CKS最速合格 & CKS合格率

さらに、ShikenPASS CKSダンプの一部が現在無料で提供されています：https://drive.google.com/open?id=1PG-7sLzCEsqOtYgqEvSwLhLbcFACpCya

テスト志向の高品質なCKS試験問題があなたにとって最良の選択であると信じています。すべての受験者がCKS試験に合格し、CKS準備ガイドの多大なメリットを享受できることを心から願っています。 CKS試験問題の合格率は99％～100％です。受験者がCKS試験に合格できるようにすることは、当社の文化において常に長所であり、購入および使用のプロセスでメールで連絡を取ることができます。できるだけ早く返信いたします。

Linux Foundation CKS（Certified Kubernetes Security Specialist）試験は、Kubernetesクラスターの確保に関する専門知識を実証したい専門家の高度な認定です。この認定は、安全なKubernetesクラスターの設計、展開、管理に必要なスキルと知識をテストするように設計されています。これは、クラウドネイティブアプリケーションとインフラストラクチャの管理に関与しているIT専門家にとって重要な認証です。

>> CKS最速合格 <<

Linux Foundation CKS合格率、CKS日本語版復習指南

Linux Foundation CKSテスト質問の回答を注文する予定です。 クレジットカードが必要です。 ほとんどの場合、クレジットカードをサポートしています。 デビットカードをお持ちの場合は、クレジットカードを申請するか、他の友人にCKSテスト質問の回答の支払いを手伝ってもらってください。 通常、候補者はPayPalで支払うことをお勧めします。ここでは、PayPalアカウントを持っている必要はありません。 [PayPal]をクリックする

と、クレジットカード支払いに振り替えられます。 CKSテストの質問の回答にSWREG支払いを選択した場合、一部の国では追加の税金がかかります。

Linux Foundation CKS（Certified Kubernetes Security Specialist）試験は、Kubernetesクラスタをセキュアに保つ専門家のスキルと知識を認定する認定プログラムです。コンテナオーケストレーションとデプロイメントの世界でKubernetesがますます人気を集めるにつれて、熟練したセキュリティスペシャリストの必要性がますます重要になっています。

Linux Foundation Certified Kubernetes Security Specialist (CKS) 認定 CKS 試験問題 (Q71-Q76):

質問 # 71
Before Making any changes build the Dockerfile with tag base:v1
Now Analyze and edit the given Dockerfile(based on ubuntu 16:04)
Fixing two instructions present in the file, Check from Security Aspect and Reduce Size point of view.
Dockerfile:
FROM ubuntu:latest
RUN apt-get update -y
RUN apt install nginx -y
COPY entrypoint.sh /
RUN useradd ubuntu
ENTRYPOINT ["/entrypoint.sh"]
USER ubuntu
entrypoint.sh
#!/bin/bash
echo "Hello from CKS"
After fixing the Dockerfile, build the docker-image with the tag base:v2

- A. To Verify: Check the size of the image before and after the build.

正解：A

質問 # 72
SIMULATION
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.

- A. Send us the Feedback on it.

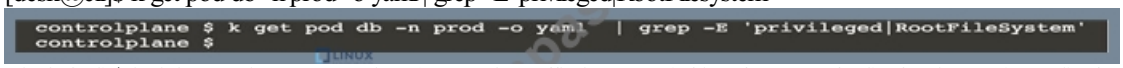
正解：A

質問 # 73
You must complete this task on the following cluster/nodes: Cluster: immutable-cluster Master node: master1 Worker node: worker1 You can switch the cluster/configuration context using the following command:
[desk@cli] $ kubectl config use-context immutable-cluster
Context: It is best practice to design containers to be stateless and immutable.
Task:
Inspect Pods running in namespace prod and delete any Pod that is either not stateless or not immutable.
Use the following strict interpretation of stateless and immutable:
1. Pods being able to store data inside containers must be treated as not stateless.
Note: You don't have to worry whether data is actually stored inside containers or not already.
2. Pods being configured to be privileged in any way must be treated as potentially not stateless or not immutable.

