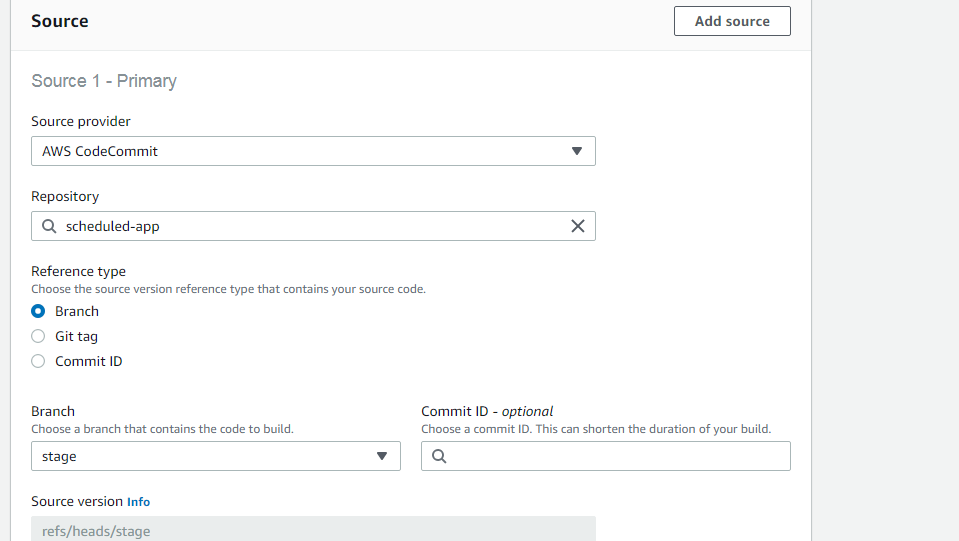
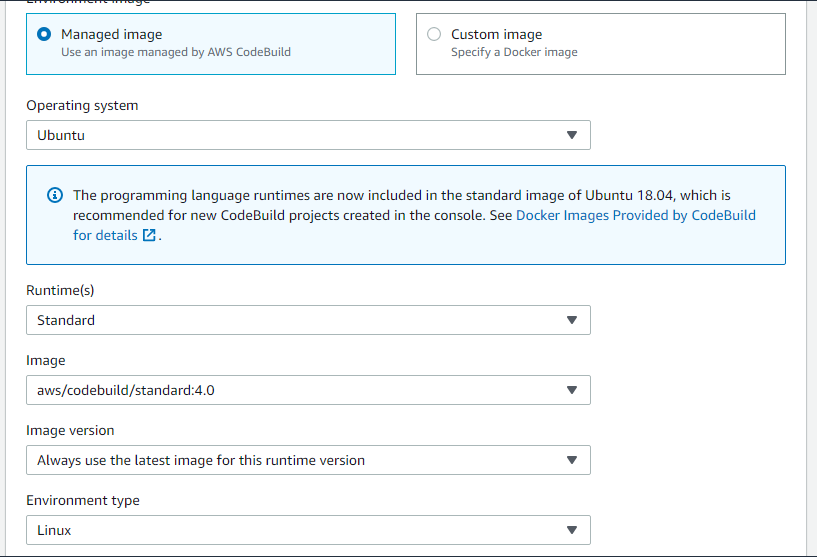
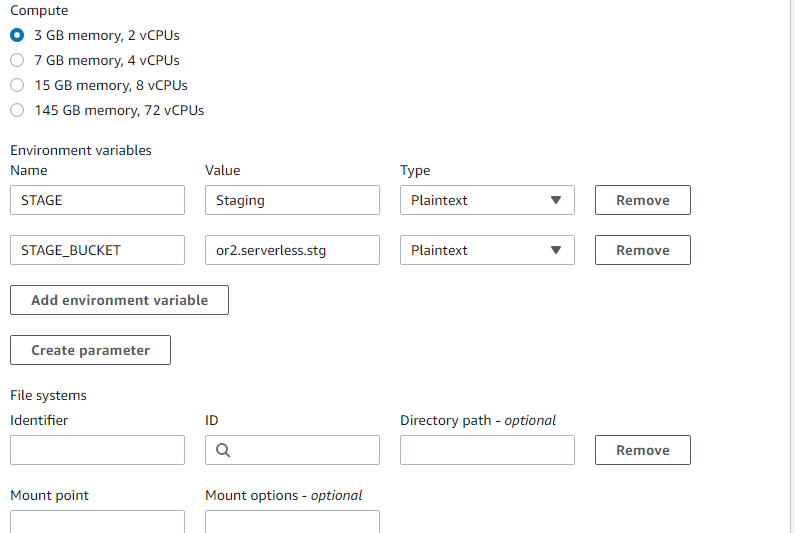
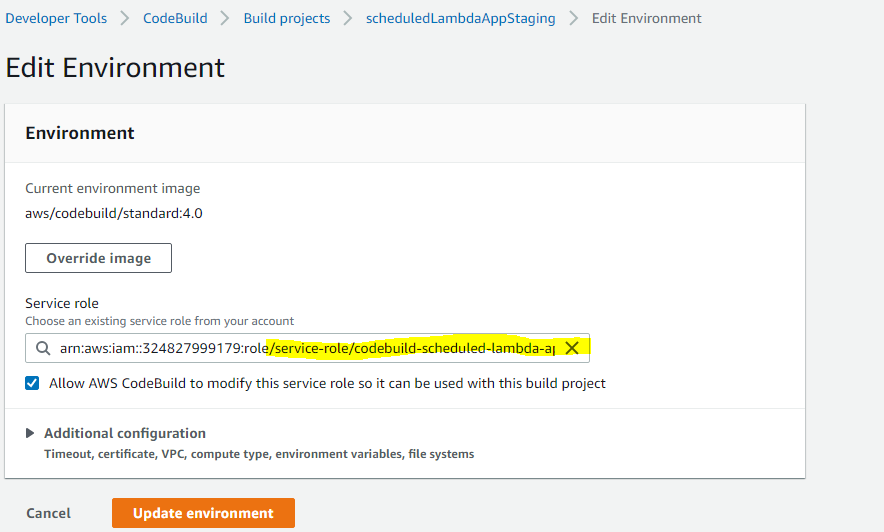
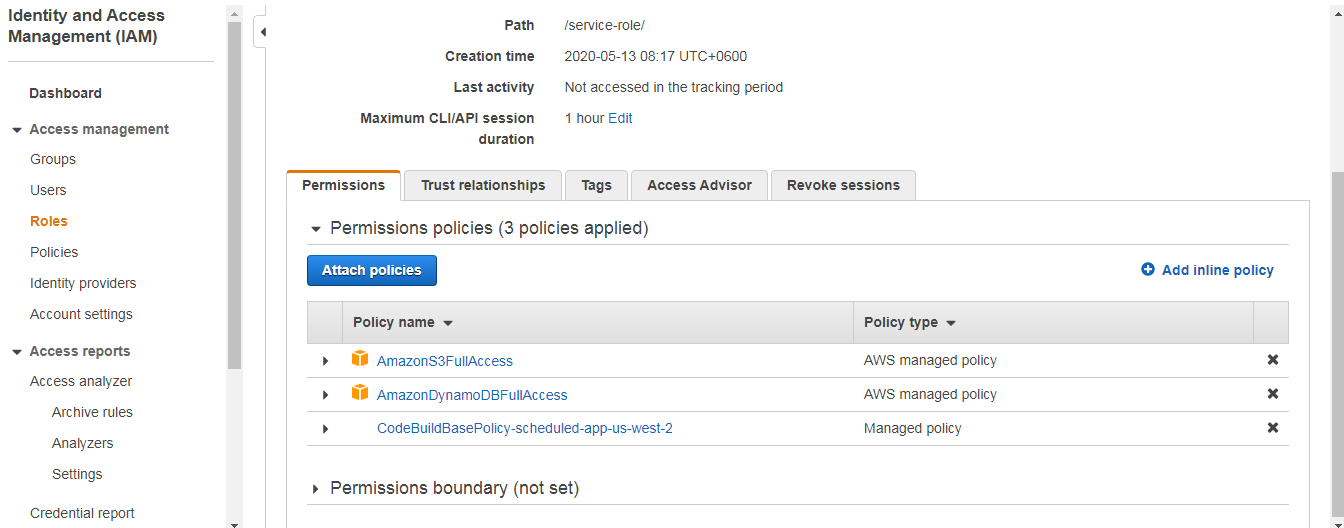
**Steps**

**1.Create a Cloudformation IAM role:** Goto AWS IAM and create a new IAM role for Cloudformation and give it **AdministratorAccess** and copy its ARN

2. **Create two CodeBuild projects for stage and master branch:**

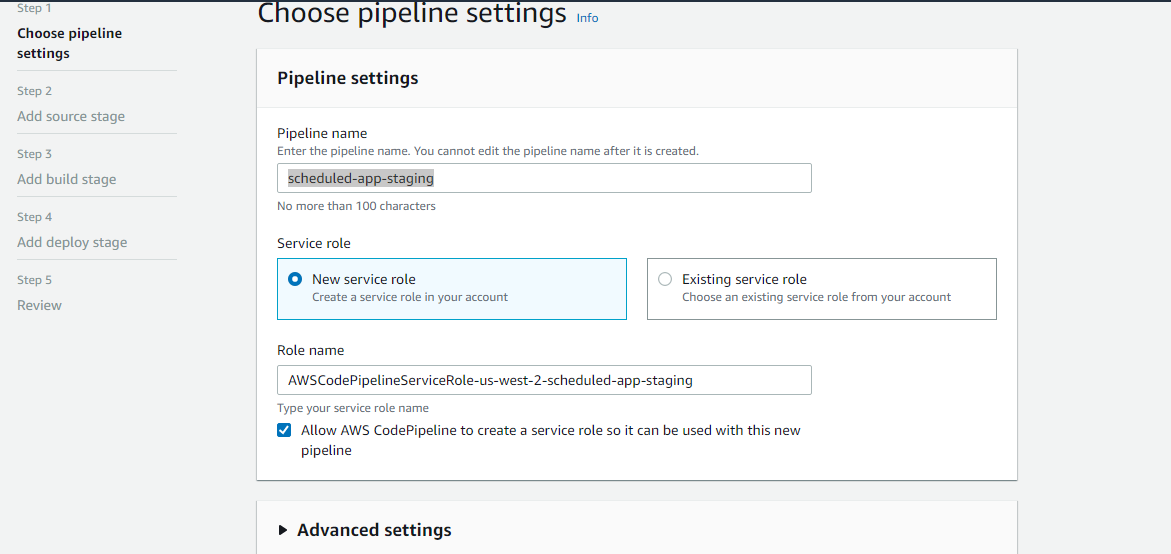
For stage branch:

* Goto AWS Code build and create a project called “scheduledLambdaAppStaging”
* Add staging branch from AWS Code commit 
