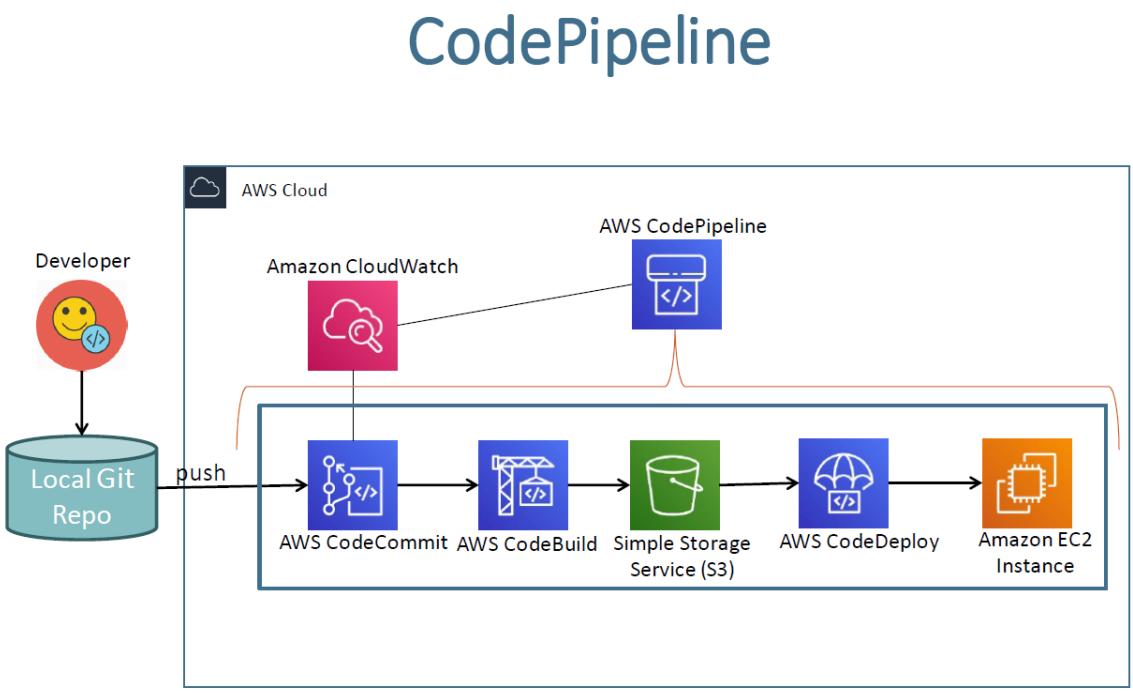
2. CI/CD in AWS - Part I

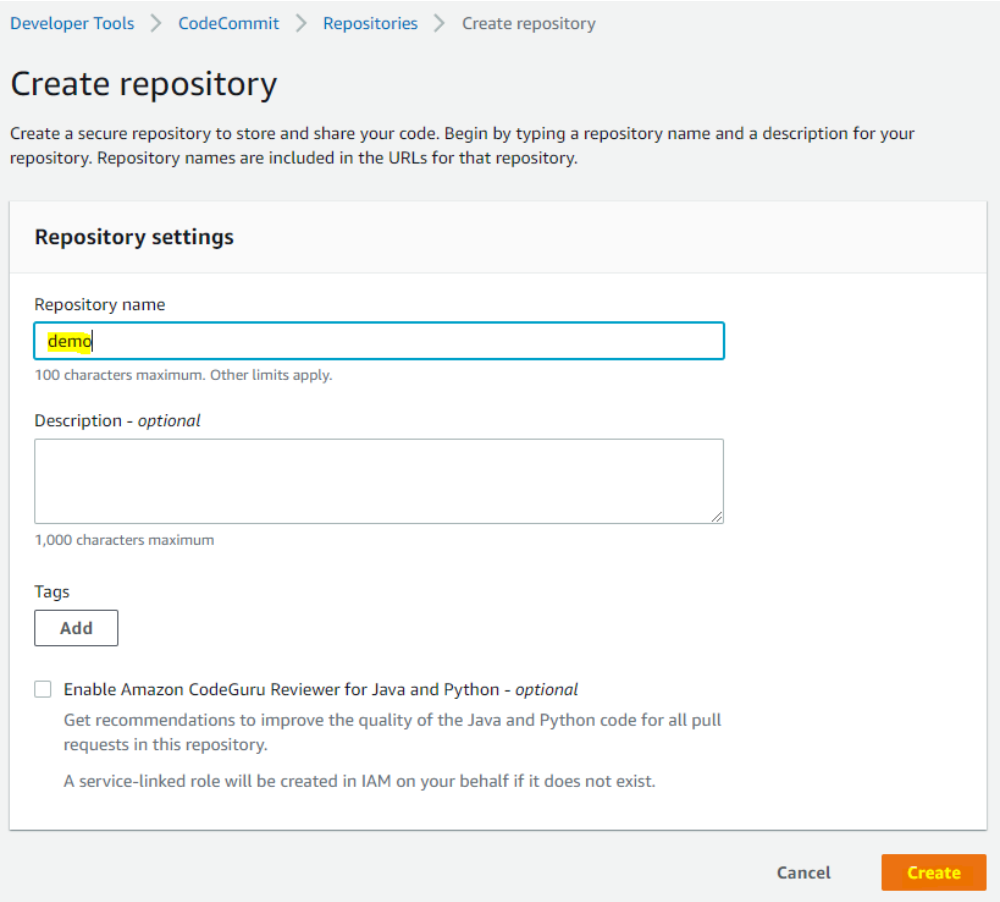
Objective -



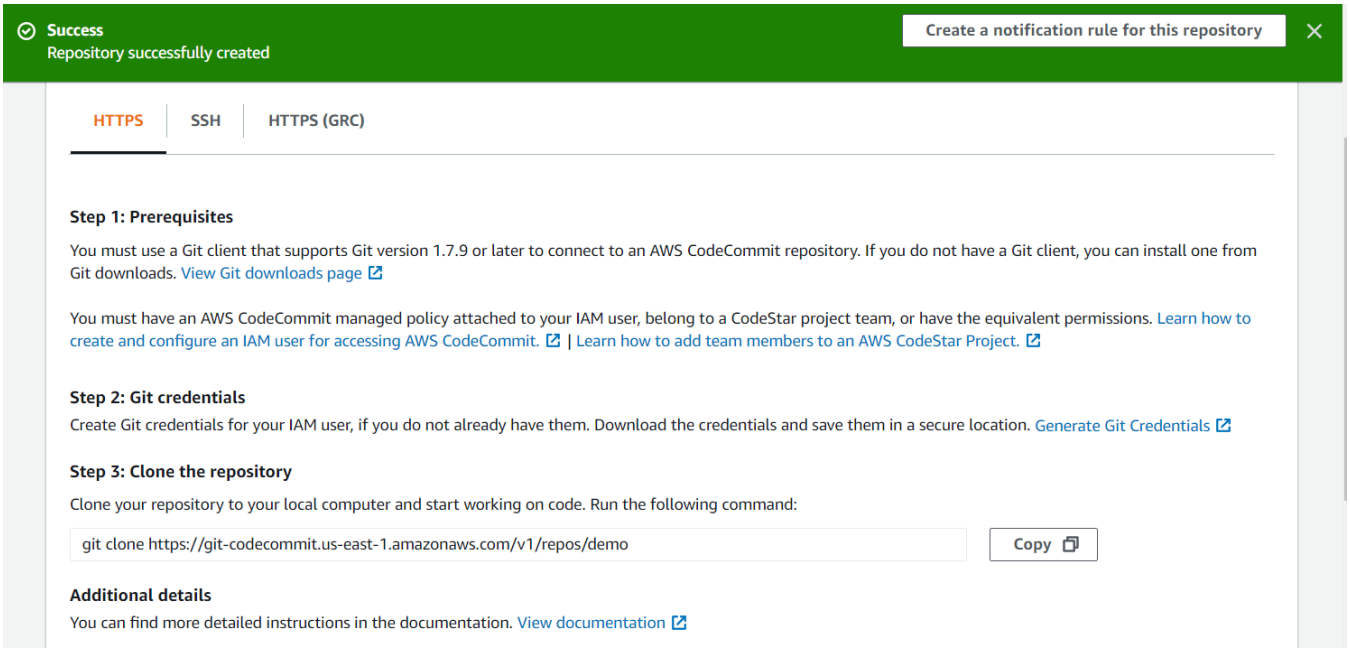
AWS CodeCommit

- AWS provided a Version control system offered by AWS
- **NOTE**: Access CodeCommit as an IAM User (NOT Root user)

