DEVOPS CALCULATOR PROJECT

MT2019114
Sravya M

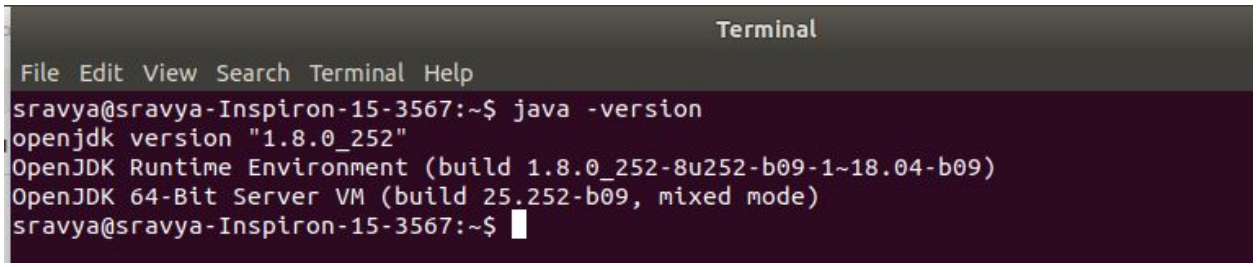
Github link: https://github.com/Sravya-M/SPE_Calculator
Docker hub link:
https://hub.docker.com/repository/registry-1.docker.io/sravya241994/specalculator/tags?page=1

Plan

Plan regarding the application we need to develop. In this case, the application is a Calculator application.

Code

Code the application as per the requirement. I am using ***Java*** as the coding language with the following specifications.



Create a Java Program for Calculator. Few useful commands:

*$**gedit** Calculator.java*	*// create file and write your code in it*
*$**javac** Calculator.java*	*// to compile the program, we get a class file (exec)*
*$**java** Calculator*	*// to execute the class file*

Build

Building a project means to integrate all the compiled files together, many other tasks including compiling and executing tests, generating reports, packaging the project into a single war, jar, etc file.

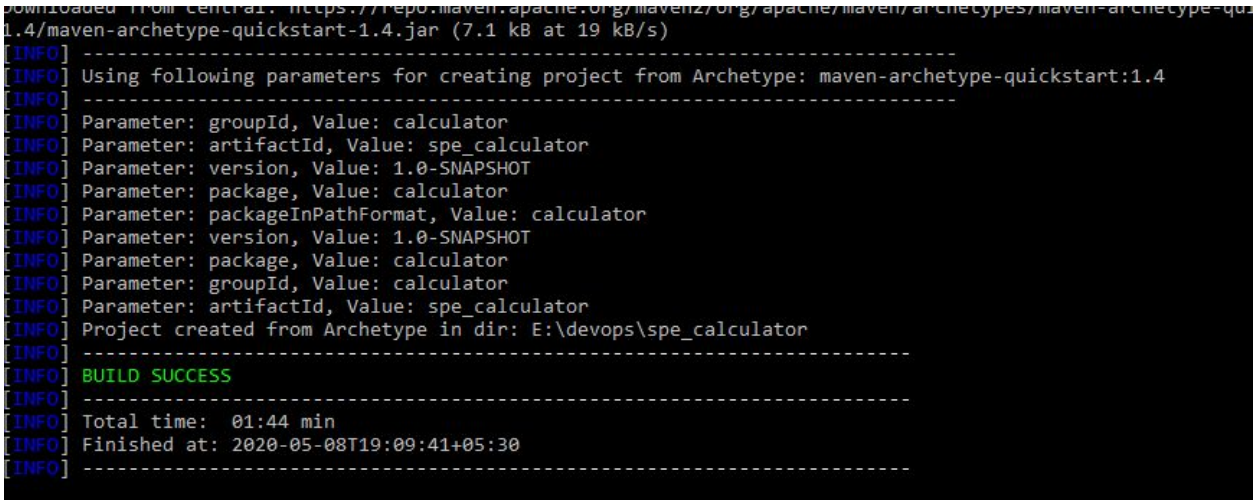
I am using ***Maven*** as the build tool.

