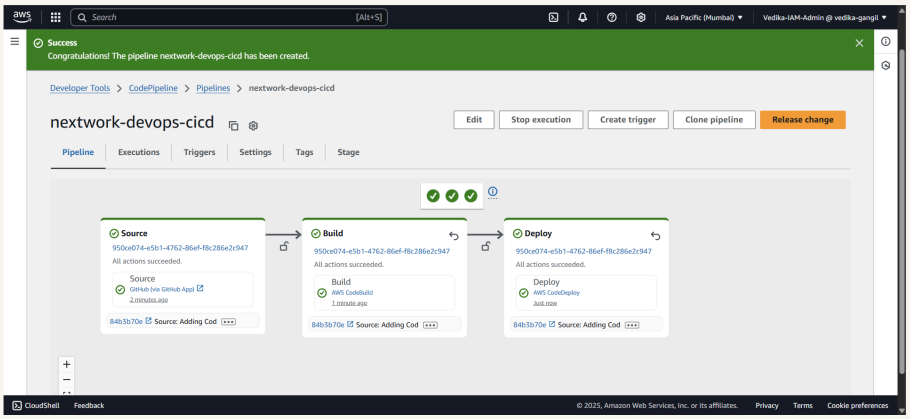
nextwork.org

Build a CI/CD Pipeline with AWS

VE vedikagangil@gmail.com



Introducing Today's Project!

Today, I'll learn how to use AWS CodePipeline to automate my entire CI/CD workflow into one seamless process! This project will bring together everything I've learned in the DevOps Challenge.

Key tools and concepts

Services I used were AWS CodePipeline (orchestration), CodeBuild (compilation), CodeDeploy (EC2 deployments), S3 (artifact storage), CloudWatch (logs), IAM (security roles), and GitHub (source control).

Project reflection

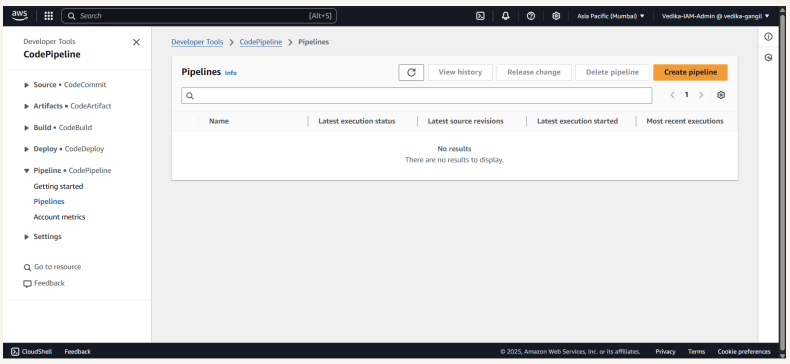
The project took me about 2 hours to complete from start to finish.

Starting a CI/CD Pipeline

AWS CodePipeline is a fully managed CI/CD service that automates your release pipeline—from code changes to builds (CodeBuild), tests, and deployments. It orchestrates stages (source, build, deploy) with manual approvals if needed.

CodePipeline offers different execution modes based on how you want to trigger and process pipeline stages. I chose Superseded mode (automatically cancels outdated in-progress runs when a new change is pushed), but there are other options as well.

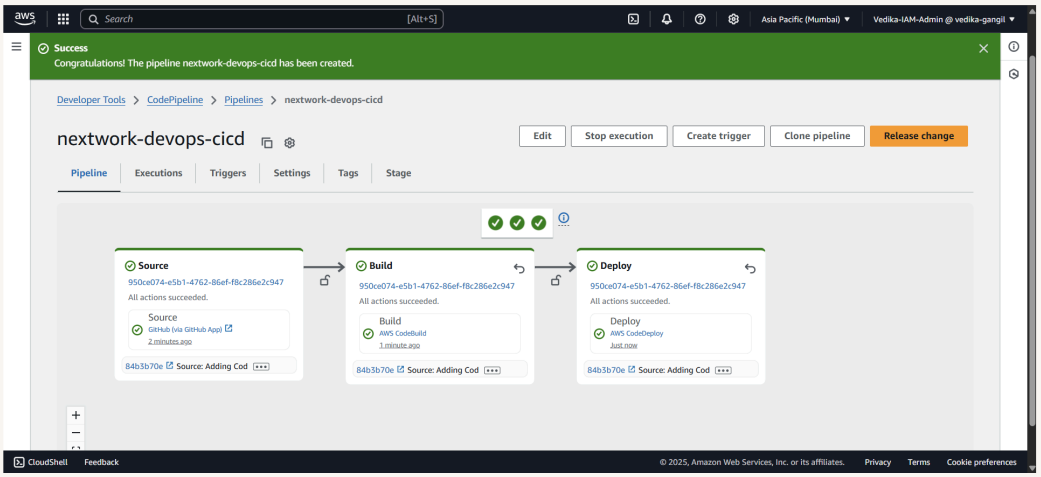
A service role gets created automatically during setup so CodePipeline can securely interact with other AWS services (e.g., pull code from S3, trigger CodeBuild, or deploy via CodeDeploy).



