NextWork.org

Deploy an App with CodeDeploy

Prajit Venkatachalam
https://www.linkedin.com/in/prajit-venkatachalam-435b2a150/

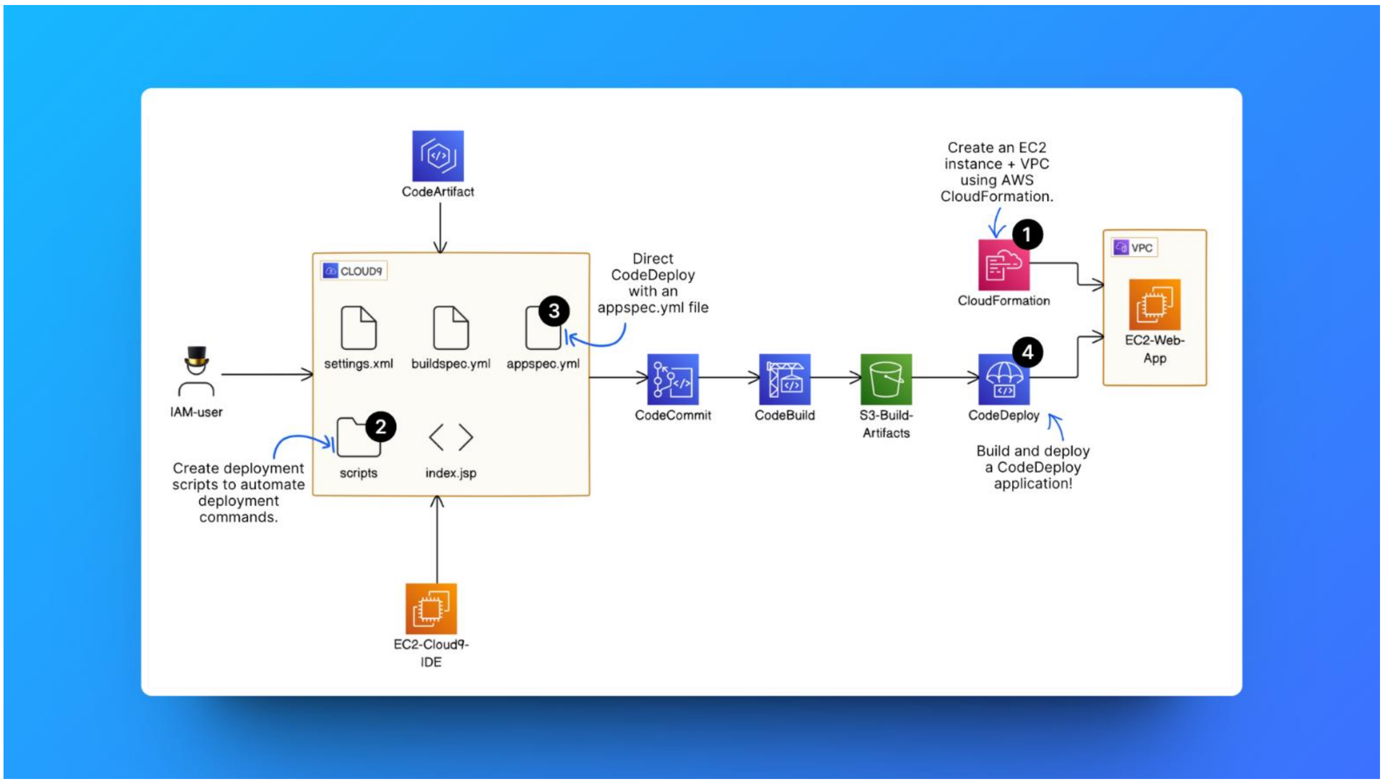

CodeArtifact
CLOUD9
settings.xml
buildspec.yml
appspec.yml
scripts
index.jsp
IAM-user
Create deployment scripts to automate deployment commands.
Direct CodeDeploy with an appspec.yml file
CodeCommit
CodeBuild
S3-Build-Artifacts
CodeDeploy
Build and deploy a CodeDeploy application!
Create an EC2 instance + VPC using AWS CloudFormation.
CloudFormation
VPC
EC2-Web-App
EC2-Cloud9-IDE
1
2
3
4

Figure 1: Project plan

Introducing today's project!

What is AWS CodeDeploy?

AWS CodeDeploy is a deployment service that automates software deployments to various computing environments, such as Amazon EC2 instances, on-premises servers, and Lambda functions. CodeDeploy is useful for automating code deployments, rolling back on failure, and supporting both in-place and blue/green deployment strategies.

How I'm using AWS CodeDeploy in this project

In this project, I used AWS CodeDeploy to automate the deployment of my Java web application on EC2 instances. After setting up the EC2 instance, VPC, and IAM roles, I configured CodeDeploy to manage the deployment process, ensuring smooth updates in both development and production environments.

I created scripts to install dependencies, start, and stop services, and used an **appspec.yml** file to guide the deployment steps. Finally, I integrated CodeDeploy with CodeCommit and S3 to deploy my application WAR file from the build process.

One thing I didn't expect...

One thing I didn't expect in this project was the complexity of the overall configuration. At first, it felt overwhelming, but as I delved deeper, I found it fascinating. The exploration process taught me a great deal of knowledge, and I'm truly grateful for the experience it provided.

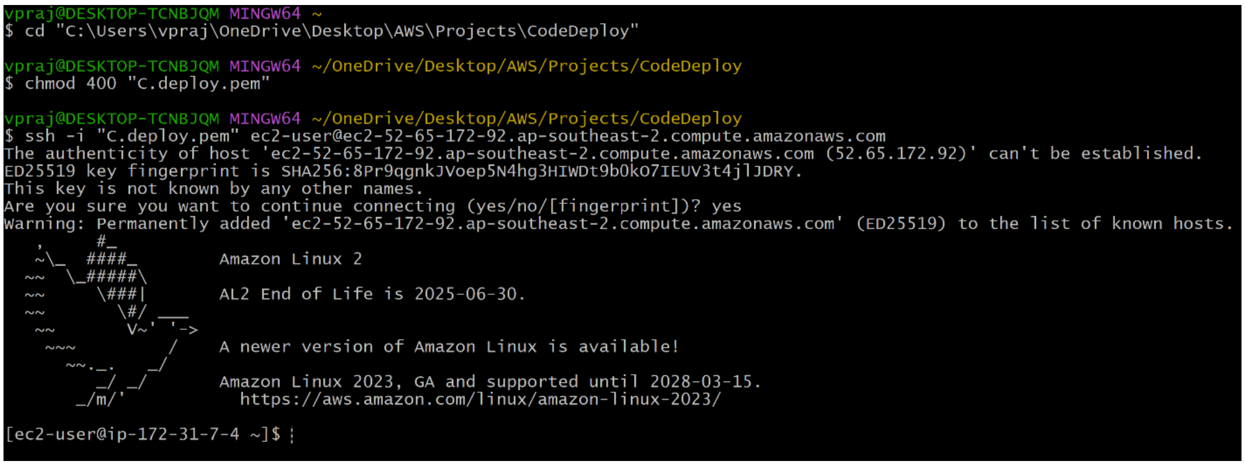
This project took me...

This project took me 4 hours to complete including pre setups and report writing.

