**1. Update the package manager**

yum update -

* yum (Yellowdog Updater, Modified) is the default package manager for RHEL-based distributions like CentOS and Amazon Linux.
* update ensures all installed packages are updated to the latest version.
* The - at the end is likely a typo; it should be:
* sudo yum update -y
* The -y flag automatically answers **"yes"** to prompts, allowing the update to proceed without manual confirmation.

**2. Install Docker**

sudo yum install -y docker

* Installs the **Docker engine** from the default package repository.
* -y automatically confirms the installation.

**3. Start the Docker service**

sudo service docker start

* service is used to manage system services.
* docker start starts the **Docker daemon**, allowing containers to be run.
* Alternative command (for systemd-based OS):
* sudo systemctl start docker

**4. Add EC2 user to the Docker group**

sudo usermod -a -G docker ec2-user

* usermod modifies user account properties.
* -a (append) ensures the user is **added to** a group without removing existing group memberships.
* -G docker adds ec2-user to the **docker group**.
* This allows ec2-user to run Docker commands **without** sudo.
* You must log out and log back in for the changes to take effect.

**5. Check Docker version**

docker version

* Displays the installed Docker **client and server version**.
* Confirms Docker is installed and running correctly.

**6. Create a Dockerfile**

vi dockerfile

* vi opens the **Vi editor** to create or edit a file named dockerfile.
* A **Dockerfile** contains instructions for building a custom Docker image.

**7. Build a Docker Image**

docker build -t latest .

* docker build creates a **Docker image** from a **Dockerfile**.
* -t latest assigns the tag **"latest"** to the image.
* . (dot) specifies the **current directory** as the build context (must contain a Dockerfile).

**8. Edit the Dockerfile Again**

vi dockerfile

* Opens the **Dockerfile** again for modifications.

**9. Build the Docker Image Again**

docker build -t latest .

* Rebuilds the Docker image after modifying the Dockerfile.

**10. List Docker Images**

docker images

* Displays all **locally available** Docker images.
* Shows:
  + **REPOSITORY** (name of the image)
  + **TAG** (e.g., latest)
  + **IMAGE ID** (unique identifier)
  + **CREATED** (time since the image was built)
  + **SIZE** (disk space used)

**11. Run a Container from the Image**

docker run -d -p 70:80 latest

* docker run creates and starts a **new container**.
* -d runs the container in **detached mode** (background).
* -p 70:80 maps **port 70 on the host** to **port 80 inside the container**.
* latest is the **name of the image** used to create the container.
* The running container serves content on **port 70** of the EC2 instance.

**12. List Running Containers**

docker container ls

* Displays currently running containers.
* Shows:
  + **Container ID**
  + **Image Name**
  + **Command** (entry point executed)
  + **Status** (e.g., "Up 5 minutes")
  + **Ports** (e.g., 0.0.0.0:70->80/tcp)

To see **all containers** (including stopped ones):

docker container ls -a

**13. Access the Running Container**

docker exec -it a3a7d17d163a bash

* docker exec runs a command inside a **running container**.
* -it:
  + -i keeps input open (interactive mode).
  + -t allocates a **pseudo-terminal**.
* a3a7d17d163a is the **container ID** (you can get it from docker ps).
* bash opens a **Bash shell** inside the container.

Once inside the container, you can navigate and inspect files like a normal Linux system.

To exit, type:

exit

**14. View Command History**

history

* Displays all previously executed commands.
* Useful for debugging or repeating past commands.

To save the history to a file:

history > commands.log

To search for a specific command:

history | grep docker

**Summary**

| **Command** | **Purpose** |
| --- | --- |
| yum update -y | Updates installed packages |
| sudo yum install -y docker | Installs Docker engine |
| sudo service docker start | Starts the Docker service |
| sudo usermod -a -G docker ec2-user | Adds ec2-user to Docker group |
| docker version | Checks installed Docker version |
| vi dockerfile | Opens/creates a Dockerfile |
| docker build -t latest . | Builds a Docker image from the Dockerfile |
| docker images | Lists available Docker images |
| docker run -d -p 70:80 latest | Runs a container with port mapping |
| docker container ls | Shows running containers |
| docker exec -it <container\_id> bash | Opens a shell inside the container |
| history | Shows command history |

**PORTAL PROBLEM ONE**

**1. Update Installed Packages**

yum update -y

* **yum**: Package manager for RHEL-based distributions (CentOS, Amazon Linux).
* **update**: Updates all installed packages to their latest versions.
* **-y**: Automatically confirms installation to proceed without user input.

**2. Install Docker**

sudo yum install -y docker

* Installs **Docker Engine** from the default repository.
* **-y** ensures automatic installation without manual confirmation.

