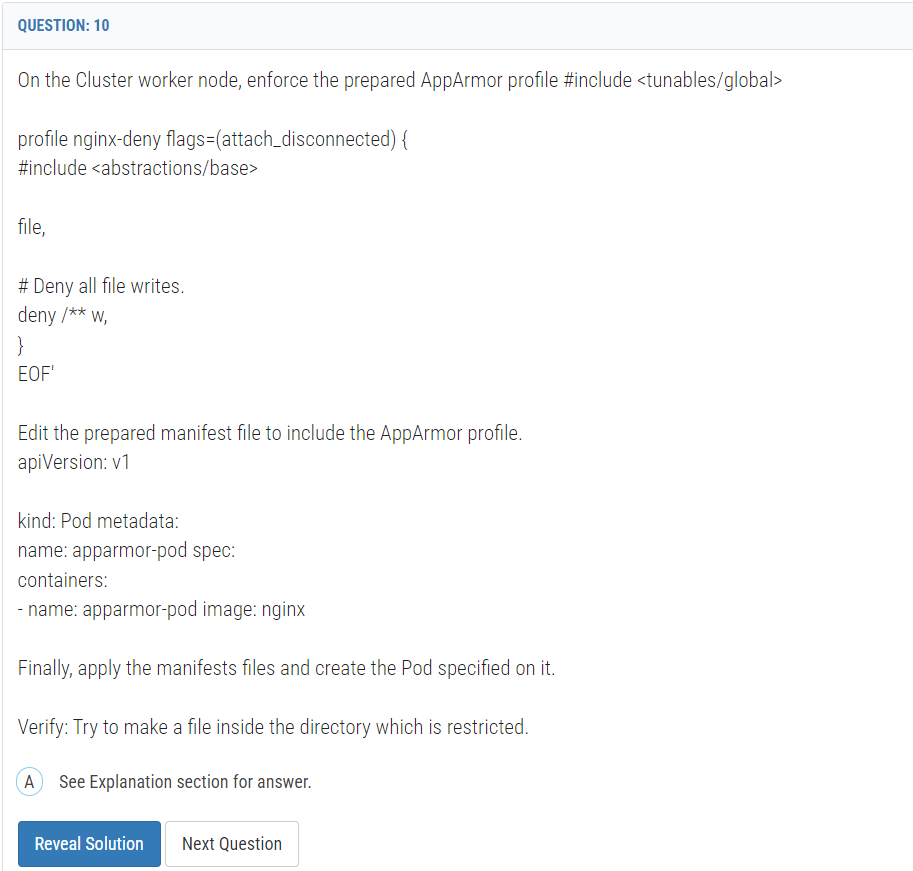
use the Trivy to scan the following images,  
1. amazonlinux:1  
2. k8s.gcr.io/kube-controller-manager:v1.18.6  
  
Look for images with HIGH or CRITICAL severity vulnerabilities and store the output of the same in /opt/trivy-vulnerable.txt

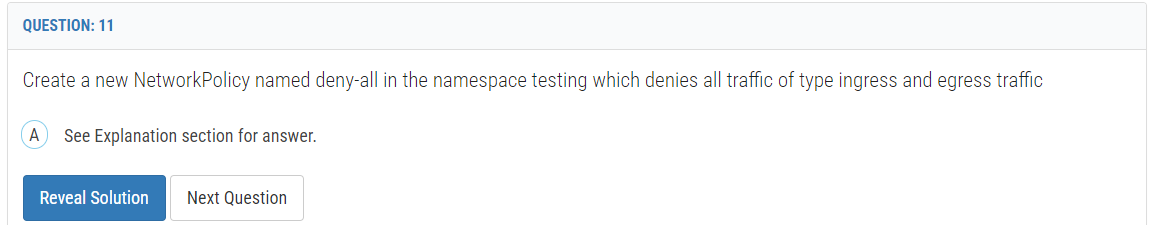






**A container image scanner is set up on the cluster.  
Given an incomplete configuration in the directory  
/etc/kubernetes/confcontrol and a functional container image scanner with HTTPS endpoint https://test-server.local.8081/image\_policy  
  
