CKAD Practice Exam: Core Concepts Solution Guide

Intermediate 20m Bookmark

Check 1: Create and Configure a Basic Pod Potential Solution

```
# Create a Pod in the cre Namespace with the following configuration: the
Pod is named basic, the Pod uses the nginx:stable-alpine-perl image for
its only container, restart the Pod only OnFailure, ensure port 80 is
open to TCP traffic
kubectl run -n cre --image=nginx:stable-alpine-perl --restart=OnFailure -
-port=80 basic
```

Commentary

The above solution uses the `kubectl run` command to create the Pod. The options for configuring the pod can be seen by using tab completion (the host runs `source<(kubectl completion bash)` automatically to enable completions but the command is always available in the (kubectl Cheat Sheet) or consulting the help output of `kubectl run --help`. The help output also has examples that can be combined to construct the required command. Ports default to referring to the TCP protocol.

Alternatively, you can achieve the result using a Pod manifest file. To generate a manifest file as a starting point, you could:

- Copy in a manifest from the official documentation. There are many examples to choose from. Try to choose one that is going to have all the fields you need.
- Use a dry-run, e.g. `kubectl -n cre run --image=nginx:stable-alpine-perl basic --dry-run=client -o yaml` The --

This option can be quite tedious for Pods due to their being hundreds of lines of output.

For the manifest file options you would need to edit the file using a command-line editor, such as `vim`. In the real exam, you could also use the exam's notepad function to edit the manifest before pasting it into a file in the exam terminal (for example entering vim manifest.yaml to open an empty file, pressing *i* to enter insert mode, pasting the text, pressing escape, and finally entering *:wq* to write the file and quit). Practicing with `vim` is highly recommended before taking the exam.

To know what fields to add to the manifest, you could look at the documentation examples, manifests of existing pods, or output from `kubectl explain` (for example `kubectl explain pod.spec`). Using manifest files is necessary sometimess but for basic tasks, adding options to the `kubectl run` command with the help of tab completions will probably be fastest, which is very important in the exam.

Content to review

- Introduction to Kubernetes - Pods
- Kubernetes Pod Design for Application Developers: Definition Basics - Reviewing Pod Definition Basics

Suggested documentation bookmark(s)

- Configure Pods and Containers
- kubectl Cheat Sheet

Check 2: Create Namespace and Launch Pod with Labels

```
# Create a new Namespace named workers and then create a Pod within it
using the following configuration: the Pod is named worker, the Pod uses
the busybox image for its only container, the Pod has the labels
compa                                                              -c
"echo
kube
kube
```

Commentary

The above solution uses the `kubectl create` command to first create the `workers` Namespace. The `worker` Pod is then launched within this namespace with the required configuration. The options for configuring the pod can be seen by using tab completion (the host runs `source<(kubectl completion bash)` automatically to enable completions but the command is always available in the kubectl Cheat Sheet) or consulting the help output of `kubectl run --help`. The help output also has examples that can be combined to construct the required command.

Content to review

- Namespaces
- Introduction to Kubernetes - Pods
- Kubernetes Pod Design for Application Developers: Definition Basics - Reviewing Pod Definition Basics

Suggested documentation bookmark(s)

- Configure Pods and Containers
- kubectl Cheat Sheet

Check 3: Update the Label on a Running Pod

```
# Edit and save pod – this will update the pod without restarting it
kubectl edit pod –n ca200 compiler
```

Commentary

The above solution uses the `kubectl edit` command to edit the `compiler` Pod. The edit command will open the Pods manifest within an editor (vim) view allowing you to

from l es

Content to review

- Managing Resources
- kubectl edit

Suggested documentation bookmark(s)

- Configure Pods and Containers
- kubectl Cheat Sheet

Check 4: Get the Pod IP Address using JSONPath

```
# Step 1 – examine the json structure for the pod in question, in
particular look to see where the pod assigned ip address is located – it
is in the .status.podIP field
kubectl get pods –n ca300 ip–podzoid –o json
# Step 2 – using the information from the previous step, build out the
actual jsonpath based expression to return only the pod ip address
kubectl get pods –n ca300 ip–podzoid –o jsonpath={.status.podIP}
# Step 3 – write out the working kubectl command to the file
/home/ubuntu/podip.sh
echo "kubectl get pods –n ca300 ip–podzoid –o jsonpath={.status.podIP}" >
/home/ubuntu/podip.sh
```

Commentary

The above solution uses the `–o jsonpath` parameter with a jsonpath expression which pulls out the pod's IP address. JSONPath can be used to build custom queries to pull out specific data attributes of a manifest for any type of resource deployed into the cluster. Knowing how to use JSONPath can be useful to discover and query facts about deployed resources.

