****

**Feedback-Rest-API setup and deployment Document**

[www.yash.com](http://www.yash.com)

 This document is exclusive property of Yash Technologies Pvt. Ltd.

No part or whole of this document may be reproduced in any form without written permission.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Version** | **Date** | **Author** | **Reviewer** | **Approver** | **Change Summary** |
| 1.0 | 25-Dec-2018 | Vishal Kumkar | Mitul Deshmukh | Sumeet Muchhal |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**Revision History**

Pre-requisite to deploy app with CI/CD Pipeline

1. Ubuntu OS 16.04
2. Java Development Toolkit 1.8
3. Jenkins
4. Docker
5. Minikube + Kubernetes 10
6. postgres driver 42.2.5

Content

1. Installation and Setup of jdk 1.8 on Ubuntu
2. Installation and Setup of Jenkins
3. Installation and Setup of Docker
4. Installation and Setup of Minikube + Kubernetes
5. Setup feedback-rest-api application with docker and Jenkins file and committing to Git.
6. Add Credentials for Github and Docker in Jenkins
7. Set up Jenkins CI/CD Pipeline

Installation and Setup of Jenkins

**Introduction**

Jenkins is an open source automation server intended to automate repetitive technical tasks involved in the continuous integration and delivery of software. Jenkins is Java-based and can be installed from Ubuntu packages or by downloading and running its Web application ARchive (WAR) file — a collection of files that make up a complete web application which is intended to be run on a server.

In this tutorial we will install Jenkins by adding its Debian package repository, then using that repository to install the package using apt-get.

### Prerequisites

To follow this tutorial, you will need:

**One Ubuntu 16.04 server** configured with a non-root sudo user.

**Step 1 – Installing Jenkins**

* The version of Jenkins included with the default Ubuntu packages is often behind the latest available version from the project itself. In order to take advantage of the latest fixes and features, we'll use the project-maintained packages to install Jenkins.
* First, we'll add the repository key to the system.

wget -q -O - https://pkg.jenkins.io/debian/jenkins-ci.org.key | sudo apt-key add –

* When the key is added, the system will return OK. Next, we'll append the Debian package repository address to the server's sources.list:

echo deb https://pkg.jenkins.io/debian-stable binary/ | sudo tee /etc/apt/sources.list.d/jenkins.list

* When both of these are in place, we'll run update so that apt-get will use the new repository:

sudo apt-get update

* Finally, we'll install Jenkins and its dependencies, including Java:

sudo apt-get install jenkins

Now that Jenkins and its dependencies are in place, we'll start the Jenkins server.

**Step 2 - Starting Jenkins**

* Using systemctl  will start Jenkins:

sudo systemctl start Jenkins

* Since systemctl doesn't display output, we'll use its status command to verify that it started successfully:

sudo systemctl status jenkins

* If everything went well, the beginning of the output should show that the service is active and configured to start at boot:

Output

jenkins.service - LSB: Start Jenkins at boot time

Loaded: loaded (/etc/init.d/jenkins; bad; vendor preset: enabled)

Active:active (exited) since Thu 2017-04-20 16:51:13 UTC; 2min 7s ago

Docs: man:systemd-sysv-generator(8)

Now that Jenkins is running, we'll adjust our firewall rules so that we can reach Jenkins from a web browser to complete the initial set up.

**Step 3 - Opening the Firewall**

* By default, Jenkins runs on port 8080, so we'll open that port using ufw:

sudo ufw allow 8080

* We can see the new rules by checking UFW's status.

sudo ufw status

We should see that traffic is allowed to port 8080 from anywhere:

Output

Status: active

To Action From

-- ------ ----

OpenSSH ALLOW Anywhere

8080 ALLOW Anywhere

OpenSSH (v6) ALLOW Anywhere (v6)

8080 (v6) ALLOW Anywhere (v6)

