LAB: Create first CodeBuild Project

You need:

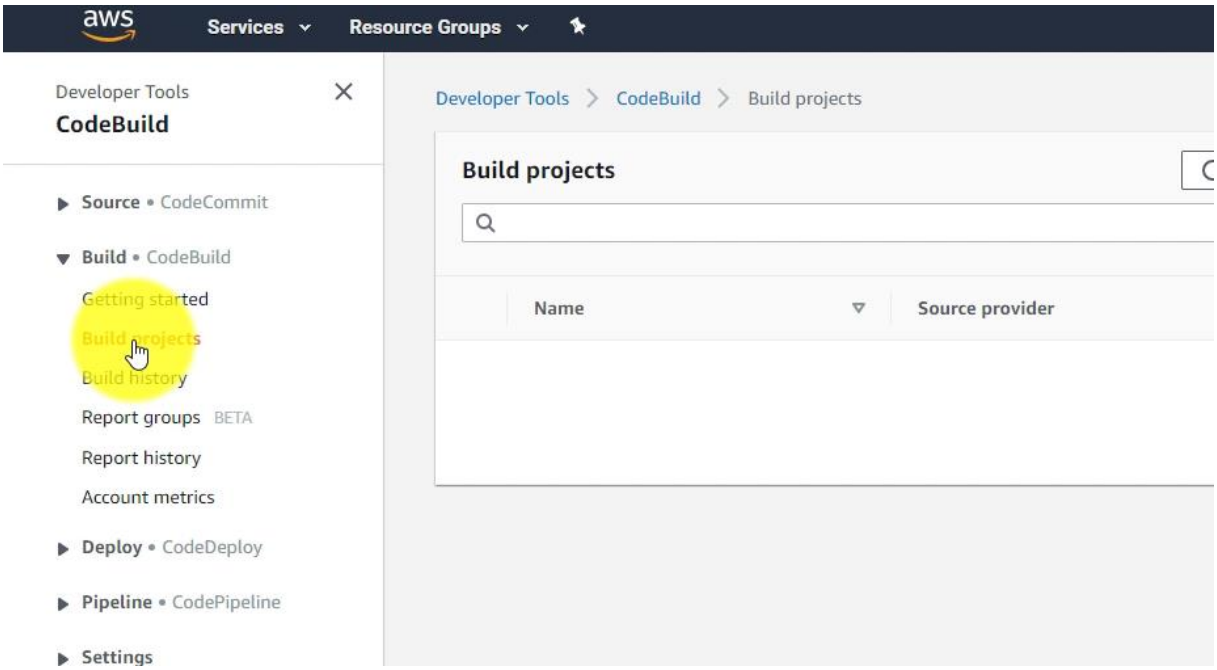
- An AWS Account
- Code in the CodeCommit Repository

Duration of the Lab: 30 Minutes.

