**You are now a part of the DevOps team in Logic Works Company, which is a leading AWS consulting and Managed Services Provider with 25 years of experience in enterprise IT. Your first project is to replicate the current environment on AWS as per the following requirements:**

1. Adoption of IAC (Infrastructure as Code) to facilitate faster replication of environments for different customers who may have a need for similar environments.

2. Infrastructure must include multi-region architecture and needs to run in two different regions for High Availability and Disaster Recovery purposes. (Hint: ensure proper networking system)

3. The applications should be run in a container to prevent errors and inconsistencies. The container images are supposed to be stored in a Docker image repository. Additionally, a container management system has to be implemented to manage growth in the number of containers in the future.

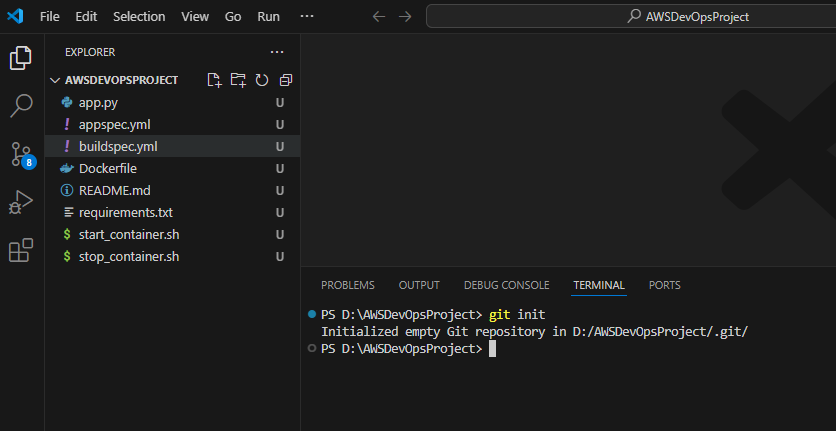
4. Manual CI/CD pipeline deployments did not fulfill the requirements. Hence there is a need to automate the CI/CD pipeline. The CI/CD pipeline should include source, build, stage, manual approval and production environment.

5. The application will be running in two different regions, so the source code repository needs to be replicated in the other region as well, in order to pull the code locally within the regions and avoid any latency issues.

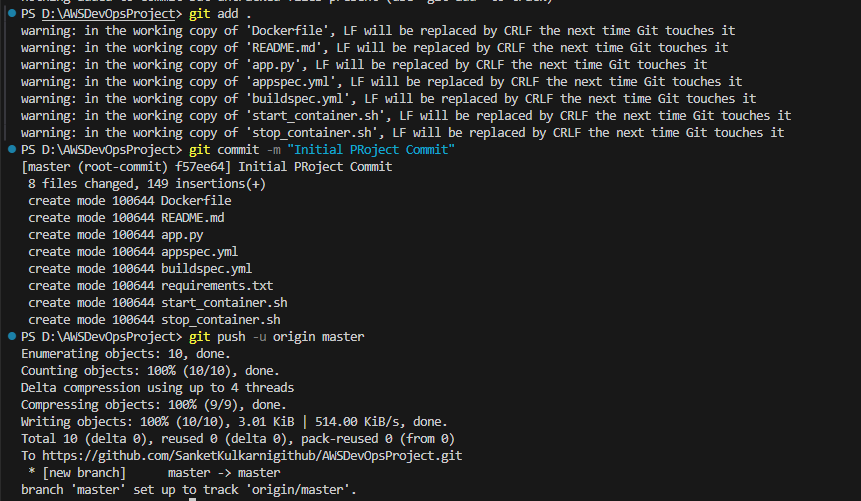
6. The infrastructure and the application need to be under continuous monitoring with notification mechanism.

7. It is necessary to have an approved CI/CD pipeline before the code gets pushed to production.

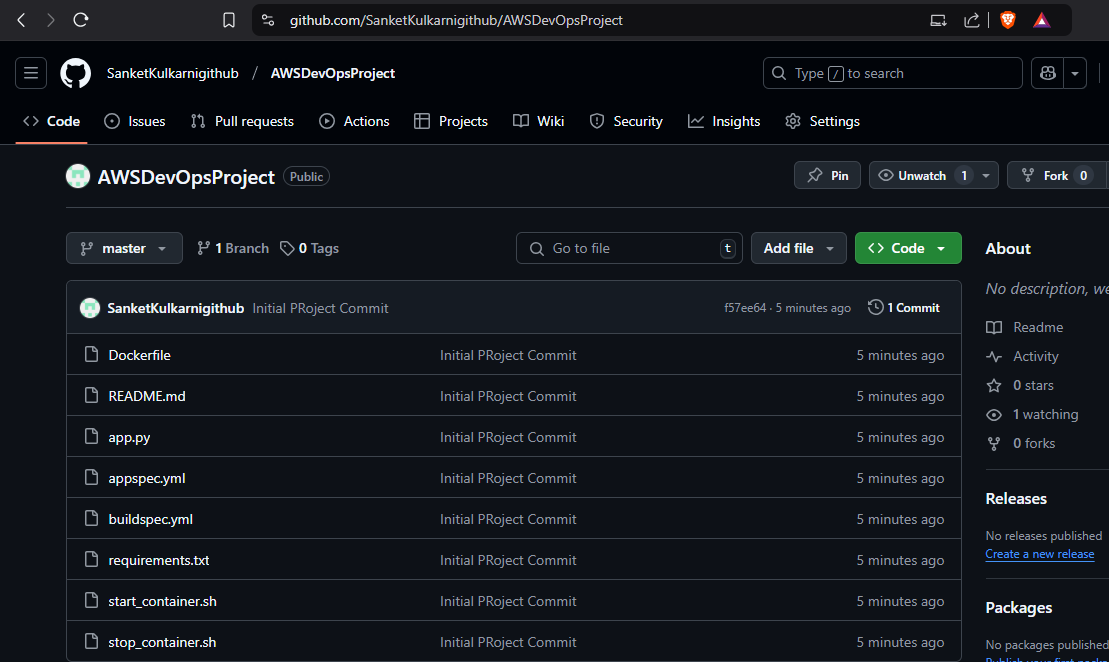
Create Project and initialize Git Repository

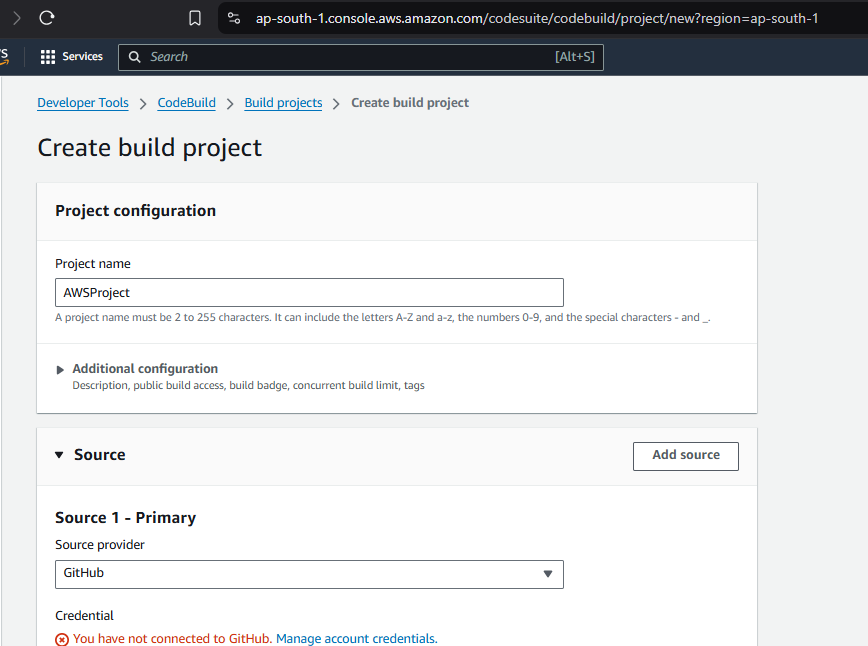


Commit code and push to GitHub repository created for this project

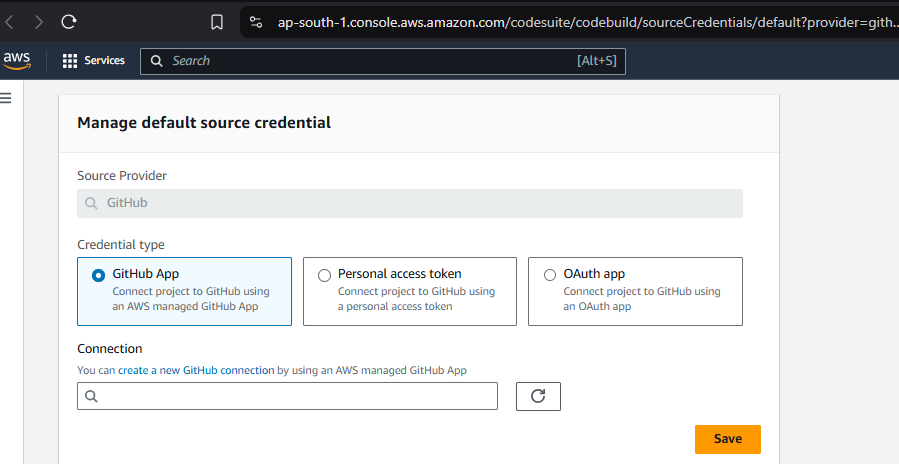


Github portal after code pushed from local repository

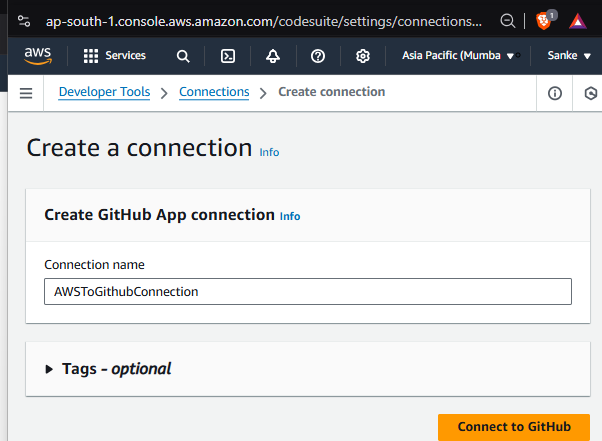


Go to AWS Portal -> Search Code Build -> Click on Create build project -> Enter Project Name -> Select Github as source 

