

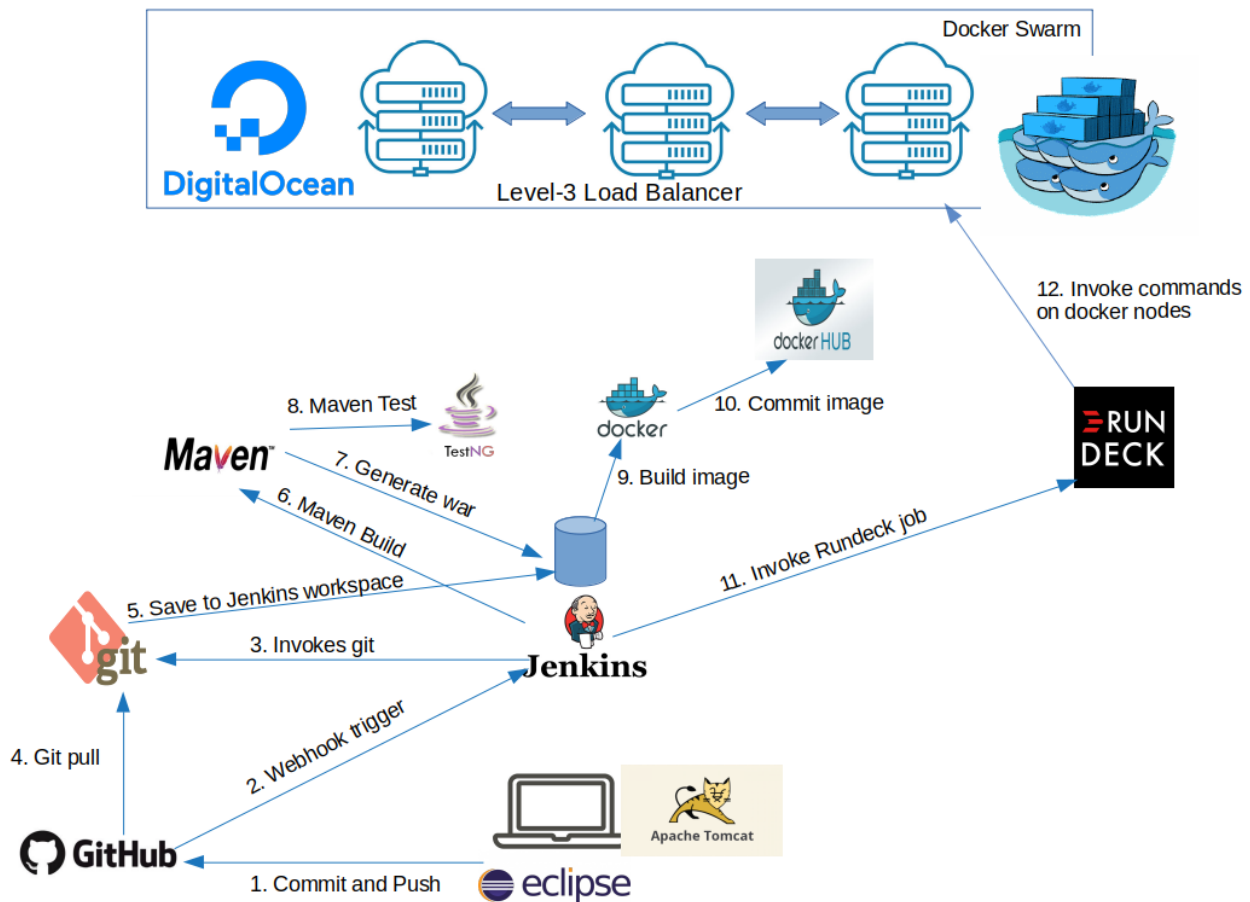
Enabling Continuous Deployment

Vishwesh Vinchurkar - MT2019135

<https://github.com/Vishvin95/Jenkins-Integration>

<https://medium.com/@vishweshvinchurkar/enabling-continuous-deployment-through-jenkins-471adbb68314>

Architecture Diagram



Technology Stack

OS	: Ubuntu 18.04.03 LTS 64-bit
Language	: Java
Java environment	: openjdk version "1.8.0_242"
Application	: Web application
Application Development	: Eclipse IDE 2020
Build tool	: Apache Maven 3.6.0
Testing tool	: TestNG over JUnit and Selenium WebDriver
Deployment	: Apache Tomcat 9.0.33
Source Code Versioning	: Git and GitHub
Docker Registry	: DockerHub
Cloud Server provider	: Digital Ocean
Server operations	: Rundeck
Dummy Domain provider	: Ngrok

Setting Up Java environment

```
~$ sudo apt-get update
~$ sudo apt install openjdk-8-jre-headless
~$ export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
~$ export PATH=$PATH:$JAVA_HOME/bin
~$ java -version
```

Setting Up Eclipse IDE for development

Eclipse IDE is a famous and widely used across industry to develop Enterprise Applications. It is quite flexible and provides plugins that help us to integrate with other technologies, so that we can use them right away from the IDE.

1. Download the tar file eclipse-inst-linux64.tar.gz
2. Unzip the tar file and run eclipse-inst file:

```
~$ cd /home/{<username>
~$ mv downloads/eclipse-inst-linux64.tar.gz .
~$ tar -xvf eclipse-inst-linux64.tar.gz
~$ cd eclipse-installer
~$ ./eclipse-inst
```

This will run the installer. Select Java EE. Accept the terms and it will be installed. It will also ask for the workspace to be created. Leave it default and finish. Note that this will not create an icon of Eclipse in your launcher menu. We need to create it manually:

```
~$ cd /home/{username}/.local/share/applications/
~$ touch eclipse.desktop
```

```
# Write following content in eclipse
[Desktop Entry]
Version=1.0
Name=Eclipse
Comment=Java IDE
Type=Application
Categories=Development;IDE;
Exec=/home/{username}/eclipse/jee-2020-03/eclipse/eclipse
Terminal=false
StartupNotify=true
Icon=/home/{username}/eclipse/jee-2020-03/eclipse/icon.xpm
Name[en_US]=Eclipse~$ chmod a+x eclipse.desktop#
```

Reboot your system and you will find eclipse in launcher menu
Add it to your favorites

Creating a web application using Maven as Dependencies handler

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. We will use it to handle dependencies and plugins, so that the versions of the libraries which we are using are automatically taken care of and we don't need to explicitly add jars.

1. Eclipse -> File -> New -> Maven Project -> Next -> From catalog, select Internal -> maven-webapp-archtype.
2. Provide Group Id as package -> Provide Artifact Id as Project name.
3. Finish.

4. If this gives an error, "javax.servlet.http.HttpServlet was not found on build path", it is due to missing dependency in pom.xml file. Add this to dependency:<!-- https://mvnrepository.com/artifact/javax.servlet/servlet-api -->

```
<dependency>
  <groupId>javax.servlet</groupId>
  <artifactId>javax.servlet-api</artifactId>
  <version>3.1.0</version>
  <scope>provided</scope>
</dependency>
```

An archetype refers to architecture type of the application. This enables building the default architecture i.e. how files are organized, dependencies to be included.

Git and GitHub

Git is a distributed version control system based on SHA1 hashes, that allows us to move back and forth between the different versions of project source code. It can be local to our file system. But, when it comes to collaboration of the members of the development team, we need a common repository that could be shared. GitHub provides us online repository hosting service, along with several additional features.

```
# Installing Git
~$ sudo apt-get update
~$ sudo apt-get install git
~$ git --version
```

Basic Git Commands

```
~$ git init           : Initialize local git repository
~$ git add .          : Adding files to staging area
~$ git status         : Tells status of files in repo
~$ git commit -m "First Commit" : Commits the changes to repository
~$ git checkout <branch> : Checks out particular branch
~$ git push           : Pushing changes to remote(GitHub)
```

Adding Git Plugin to Eclipse

Open Eclipse -> Help -> Install New Software -> Paste following URL in Work With: <http://download.eclipse.org/egit/updates> -> Enter -> Select Git Integration for Eclipse -> Finish. After Git plugin is installed, you will get a menu item **Team** in right click of the project, which has options for git.

