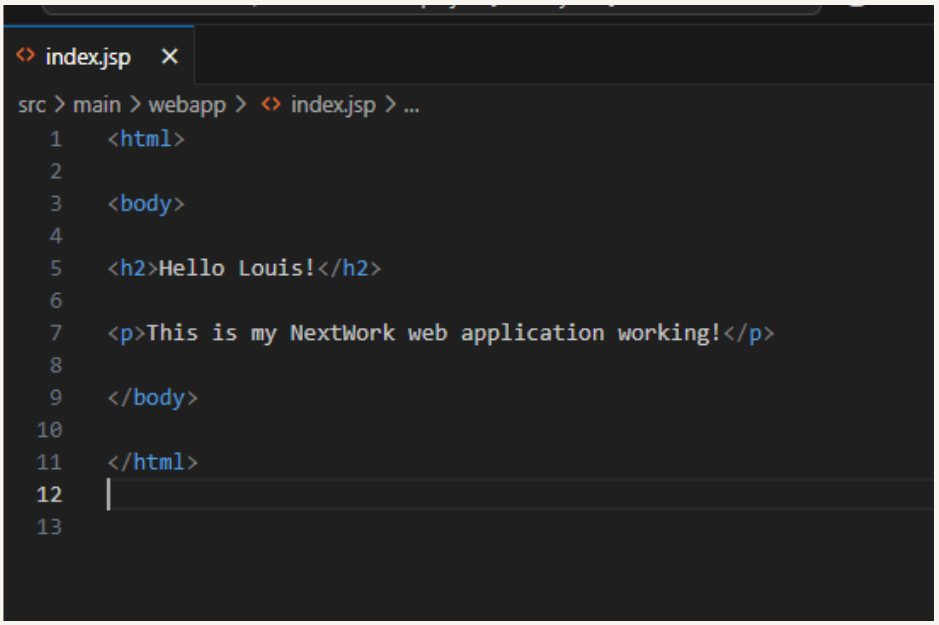
nextwork.org

Set Up a Web App Using AWS and VS Code

L Louis Moyo



Louis Moyo
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to launch a cloud server (EC2), connect to it securely, install dev tools, and generate a simple web application. I'm doing this to build the base infrastructure for CI/CD pipelines.

Key tools and concepts

Services I used were EC2, SSH, Maven, Java (JSP/Servlet), and VS Code Remote SSH. Key concepts I learnt include managing key pairs, working with .ssh/config, generating a Maven Java web app, and editing JSP files both in VS Code and directly in the terminal.

Project reflection

One thing I didn't expect in this project was how tricky SSH host key verification and permissions could be when reconnecting after IP changes.

This project took me approximately 4 hours. The most challenging part was fixing SSH/known_hosts issues, and it was most rewarding to finally see my Java web app structure and edit index.jsp successfully.

L

Louis Moyo
NextWork Student

This project is part one of a series of 7 DevOps projects where I'll build a complete CI/CD pipeline. I'll be working on the next project today.

L
Louis Moyo
NextWork Student

Launching an EC2 instance

I started this project by launching an EC2 instance because I need a dedicated virtual server in the cloud to act as the home for my web app. This instance gives me computing power, storage, and network access that I can securely connect to from anywhere. By using EC2 instead of my local machine, I'm learning how real-world DevOps teams build applications in the cloud and ensuring that all future steps in the CI/CD pipeline can be tested and deployed in an environment that mirrors production.

I also enabled SSH

SSH (Secure Shell) is a protocol that allows me to securely connect to and control a remote server over the internet. I enabled SSH so that I can safely log in to my EC2 instance from my local computer, run commands, install software, and manage my web app without exposing sensitive credentials or data.

