Report on Calculator Application using DevOps

CS 816 Software Production Engineering

Submitted By: Mohit Bansal (MT2019065)

The following tools/frameworks were used for developing the application

- SCM – GitHub
- Building – maven
- Testing - Junit
- Continuous Integration – Jenkins
- Continuous deployment – rundeck

Pre-Requisites

Following softwares commands to configure them and the sources to download then are given below.

Git:

- $ sudo apt-get install git
- $ git config --global user.name "John Doe"
- $ git config --global user.email johndoe@example.com

Java 8 – Rundeck uses openjdk 8, thus we need to configure the global environment variable to this.

- sudo apt-get install openjdk-8-jdk

For shell or bash: export JAVA_HOME=path_to_java_home (Set environment)

Install Docker: Find resources at https://docs.docker.com/engine/install/ubuntu/

Docker Hub: Create a new account and create a new repository of the required name. It can be pushed/pulled using

- $ docker pull/push <DockerHub Username>/<DockerHub Repository Name>

Jenkins can be installed from the official documentation. In *manage-jenkins*, find the plugins below, download and install them. Plugins – *maven, pipeline, rundeck, JUnit, GitHub Pull Request Builder, Build Pipeline, Dashboard & Email extension.*

Install Rundeck using the official documentation. The default port for web-interface is 4440 and default username and password is admin.

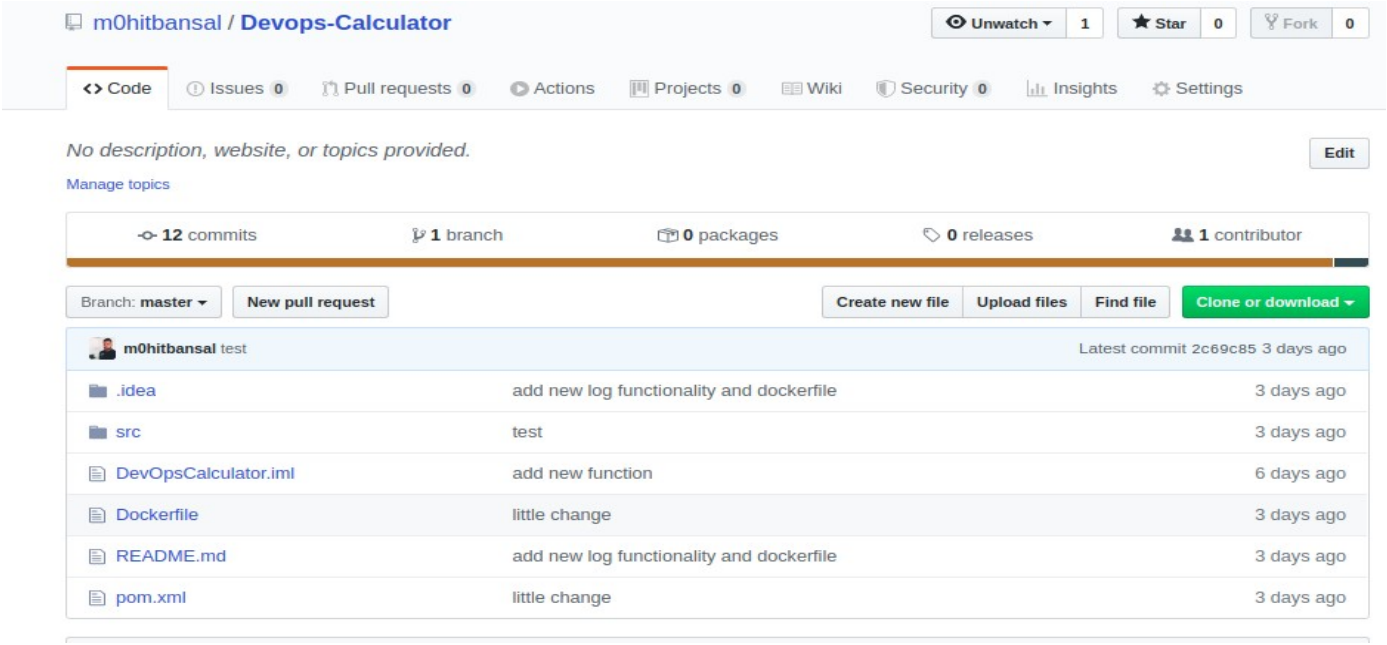
In Jenkins, head to manage-jenkins and configure global settings, there, head to Rundeck and input the Rundeck credentials and test the connection.

Ngrok is used for tunneling and can be installed using $ sudo snap install ngrok

ELK Stack can be downloaded from https://www.elastic.co/downloads/

Source Control Management - 1

Creating new repository on to https://www.github.com. This includes adding repository name anddescription. The repository name should be unique to the signed in user. The similar is done and onesuch repository created for the calculator dev-ops project is https://github.com/m0hitbansal/Devops-Calculator. The SCM handles our code and can be used to connect the input to Jenkins. Other SCM are gitlab, bitbucket.



