

Day 24,25 : Complete Jenkins CI/CD Project

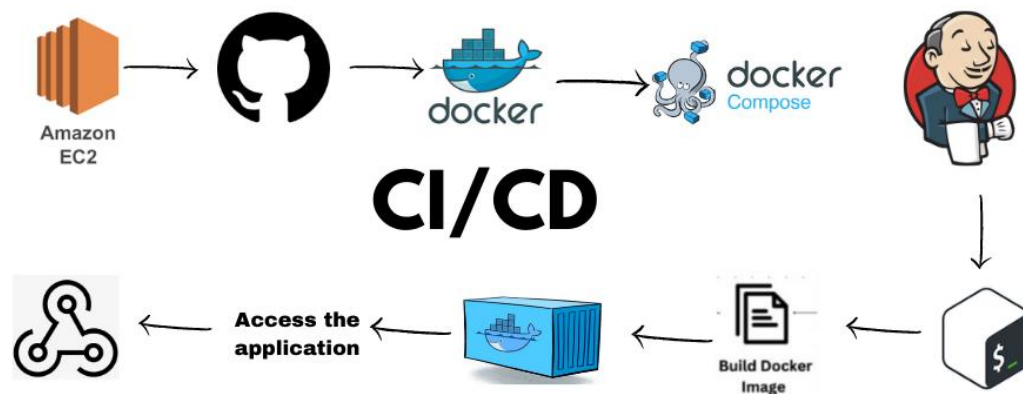
This is [#90DaysOfDevops](#) challenge under the guidance of [Shubham Londhe](#) sir.

Day 24,25 TASK

check this for task:

<https://github.com/LondheShubham153/90DaysOfDevOps/blob/master/2023/day24/tasks.md>

<https://github.com/LondheShubham153/90DaysOfDevOps/blob/master/2023/day25/tasks.md>



In this article we will be deploying a node.js application on EC2 instance and we will create a CICD pipeline using Jenkins.

Tools we will be using in the project:

1. AWS-EC2
2. GitHub
3. Docker
4. Jenkins

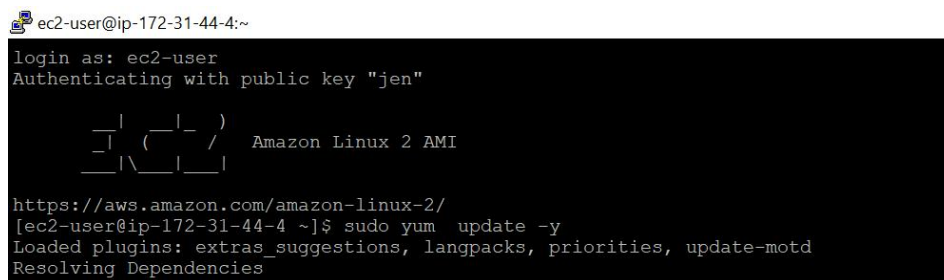
What is CICD pipeline?

CI/CD (Continuous Integration/Continuous Deployment) pipeline is a set of automated processes that helps to integrate code changes, build, test, and deploy applications continuously. The primary goal of a CI/CD pipeline is to enable fast and reliable delivery of changes to production.

A typical CI/CD pipeline consists of several stages, including:

1. **Code Integration:** In this stage, developers integrate their code changes into a shared repository.
2. **Build:** In this stage, the CI system builds the code and runs any necessary tests.
3. **Test:** In this stage, the code is tested using various testing techniques, including unit tests, integration tests, and end-to-end tests.
4. **Deployment:** In this stage, the code is deployed to production or a staging environment.
5. **Monitoring:** In this stage, the system monitors the deployed application for performance and stability.

Step 1: Create EC2 instance and ssh into it

A terminal window showing an SSH session. The prompt is 'ec2-user@ip-172-31-44-4:~'. The user enters 'login as: ec2-user' and 'Authenticating with public key "jen"'. The terminal displays the Amazon Linux 2 AMI logo and the URL 'https://aws.amazon.com/amazon-linux-2/'. The user then enters '[ec2-user@ip-172-31-44-4 ~]\$ sudo yum update -y'. The terminal shows 'Loaded plugins: extras_suggestions, langpacks, priorities, update-motd' and 'Resolving Dependencies'.

Step 2: Install Jenkins on the server

Use the below link for the installation.

<https://www.trainwithshubham.com/blog/install-jenkins-on-aws>

Jenkins is Accessible

Getting Started

Unlock Jenkins

To ensure Jenkins is securely set up by the administrator, a password has been written to the log ([not sure where to find it?](#)) and this file on the server:

`/var/lib/jenkins/secrets/initialAdminPassword`

Please copy the password from either location and paste it below.

Administrator password

Continue

Getting Started

Jenkins is ready!

