Calculator Program using DevOps Tools

Dhruv Patel (Dhruv.Patel@iiitb.org)

Table of Contents

Introduction

Aim is to create simple calculator program to learn, use and experience the DevOpstools and Continuous Integration and Continuous Deployment.

Java (OpenJDK8)

Java is a general-purpose programming language that is class-based, object-oriented, and designed to have as few implementation dependencies as possible.

Installation

```
$ apt-update

$ apt install open-jdk-8-jdk
```

Configure $JAVA_HOME path

Open /etc/environment file

```
$ sudo nano /etc/environment
```

Add following path at the end

```
JAVA_HOME="/usr/lib/jvm/java-8-openjdk-amd64/bin"
```

Restart the system to apply the changes or reload the file to apply the changes to current session.

```
$ source /etc/environment
```

Verify JAVA_HOME path

```
$ echo $JAVA_HOME
```

Git

Git is a distributed version control system, it is a tool to manage project source code history. Git is one of the most widely-used popular version control system in use today.

Installation

```
$ apt update
```

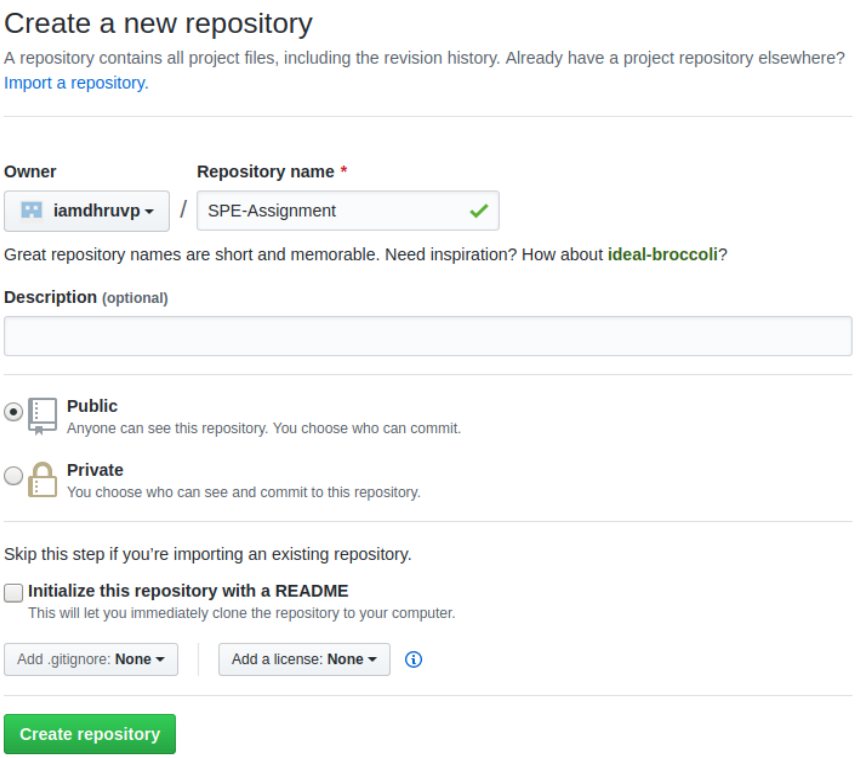
```
$ apt install git
```

Configuration

```
$ git config --global user.name "Your Name"
```

```
$ git config --global user.email "YourEmailID@domain.com”
```

```
$ git --version
```



Maven

Maven is a project development management and comprehension tool. Based on the concept of a project object model: builds, dependency management, documentation creation, site publication, and distribution publication are all controlled from the pom.xml declarative file. Maven can be extended by plugins to utilise a number of other development tools for reporting or the build process.

Installation

```
$ apt-install maven

$ mvn -version
```

Devlopment(Intellij IDE)

Using IntellijIDE, Created a maven project and the java version used is openjdk8.

Added following dependencies for Junit testing in pom.xml.

```
<dependency>
    <groupId>junit</groupId>
    <artifactId>junit</artifactId>
    <version>4.12</version>
    <scope>test</scope>
</dependency>
```

and added following plugin for maven Compiler in pom.xml.

```
<plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-compiler-plugin</artifactId>
    <version>3.7.0</version>
    <configuration>
        <source>1.8</source>
        <target>1.8</target>
    </configuration>
</plugin>
```

Build, compile, test and package the project into JAR

Run maven commands from project root directory.

```
$ mvn clean

$ mvn compile

$ mvn test

$ mvn install
```

Add project into git repository using Intellij IDE

Enabled the version control from VCS in IDE. Committed the changes in the IDE. Specified commit message. Pushed the changes to the git repository created by specifying the url. Pushed to the git repository previously created as the master branch.

