Chapter 8.  Comprehensive Review

[**Comprehensive Review**](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch08/c2358540-87d5-48de-b49e-6f23bdcd629c)

[**Lab: Deploy Web Applications**](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch08s02/c2358540-87d5-48de-b49e-6f23bdcd629c)

[**Lab: Troubleshoot and Scale Applications**](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch08s03/c2358540-87d5-48de-b49e-6f23bdcd629c)

**Abstract**

|  |  |
| --- | --- |
| **Goal** | Review tasks from *Red Hat OpenShift Administration I: Operating a Production Cluster*. |
| **Sections** | * Comprehensive Review |
| **Lab** | * Deploy Web Applications * Troubleshoot and Scale Applications |

Comprehensive Review

Objectives

After completing this section, you should have reviewed and refreshed the knowledge and skills that you learned in *Red Hat OpenShift Administration I: Operating a Production Cluster*.

Reviewing *Red Hat OpenShift Administration I: Operating a Production Cluster*

Before beginning the comprehensive review for this course, you should be comfortable with the topics covered in each chapter. Do not hesitate to ask the instructor for extra guidance or clarification on these topics.

[Chapter 1, *Introduction to Kubernetes and OpenShift*](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch01/c2358540-87d5-48de-b49e-6f23bdcd629c)

Identify the main Kubernetes cluster services and OpenShift platform services and monitor them by using the web console.

* Describe the main characteristics of containers and Kubernetes.
* Describe the relationship between OpenShift, Kubernetes, and other Open Source projects, and list key features of Red Hat OpenShift products and editions.
* Navigate the OpenShift web console to identify running applications and cluster services.
* Navigate the Events, Compute, and Observe panels of the OpenShift web console to assess the overall state of a cluster.

[Chapter 2, *Kubernetes and OpenShift Command-line Interfaces and APIs*](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch02/c2358540-87d5-48de-b49e-6f23bdcd629c)

Access an OpenShift cluster by using the command line and query its Kubernetes API resources to assess the health of a cluster.

* Access an OpenShift cluster by using the Kubernetes and OpenShift command-line interfaces.
* Query, format, and filter attributes of Kubernetes resources.
* Query the health of essential cluster services and components.

[Chapter 3, *Run Applications as Containers and Pods*](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch03/c2358540-87d5-48de-b49e-6f23bdcd629c)

Run and troubleshoot containerized applications as unmanaged Kubernetes pods.

* Run containers inside pods and identify the host OS processes and namespaces that the containers use.
* Find containerized applications in container registries and get information about the runtime parameters of supported and community container images.
* Troubleshoot a pod by starting additional processes on its containers, changing their ephemeral file systems, and opening short-lived network tunnels.

[Chapter 4, *Deploy Managed and Networked Applications on Kubernetes*](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch04/c2358540-87d5-48de-b49e-6f23bdcd629c)

Deploy applications and expose them to network access from inside and outside a Kubernetes cluster.

* Identify the main resources and settings that Kubernetes uses to manage long-lived applications and demonstrate how OpenShift simplifies common application deployment workflows.
* Deploy containerized applications as pods that Kubernetes workload resources manage.
* Interconnect applications pods inside the same cluster by using Kubernetes services.
* Expose applications to clients outside the cluster by using Kubernetes ingress and OpenShift routes.

[Chapter 5, *Manage Storage for Application Configuration and Data*](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch05/c2358540-87d5-48de-b49e-6f23bdcd629c)

Externalize application configurations in Kubernetes resources and provision storage volumes for persistent data files.

* Configure applications by using Kubernetes secrets and configuration maps to initialize environment variables and to provide text and binary configuration files.
* Provide applications with persistent storage volumes for block and file-based data.
* Match applications with storage classes that provide storage services to satisfy application requirements.
* Deploy applications that scale without sharing storage.

[Chapter 6, *Configure Applications for Reliability*](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch06/c2358540-87d5-48de-b49e-6f23bdcd629c)

Configure applications to work with Kubernetes for high availability and resilience.