**Note:** If the firewall is inactive, the following commands will make sure that OpenSSH is allowed and then enable it.

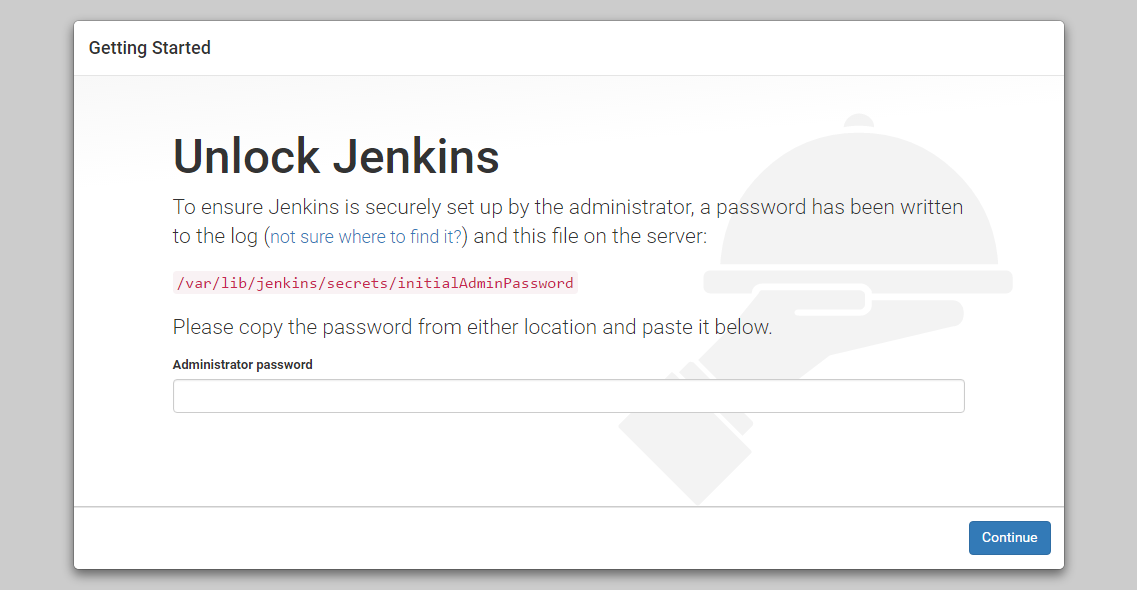
sudo ufw allow OpenSSH

sudo ufw enable

Now that Jenkins is installed and the firewall allows us to access it, we can complete the initial setup.

**Step 4 – Setting up Jenkins**

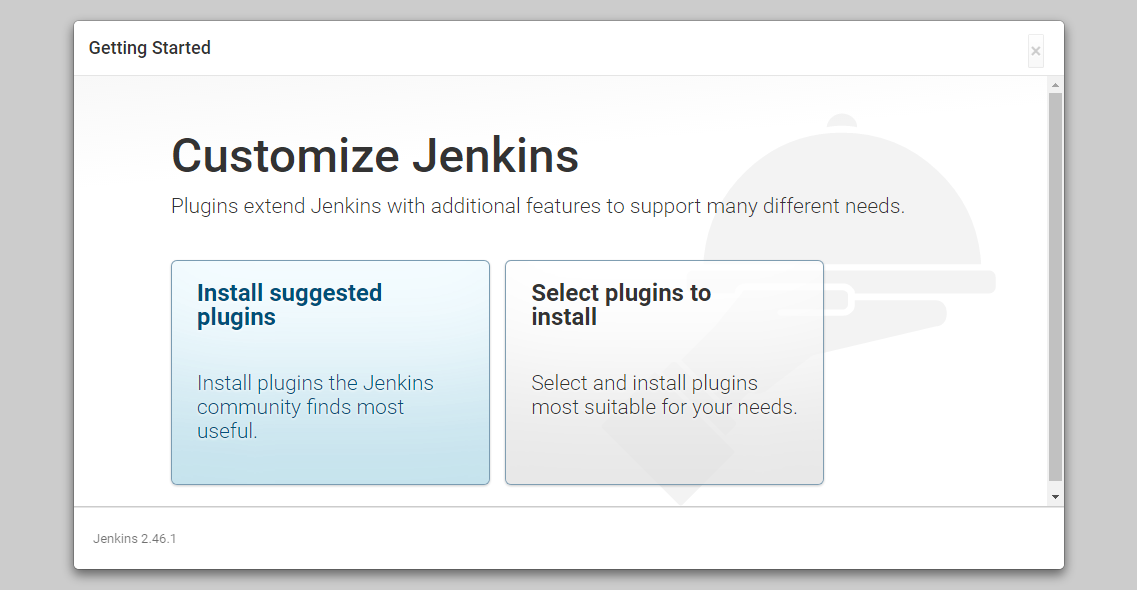
* To set up our installation, we'll visit Jenkins on its default port, 8080, using the server domain name or IP address:  <http://ip_address_or_domain_name:8080>
* We should see "Unlock Jenkins" screen, which displays the location of the initial password