Setups made before starting with AWS CodeBuild

Step 1: Setting up a Web app

1. I set up my web app by launching an EC2 instance in the AWS Console, and I connected to it via SSH using Git Bash



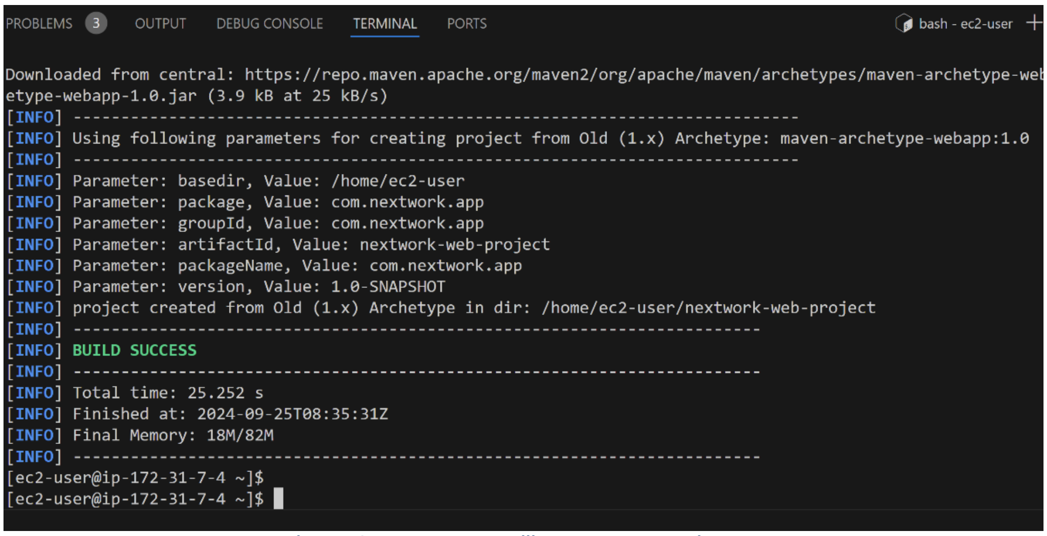
```
vpraj@DESKTOP-TCNBJQM MINGW64 ~
$ cd "C:\Users\vpraj\OneDrive\Desktop\AWS\Projects\CodeDeploy"

vpraj@DESKTOP-TCNBJQM MINGW64 ~/OneDrive/Desktop/AWS/Projects/CodeDeploy
$ chmod 400 "C.deploy.pem"

vpraj@DESKTOP-TCNBJQM MINGW64 ~/OneDrive/Desktop/AWS/Projects/CodeDeploy
$ ssh -i "C.deploy.pem" ec2-user@ec2-52-65-172-92.ap-southeast-2.compute.amazonaws.com
The authenticity of host 'ec2-52-65-172-92.ap-southeast-2.compute.amazonaws.com (52.65.172.92)' can't be established.
ED25519 key fingerprint is SHA256:8Pr9qgnkJVoep5N4hg3HIWDt9b0kO7IEUV3t4jlJDRY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'ec2-52-65-172-92.ap-southeast-2.compute.amazonaws.com' (ED25519) to the list of known hosts.
   ,     #_
   ~\_  ####_        Amazon Linux 2
  ~~  \_#####\
  ~~     \###|       AL2 End of Life is 2025-06-30.
  ~~       \#/ ___
   ~~       V~' '->
    ~~~         /    A newer version of Amazon Linux is available!
      ~~._.   _/
         _/ _/       Amazon Linux 2023, GA and supported until 2028-03-15.
       _/m/'           https://aws.amazon.com/linux/amazon-linux-2023/

[ec2-user@ip-172-31-7-4 ~]$ 
```

Figure 2: EC2 connected via SSH from Git Bash

2. After connecting to my EC2 instance, I set up the development environment by installing Apache Maven and Amazon Corretto 8 (Java version)

3. Next, I downloaded Visual Studio Code as an alternative to Cloud9. I connected to the EC2 instance via SSH and set up the Java web application by running the command:

```
mvn archetype:generate \
-DgroupId=com.nextwork.app \
-DartifactId=nextwork-web-project \
-DarchetypeArtifactId=maven-archetype-webapp \
-DinteractiveMode=false
```



```
Downloaded from central: https://repo.maven.apache.org/maven2/org/apache/maven/archetypes/maven-archetype-web
etype-webapp-1.0.jar (3.9 kB at 25 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: 25.252 s
[INFO] Finished at: 2024-09-25T08:35:31Z
[INFO] Final Memory: 18M/82M
[INFO] ------------------------------------------------------------------------
[ec2-user@ip-172-31-7-4 ~]$
[ec2-user@ip-172-31-7-4 ~]$
```

Figure 3: Maven compiling my Java web app

4. After running the above command, a folder named **'nextwork-web-project**' was created in my IDE's (VS Code) file explorer for my project.

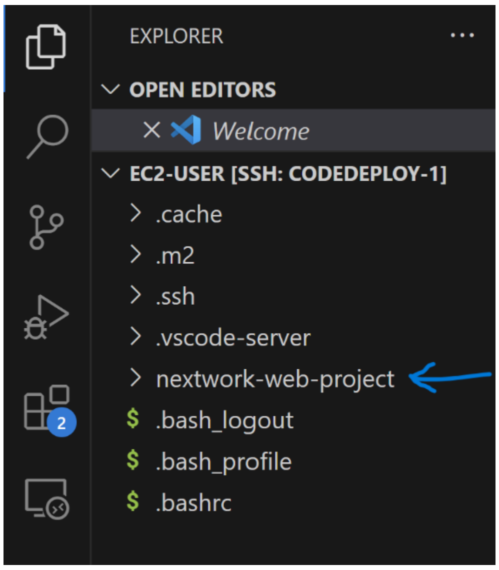
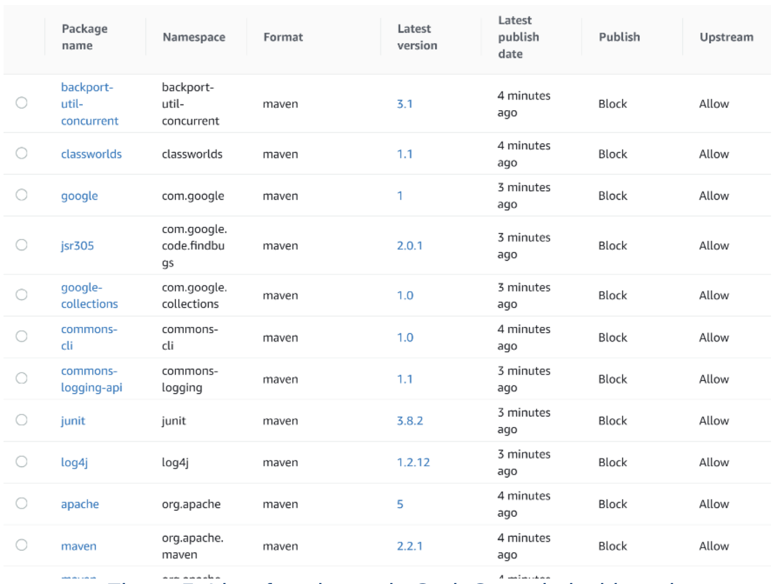


Figure 4: Project folder created in vscode file explorer

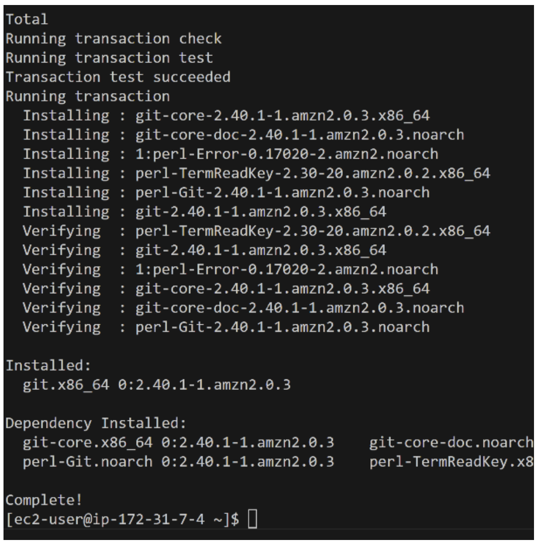
Step 2: Creating a Git repository with AWS CodeCommit



	Package name	Namespace	Format	Latest version	Latest publish date	Publish	Upstream
○	backport-util-concurrent	backport-util-concurrent	maven	3.1	4 minutes ago	Block	Allow
○	classworlds	classworlds	maven	1.1	4 minutes ago	Block	Allow
○	google	com.google	maven	1	3 minutes ago	Block	Allow
○	jsr305	com.google.code.findbugs	maven	2.0.1	3 minutes ago	Block	Allow
○	google-collections	com.google.collections	maven	1.0	3 minutes ago	Block	Allow
○	commons-cli	commons-cli	maven	1.0	4 minutes ago	Block	Allow
○	commons-logging-api	commons-logging	maven	1.1	3 minutes ago	Block	Allow
○	junit	junit	maven	3.8.2	3 minutes ago	Block	Allow
○	log4j	log4j	maven	1.2.12	3 minutes ago	Block	Allow
○	apache	org.apache	maven	5	4 minutes ago	Block	Allow
○	maven	org.apache.maven	maven	2.2.1	4 minutes ago	Block	Allow