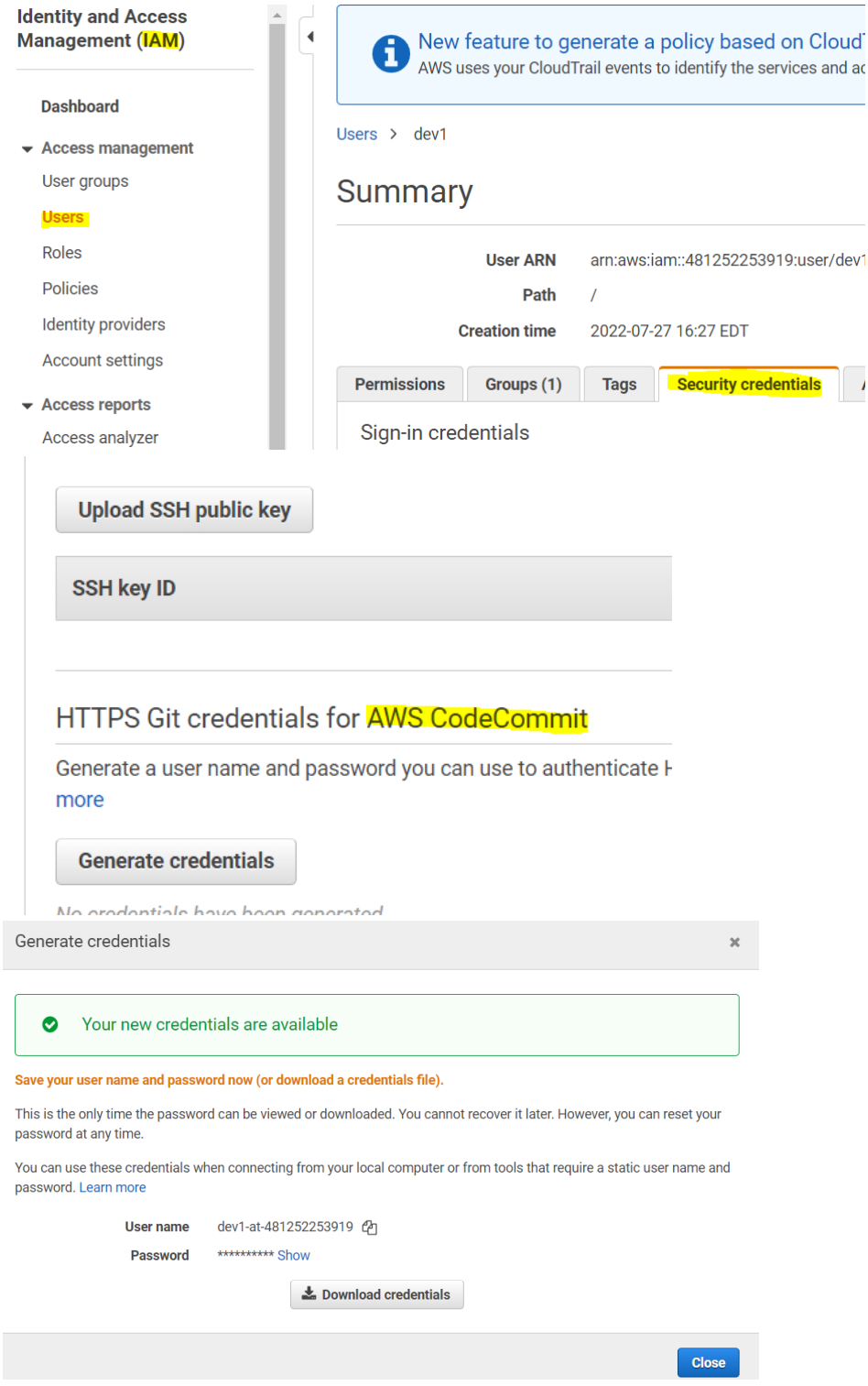
Step #1: Select CodeCommit Create Repository



Step #2: Copy and save the '*git clone {path}*'



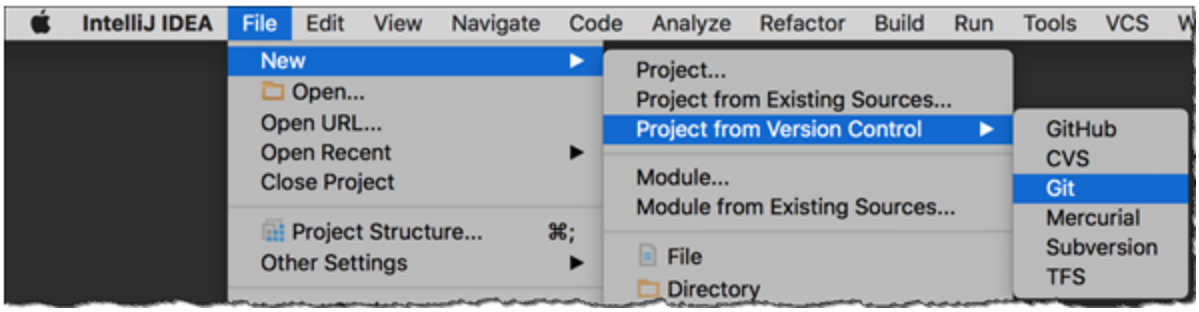
Step #3: Got to IAM Users {Your IAM user} click 'Security Credential' Scroll down to HTTPS Git credentials for AWS CodeCommit 'Generate Credentials'
Save the excel file with username and password

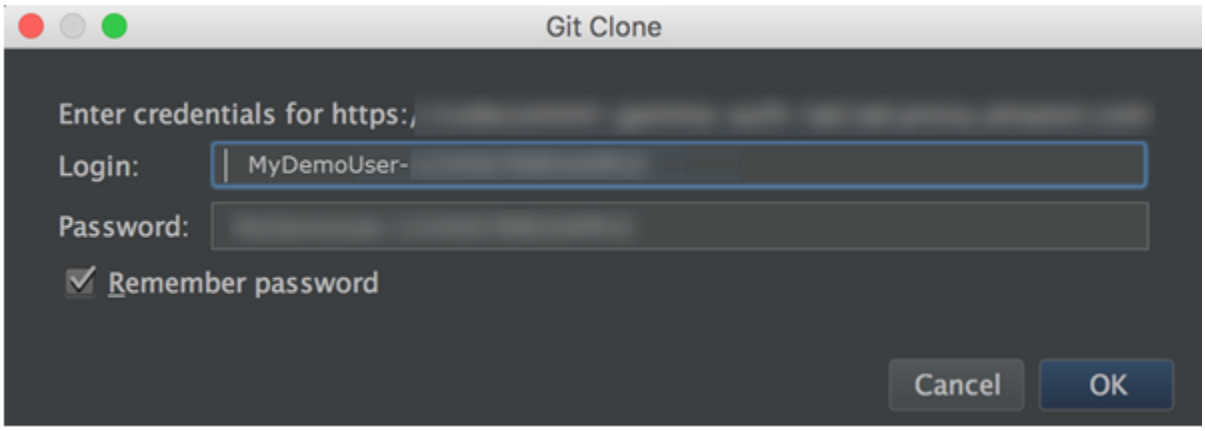


Step #4: To your local project, add git origin with HTTPS URL from Step #2 and credentials in Step #3. Push the code to CodeCommit.