CI/CD Stages

The three stages I've set up in my CI/CD pipeline are Source (GitHub code changes), Build (CodeBuild compilation/testing), and Deploy (CodeDeploy to EC2).

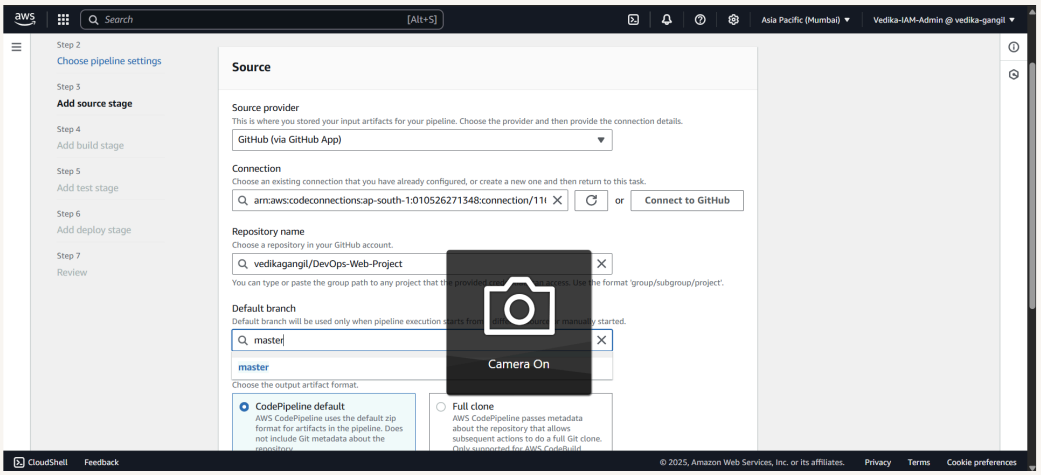
CodePipeline organizes the three stages into a visual workflow. In each stage, we can see more details on: Source: Commit ID, branch, trigger event., Build: CodeBuild logs, artifacts generated, duration, Deploy: CodeDeploy status(success/failure).



Source Stage

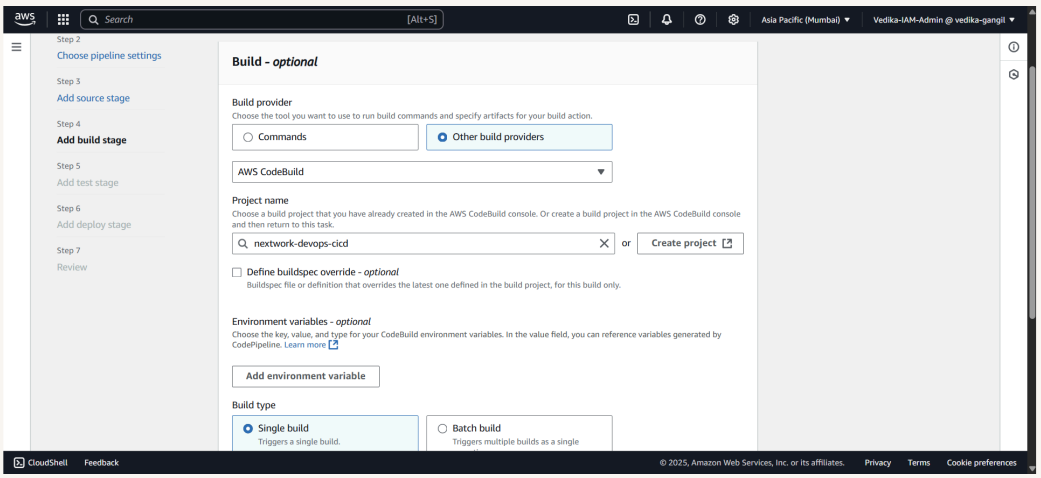
In the Source stage, the default branch tells CodePipeline which Git branch to monitor for code changes. This ensures automatic pipeline triggers only for commits to the designated branch, avoiding unnecessary builds from feature branches.

The source stage is also where you enable webhook events, which automatically trigger your pipeline (e.g., CodePipeline) whenever a commit or pull request is pushed to your GitHub/Bitbucket repo. This eliminates manual intervention.



