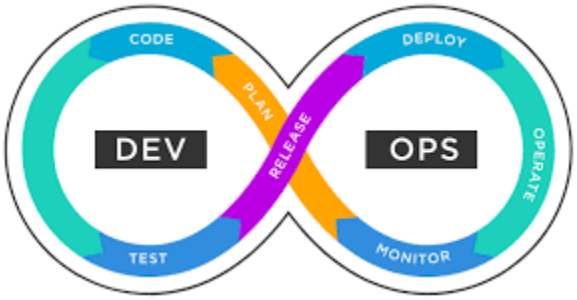
International Institute of Information Technology, Bangalore.

CS 816- Software Production Engineering – Mini Project



Scientific Calculator with DevOps

Developed By:
Name: Jayprakash Ray
Roll No: MT2021060

Table of Contents

1 Problem Description

1.1 Problem Statement

Create a scientific calculator program with user menu driven operations
● Square root function - √x
● Factorial function - x!
● Natural logarithm (base e) - ln(x)
● Power function – x^b

These are some important links:

1. **GitHub**: https://github.com/jayprakash-ray/Scientific-Calculator-With-DevOps.git
2. **DockerHub**: https://hub.docker.com/repository/docker/jpray/scientificcalcdevops

- **What is DevOps?**
    - DevOps is a group of concept emerging from the collision of two trends. It combines software development with IT ops. It aims to decrease SDLC time and provide continuous delivery and integration functionality to produce high quality software.

- **Why to use DevOps?**
    - It has now become important to include DevOps in software development as it enables faster development of product ,faster bug fixes and easier maintenance as compared to traditional method.

- **Tools Used in the Project**
    - **SCM** : Git and Github
    - **CI** : Jenkins
    - **CD** : Ansible
    - **Containerization : Docker**
    - **Framework** : Angular
    - **Build Tool** :NPM (Node Packet Manager)
    - **Monitoring** : ELK stack

CS-816 Software Production Engineering

Mini Project - Scientific Calculator with DevOps

Submitted By : Jayprakash Ray [MT2021060]

Figure 1.1.1 App Snapshot

2 DevOps Stages

2.1 Initializing Angular Application

Angular is front-end development framework. It is based on Typescript, HTML, CSS.

a. Firstly, we need to **install node.js** from the official website as the build environment for Angular.
b. Then from command prompt install angular/cli using command

   npm install -g @angular/cli

c. Now create Angular Project using following command

   ng new ScientificCalculatorWithDevOps

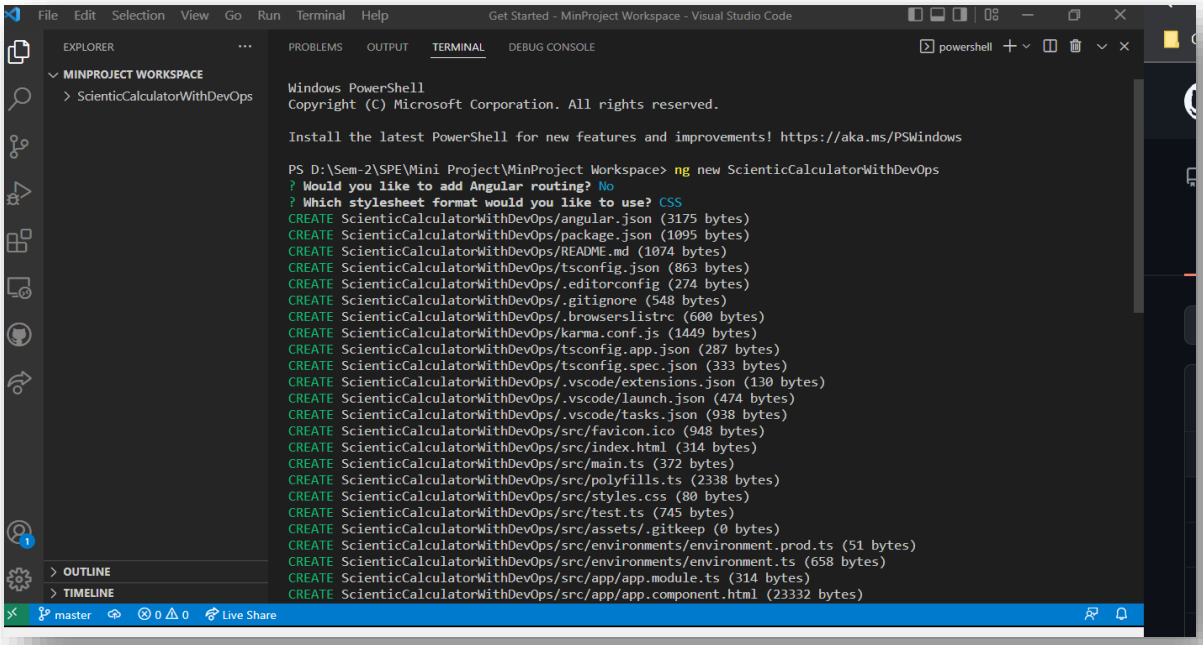


Figure 2.1.1 Initializing Angular Application

2.2 GIT and GITHUB Repository Creation

a. Now let's create a GitHub repository for source code management.

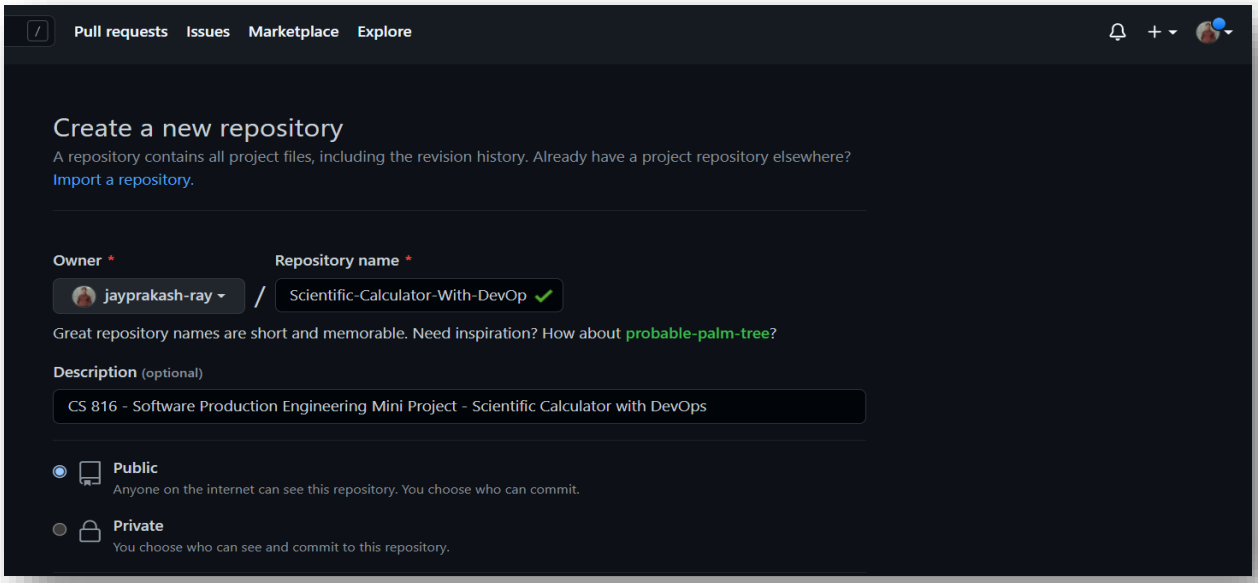


Figure 2.2.1 Creating GITHUB Repository

b. Initialize git in the local folder and push the initial commit to the created repository

Commands:

- **git init** -> to initialize git in the current directory
- **git add .** -> staging the changes
- **git remote add origin** https://github.com/jayprakash-ray/Scientific-Calculator-With-DevOps.git - > Setting the remote for fetch and push
- **git commit -m <Message>** -> to commit changes with message
- **git push origin main** -> Pushing the changes to main branch of remote origin.

```
PS D:\Sem-2\SPE\Mini Project\MinProject Workspace\ScienticCalculatorWithDevOps> git commit -m "Initial Commit"
On branch master
nothing to commit, working tree clean
PS D:\Sem-2\SPE\Mini Project\MinProject Workspace\ScienticCalculatorWithDevOps> gi
                                                                                 git branch -M main
PS D:\Sem-2\SPE\Mini Project\MinProject Workspace\ScienticCalculatorWithDevOps> git remote add origin https://
github.com/jayprakash-ray/Scientific-Calculator-With-DevOps.git
PS D:\Sem-2\SPE\Mini Project\MinProject Workspace\ScienticCalculatorWithDevOps> git push -u origin main
Enumerating objects: 34, done.
Counting objects: 100% (34/34), done.
Delta compression using up to 8 threads
Compressing objects: 100% (32/32), done.
Writing objects: 100% (34/34), 204.26 KiB | 8.88 MiB/s, done.
Total 34 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), done.
To https://github.com/jayprakash-ray/Scientific-Calculator-With-DevOps.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
PS D:\Sem-2\SPE\Mini Project\MinProject Workspace\ScienticCalculatorWithDevOps>
```

Figure 2.2.2 Initializing git and pushing first commit

2.3 Jenkins Pipeline Setup

1. On the Jenkins dashboard Click on New Item
2. Enter a suitable name (i.e *Scientific Calculator with DevOps)*
3. Select Pipeline as the project type

4. Check SCM Polling and Enter “* * * * * “to poll SCM (i.e GitHub) every minute for new commits. Or (5/H * * * * to poll every 5 minutes of new commits)
5. Apply and Save the Project
6. Now Install the Plugins that will be needed furthur by going to Dashboard > Manage Jenkins > Manage Plugins > Install Ansible, Docker, Docker API ,Docker Pipeline ,Git,Github , Git Client and Server and all the required plugins

