



Deploy an App with CodeDeploy



miguelhhuerto@gmail.com

```
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
```

Compressing objects: 100% (2/2), done.
writing objects: 100% (2/2), 106.00 KiB/s, done.
Total 2 (delta 1), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To https://github.com/miguelhhuerto/nextwork-deploy-webapp.git
 * [new branch] main -> main
 02c6c4b..4953200 main > main

Introducing today's project!

What is AWS CodeDeploy?

CodeDeploy is a managed service by AWS that assists users in automating and managing much of the deployment process for us.

How I'm using AWS CodeDeploy in this project

I used CodeDeploy to deploy my web application and make it accessible from the public internet.

One thing I didn't expect...

I didn't expect the problems I would encounter and the solutions I was able to think of to overcome them.

This project took me...

This project took me around 2 hours to set up everything from the previous projects, up until deployment of the application.

Set up an EC2 instance

I set up an EC2 instance and VPC because this is where the application will be deployed and run from, separate from the previous instance.

We manage production and development environments separately because we want to avoid any possible code changes in the production environment.

To set up my EC2 instance and VPC, I used a CloudFormation template.

Events (40)				
<input type="text"/> Search events				
Timestamp	Logical ID	Status	Detailed status	S
2025-02-07 14:29:49 UTC+0800	NextWorkEC2VPCStack	CREATE_COMPLETE	-	-
2025-02-07 14:29:47 UTC+0800	DeployRoleProfile	CREATE_COMPLETE	-	-
2025-02-07 14:28:53 UTC+0800	WebServer	CREATE_COMPLETE	-	-
2025-02-07 14:28:52 UTC+0800	NextWorkEC2VPCStack	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Ecl
2025-02-07 14:28:52 UTC+0800	WebServer	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Ecl
2025-02-07 14:28:41 UTC+0800	WebServer	CREATE_IN_PROGRESS	-	Rlr
2025-02-07 14:27:43 UTC+0800	PublicInternetRoute	CREATE_COMPLETE	-	-
2025-02-07 14:27:42		CREATE_IN_PROGRESS		R

Bash scripts

Scripts are a way to automate certain commands on the terminal.

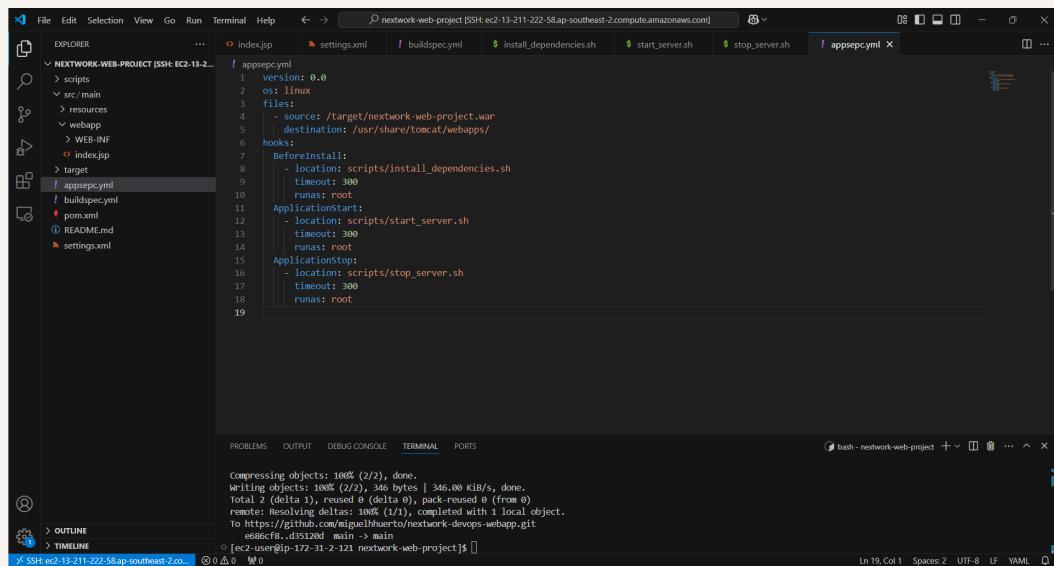
I used three scripts for my project's deployment

The first script I created was `install_dependencies.sh`. It ensures that the web server can handle and serve Java applications and web pages.

