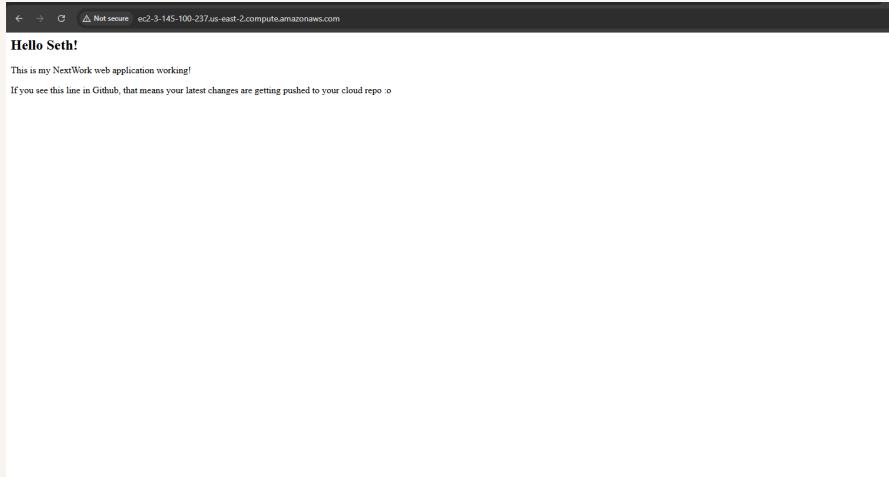


Deploy a Web App with CodeDeploy



Seth Sekyere





Seth Sekyere

Introducing Today's Project!

In this project, I will demonstrate how to automate deployments using AWS CodeDeploy. I'm doing this project to learn how to release applications reliably, minimize downtime, and integrate deployment into a CI/CD pipeline.

Key tools and concepts

Services I used were AWS CodeDeploy, EC2, S3, and CloudFormation. Key concepts I learnt include continuous deployment, deployment scripts, appspec.yml, lifecycle hooks, Infrastructure as Code, and automating application rollouts.

Project reflection

This project took me approximately 2 hours. The most challenging part was writing and configuring the deployment scripts correctly. It was most rewarding to see my web app successfully deployed and running on the EC2 instance.

This project is part five of a DevOps series where I'm building a CI/CD pipeline! I'll be working on the next project tonight to continue learning deployment strategies and automation.



Seth Sekyere

Deployment Environment

To set up for CodeDeploy, I launched an EC2 instance and VPC because the EC2 will host my web app and the VPC provides the secure networking environment it needs for deployment and access.

Instead of launching these resources manually, I used AWS CloudFormation. When I need to delete these resources, I can simply delete the stack, and CloudFormation removes everything automatically.

Other resources created in this template include a subnet, route tables, an internet gateway, and a security group. They're included to provide networking, routing, internet access, and firewall rules for the EC2 instance.

Seth Sekyere

The screenshot shows the AWS CloudFormation console with the 'Resources' tab selected. The table displays the following resources:

Logical ID	Physical ID	Type	Status	Module
DeployRoleProfile	NextWorkCodeDeployEC2Stack-DeployRoleProfile-NcVV8NVWHWN7	AWS::IAM::InstanceProfile	CREATE_COMPLETE	-
InternetGateway	igw-019c78dd159bfa506	AWS::EC2::InternetGateway	CREATE_COMPLETE	-
PublicInternetRoute	rtb-0daa02b854e166fc0 0.0.0/0	AWS::EC2::Route	CREATE_COMPLETE	-
PublicRouteTable	rtb-0daa02b854e166fc0	AWS::EC2::RouteTable	CREATE_COMPLETE	-
PublicSecurityGroup	sg-041b61ad84a71386a	AWS::EC2::SecurityGroup	CREATE_COMPLETE	-
PublicSubnetA	subnet-030aaef023be3c053	AWS::EC2::Subnet	CREATE_COMPLETE	-
PublicSubnetARouteTableAssociation	rtbassoc-043e80e31028ee305	AWS::EC2::SubnetRouteTableAssociation	CREATE_COMPLETE	-
ServerRole	NextWorkCodeDeployEC2Stack-ServerRole-4rpe5nRtPwI	AWS::IAM::Role	CREATE_COMPLETE	-
VPC	vpc-0654356a00ae7a4b	AWS::EC2::VPC	CREATE_COMPLETE	-
VPCGatewayAttachment	IGW vpc-0654356a00ae7a4b	AWS::EC2::VP CGatewayAttachment	CREATE_COMPLETE	-
WebServer	i-0181c9e0164ac3ff	AWS::EC2::Instance	CREATE_COMPLETE	-



