Extra Review

Comprehensive Review

Goal

Review tasks from *Red Hat OpenShift I: Containers & Kubernetes*

Objectives

- Review tasks from *Red Hat OpenShift I: Containers & Kubernetes*

Sections

- Comprehensive Review

Lab

- Comprehensive Review of Introduction to Containers, Kubernetes, and Red Hat OpenShift

Comprehensive Review

Objectives

After completing this section, you should be able to demonstrate knowledge and skills learned in *Red Hat OpenShift I: Containers & Kubernetes*.

Reviewing Red Hat OpenShift I: Containers & Kubernetes

Before beginning the comprehensive review lab for this course, students should be comfortable with the topics covered in the following chapters.

Chapter 1, Introducing Container Technology

Describe how applications run in containers orchestrated by Red Hat OpenShift Container Platform.

- Describe the difference between container applications and traditional deployments.
- Describe the basics of container architecture.
- Describe the benefits of orchestrating applications and OpenShift Container Platform.

Chapter 2, Creating Containerized Services

Provision a service using container technology.

- Create a database server from a container image.

Chapter 3, Managing Containers

Modify prebuilt container images to create and manage containerized services.

- Manage a container's life cycle from creation to deletion.
- Save container application data with persistent storage.
- Describe how to use port forwarding to access a container.

Chapter 4, Managing Container Images

Manage the life cycle of a container image from creation to deletion.

- Search for and pull images from remote registries.
- Export, import, and manage container images locally and in a registry.

Chapter 5, Creating Custom Container Images

Design and code a Dockerfile to build a custom container image.

- Describe the approaches for creating custom container images.

- Create a container image using common Dockerfile commands.

Chapter 6, Deploying Containerized Applications on OpenShift

Deploy single container applications on OpenShift Container Platform.

- Describe the architecture of Kubernetes and Red Hat OpenShift Container Platform.
- Create standard Kubernetes resources.
- Create a route to a service.
- Build an application using the Source-to-Image facility of OpenShift Container Platform.
- Create an application using the OpenShift web console.

Chapter 7, Deploying Multi-Container Applications

Deploy applications that are containerized using multiple container images.

- Describe considerations for containerizing applications with multiple container images.
- Deploy a multi-container application on OpenShift using a template.

Chapter 8, Troubleshooting Containerized Applications

Troubleshoot a containerized application deployed on OpenShift.

- Troubleshoot an application build and deployment on OpenShift.
- Implement techniques for troubleshooting and debugging containerized applications.

General Container, Kubernetes, and OpenShift Hints

These hints may save some time in completing the comprehensive review lab:

- The `podman` command allows you to build, run, and manage container images. Use the `man podman` command to access Podman documentation. Use the `man podman` *`subcommand`* command to get more information about each subcommand.
- The `oc` command allows you to create and manage OpenShift resources. Use the `man oc` or `oc help` commands to access OpenShift command-line documentation. OpenShift commands that are particularly useful include:

  oc login -u <username> -p <password> <master_api_url>
  Log in to OpenShift as the specified user. Find both credentials and master API URI in the lab page.

  oc new-project
  Create a new project (*namespace*) to contain OpenShift resources.

  oc project
  Select the current project (namespace) to which all subsequent commands apply.

  oc create -f
  Create a resource from a file.

oc process
Processes a template file applying the parameter values to each included resource. Create those resources with the `oc create` command.

oc get
Display the runtime status and attributes of OpenShift resources.

oc describe
Display detailed information about OpenShift resources.

oc delete
Delete OpenShift resources.

- Before mounting any volumes on the Podman and OpenShift host, ensure you apply the correct SELinux context to the directory. The correct context is `container_file_t`. Also, make sure the ownership and permissions of the directory are set according to the `USER` directive in the Dockerfile that was used to build the container being deployed. Most of the time you will have to use the numeric UID and GID rather than the user and group names to adjust ownership and permissions of the volume directory.
- In this classroom, all RPM repositories are defined locally. You must configure the repository definitions in a custom container image (Dockerfile) before running `yum` commands.
- When executing commands in a Dockerfile, combine as many related commands as possible into one `RUN` directive. This reduces the number of image layers in the container image.
- A best practice for designing a Dockerfile includes the use of environment variables for specifying repeated constants throughout the file.

