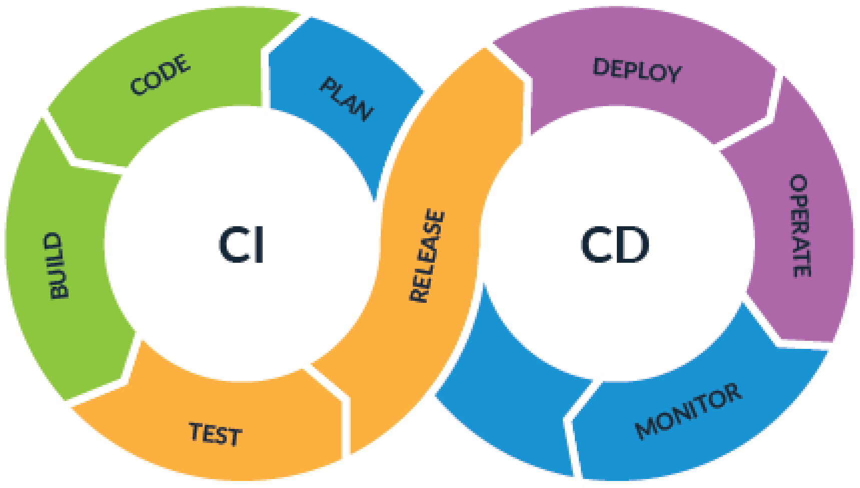
AWS CI/CD to deploy ReactJs



CI/CD is perhaps one of the best tools that facilitates the deployment of software deliverables as they are being developed. It provides a way to deploy our latest changes when push our code to our remote repository.

This project uses CI/CD pipeline provided by AWS to deploy the latest changes on the React application as I often make small adjustments to the view and may include more content to the details.

Here is what my development workflow would look like if I hadn't set up a CI/CD pipeline.

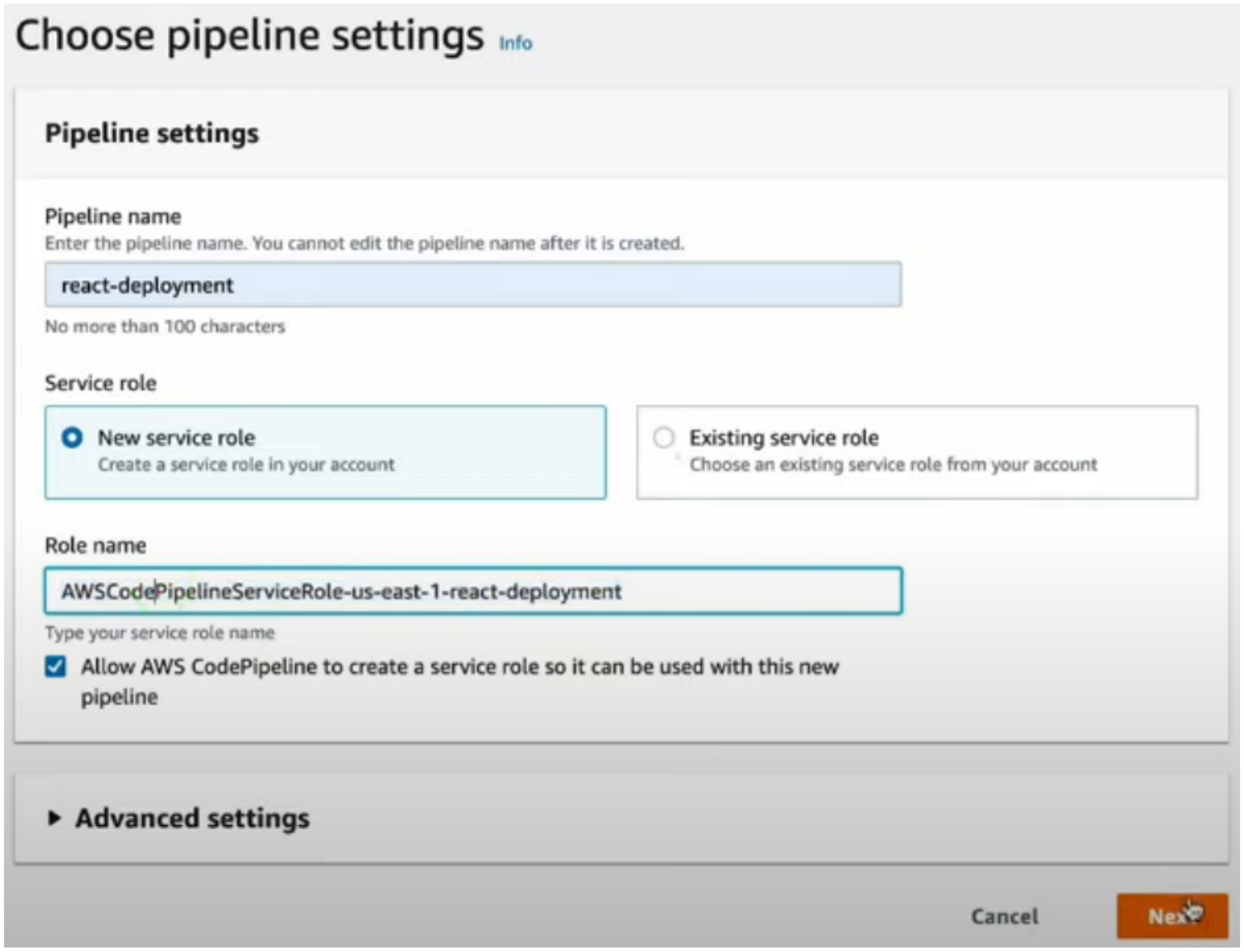
- I would make changes and commit them to a remote repository in order to keep track of my version.

