Member-only story

Sync your AWS CodeCommit repository to your GitHub repository for public code sharing or private repository backup

Paris Nakita Kejser · Follow

9 min read · Aug 27, 2022

After starting using Amazon Web Service (AWS) for real use, I'm falling in love with the CI/CD pipeline on AWS, I feel its easier to integrate with services when you use CodeCommit for Git, CodeBuild for the building process, and CodeDeploy when you deploy your software out but you can use different services out there, but in my case, I fell good to keep all my code on AWS.

The problem can be when you will do more *"open source"* projects or just share your code with the public, I have a different repository on GitHub from legacy time and now

I want to change it from GitHub to AWS CodeCommit.

Then I need in some cases to sync my Git repository from AWS CodeCommit to a GitHub repository, and it can be many cases why I want to do this, so I should find an easy way to do that.

Manually sync repository from between two GIT remote repositories

When you are starting you can do it free by running the commands manually every time you need it, its costs your time but if you see your time as free it's an easy way to start and you need to understand how it works before you are doing it automatically anyway.

First, we need to clone our primary repository

```
1  git clone --mirror https://primary_repo_url/primary_repo.git
```

git-clone-repo-as-mirror.sh hosted with ❤ by **GitHub** **view raw**

then you need to go into your repo folder and add the new remote repository.

```
1  cd primary_repo.git
2  git remote add --mirror=fetch secondary https://secondary_repo_url/secondary_repo.git
```

git-add-remote-repo.sh hosted with ❤ by **GitHub** **view raw**

Now you are ready to fetch out on your origin (primary) remote repository, and after it pushes everything to your secondary remote repository.

```
1  git fetch origin
2  git push secondary --all
```

git-fetch-push-all.sh hosted with ❤ by **GitHub** **view raw**

If you do changes in your primary (origin) repository you can just re-run these 2 lines in your terminal, then it will fetch your data from the remote origin and push it to the second remote repository.

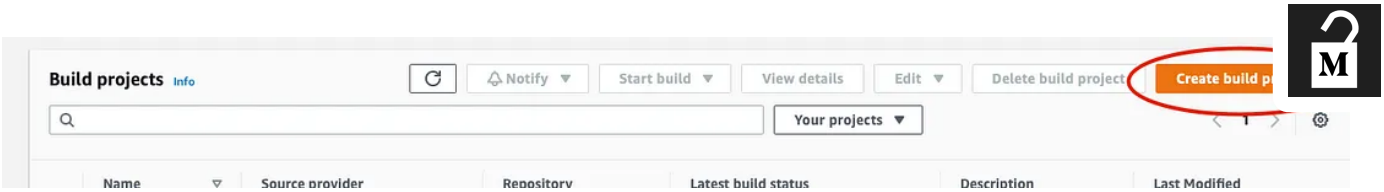
Use CodeBuild to sync the CodeCommit repository to the GitHub repository with Event Bridge and Lambda function

As said before if your time does not cost you money, then it's okay to do it manually, but automatic is much better so you don't need to remember the boring stuff over and over again.

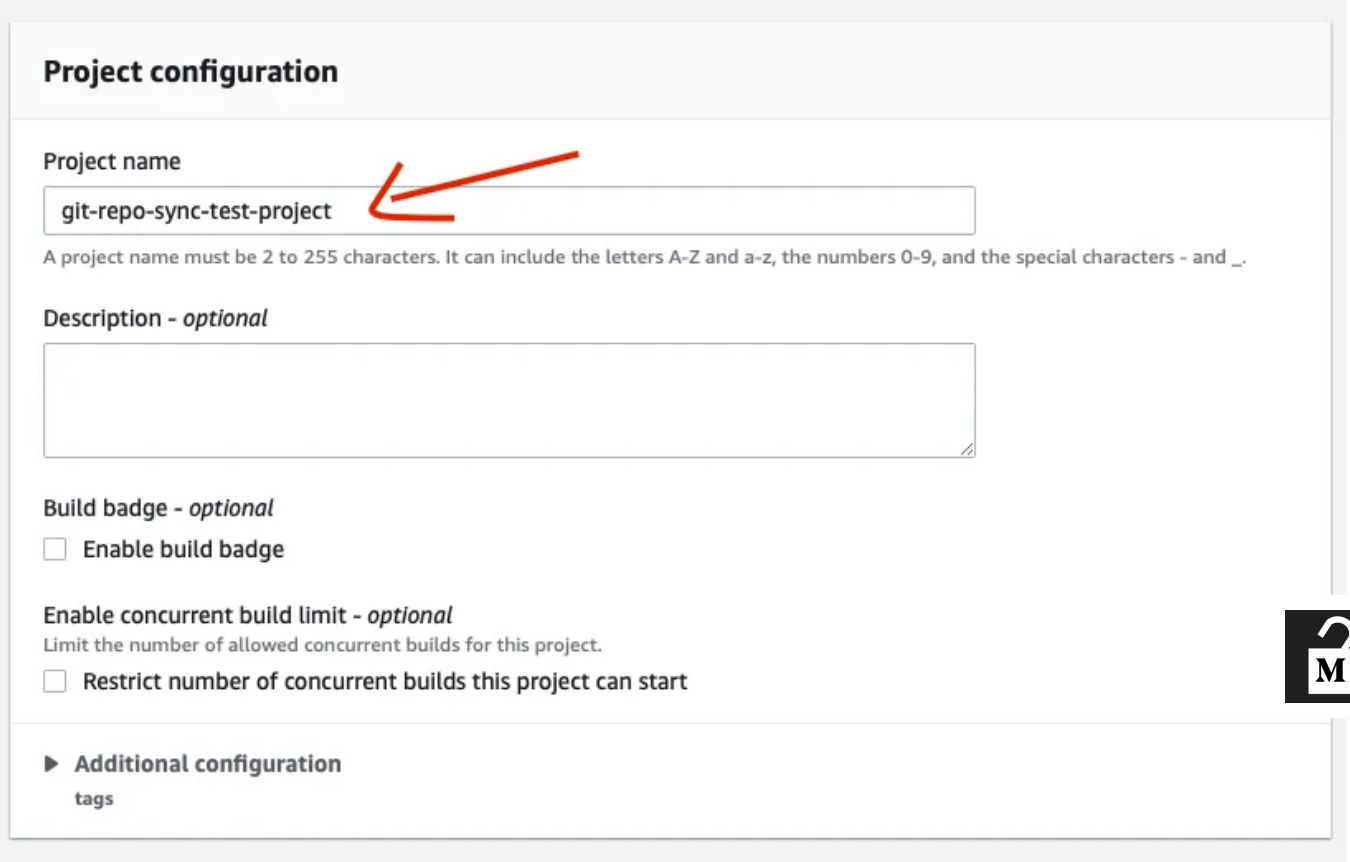
So the steps will first add your GitHub provider connection to your AWS account using CLI and after that, we want to create the AWS CodeBuild file there will do the sync between AWS CodeCommit and GitHub repositories every time there are changes on your source repositories on AWS CodeCommit using Event Bridge and a Lambda

Setup your AWS CodeBuild inline build spec YAML and run your sync inside the AWS Web Console manually

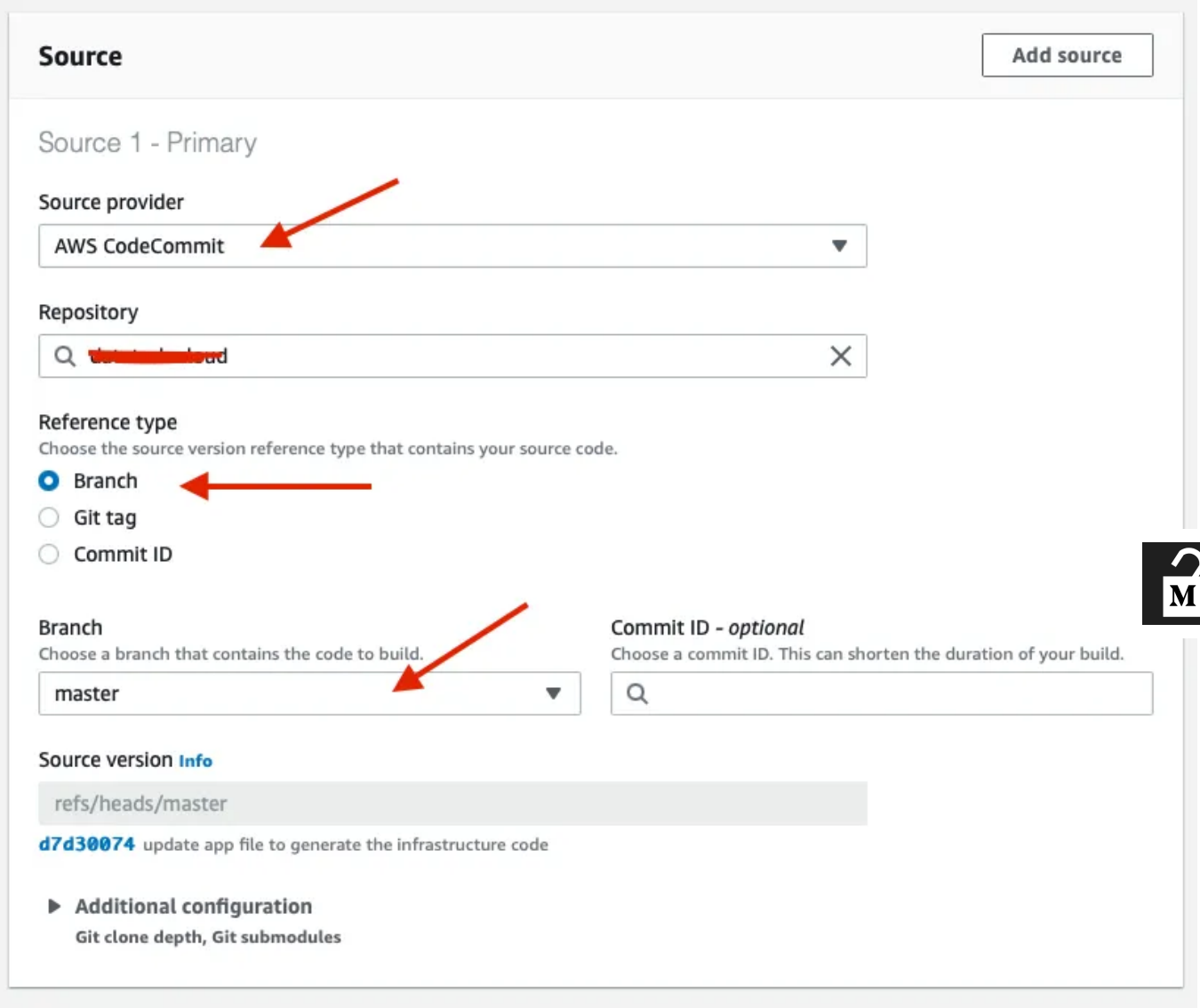
Go to AWS CodeBuild in your console, create your project and prepare the configuration for our build project.



