REPORT
ON

Calculator

Harish Sannidhanam
MT2019042

Devops Introduction :-

DevOps is a set of practices that works to automate and integrate the processes between software development and IT teams, so they can build, test, and release software faster and more reliably.

This is a demostration of applying a devops toolchain over a calculator application in an integrated manner.

Workflow Of the Pipeline:

1. Intellij Push code to github
2. SCM polling to jenkins
3. Jenkins Sends data to git
4. Git pulls data from github
5. Workspace Jenkins
6. Build the project by maven
7. Junit
8. Generate war
9. Build Image
10. Trigger rundeck from jenkins
11. Run image using rundeck

Part 1: Building Calculator Application on intelliJ and share on GIT
1. Check for JAVA version in the terminal using the command
$ java -v

I created my application on java version as 1.8 (changed from java vesion 1.11) due to dependencies in later part.

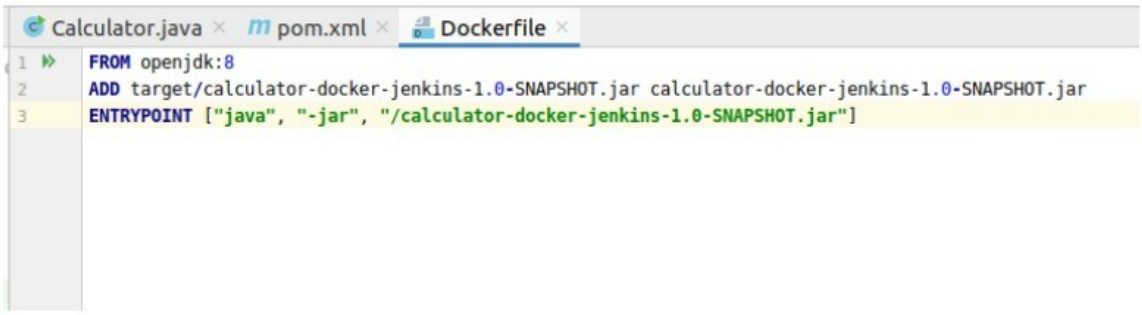
2.. Setting up IntelliJ : IntalliJ is an integrated development environment for creating JAVA applications.
Link : https://www.jetbrains.com/idea/download/

3. Create the project : Apache Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information. It handles dependencies and plugins so we don't need to add jar files for the library explicitly.
Open IntelliJ
create a project
Select Maven
Select SDK as 1.8
Enter Groupid, Artifa

4. Inside Main.Java create a new main java class Calculator.java and enter the main function and create an add() function into it.

5. Also, add test class inside test.java as testclass.java after importing Junit
-->Add dependencies for Junit in the pom.xml

6. Add a docker file in the project folder with the commands to build jar file:



Here, the name of the jar will be same as the generated jar file in the target folder.
--> Add dependencies for the jar file in pom.xml

7. Also add surefire plugin in the pom.xml. Surefire plugin is used during the test phase of
the project.

8. Add maven compiler source properties in the pom.xml.
Build and Run the Project locally.

When everything works perfectly, share the project to git using VCS menu in intelliJ.

9. Now install git in the system using terminal command.
$ sudo apt-get install git
Now go back to intelliJ :
i. Configure Git from settings->version control->Git
ii. Check Path to git executable --> autodetected
iii. if not, then add the path to git directory in the system
VCS--> Import into Version Control --> Share Project on Git

PART 2 : Installations to build PIPELINE on jenkins :

1. Setting up Jenkins :

Jenkins is an open-source automation server that automates the repetitive technical tasks involved in the continuous integration and delivery of software. It is built on Java.

#Use following commands
$ wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add -
$ echo deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list
$ sudo apt-get update
$ sudo apt install jenkins
$ sudo systemctl start jenkins
->Open port for jenkins(localhost:8080 by default)
->Install the suggested plugins
Jenkins is installed!!

Before moving forward in jenkins, lets first install docker and rundeck also.

2. Setting up docker :

Docker is an application that makes it simple and easy to run application processes in a container, which are like virtual machines, only more portable, more resource-friendly, and more dependent on the host operating system.

