

TECHNOLOGY

Computing

Caltech

**Center for Technology &
Management Education**

Configure AWS for Front End

TECHNOLOGY

Computing

Caltech

**Center for Technology &
Management Education**

Set Up AWS for Angular Apps

You Already Know

Before we begin, let's recall what we have covered till now:



Agile



Git



SQL



Angular



HTML



CSS



JavaScript



Core Java

You Already Know

Before we begin, let's recall what we have covered till now:



JDBC

JDBC



JSP

JSP



Servlets

Servlets



MongoDB



Maven

You Already Know

Before we begin, let's recall what we have covered till now:

JUnit

JUnit



Spring



Spring Boot



Webservices

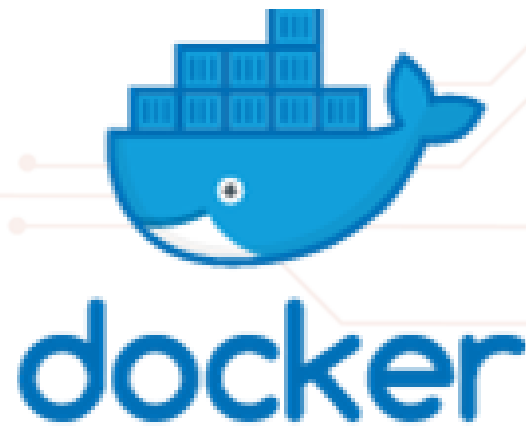


Microservices

Microservices

You Already Know

Before we begin, let's recall what we have covered till now:



Docker



Jenkins



AWS

Front-End Back-End Communication

- Used HTTP Client in Angular to communicate with Java back end
- Implemented communication for both admin and end-user projects

Jenkins Pipeline

- Used Jenkins build automation server to build angular apps and Java back end
- Created Pipeline with multiple stages

Docker

- Used Docker to build images for Angular apps and Java back end
- Used Jenkins to dockerize the Angular apps and Java back end with Jenkins



A Day in the Life of a Full Stack Developer

As a full stack web developer, our key role is to develop both client and server software.



Angular and Node can be used to build front end of the web page.



Spring Boot, Java, and MySQL or MongoDB can be used to build at the back end.



A Day in the Life of a Full Stack Developer

Bob needs to configure AWS for front end. He brainstorms a bit and finds a solution.

Let me use Jenkins, Docker, and AWS to build CI/CD Pipelines, containerize the apps, and finally host them to AWS EC2 instance.



In this lesson, we will host the angular apps on EC2 instance, use Jenkins and Docker as DevOps tools, and help Bob complete his task effectively and quickly.

Learning Objectives

By the end of this lesson, you will be able to:

- 🕒 Create and configure EC2 instance on AWS
- 🕒 Create S3 bucket
- 🕒 Use Jenkins on EC2 for CI/CD
- 🕒 Integrate Docker in Jenkins to build and release images as containers on EC2



Create and Configure EC2 in AWS

AWS

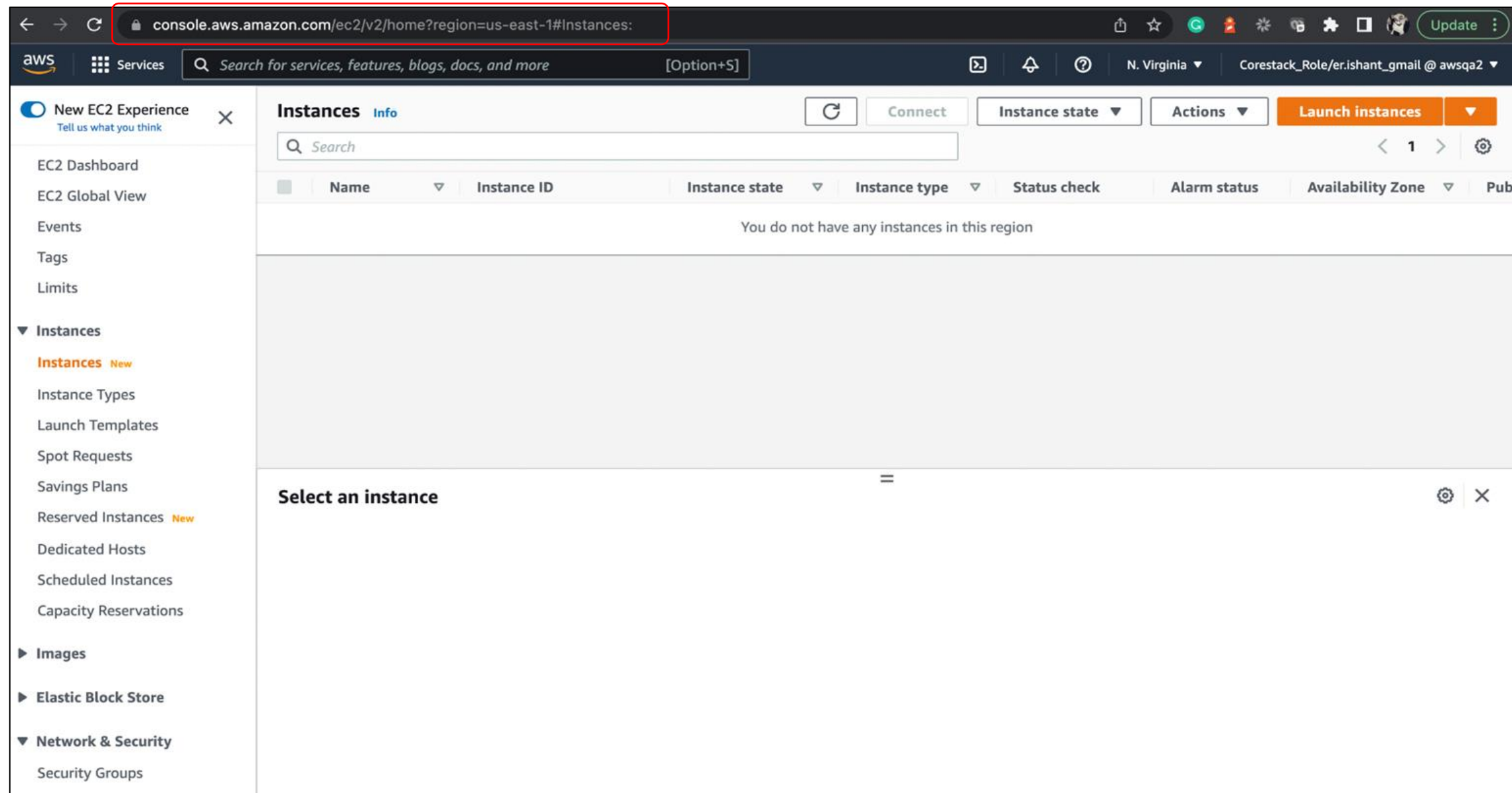
Amazon Web Services (AWS) is a cloud platform, which offers more than 200 fully featured services.



Angular web apps on EC2 instance is deployed using Jenkins and Docker.

Create an EC2 Instance

Open the **EC2 dashboard**.



Launch Instance

Launch a new Instance.

The screenshot shows the AWS Management Console interface. The browser address bar is `console.aws.amazon.com/ec2/v2/home?region=us-east-1#Instances:`. The left sidebar shows the navigation menu with 'Instances' selected. The main content area displays a table of EC2 instances. One instance, 'FrontEndEC2', is highlighted with a red box. A red arrow points from this instance to a green callout box that says 'Use the AWS Cloud Shell to perform tasks.' Another red arrow points from the 'FrontEndEC2' instance to another green callout box that says 'Create service group and key pair to access the instance from terminal.' Below the table, the details for the instance 'i-08bc6e53201548e00 (FrontEndEc2)' are shown, including its public IPv4 address (3.88.180.84) and private IPv4 address (172.31.28.55).

Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
FrontEndEC2	i-08bc6e53201548e00	Running	t2.micro	Initializing	No alarms	us-east-1d

Instance: i-08bc6e53201548e00 (FrontEndEc2)

Details	Security	Networking	Storage	Status checks	Monitoring	Tags
Instance summary						
Instance ID i-08bc6e53201548e00 (FrontEndEc2)	Public IPv4 address 3.88.180.84 open address		Private IPv4 addresses 172.31.28.55			
IPv6 address -	Instance state Running		Public IPv4 DNS ec2-3-88-180-84.compute-1.amazonaws.com open address			
Hostname type IP name: ip-172-31-28-55.ec2.internal	Private IP DNS name (IPv4 only) ip-172-31-28-55.ec2.internal		Answer private resource DNS name -			

Connect to EC2 Instance

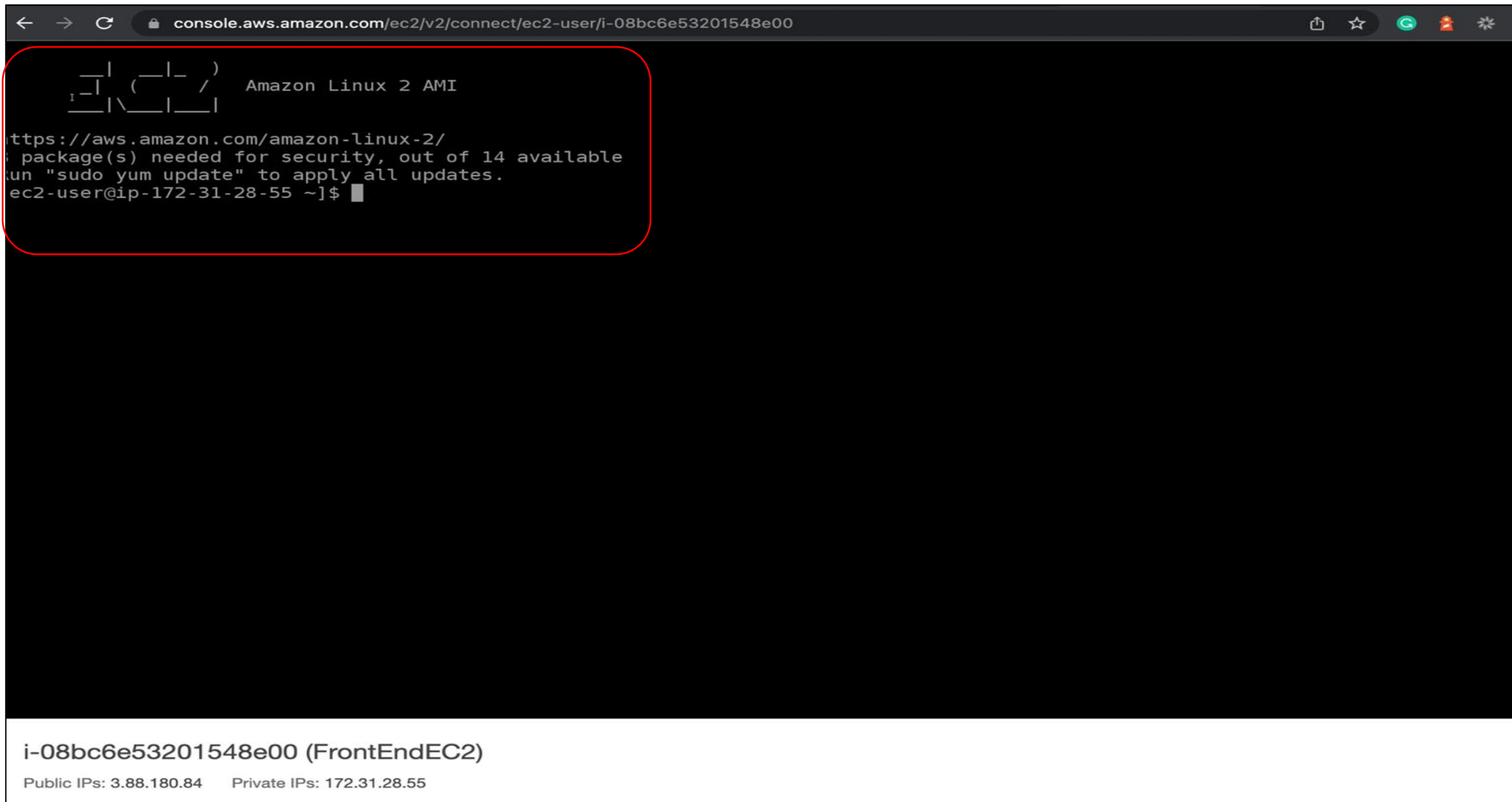
Connect through AWS Cloud Shell, which is the first option on clicking connect that is **EC2 Instance Connect**.

The screenshot shows the AWS Management Console interface for connecting to an EC2 instance. The breadcrumb navigation indicates the path: EC2 > Instances > i-08bc6e53201548e00 > Connect to instance. The main heading is "Connect to instance" with an "Info" link. Below this, a subtext says "Connect to your instance i-08bc6e53201548e00 (FrontEndEC2) using any of these options". There are four tabs: "EC2 Instance Connect" (highlighted with a red box), "Session Manager", "SSH client", and "EC2 Serial Console". Below the tabs, the instance details are listed: Instance ID (i-08bc6e53201548e00 (FrontEndEC2)), Public IP address (3.88.180.84), and User name (ec2-user). A note at the bottom states: "Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name." At the bottom right, there are "Cancel" and "Connect" buttons.

Connect using one of the methods, like ssh.

Launch the EC2 Shell and Configure the Tools

Install the tools and software required to deploy Angular apps and configure Node, Git, Docker, and Jenkins.



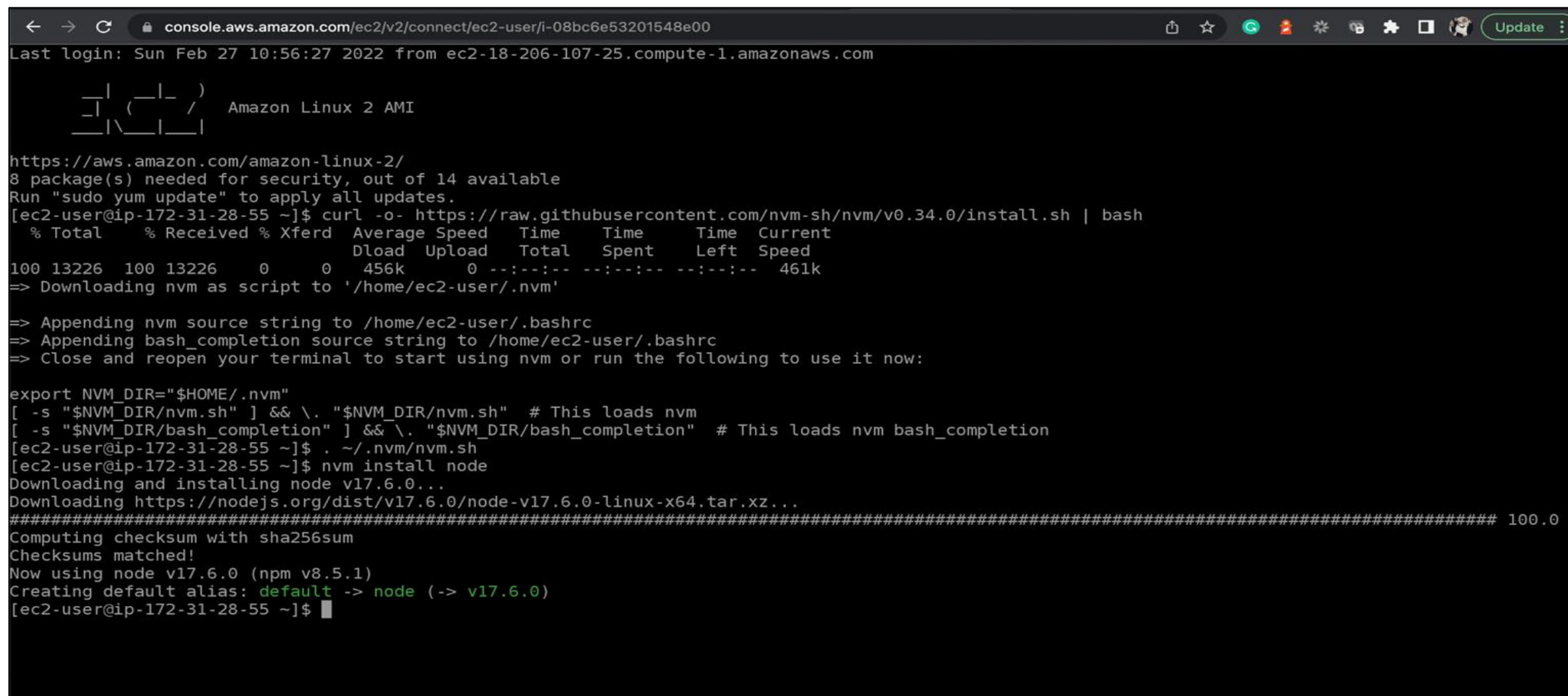
```
← → ↺ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-08bc6e53201548e00
┌─┐ ┌─┐ ┌─┐
└─┘ └─┘ └─┘ Amazon Linux 2 AMI
https://aws.amazon.com/amazon-linux-2/
package(s) needed for security, out of 14 available
run "sudo yum update" to apply all updates.
ec2-user@ip-172-31-28-55 ~]$
```

i-08bc6e53201548e00 (FrontEndEC2)
Public IPs: 3.88.180.84 Private IPs: 172.31.28.55

Configure Node on EC2

Configure the Jenkins Pipeline Project by passing GitHub repository URL.

