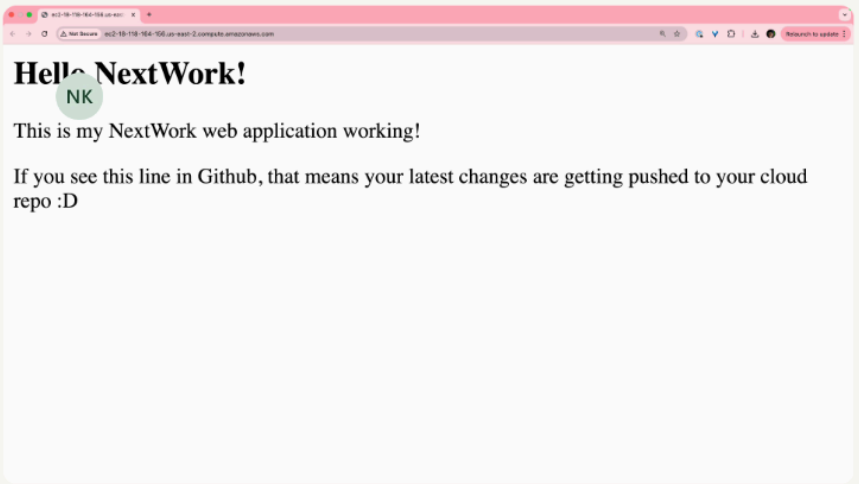
nextwork.org

Deploy a Web App with CodeDeploy

MO mohankrishnan802@gmail.com



Introducing Today's Project!

In this project, I will demonstrate how to automate application deployment using AWS CodeDeploy. I'm doing this project to learn how to deliver updates efficiently, avoid downtime, and ensure reliable deployments.

Key tools and concepts

Services I used were CloudFormation(IaC), CodeDeploy CodeBuild CodeArtifact, EC2 IAM. Key concepts I learnt include deployment scripts, rollback, continuous delivery, environment provisioning, monitoring, and versioning.

Project reflection

This project took me approximately 3 hrs to complete. The most challenging part was writing/testing the deploy scripts with seamless rollback. It was most rewarding to watch the pipeline auto-deploy updates reliably.

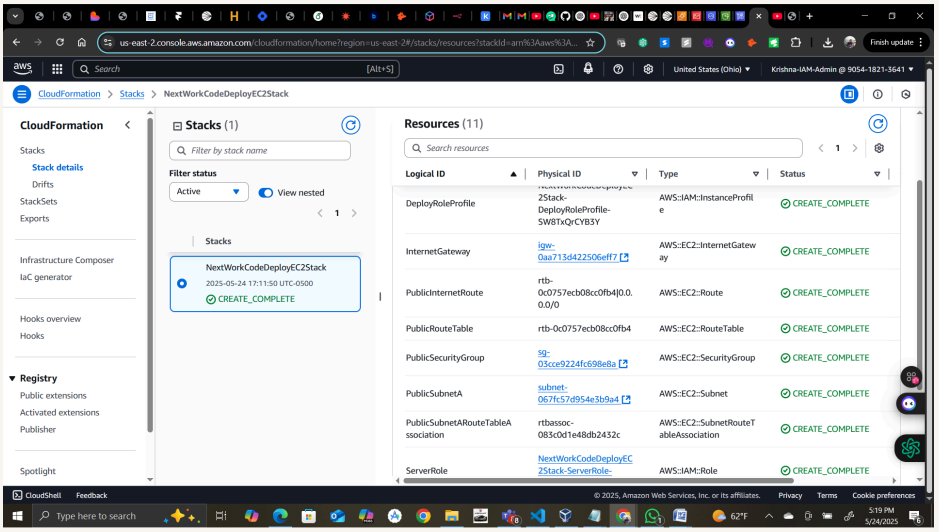
This project is part five of a series of DevOps projects where I'm building a CI/CD pipeline! I'll be working on the next project tomorrow to continue sharpening my deployment and automation skills.