#Install docker
$ 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 $(lsb_release -cs) stable"
$ sudo apt-get update
$ sudo apt-get install -y docker-ce
Docker is installed!!

(PS: follow “ https://www.digitalocean.com/community/tutorials/how-to-install-and- use-docker-on-ubuntu-16-04 ” to use docker command without using sudo.)

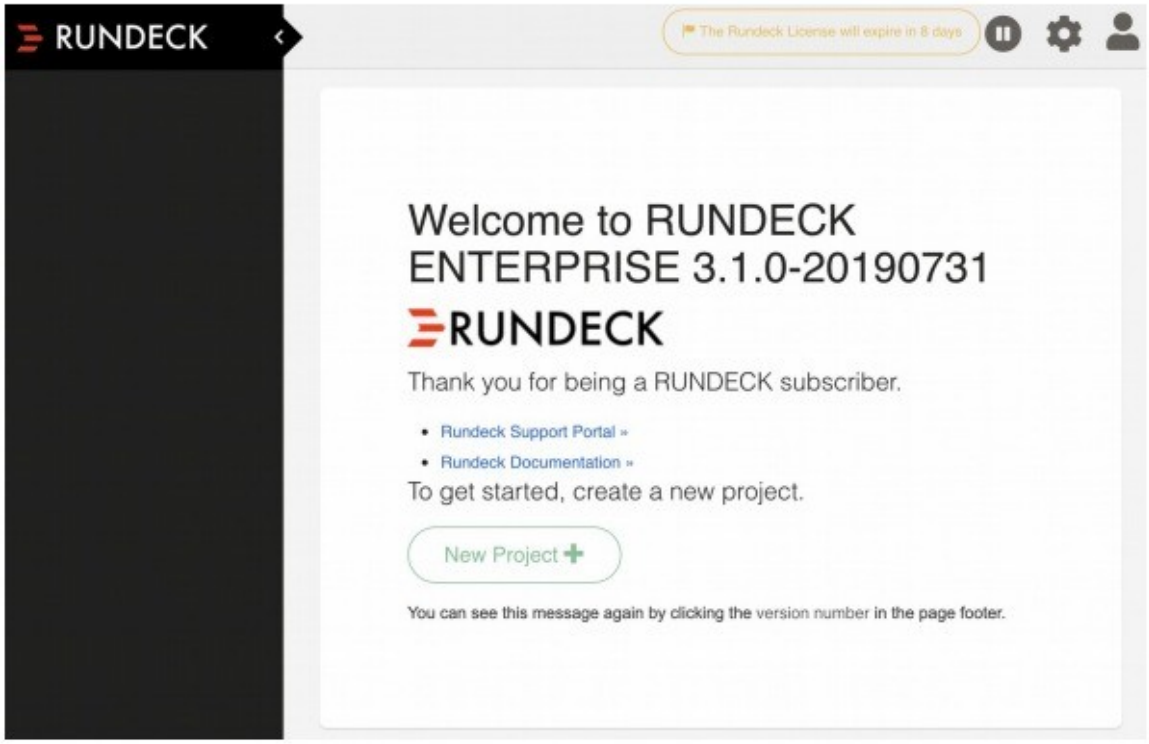
Now go to “hub.docker.com” and create a new account. This will be used when we create job on jenkins to publish docker image.

3. Setting up Rundeck :

Rundeck is automated runbooks that work the way you want to work. It is built for both modern distributed ways of working and centralized legacy environments to automate running of jobs on servers and schedule them to reduce manual effort. Thus, it enables to resolve incidents quicker and improve the productivity of your teams. It also allows assigning users and permissions for jobs.

