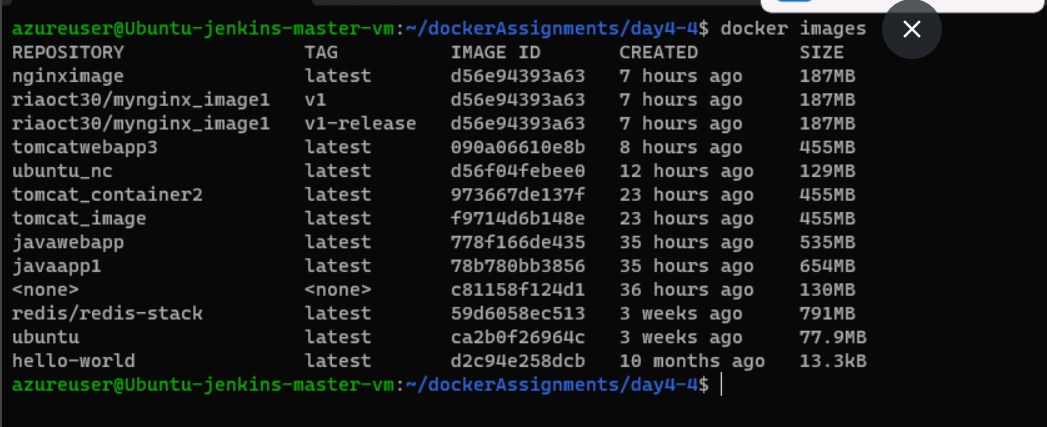
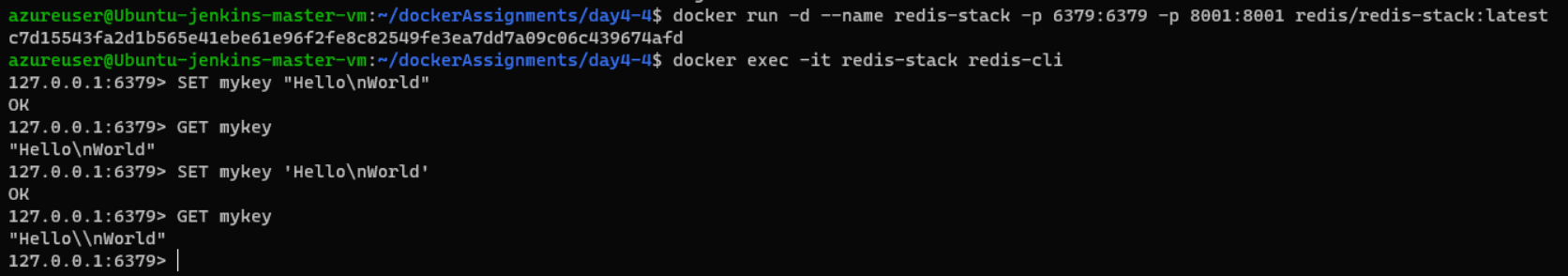
**Name: Sanghamitra Dasgupta**

**Subject: DevOps**

1. **Deploy MySQL docker image as container and use Docker volume to make MySQL data persistent.**
2. **Prepare a custom Docker image and host any example HTML source code inside Docker image and then upload that to Docker hub**
3. **Deploy Jenkins Docker image on Docker host and expose Container using 8080 host port**
4. **Deploy Redis Docker image on Docker Host and access Redis CLI within Container**