正解：

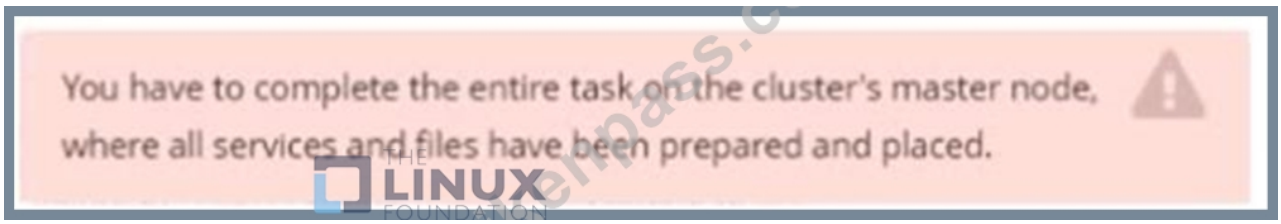
解説：
k get pods -n prod
k get pod <pod-name> -n prod -o yaml | grep -E 'privileged|ReadOnlyRootFileSystem' Delete the pods which do have any of these 2 properties privileged:true or ReadOnlyRootFileSystem: false
[desk@cli]$ k get pods -n prod
NAME READY STATUS RESTARTS AGE
cms 1/1 Running 0 68m
db 1/1 Running 0 4m
nginx 1/1 Running 0 23m
[desk@cli]$ k get pod nginx -n prod -o yaml | grep -E 'privileged|RootFileSystem'
{"apiVersion":"v1","kind":"Pod","metadata":{"annotations":{},"creationTimestamp":null,"labels":
{"run":"nginx"},"name":"nginx","namespace":"prod"},"spec":{"containers":[{"image":"nginx","name":"nginx","resources":
{},"securityContext":{"privileged":true}}],"dnsPolicy":"ClusterFirst","restartPolicy":"Always"},"status":{}} f:privileged: {} privileged: true

[desk@cli]$ k delete pod nginx -n prod
[desk@cli]$ k get pod db -n prod -o yaml | grep -E 'privileged|RootFilesystem'



[desk@cli]$ k delete pod cms -n prod Reference: https://kubernetes.io/docs/concepts/policy/pod-security-policy/ https://cloud.google.com/architecture/best-practices-for-operating-containers Reference:
[desk@cli]$ k delete pod cms -n prod Reference: https://kubernetes.io/docs/concepts/policy/pod-security-policy/ https://cloud.google.com/architecture/best-practices-for-operating-containers

質問 # 74
Context
A container image scanner is set up on the cluster, but it's not yet fully integrated into the cluster s configuration. When complete, the container image scanner shall scan for and reject the use of vulnerable images.
Task



Given an incomplete configuration in directory /etc/kubernetes/epconfig and a functional container image scanner with HTTPS endpoint https://wakanda.local:8081 /image_policy :
1. Enable the necessary plugins to create an image policy
2. Validate the control configuration and change it to an implicit deny
3. Edit the configuration to point to the provided HTTPS endpoint correctly Finally, test if the configuration is working by trying to deploy the vulnerable resource /root/KSSC00202/vulnerable-resource.yml.