#Install rundeck
#download rundeck from: http://rundeck.org/download/deb
$ sudo dpkg -i rundeck_3.2.4.20200318-1_all.deb
$ sudo service rundeckd start
#Now open the rundeck using localhost:4440 in browser
#default login is admin/adminAfter login into rundeck you will find a dashboard like this :

Now All the installations (Except for elk) have been done now and it’s time to go back to jenkins



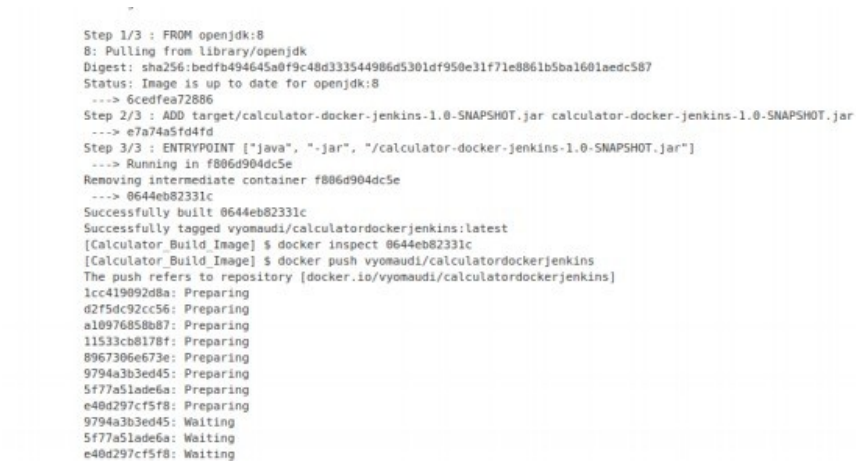
Open Jenkins in the browser and login.

4. Configure Jenkins :

Install the remaining necessary plugins(other than default) such as
1) buildpipeline plugin
2) docker pipeline
3) docker api plugin
4) docker build step plugin
5) Git plugin
6) git api plugin
7) github plugin
8) Githubapiplugin
9) Pipeline buildstep
10)Rundeck plugin

PART 3 : Building the PIPELINE

Now we ll start building items for the pipeline and connect them.
1. Create Item to Build Image.
1. Go to “New item” in the jenkins homepage
2. Select “Freestyle Project”
3. Enter name of the item and save it.
4. In the configurations, check github project and paste the github url of project.
5. In the sourcecode management, select git
6. Enter repository URL
7. Enter credentials for github account.
8. Go to build triggers and select Poll Scm and enter to poll once perhour (H * * * *).
Poll scm checks at every interval whether changes were made in the project.
9. Goto Build envirionment, select “Invoke top-level- Maven Target and type ”Install”.



This will execute maven build to the project.
10. Add a step and select “Docker build and publish”.
11. Enter repository name for the image and dockerhub credentials.

12. Save it and give a build for testing. View :-After Successful build, check in log console :

Now we will trigger rundeck Job to run the image generated. For that first we need to
create a rundeck job.

2. Item to Trigger RUNDECK to run the generated image :
So move to rundeck window:
Before creating a job lets setup the authentication mechanism of our node :
Here we ll run the rundeck job on our local machine server itself, hence we've the
name of the node as rundeck.
• Go to terminal, type
$ ssh rundeck
(For the first time, it will ask to add to known host, add it and leave the pass black if
asked)
•$ ssh-keygen
Public and private keys will be gen in home directory under ./ssh folder
• Gear Icon on top right -> Key Storage -> Add or Upload key -> Select private key ->

Add the private key of rundeck under keys/root -> Save

No, create a new Node in rundeck :
#Adding nodes
• Create project
• Enter label, description, save.
• Project Settings, edit nodes ->Sources for adding nodes -> From file -> Specify: Format as Resourcexml and path as
"/var/lib/rundeck/projects/demo_project/resource.xml"
• Modify resource.xml as shown :•
Now we'll create Steps for the job.
Project -> Jobs -> Job actions -> New Job -> Create steps in the "Workflow" -> As

commands:
• Copy the JOB ID from the JOB details
Go back to jenkins, create a new freestyle item as
“Calculator_Deployment_Rundeck”.
This time leave everything blank and directly move to postbuild actions.
Inside that select
“rundeck”
Enter rundeck instance, user authentications and paste the JOB ID copied from rundeck
job.
Save.

