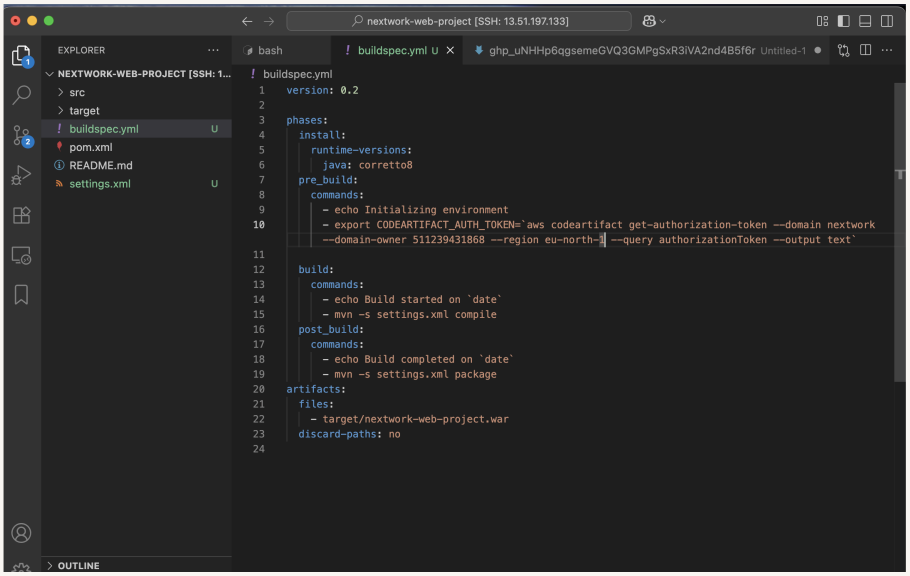
nextwork.org

Continuous Integration with CodeBuild

Kehinde Abiuwa



K
Kehinde Abiuwa
NextWork Student

Introducing Today's Project!

In this project, I will demonstrate how to automate the build process for a web application by implementing a fully functional CI/CD pipeline. The goal is to gain hands-on experience with continuous integration and continuous delivery while streamlining the deployment workflow.

Key tools and concepts

Services I used were: GitHub for version control and source code management, AWS CodeBuild for automated builds and testing. Key concepts I learned include: Setting up continuous integration pipelines to automatically build and test code after every push. Writing and configuring buildspec.yml files to define build, test, and post-build commands. Understanding how automated testing fits into the CI/CD process and ensures code quality. Integrating cloud-based build services with version control systems like GitHub. Monitoring build logs and troubleshooting build failures in a cloud environment.

Project reflection

This project took me approximately 90 minutes to complete. The most challenging part was configuring CodeBuild correctly to detect and run the test scripts without errors. It was most rewarding to see the build complete successfully with all tests passing, knowing that my CI/CD setup was working end-to-end.

K

Kehinde Abiuwa
NextWork Student

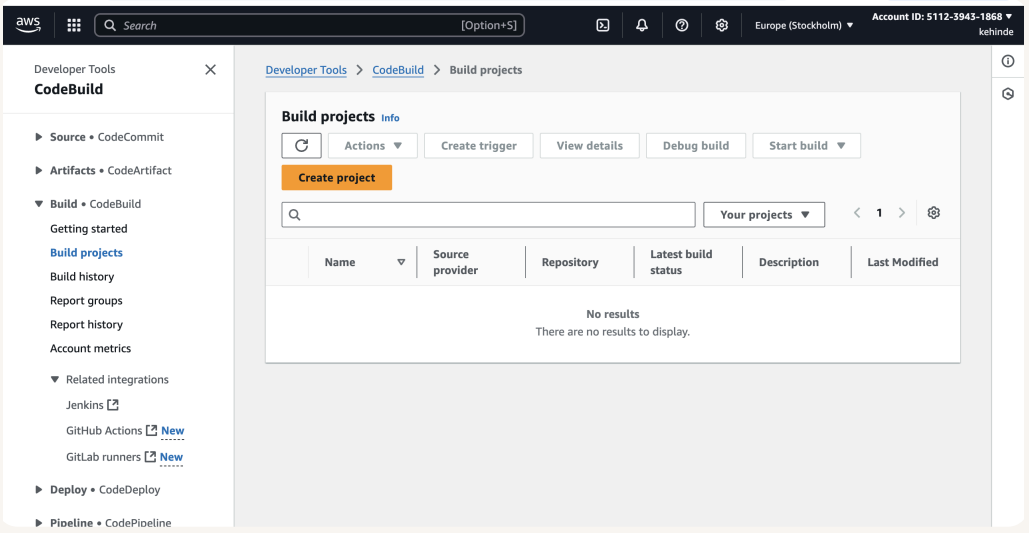
This project is part four of a series of DevOps projects where I'm building a CI/CD pipeline! I'll be working on the next project right away