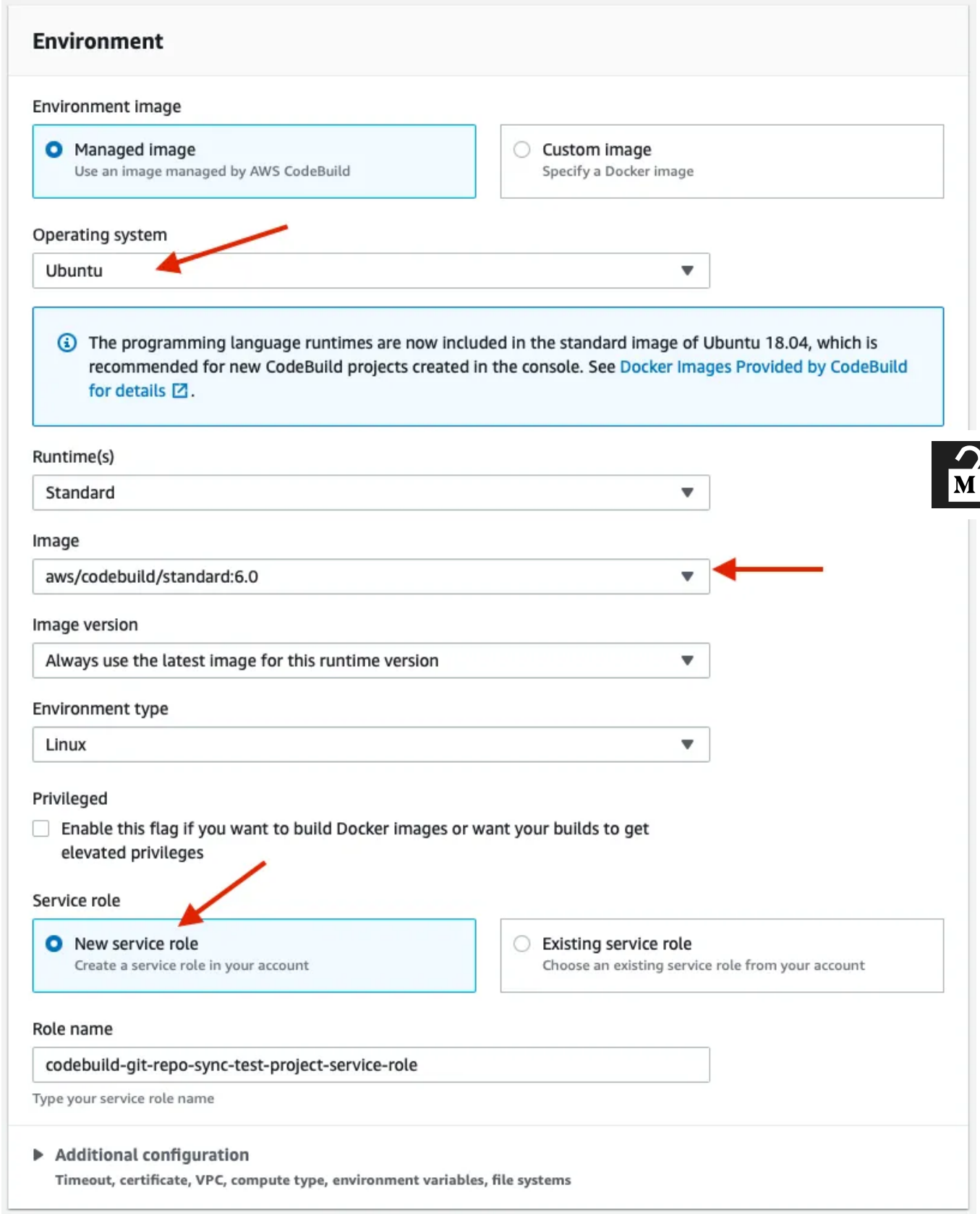
First we need to fill out the project configuration settings with a project name, if you want a build badge you can check it on, but for now we are not need this option.



Now we need to select our source provider (AWS CodeCommit) and find the repository you want to sync from an AWS CodeCommit repository to a GitHub repository, after you have selected the repository select Branch on reference type and select your main branch in my case it called “master”

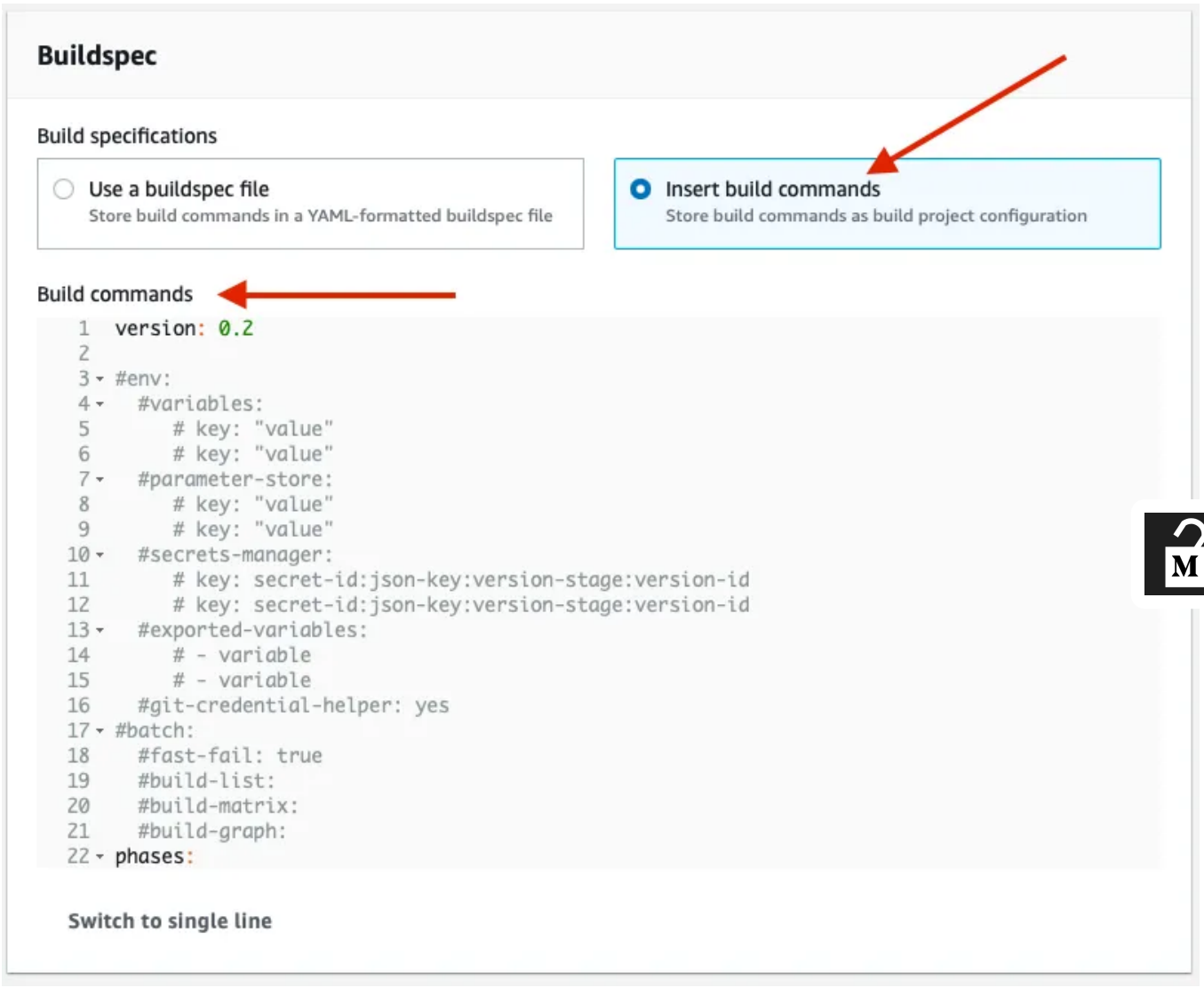


For our environment, we want to use **Ubuntu** as the operating system and the image is **aws/codebuild/standard:6.0** for now and late our environment create a new service role, you can always create your own role and then change it but in the beginning, it's easier to late CodeBuild create this role automatically when you create the project.



In Buildspec you should use **Insert build commands**, then switch to an editor so its

looks like the images below



when you have this editor ready you can paste my build spec YAML snippet into the editor

```
version: 0.2

env:
  git-credential-helper: yes

phases:
  install:
    commands:
      - echo "STARTING PYTHON INSTALLATION"
      - pip install git-remote-codecommit

  pre_build:
    commands:
      - echo CLONE DIRECTORIES
      - mkdir /usr/bin/repo
      - cd /usr/bin/repo
      - git clone --mirror codecommit::{region}://{repo} {repo}

  build:
    commands:
      - cd /usr/bin/repo/{repo}
      - git remote add --mirror=fetch github https://<token>@github.com/{username}/{repo}

  post_build:
    commands:
      - git fetch origin
      - git push github --all
```

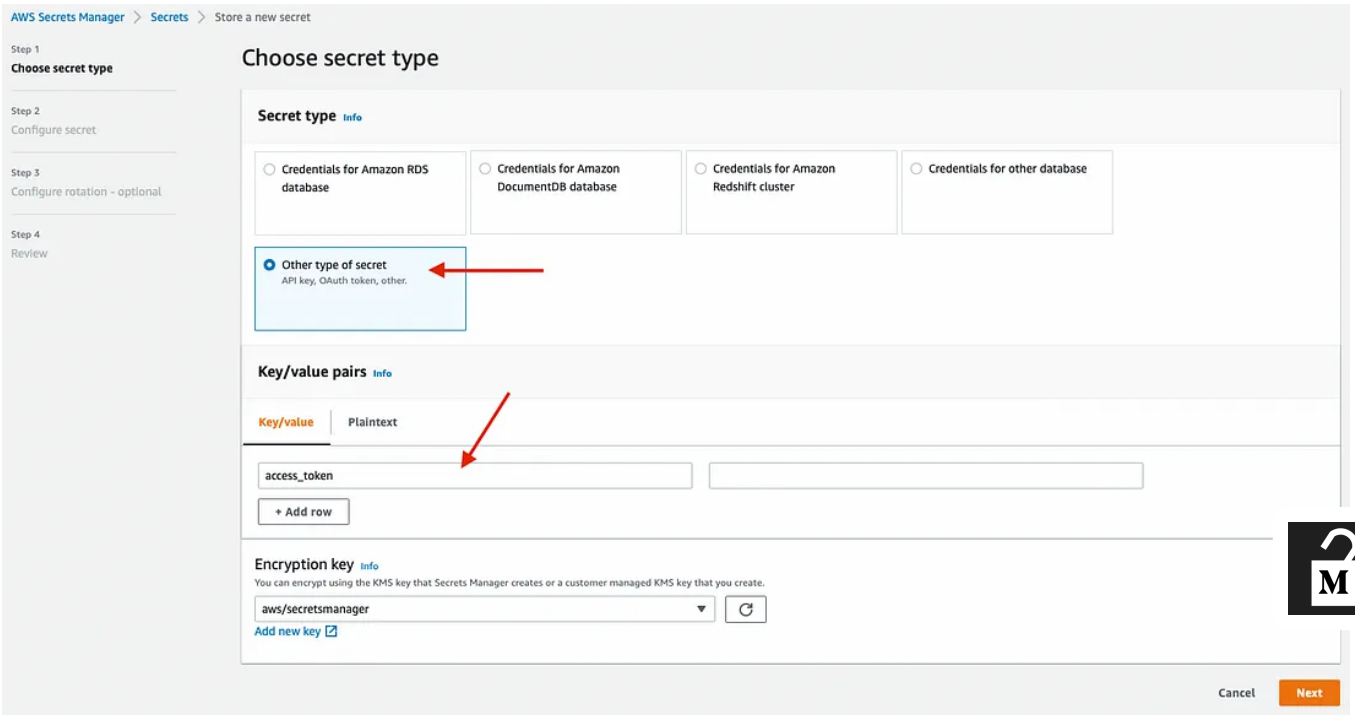
buildspec.yml hosted with ❤ by **GitHub** **view raw**

rtmedThis snippet will install the git-remote-codecommit python package, clone your CodeCommit repository with the mirror flag, then we are going into the folder and add our GitHub remote path with the token and mirror tag with fetch at the end of it we are fetching from the origin (CodeCommit) and push to GitHub repository after it.

