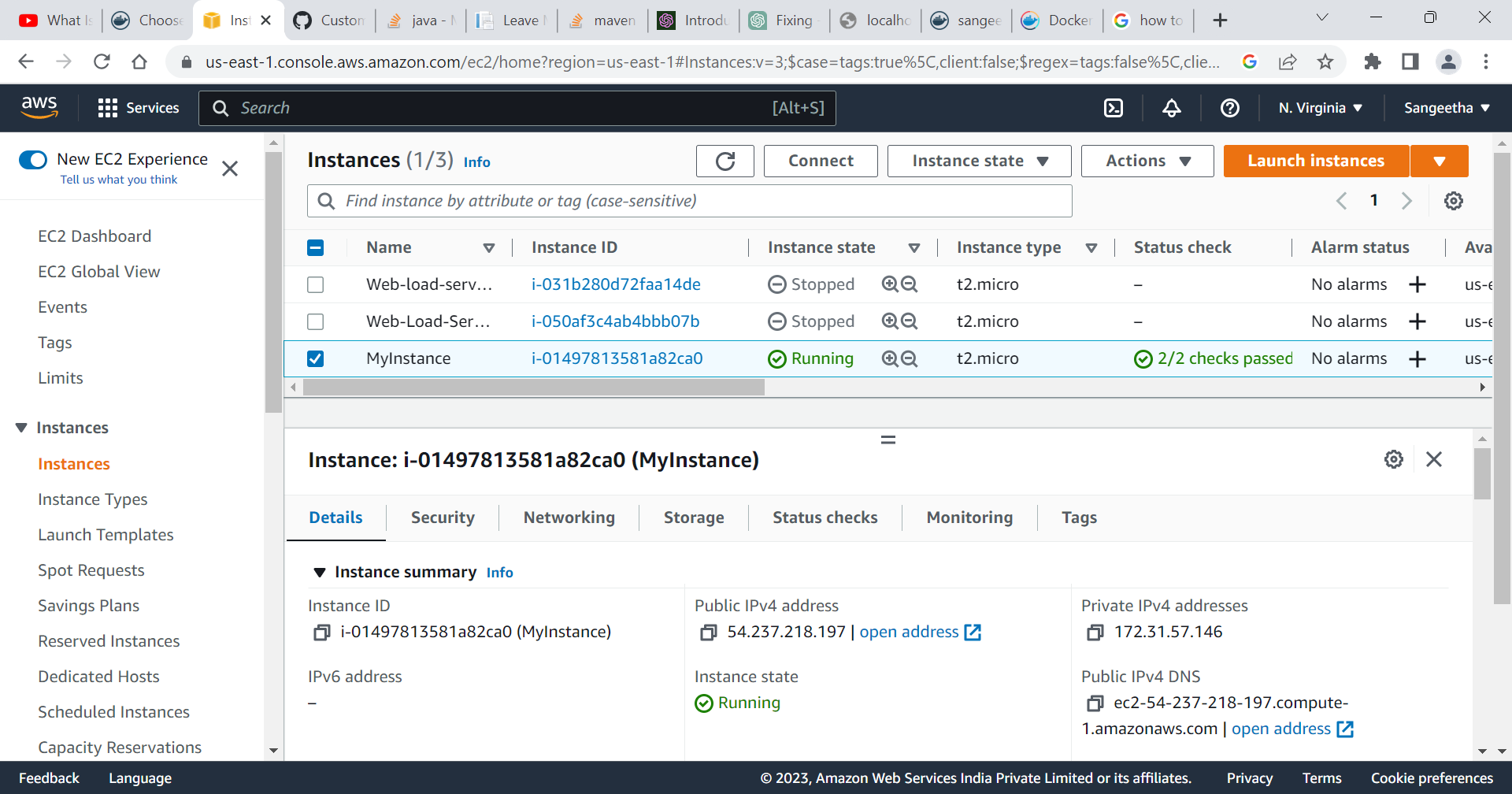
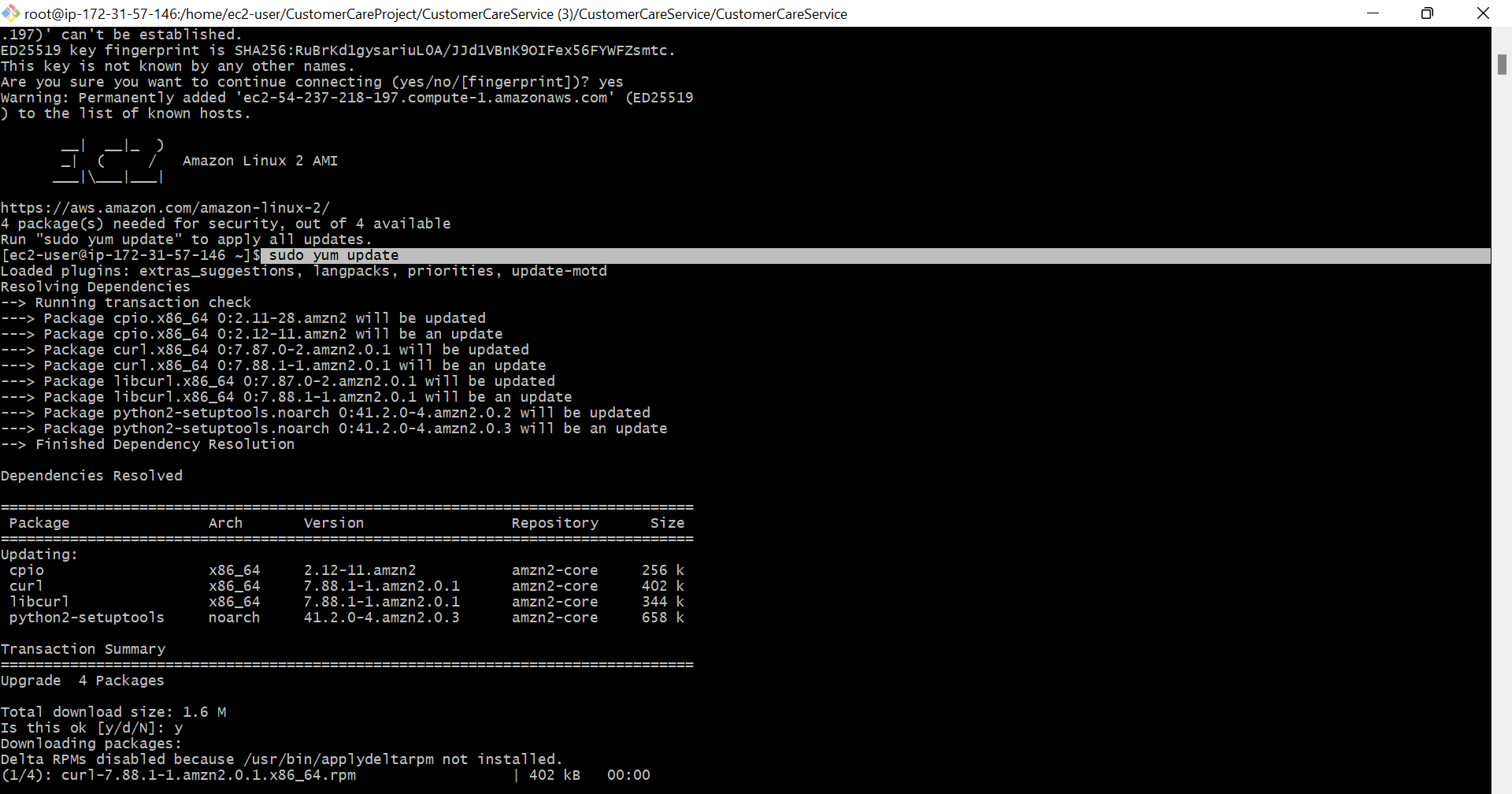
**Docker**

Step 1: create an instance in AWS account



Step 2: sudo yum update

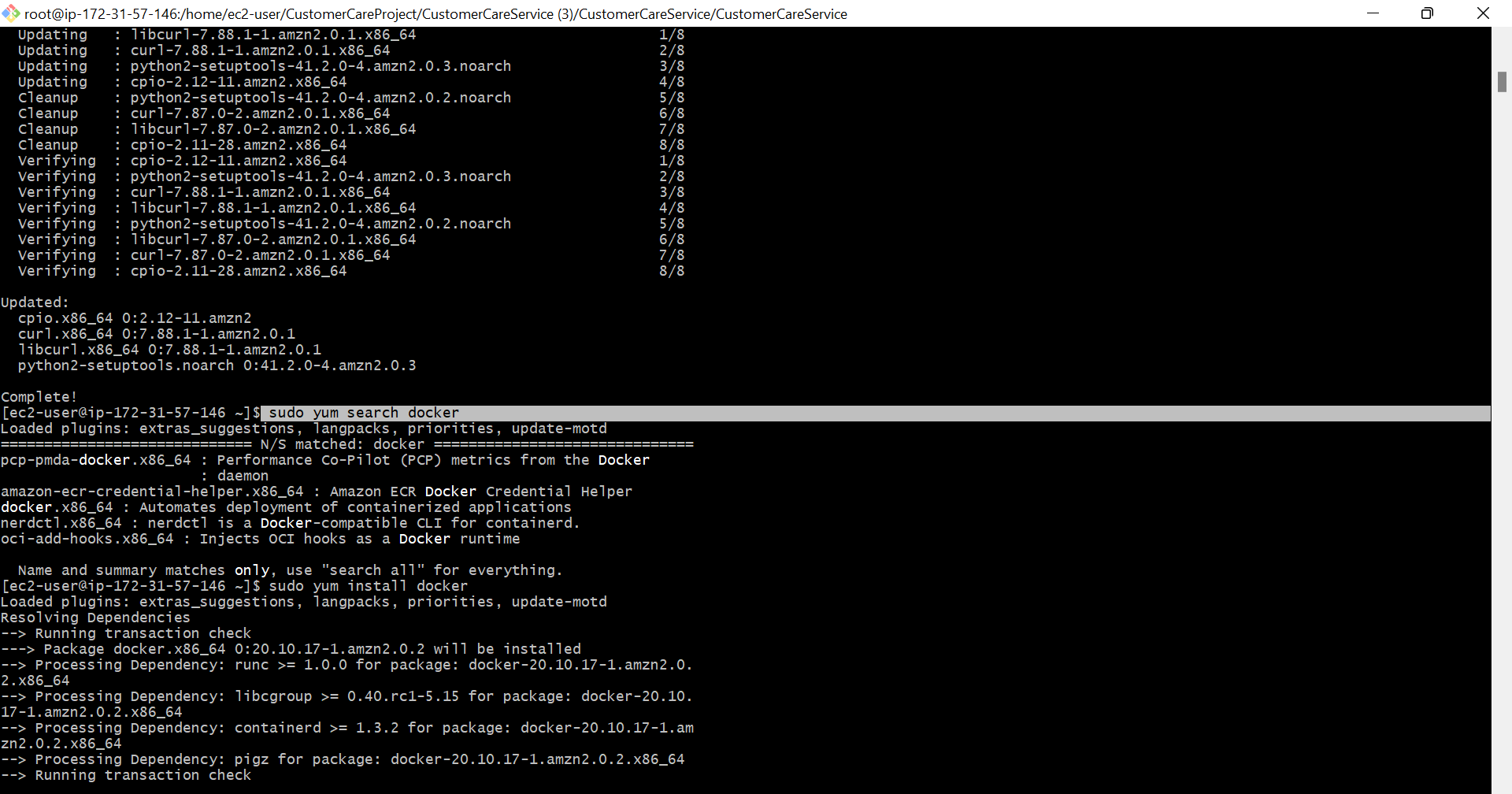
update the system's software packages and dependencies to their latest versions available in the configured software repositories.



Step 3: sudo yum search docker

to search for available packages related to Docker in the configured software repositories.