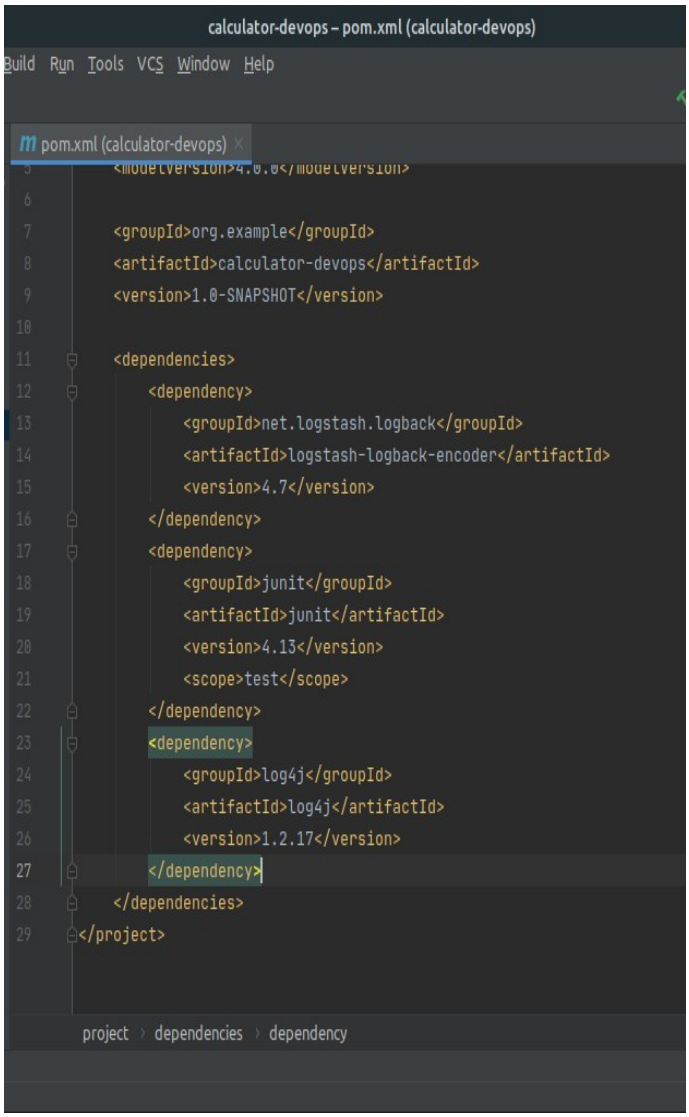
Now if you have already developed the project and want to push to yours create repo, you can initiate the current directory –

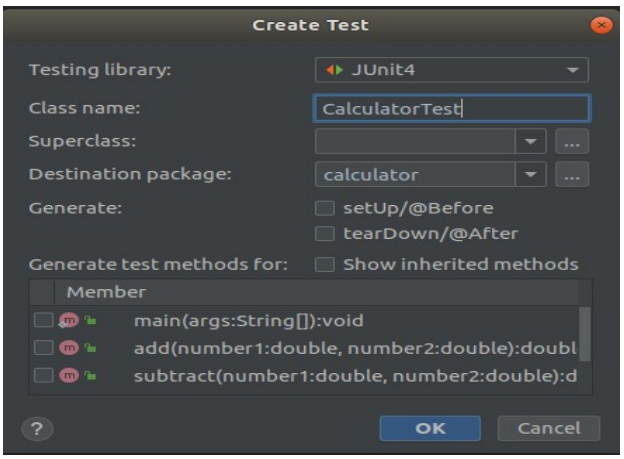
- git init ./
- git remote add origin <git_repo_url>
- git push -f origin <branch_name>

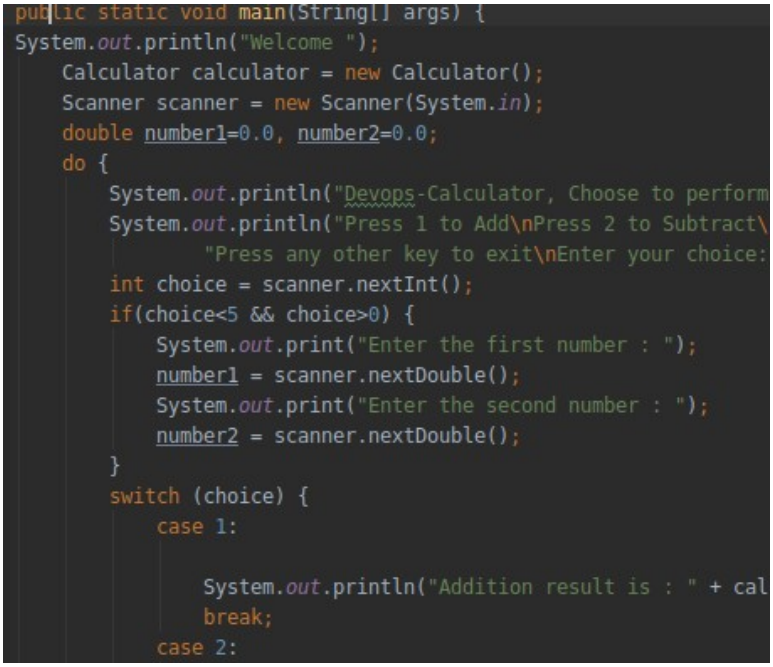
Or if you creating a new project I suggest doing a git clone <git_repo_url>
*Here we are using java 8 since rundeck supports only rundeck, you can configure to java 8 after
installing with sudo update-alternatives --config java
To push the code on to repository follow following commands
git add <changed_files_path>
git commit -m “Commit message name”
git push (only if you are on master branch, otherwise I suggest merging with master first, resolving
conflicts and then do git push)

Development and Software Build

The code was developed on the **IntelliJ IDEA** an IDE provided by Jetbrains.
Initialize a maven project.
You will have a pom.xml file generated. This file is important as it is used to resolve the dependencies required to build the project.
Create a simple class in src/main/java/<packageName>/<Class Name>
Write your simple Calculator program.
Now, test the code you did up till now using **JUnit**. Just hover over the class and have a right click, Show Context Actions -> Create Test a menu pops up, select JUnit 4. This was add a dependency of Junit4 in the pom.xml.
Create some true positive and true negative cases that test the functionality of your project. Find the sample snippet of the junit test cases below and the dependency that gets added in pom.xml.
You may also see some play buttons with each testing method to test that specific method.







