**😊 Building code with AWS Code Build**

**CodeBuild** is a fully managed continuous integration (CI) service provided by AWS that compiles source code, runs tests, and produces deployable software packages. It is designed to automate the build process in software development pipelines, integrating seamlessly with other AWS services and CI/CD tools.

**Key Features of AWS CodeBuild:**

1. **Build Automation**: Automates compiling, testing, and packaging of code.
2. **Fully Managed**: No need to provision or manage servers; AWS handles the infrastructure.
3. **Scalability**: Scales automatically to handle multiple builds concurrently.
4. **Integration**: Works with other AWS services like CodePipeline, CodeCommit, S3, and third-party repositories such as GitHub, GitLab, and Bitbucket.
5. **Custom Build Environments**: Supports pre-configured environments for popular programming languages and allows custom build environments using Docker.
6. **Pay-as-you-go Pricing**: Charges are based on the number of build minutes used.

**How AWS CodeBuild Works**

1. **Input**: The source code is stored in a repository (e.g., CodeCommit, GitHub, Bitbucket, or S3).
2. **Build Specification**: A buildspec.yml file defines the build steps, including phases like install, pre-build, build, and post-build.
3. **Build Environment**: A build container is provisioned based on the programming language and runtime environment.
4. **Output**: CodeBuild generates artifacts (e.g., binaries, packaged applications) and can store them in Amazon S3 or deploy them directly using other AWS services.

**Example Workflow**

1. **Source**: Code is pushed to a repository (e.g., GitHub).
2. **Trigger**: CodeBuild is triggered (manually or via a CI/CD tool like CodePipeline).
3. **Build Phases**:
   * **Install**: Install dependencies.
   * **Pre-Build**: Prepare the environment (e.g., database migrations).
   * **Build**: Compile source code or package the application.
   * **Post-Build**: Run tests or deploy artifacts.
4. **Artifacts**: The resulting application or output is stored in S3 or deployed.

**Benefits of CodeBuild**

1. **Ease of Use**: Simplifies build processes with managed infrastructure.
2. **Cost-Effective**: Pay only for the resources used during builds.
3. **Flexibility**: Custom build environments and support for multiple programming languages.
4. **Security**: Integrated with AWS Identity and Access Management (IAM) for secure access control.

