AWS Continuous Integration

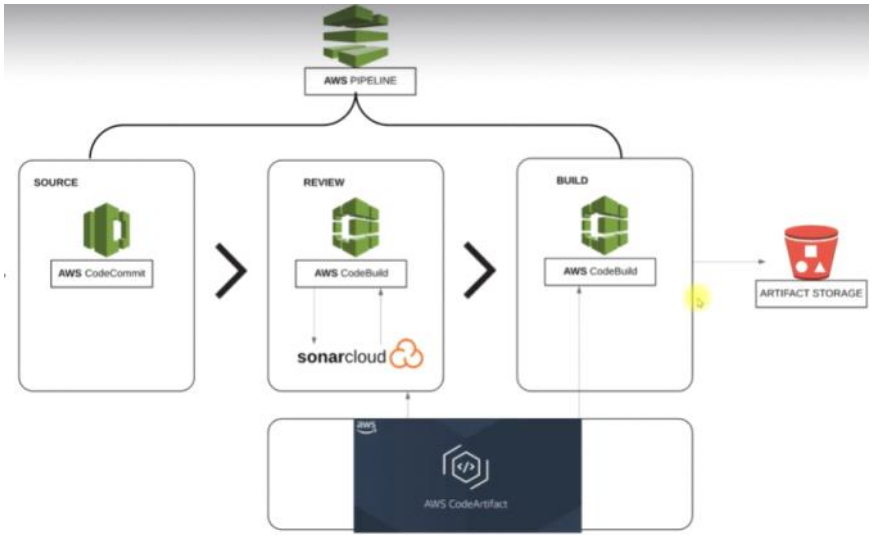
09 June 2024 10:12

Continuous Integration servers have extra overhead to manage the servers. With cloud services, we can set up these services that will have short MTTR, fault isolation, agile, no human intervention, and no ops

AWS Services used

- CodeCommit
- CodeArtifact
- CodeBuild
- CodePipeline
- CodeDeploy
- SonarCloud
- Checkstyle
- CodePipeline
- SNS
- S3

Architecture



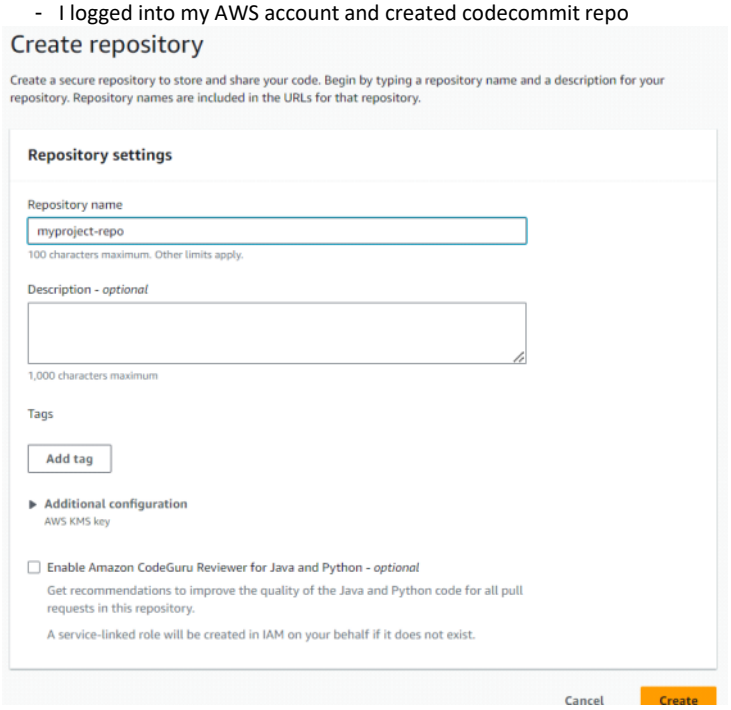
Flow of execution

- Login to AWS account
- Code commit
    - Create codecommit repo
    - Create IAM user with codecommit policies for Code commit
    - Generate ssh keys locally
    - Exchange keys with IAM user
    - Push source code from github repo to codecommit repository and push
- Code artifact
    - Create IAM user with code artifact access
    - Install AWS CLI configure
    - Export auth token
    - Update settings.xml file in source code top level directory with below details
    - Update pom.xml file with repo details
- Sonar cloud
    - Create sonar cloud account
    - Generate token
    - Create SSM parameters with sonar details
    - Create Build project
    - update codebuild role to access SSM parameterstore
- Crete notifications for sns
- Build project
    - Update pom.xml with artifact version with timestamp
    - Create variables in SSM ==> parameterstore
    - Create build project
    - Update codebuild role to access SSM parameterstore
- Create pipleline
    - Codecommit
    - Testcode
    - Build

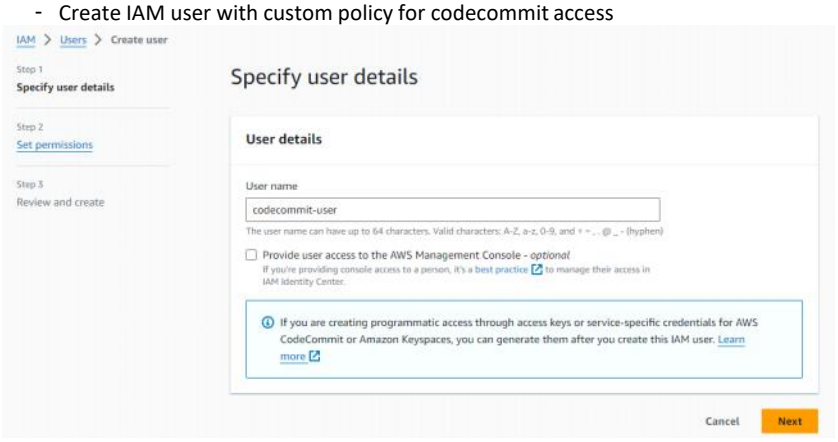
    - Deploy to s3 bucket
- Test pipeline

