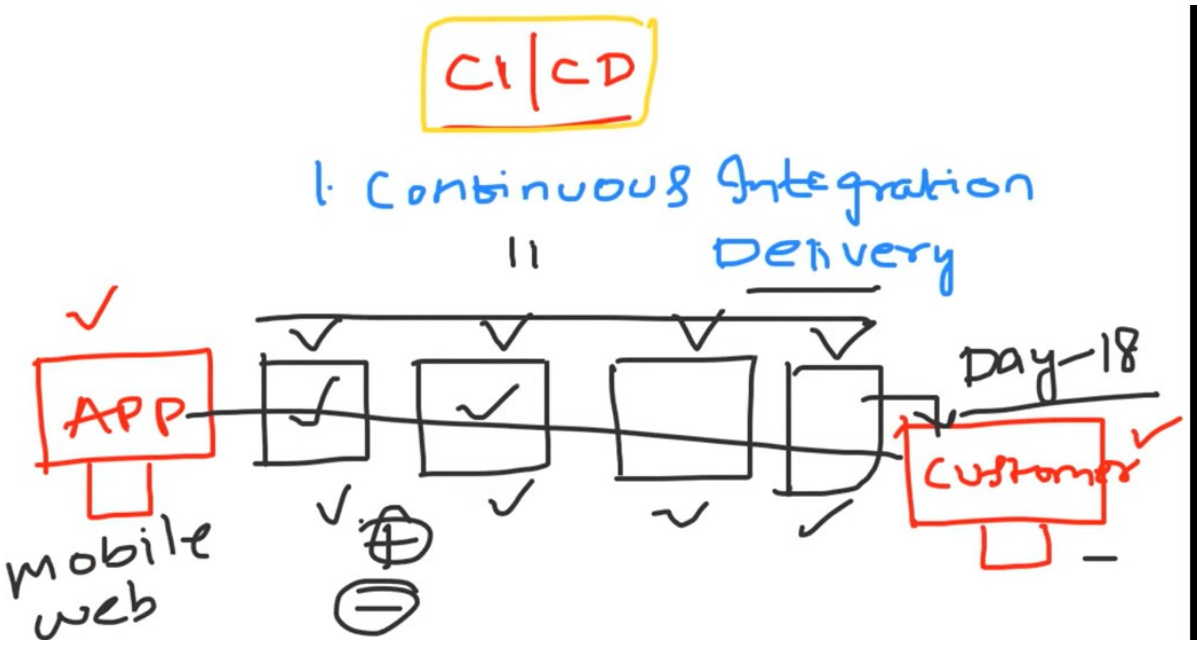
ULTIMATE CI/CD PIPELINE | JENKINS END TO END PROJECT ||



Unit testing

Static Code Analysis

Code Quality /vulnerability

Automation

Reports

Deployment

6:36. Unit testing is nothing but testing code for a block or specific functionality.

7:10. Static code analysis is to verify format, all indentation is right, syntax is right, make sure u didn't declare any unnecessary variables.

Code quality/ vulnerability testing is done to check for vulnerability, that can be misused by hackers

8:51. Automation testing/functional testing /endtoend testing, in this you verify your application in end to end manner such that one functionality doesn't affect other functionality.

Reports - reports for how many units test passed , what is code quality etc.

Deployment stage- deploying your application on a platform so that your customer can access your application.

Unit testing is a critical part of the continuous integration and continuous delivery (CI/CD) pipeline because it helps identify bugs and issues early in the development process:

Benefits

Unit testing can:

- Improve code quality
- Reduce development time
- Increase software reliability
- Save time and money

How it works

Unit tests are automated tests that validate the functionality of individual code components, such as functions, methods, or classes. They simulate all external dependencies that are used in the code base.

Best practices

To ensure effective unit testing, you can:

- Write testable code
- Test early and often
- Keep tests small and focused
- Ensure test coverage
- Use mocking
- Monitor and analyze test results

Other types of tests that can be used in CI/CD pipelines include:

- Integration tests
- Acceptance tests
- Synthetic tests
- Performance tests
- Resilience tests
- Static application security tests (SAST)
- Dynamic application security tests (DAST)

Static code analysis is a method of analyzing source code without executing the program to identify security flaws and programming mistakes. It's also known as Static Application Security Testing (SAST) or source code analysis.

Static code analysis is usually performed early in the development process to help developers: Improve code quality, Reduce the risk of security breaches, Comply with coding guidelines and industry standards, and Identify and fix issues early.

Static code analysis tools can identify potential vulnerabilities such as: NULL pointer dereferences, Buffer overflows, Memory usage errors, Injection, and Cross-site scripting.

Static code analysis tools use techniques such as data flow analysis and taint analysis.

Static code analysis is different from dynamic code analysis, which is the process of testing and assessing a program based on execution. Static code analysis is performed before running a program, while dynamic code analysis is performed after running a program.

Dynamic Application Security Testing (DAST) is a security testing method that can be integrated into a continuous integration and continuous delivery (CI/CD) pipeline to identify vulnerabilities in applications:

What is DAST?

DAST is a cybersecurity testing method that analyzes applications in real-time to identify vulnerabilities. It's also known as outside-in testing or black box testing.

How does DAST work?

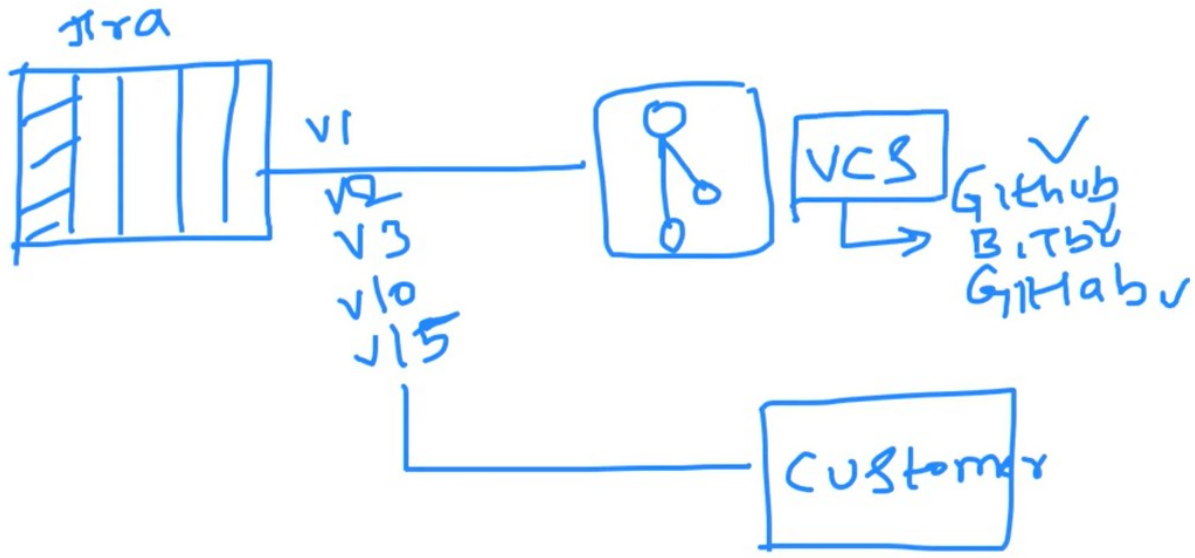
DAST simulates real-world attacks on an application to identify vulnerabilities. It does this by sending HTTP requests to the application, looking for anomalies, and reporting any vulnerabilities it finds.

Why use DAST in CI/CD?

DAST is an important tool for CI/CD pipelines because it helps to identify vulnerabilities early in the development lifecycle. This allows teams to address vulnerabilities before they become more costly and time-consuming to fix.

How to integrate DAST into CI/CD?

DAST scans can be automated in a CI/CD pipeline, or run independently on demand. A CI/CD pipeline is a series of automated steps that execute when changes are made to the codebase.

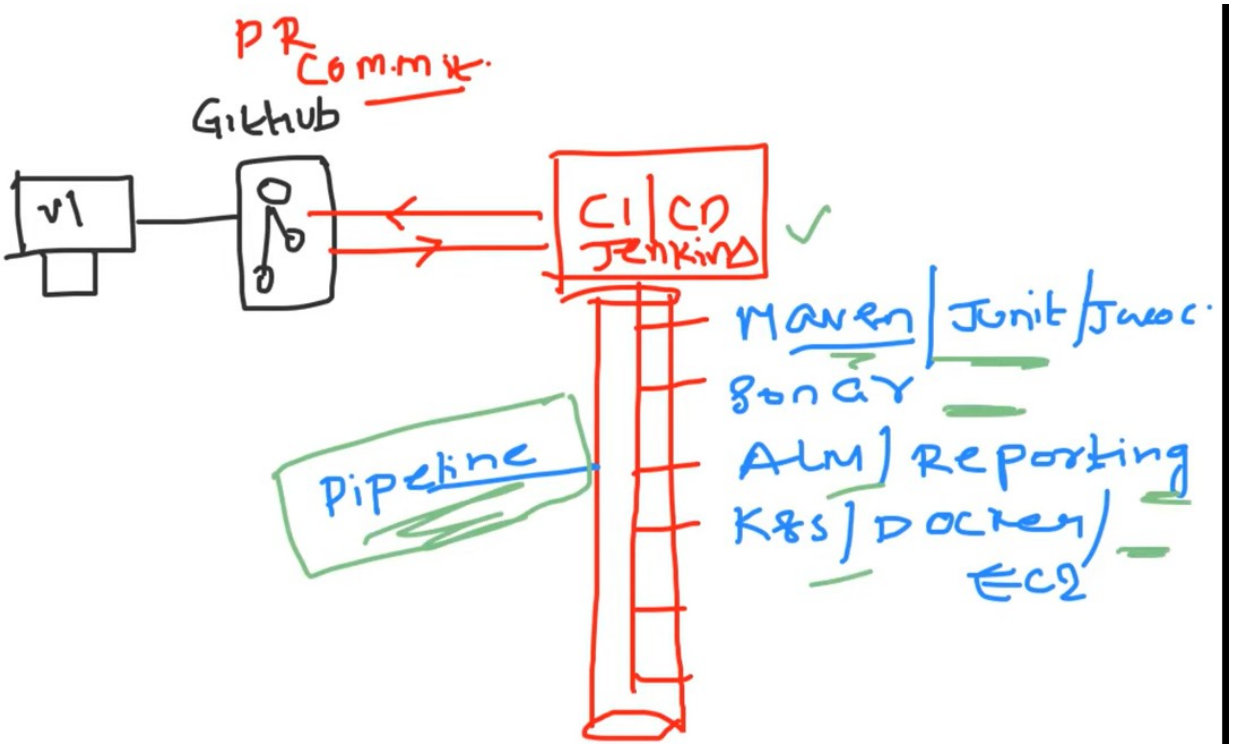


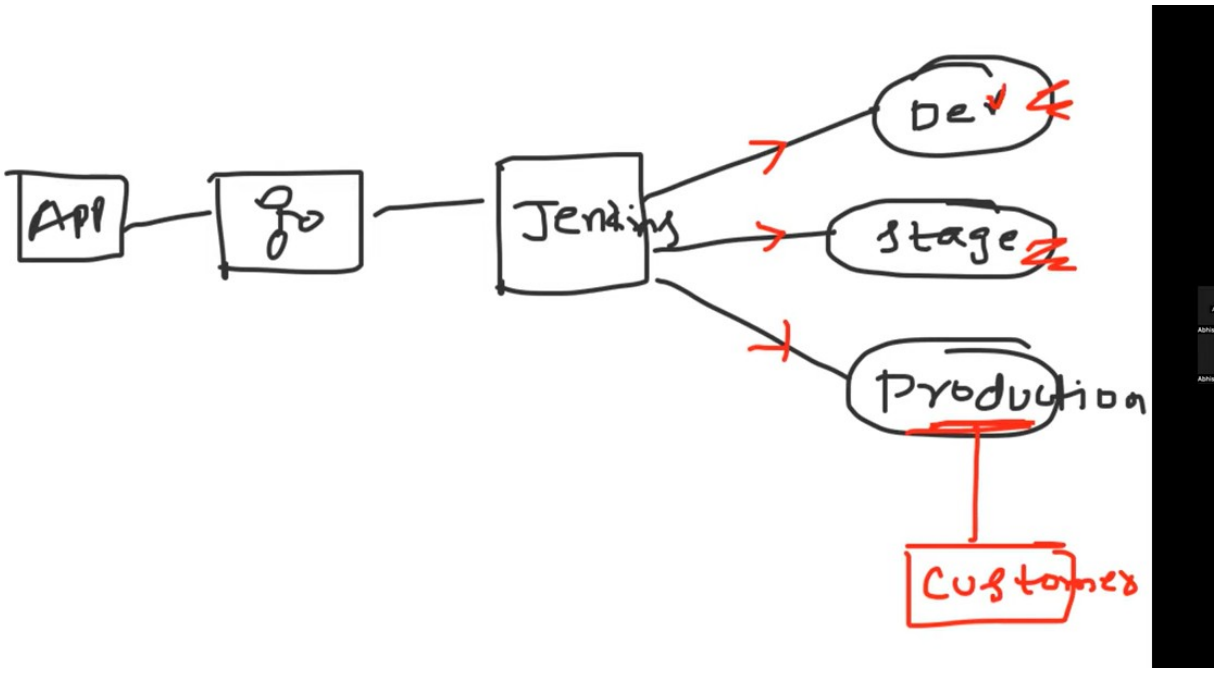
 # Define Jenkins

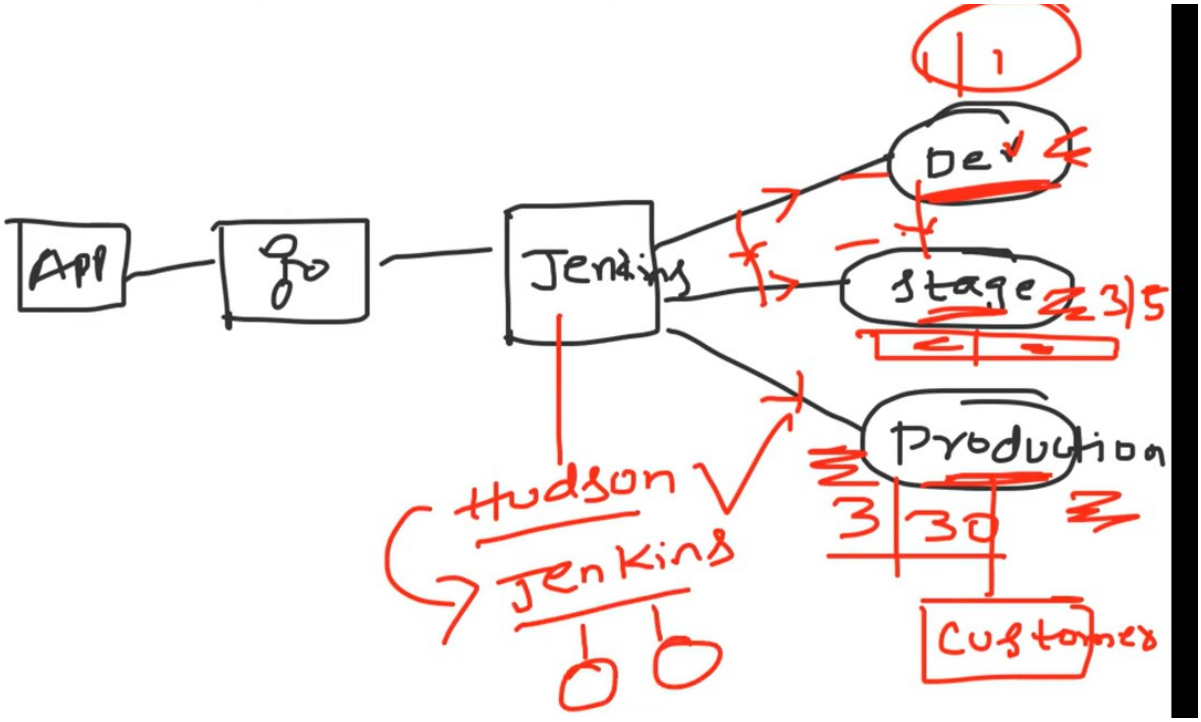
Jenkins is an open-source automation server that helps manage and control the software development process. It can be used to automate tasks like building, testing, and deploying software. Jenkins is a popular DevOps tool that can be run on Windows, Linux, macOS, and other Unix-like operating systems.

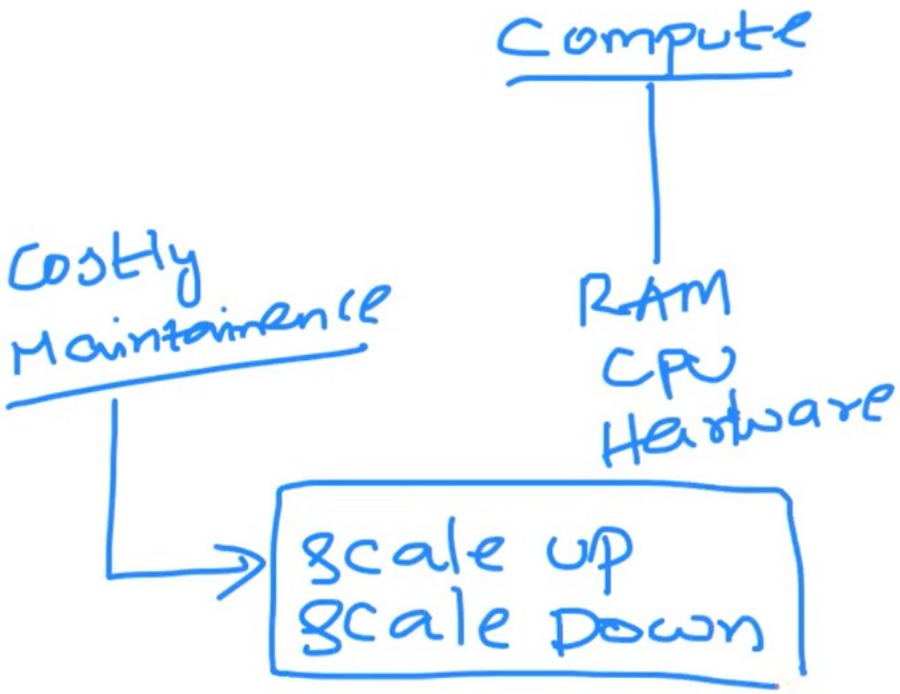
Here are some features of Jenkins:

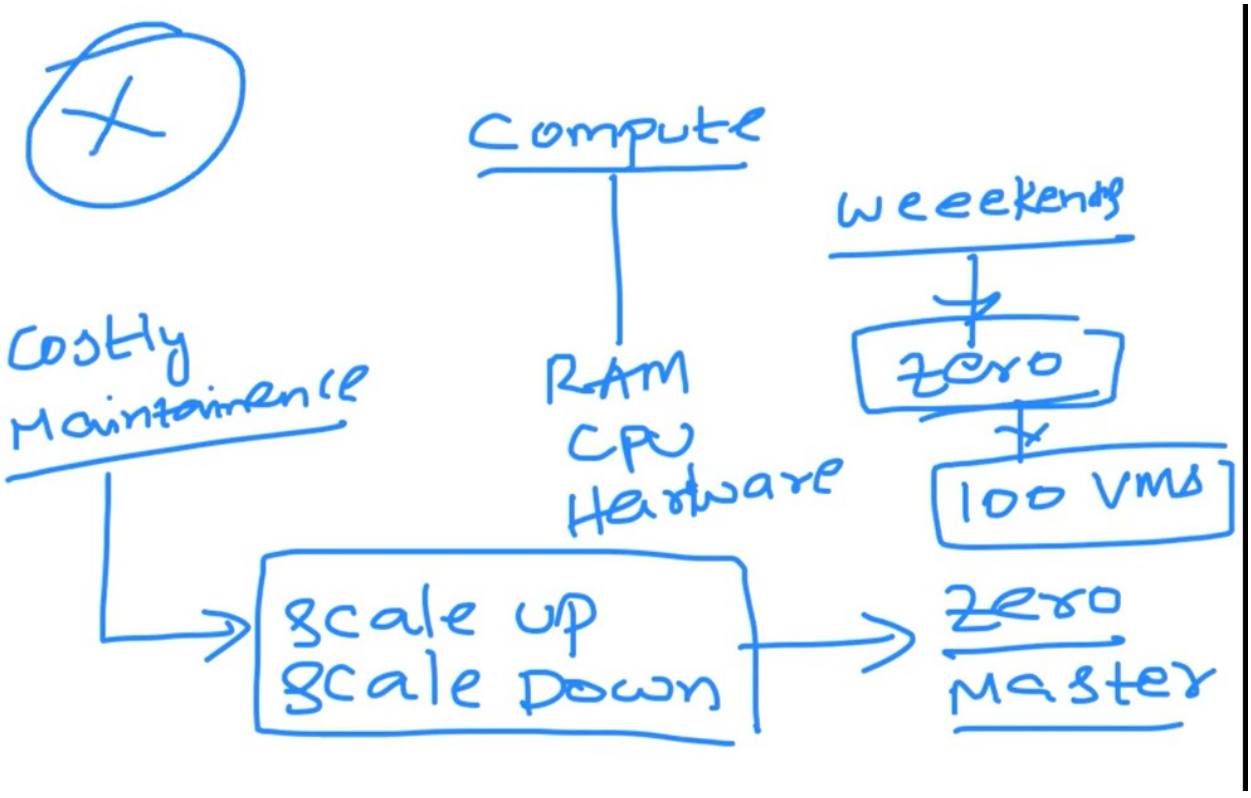
- **Easy installation**: Jenkins can be installed through native system packages, Docker, or run standalone.
- **Easy configuration**: Jenkins can be set up and configured via its web interface.
- **Plugins**: Jenkins provides hundreds of plugins for building, deploying, and automating projects.
- **CI/CD pipelines**: Jenkins can implement CI/CD workflows called pipelines to automate testing and reporting on code changes.
- **Docker support**: Jenkins can use Docker images and containers to run inside.
- **Test recording, reporting, and visualization**: Jenkins can record test failures for reporting and visualization in the web UI.



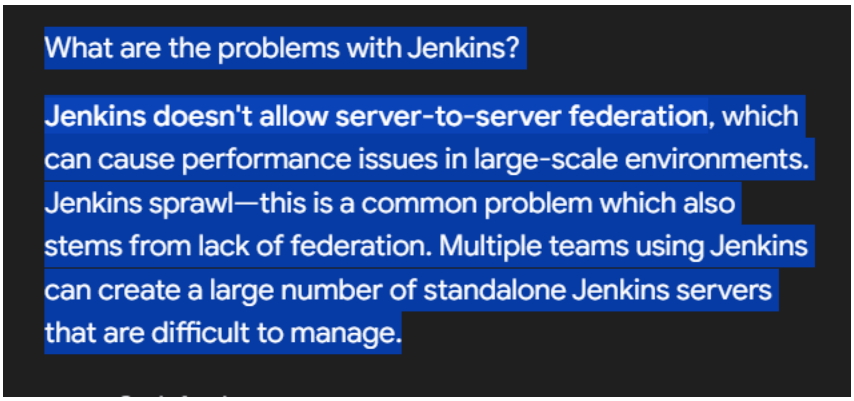

Dev
Stage
Production
Customers


1
Dev
Stage
3|5
Production
3|30
Hudson
Jenkins
Customers

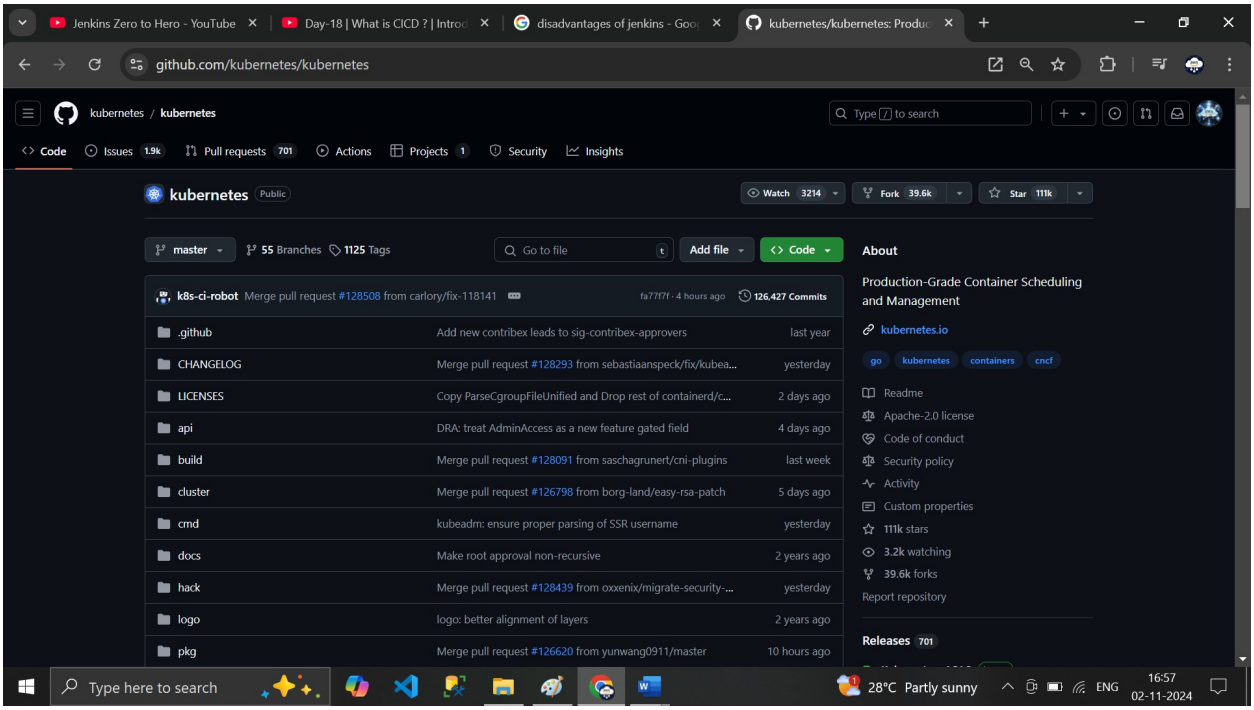




Disadvantage in Jenkins .. When I am in Weekends / holiday Server should not be Running .. Jenkins fails in this

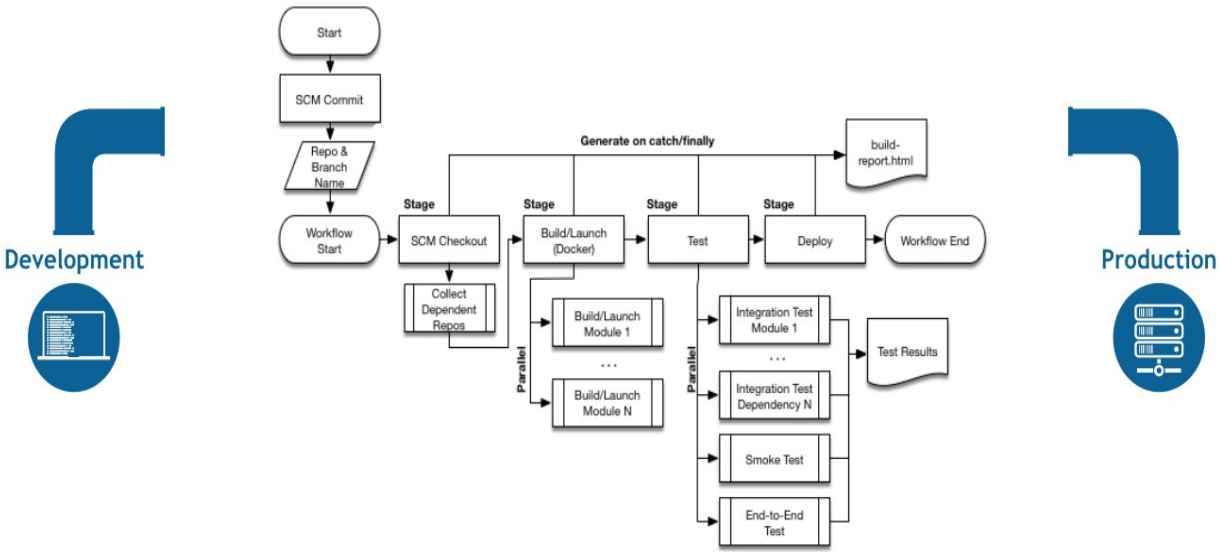


https://github.com/kubernetes/kubernetes



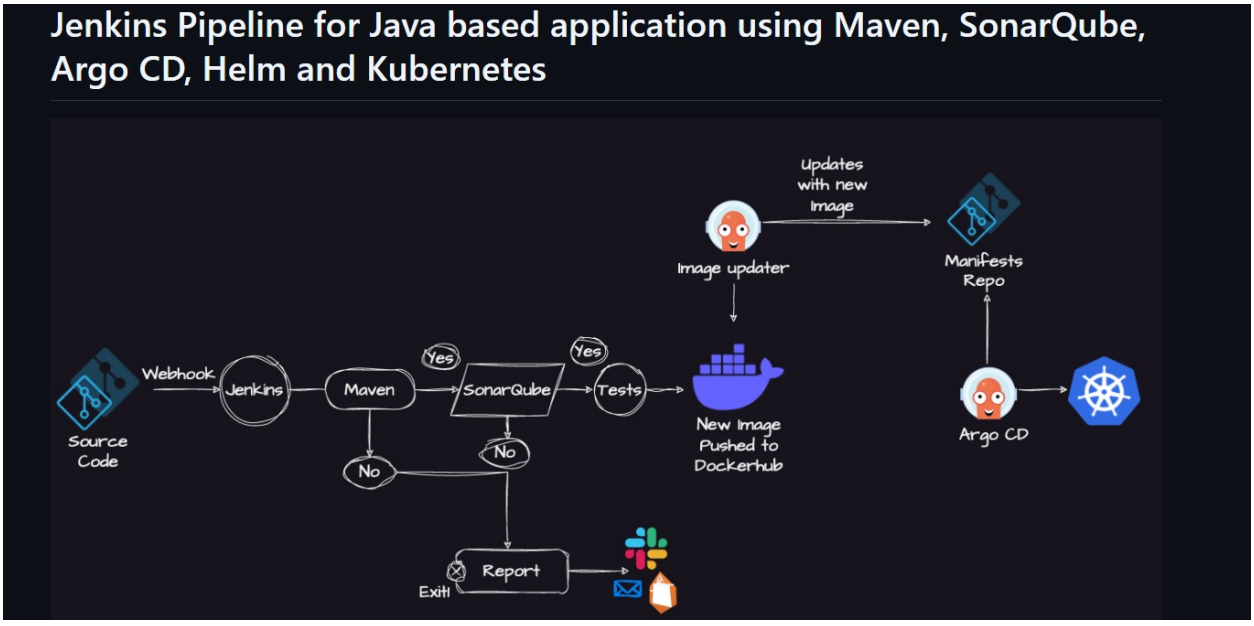
Day-19 | Jenkins ZERO to HERO | 3 Projects Live |Docker Agent |Interview

Jenkins Pipeline Project using Docker as Agent(NodeJS-APP) with AWS EC2- Deployment



