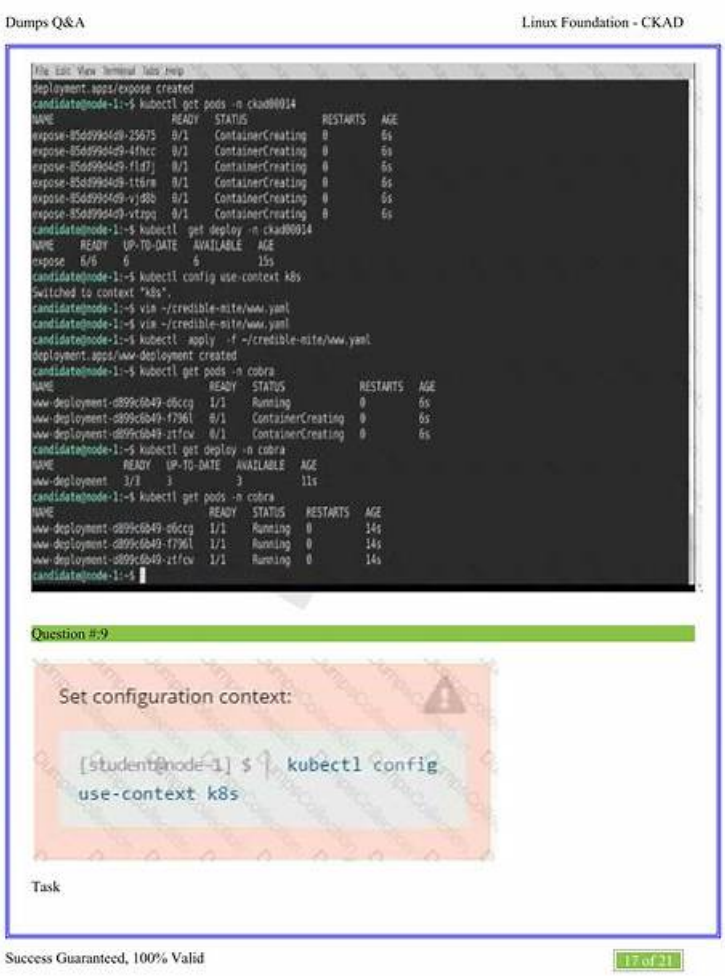
CKAD過去問題 & CKAD模擬試験最新版



P.S. Pass4TestがGoogle Driveで共有している無料かつ新しいCKADダンプ：https://drive.google.com/open?id=1DzNN5FOI5M0Pl5kh4ETC5qWt5nNodVPw

今日の社会では、能力を高めるために証明書を取得することを優先する人がますます増えています。まったく新しい観点から、CKAD学習資料は、CKAD認定の取得を目指すほとんどのオフィスワーカーに役立つように設計されています。当社のCKADテストガイドは、現代の人材開発に歩調を合わせ、すべての学習者を社会のニーズに適合させます。 CKADの最新の質問が、関連する知識の蓄積と能力強化のための最初の選択肢になることは間違いありません。

Linux Foundation Certified Kubernetes Application Developer（CKAD）試験は、Kubernetesでアプリケーションを開発および展開する際の個人のスキルを検証する認定プログラムです。 Kubernetesはオープンソースのコンテナオーケストレーションプラットフォームであり、業界全体でコンテナ管理の標準となっています。 CKAD認定は、Kubernetesアプリケーションの開発と展開に関する専門知識を示しています。

>> CKAD過去問題 <<

試験の準備方法-実際的なCKAD過去問題試験-素晴らしいCKAD模擬試験最新版

Linux FoundationのCKAD試験に合格するのは早ければ速いほどIT業界で発展られたいあなたにとってはよいです。あなたはこの重要な試験を準備するのは時間とお金がかかると聞いたことがあるかもしれませんが、それは我々提供するLinux FoundationのCKADソフトを利用しなかったからです。複雑な整理と分析の過程はもう我々に完了されました。あなたは高効率の復習とLinux FoundationのCKAD試験の成功を経験する必要があればいいで