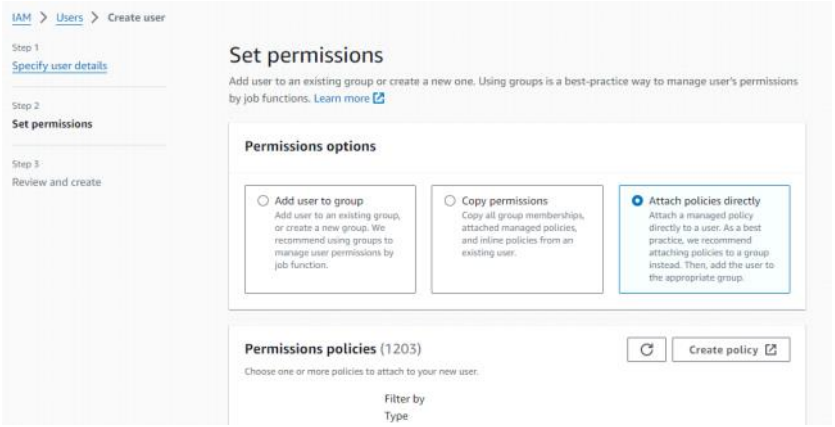
Note: in this project, it is assumed that one has to the following
- AWS account with IAM user with administrative access.
- Sonarcloud
- Github account with git installed on the user's CLI

Step 1: create CodeCommit repo and switch from github repo to codecommit
- I logged into my AWS account and created codecommit repo

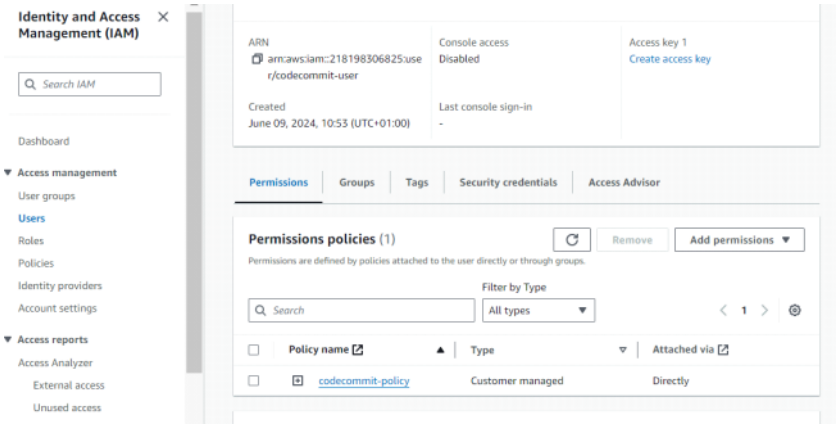


- Create IAM user with custom policy for codecommit access

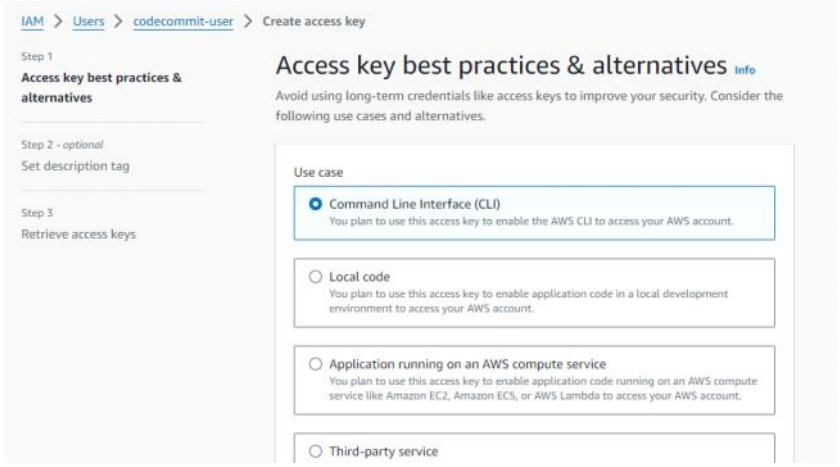




- Attach policies by giving specific commit policy to the user. When prompted, specify the ARN of the account by specifying the name of the codecommit repo (myproject-repo, in this case). Select the policy and attach it the codecommit-user



- After the creation of the user, Create SSH keys by going to the Security Credential tab of the user, scrolling down and clicking the *create access key* button.
- Give CLI access. Agree to understanding the risk at the bottom of the page and click Next



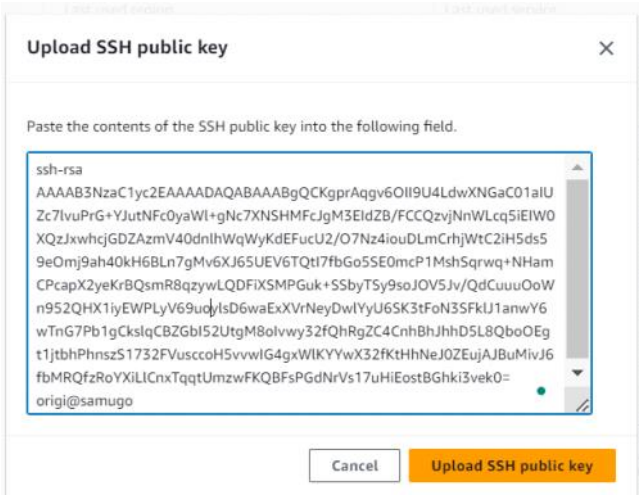
- On the next screen, I just clicked on create access key and later donwloaded the access key
- I then configured my terminal with the access key and secret keys downloaded from the previous step

```
origi@samugo MINGW64 /
$ aws configure
AWS Access Key ID [****************EQOD]: AKIATFTNB6AE6NVX63XO
AWS Secret Access Key [****************lMPH]: qgjbpXoaEfNKuJtQf5IuA4vGa4T62A98ni
u384AH
Default region name [eu-west-2]: us-east-1
Default output format [json]: json
```