Key pairs

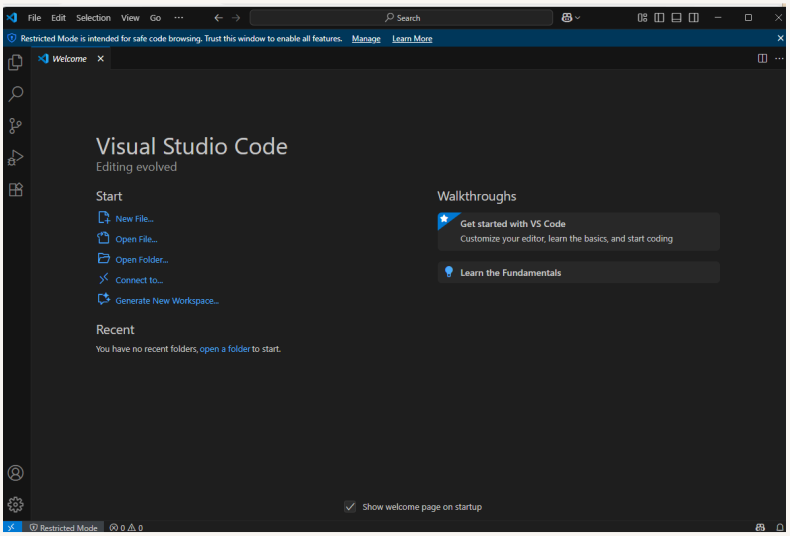
A key pair is a set of security credentials used to connect securely to my EC2 instance. It consists of a public key (stored by AWS) and a private key (stored on my computer). When I connect, AWS checks that my private key matches the public key, which proves my identity and allows me to log in without using a password.

Once I set up my key pair, AWS automatically downloaded a .pem file to my computer. This file contains my private key, which I'll use to securely connect to my EC2 instance through SSH or VS Code. Without this file, I wouldn't be able to access my server.

Set up VS Code

VS Code (Visual Studio Code) is a lightweight, powerful code editor created by Microsoft. It supports many programming languages, has an integrated terminal, and allows me to connect to remote servers like my EC2 instance.

I installed VS Code to write, edit, and manage my web app's code, as well as to securely connect to my EC2 instance through the terminal. This lets me develop directly in the cloud while using the convenience of an IDE on my local machine.

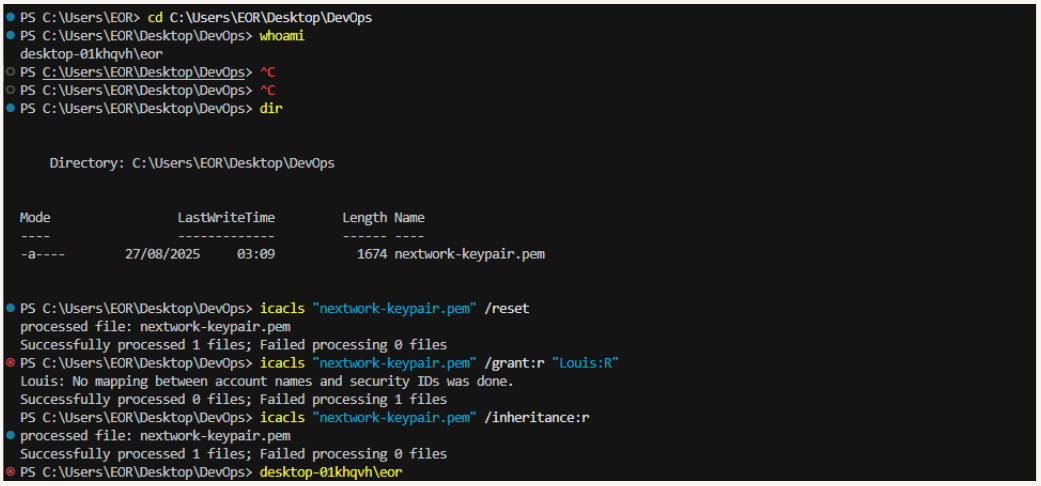


Louis Moyo
NextWork Student

My first terminal commands

A terminal is a text-based interface that lets me send instructions directly to my computer's operating system instead of using menus or clicks. The first command I ran for this project was cd C:\Users\EOR\Desktop\DevOps, which navigated my terminal into the DevOps folder where my AWS key file is stored.

