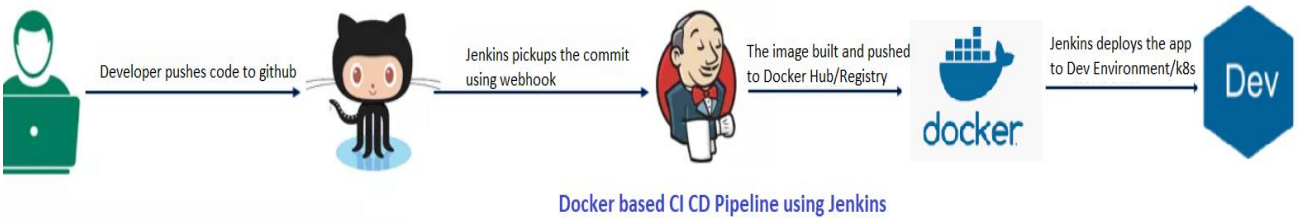
Jenkins-Docker Continuous Integration & Continuous Deployment Pipeline



1. Create Jenkins VM and install all the suggested plugins. **(VM1-Jenkins)**

2. Under Available Manage Plugins section, search for docker, there are multiple Docker plugins, docker compose, docker build plugins. Please install all docker plugins. **(VM1-Jenkins)**

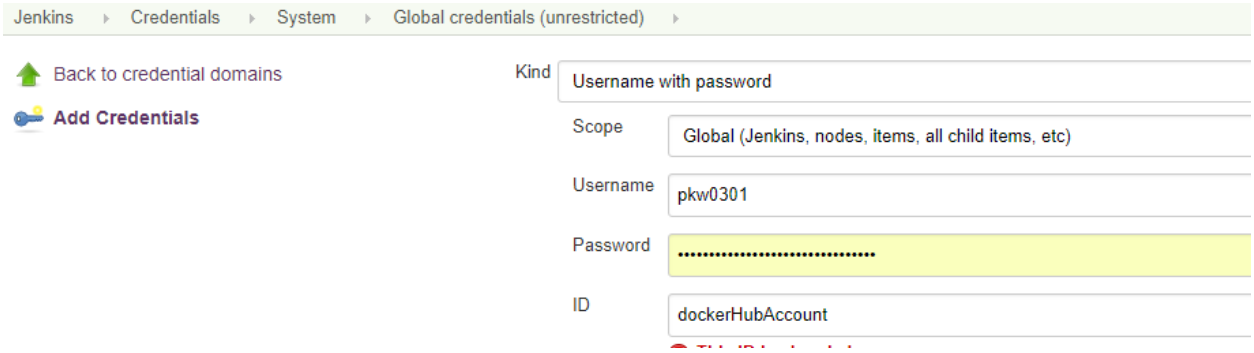
3. Install Docker on Jenkins instance so that we can create docker images. **(VM1-Jenkins)**

   yum install docker
   service docker start
   systemctl enable docker

4. Configure JDK and Maven. **(VM1-Jenkins)**

5. Since we will perform some operations such as checkout codebase and pushing an image to Docker Hub, we need to define the Docker Hub Credentials. **(VM1-Jenkins)**

   These definitions are performed under Jenkins Home Page -> Credentials ->Jenkins-> Global credentials -> Add Credentials menu and provide id and password and update ID as "dockerHubAccount"



