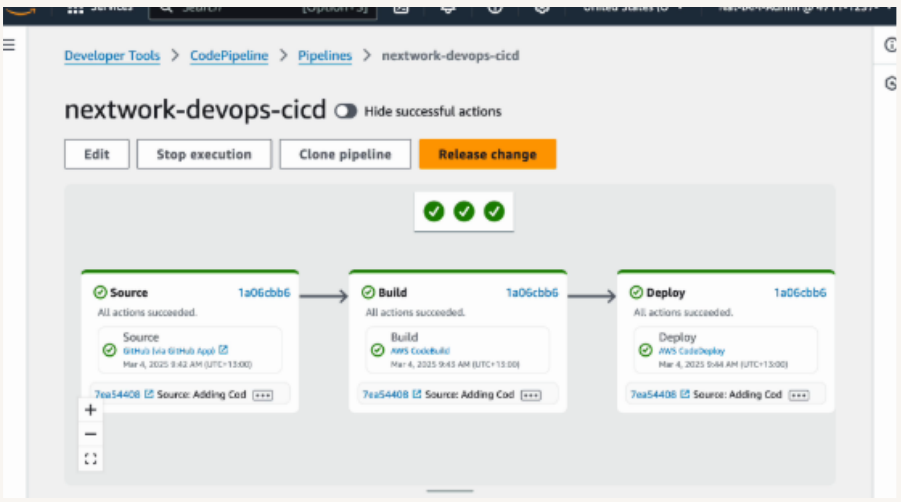
Build a CI/CD Pipeline with AWS

Seth Sekyere



Seth Sekyere

Introducing Today's Project!

In this project, I will demonstrate how to automate my CI/CD workflow using AWS CodePipeline, connecting GitHub, CodeBuild, and CodeDeploy. I'm doing this project to learn how to create fully automated, reliable, and seamless deployments.

Key tools and concepts

Services used: CodePipeline, CodeBuild, CodeDeploy, EC2, S3, IAM. Key concepts learnt: CI/CD pipelines, automated deployments, pipeline stages, webhooks, build artifacts, rollback strategies, and Infrastructure as Code.

Project reflection

This project took me approximately a few hours. The most challenging part was configuring permissions and connections correctly. It was most rewarding to see the CI/CD pipeline automatically build and deploy my web app successfully.

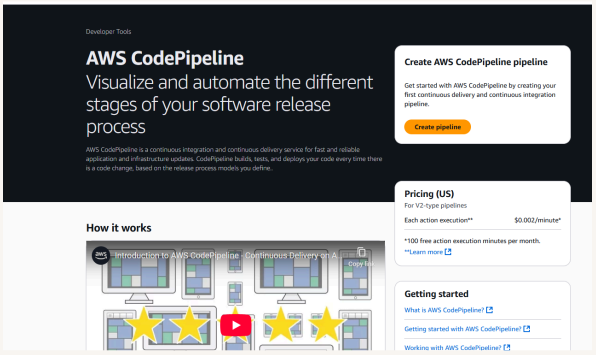
Seth Sekyere

Starting a CI/CD Pipeline

AWS CodePipeline is a fully managed service that automates the building, testing, and deployment of your code, creating a seamless CI/CD workflow from source to production.

CodePipeline offers different execution modes based on how overlapping runs are handled. I chose Superseded so new runs cancel in-progress ones. Other options include Queued (runs wait their turn) and Parallel (runs simultaneously).

A service role gets created automatically during setup so CodePipeline has permission to access AWS resources (like S3, CodeBuild, and CodeDeploy) on your behalf to run the pipeline smoothly.

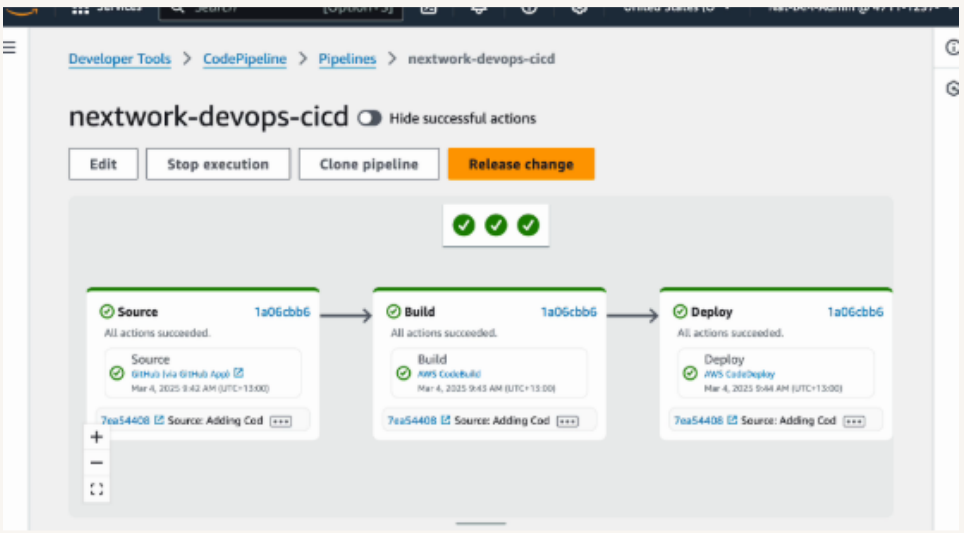


Seth Sekyere

CI/CD Stages

The three stages in my CI/CD pipeline are Source, Build, and Deploy. I learned how to fetch code from GitHub, build it with CodeBuild, and deploy it to EC2 using CodeDeploy for automated, reliable releases.

