CKAD復習過去問 & CKAD勉強の資料

さらに、Fast2test CKAD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=1P2CBrT0BDzyyTg0lvb9eEbSaB6yBR-l7

CKAD試験トレントの3つのバージョンを提供しており、PDFバージョン、PCバージョン、APPオンラインバージョンが含まれています。各バージョンの機能と使用方法は異なり、実際の状況に適した最も便利なバージョンを選択できます。たとえば、PDFバージョンは、CKADテストトレントをダウンロードして印刷するのに便利で、学習の閲覧に適しています。 PDFバージョンを使用している場合は、ペーパーで急流CKADガイドを印刷できます。 CKAD試験問題のPCバージョンは、Linux Foundation Certified Kubernetes Application Developer Exam実際の試験環境を刺激します。

Linux Foundation CKAD Exam Syllabus Topics:

Section	Weight	Objectives
Topic 1: Application Observability and Maintenance	15%	- Understand API deprecations - Implement probes and health checks - Monitor applications using CLI tools - Work with container logs - Debug applications in Kubernetes
Topic 2: Application Design and Build	20%	- Define, build and modify container images - Choose and use appropriate workload resources - Understand multi-container Pod design patterns - Utilize persistent and ephemeral volumes

Topic 3: Application Deployment	20%	- Use Helm package manager - Implement deployment strategies - Work with Kustomize - Manage Deployments, rolling updates and rollbacks
Topic 4: Services and Networking	20%	- Expose applications via Services - Troubleshoot network access - Use Ingress rules - Understand and apply NetworkPolicies
Topic 5: Application Environment, Configuration and Security	25%	- Use ServiceAccounts - Create and consume Secrets - Work with ConfigMaps - Apply application security settings - Understand authentication, authorization and admission control - Configure resource requirements, limits and quotas - Use custom resources and extensions

>> CKAD復習過去問 <<

CKAD更新される学習資料、有効なCKADpdf問題集、Linux Foundation Certified Kubernetes Application Developer Exam勉強資料

テストの気分が悪い場合は、毎回CKADのソフトテストエンジンまたはアプリテストエンジンを選択する必要があります。 これらの2つのバージョンには、実際のテストシーンをシミュレートする機能が1つあります。 時間指定試験を設定し、何度も練習することができます。 Linux Foundation CKADダンプトレントで試験のペースを感じ、テストする時間を確保できます。 あなたがしたいことをしなければならない時間と機会を利用すべきです。 CKADダンプトレントファイルを使用すると、テストの雰囲気を保つことができます。

Linux Foundation Certified Kubernetes Application Developer Exam 認定 CKAD 試験問題 (Q185-Q190):

質問 # 185
You are designing a microservice architecture where a frontend application needs to communicate with multiple backend services. The services are deployed as Pods in a Kubernetes cluster- To streamline communication and security, you decide to implement a sidecar proxy pattern Explain the benefits of using a sidecar proxy in this scenario and illustrate how you would implement it using a container image like Envoy Proxy.

正解：

解説：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Understand Sidecar Proxy Pattern:
- In the sidecar proxy pattern, a proxy container runs alongside your main application container within the same Pod.
- The proxy acts as an intermediary, handling network tratfic between your application and other services.
2. Benefits of Using a Sidecar Proxy:
- Traffic Management:
- Routing requests to different backend services.
- Load balancing across multiple instances ot a service.
- Security:
- Enforcing access control and authentication.
- Handling SSL termination.
- Observability:
- Monitoring and logging network traffic.
- Simplified Development:
- Separating networking concerns from application logic.
3. Implementing with Envoy Proxy:

- Choose Envoy Proxy:
- Envoy is a popular open-source proxy designed for high-performance network communication.
- Create a Deployment YAML:
- Define a Deployment for your application, including a main container for your application code and a sidecar container for Envoy Proxy.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app
        image: my-app-image:latest
        ports:
        - containerPort: 8080
      - name: envoy
        image: envoyproxy/envoy:v1.23.0
        ports:
        - containerPort: 9901
        command: ["/usr/local/bin/envoy", "-c", "/etc/envoy/envoy.yaml"]
        volumeMounts:
        - name: config-volume
          mountPath: /etc/envoy
      volumes:
      - name: config-volume
        configMap:
          name: envoy-config
```

4. Configure Envoy: - Create a ConfigMap: - Create a ConfigMap to hold the Envoy configuration (envoy-yaml). - Define the routes, listeners, and clusters for your services.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: envoy-config
data:
  envoy.yaml: |-
    static_resources:
      listeners:
      - name: listener_0
        address:
          socket_address:
            port_value: 9901
        filter_chains:
        - filters:
          - name: envoy.filters.network.http_connection_manager
            typed_config:
              "@type": type.googleapis.com/envoy.config.filter.network.http_connection_manager.v3.HttpConnectionManager
              stat_prefix: ingress_http
              route_config:
                name: local_route
                virtual_hosts:
                - name: my_app_host
                  domains: [""]
                  routes:
                  - match:
                      prefix: "/service1"
                    route:
                      cluster: service1_cluster
                  - match:
                      prefix: "/service2"
                    route:
                      cluster: service2_cluster
              http_filters:
              - name: envoy.filters.http.router
      clusters:
      - name: service1_cluster
        connect_timeout: 0.25s
        type: strict_dns
        lb_policy: ROUND_ROBIN
        hosts:
        - socket_address:
            address: service1
            port_value: 8080
      - name: service2_cluster
        connect_timeout: 0.25s
        type: strict_dns
        lb_policy: ROUND_ROBIN
        hosts:
        - socket_address:
            address: service2
            port_value: 8080
```

5. Deploy the Configuration: - Apply the Deployment and ConfigMap using 'kubectl apply -f deployment.yaml' and 'kubectl apply -f configmap.yaml' 6. Test tne Setup: - Access your trontend application from outside the cluster. - Verity that traffic is routed correctly to the backend services through the Envoy proxy. 7. Optional: Service Discovery: - For more dynamic environments, you can integrate Envoy with service discovery mechanisms like Kubernetes Service or Consul to automatically update its configuration based on service changes. This configuration assumes that your backend services are named "servicel " and "service2" with ports 8080. Adjust the configuration based on your specific services and their port numbers.,

質問 # 186
You are building a Kubernetes application tnat requires persistent storage for its dat a. The application needs to be able to access the data even if the pod is restarted or deleted. You have a PersistentVolumeClaim (PVC) defined for this purpose.

正解：

解説：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Create a PersistentVolume (PV):
- Define a PV with a suitable storage class, access modes (ReadWriteOnce), and a capacity that meets your application's storage requirements.
- Example:

```
apiVersion: v1
kind: PersistentVolume
metadata:
  name: my-pv
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: "/data/my-pv"
  persistentVolumeReclaimPolicy: Retain
```

2. Create a PersistentVolumeClaim (PVC): - Define a PVC with the desired storage class and access modes. - Specify the desired storage capacity. - Example:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: my-storage-class
```

3. Create a Deployment With the PVC: - In the Deployment YAML, define a volume mount that uses the PVC you created_ - Specify the volume mount path within the container. - Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-app
spec:
  replicas: 1
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
      - name: my-app-container
        image: my-app-image:latest
        volumeMounts:
        - name: my-data
          mountPath: /data
      volumes:
      - name: my-data
        persistentVolumeClaim:
          claimName: my-pvc
```

4. Create the Deployment: - Apply the Deployment YAML using 'kubectl apply -f my-app-deployment.yamr 5. Verify the Deployment - Check the status of the Deployment using 'kubectl get deployments my-app' - Verify that the Pod is running and using the PersistentVolumeClaim. - You can also check the pod's logs for confirmation that the data is stored in the mounted volume.

質問 # 187
You must connect to the correct host . Failure to do so may result in a zero score.
[candidate@base] $ ssh ckad00021
Task
Create a Cronjob named grep that executes a Pod running the following single container:
name: busybox
image: busybox:stable
command: ["grep", "-i", "nameserv
er", "/etc/resolv.conf"]
Configure the CronJob to:
* execute Once every 30 minutes
* keep 96 completed Job
* keep 192 failed Job
* never restart podsterminate pods after 8 seconds
Manually create and execute once job
named grep-test from the grep Cronjob