す。

CKAD認定試験は、実践的なパフォーマンスベースの試験です。つまり、候補者は指定された時間制限内で一連のタスクを完了する必要があります。この試験はオンラインで実施されており、候補者はKubernetesクラスターにアクセスする必要があります。認定試験では、Kubernetesアーキテクチャを理解し、アプリケーションを展開および管理し、サービスを構成および実行し、一般的な問題をトラブルシューティングする候補者の能力をテストします。認証試験は、候補者の実践的なスキルとクベルネテスの知識の証であり、クベルネテスの専門家を雇うことを検討している組織によって世界的に認識されています。

Linux Foundation Certified Kubernetes Application Developer Exam 認定 CKAD 試験問題 (Q189-Q194):

質問 # 189
You are tasked with setting up a secure Kubernetes cluster for a web application. The application has sensitive data that must be protected. You need to configure a mecnanism to restrict access to the application's pods based on user identities. Describe a method to achieve this using Kubernetes RBAC and Service Accounts, ensuring that only authorized users can access specific pods.

正解：

解説：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Service Account
- Create a Service Account for the application:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: webapp-sa
```

- Apply the Service Account configuratiom basn kubectl apply -f webapp-sa.yaml 2. Create a Role: - Define a Role that grants access to the specific pods:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: webapp-pod-reader
  namespace:
rules:
- apiGroups: ["apps"]
  resources: ["deployments"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps"]
  resources: ["deployments/finalizers"]
  verbs: ["update"]
- apiGroups: ["apps"]
  resources: ["statefulsets"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps"]
  resources: ["statefulsets/finalizers"]
  verbs: ["update"]
- apiGroups: ["extensions"]
  resources: ["ingresses"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["extensions"]
  resources: ["ingresses/finalizers"]
  verbs: ["update"]
- apiGroups: ["extensions"]
  resources: ["daemonsets"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["extensions"]
  resources: ["daemonsets/finalizers"]
  verbs: ["update"]
- apiGroups: ["batch"]
  resources: ["jobs"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["batch"]
  resources: ["jobs/finalizers"]
  verbs: ["update"]
- apiGroups: ["batch"]
  resources: ["cronjobs"]
```

```
  verbs: ["get", "list", "watch"]
- apiGroups: ["batch"]
  resources: ["cronjobs/finalizers"]
  verbs: ["update"]
- apiGroups: ["policy"]
  resources: ["podsecuritypolicies"]
  verbs: ["use"]
- apiGroups: ["admissionregistration.k8s.io"]
  resources: ["validatingwebhookconfigurations"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["admissionregistration.k8s.io"]
  resources: ["mutatingwebhookconfigurations"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["rbac.authorization.k8s.io"]
  resources: ["roles", "rolebindings"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["", "apps", "extensions", "batch"]
  resources: ["pods"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["", "apps", "extensions", "batch"]
  resources: ["pods/log"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/exec"]
  verbs: ["create"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/portforward"]
  verbs: ["create"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/proxy"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/attach"]
  verbs: ["create"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/status"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/binding"]
  verbs: ["get", "list", "watch"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/eviction"]
  verbs: ["create"]
- apiGroups: ["apps", "extensions", "batch"]
  resources: ["pods/delete"]
  verbs: ["delete"]
```

- Apply the Role configuratiom bash kubectl apply -f webapp-pod-reader.yaml 3. Create a RoleBinding: - Bind the Role to the Service Account

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: webapp-pod-reader-binding
  namespace:
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: webapp-pod-reader
subjects:
- kind: ServiceAccount
  name: webapp-sa
  namespace:
```

- Apply the RoleBinding configuration: bash kubectl apply -f webapp-pod-reader-binding_yaml 4. Configure the Application: - When deploying the application, specify the Service Account:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: webapp-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: webapp
  template:
    metadata:
      labels:
        app: webapp
    spec:
      serviceAccountName: webapp-sa
      containers:
      - name: webapp
        image:
```

5. Verify Access: - Use the 'kubectr command with the Service Account's credentials to verify that only authorized users can access the application's pods: bash kubectl -service-account=webapp-sa get pods -n This setup utilizes Kubernetes RBAC to control access to the application's pods. - The Service Account acts as an identity for the application. - The Role defines the permissions granted to the Service Account, specifically allowing access to the pods. - The RoleBinding associates the Role with the Service Account, linking the permissions to the identity. - When the application is deployed witn tne specified Service Account, it inherits the permissions defined in the RoleBinding. This ensures that only users with the necessary credentials (associated with the Service Account) can access and interact with the application's pods, safeguarding sensitive data.