Jenkins

Jenkins is an open source automation tool written in Java with plugins built for Continuous Integration purpose. Jenkins is used to build and test your software projects continuously making it easier for developers to integrate changes to the project, and making it easier for users to obtain a fresh build. It also allows you to continuously deliver your software by integrating with a large number of testing and deployment technologies.

Installation

Add the key

```
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins.io.key | apt-key add -
```

When the key is added the system will return a response OK.

Add the repository, update local package index and install

```
$ sh -c 'echo deb http://pkg.jenkins.io/debian-stable binary/ >
/etc/apt/sources.list.d/jenkins.list'

$ apt update

$ apt install jenkins
```

Starting Jenkins

```
$ service jenkins start
```

Check if it is active

```
$ service jenkins status
```

Jenkins runs on port 8080 by default, therefore to use Jenkins, open localhost:8080/

Setting up Jenkins

Use the password provided by following command to unlock jenkins

```
$ cat /var/lib/jenkins/secrets/initialAdminPassword
```

Choose install suggested plugins, configure user and it's done!

Docker

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. With Docker, you can manage your infrastructure in the same ways you manage your applications.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.

Installation

Update your existing list of packages

```
$ sudo apt update
```

Add the Docker repository to APT sources

```
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common

$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -

$ sudo add-apt-repository "deb [arch=amd64]
https://download.docker.com/linux/ubuntu bionic stable"Install Docker

$ sudo apt install docker-ce
```

Check if the status of docker daemon is running to ensure docker is installed

```
$ service docker status
```

Executing Docker commands without sudo

Add your username to the docker group

```
$ sudo usermod -aG docker ${USER}

$ su - ${USER}
```

Confirm that your user is now added to the docker group by typing:

```
$ id -nG
```

Docker Commands

To pull image and if not found on local machine pull from dockerhub

```
$ docker pull <image_name>
```

To list images downloaded to your computer

```
$ docker images
```

To run image in interactive mode

```
$ docker run -i -t --name <Container_Name> <Image_Name>
```

To list active containers

```
$ docker ps -a
```

To commit the changes in a container to a new Docker image

```
$ docker commit -m "What you did to the image" -a "Author Name" container_id
repository/new_image_name
```

Create Docker Image

To run JAR application in container, Java needs to be installed in the container.

Installing java in the base image[ubuntu] by running follwing commands

```
$ apt-get install -y ant

$ apt-get clean

$ rm -rf /var/lib/apt/lists/*

$ rm -rf /var/cache/oracle-jdk8-installer
```

Fix certificate issues, found as of

```
$ apt-get update

$ apt-get install -y ca-certificates-java

$ apt-get clean

$ update-ca-certificates -f

$ rm -rf /var/lib/apt/lists/*

$ rm -rf /var/cache/oracle-jdk8-installer

$ JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64/

$ export JAVA_HOME
```

Now commit the changes in a container to a new Docker image.

Docker File

Create a Docker file to add calculator project’s jar into new image from base image which

has jdk8 installed in it. And add it to git root directory.

```
# Set the base image

FROM iamdhruvp/ubuntu_18.04.3_lts-openjdk_1.8.0_242

# File Author

MAINTAINER Dhruv_Patel

# Set Working Directory

WORKDIR /usr/local

# Copies the files from the source on the host into the container’s set destination

ADD target/calculator-1.0-SNAPSHOT.jar .

# Default container command

ENTRYPOINT ["/usr/bin/java", "-cp", "calculator-1.0-SNAPSHOT.jar",
"com/calculator/Calculator"]
```

Rundeck

Rundeck is an open source automation service with a web console, command line tools and a WebAPI. Rundeck allows you to run tasks on any number of nodes from a web-based or command-line interface. Rundeck also includes other features that make it easy to scale up your automation efforts including: access control, workflow building, scheduling, logging, and integration with external sources for node and option data.

