DevOps Projects

(CI/CD Pipeline)

DevOps Project-1: DevOps Project for Configuring Jenkins server and Installing Tomcat onto Jenkin's Server for Deploying our Application.

Project-1 Objective: The objective of this Project is to configure Jenkins to build and deploy applications. It includes Setting up Jenkins, installing necessary plugins and configuring Jenkins to build Maven projects, and `Installing Tomcat Server for viewing the Web Page.'

DevOps Project-2: DevOps Project for Configuring Docker Machine as Jenkins Slave, build and deploy code in Docker Host as a container.

Project-2 Objective: In this Project, you will set up a Docker container as a Jenkins slave, and build a Docker image for a Java web application, and Deploy it in a Docker container.

DevOps Project-1

DevOps project for Configuring Jenkins server and Installing Tomcat onto Jenkin's Server for Deploying our Application.

Project Objective: The objective of this Project is to configure Jenkins to build and deploy applications. It includes Setting up Jenkins, installing necessary plugins and configuring Jenkins to build Maven projects, and `Installing Tomcat Server for viewing the Web Page.'

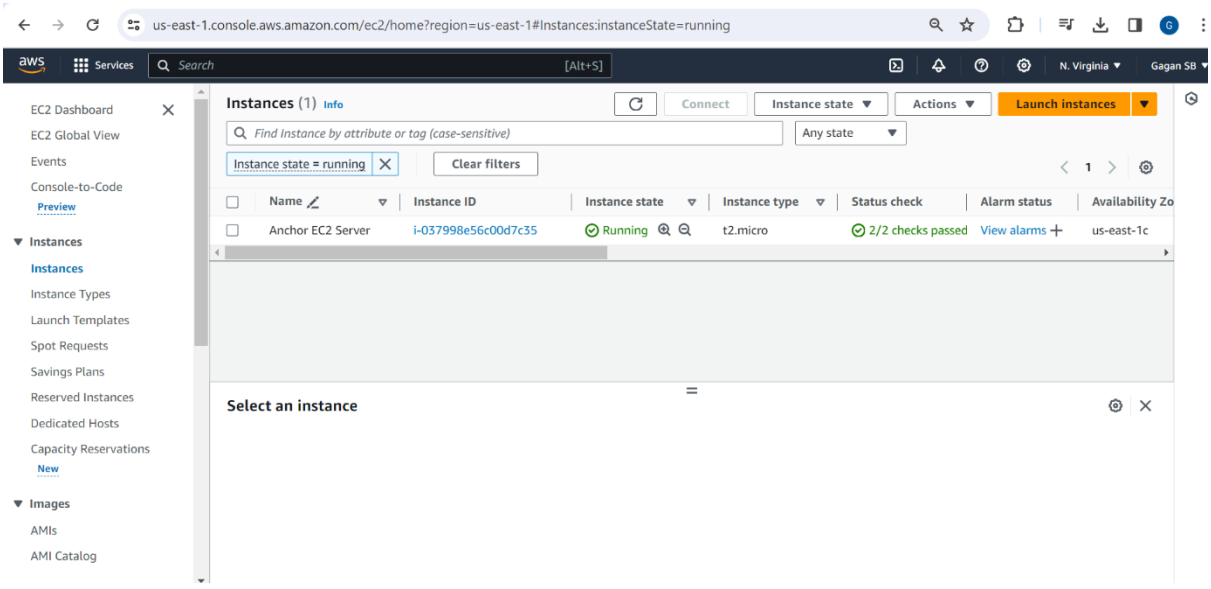
DevOps Tools used for the Project:

1. AWS Cloud Platform to create Anchor Server
2. Git/ Github
3. Terraform
4. Ansible
5. Jenkins
6. Apache Maven
7. Tomcat Server – To Deploy our Application

Other Tool Used:

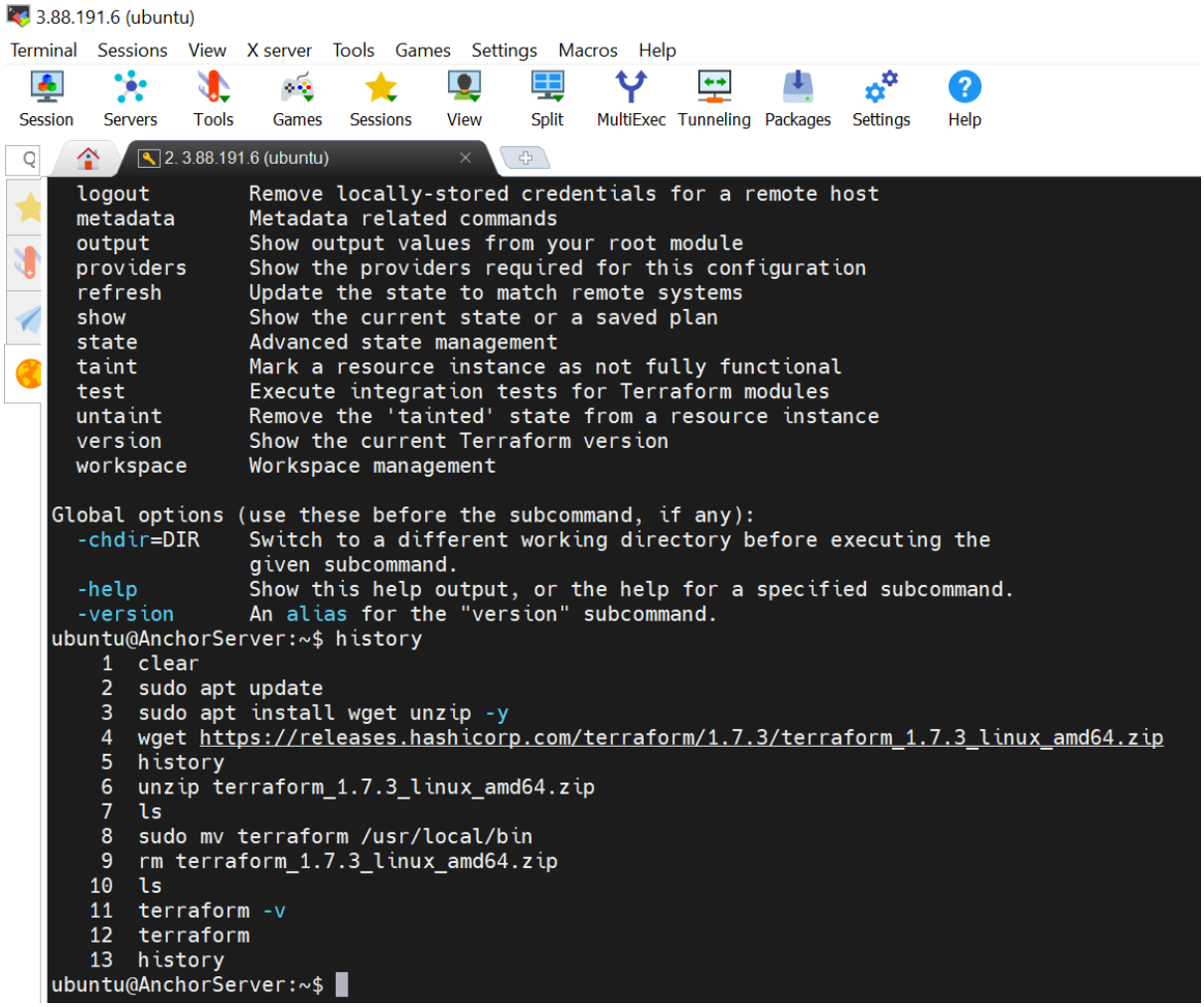
1.MobaXterm - Toolbox for remote computing in order to connect to AWS

Step1: In order to perform this project, The first step is to Manually Launch an Anchor EC2 Server on AWS.



Step2: Installing Terraform onto Anchor Server to automate the creation of 2 more EC2 instances. With the help of an Infrastructure as Code (IaC) tool called Terraform, which automates the provisioning of infrastructure, we have created two more servers for our project.

For our project, we use the MobaXterm Toolbox for remote computing in order to connect to AWS.



The above commands show the installation of Terraform onto Anchor Server

Step3: Install Python 3, pip, AWS CLI, and Ansible on to Anchor Server

Installing Ansible onto Anchor Server to automate the configuration management, application deployment, task automation, software installation and also used as Infrastructure as Code (IaC) tool.

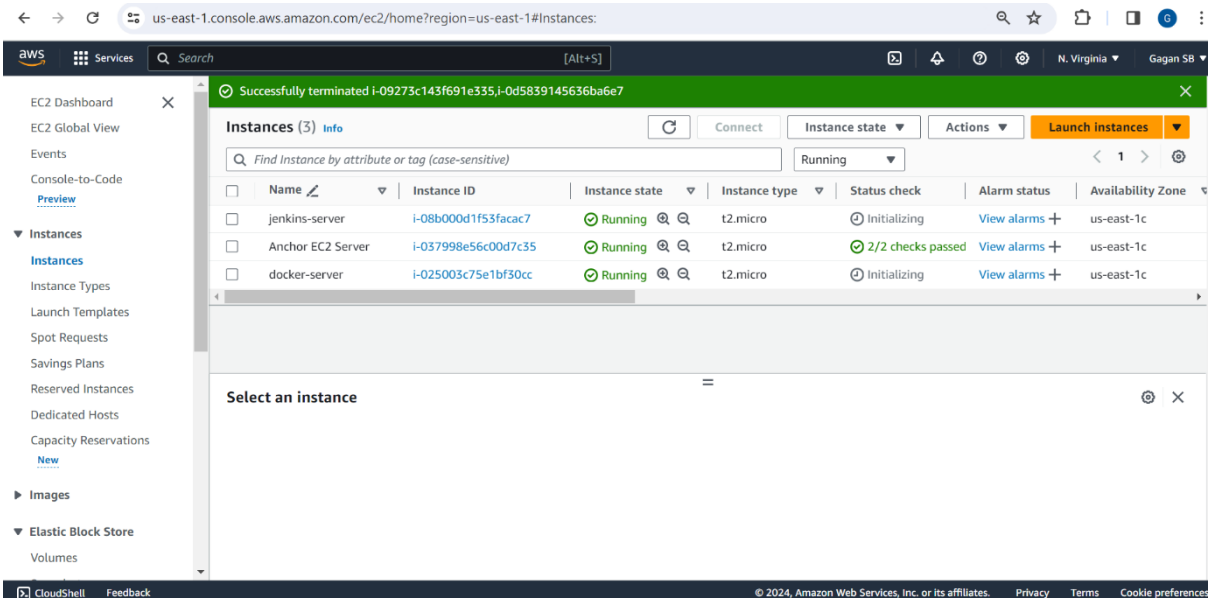
With Ansible, we will install Python as a dependency because Ansible is written using Python and Pip as a package manager for Python. With it, we will install AWS CLI, which is used as a utility to connect our MobaXterm tool to an EC2 instance.

```
   14  sudo apt-get update
   15  sudo apt-get install python3-pip -y
   16  sudo pip3 install awscli boto boto3 ansible
   17  aws configure
   18  aws ls s3
   19  aws iam list-users
   20  sudo mkdir /etc/ansible && sudo touch /etc/ansible/hosts
   21  history
ubuntu@AnchorServer:~$
```

Step4: Utilizing Terraform, initiate the deployment of two new servers: Docker-server and Jenkins-server

One server will be used for Jenkins called as jenkins-server.

Another server will be used for Docker called as docker-server.



```
   61  ssh-keygen -t rsa -b 2048
   62  mkdir devops-labs && cd devops-labs
   63  vi DevOpsServers.tf
   64  vi variables.tf
   65  terraform init
   66  terraform fmt
   67  terraform validate
   68  terraform plan
   69  terraform apply -auto-approve
   70  history
ubuntu@AnchorServer:~/devops-labs$
```

Commands for Terraform to create two more EC2 servers (Jenkins and Docker Servers), where the creation of resource codes is written in the Terraform Configuration file called DevOpsServers.tf in the HCL (HashiCorp Configuration Language**).** This creation of resources is automated with the help of Terraform.

1. **vi DevOpsServers.tf (HCL Code)**

```
provider "aws" {
  region = var.region
}


resource "aws_key_pair" "mykeypair" {
  key_name   = var.key_name
  public_key = file(var.public_key)
}


# to create 2 EC2 instances
resource "aws_instance" "my-machine" {
  # Launch 2 servers
  for_each = toset(var.my-servers)


  ami                  = var.ami_id
  key_name             = var.key_name
```

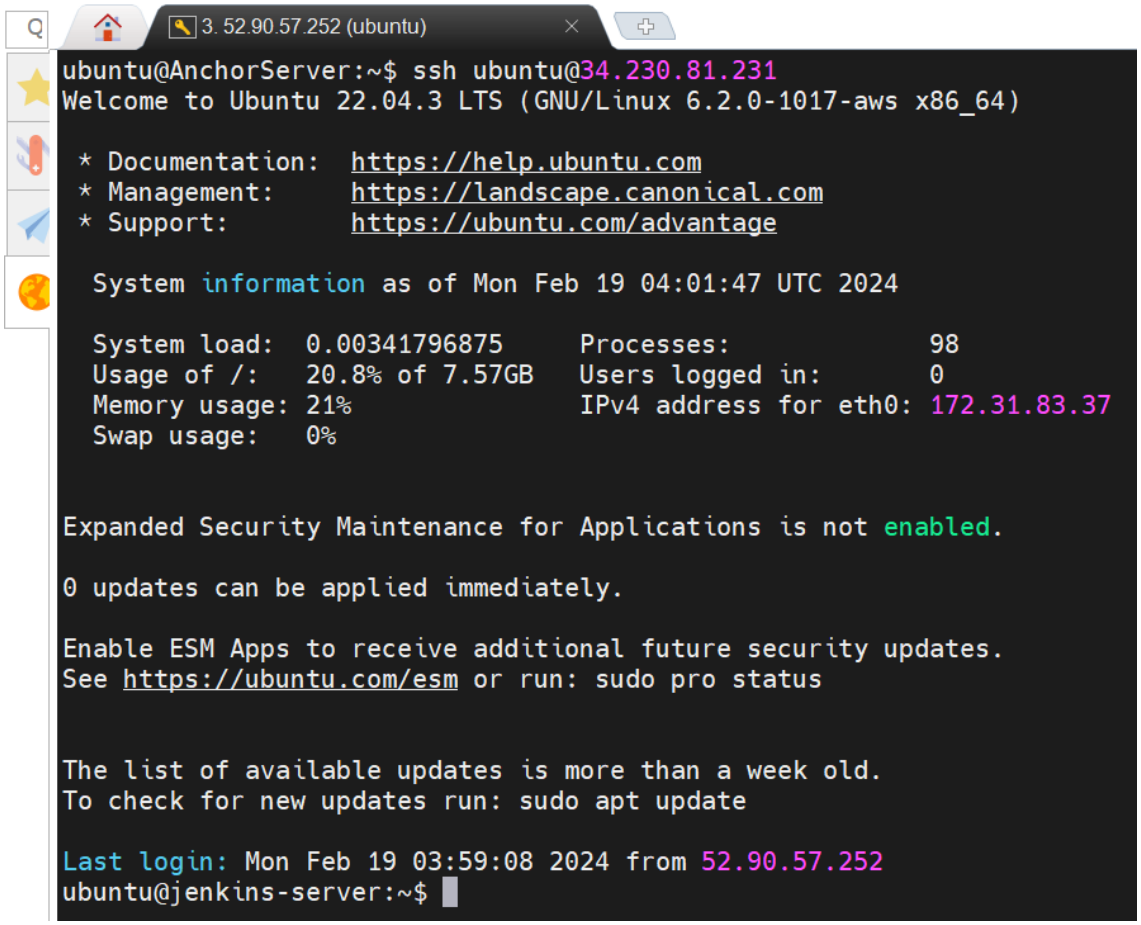
```
  vpc_security_group_ids = [var.sg_id]
  instance_type          = var.ins_type

  # Read from the list my-servers to name each server
  tags = {
    Name = each.key
  }

  provisioner "local-exec" {
    command = <<-EOT
      echo [${each.key}] >> /etc/ansible/hosts
      echo ${self.public_ip} >> /etc/ansible/hosts
    EOT
  }
}
```

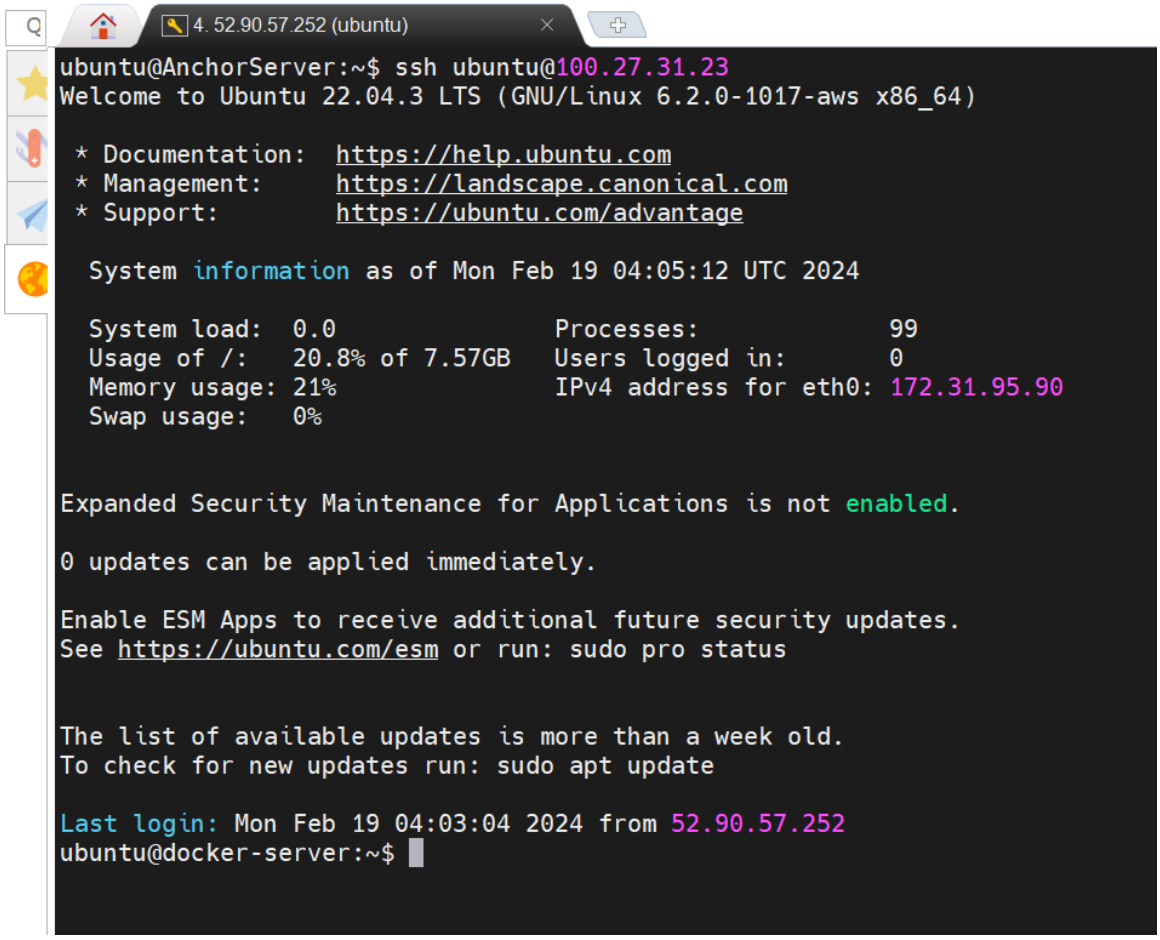
2. **vi variables.tf**

```
variable "region" {
   default = "us-east-1"
}

# Change the SG ID. You can use the same SG ID used for our Anchor server
# Basically the SG should open ports 22, 80, 8080, 9999, and 4243
variable  "sg_id" {
   default = "sg-0f0b72798c3cd671d" # us-east-1
}

# Choose a free tier Ubuntu AMI. You can use below.
variable "ami_id" {
   default = " ami-0c7217cdde317cfec" # us-east-1; Ubuntu
}

# We are only using t2.micro for this lab
variable "ins_type" {
   default = "t2.micro"
}
```

```
# Replace 'yourname' with your first name
variable key_name {
   default = "Gagan-Jenkins-Docker-KeyPair"
}

variable public_key {
   default = "/home/ubuntu/.ssh/id_rsa.pub"   #Ubuntu OS
}

variable "my-servers" {
  type    = list(string)
  default = ["jenkins-server", "docker-server"]
}
```

Step5: From Anchor Server SSH into Jenkins-Server and from Anchor Server SSH into Docker-Server and check they are accessible





Step6: Use Ansible to deploy respective packages onto each of the servers. In our project, Ansible can be used to install Jenkins on the Jenkins server and Docker on the Docker server.

Here we download the Jenkins and Docker installation package code of the Ansible Playbook file, which are written in the YAML scripting language.

1.DevOpsSetup.yml

```
---
- name: Start installing Jenkins pre-requisites before installing Jenkins
  hosts: jenkins-server
  become: yes
  become_method: sudo
  gather_facts: no
  tasks:


  - name: Update apt repository with latest packages
    apt:
```

```
    update_cache: yes
    upgrade: yes

- name: Installing jdk11 in Jenkins server
  apt:
    name: openjdk-11-jdk
    update_cache: yes
  become: yes

- name: Installing jenkins apt repository key
  apt_key:
    url: https://pkg.jenkins.io/debian/jenkins.io-2023.key
    state: present
  become: yes

- name: Configuring the apt repository
  apt_repository:
    repo: deb https://pkg.jenkins.io/debian binary/
    filename: /etc/apt/sources.list.d/jenkins.list
    state: present
  become: yes

- name: Update apt-get repository with "apt-get update"
  apt:
    update_cache: yes

- name: Finally, its time to install Jenkins
  apt: name=jenkins update_cache=yes
  become: yes

- name: Jenkins is installed. Lets start 'Jenkins' now!
  service: name=jenkins state=started
```

```
  - name: Wait until the file /var/lib/jenkins/secrets/initialAdminPassword is present before continuing
    wait_for:
      path: /var/lib/jenkins/secrets/initialAdminPassword
  - name: You can find Jenkins admin password under 'debug'
    command: cat /var/lib/jenkins/secrets/initialAdminPassword
    register: out
  - debug: var=out.stdout_lines



- name: Start the Docker installation steps
  hosts: docker-server
  become: yes
  become_method: sudo
  gather_facts: no

  tasks:

  - name: Update 'apt' repository with latest versions of packages
    apt:
      update_cache: yes

  - name: install docker prerequisite packages
    apt:
      name: ['ca-certificates', 'curl', 'gnupg', 'lsb-release']
      update_cache: yes
      state: latest

  - name: Install the docker apt repository key
    apt_key: url=https://download.docker.com/linux/ubuntu/gpg state=present
    become: yes

  - name: Configure the apt repository
    apt_repository:
      repo: deb https://download.docker.com/linux/ubuntu bionic stable
```

```
    state: present
  become: yes

- name: Update 'apt' repository
  apt:
    update_cache: yes

- name: Install Docker packages
  apt:
    name: ['docker-ce', 'docker-ce-cli', 'containerd.io']
    update_cache: yes
  become: yes

- name: Install jdk11 in Docker server. Maven needs this.
  apt:
    name: openjdk-11-jre-headless
    update_cache: yes
  become: yes

- name: Start Docker service
  service:
    name: docker
    state: started
    enabled: yes



- lineinfile:
     dest: /lib/systemd/system/docker.service
     regexp: '^ExecStart='
     line: 'ExecStart=/usr/bin/dockerd -H tcp://0.0.0.0:4243 -H unix:///var/run/docker.sock'



- name: Reload systemd
  command: systemctl daemon-reload
```

- name: docker restart

  service:

    name: docker

    state: restarted

...

```
   86  mkdir ansible && cd ansible
   87  wget https://devops-e-e.s3.ap-south-1.amazonaws.com/DevOpsSetup.yml
   88  cat DevOpsSetup.yml
   89  ansible-playbook DevOpsSetup.yml
   90  history
ubuntu@AnchorServer:~/ansible$
```

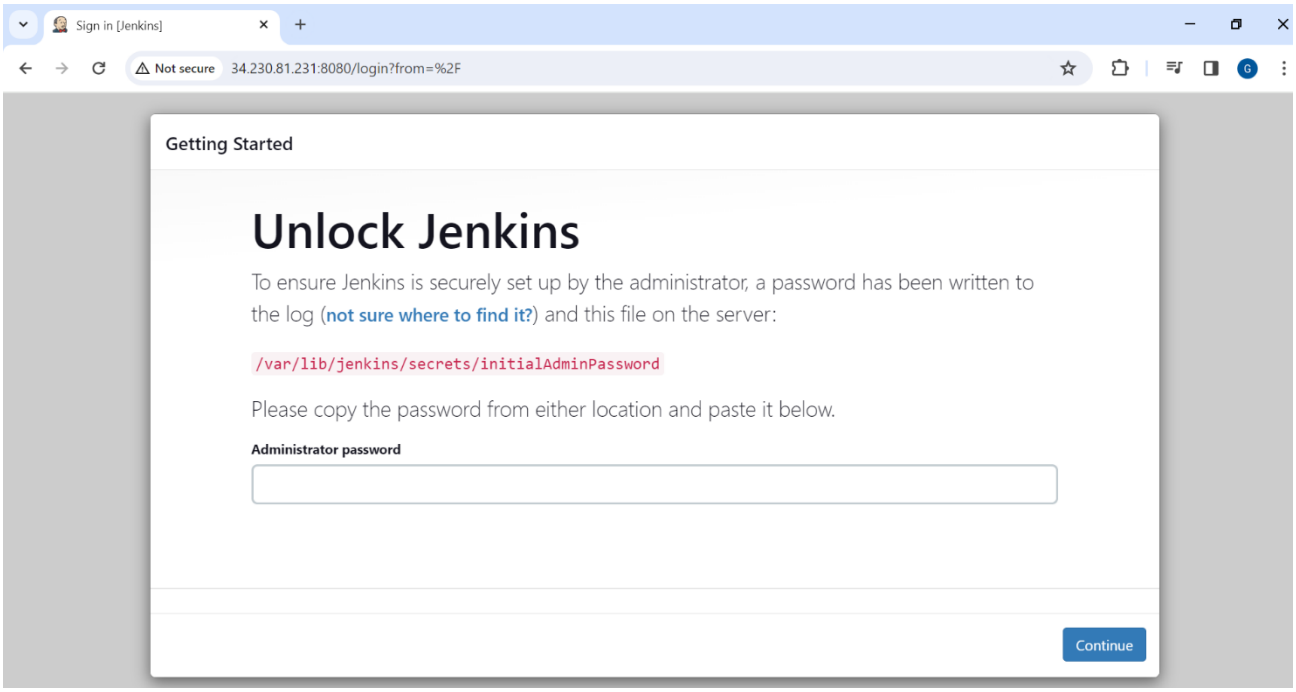
Commands to install and run the jenkins and docker packages

Step7: After installing Jenkins and Docker packages on the Jenkins and Docker servers through Ansible, verify the Jenkins landing page and the Docker landing page.

