System Provisioning and Configuration management

What we will do:

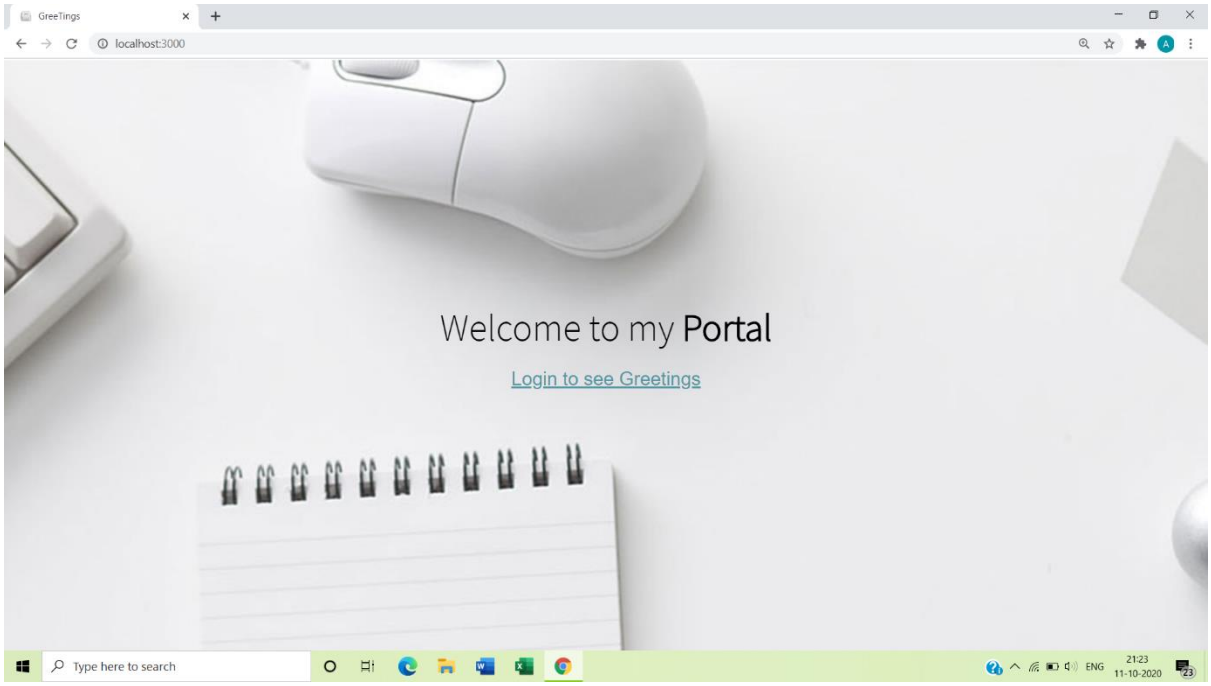
1. Create a small MERN Stack application, i.e. a web application.
2. Push this application on a specific folder of given GitHub repository.
3. Create a job in Jenkins and make a build of this application.
4. Using Jenkins and Docker Plugin, we will create an Image of developed web application.
5. Finally, we will use a Terraform script to deploy the image on AWS Docker platform.

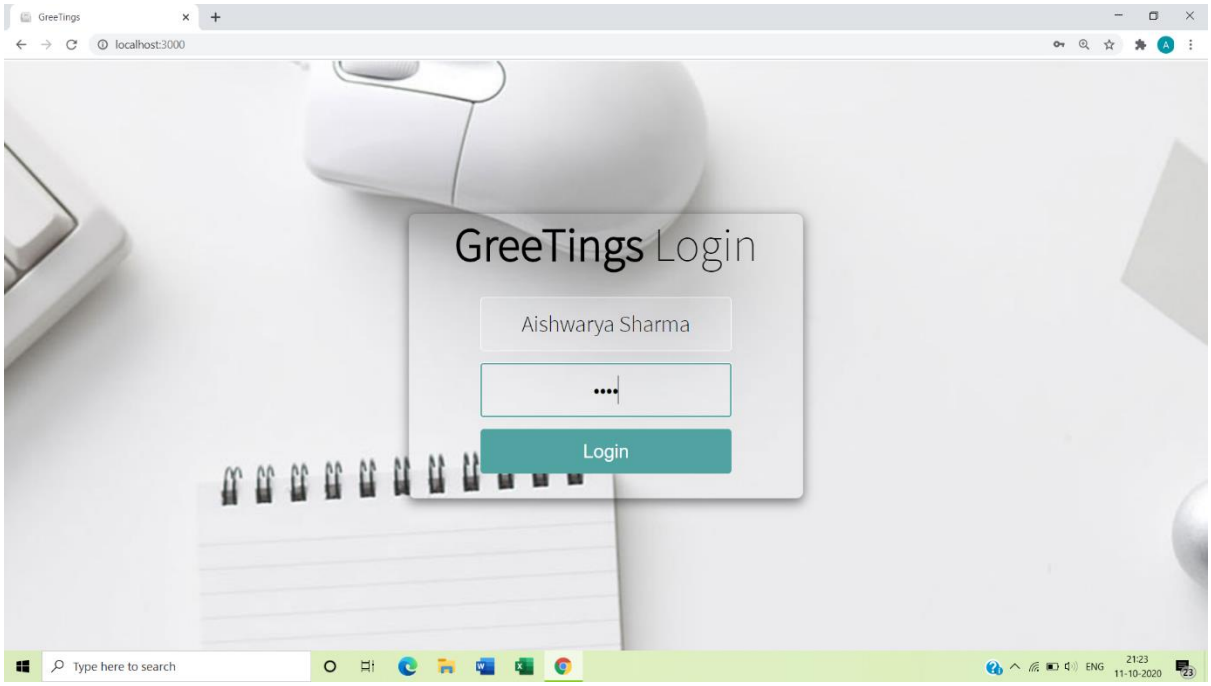
Creating MERN Stack Application

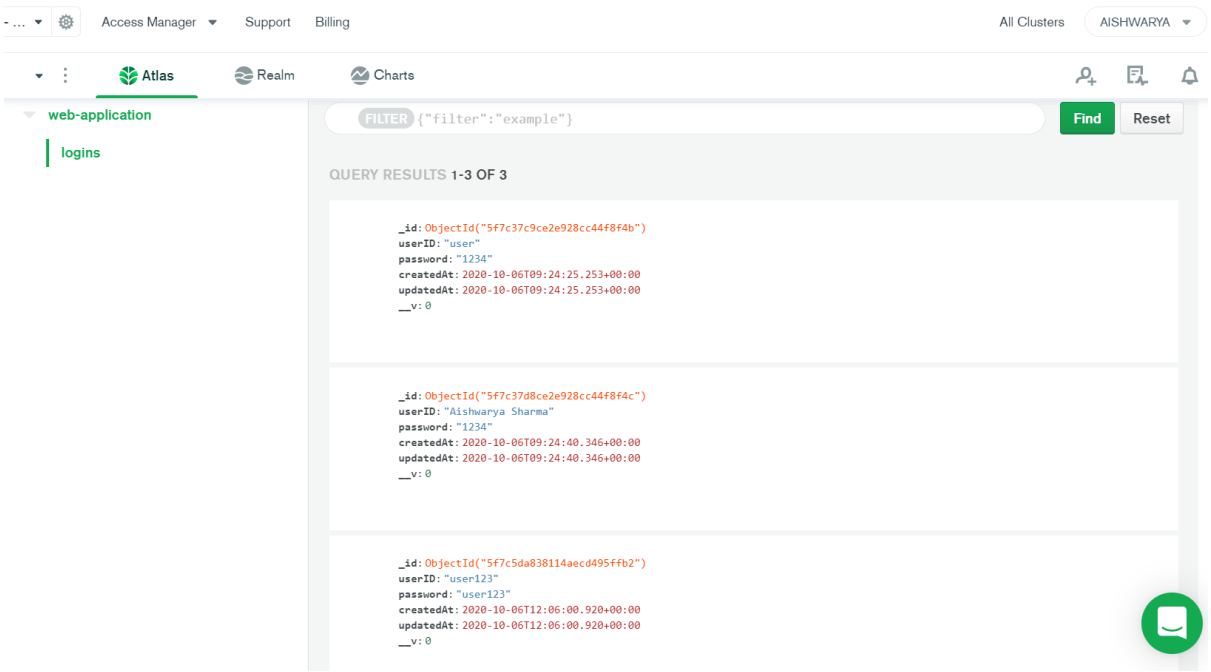
Here is the URL for the complete source code.

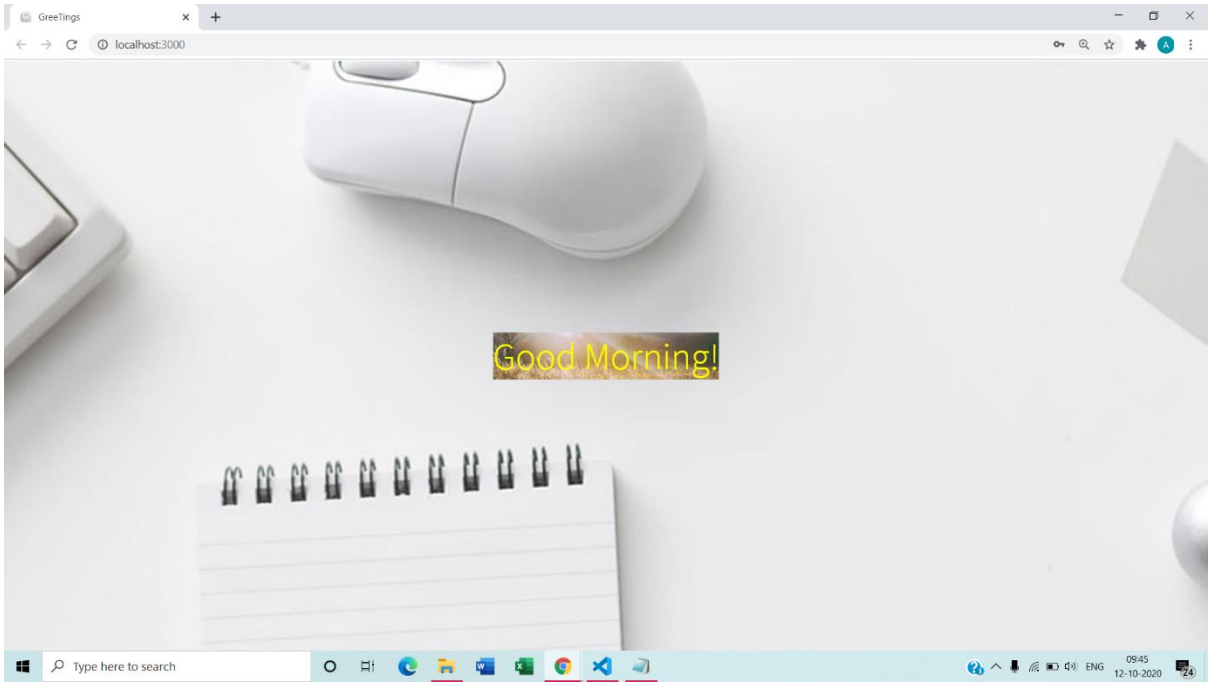
https://github.com/7Aishwarya/Simple-Web-Application1.git