Docker

Docker is officially defined as "*A set of platform as a service products that uses OS-level virtualization to deliver software in packages called containers.*"
We use a dockerfile to create the docker image. Now, we need Java 8 already installed in out docker machine along with linux. Now, we create a docker file and shown in the figure below. We also state the commands (CMD) to be executed by the image. This image can be later pushed on to Docker Hub as a public image and be used by other developers too.

Create an account on docker hub and create a repository and the image's name will be <dockerhub username>/<repository name> for example, for the calculator program it was parthendo/devops-calculator
The following commands are used to create a docker image using dockerfile. (keep the Dockerfile in the same directory)

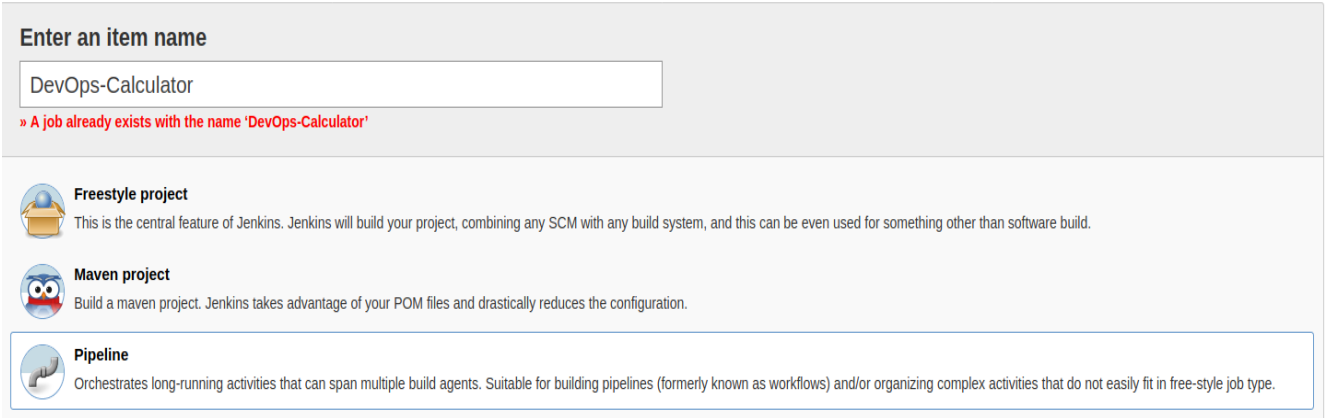
```
$ sudo -s
$ docker build -t m0hitbansal/devopscalculato .
$ docker login
$ docker push m0hitbansal/devopscalculator
$ docker run -i -t m0hitbansal/devopscalculator
```

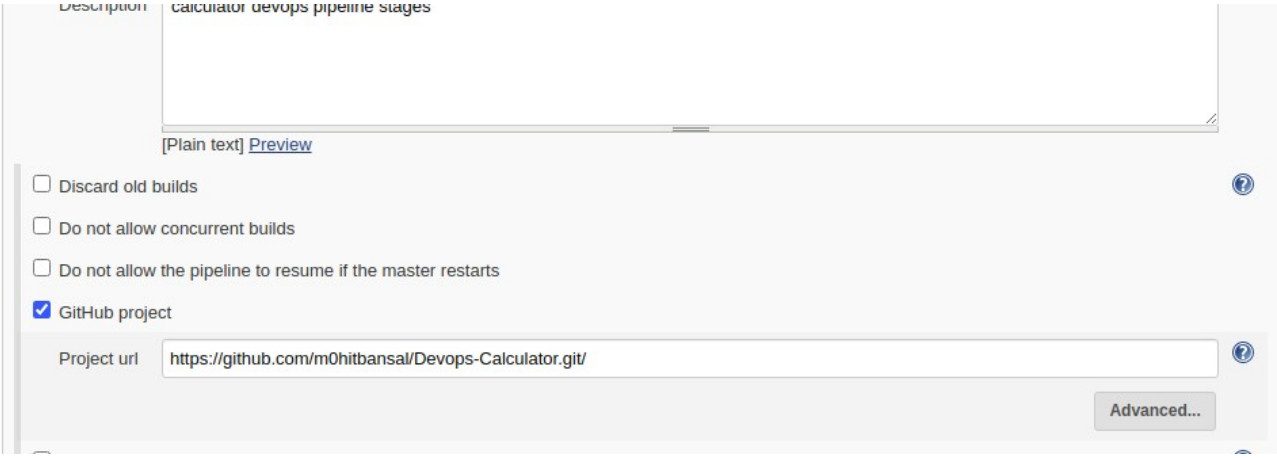
```
FROM openjdk:8
MAINTAINER Mohit Bansal mohit.bansal@iiitb.org
COPY ./target/Devops-Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar ./
WORKDIR ./
CMD ["java", "-jar", "Devops-Calculator-1.0-SNAPSHOT-jar-with-dependencies.jar"]
```