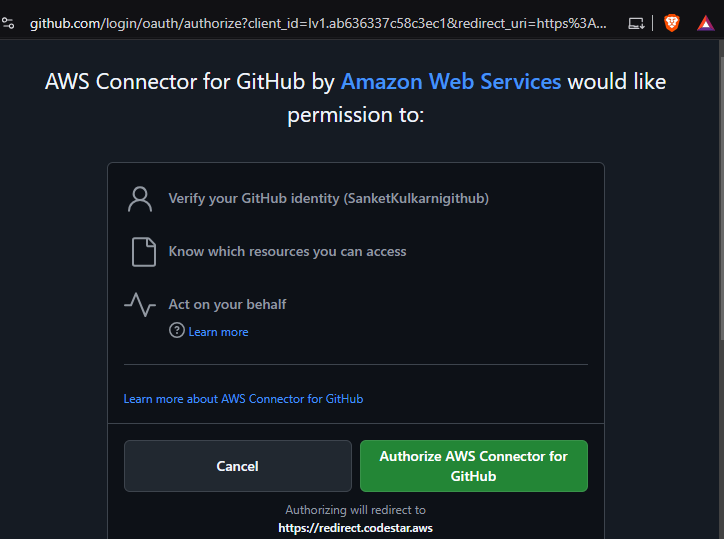
Manage account credentials of GitHub if not done already

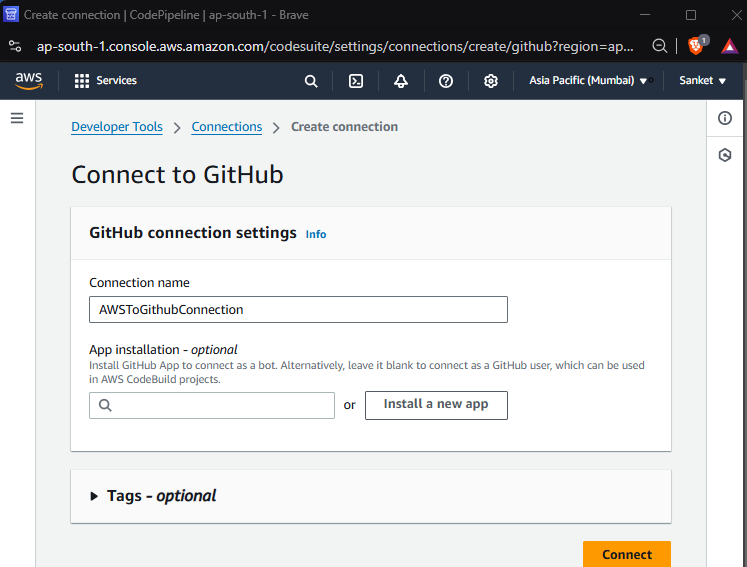


Create a new GitHub connection if one already not exist as shown below -> Click on Connect to GitHub

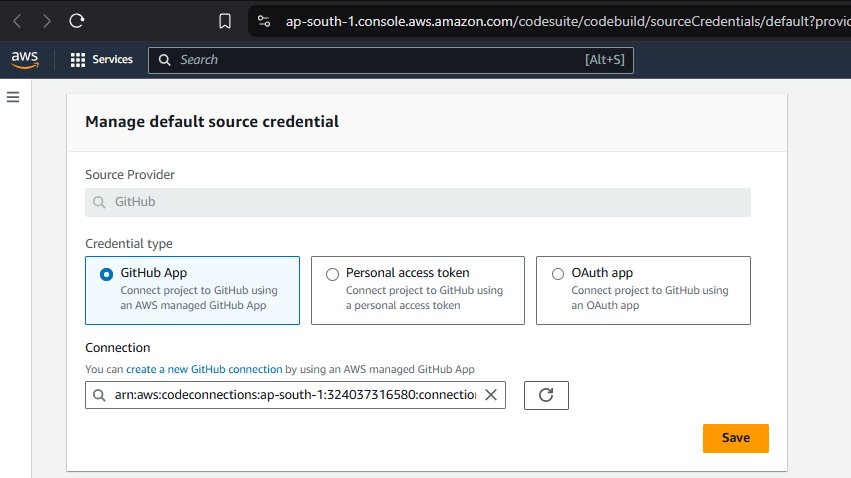


Click on Authorize AWS Connector for GitHub

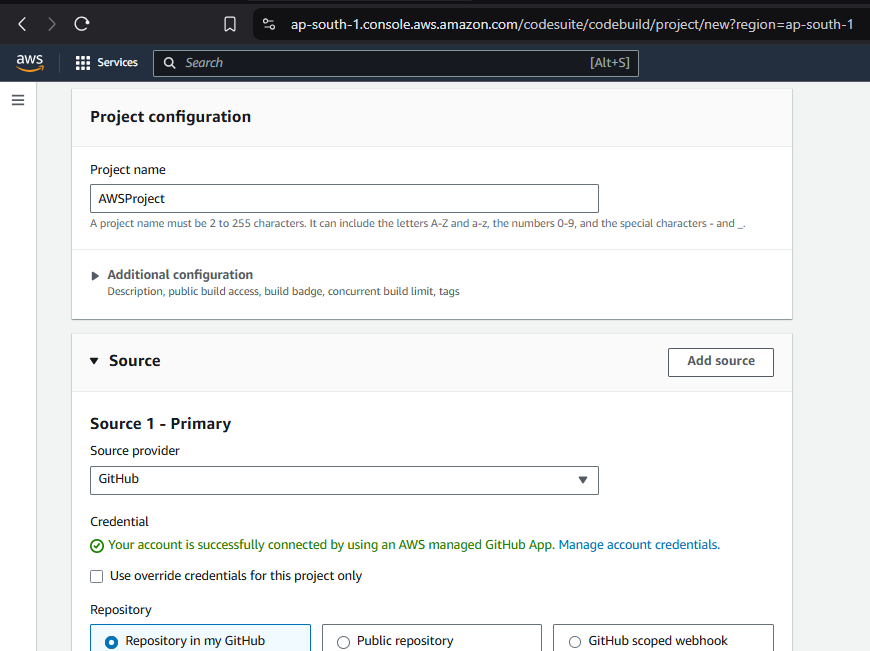


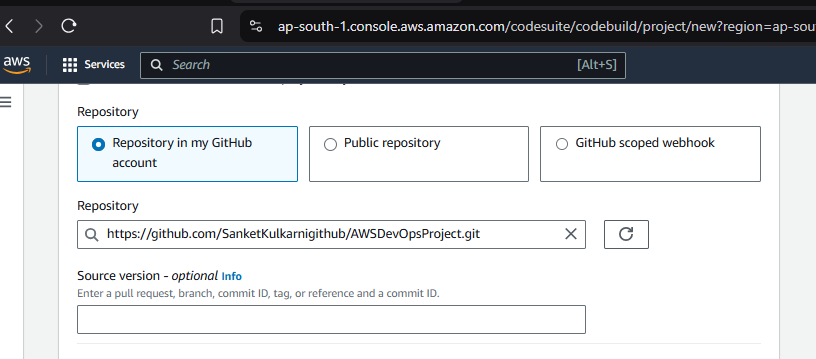
After authorization below page will open -> Click on Connect

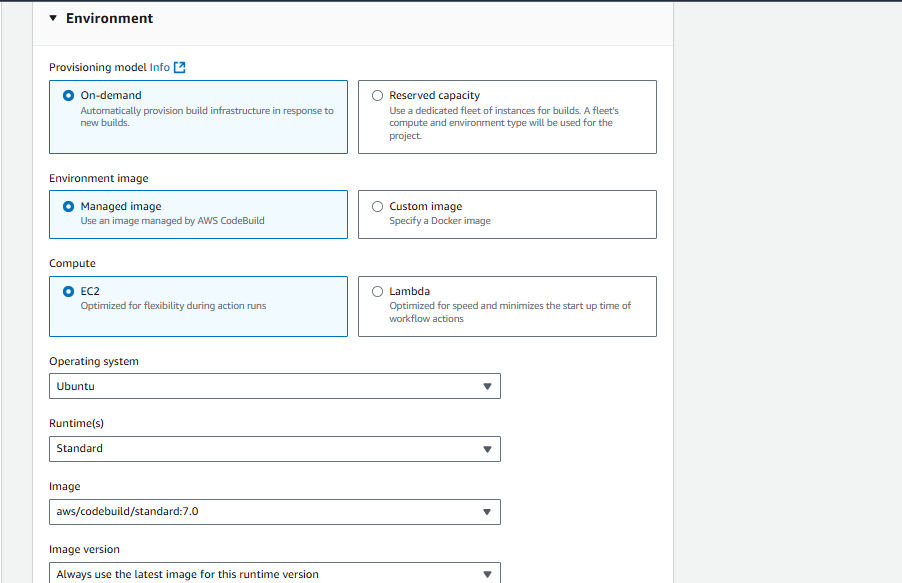
Go to Earlier page below -> Select the connection (which we created) -> Click on Save

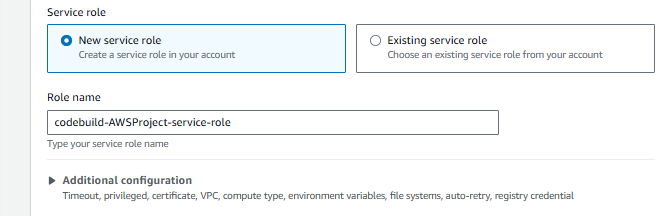


Not this is showing as Account is connected to GitHub

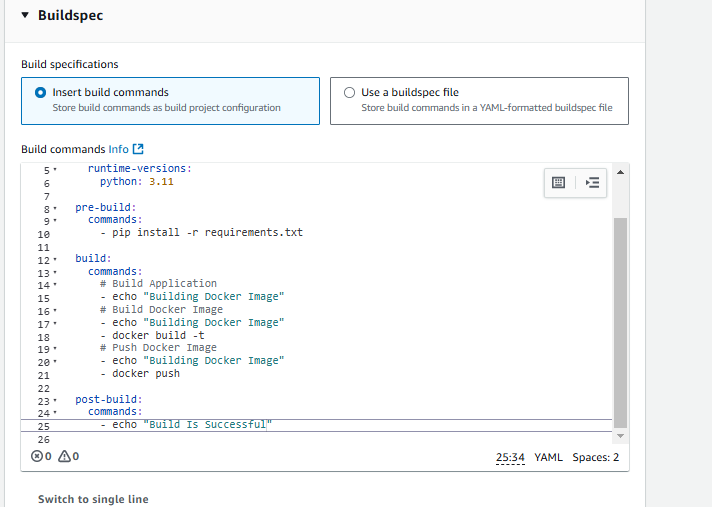


Enter the repository URL now

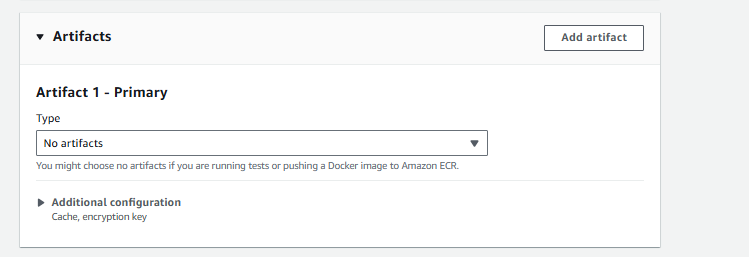
In Environment Section of same Page select values as below”  