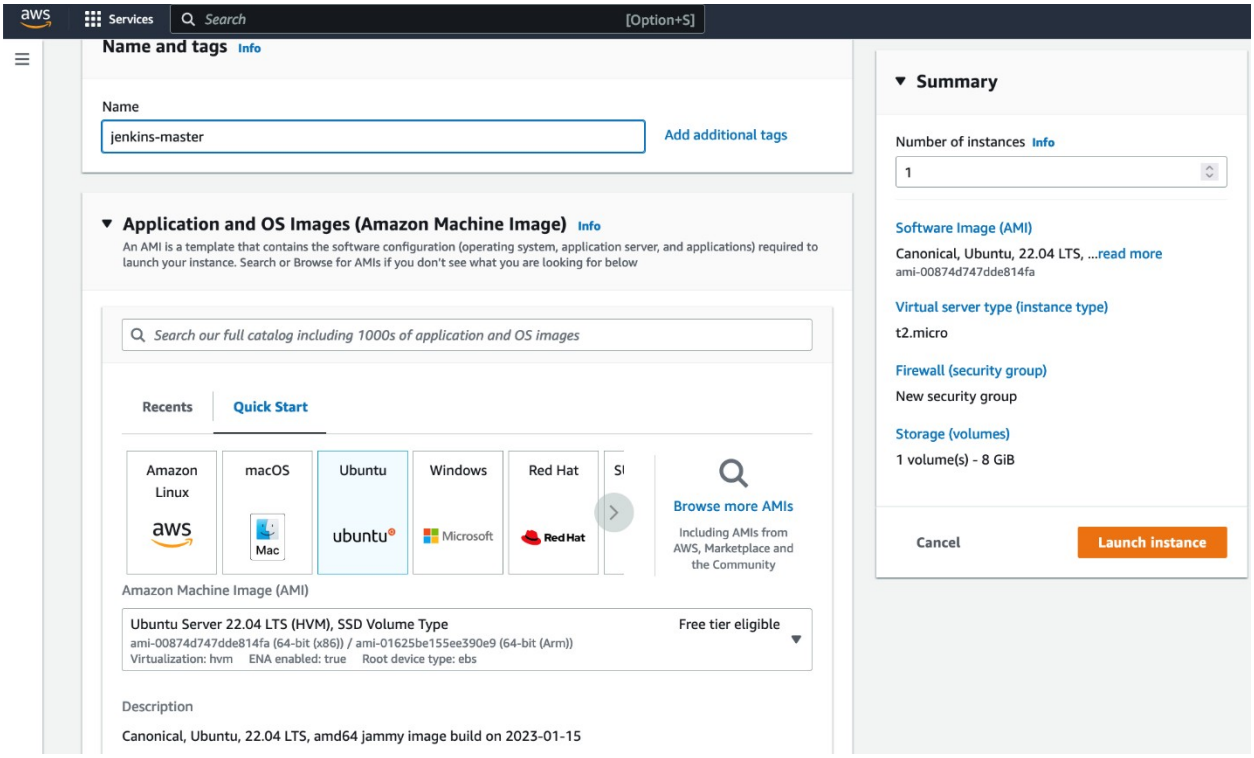
1. Install Jenkins on ec2
2. Configure Jenkins and expose to outside world
3. Use Docker as Agents against the VM approach
4. Advantages of using Docker as Agents
5. GitOps appraoch to deploy applications on to k8s

h



AWS EC2 Instance

- Go to AWS Console
- Instances(running)
- Launch instances



#install

Pre-Requisites:

- Java (JDK)

Run the below commands to install Java and Jenkins

Install Java

```
sudo apt update
sudo apt install openjdk-17-jre
```

Verify Java is Installed

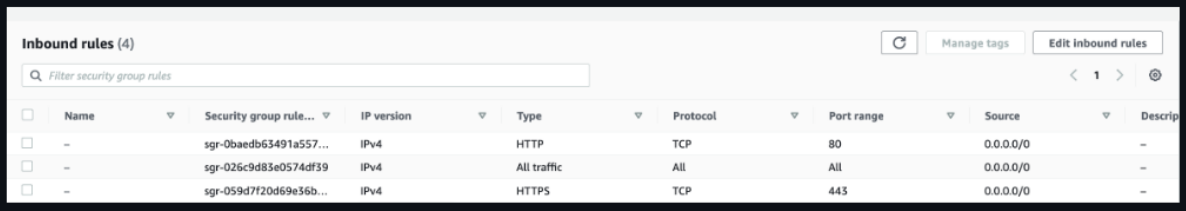
```
java -version
```

Now, you can proceed with installing Jenkins

```
curl -fsSL https://pkg.jenkins.io/debian/jenkins.io-2023.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
```

**Note: ** By default, Jenkins will not be accessible to the external world due to the inbound traffic restriction by AWS. Open port 8080 in the inbound traffic rules as show below.

- EC2 > Instances > Click on
- In the bottom tabs -> Click on Security
- Security groups
- Add inbound traffic rules as shown in the image (you can just allow TCP 8080 as well, in my case, I allowed `All traffic`).



Inbound rules (4)

Manage tags | Edit inbound rules

Filter security group rules

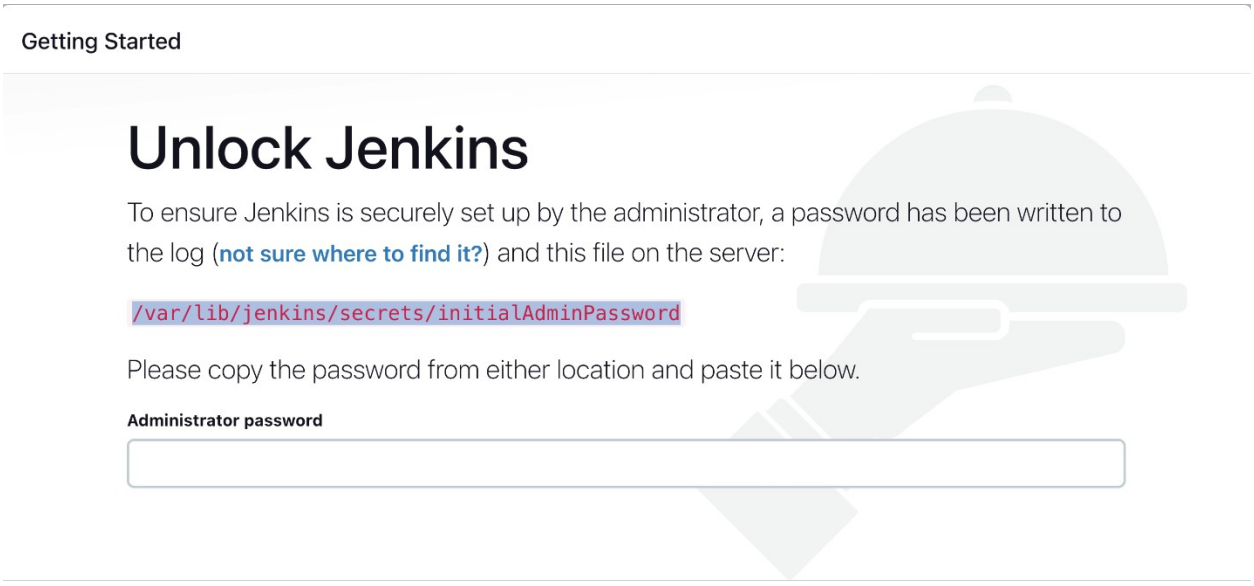
Name	Security group rule...	IP version	Type	Protocol	Port range	Source	Descrip
-	sgr-0baedb63491a557...	IPv4	HTTP	TCP	80	0.0.0.0/0	-
-	sgr-026c9d83e0574df39	IPv4	All traffic	All	All	0.0.0.0/0	-
-	sgr-059d7f20d69e36b...	IPv4	HTTPS	TCP	443	0.0.0.0/0	-

Login to Jenkins using the below URL:

http://:8080 [You can get the ec2-instance-public-ip-address from your AWS EC2 console page]

Note: If you are not interested in allowing `All Traffic` to your EC2 instance 1. Delete the inbound traffic rule for your instance 2. Edit the inbound traffic rule to only allow custom TCP port `8080`

After you login to Jenkins, - Run the command to copy the Jenkins Admin Password - `sudo cat /var/lib/jenkins/secrets/initialAdminPassword` - Enter the Administrator password



Getting Started

Unlock Jenkins

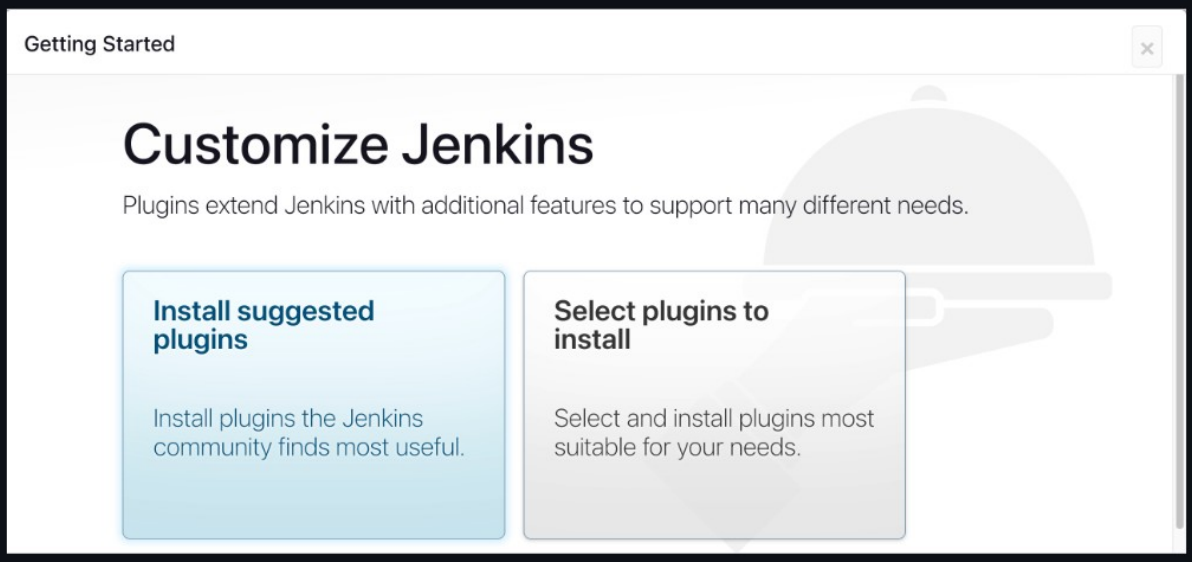
To ensure Jenkins is securely set up by the administrator, a password has been written to the log (**not sure where to find it?**) and this file on the server:

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

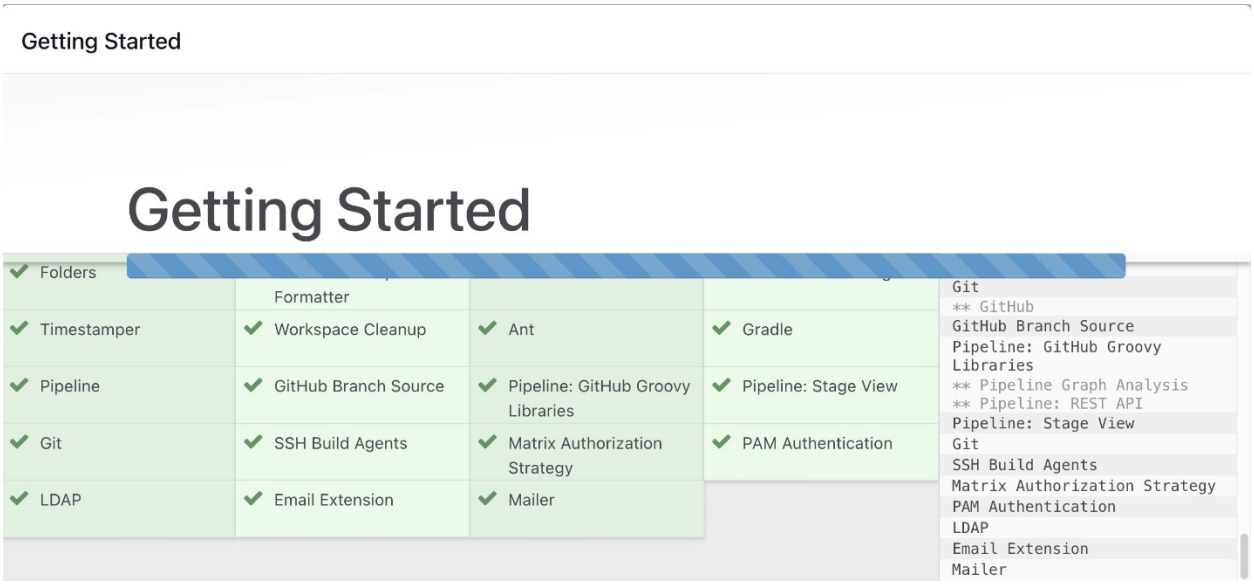
Please copy the password from either location and paste it below.

Administrator password

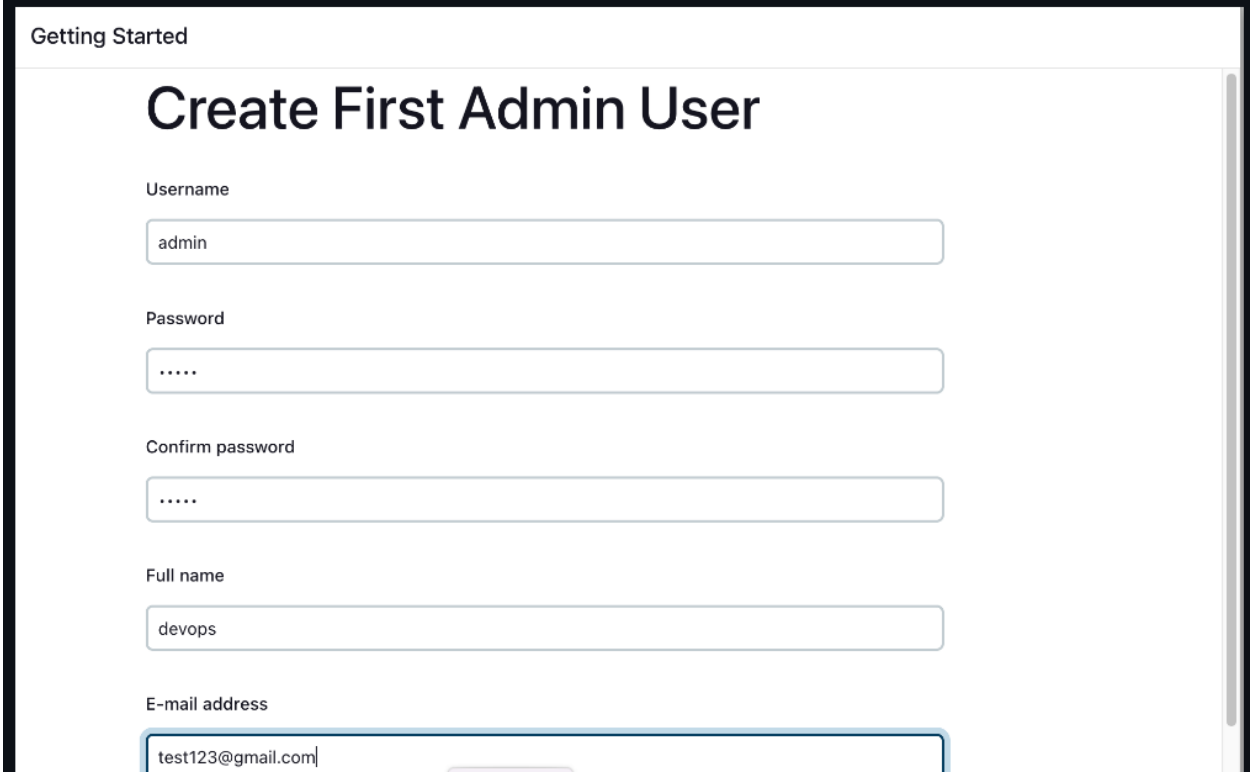
Click on Install suggested plugins



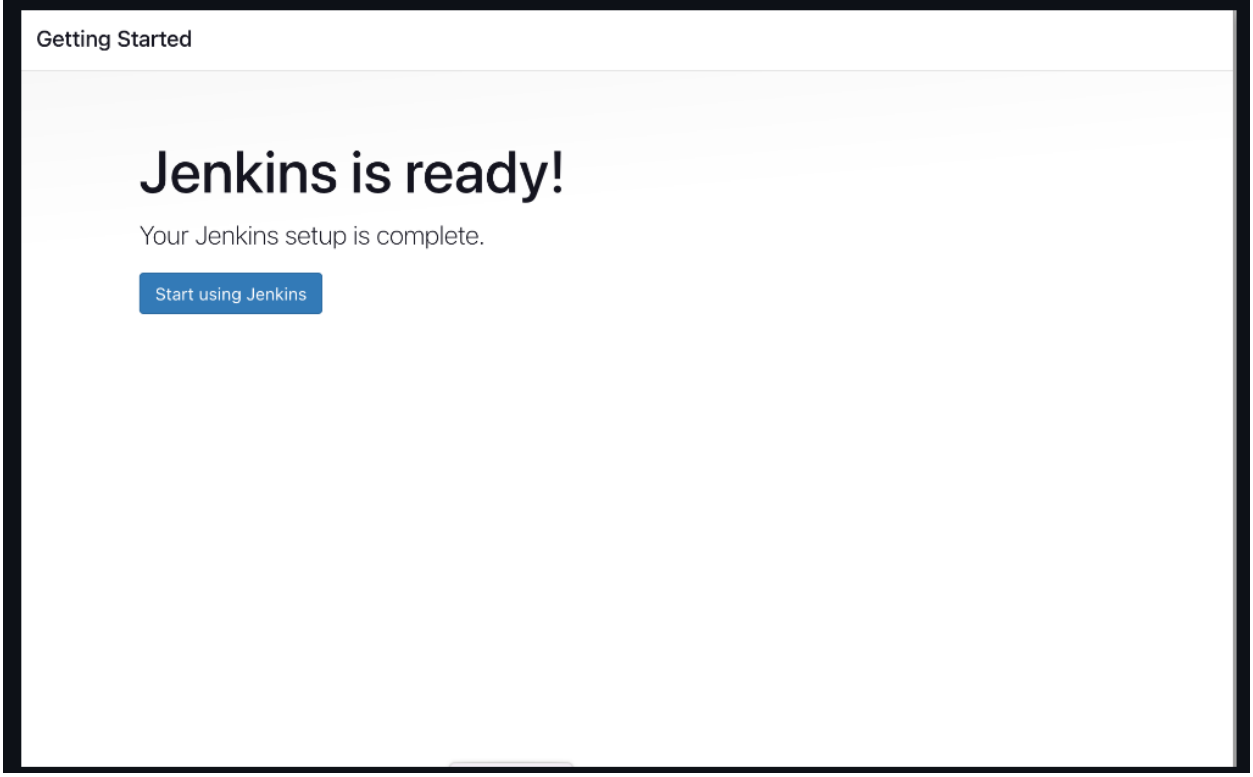
Wait for the Jenkins to Install suggested plugins



Create First Admin User or Skip the step [If you want to use this Jenkins instance for future use-cases as well, better to create admin user]

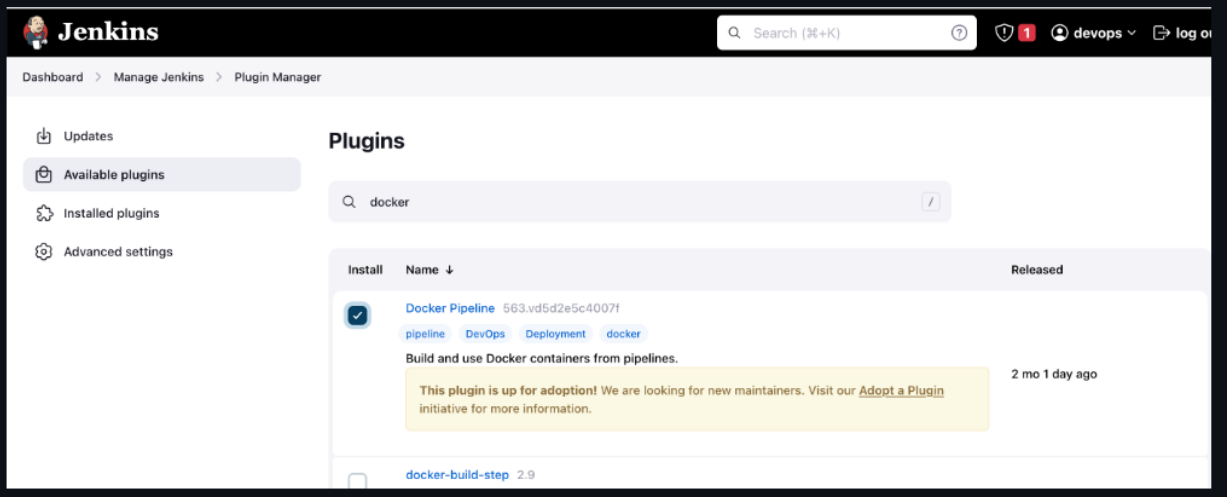


Jenkins Installation is Successful. You can now starting using the Jenkins



Install the Docker Pipeline plugin in Jenkins:

- Log in to Jenkins.
- Go to Manage Jenkins > Manage Plugins.
- In the Available tab, search for "Docker Pipeline".
- Select the plugin and click the Install button.
- Restart Jenkins after the plugin is installed.



Wait for the Jenkins to be restarted.

Docker Slave Configuration

Run the below command to Install Docker

```
sudo apt update
sudo apt install docker.io
```

Grant Jenkins user and Ubuntu user permission to docker deamon.

```
sudo su -
usermod -aG docker jenkins
usermod -aG docker ubuntu
systemctl restart docker
```

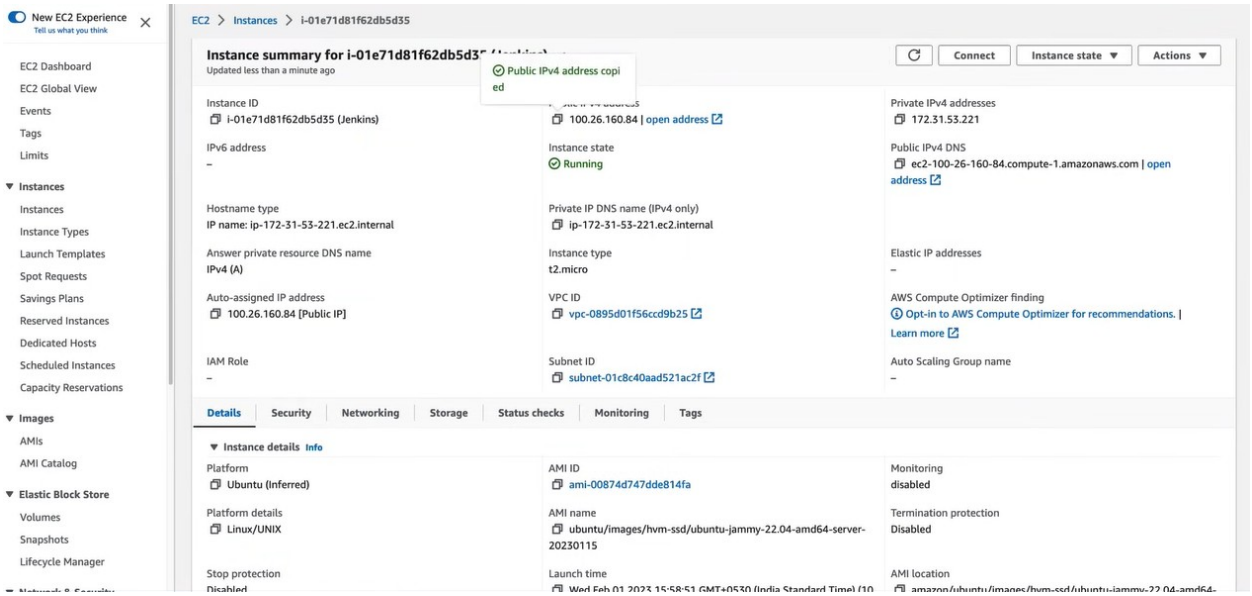
Once you are done with the above steps, it is better to restart Jenkins.

```
http://<ec2-instance-public-ip>:8080/restart
```

The docker agent configuration is now successful.

---------- Practical part

Cp the Public IP




```
➜  Jenkins-Zero-To-Hero git:(main) ssh -i /Users/aveerama/Downloads/abhishek_devops.pem ubuntu@100.26.160.84
The authenticity of host '100.26.160.84 (100.26.160.84)' can't be established.
ED25519 key fingerprint is SHA256:Vfe7tiQRhSvK3FkwM+HlS+UA8bQi7i+zzskqZaeK3Bk.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '100.26.160.84' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1028-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Wed Feb  1 10:39:11 UTC 2023

  System load:  0.0               Processes:             96
```

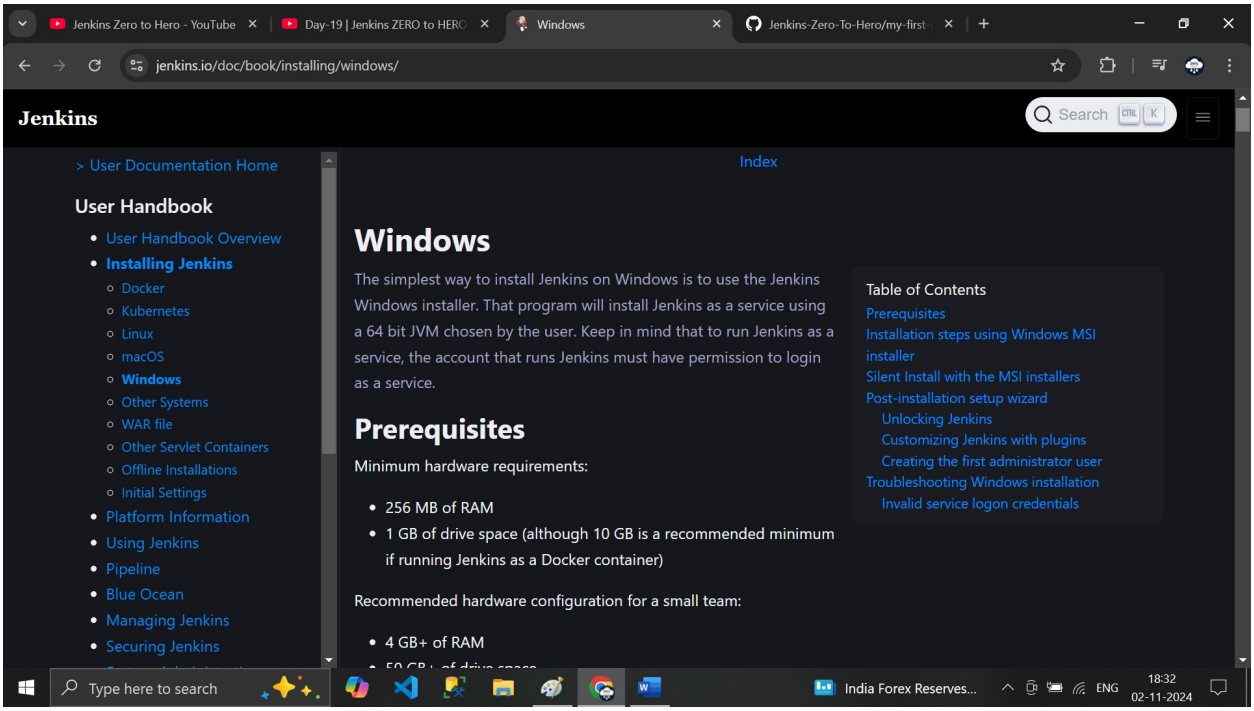
updating App packages

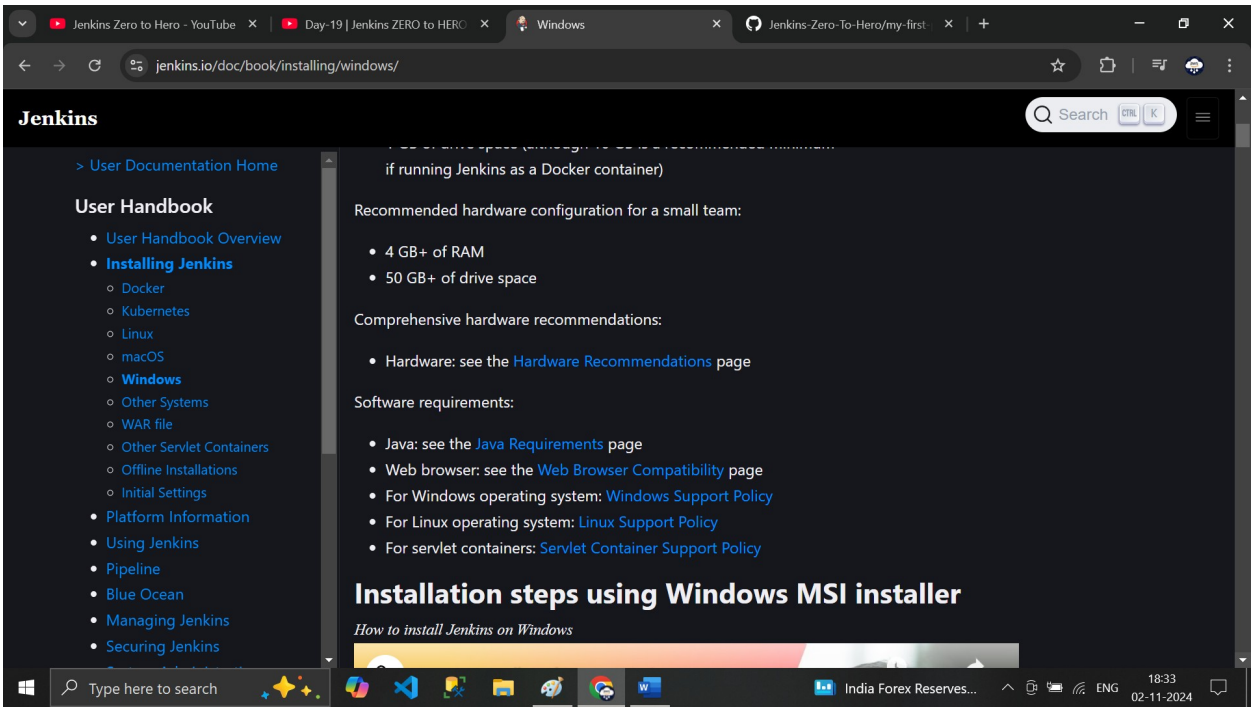
```
ubuntu@ip-172-31-53-221:~$ sudo apt update
sudo apt install openjdk-11-jre
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:4 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:5 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:6 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe Translation-en [5652 kB]
Get:7 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/universe amd64 c-n-f Metadata [286 kB]
Get:8 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 Packages [217 kB]
Get:9 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse Translation-en [112 kB]
Get:10 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy/multiverse amd64 c-n-f Metadata [8372 B]
Get:11 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [852 kB]
Get:12 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [189 kB]
Get:13 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/main amd64 c-n-f Metadata [13.1 kB]
Get:14 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [566 kB]
Get:15 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [87.1 kB]
```

