

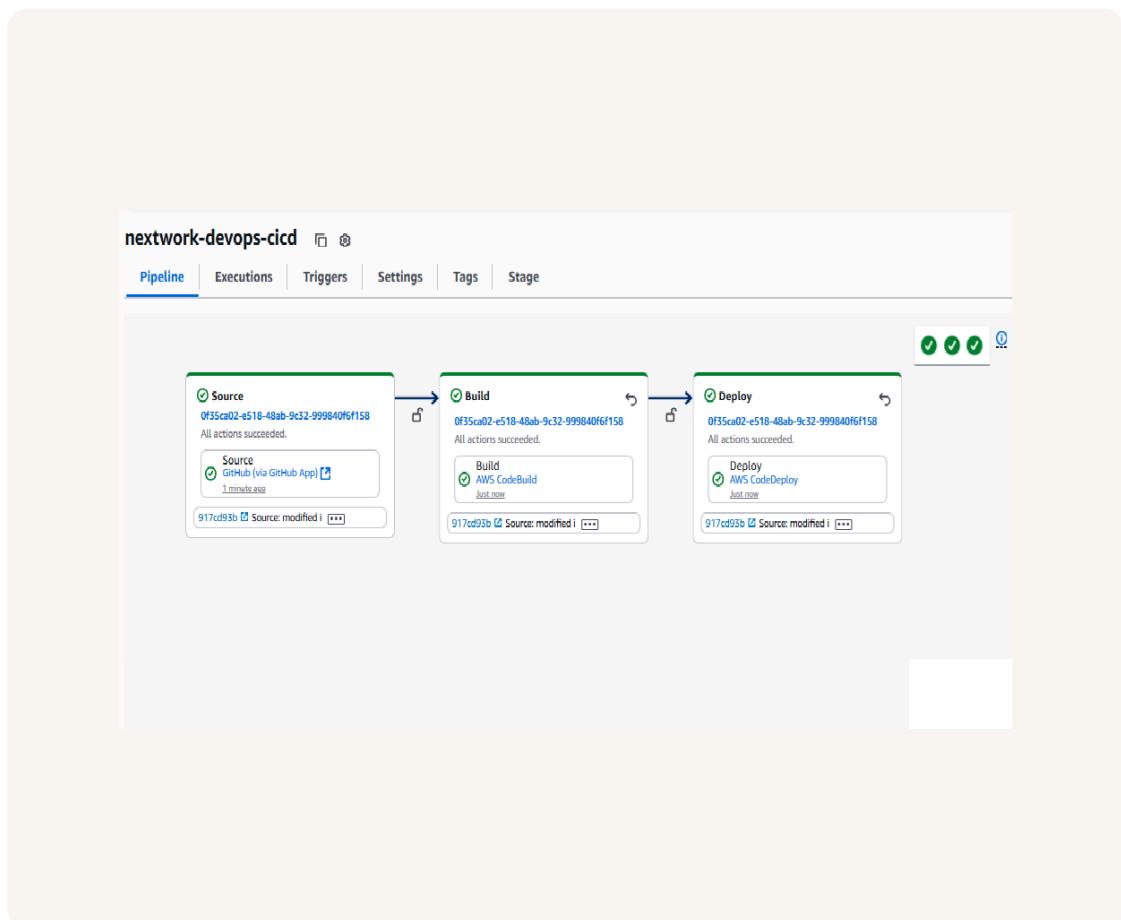


[nextwork.org](https://nextwork.org)

# Build a CI/CD Pipeline with AWS



Sanjana Tripathy





# Introducing Today's Project!

In this project, I will demonstrate how to build an automated CI/CD pipeline with AWS CodePipeline. I'm doing this project to learn how to seamlessly integrate source, build, and deploy stages for reliable, hands-free deployments.

## Key tools and concepts

Services I used were AWS CodePipeline, CodeBuild, CodeDeploy, Amazon S3, Amazon EC2, AWS IAM, GitHub, and CloudFormation. Key concepts I learnt include CI/CD, pipeline automation, deployment scripts, appspec.yml, CloudFormation stacks, and IAAC.