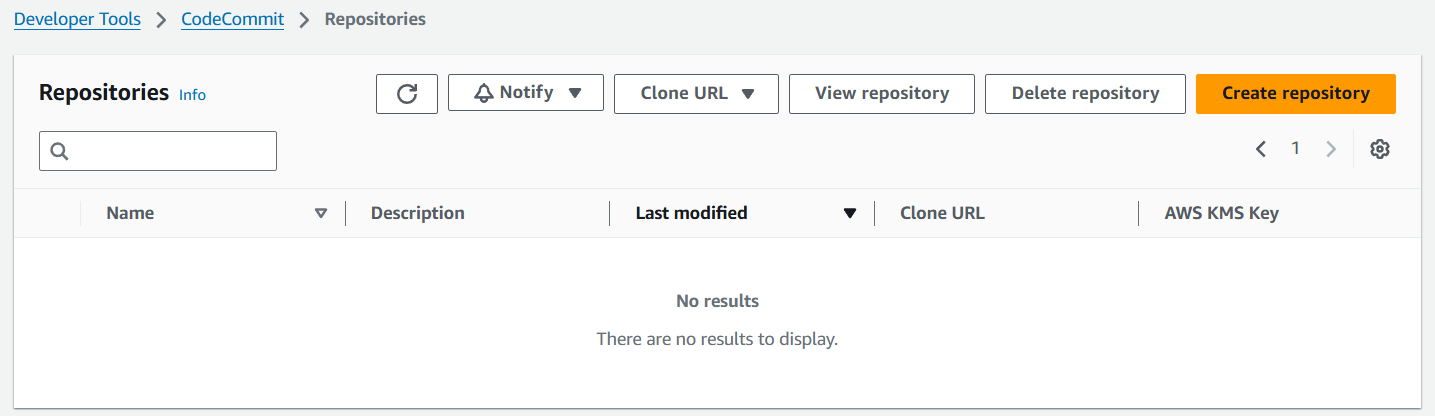
**Common Use Cases**

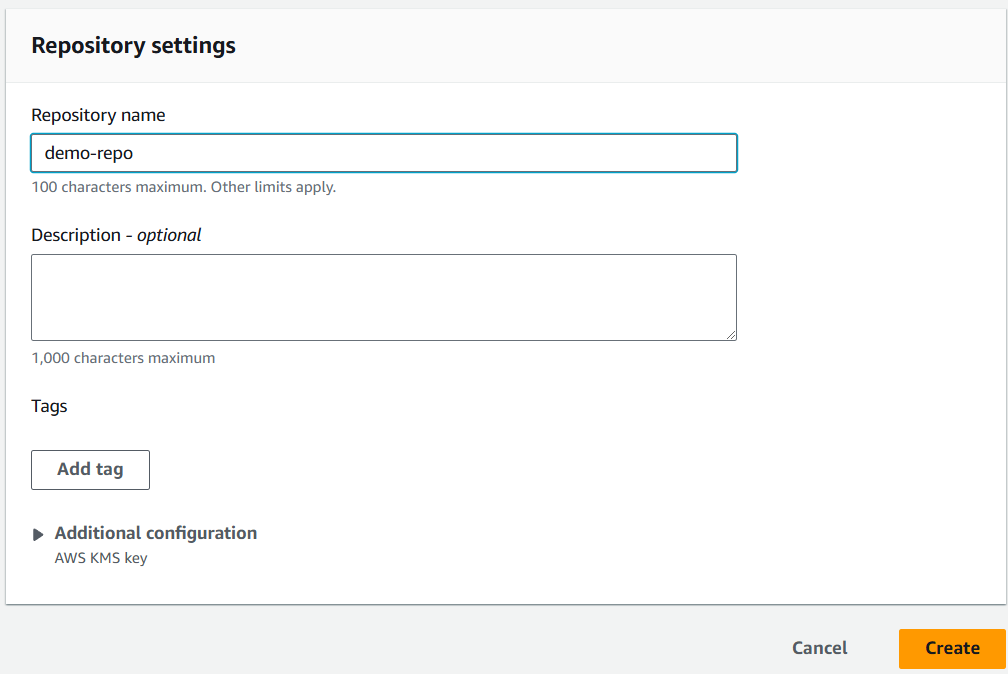
* **Continuous Integration**: Automating code builds and testing in a CI/CD pipeline.
* **Build Automation**: Simplifying repetitive tasks like dependency management and compilation.
* **Code Testing**: Running unit, integration, or performance tests during the build phase.
* **Container Builds**: Building and pushing Docker containers to Amazon Elastic Container Registry (ECR).

CodeBuild is a versatile tool for modern DevOps workflows, streamlining software delivery in cloud-native environments.

