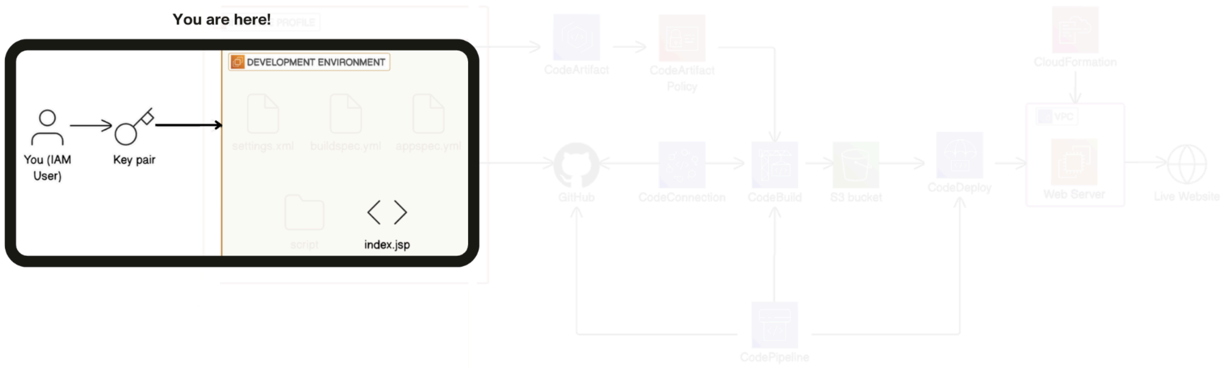
Project 1: Set Up a Web App in the Cloud



Objective:

- Set up a web app using AWS
- Connect your EC2 instance with VSCode
- Install tools to help set up your web app within VS Code

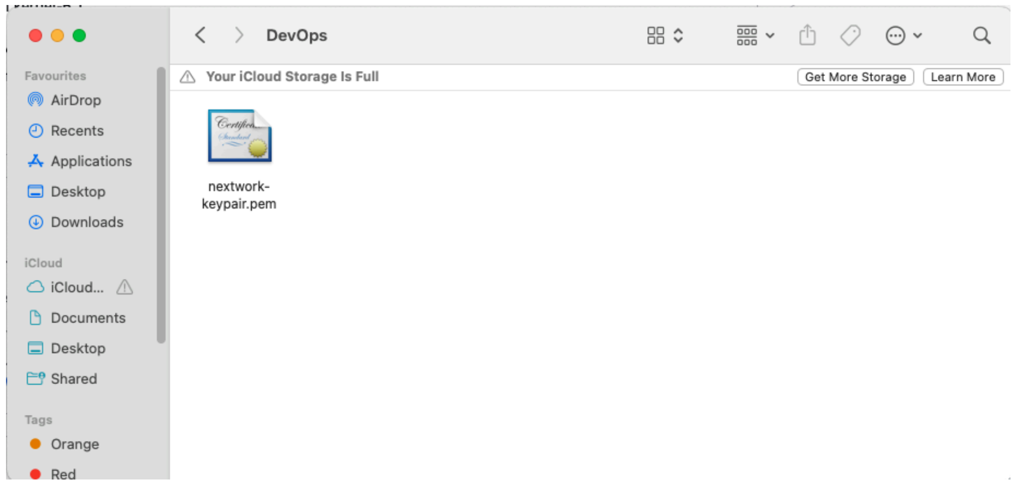
Step 1: Login with your IAM user on AWS

Visit the AWS Management console and login with your IAM user. It is best practice to avoid using the root user for routine operations or day-to-day tasks.

Step 2: Launch an EC2 instance

In this step we are launching an EC2 instance as we are running the web app in the cloud. This EC2 instance will be the virtual server to house our development work.

When launching the EC2 instance, create a key pair that can be downloaded as a private key. This will allow you to securely access the EC2 instance.



We will allow SSH traffic from ‘my IP address’ when creating the EC2 instance to allow us to access the remote server (EC2 instance). It will allow us to transfer data back and forth with our EC2 instance once we are connected to it. Thus, enabling SSH traffic from my IP address will help us to securely connect to our EC2 instance later.

Step 3: Set up terminal in VSCode

In this step we will:

1. Set up a terminal in VS Code
2. Change the permissions of your .pem file

VS Code is an IDE that we will use today to write and edit our web app’s code. It has some very useful extensions which will allow us to directly connect to an EC2 instance.