Lab

Containerizing and Deploying a Software Application

In this review, you will containerize a Nexus server, build and test it using Podman, and deploy it to an OpenShift cluster.

Outcomes

You should be able to:

- Write a Dockerfile that successfully containerizes a Nexus server.
- Build a Nexus server container image and deploy it using Podman.
- Deploy the Nexus server container image to an OpenShift cluster.

Before You Begin

Run the set-up script for this comprehensive review.

```
[student@workstation ~]$ lab comprehensive-review start
```

The lab files are located in the `/home/student/DO180/labs/comprehensive-review` directory. The solution files are located in the `/home/student/DO180/solutions/comprehensive-review` directory.

Instructions

Use the following steps to create and test a containerized Nexus server both locally and in OpenShift:

Steps

1. Create a container image that starts an instance of a Nexus server:

   - The `/home/student/DO180/labs/comprehensive-review/image` directory contains files for building the container image. Execute the `get-nexus-bundle.sh` script to retrieve the Nexus server files.
   - Write a Dockerfile that containerizes the Nexus server. The Dockerfile must be located in the `/home/student/DO180/labs/comprehensive-review/image` directory. The Dockerfile must also:
     - Use a base image of `ubi7/ubi:7.7` and set an arbitrary maintainer.
     - Set the environment variable `NEXUS_VERSION` to `2.14.3-02`, and set `NEXUS_HOME` to `/opt/nexus`.
     - Install the *java-1.8.0-openjdk-devel* package

The RPM repositories are configured in the provided `training.repo` file. Be sure to add this file to the container in the `/etc/yum.repos.d` directory.

- Run a command to create a `nexus` user and group. They both have a UID and GID of `1001`.
- Unpack the `nexus-2.14.3-02-bundle.tar.gz` file to the `${NEXUS_HOME}/` directory. Add the`nexus-start.sh` to the same directory.

  Run a command, `ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2`, to create a symlink in the container. Run a command to recursively change the ownership of the Nexus home directory to `nexus:nexus`.
- Make the container run as the `nexus` user, and set the working directory to `/opt/nexus`.
- Define a volume mount point for the `/opt/nexus/sonatype-work` container directory. The Nexus server stores data in this directory.
- Set the default container command to `nexus-start.sh`.

There are two `*.snippet` files in the `/home/student/DO180/labs/comprehensive-review/images` directory that provide the commands needed to create the nexus account and install Java. Use the files to assist you in writing the Dockerfile.

- Build the container image with the name `nexus`.

2. Build and test the container image using Podman with a volume mount:
   - Use the script `/home/student/DO180/labs/comprehensive-review/deploy/local/run-persistent.sh` to start a new container with a volume mount.
   - Review the container logs to verify that the server is started and running.
   - Test access to the container service using the URL: `http://<container IP address>:8081/nexus`.
   - Remove the test container.
3. Deploy the Nexus server container image to the OpenShift cluster. You must:
   - Tag the Nexus server container image as `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest`, and push it the private registry.
   - Create an OpenShift project with a name of `${RHT_OCP4_DEV_USER}-review`.
   - Process the `deploy/openshift/resources/nexus-template.json` template and create the Kubernetes resources.
   - Create a route for the Nexus service. Verify that you can access `http://nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/` from `workstation`.

Evaluation

After deploying the Nexus server container image to the OpenShift cluster, verify your work by running the lab grading script:

```
[student@workstation ~]$ lab comprehensive-review grade
```

Finish

On `workstation`, run the `lab comprehensive-review finish` command to complete this lab.

```
[student@workstation ~]$ lab comprehensive-review finish
```

This concludes the lab.