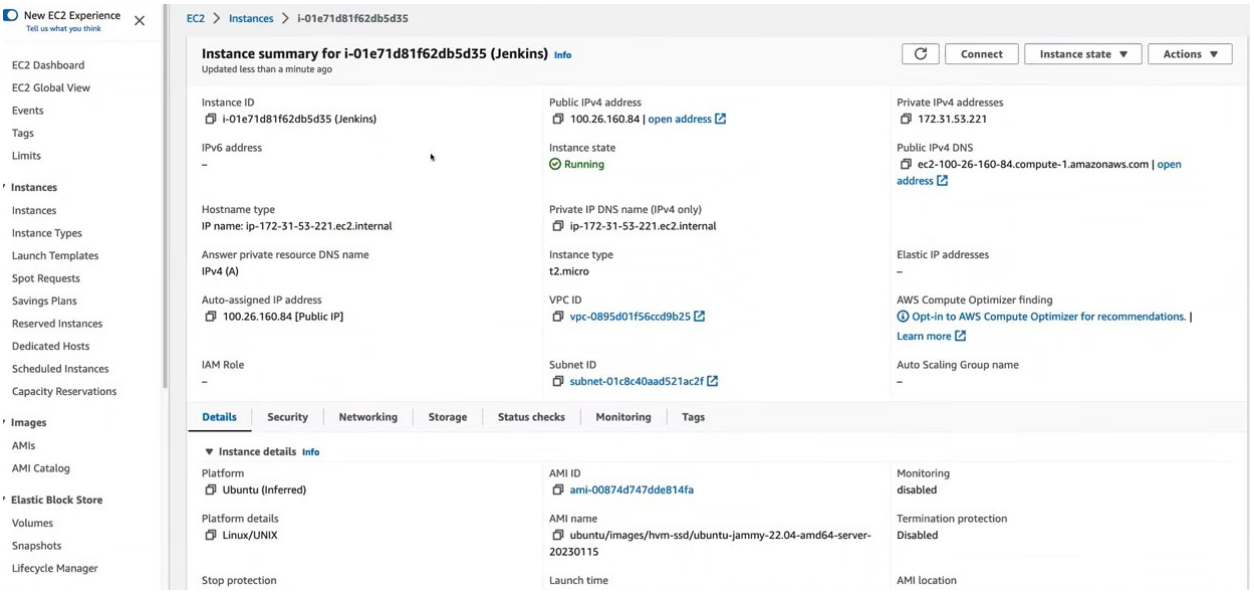
install Jenkins

```
ubuntu@ip-172-31-53-221:~$ curl -fsSL https://pkg.jenkins.io/debian-stable/jenkins.io.key | sudo tee \
  /usr/share/keyrings/jenkins-keyring.asc > /dev/null
echo deb [signed-by=/usr/share/keyrings/jenkins-keyring.asc] \
  https://pkg.jenkins.io/debian-stable binary/ | sudo tee \
  /etc/apt/sources.list.d/jenkins.list > /dev/null
sudo apt-get update
sudo apt-get install jenkins
Hit:1 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://us-east-1.ec2.archive.ubuntu.com/ubuntu jammy-backports InRelease
Ign:4 https://pkg.jenkins.io/debian-stable binary/ InRelease
Get:5 https://pkg.jenkins.io/debian-stable binary/ Release [2044 B]
Get:6 https://pkg.jenkins.io/debian-stable binary/ Release.gpg [833 B]
```

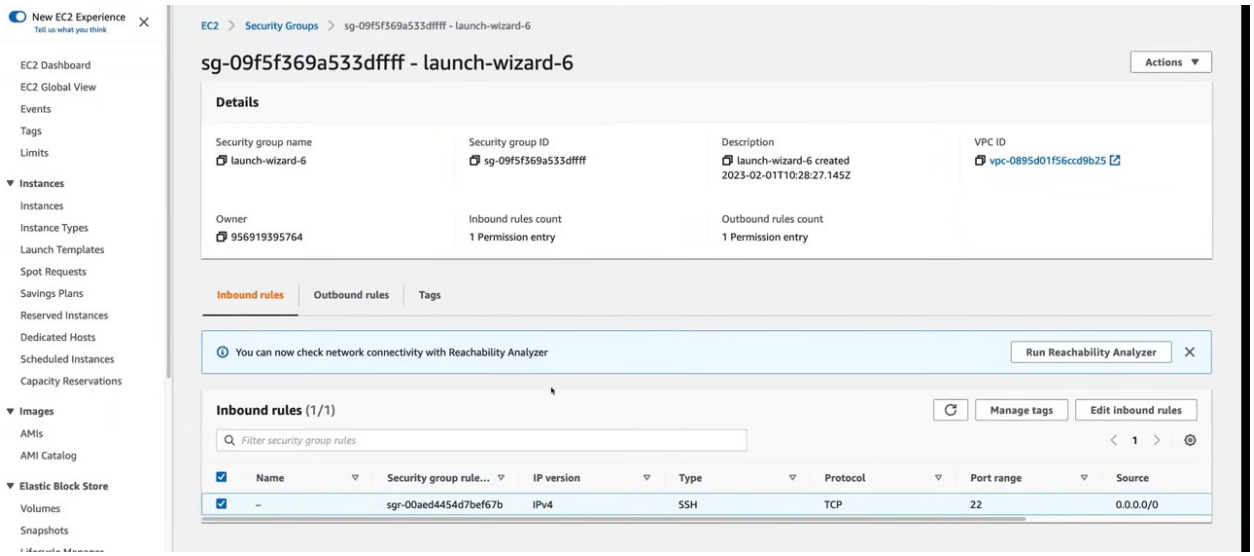
https://www.jenkins.io/doc/book/installing/

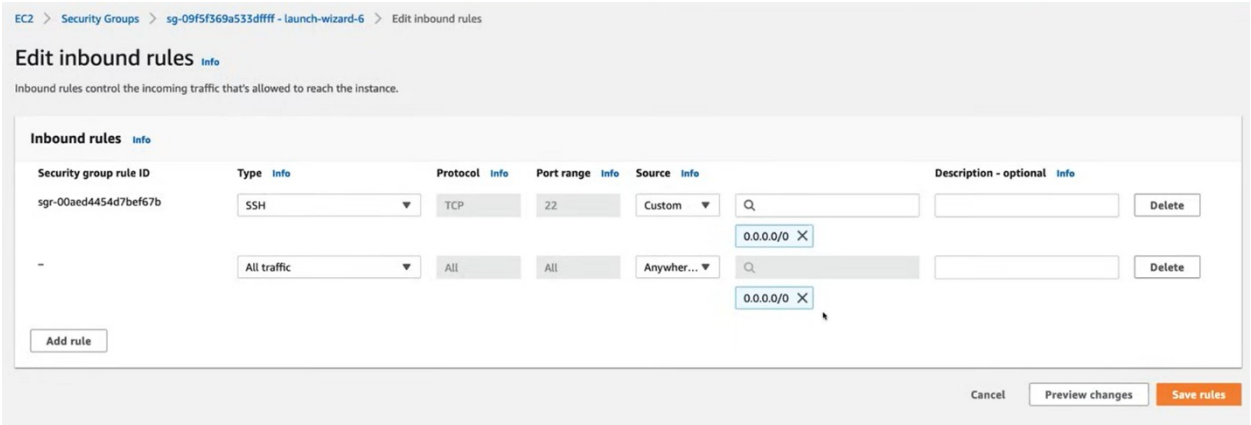




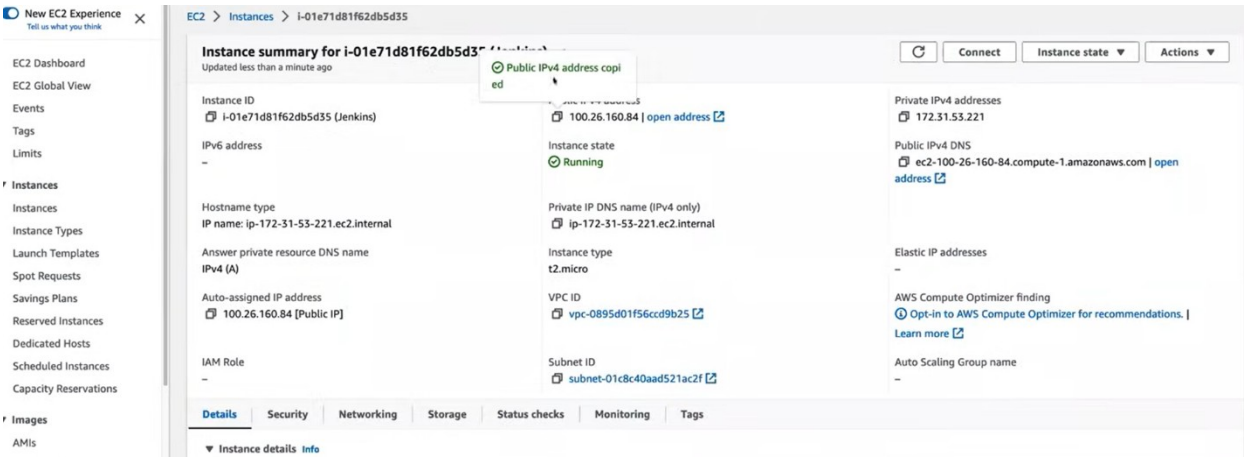


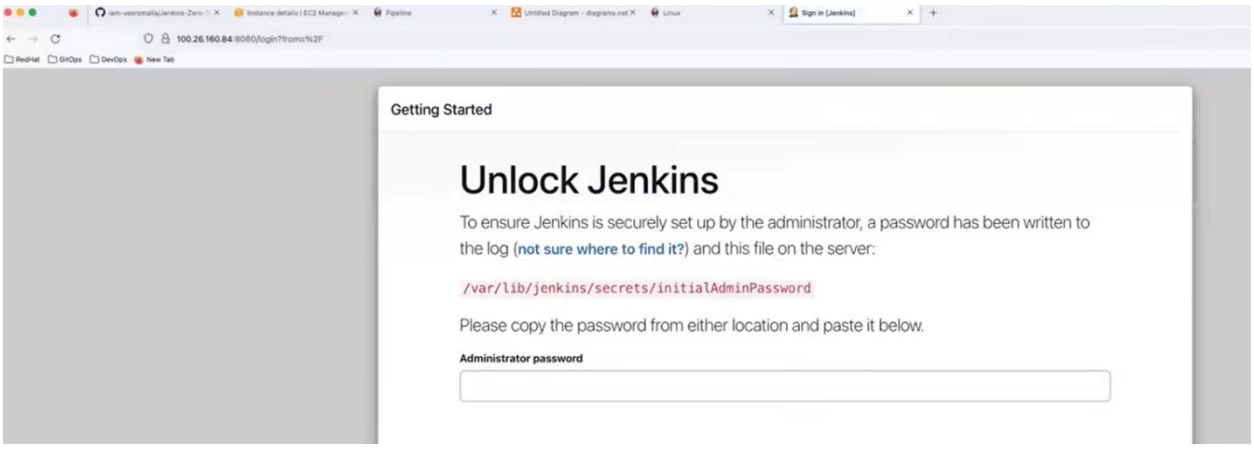
SG – Edit the inbound rule



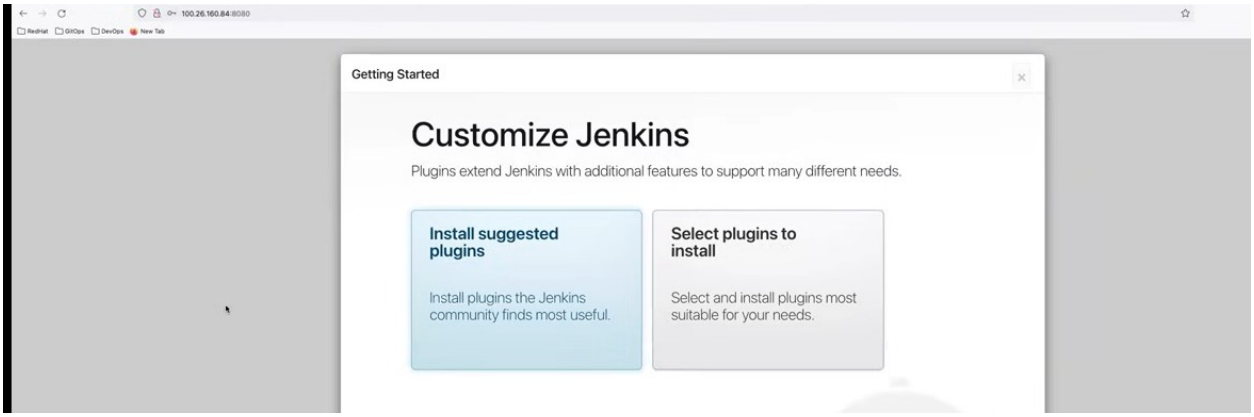


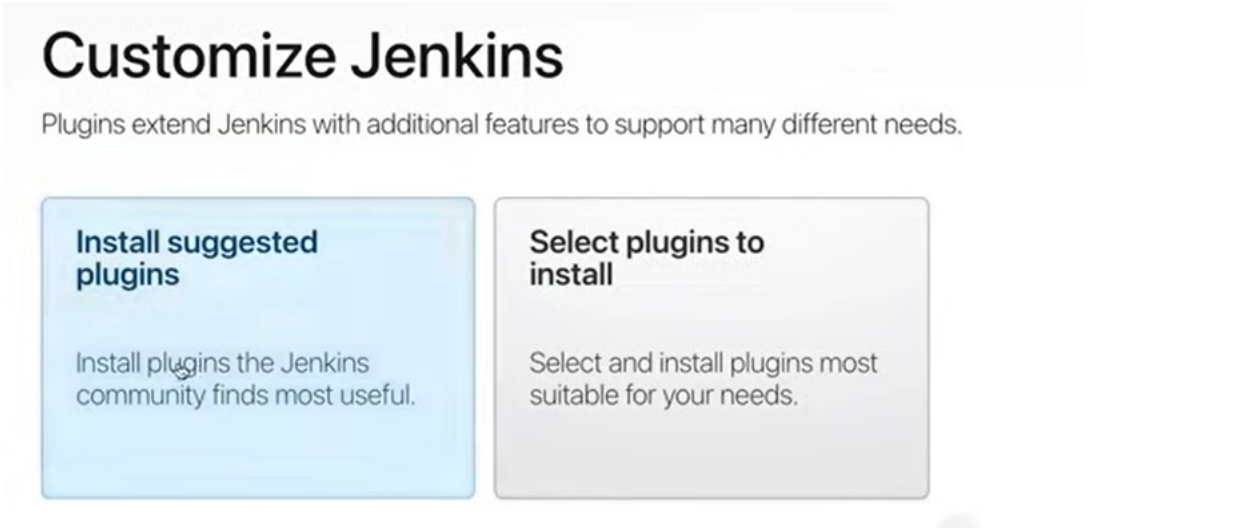
now Select Public IP address

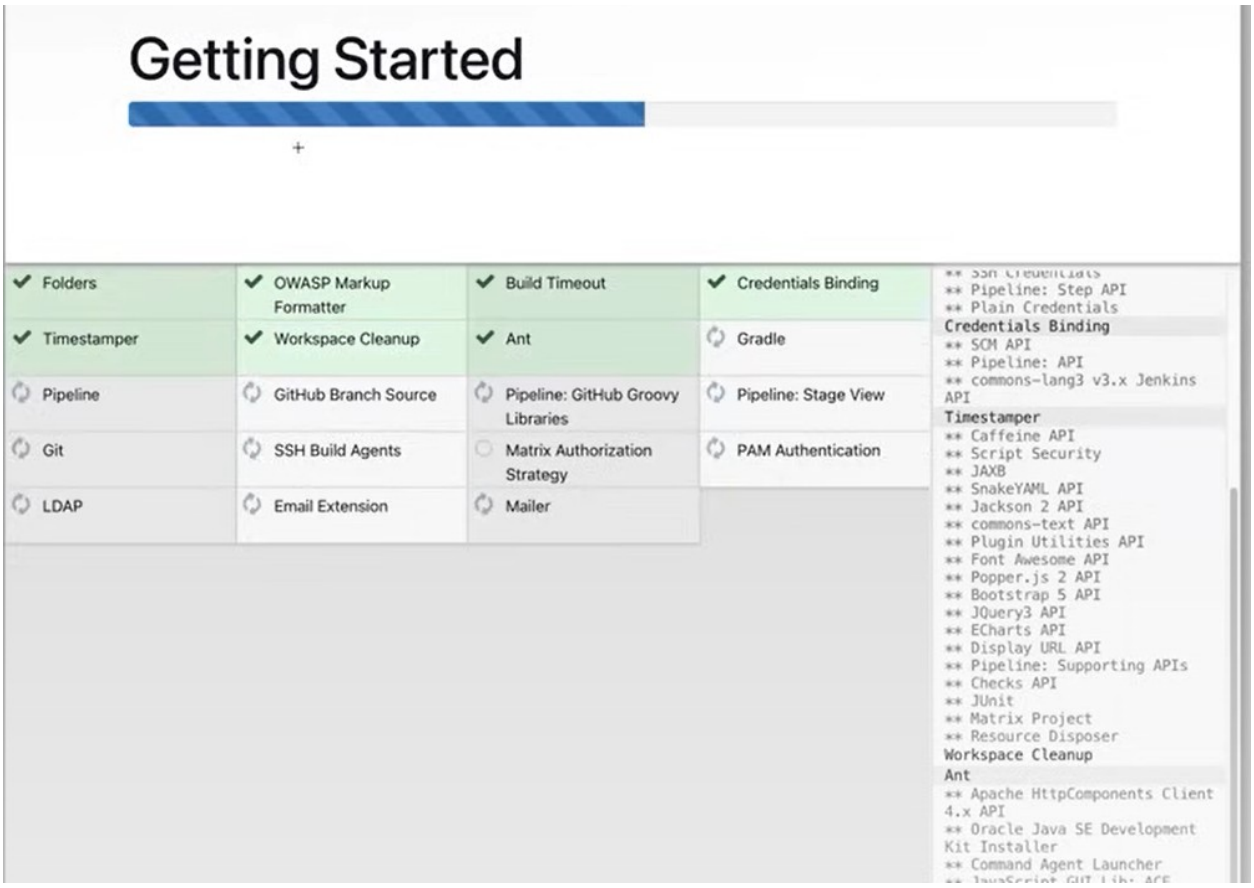





```
ubuntu@ip-172-31-53-221:~$ sudo cat /var/lib/jenkins/secrets/initialAdminPassword
c2f1e1f6e94648a99ec5ed91c0034886
ubuntu@ip-172-31-53-221:~$
```






Getting Started
Folders
OWASP Markup Formatter
Build Timeout
Credentials Binding
Timestamper
Workspace Cleanup
Ant
Gradle
Pipeline
GitHub Branch Source
Pipeline: GitHub Groovy Libraries
Pipeline: Stage View
Git
SSH Build Agents
Matrix Authorization Strategy
PAM Authentication
LDAP
Email Extension
Mailer
** Pipeline: Step API
** Plain Credentials
Credentials Binding
** SCM API
** Pipeline: API
** commons-lang3 v3.x Jenkins API
Timestamper
** Caffeine API
** Script Security
** JAXB
** SnakeYAML API
** Jackson 2 API
** commons-text API
** Plugin Utilities API
** Font Awesome API
** Popper.js 2 API
** Bootstrap 5 API
** JQuery3 API
** ECharts API
** Display URL API
** Pipeline: Supporting APIs
** Checks API
** JUnit
** Matrix Project
** Resource Disposer
Workspace Cleanup
Ant
** Apache HttpComponents Client 4.x API
** Oracle Java SE Development Kit Installer
** Command Agent Launcher

Create First Admin User

Username

admin

Password

•••••

Confirm password

•••••

Full name

DevOps

E-mail address

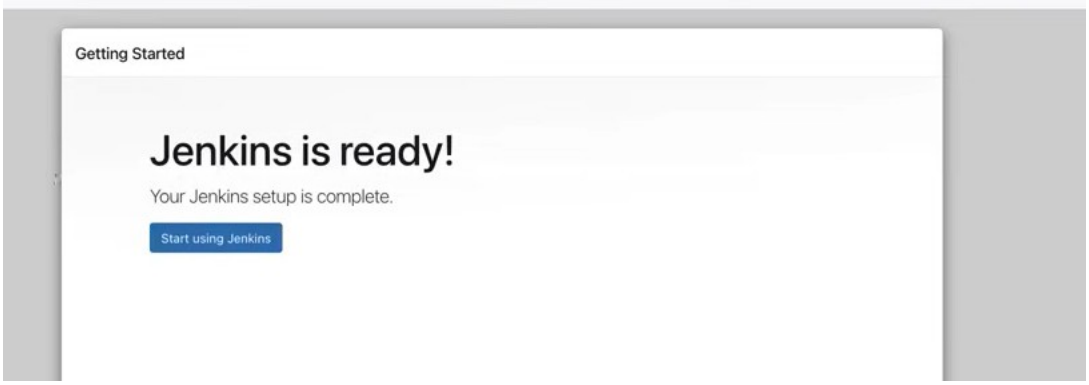
test123@gmail.com

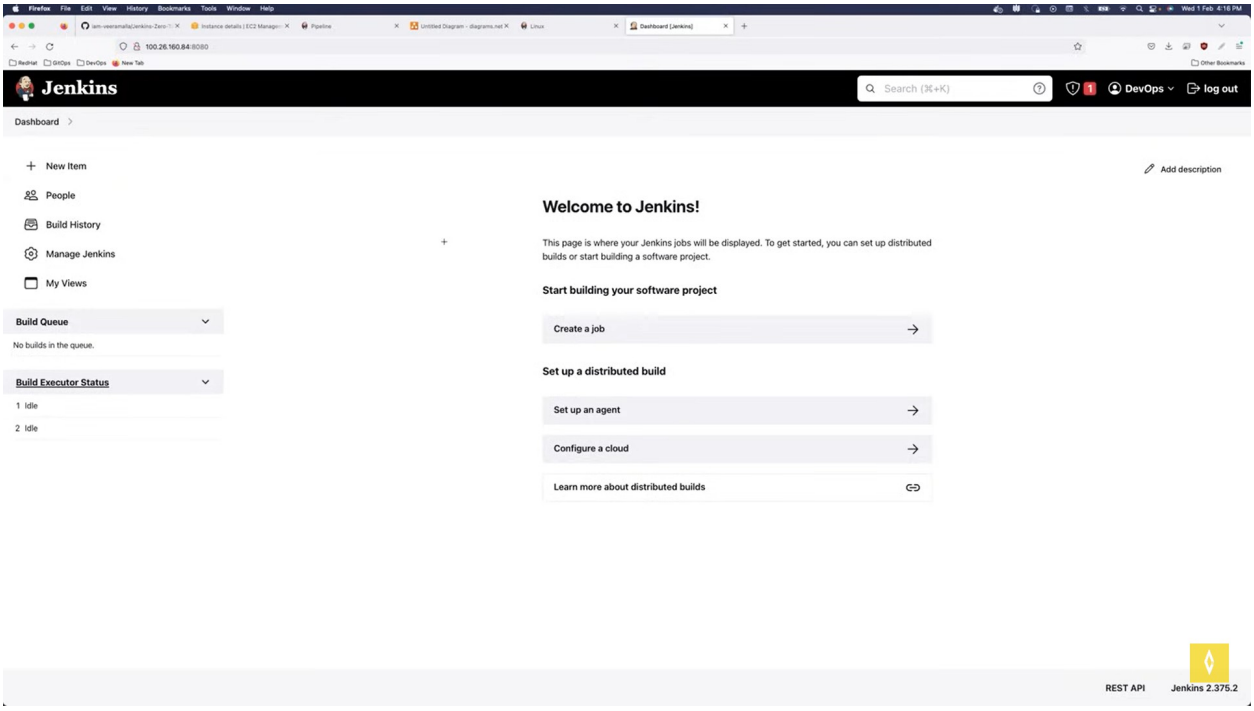
Instance Configuration

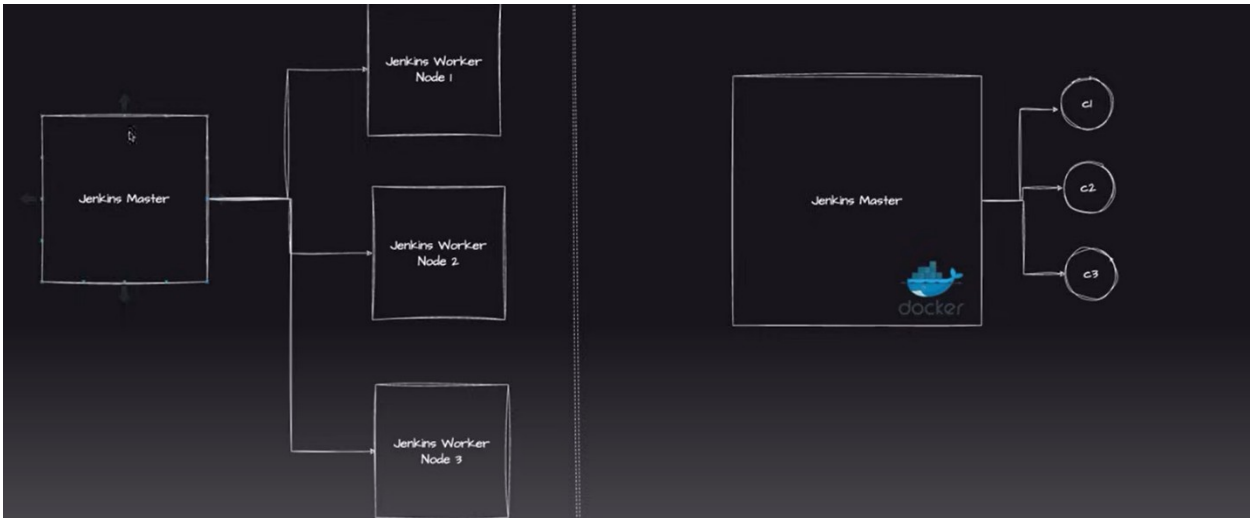
Jenkins URL: http://100.26.160.84:8080/

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 `BUILD_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.

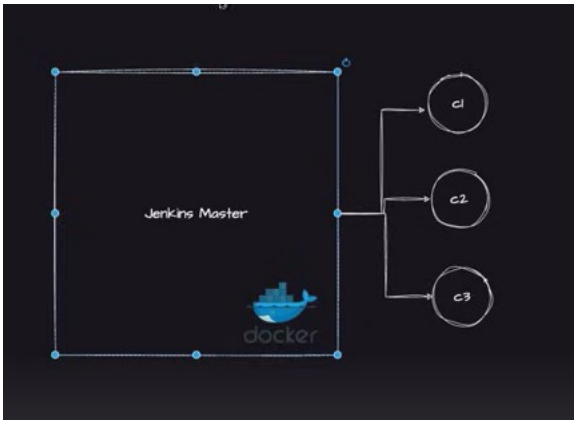






Note .. Always use Jenkins master to schedule the task .. & offload

in this Project We use Jenkins with Docker as Agents



now install Docker

```
ubuntu@ip-172-31-53-221:~$ sudo apt install docker.io
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  bridge-utils containerd dns-root-data dnsmasq-base pigz runc ubuntu-fan
Suggested packages:
  ifupdown aufs-tools cgroupfs-mount | cgroup-lite debootstrap docker-doc rinse zfs-fuse | zfsutils
```

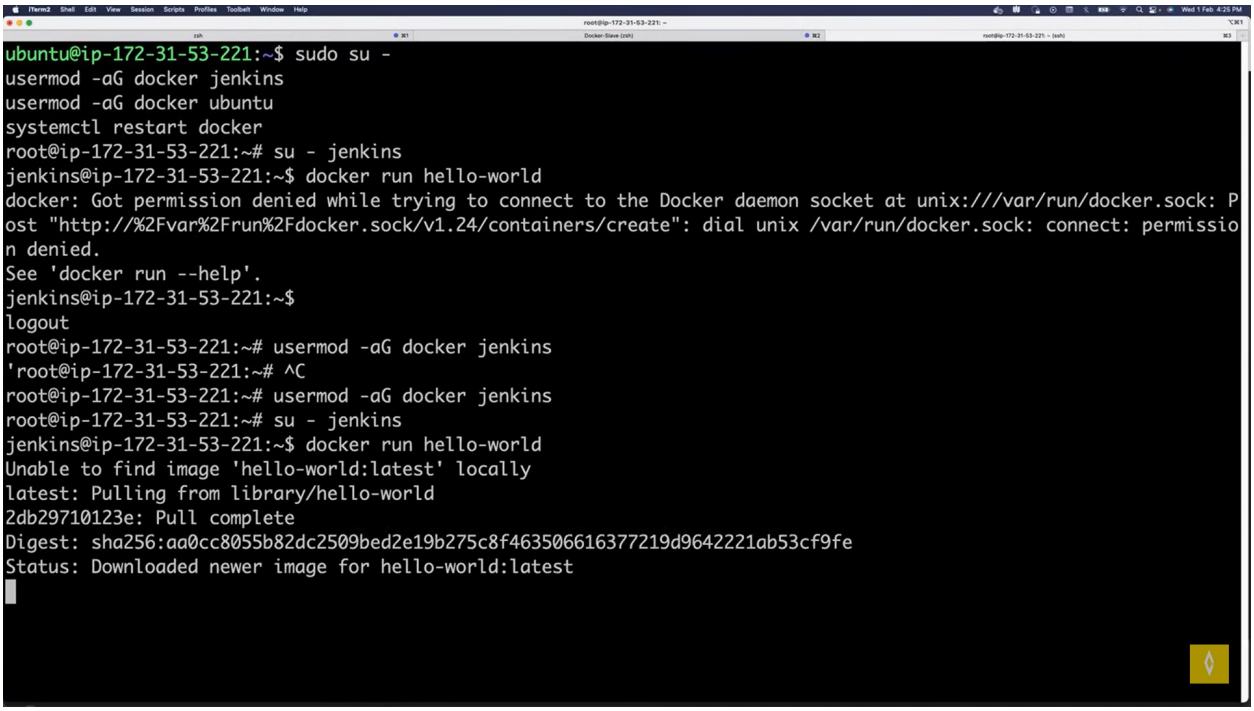
Grant Jenkins user and Ubuntu user permission to docker deamon.

