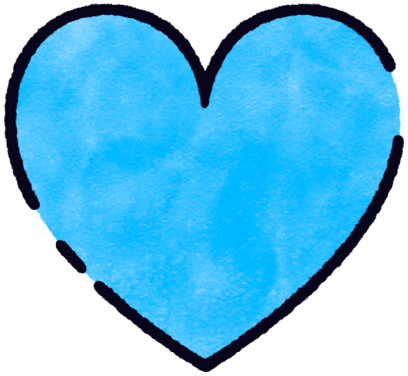
Implement Probes and Health Checks

Elle Krout
Principal Course Author, Pluralsight

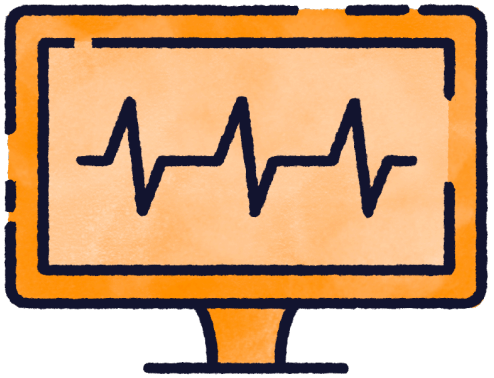
Kubernetes Probes and Health Checks

Observability is the ability to view and collect data about your pods and clusters.



Health Check

A tool for determining whether an application or service is functioning



Probe

A diagnostic mechanism use by `kubelet` to determine container health

Probes Ensure Pods Are...

Self-healing

Resilient

Stable

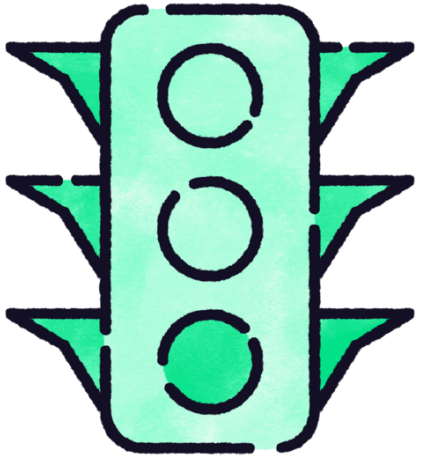
Exam Tip!

Using probes is a key part of passing this section of the CKAD... not to mention building strong Kubernetes systems in the real world.

Types of Kubernetes Probes

The Three Probes

Startup



Readiness

Liveness

Startup Probe

Used for prolonged container start times

- Databases
- Java Virtual Machines

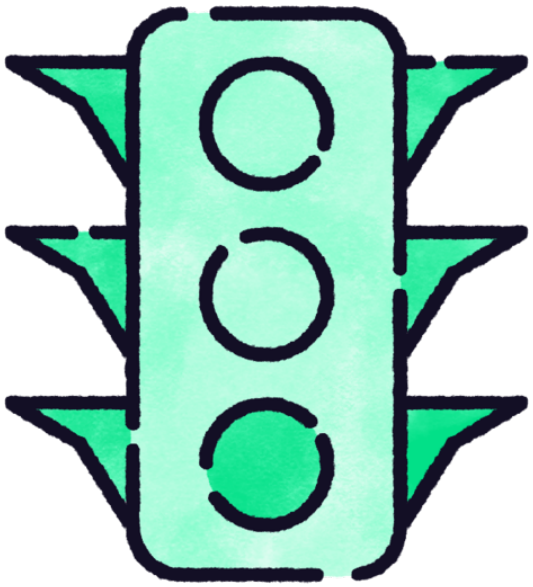
Verifies the application on the container has started

Runs at startup until probe succeeds or fails

Stops running after success/failure

Disables other probes until succeeds

Readiness Probe



Determines if endpoints can receive requests

Runs during containers lifecycle

If fails, removes pod from service

Graceful way of handling:

- Initialization
- Overloaded services

Liveness Probe

Determines if pod is healthy

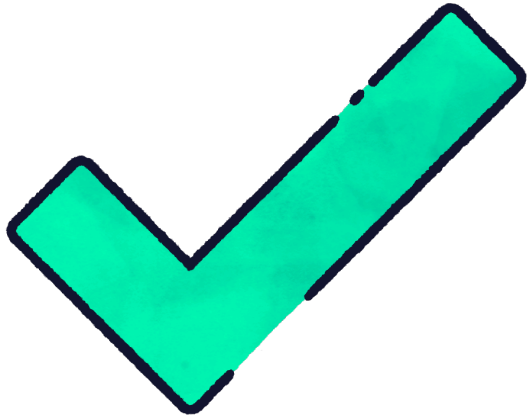
If fails, destroys pod and creates new one based on `restartPolicy`

