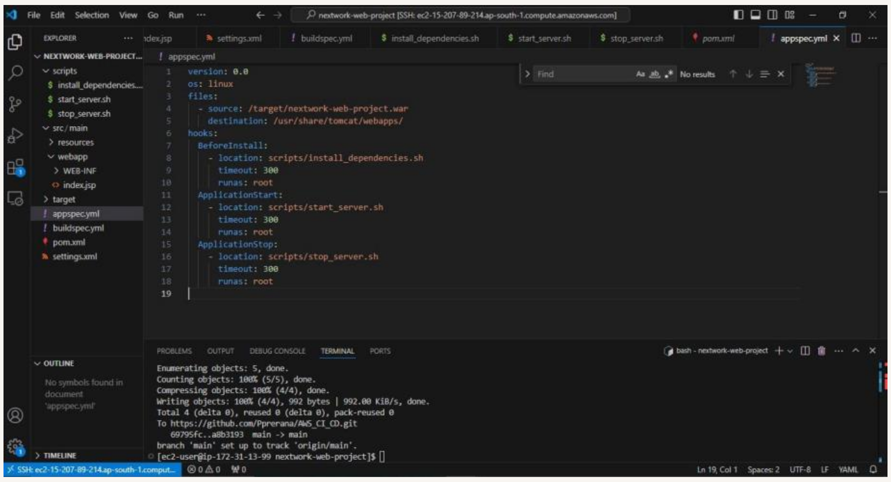
Deploy an App with CodeDeploy



```yaml
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
```

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 992 bytes | 992.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Pprerana/AWS_CI_CD.git
   69795fc..a8b3193  main -> main
branch 'main' set up to track 'origin/main'.
[ec2-user@ip-172-31-13-99 nextwork-web-project]$
```

Introducing today's project!

What is AWS CodeDeploy?

AWS CodeDeploy is a fully managed service that automates application deployment to EC2 instances, Lambda, or on-prem servers. It ensures consistent, scalable, and reliable deployments, reduces manual errors, and enables rollback in case of failure, enhancing deployment efficiency.

How I'm using AWS CodeDeploy in this project

In today's project, I used AWS CodeDeploy to automate the deployment of an application to EC2 instances. I set up a CodeDeploy application and deployment group, created an IAM role with the necessary permissions, and specified the revision location in S3 to deploy the app seamlessly.

This project took me...

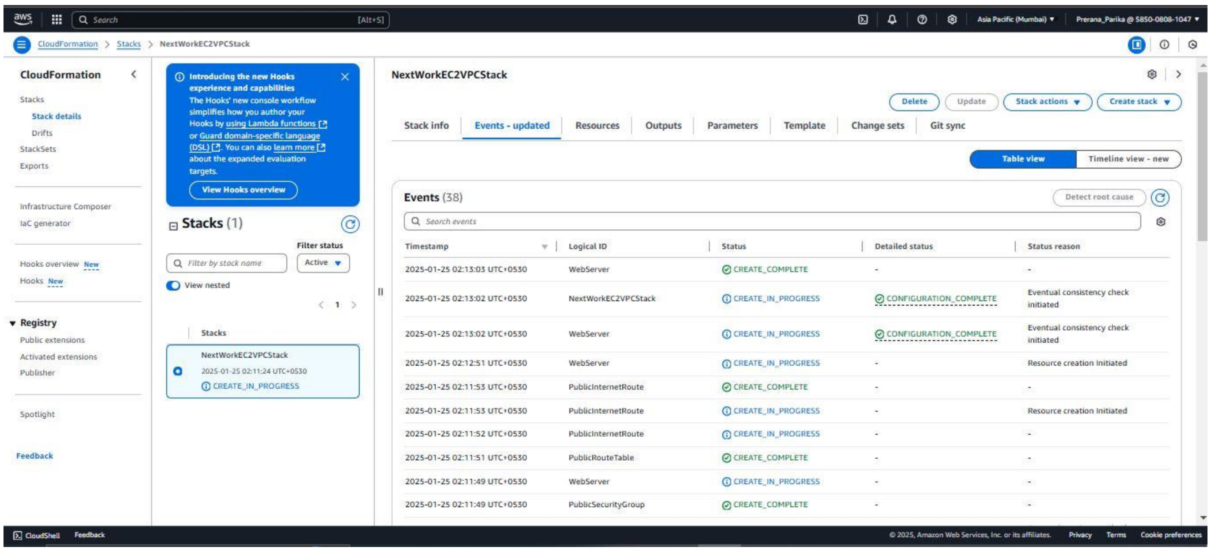
It took me around an hour

Set Up an EC2 Instance

I set up an EC2 instance and VPC because it provides a secure and scalable environment for deploying applications. The EC2 instance allows for computing power, while the VPC ensures isolated network configuration and control over resources within AWS.