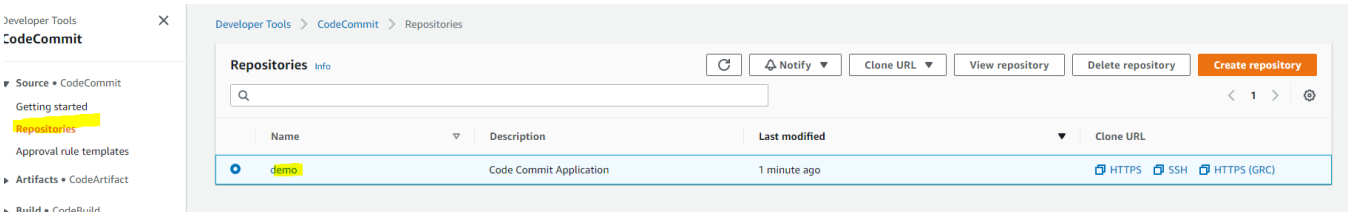
https://docs.aws.amazon.com/codecommit/latest/userguide/setting-up-ide.html

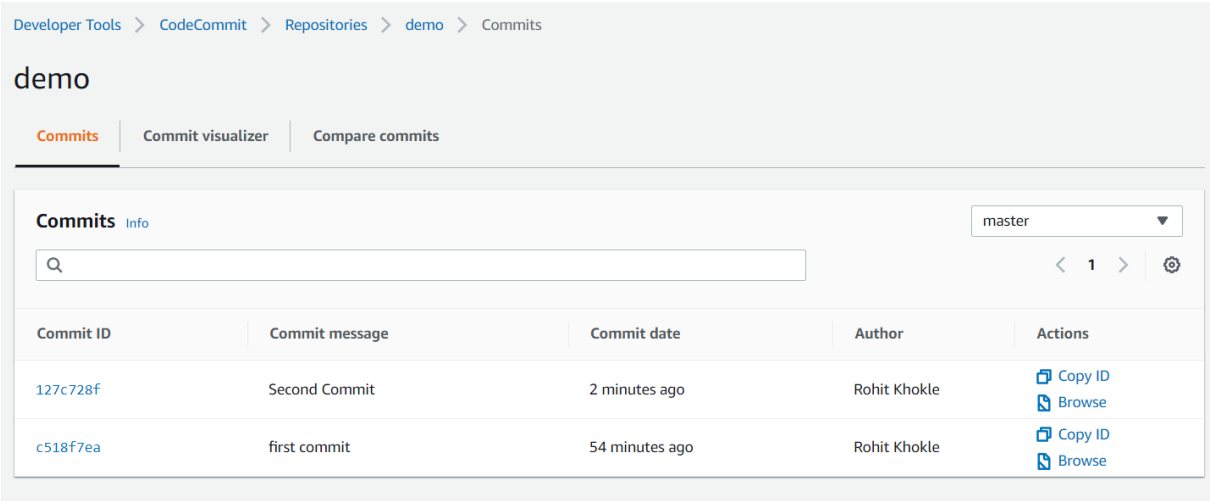
For IntelliJ:





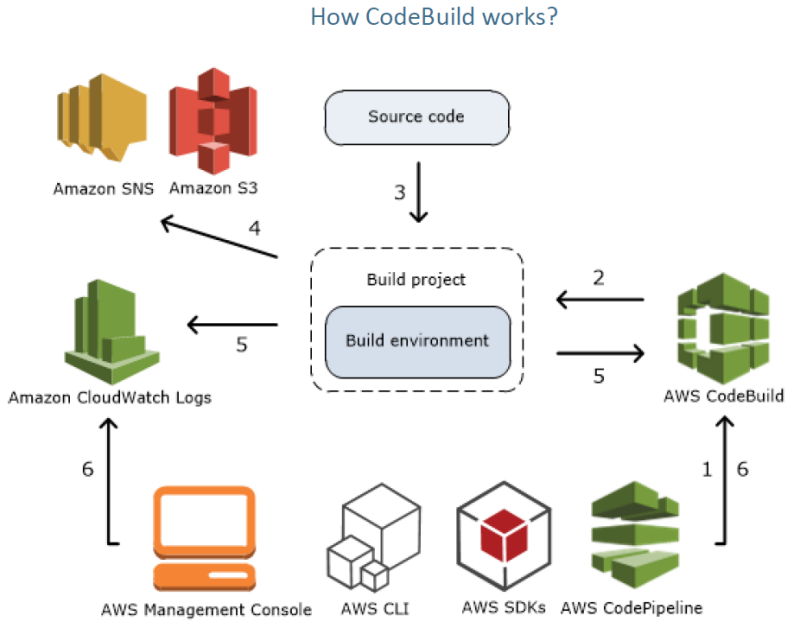
Step #5: View the repository and commits on CodeCommit branch



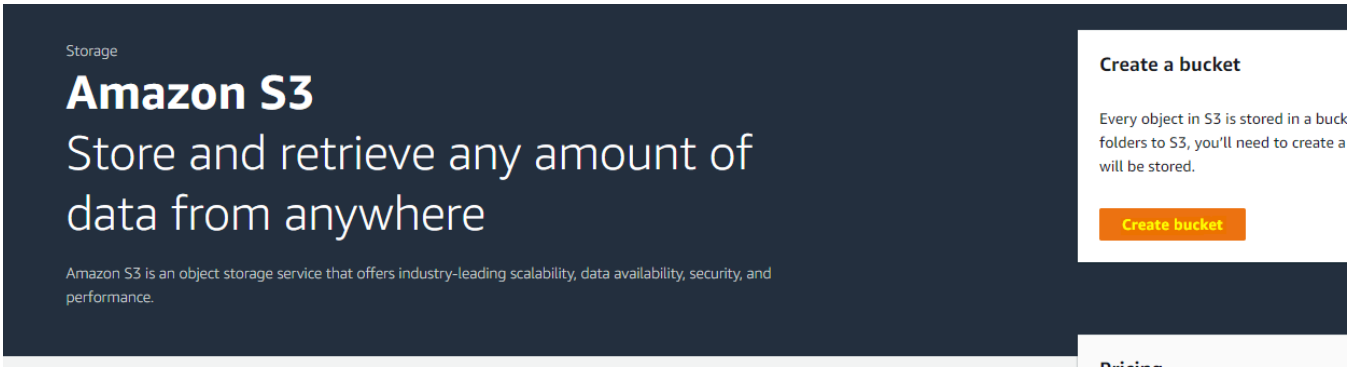


AWS CodeBuild

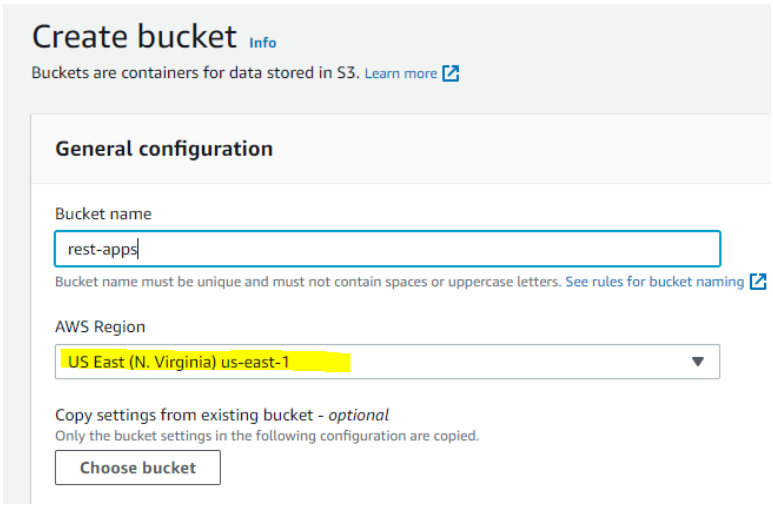
- Fully Managed Build Service in AWS
- Compiles source code, runs unit tests, and produces artifacts that are ready to deploy