The second script I created was `start_server.sh`. This script starts all the necessary services necessary for the application to run.

The third script I created was `stop_server.sh`. It is used to stop the Tomcat and HTTPD services on the EC2 instance.

Bash scripts



The screenshot shows a terminal window titled "bash - netwerk-web-project" connected via SSH to an AWS EC2 instance. The window displays a Bash script named "appsepcyml". The script defines a "version" of 0.0, specifies "os: linux", and lists "files" to be deployed from "/target/netwerk-web-project.war" to "/usr/share/tomcat/webapps". It includes a "hooks" section with "BeforeInstall" and "ApplicationStart" events, both using the "install_dependencies.sh" script with a timeout of 300 seconds. The "ApplicationStop" event uses the "stop_server.sh" script with a timeout of 300 seconds. The "runas" parameter is set to "root" for all commands.

```
version: 0.0
os: linux
files:
  - source: /target/netwerk-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
```

Below the script, the terminal shows the execution of a "git push" command, compressing objects and writing them to the repository. The command "git push" is shown at the bottom of the terminal window.

CodeDeploy's IAM Role

I created an IAM service role for CodeDeploy because it needs access and permissions to the EC2 instance.

To set up CodeDeploy's IAM role, I chose CodeDeploy as the service for the IAM Role and chose AWSCodeDeployRole.

AWSCodeDeployRole

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

[Copy JSON](#)

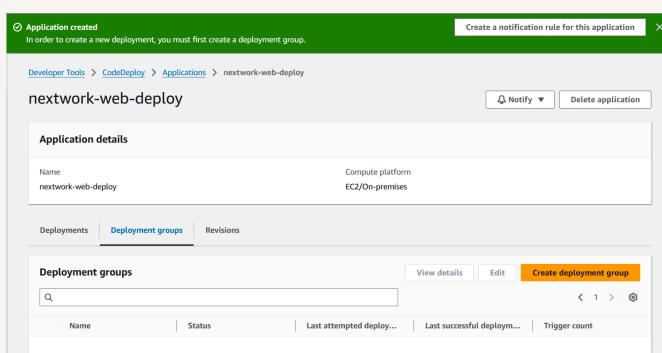
```
1 ~ [ {  
2     "Version": "2012-10-17",  
3     "Statement": [  
4         {  
5             "Effect": "Allow",  
6             "Action": [  
7                 "autoscaling:CompleteLifecycleAction",  
8                 "autoscaling:DeleteLifecycleHook",  
9                 "autoscaling:DescribeAutoScalingGroups",  
10                "autoscaling:DescribeLifecycleHooks",  
11                "autoscaling:PutLifecycleHook",  
12                "autoscaling:RecordLifecycleActionHeartbeat",  
13                "autoscaling:CreateAutoScalingGroup",  
14                "autoscaling:CreateOrUpdateTags",  
15                "autoscaling:UpdateAutoScalingGroup",  
16                "autoscaling:EnableMetricsCollection",  
17                "autoscaling:DescribePolicies",  
18                "autoscaling:DescribeScheduledActions",  
19                "autoscaling:DescribeNotificationConfigurations",  
20                "autoscaling:SuspendProcesses",  
21            ]  
22        }  
23    ]  
24}
```

CodeDeploy application

A CodeDeploy application means streamlined deployment processes and consistency across deployments.

To create a CodeDeploy application, I had to select a compute platform, which means it can deploy applications to various environment options.

The compute platform I chose was EC2/On-premises because it provides virtual servers that we can fully control.



Deployment group

A deployment group means the instructions and configurations for one specific deployment scenario.

Two key configurations for a deployment group

Environment means the kind of servers to use for the application.

A CodeDeploy Agent is a helper used to communicate with CodeDeploy for instructions on deploying the application. It is installed on the EC2 instance.