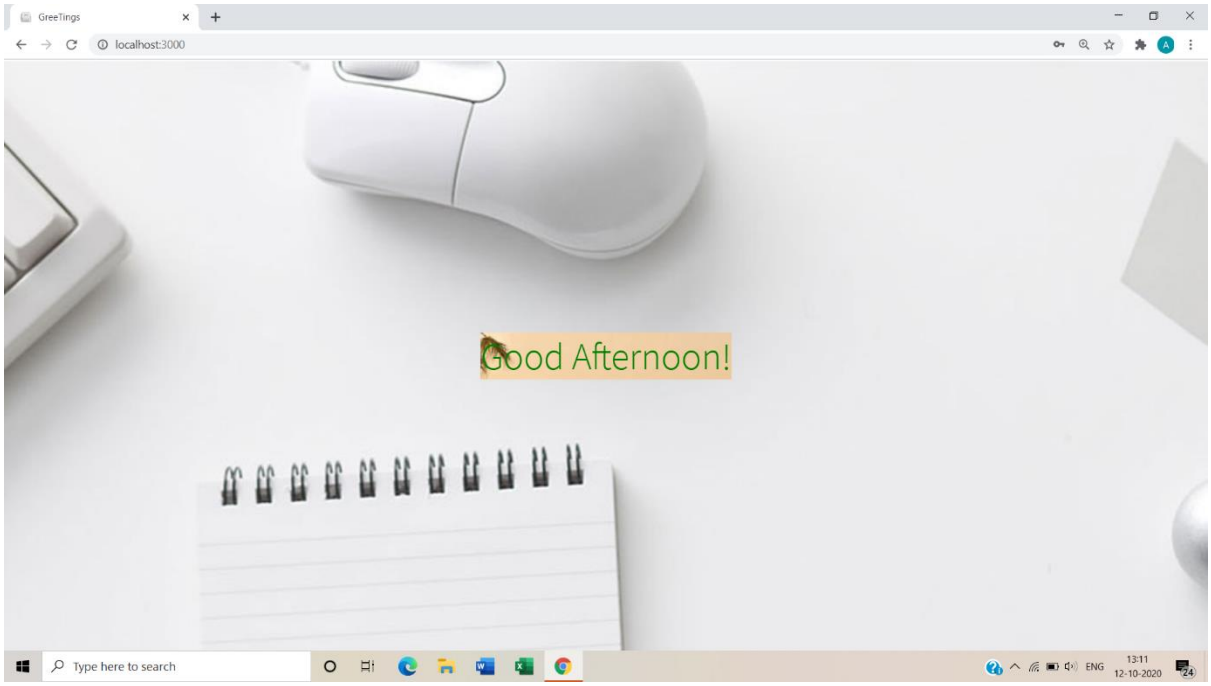
To directly use the application, clone it to your local repo and follow the below steps:
git clone "https://github.com/7Aishwarya/Simple-Web-Application1.git"
Go to the directory and type the following commands in terminal.
npm install
npm start
Open new terminal, and then
cd backend
nodemon server

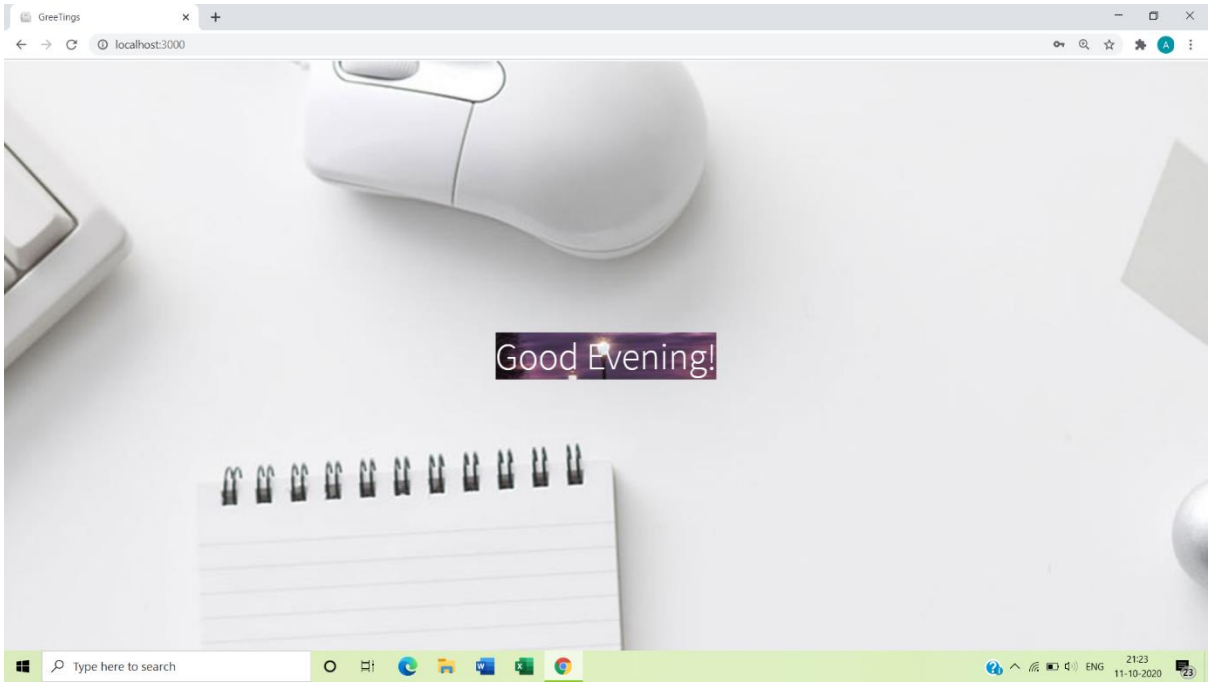



GreeTings
localhost:3000
GreeTings Login
Aishwarya Sharma
Login
Type here to search
ENG
21:23
11-10-2020


Access Manager
Support
Billing
All Clusters
AISHWARYA
Atlas
Realm
Charts
web-application
logins
FILTER {"filter":"example"}
Find
Reset
QUERY RESULTS 1-3 OF 3
_id: ObjectId("5f7c37c9ce2e928cc44f8f4b")
userID: "user"
password: "1234"
createdAt: 2020-10-06T09:24:25.253+00:00
updatedAt: 2020-10-06T09:24:25.253+00:00
__v: 0
_id: ObjectId("5f7c37d8ce2e928cc44f8f4c")
userID: "Aishwarya Sharma"
password: "1234"
createdAt: 2020-10-06T09:24:40.346+00:00
updatedAt: 2020-10-06T09:24:40.346+00:00
__v: 0
_id: ObjectId("5f7c5da838114aecd495ffb2")
userID: "user123"
password: "user123"
createdAt: 2020-10-06T12:06:00.920+00:00
updatedAt: 2020-10-06T12:06:00.920+00:00
__v: 0


GreeTings
localhost:3000
Good Morning!
Type here to search
ENG
09:45
12-10-2020

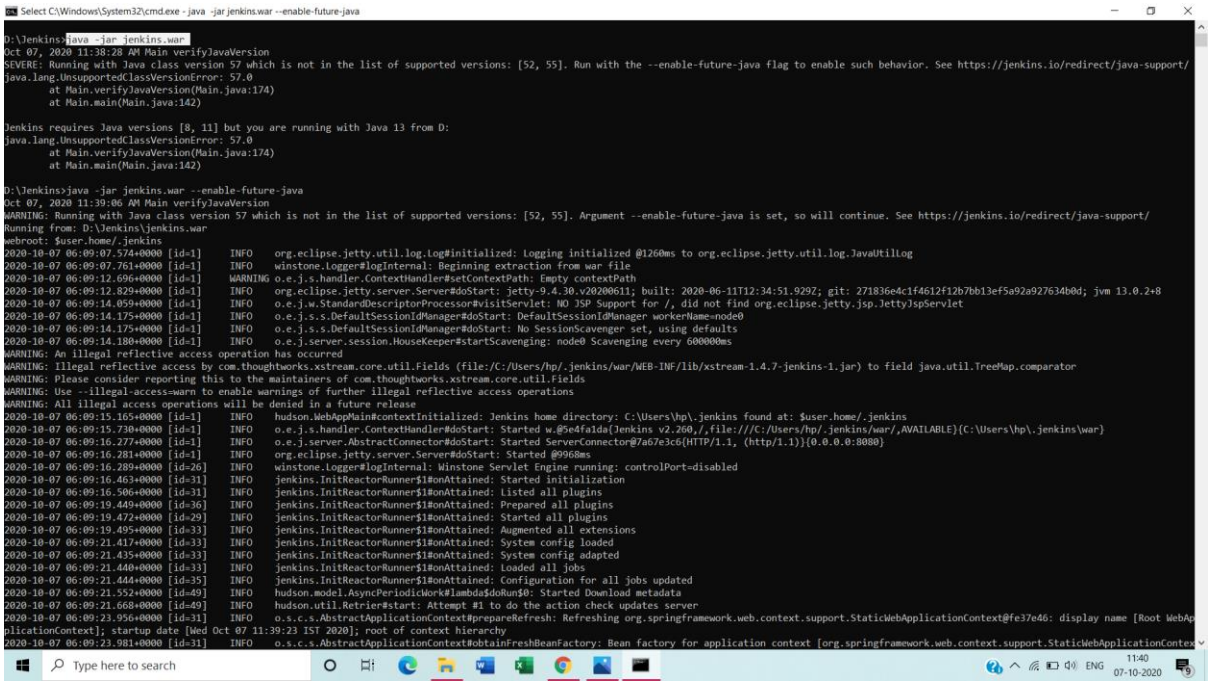

GreeTings
localhost:3000
Good Afternoon!
Type here to search
ENG
13:11
12-10-2020


GreeTings
localhost:3000
Good Evening!
Type here to search
ENG
21:23
11-10-2020

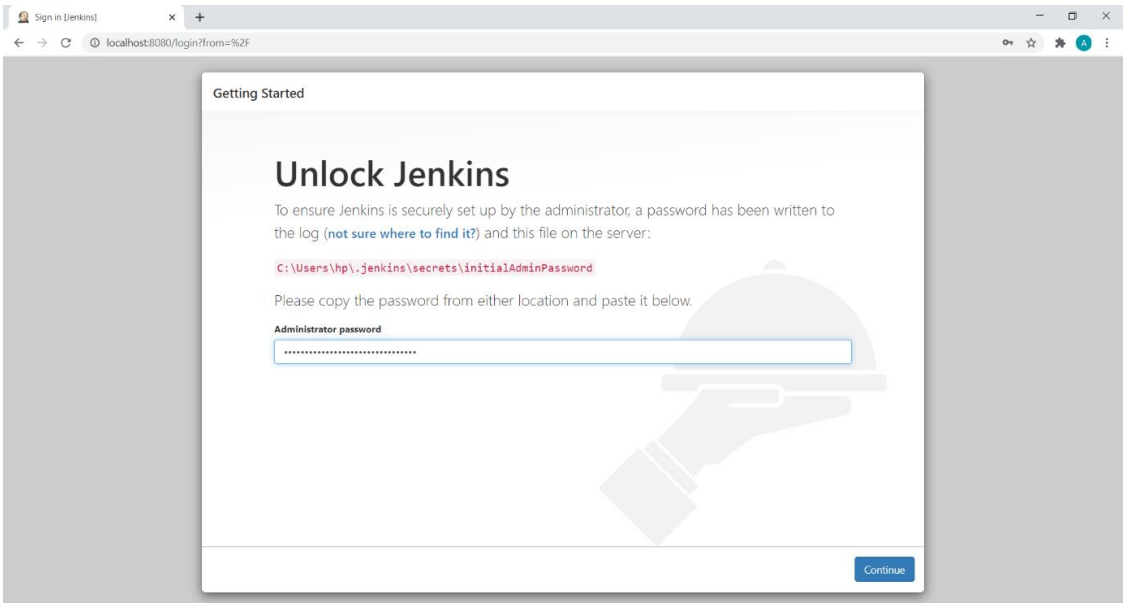
Setting up Jenkins

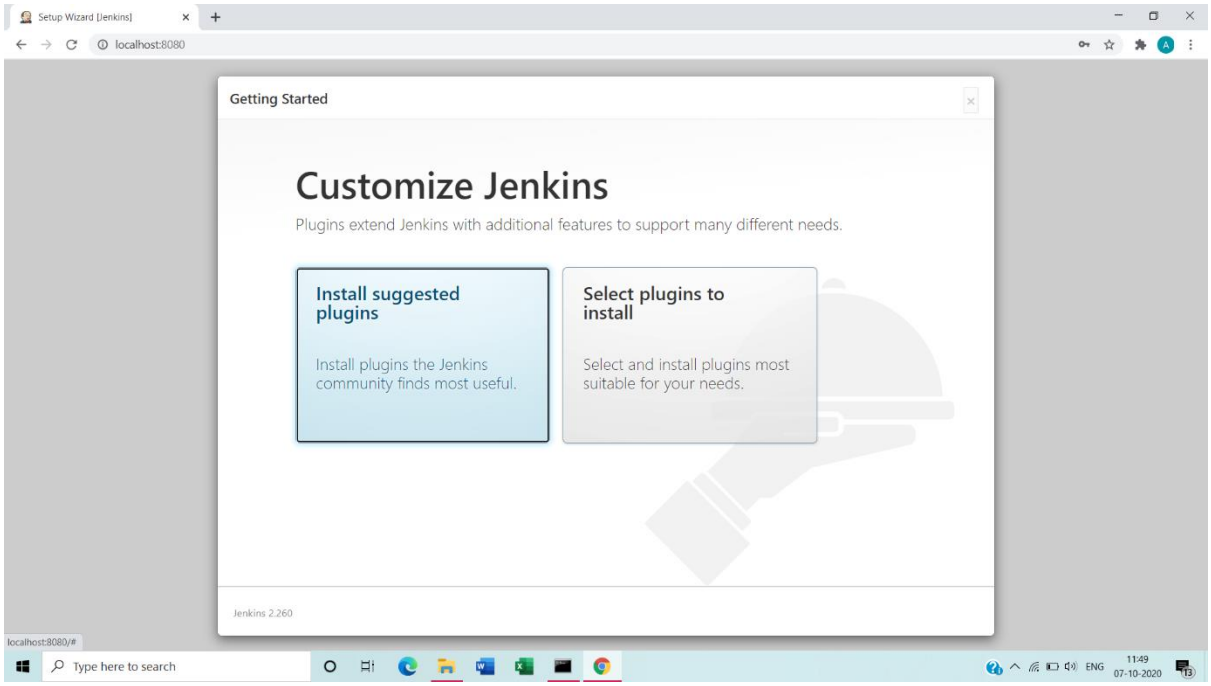
Steps:

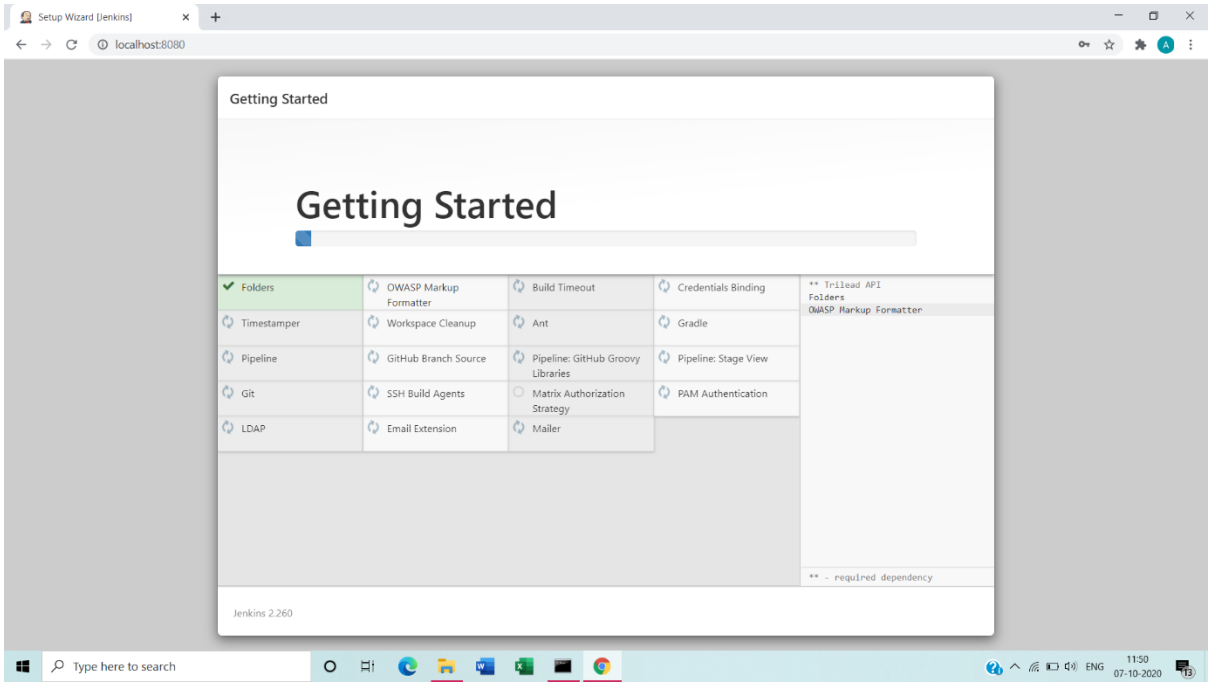
- Ensure that Java JDK and SRE are installed and the path is set in environment variables.
- Download jenkins.war file.
- Go to the directory where you have downloaded the file and open command prompt.
- Type the following command.
  # java -jar jenkins.war

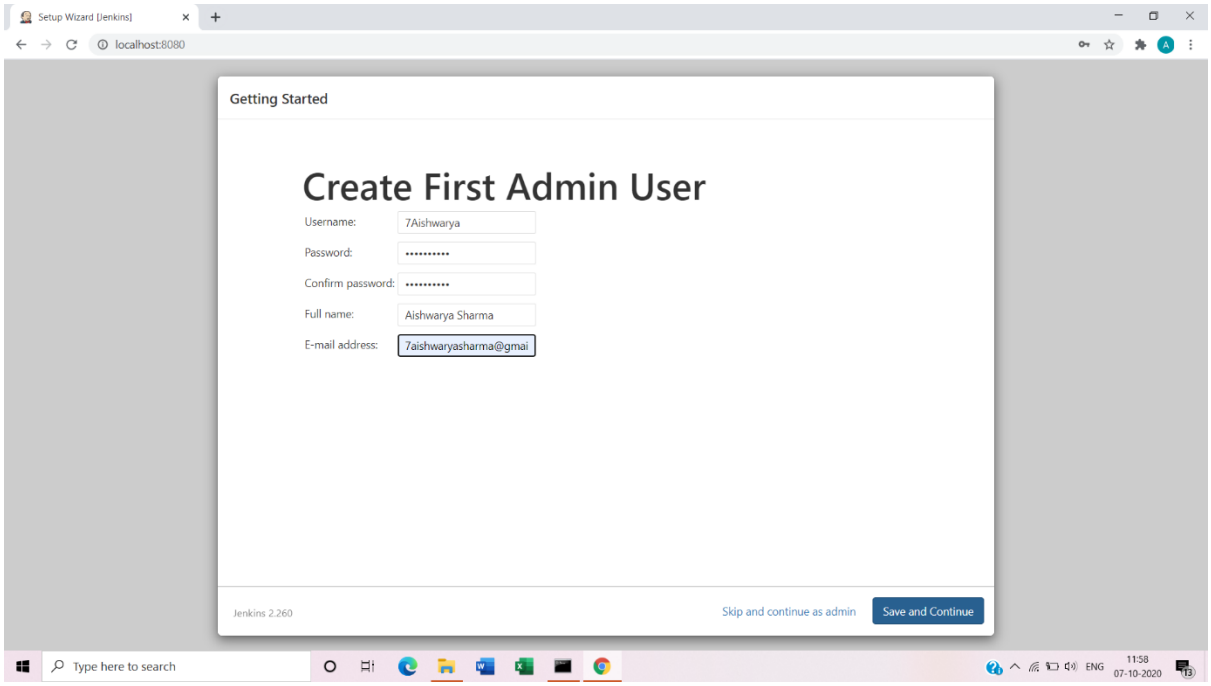


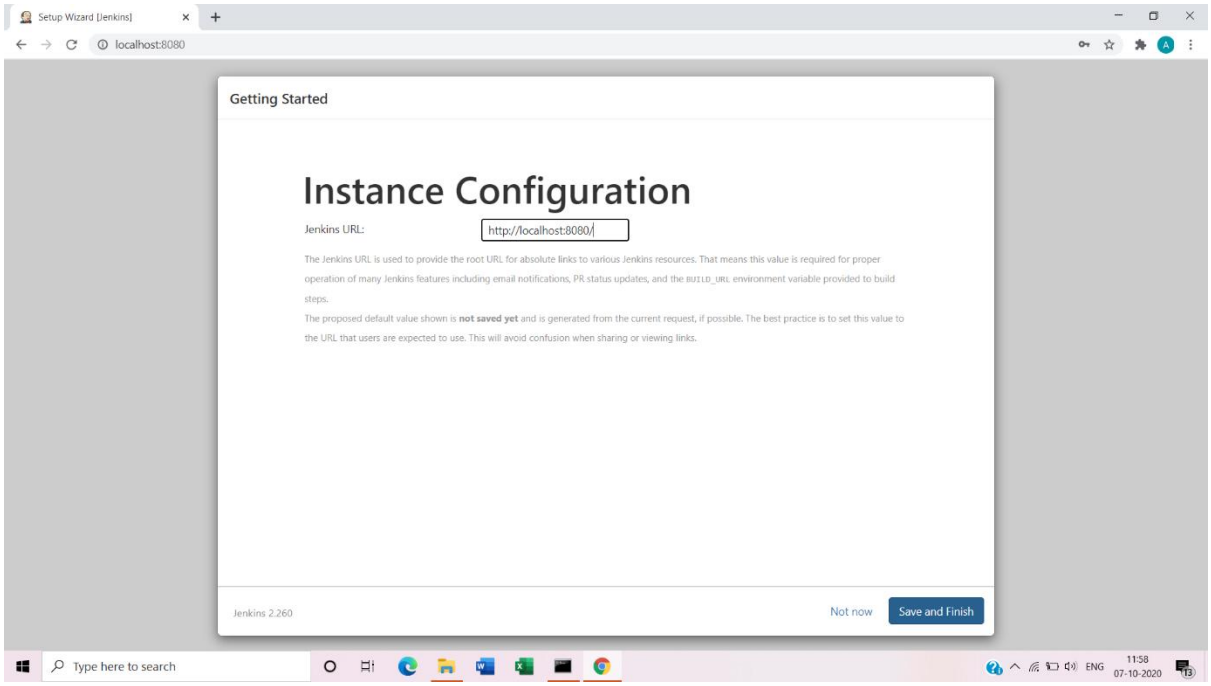
- Now, Jenkins is fully up and running. Copy the password that is generated and use it for further installation.
- Then, go to browser and navigate to localhost:8080
- After that follow the steps as shown in the screenshots below and paste the password under the column administrator password.

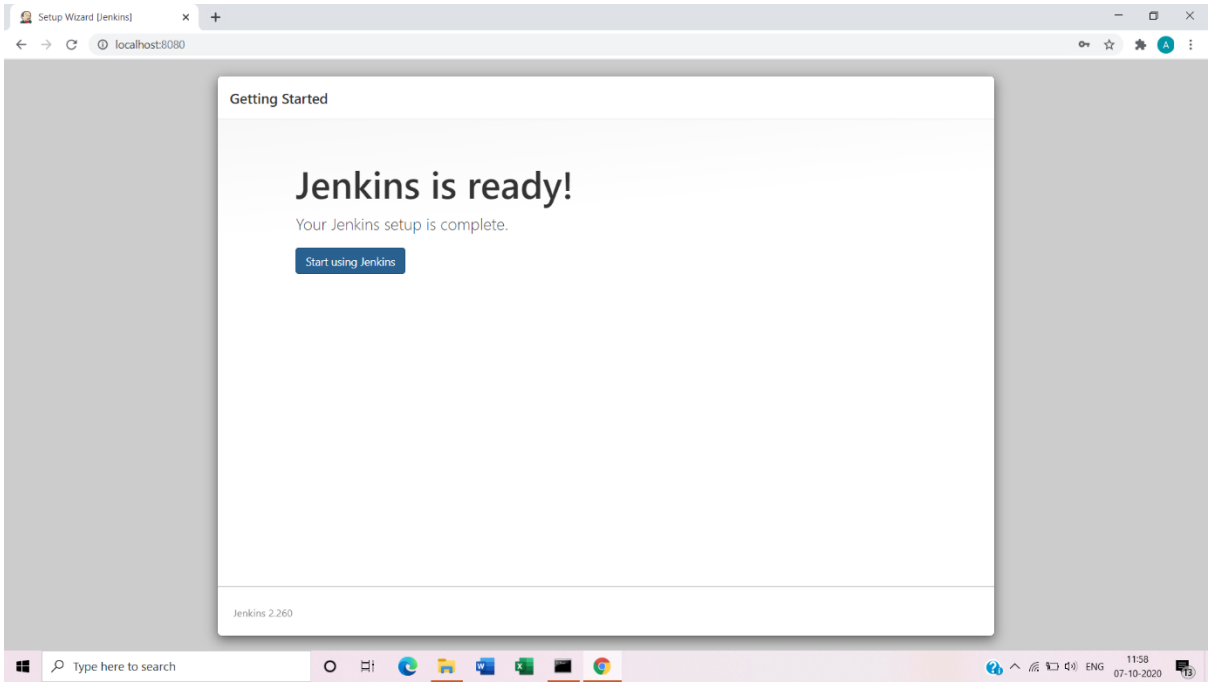



Setup Wizard [Jenkins]
localhost:8080
Getting Started
Customize Jenkins
Plugins extend Jenkins with additional features to support many different needs.
Install suggested plugins
Install plugins the Jenkins community finds most useful.
Select plugins to install
Select and install plugins most suitable for your needs.
Jenkins 2.260
localhost:8080/#
Type here to search
ENG
11:49
07-10-2020