Figure 5: List of packages in CodeCommit dashboard

1. I created a Git repository in AWS CodeCommit (main) and had to create a local repository in Visual Studio Code to make changes to the files and push them to the main repository. To do this, I installed Git in VSCode by running the command: **sudo yum install git**



```
Total
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : git-core-2.40.1-1.amzn2.0.3.x86_64
  Installing : git-core-doc-2.40.1-1.amzn2.0.3.noarch
  Installing : 1:perl-Error-0.17020-2.amzn2.noarch
  Installing : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64
  Installing : perl-Git-2.40.1-1.amzn2.0.3.noarch
  Installing : git-2.40.1-1.amzn2.0.3.x86_64
  Verifying  : perl-TermReadKey-2.30-20.amzn2.0.2.x86_64
  Verifying  : git-2.40.1-1.amzn2.0.3.x86_64
  Verifying  : 1:perl-Error-0.17020-2.amzn2.noarch
  Verifying  : git-core-2.40.1-1.amzn2.0.3.x86_64
  Verifying  : git-core-doc-2.40.1-1.amzn2.0.3.noarch
  Verifying  : perl-Git-2.40.1-1.amzn2.0.3.noarch

Installed:
  git.x86_64 0:2.40.1-1.amzn2.0.3

Dependency Installed:
  git-core.x86_64 0:2.40.1-1.amzn2.0.3    git-core-doc.noarch
  perl-Git.noarch 0:2.40.1-1.amzn2.0.3    perl-TermReadKey.x8

Complete!
[ec2-user@ip-172-31-7-4 ~]$ 
```

Figure 6: Installed Git in vscode

2. Next, I created a Git identity using the following commands

   git config --global user.name "Prajit"
   git config --global user.email "v.prajit98@gmail.com"

3. To create a local repository, I created a project directory in Visual Studio Code by cloning the CodeCommit repository (main). To clone the CodeCommit repository, I used the command:

   git clone https://git-codecommit.ap-southeast-2.amazonaws.com/v1/repos/nextwork-codedeploy

4. After running the above command, Git prompted for authentication to enter the username and password, as I was using Git over HTTPS. Therefore, I downloaded Git HTTPS credentials from IAM in the AWS Console.

5. Next, I navigated to the project directory folder to initialize my local repository. To move to the directory, I used the command:

   cd ~/nextwork-web-project, and I initialized the local repository by running the command**: git init -b main**

6. Now, I set the remote origin to the main repository by copying the CodeCommit URL from the "Clone URL" option in the repositories section of the AWS Console. I used it in the command:

   git remote add origin https://git-codecommit.ap-southeast-2.amazonaws.com/v1/repos/nextwork-codedeploy

7. Next, I committed and pushed my code by running the following commands:
   git add .
   git commit -m "Initial commit. Updated index.jsp."
   git push -u origin main

When I use the command git push -u origin main, Git prompts me for authentication again, requiring me to enter the username and password that I obtained from my HTTPS Git credentials.

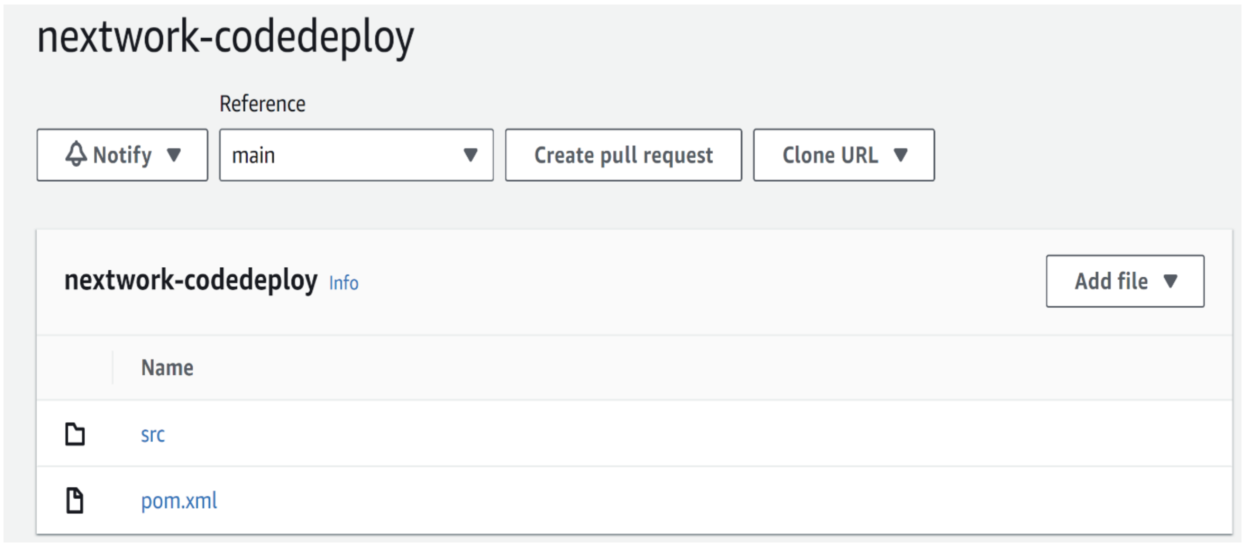
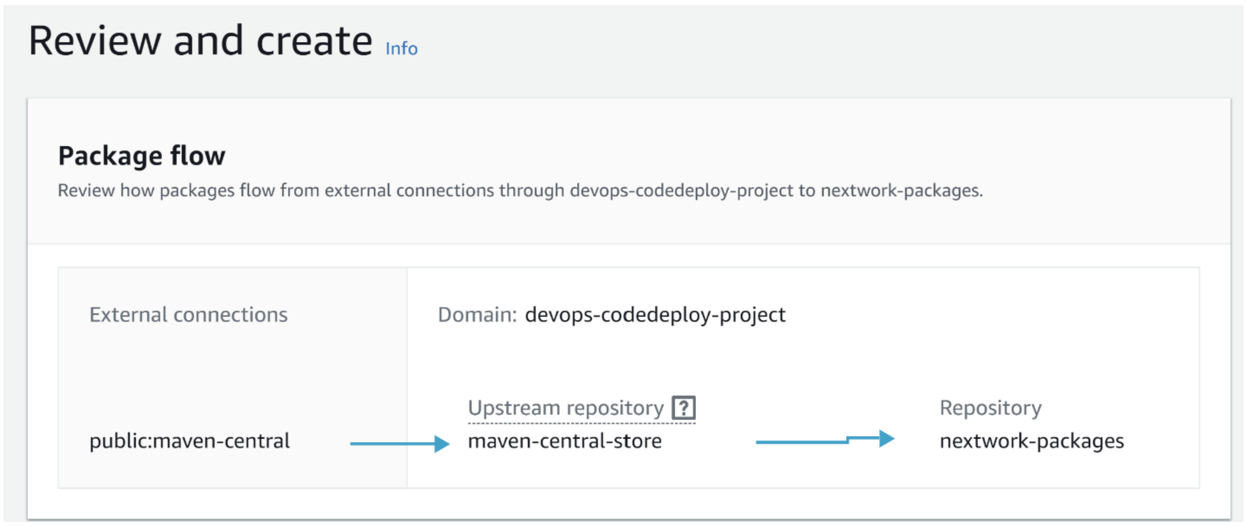


Figure 7: CodeCommit dashboard after the first git push

Step 3: Set up a AWS CodeArtifact


Review and create Info

Package flow

Review how packages flow from external connections through devops-codedeploy-project to nextwork-packages.