CodePipeline shows Source, Build, and Deploy stages. For each stage, you can see status, timing, logs, and artifacts, giving a clear view of progress and helping debug issues.

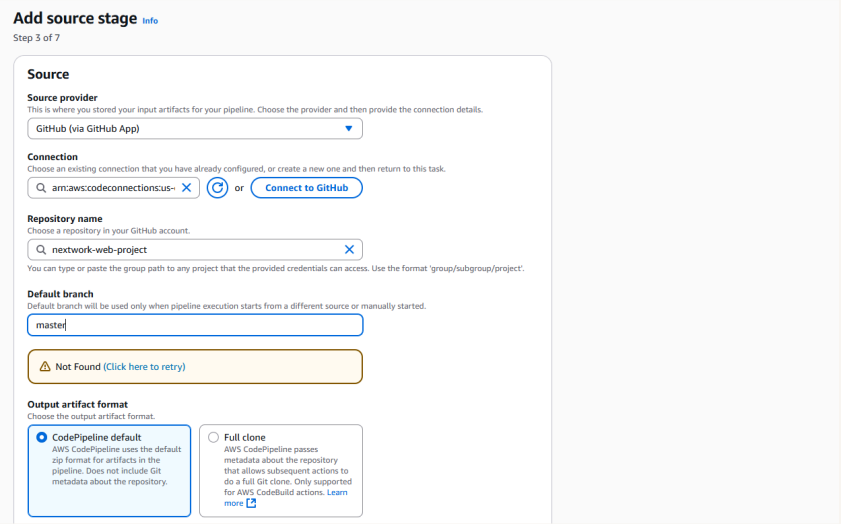


Seth Sekyere

Source Stage

In the Source stage, the default branch tells CodePipeline which branch to continuously monitor for changes. Whenever a commit is pushed to this branch, it automatically triggers the pipeline to start a new build and deployment.

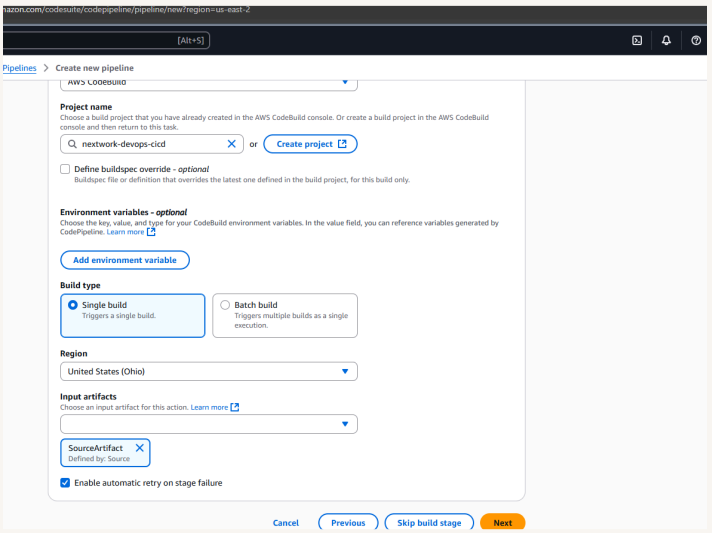
The source stage is also where you enable webhook events, which are important because they automatically trigger the pipeline whenever new code is pushed. This ensures deployments are fast, consistent, and fully continuous without manual steps.



Seth Sekyere

Build Stage

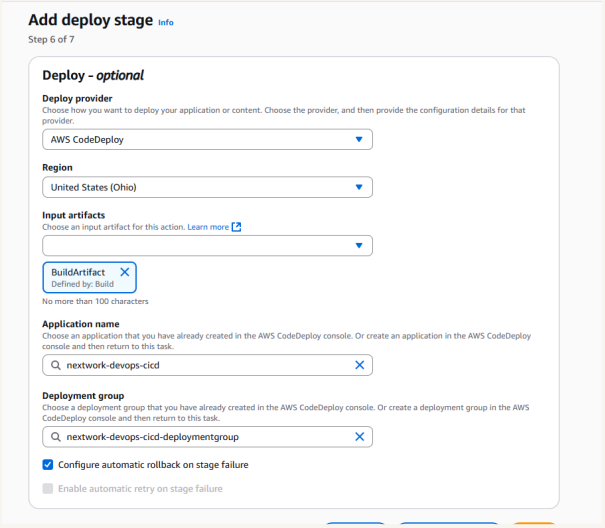
The Build stage sets up the compilation and packaging of the app. I configured it to take SourceArtifact from the Source stage as input, since that ZIP contains the latest code that needs to be built into deployable artifacts.



Seth Sekyere

Deploy Stage

The Deploy stage is where the build artifacts from CodeBuild are deployed to our EC2 instance. I configured it to use my existing CodeDeploy deployment group and enabled automatic rollback on stage failure.



Seth Sekyere

Success!

Since my CI/CD pipeline gets triggered by GitHub commits, I tested my pipeline by adding a new line to `index.jsp` and pushing it. This change verified that CodePipeline automatically built and deployed the updated web app.

The moment I pushed the code change, my pipeline automatically started a new execution. The commit message under each stage reflects the latest changes, showing that CodePipeline detected, built, and deployed my update without any manual steps.

Once my pipeline executed successfully, I checked the deployed web app in the browser and saw the new line I added to index.jsp, confirming that the latest changes were automatically built and deployed by CodePipeline.

Seth Sekyere

Hello NextWork!

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

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