Converting our project as Git enabled repository

1. Eclipse -> Right click on project -> Team -> Share Project.
2. It provides two options for creating repository:
 - In parent directory of project
/home/{username}/eclipse-workspace/{projectname}.git
 - In the common git repository folder
/home/{username}/git/{projectname}.git3.

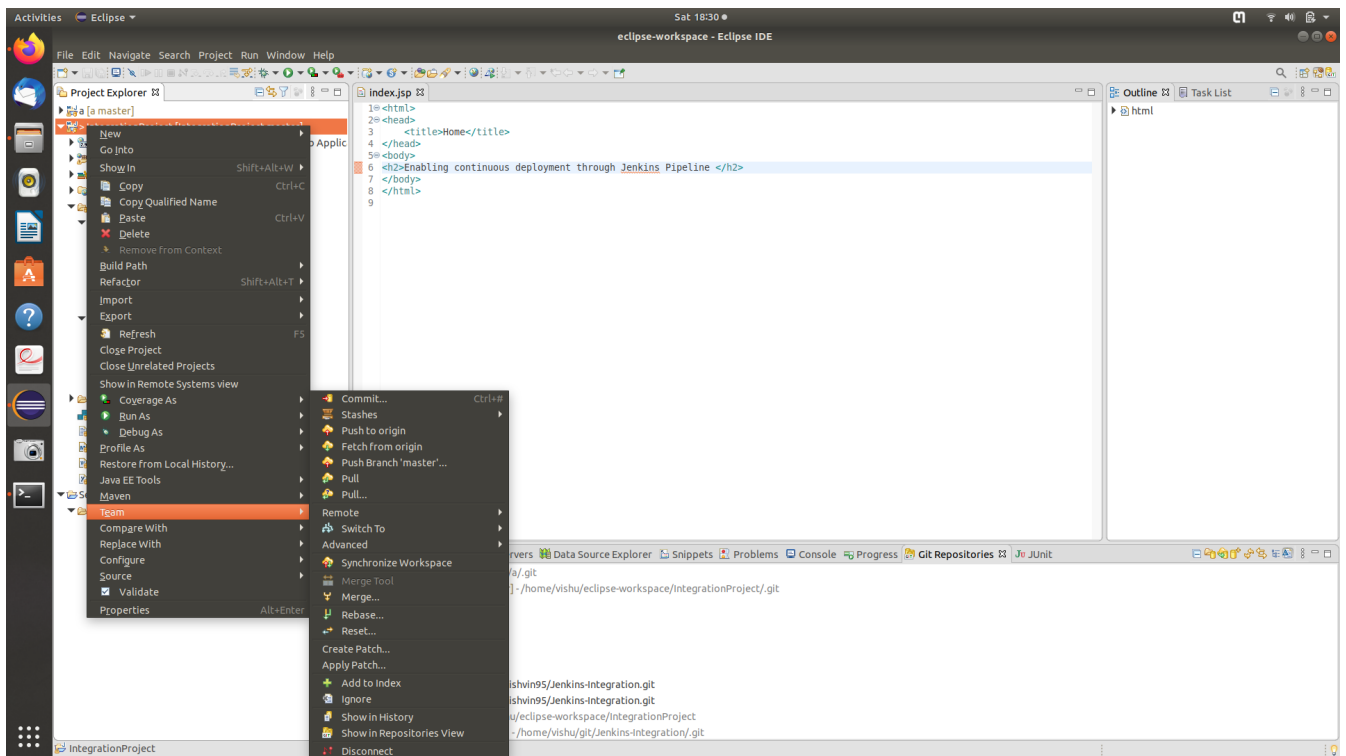
It is good idea, if we use second option, to have all projects at one place, which is second option. But, here I will show you both the options.

Parent folder

- Select the checkbox, Use or create repository in parent folder
- Create repository -> Select checkbox -> Finish.

Common Git folder

- DO NOT select the checkbox.
- Press Create...
- Replace the word repository by {project-name}.
- Finish.



Creating a GitHub repository

Using GitHub, we can create both public and private repositories. It also provides various features like fork, pull requests and issues, which are very often used in Open Source contributions.

1. Create account on <https://github.com/> and sign in.
2. On left repository pane, Click New and provide repository name.
3. DO NOT initialize it with README.md. We will manage this from Eclipse.

Linking Eclipse project repository with GitHub repository

We require this, so that whenever we make changes to project in Eclipse, we could push this to remote repository to be shared among other team members and expose this, such that it is accessible to other servers.

1. To view the repository tab:
Eclipse -> Window -> View -> Git Repository.

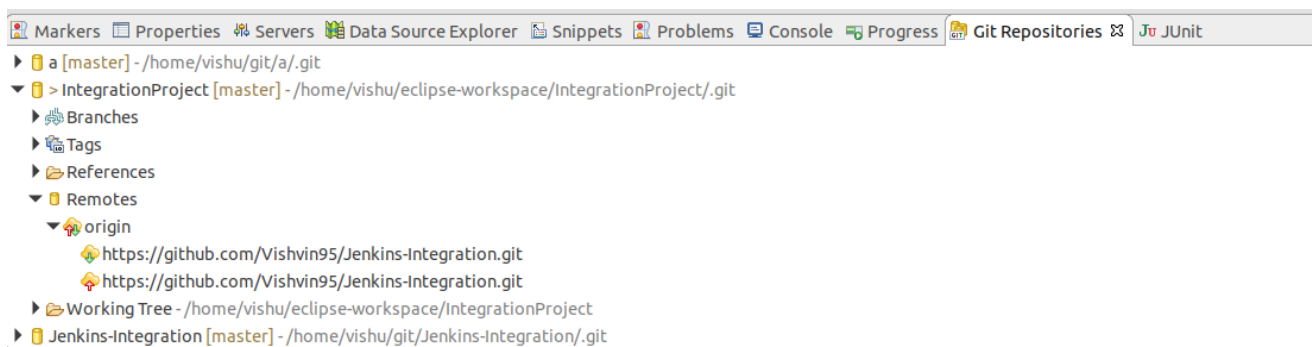
2. To add remote GitHub repository, inside repository view. In this tab, we see all repositories configured in Eclipse. To link remote repository to our local repository, based on the earlier selection, parent folder or common git folder, we have two directions:

Parent folder:

Local Repository -> Remote -> Right click -> Copy URI -> Add GitHub Repository URL -> Username -> Password for authentication -> Finish.

Common Git folder:

Local Repository -> Remote -> Create Remote -> Remote Name -> Configure push -> URI -> Change -> Add GitHub Repository URL -> Username -> Password for authentication -> Finish.



Now, whenever you make changes, simply right click on project, Go to Team menu and you will get various options for git.

Before that, we maintain .gitignore file, that could be found under working tree in repositories view. Add following contents to .gitignore file:

```
/target/  
/.classpath  
/.settings/  
/.project  
/home/
```

Right click on project -> Team -> Commit -> Type a commit message -> Select files to commit from bottom -> Commit and push -> Refresh GitHub Repository.

Apache Tomcat Setup

Apache Tomcat provides us with a web container which is an open source implementation of the Java Servlet, JavaServer Pages, Java Expression Language and Java WebSocket technologies, to allow Java applications to be deployed inside it. Let us set this up, so that we can run the web application inside it.

1. Go to <https://tomcat.apache.org/download-90.cgi>
2. Download Binary Distribution: Core tar.gz file.
3. Copy this and extract this file in /home/{username}.
4. Inside conf folder, go to server.xml and change the Connector port to 8000:
<Connector port="8000" protocol="HTTP/1.1"
 connectionTimeout="20000"
 redirectPort="8443" />** We do this, because we will use Jenkins on port 8080.

Adding Tomcat Server in Eclipse

1. File -> New -> Other -> Filter Server -> Filter Tomcat -> Select Apache Tomcat 9.0
2. Provide the path: /home/{username}/apache-tomcat-9.0.33
3. Configure Apps to run on this, if you want.
4. Finish.

Running our application on local machine

For testing our local changes, we run the application on our local machine in Apache Tomcat Server.

Right Click on Project in Explorer -> Run As -> Run on server -> Choose an existing server -> Select Tomcat -> Finish.

Testing Application using TestNG and Web Driver

TestNG is test framework for Java applications inspired by JUnit. It adds a lot of functionalities to including annotations, multi-threaded pool and data driven testing. WebDriver is a web automation framework that allows you to execute your tests against different browsers, not just Firefox, Chrome (unlike Selenium IDE). WebDriver also enables you to use a programming language in creating your test scripts.

