**Setting up a Web Application in the Cloud**

This is a seven-project series. In this, I will create a CI/CD pipeline to build and deploy a web application using AWS Code services.

Here’s what I’ll build at the end of all seven projects.

A diagram of a software application

AI-generated content may be incorrect.

Firstly, **what is a CI/CD pipeline?**

A CI/ CD (Continuous Integration and Continuous Deployment/ Delivery) pipeline is a key tool to automate steps from development to deployment, which help software get built and released even faster.

In this project, I’ll build an application using AWS and another tool called VS (Visual Studio) code.

**Tasks:**

1. Launch an EC2 instance.
2. Use VSCode to set up a remote SSH connection to connect to the EC2 instance.
3. Install Maven and Java and generate a basic web app.

So, in this project, I’ll build the following:

A diagram of a software application

AI-generated content may be incorrect.

Let’s dive in:

Step 1: Set up an IAM user

**What is an IAM user? Why are we setting one up?**

AWS Identity and Access Management (IAM) is a web service that helps you securely control access to AWS resources. With IAM, you can manage permissions that control which AWS resources users can access. You use IAM to control who is authenticated (signed in) and authorized (has permissions) to use resources. IAM provides the infrastructure necessary to control authentication and authorization for your AWS accounts.

In AWS, a **user** is a person or a computer that can do things on the AWS cloud.

When you create an AWS account for the first time, the login you get is called the **root** user of the AWS account. AWS actually recommends to **not** use your root user for everyday tasks to protect it from security breaches.

You should create IAM users instead. If a root user is a master key to your AWS account, think of IAM users as key copies. IAM users have separate usernames and passwords to your root user, and you can set them to have limited access to your account's resources.

* [Head to your AWS Account](https://signin.aws.amazon.com/signin?redirect_uri=https%3A%2F%2Fconsole.aws.amazon.com%2Fconsole%2Fhome%3FhashArgs%3D%2523%26isauthcode%3Dtrue%26state%3DhashArgsFromTB_ap-southeast-2_fffdf5be4bb1a27e&client_id=arn%3Aaws%3Asignin%3A%3A%3Aconsole%2Fcanvas&forceMobileApp=0&code_challenge=m-aiqeB2UZeXTGXNyugMP8L64zd_AGUxJl4HLnA-X1o&code_challenge_method=SHA-256) as the root user.
* Open the **AWS IAM** console.
* From the left hand navigation panel, choose **Users.**
* Choose **Create user.**
* For the User name, use Yourname-IAM-Admin‍
* Make sure to select the checkbox next to **Provide user access to the AWS Management Console - optional.‍**
* This does not apply to all accounts, but if you're prompted with a pop up panel that says **Are you providing access to a person?**, choose **I want to create an IAM user.**

A screenshot of a computer

AI-generated content may be incorrect.

* For the console password, choose **Custom password.**
* Type in a password that you will be able to remember/access in the future.
* Deselect the checkbox for **Users must create a new password at next sign-in - Recommended.**
* Choose Next.
* In the permissions set up page, choose Attach policies directly.
* From the list of Permissions policies, select AdministratorAccess.

A screenshot of a computer

AI-generated content may be incorrect.

* Choose Next.
* Choose Create user.

A screenshot of a computer

AI-generated content may be incorrect.

A screenshot of a computer

AI-generated content may be incorrect.

* Choose **Download .csv file.**
* Copy the **Console sign-in URL.**
* Now you're ready to start using your IAM user. 🏁
* Log out of your root user's AWS Account.
* Paste and go to your copied console sign-in URL.
* Open your downloaded .csv file containing your user's access instructions.
* Click on the Console sign-in url.
* Log in using your IAM user's username and password in the .csv file.

Step 2: Launch an EC2 Instance

We’ll use a virtual server (EC2 instance) for developing the web application.

A screenshot of a computer

AI-generated content may be incorrect.

In this step, we’ll launch an EC2 instance, set up a key-pair for a secure access and the set up network connection for the EC2 instance.

* Head to **Amazon EC2** in your AWS Management Console.