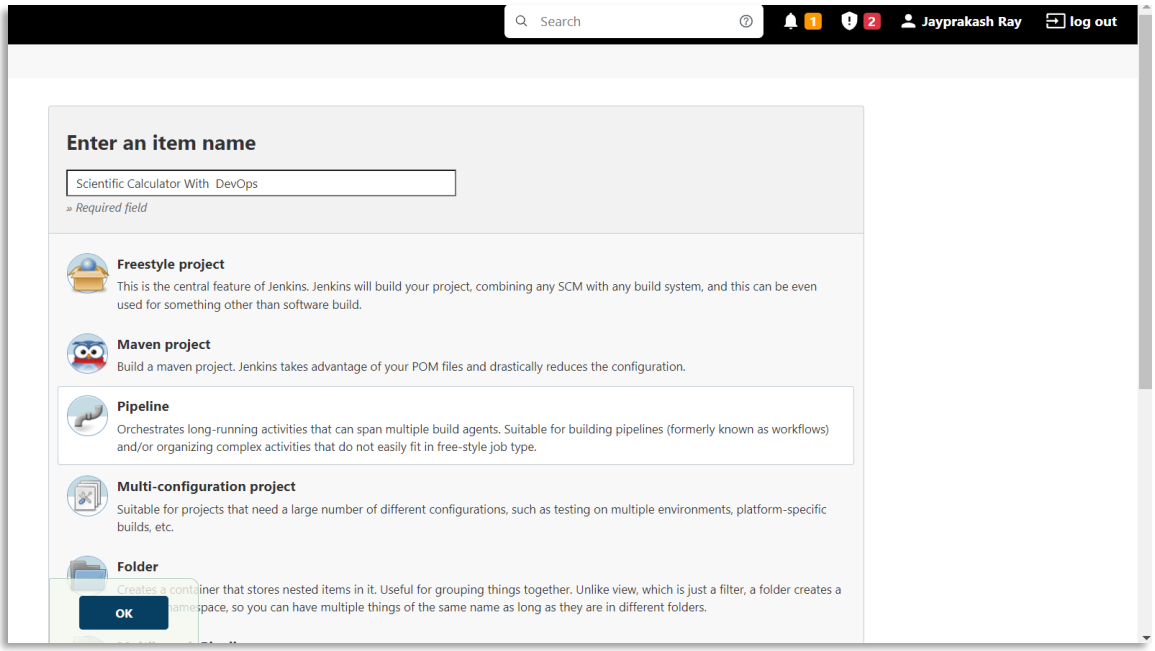


Figure 2.3.1 Creating Pipeline in Jenkins

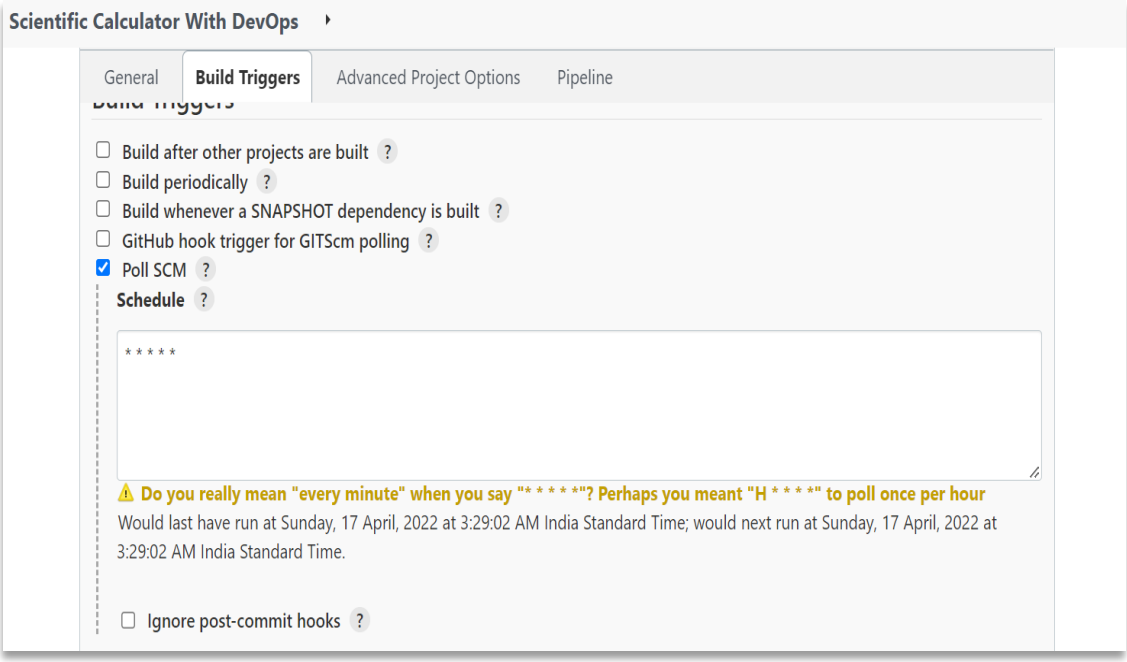


Figure 2.3.2 Setting up Poll SCM

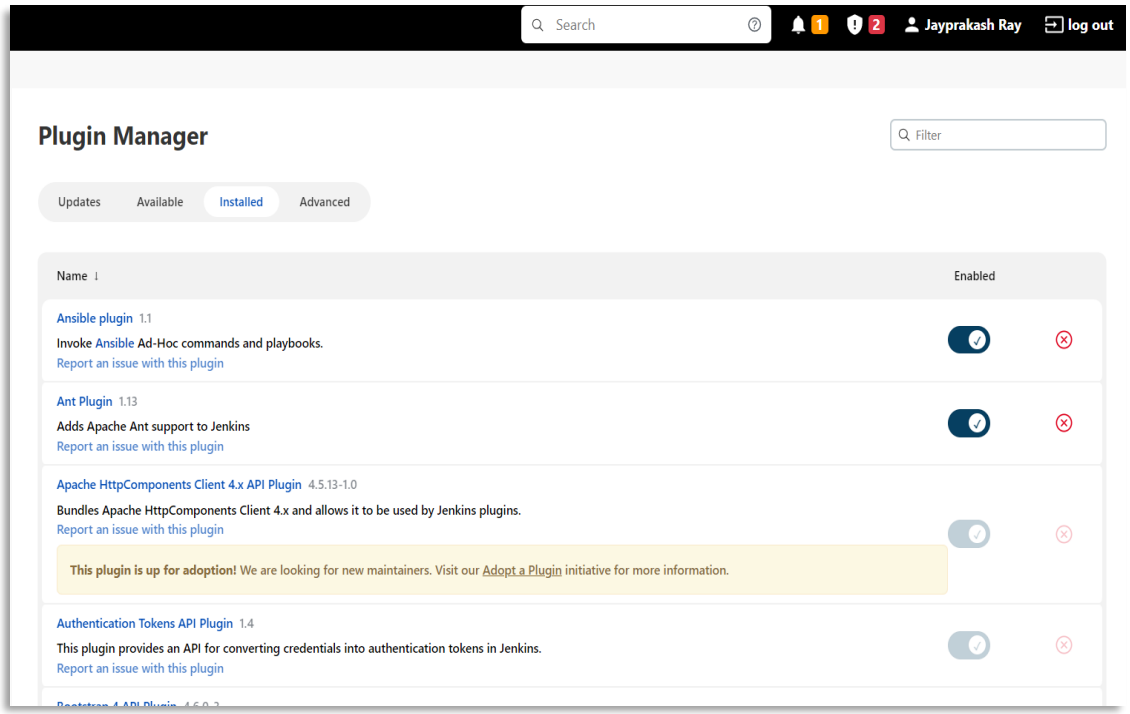


Figure 2.3.3 Installing Jenkins Plugins

2.3.1 Stage 1: GIT PULL

In this stage, Jenkins server pulls the repository provided as a URL in the pipeline script. Firstly, we need to save the GitHub credentials in the Jenkins credentials. Go to Dashboard > Manage Jenkins > Manage Credentials > New Item and add Username/Password of GitHub.

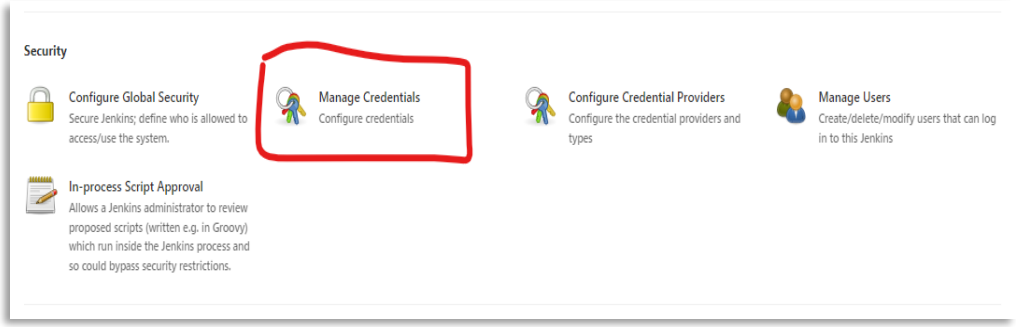


Figure 2.3.1.1 Manage Credentials

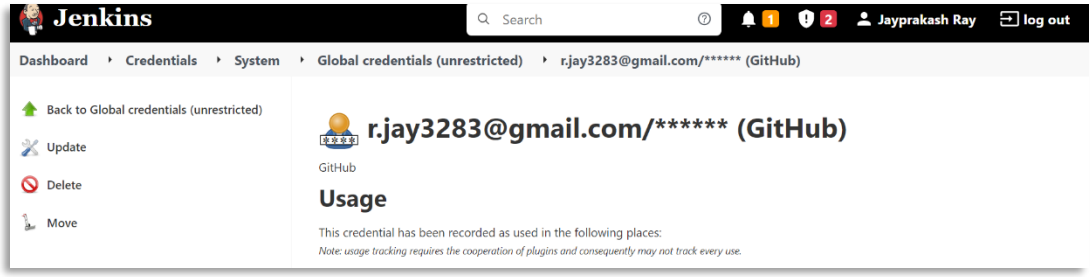


Figure 2.3.1.2 GitHub Credentials Added

Now Open the project go to configure > Pipeline Tab > Write below pipeline script.

Figure 2.3.1.3 Jenkins Pipeline Script for Git Pull

In the URL specify GitHub repository URL.
Now for every commit pipeline will run automatically as it polls GitHub every minutes.

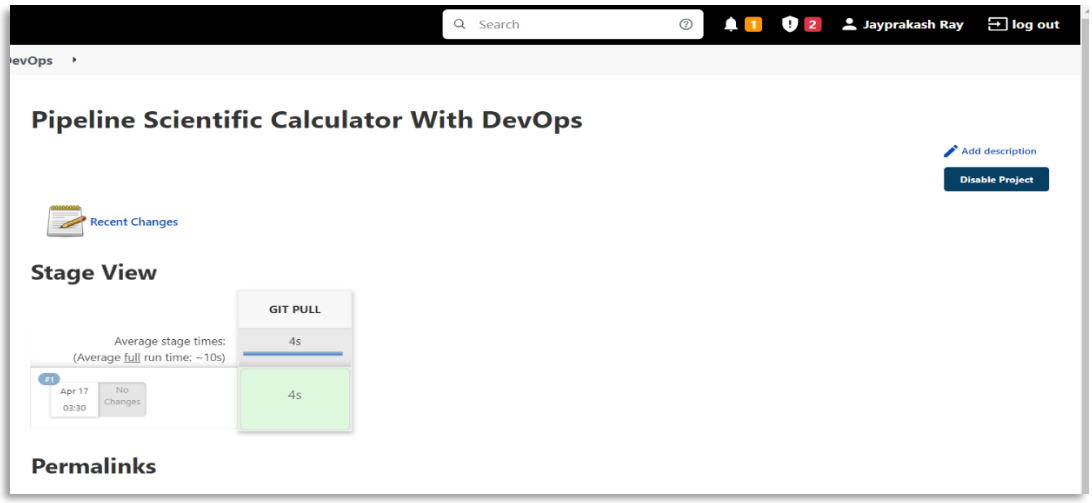


Figure 2.3.1.4 Git Pull Stage View

2.3.2 Stage 2: Dependency Installation

In order to build Angular Application, there are some dependencies which need to be installed. So, this stage mainly deals with installing all those dependencies using Node Package Manager (NPM) by running command:

npm install

```
Script ?
10                  // Get some code from a GitHub repository
11                  git url: 'https://github.com/jayprakash-ray/Scien
12                  credentialsId: 'git_cred'
13              }
14
15          }
16          stage('DEPENDENCY INSTALLATION')
17          {
18              steps
19              {
20                  sh 'npm install'
21                  echo "Modules Installed"
22              }
23          }
24          }
25      }
26
```