Download the driver geckodriver for Firefox.

1. Go to <https://github.com/mozilla/geckodriver/releases/tag/v0.26.0>
2. Scroll down and download zip file for geckodriver.

```
~$ tar -xvf geckodriver-v0.26.0-linux32.tar.gz
~$ chmod +x geckodriver
```

3. Copy this file and put inside the project folder.

Let us see, how we write tests and perform them.

1. Create directory structure src/test/java.
2. Inside this folder, create a class AppTester. *Basically, what we are doing is Loading a web driver for Firefox browser in headless mode. Headless means UI is not actually opened. Then, using this driver, we navigate to the app URL and finally @Test annotations specify that a particular method is considered as test case.*
3. I also created `testng.xml` in project folder, which directs TestNG framework, which all classes contains the test cases annotations.
4. Run As -> Server.
5. Once the application is up, Run As -> Maven test.

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.

#Installing Jenkins

What we are doing is, downloading the key for jenkins and adding it to apt-keys. Next, we add deb package to the apt list. Update is used to update the URL to apt list by reading from .list file. When we do install, it takes the package from URL.

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

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

```
~$ sudo apt update~$ sudo apt install jenkins
```

```
~$ sudo systemctl start jenkins
```

Before building the pipeline, we will first setup the further elements of our pipeline, i.e. we will first install docker on our local machine as well as, setup nodes on Digital Ocean with docker installed on them, also we will install and create jobs on Rundeck.

Setting up servers on Digital Ocean

Digital Ocean is a company providing us with cloud servers in IaaS mode. So, it provides with bare installed OS and we need to install further setup. Similarly, it also provides with Database servers, etc. One can consider it like providing a complete enterprise model in cloud with data center facilities like load balancing, node migrations, etc.

1. Go to <https://cloud.digitalocean.com>
2. Sign Up with your email/gmail/github account.
3. Prove that **you are a not a robot**, by picture based authentication
4. It will ask for setup billing, provide your card details and it will redirect to payment of Rs.70, pay it and you will get into control panel of your account.
5. You will see many resources that could be added, droplets(servers), database, load balancers, clusters, disk, DNS, etc.
6. Select Create->Droplet

Specify following details:

OS: Ubuntu 18.04 (x64)

Choose Plan: Standard with 2GB RAM, 10\$ per month. It's enough.

Data center regions: Nearest to your location.

Additional options: Private networking, User data, Monitoring

SSH Key: This can be used for authentication and remote login to our nodes.

To generate, run this command on your machine: ~\$ ssh-keygen

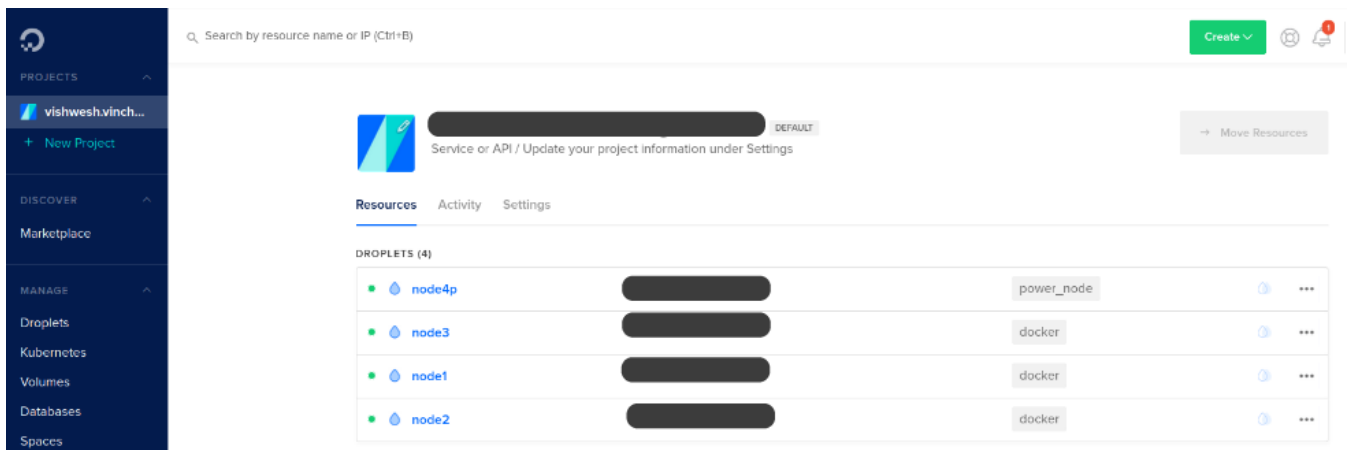
(Press enter to both, filename and passphrase. It will generate set of private and public key.)

```
~$ cd /home/{username}/.ssh
```

```
~$ cat id_rsa.pub
```

Copy the content and add this to Digital Ocean, in New SSH Key.7. Provide hostname and number of droplets to create.

8. Press create and it will redirect you to project, where you can see, a list of all resources you have. Now, you can see 3 droplets with IP address and hostnames.



Next, we setup our local machine, for ssh login, just by name, so that we don't need to write this IP address again and again.

```
~$ cd /home/{username}/.ssh
~$ touch config
~$ nano config
Write following content in it:
Host <node_name_on_digital_ocean>
  HostName <IP_on_digital_ocean>
  User root
.
.
so on.
```

Docker

Docker is like a revolution in the deployment model of the industry, where light weight containers are being run, rather than running full fledged VMs. Thus, the bare minimum configuration required for application to run is packaged as an image that could be hosted in a registry like DockerHub and runtime instance of it, called as container. ***This makes it small in size, fast bootup and run anywhere.*** Earlier, Docker emerged as containerization tool, but later it became of a suite of products including: docker-machine, docker-compose, docker swarm and many more.

We would be using a step next to basic docker, called as Docker Swarm. Before that, let's install docker on our local machine, as well as on our nodes that we created on Digital Ocean.

#Installing docker

```
~$ curl -fsSL https://get.docker.com -o get-docker.sh
~$ sh get-docker.sh
```

Docker Swarm

Docker Swarm is a distributed coordination of the nodes (managers and workers) using underlying RAFT database with protocol, that ensures that the docker service which we deploy is always up and running. It allows us to specify replicas, scale up and scale down the replicas. Thus, the important issues of cloud, *scalability and availability*, are taken care of well by Docker Swarm. It inherently provides layer-3 load balancing and hence, any node in the swarm can handle the incoming requests.

I configured 3 machines in swarm.

```
# Setting up swarm
# Note that we need to remote login in the nodes
node1-$ docker swarm init
(Initializes swarm and raft)

node1-$ docker swarm join-token manager (Get the command to join this swarm as manager)
node2-$ docker swarm join --token <token> IP:2377
node3-$ docker swarm join --token <token> IP:2377
```

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.

#Installing rundeck

Download Rundeck debian package from <http://rundeck.org/download/deb>
~\$ sudo dpkg -i rundeck_3.2.4.20200318-1_all.deb
~\$ sudo service rundeckd start

#Configuring authentication on rundeck

1. Run localhost:4440 in browser to open rundeck manager gui tool.
2. Default login is admin/admin.
3. Properties can be found at:
~\$ sudo cat /etc/rundeck/rundeck-config.properties
4. Before running a job, we need to setup authentication mechanism for the node.
Gear Icon on top right -> Key Storage -> Add or Upload key -> Select private key
-> Add the private key of node1 under keys/root -> Save.SSH into node1 and append the id_rsa.pub file to authorize_keys file inside /root/.ssh directory.