Installation

```
$ echo "deb https://rundeck.bintray.com/rundeck-deb /" | sudo tee -a
/etc/apt/sources.list.d/rundeck.list

$ curl 'https://bintray.com/user/downloadSubjectPublicKey?username=bintray' | sudo
apt-key add -

$ apt update

$ apt install rundeck
```

To start Rundeck:

```
$ service rundeckd start
```

To verify that the service started correctly, tail the logs:

```
$ tail -f /var/log/rundeck/service.log
```

The service is ready once you see something similar to:

Grails application running at http://localhost:4440 in environment: production

Logging in for the first time

Navigate to http://localhost:4440/ in a browser.

Log in with the username admin and password admin

Rundeck is now up and running!

Configure ssh to allow rundeck to access container and deploy

Generate SSH key pair of host machine into/var/lib/rundeck/.ssh/ directory.

```
$ mkdir var/lib/rundeck/.ssh
```

```
$ cd /var/lib/rundeck/.ssh
```

```
$ ssh-keygen (give file name: id_rsa)
```

Allow rundeck to access file

The keys are owned by root user and not rundeck user, which will still give permission denied. Thus, the ownership transfer needs to be done from rundeck user to rundeck group, to allow access of the file.

```
$ chown rundeck:rundeck /var/lib/rundeck/.ssh/*
```

Add public key (id_rsa.pub) of rundeck machine to container

Copy public key of host

```
$ cat ./id_rsa.pub
```

Inside container

```
$ mkdir ./root/ssh
```

```
$ cd root/.ssh
```

Paste copied public key here and save it.

```
$ cat >> authorized_keys
```

Doing all these, we don’t need to provide password authentication or ssh-key-storage-path to node.

Start ssh service

```
$ service ssh start
```

Creating a Project in Rundeck

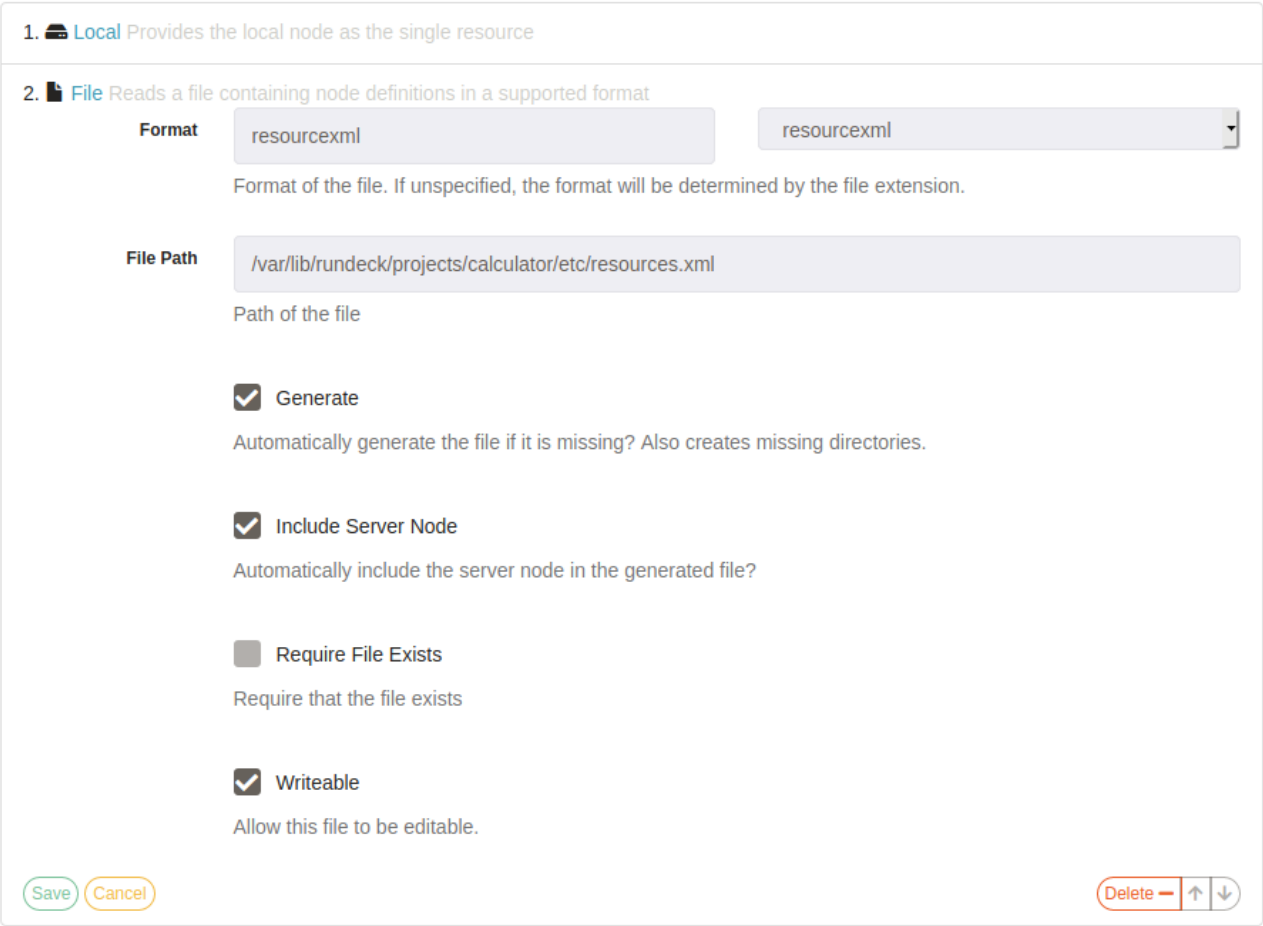
Configuring the project

```
Create project -> Specify name, desc, label -> Press create.
```