**What is EC2?**  
A legendary AWS service! Amazon EC2 is a service that lets you rent and use virtual computers in the cloud. They're like your personal computers, but they exist on the internet instead of being physically in front of you. You can create, customize, and use these computers for all different reasons, from running applications to hosting websites.

EC2 = **Elastic Compute Cloud.**

Here's what the three words mean:  
**Elastic** = flexible. This service can easily adapt and change in size and power to fit your needs.  
**Compute** = computing power. EC2 provides virtual computers that can do various tasks, just like your personal computer.  
**Cloud** = available over the internet.

* Switch your **Region** to the one closest to you.
* In your EC2 console, select Instances from the left hand navigation panel.
* Choose Launch instances.

Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Picking an EC2 instance is like customising a virtual computer that fits what you need for your project. You can customize your EC2 instance's CPU, memory, storage, and networking capacity and more!

* Now, lets set up an EC2 instance.
* Enter the name you want to.
* Choose Amazon Linux 2023 AMI under Amazon Machine Image (AMI).

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance.

A screenshot of a computer

AI-generated content may be incorrect.

* Leave **t2.micro** under **Instance type**.
* Under **Key pair (login)**, choose **Create a new key pair.**

**What is a key pair?**  
A key pair in EC2 is like the keys to your virtual computer. Just like you need a key to unlock and start your car, a key pair lets you securely access your EC2 instance.

It's made of two halves: a public key that AWS keeps, and a private key that you download.

When you use the private key, it verifies that you're the one allowed to access that specific virtual machine, keeping everything secure and just for you.

* Use nextwork-keypair as your key pair's name.
* Keep the **Key pair type** as **RSA**, and the **Private key file format** as **.pem**

A screenshot of a computer

AI-generated content may be incorrect.

**What is a key pair type? Why do we pick RSA?**  
The key pair type determines the algorithm used for generating the key pair's cryptographic keys.

We use RSA (Rivest-Shamir-Adleman), which is one of the most common cryptographic algorithms used due to its strength and security. RSA is widely supported and trusted for creating digital signatures and encrypting data.  
**What is a Private key file format? Why do we pick .pem?**  
Just like how documents can be saved in different formats (e.g. PDF, DOCX, or TXT) for different apps or systems, private keys also come in different file formats. Not every system or application can process all these formats, so choosing the right one is crucial.

The **.pem** format stands for **Privacy Enhanced Mail**. It started off as a way to secure emails and has become the go-to format for cryptographic keys because it works with many different types of servers e.g. EC2 instances!

* Select **Create key pair.**
* A new file will automatically download to your local computer - nice! This is your private key.
* Before we lose track of our .pem file, let's organise it in our computer.
* Head to your local computer's desktop.
* Create a new folder in your desktop called **DevOps** and move the pem file to this folder.
* Back to our EC2 instance setup, head to the **Network settings** section.
* For **Allow SSH traffic from**, select the dropdown and choose **My IP**. This makes sure only you can access your EC2 instance.
* Double check that the IP address under **My IP** is correct - you can check your IP by clicking [here](https://checkip.amazonaws.com).
* If your IP address is different from what's under **My IP**, select **Custom** from the dropdown instead. Enter your IP and make sure to add a /32 to the end e.g. 012.345.678.9/32

**What is SSH?**  
SSH i.e. **Secure Shell** is a protocol used to make sure only authorized users can access a remote server. When you connect to your EC2 instance later in this project, SSH verifies you have the correct private key that matches the public key on the server.

SSH is also a type of network traffic. Once SSH has authorized you, it'll set up a secure connection between you and the EC2 instance. All data transferred (including your commands and the responses from the instance) gets encrypted. This encryption makes SSH an ideal method for working with virtual servers!

A screenshot of a computer

AI-generated content may be incorrect.

* When you're ready, choose **Launch instance**.

A screenshot of a computer

AI-generated content may be incorrect.

Step 3: Install VSCode

Now that the EC2 instance is up and running, we’ll set up the web app. But how do we do that?