Deployment Environment

While we're waiting, CloudFormation is automatically creating all resources we defined —like EC2, VPC, subnets, and security groups. This automation saves time, reduces errors, and ensures consistent infrastructure setup.

Instead of launching these resources manually, I used AWS CloudFormation to deploy my EC2 instance and VPC. When I need to delete these resources, I can simply delete the stack to remove everything at once.

The CloudFormation template is creating a VPC, subnet, route tables, internet gateway, security group, and EC2 instance. This setup ensures a secure, reproducible network infrastructure for hosting web apps.


us-east-2.console.aws.amazon.com/cloudformation/home?region=us-east-2#/stacks/resources?stackId=arn%3Aaws%3A...
Finish update
aws
Search
[Alt+S]
United States (Ohio)
Krishna-IAM-Admin @ 9054-1821-3641
CloudFormation > Stacks > NextWorkCodeDeployEC2Stack
CloudFormation
Stacks
Stack details
Drifts
StackSets
Exports
Infrastructure Composer
IaC generator
Hooks overview
Hooks
Registry
Public extensions
Activated extensions
Publisher
Spotlight
Stacks (1)
Filter by stack name
Filter status
Active
View nested
Stacks
NextWorkCodeDeployEC2Stack
2025-05-24 17:11:50 UTC-0500
CREATE_COMPLETE
Resources (11)
Search resources
Logical ID
Physical ID
Type
Status
DeployRoleProfile
2Stack-DeployRoleProfile-5W8TxQrCYB3Y
AWS::IAM::InstanceProfile
CREATE_COMPLETE
InternetGateway
igw-0aa713d422506eff7
AWS::EC2::InternetGateway
CREATE_COMPLETE
PublicInternetRoute
rtb-0c0757ecb08cc0fb4|0.0.0.0/0
AWS::EC2::Route
CREATE_COMPLETE
PublicRouteTable
rtb-0c0757ecb08cc0fb4
AWS::EC2::RouteTable
CREATE_COMPLETE
PublicSecurityGroup
sg-03cce9224fc698e8a
AWS::EC2::SecurityGroup
CREATE_COMPLETE
PublicSubnetA
subnet-067fc57d954e3b9a4
AWS::EC2::Subnet
CREATE_COMPLETE
PublicSubnetARouteTableAssociation
rtbassoc-083c0d1e48db2432c
AWS::EC2::SubnetRouteTableAssociation
CREATE_COMPLETE
ServerRole
NextWorkCodeDeployEC2Stack-ServerRole-
AWS::IAM::Role
CREATE_COMPLETE
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
62°F
5:19 PM
5/24/2025

Deployment Scripts

The `install_dependencies.sh` script installs Tomcat and Apache, sets up Apache as a reverse proxy to Tomcat, and configures routing so web traffic can reach the app. It ensures the server is ready to run the web app.

install_dependencies.sh will install Tomcat and Apache, then configure Apache as a reverse proxy to forward web traffic to the Tomcat server. This setup ensures the application is hosted and accessible via the web.

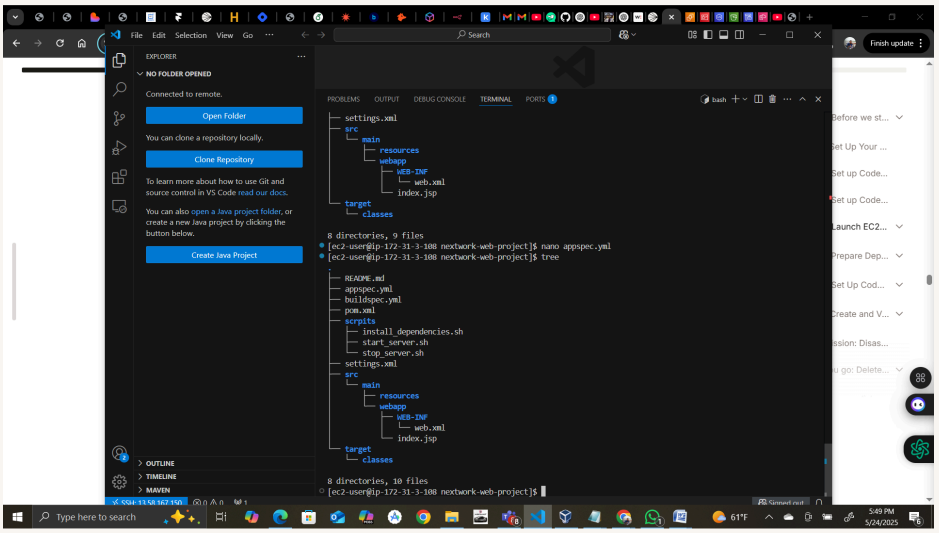
start_server.sh will start the Tomcat and Apache services and enable them to automatically restart on boot. This ensures our web app stays online even after an EC2 reboot, keeping the services always available.