正解：

解説：

```
Switched to context "KSSC00202".
candidate@cli:~$ ssh kssc00202-master
Warning: Permanently added '10.177.80.12' (ECDSA) to the list of known hosts.

The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

root@kssc00202-master:~# ls /etc/kubernetes/epconfig/
admission_configuration.json  apiserver-client-key.pem  apiserver-client.pem  kubeconfig.yaml  webhook-key.pem  webhook.pem
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
```

```
"imagePolicy":
  "kubeConfigFile": "/etc/kubernetes/epconfig/kubeconfig.yaml",
  "allowTTL": 50,
  "denyTTL": 50,
  "retryBackoff": 500,
  "defaultAllow": false
```

```
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/kubeconfig.yaml
```

```
apiVersion: v1
clusters:
- cluster:
    certificate-authority: /etc/kubernetes/epconfig/webhook.pem # CA for verifying the remote service.
    server: https://wakanda.local:8081/image_policy
  name: kubernetes
contexts:
- context:
    cluster: kubernetes
    user: kubernetes-admin
  name: kubernetes-admin@kubernetes
current-context: kubernetes-admin@kubernetes
kind: Config
preferences: {}
users:
- name: kubernetes-admin
  user:
    client-certificate: /etc/kubernetes/epconfig/apiserver-client.pem
    client-key: /etc/kubernetes/epconfig/apiserver-client-key.pem
```

```
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/admission_configuration.json
root@kssc00202-master:~# vim /etc/kubernetes/epconfig/kubeconfig.yaml
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml p
```

```
ind: Pod
etadata:
 annotations:
   kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.177.80.12:6443
 creationTimestamp: null
 labels:
   component: kube-apiserver
   tier: control-plane
 name: kube-apiserver
 namespace: kube-system
pec:
 containers:
   - command:
       - kube-apiserver
       - --advertise-address=10.177.80.12
       - --allow-privileged=true
       - --authorization-mode=Node,RBAC
       - --client-ca-file=/etc/kubernetes/pki/ca.crt
       - --enable-admission-plugins=NodeRestriction
       - --enable-bootstrap-token-auth=true
       - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
       - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
       - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
       - --etcd-servers=https://127.0.0.1:2379
       - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.cr
       - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
       - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
       - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
       - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
       - --requestheader-allowed-names=front-proxy-client
       - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
       - --requestheader-extra-headers-prefix=X-Remote-Extra-
/etc/kubernetes/manifests/kube-apiserver.yaml" 135L, 4626C
```

```
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml p
2 files to edit
root@kssc00202-master:~# rm -f p
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
```

```
apiVersion: v1
kind: Pod
metadata:
  annotations:
    kubeadm.kubernetes.io/kube-apiserver.advertise-address.endpoint: 10.177.80.12:6443
  creationTimestamp: null
  labels:
    component: kube-apiserver
    tier: control-plane
  name: kube-apiserver
  namespace: kube-system
spec:
  containers:
    - command:
        - kube-apiserver
        - --advertise-address=10.177.80.12
        - --allow-privileged=true
        - --authorization-mode=Node,RBAC
        - --client-ca-file=/etc/kubernetes/pki/ca.crt
        - --enable-admission-plugins=NodeRestriction,ImagePolicyWebHook
        - --admission-control-config-file=/etc/kubernetes/epconfig/admin.conf
        - --enable-bootstrap-token-auth=true
        - --etcd-cafile=/etc/kubernetes/pki/etcd/ca.crt
        - --etcd-certfile=/etc/kubernetes/pki/apiserver-etcd-client.crt
        - --etcd-keyfile=/etc/kubernetes/pki/apiserver-etcd-client.key
        - --etcd-servers=https://127.0.0.1:2379
        - --kubelet-client-certificate=/etc/kubernetes/pki/apiserver-kubelet-client.crt
        - --kubelet-client-key=/etc/kubernetes/pki/apiserver-kubelet-client.key
        - --kubelet-preferred-address-types=InternalIP,ExternalIP,Hostname
        - --proxy-client-cert-file=/etc/kubernetes/pki/front-proxy-client.crt
        - --proxy-client-key-file=/etc/kubernetes/pki/front-proxy-client.key
        - --requestheader-allowed-names=front-proxy-client
        - --requestheader-client-ca-file=/etc/kubernetes/pki/front-proxy-ca.crt
```

```
root@kssc00202-master:~# vim /etc/kubernetes/manifests/kube-apiserver.yaml
root@kssc00202-master:~# systemctl daemon-reload
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# systemctl restart kubelet.service
root@kssc00202-master:~# systemctl enable kubelet.service
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# ls
KSSC00202  snap
root@kssc00202-master:~# cat KSSC00202/vulnerable-resource.yml
```

```
KSSC00202  snap
root@kssc00202-master:~# cat KSSC00202/vulnerable-resource.yml
---
apiVersion: v1
kind: ReplicationController
metadata:
  name: nginx-latest
spec:
  replicas: 1
  selector:
    app: nginx-latest
  template:
    metadata:
      name: nginx-latest
      labels:
        app: nginx-latest
    spec:
      containers:
      - name: nginx-latest
        image: nginx
        ports:
        - containerPort: 80
root@kssc00202-master:~# kubectl create -f KSSC00202/vulnerable-resource.yml
```

```
root@kssc00202-master:~# kubectl create -f KSSC00202/vulnerable-resource.yml
The connection to the server 10.177.80.12:6443 was refused - did you specify the right host or port?
root@kssc00202-master:~# kubectl get pods
The connection to the server 10.177.80.12:6443 was refused - did you specify the right host or port?
root@kssc00202-master:~# ls -al .kube/
total 20
drwxr-xr-x  3 root root 4096 Aug  3 04:07 .
drwx------. 9 root root 4096 Oct 11 15:36 ..
drwxr-x---  4 root root 4096 Aug  3 04:07 cache
-rw-r--r--  1 root root 5636 Aug  3 04:07 config
root@kssc00202-master:~#  crictl ps -a
```

```
012ea858713be       a634548d10b03       2 months ago        Exited              kube-proxy          0                   1460a9f
a0f1e0              kube-proxy-cmjb5
405227dfa4900       aebe758cef4cd       2 months ago        Exited              etcd                0                   cfb6522
e720fb              etcd-kssc00202-master
root@kssc00202-master:~# ls -al .kube/ | grep kube-api
root@kssc00202-master:~# crictl ps -a | grep kube-api
WARN[0000] runtime connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:///ru
n/crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint instead.
ERRO[0000] unable to determine runtime API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Error wh
ile dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
WARN[0000] image connect using default endpoints: [unix:///var/run/dockershim.sock unix:///run/containerd/containerd.sock unix:///run/
crio/crio.sock unix:///var/run/cri-dockerd.sock]. As the default settings are now deprecated, you should set the endpoint instead.
ERRO[0000] unable to determine image API version: rpc error: code = Unavailable desc = connection error: desc = "transport: Error whil
e dialing dial unix /var/run/dockershim.sock: connect: no such file or directory"
a003b3fdfb61c       d3377ffb7177c       30 seconds ago      Exited              kube-apiserver      3                   2dadb4e
984a91              kube-apiserver-kssc00202-master
5e70b9a70f9ed       d3377ffb7177c       7 hours ago         Exited              kube-apiserver      0                   68a9f31
6c2559              kube-apiserver-kssc00202-master
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~#
root@kssc00202-master:~# exit
logout
Connection to 10.177.80.12 closed.
candidate@cli:~$
```