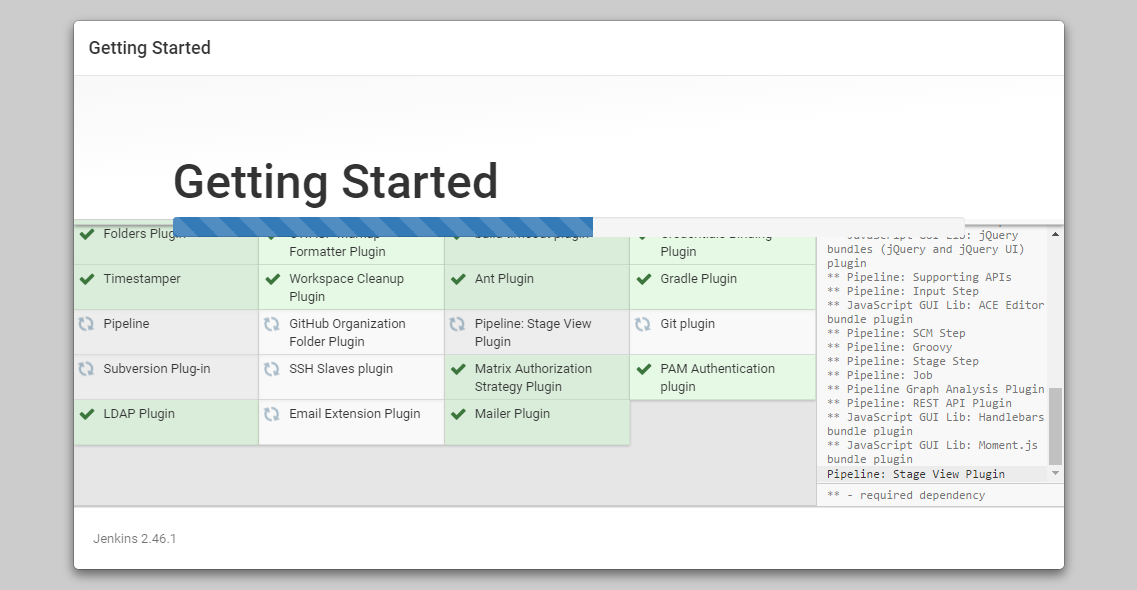
* In the terminal window, we'll use the cat command to display the password:

sudo cat /var/lib/jenkins/secrets/initialAdminPassword

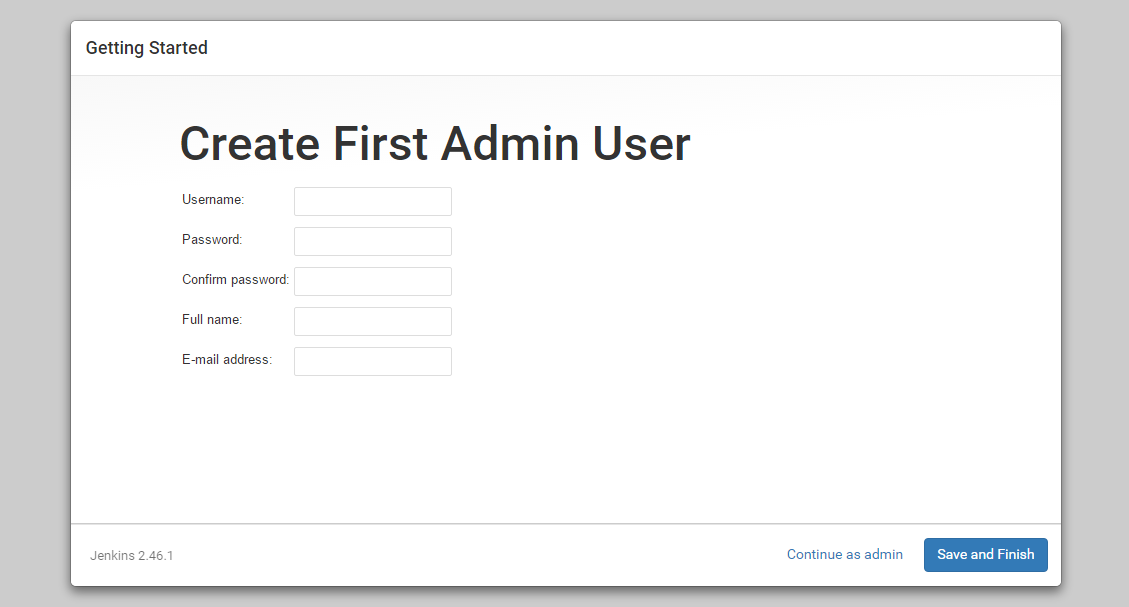
* We'll copy the 32-character alphanumeric password from the terminal and paste it into the "Administrator password" field, then click "Continue". The next screen presents the option of installing suggested plugins or selecting specific plugins.



