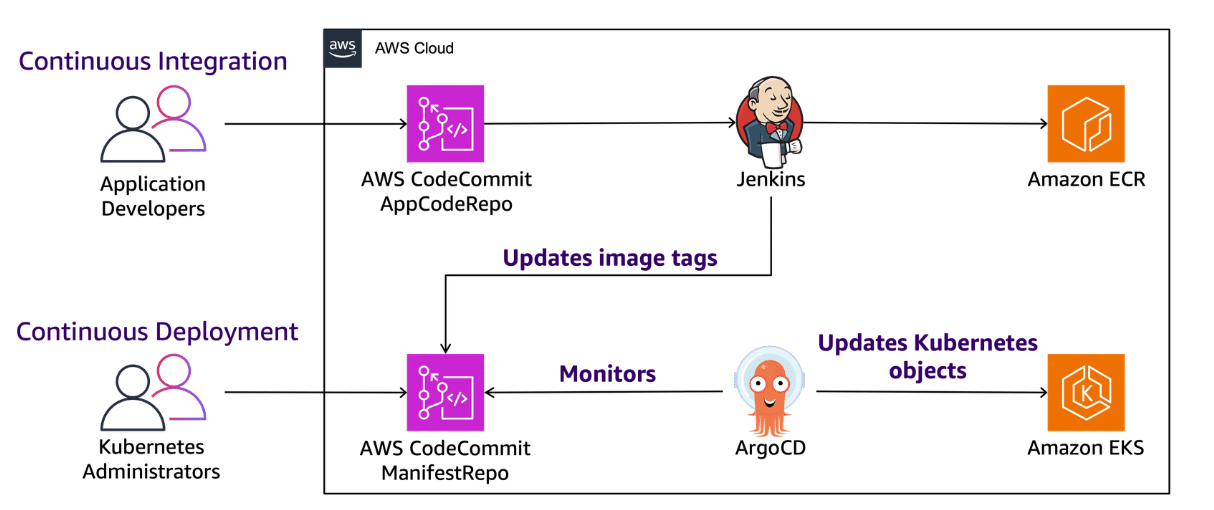
# EKS Lab-3: Continuous Deployment and GitOps

Objectives:

1. Create and use an AWS CodeCommit repository to store application code and Kubernetes manifest files according to GitOps best practices.
2. Configure a continuous integration pipeline with Jenkins.
3. Deploy Argo CD continuous delivery tool for Kubernetes.
4. Configure a continuous deployment pipeline with Argo CD.



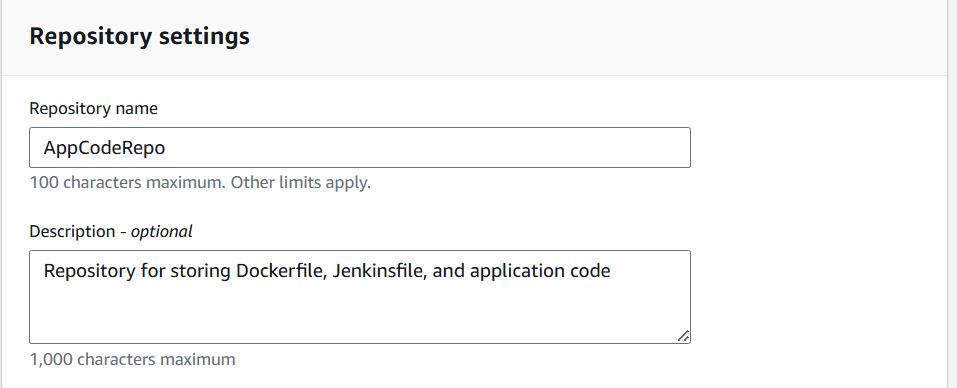
1. Devs put code in codecommit
2. Jenkins takes this code and turns it into container images, and stores these images in Amazon ECR.
3. ArgoCD takes these container images from ECR (using code commit) and deploys it on EKS

**Task 1: Create CodeCommit repositories**

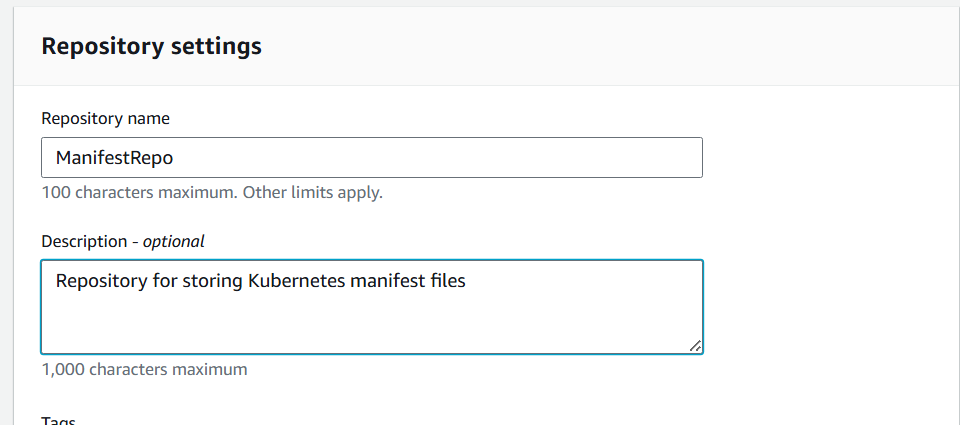
1.1 Open CodeCommit, and make two repos

1. The first repository stores the Dockerfile, Jenkinsfile, and your application code.
2. The second repository stores the Kubernetes manifest files that Argo deploys to your Amazon EKS cluster.