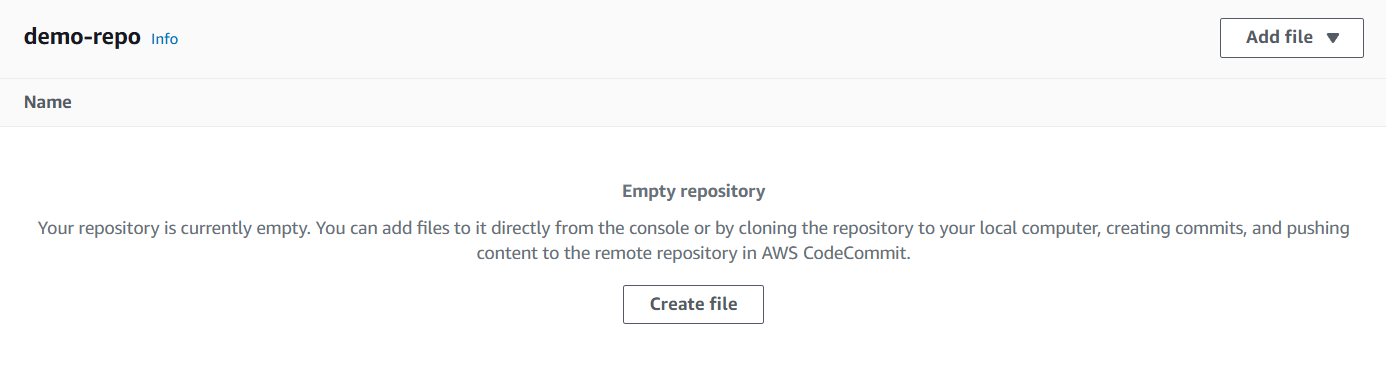
**😄 To begin with the Lab:**

1. So, before building the code build, we need a code commit repository.
2. First, we need to open the code commit and click on Create repository. Then give a name to your repository and click on create.





1. Once our repository is created then we need to create a file for that scroll down to the bottom and click on create the file.



1. Here we need to use this code to create the file. Then give your file a name, the author’s name and email address, and click on commit changes.

