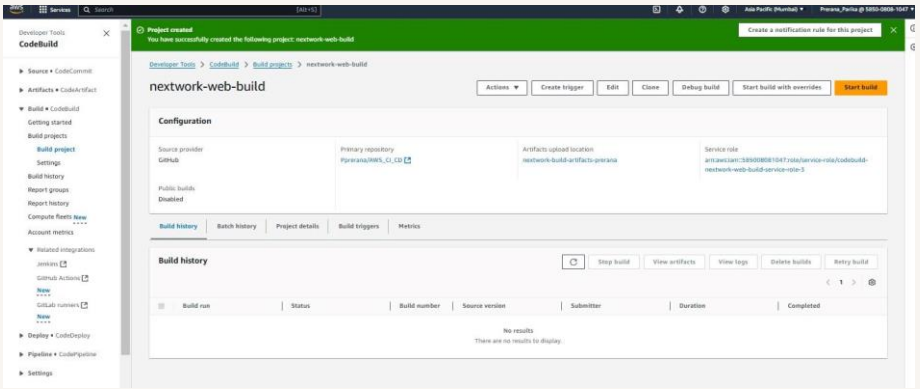
Package an App with CodeBuild



Introducing today's project!

What is AWS CodeBuild?

AWS CodeBuild is a fully managed continuous integration service that compiles source code, runs tests, and produces deployable build artifacts. It is useful because it eliminates the need to set up and manage build servers, automatically scales to handle multiple builds concurrently, and integrates seamlessly with other AWS services like CodePipeline, S3, and CloudWatch. This helps streamline the development process, automate builds, and improve efficiency in deploying applications.to ha

How I used CodeBuild in this project

I used AWS CodeBuild in today's project to automate the process of building my web app. CodeBuild pulled the source code from the repository, compiled it, ran tests, and packaged the output into a zip file (a build artifact). The build artifact was then stored in an S3 bucket, ready for deployment. This streamlined the build process and ensured everything was automated and efficient. t

One thing I didn't expect in this project was...

One thing I didn't expect in this project was attaching the policy to the role. Setting up the correct permissions for the CodeBuild role required more attention than anticipated to ensure it could access resources like the S3 bucket and CloudWatch Logs.

This project took me...

This project took me 20 minutes to complete.

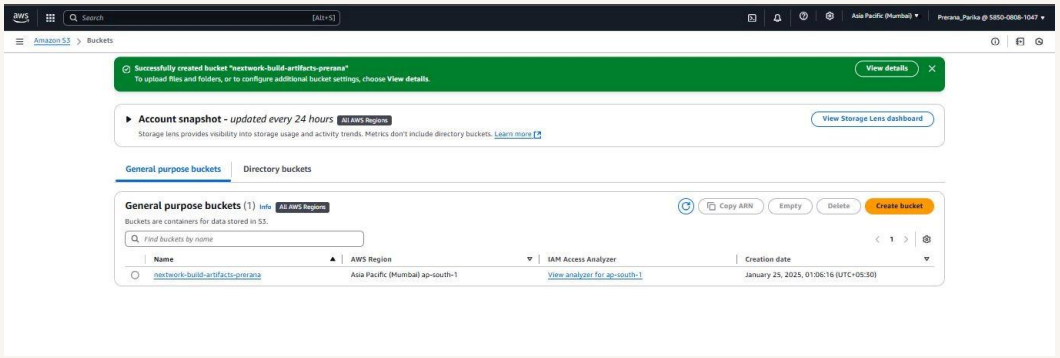
Set up an S3 bucket

I started my project by creating an S3 bucket because it serves as a centralized, reliable storage location for artifacts generated during the build process. S3 buckets are ideal for storing files that can be later retrieved or used for further processing. In the context of setting up AWS CodeBuild, the S3 bucket acts as the intermediary between the build environment and the final output files. By creating the bucket beforehand, I ensure that there is a destination ready for CodeBuild to store the files it produces during the build process. This step helps streamline the continuous integration pipeline, providing a scalable and durable storage solution for files such as build artifacts, logs, or reports.