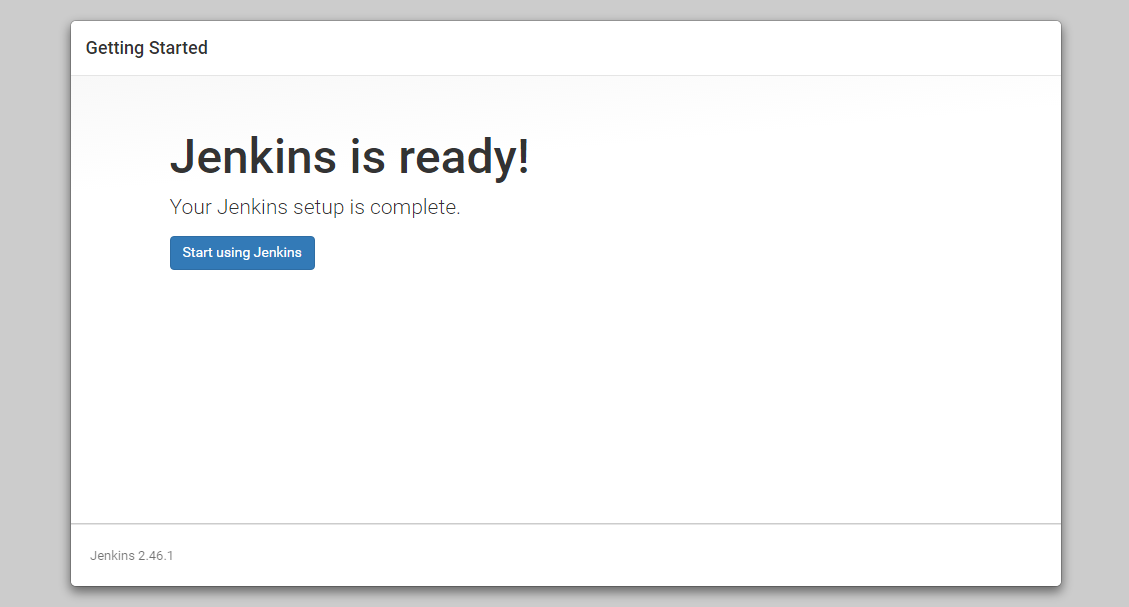
* We'll click the "Install suggested plugins" option, which will immediately begin the installation process:



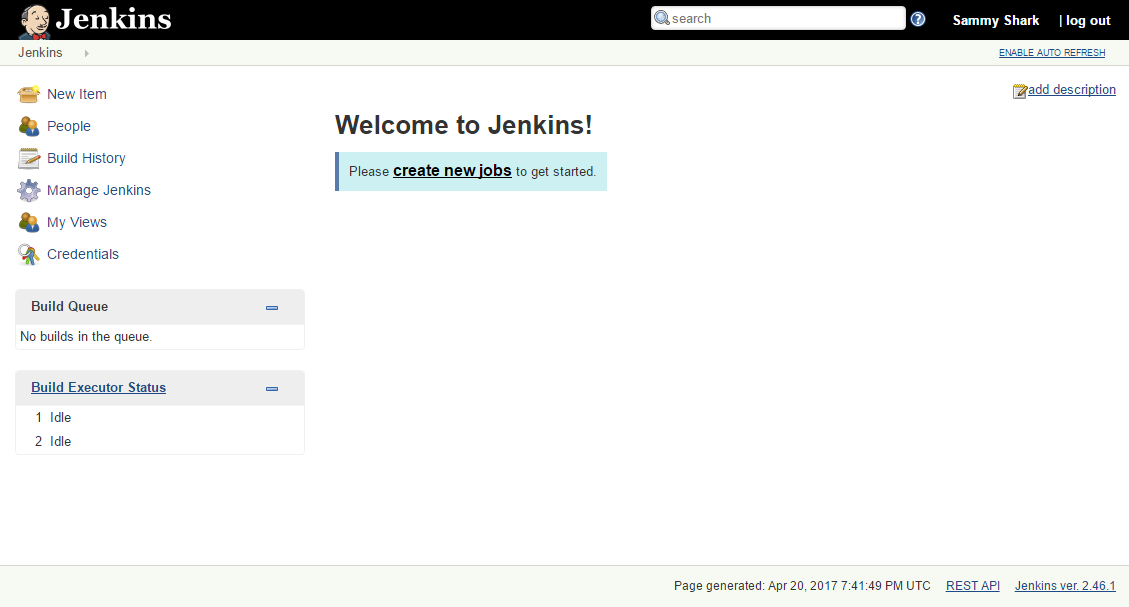
* When the installation is complete, we'll be prompted to set up the first administrative user. It's possible to skip this step and continue as admin using the initial password we used above, but we'll take a moment to create the user.



* Once the first admin user is in place, you should see a "Jenkins is ready!" confirmation screen.



* Click "Start using Jenkins" to visit the main Jenkins dashboard:



At this point, Jenkins has been successfully installed.

Installation and Setup of Docker

### Introduction

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.

**Prerequisites**

* One Ubuntu 16.04 server set up with a non-root user with sudo privileges and a basic firewall.
* An account on Docker Hub if you wish to create your own images and push them to Docker Hub, as shown in Steps 7 and 8

**Steps:**

1. First, in order to ensure the downloads are valid, add the GPG key for the official Docker repository to your system:

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

1. Add the Docker repository to APT sources:

sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu $(lsb\_release -cs) stable"

1. Next, update the package database with the Docker packages from the newly added repo:

sudo apt-get update

1. Make sure you are about to install from the Docker repo instead of the default Ubuntu 16.04 repo:

apt-cache policy docker-ce

You should see output similar to the follow:

Output of apt-cache policy docker-ce

docker-ce:

Installed: (none)

Candidate: 18.06.1~ce~3-0~ubuntu

Version table:

18.06.1~ce~3-0~ubuntu 500

500 https://download.docker.com/linux/ubuntu xenial/stable amd64 Packages

Notice that docker-ce is not installed, but the candidate for installation is from the Docker repository for Ubuntu 16.04 (xenial).

