LinuxFoundation.CKA.v2024-12-19.q62

試験コード:	CKA
試験名称:	Certified Kubernetes Administrator (CKA) Program Exam
認定資格:	Linux Foundation
無料問題数:	62
バージョン:	v2024-12-19
アクセス数:	506
ページビュー数:	620
https://www.jpnpdf.com/LinuxFoundation.CKA.v2024-12-19.q62-mondaishu.html	

最新問題: **1**
「web」名前空間内のすべてのポッドとポッドからのトラフィックを許可する
ラベル「type=monitoring」を持つポッドをラベル「app: db」に一致させる
A. kubectl 名前空間 web を作成します
kubectl ラベル namespace/web app=web
vim web-allow-all-ns-monitoring.yaml
APIバージョン: networking.k8s.io/v1
種類: ネットワークポリシー
メタデータ:
名前: web-allow-all-ns-monitoring
名前空間: デフォルト
仕様:
ポッドセレクター:
一致ラベル:
アプリ: db
進入:
- から :
- 名前空間セレクター:
一致ラベル:
アプリ: ウェブ
ポッドセレクター:
一致ラベル:
タイプ: 監視
k kubectl apply -f web-allow-all-ns-monitoring.yaml
B. kubectl 名前空間 web を作成します
kubectl ラベル namespace/web app=web
vim web-allow-all-ns-monitoring.yaml

APIバージョン: networking.k8s.io/v1
種類: ネットワークポリシー
メタデータ:
名前: web-allow-all-ns-monitoring
名前空間: デフォルト
仕様:
ポッドセレクター:
ポッドセレクター:
一致ラベル:
タイプ: 監視
k kubectl apply -f web-allow-all-ns-monitoring.yaml
Answer: A (メーッセージを残す**)**

最新問題**: 2**
「myservice」というサービスが作成されるのを待機する init コンテナを持つポッドを作成します。init コンテナが完了すると、myapp コンテナが起動し、「アプリが実行中です」というメッセージを出力して 3600 秒間スリープします。
A. vim マルチコンテナ pod.yaml
APIバージョン: v1
種類: ポッド
メタデータ:
名前: myapp-pod
ラベル:
アプリ: マイアプリ
仕様:
コンテナ:
- 名前: myapp-container
画像: ビジーボックス:1.28
コマンド: ['sh', '-c', 'echo アプリは実行中です! && sleep
3600フィート]
コンテナの初期化:
- 名前: init-myservice
画像: ビジーボックス:1.28
コマンド: ['sh', '-c', "until nslookup myservice.$(cat
/var/run/secrets/kubernetes.io/serviceaccount/namespace).s
vc.cluster.local; echo を実行して、myservice を待機します。2 秒間スリープします。
終わり"]
// 「myservice」というサービスが存在するかどうかを確認します
kubectl でサービスを取得する
注意: 「myservice」というサービスが起動しない場合はポッドは起動しません。

存在する。
// 次にポッドを作成します
kubectl apply -f マルチコンテナポッド.yaml
B. vim マルチコンテナ pod.yaml
APIバージョン: v1
種類: ポッド
メタデータ:
名前: myapp-pod
ラベル:
アプリ: マイアプリ
仕様:
コンテナ:
- 名前: myapp-container
画像: ビジーボックス:1.28
コマンド: ['sh', '-c', 'echo アプリは実行中です! && sleep
3600フィート]
コンテナの初期化:
- 名前: init-myservice
終わり"]
// 「myservice」というサービスが存在するかどうかを確認します
kubectl でサービスを取得する
注意: 「myservice」というサービスが起動しない場合はポッドは起動しません。
存在する。
// 次にポッドを作成します
kubectl apply -f マルチコンテナポッド.yaml
Answer: A (メーッセージを残す**)**

最新問題**: 3**
3 つのコンテナを含むポッドを作成しますか? (マルチコンテナ)
A. イメージ=nginx、イメージ=redis、イメージ=consul
nginxコンテナの名前を 「nginx-container」にします
redisコンテナの名前を 「redis-container」にします
Consul コンテナの名前を 「consul-container」にします。
コンテナのポッドマニフェストファイルを作成し、コンテナを追加する
残りの画像のセクション
kubectl マルチコンテナを実行します --generator=run-pod/v1 --image=nginx --
ドライラン -o yaml > マルチコンテナ.yaml
＃ それから
vim マルチコンテナ.yaml
APIバージョン: v1

種類: ポッド
メタデータ:
ラベル:
実行: マルチコンテナ
名前: マルチコンテナ
仕様:
コンテナ:
- 画像: nginx
名前: nginx-コンテナ
- 画像: redis
名前: redis コンテナ
- 画像: 領事
名前: 領事コンテナ
再起動ポリシー: 常に
B. イメージ=nginx、イメージ=redis、イメージ=consul
nginxコンテナの名前を「nginx-container」にします
redisコンテナの名前を「redis-container」にします
Consul コンテナの名前を「consul-container」にします。
コンテナのポッドマニフェストファイルを作成し、コンテナを追加する
残りの画像のセクション
kubectl マルチコンテナを実行します --generator=run-pod/v1 --image=nginx --
ドライラン -o yaml > マルチコンテナ.yaml
それから
vim マルチコンテナ.yaml
ラベル:
実行: マルチコンテナ
名前: マルチコンテナ
仕様:
コンテナ:
- 画像: nginx
名前: nginx-コンテナ
- 画像: redis
名前: 領事コンテナ
再起動ポリシー: 常に
Answer: (解答を表示する**)**