* Describe how Kubernetes tries to keep applications running after failures.
* Describe how Kubernetes uses health probes during deployment, scaling, and failover of applications.
* Configure an application with resource requests so Kubernetes can make scheduling decisions.
* Configure an application with resource limits so Kubernetes can protect other applications from it.
* Configure a horizontal pod autoscaler for an application.

[Chapter 7, *Manage Application Updates*](https://rha.ole.redhat.com/rha/app/courses/do180-4.14/pages/ch07/c2358540-87d5-48de-b49e-6f23bdcd629c)

Manage reproducible application updates and rollbacks of code and configurations.

* Relate container image tags to their identifier hashes, and identify container images from pods and containers on Kubernetes nodes.
* Update applications with minimal downtime by using deployment strategies.
* Ensure reproducibility of application deployments by using image streams and short image names.
* Ensure automatic update of application pods by using image streams with Kubernetes workload resources.

Lab: Deploy Web Applications

Use image streams with Kubernetes workload resources to ensure reproducibility of application deployments.

Configure applications by using Kubernetes secrets to initialize environment variables.

Provide applications with persistent storage volumes.

Expose applications to clients outside the cluster.

**Outcomes**

You should be able to create and configure OpenShift and Kubernetes resources, such as projects, secrets, deployments, persistent volumes, services, and routes.

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

This command ensures that all resources are available for this exercise. The command also creates the /home/student/DO180/labs/compreview-deploy/resources.txt file. The resources.txt file contains the URLs of your OpenShift cluster and the image names that you use in the exercise. You can use the file to copy and paste these URLs and image names.

[student@workstation ~]$ **lab start compreview-deploy**

**Specifications**

The API URL of your OpenShift cluster is https://api.ocp4.example.com:6443, and the oc command is already installed on your workstation machine.

The URL of the OpenShift web console is https://console-openshift-console.apps.ocp4.example.com. When you access the web console, select **Red Hat Identity Management** as the authentication mechanism.

Log in to the OpenShift cluster as the developer user with the developer password. The password for the admin user is redhatocp, although you do not need administrator privileges to complete the exercise.

In this exercise, you deploy a web application and its database for testing purposes. The resulting configuration is not ready for production, because you do not configure probes and resource limits, which are required for production. Another comprehensive review exercise covers these subjects.

Perform the following tasks to complete the exercise:

* Create a project named review to store your work.
* Configure your project so that its workloads refer to the database image by the mysql8:1 short name.
  + The short name must point to the registry.ocp4.example.com:8443/rhel9/mysql-80:1-228 container image. The database image name and its source registry are expected to change in the near future, and you want to isolate your workloads from that change.

The classroom setup copied the image from the Red Hat Ecosystem Catalog. The original image is registry.redhat.io/rhel9/mysql-80:1-228.

* + Ensure that the workload resources in the review project can use the mysql8:1 resource. You create these workload resources in a later step.
* Create the dbparams secret to store the MySQL database parameters. Both the database and the front-end deployment need these parameters. The dbparams secret must include the following variables:

| **Name** | **Value** |
| --- | --- |
| user | operator1 |
| password | redhat123 |
| database | quotesdb |

* Create the quotesdb deployment and configure it as follows:
  + Use the mysql8:1 image for the deployment.
  + The database must automatically roll out whenever the source container in the mysql8:1 resource changes.

To test your configuration, you can change the mysql8:1 image to point to the registry.ocp4.example.com:8443/rhel9/mysql-80:1-237 container image that the classroom provides, and then verify that the quotesdb deployment rolls out. Remember to reset the mysql8:1 image to the registry.ocp4.example.com:8443/rhel9/mysql-80:1-228 container image before grading your work.

