

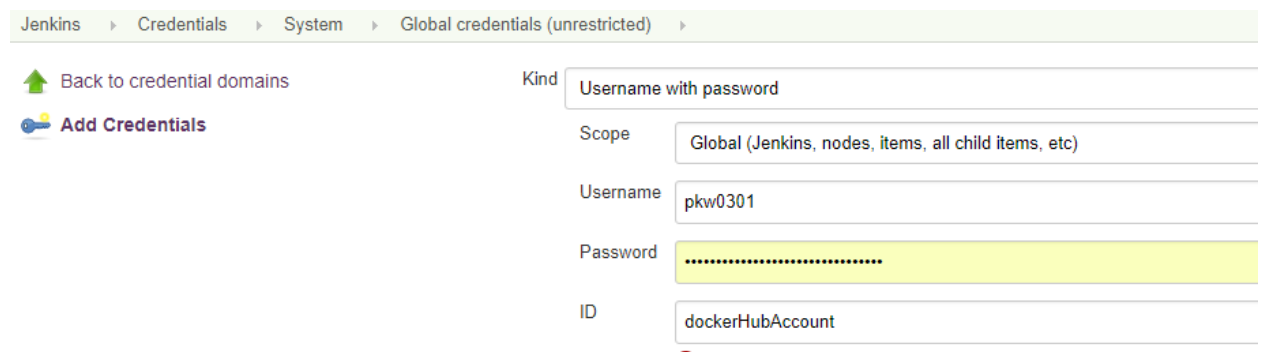
Docker Jenkins Integration for Local Testing

1. Create Jenkins VM and install all the suggested plugins.
2. Under Available Manage Plugins section, search for docker, there are multiple Docker plugins, docker compose, docker build plugins. Please install all docker plugins.
3. Install Docker on Jenkins instance so that we can create docker images.

```
yum install docker
service docker start
systemctl enable docker
```

4. Configure JDK and Maven.
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.

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"



The screenshot shows the Jenkins web interface for configuring global credentials. The breadcrumb trail at the top is 'Jenkins > Credentials > System > Global credentials (unrestricted)'. On the left, there are links for 'Back to credential domains' and 'Add Credentials'. The main form is titled 'Kind: Username with password'. It contains the following fields: 'Scope' set to 'Global (Jenkins, nodes, items, all child items, etc)', 'Username' set to 'pkw0301', 'Password' (masked with dots), and 'ID' set to 'dockerHubAccount'.

6. Select **Configure System** to access the main Jenkins settings.

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.

Docker

Name:

Docker Host URI:

URI to the Docker Host you are using. May be left blank to use the Docker default (defined by DOCKER_HOST environment variable) (typically unix:///var/run/docker.sock or tcp://127.0.0.1:2376). (from [Docker Commons Plugin](#))

Server credentials:

Docker API Version:

Connection Timeout:

Read Timeout:

Docker Hostname or IP address:

Version = 18.06.1-ce, API Version = 1.38

7. Now we have done Jenkins docker integration, in order to test we will create Jenkins job.

Create a new job, give the job a name such as Jenkins Demo, select **Freestyle project** then click OK.

(The build will depend on having access to Docker, though we already have provided access to Jenkins so we are good to go)

Source code url: <https://github.com/prakashk0301/docker-jenkins-demo.git>

We can now add a new build step using the **Add Build Step** dropdown. Select **Execute Shell**.

Because the logical of how to build is specified in our Dockerfile, Jenkins only needs to call build and specify a friendly name.

In demo, use the following commands.

Prakash-Ethans-Pune

```
docker build -t pkw0301/jenkins-demo:${BUILD_NUMBER} .
docker tag pkw0301/jenkins-demo:${BUILD_NUMBER} pkw0301/jenkins-demo:latest
docker run -d -p 12454:80 --name dockertest pkw0301/jenkins-demo:latest
docker rmi -f pkw0301/jenkins-demo:${BUILD_NUMBER}
```

Save and build:

Login to your Jenkins Instance and verify docker images and container.

Once your build done you can access sample http <VM>:12454

Done 😊

Explanation of above commands:

docker build -t pkw0301/jenkins-demo:\${BUILD_NUMBER} . = build a docker image and assign a tag as pkw0301/(build number)

docker tag pkw0301/jenkins-demo:\${BUILD_NUMBER} pkw0301/jenkins-demo:latest = changes tag to latest

docker run -d -p 12454:80 --name dockertest pkw0301/jenkins-demo:latest = run a container

docker rmi -f pkw0301/jenkins-demo:\${BUILD_NUMBER} = delete build number docker image to save disk space , because we already have retagged docker images as latest so there is no use of BUILD_Number docker images.

(If you want to auto delete old containers then edit your execute shell command)

```
docker rm -f dockertest
```

```
docker build -t pkw0301/jenkins-demo:${BUILD_NUMBER} .
```

```
docker tag pkw0301/jenkins-demo:${BUILD_NUMBER} pkw0301/jenkins-demo:latest
```

```
docker run -d -p 12454:80 --name dockertest pkw0301/jenkins-demo:latest
```

```
docker rmi -f pkw0301/jenkins-demo:${BUILD_NUMBER}
```