English ▾

Question weight: 5%

*Set configuration context:* `$ kubectl config use-context k8s`

Monitor the logs of *pod* **foo** and:

- Extract log lines corresponding to error `file-not-found`
- Write them to `/opt/KULM00201/foo`

Question: 1/24

‹ ›

Question 1▾

*Navigate All Questions*

English ▾

Question weight: 3%

*Set configuration context:* `$ kubectl config use-context k8s`

List all *persistent volumes* sorted by **capacity**, saving the full `kubectl` output to `/opt/KUCC00102/volume_list`. Use `kubectl`s own functionality for sorting the output, and do not manipulate it any further.

Question: 2/24

‹ ›

Question 2▾

*Navigate All Questions*

English ▾

Question weight: 3%

*Set configuration context:* `$ kubectl config use-context k8s`

Ensure a single instance of *pod* `nginx` is running on each *node* of the *Kubernetes cluster* where `nginx` also represents the *image* name which has to be used. Do not override any *taints* currently in place.

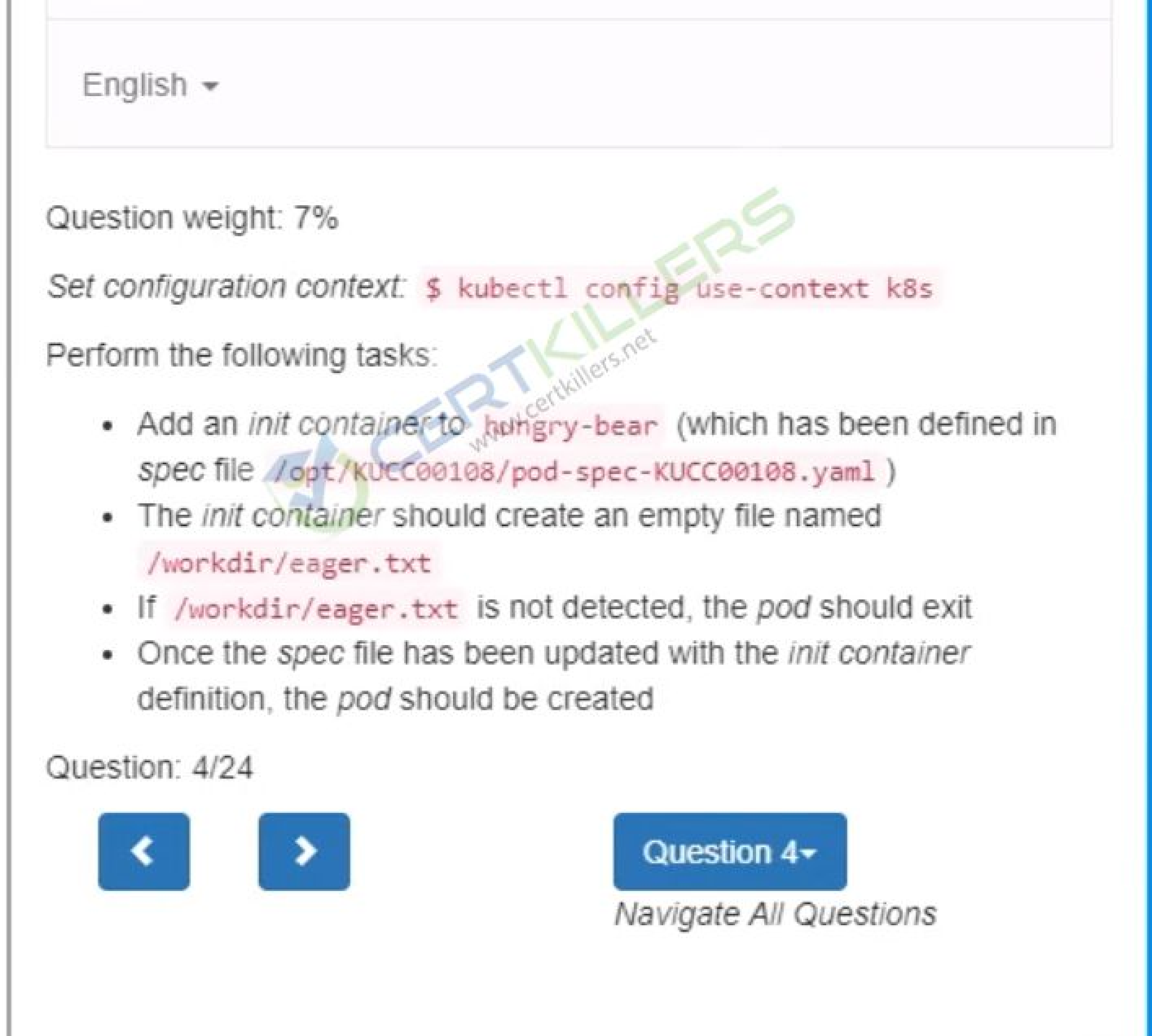
Use ***DaemonSet*** to complete this task and use `ds-kusc00201` as *DaemonSet* name.