* Select **Ubuntu** at the environment section and **stage** for branch: 
* Expand the **Additional configuration** section and set the Environment variables for **STAGE** and **STAGE\_BUCKET:** 
* Take note of the service role code build is using and find it at AWS IAM and add S3 and Dynamodb access to it:  
* Create another AWS Codebuild project called “scheduledLambdaAppProduction” and this time set the source branch to **Master** and set the Environment variables at the Additional Config section where **STAGE** will “Production” and **STAGE\_BUCKET** will be like “or.serverless.prod”
* Update it’s service role permission as well for S3 and DynamoDB or use the previous one
* Run both CodeBuild projects and see if they builds successfully

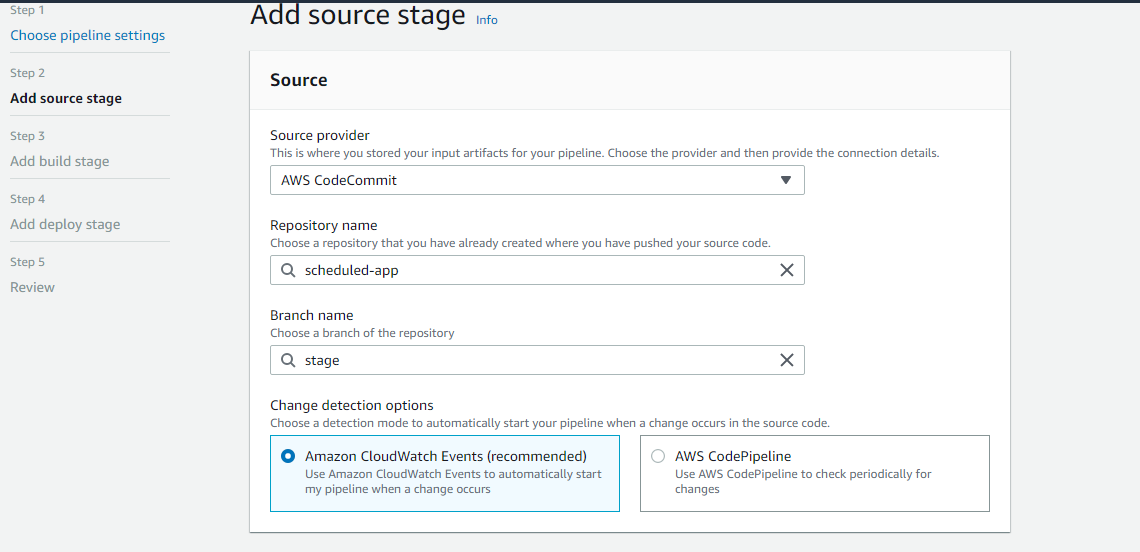
**3. Create the pipeline for staging:**

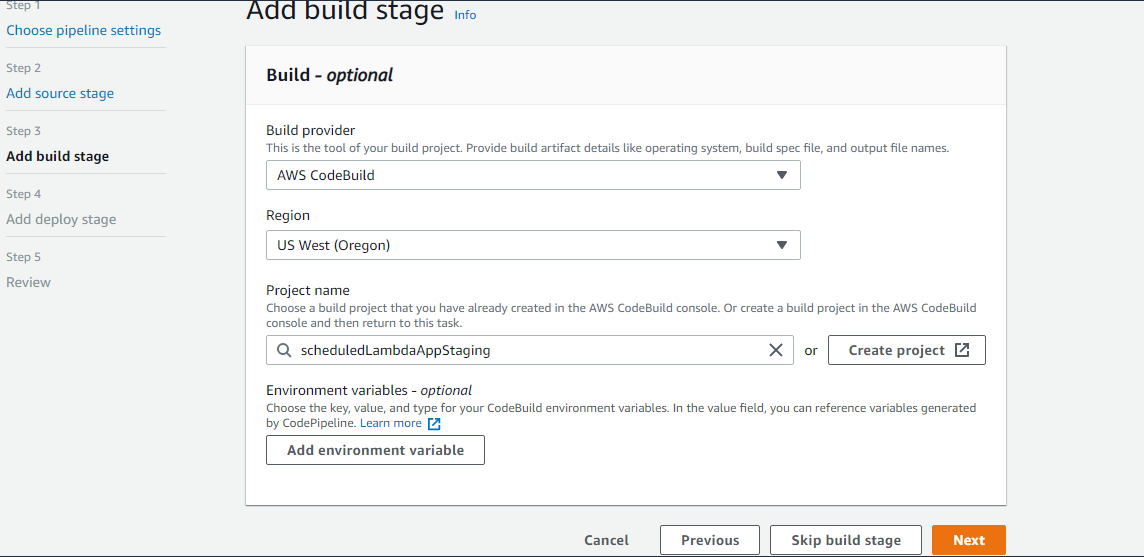
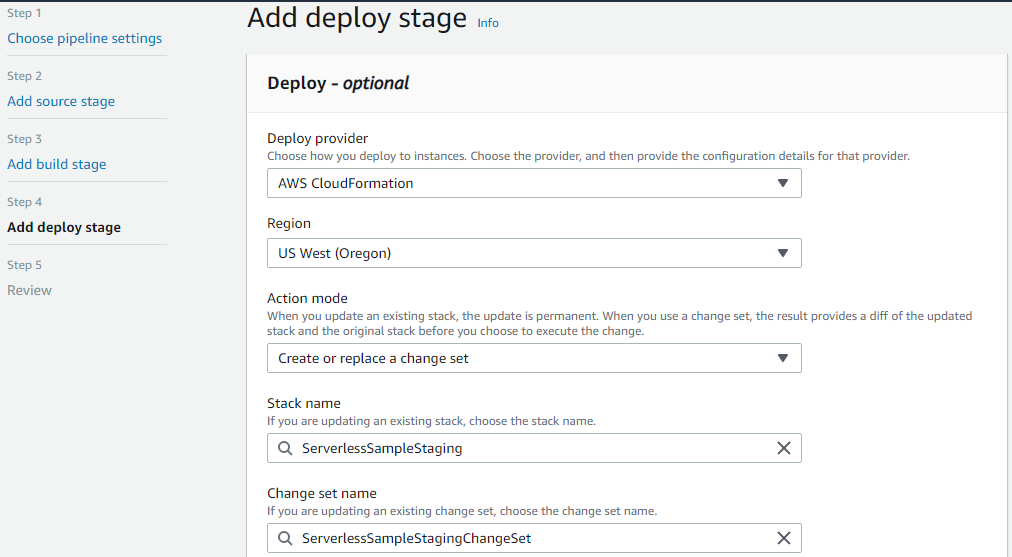
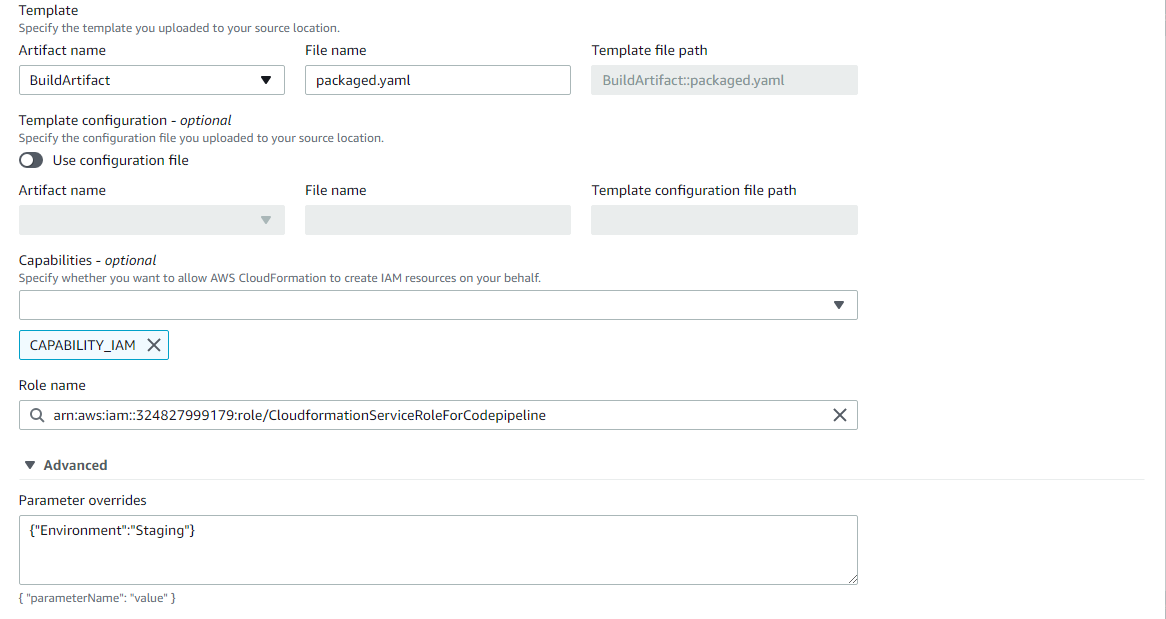
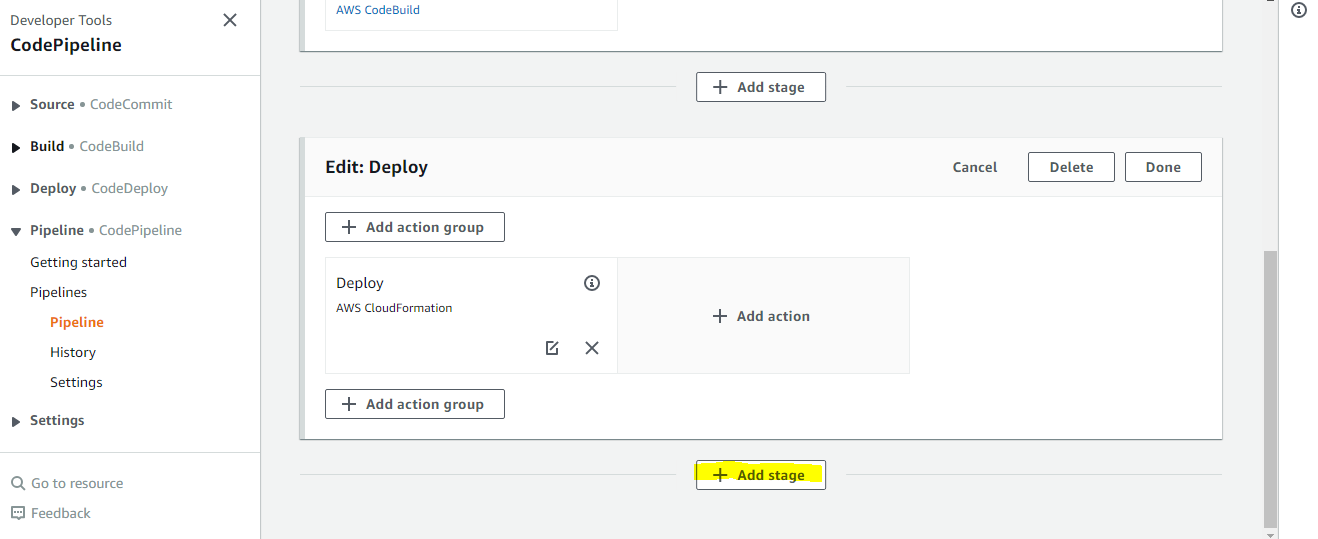
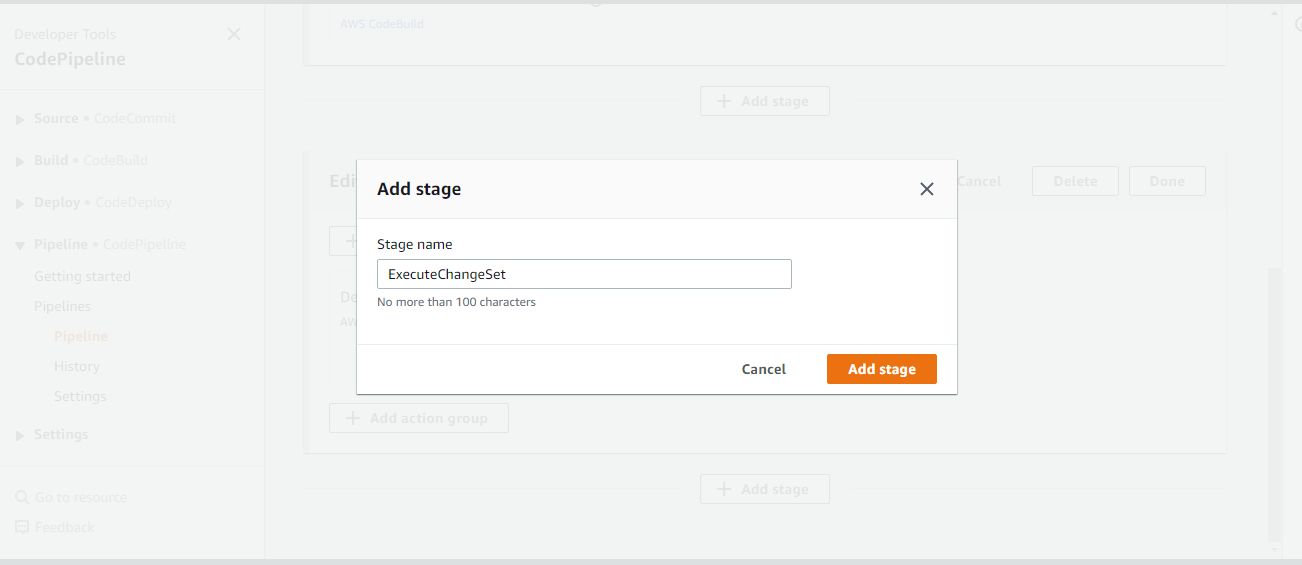
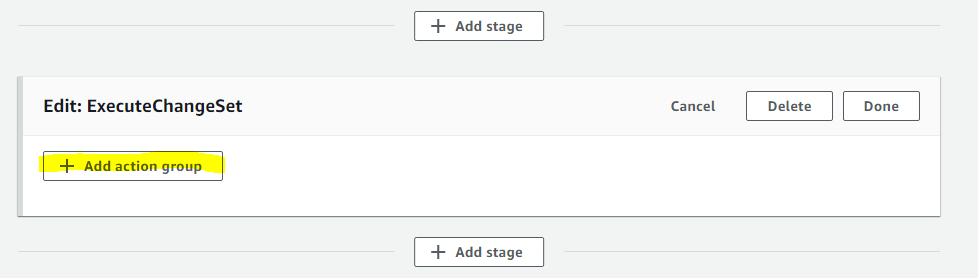
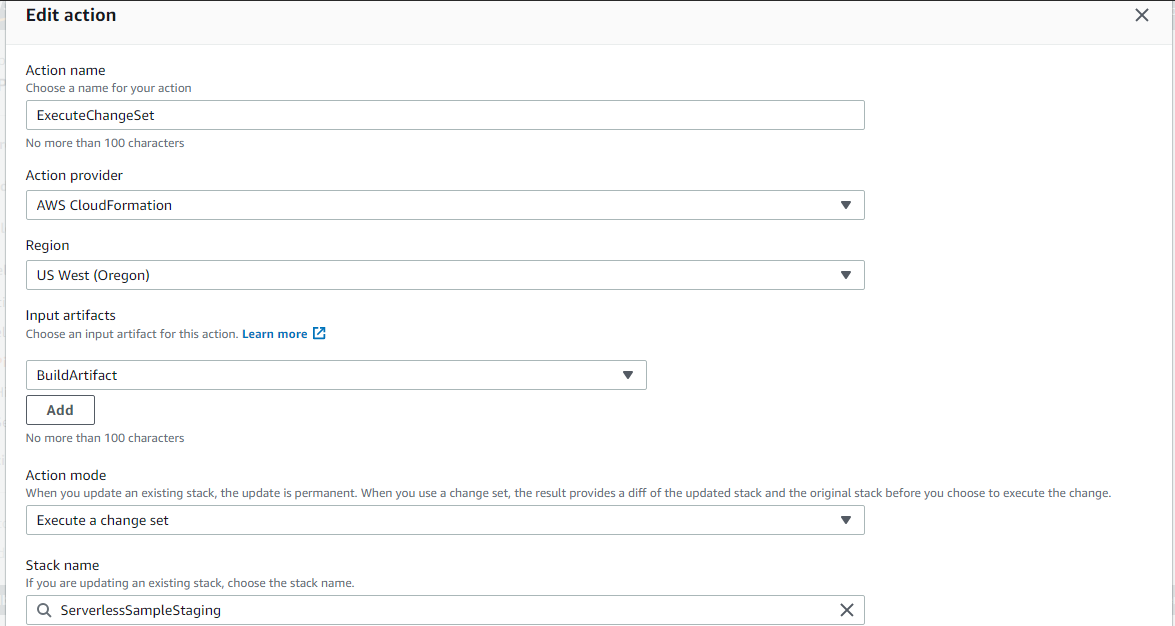
Goto AWS Code pipeline and create the staging pipeline following steps below:

* Choose pipeline settings