Figure 8:Package flow of CodeArtifact

Connecting my project with CodeArtifact

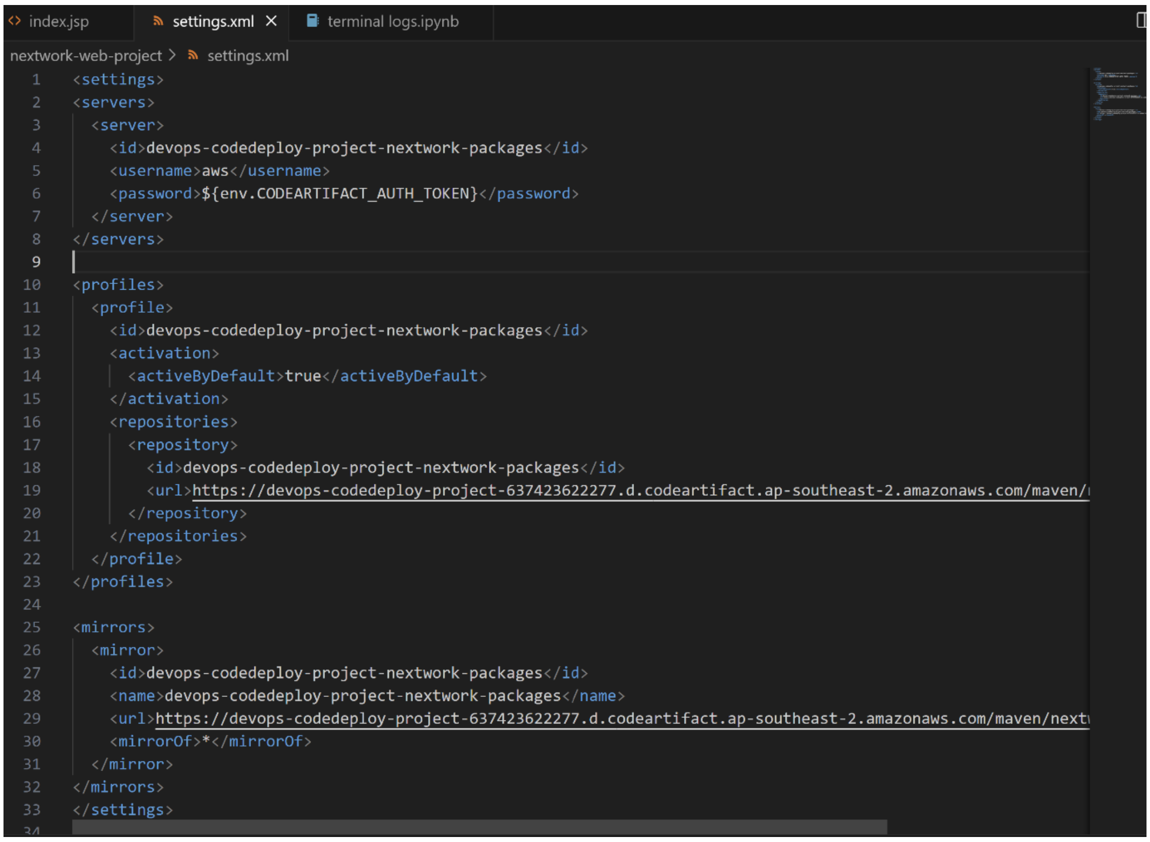
I connected my web app project (via my VS Code IDE) to CodeArtifact, so that CodeArtifact knows to which project it will store the packages or dependencies.
To connect my web app project to CodeArtifact via Visual Studio Code, I had to use a few commands. The first command is:

```
<servers>
<server>
<id>devops-3-nextwor-packages</id>
<username>aws</username>
<password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
</server>
</servers>
```

This command stores the access details for the repositories that my web app project is connecting to, so I needed to configure AWS in Visual Studio Code. I ran the command aws configure, which authenticates the AWS Access Key ID, Secret Access Key, default region name, and default output format. In this case, I had to create an AWS access key, which provides the Access Key ID and Secret Access Key.

I created a new file, **settings.xml**, in my web app. The settings.xml file provides Maven with instructions on where to find the dependencies it needs to fetch and how Maven will gain access to the repositories storing these dependencies.

Creating the settings.xml file involves writing three snippets of code in it, as shown in the picture below. To create the settings.xml file, I used the command:
cd ~/nextwork-web-project to navigate to that directory, and then ran **echo $'<settings>\n</settings>' > settings.xml** to create the file. Once the file created, it will start appearing in the "**nextwork-web-project**" folder in the vscode explorer.



index.jsp | settings.xml | terminal logs.ipynb

nextwork-web-project > settings.xml

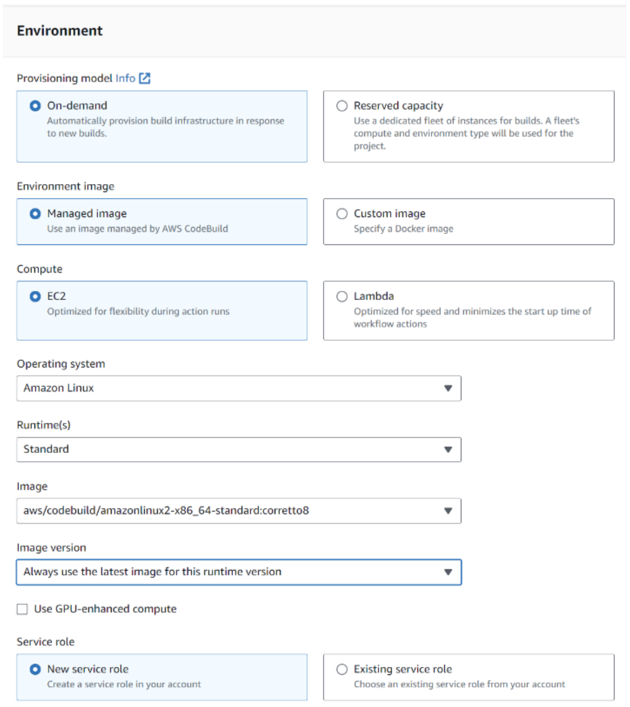
```xml
<settings>
<servers>
  <server>
    <id>devops-codedeploy-project-nextwork-packages</id>
    <username>aws</username>
    <password>${env.CODEARTIFACT_AUTH_TOKEN}</password>
  </server>
</servers>

<profiles>
  <profile>
    <id>devops-codedeploy-project-nextwork-packages</id>
    <activation>
      <activeByDefault>true</activeByDefault>
    </activation>
    <repositories>
      <repository>
        <id>devops-codedeploy-project-nextwork-packages</id>
        <url>https://devops-codedeploy-project-637423622277.d.codeartifact.ap-southeast-2.amazonaws.com/maven/
      </repository>
    </repositories>
  </profile>
</profiles>

<mirrors>
  <mirror>
    <id>devops-codedeploy-project-nextwork-packages</id>
    <name>devops-codedeploy-project-nextwork-packages</name>
    <url>https://devops-codedeploy-project-637423622277.d.codeartifact.ap-southeast-2.amazonaws.com/maven/next
    <mirrorOf>*</mirrorOf>
  </mirror>
</mirrors>
</settings>
```

Figure 9: Snippets of three code in Settings.xml file

Step 3: Set up a Codebuild

1. **Created an S3 Bucket**: Set up an S3 bucket to store the important build artifact, a WAR file, that would be generated during the CodeBuild process.

2. **Set up CodeBuild Project**: Configured a CodeBuild project to compile and package your code into a WAR file, with AWS CodeCommit as the source repository.



Environment

Provisioning model Info

[✓ On-demand] Automatically provision build infrastructure in response to new builds.
Reserved capacity: Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image

[✓ Managed image] Use an image managed by AWS CodeBuild
Custom image: Specify a Docker image

Compute

[✓ EC2] Optimized for flexibility during action runs
Lambda: Optimized for speed and minimizes the start up time of workflow actions

Operating system: Amazon Linux

Runtime(s): Standard

Image: aws/codebuild/amazonlinux2-x86_64-standard:corretto8

Image version: Always use the latest image for this runtime version

☐ Use GPU-enhanced compute

Service role

[✓ New service role] Create a service role in your account
Existing service role: Choose an existing service role from your account

Figure 10: Set up of CodeBuild environment

3. **Defined Build Environment**: Chose an EC2 instance running Amazon Linux 2 with Amazon Corretto 8 (Java runtime) for the CodeBuild environment.

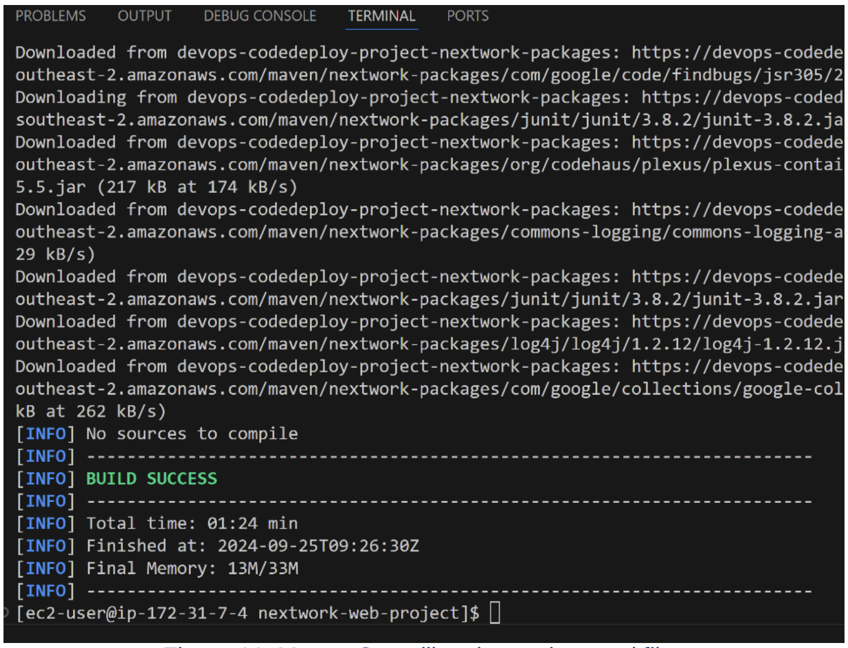
4. **Configured Build Artifacts**: Configured the S3 bucket to store the WAR file generated by CodeBuild during the build process.

