Calculator development, testing, deployment, and monitoring using DevOps tools

Software Production Engineering(CS 816)

Submitted By:
Parth Patel (MT2020057)

Guided By:
Prof. Thangaraju and TAs

Overview:

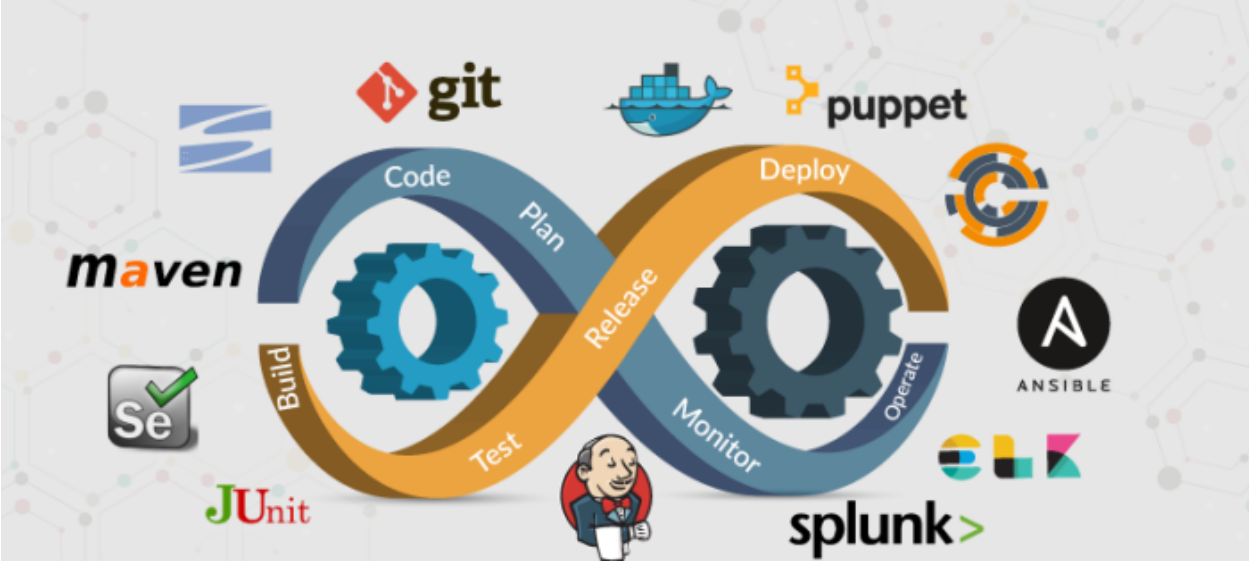


Fig 1. DevOps

The Plan is to create a CLI-based calculator that can perform square root, factorial, natural log, and exponentiation operations by applying DevOps methodology and using various DevOps tools like git for source control management, maven for building and testing application, and also for managing various dependencies. Docker helps to build images that only contain this calculator application and its dependencies that later can be deployed using Ansible on any remote machine or server by pulling that docker image from the docker hub. And finally, to monitor application health ELK has been used.

Following are Purpose and associated tools,
SCM: Github - https://github.com/pparth27743/devops_pipeline.git
Building: Maven
Testing: Junit
Docker Image:
https://hub.docker.com/repository/docker/pparth27743/hello_world
Continuous Integration: Jenkins
Continuous Deployment: Ansible
Monitoring: ELK(Elasticsearch, Logstash, and Kibana)
https://www.elastic.co/

Source Code:

Calculator Application has been developed in java using IntelliJ IDE. There are 2 main files, there first being Calculor.Java which consists of Business logic, code for square root, natural log, factorial, and exponentiation. And Second being CalculatorTest.java to perform unit testing of all the functionally mentioned above.

```java
public class Calculator {

    private static final Logger logger = LogManager.getLogger(Calculator.class);

    public static void  printAns(double ans){
        System.out.println("************************************************");
        System.out.println("Your answer is " + ans);
        System.out.println("************************************************");
    }

    public static double squareRoot(double num){
        double ans = 0;
        try {
            logger.info("Let's calculate square root of " + num);
            if(num < 0){
                ans = Double.NaN;
                throw new ArithmeticException("Given Number is negative so can't find Square Root.");
            }else {
                ans = Math.sqrt(num);
            }
        }catch (ArithmeticException e){
            logger.error("Number should not be negative : " + e.getMessage());
        }finally {
            logger.info("Result : " + ans);
        }
        return ans;
    }

    public static double factorial(double num){
        double ans = 1;
        try {
            logger.info("Let's calculate factorial of " + num);
            if(num < 0){
                ans = Double.NaN;
```

Fig 2. Calculator.java (Contains Business Logic)

```java
public class CalculatorTest {

    Calculator calculator = new Calculator();
    private static final double DELTA = 1e-9;

    @Test
    public void squareRoot_True(){
        assertEquals( message: "True: Square Root of number ",  expected: 9.0, calculator.squareRoot( num: 81), DELTA);
        assertEquals( message: "True: Square Root of number ",  expected: 4.7958315233127 , calculator.squareRoot( num: 23), DELTA);
        assertEquals( message: "True: Square Root of number ", Double.NaN, calculator.squareRoot( num: -10), DELTA);
    }

    @Test
    public void squareRoot_False(){
        assertNotEquals( message: "False: Square Root of number ",  unexpected: 1.0, calculator.squareRoot( num: 18), DELTA);
        assertNotEquals( message: "False: Square Root of number ",  unexpected: 1.0, calculator.squareRoot( num: 32), DELTA);
        assertNotEquals( message: "False: Square Root of number ",  unexpected: 1.0, calculator.squareRoot( num: -16), DELTA);
    }

    @Test
    public void factorial_True(){
        assertEquals( message: "True: Factorial of number for True positive",  expected: 24.0, calculator.factorial( num: 4), DELTA);
        assertEquals( message: "True: Factorial of number for True positive",  expected: 720.0, calculator.factorial( num: 6), DELTA);
        assertEquals( message: "True: Factorial of number for True positive", Double.NaN, calculator.factorial( num: -15), DELTA);
        assertEquals( message: "True: Factorial of number for True positive", Double.NaN, calculator.factorial( num: 10.5123), DELTA);
    }

    @Test
    public void factorial_False(){
        assertNotEquals( message: "False: Factorial of number for False positive",  unexpected: 1.0, calculator.factorial( num: 2), DELTA);
        assertNotEquals( message: "False: Factorial of number for False positive",  unexpected: 1.0, calculator.factorial( num: 10), DELTA);
        assertNotEquals( message: "False: Factorial of number for False positive",  unexpected: 1.0, calculator.factorial( num: -4), DELTA);


    }
```

Fig 3. CalculatorTest.java(contains Unit test cases)

Building:

When we are working with Big project at that time managing dependencies of our project and building them and creating artifacts can be a tedious task, so I have used Maven to address all also issues. Maven can do the following things,

- Downloading dependencies
- Comping source code
- Packaging that code in war/jar files
- Help to create documentation of the application.

All the dependencies are mentioned in the pom.xml file and maven reading that file installs all required dependencies and provides them to the application.
Following dependencies have been used in Project.