Repo 1-



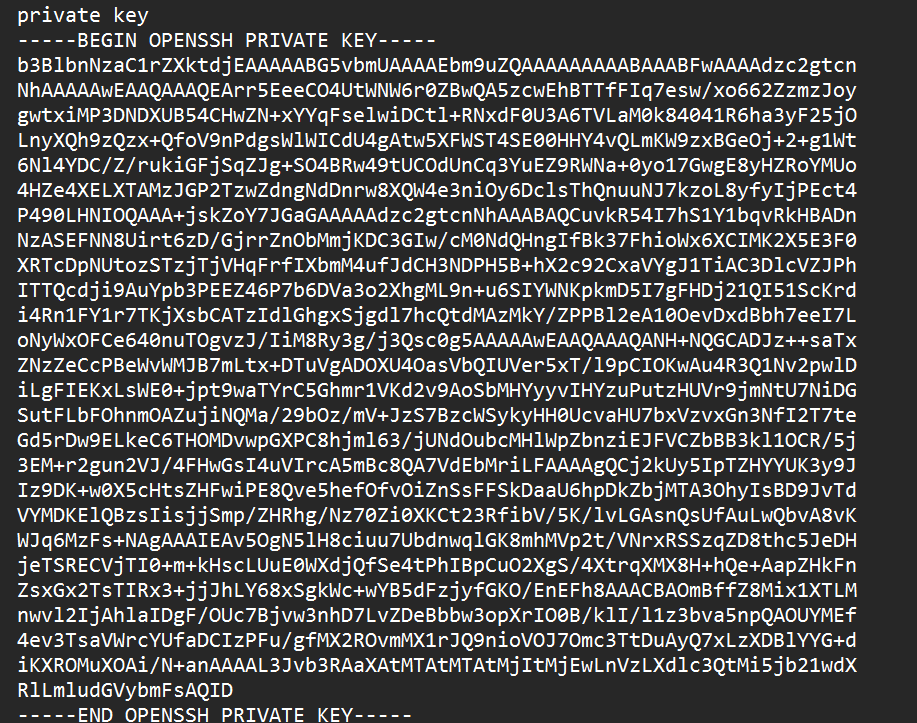
Repo 2-



**Task 2: Connect to the lab bastion host and review SSH keys for AWS CodeCommit and Amazon ECR connectivity**

2.1 Connect to Bastion Host from EC2 Instances page

2.2 Verify the SSH keys using CLI, and copy the private key locally



As per my knowledge, these steps are just for security and roles/permissions,etc. So skipping them from the document

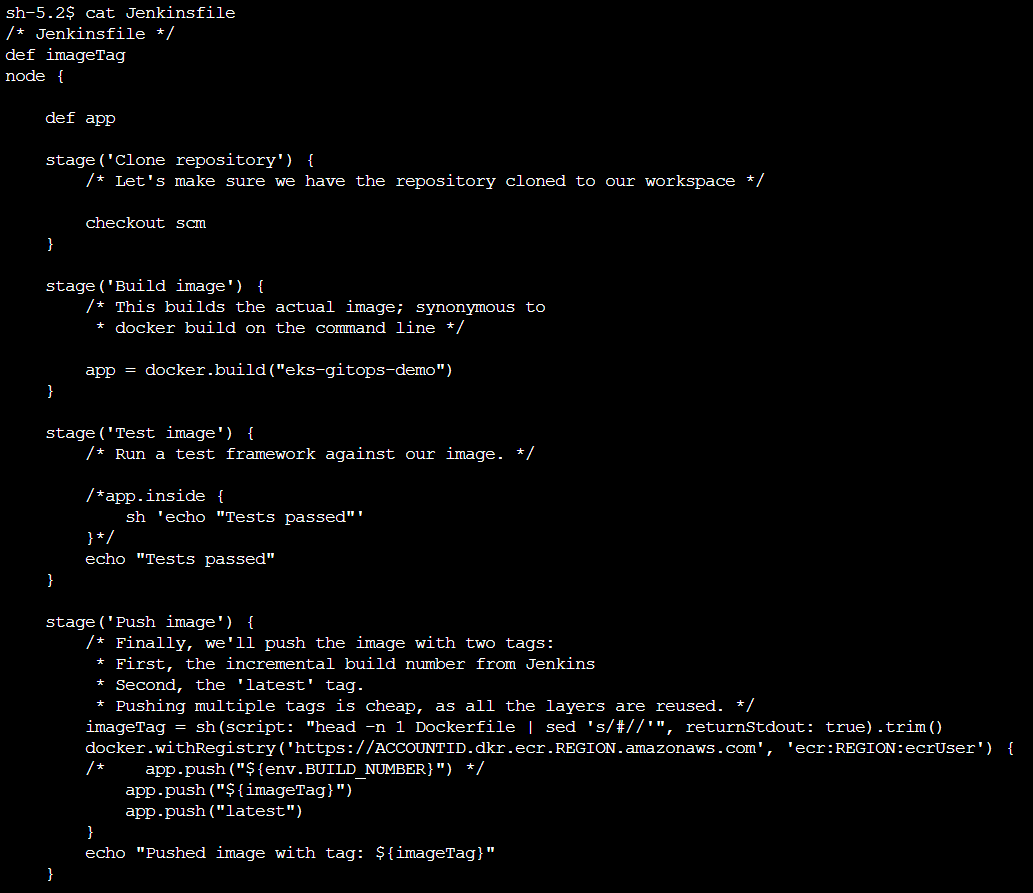
**Task 3: Review the application code and push it to AWS CodeCommit repository**

3.1 To view the pre-created files for the lab



1. Dockerfile: Instructions to create the initial container image for your sample application.
2. Dockerfile-Final: Revised instructions to create an optimized container image for your sample application.
3. Jenkinsfile: Defines stages for the initial Jenkins pipeline to build the container image.
4. Jenkinsfile-Final: Defines stages for the final, possibly enhanced, Jenkins pipeline.
5. src: Contains source code for the sample "Hello World" application using nginx.

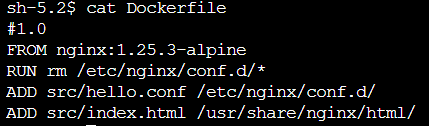
3.2 View the jenkinsfile



1. Clone Repository: Gets the latest code from AWS CodeCommit, using checkout scm command.
2. Build Image: Creates a Docker image for the application.
3. Test Image: Placeholder for testing the image, currently prints a message.
4. Push Image: Uploads the Docker image to Amazon ECR with appropriate tags.