Edit Nodes-> Sources (for adding nodes) -> From file -> Specify Format : resourcexml

File path: /var/lib/rundeck/projects/Calculator/etc/resources.xml

Select checkboxes :Generate, Include server node and Writeable ->Save.



File gets generated -> Go in Edit tab -> Insert Node tag with following details and Save:

<node name="client" description="Client node" tags="" hostname="172.17.0.2" osArch="amd64" osFamily="unix" osName="Linux" osVersion="5.3.0-40-generic" username="root" sudo-command-enabled="true" sudo-password-option="option.sudoPassword"/>

Edit Nodes File

/var/lib/rundeck/projects/calculator/etc/resources.xml

Source 2. File Reads a file containing node definitions in a supported format

Format `xml`

Description /var/lib/rundeck/projects/calculator/etc/resources.xml

Soft Wrap

```xml
<?xml version="1.0" encoding="UTF-8"?>

<project>
  <node name="localhost"description="Rundeck server node" tags="" hostname="localhost" osArch="amd64" osFamily="unix" osName="Linux" osVersi

  <node name="client"
  description="Client node"
  tags=""
  hostname="172.17.0.2"
  osArch="amd64"
  osFamily="unix"
  osName="Linux"
  osVersion="5.3.0-40-generic"
  username="root"
  sudo-command-enabled="true"
  sudo-password-option="option.sudoPassword"/>
</project>
```

Test whether nodes were added successfully

Commands tab on left -> Choose node -> Execute command: uname -a -> Run.