Setup Wizard [Jenkins]
localhost:8080
Getting Started
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
** Trilead API
Folders
OWASP Markup Formatter
** - required dependency
Jenkins 2.260
Type here to search
ENG
11:50
07-10-2020


Setup Wizard [Jenkins]
localhost:8080
Getting Started
Create First Admin User
Username:
7Aishwarya
Password:
Confirm password:
Full name:
Aishwarya Sharma
E-mail address:
7aishwaryasharma@gmai
Jenkins 2.260
Skip and continue as admin
Save and Continue
Type here to search
ENG
11:58
07-10-2020


Setup Wizard [Jenkins]
localhost:8080
Getting Started
Instance Configuration
Jenkins URL:
http://localhost: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.
Jenkins 2.260
Not now
Save and Finish
Type here to search
ENG
11:58
07-10-2020


Setup Wizard [Jenkins]
localhost:8080
Getting Started
Jenkins is ready!
Your Jenkins setup is complete.
Start using Jenkins
Jenkins 2.260
Type here to search
ENG
11:58
07-10-2020

Pushing code to GitHub

Steps:

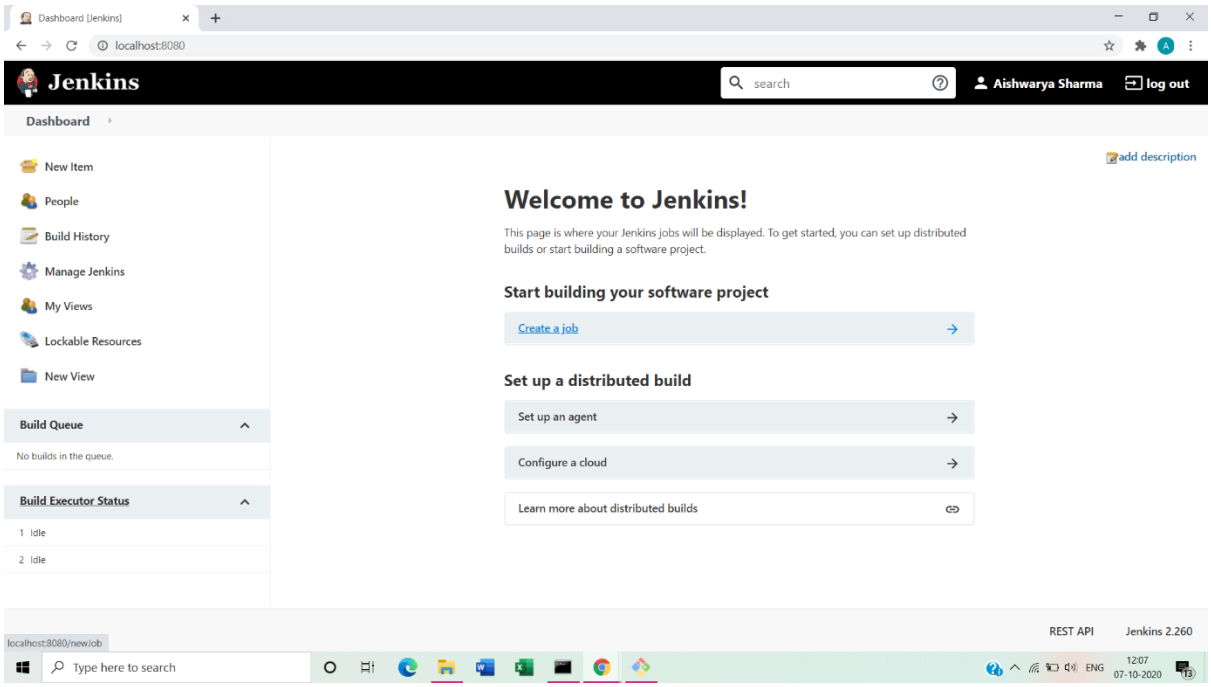
- Create a GitHub repository and clone it on your local system
- Copy all your files and folders of MERN stack application to this folder.
- Type the following commands:
  # git add .
  # git commit -m “commit message”
  # git push -u origin master

```
hp@LAPTOP-0O7S0DF9 MINGW64 ~/Desktop/Simple-web-application (main)
$ git add .

hp@LAPTOP-0O7S0DF9 MINGW64 ~/Desktop/Simple-web-application (main)
$ git commit -m "simple web application"
[main ec8d45b] simple web application
 23 files changed, 13847 insertions(+)
 create mode 100644 greetings-portal/.gitignore
 create mode 100644 greetings-portal/backend/.env
 create mode 100644 greetings-portal/backend/models/login.model.js
 create mode 100644 greetings-portal/backend/routes/page1.js
 create mode 100644 greetings-portal/backend/server.js
 create mode 100644 greetings-portal/package-lock.json
 create mode 100644 greetings-portal/package.json
 create mode 100644 greetings-portal/public/index.html
 create mode 100644 greetings-portal/src/Welcome.js
 create mode 100644 greetings-portal/src/components/App.js
 create mode 100644 greetings-portal/src/components/Form.js
 create mode 100644 greetings-portal/src/components/Greeting.js
 create mode 100644 greetings-portal/src/components/Login.js
 create mode 100644 greetings-portal/src/components/afternoon.jpg
 create mode 100644 greetings-portal/src/components/evening.jpeg
 create mode 100644 greetings-portal/src/components/images/H.ico
 create mode 100644 greetings-portal/src/components/images/afternoon.jpg
 create mode 100644 greetings-portal/src/components/images/evening.jpeg
 create mode 100644 greetings-portal/src/components/images/helpdesk.jpg
 create mode 100644 greetings-portal/src/components/images/morning.jpg
 create mode 100644 greetings-portal/src/components/morning.jpg
 create mode 100644 greetings-portal/src/components/styles.css
 create mode 100644 greetings-portal/src/index.js

hp@LAPTOP-0O7S0DF9 MINGW64 ~/Desktop/Simple-web-application (main)
$ git push -u origin master
error: src refspec master does not match any
error: failed to push some refs to 'https://github.com/7Aishwarya/Simple-web-application.git'

hp@LAPTOP-0O7S0DF9 MINGW64 ~/Desktop/Simple-web-application (main)
$ git push -u origin main
Enumerating objects: 31, done.
Counting objects: 100% (31/31), done.
Delta compression using up to 4 threads
Compressing objects: 100% (26/26), done.
Writing objects: 100% (30/30), 338.89 KiB | 5.06 MiB/s, done.
Total 30 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/7Aishwarya/Simple-web-application.git
   cf55bc7..ec8d45b  main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

hp@LAPTOP-0O7S0DF9 MINGW64 ~/Desktop/Simple-web-application (main)
$
```

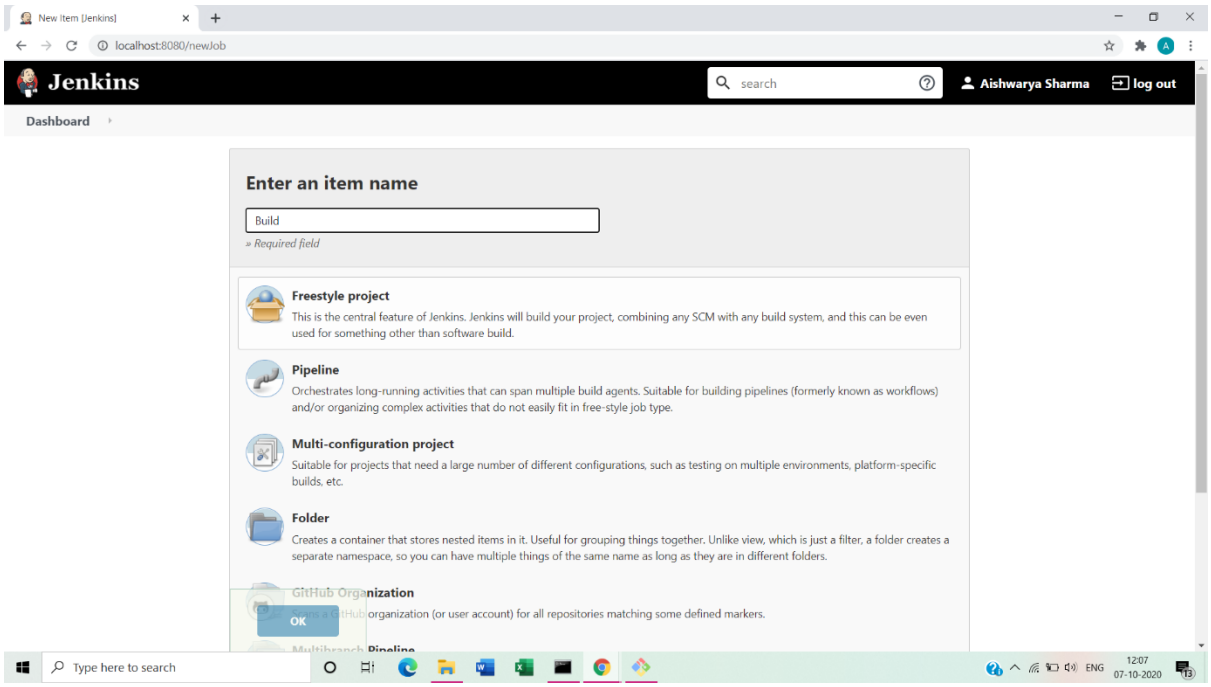
Creating Job in Jenkins

Steps:

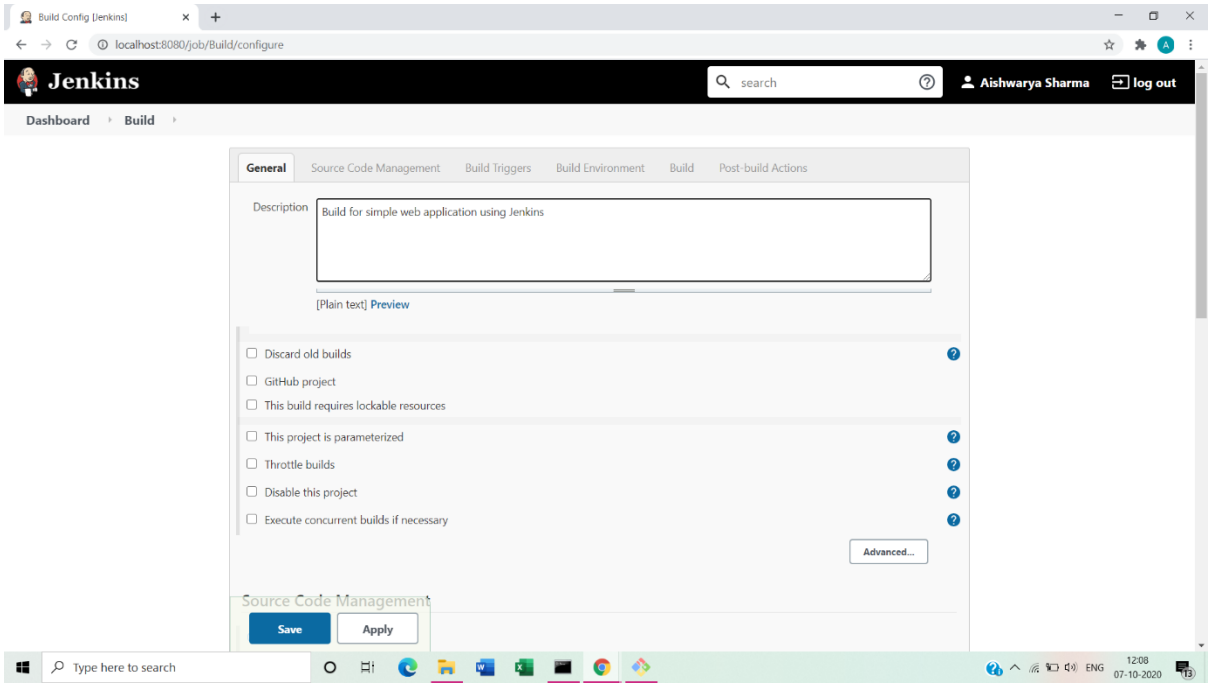
- Click on ‘Create a Job’