#Adding nodes

Create project -> Specify name, desc, label -> Press create.
Sources for adding nodes -> From file -> Specify:Format: resourcesxml
File path: /var/lib/rundeck/projects/Linux/etc/resources.xml
Select checkboxes for generate, Writeable, include server node.
Save.File gets generated -> Go in edit tab -> Insert Node tag with details:

```
<node name="node1" description="node1" tags="node1" hostname="<IP>"
osArch="amd64" osFamily="unix" username="root" ssh-key-storage-path="keys/root"/
>
```

Alternative:

1. The SSH key pair of rundeck server machine, should be copied to /var/lib/rundeck/.ssh/ directory.

```
~rundeck_machine$ cp .ssh/* /var/lib/rundeck/.ssh/
```

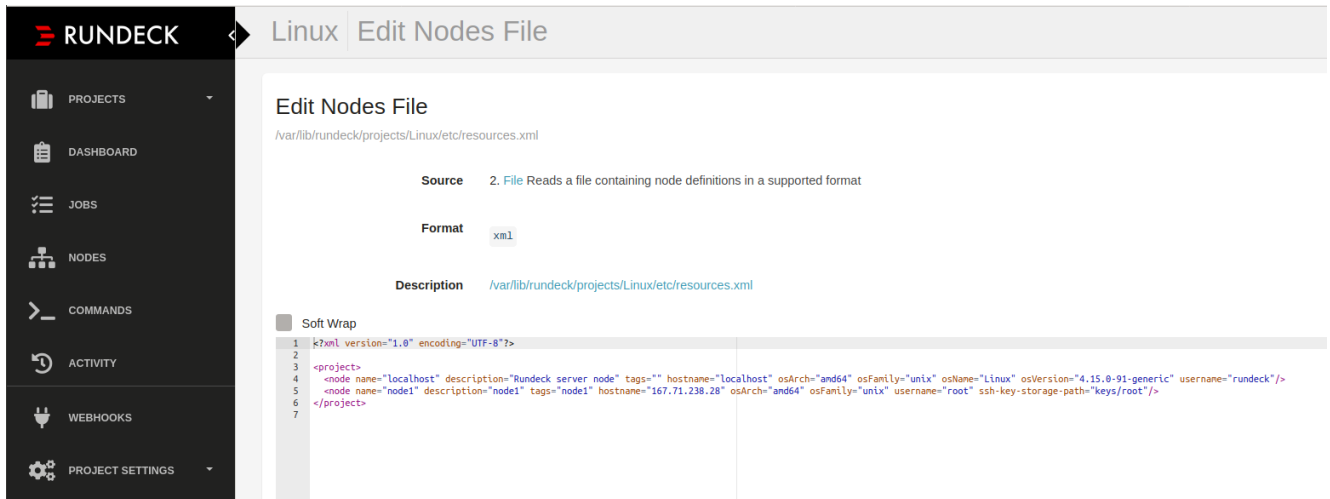
2. Next, add the public key id_rsa.pub of rundeck machine to .ssh/authorize_keys file in node machine.

3. However, 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.

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

If we specify **ssh-key-storage-path**, then the authentication mechanism of <node> tag will take priority over /var/lib/rundeck/.ssh keys. Using <node> tag authentication provides more flexibility using paraphrases.



Let us create a job in Rundeck that we want to run on our Swarm node. This will require a workflow of commands to be run.

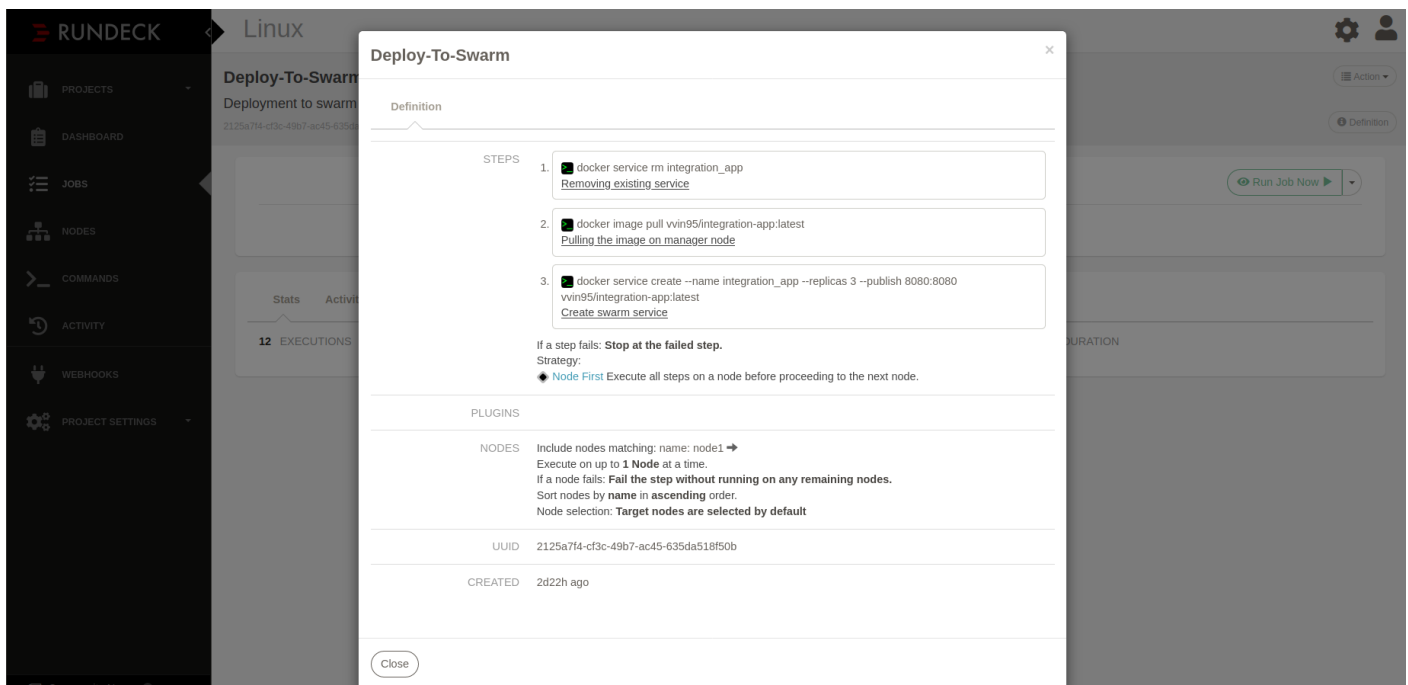
Project -> Jobs -> Job actions -> New Job -> Create steps -> As commands: Name the steps as you want and add the following commands.

Step1: `docker service rm integration_app`

Step2: `docker image pull vvin95/integration-app:latest`

Step3: `docker service create --name integration_app --replicas 3 --publish 8080:8080 vvin95/integration-app:latest`

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



Integrating SCM, Build Image and Deploy through Rundeck using Jenkins:

1. Go to localhost:8080 in browser, where Jenkins GUI is present.
2. It will ask for initialAdminPassword and also provide the path where this password is saved.
~\$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
3. Install General Plugins.
4. create a new user and Finish. Start using Jenkins.

Jenkins is based more on plugins and many people freak out about the sole purpose of making Jenkins. They say, we do automation for speeding the innovation process, while concentrating less on the repetitive tasks, but it turns out that, typically teams have to handle breakdown of Jenkins plugins as well many times. So, always prefer stable and non-vulnerable plugins.