We manage production and development environments separately because it ensures stability and security in production while allowing for experimentation and testing in development. This separation helps prevent disruptions and ensures that changes are thoroughly tested before deployment.

To set up my EC2 instance and VPC, I used AWS CloudFormation. This service allows me to define the infrastructure as code, ensuring consistent, repeatable deployments of both the EC2 instance and VPC with minimal manual configuration.



NextWorkEC2VPCStack

Events (38)

Timestamp	Logical ID	Status	Detailed status	Status reason
2025-01-25 02:13:03 UTC+0530	WebServer	CREATE_COMPLETE	-	-
2025-01-25 02:13:02 UTC+0530	NextWorkEC2VPCStack	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Eventual consistency check initiated
2025-01-25 02:13:02 UTC+0530	WebServer	CREATE_IN_PROGRESS	CONFIGURATION_COMPLETE	Eventual consistency check initiated
2025-01-25 02:12:51 UTC+0530	WebServer	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-01-25 02:11:53 UTC+0530	PublicInternetRoute	CREATE_COMPLETE	-	-
2025-01-25 02:11:53 UTC+0530	PublicInternetRoute	CREATE_IN_PROGRESS	-	Resource creation Initiated
2025-01-25 02:11:52 UTC+0530	PublicInternetRoute	CREATE_IN_PROGRESS	-	-
2025-01-25 02:11:51 UTC+0530	PublicRouteTable	CREATE_COMPLETE	-	-
2025-01-25 02:11:49 UTC+0530	WebServer	CREATE_IN_PROGRESS	-	-
2025-01-25 02:11:49 UTC+0530	PublicSecurityGroup	CREATE_COMPLETE	-	-

Bash scripts

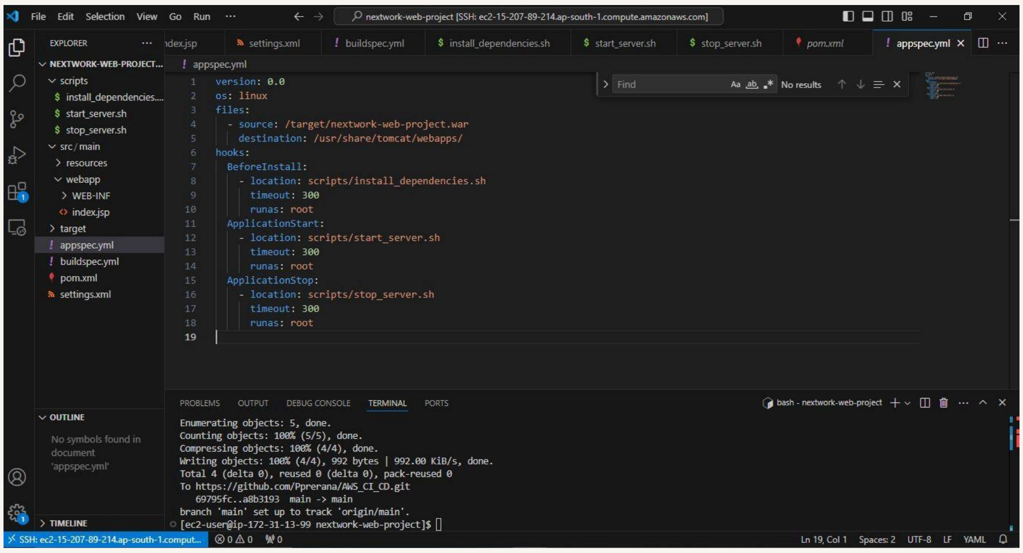
Scripts are files that contain a series of commands or instructions written in a programming or scripting language, such as Bash. Bash is a command-line interface and scripting language used in Unix-like operating systems to automate tasks and manage system operations.