* + Define the following environment variables in the deployment from the keys in the dbparams secret:

| **Environment variable** | **dbparams secret key** |
| --- | --- |
| MYSQL\_USER | user |
| MYSQL\_PASSWORD | password |
| MYSQL\_DATABASE | database |

* + Ensure that OpenShift preserves the database data between pod restarts. This data does not consume more than 2 GiB of disk space. The MySQL database stores its data under the /﻿var/lib/mysql directory. Use the lvms-vg1 storage class for the volume.
* Create a quotesdb service to make the database available to the front-end web application. The database service is listening on port 3306.
* Create the frontend deployment and configure it as follows:
  + Use the registry.ocp4.example.com:8443/redhattraining/famous-quotes:2-42 image. For this deployment, you refer to the image by its full name, because your organization develops the image and controls its release process.
  + Define the following environment variables in the deployment:

| **Environment variable name** | **Value** |
| --- | --- |
| QUOTES\_USER | The user key from the dbparams secret |
| QUOTES\_PASSWORD | The password key from the dbparams secret |
| QUOTES\_DATABASE | The database key from the dbparams secret |
| QUOTES\_HOSTNAME | quotesdb |

* You cannot yet test the application from outside the cluster. Expose the frontend deployment so that the application can be reached at http://frontend-review.apps.ocp4.example.com.

The frontend deployment is listening to port 8000.

When you access the http://frontend-review.apps.ocp4.example.com URL, the application returns a list of quotations from famous authors.

1. Log in to the OpenShift cluster from the command line, and then create the review project.
   1. Log in as the developer user.
   2. [student@workstation ~]$ **oc login -u developer -p developer \**
   3. **https://api.ocp4.example.com:6443**
   4. Login successful.

*...output omitted...*

* 1. Create the review project.
  2. [student@workstation ~]$ **oc new-project review**
  3. Now using project "review" on server "https://api.ocp4.example.com:6443".

*...output omitted...*

1. Create the mysql8:1 image stream tag from the registry.ocp4.example.com:8443/rhel9/mysql-80:1-228 image. Enable image stream resolution for the mysql8 image stream so that Kubernetes resources in the current project can use it.
   1. Use the oc create istag command to create the image stream and the image stream tag.
   2. [student@workstation ~]$ **oc create istag mysql8:1 \**
   3. **--from-image registry.ocp4.example.com:8443/rhel9/mysql-80:1-228**

imagestreamtag.image.openshift.io/mysql8:1 created

* 1. Use the oc set image-lookup command to enable image lookup resolution.
  2. [student@workstation ~]$ **oc set image-lookup mysql8**

imagestream.image.openshift.io/mysql8 image lookup updated

* 1. Run the oc set image-lookup command without any arguments to verify your work.
  2. [student@workstation ~]$ **oc set image-lookup**
  3. NAME LOCAL

mysql8 **true**

1. Create the dbparams secret.
2. [student@workstation ~]$ **oc create secret generic dbparams \**
3. **--from-literal user=operator1 --from-literal password=redhat123 \**
4. **--from-literal database=quotesdb**

secret/dbparams created

1. Create the quotesdb deployment from the mysql8:1 image stream tag. Set the number of replicas to zero, to prevent OpenShift from deploying the database before you finish its configuration.
2. [student@workstation ~]$ **oc create deployment quotesdb --image mysql8:1 \**
3. **--replicas 0**

deployment.apps/quotesdb created

1. Add an image trigger to the quotesdb deployment.
   1. Retrieve the name of the container from the quotesdb deployment.
   2. [student@workstation ~]$ **oc get deployment quotesdb -o wide**
   3. NAME READY UP-TO-DATE AVAILABLE AGE CONTAINERS ...

quotesdb 0/0 0 0 11s **mysql8** ...

* 1. Use the oc set triggers command to add the trigger for the mysql8:1 image stream tag to the mysql8 container.
  2. [student@workstation ~]$ **oc set triggers deployment/quotesdb \**
  3. **--from-image mysql8:1 --containers mysql8**

deployment.apps/quotesdb triggers updated

1. Add environment variables to the quotesdb deployment from the dbparams secret. Add the MYSQL\_ prefix to each variable name.
2. [student@workstation ~]$ **oc set env deployment/quotesdb \**
3. **--from secret/dbparams --prefix MYSQL\_**

deployment.apps/quotesdb updated

1. Add a 2 GiB persistent volume to the quotesdb deployment. Use the lvms-vg1 storage class. Inside the pods, mount the volume under the /var/lib/mysql directory.
2. [student@workstation ~]$ **oc set volumes deployment/quotesdb --add \**
3. **--claim-class lvms-vg1 --claim-size 2Gi --mount-path /var/lib/mysql**
4. info: Generated volume name: volume-n7xpd

