Module 8: Containerization, Docker, and Docker Hub

Abhijit Lalasaheb Zende

1. L1 – Create Docker file and build the docker container Application Image for the application build in Jenkins Module

Ans.

(*** Note: Screen shots attached to end of each question ***)

Step 1: Start the Jenkins slave node that we created in the Jenkins module assignment:

Boot up the EC2 instance configured as the Jenkins slave node. This instance, set up in our master-slave architecture, is necessary for executing the builds we have scheduled on the slave, as directed by the Jenkins master server.

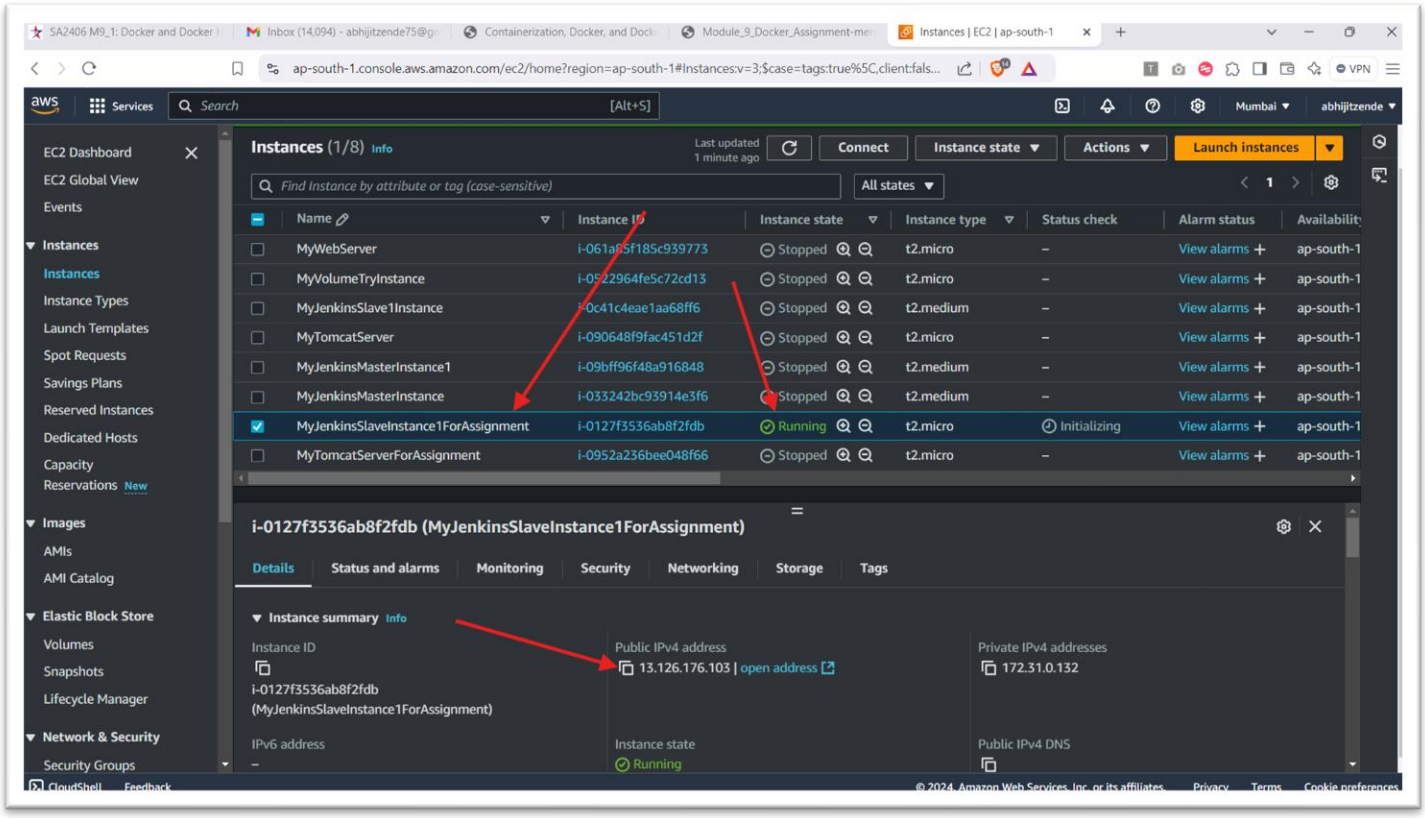


Fig. 1.01: Starting the Jenkins slave node EC2 instance

Step 2: SSH into the started instance:

Launch MobaXterm and gain root privileges. Then, switch to the 'devopsadmin' user account we previously created. This account has access to the build artifacts, which are stored in the 'workspace/sample_pipeline-project/target/' directory. In this location, you will find the WAR files generated during our application's build process.

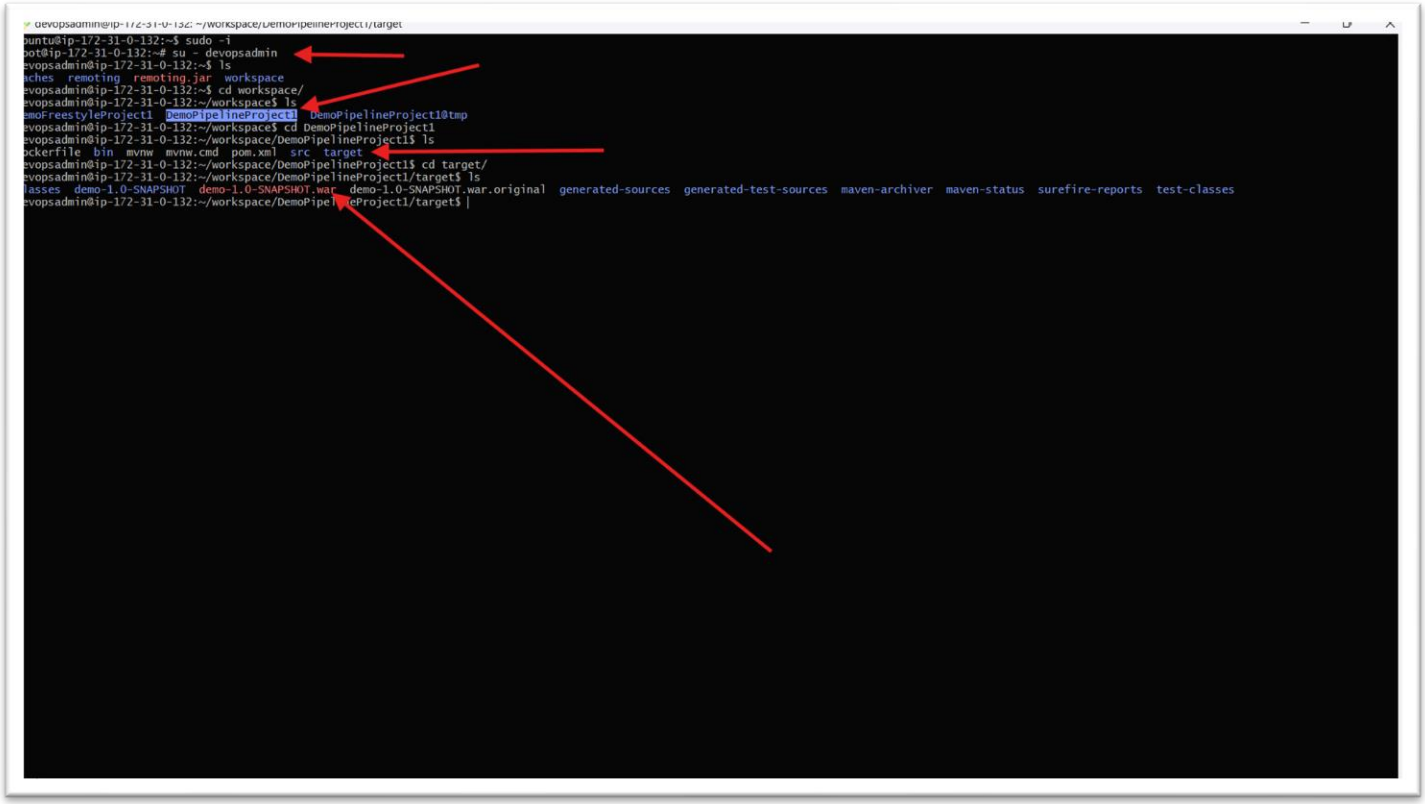


Fig. 1.02: Project Preview

Step 3: Install Docker:

Install Docker by switching to root and running `apt install docker.io`. Verify the installation by checking the Docker version. Grant the 'devopsadmin' user Docker access by adding them to the docker group with `usermod -aG docker devopsadmin`. With Docker set up, we're now ready to create our application image by packaging the app and its dependencies using a Dockerfile.