I used three scripts for my project's deployment

The first script I created was to install and configure Tomcat and Apache HTTP server. It installs Tomcat and HTTPD, then creates a custom Apache configuration file (`tomcat_manager.conf`) to set up a reverse proxy to Tomcat on port 8080 for the Nextwork web project.

The second script I created was to start and enable the Tomcat and Apache HTTPD services. It ensures that both services are started and automatically begin running on system boot by using `systemctl` to manage the services.

The third script I created was to stop the Apache HTTPD and Tomcat services if they are running. It checks if each service is active using pgrep, and if found, it stops the corresponding service with systemctl. This helps in safely shutting down the servers before making changes.


File Edit Selection View Go Run
nextwork-web-project [SSH: ec2-15-207-89-214.ap-south-1.compute.amazonaws.com]
EXPLORER
NEXTWORK-WEB-PROJECT...
scripts
install_dependencies....
start_server.sh
stop_server.sh
src / main
resources
webapp
WEB-INF
index.jsp
target
appspec.yml
buildspec.yml
pom.xml
settings.xml
OUTLINE
No symbols found in document 'appspec.yml'
TIMELINE
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS
bash - nextwork-web-project
SSH: ec2-15-207-89-214.ap-south-1.comput...
Ln 19, Col 1 Spaces: 2 UTF-8 LF YAML

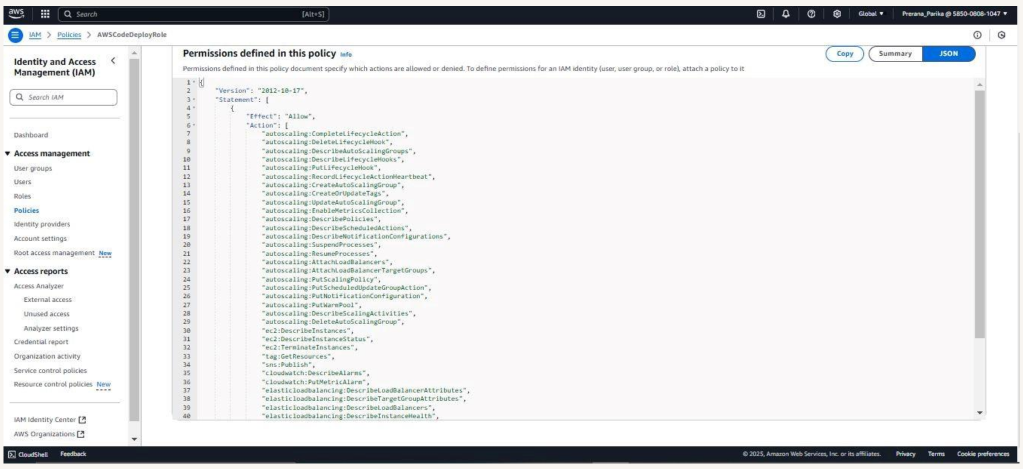
```yaml
version: 0.0
os: linux
files:
  - source: /target/nextwork-web-project.war
    destination: /usr/share/tomcat/webapps/
hooks:
  BeforeInstall:
    - location: scripts/install_dependencies.sh
      timeout: 300
      runas: root
  ApplicationStart:
    - location: scripts/start_server.sh
      timeout: 300
      runas: root
  ApplicationStop:
    - location: scripts/stop_server.sh
      timeout: 300
      runas: root
```

```
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Compressing objects: 100% (4/4), done.
Writing objects: 100% (4/4), 992 bytes | 992.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/Pprerana/AWS_CI_CD.git
   69795fc..a8b3193  main -> main
branch 'main' set up to track 'origin/main'.
