

# Installing Jenkins using Docker containers on linux

## Run the following into the Dockerfile on linux machine:

```
FROM jenkins/jenkins:2.414.2-jdk17
USER root
RUN apt-get update && apt-get install -y lsb-release python3-pip
RUN curl -fsSL /usr/share/keyrings/docker-archive-keyring.asc \
https://download.docker.com/linux/debian/gpg \
  signed-by=/usr/share/keyrings/docker-archive-keyring.asc \
https://download.docker.com/linux/debian \
  $(lsb_release -cs) stable" > /etc/apt/sources.list.d/docker.list
RUN apt-get update && apt-get install -y docker-ce-cli
USER jenkins
RUN jenkins-plugin-cli --plugins "blueocean:1.25.3 docker-workflow:1.28"
```

## Build the image from this:

```
docker build -t myjenkins-app-1 .
```

## Create docker network:

```
docker network create jenkins
```

## Creating docker container:

```
docker run --name jenkins-app --restart=on-failure --detach \
  --network jenkins --env DOCKER_HOST=tcp://docker:2376 \
  --env DOCKER_CERT_PATH=/certs/client --env DOCKER_TLS_VERIFY=1 \
  --publish 8080:8080 --publish 50000:50000 \
  --volume jenkins-data:/var/jenkins_home \
  --volume jenkins-docker-certs:/certs/client:ro \
  myjenkins-app-1
```

## Check the docker container:

```
docker ps
```

## login to web browser:

```
127.0.0.1:8080
```

## Execute the below cmd to list PWD for jenkins

```
docker exec jenkins-app cat /var/jenkins_home/secrets/initialAdminPassword
```

--- copy the password & paste it into the web browser.

## While installation:

```
# install suggested plugins.
```

```
# give
```

```
- username
```

- password
- name
- save & continue.

## DEMO - 1:

### On Web browser

- Browse through the UI.
- Create a new/first job
- Without filling GitHub URL.
- In BUILD, select "execute shell" & write some commands like:
  - o `echo "Hello world"`
    - Save & build project
    - Check the console logs
- Then edit the shell again with environmental variables
  - o `echo "Build ID is: ${BUILD_ID}"`
  - o `echo "Build URL is: ${BUILD_URL}"`
    - Save & build project
    - Check the console logs
- Then edit the shell again:
  - o Create file and verify it on browser & container location as well.
    - `echo "this is Jenkins Pipeline" > ReadMe.txt`
    - Save & build project
    - Check the console logs

Now, login to docker container

**To access the file system in jenkins container within docker**

```
docker exec -it jenkins-app bash
```

Execute commands:

```
# pwd
# ls -lrt
# cat ReadMe.txt
```

**# go to**

```
/var/jenkins_home/workspace/my-first-proj-1
```

## DEMO - 2 :- Fetching info from GitHub account

- Create a new job.
- Provide the GitHub URL in the source code management under 'repository URL'.  
<https://github.com/jitendrastomar5593/gitdemo>
- In Build, select 'execute shell' & run below command.  
Python3 helloworld.py
  - Save & check.
  - Build it.
  - Check console.

=====

Dashboard --> Manage Jenkins

--> Plugins

- Install blueOcean plugins.

--> Manage Nodes & Clouds --> cloud providers

- Docker
- Amazon EC2 (time taking process)
  - Check Install after restart.

# Integrating GitHub with Jenkins on AWS EC2 (Ubuntu)

Saturday, March 16, 2024

11:16 PM

1. Create an Ubuntu VM on any cloud (AWS) with port number
  - a. 80
  - b. 443
  - c. 8080 (for Jenkins application)
  - d. 22
2. Connect to the EC2 instance using SSH via
  - a. Putty
  - b. MT Putty
  - c. MobaXterm
  - d. PowerShell
  - e. Windows Terminal
3. Update the ubuntu server and install docker on it.
  - a. Set up Docker's apt repository

```
# Add Docker's official GPG key:
```

```
sudo apt-get update -y
```

```
sudo apt-get install ca-certificates curl -y
```

```
sudo install -m 0755 -d /etc/apt/keyrings
```

```
sudo curl -fsSL https://download.docker.com/linux/ubuntu/gpg -o /etc/apt/keyrings/docker.asc
```

```
sudo chmod a+r /etc/apt/keyrings/docker.asc
```

```
# Add the repository to Apt sources:
```

```
echo \
```

```
"deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.asc]
```

```
https://download.docker.com/linux/ubuntu \
```

```
$(. /etc/os-release && echo "$VERSION_CODENAME") stable" | \
```

```
sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

```
sudo apt-get update -y
```

- b. Install docker packages

```
apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin -y
```

- c. Verify that the Docker Engine installation is successfully

```
docker run hello-world
```

```
# systemctl enable docker
```

```
# systemctl start docker
```

```
# usermod -a -G docker <ubuntu>
```

- d. Pulling Jenkins image from dockerhub.

