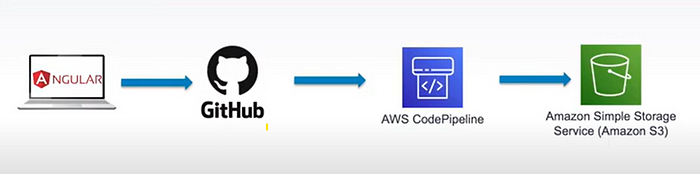
Continuous Integration/Continuous Deployment (CI/CD) is crucial in modern software development as it streamlines the process of delivering high-quality software at a rapid pace. CI/CD automates and integrates various stages of software development, from code integration and testing to deployment and monitoring.

In this article we will go through the steps of creating a CI/CD pipeline for your angular application using AWS code pipeline.

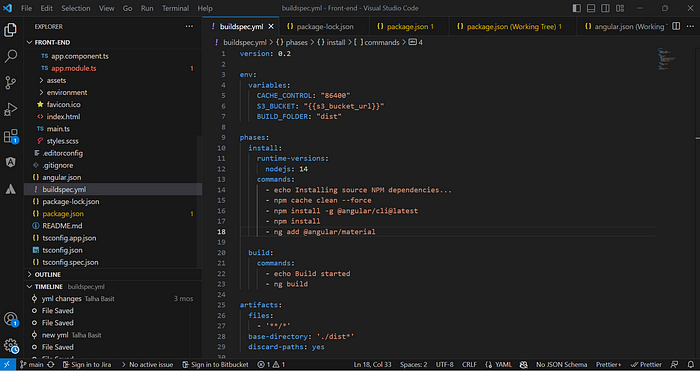
The Deployment architecture that we will be using looks as follows.



We will be using an angular application located in a github repository. Then we will configure the AWS code pipeline so that when ever we commit any changes, AWS code pipeline will detect those changes, and will start the deployment process to our s3 bucket.

Lets get started.

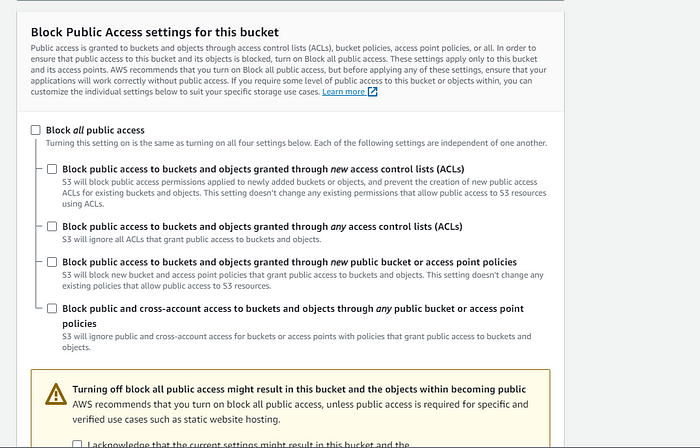
**Step 1**: In the root directory of your angular project, create a yaml file and the name of this file must be **build.spec.yml**.



This file will contain all the instructions for the build stage of our code Pipeline.

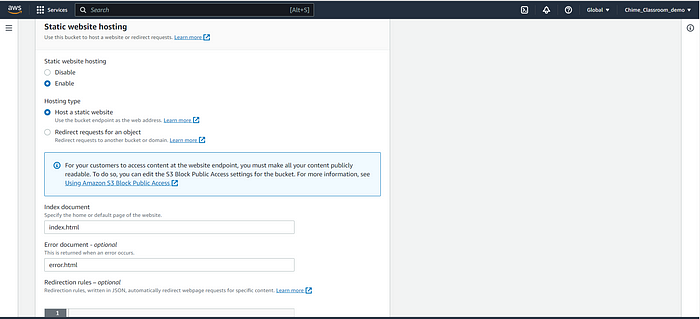
Make sure to commit the changes of this file and push it to the remote repository.

**Step 2**: Log in to your AWS account and create a new s3 bucket.



Make sure to uncheck the **Block all public access**checkbox so that we can access this bucket from anywhere.

After the bucket has been created, open that bucket in you AWS console and click on the properties tab. Now scroll down to the bottom and you’ll see **Static website hosting.**Click on edit.

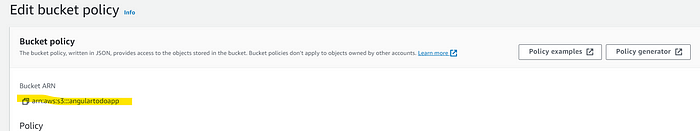


Click on edit to Enable the static website hosting option and also fill the Index document and the Error document with index.html and error.html respectively.

Now click on the permissions tab for this bucket and scroll to **Bucket policy.**Edit the bucket policy and paste the following code in the bucket policy code.

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

Do not forget to replace the Resource value with your aws bucket arn.