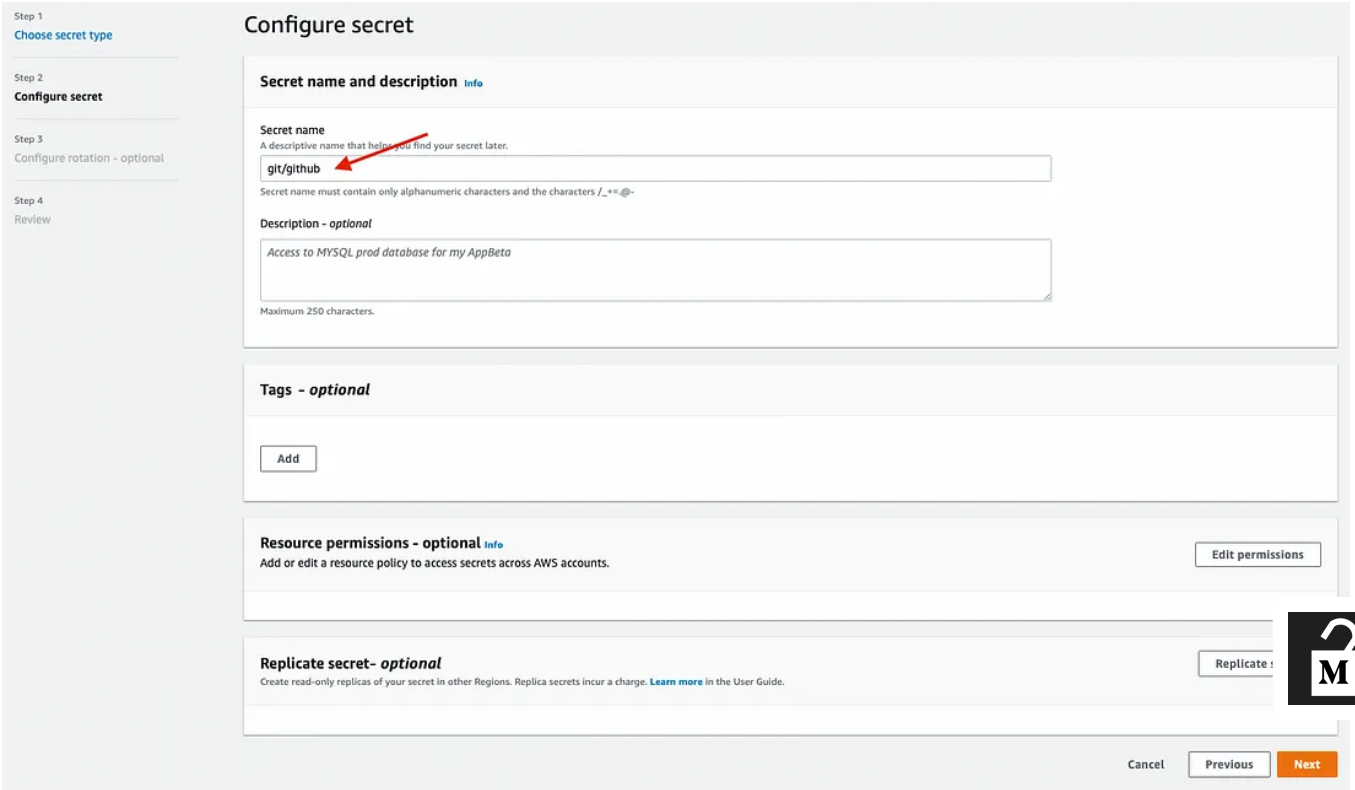
Add your GitHub access token to Secret Manager

Using Secret Manager will be the best way to keep your GitHub access token saved and easy to rotate if needed, if you keep your access token static it will be hard to change it automatically if you build an access token rotation system later or if something happens and you need to change it quickly.

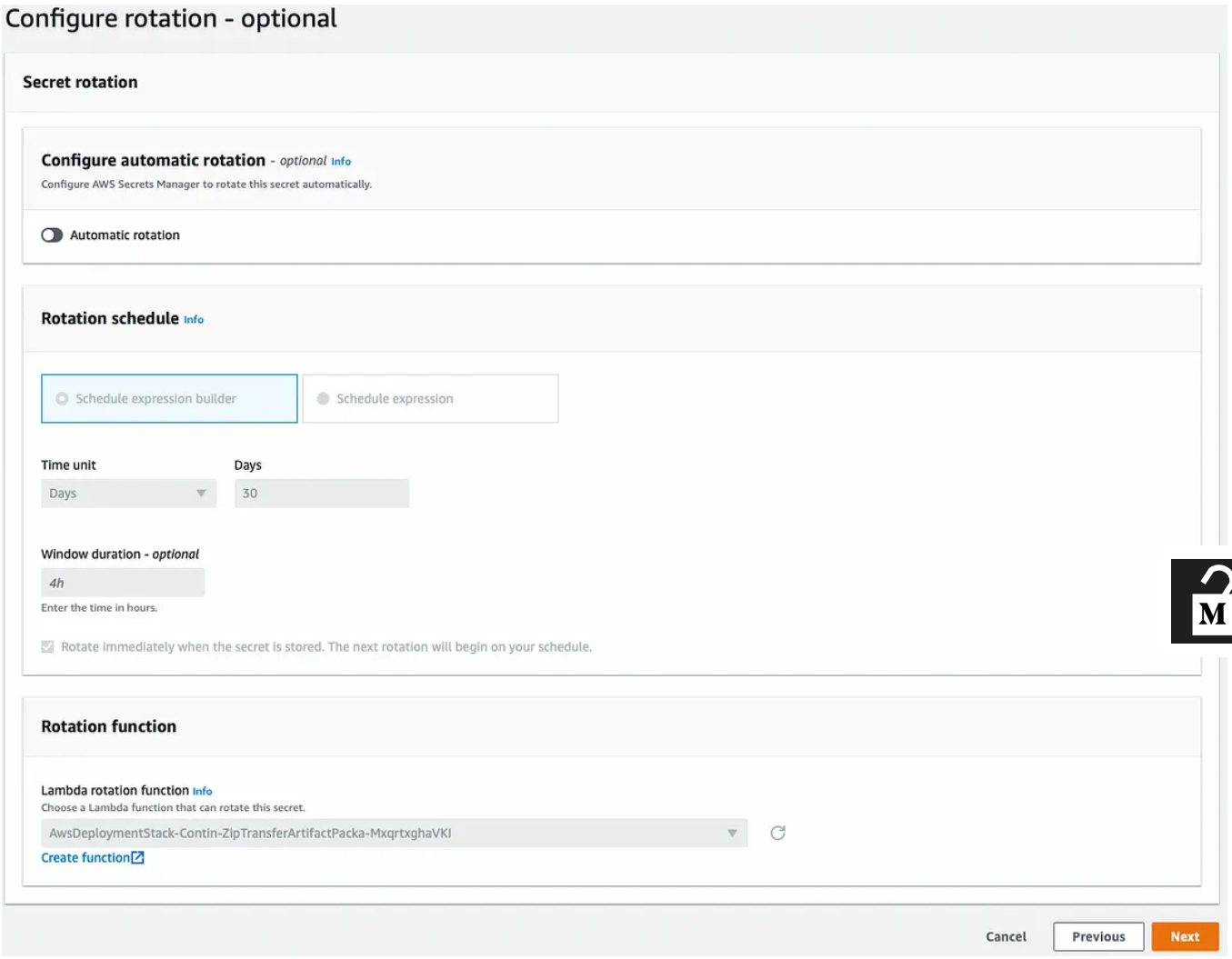
First, let's go into Secrets Manager and create our Secret with Secrets Manager.



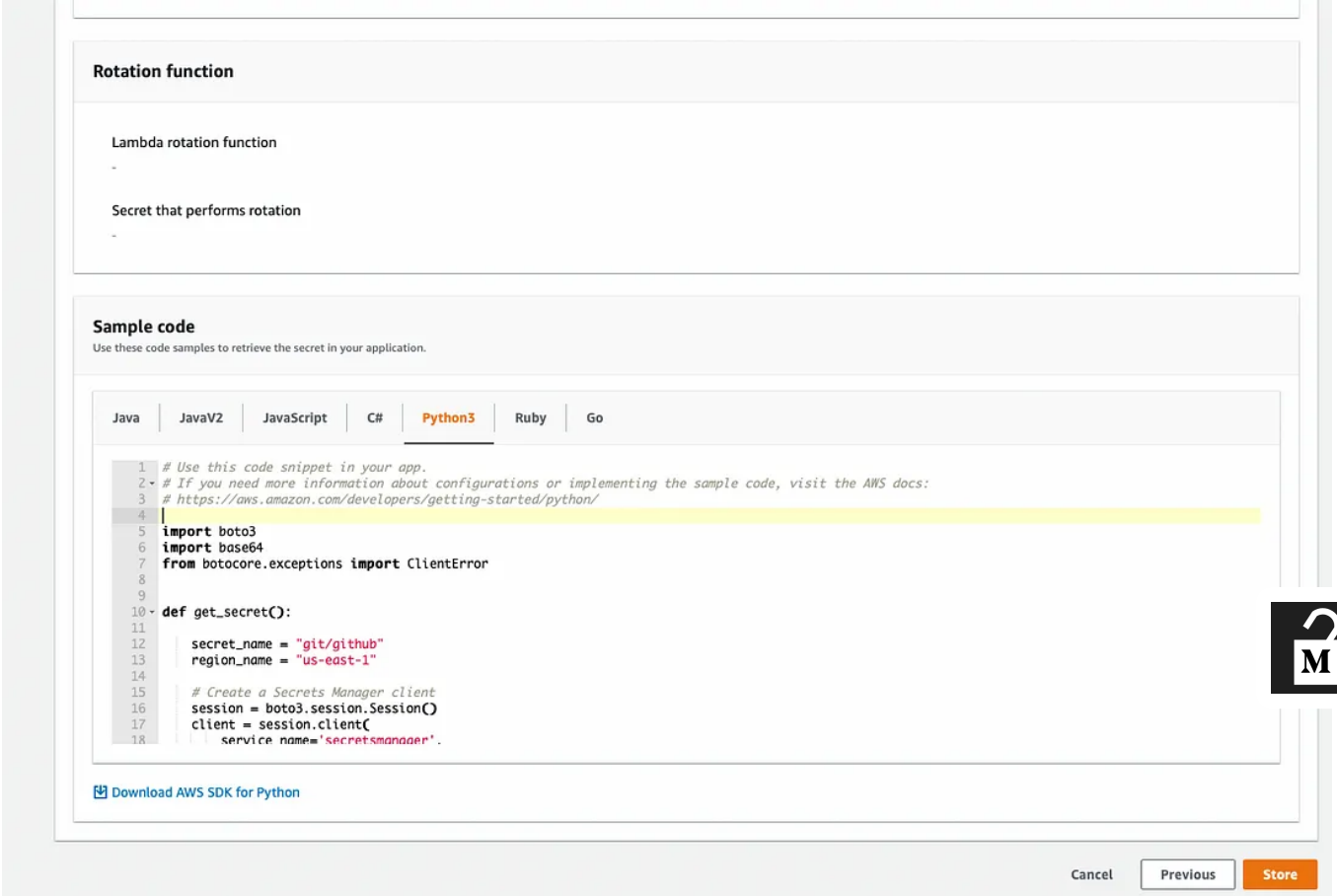
The secret type we want to pick Other type of secret to store our access token for GitHub, then the key name is access_token, and paste your token string inside the value field on the right side, after clicking next.