Create a job in Rundeck

This job will run inside node(container) which have been configured above

Project -> Jobs -> Job actions -> New Job -> Job Name

Go to workflow tab -> Add steps -> As commands

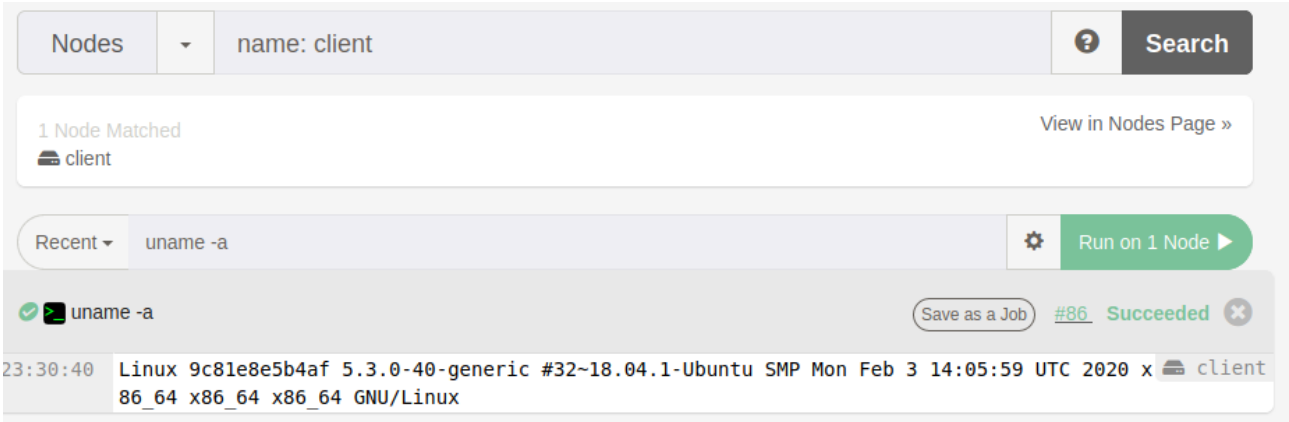
docker rm -f calc_container

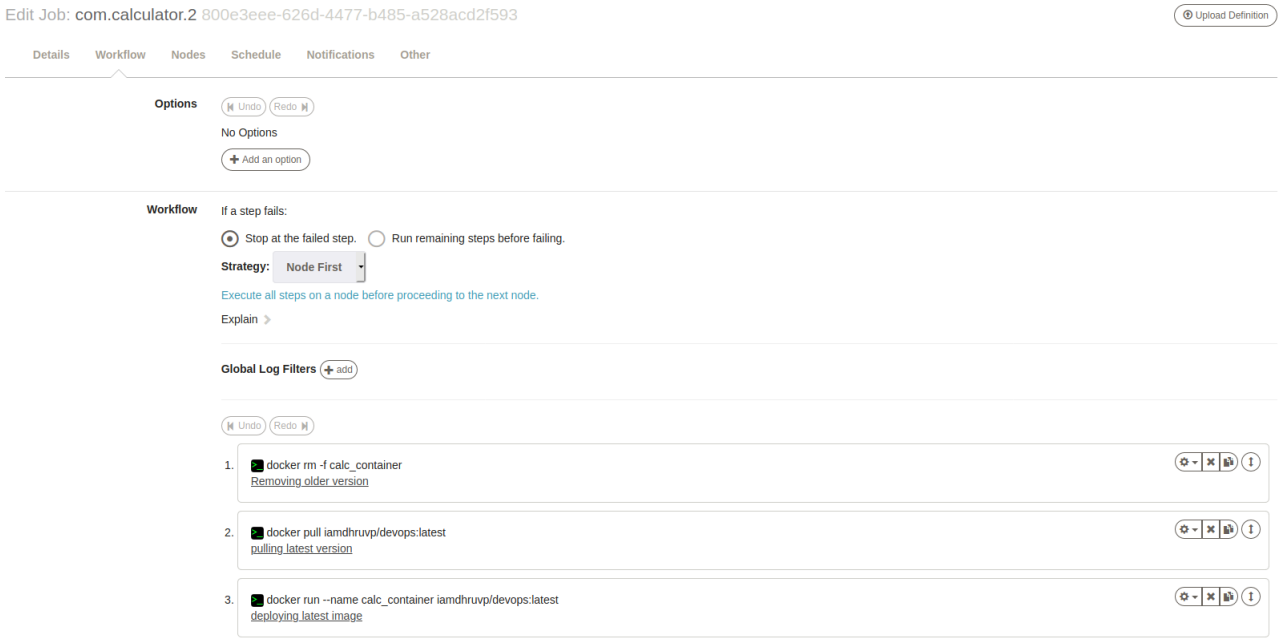
docker pull iamdhruvp/devops:latest

docker run --name calc_container iamdhruvp/devops:latest

Go to Nodes tab -> select Dispatch to Nodes -> Search Node -> Select -> Create.

Note the JobId of the job created. It will be required to trigger from Jenkins.





Creating a pipeline to Integrate SCM, Build Image and Deploy through Rundeck using Jenkins

Configure Jenkins by installing plugins and make some configuration to run project in automated pipeline manner.