Solution

Containerizing and Deploying a Software Application

In this review, you will containerize a Nexus server, build and test it using Podman, and deploy it to an OpenShift cluster.

Outcomes

You should be able to:

- Write a Dockerfile that successfully containerizes a Nexus server.
- Build a Nexus server container image and deploy it using Podman.
- Deploy the Nexus server container image to an OpenShift cluster.

Before You Begin

Run the set-up script for this comprehensive review.

```
[student@workstation ~]$ lab comprehensive-review start
```

The lab files are located in the `/home/student/DO180/labs/comprehensive-review` directory. The solution files are located in the `/home/student/DO180/solutions/comprehensive-review` directory.

Instructions

Use the following steps to create and test a containerized Nexus server both locally and in OpenShift:

Steps

1. Create a container image that starts an instance of a Nexus server:

   - The `/home/student/DO180/labs/comprehensive-review/image` directory contains files for building the container image. Execute the `get-nexus-bundle.sh` script to retrieve the Nexus server files.
   - Write a Dockerfile that containerizes the Nexus server. The Dockerfile must be located in the `/home/student/DO180/labs/comprehensive-review/image` directory. The Dockerfile must also:
     - Use a base image of `ubi7/ubi:7.7` and set an arbitrary maintainer.
     - Set the environment variable `NEXUS_VERSION` to `2.14.3-02`, and set `NEXUS_HOME` to `/opt/nexus`.
     - Install the *java-1.8.0-openjdk-devel* package

The RPM repositories are configured in the provided `training.repo` file. Be sure to add this file to the container in the `/etc/yum.repos.d` directory.

- Run a command to create a `nexus` user and group. They both have a UID and GID of `1001`.
- Unpack the `nexus-2.14.3-02-bundle.tar.gz` file to the `${NEXUS_HOME}/` directory. Add the`nexus-start.sh` to the same directory.

  Run a command, `ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} ${NEXUS_HOME}/nexus2`, to create a symlink in the container. Run a command to recursively change the ownership of the Nexus home directory to `nexus:nexus`.
- Make the container run as the `nexus` user, and set the working directory to `/opt/nexus`.
- Define a volume mount point for the `/opt/nexus/sonatype-work` container directory. The Nexus server stores data in this directory.
- Set the default container command to `nexus-start.sh`.

There are two `*.snippet` files in the `/home/student/DO180/labs/comprehensive-review/images` directory that provide the commands needed to create the nexus account and install Java. Use the files to assist you in writing the Dockerfile.

- Build the container image with the name `nexus`.

1.1. Execute the `get-nexus-bundle.sh` script to retrieve the Nexus server files.

```
[student@workstation ~]$ cd /home/student/DO180/labs/comprehensive-review/image
[student@workstation image]$ ./get-nexus-bundle.sh
######################################################################## 100.0%
Nexus bundle download successful
```

1.2. Write a Dockerfile that containerizes the Nexus server. Go to the `/home/student/DO180/labs/comprehensive-review/image` directory and create the Dockerfile.

1.2.1. Specify the base image to use:

```
FROM ubi7/ubi:7.7
```

1.2.2. Enter an arbitrary name and email as the maintainer:

```
FROM ubi7/ubi:7.7

MAINTAINER username <username@example.com>
```

1.2.3. Set a build argument for `NEXUS_VERSION` and an environment variable for `NEXUS_HOME`:

```
FROM ubi7/ubi:7.7

MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus
```

1.2.4. Add the `training.repo` repository to the `/etc/yum.repos.d` directory. Install the Java package using `yum` command.

```
...
ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus

RUN yum install -y --setopt=tsflags=nodocs java-1.8.0-openjdk-devel && \
    yum clean all -y
```

1.2.5. Create the server home directory and service account and group. Make home directory owned by the service account.

```
...
RUN groupadd -r nexus -f -g 1001 && \
    useradd -u 1001 -r -g nexus -m -d ${NEXUS_HOME} -s /sbin/nologin \
            -c "Nexus User" nexus && \
    chown -R nexus:nexus ${NEXUS_HOME} && \
    chmod -R 755 ${NEXUS_HOME}
```

1.2.6. Make the container run as the `nexus` user.

```
...
USER nexus
```

1.2.7. Install the Nexus server software at `NEXUS_HOME` and add the startup script. Note that the `ADD` directive will extract the Nexus files.

