CKAD Practice Exam: Observability Solution Guide

Intermediate 20m 5/5 Bookmark

Check 1: Potential Solution

```
# Solution commands
kubectl run nginx -n ca1 --image=nginx --restart=Never --port=80 --dry-
run=client -o yaml > pod-livenessprobe.yaml

# Edit and save the file using vim
vim pod-livenessprobe.yaml

apiVersion: v1
kind: Pod
metadata:
 creationTimestamp: null
 labels:
   run: nginx
 name: nginx
 namespace: ca1
spec:
 containers:
 - image: nginx
   name: nginx
   ports:
   - containerPort: 80
   resources: {}
   livenessProbe:
     httpGet:
       path: /
       port: 80
     initialDelaySeconds: 10
     periodSeconds: 5
 dnsPolicy: ClusterFirst
 restartPolicy: Never
status: {}

# Apply the manifest to create the Pod within the cluster
kubectl apply -f pod-livenessprobe.yaml
```

output to a file named `pod-livenessprobe.yaml`. Open this file using vim (terminal editor) and update the file to now include the livenessProbe configuration as seen above. Save the file. Finally create the Pod resource within the cluster using the `kubectl apply` command.

Content to review

- Kubernetes Observability: Monitoring, and Debugging - Using Probes to Better Understand Pod Health

Suggested documentation bookmark(s)

- Configure Liveness, Readiness and Startup Probes

Check 2: Potential Solution

```
# Solution commands
cat << EOF > patch.yaml
spec:
  template:
    spec:
      containers:
      - name: web2
        readinessProbe:
          httpGet:
            port: 80
EOF
kubectl -n hosting patch deployments web2 --patch "$(cat patch.yaml)"
```

Commentary

To start diagnosing the problem, list all the Services in the Namespace by entering:

```
kubectl -n hosting get service -o wide
```

The output `wide` format includes the Service selectors which could be an issue if they don't match the labels on their respective Pods. Both Services are of type N the specifi an be

```
kubectl –n hosting get endpoints
```

The **web2** Service has no endpoints which means it cannot serve any requests. The selector for web2 is app=web2. List all of the corresponding Pods with:

```
kubectl –n hosting get pods ––selector app=web2
```

There are two Pods matching the selector so there isn't an issue with the labels. The naming convention of the Pods also suggests they. are part of a Deployment. The problem is that. the Pods are not in the ready state. Describe the Pods to try to understand why:

```
kubectl –n hosting describe pods ––selector app=web2
```

The **Events** section explains that a **Readiness probe failed**. Recall that Services do not serve traffic to Pods that are not in the ready state. In the **Containers** section, the **Readiness** probe is summarized as follows:

Readiness: http-get http://:30/ delay=3s timeout=1s period=3s #success=1 #failure=3

The port the request is being sent to is **30**, however, the port the container is listening on is **80** (shown a few lines above in the `describe` output). The ReadinessProbe needs to be changed to send the request to port 80 and the Pods should then reach the ready state.

In the exam, you should edit the Deployment using `kubectl edit` and change `port: 30` to `port: 80`. The solution above uses a `patch` which is equivalent but more time-consuming to construct. It is used for the solution because it allows the solution to be written as one cohesive command and doesn't depend on specifics of a command-line editor, such as `vim` (used by `kubectl edit`

returned. Because the actual exam is performance-based, you can usually check your solutions are working. However, you should be careful about spending too much time checking to make sure you have time left to attempt all the questions. Practice enough before the exam to be confident enough to not have to check everything you do.

Content to review

- Kubernetes Observability: Monitoring, and Debugging - Using Probes to Better Understand Pod Health

Suggested documentation bookmark(s)

- Configure Liveness, Readiness and Startup Probes

Check 3: Potential Solution

```
# Solution commands
kubectl logs -n ca2 -l app=prod | wc -l > /home/ubuntu/combined-row-
count-prod.txt
```

Commentary

Use the command `kubectl logs` in the `ca2` namespace to combine all pod logs with label `app=prod`. Pipe the combined log file through the `wc -l` command to get a row count for the combined pod log, and then finally redirect the result out to the file `/home/ubuntu/combined-row-count-prod.txt`.

Content to review

- Logging Architecture

Suggested documentation bookmark(s)

- K

```
# Solution commands
kubectl exec -n ca2 skynet -- cat /skynet/t2-specs.txt > /home/ubuntu/t2-
specs.txt
```

Commentary

Use the command `kubectl exec` on the pod named `skynet` in the `ca2` namespace to execute the `cat` command to output the contents of the `/skynet/t2-specs.txt` pod file - and then finally redirect this output to the file `/home/ubuntu/t2-specs.txt`.

Content to review

- Get a Shell to a Running Container

Suggested documentation bookmark(s)

- Kubernetes Cheatsheet (exec)

Check 5: Potential Solution

```
# Solution commands
kubectl top pods -n matrix --sort-by=cpu --no-headers=true | head -n1 |
cut -d" " -f1 > /home/ubuntu/max-cpu-podname.txt
```

Commentary

Use the `kubectl top` command in the `matrix` namespace, sorting the results by CPU using the `--sort-by=cpu` parameter. The `--no-headers=true` parameter can be added to remove the header row from the results. The remaining results are then piped through the head -n1 command to grab just the top row. Finally the pod name is extracted by piping the previous result through the cut command, which is configured to split on whitespace. The pod na
the /h

Suggested documentation bookmark(s)

- Kubernetes Cheatsheet (top)

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.

Cover

ABOUT US

About Cloud Academy

About QA

About Circus Street

COMMUNITY

Join Discord Channel

HELP

Help Center