



# Building Code with AWS CodeBuild

AWS CodeBuild is a fully managed continuous integration service that helps developers compile source code, run tests, and generate deployable artifacts. It automates the building process, providing a scalable and cost-effective way to ensure code is ready for deployment. Here are the key details:

## Key Features

### 1. Fully Managed Service:

- AWS CodeBuild takes care of provisioning, managing, and scaling the build servers. This means that users don't need to maintain or configure their own infrastructure for building and testing code.

### 2. Build Environment Customization:

- Users can select predefined build environments or create custom ones using Docker images. This flexibility supports various programming languages and tools, making it versatile for different development stacks.

### 3. Scalability:

- CodeBuild scales automatically to handle the build load, enabling it to run multiple builds in parallel. This is especially useful for large teams or projects with frequent code changes.

### 4. Integration with AWS Services:

- CodeBuild integrates seamlessly with other AWS services such as:
  - **AWS CodePipeline:** For automating the CI/CD pipeline.
  - **AWS CodeCommit:** For source control.
  - **AWS CodeDeploy:** For automating application deployments.
  - **Amazon S3:** For storing build artifacts.
- This integration helps create a cohesive workflow from source control to deployment.

### 5. Security:

- AWS CodeBuild offers robust security features, including:
  - **Encryption:** Data can be encrypted both in transit and at rest.
  - **IAM Roles and Policies:** Fine-grained access control can be implemented to restrict who can start builds, view logs, and access other resources.

### 6. Pay-as-You-Go Pricing:

- Users are billed based on the compute resources consumed during the build process, making it a cost-effective solution, particularly for small or sporadic build requirements.

## 7. Build Specifications:

- Builds are defined using a buildspec.yml file, which outlines the phases and commands to be executed during the build. This file can specify environment variables, build commands, and more.

## 8. Logging and Monitoring:

- Detailed logs and metrics are available for each build, which can be accessed via the AWS Management Console, AWS CLI, or programmatically via AWS SDKs. This helps in troubleshooting and optimizing the build process.

## How It Works

### 1. Source Code Fetching:

- CodeBuild retrieves source code from sources like AWS CodeCommit, GitHub, Bitbucket, or Amazon S3.

### 2. Build Execution:

- The build is executed according to the instructions defined in the buildspec.yml file. This includes compiling code, running tests, and generating artifacts.

### 3. Artifact Storage:

- The resulting artifacts are stored in Amazon S3 or can be pushed to a container registry like Amazon ECR.

### 4. Notifications and Reporting:

- Build status and results can be communicated via Amazon CloudWatch, SNS, or other notification services, helping keep teams informed about build outcomes.

## Use Cases

- **Continuous Integration:** Automatically build and test code whenever changes are pushed to the source repository.
- **Automated Testing:** Run unit, integration, and other automated tests to ensure code quality.
- **Multi-Platform Builds:** Compile and test code for different platforms, such as mobile, web, and desktop.

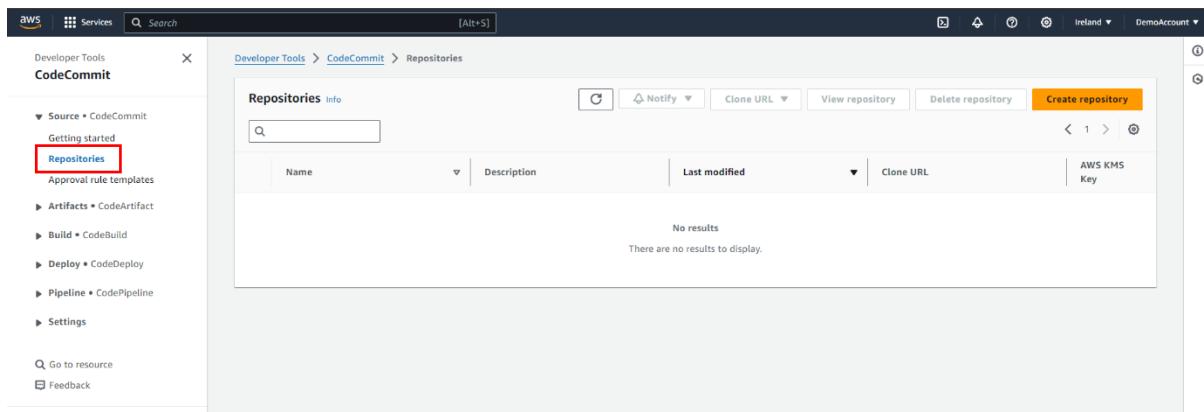
AWS CodeBuild simplifies the process of compiling, testing, and packaging code, allowing developers to focus on writing quality software rather than managing build infrastructure.

**End Goal:** The end goal is to demonstrate how to set up an automated build and deployment pipeline using AWS CodeBuild and CodePipeline. By following the steps provided, you'll create a repository, push code to AWS CodeCommit, configure a buildspec file, set up a CodePipeline, and deploy a static website to an S3 bucket. This process ensures that any changes made to the code are automatically built, tested, and deployed, streamlining development workflows and enhancing productivity.

## 😊 To begin with the Lab:

### 😊 Step 1: Creating Repo and deployment Bucket

1. In your AWS Console, from the AWS Developer tools you need to expand the source and go to Repositories. Now you need to click on Create Repository.



2. Here you just need to give it a name and click on create.

## Repository settings

Repository name  
 100 characters maximum. Other limits apply.

