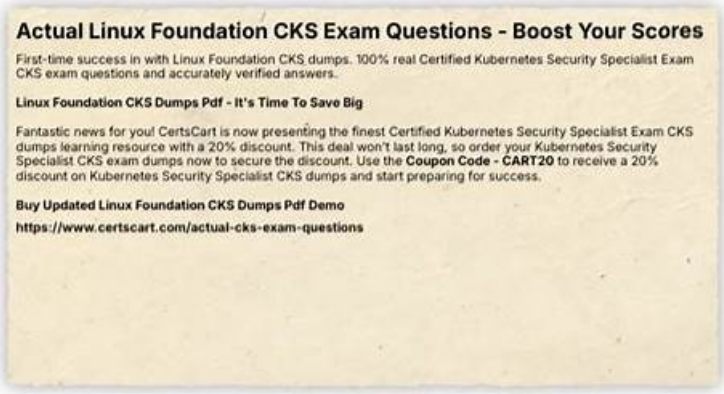
100% Pass-Rate Reliable CKS Exam Answers - Win Your Linux Foundation Certificate with Top Score



BONUS!!! Download part of TestValid CKS dumps for free: https://drive.google.com/open?id=1Tuzsp4GbtbATMi5VqwFRH-t8ooKg3H51

In addition, a 24/7 customer assistance is also available at CKS to assist you in using the product during any technical hitch. In summary, getting ready for 60 certification test might be challenging, but with the appropriate strategy and our CKS Actual Exam questions, you can clear the test in a short time.

Our CKS practice questions and answers are created according to the requirement of the certification center and the latest exam information. Our CKS real dumps cover the comprehensive knowledge points and latest practice materials that enough to help you Clear CKS Exam tests. You will get our valid CKS dumps torrent and instantly download the exam pdf after payment.

>> Reliable CKS Exam Answers <<

Latest CKS Practice Exam Guide Materials: Certified Kubernetes Security Specialist (CKS) - TestValid

The Linux Foundation CKS PDF is the most convenient format to go through all exam questions easily. It is a compilation of actual Linux Foundation CKS exam questions and answers. The PDF is also printable so you can conveniently have a hard copy of Linux Foundation CKS Dumps with you on occasions when you have spare time for quick revision. The PDF is easily downloadable from our website and also has a free demo version available.

The CKS exam is designed to assess the candidate's proficiency in security best practices for Kubernetes platforms and containerized workloads, including securing Kubernetes components, securing container images and registries, securing network communication, and configuring security contexts. CKS exam is a performance-based test, which means that the candidate must complete a series of tasks in a live Kubernetes environment, demonstrating their ability to secure Kubernetes platforms and containerized workloads.

To take the CKS Certification Exam, candidates must have a valid CNCF (Cloud Native Computing Foundation) CKA (Certified Kubernetes Administrator) certification, which demonstrates their proficiency in Kubernetes administration. Candidates must also have experience working with Kubernetes in production environments and have a good understanding of Linux command-line tools and utilities.

Linux Foundation Certified Kubernetes Security Specialist (CKS) Sample Questions (Q55-Q60):

NEW QUESTION # 55
You are responsible for hardening a Kubernetes cluster hosting sensitive financial data. One of the key security concerns is preventing data exfiltration. How can you use Kubernetes Network Policy to enforce network isolation and prevent unauthorized data access?

Answer:

Explanation:
Solution (Step by Step) :
1. Define the network policy rules:
- Identify tne pods that contain sensitive data and the services they interact with. Use labels to identify these pods and services.
- Create network policies that restrict communication between these pods and the outside world. These policies should only allow traffic from authorized sources, such as internal services or authorized user applications.
2. Create the network policy:
- Define the policy using the 'kubectl apply' command. The policy should specify the target pods, allowed ingress and egress traffic, and the allowed ports and protocols.
3. Deploy the network policy:
- Apply the network policy to the cluster. The policy will be enforced by the Kubemetes network plugin.
Example network policy:

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: financial-data-policy
  namespace: default
spec:
  podSelector:
    matchLabels:
      app: financial-data
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: internal-service
    - ipBlock:
        cidr: 10.0.0.0/16
  egress:
  - to:
    - podSelector:
        matchLabels:
          app: financial-data
    - ipBlock:
        cidr: 10.0.0.0/16
  policyTypes:
  - Ingress
  - Egress
```


This policy restricts the communication of pods with the label Sapp: financial-data' to only internal services with the label Sapp: internal-service' and to specific IP addresses in the range ' 10.0.0.0/16'. This helps prevent data exfiltration by restricting access to external services or unauthorized clients.

NEW QUESTION # 56
You nave a Kubernetes cluster with a Deployment named 'web-app- that runs multiple replicas of a web application. You need to create a network policy that allows only traffic from pods in the same namespace to access the web application's API endpoint on port 8080.

Answer:

Explanation:
Solution (Step by Step) :
1. Create a NetworkP01icy:
- Define a NetworkPolicy resource with a 'podSelector that matches the 'web-app' Deployment.
- Create an 'ingress' rule that allows traffic from pods within the same namespace.
- Use the 'from' field to specify the namespace and set the 'namespaceselector' to 'matchLabels: {}' to include all pods in the namespace.
- Ensure that the port 8080 is included in the 'ports' field.

2. Apply the NetworkPolicy: - Apply the YAML file using 'kubectl apply -f web-app-namespace-policy.yaml 3. Verify the NetworkPoIicy: - Use 'kubectl get networkpolicies' to list the available network policies. - Use 'kubectl describe networkpolicy web-app-namespace-policy' to view the details of the applied policy. 4. Test the NetworkPolicy: - Deploy a pod in the same namespace as the 'web-app' Deployment and attempt to access the API endpoint Verify that the connection is successful. - Deploy a pod in a different namespace and attempt to access the API endpoint Verity that the connection is denied.

NEW QUESTION # 57
You are tasked with securing a Kubemetes cluster that is running on AWS- One of the security best practices you want to implement is to limit tne number of IP addresses that can access the Kubernetes API server. You need to configure the 'kube-apiserver' to only allow access from specific IP addresses, using the '--insecure-bind-address' flag to restrict access.
How would you configure 'kube-apiserver' to achieve this using an '--insecure-bind-address' flag, but allow access from only specific IP addresses?

Answer:

Explanation:
Solution (Step by Step) :
1 . Identify Allowed IP Addresses: Determine the specific IP addresses that should be allowed to access the Kubernetes API server. For example, you might allow access from your local machine's IP address (e.g., 192.168.1.100), and the IP addresses of any bastion hosts that are used for remote management.
2. Modify the 'kube-apiserver' Configuration:
- Locate the 'kube-apjserver' configuration file (typically found at "etc/kubernetestmanifests/kube-apiserver.yaml or similar).
- In tne 'kube-apiserver' configuration file, find tne '--insecure-bind-address' flag.
- Set the '--insecure-bind-address' flag to '0.0.0.0' to allow access from all IP addresses.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: kube-apiserver
spec:
  replicas: 1
  selector:
    matchLabels:
      app: kube-apiserver
  template:
    metadata:
      labels:
        app: kube-apiserver
    spec:
      containers:
      - name: kube-apiserver
        image: k8s.gcr.io/kube-apiserver:v1.24.3
        command:
        - kube-apiserver
        - --insecure-bind-address=0.0.0.0
        - --authorization-mode=RBAC
        - --client-ca-file=/etc/kubernetes/pki/ca.crt
        - --tls-cert-file=/etc/kubernetes/pki/apiserver.crt
        - --tls-private-key-file=/etc/kubernetes/pki/apiserver.key
        # Additional parameters for kube-apiserver
      # Define the security context for the container
      securityContext:
        # Set the privileged flag to false
        privileged: false
        # Set the runAsNonRoot flag to true
        runAsNonRoot: true
        # Set the allowPrivilegeEscalation flag to false
        allowPrivilegeEscalation: false
        # Set the runAsUser to 1000
        runAsUser: 1000
```


3. Restart 'kube-apiserver': Apply the updated configuration file. Depending on how the Kubernetes cluster is deployed, you may need to restart the 'kube-apisepver' pod or container. 4. Verify the Configuration: - After restarting 'kube-apiservers , test that you can access the API server from the allowed IP addresses. - Test from any disallowed IP addresses to confirm access is blocked.

NEW QUESTION # 58
SIMULATION
Use the kubesec docker images to scan the given YAML manifest, edit and apply the advised changes, and passed with a score of 4 points.
kubesec-test.yaml
apiVersion: v1
kind: Pod
metadata:
name: kubesec-demo
spec:
containers:
- name: kubesec-demo
image: gcr.io/google-samples/node-hello:1.0
securityContext:
readOnlyRootFilesystem: true
Hint: docker run -i kubesec/kubesec:512c5e0 scan /dev/stdin < kubesec-test.yaml

- A. Send us the Feedback on it.

Answer: A

NEW QUESTION # 59
You can switch the cluster/configuration context using the following command: [desk@cli] $ kubectl config use-context stage
Context: A PodSecurityPolicy shall prevent the creation of privileged Pods in a specific namespace. Task: 1. Create a new PodSecurityPolcy named deny-policy, which prevents the creation of privileged Pods. 2. Create a new ClusterRole name deny-access-role, which uses the newly created PodSecurityPolicy deny-policy. 3. Create a new ServiceAccount named psd-denial-sa in the existing namespace development. Finally, create a new ClusterRoleBindind named restrict-access-bind, which binds the newly created ClusterRole deny-access-role to the newly created ServiceAccount psp-denial-sa

Answer:

Explanation:
Create psp to disallow privileged container
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: deny-access-role
rules:
- apiGroups: ['policy']
resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- "deny-policy"
k create sa psp-denial-sa -n development
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRoleBinding
metadata:
name: restrict-access-bing
roleRef:
kind: ClusterRole
name: deny-access-role
apiGroup: rbac.authorization.k8s.io
subjects:
- kind: ServiceAccount
name: psp-denial-sa
namespace: development
Explanation
master1 $ vim psp.yaml
apiVersion: policy/v1beta1
kind: PodSecurityPolicy
metadata:
name: deny-policy
spec:
privileged: false # Don't allow privileged pods!
seLinux:
rule: RunAsAny
supplementalGroups:
rule: RunAsAny
runAsUser:
rule: RunAsAny
fsGroup:
rule: RunAsAny
volumes:
- '*'
master1 $ vim cr1.yaml
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
name: deny-access-role
rules:

- apiGroups: ['policy']
resources: ['podsecuritypolicies']
verbs: ['use']
resourceNames:
- "deny-policy"
master1 $ k create sa psp-denial-sa -n development master1 $ vim cb1.yaml apiVersion: rbac.authorization.k8s.io/v1 kind: ClusterRoleBinding metadata:
name: restrict-access-bing
roleRef:
kind: ClusterRole
name: deny-access-role
apiGroup: rbac.authorization.k8s.io
subjects:
Authorize specific service accounts:
- kind: ServiceAccount
name: psp-denial-sa
namespace: development
master1 $ k apply -f psp.yaml master1 $ k apply -f cr1.yaml master1 $ k apply -f cb1.yaml Reference: https://kubernetes.io/docs/concepts/policy/pod-security-policy/

NEW QUESTION # 60
.....