Reference link on AWS Documentation: <https://docs.aws.amazon.com/sdk-for-javascript/v2/developer-guide/setting-up-node-on-ec2-instance.html>



```
console.aws.amazon.com/ec2/v2/connect/ec2-user/i-08bc6e53201548e00
Last login: Sun Feb 27 10:56:27 2022 from ec2-18-206-107-25.compute-1.amazonaws.com

 _ _ | ( _ _ )
 _ _ | \ _ _ |   Amazon Linux 2 AMI

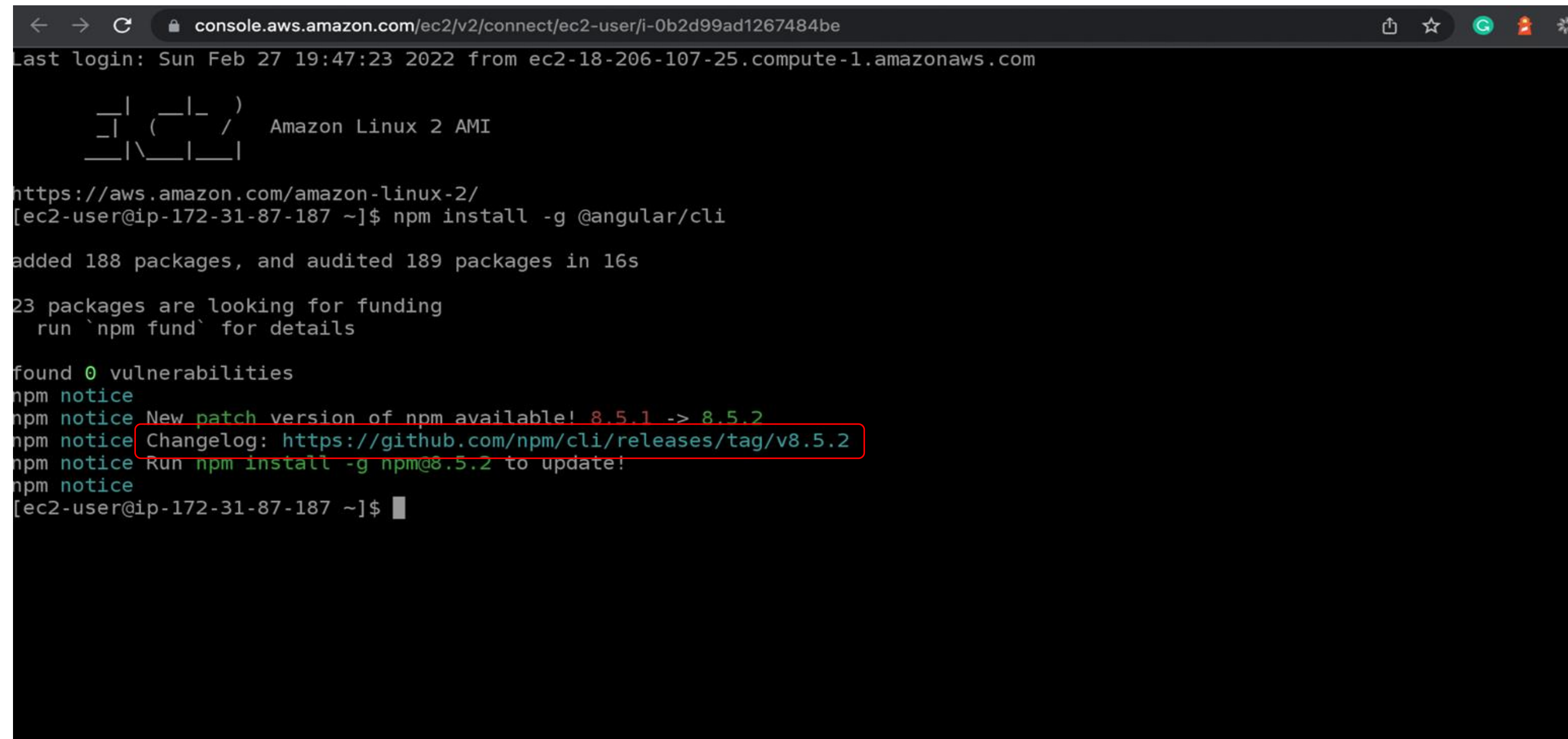
https://aws.amazon.com/amazon-linux-2/
8 package(s) needed for security, out of 14 available
Run "sudo yum update" to apply all updates.
[ec2-user@ip-172-31-28-55 ~]$ curl -o- https://raw.githubusercontent.com/nvm-sh/nvm/v0.34.0/install.sh | bash
% Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
           Dload  Upload   Total   Spent    Left  Speed
100 13226  100 13226    0     0  456k      0  --:--:-- --:--:-- --:--:--  461k
=> Downloading nvm as script to '/home/ec2-user/.nvm'

=> Appending nvm source string to /home/ec2-user/.bashrc
=> Appending bash_completion source string to /home/ec2-user/.bashrc
=> Close and reopen your terminal to start using nvm or run the following to use it now:

export NVM_DIR="$HOME/.nvm"
[ -s "$NVM_DIR/nvm.sh" ] && \. "$NVM_DIR/nvm.sh" # This loads nvm
[ -s "$NVM_DIR/bash_completion" ] && \. "$NVM_DIR/bash_completion" # This loads nvm bash_completion
[ec2-user@ip-172-31-28-55 ~]$ . ~/.nvm/nvm.sh
[ec2-user@ip-172-31-28-55 ~]$ nvm install node
Downloading and installing node v17.6.0...
Downloading https://nodejs.org/dist/v17.6.0/node-v17.6.0-linux-x64.tar.xz...
##### 100.0
Computing checksum with sha256sum
Checksums matched!
Now using node v17.6.0 (npm v8.5.1)
Creating default alias: default -> node (-> v17.6.0)
[ec2-user@ip-172-31-28-55 ~]$
```

Configure Angular CLI on EC2

Add Angular CLI to run the Angular project.



```
← → ↺ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
Last login: Sun Feb 27 19:47:23 2022 from ec2-18-206-107-25.compute-1.amazonaws.com

  _|_  _|_  )
 _|_  ( _|_ /  Amazon Linux 2 AMI
 _|_ \ _|_ |

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-87-187 ~]$ npm install -g @angular/cli

added 188 packages, and audited 189 packages in 16s

23 packages are looking for funding
  run `npm fund` for details

found 0 vulnerabilities
npm notice
npm notice New patch version of npm available! 8.5.1 -> 8.5.2
npm notice Changelog: https://github.com/npm/cli/releases/tag/v8.5.2
npm notice Run npm install -g npm@8.5.2 to update!
npm notice
[ec2-user@ip-172-31-87-187 ~]$
```

Configure Git on EC2 Instance

Configure Git.

```
ec2-user@ip-172-31-28-55 ~]$ sudo yum update -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
No packages marked for update
ec2-user@ip-172-31-28-55 ~]$ sudo yum install git -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Package git-2.32.0-1.amzn2.0.1.x86_64 already installed and latest version
Nothing to do
ec2-user@ip-172-31-28-55 ~]$ git version
git version 2.32.0
ec2-user@ip-172-31-28-55 ~]$
```


Configure Docker on EC2 Instance

Install docker for containerizing the Angular apps.

```
← → ↻ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
208 packages excluded due to repository priority protections
Package docker-20.10.7-5.amzn2.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-87-187 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2022-02-27 20:53:28 UTC; 17min ago
     Docs: https://docs.docker.com
   Process: 3052 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Process: 3039 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
  Main PID: 3057 (dockerd)
    Tasks: 7
   Memory: 42.7M
    CGroup: /system.slice/docker.service
            └─3057 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.581813062Z" level=info msg="[graphdriver] using prior storage driver: overlay2"
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594013561Z" level=warning msg="Your kernel does not support swap memory limit. Swap limit may be locked by cgroupswap."
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594040114Z" level=warning msg="Your kernel does not support swap memory limit. Swap limit may be locked by cgroupswap."
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594919996Z" level=info msg="Loading containerd: done"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.247258940Z" level=info msg="Default bridge set to veth"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.415246900Z" level=info msg="Loading containerd: done"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.560515372Z" level=info msg="Docker daemon"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.562805066Z" level=info msg="Daemon has completed initialization"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal systemd[1]: Started Docker Application Container Engine.
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.689510365Z" level=info msg="API listen on /var/run/docker.sock"
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-87-187 ~]$
```

Install Docker : sudo yum install docker

Configure Docker on EC2 Instance

Add ec2-user to docker group, which can execute Docker commands without using sudo and exit the terminal. Re-login to make the change effective.