最新問題**: 4**
「development」という名前空間と、この名前空間に nginx というイメージ nginx を含むポッドを作成します。
Answer:

kubectl 名前空間開発を作成 kubectl nginx を実行 --image=nginx --restart=Never -n development

最新問題**: 5**
環境変数をvar1=value1としてポッドを作成します。ポッド内の環境変数を確認します
Answer:
kubectl run nginx --image=nginx --restart=Never --env=var1=value1
＃ それから
kubectl exec -it nginx --env を実行します。
＃ または
kubectl exec -it nginx --sh -c 'echo $var1'
＃ または
kubectl describe po nginx | grep 値1

最新問題**: 6**
スコア: 7%
タスク
バージョン 1.20.0 を実行している既存の Kubernetes クラスターがある場合は、マスターノード上のすべての Kubernetes コントロール プレーンとノード コンポーネントをバージョン 1.20.1 にアップグレードします。
アップグレードする前に必ずマスター ノードをドレインし、アップグレード後に遮断を解除してください。
マスターノード上の kubelet と kubectl もアップグレードする必要があります。
Answer:
解決：
[学生@node-1] > ssh ek8s
kubectl コルドン k8s-マスター
kubectl で k8s-master をドレインし、--delete-local-data、--ignore-daemonsets、--force apt-get install kubeadm=1.20.1-00 kubelet=1.20.1-00 kubectl=1.20.1-00 --disableexcludes=kubernetes を実行します。kubeadm でアップグレードを実行し、1.20.1 を適用します。--etcd-upgrade=false を実行します。systemctl daemon-reload、systemctl restart kubelet、kubectl uncordon k8s-master を実行します。

最新問題**: 7**
展開履歴を確認する
Answer:
kubectlロールアウト履歴デプロイメントWebアプリ

最新問題**: 8**

以前ラボで使用されていた Palo Alto Networks ファイアウォールの初期構成をロードするために、Windows ワークステーションを使用してブートストラップ USB フラッシュ ドライブが準備されました。USB フラッシュ ドライブはファイル システム FAT32 を使用してフォーマットされ、初期構成は init-cfg.txt という名前のファイルに保存されています。ファイアウォールは現在 PAN-OS 10.0 を実行しており、ラボ構成を使用しています。USB フラッシュ ドライブの init-cgf.txt の内容は次のとおりです。
タイプ=dhcpクライアント
IPアドレス
デフォルトゲートウェイ
ネットマスク
IPv6アドレス
IPv6デフォルトゲートウェイ
ホスト名=Ca-FW-DC1
パノラマサーバー=10.5.107.20
パノラマサーバー2=10.5.107.21
tplname=FINANCE_TG4
dgname=財務dg
dnsプライマリ=10.5.6.6
dnsセカンダリ=10.5.6.7
op-command-modes-multi-vsys.jumbo-frame
dhcp-send-hostname=はい
dhcp-send-client-id=はい
DHCP 受け入れサーバホスト名 = yes
DHCP 受け入れサーバードメイン = はい
USB フラッシュ ドライブがファイアウォールの USB ポートに挿入され、コマンド > request restart system を使用してファイアウォールが再起動されました。再起動後、ファイアウォールはブートストラップ プロセスを開始できません。この失敗の原因は次のとおりです。
A. ファイアウォールは工場出荷時のデフォルト状態になっているか、ブートストラップのためにすべてのプライベートデータが削除されている必要があります。
B. PAN-OS のバージョンは最低でも 9.1 x である必要がありますが、ファイアウォールは 10.0x を実行しています
C. ブートストラップXMLファイルは必須ファイルですが、見つかりません
D. USBはexi3ファイルシステムでフォーマットされている必要があります。FAT32は
E. ホスト名は必須パラメータですが、init-cfg.txt にありません。
Answer: (解答を表示する**)**

最新問題**: 9**
シークレットを環境変数として読み込むnginxポッドを作成する
A. // ymlファイルを作成する

kubectl run nginx --image=nginx --restart=Never --dry-run -o
yaml > nginx.yml
// 以下にenvセクションを追加して作成
vim nginx.yaml
実行: nginx
名前: nginx
仕様:
コンテナ:
- 画像: nginx
名前: nginx
envFrom:
- シークレットRef:
名前: 私の秘密
再起動ポリシー: なし
kubectl を適用 -f nginx.yaml
//確認する
kubectl exec -it nginx - env
B. // ymlファイルを作成する
kubectl run nginx --image=nginx --restart=Never --dry-run -o
yaml > nginx.yml
// 以下にenvセクションを追加して作成
vim nginx.yaml
APIバージョン: v1
種類: ポッド
メタデータ:
ラベル:
実行: nginx
名前: nginx
仕様:
コンテナ:
- 画像: nginx
名前: nginx
envFrom:
- シークレットRef:
名前: 私の秘密
再起動ポリシー: なし
kubectl を適用 -f nginx.yaml
//確認する
kubectl exec -it nginx - env
Answer: B (メーッセージを残す**)**