正解：

解説：
See the Explanation below for complete solution.
Explanation:
ssh ckad00021
Below is the clean, CKAD-friendly way (YAML + apply + verify + manual job).
1) Create the CronJob grep
Create a file (anywhere, e.g. in your home):
cat <<'EOF' > grep-cronjob.yaml
apiVersion: batch/v1
kind: CronJob
metadata:
name: grep
spec:
schedule: "*/30 * * * *"
successfulJobsHistoryLimit: 96
failedJobsHistoryLimit: 192
jobTemplate:
spec:
activeDeadlineSeconds: 8
template:

spec:
restartPolicy: Never
containers:
- name: busybox
image: busybox:stable
command: ["grep", "-i", "nameserver", "/etc/resolv.conf"]
EOF
Apply it:
kubectl apply -f grep-cronjob.yaml
Verify:
kubectl get cronjob grep
kubectl describe cronjob grep
Confirm the key fields quickly:
kubectl get cronjob grep -o jsonpath='{.spec.schedule} {"\n"} {.spec.successfulJobsHistoryLimit} {"\n"} {.spec.
failedJobsHistoryLimit} {"\n"}'
kubectl get cronjob grep -o jsonpath='{.spec.jobTemplate.spec.activeDeadlineSeconds} {"\n"} {.spec.
jobTemplate.spec.template.spec.restartPolicy} {"\n"}'
2) Manually create and execute the one-off Job grep-test from the CronJob Create the Job from the CronJob:
kubectl create job --from=cronjob/grep grep-test
Watch it:
kubectl get jobs grep-test
kubectl get pods -l job-name=grep-test
Get logs (most important proof):
POD=$(kubectl get pods -l job-name=grep-test -o jsonpath='{.items[0].metadata.name}') kubectl logs "$POD" You should see one or more nameserver ... lines from /etc/resolv.conf.

質問 # 188
You have a web application tnat requires a dedicated sidecar container to manage logging and monitoring. The sidecar container should be deployed alongside every pod of the application. You need to ensure that the sidecar container is always available alongside the application pods, even if the main application container ex;mences failures. Which Kubernetes resource is most suitable for this scenario and wny?

正解：

解説：
See the solution below with Step by Step Explanation.
Explanation:
Solution (Step by Step) :
1. Choose DaemonSet The most suitable Kubernetes resource for this scenario is a DaemonSet.
2. Daemonset Functionality: Daemonsets ensure that a pod is running on every node in your cluster. This is ideal tor sidecar containers because they need to be present alongside tne main application pod on each node.
3. Daemonset Benefits:
- Guaranteed Availability: Daemonsets guarantee that the sidecar container is always available on the same node as the main application pod, even if the application pod is restarted or fails.
- Pod Management: DaemonSets manage the lifecycle of the sidecar container, ensuring its availability and resource allocation.
- Node-Level Deployment: Daemonsets deploy pods on all nodes, ensuring consistent functionality across the cluster
4. Implementation Example:

```yaml
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: logging-sidecar
spec:
  selector:
    matchLabels:
      app: logging-sidecar
  template:
    metadata:
      labels:
        app: logging-sidecar
    spec:
      containers:
      - name: logging-sidecar
        image: your-logging-image:latest
        # ... other container configuration
      - name: your-application
        image: your-app-image:latest
        # ... other container configuration
```

This DaemonSet definition specifies a pod with two containers: the 'logging-sidecar' and 'your-application'. The Slogging-sidecar' is your sidecar container, and 'your-application' represents your main application. - Important: The Daemonset will ensure that a pod with these containers is deployed on every node of your Kubernetes cluster 5. Deployment and Monitoring: - Deployment: Use 'kubectl apply -f logging-sidecar.yamr to deploy the DaemonSet. - Monitoring: Observe the pods created by the Daemonset using 'kubectl get pods'. You should see a pod with the 'logging-sidecar and 'your- application' containers running on each node- 6. Conclusion: - Using a DaemonSet to manage your sidecar container ensures its consistent availability alongside the main application pods, guaranteeing logging and monitoring capabilities even in case of pod failures-,