Create the `nexus2` symbolic link pointing to the Nexus server directory. Recursively change the ownership of the `${NEXUS_HOME}` directory to `nexus:nexus`.

```
...
ADD nexus-${NEXUS_VERSION}-bundle.tar.gz ${NEXUS_HOME}
ADD nexus-start.sh ${NEXUS_HOME}/

RUN ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} \
          ${NEXUS_HOME}/nexus2
```

1.2.8. Make `/opt/nexus` the current working directory:

```
...
WORKDIR ${NEXUS_HOME}
```

1.2.9. Define a volume mount point to store the Nexus server persistent data:

```
...
VOLUME ["/opt/nexus/sonatype-work"]
```

1.2.10. Set the `CMD` instruction to `["sh", "nexus-start.sh"]`. The completed Dockerfile reads as follows:

```
FROM ubi7/ubi:7.7

MAINTAINER username <username@example.com>

ARG NEXUS_VERSION=2.14.3-02
ENV NEXUS_HOME=/opt/nexus

RUN yum install -y --setopt=tsflags=nodocs java-1.8.0-openjdk-devel && \
    yum clean all -y

RUN groupadd -r nexus -f -g 1001 && \
    useradd -u 1001 -r -g nexus -m -d ${NEXUS_HOME} -s /sbin/nologin \
            -c "Nexus User" nexus && \
    chown -R nexus:nexus ${NEXUS_HOME} && \
    chmod -R 755 ${NEXUS_HOME}

USER nexus

ADD nexus-${NEXUS_VERSION}-bundle.tar.gz ${NEXUS_HOME}
ADD nexus-start.sh ${NEXUS_HOME}/

RUN ln -s ${NEXUS_HOME}/nexus-${NEXUS_VERSION} \
          ${NEXUS_HOME}/nexus2

WORKDIR ${NEXUS_HOME}

VOLUME ["/opt/nexus/sonatype-work"]

CMD ["sh", "nexus-start.sh"]
```

1.3. Build the container image with the name `nexus`.

```
[student@workstation image]$ sudo podman build --layers=false -t nexus .
STEP 1: FROM ubi7/ubi:7.7
Getting image source signatures
...output omitted...
STEP 14: COMMIT ...output omitted...localhost/nexus:latest
...output omitted...
```

2. Build and test the container image using Podman with a volume mount:

- Use the script `/home/student/DO180/labs/comprehensive-review/deploy/local/run-persistent.sh` to start a new container with a volume mount.
- Review the container logs to verify that the server is started and running.

- Test access to the container service using the URL: `http://<container IP address>:8081/nexus`.
- Remove the test container.

2.1. Execute the `run-persistent.sh` script. Replace the container name as shown in the output of the `podman ps` command.

```
[student@workstation images]$ cd /home/student/DO180/labs/comprehensive-review
[student@workstation comprehensive-review]$ cd deploy/local
[student@workstation local]$ ./run-persistent.sh
80970007036bbb313d8eeb7621fada0ed3f0b4115529dc50da4dccef0da34533
```

2.2. Review the container logs to verify that the server is started and running.

```
[student@workstation local]$ sudo podman ps \
> --format="{{.ID}} {{.Names}} {{.Image}}"
81f480f21d47   inspiring_poincare   localhost/nexus:latest
[student@workstation local]$ sudo podman logs inspiring_poincare
...output omitted...
... INFO  [jetty-main-1] ...jetty.JettyServer - Running
... INFO  [main] ...jetty.JettyServer - Started
```

