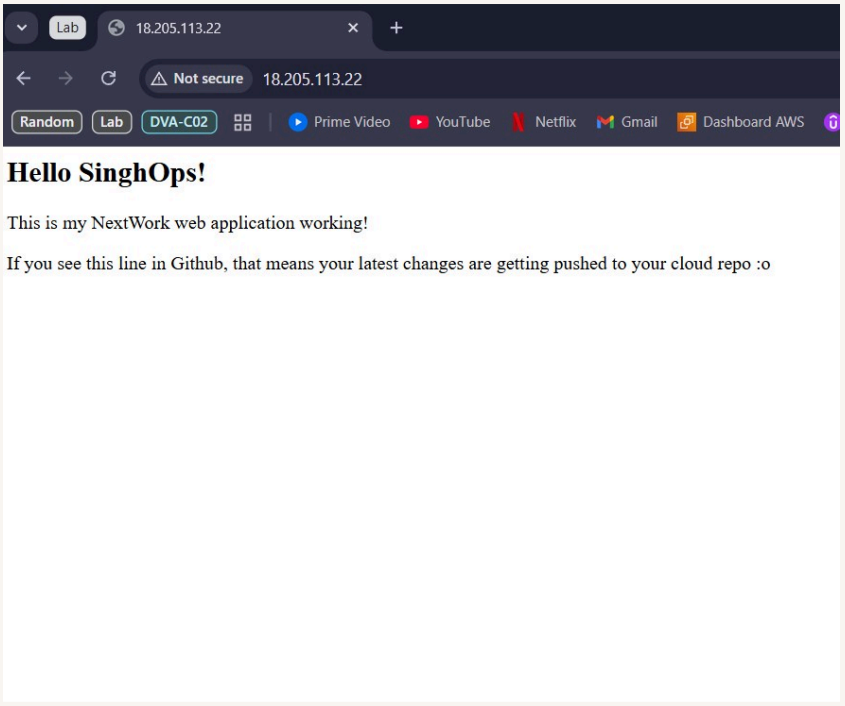
nextwork.org

Deploy a Web App with CodeDeploy

Gursimran Singh



Gursimran Singh
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to deploy a web app using CodeDeploy and set up infrastructure with CloudFormation. I'm doing this project to learn automation, streamline deployment, and implement rollback strategies for disaster recovery.

Key tools and concepts

Services I used were AWS CloudFormation, EC2, IAM, S3, and CodeDeploy. Key concepts I learnt include Infrastructure as Code, writing and using deployment scripts, using `appspec.yml`, and managing deployments with rollback capabilities.

Project reflection

This project took me approximately 2 hours to complete. The most challenging part was configuring the `appspec.yml` and deployment scripts correctly. It was most rewarding to see my web app successfully deployed and live through CodeDeploy.

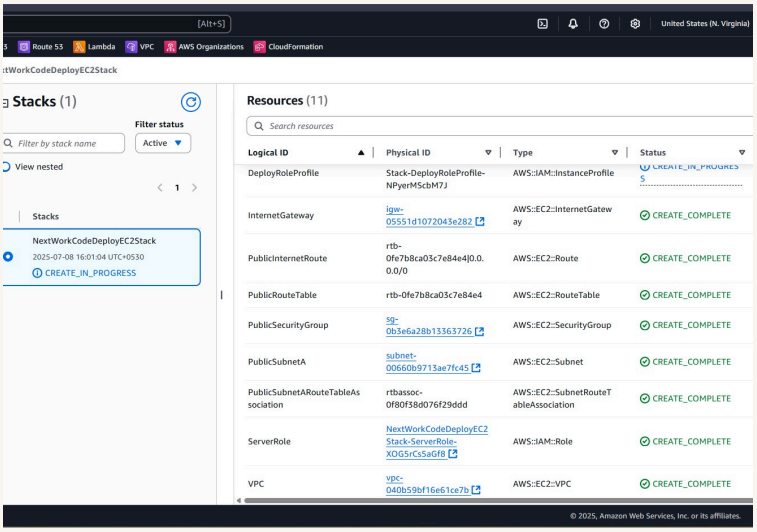
This project is part five of a series of DevOps projects where I'm building a CI/CD pipeline! I'll be working on the next project tomorrow to continue strengthening my deployment automation skills and learn new AWS services.

Deployment Environment

To set up for CodeDeploy, I launched an EC2 instance and VPC because they provide the environment where my web app will run. The EC2 instance hosts the app, while the VPC manages network access securely and reliably during deployments.

Instead of launching these resources manually, I used AWS CloudFormation, which automates the creation of EC2 and VPC using a template. When I need to delete these resources, I can simply delete the stack and everything is cleaned up automatically.

Other resources created in this template include a VPC, subnet, internet gateway, and route table. They're also in the template because they provide the networking setup needed for the EC2 instance to communicate with the internet and other services.