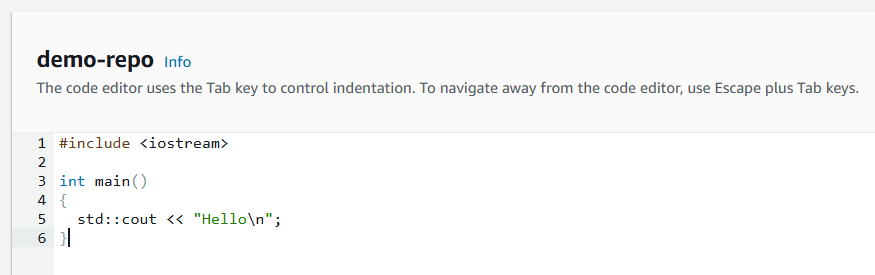
**#include <iostream>**

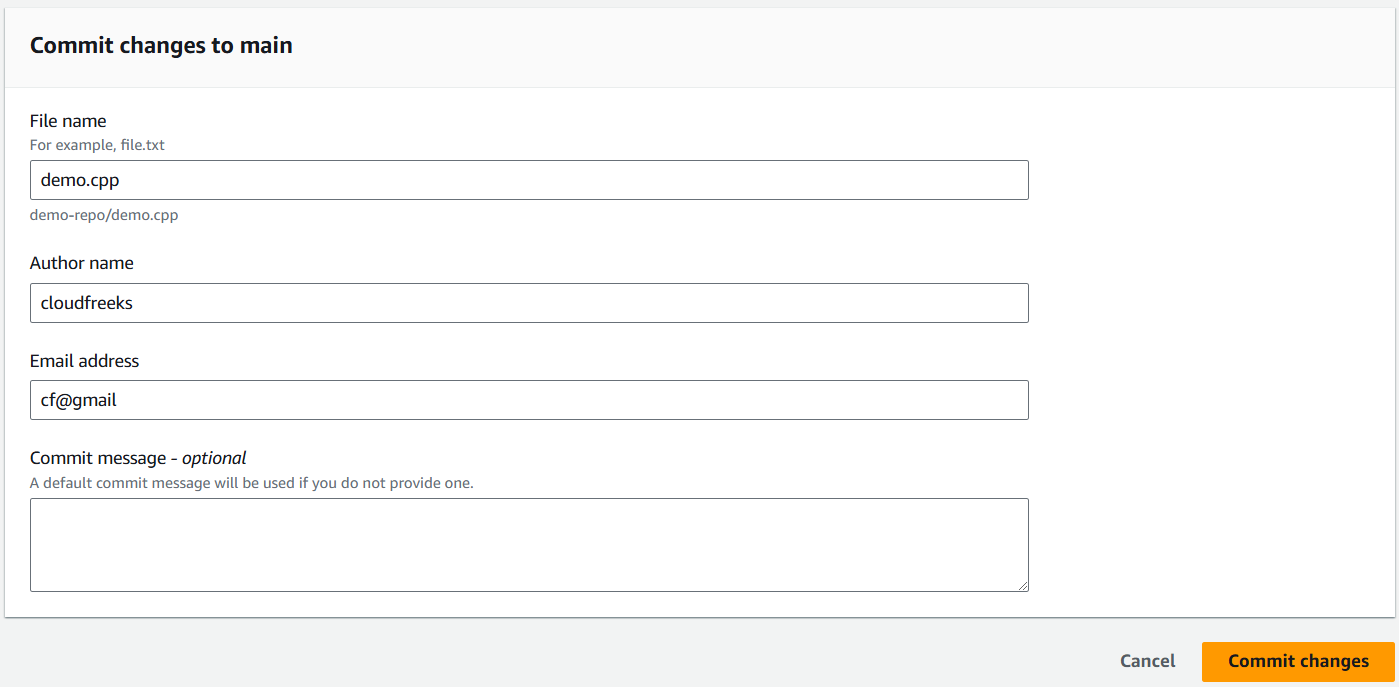
**int main()**

**{**

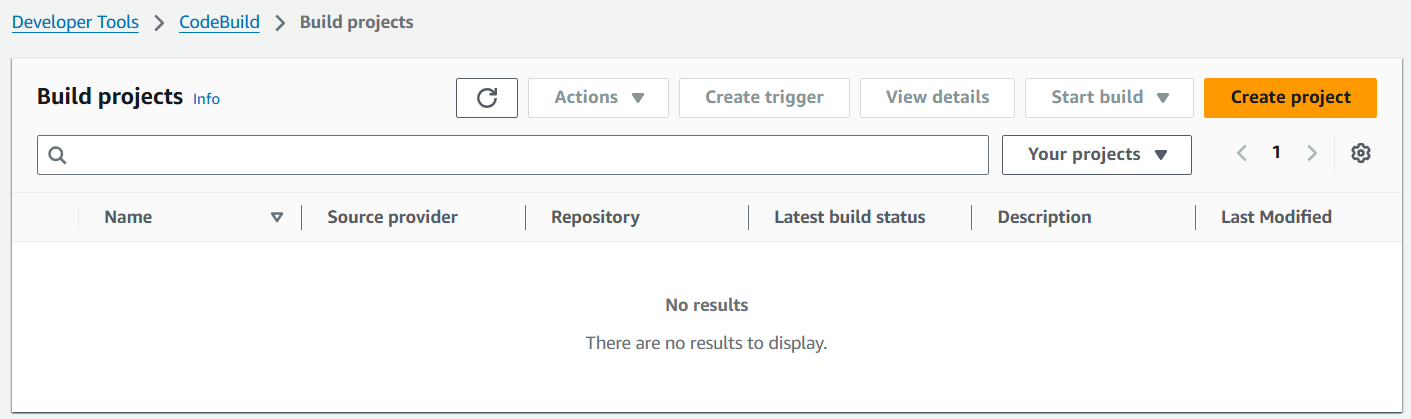