Now we have second Item of our pipeline ready.
We need to build pipeline to see the execution in flow. For that,
• Go to the first Job “Calculator_Build_Image”-> Configure-> post build actions ->
Build a new project -> Select “Calculator_Deployment_Rundeck” and select “Trigger
only if the build is stable”. This will trigger the Second job right after the first.
• Create a new item-> Select Pipeline -> Enter name -> save -> pipeline flow -> give the name of the initial job ie “Calculator_Build Image” and Save.
Now we can see a new tab near “all” as the name of pipeline we created which can be visualised. To test what we have crated till now. Go to jenkins and trigger a Push and commit to the github and wait for the item to get finished. On the completion, open the pipeline created and we can see boxes under green color as the result of successfull completion.
Also you can go to rundeck and check for the job execution log:This output shows that our first build was successfull and docker image was successfully retrieved from the dockerhub.

PART 4 : ELK installation and integration into jenkins :

PART 4 : ELK installation and integration into jenkins :

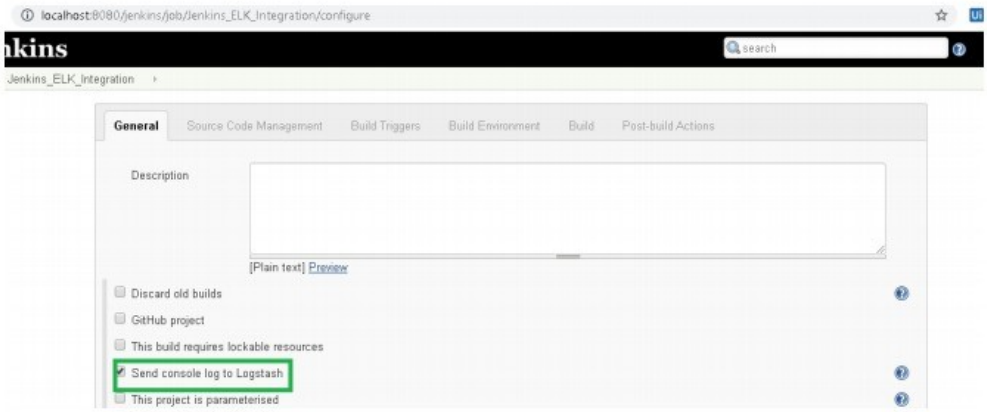
"ELK" is the acronym for three open source projects: Elasticsearch, Logstash, and Kibana. Elasticsearch is a search and analytics engine. Logstash is a server-side data processing pipeline that ingests data from multiple sources simultaneously, transforms it, and then sends it to a "stash" like Elasticsearch.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

"ELK" is the acronym for three open source projects:

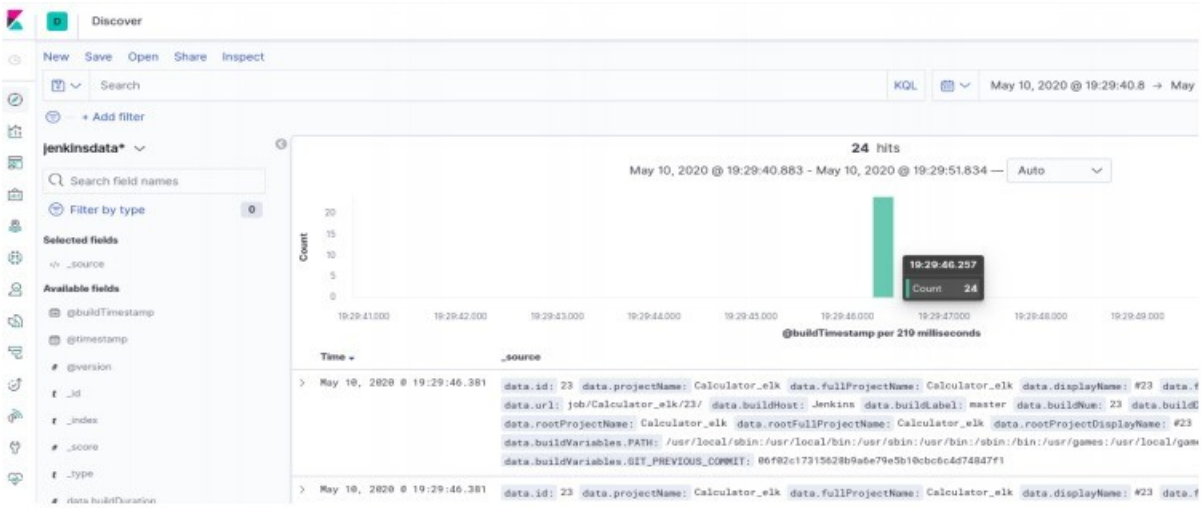

```
{
  "name" : "DESKTOP-KEHKU6I",
  "cluster_name" : "elasticsearch",
  "cluster_uuid" : "dWfKp2JBTTeJDDINi0d_tg",
  "version" : {
    "number" : "7.2.0",
    "build_flavor" : "default",
    "build_type" : "zip",
    "build_hash" : "508c38a",
    "build_date" : "2019-06-20T15:54:18.811730Z",
    "build_snapshot" : false,
    "lucene_version" : "8.0.0",
    "minimum_wire_compatibility_version" : "6.8.0",
    "minimum_index_compatibility_version" : "6.0.0-beta1"
  },
  "tagline" : "You Know, for Search"
}
```