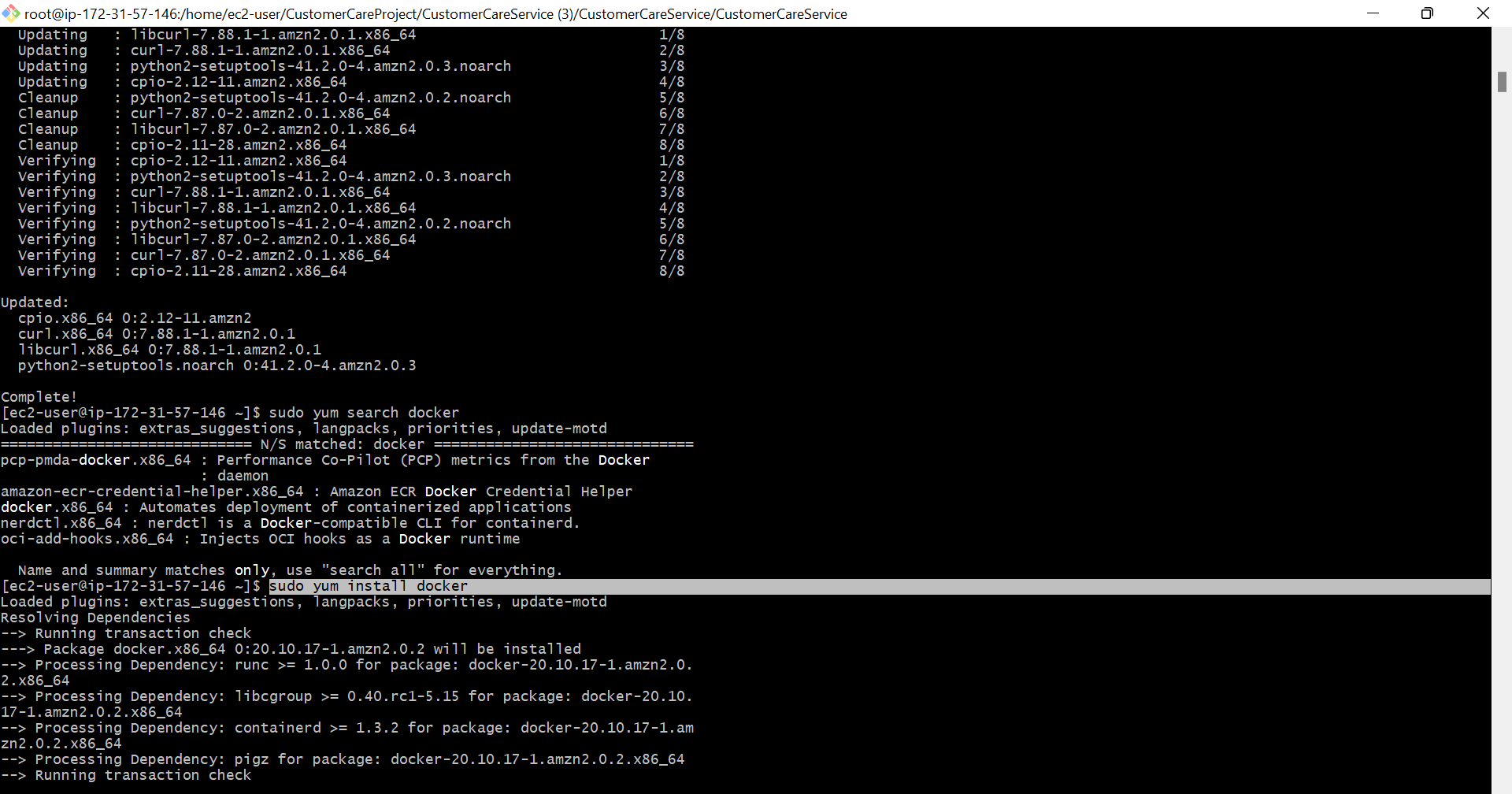
If executed, it will show a list of packages related to Docker that are available to install on your system. We can use this command to search for packages related to Docker that you may want to install or update on your system.



Step 4: sudo yum install docker

To install the Docker package on the system.

If executed, this command will install the latest available version of the Docker package and its dependencies from the configured software repositories. Once the installation is complete, you can start using Docker to create and manage containers.

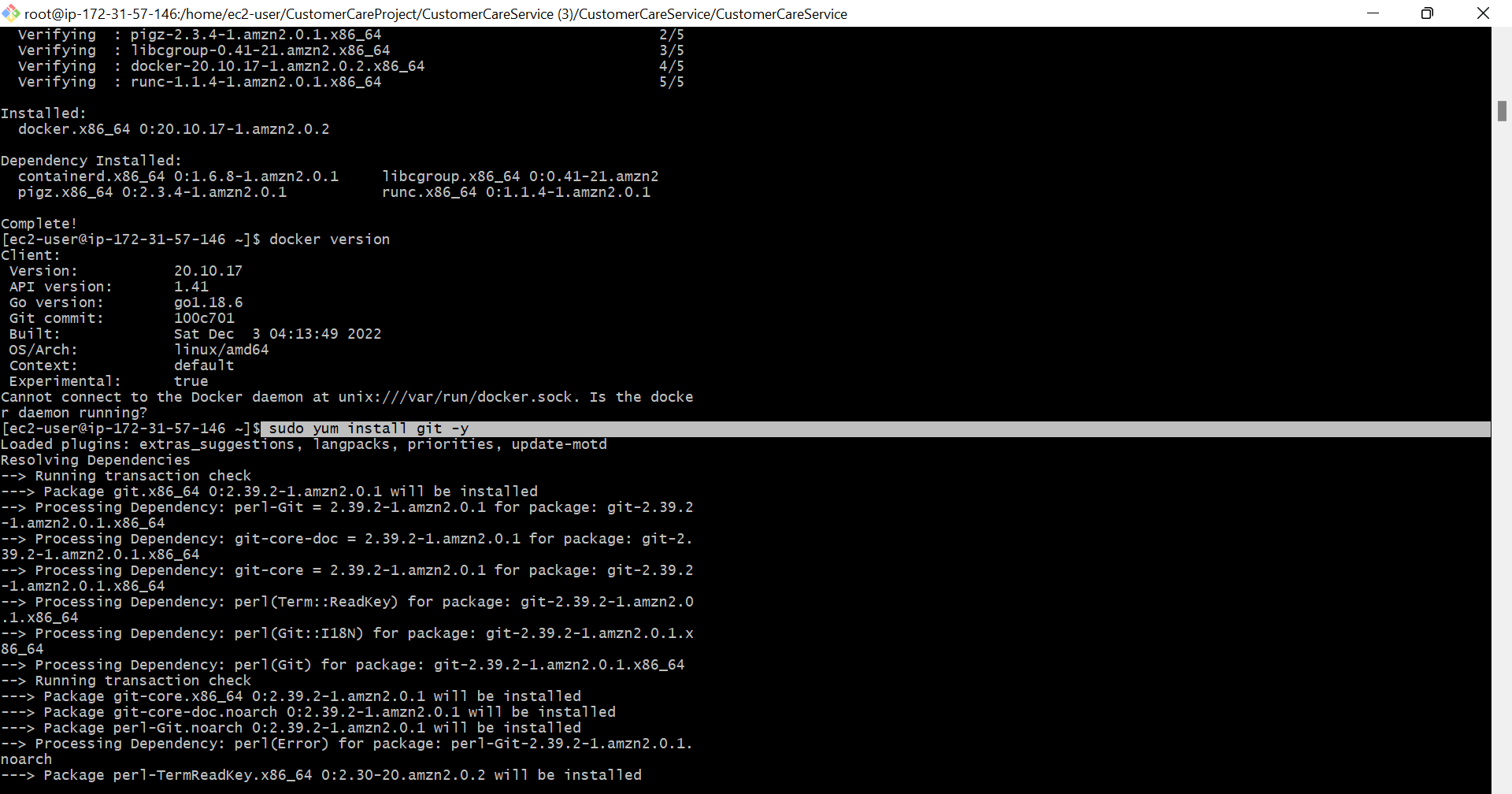


Step 5: sudo yum install git -y

To install the Git package on the system.

If executed, this command will install the latest available version of Git and its dependencies from the configured software repositories. The "-y" option is used to automatically answer "yes" to any prompts that may appear during the installation process.

Once the installation is complete, you can start using Git to manage version control of your code repositories.

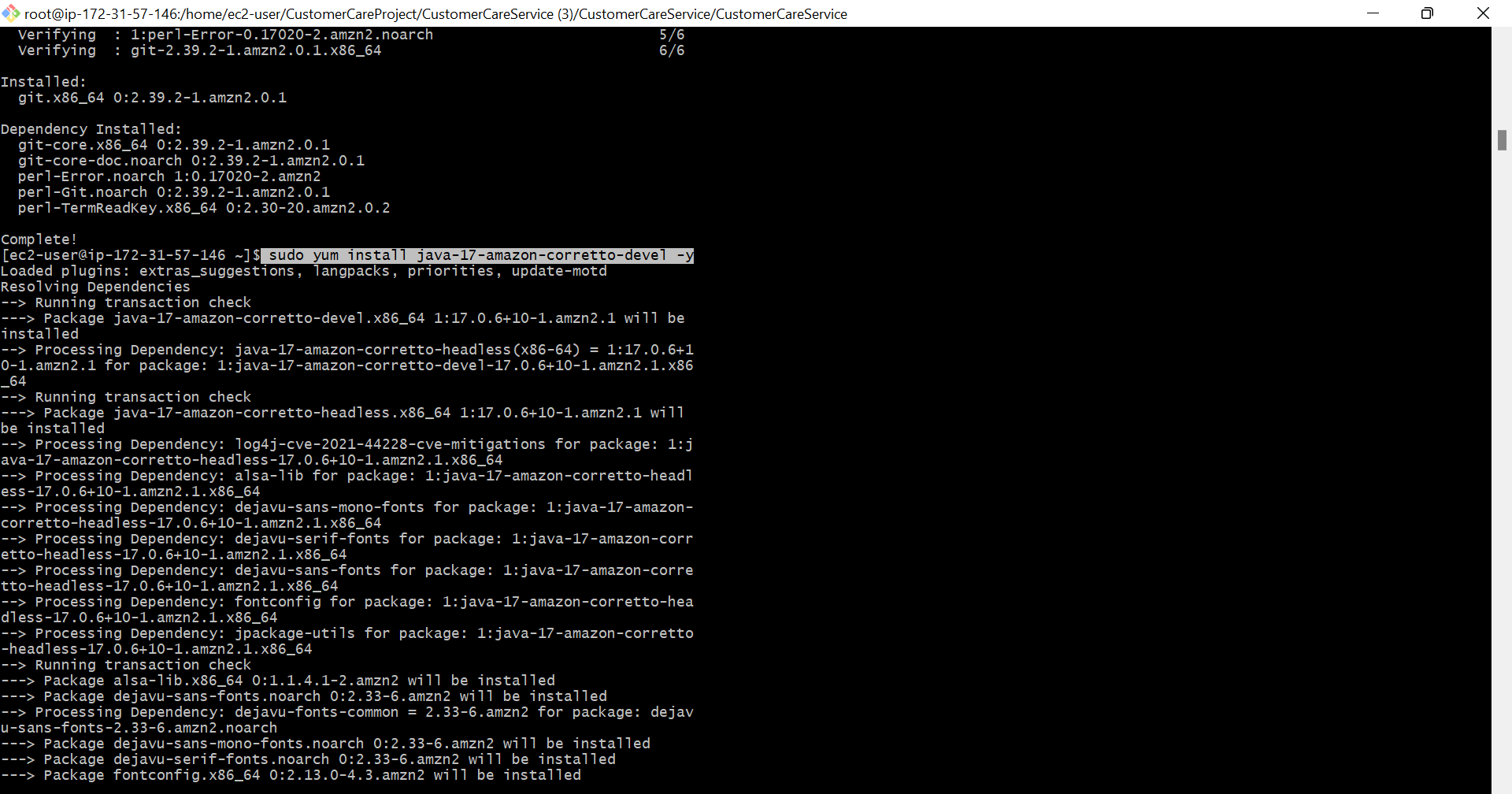


Step 6: sudo yum install java-17-amazon-corretto-devel y

Amazon Linux to install the Amazon Corretto 17 JDK (Java Development Kit) package on the system.