Kehinde Abiuwa
NextWork Student

Setting up a CodeBuild Project

CodeBuild is a continuous integration service, which means it automatically builds and tests your application every time you commit changes to your source code repository. This ensures that new code integrates smoothly with the existing codebase and helps detect issues early in the development cycle. Engineering teams use it because it eliminates the need to provision and maintain dedicated build servers, scales seamlessly to handle multiple builds in parallel, and integrates easily with other AWS services like CodeCommit, CodePipeline, and CodeArtifact. This makes the software delivery process faster, more reliable, and cost-efficient.

My CodeBuild project's source configuration means the location of the code that CodeBuild will fetch, compile, and package into a file you can deploy and I selected github as this is where I have stored my web app's code.

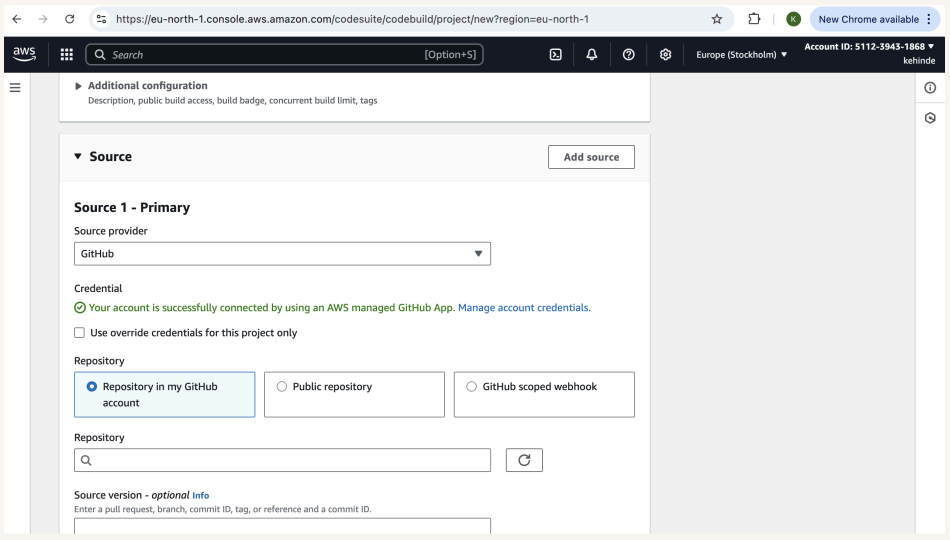


Kehinde Abiuwa
NextWork Student

Connecting CodeBuild with GitHub

There are multiple credential types for GitHub, like github app, personal access token & OAuth app. I used GitHub App because it is generally the simplest and most secure option.AWS manages the application and connection, reducing the need for me to handle tokens or keys directly. It's recommended for most use cases due to its easue of use and enhances security.

The service that helped connect our project with GitHub is AWS CodeConnections. It acts as a secure bridge between GitHub and AWS services, allowing CodeBuild to automatically access the source code stored in GitHub. By using CodeConnections, we can trigger builds and deployments whenever changes are pushed to the repository, enabling seamless integration between my version control system and my CI/CD pipeline



CodeBuild Configurations

Environment

My CodeBuild project's Environment configuration means that I defined how the build should run, including the provisioning model, operating system, runtime, and permissions. Specifically, I chose a managed image provided by AWS CodeBuild, running in "Instance" mode (on an EC2 instance). I selected Amazon Linux as the operating system with the base runtime, which ensures a stable and lightweight environment for building my web app. The environment uses the AWS-managed image aws/codebuild/ami/amazonlinux-x86_64-base:latest. For permissions, I created a new service role (codebuild-nextwork-devops-cicd-service-role) that grants CodeBuild the necessary access to AWS resources such as S3, CloudWatch, and CodeArtifact. This setup provides a secure, scalable, and preconfigured environment for compiling and packaging my application.