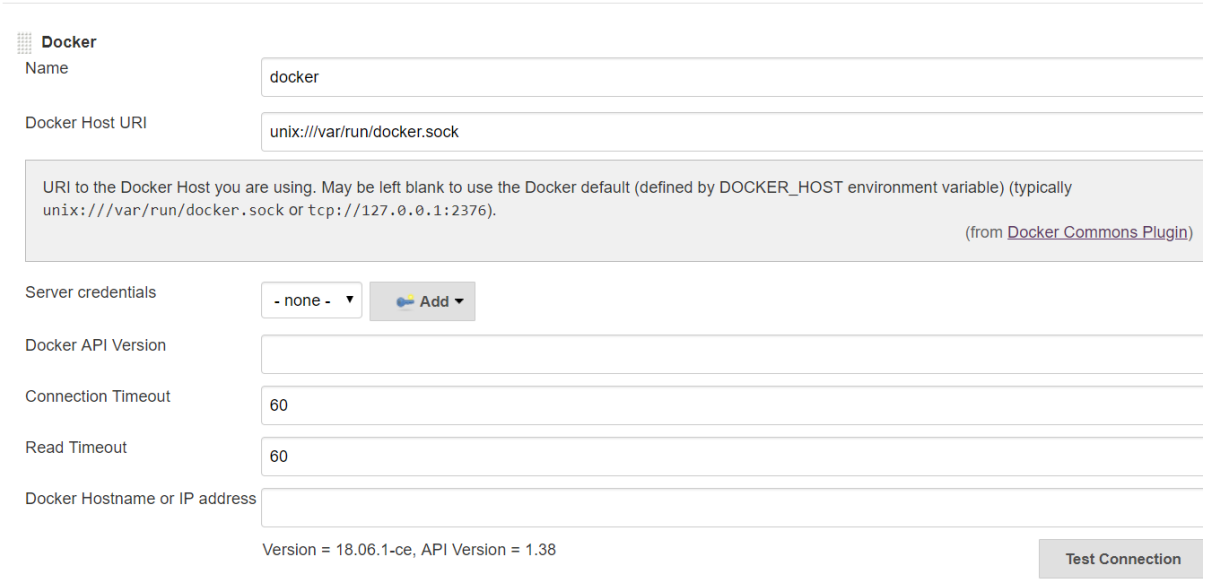
6. Select **Configure System** to access the main Jenkins settings. **(VM1-Jenkins)**

   At the bottom, there is a dropdown called **Add a new cloud**. Select **Docker** from the list. Now we need docker host uri, the docker.sock file is owned by root and does not allow write permissions by other. You have to make it such that Jenkins can read/write to that socket file when mounted.

To enable the docker host URL you need to change permission on your Jenkins ec2 **instance**.

chmod 777 /var/run/docker.sock

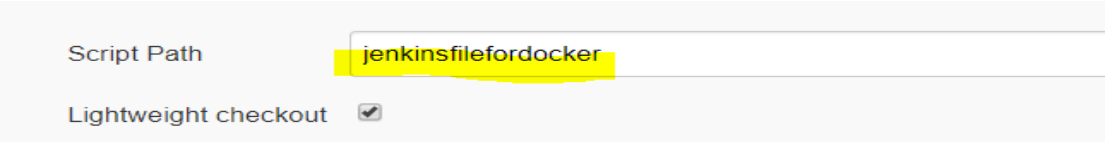
Once you change permission then at dashboard, under Docker Host URI put **unix://var/run/docker.sock** And click on Test Connection, it will display installed docker and API version.



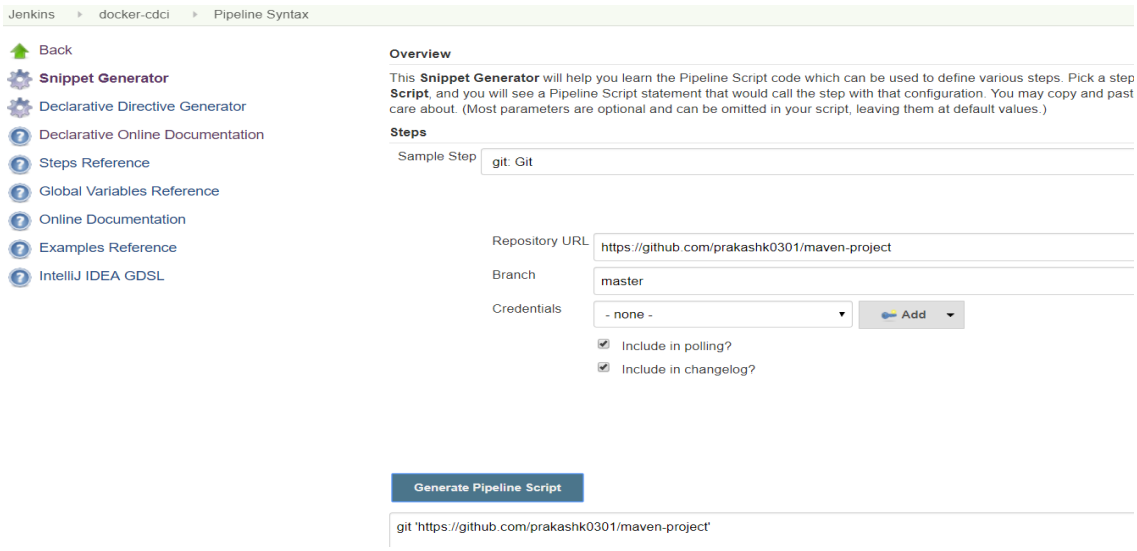
7. We have installed docker on Jenkins instance, now we are good to build docker images **(VM1-Jenkins)**

