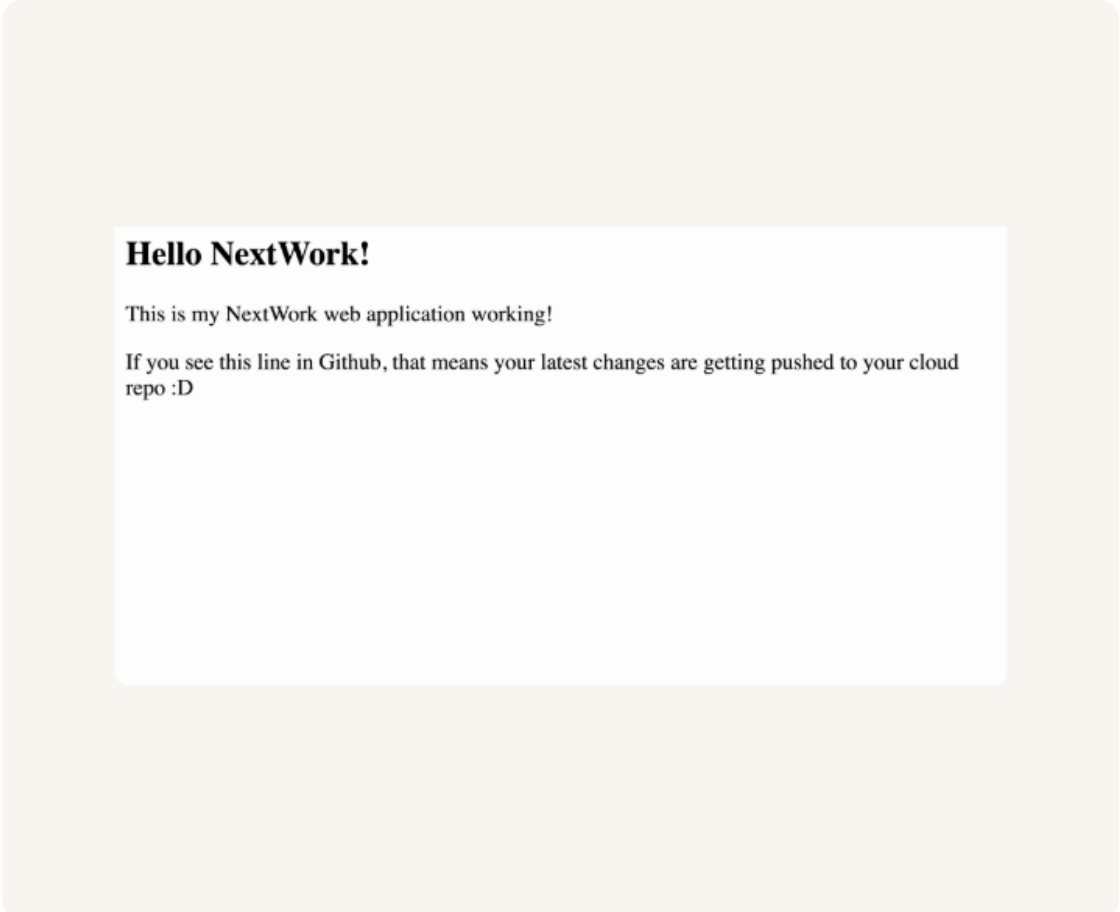
nextwork.org

Deploy a Web App with CodeDeploy

ZA zakaria.belkacem94@gmail.com



Introducing Today's Project!

In this project, I will demonstrate AWS CodeDeploy automation to replace manual deployments. I'm doing this project to learn how CodeDeploy manages app releases.

Key tools and concepts

Services I used were CodeArtifact, CodeBuild, CodeDeploy, CloudFormation, S3, and IAM. Key concepts I learnt include CI/CD pipelines, infrastructure-as-code, deployment strategies, and secure AWS permissions management.

Project reflection

This project took me approximately 3 hours. The most challenging part was debugging deployment failures, but it was most rewarding to see the app auto-deploy after fixing the scripts.