1. Enable the admission plugin.  
2. Validate the control configuration and change it to implicit deny.  
  
Finally, test the configuration by deploying the pod having the image tag as latest.**

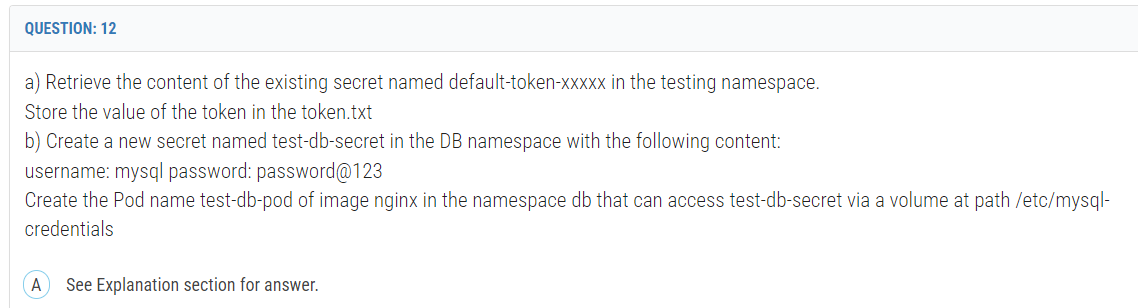




Create a new NetworkPolicy named deny-all in the namespace testing which denies all traffic of type ingress and egress traffic

**Explanation:**

You can create a "default" isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any ingress traffic to those pods.  
---  
apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy metadata:  
name: default-deny-ingress spec:  
podSelector: {}  
policyTypes:  
- Ingress  
  
You can create a "default" egress isolation policy for a namespace by creating a NetworkPolicy that selects all pods but does not allow any egress traffic from those pods.  
  
---  
apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy metadata:  
name: allow-all-egress spec:  
podSelector: {}  
egress:  
- {}  
policyTypes:  
- Egress  
  
Default deny all ingress and all egress traffic  
You can create a "default" policy for a namespace which prevents all ingress AND egress traffic by creating the following NetworkPolicy in that namespace.  
  
---  
apiVersion: networking.k8s.io/v1  
kind: NetworkPolicy metadata:  
name: default-deny-all spec:  
podSelector: {}  
policyTypes:  
- Ingress  
- Egress  
  
This ensures that even pods that aren't selected by any other NetworkPolicy will not be allowed ingress or egress traffic.



Create a new ServiceAccount named backend-sa in the existing namespace default, which has the capability to list the pods inside the namespace default. Create a new Pod named backend-pod in the namespace default, mount the newly created sa backend-sa to the pod, and Verify that the pod is able to list pods.  
Ensure that the Pod is running.

Fix all issues via configuration and restart the affected components to ensure the new setting takes effect.  
Fix all of the following violations that were found against the API server:-  
  
1.Ensure the --authorization-mode argument includes RBAC  
2.Ensure the --authorization-mode argument includes Node  
3.Ensure that the --profiling argument is set to false Fix all of the following violations that were found against the Kubelet:-  
4.Ensure the --anonymous-auth argument is set to false.  
5.Ensure that the --authorization-mode argument is set to Webhook.  
6.Fix all of the following violations that were found against the ETCD:-  
7.Ensure that the --auto-tls argument is not set to true Hint: Take the use of Tool Kube-Bench

**Context  
  
A Role bound to a Pod's ServiceAccount grants overly permissive permissions. Complete the following tasks to reduce the set of permissions.  
  
Task  
  
Given an existing Pod named web-pod running in the namespace security.  
  
Edit the existing Role bound to the Pod's ServiceAccount sa-dev-1 to only allow performing watch operations, only on resources of type services.  
  
Create a new Role named role-2 in the namespace security, which only allows performing update operations, only on resources of type namespaces.  
  
Create a new RoleBinding named role-2-binding binding the newly created Role to the Pod's ServiceAccount.**

Create a User named john, create the CSR Request, fetch the certificate of the user after approving it. Create a Role name john-role to list secrets, pods in namespace john Finally, Create a RoleBinding named john-role-binding to attach the newly created role john-role to the user john in the namespace john.  
  
To Verify: Use the kubectl auth CLI command to verify the permissions.