最新問題**: 10**
スコア: 7%
タスク
既存のデプロイメント フロントエンドを再構成し、既存のコンテナ nginx のポート 80/tcp を公開する http という名前のポート仕様を追加します。
コンテナ ポート http を公開する front-end-svc という名前の新しいサービスを作成します。
新しいサービスを設定して、個々の Pod がスケジュールされているノード上の NodePort 経由でも公開されるようにします。
Answer:
解決：
kubectl get デプロイ フロントエンド
kubectl edit deploy フロントエンド -o yaml
#http という名前のポート指定
#サービス.yaml
APIバージョン: v1
種類: サービス
メタデータ:
名前: フロントエンドサービス
ラベル:
アプリ: nginx
仕様:
ポート:
- ポート: 80
プロトコル: tcp
名前: http
セレクタ：
アプリ: nginx
タイプ: NodePort
kubectl create -f サービス.yaml
kubectl でサービスを取得します
http という名前のポート指定
kubectl 公開デプロイメント フロントエンド --name=front-end-svc --port=80 --tarport=80 --type=NodePort

最新問題**: 11**
「frontend」という名前のポッドログを一覧表示し、「started」というパターンを検索して、ファイル「/opt/error-logs」に書き込みます。
Answer:

Kubectl ログ フロントエンド | grep -i "started" > /opt/error-logs

最新問題: **12**
次のタスクを実行します。
仕様ファイルで定義された hungry-bear に init コンテナを追加します。
/opt/KUCC00108/sub-spec-KUC
initコンテナは/workdir/calm.txtを作成する必要があります。
/workdir/calm.txtがない場合
スペックファイルが定義されたら、ポッドを作成する必要があります。
Answer:
以下の解決策を参照してください。
説明
解決
F:\Work\データ入力 Work\データ入力\20200827\CKA\4 B.JPG
F:\Work\データ入力 Work\データ入力\20200827\CKA\4 C.JPG
F:\Work\データ入力 Work\データ入力\20200827\CKA\4 D.JPG

最新問題: **13**
準備ができているノードをチェックし、それをファイル /opt/nodestatus に出力します。
A. JSONPATH='{範囲 .items[*]}{@.metadata.name}:{範囲
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \
//確認する
cat /opt/ノードステータス
B. JSONPATH='{範囲 .items[*]}{@.metadata.name}:{範囲
@.status.conditions[*]}{@.type}={@.status};{end}{end}' \
&& kubectl get nodes -o jsonpath="$JSONPATH" | grep
「準備完了=True」 > /opt/node-status
//確認する
cat /opt/ノードステータス
Answer: (解答を表示する**)**

最新問題: **14**
describeコマンドを使用せずにポッド内のイメージバージョンを確認する
Answer:
kubectl get po nginx -o
jsonpath='{.spec.containers[].image}{"\n"}'

最新問題: **15**
容量順に並べられたすべての永続ボリュームを一覧表示し、kubectlの出力全体を保存

/opt/KUCC00102/volume_list。出力のソートには kubectl 独自の機能を使用し、それ以上操作しないでください。
Answer:
以下の解決策を参照してください。
説明
解決
F:\Work\データ入力 Work\データ入力\20200827\CKA\2 C.JPG

最新問題: **16**
容量順に並べられたすべての永続ボリュームを一覧表示し、fullkubectloutputを保存します。
/opt/KUCC00102/volume_list。出力をソートするためにkubectl独自の機能を使用し、それ以上操作しないでください。
Answer:
以下の解決策を参照してください。
説明
解決

最新問題: **17**
以前のバージョンで展開を元に戻し、検証する
すべて大丈夫です
Answer:
kubectl ロールアウト ウェブアプリのデプロイを元に戻す kubectl ロールアウト ステータス ウェブアプリのデプロイ kubectl ポッドの取得

最新問題: **18**
デプロイメントを5つのレプリカに拡張する
Answer:
kubectl scale deploy webapp -replicas=5 // kubectl get deploy を検証 kubectl get po,rs

最新問題: **19**

基本的な WildFire サービスの一部としてサポートされているファイル タイプのアップロードは何ですか?
A. PE
B. エルフ
C. VBS
D. バット
Answer: A (メーッセージを残す**)**

最新問題**: 20**
展開のロールアウトを一時停止する
Answer:
kubectl ロールアウト 一時停止 ウェブアプリのデプロイ

最新問題**: 21**
wk8s-node-0 という名前の Kubernetes ワーカー ノードが NotReady 状態です。この原因を調査し、適切な手順を実行してノードを Ready 状態にし、変更が永続的になるようにします。
次のコマンドを使用して、障害が発生したノードに SSH 接続できます。
[学生@node-1] $ | ssh Wk8s-node-0
次のコマンドを使用して、ノード上で昇格された権限を取得できます。
[学生@w8ks-node-0] $ | sudo -i
Answer:
以下の解決策を参照してください。
説明
解決

最新問題**: 22**
コマンド「env」を実行し、出力を「envpod」ファイルに保存するbusyboxポッドを作成します。
Answer:
以下の解決策を参照してください。
説明
kubectl run busybox --image=busybox --restart=Never --rm -it --env > envpod.yaml を実行します。

最新問題**: 23**
「hello world」をエコーして終了するポッドを作成します。完了したらポッドを自動的に削除します。
Answer:
以下の解決策を参照してください。

説明
kubectl 実行 busybox --image=busybox -it --rm --restart=Never --
/bin/sh -c 'エコーハローワールド'
kubectl get po # 「busybox」という名前のポッドは表示されないはずです

最新問題**: 24**
ポッドの IP アドレスを取得します - "nginx-dev"
Answer:
Kubect1 は幅を広げます
JsonPathの使用
kubect1 get pods -o=jsonpath='{範囲
.items[*]}{.metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'

最新問題**: 25**
このデプロイメントに自動スケーリングを適用し、最小 10 個、最大 20 個のレプリカと 85% のターゲット CPU で hpa が作成され、レプリカが 1 から 10 に増加したことを確認します。
Answer:
kubectl autoscale deploy webapp --min=10 --max=20 --cpu percent=85 kubectl get hpa
kubectl get pod -l app=webapp