(Jenkins port no-8080 | docker port no-4243)

1. Verify the Jenkins Landing Page:

- Open a web browser and navigate to our Jenkins landing page using your Jenkins server's public IP address followed by Jenkins port number 8080



2. Verify the Docker Landing Page:

- Open a web browser and access the Docker landing page using your Docker server's public IP address followed by docker port number 4243

Step8: Git and GitHub Operations: This Step focuses on Git and GitHub operations, including initializing a Git repository, making commits, creating branches, and pushing code to a GitHub repository. Here we download the Java code that we are going to use in the CICD pipeline from the browser for the web application.

Task-1: Initializing the local git repository and committing changes

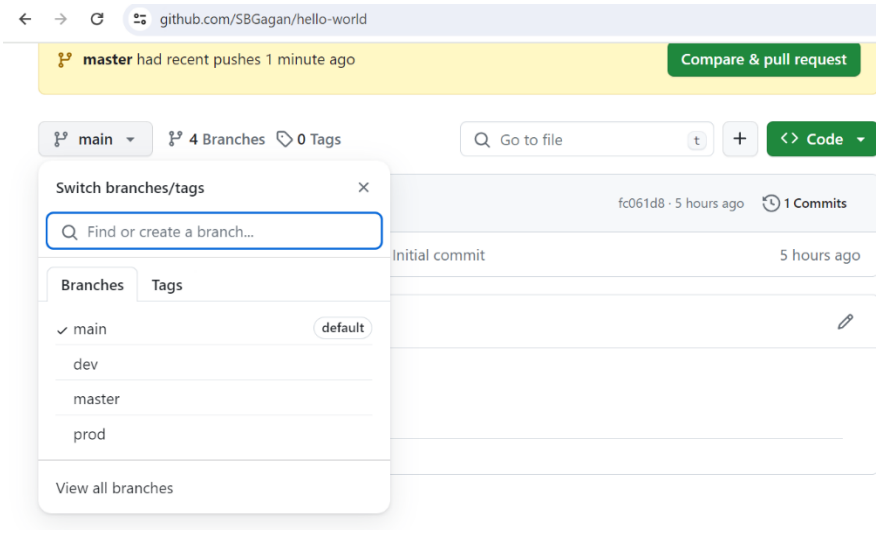
```
   91  cd ~
   92  git --version
   93  wget https://devops-e-e.s3.ap-south-1.amazonaws.com/hello-world-master.zip
   94  unzip hello-world-master.zip
   95  rm hello-world-master.zip
   96  cd hello-world-master
   97  ls
   98  git init .
   99  git config --global user.email "gagansb37@gmai.com"
  100  git config --global user.name "GaganSB"
  101  git status
  102  git add .
  103  git log
  104  git status
  105  git commit -m "This is the first commit"
  106  git log
  107  git status
  108  history
ubuntu@AnchorServer:~/hello-world-master$
```

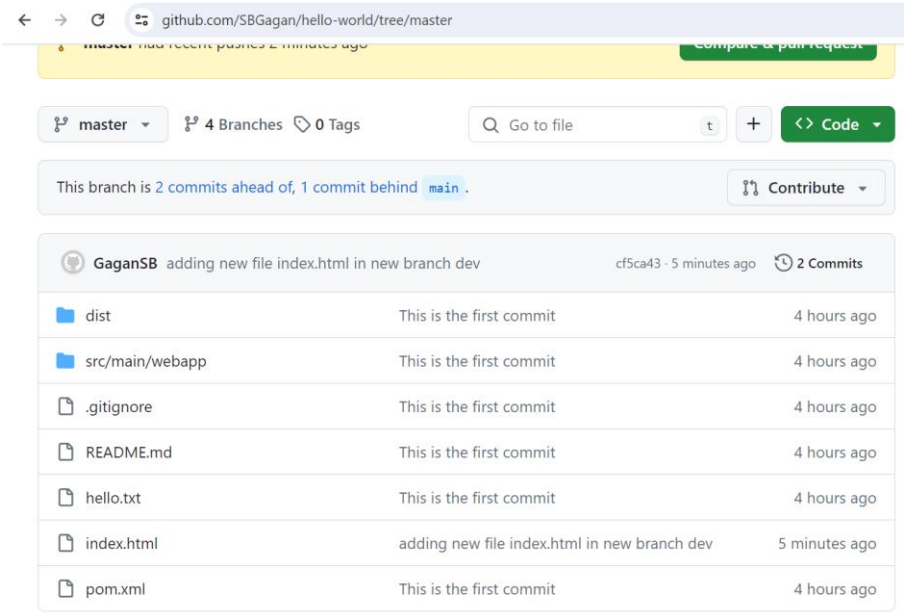
Task-2: Pushing the Code to your Remote GitHub Repository

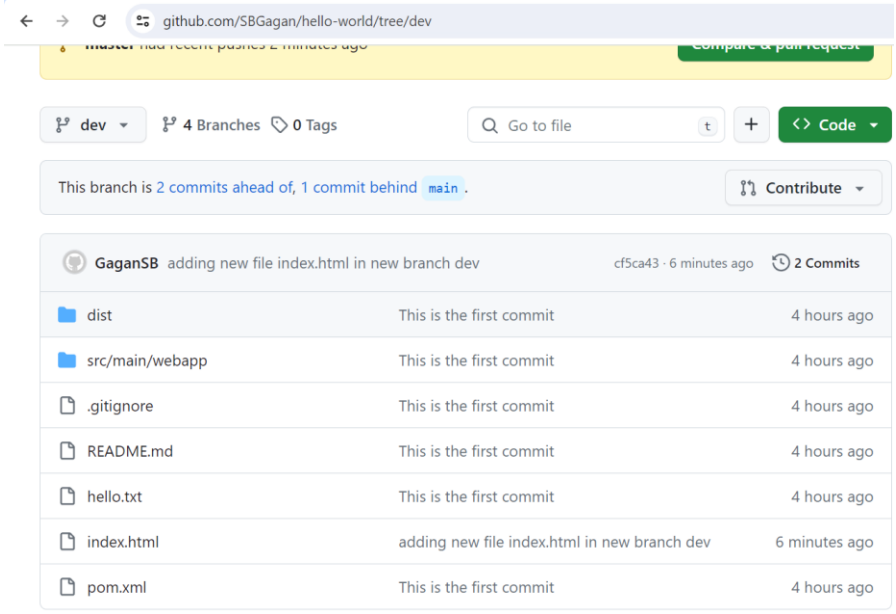
```
  112  git remote add origin https://github.com/SBGagan/hello-world.git
  113  git remote show origin
  114  git push origin master
  115  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
  116  git push origin master
  117  history
ubuntu@AnchorServer:~/hello-world-master$
```

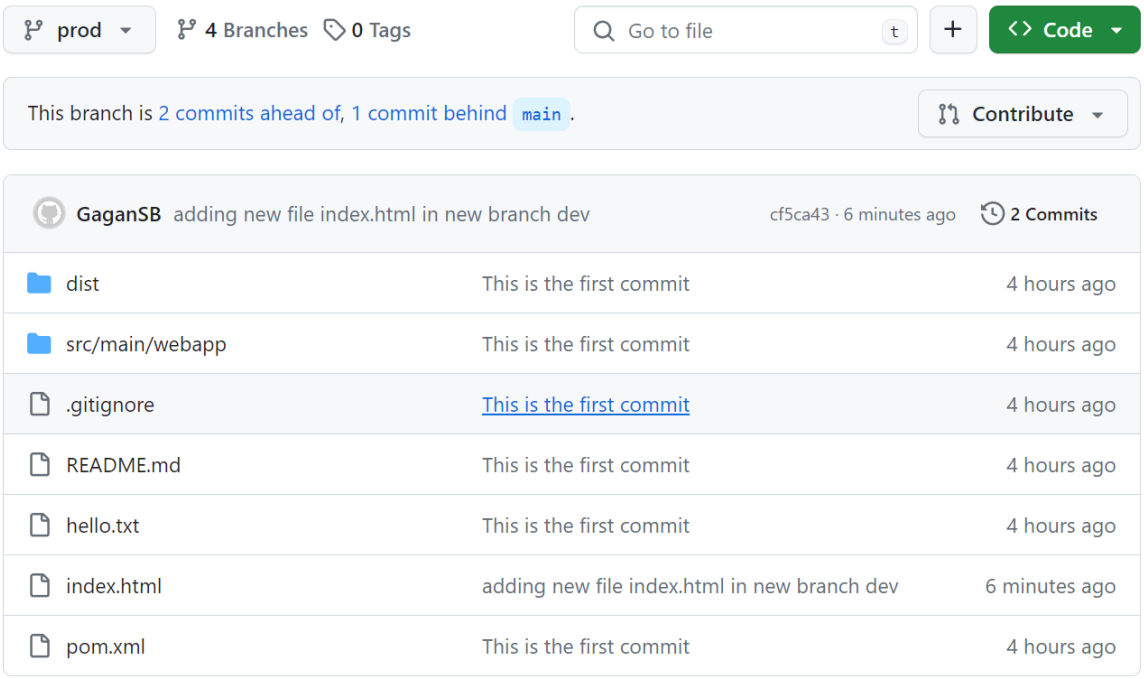
Task 3: Creating a Git Branch and Pushing Code to the Remote Repository

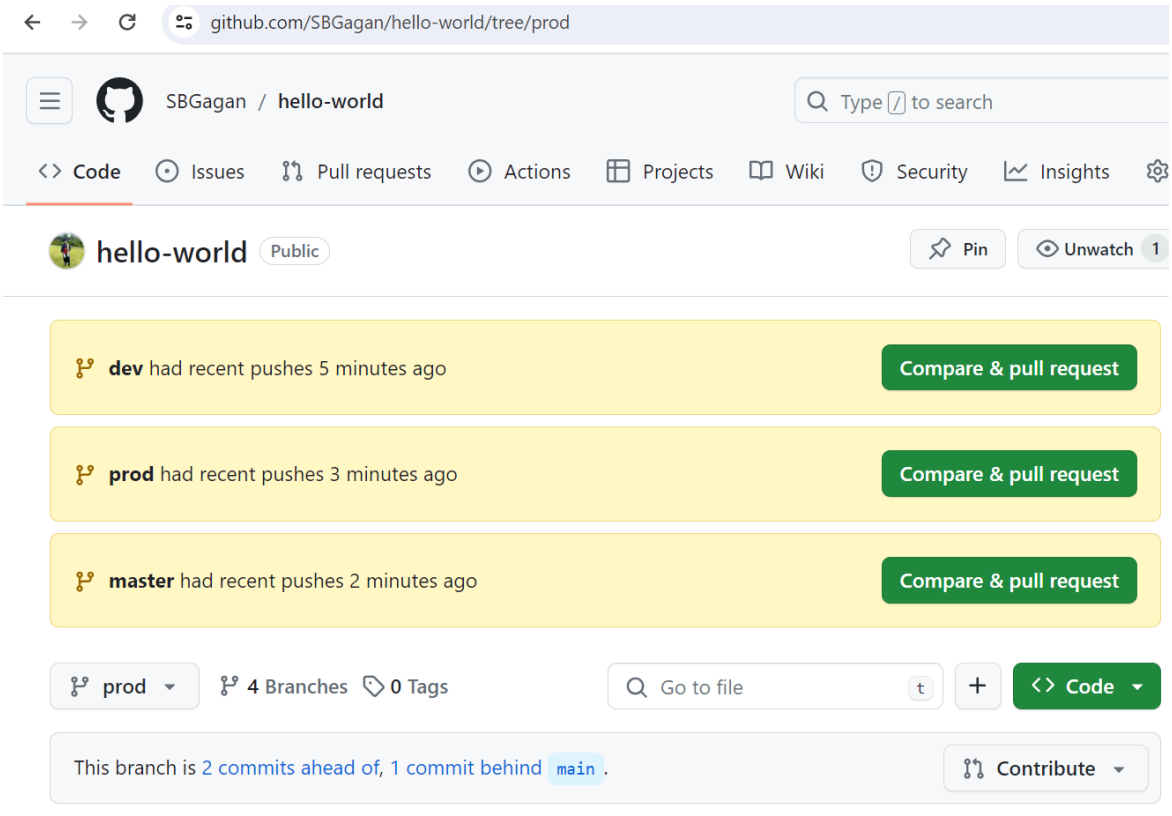
```
  118  git branch dev
  119  git branch
  120  git checkout dev
  121  git branch
  122  vi index.html
  123  cat index.html
  124  git status
  125  git add index.html
  126  git status
  127  git commit -m "adding new file index.html in new branch dev"
  128  git log
  129  git push origin dev
  130  git branch
  131  git branch prod
  132  git branch
  133  git checkout prod
  134  git branch
  135  git merge dev
  136  git push origin prod
  137  git checkout master
  138  git merge prod
  139  git push origin master
  140  history
ubuntu@AnchorServer:~/hello-world-master$
```


github.com/SBGagan/hello-world
master had recent pushes 1 minute ago
Compare & pull request
main
4 Branches
0 Tags
Go to file
t
Code
Switch branches/tags
Find or create a branch...
Branches
Tags
main
default
dev
master
prod
View all branches
fc061d8 · 5 hours ago
1 Commits
Initial commit
5 hours ago


github.com/SBGagan/hello-world/tree/master
master
4 Branches
0 Tags
Go to file
t
Code
This branch is 2 commits ahead of, 1 commit behind main.
Contribute
GaganSB adding new file index.html in new branch dev
cf5ca43 · 5 minutes ago
2 Commits
dist This is the first commit 4 hours ago
src/main/webapp This is the first commit 4 hours ago
.gitignore This is the first commit 4 hours ago
README.md This is the first commit 4 hours ago
hello.txt This is the first commit 4 hours ago
index.html adding new file index.html in new branch dev 5 minutes ago
pom.xml This is the first commit 4 hours ago


github.com/SBGagan/hello-world/tree/dev
dev
4 Branches
0 Tags
Go to file
t
Code
This branch is 2 commits ahead of, 1 commit behind main.
Contribute
GaganSB adding new file index.html in new branch dev
cf5ca43 · 6 minutes ago
2 Commits
dist This is the first commit 4 hours ago
src/main/webapp This is the first commit 4 hours ago
.gitignore This is the first commit 4 hours ago
README.md This is the first commit 4 hours ago
hello.txt This is the first commit 4 hours ago
index.html adding new file index.html in new branch dev 6 minutes ago
pom.xml This is the first commit 4 hours ago


prod
4 Branches
0 Tags
Go to file
t
Code
This branch is 2 commits ahead of, 1 commit behind main.
Contribute
GaganSB adding new file index.html in new branch dev
cf5ca43 · 6 minutes ago
2 Commits
dist This is the first commit 4 hours ago
src/main/webapp This is the first commit 4 hours ago
.gitignore This is the first commit 4 hours ago
README.md This is the first commit 4 hours ago
hello.txt This is the first commit 4 hours ago
index.html adding new file index.html in new branch dev 6 minutes ago
pom.xml This is the first commit 4 hours ago


github.com/SBGagan/hello-world/tree/prod
SBGagan / hello-world
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
hello-world
Public
Pin
Unwatch 1
dev had recent pushes 5 minutes ago
Compare & pull request
prod had recent pushes 3 minutes ago
Compare & pull request
master had recent pushes 2 minutes ago
Compare & pull request
prod
4 Branches
0 Tags
Go to file
t
Code
This branch is 2 commits ahead of, 1 commit behind main.
Contribute

Step9: Configuring Jenkins server and Installing Tomcat onto Jenkin's Server for Deploying our Application.

Objective: The objective of this project is to configure Jenkins to build and deploy applications. It includes Setting up Jenkins, installing necessary plugins and configuring Jenkins to build Maven projects, and `Installing Tomcat Server for viewing the Web Page.'

Task-1: Configure Jenkins Server

Initially, I Copied the private key from Anchor Server to the Jenkins Server & Docker Server. so, that I can SSH from Jenkins Server to Docker Server and viseversa.

1. **ansible jenkins-server -m copy -a "src=/home/ubuntu/.ssh/id_rsa dest=/home/ubuntu/.ssh/id_rsa" -b**

Here's a breakdown of the command:

- **ansible:** The Ansible command-line tool.
- **jenkins-server:** The target machine specified in our inventory file.
- **-m copy:** Specifies the Ansible module to use, in this case, the copy module.
- **-a:** "src=/home/ubuntu/.ssh/id_rsa dest=/home/ubuntu/.ssh/id_rsa": Specifies the arguments for the module, indicating the source and destination paths for the file copy.
- **-b:** Running the Ansible command with elevated privileges.

2. **ansible docker-server -m copy -a "src=/home/ubuntu/.ssh/id_rsa dest=/home/ubuntu/.ssh/id_rsa" -b**

Here's a breakdown of the command:

- **ansible:** The Ansible command-line tool.
- **docker-server:** The target machine specified in our inventory file.
- **-m copy:** Specifies the Ansible module to use, in this case, the copy module.

- **-a:** "src=/home/ubuntu/.ssh/id_rsa dest=/home/ubuntu/.ssh/id_rsa": Specifies the arguments for the module, indicating the source and destination paths for the file copy.
- **-b:** Running the Ansible command with elevated privileges.

```
  144  sudo vi /etc/ansible/hosts
  145  ansible jenkins-server -m copy -a "src=/home/ubuntu/.ssh/id_rsa dest=/home/ubuntu/.ssh/id_rsa" -b
  146  ansible docker-server -m copy -a "src=/home/ubuntu/.ssh/id_rsa dest=/home/ubuntu/.ssh/id_rsa" -b
  147  ssh ubuntu@44.203.93.127
  148  history
ubuntu@AnchorServer:~$
```

Step-1:

1. Go to the **Web Browser** and open a new tab then enter the URL as shown:

   http://< Jenkin's Public IP>:8080/ (It requests the **InitialAdminPassword** during the setup.)

2. To obtain the **InitialAdminPassword**, access the Jenkins Server by SSHing from the Anchor Server, utilizing Jenkins' Public IP.


```
    8  sudo cat /var/lib/jenkins/secrets/initialAdminPassword
    9  history
   10  exit
   11  history
ubuntu@jenkins-server:~$
```

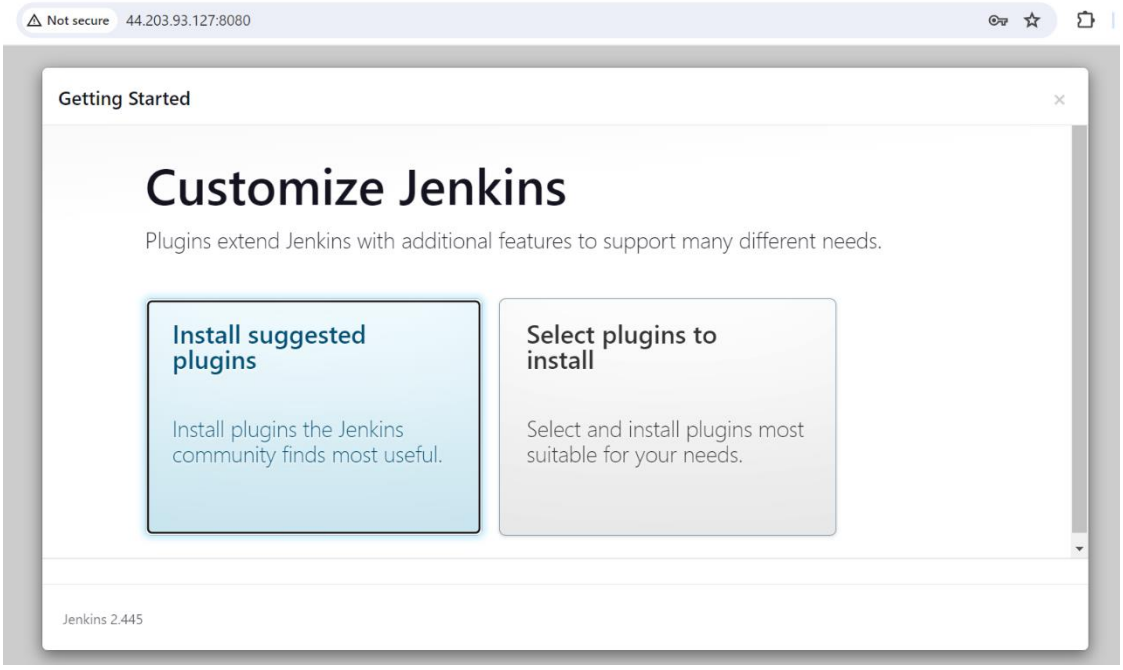
Step-2:

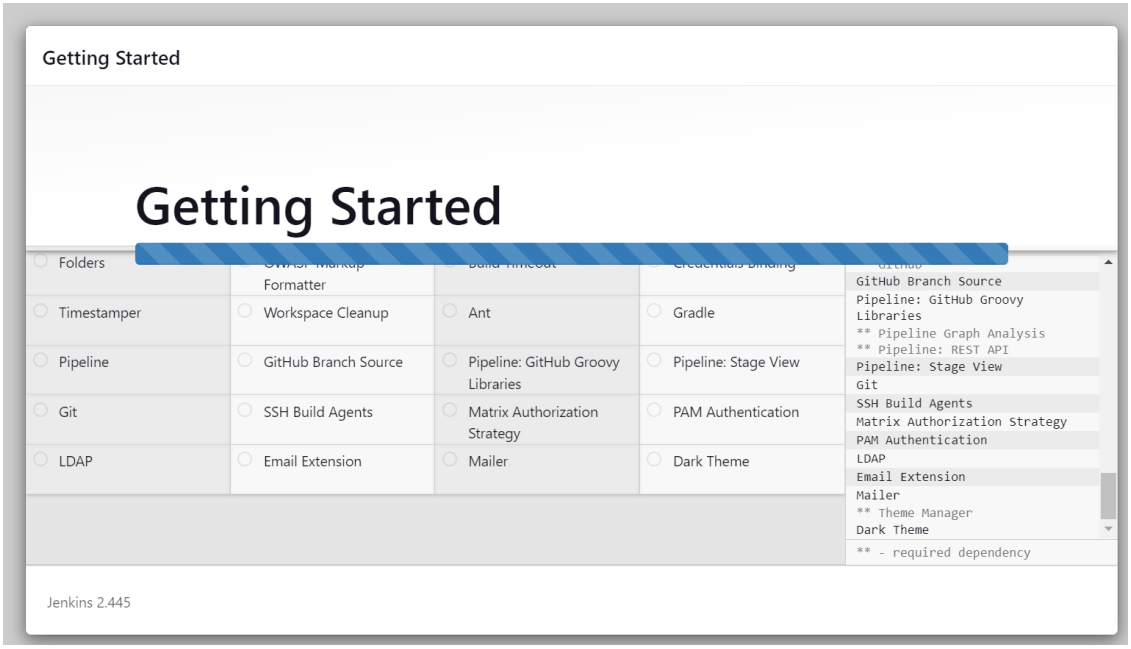
1. Now, go back to jenkin's landing page in the **Web Browser**:

   (Enter the Jenkins URL as shown: **http://< Jenkin's Public IP>:8080/**)

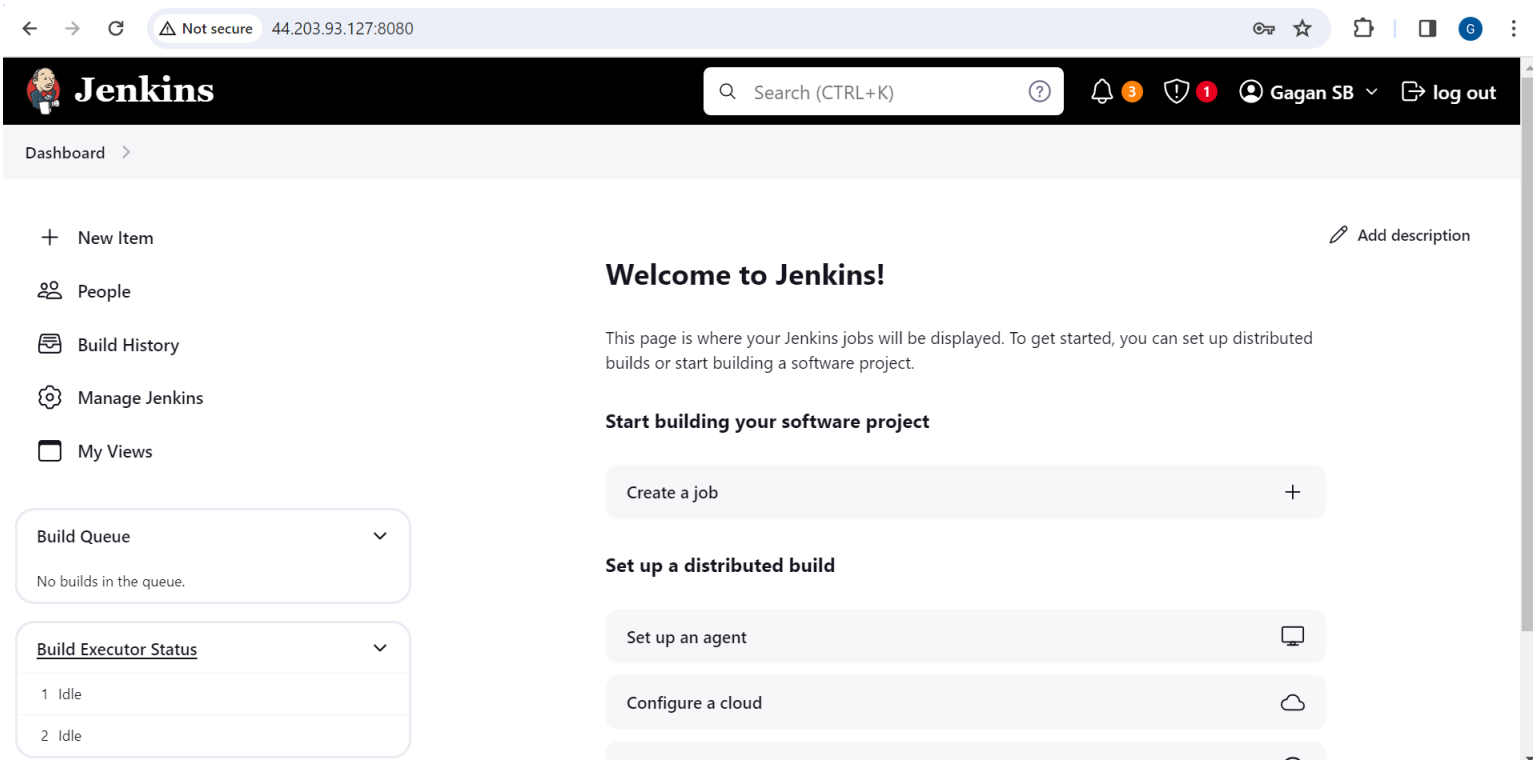
2. Under Unlock Jenkins, enter the above **initialAdminPassword** & click **Continue**.

3. Click on **Install suggested Plugins** on the Customize Jenkins page.
4. Once the plugins are installed, it gives you the page where you can create a New **Admin User**.
5. Enter the **User Id** and **Password** followed by **Name** and **E-Mail ID** then click on **Save & Continue**.
6. In the next step, on the Instance Configuration Page, verify your **Jenkins Public IP** and **Port Number** then click on **Save and Finish**





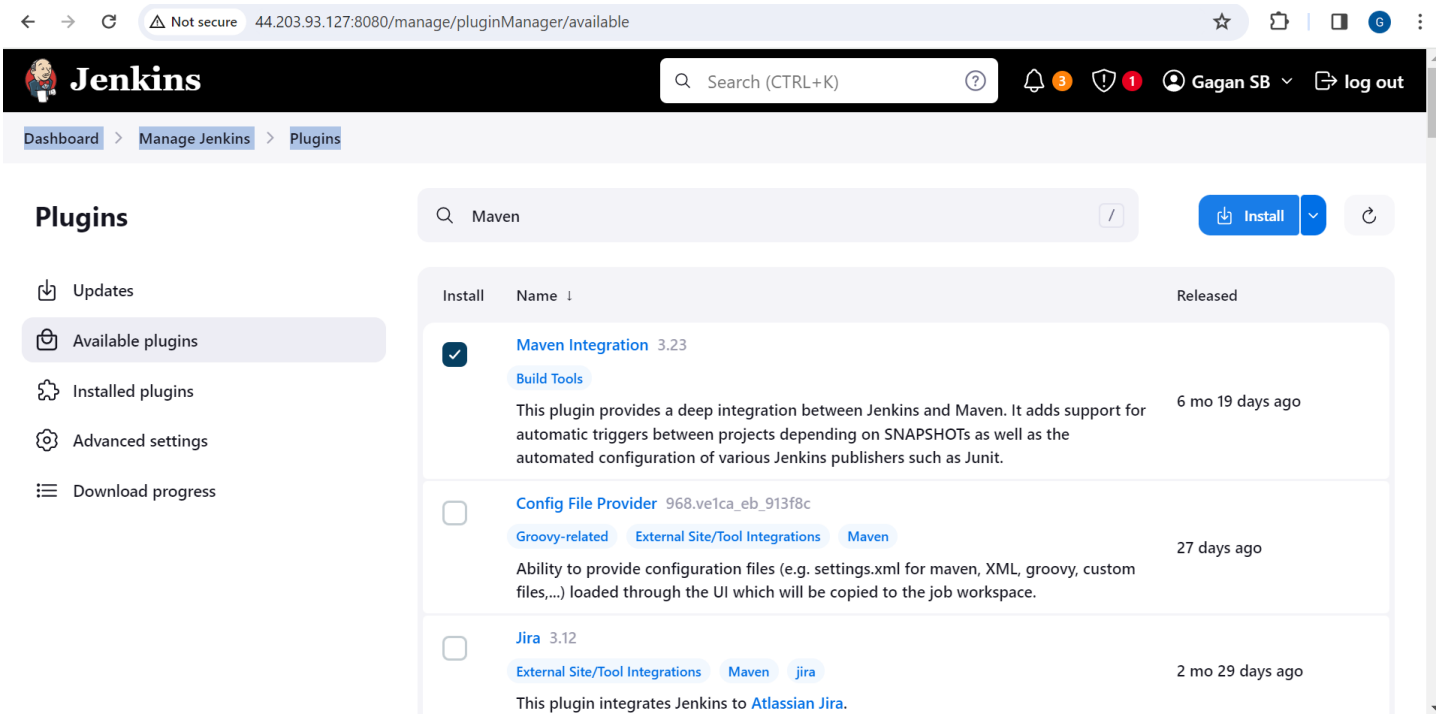

Getting Started
Create First Admin User
Username
Gagan SB
Password
Confirm password
Jenkins 2.445
Skip and continue as admin
Save and Continue

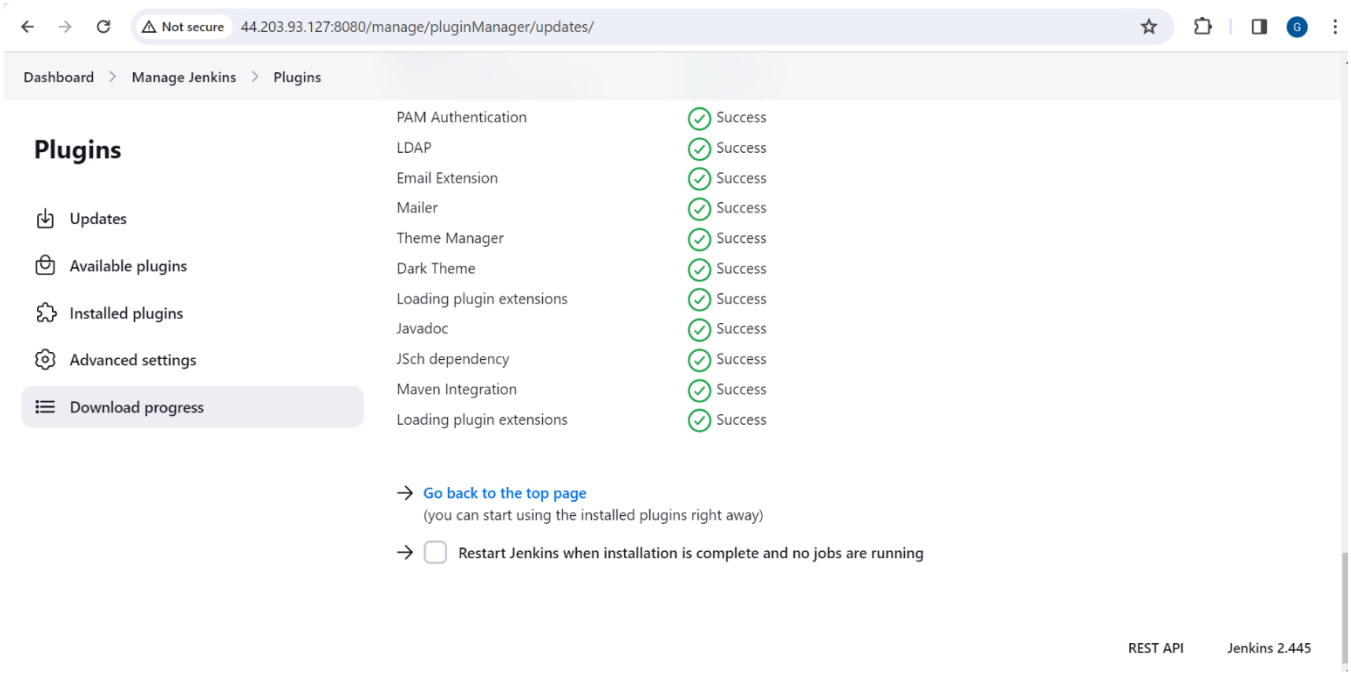

Not secure
44.203.93.127:8080
Jenkins
Search (CTRL+K)
Gagan SB
log out
Dashboard
New Item
People
Build History
Manage Jenkins
My Views
Build Queue
No builds in the queue.
Build Executor Status
1 Idle
2 Idle
Add description
Welcome to Jenkins!
This page is where your Jenkins jobs will be displayed. To 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

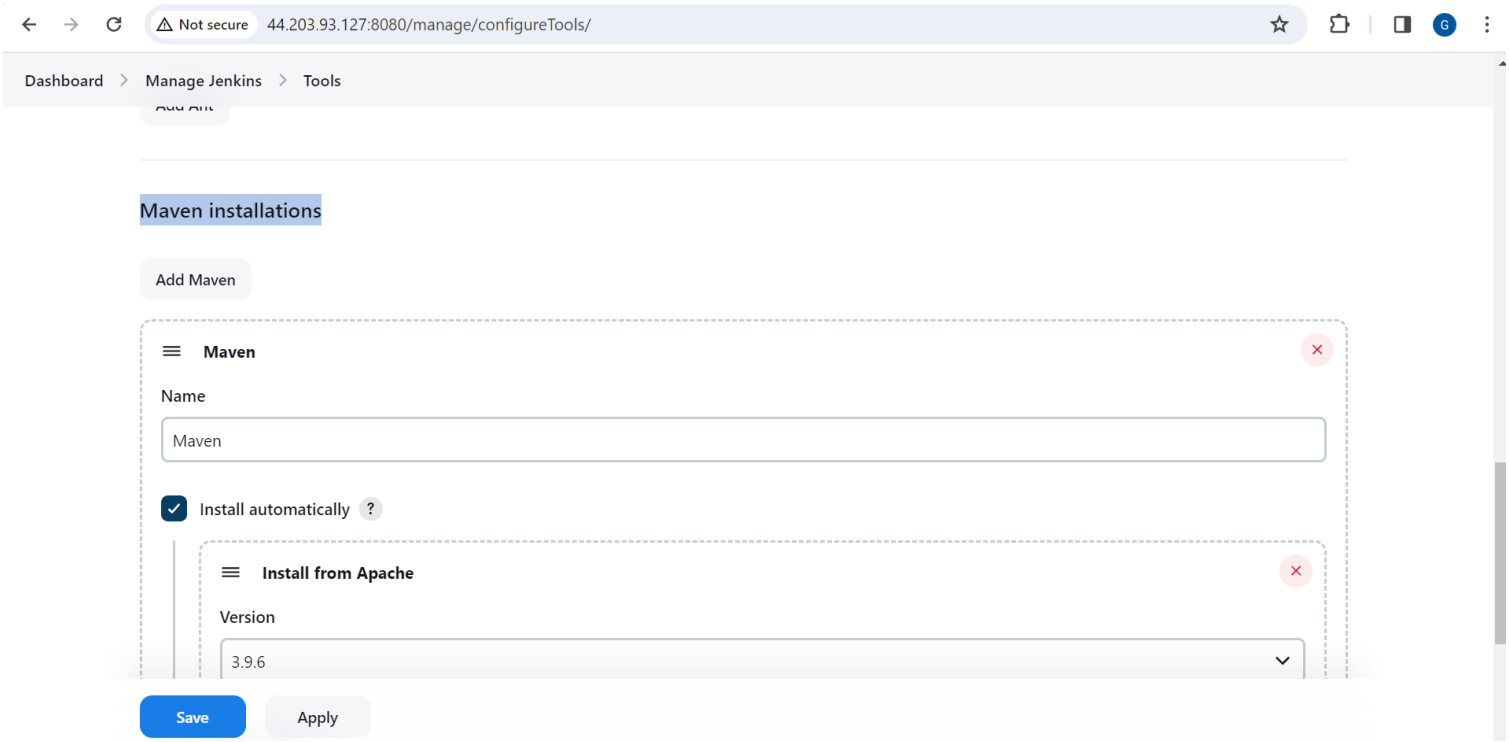
Step-3: Configuring Maven in Jenkins

Do the below in Jenkin's Dashboard:

1. Click on **Manage Jenkins** > **Plugins** > **Available Plugins** tab and search for **Maven**.
2. Select the "**Maven Integration**" Plugin and click **Install** (without restart).
3. Once the installation is completed, click on **Go back to the top page**.
4. On Home Page select **Manage Jenkins** > **Tool**.
5. Inside Tool Configuration, look for **Maven installations**, click **Add Maven**.
6. Give the Name as **"Maven"**, Version-Default one (Latest), and **Save** the configuration.







Step-4:

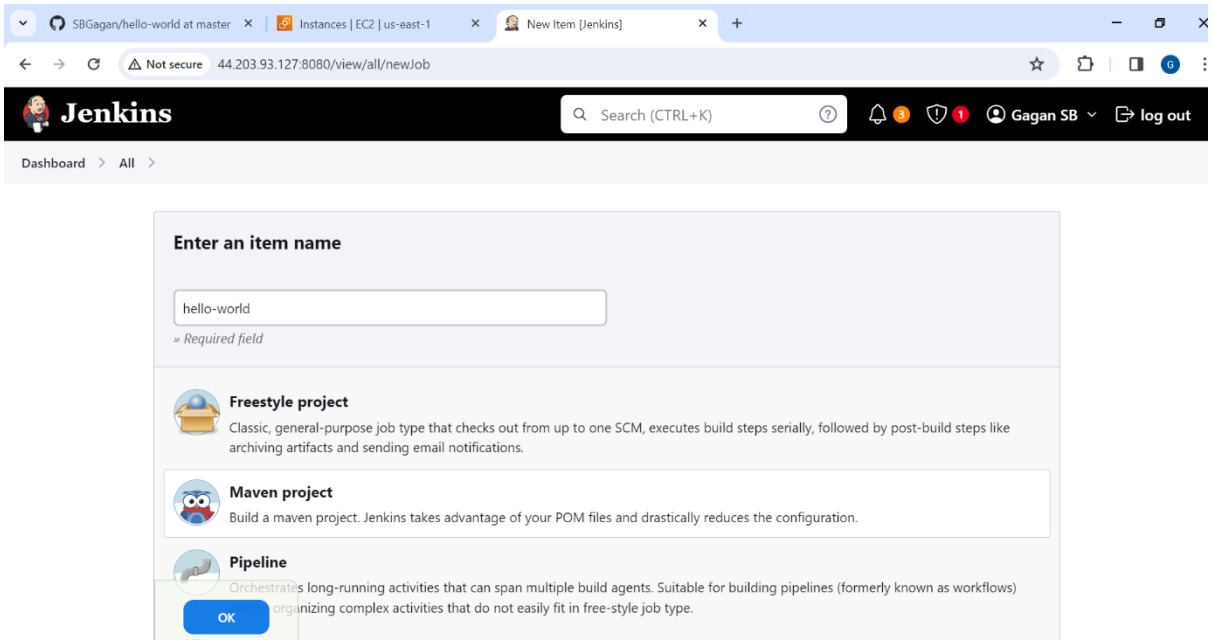
1. Create a new project for our application build by selecting **New Item** from the Jenkins homepage.
2. Enter an item name as **hello-world** and select the project as **Maven Project** and then click **OK.** (We will be prompted to the configure page inside the hello-world project.)
3. Go to the "**Source Code Management**" tab, and select Source Code Management as **Git**, Now we need to provide the GitHub Repository's **Master Branch URL** and **GitHub Account Credentials**.
4. In the Credentials field, we have to click **Add** and then click on **Jenkins**.
5. Then we will be prompted to the **Jenkins Credentials Provider** page. Under Add Credentials, we can add your **GitHub Username**, **Password**, and **Description**. Then click on **Add**.
6. In the Source Code Management page, navigate to Credentials, and select your GitHub credentials.
7. Leave all other values as default, navigate to the "**Build**" tab, and in the "Goals and options" section, input "**clean package**". Save the configuration.

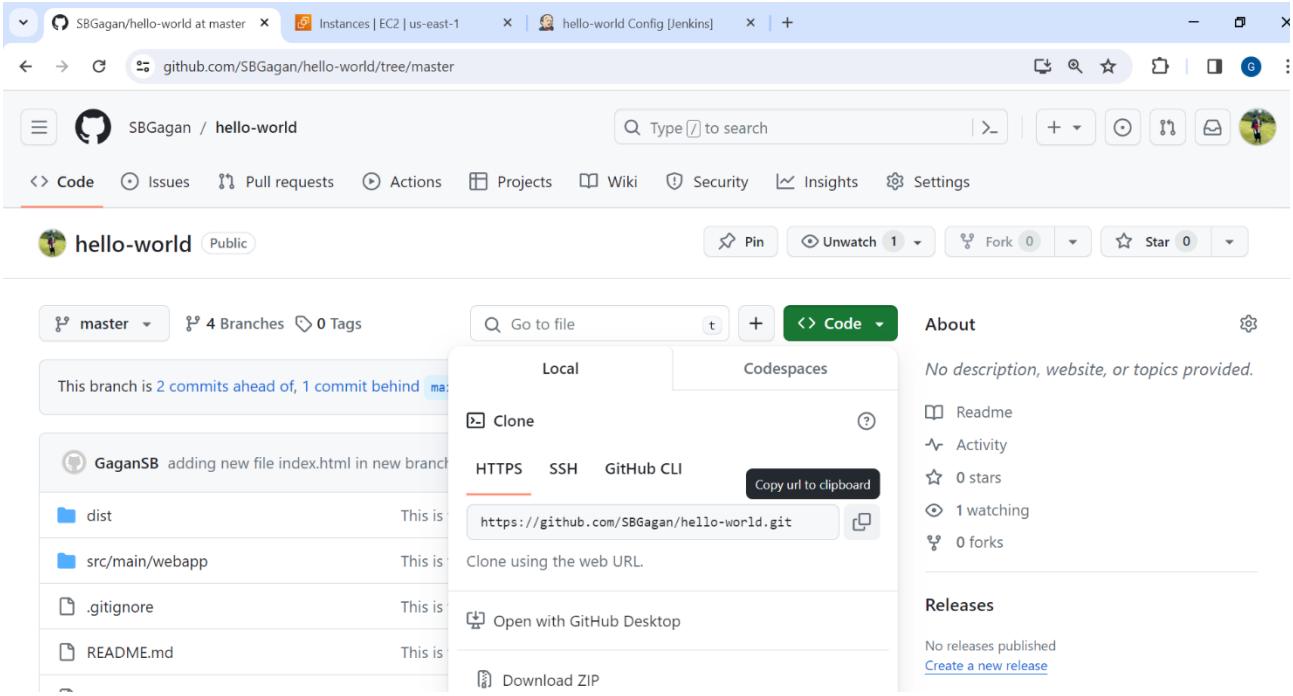
   Note:

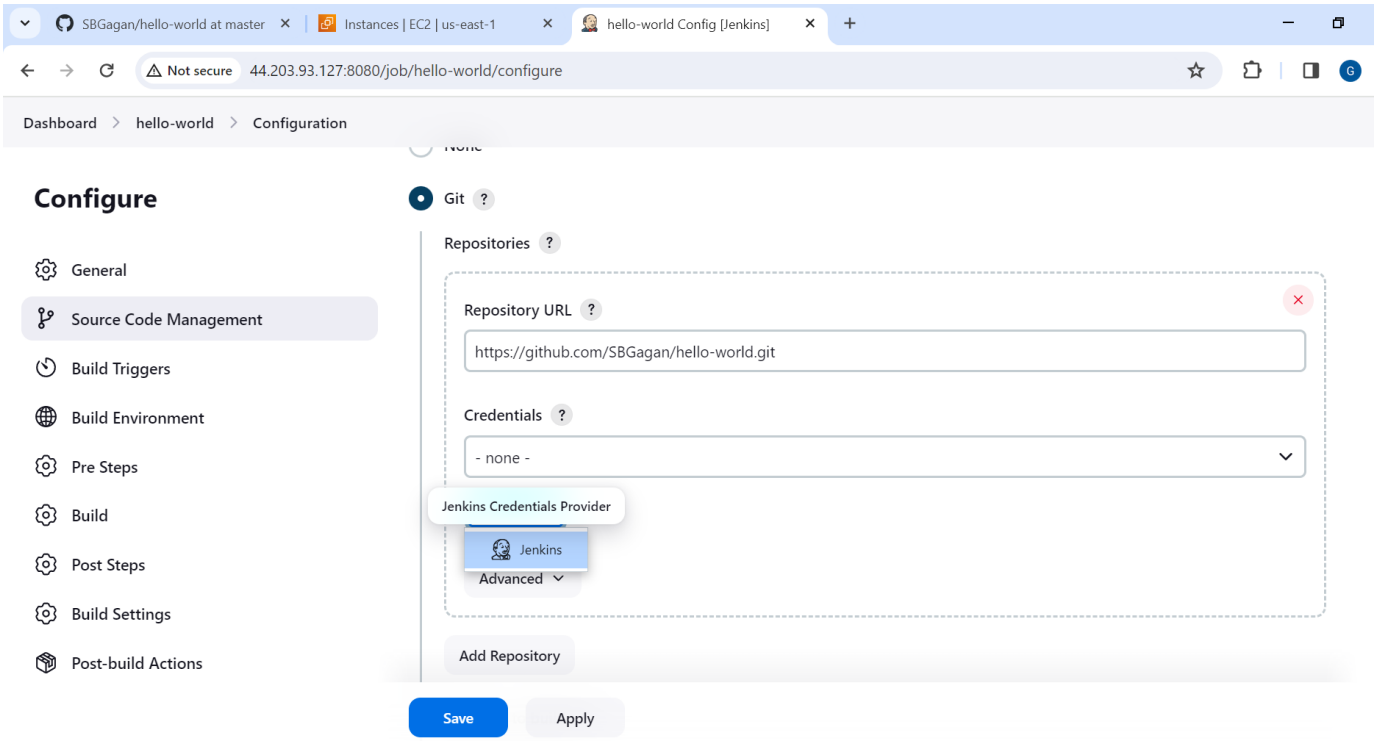
   The 'clean package' command clears the target directory, Builds the project, and packages the resulting WAR file into the target directory.