**3. Start the Docker Service**

sudo service docker start

* Starts the **Docker daemon**, which manages containers.
* Alternative command for **systemd-based** OS:
* sudo systemctl start docker

**4. Add EC2 User to the Docker Group**

sudo usermod -a -G docker ec2-user

* Adds ec2-user to the **docker group**, so they can run Docker commands without sudo.
* **-a -G docker** appends the user to the Docker group without affecting other group memberships.
* Changes take effect **after logging out and back in**.

**5. Verify Docker Installation**

docker version

* Displays installed **Docker client and server versions** to confirm Docker is running.

**6. Pull Apache TomEE Docker Image**

docker pull tomee

* **Downloads the tomee image** (Apache Tomcat with EE capabilities) from **Docker Hub**.
* tomee provides a **pre-configured Tomcat server** for running Java web applications.

To verify the downloaded image:

docker images

**7. Run the Tomcat Container**

docker run -d --name tomcat-container-1 -p 80:8080 tomee

* docker run: Creates and starts a new container.
* -d: Runs the container in **detached mode** (in the background).
* --name tomcat-container-1: Assigns a custom **container name** (tomcat-container-1).
* -p 80:8080: Maps **port 80 (host)** to **port 8080 (container)**.
  + Access the Tomcat server at http://<EC2-PUBLIC-IP>
* tomee: Specifies the **Docker image** used to create the container.

To check running containers:

docker container ls

**8. Create a Directory for the Website**

mkdir my-website

* Creates a new directory **my-website** to store the webpage file.

**9. List Files in Current Directory**

ls

* Lists the files and directories in the **current directory**.
* Confirms that my-website was successfully created.

**10. Navigate to the my-website Directory**

cd my-website

* Moves into the my-website directory.

**11. Create an HTML File for the Website**

echo '<html><body><h1>Hello from Tomcat on EC2!</h1></body></html>' > index.html

* **echo**: Outputs the provided string.
* > Redirects the output into a new file named **index.html**.
* Creates a basic HTML file that displays **"Hello from Tomcat on EC2!"**.

To confirm the file was created:

ls

cat index.html

**12. Check Running Containers**

docker container ls

* Displays **active** containers with details like:
  + **Container ID**
  + **Image name**
  + **Running status**
  + **Port mapping**

**13. Access the Running Tomcat Container**

docker exec -it tomcat-container-1 bash

* docker exec: Executes a command inside a running container.
* -it:
  + -i Keeps the input open (interactive mode).
  + -t Allocates a pseudo-terminal.
* tomcat-container-1: Specifies the **container name**.
* bash: Opens an interactive **Bash shell** inside the container.

To exit the container shell:

exit

**14. Copy index.html into Tomcat’s Web Server Directory**

docker cp index.html tomcat-container-1:/usr/local/tomee/webapps/ROOT/index.html

* docker cp: Copies files **from the host machine to the container**.
* index.html: The file to copy (from the **host machine**).
* tomcat-container-1:/usr/local/tomee/webapps/ROOT/index.html:
  + Copies the file **into the container** at Tomcat’s web root directory.
  + Overwrites the default **index.html** file.

To verify inside the container:

docker exec -it tomcat-container-1 ls /usr/local/tomee/webapps/ROOT/

**15. Restart the Tomcat Container**

docker restart tomcat-container-1

* **Restarts the container** to apply the new index.html.
* Required because Tomcat **does not auto-detect new files** in its web directory.

Now, the custom web page should be available at:

http://<EC2-PUBLIC-IP>

**Summary of Commands**

| **Command** | **Purpose** |
| --- | --- |
| yum update -y | Updates installed packages |
| sudo yum install -y docker | Installs Docker |
| sudo service docker start | Starts Docker daemon |
| sudo usermod -a -G docker ec2-user | Adds ec2-user to the Docker group |
| docker version | Checks installed Docker version |
| docker pull tomee | Downloads Apache TomEE image from Docker Hub |
| docker run -d --name tomcat-container-1 -p 80:8080 tomee | Runs Tomcat container with port mapping |
| mkdir my-website | Creates a directory for the website |
| ls | Lists files in the current directory |
| cd my-website | Moves into the my-website directory |
| echo '<html>...</html>' > index.html | Creates a basic HTML file |
| docker container ls | Lists running containers |
| docker exec -it tomcat-container-1 bash | Opens a Bash shell inside the Tomcat container |
| docker cp index.html tomcat-container-1:/usr/local/tomee/webapps/ROOT/index.html | Copies index.html into the Tomcat server directory |
| docker restart tomcat-container-1 | Restarts the Tomcat container to apply changes |

**Expected Result**

* TomEE runs on **port 80** of the EC2 instance.
* The custom webpage is **live** at:
* http://<EC2-PUBLIC-IP>
* It should display:
* <html><body><h1>Hello from Tomcat on EC2!</h1></body></html>

**Next Steps**

1. **Allow HTTP Traffic in AWS Security Group**:
   * Ensure that **port 80** is open in your **EC2 security group** settings.
   * Otherwise, the webpage won’t be accessible from the browser.
2. **Check Logs if Issues Occur**:
3. docker logs tomcat-container-1