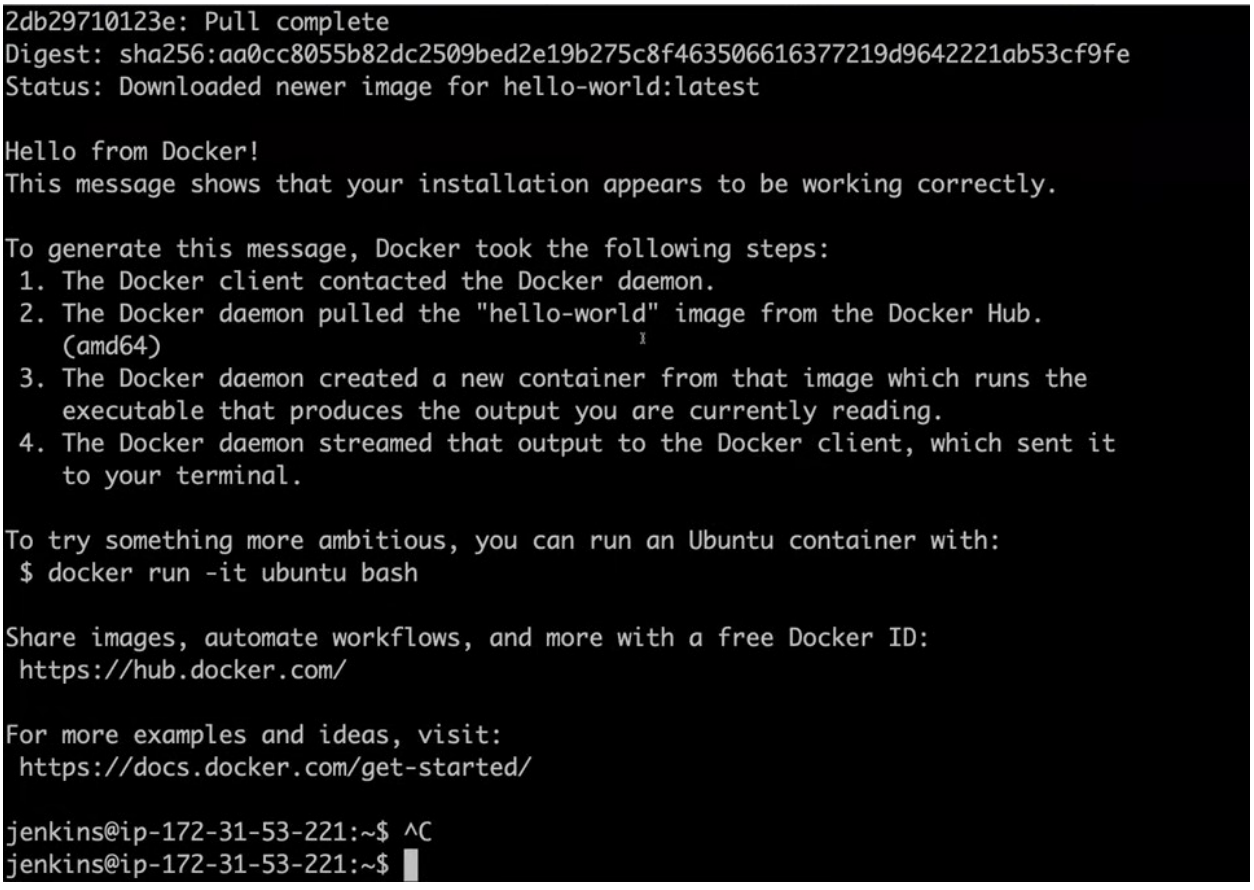
```
sudo su -
usermod -aG docker jenkins
usermod -aG docker ubuntu
systemctl restart docker
```

Once you are done with the above steps, it is better to restart Jenkins.

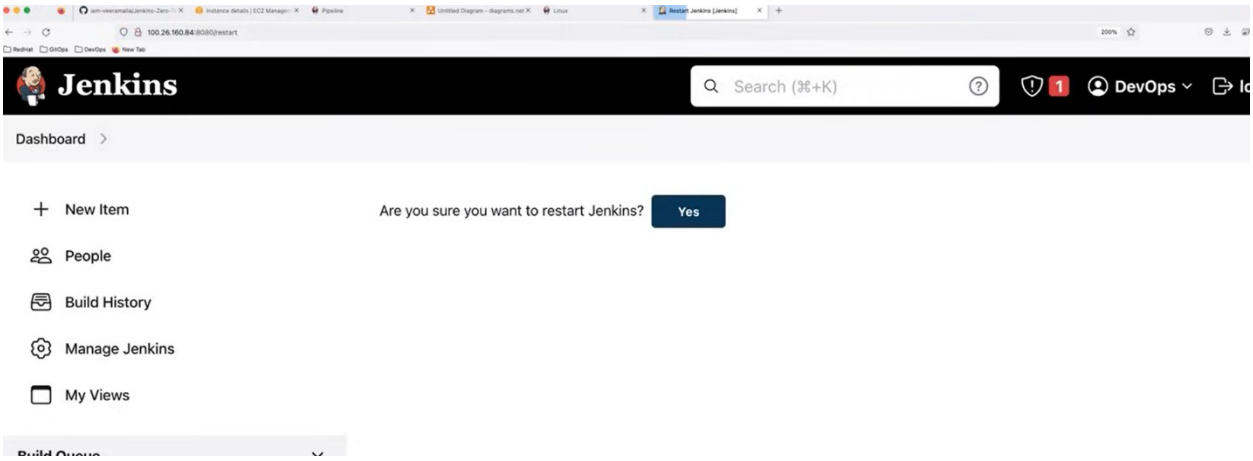
```
http://<ec2-instance-public-ip>:8080/restart
```

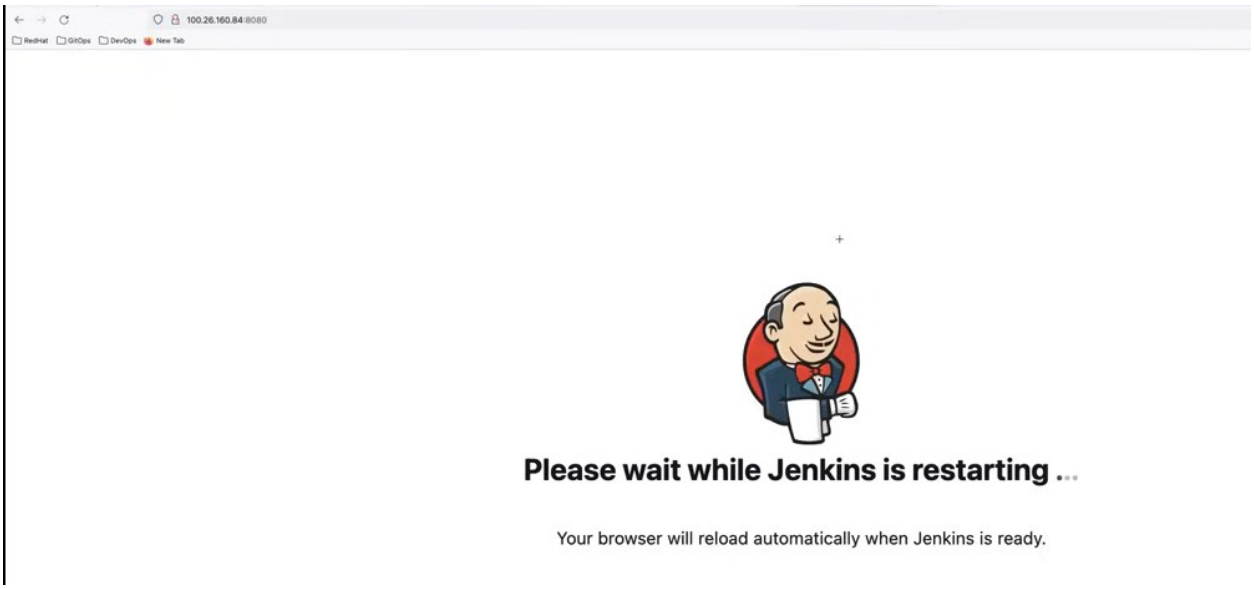
The docker agent configuration is now successful.

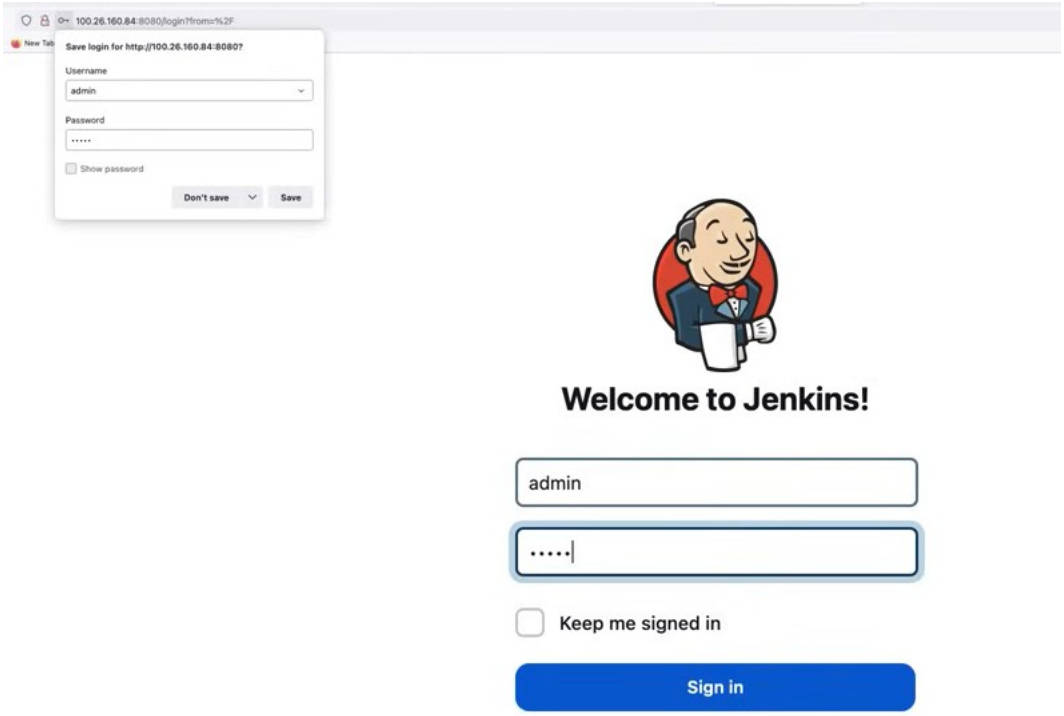




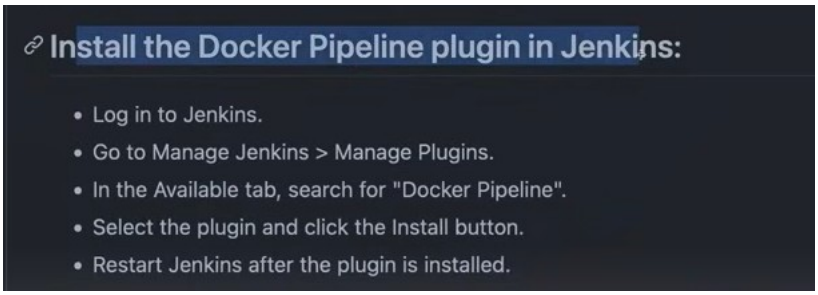
Restart Jenkins


100.26.160.84:8080/restart
Jenkins
Search (⌘+K)
1
DevOps
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Are you sure you want to restart Jenkins?
Yes

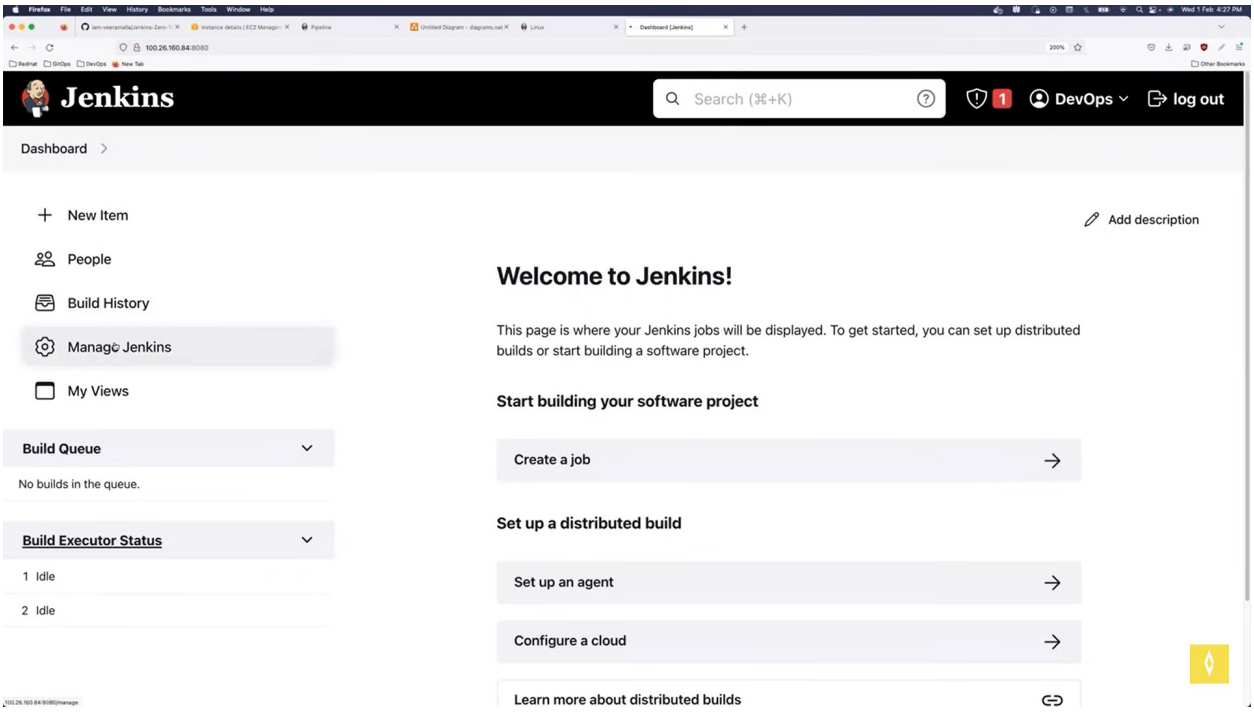

100.26.160.84:8080
Please wait while Jenkins is restarting ...
Your browser will reload automatically when Jenkins is ready.



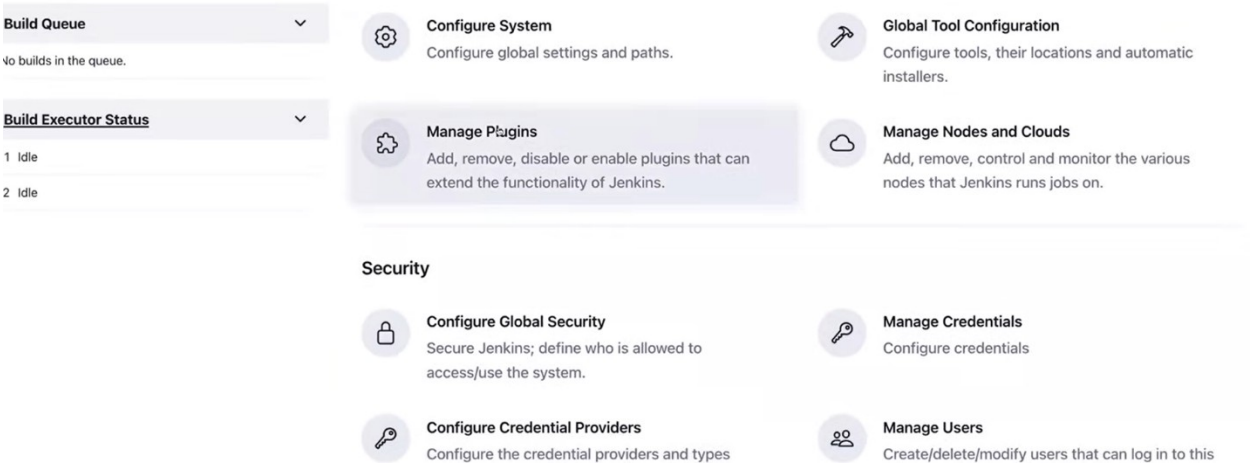
now Install Docker Plugins inside the Jenkins

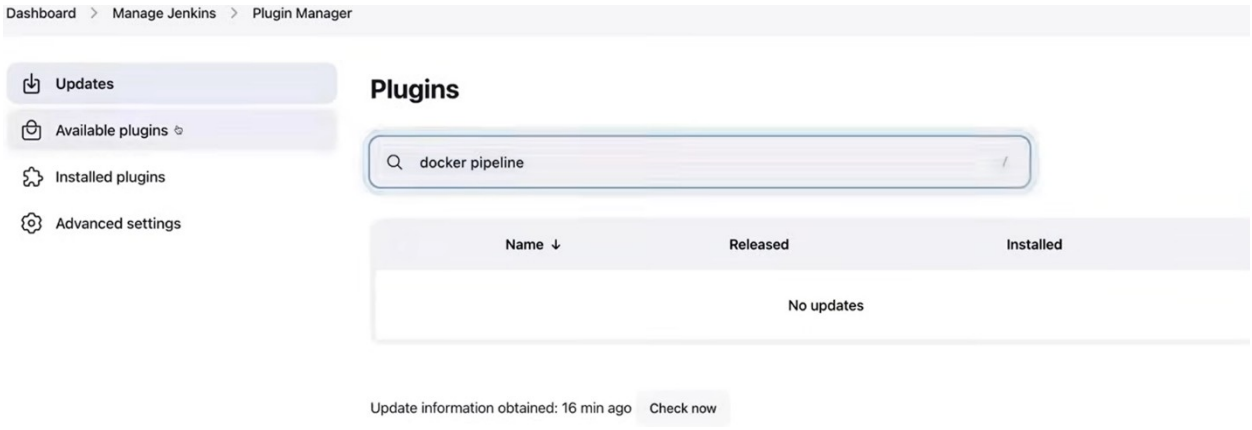


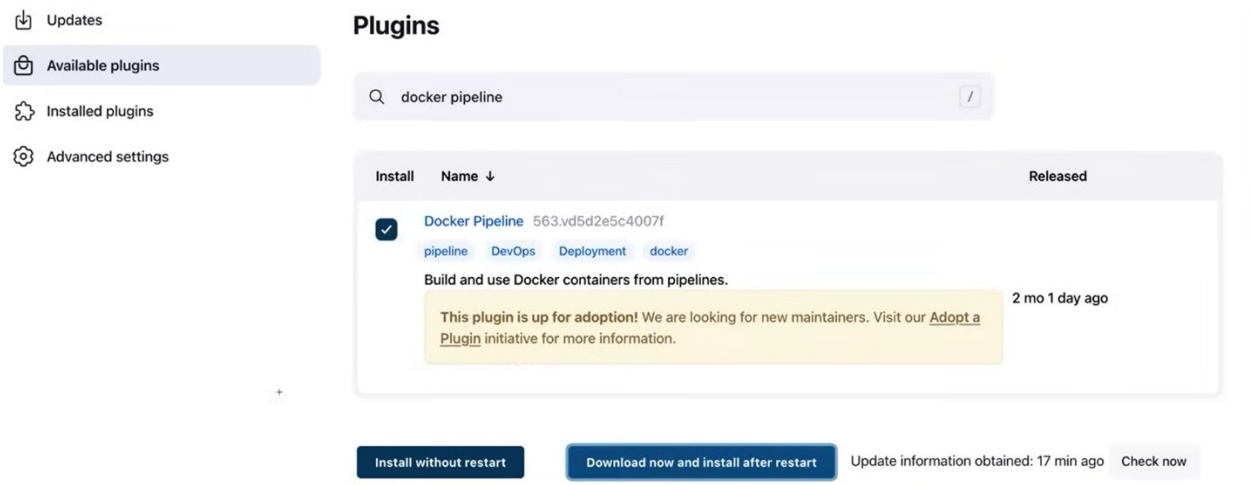
Click on Manage Jenkins

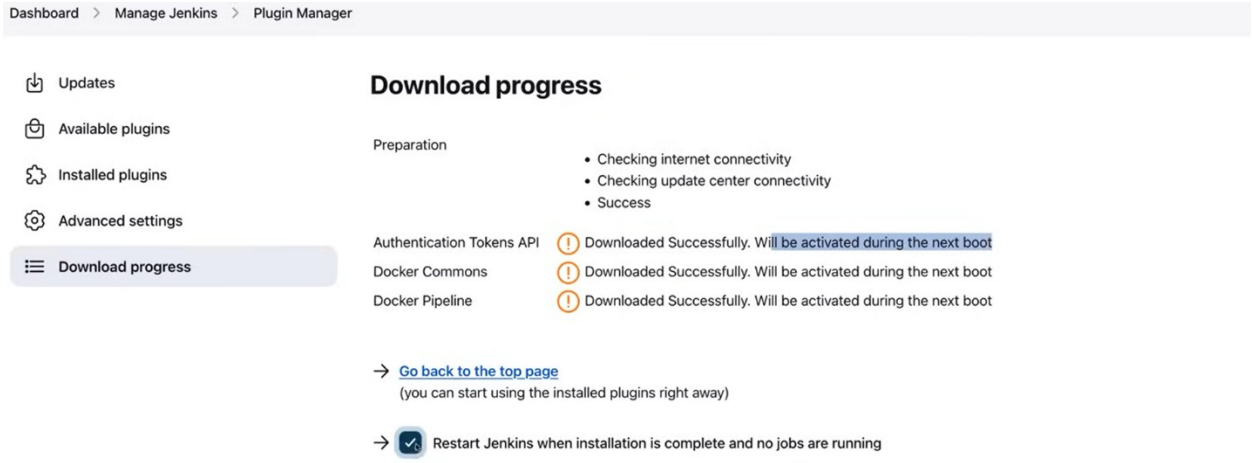


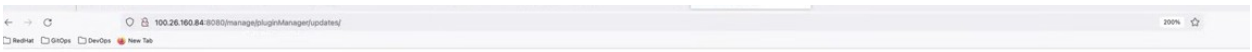
Click on manage Plugins

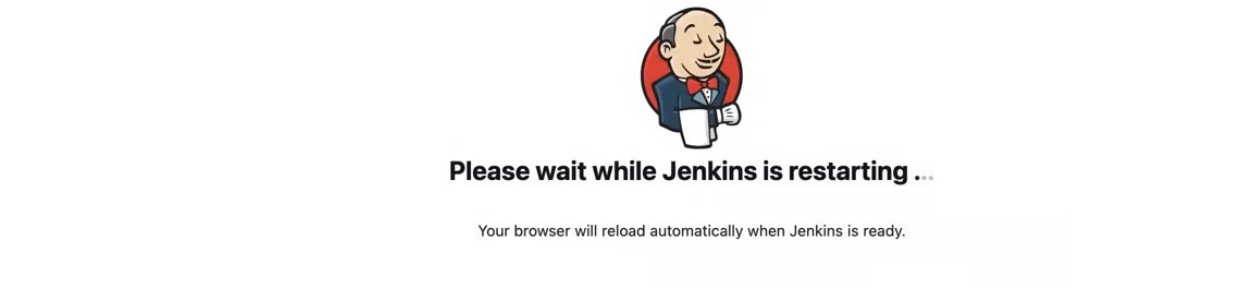


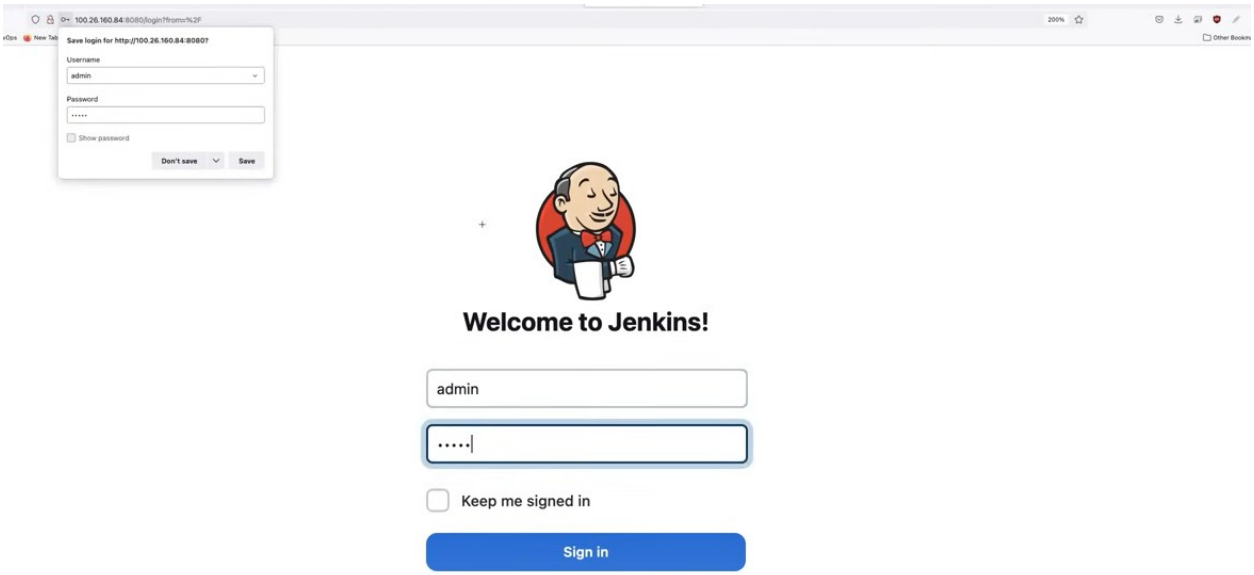

Dashboard > Manage Jenkins > Plugin Manager
Updates
Available plugins
Installed plugins
Advanced settings
Plugins
docker pipeline
Name ↓
Released
Installed
No updates
Update information obtained: 16 min ago
Check now


Updates
Available plugins
Installed plugins
Advanced settings
Plugins
docker pipeline
Install
Name ↓
Released
Docker Pipeline 563.vd5d2e5c4007f
pipeline
DevOps
Deployment
docker
Build and use Docker containers from pipelines.
This plugin is up for adoption! We are looking for new maintainers. Visit our Adopt a Plugin initiative for more information.
2 mo 1 day ago
Install without restart
Download now and install after restart
Update information obtained: 17 min ago
Check now

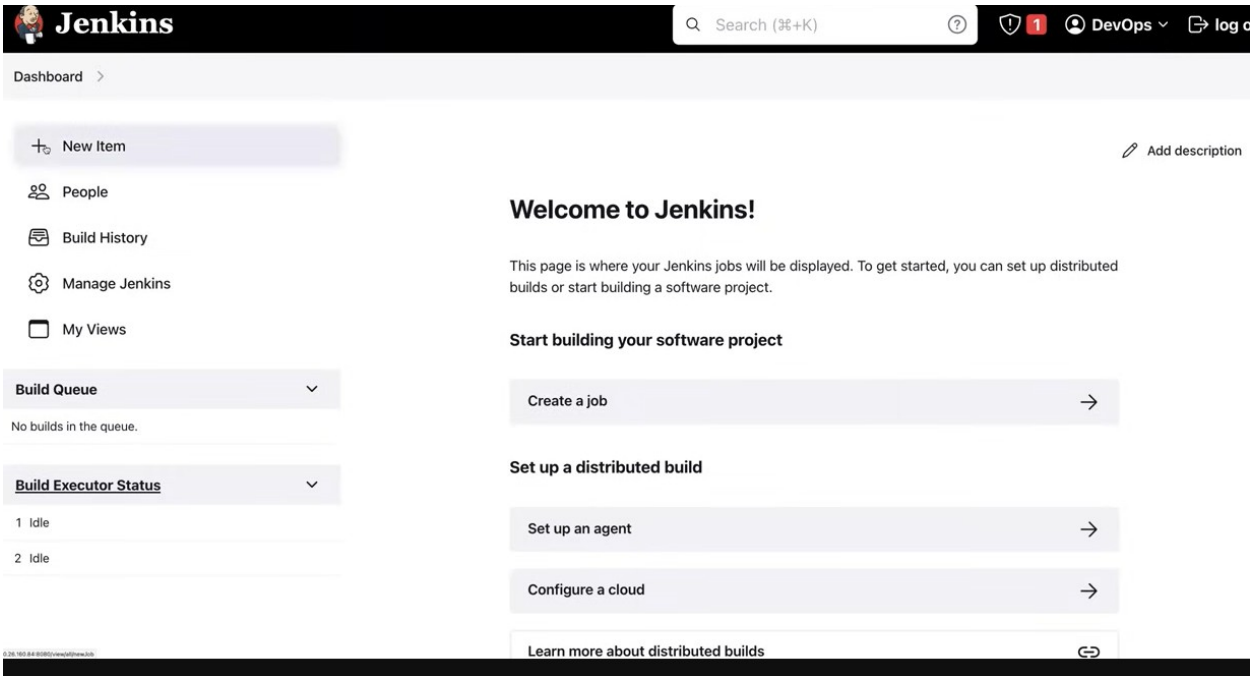

Dashboard > Manage Jenkins > Plugin Manager
Updates
Available plugins
Installed plugins
Advanced settings
Download progress
Download progress
Preparation
Checking internet connectivity
Checking update center connectivity
Success
Authentication Tokens API
Docker Commons
Docker Pipeline
Downloaded Successfully. Will be activated during the next boot
Go back to the top page
(you can start using the installed plugins right away)
Restart Jenkins when installation is complete and no jobs are running