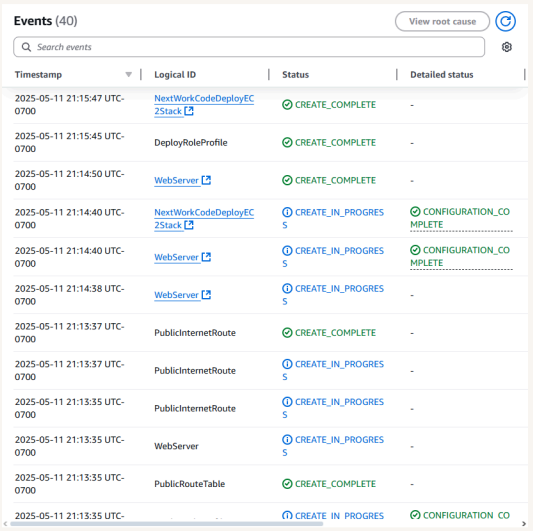
This project is part five of a series of DevOps projects where I'm building a CI/CD pipeline! I'll be working on the next project tomorrow.

Deployment Environment

To set up for CodeDeploy, I launched an EC2 instance and VPC because CodeDeploy needs compute resources to deploy to and a network environment to operate within.

Instead of launching these resources manually, I used AWS CloudFormation to automate my EC2 and VPC deployment with IaC. When I need to delete these resources, I'll just delete the CloudFormation stack to clean up everything automatically.

Other resources created in this template include security groups, IAM roles, and subnets. they're in the template because security groups protect the instance, IAM roles for automated deployments, and subnets ensure proper network isolation.


Events (40)
View root cause
Search events
Timestamp
Logical ID
Status
Detailed status
2025-05-11 21:15:47 UTC-0700
NextWorkCodeDeployEC2Stack
CREATE_COMPLETE
-
2025-05-11 21:15:45 UTC-0700
DeployRoleProfile
CREATE_COMPLETE
-
2025-05-11 21:14:50 UTC-0700
WebServer
CREATE_COMPLETE
-
2025-05-11 21:14:40 UTC-0700
NextWorkCodeDeployEC2Stack
CREATE_IN_PROGRESS
CONFIGURATION_COMPLETE
2025-05-11 21:14:40 UTC-0700
WebServer
CREATE_IN_PROGRESS
CONFIGURATION_COMPLETE
2025-05-11 21:14:38 UTC-0700
WebServer
CREATE_IN_PROGRESS
-
2025-05-11 21:13:37 UTC-0700
PublicInternetRoute
CREATE_COMPLETE
-
2025-05-11 21:13:37 UTC-0700
PublicInternetRoute
CREATE_IN_PROGRESS
-
2025-05-11 21:13:35 UTC-0700
PublicInternetRoute
CREATE_IN_PROGRESS
-
2025-05-11 21:13:35 UTC-0700
WebServer
CREATE_IN_PROGRESS
-
2025-05-11 21:13:35 UTC-0700
PublicRouteTable
CREATE_COMPLETE
-
2025-05-11 21:13:35 UTC-
CREATE_IN_PROGRES
CONFIGURATION_CO

Deployment Scripts

Scripts are executable files containing commands to automate tasks. To set up CodeDeploy, I also wrote scripts to install all the dependencies needed.

Install_dependencies will set up all the software needed to run our website by installing programs like Tomcat and Apache, It then creates special settings that let these programs work together.

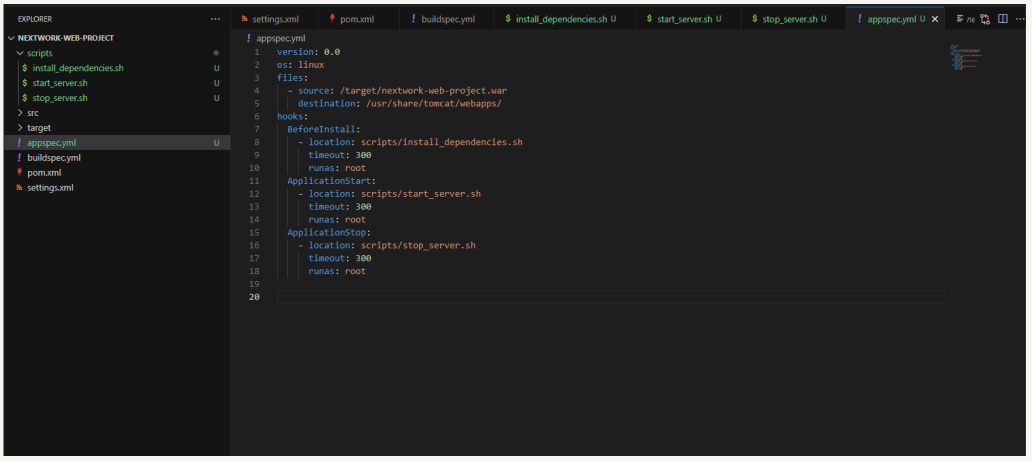
start_server.sh will start Tomcat and Apache and make sure they restart automatically if the EC2 instance reboots.