Question: 3/24

< >

Question 3▾

*Navigate All Questions*



Question weight: 7%

*Set configuration context:* `$ kubectl config use-context k8s`

Perform the following tasks:

- Add an *init container* to `hungry-bear` (which has been defined in *spec* file `/opt/KUCC00108/pod-spec-KUCC00108.yaml` )
- The *init container* should create an empty file named `/workdir/eager.txt`
- If `/workdir/eager.txt` is not detected, the *pod* should exit
- Once the *spec* file has been updated with the *init container* definition, the *pod* should be created

Question: 4/24

English ▾

Question weight: 4%

*Set configuration context:* `$ kubectl config use-context k8s`

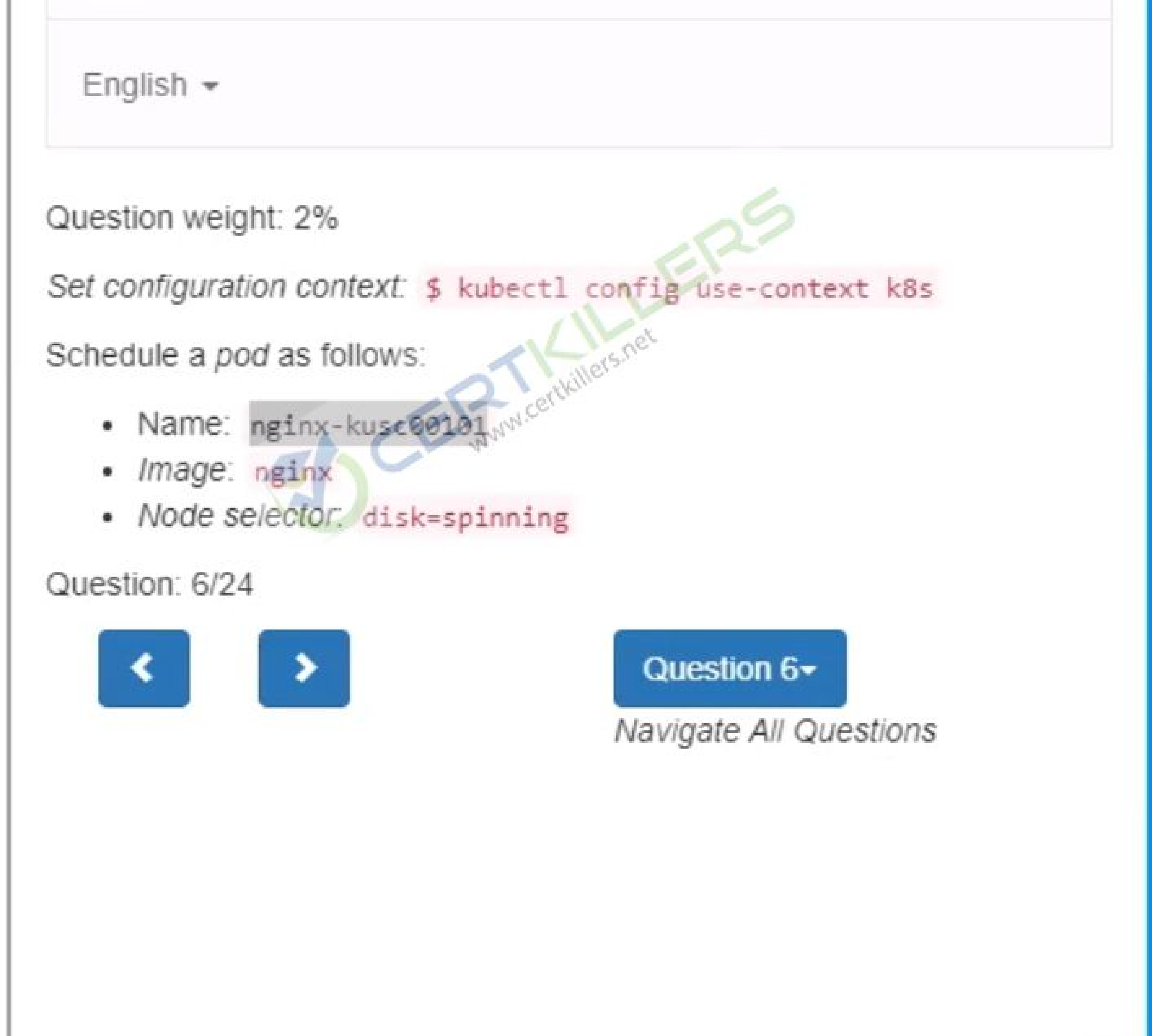
Create a *pod* named `kucc8` with a single *app container* for each of the following *images* running inside (there may be between 1 and 4 *images* specified): `nginx + redis` .