Description - *optional*  
 1,000 characters maximum

Tags

▶ Additional configuration  
AWS KMS key

Enable Amazon CodeGuru Reviewer for Java and Python - *optional*  
Get recommendations to improve the quality of the Java and Python code for all pull requests in this repository.  
A service-linked role will be created in IAM on your behalf if it does not exist.

- Once your repository is created then you need to click on the clone URL and choose HTTPS

Developer Tools > CodeCommit > Repositories > DemoAngularRepo

DemoAngularRepo

▼ Connection steps

HTTPS    SSH    HTTPS (GRC)

**⚠** You are signed in using a root account. You cannot configure SSH connections for a root account, and HTTPS connections for a root account are not recommended. Consider signing in as an IAM user and then setting up your connection.

- Now you must download a zipped folder named my-angular-project from our GitHub. After that unzip the folder and open the whole folder in VS Code.
- When you create an Angular project using Angular CLI, it initializes a local git repository by default. I already made an initial Git commit in our project package, as we did in the previous labs. The only thing we need to do is push it to our new CodeCommit repository. So first, let's set our new CodeCommit repository as the remote repository by typing the 'git remote add origin' command, and pasting the HTTPS URL we cloned from the CodeCommit Console.
- Below you can see that we have cloned our project to our repository

```

● PS C:\Users\PULKIT\Downloads\my-angular-project> git remote add origin https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DemoAngularRepo
● PS C:\Users\PULKIT\Downloads\my-angular-project> git push origin master
Enumerating objects: 69, done.
Counting objects: 100% (69/69), done.
Delta compression using up to 12 threads
Compressing objects: 100% (67/67), done.
Writing objects: 100% (69/69), 255.66 KiB | 8.25 MiB/s, done.
Total 69 (delta 22), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DemoAngularRepo
 * [new branch]      master -> master
o PS C:\Users\PULKIT\Downloads\my-angular-project>

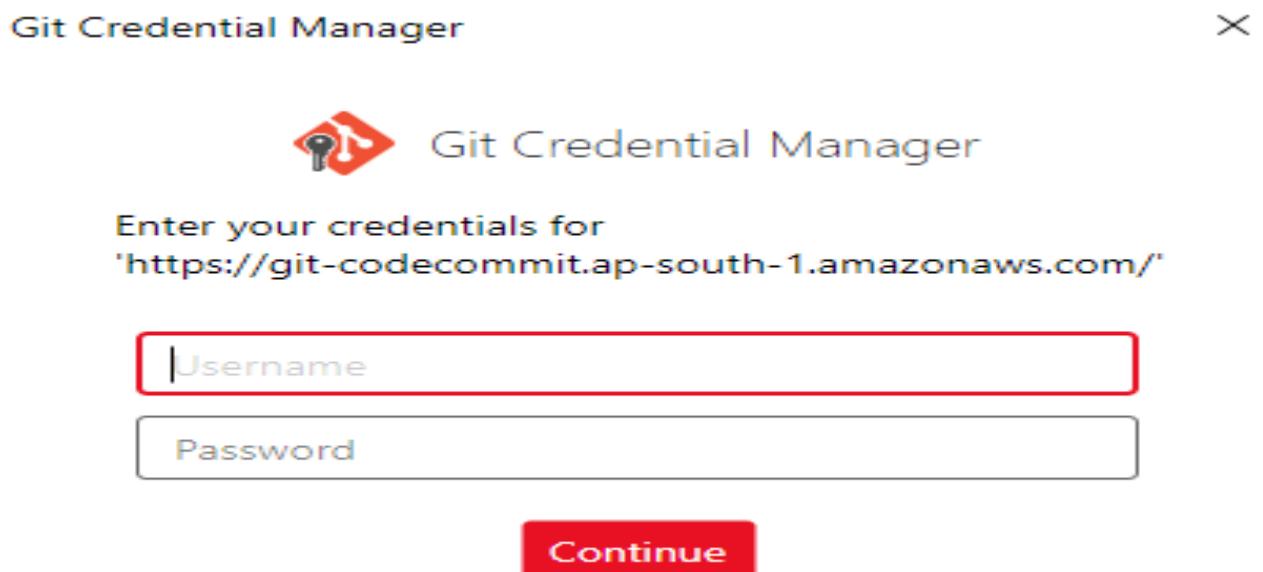
```

- Now if you go to your repository and refresh the page you will see all your project data here.

The screenshot shows the AWS CodeCommit interface. On the left, there's a sidebar with 'Developer Tools' and 'CodeCommit' selected. Under 'CodeCommit', there are sections for 'Source', 'Code', 'Artifacts', 'Build', 'Deploy', 'Pipeline', and 'Settings'. Below these are 'Go to resource' and 'Feedback' links. The main area displays the 'DemoAngularRepo' repository. At the top, it shows the path 'Developer Tools > CodeCommit > Repositories > DemoAngularRepo'. Below that is a 'Reference' section with 'Notify' set to 'master', 'Create pull request', and 'Clone URL'. The central part of the screen shows the repository's contents under 'DemoAngularRepo info'. It lists files and folders: e2e, src, .browserslistrc, .editorconfig, .gitignore, angular.json, karma.conf.js, package-lock.json, package.json, README.md, tsconfig.app.json, tsconfig.json, and tsconfig.json snar icon. There are also 'Add file' and 'Edit file' buttons.

- Also, if you get this type of error while cloning the project to the repository, you can visit the link below.