Stop_server.sh will automatically stop web server services by first checking if they're running. This prevents errors that could occur from trying to stop services that aren't running.

appspec.yml

Then, I wrote an appspec.yml file to give instruction to CodeDeploy, The key sections in appspec.yml are version: 0.0, os: linux, files, hooks.

I also updated buildspec.yml because it needs to include the additional files we just created. because CodeDeploy will need them to properly deploy the application.

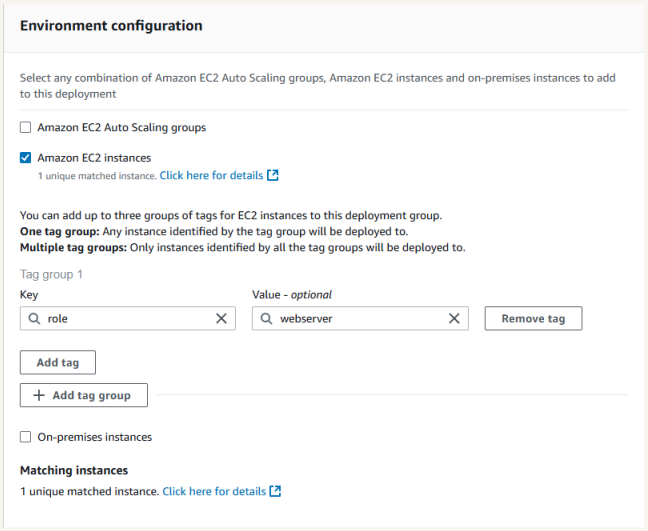


Setting Up CodeDeploy

A deployment group is where code deploys (target instances + rules). A CodeDeploy application is what deploys (container for code + configs).

To set up a deployment group, you also need to create an IAM role to grant CodeDeploy permissions to access your AWS resources (like EC2, S3, or Lambda) during deployments. ensuring secure and controlled automation

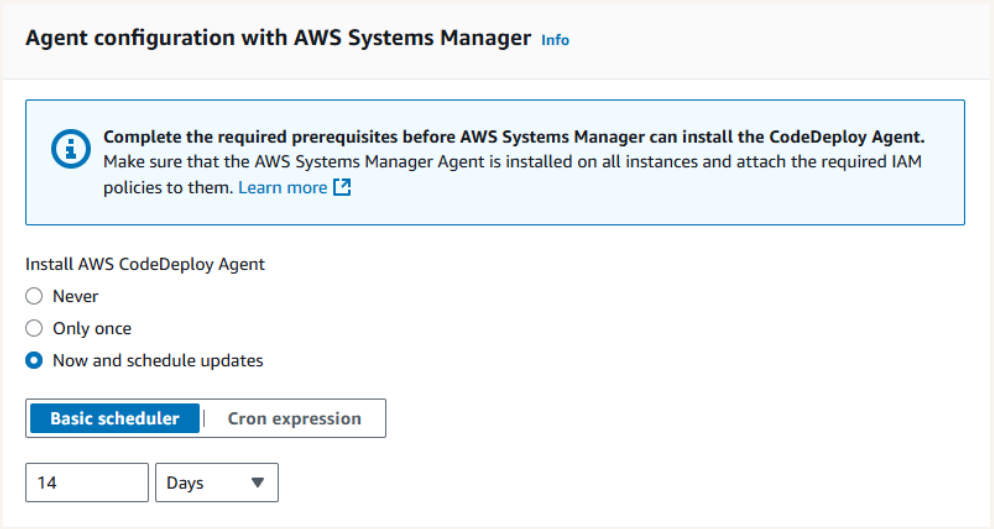
Tags are helpful for dynamically grouping EC2 instances without hardcoding IDs. I used the tag role=webserver to let CodeDeploy automatically find and deploy to all matching instances in the Cloudfornation template.