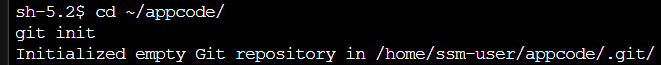
These steps ensure that only the latest code is used for building the image, and that too after testing the code

3.3 View the dockerfile

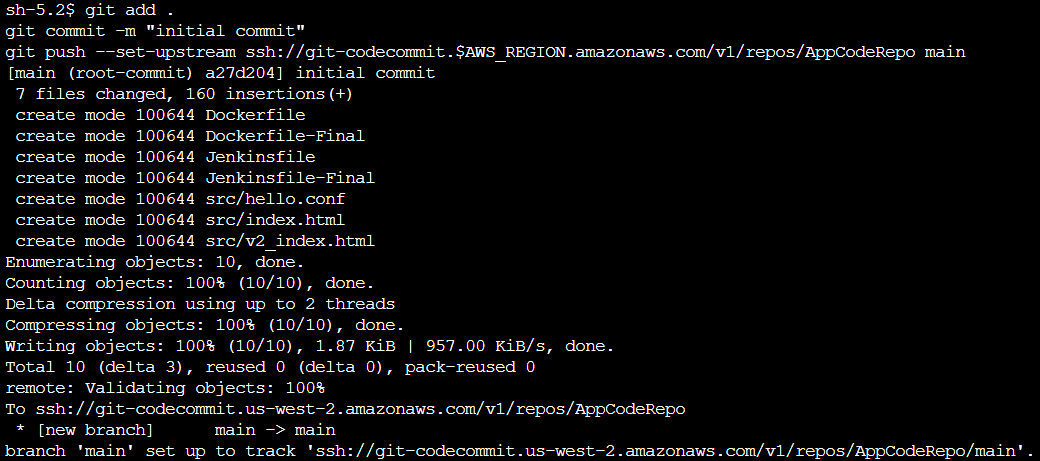


1. FROM is used to specify the base image to build the new image from.
2. RUN is used to execute commands and shell scripts during the build process of the Docker image.
3. ADD is used to copy files from the host machine into the Docker image.

