1.

```
ENV SERVER_PORT=3000
ENV NODE_ENV="production"

WORKDIR /tmp/hello-server
```

> **NOTE**
>
> Setting the `NODE_ENV` environment variable to `production` instructs NPM to ignore the dependencies listed in the `devDependencies` section of the `package.json` file.

Because the `devDependencies` packages are not necessary to run the application, setting the `NODE_ENV` variable to `production` reduces the image size.

The application uses the `SERVER_PORT` environment variable to determine the port to which it binds.

Update the working directory to `/opt/app-root/src`, which is a better location than `/tmp/hello-server`.

```
WORKDIR /opt/app-root/src
```

2. Apply appropriate metadata to the image.

Add a `LABEL` instruction to indicate who is responsible for maintaining the image.

```
FROM registry.ocp4.example.com:8443/ubi9/nodejs-22-minimal:1

LABEL org.opencontainers.image.authors="Your Name"

ENV SERVER_PORT=3000
```

Add further `LABEL` instructions to provide hints as to the version and intended usage of the container image.

```
LABEL org.opencontainers.image.authors="Your Name"
LABEL com.example.environment="production"
LABEL com.example.version="0.0.1"

ENV SERVER_PORT=3000
```

Add an `EXPOSE` instruction to indicate that the application within the container binds to the port defined in the `SERVER_PORT` environment variable.

```
ENV SERVER_PORT=3000
ENV NODE_ENV="production"

EXPOSE $SERVER_PORT

WORKDIR /opt/app-root/src
```

The `EXPOSE` instruction serves for documentation purposes. It does not bind the port on the host running the container.

The final Containerfile contains the following instructions:

```
FROM registry.ocp4.example.com:8443/ubi9/nodejs-22-minimal:1

LABEL org.opencontainers.image.authors="Your Name"
LABEL com.example.environment="production"
LABEL com.example.version="0.0.1"

ENV SERVER_PORT=3000
ENV NODE_ENV="production"

EXPOSE $SERVER_PORT

WORKDIR /opt/app-root/src

COPY . .

RUN npm install

CMD npm start
```

Build the container image by using the updated Containerfile. Use the `hello-server:best` image tag.

```
[student@workstation hello-server]$ podman build -t hello-server:best .
...output omitted...
Successfully tagged localhost/hello-server:best
...output omitted...
```

Inspect the container image to observe the added metadata.

```
[student@workstation hello-server]$ podman inspect hello-server:best \
  -f '{{.Config.Env}}'
...output omitted... SERVER_PORT=3000 NODE_ENV=production]
```

Verify that the application works.

```
[student@workstation hello-server]$ podman run -d --rm --name hello-best \
  -p 3000:3000 hello-server:best
d6c3...9f21
[student@workstation hello-server]$ curl http://localhost:3000/greet ;echo
{"hello":"world"}
[student@workstation hello-server]$ podman stop hello-best
hello-best
```

Change to the home directory.

```
[student@workstation hello-server]$ cd
no output expected
```

Finish

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.

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