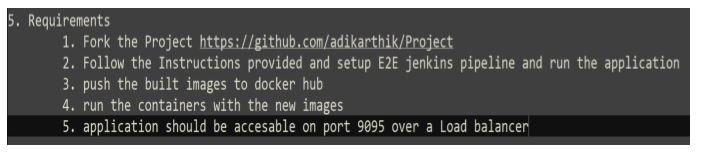
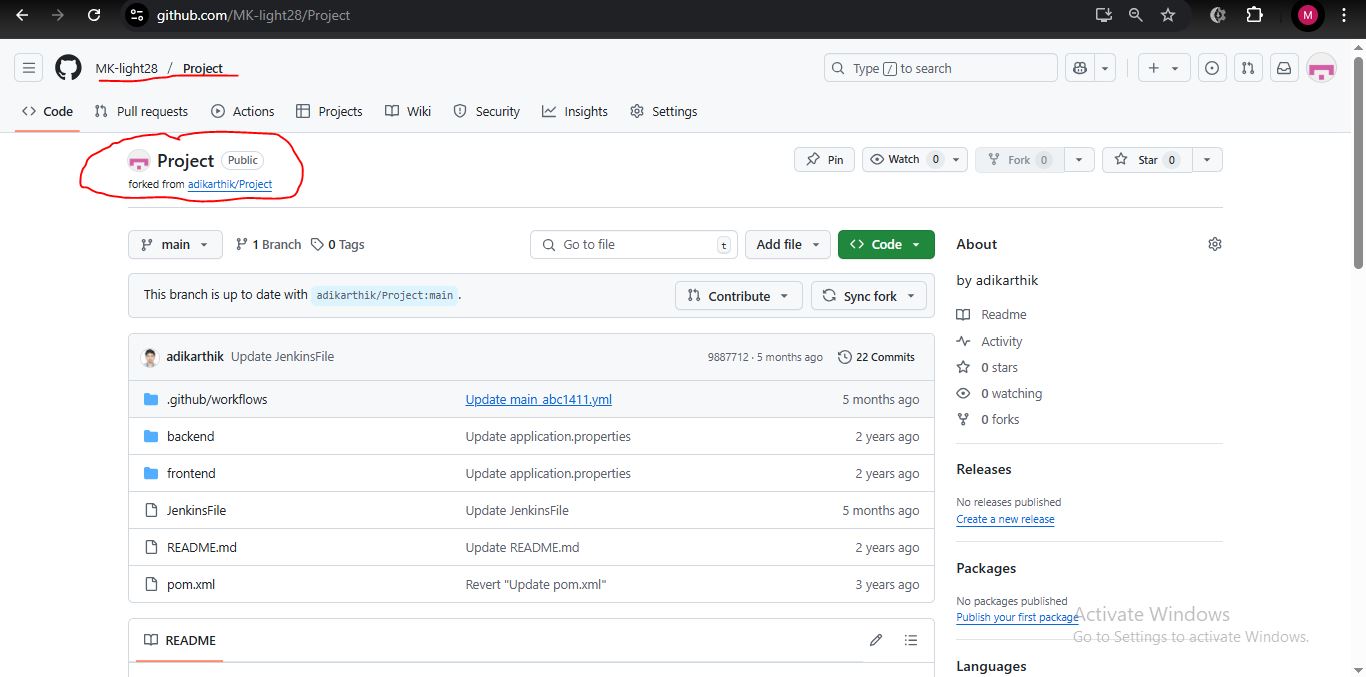
Mahesh Kumar