If executed, this command will install the latest available version of the Amazon Corretto 17 JDK package and its dependencies from the configured software repositories. The "-y" option is used to automatically answer "yes" to any prompts that may appear during the installation process.

Once the installation is complete, you can start using the Java development tools and runtime environment provided by the Amazon Corretto 17 JDK to develop and run Java applications.

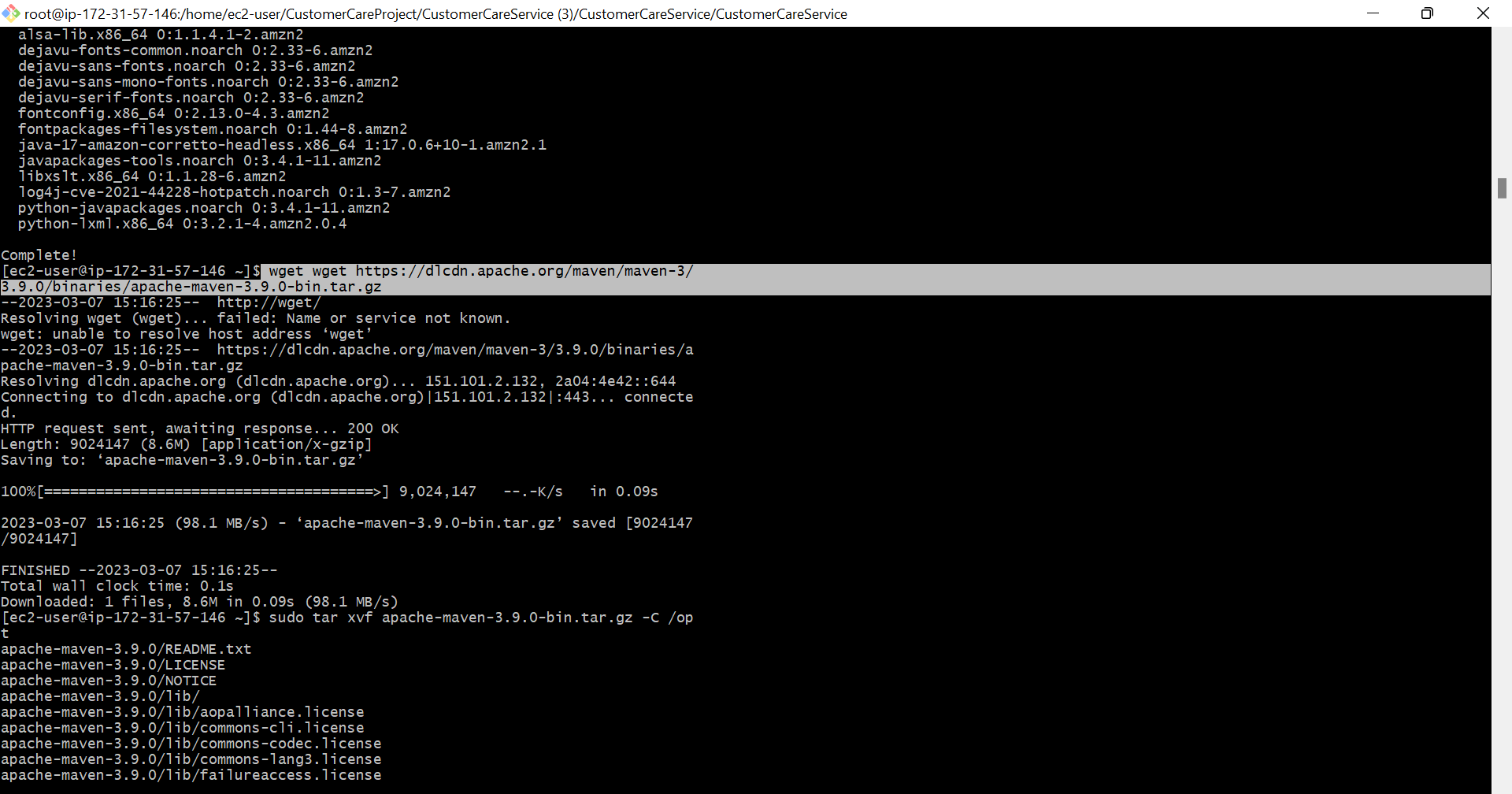


Step 7: wget <https://dlcdn.apache.org/maven/maven-3/3.9.0/binaries/apache-maven-3.9.0-bin.tar.gz>

used to download the Apache Maven 3.9.0 binary distribution package from the Apache Maven website.

If executed, this command will download the Maven binary distribution package in a compressed tarball format (.tar.gz) to the current working directory of your terminal session.

Once the download is complete, you can extract the package contents to a suitable location on your system and set the appropriate environment variables to start using Maven for your Java projects.



Step 8: sudo tar xvf apache-maven-3.9.0-bin.tar.gz -C /opt

used to extract the contents of the Apache Maven binary distribution package that was downloaded in the previous step to the "/opt" directory of the system.

If executed, this command will extract the contents of the "apache-maven-3.9.0-bin.tar.gz" file to the "/opt/apache-maven-3.9.0" directory on the system. The "-C" option is used to specify the target directory where the extracted files should be placed.

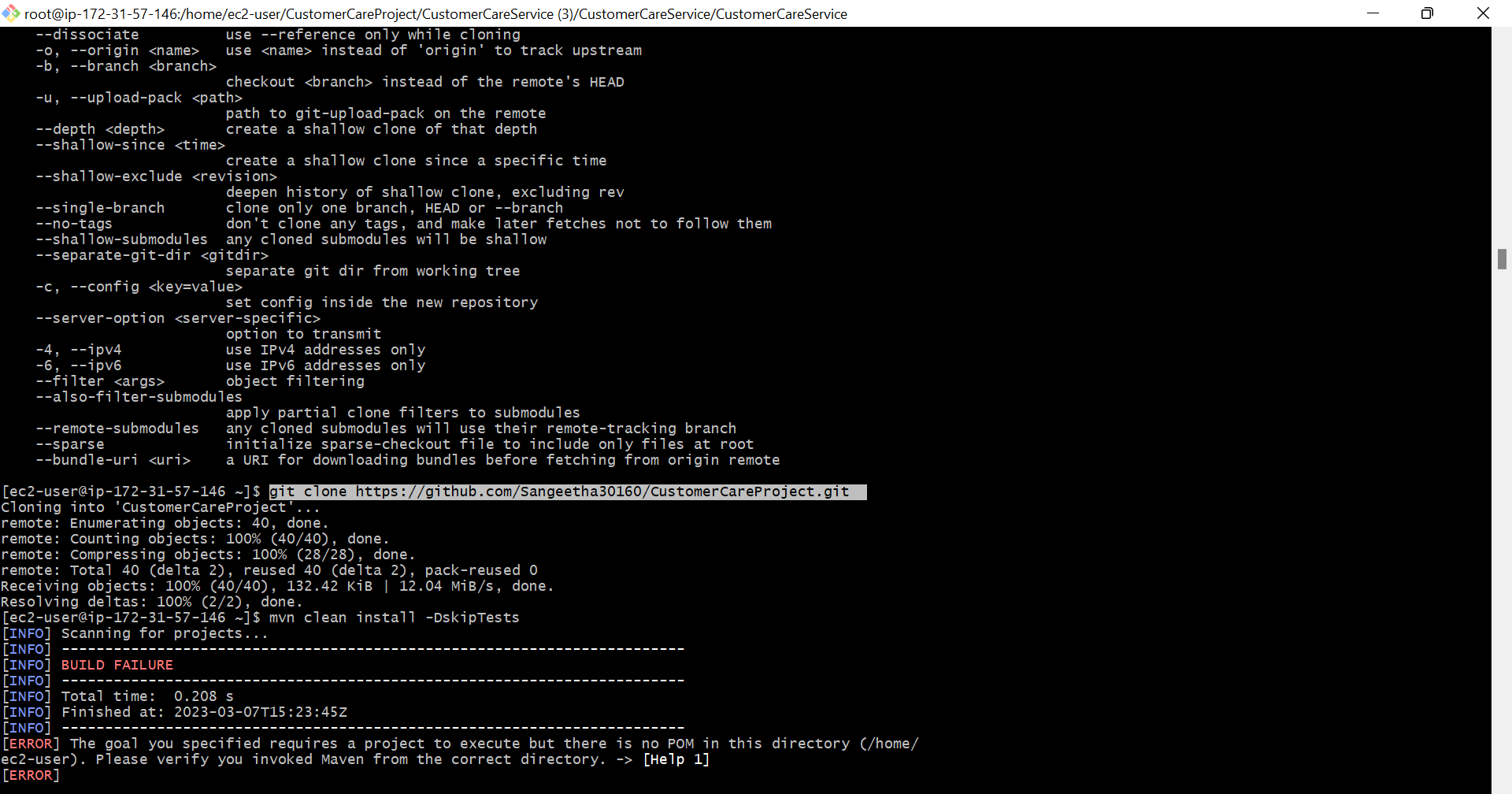
Once the extraction is complete, you can set the appropriate environment variables to configure the Maven installation on your system and start using it for your Java projects.

Step 9: git clone <https://github.com/sangeetha30160/customerProject.git>

used to create a copy of a Git repository on your local machine.

If executed, this command will create a local copy of the specified remote Git repository, including all the files and directories, commit history, and branches.

By default, the cloned repository will be placed in a directory with the same name as the repository. Once the cloning process is complete, you can make changes to the local copy of the repository and push them back to the remote repository as needed.



Step 10: sudo ln -s /opt/apache-maven-3.9.0 /opt/maven

used to create a symbolic link to the Maven installation directory on the system.

If executed, this command will create a symbolic link named "maven" in the "/opt" directory that points to the "apache-maven-3.9.0" directory. This can be useful if you need to switch between different versions of Maven on the same system or if you want to use a shorter name to refer to the Maven installation directory.

Once the symbolic link is created, you can set the appropriate environment variables to configure the Maven installation on your system and start using it for your Java projects.