5. **Enabled Logging**: Enabled AWS CloudWatch Logs to capture and store logs generated during the build process for monitoring and troubleshooting.

6. **Created buildspec.yml File**: Created a buildspec.yml file to automate the build process, specifying the install, pre-build, build, and post-build phases for CodeBuild to follow.

7. **Pre-Build Phase**: Added commands to initialize the environment and retrieve an authorization token from AWS CodeArtifact for secure Maven access.

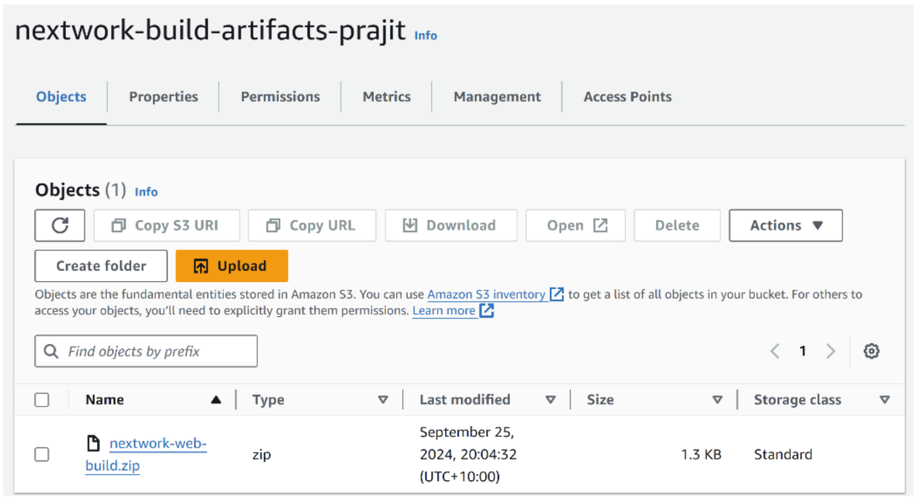
8. **Build Phase**: Added commands to compile the code using Maven and settings from settings.xml, printing the start of the build to the terminal.



```
Downloaded from devops-codedeploy-project-nextwork-packages: https://devops-codede
outheast-2.amazonaws.com/maven/nextwork-packages/com/google/code/findbugs/jsr305/2
Downloading from devops-codedeploy-project-nextwork-packages: https://devops-coded
southeast-2.amazonaws.com/maven/nextwork-packages/junit/junit/3.8.2/junit-3.8.2.ja
Downloaded from devops-codedeploy-project-nextwork-packages: https://devops-codede
outheast-2.amazonaws.com/maven/nextwork-packages/org/codehaus/plexus/plexus-contai
5.5.jar (217 kB at 174 kB/s)
Downloaded from devops-codedeploy-project-nextwork-packages: https://devops-codede
outheast-2.amazonaws.com/maven/nextwork-packages/commons-logging/commons-logging-a
29 kB/s)
Downloaded from devops-codedeploy-project-nextwork-packages: https://devops-codede
outheast-2.amazonaws.com/maven/nextwork-packages/junit/junit/3.8.2/junit-3.8.2.jar
Downloaded from devops-codedeploy-project-nextwork-packages: https://devops-codede
outheast-2.amazonaws.com/maven/nextwork-packages/log4j/log4j/1.2.12/log4j-1.2.12.j
Downloaded from devops-codedeploy-project-nextwork-packages: https://devops-codede
outheast-2.amazonaws.com/maven/nextwork-packages/com/google/collections/google-col
kB at 262 kB/s)
[INFO] No sources to compile
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time: 01:24 min
[INFO] Finished at: 2024-09-25T09:26:30Z
[INFO] Final Memory: 13M/33M
[INFO] ------------------------------------------------------------------------
[ec2-user@ip-172-31-7-4 nextwork-web-project]$
```

Figure 11: Maven Compiling the settings.xml file

9. **Post-Build Phase**: Configured Maven to package the compiled code into a WAR file and stored it in the S3 bucket, also printing the build completion time.



Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more

Name	Type	Last modified	Size	Storage class
nextwork-web-build.zip	zip	September 25, 2024, 20:04:32 (UTC+10:00)	1.3 KB	Standard

Figure 12: WAR file generated in S3

10. **Modified IAM Role**: Updated the IAM service role for CodeBuild by attaching the codeartifact-nextwork-consumer-policy, granting it access to necessary packages for the build.

Starting with AWS CodeDeploy

Set up an EC2 instance

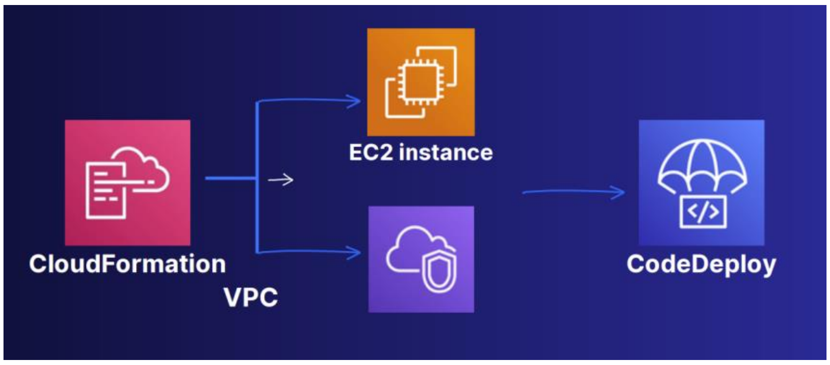
I set up an EC2 instance and VPC because this EC2 instance will host our Java application. It's like a computer in the cloud that will keep our app running and available for users to access.

I also set up the VPC (Virtual Private Cloud), which is another AWS service that helps us control who on the internet or other networks can access our web app. In this project, our VPC will ensure that our web app is discoverable on the internet.

We manage production and development environments separately to avoid the risk of testing code changes or new features in a live production environment, which users might otherwise see, and both EC2 instances serve different purposes. The EC2 instance in the development environment is used to create and edit code for our web app, while the EC2 instance in the production environment is for running my application in a live, production environment, which users will access.

I used AWS CloudFormation to set up my EC2 instance and VPC for deploying my application. CloudFormation is a service that allows you to create or update resources in your AWS account by using predefined templates written in code.

The below diagram represents the CloudFormation flow to create EC2 instance and VPC.



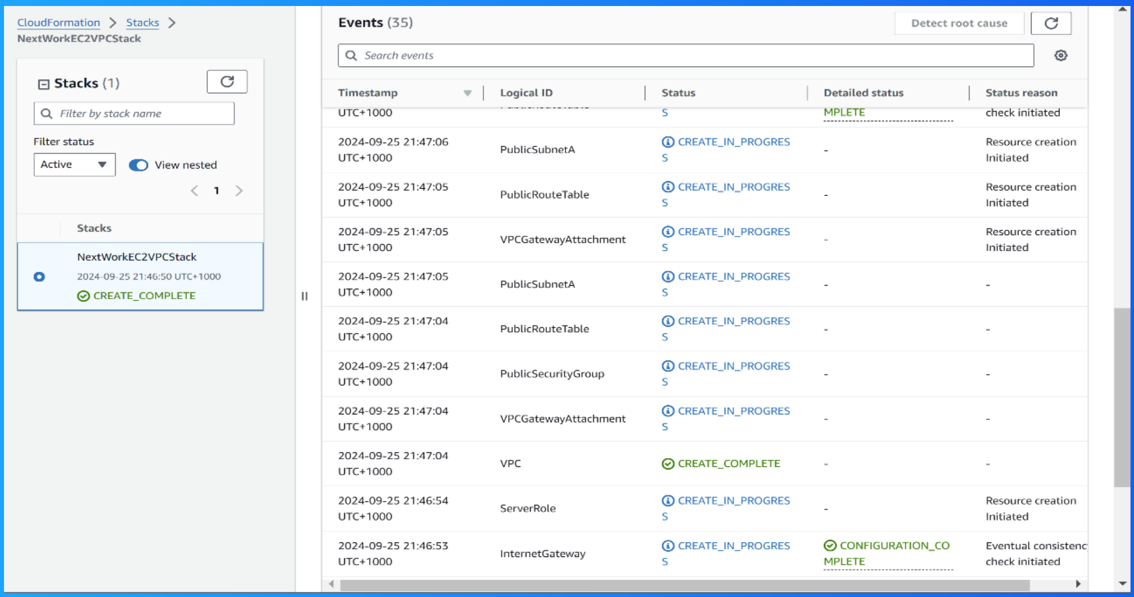
Prajit Venkatachalam
NextWork Student

How I set up CloudFormation

In the CloudFormation console, I created a stack with new resources (standard), and I uploaded the template file which is **nextworkwebapp.yml.**

Under **Parameters**, I entered my IP address in the **MyIP** field. I found my IP address by visiting http://checkip.amazonaws.com/ and added /32 to specify that only my specific IP address, meaning only my computer, can access the resources I'm configuring. This helps ensure security.