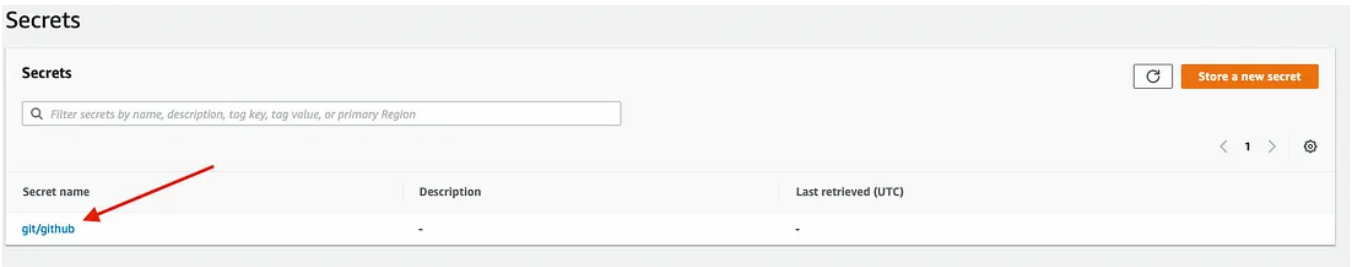
Now select a secret name, I pick **git/github** so I'm ready to use this path for multi GIT providers if needed, when you have done it goes to the next step.



We are not gonna use rotation for this right now, so let's just skip this step with click next.



On the review pages at the bottom of the pages, you can quickly copy code samples if you want to speak with the Secrets Manager from the SDK.



When it's storage you can see it in your overview for Secrets Manager, if not then click refresh one time, sometimes the can take a few seconds.

Show we are ready to demonstrate the CLI command to get it out again so we can test it's working before we add it to our AWS CodeBuild process.

```
pnk@MacBook-Pro aws-github-source-provider % aws secretsmanager get-secret-value --secret-id git/github --region us-east-1
{
    "ARN": "arn:aws:secretsmanager:us-east-1:788962042046:secret:git/github-A9DD22",
    "Name": "git/github",
    "VersionId": "29b39d89-f8f4-4949-b248-e507e7488cc8",
    "SecretString": "{\"access_token\":\"\"}",
    "VersionStages": [
        "AWSCURRENT"
    ],
    "CreatedDate": "2022-08-26T11:30:01.829000+02:00"
}
```

As you can see my SecretString is empty for my test here, in the real world it will contain my GitHub access token.

Let's take only the SecretScring object out using jq in our terminal so it's easier to work with and convert or string to JSON so it's easier grep our GitHub access token out.

in my case its returns “test” I have to change my value to “test” for our access_token so we have something to print out.

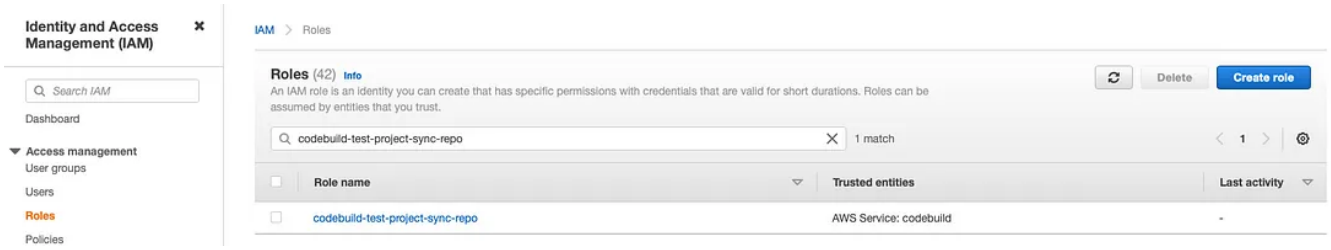
```
pnk@MacBook-Pro aws-github-source-provider % aws secretsmanager get-secret-value --secret-id git/github --region us-east-1 | jq
 -r ".SecretString" | jq -r ".access_token"
test
```

Now you are ready to put this command into our AWS CodeBuild process to replace our static access token.

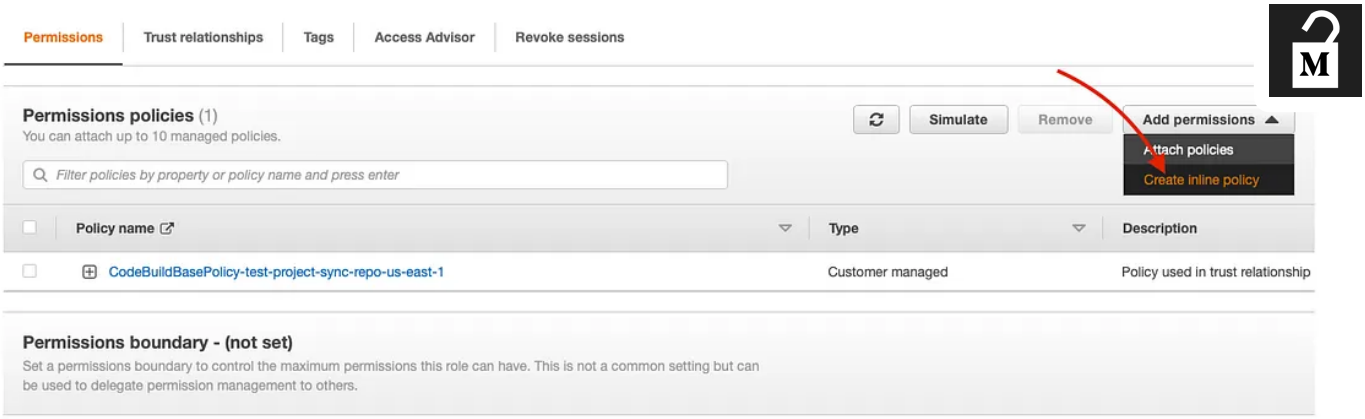
When you now run the build in AWS CodeBuild you will meet the error AccessDeniedException, its because your role did not allow to get access from Secrets Manager

```
278
279 An error occurred (AccessDeniedException) when calling the GetSecretValue operation: User: arn:aws:sts::788962042046:assumed-role/codebuild-test-project-sync-repo/AWSCodeBuild-179c1fe9-75cc-455f-9a9c-
    ffd27a81328e is not authorized to perform: secretsmanager:GetSecretValue on resource: git/github because no identity-based policy allows the secretsmanager:GetSecretValue action
280
```

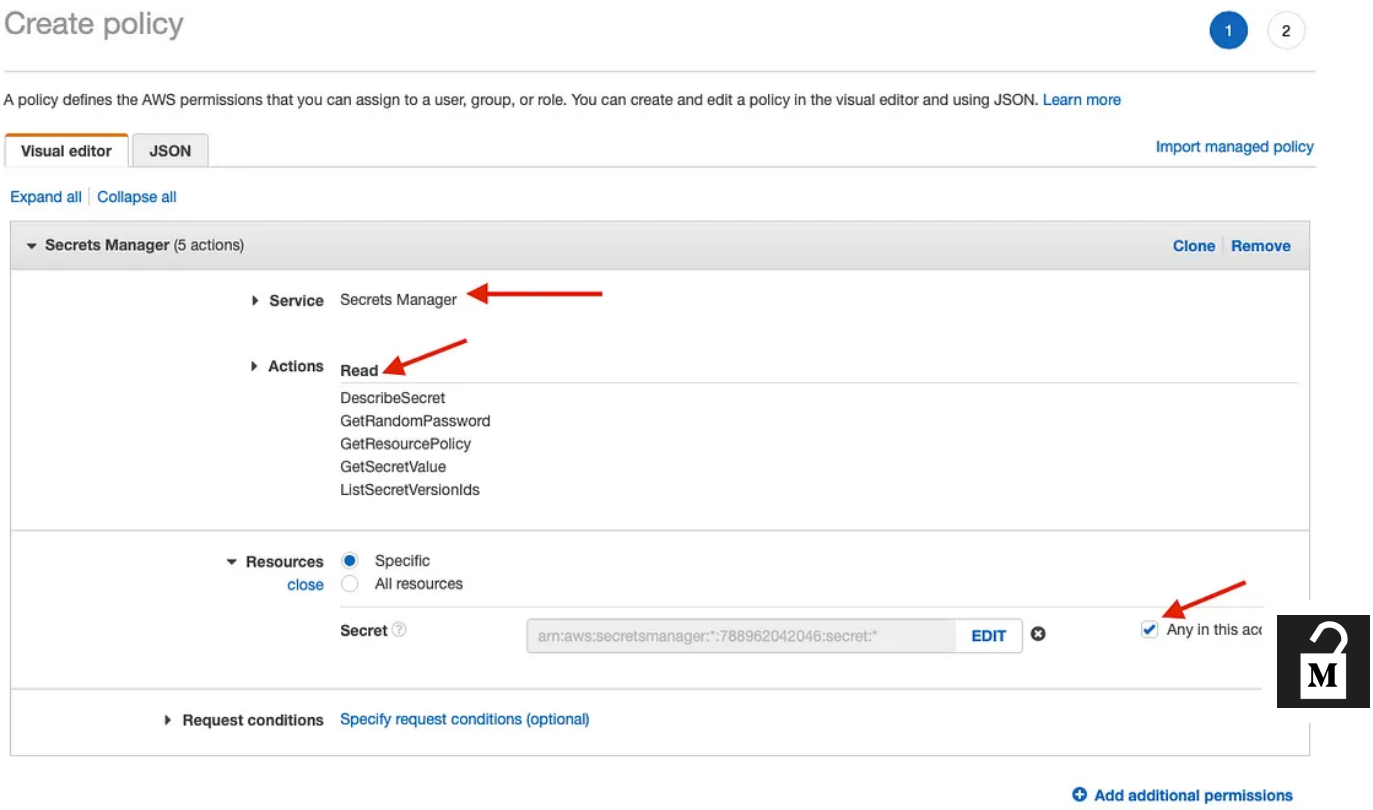
So let's go inside IAM and find our role and give it the needed access, and run the process one more time.



