CODEPIPELINE

Introduction

Welcome to a comprehensive guide on Amazon Web Services (AWS) CodePipeline! This documentation will take you through the basics of CodePipeline and its integration with CodeCommit and CodeBuild. These tools collectively orchestrate continuous integration and delivery, providing a streamlined approach to software delivery on AWS.

In these documentation you will understand what CodePipeline is all about. Diving in-dept to it you will also get to know what CodeCommit, CodeBuild and CodeDeploy are and how they work with CodePipeline .

Amazon Web Services (AWS) CodePipeline orchestrates continuous integration and delivery, integrating CodeCommit, CodeBuild, and CodeDeploy to streamline software delivery.

what we will understand

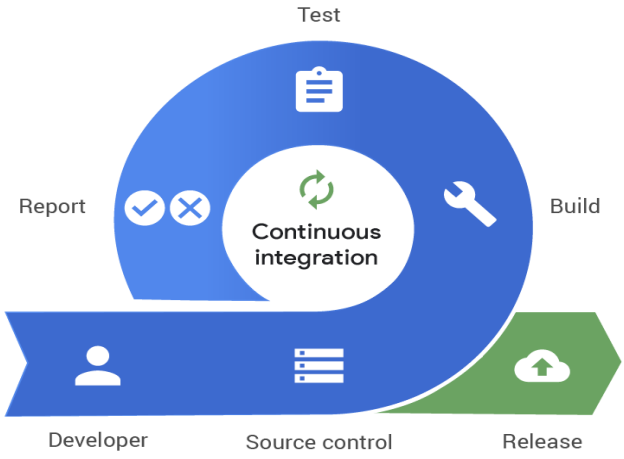
- *The basics of CodePipeline*
- *Then integration of it with CodeCommit*
- *And with CodeBuild*

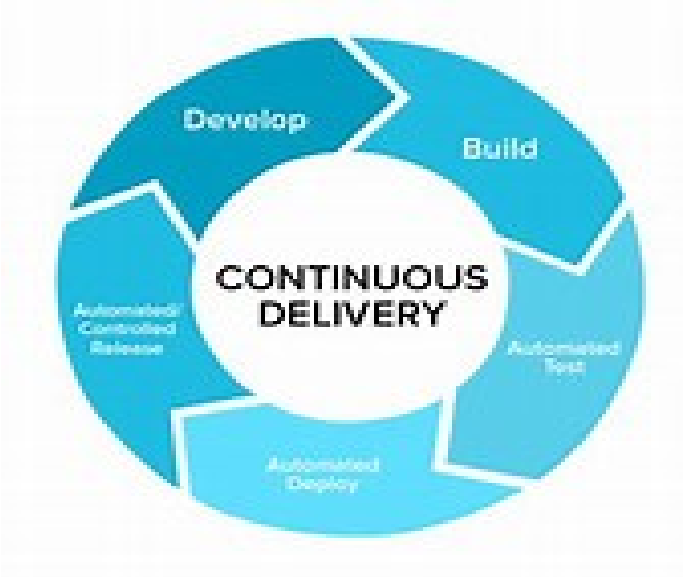
HERE WE GO

AWS-CodePipeline

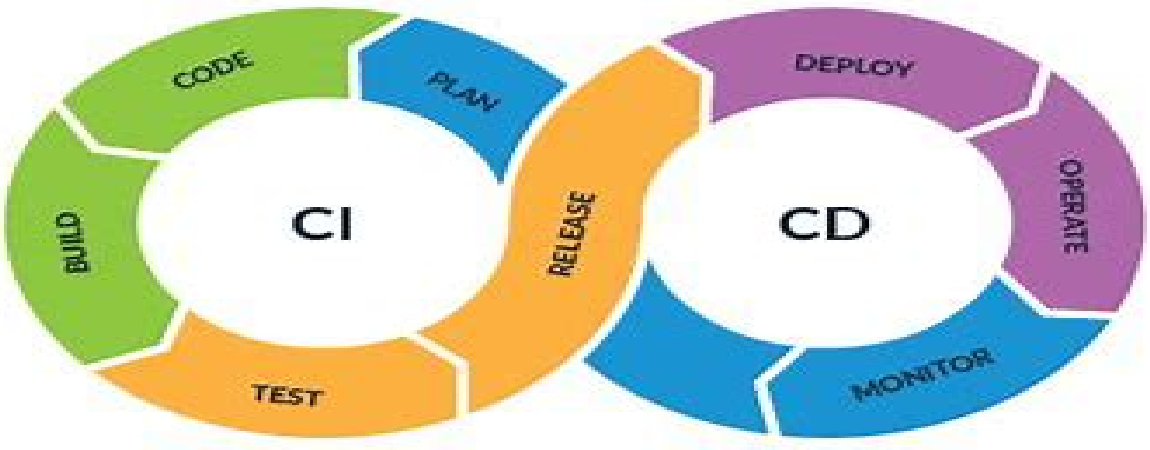
In order to understand the term CodePipeline. You should know about what is CICD

CI - CI (Continuous Integration) is an automated software development practice, frequently integrating code changes and automating testing to ensure software reliability.
Here, developers commit code changes frequently, several times a day, to the code repository.
In every code submission, automated builds and unit tests are run.