Select IAM Role to be used by Code Build

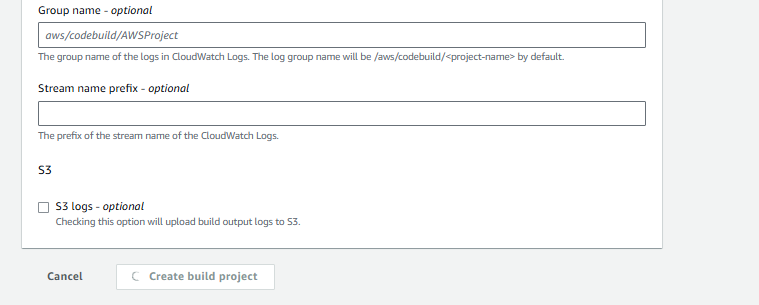
In Build Spec Section – Select Insert build commands -> Write the commands as shown below Need to modify later

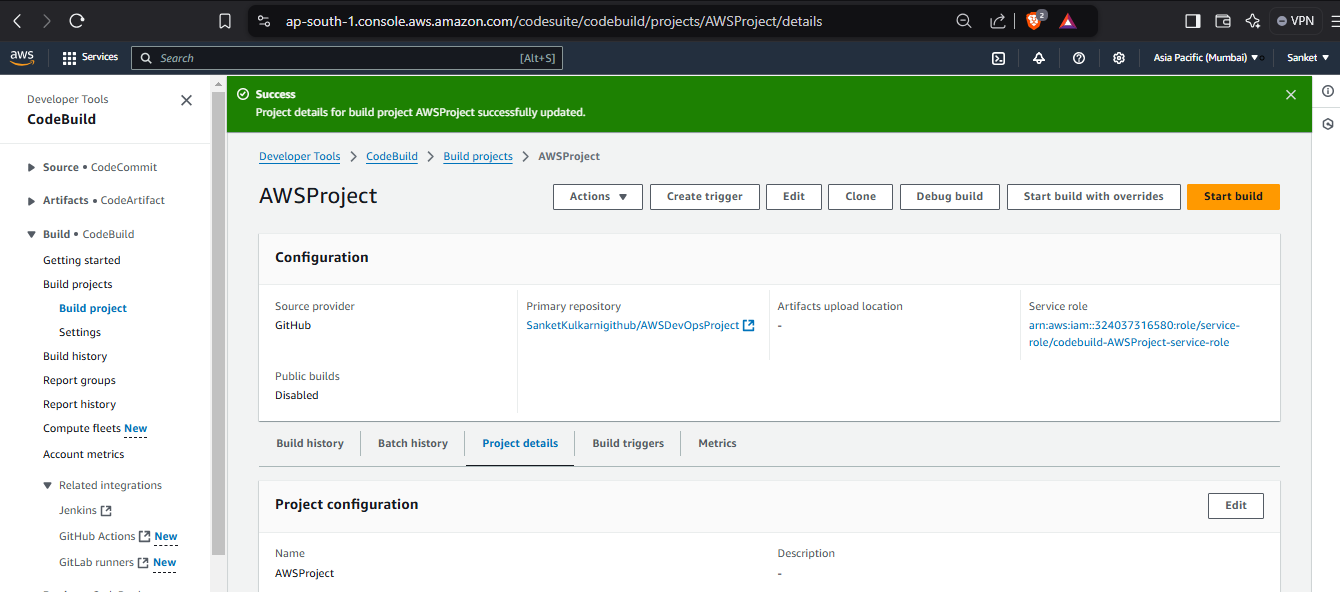


Artifact

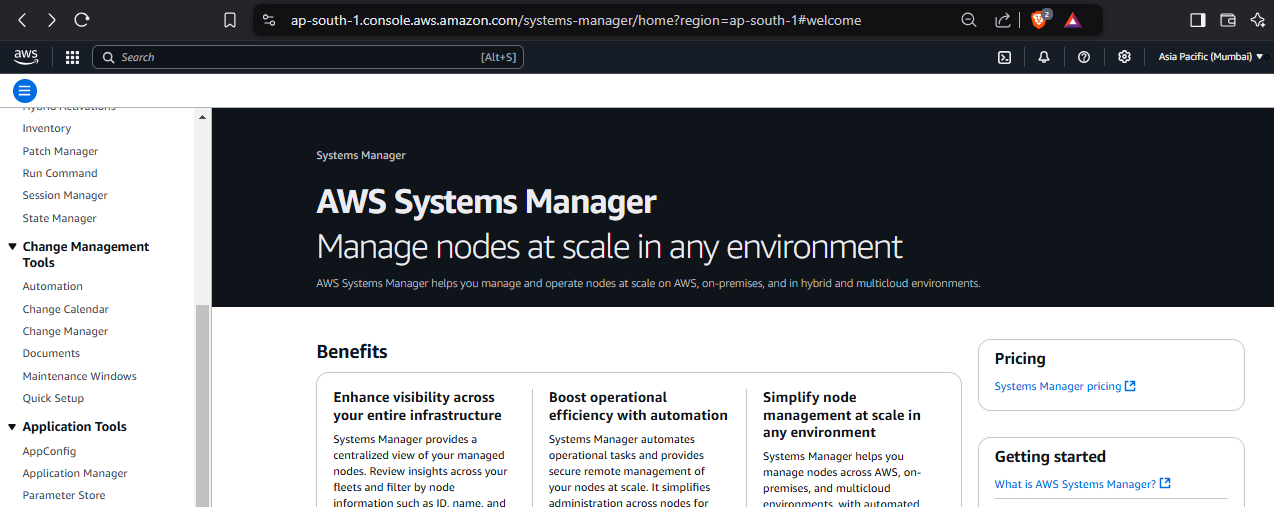


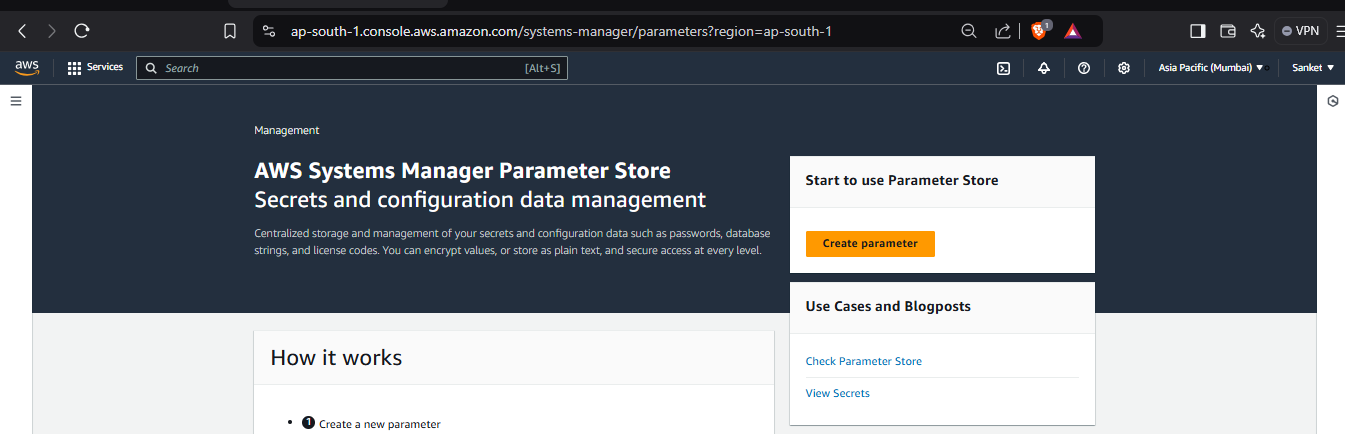
Create Build Project

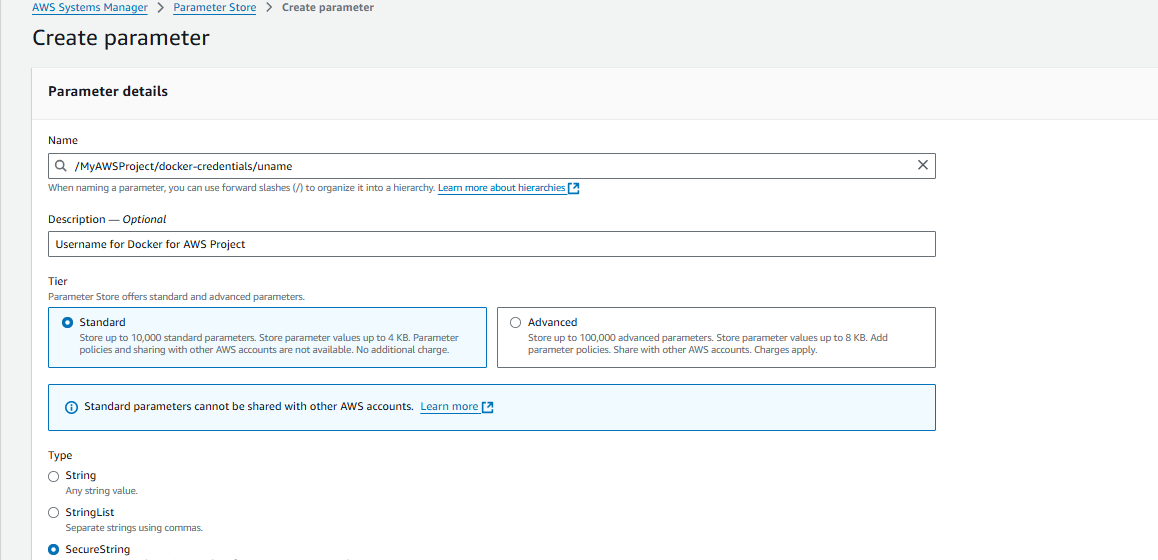


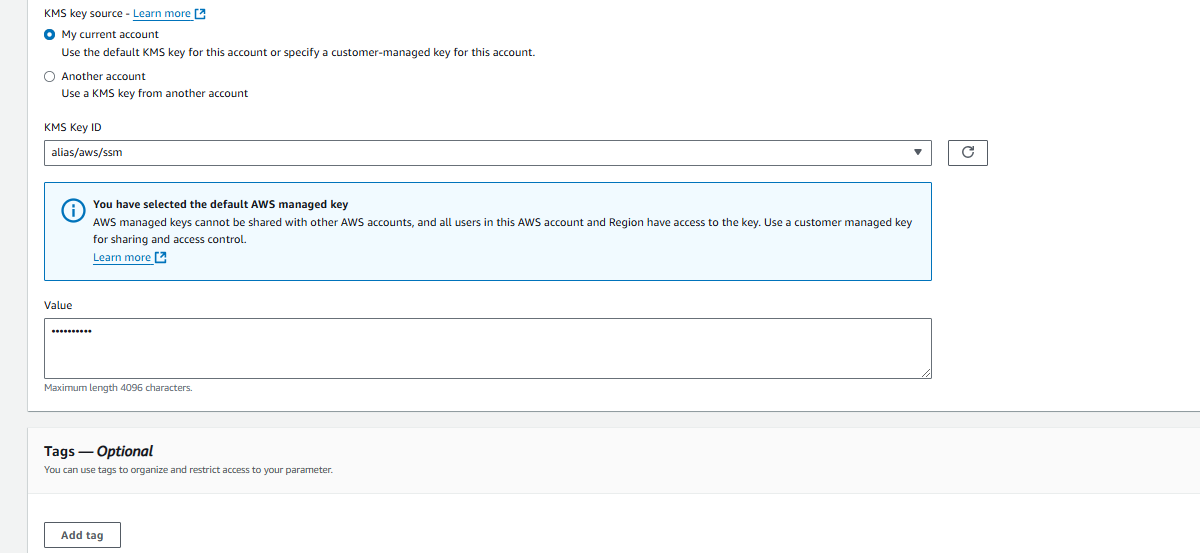


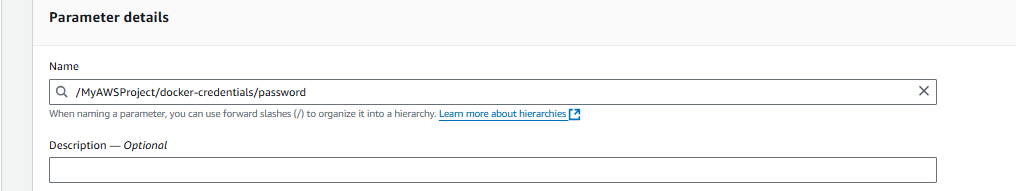
Open AWS Systems Manager -> Left hand side, select Parameter Store under Application Tools