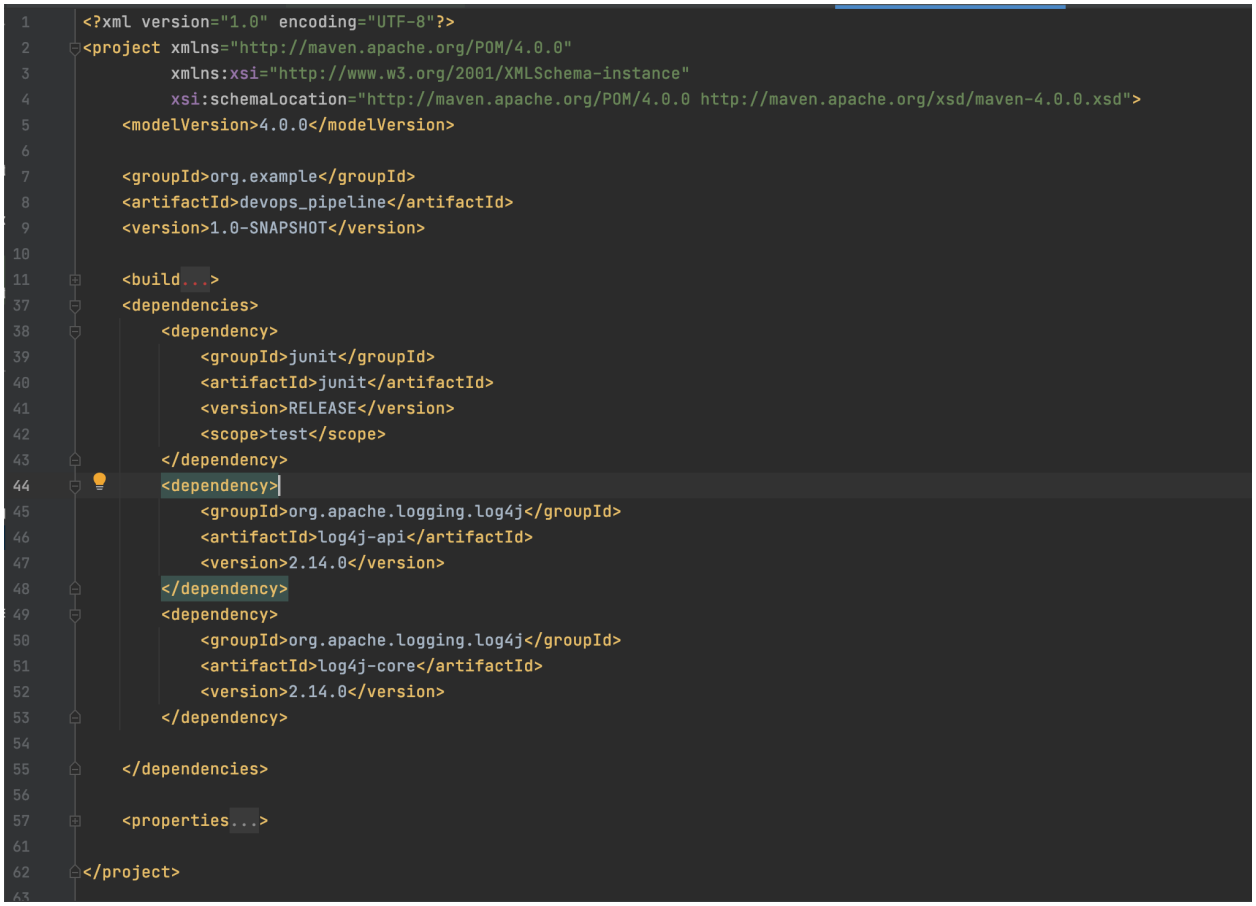


Fig 4. Pom.xml

Testing:

I have done unit testing and to do we need to create test cases. Unit test cases are pieces of code that check that whether program logic works well or not. Following is the way I have created that case.

Logging:

I have added log4j2.xml for logging every operation performed by the calculator application. And it helps to create a calculator.log file which later will be used for ELK.

```java
public class CalculatorTest {

    Calculator calculator = new Calculator();
    private static final double DELTA = 1e-9;

    @Test
    public void squareRoot_True(){
        assertEquals( message: "True: Square Root of number ",  expected: 9.0, calculator.squareRoot( num: 81), DELTA);
        assertEquals( message: "True: Square Root of number ",  expected: 4.7958315233127 , calculator.squareRoot( num: 23), DELTA);
        assertEquals( message: "True: Square Root of number ", Double.NaN, calculator.squareRoot( num: -10), DELTA);
    }

    @Test
    public void squareRoot_False(){
        assertNotEquals( message: "False: Square Root of number ",  unexpected: 1.0, calculator.squareRoot( num: 18), DELTA);
        assertNotEquals( message: "False: Square Root of number ",  unexpected: 1.0, calculator.squareRoot( num: 32), DELTA);
        assertNotEquals( message: "False: Square Root of number ",  unexpected: 1.0, calculator.squareRoot( num: -16), DELTA);
    }

    @Test
    public void factorial_True(){
        assertEquals( message: "True: Factorial of number for True positive",  expected: 24.0, calculator.factorial( num: 4), DELTA);
        assertEquals( message: "True: Factorial of number for True positive",  expected: 720.0, calculator.factorial( num: 6), DELTA);
        assertEquals( message: "True: Factorial of number for True positive", Double.NaN, calculator.factorial( num: -15), DELTA);
        assertEquals( message: "True: Factorial of number for True positive", Double.NaN, calculator.factorial( num: 10.5123), DELTA);
    }

    @Test
    public void factorial_False(){
        assertNotEquals( message: "False: Factorial of number for False positive",  unexpected: 1.0, calculator.factorial( num: 2), DELTA);
        assertNotEquals( message: "False: Factorial of number for False positive",  unexpected: 1.0, calculator.factorial( num: 10), DELTA);
        assertNotEquals( message: "False: Factorial of number for False positive",  unexpected: 1.0, calculator.factorial( num: -4), DELTA);


    }
```

Fig 5. CalculatorTest.java (Contains unit test cases)

Source Code Management (SCM):

To track all the modifications that we have done during the development of the application, an SCM tool like Github is used. Generally, as project grow bigger and bigger we need to maintain the previous version of our code because help us to rollback if new features do not work out or for debugging as well.

The following link has by GitHub repository which contains the source code of the calculator application.

https://github.com/pparth27743/devops_pipeline.git

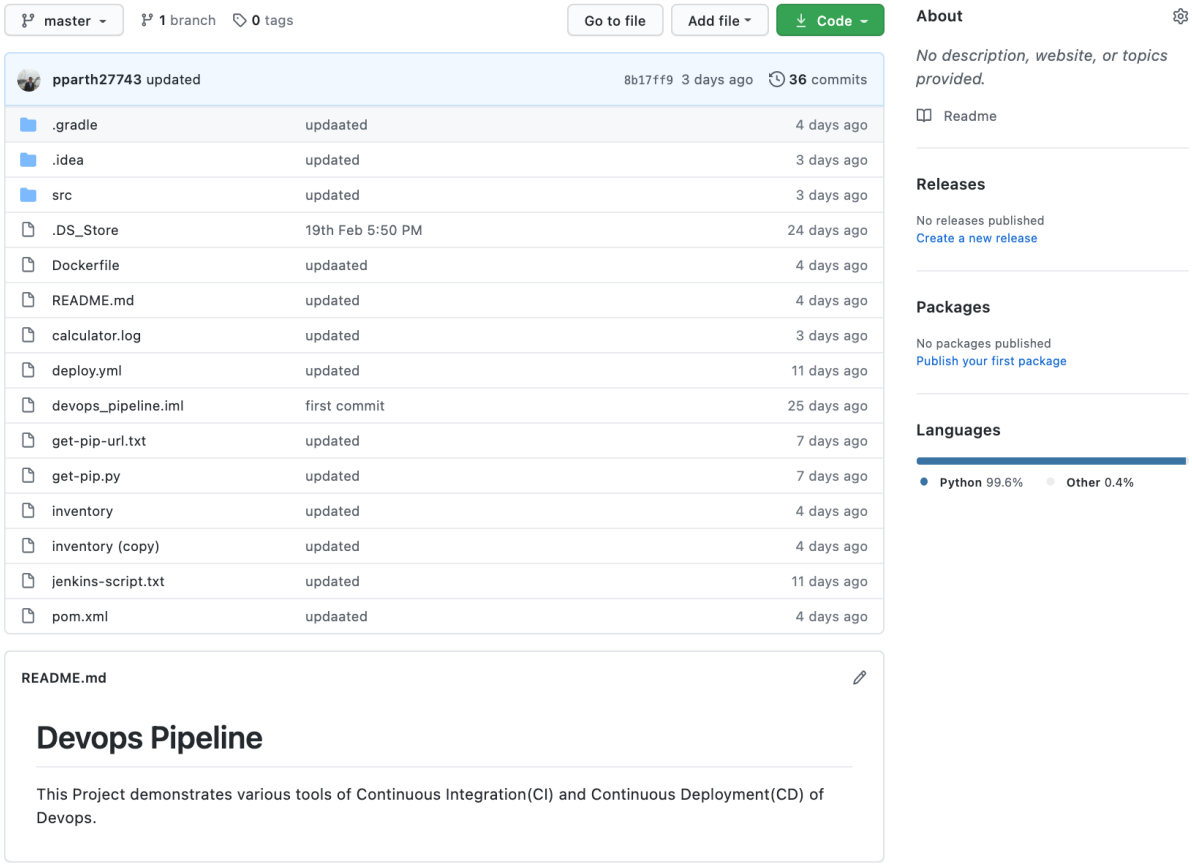


Fig 6. Github repository

Now we have developed our project and want to push that to GitHub the following command are used.

```
git init  // to initialize current directory as git repository

git remote add origin <git_repo_url> // it will map remote
report with current git repository

git push -u origin <branch_name>    // it will push all the
newly created code to GitHub repository
```

After doing the first commit, as and when we update the code we can push that code to Github by using the following command.

```
git add . // put all the updated and newly created files into the staging area

git commit -m "message"  // it will commit all the changes and save it with a provided message

git push -u origin <branch_name>    // it will push all the changes GitHub repository
```

Docker, Dockerfile and DockerHub:

Docker container is popularly known as a lightweight virtual machine that only contains the application and its dependencies. On a machine, we can run thousands of containers whereas fewer Virtual Machines can be created on the machine.

To create a docker container of our application we need to create a Docker Image and for that, we first need to create Dockerfile which has all the instructions and dependencies that are downloaded when the image is built.