CD- CD (Continuous Delivery) is an automated software development practice for regularly delivering code changes to production, enhancing development efficiency and user satisfaction.
It actually has two means
Continuous delivery- where manual approval is required to deploy
Continuous deployment- where no approval is required to deploy



Now what is CodePipeline? 🤔

- It is a AWS service that provides visualizing, automating, and modeling software release processes, promoting continuous delivery.
- You can model continuous deployment or continuous delivery to build ,test and deploy to a production environment.
- You can integrate third party tools like jenkins, blazemaster or add custom action using AWS lambda.
- AWS CodePipeline is the orchestrator of CICD flow.

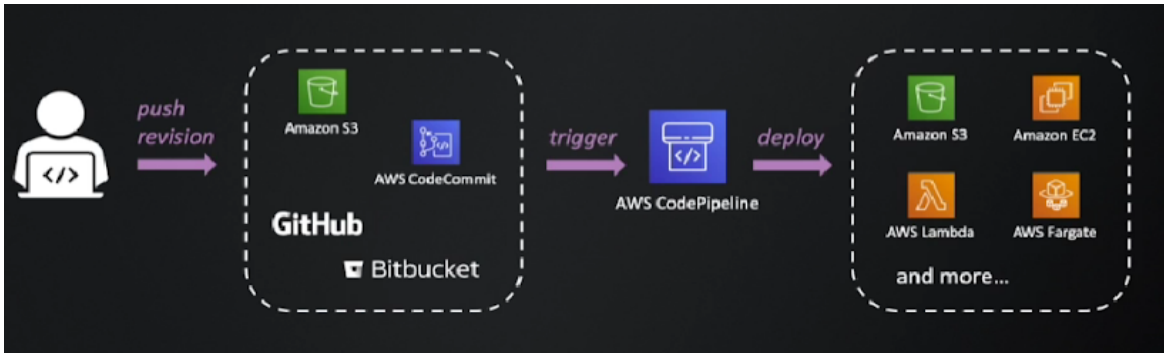
Some concepts in it are:

- Pipeline → describing how software changes through a release process
- Source Revision → Triggering source version in pipeline.
- Action → Task on source revision, sequential or parallel.
- Stage → Group of pipeline actions.
- Transition → Link between stages.
- Artifact → Action output, usable input.

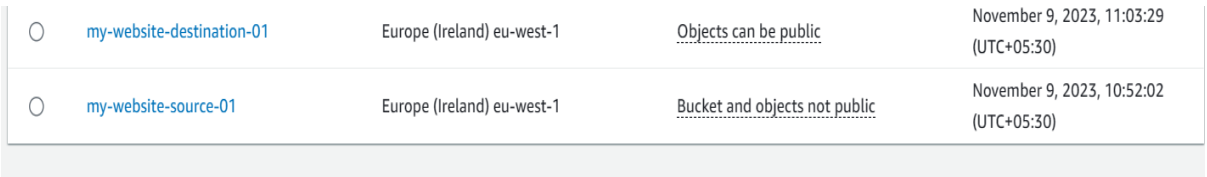
How does it work?

Submit your code changes to the pipeline's source location, generating a source revision.
The pipeline activates, and the source revision progresses through various stages.
The source revision is then deployed to the production environment.

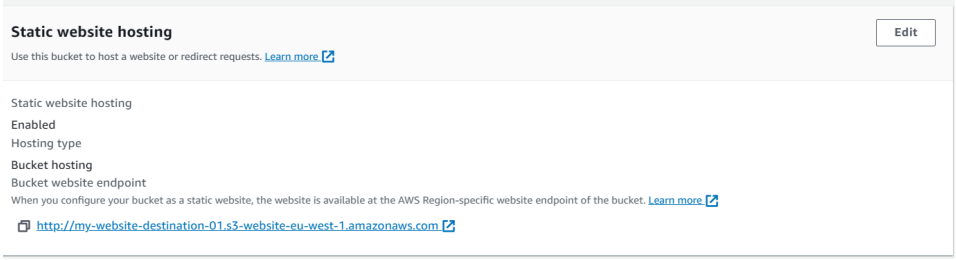
In this example, we are going to use amazon s3 as source as well as destination.



Let us try to understand what this is with the help of an example where we will make a pipeline from bucket to bucket where 1st bucket has source code and 2nd bucket is production ready bucket.