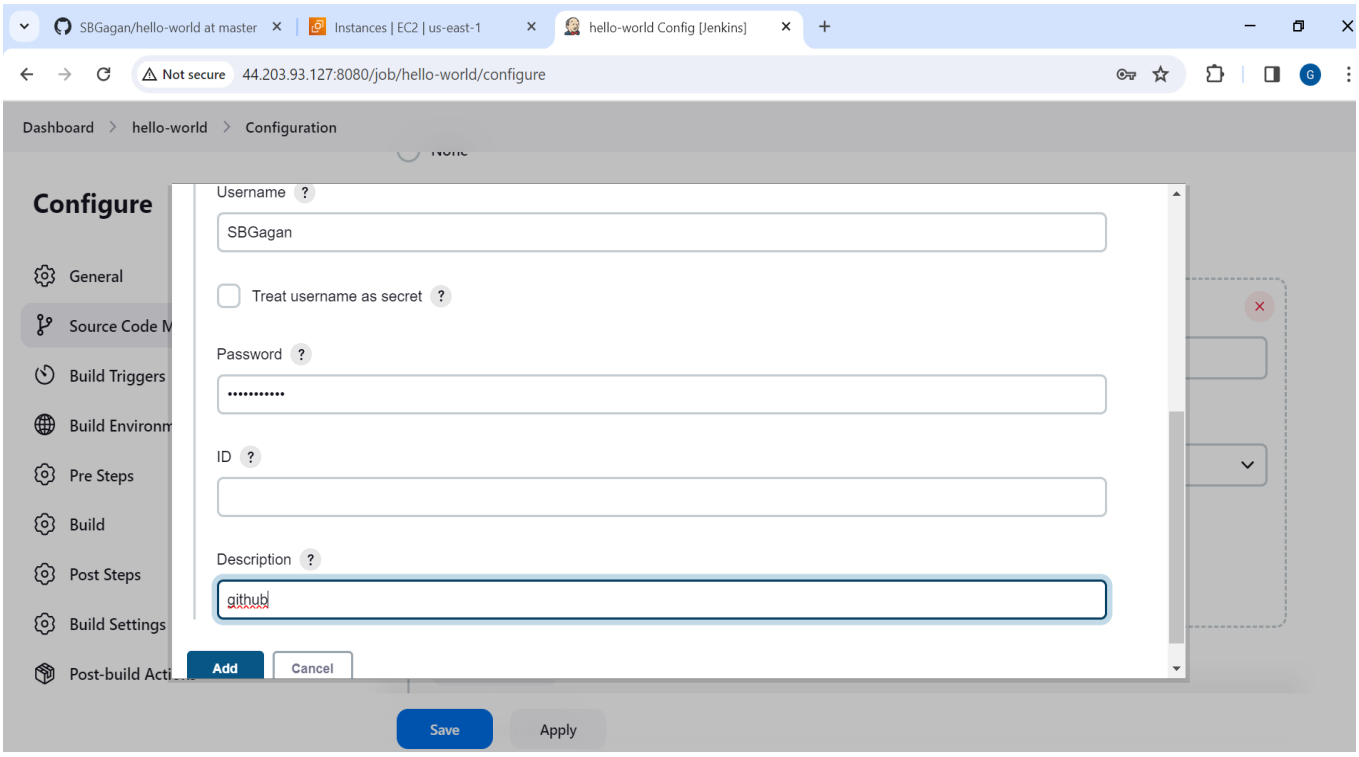
8. Return to the Maven project "**hello-world**" and click on "**Build Now**" to initiate the build process for your application's **.war** file.

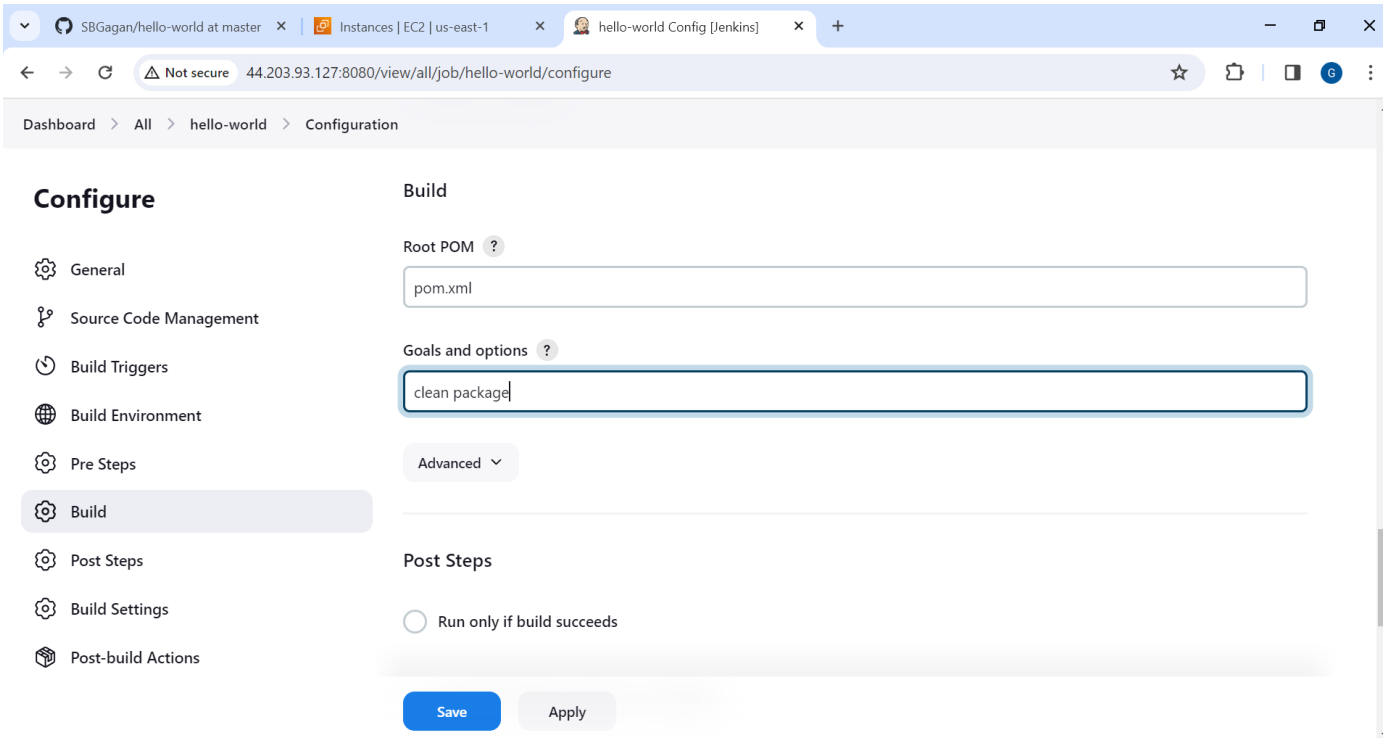
    - we can go to **Workspace** > **dist** folder to see that the **.war** file is created there.
    - war file will be created in **/var/lib/jenkins/workspace/hello-world/target/**

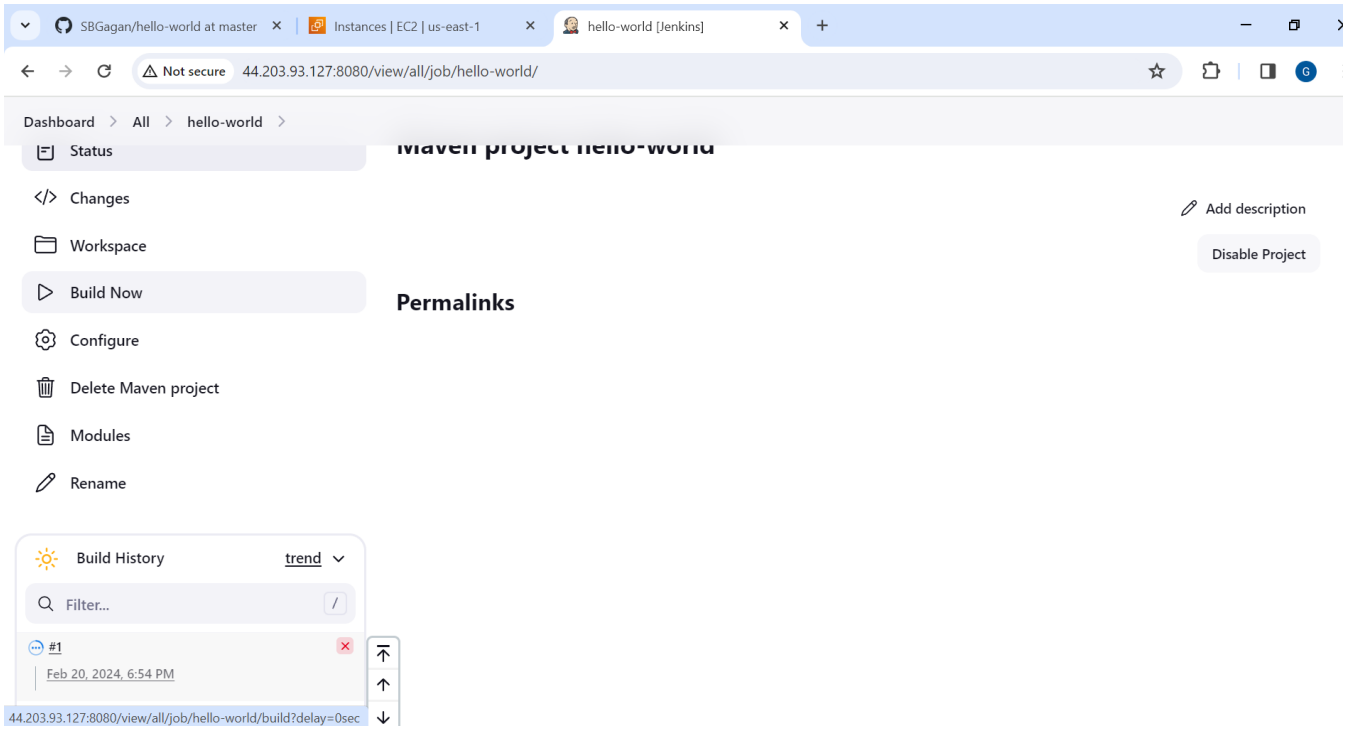


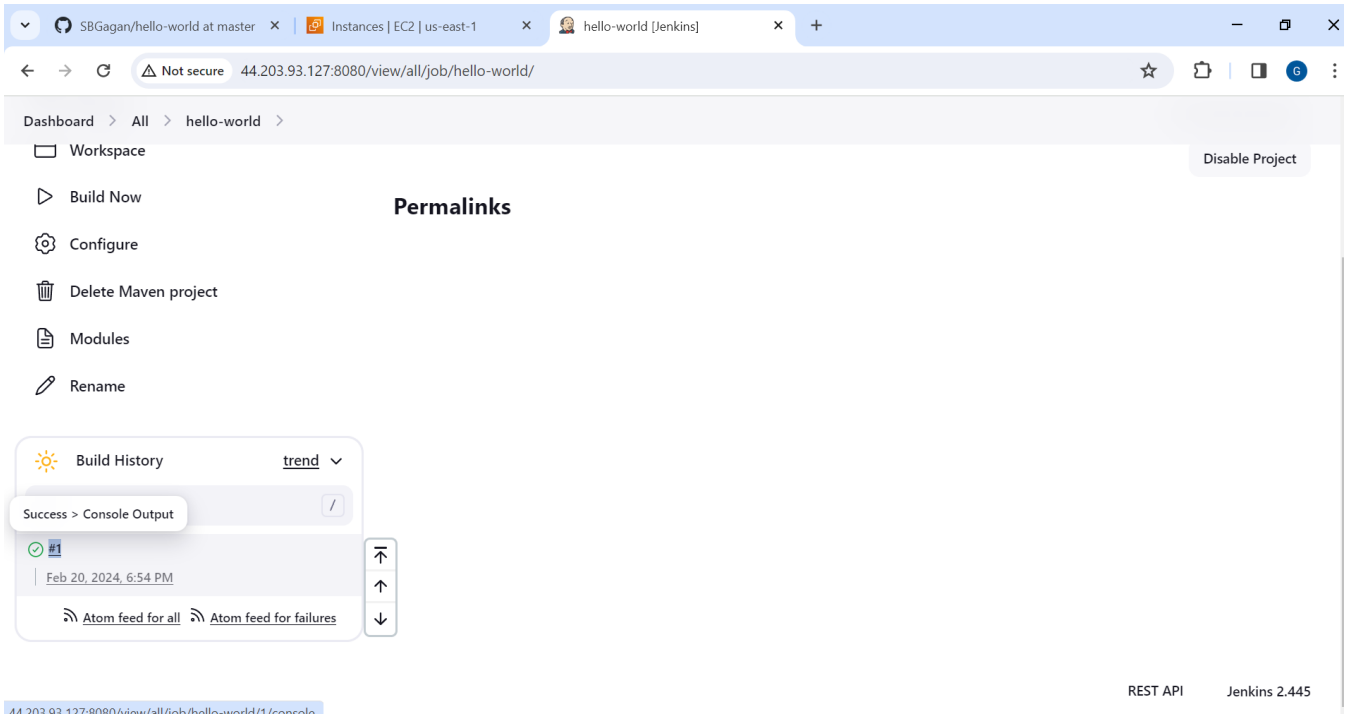


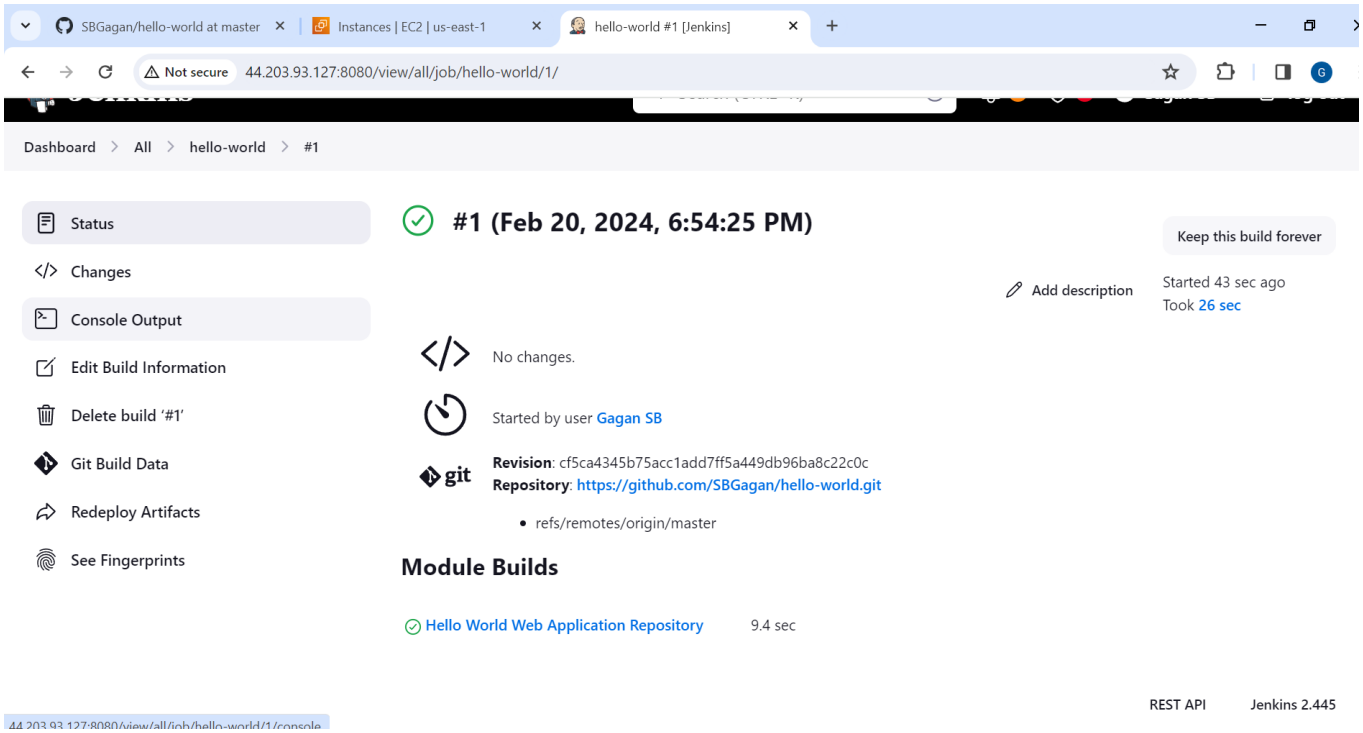

SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
44.203.93.127:8080/job/hello-world/configure
Dashboard
hello-world
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Git
Repositories
Repository URL
https://github.com/SBGagan/hello-world.git
Credentials
- none -
Jenkins Credentials Provider
Jenkins
Advanced
Add Repository
Save
Apply

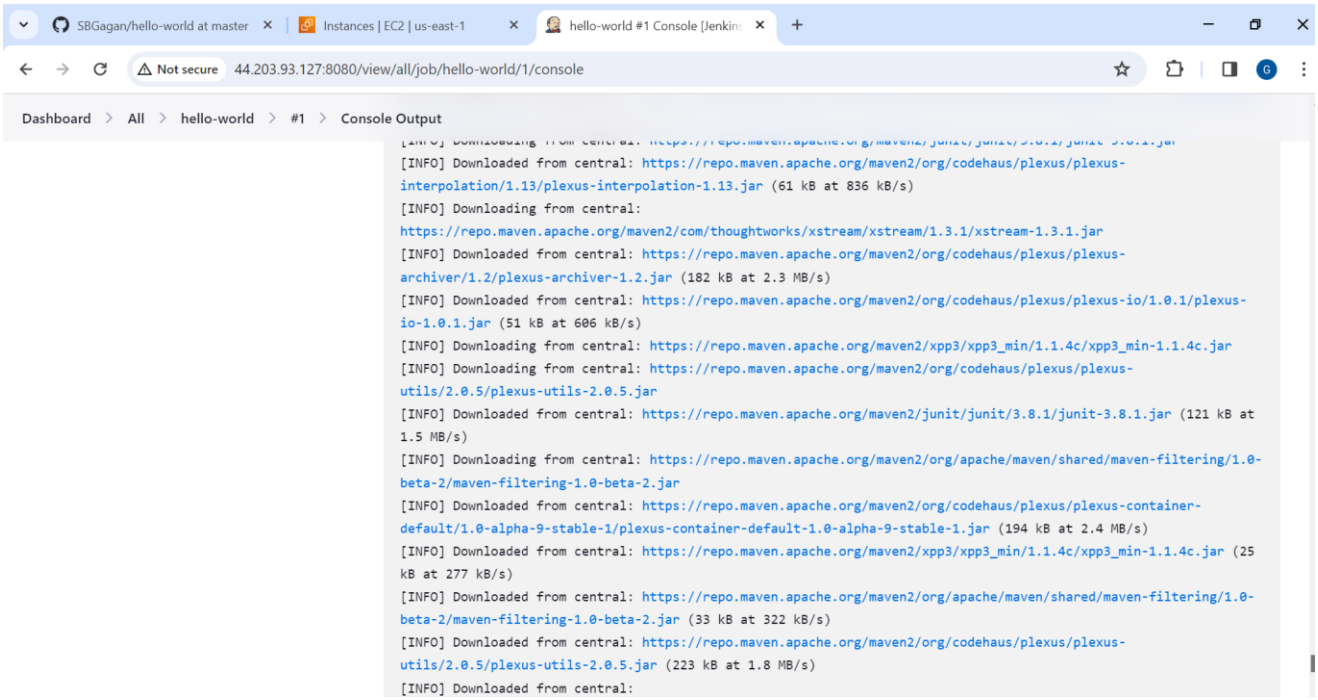

SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
44.203.93.127:8080/job/hello-world/configure
Dashboard
hello-world
Configuration
Configure
Username
SBGagan
Treat username as secret
Password
ID
Description
github
Add
Cancel
Save
Apply

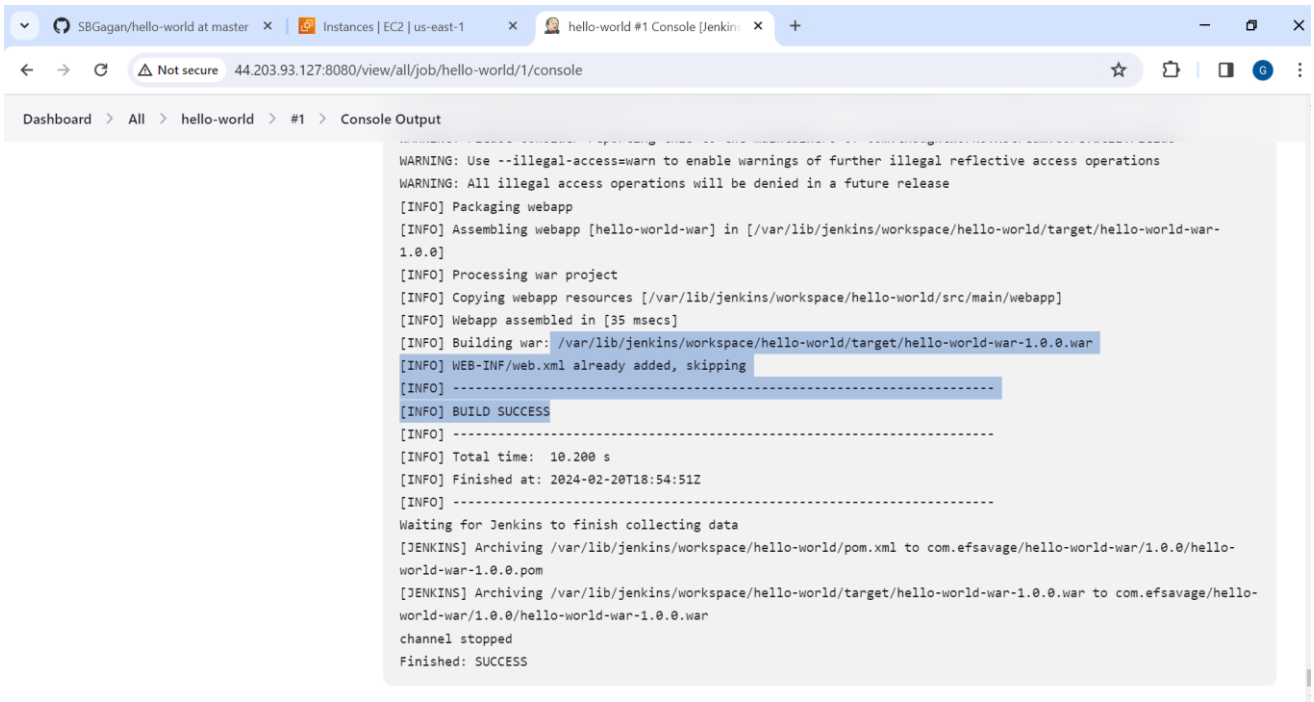

SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
44.203.93.127:8080/view/all/job/hello-world/configure
Dashboard > All > hello-world > Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build
Root POM
pom.xml
Goals and options
clean package
Advanced
Post Steps
Run only if build succeeds
Save
Apply

