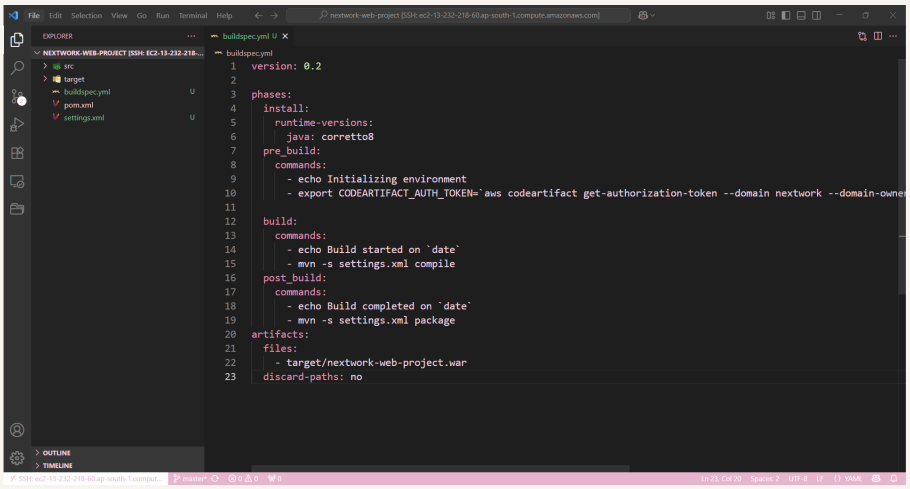
nextwork.org

Continuous Integration with CodeBuild

VE vedikagangil@gmail.com



VE
vedikagangil@gmail.com
NextWork Student

Introducing Today's Project!

In today's project I will create and configure a CodeBuild project from scratch. Then I will connect my CodeBuild project to my GitHub repository. Next define my build process using a buildspec.yml file. Lastly automate testing using CodeBuild too!

Key tools and concepts

The key service I used today was CodeBuild which helped me create my project completely on cloud. I also integrated various other AWS services like S3 buckets, IAM roles, IAM policies, etc.

Project reflection

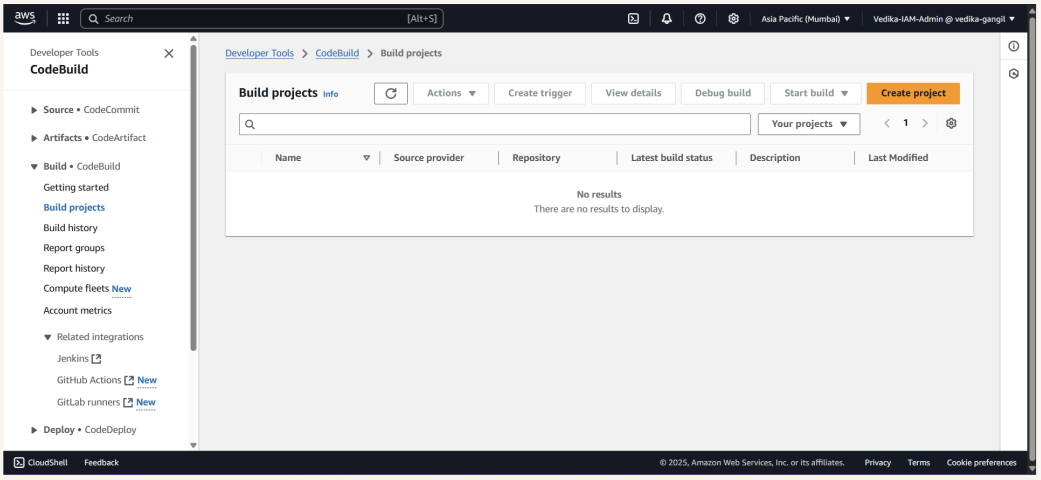
This project took me 2 hous to complete.

This project is part four of a series of DevOps projects where I'm building a CI/CD pipeline! I'll be working on the next project tomorrow!