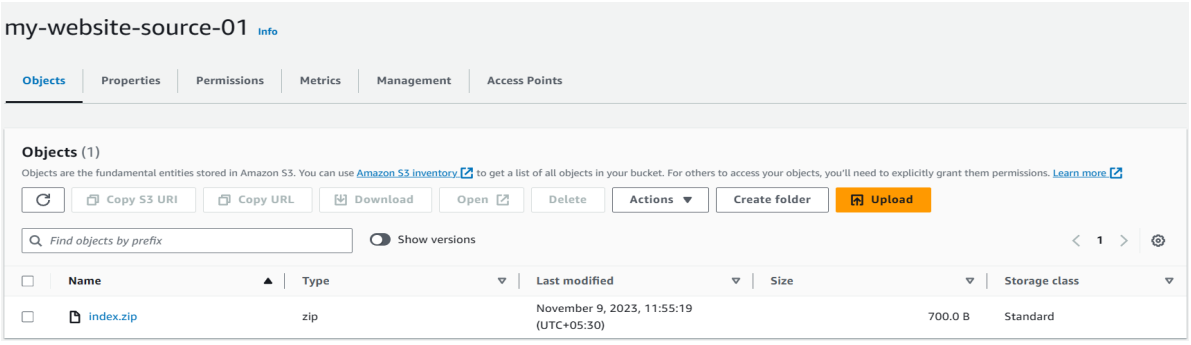


Now go inside destination or prod bucket and in the properties section make the static website enable and provide the index.html as page.You can find sample index.html here.



Now if you try to open up the link now it will show “403 forbidden” because we haven’t provided what to display.

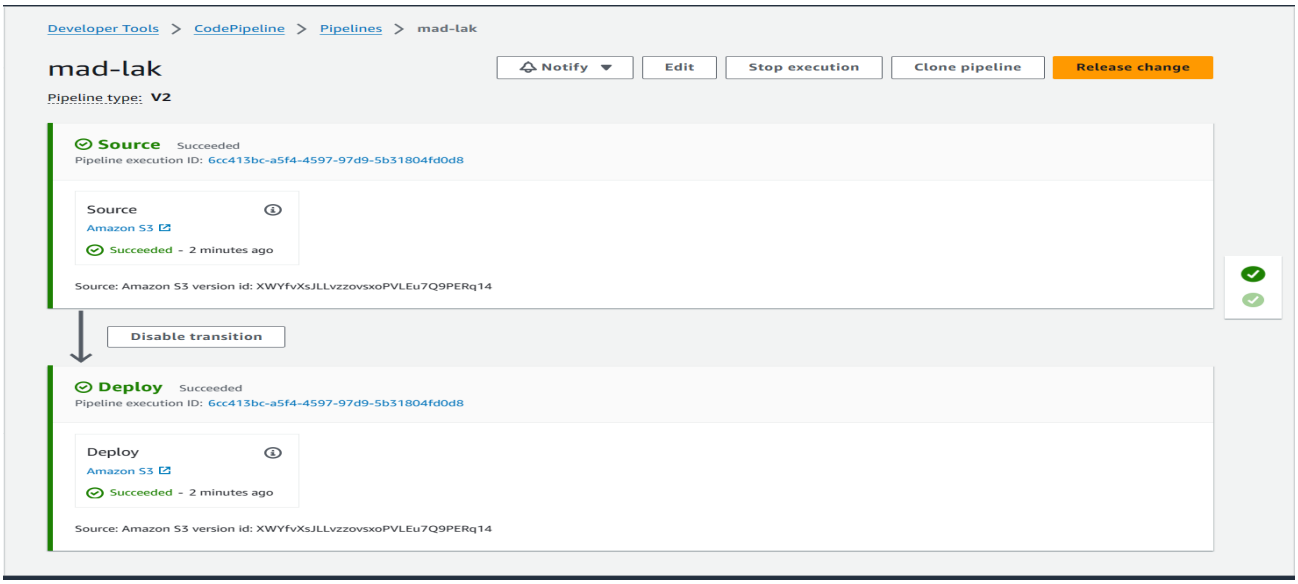
Now upload the index.html to source bucket but remember that whenever you are using S3 bucket as source then the file that you are using should be **“a single file in the form of zip package”**.