To create a maven project,

```
$mvn archetype:generate -DgroupId=calculator -DartifactId=spe_calculator
-DarchetypeArtifactId=maven-archetype-quickstart -DarchetypeVersion=1.4
-DinteractiveMode=false
```

Groupid: group is like a package, Artefact Id: artefact is a project, all other parameters are for configuring basic dependencies in your project, On the first usage of this command, maven downloads all the configuration files from Maven Repository.

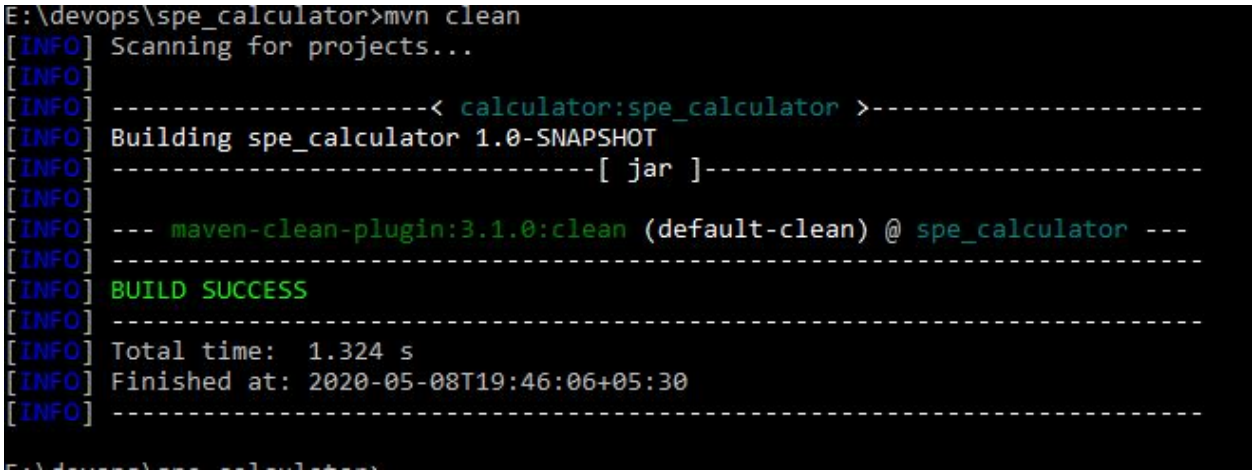
In the current folder where you execute this command, maven creates a folder with artefact id as the project name, hierarchy will be {projectname}/src/main/java/{groupname}/App.java



Replace App.java with your java file. Add “package {groupname};” at top of java file. And run “mvn clean” in the project folder where Maven will find the pom.xml file.

*$**mvn** clean*	*// to clean if there is a previous build target folder*
*$**mvn** compile*	*// To compile all the source code files in the package we created*
*$**mvn** install*	*// Creates a binary executable .jar file in target folder*
*$**mvn** site*	*// To automatically create the project doc report in target/site folder*

You will get “BUILD SUCCESS” after the execution of each command if there are no errors.



Source Code Management

SCM is used to track revisions in software. Each revision is given a timestamp and includes the name of the person who is responsible for the change.