Create a **Pipeline** Jenkins job, you can name it as docker-ci-cd

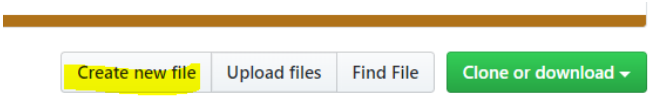
Source Code URL: https://github.com/prakashk0301/maven-project/ and Jenkinsfile name as jenkinsfilefordocker and save.



8. From Pipeline Syntax->select git under Sample Step. Provide all the details. **(VM1-Jenkins)**



9. Open Github repo and create a file called **jenkinsfilefordocker** and paste Pipeline syntax content and commit your changes. **(VM1-Jenkins)**




```
node{

  stage ('scm checkout') {

    git 'https://github.com/prakashk0301/maven-project'

  }

}
```

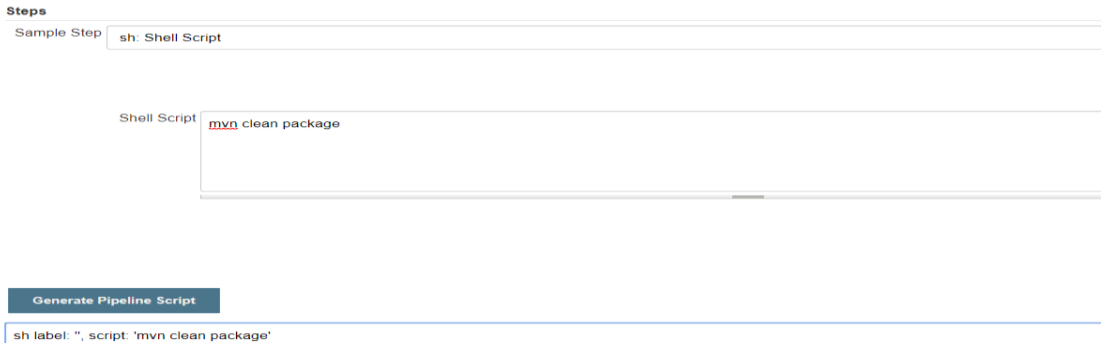
```

node{
    stage ('scm checkout') {
        git 'https://github.com/prakashk0301/maven-project'
    }


  }
```

10. Save and build your job (just to verify)

11. In case of multiple branches and if you want to build through to any specific branch then

```
node{
    stage ('scm checkout') {
       git 'https://github.com/prakashk0301/maven-project'
    }

        stage('Checkout to different branch') {
        sh "git checkout try-docker"
    }


 }
```

12. It's time to build the job and create a package. From Pipeline syntax search for **shell script,** and Create a script for maven package goal, and add script in docker-jenkinsfile file and build.




```
node{
  stage ('scm checkout') {
        git ('https://github.com/prakashk0301/maven-project')
        }
        stage ('Checkout to different branch') {
     sh "git branch -r"
     sh "git checkout master"
  }
    stage ('package stage') {
     sh label: '', script: 'mvn clean package '
    }
}
```

13. We can define a new stage for docker image creation. Our docker images will contains Jenkins job's artifact.

```
stage ('docker image build') {

   sh 'docker build -t pkw0301/prakash-app:1.0.0 .'

          }
```

```
node{
    stage ('scm checkout') {
        git ('https://github.com/prakashk0301/maven-project')
        }

        stage('Checkout to different branch') {
           sh "git branch -r"
           sh "git checkout try-docker"
    }
        stage ('package stage') {
           sh label: '', script: 'mvn clean package '
        }

        stage ('docker image build') {
           sh 'docker build -t pkw0301/prakash-app:1.0.0 .'
        }

 }
```

In order to copy Jenkins job artifact we have to write a Dockerfile, please keep it with your source code.