- I would run a build command *npm run build* locally to get built files for production.
- Then I have to upload those files to S3 bucket where my static website is currently being deployed.
- I would need to invalidate my ***CloudFront*** distributions so that any cached version of my previous website would be replaced by my new one.

All these would have to be done manually which is obviously tiresome. AWS provides services that will handle these steps behind the scenes, so that the only thing required from me is to push my code, wait for awhile until the process finishes [*get some coffee perhaps* ☕☕☕ 😄😄] and check my hosted site.

Here are the steps I took to set up a CI/CD pipeline for my workflow.

- Search an AWS service called ***CodePipeline*** and click on Create pipeline
- Give your pipeline a name and attach a role to it. For this demo, use the *Create new role* option**.**


Choose pipeline settings Info

Pipeline settings

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

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
AWSCodePipelineServiceRole-us-east-1-react-deployment
Type your service role name
☑ Allow AWS CodePipeline to create a service role so it can be used with this new pipeline

▸ Advanced settings

- Choose your source stage. This basically means where your remote repository resides. For this project lives in **Github,** so I will choose Github-V2. You will be redirected to the Github login window. You will need to create a ***Connection*** to Github and make sure to install the app required to the connection between AWS and Github.
- Choose the project repository and the deployment branch.
- Next, we will add a **Build Stage.** So, choose ***CodeBuild*** which is another AWS service.