- Next I generated the ssh keys using the ssh-keygen command

```
origi@samugo MINGW64 /
$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/c/Users/origi/.ssh/id_rsa): /c/Users/origi
/.ssh/myproject-repo_rsa  ub
The key fingerprint is: saved in /c/Users/origi/.ssh/myproject-repo_rsa
SHA256:AzA3S/Mm8ahijyYiqLRSt4PrMDBVBGDf1WJBXUClocc origi@samugo
The key's randomart image is:saved in /c/Users/origi/.ssh/myproject-repo_rsa
+---[RSA 3072]----+
|.o.o= B+=++o     |
|. ...=.@ooo      |
|  .. .*.=E       |
| .    . +.       |
|o o .   S        |
|oo.+.    .       |
|Booo..           |
|*=o o            |
|ooo. .           |
+----[SHA256]-----+
```

- I then cat the public key, copied and saved it in the code commit repo



- Next I created an ssh config file to be used when accessing different git repos. For this repo, the content of this config references the repo as thus, where the User is the access key created in IAM and the identity file, the path to the private ssh key:

```
Host git-codecommit.*.amazonaws.com
        User APKATFTNB6AE2C6PSN7R
        IdentityFile ~/.ssh/myproject-repo_rsa
```

This verifies connectivity:

```
$ ssh git-codecommit.us-east-1.amazonaws.com
The authenticity of host 'git-codecommit.us-east-1.amazonaws.com (52.94.233.146)
' can't be established.
RSA key fingerprint is SHA256:eLMY1j0DKA4uvDZcl/KgtIayZANwX6t8+8isPtotBoY.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'git-codecommit.us-east-1.amazonaws.com' (RSA) to the
 list of known hosts.
You have successfully authenticated over SSH. You can use Git to interact with A
WS CodeCommit. Interactive shells are not supported.Connection to git-codecommit
.us-east-1.amazonaws.com closed by remote host.
Connection to git-codecommit.us-east-1.amazonaws.com closed.
```

The next major step is to migrate my github repo to codecommit

- Created a folder and cloned the github repo (https://github.com/hkhcoder/vprofile-project.git) in the folder
- Checked out all the branches using bash script

```
origi@samugo MINGW64 ~/OneDrive - University of the People/it_trainings/aws_solu
tions_architect/cloud-project-5/vprofile-project (ci-aws)
$ git branch -a
* ci-aws
  main
  remotes/origin/HEAD -> origin/main
  remotes/origin/aws-LiftAndShift
  remotes/origin/aws-refactor
  remotes/origin/cd-aws
  remotes/origin/ci-aws
  remotes/origin/ci-jenkins
  remotes/origin/containers
  remotes/origin/docker
  remotes/origin/main
  remotes/origin/terraform-eks

origi@samugo MINGW64 ~/OneDrive - University of the People/it_trainings/aws_solu
tions_architect/cloud-project-5/vprofile-project (ci-aws)
$ git branch -a | grep remotes | grep -v HEAD | cut -d / -f3 > /tmp/br

origi@samugo MINGW64 ~/OneDrive - University of the People/it_trainings/aws_solu
tions_architect/cloud-project-5/vprofile-project (ci-aws)
$ for i in `cat /tmp/br`; do git checkout $i; done
Switched to a new branch 'aws-LiftAndShift'
branch 'aws-LiftAndShift' set up to track 'origin/aws-LiftAndShift'.
Switched to a new branch 'aws-refactor'
branch 'aws-refactor' set up to track 'origin/aws-refactor'.
Switched to a new branch 'cd-aws'
branch 'cd-aws' set up to track 'origin/cd-aws'.
Switched to branch 'ci-aws'
Your branch is up to date with 'origin/ci-aws'.
Switched to a new branch 'ci-jenkins'
branch 'ci-jenkins' set up to track 'origin/ci-jenkins'.
Switched to a new branch 'containers'
branch 'containers' set up to track 'origin/containers'.
Switched to a new branch 'docker'
branch 'docker' set up to track 'origin/docker'.
Switched to branch 'main'
Your branch is up to date with 'origin/main'.
Switched to a new branch 'terraform-eks'
branch 'terraform-eks' set up to track 'origin/terraform-eks'.
```

```
origi@samugo MINGW64 ~/OneDrive - University of the People/it_trainings/aws_solu
tions_architect/cloud-project-5/vprofile-project (terraform-eks)
$ git branch -a
  aws-LiftAndShift
  aws-refactor
  cd-aws
  ci-aws
  ci-jenkins
  containers
  docker
  main
* terraform-eks
  remotes/origin/HEAD -> origin/main
  remotes/origin/aws-LiftAndShift
  remotes/origin/aws-refactor
  remotes/origin/cd-aws
  remotes/origin/ci-aws
  remotes/origin/ci-jenkins
  remotes/origin/containers
  remotes/origin/docker
  remotes/origin/main
  remotes/origin/terraform-eks
```

- Switch remote repository -->
    - git remote set-url origin ssh://commit-url OR
    - git rm origin and git add origin ssh://commit-url

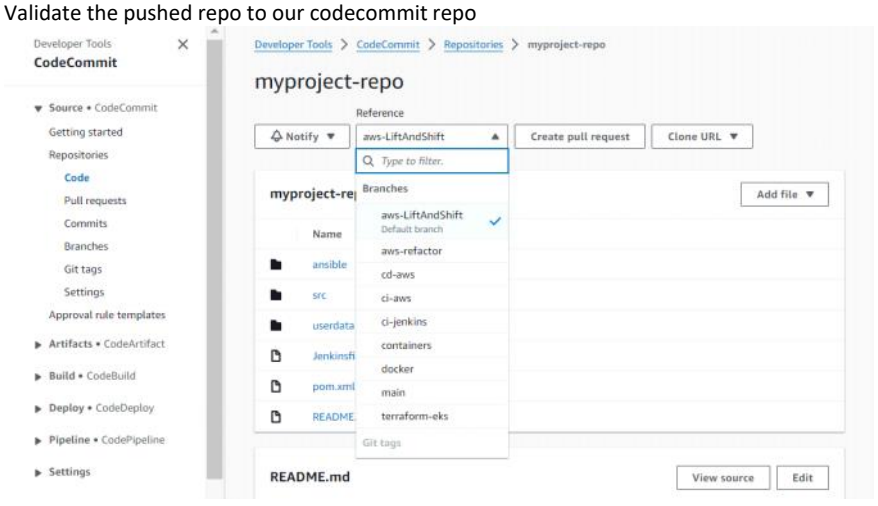
```
origi@samugo MINGW64 ~/OneDrive - University of the People/it_trainings/aws_solutions_
architect/cloud-project-5/vprofile-project (terraform-eks)
$ git remote set-url origin ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/mypr
oject-repo

origi@samugo MINGW64 ~/OneDrive - University of the People/it_trainings/aws_solutions_
architect/cloud-project-5/vprofile-project (terraform-eks)
$ git remote -v
origin  ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/myproject-repo (fetch)
origin  ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/myproject-repo (push)
```