Command: `sudo usermod -a -G docker ec2-user`

```
console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
208 packages excluded due to repository priority protections
Package docker-20.10.7-5.amzn2.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-87-187 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2022-02-27 20:53:28 UTC; 17min ago
     Docs: https://docs.docker.com
   Process: 3052 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Process: 3039 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
  Main PID: 3057 (dockerd)
    Tasks: 7
   Memory: 42.7M
   CGroup: /system.slice/docker.service
           └─3057 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.581813062Z" level=info msg="[graphdriver] using prior storage driver: overlay2"
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594013561Z" level=warning msg="Your kernel does not support swap limit capabilities. To enable swap support, you need to mount the swap device with the 'x' option. You may want to add 'swap' to the 'features' field in the 'storage' field in the 'daemon' configuration file."
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594040114Z" level=warning msg="Your kernel does not support swap limit capabilities. To enable swap support, you need to mount the swap device with the 'x' option. You may want to add 'swap' to the 'features' field in the 'daemon' configuration file."
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594919996Z" level=info msg="Loading container config files"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.247258940Z" level=info msg="Default bridge (veth) is deprecated. Please use the 'veth' network driver instead."
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.415246900Z" level=info msg="Loading containerd"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.560515372Z" level=info msg="Docker daemon"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.562805066Z" level=info msg="Daemon has completed initialization"
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal systemd[1]: Started Docker Application Container Engine.
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.689510365Z" level=info msg="API listen on /var/run/docker.sock"
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-87-187 ~]$
```

Configure Docker on EC2 Instance

Enable docker service : **sudo systemctl enable docker**
Start docker service : **sudo systemctl start docker**

```
← → ↻ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install docker -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
amzn2-core
208 packages excluded due to repository priority protections
Package docker-20.10.7-5.amzn2.x86_64 already installed and latest version
Nothing to do
[ec2-user@ip-172-31-87-187 ~]$ sudo systemctl status docker
● docker.service - Docker Application Container Engine
   Loaded: loaded (/usr/lib/systemd/system/docker.service; enabled; vendor preset: disabled)
   Active: active (running) since Sun 2022-02-27 20:53:28 UTC; 17min ago
     Docs: https://docs.docker.com
   Process: 3052 ExecStartPre=/usr/libexec/docker/docker-setup-runtimes.sh (code=exited, status=0/SUCCESS)
   Process: 3039 ExecStartPre=/bin/mkdir -p /run/docker (code=exited, status=0/SUCCESS)
  Main PID: 3057 (dockerd)
    Tasks: 7
   Memory: 42.7M
    CGroup: /system.slice/docker.service
            └─3057 /usr/bin/dockerd -H fd:// --containerd=/run/containerd/containerd.sock --default-ulimit nofile=32768:65536

Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.581813062Z" level=info msg="[graphdriver] using prior storage driver: overlay2"
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594013561Z" level=warning msg="Your kernel does not support swap limit capabilities. To enable swap support, you need to mount the swap device with the 'noauto' option."
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594040114Z" level=warning msg="Your kernel does not support swap limit capabilities. To enable swap support, you need to mount the swap device with the 'noauto' option."
Feb 27 20:53:27 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:27.594919996Z" level=info msg="Loading containers: start."
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.247258940Z" level=info msg="Default bridge (docker0) is already up. To create new bridges, use 'docker network create'. If you don't want the bridge on the host, use 'docker network create --opt disable-chaining=true'."
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.415246900Z" level=info msg="Loading containers: done."
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.560515372Z" level=info msg="Docker daemon 20.10.7-5.amzn2.x86_64 is starting."
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.562805066Z" level=info msg="Daemon has completed initialization."
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal systemd[1]: Started Docker Application Container Engine.
Feb 27 20:53:28 ip-172-31-87-187.ec2.internal dockerd[3057]: time="2022-02-27T20:53:28.689510365Z" level=info msg="API listen on /var/run/docker.sock"
Hint: Some lines were ellipsized, use -l to show in full.
[ec2-user@ip-172-31-87-187 ~]$
```

Check the docker service.

Configure Jenkins on EC2 Instance

Configure Jenkins on EC2 Instance. Install Java and use these commands.

Install JDK 8 : `sudo yum update -y` or `sudo yum install java-1.8.0-openjdk`.

```
← → ↻ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
208 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package jenkins.noarch 0:2.319.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                               Arch          Version           Repository
=====
Installing:
jenkins                               noarch        2.319.3-1.1       jenkins

Transaction Summary
=====
Install 1 Package

Total download size: 69 M
Installed size: 69 M
Downloading packages:
jenkins-2.319.3-1.1.noarch.rpm        | 69 MB  00:00:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.319.3-1.1.noarch
   Verifying : jenkins-2.319.3-1.1.noarch

Installed:
jenkins.noarch 0:2.319.3-1.1

Complete!
[ec2-user@ip-172-31-87-187 ~]$
```

Configure Jenkins on EC2 Instance

Install JDK 11 – Java : **sudo amazon-linux-extras install java-openjdk11.**

Install JDK 11 Development – javac : **sudo yum install java-11-openjdk-devel.**

```
← → ↻ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
208 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.319.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                        Arch          Version           Repository
=====
Installing:
jenkins                        noarch        2.319.3-1.1       jenkins

Transaction Summary
=====
Install 1 Package

Total download size: 69 M
Installed size: 69 M
Downloading packages:
jenkins-2.319.3-1.1.noarch.rpm | 69 MB 00:00:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.319.3-1.1.noarch
  Verifying  : jenkins-2.319.3-1.1.noarch

Installed:
jenkins.noarch 0:2.319.3-1.1

Complete!
[ec2-user@ip-172-31-87-187 ~]$
```

Configure Jenkins on EC2 Instance

Configure Java Version 8 or 11 : **sudo alternatives --config java.**

Read OS name : **cat /etc/os-release**

```
console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
208 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.319.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                                Arch                                Version                                Repository
=====
Installing:
jenkins                                noarch                              2.319.3-1.1                            jenkins
=====
Transaction Summary
=====
Install 1 Package

Total download size: 69 M
Installed size: 69 M
Downloading packages:
jenkins-2.319.3-1.1.noarch.rpm          | 69 MB  00:00:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.319.3-1.1.noarch
  Verifying  : jenkins-2.319.3-1.1.noarch

Installed:
jenkins.noarch 0:2.319.3-1.1

Complete!
[ec2-user@ip-172-31-87-187 ~]$
```

Configure Jenkins on EC2 Instance

Jenkins Installation document: [<https://www.jenkins.io/doc/book/installing/linux/>].
Follow the instructions.

```
sudo amazon-linux-extras install epel
sudo yum-config-manager --enable epel
sudo yum install daemonize -y
```

```
← → ↺ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
208 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
--> Package jenkins.noarch 0:2.319.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                        Arch          Version           Repository
=====
Installing:
jenkins                        noarch        2.319.3-1.1       jenkins

Transaction Summary
=====
Install 1 Package

Total download size: 69 M
Installed size: 69 M
Downloading packages:
jenkins-2.319.3-1.1.noarch.rpm | 69 MB 00:00:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.319.3-1.1.noarch
  Verifying  : jenkins-2.319.3-1.1.noarch

Installed:
jenkins.noarch 0:2.319.3-1.1

Complete!
[ec2-user@ip-172-31-87-187 ~]$
```


Configure Jenkins on EC2 Instance

```
sudo yum install jenkins -y
sudo systemctl daemon-reload
sudo systemctl enable jenkins
sudo systemctl start jenkins
sudo systemctl status jenkins
```

```
← → ↻ console.aws.amazon.com/ec2/v2/connect/ec2-user/i-0b2d99ad1267484be
[ec2-user@ip-172-31-87-187 ~]$ sudo yum install jenkins -y
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
208 packages excluded due to repository priority protections
Resolving Dependencies
--> Running transaction check
---> Package jenkins.noarch 0:2.319.3-1.1 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

=====
Package                               Arch          Version           Repository
=====
Installing:
jenkins                               noarch        2.319.3-1.1       jenkins
=====
Transaction Summary
=====
Install 1 Package

Total download size: 69 M
Installed size: 69 M
Downloading packages:
jenkins-2.319.3-1.1.noarch.rpm        | 69 MB 00:00:
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : jenkins-2.319.3-1.1.noarch
  Verifying  : jenkins-2.319.3-1.1.noarch

Installed:
jenkins.noarch 0:2.319.3-1.1

Complete!
[ec2-user@ip-172-31-87-187 ~]$
```

Configure Jenkins on EC2

Edit the inbound rules of security group associated with EC2 instance.