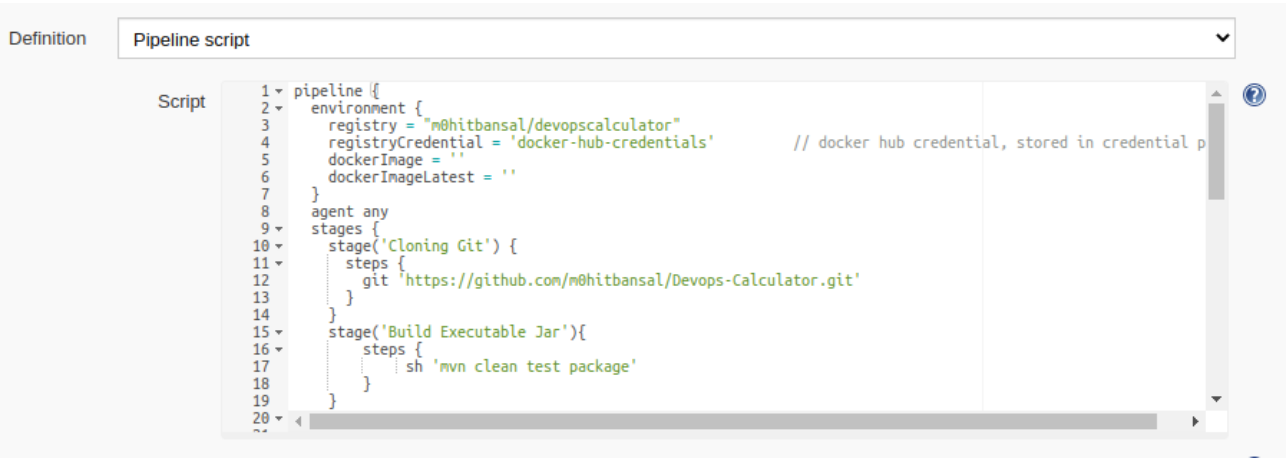
Environment setup - Jenkins

The development process is complete and we have successfully pushed the code on to git, now wesetup Jenkins and other plugins of it.
Make sure to add Jenkins to docker group, so that Jenkins can use docker for build procedure.
sudo usermod –aG docker Jenkins, you can verify it with s
sudo grep jenkins /etc/gshadow
We can now start Jenkins with, if user
sudo systemctl start jenkins, jenkins start at port number 8080, so login on to http://localhost:8080 ontobrowser.
Now we manage the plugins of jenkins under manage plugins in manage jenkins. We download forBuild pipeline plugin, Docker plugin, GitHub, Maven integration plugin, Rundeck plugin.After its done downloading, jenkins will restart and now you

can create a new job for jenkins, enter jenkins job name along with that job is a pipeline. Properties of pipeline is it is script based and each
stage of pipeline script runs one after another. Making it perfect for continuous integration. Properties of continuous integration is it includes SCM, unit testing and integration testing.