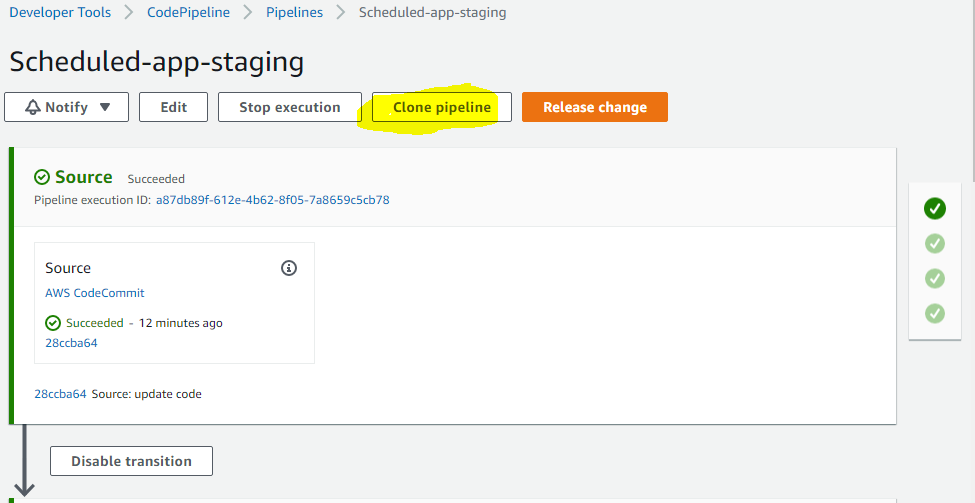
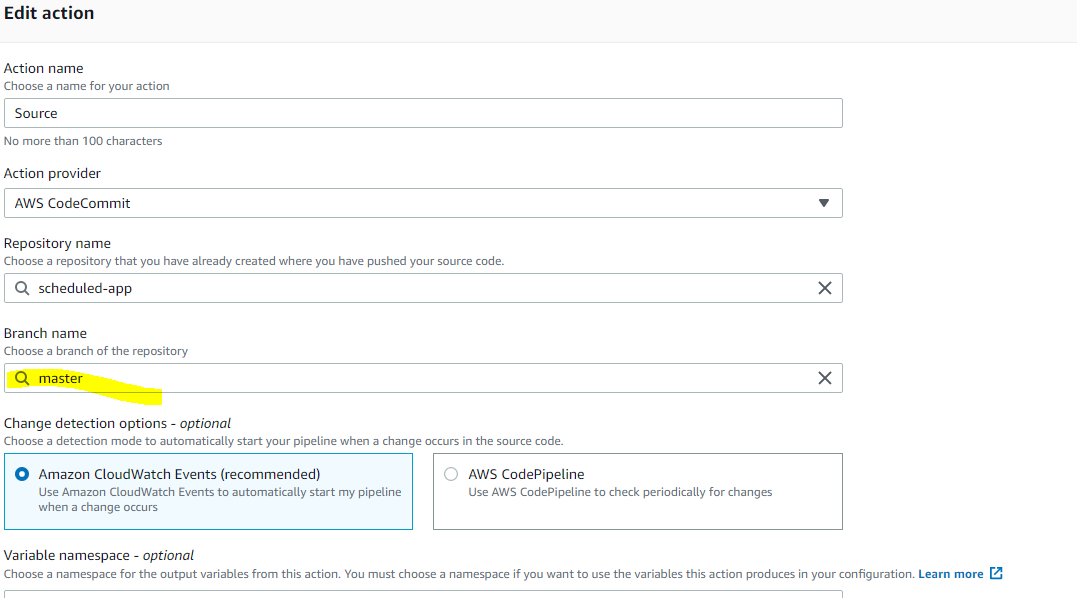
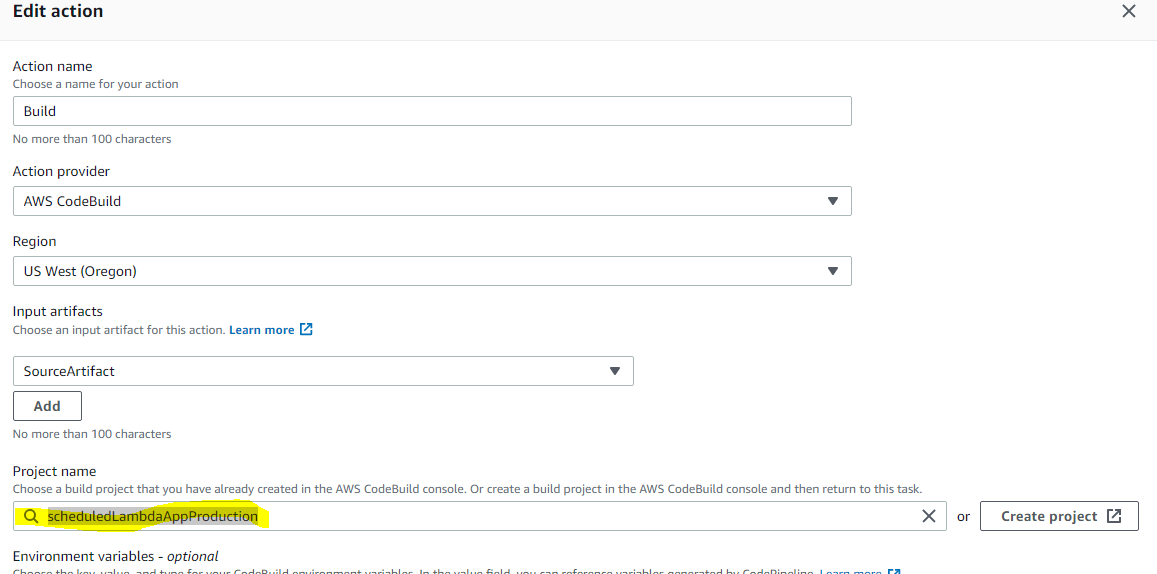
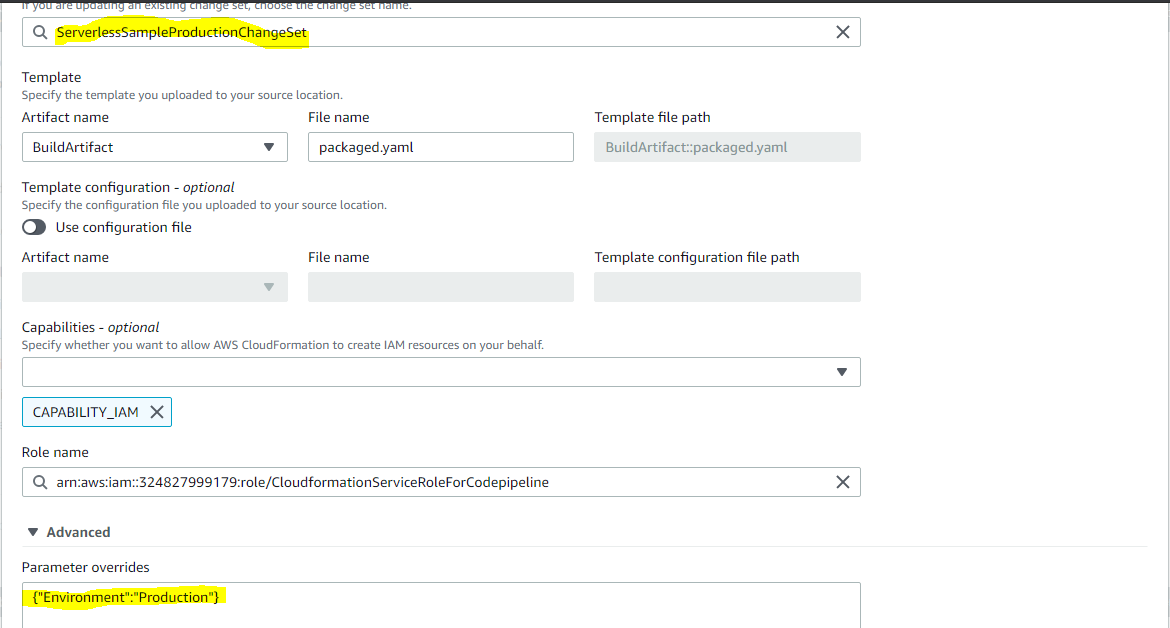
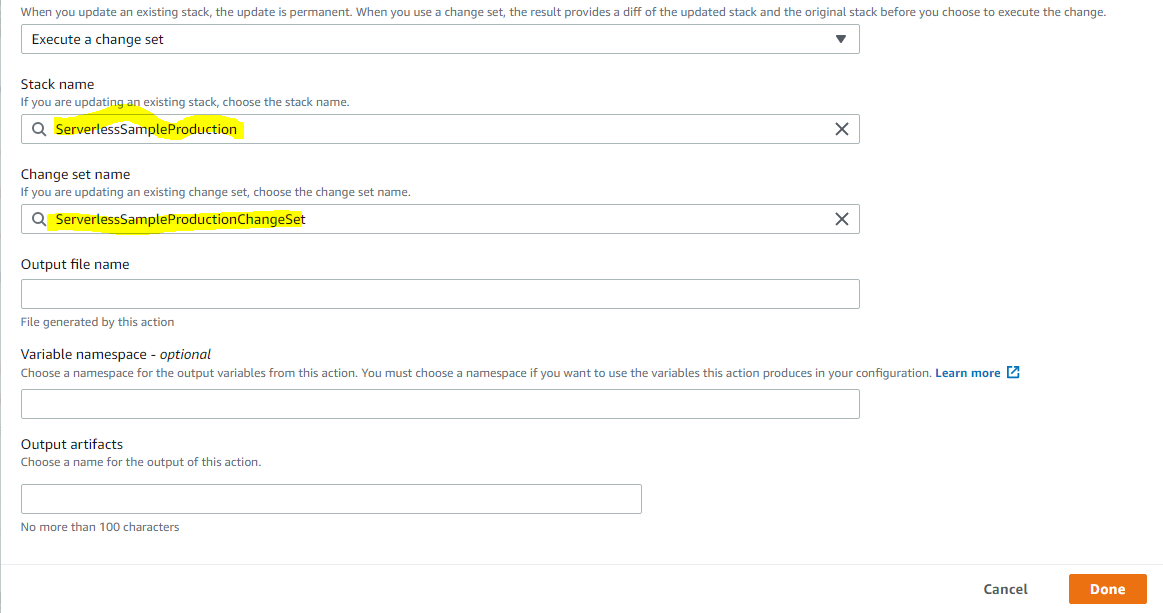
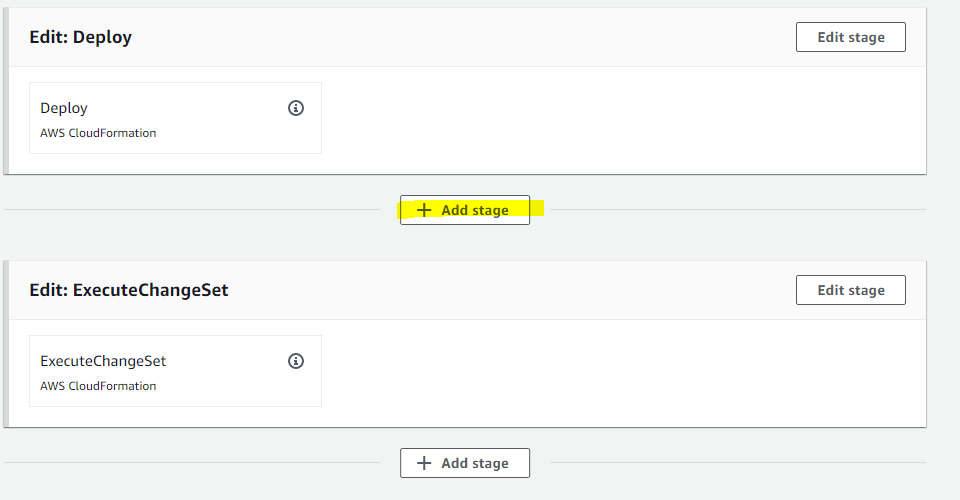
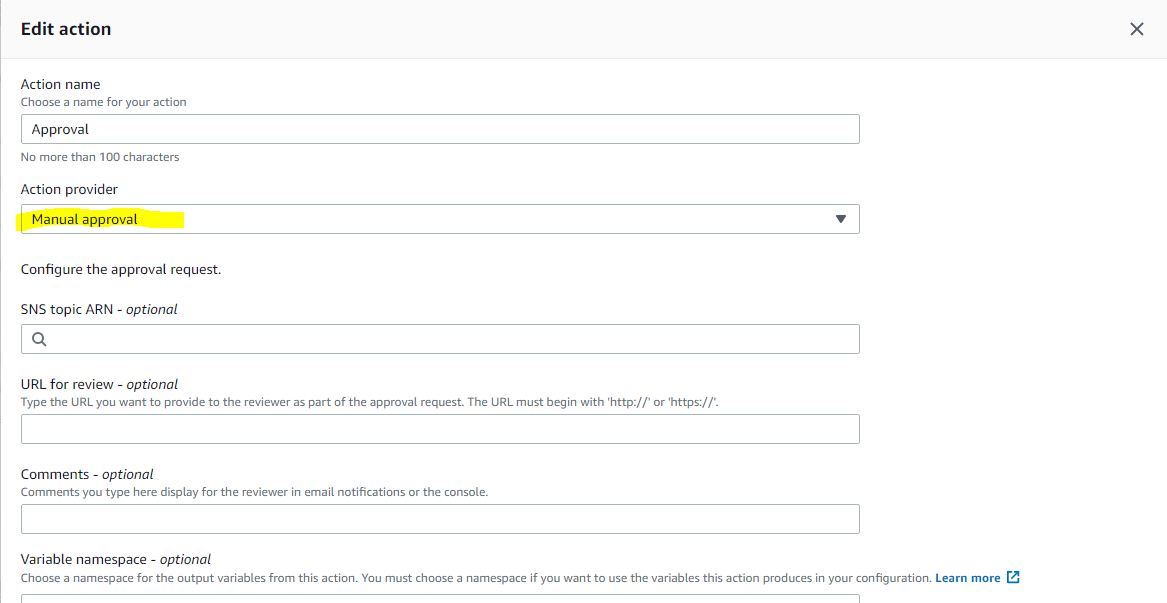


* add AWS codecommit as source and add the repository **stage** branch we’ve created at 2.Cloud9-CodeCommit-S3-Setup and Staging-Production env initiate.docx file



* Add CodeBuild project from suggestion created earlier “scheduledLambdaAppStaging” at the Build stage of the pipeline