**std::cout << "Hello\n";**

**}**

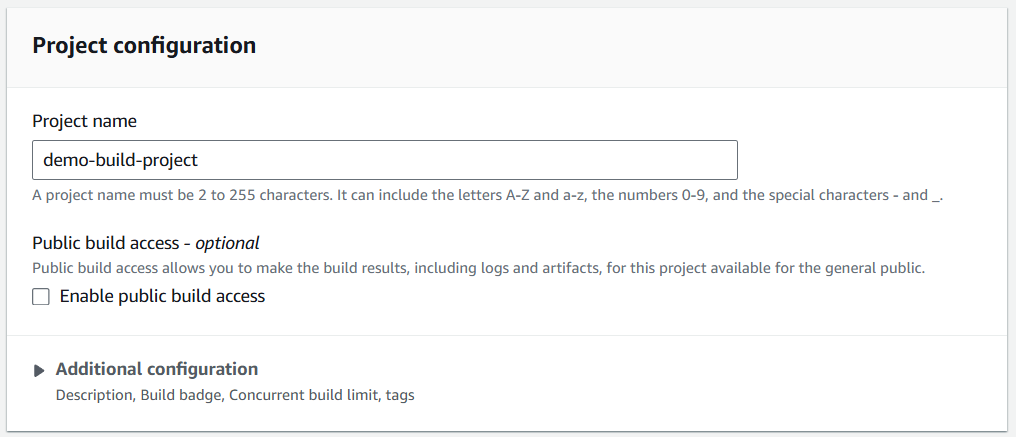




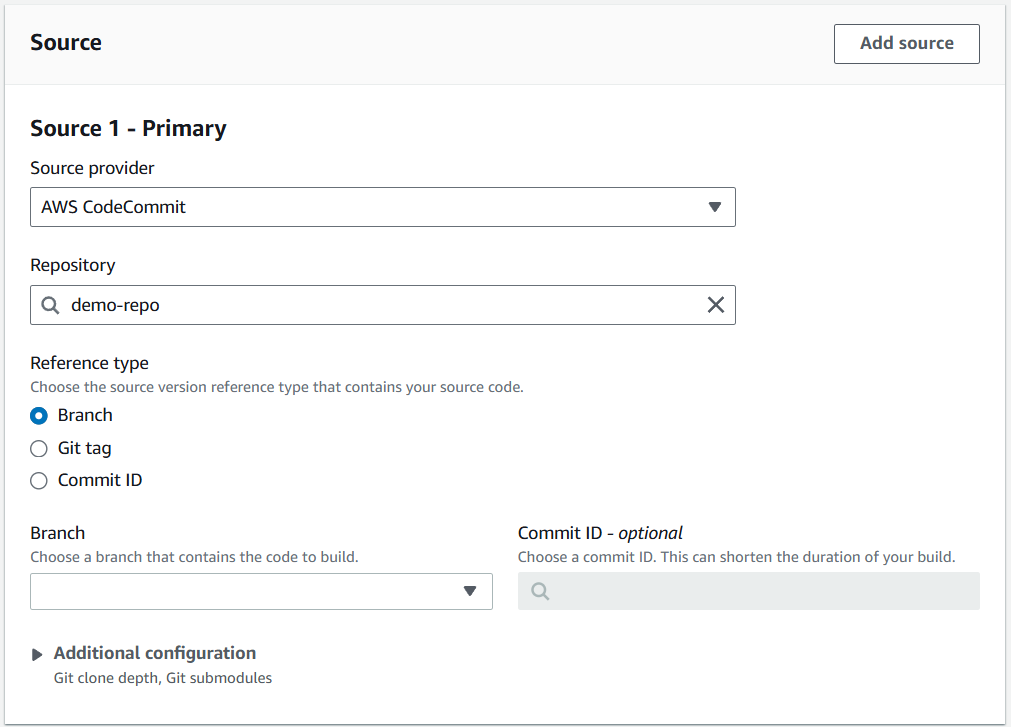
1. Now navigate to code build to build your first project. Click on Create project.