3.4 convert the ~/appcode directory into a local Git repository,

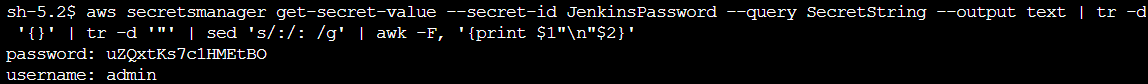


3.5 push the application code to the main branch in your AppCodeRepo repository

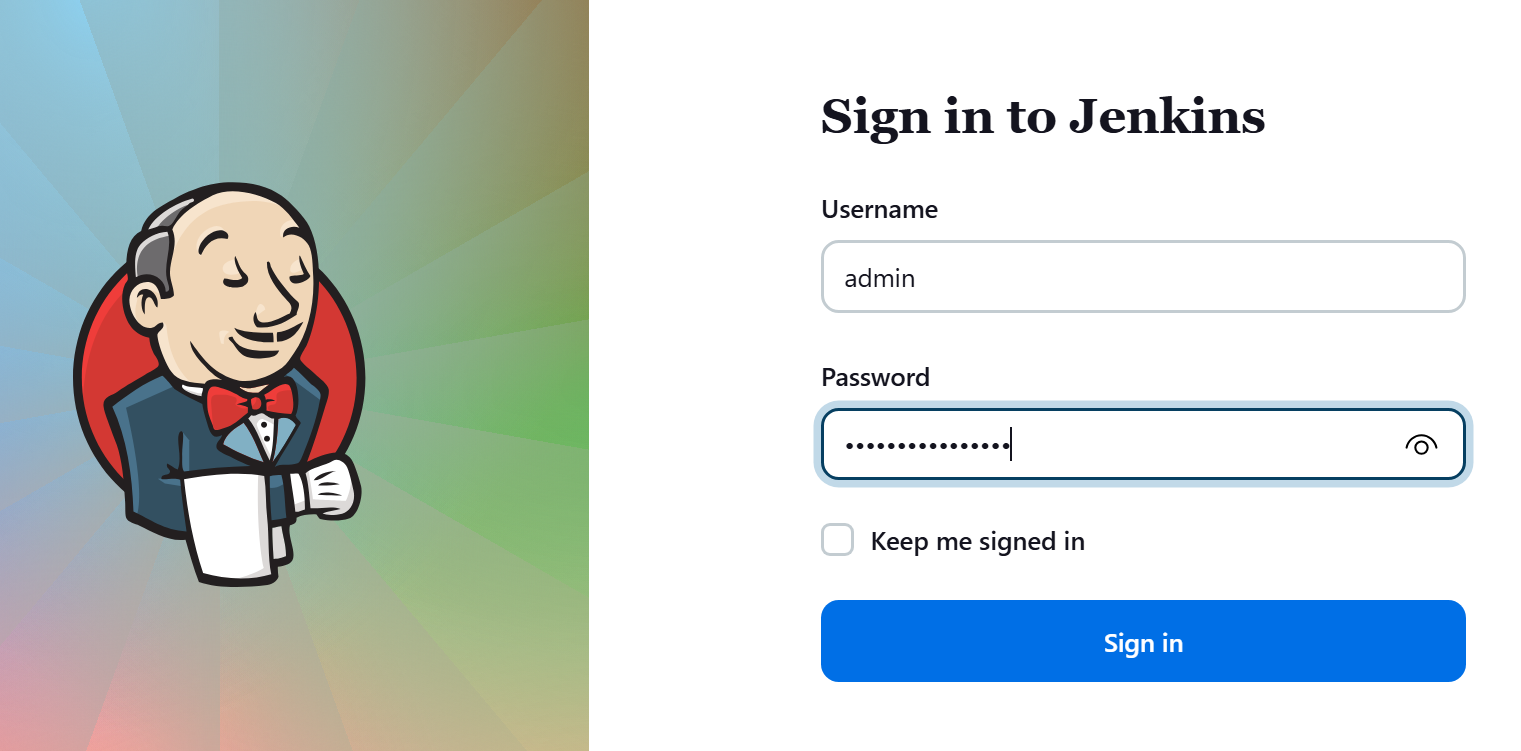


**Task 4: Configure the Jenkins server**

4.1 To log in to the Jenkins server, you need to fetch login credentials from AWS Secrets Manager.

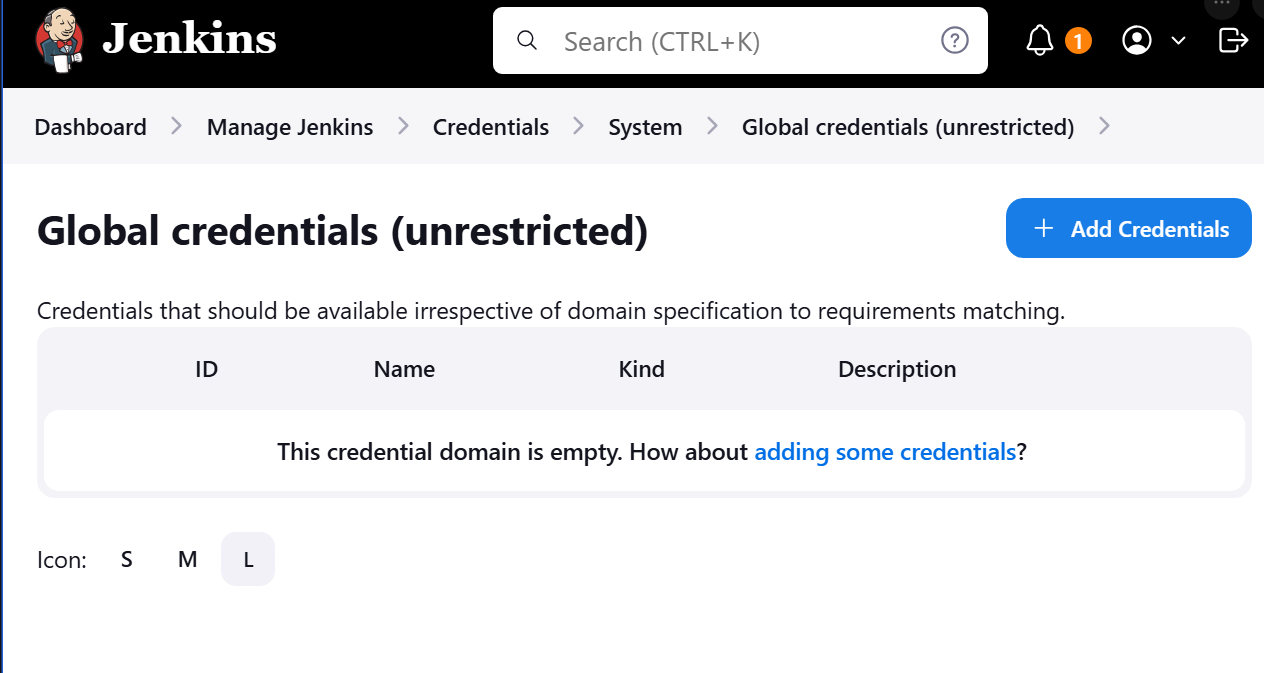


4.2 Signin Jenkins

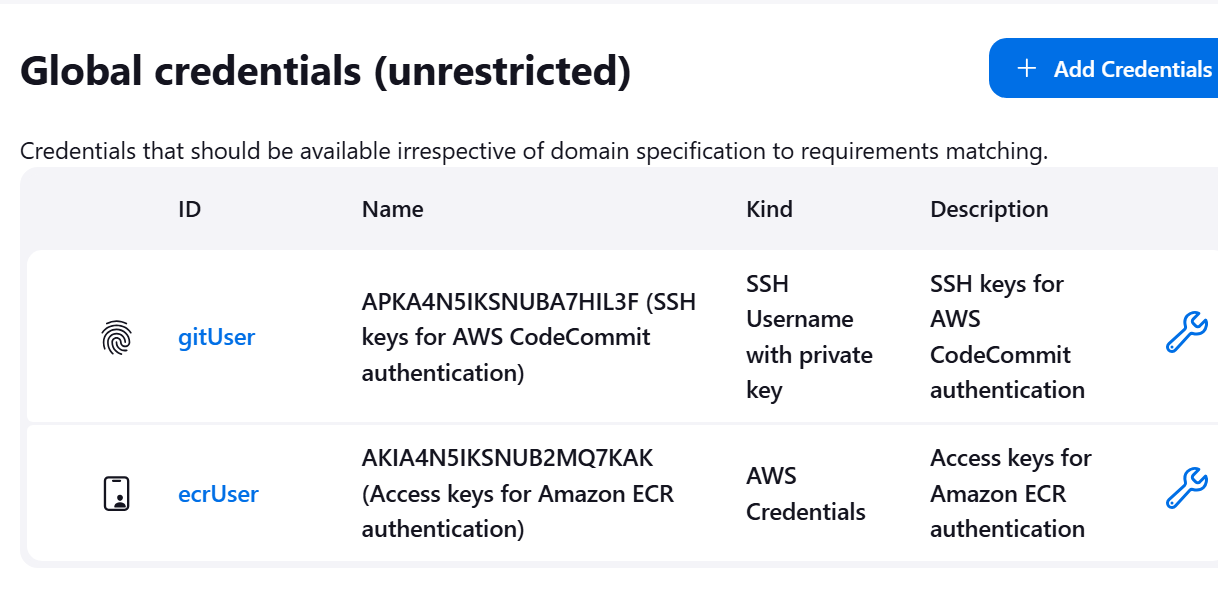