To finish the setup, I clicked **Create Stack**, and it took a few minutes to complete the stack creation.



NextWorkEC2VPCStack
2024-09-25 21:46:50 UTC+1000
CREATE_COMPLETE

Events (35)

Timestamp	Logical ID	Status	Detailed status	Status reason
UTC+1000		S	MPLETE	check initiated
2024-09-25 21:47:06 UTC+1000	PublicSubnetA	CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-09-25 21:47:05 UTC+1000	PublicRouteTable	CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-09-25 21:47:05 UTC+1000	VPCGatewayAttachment	CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-09-25 21:47:05 UTC+1000	PublicSubnetA	CREATE_IN_PROGRESS	-	-
2024-09-25 21:47:04 UTC+1000	PublicRouteTable	CREATE_IN_PROGRESS	-	-
2024-09-25 21:47:04 UTC+1000	PublicSecurityGroup	CREATE_IN_PROGRESS	-	-
2024-09-25 21:47:04 UTC+1000	VPCGatewayAttachment	CREATE_IN_PROGRESS	-	-
2024-09-25 21:47:04 UTC+1000	VPC	CREATE_COMPLETE	-	-
2024-09-25 21:46:54 UTC+1000	ServerRole	CREATE_IN_PROGRESS	-	Resource creation Initiated
2024-09-25 21:46:53 UTC+1000	InternetGateway	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Eventual consistenc check initiated

Create scripts to run the application

Scripts are a set of commands that instruct the terminal to run tasks or automate processes line by line.

In this project, I set up three different scripts in my **nextwork-web-project** folder, and CodeDeploy will execute these scripts to set up and deploy my web application on the target EC2 instance.

The first script I created was **install_dependencies.sh**

```
#!/bin/bash
sudo yum install tomcat -y
sudo yum -y install httpd
sudo cat << EOF > /etc/httpd/conf.d/tomcat_manager.conf
<VirtualHost *:80>
  ServerAdmin root@localhost
  ServerName app.nextwork.com
  DefaultType text/html
  ProxyRequests off
  ProxyPreserveHost On
  ProxyPass / http://localhost:8080/nextwork-web-project/
  ProxyPassReverse / http://localhost:8080/nextwork-web-project/
</VirtualHost>
EOF
```

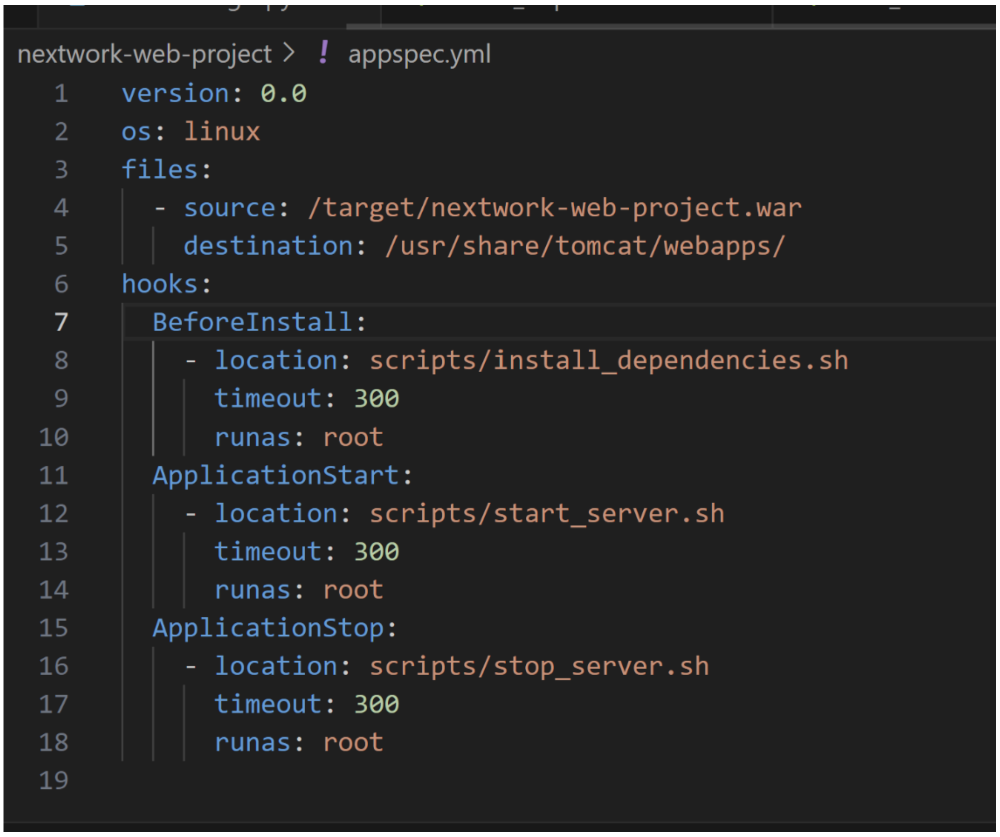
The second script that I used was **start_server.sh**

```
#!/bin/bash
sudo systemctl start tomcat.service
sudo systemctl enable tomcat.service
sudo systemctl start httpd.service
sudo systemctl enable httpd.service
```

The third script that I used was **stop_server.sh**

```
#!/bin/bash
isExistApp="$(pgrep httpd)"
if [[ -n $isExistApp ]]; then
sudo systemctl stop httpd.service
fi
isExistApp="$(pgrep tomcat)"
if [[ -n $isExistApp ]]; then
sudo systemctl stop tomcat.service
fi
```

Next, I created an **appspec.yml** file to instruct CodeDeploy on the steps and files required for a deployment. It specifies what CodeDeploy should do at various stages of the deployment process, such as which scripts to run and where to place the files.


nextwork-web-project > appspec.yml

```yaml
version: 0.0
os: linux
files:
  - source: /target/nextwork-web-project.war
    destination: /usr/share/tomcat/webapps/
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

Figure 13: appspec.yml

I added the below lines to my buildspec.yml file to include the newly added scripts folder and appspec.yml file in the WAR file that CodeBuild will create during the build process. By incorporating these new items, it is ensuring that CodeDeploy has all the necessary files to successfully deploy our application.

- appspec.yml
- scripts//***

In buildspec.yml script, the above two lines should be inserted at the particular space where it is bolded.