最新問題**: 26**
次のようにポッドをスケジュールします。
名前: nginx-kusc00101
画像: nginx
ノードセレクター: disk=ssd
Answer:
以下の解決策を参照してください。
説明
解決
F:\Work\データ入力 Work\データ入力\20200827\CKA\6 B.JPG
F:\Work\データ入力 Work\データ入力\20200827\CKA\6 C.JPG
F:\Work\データ入力 Work\データ入力\20200827\CKA\6 D.JPG

最新問題**: 27**
次のようにポッドを作成します。
名前: 非永続的 Redis
コンテナイメージ: redis
名前が cache-control のボリューム
マウントパス: /data/redis

ポッドはステージング名前空間で起動する必要があり、ボリュームは永続的であってはなりません。
Answer:
解決

最新問題**: 28**
ファイルを作成します:
名前空間開発でサービス baz を実装するすべてのポッドをリストする /opt/KUCC00302/kucc00302.txt。
ファイルの形式は、1 行に 1 つのポッド名である必要があります。
Answer:
解決

最新問題**: 29**
「Hello I'm running job」とエコーする画像busyboxを使用して、「hello-job」という名前のジョブを作成します。
A. kubectl ジョブ hello-job --image=busybox --dry-run -o yaml を作成します
-- echo "こんにちは、ジョブを実行しています" > hello-job.yaml
kubectl 作成 -f hello-job.yaml
//ジョブの検証
kubectl ゲットポー
kubectl ログ hello-job-*
B. kubectl ジョブ hello-job --image=busybox --dry-run -o yaml を作成します
-- echo "こんにちは、ジョブを実行しています" > hello-job.yaml
kubectl 作成 -f hello-job.yaml
//ジョブの検証
kubectl ジョブを取得する
kubectl ゲットポー
kubectl ログ hello-job-*
Answer: (解答を表示する**)**

最新問題**: 30**
リテラル値を持つmyconfigmapというconfigmapを作成します。
アプリ名=myapp
A. kubectl create cm myconfigmap --from-literal=appname=myapp
// 確認する
kubectl で cm -o yaml を取得します
（または）
kubectl 記述 cm
B. kubectl create cm myconfigmap --from-literal=appname=myapp

// 確認する
または）
kubectl 記述 cm
Answer: A (メーッセージを残す**)**

最新問題**: 31**
ポッド ラベル name=cpu-utilizer から、CPU ワークロードの高いポッドを見つけ、最も多くの CPU を消費しているポッドの名前をファイル /opt/KUTR00102/KUTR00102.txt (既に存在) に書き込みます。
Answer:
以下の解決策を参照してください。
説明
解決
F:\Work\データ入力 Work\データ入力\20200827\CKA\16 B.JPG
F:\Work\データ入力 Work\データ入力\20200827\CKA\16 C.JPG

最新問題**: 32**
エンジニアリング名前空間に env=test というラベルの nginx ポッドを作成します。
Answer:
kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml > nginx-pod.yaml kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml | kubectl create -n engineering -f - YAML ファイル:
APIバージョン: v1
種類: ポッド
メタデータ:
名前: nginx
名前空間: エンジニアリング
ラベル:
環境: テスト
仕様:
コンテナ:

- 名前: nginx
画像: nginx
イメージプルポリシー: IfNotPresent
再起動ポリシー: なし
kubectl 作成 -f nginx-pod.yaml

最新問題**: 33**
ポッドセレクターアプリを使用して、ポート 32767 の nginx ポッドで NodePort タイプとしてサービスを作成します: my-nginx
Answer:
kubectl run nginx --image=nginx --restart=Never --labels=app=nginx --port=80 --dry-run -o yaml > nginx-pod.yaml

最新問題**: 34**
ポッドの1つのラベルをenv=uatに変更し、検証するポッドをすべてリストします。
Answer:
kubectl ラベル pod/nginx-dev3 env=uat --overwrite kubectl ポッドを取得 --show-labels

最新問題**: 35**
describeコマンドを使用せずにポッド内のイメージバージョンを確認する
Answer:
以下の解決策を参照してください。
説明
kubectl get po nginx -o
jsonpath='{.spec.containers[].image}{"\n"}'

最新問題**: 36**
すべての名前空間内のすべてのポッドのリストを取得し、それをファイル「/opt/pods-list.yaml」に書き込みます。
Answer:
kubectl get po -all-namespaces > /opt/pods-list.yaml

最新問題**: 37**
kubeadm を使用して、マスター 1 つとワーカー 1 つを含む Kubernetes クラスターをインストールする
A. これは簡単な質問です。マスター 1 つとワーカー 1 つを含む kubeadm を使用して Kubernetes クラスターをインストールする必要があります。
マスターとワーカーの両方がインストールに成功したとみなされます。
ノードが利用可能になります。
参照: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/

B. これは簡単な質問です。マスター 1 つとワーカー 1 つを含む kubeadm を使用して Kubernetes クラスターをインストールする必要があります。
参照: https://kubernetes.io/docs/setup/production-environment/tools/kubeadm/
Answer: A (メーッセージを残す**)**