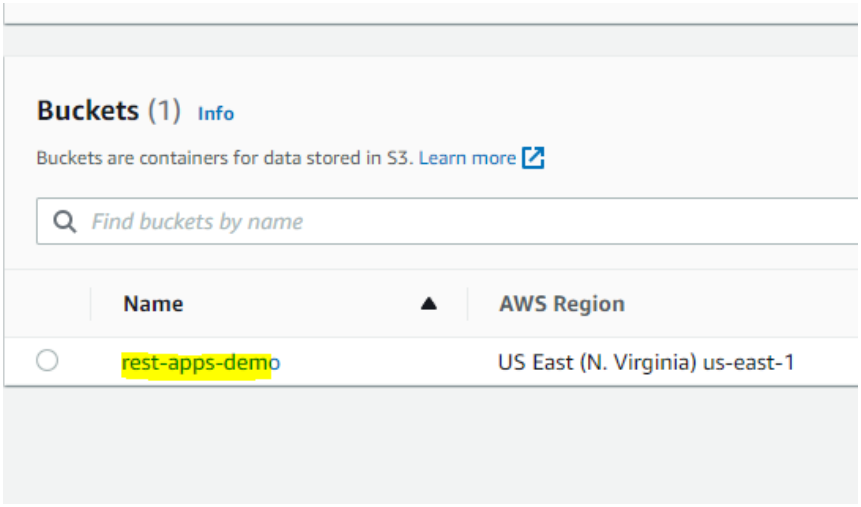
Step #1: AWS Console S3 Create Bucket

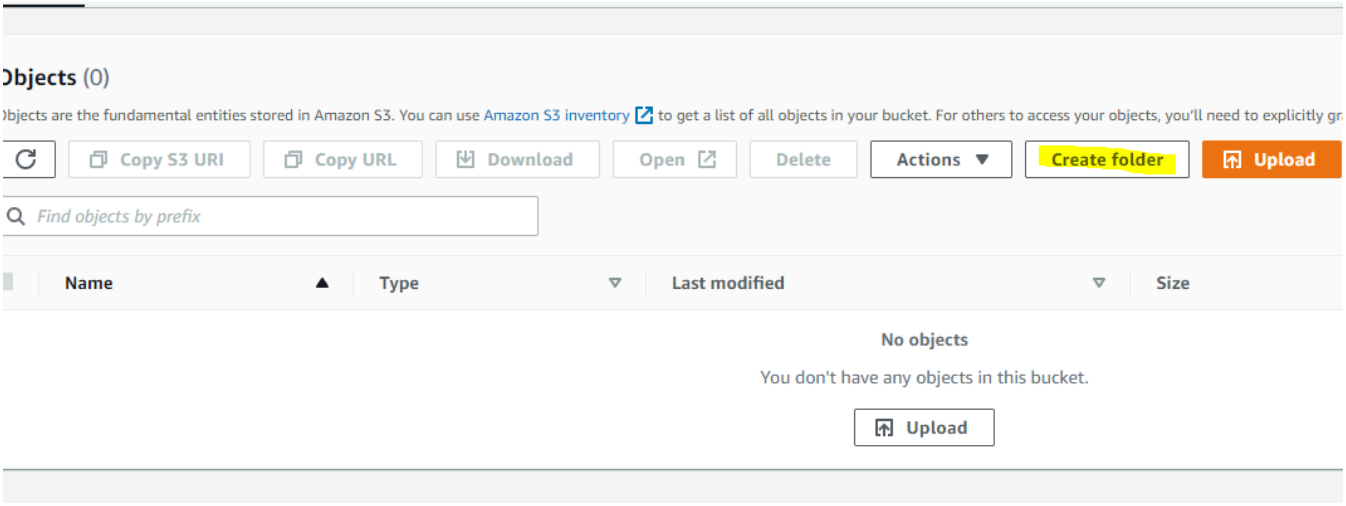


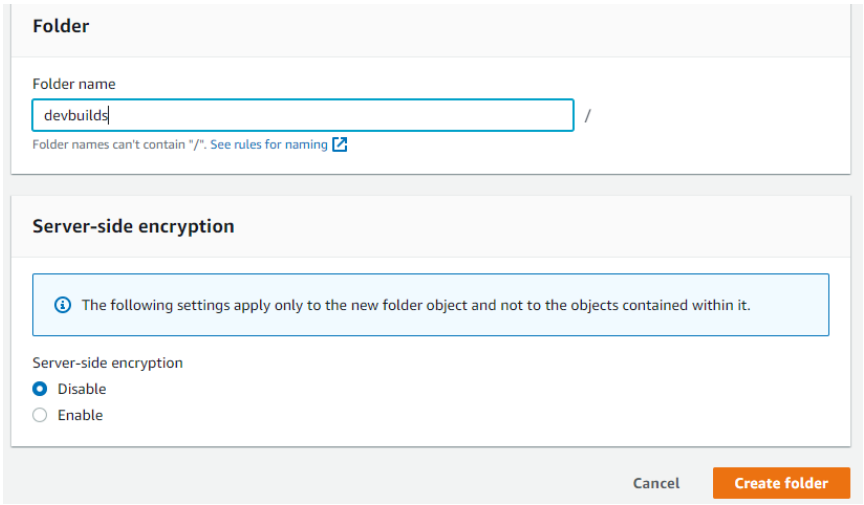
#1.1: Create Bucket with a name and region (**NOTE**: Region has to be same as CodeCommit region.)

