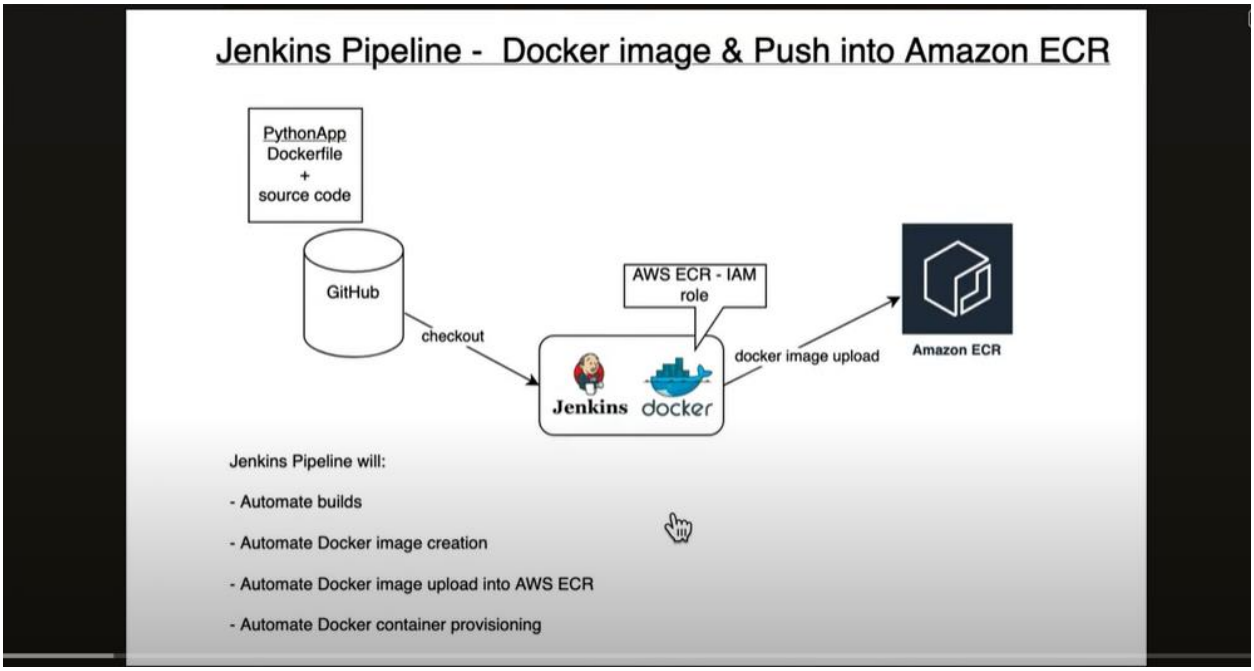
PROJECT 4

Creating docker image using Jenkins Declarative Pipeline and Upload to AWS ECR



In this project, you are going to be able to carry out the following 4 main tasks:

Automation Processes

1. Automate the provision of Jenkins CI server using terraform
   Install the following tools in the Jenkins ci server
   - Docker.io
   - Maven
   - Git
   - Aws cli
   - Ansible
   - Terraform

If already installed use the which command to investigate the path in which the tool is installed

Example

which terraform

Output

/usr/bin/terraform

2. Create a Jenkin's declarative pipeline script that will carry out the following:
    - Automate the cloning of the source code
    - Automate the builds process using gradle
    - Automate Docker image build using docker dependencies
    - Automate Docker image tag
    - Automate Docker image push to ECR
    - Automate the process of stopping previous containers running in the docker engine
    - Automate the process of running a Docker container using the image created and stored in ECR.

Manual Processes

You are going to manually carry out the following:

- Login to your aws console and create an ECR
- Remember Jenkins and docker will work together to automate the process of building the docker image.
- Create an Amazon Elastic container registry

- Create an IAM role with AmazonEC2ContainerRegistryFullAccess policy, attach to Jenkins EC2 instance