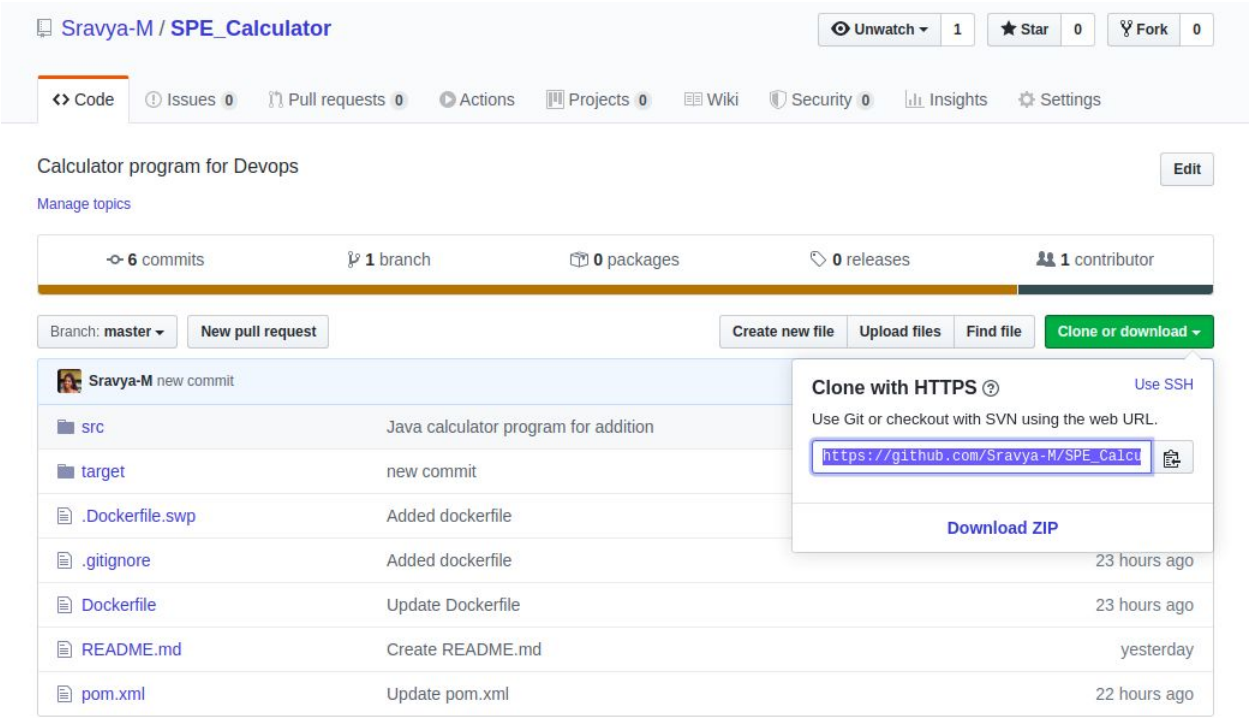
I am using ***Git*** on my local machine to manage source code, ***github*** (online service) for uploading, downloading and sharing of repositories. (github will be useful to link my source code to other services like jenkins)

Create a new repository in Github (keep it public).

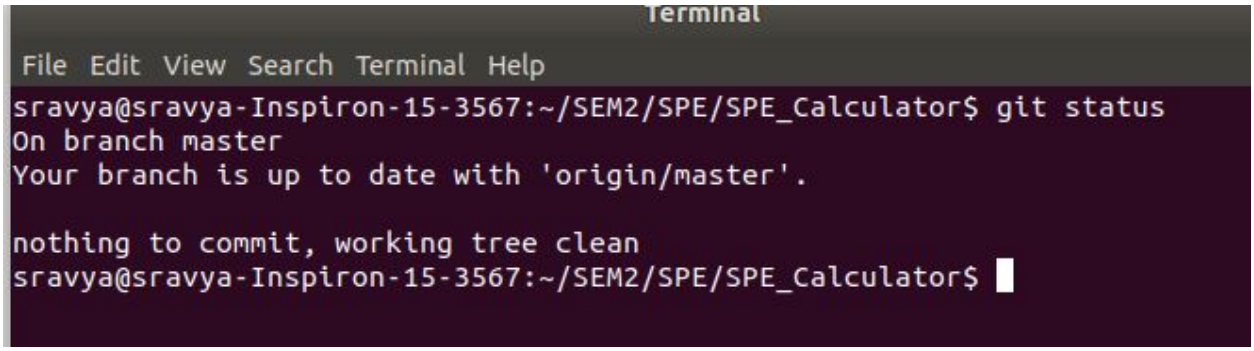
In the local project folder:

*$**git** init*	*// To tell computer that Demo is a directory managed by the Git program*

*$**git** remote add origin <github folder url>*	*// Connect your GitHub repo with your computer*
*$**git** remote -v*	*// to check if remote has been added*
*$**git** pull origin master*	*// downloading from remote repo*
*$**git** status*	*// checking the status of local repo*
*$**git** add .*	*// track any changes to the file*
*$**git** commit -m "comments"*	*// committing to local repo*
*$**git** push origin master*	*// push to remote repo at Github*



After executing the above commands, status of the local source code repo looks like



Github link: https://github.com/Sravya-M/SPE_Calculator

Jenkins

Jenkins – an open source automation server which enables developers around the world to reliably build, test, and deploy their software.

