****

**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

1. If required, ensure you are logged in to Jenkins (as a user with the **Credentials > Create** permission).

Installation and Setup of Docker

1. If required, ensure you are logged in to Jenkins (as a user with the **Credentials > Create** permission).

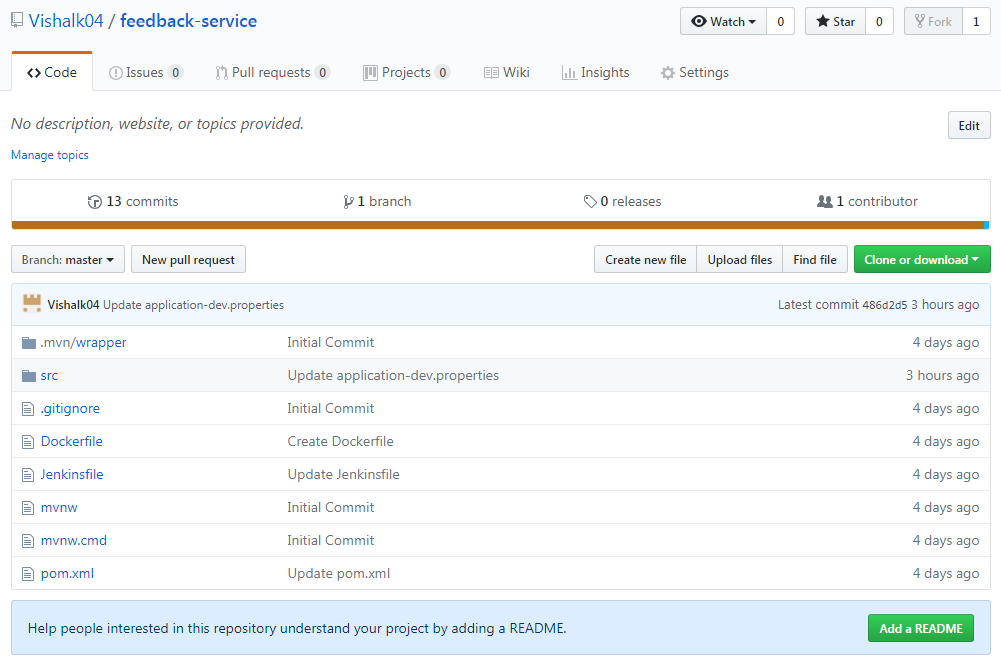
Installation and Setup of Minikube + Kubernetes

1. If required, ensure you are logged in to Jenkins (as a user with the **Credentials > Create** permission).

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.

|  |
| --- |
| 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**

In this stage compile the code with **mvn clean install*.***

The last line in below stage builds the docker image.

|  |
| --- |
| 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") |
|  | } |

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.

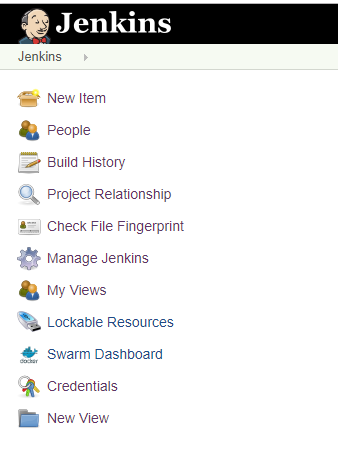
|  |
| --- |
| stage('kubectl deploy'){ |
|  | sh 'minikube start' |
|  | /\*sh 'kubectl delete deployment feedback-service' |
|  | sh 'kubectl delete svc feedback-service'\*/ |
|  | sh 'kubectl run feedback-service --replicas=2 --labels="run=load-balancer-example" --image=kartikjalgaonkar/feedback-service --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