質問 # 75
You're in charge of enforcing a secure supply chain in your Kubernetes environment. You need to ensure that all container images deployed to your cluster are scanned for known vulnerabilities before being deployed. How would you achieve this?

正解：

解説：
Solution (Step by Step) :
1. Choose a Vulnerability Scanner:
- Select a reputable container image vulnerability scanner. Popular options include:
- Aqua Security: A comprehensive platform that offers image scanning, runtime security, and policy enforcement.
- JFrog Xray: A vulnerability scanner that integrates with JFrog Artifactory, providing deep scanning capabilities.
- Ancnore Engine: An open-source scanner that can be deployed on-premises or in the Cloud.
2. Integrate with Your Registry (if applicable):
- If your vulnerability scanner support integration with your registry (e.g., Docker Hub, Harbor), configure it to scan images automatically as they are pushed.
- This approach provides real-time vulnerability scanning, ensuring that only secure images are available for deployment.
3. Implement a Scanning Pipeline (if needed):
- If your chosen scanner doesn't integrate with your registry, build a scanning pipeline using a CI/CD tool like Jenkins, GitLab Cl, or CircleCl.
- The pipeline should:
- Pull the image from the registry.
- Run the vulnerability scanner against the image.
- Fail the build if any critical vulnerabilities are found.
- If no critical vulnerabilities are found, push the scanned image to the registry with a tag indicating its scan status.
4. Configure Kubernetes Policies:
- Use Kubernetes policies (like Pod Security Policies or Admission Controllers) to enforce the following:
- Restrict deployments to images with a "scanned" tag: This ensures only images that have undergone vulnerability scanning are deployed.
- Block deployments of images with known critical vulnerabilities: This prevents deployment of images with unacceptable risks.
5. Monitor Scanning Results:
- Continuously monitor vulnerability scanning results.
- Keep track of vulnerabilities found and their severity.
- Update your policies to reflect changes in vulnerability scanning results.
6. Remediation and Patching:
- Have a process in place to remediate and patch vulnerabilities found in images.
- Work with developers and security teams to address vulnerabilities promptly.

質問 # 76
.....

て、無料でダウンロードしてくださいCKS勉強の資料

- CKSリンクグローバル □ CKS試験概要 □ CKS無料ダウンロード □ { CKS }を無料でダウンロード✔ jp.fast2test.com □✔□で検索するだけCKSリンクグローバル
- ハイパスレートLinux Foundation CKS最速合格 - 一番いいGoShiken - 資格試験のリーダープロバイダー □ 【 www.goshiken.com 】から簡単に“ CKS ”を無料でダウンロードできますCKS受験対策書
- www.mogiexam.comはLinux Foundation CKS試験の実践訓練を提供する □ 《 www.mogiexam.com 》から簡単に➡ CKS □を無料でダウンロードできますCKS参考書内容
- smartrepair.courses, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, lms24.blogdu.de, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, myportal.utt.edu.tt, ascentleadershipinstitute.org, www.stes.tyc.edu.tw, Disposable vapes

さらに、ShikenPASS CKSダンプの一部が現在無料で提供されています：https://drive.google.com/open?id=1PG-7sLzCEsqOtYgqEvSwLhLbcFACpCya