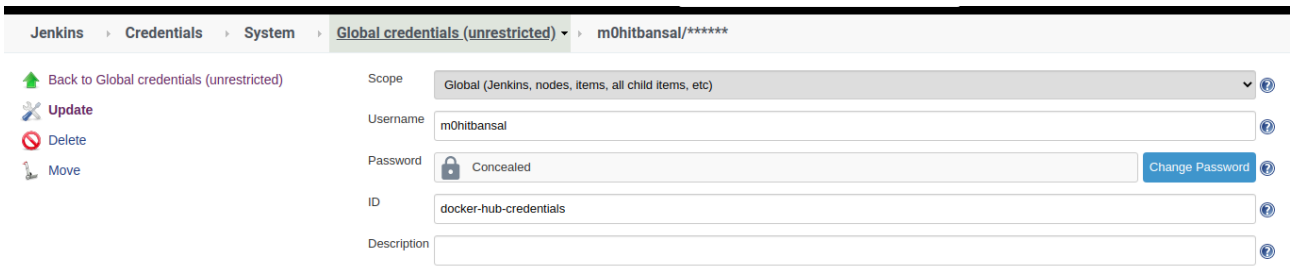


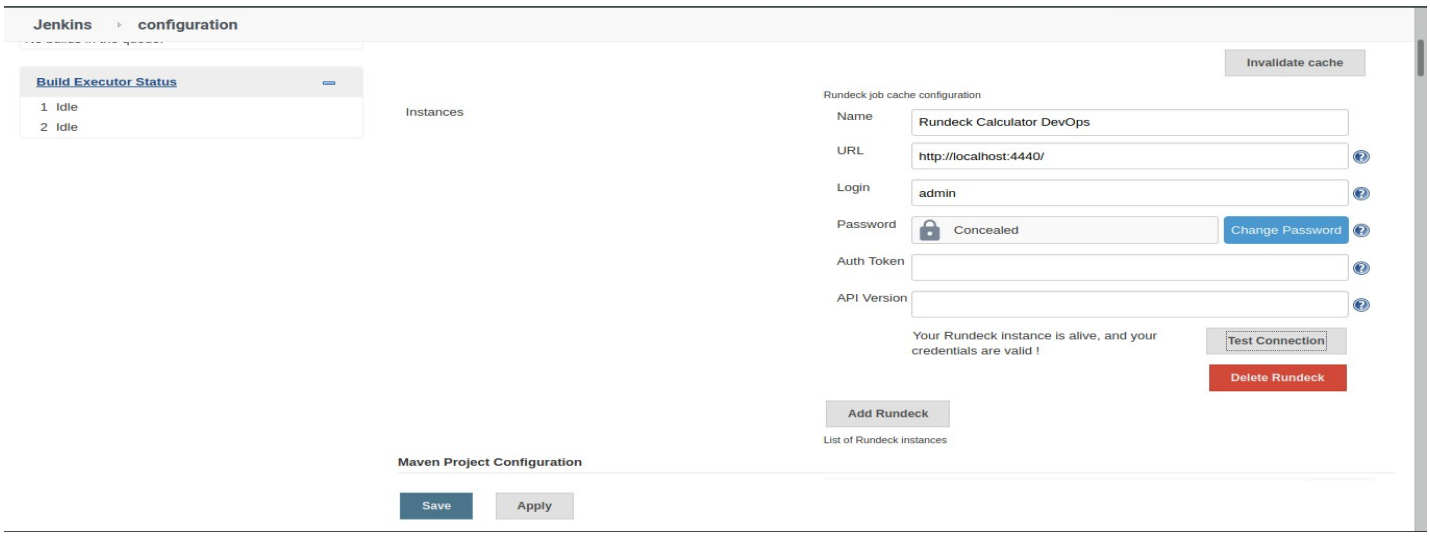


We declare the pipleline script with initially declaring the environment. The above script is to declare the docker hub image. Refer the previous slide, where registry = m0hitbansal/devopscalculator

We need to configure the dockerhub credentials in our Jenkins.
Toggle to Jenkins home
Select *credentials* link present of the menu. Toggle to *global* here. These are the *global* credentials

On the left menu, *add credentials*
Here, add docker hub credentials, i.e. username and password. In the “ID” field, the identity variable is the same used in the *registryCredential* above. i.e. *docker-hub* (refer previous slide)

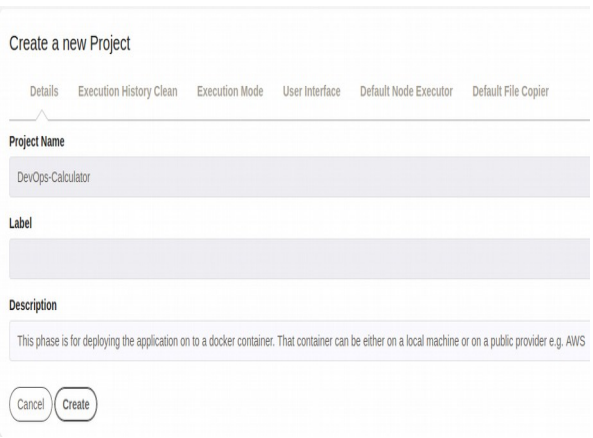


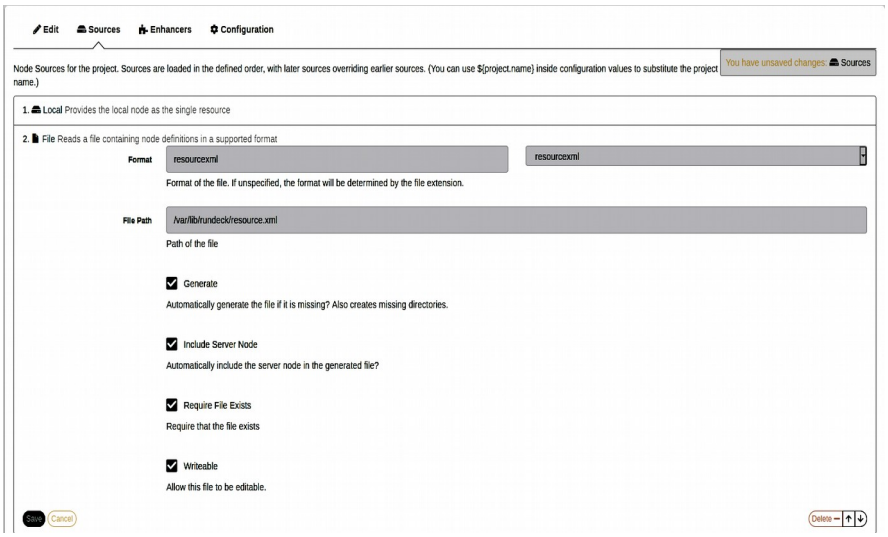


- The second step is to add the GitHub project URL.
- The second part is to build the code. Here, we mention the maven goals we want to achieve. i.e. “test” and “package”
- The next step is where the docker image is build. The name of the image can be generic or declared in global variables present in the previous slide
- The next step is to deploy the image on DockerHub. Here, we use the *registryCredential* we configured in the previous slide and then push the image to the repository <dockerHub Username>/<repository name> i.e.

