Guided Exercise: Application Health Probes

Configure health probes in a deployment and verify that network clients are insulated from application failures.

Outcomes

- Assess potential issues with an application that is not configured with health probes.
- Configure startup, liveness, and readiness probes for the application.

As the `student` user on the `workstation` machine, use the `lab` command to prepare your system for this exercise.

This command ensures that the following conditions are true:

- The `reliability-probes` project exists.
- The resource files are available in the course directory.
- The classroom registry has the `long-load` container image.

The `registry.ocp4.example.com:8443/redhattraining/long-load:v1` container image contains an application with utility endpoints. These endpoints perform such tasks as crashing the process and toggling the server's health status.

```
[student@workstation ~]$ lab start reliability-probes
```

Instructions

1. As the `developer` user, deploy the `long-load` application in the `reliability-probes` project.

   Log in to the OpenShift cluster as the `developer` user with `developer` as the password.

   ```
   [student@workstation ~]$ oc login -u developer -p developer \
   https://api.ocp4.example.com:6443
   Login successful.
   ...output omitted...
   ```

   Set the `reliability-probes` project as the active project.

   ```
   [student@workstation ~]$ oc project reliability-probes
   Now using project reliability-probes on server "https://api.ocp4.example.com:6443".
   ```

   Apply the `long-load-deploy.yaml` file to create the pod. Move to the next step within one minute.

   ```
   [student@workstation ~]$ oc apply -f \
     ~/DO180/labs/reliability-probes/long-load-deploy.yaml
   deployment.apps/long-load created
   service/long-load created
   route.route.openshift.io/long-load created
   ```

   Verify that the pods take several minutes to start by sending a request to a pod in the deployment.

   ```
   [student@workstation ~]$ oc exec deploy/long-load -- \
     curl -s localhost:3000/health
   app is still starting
   ```

   Observe that the pods are ready even though the application is not ready.

   ```
   [student@workstation ~]$ oc get pods
   NAME                         READY   STATUS    RESTARTS   AGE
   long-load-8564d998cc-579nx   1/1     Running   0          30s
   long-load-8564d998cc-ttqpg   1/1     Running   0          30s
   long-load-8564d998cc-wjtfw   1/1     Running   0          30s
   ```

2. Add a startup probe to the pods so that the cluster knows when the pods are ready.

   Modify the deployment in the `~/DO180/labs/reliability-probes/long-load-deploy.yaml` YAML file by defining a startup probe. The probe runs every three seconds and triggers a pod as failed after 30 failed attempts. The file should match the following excerpt:

```
...output omitted...
spec:
  ...output omitted...
  template:
    ...output omitted...
    spec:
      containers:
      - image: registry.ocp4.example.com:8443/redhattraining/long-load:v1
        imagePullPolicy: Always
        name: long-load
        startupProbe:
          failureThreshold: 30
          periodSeconds: 3
          httpGet:
            path: /health
            port: 3000
        env:
...output omitted...
```

Apply the updated `long-load-deploy.yaml` file.

```
[student@workstation ~] oc apply -f \
  ~/DO180/labs/reliability-probes/long-load-deploy.yaml
deployment.apps/long-load configured
service/long-load unchanged
route.route.openshift.io/long-load configured
```

Observe that the pods do not show as ready until the application is ready and the startup probe succeeds. Wait for the three pods to reach the ready state. Press **Ctrl**+**c** to stop the watch command.

```
[student@workstation ~]$ watch oc get pods
Every 2.0s: oc get pods

NAME                         READY   STATUS    RESTARTS   AGE
long-load-785b5b4fc8-7x5ln   1/1     Running   0          90s
long-load-785b5b4fc8-f7pdk   1/1     Running   0          90s
long-load-785b5b4fc8-r2nqj   1/1     Running   0          90s
```

3. Add a liveness probe so that the cluster restarts broken instances of the application.

   Start the load test script.

   ```
   [student@workstation ~]$ ~/DO180/labs/reliability-probes/load-test.sh
   Ok
   Ok
   Ok
   ...output omitted...
   ```

   Keep the script running in a visible window.

   In a new terminal window, use the `/togglesick` endpoint to make one of the pods unhealthy. Move to the next step within one minute.

   ```
   [student@workstation ~]$ oc exec \
     deploy/long-load -- curl -s localhost:3000/togglesick
   no output expected
   ```

   The load test window begins to show `app is unhealthy`. Because only one pod is unhealthy, the remaining pods still respond with `Ok`.

   Update the deployment in the `~/DO180/labs/reliability-probes/long-load-deploy.yaml` file to add a liveness probe. The probe runs every three seconds and triggers the pod as failed after three failed attempts. Modify the `spec.template.spec.containers` object in the file to match the following excerpt.

```
spec:
  ...output omitted...
  template:
    ...output omitted...
    spec:
      containers:
      - image: registry.ocp4.example.com:8443/redhattraining/long-load:v1
        ...output omitted...
        startupProbe:
          failureThreshold: 30
          periodSeconds: 3
          httpGet:
            path: /health
            port: 3000
        livenessProbe:
          failureThreshold: 3
          periodSeconds: 3
          httpGet:
            path: /health
            port: 3000
        env:
...output omitted...
```

Scale down the deployment to zero replicas.

```
[student@workstation ~]$ oc scale deploy/long-load --replicas 0
deployment.apps/long-load scaled
```

The load test script shows that the application is not available.