1. **Hosting a Java Web application on a Dockerized Tomcat Server**

**Objective:**

**To build an Image to host a java web application on a dockerized Tomcat server.**

**Download the war file from here https://labadmin.hcl-edtech.com/uploads/File/p10874/HelloWorld.war**

**Steps**

**1. Create a new directory**

**2. Copy the given war file and paste it inside the directory**

**3. Create a dockerfile and copy the contents below.**

**FROM tomcat:7.0-jre7**

**COPY HelloWorld.war /usr/local/tomcat/webapps/myapp.war**

**WORKDIR /usr/local/tomcat/bin**

**CMD ["sh","catalina.sh","run"]**

**4. Open a terminal and move to the created directory -**

**5. Now run the build command.**

**docker build --tag=java-web-app:v0.1 .**

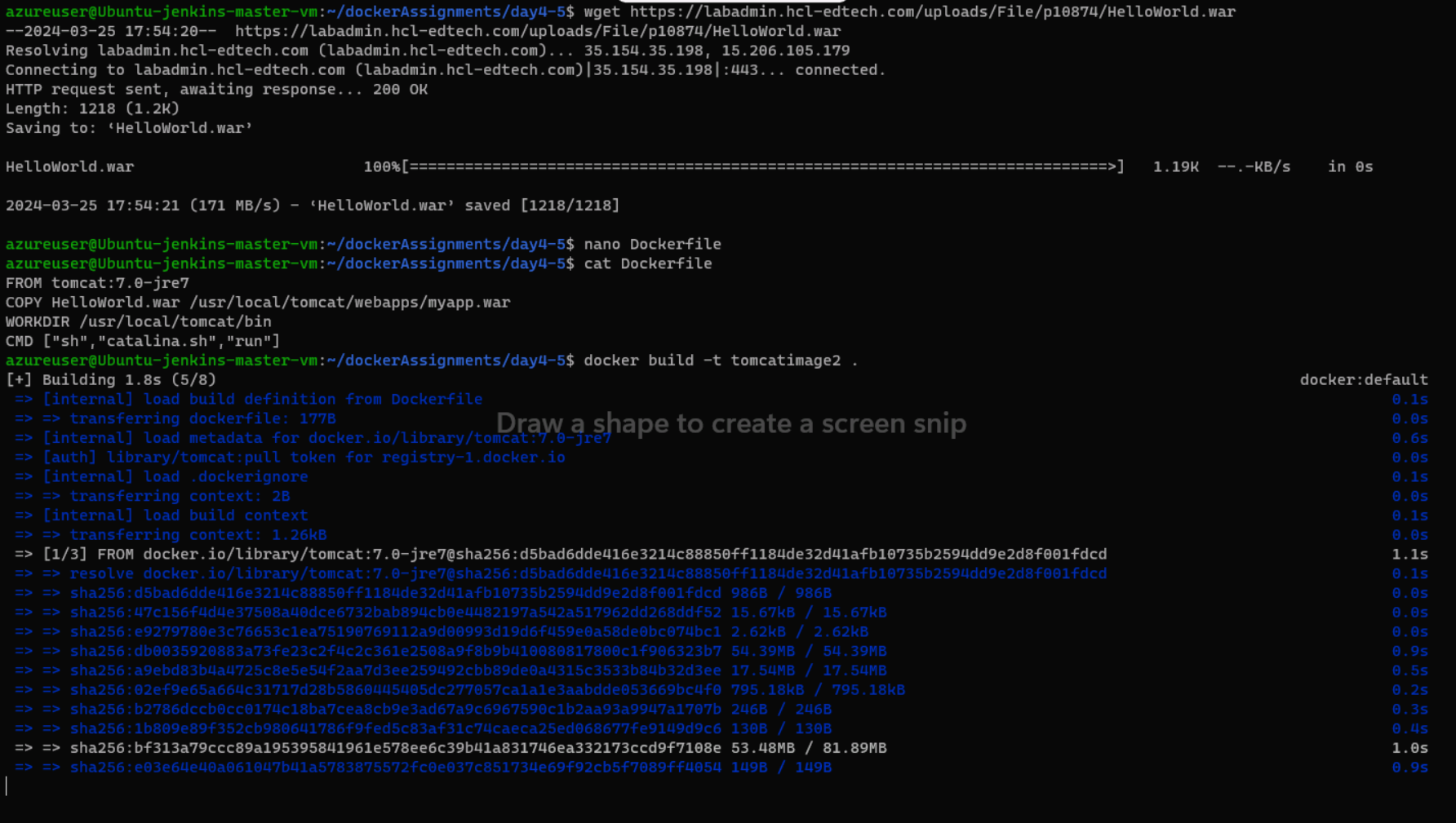
**This command is used to build a Docker image with the name java-web-app and the tag v0.1.**