Figure 2.3.2.1 Dependency Pipeline Script

```
 > /usr/bin/git rev-list --no-walk 1f1dadba87fe8facddfd68e25a56750afccedd17 # timeout=10
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (DEPENDENCY INSTALLATION)
[Pipeline] sh
+ npm install
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@angular-devkit/architect@0.1302.6',
npm WARN EBADENGINE   required: {
npm WARN EBADENGINE     node: '^12.20.0 || ^14.15.0 || >=16.10.0',
npm WARN EBADENGINE     npm: '^6.11.0 || ^7.5.6 || >=8.0.0',
npm WARN EBADENGINE     yarn: '>= 1.13.0'
npm WARN EBADENGINE   },
npm WARN EBADENGINE   current: { node: 'v12.21.0', npm: '7.5.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@angular-devkit/build-angular@13.2.6',
npm WARN EBADENGINE   required: {
npm WARN EBADENGINE     node: '^12.20.0 || ^14.15.0 || >=16.10.0',
npm WARN EBADENGINE     npm: '^6.11.0 || ^7.5.6 || >=8.0.0',
npm WARN EBADENGINE     yarn: '>= 1.13.0'
npm WARN EBADENGINE   },
npm WARN EBADENGINE   current: { node: 'v12.21.0', npm: '7.5.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@angular-devkit/build-webpack@0.1302.6',
npm WARN EBADENGINE   required: {
npm WARN EBADENGINE     node: '^12.20.0 || ^14.15.0 || >=16.10.0',
npm WARN EBADENGINE     npm: '^6.11.0 || ^7.5.6 || >=8.0.0',
npm WARN EBADENGINE     yarn: '>= 1.13.0'
npm WARN EBADENGINE   },
npm WARN EBADENGINE   current: { node: 'v12.21.0', npm: '7.5.2' }
npm WARN EBADENGINE }
npm WARN EBADENGINE Unsupported engine {
npm WARN EBADENGINE   package: '@angular-devkit/core@13.2.6',
npm WARN EBADENGINE   required: {
npm WARN EBADENGINE     node: '^12.20.0 || ^14.15.0 || >=16.10.0',
npm WARN EBADENGINE     npm: '^6.11.0 || ^7.5.6 || >=8.0.0',
npm WARN EBADENGINE     yarn: '>= 1.13.0'
npm WARN EBADENGINE   },
npm WARN EBADENGINE   current: { node: 'v12.21.0', npm: '7.5.2' }
```

Figure 2.3.2.2 Dependency Installation console output

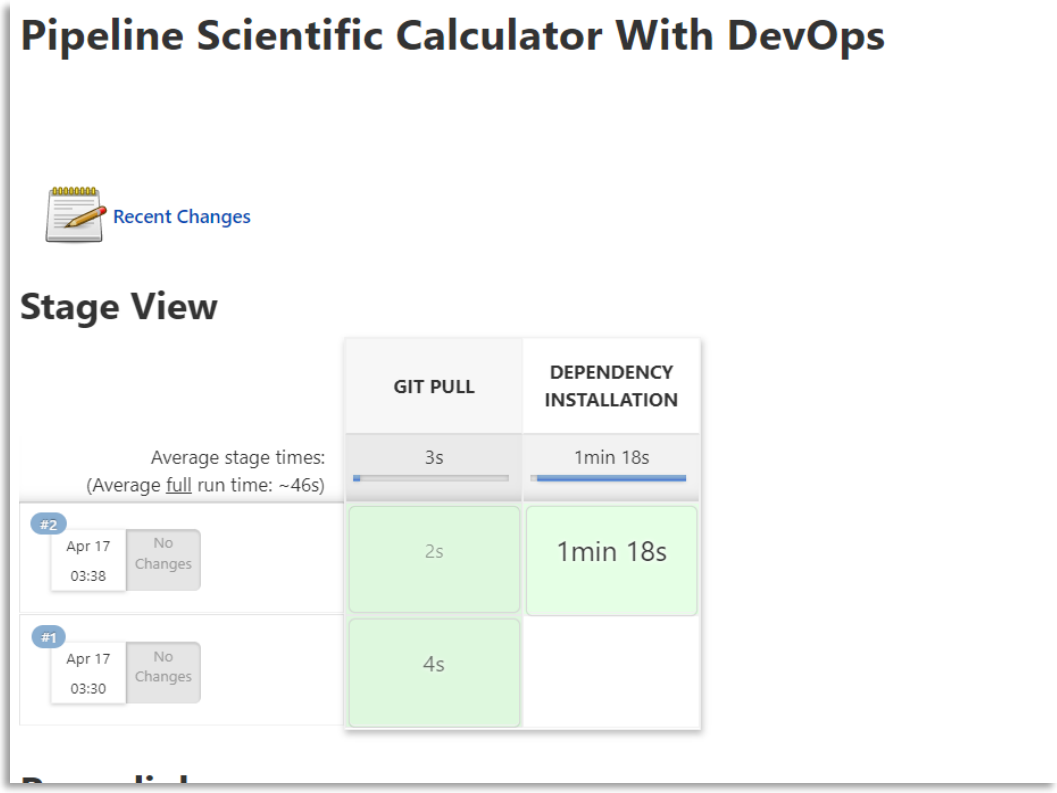


Figure 2.3.2.3 Dependency Installation Stage View

2.3.3 Stage 3: Build

This stage mainly deals with building the application by analyzing and optimizing the files. The command used in this stage for build angular application is as below:

npm run build

```
}
stage('BUILD')
{
    steps
    {
      sh 'npm run build'
      echo "Build completed"
    }
}
}
```

Figure 2.3.3.1 Build Script for Jenkins Pipeline

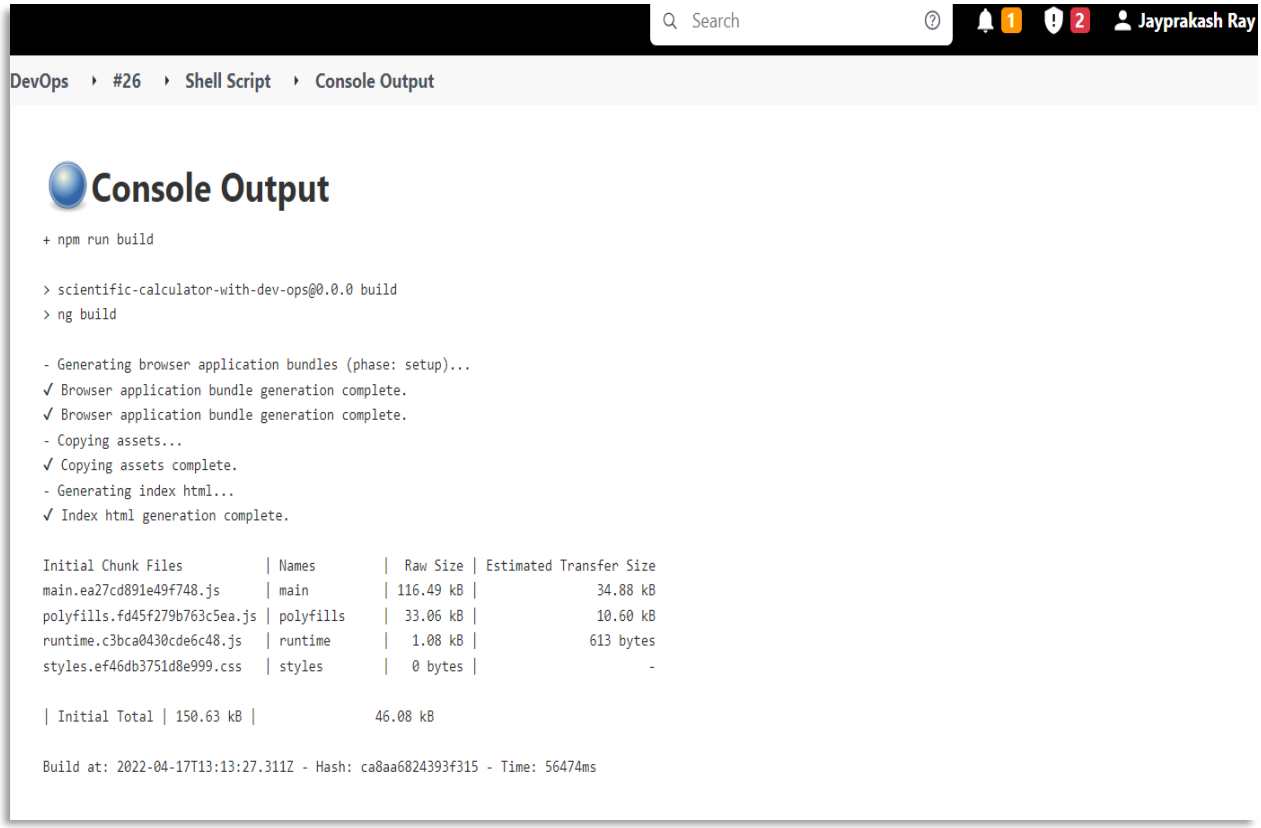


Figure 2.3.3.2 Build Stage Console Output

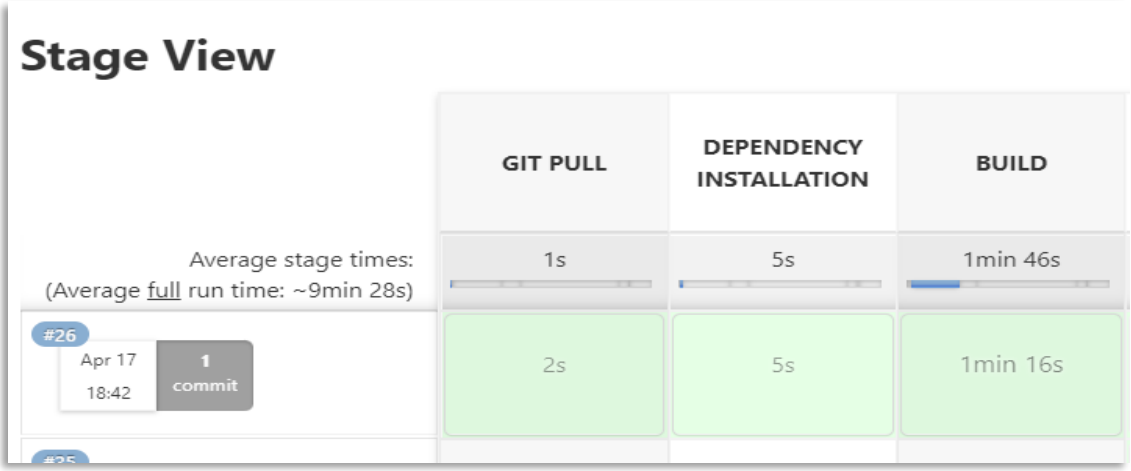


Figure 2.3.3.3 Build Stage View

2.3.4 Stage 4: Running Unit Test Cases

This stage tests the execution of build application by executing tests using Jasmine Framework and the Karma test runner.

