Jenkins Documentation

Jenkins

USER MANUAL

Revision History

Date Created	version	Description	Created by
08-02-2023	1.0.0	Initial document release	Madhavi K.

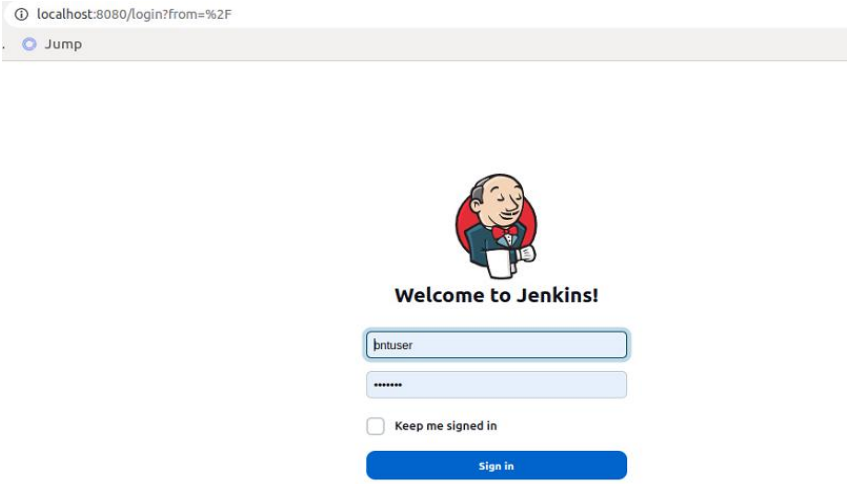
Table of Contents

1. Introduction Jenkins:

Jenkins is a self-contained, open source automation server which can be used to automate all sorts of tasks related to building, testing, and delivering or deploying software.
Jenkins can be installed through native system packages, Docker, or even run standalone by any machine with a Java Runtime Environment (JRE) install.

2. Environment Details:

Server :172.31.21.62
Url- http://localhost:8080/
Username: ##
Password: ##

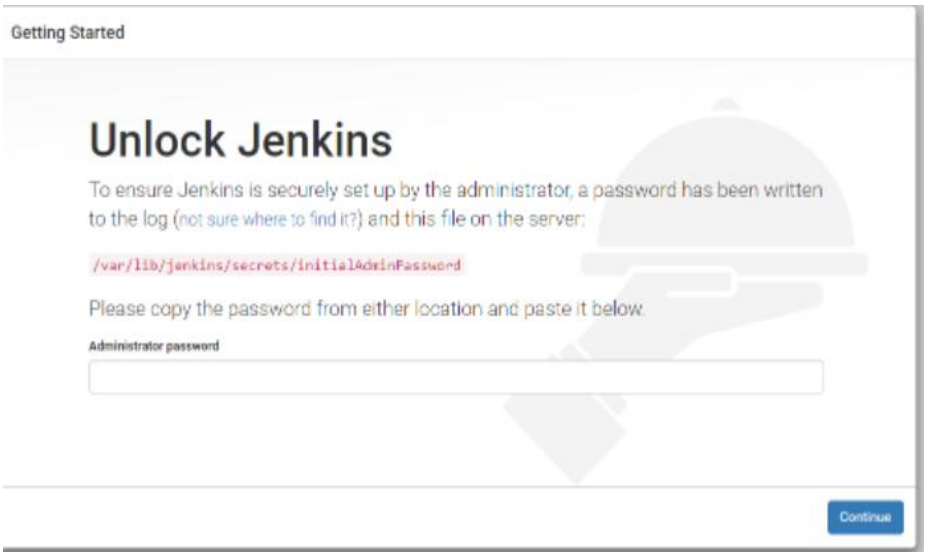


(Login page)

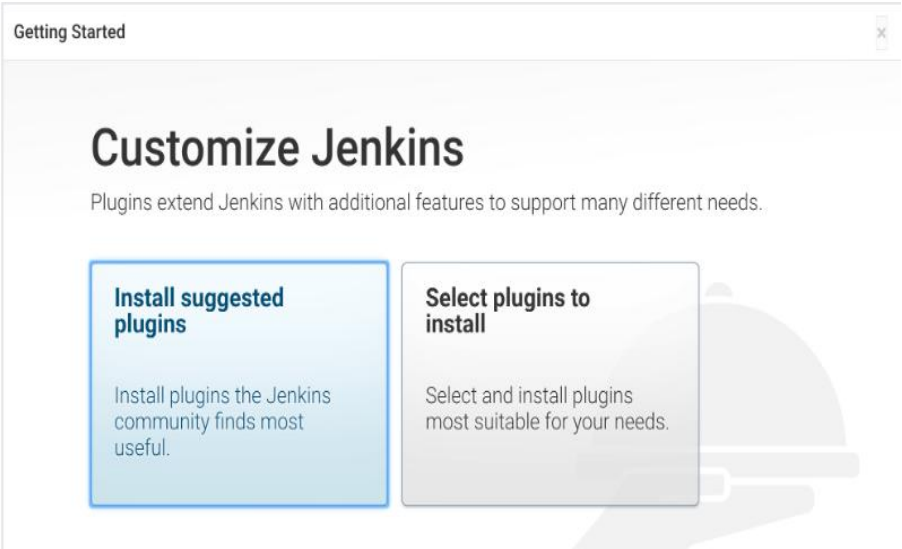
3. Required Tools To Be Install:

3.1. Jenkins On Local Machine.(for ubuntu)

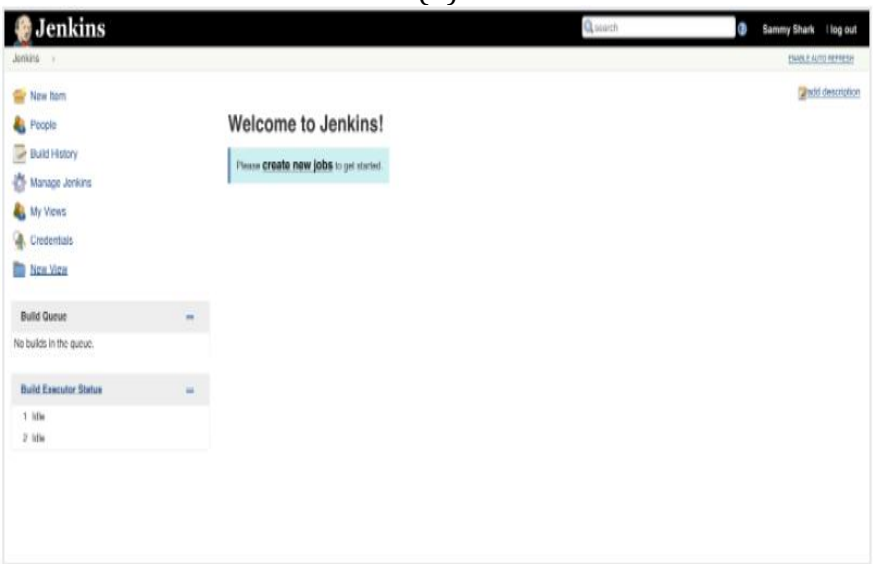
1.Install jdk 11
2.Link to install : https://www.digitalocean.com/community/tutorials/how-to-install-jenkins-on-ubuntu-18-04



(1)



(2)



(3)

----------At this point, you have successfully install Jenkins----------

3.2. Docker On Same Machine.

Commands to run on terminal:

```
$ sudo apt update
$ sudo apt install apt-transport-https ca-certificates curl software-properties-common
$ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo apt-key add -
$ sudo add-apt-repository "deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic
  stable"
$ sudo apt update
$ sudo apt install docker-ce
$ sudo systemctl status docker
```

----------Docker installation done.----------

Try out below commands:

$ docker --version /docker -v
$ docker images (showing all docker images)
$ docker ps -a (checking running containers)

```
ag@ag-System-Product-Name:~$ docker -v
Docker version 20.10.7, build 20.10.7-0ubuntu5~18.04.3
ag@ag-System-Product-Name:~$ docker ps -a
CONTAINER ID   IMAGE     COMMAND   CREATED   STATUS    PORTS     NAMES
ag@ag-System-Product-Name:~$ docker images
REPOSITORY                                                        TAG              IMAGE ID       CREATED        SIZE
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      latest           f589e3f0e529   3 days ago     121MB
jenkinspipeline                                                   latest           f589e3f0e529   3 days ago     121MB
mydemo_6                                                          latest           f589e3f0e529   3 days ago     121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           4c63c2bde4a7   3 days ago     121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           6a2f96486648   3 days ago     121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           c8f3ea03df7f   2 weeks ago    121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           22a0086ff8d5   2 weeks ago    121MB
myrepo-agora/mydemo_1                                             latest           1c302660c316   2 weeks ago    121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           1c302660c316   2 weeks ago    121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           3d178874bc93   2 weeks ago    121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           9aff9701338b   2 weeks ago    121MB
180522143609.dkr.ecr.us-east-1.amazonaws.com/jenkinspipeline      <none>           d151d9e7f0e0   2 weeks ago    121MB
<none>                                                            <none>           00489c8ae180   2 weeks ago    121MB
<none>                                                            <none>           5ae1b3e7eed4   2 weeks ago    121MB
node                                                              14.18.3-alpine   194cd0d85d8a   3 months ago   118MB
ag@ag-System-Product-Name:~$
```