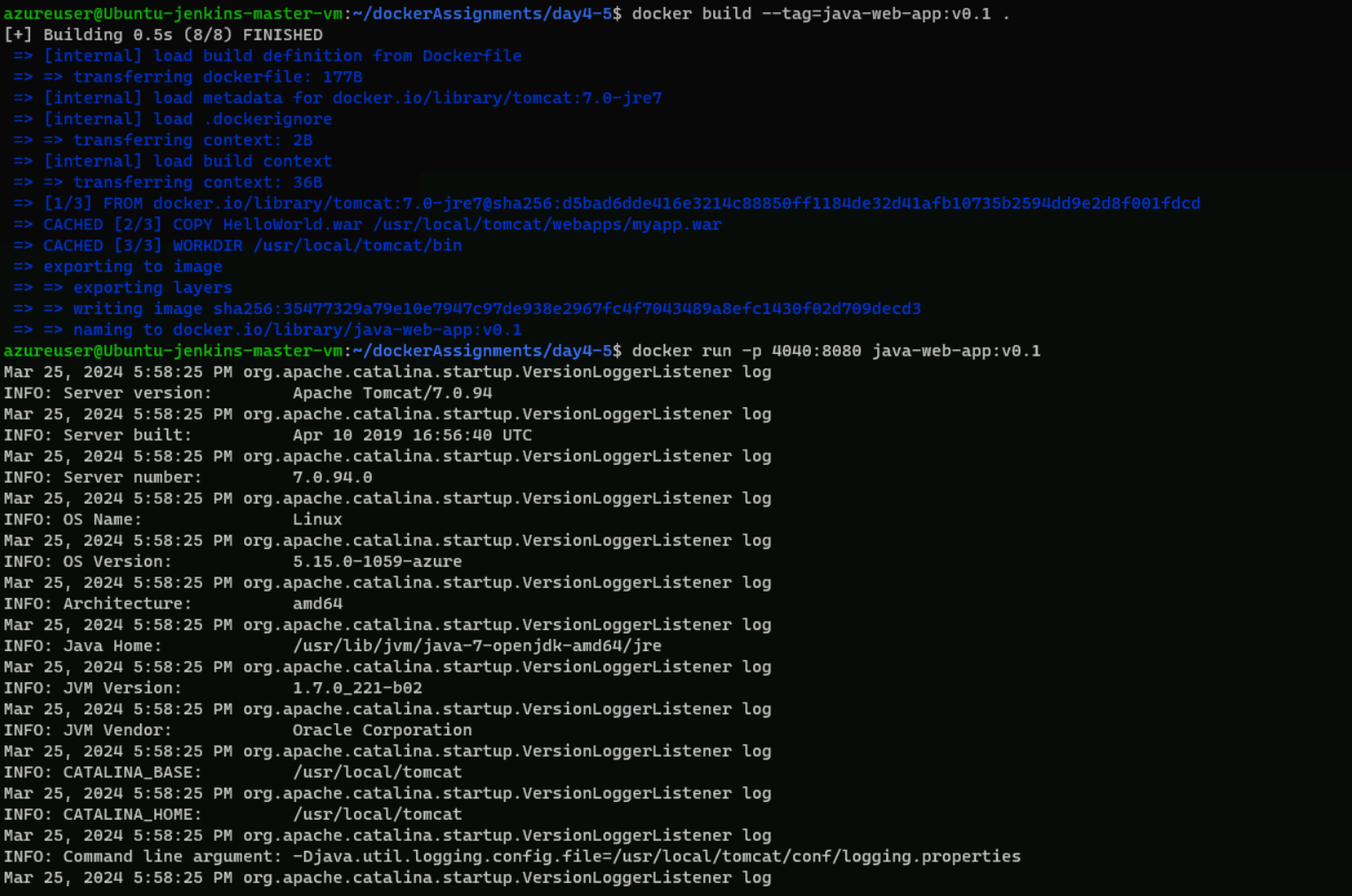
**6. Run the app, mapping your machine’s port 4040 to the container’s published port 8080 using -p:**