質問 # 190
You have a container image that contains a Python application. The application depends on specific libraries that are not included in the base image used for the container. Describe the steps involved in modifying the image to install the necessary libraries Without rebuilding the entire application.

正解：

解説：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Dockerfile:
- Create a new 'Dockerfile' with the following content

```
FROM existing-image:latest # Use the existing image as the base
COPY --from=existing-image:latest /app /app
WORKDIR /app
RUN pip install -r requirements.txt # Install dependencies
CMD ["python", "app.py"] # Start the application
```

- Replace 'existing-image:latest' with the name of your current container image. - Replace 'requirements-txt' with the name of your file containing the list of Python libraries. 2. Build the Image: - Build the new image using the Dockerfile: docker build -t updated-image:latest 3. Update the Deployment - Modify your Deployment YAML file to use the newly built image:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-python-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-python-app
  template:
    metadata:
      labels:
        app: my-python-app
    spec:
      containers:
      - name: my-python-app
        image: updated-image:latest  # Use the new image name
        ports:
        - containerPort: 8080
      restartPolicy: Always
```

4. Apply the Changes: - Apply the updated Deployment using Skubectl apply -f deployment.yamr. This will trigger a rolling update to the pods using the new image. 5. Verify the Update: - Check the logs of the pods using 'kubectl logs -f . You should see the application running with the installed libraries. 6. Test the Application: - Access your application and ensure it functions correctly with the new libraries.

質問 # 191
You are tasked Witn designing a multi-container Pod tnat nosts botn a web server and a database. The web server should be able to connect to the database within the pod- How would you implement this design, including networking considerations?

正解：

解説：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Define the Pod YAMI-: Create a Pod definition that includes two containers: one tor the web server and one for the database.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: web-db-pod
spec:
  containers:
  - name: webserver
    image: nginx:latest
    ports:
    - containerPort: 80
    env:
    - name: DB_HOST
      value: "db"
  - name: db
    image: mysql:latest
    ports:
    - containerPort: 3306
```

2. Configure Networking: The key to allowing the web server to connect to the database is to use the pod's internal network. Since containers Within a pod share the same network namespace, you can configure the webserver to connect to the database using the name "db". 3. Environment Variables: Set an environment variable CDB HOST') within the webserver container to point to the database container by its name. This ensures the web server can correctly connect to the database within the pod. 4. Pod Deployment: Apply the YAML to create the pod using 'kubectl apply -f web-db-pod.yamr. 5. Verification: To check the pod's

status: - Run 'kubectl get pods' - Check the logs of the web server container to confirm it can connect to the database. 6. Important Note: In this example, we're using the default pod networking within Kubernetes. For more complex applications, consider using a service to expose the database container This will allow access to the database from outside the pod.,

質問 # 192
You have a Node.js application that runs in a Kubernetes cluster. The application requires access to a MySQL database hosted externally on a different server. Due to security concerns, you cannot directly expose the database to the application pod. Describe how you can implement a network policy to enable secure communication between the application pod and the MySQL database.

正解：

解説：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a Network Policy:
- Create a Network Policy that allows traffic only from the application pods to the MySQL database server-
- Define the podSelector' to specify the application pods that should be allowed to connect to the database.
- Use 'ingress' rules to define the allowed incoming traffic from the application pods.
- Specify the 'from' field to identify the source pods using labels or namespaces-
- Set the 'to' field to specify the target IP address or range of the MySQL database server

```
apiVersion: networking.k8s.io/v1
kind: NetworkPolicy
metadata:
  name: mysql-access
spec:
  podSelector:
    matchLabels:
      app: my-node-app
  ingress:
  - from:
    - podSelector:
        matchLabels:
          app: my-node-app
    to:
    - ipBlock:
        cidr: 192.168.1.100/32 # Replace with the actual MySQL server IP address
        except: []
    ports:
    - protocol: TCP
      port: 3306
```