```
artifacts:
  files:
    - target/nextwork-web-project.war
    - appspec.yml
    - scripts/**/*
  discard-paths: no
```

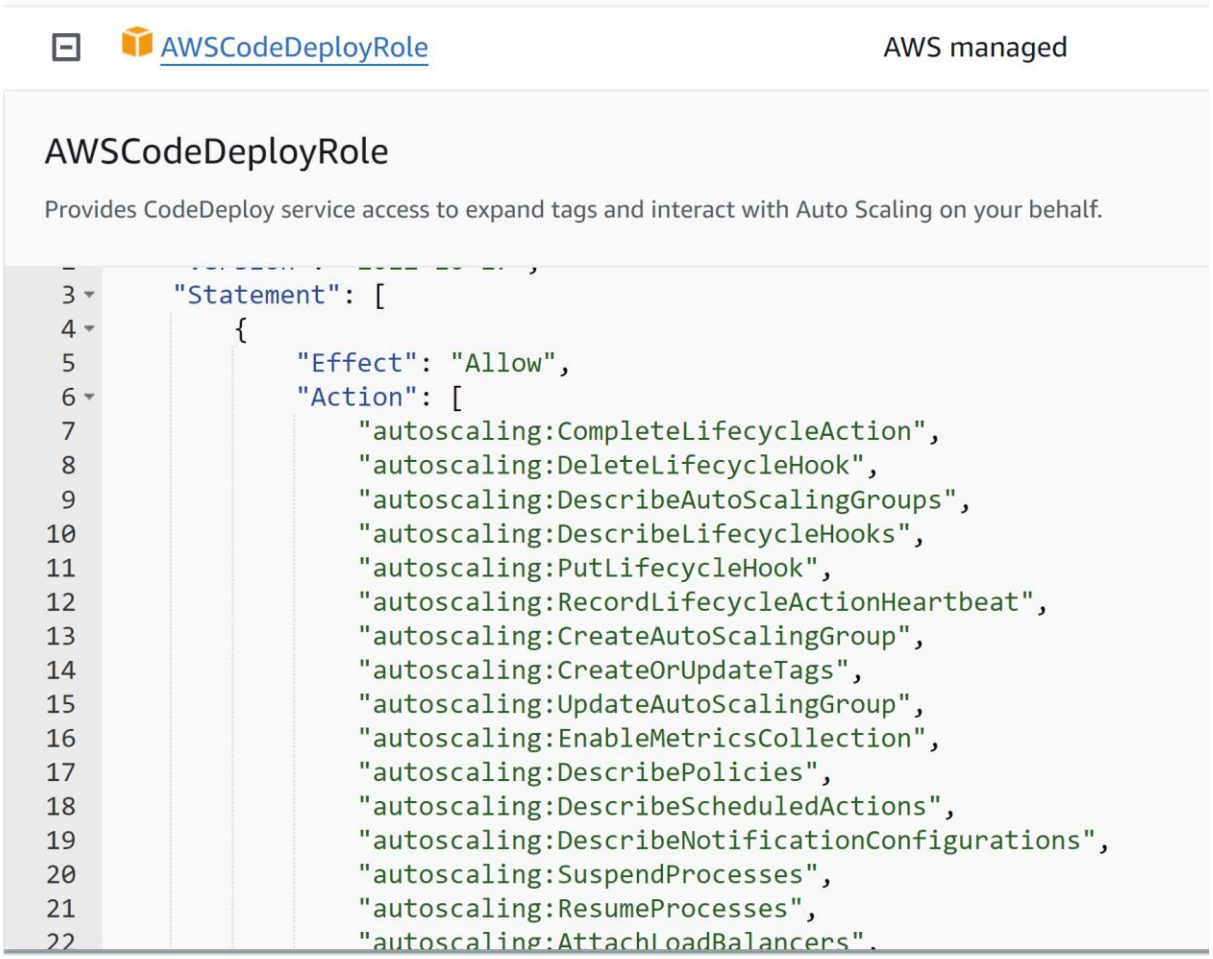
Now I saved all the files and committed all the changes to push in the CodeCommit by running the commands:

```
cd ~/nextwork-web-project
git add .
git commit -m "Adding CodeDeploy files"
git push -u origin main
```

CodeDeploy’s IAM Role

I created an IAM service role for CodeDeploy because it doesn’t have access to the EC2 instance for deployments. To grant that access, I established the IAM service role for CodeDeploy.

To set up CodeDeploy’s IAM role, I attached an AWS managed policy called **AWSCodeDeployRole**, which automatically adds the default permissions that CodeDeploy typically needs.



AWSCodeDeployRole

Provides CodeDeploy service access to expand tags and interact with Auto Scaling on your behalf.