<u>Setting up a terminal in VS Code:</u>

The first command we run in the terminal section of VS Code is:

cd ~/Desktop/DevOps

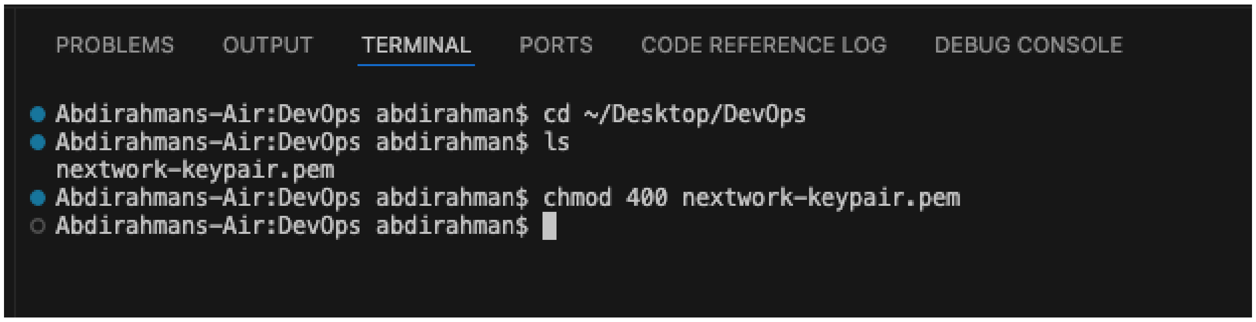
This command navigates the terminal to the DevOps folder we created earlier.

Next, we can run the list command (ls) to check the active directories that exist in the DevOps folder. This is where we should find the private key we created in AWS.

<u>Changing the permissions of the .pem file:</u>

Change the permissions of the .pem file so that is only readable by you (the owner) and restrict access for everyone else. This is done using the command below:

chmod 400 nextwork-keypair.pem

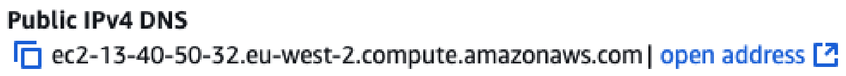


```
Abdirahmans-Air:DevOps abdirahman$ cd ~/Desktop/DevOps
Abdirahmans-Air:DevOps abdirahman$ ls
nextwork-keypair.pem
Abdirahmans-Air:DevOps abdirahman$ chmod 400 nextwork-keypair.pem
Abdirahmans-Air:DevOps abdirahman$
```

Step 4: Connect to you EC2 instance

Within the terminal, we are going to now utilise SSH to connect to our EC2 instance.

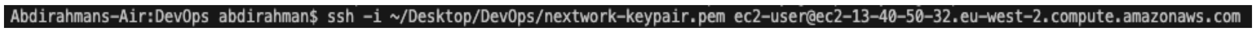
First, return to your AWS management console and find the EC2 instance you launched earlier. Locate the Public IPv4 DNS within the instance details.



Public IPv4 DNS
ec2-13-40-50-32.eu-west-2.compute.amazonaws.com | open address

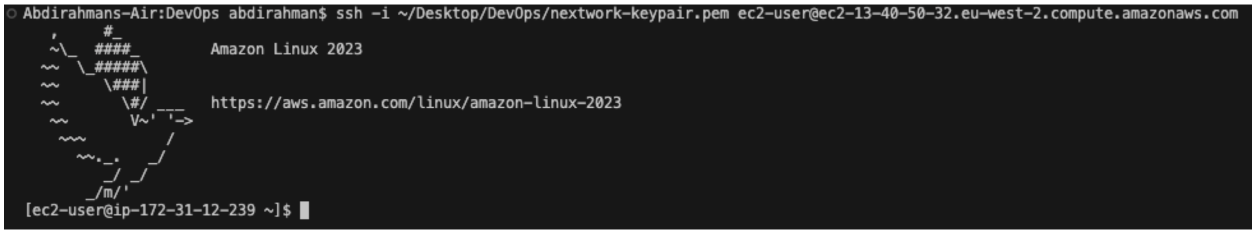
A public IPv4 DNS is a domain name system (DNS) record that maps a human-readable domain name (e.g., example.com) to a public IPv4 address (e.g., 203.0.113.1). It allows users to access resources (like websites or servers) on the internet using a domain name instead of a numeric IP address. Public IPv4 DNS records are managed by DNS servers and are accessible globally.