1. Finally, install Docker:

sudo apt-get install -y docker-ce

1. Docker should now be installed, the daemon started, and the process enabled to start on boot. Check that it's running:

sudo systemctl status docker

The output should be similar to the following, showing that the service is active and running:

Output

● docker.service - Docker Application Container Engine

Loaded: loaded (/lib/systemd/system/docker.service; enabled; vendor preset: enabled)

Active: active (running) since Thu 2018-10-18 20:28:23 UTC; 35s ago

Docs: https://docs.docker.com

Main PID: 13412 (dockerd)

CGroup: /system.slice/docker.service

├─13412 /usr/bin/dockerd -H fd://

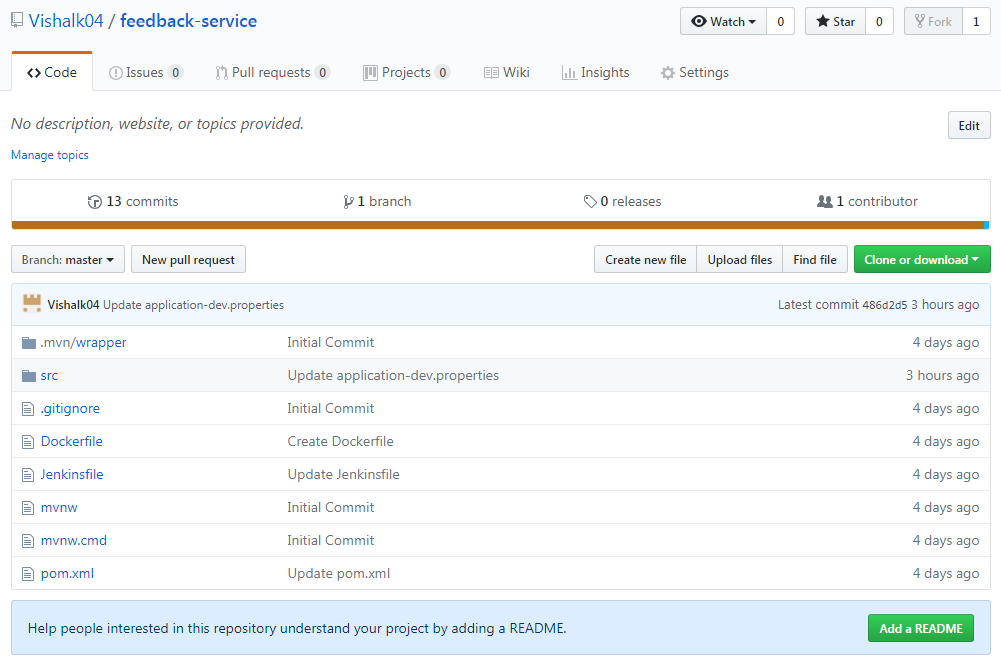
└─13421 docker-containerd --config /var/run/docker/containerd/containerd.toml

Installation and Setup of Minikube + kubernetes

Setup feedback-rest-api application with Docker and Jenkins file and Commit to Git

* The Feedback Rest API is RESTful Service implemented with spring boot framework.
* In order to deploy it on Docker and Kubernetes with Jenkins, we required to add the Dockerfile and Jenkinsfile on the root path of project. And then we will pushed the application on Git Repository.

The folder Structure of the Git repository should look like below.



* We will use this Git repository in Jenkins CI/CD pipeline in next step to build the deployment.
* The Jenkins pipeline will set up the in the way that, when the commit will be performed on this repository, it will trigger the Jenkins Pipeline.
* The pipeline will fetch the Jenkinsfile from the root path of project and it will run the script defined in this file. This file is written in the groovy script. The Jenkinsfile is attached below.



**We have defined the below stages in Jenkinsfile:**

1. Checkout project from Git Repository.
2. Compile Project and Build the Docker Image.
3. Push Docker Image to Docker Repository.
4. Deploy the Docker Image with Kubernetes.
5. **Checkout project from Git Repository**

The below script will checkout the code from git repository.

Jenkins will get the details about this GIT Repository from SCM.