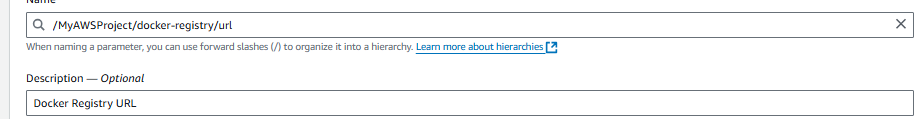


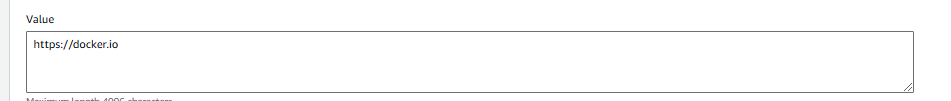
Parameter Store

Click on Create Parameter ->Enter Name, Description, Tier, Type, Valie as shown and click on Create Parameter

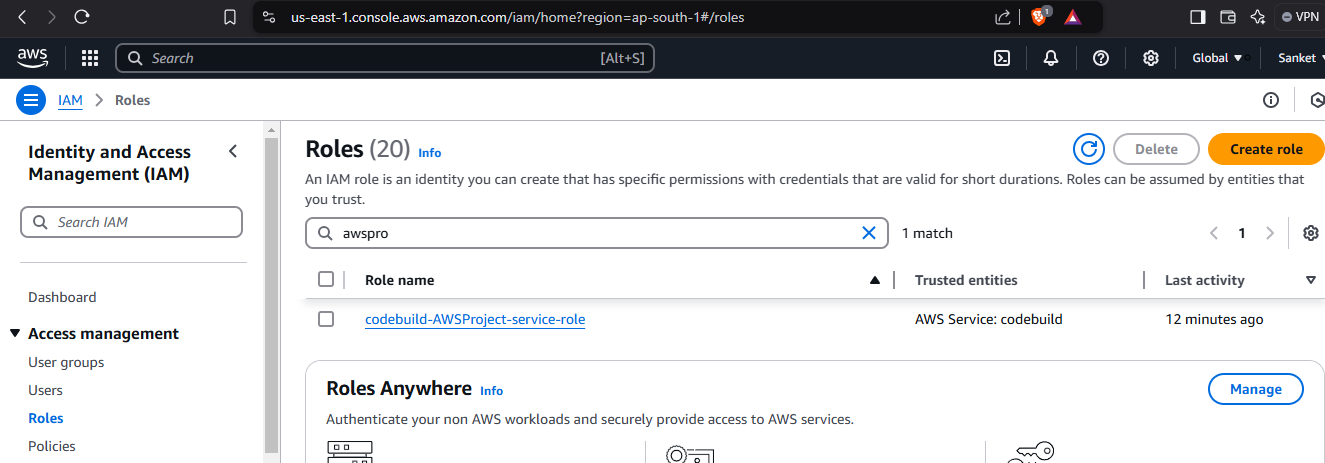


Same way, create for Password and Docker URL

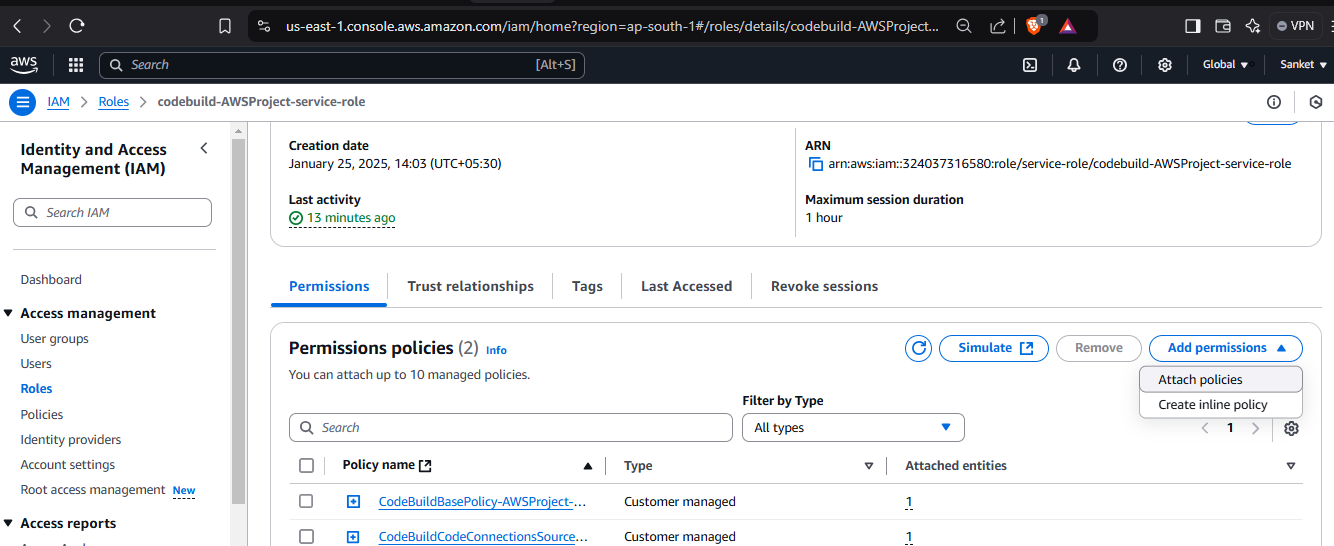




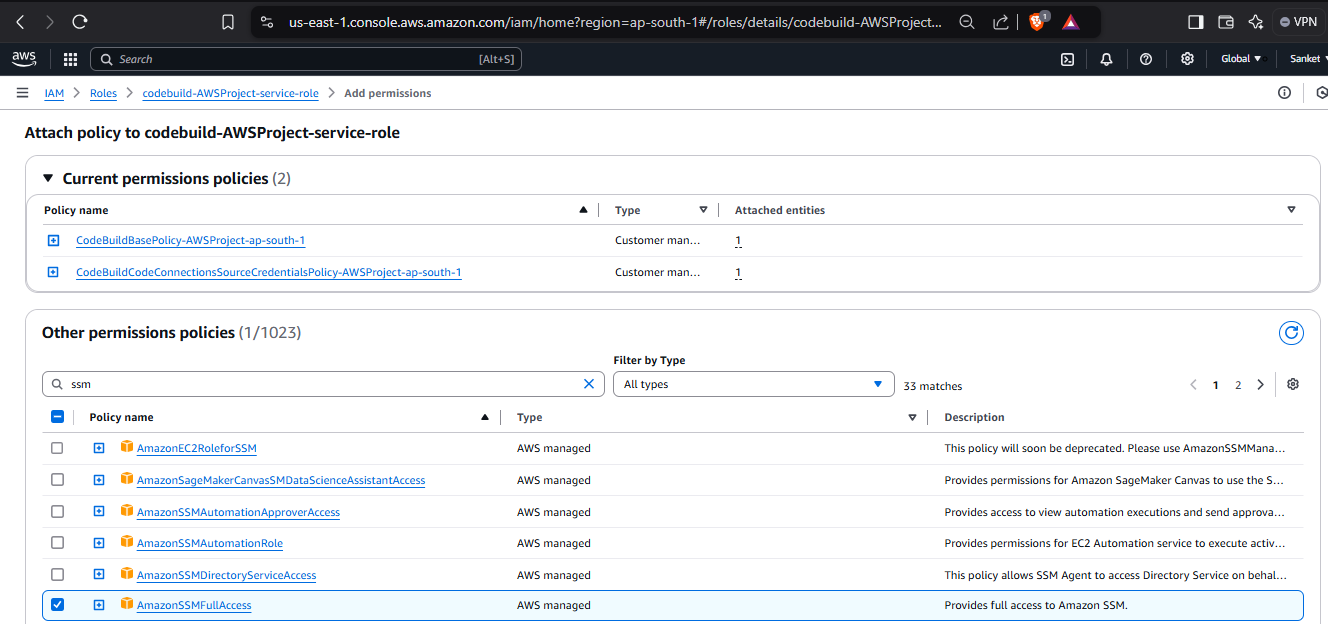
Search IAM Service -> On Right Side -> Access Management -> Click on Roles -> and Search and click our role (created earlier)