```
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CompleteLifecycleAction",
                "autoscaling:DeleteLifecycleHook",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeLifecycleHooks",
                "autoscaling:PutLifecycleHook",
                "autoscaling:RecordLifecycleActionHeartbeat",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:EnableMetricsCollection",
                "autoscaling:DescribePolicies",
                "autoscaling:DescribeScheduledActions",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:SuspendProcesses",
                "autoscaling:ResumeProcesses",
                "autoscaling:AttachLoadBalancers",
```

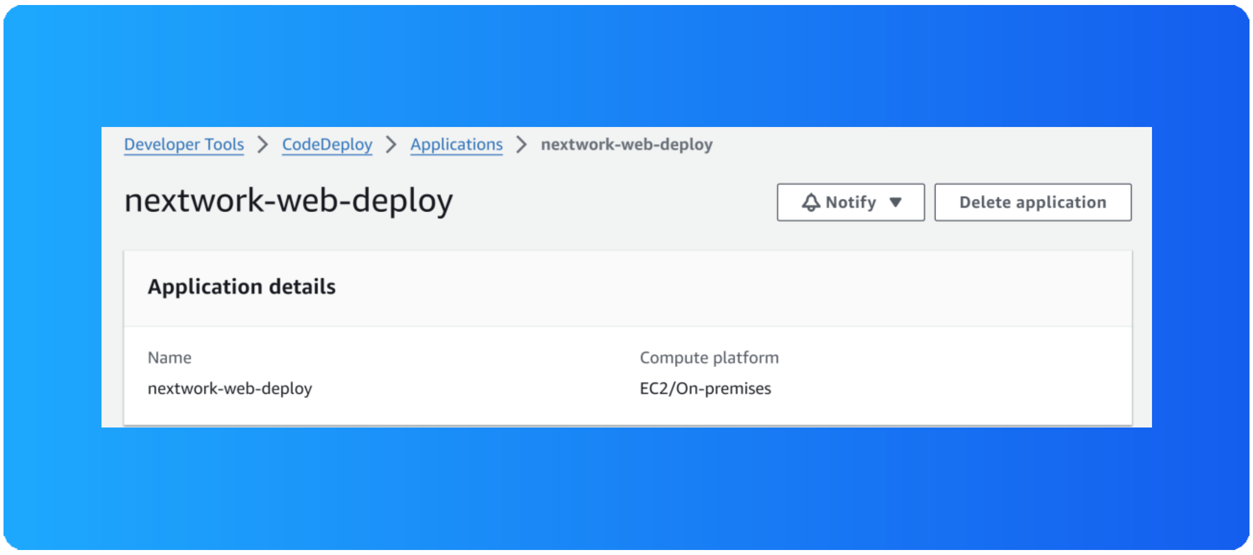
Figure 14: IAM role for CodeDeploy

CodeDeploy application

A CodeDeploy application refers to a saved configuration/setup template for deploying my web app.

To create a CodeDeploy application, I needed to select a compute platform, which determines the environment where my web application will be hosted. Each compute platform comes with its own set of requirements for deploying a web app.

The compute platform I chose was EC2 because I am using an EC2 instance as my web server. Choosing EC2 gives me the most control of all the three compute platform options, and it's also beneficial for my learning, as I will be exposed to more configuration options


Developer Tools > CodeDeploy > Applications > nextwork-web-deploy
Notify
Delete application

nextwork-web-deploy

Application details

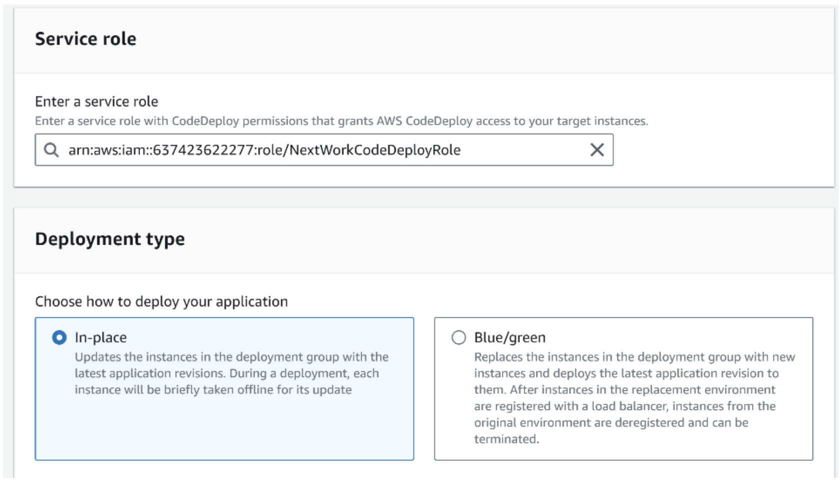
Name	Compute platform
nextwork-web-deploy	EC2/On-premises

Deployment group

A deployment group means the configurations for a specific deployment scenario.

To create a deployment group, I set up a:

1. A service role: It is an IAM role that I am using for my deployment group to grant CodeDeploy access to my EC2 web server instance.
2. Deployment type: The deployment type is the mode that CodeDeploy will use to carry out your web app's deployment. I chose **In-place**, which means the application will be updated on the existing EC2 instances.



Service role

Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
arn:aws:iam::637423622277:role/NextWorkCodeDeployRole

Deployment type

Choose how to deploy your application

[✓ In-place] Updates the instances in the deployment group with the latest application revisions. During a deployment, each instance will be briefly taken offline for its update

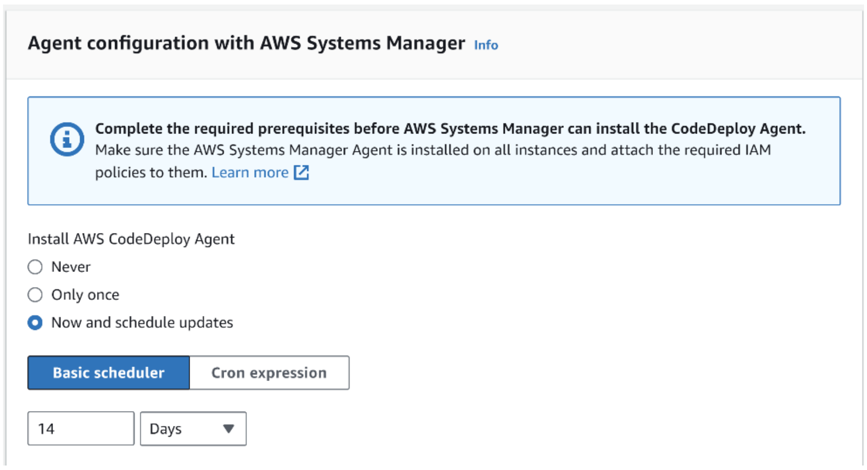
Blue/green: Replaces the instances in the deployment group with new instances and deploys the latest application revision to them. After instances in the replacement environment are registered with a load balancer, instances from the original environment are deregistered and can be terminated.

Figure 15: service role and deployment type in settings

3. Environment configuration: It refers to the type of servers that will be used to deploy my web app—in this case, I used Amazon EC2 instances without the need for Auto Scaling groups.

4. The CodeDeploy Agent: It is a CodeDeploy's helper that communicates with my EC2 web server and ensures it follows the deployment instructions.

5. Deployment setting: It is like rules for how the deployment should happen, whether the deployment should occur in different stages or all at once.


Agent configuration with AWS Systems Manager Info

Complete the required prerequisites before AWS Systems Manager can install the CodeDeploy Agent.
Make sure the AWS Systems Manager Agent is installed on all instances and attach the required IAM policies to them. Learn more

Install AWS CodeDeploy Agent
- [ ] Never
- [ ] Only once
- [✓] Now and schedule updates

14 Days

Creating the deployment

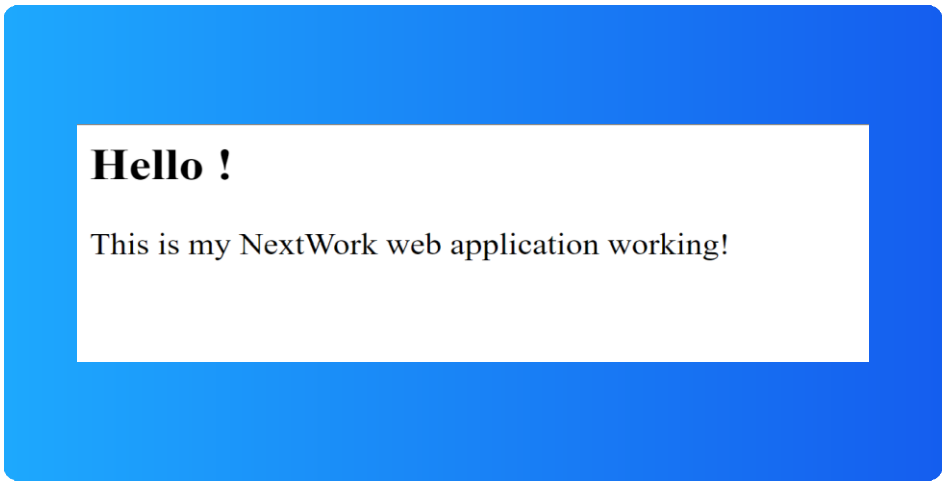
1. I pasted the S3 URI of the WAR file stored in S3 bucket into the Revision location field in the deployment settings.

2. .zip is still selected as the Revision file type.

3. Now, I clicked **create deployment**.

4. Next, I went to EC2 instance and opened the Public IPv4 address. It opened with https format which I edited manually as http.

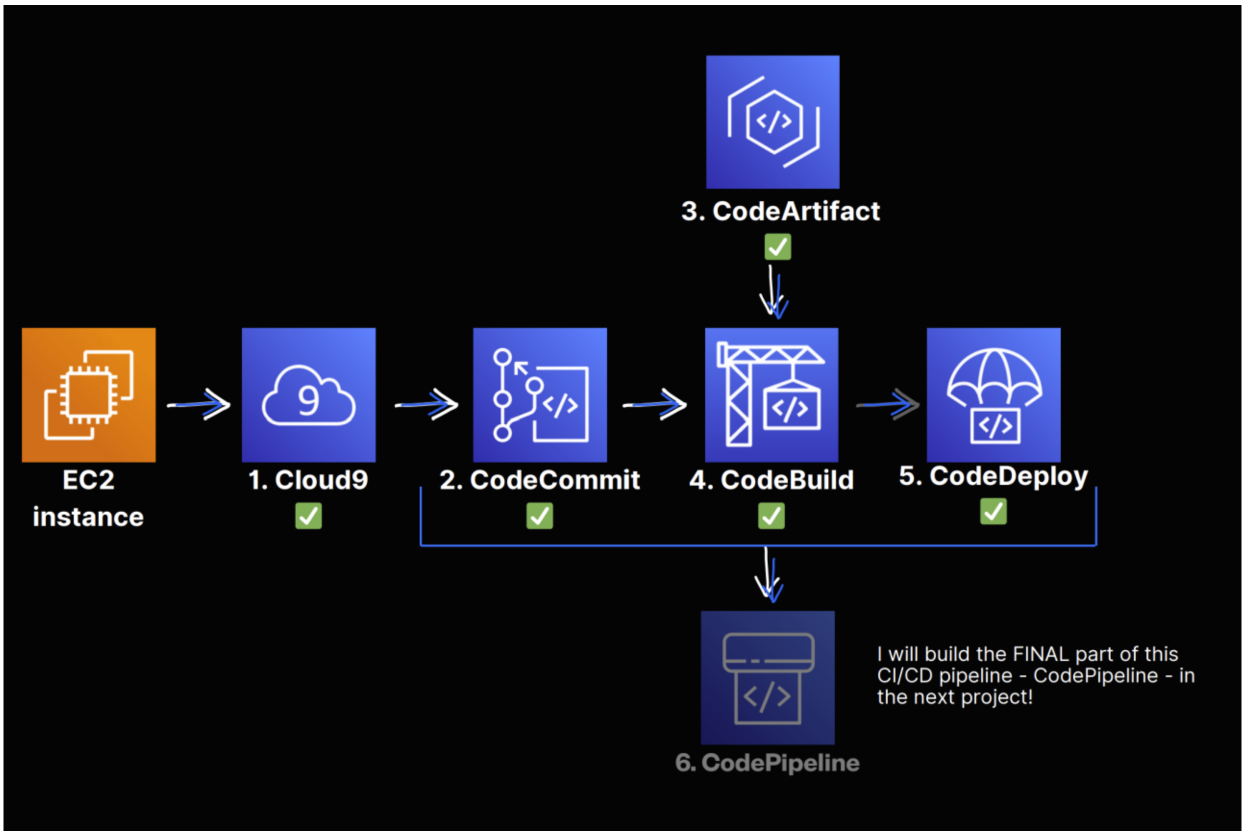
After the webpage was loading sometime, it displayed my web app. The Below diagram shows the display of my web app.


Hello !

This is my NextWork web application working!

My CI/CD pipeline so far...

1. AWS Cloud9 is responsible for debugging, editing and modifying the code and output of my webapp.
2. AWS CodeCommit is responsible for storing my progress and changes I make on my IDE.
3. AWS CodeArtifact is responsible for securing my dependencies needed for developments.
4. AWS CodeBuild is responsible for compiling my source codes, run tests and produce packages which will be used for deployments.
5. AWS CodeDeploy is responsible for making my web application live to the public users.



Key learnings from this project:

1. **Understanding Deployment Pipelines**: I gained valuable insights into setting up and managing deployment pipelines using AWS CodeDeploy, which enhanced my understanding of automating application deployments and ensuring consistent updates across environments.

2. **Configuration Management**: The project deepened my knowledge of configuration management tools and best practices, allowing me to effectively handle complex configurations and streamline the deployment process.

3. **Collaboration and Version Control**: I learned the importance of collaboration in a team setting and how using version control systems like AWS CodeCommit facilitates smooth workflows, enabling efficient tracking of changes and improving team productivity.

Everyone should be in a job they love.

Check out nextwork.org for more projects