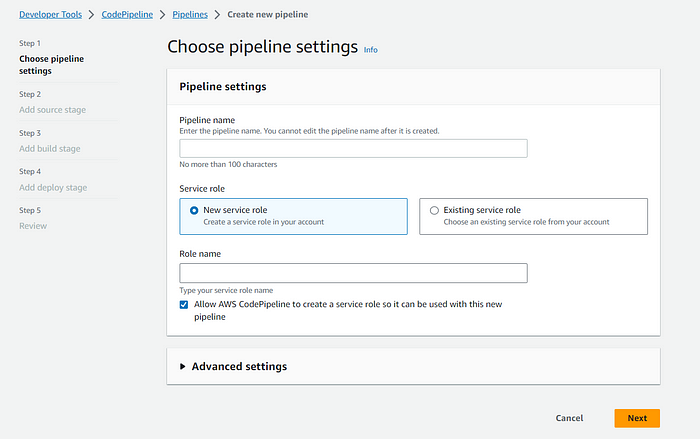


Now your aws bucket has a public access. You can verify this by going back to the properties tab, scroll down to the static website hosting section and click on the bucket link. You will be routed to a new webpage which will show a 404 error which means that the bucket is now public and will open the resources once we upload anything on to the bucket.

**Step 3**: Create a new AWS codePipeline.

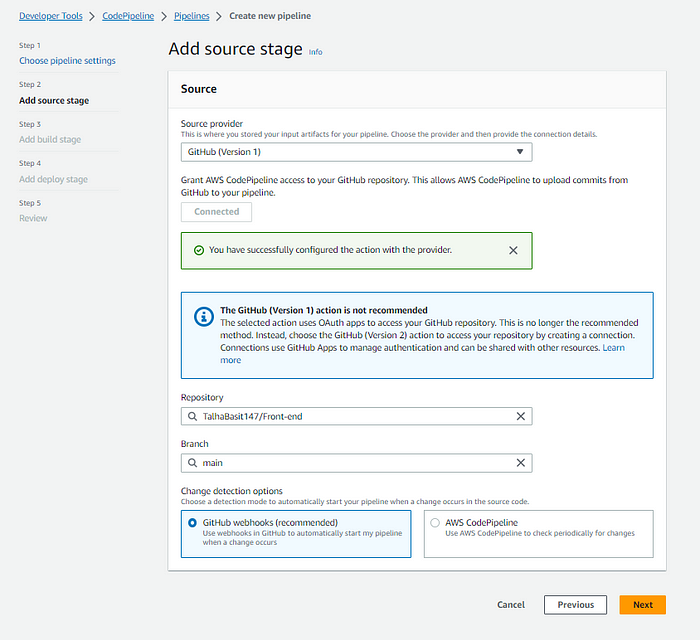
Search for the CodePipeline service in your aws management console.

Click on create pipeline which will take you to a 5 step wizard.



Give your pipeline an appropriate name and click on next to move to the source stage.

In the source stage, you need to configure the source where your code will be pulled from. In our case we will configure it with the github account where our angular code repository resides.

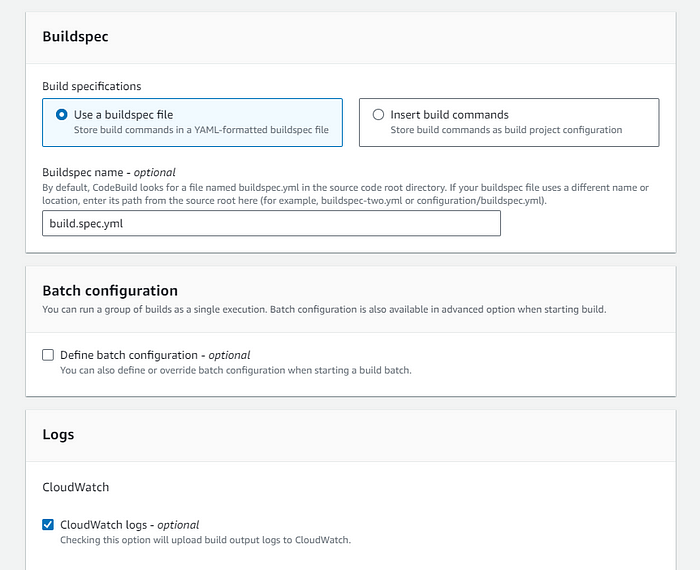


Select the appropriate repository and branch which you want to configure and click next.

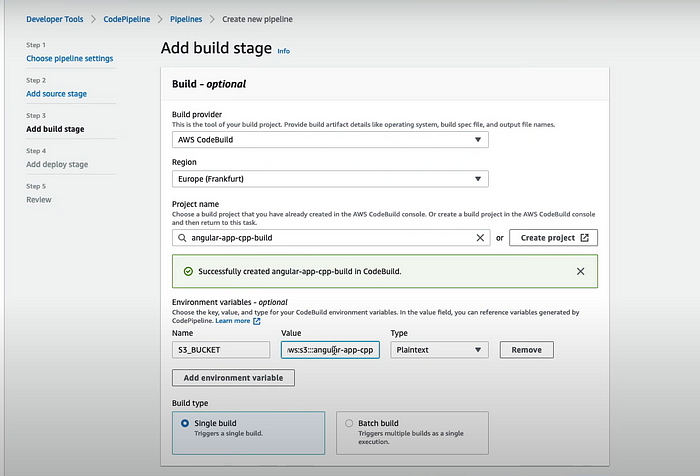
Next we move to the build stage.