5th Task

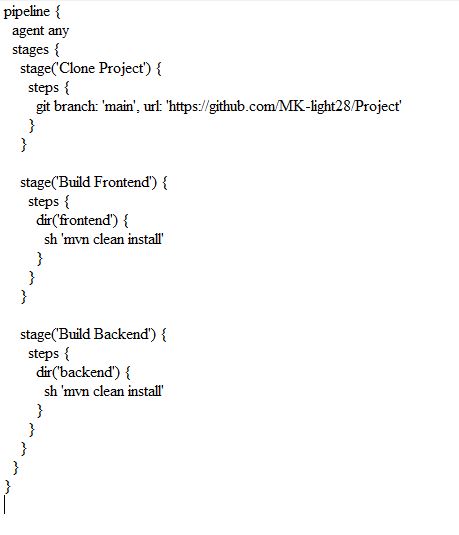
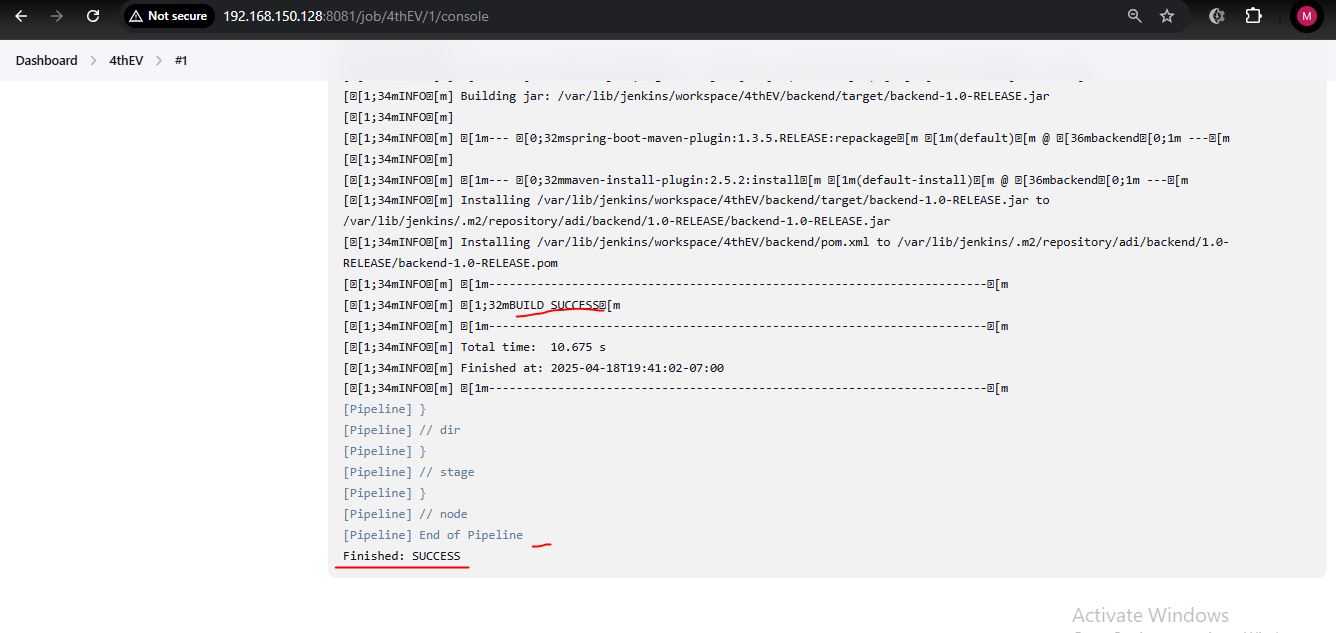
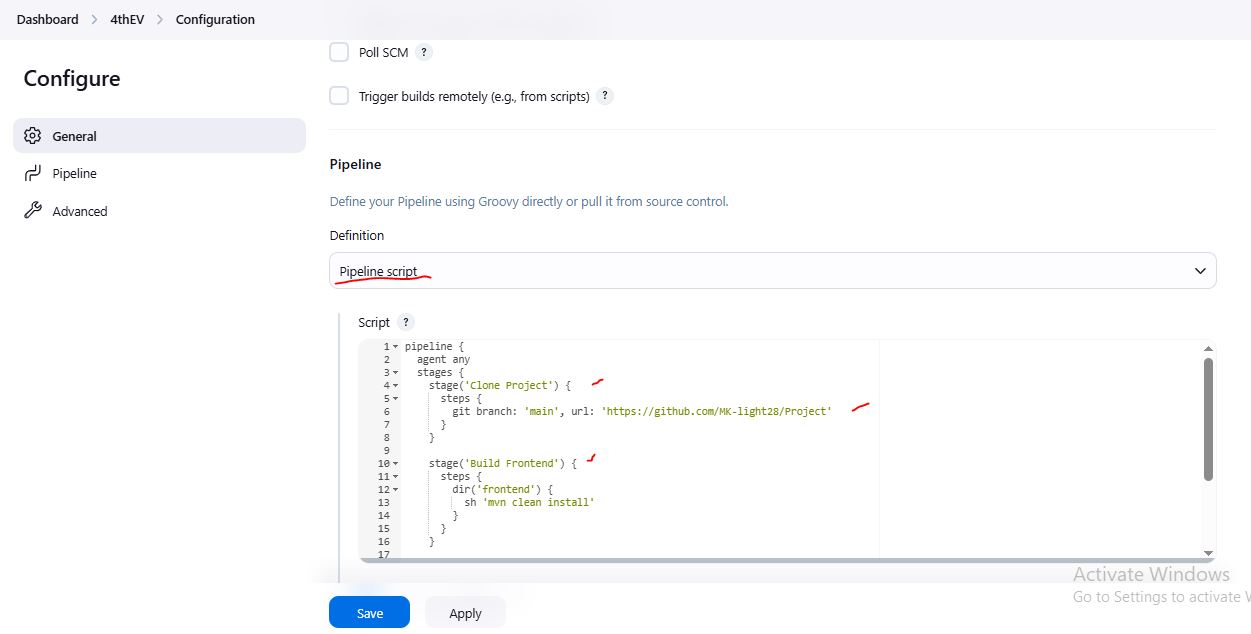
19-April-2025