Seth Sekyere

Deployment Scripts

Scripts are text files containing commands that automate tasks. To set up CodeDeploy, I also wrote scripts to install dependencies, start and stop the server, and configure the environment for deploying my web app.

The `install_dependencies.sh` script will install all software needed to run the web app (Tomcat and Apache) and configure Apache as a reverse proxy so it forwards web traffic to Tomcat.

The `start_server.sh` script will start Tomcat and Apache services and configure them to automatically run on server reboot, ensuring the web application stays up and accessible.

The `stop_server.sh` script will check if Apache ('httpd') and Tomcat are running, and if they are, it will stop their services. This ensures the web application and its server processes are safely shut down.

Seth Sekyere

appspec.yml

Then, I wrote an `appspec.yml` file to tell CodeDeploy how to deploy my application. The key sections in `appspec.yml` are version, os, files, and hooks, which define what files to copy and which scripts to run at each deployment stage

I also updated buildspec.yml because I needed to include the appspec.yml file and the scripts folder as artifacts, so CodeDeploy would have all the files required to deploy my application correctly.

```
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
```



Seth Sekyere

Setting Up CodeDeploy

A deployment group is the set of instances where the app is deployed with specific settings. A CodeDeploy application is the main container that organizes all its deployment groups and revisions.

To set up a deployment group, you also need to create an IAM role to give CodeDeploy permission to access EC2 instances, read artifacts from S3, and write logs to CloudWatch.

Tags are helpful for identifying and organizing resources. I used the tag 'Environment=Production' to tell CodeDeploy which EC2 instance should receive the deployment.



Seth Sekyere

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 - <i>optional</i>	Remove tag
<input type="text" value="role"/> X	<input type="text" value="webserver"/> X	<button>Remove tag</button>

[Add tag](#) [+ Add tag group](#)

On-premises instances

Matching instances
1 unique matched instance. [Click here for details](#)



Seth Sekyere

Deployment configurations

Another key setting is the deployment configuration, which affects how quickly and safely your app is rolled out. I used `CodeDeployDefault.AllAtOnce`, so the app deploys to all instances at once—fast but risky for multiple instances.

In order to connect CodeDeploy with an EC2 instance, a CodeDeploy Agent is installed on the instance. It receives deployment instructions from CodeDeploy and executes them, ensuring the application is deployed correctly.

The screenshot shows the 'Agent configuration with AWS Systems Manager' interface. At the top, there is a note: '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](#)'.

Under 'Install AWS CodeDeploy Agent', the option 'Now and schedule updates' is selected. Below this, there is a 'Basic scheduler' section with a cron expression set to '14 Days'.

The 'Deployment settings' section includes a 'Deployment configuration' dropdown set to 'CodeDeployDefault.AllAtOnce' and a 'Create deployment configuration' button. A note below says: 'Choose from a list of default and custom deployment configurations. A deployment configuration is a set of rules that determines how fast an application is deployed and the success or failure conditions for a deployment.'

The 'Load balancer' section contains the instruction: 'Select a load balancer to manage incoming traffic during the deployment process. The load balancer blocks traffic from each instance while it's being deployed and allows traffic to it again after the deployment succeeds.' A checkbox labeled 'Enable load balancing' is checked.



Seth Sekyere

Success!

A CodeDeploy deployment is a single rollout of your application, specifying which revision to deploy and how. The difference to a deployment group is that the group defines where and how deployments happen, while a deployment is an actual execution

I had to configure a revision location, which means telling CodeDeploy where to find the build artifacts for deployment. My revision location was the S3 bucket storing the WAR/zip file of my web app.

To check that the deployment was a success, I visited the public URL of my EC2 instance. I saw my web app running correctly, confirming the WAR file was deployed and the servers were up.



Seth Sekyere