```
# docker pull jenkins/jenkins
```

- e. Listing images:

```
# docker images
```

- f. Create a new directory:

```
# mkdir Jenkins
```

- g. Running Jenkins container on port 8080

```
# docker run -d --name jenkins -p 8080:8080 -v $PWD/jenkins/ jenkins/jenkins
```

```
# docker ps
```

- h. Login to <EC2-INSTANCE-PUBLIC-IP>:<PORT-NUMBER>

```
http://<ec-instance>:<8080>
```

```
Unlock the Jenkins using
```

```
# docker ps
```

```
# docker exec -it <e580724f6f66> cat /var/jenkins_home/secrets/initialAdminPassword
```

```
root@ip-172-31-41-154:~# docker exec -it 05d0724f8164 cat /var/jenkins_home/secrets/initialAdminPassword
097cfe54c27641798b403512e4644b51
```

Copy this password and paste it to browser page.

Select **install suggested plugins**:

A screenshot of the Jenkins 'Getting Started' page. The page has a light gray header with the text 'Getting Started' and a small 'x' icon. The main heading is 'Customize Jenkins' in a large, bold, black font. Below it, a paragraph states: 'Plugins extend Jenkins with additional features to support many different needs.' At the bottom, there are two light blue rounded rectangular boxes. The left box is titled 'Install suggested plugins' and contains the text 'Install plugins the Jenkins community finds most useful.' The right box is titled 'Select plugins to install' and contains the text 'Select and install plugins most suitable for your needs.'

Wait for it:

## Getting Started

### Getting Started

<input checked="" type="checkbox"/> Folders	<input checked="" type="checkbox"/> OWASP Markup Formatter	<input checked="" type="checkbox"/> Build Timeout	<input type="checkbox"/> Credentials Binding
<input type="checkbox"/> Timestampers	<input type="checkbox"/> Workspace Cleanup	<input type="checkbox"/> Ant	<input type="checkbox"/> Gradle
<input type="checkbox"/> Pipeline	<input type="checkbox"/> GitHub Branch Source	<input type="checkbox"/> Pipeline: GitHub Groovy Libraries	<input type="checkbox"/> Pipeline Graph View
<input type="checkbox"/> Git	<input type="checkbox"/> SSH Build Agents	<input type="checkbox"/> Matrix Authorization Strategy	<input type="checkbox"/> RAM Authentication
<input type="checkbox"/> LDAP	<input type="checkbox"/> Email Extension	<input type="checkbox"/> Mailer	<input type="checkbox"/> Dark Theme