The screenshot shows the AWS Management Console interface for editing inbound rules of a security group. The breadcrumb trail indicates the path: EC2 > Security Groups > sg-02326a39c3dd3f1a6 - launch-wizard-1 > Edit inbound rules. The main heading is "Edit inbound rules" with an "Info" link. Below this, a note states: "Inbound rules control the incoming traffic that's allowed to reach the instance."

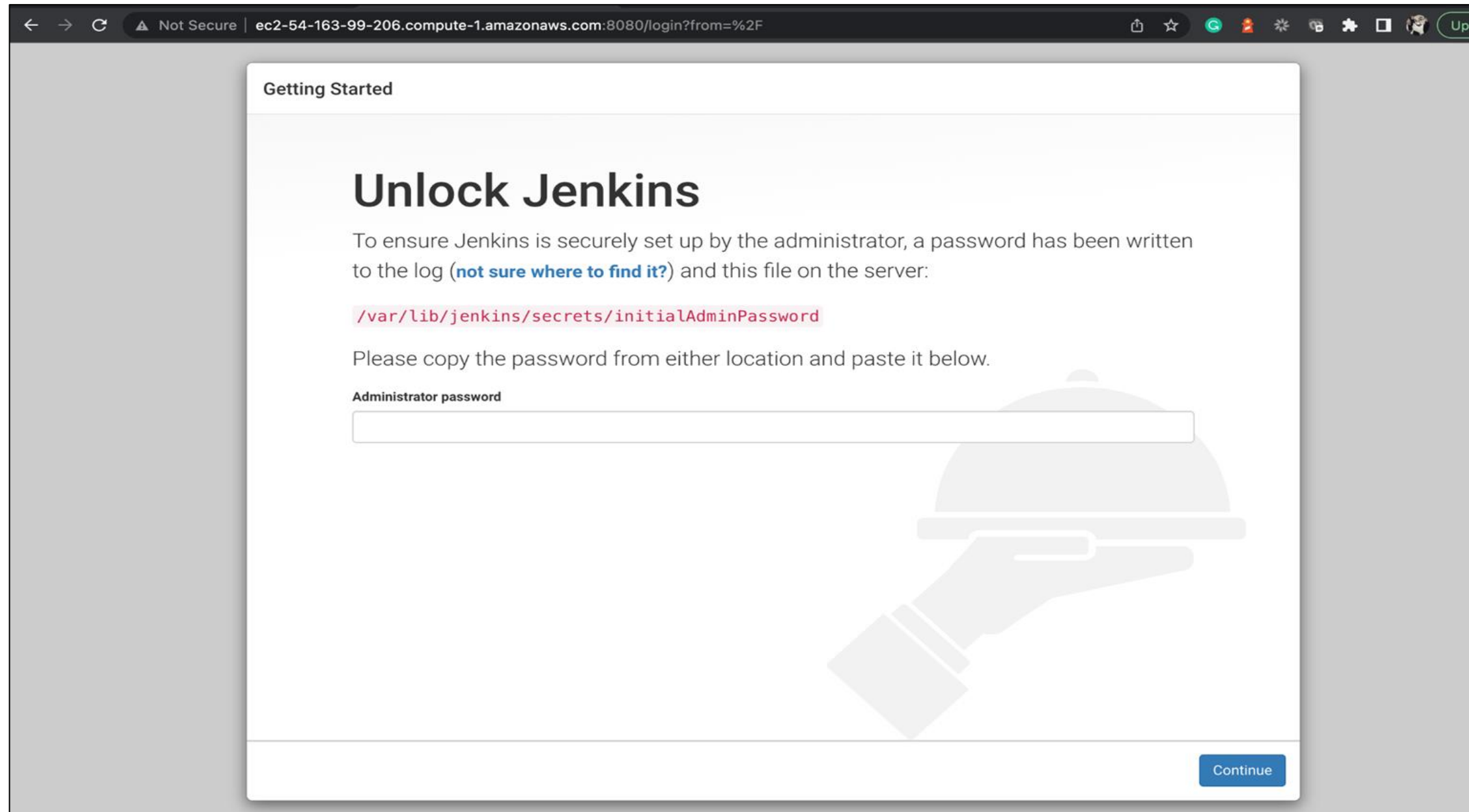
The "Inbound rules" section contains a table with the following columns: Security group rule ID, Type, Protocol, Port range, Source, and Description - optional. There are two rules listed:

Security group rule ID	Type	Protocol	Port range	Source	Description - optional
sgr-0786ae8ec9f00898a	SSH	TCP	22	Custom	
sgr-0d6217835c899c230	Custom TCP	TCP	8080	Custom	

A red box highlights the "8080" port range in the second rule, with a red arrow pointing down to a green callout box that says "Open port 8080 so that Jenkins runs." The interface also includes an "Add rule" button and "Cancel", "Preview changes", and "Save rules" buttons at the bottom.