Question: 5/24

‹ ›

Question 5▾

*Navigate All Questions*



English ▾

Question weight: 2%

*Set configuration context:* `$ kubectl config use-context k8s`

Schedule a *pod* as follows:

- Name: `nginx-kusc00101`
- *Image:* `nginx`
- *Node selector:* `disk=spinning`

Question: 6/24

*Navigate All Questions*

English ▾

Question weight: 4%

*Set configuration context:* `$ kubectl config use-context k8s`

Create a *deployment* as follows:

- Name: `nginx-app`
- Using *container* `nginx` with version `1.11.9-alpine`
- The *deployment* should contain `3` replicas

Next, deploy the application with new version `1.12.0-alpine`, by performing a rolling update, and record that update.

Finally, rollback that update to the previous version `1.11.9-alpine`.

Question: 7/24

‹ ›

Question 7▾

*Navigate All Questions*

English ▾

Question weight: 4%

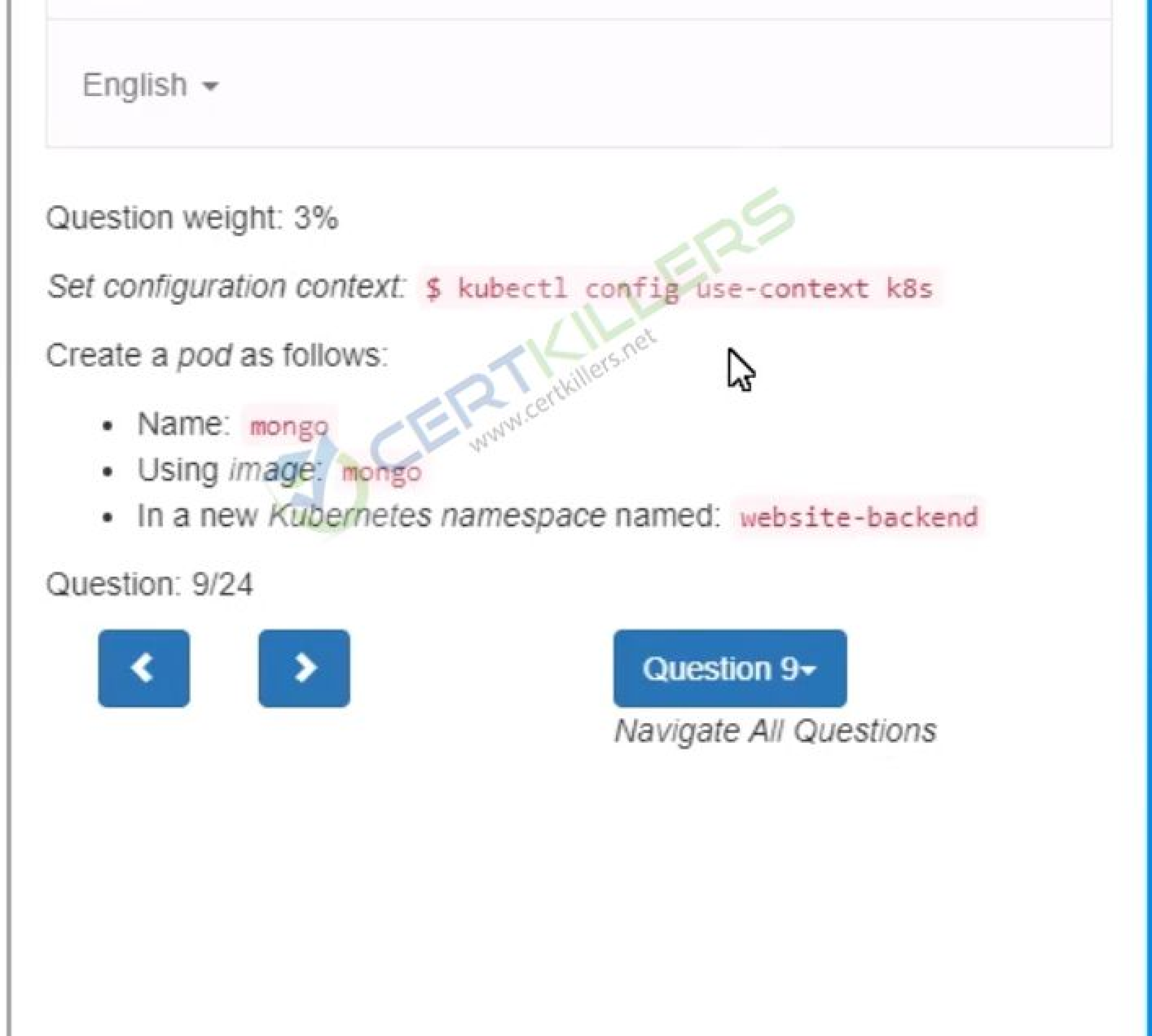
*Set configuration context:* `$ kubectl config use-context k8s`

Create and configure the *service* `front-end-service` so it's accessible through `NodePort` and routes to the existing *pod* named `front-end` .

Question: 8/24

Question 8▾

*Navigate All Questions*



Question weight: 3%

*Set configuration context:* `$ kubectl config use-context k8s`

Create a *pod* as follows:

- Name: `mongo`
- Using *image*: `mongo`
- In a new *Kubernetes namespace* named: `website-backend`

Question: 9/24

English ▾

Question weight: 3%

*Set configuration context:* `$ kubectl config use-context k8s`

Create a *deployment spec* file that will:

- Launch `3` replicas of the `nginx` *image* with the *label* `app_runtime_stage=dev`
- *deployment* name: `kual00201`

Save a copy of this *spec* file to `/opt/KUAL00201/deployment_spec.yaml` (or `.json` ).