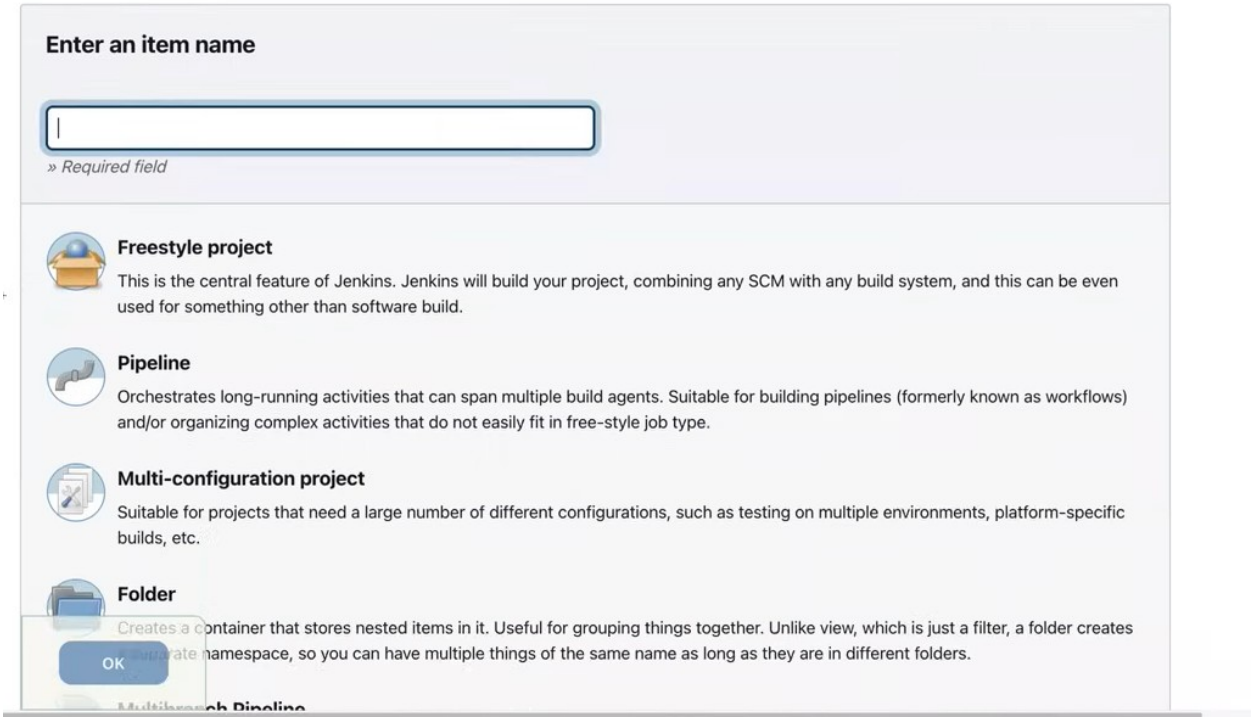


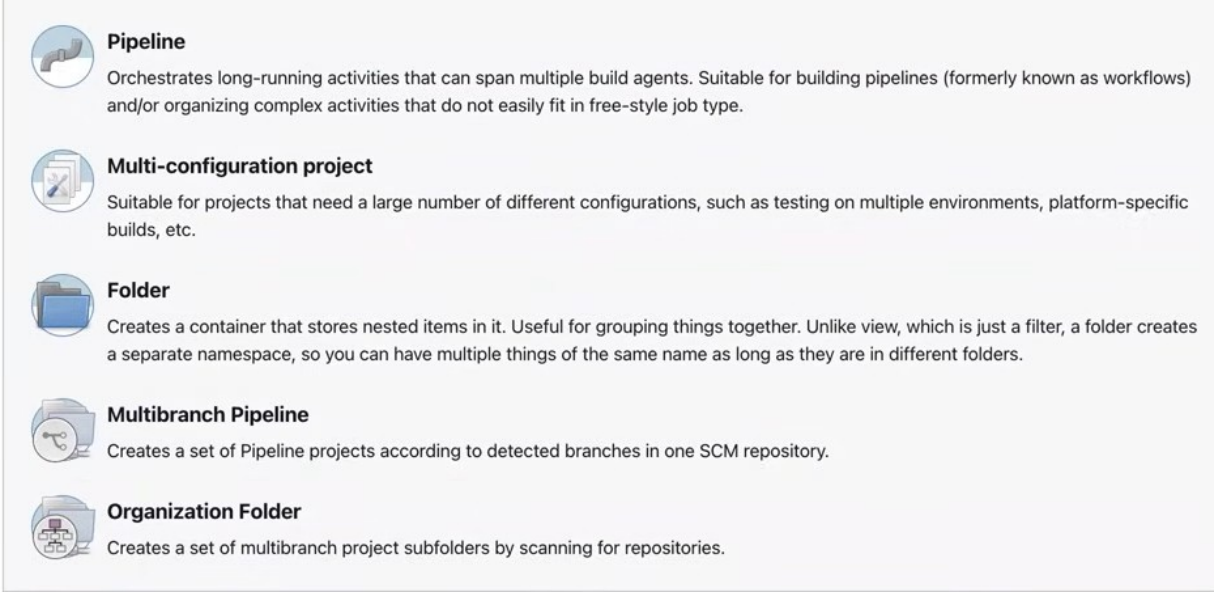




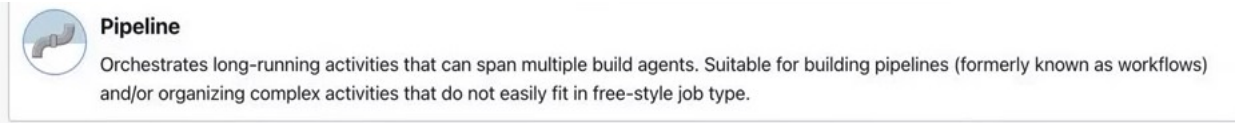
Click on New Item


Jenkins
Search (⌘+K)
1
DevOps
log o
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To get started, you can set up distributed builds or start building a software project.
Start building your software project
Create a job
Set up a distributed build
Set up an agent
Configure a cloud
Learn more about distributed builds

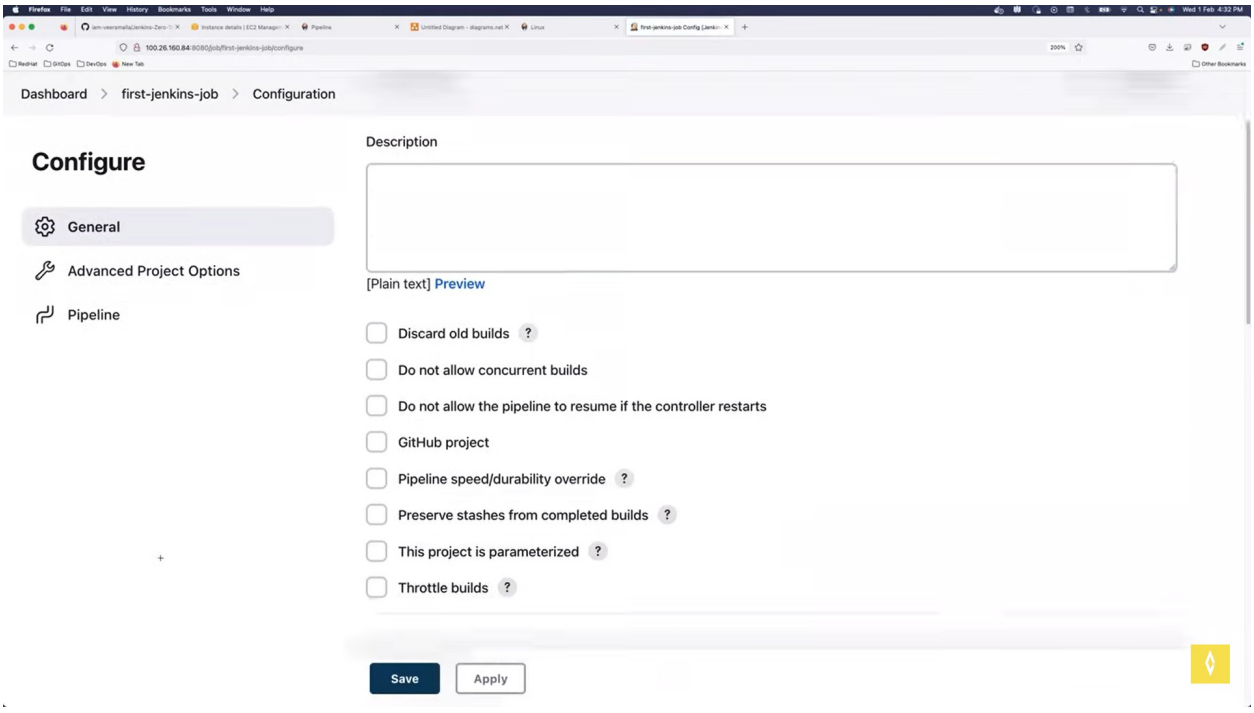

Enter an item name
» Required field
Freestyle project
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any build system, and this can be even used for something other than software build.
Pipeline
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipelines (formerly known as workflows) and/or organizing complex activities that do not easily fit in free-style job type.
Multi-configuration project
Suitable for projects that need a large number of different configurations, such as testing on multiple environments, platform-specific builds, etc.
Folder
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, which is just a filter, a folder creates
namespace, so you can have multiple things of the same name as long as they are in different folders.
OK

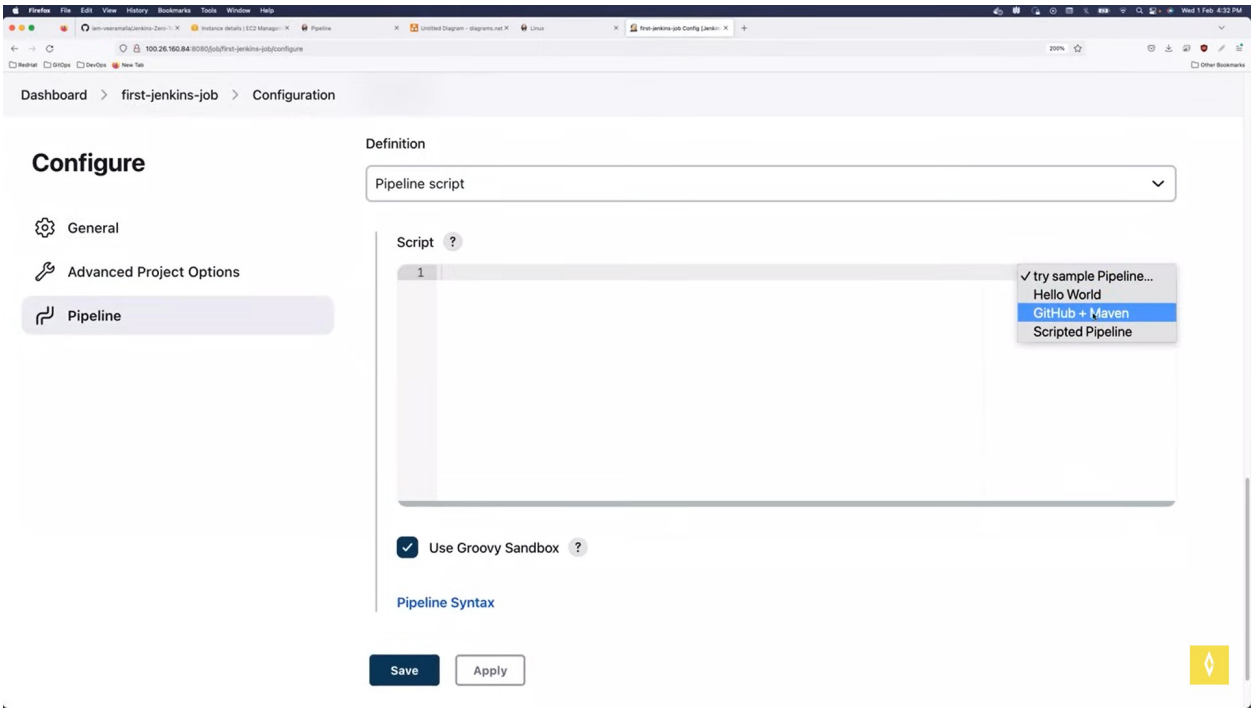


select

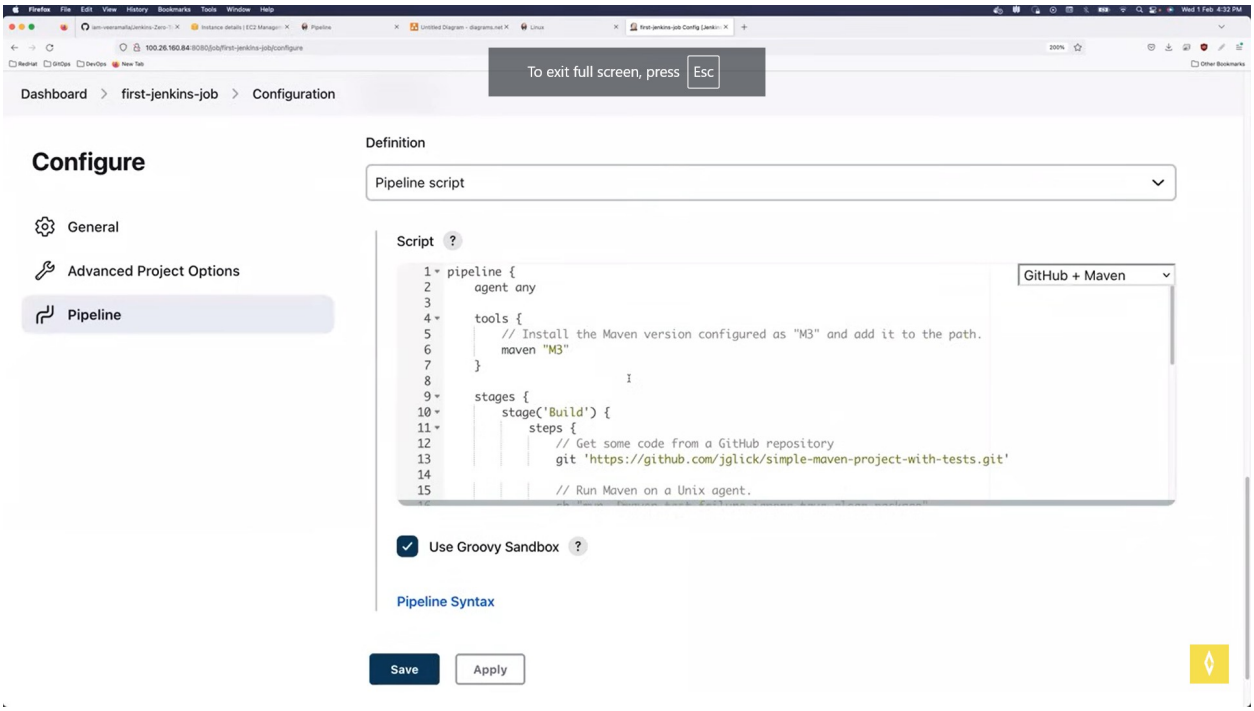


<u>Advantage of selecting pipeline is we can write our own code (declarative or scripted pipelines)</u>

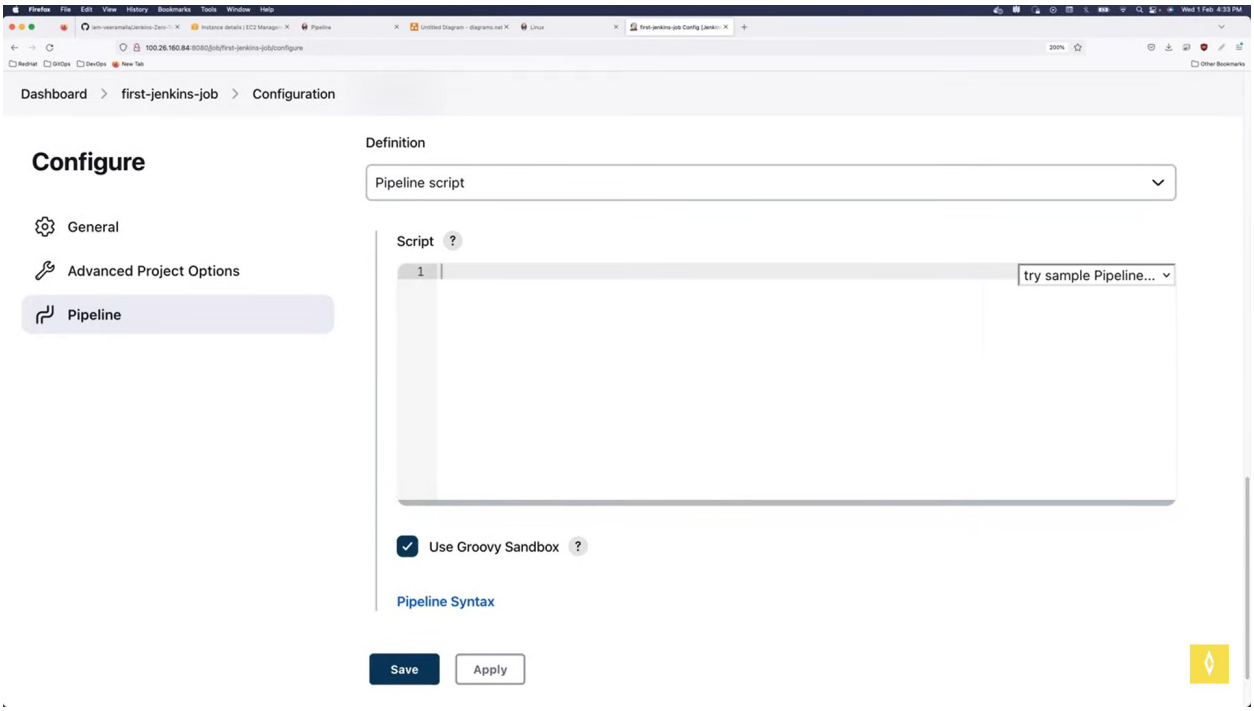




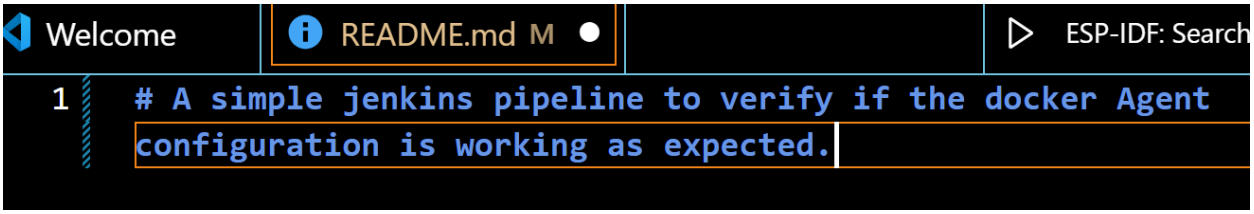
inbuilt template Draft

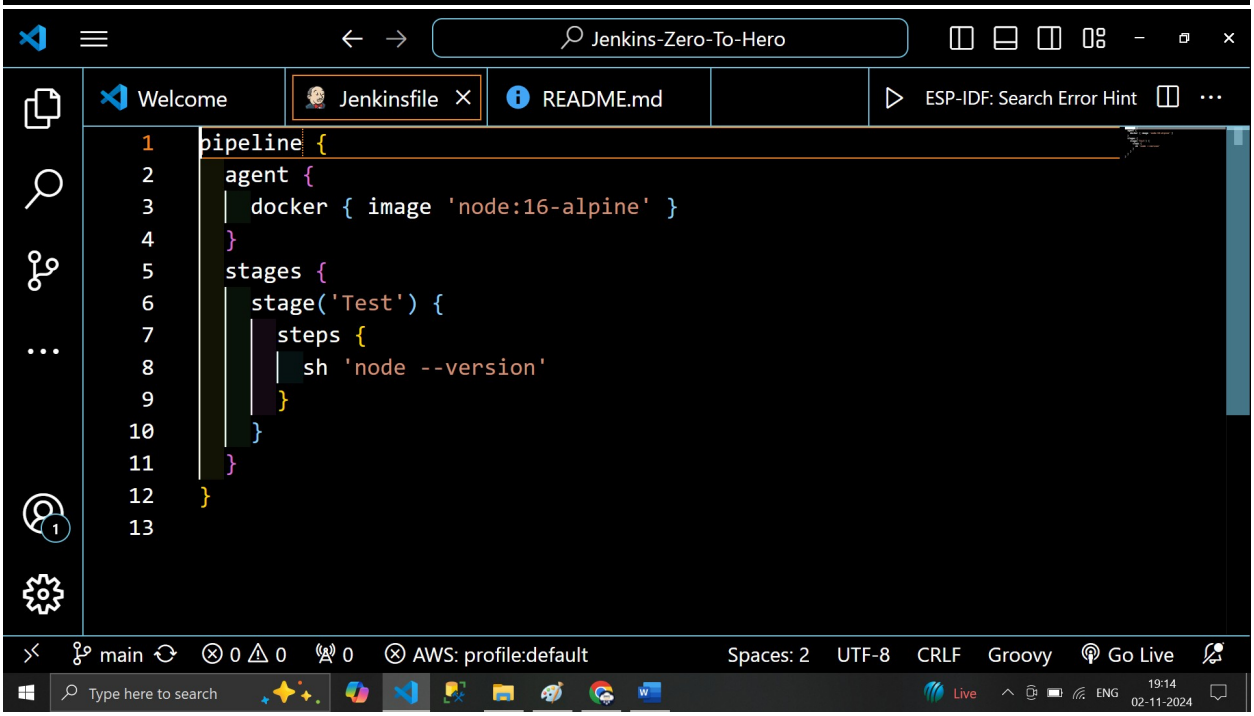

To exit full screen, press Esc
Dashboard > first-jenkins-job > Configuration
Configure
General
Advanced Project Options
Pipeline
Definition
Pipeline script
Script
GitHub + Maven
pipeline {
agent any
tools {
// Install the Maven version configured as "M3" and add it to the path.
maven "M3"
}
stages {
stage('Build') {
steps {
// Get some code from a GitHub repository
git 'https://github.com/jglick/simple-maven-project-with-tests.git'
// Run Maven on a Unix agent.
Use Groovy Sandbox
Pipeline Syntax
Save
Apply

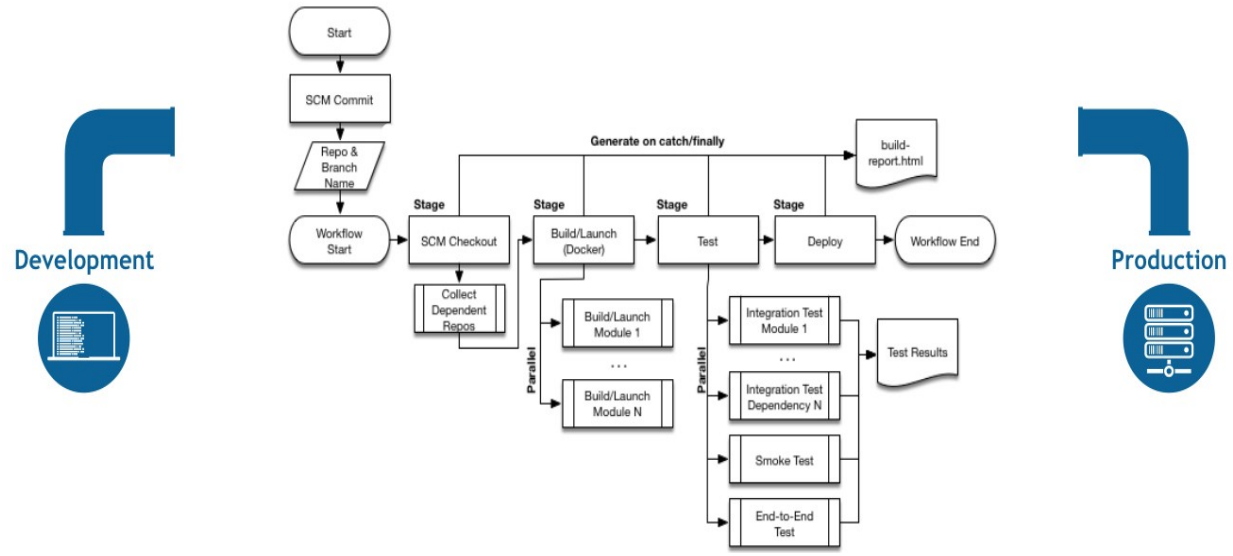

Configure
General
Advanced Project Options
Pipeline
Pipeline script
Script
GitHub + Maven
pipeline {
agent any
tools {
// Install the Maven version configured as "M3" and add it to the path.
maven "M3"
}
stages {
stage('Build') {
steps {
// Get some code from a GitHub repository
git 'https://github.com/jglick/simple-maven-project-with-tests.git'
// Run Maven on a Unix agent.
Use Groovy Sandbox

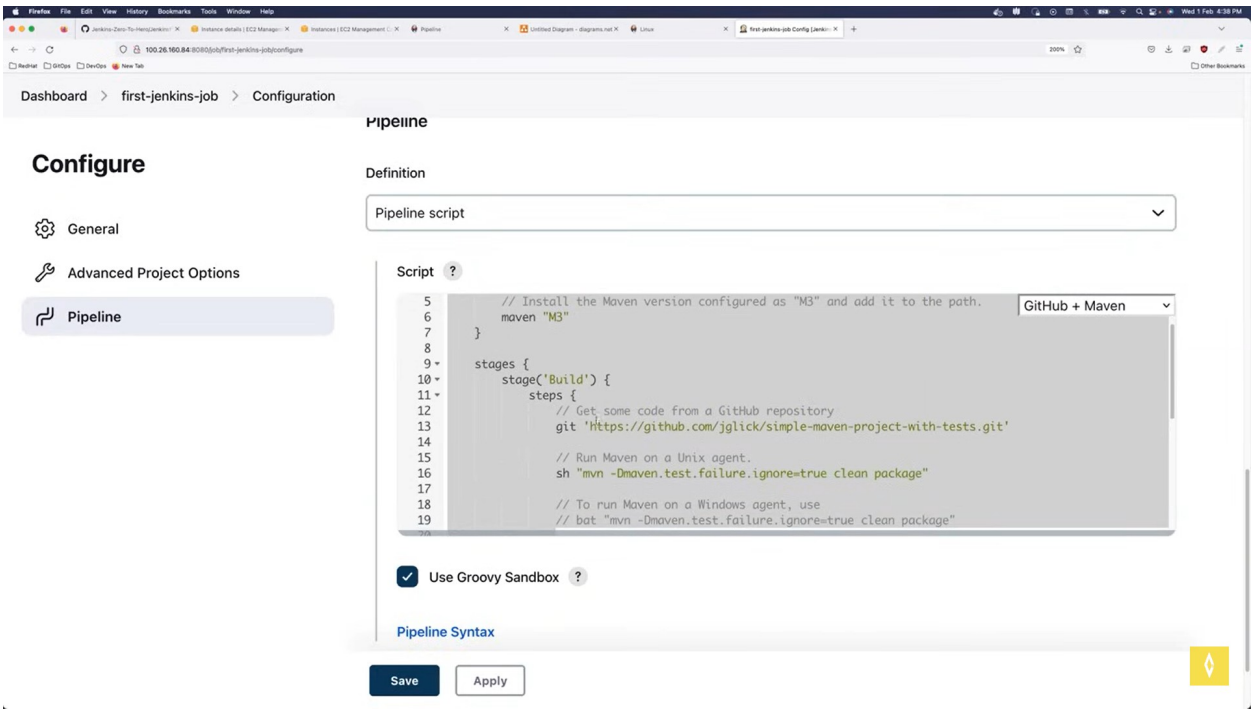

Dashboard
first-jenkins-job
Configuration
Configure
General
Advanced Project Options
Pipeline
Definition
Pipeline script
Script
try sample Pipeline...
Use Groovy Sandbox
Pipeline Syntax
Save
Apply

jenkinfile

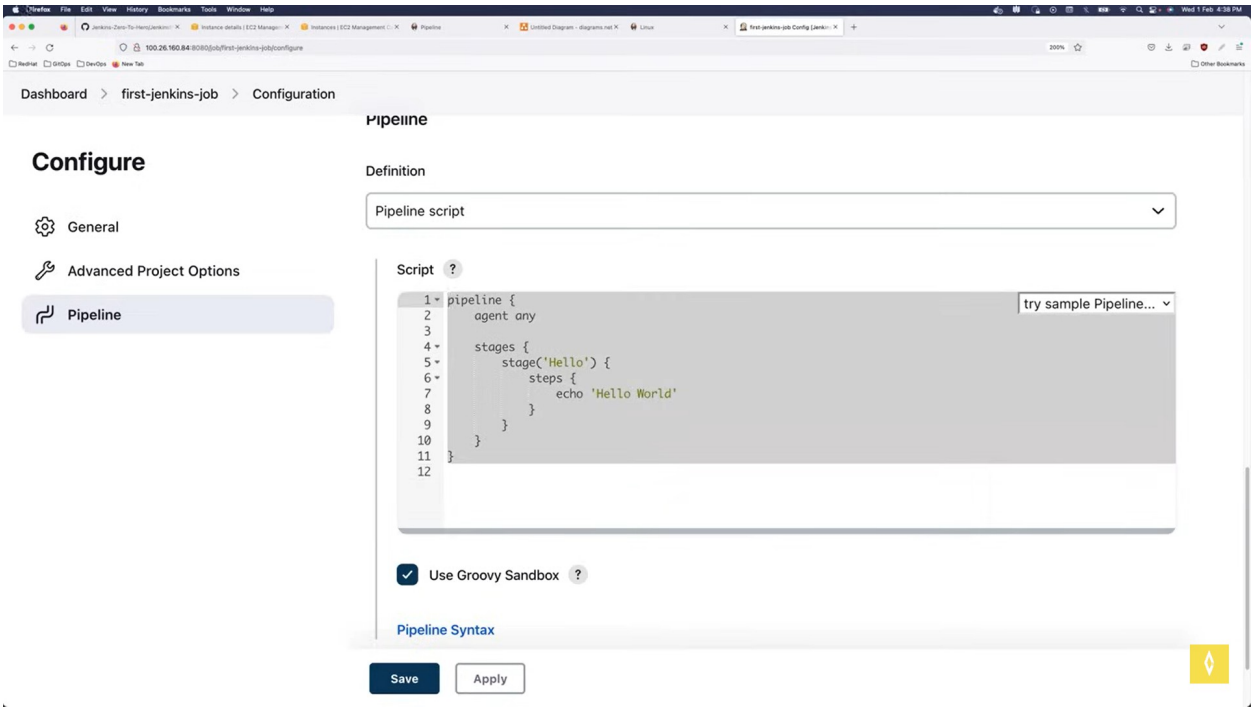

Welcome
README.md M
ESP-IDF: Search
1
A simple jenkins pipeline to verify if the docker Agent configuration is working as expected.