* At the deployment section select the “Cloudformation” as deployment provider and select “Create or replace changeset” at action mode and select the “ServerlessSampleStaging” as stack name and give the changeset a name: 
* Select “CAPABILITY\_IAM” at Capabilities, give **Cloudformation IAM role**(created at step 1) at the Role name, give “packaged.yaml” at artifact file name and put “{"Environment":"Staging"}” at the advanced section and create the pipeline: 
* Now Edit the pipeline to add another stage: Give it a name like ExecuteChangeSet and click Add stage: Now add a action group to the new Stage: Give the action group a name and select **Cloudformation** as provider and select **Execute a changeset** for action mode and select **Buildartifact** for Input Artifact and select **ServerlessSampleStaging** as stack name like following: 
* Now save this settings and rerun this staging Pipeline to test it by click **Release Change** button. If the pipeline is successful the you have successfully release your changes to Staging environments. This pipeline is connected to “stage” branch of your source code and will execute automatically when change is merged/pushed into it!

For master branch:

* Clone the staging pipeline 
* Name it like **Scheduled-app-production**
* Edit the Source branch from **stage** to **master** 
* Edit the build stage to set **scheduledLambdaAppProduction** as project name 
* Edit the deploy stage to set the stackname as **ServerlessSampleProduction,** give the changeset name as **ServerlessSampleProductionChangeSet** and change the Parameter Overrides at Advanced section as “{"Environment":"Production"}” and save the changes
* Edit the **ExecuteChangeSet** action in **ExecuteChangeSet** stage, select **ServerlessSampleProduction** as stack name and **ServerlessSampleProductionChangeSet** as change set name and save the pipeline 
* Run the pipeline and check
* If everything ran successfully then we need to create a manual approval action for this pipeline before change set execution. To do it edit the pipeline and add a stage after **Deploy** 
* Name it **Approval** and add an action to it called **Approval**  and select Manual Approval at Action provider and hit done and save the pipeline. 
* Now run the pipline line again and this time at Approval stage the changeset execution need to be approved manually
* Now this pipeline will be triggerd when the **Master** code commit changes and if approved the changes will be released to Production stack