[https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html?icmpid=docs\\_acc\\_console\\_connect\\_np](https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-gc.html?icmpid=docs_acc_console_connect_np)



9. Let's create a new bucket for static website hosting as before. We will use it to deploy our sample website after building with CodeBuild. We are creating this bucket in the 'Europe (Ireland)' region.
10. So, basically, we are going to create a public bucket, click on create bucket give it a unique name, and turn off block public access, then create your bucket.

**Create bucket** Info

Buckets are containers for data stored in S3.

**General configuration**

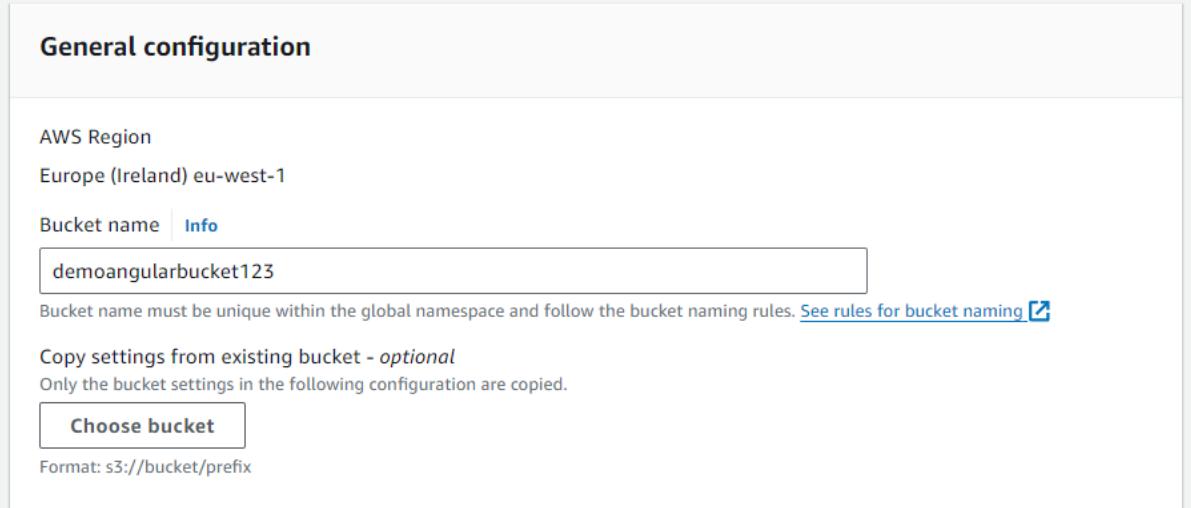
AWS Region  
Europe (Ireland) eu-west-1

Bucket name Info

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#) ↗

Copy settings from existing bucket - *optional*  
Only the bucket settings in the following configuration are copied.  
[Choose bucket](#)

Format: s3://bucket/prefix



## Block Public Access settings for this bucket

Public access is granted to buckets and objects through access control lists (ACLs), bucket policies, access point policies, or all. In order to ensure that public access to this bucket and its objects is blocked, turn on Block all public access. These settings apply only to this bucket and its access points. AWS recommends that you turn on Block all public access, but before applying any of these settings, ensure that your applications will work correctly without public access. If you require some level of public access to this bucket or objects within, you can customize the individual settings below to suit your specific storage use cases. [Learn more](#)

### **Block all public access**

Turning this setting on is the same as turning on all four settings below. Each of the following settings are independent of one another.

#### **Block public access to buckets and objects granted through new access control lists (ACLs)**

S3 will block public access permissions applied to newly added buckets or objects, and prevent the creation of new public access ACLs for existing buckets and objects. This setting doesn't change any existing permissions that allow public access to S3 resources using ACLs.

#### **Block public access to buckets and objects granted through any access control lists (ACLs)**

S3 will ignore all ACLs that grant public access to buckets and objects.

#### **Block public access to buckets and objects granted through new public bucket or access point policies**

S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.

#### **Block public and cross-account access to buckets and objects through any public bucket or access point policies**

S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.



#### Turning off block all public access might result in this bucket and the objects within becoming public

AWS recommends that you turn on block all public access, unless public access is required for specific and verified use cases such as static website hosting.

I acknowledge that the current settings might result in this bucket and the objects within becoming public.