Jenkins-Zero-To-Hero
Welcome
Jenkinsfile
README.md
ESP-IDF: Search Error Hint
pipeline {
agent {
docker { image 'node:16-alpine' }
}
stages {
stage('Test') {
steps {
sh 'node --version'
}
}
}
}
main
AWS: profile:default
Spaces: 2
UTF-8
CRLF
Groovy
Go Live
Type here to search
19:14
02-11-2024

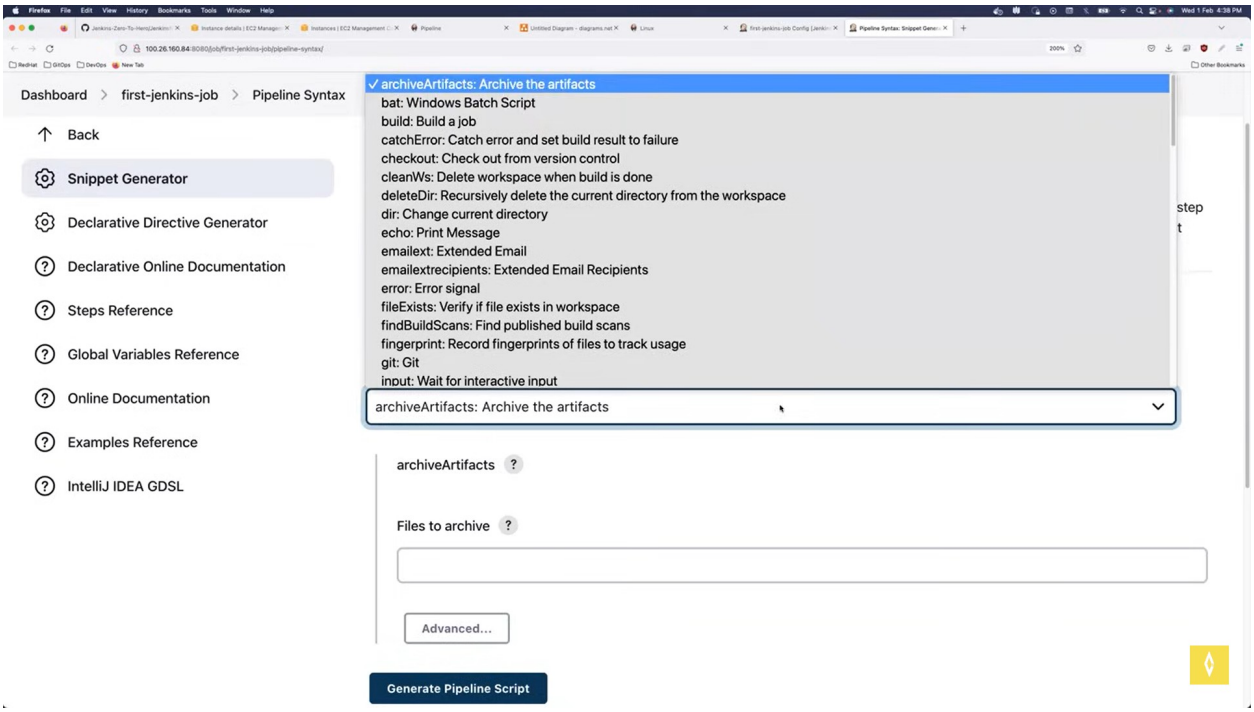

Development
Start
SCM Commit
Repo & Branch Name
Workflow Start
Stage
SCM Checkout
Collect Dependent Repos
Stage
Build/Launch (Docker)
Parallel
Build/Launch Module 1
Build/Launch Module N
Stage
Test
Parallel
Integration Test Module 1
Integration Test Dependency N
Smoke Test
End-to-End Test
Test Results
Stage
Deploy
Generate on catch/finally
build-report.html
Workflow End
Production

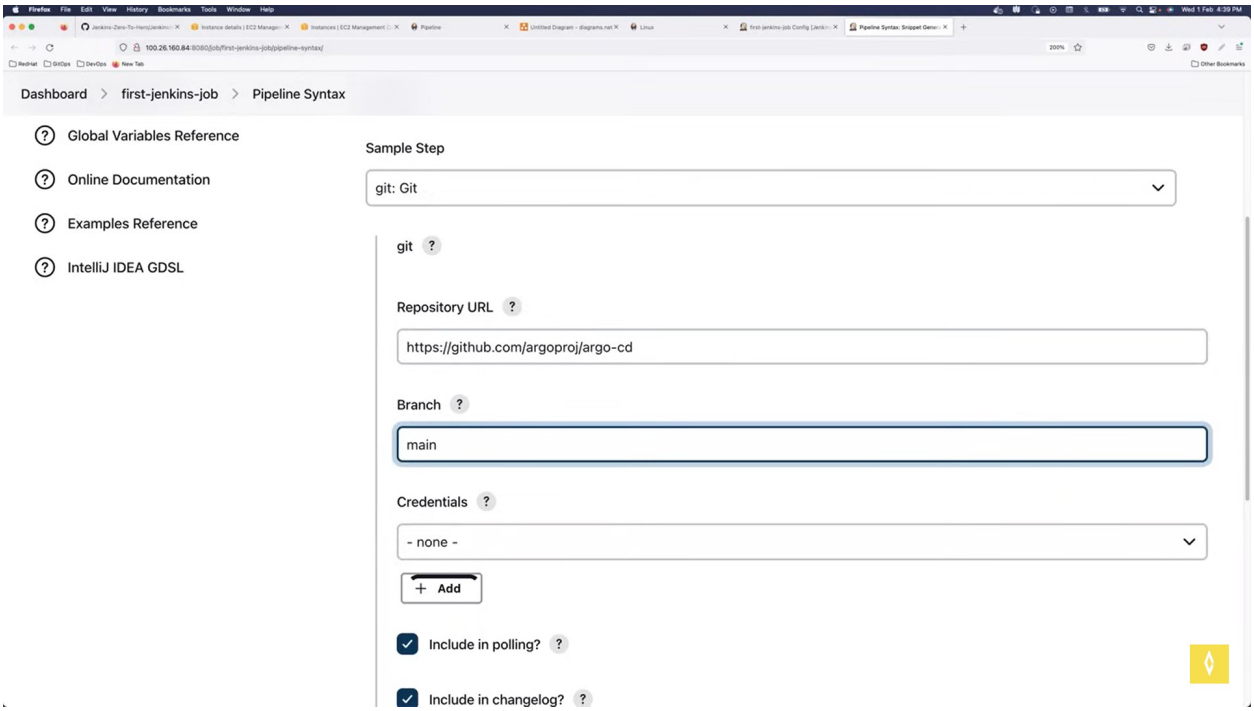

Dashboard
first-jenkins-job
Configuration
Configure
General
Advanced Project Options
Pipeline
Pipeline
Definition
Pipeline script
Script
GitHub + Maven
// Install the Maven version configured as "M3" and add it to the path.
maven "M3"
}
stages {
stage('Build') {
steps {
// Get some code from a GitHub repository
git 'https://github.com/jglick/simple-maven-project-with-tests.git'
// Run Maven on a Unix agent.
sh "mvn -Dmaven.test.failure.ignore=true clean package"
// To run Maven on a Windows agent, use
// bat "mvn -Dmaven.test.failure.ignore=true clean package"
Use Groovy Sandbox
Pipeline Syntax
Save
Apply