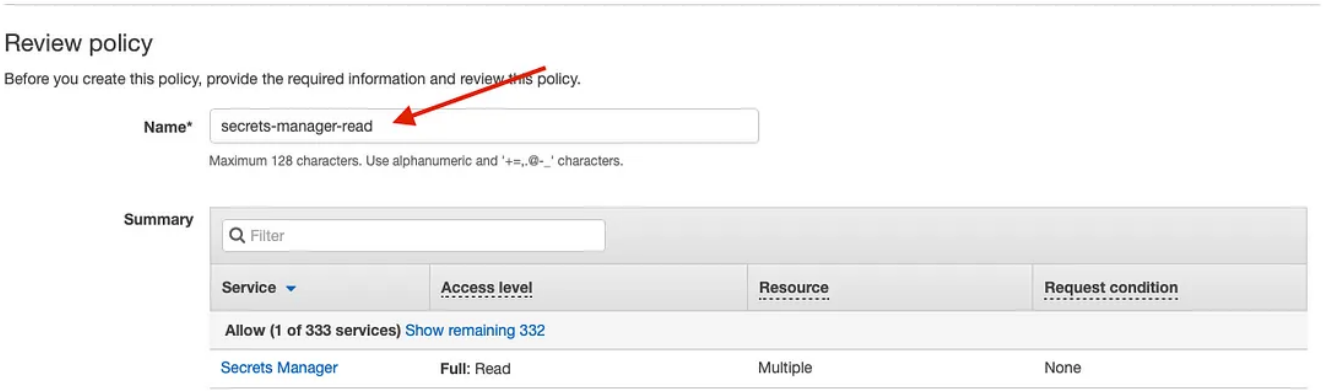
Select your role and go to the permissions tab and click “Add permissions > Create inline policy”



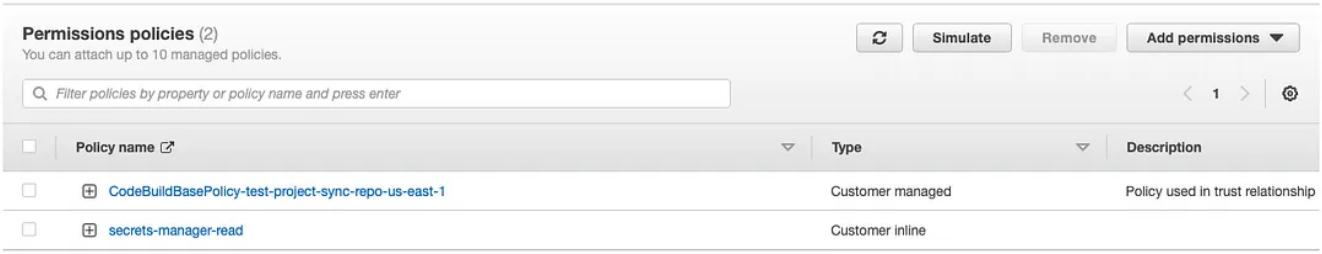
When you are creating the inline policy you should select Secrets Manager services and only allow Read access to it, then select Any in this account under resources.



Pick a simple name so you can find it for other roles there need reading from Secrets Manager, I name it secrets-manager-read in my case.



When you finished and have added the inline policy it should be showing with the already existing role.



After you run the AWS CodeBuild again, you will see its success and everything should work as you expect when you trigger this build manually.

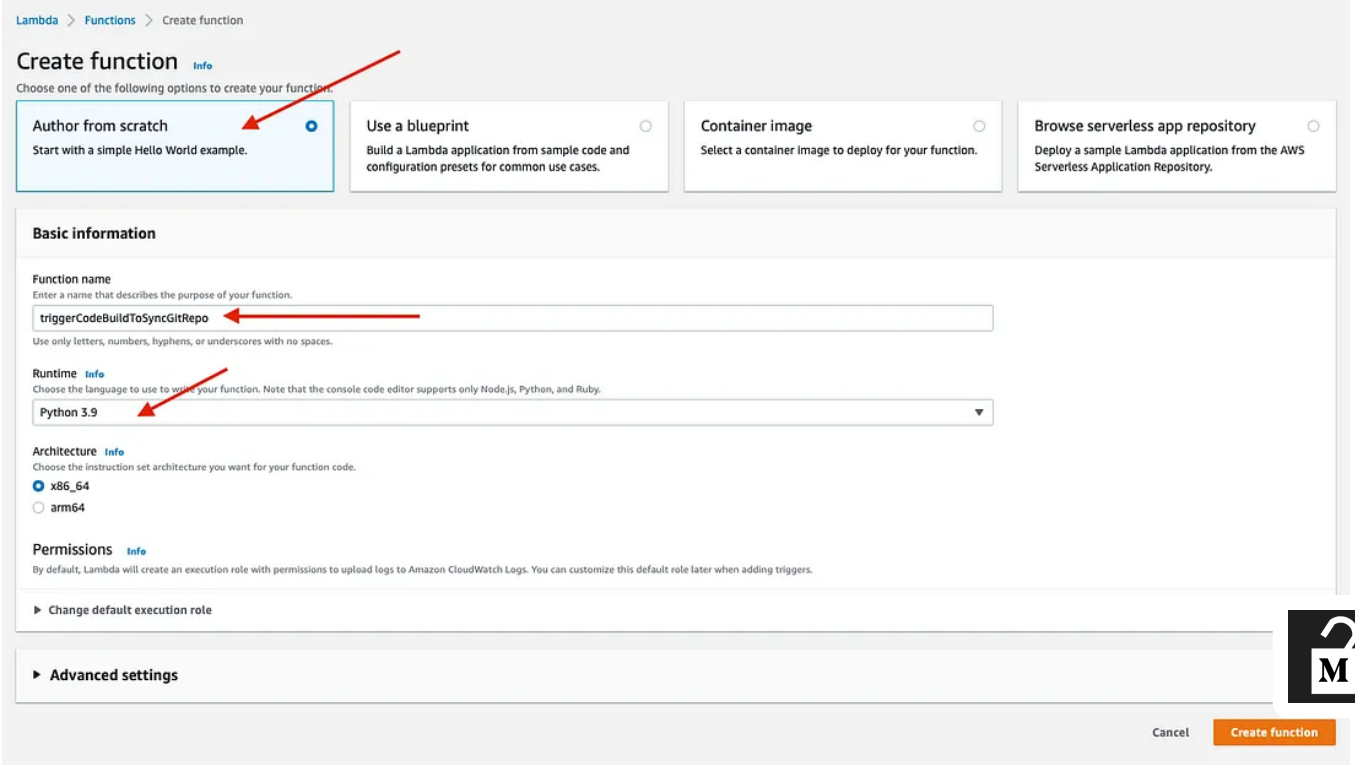
```
271
272 [Container] 2022/08/26 10:02:35 Phase complete: PRE_BUILD State: SUCCEEDED
273 [Container] 2022/08/26 10:02:35 Phase context status code:  Message:
274 [Container] 2022/08/26 10:02:35 Entering phase BUILD
275 [Container] 2022/08/26 10:02:35 Running command cd /usr/bin/repo/datatask-cloud
276
277 [Container] 2022/08/26 10:02:35 Running command git remote add --mirror=fetch github https://$(aws secretsmanager get-secret-value --secret-id git/github | jq -r ".SecretString" | jq -r
    ".access_token")@github.com/parisnakitakejser/datatask-cloud.git
278
279 [Container] 2022/08/26 10:02:37 Phase complete: BUILD State: SUCCEEDED
280 [Container] 2022/08/26 10:02:37 Phase context status code:  Message:
281 [Container] 2022/08/26 10:02:37 Entering phase POST_BUILD
282 [Container] 2022/08/26 10:02:37 Running command git fetch origin
283
284 [Container] 2022/08/26 10:02:37 Running command git push github --all
285 Everything up-to-date
286
287 [Container] 2022/08/26 10:02:37 Phase complete: POST_BUILD State: SUCCEEDED
288 [Container] 2022/08/26 10:02:37 Phase context status code:  Message:
289
```

Finished your automatic using Event Bridge and Lambda to sync automatically

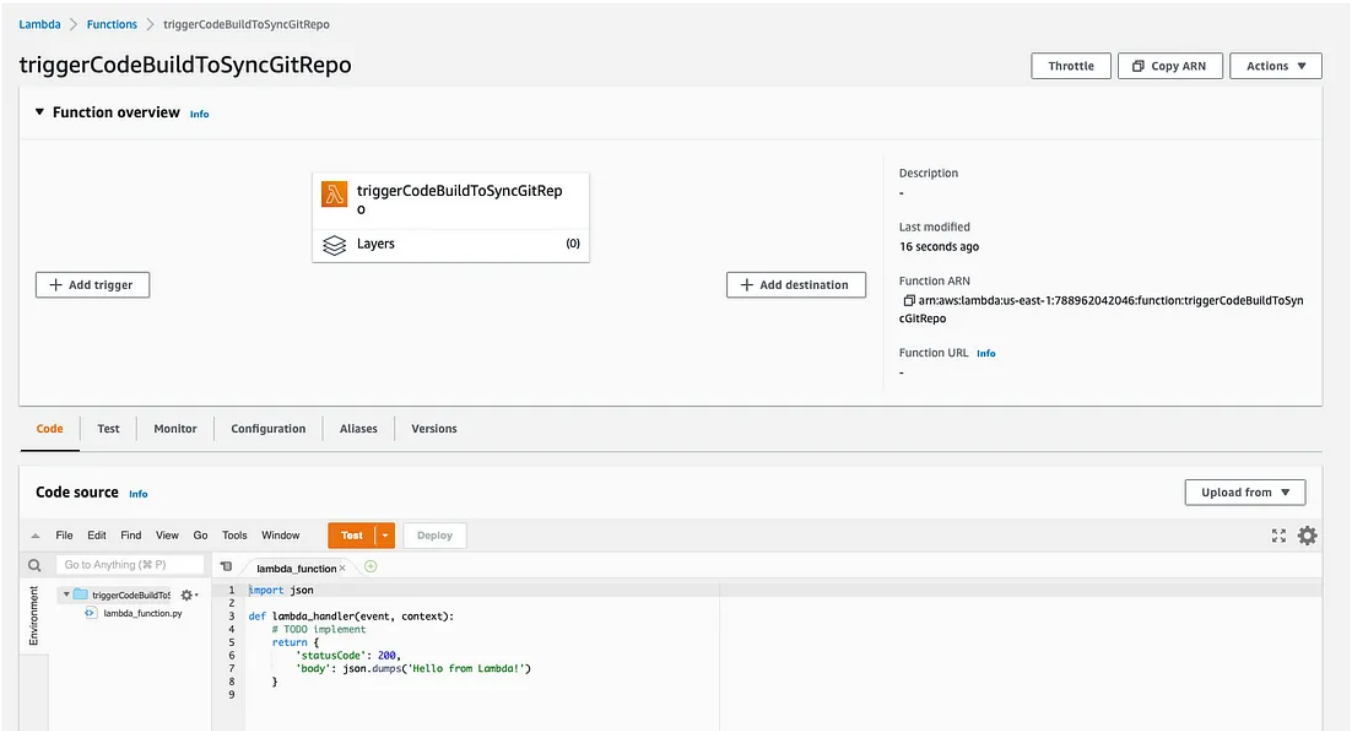
What we will do in this section is create a trigger on every event from the AWS CodeCommit repository and send the event a Lambda function there will start our AWS CodeBuild process to sync between AWS CodeCommit and your GitHub repository.