We’ll use VSCode to connect to the EC2 instance, so that we can create and edit the web app.

So, in this step we will download VSCode in the computer. Set up a terminal in VSCode, to communicate with the EC2 instance. Update the key pair's permission settings, so that we can use it to log into the EC2 instance later.

A screenshot of a computer

AI-generated content may be incorrect.

* Open terminal/command prompt and choose the Devops folder where we have copied the pem file.
* Once you’re in the DevOps folder, you might want to check if your .pem file is there.

In the terminal, copy

icacls "nextwork-keypair.pem" /reset

icacls "nextwork-keypair.pem" /grant:r "USERNAME:R"

icacls "nextwork-keypair.pem" /inheritance:r

* Make sure to replace "USERNAME" with your Windows username.

**What is icacls?**  
Icacls (which stands for Integrity Control Access Control Lists) is a tool for Windows that lets you decide who can open or change the files on your system. In these icacls commands, you're using:

* /reset to remove default permission settings on the file
* /grant:r "USERNAME:R" to give the current user (that's you!) read access to your secret key
* /inheritance:r to make sure changes in the permissions of other files and the DevOps folder won't change the permission settings for this file.

A screenshot of a computer screen

AI-generated content may be incorrect.

Step 4: Connect to your EC2 instance.

Once we're connected, we can work inside your EC2 instance to set up that web app.

A screenshot of a computer

AI-generated content may be incorrect.

* Head back to your AWS Management Console.
* Click on **Instances** from the left hand navigation panel.
* Click on the checkbox next to your EC2 instance to view its details.
* Under the **Details** tab, look for **Public IPv4 DNS**.
* We'll need this DNS in a second - so keep this tab open!

**What is a Public IPv4 DNS?**  
A Public IPv4 DNS (which stands for Domain Name System) is the public address for your EC2 server that the internet uses to find and connect to it. The local computer you're using to do this project will find and connect to your EC2 instance through this IPv4 DNS.

* Now we'll connect to our instance via SSH.
* Head back to VSCode and open your terminal again.
* Use the following command to connect to your EC2 instance: ssh -i [PATH TO YOUR .PEM FILE] ec2-user@[YOUR PUBLIC IPV4 DNS]
  + Replace **[PATH TO YOUR .PEM FILE]** with the actual path to your private key file (e.g., ~/Desktop/DevOps/nextwork-keypair.pem). Delete the square brackets!
  + Replace **[YOUR PUBLIC IPV4 DNS]** with the Public DNS you just found. Delete the square brackets!

**What does this command do?**

* **ssh** starts a secure shell connection to your EC2 instance.
* **-i** specifies the identity file (your .pem file) you're using to authenticate a connection.
* **ec2-user@[YOUR PUBLIC IPV4 DNS]** specifies the username (ec2-user) and the address of the EC2 instance (Public DNS) to connect to.
* Your terminal will ask if you want to continue connecting to this EC2 instance. This is SSH's way of asking if you trust this server.
* Enter yes to continue connecting.
* Congrats! You've connected your EC2 instance via SSH.

A computer screen with white text

AI-generated content may be incorrect.

Step 5: Install Apache Maven and Amazon Corretto 8

Our goal today is to set up a web app inside this instance, so let's install two tools that are going to help us build Java web apps.

**In this step, you're going to:**

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

**What is Apache Maven?**  
Apache Maven is a tool that helps developers build and organize Java software projects. It's also a package manager, which means it automatically download any external pieces of code your project depends on to work.

We're also using Maven today because it's really useful for kick-starting web projects! It uses something called **archetypes**, which are like templates, to lay out the foundations for different types of projects e.g. web apps.

* Install Apache Maven using the commands below. You can copy and paste **all** of these lines into the terminal together, no need to run them line by line.

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

or

sudo yum install maven -y

* Once you've pasted these commands, don't forget to press **Enter** on your keyboard.

**What do these commands do?**  
These commands first download a setup file that tells your computer where to find Apache Maven. Then, they install Maven automatically, all in the space of a few lines.