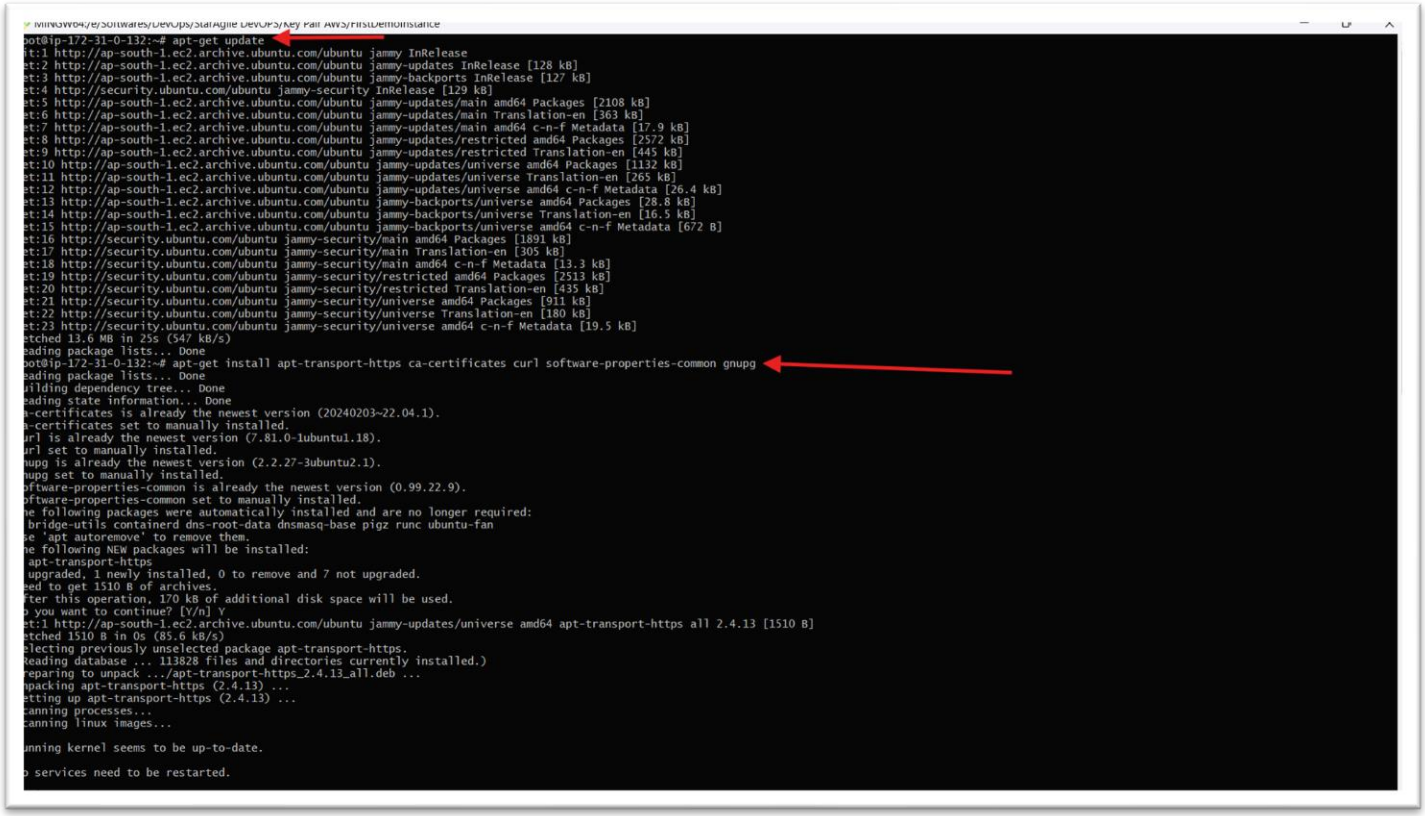


Fig. 1.03: Update packages

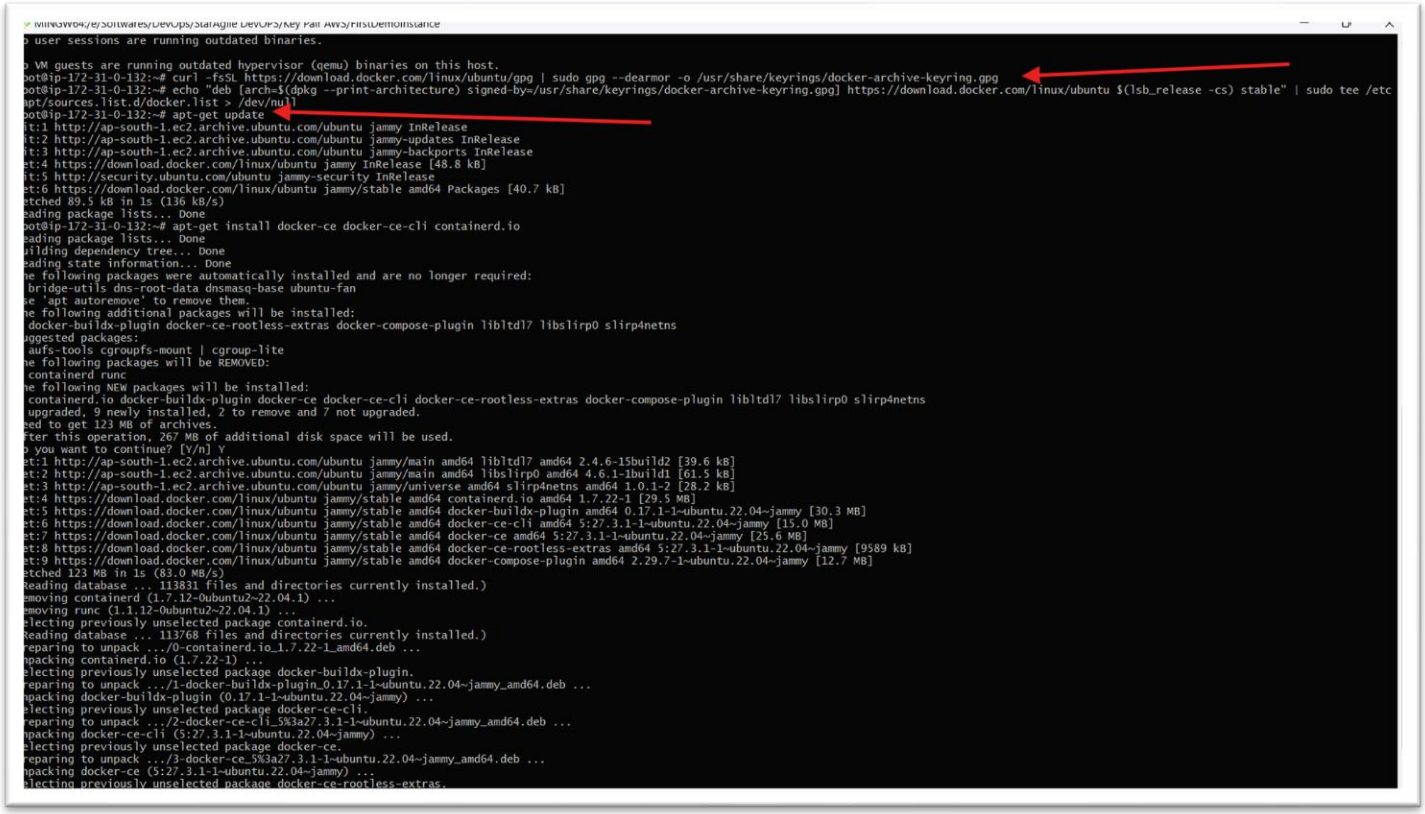


Fig. 1.04: Add Docker's Official GPG Key and Add Docker Repository

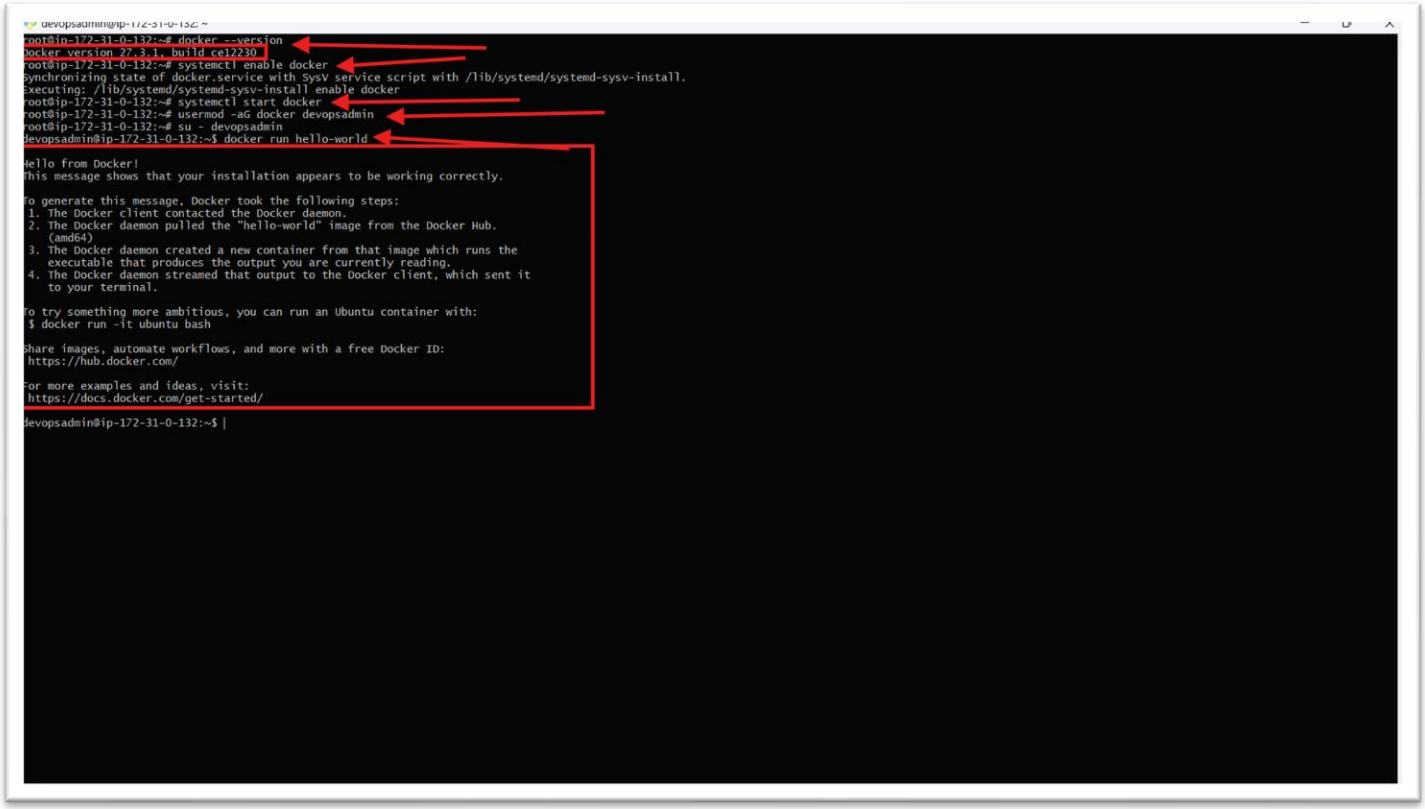


Fig. 1.05: Install Docker Engine and enable it on startup

Step 4: Create Dockerfile for docker image:

Change to the DemoPipelineProject1 directory. Create a Dockerfile using 'vi Dockerfile'. In this file, use the 'FROM' instruction to specify the base image, 'COPY' to transfer WAR files from their source to the target location within the image, and 'EXPOSE' to define the port for internet access. After adding these instructions, save the Dockerfile.