Copy the public IPv4 DNS details and run the ssh -i command within the terminal:



```
Abdirahmans-Air:DevOps abdirahman$ ssh -i ~/Desktop/DevOps/nextwork-keypair.pem ec2-user@ec2-13-40-50-32.eu-west-2.compute.amazonaws.com
```

We are now connected to our EC2 instance.



```
Abdirahmans-Air:DevOps abdirahman$ ssh -i ~/Desktop/DevOps/nextwork-keypair.pem ec2-user@ec2-13-40-50-32.eu-west-2.compute.amazonaws.com
   ,     #_
   ~\_  ####_        Amazon Linux 2023
  ~~  \_#####\
  ~~     \###|
  ~~       \#/ ___   https://aws.amazon.com/linux/amazon-linux-2023
   ~~       V~' '->
    ~~~         /
      ~~._.   _/
         _/ _/
       _/m/'
[ec2-user@ip-172-31-12-239 ~]$ 
```

Step 5: Install Apache Maven and Amazon Corretto 8

Apache Maven and Amazon Corretto 8 are two helpful tools that will help us to build java web apps from scratch.

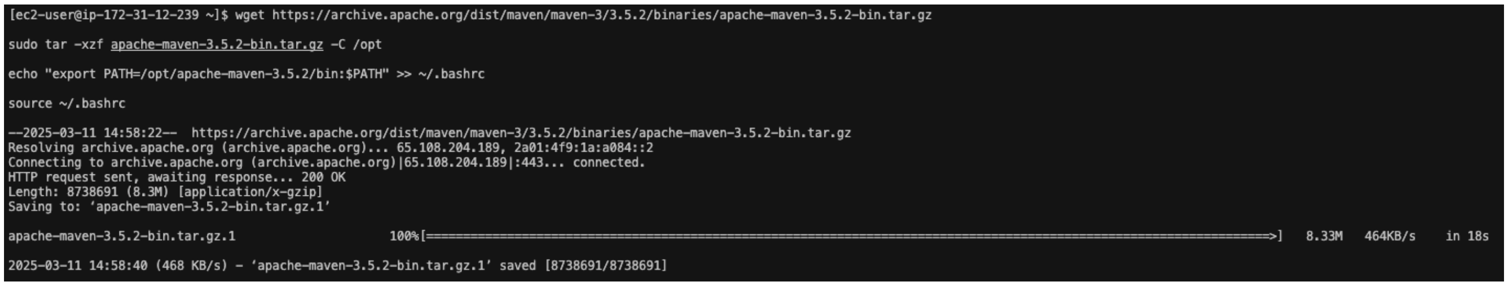
In this step, you're going to:

- Install Apache Maven on your EC2 instance.
- Install Amazon Corretto 8, a version of Java.
- Verify the installations.

<u>Install Apache Maven:</u>

Apache Maven is a tool that helps developers build and organize Java software projects (like this web app). It comes with a lot of use cases, like being a package manager (downloading external pieces of code) and the tool uses archetypes (templates) for spinning up projects like web apps instantly.

Below is the Bash code used to install Apache Maven.

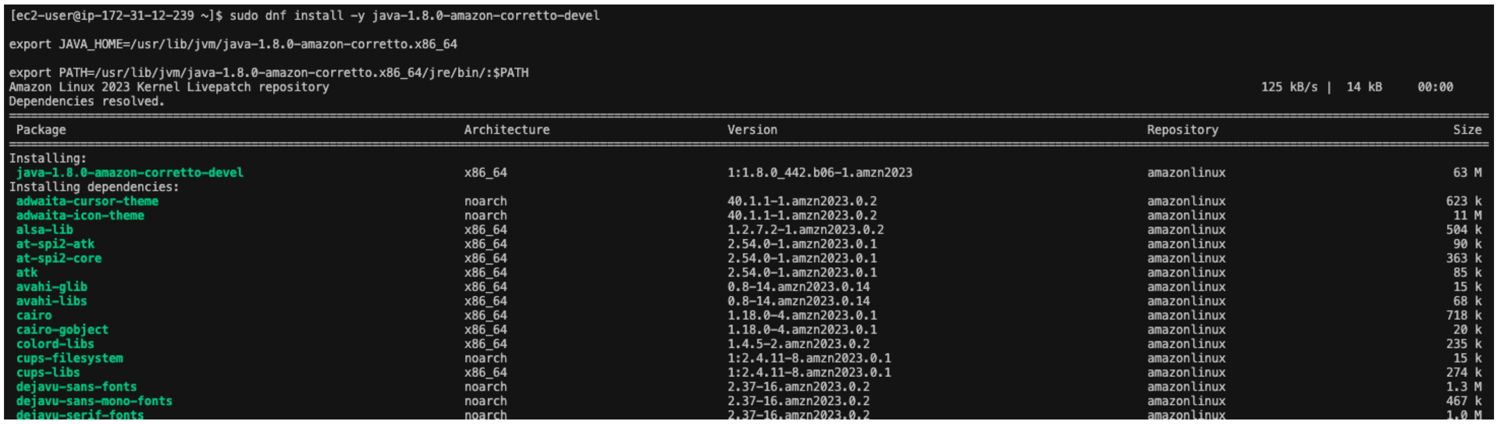


```
[ec2-user@ip-172-31-12-239 ~]$ wget https://archive.apache.org/dist/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz

sudo tar -xzf apache-maven-3.5.2-bin.tar.gz -C /opt

echo "export PATH=/opt/apache-maven-3.5.2/bin:$PATH" >> ~/.bashrc

source ~/.bashrc

--2025-03-11 14:58:22--  https://archive.apache.org/dist/maven/maven-3/3.5.2/binaries/apache-maven-3.5.2-bin.tar.gz
Resolving archive.apache.org (archive.apache.org)... 65.108.204.189, 2a01:4f9:1a:a084::2
Connecting to archive.apache.org (archive.apache.org)|65.108.204.189|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 8738691 (8.3M) [application/x-gzip]
Saving to: ‘apache-maven-3.5.2-bin.tar.gz.1’

apache-maven-3.5.2-bin.tar.gz.1                 100%[===========================================================================================================>]   8.33M   464KB/s    in 18s

2025-03-11 14:58:40 (468 KB/s) - ‘apache-maven-3.5.2-bin.tar.gz.1’ saved [8738691/8738691]
```