最新問題**: 38**
ノード「worker-2」に「NoSchedule」の効果を持つ汚染を追加し、
汚染効果を持つノードを「NoSchedule」としてリストします
A. // ノード「worker-2」に汚染を追加する
kubectl taint ノード ワーカー 2 キー=値:NoSchedule
.items[*]}{.metadata.name} {.spec.taints[?(
@.effect=='NoSchedule')].effect}{\"\n\"}{end}" | awk 'NF==2
{$0 を印刷}'
B. // ノード「worker-2」に汚染を追加する
kubectl taint ノード ワーカー 2 キー=値:NoSchedule
// 確認する
// 「custom-columns」を使用すると、どの列を表示するかをカスタマイズできます。
印刷される
kubectl get nodes -o customcolumns=NAME:.metadata.name,TAINTS:.spec.taints --no-headers
// jsonpath を使用する
kubectl get nodes -o jsonpath="{範囲
.items[*]}{.metadata.name} {.spec.taints[?(
@.effect=='NoSchedule')].effect}{\"\n\"}{end}" | awk 'NF==2
{$0 を印刷}'
Answer: (解答を表示する**)**

最新問題**: 39**
スコア: 4%
タスク
ek8s-node-1 という名前のノードを使用不可に設定し、そのノード上で実行されているすべてのポッドを再スケジュールします。
Answer:
解決：
[学生@node-1] > ssh ek8s
kubectl コルドン ek8s-node-1
kubectl ドレイン ek8s-node-1 --delete-local-data --ignore-daemonsets --force

最新問題**: 40**

サービス front-end-service を作成して構成し、NodePort 経由でアクセスできるようにして、front-end という名前の既存のポッドにルーティングします。
Answer:
解決

最新問題**: 41**
環境変数をvar1=value1としてポッドを作成します。ポッド内の環境変数を確認します
Answer:
以下の解決策を参照してください。
説明
kubectl run nginx --image=nginx --restart=Never --env=var1=value1
＃ それから
kubectl exec -it nginx --env を実行します。
＃ または
kubectl exec -it nginx --sh -c 'echo $var1'
＃ または
kubectl describe po nginx | grep 値1

最新問題**: 42**
スコア: 5%
タスク
ポッド ラベル name=cpu-utilizer から、CPU ワークロードの高いポッドを見つけ、最も多くの CPU を消費しているポッドの名前をファイル /opt/KUTR00401/KUTR00401.txt (既に存在) に書き込みます。
Answer:
解決：
kubectl top -l 名前=CPUユーザー -A
echo 'ポッド名' >> /opt/KUT00401/KUT00401.txt

最新問題**: 43**
展開のロールアウトステータスを取得する
Answer:
kubectl ロールアウト ステータス デプロイ Web アプリ

最新問題**: 44**
以前のバージョン1.17.1への展開を元に戻し、イメージが以前のバージョンであることを確認します。
Answer:
kubectl rollout undo deploy webapp kubectl describe deploy webapp | grep イメージ

最新問題**: 45**
スコア: 7%
タスク
まず、https://127.0.0.1:2379 で実行されている既存の etcd インスタンスのスナップショットを作成し、そのスナップショットを /srv/data/etcd-snapshot.db に保存します。
次に、/var/lib/backup/etcd-snapshot-previo us.dbにある既存の以前のスナップショットを復元します。
Answer:
解決：
#バックアップ
ETCDCTL_API=3 etcdctl --endpoints="https://127.0.0.1:2379" --cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --key=/opt/KUIN000601/etcd-client.key スナップショットを保存し、/etc/data/etcd-snapshot.db に保存します。
#復元する
ETCDCTL_API=3 etcdctl --endpoints="https://127.0.0.1:2379" --cacert=/opt/KUIN000601/ca.crt --cert=/opt/KUIN000601/etcd-client.crt --key=/opt/KUIN000601/etcd-client.key スナップショットを復元 /var/lib/backup/etcd-snapshot-previoys.db

最新問題**: 46**
ストレージ10Gi、アクセスモードReadWriteOnce、ストレージクラス名manual、ボリューム/mnt/dataでtask-pv-volumeという名前のhostPath PersistentVolumeを作成し、検証します。
A. vim タスク-pvボリューム.yaml
APIバージョン: v1
種類: 永続ボリューム
メタデータ:
名前: タスクPVボリューム
ラベル:
タイプ: ローカル
仕様:
ストレージクラス名: ""
容量：
ストレージ: 5Gi
アクセスモード:
- 一度だけ読み書き可能
ホストパス:
パス: "/mnt/data"
kubectl を適用 -f タスク pv ボリューム.yaml

//確認する
kubectl PV を取得する
名前 容量 アクセス
モード クレーム ポリシー ステータス クレーム
ストレージクラス理由年齢
タスクPVボリューム5GiRWO
利用可能な状態を維持する
3秒
B. vim タスク-pvボリューム.yaml
APIバージョン: v1
種類: 永続ボリューム
メタデータ:
名前: タスクPVボリューム
ラベル:
タイプ: ローカル
仕様:
ストレージクラス名: ""
容量 :
ストレージ: 5Gi
アクセスモード:
- 一度だけ読み書き可能
ホストパス:
パス: "/mnt/data"
kubectl を適用 -f タスク pv ボリューム.yaml
//確認する
kubectl PV を取得する
名前 容量 アクセス
モード クレーム ポリシー ステータス クレーム
ストレージクラス理由年齢
タスクPVボリューム4GiRWO
利用可能な状態を維持する
8秒
Answer: A (メーッセージを残す**)**