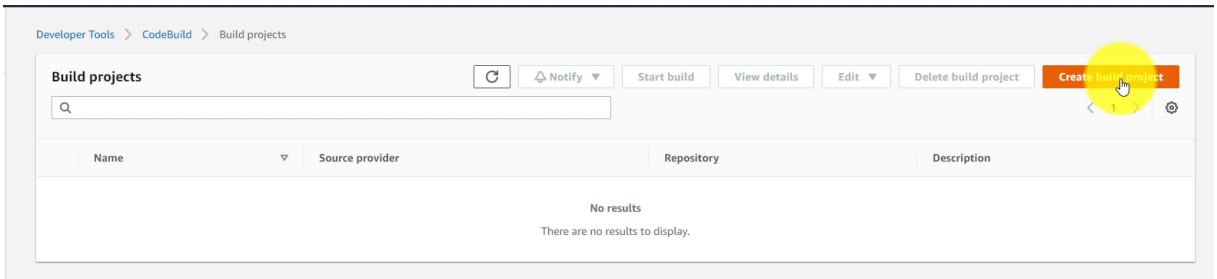
Difficulty: medium

Create a first CodeBuild Project

Open the CodeBuild section of the AWS Developer Tools Dashboard



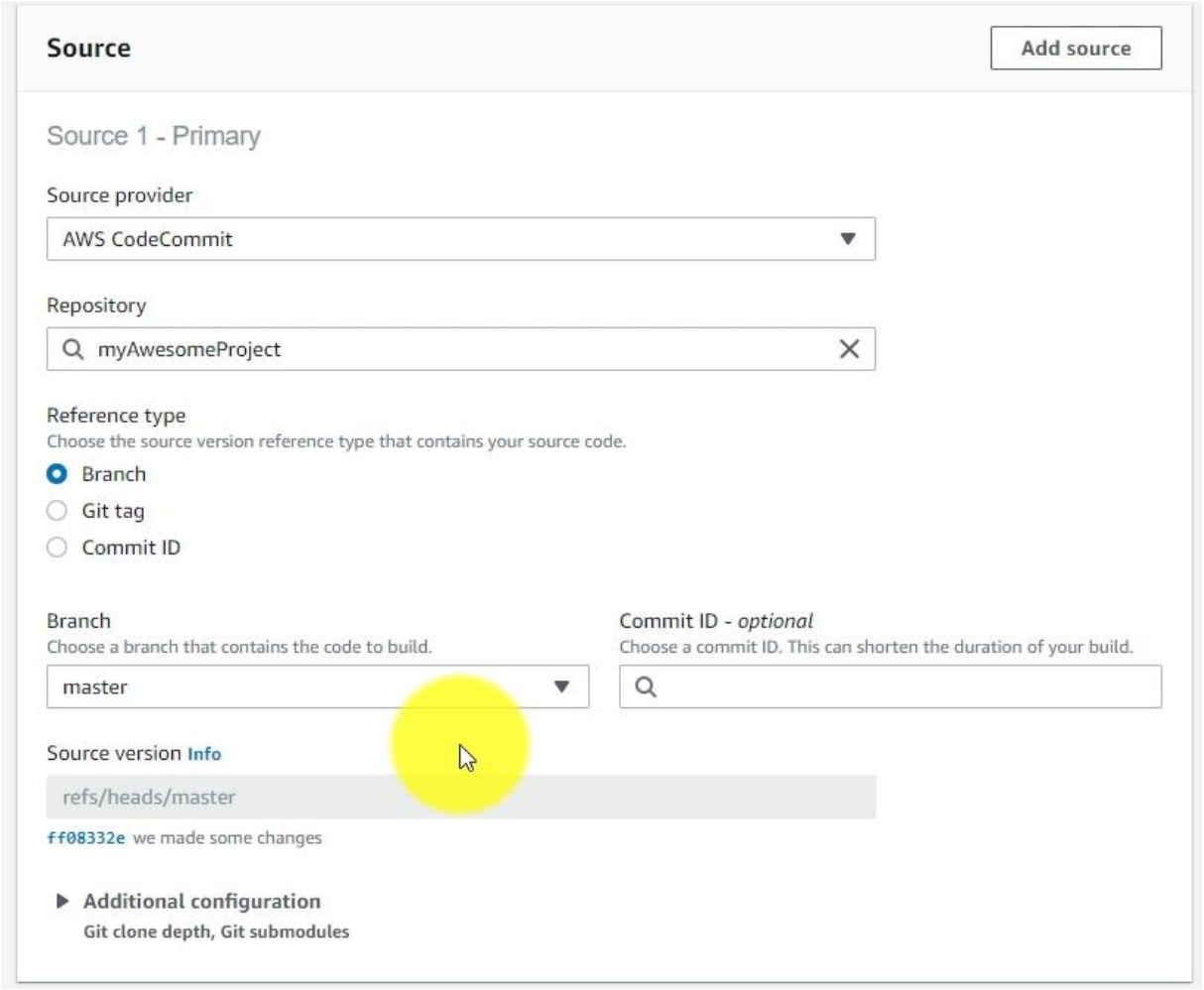
Create a new Project:



Give the Project a name of your choice, e.g. “buildMyAwesomeProject”.

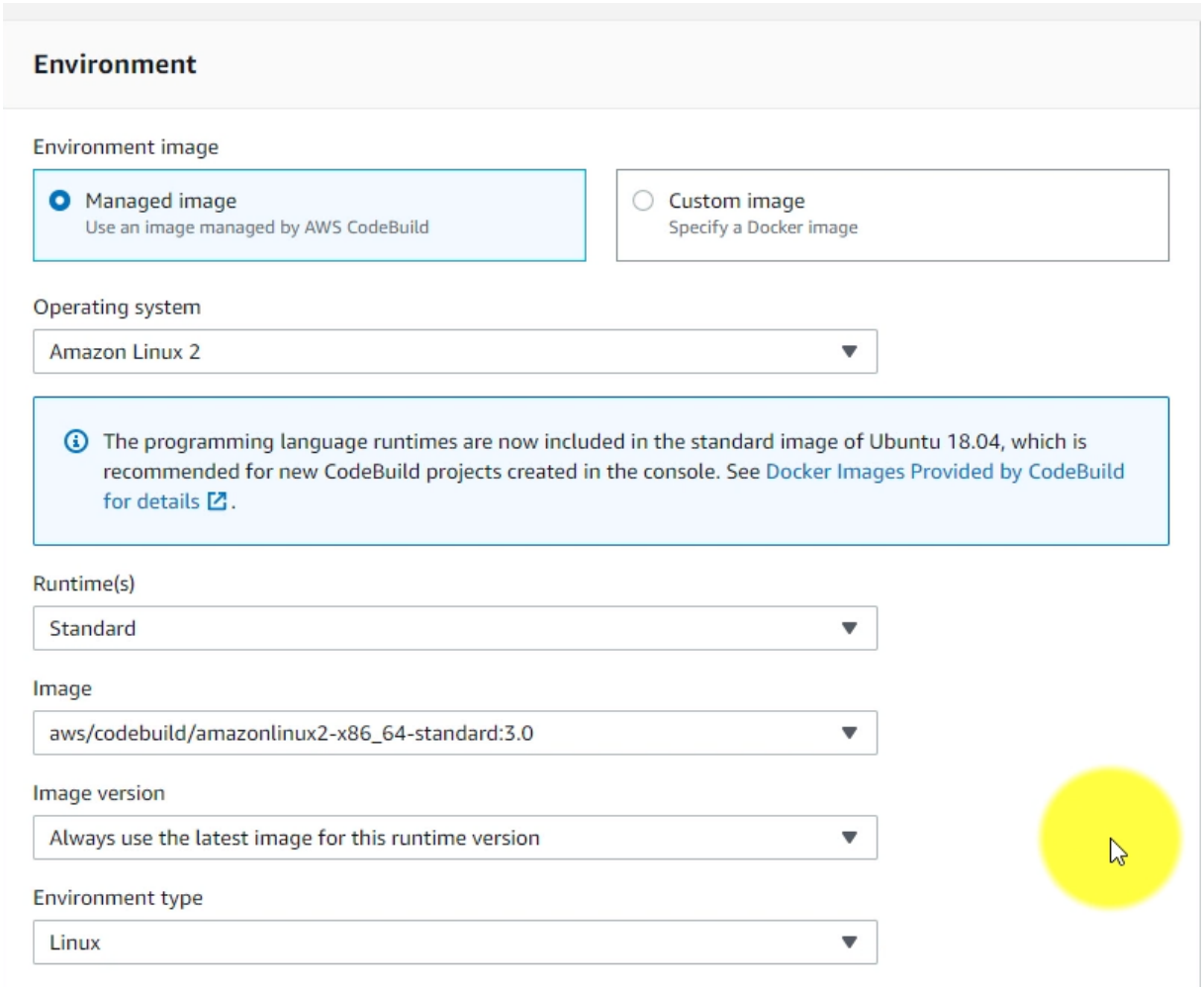
As a source select “CodeCommit” as our code is still in the CodeCommit Repository.