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.

- Go back to manage jenkins jenkins, install plugin “logstash”
- Create new job for ELK

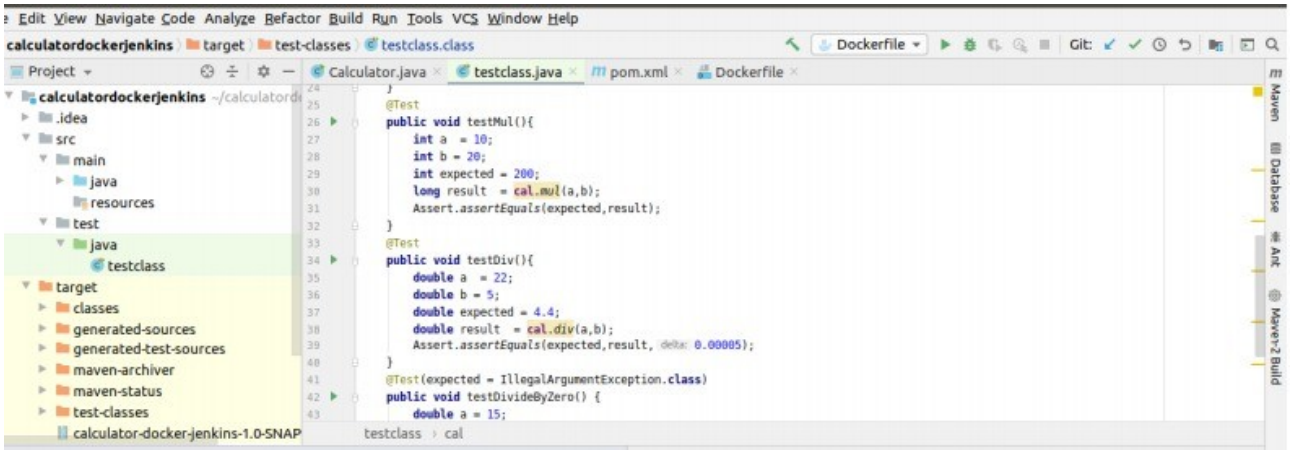


- Source code management as GIT, enter credentials.
- In the build triggers, select Githubhook triggers.
- In the post build action select Send log to lostash.

