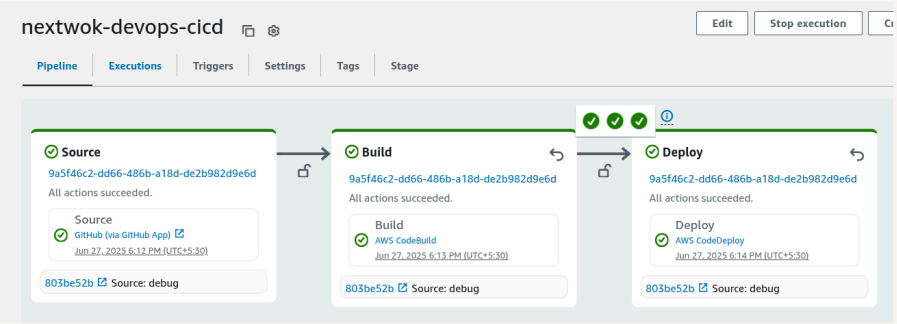
nextwork.org

Build a CI/CD Pipeline with AWS

MA Harish madhavan J U



Introducing Today's Project!

In this project I will demonstrate how to set up a CI/CD pipeline using CodePipeline . I'm doin this project to automate tracking changes. creating a build and deploying the artifact to the instance all while having an option to rollback if anything fails.

Key tools and concepts

Service I used were AWS codepipeline, CodeBuild, CodeDeploy, IAM. Key concepts I learnt include Automating CI/CD pipeline using CodePipeline, Trigger a rollback in CodePipeline.

Project reflection

This project took me approximately 5 hrs The most challenging part was configuring the pipeline It was most rewarding to watch the pipeline automatically deploying the changes to our webapp.

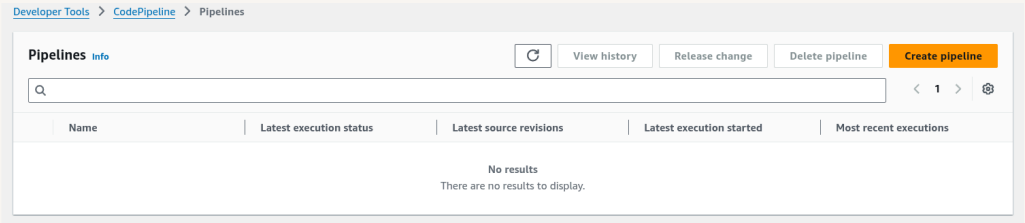
MA
Harish madhavan J U
NextWork Student

Starting a CI/CD Pipeline

AWS CodePipeline is a DevOps tool that automates workflow involved in moving code from github to Codebuild for artifact and transferring the same to CodeDeploy to deploy, which makes build more reliable and consistent with less risk of human error.

CodePipeline offers different execution modes based on how it handles new triggers in the pipeline. I choose Superseed because if there is a new change while working with previous change the new one will take over ensuring that only latest changes reflect other options include Queued where new runs are handled in ordered manner and Parallel multiple runs are handled at the same time.

A service role gets created automatically during setup so CodePipeline has access to other AWS services like CodeBuild, CodeDeploy to automate the CI/CD pipeline.

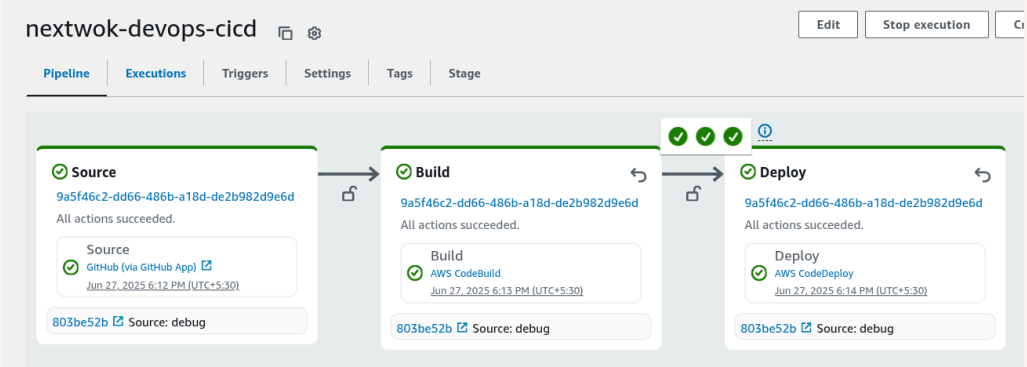


MA
Harish madhavan J U
NextWork Student

CI/CD Stages

The three stages I've set up in my CI/CD pipeline are Source (Retrieving the source code of our web app from github), Build(Creating a build using CodeBuild) and Deploy(deploy the changes to our web app) While setting up each part, I learnt about how AWS services can be used to automate CI/CD pipeline.