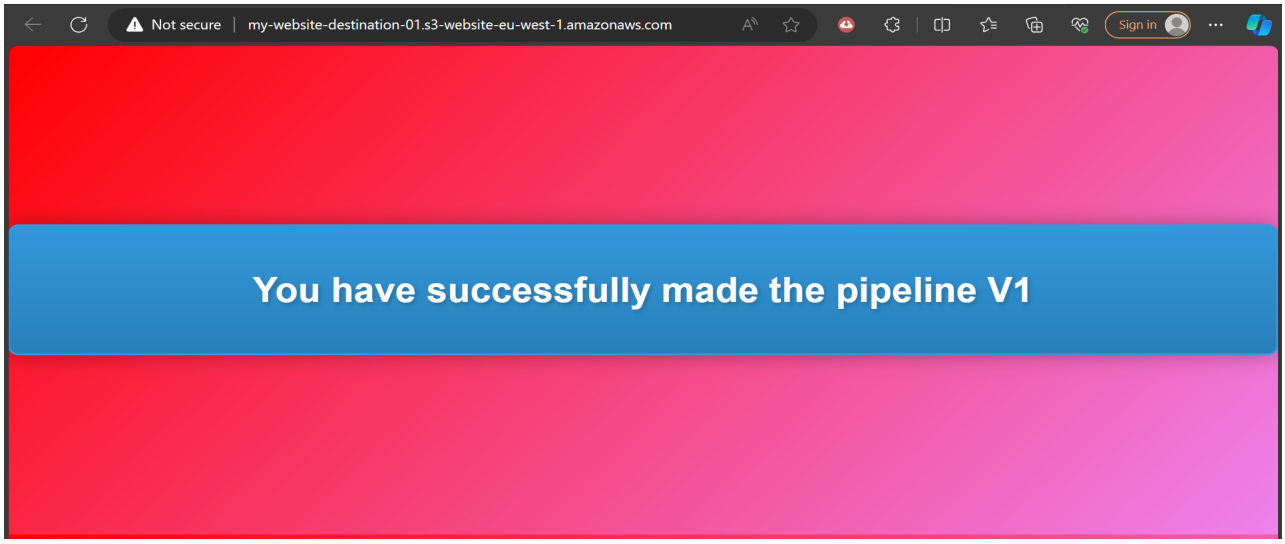
Now go to Codepipeline page and follow the steps-

- Create pipeline
- provide it a name and keep all default
- on source page, select S3 & provide object key as “index.zip”, detection mode as cloudwatch
- Skip build stage for now, then use deploy provider as S3 bucket “my-website-destination-01” and check the box “extract file before display”, in addition, make canned ACL as public-read.
- now review the pipeline and create a pipeline.

You will see a page like this in which green color confirms that the pipeline and its components are working fine.



When you visit to production or destination bucket's static website hosting section you will find a link to showcase the result when you click on it you will find an output of your index.html file.



DO IT BY YOURSELF

Wonder how you can trigger your pipeline ?

Hint just go & make some changes in the source file or replace it. Pipeline will trigger automatically. Disabling the transition would be required first.*

AWS-CodeCommit

Let me give a short introduction about git first :--

Git

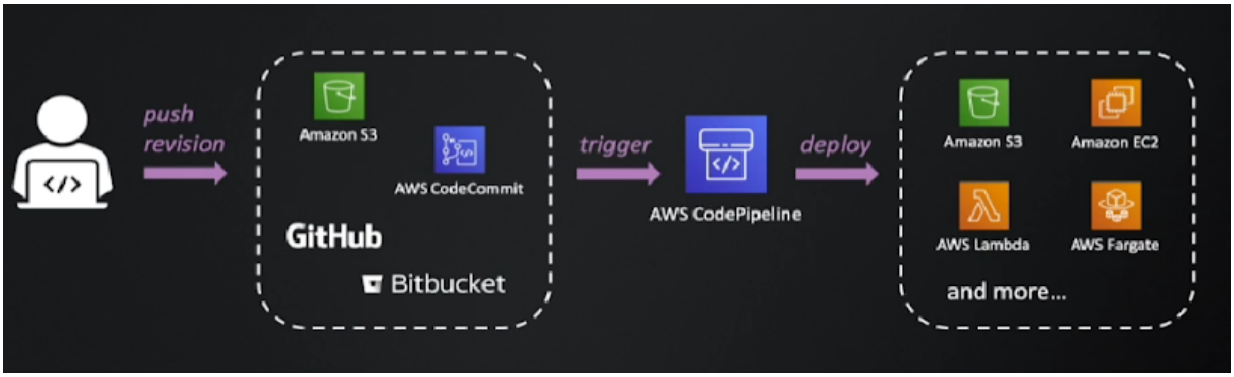
Concepts in git :

- Repository : the collection of files and folder
- Commit : A snapshot of git repository. Git records the entire content of each file in every commit.
- Local repository : the version of a project on your computer which you locally work on your own.
- Remote repository: the version of a project on the internet or network.