#If not-vulnerable

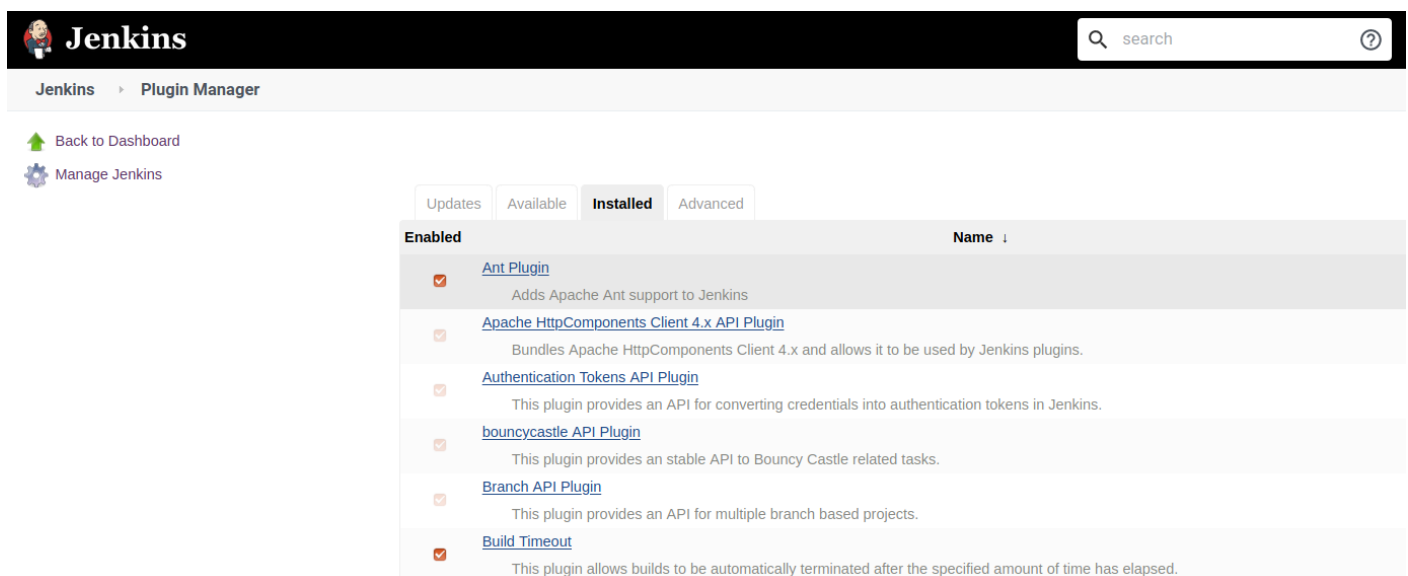
Manage Jenkins -> Manage Plugins -> Available -> Filter -> Install without restart

#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. Unleash Maven plugin
4. Docker plugin
5. Pipeline
6. Rundeck



Configure System in Jenkins

This involves configuring URLs (if we are using any external application server like rundeck) for the various components. Let's see step by step.

Manage Jenkins -> Configure System After installing above plugins, you will see following configuration items in this menu:

#Rundeck

-> Add Rundeck

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

#Maven Project Configuration

For this, leave default settings, we don't need to specify anything.

#Jenkins Location

Here, we can change the hostname and port number if we want.
Specify *System Admin Email* which will be used to send emails.

#Extended Email Notification

Provided the details of SMTP server.

SMTP Server : smtp.gmail.com
Default user E-mail suffix : @gmail.com
Use SMTP Authentication : Select Checkbox
Username : <Gmail account>
Password : <App password set in Gmail Privacy>
SMTP Port : 465
Default Recipients : cc: abc@gmail.com, bcc:xyz@gmail.com
Default Content : Modify this as per your wish

#Email Notification

Configure this same as above.** For app password, we need to enable 2-step authentication in gmail security settings. Otherwise, gmail does not allow sending emails by third party apps directly.

#Cloud

This will say, "The cloud configuration has moved to [a separate configuration page](#)."Go to different configuration page -> Add Cloud -> Docker.Provide Cloud details:

Name : Docker
Docker Host URI : tcp://127.0.0.1:4243
Expose DOCKER_HOST: Select

In Advanced tab, provide docker host IP. Test Connection. For this, we need to provide permissions to read and execute:

```
~$ sudo chmod 666 /var/run/docker.sock
```

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
Ant and Gradle : default
Maven : /usr/share/maven
Docker : default

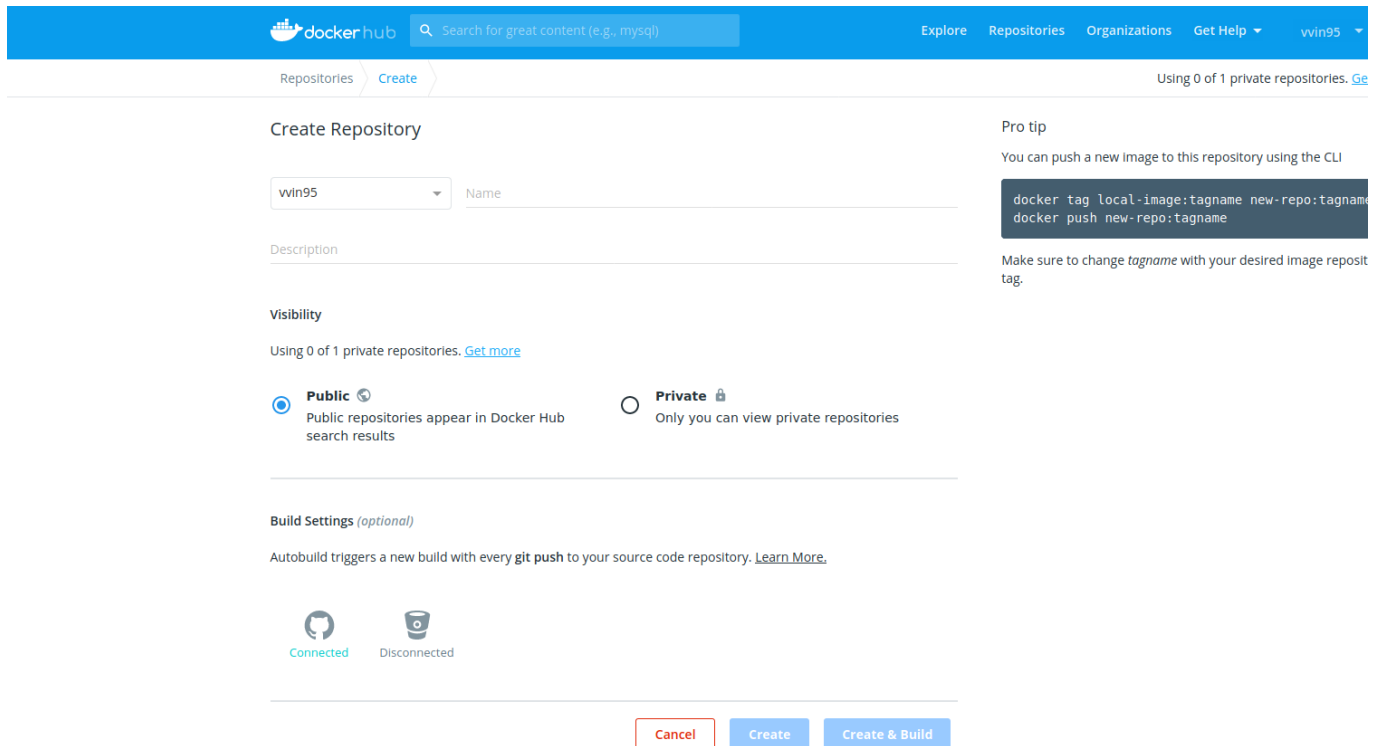
Creating DockerHub repository

DockerHub is the default registry for docker, where we can find both official images from companies and customized images created by different users. Let us create our own repository.

Go to <https://hub.docker.com/> .

Sign Up and go into account.

Create Repository -> Provide Name and Description -> Create.



Repositories [Create](#) Using 0 of 1 private repositories. [Get more](#)

Create Repository

Name

Description

Visibility

Using 0 of 1 private repositories. [Get more](#)

☒ **Public** Public repositories appear in Docker Hub search results

☐ **Private** Only you can view private repositories

Build Settings *(optional)*

Autobuild triggers a new build with every **git push** to your source code repository. [Learn More](#).

Connected Disconnected

[Cancel](#) [Create](#) [Create & Build](#)

Pro tip

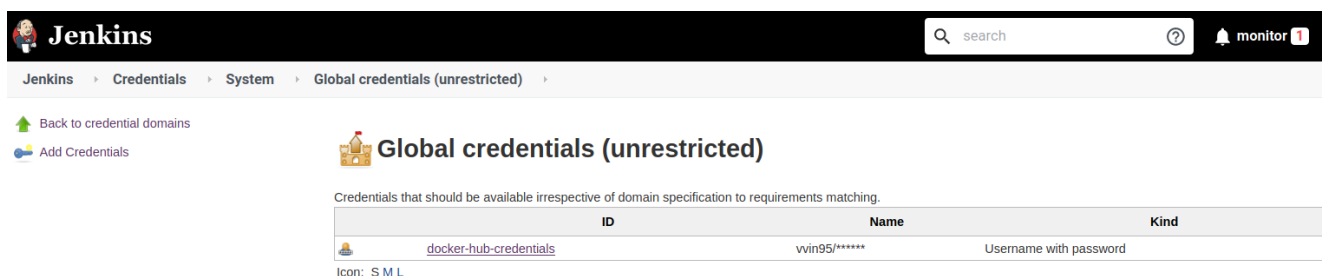
You can push a new image to this repository using the CLI

```
docker tag local-image:tagname new-repo:tagname
docker push new-repo:tagname
```

Make sure to change *tagname* with your desired image repository tag.