It will be very simple, and we will don't focus on SNS and SQS in this area, I think it's something you should challenge yourself with after you know how these steps are working.

Start to create your Lambda function there can trigger our AWS CodeBuild from our trigger, so go to Lambda > Functions > Create function

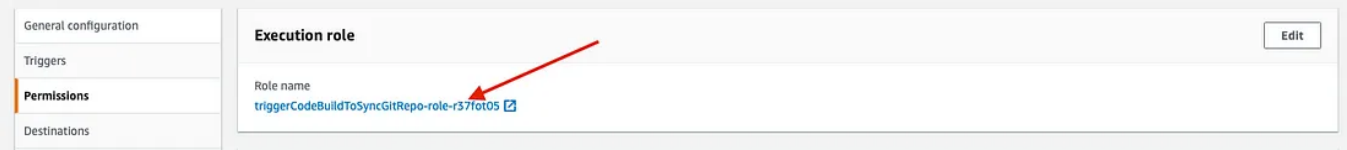


When you create the function, we will create it from scratch and use the Python 3.9 runtime, I have picked a meaningful name so it's easier to find after.

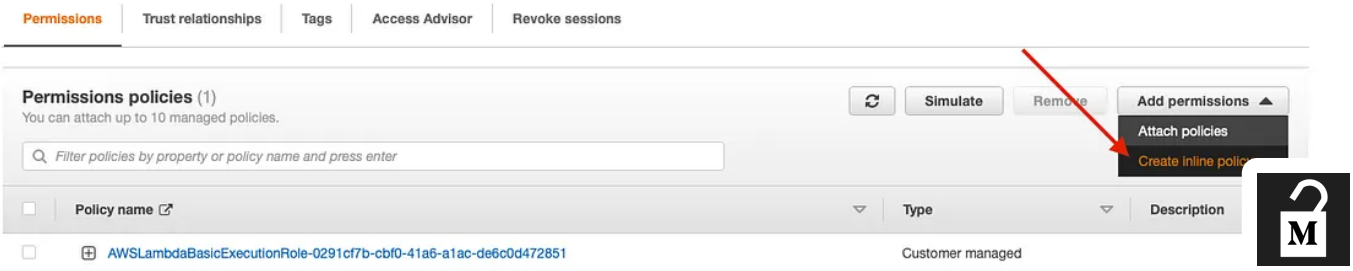


Now you should end up with your Lambda function screen, now we need to allow this

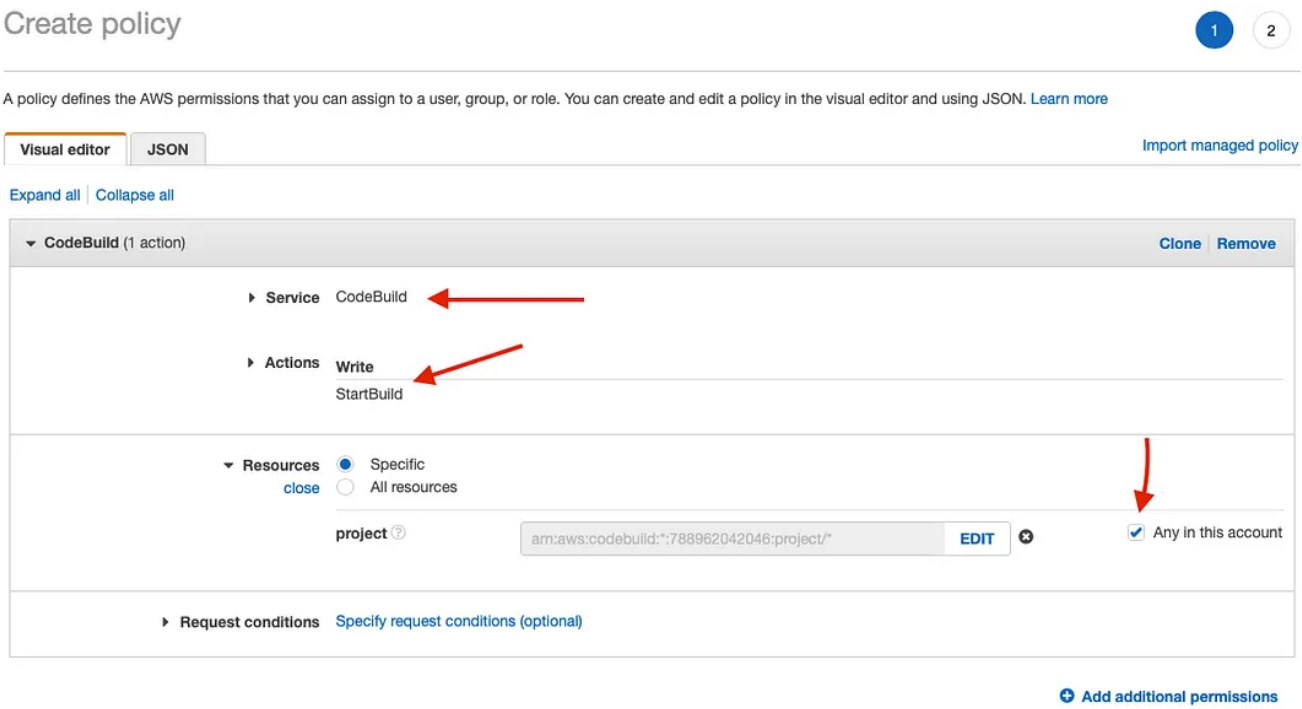
function to start our AWS CodeBuild by going to Configuration > Permissions > Execution Role here we need to edit this Execution Role by clicking on the name



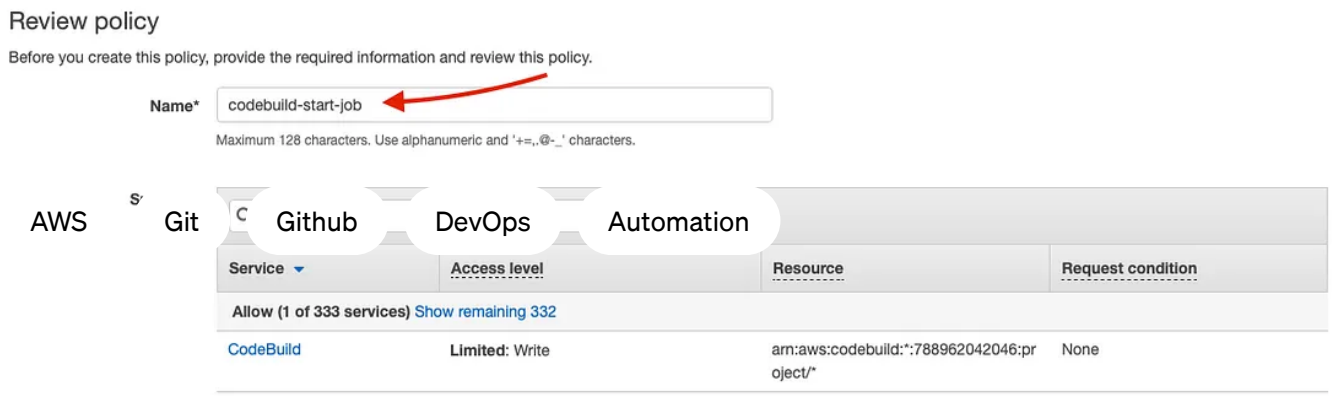
It's open in a new tab and we need to “Create inline policy” for this role



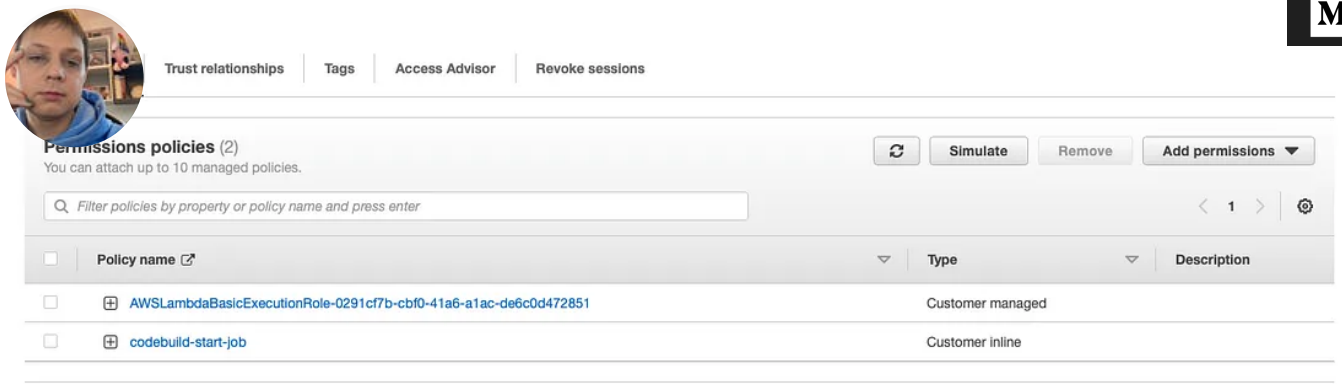
Select CodeBuild as a service and with actions you pick Write > StartBuild to allow this role to start builds and then allow all resources in this account.



Give it a meaningful name eg. codebuild-start-job so it's easy to reuse for other role policy.

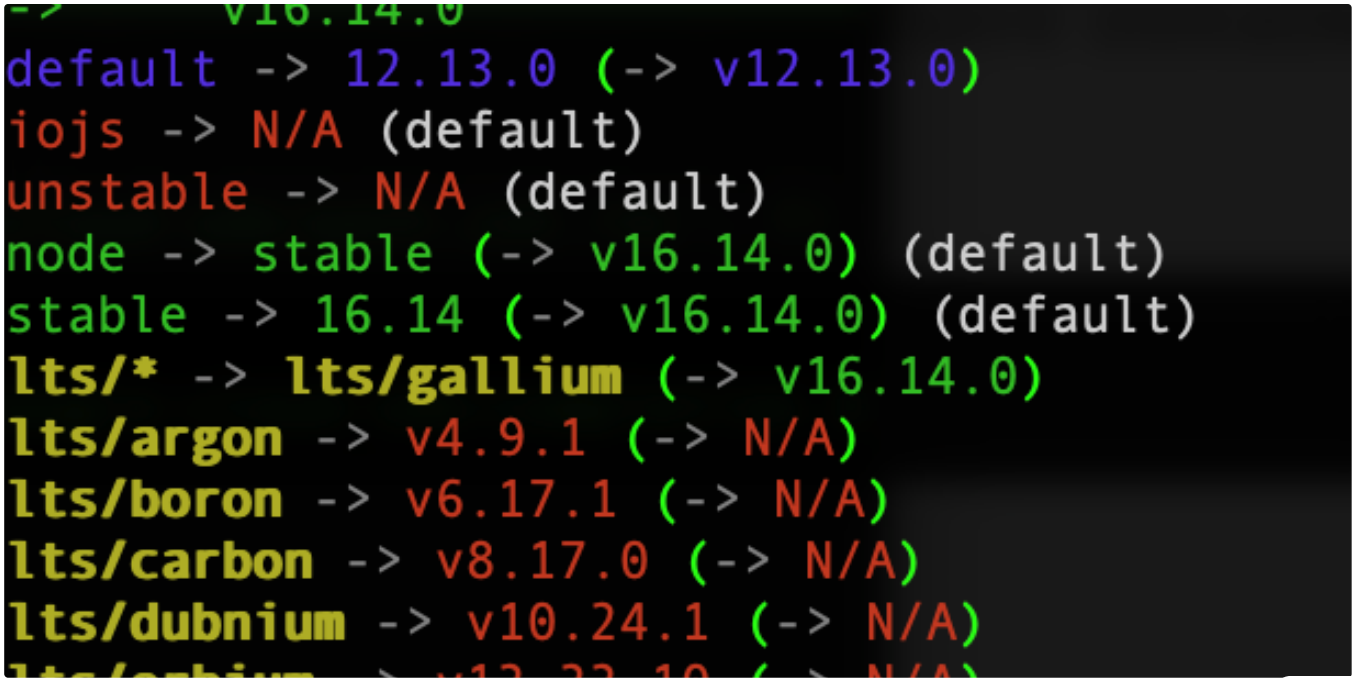


Then you end up with 2 permissions policies for our execute role there are attached to our Lambda function