3.3. AWS Account.

- Install AWS cli

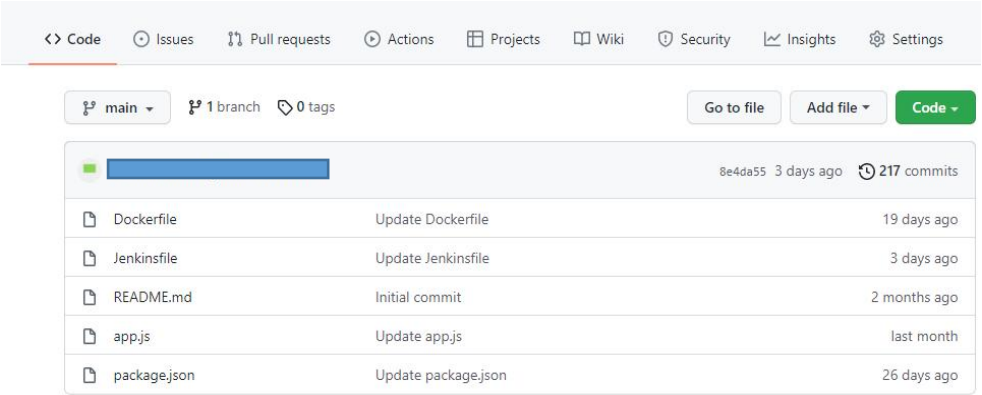
$ AWS configure

--->Configure AWS through terminal by adding IAM user Access key and Secrete key,default region,Default output format

```
ag@ag-System-Product-Name:~$ aws configure
AWS Access Key ID [****************ZBJN]:
AWS Secret Access Key [****************uOXz]:
Default region name [us-east-1]:
Default output format [json]:
ag@ag-System-Product-Name:~$
```

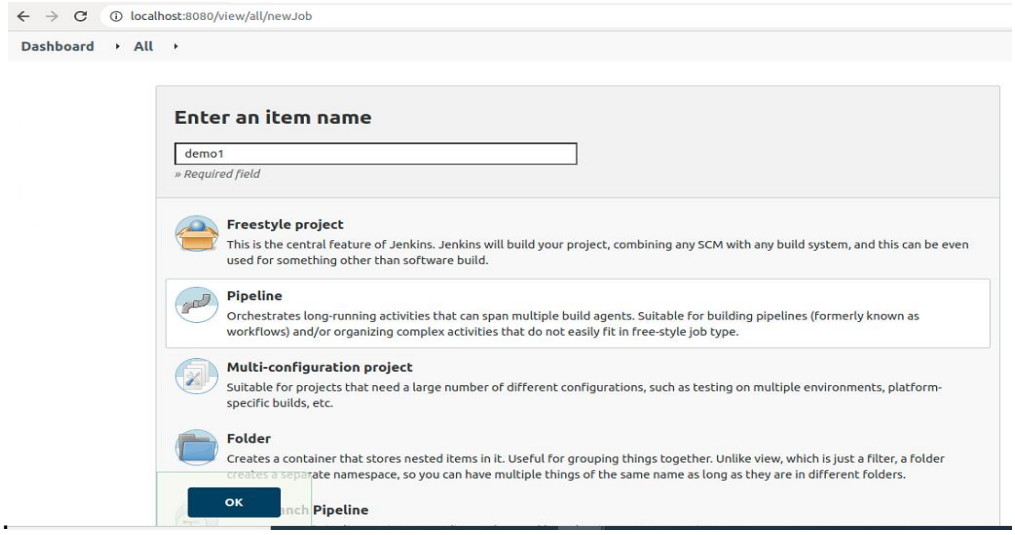
3.4. Git Account/Repository.

- Create git repository with docker file with name: Dockerfile, Jenkinsfile with name: Jenkinsfile, application code,package.json.
  Dockerfile and Jenkinsfile should on same location.
- For continuous integration add webhook in git (not allowed for localhost:8080 Jenkins)add proper IP address.
- Currently we are not using webhook.