2. Deploy the Network Policy: - Apply the Network Policy to your Kubernetes cluster using 'kubectl apply -f mysql-access.yamr 3. Configure the Application: - Configure your Node.js application to connect to the MySQL database using the IP address or hostname of the database server. - Ensure that the Node.js application has appropriate security credentials to access the database. 4. Test the Application: - Run your application and verify that it can connect to the MySQL database successfully. Note: This example provides a basic implementation. You might need to adjust the configuration based on your specific security requirements and network setup. You can further enhance the network policy by using specific ports, protocols, and other security measures as needed.,

質問 # 193
You have a microservice that is deployed in a Kubernetes cluster, and you want to monitor its performance and health using Prometheus and Grafan a. How can you configure Prometneus to scrape metrics from your microservice and create dashboards in Grafana?

正解：

解説：
See the solution below with Step by Step Explanation.

Explanation:
Solution (Step by Step) :
1. Enable Metrics in Your Microservice:
- Ensure your microservice exposes metrics through an HTTP endpoint using a library like Prometheus Client (for Java), Go metrics, or StatsD.
- Define metrics such as request count, latency, error rate, and other relevant performance indicators.
2. Deploy Prometheus:
- Deploy Prometheus using a 'Deployment and a 'Service'
- Configure Prometheus to scrape metrics trom the microservice by adding its endpoint to the 'scrape_configs' in the 'prometheus.yaml' file.

3. Create a Service for Prometheus to Access the Microservice: - Create a 'Service' of type 'ClusterlP' that exposes the microservice's metrics endpoint (usually port 9100). - Ensure Prometheus can reacn this service. 4. Deploy Grafana: - Deploy Gratana using a 'Deployment' and a 'Service' - Configure Grafana to connect to Prometheus as a data source. 5. Create Dashboards in Grafana: - Use Grafana's dashboard builder to create custom dashboards that visualize the metrics collected by Prometheus. - Add panels to display graphs, charts, and tables that show the performance and health ot your microservice. 6. Configure Alerts in Grafana: - Configure alerts in Grafana based on specific metrics and thresholds. - Set up notifications to alert you when critical issues arise with the microservice. Note: This approach provides comprehensive monitoring for your microservice. Prometheus scrapes metrics from the microservice, stores them in its time series database, and Grafana visualizes these metrics and provides alerts for potential issues. Example Prometheus Scrape Configuration:

```
scrape_configs:
- job_name: 'my-microservice'
  static_configs:
  - targets: ['my-microservice-service:9100']
  # Optional: Use a service discovery mechanism to automatically detect microservice pods
  # discovery:
  #   kubernetes_sd_configs:
  #   - role: service
  #     names: ['my-microservice']
```

Example Grafana Dashboard: - Create a dashboard with panels that show the following metrics: - Request count per minute - Average request latency - Error rate - CPU and memory usage of the microservice container - Set up alerts to notify you it: - The request count exceeds a certain threshold - The average latency exceeds a certain threshold - The error rate exceeds a certain threshold - The CPU or memory usage exceeds a certain threshold,

質問 # 194
.....

- Linux Foundation CKADをパスします: Linux Foundation Certified Kubernetes Application Developer Exam試験は有効的に検証するCKAD過去問題 □ ウェブサイト▷ www.goshiken.com◁から□ CKAD □を開いて検索し、無料でダウンロードしてくださいCKAD最速合格
- CKAD日本語対策 □ CKAD合格資料 □ CKAD資格認証攻略 □ ⇛ www.goshiken.com ⇚で⇛ CKAD ⇚を検索し、無料でダウンロードしてくださいCKAD日本語対策
- 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, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, project.gabus.lt, www.stes.tyc.edu.tw, zahitech.com, www.posteezy.com, www.stes.tyc.edu.tw, Disposable vapes

P.S. Pass4TestがGoogle Driveで共有している無料かつ新しいCKADダンプ：https://drive.google.com/open?id=1DzNN5FOI5M0Pl5kh4ETC5qWt5nNodVPw