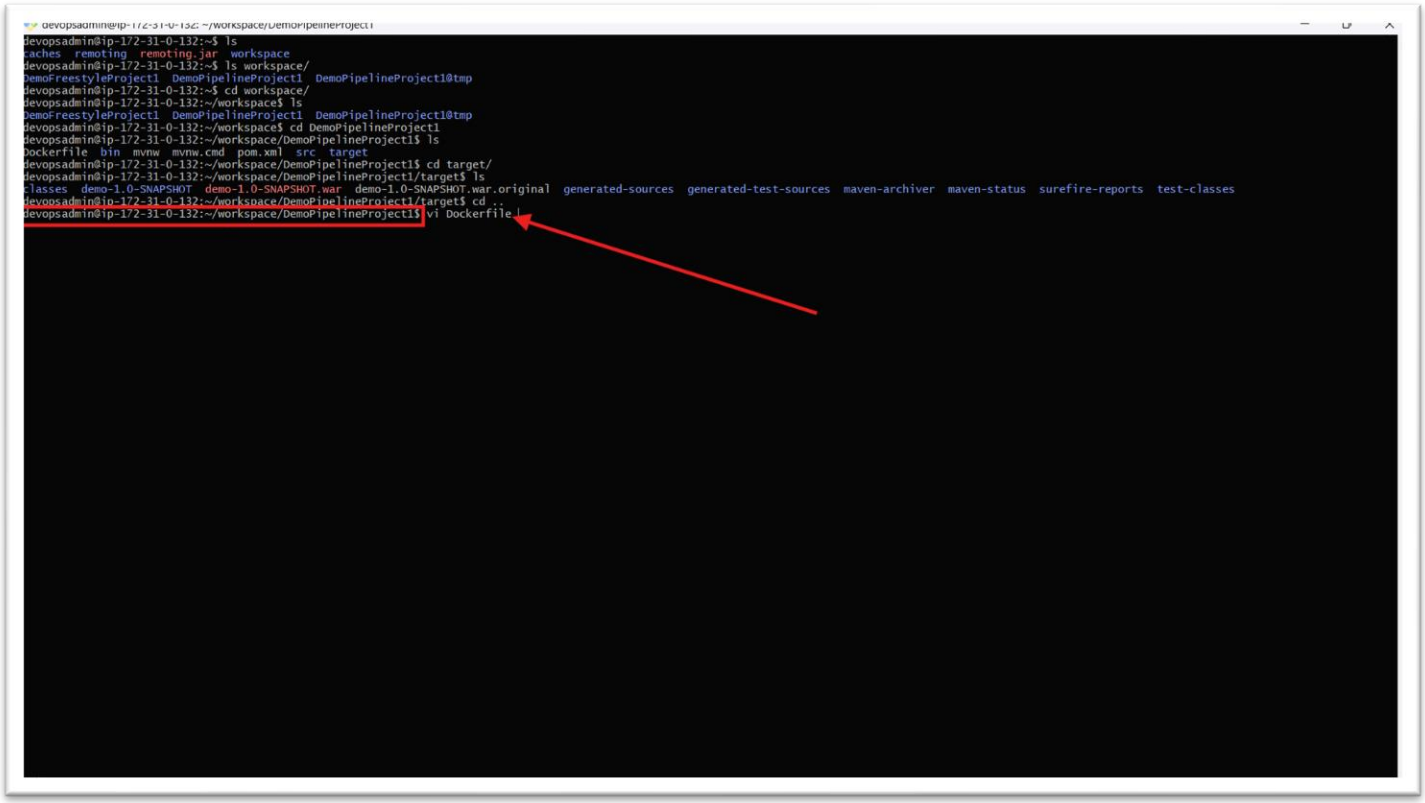


Fig. 1.06: Navigation to Dockerfile

Fig. 1.07: Contents of Dockerfile

Step 5: Build the docker image:

Execute the command `docker build -t abhiz2411/tomcat_web_img:8.1 .` to create a new Docker image based on our Dockerfile. In this command, 'abhiz2411' is the DockerHub registry, 'tomcat_web_img:8.1' specifies the image name and its tag/version, and the final '.' indicates that the Dockerfile is in the current directory.