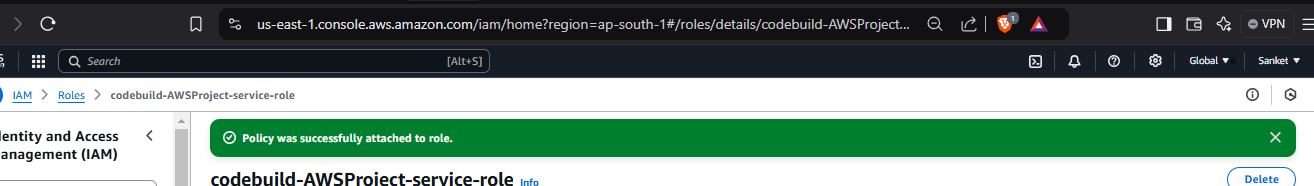


Click on Add Permissions -> Attach Policies

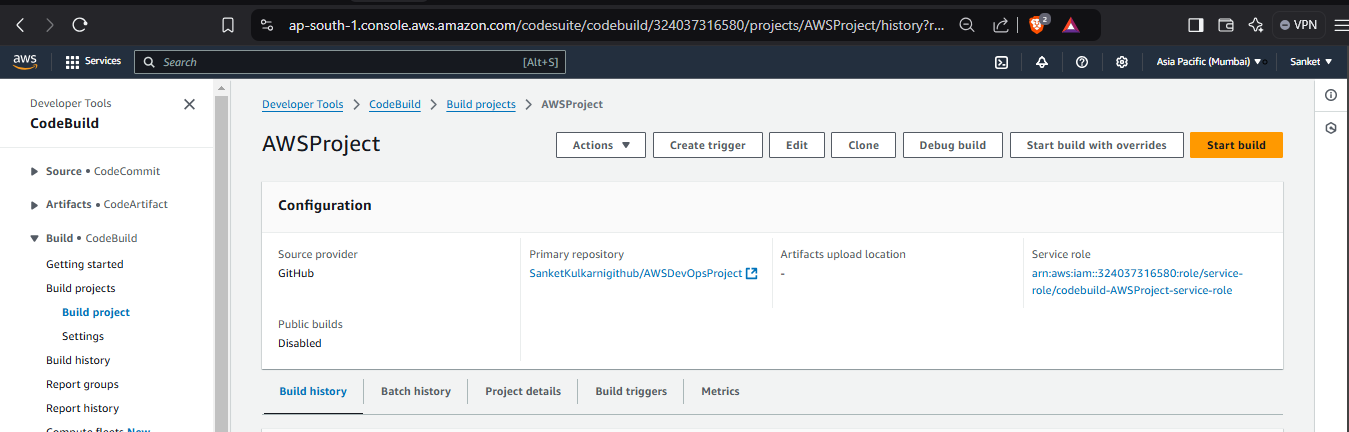


Search ‘ssm’ in policy (policy for system manager) -> Select the Full access and click on Grant Permission

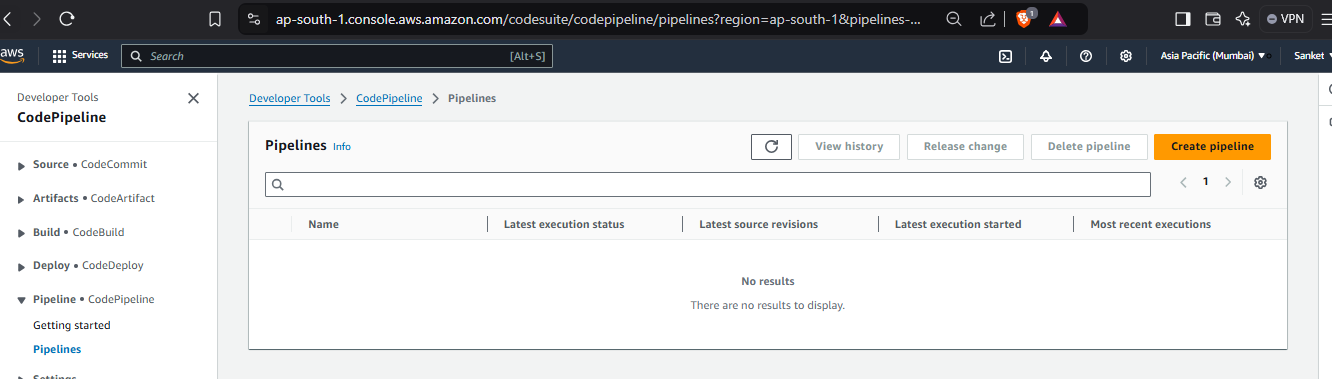


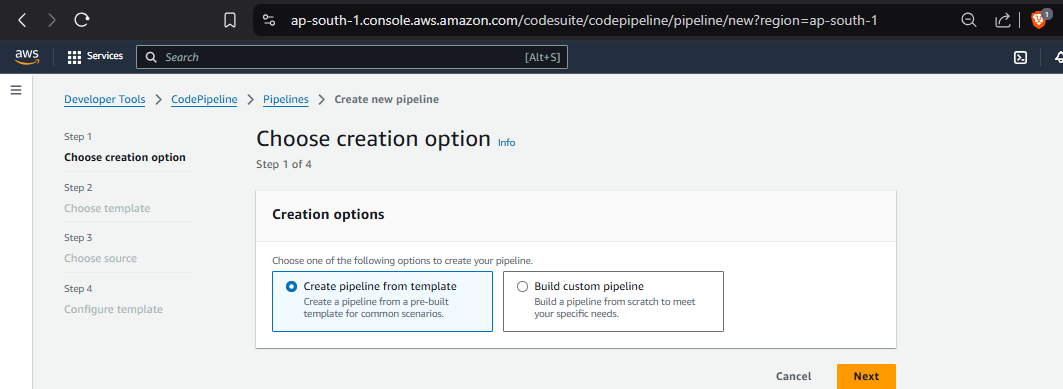


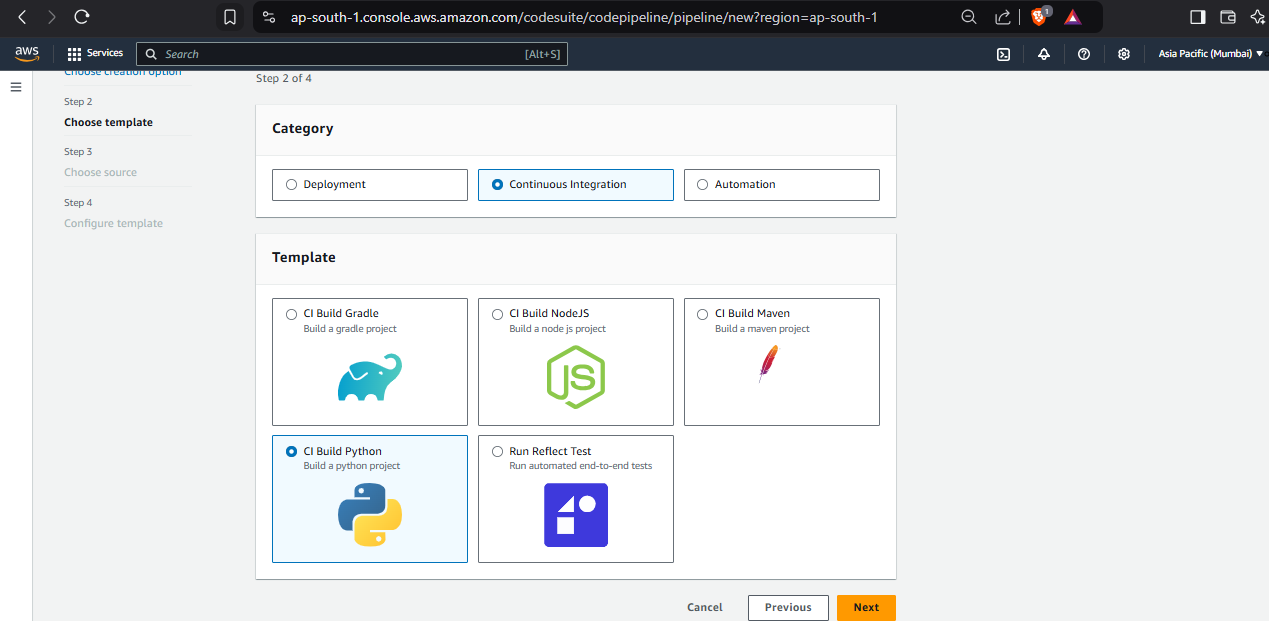
Go to our Code Build created earlier and click on Start Build

