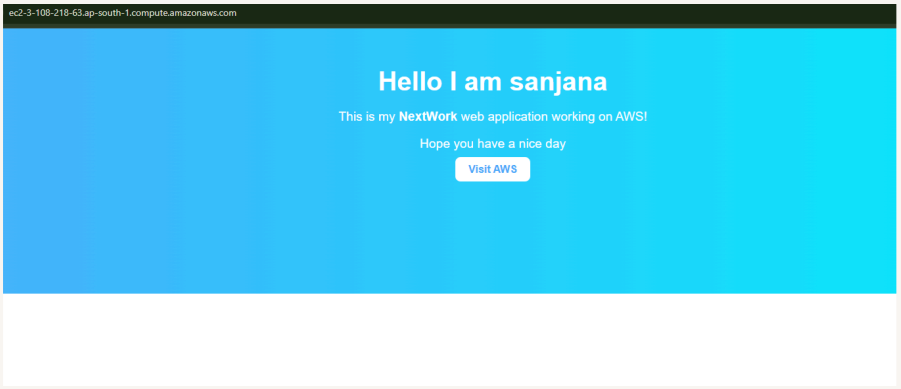
nextwork.org

Deploy a Web App with CodeDeploy

Sanjana Tripathy



Sanjana Tripathy
NextWork Student

Introducing Today's Project!

In this project, I'm learning AWS CodeDeploy by automating web app deployments. Using VS Code, GitHub, CodeArtifact, and CodeBuild, I'll deploy a WAR file to servers, write scripts, and practice rollback for reliable releases.

Key tools and concepts

Services I used were AWS CodeArtifact, CodeBuild, CodeDeploy, CloudFormation, EC2, and IAM. Key concepts I learnt include Infrastructure as Code, deployment scripts, lifecycle hooks, and automating deployments reliably,deployment groups

Project reflection

This project took me approximately 3 hours. The most challenging part was minimal, as the steps were clear. It was most rewarding to complete each step successfully and see the web app live on my screen.

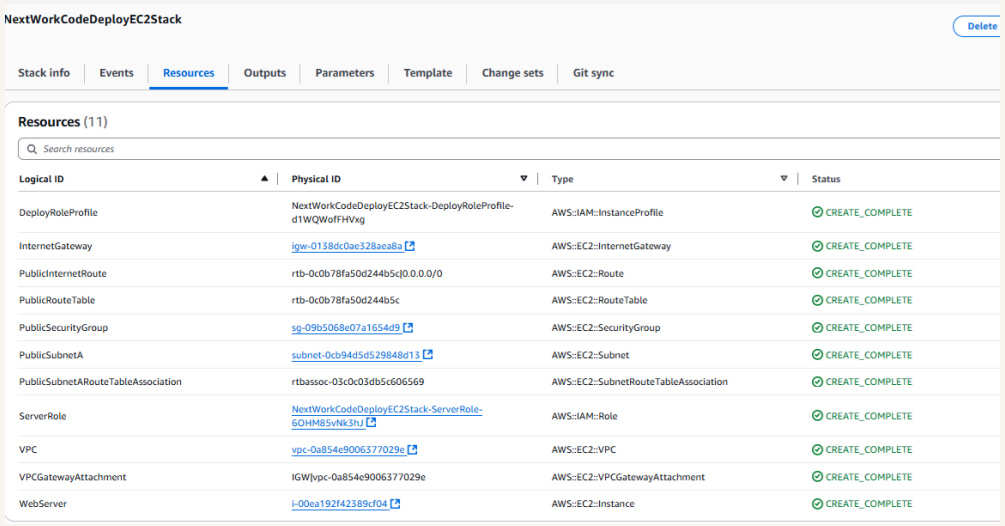
This project is part five of a series of DevOps projects where I'm building a CI/CD pipeline! I'll be working on the next project soon to continue enhancing my hands-on skills in automated deployments and AWS DevOps tools.

Deployment Environment

To set up for CodeDeploy, I launched an EC2 instance and VPC because I need a secure server and network for my application. This ensures deployments happen in a dedicated production environment, separate from development, for safety and reliability.

Instead of launching these resources manually, I used CloudFormation to automate setup through Infrastructure as Code. When I need to delete these resources, I can simply remove the CloudFormation stack, and everything is cleaned up at once.

Other resources created in this template include a subnet, route table, internet gateway, and security group. They're also in the template because they because a web server needs more than just compute—it needs secure netorking and internet access.