> **NOTE**
>
> Red Hat recommends scaling down an application to zero replicas before making multiple changes to a deployment, such as adding health probes, resource requests, or environment variables. Scaling down to zero replicas prevents creating any pods, and enables pods to terminate gracefully after finishing all current requests.
>
> For multiple deployment updates, scaling down to zero prevents a noisy event log from Kubernetes responding to every change, and conserves cluster resources.

Apply the updated `long-load-deploy.yaml` file to update the deployment, which triggers the deployment to re-create its pods.

```
[student@workstation ~]$ oc apply -f \
  ~/DO180/labs/reliability-probes/long-load-deploy.yaml
deployment.apps/long-load configured
service/long-load unchanged
route.route.openshift.io/long-load unchanged
```

Wait for the three new pods to reach the ready state. Press **Ctrl**+**c** to stop the watch command.

```
[student@workstation ~]$ watch oc get pods
Every 2.0s: oc get pods

NAME                         READY   STATUS    RESTARTS   AGE
long-load-785b5b4fc8-8x5ln   1/1     Running   0          70s
long-load-785b5b4fc8-f8pdk   1/1     Running   0          70s
long-load-785b5b4fc8-r9nqj   1/1     Running   0          70s
```

Wait for the load test window to show `0k` for all responses, and then toggle one of the pods to be unhealthy.

```
[student@workstation ~]$ oc exec \
  deploy/long-load -- curl -s localhost:3000/togglesick
no output expected
```

The load test window might show `app is unhealthy` several times before the cluster restarts the pod.

Observe that the cluster restarts the unhealthy pod after the liveness probe fails. After the pod is restarted, the load test window shows only `0k`.

```
[student@workstation ~]$ watch oc get pods
NAME                        READY   STATUS    RESTARTS       AGE
long-load-fbb7468d9-8xm8j   1/1     Running   0              9m42s
long-load-fbb7468d9-k66dm   1/1     Running   0              8m38s
long-load-fbb7468d9-ncxkh   0/1     Running   1 (11s ago)    10m
```

4. Add a readiness probe so that traffic goes only to pods that are ready and healthy.

   Scale down the deployment to zero replicas.

   ```
   [student@workstation ~]$ oc scale deploy/long-load --replicas 0
   deployment.apps/long-load scaled
   ```

   Use the `oc set probe` command to add the readiness probe.

   ```
   [student@workstation ~]$ oc set probe deploy/long-load \
     --readiness --failure-threshold 1 --period-seconds 3 \
     --get-url http://:3000/health
   deployment.apps/long-load probes updated
   ```

   Scale up the deployment to three replicas.

   ```
   [student@workstation ~]$ oc scale deploy/long-load --replicas 3
   deployment.apps/long-load scaled
   ```

   Observe the status of the pods by using the `watch` command.

   ```
   [student@workstation ~]$ watch oc get pods
   NAME                        READY   STATUS    RESTARTS   AGE
   long-load-d5794d744-8hqlh   0/1     Running   0          48s
   long-load-d5794d744-hphgb   0/1     Running   0          48s
   long-load-d5794d744-lgkns   0/1     Running   0          48s
   ```

   The command does not immediately finish, but continues to show updates to the pods' status. Leave this command running in a visible window.

   Wait for the pods to show as ready. Then, in a new terminal window, make one of the pods unhealthy for five seconds by using the `/hiccup` endpoint.

   ```
   [student@workstation ~]$ oc exec \
     deploy/long-load -- curl -s localhost:3000/hiccup?time=5
   no output expected
   ```

   The pod status window shows that one of the pods is no longer ready.

   ```
   [student@workstation ~]$ watch oc get pods
   NAME                        READY   STATUS    RESTARTS   AGE
   long-load-d5794d744-8hqlh   1/1     Running   0          83s
   long-load-d5794d744-hphgb   0/1     Running   0          83s
   long-load-d5794d744-lgkns   1/1     Running   0          83s
   ```

   After five seconds, the pod is healthy again and shows as ready.

   The load test window might show `app is unhealthy` one time before the pod is set as not ready. After the cluster determines that the pod is no longer ready, it stops sending traffic to the pod until either the pod recovers or the liveness probe fails. Because the pod is sick for only five seconds, it is enough time for the readiness probe to fail, but not the liveness probe.

   > **NOTE**
   >
   > Optionally, repeat this step and observe as the temporarily sick pod's status changes.

   Stop the load test and status commands by pressing **Ctrl**+**c** in their respective windows.

Finish

On the `workstation` machine, use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish reliability-probes
```