S3 is often the default choice for such tasks due to its low-cost, high-availability, and ease of integration with various AWS services like CodeBuild. Moreover, CodeBuild requires the S3 bucket to be pre-configured to avoid delays and errors when the build process is initiated. This advanced setup also simplifies automation because it ensures that whenever the build completes, the necessary files are already stored in an accessible location, which can later be used by other services or workflows. Furthermore, by creating the S3 bucket early, we also set up permissions properly, making it easier to manage access control later. We can configure the bucket's permissions to ensure that only the appropriate services and users can access the stored files. By preparing this environment ahead of time, we reduce the chances of encountering permission-related issues during the actual build process. Proper configuration of these permissions is crucial for maintaining security and operational efficiency within the AWS ecosystem.
In summary, setting up the S3 bucket in advance is an essential step in ensuring that AWS CodeBuild has a reliable and appropriately configured location to store its output files. By doing so, I streamline the build process and ensure that all necessary resources are in place for subsequent steps in the project.

The build process translates web app code into machine code that servers can run. It involves compiling, testing, and packaging the app. The key build artifact is a WAR file, which bundles everything a server needs to host the web app, similar to a zip file.

This artifact is important because it bundles all the necessary files and resources required to deploy and run the web app on a server. The WAR file simplifies deployment by packaging everything into a single, easily transferable file that servers can extract and execute.



Set up a CodeBuild project

Source

My CodeBuild project's Source configuration means that CodeBuild will pull the source code for the build process from a specified location. I selected a repository in AWS CodeCommit, GitHub, or another supported source control service where my web app's code is stored. This ensures CodeBuild has access to the latest version of the code to build and package into the desired artifact.

Environment

My CodeBuild project's Environment configuration means that I am specifying the runtime environment in which the build will execute. I selected the appropriate build environment, such as an operating system (e.g., Amazon Linux 2) and a runtime (e.g., Java, Node.js), to ensure CodeBuild has the necessary tools and settings to build my web app correctly.

Artifacts

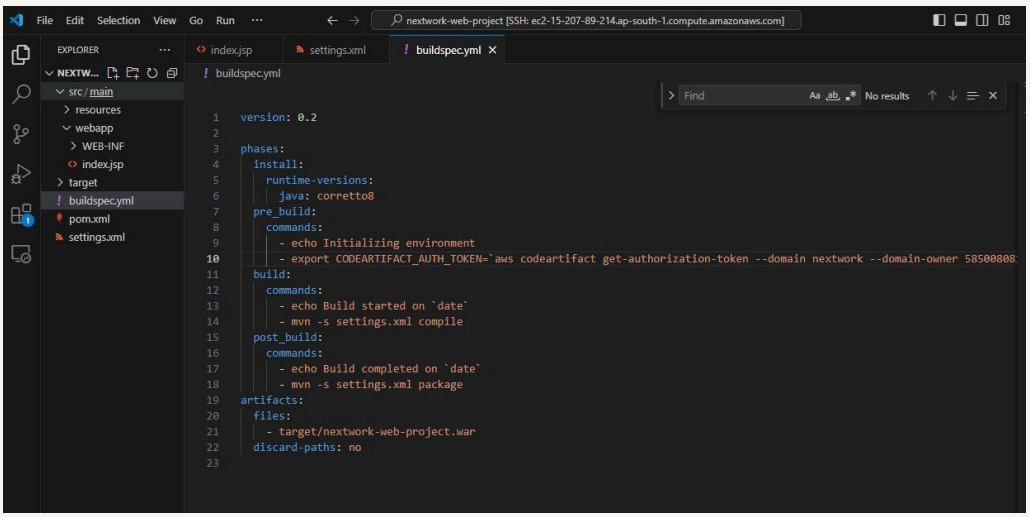
CodeBuild project's Artifacts configuration means that I am specifying where and how the output of the build process will be stored after the build completes. I selected an S3 bucket to store the build artifact, which is a zip or WAR file, making it accessible for deployment or further use in the pipeline.