Deployment: Final Step of the pipeline

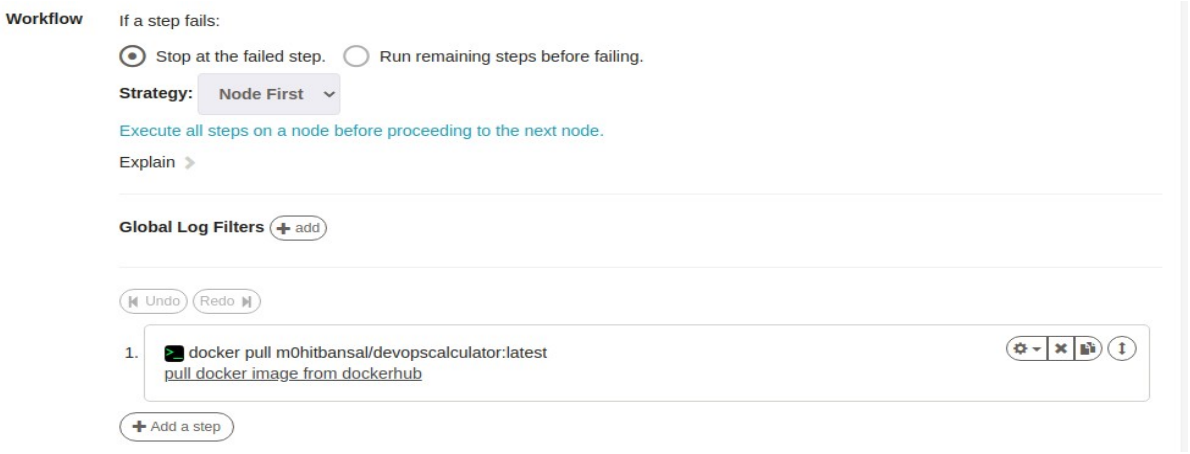
- The final task of the pipeline is deployment i.e. the docker image pushed on Docker Hub in the previous step needs to be pulled on to a docker container. That docker containers need to be accessible for connection i.e. ssh enabled
- We connect this docker container as a *node* in the **Rundeck** project we created and then use Rundeck to host the image on to the container.
- We connect Rundeck to Jenkins to automate the whole process.
- Create a new project. Toggle Edit Nodes and to sources, *add a new Node Source*. Add the resource xml as in next slide.Follow the snapshots for better understanding.





Deployment SetupRundeck: Jobs

A job is an activity that can be done on a node connected to Rundeck. Here, we initialise a node and in the workflow, add a command. Here, we are just testing the connectivity of docker with local machine. So, we view all the images.
After saving the job, run the job and check the output to see the connectivity. At actual task of hosting will be performed by rundeck using jenkins.



If we need to run docker on Rundeck, we need to add Rundeck to the root group.

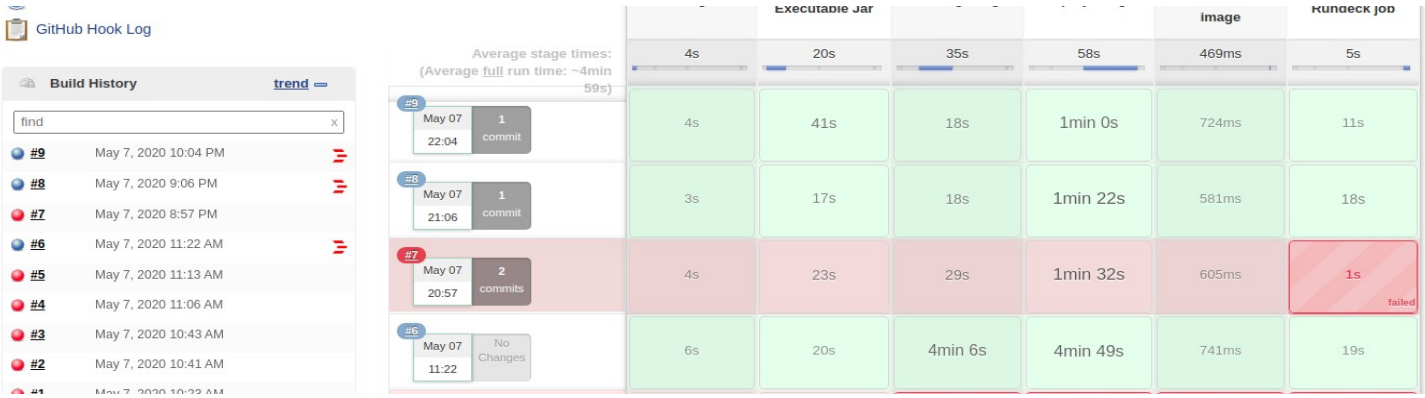
$ usermod -aG docker rundeck

Restart Rundeck i.e.

$ systemctl restart rundeckd

This way, we can see how rundeck belongs to the same group and now we can run *root* commands in rundeck without using *sudo*.

Pipeline in function



After following the steps, we can see the pipeline completely functional, following all the steps of a DevOps lifecycle.

#6 shows there is no change in the GitHub repository.

#7 shows there are 2 commit changes in the GitHub repository but there is a build fail as shown in the reports