Now go back to your Lambda function code editor and add the needed code to start the AWS CodeBuild process

More from Paris Nakita Kejser



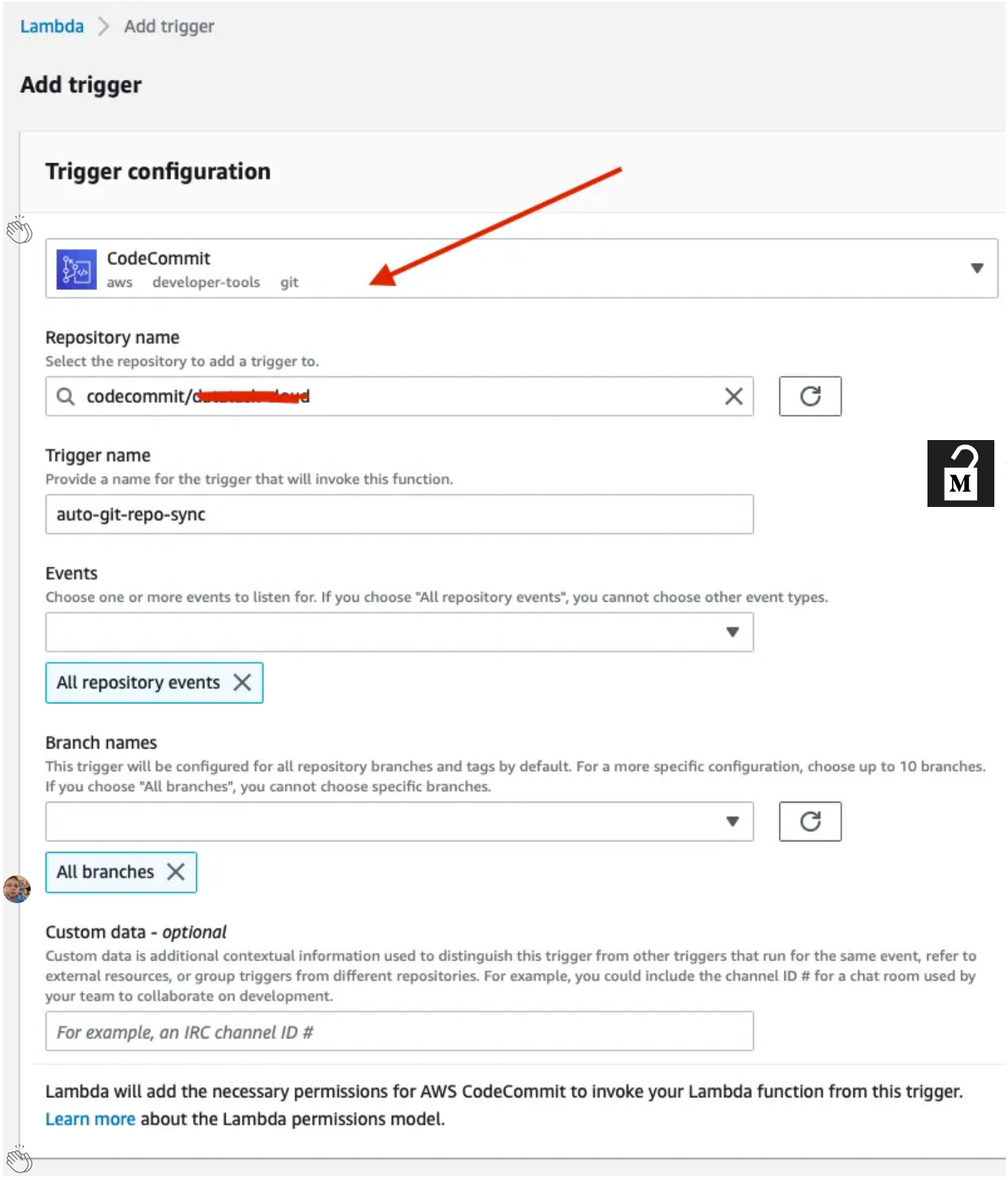
Paris Nakita Kejser in DevOps Engineer, Software Architect and Software Developering

How To Install NVM (Node Version Manager) on macOS with Homebrew

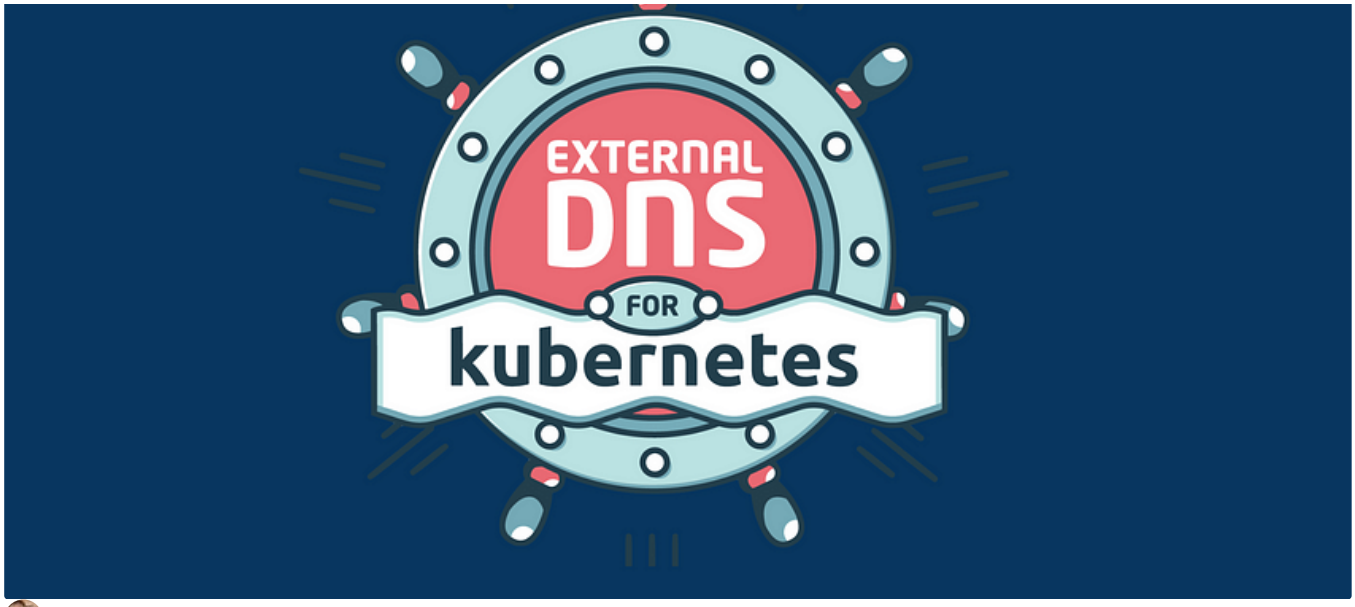
Node Version Manager (NVM) is used to develop NodeJS applications in multi versions, if you install NodeJS native on your system you will...

· 2 min read · Mar 1, 2022

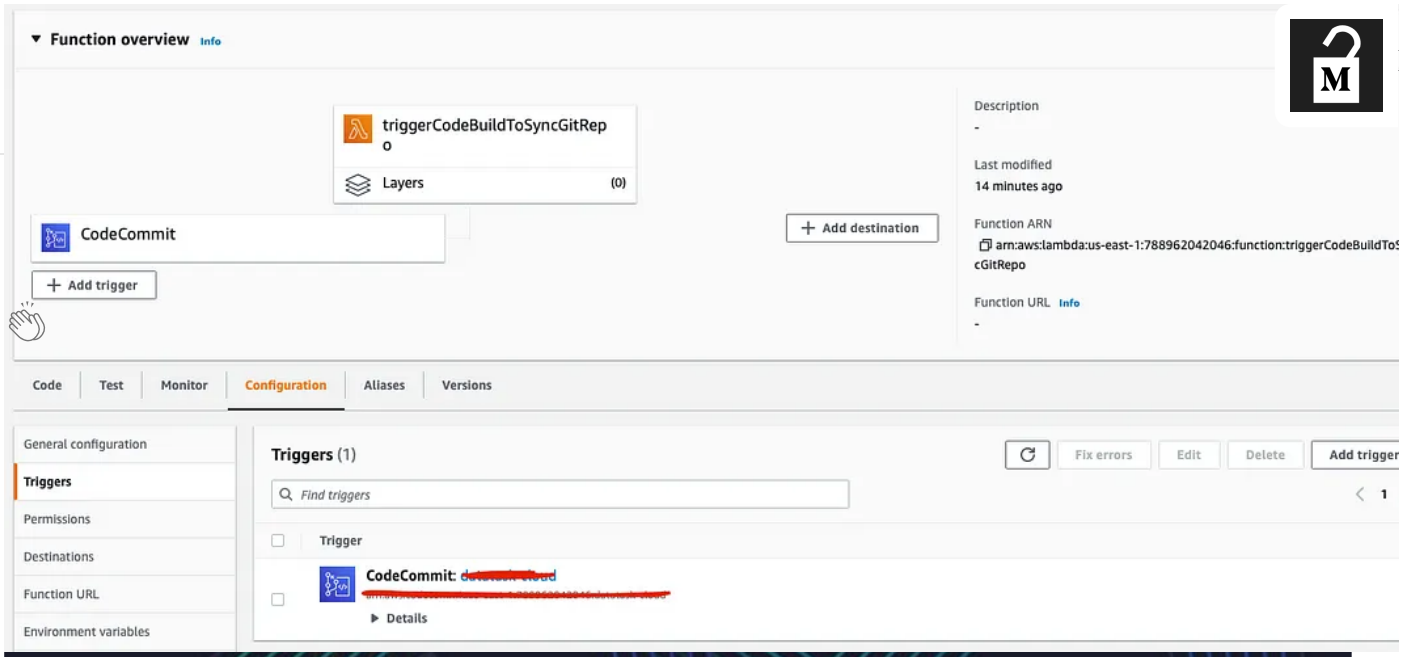
Paris Nakita Kejser in DevOps Engineer, Software Architect and Software Developering