Then select he repository name we created earlier in the course including the master-branch:



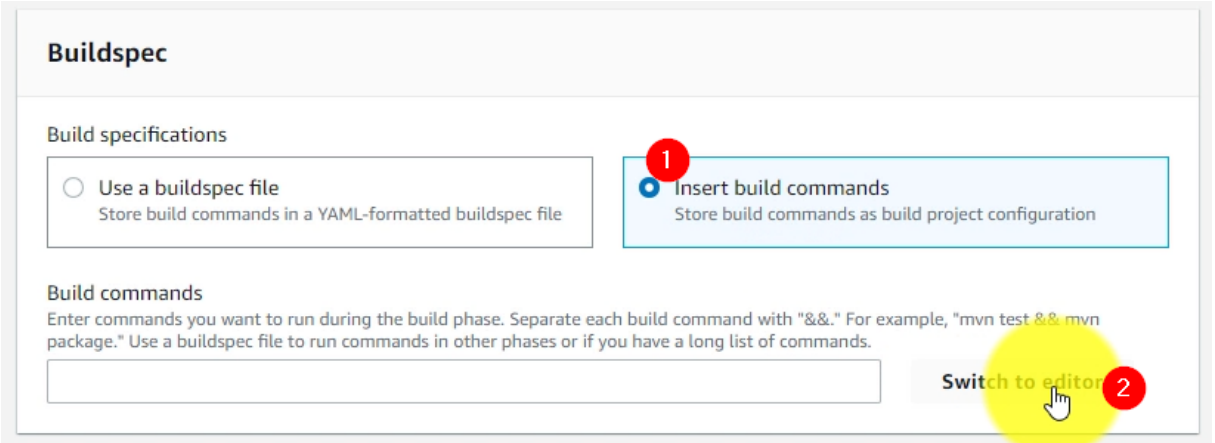
As Environment choose a managed Image from AWS:

Amazon Linux 2, Standard Runtime, version 3.0:

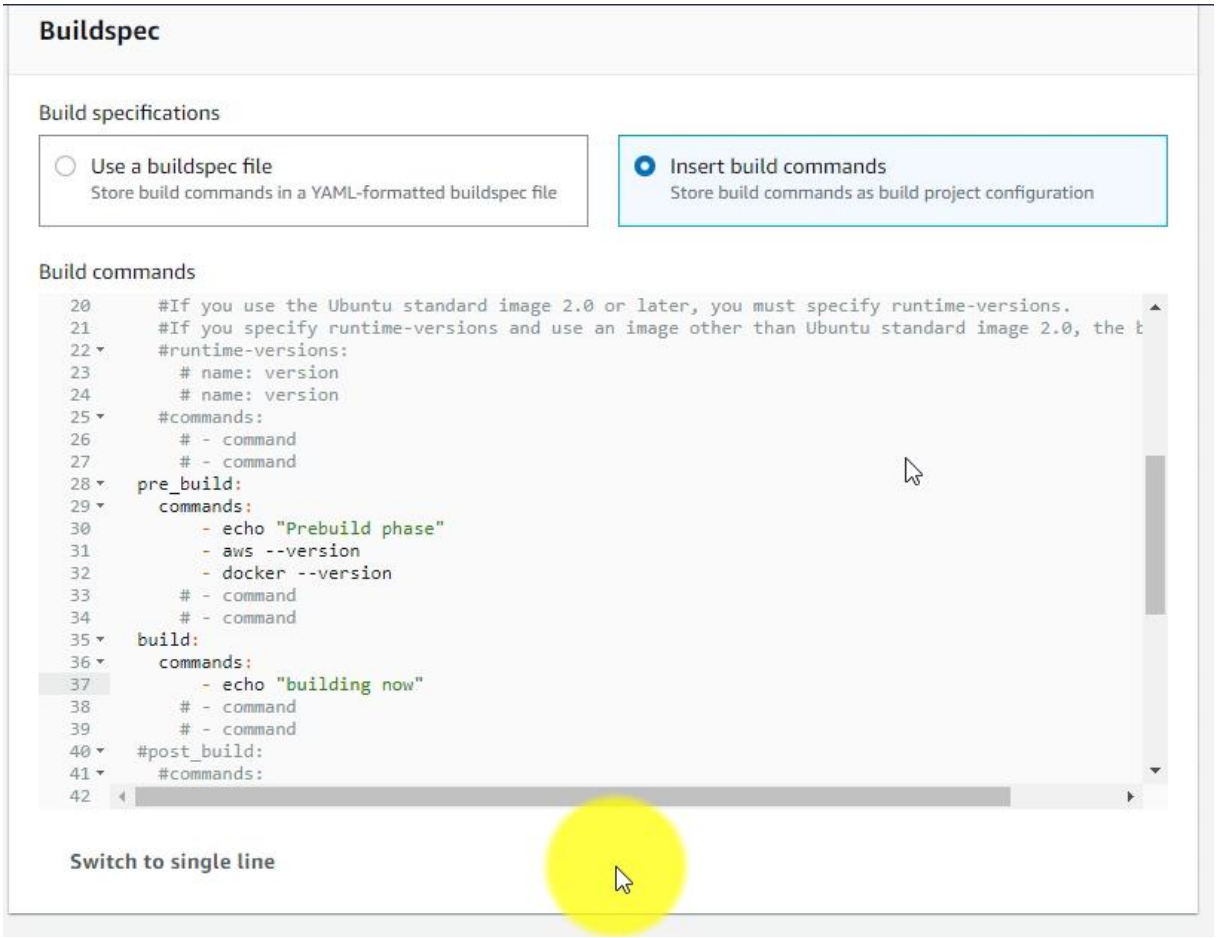


Because we want to build Docker-Containers on CodeBuild, we have to give privileged permissions:

Scroll down to the Buildspec file. Instead of uploading our own buildspec file we will create one directly in CodeBuild:



Switch to the Editor and write down some simple commands:



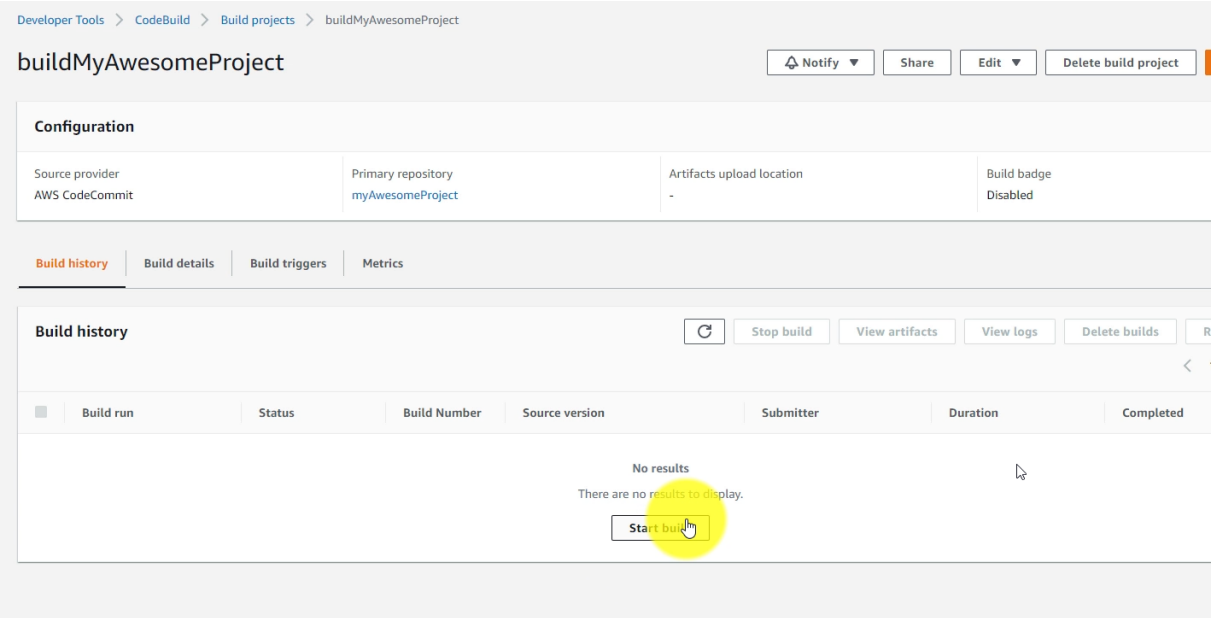
Be careful with indentation – yaml works off spaces.

```
pre_build:
  commands:
    - echo "Prebuild phase"
    - aws --version
    - docker --version
build:
  commands:
    - echo "building now"
```