```

** Enable API
Folders
OWASP Markup Formatter:
** API API
** OWASP Policy API
** Structure
** Pipeline Step API
** Token Reuse
Build Timeout
** Credentials
** State Credentials
** Scan API
** Trained API
  
```

Fill the below details:

Getting Started

## Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Getting Started

## Create First Admin User

Username

Password

Confirm password

Full name

E-mail address

Save & continue.

Check the URL: <http://<IP-OF-EC2-INSTANCE>:8080/>

Getting Started

## Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the `ACLUZ_URL` environment variable provided to build steps.

The proposed default value shown is not saved yet and is generated from the current request. If possible, the best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Save & finish.

# Jenkins is ready!

Your Jenkins setup is complete.



Start using Jenkins

Switch to GitHub & create a new public repository

## Create a new repository


A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Required fields are marked with an asterisk (\*).

Owner \*  jitendrastomar5593 / Repository name \*    
  jenkinsgitrepo is available.

Great repository names are short and memorable. Need inspiration? How about [fantastic-octo-umbrella](#) ?

Description (optional)

- ☒  **Public**  
Anyone on the internet can see this repository. You choose who can commit.
- ☐  **Private**  
You choose who can see and commit to this repository.

Initialize this repository with:

- ☒ **Add a README file**  
This is where you can write a long description for your project. [Learn more about READMEs](#).

Add .gitignore

.gitignore template: **None**

Choose which files not to track from a list of templates. [Learn more about ignoring files](#).

Choose a license

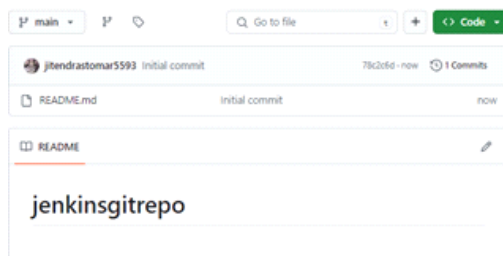
License: **None**

A license tells others what they can and can't do with your code. [Learn more about licenses](#).

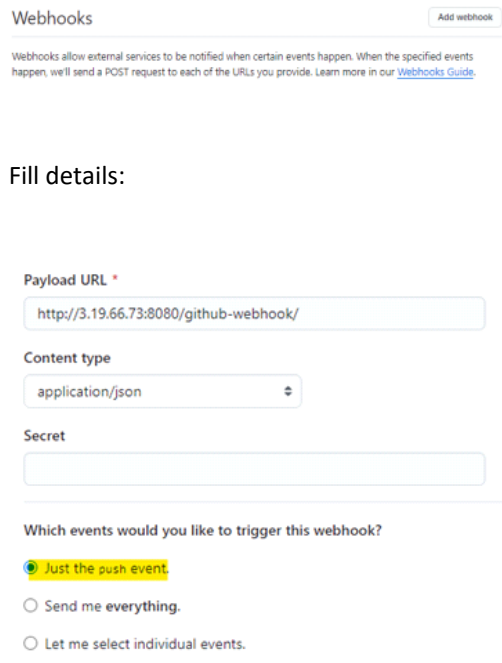
This will set **main** as the default branch. Change the default name in your [settings](#).

 You are creating a public repository in your personal account.

Create repository



On GitHub, go to “settings” & then “webhooks”. Click “Add webhook”.

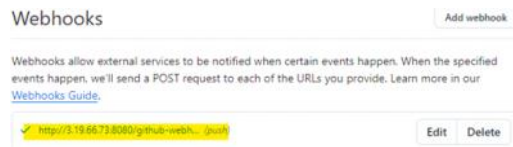


Payload URL → <http://<jenkins-svr-public-IP>:<port>/github-webhook/>

Content type → application/json

Which events would you like to trigger this webhook → Just the push event

Ensure “Active” is check marked. & click on “Add webhook”. & wait for some time.



Now switch back Jenkins portal, Jenkins dashboard: - Create “new item”.

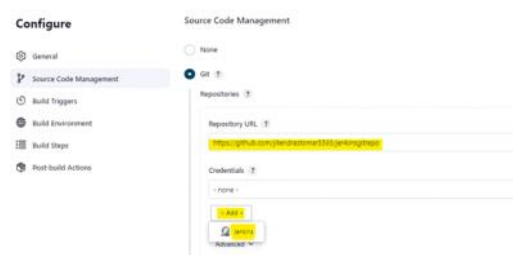


Enter an item name:



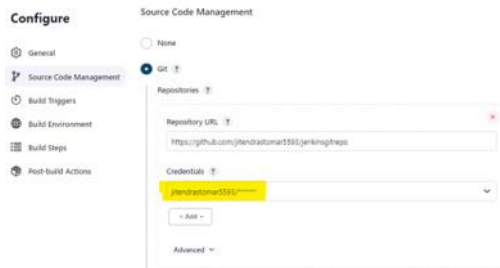
Click OK.

Under “Configure” select “source code management”, select GIT & fill ‘repository URL’.

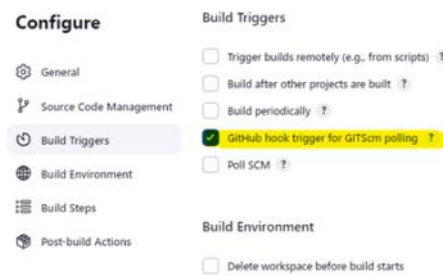


Select “Jenkins” & under credentials, select username & password & fill those.

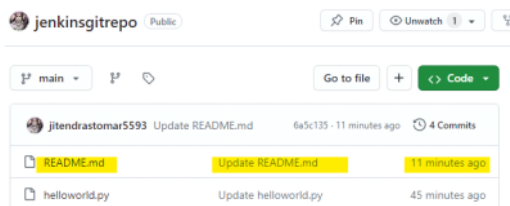




Click on “Build Triggers”. Select GitHub hook.



Click “Build Steps”, Nothing to select and save. Switch back to GitHub & edit the README.md file & commit it.



Go back to jenkins dashboard page & wait for build history to show up, automatically.