NextWorkCodeDeployEC2Stack
Delete
Stack info
Events
Resources
Outputs
Parameters
Template
Change sets
Git sync
Resources (11)
Search resources
Logical ID
Physical ID
Type
Status
DeployRoleProfile
NextWorkCodeDeployEC2Stack-DeployRoleProfile-d1WQWofFHVxg
AWS::IAM::InstanceProfile
CREATE_COMPLETE
InternetGateway
igw-0138dc0ae328aea8a
AWS::EC2::InternetGateway
CREATE_COMPLETE
PublicInternetRoute
rtb-0c0b78fa50d244b5c|0.0.0.0/0
AWS::EC2::Route
CREATE_COMPLETE
PublicRouteTable
rtb-0c0b78fa50d244b5c
AWS::EC2::RouteTable
CREATE_COMPLETE
PublicSecurityGroup
sg-09b5068e07a1654d9
AWS::EC2::SecurityGroup
CREATE_COMPLETE
PublicSubnetA
subnet-0cb94d5d529848d13
AWS::EC2::Subnet
CREATE_COMPLETE
PublicSubnetARouteTableAssociation
rtbassoc-03c0c03db5c606569
AWS::EC2::SubnetRouteTableAssociation
CREATE_COMPLETE
ServerRole
NextWorkCodeDeployEC2Stack-ServerRole-6OHM85vNk3hJ
AWS::IAM::Role
CREATE_COMPLETE
VPC
vpc-0a854e9006377029e
AWS::EC2::VPC
CREATE_COMPLETE
VPCGatewayAttachment
IGW|vpc-0a854e9006377029e
AWS::EC2::VPCGatewayAttachment
CREATE_COMPLETE
WebServer
i-00ea192f42389cf04
AWS::EC2::Instance
CREATE_COMPLETE

Deployment Scripts

Scripts are small programs made of commands that automate tasks you'd otherwise type manually. To set up CodeDeploy, I also wrote scripts to install dependencies, stop the server, and start it again after deployment.

The install_dependencies.sh will set up the software needed for deployment by installing Apache and Tomcat, then configuring them to work together so the web app can run smoothly and be accessible to users on the internet.

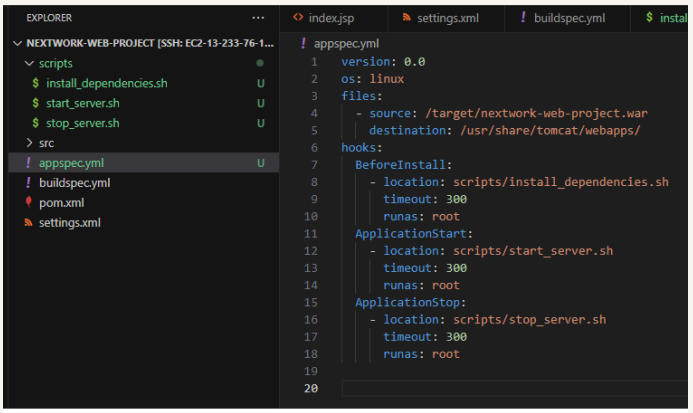
The start_server.sh will start Apache and Tomcat, ensuring both services run properly. It also configures them to restart automatically if the EC2 instance reboots, keeping the web app consistently available.

The stop_server.sh will safely stop Apache and Tomcat by first checking if they're running. It only stops active services, preventing errors and ensuring the deployment process continues smoothly without interruptions.

appspec.yml

Then, I wrote an appspec.yml file to act as CodeDeploy's instruction manual, mapping files and scripts to each step in deployment. The key sections in appspec.yml are version, os, files, and hooks that run scripts at specific lifecycle events.

I also updated buildspec.yml because CodeBuild needs to package not just the compiled app but also appspec.yml and scripts. This ensures CodeDeploy has the instructions and files it needs to deploy the application correctly.

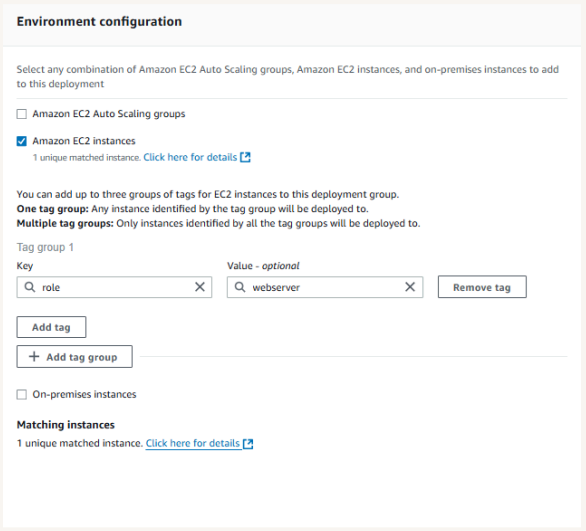


Setting Up CodeDeploy

A CodeDeploy application is like a container that represents what you want to deploy. A deployment group is a set of rules and target instances that define where and how that application gets deployed.