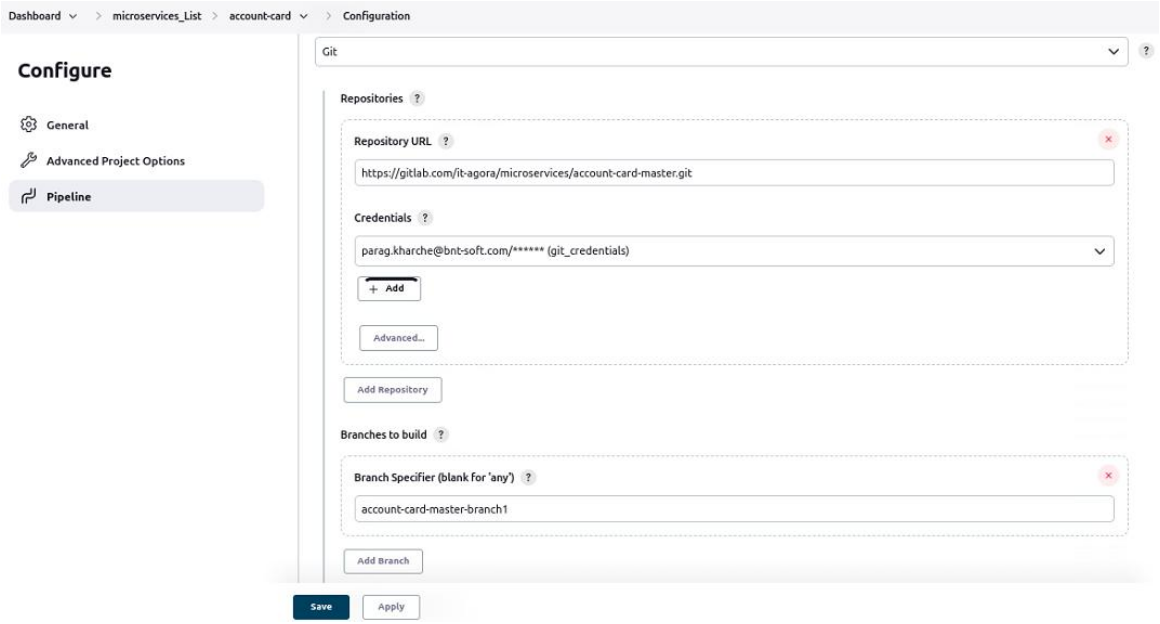


4. CICD Pipeline With Jenkins:

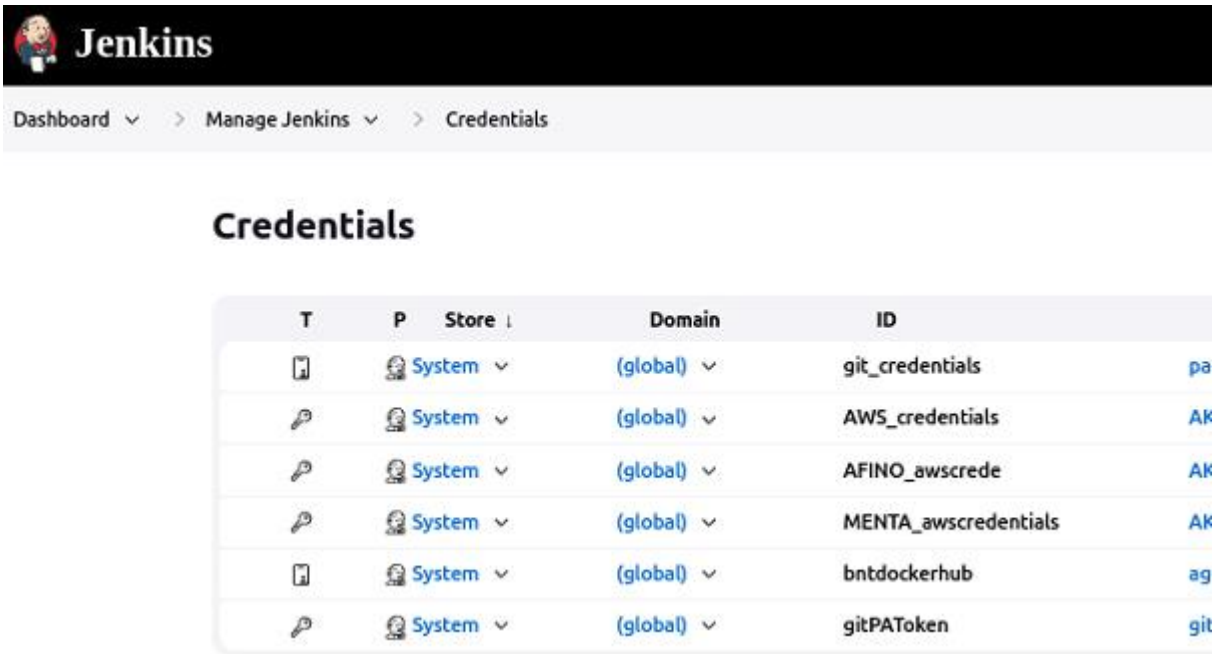
4.1. Create a pipeline job in Jenkins:



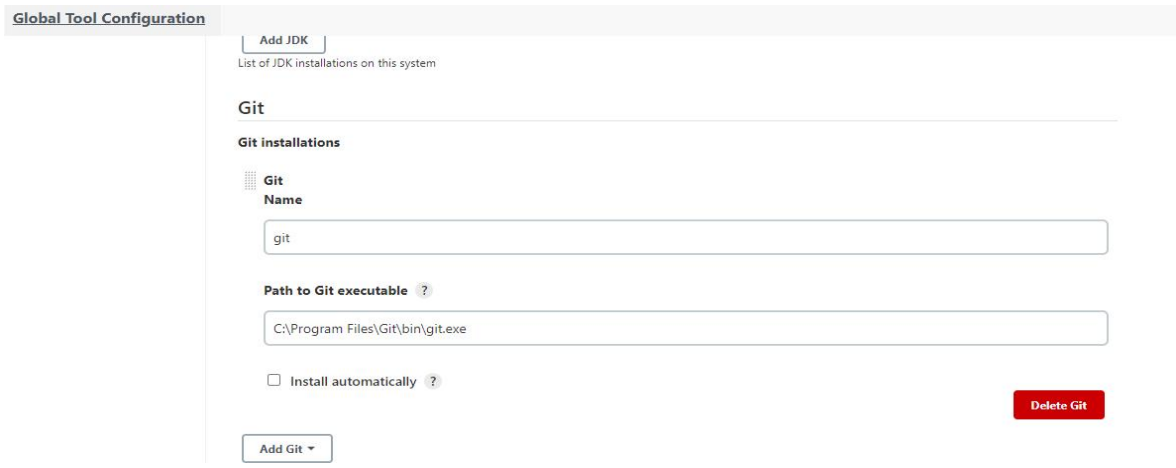
1.add name of build and select pipeline



2.add repo name and credential of git



3.add credentials of git , docker and AWS IAM user



4.add git path in Jenkins

4.2. Add plugins in jenkins:

For **git** :*Git plugin,Pipeline: GitHub Groovy Libraries, GitLab.*
docker:*CloudBees Docker Build and Publish plugin, Docker, Docker Pipeline.*
aws :*CloudBees AWS Credentials Plugin, Amazon ECR.*

4.3. Manage Jenkins-----> manage credentials --->add credentials--->add docker & git username &password.

4.4. Add credentials of git account with username and password(no need for public repository).

4.5. Add docker credentials with user_name and personal token (generated from dockerhub/docker desktop)/password.

4.6. Add aws credentials with IAM user accesskey and secretekey.

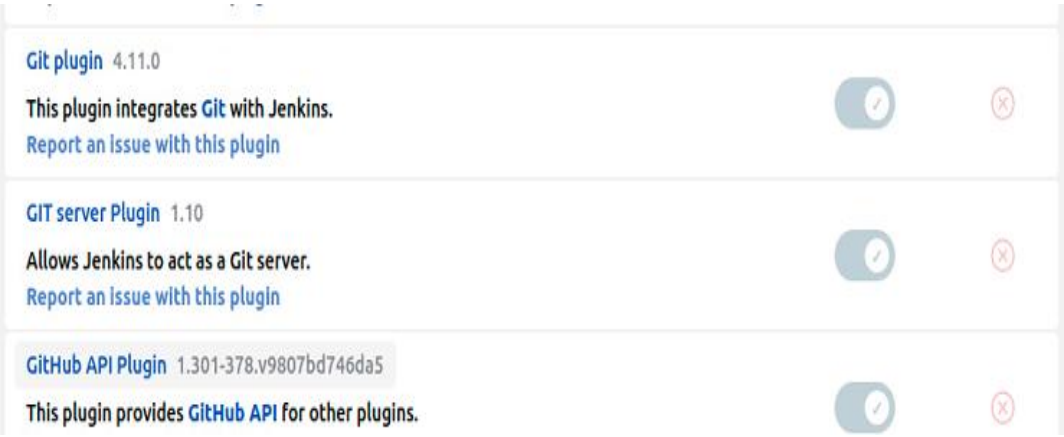
4.7. The git code with jenkins file and docker file (docker file is for creating docker image and jenkinsfile contains instruction for to do so.)

4.8. Add git repo in pipeline script from scm and select git credentials.

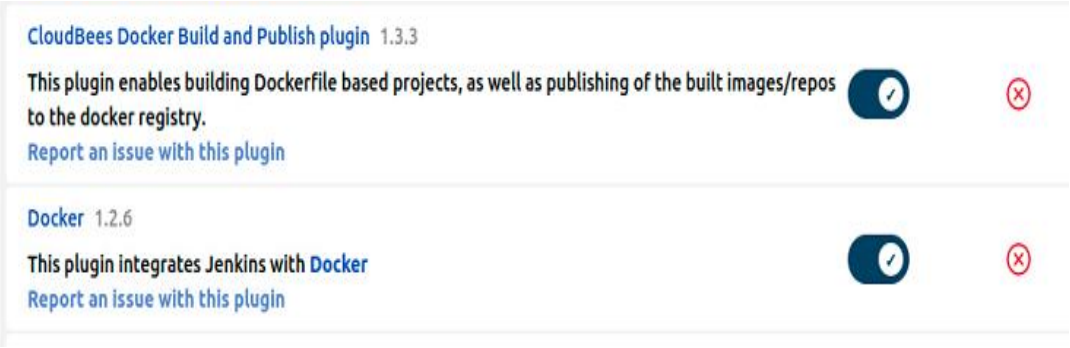
4.9. Click on apply and save.