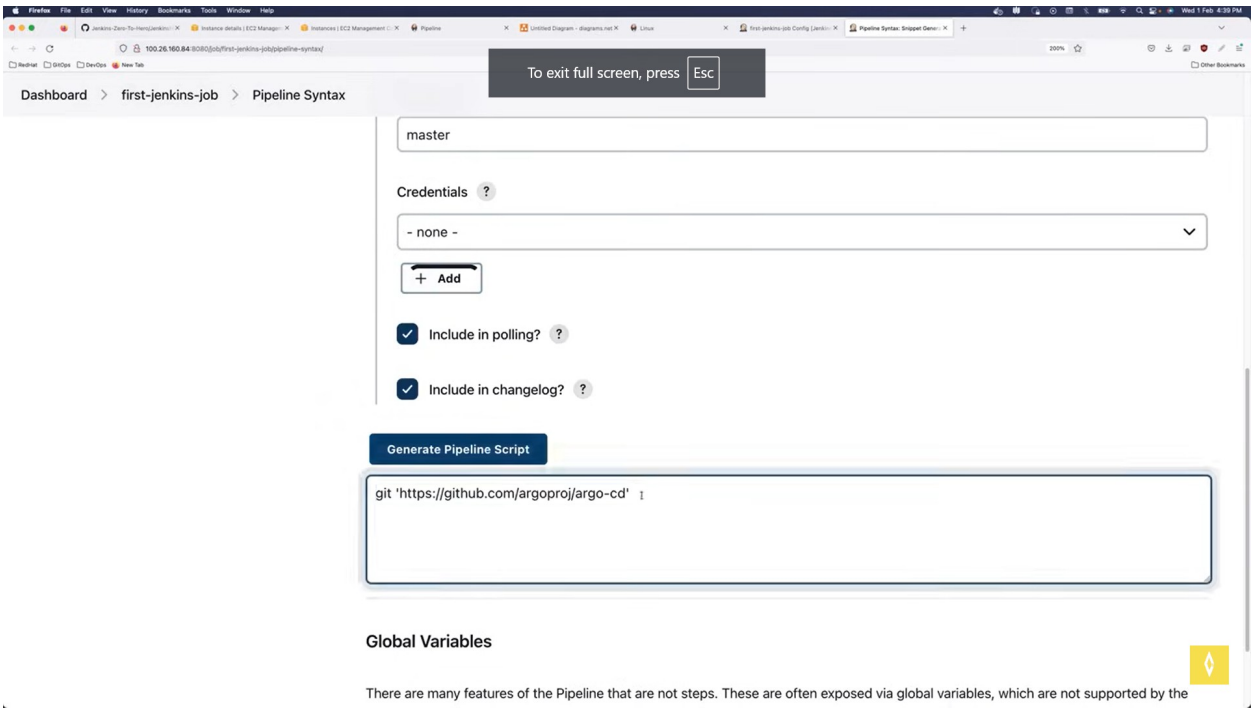
Sample



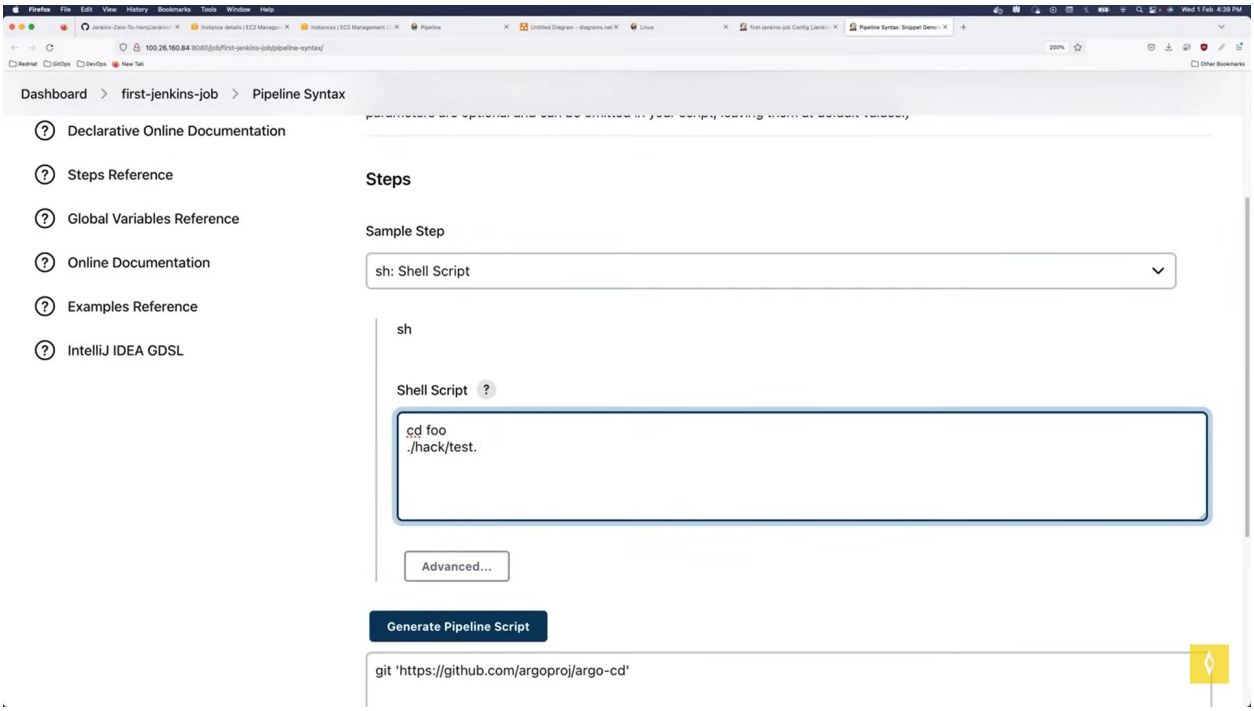
Click on Pipeline Syntax

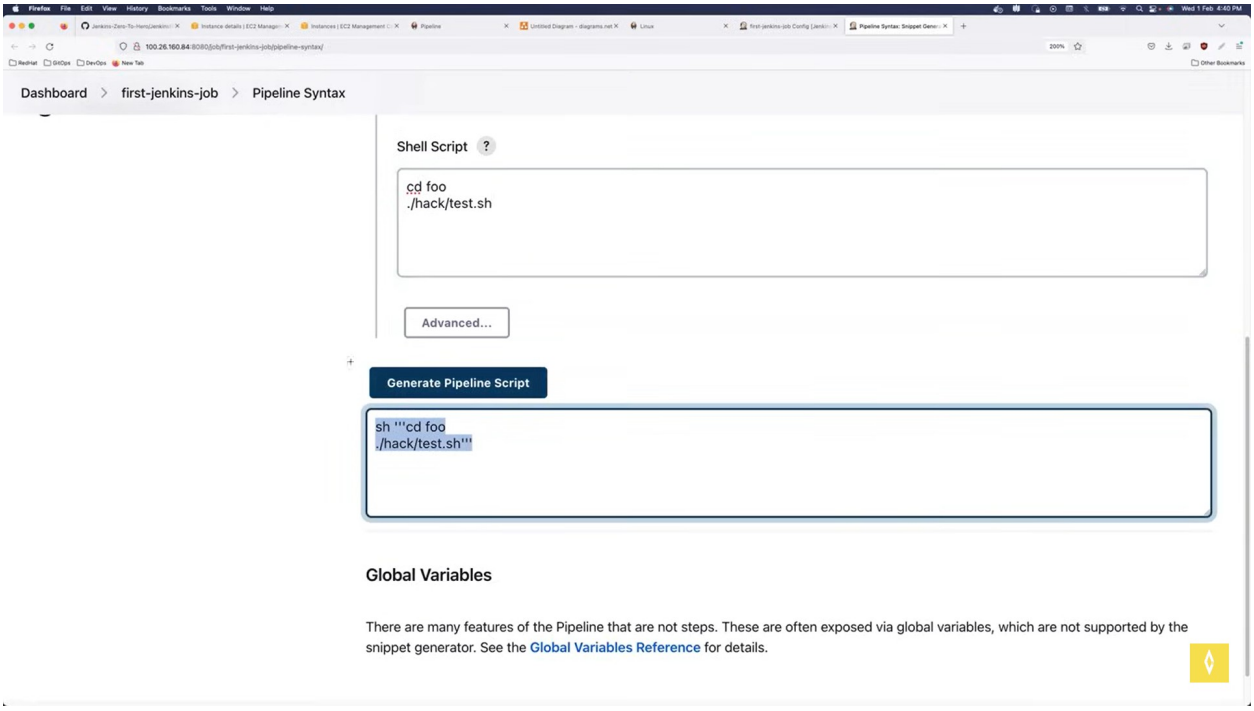




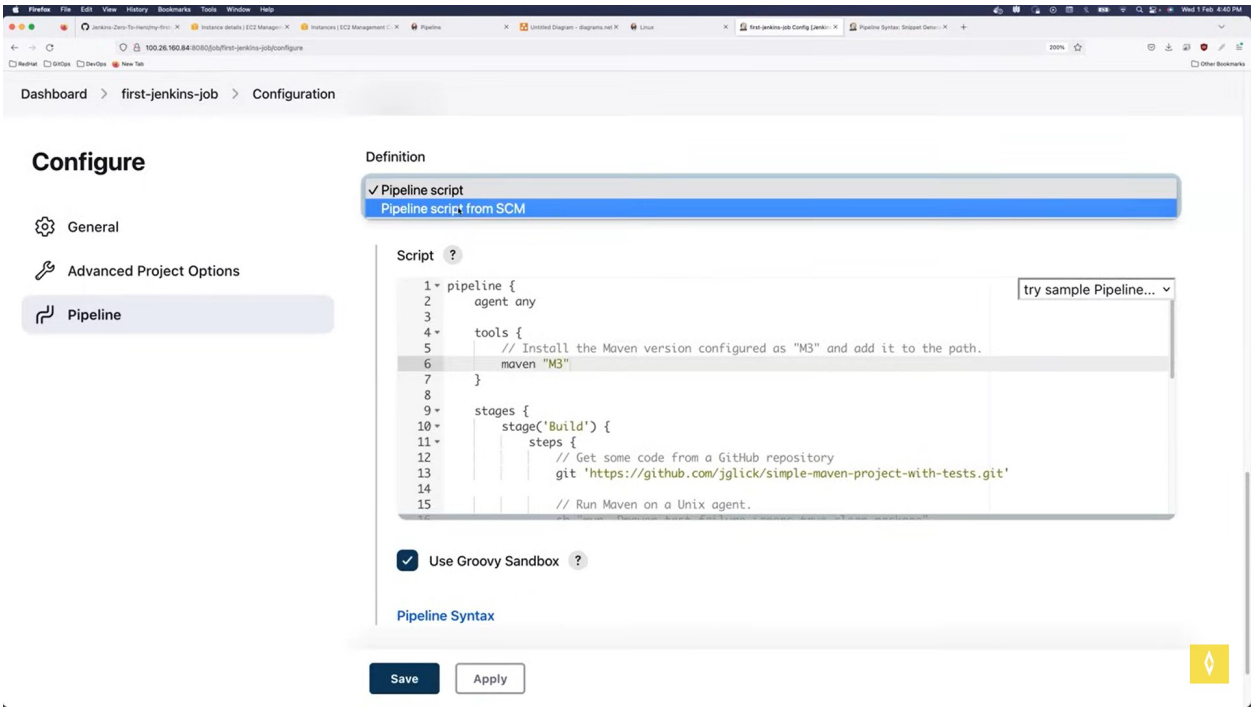


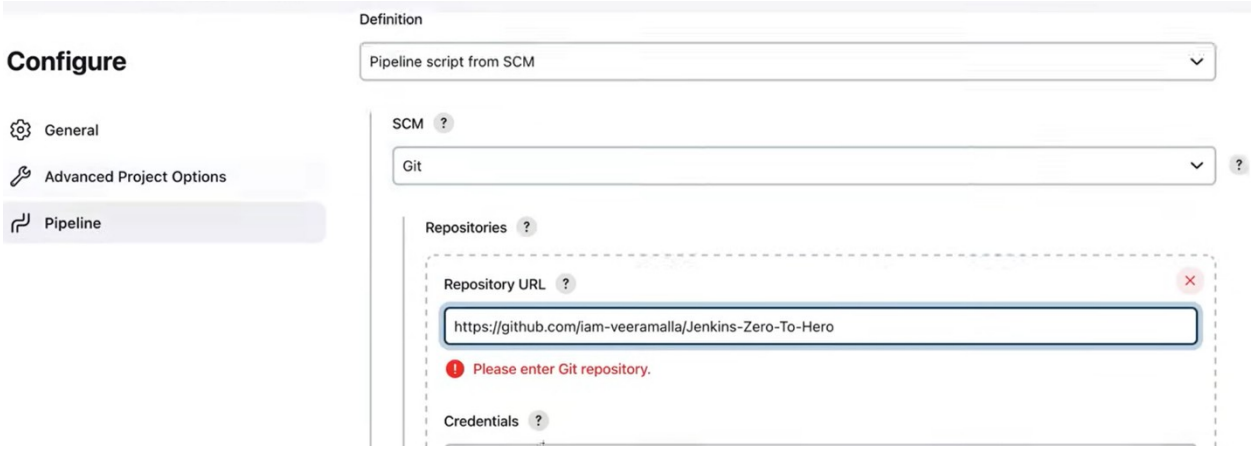
Shell Script



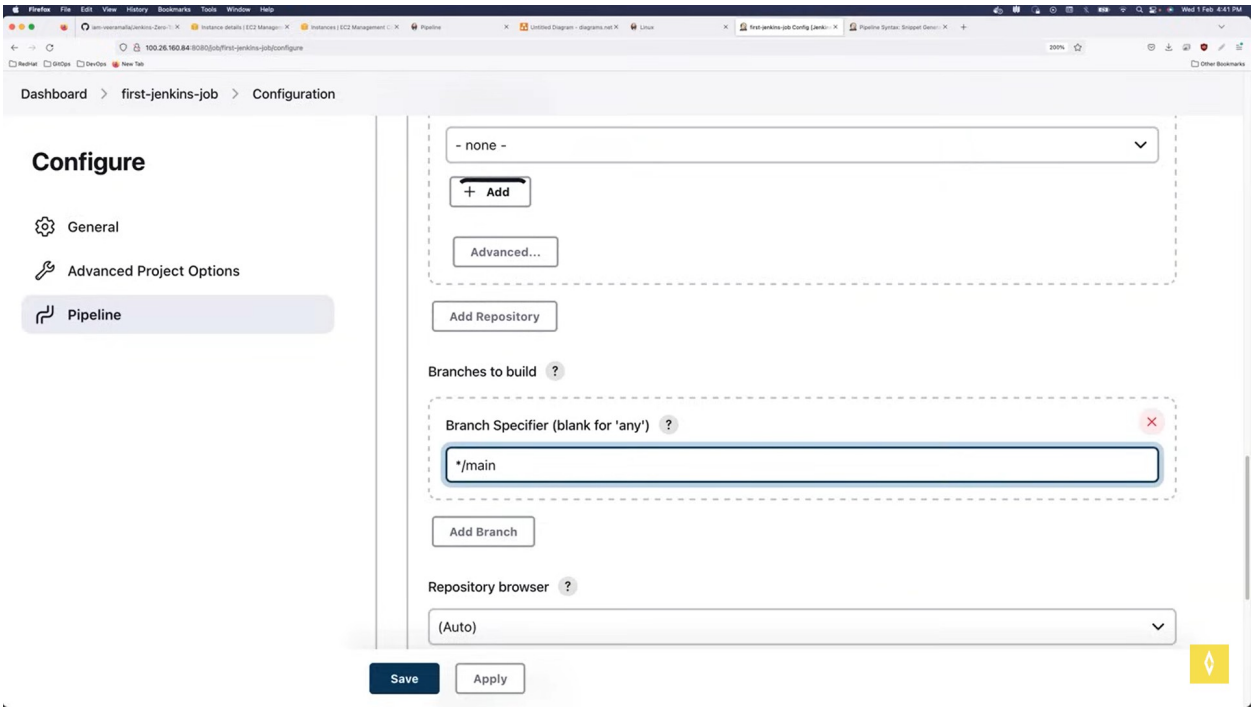


Pipeline script from SCM – Source Code Management Repo

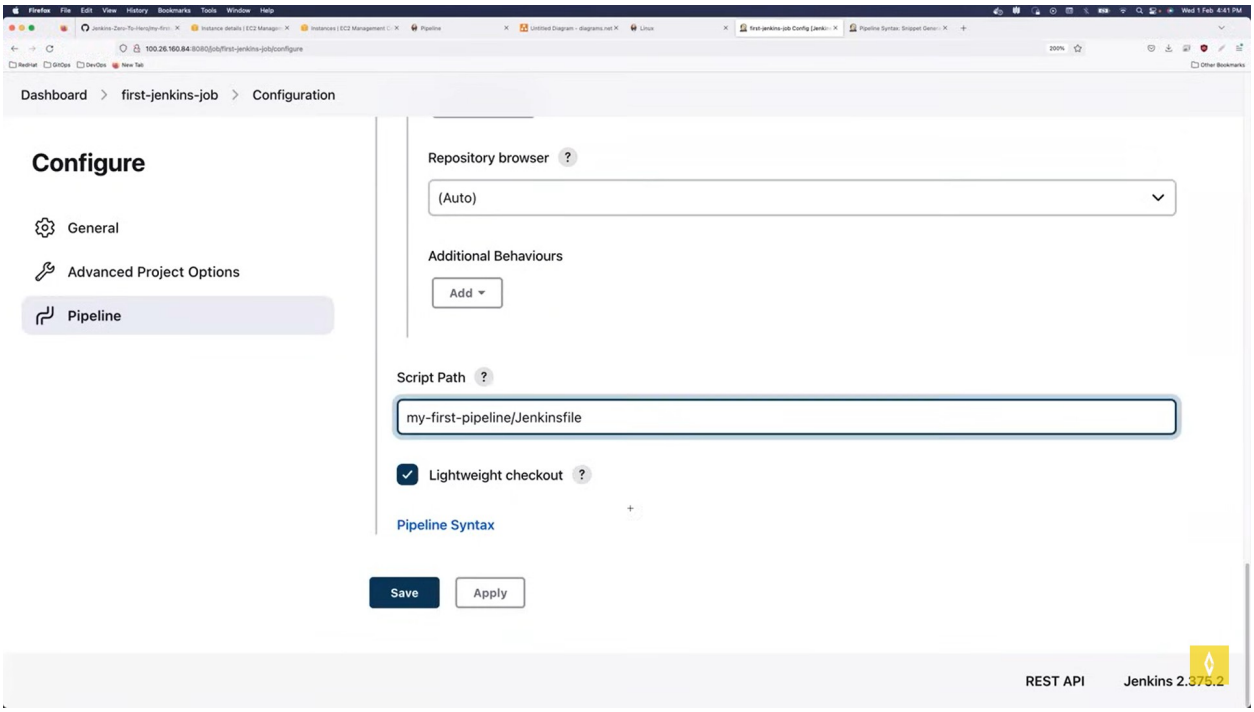




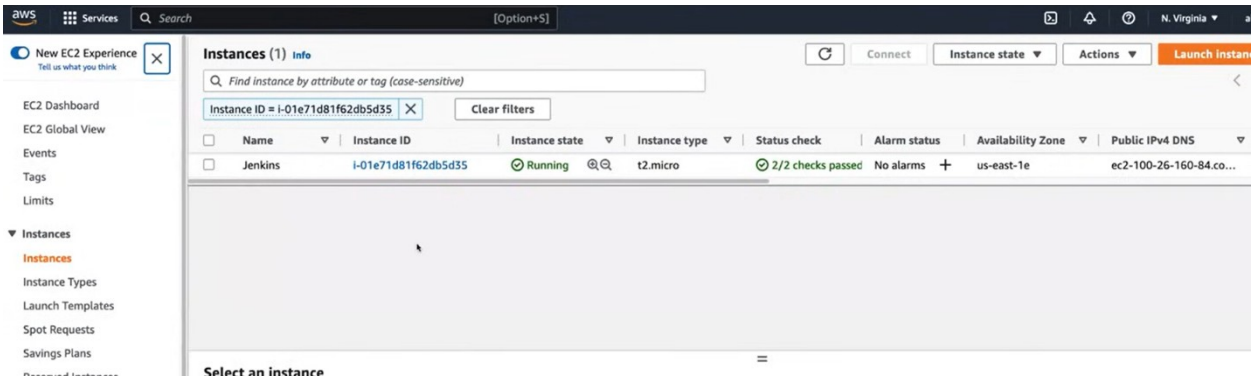
change Branch to Main



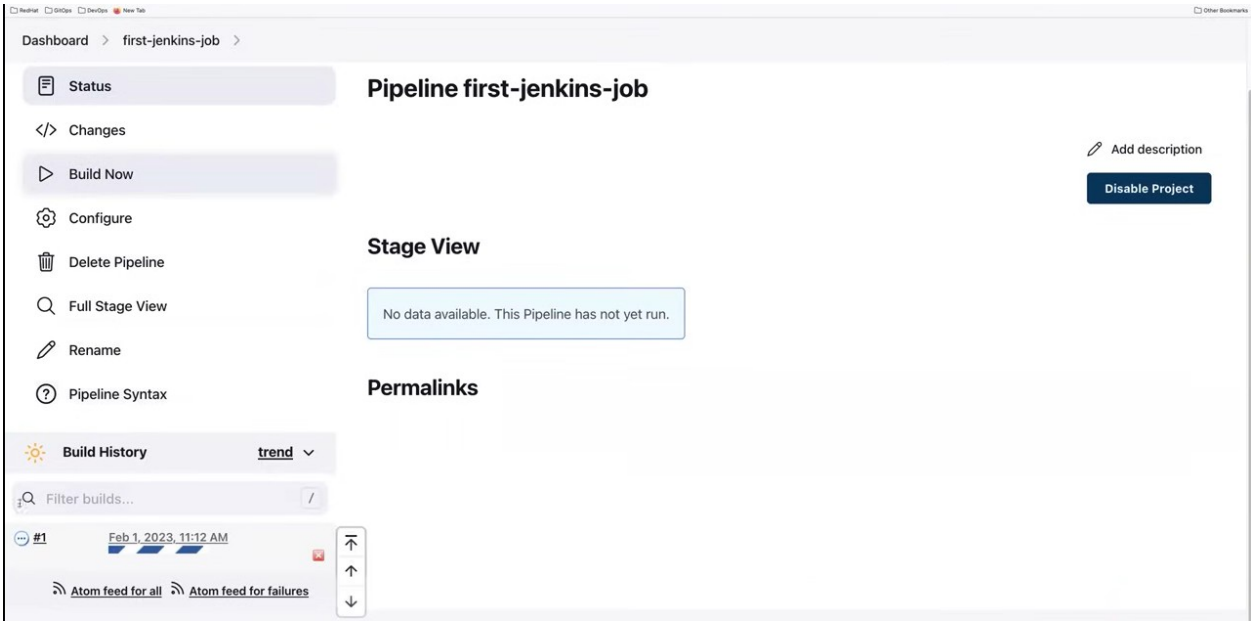
Path



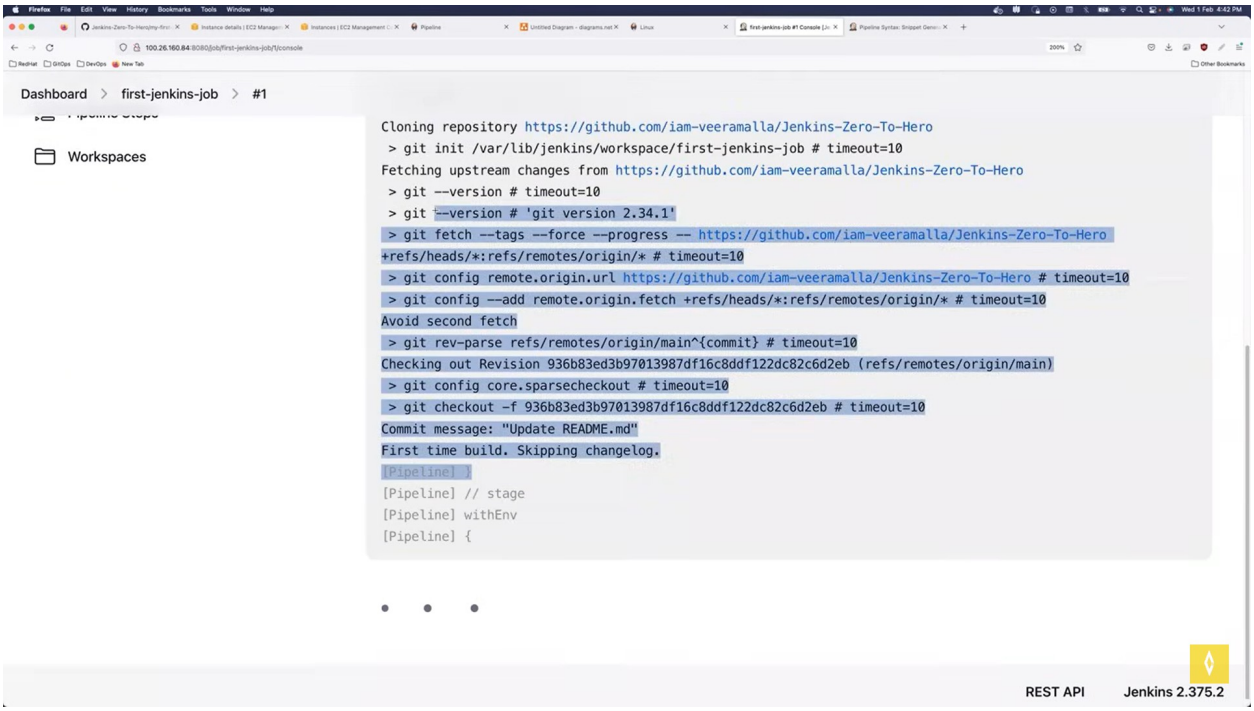
EC2 instance



 # Go back to Jenkins – Click on Build now



now Fetching the Code from Github



in Docker ,, node-16 Alpine img

```
-e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** -e ******** node:16-
alpine cat
$ docker top e7cb3165715254cbb15fa46456261e80b50bb6e7b208044fa8125accb3508c9e -eo pid,comm
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] sh (hide)
+ node --version
v16.19.0
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
$ docker stop --time=1 e7cb3165715254cbb15fa46456261e80b50bb6e7b208044fa8125accb3508c9e
$ docker rm -f --volumes e7cb3165715254cbb15fa46456261e80b50bb6e7b208044fa8125accb3508c9e
[Pipeline] // withDockerContainer
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

Project 2 .. Multistage- MultiAgent - Maven as Backend & node- 16 alpine as Front end

Multi Stage Multi Agent

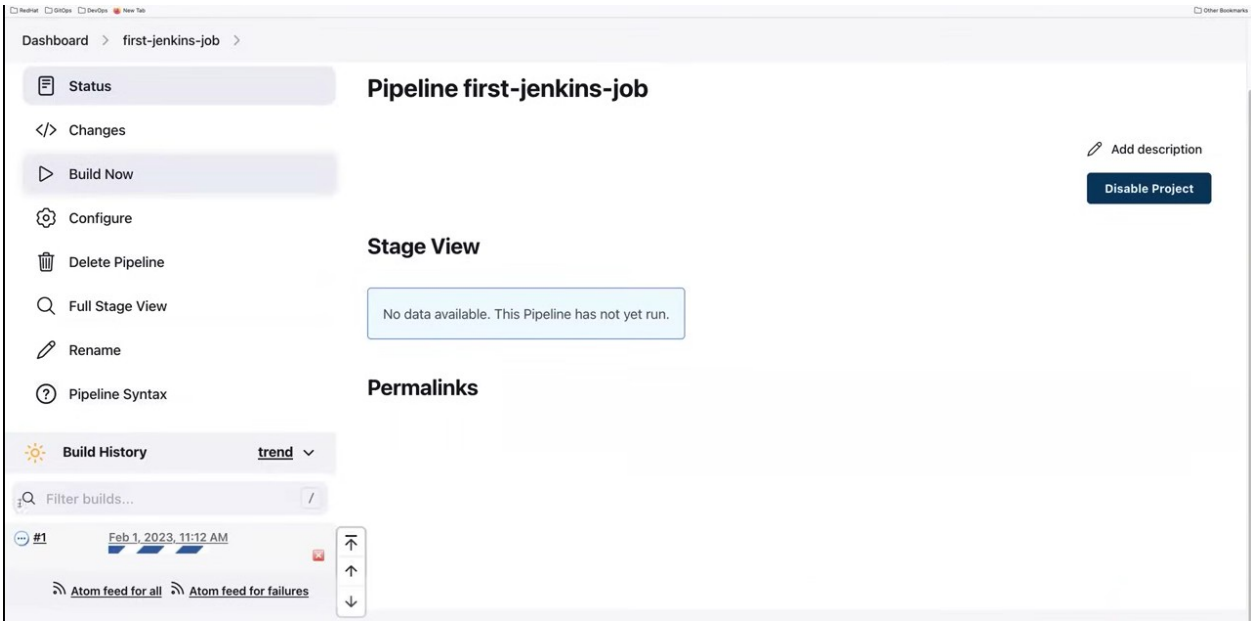
Set up a multi stage jenkins pipeline where each stage is run on a unique agent. This is a very useful approach when you have multi language application or application that has conflicting dependencies.

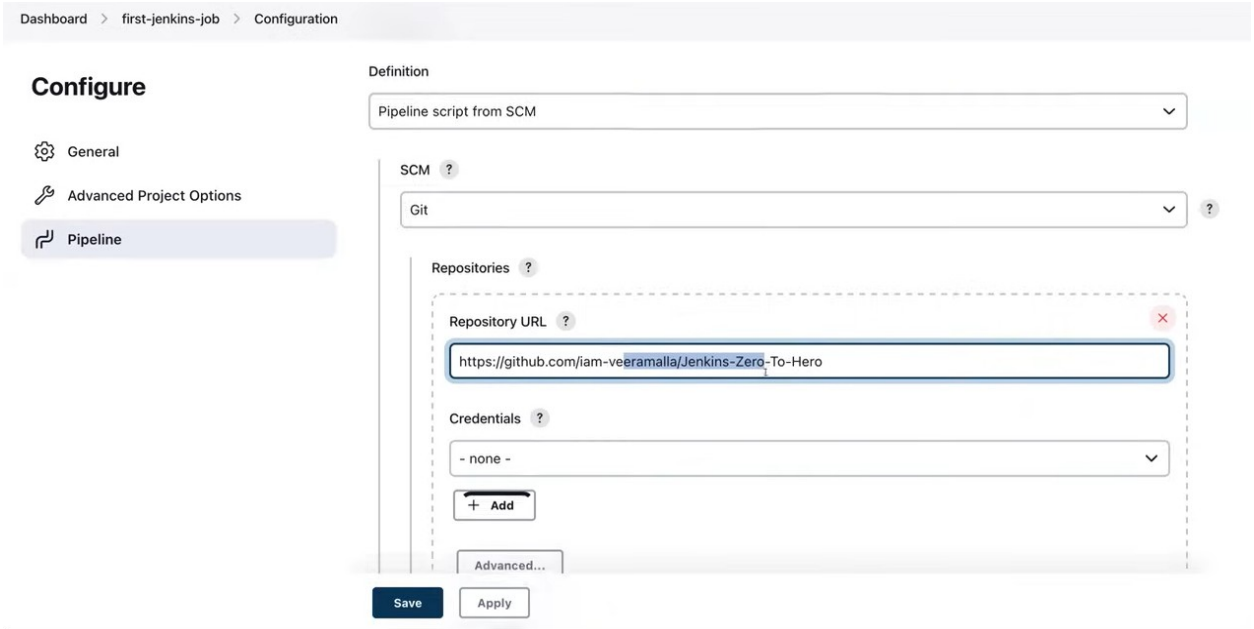
Welcome | Jenkinsfile

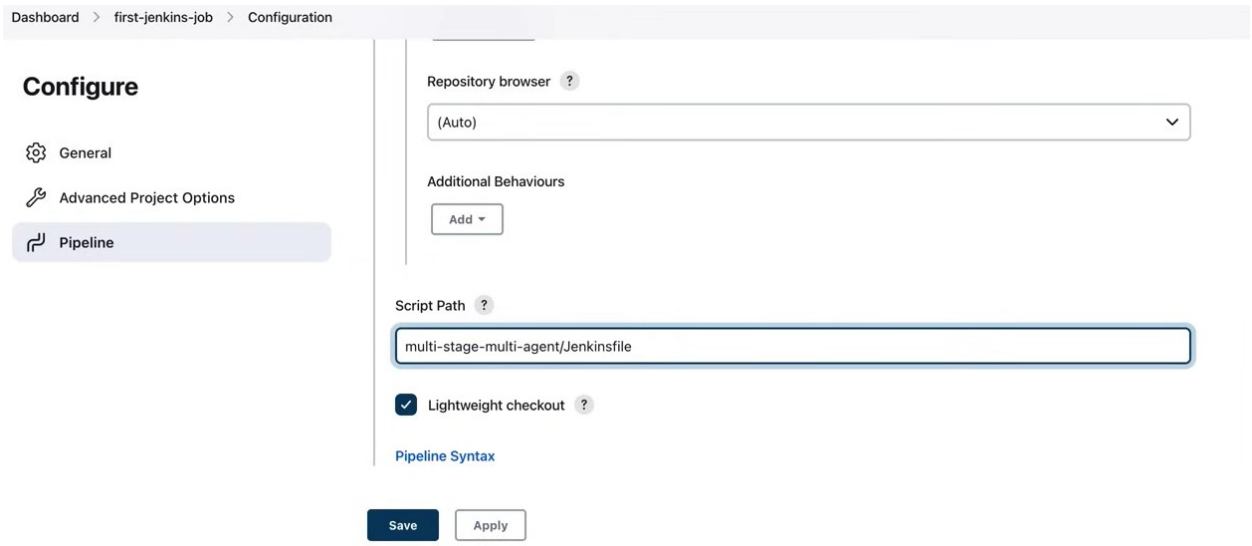
```
pipeline {
  agent none
  stages {
    stage('Back-end') {
      agent {
        docker { image 'maven:3.8.1-adoptopenjdk-11' }
      }
      steps {
        sh 'mvn --version'
      }
    }
    stage('Front-end') {
      agent {
        docker { image 'node:16-alpine' }
      }
      steps {
        sh 'node --version'
      }
    }
```

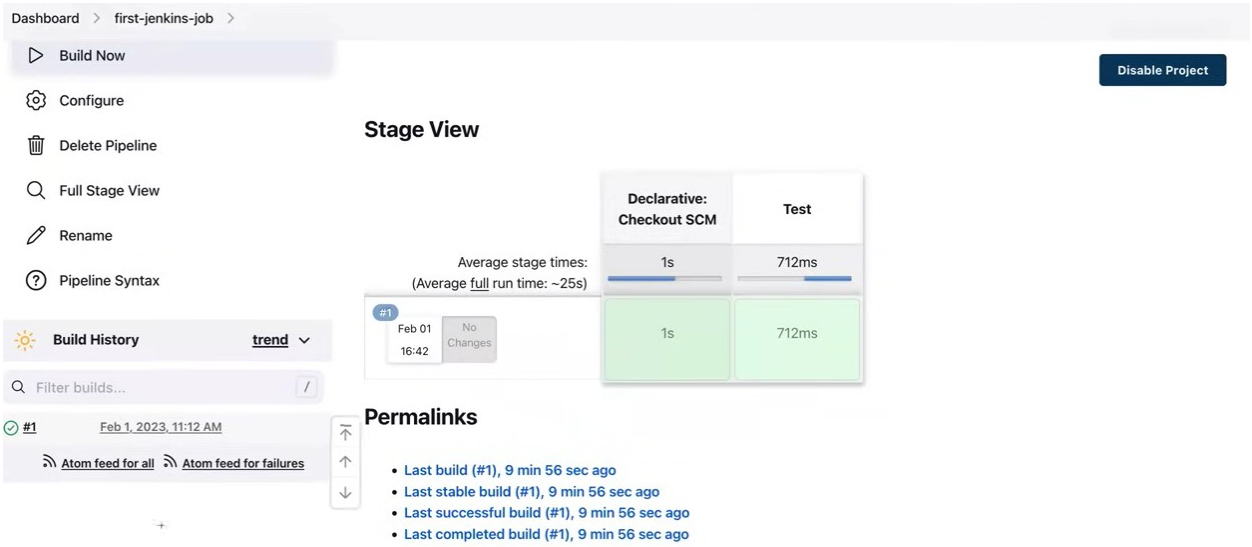
Note .. two Containers will be Created

Click on Configure









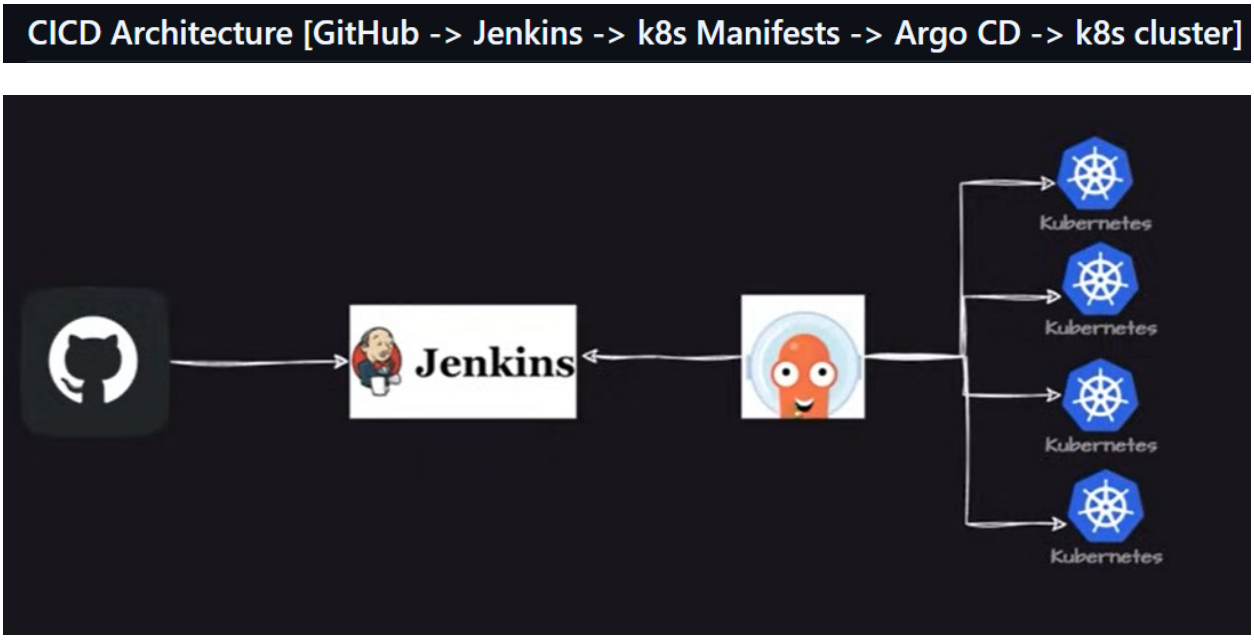

```
CONTAINER ID   IMAGE                          COMMAND                  CREATED          STATUS          PORTS     NAMES
a3c94435e4bf   maven:3.8.1-adoptopenjdk-11    "/usr/local/bin/mvn-…"   4 seconds ago    Up 1 second               inter
esting_chandrasekhar
jenkins@ip-172-31-53-221:~$ docker ps
CONTAINER ID   IMAGE           COMMAND                  CREATED          STATUS                    PORTS     NAMES
7958bbd67307   node:16-alpine  "docker-entrypoint.s…"   2 seconds ago    Up Less than a second               pedantic
_boyd
jenkins@ip-172-31-53-221:~$ docker ps
```

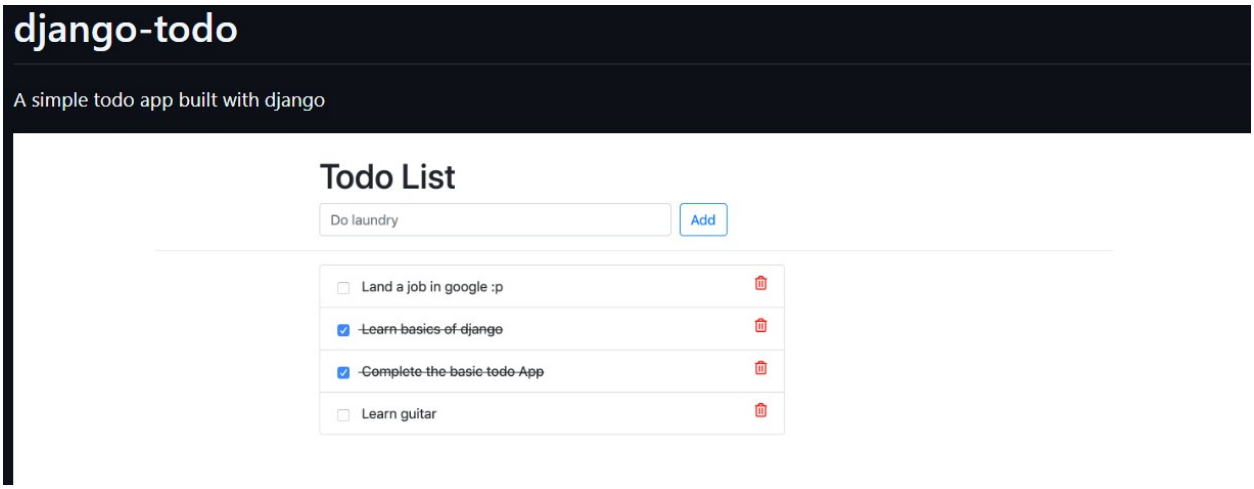
```
jenkins@ip-172-31-53-221:~$ docker ps
CONTAINER ID   IMAGE                          COMMAND                  CREATED         STATUS         PORTS     NAMES
a3c94435e4bf   maven:3.8.1-adoptopenjdk-11    "/usr/local/bin/mvn-…"   4 seconds ago   Up 1 second              inter
esting_chandrasekhar
jenkins@ip-172-31-53-221:~$ docker ps
CONTAINER ID   IMAGE            COMMAND                  CREATED         STATUS                  PORTS     NAMES
7958bbd67307   node:16-alpine   "docker-entrypoint.s…"   2 seconds ago   Up Less than a second             pedantic
_boyd
jenkins@ip-172-31-53-221:~$ docker ps
```

now docker Container are Terminated

```
_boyd
jenkins@ip-172-31-53-221:~$ docker ps
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
jenkins@ip-172-31-53-221:~$
```

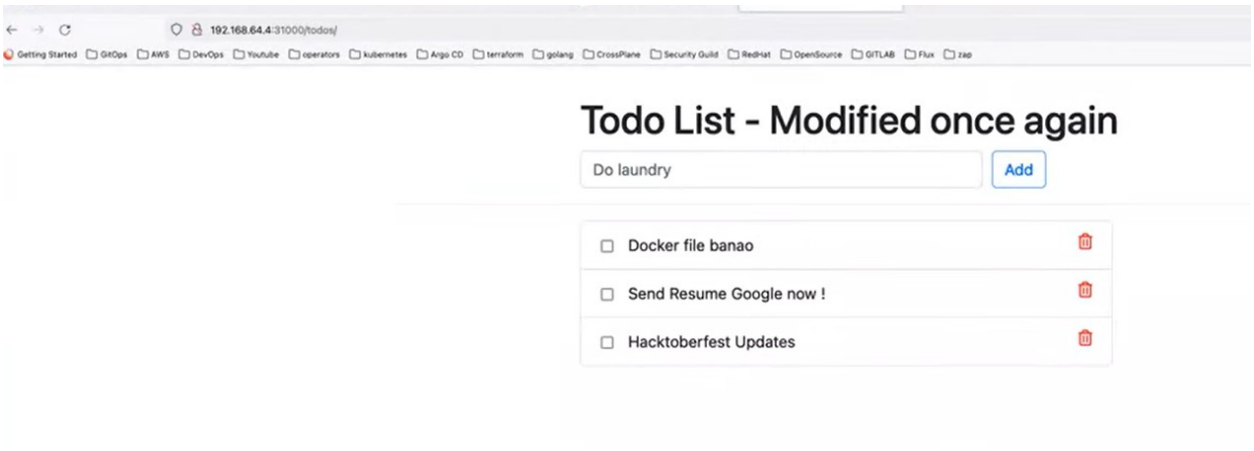
3rd Project – Python Jenkins – argocd-Kubernetes

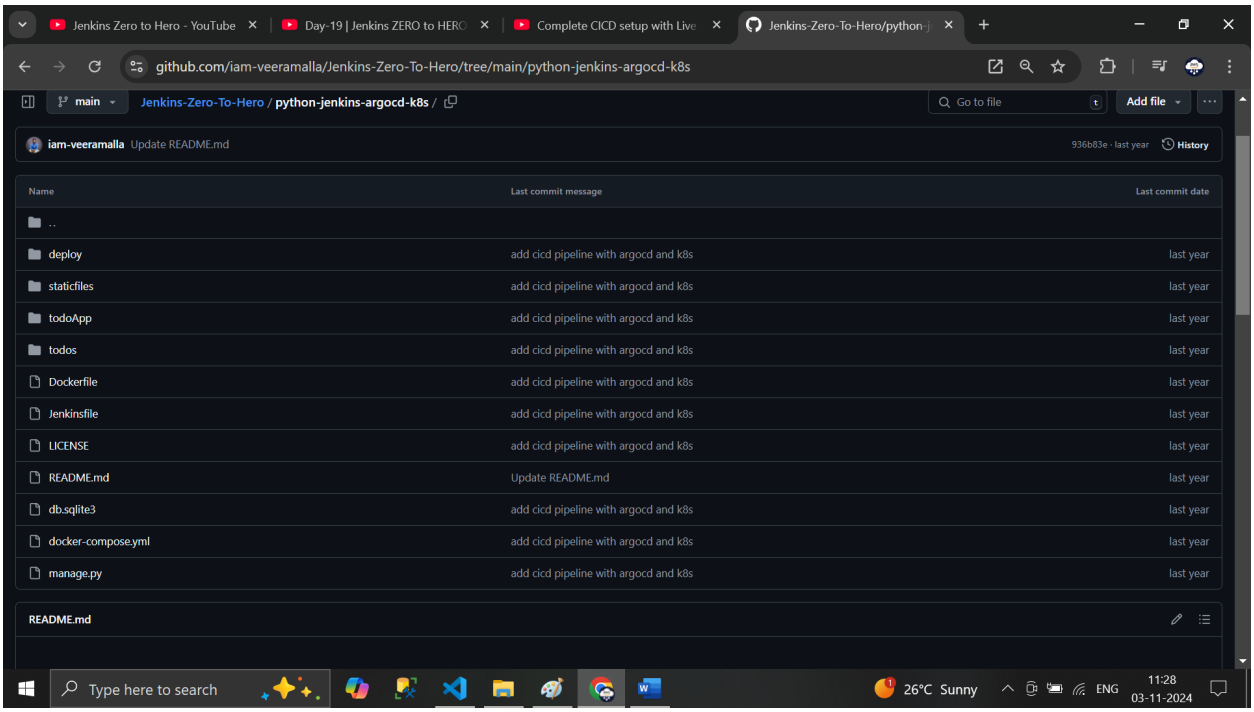


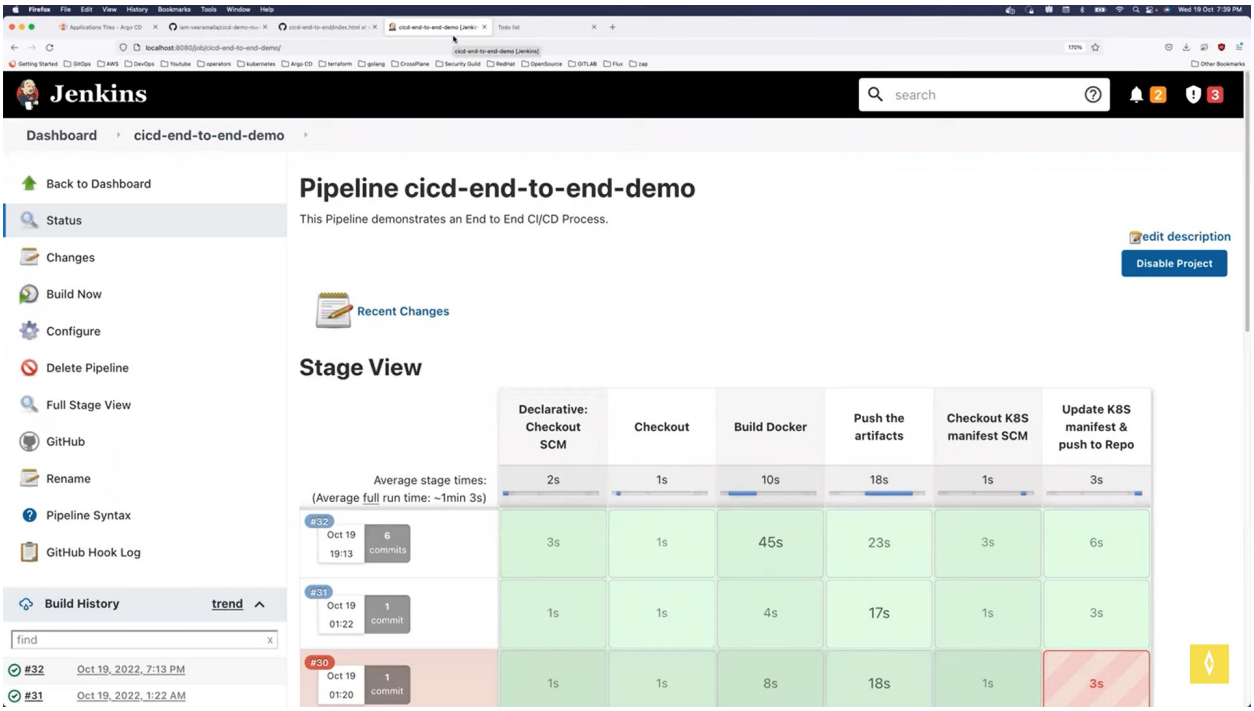