Following is the Dockerfile I have created for the pack calculator application into the container.

```
FROM openjdk:8
COPY ./target/devops_pipeline-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java", "-jar", "devops_pipeline-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

Fig 7. Dockerfile

First I have included the base Image of Java 1.8 as a calculator application run on JVM. Then the jar file has been moved to the root directory of the docker image and then made that directory a working directory and finally, I stated the command to run the jar file. Now as soon as we create any docker container, it will directly run the application.

Based on the above docker file docker will create docker image name as pparth27743/hello_world and that images will be a push to docker hub(Docker Hub is the world's largest library and community for container images). Following is the image that shows Docker Image has been uploaded on Docker Hub.

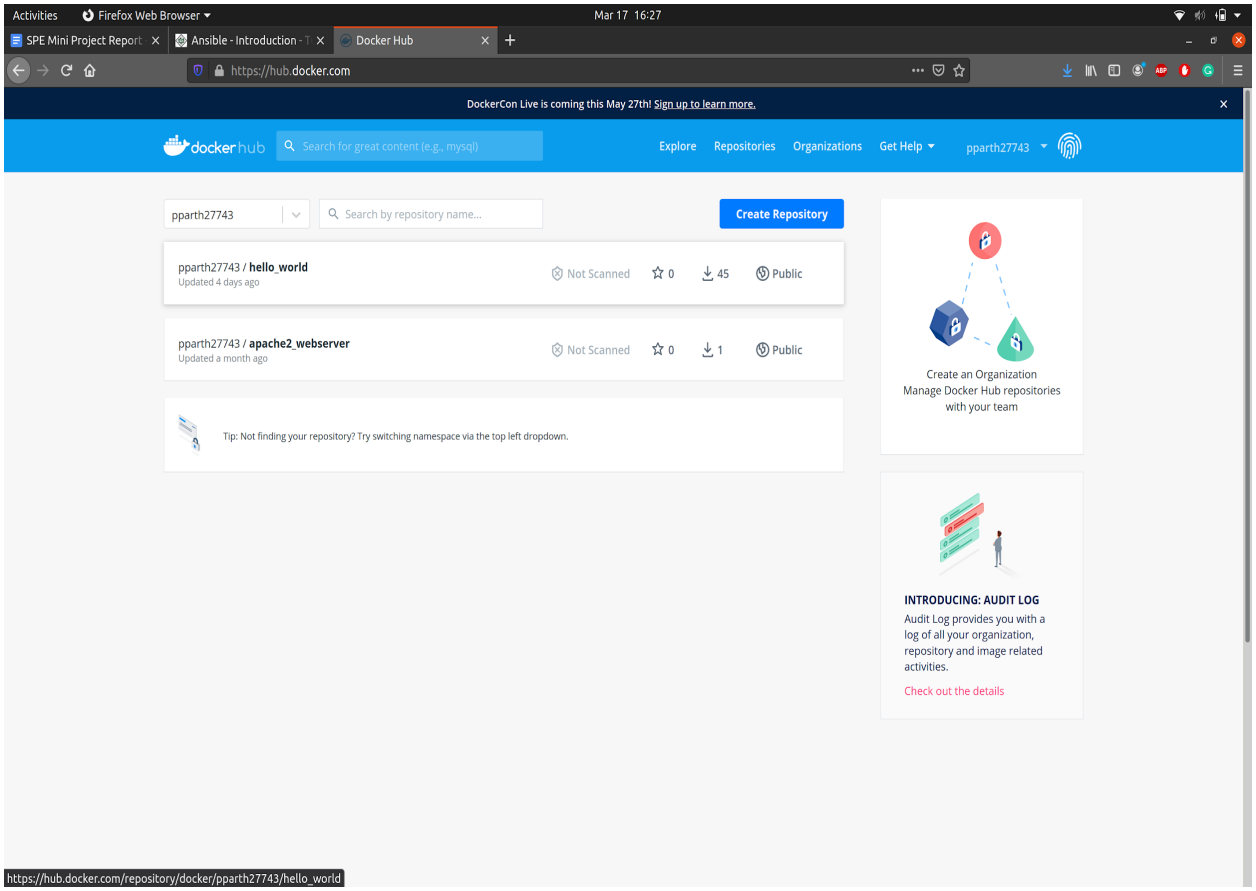


Fig 8. Docker Hub

Ansible:

Ansible is a simple open-source IT engine that automates application deployment, intra service orchestration, cloud provisioning, and many other IT tools.

Ansible is easy to deploy as it does not have any agent (agentless) tools. It only needs to be installed on the controlled node. The managed node need not install ansible in it. Ansible uses a playbook to define tasks that need to be performed on the managed host and these tasks are known as play in an ansible world. Ansible uses ssh for logging into the managed host. And this host information is mentioned inside the inventory file. Ansible playbooks are in a very simple language that is YAML.

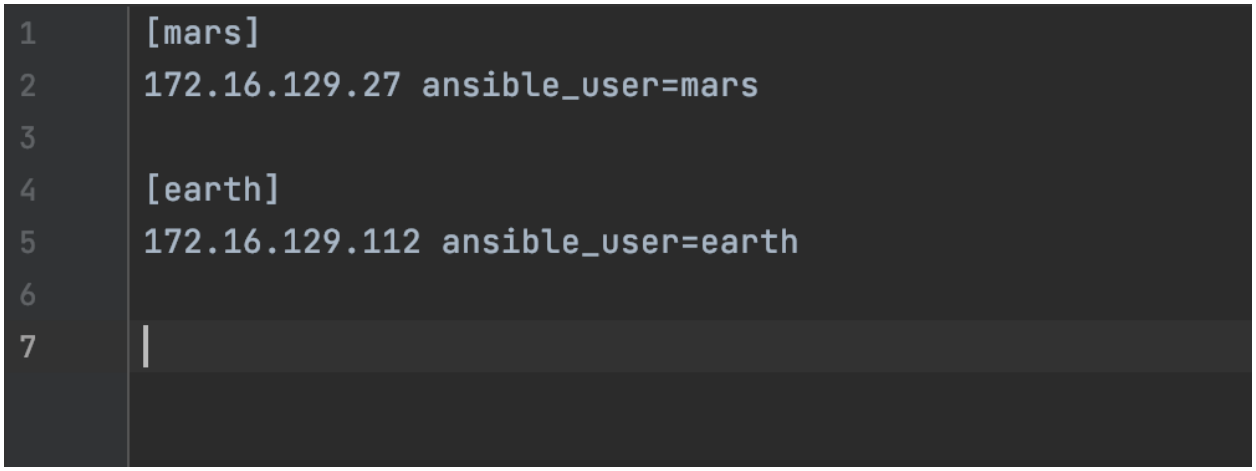


Fig 9. Inventory file

There are 2 managed hosts namely mars and earth. I have created these 2 users on the Virtual Machine using Ubuntu Live server image.

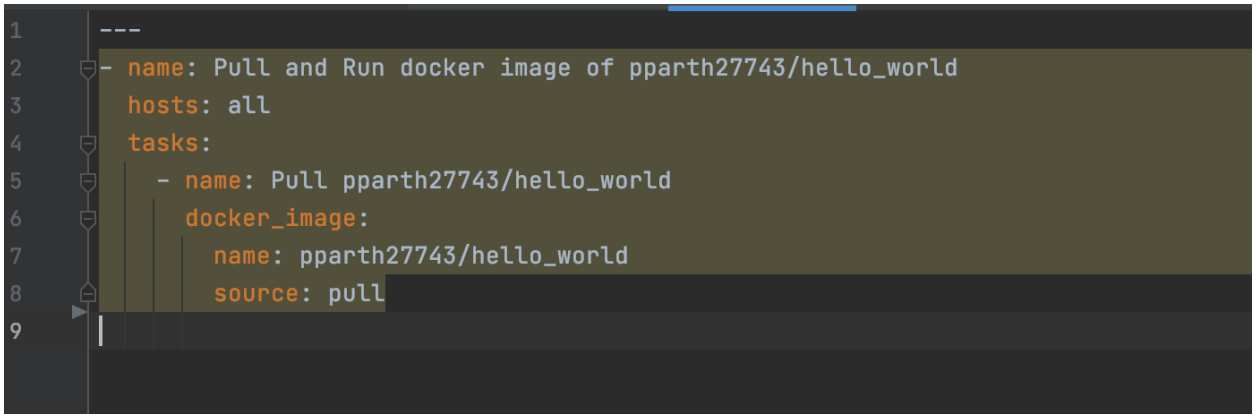


Fig 10. Ansible playbook

The above playbook will pull an image named ‘pparth27743/hello_world’ into a managed host that is mentioned in the inventory file.

Jenkins and Pipeline:

All the above steps can be done in an automated fashion by using Jenkins. Jenkins is is a free and open-source automation server. It helps automate the parts of software development related to building, testing, and deploying, facilitating continuous integration and continuous delivery. Jenkins by default runs on 8080 port.

We need to install the necessary plugins for Git, Maven, Ansible and need to store credentials to push docker image on to docker hub.

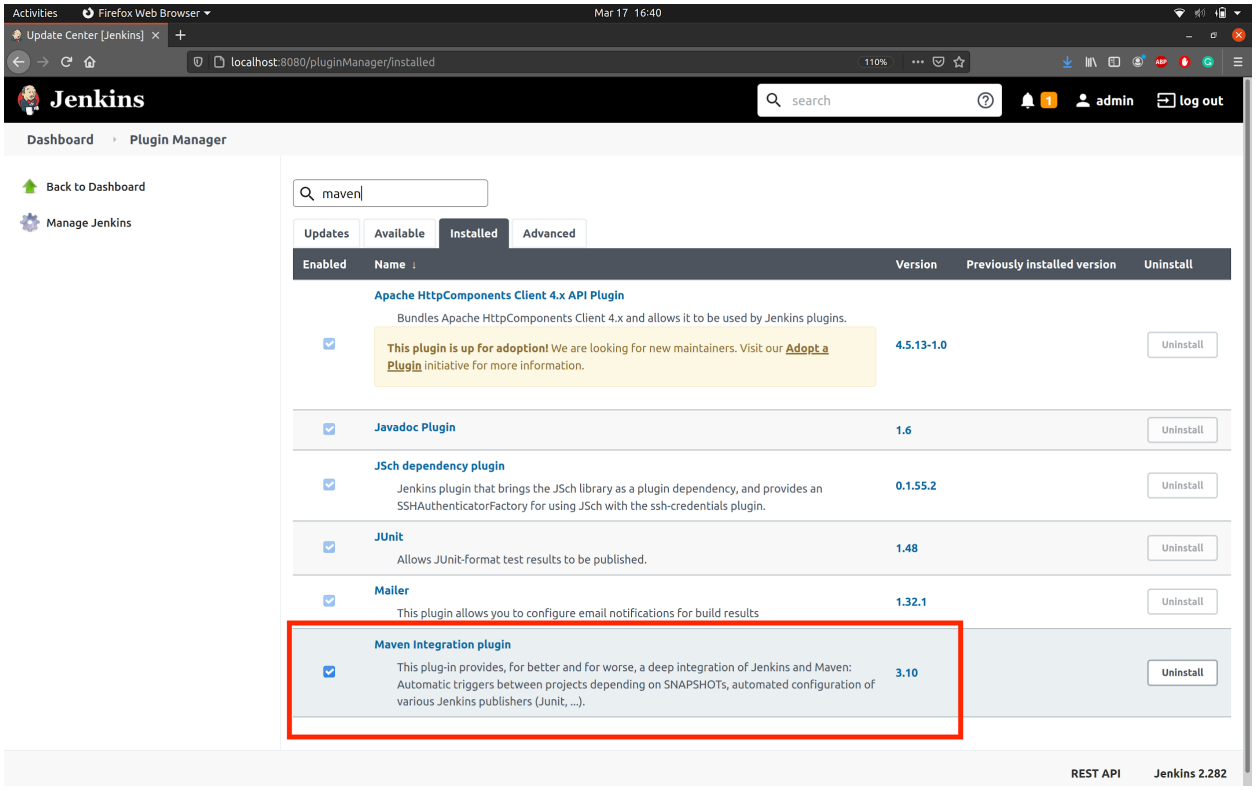


Fig 11. Install Maven Plugin

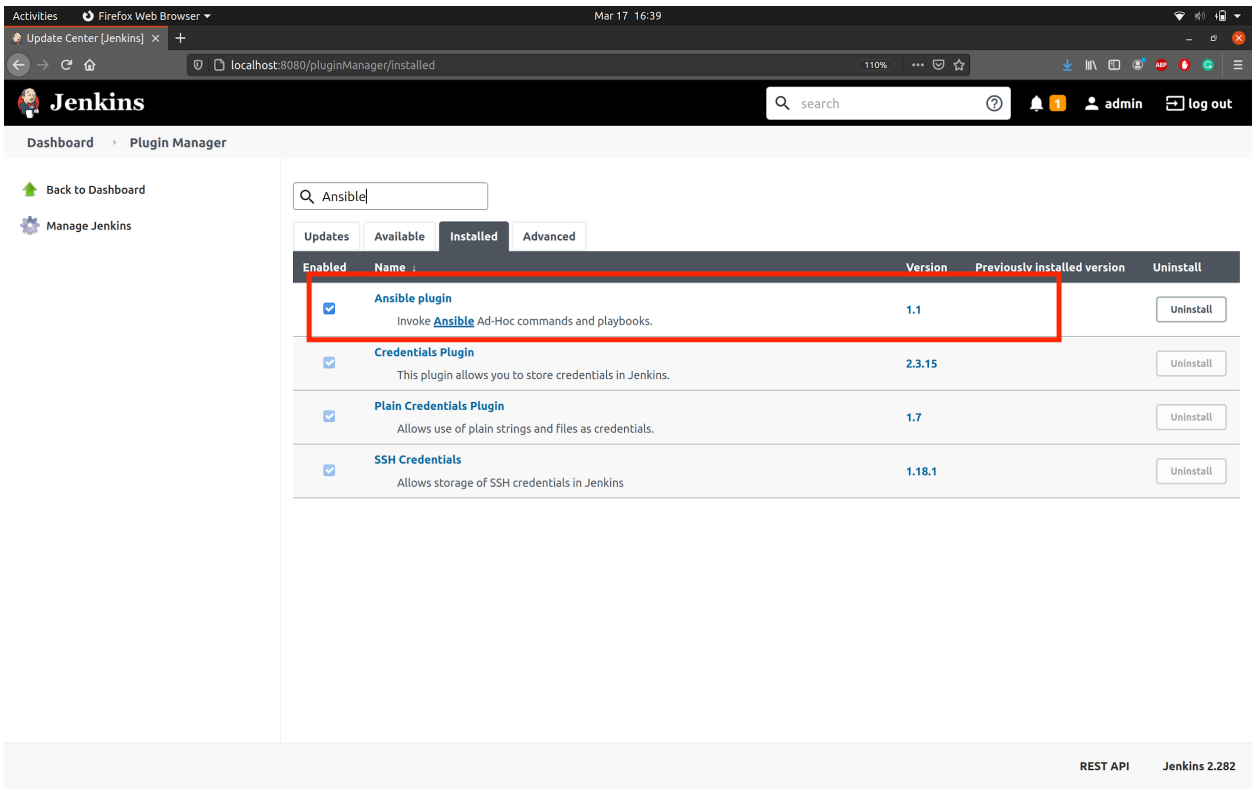


Fig 12. Install Ansible Plugin

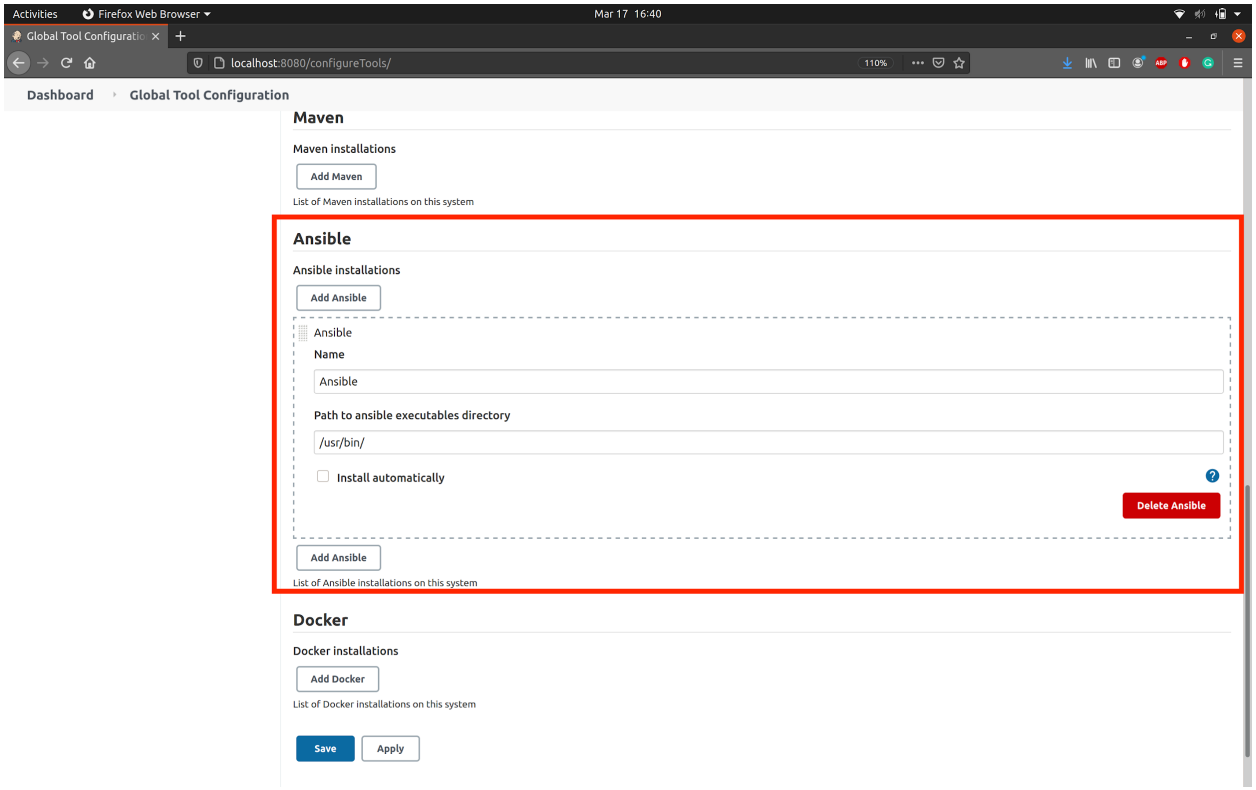


Fig 13. Add ansible path in Jenkins

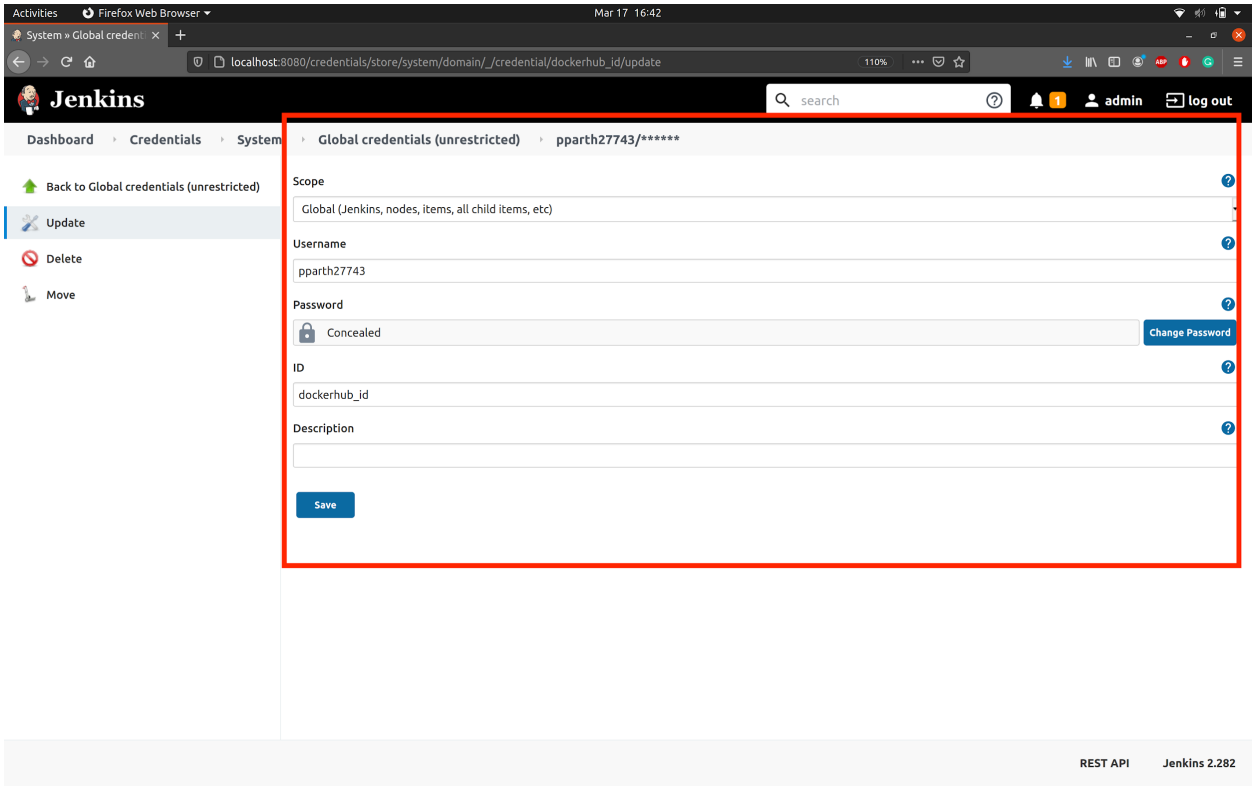


Fig 14. Add credentials for Docker hub into Jenkins, so Jenkins can upload the built image to Docker Hub

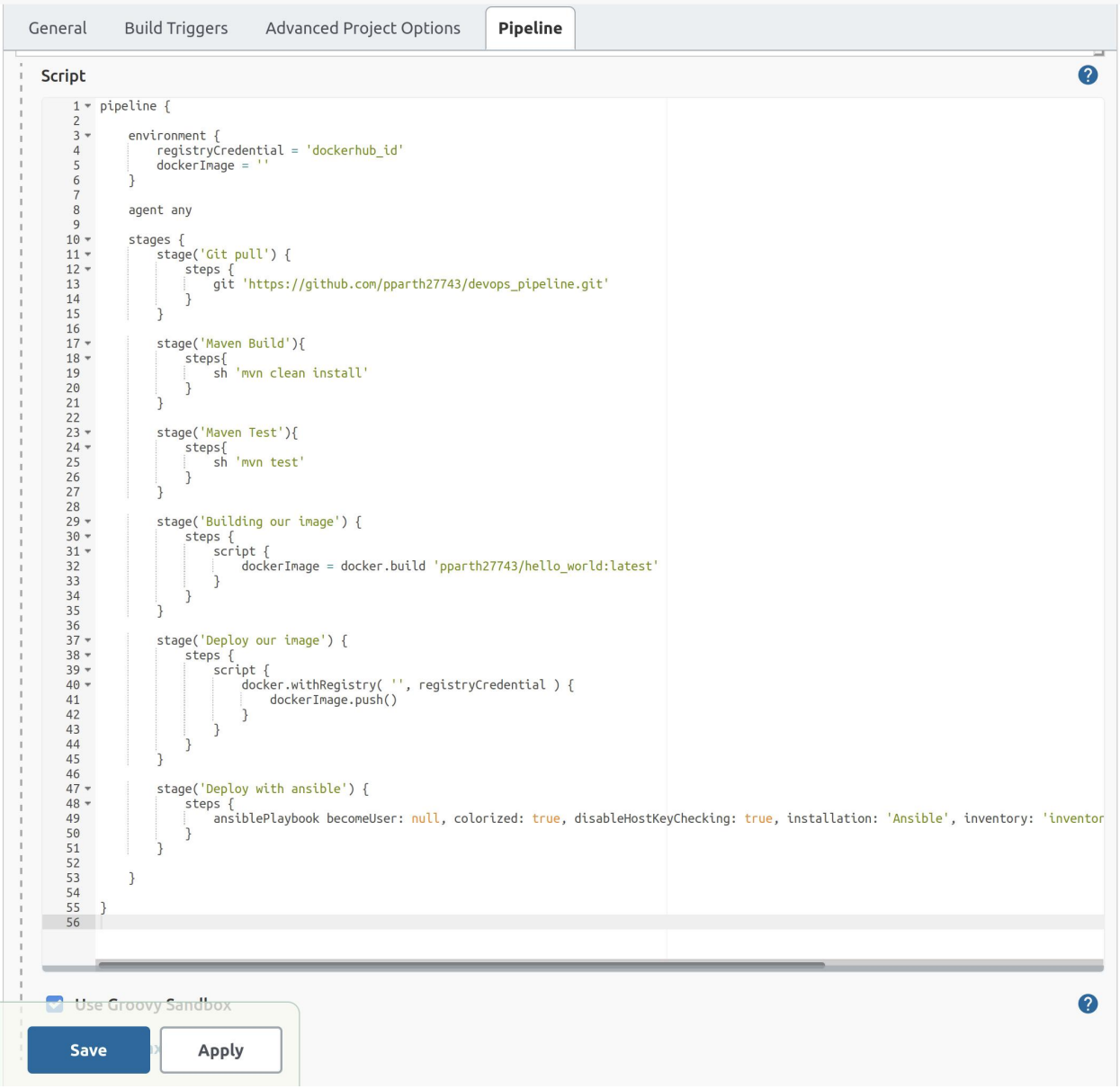


Fig 15. Jenkins Pipeline Script

In this script first GitHub repository will be pulled from the mention GitHub URL, then Jenkins will build this application and also run unit test cases using Maven. Then come to the Image Building stage where docker will build images using the docker file that I