Logs

CodeBuild project's Logs configuration means that I am specifying where the logs of the build process will be stored for monitoring and troubleshooting. I selected Amazon CloudWatch Logs to capture and view detailed logs of the build, allowing me to track progress, errors, and other important events during the build process.

Create a buildspec.yml file

I created a buildspec.yml file in my project because it defines the build instructions for AWS CodeBuild. This file specifies the phases of the build process, such as installing dependencies, running tests, and packaging the application, along with the commands and environment variables needed to execute these steps.



Create a CodeBuild build project

My buildspec.yml file has four stages

The first two phases in my buildspec.yml file were the install phase, where I set up dependencies and tools needed for the build, and the pre_build phase, where I ran preparatory commands such as configuring the environment or validating prerequisite

The third phase in my buildspec.yml file was the **build** phase, where the actual build process took place. In this phase, I compiled the code, ran tests, and created the build artifact, such as a zip file or WAR file, which was then stored in the designated S3 bucket.

The fourth phase in my buildspec.yml file was the **post_build** phase, where any post-build actions were executed. This typically included steps like notifying other services,

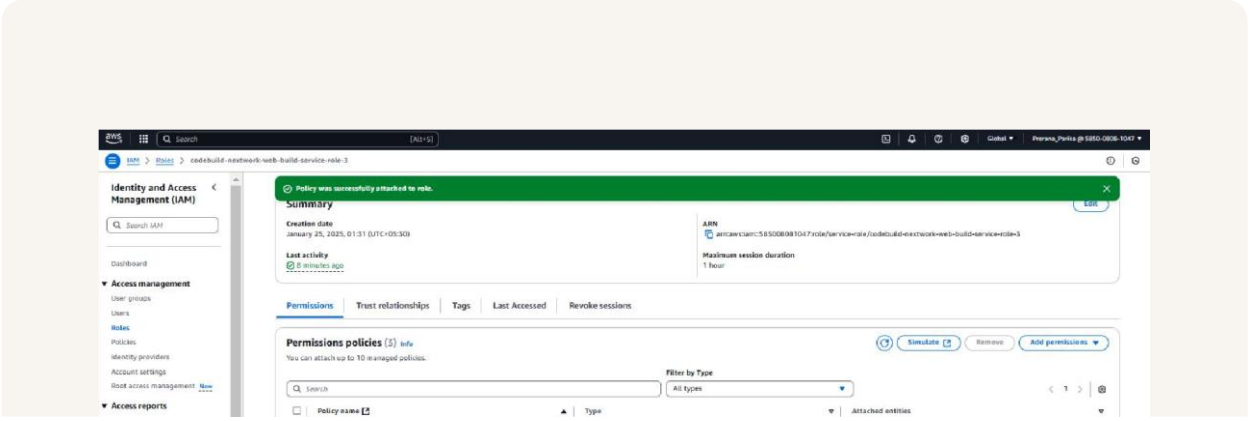
uploading build artifacts to an S3 bucket, or running additional scripts to finalize the build process.

Modify CodeBuild’s IAM role

Before building my CodeBuild project, I modified its service role first.

My CodeBuild project's service role was first created when I set up the CodeBuild project. This role was automatically generated to allow CodeBuild to interact with other AWS services, such as S3 for storing build artifacts and CloudWatch for logging.

I attached a new policy called `codeartifact-nextwork-consumer-policy` and `CodeBuildCloudWatchLogsAccess` to my CodeBuild role. The `codeartifact-nextwork-consumer-policy` allows CodeBuild to access resources in AWS CodeArtifact, while `CodeBuildCloudWatchLogsAccess` grants permissions to write logs to Amazon CloudWatch for monitoring and troubleshooting the build process.



My first project build

To build my project, all I had to do was navigate to the AWS CodeBuild console, select my project, and click the "Start Build" button. This initiated the build process using the configured source, environment, and settings.

The build process in CodeBuild took 1 minute and 19 seconds to complete.

Once the build was complete, I checked the S3 bucket where the zip file, containing the build artifact, was created and stored.

I saw a zip file in the S3 bucket, which verified that the build was completed successfully.