Install Java 8 using Amazon Corretto 8:

Java is a commonly used programming language to build many different applications. Amazon Corretto 8 is a version of Java that we are using in this project. Java is useful in this project as it lays the foundation of writing our web app code. Maven also needs Java in order to work.

Below is the Bash code used to install Amazon Corretto 8:

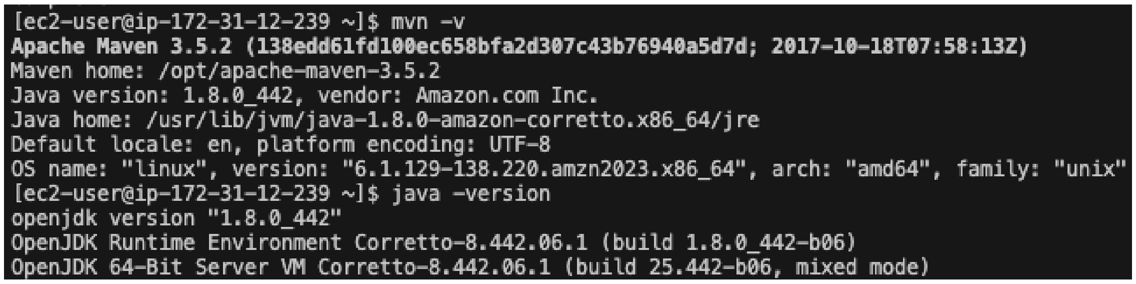


```
[ec2-user@ip-172-31-12-239 ~]$ sudo dnf install -y java-1.8.0-amazon-corretto-devel

export JAVA_HOME=/usr/lib/jvm/java-1.8.0-amazon-corretto.x86_64

export PATH=/usr/lib/jvm/java-1.8.0-amazon-corretto.x86_64/jre/bin/:$PATH
Amazon Linux 2023 Kernel Livepatch repository                                                                                          125 kB/s |  14 kB     00:00
Dependencies resolved.
========================================================================================================================================================================
 Package                                          Architecture                     Version                                         Repository                      Size
========================================================================================================================================================================
Installing:
 java-1.8.0-amazon-corretto-devel                 x86_64                           1:1.8.0_442.b06-1.amzn2023                      amazonlinux                     63 M
Installing dependencies:
 adwaita-cursor-theme                             noarch                           40.1.1-1.amzn2023.0.2                           amazonlinux                    623 k
 adwaita-icon-theme                               noarch                           40.1.1-1.amzn2023.0.2                           amazonlinux                     11 M
 alsa-lib                                         x86_64                           1.2.7.2-1.amzn2023.0.2                          amazonlinux                    504 k
 at-spi2-atk                                      x86_64                           2.54.0-1.amzn2023.0.1                           amazonlinux                     90 k
 at-spi2-core                                     x86_64                           2.54.0-1.amzn2023.0.1                           amazonlinux                    363 k
 atk                                              x86_64                           2.54.0-1.amzn2023.0.1                           amazonlinux                     85 k
 avahi-glib                                       x86_64                           0.8-14.amzn2023.0.14                            amazonlinux                     15 k
 avahi-libs                                       x86_64                           0.8-14.amzn2023.0.14                            amazonlinux                     68 k
 cairo                                            x86_64                           1.18.0-4.amzn2023.0.1                           amazonlinux                    718 k
 cairo-gobject                                    x86_64                           1.18.0-4.amzn2023.0.1                           amazonlinux                     20 k
 colord-libs                                      x86_64                           1.4.5-2.amzn2023.0.2                            amazonlinux                    235 k
 cups-filesystem                                  noarch                           1:2.4.11-8.amzn2023.0.1                         amazonlinux                     15 k
 cups-libs                                        x86_64                           1:2.4.11-8.amzn2023.0.1                         amazonlinux                    274 k
 dejavu-sans-fonts                                noarch                           2.37-16.amzn2023.0.2                            amazonlinux                    1.3 M
 dejavu-sans-mono-fonts                           noarch                           2.37-16.amzn2023.0.2                            amazonlinux                    467 k
 dejavu-serif-fonts                               noarch                           2.37-16.amzn2023.0.2                            amazonlinux                    1.0 M
```