Install Plugins

If not-vulnerable

> Manage Jenkins -> Manage Plugins -> Available -> Filter -> Install without restart

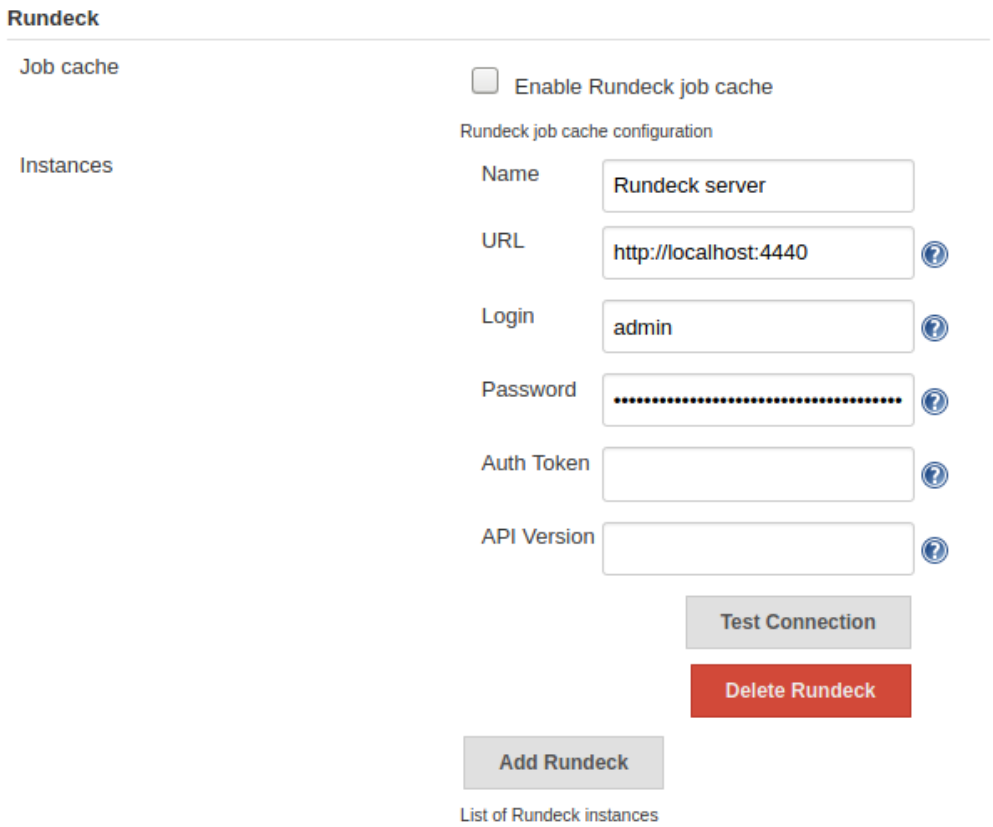
If vulnerable

> Download stable plugin file from https://updates.jenkins-ci.org/download/plugins/
>
> Manage Jenkins -> Manage Plugins -> Advanced -> Upload HPI file -> Install without restart
>
> 1. Git
> 2. GitHub plugin
> 3. Maven Integration plugin
> 4. Docker plugin
> 5. Pipeline
> 6. Rundeck

Configure Systems

Manage Jenkins -> Configure System-> Add Rundeck

- Instances:
  - Provide name
  - Provide URL of Rundeck server (we have, localhost:4440)
  - Login ID
  - Password
- Test connection



Global Tool Configuration for Jenkins

This involves providing path to various binaries to be used for Java, maven building, Git, etc. Considering that you followed the exact instructions, set the following values or if you installed in different directory, you need to adjust accordingly:

Manage Jenkins -> Global Tool Configuration.

- Maven Configuration: default
- JDK : /usr/lib/jvm/java-8-openjdk-amd64
- Git : /usr/bin/git
- Maven : /usr/share/maven

Docker : default

Create a DockerHub repository

Docker Hub is the world's easiest way to create, manage, and deliver your teams' container applications. Here, we can find official images created by companies as well as customized images from different users. We create our own repository.

Creating an account at DockerHub and creating a repository

Signin into https://hub.docker.com/ account create a repository.

Adding the credentials of DockerHub account to Jenkins

Credentials ->System - > Provide username and password of DockerHub -> Provide an ID for this credentials as DockerHub.