deployment.apps/quotesdb volume updated

1. Start the database by scaling up the quotesdb deployment to one replica.
   1. Scale up the deployment.
   2. [student@workstation ~]$ **oc scale deployment/quotesdb --replicas 1**

deployment.apps/quotesdb scaled

* 1. Wait for the pod to start. You might have to rerun the command several times for the pod to report a Running status. The name of the pod on your system probably differs.
  2. [student@workstation ~]$ **oc get pods**
  3. NAME READY STATUS RESTARTS AGE

quotesdb-99f9b4ff8-ggs7z 1/1 **Running** 0 4s

1. Create the quotesdb service for the quotesdb deployment. The database server is listening on port 3306.
   1. Use the oc expose deployment command to create the service.
   2. [student@workstation ~]$ **oc expose deployment quotesdb --port 3306**

service/quotesdb exposed

* 1. Verify that OpenShift associates the IP address of the MySQL server with the endpoint. The endpoint IP address on your system probably differs.
  2. [student@workstation ~]$ **oc describe service quotesdb**
  3. Name: quotesdb
  4. Namespace: review
  5. *...output omitted...*
  6. TargetPort: 3306/TCP
  7. **Endpoints: 10.8.0.123:3306**
  8. Session Affinity: None

Events: <none>

1. Create the frontend deployment from the registry.ocp4.example.com:8443/redhattraining/famous-quotes:2-42 image. Set the number of replicas to zero, to prevent OpenShift from deploying the application before you finish its configuration.
2. [student@workstation ~]$ **oc create deployment frontend \**
3. **--image registry.ocp4.example.com:8443/redhattraining/famous-quotes:2-42 \**
4. **--replicas 0**

deployment.apps/frontend created

1. Add environment variables to the frontend deployment from the dbparams secret, and add the QUOTES\_HOSTNAME variable with the quotesdb value.
   1. Add the variables from the dbparams secret. Add the QUOTES\_ prefix to each variable name.
   2. [student@workstation ~]$ **oc set env deployment/frontend \**
   3. **--from secret/dbparams --prefix QUOTES\_**

deployment.apps/frontend updated

* 1. Declare the QUOTES\_HOSTNAME variable.
  2. [student@workstation ~]$ **oc set env deployment/frontend QUOTES\_HOSTNAME=quotesdb**

deployment.apps/frontend updated

1. Start the application by scaling up the frontend deployment to one replica.
   1. Scale up the deployment.
   2. [student@workstation ~]$ **oc scale deployment/frontend --replicas 1**

deployment.apps/frontend scaled

* 1. Wait for the pod to start. You might have to rerun the command several times for the pod to report a Running status. The name of the pod on your system probably differs.
  2. [student@workstation ~]$ **oc get pods**
  3. NAME READY STATUS RESTARTS AGE
  4. **frontend-86cdd7c7bf-hpnwz** 1/1 **Running** 0 44s

quotesdb-99f9b4ff8-ggs7z 1/1 Running 0 2m11s

1. Expose the frontend deployment so that the application is accessible from outside the cluster. The web application is listening on port 8000.
   1. Create the frontend service for the frontend deployment.
   2. [student@workstation ~]$ **oc expose deployment frontend --port 8000**

service/frontend exposed

* 1. Create the route.
  2. [student@workstation ~]$ **oc expose service frontend**

route.route.openshift.io/frontend exposed

* 1. Retrieve the application URL from the route.
  2. [student@workstation ~]$ **oc get route**
  3. NAME HOST/PORT PATH SERVICES ...

frontend **frontend-review.apps.ocp4.example.com** frontend ...

* 1. Use the curl command to test the application.
  2. [student@workstation ~]$ **curl http://frontend-review.apps.ocp4.example.com**
  3. <html>
  4. <head>
  5. <title>Quotes</title>
  6. </head>
  7. <body>
  8. <h1>Quote List</h1>
  9. <ul>
  10. <li>1: When words fail, music speaks.
  11. - William Shakespeare
  12. </li>

*...output omitted...*

Hide Solution

**Evaluation**

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

[student@workstation ~]$ **lab grade compreview-deploy**

**Finish**

As the student user 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 compreview-deploy**

Lab: Troubleshoot and Scale Applications

Navigate the OpenShift web console to identify CPU-consuming workloads.