Access Jenkins on Port 8080 of EC2

From here, proceed by **logging** into the Jenkins with the initial 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
```

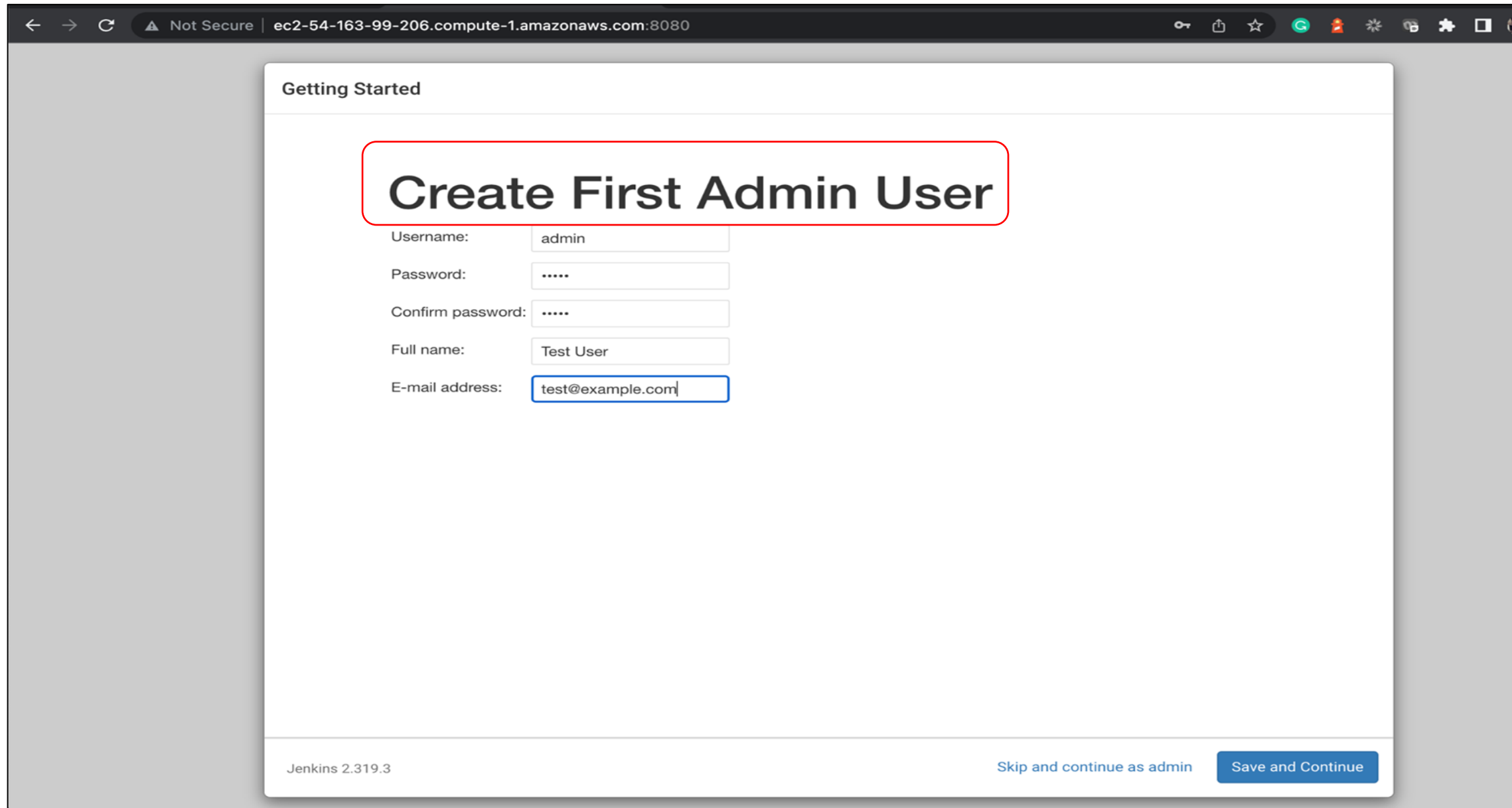
Please copy the password from either location and paste it below.

Administrator password

Continue

Access Jenkins on Port 8080 of EC2

Create a new user.



The screenshot shows a web browser window with the address bar displaying "Not Secure | ec2-54-163-99-206.compute-1.amazonaws.com:8080". The page title is "Getting Started". The main heading is "Create First Admin User", which is highlighted with a red rounded rectangle. Below the heading are five input fields: "Username:" with the value "admin", "Password:" with masked characters ".....", "Confirm password:" with masked characters ".....", "Full name:" with the value "Test User", and "E-mail address:" with the value "test@example.com". The "E-mail address" field is highlighted with a blue rounded rectangle. At the bottom left, it says "Jenkins 2.319.3". At the bottom right, there are two buttons: "Skip and continue as admin" and "Save and Continue".

Getting Started

Create First Admin User

Username:

Password:

Confirm password:

Full name:

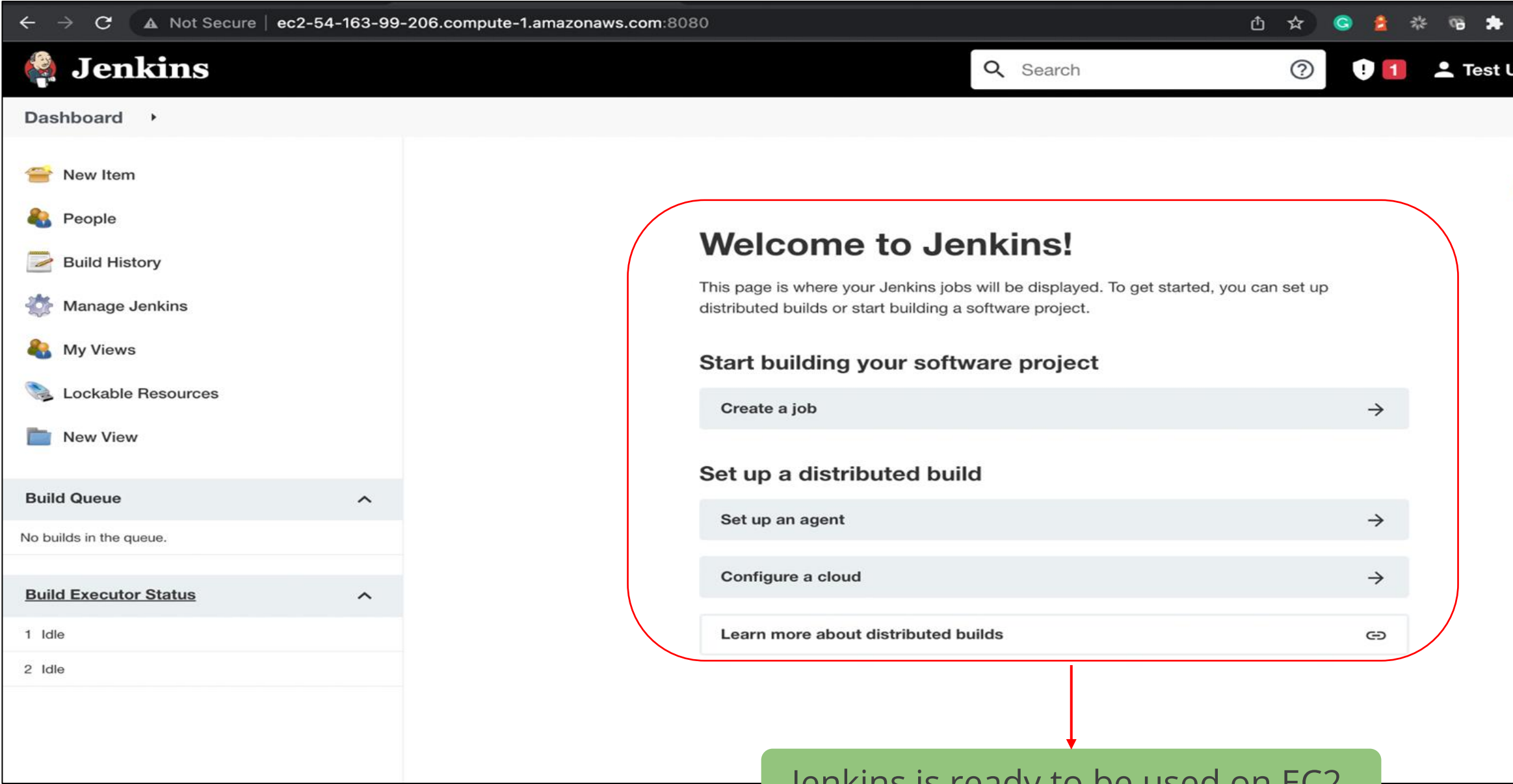
E-mail address:

Jenkins 2.319.3

[Skip and continue as admin](#) [Save and Continue](#)

Configure Jenkins on EC2

Install desired plugins, e.g., NodeJS Plugin to be used for npm command on EC2.



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `ec2-54-163-99-206.compute-1.amazonaws.com:8080`. The Jenkins logo and name are in the top left. A search bar and user profile (Test User) are in the top right. The left sidebar contains navigation links: New Item, People, Build History, Manage Jenkins, My Views, Lockable Resources, and New View. The main content area displays a 'Welcome to Jenkins!' message, followed by instructions and a list of actions: 'Create a job', 'Set up a distributed build' (with sub-options 'Set up an agent' and 'Configure a cloud'), and 'Learn more about distributed builds'. A red rounded rectangle highlights the 'Welcome to Jenkins!' section and the 'Set up a distributed build' options. A red arrow points from the bottom of this rectangle to a green box at the bottom of the slide.

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 →

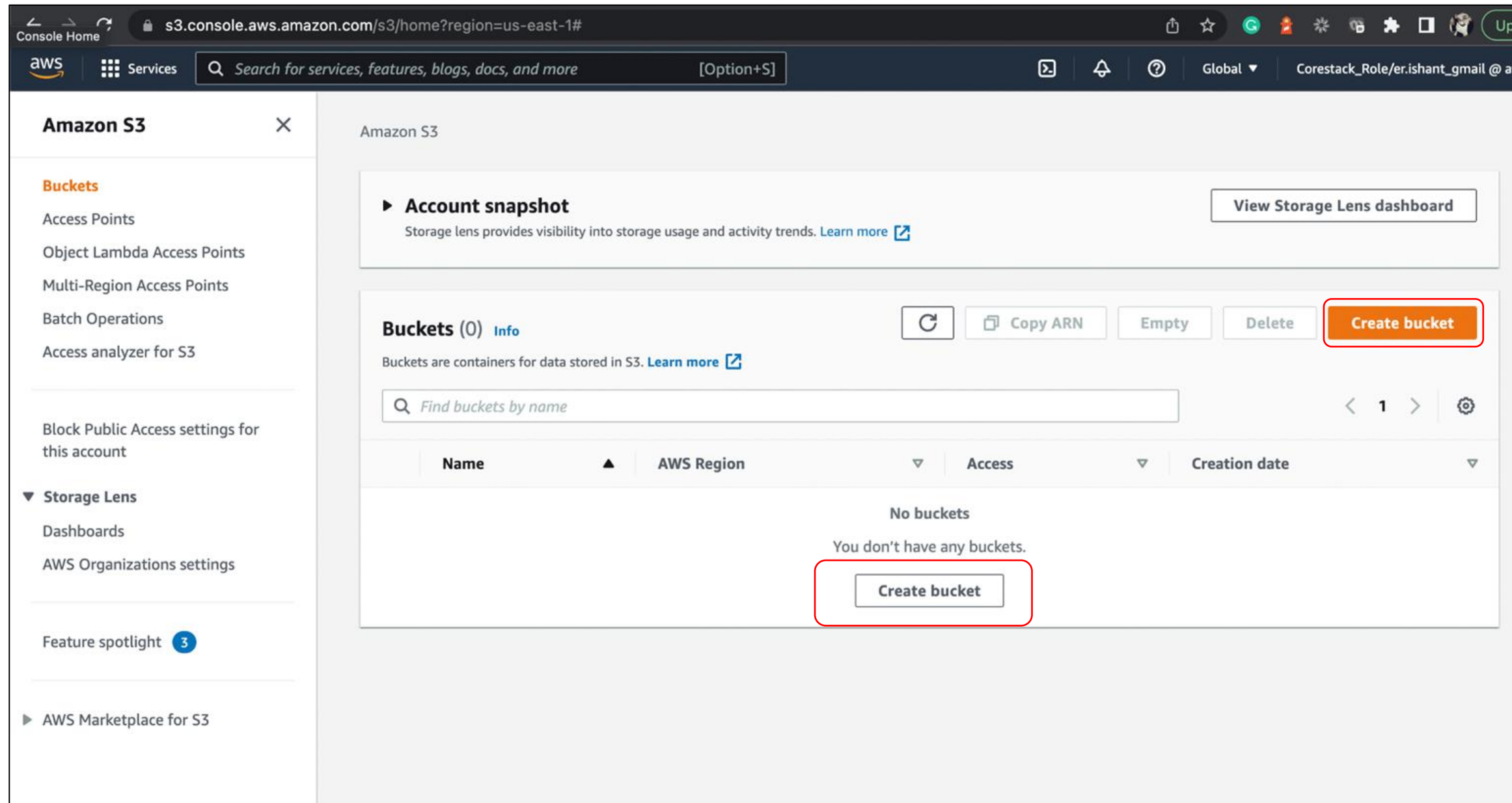
Learn more about distributed builds ↗

Jenkins is ready to be used on EC2.

Create AWS S3 Bucket

Create S3 Bucket

From the S3 dashboard, click on **Create bucket**.



Create S3 Bucket

Make bucket accessible to public for the images to be read or written by Angular apps.

Amazon S3 > Create bucket

Create bucket [Info](#)

Buckets are containers for data stored in S3. [Learn more](#)

General configuration

Bucket name

Bucket name must be unique and must not contain spaces or uppercase letters. [See rules for bucket naming](#)

AWS Region

US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Object Ownership [Info](#)

Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.

☒ **ACLs disabled (recommended)**
All objects in this bucket are owned by this account. Access to this bucket and its objects is specified using only policies.

☐ **ACLs enabled**
Objects in this bucket can be owned by other AWS accounts. Access to this bucket and its objects can be specified using ACLs.

Give a name and choose the default settings for the bucket.

Create S3 Bucket

The bucket is **configured successfully**.

The screenshot displays the AWS S3 console interface. The left sidebar contains navigation options: Buckets, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, Access analyzer for S3, Storage Lens, Dashboards, AWS Organizations settings, Feature spotlight (3), and AWS Marketplace for S3. The main content area shows the 'Buckets (1)' section with a table of existing buckets. A red box highlights the first bucket, 'estore-app-bucket', which is located in the 'US East (N. Virginia) us-east-1' region, has public access enabled ('Objects can be public'), and was created on February 28, 2022, at 00:39:27 (UTC+05:30). Above the table, there are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. The 'Create bucket' button is highlighted in orange.

Name	AWS Region	Access	Creation date
estore-app-bucket	US East (N. Virginia) us-east-1	Objects can be public	February 28, 2022, 00:39:27 (UTC+05:30)

TECHNOLOGY

Computing

Caltech