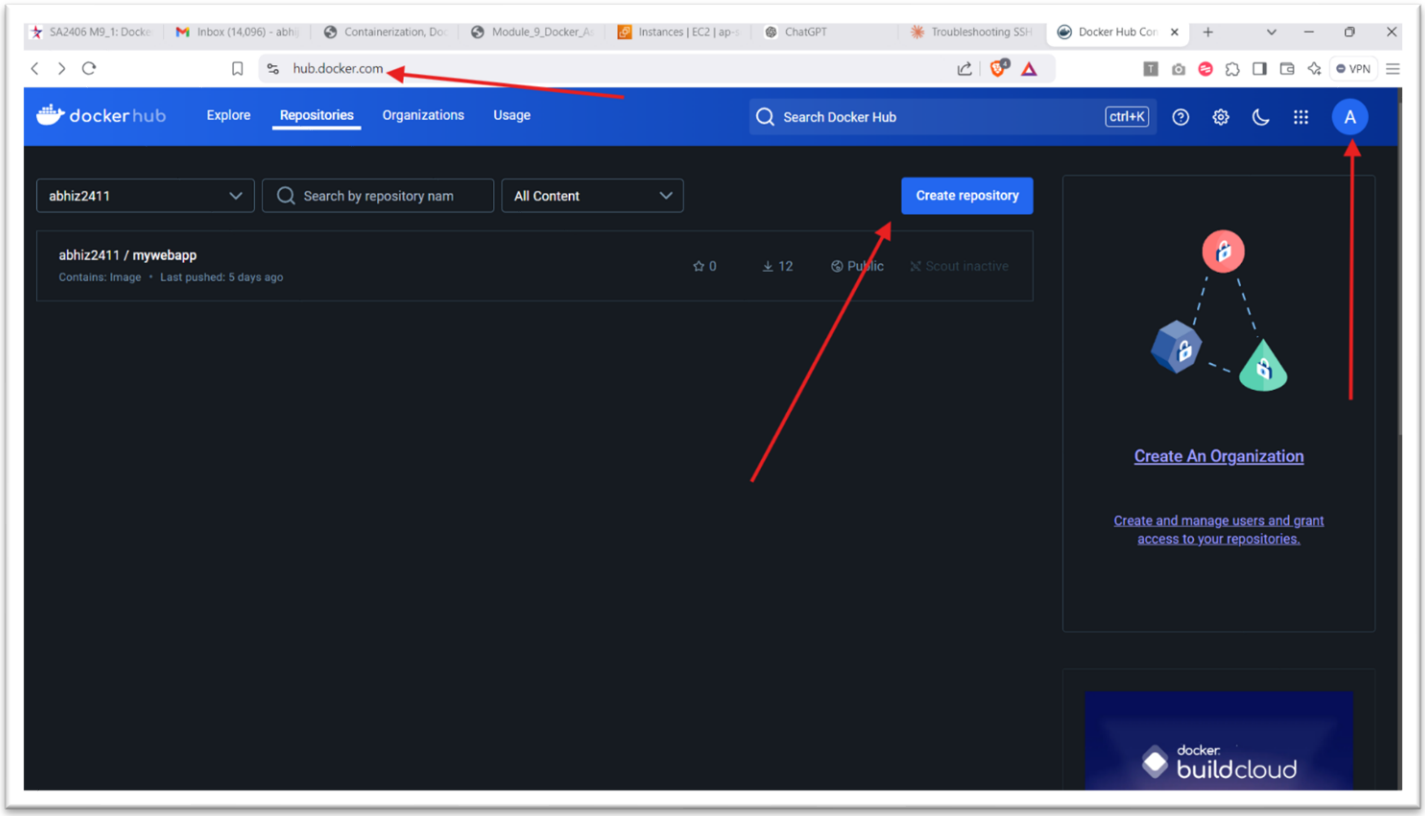


Fig. 1.08: Create new repository on Docker Hub

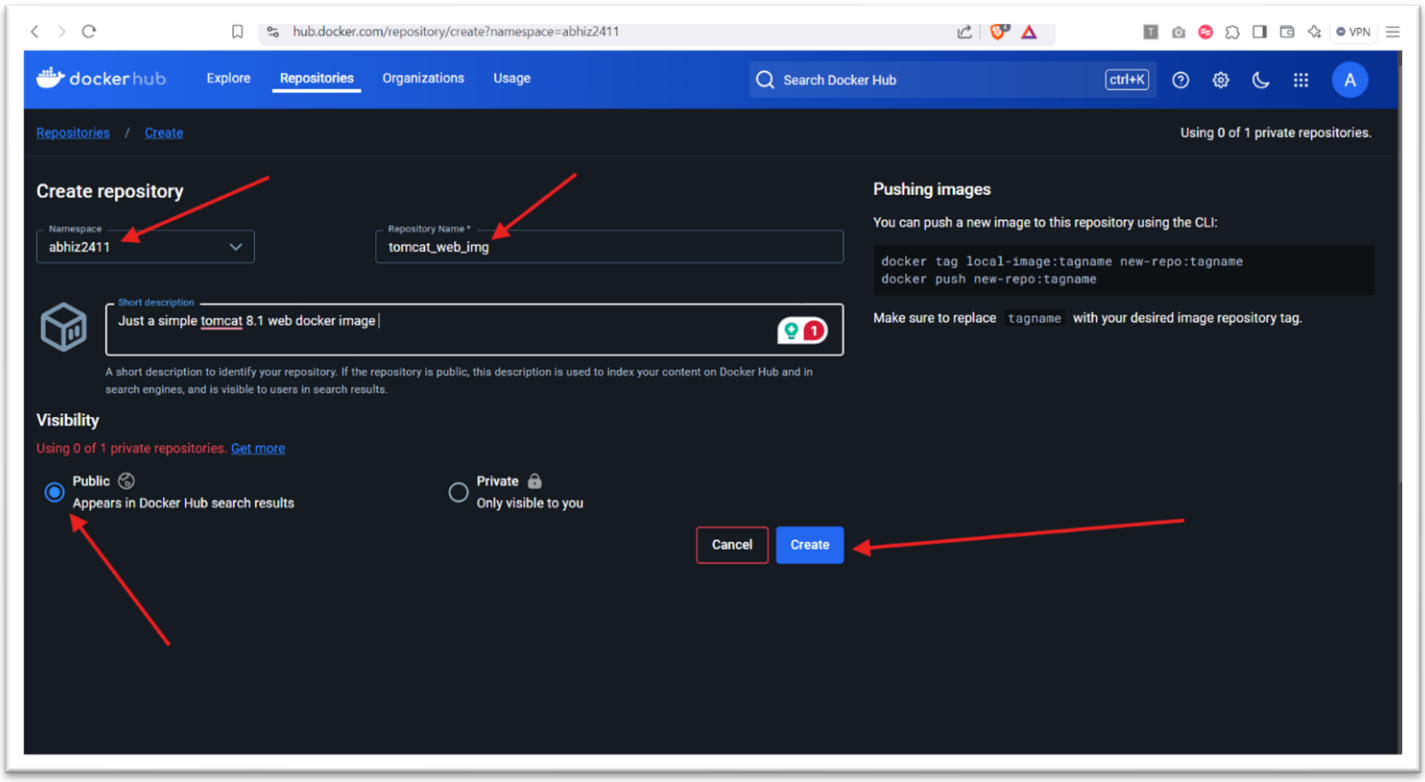


Fig. 1.09: Contents of Docker Hub repository

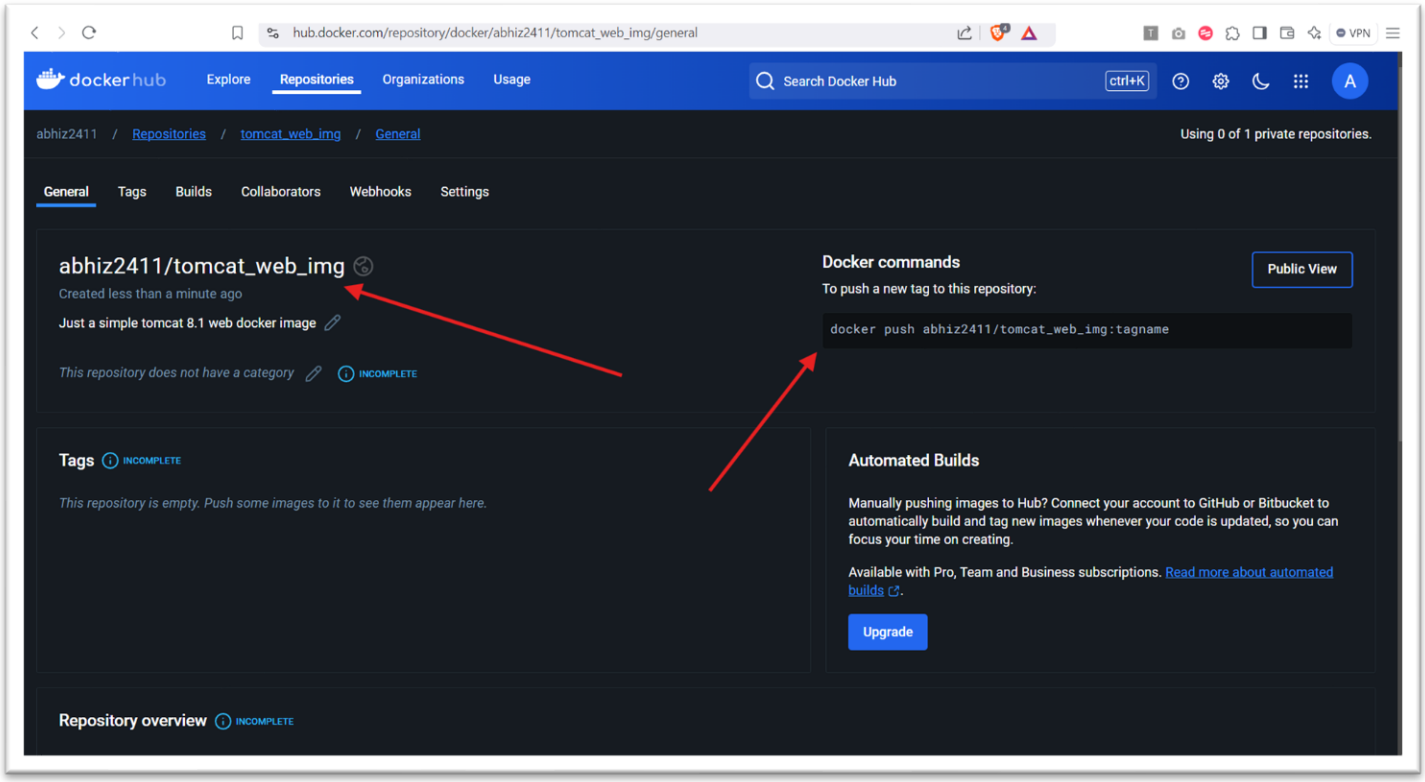


Fig. 1.10: Successful Docker Hub repository creation

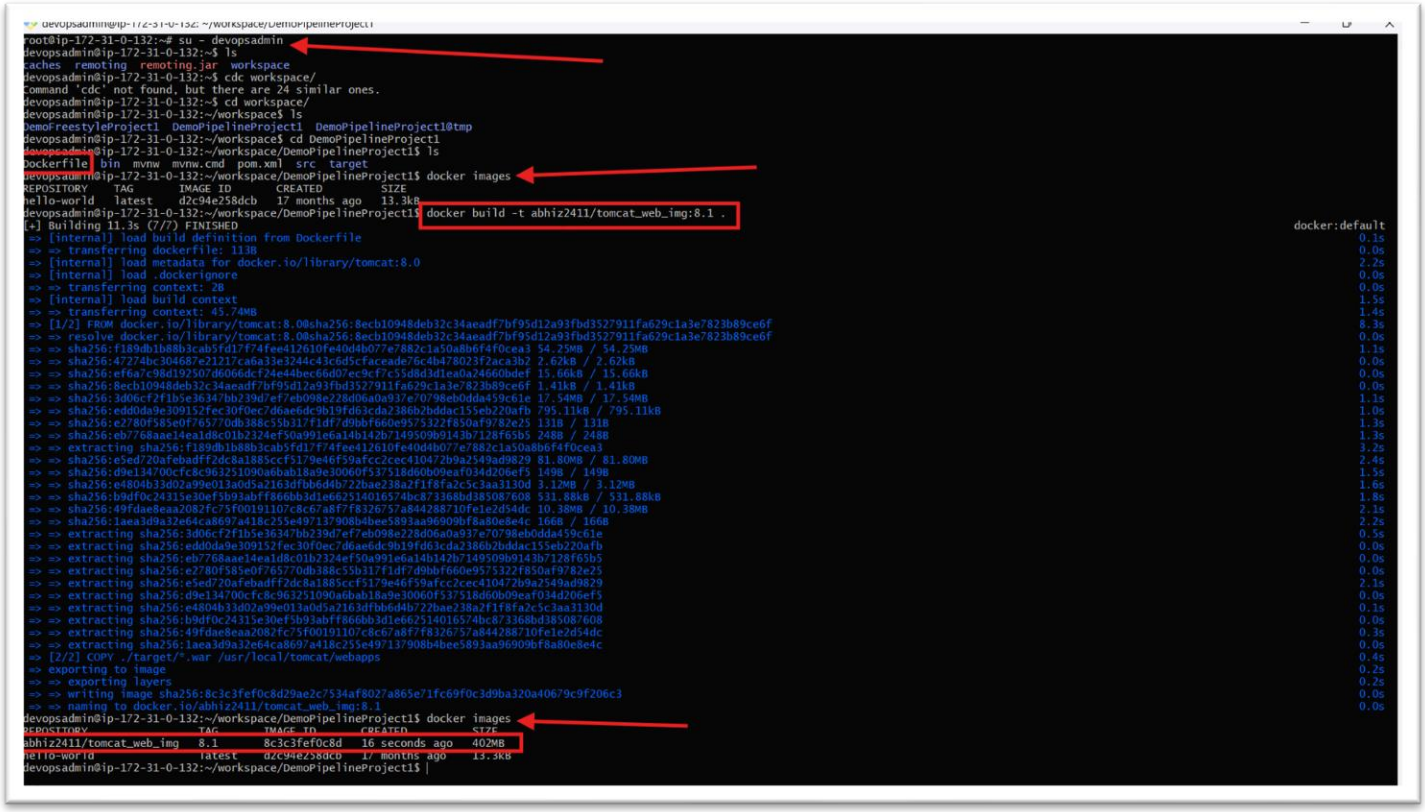


Fig. 1.11: Build Docker image

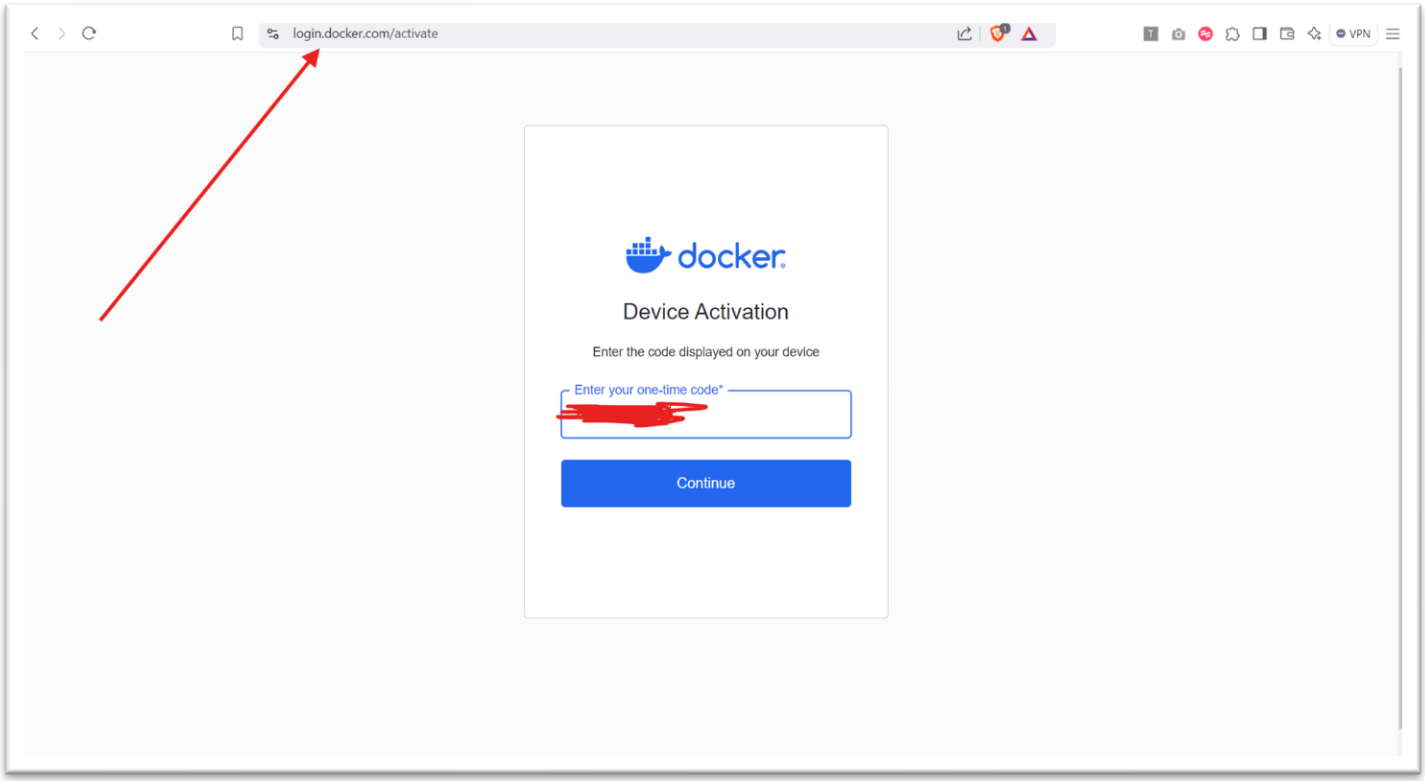


Fig. 1.12: Login Docker on the machine using command `docker login`

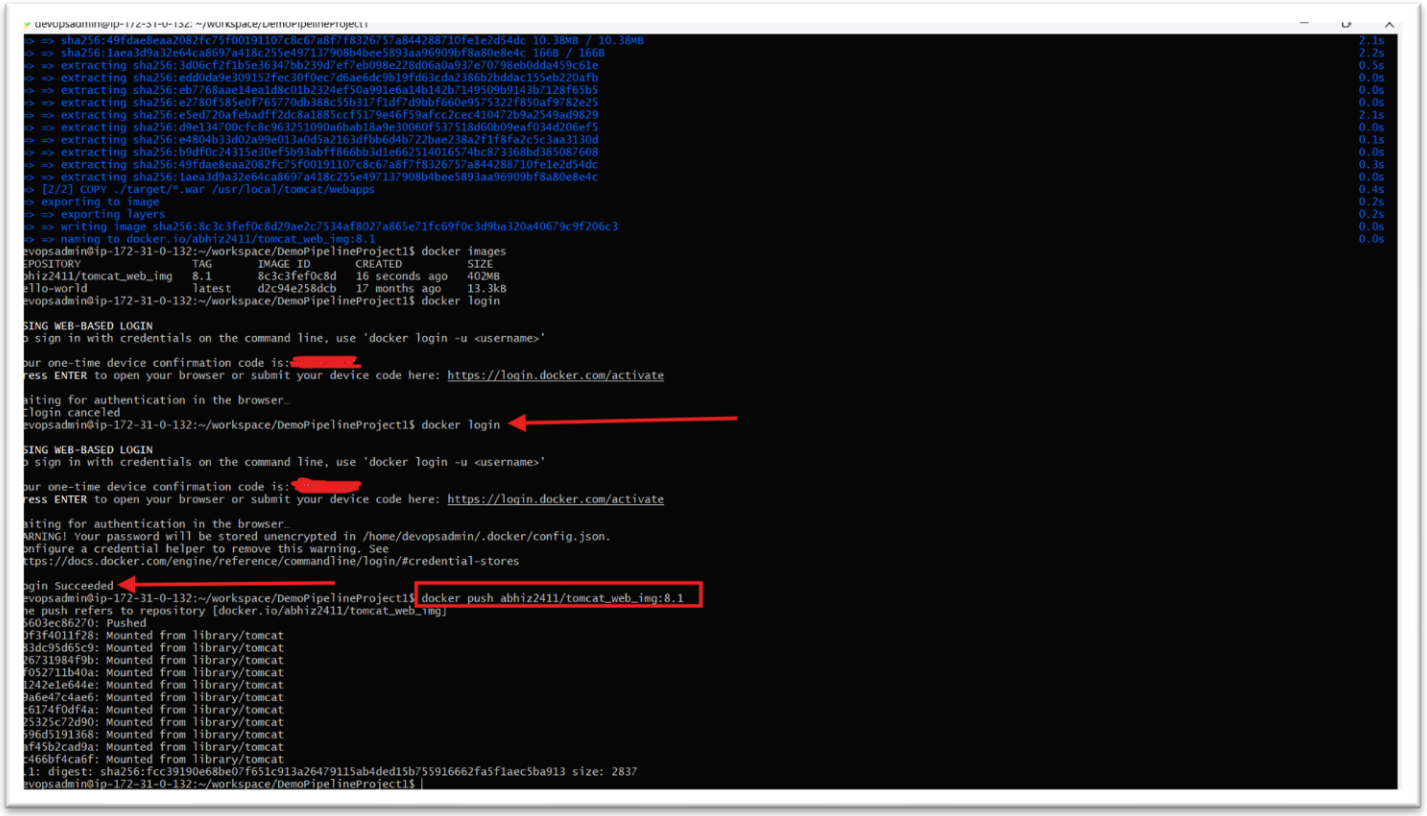


Fig. 1.13: Pushing Docker image to Docker Hub

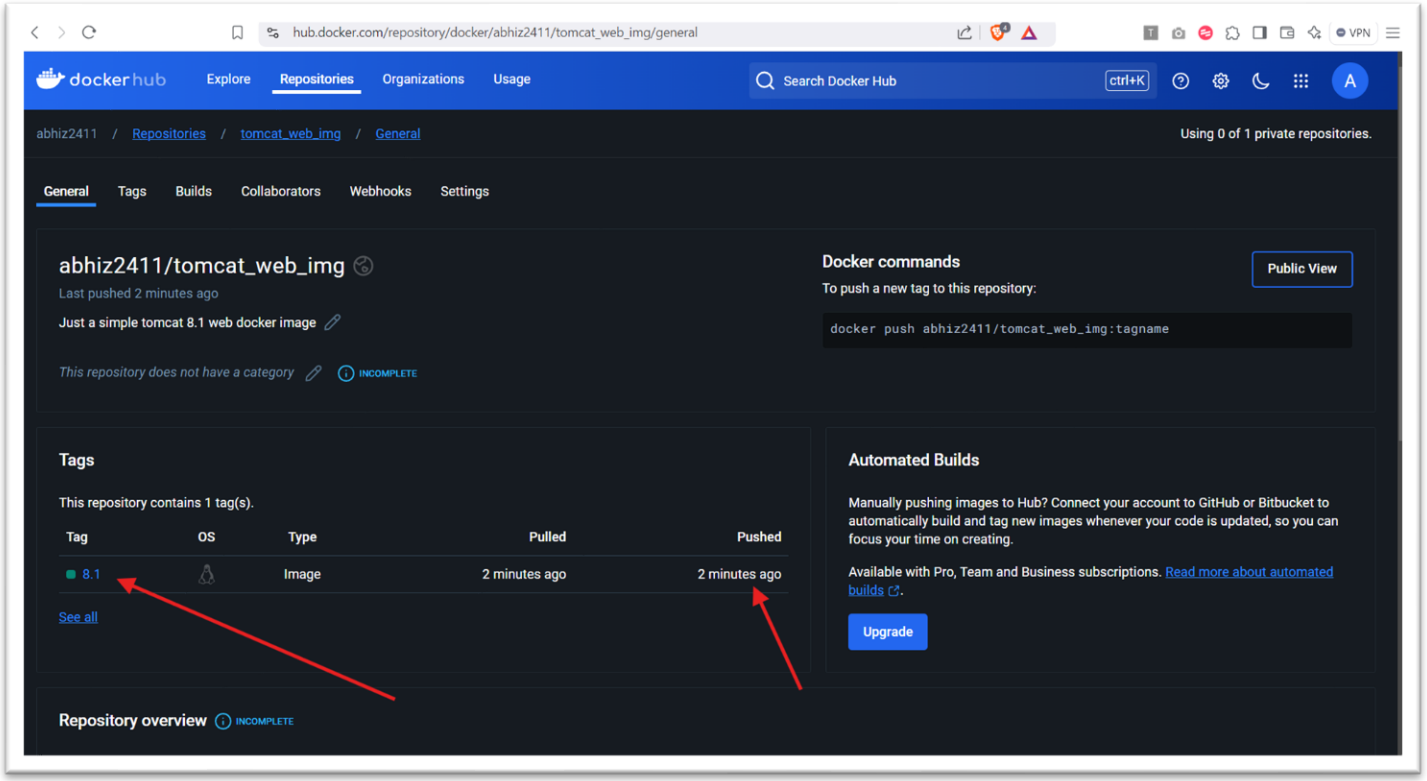


Fig. 1.14: Web view of the docker image we pushed

2. L2 - Create the Container using the same Application Image and run the application in a Web Browser using container port mapping

Ans.

Step 1: Creating and running a Docker container from the image:

To create and run a container from our newly built image, use the command `docker run -it [image_name:tag]`. This starts the container in interactive mode. To make the application accessible via web browser, we need to bind the container's port to a host port. Use the command `docker run -d -p [host_port]:[container_port] [image_name:tag]`. This maps the container's port to a port on the host, allowing web browser access to the application.

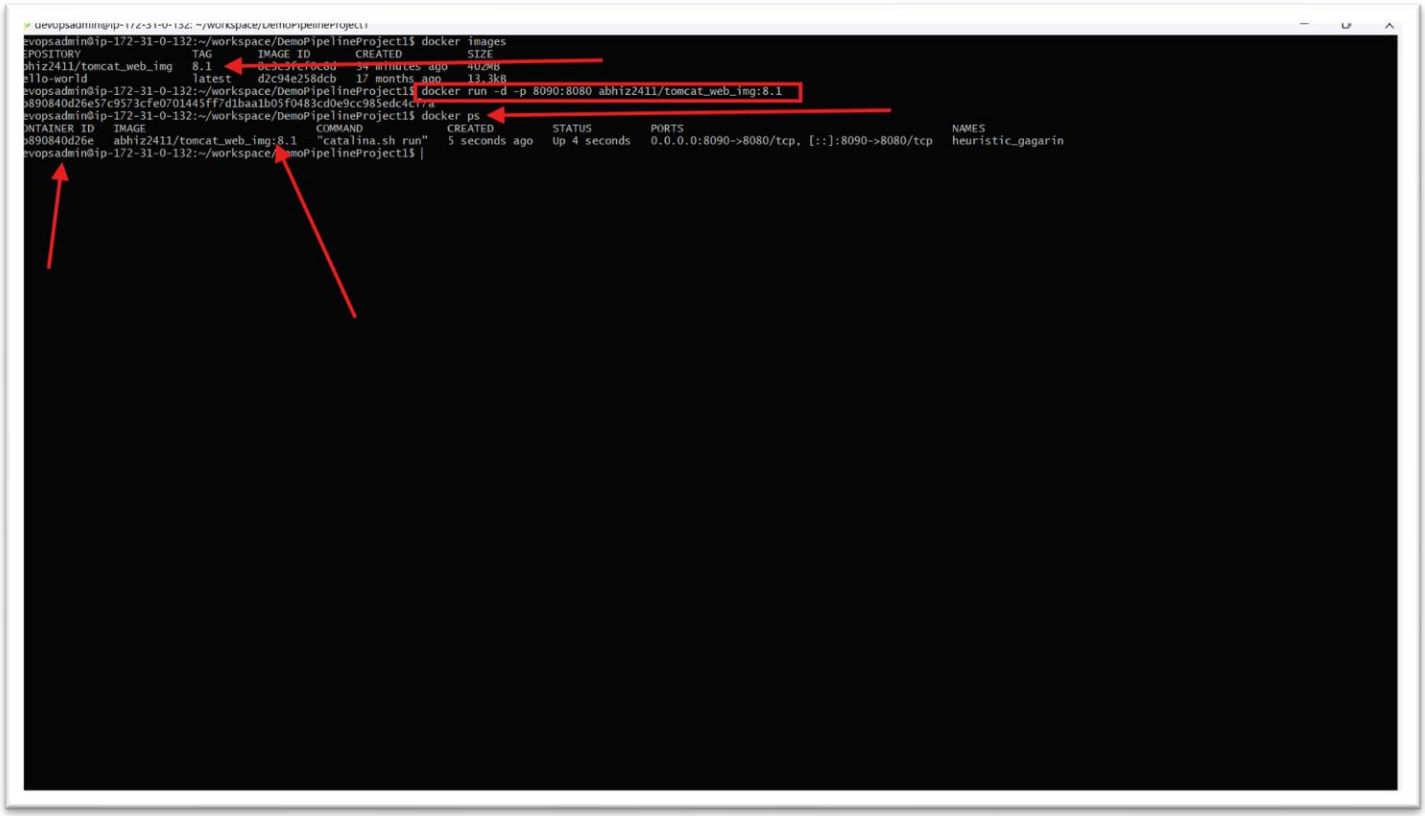


Fig. 2.01: Running docker container and port mapping

Step 2: Verifying container creation and configuring security group for web access:

Confirm the container is running by using `docker ps -a` to list all containers. To access the web application from a browser, modify the instance's security group. Edit the inbound rules, adding a new rule that either specifies ports 8080-8089 or allows all traffic. Save these changes to enable external access to your containerized application.