- Push checked out branches to new repo

```
origi@samugo MINGW64 ~/OneDrive - University of the People/it_trainings/aws_solutions_
architect/cloud-project-5/vprofile-project (terraform-eks)
$ git push origin --all
Enumerating objects: 487, done.
Counting objects: 100% (487/487), done.
Delta compression using up to 8 threads
Compressing objects: 100% (277/277), done.
Writing objects: 100% (487/487), 7.69 MiB | 2.16 MiB/s, done.
Total 487 (delta 186), reused 487 (delta 186), pack-reused 0
remote: Validating objects: 100%
To ssh://git-codecommit.us-east-1.amazonaws.com/v1/repos/myproject-repo
 * [new branch]      aws-LiftAndShift -> aws-LiftAndShift
 * [new branch]      aws-refactor -> aws-refactor
 * [new branch]      cd-aws -> cd-aws
 * [new branch]      ci-aws -> ci-aws
 * [new branch]      ci-jenkins -> ci-jenkins
 * [new branch]      containers -> containers
 * [new branch]      docker -> docker
 * [new branch]      main -> main
 * [new branch]      terraform-eks -> terraform-eks
```

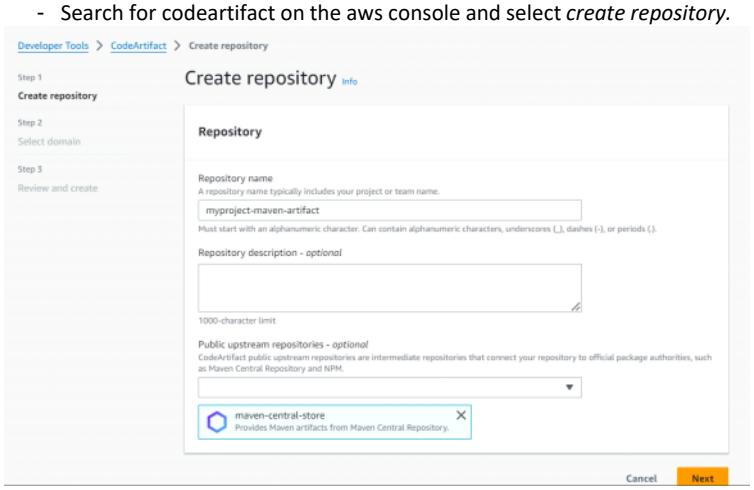
Validate the pushed repo to our codecommit repo



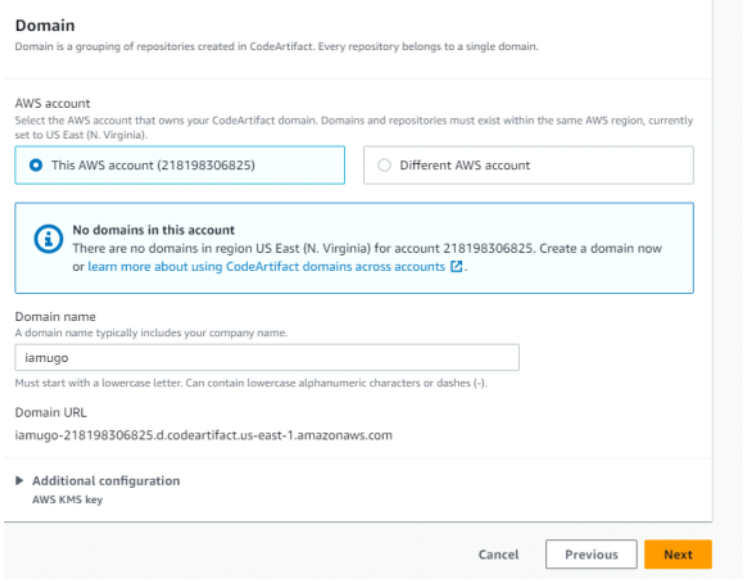
Step 2: create code artifact repository

Code artifact repo is used to store dependencies for the build tools, like maven. The reason for doing this that many projects will have requirements to use dependencies they control rather than downloading them from the internet. Even if one is not using aws for the build process but using jenkins, you can still point the build tool to code artifact repo

- Search for codeartifact on the aws console and select *create repository.*



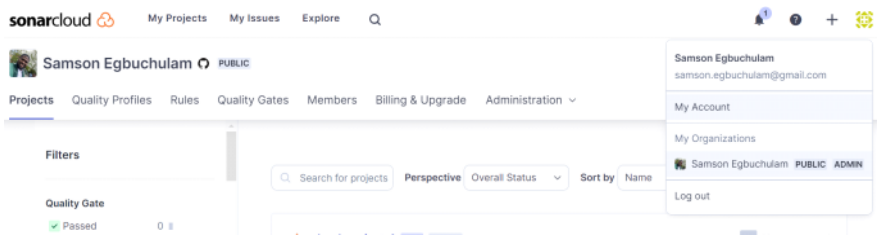
- I chose maven central store (since the build process will be done with maven)
- Next, I selected that the repo is for my own account. I also gave a domain and clicked on Next --> Create.