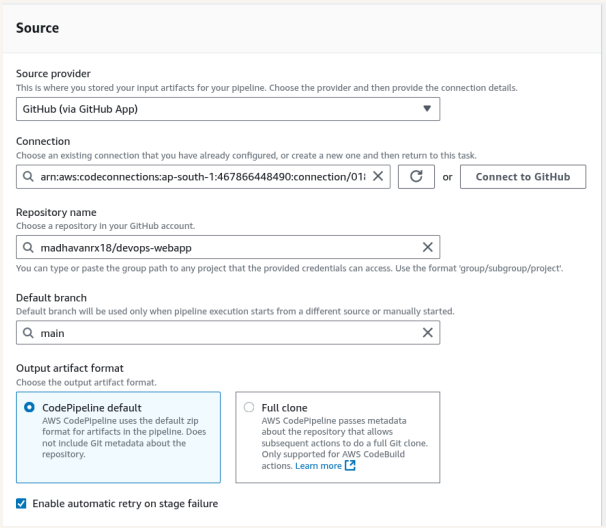
CodePipeline organises the three stages into a pipeline where In each stage we can see more details on what triggered the action, input and output artifact and the deployment details



Source Stage

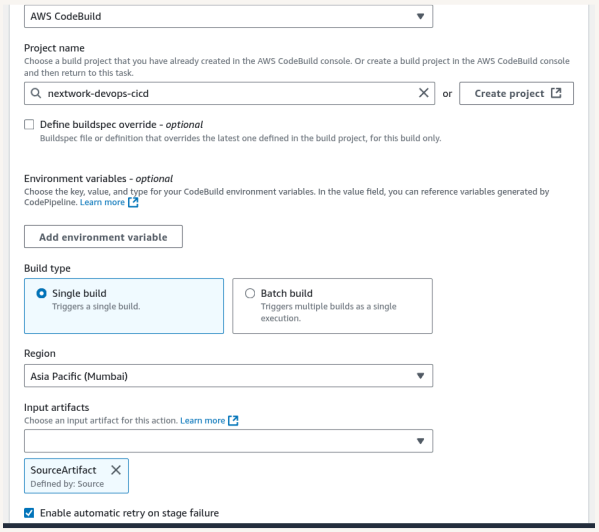
In the source stage of CodePipeline, the default branch refers to the branch in our GitHub repository where the source code resides. This setting tells CodePipeline to monitor that specific branch for any new commits, which then trigger the workflow—building an artifact using CodeBuild and deploying it using CodeDeploy. Having multiple branches in the repository allows us to maintain different versions of the project, and by specifying a particular branch in the pipeline, we can control which version gets built and deployed.

The source stage is also where you enable webhook events, which looks for any commit to the repository and triggers the CodePipeline for a new deployment.



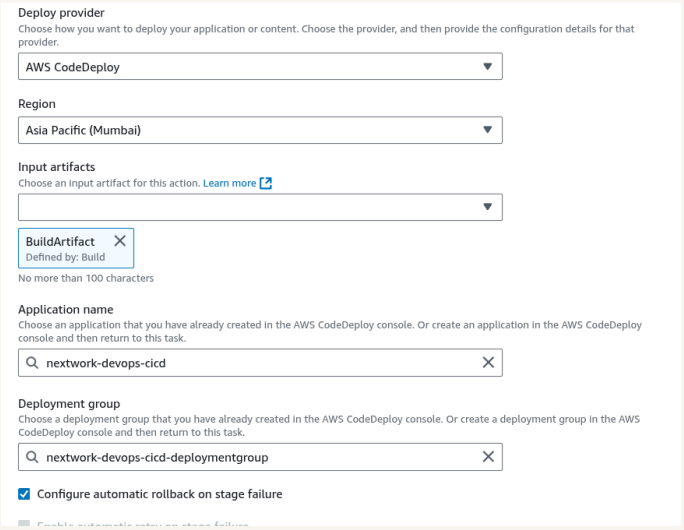
Build Stage

The Build stage defines how we compile and prepare our web application for deployment. I configured CodeBuild to take the output from the Source stage and use it to build the application. The input artifact for this stage is the SourceArtifact, which contains the compressed source code pulled from GitHub



Deploy Stage

The Deploy stage is where CodePipeline takes the output from the Build stage and deploys the application. I configured CodePipeline to use CodeDeploy for this process, deploying the built artifact to the target environment specified in a CodeDeploy deployment group.