Step 11: mvn clean install -dskiptests

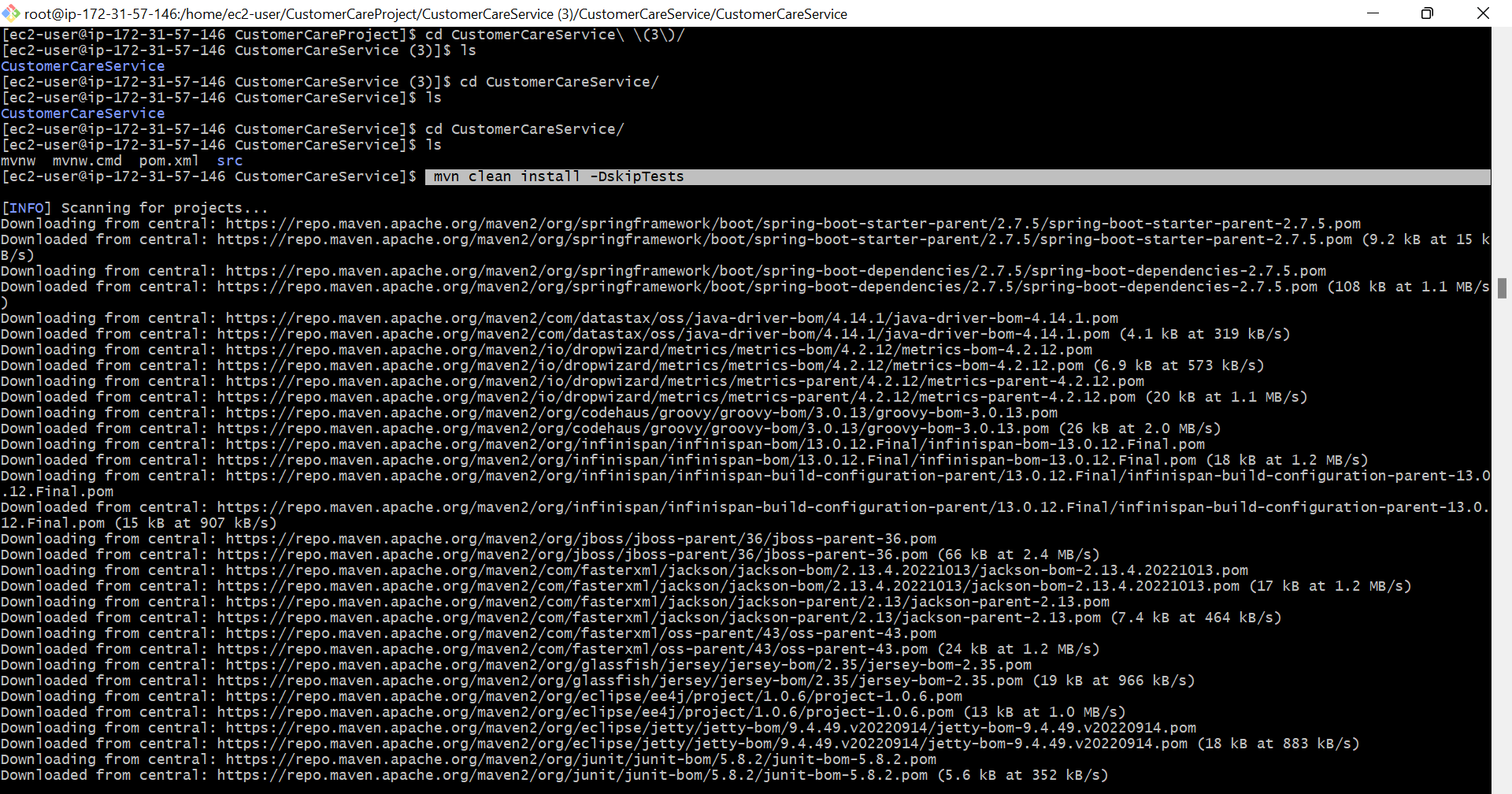
used to build and package a Maven project on your system without running the test cases.

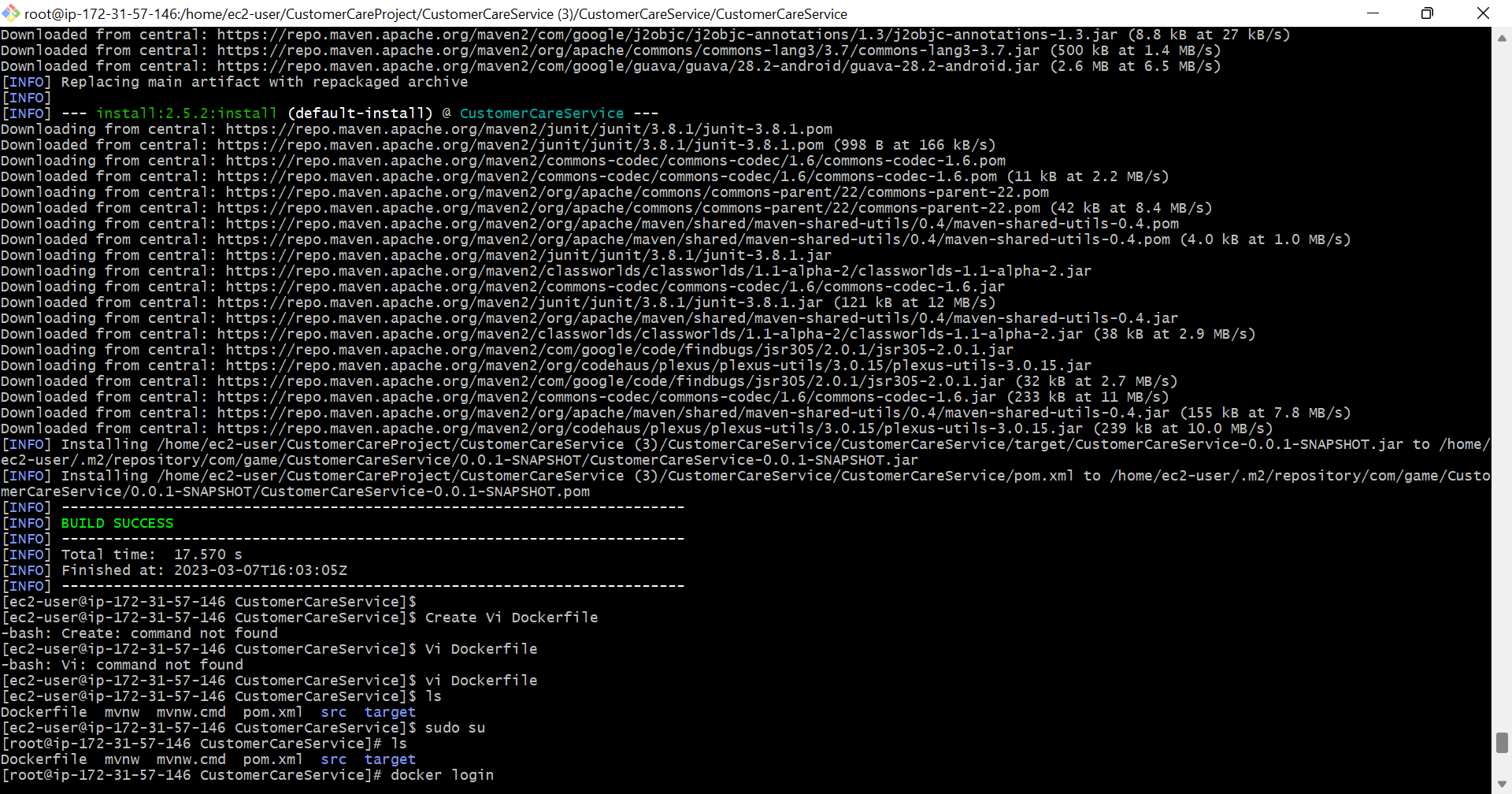
If executed in the root directory of a Maven project, this command will perform the following actions:

"clean": Deletes the "target" directory of the project, which contains the compiled code, resources, and other build artifacts from previous builds. This ensures a clean build environment for the current build.

"install": Compiles the source code of the project and packages it into a distributable format, such as a JAR file. It also installs the packaged artifact into the local Maven repository, which can be used by other Maven projects as a dependency.

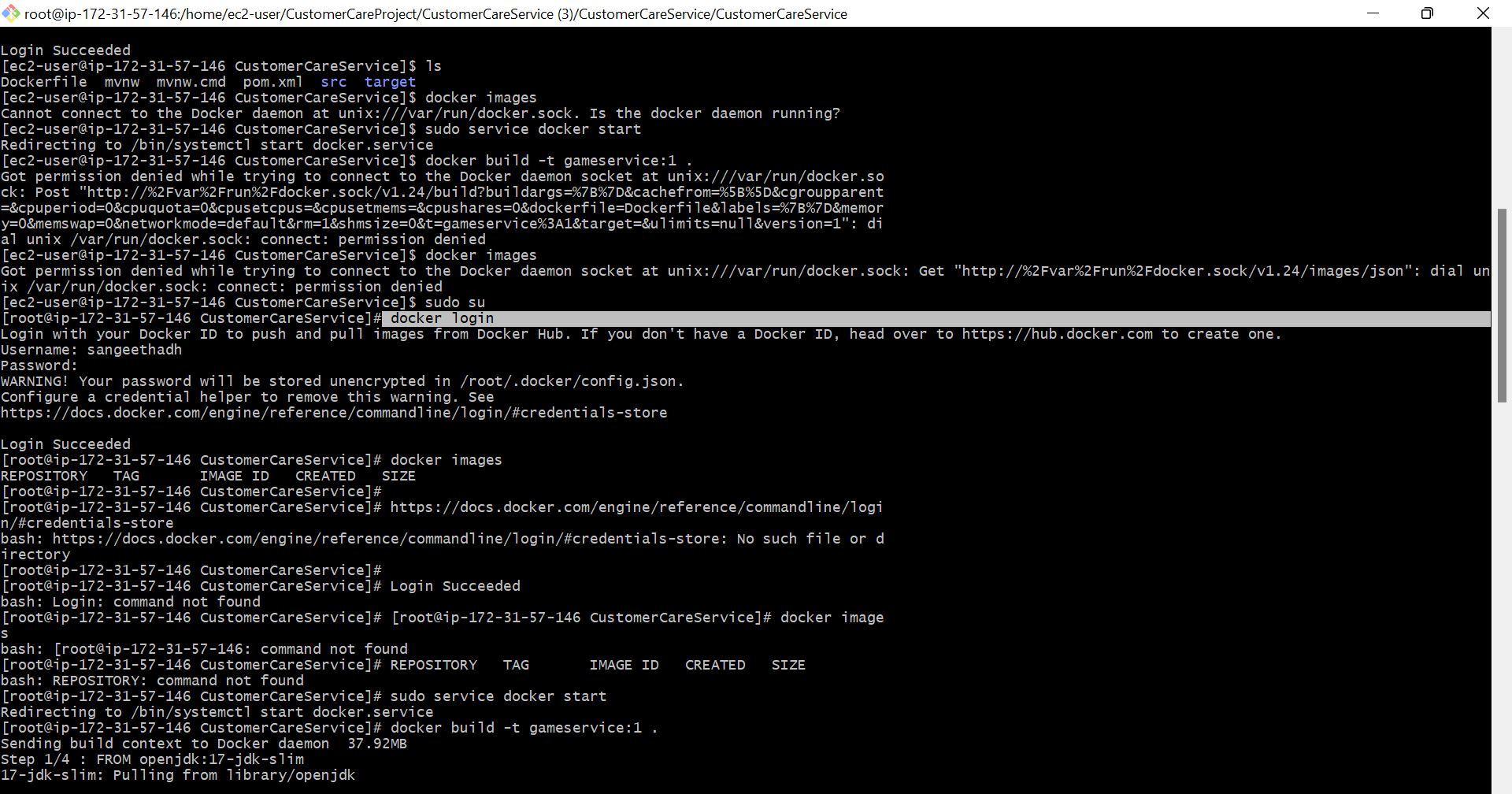
"-DskipTests": Skips the execution of any test cases included in the project during the build process. This can speed up the build process, especially if the test suite is large or time-consuming.