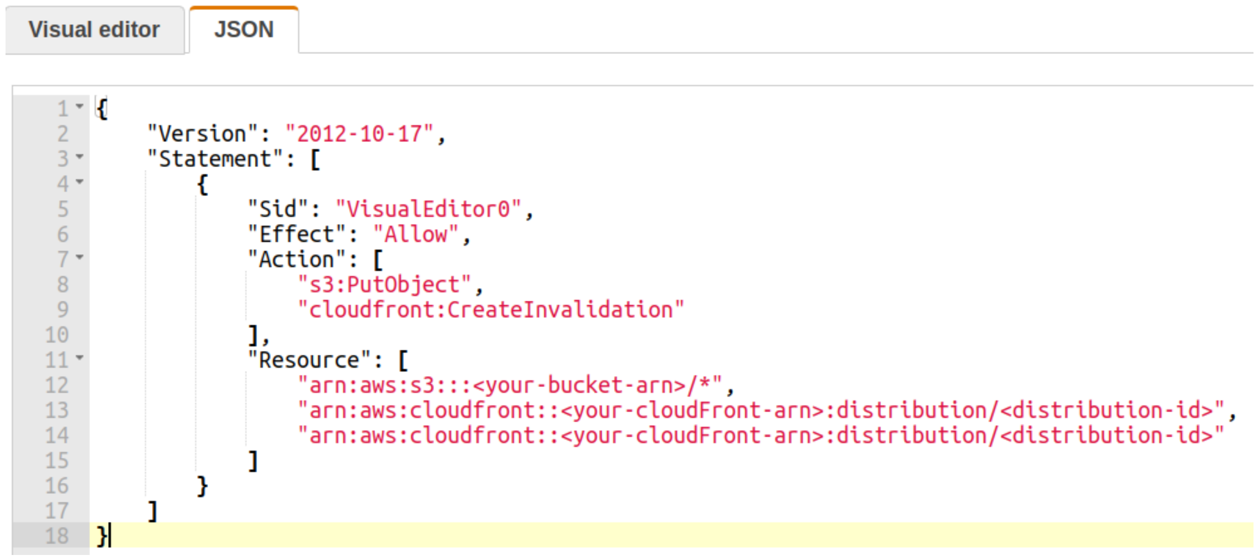
- Provide a name and choose the region and click on **Create Project.** This is the virtual environment where your code will be built and run deployment commands.
- For **Operating System**, choose Amazon Linux 2. And **Runtime,** choose *Standard.* For Image version choose the latest one.
- Choose the **New Service Role,** to make AWS create a new role on our behalf and will **update** that role **later**. Make sure to remember the name of the role.
- Then click **Continue to CodePipeline.**
- Then on the CodePipeline creation page, click **Next.** And then Skip on the Deploy Stage page.
- Review your settings and click on the **Create pipeline** button.

At this point, the pipeline will be created and it will run for the first time. But note that it will fail. This is because we have not yet added a configuration file for the **CodeBuild** project. This file is called ***buildspec.yml***

Before we add the buildspec.yml file in our React project, we need to modify our Service Role that was created for us when we set up the ***CodeBuild.*** This is because, for our case, CodeBuild needs to *PutItem* in ***S3*** and *Create-Invalidation* in ***CloudFront.***

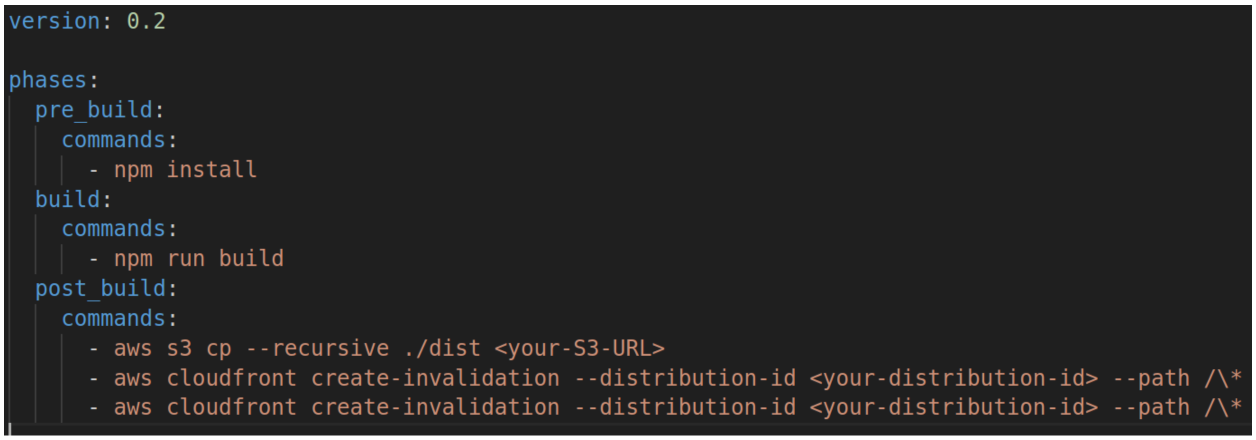
- Go to ***IAM roles*** and find the role that was created by our new ***CodeBuild*** project.