Setting up a CodeBuild Project

CodeBuild is a continuous integration service, which means it automates building, testing, and packaging code changes whenever developers push updates. Engineering teams use it because it catches bugs early, ensures consistent builds, etc.

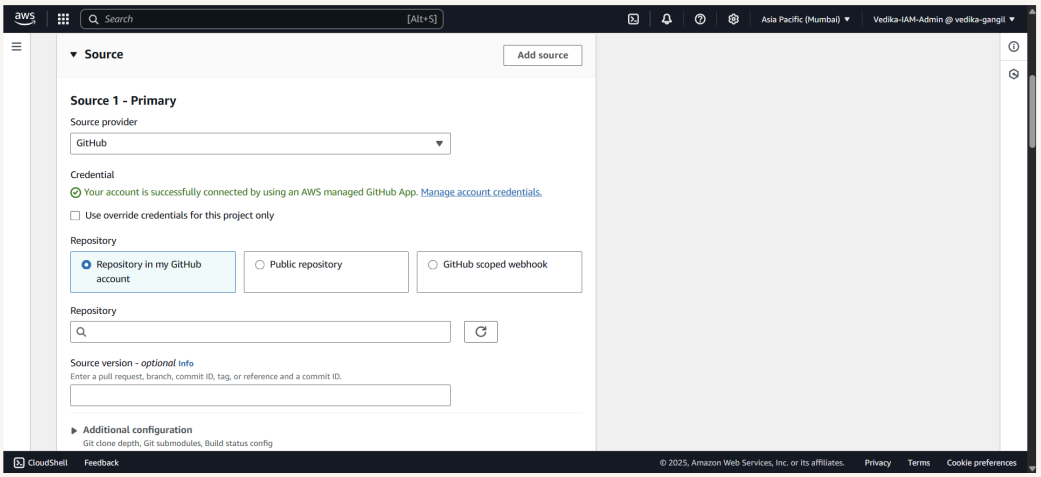
My CodeBuild project's source configuration specifies the repository (e.g., GitHub) and branch to pull code from, plus triggers like automatic builds on commit.



Connecting CodeBuild with GitHub

There are multiple credential types for GitHub, like personal access tokens (PATs), SSH keys, and OAuth apps. I used GitHub App because it offers fine-grained permissions, tighter security, and seamless integration with workflows like CI/CD.

AWS CodeStar Connections securely linked our AWS services (like CodePipeline) to GitHub by creating a token-based bridge, bypassing OAuth. This allowed automated, permission-controlled code syncs without storing GitHub credentials in AWS.



CodeBuild Configurations

Environment

My CodeBuild project's environment is configured with an AWS-managed Linux image and on-demand compute for reliable builds.

Artifacts

Build artifacts are the compiled outputs generated during a CI/CD pipeline run. They're important because they contain the deployable code and test results needed for later stages. To store them, I have created an S3 bucket.

Packaging

When setting up CodeBuild, I also chose to package artifacts in a ZIP file because it compresses multiple build outputs into a single portable unit. This simplifies storage, accelerates deployments, and ensures consistency.