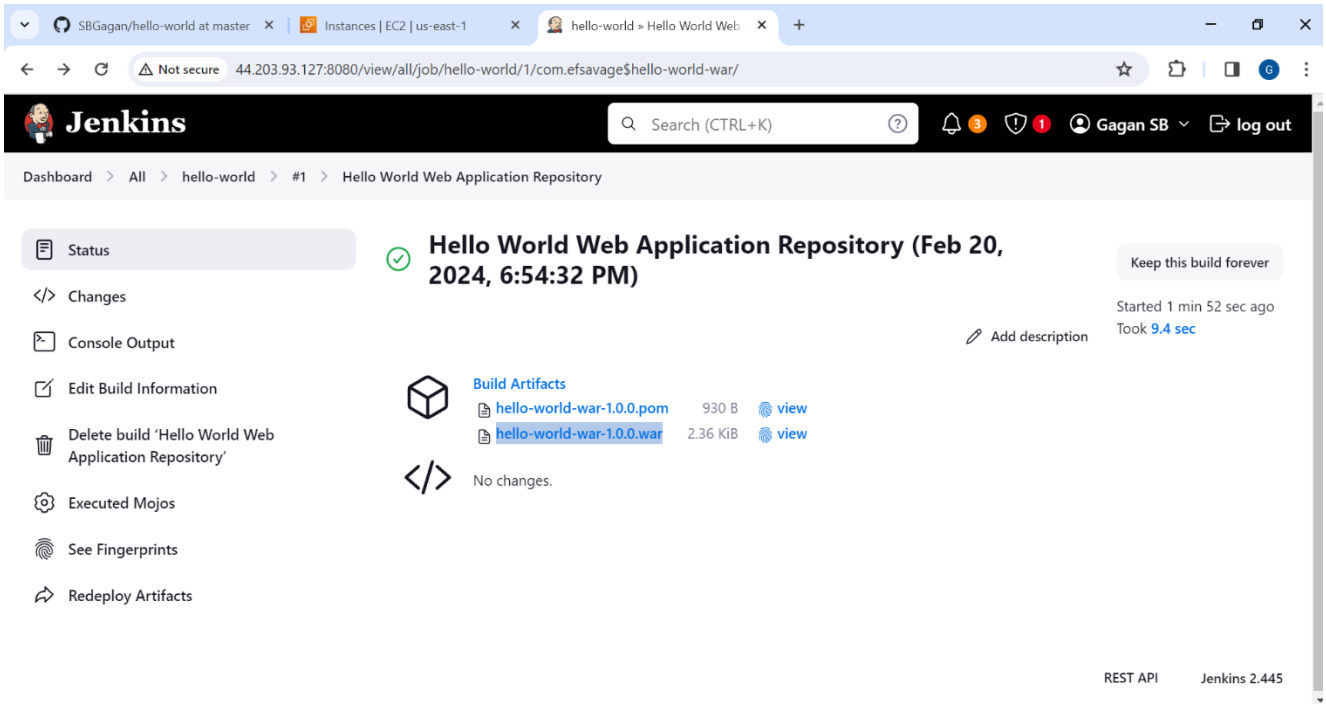

SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world [Jenkins]
Not secure
44.203.93.127:8080/view/all/job/hello-world/
Dashboard > All > hello-world >
Status
Changes
Workspace
Build Now
Configure
Delete Maven project
Modules
Rename
Add description
Disable Project
Permalinks
Build History
trend
Filter...
#1
Feb 20, 2024, 6:54 PM
44.203.93.127:8080/view/all/job/hello-world/build?delay=0sec

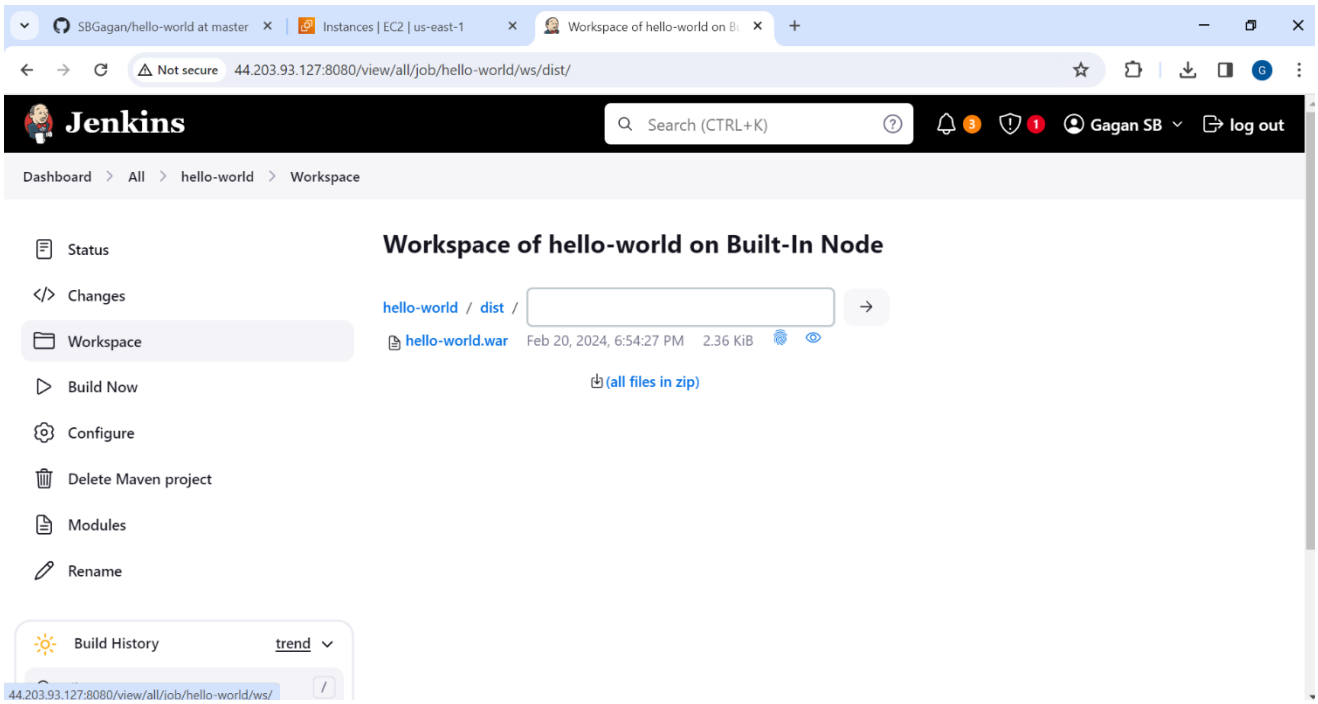

SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world [Jenkins]
Not secure
44.203.93.127:8080/view/all/job/hello-world/
Dashboard
All
hello-world
Workspace
Build Now
Configure
Delete Maven project
Modules
Rename
Disable Project
Permalinks
Build History
trend
Success > Console Output
#1
Feb 20, 2024, 6:54 PM
Atom feed for all
Atom feed for failures
REST API
Jenkins 2.445


SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world #1 [Jenkins]
Not secure
44.203.93.127:8080/view/all/job/hello-world/1/
Dashboard
All
hello-world
#1
Status
Changes
Console Output
Edit Build Information
Delete build '#1'
Git Build Data
Redeploy Artifacts
See Fingerprints
#1 (Feb 20, 2024, 6:54:25 PM)
Keep this build forever
Add description
Started 43 sec ago
Took 26 sec
No changes.
Started by user Gagan SB
Revision: cf5ca4345b75acc1add7ff5a449db96ba8c22c0c
Repository: https://github.com/SBGagan/hello-world.git
refs/remotes/origin/master
Module Builds
Hello World Web Application Repository
9.4 sec
REST API
Jenkins 2.445


SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world #1 Console [Jenkins
Not secure
44.203.93.127:8080/view/all/job/hello-world/1/console
Dashboard > All > hello-world > #1 > Console Output
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-interpolation/1.13/plexus-interpolation-1.13.jar (61 kB at 836 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/com/thoughtworks/xstream/xstream/1.3.1/xstream-1.3.1.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-archiver/1.2/plexus-archiver-1.2.jar (182 kB at 2.3 MB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-io/1.0.1/plexus-io-1.0.1.jar (51 kB at 606 kB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/xpp3/xpp3_min/1.1.4c/xpp3_min-1.1.4c.jar
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.5/plexus-utils-2.0.5.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/junit/junit/3.8.1/junit-3.8.1.jar (121 kB at 1.5 MB/s)
[INFO] Downloading from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/1.0-beta-2/maven-filtering-1.0-beta-2.jar
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-container-default/1.0-alpha-9-stable-1/plexus-container-default-1.0-alpha-9-stable-1.jar (194 kB at 2.4 MB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/xpp3/xpp3_min/1.1.4c/xpp3_min-1.1.4c.jar (25 kB at 277 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/shared/maven-filtering/1.0-beta-2/maven-filtering-1.0-beta-2.jar (33 kB at 322 kB/s)
[INFO] Downloaded from central: https://repo.maven.apache.org/maven2/org/codehaus/plexus/plexus-utils/2.0.5/plexus-utils-2.0.5.jar (223 kB at 1.8 MB/s)
[INFO] Downloaded from central:


SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world #1 Console [Jenkins
Not secure
44.203.93.127:8080/view/all/job/hello-world/1/console
Dashboard > All > hello-world > #1 > Console Output
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflective access operations
WARNING: All illegal access operations will be denied in a future release
[INFO] Packaging webapp
[INFO] Assembling webapp [hello-world-war] in [/var/lib/jenkins/workspace/hello-world/target/hello-world-war-1.0.0]
[INFO] Processing war project
[INFO] Copying webapp resources [/var/lib/jenkins/workspace/hello-world/src/main/webapp]
[INFO] Webapp assembled in [35 msecs]
[INFO] Building war: /var/lib/jenkins/workspace/hello-world/target/hello-world-war-1.0.0.war
[INFO] WEB-INF/web.xml already added, skipping
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 10.200 s
[INFO] Finished at: 2024-02-20T18:54:51Z
[INFO] --
Waiting for Jenkins to finish collecting data
[JENKINS] Archiving /var/lib/jenkins/workspace/hello-world/pom.xml to com.efsavage/hello-world-war/1.0.0/hello-world-war-1.0.0.pom
[JENKINS] Archiving /var/lib/jenkins/workspace/hello-world/target/hello-world-war-1.0.0.war to com.efsavage/hello-world-war/1.0.0/hello-world-war-1.0.0.war
channel stopped
Finished: SUCCESS


SBGagan/hello-world at master
Instances | EC2 | us-east-1
hello-world » Hello World Web
Not secure
44.203.93.127:8080/view/all/job/hello-world/1/com.efsavage$hello-world-war/
Jenkins
Search (CTRL+K)
Gagan SB
log out
Dashboard > All > hello-world > #1 > Hello World Web Application Repository
Status
Changes
Console Output
Edit Build Information
Delete build 'Hello World Web Application Repository'
Executed Mojos
See Fingerprints
Redeploy Artifacts
Hello World Web Application Repository (Feb 20, 2024, 6:54:32 PM)
Keep this build forever
Started 1 min 52 sec ago
Took 9.4 sec
Add description
Build Artifacts
hello-world-war-1.0.0.pom 930 B view
hello-world-war-1.0.0.war 2.36 KiB view
No changes.
REST API
Jenkins 2.445


SBGagan/hello-world at master
Instances | EC2 | us-east-1
Workspace of hello-world on B
Not secure
44.203.93.127:8080/view/all/job/hello-world/ws/dist/
Jenkins
Search (CTRL+K)
Gagan SB
log out
Dashboard > All > hello-world > Workspace
Status
Changes
Workspace
Build Now
Configure
Delete Maven project
Modules
Rename
Build History
trend
Workspace of hello-world on Built-In Node
hello-world / dist /
hello-world.war Feb 20, 2024, 6:54:27 PM 2.36 KiB
(all files in zip)
44.203.93.127:8080/view/all/job/hello-world/ws/

Step10: **Installing and Configuring Tomcat for Deploying our Application on the same Jenkins Server**

Step-1: Install Tomcat on to Jenkins Server

- SSH into the Jenkins server as the root user and install the Tomcat web server.

```
   14  sudo apt update
   15  sudo apt install tomcat9 tomcat9-admin -y
   16  ss -ltn
   17  sudo systemctl enable tomcat9
   18  history
ubuntu@jenkins-server:~$
```

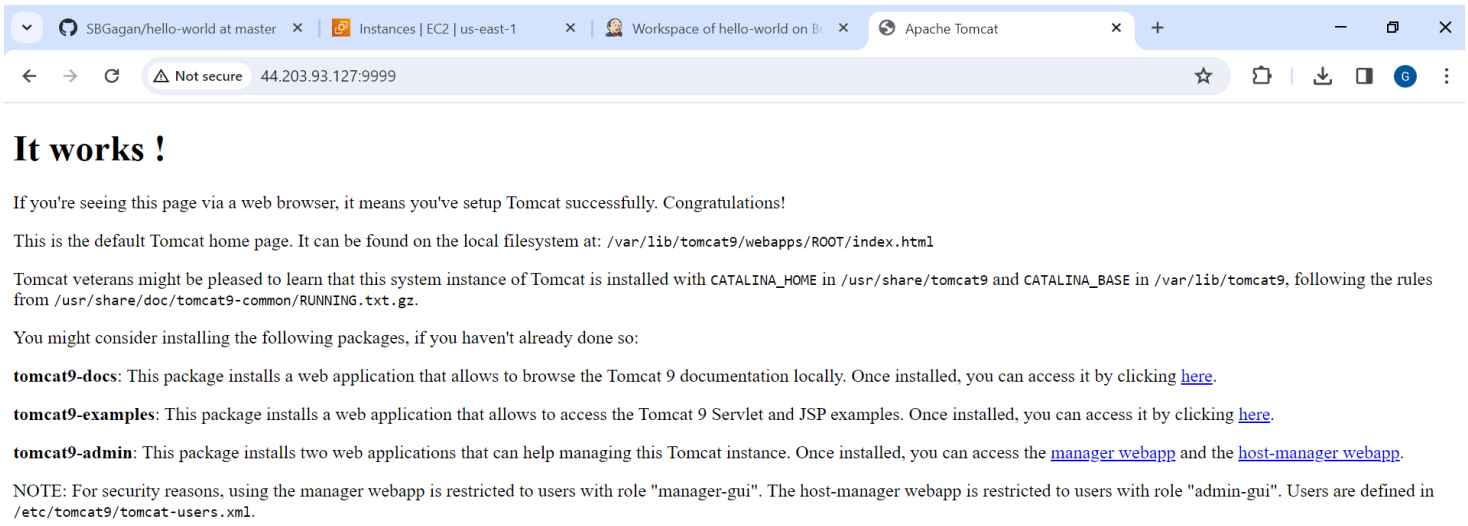
Step-2: Navigate to server.xml and change the Tomcat port number from 8080 to 9999 since port 8080 is already in use by Jenkins.

```
   14  sudo apt update
   15  sudo apt install tomcat9 tomcat9-admin -y
   16  ss -ltn
   17  sudo systemctl enable tomcat9
   18  history
   19  sudo vi /etc/tomcat9/server.xml
   20  cat /etc/tomcat9/server.xml
   21  sudo chmod 766 /etc/tomcat9/server.xml
   22  sudo service tomcat9 restart
   23  sudo service tomcat9 status
   24  history
ubuntu@jenkins-server:~$
```

- Once the Tomcat service restart is successful, go to web browser and enter **Jenkins Server's Public IP address** followed by **9999** port.

  (Example: **http://< Jenkins Public IP >:9999**)

  - Now we can check the Tomcat running on **port 9999** on the same machine and it shows the WebPage.



Step-3: Copy the previously built .war

- Copy the previously built **.war** file from the Jenkins workspace to the Tomcat webapps directory to deploy the web content.

sudo cp -R /var/lib/jenkins/workspace/hello-world/target/hello-world-war-1.0.0.war /var/lib/tomcat9/webapps

Here's a breakdown of the command:
The above command is copying a WAR (Web Application Archive) file from the Jenkins workspace to the Tomcat web apps directory. Let's break down the command:

- **sudo:** Run the command with superuser (root) privileges, as copying files to system directories often requires elevated permissions.
- **cp:** The copy command in Linux.
- **-R:** Recursive option, used when copying directories. It ensures that the entire directory structure is copied.
- /var/lib/jenkins/workspace/hello-world/target/hello-world-war-1.0.0.war: Source path, specifying the location of the WAR file in the Jenkins workspace.
- /var/lib/tomcat9/webapps: Destination path, indicating the Tomcat webapps directory where the WAR file is being copied.

Note: Assuming your Jenkins job generated hello-world-war-1.0.0.war in the workspace, this command copies it to Tomcat's webapps directory, enabling deployment and execution.

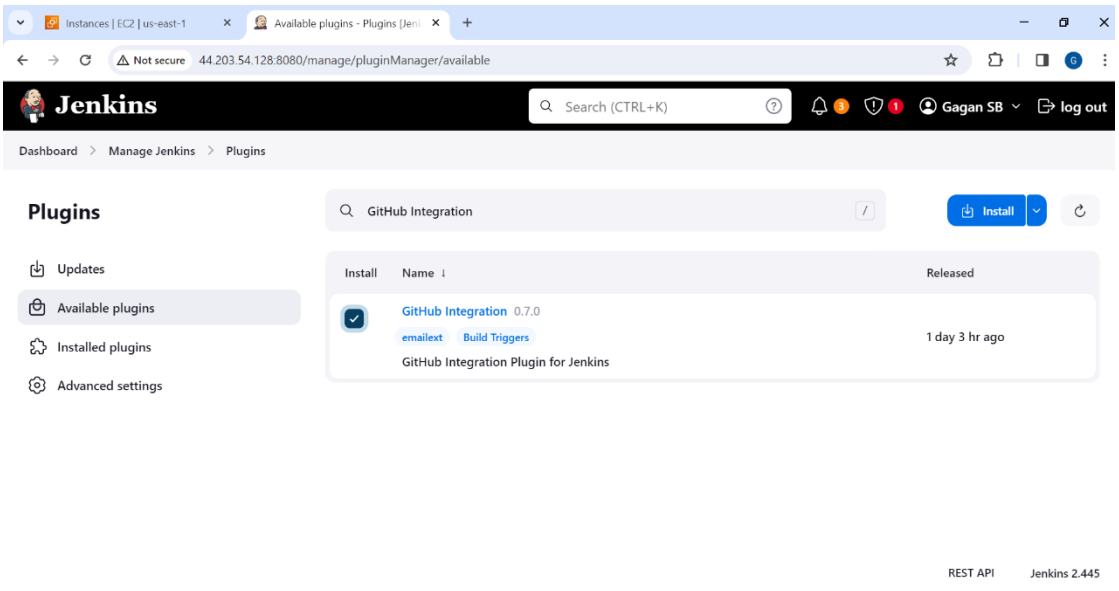
- Access your browser and enter the Jenkins Public IP address, followed by port 9999 and the path (URL: **http://< Jenkins Public IP >:9999/hello-world-war-1.0.0/**).
- Confirm Tomcat is serving your web page.

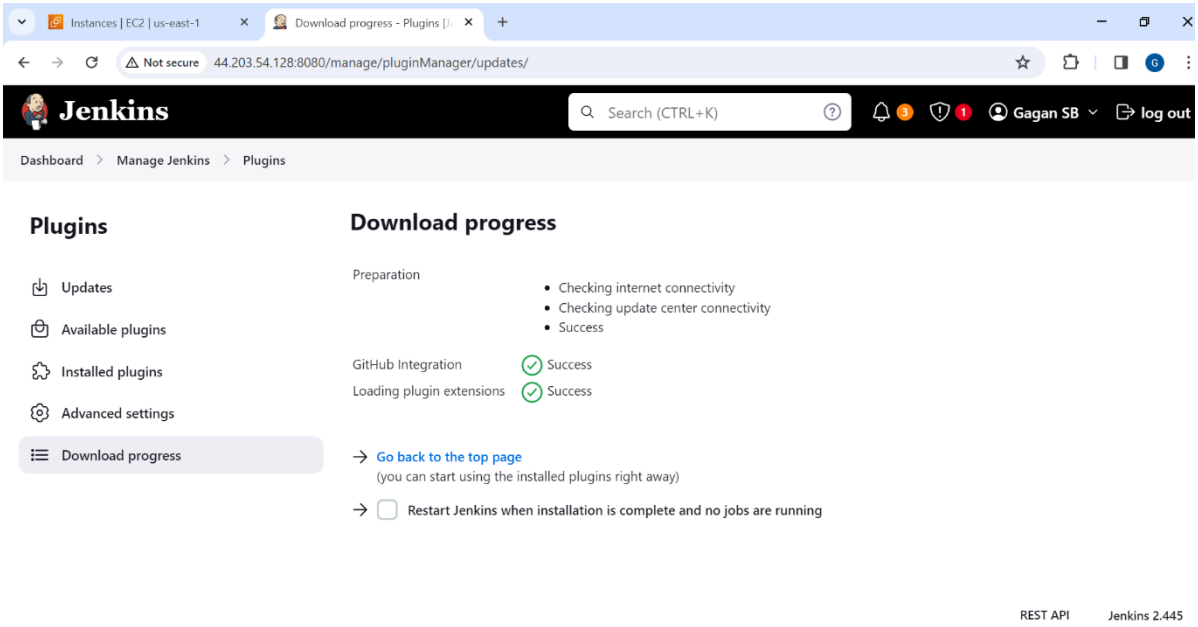
Step11: Using GitWebHook to build your code **automatically using Jenkins**

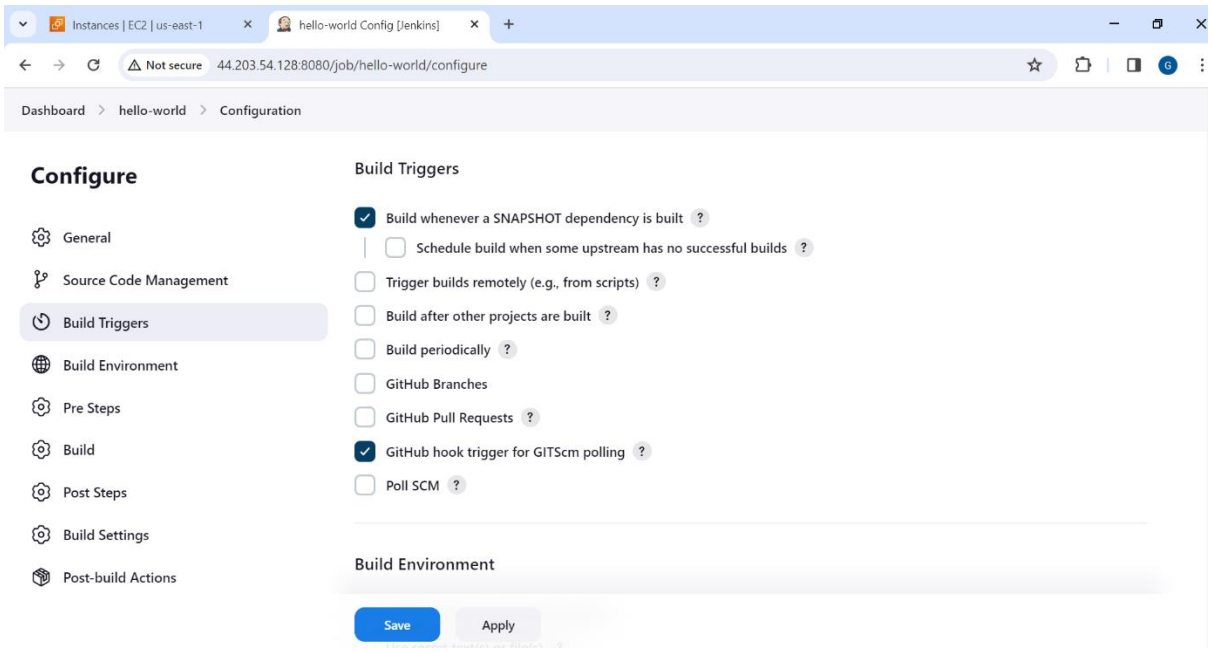
Objective: This step focuses on configuring Git WebHooks to automatically **trigger Jenkins builds** when code changes are pushed to a GitHub repository.