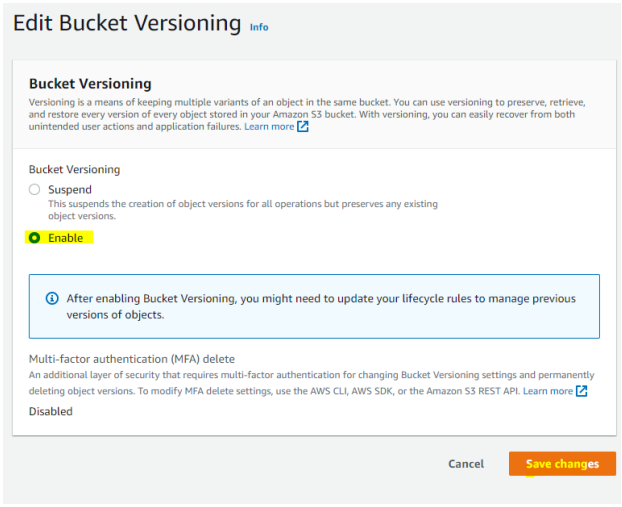


#**1.2**: Create a folder inside the newly created bucket

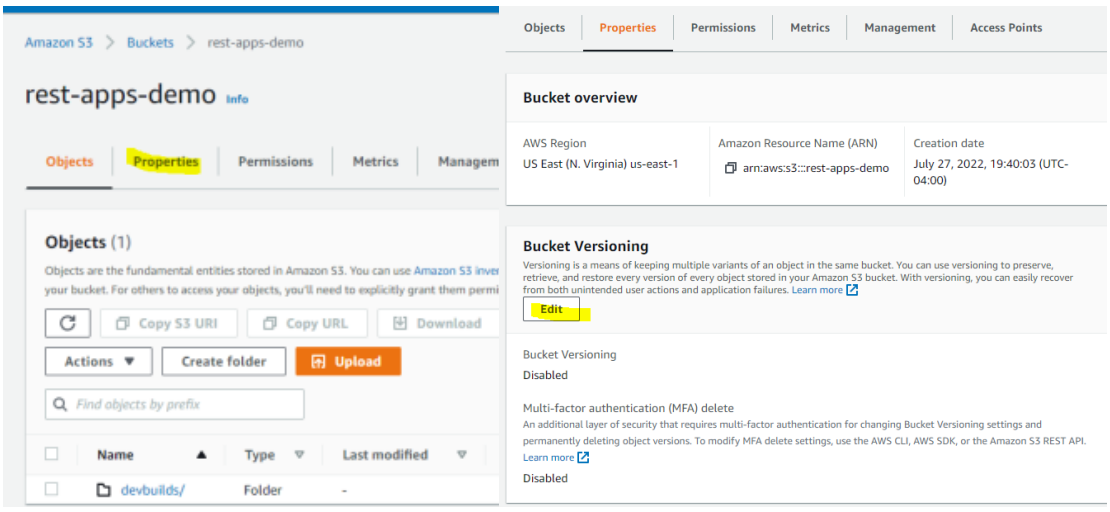




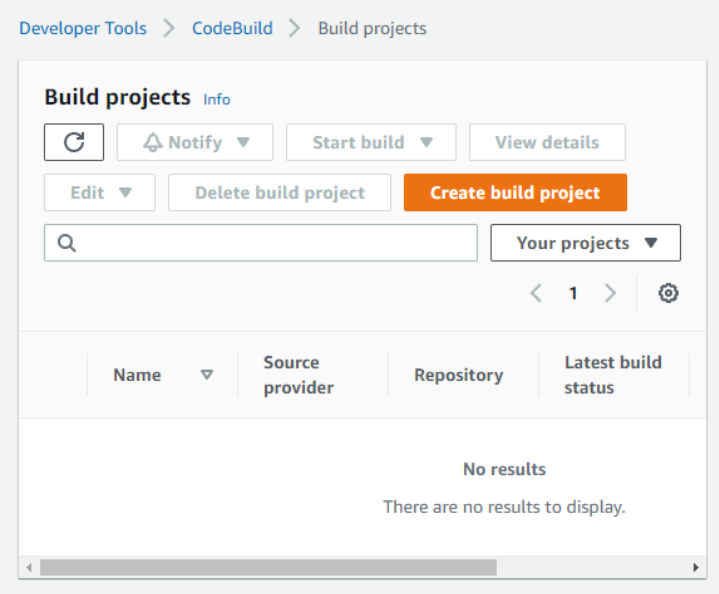




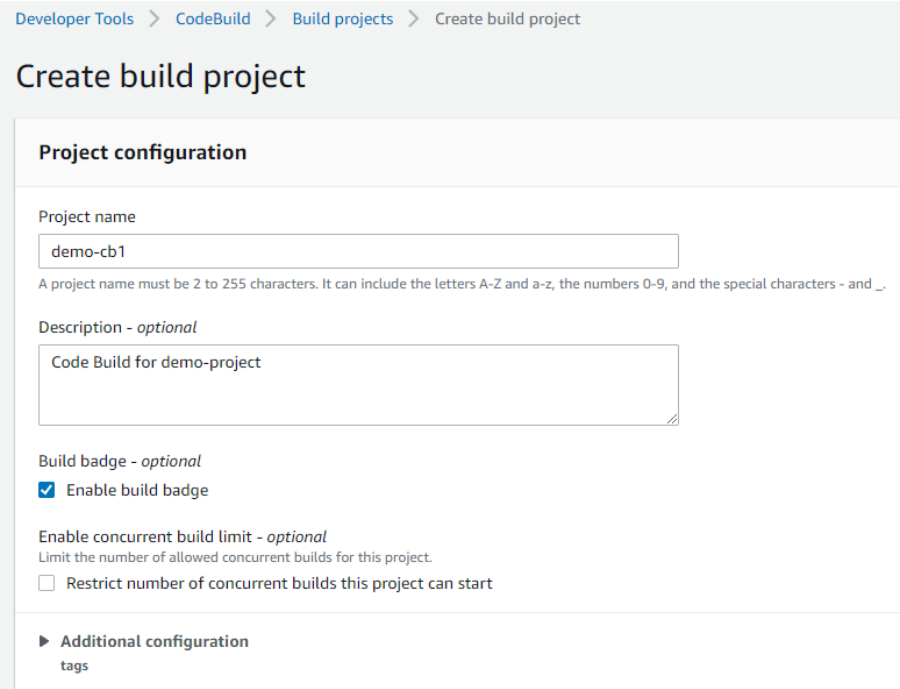
#1.3: Enable versioning



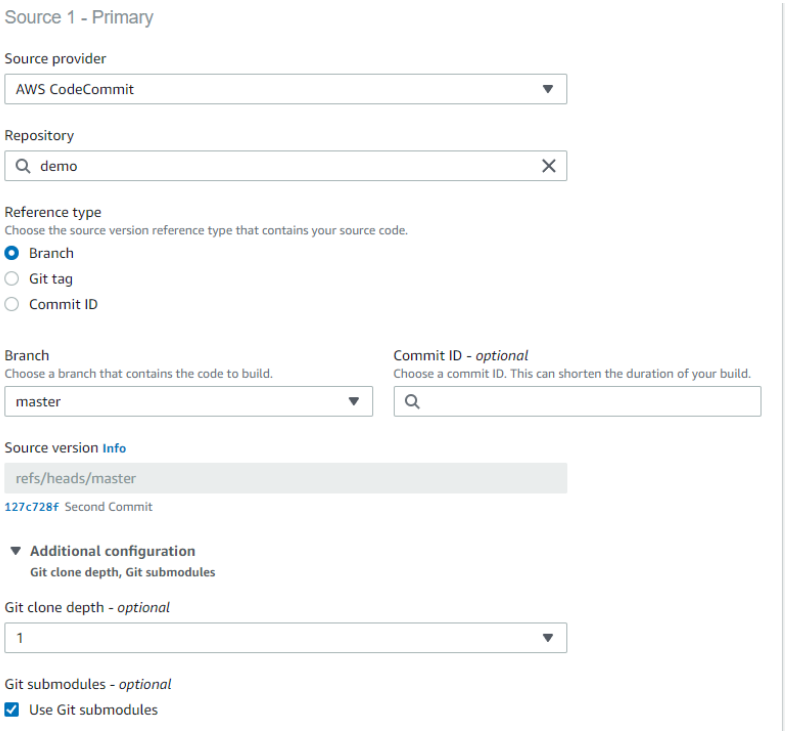
Step #2: AWS Console CodeBuild



#2.1: Project Configuration
Build Badge is notify the integrated applications (GitHub/Git Enterprise) if the build is successful/failed



#2.2: Source Provider



Git Clone depth # of commits that will be cloned. eg, if it is set to 5, it will clone/download the last 5 commits
Git submodules Let's use other repository in the project (different repos)

#2.3: Environment

Environment

Environment image

- Managed image
  Use an image managed by AWS CodeBuild
- Custom image
  Specify a Docker image

Operating system

Ubuntu

The programming language runtimes are now included in the standard image of Ubuntu 18.04, which is recommended for new CodeBuild projects created in the console. See Docker Images Provided by CodeBuild for details.

