What is Jenkins?

Jenkins is an open-source automation server that you can use to automate tasks related to software development, testing, or deployments.

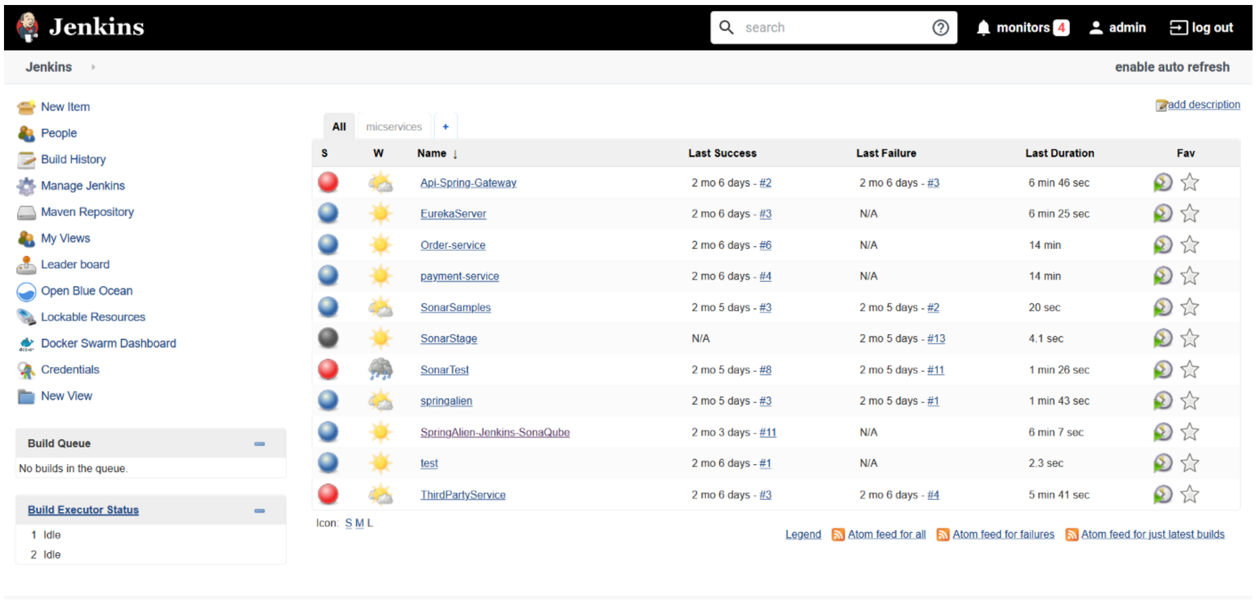
What is Docker?

Docker is an open-source software platform built to make it easier to create, deploy, and run applications using containers. A container is a standardized package of software, which can include libraries and other dependencies.

What is Kubernetes?

Kubernetes is an open-source container orchestration platform for automating deployment and scaling and managing containerized workloads and services.

After successful installation of Jenkins does register and login to the Jenkins dashboard with the help of a master key generated inside the directory created by Jenkins. The dashboard of Jenkins will look like this.



No builds in the queue.

1 Idle
2 Idle

S	W	Name ↓	Last Success	Last Failure	Last Duration	Fav
🔴	⛅	Api-Spring-Gateway	2 mo 6 days - #2	2 mo 6 days - #3	6 min 46 sec	☆
🔵	☀	EurekaServer	2 mo 6 days - #3	N/A	6 min 25 sec	☆
🔵	☀	Order-service	2 mo 6 days - #6	N/A	14 min	☆
🔵	☀	payment-service	2 mo 6 days - #4	N/A	14 min	☆
🔵	⛅	SonarSamples	2 mo 5 days - #3	2 mo 5 days - #2	20 sec	☆
⚫	☀	SonarStage	N/A	2 mo 5 days - #13	4.1 sec	☆
🔴	🌧	SonarTest	2 mo 5 days - #8	2 mo 5 days - #11	1 min 26 sec	☆
🔵	⛅	springalien	2 mo 5 days - #3	2 mo 5 days - #1	1 min 43 sec	☆
🔵	☀	SpringAlien-Jenkins-SonaQube	2 mo 3 days - #11	N/A	6 min 7 sec	☆
🔵	☀	test	2 mo 6 days - #1	N/A	2.3 sec	☆
🔴	⛅	ThirdPartyService	2 mo 6 days - #3	2 mo 6 days - #4	5 min 41 sec	☆

Basic Requirements

ssh plugin

GIt or Bitbucket Plugin

A bastion host has access to AWS ECR and Kubernetes.