Does not wait for readiness probes to run

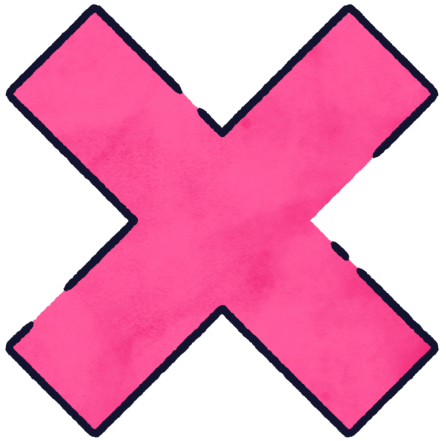
Ideal when applications experience:

- Deadlocks
- Hung processes
- Unresponsive services

Probe Results



Success
Everything works!



Failure
Remove or restart

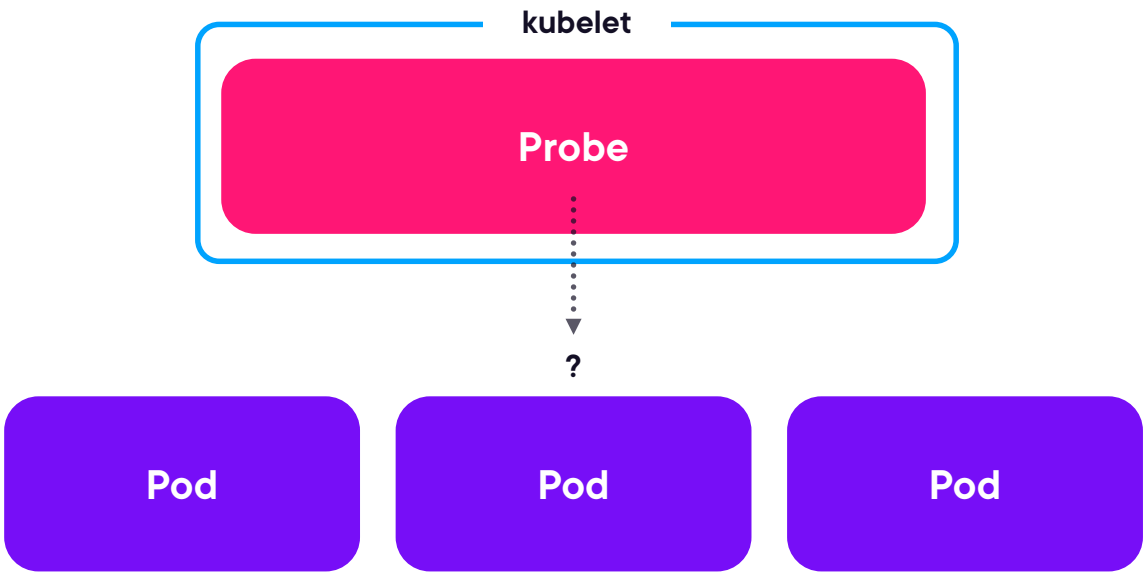
Unknown
Treated as failure

Probes help build resilient Kubernetes applications

Know the different and when to use them to pass the CKAD exam

Working with Kubernetes Probes

Kuberetes Probes



`httpGet`

Probe creates an HTTP GET request against container; expected 200-399 response

pod.yaml

```yaml
livenessProbe:
  httpGet:
    port: 80 # Any number between 1-65535; default 80
```

`httpGet`

Probe creates an HTTP GET request against container; expected 200-399 response

pod.yaml

```
livenessProbe:
  httpGet:
    port: 80
    host: k8s.example.local # Default IP address
```

`httpGet`

Probe creates an HTTP GET request against container; expected 200-399 response

pod.yaml

```
livenessProbe:
  httpGet:
    port: 80
    host: k8s.example.local
    scheme: http # Default HTTPS
```

`httpGet`

Probe creates an HTTP GET request against container; expected 200-399 response

pod.yaml

```
livenessProbe:
  httpGet:
    port: 80
    host: k8s.example.local
    scheme: http
    path: /healthz # Default /
```

`httpGet`

Probe creates an HTTP GET request against container; expected 200-399 response

pod.yaml

```
livenessProbe:
  httpGet:
    port: 80
    host: k8s.example.local
    scheme: http
    path: /healthz
    httpHeaders:
    - name: Custom
      value: Hello-Worlds
```

tcpSocket

Checks against a port at the container's IP address

pod.yaml

```yaml
readinessProbe:
  tcpSocker:
    port: 8080 # Any number between 1-65535
```

exec

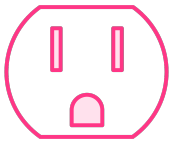
Probe performs action inside of container to check health

pod.yaml

```
startupProbe:
  exec:
    command:
    - cat
    - /tmp/started
```

Which Handlers Should You Use?