[ec2-user@ip-172-31-13-99 nextwork-web-project]$
```

CodeDeploy’s IAM Role

I created an IAM service role for CodeDeploy because CodeDeploy needs permission to interact with EC2 instances to deploy applications. The role, using the AWS Managed AWSCodeDeployRole policy, grants the necessary access for seamless deployment to EC2

To set up CodeDeploy’s IAM role, I created a new IAM role in the IAM console and selected the "CodeDeploy" service as the trusted entity. Then, I attached the AWS managed policy AWSCodeDeployRole to the role to grant the necessary permissions for deployment to EC2 instances.


IAM > Policies > AWSCodeDeployRole
Identity and Access Management (IAM)
Copy
Summary
JSON

Permissions defined in this policy

Permissions defined in this policy document specify which actions are allowed or denied. To define permissions for an IAM identity (user, user group, or role), attach a policy to it

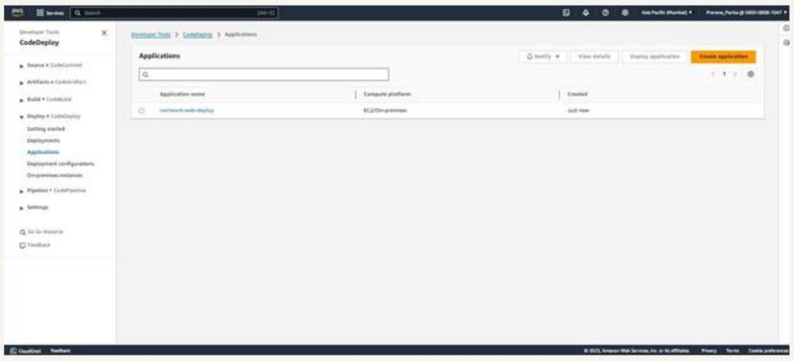
```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Action": [
                "autoscaling:CompleteLifecycleAction",
                "autoscaling:DeleteLifecycleHook",
                "autoscaling:DescribeAutoScalingGroups",
                "autoscaling:DescribeLifecycleHooks",
                "autoscaling:PutLifecycleHook",
                "autoscaling:RecordLifecycleActionHeartbeat",
                "autoscaling:CreateAutoScalingGroup",
                "autoscaling:CreateOrUpdateTags",
                "autoscaling:UpdateAutoScalingGroup",
                "autoscaling:EnableMetricsCollection",
                "autoscaling:DescribePolicies",
                "autoscaling:DescribeScheduledActions",
                "autoscaling:DescribeNotificationConfigurations",
                "autoscaling:SuspendProcesses",
                "autoscaling:ResumeProcesses",
                "autoscaling:AttachLoadBalancers",
                "autoscaling:AttachLoadBalancerTargetGroups",
                "autoscaling:PutScalingPolicy",
                "autoscaling:PutScheduledUpdateGroupAction",
                "autoscaling:PutNotificationConfiguration",
                "autoscaling:PutWarmPool",
                "autoscaling:DescribeScalingActivities",
                "autoscaling:DeleteAutoScalingGroup",
                "ec2:DescribeInstances",
                "ec2:DescribeInstanceStatus",
                "ec2:TerminateInstances",
                "tag:GetResources",
                "sns:Publish",
                "cloudwatch:DescribeAlarms",
                "cloudwatch:PutMetricAlarm",
                "elasticloadbalancing:DescribeLoadBalancerAttributes",
                "elasticloadbalancing:DescribeTargetGroupAttributes",
                "elasticloadbalancing:DescribeLoadBalancers",
                "elasticloadbalancing:DescribeInstanceHealth",
```

CodeDeploy application

A CodeDeploy application means a logical entity within AWS CodeDeploy that represents the deployment of code to one or more compute resources, such as EC2 instances or Lambda functions. It defines the deployment settings, configurations, and target environments for the application code.

To create a CodeDeploy application, I had to select a compute platform, which means choosing the type of resource where the application will be deployed. This could be EC2 instances, on-premises servers, or AWS Lambda functions, depending on the application's deployment target.