I've saved stop_server.sh and confirmed that install_dependencies.sh, start_server.sh, and stop_server.sh are all inside the scripts folder. These scripts automate setup, startup, and shutdown of our app services.

appspec.yml

The appspec.yml is a deployment plan for CodeDeploy. It defines what to install, where to place files, and which scripts to run during each lifecycle event—ensuring a smooth, repeatable deployment process on Linux EC2.

buildspec.yml tells CodeBuild how to build and package our app. I updated the artifacts section to include appspec.yml and scripts so CodeDeploy knows how to install, start, and stop the app during deployment.

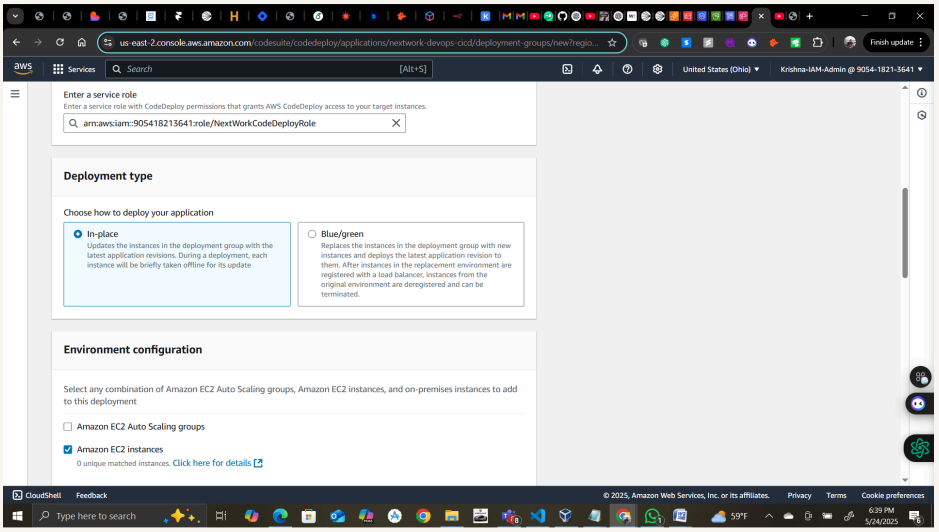


Setting Up CodeDeploy

A deployment group is a set of target instances and rules for deploying a specific version. A CodeDeploy application is the overall unit being deployed. One application can have many deployment groups.

To set up a deployment group, you also need to create an IAM role to give CodeDeploy the permissions it needs to access and manage your EC2 instance, run scripts, and deploy your application securely.