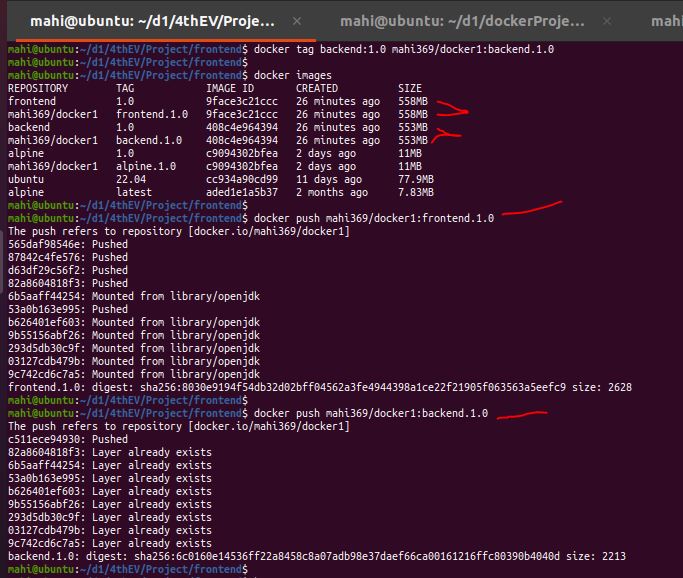
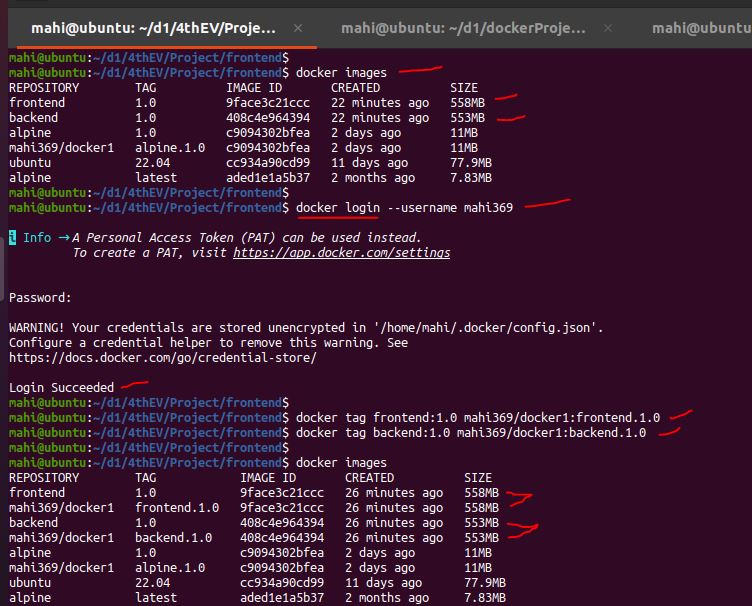
1. Forking the github repo



