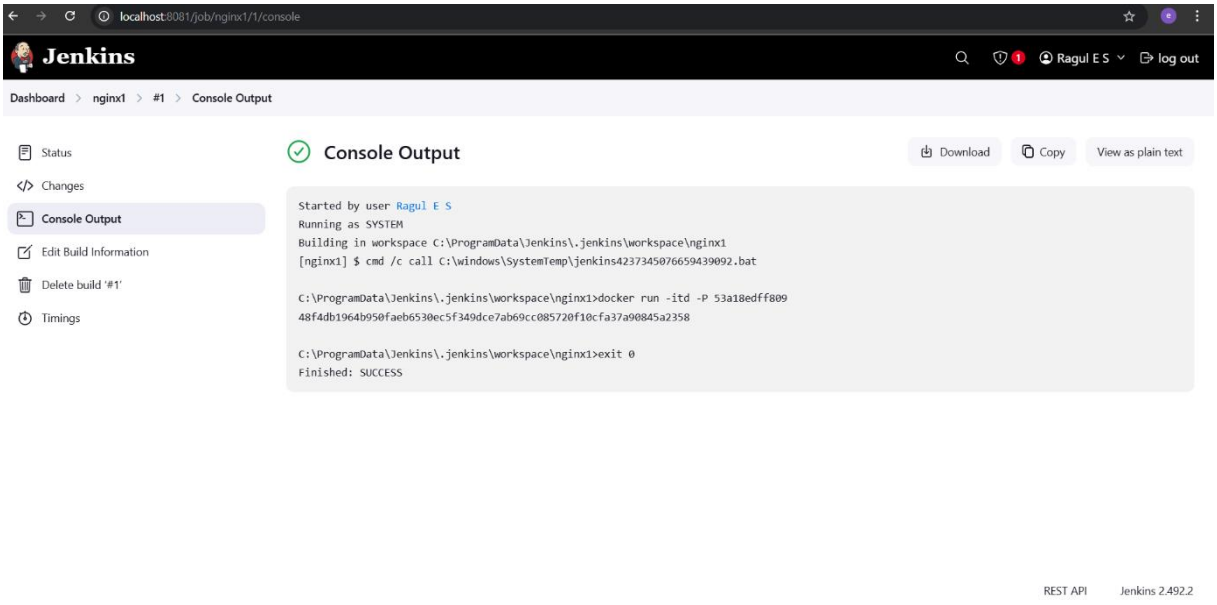
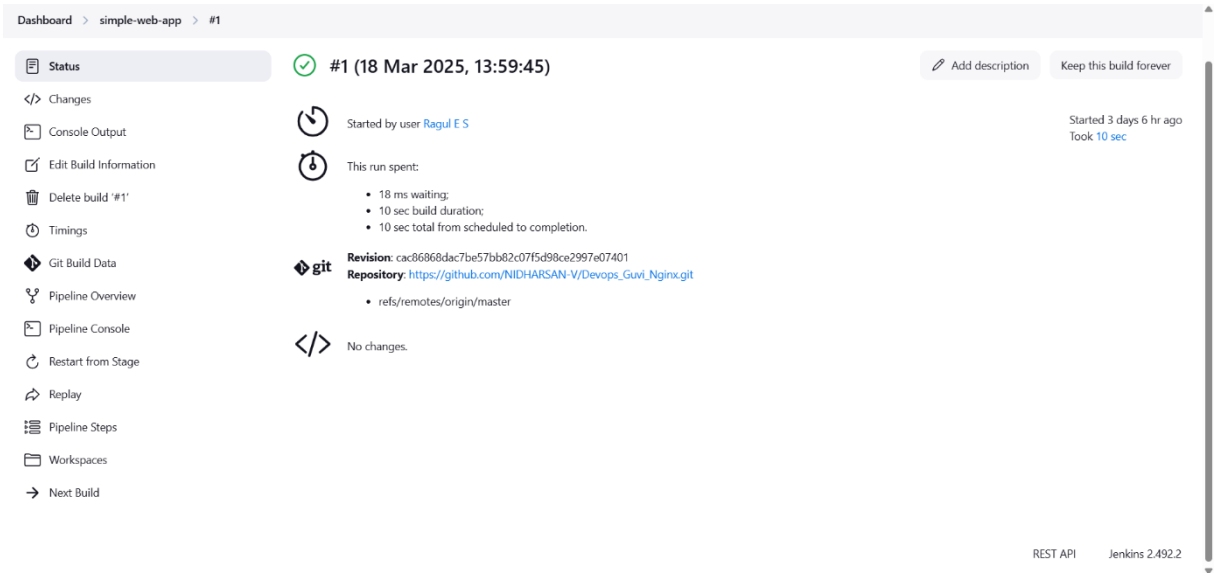