4.3 On the Manage Jenkins page, in the Security section, choose Credentials.

4.4 Click on add creds

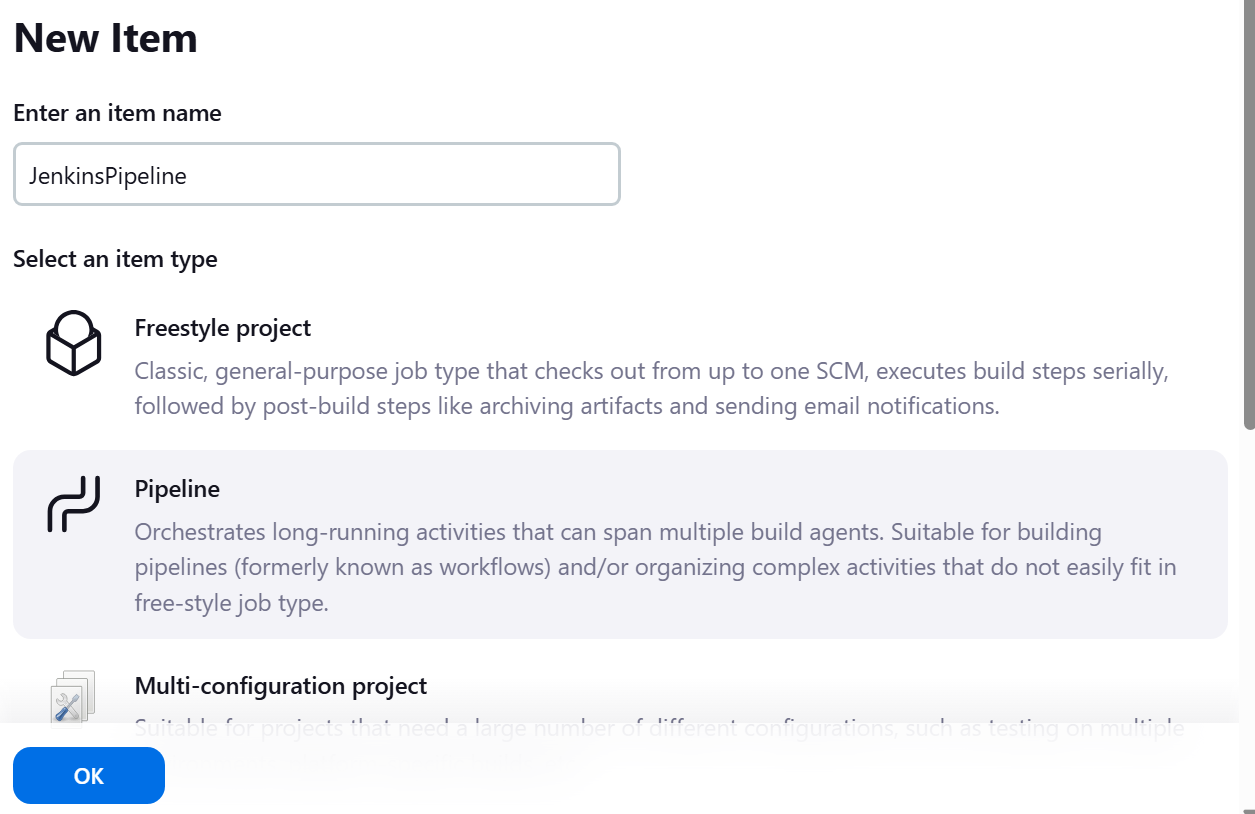


4.5 Build the two credentials using lab provided values



**To create the pipeline**

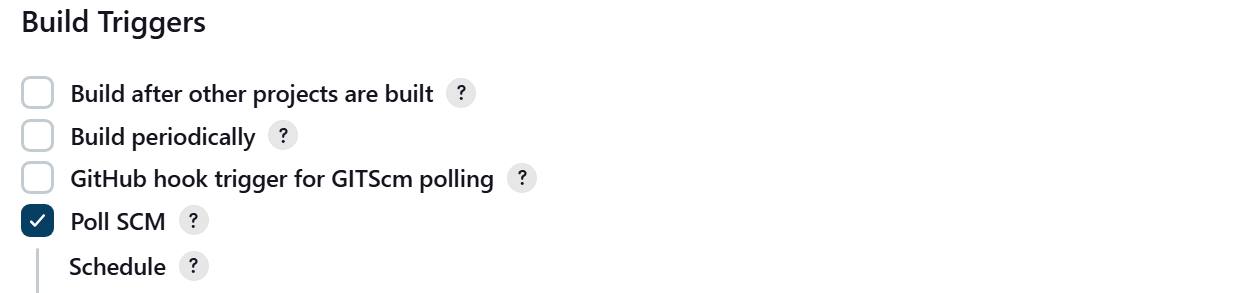
4.6 Go to dashboard, enter name, choose pipeline



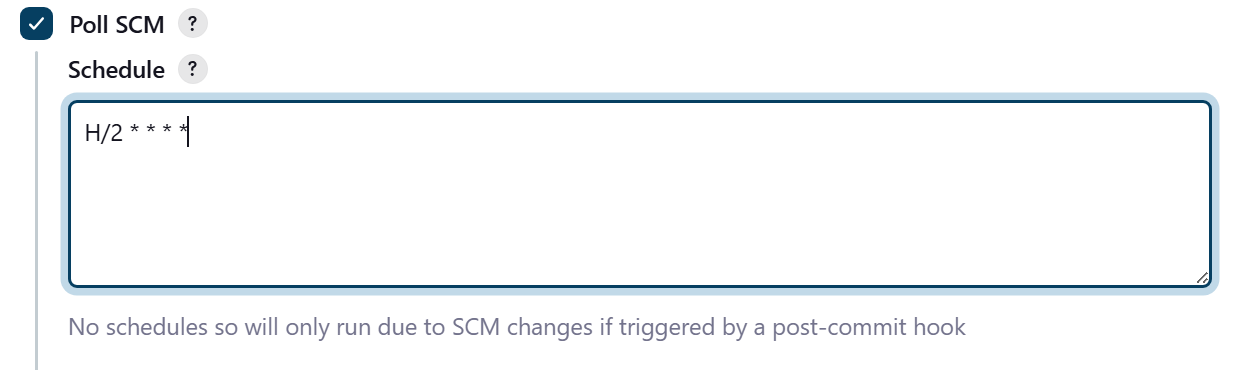
4.7 Configure pipeline using lab instructions:

In the following steps you configure source control management (SCM) for your pipeline, point it towards your AWS CodeCommit repository.