最新問題: 47
スコア:7%
タスク
新しいPersistentVolumeClaimを作成する
* 名前: pv-volume
* クラス: csi-hostpath-sc
* 容量: 10マイル
PersistentVolumeClaim をボリュームとしてマウントする新しい Pod を作成します。
* 名前: ウェブサーバー
* 画像: nginx
* マウントパス: /usr/share/nginx/html
新しい Pod がボリュームに対して ReadWriteOnce アクセスを持つように設定します。
最後に、kubectl edit または kubectl patch を使用して、PersistentVolumeClaim を 70Mi の容量まで拡張し、その変更を記録します。
Answer:
解決：
vi pvc.yaml
ストレージクラス PVC
APIバージョン: v1
種類: PersistentVolumeClaim
メタデータ:
名前: pv-ボリューム
仕様:
アクセスモード:
- 一度だけ読み書き可能
ボリュームモード: ファイルシステム
リソース：
リクエスト:
ストレージ: 10Mi
ストレージクラス名: csi-hostpath-sc
vi pod-pvc.yaml
APIバージョン: v1
種類: ポッド
メタデータ:
名前: ウェブサーバー
仕様:
コンテナ:

- 名前: ウェブサーバー
画像: nginx
ボリュームマウント:
- マウントパス: "/usr/share/nginx/html"
名前: my-volume
ボリューム:
- 名前: my-volume
永続ボリュームクレーム:
クレーム名: pv-volume
作成
kubectl 作成 -f pod-pvc.yaml
＃編集
kubectl 編集 pvc pv-ボリューム --record

最新問題**: 48**
エンジニアリング名前空間に env=test というラベルの nginx ポッドを作成します。
Answer:
以下の解決策を参照してください。
説明
kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml > nginx-pod.yaml kubectl run nginx --image=nginx --restart=Never --labels=env=test --namespace=engineering --dry-run -o yaml | kubectl create -nengineering-f - YAML ファイル:
APIバージョン: v1
種類: ポッド
メタデータ:
名前: nginx
名前空間: エンジニアリング
ラベル:
環境: テスト
仕様:
コンテナ:
- 名前: nginx
画像: nginx
イメージプルポリシー: IfNotPresent
再起動ポリシー: なし
kubectl 作成 -f nginx-pod.yaml

最新問題**: 49**
次のようにデプロイメントを作成します。

* 名前: nginx-app
* バージョン 1.11.10-alpine のコンテナ nginx を使用する
* デプロイメントには3つのレプリカが含まれている必要があります
次に、ローリング アップデートを実行して、新しいバージョン 1.11.13-alpine でアプリケーションをデプロイします。
最後に、その更新を以前のバージョン 1.11.10-alpine にロールバックします。
Answer:
以下の解決策を参照してください。
説明
解決

最新問題: **50**
5つのnginxポッドを作成し、そのうち2つにはenv=prodというラベルを付け、
そのうち3つはenv=devとラベル付けされています
A. kubectl run nginx-dev1 --image=nginx --restart=Never --
ラベル=env=dev
kubectl run nginx-dev2 --image=nginx --restart=Never --
ラベル=env=dev
kubectl run nginx-prod1 --image=nginx --restart=Never --
ラベル=env=prod
kubectl run nginx-prod2 --image=nginx --restart=Never --
ラベル=env=prod
B. kubectl run nginx-dev1 --image=nginx --restart=Never --
ラベル=env=dev
kubectl run nginx-dev2 --image=nginx --restart=Never --
ラベル=env=dev
kubectl run nginx-dev3 --image=nginx --restart=Never --
ラベル=env=dev
kubectl run nginx-prod1 --image=nginx --restart=Never --
ラベル=env=prod
kubectl run nginx-prod2 --image=nginx --restart=Never --
ラベル=env=prod
Answer: B (メーッセージを残す**)**

最新問題: **51**
デプロイメントを1つのレプリカにスケールダウンする
Answer:
kubectl scale deploy webapp -replicas=1 // kubectl get deploy を検証 kubectl get po,rs

最新問題: **52**

コンテナ ポート 80 で nginx ポッドを作成します。このポッドはトラフィックのみを受信し、ポート 80 のエンドポイント / をチェックしてポッドを検証および削除します。
A. kubectl run nginx --image=nginx --restart=Never --port=80 --
ドライラン -o yaml > nginx-pod.yaml
// readinessProbeセクションを追加して作成
vim nginx-pod.yaml
実行: nginx
名前: nginx
仕様:
コンテナ:
- 画像: nginx
名前: nginx
ポート:
- コンテナポート: 60
準備プローブ:
httpGet:
パス : /
ポート: 60
再起動ポリシー: なし
kubectl を適用 -f nginx-pod.yaml
// 確認する
kubectl でポッド nginx を記述します | grep -i 準備状況
kubectl po nginx を削除します
B. kubectl run nginx --image=nginx --restart=Never --port=80 --
ドライラン -o yaml > nginx-pod.yaml
// readinessProbeセクションを追加して作成
vim nginx-pod.yaml
APIバージョン: v1
種類: ポッド
メタデータ:
ラベル:
実行: nginx
名前: nginx
仕様:
コンテナ:
- 画像: nginx
名前: nginx
ポート:
- コンテナポート: 80
準備プローブ:

httpGet:
パス : /
ポート: 80
再起動ポリシー: なし
kubectl を適用 -f nginx-pod.yaml
// 確認する
kubectl でポッド nginx を記述します | grep -i 準備状況
kubectl po nginx を削除します
Answer: (解答を表示する**)**

最新問題**: 53**
ファイルを作成します:
名前空間開発でサービスを実装するすべてのポッドをリストす
る /opt/KUCC00302/kucc00302.txt。
ファイルの形式は、1 行に 1 つのポッド名である必要があります。
Answer:
以下の解決策を参照してください。
説明
解決