Step 12: docker login

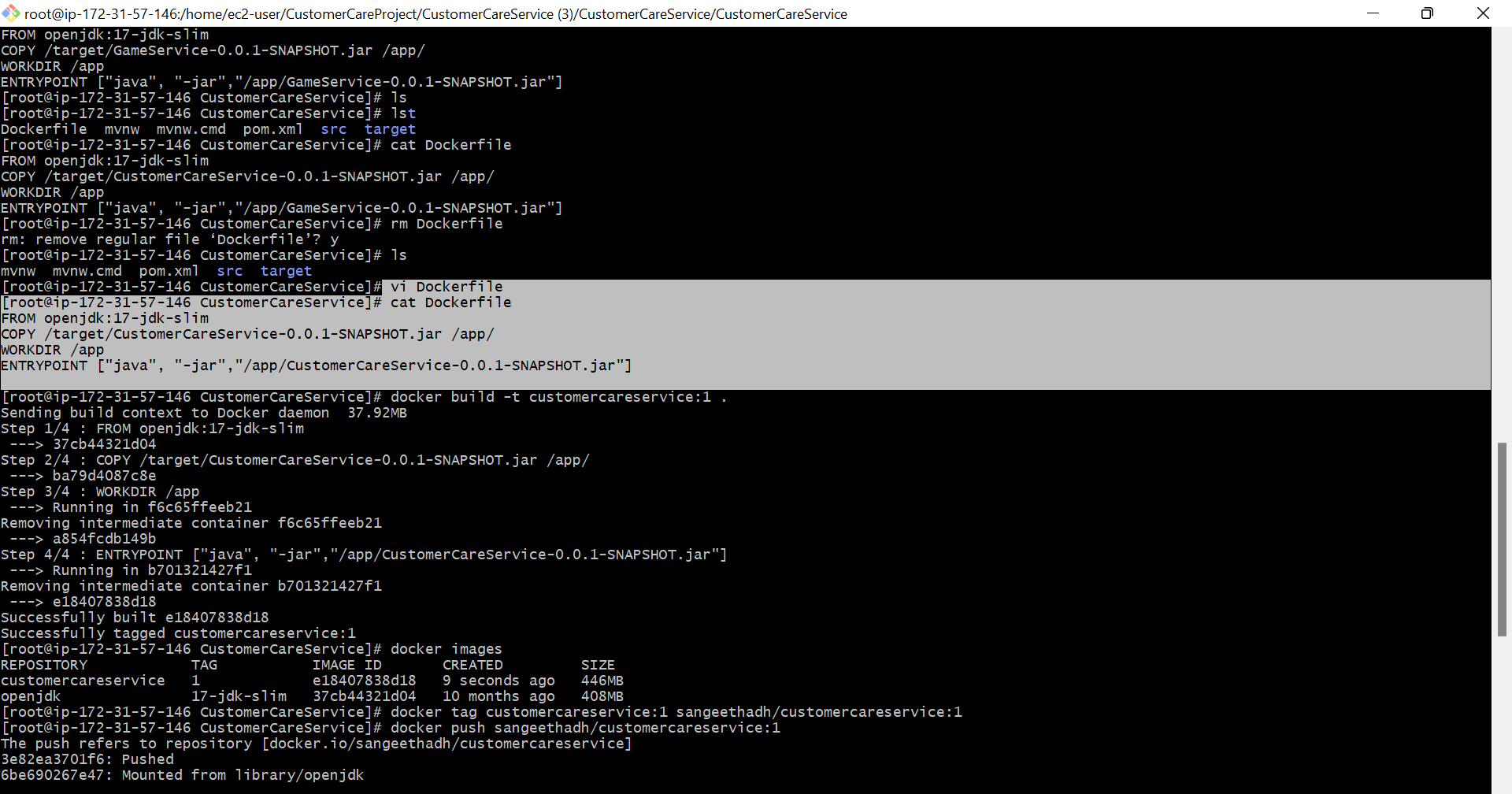
used to authenticate with a Docker registry. It is typically used before pushing or pulling images to/from a Docker registry.



Step 13: vi dockerfile

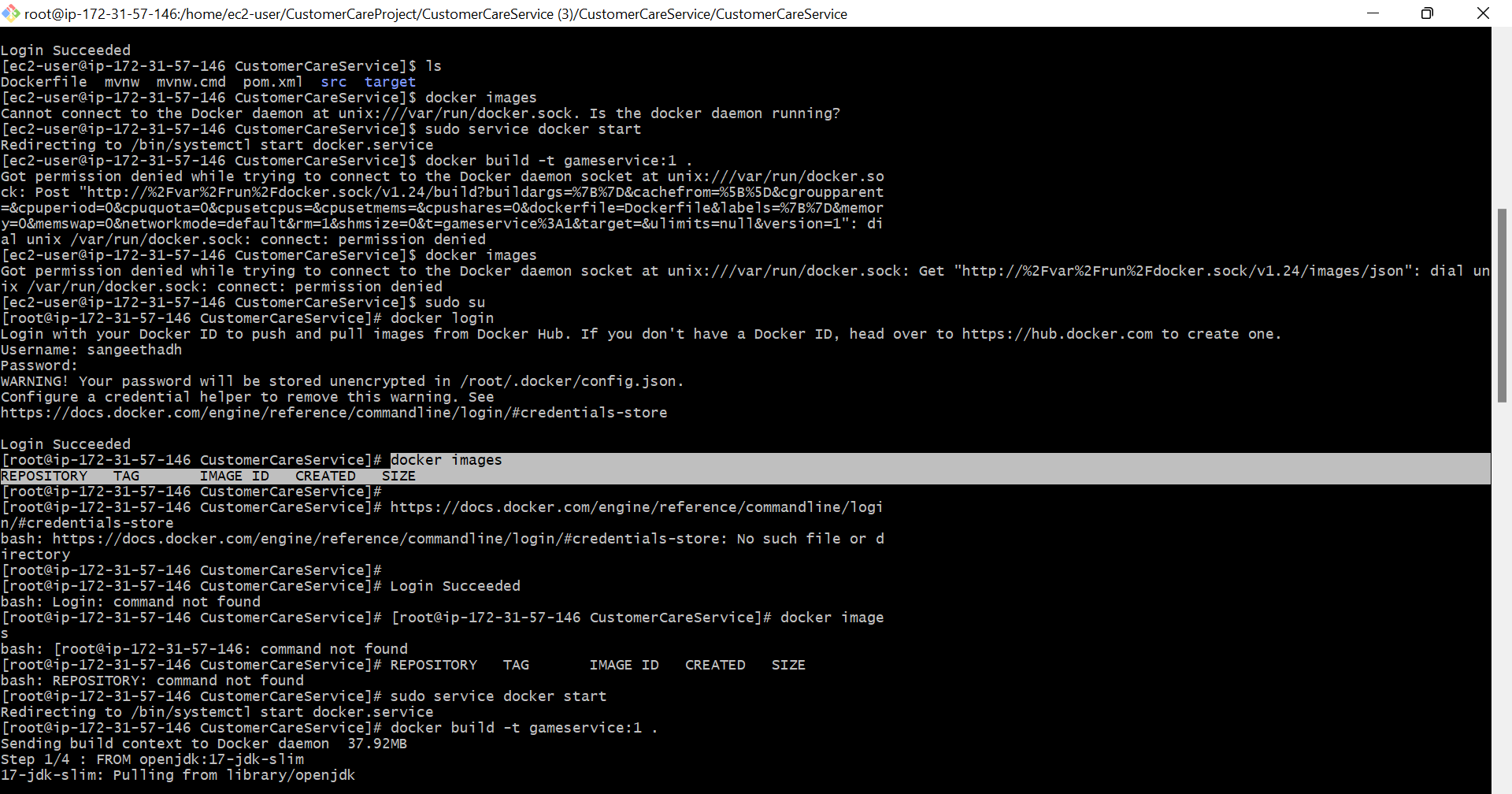
used to edit files.

To create or edit a Dockerfile using **vi**



Step 14: docker images

used to list the Docker images that are currently stored on your system.



Step 15: docker build -t customerservice:1 .

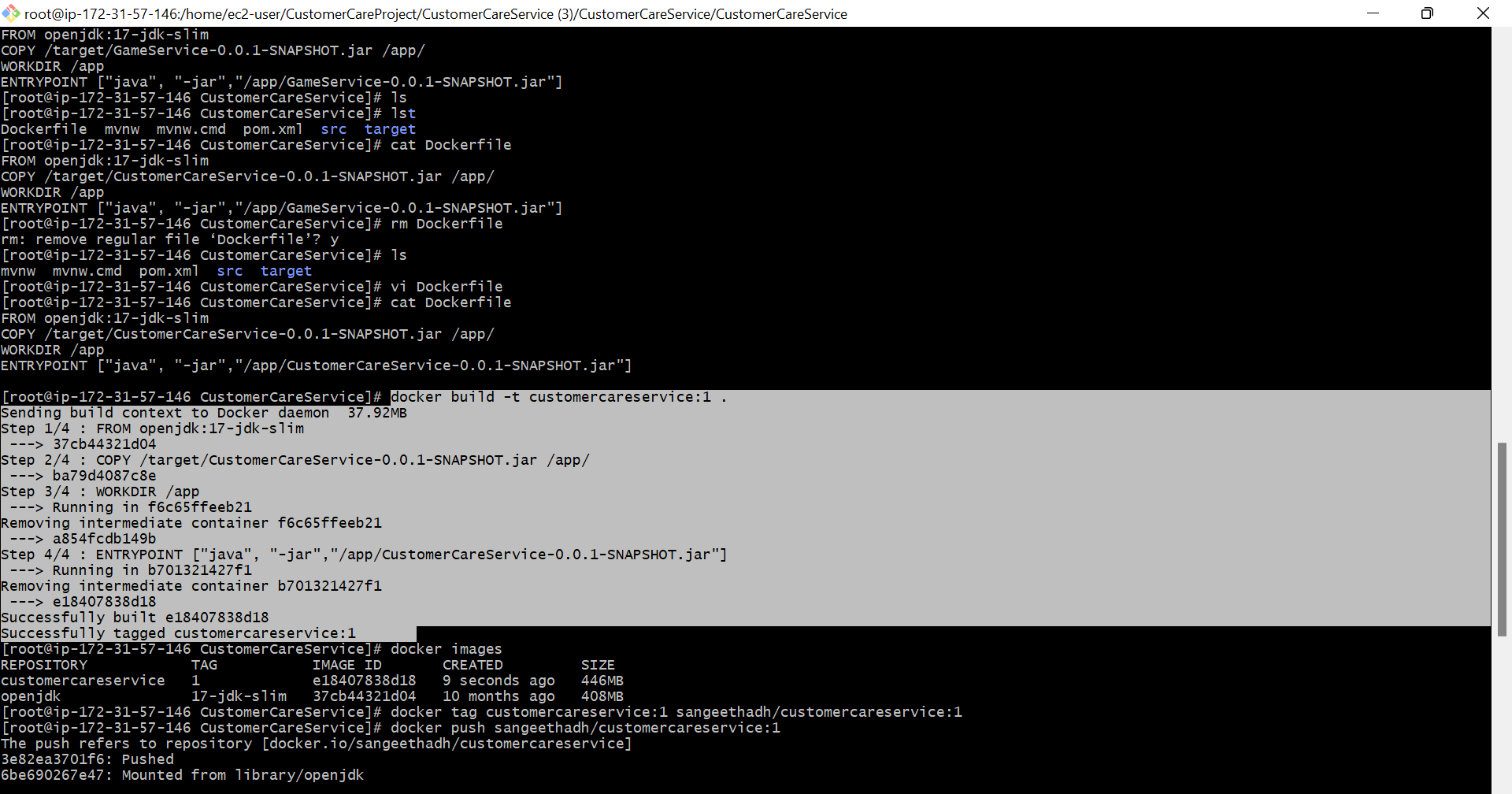
command used in the Docker platform to build images from a Dockerfile.

Docker file:

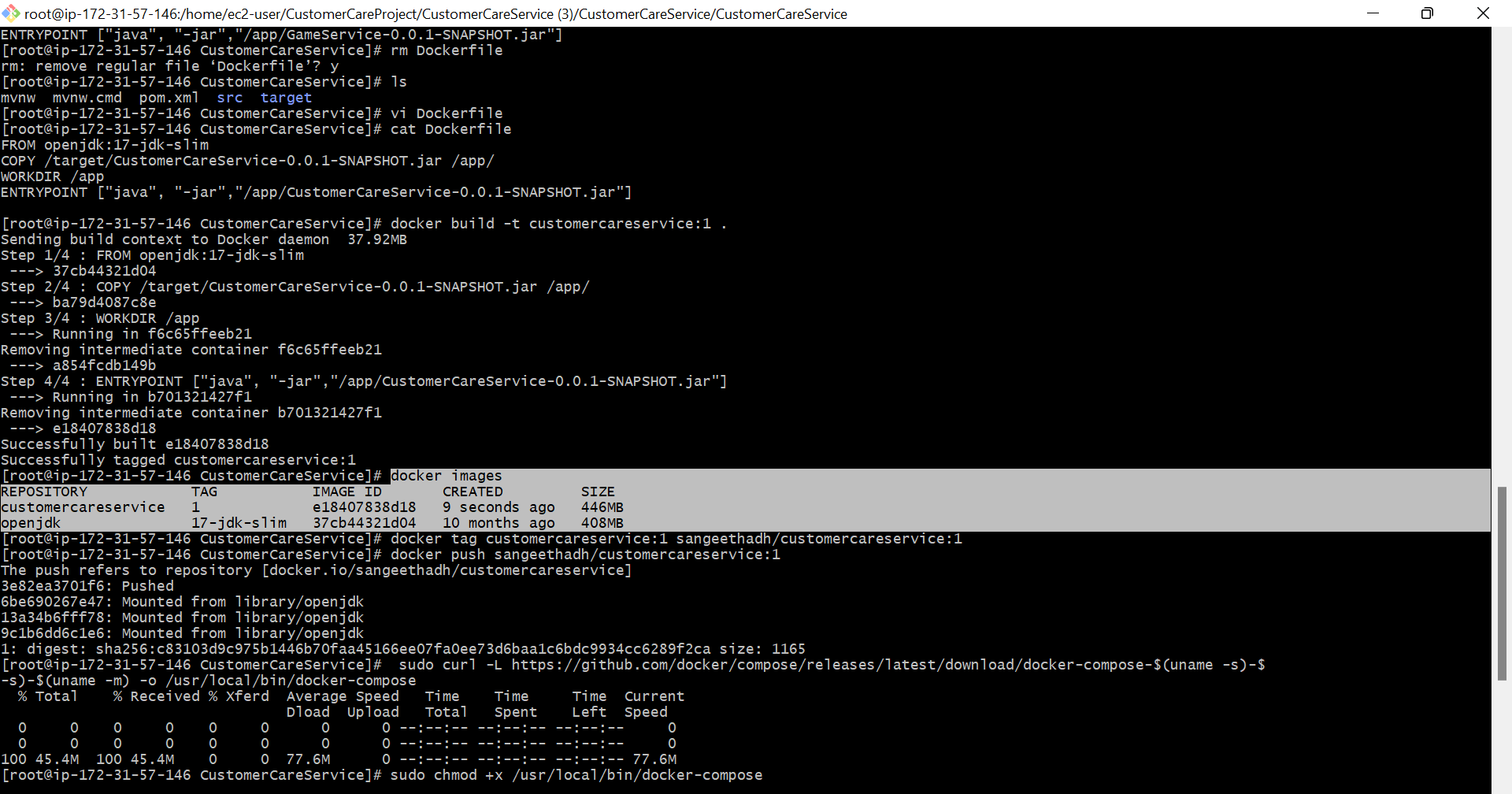
A Dockerfile is a text file that contains all the commands needed to assemble a Docker image.

The "docker build" command reads the Dockerfile and builds an image by executing each command in the file.

Docker to build an image with the tag "myimage" using the Dockerfile located in the current directory "."



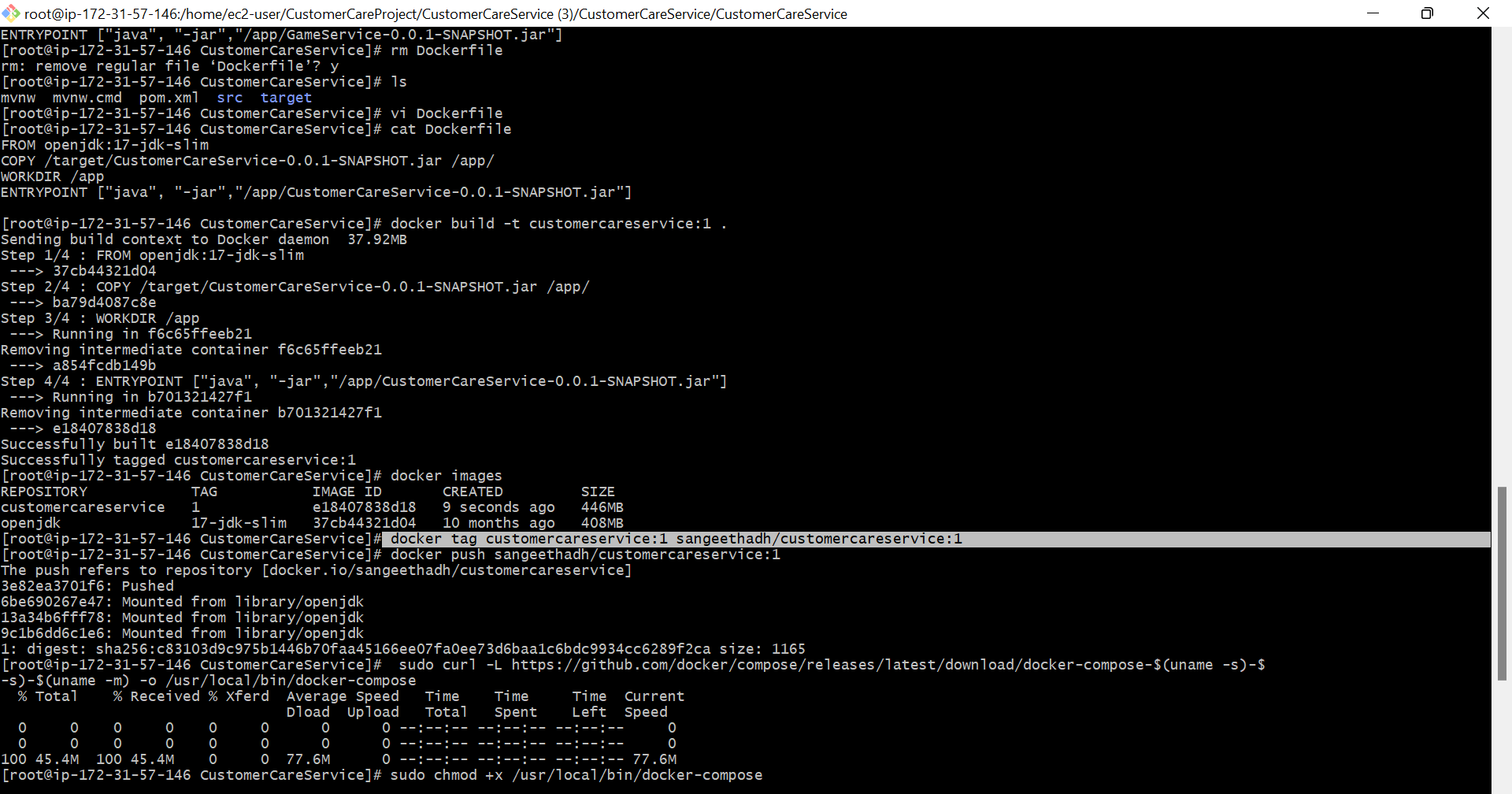
Step 16: images as been created



Step 17: Docker tag customerservice:1 sangeethdh/customerservice:1

"Docker tag" is a command used to assign a new name and optionally a new tag to an existing image in your Docker registry.

This command creates a new image with the name "username/image" and the tag "1", and it points to the existing image "image" with the tag "1". You can then push this new image to your Docker registry using the "docker push" command.

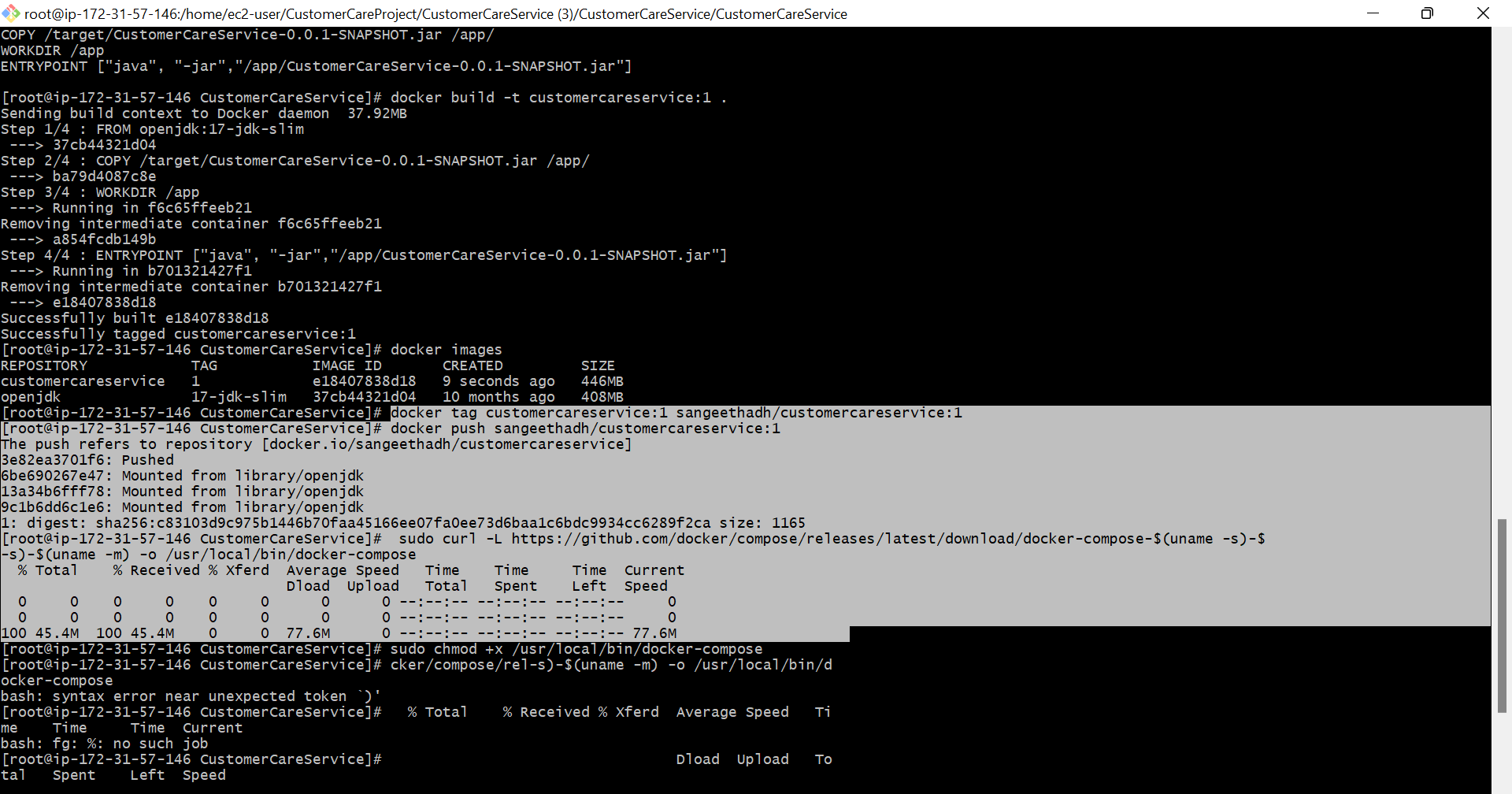


Step 18: docker push sangeethdh/customerservice:1

Used to push a Docker image from your local machine to a Docker registry.