The procedure to create SCM is described in ***Set up CI/CD Pipeline*** Section of this document.

|  |
| --- |
| stage('Clone repository') { |
|  | /\* Let's make sure we have the repository cloned to our workspace \*/ |
|  |  |
|  | checkout scm |
|  | } |

1. **Compile Project and Build the Docker Image**

* This stage will compile the code with sh 'mvn clean install' and generate the jar***.***
* The below command in this stage builds the docker image.

app = docker.build("kartikjalgaonkar/feedback-service")

|  |
| --- |
| stage('Build image') { |
|  | sh 'mvn clean install' |
|  |  |
|  |  |
|  | /\* This builds the actual image; synonymous to |
|  | \* docker build on the command line \*/ |
|  |  |
|  | app = docker.build("kartikjalgaonkar/feedback-service") |
|  | } |

* To build the Docker Image Jenkins will search for Dockerfile in root path of application. This Dockerfile contain the script to build the docker image as below.

|  |
| --- |
| FROM java:alpine |
|  | ADD target/feedback-service.jar feedback-service.jar |
|  | EXPOSE 8093 |
|  | ENTRYPOINT ["java","-jar","feedback-service.jar"] |

* This script will use the java: alpine template to generate the docker image.
* Below command will build the Docker Image of jar file from source folder target/feedback-service.jar.

ADD target/feedback-service.jar feedback-service.jar

* This image will expose on the post 8093. ENTRYPOINT defines the metadata like path, type about the docker image.

|  |
| --- |
|  |

1. **Push Docker Image to Docker Repository.**

Once the docker image is build it will pushed the docker repository.

Jenkins will login to the docker repository using the docker credentials with id docker\_credentials and will push the docker image there. Procedure to create the credential in Jenkins is described later in this document.

Below is the script for this stage.

|  |
| --- |
| stage('Push image') { |
|  | /\* Finally, we'll push the image with two tags: |
|  | \* First, the incremental build number from Jenkins |
|  | \* Second, the 'latest' tag. |
|  | \* Pushing multiple tags is cheap, as all the layers are reused. \*/ |
|  | docker.withRegistry('https://registry.hub.docker.com', 'docker\_credentials') { |
|  | app.push("${env.BUILD\_NUMBER}") |
|  | app.push("latest") |
|  | } |
|  | } |

The Docker repository look like below once the docker image is pushed successfully.

//screenshot

1. **Deploy the Docker Image with Kubernetes.**

* We are using Minikube to deploy docker image with Kubernetes. So we start the Minikube at beginning of this stage with sh 'minikube start’ command.
* Below script is used to deploy the application with Kubernetes.

sh 'kubectl run feedback-service --replicas=2 --labels="run=load-balancer-example" --image=kartikjalgaonkar/feedback-service --port=8084'

This command gets the docker image from docker repository described with

*--image* and then create the Kubernetes image of the application and run on the given port. It also creates the two replicas of the deployment.

* Then we expose the application over the internet as below.

sh 'kubectl expose deployment feedback-service --type=LoadBalancer --name=my-feedback-service'

It will assign the port and also creates the URL to access Feedback Service API over the network.

* We get this URL and Port using below command.

sh 'kubectl get services my-feedback-service'

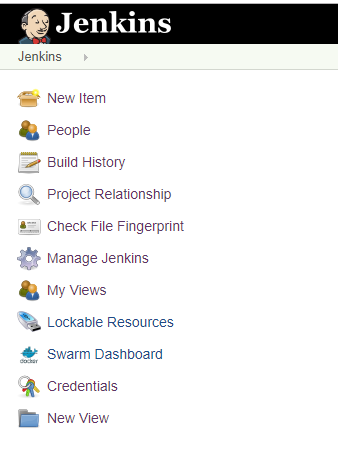
|  |
| --- |
| stage('kubectl deploy'){ |
|  | sh 'minikube start' |
|  | /\*sh 'kubectl delete deployment feedback-service' |
|  | sh 'kubectl delete svc feedback-service'\*/ |
|  | sh 'kubectl run feedback-service --replicas=2 --labels="run=load-balancer-example" --image=kartikjalgaonkar/feedback-service --port=8084' |
|  | sleep 60 |
|  | sh 'kubectl get deployments feedback-service' |
|  | sh 'kubectl describe deployments feedback-service' |
|  | sh 'kubectl get replicasets' |
|  | sh 'kubectl describe replicasets' |
|  | sh 'kubectl expose deployment feedback-service --type=LoadBalancer --name=my-feedback-service' |
|  | sh 'kubectl get services my-feedback-service' |
|  | sleep 100 |
|  | sh 'kubectl get services my-feedback-service' |
|  | sh 'kubectl describe services my-feedback-service' |
|  | sh 'kubectl get pods --output=wide' |
|  | sh 'minikube service my-feedback-service' |
|  | sh 'minikube dashboard' |
|  | } |