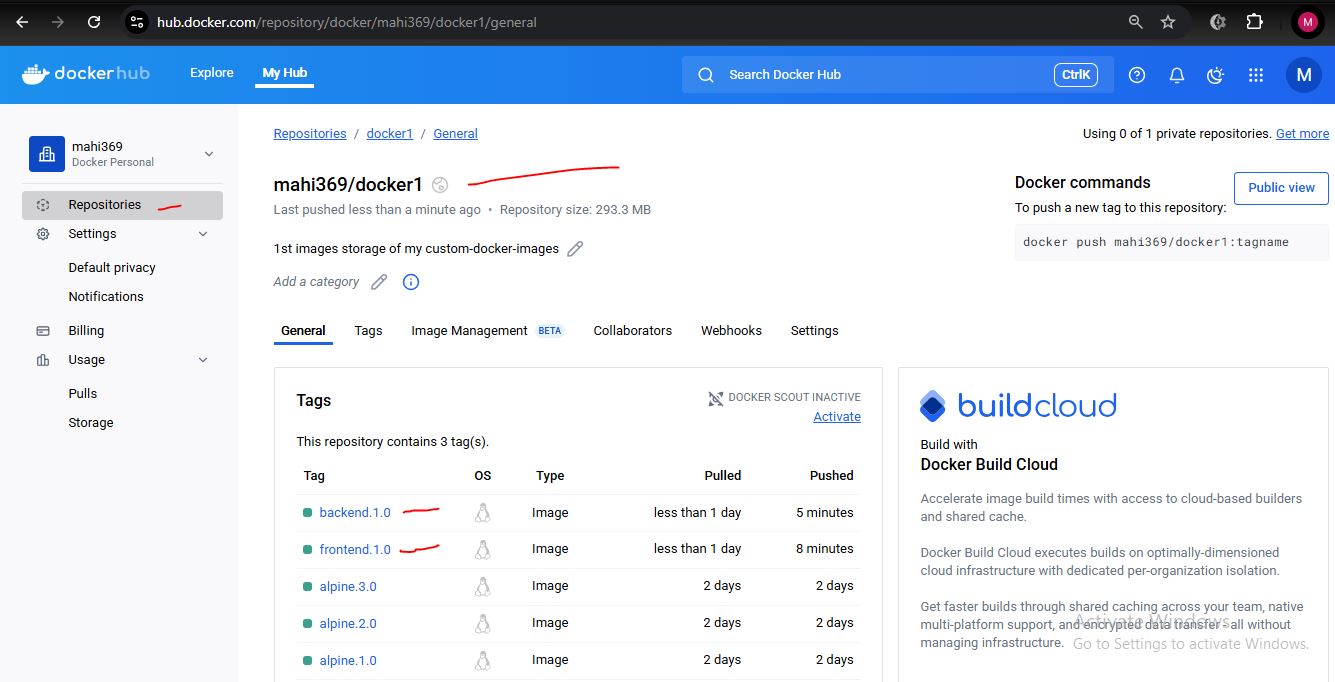
2. Jenkins Pipeline job



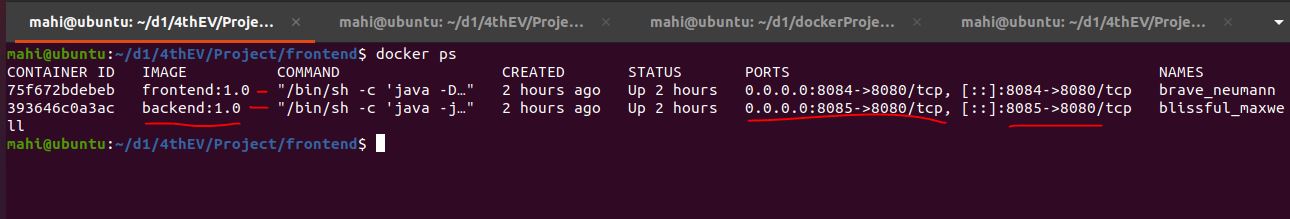
3. Docker images built and tagged and pushed to docker-hub



4. Docker-hub images



5. containers running – frontend and backend containers



**COMMANDS USED:**

1. FORKING

forked repo from "adikarthik" github to my personal github account "MK-light28", using github GUI

<https://github.com/MK-light28/Project>

#repo URL after forking

------------------------------------------------------------

2. PIPELINE SCRIPT

pipeline {

agent any

stages {

stage('Clone Project') {

steps {

git branch: 'main', url: 'https://github.com/MK-light28/Project'

}

}

stage('Build Frontend') {

steps {

dir('frontend') {

sh 'mvn clean install'

}

}

}

stage('Build Backend') {

steps {

dir('backend') {

sh 'mvn clean install'

}

}

}

}

}

------------------------------------------------------------

3. DOCKER IMAGE BUILDING

$ git clone https://github.com/MK-light28/Project

#the project is downloaded into the local linux system

In backend Directory

/Project/backend$ ls

Dockerfile pom.xml Procfile src system.properties target

$ docker build .

#backend image is build, without a name

In Frontend directory

/Project/frontend$ ls

Dockerfile pom.xml Procfile src system.properties target

$ docker build .

#frontend image is build, without a name

$ docker tag 9face3c21ccc frontend:1.0

$ docker tag 408c4e964394 backend:1.0

#Naming the images created

$ docker login --username mahi369

#login to dockerhub account from local docker

$ docker tag frontend:1.0 mahi369/docker1:frontend.1.0

$ docker tag backend:1.0 mahi369/docker1:backend.1.0