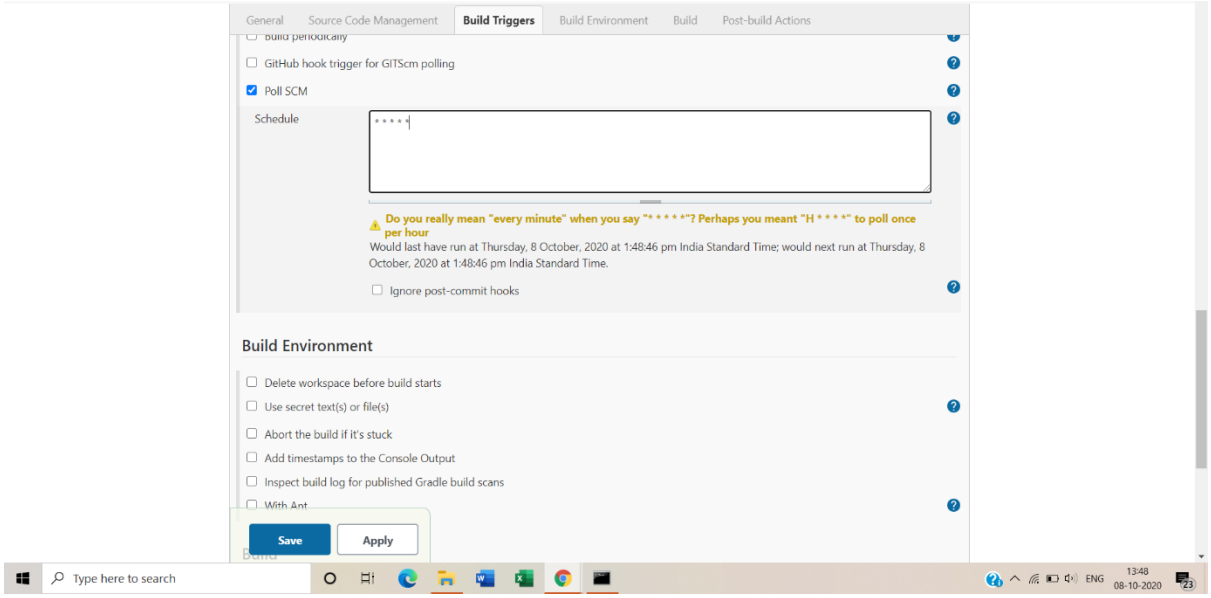
- Select free style project and click OK.



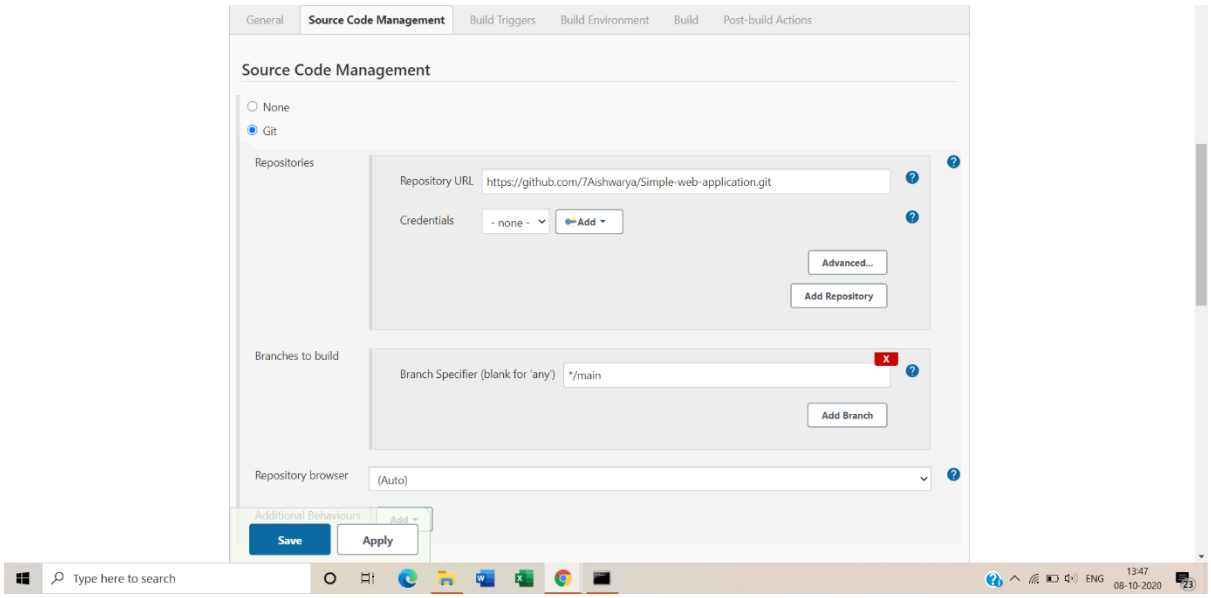
- Follow the steps as shown in the screenshots below.
- Add description



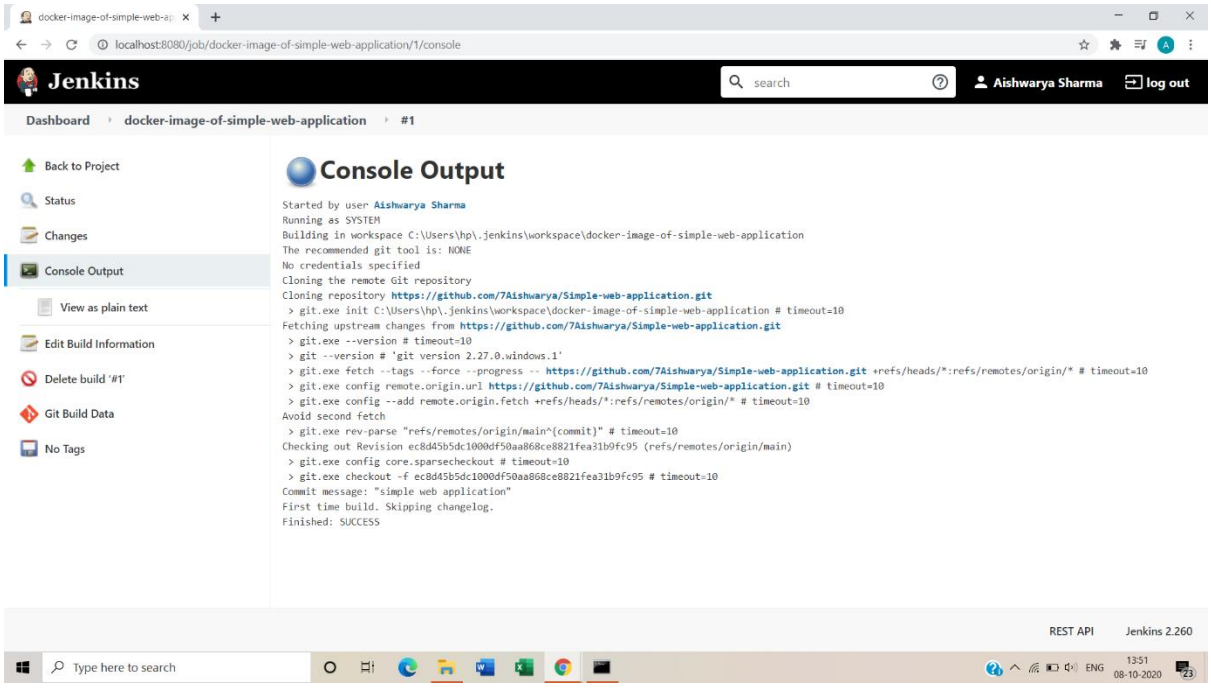
- Build Triggers: Poll SCM and type 5 stars with a space in between.
- (With this step on every commit a build is created)



Source Code Management: Select Git and add the URL of your GitHub repo. Apply and save



- **Create build** and check the output under Console output section on the left side of the dashboard.



Integrating Git and GitHub with Jenkins + Creating and pushing docker image on Docker Hub.

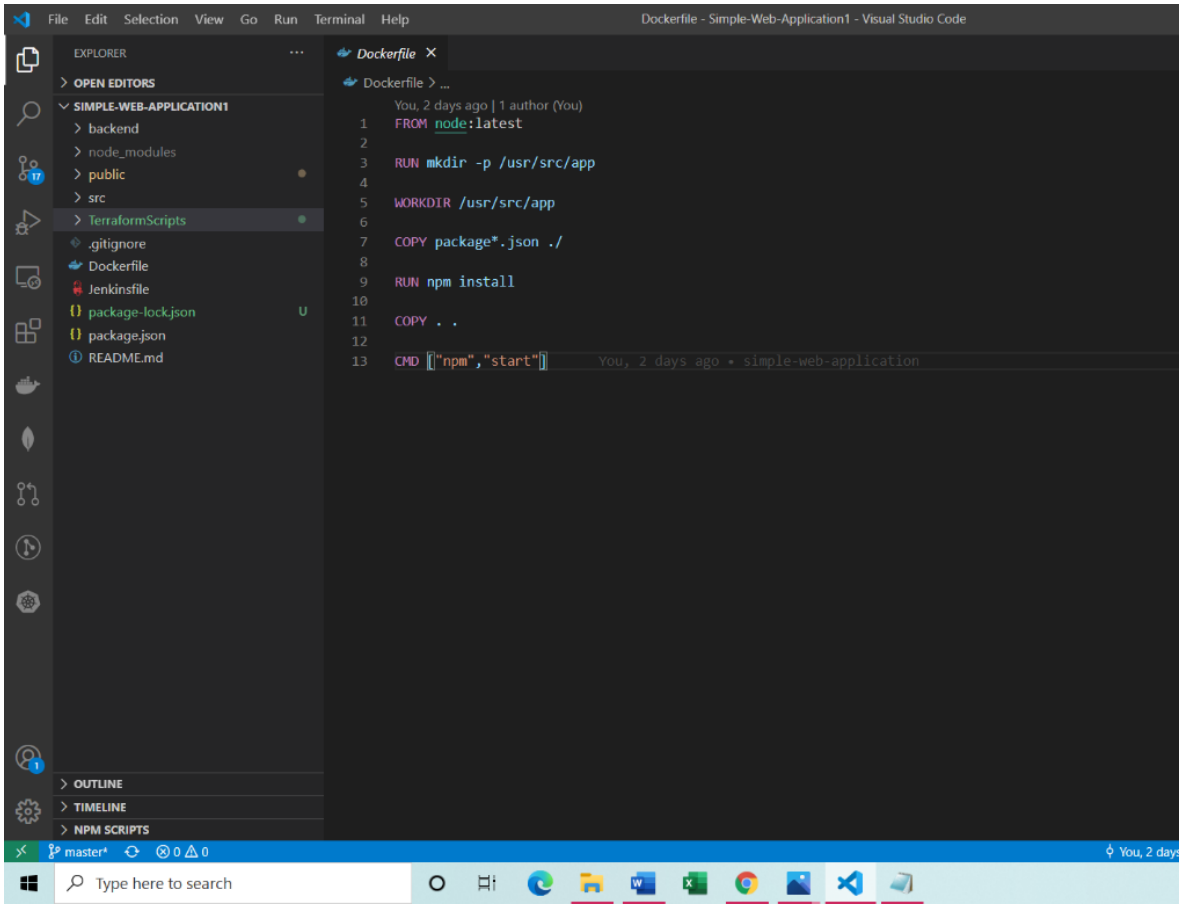
First add Docker File and Jenkins file to your project directory and push them to GitHub using the following commands:

git add .