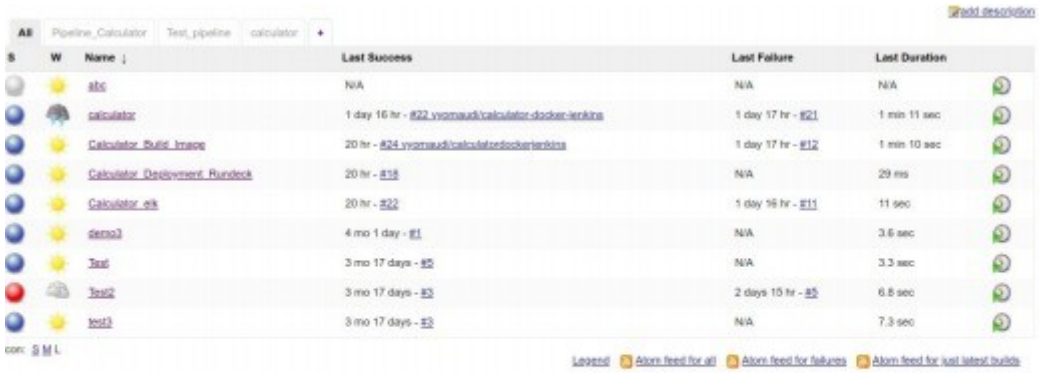


Execution :

1. Push the updated java code to github.



2. Build Is triggered in the Jenkins:



3. Console Output Job 1 ->Calculator_Build_Image:

3. Console Output Job 1 ->Calculator_Build_Image:

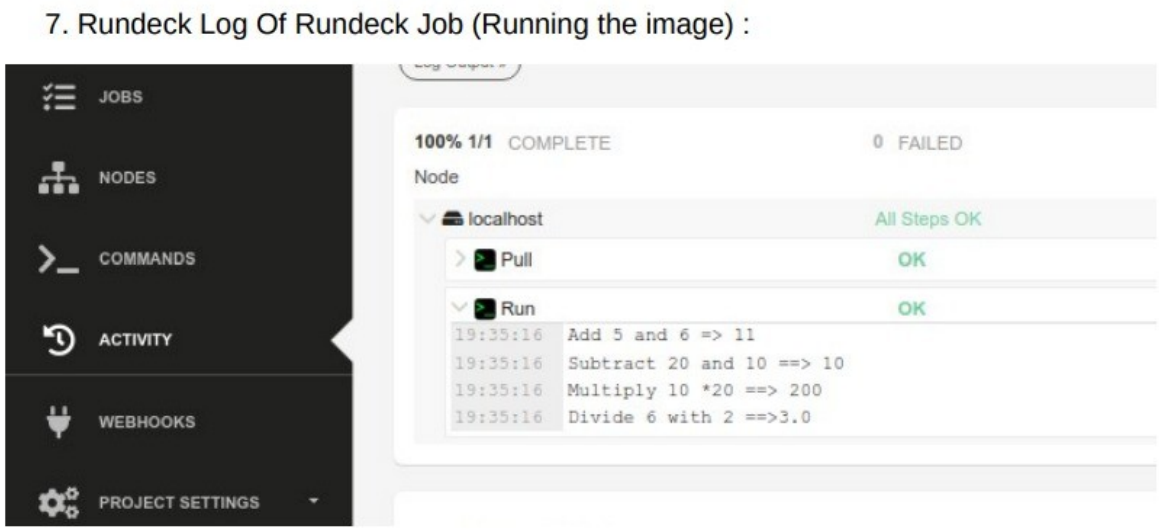
```
 > git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10
Checking out Revision 1c09c3b2c6999e9a85123671b6f71ebace7e4602 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 1c09c3b2c6999e9a85123671b6f71ebace7e4602 # timeout=10
Commit message: "Final Build==>ADD,SUBTRACT,MULTIPLY,DIVIDE"
 > git rev-list --no-walk 06f02c17315628b9a6e79e5b10cbc6c4d74847f1 # timeout=10
[Calculator_Build_Image] $ mvn install
[□[1;34mINFO□[m] Scanning for projects...
[□[1;34mINFO□[m]
[□[1;34mINFO□[m] □[1m-------------< □[0;36mcom.calculator:calculator-docker-jenkins□[0;1m >------------
```

4. Console Output Job 2->Calculator_Deployment_Rundeck:

5. Console Output Job3->Calculator_ELK

6. Pipeline For the three Jobs in Jenkins

7. Rundeck Log Of Rundeck Job (Running the image) :



8. Kibana Logs : The green bar gives the count of data transfered during build.