[Alt+S]
United States (N. Virginia)
Route 53
Lambda
VPC
AWS Organizations
CloudFormation
tWorkCodeDeployEC2Stack
Stacks (1)
Filter by stack name
Active
View nested
1
Stacks
NextWorkCodeDeployEC2Stack
2025-07-08 16:01:04 UTC+0530
CREATE_IN_PROGRESS
Resources (11)
Search resources
Logical ID
Physical ID
Type
Status
DeployRoleProfile
Stack-DeployRoleProfile-NPyerMScbM7J
AWS::IAM::InstanceProfile
InternetGateway
igw-05551d1072043e282
AWS::EC2::InternetGateway
CREATE_COMPLETE
PublicInternetRoute
rtb-0fe7b8ca03c7e84e4|0.0.0.0/0
AWS::EC2::Route
CREATE_COMPLETE
PublicRouteTable
rtb-0fe7b8ca03c7e84e4
AWS::EC2::RouteTable
CREATE_COMPLETE
PublicSecurityGroup
sg-0b3e6a28b13363726
AWS::EC2::SecurityGroup
CREATE_COMPLETE
PublicSubnetA
subnet-00660b9713ae7fc45
AWS::EC2::Subnet
CREATE_COMPLETE
PublicSubnetARouteTableAssociation
rtbassoc-0f80f38d076f29ddd
AWS::EC2::SubnetRouteTableAssociation
CREATE_COMPLETE
ServerRole
NextWorkCodeDeployEC2Stack-ServerRole-XOG5rCs5aGf8
AWS::IAM::Role
CREATE_COMPLETE
VPC
vpc-040b59bf16e61ce7b
AWS::EC2::VPC
CREATE_COMPLETE
© 2025, Amazon Web Services, Inc. or its affiliates.

Deployment Scripts

Scripts are mini-programs that automate tasks. To set up CodeDeploy, I also wrote scripts to stop the current app, install the new version, and start the app again—making the deployment process repeatable, fast, and error-free.

`install_dependencies.sh` will install all necessary software packages the app needs to run—like web servers, runtime environments, or libraries—so that the EC2 instance is fully prepared to host the application.

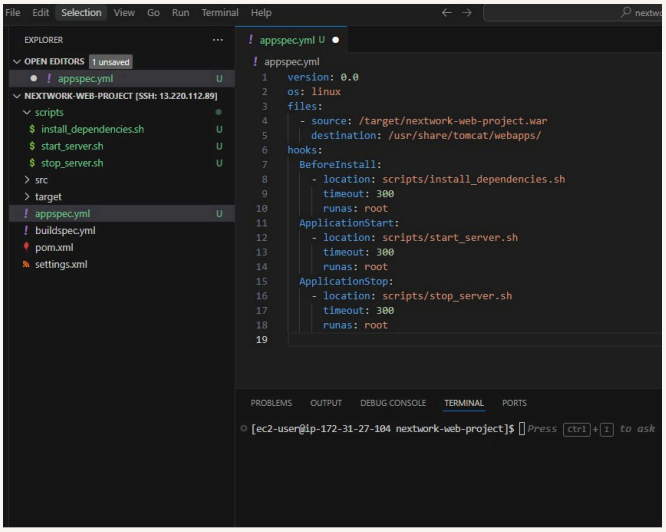
`start_server.sh` will start and enable both the Tomcat and Apache (httpd) services on the EC2 instance, ensuring the web server and application server are running and will automatically start on reboot.

`stop_server.sh` will stop both the Apache (httpd) and Tomcat services if they are currently running. It checks if each service is active using `pgrep` and stops them gracefully using `systemctl`.

appspec.yml

Then, I wrote an `appspec.yml` file to define how CodeDeploy should handle deployment. The key sections are `files` to move the WAR file, and `hooks` to run scripts before install, to stop the server, and after install, to start it again.

I also updated `buildspec.yml` because I needed to include the WAR file, `appspec.yml`, and deployment scripts in the output artifacts. This ensures CodeDeploy gets all necessary files to run the deployment properly on the EC2 instance.

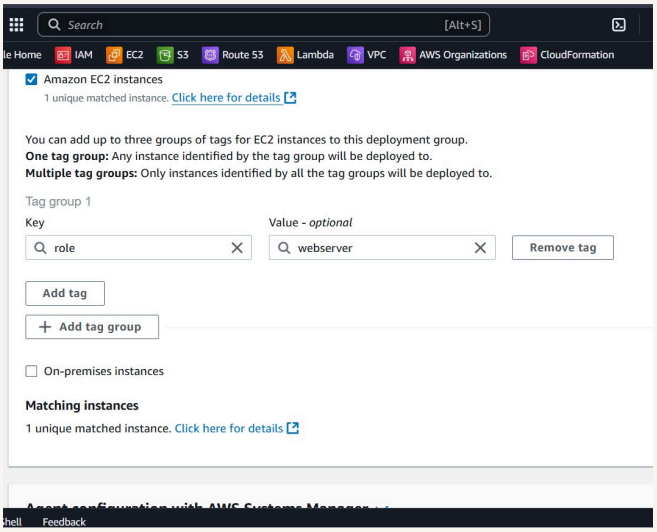


Setting Up CodeDeploy

A deployment group is a set of tagged EC2 instances where the app will be deployed, along with settings like rollback rules. A CodeDeploy application is the overall container that manages and tracks the deployment process for a specific app.