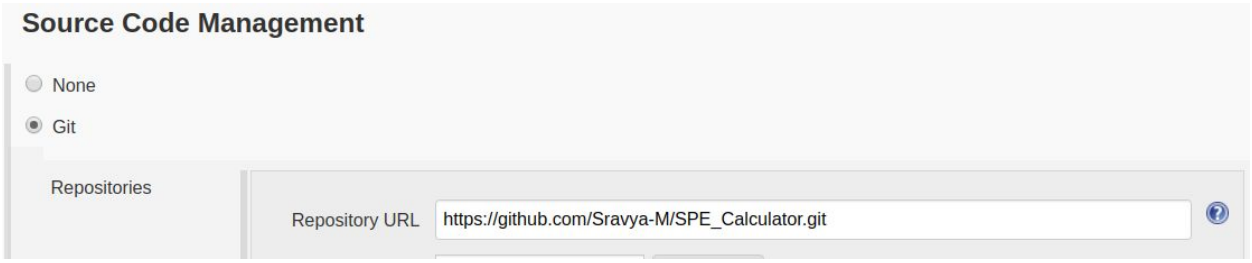
We can use Jenkins for Continuous Integration or continue till Continuous Delivery using the pipeline project. Here, I am showing how to do a freestyle project.

Enter localhost://8080 in the browser to open Jenkins and sign in.

Create New Item

Select Freestyle Project and click on OK

Under Source Code Management Tab, Select Git, Enter GitHub Repository URL,



Under Build Environment, Add Build Step as Ínvoke top-level Maven targets' and specify goals as "clean compile install" in order

Save

Project dashboard is displayed, Click on Build now

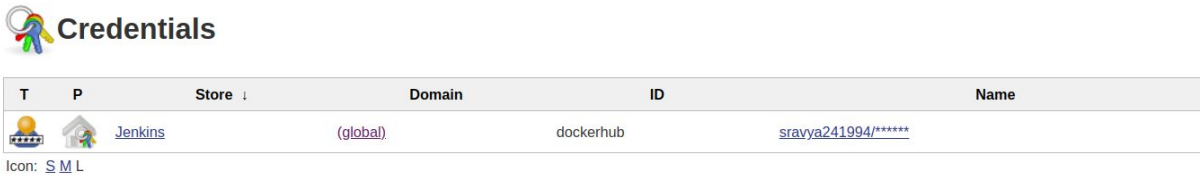
Once the build is finished, click on the #num and check Console log.

Artifact

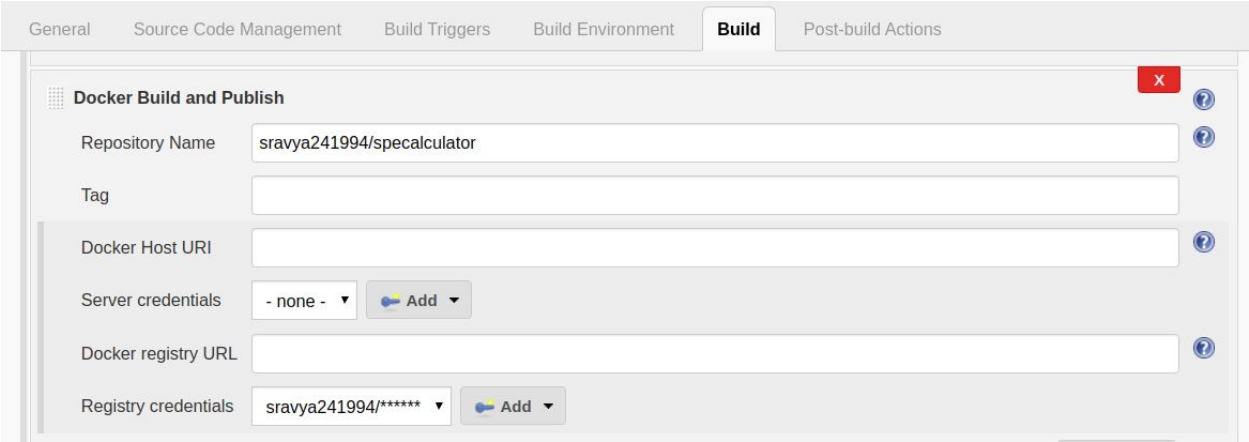
Docker is an open platform for developing, shipping, and running applications. Docker enables you to separate your applications from your infrastructure so you can deliver software quickly. Docker provides the ability to package and run an application in a loosely isolated environment called a container. The isolation and security allow you to run many containers simultaneously on a given host. Containers are lightweight because they don't need the extra load of a hypervisor, but run directly within the host machine's kernel. This means you can run more containers on a given hardware combination than if you were using virtual machines.