To set up a deployment group, you also need to create an IAM role to give CodeDeploy permissions to access EC2, pull artifacts from S3, update Auto Scaling groups, and write deployment logs to CloudWatch.

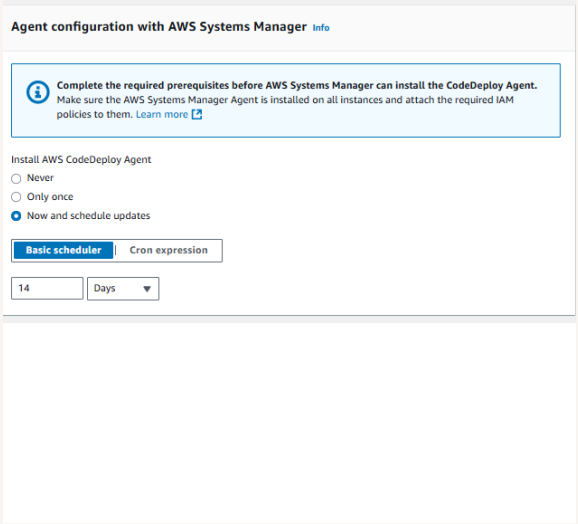
Tags are helpful for targeting resources, scaling easily, and documenting roles. I used the tag role:webserver to let CodeDeploy identify and deploy only to the correct EC2 instance.


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 quickly and safely updates are rolled out.I used CodeDeployDefault.AllAtOnce,so the app deploys to all instances at the same time,which is fast and fine for a single-instance setup

In order to connect CodeDeploy to the EC2 instance, I installed the CodeDeploy Agent. A CodeDeploy Agent is also set up to communicate with AWS, receive deployment instructions, and execute scripts on the instance.

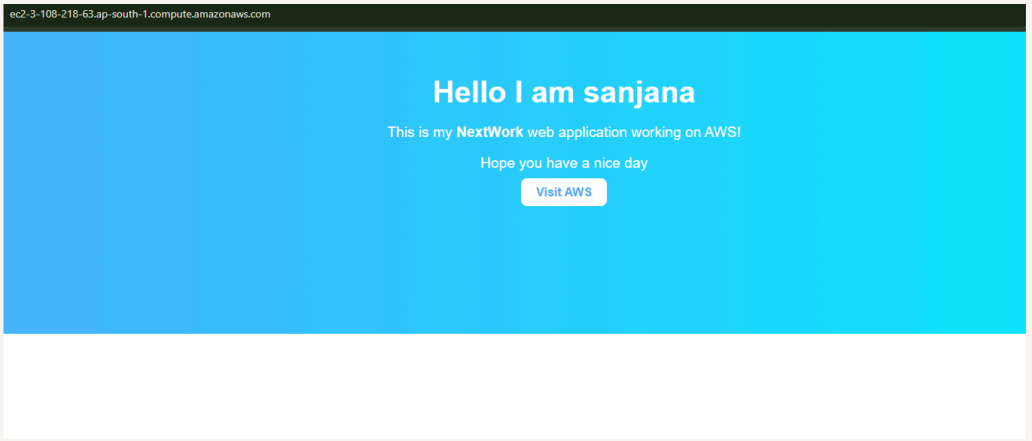


Success!

A CodeDeploy deployment is the actual process of delivering and updating an application to target instances. The difference to a deployment group is that the group defines where and how deployments happen, while the deployment is the execution itself

I had to configure a revision location, which means specifying where CodeDeploy can find the build artifacts for deployment. My revision location was the S3 bucket containing the WAR file for my web application.

To check that the deployment was a success, I visited the public IPv4 DNS of my EC2 instance in a browser. I saw my web application running, confirming that CodeDeploy successfully deployed the WAR file to the server.


ec2-3-108-218-63.ap-south-1.compute.amazonaws.com
Hello I am sanjana
This is my NextWork web application working on AWS!
Hope you have a nice day
Visit AWS

The place to learn & showcase your skills

Check out nextwork.org for more projects