Lab: Custom Container Images

Complete the Containerfile for an application that generates a QR code from a given text.

Outcomes

You should be able to:

- Understand multistage builds.
- Run commands within a container.
- Set environment variables.
- Set a working directory.
- Set an entry point.
- Change the user that executes commands.

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

```
[student@workstation ~]$ lab start custom-lab
```

The `start` command copies a Node.js application, which generates a QR code from a given text, to the `labs/custom-lab` directory of your workspace. The command also generates an `.npmrc` file that configures the Node.js application to use an internal NPM registry.

The lab script continuously evaluates the objectives of this lab. Keep the script running in a terminal window and complete the objectives of this lab from a new terminal window.

The application contains a Containerfile that you must complete throughout this exercise. The Containerfile uses a multistage build. The first stage uses the `registry.ocp4.example.com:8443/redhattraining/podman-certificate-generator` image to generate self-signed certificates.

In the second stage, the application uses the certificates to enable a TLS connection.

Instructions

1. Go to the `/home/student/DO188/labs/custom-lab` directory, which contains the application that converts a text into a QR code image. Then, run the app on the host machine by using the `npm install` and `npm start` commands. Verify that the application fails gracefully because an environment variable is missing.

   Go to the `~/DO188/labs/custom-lab` directory.

   ```
   [student@workstation ~]$ cd ~/DO188/labs/custom-lab
   no output expected
   ```

   Install the application dependencies.

   ```
   [student@workstation custom-lab]$ npm install

   added 201 packages, and audited 202 packages in 1s
   ...output omitted...
   ```

   Start the application. The application exits because the HTTP port is not set.

   ```
   [student@workstation custom-lab]$ npm start

   > custom-images-lab@1.0.0 start
   > node index.js

   HTTP PORT not found. Set the env variable to proceed.
   ```

2. In the build stage of the Containerfile, generate the TLS certificates by using the `./gen_certificates.sh` command.

   The `./gen_certificates.sh` command is included in the provided container.

   Use the `RUN` instruction to generate the TLS certificates.

```
FROM registry.ocp4.example.com:8443/redhattraining/podman-certificate-generator as certs

RUN ./gen_certificates.sh

FROM registry.ocp4.example.com:8443/ubi9/nodejs-22:1
USER root
RUN groupadd -r student && useradd -r -m -g student student && \
    npm config set cache /tmp/.npm --global

COPY --from=certs --chown=student:student /app/*.pem /etc/pki/tls/private/certs/
COPY --chown=student:student . /app/
```

3. In the final stage of the Containerfile, set the following environment variables:

   - `TLS_PORT=8443` (the port for TLS traffic)
   - `HTTP_PORT=8080` (the port for HTTP traffic)
   - `CERTS_PATH=/etc/pki/tls/private/certs` (the path that contains the TLS certificates)

   Build the container image with the name `localhost/podman-qr-app`.

   Use the `ENV` instruction to add the environment variables to the Containerfile.

```
FROM registry.ocp4.example.com:8443/redhattraining/podman-certificate-generator as certs

RUN ./gen_certificates.sh

FROM registry.ocp4.example.com:8443/ubi9/nodejs-22:1
USER root
RUN groupadd -r student && useradd -r -m -g student student && \
    npm config set cache /tmp/.npm --global

COPY --from=certs --chown=student:student /app/*.pem /etc/pki/tls/private/certs/
COPY --chown=student:student . /app/

ENV TLS_PORT=8443 \
    HTTP_PORT=8080 \
    CERTS_PATH="/etc/pki/tls/private/certs"
```

   Build the container image.

```
[student@workstation custom-lab]$ podman build -t localhost/podman-qr-app .
...output omitted...
Successfully tagged localhost/podman-qr-app:latest
201...cc8
```

4. In the final stage of the Containerfile, set the working directory of the application to the `/app` path.

   Then, build the container image with the name `localhost/podman-qr-app`.

