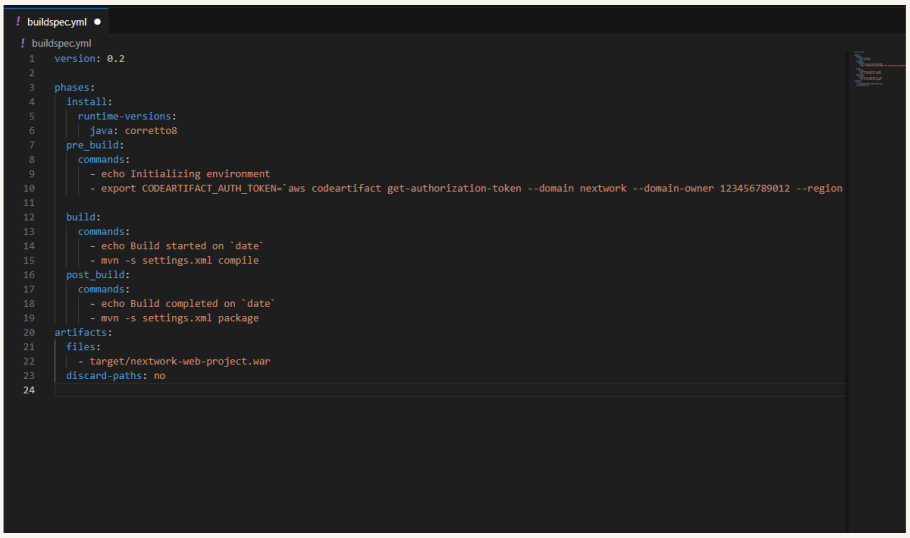
nextwork.org

Continuous Integration with CodeBuild

L Louis Moyo



L
Louis Moyo
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to set up continuous integration for my Java web app using AWS CodeBuild. I'm doing this project to learn how to automate the build process so that whenever code changes are pushed to GitHub, CodeBuild can automatically compile, test, and package the application. This helps me understand how CI pipelines reduce manual effort, prevent build errors, and ensure consistent, reliable deployments in real-world DevOps workflows.

Key tools and concepts

Services I used were AWS CodeBuild, CodeArtifact, S3, IAM, and GitHub. Key concepts I learnt include automating builds, handling IAM permissions, running test scripts within buildspec.yml, and verifying artifacts in S3 as part of a CI/CD pipeline.

Project reflection

This project took me approximately 3 hours. The most challenging part was troubleshooting permissions and getting CodeBuild to access settings.xml. The most rewarding part was seeing the build succeed with automated tests and an artifact created in S3.

L

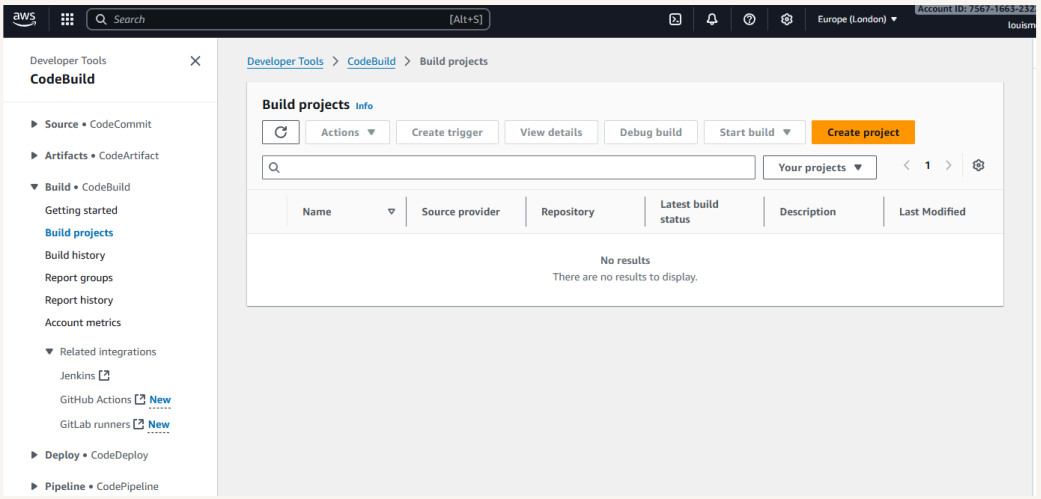
Louis Moyo
NextWork Student

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, where I'll continue integrating more AWS services and extend the pipeline toward deployment automation.