最新問題**: 54**
次のように Kubernetes シークレットを作成します。
名前: 超秘密
パスワード: bob
redis イメージを使用して、pod-secrets-via-file という名前のポッドを作成します。このポッドは、/secrets に super-secret という名前のシークレットをマウントします。
redisイメージを使用して、pod-secrets-via-envという名前の2番目のポッドを作成し、パスワードをCONFIDENTIALとしてエクスポートします。
Answer:
解決

最新問題**: 55**
ポッドの IP アドレスを取得します - "nginx-dev"
Answer:
以下の解決策を参照してください。
説明
Kubect1 は幅を広げます
JsonPathの使用
kubect1 get pods -o=jsonpath='{範囲
アイテム[*]}{.metadata.name}{"\t"}{.status.podIP}{"\n"}{end}'

最新問題**: 56**
次のデプロイメント仕様ファイルを作成します。
app_runtime_stage=dev というラベルの nginx イメージのレプリカを 7 つ起動します。デプロイメント名: kual00201 この仕様ファイルのコピー
を /opt/KUAL00201/spec_deployment.yaml (または /opt/KUAL00201/spec_deployment.json) に保存します。
完了したら、このタスク中に作成した新しい Kubernetes API オブジェクトをクリーンアップ (削除) します。
Answer:
解決

最新問題**: 57**
「development」という名前空間と、この名前空間に nginx というイメージ nginx を含むポッドを作成します。
Answer:
kubectl 名前空間の作成開発
kubectl run nginx --image=nginx --restart=Never -n 開発

最新問題**: 58**
名前が app-data、容量が 2Gi、アクセス モードが ReadWriteMany の永続ボリュームを作成します。ボリュームのタイプは hostPath で、場所は /srv/app-data です。
Answer:
解決
永続ボリューム
永続ボリュームは、Kubernetes クラスター内のストレージの一部です。永続ボリュームは、ノードのようなクラスター レベルのリソースであり、どの名前空間にも属しません。管理者によってプロビジョニングされ、特定のファイル サイズを持ちます。これにより、Kubernetes にアプリケーションをデプロイする開発者は、基盤となるインフラストラクチャを知る必要がありません。開発者がアプリケーション用に一定量の永続ストレージを必要とする場合、システム管理者は、プロビジョニングされた永続ボリュームを簡単に使用できるようにクラスターを構成します。
永続ボリュームの作成
kind: PersistentVolume apiVersion: v1 metadata: name:app-data spec: capacity: # 作成する PV の容量を定義します storage: 2Gi # 要求するストレージの量 accessModes: # 作成するボリュームの権限を定義します - ReadWriteMany hostPath: path: "/srv/app-data" # ボリュームを作成するパス 課題 アクセス モードが ReadWriteMany、ストレージ クラス名が shared、ストレージ容量が 2Gi、ホスト パスが /srv/app-data の、app-data という名前の永続ボリュームを作成します。
2. ファイルを保存し、永続ボリュームを作成します。

3. 永続ボリュームを表示します。
永続ボリュームのステータスは 「使用可能」です。つまり、使用可能であり、まだマウントされていません。persistentVolume を persistentVolumeClaim にマウントすると、このステータスは変わります。
永続ボリュームクレーム
実際のエコシステムでは、システム管理者が PersistentVolume を作成し、開発者がポッドで参照される PersistentVolumeClaim を作成します。PersistentVolumeClaim は、persistentVolume に必要な最小サイズとアクセス モードを指定して作成されます。
チャレンジ
上記で作成した永続ボリュームを要求する永続ボリューム要求を作成します。要求は 2Gi を要求する必要があります。永続ボリューム要求の storageClassName が、以前に作成した persistentVolume と同じであることを確認します。
種類: PersistentVolume apiVersion: v1 メタデータ: name:app-data
仕様:
accessModes: - ReadWriteManyリソース:
リクエスト: ストレージ: 2Gi
ストレージクラス名: 共有
2. 保存してPVCを作成する
njerry191@cloudshell:~ (extreme-clone-2654111)$ kubect1 create -f app-data.yaml
persistentvolumeclaim/app-data が作成されました
3. PVCを見る
4. 最初に作成した PV で何が変わったか確認してみましょう。
ステータスが 「利用可能」から 「バインド済み」に変更されました。
5. パス /var/app/config を使用して永続ボリューム要求をマウントするために使用される、イメージ nginx を使用して myapp という名前の新しいポッドを作成します。
主張を展開する
apiVersion: v1 種類: Pod メタデータ: creationTimestamp: null 名前: app-data 仕様: ボリューム: - name:congigpvc persistenVolumeClaim: claimName: app-data コンテナー: - イメージ: nginx 名前: app volumeMounts: - mountPath: "/srv/app-data " 名前: configpvc

最新問題**: 59**
2つのレプリカを持つ 「myapp」という名前のデプロイメントを作成します。
nginx イメージを作成し、 「myservice」という名前のサービスとしてデプロイメントを公開します。
A. // YAMLテンプレートを作成する
kubectl create deploy myapp --image=nginx --dry-run -o yaml >
マイアプリ.yaml
//myapp.yaml ファイルで replicas=2 を更新します
APIバージョン: アプリ/v1