I updated my private key's permissions by running the icacls command in the Windows terminal while inside my DevOps folder. I used icacls "nextwork-keypair.pem" /reset to remove the default permissions, icacls "nextwork-keypair.pem" /grant:r "%USERNAME%:R" to give myself read-only access, and icacls "nextwork-keypair.pem" /inheritance:r to prevent other folders or files from changing its permissions. This made my .pem file secure and only accessible by me, just like running chmod 400 on Linux or macOS.



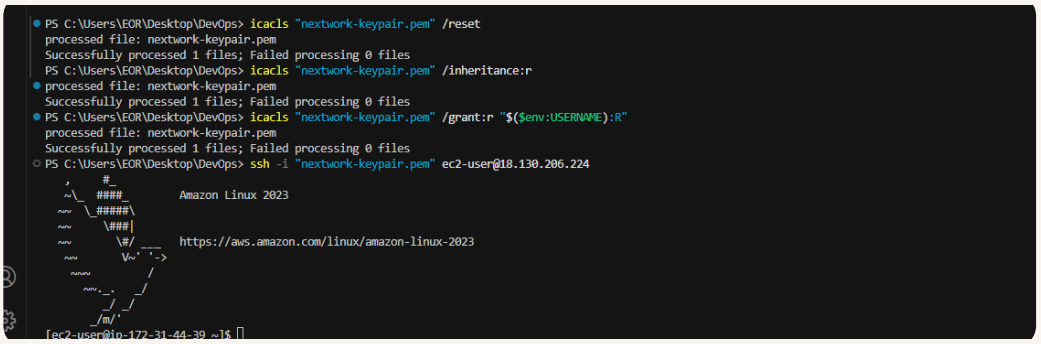
Louis Moyo
NextWork Student

SSH connection to EC2 instance

To connect to my EC2 instance, I ran the command: ssh -i "nextwork-keypair.pem" ec2-user@18.130.206.224 This command told my computer to use my private key (nextwork-keypair.pem) to securely log into the server at the public IPv4 DNS 18.130.206.224 with the default Amazon Linux username ec2-user.

This command required an IPv4 address

A server's IPv4 DNS is the public internet address that maps the server's IP to a human-readable name, allowing other computers to locate and connect to it. In AWS, this DNS is what I use from my local machine to securely connect to my EC2 instance over the internet.



Maven & Java

Apache Maven is a build automation and dependency management tool for Java projects. It helps developers compile, package, and manage external libraries automatically, and it can generate project structures using templates called archetypes.

Maven is required in this project because it sets up the structure for our Java web app and automatically manages the external code dependencies we need. This saves time and lets us focus on building features instead of manually configuring libraries and project files.

Java is a widely used programming language and platform that allows developers to build applications ranging from web apps to mobile and enterprise systems. It's portable, meaning the same Java program can run on many different devices without modification.

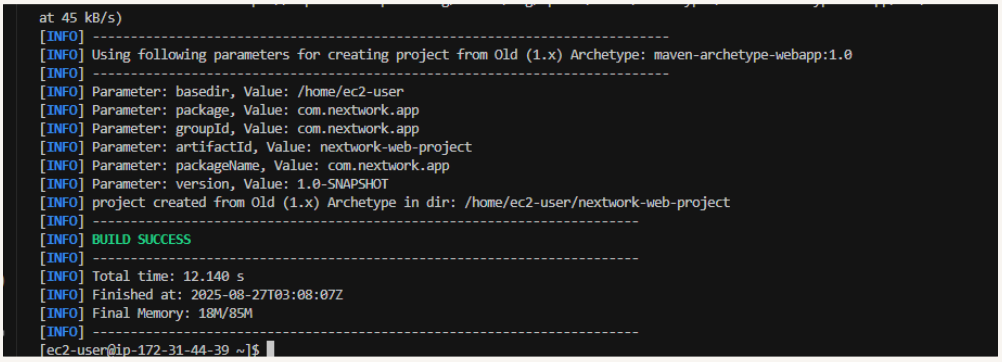
Java is required in this project because Apache Maven needs Java to run, and our web app itself will be written in Java. By installing Amazon Corretto 8 (a free, secure version of Java provided by AWS), we ensure that our EC2 instance can compile, build, and execute the web app we're going to create.