* Now we're going to install Java 8, or more specifically, [Amazon Correto 8](https://docs.aws.amazon.com/corretto/latest/corretto-8-ug/what-is-corretto-8.html).
* Run these commands:

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

**What is Java? What is Amazon Correto 8?**  
Java is a popular programming language used to build different types of applications, from mobile apps to large enterprise systems.

Maven, which we just downloaded, is a tool that NEEDS Java to operate. So if we don't install Java, we won't be able to use Maven to generate/build our web app today.

Amazon Corretto 8 is a version of Java that we're using for this project. It's free, reliable and provided by Amazon.

**What do these commands do?**  
The first command installs Java Amazon Corretto 8. Then, the second command tells your EC2 instance how it can find Java, because it'll need to know Java's location to run Java commands! The last command is a time saver - you're saving Java's location so you can run Java commands from anywhere in your EC2 instance without needing to type out the full location each time.

**Woah! What's all this text popping up in the terminal?**  
The text you see after these commands is the terminal keeping you updated about it's progress with installing Java. It shows the specific packages it's going to install, downloading status, and even verifying that everything was installed.

* To verify that Maven is installed correctly, run the following command next:

mvn -v

* To verify that you've installed Java 8 correctly, run this next:

java -version

Step 6: Create an application

* Now that we have installed java and maven into our EC2 instance. Lets start generating our project.
* In this, we are going to run maven commands into the terminal to generate a java web app.
* Use **mvn** to generate a Java web app. To do this, use these commands:

mvn archetype:generate \

-DgroupId=com.nextwork.app \

-DartifactId=nextwork-web-project \

-DarchetypeArtifactId=maven-archetype-webapp \

-DinteractiveMode=false

If you run into an error saying directory present, then you run the following command and then rerun the above command.

rm -rf nextwork-web-project

A screen shot of a computer

AI-generated content may be incorrect.

**What is mvn?**   
When you run **mvn** commands, you're asking Maven to perform tasks (like creating a new project or building an existing one).

The **mvn archetype:generate** command specifically tells Maven to create a new project from a template (which Maven calls an archetype). This command sets up a basic structure for your project, so you don't have to start from scratch.

**Extra for Experts:** Some of the details you've specified in this command are...

-DartifactId=nextwork-web-project names your project

-DarchetypeArtifactId=maven-archetype-webapp specifies that you're creating a web application.

-DinteractiveMode=false runs the command without pausing for user input, so Maven will go ahead and install everything without waiting for your confirmation.

* Watch out for a **BUILD SUCCESS** message in your terminal once your application is all set up.

Step 7: Connect VSCode with your EC2 Instance

In this step, you'll connect VSCode to your EC2 instance so you can see and edit the web app you've just created.

* Install an extension in VSCode.
* Use the extension to set up a connection between VSCode and your EC2 instance.
* Explore and edit your Java web app's files using VSCode.

A screenshot of a computer

AI-generated content may be incorrect.

**Wait...aren't I already connected to my EC2 instance with SSH?**  
Yes, connecting with SSH in VSCode's **terminal** lets you send text commands to your EC2 instance, but you don't get all the benefits of having an IDE like VSCode.

When you connect VSCode itself to your EC2 instance (not just your terminal), you unlock VSCode’s IDE features (like file navigation and code editing) directly on your EC2 instance.

* Clicking on the **Extensions** icon at the side of your VSCode window.

A screenshot of a computer

AI-generated content may be incorrect.

* In the search bar, type Remote - SSH and click **Install** for the extension.

**Why are we installing Remote - SSH?**  
The **Remote - SSH extension** in VSCode lets you connect directly via SSH to another computer securely over the internet. This lets you use VSCode to work on files or run programs on that server as if you were doing it on your own computer, which will come in handy when we edit the web app in your EC2 instance!

A screenshot of a computer

AI-generated content may be incorrect.

* Click on the double arrow icon at the bottom left corner of your VSCode window. This button is a shortcut to use Remote - SSH.

A screen shot of a computer

AI-generated content may be incorrect.