4.10. Click on build now.

Git Plugins:

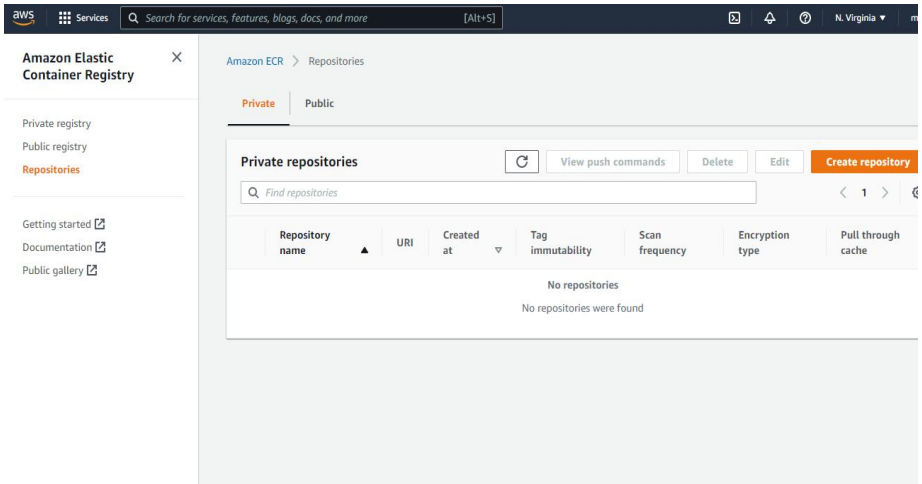


Docker Plugins:

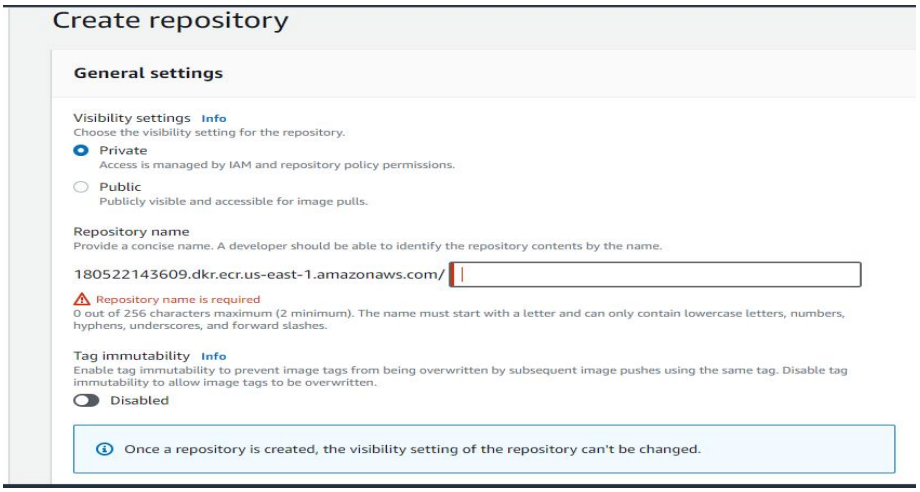


AWS Plugins:

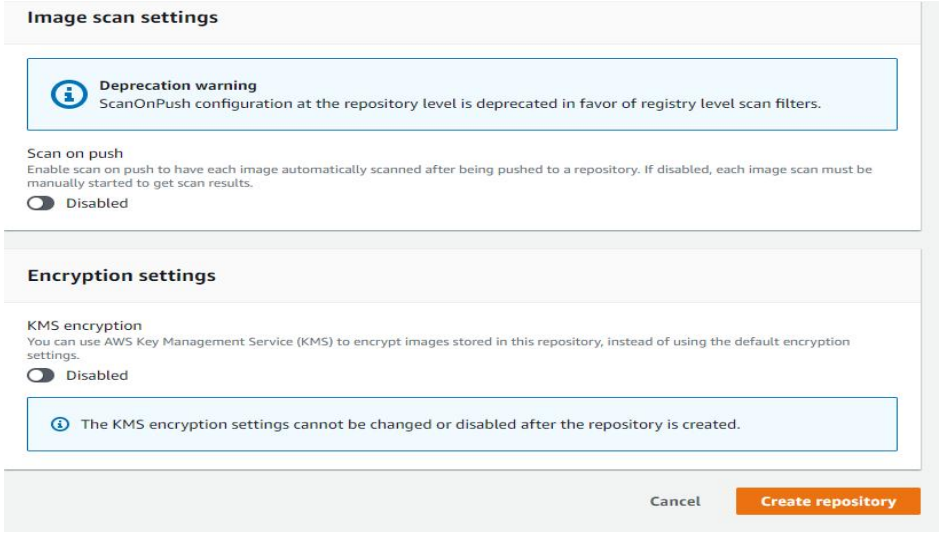
5. Create ECR On AWS :



1.create repository on ecr.