The command executed in this stage is as follows:

ng test --sourceMap=false --browsers=ChromeHeadless --watch=false --progress=false

This command uses Karma Test runner to execute tests. *ChromeHeadless* indicates to run the tests without Chrome GUI.

```
stage('RUNING UNIT TEST')
{
    steps
    {
        sh 'ng test --sourceMap=false --browsers=ChromeHeadless --watch=false --progress=false'
        echo "Test completed"
    }
}
```

Figure 2.3.4.1 Unit Test Pipeline Script

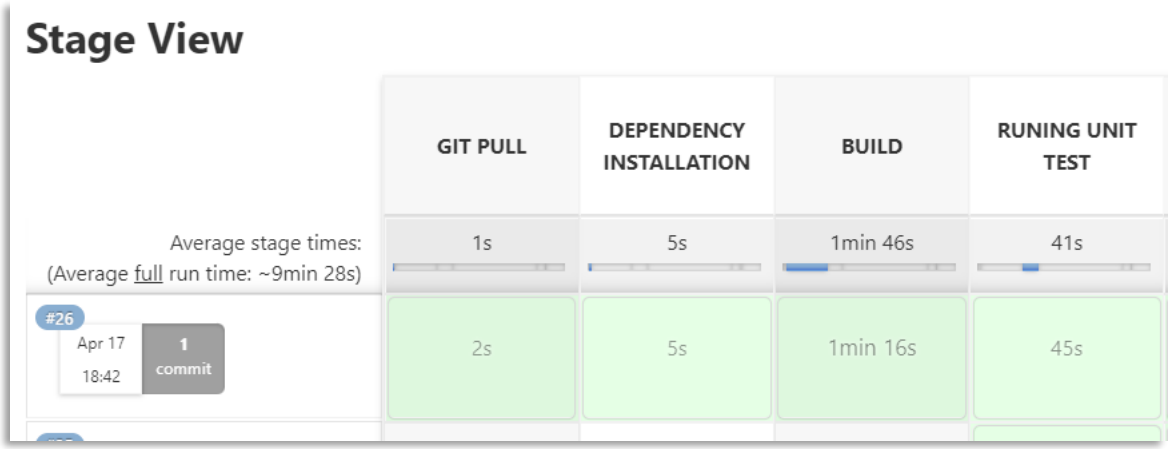


Figure 2.3.4.2 Unit Test Stage View

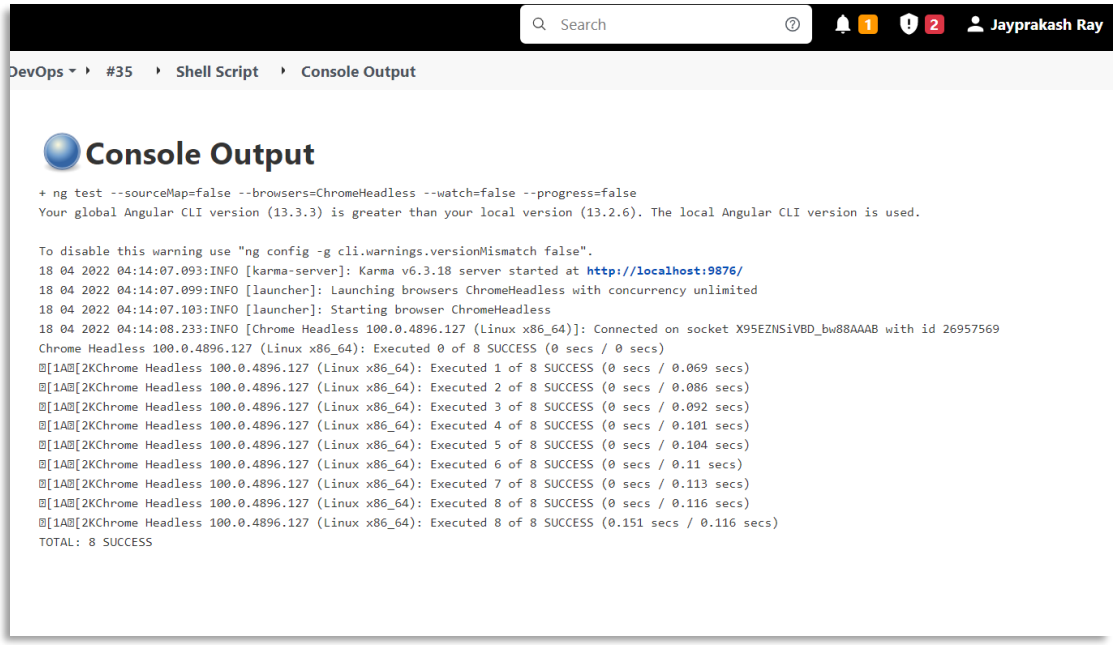


Figure 2.3.4.3 Unit Test Console Output

2.3.5 Stage 5: Docker Image Build

In this stage docker image is build which can be furthur executed in docker container on deployment server.

docker build . -t jpray/scientificcalcdevops:latest

It takes build image name as a argument and uses **Dockerfile** and **nginx.conf** file for execution.

```
stage('DOCKER IMAGE BUILD')
{
    steps
     {
          sh 'docker build . -t jpray/scientificcalcdevops:latest'
     }
}
```

Figure 2.3.5.1 DOCKER Image Build Pipeline Script

jayprakash-ray Added DockerFile

1 contributor

11 lines (11 sloc) | 335 Bytes

```
### STAGE 1: Build ###
FROM node:16.10-alpine AS build
WORKDIR /usr/src/app
COPY package.json package-lock.json ./
RUN npm install
COPY . .
RUN npm run build
### STAGE 2: Run ###
FROM nginx:1.17.1-alpine
COPY nginx.conf /etc/nginx/nginx.conf
COPY --from=build /usr/src/app/dist/scientific-calculator-with-dev-ops /usr/share/nginx/html
```

Figure 2.3.5.2 Docker file

jayprakash-ray Added:nginx.conf file

1 contributor

13 lines (13 sloc) | 261 Bytes

```
events{}
http {
    include /etc/nginx/mime.types;
    server {
        listen 80;
        server_name localhost;
        root /usr/share/nginx/html;
        index index.html;
        location / {
            try_files $uri $uri/ /index.html;
        }
    }
}
```

Figure 2.3.5.3 NGINX.CONF file

Search 1 2 Jayprakash Ray

ith DevOps › #26 › Shell Script › Console Output

Console Output

```
+ docker build . -t jpray/scientificcalcdevops:latest
Sending build context to Docker daemon  520.5MB

Step 1/9 : FROM node:16.10-alpine AS build
 ---> a3c0a72e086a
Step 2/9 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 5c6951b7a0eb
Step 3/9 : COPY package.json package-lock.json ./
 ---> Using cache
 ---> cb843a636c74
Step 4/9 : RUN npm install
 ---> Using cache
 ---> 1fd7c683c027
Step 5/9 : COPY . .
 ---> 0785614d91d6
Step 6/9 : RUN npm run build
 ---> Running in 946db4d9cae4
```

```
> scientific-calculator-with-dev-ops@0.0.0 build
> ng build

[91m- Generating browser application bundles (phase: setup)...
[0m[91m Browser application bundle generation complete.
[0m[91m Browser application bundle generation complete.
[0m[91m- Copying assets...
[0m[91m Copying assets complete.
[0m[91m- Generating index html...
[0m[91m Index html generation complete.
[0m
Initial Chunk Files           | Names         |  Raw Size | Estimated Transfer Size
main.ea27cd891e49f748.js      | main          | 116.49 kB |                34.88 kB
polyfills.fd45f279b763c5ea.js | polyfills     |  33.06 kB |                10.60 kB
runtime.c3bca0430cde6c48.js   | runtime       |   1.08 kB |               613 bytes
styles.ef46db3751d8e999.css   | styles        |   0 bytes |                      -

| Initial Total | 150.63 kB |                46.08 kB

Build at: 2022-04-17T13:19:49.575Z - Hash: ca8aa6824393f315 - Time: 71622ms
Removing intermediate container 946db4d9cae4
 ---> e51070ab2205
Step 7/9 : FROM nginx:1.17.1-alpine
 ---> ea1193fd3dde
Step 8/9 : COPY nginx.conf /etc/nginx/nginx.conf
 ---> Using cache
 ---> 6a1ce720646a
Step 9/9 : COPY --from=build /usr/src/app/dist/scientific-calculator-with-dev-ops /usr/share/nginx/html
 ---> f3fc64349dfa
Successfully built f3fc64349dfa
Successfully tagged jpray/scientificcalcdevops:latest
```

Figure 2.3.5.4 Build Docker Image Console Output

The Docker Image of the **name jpray/scientificcalcdevops:latest** will be created in the Jenkins workspace

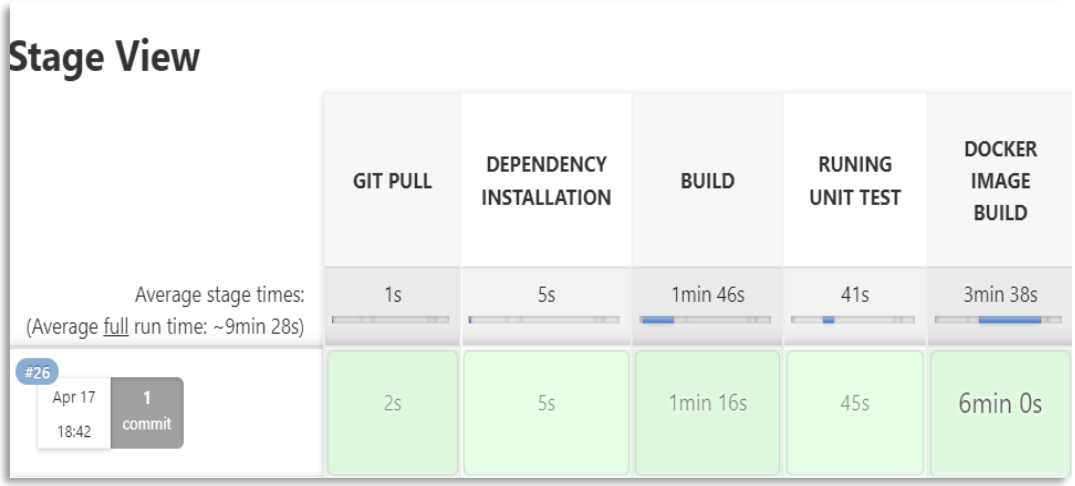


Figure 2.3.5.5 Build Docker Image Stage View

2.3.6 Stage 6: Push Image to DockerHub

This stage pushes the build image in the previous stage to Docker hub where it can be stored and fetched later for execution.
Configure Jenkins and add docker Hub credentials through manage credentials.