Add Credentials for Github and Docker in Jenkins

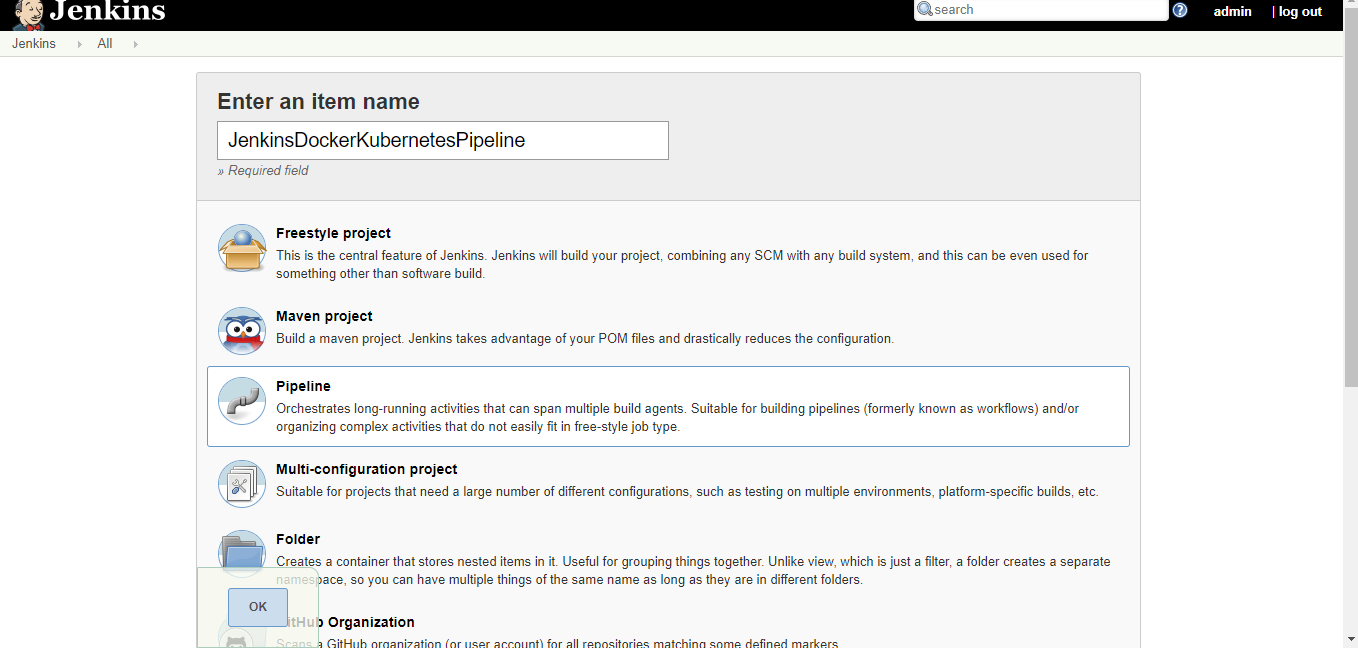
1. Ensure you are logged in to Jenkins (as a user with the **Credentials > Create** permission).
2. From the Jenkins home page (i.e. the Dashboard of the Jenkins classic UI), click **Credentials > System** on the left.
3. Under **System**, click the **Global credentials (unrestricted)** link to access this default domain.
4. Click **Add Credentials** on the left.
5. From the **Kind** field, choose **Username and password** for both GitHub and Docker Credentials.
6. From the **Scope** field choose **Global** for both GitHub and Docker Credentials.
7. Add the credentials themselves into the appropriate fields for your chosen credential type:
   * **Username and password** - specify the credential’s **Username** and **Password** in their respective fields.
8. In the **ID** field, specify a meaningful credential ID value - for example, docker\_credential.
9. Specify an optional **Description** for the credential/s.
10. Click **OK** to save the credentials.