11. Once your bucket is created now should go to bucket policies and update its policy.

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "PublicReadGetObject",  
      "Effect": "Allow",  
      "Principal": "*",  
      "Action": [  
        "s3:GetObject"  
      ],  
      "Resource": [  
        "arn:aws:s3:::YOUR-BUCKET-NAME/*"  
      ]  
    }  
  ]  
}
```

12. After that you need to enable static website hosting. Now when you host a single-page application, like Angular, on an S3 bucket, you need to configure both index and error document settings as 'index.html', the main page of your app.

The screenshot shows the 'Edit static website hosting' configuration page for an S3 bucket. It includes sections for enabling static website hosting, setting the hosting type (Host a static website), and defining index and error documents. A note about making content publicly readable is also present.

**Static website hosting**  
Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting  
 Disable  
 Enable

Hosting type  
 Host a static website  
Use the bucket endpoint as the web address. [Learn more](#)  
 Redirect requests for an object  
Redirect requests to another bucket or domain. [Learn more](#)

**Index document**  
Specify the home or default page of the website.  
index.html

**Error document - optional**  
This is returned when an error occurs.  
index.html

**Static website hosting**  
Use this bucket to host a website or redirect requests. [Learn more](#)

Static website hosting  
Enabled

Hosting type  
Bucket hosting

Bucket website endpoint  
When you configure your bucket as a static website, the website is available at the AWS Region-specific website endpoint of the bucket. [Learn more](#)  
http://[demoangularbucket123.s3-website-eu-west-1.amazonaws.com](http://demoangularbucket123.s3-website-eu-west-1.amazonaws.com)

## Step 2: Create a pipeline with the build stage

1. Now go to code pipeline and click on create pipeline. Now you need to give your pipeline a name and then move to the next step.

Developer Tools > [CodePipeline](#) > [Pipelines](#) > Create new pipeline

Step 1  
**Choose pipeline settings**  
Step 1 of 5

Step 2  
Add source stage

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

## Pipeline settings

**Pipeline name**  
Enter the pipeline name. You cannot edit the pipeline name after it is created.  
 No more than 100 characters

**Pipeline type**

Info You can no longer create V1 pipelines through the console. We recommend you use the V2 pipeline type with improved release safety, pipeline triggers, parameterized pipelines, and a new billing model.

**Execution mode**  
Choose the execution mode for your pipeline. This determines how the pipeline is run.

Superseded  
A more recent execution can overtake an older one. This is the default.

Queued (Pipeline type V2 required)  
Executions are processed one by one in the order that they are queued.

Parallel (Pipeline type V2 required)  
Executions don't wait for other runs to complete before starting or finishing.

2. In step 2 we need to add the source provider, which is AWS code commit, choose it, and then select the repository name and choose your branch after that keep everything to default and move to the next step.

Step 2  
**Add source stage**

Step 3  
Add build stage

Step 4  
Add deploy stage

Step 5  
Review

## Source

**Source provider**  
This is where you stored your input artifacts for your pipeline. Choose the provider and then provide the connection details.

**Repository name**  
Choose a repository that you have already created where you have pushed your source code.  
 X

**Branch name**  
Choose a branch of the repository  
 X  
**master**

Choose a detection mode to automatically start your pipeline when a change occurs in the source code.

Amazon CloudWatch Events (recommended)  
Use Amazon CloudWatch Events to automatically start my pipeline when a change occurs

AWS CodePipeline  
Use AWS CodePipeline to check periodically for changes

**Output artifact format**  
Choose the output artifact format.

CodePipeline default  
AWS CodePipeline uses the default zip format for artifacts in the pipeline.  
Does not include Git metadata about the repository

Full clone  
AWS CodePipeline passes metadata about the repository that allows subsequent actions to do a full Git clone. Only supported for AWS CodeBuild actions.

3. In step 3 you need to add a build stage, for that first you need to choose your build provider which is AWS Code build. Then choose your region, you need to create a new project for that click on Create a project and it will take you to a new tab.

Step 1  
Choose pipeline settings

Step 2  
Add source stage

Step 3  
**Add build stage**

Step 4  
Add deploy stage

Step 5  
Review

## Add build stage Info

Step 3 of 5

**Build - optional**

**Build provider**  
This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

**Region**  
Europe (Ireland)

**Project name**  
Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

or **Create project**

**Environment variables - optional**  
Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

**Add environment variable**

**Build type**

Single build  
Triggers a single build.

Batch build  
Triggers multiple builds as a single execution.