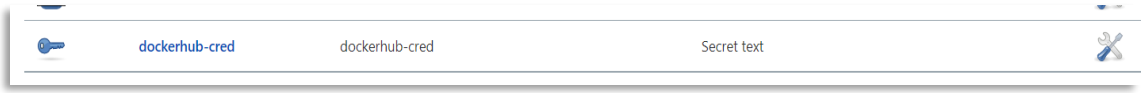


Figure 2.3.6.1 Docker Hub Credentials

The command used for pushing docker image to dockerhub is as below :

```
docker push  jpray/scientificcalcdevops:latest
```

```
stage('PUSH IMAGE TO DOCKERHUB')
{
    steps{
            withCredentials([string(credentialsId: 'dockerhub-cred', variable: 'dockerHubPassword')])
            {
                sh "docker login -u jpray -p ${dockerHubPassword}"
            }
            sh 'docker push jpray/scientificcalcdevops:latest'
        }
}
```

Figure 2.3.6.2 DockerHub Push Pipeline Script

```
[Pipeline] { (PUSH IMAGE TO DOCKERHUB)
[Pipeline] withCredentials
Masking supported pattern matches of $dockerHubPassword
[Pipeline] {
[Pipeline] sh
Warning: A secret was passed to "sh" using Groovy String interpolation, which is insecure.
                 Affected argument(s) used the following variable(s): [dockerHubPassword]
                 See https://jenkins.io/redirect/groovy-string-interpolation for details.
+ docker login -u jpray -p ****
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /var/lib/jenkins/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[Pipeline] }
[Pipeline] // withCredentials
[Pipeline] sh
+ docker push jpray/scientificcalcdevops:latest
The push refers to repository [docker.io/jpray/scientificcalcdevops]
cb8ae30b04d9: Preparing
ac1be46b0018: Preparing
fbe0fc9bcf95: Preparing
f1b5933fe4b5: Preparing
fbe0fc9bcf95: Layer already exists
f1b5933fe4b5: Layer already exists
ac1be46b0018: Layer already exists
cb8ae30b04d9: Pushed
latest: digest: sha256:bb4de84f67a5be851d0c3590e4e5299fc7d72dafe65169f8910c287742306c7e size: 1155
```

Figure 2.3.6.3 DockerHub Push Console Output

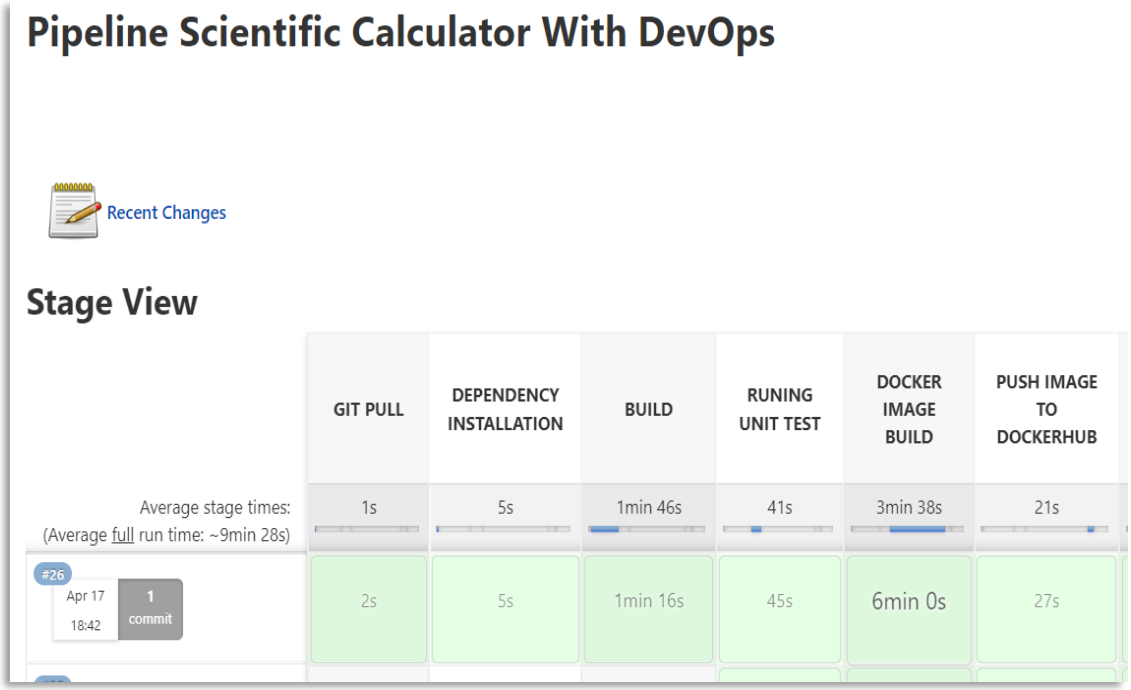


Figure 2.3.6.4 DockerHub Push Stage View

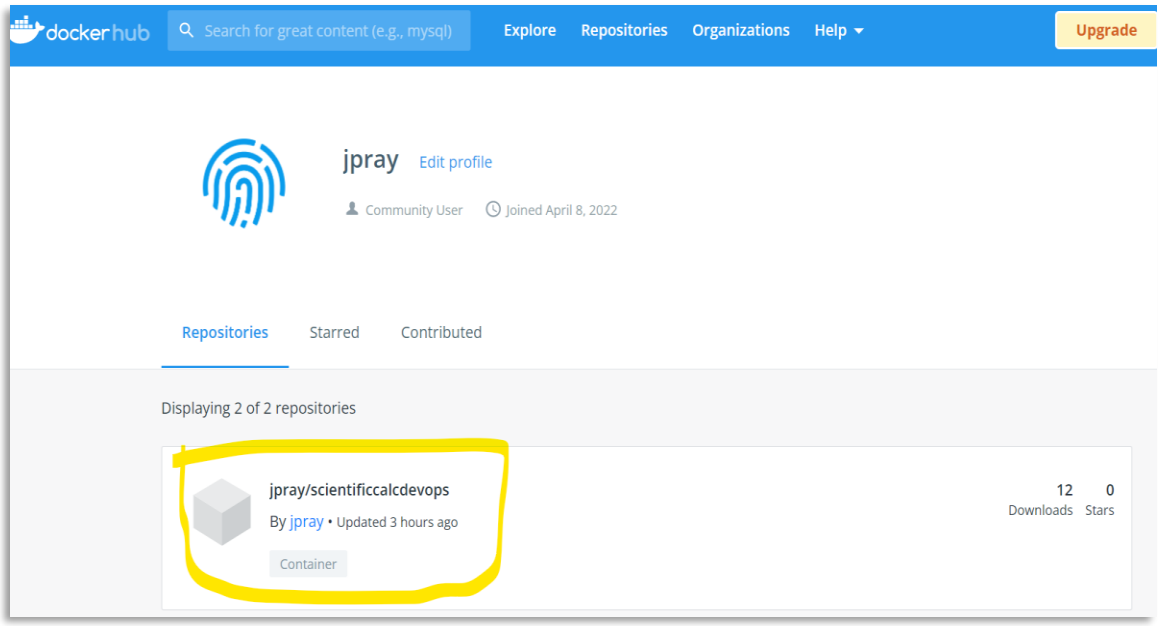


Figure 2.3.6.5 DockerHub

2.3.7 Stage 7: Removing Previous Builds

This stage removes all the previous build to avoid duplicate builds using below command:

docker rmi jpray/scientificcalcdevops:latest

```
stage('REMOVING PREVIOUS BUILD IMAGES')
{
    steps
    {
        sh "docker rmi $registry:latest"
        echo "Previous Build Removed!"
    }
}
```

Figure 2.3.7.1 Remove Previous Builds Pipeline Script

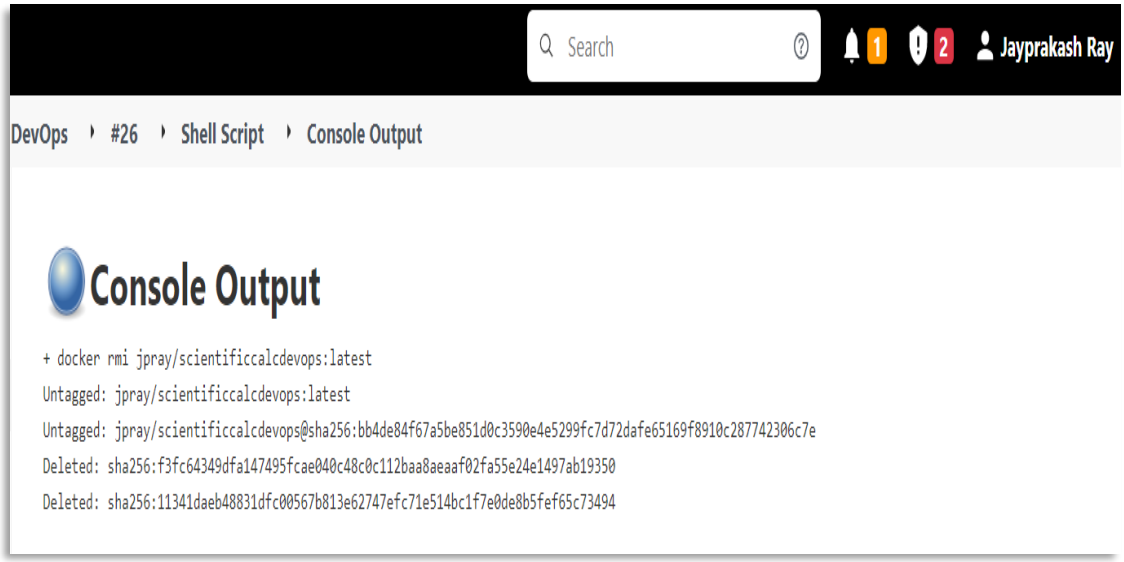


Figure 2.3.7.2 Remove Previous Builds Console Output

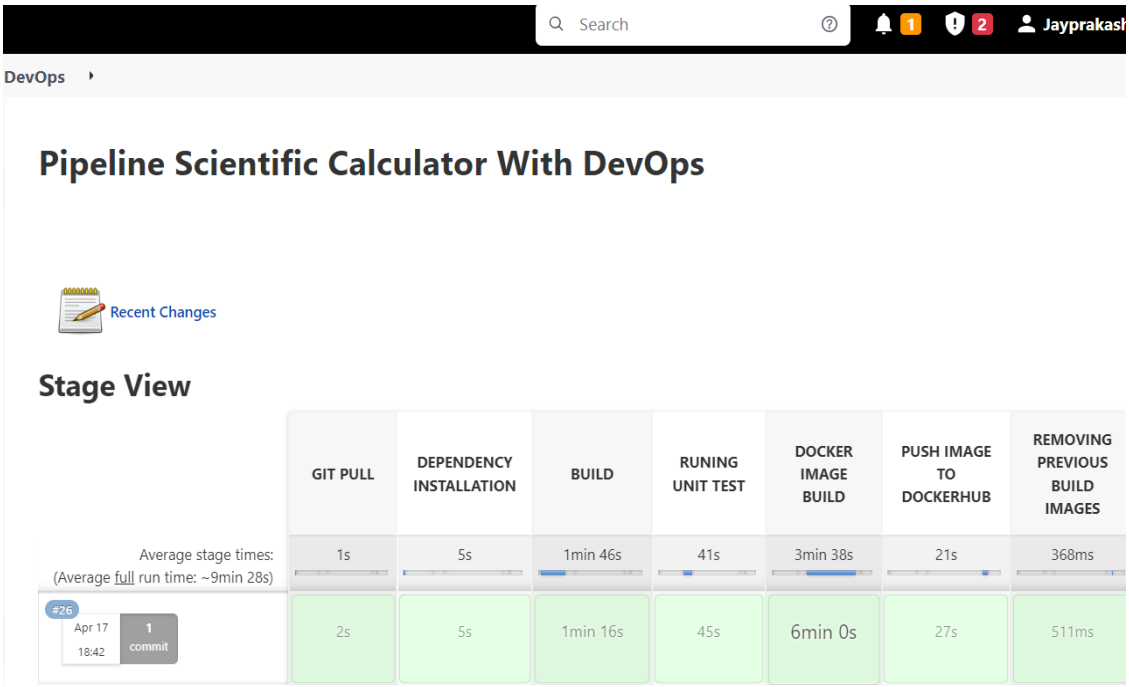


Figure 2.3.7.3 Remove Previous Builds Stage View

2.3.8 Stage 8: Ansible Deployment

In this stage, the docker image pushed to dockerhub is pulled and deployed on the specified server in the **inventory** file using steps/commands specified in **deploy_container.yml** file.