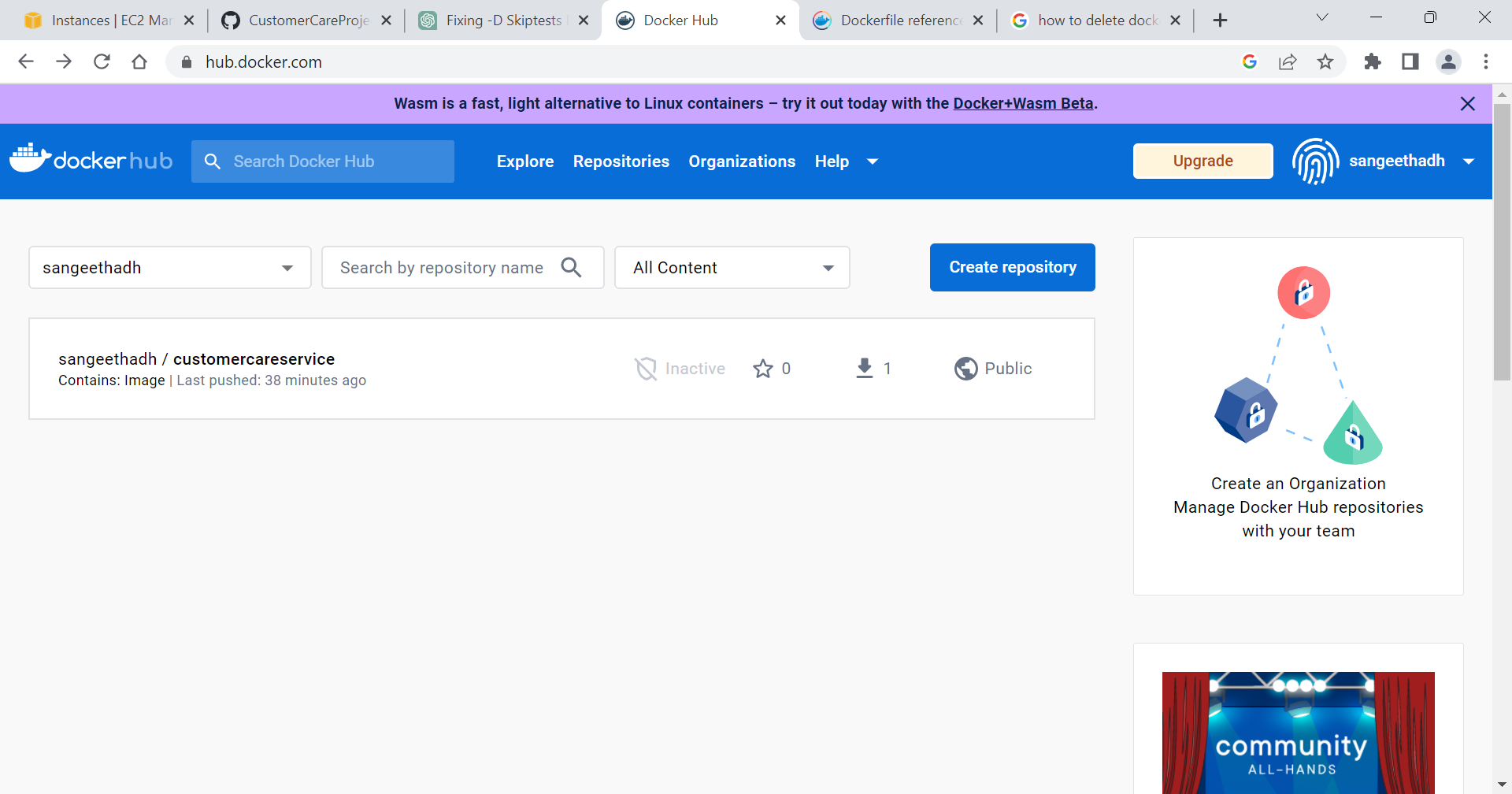
**Docker Registry:**

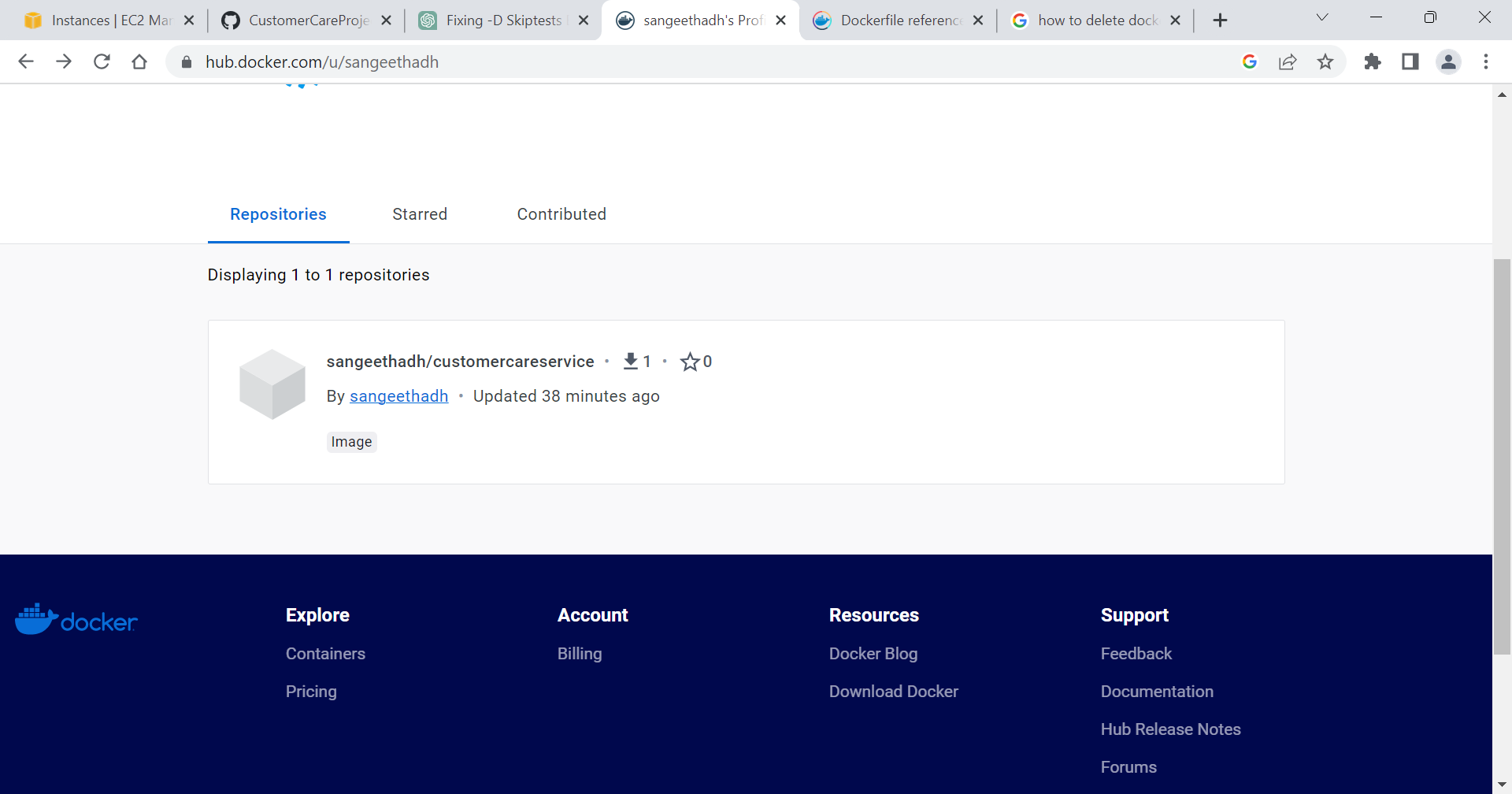
A Docker registry is a repository for Docker images, where you can store and share your images with others.

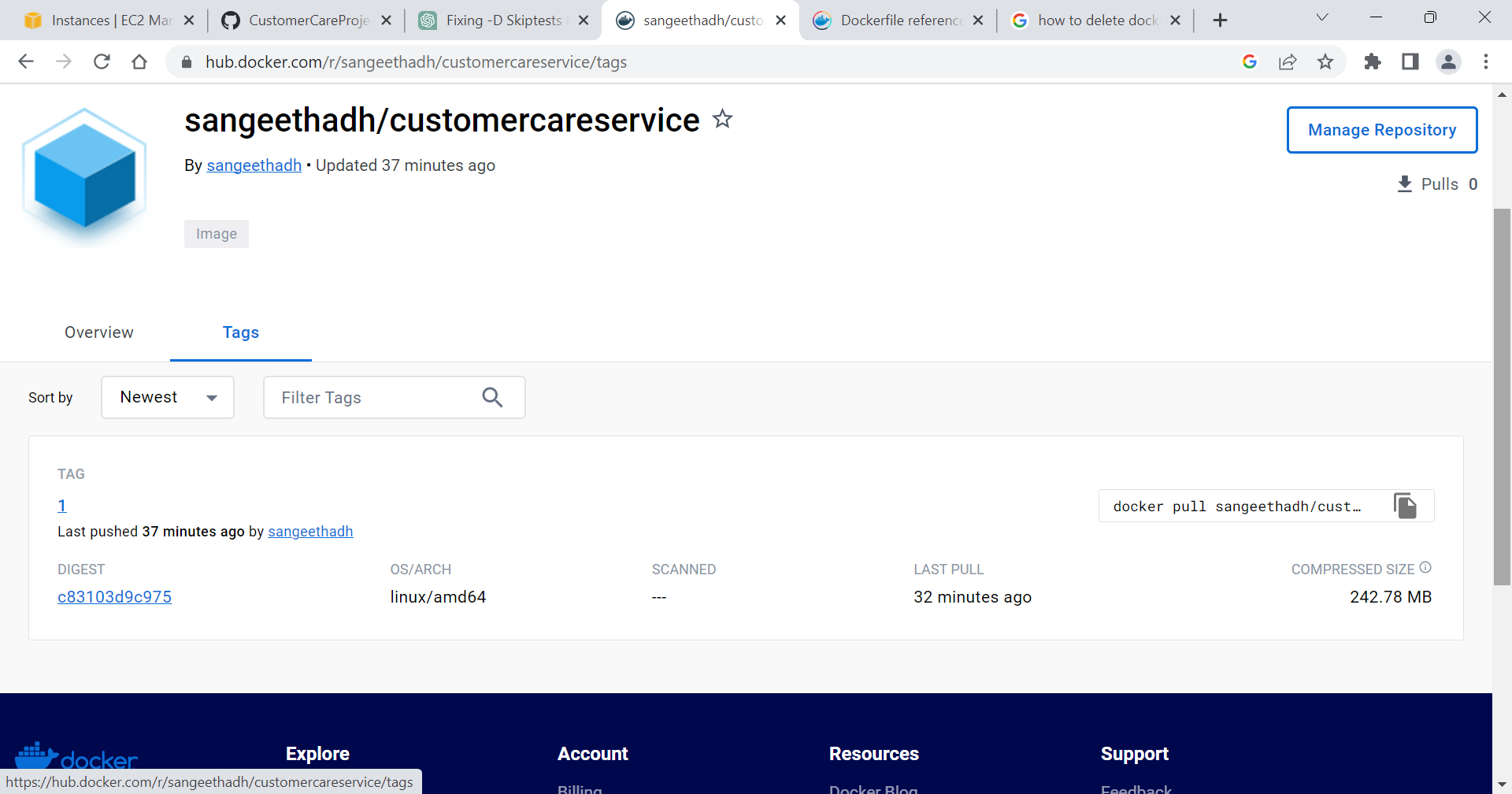


Step 19:Docker hub

Here, created docker image and pushed in docker registry







Step 20: **sudo chmod +x /usr/local/bin/docker-compose**

changes the permissions of the Docker Compose binary file to make it executable. By default, downloaded files are not executable, so you need to change the file permissions to allow you to run it as a program.

* "sudo": This is used to run the command with administrative privileges.
* "chmod": This is a command that changes the permissions of a file or directory.
* "+x": This specifies that the "execute" permission should be added to the file's existing permissions. This allows you to run the file as a program.
* "/usr/local/bin/docker-compose": This is the path to the Docker Compose binary file.

By running this command, you'll be able to use the Docker Compose binary file as a program. You can verify that the permissions have been changed by running "ls -l /usr/local/bin/docker-compose" and checking that the file has the "x" permission in the "Owner" section of the output.

Step 21: sudo curl -L https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m) -o /usr/local/bin/docker-compose

used to download the latest version of Docker Compose, which is a tool for defining and running multi-container Docker applications. The command downloads the binary file for Docker Compose and saves it to the "/usr/local/bin" directory on your machine.

• "sudo": This is used to run the command with administrative privileges.

• "curl": This is a command-line tool for transferring data from or to a server, using one of the supported protocols (HTTP, HTTPS, FTP, etc.).