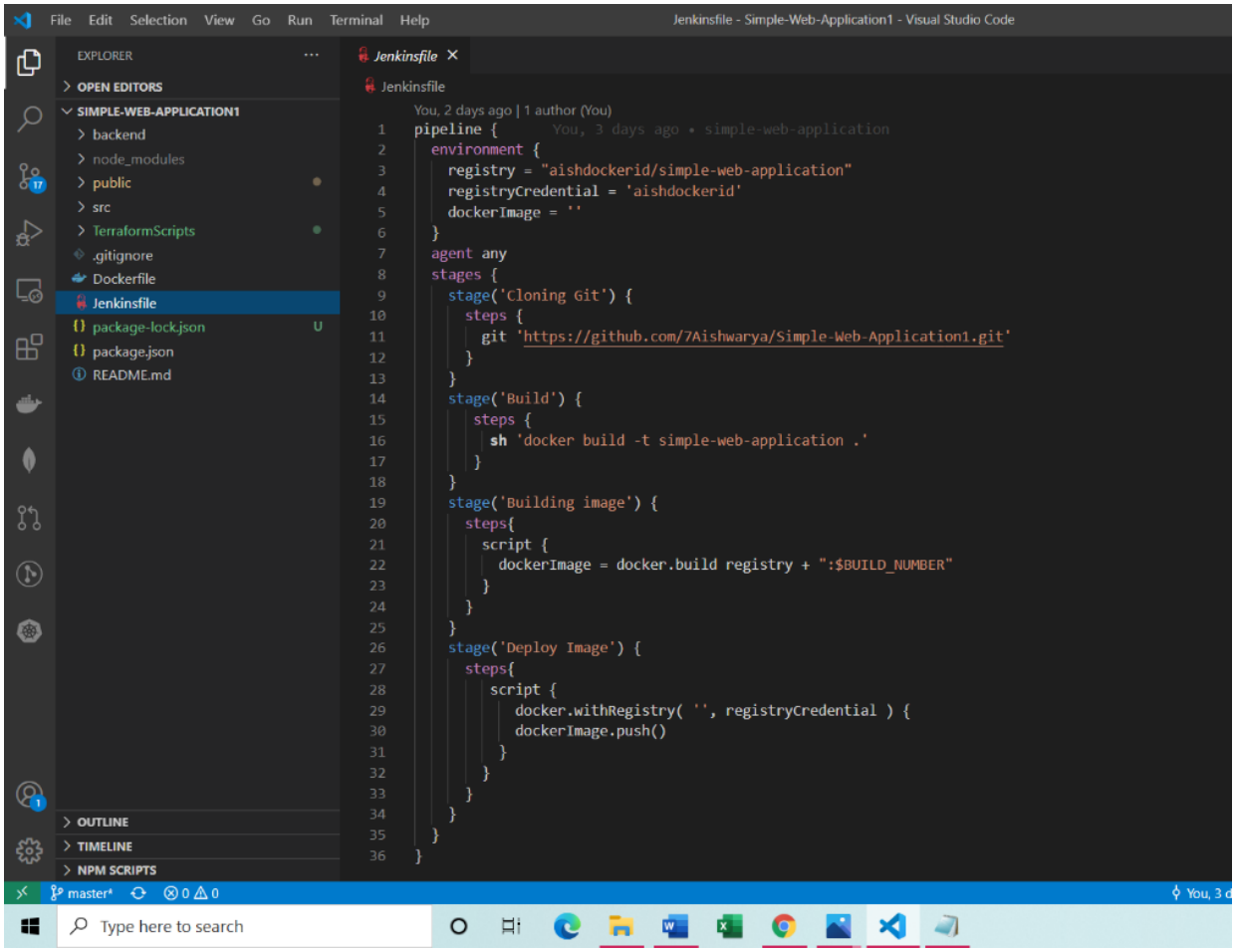
git commit -m “Dockerfile and Jenkinsfile added”

git push -u origin master

Dockerfile

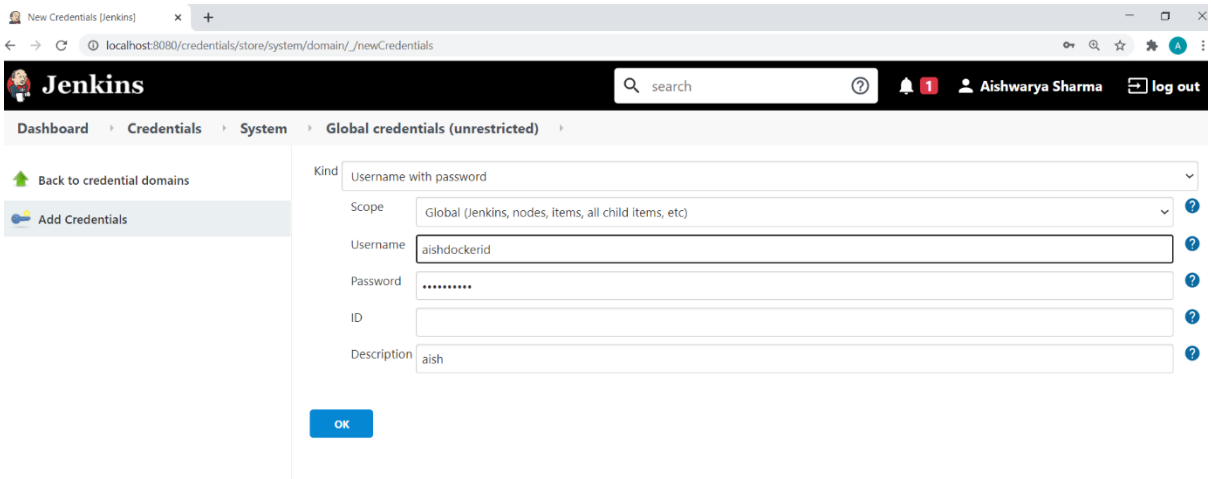


Jenkinsfile

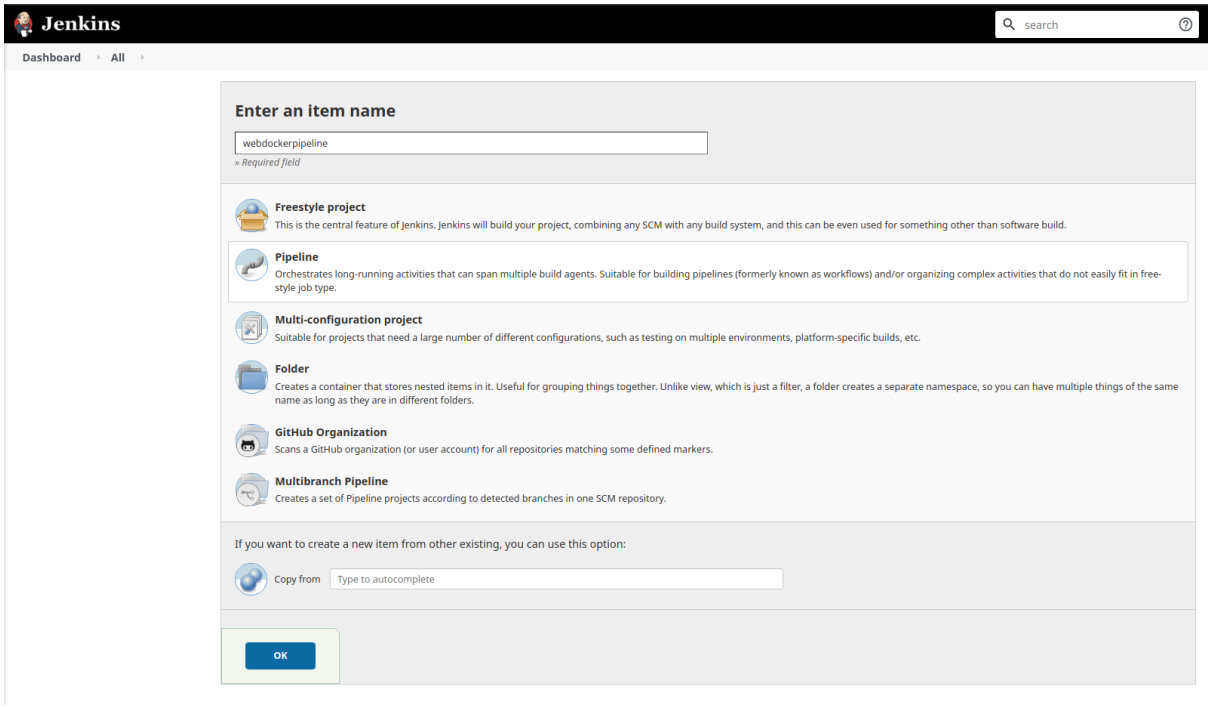


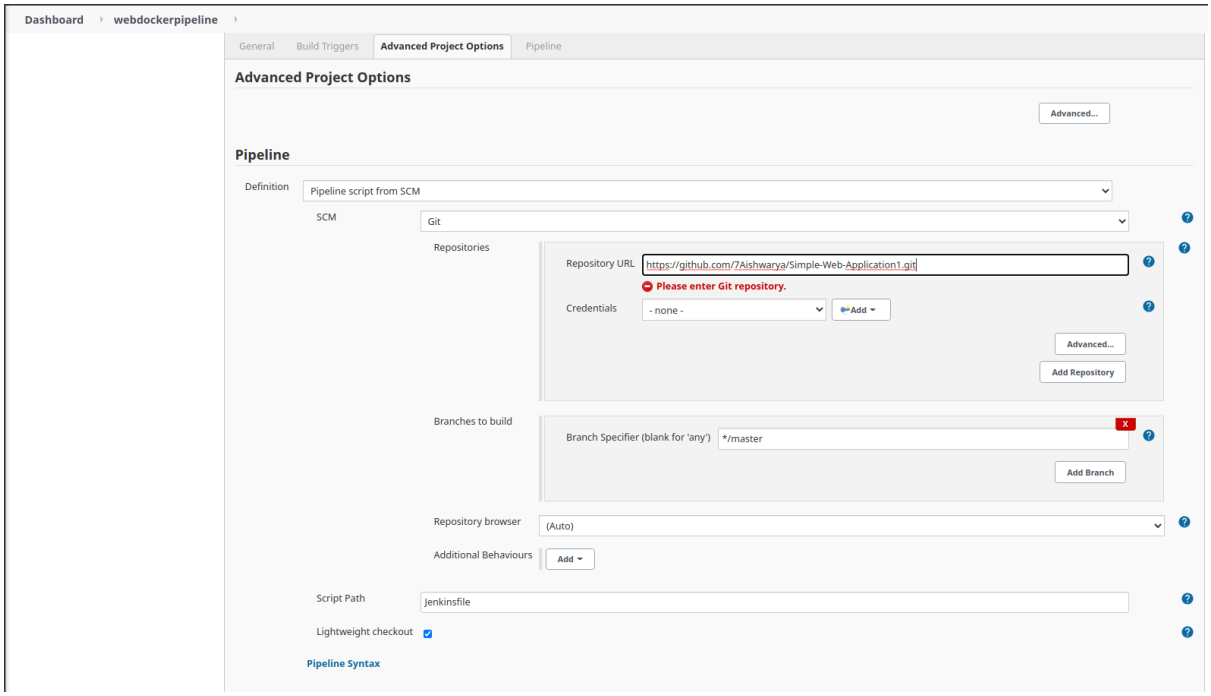
Add docker hub credentials to Jenkins

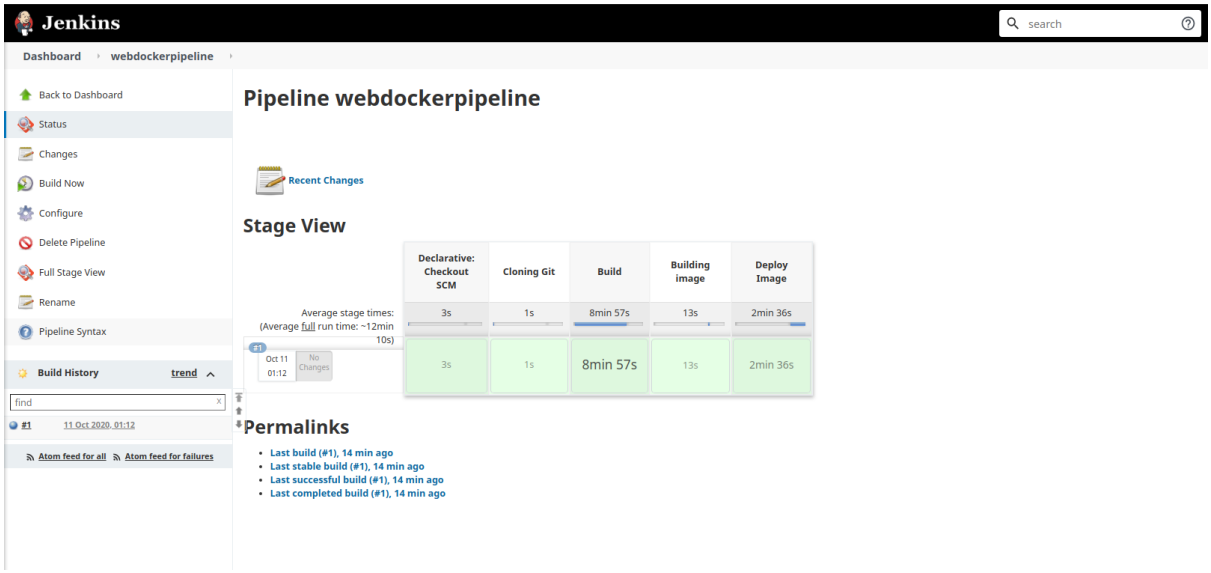
Dashboard -> Credentials -> Global credentials -> Add credentials