Now you should have the trigger listed on your Lambda function Triggers page

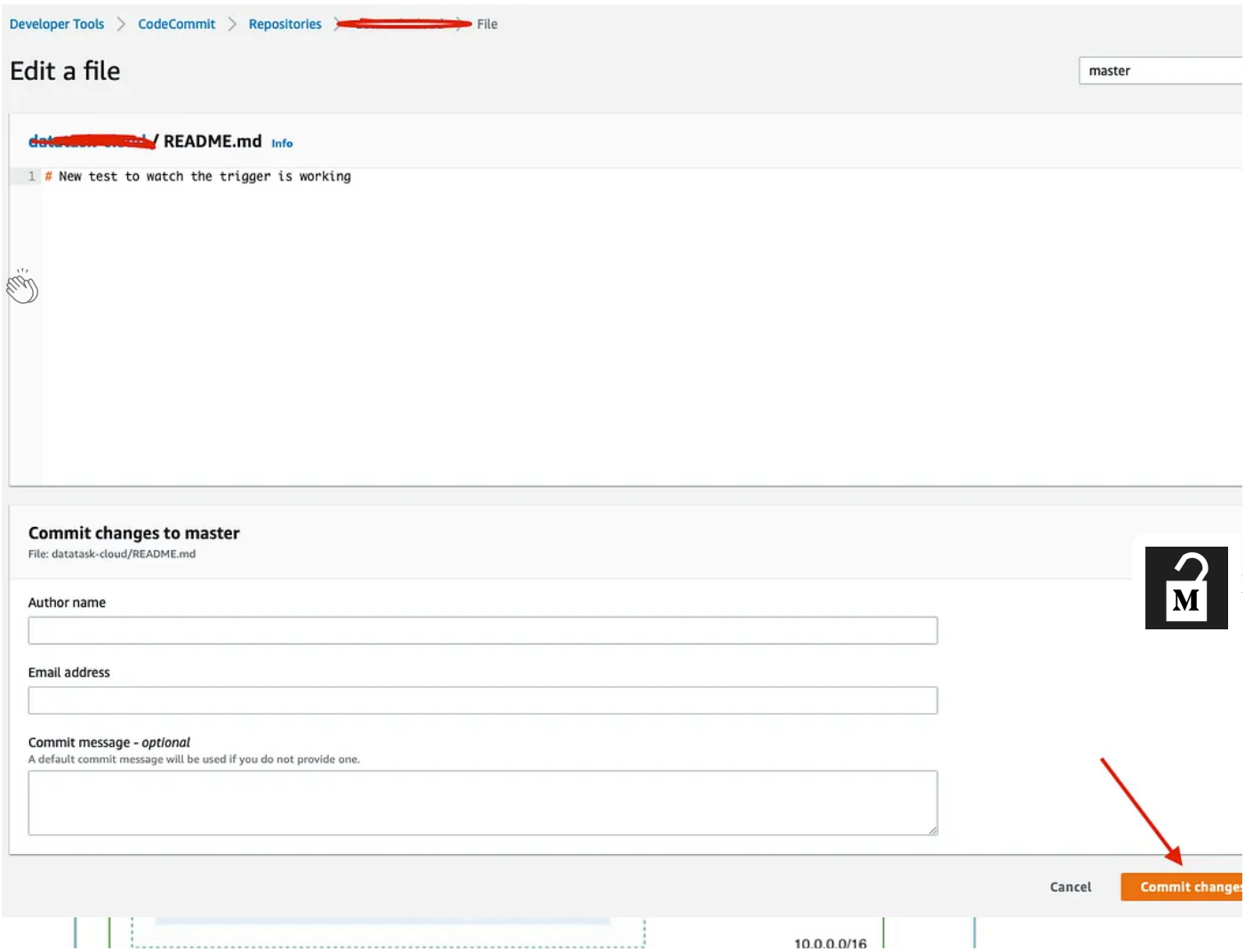


Paris Nakita Kejser in DevOps Engineer, Software Architect and Software Developering



le,

Dmit in DevOps.dev

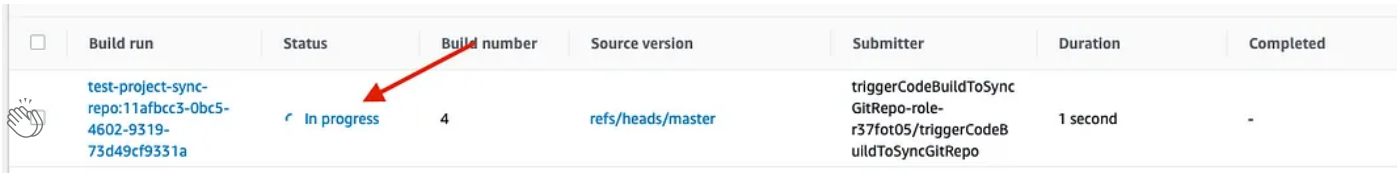


Fill out with the Author name, email, and a commit message after it clicks on “Commit changes” go to your AWS CodeBuild and your project then you can see the build status go in

headinthecloudsin AWS Tip

Deploy WordPress and MySQL with Terraform on AWS EC2 Like a Pro

Prerequisites:



Lists

Staff Picks

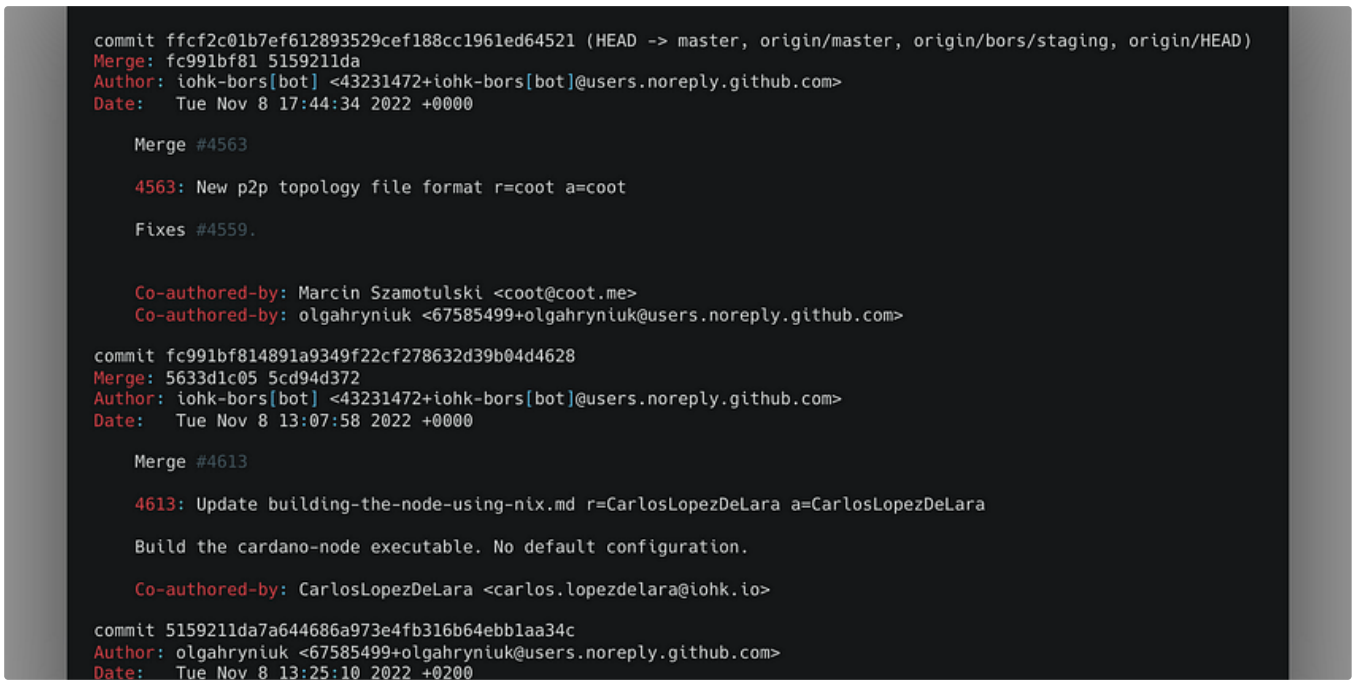
302 stories · 62 saves

CodeCircuit in CodeX

5 New DevOps Tools Expected to Make a Huge Impact in 2023

Enhance your DevOps collaboration with the latest and most innovative tools to shake up the industry in 2023.

· 6 min read · Mar 25



Jacob Bennett in Level Up Coding

Use Git like a senior engineer

Git is a powerful tool that feels great to use when you know how to use it.

· 4 min read · Nov 15, 2022

4.6K 51

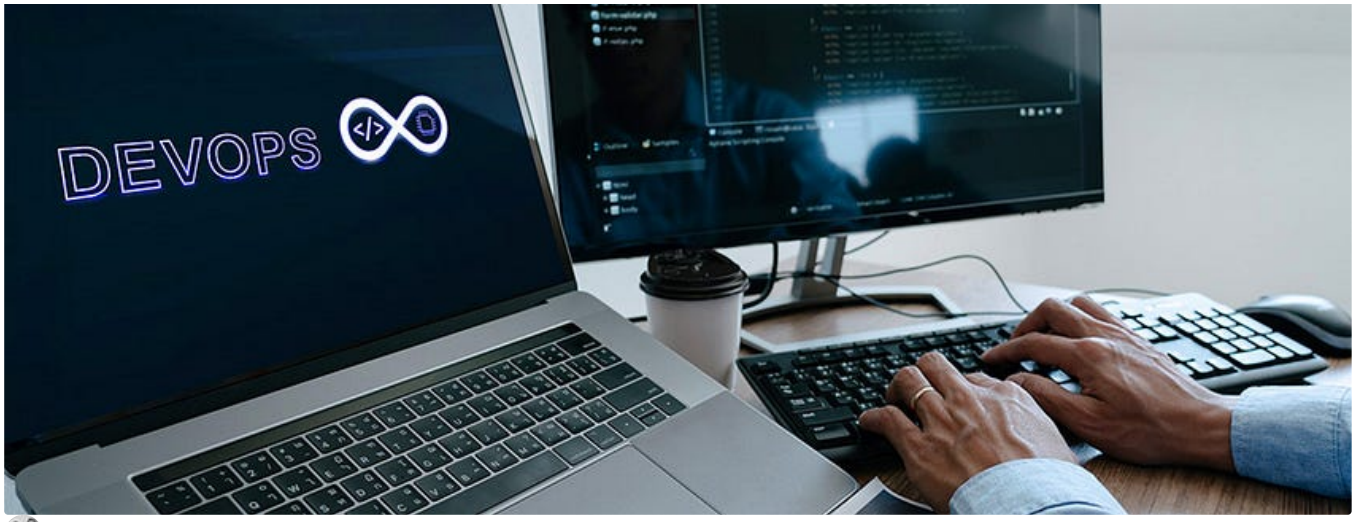
George Baidoo Jr.

CI/CD Pipeline: Deploy a Simple Application to an AWS EC2 Instance Via CodeDeploy

In this project I will be showing you how to deploy an application via a CI/CD Pipeline, to an AWS EC2 Instance via CodeDeploy & a AWS...

· 8 min read · Dec 29, 2022

16




headinthecloudsinAWS Tip

Full Stack DevOps Environment Setup Bash Script for Ubuntu, CentOS and MacOS

Introduction

· 8 min read · Jan 25


149


6

See more recommendations