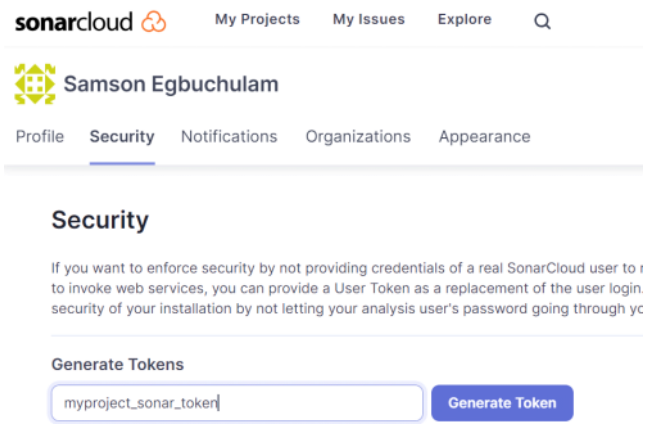
The variable, CODEARTIFACT_AUTH_TOKEN will be exported

Step 3: create sonarcloud account with github and configure with

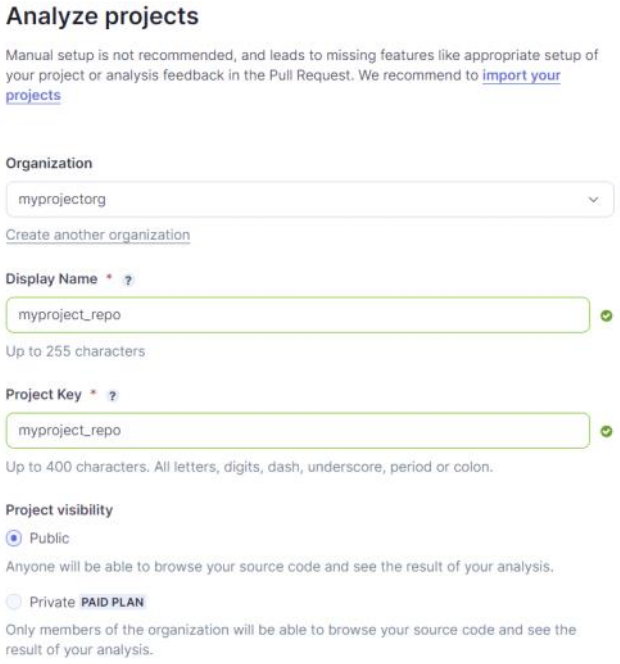
-



- I went to the *Security* tab and generated a token



- Copy the token to a safe location
- Click on the *Organizations* tab to access your organization. If you don't have an organization, create one manually by clicking on the '+' sign at the top right corner of the page.
- From the same '+' sign, click *Analyze New Project*

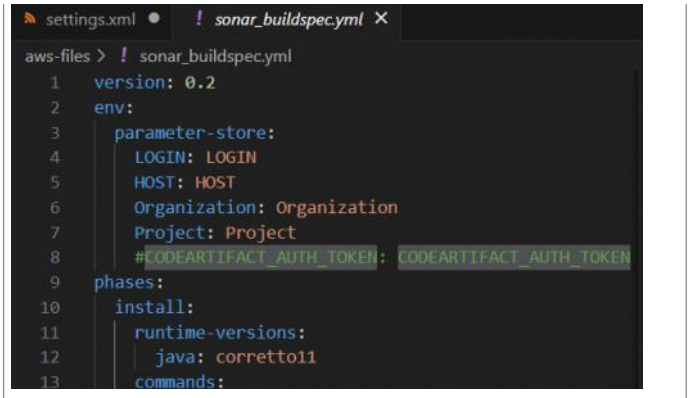


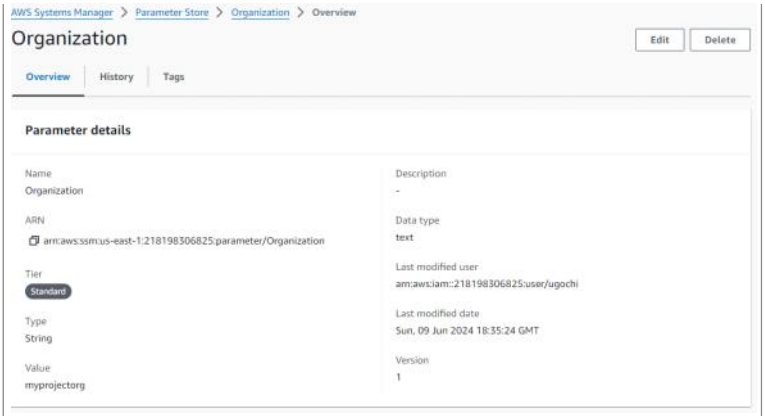
- On the next screen, I chose 'Previous Version' to enable the analysis happen on any new version of the code. Click create
- Next I copied the Project key and Organization key to my notepad

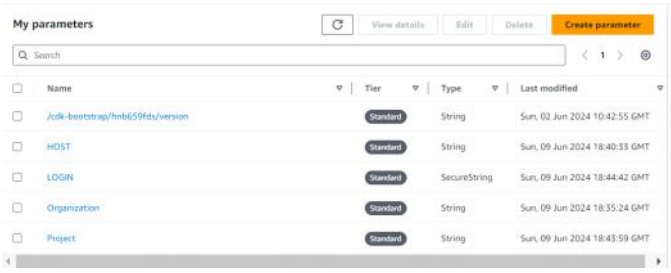
AWS Parameter store
AWS Parameter stores are used to store the values of sensitive variables away from the public. This is because I am going to be using these variables in the project and storing them in the code pose a serious threat; hence the reason why I am using parameter store. Secrets manager could be used but it costs a lot.
The parameters are mentioned in the build_spec file
The Parameter Store is a service under the AWS Secrets Manager .

- Search for parameter store on the AWS console. Create new parameter
- Create entries in the parameter store, each for LOGIN, HOST, Organization and Project. The type is string for all except LOGIN which has a type of Secure String

Sonar build-spec file	Sample of a parameter in parameter store







Step 4: Edit pom.xml, buildspec.yml and settings.xml files with the right information from coder artifact and start CodeBuild project.