Create pipeline







Console Output

Started by user **Aishwarya Sharma**

```
Obtained Jenkinsfile from git https://github.com/7Aishwarya/Simple-Web-Application1.git
Running in Durability level: MAX_SURVIVABILITY
[Pipeline] Start of Pipeline
[Pipeline] node
Running on Jenkins in /var/lib/jenkins/workspace/webdockerpipeline
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Declarative: Checkout SCM)
[Pipeline] checkout
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
Cloning the remote Git repository
Cloning repository https://github.com/7Aishwarya/Simple-Web-Application1.git
 > git init /var/lib/jenkins/workspace/webdockerpipeline # timeout=10
Fetching upstream changes from https://github.com/7Aishwarya/Simple-Web-Application1.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/7Aishwarya/Simple-Web-Application1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git config remote.origin.url https://github.com/7Aishwarya/Simple-Web-Application1.git # timeout=10
 > git config --add remote.origin.fetch +refs/heads/*:refs/remotes/origin/* # timeout=10
Avoid second fetch
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 82811e0e50450f0f3ef629b94cff565402208684 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 82811e0e50450f0f3ef629b94cff565402208684 # timeout=10
Commit message: "simple-web-application"
First time build. Skipping changelog.
[Pipeline] }
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Cloning Git)
[Pipeline] git
Selected Git installation does not exist. Using Default
The recommended git tool is: NONE
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/7Aishwarya/Simple-Web-Application1.git # timeout=10
Fetching upstream changes from https://github.com/7Aishwarya/Simple-Web-Application1.git
 > git --version # timeout=10
```

Dashboard › webdockerpipeline › #1

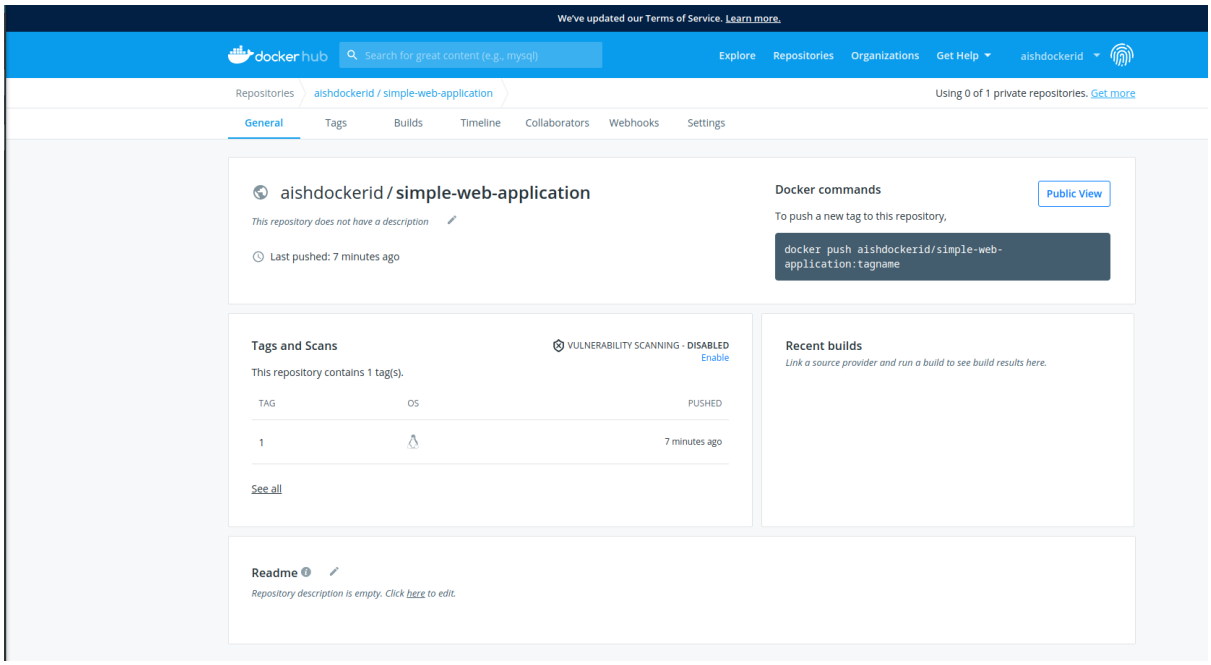
```
No credentials specified
 > git rev-parse --is-inside-work-tree # timeout=10
Fetching changes from the remote Git repository
 > git config remote.origin.url https://github.com/7Aishwarya/Simple-Web-Application1.git # timeout=10
Fetching upstream changes from https://github.com/7Aishwarya/Simple-Web-Application1.git
 > git --version # timeout=10
 > git --version # 'git version 2.25.1'
 > git fetch --tags --force --progress -- https://github.com/7Aishwarya/Simple-Web-Application1.git +refs/heads/*:refs/remotes/origin/* # timeout=10
 > git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision 82811e0e50450f0f3ef629b94cff565402208684 (refs/remotes/origin/master)
 > git config core.sparsecheckout # timeout=10
 > git checkout -f 82811e0e50450f0f3ef629b94cff565402208684 # timeout=10
 > git branch -a -v --no-abbrev # timeout=10
 > git checkout -b master 82811e0e50450f0f3ef629b94cff565402208684 # timeout=10
Commit message: "simple-web-application"
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] sh
+ docker build -t simple-web-application .
Sending build context to Docker daemon  789.5kB

Step 1/7 : FROM node:latest
 ---> c0a69c6687d2
Step 2/7 : RUN mkdir -p /usr/src/app
 ---> Using cache
 ---> 27eb664b916d
Step 3/7 : WORKDIR /usr/src/app
 ---> Using cache
 ---> 909b813e7f79
Step 4/7 : COPY package*.json ./
 ---> 359b8d41ca98
Step 5/7 : RUN npm install
 ---> Running in 0d520370d1b6
```

Dashboard › webdockerpipeline › #1

```
$ docker login -u aishdockerid -p ******** https://index.docker.io/v1/
WARNING! Using --password via the CLI is insecure. Use --password-stdin.
WARNING! Your password will be stored unencrypted in /var/lib/jenkins/workspace/webdockerpipeline@tmp/9147ed6c-2468-417d-b574-79fe66459264/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Succeeded
[Pipeline] {
[Pipeline] isUnix
[Pipeline] sh
+ docker tag aishdockerid/simple-web-application:1 aishdockerid/simple-web-application:1
[Pipeline] isUnix
[Pipeline] sh
+ docker push aishdockerid/simple-web-application:1
The push refers to repository [docker.io/aishdockerid/simple-web-application]
0aa614212ce0: Preparing
7744bea74119: Preparing
1b75f28143c6: Preparing
74a76214cccb: Preparing
74ef25c221ef: Preparing
193b269f2c3f: Preparing
4a0e6e0a73a2: Preparing
193b269f2c3f: Waiting
174e334f3f46: Preparing
4a0e6e0a73a2: Waiting
cbe6bbd0c86f: Preparing
ef5de533cb53: Preparing
174e334f3f46: Waiting
a4c504f73441: Preparing
ef5de533cb53: Waiting
e8847c2734e1: Preparing
b323b70996e4: Preparing
e8847c2734e1: Waiting
b323b70996e4: Waiting
a4c504f73441: Waiting
```

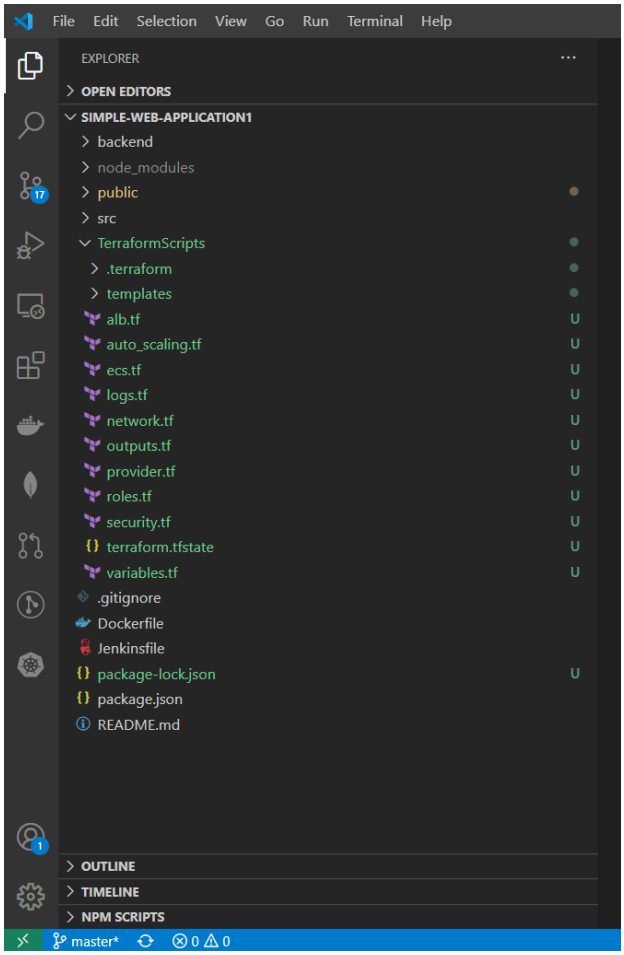
```
7744bea74119: Pushed
1: digest: sha256:d1f0ebabe05f2a55e11ca990f74d7c97f33c74cbf79c0f6201d298c619471c97 size: 3051
[Pipeline] }
[Pipeline] // withDockerRegistry
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // withEnv
[Pipeline] }
[Pipeline] // node
[Pipeline] End of Pipeline
Finished: SUCCESS
```

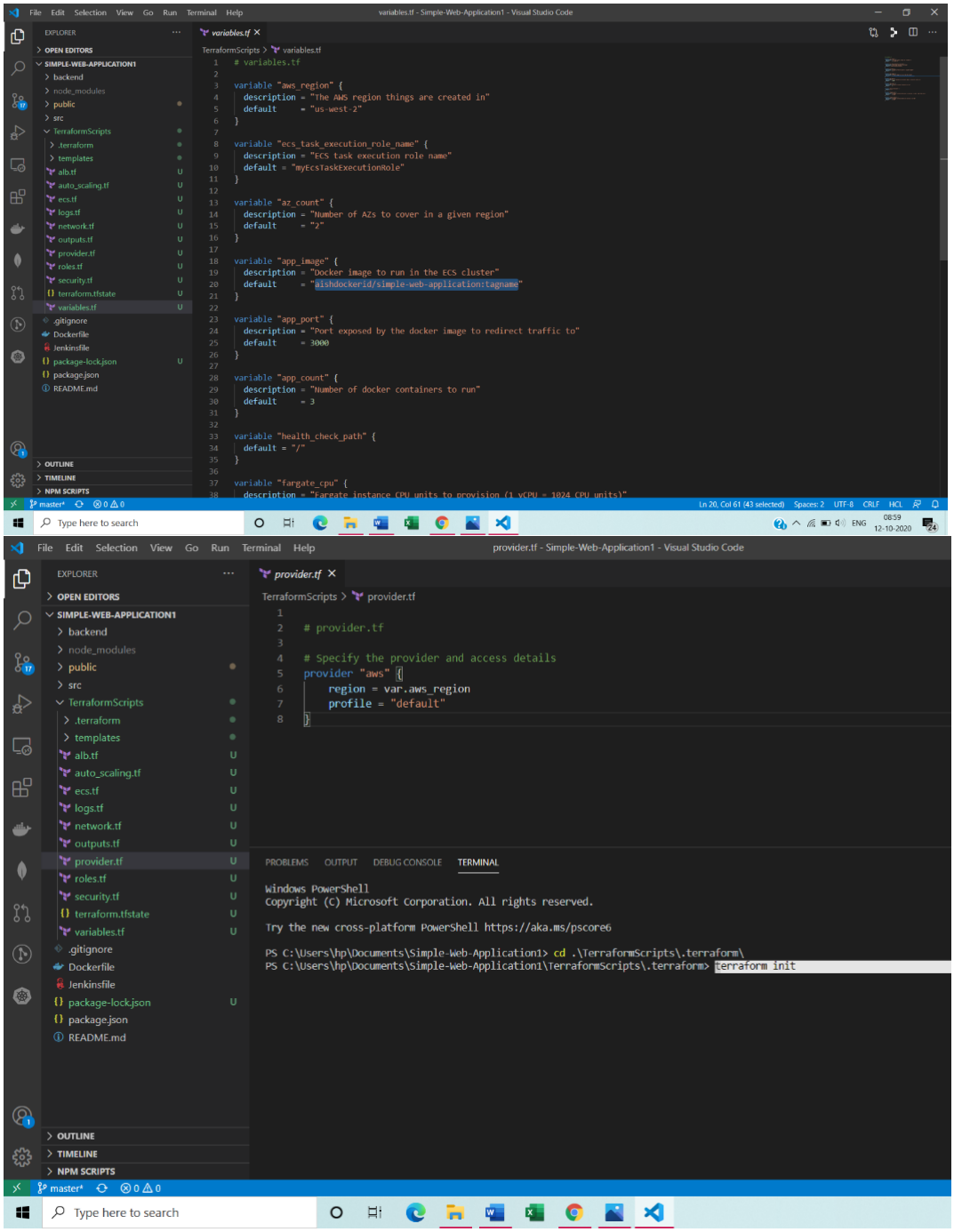
Docker Image pushed on Docker Hub



Deploying Image using Terraform and AWS platform

Add Terraform scripts to your project directory





Refer to the screenshots above to make changes to variables.tf and provider.tf file.