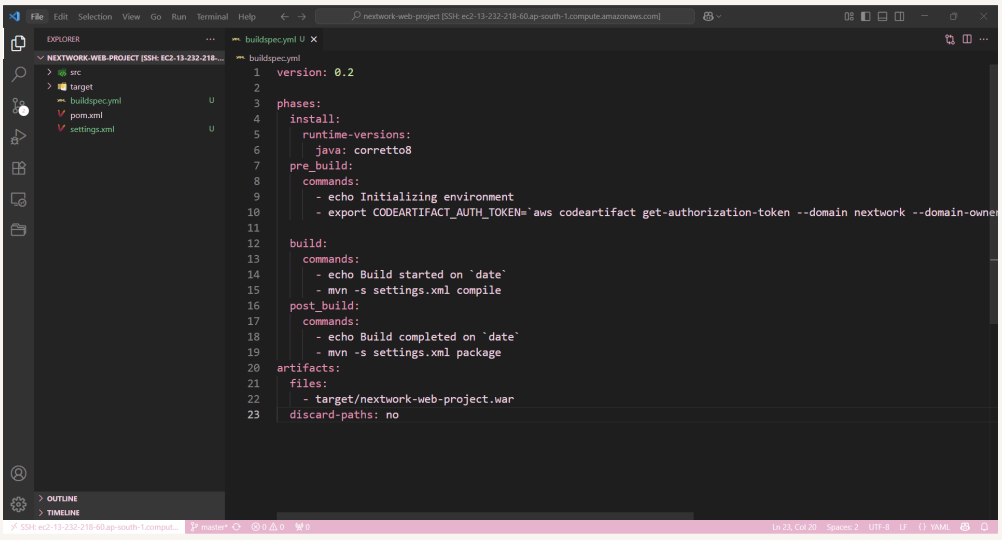
Monitoring

CloudWatch Logs captures real-time CodeBuild output, storing logs for debugging, auditing, and alerts. I enabled it to track failures, analyze trends, and meet compliance needs with searchable, retained logs even after builds complete.

buildspec.yml

My first build failed because CodeBuild couldn't find the buildspec.yml file in my GitHub repository, which makes sense because I haven't created it yet. A buildspec.yml file is needed because CodeBuild will read the file like a instruction manual.

The first two phases in my buildspec.yml file (install and pre_build) set up dependencies and pre-build checks. The third phase (build) compiles code/runs tests. The fourth phase (post_build) packages artifacts (e.g., ZIPs) and triggers deployments.

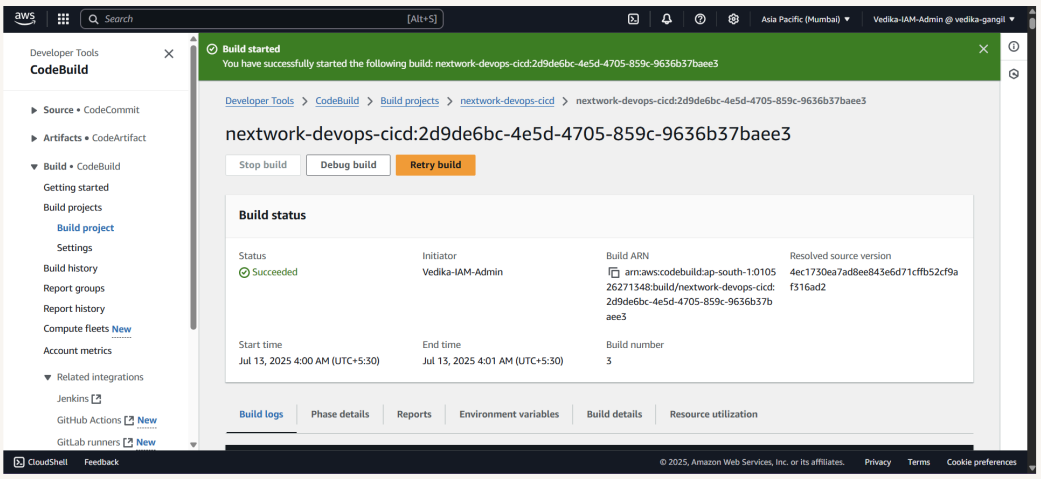


Success!

My second build also failed, but with a different error that said "no matching artifact paths found". To fix this I need to grant CodeBuild's IAM role the permission to access CodeArtifact.

To resolve the second error, I enabled my IAM role to have a specific IAM policy which was necessary for a successful bulid. When I built my project again, I saw that everything was working properly and the build was successful.

To verify the build, I checked the S3 bucket configured in CodeBuild's artifacts section. Seeing the artifact tells me the build succeeded and the output is ready for deployment.



The place to learn & showcase your skills

Check out nextwork.org for more projects