Artifacts

Build artifacts are the tangible outputs of a build process. They're important because theyre what I will actually deploy to my servers or distribute to users. My build process will create a build artifact that packages up everything a server could need to host my web app in one neat bundle. This bundle is called. WAR file. To store them, I created a S3 bucket because the build needs to be stored somewhere safe and accessible after the build finishes. S3 is perfect for this as it's a highly reliable and scalable storage solution that's also in our AWS environment

Packaging

When setting up CodeBuild, I also chose to package artifacts in a Zip file because it reduces storage size, speeds up transfers, and keeps everything organized in a single package. By compressing the build output, uploads to Amazon S3 are faster and storage costs are lower. Having one tidy archive also simplifies deployment, since I only need to handle a single file instead of multiple scattered build outputs. This makes sharing, deploying, and retrieving artifacts much more efficient and reliable within the CI/CD pipeline.

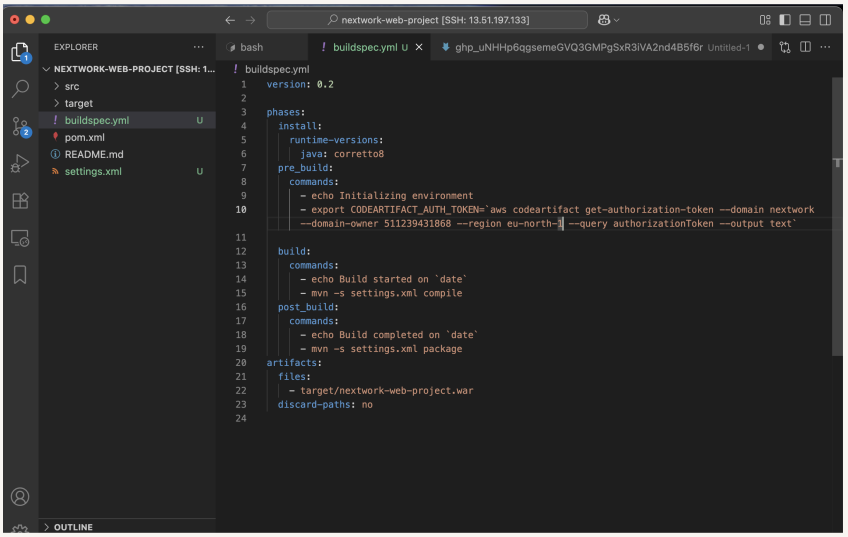
Monitoring

For monitoring, I enabled CloudWatch Logs, which is a AWS monitoring service that collects and tracks logs from AWS services

buildspec.yml

My first build failed because codebuild couldnt find a buildspec.yml file in the root directory of my webapp in github. A buildspec.yml file is needed because it tells CodeBuild the specific commands to run, the environment to use, and the artifacts to produce during the build process. Without it, CodeBuild doesn't know how to build, test, or package my application, which causes the build to fail.

The first two phases in my buildspec.yml file: install phase: This sets up the runtime environment needed for the build. In your file, it specifies Java Corretto 8 so that Maven can run properly. pre_build phase: This prepares the environment before the main build commands run. In your file, it echoes a message and fetches a CodeArtifact authorization token so your build can access private dependencies. The third phase in my buildspec.yml file: build phase: This is where the actual compilation or building of your code happens. Your commands echo the build start time and run mvn compile to compile the Java project using Maven. The fourth phase in my buildspec.yml file: post_build phase: This handles tasks after the main build is finished. In your file, it echoes the build completion time and runs mvn package to package your compiled code into a .war file ready for deployment.


nextwork-web-project [SSH: 13.51.197.133]
EXPLORER
NEXTWORK-WEB-PROJECT [SSH: 1...
src
target
buildspec.yml
pom.xml
README.md
settings.xml
bash
buildspec.yml
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
Untitled-1
OUTLINE
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 511239431868 --region eu-north-1 --query authorizationToken --output text`
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