Task-1: Setting up GitHub Integration in Jenkins for Automated Builds

1. Navigate to the Jenkins webpage and select **Manage Jenkins** > **Plugins**.
2. In the **Available** tab, search for **GitHub Integration**. Choose the **GitHub Integration Plugin** and click **Install** (without restart).
3. After installation, return to the top page.
4. In your **hello-world project**, click on **Configure**.
5. Under **Build Triggers**, enable **GitHub hook trigger for GITScm polling** and click **Save**.




Instances | EC2 | us-east-1
Download progress - Plugins
Not secure
44.203.54.128:8080/manage/pluginManager/updates/
Jenkins
Search (CTRL+K)
Gagan SB
log out
Dashboard
Manage Jenkins
Plugins
Plugins
Updates
Available plugins
Installed plugins
Advanced settings
Download progress
Download progress
Preparation
Checking internet connectivity
Checking update center connectivity
Success
GitHub Integration
Success
Loading plugin extensions
Success
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
REST API
Jenkins 2.445

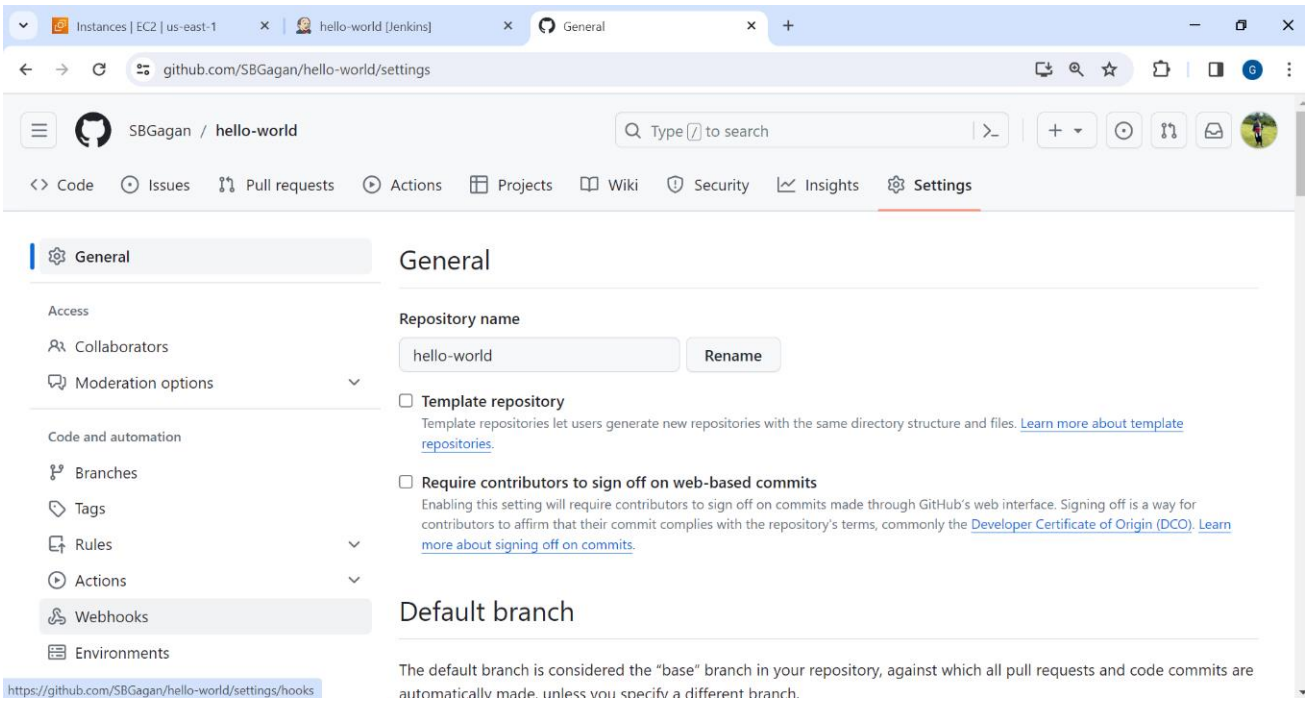

Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
44.203.54.128:8080/job/hello-world/configure
Dashboard
hello-world
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Build Triggers
Build whenever a SNAPSHOT dependency is built
Schedule build when some upstream has no successful builds
Trigger builds remotely (e.g., from scripts)
Build after other projects are built
Build periodically
GitHub Branches
GitHub Pull Requests
GitHub hook trigger for GITScm polling
Poll SCM
Build Environment
Save
Apply

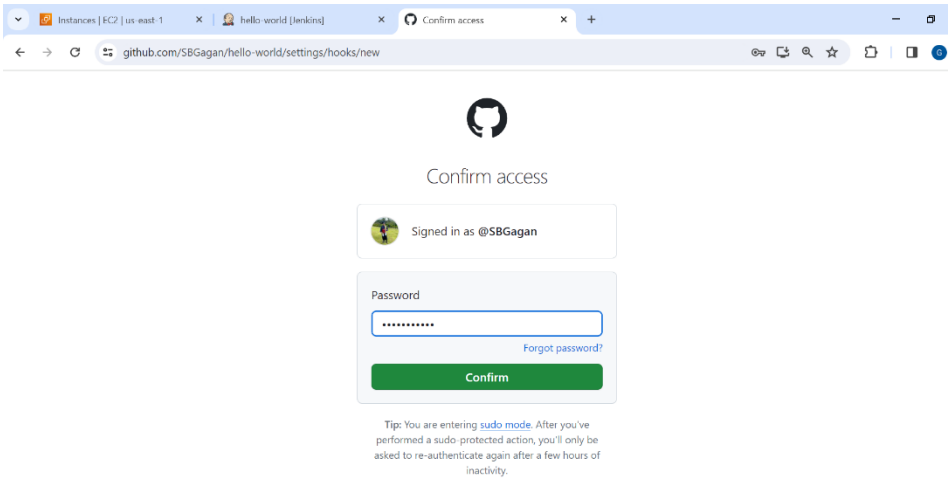
Task-2: Configuring GitHub Webhook in GitHub:

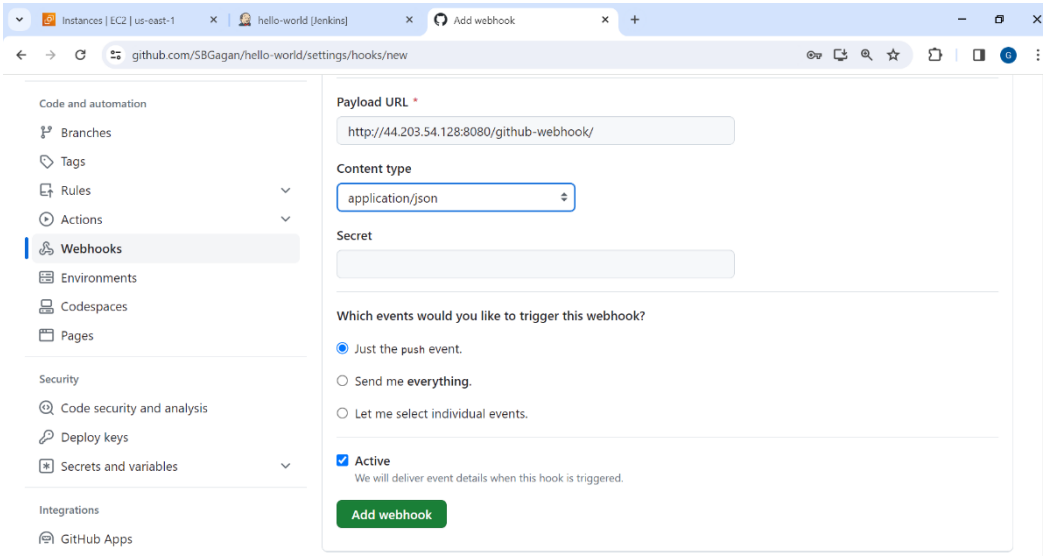
1. On **GitHub website**, go to the **hello-world** repository > **Settings** > **Webhooks**. Click **Add Webhook**.
2. Fill in the details as follows:
    - **Payload URL Example:**

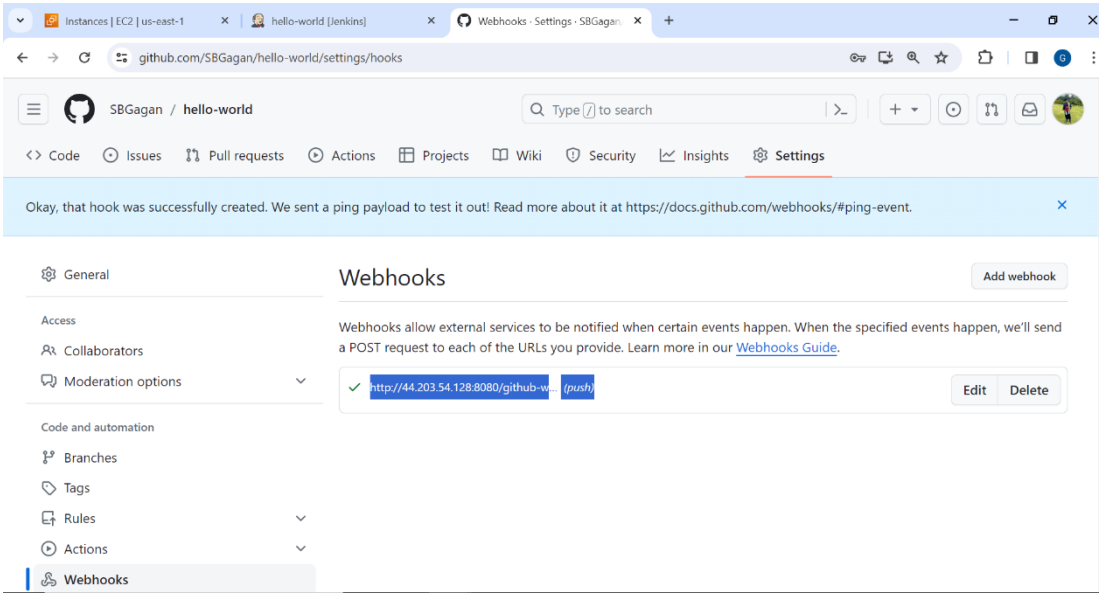
      http://< jenkins-PublicIP >:8080/github-webhook/
      (Example: http://184.72.112.155:8080/github-webhook/)

    - **Content type:** application/JSON
    - Click **Add Webhook**.



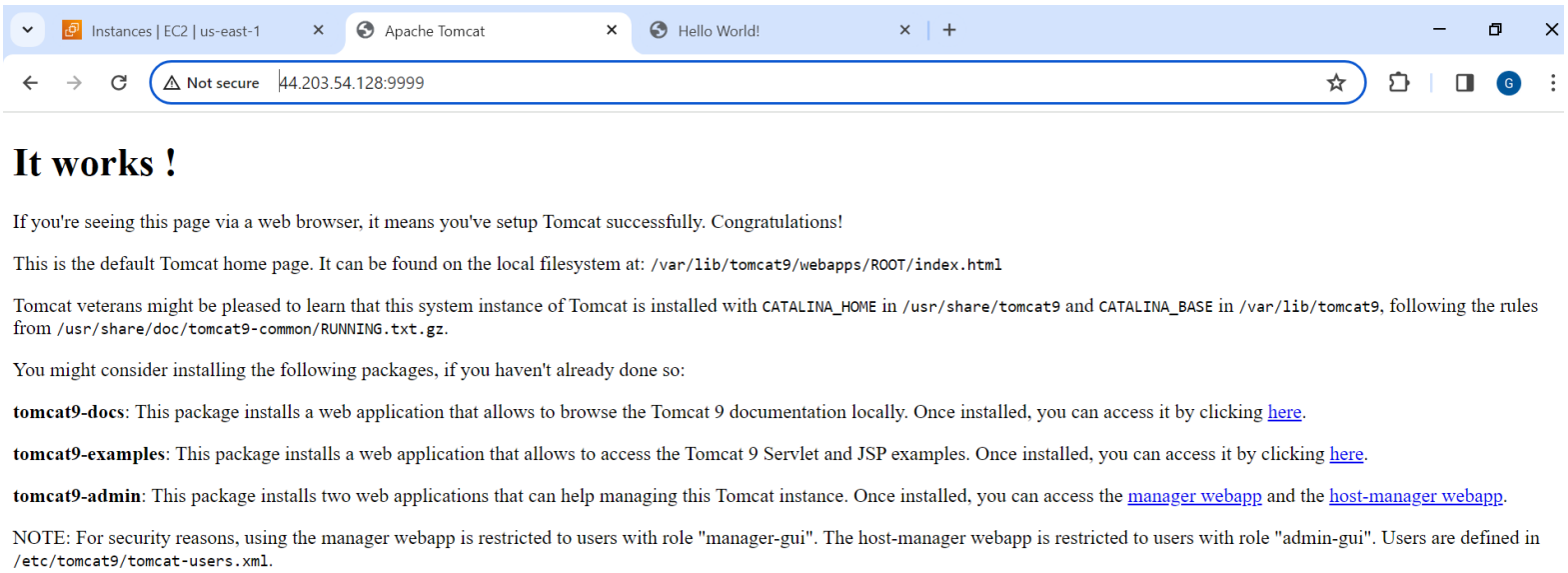

Instances | EC2 | us-east-1
hello-world [Jenkins]
Confirm access
github.com/SBGagan/hello-world/settings/hooks/new
Confirm access
Signed in as @SBGagan
Password
Forgot password?
Confirm
Tip: You are entering sudo mode. After you've performed a sudo-protected action, you'll only be asked to re-authenticate again after a few hours of inactivity.


Instances | EC2 | us-east-1
hello-world [Jenkins]
Add webhook
github.com/SBGagan/hello-world/settings/hooks/new
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Environments
Codespaces
Pages
Security
Code security and analysis
Deploy keys
Secrets and variables
Integrations
GitHub Apps
Payload URL *
http://44.203.54.128:8080/github-webhook/
Content type
application/json
Secret
Which events would you like to trigger this webhook?
Just the push event.
Send me everything.
Let me select individual events.
Active
We will deliver event details when this hook is triggered.
Add webhook

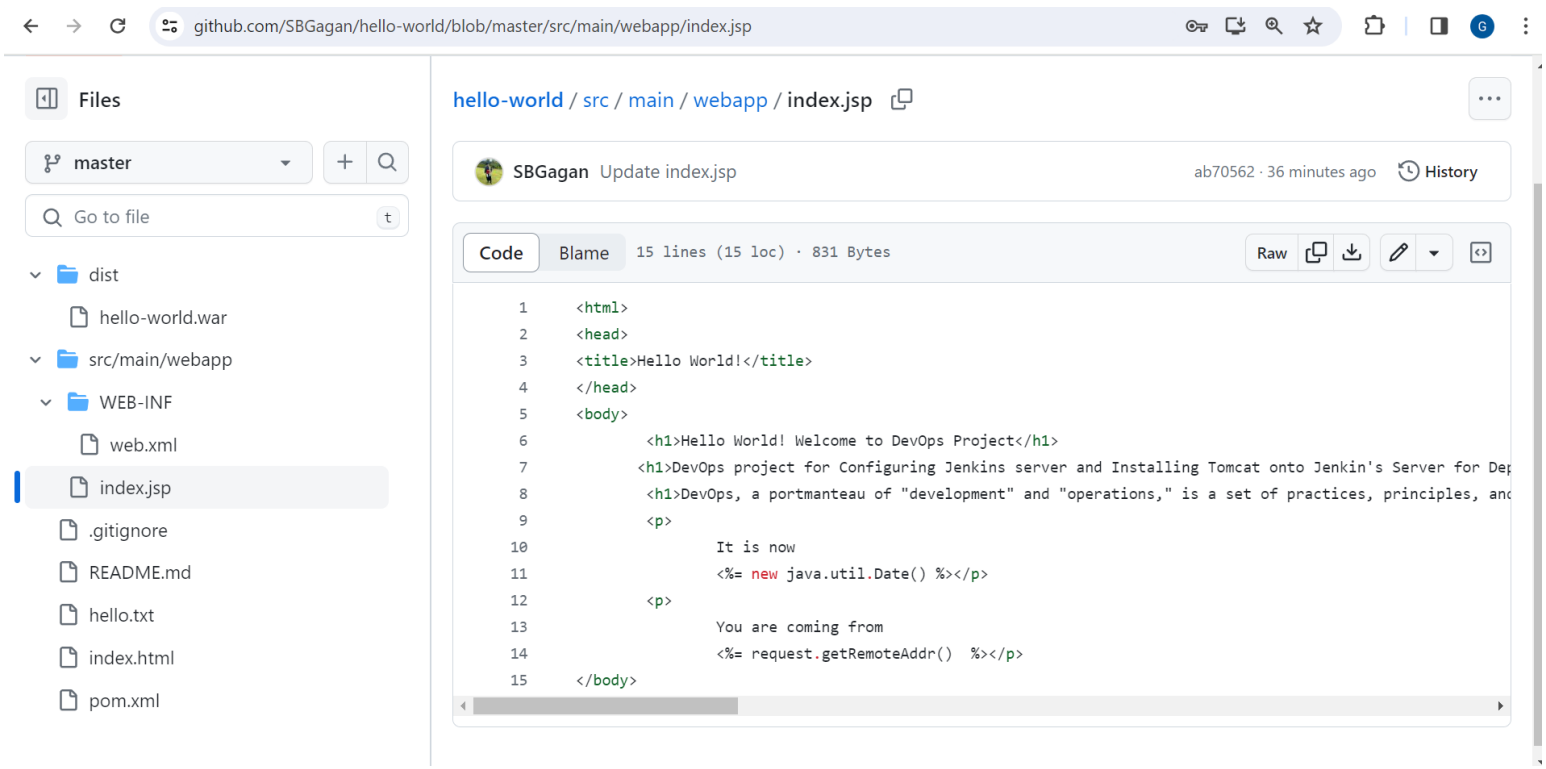

Instances | EC2 | us-east-1
hello-world [Jenkins]
Webhooks · Settings · SBGagan
github.com/SBGagan/hello-world/settings/hooks
SBGagan / hello-world
Type / to search
Code
Issues
Pull requests
Actions
Projects
Wiki
Security
Insights
Settings
Okay, that hook was successfully created. We sent a ping payload to test it out! Read more about it at https://docs.github.com/webhooks/#ping-event.
General
Access
Collaborators
Moderation options
Code and automation
Branches
Tags
Rules
Actions
Webhooks
Webhooks
Add webhook
Webhooks allow external services to be notified when certain events happen. When the specified events happen, we'll send a POST request to each of the URLs you provide. Learn more in our Webhooks Guide.
http://44.203.54.128:8080/github-w... (push)
Edit
Delete

Task-3: Verifying WebHook Functionality:

1. Make a minor change and commit in GitHub's **hello-world repository** by editing the **hello.txt** file.
2. The WebHook triggers Jenkins upon source code modification, initiating an automatic build.
3. Navigate to Jenkins, where you'll observe the ongoing build.
4. Monitor the Jenkins page for the successful completion of the build.

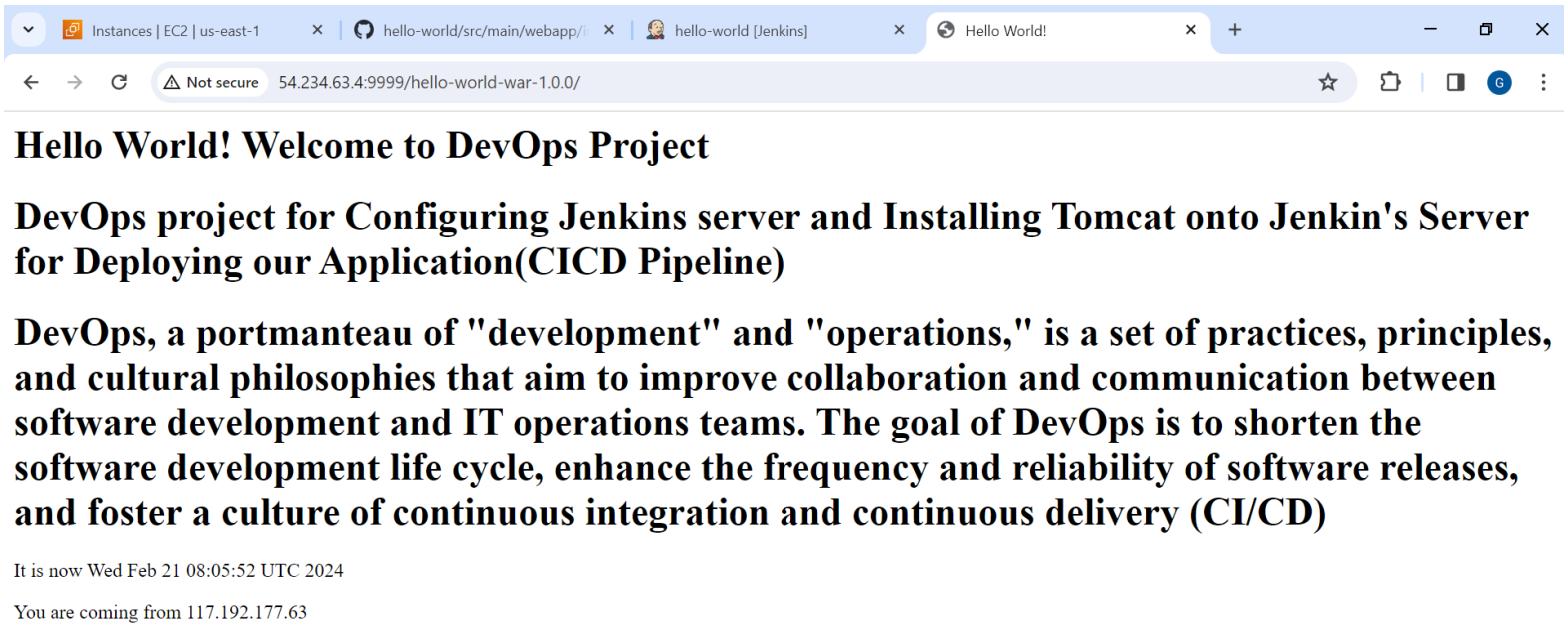


Input of application ---



Output Results: Successfully Deployed our Application onto Tomcat Server Web Page.

Tomcat Server Port Number – 9999 (As we could see below)



DevOps Project-2

DevOps Project for Configuring Docker Machine as Jenkins Slave, build and deploy code in Docker Host as a container.

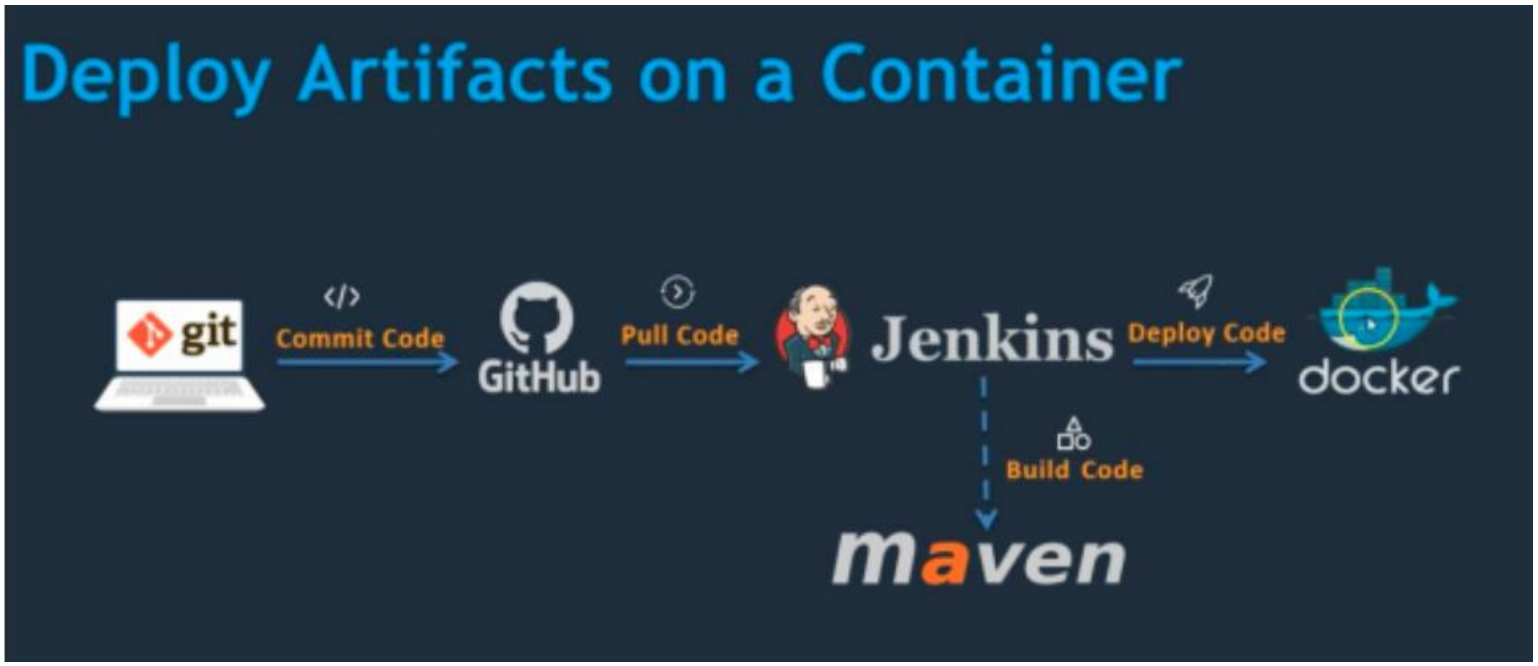
Project-2 Objective: In this Project, you will set up a Docker container as a Jenkins slave, and build a Docker image for a Java web application, and Deploy it in a Docker container.