have written earlier. And then that image will be a push to the docker hub. Here Already I have stored my credentials for the docker hub in Jenkins which we are used here in this

stage. And Lastly, ansible will run the playbook with an inventory file to deploy images to a managed host.

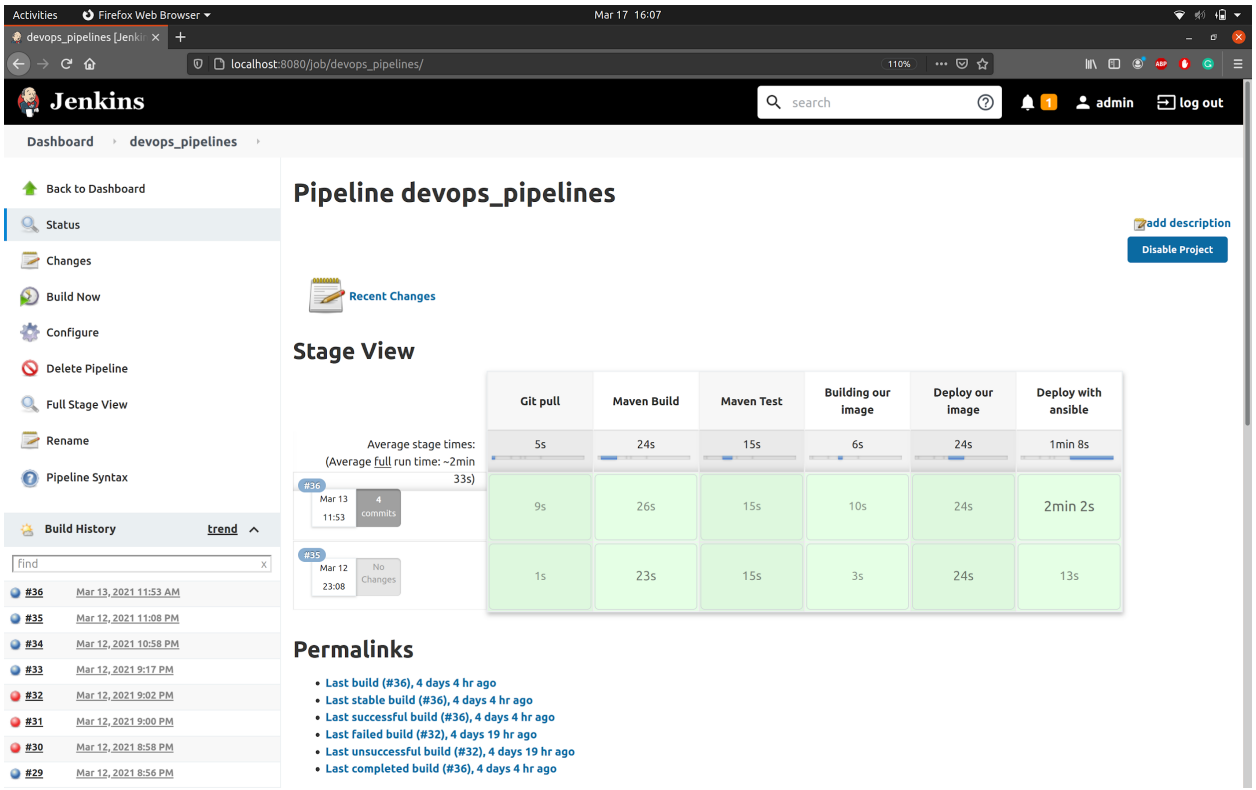


Fig 16. Successful Build of the pipeline.

As Pipeline has successfully run. That means the managed host has a docker image. As I mentioned earlier, I have 2 managed host namely mars and earth so following are some screenshot that shows that Calculator images successfully have been installed and also container ran successfully running our calculator application.

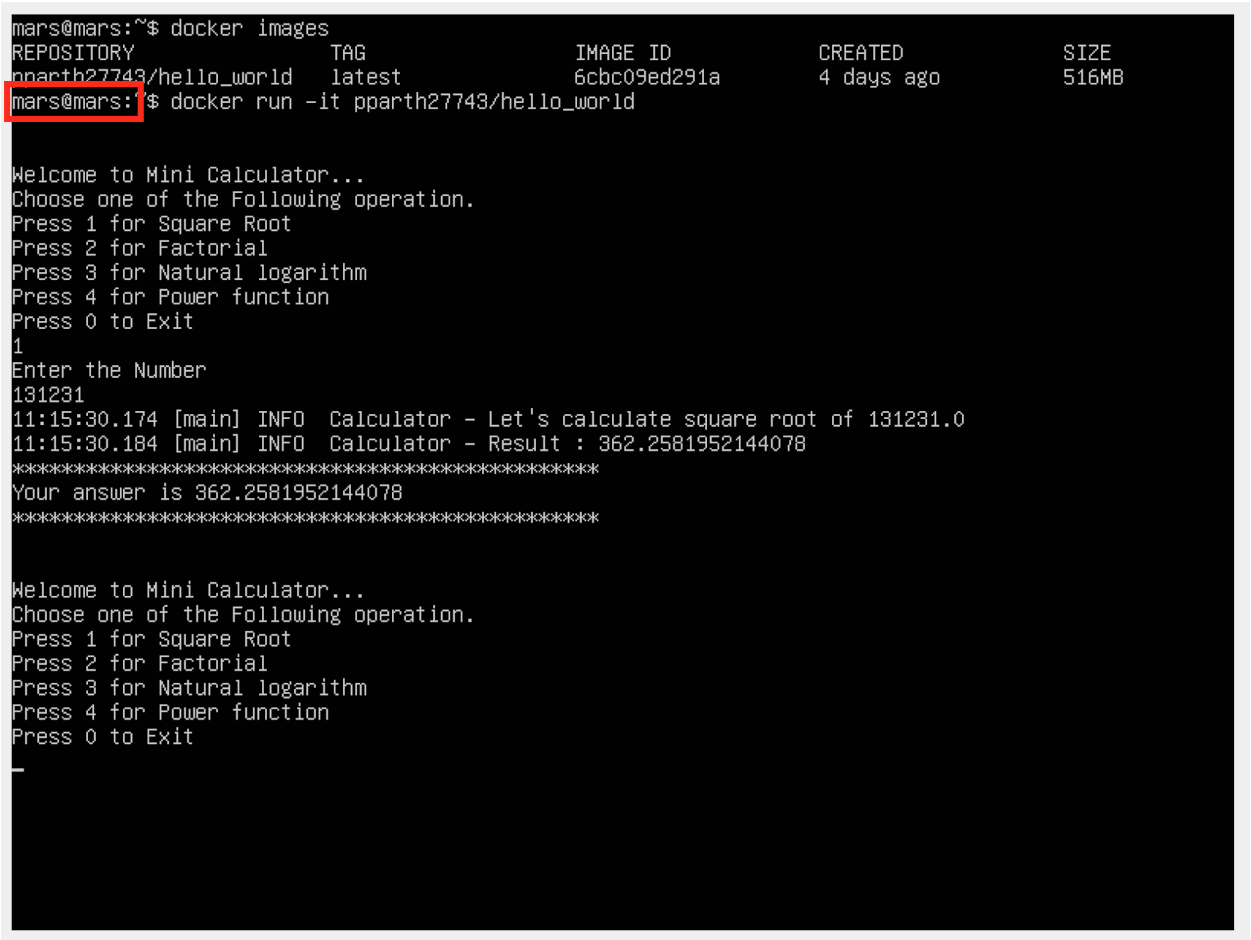


Fig 17. Mars (Managed Host)

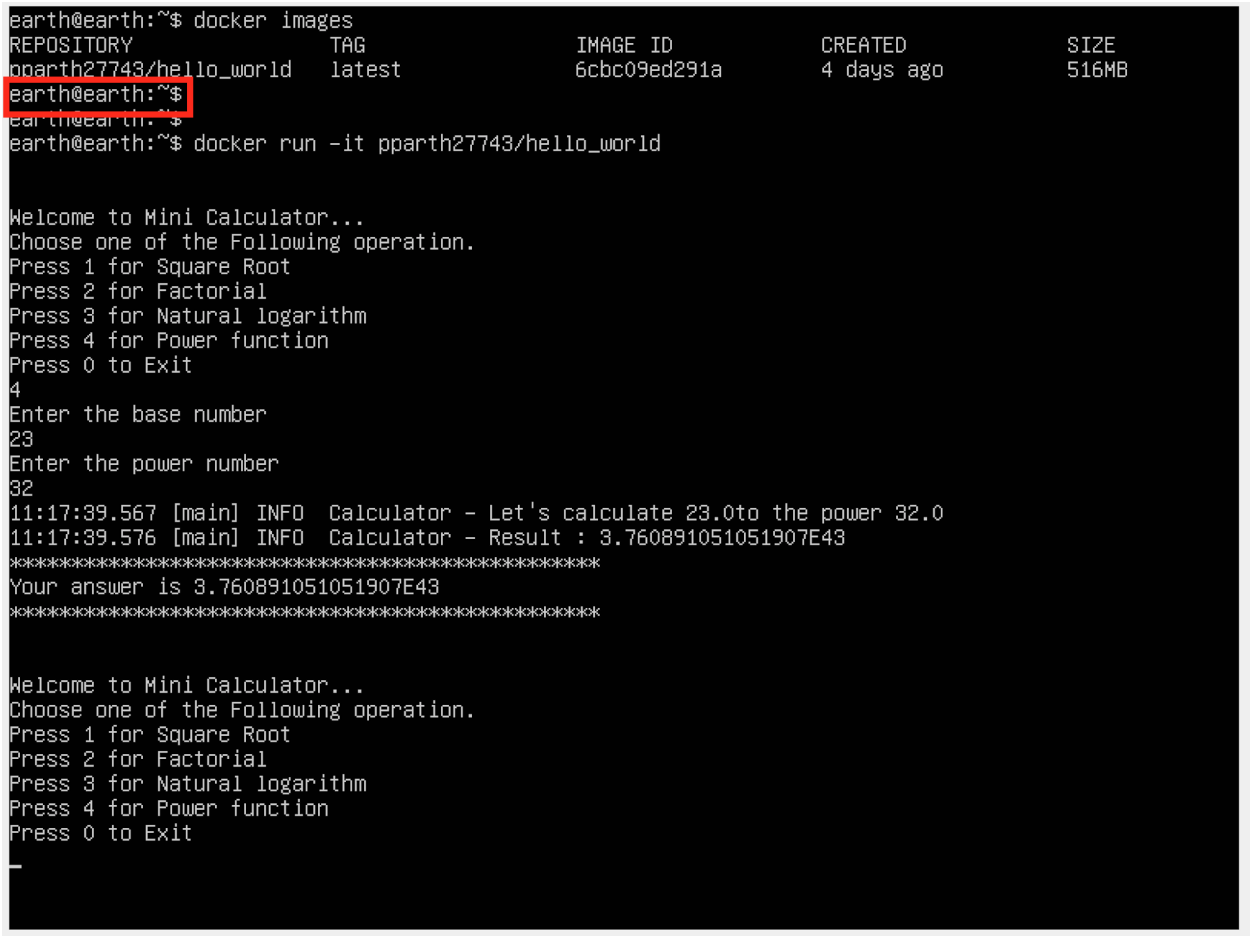
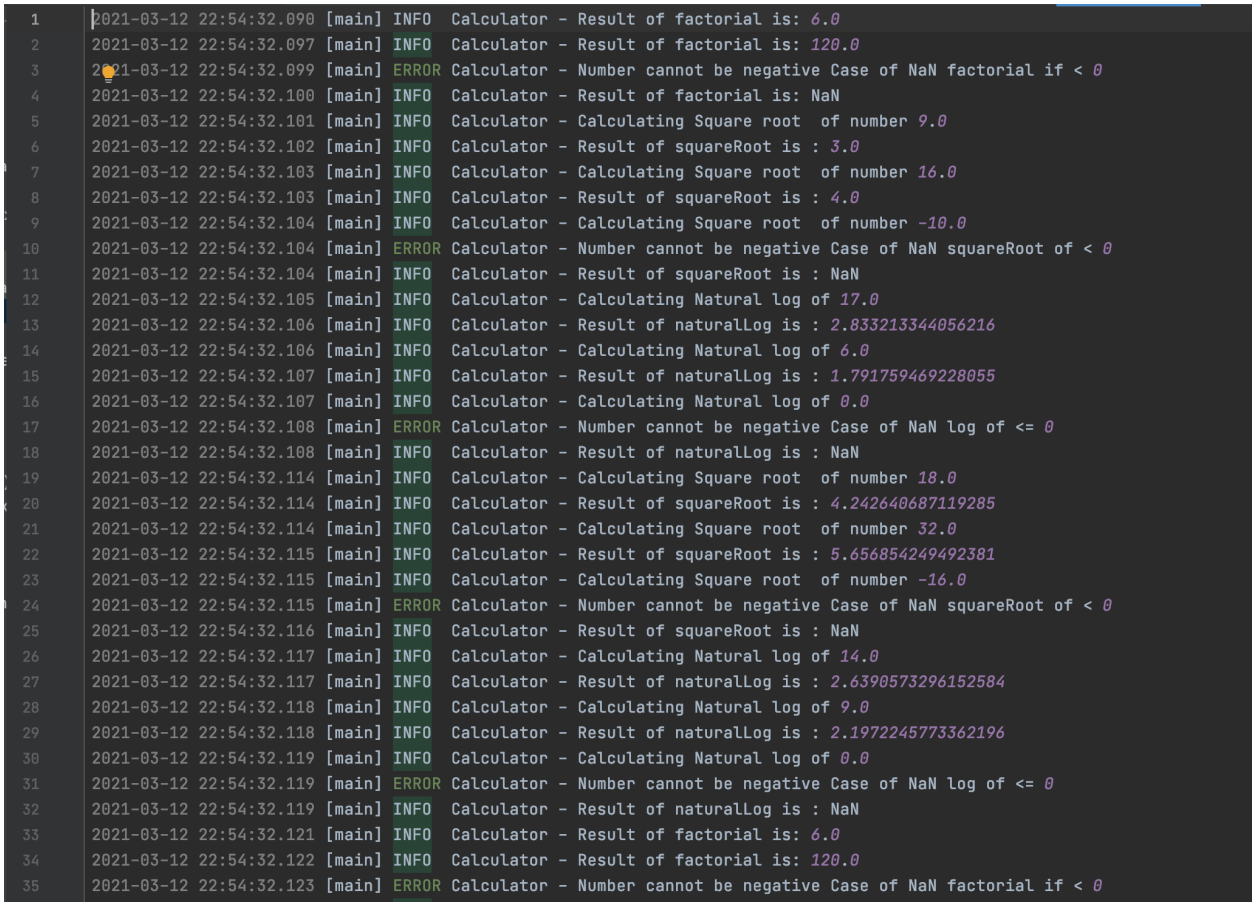


Fig 18. Earth (Managed Host)

Monitoring using ELK:

"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch. Kibana lets users visualize data with charts and graphs in Elasticsearch. The Elastic Stack is the next evolution of the ELK Stack.