Troubleshoot and fix a failed MySQL pod.

Manually scale an application.

Configure health probes.

**Outcomes**

You should be able to troubleshoot malfunctioning workloads, configure deployments, and scale applications.

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

This command ensures that all resources are available for this exercise. The command also creates the compreview-scale project and deploys some applications in that project.

The command creates the /home/student/DO180/labs/compreview-scale/resources.txt file. The resources.txt file contains the URLs of your OpenShift cluster and the name of the images that you use during the exercise. You can use the file to copy and paste these URLs and image names.

[student@workstation ~]$ **lab start compreview-scale**

**Specifications**

The API URL of your OpenShift cluster is https://api.ocp4.example.com:6443, and the oc command is already installed on your workstation machine.

The URL of the OpenShift web console is https://console-openshift-console.apps.ocp4.example.com. When you access the web console, select **Red Hat Identity Management** as the authentication mechanism.

Log in to the OpenShift cluster as the developer user with the developer password. The password for the admin user is redhatocp.

Perform the following tasks to complete the comprehensive review:

* A pod in the cluster is consuming excessive CPU and is interfering with other tasks. Identify the pod and remove its workload.
* The compreview-scale project already includes a web application at http://frontend-compreview-scale.apps.ocp4.example.com. When you access this URL, the application returns a list of quotations from famous authors. The application is broken for now, and is missing some configuration to be ready for production.

The application uses two Kubernetes Deployment objects. The frontend deployment provides the application web pages, and relies on the quotesdb deployment that runs a MySQL database. The lab command already created the services and routes that connect the application components and that make the application available from outside the cluster.

Fix the application and make it ready for production:

* + The quotesdb deployment in the compreview-scale project starts a MySQL server, but the database is failing. Review the logs of the pod to identify and then fix the issue.

Use the following parameters for the database:

| **Name** | **Value** |
| --- | --- |
| Username | operator1 |
| Password | redhat123 |
| Database name | quotes |

* + You security team validated a new version of the MySQL container image that fixes a security issue. The new container image is registry.ocp4.example.com:8443/rhel9/mysql-80:1-237.

Update the quotesdb deployment to use this image. Ensure that the database redeploys.

The classroom setup copied the image from the Red Hat Ecosystem Catalog. The original image is registry.redhat.io/rhel9/mysql-80:1-237.

* + Add a probe to the quotesdb deployment so that OpenShift can detect when the database is ready to accept requests. Use the mysqladmin ping command for the probe.
  + Add a second probe that regularly verifies the status of the database. Use the mysqladmin ping command as well.
  + Configure CPU and memory usage for the quotesdb deployment. The deployment needs 200 millicores of CPU and 256 MiB of memory to run, and you must restrict its CPU usage to 500 millicores and its memory usage to 1 GiB.
  + Add a probe to the frontend deployment so that OpenShift can detect when the web application is ready to accept requests. The application is ready when an HTTP request on port 8000 to the /status path is successful.
  + Add a second probe that regularly verifies the status of the web front end. The front end works as expected when an HTTP request on port 8000 to the /env path is successful.
  + Configure CPU and memory usage for the frontend deployment. The deployment needs 200 millicores of CPU and 256 MiB of memory to run, and you must restrict its CPU usage to 500 millicores and its memory usage to 512 MiB.
  + Scale the frontend application to three pods to accommodate for the estimated production load.
  + To verify your work, access the http://frontend-compreview-scale.apps.ocp4.example.com URL. The application returns a list of quotations from famous authors.

1. Use the OpenShift web console to identify and then delete the pod that consumes excessive CPU.
   1. Use a web browser to access the https://console-openshift-console.apps.ocp4.example.com URL.
   2. Select **Red Hat Identity Management**, and then log in as the admin user with the redhatocp password. Click **Skip tour** if the **Welcome to the Developer Perspective** message is displayed.
   3. Switch to the **Administrator** perspective and then navigate to **Observe** → **Dashboards**.

