=== 1

$$$$$$
ADD command copies and unpacks compressed files. But it turns out that it doesn't support the zip format. It works with other formats like .tar and .tar.gz but it doesn't work with .zip files.

In question 1, you are asked to copy a zip file from the host to the container in the building process. Using the ADD command alone will be useless and will result in the failure of the build.

To unzip the file, we should run the command unzip directly after copying the zip file. Use COPY command and then use RUN to unzip.

```
COPY <file name>.zip <destination dir>
RUN unzip <file name>.zip
```

$$$$$$

Dockerfile will be given in the exam so, we need to modify the same file as per requirements:
---
sudo curl -O https://download.jboss.org/wildfly/20.0.1.Final/wildfly-20.0.1.Final.tar.gz
Note: Image had already downloaded in exam
sudo mkdir jboss-project
sudo cd jboss-project
sudo ls # make sure Dockerfile and their required files must be available here (Dockerfile & wildfly-20.0.1.Final.zip)

#SET USERNAME
#COPY ZIP FILE TO /OPT/JBOSS, USING RUN

There will be a directory inside user's home directory. The zip file is already available in it. (users home dir) - material
Different directory will be containing dockerfile.

^^^^ Sample Dockerfile ^^^^

```
FROM jboss/wildfly:23.0.1.Final
MAINTAINER "your name"
ENV LAUNCH_JBOSS_IN_BACKGROUND true
ENV JBOSS_HOME /opt/jboss/wildfly
USER jboss
COPY wildfly-preview-23.0.1.Final.zip /opt/jboss
RUN unzip wildfly-preview-23.0.1.Final.zip -d /opt/jboss
EXPOSE 8080
CMD ["/opt/jboss/wildfly/bin/standalone.sh", "-b", "0.0.0.0"]
```

^^^^^^ File end here

sudo podman build -t eap/jboss .

sudo podman images
sudo podman run -d --name eap-jboss -p 8080:8080 -p 9990:9990 -p 9999:9999 eap/jboss
sudo podman container ls

Note: open the browser and hit the url : 127.0.0.1:8080 then provide username and password once got the UI

=== 2

Run mysql container inside pod, using script, make script executable
---
sudo mkdir -p do180/mysql
sudo chcon -R -t container_file_t do180/mysql
sudo chown 27:27 do180/mysql
sudo podman run --pod <pod_name> --name <container_name> -d -e MYSQL_DATABASE=wordpress -e MYSQL_ROOT_PASSWORD=redhat -e MYSQL_USER=test1 -e MYSQL_PASSWORD=test1 -v "/home/student/do180/mysql":/var/lib/mysql rhscl/mysql-57-rhel7

=== 3

Run wordpress container inside pod, using script, make script executable
---
sudo podman run --pod <pod_name> --name <container_name> -d -e WORDPRESS_DB_PASSWORD=test1 -e WORDPRESS_DB_USER=test1 -e WORDPRESS_DB_NAME=wordpress -e WORDPRESS_DB_HOST=127.0.0.1:3306 docker.io/wordpress

sudo podman ps -a --pod [or]
sudo podman container ls -a

=== 4
Grab the log from container, using script :
---
sudo podman logs --tail 10 <container_name>

Stop and clean the container:
---
sudo podman container stop <container_name>
sudo podman container rm <container_name>   # use -f for forcefully

=== 5
Tag & push the image into registry:
---
sudo podman images
sudo podman save -o filename.tar registry-name/image-name:tag
sudo podman tag localhost/eap/jboss quay.io/registry-name/jboss:test1
                    <image_name>   <registry_url>

sudo podman images
sudo podman login quay.io
sudo podman push quay.io/registry-name/jboss:test1
<registry_url>

=== 6
Create Openshift Database Server - Template

Create a project named messages-info with the display message "Information Messages"

Create an application from a previously existing Template "POSTGRESQL-EMPHERAL"
For your knowledge the template is given at /home/workspace/acme-postgresql/.yml

Use oc describe to get variables

User - acme
Password - RedHat
DB Host - infodb

=== 7
Create Openshift Application - S2I

In the same project messages-info, create an application messages-info

Use an external template given at a URL and connect it to the existing db server created
in the previous task

DB_USER
DB_PASS
DB_HOST

=== 8
Troubleshoot Openshift Applications

In directory /home/desktop/workspace/acme-troubleshoot/

You will find a .sh file getlogs.sh, this file should provide the logs of the pods running
the application messages-info

Copy the content to /home/desktop/workspace/acme-troubleshoot/tools to the pod messages-info

Run the script tools/native.sh in the running pods of the application messages-info

---
========

Pls Note whatever variable value passed in below solution is just and example, you need to check variable and values .

oc login <API token> -u username

password:

oc new-project message-info --display-name="Information messages"

oc process --parameters postgresql-ephemeral -n message-info

oc new-app --template postgresql-ephemeral -p POSTGRESQL_USER=user -p POSTGRESQL_PASSWORD=redhat -p POSTGRESQL_DATABASE=acmedb -p DATABASE_SERVICE_NAME=infodbservice

oc label all --all app=acme --overwrite

oc label all --all db=infodb --overwrite

Check

oc get all -l app=acme

oc get all -l db=infodb

oc new-app --name message-info acme/message-info -e MYSQL_USER=user -e MYSQL_PASSWORD=pass -e MYSQL_DATABASE=testdb -l app=message-info

oc expose svc/<> --hostname=URL

NOTE >> do not use http:// in the URL. it is not accepted & returns error

Put the remaining path after "http://" in the URL (no quotes required)

To get logs

oc logs -f $(oc get pods -o custom-columns=POD:metadata.name --no-headers -l app=message-info)

and similar way you can use variables for other troubleshooting question

for example:

copy from host to container

oc cp /home/desktop/abc/xyz $(oc get pods -o custom-columns=POD:metadata.name --no-headers -l app=message-info):/tmp

To run script

oc exec -it $(oc get pods -o custom-columns=POD:metadata.name --no-headers -l app=message-info) -- /tmp/tools/abc.sh