Install Docker on the local machine, create a docker hub account.
Create a new repository in docker hub.
Write a Dockerfile in the project root folder by specifying the commands needed to build the image.

We can create a docker image from the local machine and push it to docker hub or add another build step in Jenkins to automatically do it each time.
Jenkins -> Manage Jenkins -> Manage Plugins -> Add docker related plugins and install them
Jenkins -> Credentials -> global -> Add Credentials -> Fill username, password of dockerhub and save
You will see something like below

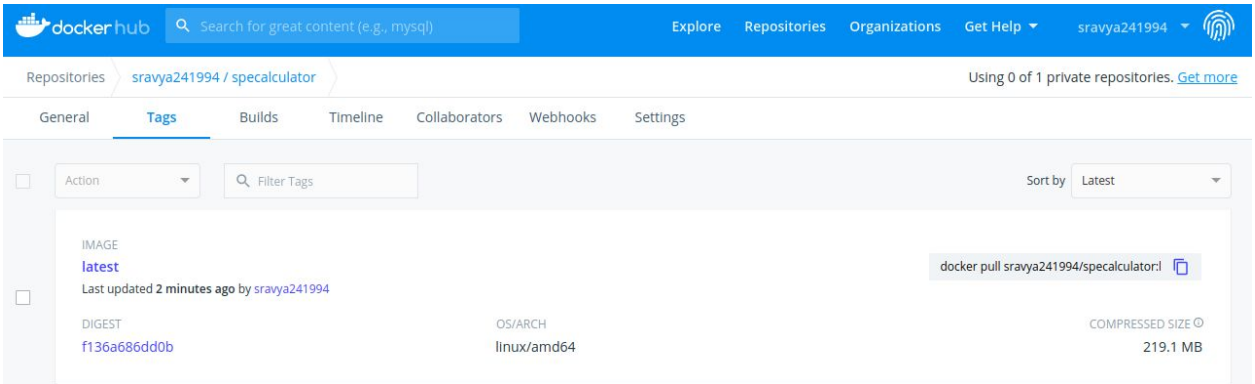


In the freestyle project, we have to add another build step, with Docker Build and Publish, give docker hub repository name, select credentials from Jenkins and Save.



Build Now

Go to dockerhub, we will find a new image created in the repository mentioned.



You can run the image on the local machine too.

*$sudo **docker** login*	*// enter credentials of docker hub*
*$sudo **docker** images*	*// displays images from local and docker hub*
*$sudo **docker** run -it {imagename}*	*// to run the image*
*$sudo **docker** swarm init*	*// initializing swarm*
*$sudo **docker** service ls*	*// display the services*
*$sudo **docker** service create --mode global --name sravya_image sravya241994/specalculator*	*// to create a service with that image*

```
sravya@sravya-Inspiron-15-3567:~/SEM2/SPE/SPE_Calculator$ sudo docker images
REPOSITORY                    TAG                 IMAGE ID            CREATED             SIZE
sravya241994/specalculator    latest              55eedb2d2f68        38 minutes ago      510MB
sravya241994/specalculator    <none>              6ae175795425        16 hours ago        510MB
calci                         latest              4cfb7ee80006        19 hours ago        510MB
openjdk                       8                   6cedfea72886        2 weeks ago         510MB
hello-world                   latest              bf756fb1ae65        4 months ago        13.3kB
```