   Use the `WORKDIR` instruction to define the working directory.

```
FROM registry.ocp4.example.com:8443/redhattraining/podman-certificate-generator as certs

RUN ./gen_certificates.sh

FROM registry.ocp4.example.com:8443/ubi9/nodejs-22:1
USER root
RUN groupadd -r student && useradd -r -m -g student student && \
    npm config set cache /tmp/.npm --global

COPY --from=certs --chown=student:student /app/*.pem /etc/pki/tls/private/certs/
COPY --chown=student:student . /app/

ENV TLS_PORT=8443 \
    HTTP_PORT=8080 \
    CERTS_PATH="/etc/pki/tls/private/certs"

WORKDIR /app
```

Build the container image.

```
[student@workstation custom-lab]$ podman build -t localhost/podman-qr-app .
...output omitted...
Successfully tagged localhost/podman-qr-app:latest
201...cc8
```

5. In the final stage of the Containerfile, set the `student` user as the user that runs the application. The `student` user exists in the Containerfile.

   Then, build the container image with the name `localhost/podman-qr-app`.

   Use the `USER` instruction.

```
FROM registry.ocp4.example.com:8443/redhattraining/podman-certificate-generator as certs

RUN ./gen_certificates.sh

FROM registry.ocp4.example.com:8443/ubi9/nodejs-22:1
USER root
RUN groupadd -r student && useradd -r -m -g student student && \
    npm config set cache /tmp/.npm --global

COPY --from=certs --chown=student:student /app/*.pem /etc/pki/tls/private/certs/
COPY --chown=student:student . /app/

ENV TLS_PORT=8443 \
    HTTP_PORT=8080 \
    CERTS_PATH="/etc/pki/tls/private/certs"

WORKDIR /app

USER student
```

   Build the container image.

```
[student@workstation custom-lab]$ podman build -t localhost/podman-qr-app .
...output omitted...
Successfully tagged localhost/podman-qr-app:latest
201a...ecc8
```

6. In the final stage of the Containerfile, run the `npm install --omit=dev` command to install the production dependencies of the Node.js application.

   Then, build the container image with the name `localhost/podman-qr-app`.

   Use the `RUN` instruction to execute the command.

```
FROM registry.ocp4.example.com:8443/redhattraining/podman-certificate-generator as certs

RUN ./gen_certificates.sh

FROM registry.ocp4.example.com:8443/ubi9/nodejs-22:1
USER root
RUN groupadd -r student && useradd -r -m -g student student && \
    npm config set cache /tmp/.npm --global

COPY --from=certs --chown=student:student /app/*.pem /etc/pki/tls/private/certs/
COPY --chown=student:student . /app/

ENV TLS_PORT=8443 \
    HTTP_PORT=8080 \
    CERTS_PATH="/etc/pki/tls/private/certs"

WORKDIR /app

USER student

RUN npm install --omit=dev
```

Build the container image.

```
[student@workstation custom-lab]$ podman build -t localhost/podman-qr-app .
...output omitted...
Successfully tagged localhost/podman-qr-app:latest
201...cc8
```

7. In the final stage of the Containerfile, make `npm start` the default command for this image. Additional runtime arguments should not override the default command.

   Then, build the container image with the name `localhost/podman-qr-app`.

   Use the `ENTRYPOINT` instruction to execute the command when the container is started.

```
FROM registry.ocp4.example.com:8443/redhattraining/podman-certificate-generator as certs

RUN ./gen_certificates.sh

FROM registry.ocp4.example.com:8443/ubi9/nodejs-22:1
USER root
RUN groupadd -r student && useradd -r -m -g student student && \
    npm config set cache /tmp/.npm --global

COPY --from=certs --chown=student:student /app/*.pem /etc/pki/tls/private/certs/
COPY --chown=student:student . /app/

ENV TLS_PORT=8443 \
    HTTP_PORT=8080 \
    CERTS_PATH="/etc/pki/tls/private/certs"

WORKDIR /app

USER student

RUN npm install --omit=dev

ENTRYPOINT npm start
```

Build the container image.

```
[student@workstation custom-lab]$ podman build -t localhost/podman-qr-app .
...output omitted...
Successfully tagged localhost/podman-qr-app:latest
201...cc8
```

8. Start the `podman-qr-app` container. Call the container `custom-lab` and forward ports `8080` and `8443`.

Use the `podman run` command to start the application and bind the corresponding ports.

```
[student@workstation custom-lab]$ podman run --name=custom-lab \
  -p 8080:8080 -p 8443:8443 podman-qr-app
...output omitted...
TLS Server running on port 8443
Server running on port  8080
```

Optionally, test the application by navigating to `http://localhost:8080` in a web browser.

Finish

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

Press **y** when the `lab start` command prompts you to execute the finish function. Alternatively, execute the following command:

```
[student@workstation ~]$ lab finish custom-lab
```