Use a name you can remember for the IAM role

For example

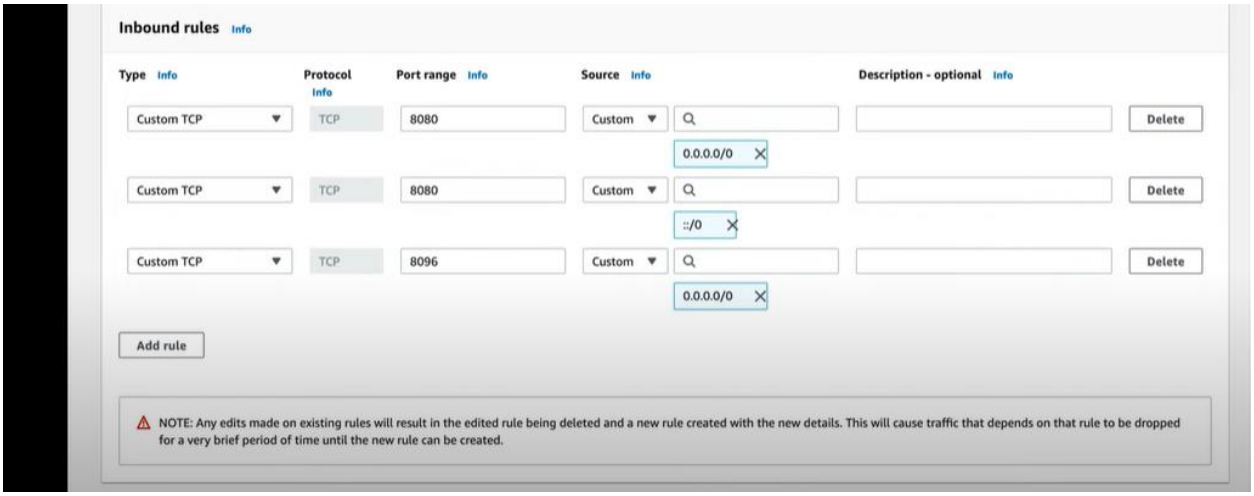
jenkins-ec2-ecr-role

Assign this I am role

AmazonEC2ContainerRegistryFullAccess policy

Go to the instance console and attach jenkins-ec2-ecr-role to the running Jenkins ci server instance

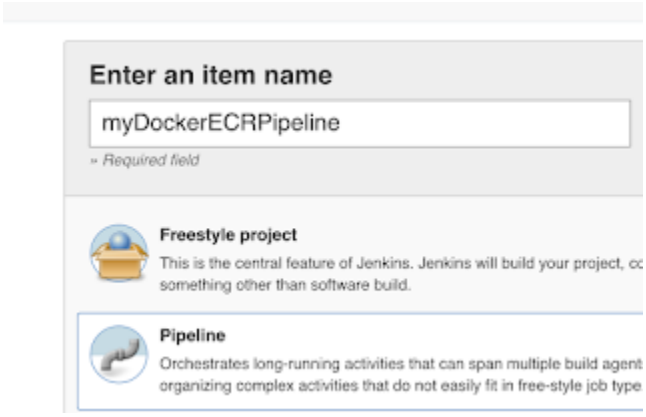
- Open port 8096 on the Jenkins CI server. This is the port we are going to use to run docker container in jenkins's server.