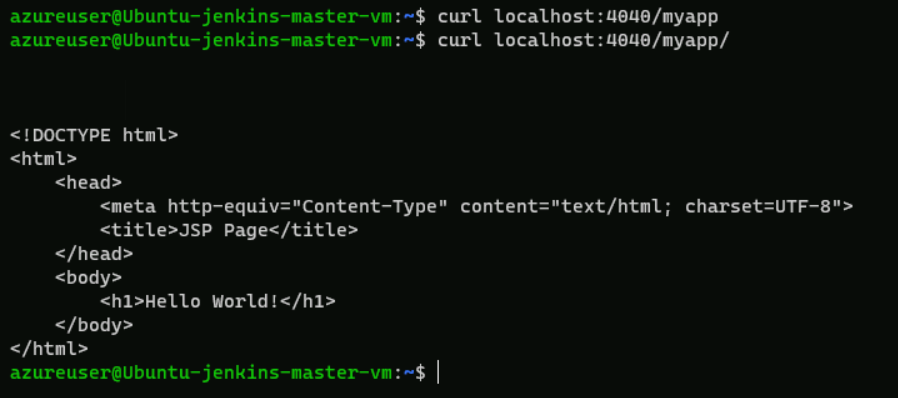
**docker run -p 4040:8080 java-web-app:v0.1**

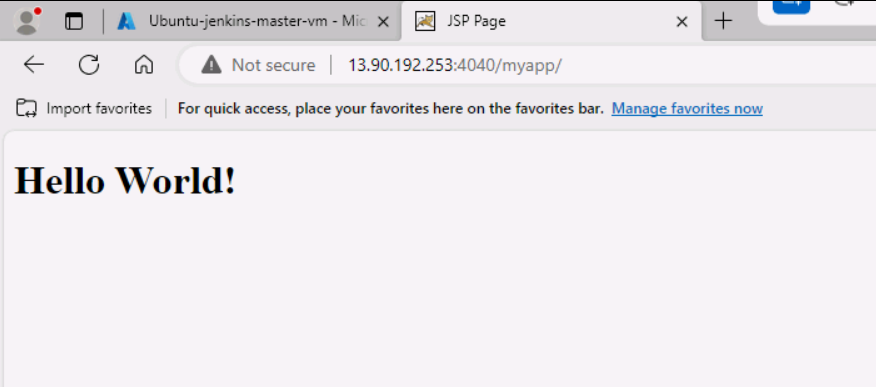
**The tomcat is now started and it servers on URL http://localhost:4040/ and the app is available on http://localhost:4040/myapp**

**7. Open a browser and goto the url http://localhost:4040/myapp**









1. **Docker - Publishing an image in Docker hub**

**Objective:**

**To build an Image publish it in docker hub and pull from docker hub.**

**1. Use any of the dokcer image built above**

**2. Log in with your Docker ID. If you don’t have a Docker account, sign up for one at hub.docker.com. Make note of your username. Log in to the Docker public registry on your local machine.**

**docker login**

**3. Enter the username and password to login using the docker account.**

**4. Tag the image. The notation for associating a local image with a repository on a registry is username/repository:tag**

**The tag is optional, but recommended, since it is the mechanism that registries use to give Docker images a version.**

**Give the repository and tag meaningful names for the context.**

**Now, put it all together to tag the image.**

**5. Run docker tag image with your username, repository, and tag names so that the image uploads to your desired destination.**

**The syntax of the command is:**

**docker tag image:tag username/repository:tag**

**docker tag java-web-app:v2.0 username/java-web-app:v2**

**Replace the username with your actual username**

**6. Now, get the list of all images**

**docker images**

**7. Publish the image. Upload your tagged image to the repository:**

**docker push username/repository:tag**

**Now the docker image is pushed to the docker hub repository.**

**You can now go to docker hub website and see the repository.**

**Once a repository is created, you can push new versions by modifying the tag.**

**8. Pull and run the image from the remote repository**

**From now on, you can use docker run and run your app on any machine with this command:**

**docker run -p 4000:8080 username/repository:tag**

**If the image isn’t available locally on the machine, Docker pulls it from the repository.**

**No matter where docker run executes, it pulls your image, along with Tomcat and all the dependencies, and runs your code.**

**It all travels together in a neat little package, and you don’t need to install anything on the host machine for Docker to run it.**