Verify that we have installed both Apache Maven and Amazon Corretto 8 correctly by running the below commands:

mvn -v

java -version

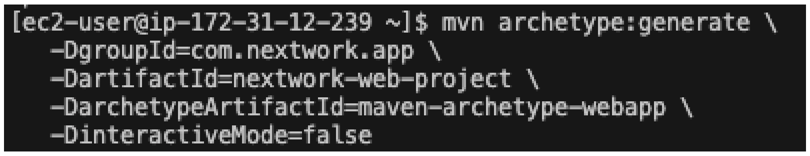


```
[ec2-user@ip-172-31-12-239 ~]$ mvn -v
Apache Maven 3.5.2 (138edd61fd100ec658bfa2d307c43b76940a5d7d; 2017-10-18T07:58:13Z)
Maven home: /opt/apache-maven-3.5.2
Java version: 1.8.0_442, vendor: Amazon.com Inc.
Java home: /usr/lib/jvm/java-1.8.0-amazon-corretto.x86_64/jre
Default locale: en, platform encoding: UTF-8
OS name: "linux", version: "6.1.129-138.220.amzn2023.x86_64", arch: "amd64", family: "unix"
[ec2-user@ip-172-31-12-239 ~]$ java -version
openjdk version "1.8.0_442"
OpenJDK Runtime Environment Corretto-8.442.06.1 (build 1.8.0_442-b06)
OpenJDK 64-Bit Server VM Corretto-8.442.06.1 (build 25.442-b06, mixed mode)
```

Step 6: Create the application

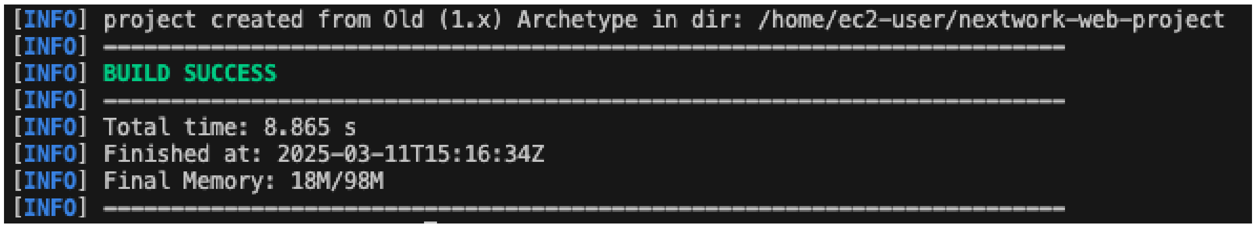
In this step, we are generating our web app inside our EC2 instance using Apache Maven and Java.

First, use Maven to generate a Java web app using the command below:



```
[ec2-user@ip-172-31-12-239 ~]$ mvn archetype:generate \
   -DgroupId=com.nextwork.app \
   -DartifactId=nextwork-web-project \
   -DarchetypeArtifactId=maven-archetype-webapp \
   -DinteractiveMode=false
```

Below is the build success:



```
[INFO] project created from Old (1.x) Archetype in dir: /home/ec2-user/nextwork-web-project
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 8.865 s
[INFO] Finished at: 2025-03-11T15:16:34Z
[INFO] Final Memory: 18M/98M
[INFO] ------------------------------------------------------------------------
```

The command used above tells Maven to generate a web app using an existing template that it has. It also tells Maven to call the generated web app project ‘nextwork-web-project’.

Step 7: Connect VS Code to our EC2 instance

So far, we have connected our EC2 instance to our terminal (i.e. our local computer). Now, we need VS Code to our EC2 instance (using extensions within VS Code)

In this step, you're going to:

1. Install an extension in VS Code.
2. Use the extension to set up a connection between VS Code and your EC2 instance.
3. Explore and edit your Java web app's files using VS Code.

<u>Install Remote - SSH extension:</u>

We installed Remote - SSH (which is an extension within VS Code). This extension allows to connect VS Code directly to remote server (like an EC2 instance) using SSH.