- Connect docker plugin with Jenkins

Environment setting is done

Step # 1 - Create a pipeline in Jenkins, name can be anything



Go to GitHub and fork this repository

Link:
https://github.com/joshking1/jenkinsECRPythonDockerRepo.git

Go to the src code and pay attention to the docker file

This project is using python source code. Maven will not be required

COPY app.py /usr/src/app/

COPY templates/index.html /usr/src/app/templates/

Use the Jenkins declarative Pipeline for this project

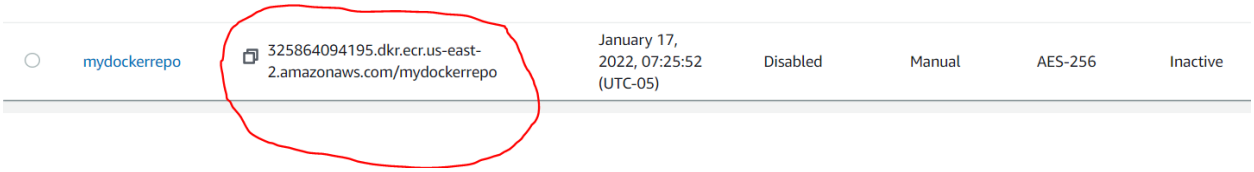
Declarative Jenkins pipeline

The pipeline must start with pipeline {

Declarative Jenkins Pipeline Code

First Stage – Define the agent and environment

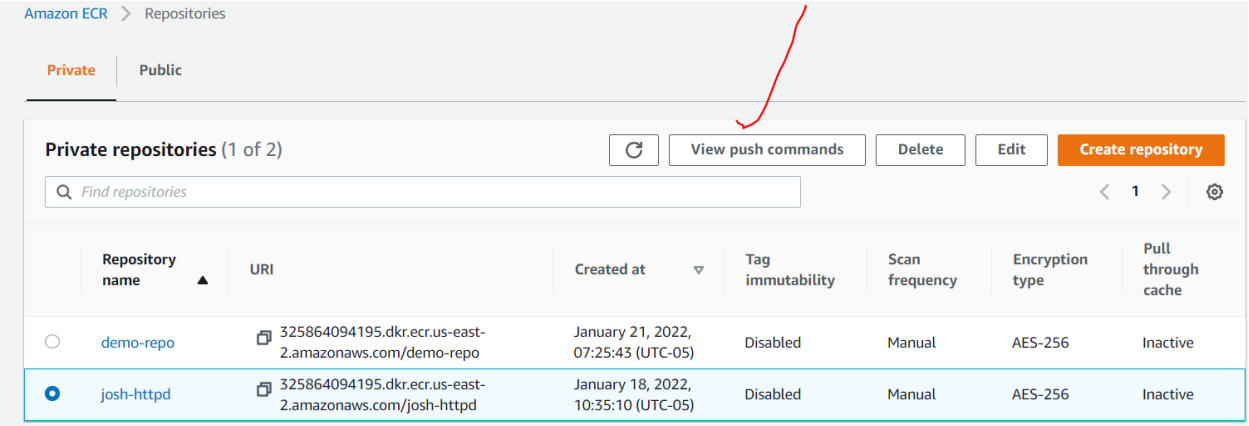
```
pipeline {
    agent any     [you do not need to put anything for the agent]

    environment {
        registry = "232110768834.dkr.ecr.us-east-2.amazonaws.com/mydockerrepo"
    }
```



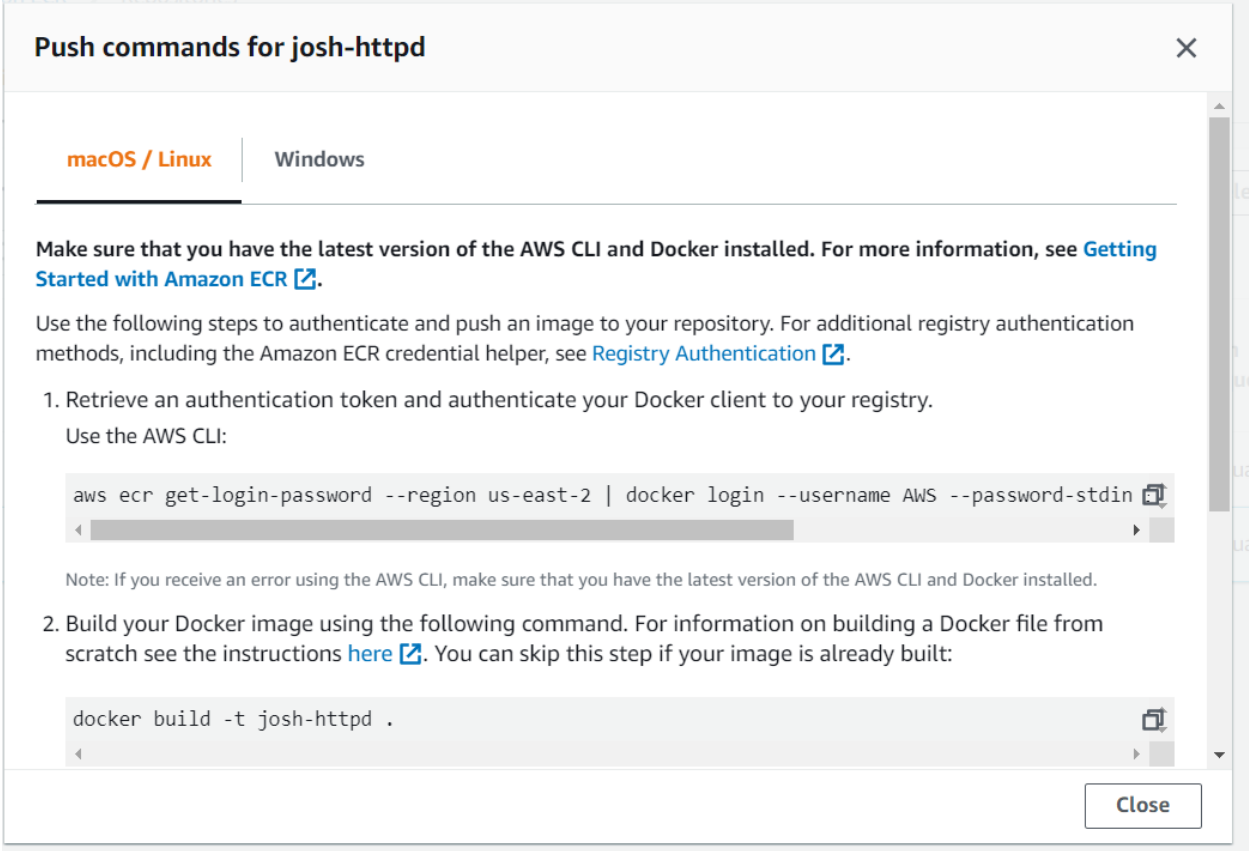
Registry information is contained in your AWS ECR