DevOps Tools used for the Project:

1. AWS Cloud Platform to create Anchor Server
2. Git/ Github
3. Terraform
4. Ansible
5. Jenkins
6. Apache Maven
7. Docker Image and **Docker Container** (To Deploy our Application)

Other Tool Used:

1.MobaXterm - Toolbox for remote computing in order to connect to AWS



STEP-1: Configuring Docker Machine as Jenkins Slave.

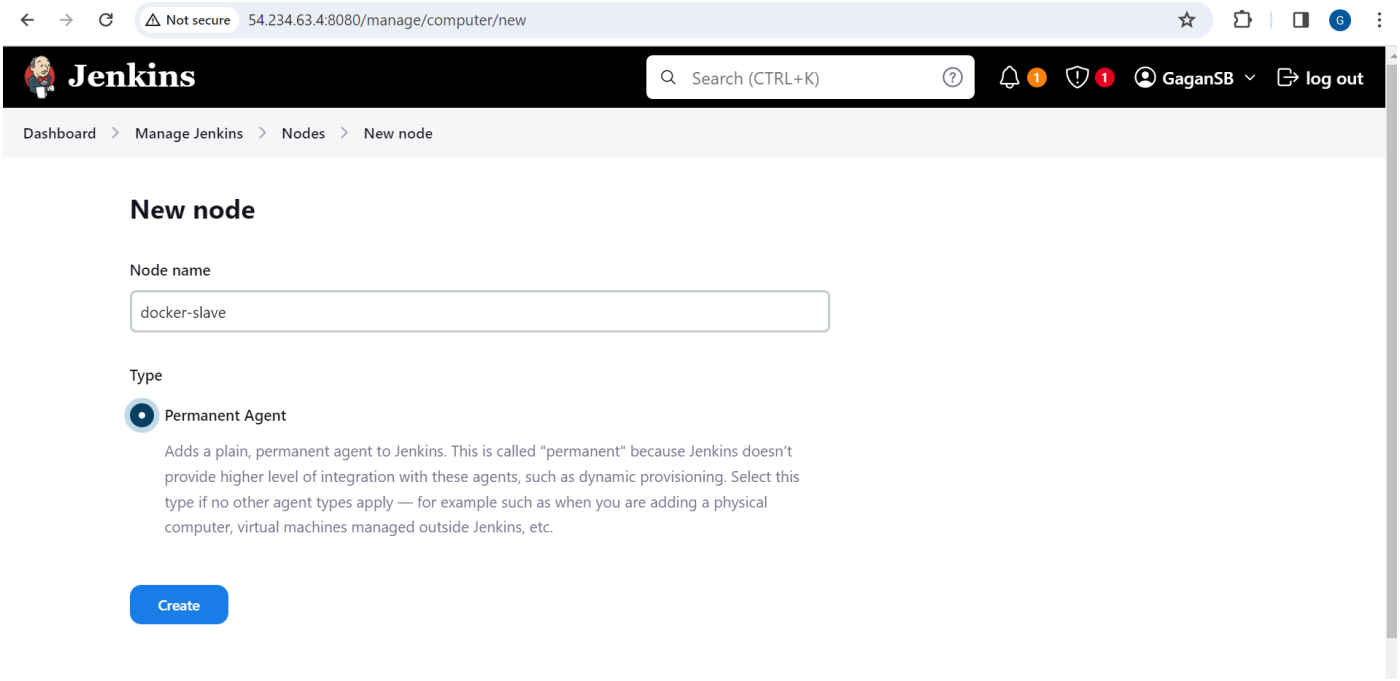
1. In Jenkins Webpage, Navigate to **Jenkin's Dashboard** and click on the **Manage Jenkins** and **Nodes**.

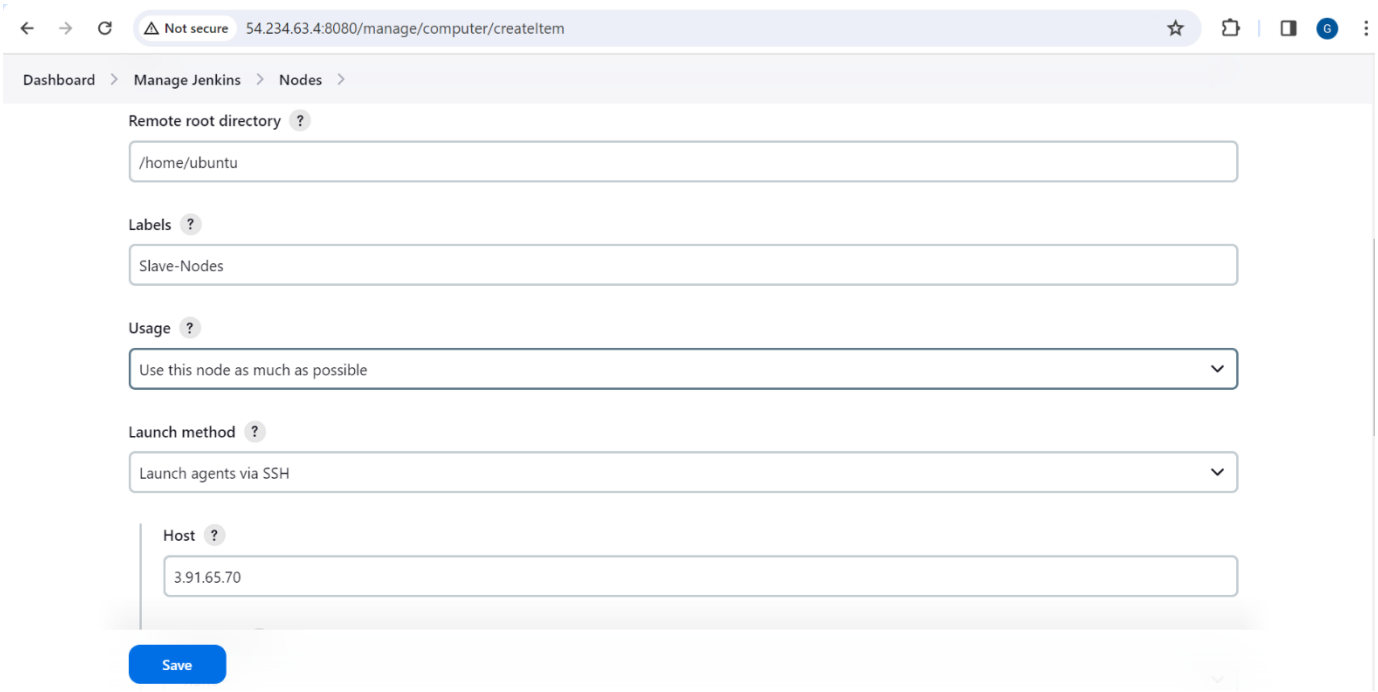
2. Click on **New Node** in the next window. Give the node name as docker-slave and Select Permanent Agent

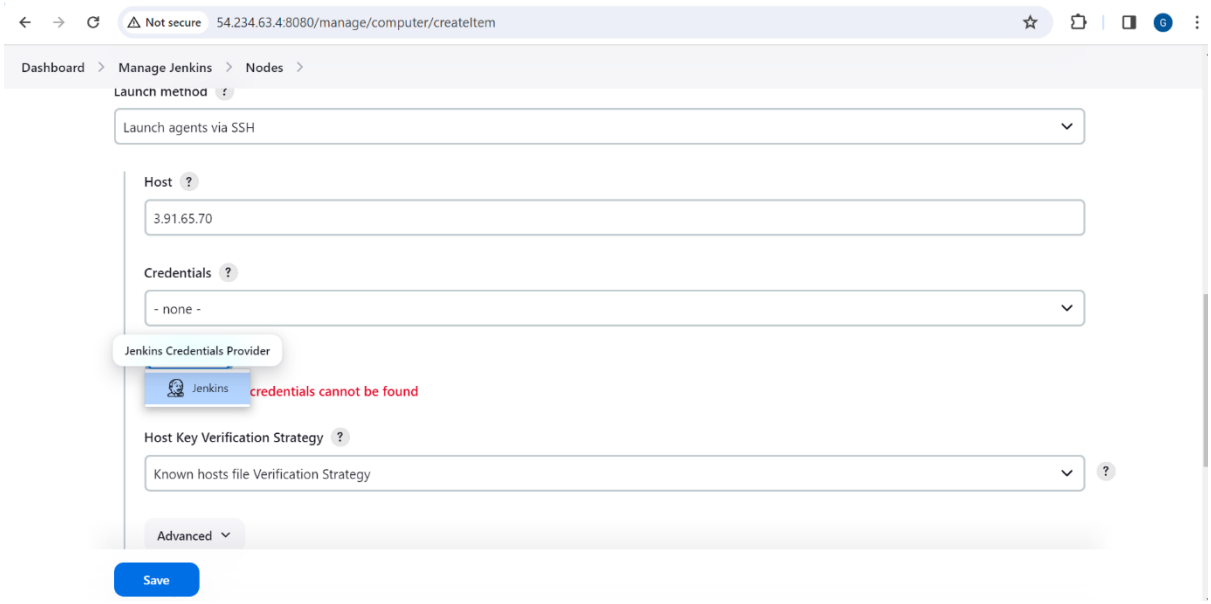
3. Fill out the details for the node docker-slave as given below.

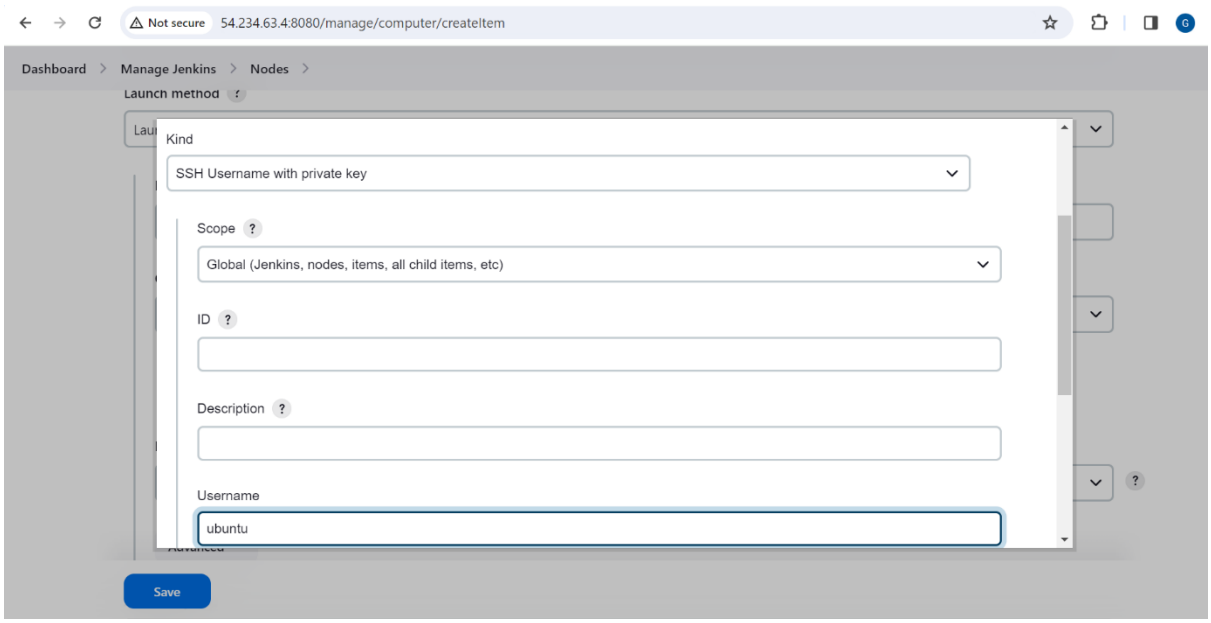
    - The name should be given as **docker-slave**,
    - Remote Root Directory as **/home/ubuntu**,
    - labels to be **Slave-Nodes**,
    - usage to be given as **"use this node as much as possible"**
    - Launch method to be set as **"launch agents via SSH"**.
    - In the host section, give the **Public IP of the Docker instance**.
    - For Credentials for this Docker node, click on the dropdown button named **Add** and then click on **Jenkins**;
    - Then in the next window, in kind select **SSH username with private key** (Give username as ubuntu),
    - In **Private Key** Select **Enter directly**

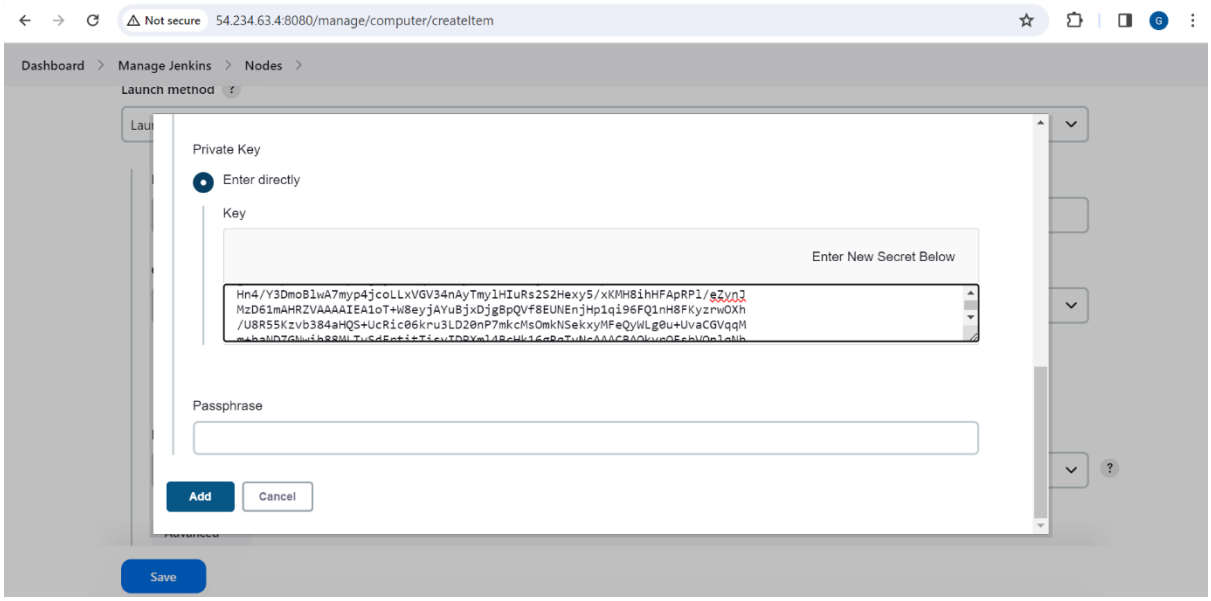
    Note: To get the Private Key go to Anchor Server (Not Jenkins/Docker) and Execute the below command:

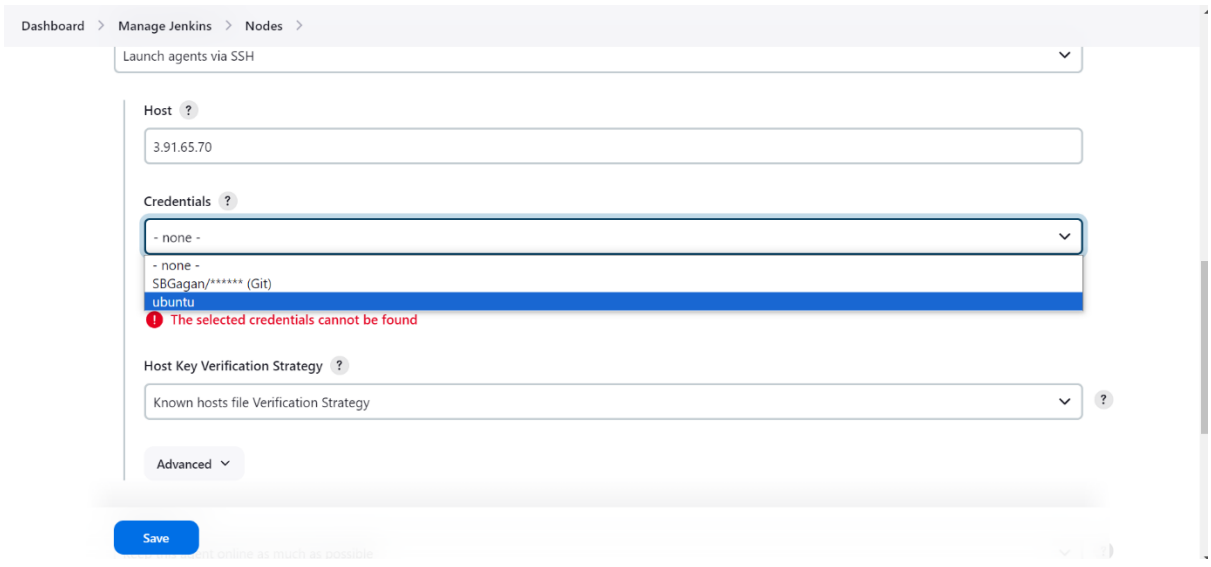

Not secure
54.234.63.4:8080/manage/computer/new
Jenkins
Search (CTRL+K)
GaganSB
log out
Dashboard
Manage Jenkins
Nodes
New node
New node
Node name
docker-slave
Type
Permanent Agent
Adds a plain, permanent agent to Jenkins. This is called "permanent" because Jenkins doesn't provide higher level of integration with these agents, such as dynamic provisioning. Select this type if no other agent types apply — for example such as when you are adding a physical computer, virtual machines managed outside Jenkins, etc.
Create

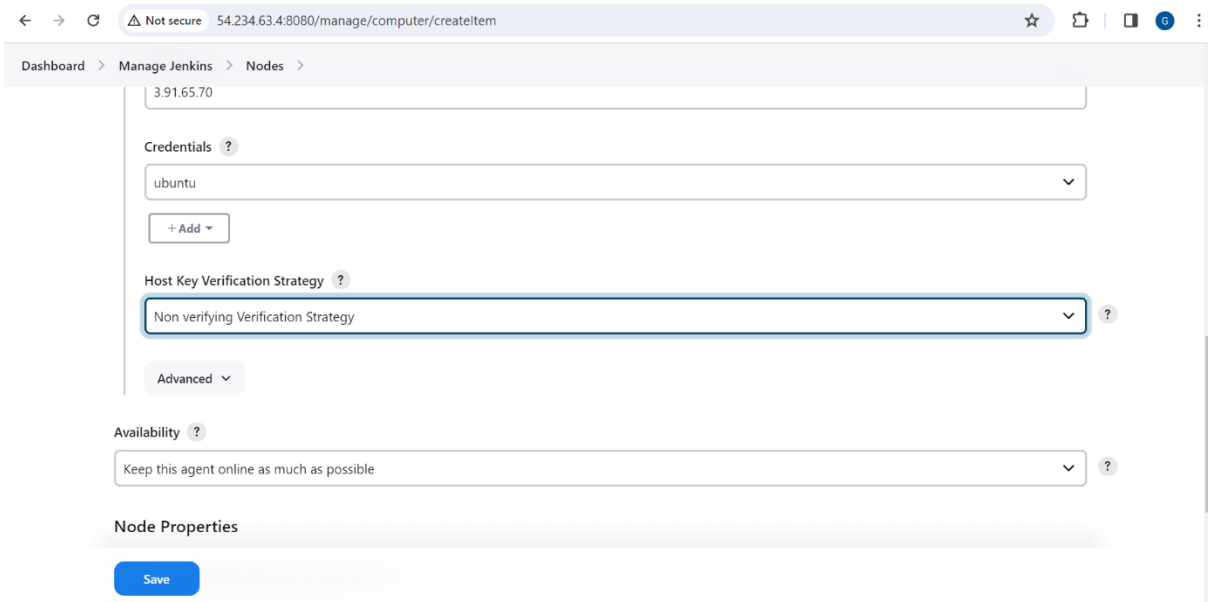

Not secure
54.234.63.4:8080/manage/computer/createItem
Dashboard
Manage Jenkins
Nodes
Remote root directory
/home/ubuntu
Labels
Slave-Nodes
Usage
Use this node as much as possible
Launch method
Launch agents via SSH
Host
3.91.65.70
Save

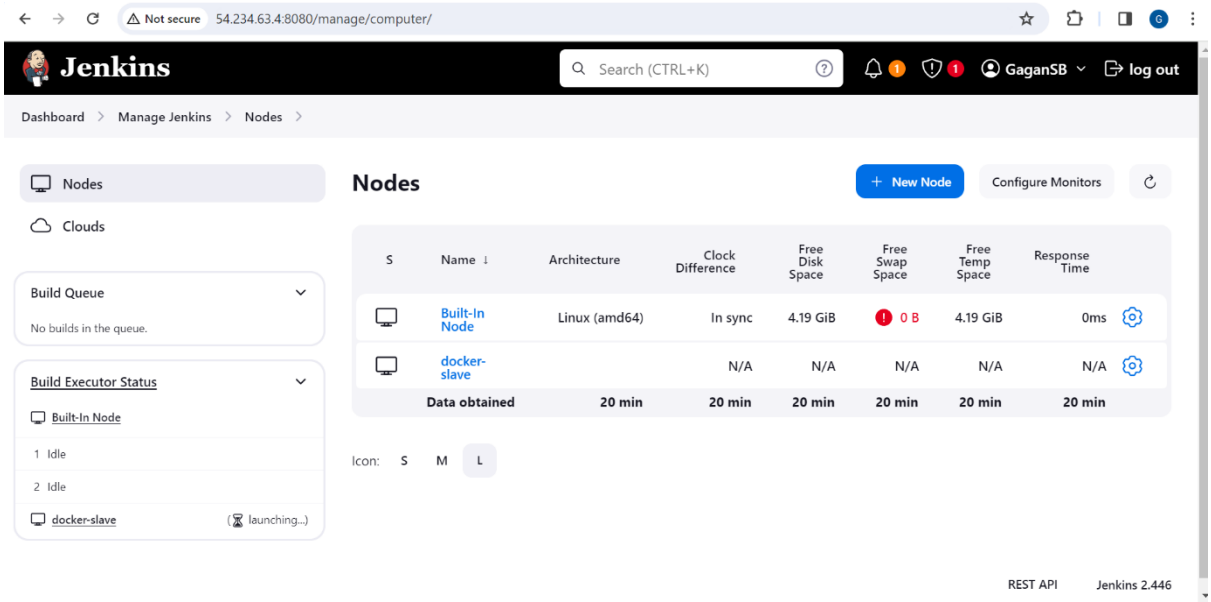

Not secure
54.234.63.4:8080/manage/computer/createItem
Dashboard
Manage Jenkins
Nodes
Launch method
Launch agents via SSH
Host
3.91.65.70
Credentials
- none -
Jenkins Credentials Provider
Jenkins
credentials cannot be found
Host Key Verification Strategy
Known hosts file Verification Strategy
Advanced
Save


Not secure
54.234.63.4:8080/manage/computer/createItem
Dashboard
Manage Jenkins
Nodes
Launch method
Kind
SSH Username with private key
Scope
Global (Jenkins, nodes, items, all child items, etc)
ID
Description
Username
ubuntu
Save


Not secure
54.234.63.4:8080/manage/computer/createItem
Dashboard
Manage Jenkins
Nodes
Launch method
Private Key
Enter directly
Key
Enter New Secret Below
Hn4/Y3DmoBlwA7myp4jcoLLxVGV34nAyTmylHIuRs2S2Hexy5/xKMH8ihHFApRPl/eZvnJ
MzD61mAHRZVAAAAIEA1oT+W8eyjAYuBjxDjgBpQVf8EUNEnjHp1qi96FQ1nH8FKyzrwOXh
/U8R55Kzvb384aHQS+UcRic06kru3LD20nP7mkcMsOmkNSekxyMFeQyWLg0u+UvaCGVqqM
Passphrase
Add
Cancel
Save