Setting up a CodeBuild Project

CodeBuild is a continuous integration (CI) service, which means it automatically builds, tests, and packages code whenever changes are pushed to a repository. Instead of developers manually compiling and testing code, CI services like CodeBuild run this process in a repeatable, isolated environment. Engineering teams use CI services because they improve speed, consistency, and quality -every commit is verified, bugs are caught earlier, and the risk of integration issues is reduced. This automation also saves developer time and enables teams to collaborate more effectively on large projects.

My CodeBuild project's source configuration means the location where CodeBuild will fetch my application's code before building it. I selected GitHub as the source provider because that's where my web app is stored, and this allows CodeBuild to automatically pull the latest changes whenever I update my repository. By setting GitHub as the source, I'm ensuring a smooth connection between my codebase and the CI pipeline, so that every commit can trigger a consistent and automated build process.

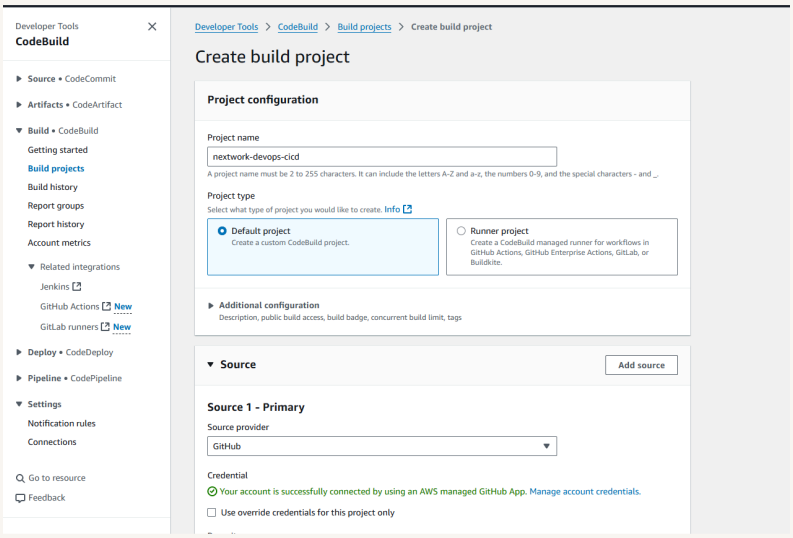

aws
Search
[Alt+S]
Europe (London)
Developer Tools
CodeBuild
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build history
Report groups
Report history
Account metrics
Related integrations
Jenkins
GitHub Actions New
GitLab runners New
Deploy • CodeDeploy
Pipeline • CodePipeline
Developer Tools > CodeBuild > Build projects
Build projects Info
Actions
Create trigger
View details
Debug build
Start build
Create project
Your projects
Name
Source provider
Repository
Latest build status
Description
Last Modified
No results
There are no results to display.

Louis Moyo
NextWork Student

Connecting CodeBuild with GitHub

There are multiple credential types for GitHub, like personal access tokens and OAuth apps, but I used GitHub App because it's the most secure and easiest to manage. With GitHub App, AWS automatically handles the connection and credential rotation, so I don't need to store or update tokens myself. This reduces operational overhead while ensuring CodeBuild always has a safe, authorised link to my repository.