Run the image:

```
sravya@sravya-Inspiron-15-3567:~/SEM2/SPE/SPE_Calculator$ sudo docker run -it 55eedb2d2f68
CALCULATOR
Option Menu
1. Addition
2. Exit
Enter your choice
1
Enter 2 numbers
First operand:
2
Second operand:
3
5.0

Option Menu
1. Addition
2. Exit
Enter your choice
2

sravya@sravya-Inspiron-15-3567:~/SEM2/SPE/SPE_Calculator$
```

Deploy

Rundeck allows you to run tasks on any number of nodes from a web-based or command-line interface.

To connect my system to the Rundeck node, I have used ssh.

Few commands in doing that:

Find .ssh directory. If it is in home,

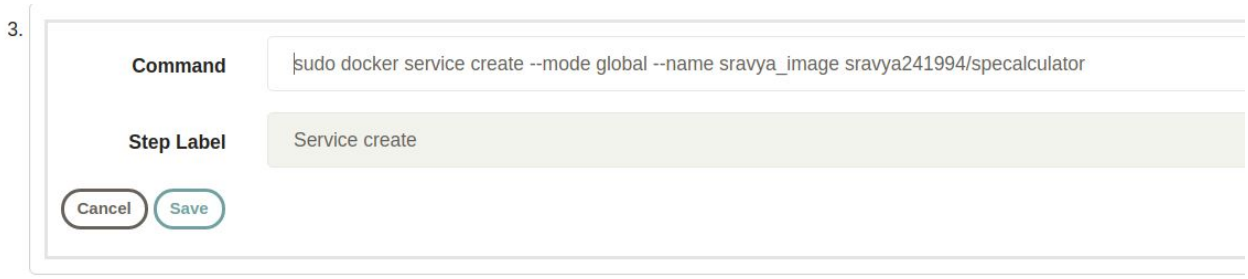
```
~$ls -al                                          // display the files, folders in home
~$sudo ssh-keygen                                 // generates an ssh key value pair
~$cd  /var/lib/rundeck                            // go to rundeck path
$sudo mkdir .ssh                                  // create .ssh directory
~$sudo cp .ssh/* /var/lib/rundeck/.ssh            // copy files in .ssh from home to rundeck
$sudo chown rundeck:rundeck /var/lib/rundeck/.ssh/*
$sudo service rundeckd start
```

Rundeck uses port number 4440. Enter http://localhost:4440 in browser to open rundeck.

Create a new project.

A node will be created by default.

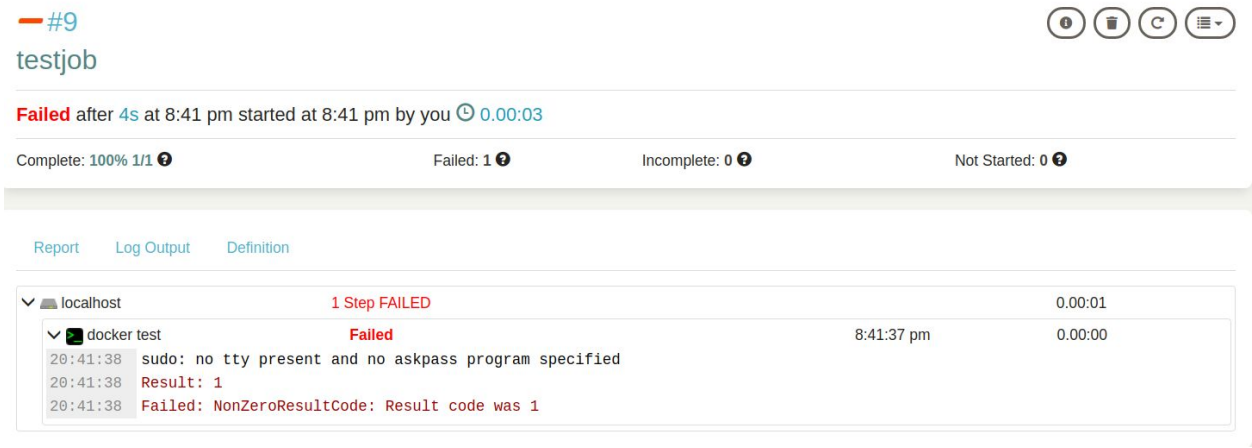
Create a new job in the node with job name, Add commands like following