CodeBuild is the start of the pipeline to used.

- The first step is to edit the pom.xml file with the right information
    - I went to Code artifact and copied the url of the artifact repo. This is used to edit the url section of the build repository in the pom.xml file

```
                        <goals>
                            <goal>report</goal>
                        </goals>
                    </execution>
                </executions>
            </plugin>

            </plugins>
        </build>
        <repositories>
          <repository>
            <id>codeartifact</id>
            <name>codeartifact</name>
        <url>https://iamugo-218198306825.d.codeartifact.us-east-1.amazonaws.com/maven/maven-central-store/</url>
          </repository>
        </repositories>
    </project>

```

    - I also edited the settings file with the url of the artifact

```
<profiles>
  <profile>
    <id>default</id>
    <repositories>
      <repository>
        <id>codeartifact</id>
    <url>https://iamugo-218198306825.d.codeartifact.us-east-1.amazonaws.com/maven/maven-central-store/</url>
      </repository>
    </repositories>
  </profile>
</profiles>
<activeProfiles>
        <activeProfile>default</activeProfile>
    </activeProfiles>
<mirrors>
  <mirror>
    <id>codeartifact</id>
    <name>domainname--maven-central-store</name>
    <url>https://iamugo-218198306825.d.codeartifact.us-east-1.amazonaws.com/maven/maven-central-store/</url>
    <mirrorOf>*</mirrorOf>
```

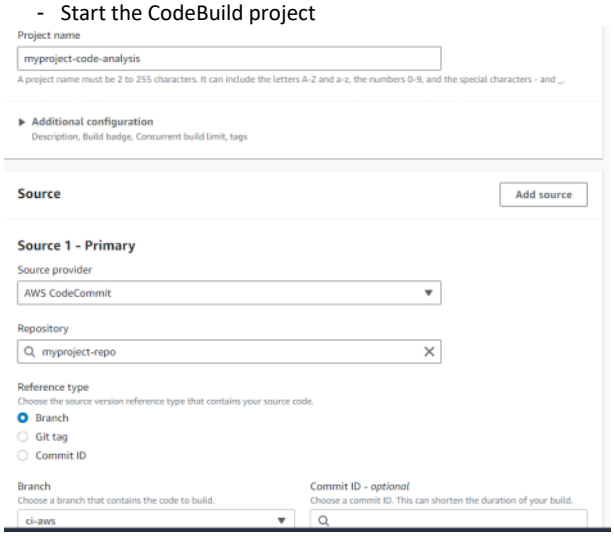
I had to ensure the sonar_buildspec.yml file be in the root directory of the code repo. I renamed it to buildspec.yml

- I then edited the settings.xml file to reflect the right auth_token gotten from code artifact

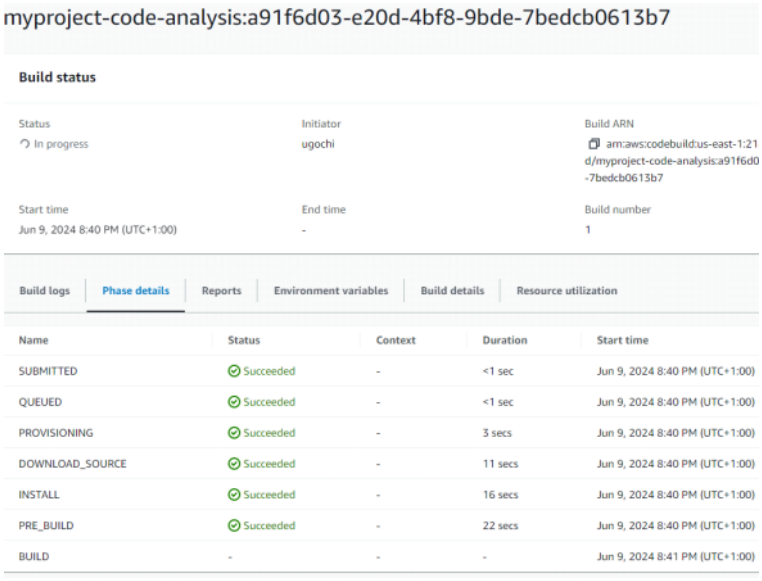
```
phases:
  install:
    runtime-versions:
      java: corretto11
    commands:
    - cp ./settings.xml /root/.m2/settings.xml
    - export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact get-authorization-token --domain iamugo --domain-owner 21
  pre_build:
```

- Saved and pushed the changes to the code commit repo

- Start the CodeBuild project



  - The role created during the creation of the build process will not have access to the parameter store. I had to edit the role after the creation of the build process to add a policy for parameter store access.
  - I used ubuntu as the operating system and chose the aws/codebuild/standard:5.0 as the image
  - I also enabled CloudWatch logs
- Next was to edit the roles to add read access for artifact and parameter store to the CodeBuild roles.
- Started the build process



- The build job takes a while to build but in the end, success

Build logs

Succeeded Start time: 13 minutes ago

```
22631 [WARNING]   * src/test/java/com/visualpathit/account/modeltest/RoleTest.java
22632 [WARNING]   * src/main/webapp/WEB-INF/views/registration.jsp
22633 [WARNING]   * src/main/java/com/visualpathit/account/controller/UserController.java
22634 [WARNING]   * src/main/webapp/WEB-INF/views/userUpdate.jsp
22635 [WARNING]   * src/test/java/com/visualpathit/account/setup/StandaloneMvcTestViewResolver.java
22636 [WARNING]   * src/main/webapp/WEB-INF/appconfig-security.xml
22637 [WARNING]   * src/main/java/com/visualpathit/account/service/UserDetailsServiceImpl.java
22638 [WARNING]   * src/main/java/com/visualpathit/account/service/SecurityServiceImpl.java
22639 [WARNING] This may lead to missing/broken features in SonarCloud
22640 [INFO] CPD Executor 7 files had no CPD blocks
22641 [INFO] CPD Executor Calculating CPD for 24 files
22642 [INFO] CPD Executor CPD calculation finished (done) | time=27ms
22643 [INFO] Analysis report generated in 2250ms, dir size=669 KB
22644 [INFO] Analysis report compressed in 126ms, zip size=209 KB
22645 [INFO] Analysis report uploaded in 906ms
22646 [INFO] ANALYSIS SUCCESSFUL, you can find the results at: ***/dashboard?id=***
22647 [INFO] Note that you will be able to access the updated dashboard once the server has processed the submitted analysis report
22648 [INFO] More about the report processing at ***/api/ce/task?id=AYtgoh1z11zb88W9NqgN
22649 [INFO] Sensor cache published successfully
22650 [INFO] Analysis total time: 49.385 s
22651 [INFO] ------------------------------------------------------------------------
22652 [INFO] BUILD SUCCESS
22653 [INFO] ------------------------------------------------------------------------
22654 [INFO] Total time:  01:00 min
22655 [INFO] Finished at: 2023-10-24T07:42:05Z
22656 [INFO] ------------------------------------------------------------------------
22657
```