Environment configuration
Select any combination of Amazon EC2 Auto Scaling groups, Amazon EC2 instances and on-premises instances to add to this deployment
Amazon EC2 Auto Scaling groups
Amazon EC2 instances
1 unique matched instance. Click here for details
You can add up to three groups of tags for EC2 instances to this deployment group.
One tag group: Any instance identified by the tag group will be deployed to.
Multiple tag groups: Only instances identified by all the tag groups will be deployed to.
Tag group 1
Key
Value - optional
role
webserver
Remove tag
Add tag
Add tag group
On-premises instances
Matching instances
1 unique matched instance. Click here for details

Deployment configurations

Another key setting is the deployment configuration, which affects how your app rolls out across instances. I used CodeDeployDefault.AllAtOnce, so it deploys to all instances simultaneously for fast updates (ideal for dev/staging).

In order to connect your EC2 instances to CodeDeploy, a CodeDeploy Agent is also set up to update every 14 days to make sure it's always up to date.

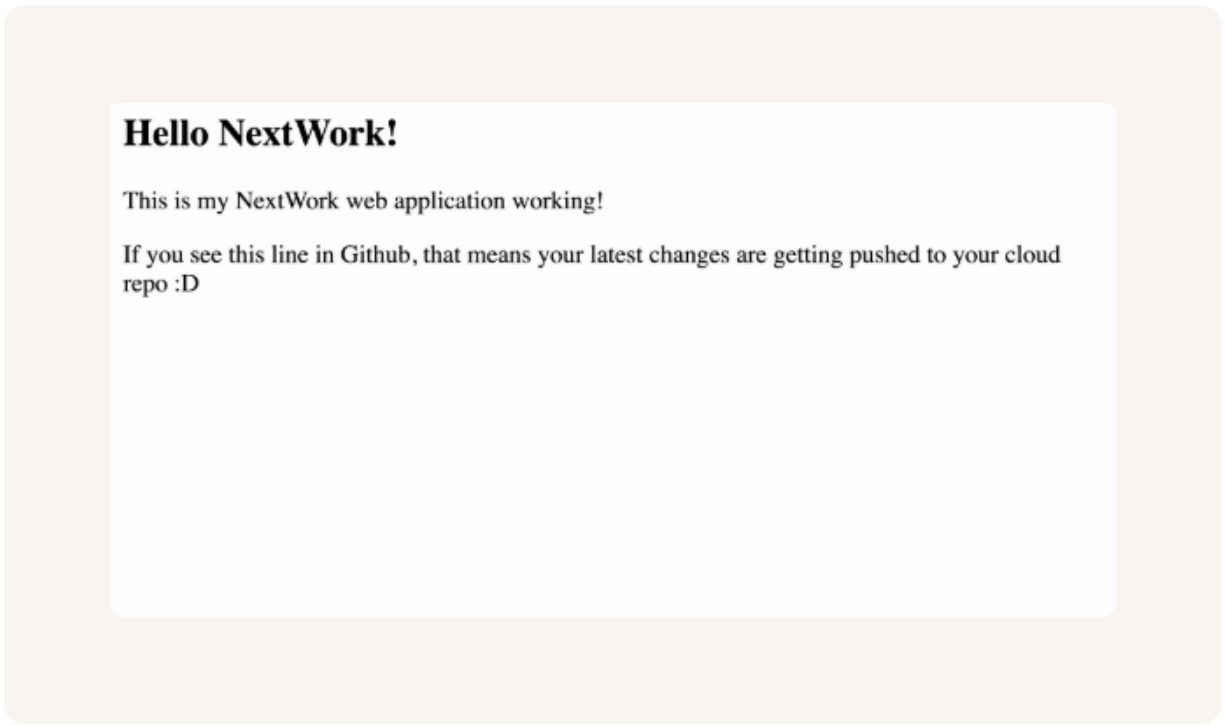


Success!

A CodeDeploy deployment is the process of releasing a specific application version to instances. The difference to a deployment group is that the group defines where to deploy, while a deployment is the execution of that rollout.

I had to configure a revision location, which means specifying where CodeDeploy finds your application files (like S3 or GitHub). My revision location was an S3 bucket containing my bundled app code and appspec.yml.

To check that the deployment was a success, I visited the CodeDeploy console's deployment details tab and saw the succeeded status and then I hit the ec2 endpoint and verified as well.


Hello NextWork!
This is my NextWork web application working!
If you see this line in Github, that means your latest changes are getting pushed to your cloud repo :D

The place to learn & showcase your skills

Check out nextwork.org for more projects