Jenkins - Install the “SSH Pipeline Steps” plugin and “Gradle”

SSH Pipeline Steps

Moving ahead we need to install one more plugin SSH Pipeline Steps which we are going to use for SSH into the bastion server.

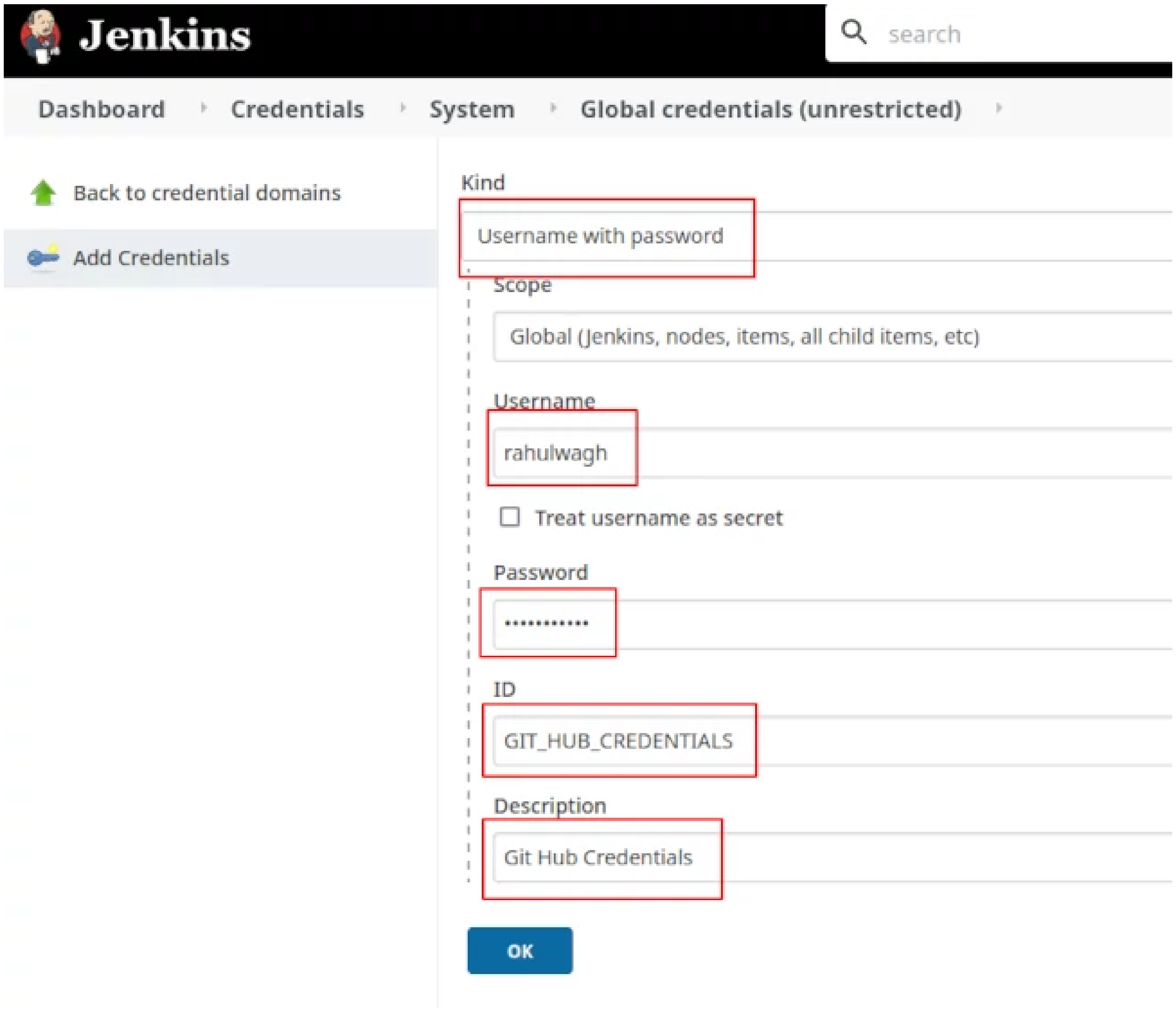
For installing the plugin please go to - Manage Jenkins -> Manage Plugin -> Available then in the search box type SSH Pipeline Steps.

Select the plugin and install it without restarting.