`httpGet` for web apps or application with a health or status endpoint



`tcpSocket` for services with raw TCP endpoints

`exec` for applications where state can only be verified internally

Sample Manifest

```
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2
      periodSeconds: 5
```

Sample Manifest

```
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2
      periodSeconds: 5
```

Sample Manifest

```
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2 # How long in seconds before probe runs
      periodSeconds: 5
```

Sample Manifest

```
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2
      periodSeconds: 5 # How often the probe will run
```

Sample Manifest

```
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2
      periodSeconds: 5
      timeoutSeconds: 5 # How long the probe will wait before timeout
```

Sample Manifest

```yaml
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2
      periodSeconds: 5
      timeoutSeconds: 5
      failureThreshold: 2 # How many times a probe can fail
```

Sample Manifest

```
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2
      periodSeconds: 5
      timeoutSeconds: 5
      failureThreshold: 2
      successThreshold: 2 # How many times a probe needs to succeed for success
```

Sample Manifest

```
spec:
  containers:
  - name: web-app
    image: nginx:latest
    readinessProbe:
      httpGet:
        path: /
        scheme: http
        port: 80
      initialDelaySeconds: 2
      periodSeconds: 5
      timeoutSeconds: 5
      failureThreshold: 2
      successThreshold: 2
      terminationGracePeriodSeconds: 20 # Time between requested shutdown and actual
                                          shutdown
```

Demo: Adding and Testing Kubernetes Probes

Exam Scenario

The `slow-start.yaml` pod manifest contains the configuration for a slow starting application that listens on port 9090. The connection starts before the service is fully ready; a file under `/tmp/live` indicates the pod has fully started up.

Add a combination of startup, readiness, and liveness probes to the manifest.

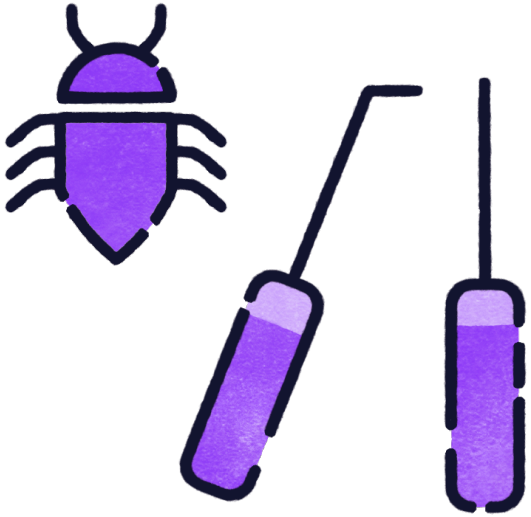
Ensure the startup probe confirms the pod is ready by accessing the `/tmp/live` and wait 30 seconds before it is run. The startup probe should be able to fail up to 5 times.

The readiness probe should succeed twice for the pod to be considered ready. It should check the connection at 9090.

Delay the liveness probe for 10 seconds. Have it run every 30 seconds after that. It should similarly check the connection at 9090.

Troubleshooting Kubernetes Probes

Troubleshooting Tips



Check the probe configuration

Check events:

```
kubectl describe pod <pod-name>
```

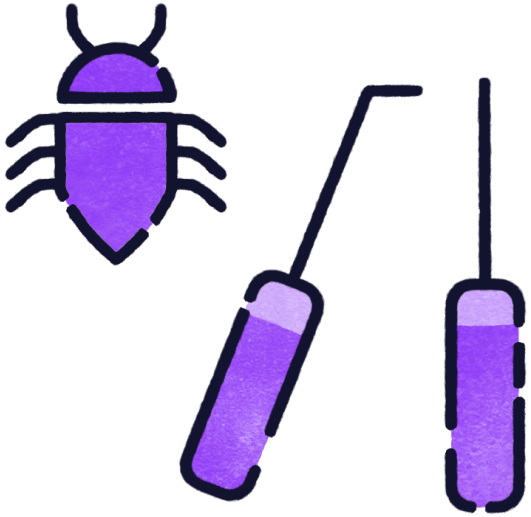
Check cluster events:

```
kubectl get events --sort-by=
'.lastTimestamp'
```

Check logs:

```
kubectl logs <pod-name> [-c
<container-name>]
```

Troubleshooting Tips



Check the probe configuration

Check events:

```
kubectl describe pod <pod-name>
```

Check cluster events:

```
kubectl get events --sort-by=
'.lastTimestamp'
```

Check logs:

```
kubectl logs <pod-name> [-c
<container-name>] --previous
```