In the build provider select AWS Codebuild.

Then click on create project to configure the machine specifications for your project. Select Ubuntu standard image and in the build specifications, select the option to use a build spec file.

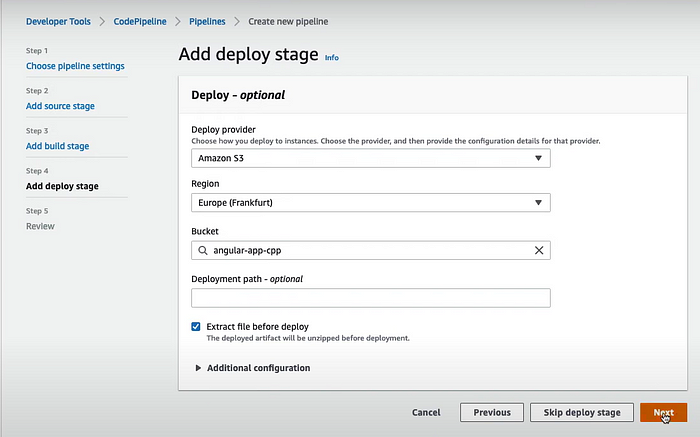


Now back to the build stage, click on add environment variables.



Now the last stage of the code pipleine is the deploy stage.

Select Amazon s3 as the deploy provider, Select the bucket name where you want to deploy the build files and do not forget to check the **Extract file before deploy**checkbox.



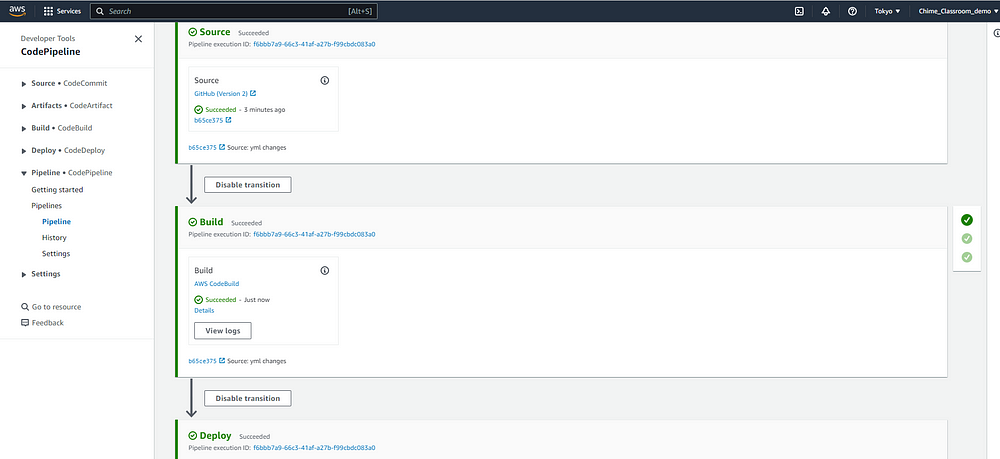
Click next and review your pipeline before you click the create pipeline button.

**Step 4**: Test your AWS CodePipleine.

Now that we have created our lets test it. Open your code pipleine, and click on the release changes button.

AWS codepipeline will run three steps.

First it will get the latest code from the configured github branch. The in the secode stage it will build the code based on the build commands provided in our **build.spec.yml** file. After the build is successfull, the files will be uploaded to your S3 bucket.



Once all the above stages are successfull, You can verify this by going to your S3 bucket, click on properties tab, scroll down to the static website hosting section and click on the bucket link.

In today’s fast-paced software development landscape, agility and reliability are paramount. We’ve explored the power of Continuous Integration and Continuous Deployment (CI/CD) using AWS CodePipeline as a crucial tool in achieving these goals. Through this journey, we’ve seen how CI/CD can streamline your development process, leading to faster releases, higher software quality, and reduced risk.

As we conclude our exploration of CI/CD and AWS CodePipeline, remember that this is just the beginning of your journey towards a more efficient and effective development workflow. The world of cloud computing and DevOps is ever-evolving, and embracing continuous improvement is key.

I encourage you to dive deeper into the AWS ecosystem, exploring other services that complement CodePipeline, and to keep an eye on emerging best practices in CI/CD. Embrace the culture of continuous learning, for it is through learning and adapting that we stay ahead in the tech industry.

Thank you for joining us on this journey. I look forward to hearing about your experiences and successes with CI/CD in the comments section below. As a community of developers, we can learn from each other and drive innovation together. Happy coding!