```
2021-03-12 22:54:32.090 [main] INFO  Calculator - Result of factorial is: 6.0
2021-03-12 22:54:32.097 [main] INFO  Calculator - Result of factorial is: 120.0
2021-03-12 22:54:32.099 [main] ERROR Calculator - Number cannot be negative Case of NaN factorial if < 0
2021-03-12 22:54:32.100 [main] INFO  Calculator - Result of factorial is: NaN
2021-03-12 22:54:32.101 [main] INFO  Calculator - Calculating Square root  of number 9.0
2021-03-12 22:54:32.102 [main] INFO  Calculator - Result of squareRoot is : 3.0
2021-03-12 22:54:32.103 [main] INFO  Calculator - Calculating Square root  of number 16.0
2021-03-12 22:54:32.103 [main] INFO  Calculator - Result of squareRoot is : 4.0
2021-03-12 22:54:32.104 [main] INFO  Calculator - Calculating Square root  of number -10.0
2021-03-12 22:54:32.104 [main] ERROR Calculator - Number cannot be negative Case of NaN squareRoot of < 0
2021-03-12 22:54:32.104 [main] INFO  Calculator - Result of squareRoot is : NaN
2021-03-12 22:54:32.105 [main] INFO  Calculator - Calculating Natural log of 17.0
2021-03-12 22:54:32.106 [main] INFO  Calculator - Result of naturalLog is : 2.833213344056216
2021-03-12 22:54:32.106 [main] INFO  Calculator - Calculating Natural log of 6.0
2021-03-12 22:54:32.107 [main] INFO  Calculator - Result of naturalLog is : 1.791759469228055
2021-03-12 22:54:32.107 [main] INFO  Calculator - Calculating Natural log of 0.0
2021-03-12 22:54:32.108 [main] ERROR Calculator - Number cannot be negative Case of NaN log of <= 0
2021-03-12 22:54:32.108 [main] INFO  Calculator - Result of naturalLog is : NaN
2021-03-12 22:54:32.114 [main] INFO  Calculator - Calculating Square root  of number 18.0
2021-03-12 22:54:32.114 [main] INFO  Calculator - Result of squareRoot is : 4.242640687119285
2021-03-12 22:54:32.114 [main] INFO  Calculator - Calculating Square root  of number 32.0
2021-03-12 22:54:32.115 [main] INFO  Calculator - Result of squareRoot is : 5.656854249492381
2021-03-12 22:54:32.115 [main] INFO  Calculator - Calculating Square root  of number -16.0
2021-03-12 22:54:32.115 [main] ERROR Calculator - Number cannot be negative Case of NaN squareRoot of < 0
2021-03-12 22:54:32.116 [main] INFO  Calculator - Result of squareRoot is : NaN
2021-03-12 22:54:32.117 [main] INFO  Calculator - Calculating Natural log of 14.0
2021-03-12 22:54:32.117 [main] INFO  Calculator - Result of naturalLog is : 2.6390573296152584
2021-03-12 22:54:32.118 [main] INFO  Calculator - Calculating Natural log of 9.0