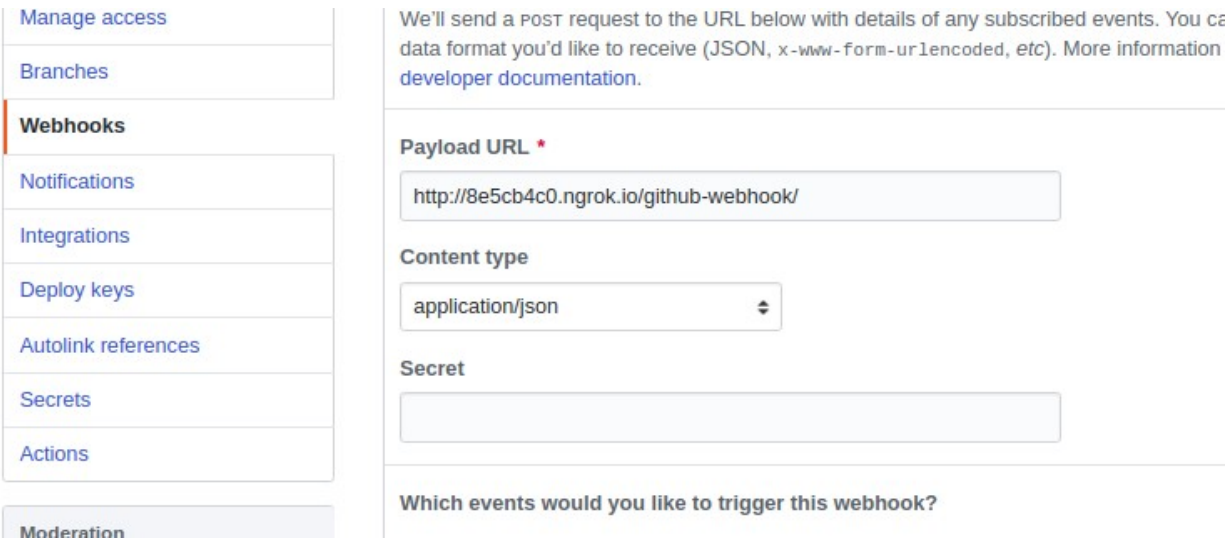
#8 shows there is 1 new commit and all the steps of pipeline are successful.

All the logs are generate step-wise and in a compact manner to view the overall process.

Webhooks

- The next problem in the pipeline was that each time, we manually needed to build the pipeline in Jenkins. We can automate this using webhooks which is a feature provided by GitHub.
- Thus, webhooks is a feature, where after each new commit/change in the repository, it calls the Jenkins for a built i.e.
- Jenkins runs on localhost. To make it available, we need a tunneling application to make local ports publically available and we use **ngrok** for that purpose. Install it using
- $ sudo snap install ngrok
- Run ngrok on the same port on which your Jenkins is running i.e.$ngrok http 8080

```
ngrok by @inconshreveable                                       (Ctrl+C to quit)

Session Status                online
Account                       Mohit Bansal (Plan: Free)
Version                       2.3.35
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://92361dd3.ngrok.io -> http://localhost:8080
Forwarding                    https://92361dd3.ngrok.io -> http://localhost:8080

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```



Open Jenkins and follow the steps to configure authorization credentials.
Add a new Jenkins plugin: GitHub Pull Request Builder
In Jenkins>Manage Jenkins>Configure System, locate GitHub server and input the credentials as in the given image and test connection.On the same configure system, toggle to *GitHub PullRequest Builder* and select the credentials which will automatically apper after you have completed the abovestep.

Monitoring

- Until now, we can see the building and the hosting logs. These can be toggled and viewed on Jenkins. We also maintained standard logs of the running machine. We have displayed the *log4j* library to generate logs. Right now the docker image is running on the local machine, thus the log generated can be stored on the local machine too. Thus, we stored the calculator logs in

  /var/log/ DevOps-Calculator/calculator.log