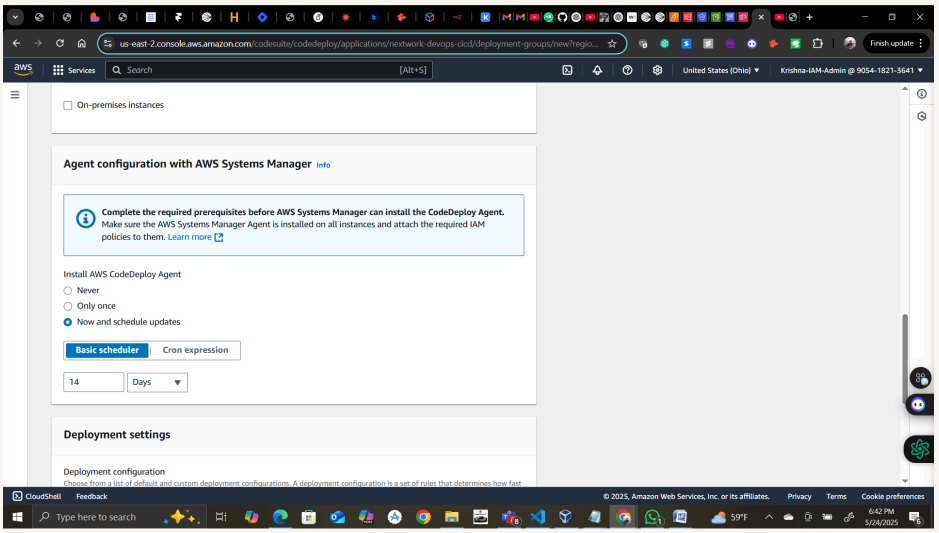
Tags are helpful for identifying and organizing AWS resources. I used the tag `CodeDeployInstance` to let CodeDeploy target the correct EC2 instance for deployment without hardcoding instance IDs.


Enter a service role
Enter a service role with CodeDeploy permissions that grants AWS CodeDeploy access to your target instances.
arn:aws:iam::905418213641: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.
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
0 unique matched instances. Click here for details
CloudShell
Feedback
© 2025, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

Deployment configurations

Another key setting is the deployment configuration, which affects how traffic is shifted during deployment. I used CodeDeployDefault.AllAtOnce, so all instances are updated at once, speeding deployment but risking full downtime if issues occur.

In order to connect EC2 with CodeDeploy, a CodeDeploy Agent is also set up to listen for deployment commands. It runs on the EC2 instance and executes instructions from the appspec.yml file.

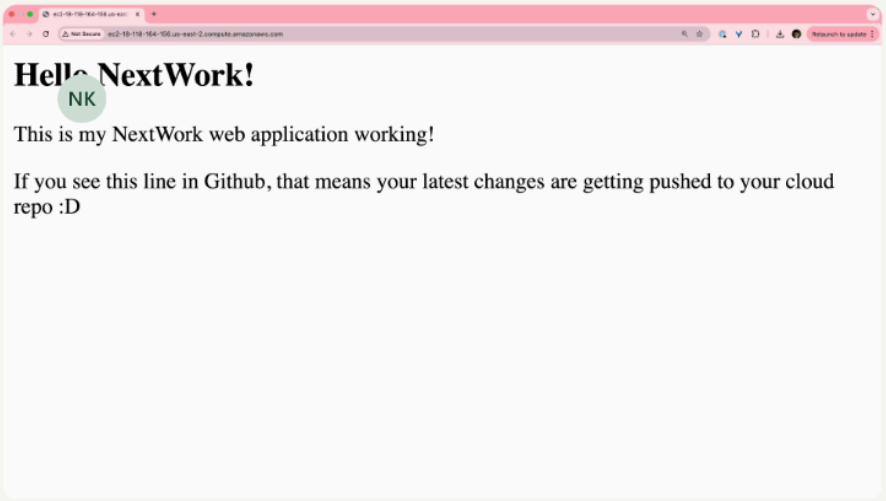


Success!

A CodeDeploy deployment is the actual process of delivering an application version to instances. The difference to a deployment group is that the group defines where and how to deploy, while the deployment is the execution of that setup.

I had to configure a revision location, which means specifying where CodeDeploy finds my app files for deployment. My revision location was the S3 bucket storing the build artifact zip with my app and scripts.

To check that the deployment was a success, I visited the web app URL. I saw the live application running without errors, confirming CodeDeploy completed and deployed correctly.


Hello NextWork!
NK
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