2.select private or public,add repo name.



3.click on create repository.

Output:

4.output of successful build.

---

6. Code:

Jenkinsfile present in every git repository which will deploy on ecr.--provide Jenkinsfile path in jenkins job to execute.

6.1. Jenkinsfile:

```
node {
   def app
   stage('Clone repository') {
      /* Cloning the Repository to our Workspace */
      echo '### Started cloning the repository..'
      checkout scm
      echo '### Repository cloned successfully'
   }
   stage('Build image') {
      /* This builds the actual image */
      echo '### Started Building the docker image..'
          app = docker.build (‘image_name’)
      echo '### Docker build successful.'
    }
   stage('Test image') {
      app.inside {
         echo "Tests passed"
       }
    }
   stage('Push image') {
     echo '### Started pushing the docker image..'
      /* You would need to first register with DockerHub before you can push images to your account
*/
   docker.withRegistry('https://<aws -acc-no>.dkr.ecr.<aws_region>.amazonaws.com',
'ecr:<aws_region>:<aws-credetials-In-Jenkins>') {
      app=docker.build(image_name')
  app.push ('latest')
       }
        docker.withRegistry('https://hub.docker.com/repository/abc/xyz', '<docker-crede-in-jenkins>')
{
  app=docker.build('image_name')
//            app.push("${env.BUILD_NUMBER}")
          app.push('latest')
      }
      echo '### Docker image pushed successfully.'
   }
}
```

6.2. Dockerfile:

```
FROM node:14.18.3-alpine
RUN mkdir -p /usr/src/app
WORKDIR /usr/src/app
COPY package*.json ./
RUN npm install
COPY . .
EXPOSE 3000
CMD [ "npm", "start" ]
```

Note: This CICD pipeline for nodejs application

Links to refer:
1.https://tutorials.releaseworksacademy.com/learn/building-your-first-docker-image-with-jenkins-2-guide-for-developers
2.https://medium.com/@vijulpatel865/building-docker-image-using-jenkins-pipeline-push-it-to-aws-ecr-aa02cc7a295e

Docker permission issues:
sudo chmod 666 /var/run/docker.sock
https://www.codegrepper.com/code-examples/whatever/Got+permission+denied+while+trying+to+connect+to+the+Docker+daemon+socket+at

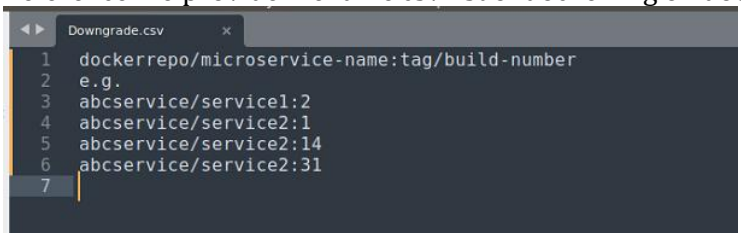
---

- Before run this job -----Docker images present on Docker hub.
- Pull docker image --tag with latest ----and push on AWS environment.
- **Downgrade the Docker image tag it and push on AWS ECR:**

6.3. Job name: Downgrade_dockerimage

```
node {
def trash =[]
docker.withRegistry('', 'bntdockerhub') {
def DockImgToPull = readFile(file: '<file-path>/filename.csv') //provide filenamelist of dockerimg.
echo "${DockImgToPull}"
            def DockerImg = DockImgToPull.split("\r?\n")
 for (i in DockerImg){
    def PulledImgName = i.split(":")
    def repo ="${PulledImgName[0]}"
    def build_num = "${PulledImgName[1]}"
    def tag_name ='latest'
    def ImageName = repo +":"+ build_num
    echo "${ImageName}"
    def MicroServiceName =repo.split("/")
    echo "MicroServiceName=${MicroServiceName[1]}"
    def app = "${MicroServiceName[1]}" +":"+ tag_name
      echo "${app}"
stage("pulling docker image") {
    echo 'looking forword '
     try{
       docker.image (ImageName).pull()
      sh ("docker tag ${ImageName} ${app}")
      app = docker.image("${MicroServiceName[1]}" +":"+ tag_name)
              stage('Push image to aws ecr') {
    echo '### Started pushing the docker image..'
    docker.withRegistry('< url_of_aws-ecr>', '<ecr-region>:<aws-credential>') {
   app.push()
       echo '### Docker image pushed on aws ecr successfully.'
                        }
            }
    }
       catch(Exception e) {
       echo "${repo} image is failed to push"
          trash.push("${repo}")
             }
         }
      }
             if ("${trash}"!=0){
    echo "list of microservices that failed to run: ${trash} " }
          }
   }
```

Reference file:provide filename.csv list of dockerimg on docker hub.