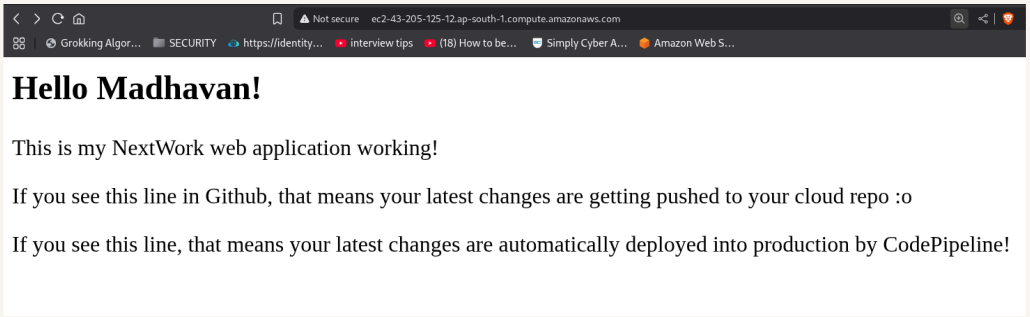


Success!

Since my CI/CD pipeline gets triggered by a commit in the repo. I tested my pipeline by adding a line to index.jsp and pushing the same to my repo. Webhook events will look for any change in the repo and trigger codepipeline if any occurs

The moment I pushed the code change Codepipeline was triggered to deploy the change in our web app. The commit message under each stage reflects only latest commit is being processed to ensure only latest changes are deployed in our web app.

Once my pipeline executed successfully, I checked the production environment instance for public dns, from where we can access the website and witnessed the latest code change in our website.

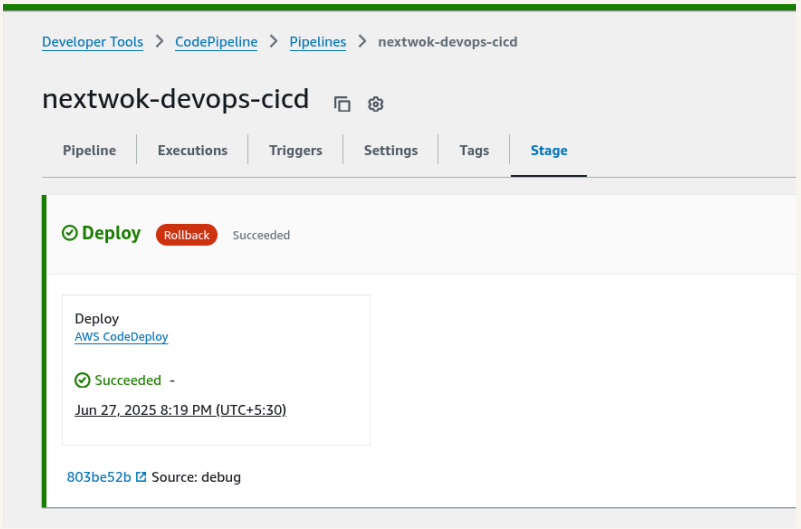

Not secure
ec2-43-205-125-12.ap-south-1.compute.amazonaws.com
Grokking Algor...
SECURITY
https://identity...
interview tips
(18) How to be...
Simply Cyber A...
Amazon Web S...
Hello Madhavan!
This is my NextWork web application working!
If you see this line in Github, that means your latest changes are getting pushed to your cloud repo :o
If you see this line, that means your latest changes are automatically deployed into production by CodePipeline!

Testing the Pipeline

In a project extension, I initiated a rollback on Deploy stage in CodePipeline. Automatic rollback is important for ensuring latest changes does not break the web app, if any error occurs it reverts back to the previous version.

During the rollback, the source and build stage are not affected because they are preceeding stages. I could verify this by comparing latest commit message in the stages where deploy has a earlier commit compared to other two stages.

After rollback, the live web app reverted to the previous version where the new line we added to verify codepipeline disappeared.


Developer Tools > CodePipeline > Pipelines > nextwok-devops-cicd
nextwok-devops-cicd
Pipeline
Executions
Triggers
Settings
Tags
Stage
Deploy
Rollback
Succeeded
Deploy
AWS CodeDeploy
Succeeded -
Jun 27, 2025 8:19 PM (UTC+5:30)
803be52b
Source: debug

The place to learn & showcase your skills

Check out nextwork.org for more projects