2.3. Inspect the running container to determine its IP address. Provide this IP address to the `curl` command to test the container.

```
[student@workstation local]$ sudo podman inspect \
> -f '{{.NetworkSettings.IPAddress}}' inspiring_poincare
10.88.0.12
[student@workstation local]$ curl -v 10.88.0.12:8081/nexus/ 2>&1 \
> | grep -E 'HTTP|<title>'
> GET /nexus/ HTTP/1.1
< HTTP/1.1 200 OK
  <title>Nexus Repository Manager</title>
```

2.4. Remove the test container.

```
[student@workstation local]$ sudo podman kill inspiring_poincare
81f480f21d475af683b4b003ca6e002d37e6aaa581393d3f2f95a1a7b7eb768b
```

3. Deploy the Nexus server container image to the OpenShift cluster. You must:

- Tag the Nexus server container image as `quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest`, and push it the private registry.
- Create an OpenShift project with a name of `${RHT_OCP4_DEV_USER}-review`.
- Process the `deploy/openshift/resources/nexus-template.json` template and create the Kubernetes resources.
- Create a route for the Nexus service. Verify that you can access `http://nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/` from `workstation`.

3.1. Log in to your Quay.io account.

```
[student@workstation local]$ sudo podman login -u ${RHT_OCP4_QUAY_USER} quay.io
Password: your_quay_password
Login Succeeded!
```

3.2. Publish the Nexus server container image to your `quay.io` registry.

```
[student@workstation local]$ sudo podman push localhost/nexus:latest \
> quay.io/${RHT_OCP4_QUAY_USER}/nexus:latest
Getting image source signatures
...output omitted...
Writing manifest to image destination
Storing signatures
```

3.3. Repositories created by pushing images to `quay.io` are private by default. Refer to the `Repositories Visibility`section of the Appendix C to read details about how change repository visibility.

3.4. Create the OpenShift project:

```
[student@workstation local]$ cd ~/DO180/labs/comprehensive-review/deploy/openshift
[student@workstation openshift]$ oc login -u ${RHT_OCP4_DEV_USER} \
>  -p ${RHT_OCP4_DEV_PASSWORD} ${RHT_OCP4_MASTER_API}
Login successful.
...output omitted...
[student@workstation openshift]$ oc new-project ${RHT_OCP4_DEV_USER}-review
Now using project ...output omitted...
```

3.5. Process the template and create the Kubernetes resources:

```
[student@workstation openshift]$ oc process -f resources/nexus-template.json \
> -p RHT_OCP4_QUAY_USER=${RHT_OCP4_QUAY_USER} \
>  | oc create -f -
service/nexus created
persistentvolumeclaim/nexus created
deploymentconfig.apps.openshift.io/nexus created
[student@workstation openshift]$ oc get pods
NAME             READY   STATUS      RESTARTS   AGE
nexus-1-wk8rv    1/1     Running     1          1m
nexus-1-deploy   0/1     Completed   0          2m
```

3.6. Expose the service by creating a route:

```
[student@workstation openshift]$ oc expose svc/nexus
route.route.openshift.io/nexus exposed.
[student@workstation openshift]$ oc get route -o yaml
apiVersion: v1
items:
- apiVersion: route.openshift.io/v1
  kind: Route
...output omitted...
```

```
  spec:
    host: nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}
...output omitted...
```

3.7. Use a browser to connect to the Nexus server web application at `http://nexus-${RHT_OCP4_DEV_USER}-review.${RHT_OCP4_WILDCARD_DOMAIN}/nexus/`.

Evaluation

After deploying the Nexus server container image to the OpenShift cluster, verify your work by running the lab grading script:

```
[student@workstation ~]$ lab comprehensive-review grade
```

Finish

On `workstation`, run the `lab comprehensive-review finish` command to complete this lab.

```
[student@workstation ~]$ lab comprehensive-review finish
```

This concludes the lab.