## Project reflection

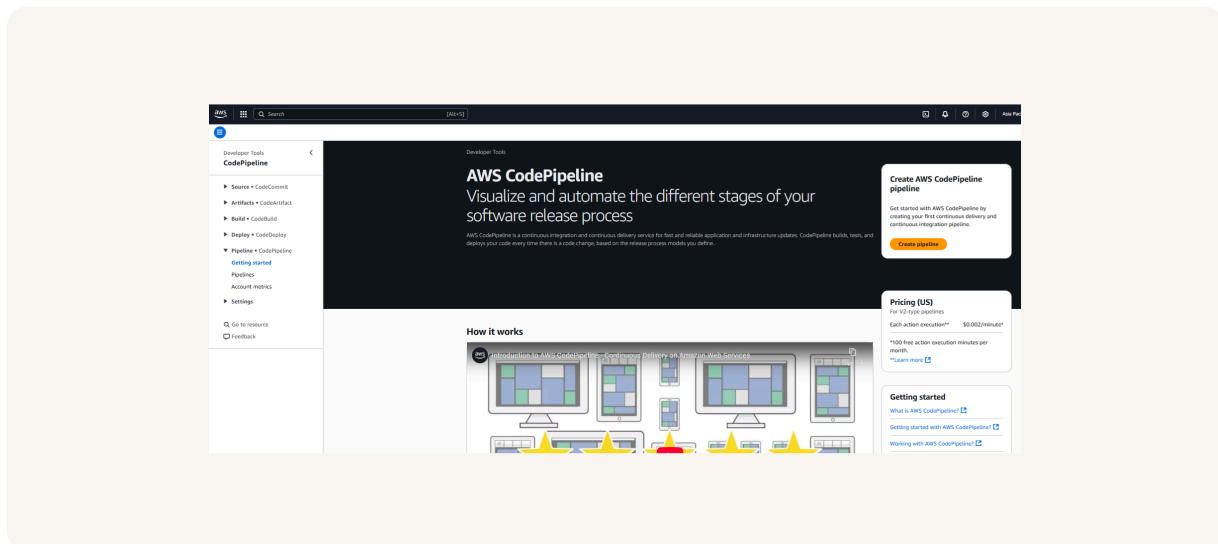
This project took me approximately 1 hour. The most challenging part was nothing as every step was enriching. It was most rewarding to see the pipeline in action.

# Starting a CI/CD Pipeline

AWS CodePipeline is a fully managed CI/CD service that automates the steps of releasing applications, from source to build to deployment, ensuring faster, reliable, and consistent software delivery.

CodePipeline offers different execution modes based on how concurrent runs are handled. I chose Superseded mode, where new runs cancel older ones, ensuring only the latest code is deployed. Other options include Queued (wait in line) and Parallel.