Create the Application

I generated a Java web app using the command: mvn archetype:generate -DgroupId=com.mycompany.app \ -DartifactId=my-webapp \ -DarchetypeArtifactId=maven-archetype-webapp \ -DinteractiveMode=false This created a starter project with the standard Java web application folder structure and files.

I installed Remote - SSH, which is a VS Code extension that lets me securely connect to a remote server over SSH. I installed it to connect VS Code directly to my EC2 instance so I can easily navigate folders, open and edit files like index.jsp, and use all the IDE features instead of only working in the terminal.

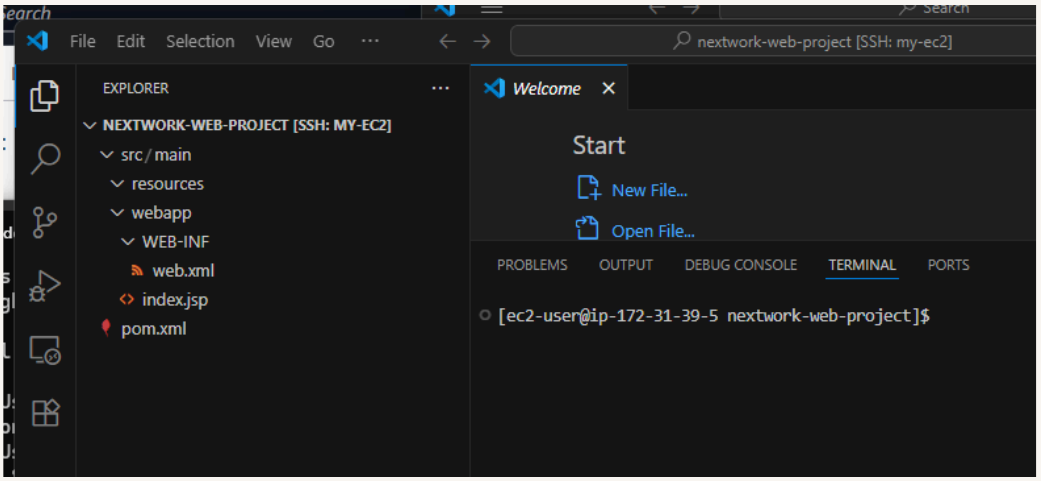
Configuration details required to set up a remote connection include the Host (my EC2 instance's public IPv4 DNS), the User (ec2-user), and the IdentityFile (the path to my nextwork-keypair.pem). These settings tell VS Code how to authenticate and connect securely to my EC2 instance.


at 45 kB/s)
[INFO] ----------
[INFO] Using following parameters for creating project from Old (1.x) Archetype: maven-archetype-webapp:1.0
[INFO] ----------
[INFO] Parameter: basedir, Value: /home/ec2-user
[INFO] Parameter: package, Value: com.nextwork.app
[INFO] Parameter: groupId, Value: com.nextwork.app
[INFO] Parameter: artifactId, Value: nextwork-web-project
[INFO] Parameter: packageName, Value: com.nextwork.app
[INFO] Parameter: version, Value: 1.0-SNAPSHOT
[INFO] project created from Old (1.x) Archetype in dir: /home/ec2-user/nextwork-web-project
[INFO] ----------
[INFO] BUILD SUCCESS
[INFO] ----------
[INFO] Total time: 12.140 s
[INFO] Finished at: 2025-08-27T03:08:07Z
[INFO] Final Memory: 18M/85M
[INFO] ----------
[ec2-user@ip-172-31-44-39 ~]$

Create the Application

Using VS Code's file explorer, I could see the project structure with pom.xml at the root, a src folder containing main and test code, and inside src/main/webapp I saw the web resources like index.jsp, WEB-INF, and configuration files such as web.xml. The explorer clearly showed how Maven organized the source code, resources, and web application files for easy navigation and editing.