The screenshot shows a success message: "Success Deployment group created". Below it, the navigation path is: Developer Tools > CodeDeploy > Applications > nextwork-web-deploy > nextwork-web-deploy-group. The main title is "nextwork-web-deploy-group". On the right, there are "Edit", "Delete", and "Create deployment" buttons. The "Create deployment" button is highlighted in orange. The "Deployment group details" section contains the following information:

Deployment group name	Application name	Compute platform
nextwork-web-deploy-group	nextwork-web-deploy	EC2/On-premises
Deployment type	Service role ARN	Deployment configuration
In-place	arn:aws:iam::084828560751:role/NextWorkCodeDeployRole	CodeDeployDefault.AllAtOnce
Rollback enabled	Agent update scheduler	
False	Learn to schedule update in AWS Systems Manager	

The "Environment configuration: Amazon EC2 instances" section is present but empty, showing a table with columns "Key" and "Value".

MI

miguelhhuerto@gmail.com

NextWork Student

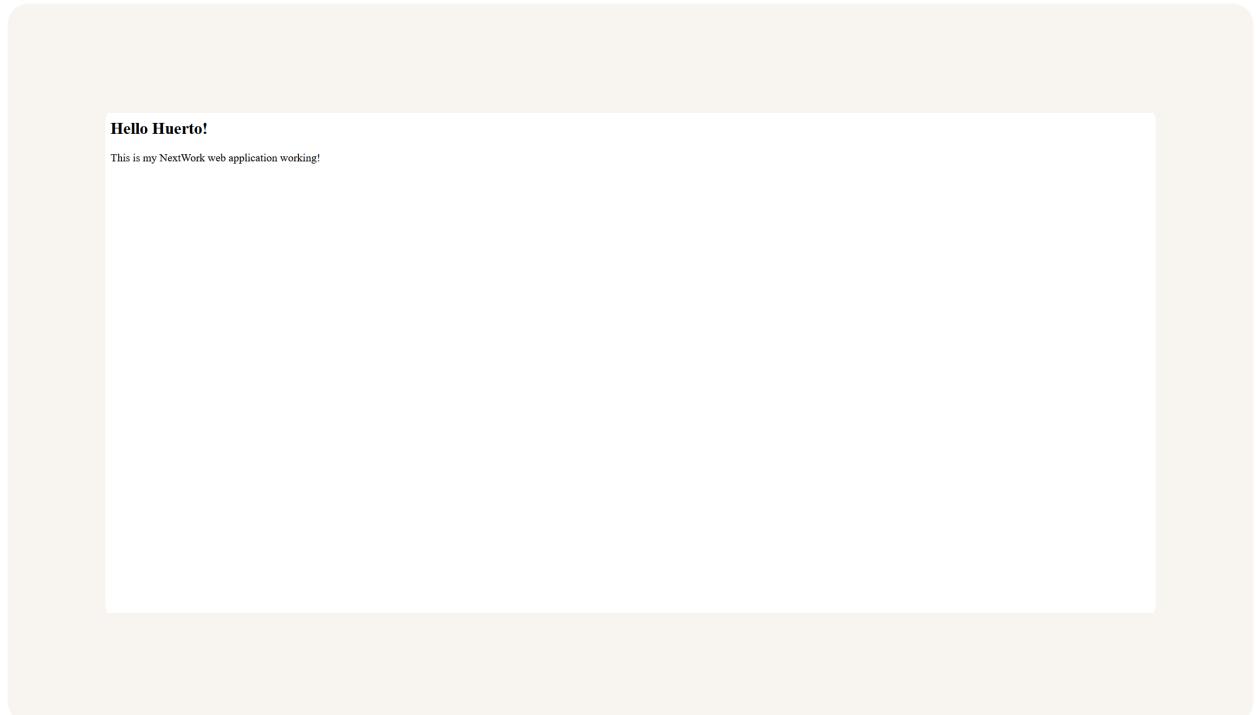
NextWork.org

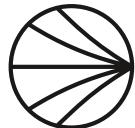
CodeDeploy application

To create my deployment, I had to set up a revision location, which means the location of where the application's build artifacts(zip file or WAR file). This is where CodeDeploy is instructed to look.

My revision location was my S3 bucket where the WAR file was stored.

To visit my web app, I had to visit the public ipv4 address of the web app instance using http.





NextWork.org

Everyone should be in a job they love.

Check out nextwork.org for
more projects