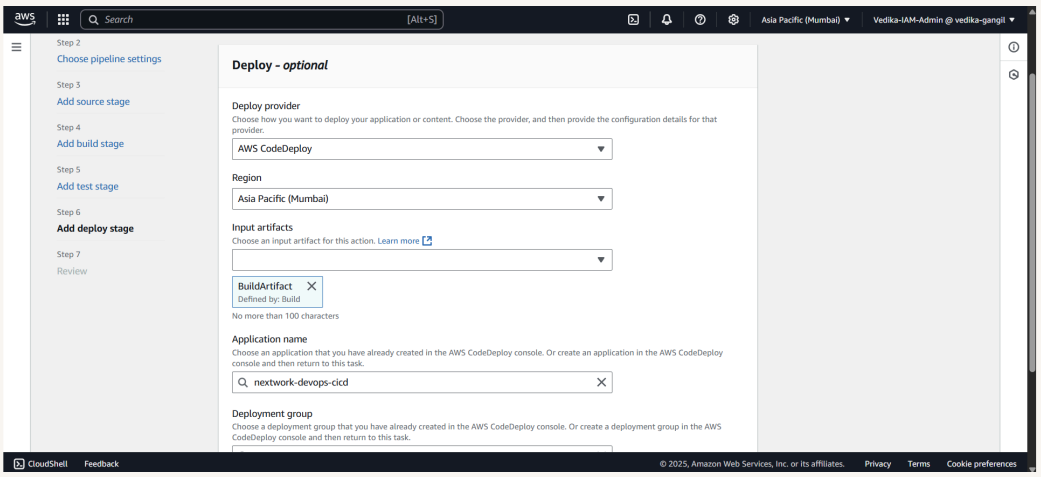
Build Stage

The Build stage sets up the transformation of raw source code into deployable artifacts. I configured CodeBuild to process the source files. The input artifact for the build stage is SourceArtifact.



Deploy Stage

The Deploy stage is where the built artifactsare installed on target infrastructure. I configured CodeDeploy to fetch artifacts from S3 (output of Build stage), follow appspec.yml to run hooks and roll back on failure.

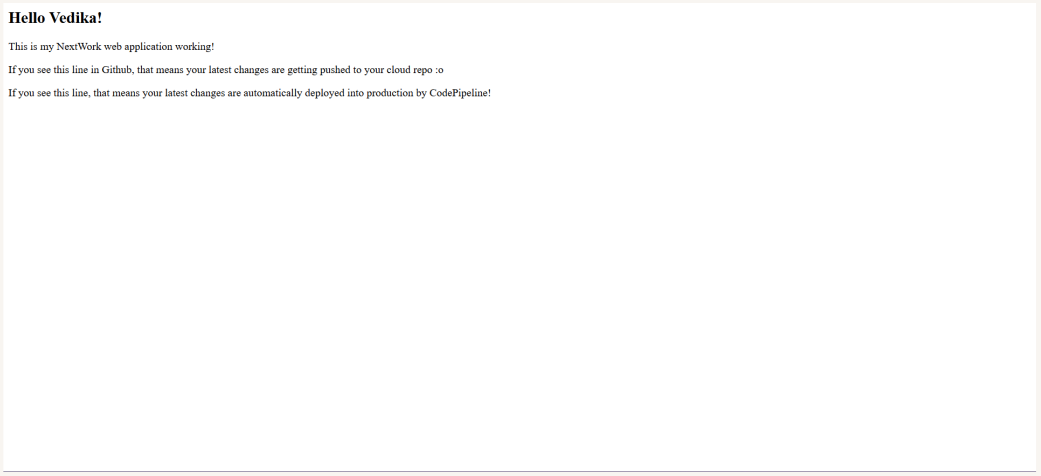


Success!

Since my CI/CD pipeline gets triggered by GitHub commits, I tested it by adding a new line to my index.jsp file). This change automatically kicked off the pipeline, proving the integration worked end-to-end.

The moment I pushed the code change, the pipeline's Source stage immediately detected the GitHub webhook trigger, pulling the latest commit. The commit message under each stage reflects the same Git commit ID ensuring traceability

Once my pipeline executed successfully, I checked the CodePipeline console to verify all stages (Source, Build, Deploy) showed "Succeeded" status. I then confirmed the EC2 instance was serving the updated index.jsp by viewing the live endpoint.


Hello Vedika!
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
If you see this line, that means your latest changes are automatically deployed into production by CodePipeline!

The place to learn & showcase your skills

Check out nextwork.org for more projects