Adding DockerHub credentials in Jenkins

In left menu, click credentials -> System -> Global Credentials -> Provide username and password of DockerHub -> Provide an ID for this credentials as *docker-hub-credentials*. We will use this later, so make a note of it.



Jenkins [search](#) [monitor](#)

Jenkins > Credentials > System > Global credentials (unrestricted)

[Back to credential domains](#) [Add Credentials](#)

Global credentials (unrestricted)

Credentials that should be available irrespective of domain specification to requirements matching.

ID	Name	Kind
docker-hub-credentials	vvin95/*****	Username with password

Icon: [S](#) [M](#) [L](#)

Creating Jenkins Pipeline

The aim of the pipeline is to trigger whenever we push changes to our 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 swarm of nodes.

Config the github webhook: This is an API which sends POST message to Jenkins that GitHub repository has gone some changes. GitHub requires a public URL for Jenkins server, but we are running on localhost, so we use **ngrok** which will generate a tunnel for us, so that trigger could be forwarded to us. If you are using Jenkins on some public server, then do provide that URL.

#Generating public URL

1. Go to <https://ngrok.com/>
2. Create a account and login.
3. Go to Auth option in left pane in account. Copy the command from there, along with authentication token.
4. On terminal, run:
~\$ sudo snap install ngrok
~\$./ngrok authtoken <auth_token>
~\$ ngrok http 8080

This will generate the public URL for us in terminal.
<http://6bbb4a4e.ngrok.io>

#GitHub Webhook

GitHub repository -> Settings -> Webhooks -> Add Webhook.

Payload URL: <http://6bbb4a4e.ngrok.io/github-webhook/>

Select just the push event. Save. Now, whenever we will push changes to this repository, it will trigger Jenkins.

Creating Jenkins pipeline

Jenkins -> New Item -> Pipeline -> Enter item name -> Ok

General Section: Leave it default

Build triggers: This specifies when will our pipeline trigger. It has various options.

Build periodically: *Every specified period, run pipeline.*

Poll SCM : *Poll every specified period, if any changes, then run pipeline*

GitHub hook trigger for GITScm polling: We use this one.

Advanced Project Options: Leave it default:

Pipeline: Select pipeline as script.

```
pipeline {
  environment {
    registry = "<DockerHubUsername>/<RepositoryName>"
    registryCredential = 'docker-hub-credentials'
    dockerImage = ''
    dockerImageLatest = ''
  }
  agent any
  stages {
    stage('Cloning Git') {
      steps {
        git '<GitHubRepository URL>_'
      }
    }
    stage('Build Executable Jar'){
      steps {
        sh 'mvn clean test package'
      }
    }
    stage('Building image') {
      steps{
```

```

        script {
            dockerImage = docker.build registry + ":$BUILD_NUMBER"
            dockerImageLatest = docker.build registry + ":latest"
        }
    }
}
stage('Deploy Image') {
    steps{
        script {
            docker.withRegistry( '', registryCredential ) {
                dockerImage.push()
                dockerImageLatest.push()
            }
        }
    }
}
stage('Remove Unused docker image') {
    steps{
        sh "docker rmi $registry:$BUILD_NUMBER"
    }
}
stage('Execute Rundeck job') {
    steps {
        script {
            step([$class: "RundeckNotifier",
                includeRundeckLogs: true,
                jobId: "2125a7f4-cf3c-49b7-ac45-635da518f50b",
                rundeckInstance: "Rundeck",
                shouldFailTheBuild: true,
                shouldWaitForRundeckJob: true,
                tailLog: true])
        }
    }
}
}
post {
    always{
        emailx body: "Dear Sir/Mam, <br/><br/> ${currentBuild.currentResult}:
Job ${env.JOB_NAME} build ${env.BUILD_NUMBER} <br/> More info at: $
${env.BUILD_URL} <br/><br/> THIS IS A SYSTEM GENERATED EMAIL. PLEASE DO NOT
REPLY.",
                recipientProviders: [[${class: 'DevelopersRecipientProvider'},
[${class: 'RequesterRecipientProvider'}]],
                subject: "Jenkins Build ${currentBuild.currentResult}: Job $
${env.JOB_NAME}"
    }
}
}
}

```

Added Dockerfile to the project in Eclipse, which is used to create an image.

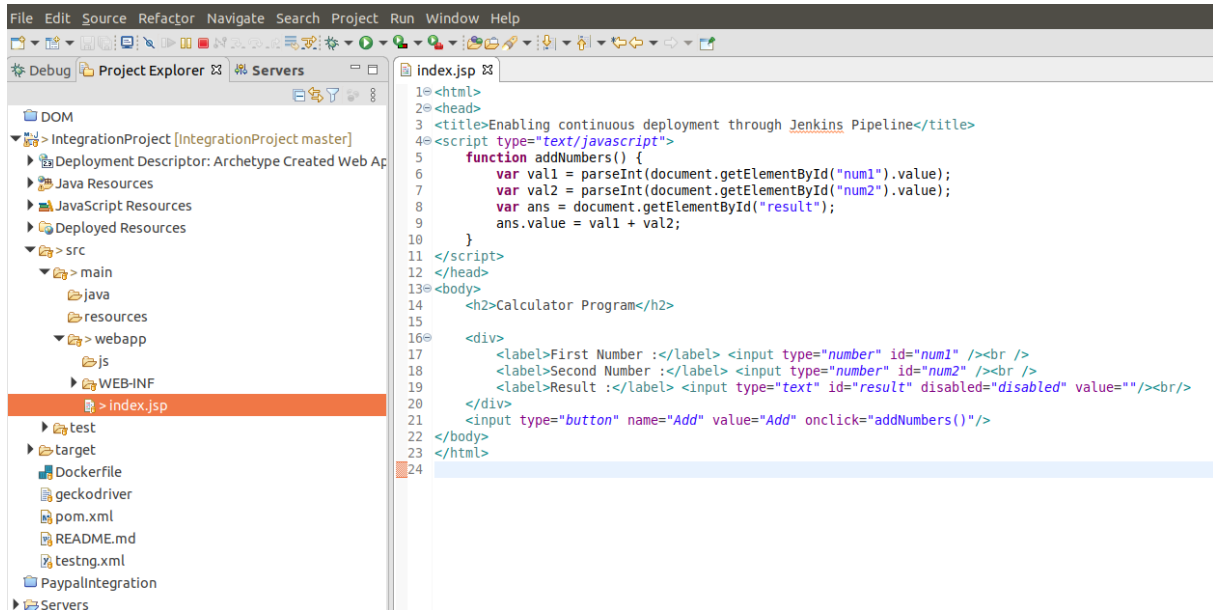
```

FROM tomcat:9.0
WORKDIR /usr/local/tomcat
ADD target/IntegrationProject.war webapps/

```

Snapshots of the work:

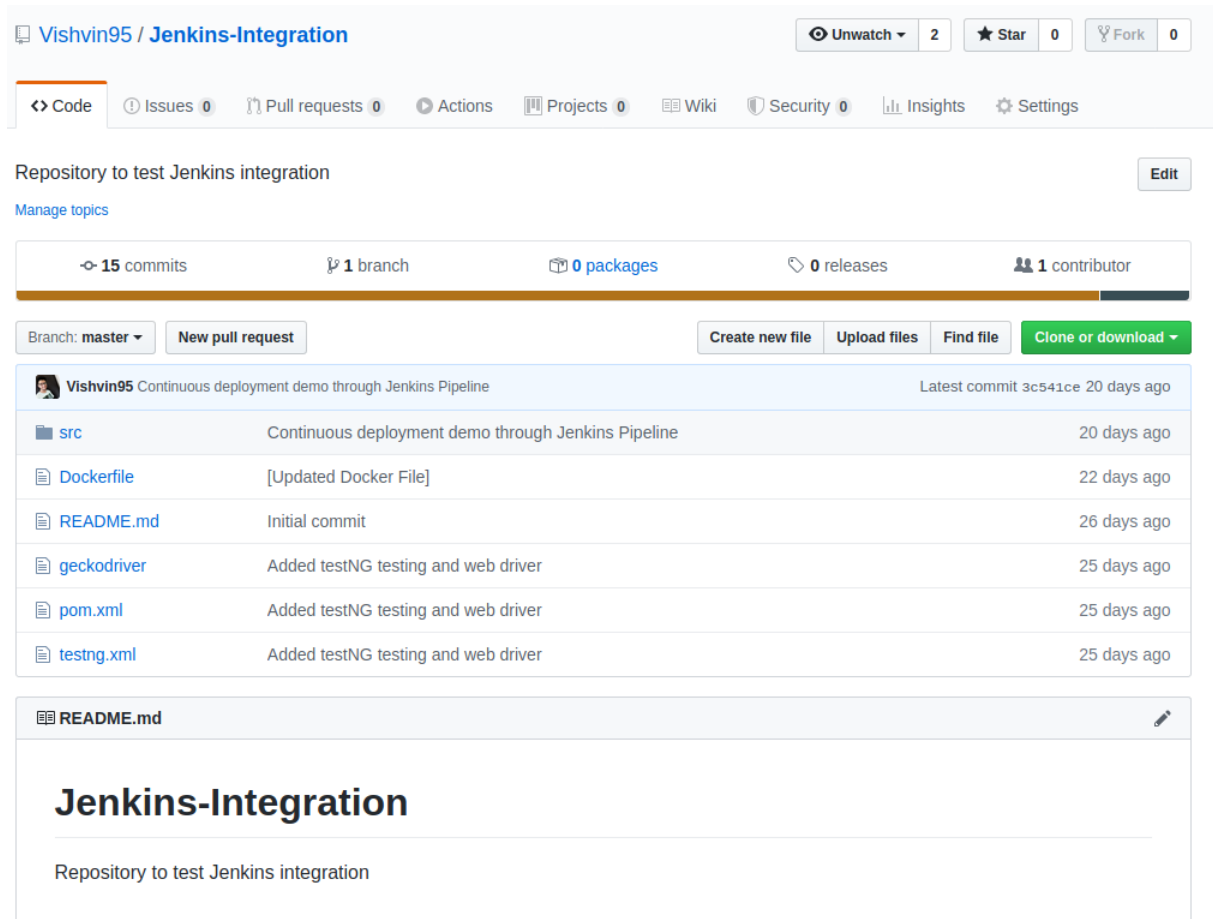
1. Added add operator in code



```
File Edit Source Refactor Navigate Search Project Run Window Help
Debug Project Explorer Servers index.jsp
DOM
IntegrationProject [IntegrationProject master]
  Deployment Descriptor: Archetype Created Web App
  Java Resources
  JavaScript Resources
  Deployed Resources
  src
    main
      java
      resources
      webapp
        js
          WEB-INF
            index.jsp
        test
        target
        Dockerfile
        geckodriver
        pom.xml
        README.md
        testng.xml
        PayPalIntegration
  Servers

1<@<html>
2<@<head>
3  <title>Enabling continuous deployment through Jenkins Pipeline</title>
4<@<script type="text/javascript">
5    function addNumbers() {
6      var val1 = parseInt(document.getElementById("num1").value);
7      var val2 = parseInt(document.getElementById("num2").value);
8      var ans = document.getElementById("result");
9      ans.value = val1 + val2;
10   }
11 </script>
12 </head>
13<@<body>
14   <h2>Calculator Program</h2>
15
16<@   <div>
17     <label>First Number :</label> <input type="number" id="num1" /><br />
18     <label>Second Number :</label> <input type="number" id="num2" /><br />
19     <label>Result :</label> <input type="text" id="result" disabled="disabled" value=""/><br/>
20   </div>
21   <input type="button" name="Add" value="Add" onclick="addNumbers()" />
22 </body>
23 </html>
24
```

2. GitHub Repository



Vishvin95 / Jenkins-Integration

Unwatch 2 Star 0 Fork 0

Code Issues 0 Pull requests 0 Actions Projects 0 Wiki Security 0 Insights Settings

Repository to test Jenkins integration [Edit](#)

Manage topics

15 commits 1 branch 0 packages 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

File	Commit Message	Time
src	Continuous deployment demo through Jenkins Pipeline	20 days ago
Dockerfile	[Updated Docker File]	22 days ago
README.md	Initial commit	26 days ago
geckodriver	Added testNG testing and web driver	25 days ago
pom.xml	Added testNG testing and web driver	25 days ago
testng.xml	Added testNG testing and web driver	25 days ago