#tagging both created images to push to dockerhub cloud

$ docker push mahi369/docker1:frontend.1.0

$ docker push mahi369/docker1:backend.1.0

#pushing the tagged-images in to dockerhub cloud

------------------------------------------------------------

4. RUNNING THE FRONTEND and BACKEND CONTAINERS USING "docker run"

$ docker run -dit -p 8085:8080 backend:1.0

$ docker run -dit -p 8084:8080 -e export BACKEND=http://localhost:8085 frontend:1.0

#running both containers, backend container is run first, then frontend

$ docker ps

#check if containers are running

$ docker exec -it 75f672bdebeb sh

where, 75f672bdebeb = frontend-container-ID

#accessing the container for files exploring and trouble shooting

$ docker exec -it 393646c0a3ac sh

where, 393646c0a3ac = backend-container-ID

#accessing the container for files exploring and trouble shooting

------------------------------------------------------------

5. OTHER COMMANDS USED

$ mvn clean spring-boot:run

$ mvn clean install

#builds the project artifacts using pom.xml

$ docker images

#lists all images built and downloaded

$ docker ps

#lists all running containers

$ docker top <containerID>

#to know the containers process ID

$ sudo kill -9 <containerPID>

#stops the container forcefully

-------------------------------------------------

6. URLs USED:

http://localhost:8080/

#frontend webpage accessing from browser

http://localhost:8080/docs

#backend files accessing from browser

-- after running docker containers successfully

http://localhost:8084/

#frontend webpage accessing from browser

http://localhost:8085/docs

#backend files accessing from browser