```
→  cicd-end-to-end git:(main) kubectl get svc
NAME                          TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)             AGE
example-argocd-metrics        ClusterIP   10.108.124.138   <none>        8082/TCP            99m
example-argocd-redis          ClusterIP   10.107.179.212   <none>        6379/TCP            99m
example-argocd-repo-server    ClusterIP   10.96.90.219     <none>        8081/TCP,8084/TCP   99m
example-argocd-server         ClusterIP   10.109.202.90    <none>        80/TCP,443/TCP      99m
example-argocd-server-metrics ClusterIP   10.101.192.62    <none>        8083/TCP            99m
kubernetes                    ClusterIP   10.96.0.1        <none>        443/TCP             13d
todo-service                  NodePort    10.111.83.255    <none>        80:31000/TCP        64m
→  cicd-end-to-end git:(main)
```

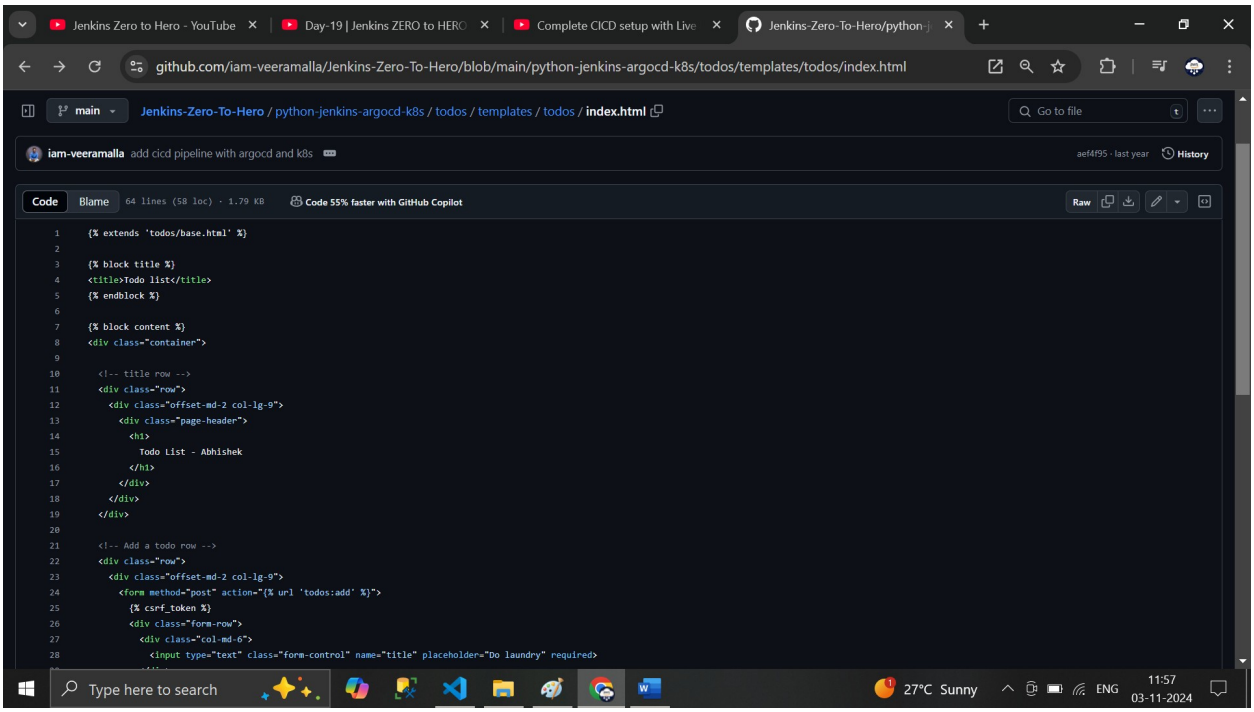
```
→  cicd-end-to-end git:(main) minikube service todo-service
→  cicd-end-to-end git:(main) minikube service list
|-------------|---------------------------------|-------------------|--------------------------|
|  NAMESPACE  |              NAME               |    TARGET PORT    |           URL            |
|-------------|---------------------------------|-------------------|--------------------------|
| argo-rollouts | argo-rollouts-metrics         | No node port      |                          |
| default     | example-argocd-metrics          | No node port      |                          |
| default     | example-argocd-redis            | No node port      |                          |
| default     | example-argocd-repo-server      | No node port      |                          |
| default     | example-argocd-server           | No node port      |                          |
| default     | example-argocd-server-metrics   | No node port      |                          |
| default     | kubernetes                      | No node port      |                          |
| default     | todo-service                    |                80 | http://192.168.64.4:31000 |
| ingress-nginx | ingress-nginx-controller      | http/80           | http://192.168.64.4:31515 |
|             |                                 | https/443         | http://192.168.64.4:31288 |
| ingress-nginx | ingress-nginx-controller-admission | No node port   |                          |
| istio-system | istio-egressgateway            | No node port      |                          |
| istio-system | istio-ingressgateway           | status-port/15021 | http://192.168.64.4:31173 |
|             |                                 | http2/80          | http://192.168.64.4:30257 |
|             |                                 | https/443         | http://192.168.64.4:31342 |
|             |                                 | tcp/31400         | http://192.168.64.4:30280 |
|             |                                 | tls/15443         | http://192.168.64.4:30045 |
| istio-system | istiod                         | No node port      |                          |
| kube-system | kube-dns                        | No node port      |                          |
|-------------|---------------------------------|-------------------|--------------------------|
→  cicd-end-to-end git:(main)
```


192.168.64.4:31000/todos/
Todo List - Modified once again
Do laundry
Add
Docker file banao
Send Resume Google now !
Hacktoberfest Updates

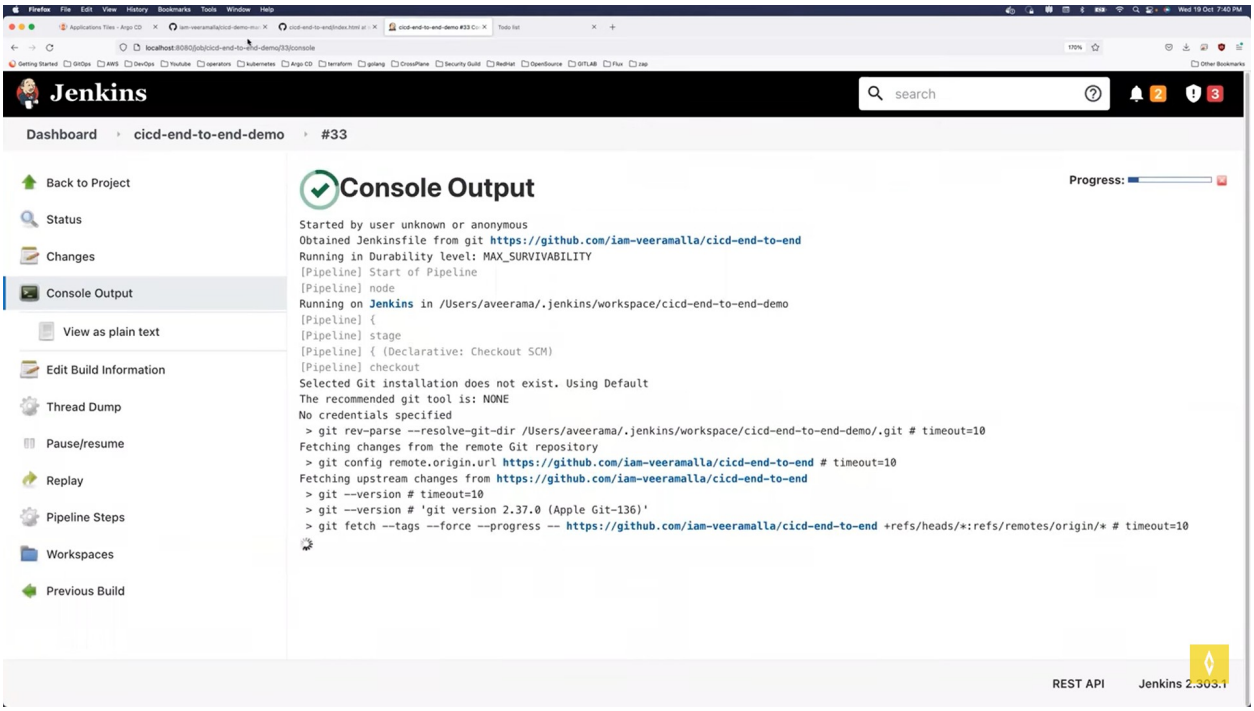

Jenkins Zero to Hero - YouTube
Day-19 | Jenkins ZERO to HERO
Complete CICD setup with Live
Jenkins-Zero-To-Hero/python-j
github.com/iam-veeramalla/Jenkins-Zero-To-Hero/tree/main/python-jenkins-argocd-k8s
main
Jenkins-Zero-To-Hero / python-jenkins-argocd-k8s /
Go to file
Add file
iam-veeramalla Update README.md
936b83e · last year
History
Name
Last commit message
Last commit date
deploy
staticfiles
todoApp
todos
Dockerfile
Jenkinsfile
LICENSE
README.md
Update README.md
db.sqlite3
docker-compose.yml
manage.py
add cicd pipeline with argocd and k8s
last year
README.md
Type here to search
26°C Sunny
ENG
11:28
03-11-2024



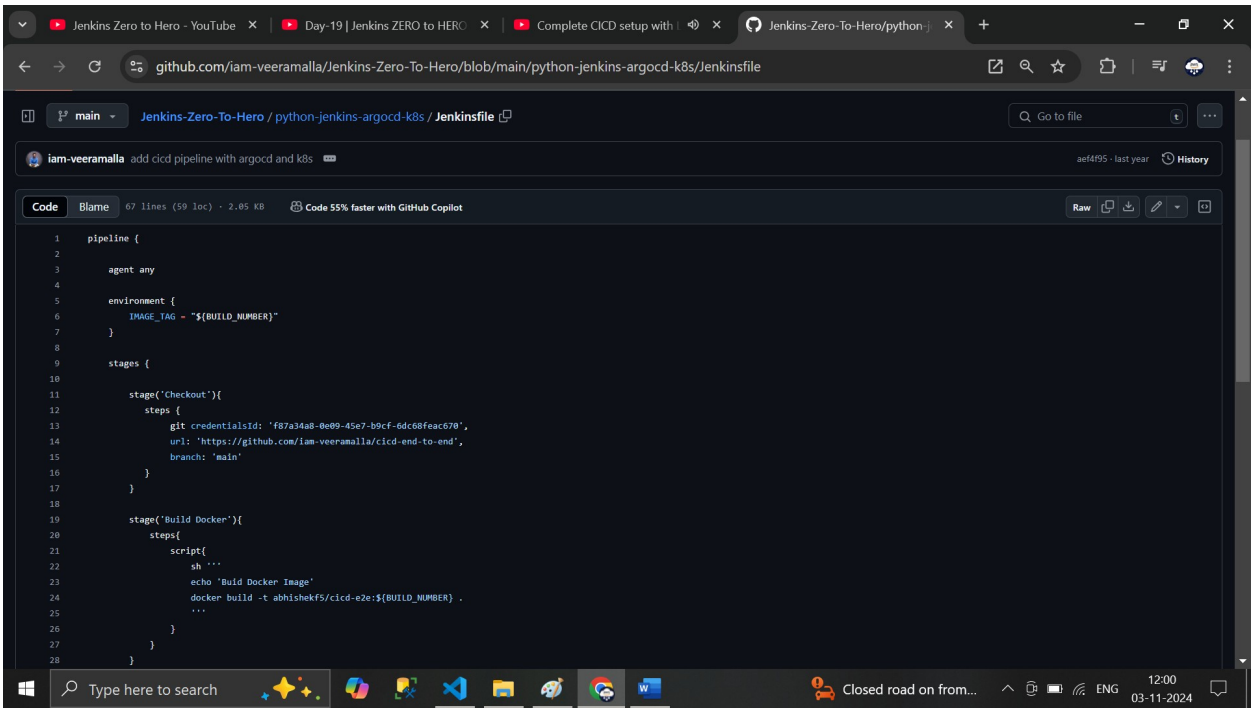
make Change in index,html

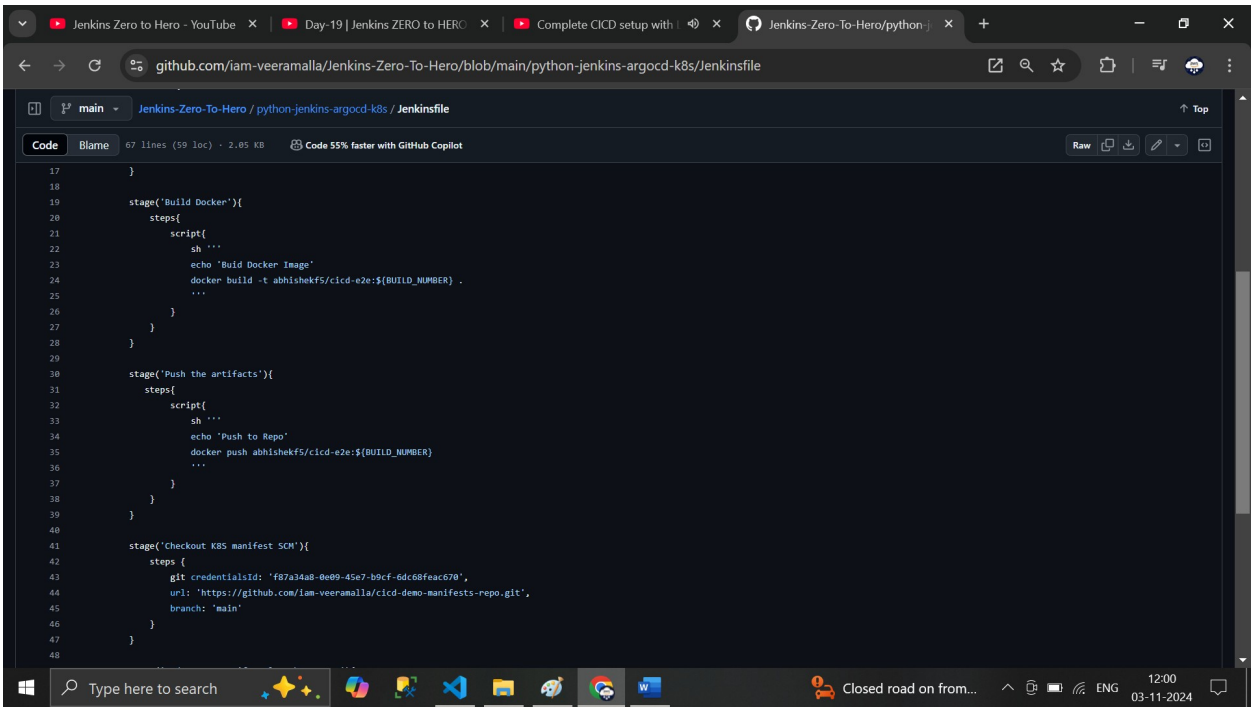


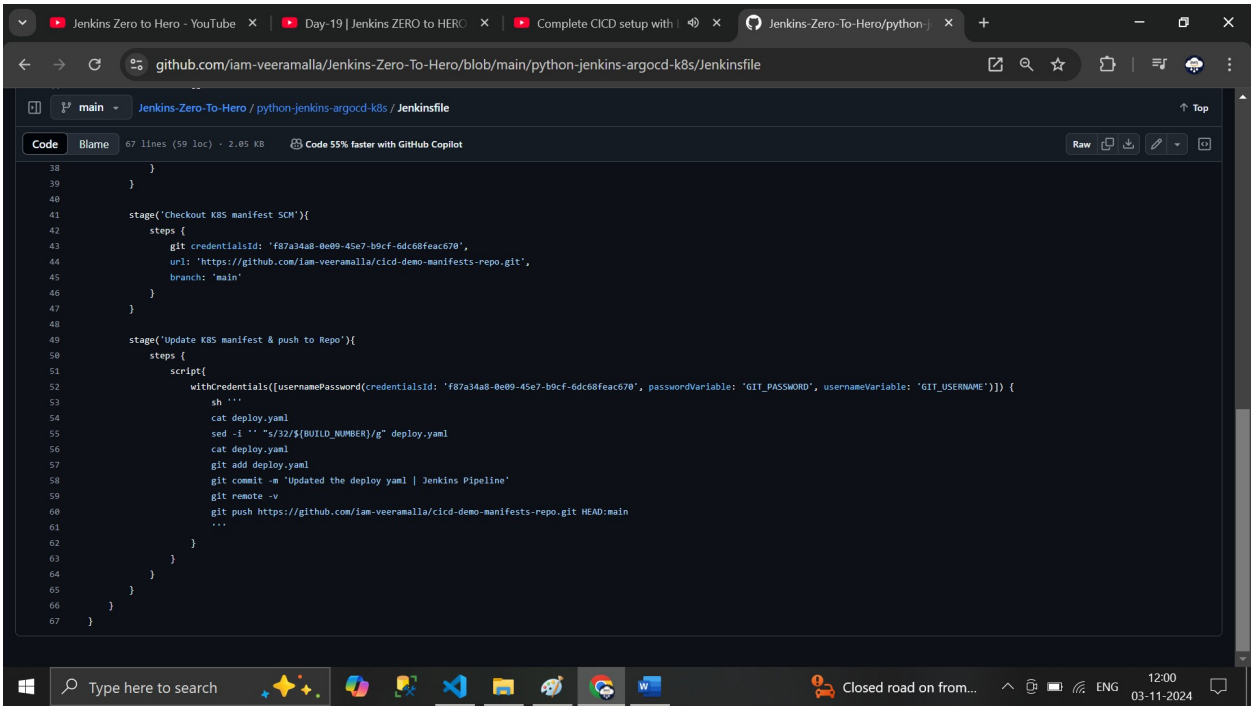
Click on build now in Jenkins



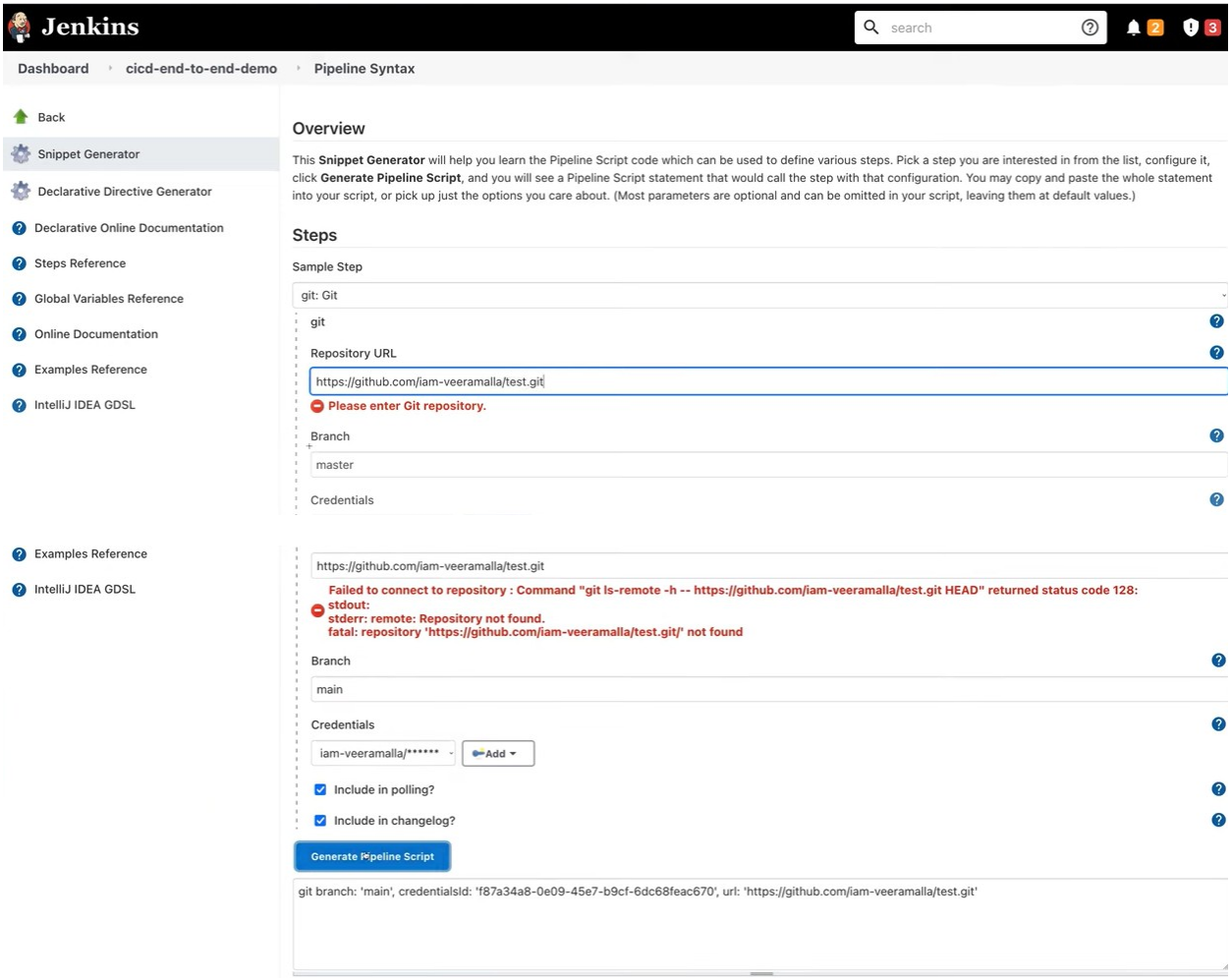
Jenkinsfile

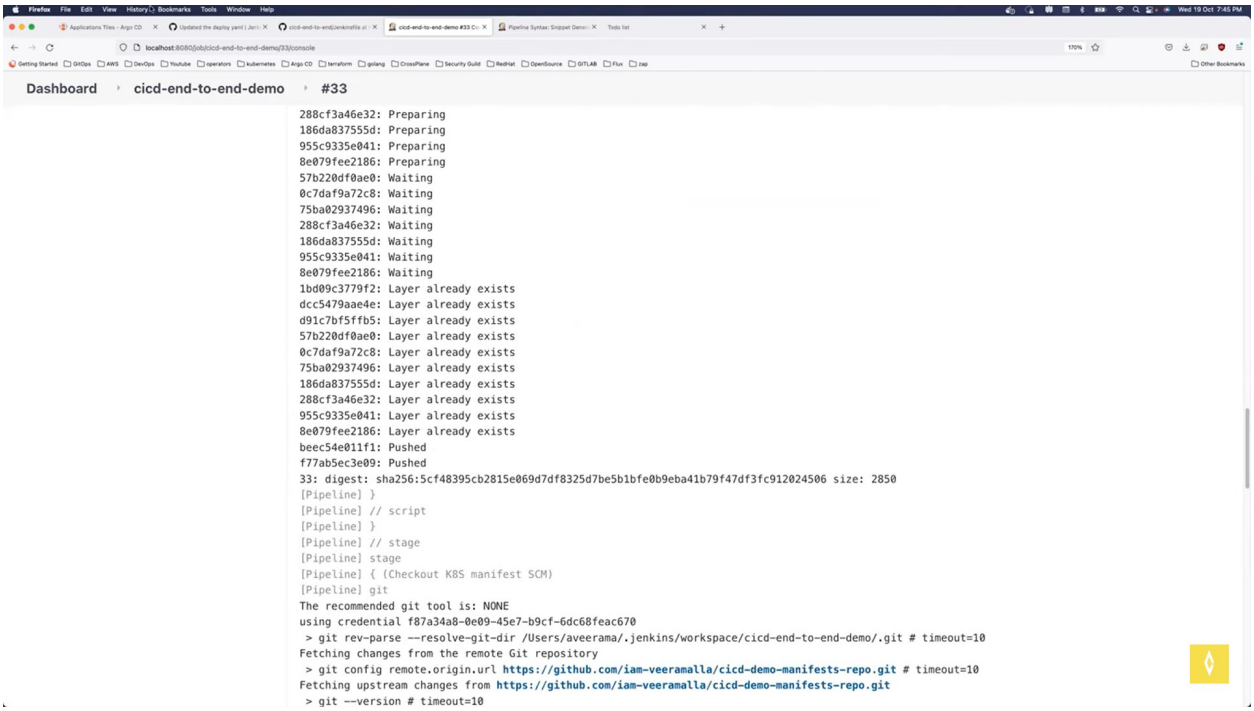


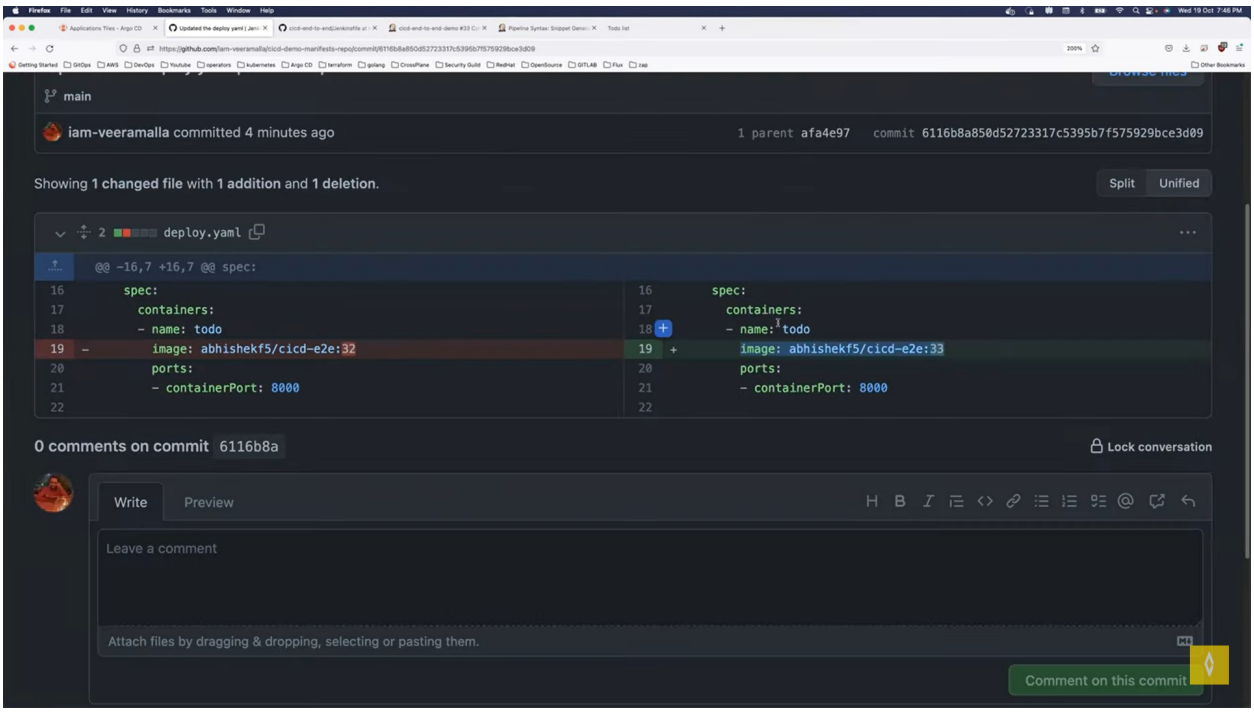


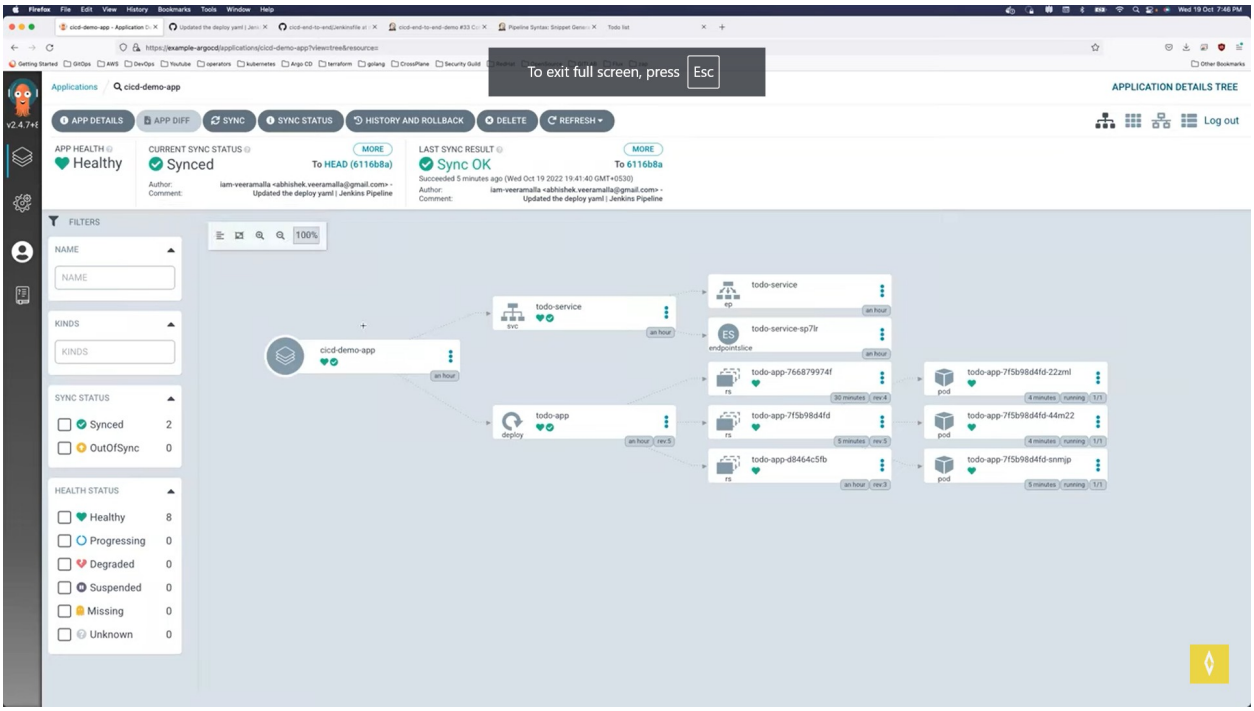


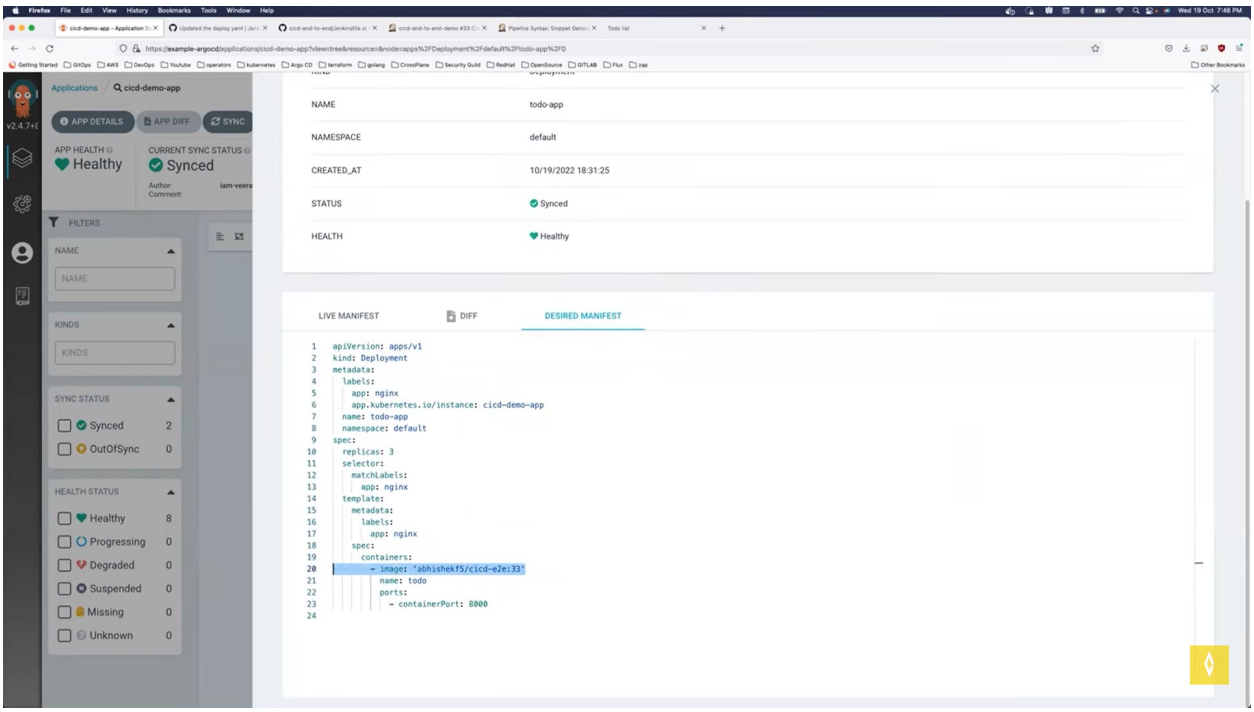
now for Code Generator snippet .. in Jenkins .. it gives You Git ID ...


Jenkins
search
2
3
Dashboard
cicd-end-to-end-demo
Pipeline Syntax
Back
Snippet Generator
Declarative Directive Generator
Declarative Online Documentation
Steps Reference
Global Variables Reference
Online Documentation
Examples Reference
IntelliJ IDEA GDSL
Overview
This Snippet Generator will help you learn the Pipeline Script code which can be used to define various steps. Pick a step you are interested in from the list, configure it, click Generate Pipeline Script, and you will see a Pipeline Script statement that would call the step with that configuration. You may copy and paste the whole statement into your script, or pick up just the options you care about. (Most parameters are optional and can be omitted in your script, leaving them at default values.)
Steps
Sample Step
git: Git
git
Repository URL
https://github.com/iam-veeramalla/test.git
Please enter Git repository.
Branch
master
Credentials
Examples Reference
IntelliJ IDEA GDSL
https://github.com/iam-veeramalla/test.git
Failed to connect to repository : Command "git ls-remote -h -- https://github.com/iam-veeramalla/test.git HEAD" returned status code 128:
stdout:
stderr: remote: Repository not found.
fatal: repository 'https://github.com/iam-veeramalla/test.git/' not found
Branch
main
Credentials
iam-veeramalla/******
Add
Include in polling?
Include in changelog?
Generate Pipeline Script
git branch: 'main', credentialsId: 'f87a34a8-0e09-45e7-b9cf-6dc68feac670', url: 'https://github.com/iam-veeramalla/test.git'


Dashboard
cicd-end-to-end-demo
#33
288cf3a46e32: Preparing
186da837555d: Preparing
955c9335e041: Preparing
8e079fee2186: Preparing
57b220df0ae0: Waiting
0c7daf9a72c8: Waiting
75ba02937496: Waiting
288cf3a46e32: Waiting
186da837555d: Waiting
955c9335e041: Waiting
8e079fee2186: Waiting
1bd09c3779f2: Layer already exists
dcc5479aae4e: Layer already exists
d91c7bf5ffb5: Layer already exists
57b220df0ae0: Layer already exists
0c7daf9a72c8: Layer already exists
75ba02937496: Layer already exists
186da837555d: Layer already exists
288cf3a46e32: Layer already exists
955c9335e041: Layer already exists
8e079fee2186: Layer already exists
beec54e011f1: Pushed
f77ab5ec3e09: Pushed
33: digest: sha256:5cf48395cb2815e069d7df8325d7be5b1bfe0b9eba41b79f47df3fc912024506 size: 2850
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Checkout K8S manifest SCM)
[Pipeline] git
The recommended git tool is: NONE
using credential f87a34a8-0e09-45e7-b9cf-6dc68feac670
> git rev-parse --resolve-git-dir /Users/aveerama/.jenkins/workspace/cicd-end-to-end-demo/.git # timeout=10
Fetching changes from the remote Git repository
> git config remote.origin.url https://github.com/iam-veeramalla/cicd-demo-manifests-repo.git # timeout=10
Fetching upstream changes from https://github.com/iam-veeramalla/cicd-demo-manifests-repo.git
> git --version # timeout=10


main
iam-veeramalla committed 4 minutes ago
1 parent afa4e97
commit 6116b8a850d52723317c5395b7f575929bce3d09
Showing 1 changed file with 1 addition and 1 deletion.
Split
Unified
2
deploy.yaml
@@ -16,7 +16,7 @@ spec:
spec:
containers:
- name: todo
- image: abhishekf5/cicd-e2e:32
+ image: abhishekf5/cicd-e2e:33
ports:
- containerPort: 8000
0 comments on commit 6116b8a
Lock conversation
Write
Preview
Leave a comment
Attach files by dragging & dropping, selecting or pasting them.
Comment on this commit


To exit full screen, press Esc
Applications
cicd-demo-app
APPLICATION DETAILS TREE
APP DETAILS
APP DIFF
SYNC
SYNC STATUS
HISTORY AND ROLLBACK
DELETE
REFRESH
Log out
APP HEALTH
Healthy
CURRENT SYNC STATUS
Synced
To HEAD (6116b8a)
LAST SYNC RESULT
Sync OK
To 6116b8a
FILTERS
NAME
KINDS
SYNC STATUS
Synced 2
OutOfSync 0
HEALTH STATUS
Healthy 8
Progressing 0
Degraded 0
Suspended 0
Missing 0
Unknown 0
cicd-demo-app
todo-service
todo-service-sp7lr
todo-app
todo-app-766879974f
todo-app-7f5b98d4fd
todo-app-d8464c5fb
todo-app-7f5b98d4fd-22zml
todo-app-7f5b98d4fd-44m22
todo-app-7f5b98d4fd-snmjp


NAME todo-app
NAMESPACE default
CREATED_AT 10/19/2022 18:31:25
STATUS Synced
HEALTH Healthy
LIVE MANIFEST
DIFF
DESIRED MANIFEST
apiVersion: apps/v1
kind: Deployment
metadata:
labels:
app: nginx
app.kubernetes.io/instance: cicd-demo-app
name: todo-app
namespace: default
spec:
replicas: 3
selector:
matchLabels:
app: nginx
template:
metadata:
labels:
app: nginx
spec:
containers:
- image: 'abhishekf5/cicd-e2e:33'
name: todo
ports:
- containerPort: 8000

```
→  cicd-end-to-end git:(main) kubectl get deploy
NAME                          READY   UP-TO-DATE   AVAILABLE   AGE
example-argocd-redis          1/1     1            1           110m
example-argocd-repo-server    1/1     1            1           110m
example-argocd-server         1/1     1            1           110m
todo-app                      3/3     3            3           75m
→  cicd-end-to-end git:(main) kubectl edit deploy todo-app
```

#img tag updated