Setup GitHub Username and password into Jenkins

Now we add one more username and password for GitHub.

Go to -> Jenkins -> Manage Jenkins -> Manage Credentials -> Stored scoped to jenkins -> global -> Add Credentials



Kind: Username with password

Scope: Global (Jenkins, nodes, items, all child items, etc)

Username: rahulwagh

☐ Treat username as secret

Password: ••••••••••••

ID: GIT_HUB_CREDENTIALS

Description: Git Hub Credentials

Create Jenkins Pipeline

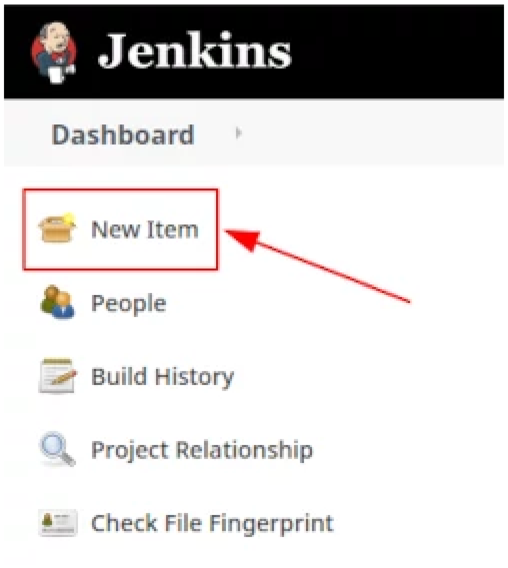
The first step for you would be to create a pipeline.

Goto: Jenkins -> New Items

Enter an item name: test

Select Freestyle

Click Ok



Build a modified Jenkins image

Early on, Jenkins was designed to run on physical machines without any containerization technology. As containerization become more popular, Jenkins adapted its solution to the new containerized world. Unfortunately, this change brought some challenges. For instance, Jenkins requires Docker to build Docker images. However, a containerized version of Jenkins does not contain Docker and Docker CLI by default. For this reason, a new Docker image that contains Docker CLI and other tools must be created by using a Jenkins image as a base image. To do this, Docker uses Dockerfile to build custom images. In the modified-Jenkins directory, there is a Dockerfile to build a custom modified Jenkins image. I'll break this down, step-by-step.

The base image should look something like this:

```
FROM node:14
WORKDIR /usr/src/app
ENV NODE_ENV test
COPY package*.json ./
RUN npm i
COPY . .
CMD ["npm", "start"]
```

here in the docker file we are taking the base image as a node and inside the image, we are going to create the working directory by name /usr/src/app and we are passing the env test.

we are coping the package folder to the current working directory of the image

we are installing the dependencies and coping it from local to the image.

then we are passed the command which we will run whenever our new container will be created.

with the help of this dockerfile we are going to build the new docker image.

```
docker build -t 0XXXXXXXX.dkr.ecr.ap-south-1.amazonaws.com/demo:1.${BUILD_NUMBER} .
```

here we are adding the build number for tagging so that later if we need we can roll back.

Docker Login via CLI

Since I am working inside Jenkins so every step I perform I need to write a pipeline script. Now after building and tagging the Docker Image we need to push it to the AWS ECR. But before you push to DockerHub you need to authenticate yourself via CLI(command-line interface) using docker login

So here is the pipeline step for Docker Login

```
aws ecr get-login-password --region ap-south-1 | docker login --username AWS
--password-stdin  XXXXXXXXXX21.dkr.ecr.ap-south-1.amazonaws.com
```

Push Docker Image into ECR

After successful Docker login now we need to push the image to ECR

```
docker push 0XXXXXXXX21.dkr.ecr.ap-south-1.amazonaws.com/demo:1.${BUILD_NUMBER}
```

SSH Into the bastion server

If you remember we have installed SSH Pipeline Steps, now we are going to use that plugin to SSH into the bastion host which will have access to all Kubernetes stuff.

```
sh  -o StrictHostKeyChecking=no ubuntu@x.x.x.x << EOF
```

Set New Image To EKS Deployment

```
kubectl set image deploy demo
demo=x.x.x.x.dkr.ecr.ap-south-1.amazonaws.com/demo:1.${BUILD_NUMBER}
```

Rollout The Deployment

```
kubectl rollout restart deploy demo
```

Here demo is the name of the deployment and the name of the container.

Build and Run Pipeline:

I hope you enjoyed this article.