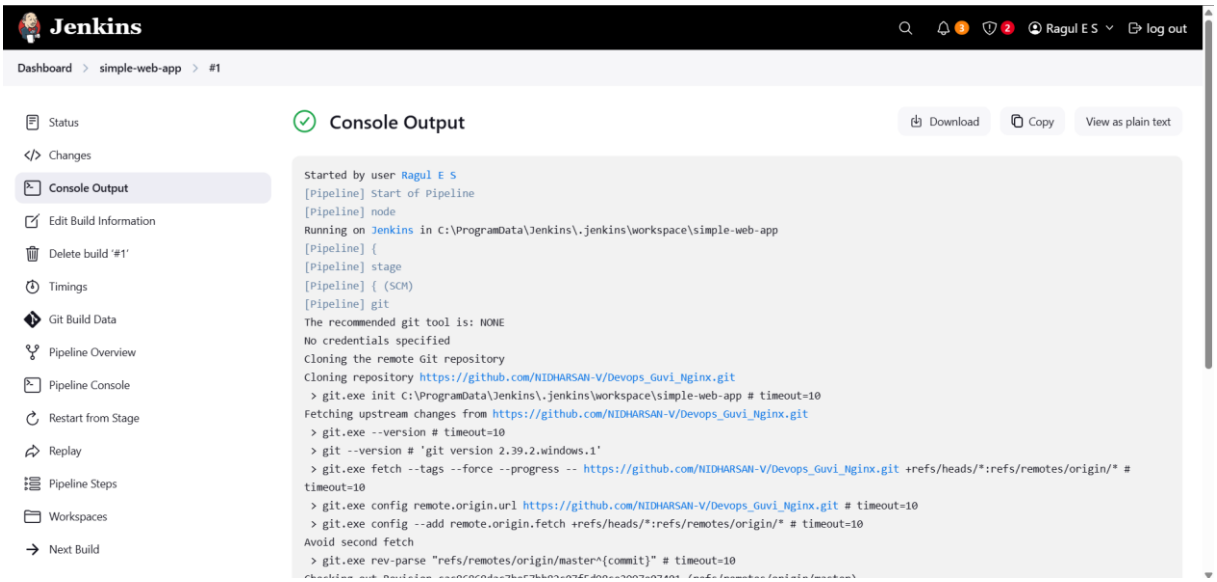
DevOps Day-2