A screenshot of a graph

Description automatically generated

* 1. Select the **Kubernetes / Compute Resources / Cluster** dashboard, and then click **Inspect** in the **CPU Usage** graph.

A screenshot of a computer

Description automatically generated

* 1. Set the zoom to five minutes and then hover over the graph. Notice that the interface lists the compreview-scale-load namespace in the first position, which indicated that this namespace is the first CPU consumer.

A screenshot of a computer

Description automatically generated

* 1. Navigate to **Observe** → **Dashboards** and then select the **Kubernetes / Compute Resources / Namespace (Workloads)** dashboard. Select the compreview-scale-load namespace and then set the time range to the last five minutes. The computeprime deployment is the workload that consumes excessive CPU.

A screenshot of a computer screen

Description automatically generated

* 1. Navigate to **Workloads** → **Deployments** and then select the compreview-scale-load project. Select the menu for the computeprime deployment and then click **Delete Deployment**. Click **Delete** to confirm the operation.

A screenshot of a computer

Description automatically generated

1. Review the logs of the pod that is failing for the quotesdb deployment. Set the missing environment variables in the quotesdb deployment.
   1. Log in to the OpenShift cluster from the command line.
   2. [student@workstation ~]$ **oc login -u developer -p developer \**
   3. **https://api.ocp4.example.com:6443**
   4. Login successful.

*...output omitted...*

* 1. Set the compreview-scale project as the active project.
  2. [student@workstation ~]$ **oc project compreview-scale**

*...output omitted...*

* 1. List the pods to identify the failing pod from the quotesdb deployment. The names of the pods on your system probably differ.
  2. [student@workstation ~]$ **oc get pods**
  3. NAME READY STATUS RESTARTS AGE
  4. frontend-5fb85b4c75-5s7xr 0/1 CrashLoopBackOff 14 (2m52s ago) 50m

**quotesdb-9b9776479-4z4g9** 0/1 CrashLoopBackOff 14 (3m4s ago) 50m

* 1. Retrieve the logs for the failing pod. Some environment variables are missing.
  2. [student@workstation ~]$ **oc logs quotesdb-*9b9776479-4z4g9***
  3. => sourcing 20-validate-variables.sh ...
  4. You must either specify the following environment variables:
  5. **MYSQL\_USER** (regex: '^[a-zA-Z0-9\_]+$')
  6. **MYSQL\_PASSWORD** (regex: '[a-zA-Z0-9\_~!@#$%&\*()-=<>,.?;:|]+$')
  7. **MYSQL\_DATABASE** (regex: '^[a-zA-Z0-9\_]+$')

*...output omitted...*

* 1. Add the missing environment variables to the quotesdb deployment.
  2. [student@workstation ~]$ **oc set env deployment/quotesdb \**
  3. **MYSQL\_USER=operator1 MYSQL\_PASSWORD=redhat123 MYSQL\_DATABASE=quotes**

deployment.apps/quotesdb updated

1. Update the MySQL container image for the quotesdb deployment.
   1. Retrieve the name of the container that is running inside the pod. You need the container name to update its image.
   2. [student@workstation ~]$ **oc get deployment/quotesdb -o wide**
   3. NAME READY UP-TO-DATE AVAILABLE AGE CONTAINERS ...

quotesdb 1/1 1 1 59m **mysql-80** ...

* 1. Set the image to registry.ocp4.example.com:8443/rhel9/mysql-80:1-237.
  2. [student@workstation ~]$ **oc set image deployment/quotesdb \**
  3. **mysql-80=registry.ocp4.example.com:8443/rhel9/mysql-80:1-237**

deployment.apps/quotesdb image updated

* 1. Verify your work.
  2. [student@workstation ~]$ **oc get deployment/quotesdb -o wide**
  3. NAME ... CONTAINERS IMAGES

quotesdb ... mysql-80 **registry.ocp4.example.com:8443/rhel9/mysql-80:1-237**

* 1. Wait for the deployment to roll out. You might have to rerun the command several times for the pod to report a Running status. The name of the pod on your system probably differs.
  2. [student@workstation ~]$ **oc get pods**
  3. NAME READY STATUS RESTARTS AGE
  4. frontend-5fb85b4c75-5s7xr 0/1 CrashLoopBackOff 15 (3m39s ago) 56m