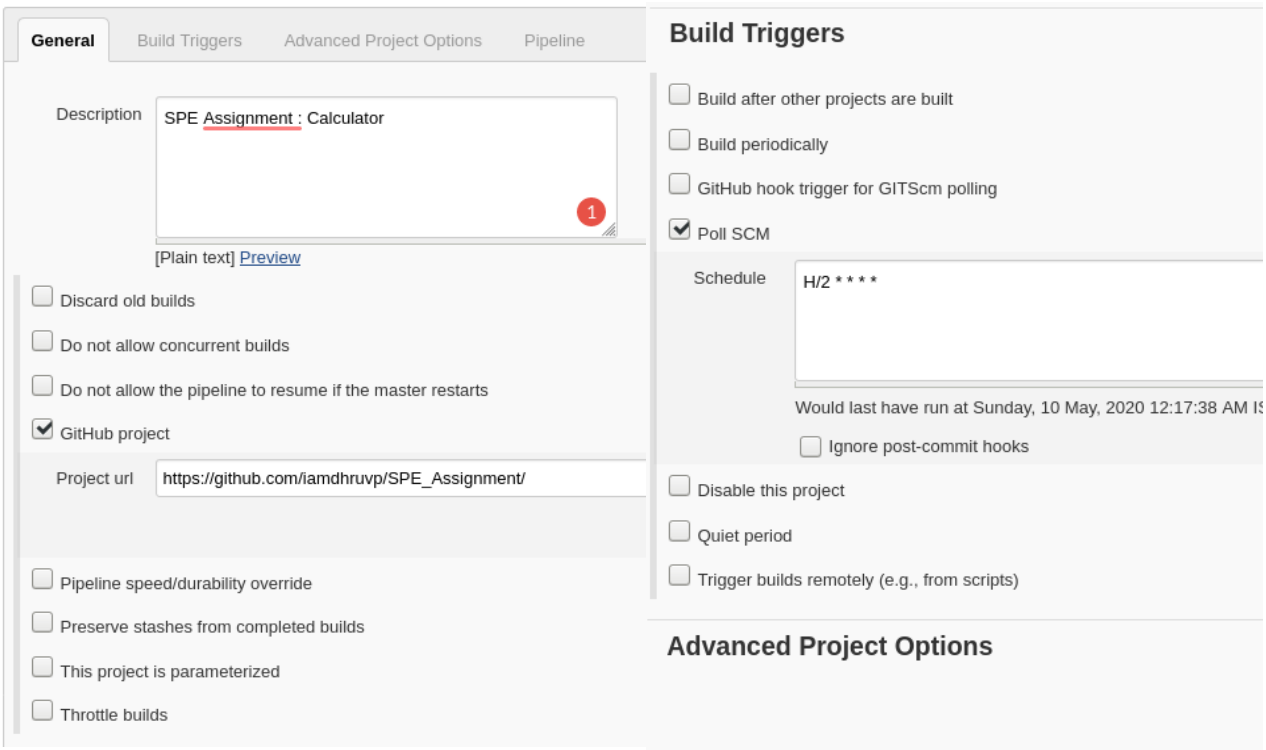
We will be using this later in the pipeline to build and deploy image on our DockerHub repository.

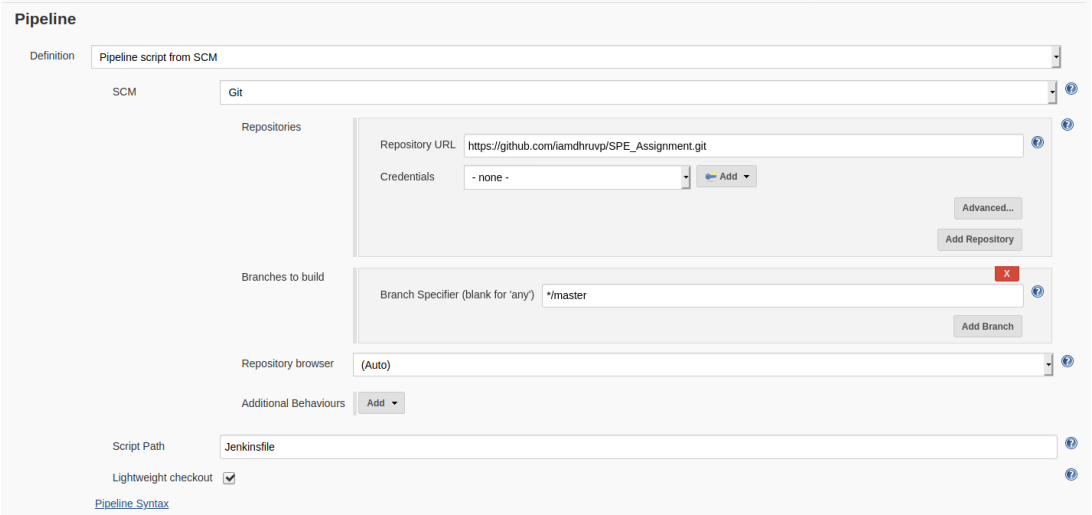
Create Jenkins Pipeline

The aim of the pipeline is to trigger whenever a commit happen on GitHub repository, the build should happen using dependencies defined in pom.xml and it should be tested automatically. If test is successful, then it builds an image from Dockerfile and pushes to DockerHub. If this is success, then Rundeck job is triggered from here, which will deploy the final containerized application to the host node(s).