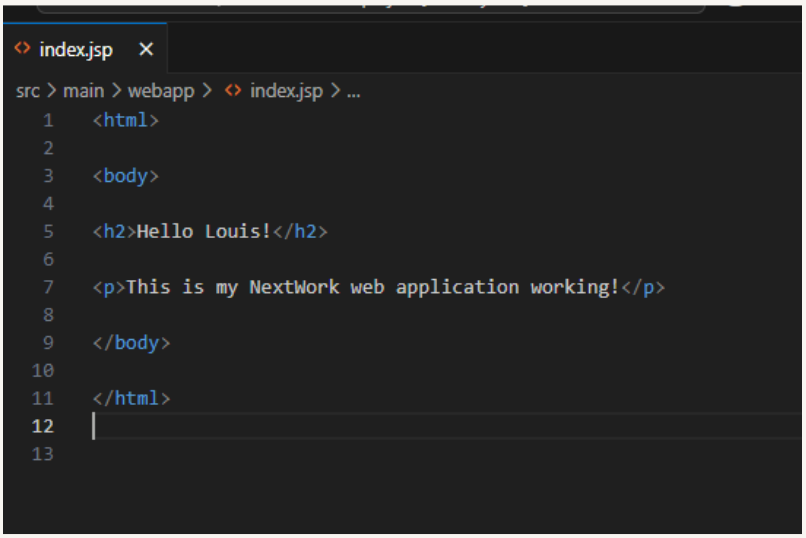
Two important project folders created by Maven are src and webapp. The src folder contains the source code and resources, including main for application code and test for unit tests. The webapp folder, inside src/main, stores web-related files like index.jsp, static resources, and WEB-INF where configuration files such as web.xml are placed. These folders help organise code and web content in a standard structure.



Using Remote - SSH

The index.jsp is the default Java Server Page of the web application. It acts as the entry point when users visit the app in a browser. It can contain HTML, CSS, and Java code to dynamically generate content for the client.

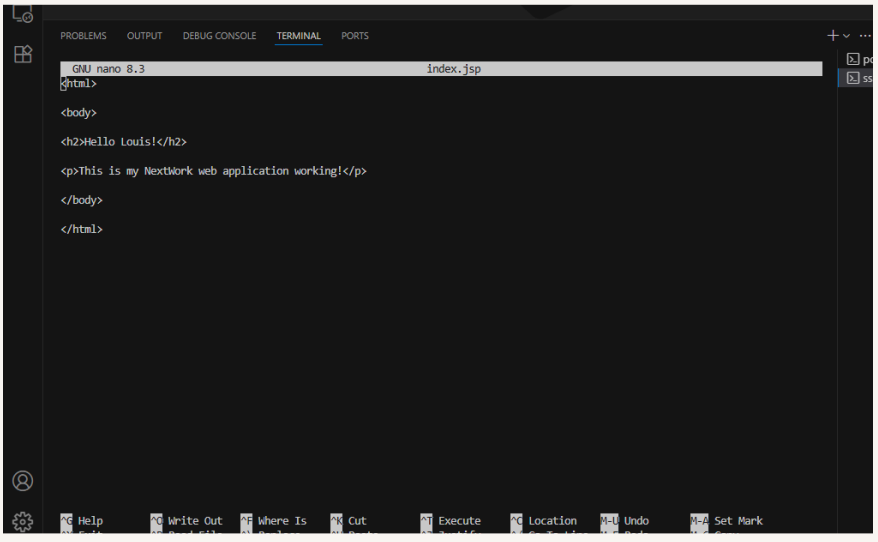
I edited index.jsp by opening it in VS Code, modifying the HTML/JSP content inside, and saving the file. This allowed me to customise what is displayed when the app loads.



Using nano

An alternative to using IDEs is editing directly in the terminal. To edit index.jsp, I navigated into the src/main/webapp folder and ran: nano index.jsp This opened the file inside the Nano editor, where I could view and change the JSP code using only the keyboard. After editing, I saved with CTRL + O and exited with CTRL + X. This method proves I can manage code even without an IDE.

Compared to using an IDE, editing index.jsp in the terminal felt slower and less intuitive because I had no syntax highlighting, auto-completion, or file navigation support. It was useful for learning the basics, but for larger projects I'd be more likely to use an IDE like VS Code since it makes editing, debugging, and managing files much faster and easier.



The place to learn & showcase your skills

Check out nextwork.org for more projects