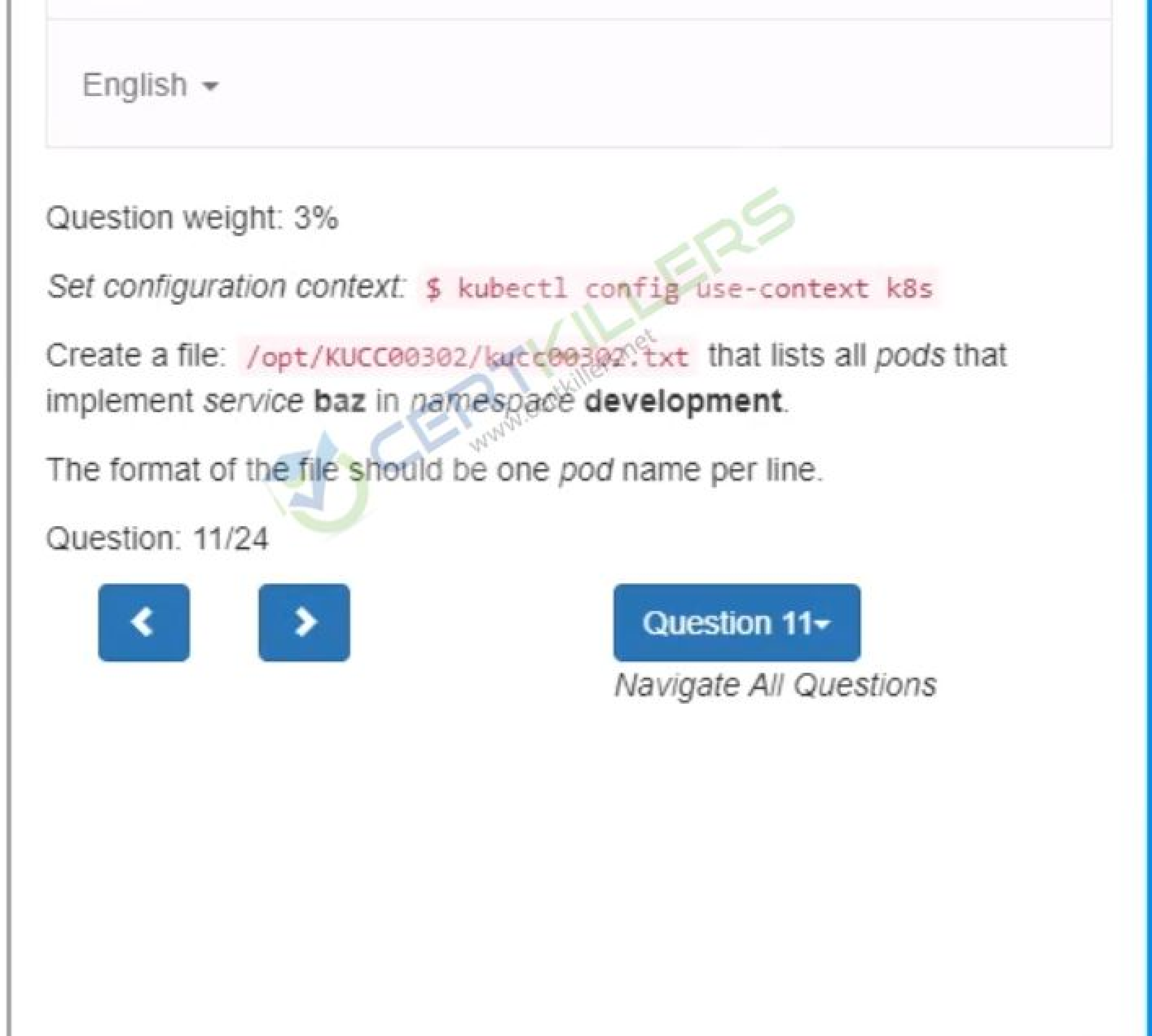
When you are done, clean up (delete) any new *Kubernetes API* object that you produced during this task.

Question: 10/24

‹ ›

Question 10▾

*Navigate All Questions*



English

Question weight: 3%

*Set configuration context:* `$ kubectl config use-context k8s`

Create a file: `/opt/KUCC00302/kucc00302.txt` that lists all *pods* that implement *service* **baz** in *namespace* **development**.

The format of the file should be one *pod* name per line.