**quotesdb-54d64749c4-chhq6** 1/1 **Running** 0 106s

1. Add a readiness and a liveness probe to the quotesdb deployment that runs the mysqladmin ping command.
   1. Use the oc set probe command with the --readiness option to add the readiness probe.
   2. [student@workstation ~]$ **oc set probe deployment/quotesdb \**
   3. **--readiness -- mysqladmin ping**

deployment.apps/quotesdb probes updated

* 1. Use the oc set probe command with the --liveness option to add the liveness probe.
  2. [student@workstation ~]$ **oc set probe deployment/quotesdb \**
  3. **--liveness -- mysqladmin ping**

deployment.apps/quotesdb probes updated

1. Define resource limits for the quotesdb deployment. Set the CPU request to 200 millicores and the memory request to 256 MiB. Set the CPU limit to 500 millicores and the memory limit to 1 GiB.
2. [student@workstation ~]$ **oc set resources deployment/quotesdb \**
3. **--requests cpu=200m,memory=256Mi --limits cpu=500m,memory=1Gi**

deployment.apps/quotesdb resource requirements updated

1. Add a readiness and a liveness probe to the frontend deployment.
   1. Use the oc set probe command with the --readiness option to add the readiness probe that tests the /status path on HTTP port 8000.
   2. [student@workstation ~]$ **oc set probe deployment/frontend --readiness \**
   3. **--get-url http://:8000/status**

deployment.apps/frontend probes updated

* 1. Use the oc set probe command with the --liveness option to add the liveness probe that tests the /env path on HTTP port 8000.
  2. [student@workstation ~]$ **oc set probe deployment/frontend --liveness \**
  3. **--get-url http://:8000/env**

deployment.apps/frontend probes updated

1. Define resource limits for the frontend deployment. Set the CPU request to 200 millicores and the memory request to 256 MiB. Set the CPU limit to 500 millicores and the memory limit to 512 MiB.
2. [student@workstation ~]$ **oc set resources deployment/frontend \**
3. **--requests cpu=200m,memory=256Mi --limits cpu=500m,memory=512Mi**

deployment.apps/frontend resource requirements updated

1. Scale the frontend deployment to three pods.
   1. Scale the deployment.
   2. [student@workstation ~]$ **oc scale deployment/frontend --replicas 3**

deployment.apps/frontend scaled

* 1. Wait for the deployment to scale up. You might have to rerun the command several times for the pods to report a Running status. The names of the pods on your system probably differ.
  2. [student@workstation ~]$ **oc get pods**
  3. NAME READY STATUS RESTARTS AGE
  4. **frontend-86cdd7c7bf-8vrrs** 1/1 **Running** 0 3m10s
  5. **frontend-86cdd7c7bf-ds79w** 1/1 **Running** 0 44s
  6. **frontend-86cdd7c7bf-hpnwz** 1/1 **Running** 0 44s

quotesdb-66ff98b88c-fhwhs 1/1 Running 0 12m

1. Verify that the application responds to web requests.
   1. Retrieve the URL of the application.
   2. [student@workstation ~]$ **oc get route**
   3. NAME HOST/PORT PATH SERVICES ...

frontend **frontend-compreview-scale.apps.ocp4.example.com** frontend ...

* 1. Use the curl command to test the application.
  2. [student@workstation ~]$ **curl \**
  3. **http://frontend-compreview-scale.apps.ocp4.example.com**
  4. <html>
  5. <head>
  6. <title>Quotes</title>
  7. </head>
  8. <body>
  9. <h1>Quote List</h1>
  10. <ul>
  11. <li>1: When words fail, music speaks.
  12. - William Shakespeare
  13. </li>

*...output omitted...*

Hide Solution

**Evaluation**

As the student user on the workstation machine, use the lab command to grade your work. Correct any reported failures and rerun the command until successful.

[student@workstation ~]$ **lab grade compreview-scale**

**Finish**

As the student user 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 compreview-scale**