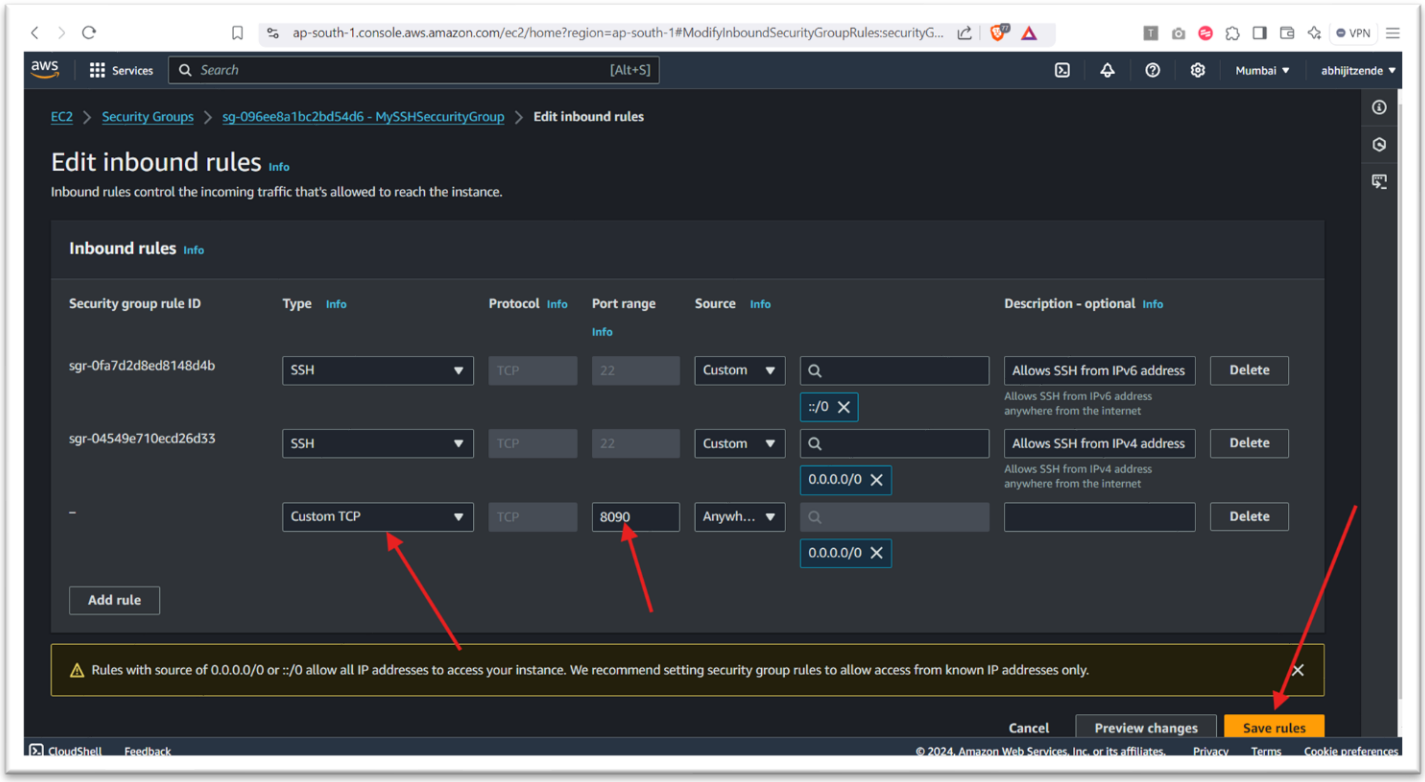


Fig. 2.02: Edit inbound rule of attached security group to allow traffic to host port

Step 3: Viewing the application in the browser:

Copy the EC2 instance's public IP address, open a browser, and enter the IP along with the Host Port and the path to the WAR file to access the application.