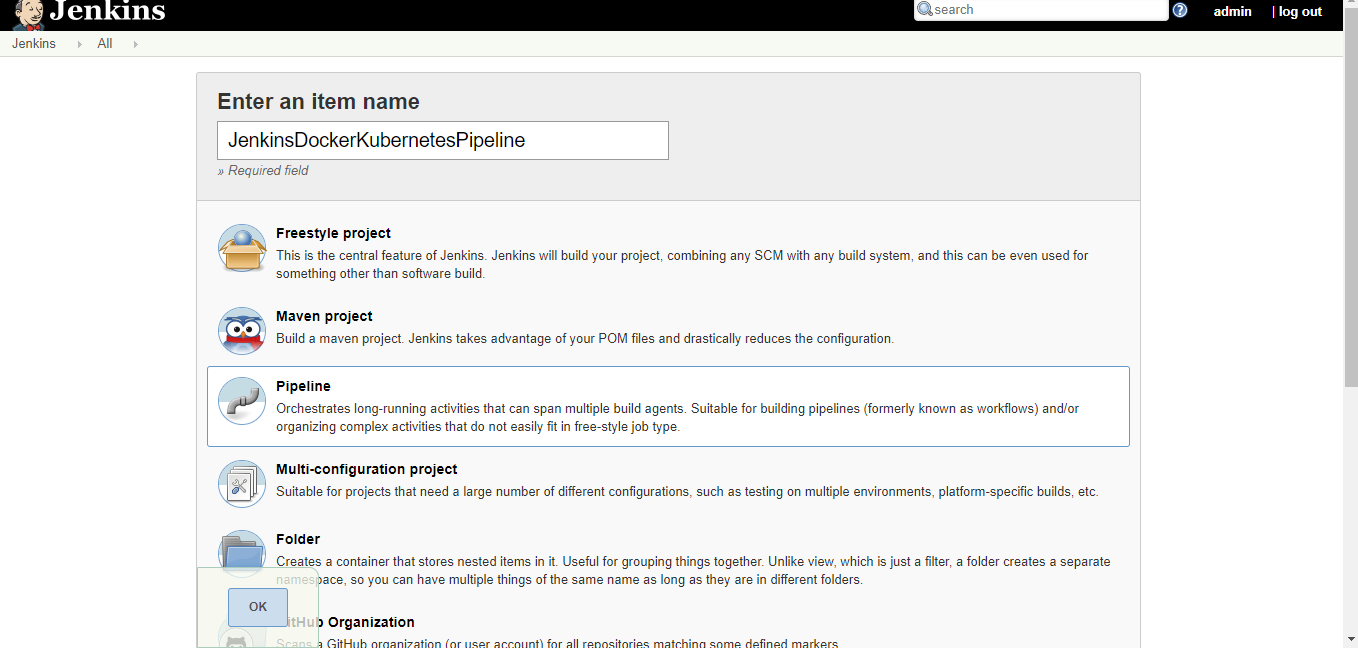
1. Ensure you are logged in to Jenkins (as a user with the **Credentials > Create** permission).
2. From the Jenkins home page (i.e. the Dashboard of the Jenkins classic UI), click **Credentials > System** on the left.
3. Under **System**, click the **Global credentials (unrestricted)** link to access 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

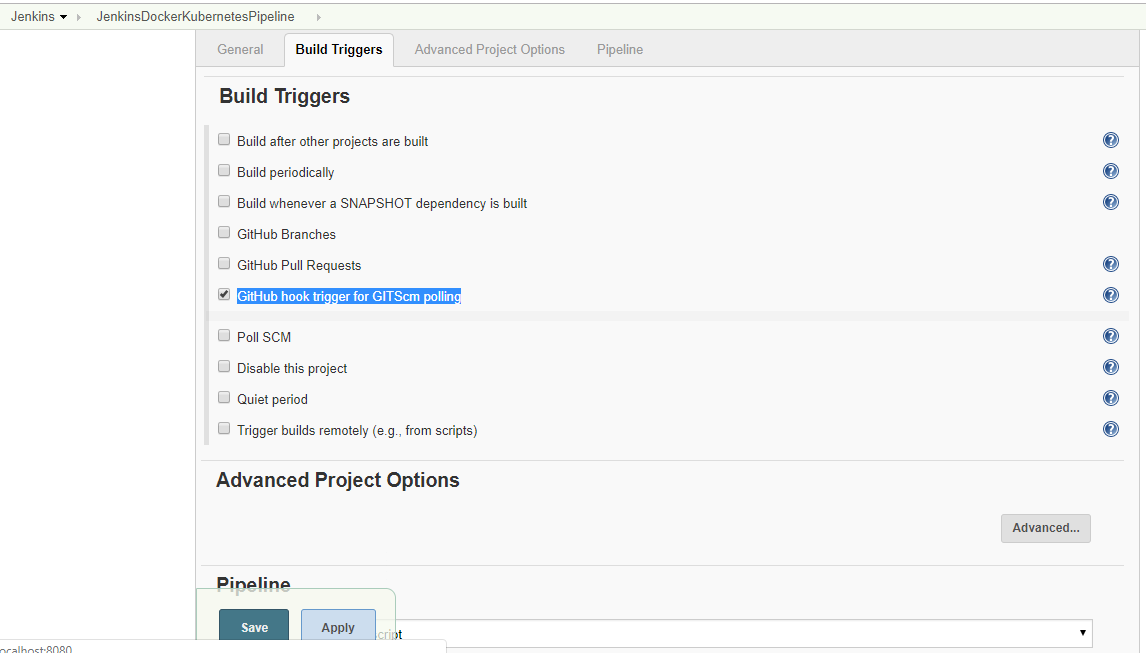
1. Click the **New Item** menu within Jenkins.