Question: 11/24

Question 11

*Navigate All Questions*

English ▾

Question weight: 9%

*Set configuration context:* `$ kubectl config use-context k8s`

Create a *Kubernetes secret* as follows:

- Name: `super-secret`
- credential: `s3kr3t`

Create a *pod* named `pod-secrets-via-file` , using the ***redis*** *image*, which mounts a *secret* named `super-secret` at `/secrets` .

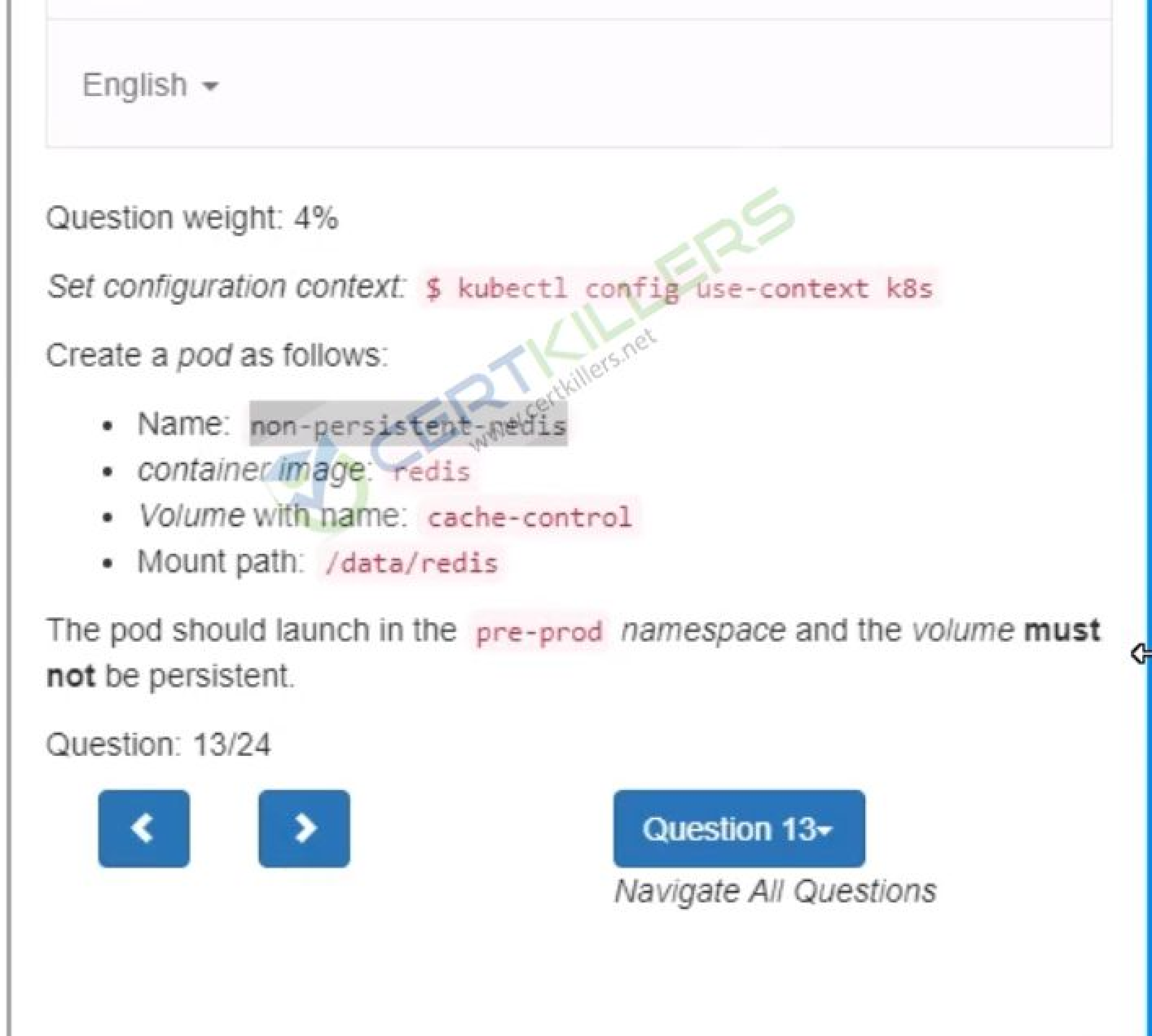
Create a second *pod* named `pod-secrets-via-env` , using the ***redis*** *image*, which exports **credential** as `CREDENTIALS` .