質問 # 189
You have a Deployment running a web application built With a Node.js container. The application currently uses an older version of the Node.js runtime, and you need to upgrade to a newer versiom Describe the steps involved in modifying the container image to include the new Node.js runtime 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 node:16-alpine # Use the desired Node.js version
COPY --from=existing-image:latest /app /app
WORKDIR /app
CMD ["npm", "start"]
```

- Replace With the name of the existing Docker image used by your Deployment. - This Dockefflle uses a multi-stage build approach. It starts with a new Node.js base image and copies the application code from the existing image. This allows you to update the runtime without rebuilding the entire application. 2. Build the New Image: - Build tne image using the Dockerflle: 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-node-app
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-node-app
  template:
    metadata:
      labels:
        app: my-node-app
    spec:
      containers:
      - name: my-node-app
        image: updated-image:latest  # Use the new image name
        ports:
        - containerPort: 8080
      restartPolicy: Always
```

4. Apply the Changes: - Apply the updated Deployment using 'kubectl 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 updated Node.js version. 6. Test the Application: - Access your application and ensure it functions correctly with the new Node.js runtime.

質問 # 190

.....

当社Linux FoundationのCKADガイド急流は、過去の試験論文と業界での人気の傾向に基づいて、厳密な分析と要約を行っており、改訂および更新されています。 CKAD試験問題により、洗練された概念が簡素化されました。このソフトウェアは、さまざまな自己学習および自己評価機能を強化して、学習結果を確認します。 CKADテストトレントのソフトウェアは、統計レポート機能を提供し、学生が脆弱なリンクを見つけて対処するのに役立ちます。 CKAD試験問題のこのバージョンを使用すると、試験に簡単に合格することができます。

CKAD勉強の資料: https://jp.fast2test.com/CKAD-premium-file.html

- 100% パスレートCKAD復習過去問 - 認定試験のリーダー - 現実的なCKAD勉強の資料 □ 【 www.goshiken.com 】で➽ CKAD □を検索して、無料で簡単にダウンロードできますCKAD最新知識
- CKAD受験対策 □ CKAD資格認定 □ CKAD日本語 □ □ www.goshiken.com □サイトにて➤ CKAD □問題集を無料で使おうCKAD最新知識
- CKAD認定資格試験 □ CKAD日本語版対策ガイド □ CKAD受験料 □ 【 www.mogiexam.com 】を開き、➤ CKAD □を入力して、無料でダウンロードしてくださいCKAD的中問題集
- CKAD日本語版復習資料 □ CKADコンポーネント □ CKAD認定資格試験 □ ➽ www.goshiken.com □で「 CKAD 」を検索して、無料で簡単にダウンロードできますCKAD認定資格試験
- CKAD試験問題解説集 □ CKAD受験対策 □ CKAD日本語版問題解説 □ ➤ www.goshiken.com □には無料の□ CKAD □問題集がありますCKAD日本語版対策ガイド
- 高品質CKAD復習過去問 - 正確的なLinux Foundation 認定トレーリング - 正確的なLinux Foundation Linux Foundation Certified Kubernetes Application Developer Exam □ Open Webサイト《 www.goshiken.com 》検索➠ CKAD □無料ダウンロードCKAD受験料
- Linux Foundation CKAD復習過去問 - www.topexam.jp - 資格試験のリーダー □ ▹ www.topexam.jp ◃から簡単に▹ CKAD ◃を無料でダウンロードできますCKAD日本語版問題解説
- 有効的なCKAD復習過去問 - 合格スムーズCKAD勉強の資料 | 一番優秀なCKAD前提条件 Linux Foundation Certified Kubernetes Application Developer Exam □ { www.goshiken.com }で使える無料オンライン版⇒ CKAD ⇐の試験問題CKAD認定資格試験
- CKAD試験復習赤本 □ CKAD日本語版問題解説 □ CKAD日本語 □ 時間限定無料で使える（ CKAD ）の試験問題は“ www.it-passports.com ”サイトで検索CKAD受験記
- 効率的なCKAD復習過去問一回合格-権威のあるCKAD勉強の資料 ※ ⇛ CKAD ⇚を無料でダウンロード“ www.goshiken.com ”で検索するだけCKAD試験参考書
- CKAD日本語 □ CKAD受験対策 □ CKAD日本語版対策ガイド □ □ www.shikenpass.com □を開き、□ CKAD □を入力して、無料でダウンロードしてくださいCKAD試験復習赤本
- www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, www.stes.tyc.edu.tw, 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, Disposable vapes

ちなみに、Fast2test CKADの一部をクラウドストレージからダウンロードできます：https://drive.google.com/open?id=1P2CBrT0BDzyyTg0lvb9eEbSaB6yBR-l7