Save

Click “Run Job Now”

You may get an error like this:



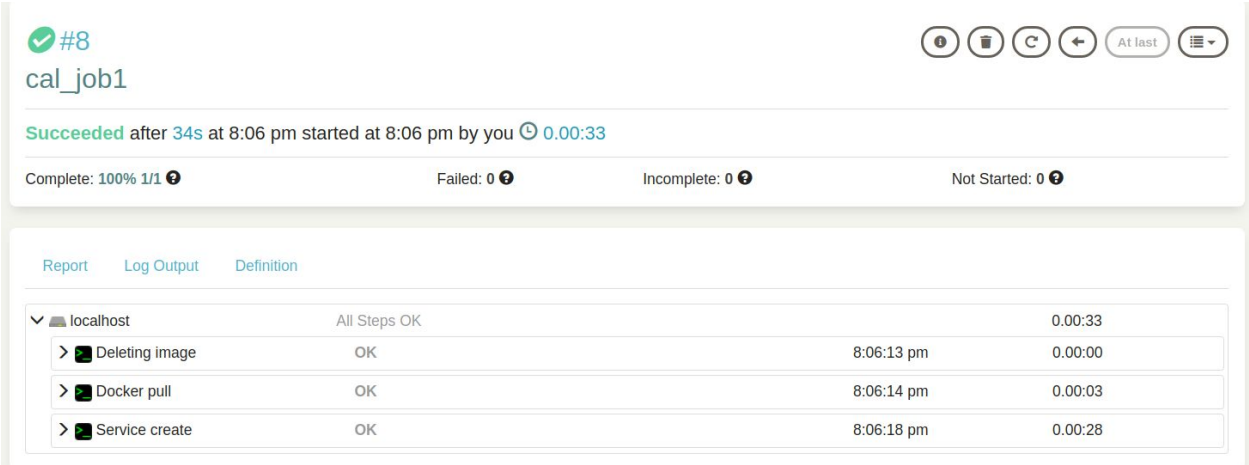
This error is because docker cannot be accessed without sudo or root user permissions.

So, we need to give rundeck the root user permissions.

This can be done using

$sudo gedit /etc/sudoers	// open sudoers file in gedit
Add the below line in the file and Save	
rundeck ALL=(ALL) NOPASSWD: ALL	
$sudo service rundeckd restart	// Restart Rundeck

Now “Run Job Now”



Final step is to integrate all the above steps using Jenkins Pipeline.

CI/CD

This can be achieved using the Jenkins pipeline. Create a New Item in Jenkins with Pipeline Project and Save