Question: 12/24

‹ ›

Question 12▾

*Navigate All Questions*



Question weight: 4%

*Set configuration context:* `$ kubectl config use-context k8s`

Create a *pod* as follows:

- Name: `non-persistent-redis`
- *container image:* `redis`
- *Volume* with name: `cache-control`
- Mount path: `/data/redis`

The pod should launch in the `pre-prod` *namespace* and the *volume* **must not** be persistent.

Question: 13/24

English ▾

Question weight: 1%

*Set configuration context:* `$ kubectl config use-context k8s`

Scale the *deployment* **guestbook** to **3** *pods.*

Question: 14/24

‹ ›

Question 14▾

*Navigate All Questions*

English ▾

Question weight: 2%

*Set configuration context:* `$ kubectl config use-context k8s`

Check to see how many *nodes* are ready (not including *nodes* tainted `NoSchedule` ) and write the number to `/opt/KUCC00104/kucc00104.txt` .

Question: 15/24

‹ ›

Question 15▾

*Navigate All Questions*

English ▾

Question weight: 2%

*Set configuration context:* `$ kubectl config use-context k8s`

From the *pod label* **name=overloaded-cpu**, find *pods* running high CPU workloads and write the name of the *pod* consuming most CPU to the file `/opt/KUTR00102/KUTR00102.txt` (which already exists).

Question: 16/24

Question 16▾

*Navigate All Questions*

English ▾

Question weight: 7%

*Set configuration context:* `$ kubectl config use-context k8s`

Create a *deployment* as follows:

- Name: `nginx-dns`
- Exposed via a *service* `nginx-dns`
- Ensure that the *service* & *pod* are accessible via their respective *DNS* records
- The *container(s)* within any *pod(s)* running as a part of this *deployment* should use the ***nginx*** *image*

Next, use the utility `nslookup` to look up the *DNS* records of the *service* & *pod* and write the output to `/opt/KUNW00601/service.dns` and `/opt/KUNW00601/pod.dns` respectively. Ensure you use the `busybox:1.28` *image* (or earlier) for any testing, as the latest release has an *upstream* bug which impacts the use of `nslookup` .

Question: 17/24

< > Question 17▾

English ▾

Question weight: 7%

*No configuration context change required for this item*

Create a snapshot of the `etcd` instance running at `https://127.0.0.1:2379`, saving the snapshot to the file path `/data/backup/etcd-snapshot.db`.

The `etcd` instance is running `etcd` version 3.3.10.

The following *TLS* certificates/key are supplied for connecting to the server with `etcdctl`:

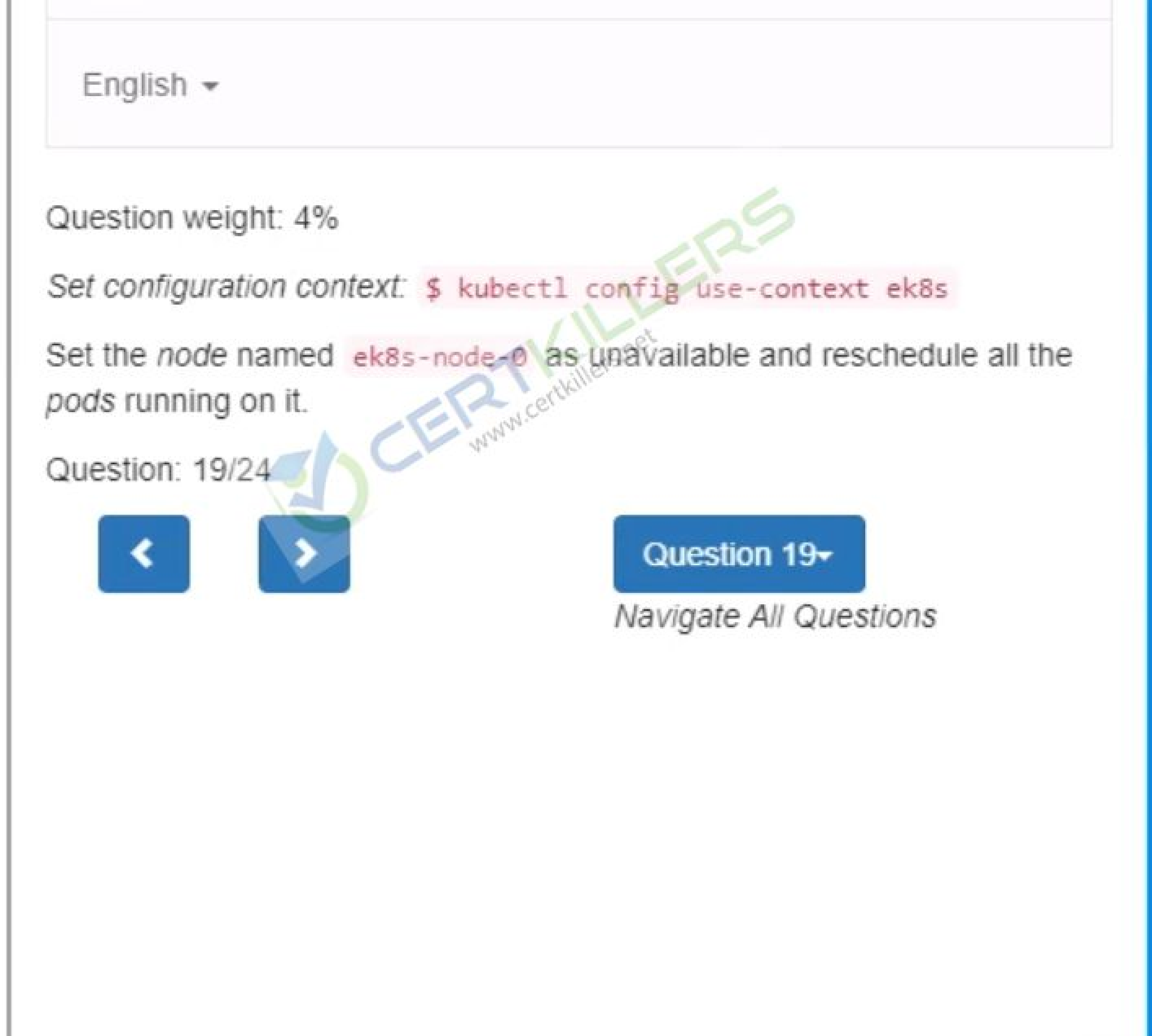
- *CA* certificate: `/opt/KUCM00302/ca.crt`
- Client certificate: `/opt/KUCM00302/etcd-client.crt`
- Client key: `/opt/KUCM00302/etcd-client.key`

Question: 18/24

‹ ›

Question 18▾

*Navigate All Questions*



English

Question weight: 4%

*Set configuration context:* `$ kubectl config use-context ek8s`

Set the *node* named `ek8s-node-0` as unavailable and reschedule all the *pods* running on it.

Question: 19/24

English ▾

Question weight: 4%

*Set configuration context:* `$ kubectl config use-context wk8s`

A *Kubernetes worker node*, named **wk8s-node-0** is in state `NotReady` . Investigate why this is the case, and perform any appropriate steps to bring the *node* to a `Ready` state, ensuring that any changes are made permanent.

*Hints*:

- You can `ssh` to the failed *node* using: `$ ssh wk8s-node-0`
- You can assume elevated privileges on the *node* with the following command: `$ sudo -i`

Question: 20/24