4. So, here first you need to give a project name and scroll down.

Developer Tools > CodeBuild > Build projects > Create build project

**Continue to CodePipeline**  
Create a new CodeBuild build project and return to CodePipeline to finish configuring your pipeline.

## Create build project

**Project configuration**

**Project name**  
  
A project name must be 2 to 255 characters. It can include the letters A-Z and a-z, the numbers 0-9, and the special characters - and \_.

**► Additional configuration**  
Description, Build badge, Concurrent build limit, tags

5. Now for the environment choose the same settings as shown below.

## Environment

Provisioning model [Info](#)

On-demand

Automatically provision build infrastructure in response to new builds.

Reserved capacity

Use a dedicated fleet of instances for builds. A fleet's compute and environment type will be used for the project.

Environment image

Managed image

Use an image managed by AWS CodeBuild

Custom image

Specify a Docker image

Compute

EC2

Optimized for flexibility during action runs

Lambda

Optimized for speed and minimizes the start up time of workflow actions

Operating system

Ubuntu

Runtime(s)

Standard

Image

aws/codebuild/standard:7.0

Image version

Always use the latest image for this runtime version

Use GPU-enhanced compute

- After that choose to create a new service role and give it a role name then scroll down and keep everything to default and click on continue to pipeline.

Service role

New service role

Create a service role in your account

Existing service role

Choose an existing service role from your account

Role name

demo-angularservicerole

Type your service role name

► Additional configuration

Timeout, privileged, certificate, VPC, compute type, environment variables, file systems

- Also in the buildspec choose to use a buildspec file.

## Buildspec

### Build specifications

#### Insert build commands

Store build commands as build project configuration

#### Use a buildspec file

Store build commands in a YAML-formatted buildspec file

### Buildspec name - *optional*

By default, CodeBuild looks for a file named buildspec.yml in the source code root directory. If your buildspec file uses a different name or location, enter its path from the source root here (for example, buildspec-two.yml or configuration/buildspec.yml).

buildspec.yml

8. Below you can see that we can see our project name now click on next.

## Build - *optional*

### Build provider

This is the tool of your build project. Provide build artifact details like operating system, build spec file, and output file names.

AWS CodeBuild

### Region

Europe (Ireland)

### Project name

Choose a build project that you have already created in the AWS CodeBuild console. Or create a build project in the AWS CodeBuild console and then return to this task.

demoangularproject



or

[Create project](#)

### Environment variables - *optional*

Choose the key, value, and type for your CodeBuild environment variables. In the value field, you can reference variables generated by CodePipeline. [Learn more](#)

[Add environment variable](#)

### Build type

#### Single build

Triggers a single build.

#### Batch build

Triggers multiple builds as a single execution.

[Cancel](#)

[Previous](#)

[Skip build stage](#)

[Next](#)

9. Now in step 4 you need to choose where you want to deploy your project, so we need to choose Amazon S3 and then choose our bucket then choose to extract file before deploy and move to review page then click to create your pipeline.

Developer Tools > CodePipeline > Pipelines > Create new pipeline

Step 1 Choose pipeline settings

Step 2 Add source stage

Step 3 Add build stage

Step 4 Add deploy stage

Step 5 Review

## Add deploy stage Info

Step 4 of 5

### Deploy - optional

Deploy provider  
Choose how you deploy to instances. Choose the provider, and then provide the configuration details for that provider.

Amazon S3

Region  
Europe (Ireland)

Bucket  
demoangularbucket123

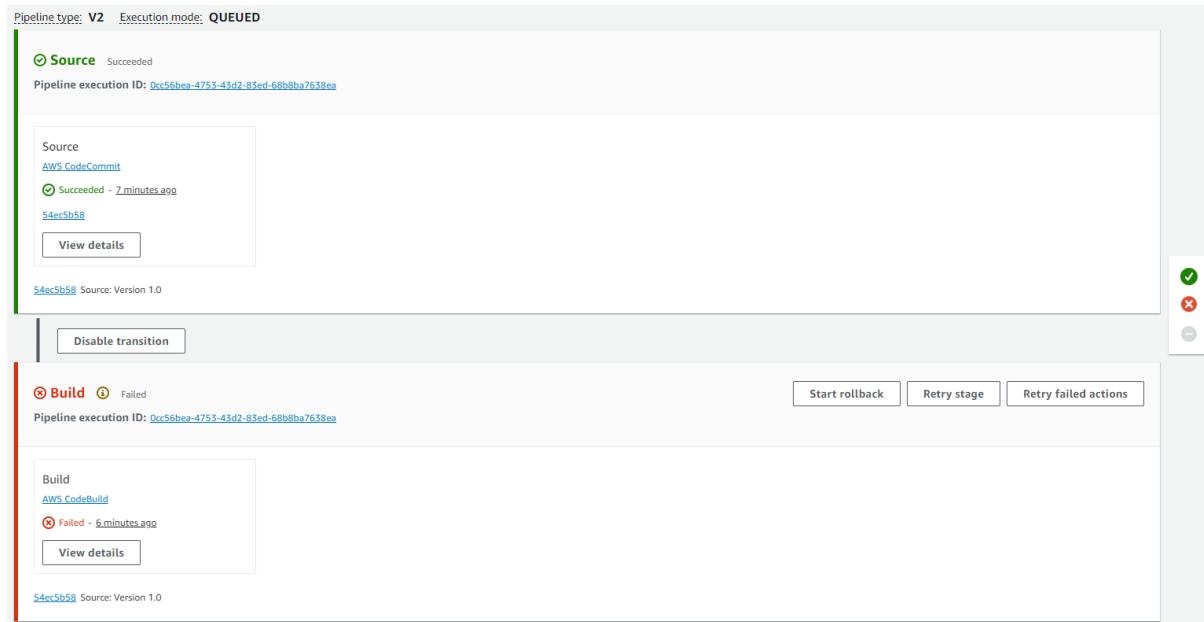
Deployment path - optional

Extract file before deploy  
The deployed artifact will be unzipped before deployment.

**► Additional configuration**

Configure automatic rollback on stage failure

10. Below you can see that we created our pipeline and in the source stage it was successfully executed but during the build stage, it failed.



11. So, to check why it happened we need to click on the view details option available in the build stage. Here you can see that the failure was related to the YAML file.

Action execution details

Action name: Build Status: Failed

**Summary** | **Logs** | **Input**

Status Last updated  
✖ Failed 8 minutes ago

Action execution ID  
2d8b4b83-781b-48ab-b70c-874f54c688ed

Error code  
Action execution failed

Error message  
Build terminated with state: FAILED

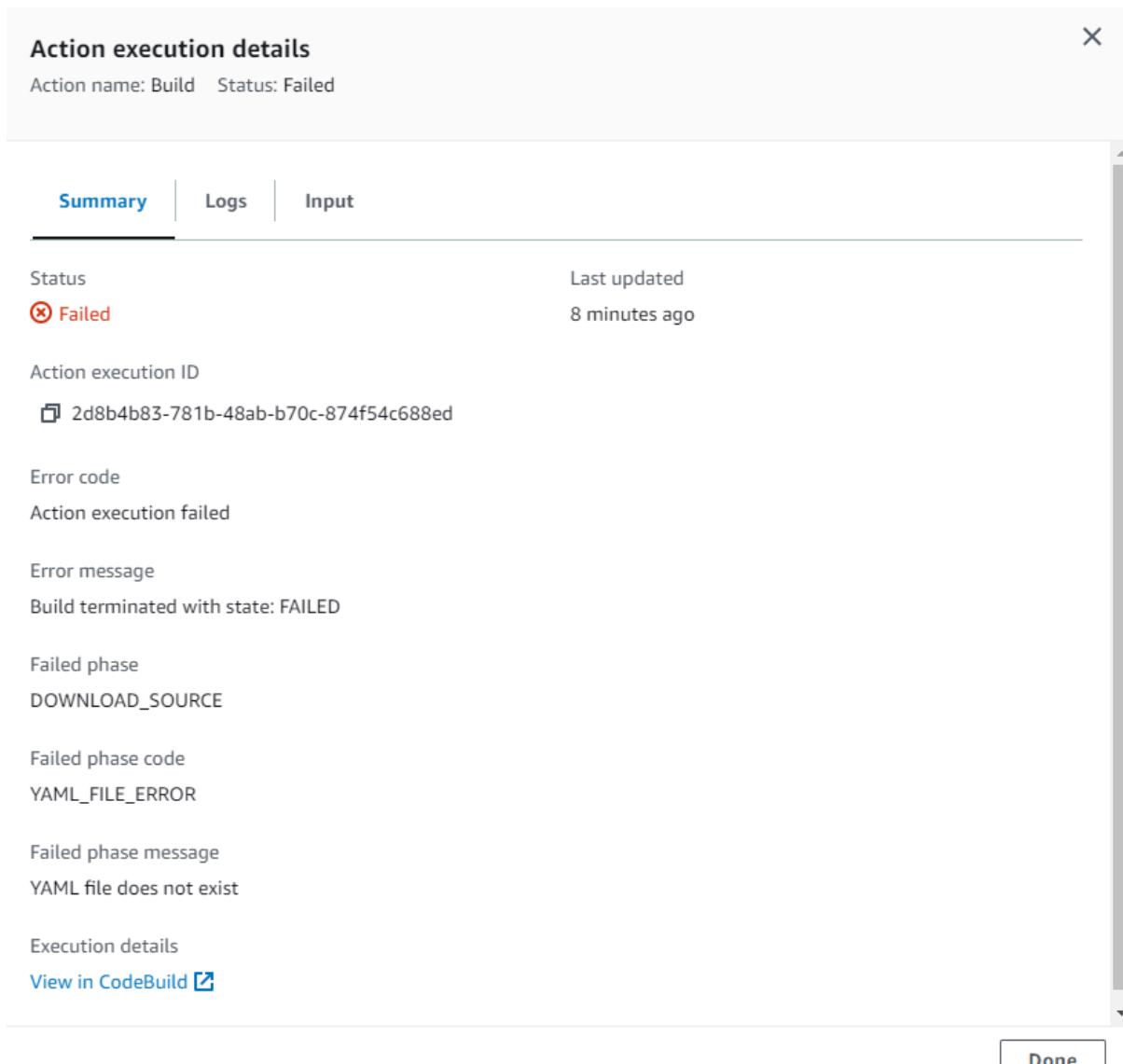
Failed phase  
DOWNLOAD\_SOURCE

Failed phase code  
YAML\_FILE\_ERROR

Failed phase message  
YAML file does not exist

Execution details  
[View in CodeBuild](#)

**Done**



12. Also, if you go to the logs section here you can see the context of failure.

**Summary** | **Logs** | **Input**

✖ Failed Start time: 8 minutes ago

Showing the last 7 lines of the build log. [View entire log](#)

[▲ Show previous logs](#)