Your Jenkins setup is complete.

Start using Jenkins

Before starting with the Jenkins configurations we need to add the public key so that we can create a bridge between jenkins and Github for accessing the source code from the Github repo.

Go to your EC2 instance and type command

```
#ssh-keygen
```

It will create 2 keys. CD to .ssh and check the keys.

```

ubuntu@ip-172-31-43-190:~/.ssh$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/ubuntu/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/ubuntu/.ssh/id_rsa
Your public key has been saved in /home/ubuntu/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:plo6X2mnzPbdlwriW4uyytMQQYtNrtZEZSoHHI+qjdw ubuntu@ip-172-31-43-190
The key's randomart image is:
+---[RSA 3072]-----+
|  .oo+.o          |
|  .X.+           |
|  + O.           |
|  . *.           |
|  . o ..S        |
| .+. . .o .      |
|o..E oo+ o o .   |
|  .,=++ = + o|   |
|  oo+++=+.o oo |
+-----[SHA256]-----+
ubuntu@ip-172-31-43-190:~/.ssh$ ls
authorized_keys  id_rsa  id_rsa.pub

```

We need to add the Public key to Github so that we can create the bridge.


Go to Github -> setting -> SSH and GPG key -> Add new key -> do cat id_rsa_pub in the EC2instance and copy the key -> give a name to the key -> paste the copied key.

Create a item in jenkins -> Add the URL of the repo

Enter an item name

nodejsapp

» Required field


Freestyle project

This is the central feature of Jenkins. Jenkins will build your project, combining for something other than software build.

☐ Discard old builds

☒ GitHub project

Project url

https://github.com/rajani103/node-todo-cicd.git

Source Code Management

☐ None

☒ Git ?

Repositories ?

Repository URL ?

`https://github.com/rajani103/node-todo-cicd.git`

Now we will add the credentials so that Jenkins can access the code from the Github.

In Source Code Management

Go to Add credentials

Jenkins Credentials Provider: Jenkins

Add Credentials

Domain

Global credentials (unrestricted)

Kind

SSH Username with private key

Scope ?

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

ID ?

project

Description ?

github jenkins integration

Username

ubuntu

☐ Treat username as secret ?

Private Key

☒ Enter directly

Key

```
xDAhcFUGRf1+mSpH+KuStKE/+tjy/qK73P/t7oJhw3zFJCRT4dnWzk/dp+LVBoAkwTvRzV
ZuVAVRuRaRjnknAAAAF3VidW50dUBpcC0xNzItMzEtNDMtMTkwAQID
-----END OPENSSH PRIVATE KEY-----
```

Passphrase

Add Cancel

Click on Add -> save it

Now we are all ready to Build the job now. Click on Build now.

It will get build.

Dashboard > nodejs > #1 > Console Output

Status

</> Changes


Console Output

View as plain text

Edit Build Information

Delete build '#1'

Git Build Data

 **Console Output**

```
Started by user raj
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/nodejs
The recommended git tool is: NONE
using credential github
Cloning the remote Git repository
Cloning repository https://github.com/rajani103/node-todo-cicd.git
> git init /var/lib/jenkins/workspace/nodejs # timeout=10
Fetching upstream changes from https://github.com/rajani103/node-todo-cicd.git
```

Go to instance and check if the repo is cloned there.

```
ubuntu@ip-172-31-43-190:/var/lib/jenkins/workspace/nodejs$ ls
Dockerfile README.md app.js package-lock.json package.json  views
ubuntu@ip-172-31-43-190:/var/lib/jenkins/workspace/nodejs$
```

Now check the readme and install the necessary dependencies.

README.md

node-todo-cicd

```
sudo apt install nodejs sudo apt install npm
```

```
npm install
```

```
node app.js
```

After executing all these commands. check if you can access the URL . It is not accessible as we have not given access to the port 8000.

```
To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
root@ip-172-31-43-190:/var/lib/jenkins/workspace/nodejs# node app.js
Todolist running on http://0.0.0.0:8000
```

Go to instance -> Security -> Edit inbound rules -> Add rule

-

Custom TCP ▼

TCP

8000

Anywhere... ▼

0.0.0.0/0 X

0.0.0.0/0 X

Delete

Save it.

Now take public ip of the instance and port 8000 and you can access the application.

TrainWithShubham Community is Super Awesome

What should I do?

Now we will dockerize the application so that it can be accessed anywhere by anyone.