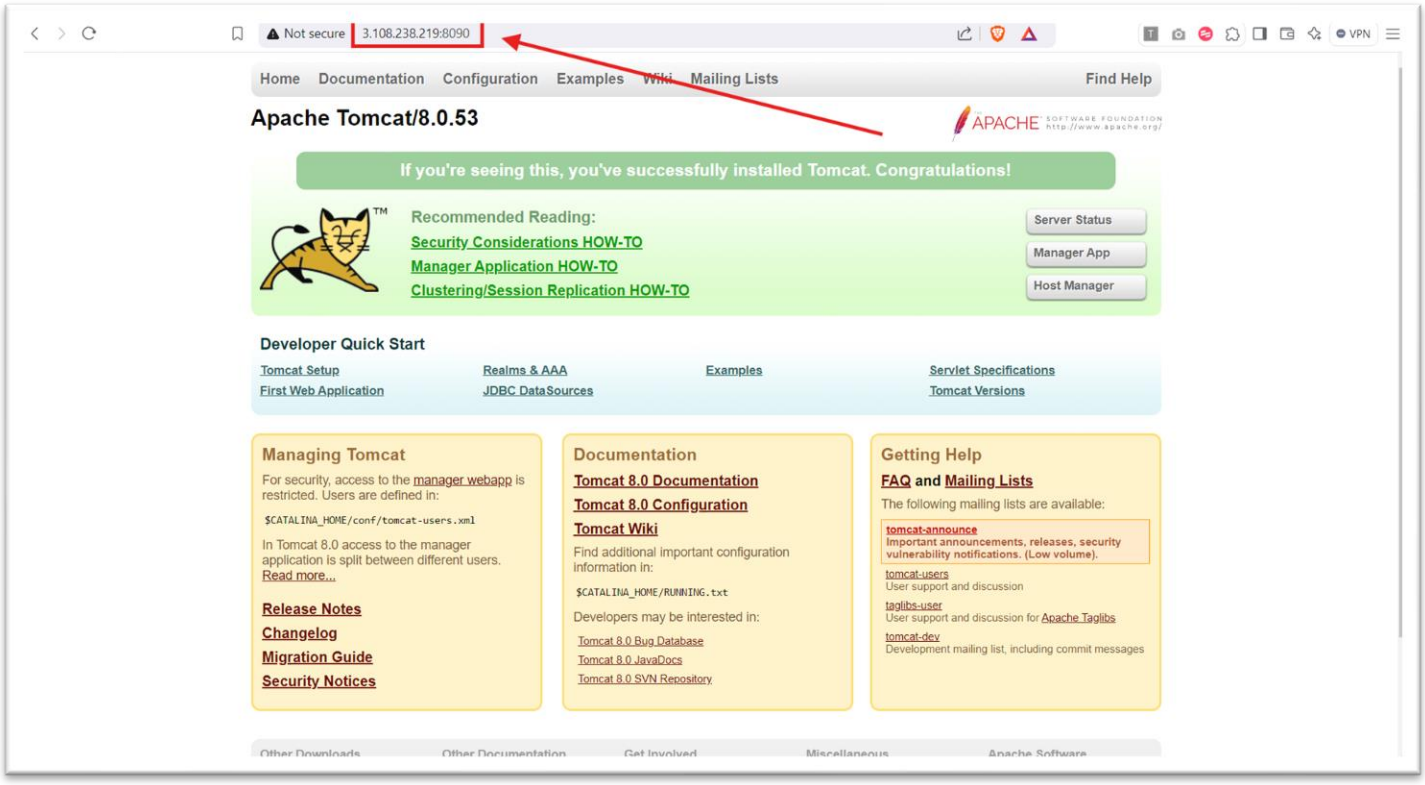


Fig. 3.01: Web view of tomcat application

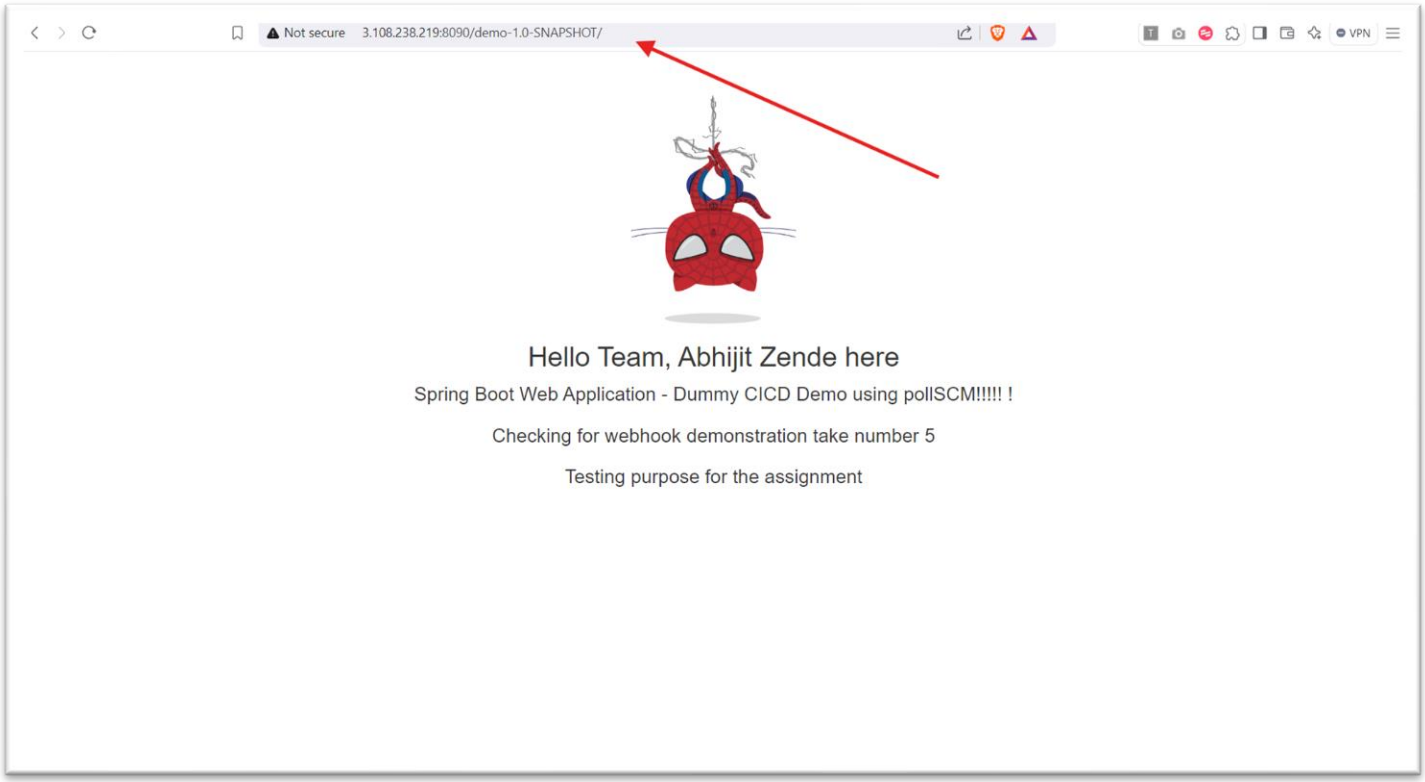


Fig. 3.02: Application web preview

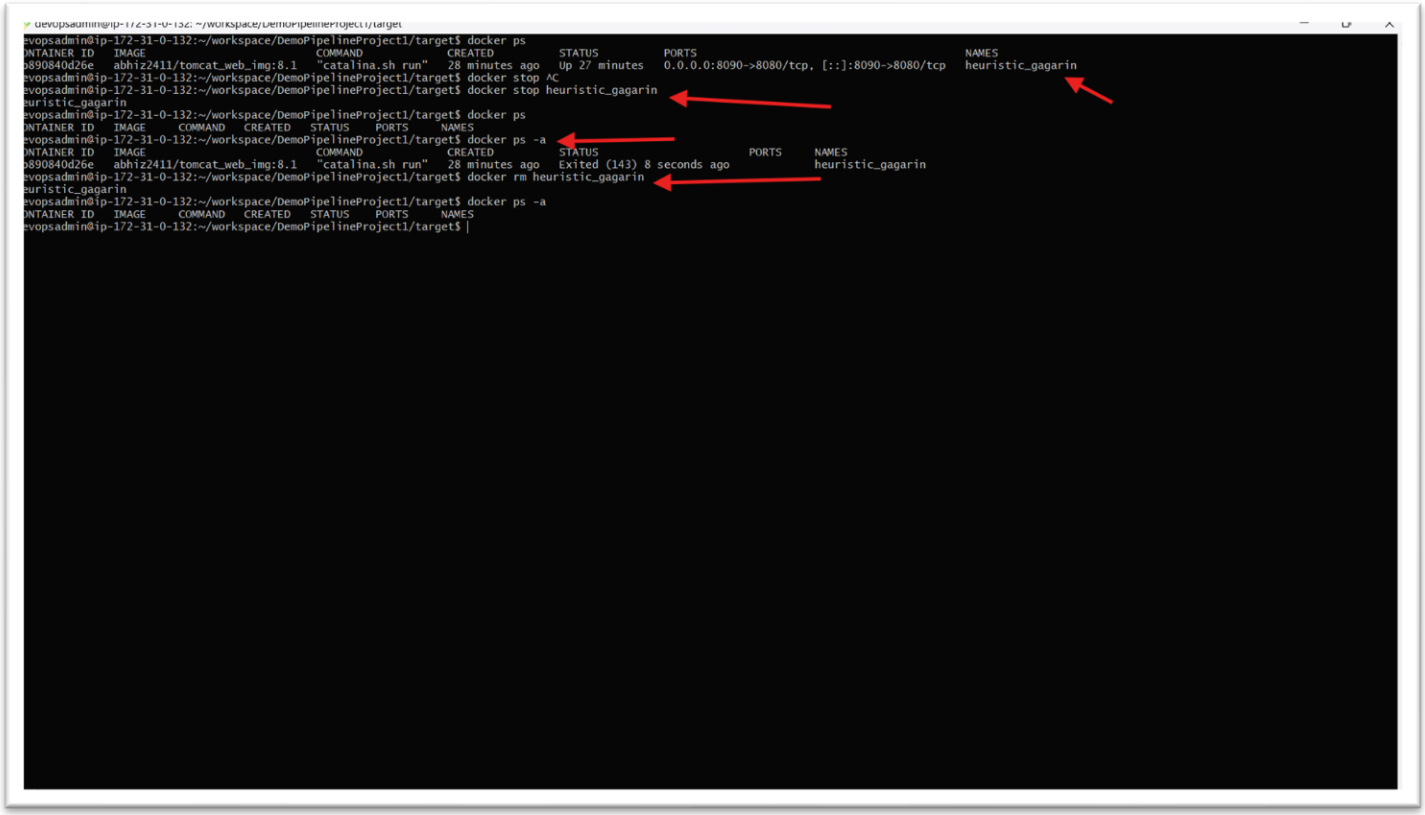


Fig. 3.03: Stop and remove container once done

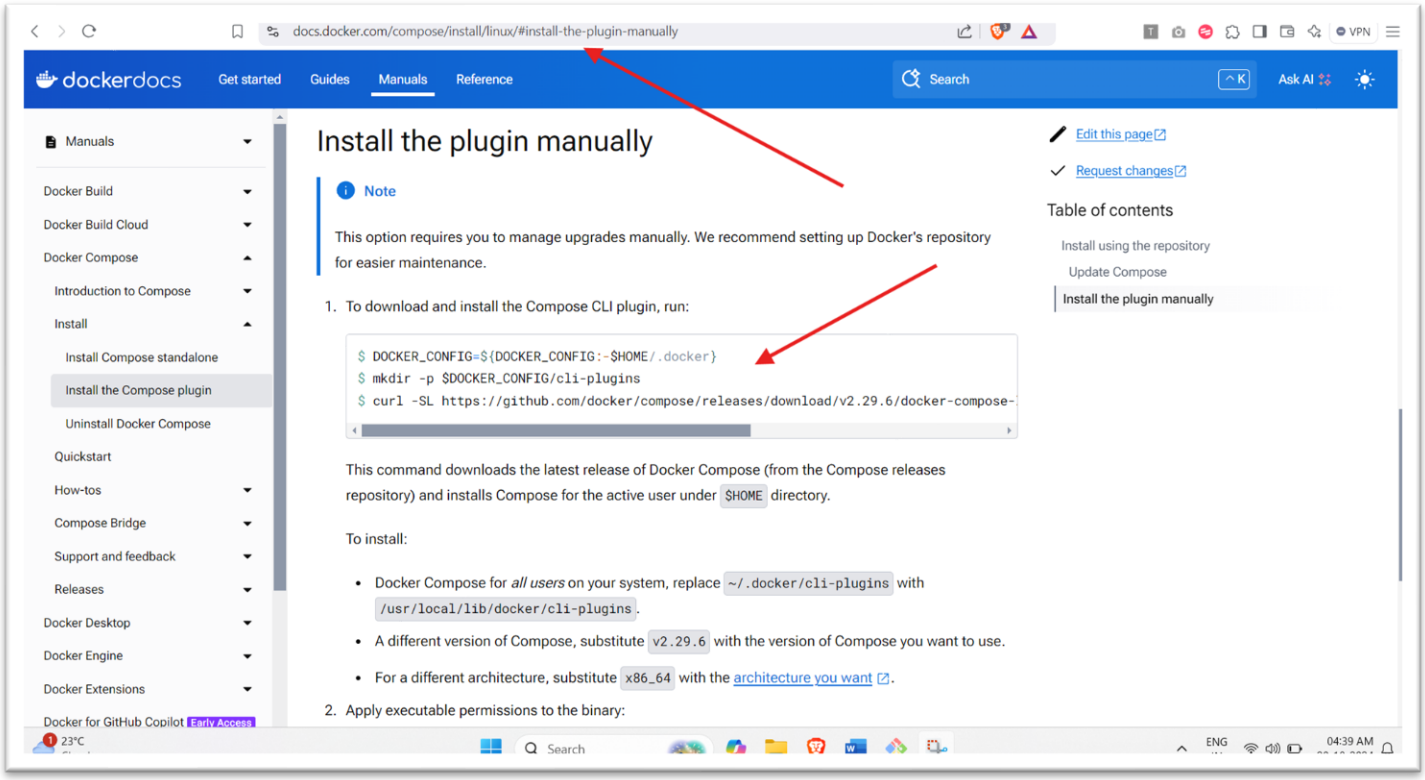
3. L3 - Demonstrate Docker Compose using the Application Image and MySQL Image to start and stop all container services

Ans.

Step 1: Install Docker compose:

Install Docker Compose on your machine by following the steps provided in Docker's official documentation. After installation, validate the setup by checking the version with:

` docker compose version`



Fig/ 3.01: Docker compose installation official manual

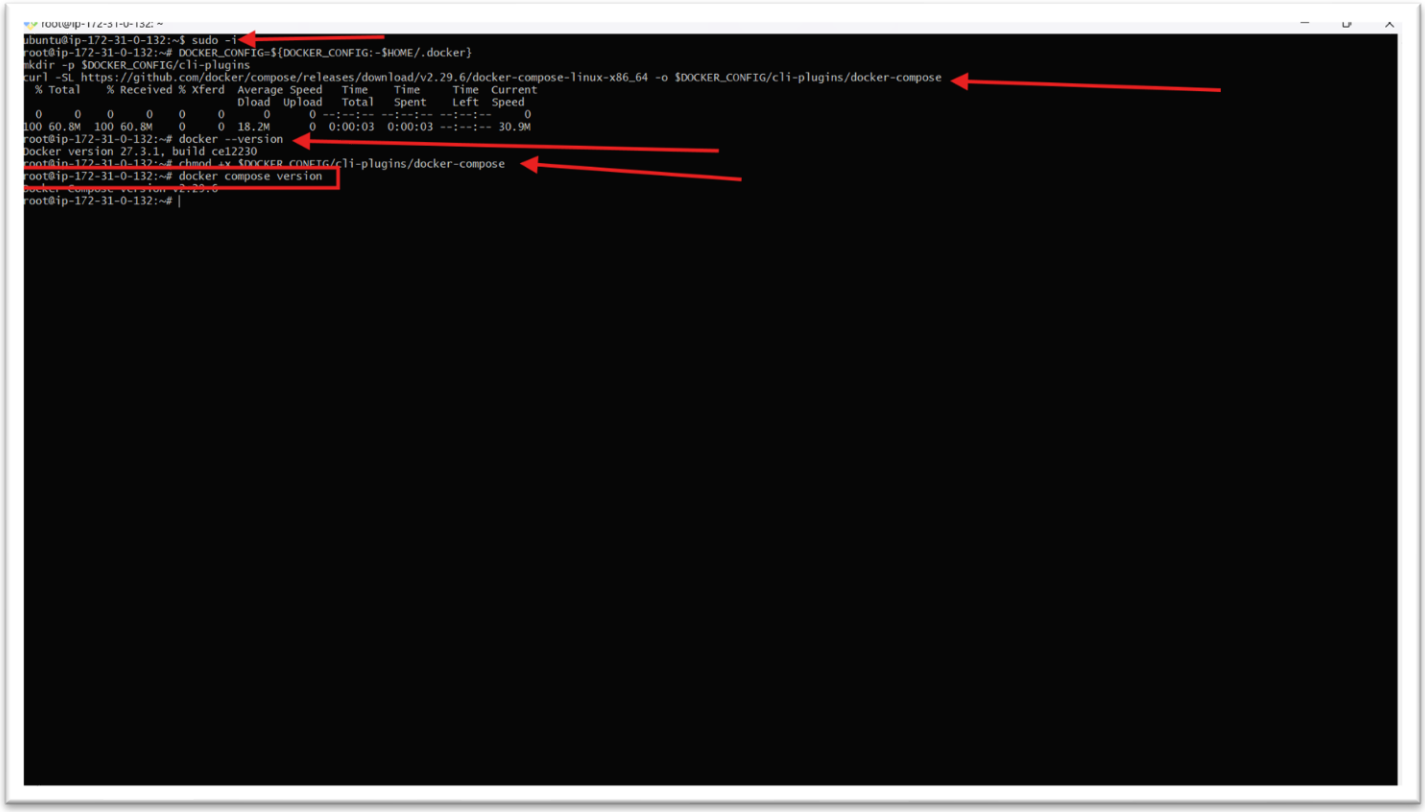


Fig. 3.02: Docker compose installation

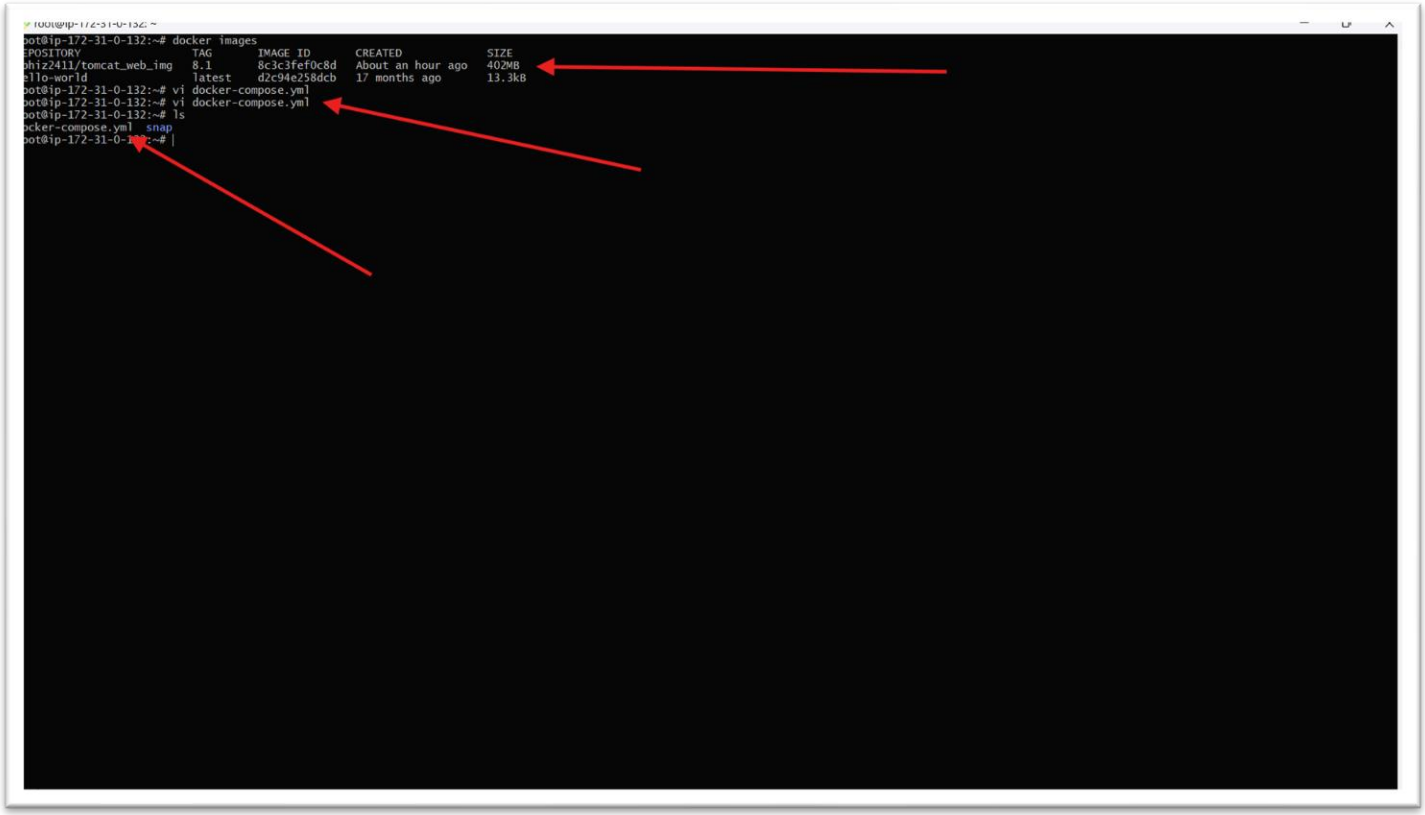
Step 2: Create a docker-compose.yml file:

To run multiple container services simultaneously with Docker Compose, create a .yaml file that defines all the services. Use the following command:

` vi docker-compose.yaml`

In the file, specify all the necessary services, then save and exit

Fig. 3.03: Contents of docker-compose.yml file



Fig, 3.04: docker-compose.yml file creation

Step 3: Compose up the container:

To start the container services defined in the docker-compose.yaml file, use the following command:

` docker compose up -d`

This will initiate the containers and run them in detached mode.

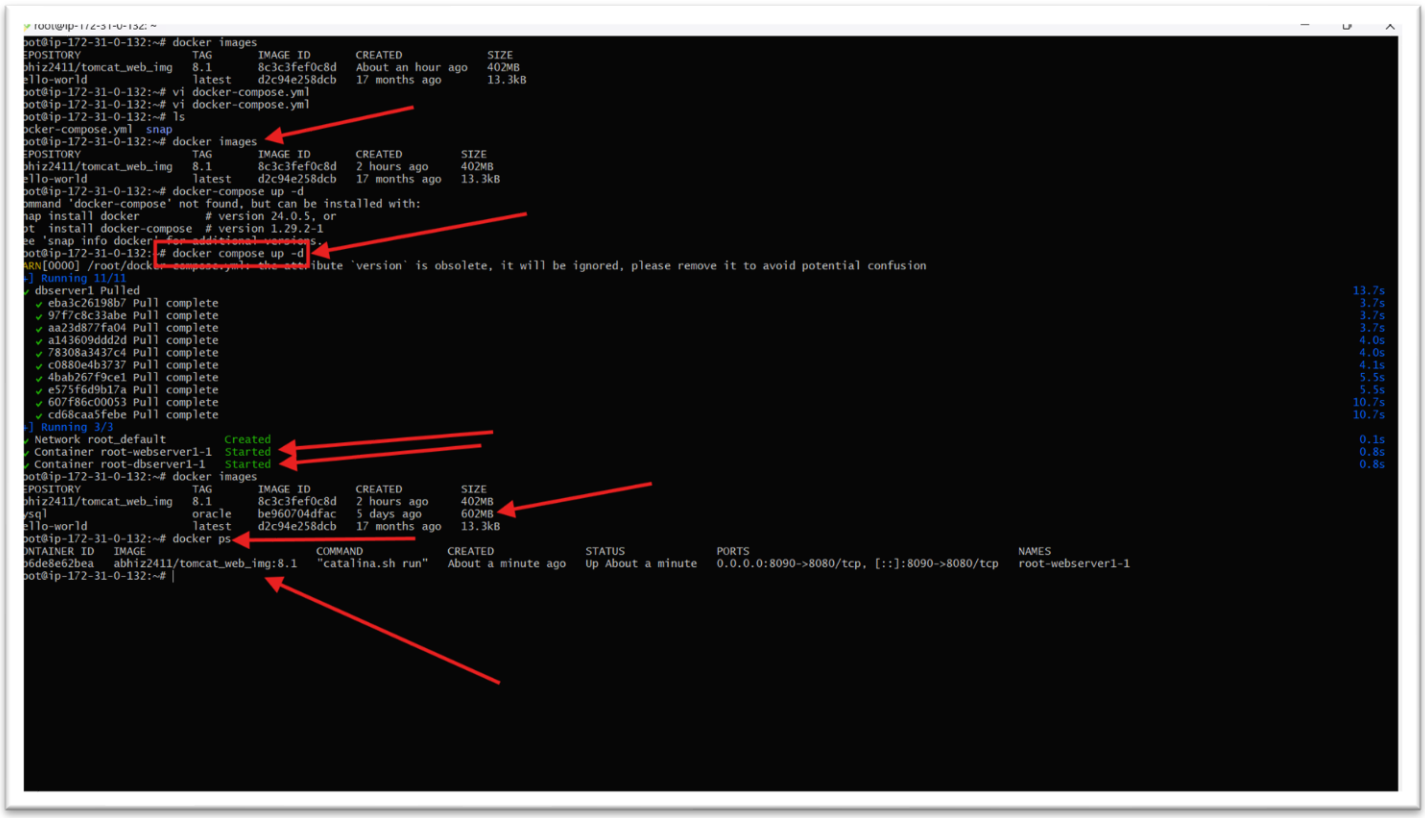


Fig. 3.05: docker compose up command

Step 4: docker compose down:

The containers are now running in detached mode. To stop and remove the services, execute the following command:

` docker compose down`

This command will halt the running containers and remove them from the containers list.

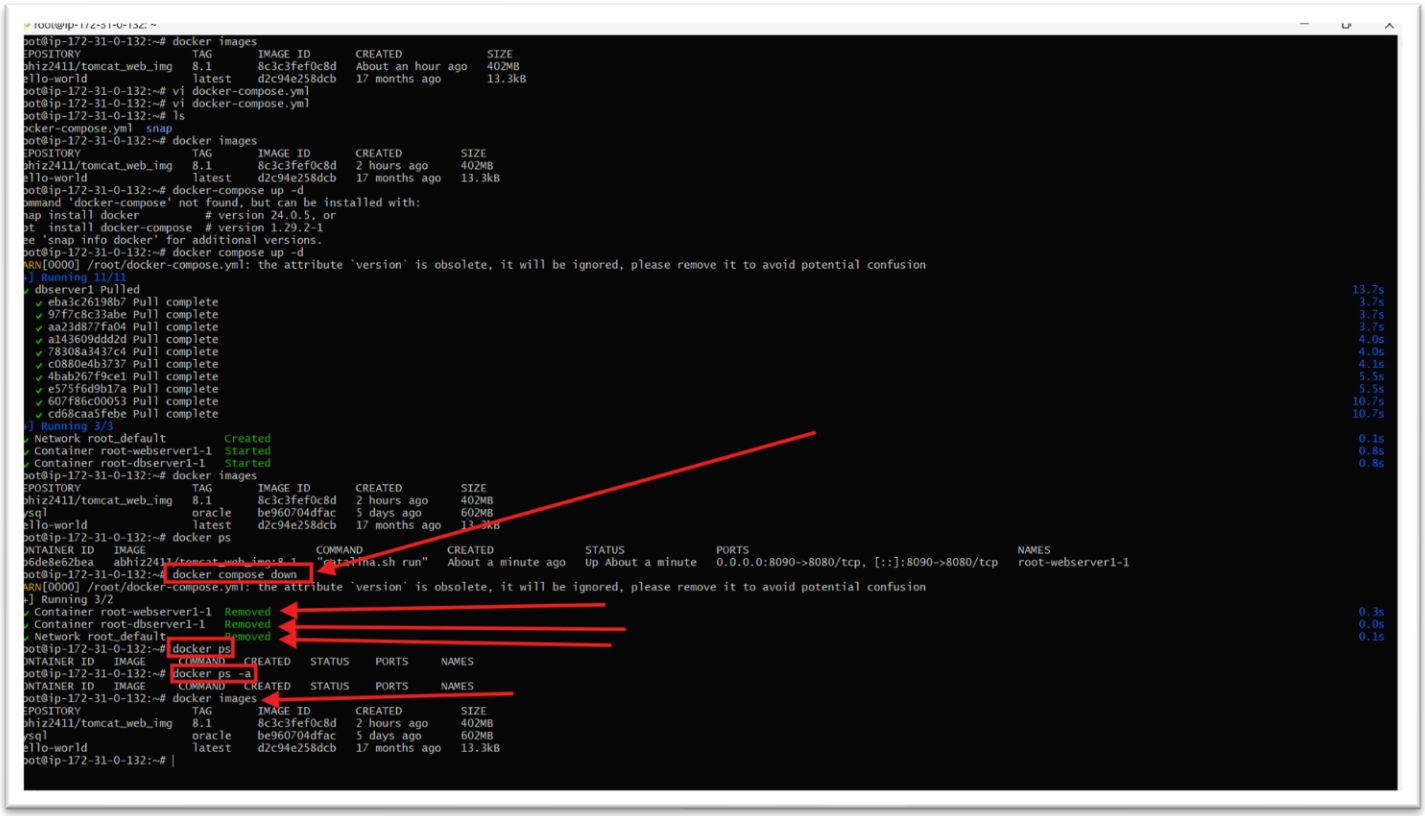


Fig. 3.06: docker compose down