種類: デプロイメント
メタデータ:
ラベル:
アプリ: マイアプリ
名前: myapp
仕様:
レプリカ: 2
セレクタ :
一致ラベル:
アプリ: マイアプリ
テンプレート :
メタデータ:
ラベル:
アプリ: マイアプリ
仕様:
コンテナ:
- 画像: nginx
名前: nginx
// デプロイメントを作成する
kubectl 作成 -f myapp.yaml
// サービス用の YAML テンプレートを作成
kubectl 公開デプロイメント myapp --type=ClusterIP --port=80 --
ターゲットポート=80 --name=myservice --dry-run -o yaml >
マイサービス.yaml
YAML ファイル:
APIバージョン: v1
種類: サービス
メタデータ:
ラベル:
アプリ: マイアプリ
名前: myservice
仕様:
ポート:
- ポート: 80
プロトコル: TCP
ターゲットポート: 80
セレクタ :
アプリ: マイアプリ
タイプ: ClusterIP
kubectl でサービスを取得する

名前 タイプ クラスター IP 外部 IP ポート
年
kubernetes ClusterIP 10.2.0.1 <なし> 443/TCP
158日
myservice ClusterIP 10.2.96.175 <なし> 80/TCP
40代
B. // YAMLテンプレートを作成する
kubectl create deploy myapp --image=nginx --dry-run -o yaml >
マイアプリ.yaml
//myapp.yaml ファイルで replicas=2 を更新します
APIバージョン: アプリ/v1
種類: デプロイメント
メタデータ:
ラベル:
アプリ: マイアプリ
名前: myapp
仕様:
レプリカ: 2
セレクタ :
一致ラベル:
アプリ: マイアプリ
テンプレート :
メタデータ:
ラベル:
アプリ: マイアプリ
仕様:
コンテナ:
- 画像: nginx
名前: nginx
// デプロイメントを作成する
kubectl 作成 -f myapp.yaml
// サービス用の YAML テンプレートを作成
kubectl 公開デプロイメント myapp --type=ClusterIP --port=60 --
ターゲットポート=60 --name=myservice --dry-run -o yaml >
マイサービス.yaml
YAML ファイル:
APIバージョン: v1
種類: サービス
メタデータ:
ラベル:

アプリ: マイアプリ
名前: myservice
仕様:
ポート:
- ポート: 60
プロトコル: TCP
ターゲットポート: 80
セレクタ :
アプリ: マイアプリ
タイプ: ClusterIP
kubectl でサービスを取得する
名前 タイプ クラスター IP 外部 IP ポート
年
kubernetes ClusterIP 10.2.0.1 <なし> 443/TCP
158日
myservice ClusterIP 10.2.96.175 <なし> 80/TCP
40代
Answer: A (メーッセージを残す**)**

最新問題**: 60**
Kubernetesワーカーノード。なぜそうなるのか調査してください。
適切な手順を実行してノードを適切な状態にし、変更が永続的であることを確認します。
障害が発生したノードに ssh するには、次の操作を実行します。
[学生@ノード-1]$ | sshWk8s-ノード-0
次のコマンドを使用して、ノード上で昇格された権限を取得できます。
[学生@w8ks-node-0] $ |sudo -i
Answer:
以下の解決策を参照してください。
説明
解決

最新問題**: 61**
cfgvolumeというconfigmapを作成し、値var1=val1を設定します。
var2=val2 でボリュームnginx-volumeを持つnginxポッドを作成します。
このconfigmap cfgvolumeからデータを読み取り、パスに配置します
/etc/cfg
A. // まずconfigmap cfgvolumeを作成します
kubectl は cm cfgvolume を作成します --from-literal=var1=val1 --from-literal=var2=val2
// configmap を検証する
kubectl で cm cfgvolume を記述します

// 設定マップを作成する
kubectl 作成 -f nginx-volume.yml
vim nginx-configmap-pod.yaml
APIバージョン: v1
種類: ポッド
メタデータ:
ラベル:
実行: nginx
名前: nginx
仕様:
ボリューム:
- 名前: nginx-volume
構成マップ:
名前: cfgvolume
コンテナ:
- 画像: nginx
名前: nginx
ボリュームマウント:
- 名前: nginx-volume
マウントパス: /etc/cfg
再起動ポリシー: 常に
kubectl を適用 -f nginx-configmap-pod.yaml
/ // 確認する
// ポッドにexecする
kubectl exec -it nginx -- /bin/sh
// パスを確認する
/etc/cfg をコピーする
B. // まずconfigmap cfgvolumeを作成します
kubectl は cm cfgvolume を作成します --from-literal=var1=val1 --from-literal=var2=val2
// configmap を検証する
kubectl で cm cfgvolume を記述します
// 設定マップを作成する
kubectl 作成 -f nginx-volume.yml
vim nginx-configmap-pod.yaml
APIバージョン: v1
種類: ポッド
- 名前: nginx-volume
構成マップ:
名前: cfgvolume
コンテナ:

- 画像: nginx
名前: nginx
ボリュームマウント:
- 名前: nginx-volume
マウントパス: /etc/cfg
再起動ポリシー: 常に
kubectl を適用 -f nginx-configmap-pod.yaml
/ // 確認する
// ポッドにexecする
kubectl exec -it nginx -- /bin/sh
// パスを確認する
/etc/cfg をコピーする
Answer: A (メーッセージを残す**)**

最新問題: **62**
「frontend」という名前のポッドログを一覧表示し、「started」というパターンを検索して、ファイル「/opt/error-logs」に書き込みます。
Answer:
以下の解決策を参照してください。
説明
Kubectl ログ フロントエンド | grep -i "started" > /opt/error-logs