The service that helped connect my AWS account to GitHub is AWS CodeConnections. It acted as the secure bridge that allowed CodeBuild to authenticate with my GitHub repository without me having to manage tokens or credentials manually. By using CodeConnections, I was able to establish a trusted, ongoing integration so that CodeBuild can automatically fetch code from GitHub whenever I push changes, enabling a seamless CI workflow.



CodeBuild Configurations

Environment

My CodeBuild project's Environment configuration means I set up the resources that will actually run my builds. I chose the On-demand provisioning model, so AWS spins up resources only when needed, making it cost-effective. For the environment image, I used a Managed image, since AWS provides preconfigured templates with the required tools. I set the compute type to EC2, which gives me flexible and powerful instances to run my Java build, and configured the OS as Amazon Linux, the runtime as Standard, and the image as Corretto 8. Finally, I created a new service role so CodeBuild has the right permissions for this project.

Artifacts

Build artifacts are the packaged outputs generated at the end of a build process, and they're important because they are what actually gets deployed or distributed. My build process will create a WAR file (Web Application Resource), which bundles up everything my Java web app needs to run on a server. To store this artifact securely and make it easy to access for deployment later, I created an Amazon S3 bucket named nextwork-devops-cicd-louis in the same region as my CodeBuild project.

Packaging

When setting up CodeBuild, I also chose to package artifacts in a Zip file because it keeps everything lightweight, organised, and easy to handle. Zip compression reduces the file size, which makes uploads to S3 faster and saves on storage costs.

It also bundles multiple build outputs into a single tidy package, making it simpler to manage, deploy, and share compared to handling many separate files.

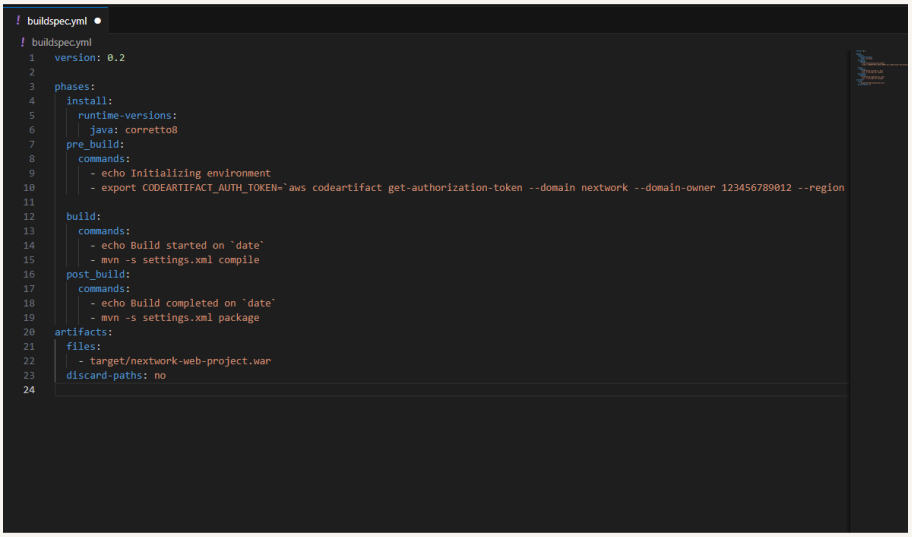
Monitoring

For monitoring, I enabled CloudWatch Logs, which is AWS's service for collecting and storing log data. This ensures that every step of my build process - from commands executed to errors encountered - is recorded in one place. I'm using it because it makes debugging much easier, provides full visibility into the build process, and creates an auditable history that I can reference later if something goes wrong. By setting a custom log group name, I can also keep my project's logs organised and easy to find.

buildspec.yml

My first build failed because CodeBuild couldn't find a buildspec.yml file in the root of my repository. This file is required since it tells CodeBuild exactly how to build my project - including which dependencies to install, which commands to run, and what artifacts to produce. Without it, CodeBuild has no instructions to follow, so the build process stops immediately with a YAML_FILE_ERROR. A buildspec.yml file is needed because it serves as the blueprint for automating builds, ensuring that the same process can be repeated consistently and reliably every time new code is pushed.