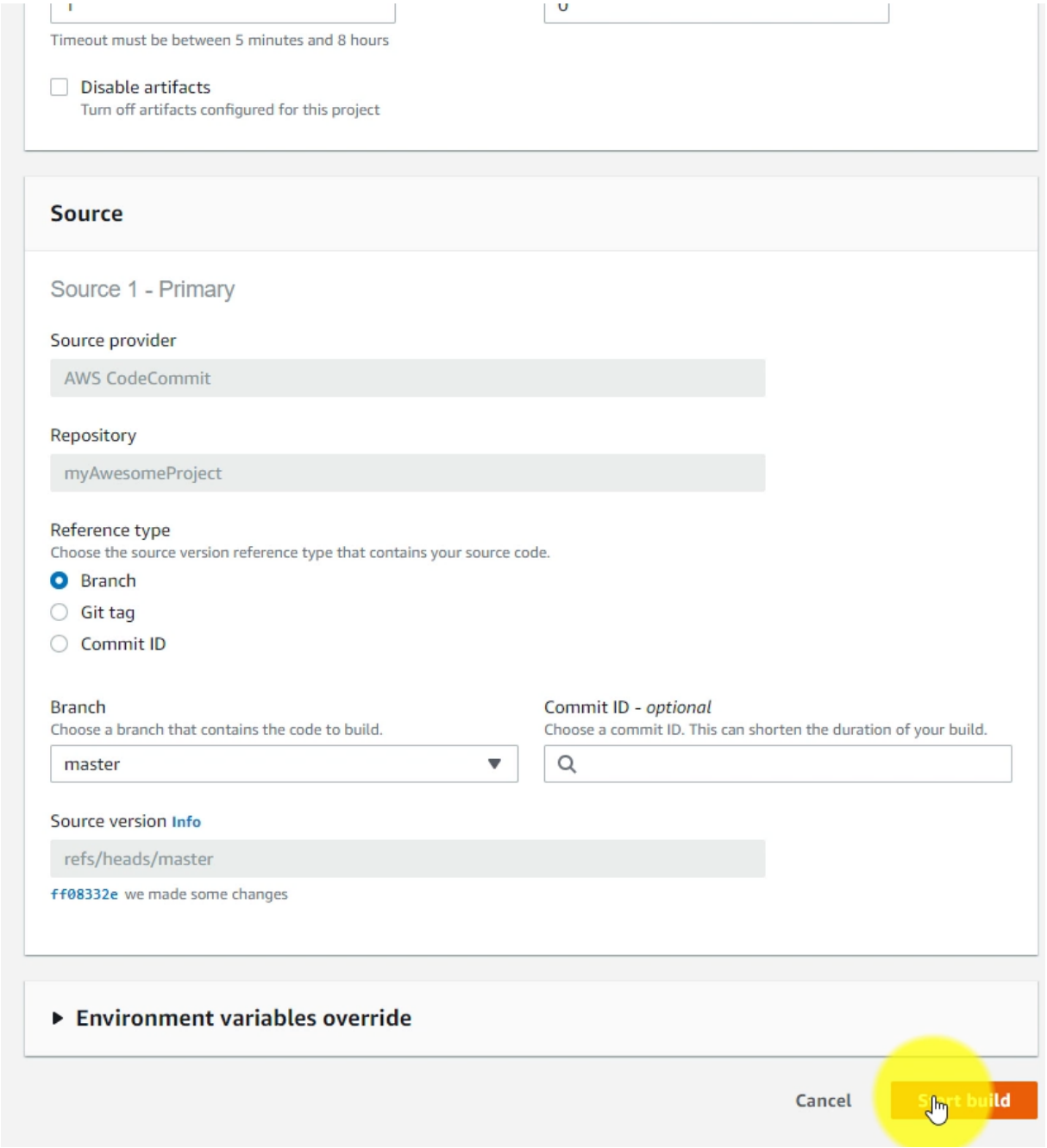
Then scroll down and create the build project.

Starting the Build

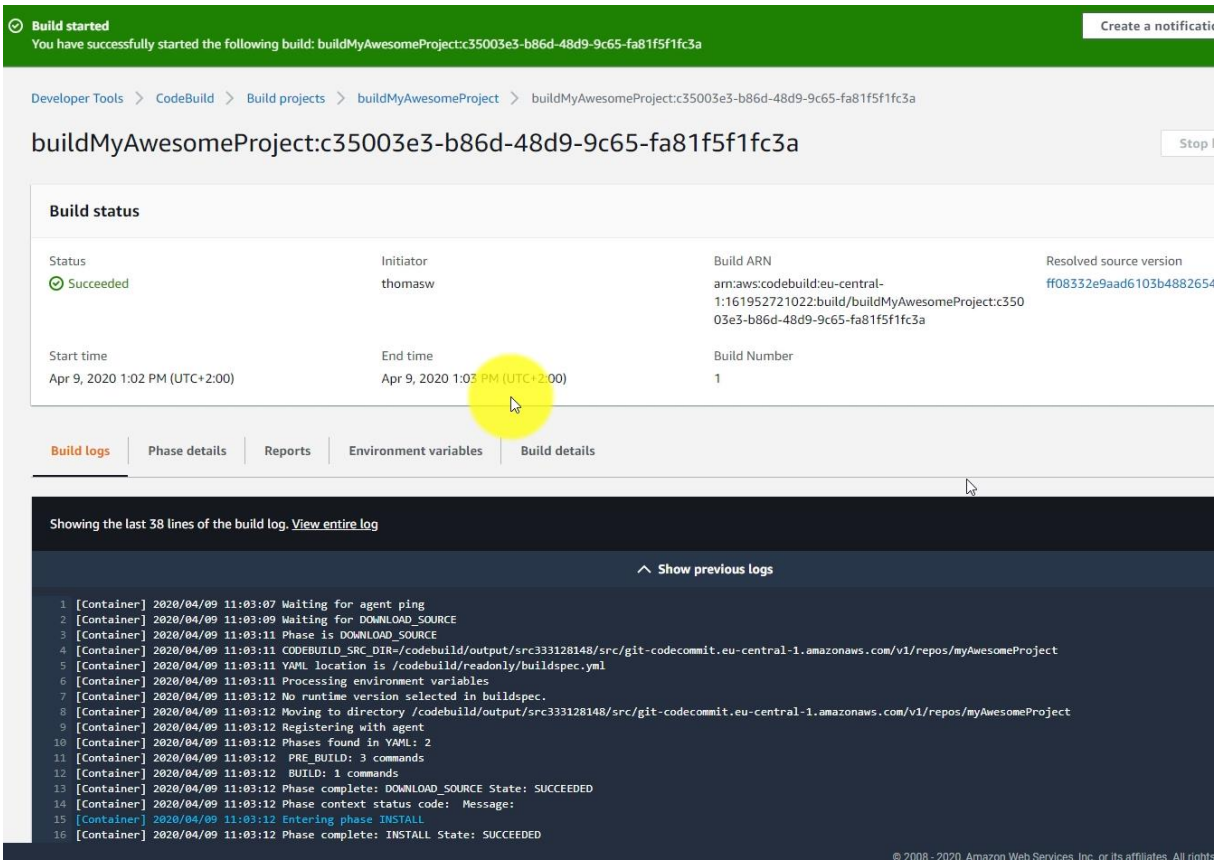
Now it's time to start the build:



Leave everything on default and start the build:



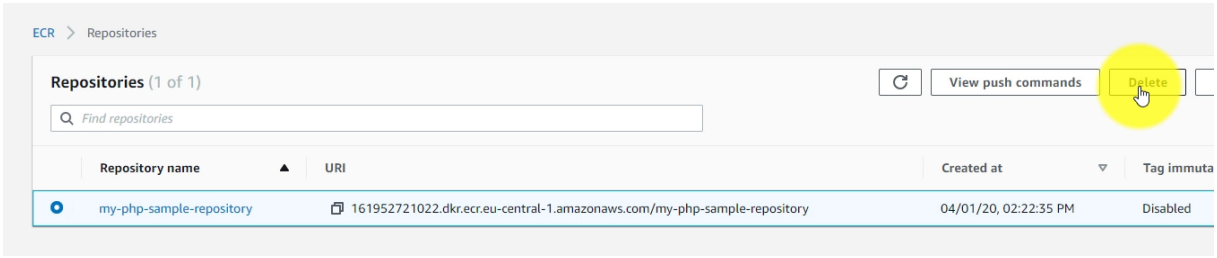
After a while it should simply print the container output in the log window:



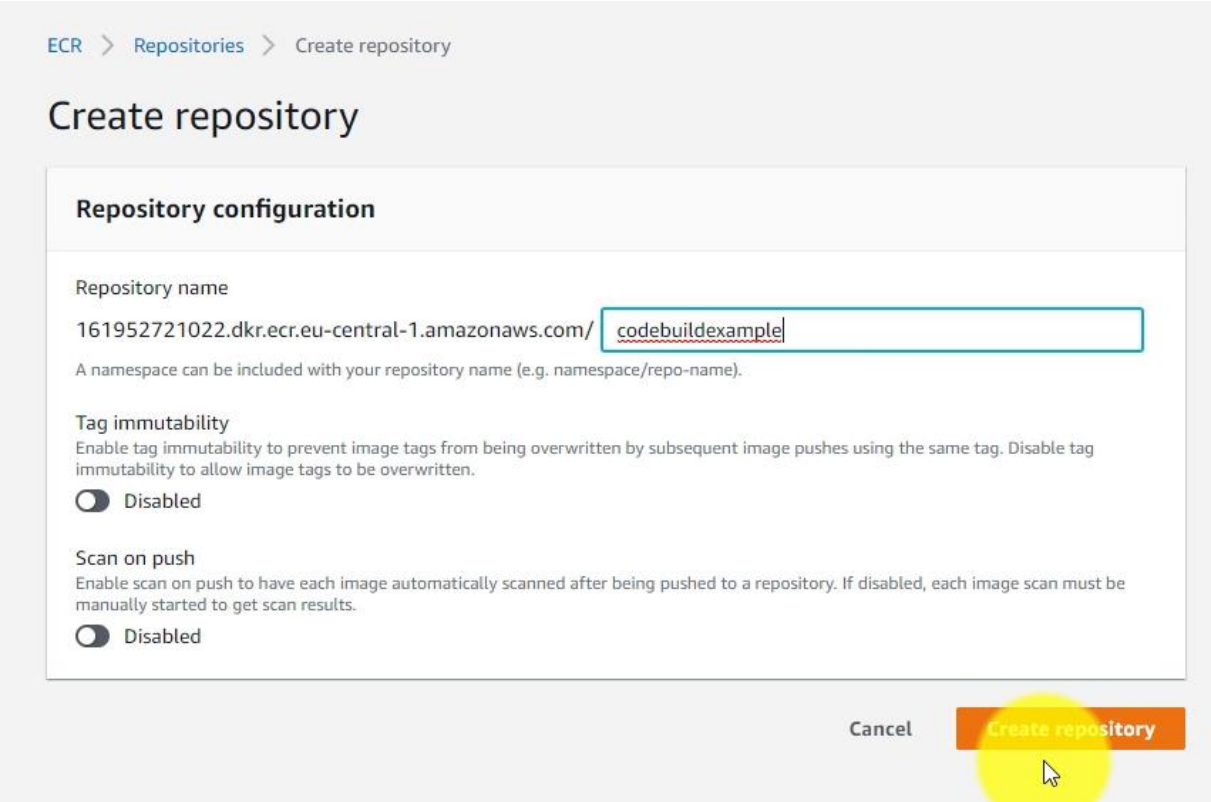
So, the build project is running, but it doesn't do anything yet. Let's use CodeBuild to upload a new docker image when we push something to the repository.

Create and Upload containers to ECR with CodeBuild

Delete your old ECR Repository if you still have it, because we will not need this anymore. We will create a new one.