Runtime(s)

Standard

Image

aws/codebuild/standard:6.0

Image version

Always use the latest image for this runtime version

Environment type

Linux

Privileged

Enable this flag if you want to build Docker images or want your builds to get elevated privileges

Service role

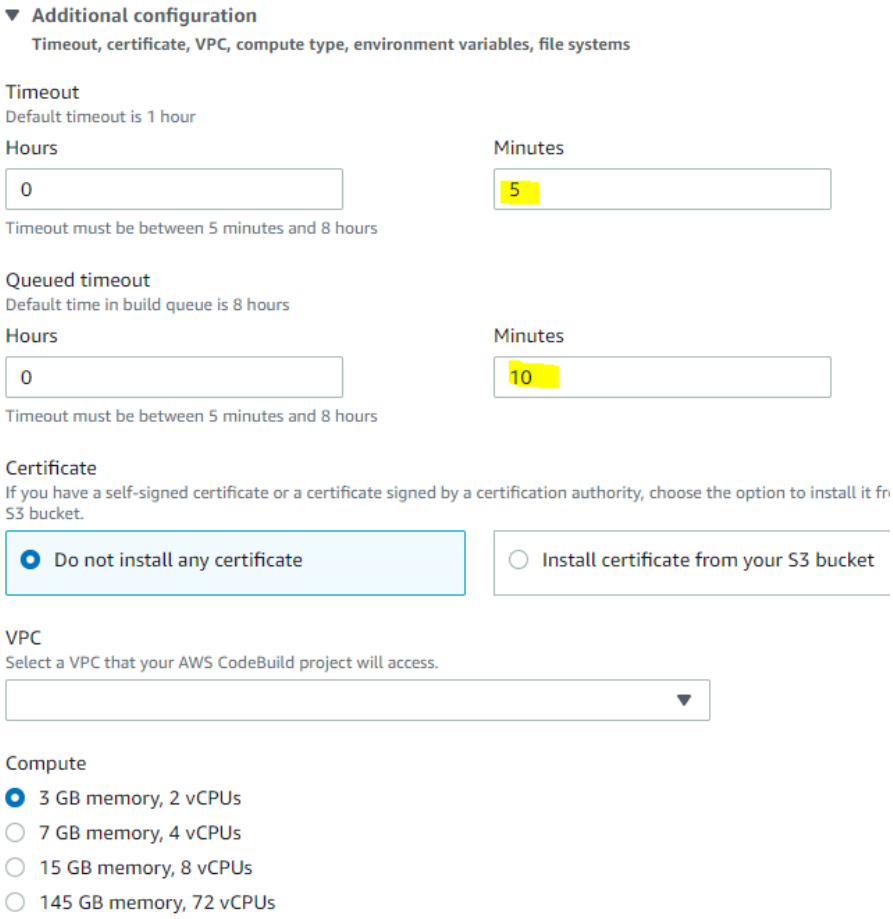
- New service role
  Create a service role in your account
- Existing service role
  Choose an existing service role from your account

Role name

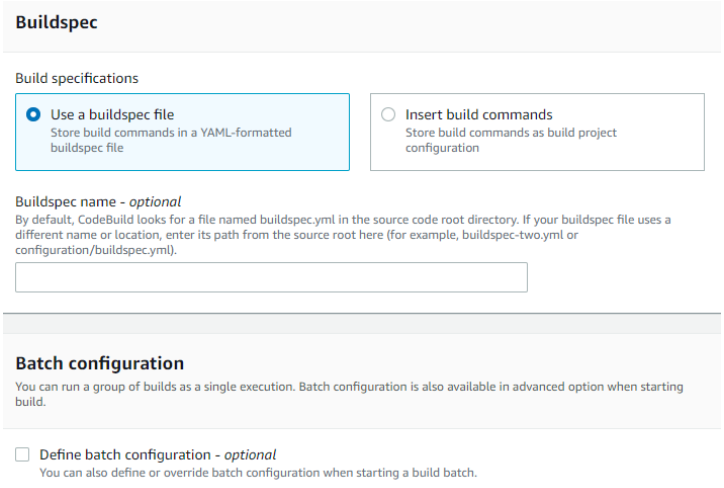
codebuild-demo-cb1-service-role

Type your service role name

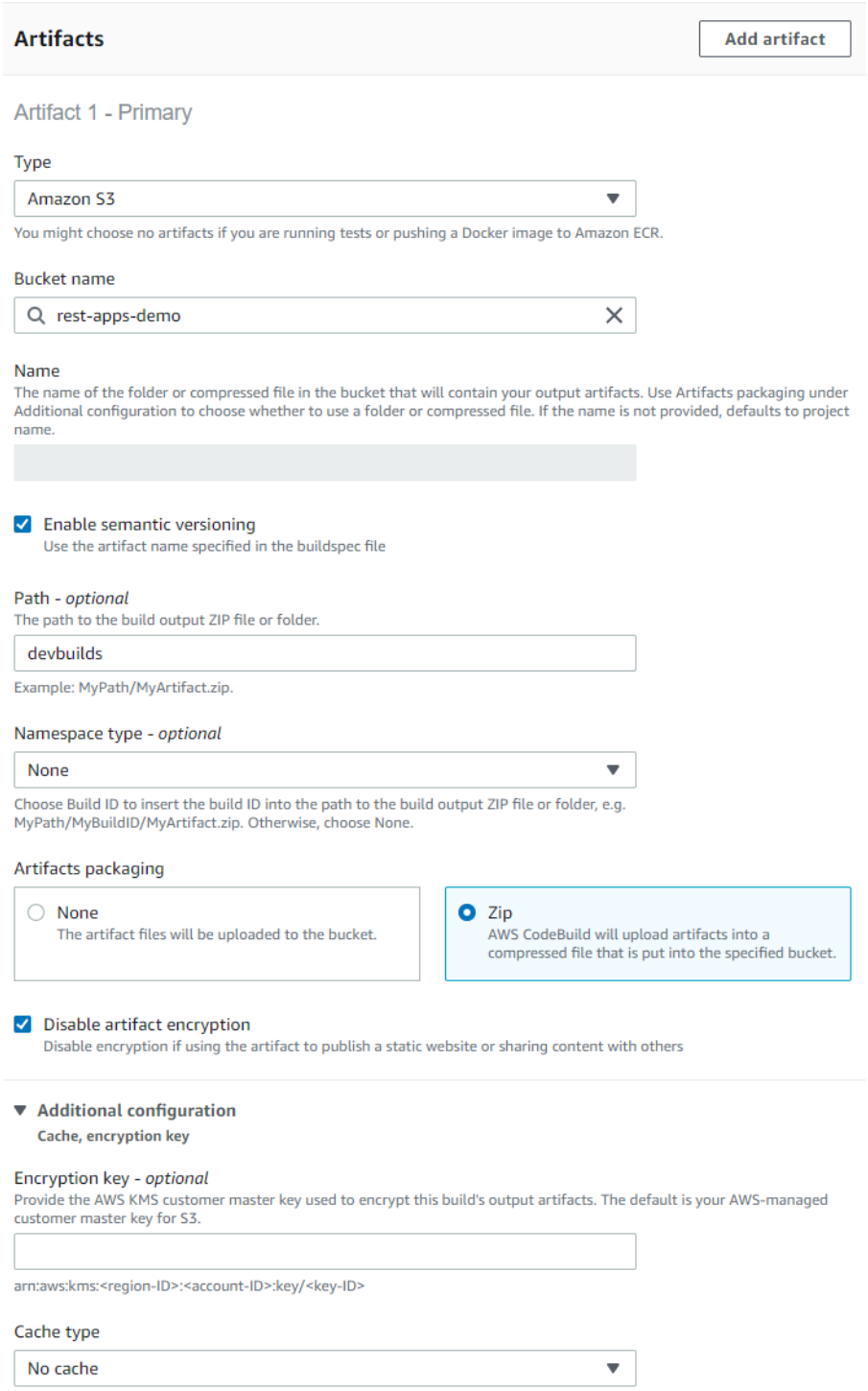
▶ **Additional configuration**
Timeout, certificate, VPC, compute type, environment variables, file systems