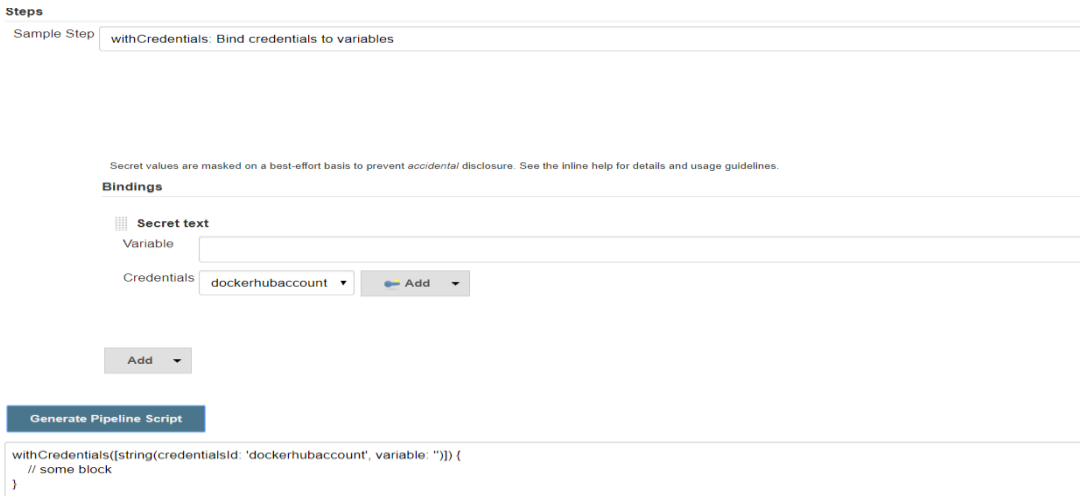
```
FROM tomcat:8

COPY /webapp/target/*.war /usr/local/tomcat/webapps/
```

server	file created
webapp	Update index.jsp
.gitignore	file created
Dockerfile	Create Dockerfile
README.md	file created
jenkinsfilefordocker	Update and rename
pom.xml	Update pom.xml

14. Save and build job, ssh your Jenkins instance and verify Docker images.(If you get any permission denied error then please follow below commands and re-run your job) **(VM1-Jenkins)**

```
chmod 777 /var/run/docker.sock
docker images
```

15. Push Docker image to Dockerhub/Registry. To do that we have to generate a script. **Pipeline syntax**-> select **withcredentials** from drop down -> secret text - > select **dockerHubaccount ->** generate script **(VM1-Jenkins)**



Add stage to push docker images

```
stage ('Push Docker image to DockerHub') {
  withCredentials([string(credentialsId: 'dockerhubaccount', variable: 'dockerhubaccount')]) {
        sh "docker login -u pkw0301 -p ${dockerhubaccount}"
    }
        sh 'docker push pkw0301/prakash-app:1.0.0'

    }
```

Please provide your dockerhub id during login login command, so that you also can login and push images. Once you add stage, build you job. Verify docker images by login to dockerhub.

16. Now we have pushed docker images to dockerhub, In order to deploy these images to Development/QA/Prod environment, we need instaces

Create an ec2 instance for Development environment and install docker. **(VM2-Dev)**

```
yum install docker

service docker start

systemctl enable docker

chmod 777 /var/run/docker.sock
```

From Pipeline syntax select sshagent->Add->Jenkins->

Kind: SSH Username with private key

ID: deploy-to-dev-docker

Username: ec2-user

Private key: please paste pem key of dev instance

Update your Jenkinsfile and add a new stage "deploy-to-dev"

```
stage ('Deploy to Dev') {
    def dockerRun = 'docker run -d -p 9000:8080 --name my-tomcat-app pkw0301/prakash-app:1.0.0'
    sshagent(['deploy-to-dev-docker']) {
        sh "ssh -o StrictHostKeyChecking=no ec2-user@172.31.46.1 ${dockerRun}"
    }
}
```

Save your changes and build.

You can access docker based application by the following URL.

<VM2 dev public ip>:9000/webapp

Done ☺

(You also can refer my jenkinsfile for docker ci/cd

Github URL: https://github.com/prakashk0301/maven-project/tree/ci-cd-with-docker

Branch: ci-cd-with-docker)