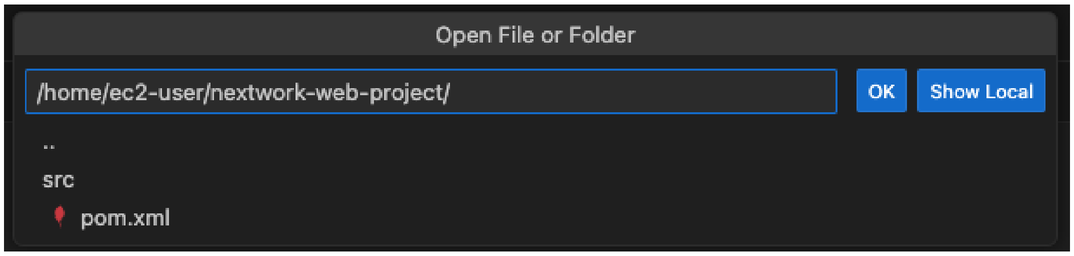
Configuration details required to set up a remote connection include the host (i.e. the EC2 instance address), the identity file (i.e. the location of our private key) and our user (i.e. the user that we are logging into for our instance).

Once we are connected to our EC2 instance, we should see our EC2 public IPv4 DNS connection in the status bar of the VS Code window:

SSH: ec2-13-40-50-32.eu-west-2.compute.a...

Exploring our web app’s files:

From the VS Code window, open the web app folder:

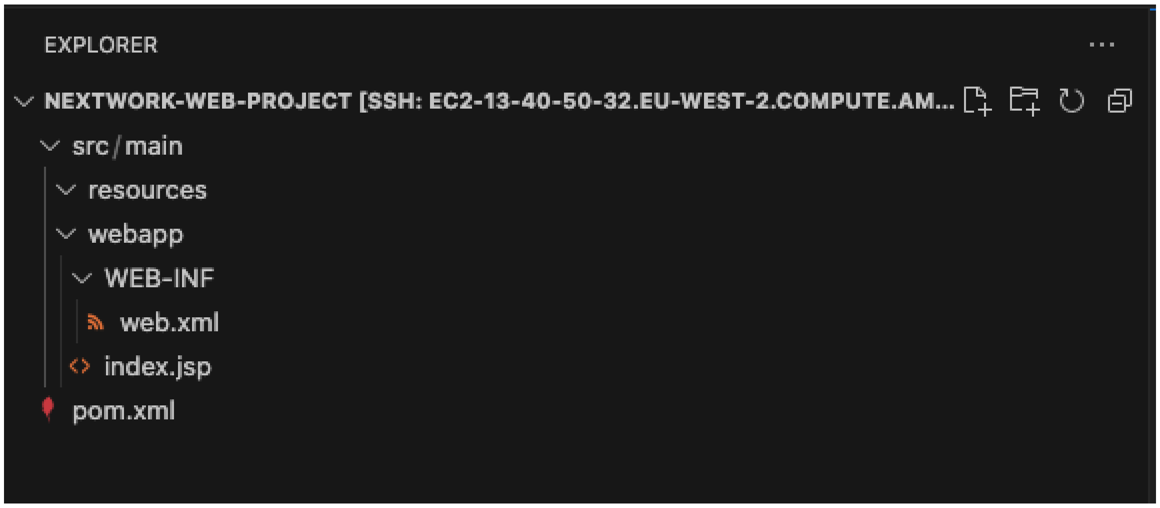


```
/home/ec2-user/nextwork-web-project/
..
src
pom.xml
```

Now click the explorer tab in the left side menu of VS Code. This will show all the files and subfolders within our web app folder. Two of the project folders created by Maven are src and webapp, which are defined below:

src: source folder that holds all the source code files that define how our web app will look and works.

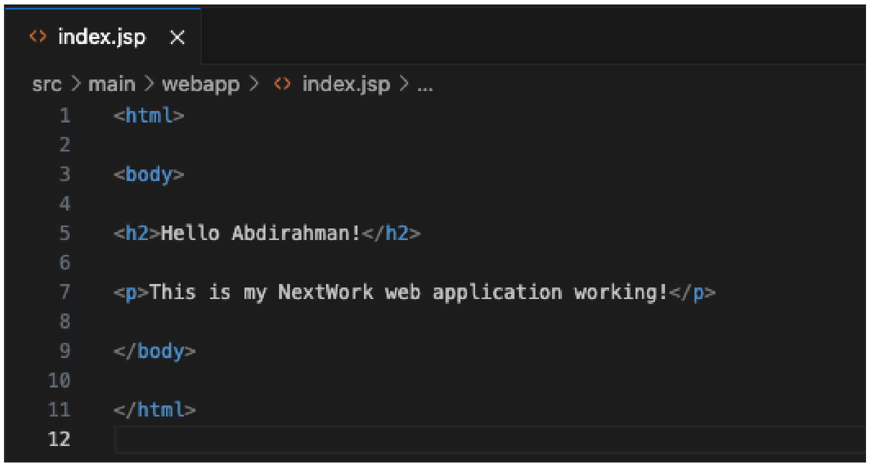
webapp: this is where our webapp files are stored (e.g. .xml and .jsp files) which are the configuration files a web app might need.



```
NEXTWORK-WEB-PROJECT [SSH: EC2-13-40-50-32.EU-WEST-2.COMPUTE.AM...
  src / main
    resources
    webapp
      WEB-INF
        web.xml
      index.jsp
  pom.xml
```