jayprakash-ray Added : deploy_container.yml and inventory file for ansible deployment

1 contributor

29 lines (24 sloc) | 768 Bytes

```yaml
---
- name: Pull docker image of ToDoList
  hosts: all
  vars:
    ansible_python_interpreter: /usr/bin/python3
  tasks:

    - name: Stop Docker Container
      shell: docker stop scientificcalcdevops

    - name: Remove Docker Container
      shell: docker rm -f scientificcalcdevops

    - name: Remove Docker Image
      shell: docker image rm -f jpray/scientificcalcdevops:latest

    - name: Pull calculator devops image
      docker_image:
        name: jpray/scientificcalcdevops:latest
        source: pull

    - name: deploy calculator docker container
      docker_container:
        image: jpray/scientificcalcdevops:latest
        name: scientificcalcdevops
        state: started
        ports:
          - "8090:80"
...
```

Figure 2.3.8.1 deploy_container.yml File

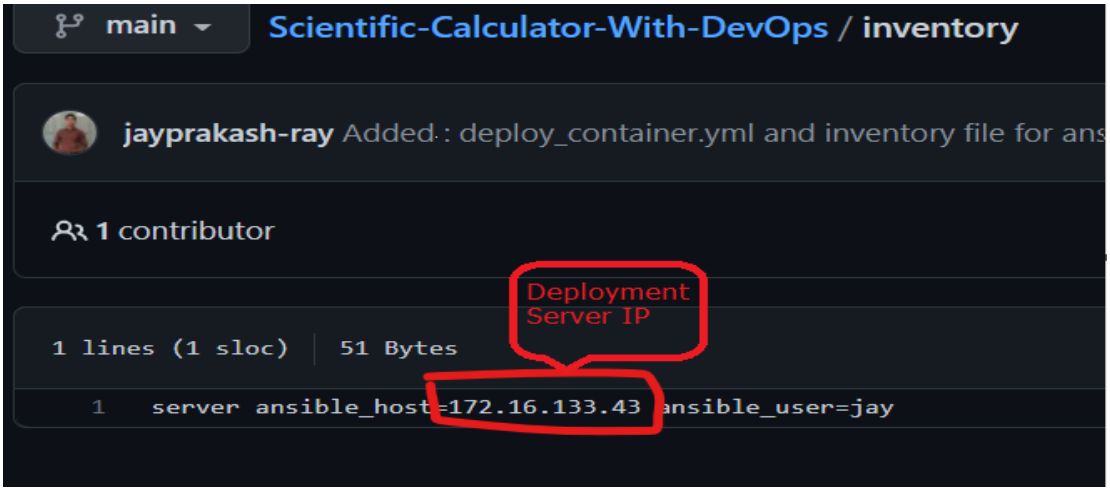


Figure 2.3.8.2 Inventory File

Now we will test remote SSH connection from host to deployment server

i. Execute Jenkins in super user mode using command:

sudo su - jenkins

ii. Generate RSA key pair

ssh-keygen -t rsa

```
jp@jp-VirtualBox:~$ sudo su - jenkins
[sudo] password for jp:
jenkins@jp-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
/var/lib/jenkins/.ssh/id_rsa already exists.
Overwrite (y/n)?
jenkins@jp-VirtualBox:~$
jenkins@jp-VirtualBox:~$ ssh-keygen -t rsa
Generating public/private rsa key pair.
Enter file in which to save the key (/var/lib/jenkins/.ssh/id_rsa):
/var/lib/jenkins/.ssh/id_rsa already exists.
Overwrite (y/n)? y
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /var/lib/jenkins/.ssh/id_rsa
Your public key has been saved in /var/lib/jenkins/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:1wO6GhdZTObP7tzONfW+bXMjdAQNmnHKQg6+wJD3fmk jenkins@jp-VirtualBox
The key's randomart image is:
+---[RSA 3072]----+
|    ..  . .o. oo  |
|    .o.. +=. *. . |
|     .o.. o==  .  |
|       ...+.=   . |
|       ..S o = . .|
|        . E . o .o|
|       . =   o ..o|
|        +   o + ==|
|       .     o.=oB|
+----[SHA256]-----+
```

iii. Copy key to deployment server

ssh-copy-id jay@172.16.133.43

```
jenkins@jp-VirtualBox:~$ ssh-copy-id jay@172.16.133.43
/usr/bin/ssh-copy-id: INFO: Source of key(s) to be installed: "/var/lib/jenkins/.ssh/id_rsa.pub"
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
jay@172.16.133.43's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'jay@172.16.133.43'"
and check to make sure that only the key(s) you wanted were added.
```

iv. Now try remote login using command

ssh jay@172.16.133.43

```
jenkins@jp-VirtualBox:~$ ssh jay@172.16.133.43
Welcome to Ubuntu 18.04.6 LTS (GNU/Linux 4.15.0-175-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sun Apr 17 17:53:55 UTC 2022

  System load:  0.09              Processes:             92
  Usage of /:   27.5% of 19.56GB  Users logged in:       1
  Memory usage: 10%               IP address for enp0s3: 172.16.133.43
  Swap usage:   0%                IP address for docker0: 172.17.0.1

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation

23 updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Failed to connect to https://changelogs.ubuntu.com/meta-release-lts. Check your Internet connection or proxy settings


Last login: Sun Apr 17 17:51:59 2022
jay@ubuntu_server: $
```

Figure 2.3.8.3 Remote Login to Deployment Server

```
stage('ANSIBLE DEPLOYMENT')
{
 steps
 {
    ansiblePlaybook disableHostKeyChecking: true,  installation: 'ansible', inventory: 'inventory', playbook: 'deploy_container.yml'
 }

}
```

Figure 2.3.8.4 Ansible Deployment Pipeline Script

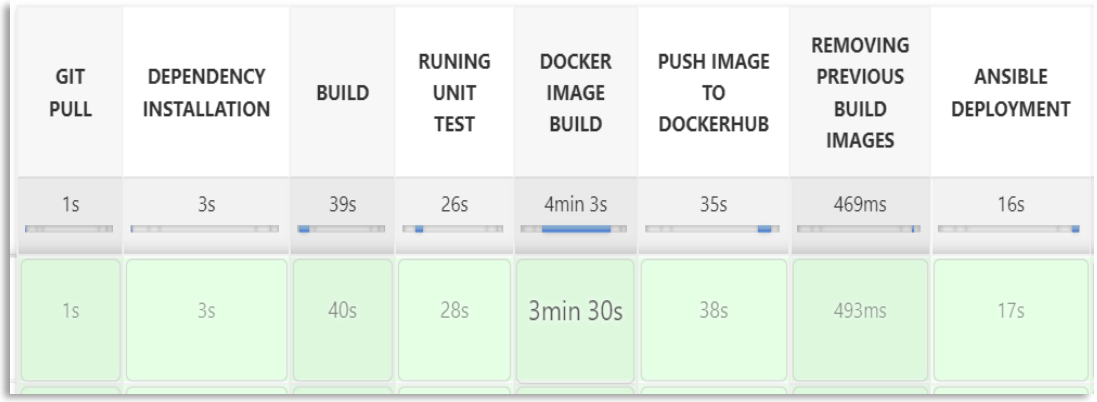
Search 1 2 Jayprakash Ray

vOps › #34 › Invoke an ansible playbook › Console Output

Console Output

```
[Scientific Calculator With DevOps] $ /usr/bin/ansible-playbook deploy_container.yml -i inventory

PLAY [Pull docker image of ToDoList] *******************************************

TASK [Gathering Facts] *********************************************************
ok: [server]

TASK [Stop Docker Container] ***************************************************
changed: [server]

TASK [Remove Docker Container] *************************************************
changed: [server]

TASK [Remove Docker Image] *****************************************************
changed: [server]

TASK [Pull calculator devops image] ********************************************
changed: [server]

TASK [deploy calculator docker container] **************************************
[DEPRECATION WARNING]: The container_default_behavior option will change its
default value from "compatibility" to "no_defaults" in community.general 3.0.0.
 To remove this warning, please specify an explicit value for it now. This
feature will be removed from community.general in version 3.0.0. Deprecation
warnings can be disabled by setting deprecation_warnings=False in ansible.cfg.
changed: [server]

PLAY RECAP *********************************************************************
server                     : ok=6    changed=5    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Figure 2.3.8.5 Ansible Deployment Console Output



GIT PULL	DEPENDENCY INSTALLATION	BUILD	RUNING UNIT TEST	DOCKER IMAGE BUILD	PUSH IMAGE TO DOCKERHUB	REMOVING PREVIOUS BUILD IMAGES	ANSIBLE DEPLOYMENT
1s	3s	39s	26s	4min 3s	35s	469ms	16s
1s	3s	40s	28s	3min 30s	38s	493ms	17s

Figure 2.3.8.6 Ansible Deployment Stage View

After Deploying image on the server. Let's check for the images using **docker image ls** command:

```
jay@ubuntu_server:~$ docker image ls
REPOSITORY                    TAG       IMAGE ID       CREATED          SIZE
jpray/scientificcalcdevops    latest    d972624b7f9c   15 minutes ago   20.9MB
```

Also Checking for created container using **docker ps -a**

```
ay@ubuntu_server:~$ docker ps -a
ONTAINER ID   IMAGE                                  COMMAND                  CREATED          STATUS
                PORTS                     NAMES
d5503969e8c   jpray/scientificcalcdevops:latest      "nginx -g 'daemon of…"   18 minutes ago   Up 18 m
nutes           0.0.0.0:8090->80/tcp      scientificcalcdevops
5fb956281e4   hello-world                            "/hello"                 7 days ago       Exited
0) 7 days ago                           gifted_mestorf
ay@ubuntu_server:~$ _
```

Hence deployment is successful!

3 Final Jenkins Pipeline

GIT PULL	DEPENDENCY INSTALLATION	BUILD	RUNING UNIT TEST	DOCKER IMAGE BUILD	PUSH IMAGE TO DOCKERHUB	REMOVING PREVIOUS BUILD IMAGES	ANSIBLE DEPLOYMENT	Declarative: Post Actions
1s	3s	37s	24s	3min 52s	31s	433ms	15s	395ms
1s	3s	33s	20s	3min 29s	21s	363ms	13s	364ms

Fig.3.1 Final Jenkins Pipeline Stage View