Setup Jenkins CI/CD Pipeline for Deployment

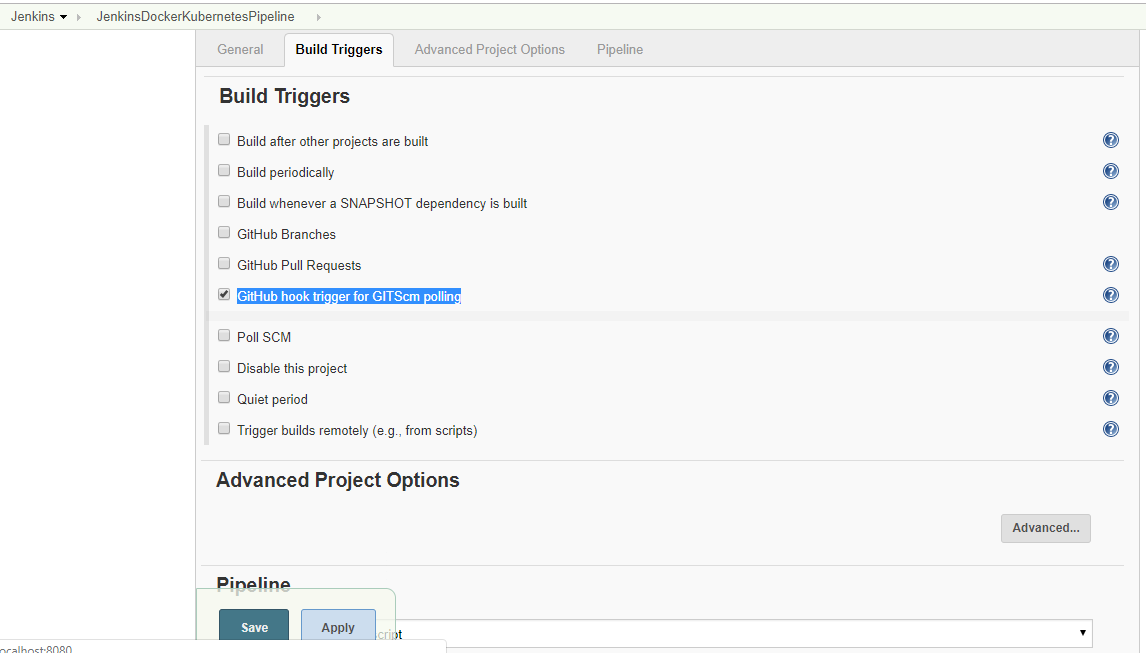
1. Click the **New Item** menu within Jenkins.



1. Provide a name for your new item (e.g. **JenkinsDockerKubernetesPipeline**) and select **Multi branch Pipeline**

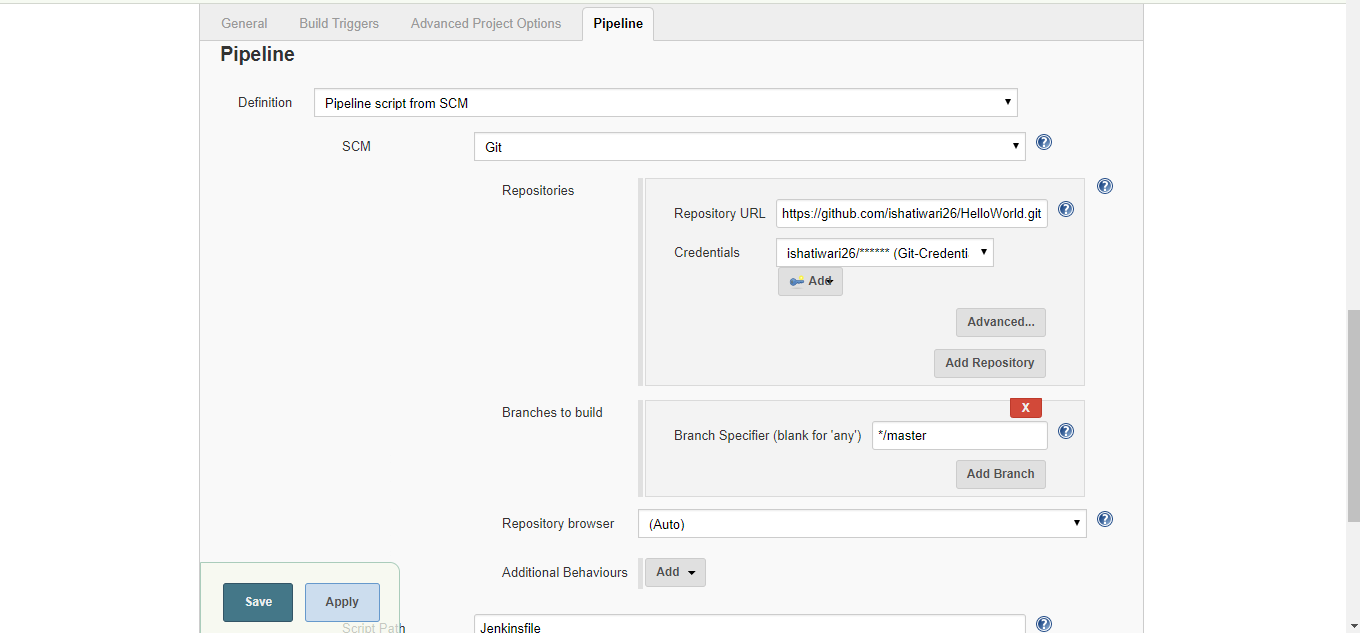


1. Under the Build Triggers tab click the **GitHub hook trigger for GITSCM polling** option.



1. Click the **Pipeline** tab and provide the appropriate values to configure git repository which we are going to deploy.

* **Definition:**  Select the **Pipeline** **script for SCM** option.
* **SCM:** Select the Git
* **Repositories:** Add the **Repository URL** and **Credential** for git Repository to be added. Select The **Branches to build.**
* **Script Path:** Add the name of the Jenkins script file which are added to project root path.



1. Click on Save and Apply.