Then create a new ECR Repository:



Integrate a buildspec.yml file

Now we need to integrate a buildspec.yml file into our codecommit repository, so that codebuild knows what to do. I took this example file from the following amazon blog post:

https://aws.amazon.com/blogs/devops/build-a-continuous-delivery-pipeline-for-your-container-images-with-amazon-ecr-as-source/

add the following buildspec.yml file to your codecommit repository containing the index.php and the Dockerfile:

```
version: 0.2

phases:
  pre_build:
    commands:
      - echo Logging in to Amazon ECR...
      - aws --version
      - $(aws ecr get-login --region $AWS_DEFAULT_REGION --no-include-email)
      - REPOSITORY_URI=012345678910.dkr.ecr.us-east-1.amazonaws.com/base-image
      - COMMIT_HASH=$(echo $CODEBUILD_RESOLVED_SOURCE_VERSION | cut -c 1-7)
      - IMAGE_TAG=${COMMIT_HASH:=latest}
  build:
    commands:
      - echo Build started on `date`
      - echo Building the Docker image...
      - docker build -t $REPOSITORY_URI:latest .
      - docker tag $REPOSITORY_URI:latest $REPOSITORY_URI:$IMAGE_TAG
  post_build:
    commands:
      - echo Build completed on `date`
      - echo Pushing the Docker images...
      - docker push $REPOSITORY_URI:latest
      - docker push $REPOSITORY_URI:$IMAGE_TAG
```

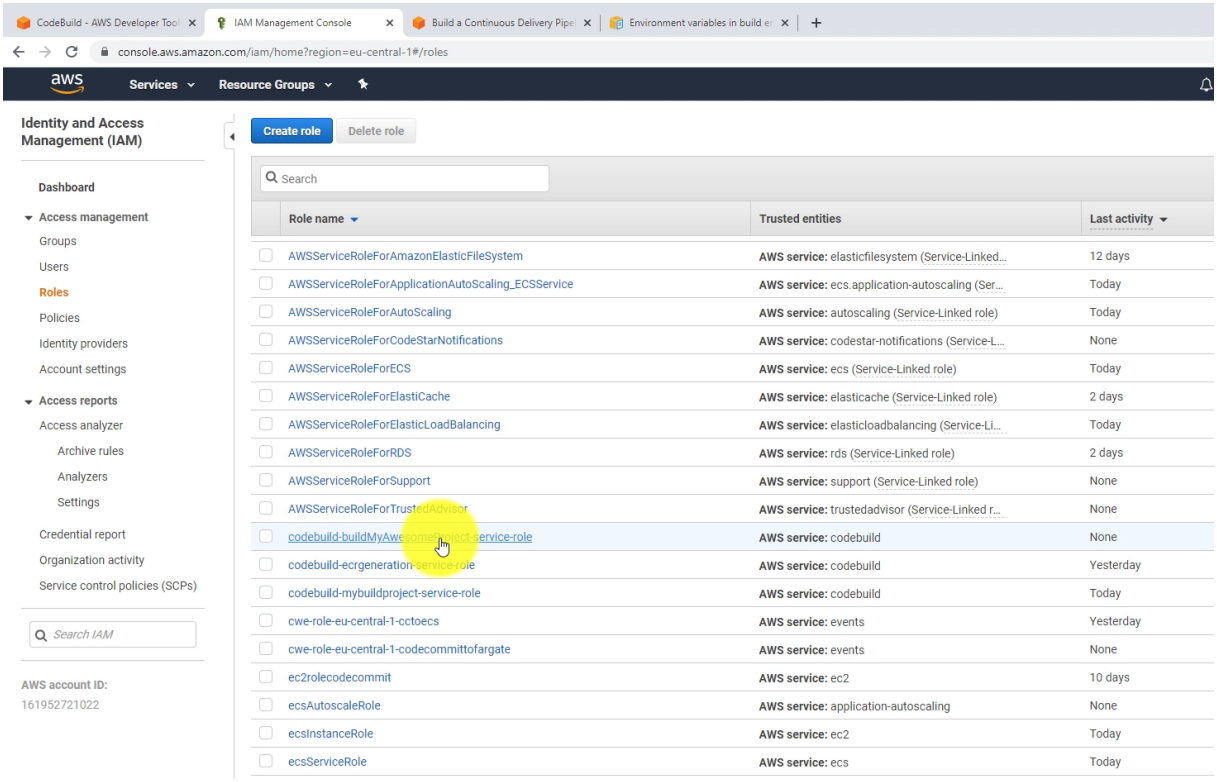
Edit the REPOSITORY_URI to the ECR URI and then stage, commit and push it:

```
git add .
git commit -a -m "added buildspec.yml file"
git push origin master
```

After that edit the IAM Role for the Build project, so that it can actually push to ECR.