To set up a deployment group, you also need to create an IAM role to give CodeDeploy permission to access and manage EC2 instances during deployments. This allows it to run scripts, copy files, and monitor the deployment process securely.

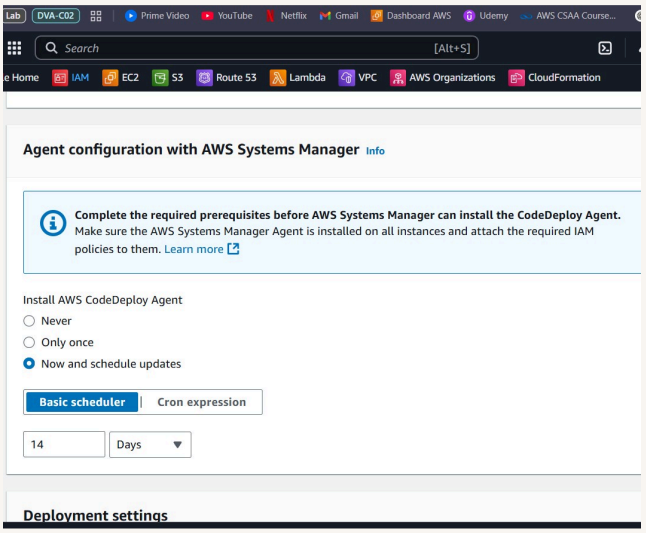
Tags are helpful for identifying and targeting specific EC2 instances in a scalable way. I used the tag Role = webserver to ensure CodeDeploy deploys only to instances meant to run the application, without needing to reference instance IDs directly.


Search
[Alt+S]
le Home
IAM
EC2
S3
Route 53
Lambda
VPC
AWS Organizations
CloudFormation
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
hell
Feedback

Deployment configurations

Another key setting is the deployment configuration, which affects how updates are rolled out. I used CodeDeployDefault.AllAtOnce, so all EC2 instances in the group were updated simultaneously, making the deployment fast but with more risk.

In order to connect the EC2 instance with CodeDeploy, a CodeDeploy Agent is also set up to listen for deployment instructions, pull files from S3, run the lifecycle scripts from appspec.yml, and manage the deployment steps on the instance.

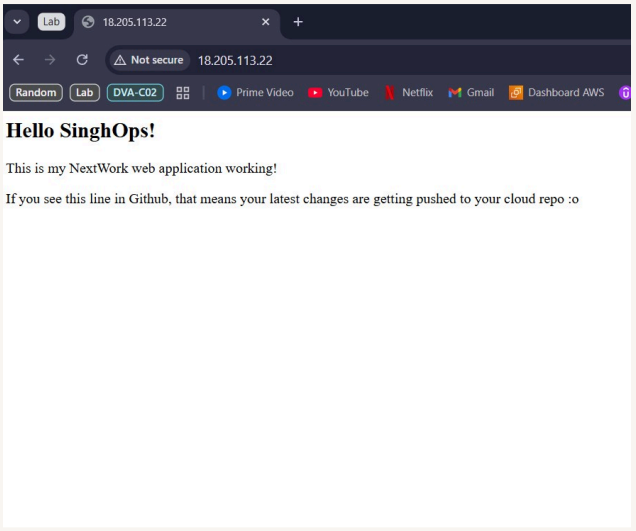


Success!

A CodeDeploy deployment is the process of delivering updated code and resources to target instances. The difference to a deployment group is that the group defines the set of instances and settings, while the deployment executes the actual update.

I had to configure a revision location, which means specifying where CodeDeploy should fetch the deployment package. My revision location was an S3 bucket containing a ZIP file with the WAR file, scripts, and appspec.yml.

To check that the deployment was a success, I visited the public IP of the EC2 instance created by CloudFormation. I saw the deployed web application running, which confirmed that CodeDeploy successfully completed the deployment.


Lab
18.205.113.22
Not secure
18.205.113.22
Random
Lab
DVA-C02
Prime Video
YouTube
Netflix
Gmail
Dashboard AWS
Hello SinghOps!
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 :o

The place to learn & showcase your skills

Check out nextwork.org for more projects