```
1 [Container] 2024/08/07 08:34:47.391928 Running on CodeBuild On-demand
2 [Container] 2024/08/07 08:34:47.391944 Waiting for agent ping
3 [Container] 2024/08/07 08:34:47.492958 Waiting for DOWNLOAD_SOURCE
4 [Container] 2024/08/07 08:34:48.743679 Phase is DOWNLOAD_SOURCE
5 [Container] 2024/08/07 08:34:48.744783 CODEBUILD_SRC_DIR=/codebuild/output/src1976837845/src
6 [Container] 2024/08/07 08:34:48.752483 Phase complete: DOWNLOAD_SOURCE State: FAILED
7 [Container] 2024/08/07 08:34:48.752505 Phase context status code: YAML_FILE_ERROR Message: YAML file does not exist
8
```

13. This happens because CodeBuild looks for a buildspec file in our source code, which we have not yet provided. Hence, it doesn't know how to build the code.



# Creating Buildspec file

A build specification (build spec) file is a configuration file that defines how a software build process should be carried out. It contains instructions and parameters that guide the build system or tool in compiling and packaging the software application. Build spec files are used in various build automation tools and systems to streamline and standardize the build process.

## Key Components of a Build Spec File

### 1. Environment Setup:

- **Dependencies:** Specifies libraries, frameworks, and other dependencies needed for the build.
- **Environment Variables:** Defines any required environment variables.

### 2. Build Instructions:

- **Commands:** Specifies the sequence of commands to compile, test, and package the software.
- **Scripts:** References scripts that need to be executed during the build process.

### 3. Source Code:

- **Source Directories:** Defines where the source code is located.
- **Inclusions:** Lists files or directories to be included in the build.

### 4. Build Artifacts:

- **Output Directories:** Specifies where the build outputs (artifacts) should be stored.
- **Artifact Management:** Defines how artifacts are managed, such as versioning and storage.

### 5. Build Triggers:

- **Conditions:** Specifies conditions under which the build should be triggered (e.g., code commits, schedule).



## To begin with the Lab:

1. First you need to download a zipped folder from our GitHub named my-angular-project\_final. Then extract it and open it in VS Code.
2. So, in this project you will find a YAML file named buildspec.yaml. So, in our previous lab, our build stage failed because we didn't have this file in our source code.

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with project files: e2e, src, .browserslistrc, .editorconfig, .gitignore, angular.json, buildspec.yml, karma.conf.js, package-lock.json, package.json, README.md, tsconfig.app.json, tsconfig.json, tsconfig.spec.json, and tslint.json. The buildspec.yml file is selected and highlighted with a red box. The main pane displays the YAML content of the buildspec.yml file:

```
version: 0.2
phases:
  install:
    runtime-versions:
      nodejs: 20
    commands:
      - npm install -g @angular/cli@17
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - ng build -c production
  artifacts:
    base-directory: dist/my-angular-project
    files:
      - '**/*'
```

3. Now you need to copy the content from this file and create a similar file in the previous project which we had uploaded to our repository.
4. Once you have done that then you need to open the terminal and run the git status command to check what needs to be committed. After that run the git add command.

**git status**

**git add .**

The screenshot shows a code editor interface with a dark theme. On the left is a sidebar with project files: e2e, src, .browserslistrc, .editorconfig, .gitignore, angular.json, buildspec.yml, karma.conf.js, package-lock.json, package.json, README.md, tsconfig.app.json, tsconfig.json, tsconfig.spec.json, and tslint.json. The buildspec.yml file is selected and highlighted with a red box. The main pane displays the YAML content of the buildspec.yml file. Below the editor is a terminal window with the following output:

```
PS C:\Users\PULKIT\Downloads\my-angular-project> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    buildspec.yml

nothing added to commit but untracked files present (use "git add" to track)
PS C:\Users\PULKIT\Downloads\my-angular-project>
```

5. Below you can see that our new file has been staged now we need to commit our changes and push the updates to our repository.

PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    POSTMAN CONSOLE    AZURE

```
PS C:\Users\PULKIT\Downloads\my-angular-project> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    buildspec.yml

nothing added to commit but untracked files present (use "git add" to track)
● PS C:\Users\PULKIT\Downloads\my-angular-project> git add .
● PS C:\Users\PULKIT\Downloads\my-angular-project> git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   buildspec.yml

○ PS C:\Users\PULKIT\Downloads\my-angular-project>
```

- For that you need to run the below command.

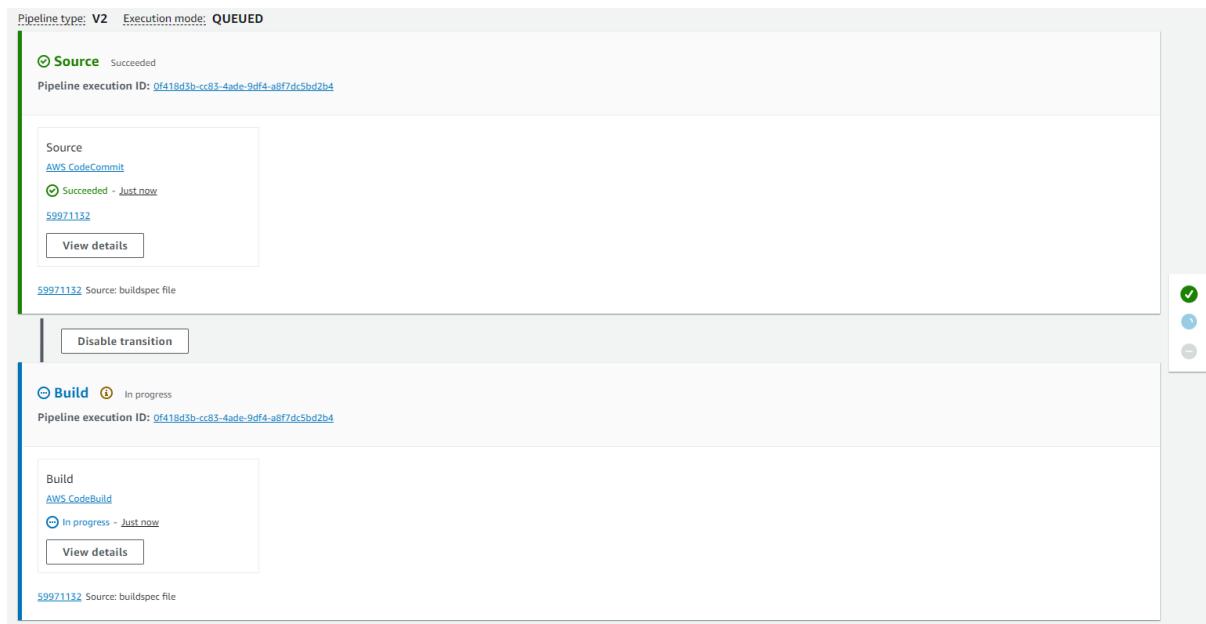
**git commit -m “buildspec file”**  
**git push origin master.**

- Below you can see that we have committed our file and pushed it onto our repository.

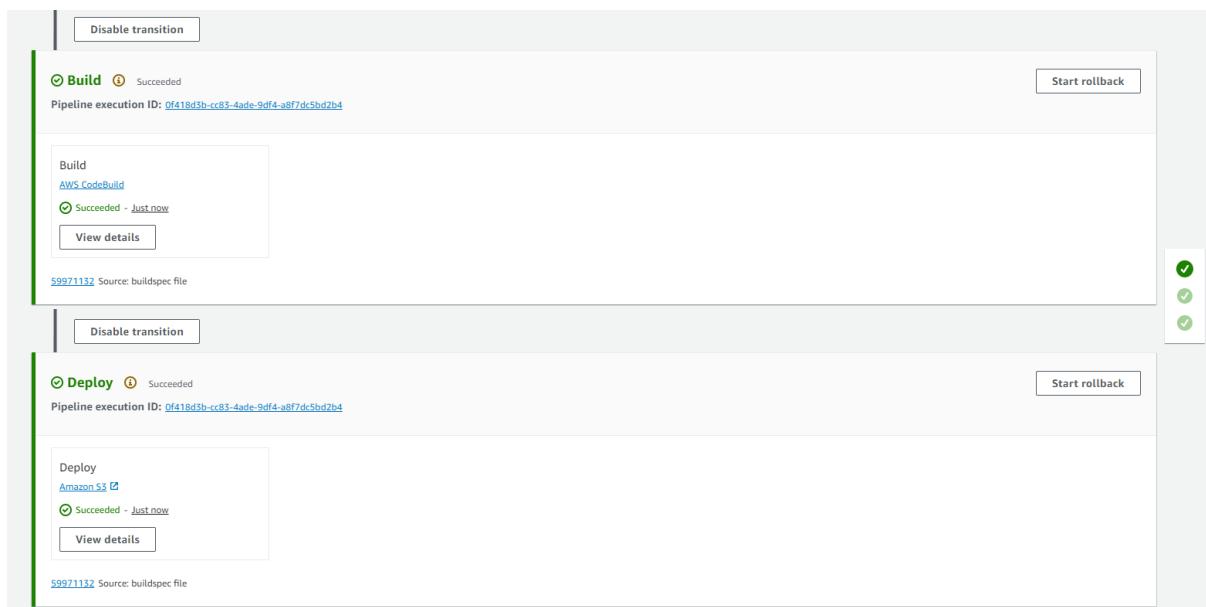
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL    PORTS    POSTMAN CONSOLE    AZURE