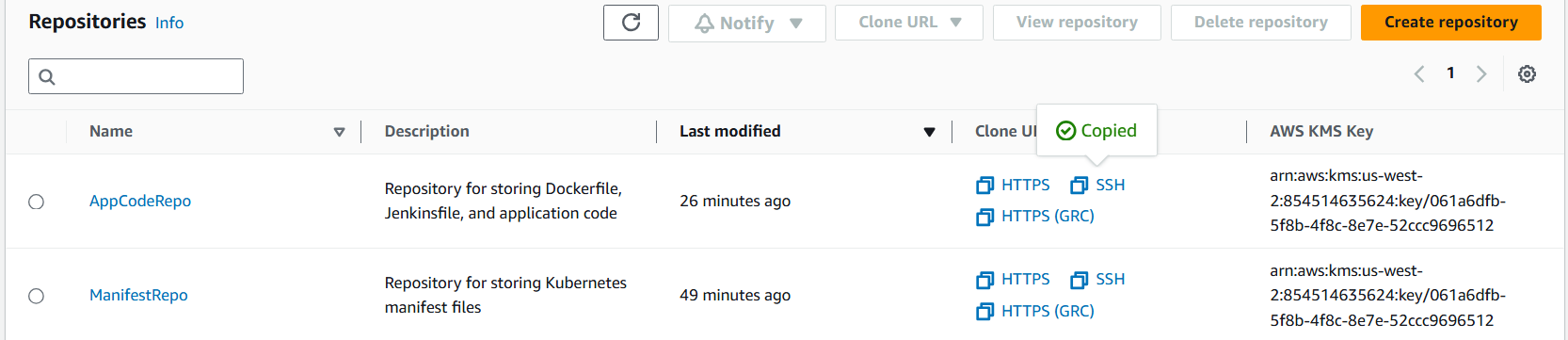
Choose SCM(Source control management)



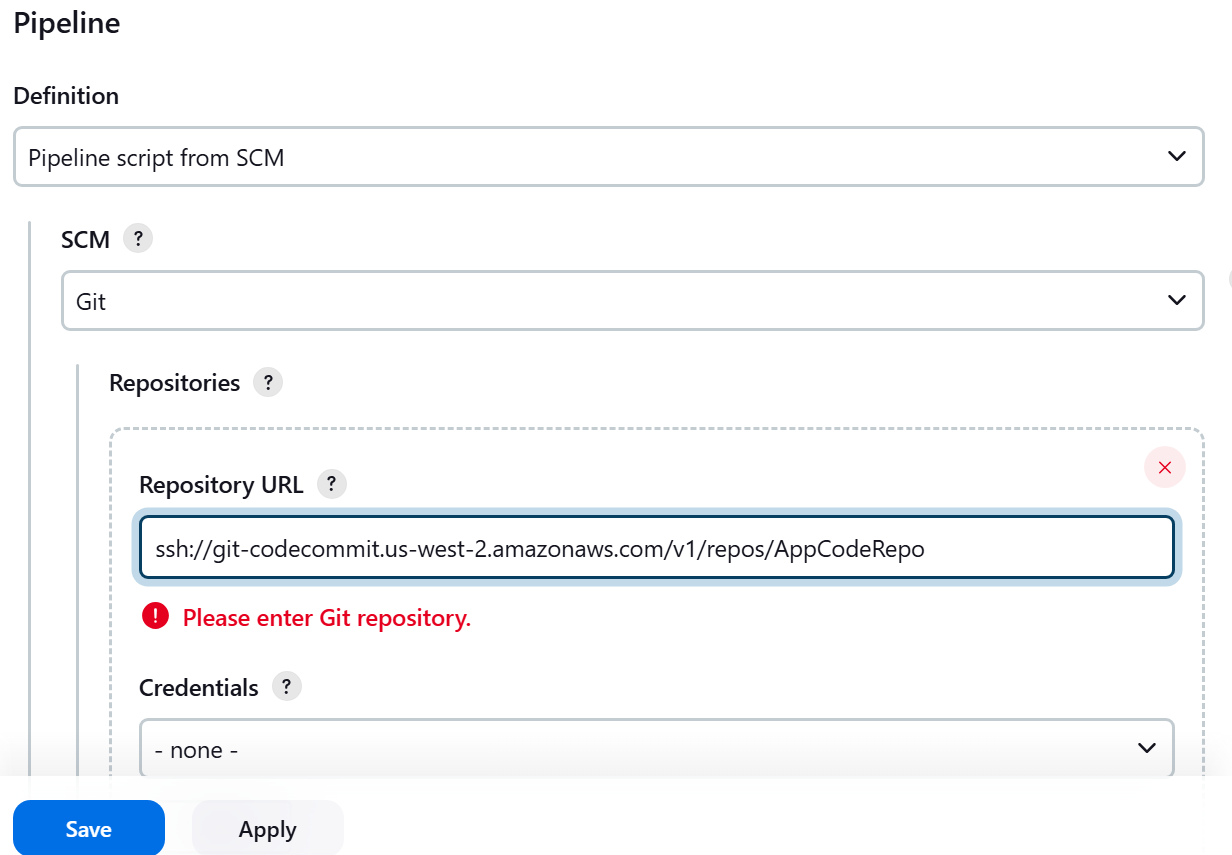
A text box named Schedule appears under Poll SCM. You use this field to create a cronjob that checks your AWS CodeCommit repository every two minutes for new commits. Whenever a new commit is detected, Jenkins will automatically build a new image and push it to Amazon ECR.