The install phase sets up dependencies like Maven. The pre_build phase runs prep tasks such as exporting variables or logging into AWS. The build phase is where the code compiles, tests run, and the WAR file is generated. Finally, the post_build phase handles cleanup and outputs, like zipping and uploading artifacts to S3. Each phase ensures the build runs in order and completes reliably.

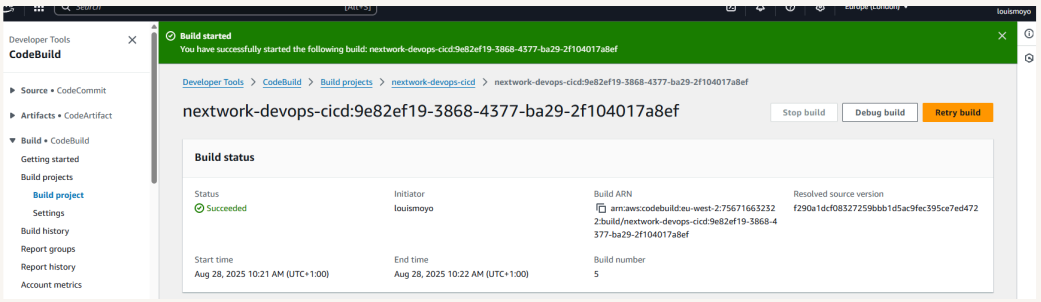

buildspec.yml
buildspec.yml
version: 0.2
phases:
install:
runtime-versions:
java: corretto8
pre_build:
commands:
- echo Initializing environment
- export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain nextwork --domain-owner 123456789012 --region
build:
commands:
- echo Build started on `date`
- mvn -s settings.xml compile
post_build:
commands:
- echo Build completed on `date`
- mvn -s settings.xml package
artifacts:
files:
- target/nextwork-web-project.war
discard-paths: no

Success!

My second build failed with an error that CodeBuild couldn't access settings.xml or connect to CodeArtifact for dependencies. This happens because the CodeBuild role lacks the right IAM permissions. To fix it, I need to add policies like AWSCodeArtifactReadOnlyAccess, AmazonSSMReadOnlyAccess (for tokens), and AmazonS3FullAccess so CodeBuild can download dependencies and store artifacts successfully.

To resolve the second error, I realized that CodeBuild was failing because it could not find the settings.xml file, as it was being ignored by .gitignore. This meant that although the file existed locally in my EC2 environment, it was never being committed and pushed to GitHub, so CodeBuild had nothing to use when running Maven commands. To fix this, I force-added the settings.xml file with git add -f settings.xml, created a new commit, and pushed the changes to my GitHub repository. Once the file was included in source control, CodeBuild was able to detect and download it during the build process. When I retried the build, Maven could now access the settings configuration correctly and progress past the earlier failure. I confirmed this by seeing the build steps execute successfully where it previously failed, proving that the missing settings.xml was the root cause.

To verify the build, I checked the S3 bucket linked to my CodeBuild project (nextwork-devops-cicd). After refreshing the bucket contents, I saw the nextwork-devops-cicd-artifact.zip file had been uploaded. Inside that archive, the nextwork-web-project.war file was present, which confirms that the Maven build successfully compiled the project and packaged it. Seeing the artifact in S3 tells me the CI pipeline worked end-to-end: the code compiled, the artifact was created, and it was stored in the correct destination for deployment.