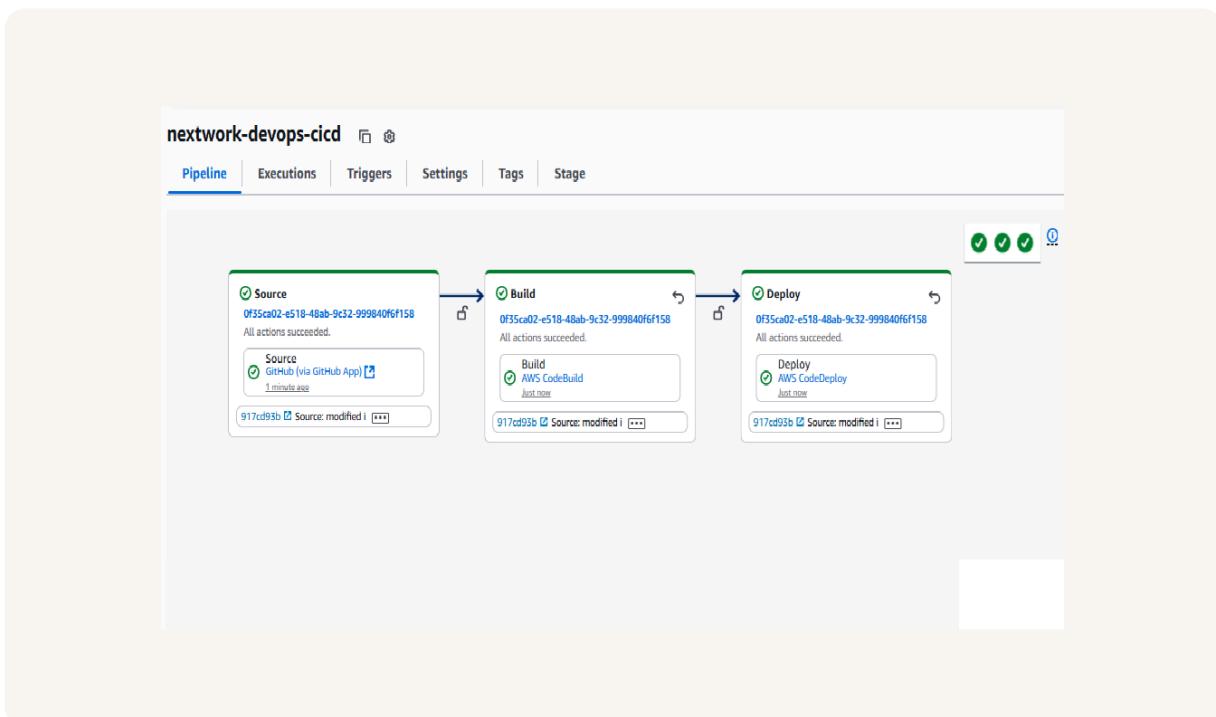
A service role gets created automatically during setup so CodePipeline has the permissions it needs to access and manage resources like S3, CodeBuild, and CodeDeploy on my behalf, ensuring the pipeline runs smoothly without manual intervention.



# CI/CD Stages

The three stages I've set up in my CI/CD pipeline are Source, Build, and Deploy. While setting up each part, I learned about webhooks for automation, input/output artifacts, and how CodeDeploy integrates deployments.

CodePipeline organizes the three stages into a visual pipeline, showing the flow from source to build to deploy. In each stage, you can see more details on execution status, artifacts, and any errors or logs for troubleshooting.



**Sanjana Tripathy**  
NextWork Student

[nextwork.org](http://nextwork.org)

# Source Stage

In the Source stage, the default branch tells CodePipeline which branch of my GitHub repo to monitor for changes, ensuring only updates from the right branch trigger the pipeline.

The Source stage is also where you enable webhook events, which let CodePipeline automatically detect changes in the GitHub repo and trigger the pipeline without manual intervention. This ensures continuous integration kicks in the moment code is pushed.

Add source stage [Info](#)  
Step 3 of 7

**Source**

**Source provider**  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

GitHub (via GitHub App)

**Connection**  
Choose an existing connection that you have already configured, or create a new one and then return to this task.  
 or

**Repository name**  
Choose a repository in your GitHub account.

You can type or paste the group path to any project that the provided credentials can access. Use the format 'group/subgroup/project'.

**Default branch**  
Default branch will be used only when pipeline execution starts from a different source or manually started.

**Output artifact**  
Choose the artifact format.

**CodePipeline default**  
AWS CodePipeline uses the default zip format for artifacts in the pipeline. Does not include Git metadata about the repository.

**Full clone**  
AWS CodePipeline passes metadata about the repository. This means it needs to do a full Git clone. Only supported for AWS CodeBuild actions. [Learn more](#)

Enable automatic retry on stage failure

**Webhook events**

**Webhook - optional**  
 Start your pipeline on push and pull request events.

**Webhook event filters - optional**  
Starts your pipeline on a specific event.

**Sanjana Tripathy**  
NextWork Student

[nextwork.org](http://nextwork.org)

# Build Stage

The Build stage sets up the process of compiling and packaging my application. I configured it to use CodeBuild with the buildspec.yml file for instructions. The input artifact for the build stage is the source code from my GitHub repository.