Now what is CodeCommit? 🤔

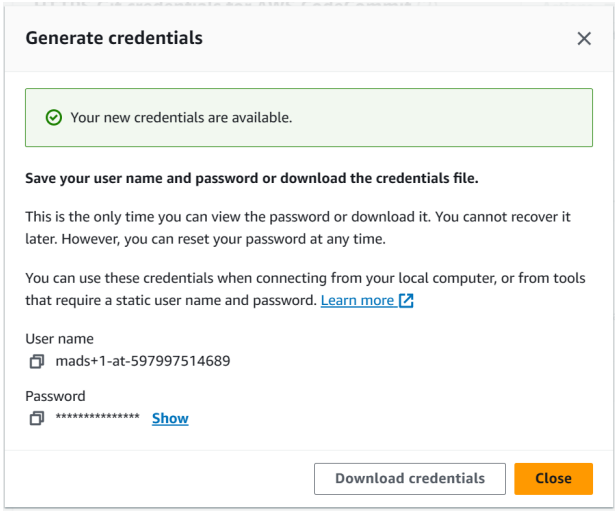
- It is an AWS service that offers a serverless, fully managed service for hosting private git repositories.
- Enjoy high durability, availability, and scalability with no constraints on file time or size, although S3 is recommended for files exceeding 5MB.
- This service ensures optimal performance for secure and efficient version control of your codebase.
- You can use standard git commands and tools to operate or utilize the repositories.
- It is a part of AWS developer tools and can be integrated with various AWS services like cloudformation, EBS, lambda, cloudwatch etc.

In our example , we are going to use AWS CodeCommit as source & S3 as destination.



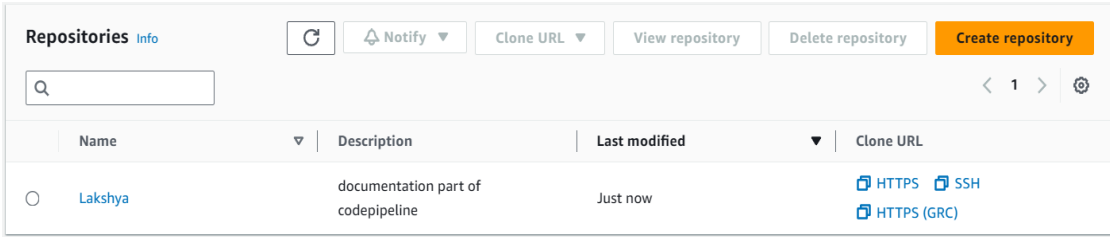
We are going to use push things from local repository through git to codecommit's repository (that we are gonna to make) and for that we require the following

Before starting just quickly visit the to user section in IAM role
In security credential section visit to *HTTPS git credential for AWS Codecommit* and then generate credentials
don't forget to download or save it as you can't see the screen again

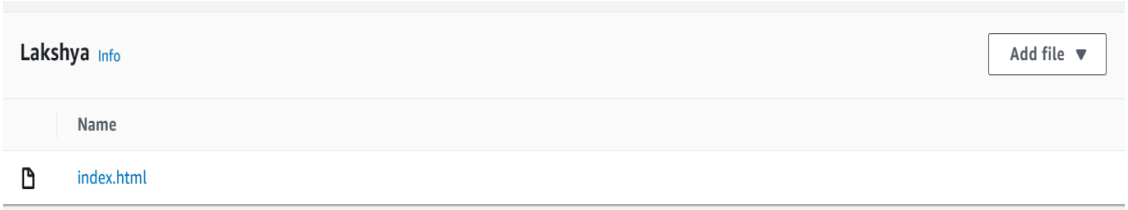


Let's start now ,

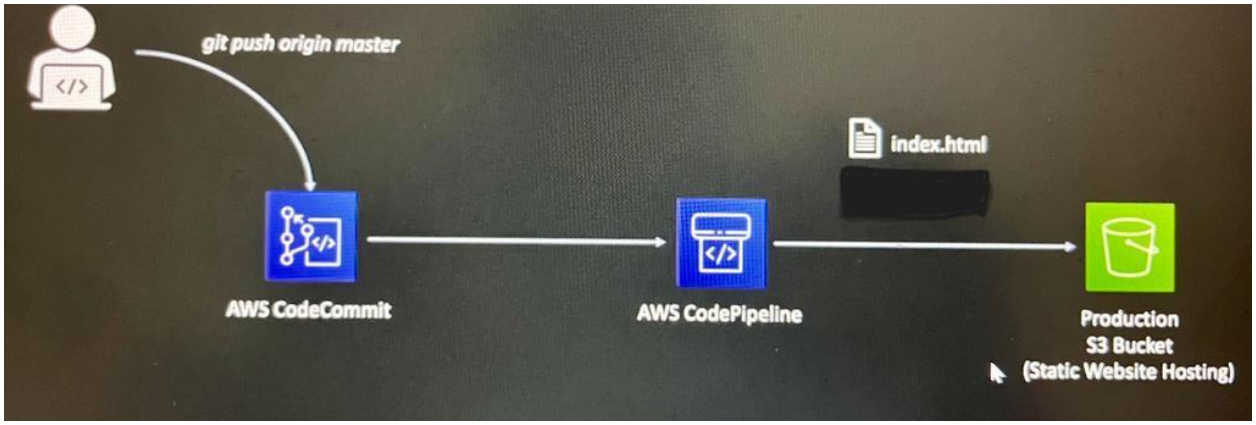
- Navigate to CodeCommit page, in repository section and click create repository
- Fill up the name of repository