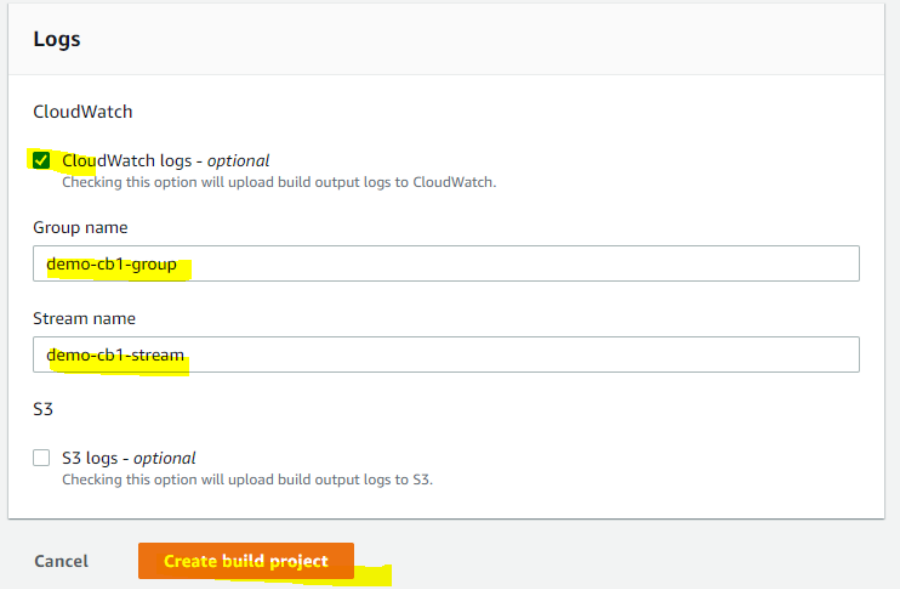
#**2.4**: Build Spec
-- Custom Image can be selected for Docker



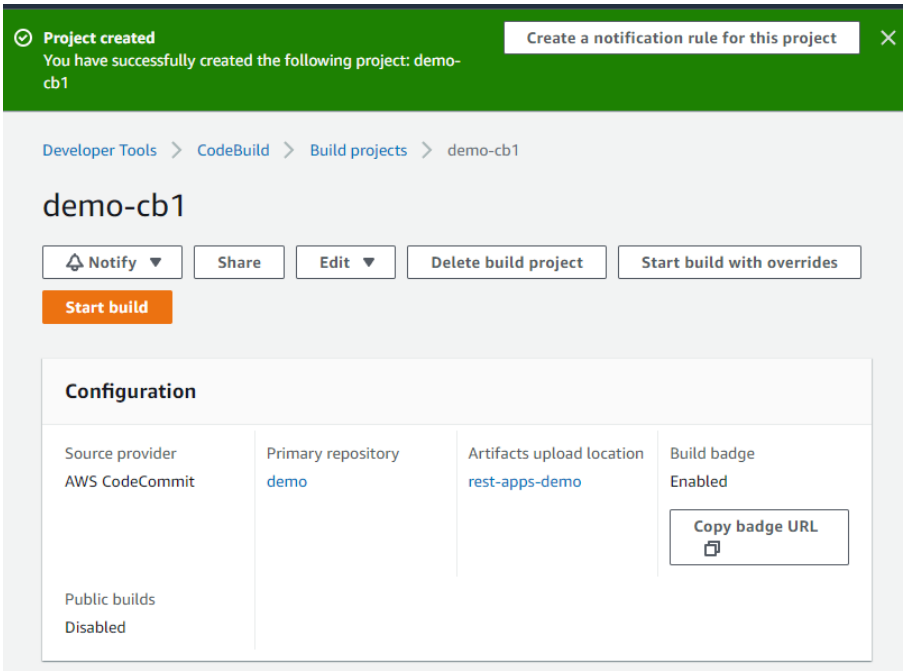
#**2.5**: Artifacts


Artifacts
Add artifact
Artifact 1 - Primary
Type
Amazon S3
You might choose no artifacts if you are running tests or pushing a Docker image to Amazon ECR.
Bucket name
rest-apps-demo
Name
The name of the folder or compressed file in the bucket that will contain your output artifacts. Use Artifacts packaging under Additional configuration to choose whether to use a folder or compressed file. If the name is not provided, defaults to project name.
Enable semantic versioning
Use the artifact name specified in the buildspec file
Path - optional
The path to the build output ZIP file or folder.
devbuilds
Example: MyPath/MyArtifact.zip.
Namespace type - optional
None
Choose Build ID to insert the build ID into the path to the build output ZIP file or folder, e.g. MyPath/MyBuildID/MyArtifact.zip. Otherwise, choose None.
Artifacts packaging
None
The artifact files will be uploaded to the bucket.
Zip
AWS CodeBuild will upload artifacts into a compressed file that is put into the specified bucket.
Disable artifact encryption
Disable encryption if using the artifact to publish a static website or sharing content with others
Additional configuration
Cache, encryption key
Encryption key - optional
Provide the AWS KMS customer master key used to encrypt this build's output artifacts. The default is your AWS-managed customer master key for S3.
arn:aws:kms:<region-ID>:<account-ID>:key/<key-ID>
Cache type
No cache