Dashboard
Manage Jenkins
Nodes
Launch agents via SSH
Host
3.91.65.70
Credentials
- none -
- none -
SBGagan/****** (Git)
ubuntu
The selected credentials cannot be found
Host Key Verification Strategy
Known hosts file Verification Strategy
Advanced
Save

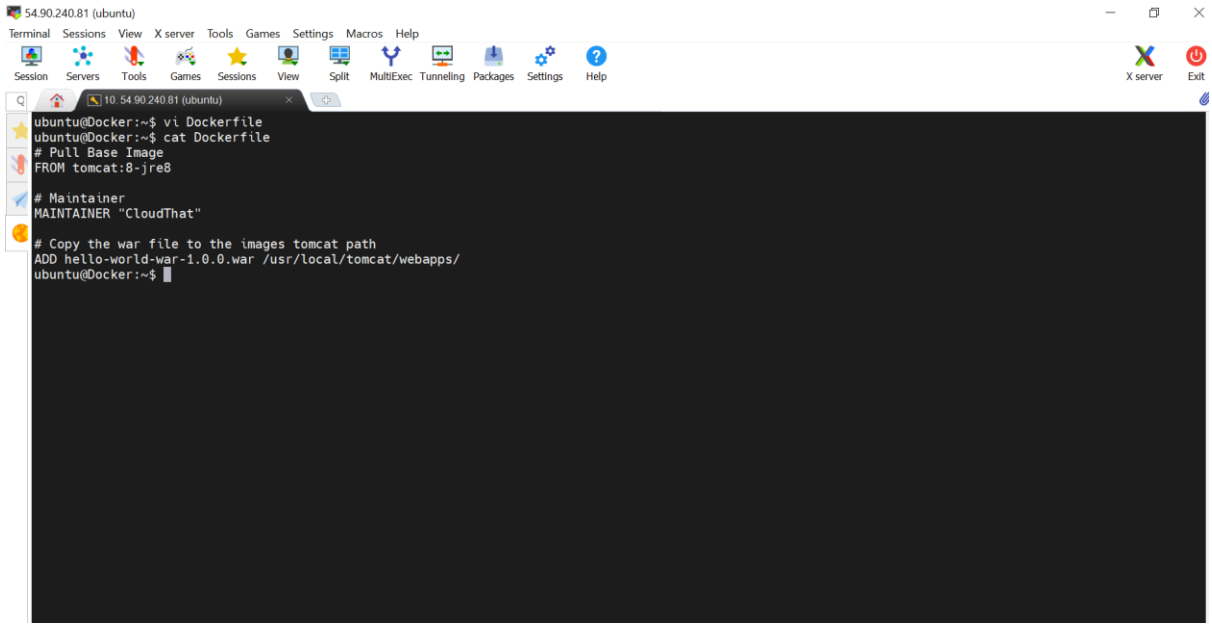

Not secure
54.234.63.4:8080/manage/computer/createItem
Dashboard
Manage Jenkins
Nodes
3.91.65.70
Credentials
ubuntu
+ Add
Host Key Verification Strategy
Non verifying Verification Strategy
Advanced
Availability
Keep this agent online as much as possible
Node Properties
Save


Not secure
54.234.63.4:8080/manage/computer/
Jenkins
Search (CTRL+K)
GaganSB
log out
Dashboard
Manage Jenkins
Nodes
Nodes
Clouds
Build Queue
No builds in the queue.
Build Executor Status
Built-In Node
1 Idle
2 Idle
docker-slave
launching...
Nodes
+ New Node
Configure Monitors
S
Name
Architecture
Clock Difference
Free Disk Space
Free Swap Space
Free Temp Space
Response Time
Built-In Node
Linux (amd64)
In sync
4.19 GiB
0 B
4.19 GiB
0ms
docker-slave
N/A
N/A
N/A
N/A
N/A
Data obtained
20 min
20 min
20 min
20 min
20 min
20 min
Icon: S M L
REST API
Jenkins 2.446

Task-3: Build and deploy code in Docker Host on the container.

Once the Slave Node is Online, then continue the below.

Now, In the CLI, SSH into your Docker Host and execute the following steps to create a "Dockerfile" in the /home/ubuntu directory.



vi Dockerfile

Pull Base Image

FROM tomcat:8-jre8

Maintainer

MAINTAINER "CloudThat"

Copy the war file to the images tomcat path

ADD hello-world-war-1.0.0.war /usr/local/tomcat/webapps/

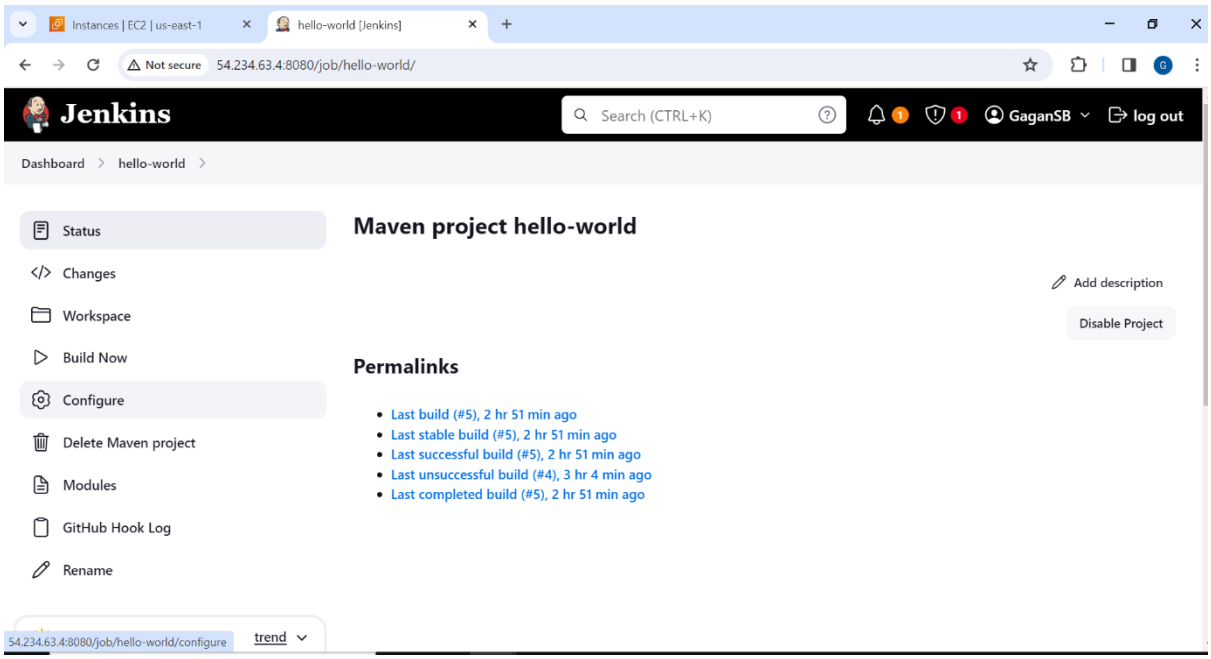
Explanatory Notes of Above Docker FIle:
The provided Dockerfile is a basic configuration for deploying a Java web application using the Tomcat web server. Here's the breakdown of each part:

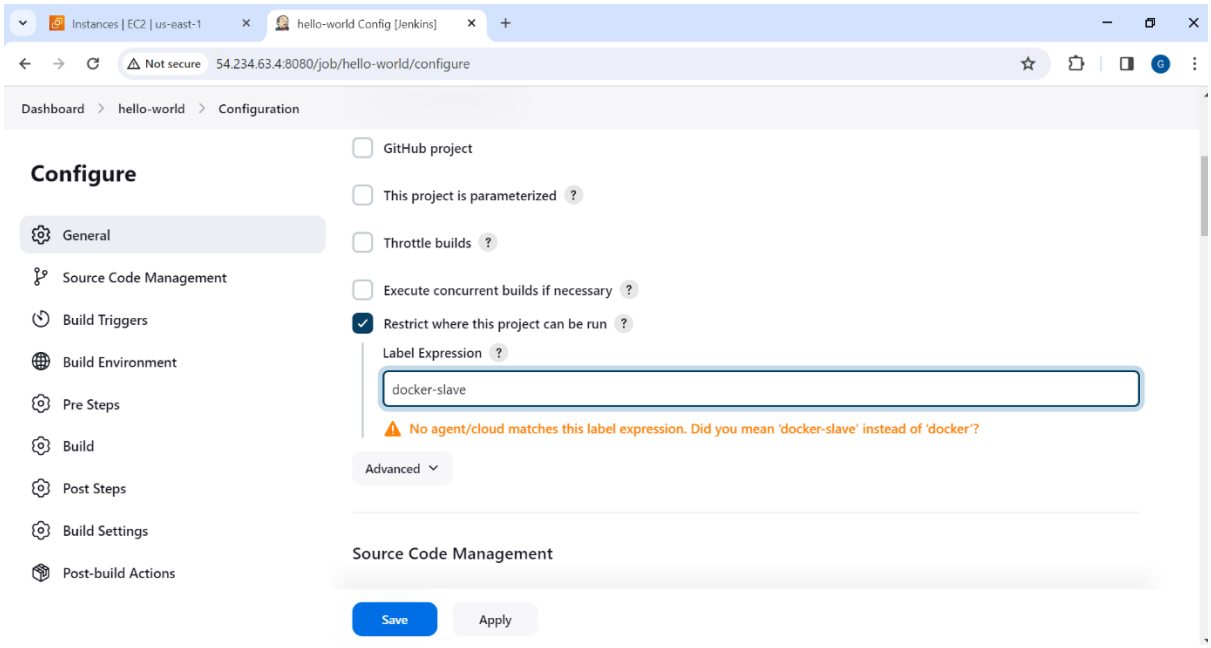
1. FROM tomcat:8-jre8: Specifies the base image for the Docker container, utilizing the official Tomcat 8 image with Java 8 runtime. This image includes the Tomcat web server and Java runtime environment.
2. MAINTAINER "CloudThat": A comment indicating the maintainer or author of the Dockerfile.
3. ADD hello-world-war-1.0.0.war /usr/local/tomcat/webapps/: Copies the Java web application's WAR file (hello-world-war-1.0.0.war) into the /usr/local/tomcat/webapps/ directory within the container. Upon Tomcat startup, the WAR file is automatically deployed as a web application.

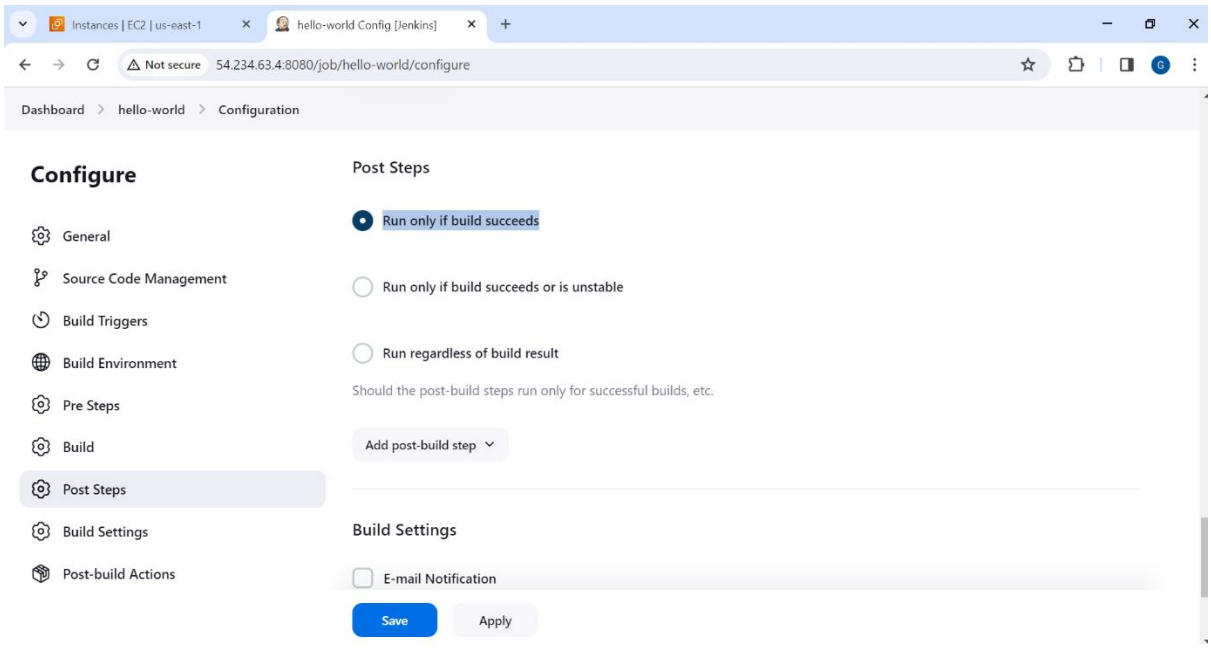
This Dockerfile packages a Java web app in a container using a Tomcat base image. Customize it for your needs. After building and running a container, your app should be accessible via Tomcat on port 8080, following Jenkins job steps.

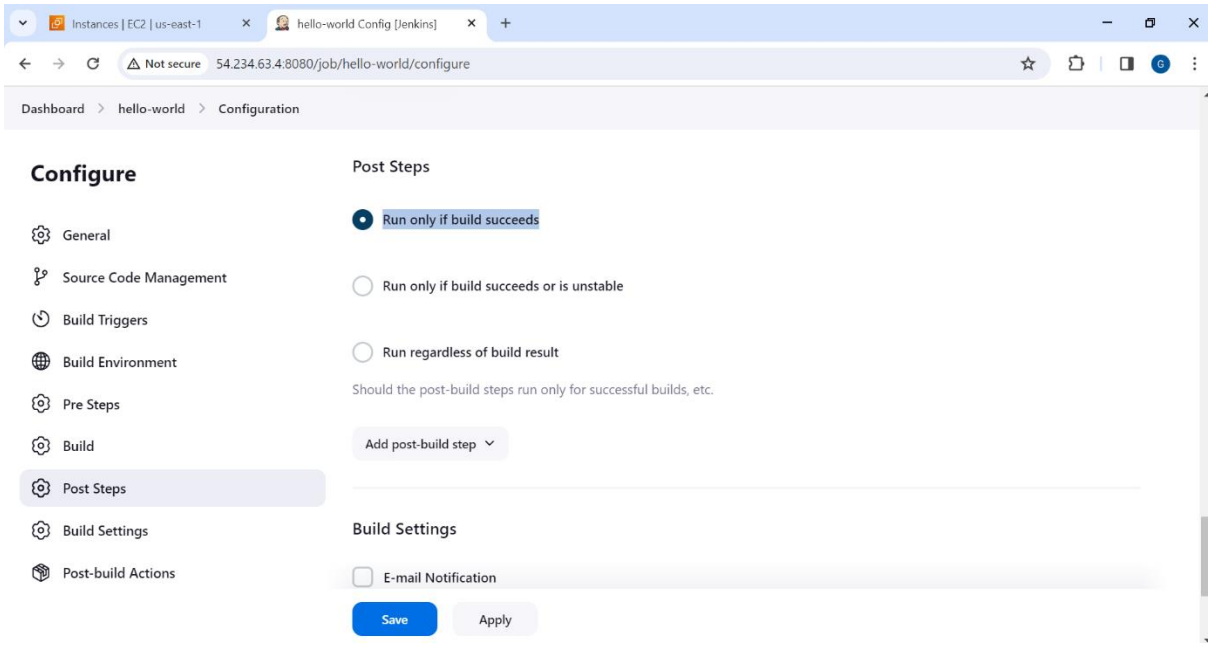
Task-4:

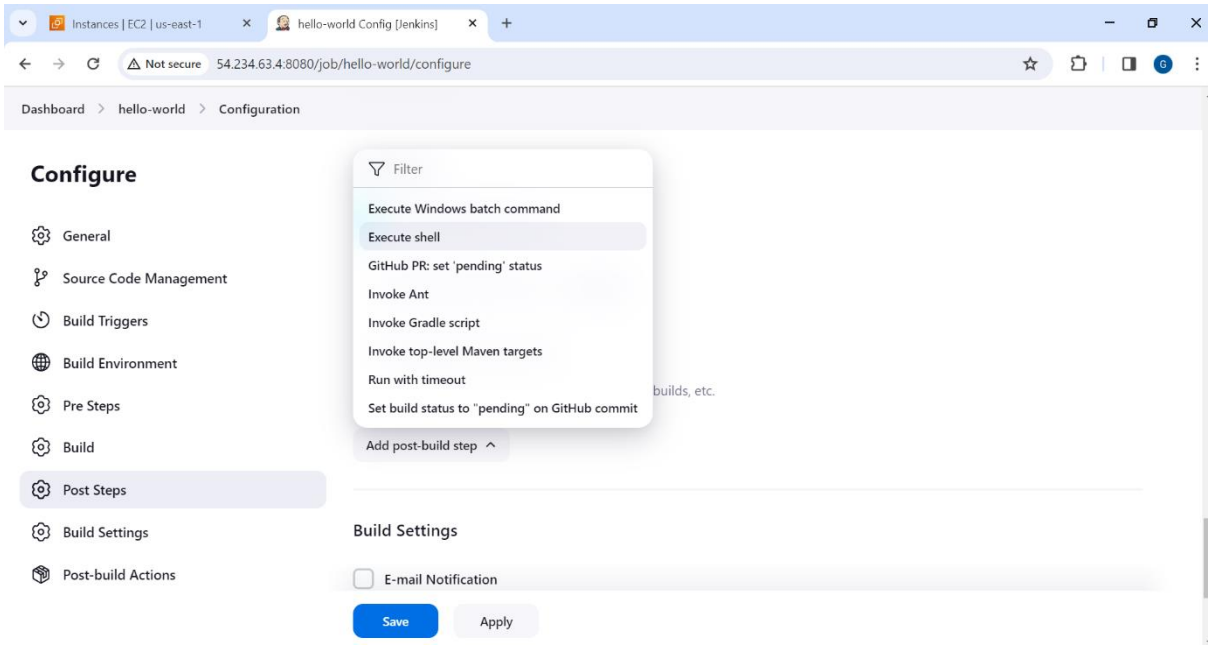
1. Navigate to your **Jenkins Home page**, choose the **hello-world project** from the drop-down, and click on the Configure tab.
2. In the **General Tab**, choose **Restrict where this project can be run** and set the Label Expression to **Slave-Nodes.**
3. Move to the **Post Build Steps Tab**, select **"Run only if the build succeeds,"** and then select add a post-build step, choose **Execute shell** from the drop-down, paste the provided commands (below) into the shell, and click **Save.**

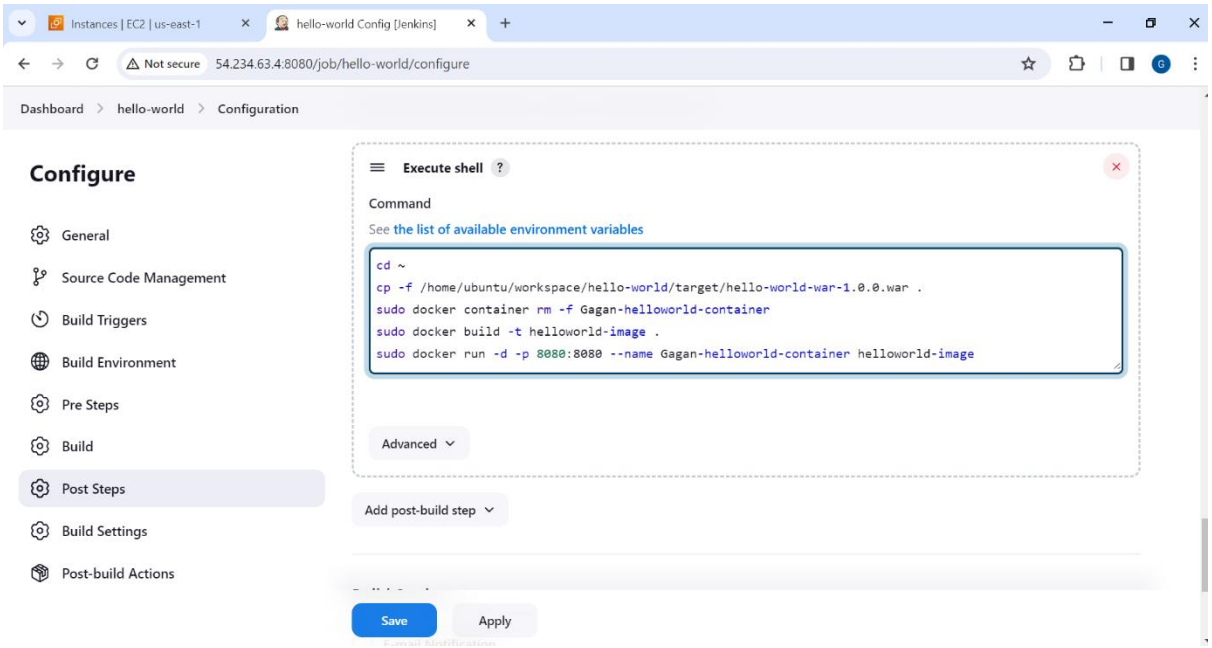



Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
54.234.63.4:8080/job/hello-world/configure
Dashboard
hello-world
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
GitHub project
This project is parameterized
Throttle builds
Execute concurrent builds if necessary
Restrict where this project can be run
Label Expression
docker-slave
No agent/cloud matches this label expression. Did you mean 'docker-slave' instead of 'docker'?
Advanced
Source Code Management
Save
Apply


Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
54.234.63.4:8080/job/hello-world/configure
Dashboard
hello-world
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Post Steps
Run only if build succeeds
Run only if build succeeds or is unstable
Run regardless of build result
Should the post-build steps run only for successful builds, etc.
Add post-build step
Build Settings
E-mail Notification
Save
Apply


Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
54.234.63.4:8080/job/hello-world/configure
Dashboard
hello-world
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Post Steps
Run only if build succeeds
Run only if build succeeds or is unstable
Run regardless of build result
Should the post-build steps run only for successful builds, etc.
Add post-build step
Build Settings
E-mail Notification
Save
Apply


Instances | EC2 | us-east-1
hello-world Config [Jenkins]
Not secure
54.234.63.4:8080/job/hello-world/configure
Dashboard
hello-world
Configuration
Configure
General
Source Code Management
Build Triggers
Build Environment
Pre Steps
Build
Post Steps
Build Settings
Post-build Actions
Filter
Execute Windows batch command
Execute shell
GitHub PR: set 'pending' status
Invoke Ant
Invoke Gradle script
Invoke top-level Maven targets
Run with timeout
Set build status to "pending" on GitHub commit
builds, etc.
Add post-build step
Build Settings
E-mail Notification
Save
Apply



cd ~

cp -f /home/ubuntu/workspace/hello-world/target/hello-world-war-1.0.0.war .

sudo docker container rm -f yourname-helloworld-container

sudo docker build -t helloworld-image .

sudo docker run -d -p 8080:8080 --name yourname-helloworld-container helloworld-image

1. cd ~: Change the working directory to the user's home directory.
2. cp -f /home/ubuntu/workspace/hello-world/target/hello-world-war-1.0.0.war .: Copy the WAR file (presumably the artifact of your Java web application) from the Jenkins workspace to the current directory (~), where you'll perform the Docker build.
3. sudo docker container rm -f yourname-helloworld-container: Remove any existing Docker container with the name "yourname-helloworld-container" forcefully if it exists. You should replace "yourname" with your actual first name.
4. sudo docker build -t helloworld-image .: Build a Docker image with the tag "helloworld-image" based on the Dockerfile located in the current directory (.). The Dockerfile you created earlier specifies how the image should be built.
5. sudo docker run -d -p 8080:8080 --name yourname-helloworld-container helloworld-image: Run a Docker container named "yourname-helloworld-container" from the "helloworld-image" image. This container will be detached (-d) and will map port 8080 on the host to port 8080 inside the container. You should replace "yourname" with your actual first name.

Upon executing these commands, your Java web application should be deployed within a Docker container, accessible on port 8080 of your Docker host.

Task-5: Building the hello-world project

Either:

1. Manually click on **"Build Now"** in Jenkins.
2. Make a small change in GitHub files.

After a successful build, access the Tomcat server page

Output Results: Java web application is deployed within a Docker container, accessible on port 8080 of our Docker host