2021-03-12 22:54:32.118 [main] INFO  Calculator - Result of naturalLog is : 2.1972245773362196
2021-03-12 22:54:32.119 [main] INFO  Calculator - Calculating Natural log of 0.0
2021-03-12 22:54:32.119 [main] ERROR Calculator - Number cannot be negative Case of NaN log of <= 0
2021-03-12 22:54:32.119 [main] INFO  Calculator - Result of naturalLog is : NaN
2021-03-12 22:54:32.121 [main] INFO  Calculator - Result of factorial is: 6.0
2021-03-12 22:54:32.122 [main] INFO  Calculator - Result of factorial is: 120.0
2021-03-12 22:54:32.123 [main] ERROR Calculator - Number cannot be negative Case of NaN factorial if < 0
```

Fig 19. Calculator.log file

Now, this log file needs to be uploaded to https://cloud.elastic.co. It is an online service that provides ELK.

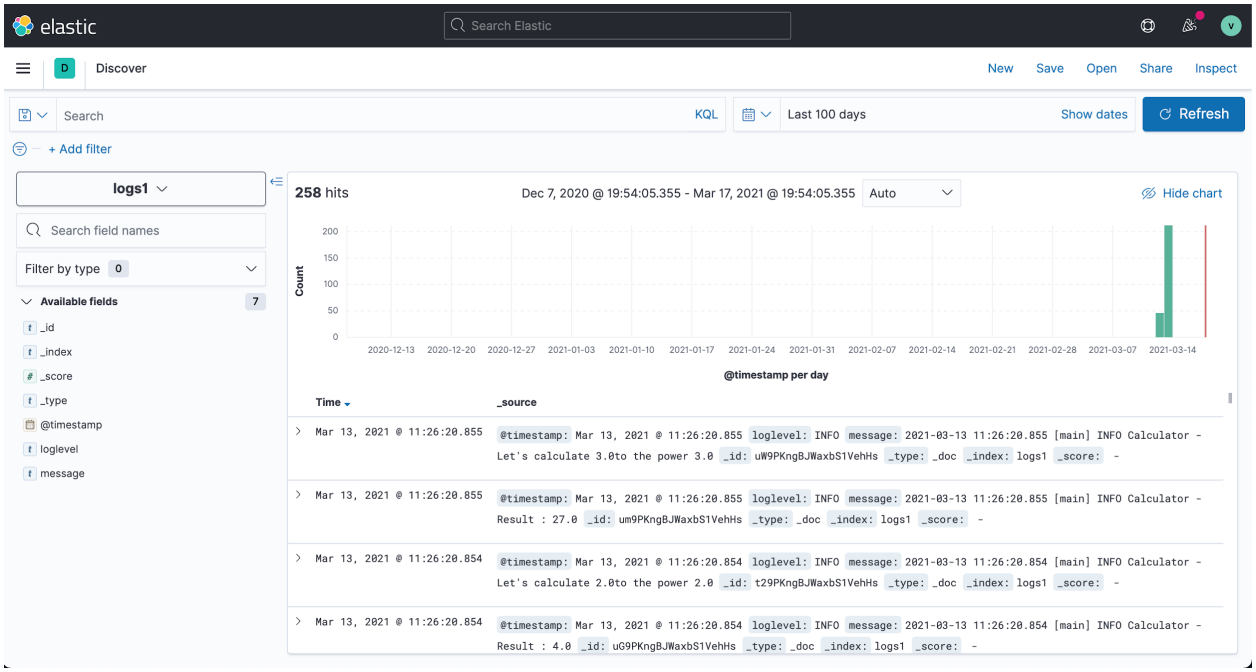


Fig 20. ELK Monitoring

Challenge and Resolution:

1. **Problem:** In Jenkins Pipeline when Ansible tried to pull the image from the docker hub at that time following error was appearing.

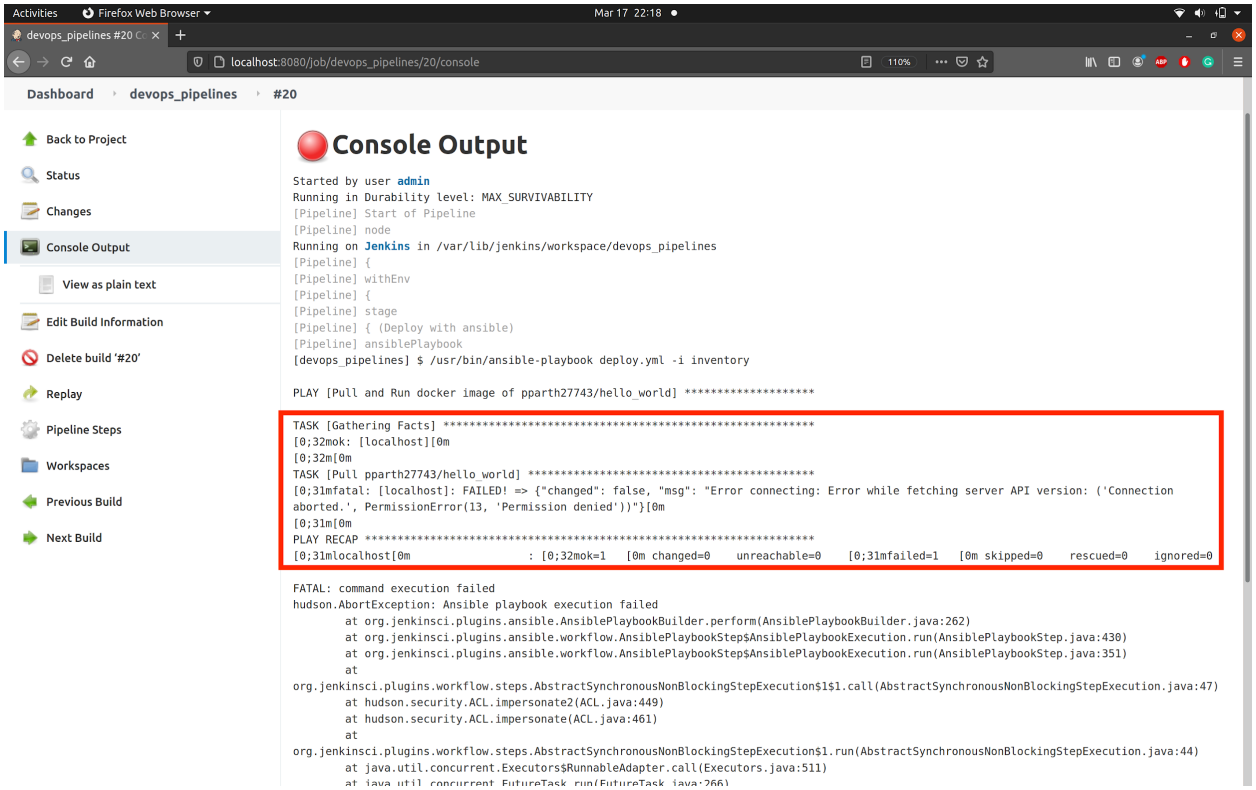


Fig 21. Error 1

Resolution: To solve this error we needed to install docker into the managed host.

2. **Problem:** In Jenkins pipeline, when ansible tried to pull image from docker hub at that time following error was appearing. This error is different than what I had above.

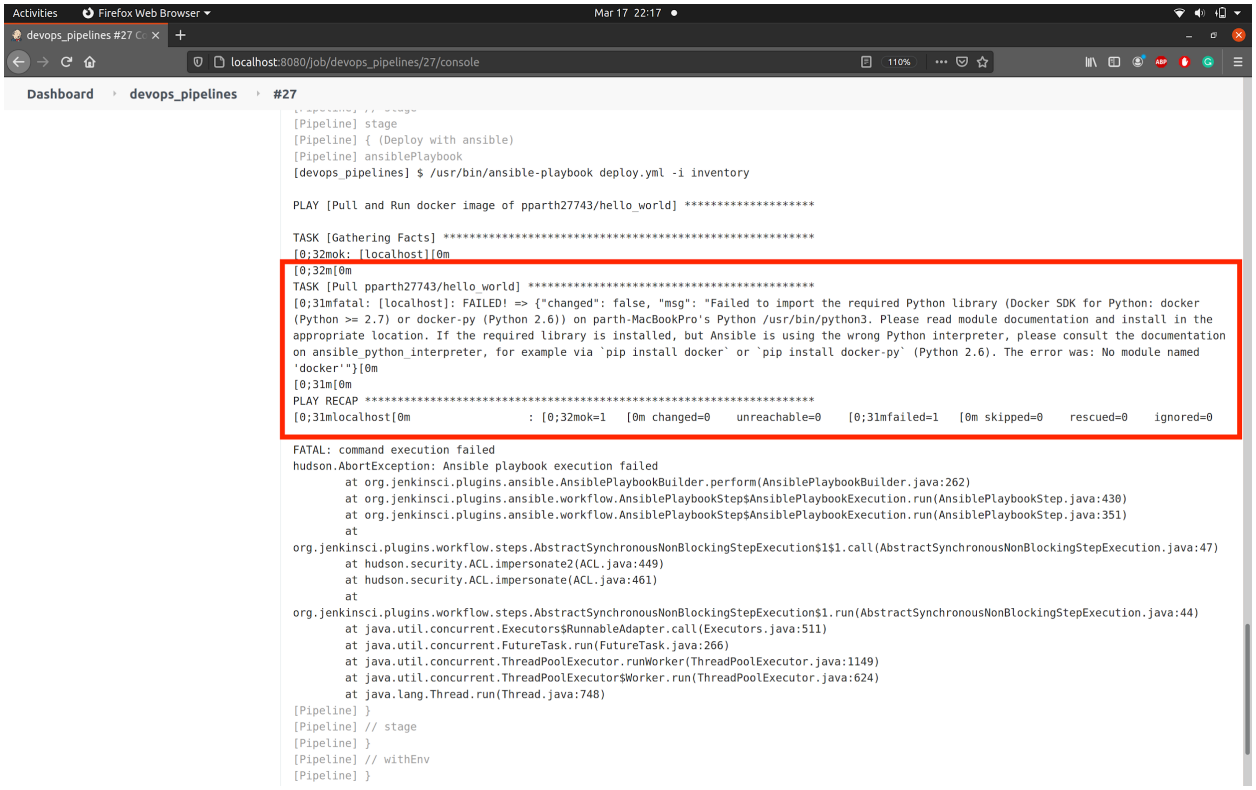


Fig 22. Error 2

Resolution: For this, I needed to install the docker package in Python 3 and as well as Python 2.7. Installing the docker package in python3 was easy but to install the same in python 2.7 firstly I needed to install pip 2 and it was easy to install pip2 either. To install pip2, I had to download a get-pip.py named python file I found on stack overflow. And after that had to run that python file using python2.7 so that pip2 can point to 2.7. And finally, I install the docker package in python2.7.

3. **Problem:** In the below 2.8 Ansible version there was no docker_image module so it was giving an error that no module found ‘docker_image’.

   Resolution: I had to uninstall ansible and then I installed the latest ansible and then it worked.

References:

[1] https://www.jenkins.io/doc/
[2] https://docs.ansible.com/ansible/latest/scenario_guides/guide_docker.html
[3] https://docs.github.com/en
[4] https://junit.org/junit5/docs/current/user-guide/
[5] https://www.google.co.in/
[6] https://stackoverflow.com/