- Next, I built the artifact and stored it in an s3 bucket. To do this:
    - I edited the build_spec file with the right CODEARTIFACT_AUTH_TOKEN (line11) from code artifact. Save and commit

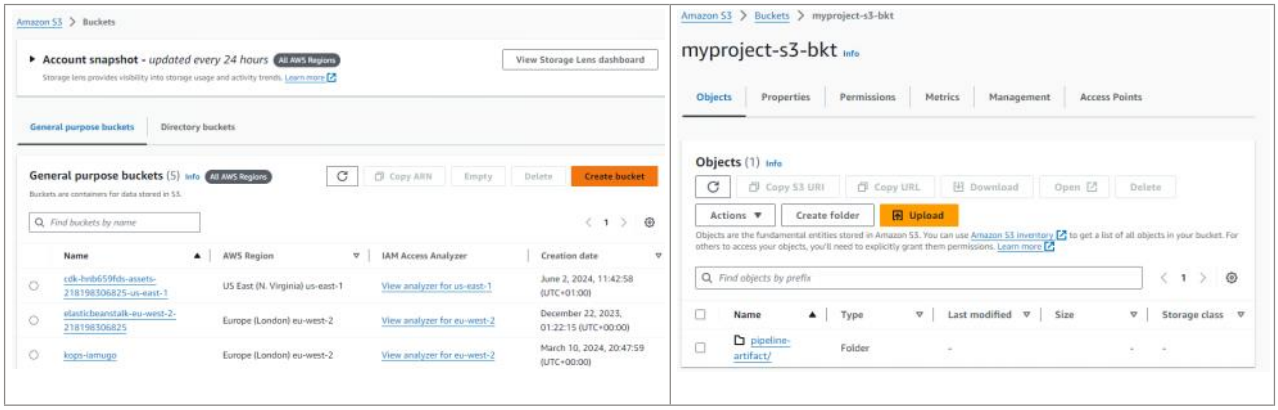
aws-files > ! build_buildspec.yml

```
1  version: 0.2
2  #env:
3    #parameter-store:
4      #CODEARTIFACT_AUTH_TOKEN: CODEARTIFACT_AUTH_TOKEN
5  phases:
6    install:
7      runtime-versions:
8        java: corretto11
9      commands:
10       - cp ./settings.xml /root/.m2/settings.xml
11       - export CODEARTIFACT_AUTH_TOKEN=`aws codeartifact g
12   pre_build:
13     commands:
14       - apt-get update
15       - apt-get install -y jq
16       - wget https://dlcdn.apache.org/maven/maven-3/3.9.4/
17       - tar xzvf apache-maven-3.9.4-bin.tar.gz
18       - ln -s apache-maven-3.9.4 maven
19   build:
20     commands:
21       - mvn clean install -DskipTests
22 artifacts:
23   files:
24     - target/**/*.war
25   discard-paths: yes
```

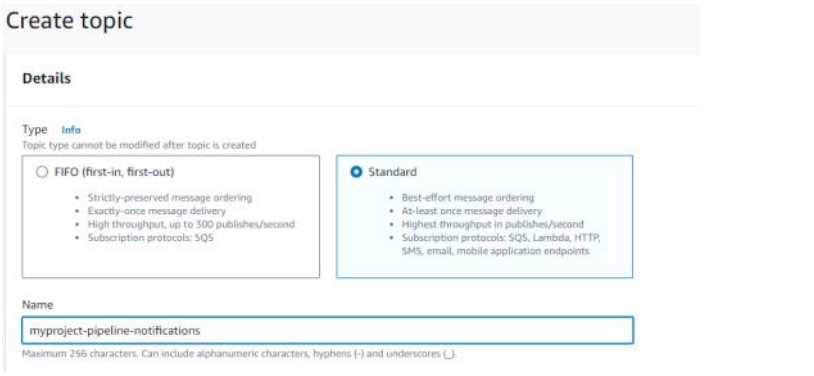
    - This build job produces the artifact which I will send to the s3 bucket
    - Created a build job for this build_buildspec.yml file. This is same as above but this time, I ensured that the file path (aws-files/build_buildspec.yml) was specified in the buildspec path
    - I had to add the code artifact read policy to the role
    - Start the build job

Step 5: Create S3 bucket to store the artifact and SNS notification for the build pipeline

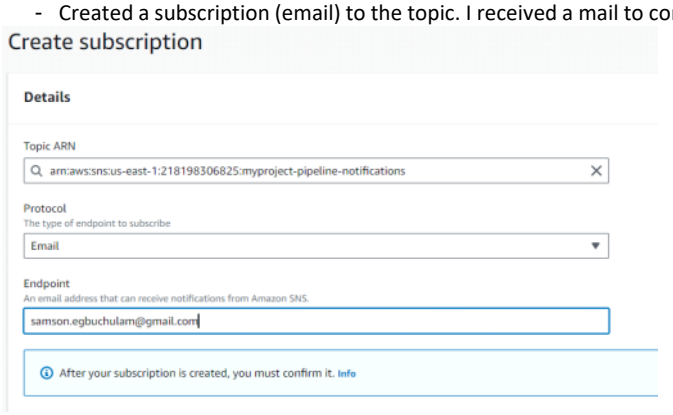
- Go to the s3 service and created a bucket
- I also created a folder to store the artifact