I circled it for easy recognition – Your repository registry is not going to be like mine

Use the View Push commands for the information you need to push the image to the ECR



Example of push commands



View push command to Authentication

aws ecr get-login-password --region us-east-2 | docker login --username AWS --password-stdin 325864094195.dkr.ecr.us-east-2.amazonaws.com

View push Commands to build

docker build -t josh-httpd .

View command to tag the image

docker tag josh-httpd:latest 325864094195.dkr.ecr.us-east-2.amazonaws.com/josh-httpd:latest

View command to push image

docker push 325864094195.dkr.ecr.us-east-2.amazonaws.com/josh-httpd:latest

How to write the declarative Jenkins pipeline code

```
pipeline {
    agent any


    environment {
        registry = "325864094195.dkr.ecr.us-east-2.amazonaws.com/mydockerrepo"
    }
    stages {
        stage ("Git Checkout") {
            steps {
                checkout([$class: 'GitSCM', branches: [[name: '*/master']], extensions: [], userRemoteConfigs: [[credentialsId: 'GIT_HUB_CREDENTIALS', url: 'https://github.com/joshking1/jenkinsECRPythonDockerRepo.git']]])
            }
        }
        stage ("Docker Version") {
            steps {
                script {
```

```
            sh "docker version"
        }
    }
}
stage ("Building image") {
    steps {
        script {
            sh "docker build -t mydockerrepo ."
        }
    }
}

stage ("Docker Image List") {
    steps {
        script {
            sh "docker image list"
        }
    }
}
stage('Image Tag') {
    steps {
        script {
```

```
                sh "docker tag mydockerrepo:latest
325864094195.dkr.ecr.us-east-2.amazonaws.com/mydockerrepo:latest"
            }
        }
    }
    stage('Image Login') {
        steps {
            script {
                sh "sudo aws ecr get-login-password --region us-east-2 |
docker login --username AWS --password-stdin
325864094195.dkr.ecr.us-east-2.amazonaws.com"
             }
        }
    }
    stage('Pushing to ECR') {
        steps {
            script {
                sh 'docker push 232110768834.dkr.ecr.us-east-
2.amazonaws.com/mydockerrepo:latest'
            }
        }
    }
    stage('stop previous containers') {
        steps {
```

```
            sh 'docker ps -f name=mypythonContainer -q | xargs --no-run-if-empty docker container stop'
            sh 'docker container ls -a -fname=mypythonContainer -q | xargs -r docker container rm'
        }
    }
    stage('Docker Run') {
        steps {
            script {
                sh 'docker run -itd -p 8096:5000 --name mypythonContainer 232110768834.dkr.ecr.us-east-2.amazonaws.com/mydockerrepo:latest'
            }
        }
    }
  }
}
```

Good luck !!!