```
pipeline
{
    environment
    {
        registry = "jpray/scientificcalcdevops"
        registryCredential = 'docker-cred'
        dockerImage = ''
    }
    agent any
    stages
    {
        stage('GIT PULL')
        {
          steps
          {
            // Get some code from a GitHub repository
            git url: 'https://github.com/jayprakash-ray/Scientific-Calculator-With-DevOps.git', branch: 'main',
            credentialsId: 'git_cred'
          }

        }
      stage('DEPENDENCY INSTALLATION')
      {
          steps
          {
                sh 'npm install'
                echo "Modules Installed"
          }
    }
```

```
stage('BUILD')
{
    steps
    {
      sh 'npm run build'
      echo "Build completed"
    }
}
stage('RUNING UNIT TEST')
{
    steps
   {
      sh 'ng test --sourceMap=false --browsers=ChromeHeadless --watch=false --progress=false'
      echo "Test completed"
    }
}


stage('DOCKER IMAGE BUILD')
{
    steps
    {
        sh 'docker build . -t jpray/scientificcalcdevops:latest'
    }
}
stage('PUSH IMAGE TO DOCKERHUB')
{
    steps{
        withCredentials([string(credentialsId: 'dockerhub-cred', variable: 'dockerHubPassword')])
        {
            sh "docker login -u jpray -p ${dockerHubPassword}"
        }
        sh 'docker push jpray/scientificcalcdevops:latest'
      }
}
stage('REMOVING PREVIOUS BUILD IMAGES')
{
    steps
    {
        sh "docker rmi $registry:latest"
        echo "Previous Build Removed!"
    }
}
```

```
stage('ANSIBLE DEPLOYMENT')
{
  steps
  {
    ansiblePlaybook disableHostKeyChecking: true,  installation: 'ansible', inventory: 'inventory', playbook: 'deploy_container.yml'
  }

}

}
post
{
    always
    {
      sh 'docker logout'
    }
}

}
```

Fig.3.2 Final Jenkins Pipeline Script

4 Monitoring

The ELK stack is an acronym used to describe a collection of three open source projects – Elasticsearch, Logstash, and Kibana. Elasticsearch is a full text search and analytics engine. Logstash is a log aggregator that collects and processes data from multiple sources, converts, and ships it to various destinations, such as Elasticsearch. And finally, Kibana provides a user interface, allowing users to visualize, query, and analyse their data via graphs and charts.

Log File :

ScientificCalculatorWithDevOps > ScientificCalc.log

```
ngx-logger.mjs:573 2022-04-18T09:05:44.348Z INFO [main.js:29:21] factorial() function called
ngx-logger.mjs:573 2022-04-18T09:07:27.789Z INFO [main.js:29:21] factorial() function called
ngx-logger.mjs:573 2022-04-18T09:07:31.512Z INFO [main.js:29:21] factorial() function called
ngx-logger.mjs:573 2022-04-18T09:07:34.023Z INFO [main.js:29:21] power() function called
ngx-logger.mjs:573 2022-04-18T09:07:35.941Z INFO [main.js:29:21] Calculate
ngx-logger.mjs:573 2022-04-18T09:07:40.493Z INFO [main.js:29:21] Cancel
ngx-logger.mjs:573 2022-04-18T09:07:43.663Z INFO [main.js:29:21] sqroot() function called
ngx-logger.mjs:573 2022-04-18T09:07:49.445Z INFO [main.js:29:21] sqroot() function called
ngx-logger.mjs:573 2022-04-18T09:07:54.799Z INFO [main.js:29:21] log() function called
ngx-logger.mjs:573 2022-04-18T09:08:04.276Z INFO [main.js:29:21] log() function called
ngx-logger.mjs:573 2022-04-18T09:08:07.151Z INFO [main.js:29:21] sqroot() function called
ngx-logger.mjs:573 2022-04-18T09:08:09.207Z INFO [main.js:29:21] sqroot() function called
ngx-logger.mjs:573 2022-04-18T09:08:11.795Z INFO [main.js:29:21] factorial() function called
ngx-logger.mjs:573 2022-04-18T09:08:13.670Z INFO [main.js:29:21] Cancel
ngx-logger.mjs:573 2022-04-18T09:08:20.327Z INFO [main.js:29:21] sqroot() function called
ngx-logger.mjs:573 2022-04-18T09:08:24.422Z INFO [main.js:29:21] power() function called
ngx-logger.mjs:573 2022-04-18T09:08:27.057Z INFO [main.js:29:21] Calculate
```

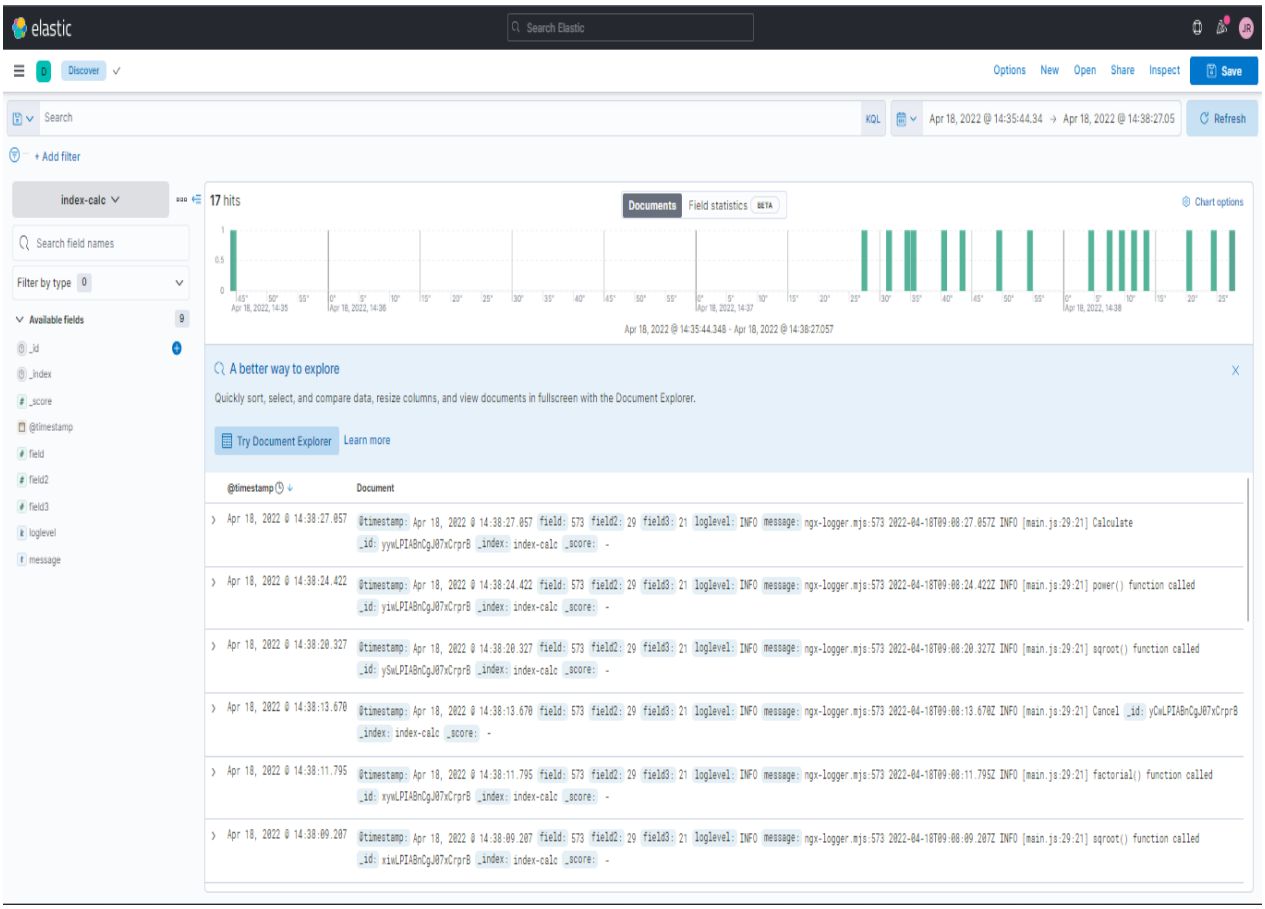


Figure 4.2 ELK Visualization of log file

5 Issue Faced

Missing $Display error while running test cases:

```
+ npm run test

> scientic-calculator-with-dev-ops@0.0.0 test
> ng test

- Generating browser application bundles (phase: setup)...
✓ Browser application bundle generation complete.
17 04 2022 12:48:09.972:INFO [karma-server]: Karma v6.3.18 server started at http://localhost:9876/
17 04 2022 12:48:09.979:INFO [launcher]: Launching browsers Chrome with concurrency unlimited
17 04 2022 12:48:09.994:INFO [launcher]: Starting browser Chrome
17 04 2022 12:48:10.211:ERROR [launcher]: Cannot start Chrome
        [13264:13264:0417/124810.180098:ERROR:ozone_platform_x11.cc(247)] Missing X server or $DISPLAY
[13264:13264:0417/124810.181048:ERROR:env.cc(225)] The platform failed to initialize.  Exiting.

17 04 2022 12:48:10.211:ERROR [launcher]: Chrome stdout:
17 04 2022 12:48:10.211:ERROR [launcher]: Chrome stderr: [13264:13264:0417/124810.180098:ERROR:ozone_platform_x11.cc(247)] Missing X server or $DISPLAY
[13264:13264:0417/124810.181048:ERROR:env.cc(225)] The platform failed to initialize.  Exiting.

17 04 2022 12:48:10.217:INFO [launcher]: Trying to start Chrome again (1/2).
17 04 2022 12:48:10.437:ERROR [launcher]: Cannot start Chrome
        [13297:13297:0417/124810.410313:ERROR:ozone_platform_x11.cc(247)] Missing X server or $DISPLAY
[13297:13297:0417/124810.411000:ERROR:env.cc(225)] The platform failed to initialize.  Exiting.

17 04 2022 12:48:10.440:ERROR [launcher]: Chrome stdout:
17 04 2022 12:48:10.440:ERROR [launcher]: Chrome stderr: [13297:13297:0417/124810.410313:ERROR:ozone_platform_x11.cc(247)] Missing X server or $DISPLAY
[13297:13297:0417/124810.411000:ERROR:env.cc(225)] The platform failed to initialize.  Exiting.