4.8 To connect the source of this pipeline as the CodeCommit Repo, copy Clone SSH link

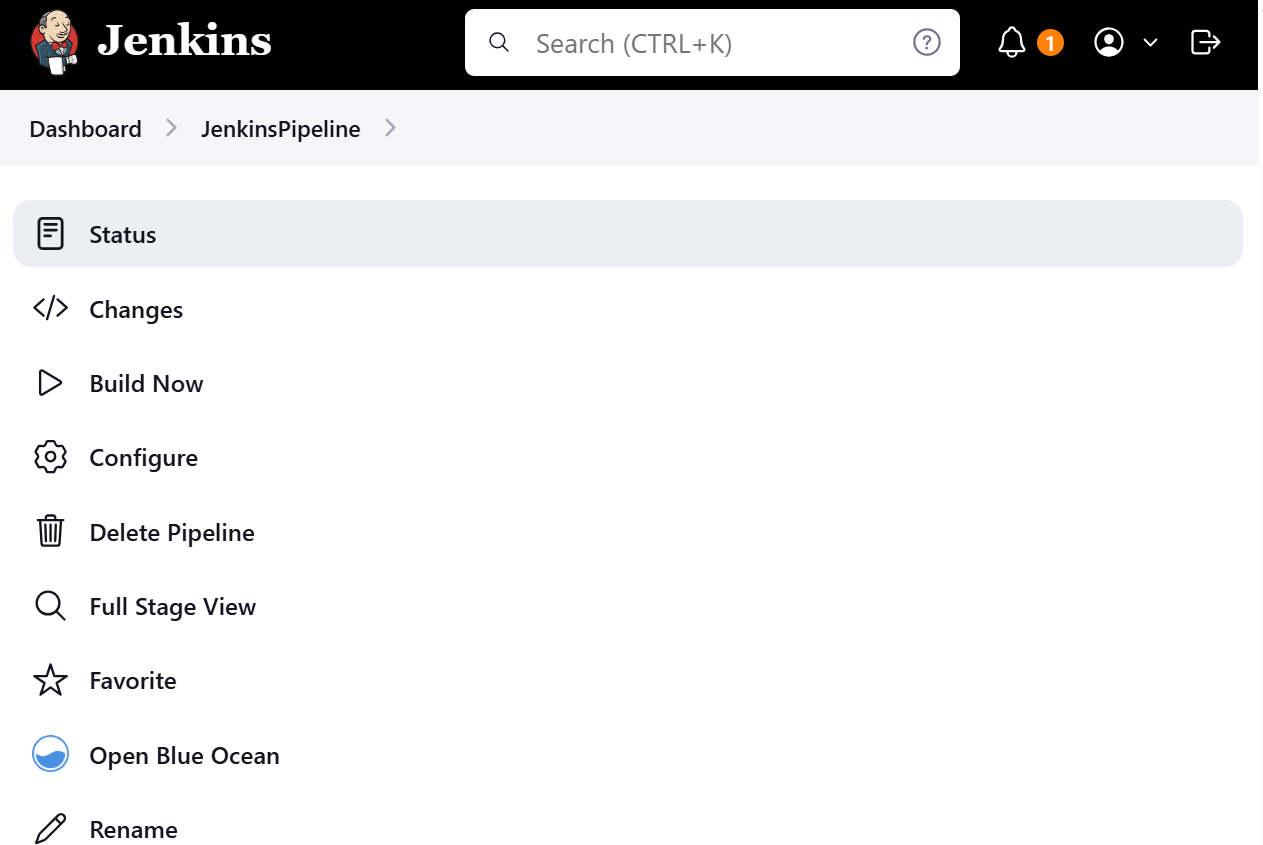


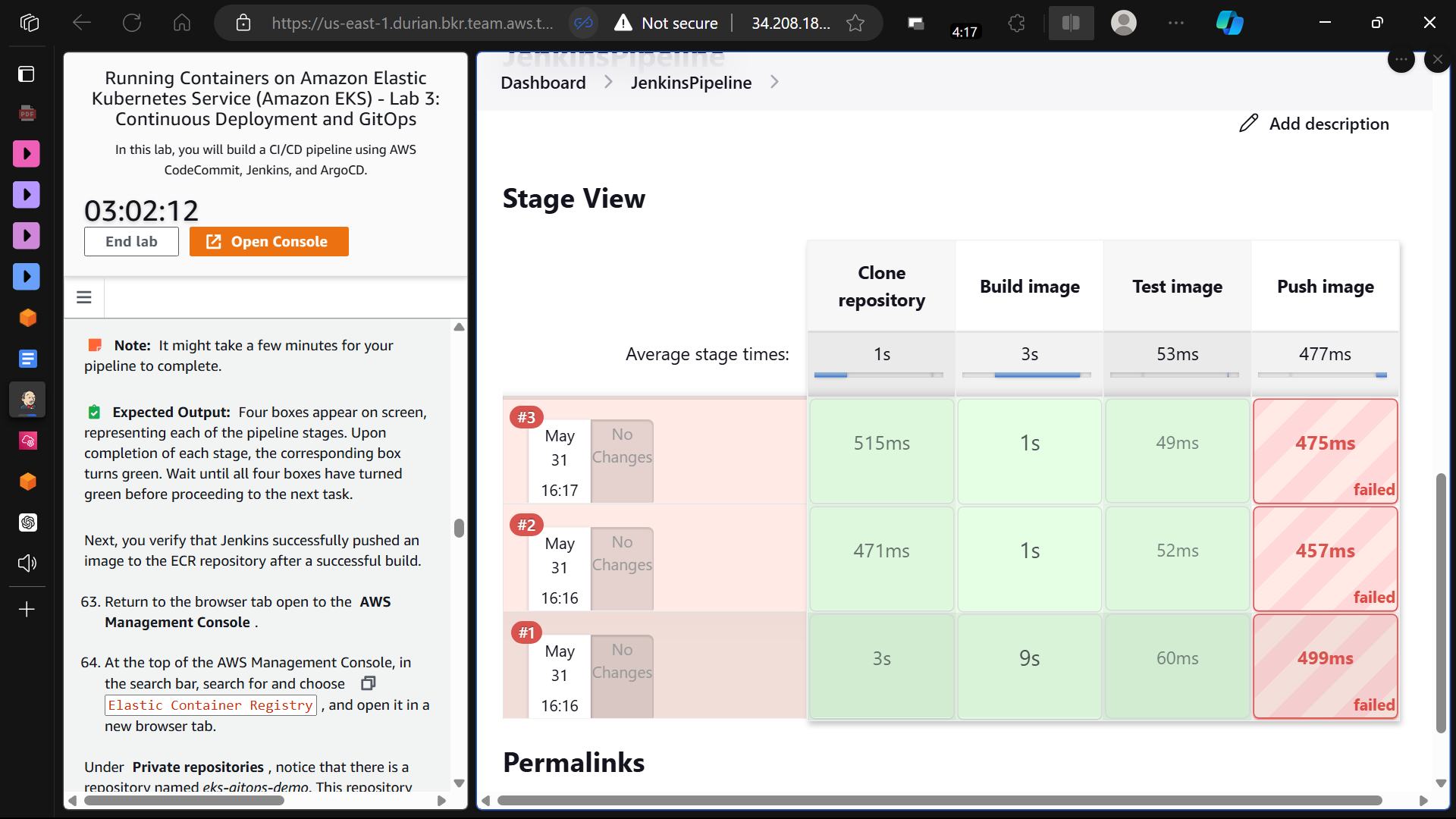
4.9 Paste the value in Pipeline SCM