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

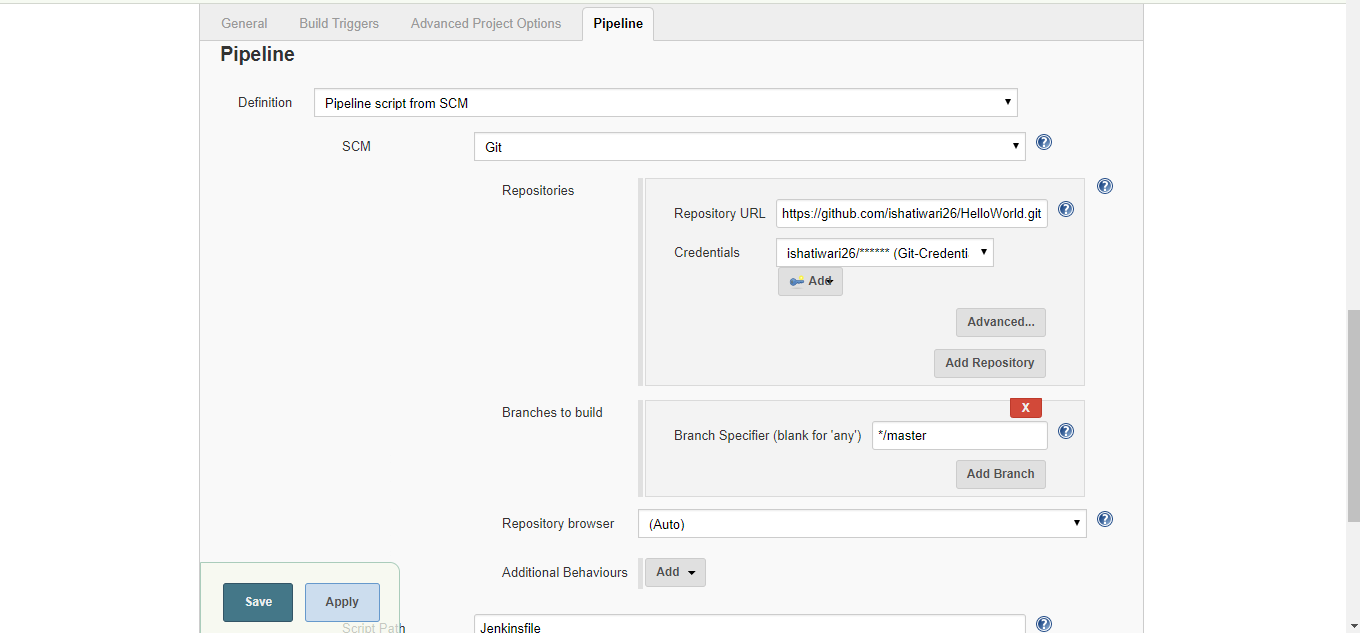


1. Under the Build Triggers tab click the **GitHub hook trigger for GITScm polling** option.



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 **Cedential** 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.