You have to provide your dockerhub credentials and github repo in cariables.tf

In provider.tf in place of default you have to enter the [name] as mentioned in credentials file inside .aws folder.

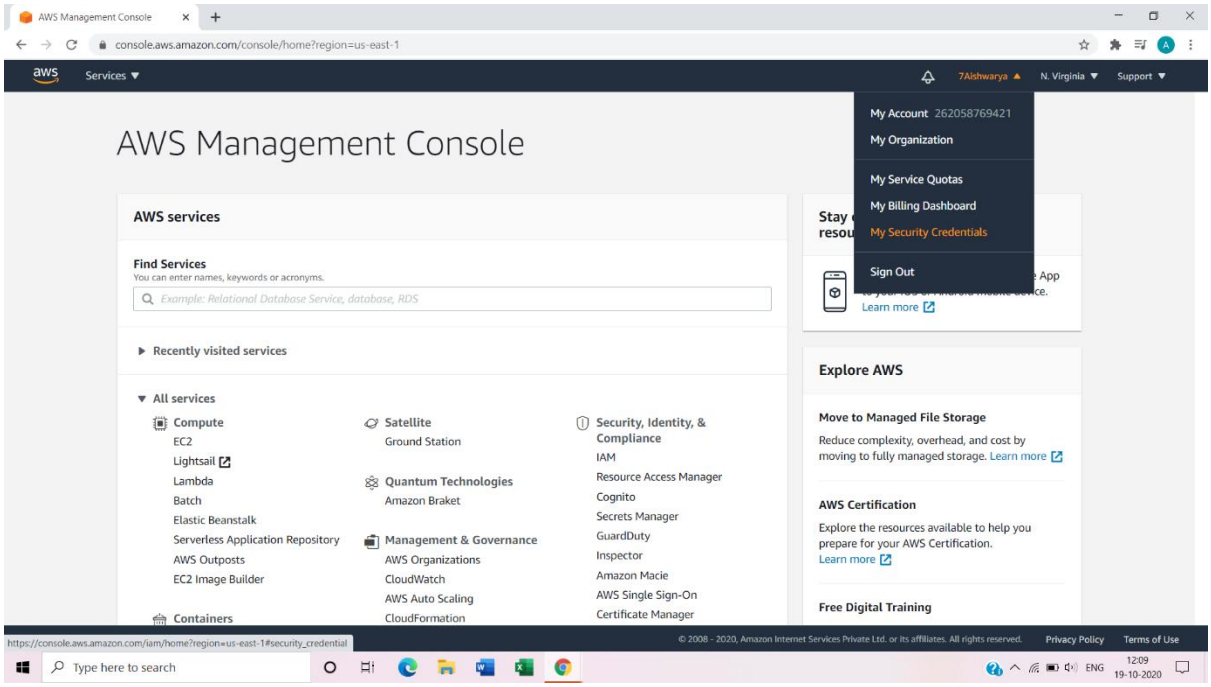
Then initialize the working directory with terraform configuration files using the following command.

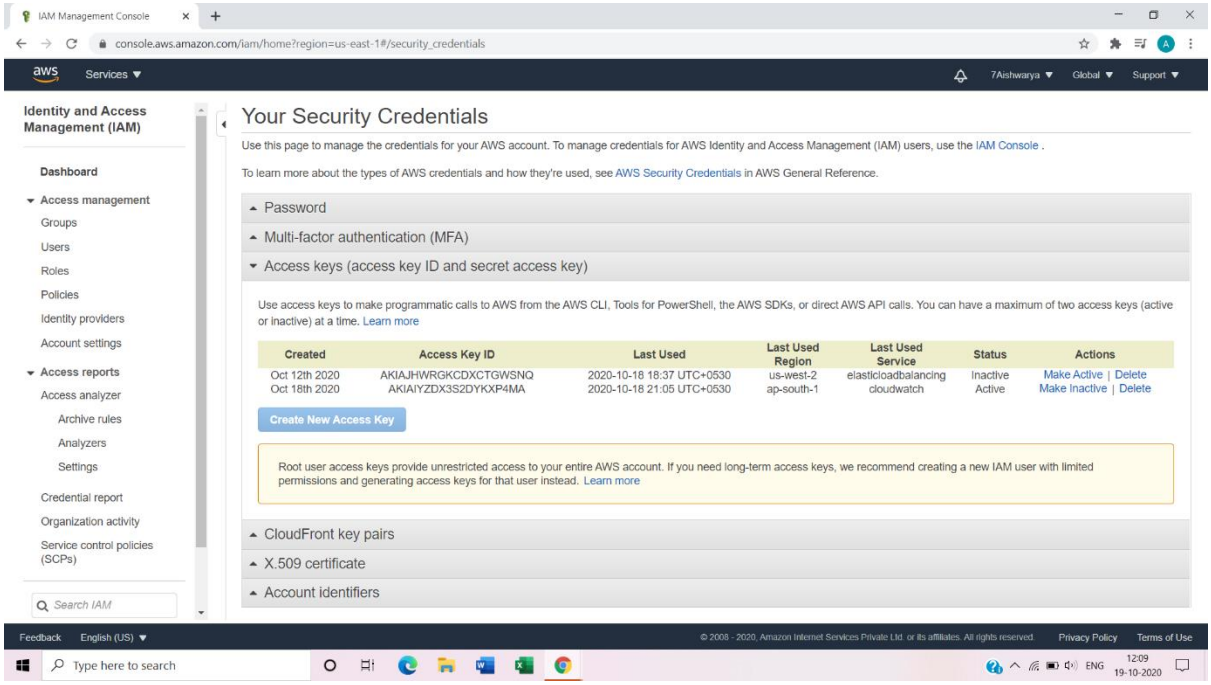
terraform init

configure aws

//provide credentials (you get the credentials from your aws console)

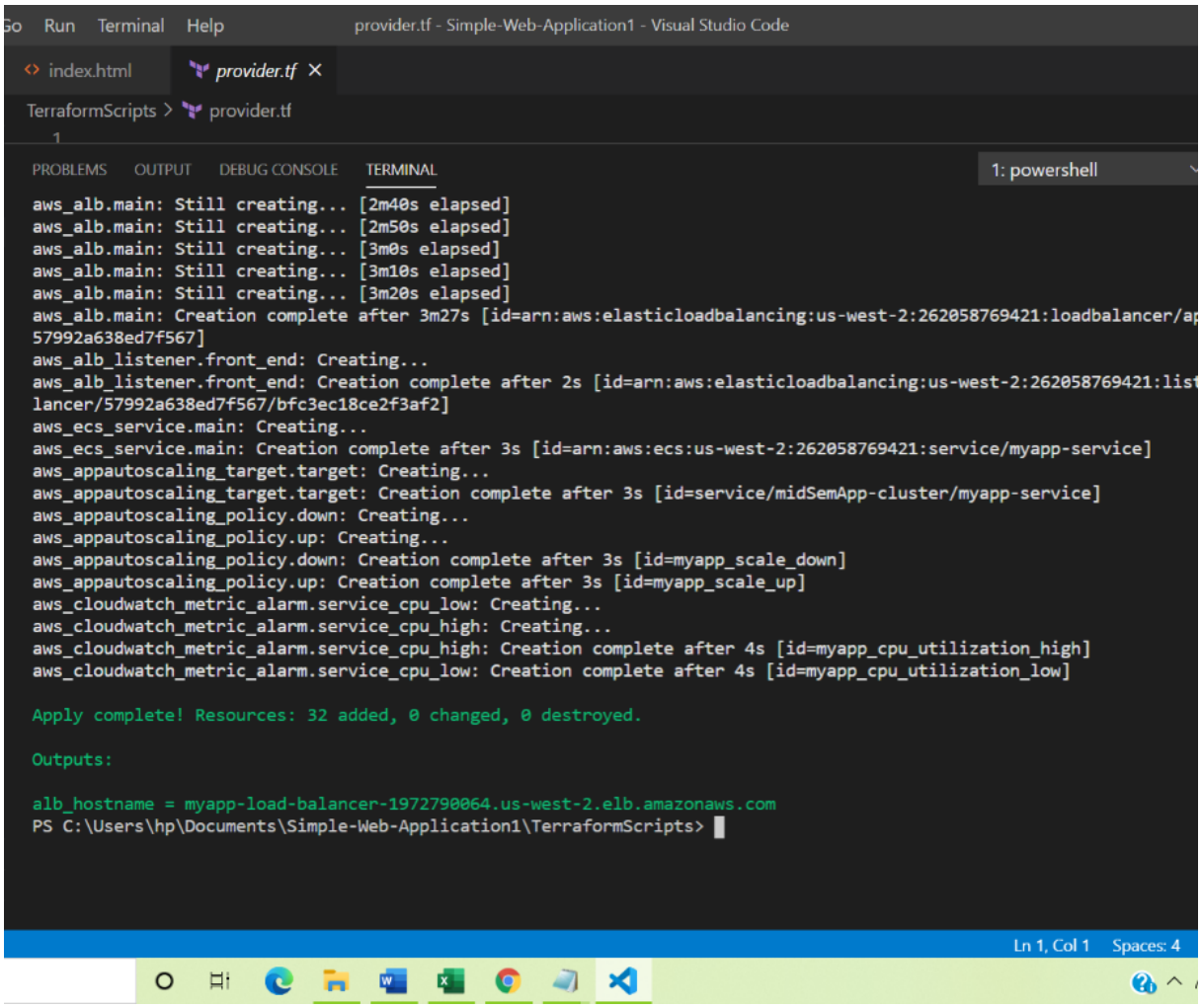
Go to AWS console -> select your username at the top -> My security credentials -> Acess keys -> Create new Access keys or use existing keys



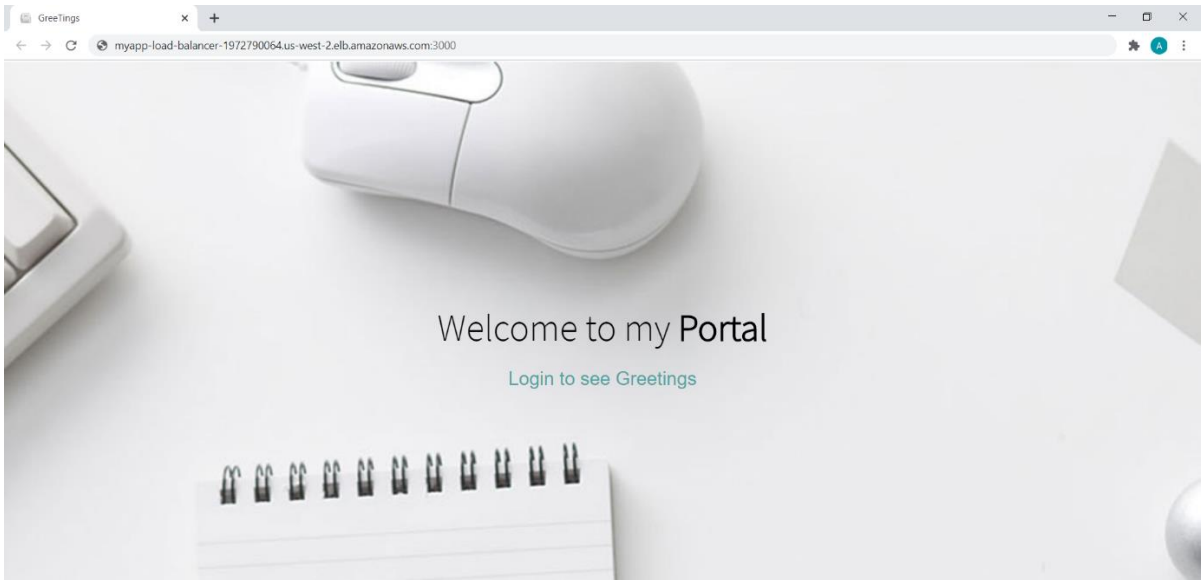


terraform plan

terraform apply



Now you can access your application through the link which you get as alb_hostname.



Everything is done! Cheers :D