17 04 2022 12:48:10.452:INFO [launcher]: Trying to start Chrome again (2/2).
17 04 2022 12:48:10.652:ERROR [launcher]: Cannot start Chrome
        [13329:13329:0417/124810.634376:ERROR:ozone_platform_x11.cc(247)] Missing X server or $DISPLAY
[13329:13329:0417/124810.635022:ERROR:env.cc(225)] The platform failed to initialize.  Exiting.
```

Solution:

Instead of running the test through npm and using chrome browser, If we use ng command with chromeheadless browser, the issue get resolved.

Docker.sock Permission error :

```
Build at: 2022-04-17T09:08:38.882Z - Hash: ca8aa6824393f315 - Time: 75807ms
[Pipeline] echo
Build completed
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (DOCKER IMAGE BUILD)
[Pipeline] sh
+ docker build . -t jpray/scientificcalcdevops:latest
Got permission denied while trying to connect to the Docker daemon socket at unix:///var/run/docker.sock: Post http://%2Fvar%2Frun%2Fdocker.sock/v1.24/build?
buildargs=%7B%7D&cachefrom=%5B%5D&cgroupparent=&cpuperiod=0&cpuquota=0&cpusetcpus=&cpusetmems=&cpushares=0&dockerfile=Dockerfile&labels=%7B%7D&memory=0&memswap=0&networkmode=d
efault&rm=1&shmsize=0&t=jpray%2Fscientificcalcdevops%3Alatest&target=&ulimits=null&version=1: dial unix /var/run/docker.sock: connect: permission denied
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
ERROR: script returned exit code 1
Finished: FAILURE
```

Solution :

Run command **sudo chmod 666 /var/run/docker.sock**
OR
By adding Jenkins to sudo group using command
sudo usermod -aG group jenkins

5 Application Screenshots and Testing

Test Cases :

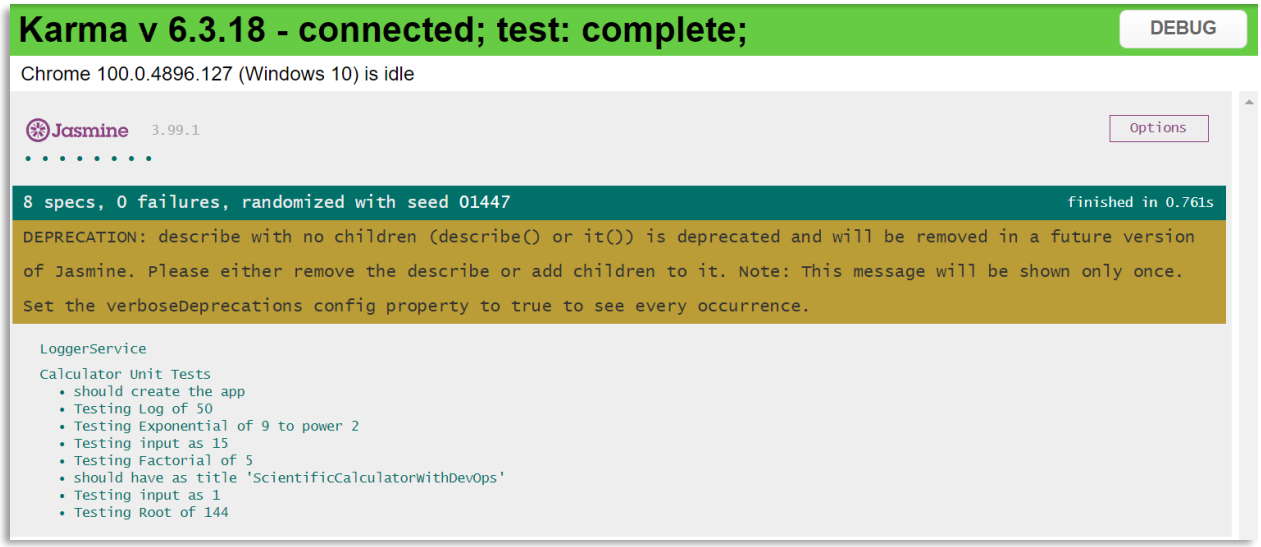


Fig 5.3 Karma Test Runner

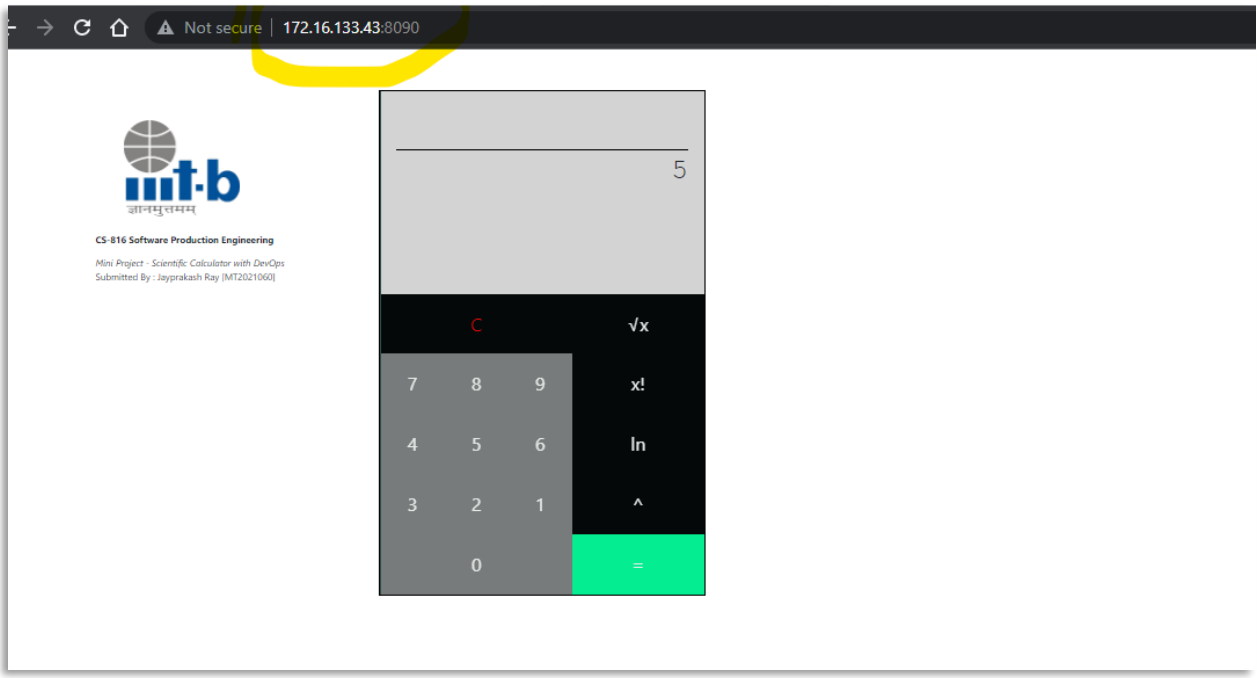
```
import { ComponentFixture, TestBed } from '@angular/core/testing';
import { LoggerModule } from 'ngx-logger';
import { LoggerService } from './_services/logger.service';
import { AppComponent } from './app.component';

describe('Calculator Unit Tests', () => {
  let component: AppComponent;
  let fixture: ComponentFixture<AppComponent>;

  beforeEach(async () => {
    await TestBed.configureTestingModule({
      declarations: [
        AppComponent ]
    })
    .compileComponents();
  });

  beforeEach(() => {
    fixture = TestBed.createComponent(AppComponent);
    component = fixture.componentInstance;
    fixture.detectChanges();
  });

  it('should create the app', () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app).toBeTruthy();
  });

  it(`should have as title 'ScientificCalculatorWithDevOps'`, () => {
    const fixture = TestBed.createComponent(AppComponent);
    const app = fixture.componentInstance;
    expect(app.title).toEqual('ScientificCalculatorWithDevOps');
  });

  it("Testing input as  1", () => {
    const cal = fixture.componentInstance;
    cal.keyPress(1);
    expect(cal.operand1).toEqual(1);
  })
```

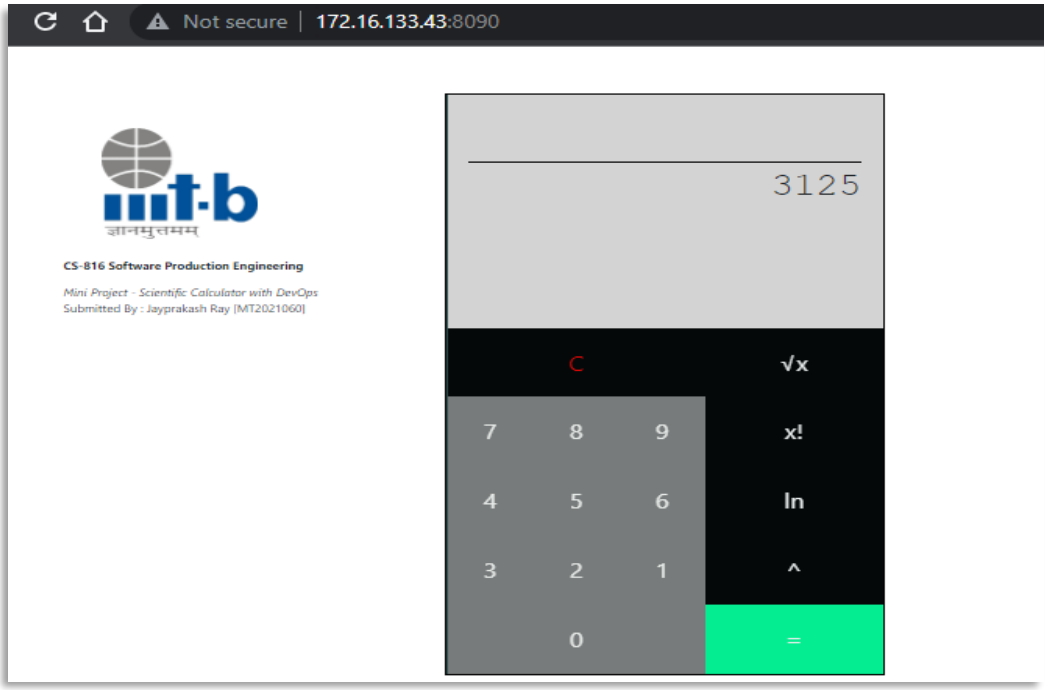
Fig 5.2 Test Script

```
  it("Testing Root of 144", () => {
    const cal = fixture.componentInstance;
    cal.keyPress(1);
    cal.keyPress(4);
    cal.keyPress(4);
    cal.sqroot()
    expect(cal.operand1).toEqual(12);
  })

  it("Testing Factorial of 5", () => {
    const cal = fixture.componentInstance;
    cal.keyPress(5);
    cal.factorial()
    expect(cal.operand1).toEqual(120);
  })

  it("Testing Exponential of 9 to power 2", () => {
    const cal = fixture.componentInstance;
    cal.keyPress(9);
    cal.power()
    cal.keyPress(2)
    cal.calculate()
    expect(cal.operand1).toEqual(81);
  })

  it("Testing Log of 50", () => {
    const cal = fixture.componentInstance;
    cal.keyPress(5);
    cal.keyPress(0);
    cal.log();
    expect(cal.operand1).toEqual(3.912023005428146);
  })

});
```

Fig 5.3 Test Script

Testing 5^5





**yellow marked IP is the deployment server IP where container is runing.*