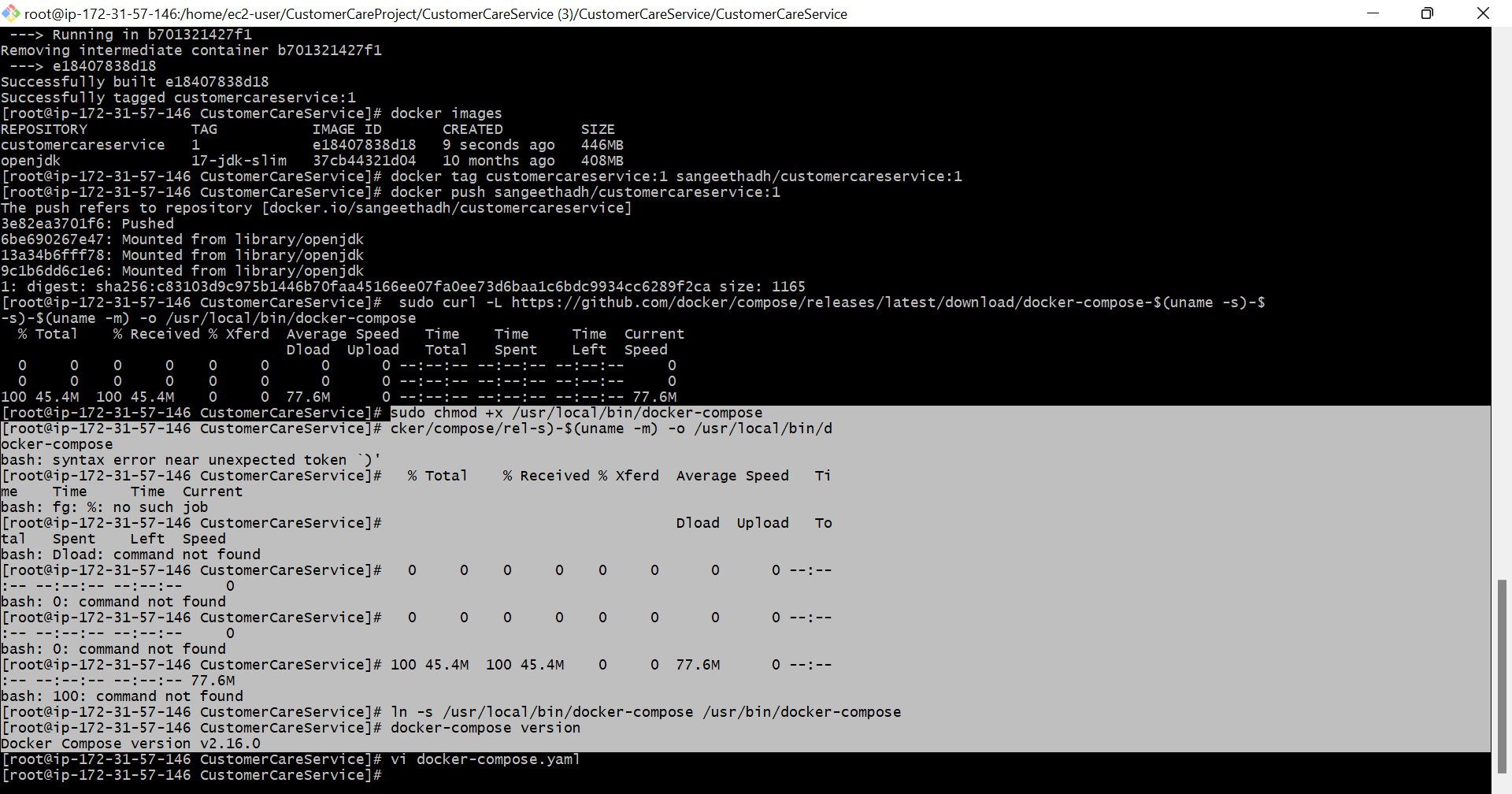
• "-L": This tells curl to follow any redirects that might be encountered during the download.

• "https://github.com/docker/compose/releases/latest/download/docker-compose-$(uname -s)-$(uname -m)": This is the URL of the Docker Compose binary file, which includes the latest release version, operating system, and machine architecture.

•"-o /usr/local/bin/docker-compose": This specifies the output file path for the downloaded binary file, which is the "/usr/local/bin" directory and the filename "docker-compose".

Step 22:Docker-compose version

used to display the version information for Docker Compose. It shows the version of Docker Compose, the version of Docker Engine that it's compatible with, and the location of the Docker Compose configuration file.

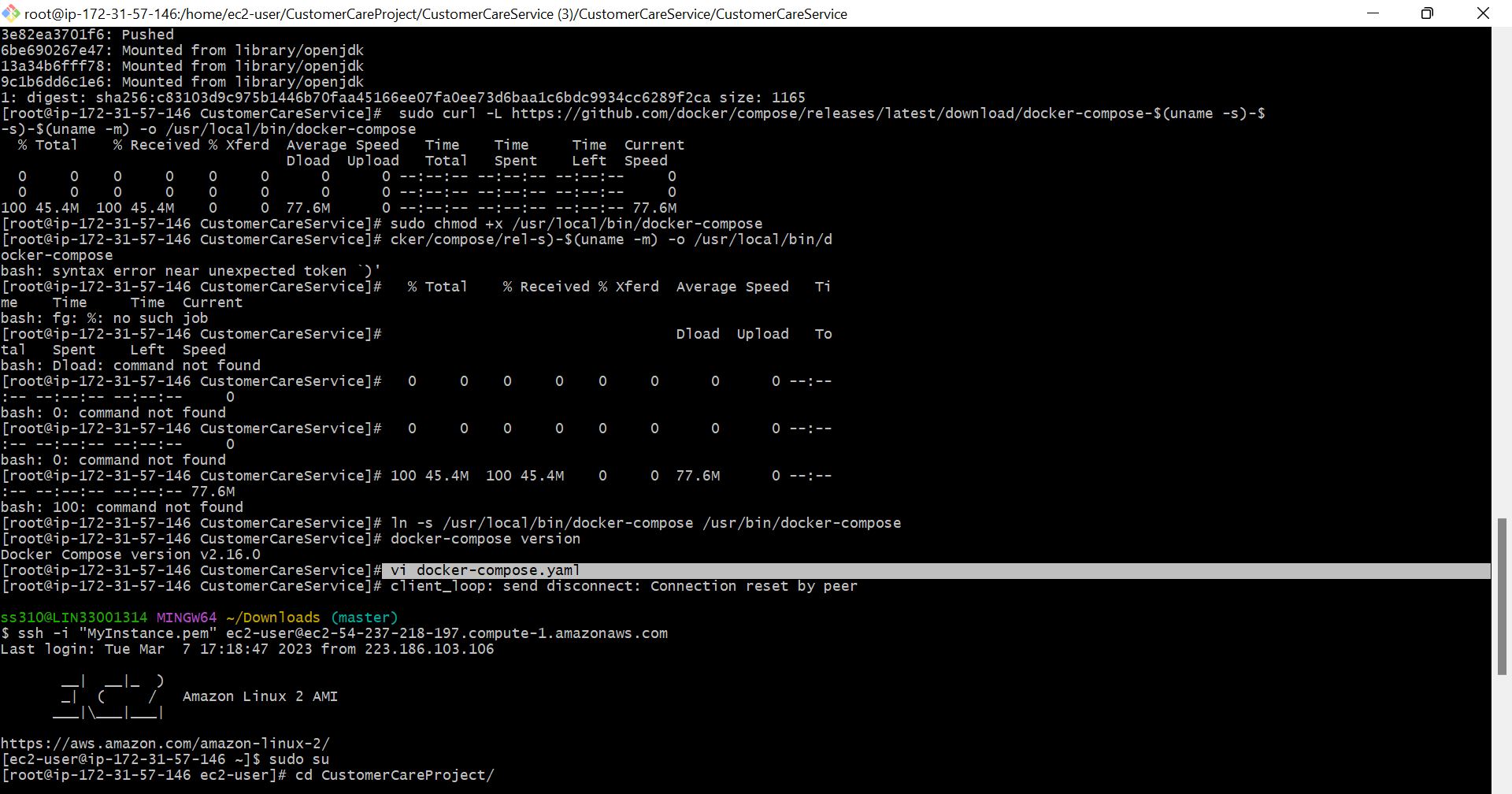


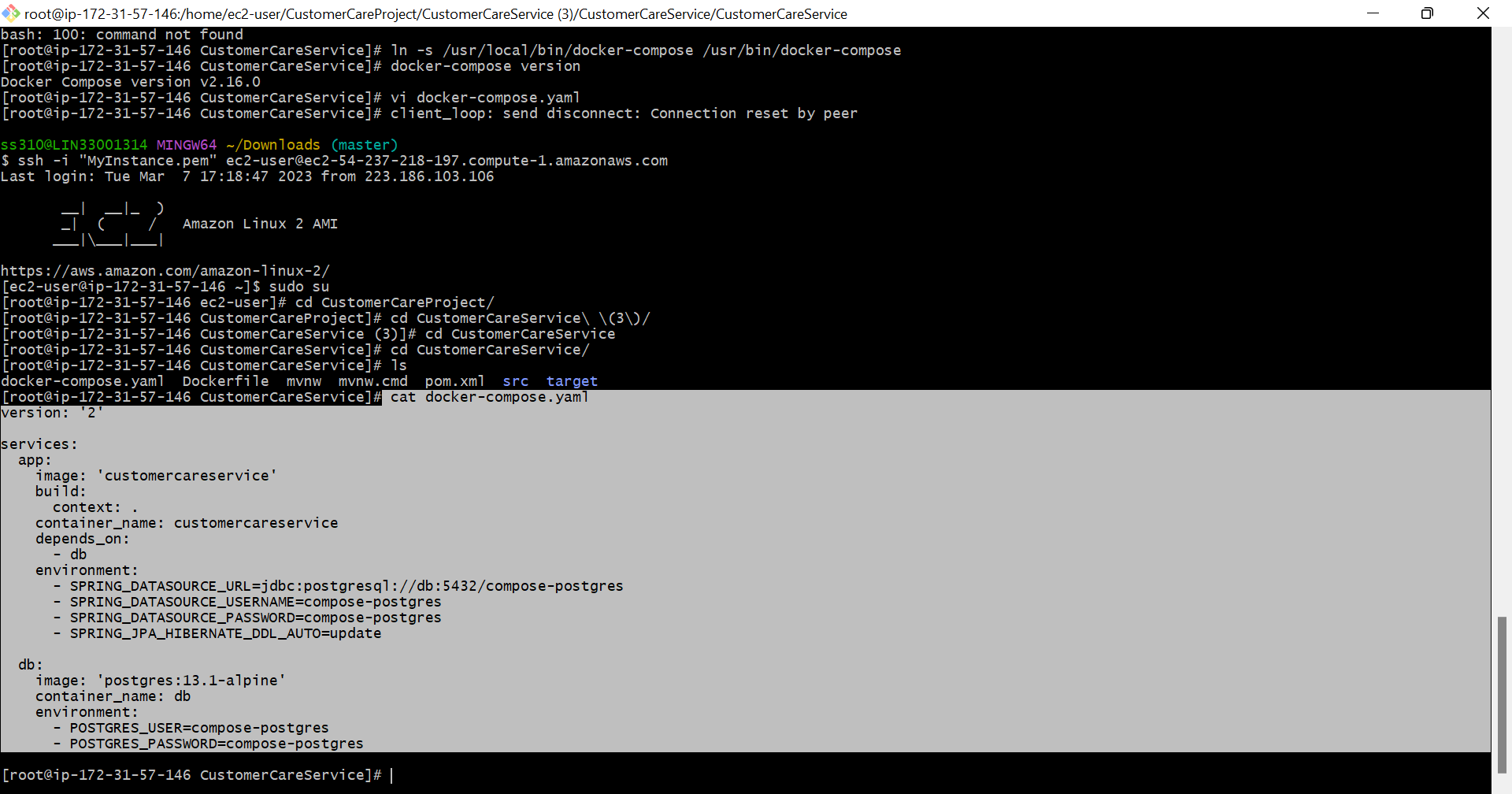
Step 23: **vi docker-compose.yaml**

"vi docker-compose.yaml" is a command to open the Docker Compose configuration file in the Vi editor. The Docker Compose configuration file is a YAML file that defines the services, networks, and volumes for a Docker application.

•vi": This is a command-line text editor that's available on most Unix-like systems.

•"docker-compose.yaml": This is the name of the Docker Compose configuration file.





Step 23: Docker-compose up

**docker-compose up** is a command that starts up all the services defined in your Docker Compose YAML file. It creates and starts containers for each service, and then attaches to the container logs,