■ **Create Repository On AWS ECR-**

6.4. Job Name: create_ecrRepo

```
node {
        stage('AWS ecr') {
      echo '### Started creating ecr repo..'

         //login details:
     sh' aws configure set aws_access_key_id  <access-key>'
     sh'  aws configure set aws_secret_access_key <secret-access-key>'
     sh'  aws configure set region  <aws-region>'


         //create repository--
sh 'aws ecr create-repository --repository-name <microservice-repo-name>'

      echo '###Created repository on aws ecr successfully.'

                    }
   }
```

- **Provide a list to run multiple Jenkins jobs one-by-one or simultaneously on agents**

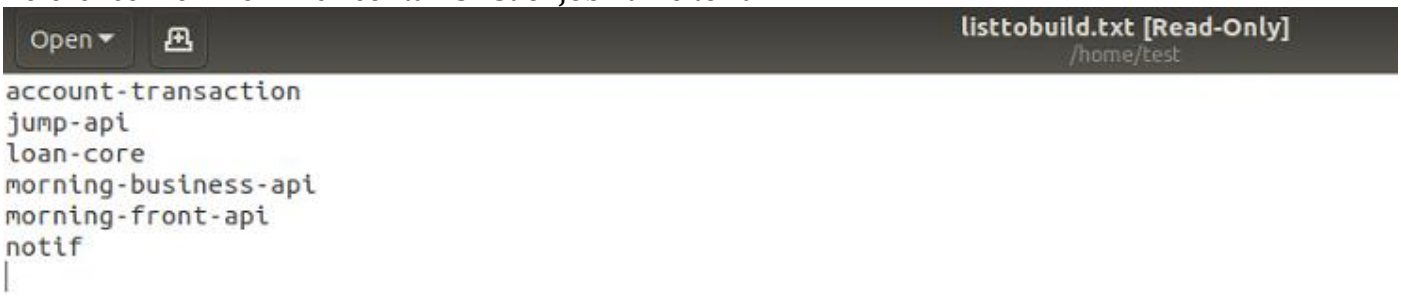
6.5. Job Name: Masterjob_for_deployment

```
node {
  def JOBNAME = readFile (file: '<Provide-File-Path>')// file which contains list of job name to run.
   def failedToRunMicroservice =[]
 echo "${JOBNAME}"

   def microservice_name = JOBNAME.split("\r?\n")
 for (i in microservice_name){
  stage ('started to run microservice')
   {
      echo "${i}"
      try{
       build "${i}"
    echo "${i} is running now"
   }
   catch(Exception e) {
      echo "build fail to run,moving next job to build"
      failedToRunMicroservice.push("${i}")
              }
      }
   }
    if ("${failedToRunMicroservice}"!=0){
    echo "list of microservices that to run: ${failedToRunMicroservice} "
         }

 }
```

Reference file : file which contains list of job name to run.



- **Setup Database -by providing DB script**

6.6. Job Name: Create_MasterDATA_job

```
node {
  def NameOfDbCreate = readFile (file: '<filepath-list-of-name-dbcreate>')
   def trash =[]
 echo "${NameOfDbCreate}"
    stage ('Databases are creating') {

                    sh """
                    export PGPASSWORD=postgres
psql -h 192.168.83.251 -U postgres -p 5432 -a -w -f/home/folder1/queriestocreatedb.sql

                    """
          echo "database is created"
   }
   def microservice_name = NameOfDbCreate.split("\r?\n")

 for (i in microservice_name){

  stage ('Dbscript are going run') {
    try{
                    sh """
            export PGPASSWORD=postgres
//provide host IP here
psql -h 192.168.83.251 -d ${i} -U postgres -p 5432 -a -w -f <dbscripts-stored-path>/${i}.sql
      """

    echo "${i} is running now"
   }
   catch(Exception e) {
      echo "build fail to run,moving next job to build"
      trash.push("${i}")
                    }
            }

   }
    if ("${trash}"!=0){
    echo "list of microservices that to run: ${trash} "
         }

 }
```

- **Gitlab backup with specific branch**

6.7. Job Name: GitLab_Backup

Shell script:

```
 cd /<path-to-takebackup>
for repo in $(curl -s --header "PRIVATE-TOKEN:<Personal-Token-gitlab>"
https://gitlab.com/api/v4/groups/7433233 | jq -r ".projects[].ssh_url_to_repo");
 do
 while read i;
 do git clone --branch "$i"-branch1 git@gitlab.com:it-agora/microservices/"$i".git >> cd  /<path-to-
takebackup>/"$i"/branch1;
done </<servicename-to-clone>.txt
done;
```

Steps:

- Create agora_backup_jan2023 with proper path and provide in code to take all backup in respective folder.
- Create gitlab personal token and apply it for configuration.
- Create jenkins ssh key and paste it into gitlab ssh-keys.