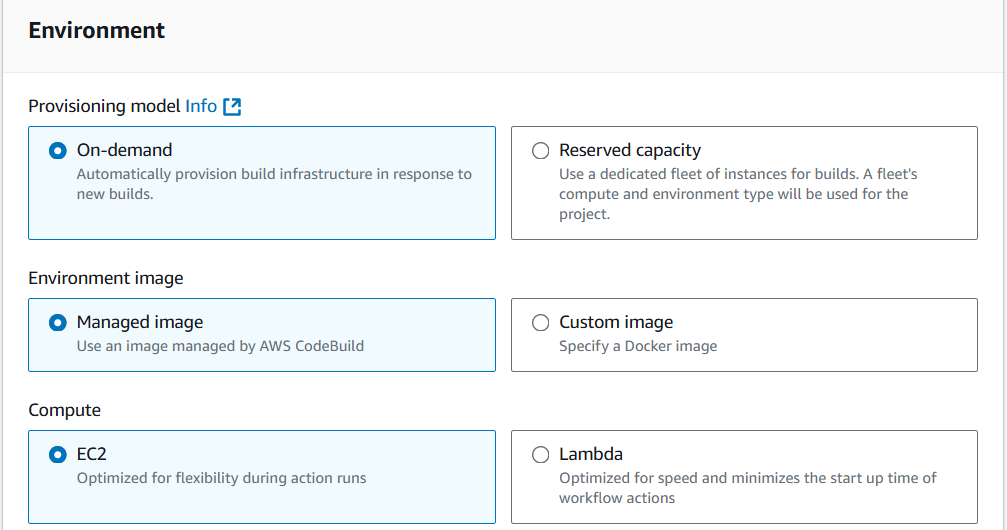
1. We need to give it a project name and scroll down.