The index.jsp file is the file in our web app that defines both HTML content (i.e. the static elements that go into our web app’s page), as well as any code for generating dynamic content.

We edited the index.jsp file by editing the HTML code to also sat ‘Hello Abdirahman’ and we added a new paragraph (<p>) with a line stating ‘This is my NextWork web application working!’.



index.jsp

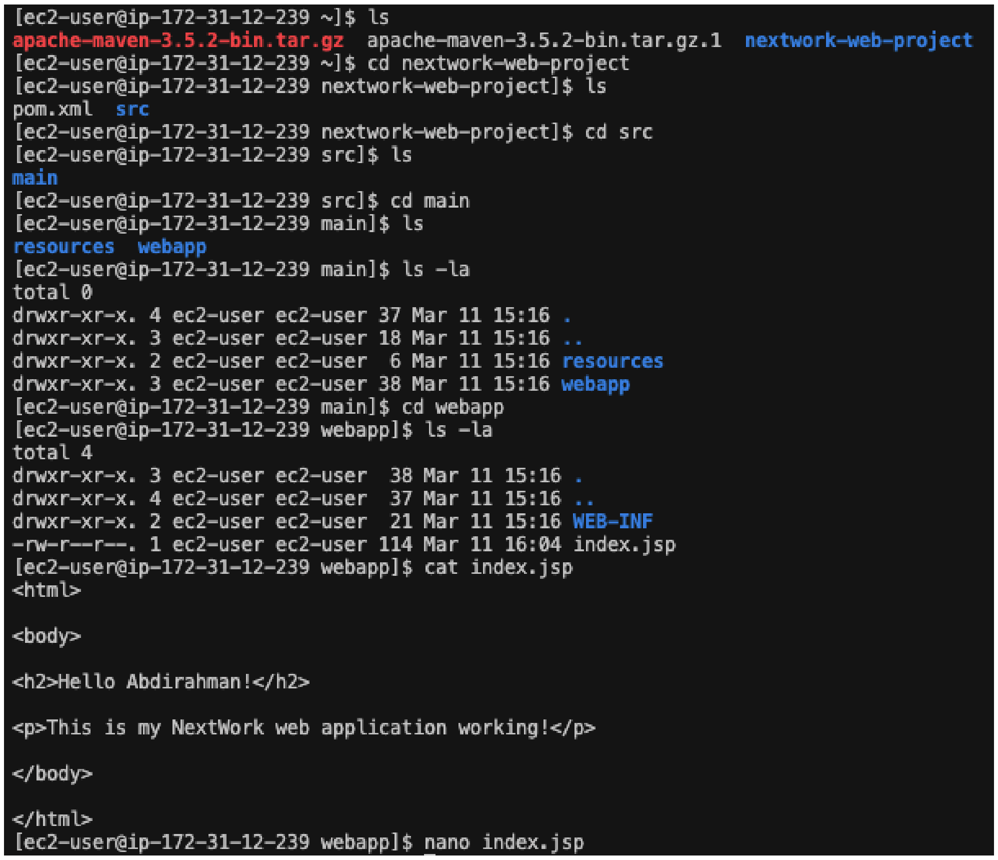
src > main > webapp > index.jsp > ...

```
<html>

<body>

<h2>Hello Abdirahman!</h2>

<p>This is my NextWork web application working!</p>

</body>

</html>

```

We have now set up our web app in AWS using the IDE.