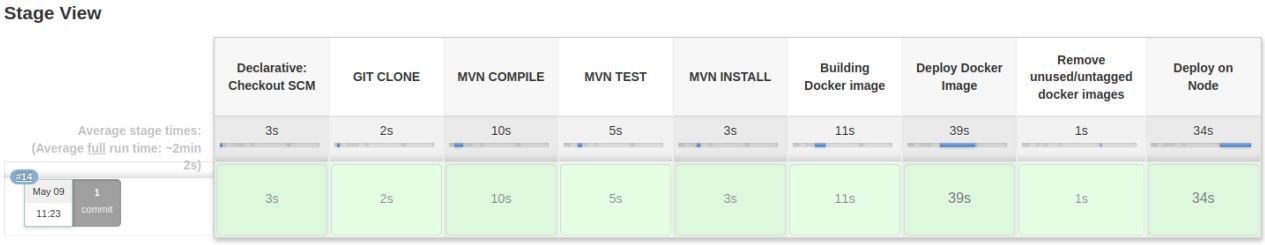
Create Jenkinsfile and it to git.

Jenkins -> New Item -> Enter Item Name-> Pipeline -> OK

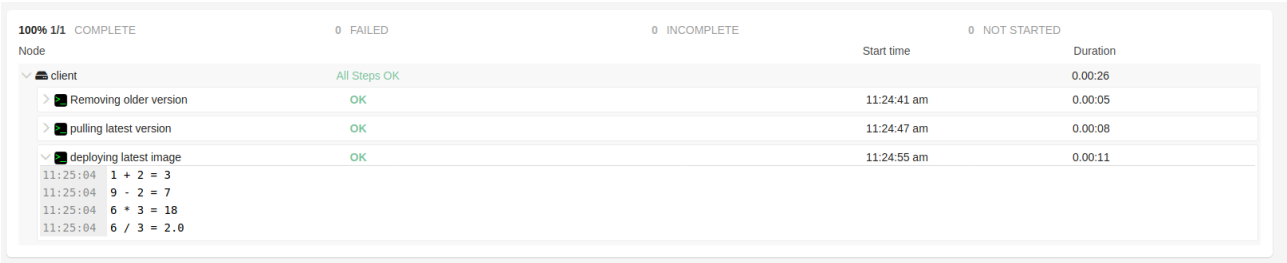




Pipeline Execution



Rundeck Deployment Logs



Add some new features and push it to GitHub. Pipeline will do the rest of the work for you.

Common Errors and Solutions

Localhost:4440/error

check java version

```
$ java --version
```

if it is not java8, switch default java

```
$ sudo update-alternatives –config java
```

Unable to run jenkins http://localhost:8080

Enable Firewall

```
$ sudo ufw allow OpenSSH
```

```
$ sudo ufw enable
```

bash: docker: command not found

install docker inside node(container)

Failed: ConnectionFailure: Connection refused (Connection refused)

start ssh service in node

```
$ service ssh start
```

Failed: ConnectionFailure: No route to host (Host unreachable)

Check container status exited.

```
$ docker ps -a
```

Restart the container

```
$ docker start <container_id/name>
```

Exit/detach container by pressing Ctr+p and Ctr+q to detach the container without exiting it.

Using Docker in Docker

https://jpetazzo.github.io/2015/09/03/do-not-use-docker-in-docker-for-ci/

Links

GitHub

https://github.com/iamdhruvp/SPE_Assignment/

DockerHub

https://hub.docker.com/r/iamdhruvp/devops

References

Maven commands

https://www.journaldev.com/33645/maven-commands-options-cheat-sheet

Installing Docker

https://www.digitalocean.com/community/tutorials/how-to-install-and-use-docker-on-ubuntu-18-04

Create Jenkin pipeline

https://www.edureka.co/community/55640/jenkins-docker-docker-image-jenkins-pipeline-docker-registry