Go to instance -> install docker

```
#sudo install docker.io (#apt-get install docker)(apt install docker.io)
```

Create your Dockerfile

```
FROM node:12.2.0-alpine
WORKDIR app
COPY . .
RUN npm install
EXPOSE 8000
CMD ["node","app.js"]
```

Now build the image using this Dockerfile.

```
#docker build . -t todoapp
```

```
Removing intermediate container 9f050ee5c5f7
---> fc9c9152abb1
Step 5/6 : EXPOSE 8000
---> Running in 68ce9ab12ed1
Removing intermediate container 68ce9ab12ed1
---> 5d79f77289ee
Step 6/6 : CMD ["node","app.js"]
---> Running in dcae66873979
Removing intermediate container dcae66873979
---> 50b418f876d5
Successfully built 50b418f876d5
Successfully tagged todo:latest
```


Image (todo:latest) has been created successfully.

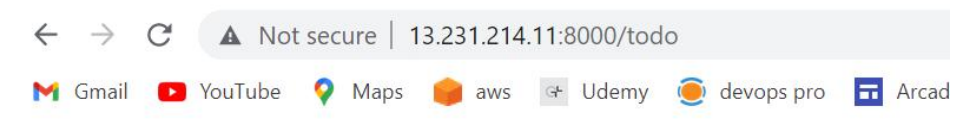
Now we will create container from this image.

```
#docker run -d --name todoappnode -p 8000:8000 todo:latest
```

Container will be created after this.

```
root@ip-172-31-43-190:/var/lib/jenkins/workspace/nodejs# docker ps
CONTAINER ID   IMAGE          COMMAND                  CREATED        STATUS        PORTS
58d27de56d99   50b418f876d5   "node app.js"           3 minutes ago  Up 3 minutes  0.0.0.0:8000->8000/tcp, :::8000->8000/tcp
todoappnode
```

Try accessing the same using the public IP and port 8000.



TrainWithShubham Community is Su

What should I do?

It is accessible.

Now we will automate this process by adding the commands in the shell.

Build Steps

≡ Execute shell ?

Command

See [the list of available environment variables](#)

```
docker build . -t nodeapp
docker run -d --name nodetodoapp -p 8000:8000 nodeapp
```

Build is successful. We can access the same on browser.

The screenshot shows the Jenkins web interface. The breadcrumb navigation at the top reads 'Dashboard > nodejs > #3 > Console Output'. On the left sidebar, the 'Console Output' tab is selected, with other options like 'Status', 'Changes', 'View as plain text', 'Edit Build Information', 'Delete build '#3'', 'Git Build Data', and 'Previous Build'. The main area displays the 'Console Output' with a green checkmark icon. The output text shows the build process: 'Started by user raj', 'Running as SYSTEM', 'Building in workspace /var/lib/jenkins/workspace/nodejs', 'The recommended git tool is: NONE', 'using credential github', and a series of git commands including 'git rev-parse', 'git config remote.origin.url', and 'git --version', all of which executed successfully.

Now all the processes that we were doing manually will be performed automatically as we did the automation.

Now we will configure web-hook so that every time there is any updation, deletion on the repository the job should be automatically triggered and it should perform the upcoming processes.

Kill the existing container first.

Jenkins -> Manage Jenkins -> Manage Plugins -> Install github integration plugin

Go to repository settings -> webhook -> Add webhook -> Payload URL add jenkins URL here -> (<http://13.231.214.11:8080/github-webhook/>)

Content type -> application.json

Add webhook.

Webhooks

Webhooks allow external services to be notified when certain events happen. When the specific event happens, Jenkins sends a POST request to each of the URLs you provide. Learn more in our [Webhooks Guide](#).

✓ <http://13.231.214.11:8080/github-w...> (push)

It has been added successfully.

Go to jenkins -> configure -> tick this

- ☐ GitHub Pull Requests ?
- ☒ GitHub hook trigger for GITScm polling ?
- ☐ Poll SCM ?

Now just update something in the repo and your job will be triggered automatically.

✓ #6

Feb 10, 2023, 1:04 PM

✓ #5

↑

↑

↓

Console Output

```
Started by GitHub push by rajani103
Running as SYSTEM
Building in workspace /var/lib/jenkins/workspace/nodejs
The recommended git tool is: NONE
using credential github
> git rev-parse --resolve-git-dir /var/lib/jenkins/workspace/nodejs/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/rajani103/node-todo-cicd.git # timeout=10
Fetching upstream changes from https://github.com/rajani103/node-todo-cicd.git
> git --version # timeout=10
```

So the build is successful.

Resources used :

Video : <https://www.youtube.com/watch?v=nplH3BzKHPk&t=5689s>

Github : <https://github.com/rajani103/node-todo-cicd>

Notes : <https://docs.google.com/document/d/1qos4eUfY4vZojnZLSGw8D3A46Yy2r42uiZPyPxL17A/edit#>

Please, feel free to drop any questions in the comments below. I would be happy to answer them.

If this post was helpful, please do follow and click the clap

_Thank you for reading

_Rajani