Configure the project with Pipeline Script like below

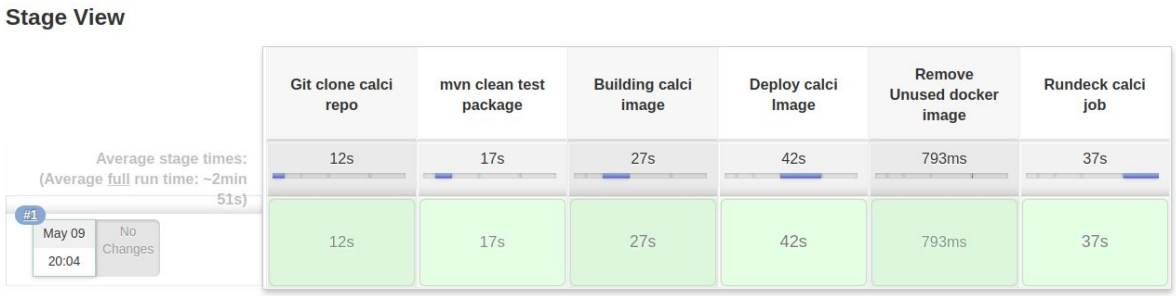
```
pipeline {
  environment {
    registry = "sravya241994/specalculator"
    registryCredential = 'dockerhub'
    dockerImage = ''
    dockerImageLatest = ''
  }
  agent any
  stages {
    stage('Git clone calci repo') {
      steps {
        git 'https://github.com/Sravya-M/SPE_Calculator.git'
      }
    }
    stage('mvn clean test package'){
        steps {
            sh 'mvn clean test package'
```

```
    }
}
stage('Building calci image') {
  steps{
    script {
      dockerImage = docker.build registry + ":$BUILD_NUMBER"
      dockerImageLatest = docker.build registry + ":latest"
    }}}
stage('Deploy calci Image') {
  steps{script { docker.withRegistry( '', registryCredential ) {
        dockerImage.push()
        dockerImageLatest.push()}}}}
stage('Remove Unused docker image') { steps{ sh "docker rmi $registry:$BUILD_NUMBER"}}
stage('Rundeck calci job') {
    steps {
      script {
        step([$class: "RundeckNotifier",
              includeRundeckLogs: true,
              rundeckInstance: "rundeck",
              jobId: 'fe902729-9bb2-4e0f-86d4-b774fb06f06d',
              shouldFailTheBuild: true,
              shouldWaitForRundeckJob: true,
              tailLog: true])
      }}}}}
```

Click “Build Now”

We will get a stage view like this.

After implementing Addition in Calculator:



Now we can add changes to the project incrementally and push to git hub, then build the pipeline in Jenkins.

After incrementally pushing changes to github, building the Jenkins pipeline, the stage view is like below.

Installation Guide:

Java:

// to install jdk11

$sudo apt-get update

$sudo apt-get install default-jdk

$java -version

// to install openjdk8

$ sudo apt-get install openjdk-8-jdk

// to set default jdk as openjdk8

$sudo update-alternatives --config java

```
sravya@sravya-Inspiron-15-3567:~/SEM2/SPE/SPE_Calculator$ sudo update-alternatives --config java
There are 3 choices for the alternative java (providing /usr/bin/java).

  Selection    Path                                            Priority   Status
------------------------------------------------------------
  0            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      auto mode
  1            /usr/lib/jvm/java-11-openjdk-amd64/bin/java      1111      manual mode
* 2            /usr/lib/jvm/java-8-openjdk-amd64/jre/bin/java   1081      manual mode
  3            /usr/lib/jvm/jdk1.8.0_251/bin/java               1         manual mode
```

Enter the number of your choice.

Set JAVA_HOME variable in /etc/environment/ file, append the java home bin folder to PATH.

$sudo gedit /etc/environment

Now, check

$java -version

$javac -version

Maven:

$sudo apt-get update

$sudo apt install maven

$ mvn -version

```
sravya@sravya-Inspiron-15-3567:~/SEM2/SPE/SPE_Calculator$ mvn -version
Apache Maven 3.6.0
Maven home: /usr/share/maven
Java version: 1.8.0_252, vendor: Private Build, runtime: /usr/lib/jvm/java-8-openjdk-amd64/jre
Default locale: en_IN, platform encoding: UTF-8
OS name: "linux", version: "5.3.0-51-generic", arch: "amd64", family: "unix"
```

Git:

$sudo apt-get install git

$git --version

Introduce yourself to Git

git config --global user.name “name”

git config --global user.email "mail id"

Rundeck:

$sudo apt-get update

$wget https://dl.bintray.com/rundeck/rundeck-deb/rundeck_3.0.19.20190327-1.201903272311_all.deb

$sudo dpkg -i rundeck_3.0.19.20190327-1.201903272311_all.deb

$sudo service rundeckd start