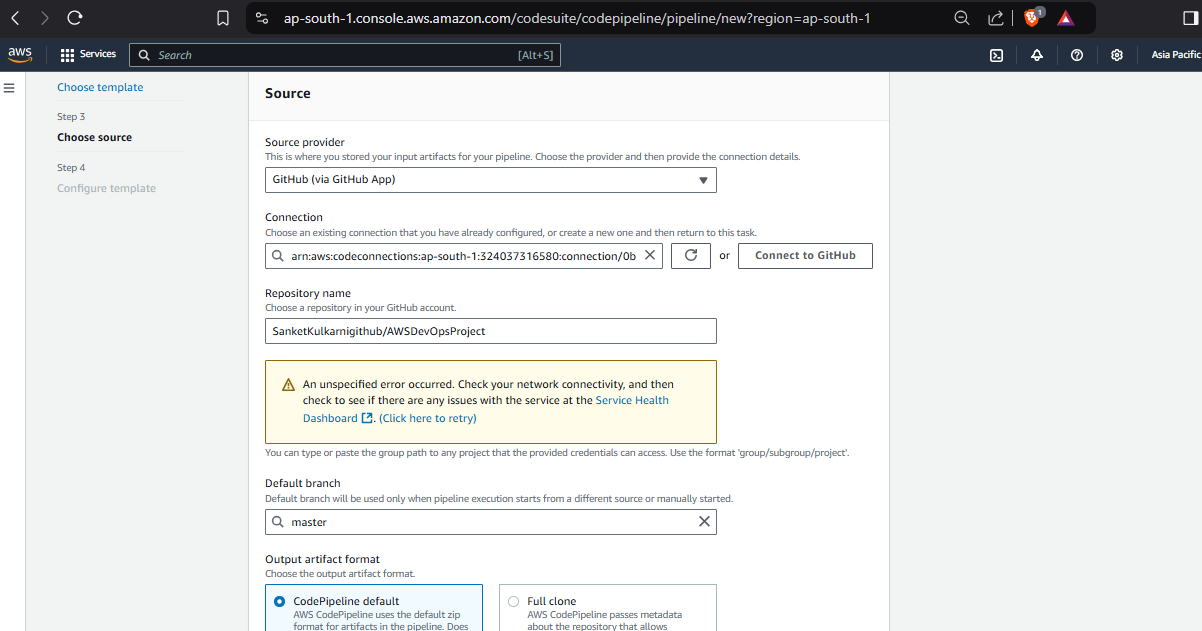


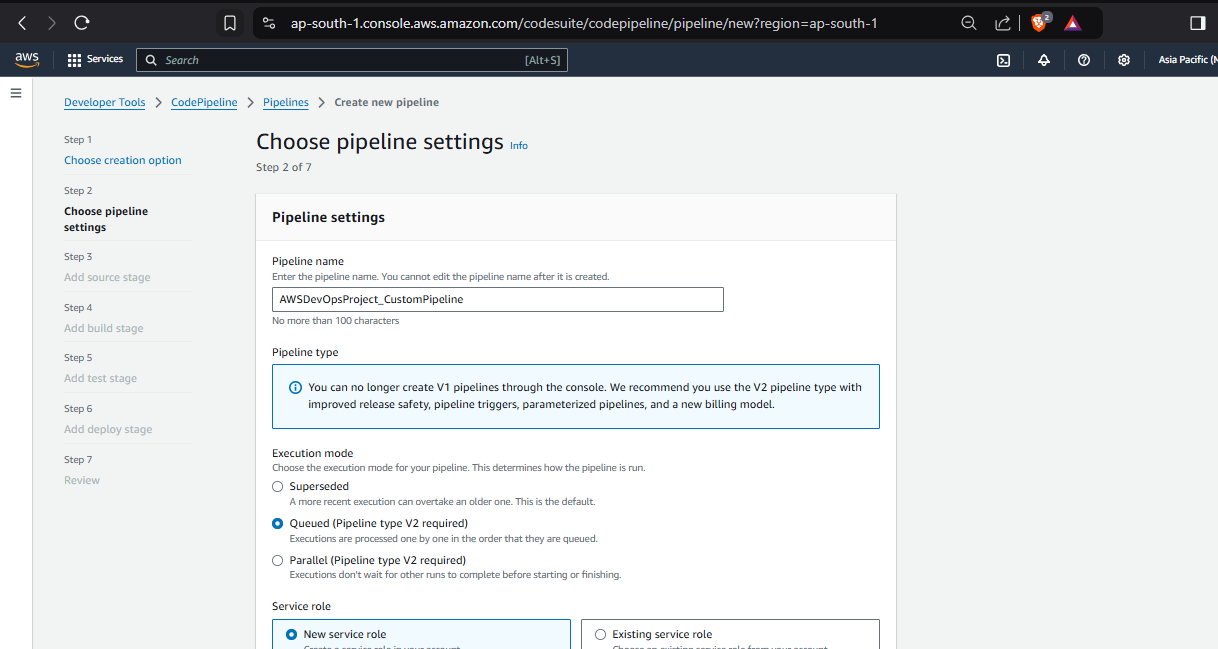
Search CodePipeline from Home page -> Click on Create Pipeline