README.md

Jenkins-Integration

Repository to test Jenkins integration

3. Generating Tunneling URL using Ngrok

```
ngrok by @inconshreveable

Session Status      online
Account             vishwesh.vinchurkar@gmail.com (Plan: Free)
Version             2.3.35
Region              United States (us)
Web Interface        http://127.0.0.1:4040
Forwarding           http://5a2423c5.ngrok.io -> http://localhost:8080
Forwarding           https://5a2423c5.ngrok.io -> http://localhost:8080

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

4. Configuring Webhook in GitHub Repository

Options

Manage access

Branches

Webhooks

Notifications

Integrations

Deploy keys

Secrets

Actions

Moderation

Interaction limits

Webhooks / Manage webhook

We'll send a POST request to the URL below with details of any subscribed events. You can also specify which data format you'd like to receive (JSON, x-www-form-urlencoded, etc). More information can be found in [our developer documentation](#).

Payload URL *

http://ae917ebe.ngrok.io/github-webhook/

Content type

application/x-www-form-urlencoded

Secret

Which events would you like to trigger this webhook?

☒ Just the push event.

☐ Send me **everything**.

☐ Let me select individual events.

☒ Active

We will deliver event details when this hook is triggered.

Update webhook

Delete webhook

5. Incremental commit to GitHub repository

Vishvin95 / Jenkins-Integration

Unwatch 2

Star 0

Fork 0

<> Code

Issues 0

Pull requests 0

Actions

Projects 0

Wiki

Security 0

Insights

Settings

Branch: master

Commits on May 1, 2020

[Addition]: Added division operator for calculator

Vishvin95 committed 14 seconds ago

f5879fd

<>

[Added]: Added multiply operation to calculator

Vishvin95 committed 4 minutes ago

3c1f535

<>

[Added]: Added subtraction operator

Vishvin95 committed 9 minutes ago

f516bf0

<>

[Added]: Add two numbers





Vishvin95 committed 14 minutes ago

fcc6ceb

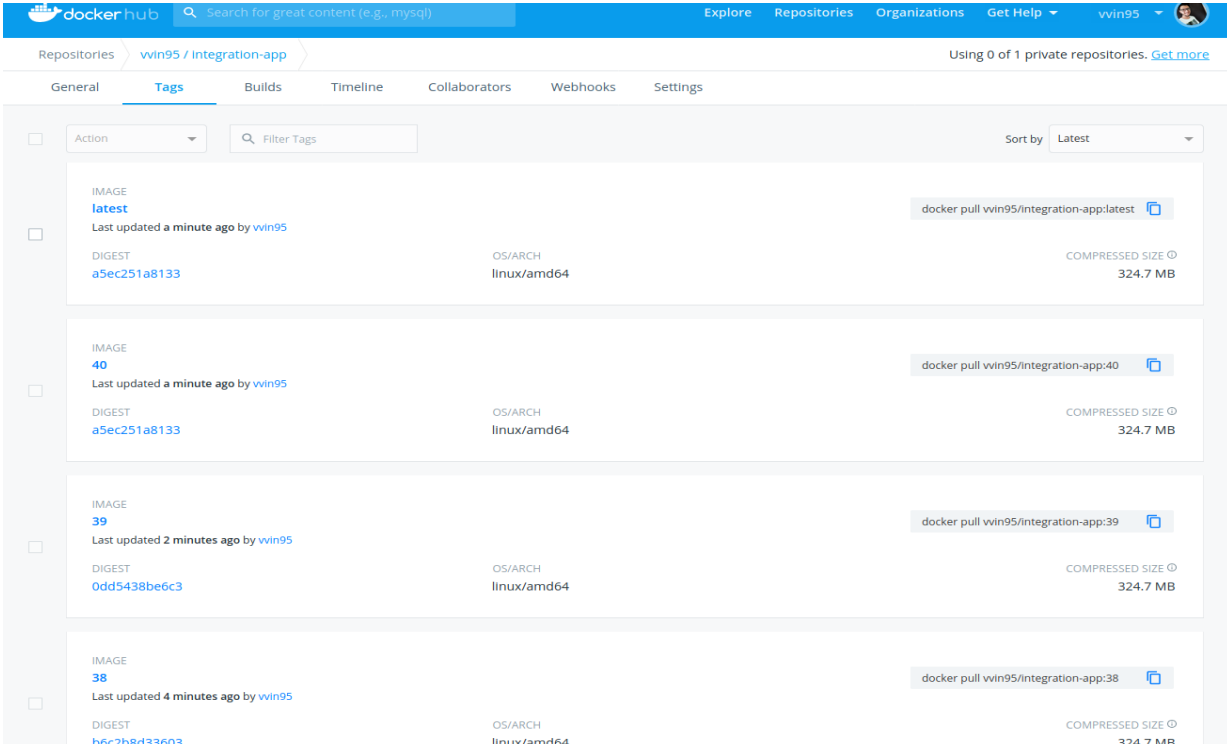
<>

6. Jenkins pipeline triggered from GitHub webhook

Recent Deliveries

✓	 26dd1020-8b7c-11ea-9178-2ccc8749621c	2020-05-01 12:49:53	...
✓	 ac3fd582-8b7b-11ea-9d29-ad95b754ec00	2020-05-01 12:46:27	...
✓	 f8a46312-8b7a-11ea-8704-93af8b4f00a8	2020-05-01 12:41:26	...
✓	 383e1ece-8b7a-11ea-8741-ff6a60f739df	2020-05-01 12:36:03	...

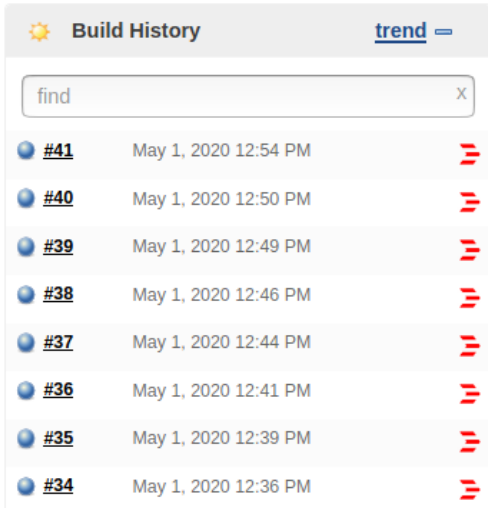
7. Image pushed to DockerHub



The screenshot shows the DockerHub interface for the repository 'vwin95 / integration-app'. The 'Tags' tab is selected, displaying a list of image tags. Each tag entry includes a checkbox, the tag name, the last update time, the user 'vwin95', the digest, the OS/ARCH (linux/amd64), and the compressed size (324.7 MB). A pull command is provided for each tag.

Image	Tag	Last updated	User	Digest	OS/ARCH	Compressed Size	Command
IMAGE	latest	Last updated a minute ago	vwin95	a5ec251a8133	linux/amd64	324.7 MB	docker pull vwin95/integration-app:latest
IMAGE	40	Last updated a minute ago	vwin95	a5ec251a8133	linux/amd64	324.7 MB	docker pull vwin95/integration-app:40
IMAGE	39	Last updated 2 minutes ago	vwin95	Odd5438be6c3	linux/amd64	324.7 MB	docker pull vwin95/integration-app:39
IMAGE	38	Last updated 4 minutes ago	vwin95	b6c2b8d33603	linux/amd64	324.7 MB	docker pull vwin95/integration-app:38

8. Jenkins Build History



The screenshot shows the Jenkins 'Build History' page. It features a search bar with the text 'find' and a 'trend' link. Below the search bar is a list of build entries, each with a blue circle icon, a build number, a timestamp, and a red 'X' icon.

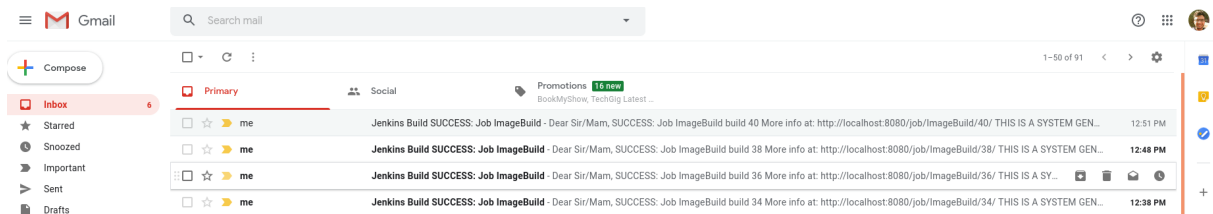
Build Number	Timestamp
#41	May 1, 2020 12:54 PM
#40	May 1, 2020 12:50 PM
#39	May 1, 2020 12:49 PM
#38	May 1, 2020 12:46 PM
#37	May 1, 2020 12:44 PM
#36	May 1, 2020 12:41 PM
#35	May 1, 2020 12:39 PM
#34	May 1, 2020 12:36 PM

9. Jenkins Pipeline View

Stage View

		Cloning Git	Build Executable Jar	Building image	Deploy Image	Remove Unused docker image	Execute Rundeck job	Declarative: Post Actions
Average stage times: (Average full run time: ~1min 49s)		1s	15s	4s	38s	516ms	22s	2s
#41	May 01 12:54 No Changes	3s	13s	7s	1min 8s	343ms	29s	195ms
#40	May 01 12:50 1 commit	2s	11s	5s	39s	353ms	28s	5s
#39	May 01 12:49 No Changes	1s	12s	5s	49s	392ms	29s	104ms
#38	May 01 12:46 1 commit	1s	12s	4s	57s	356ms	30s	4s
#37	May 01 12:44 No Changes	1s	13s	5s	50s	377ms	28s	92ms
#36	May 01 12:41 1 commit	1s	14s	4s	51s	367ms	28s	4s
#35	May 01 12:39 No Changes	1s	12s	5s	56s	2s	32s	112ms
#34	May 01 12:36 1 commit	1s	12s	9s	46s	367ms	43s	4s

10. Sending of Notifications



11. Rundeck Job View

Activity for Jobs

1 - 10 of 20 Executions any time [Save Filter...](#) [Bulk Delete](#)

✓	05/01/2020 12:56 PM Yesterday at 12:56 PM	1 ok	27 seconds	by admin	Deploy-To-Swarm	#131
✓	05/01/2020 12:51 PM Yesterday at 12:51 PM	1 ok	27 seconds	by admin	Deploy-To-Swarm	#130
✓	05/01/2020 12:50 PM Yesterday at 12:50 PM	1 ok	27 seconds	by admin	Deploy-To-Swarm	#129
✓	05/01/2020 12:48 PM Yesterday at 12:48 PM	1 ok	28 seconds	by admin	Deploy-To-Swarm	#128

✓ **Deploy-To-Swarm**

Deployment to swarm the docker image

Succeeded 00:00:26 at Fri 12:56 pm
you [Run Again](#)

Log Output

100% 1/1 COMPLETE 0 FAILED 0 INCOMPLETE 0 NOT STARTED

Node

node1 All Steps OK

Removing existing service OK 12:55:44 pm 0.00:01

Pulling the image on manager node OK 12:55:46 pm 0.00:06

Create swarm service OK 12:55:53 pm 0.00:17

12. Incremental view of application running of digital ocean

←

→

↺

🏠

🔒

167.71.238.28:8080/IntegrationProject/

Calculator Program

First Number :

Second Number :

Answer :

Add

←

→

↺

🏠

🔒

167.71.238.28:8080/IntegrationProject/

Calculator Program

First Number :

Second Number :

Answer :

Add

Subtract

←

→

↺

🏠

🔒

167.71.238.28:8080/IntegrationProject/

Calculator Program

First Number :

Second Number :

Answer :

Add

Subtract

Multiply

←

→

↺

🏠

🔒

167.71.238.28:8080/IntegrationProject/

Calculator Program

First Number :

Second Number :

Answer :

Add

Subtract

Multiply

Divide