The screenshot shows the 'Add build stage' configuration page in the AWS CodePipeline console. It is Step 4 of 7. The form is titled 'Build - optional' and includes the following fields:

- Build provider:** A radio button group where 'Other build providers' is selected, and 'AWS CodeBuild' is chosen from a dropdown.
- Project name:** A search bar containing 'Q. sargana-devops' with a clear button and a 'Create project' button.
- Define buildspec override - optional:** A checkbox that is unchecked.
- Environment variables - optional:** A section with a 'Add environment variable' button.
- Build type:** A radio button group where 'Single build' is selected, with the note 'Triggers a single build.'
- Region:** A dropdown menu set to 'Asia Pacific (Mumbai)'.
- Input artifacts:** A section with a dropdown menu showing 'SourceArtifact' and a note 'Defined by: Sources'.
- Enable automatic retry on stage failure:** A checked checkbox.

# Deploy Stage

The Deploy stage is where my application gets released onto the target environment. I configured it to use CodeDeploy with my deployment group, which runs scripts and follows the appspec.yml file to complete the deployment.

The screenshot shows the 'Add deploy stage' configuration screen in the AWS CodeDeploy console. It is Step 6 of 7. The form is titled 'Deploy - optional' and includes fields for Deploy provider (set to AWS CodeDeploy), Region (Asia Pacific (Mumbai)), Input artifacts (BuildArtifact selected from a dropdown), Application name (nextwork-devops-cicd), Deployment group (nextwork-devops-cicd-deploymentgroup), and two checkboxes for Configure automatic rollback on stage failure and Enable automatic retry on stage failure. The 'Configure automatic rollback on stage failure' checkbox is checked.

A circular profile picture of a young woman with dark hair, wearing a pink top, sitting on a blue couch in an office setting.

**Sanjana Tripathy**  
NextWork Student

[nextwork.org](http://nextwork.org)

# Success!

Since my CI/CD pipeline gets triggered by changes pushed to GitHub, I tested my pipeline by adding a new line in index.jsp, committing the change, and pushing it to the repository to verify that CodePipeline automatically deployed it to production.

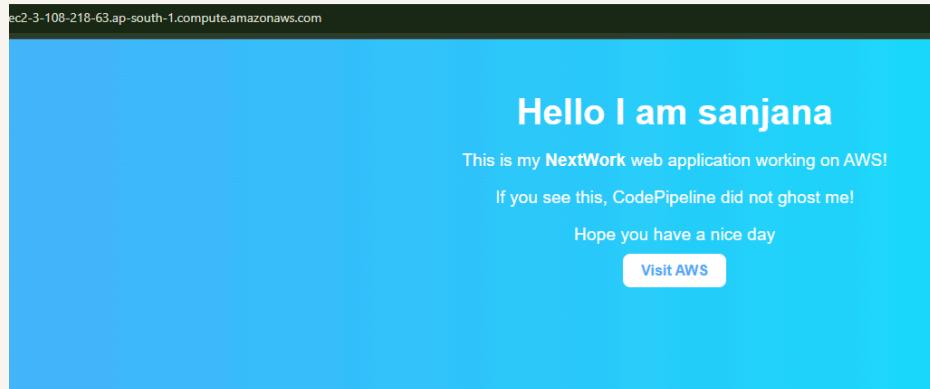
The moment I pushed the code change, a new pipeline execution started automatically. The commit message under each stage reflects the changes I pushed, allowing me to track exactly which update is being built and deployed.

Once my pipeline executed successfully, I checked the deployed web application by finding the Public IPv4 DNS of the EC2 instance via the CodeDeploy console and opening it in a browser, confirming the new line in index.jsp appeared.



**Sanjana Tripathy**  
NextWork Student

[nextwork.org](http://nextwork.org)





[nextwork.org](https://nextwork.org)

# The place to learn & showcase your skills

Check out [nextwork.org](https://nextwork.org) for more projects

