CKAD Simulator Preview Kubernetes 1.24

https://killer.sh

This is a preview of the full CKAD Simulator course content.

The full course contains 22 questions and scenarios which cover all the CKAD areas. The course also provides a browser terminal which is a very close replica of the original one. This is great to get used and comfortable before the real exam. After the test session (120 minutes), or if you stop it early, you'll get access to all questions and their detailed solutions. You'll have 36 hours cluster access in total which means even after the session, once you have the solutions, you can still play around.

The following preview will give you an idea of what the full course will provide. These preview questions are not part of the 22 in the full course but in addition to it. But the preview questions are part of the same CKAD simulation environment which we setup for you, so with access to the full course you can solve these too.

The answers provided here assume that you did run the initial terminal setup suggestions as provided in the tips section, but especially:

```
alias k=kubectl

export do="--dry-run=client -o yaml"
```

These questions can be solved in the test environment provided through the CKA Simulator

Preview Question 1

In *Namespace* `pluto` there is a *Deployment* named `project-23-api`. It has been working okay for a while but Team Pluto needs it to be more reliable. Implement a liveness-probe which checks the container to be reachable on port 80. Initially the probe should wait *10*, periodically *15* seconds.

The original *Deployment* yaml is available at `/opt/course/p1/project-23-api.yaml`. Save your changes at `/opt/course/p1/project-23-api-new.yaml` and apply the changes.

Answer

First we get an overview:

```
➜ k -n pluto get all -o wide
NAME                                 READY   STATUS    ... IP           ...
pod/holy-api                         1/1     Running   ... 10.12.0.26   ...
pod/project-23-api-784857f54c-dx6h6  1/1     Running   ... 10.12.2.15   ...
pod/project-23-api-784857f54c-sj8df  1/1     Running   ... 10.12.1.18   ...
pod/project-23-api-784857f54c-t4xmh  1/1     Running   ... 10.12.0.23   ...

NAME                              READY   UP-TO-DATE   AVAILABLE   ...
deployment.apps/project-23-api    3/3     3            3           ...
```

To note: we see another *Pod* here called `holy-api` which is part of another section. This is often the case in the provided scenarios, so be careful to only manipulate the resources you need to. Just like in the real world and in the exam.

Next we use `nginx:alpine` and `curl` to check if one *Pod* is accessible on port 80:

```
➜ k run tmp --restart=Never --rm -i --image=nginx:alpine -- curl -m 5 10.12.2.15
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

We could also use `busybox` and `wget` for this:

```
➜ k run tmp --restart=Never --rm --image=busybox -i -- wget -O- 10.12.2.15
Connecting to 10.12.2.15 (10.12.2.15:80)
writing to stdout
-                    100% |********************************|   612  0:00:00 ETA
written to stdout
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
```

Now that we're sure the *Deployment* works we can continue with altering the provided yaml:

```
cp /opt/course/p1/project-23-api.yaml /opt/course/p1/project-23-api-new.yaml
vim /opt/course/p1/project-23-api-new.yaml
```

Add the liveness-probe to the yaml:

```
# /opt/course/p1/project-23-api-new.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: project-23-api
  namespace: pluto
spec:
  replicas: 3
  selector:
    matchLabels:
      app: project-23-api
  template:
    metadata:
      labels:
        app: project-23-api
    spec:
      volumes:
      - name: cache-volume1
        emptyDir: {}
      - name: cache-volume2
        emptyDir: {}
      - name: cache-volume3
        emptyDir: {}
      containers:
      - image: httpd:2.4-alpine
        name: httpd
        volumeMounts:
        - mountPath: /cache1
          name: cache-volume1
        - mountPath: /cache2
          name: cache-volume2
        - mountPath: /cache3
          name: cache-volume3
        env:
        - name: APP_ENV
          value: "prod"
        - name: APP_SECRET_N1
          value: "IO=a4L/XkRdvN8jM=Y+"
        - name: APP_SECRET_P1
          value: "-7PAO_Z]>{pwa43r)__"
        livenessProbe:                  # add
          tcpSocket:                    # add
            port: 80                    # add
          initialDelaySeconds: 10       # add
          periodSeconds: 15             # add
```

Then let's apply the changes:

```
k -f /opt/course/p1/project-23-api-new.yaml apply
```

Next we wait 10 seconds and confirm the *Pods* are still running:

```
➜ k -n pluto get pod
NAME                              READY   STATUS    RESTARTS   AGE
holy-api                          1/1     Running   0          144m
project-23-api-5b4579fd49-8knh8   1/1     Running   0          90s
project-23-api-5b4579fd49-cbgph   1/1     Running   0          88s
project-23-api-5b4579fd49-tcfq5   1/1     Running   0          86s
```

We can also check the configured liveness-probe settings on a *Pod* or the *Deployment*:

```
➜ k -n pluto describe pod project-23-api-5b4579fd49-8knh8 | grep Liveness
    Liveness:   tcp-socket :80 delay=10s timeout=1s period=15s #success=1 #failure=3

➜  k -n pluto describe deploy project-23-api | grep Liveness
    Liveness:   tcp-socket :80 delay=10s timeout=1s period=15s #success=1 #failure=3
```

Preview Question 2

Team Sun needs a new *Deployment* named `sunny` with 4 replicas of image `nginx:1.17.3-alpine` in *Namespace* `sun`. The *Deployment* and its *Pods* should use the existing *ServiceAccount* `sa-sun-deploy`.

Expose the *Deployment* internally using a ClusterIP *Service* named `sun-srv` on port 9999. The nginx containers should run as default on port 80. The management of Team Sun would like to execute a command to check that all *Pods* are running on occasion. Write that command into file `/opt/course/p2/sunny_status_command.sh`. The command should use `kubectl`.

Answer

```
k -n sun create deployment -h #help

# check the export on the very top of this document so we can use $do
k -n sun create deployment sunny --image=nginx:1.17.3-alpine $do > p2_sunny.yaml

vim p2_sunny.yaml
```

Then alter its yaml to include the requirements:

```
# p2_sunny.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  creationTimestamp: null
  labels:
    app: sunny
  name: sunny
  namespace: sun
spec:
  replicas: 4                                # change
  selector:
    matchLabels:
      app: sunny
  strategy: {}
  template:
    metadata:
      creationTimestamp: null
      labels:
        app: sunny
    spec:
      serviceAccountName: sa-sun-deploy      # add
      containers:
      - image: nginx:1.17.3-alpine
        name: nginx
        resources: {}
status: {}
```

Now create the yaml and confirm its running:

```
➜ k create -f p2_sunny.yaml
deployment.apps/sunny created

➜ k -n sun get pod
NAME                     READY   STATUS        RESTARTS   AGE
0509649a                 1/1     Running       0          149m
0509649b                 1/1     Running       0          149m
1428721e                 1/1     Running       0          149m
...
sunny-64df8dbdbb-9mxbw   1/1     Running       0          10s
sunny-64df8dbdbb-mp5cf   1/1     Running       0          10s
sunny-64df8dbdbb-pggdf   1/1     Running       0          6s
sunny-64df8dbdbb-zvqth   1/1     Running       0          7s
```

Confirmed, the AGE column is always in important information about if changes were applied. Next we expose the *Pods* by created the *Service*:

```
k -n sun expose -h # help
k -n sun expose deployment sunny --name sun-srv --port 9999 --target-port 80
```

Using expose instead of `kubectl create service clusterip` is faster because it already sets the correct selector-labels. The previous command would produce this yaml:

```
# k -n sun expose deployment sunny --name sun-srv --port 9999 --target-port 80
apiVersion: v1
kind: Service
metadata:
  creationTimestamp: null
  labels:
    app: sunny
  name: sun-srv         # required by task
spec:
  ports:
  - port: 9999          # service port
    protocol: TCP
    targetPort: 80      # target port
  selector:
    app: sunny          # selector is important
status:
  loadBalancer: {}
```

Let's test the *Service* using `wget` from a temporary *Pod*:

```
➜ k run tmp --restart=Never --rm -i --image=nginx:alpine -- curl -m 5 sun-srv.sun:9999
Connecting to sun-srv.sun:9999 (10.23.253.120:9999)
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
...
```

Because the *Service* is in a different *Namespace* as our temporary *Pod*, it is reachable using the names `sun-srv.sun` or fully: `sun-srv.sun.svc.cluster.local`.

Finally we need a command which can be executed to check if all *Pods* are runing, this can be done with:

```
vim /opt/course/p2/sunny_status_command.sh
```

```
# /opt/course/p2/sunny_status_command.sh
kubectl -n sun get deployment sunny
```

To run the command:

```
➜ sh /opt/course/p2/sunny_status_command.sh
NAME    READY   UP-TO-DATE   AVAILABLE   AGE
sunny   4/4     4            4           13m
```

Preview Question 3

Management of EarthAG recorded that one of their *Services* stopped working. Dirk, the administrator, left already for the long weekend. All the information they could give you is that it was located in *Namespace* `earth` and that it stopped working after the latest rollout. All *Services* of EarthAG should be reachable from inside the cluster.

Find the *Service*, fix any issues and confirm its working again. Write the reason of the error into file `/opt/course/p3/ticket-654.txt` so Dirk knows what the issue was.

Answer

First we get an overview of the resources in *Namespace* `earth`:

```
➜ k -n earth get all
NAME                                        READY   STATUS    RESTARTS   AGE
pod/earth-2x3-api-584df69757-ngnwp          1/1     Running   0          116m
pod/earth-2x3-api-584df69757-ps8cs          1/1     Running   0          116m
pod/earth-2x3-api-584df69757-ww9q8          1/1     Running   0          116m
pod/earth-2x3-web-85c5b7986c-48vjt          1/1     Running   0          116m
pod/earth-2x3-web-85c5b7986c-6mqmb          1/1     Running   0          116m
pod/earth-2x3-web-85c5b7986c-6vjll          1/1     Running   0          116m
pod/earth-2x3-web-85c5b7986c-fnkbp          1/1     Running   0          116m
pod/earth-2x3-web-85c5b7986c-pjm5m          1/1     Running   0          116m
pod/earth-2x3-web-85c5b7986c-pwfvj          1/1     Running   0          116m
pod/earth-3cc-runner-6cb6cc6974-8wm5x       1/1     Running   0          116m
pod/earth-3cc-runner-6cb6cc6974-9fx8b       1/1     Running   0          116m
pod/earth-3cc-runner-6cb6cc6974-b9nrv       1/1     Running   0          116m
pod/earth-3cc-runner-heavy-6bf876f46d-b47vq 1/1     Running   0          116m
pod/earth-3cc-runner-heavy-6bf876f46d-mrzqd 1/1     Running   0          116m
pod/earth-3cc-runner-heavy-6bf876f46d-qkd74 1/1     Running   0          116m
pod/earth-3cc-web-6bfdf8b848-f74cj          0/1     Running   0          116m
pod/earth-3cc-web-6bfdf8b848-n4z7z          0/1     Running   0          116m
pod/earth-3cc-web-6bfdf8b848-rcmxs          0/1     Running   0          116m
pod/earth-3cc-web-6bfdf8b848-xl467          0/1     Running   0          116m


NAME                       TYPE        CLUSTER-IP     EXTERNAL-IP   PORT(S)    AGE
service/earth-2x3-api-svc  ClusterIP   10.3.241.242   <none>        4546/TCP   116m
service/earth-2x3-web-svc  ClusterIP   10.3.250.247   <none>        4545/TCP   116m
service/earth-3cc-web      ClusterIP   10.3.243.24    <none>        6363/TCP   116m


NAME                                   READY   UP-TO-DATE   AVAILABLE   AGE
deployment.apps/earth-2x3-api          3/3     3            3           116m
deployment.apps/earth-2x3-web          6/6     6            6           116m
deployment.apps/earth-3cc-runner       3/3     3            3           116m
deployment.apps/earth-3cc-runner-heavy 3/3     3            3           116m
deployment.apps/earth-3cc-web          0/4     4            0           116m

NAME                                              DESIRED   CURRENT   READY   AGE
replicaset.apps/earth-2x3-api-584df69757          3         3         3       116m
replicaset.apps/earth-2x3-web-85c5b7986c          6         6         6       116m
replicaset.apps/earth-3cc-runner-6cb6cc6974       3         3         3       116m
replicaset.apps/earth-3cc-runner-heavy-6bf876f46d 3         3         3       116m
```

```
replicaset.apps/earth-3cc-web-6895587dc7            0         0         0       116m
replicaset.apps/earth-3cc-web-6bfdf8b848            4         4         0       116m
replicaset.apps/earth-3cc-web-d49645966             0         0         0       116m
```

First impression could be that all *Pods* are in status RUNNING. But looking closely we see that some of the *Pods* are not ready, which also confirms what we see about one *Deployment* and one *replicaset*. This could be our error to further investigate.

Another approach could be to check the *Services* for missing endpoints:

```
➜ k -n earth get ep
NAME               ENDPOINTS                                               AGE
earth-2x3-api-svc  10.0.0.10:80,10.0.1.5:80,10.0.2.4:80                    116m
earth-2x3-web-svc  10.0.0.11:80,10.0.0.12:80,10.0.1.6:80 + 3 more...       116m
earth-3cc-web
```

Service `earth-3cc-web` doesn't have endpoints. This could be a selector/label misconfiguration or the endpoints are actually not available/ready.

Checking all *Services* for connectivity should show the same (this step is optional and just for demonstration):

```
➜ k run tmp --restart=Never --rm -i --image=nginx:alpine -- curl -m 5 earth-2x3-api-svc.earth:4546
...
<html><body><h1>It works!</h1></body></html>

➜ k run tmp --restart=Never --rm -i --image=nginx:alpine -- curl -m 5 earth-2x3-web-svc.earth:4545
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    45  100    45    0     0   5000      0 --:--:-- --:--:-- --:--:--  5000
<html><body><h1>It works!</h1></body></html>

➜ k run tmp --restart=Never --rm -i --image=nginx:alpine -- curl -m 5 earth-3cc-web.earth:6363
If you don't see a command prompt, try pressing enter.
  0     0    0     0    0     0      0      0 --:--:--  0:00:05 --:--:--     0
curl: (28) Connection timed out after 5000 milliseconds
pod "tmp" deleted
pod default/tmp terminated (Error)
```

Notice that we use here for example `earth-2x3-api-svc.earth`. We could also spin up a temporary *Pod* in *Namespace* `earth` and connect directly to `earth-2x3-api-svc`.

We get no connection to `earth-3cc-web.earth:6363`. Let's look at the *Deployment* `earth-3cc-web`. Here we see that the requested amount of replicas is not available/ready:

```
➜ k -n earth get deploy earth-3cc-web
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
earth-3cc-web  0/4     4            0           7m18s
```

To continue we check the *Deployment* yaml for some misconfiguration:

```
k -n earth edit deploy earth-3cc-web
```

```
# k -n earth edit deploy earth-3cc-web
apiVersion: extensions/v1beta1
kind: Deployment
metadata:
...
  generation: 3                    # there have been rollouts
  name: earth-3cc-web
  namespace: earth
...
spec:
...
  template:
    metadata:
      creationTimestamp: null
      labels:
        id: earth-3cc-web
    spec:
      containers:
      - image: nginx:1.16.1-alpine
        imagePullPolicy: IfNotPresent
        name: nginx
        readinessProbe:
          failureThreshold: 3
          initialDelaySeconds: 10
          periodSeconds: 20
          successThreshold: 1
          tcpSocket:
            port: 82               # this port doesn't seem to be right, should be 80
          timeoutSeconds: 1
...
```

We change the readiness-probe port, save and check the *Pods*:

```
➜ k -n earth get pod -l id=earth-3cc-web
NAME                            READY   STATUS    RESTARTS   AGE
earth-3cc-web-d49645966-52vb9   0/1     Running   0          6s
earth-3cc-web-d49645966-5tts6   0/1     Running   0          6s
earth-3cc-web-d49645966-db5gp   0/1     Running   0          6s
earth-3cc-web-d49645966-mk7gr   0/1     Running   0          6s
```

Running, but still not in ready state. Wait 10 seconds (initialDelaySeconds of readinessProbe) and check again:

```
➜ k -n earth get pod -l id=earth-3cc-web
NAME                            READY   STATUS    RESTARTS   AGE
earth-3cc-web-d49645966-52vb9   1/1     Running   0          32s
earth-3cc-web-d49645966-5tts6   1/1     Running   0          32s
earth-3cc-web-d49645966-db5gp   1/1     Running   0          32s
earth-3cc-web-d49645966-mk7gr   1/1     Running   0          32s
```

Let's check the service again:

```
➜ k run tmp --restart=Never --rm -i --image=nginx:alpine -- curl -m 5 earth-3cc-web.earth:6363
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   612  100   612    0     0  55636      0 --:--:-- --:--:-- --:--:-- 55636
<!DOCTYPE html>
<html>
<head>
<title>Welcome to nginx!</title>
<style>
    body {
        width: 35em;
        margin: 0 auto;
        font-family: Tahoma, Verdana, Arial, sans-serif;
    }
</style>
</head>
<body>
<h1>Welcome to nginx!</h1>
...
```

We did it! Finally we write the reason into the requested location:

```
vim /opt/course/p3/ticket-654.txt
```

```
# /opt/course/p3/ticket-654.txt
yo Dirk, wrong port for readinessProbe defined!
```