Edit IAM Roles

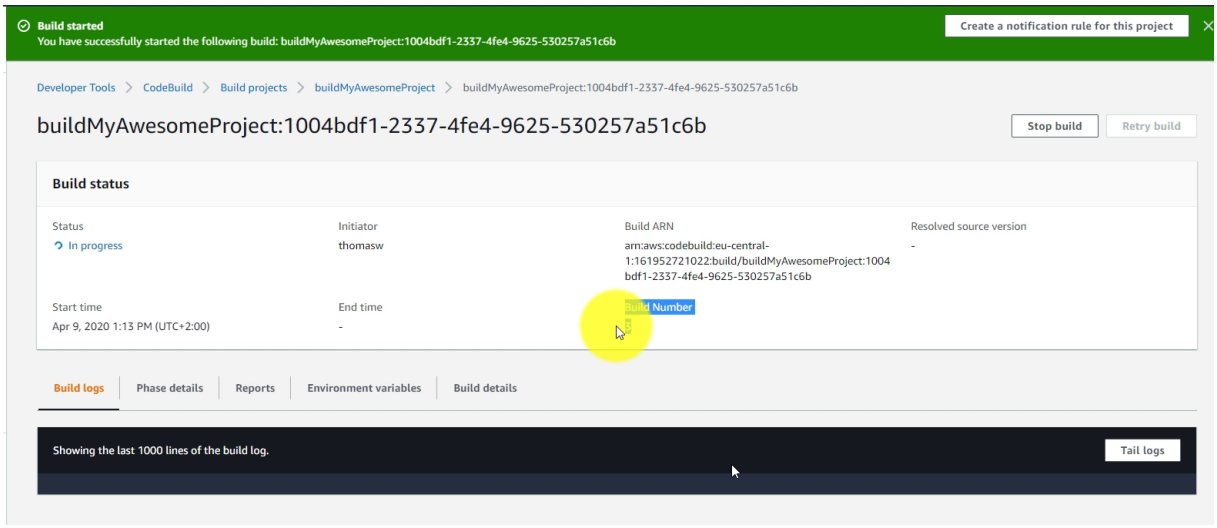
Find the Role for the codebuild project in the IAM console:



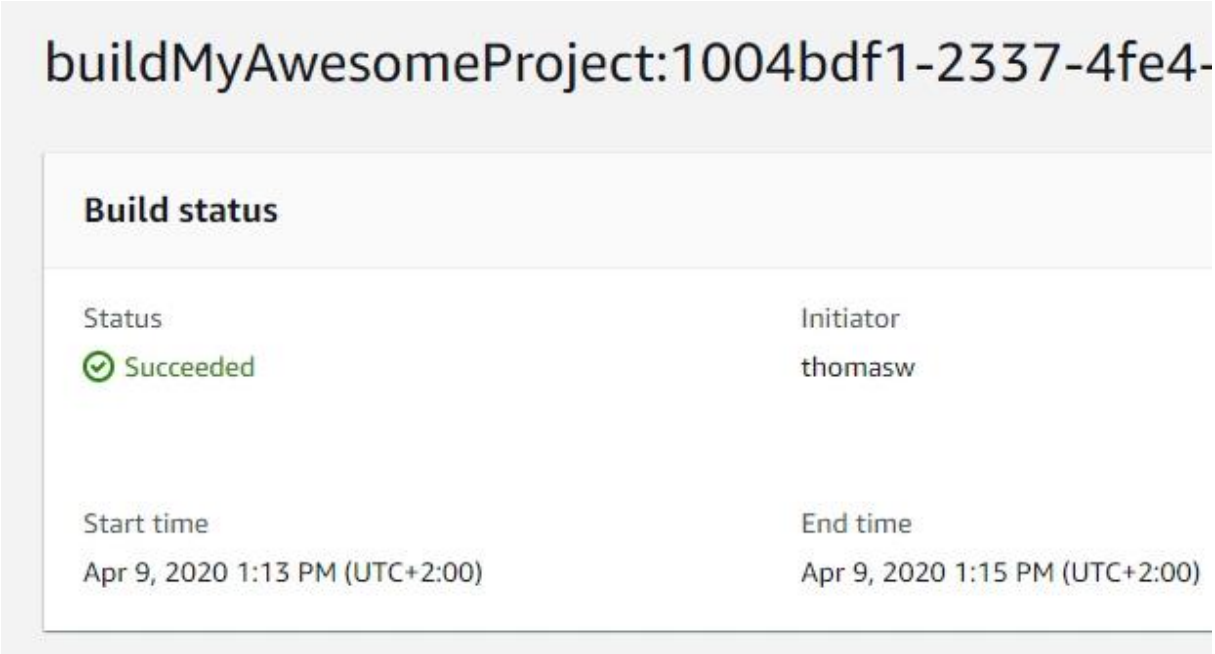
And attach a new policy, called “AmazonEC2ContainerRegistryPowerUser”:

Start the Build

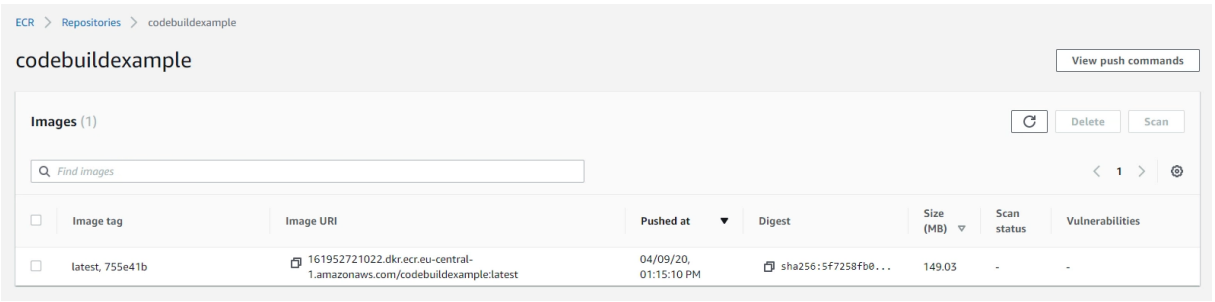
Then simply start the build in CodeBuild:



Wait until it says “Build Status: succeeded”:



And head over to ECR to check if the image was uploaded correctly:



Lab End