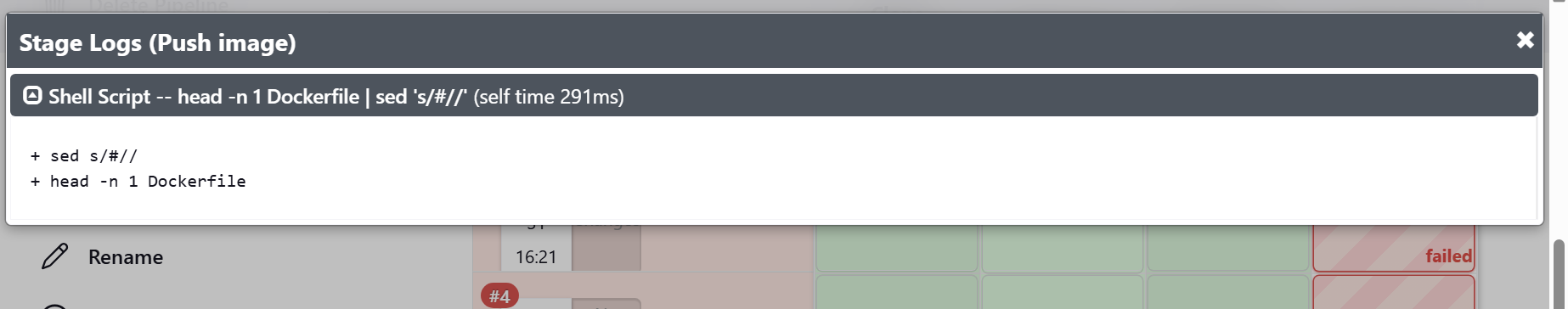


4.10 Click on save

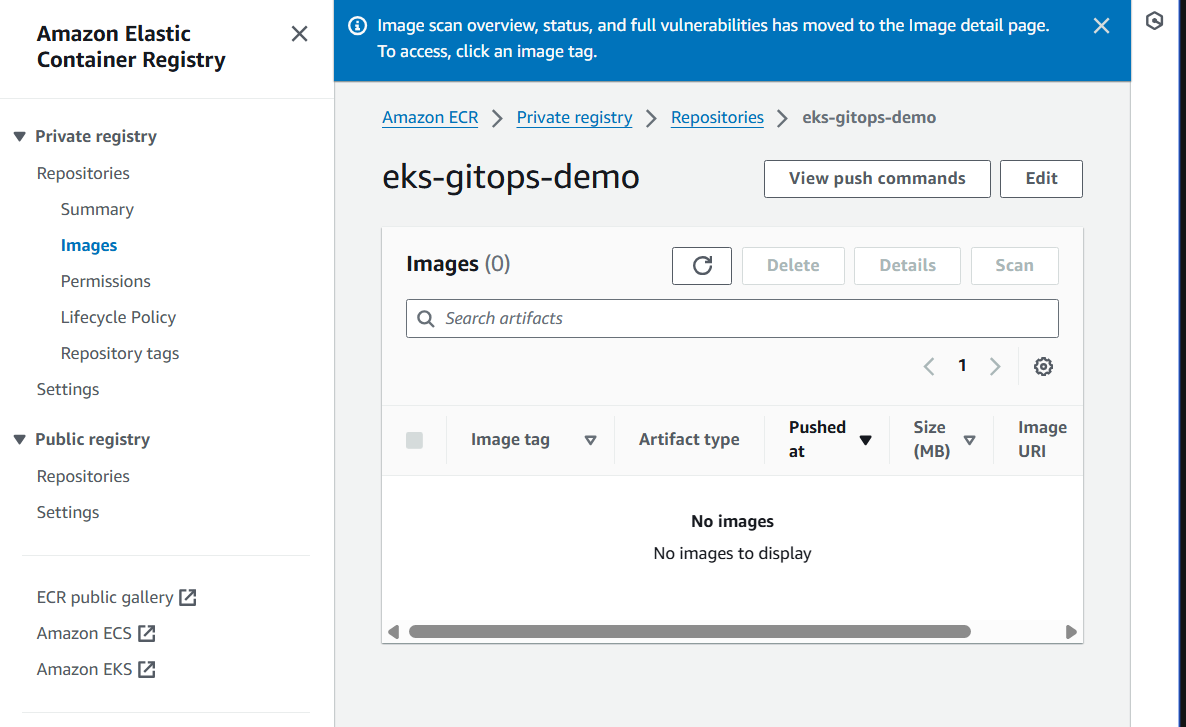
4.11 Click on build now



**ERROR - **

****

**Due to this error, in ECR, no image is being pushed by the Jenkins Pipeline**

****