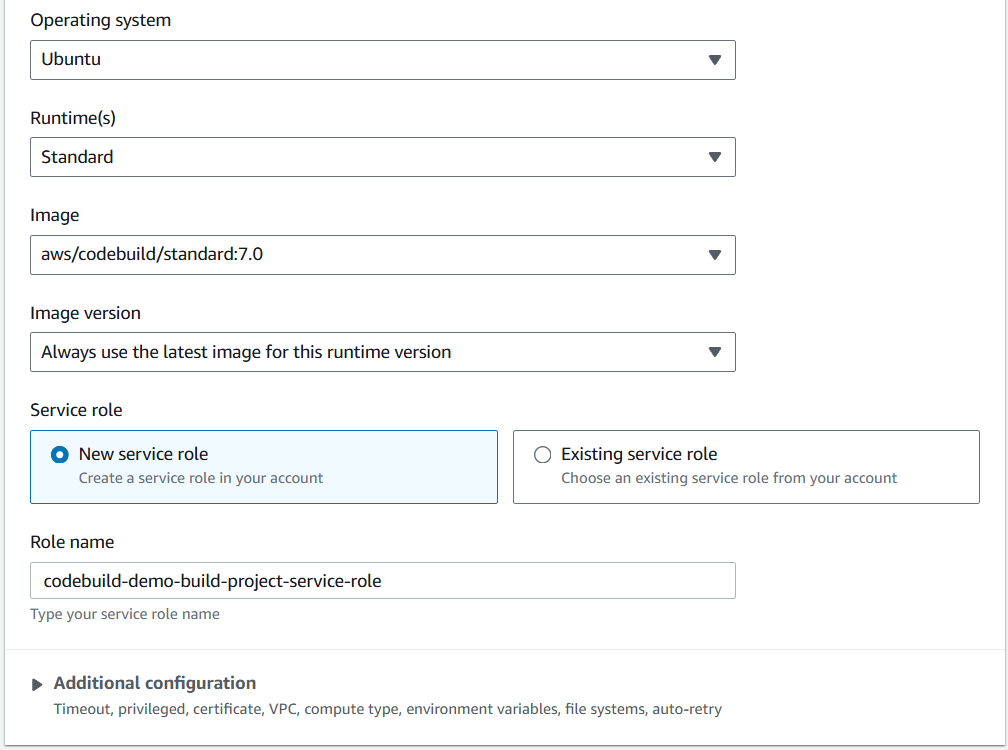


1. In the source choose AWS Code commit as your source provider and repository is the demo repo and scroll down.



1. For the environment choose the same settings as shown below.





1. Then for the build spec file choose to insert the build command and use the code given below.

**version: 0.2**

**phases:**

**build:**

**commands:**

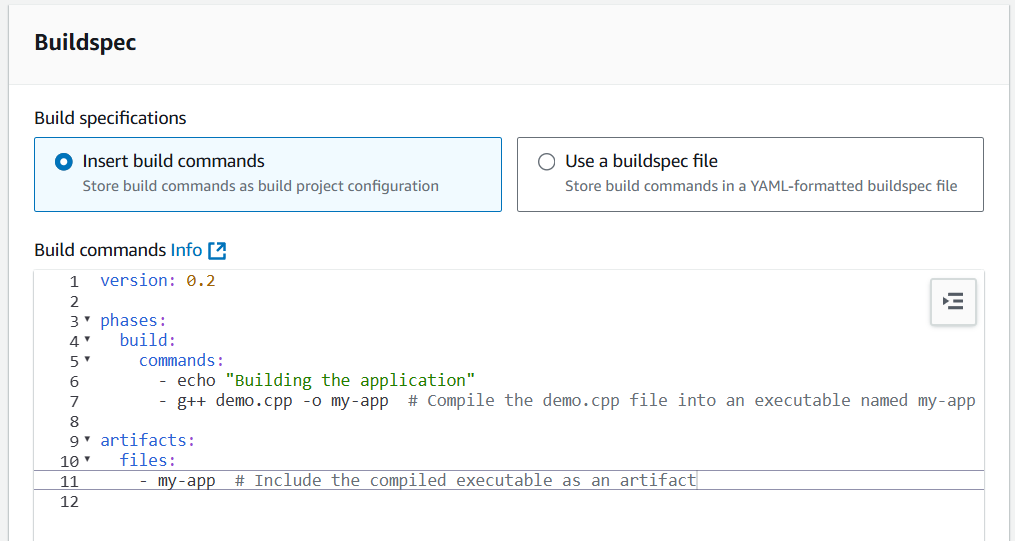
**- echo "Building the application"**

**- g++ demo.cpp -o my-app # Compile the demo.cpp file into an executable named my-app**

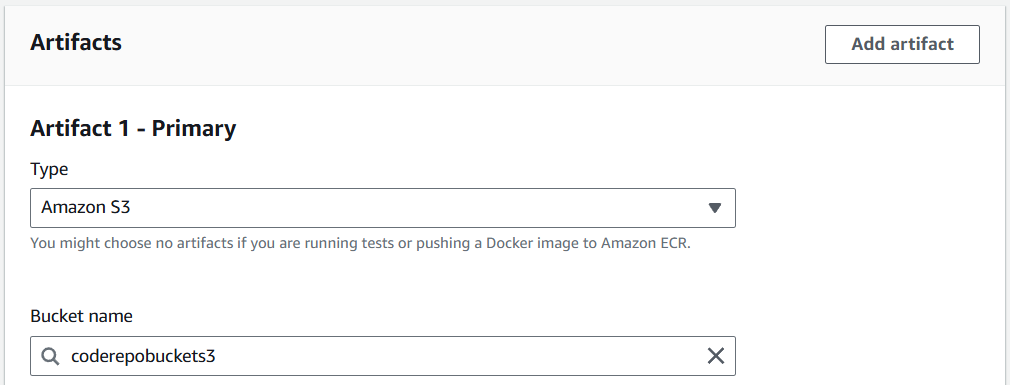