Developer Tools
CodeBuild
Source • CodeCommit
Artifacts • CodeArtifact
Build • CodeBuild
Getting started
Build projects
Build project
Settings
Build history
Report groups
Report history
Account metrics
Build started
You have successfully started the following build: nextwork-devops-cicd:9e82ef19-3868-4377-ba29-2f104017a8ef
Developer Tools > CodeBuild > Build projects > nextwork-devops-cicd > nextwork-devops-cicd:9e82ef19-3868-4377-ba29-2f104017a8ef
nextwork-devops-cicd:9e82ef19-3868-4377-ba29-2f104017a8ef
Stop build
Debug build
Retry build
Build status
Status
Succeeded
Initiator
louismoyo
Build ARN
arn:aws:codebuild:eu-west-2:756716632322:build/nextwork-devops-cicd:9e82ef19-3868-4377-ba29-2f104017a8ef
Resolved source version
f290a1dcf08327259bbb1d5ac9fec395ce7ed472
Start time
Aug 28, 2025 10:21 AM (UTC+1:00)
End time
Aug 28, 2025 10:22 AM (UTC+1:00)
Build number
5

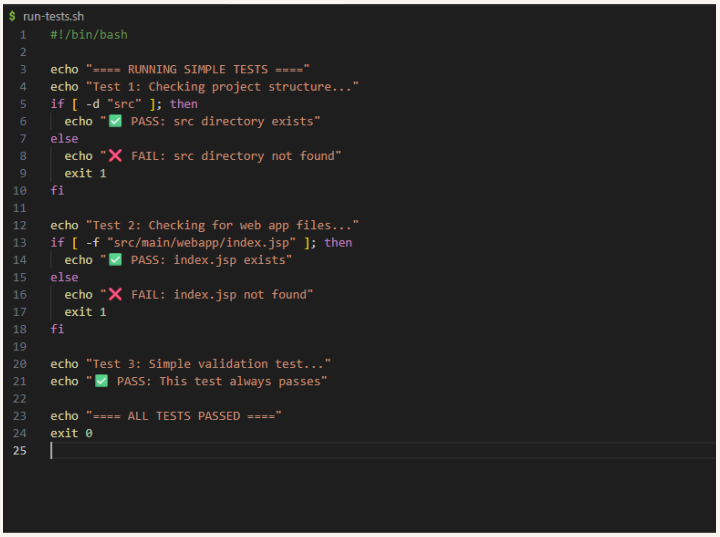
Louis Moyo
NextWork Student

Automating Testing

In a project extension, I added a test script that verifies the basic project structure and key files. It checks whether the src directory exists, confirms that index.jsp is present inside src/main/webapp, and includes a simple validation test. This ensures that the build process only continues if the essential files and structure are in place.

To add my test script to the build process, I created a run-tests.sh script in the project root to check that the src directory and index.jsp file exist. I made it executable with chmod +x and verified it locally. Then I updated buildspec.yml so CodeBuild runs the script during the build phase, adding clear markers to track progress in the logs. After committing and pushing to GitHub, CodeBuild automatically included the test step in the CI pipeline.

After pushing my code to GitHub, I triggered a new build in CodeBuild. In the build logs, I looked for the custom markers I had added in buildspec.yml. Around lines 45–50, I saw the test phase start, the run-tests.sh script execute, and the “ALL TESTS PASSED” output. This confirmed that my CI pipeline was running automated tests before compiling and packaging the project, proving that testing had been fully integrated into the build process.


$ run-tests.sh
#!/bin/bash
echo "==== RUNNING SIMPLE TESTS ===="
echo "Test 1: Checking project structure..."
if [-d "src"]; then
echo "✅ PASS: src directory exists"
else
echo "❌ FAIL: src directory not found"
exit 1
fi
echo "Test 2: Checking for web app files..."
if [-f "src/main/webapp/index.jsp"]; then
echo "✅ PASS: index.jsp exists"
else
echo "❌ FAIL: index.jsp not found"
exit 1
fi
echo "Test 3: Simple validation test..."
echo "✅ PASS: This test always passes"
echo "==== ALL TESTS PASSED ===="
exit 0

The place to learn & showcase your skills

Check out nextwork.org for more projects