* Select **Remote-SSH: Connect to Host...**
* Select **+ Add New SSH Host...**

A screenshot of a computer

AI-generated content may be incorrect.

**What is an SSH Host?**  
**An SSH Host** is the computer or server you're connecting to using SSH. It's the target location where you want to run commands or manage files; in our case, the SSH Host is the EC2 instance we created.

* Enter the SSH command you used to connect to your EC2 instance: ssh -i [PATH TO YOUR .PEM FILE] ec2-user@[YOUR PUBLIC IPV4 DNS]
  + - Replace **[PATH TO YOUR .PEM FILE]** with the actual path to your private key file (e.g., ~/Desktop/DevOps/nextwork-keypair.pem). Delete the square brackets!
    - Replace **[YOUR PUBLIC IPV4 DNS]** with the Public DNS you just found. Delete the square brackets!
* Select the configuration file at the top of your window. It should look similar to /Users/username/.ssh/config

A screenshot of a computer

AI-generated content may be incorrect.

* A **Host added!** popup will confirm that you've set up your SSH Host - yay!
* Select the blue **Open Config** button on that popup.
* Confirm that all the details in your configuration file look correct:
  + **Host** should match up with your EC2 instance's IPv4 DNS.
  + **IdentityFile** should match up to nextwork-keypair.pem's location in your local computer.
  + **User** should say ec2-user

A black rectangular object with a black border

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

* Now you’re ready to connect VSCode with your EC2 instance!
* Click on the double arrow button on the bottom left corner and select **Connect to Host** again.
* You should now see your EC2 instance listed at the top.

A screen shot of a computer

AI-generated content may be incorrect.

* Select the EC2 instance and off we gooooooooooo to a new VSCode window ✈️
* Check the bottom right hand corner of your new VSCode window - it should show your EC2 instance's IPV4 DNS.

Now let's open up your web app's files.

* From VSCode's left hand navigation bar, select the **Explorer** icon.
* Select **Open folder**.
* At the top of your VSCode window, you should see a drop down of different file and folder names. Ooooo, this is VSCode asking you which specific file/folder you'd like to open!
* Enter /home/ec2-user/nextwork-web-project.
* Press **OK.**

A screenshot of a computer program

AI-generated content may be incorrect.

* VSCode might show you a popup asking if you trust the authors of the files in this folder. If you see this popup, select **Yes, I trust the authors.**
* Check your VSCode window's file explorer again - a folder called **nextwork-web-project** is here!
* Try expanding all the subfolders in the file explorer. All folders have a > icon next to their name.

A screenshot of a computer

AI-generated content may be incorrect.

Let's get to know some of these web app files/folders:

* The src (source) folder holds all the source code files that define how your web app looks and works.
* src is further divided into webapp, which are the web app's files e.g. HTML, CSS, JavaScript, and JSP files, and resources, which are the configuration files a web app might need e.g. connection settings to a database.
* pom.xml is a Maven **Project Object Model** file. It stores information and configuration details that Maven will use to build the project.
* Exploring done! So how can VSCode help you edit your application files? Let's find out.
* From your file explorer, click into **index.jsp**.

**What is index.jsp? What's the difference from HTML?**  
**index.jsp** is a file used in Java web apps. It's similar to an HTML file because it contains markup to display web pages.

However, index.jsp can also include Java code, which lets it generate dynamic content.

This means content can change depending on things like user input or data from a database. Social media apps are great examples of web apps because the content you see is always changing, updating and personalised to you. HTML files are static and can’t include Java code. That's why it's so important to install Java in your EC2 instance - so you can run the Java code in your web app!

* Welcome to editor view of index.jsp. Now we're really using VSCode's IDE abilities - editing code is much easier here than in the terminal.
* Let's try modifying **index.jsp** by changing the placeholder code to the code snippet below. Don't forget to replace **{YOUR NAME}** from the following code with your name:

A computer screen with white text

AI-generated content may be incorrect.

* Save the changes you've made to **index.jsp** by selecting **Command/Ctrl + S** on your keyboard.

So, finally we have created a web application in the cloud.