**artifacts:**

**files:**

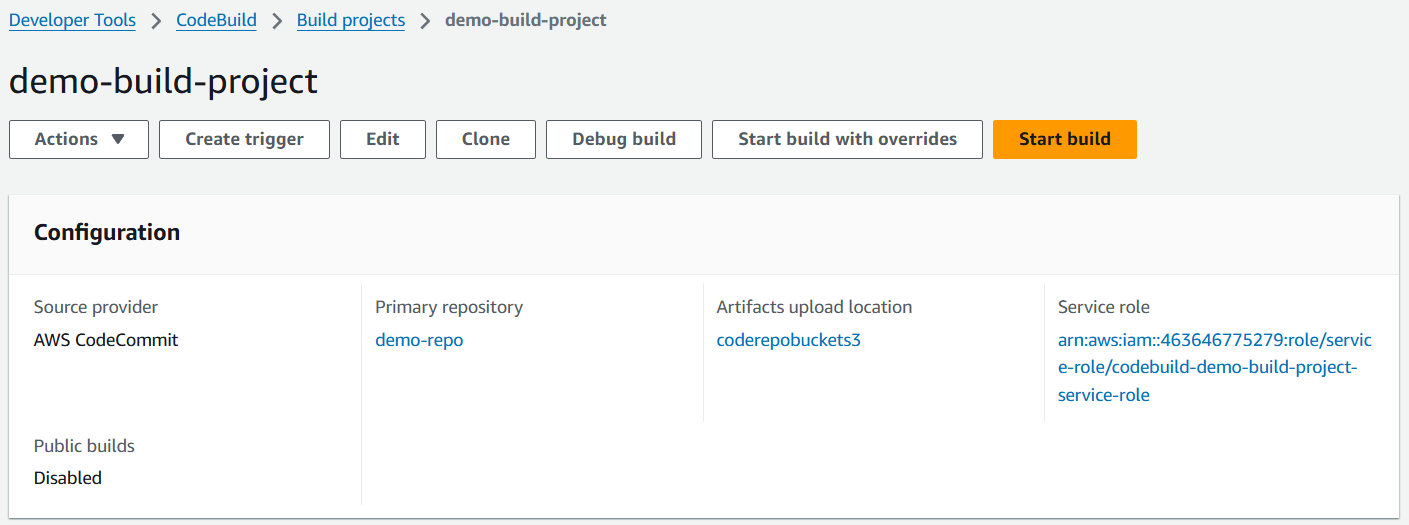
**- my-app # Include the compiled executable as an artifact**



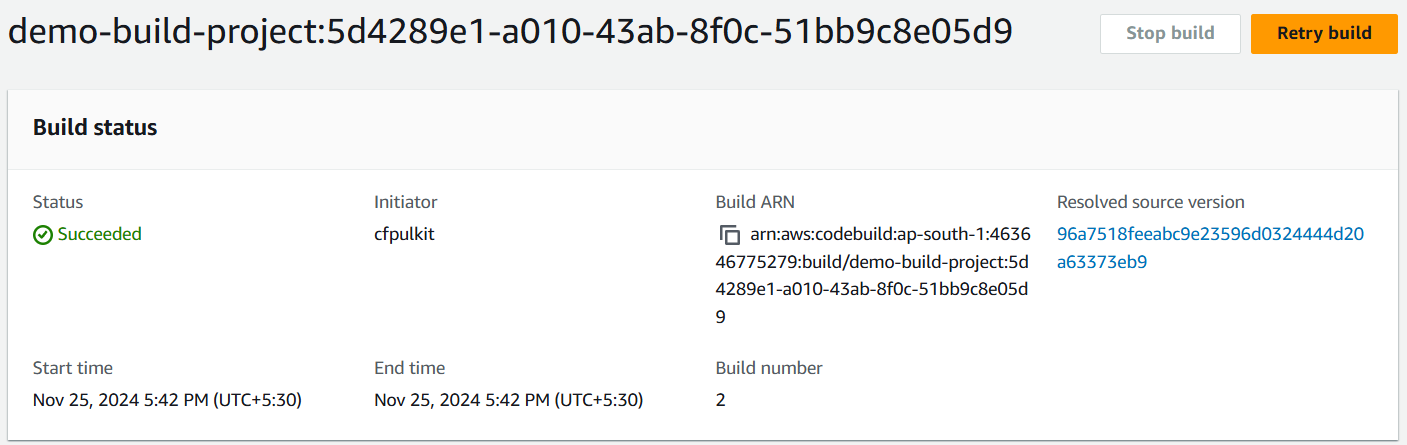
1. Then move to the artifacts here you need to choose amazon S3 and choose a bucket in the same region where you are working.



1. Once it is done scroll down and turn off the cloud watch logs click on Create.
2. Now you need to click on Start build so that you can build your project.



1. Below you can see that our build has been succeeded.



1. Once this is done navigate to S3 and in your bucket which you defined as the artifact you will the binary here.