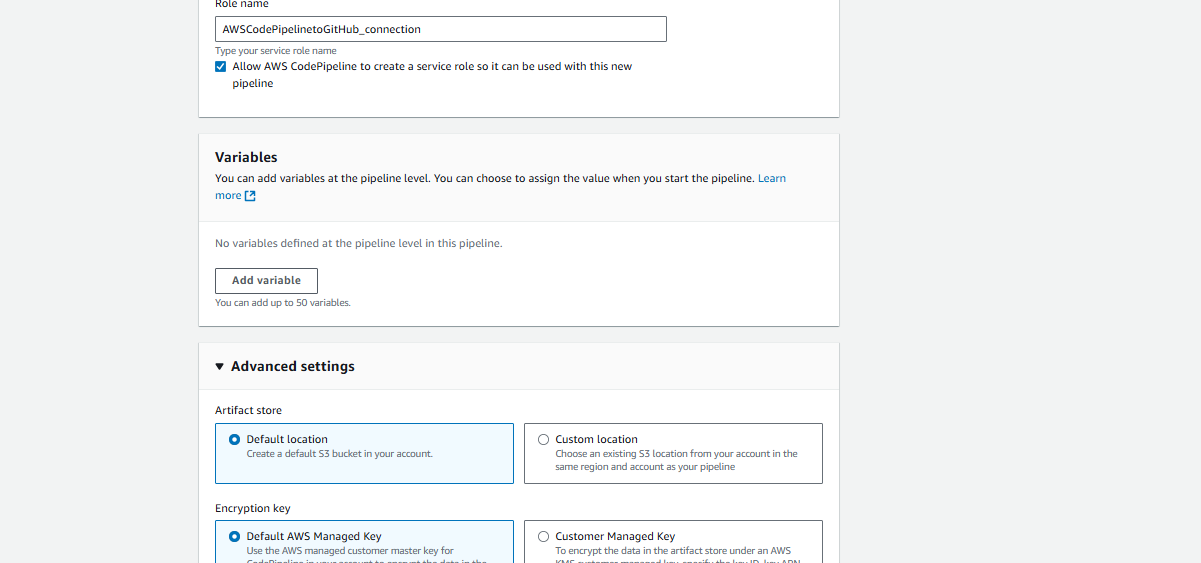


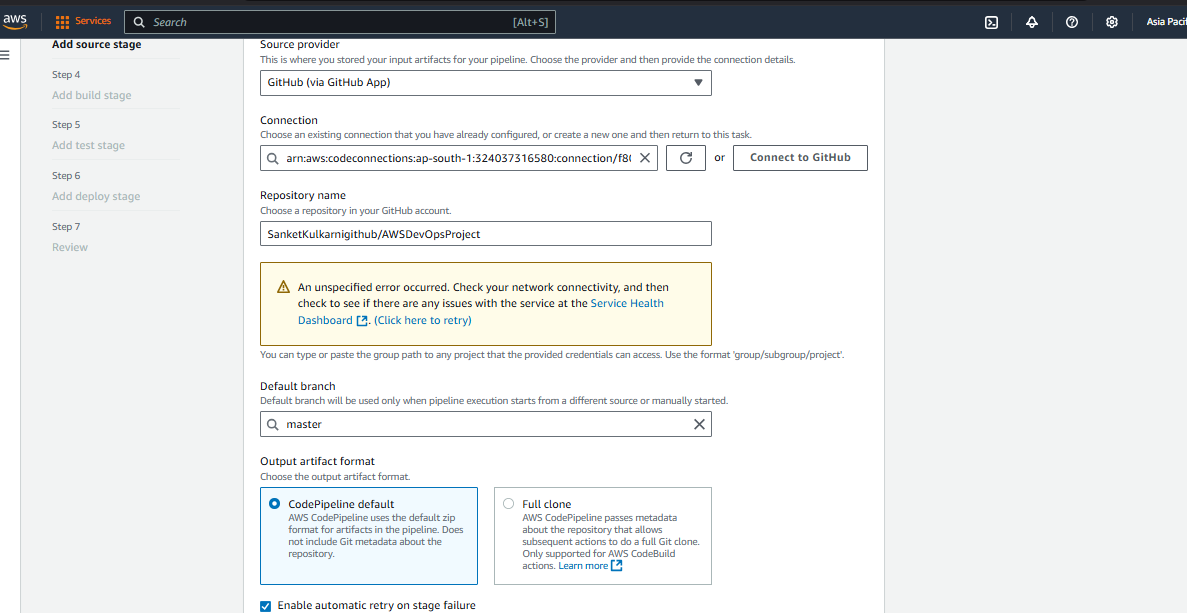


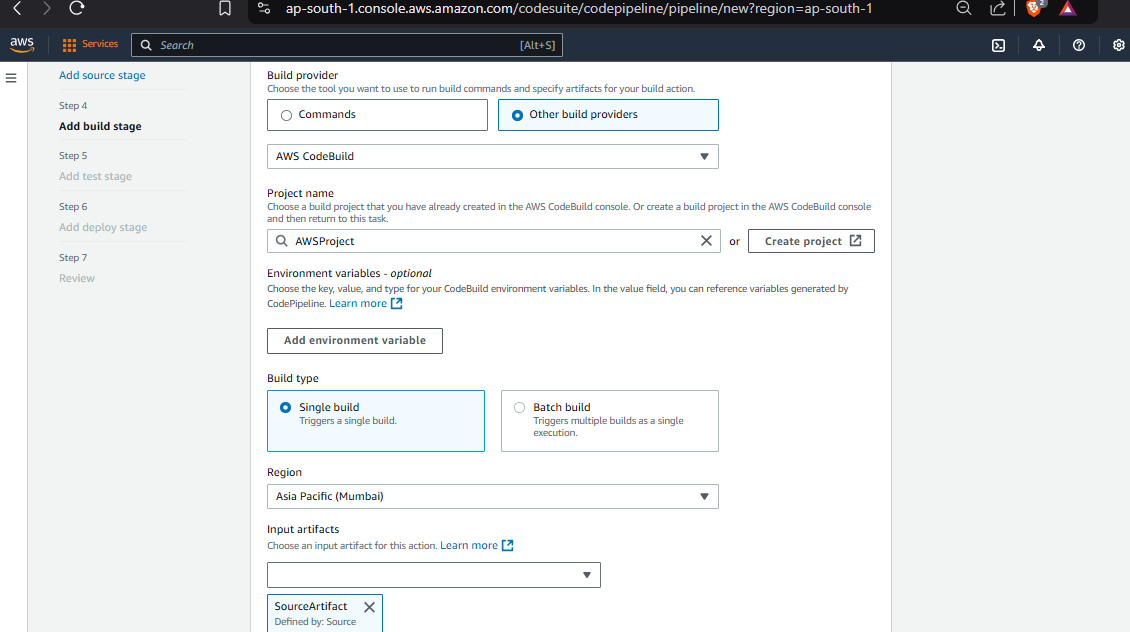


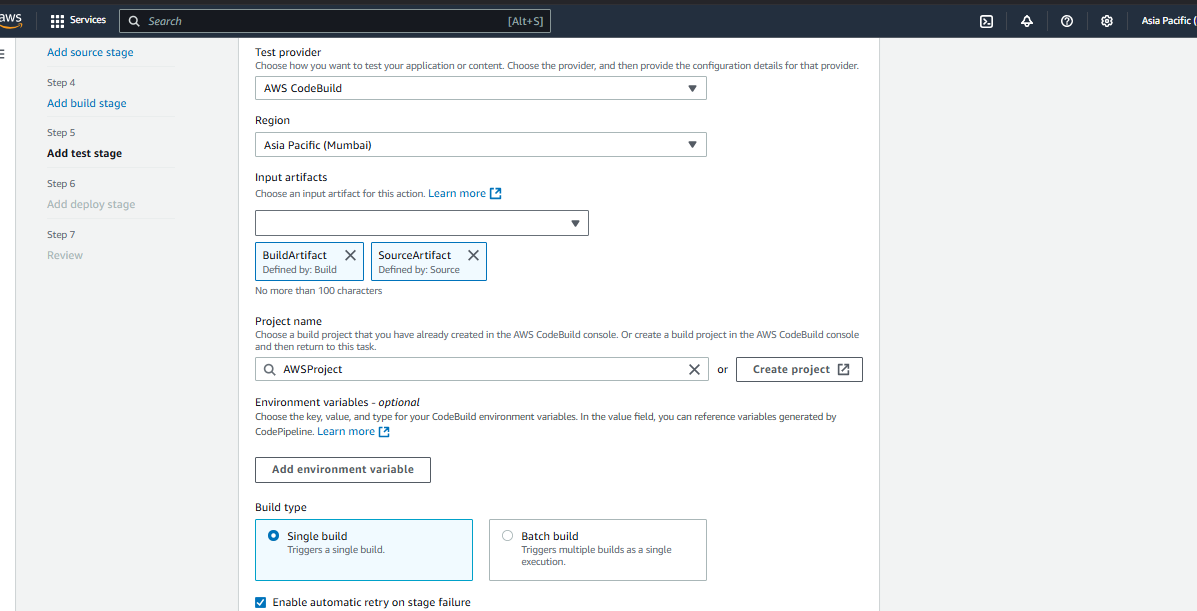




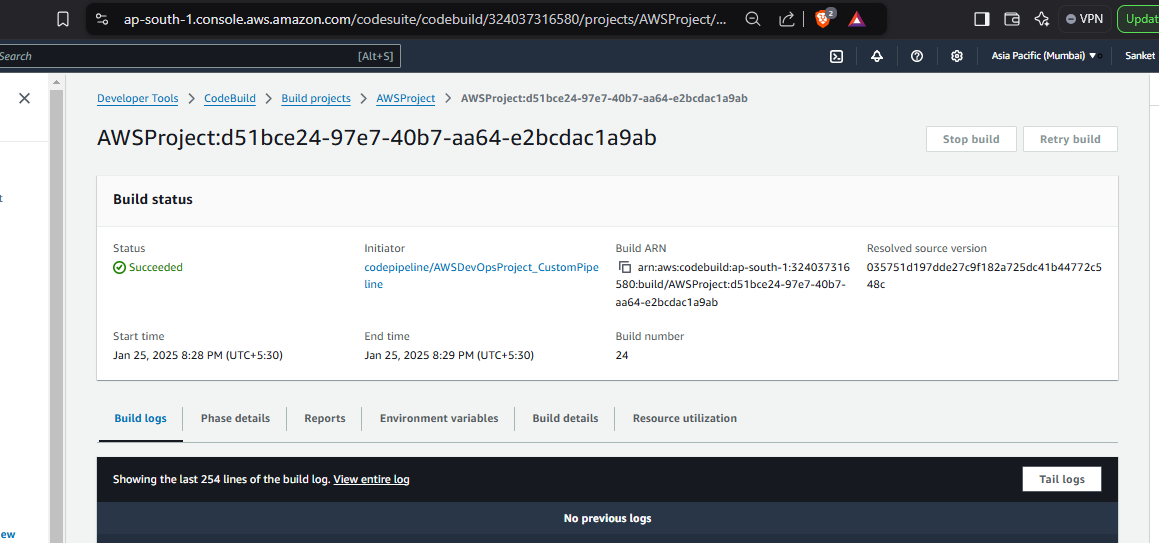


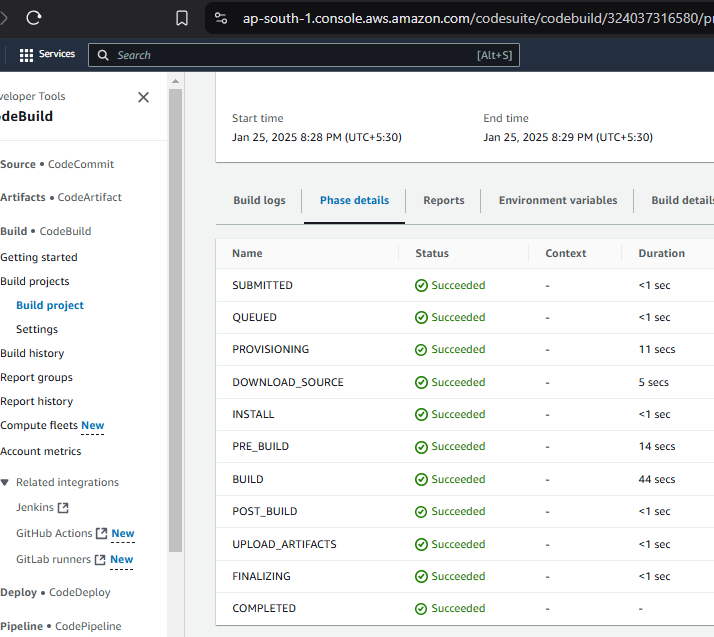
Source Stage ->



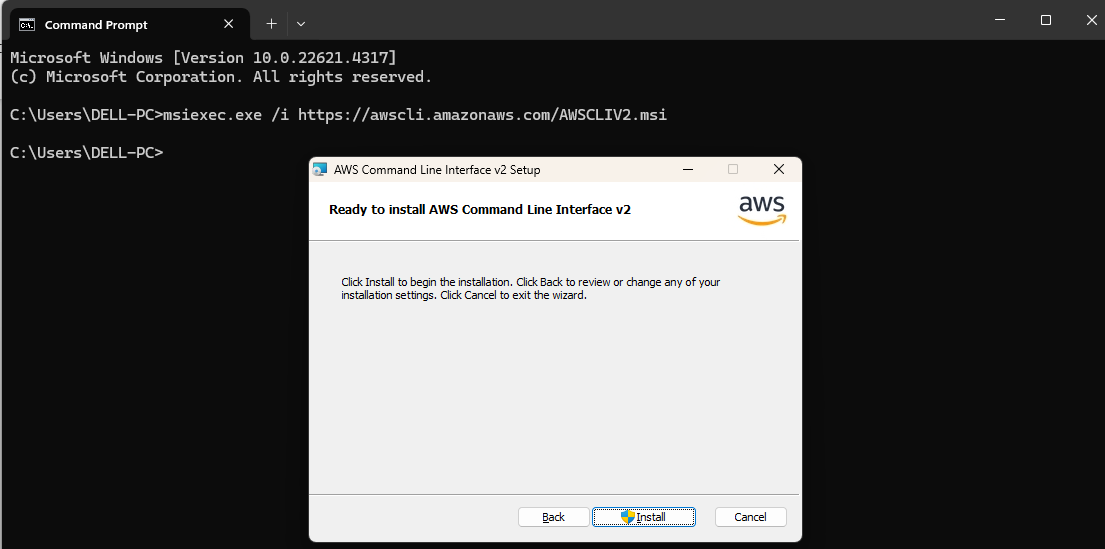


Execute build -> Build is successful





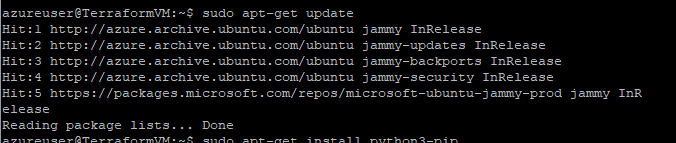
Install AWS CLI – Run command in CMD and a popo up will come -> Click on Install

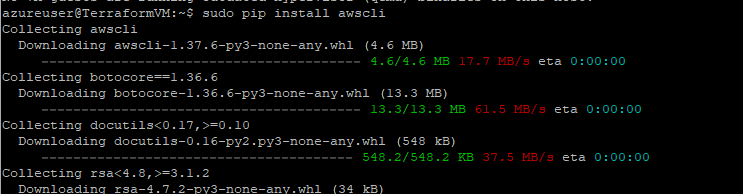


Close CMD and open again to check the installation is successful or not version



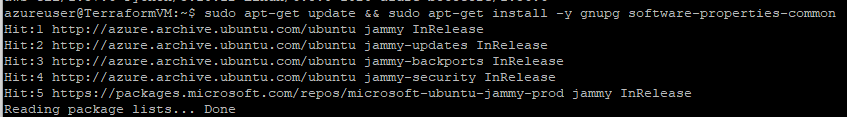
Terraform Installation is not working on desktop. So, let’s create VM and install AWS CLI and Terraform there