```
mohit@mohit-300E4Z-300E5Z-300E7Z:~$ tail /var/log/Devops-Calculator/calculator.log
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.subtract(Calculator.java:62) ] - Subtracting two numbers 2.0 and 2.0
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.subtract(Calculator.java:64) ] - Result of subtraction is 0.0
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.subtract(Calculator.java:62) ] - Subtracting two numbers 2.1 and 3.2
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.subtract(Calculator.java:64) ] - Result of subtraction is -1.1
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.multiply(Calculator.java:70) ] - Multiplying two numbers 2.0 and 2.0
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.multiply(Calculator.java:72) ] - Result of multiplication is 4.0
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.multiply(Calculator.java:70) ] - Multiplying two numbers 2.1 and 3.2
[INFO ] 2020-05-07 22:05:34  [ calculator.Calculator.multiply(Calculator.java:72) ] - Result of multiplication is 6.7200000
1
[INFO ] 2020-05-10 13:54:48  [ calculator.Calculator.subtract(Calculator.java:62) ] - Subtracting two numbers 5.0 and 4.0
[INFO ] 2020-05-10 13:54:48  [ calculator.Calculator.subtract(Calculator.java:64) ] - Result of subtraction is 1.0
mohit@mohit-300E4Z-300E5Z-300E7Z:~$
```

The next slide shows how to use *logstash* to build logs and *elastic-search* to read the logs and the *kibana* dashboard

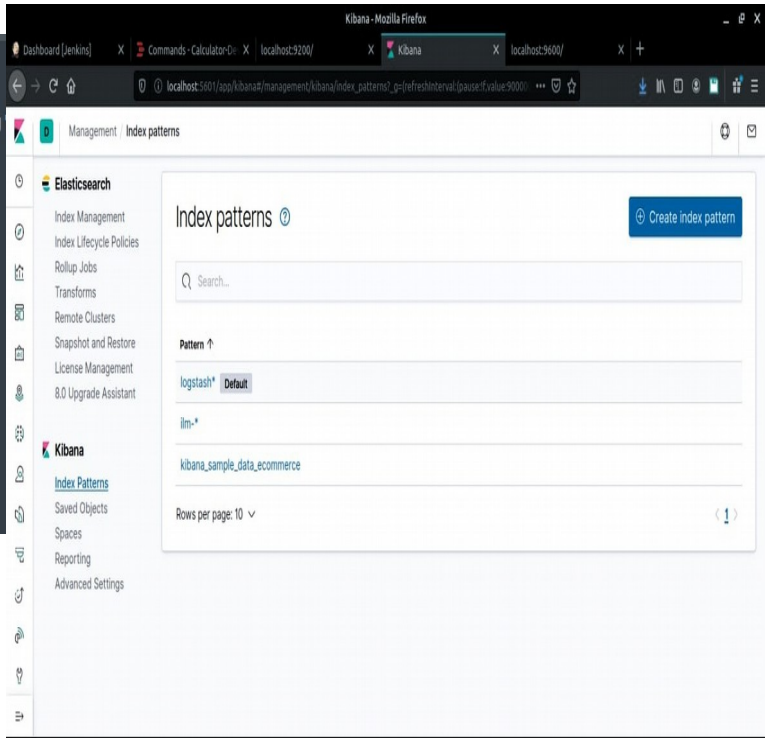
Extract all the zipped files in a folder. To run each software, follow the commands below

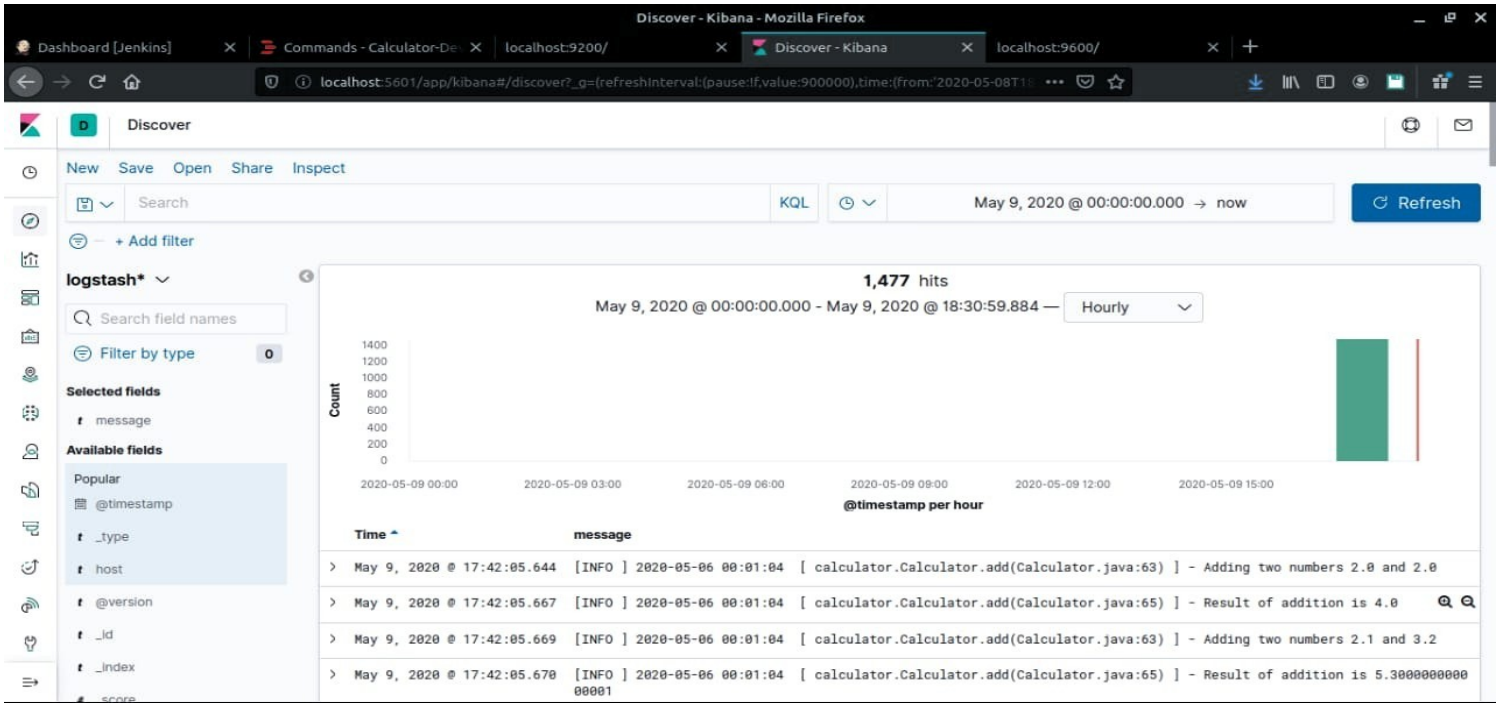
$ ./elasticsearch-7.6.2/bin/elasticsearch

$ ./logstash-7.6.2/bin/logstash -f ./apache.conf

$ ./kibana-7.6.2-linux-x86_64/bin/kibana

```
input
{
    file {
        path => "/var/log/DevOps-Calculator/calculator.log
        type => "logs"
        start_position => "beginning"
    }

}

output
{
    elasticsearch {
        hosts => ["localhost:9200"]
    }

    stdout {
        codec => dots
    }

}
```





Conclusion:

This way, DevOps can be used to deploy a simple CLI based calculator application.

References:

The official documentation can be found below:

- https://www.elastic.co/guide/index.html
- https://www.jenkins.io/doc/
- www.stackoverflow.com
- https://docs.rundeck.com/docs/
- https://help.github.com/en
- https://docs.docker.com/
- https://github.com/Alakazam03