- Now go to local repository which contains the file that we want as a source file (in our case index.html)
- Make the initial steps of git i.e, “git init” followed by “git add .” and “git commit”
- Now to add a remote repository of codecommit , write “ git remote add origin (clone https of repository) “ .
- A screen will be prompted which will ask username and password , write the credentials that we have downloaded from IAM.
- Then write “git push origin master” to push.
- After that you can look into the repo, it will no longer be empty .

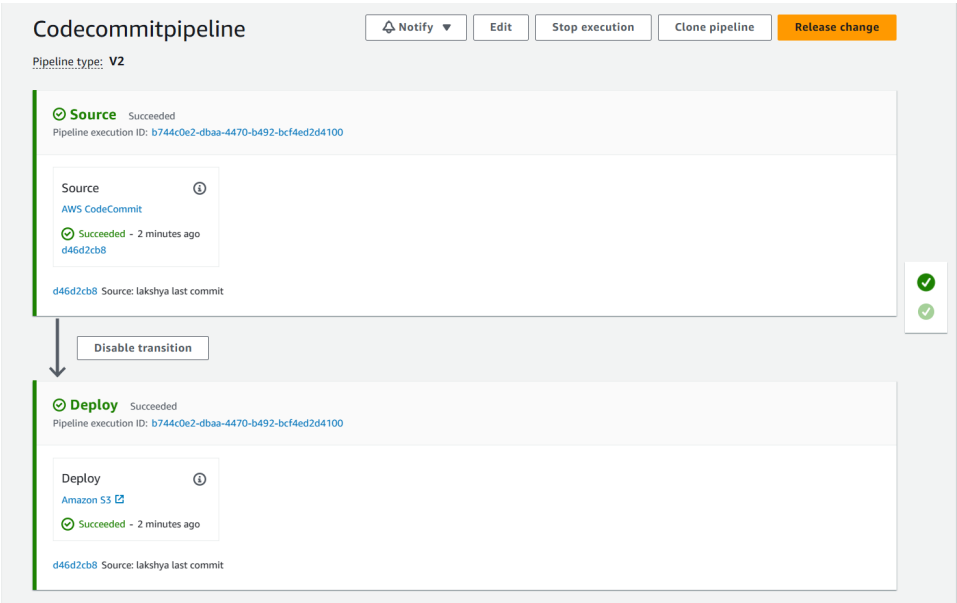


We are done with codeCommit and S3 Bucket here now we remain only with Pipeline



Now the steps to create the pipeline :-

- Navigate to the pipeline page
- Write the name of pipeline
- Use the new service role that will be created by aws itself and click next.
- In source provider use aws codecommit, write the repository name and branch name in which you have pushed from local repository then for detection use cloudwatch and click next
- Skip the build for now
- In the deploy section write the same as we did in the previous task.



Yes we had successfully used the codepipeline with codecommit

DO IT YOURSELF :-

To test automatic triggering of pipeline

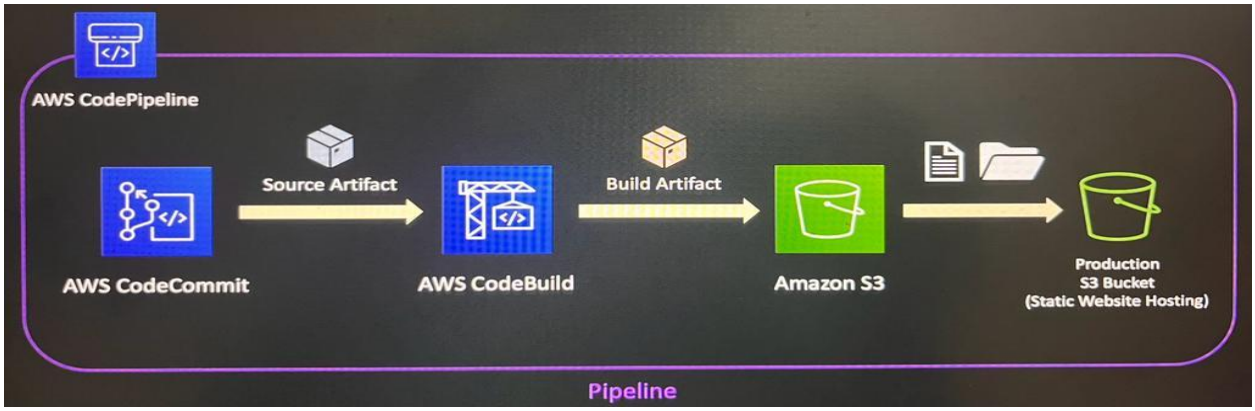
Hint: just visit the local repo and make changes in it after that, make use of the git command to push again to the remote repo on codecommit or you can directly edit the files in the repository.