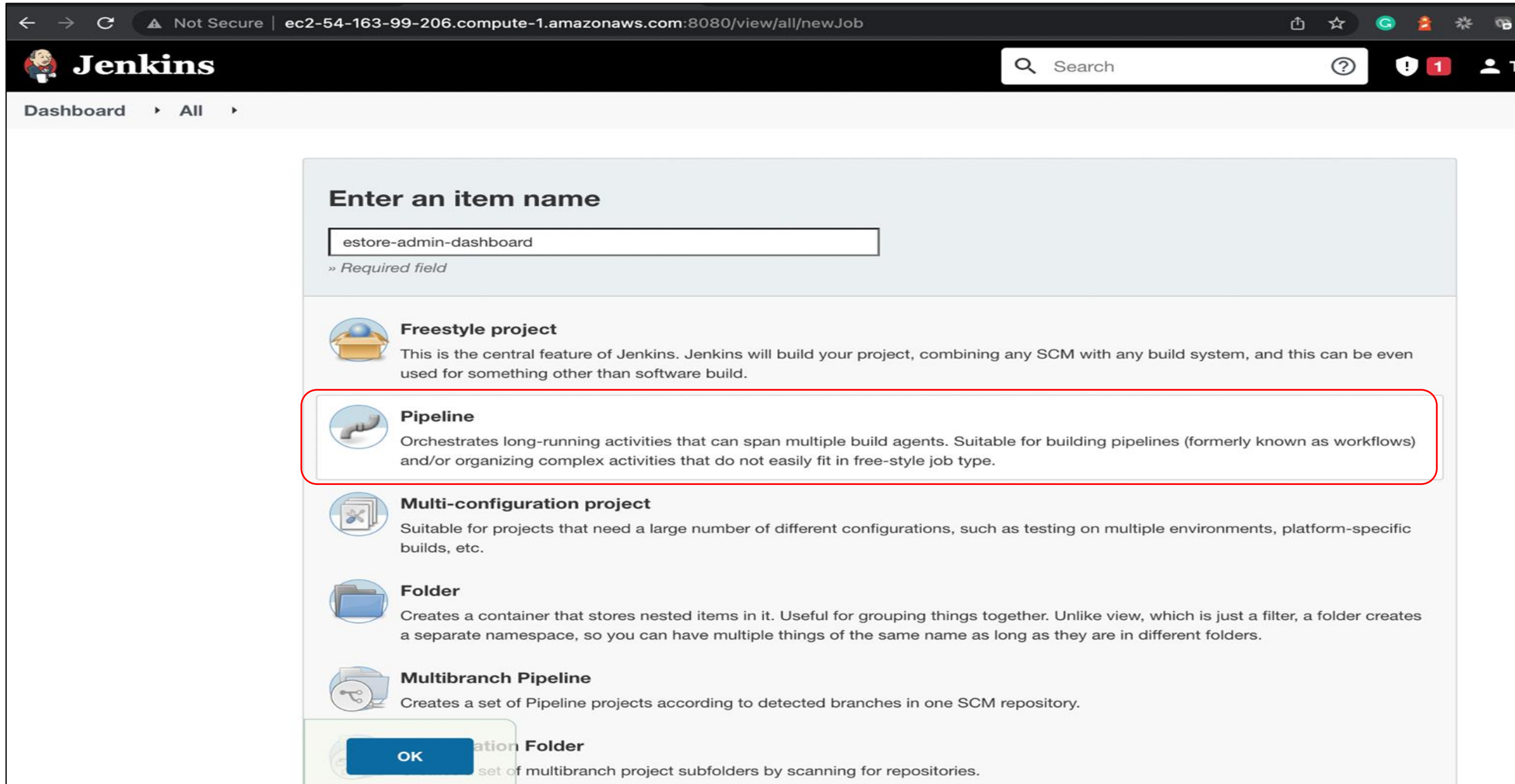
**Center for Technology &
Management Education**

Build and Deploy with Jenkins Docker

Set Up Jenkins for the Angular Web-Admin Project

Create a Jenkins Pipeline Project for the Web-Admin Project

Create a new project in Jenkins of type **Pipeline**.



The screenshot shows the Jenkins web interface in a browser. The address bar indicates the URL is `ec2-54-163-99-206.compute-1.amazonaws.com:8080/view/all/newJob`. The Jenkins logo and a search bar are at the top. Below the navigation bar, the page title is "Enter an item name". A text input field contains the name "estore-admin-dashboard". Below the input field, it says "» Required field". A list of project types is shown below: "Freestyle project", "Pipeline", "Multi-configuration project", "Folder", and "Multibranch Pipeline". The "Pipeline" option is highlighted with a red border. At the bottom, there is an "OK" button and a partially visible "Folder" option.

Enter an item name

estore-admin-dashboard

» 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 a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK

Folder
set of multibranch project subfolders by scanning for repositories.

Configure Jenkins Pipeline SCM for the Web-Admin Project

Configure the Jenkins Pipeline Project by passing **GitHub repository URL**.

The screenshot shows the Jenkins configuration interface for a project named 'estore-admin-dashboard'. The 'Pipeline' tab is selected, and the 'Definition' is set to 'Pipeline script from SCM'. The 'SCM' is configured as 'Git'. Under the 'Repositories' section, the 'Repository URL' is set to 'https://github.com/tuser6794/estore-admin-dashboard.git', which is highlighted with a red rectangle. The 'Credentials' are set to '- none -'. The 'Branches to build' section shows the 'Branch Specifier (blank for 'any')' set to '*/master'. The 'Save' button is highlighted with a green box.

Dashboard > estore-admin-dashboard >

General Build Triggers **Advanced Project Options** Pipeline

Advanced...

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/tuser6794/estore-admin-dashboard.git

Credentials

- none - Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

Save Apply

Configure Jenkinsfile in Jenkins for the Web-Admin Project

Script Path contains the Jenkinsfile, which is created in the root directory of our project.

Select the default branch, which is master.

Dashboard ▾ ▸ estore-admin-dashboard ▸

General Build Triggers Advanced Project Options **Pipeline**

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

Add Branch

Repository browser

(Auto)

Additional Behaviours

Add ▾

Script Path

Jenkinsfile

☒ Lightweight checkout

Pipeline Syntax

Save Apply

Configure GitHub Hook Trigger for the Web-Admin Project

Configure GitHub trigger for GITScm. Polling and option will work when Jenkins is running with a proper URL instead of localhost.

Dashboard > estore-admin-dashboard >

General Build Triggers Advanced Project Options Pipeline

☐ Build after other projects are built

☐ Build periodically

☒ GitHub hook trigger for GITScm polling

☐ Poll SCM

☐ Disable this project

☐ Quiet period

☐ Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Advanced...

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/tuser6794/estore-admin-dashboard.git

Credentials

- none - Add

Save Apply

Advanced...

Configure the same using ngrok.

Run the Jenkins Pipeline Project for the Web-Admin Project

Notice stages appearing as mentioned in the Jenkinsfile.
Source > Test > Build > Containerize > Deploy

The screenshot shows the Jenkins web interface for a pipeline named 'estore-admin-dashboard'. On the left sidebar, the 'Build Now' button is highlighted with a red box. A red arrow points from this button to a green callout box that says 'Click on Build Now and build project.' The main area displays the 'Stage View' for the pipeline, which includes a table of stage execution times for two builds.

Pipeline estore-admin-dashboard
Angular Admin Web Project

Build Now (highlighted in red box)

Stage View

	Declarative: Checkout SCM	Source	Test	Build	Containerize	Deploy
Average stage times: (Average full run time: ~1min 39s)	326ms	14s	17s	27s	1min 13s	565ms
#2 Feb 26 13:01 No Changes	225ms	3s	17s	25s	1min 13s	565ms
#1 Feb 26 11:43 No Changes	427ms	26s	18s	30s		

Permalinks

Set Up Jenkins for the Angular End-User Project

Create a Jenkins Pipeline Project for the End-User Project