#**2.6**: Logs



Create Build project



- Start the build we require a buildspec.yml

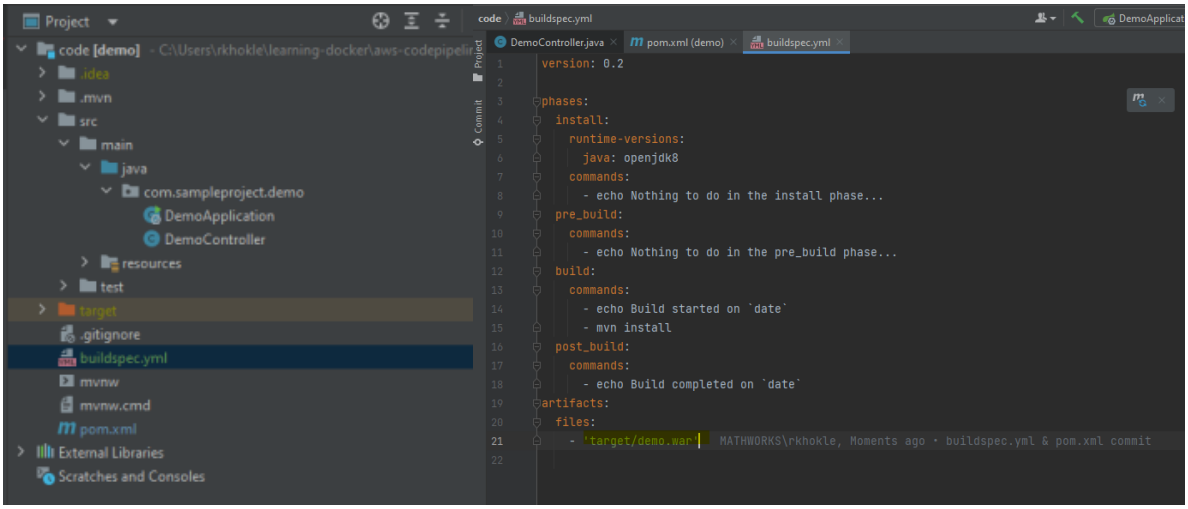
Create buildspec.yml

Step #1: Create buildspec.yml and check in code

1.1: In pom.xml, add finalName as artifactId

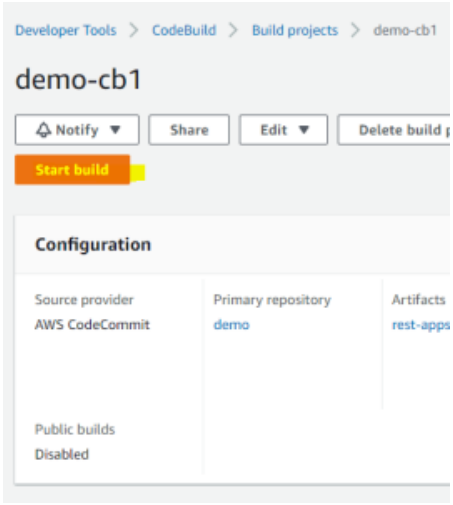
```
            <scope>test</scope>
        </dependency>
    </dependencies>

    <build>
        <plugins>
            <plugin>
                <groupId>org.springframework.boot</groupId>
                <artifactId>spring-boot-maven-plugin</artifactId>
            </plugin>
        </plugins>
        <finalName>${project.artifactId}</finalName>
    </build>

</project>
```

1.2: Create a new file with name - buildspec.yml

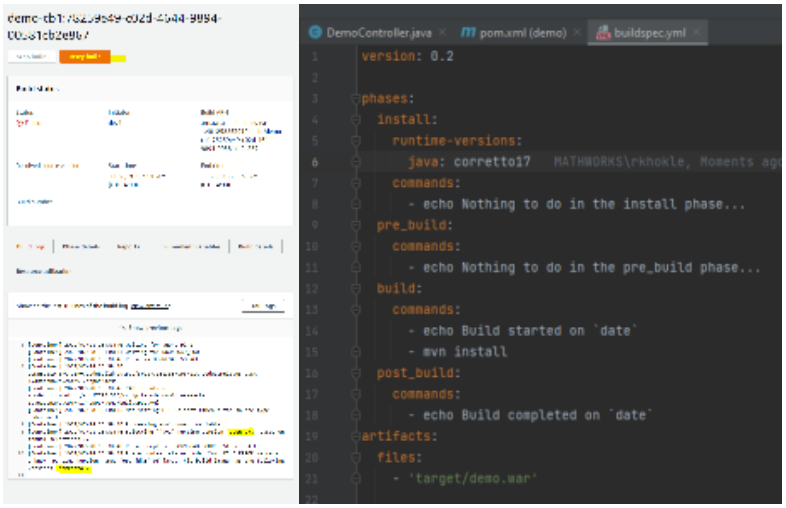


Commit and push above changes to CodeCommit

1.3: Build Project in CodeBuild

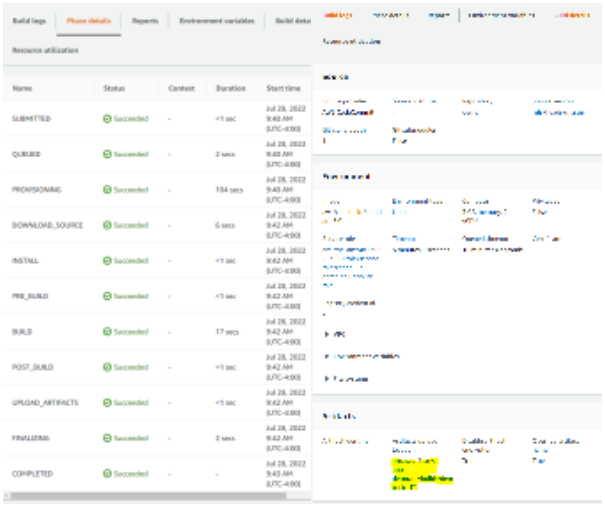


NOTE: Results in an error, troubleshoot solution below, change jdk8 in spec file to corretto17 and retry build.

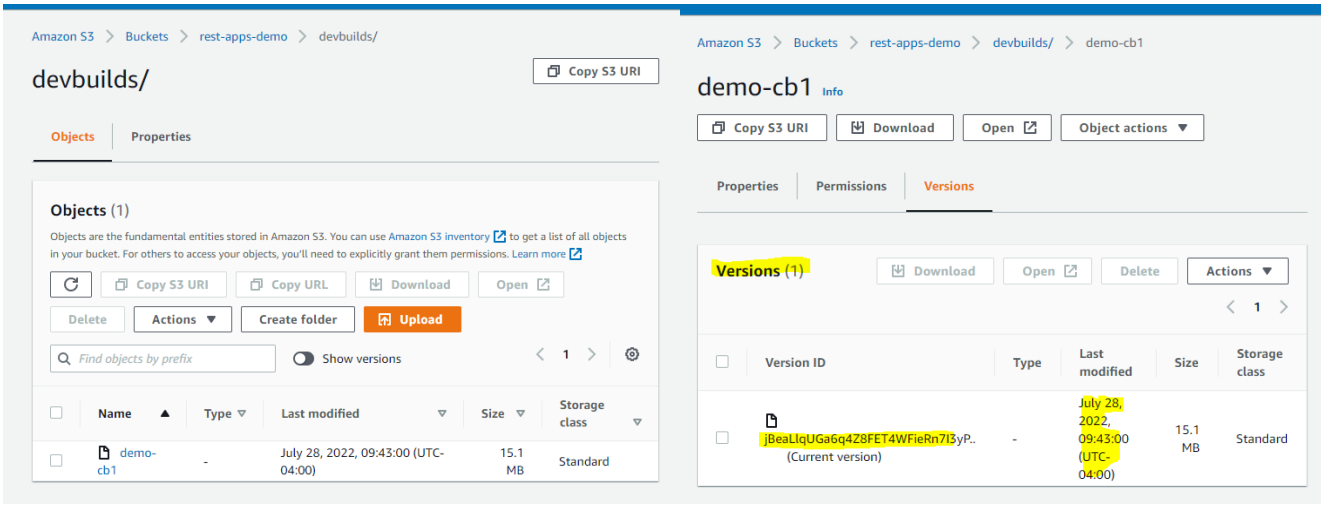



```
version: 0.2

phases:
  install:
    runtime-versions:
      java: corretto17
    commands:
      - echo Nothing to do in the install phase...
  pre_build:
    commands:
      - echo Nothing to do in the pre_build phase...
  build:
    commands:
      - echo Build started on `date`
      - mvn install
  post_build:
    commands:
      - echo Build completed on `date`
artifacts:
  files:
    - 'target/demo.war'
```

Step #2: Start Build, Verify Build Logs, Verify and phase details



Step #3: Download Artifacts from S3, unzip and review



Step #4: Run one more build and see versioning in S3