Learned about Jenkins in windows used to create docker image and push to a docker container using functions.

Output of building a simple freestyle project:

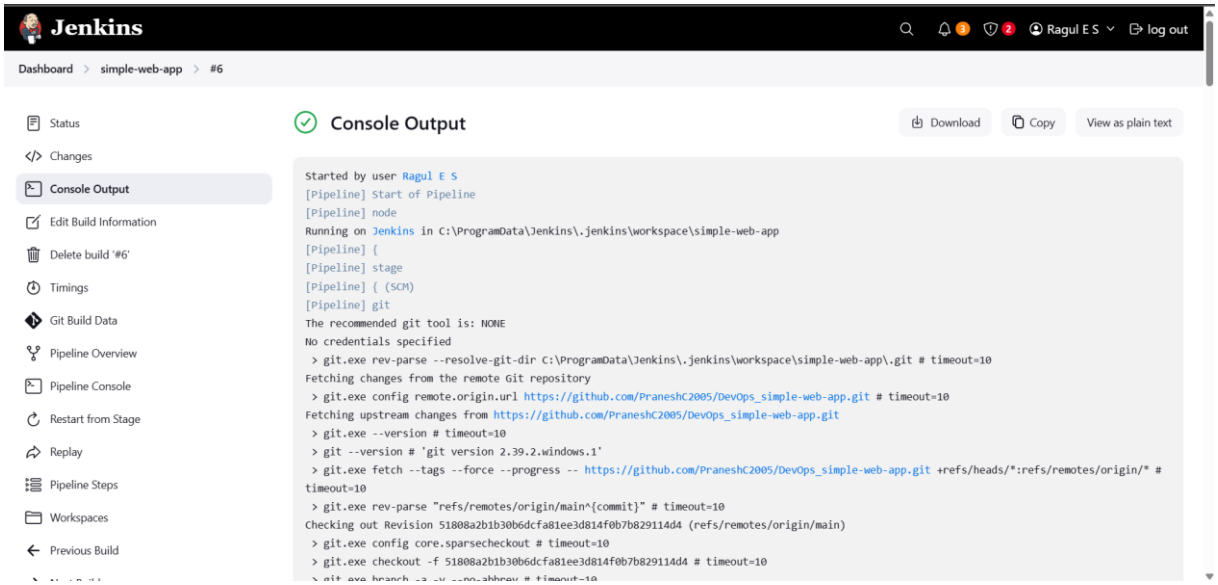


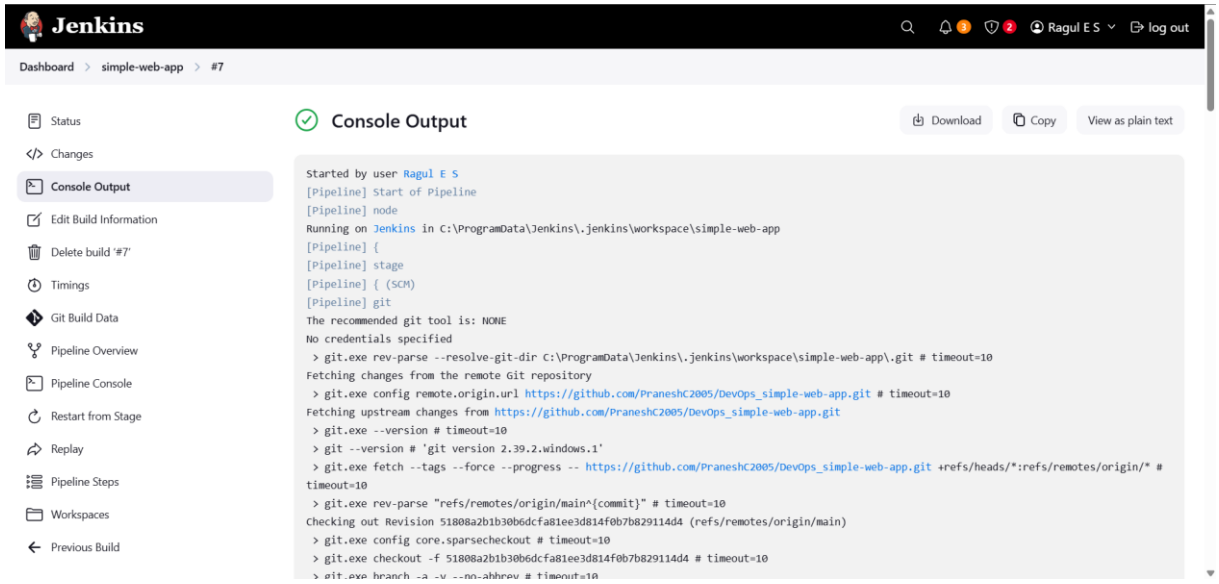


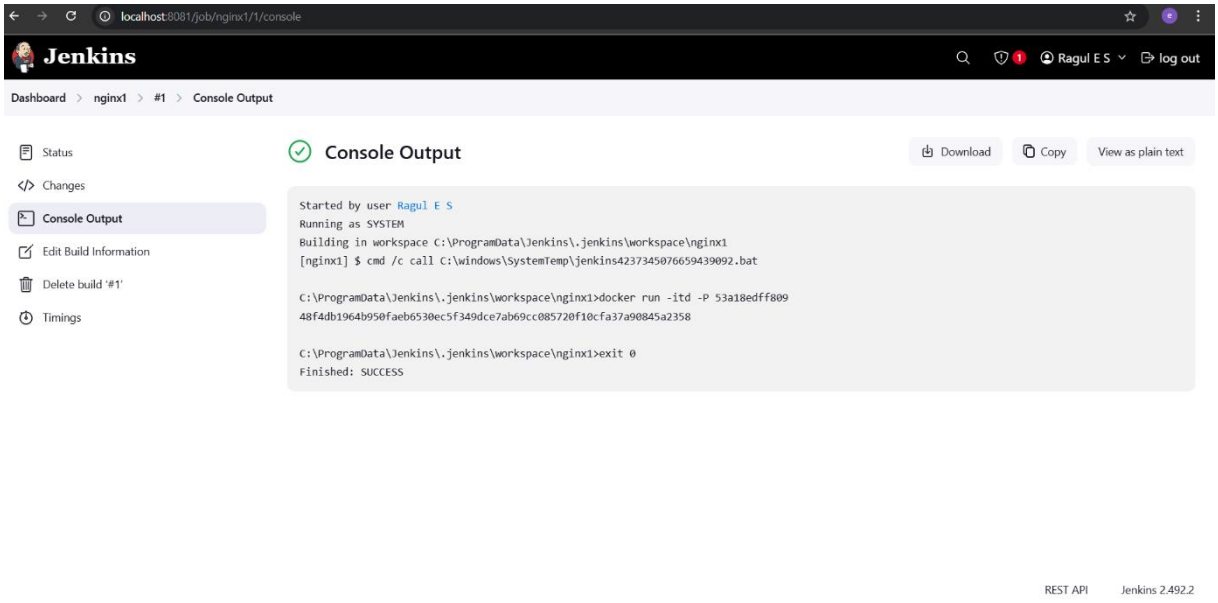


Script ?

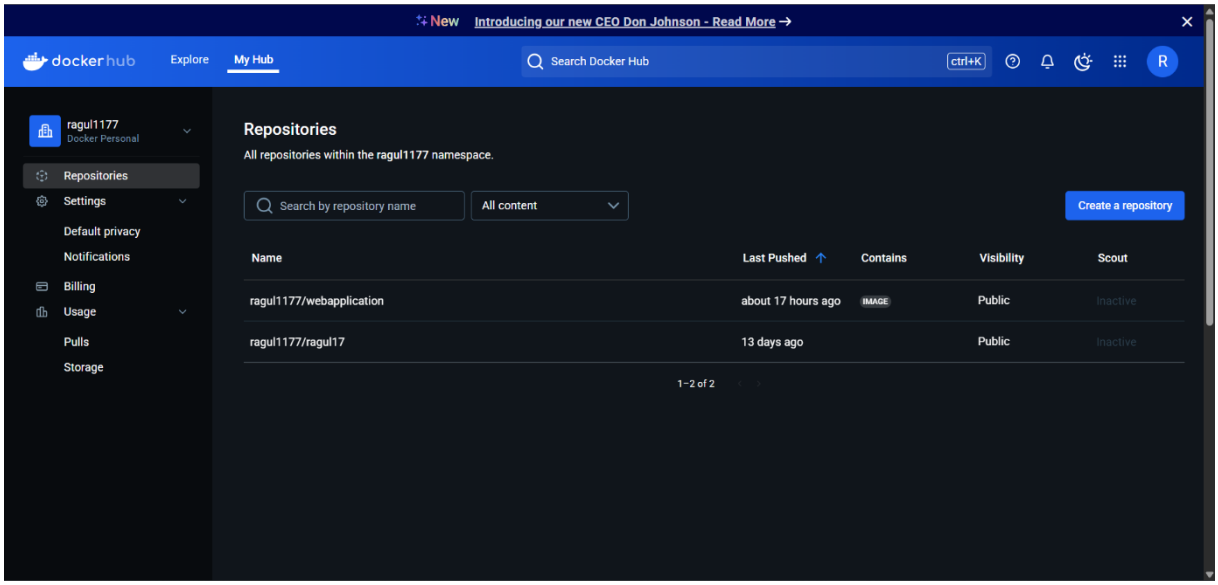
```
pipeline {
    agent any

    stages {
        stage('SCM') {
            steps {
                git branch: 'main', url: 'https://github.com/PraneshC2005/DevOps_simple-web-app.git'
            }
        }
        stage('Build'){
            steps {
                bat 'mvn clean install'
            }
        }
        stage('build to images') {
            steps {
                script{
                    bat "docker build -t ragul1177/webapplication ."
```