```
PS C:\Users\PULKIT\Downloads\my-angular-project> git commit -m "buildspec file"
[master 5997113] buildspec file
 1 file changed, 17 insertions(+)
 create mode 100644 buildspec.yml
● PS C:\Users\PULKIT\Downloads\my-angular-project> git push origin master
Enumerating objects: 4, done.
Counting objects: 100% (4/4), done.
Delta compression using up to 12 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 452 bytes | 452.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Validating objects: 100%
To https://git-codecommit.eu-west-1.amazonaws.com/v1/repos/DemoAngularRepo
 54ec5b..5997113  master -> master
○ PS C:\Users\PULKIT\Downloads\my-angular-project>
```

- Now if you go back to the code pipeline console you will see that our pipeline has started running and you can see that our build stage is in progress. You can also see the same commit message in the build stage. This build stage will take some time.



9. And below you can see that our pipeline has been executed successfully. All three stages has been executed. Source, build and deploy.



10. Now if you go to the S3 bucket you will see that the objects have been added here from the repository. Now go to properties scroll down to the bottom and open your static website link.

Amazon S3 > Buckets > demoangularbucket123						
demoangularbucket123 <a href="#">Info</a>						
<a href="#">Objects</a> <a href="#">Properties</a> <a href="#">Permissions</a> <a href="#">Metrics</a> <a href="#">Management</a> <a href="#">Access Points</a>						
<b>Objects (7) <a href="#">Info</a></b>						
Objects are the fundamental entities stored in Amazon S3. You can use <a href="#">Amazon S3 inventory</a> to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. <a href="#">Learn more</a>						
<input type="text" value="Find objects by prefix"/> <span style="float: right;">&lt; 1 &gt; </span>						
Name	Type	Last modified	Size	Storage class		
<a href="#">3rdpartylicenses.txt</a>	txt	August 7, 2024, 14:49:58 (UTC+05:45)	14.2 KB	Standard		
<a href="#">favicon.ico</a>	ico	August 7, 2024, 14:49:58 (UTC+05:45)	948.0 B	Standard		
<a href="#">index.html</a>	html	August 7, 2024, 14:49:58 (UTC+05:45)	5.0 KB	Standard		
<a href="#">main.94328ed64d2afe53.js</a>	js	August 7, 2024, 14:49:58 (UTC+05:45)	271.4 KB	Standard		
<a href="#">polyfills.f35e31f90dcc29a9.js</a>	js	August 7, 2024, 14:49:58 (UTC+05:45)	33.6 KB	Standard		
<a href="#">runtime.21abf645c46126d.js</a>	js	August 7, 2024, 14:49:58 (UTC+05:45)	1.1 KB	Standard		
<a href="#">styles.4af885ee8aa244ff.css</a>	css	August 7, 2024, 14:49:58 (UTC+05:45)	245.9 KB	Standard		

11. Below you can see that our static website is working perfectly.

## Sample Angular App for AWS CodePipeline Step by Step

Version: 1.0

Congratulations! You successfully built and deployed your code.

This is a simple single-page calculator app developed with Angular and Bootstrap for the build examples on the [AWS CodePipeline Step by Step](#) course.

### Simple Calculator

<input type="text" value="0"/>	<input type="text" value="Please select an operator"/>	<input type="text" value="Your second input"/>
		<input type="button" value="Clear"/> <input type="button" value="Calculate"/>

## Sample Angular App for AWS CodePipeline Step by Step

Version: 1.0

Congratulations! You successfully built and deployed your code.

This is a simple single-page calculator app developed with Angular and Bootstrap for the build examples on the [AWS CodePipeline Step by Step](#) course.

### Simple Calculator

<input type="text" value="5"/>	<input style="width: 40px; height: 40px; vertical-align: middle; text-align: center; font-size: 2em; line-height: 1.2; border: none; padding: 0; margin: 0;" type="text" value="+"/>	<input type="text" value="8"/>
<input type="button" value="Clear"/> <input type="button" value="Calculate"/>		

**Result:** 13