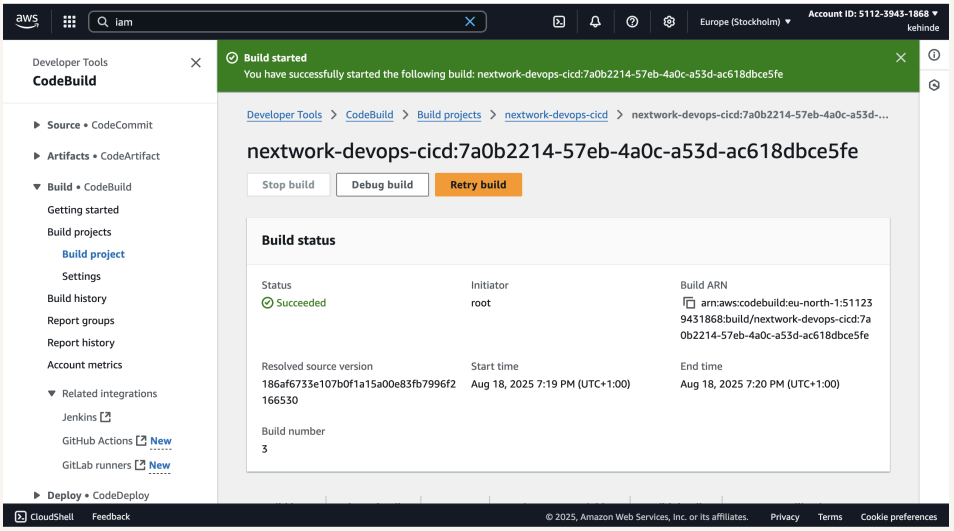
Success!

My second build also failed, but with a different error that said CodeBuild could not access CodeArtifact to download the project dependencies. To fix this, I updated the CodeBuild service role to grant it permission to access CodeArtifact, allowing it to authenticate and retrieve the necessary dependencies for the build.

To resolve the second error, I updated the CodeBuild service role to give it permission to access CodeArtifact and ensure it could download all the project dependencies. When I built my project again, I saw the build complete successfully, and the .war artifact was generated in the target directory as expected.

Seeing the artifact in my S3 bucket tells me that the build was successful, the project was packaged correctly, and CodeBuild was able to run all the specified commands in the buildspec.yml file. It confirms that the .war file is ready for deployment.

Kehinde Abiuwa
NextWork Student

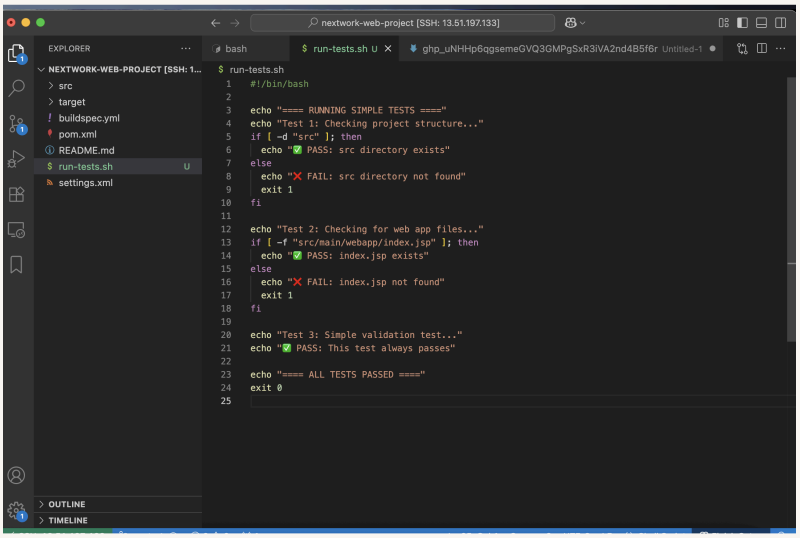


Kehinde Abiuwa
NextWork Student

Automating Testing

In a project extension, I'm setting up a testing automation process with CodeBuild. I added a test script that checks whether the essential project structure and web app files exist, and runs a simple validation test. This allows CodeBuild to automatically verify that the project is set up correctly and ready for further development or deployment.

After pushing my code to GitHub, I ran the Start Build in CodeBuild, and I could see that it automatically added the test script to the build process



The place to learn & showcase your skills

Check out nextwork.org for more projects