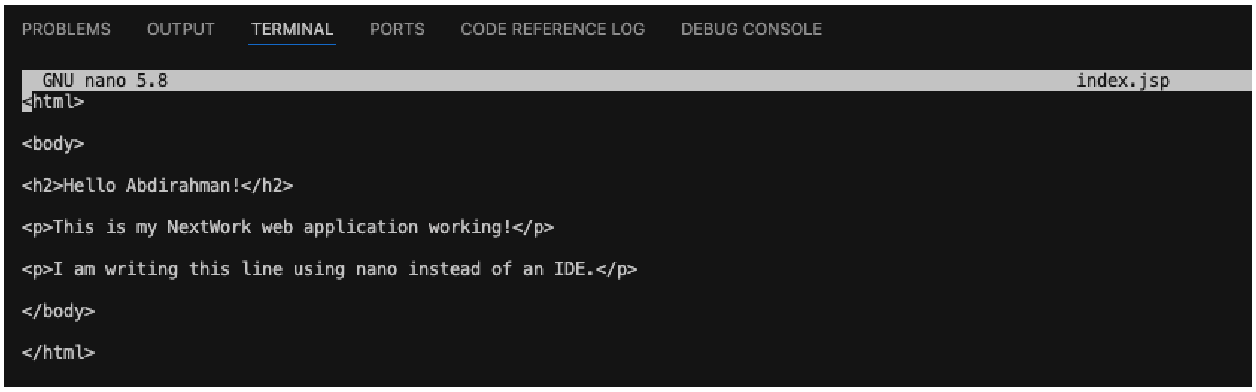
<u>Alternate method to edit the index.jsp using the terminal:</u>

We can also edit files using the terminal rather than the IDE. To do this we can run the ‘cd’ commands to navigate our terminal to the right folder and then the ‘nano’ command to edit our index.jsp file (shown below).



```
[ec2-user@ip-172-31-12-239 ~]$ ls
apache-maven-3.5.2-bin.tar.gz  apache-maven-3.5.2-bin.tar.gz.1  nextwork-web-project
[ec2-user@ip-172-31-12-239 ~]$ cd nextwork-web-project
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ ls
pom.xml  src
[ec2-user@ip-172-31-12-239 nextwork-web-project]$ cd src
[ec2-user@ip-172-31-12-239 src]$ ls
main
[ec2-user@ip-172-31-12-239 src]$ cd main
[ec2-user@ip-172-31-12-239 main]$ ls
resources  webapp
[ec2-user@ip-172-31-12-239 main]$ ls -la
total 0
drwxr-xr-x. 4 ec2-user ec2-user 37 Mar 11 15:16 .
drwxr-xr-x. 3 ec2-user ec2-user 18 Mar 11 15:16 ..
drwxr-xr-x. 2 ec2-user ec2-user  6 Mar 11 15:16 resources
drwxr-xr-x. 3 ec2-user ec2-user 38 Mar 11 15:16 webapp
[ec2-user@ip-172-31-12-239 main]$ cd webapp
[ec2-user@ip-172-31-12-239 webapp]$ ls -la
total 4
drwxr-xr-x. 3 ec2-user ec2-user  38 Mar 11 15:16 .
drwxr-xr-x. 4 ec2-user ec2-user  37 Mar 11 15:16 ..
drwxr-xr-x. 2 ec2-user ec2-user  21 Mar 11 15:16 WEB-INF
-rw-r--r--. 1 ec2-user ec2-user 114 Mar 11 16:04 index.jsp
[ec2-user@ip-172-31-12-239 webapp]$ cat index.jsp
<html>

<body>

<h2>Hello Abdirahman!</h2>

<p>This is my NextWork web application working!</p>

</body>

</html>
[ec2-user@ip-172-31-12-239 webapp]$ nano index.jsp
```

Below is the result of using the ‘nano’ command. We can now edit the index.jsp file and include another line ‘I am writing this line using nano instead of an IDE.’.



```
  GNU nano 5.8                                                                                   index.jsp
<html>

<body>

<h2>Hello Abdirahman!</h2>

<p>This is my NextWork web application working!</p>

<p>I am writing this line using nano instead of an IDE.</p>

</body>

</html>
```

Compared to an IDE, editing the index.jsp file using the terminal challenged my linux skills more and was less intuitive. I would more likely use the IDE for simplicity especially for larger pieces of code that require more editing or if there is lots of navigating of files.

To verify our editing work in the terminal, I opened up the VS Code window that had the SSH connection to our EC2 instance. It was possible to see our updates right away as the window shared the same SSH connection as the terminal.

To summarise:

Services I used were AWS EC2, VS Code, Terminal. Key concepts I learnt include:

- using IAM roles instead of Root users to securely complete daily tasks (best practice)
- Launching an EC2 instance, setting up a key pair for secure access and setting up network settings to allow SSH traffic
- Setting up a terminal within VS Code and using the terminal to change permissions in a .pem file
- Connecting to an EC2 instance
- Installing Apache Maven and Amazon Corretto 8 - also verifying correct installation
- Creating an application using Maven within the terminal
- Connecting VS Code with an EC2 instance (using remote - ssh extension) and editing the index.jsp file using the IDE
- Alternatively using the terminal to edit a file using the ‘nano’ command

Next project: Connect a GitHub Repo with AWS