Content to review

- JSONPath Support

Sugge

Check 5: Generate Pod YAML Manifest File

```
# Use kubectl run command specifically with the --dry-run=client and -o
yaml parameters - and then redirect the output to file
kubectl run -n core-system borg1 --image=busybox --restart=Always --
labels="platform=prod" --env system=borg -o yaml --dry-run=client --
/bin/sh -c "echo borg.running... && sleep 3600" > /home/ubuntu/pod.yaml
```

Commentary

The above solution uses the imperative `kubectl run` command specifically with the `--dry-run=client` and `-o yaml` parameters to generate the Pod manifest yaml. The resulting YAML output is then redirected to the `/home/ubuntu/pod.yaml` file as requested.

Content to review

- Managing Kubernetes Objects using Imperative Commands

Suggested documentation bookmark(s)

- kubectl Cheat Sheet

Check 6: Launch Pod and Configure it's Termination Shutdown Time

```
# Step 1 - Use kubectl run command specifically with the --dry-run=client
and -o yaml parameters - and then redirect the output to file
kubectl run web-zeroshutdown -n sys2 --image=nginx --restart=Never --
port=80 -o yaml --dry-run=client > pod-zeroshutdown.yaml
# Step 2 - use kubectl explain pod.spec to check the syntax and how the
terminationGracePeriodSeconds attribute should be configured
kubectl explain pod.spec
# Step 3 - use vim and edit and update the pod-zeroshutdown.yaml file,
adding in -> terminationGracePeriodSeconds: 0
apiVersion: v1
kind: Pod
metadata:
  cr
  lab
    {
  nam
```

```
    - image: nginx
      name: web-zeroshutdown
      ports:
      - containerPort: 80
      resources: {}
  dnsPolicy: ClusterFirst
  restartPolicy: Never
status: {}
# Step 4 - use kubectl apply -f pod-zeroshutdown.yaml to create the pod
resource within the cluster
kubectl apply -f pod-zeroshutdown.yaml
```

Commentary

The above solution uses the imperative `kubectl run` command specifically with the `--dry-run=client` and `-o yaml` parameters to generate the Pod manifest yaml. The resulting YAML output is then redirected to the `pod-zeroshutdown.yaml` file as requested. The imperative `kubectl run` command uses as many of the available configuration parameters possible to establish the manifest YAML file, note - the `terminationGracePeriodSeconds` value is **not** available as a configuration parameter - therefore this needs to be edited into the manifest file manually using an editor such as Vim.

The `kubectl explain pod.spec` command is used to check the syntax, location, and how the `terminationGracePeriodSeconds` attribute is expected to be configured. The Vim editor is then used to edit and update the `pod-zeroshutdown.yaml` file with the `terminationGracePeriodSeconds` attribute set to `0` in the correct location (directly beneath `spec:`). With the edit completed, the `pod-zeroshutdown.yaml` file is saved. Finally, the Pod resource is created within the cluster using the `kubectl apply -f pod-zeroshutdown.yaml` command.

Content to review

- Managing Kubernetes Objects using Imperative Commands
- Kubectl Operations (explain)

Sugge

Mark it or miss it!
Make sure to mark this content as completed; otherwise, it will not be displayed as such.

Mark as completed

Did you like this Resource? **Report an issue**

About the author

Cloud Academy
Instructor

Students	Labs	Courses	Learning paths
2,617	83	9	18

Digital skills are built at the intersection of knowledge, experience, and context. The fundamental building blocks of the training templates in our Library meet teams wherever they are along the cloud maturity curve, imparting the knowledge and experience needed to take them to the next level. Our training platform is intuitive and scalable. Most importantly, all training is easily customizable, which enables organizations to provide context and guidance for teams of any size. Teams leveraging Cloud Academy hit the ground running.

Covered topics

Deployment Compute DevOps Kubernetes

About Cloud Academy

About QA

About Circus Street

COMMUNITY

Join Discord Channel

HELP

Help Center