- Click on **Add Permission** and **Create Inline Policy.** Below is a sample policy. Update the resource ARN and as many ***CloudFront*** distributions you have. This sample contains two.



```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "VisualEditor0",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "cloudfront:CreateInvalidation"
            ],
            "Resource": [
                "arn:aws:s3:::<your-bucket-arn>/*",
                "arn:aws:cloudfront::<your-cloudFront-arn>:distribution/<distribution-id>",
                "arn:aws:cloudfront::<your-cloudFront-arn>:distribution/<distribution-id>"
            ]
        }
    ]
}
```

Now, let's go back to our React code. Create a file *buildspec.yml* in the root directory and add this sample YAML code.



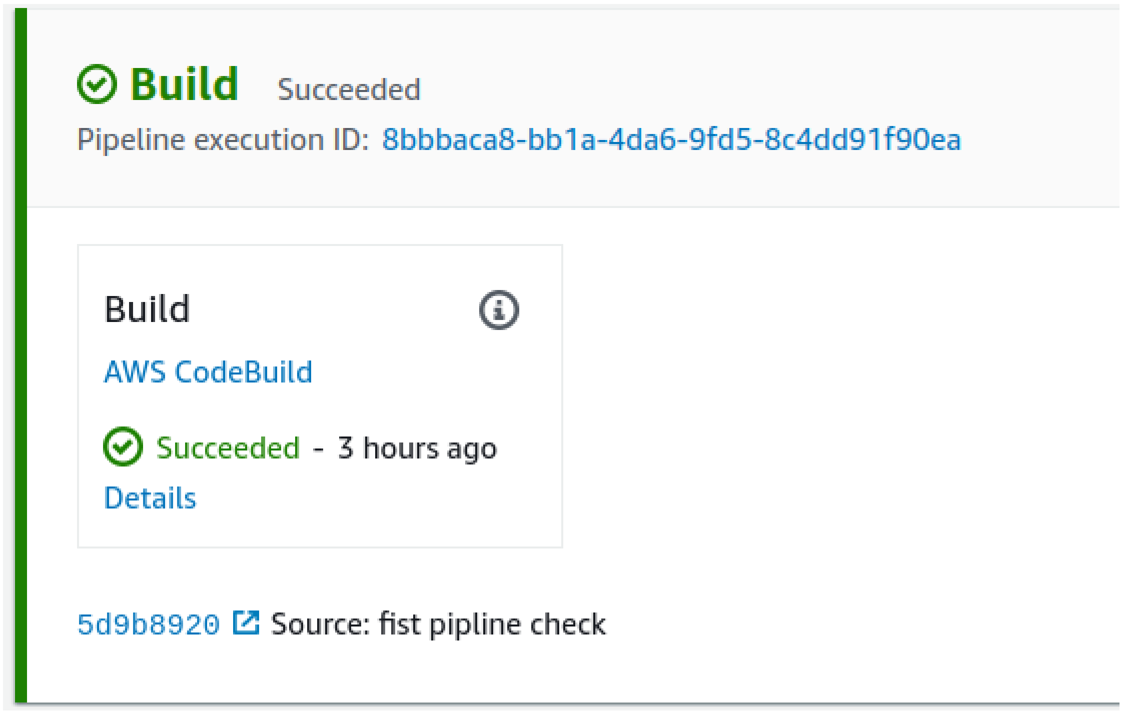
```
version: 0.2

phases:
  pre_build:
    commands:
      - npm install
  build:
    commands:
      - npm run build
  post_build:
    commands:
      - aws s3 cp --recursive ./dist <your-S3-URL>
      - aws cloudfront create-invalidation --distribution-id <your-distribution-id> --path /\*
      - aws cloudfront create-invalidation --distribution-id <your-distribution-id> --path /\*
```

This configuration contains three sections. The first installs the npm dependencies before building. The second runs the build command *npm run build* and the third section:

- Copies the output of the build command to your S3 bucket. Here, make sure to change the directory of your build output based on your Javascript bundler. It may be ./build
- Then, it invalidates the ***CloudFront*** distributions based on their id.

That's it. Congratulations. You can make some noticeable changes to your React application and push it. This will trigger the CI/CD and, in a few minutes, deploys the new version to ***CloudFront.***



Build Succeeded
Pipeline execution ID: 8bbbaca8-bb1a-4da6-9fd5-8c4dd91f90ea

Build
AWS CodeBuild
Succeeded - 3 hours ago
Details

5d9b8920 Source: fist pipline check