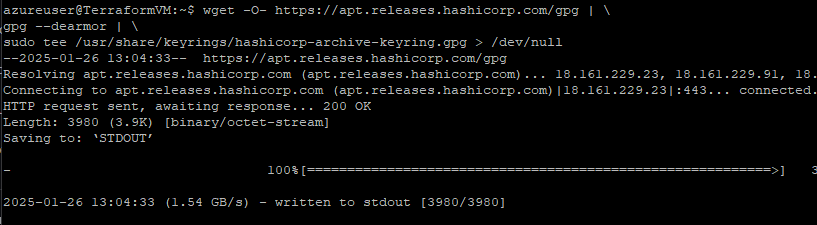


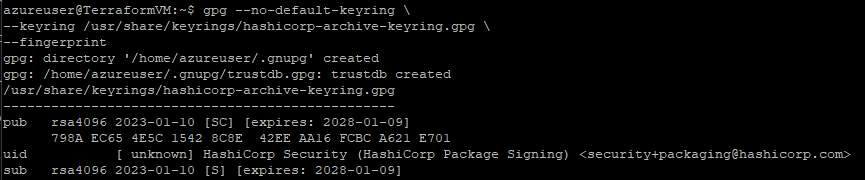


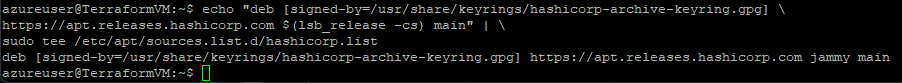


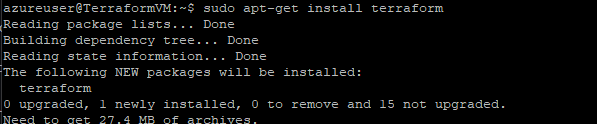
Terraform Installation -



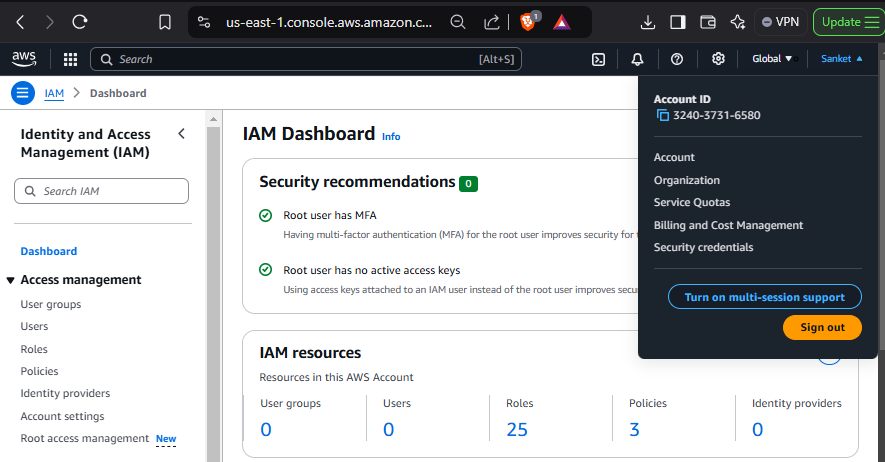




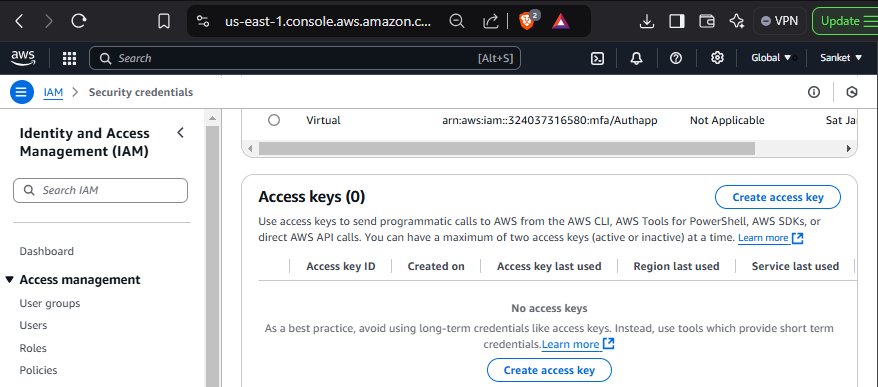




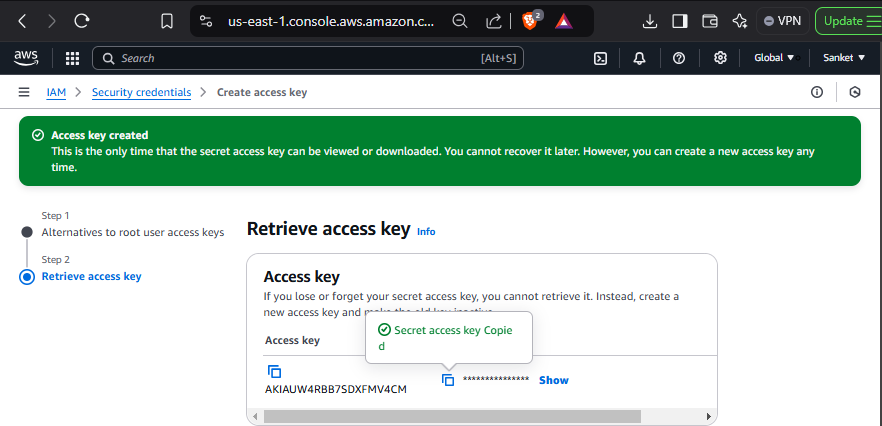
Create Access Key to connect AWS from Terraform

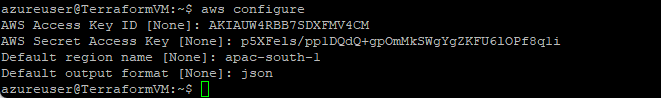
Click on Name at the right top corner -> Click on Security Credentials

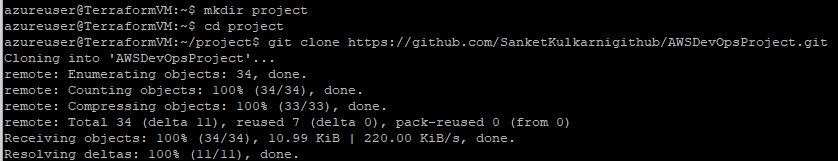
Click on Create Access Key

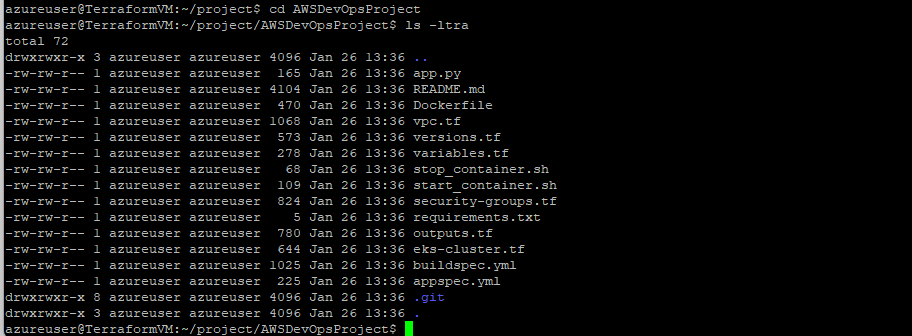


Copy the secret access key

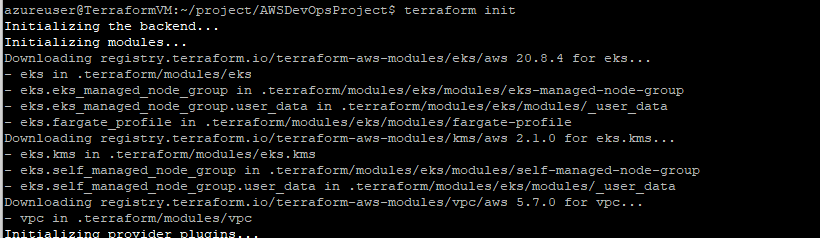


Go to VM again and run aws configure command and enter above copied access key id and secret key and other details

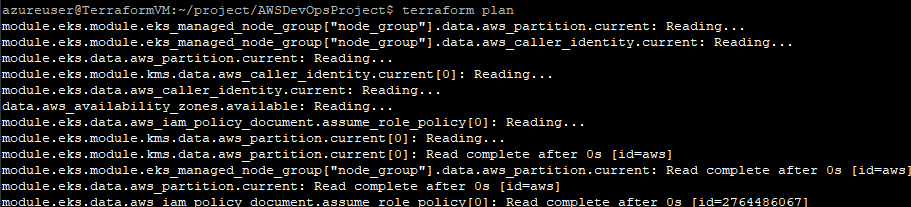
Clone the repository on VM now



Execute command terraform init

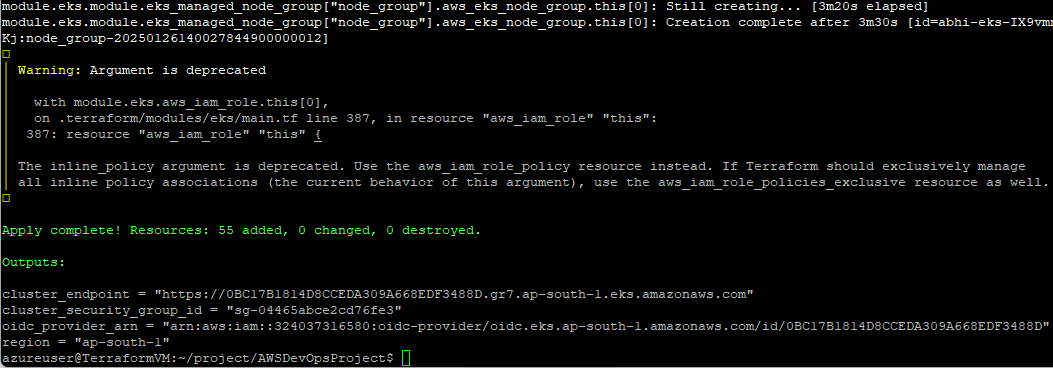


Execute command terraform plan

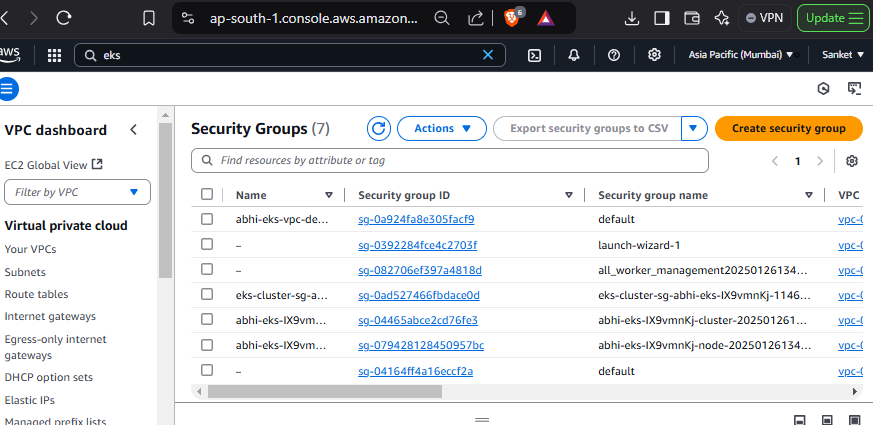


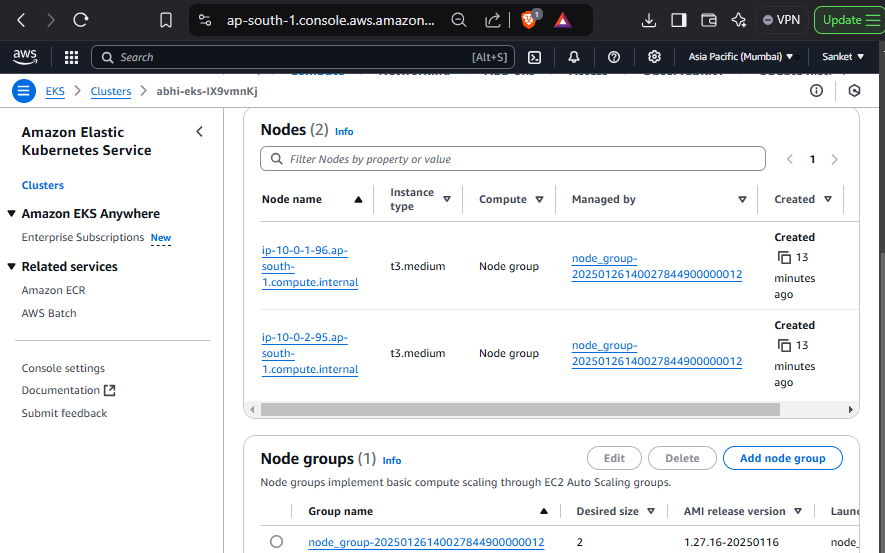
Execute command terraform apply –auto-approve





Various Components created through Terraform are reflecting in AWS Portal

****



Facing error to go ahead with deployments on AWS EKS ☹