- Next I created an SNS topic



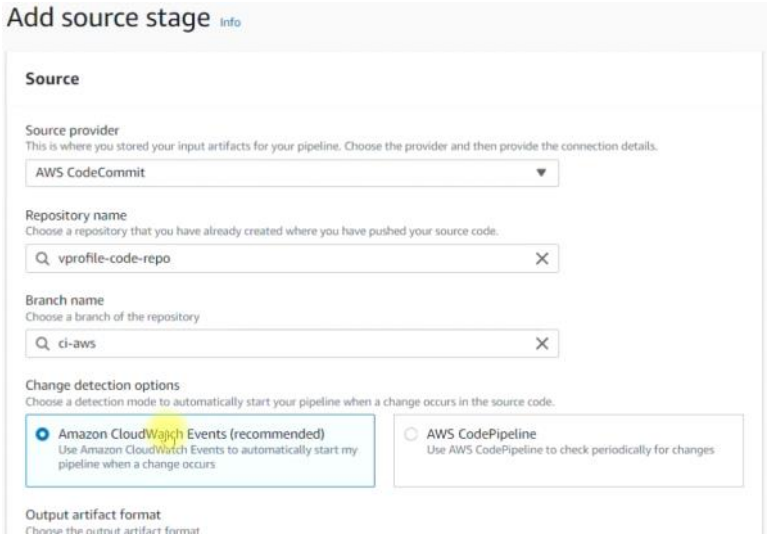
- Created a subscription (email) to the topic. I received a mail to confirm my subscription



- Confirmed the email using the link sent.

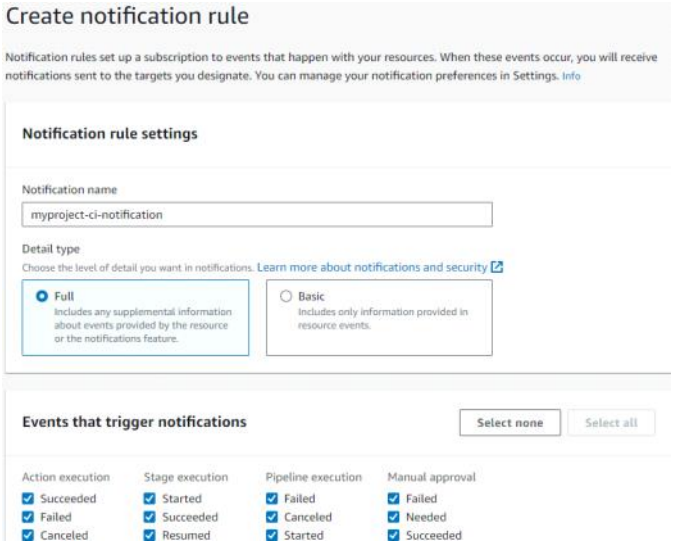
Step 6: create pipeline

- Go to code pipeline. Click create pipeline

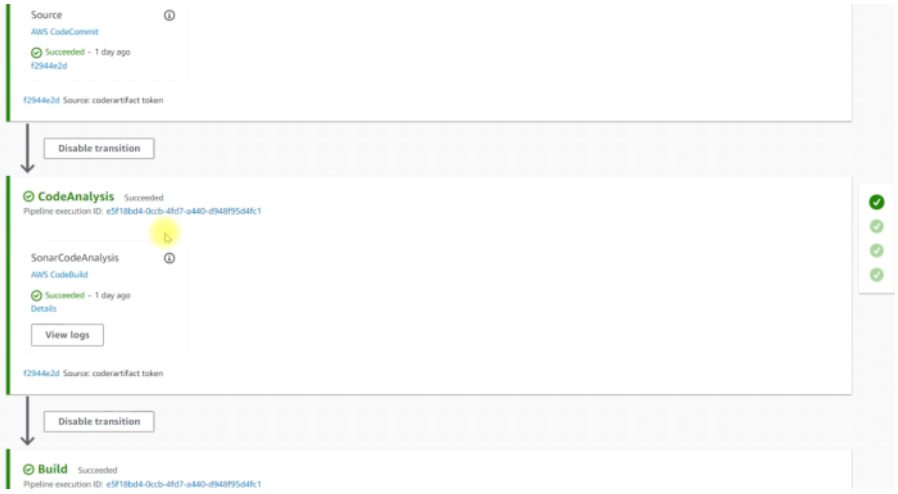


- I gave the settings seen above, clicked Next
- On the next page, I chose the AWS code commit repo as the build provider and the branch and opted to trigger the pipeline from CloudWatch Events instead. I skipped the deploy section. Create pipeline
- As soon as the pipeline is created is created, it begins to run. I stopped (Stop Execution button)

- I then edited the pipeline to include a stage for the code analysis (action group: the code build job; project name: the code analysis)
- Deployment to s3 is another stage I added at the end of the pipeline (input artifact is the code build job, the product key is the name of the folder - pipeline-artifact I created in s3. I ensured the check mark for *extract before deploy* is checked.
- Saved pipeline
- In the pipeline settings, I created a notification rule to publish to the sns topic I created earlier. For this project, I selected all the events but in real life, not all the notifications will be selected.



- After everything, I clicked on Release change. Another way to test the pipeline is to trigger the pipeline by making a code commit.
- Everything is tied together and the pipeline works!!!



Challenges:

- The decision of which CI tool to use was something I had to consider. Using jenkins or any other CI tool would have accomplish the job, but would also had come with the complexity of managing the server. I chose AWS CodeBuild because it is a managed service and I only pay for the build time, not for setting it up. In contrast, Jenkins will require me to run an EC2 instance continuously for as long as I need the server running, which is not cost effective. For an organization's build, Jenkins will be better due to the number of build process that will be going on
- Deciding which image to use during the creation of the code build process. Version 5 was used because it had jdk11, which I needed for this project.