The compute platform I chose was **EC2** because I am deploying the application to EC2 instances, which provides the necessary compute resources and scalability for the application. EC2 instances offer flexibility and control over the environment for running the application.


Developer Tools > CodeDeploy > Applications

Applications

	Application name	Compute platform	Created
○	nextwork-web-deploy	EC2/On-premises	Just now

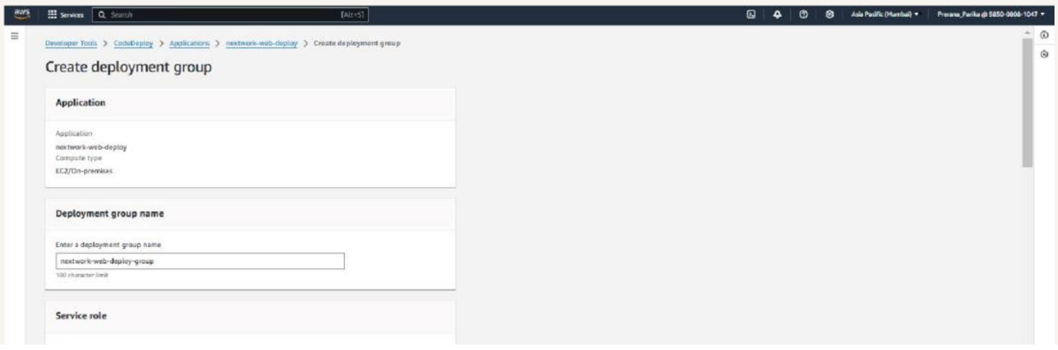
Deployment group

A deployment group means a set of EC2 instances or on-premises servers that CodeDeploy targets for deployment. It allows you to organize instances based on tags, Auto Scaling groups, or manually defined groups, ensuring that the correct instances receive the application updates.

Two key configurations for a deployment group

Environment means the specific settings and configurations that define the target deployment infrastructure for an application, such as the compute platform, deployment group, and other resources. It ensures the proper setup for where and how the application will be deployed.

A CodeDeploy Agent is a software component installed on the target EC2 instances or on-premises servers. It communicates with the CodeDeploy service, enabling the


Developer Tools > CodeDeploy > Applications > nextwork-web-deploy > Create deployment group

Create deployment group

Application

Application
nextwork-web-deploy
Compute type
EC2/On-premises

Deployment group name

Enter a deployment group name
nextwork-web-deploy-group
100 character limit

Service role

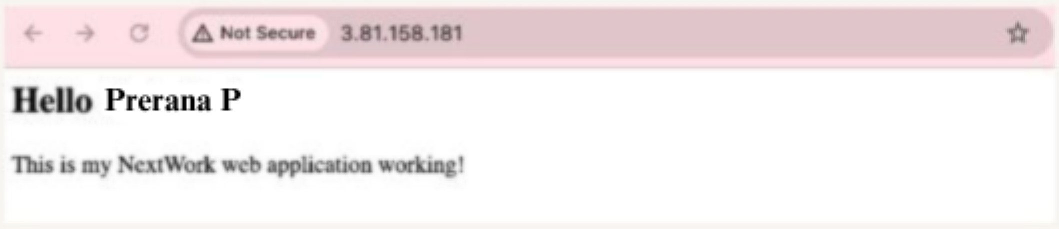
deployment of application revisions to those instances. The agent ensures the proper installation and configuration of the application during deployments.

CodeDeploy application

To create my deployment, I had to set up a revision location, which means specifying the location of the application's source code or deployment package. This could be a file stored in an Amazon S3 bucket or a GitHub repository that CodeDeploy will use to deploy the application to the target instances.

My revision location was my **Amazon S3 bucket** where I uploaded the application's deployment package. The S3 URI pointed to the specific location of the zipped application code or artifact that CodeDeploy would retrieve and deploy to the target EC2 instances.

To visit my web app, I had to visit the **public IP address or DNS name** of the EC2 instance that was running the application. This address, combined with the appropriate port (e.g., 80 for HTTP), provided the link to access the deployed web app.



Hello Prerana P

This is my NextWork web application working!