Jenkins
Ragul E S
log out
Dashboard > simple-web-app > #6
Status
Changes
Console Output
Edit Build Information
Delete build '#6'
Timings
Git Build Data
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build
Console Output
Download
Copy
View as plain text
Started by user Ragul E S
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\simple-web-app
[Pipeline] {
[Pipeline] stage
[Pipeline] { (SCM)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\simple-web-app\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/PraneshC2005/DevOps_simple-web-app.git # timeout=10
Fetching upstream changes from https://github.com/PraneshC2005/DevOps_simple-web-app.git
> git.exe --version # timeout=10
> git --version # 'git version 2.39.2.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/PraneshC2005/DevOps_simple-web-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 51808a2b1b30b6dcfa81ee3d814f0b7b829114d4 (refs/remotes/origin/main)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 51808a2b1b30b6dcfa81ee3d814f0b7b829114d4 # timeout=10


Jenkins
Ragul E S
log out
Dashboard > simple-web-app > #7
Status
Changes
Console Output
Edit Build Information
Delete build '#7'
Timings
Git Build Data
Pipeline Overview
Pipeline Console
Restart from Stage
Replay
Pipeline Steps
Workspaces
Previous Build
Console Output
Download
Copy
View as plain text
Started by user Ragul E S
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in C:\ProgramData\Jenkins\.jenkins\workspace\simple-web-app
[Pipeline] {
[Pipeline] stage
[Pipeline] { (SCM)
[Pipeline] git
The recommended git tool is: NONE
No credentials specified
> git.exe rev-parse --resolve-git-dir C:\ProgramData\Jenkins\.jenkins\workspace\simple-web-app\.git # timeout=10
Fetching changes from the remote Git repository
> git.exe config remote.origin.url https://github.com/PraneshC2005/DevOps_simple-web-app.git # timeout=10
Fetching upstream changes from https://github.com/PraneshC2005/DevOps_simple-web-app.git
> git.exe --version # timeout=10
> git --version # 'git version 2.39.2.windows.1'
> git.exe fetch --tags --force --progress -- https://github.com/PraneshC2005/DevOps_simple-web-app.git +refs/heads/*:refs/remotes/origin/* # timeout=10
> git.exe rev-parse "refs/remotes/origin/main^{commit}" # timeout=10
Checking out Revision 51808a2b1b30b6dcfa81ee3d814f0b7b829114d4 (refs/remotes/origin/main)
> git.exe config core.sparsecheckout # timeout=10
> git.exe checkout -f 51808a2b1b30b6dcfa81ee3d814f0b7b829114d4 # timeout=10



Pushed to Docker Hub:



Executed this project using the Jenkins File:

```
pipeline {
    agent any
```

```
    stages {
        stage('SCM') {
            steps {
                git branch: 'main', url:
'https://github.com/PraneshC2005/DevOps_simple-web-app.git'
            }
        }
        stage('Build'){
            steps {
                bat 'mvn clean install'
            }
        }
        stage('build to images') {
            steps {
                script{
                    bat "docker build -t ragul1177/webapplication ."
                }
            }
        }
        stage('docker push hub') {
            steps {
                script{
                    withDockerRegistry(credentialsId: 'docker_crud', url:
'https://index.docker.io/v1/') {
```

```
    bat 'docker push ragul1177/webapplication'
}
            }
          }
        }
      }
}
```

Docker is a platform that provides virtual containers on which an application can be deployed independent of the underlying OS of the server. Further the container can be created from a replica called docker image which contains all the dependencies and can run on any OS that has docker engine, with similar results.

BASIC DOCKER COMMANDS

Display docker images available in our machine

$ docker images

Download docker image.

$ docker pull <image-name / image-id> Run

docker image.

$ docker run <image-name / image-id> Delete

docker image.

$ docker rmi <image-name / image-id> Display

all running docker containers.

$ docker ps

Display all running and stopped containers.

$ docker ps -a

Delete docker container.

$ docker rm <container-id>

Delete docker image forcefully.

$ docker rmi -f <image-id> Stop

Docker container.

$ docker stop <container-id>

JENKINS

Jenkins is an open-source automation tool written in Java programming language that allows continuous integration. Jenkins offers a straightforward way to set up a continuous integration or continuous delivery environment for almost any combination of languages and source code repositories using pipelines, as well as automating other routine development tasks.

The following are the main or most popular Jenkins use cases:

●Continuous Integration: With Jenkins pipelines, we can achieve CI for both applications and infrastructure as code.

●Continuous Delivery: You can set up well-defined and automated application delivery workflows with Jenkins pipelines.

Jenkins achieves CI (Continuous Integration) and CD (Continuous Deployment) with the help of plugins. Plugins are used to allow the integration of various DevOps stages. If you want to integrate a particular tool, you must install the plugins for that tool.