Create a new project in Jenkins of type **Pipeline**.

← → ↻ Not Secure | ec2-54-166-196-41.compute-1.amazonaws.com:8080/view/all/newJob

Jenkins Search ? 1

Dashboard > All >

Enter an item name

estore-end-user
» 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 a separate namespace, so you can have multiple things of the same name as long as they are in different folders.
- Multibranch Pipeline**
Creates a set of Pipeline projects according to detected branches in one SCM repository.

OK Multibranch Pipeline Folder
set of multibranch project subfolders by scanning for repositories.

Configure Jenkins Pipeline SCM for the End-User Project

Configure the Jenkins Pipeline Project by passing **GitHub repository URL**.

Dashboard > estore-end-user >

General Build Triggers Advanced Project Options **Pipeline**

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/tuser6794/estore-end-user.git

Credentials

- none - Add

Advanced...

Add Repository

Branches to build

Branch Specifier (blank for 'any')

*/master

Save Apply

Configure Jenkinsfile in Jenkins for the End-User Project

Script Path contains the Jenkinsfile, which is created in the root directory of our project.

Select the default branch, which is master.

The screenshot shows the Jenkins configuration interface for a project named 'estore-end-user'. The 'Pipeline' tab is selected, and the 'Branches to build' section is highlighted with a red box. Inside this section, the 'Branch Specifier (blank for 'any')' field contains the text '*/master'. A green callout box on the left points to this field with the text 'Select the default branch, which is master.' Below the 'Branches to build' section, the 'Repository browser' is set to '(Auto)', and the 'Script Path' is set to 'Jenkinsfile'. The 'Lightweight checkout' checkbox is checked. At the bottom, there are 'Save' and 'Apply' buttons.

Optional: GitHub Hook Trigger for the End-User Project

Configure **GitHub trigger** for GITScm. Polling and option will work when Jenkins is running with a proper URL instead of localhost.

Dashboard > estore-end-user >

General Build Triggers Advanced Project Options Pipeline

☐ Build after other projects are built

☐ Build periodically

☒ GitHub hook trigger for GITScm polling

☐ Poll SCM

☐ Disable this project

☐ Quiet period

☐ Trigger builds remotely (e.g., from scripts)

Advanced Project Options

Advanced...

Pipeline

Definition

Pipeline script from SCM

SCM

Git

Repositories

Repository URL

https://github.com/tuser6794/estore-end-user.git

Credentials

- none - Add

Save Apply

Advanced...

Configure the same using ngrok.

Run the Jenkins Pipeline Project for the End-User Project

Notice stages appearing as mentioned in the Jenkinsfile.
Source > Test > Build > Containerize > Deploy

The screenshot shows the Jenkins web interface for the pipeline 'estore-end-user'. The left sidebar contains navigation options: Back to Dashboard, Status, Changes, Build Now (highlighted with a red box), Configure, Delete Pipeline, Full Stage View, Rename, and Pipeline Syntax. The main area displays the 'Pipeline estore-end-user' page with a 'Recent Changes' section and a 'Stage View' table. A green callout box with an arrow points to the 'Build Now' button, containing the text 'Click on Build Now and build project.' The 'Stage View' table shows the following data:

	Declarative: Checkout SCM	Source	Test	Build	Containerize	Deploy
Average stage times: (Average full run time: ~1min 30s)	305ms	9s	12s	30s	1min 12s	867ms
#2 Feb 26 13:25 1 commit	254ms	4s	11s	29s	1min 12s	867ms
#1 Feb 26 12:17 No Changes	356ms	15s	13s	30s		

Below the table is the 'Permalinks' section with up and down arrows.

Key Takeaways

- EC2 instance is set up in AWS.
- Tools and dev kits are configured on EC2 instance.
- Jenkins Pipeline is built to dockerize the Angular apps.
- Angular apps are made live.



Before the Next Class

You have successfully completed this session.

For our next end user module discussion, you should:

- Review AWS services
- Explore steps to create EC2 and S3 buckets
- Explore how to connect to EC2 instance
- Review Jenkins
- Review Docker commands



What's Next?

Now we have finished our classes and design Pattern for the back-end project with respect to end-user module.

In our next live session, we will:

- See how to create EC2 Instance on AWS
- See how to work with RDS MySQL
- Deploy Java back end on EC2
- Use Jenkins and Docker as DevOps tools on EC2