AWS-CodeBuild

- AWS CodeBuild is a cloud-based build service and a part of AWS Developer Tools.
- It operates in a serverless, fully managed fashion, scaling automatically based on demand.
- It seamlessly integrates into your pipeline as a command line.A scalable solution offering flexibility and efficiency in the build process.
- Cost-effectiveness is ensured as you pay only for the minutes your build project consumes.
- Utilizes Docker containers for building environments. Supports Docker images from CodeDeploy, DockerHub, or Amazon ECR.

In addition to all it, one should also know about Buildspec file

- A YAML file, your buildspec, houses commands and runtime settings.
- Embed it with your source code or define it on AWS CodeBuild console during project creation.
- Organize commands under build phases like install, pre_build, build, post_build for execution.
- Output artifacts are specified in the buildspec, determining included files and their base directory.
- This file allows precise configuration, streamlining the build process in AWS CodeBuild.



So what we are building now, we have added code build in which we will write the dependencies to run our angular code

First push angular code to codecommit that we are going to use in these example
This is the link to the code **CODE**
If you don't know angular , no worries , our main focus is on understanding CodeBuild.
We have also included a Buildspec file in the code , let us understand what is it

```
version: 0.2
phases:
  install:
    runtime-versions:
      nodejs: 12
    commands:
      - npm install -g @angular/cli@9.0.6
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - ng build --prod
artifacts:
  base-directory: dist/my-angular-project
  files:
    - '**/*'
```

Install Phase:

- This phase sets the Node.js runtime version to 12.
- Installs the Angular CLI globally with a specific version (9.0.6).

Pre-Build Phase:

- Installs project dependencies using npm.

Build Phase:

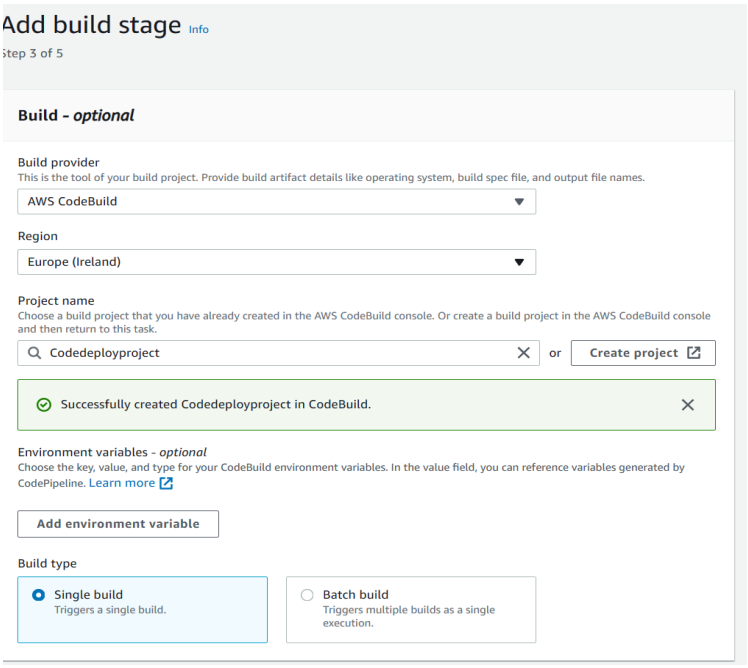
- Builds the Angular project in production mode using the Angular CLI (ng build --prod).

Artifacts:

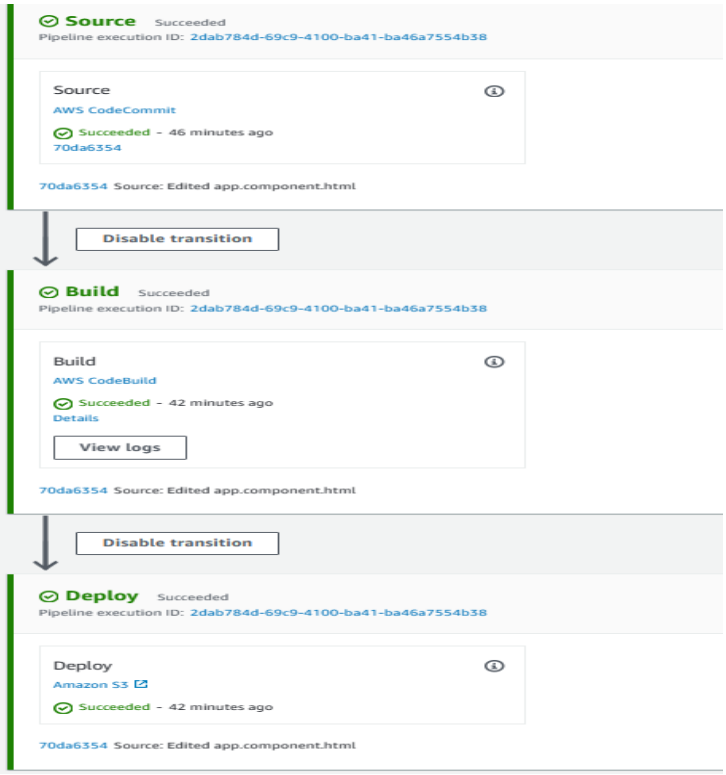
- Specifies the base directory for artifacts as dist/my-angular-project.
- Includes all files and directories ('**/*') from the specified base directory in the artifact.

Let's start creating the pipeline:

- Navigate to the pipeline page .Write the name of pipeline
- Use the new service role that will be created by aws itself and click next.
- In source provider use aws codecommit, write the repository name and branch name in which you have pushed from local repository then for detection use cloudwatch and click next
- In the build stage, Select AWS CodeBuild and then select region
- Then select creating project
  -
    - Give the name to project.Write a description (can also add the tags)
    - Select service role or create new one
    - In the buildspec section make a use of buildspec.yaml file that is actually the file that we had created earlier.
    - Then select image and environment type.
    - And then create project
    - After that it shows a screen of succession like this→

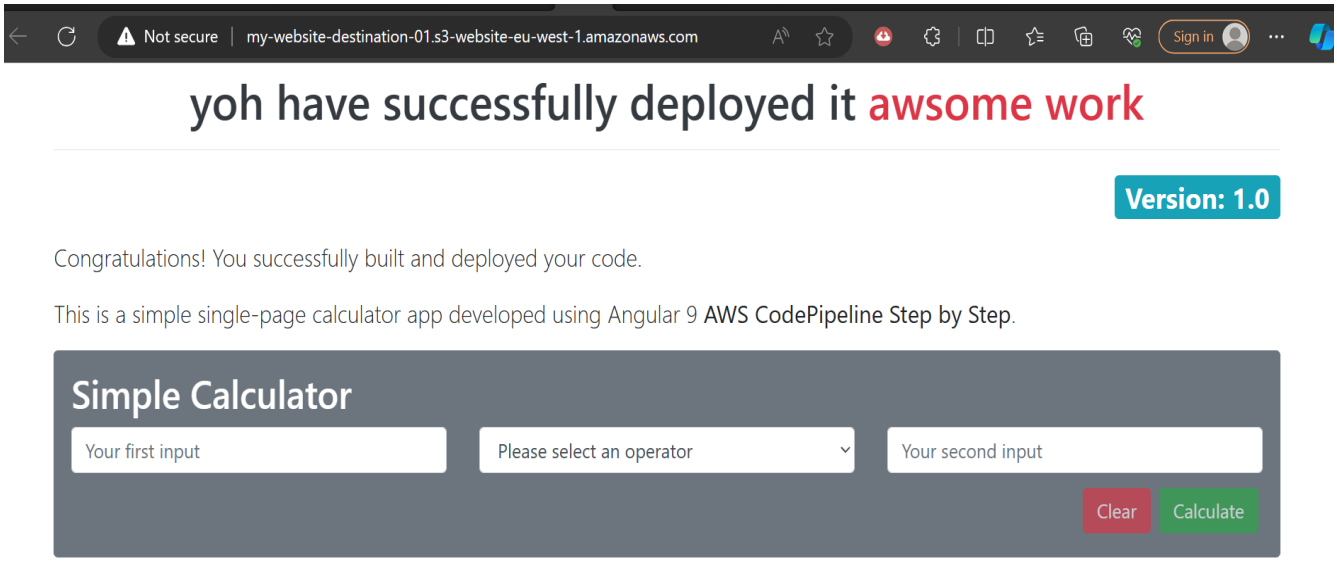


- In the deploy section, select the S3 as deploy provider
- check the box “extract file before display”, in addition, make canned ACLs public-read .
- After successfully doing the whole steps, you will be prompted to a screen for review.
- Then you will be taken to pipeline structure as follows



The message of succession i.e, all deployments and builds are successful.

And the static website on S3 will look like as :-



YOU HAVE DONE IT

BEST OF LUCK