

We need to install plugins

- [CloudBees AWS Credentials](#)
- [Amazon ECR](#)
- [Docker Pipeline](#)

we need set up access key and secret key

[] login to **jenkins dashboard** and select **manage jenkins**

[] select system and scroll down in **Global properties** select **Environment variables**

[] **add**

[] in **name** AWS_ACCESS_KEY_ID (we can give whatever,we use this name in pipeline)

[] in **value** AKIASM6XNBZOR5U2YE7 (I gave access key id what I created in aws)

[] **add**

[] in **name** AWS_SECRET_ACCESS_KEY (we can give whatever,we use this name in pipeline)

[] in **value** vK6JhW+lToii27wLUiDgnBLJnJzxSlayFD97Ect4 (I gave secret access key id what I created in aws)

[] **save**

Global properties

☐ Disable deferred wipeout on this node ?

☒ Environment variables

List of variables ?

Name

AWS_ACCESS_KEY_ID

Value

AKIASM6XNBZOR5U2YE7Q

Name

AWS_SECRET_ACCESS_KEY

Value

vK6JhW+IToii27wLUIDgnBLnJzxSlayFD97Ect4

Add

☐ Tool Locations

Save Apply

EXAMPLE: (I use like this in code)

[] AWS_ACCESS_KEY_ID="\${env.AWS_ACCESS_KEY_ID}"

[] AWS_SECRET_ACCESS_KEY="\${env.AWS_SECRET_ACCESS_KEY}"

1) Code for to push docker image to ECR

pipeline {

agent any

environment {

AWS_ACCESS_KEY_ID="\${env.AWS_ACCESS_KEY_ID}"

AWS_SECRET_ACCESS_KEY="\${env.AWS_SECRET_ACCESS_KEY}"

AWS_REGION=('us-east-1')

}

stages {

stage('Cloning Git') {

steps {

git credentialsId: 'PAT', url:

'https://github.com/kanchana08/Dockerfile_python.git'

```

}
}
stage('Build Docker Image') {
    steps {
        sh "docker build -t lamda_ply:latest ."
    }
}
stage('Publish to ECR') {
    steps {
        script {
            sh "aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin 165271113309.dkr.ecr.us-east-1.amazonaws.com"

            sh "docker tag lamda_ply:latest 165271113309.dkr.ecr.us-east-
1.amazonaws.com/lamda_ply:latest"

            sh "docker push 165271113309.dkr.ecr.us-east-
1.amazonaws.com/lamda_ply:latest"

        }
    }
}
}
}
}

```

=====

2) Code for to push docker image to ECR (I put build function in stage('Publish to ECR'))

```

pipeline {
    agent any
    environment {
        AWS_ACCESS_KEY_ID="${env.AWS_ACCESS_KEY_ID}"
        AWS_SECRET_ACCESS_KEY="${env.AWS_SECRET_ACCESS_KEY}"
        AWS_REGION=('us-east-1')
    }
    stages {
        stage('Cloning Git') {
            steps {
                git credentialsId: 'PAT', url:
                'https://github.com/kanchana08/Dockerfile_python.git'
            }
        }
        stage('Publish to ECR') {
            steps {
                script {
                    sh "aws ecr get-login-password --region us-east-1 | docker login --
                    username AWS --password-stdin 165271113309.dkr.ecr.us-east-1.amazonaws.com"
                    sh "docker build -t lamda_ply ."
                    sh "docker tag lamda_ply:latest 165271113309.dkr.ecr.us-east-
                    1.amazonaws.com/lamda_ply:latest"
                    sh "docker push 165271113309.dkr.ecr.us-east-
                    1.amazonaws.com/lamda_ply:latest"
                }
            }
        }
    }
}

```

```
    }  
  
    }  
  
    }  
}
```

=====

Code for s3 bucket :

```
pipeline{  
  agent any  
  environment {  
    AWS_ACCESS_KEY_ID="${env.AWS_ACCESS_KEY_ID}"  
    AWS_SECRET_ACCESS_KEY="${env.AWS_SECRET_ACCESS_KEY}"
```

```
    AWS_REGION=('us-east-1')
}
stages{
  stage('exm stage'){
    steps {
      script{

        sh"aws s3 ls"

      }
    }
  }
}
```

=====

Deleted previous latest tag images first and then push new images

```

pipeline {
    agent any
    environment {
        AWS_ACCESS_KEY_ID="${env.AWS_ACCESS_KEY_ID}"
        AWS_SECRET_ACCESS_KEY="${env.AWS_SECRET_ACCESS_KEY}"
        // AWS_REGION=('us-east-1')
        AWS_REGION="${env.AWS_DEFAULT_REGION}"
    }
    stages {
        stage('Cloning Git') {
            steps {
                git credentialsId: 'PAT', url:
                'https://github.com/kanchana08/Dockerfile_python.git'
            }
        }
        stage('Build Docker Image') {
            steps {
                // script {
                // dockerImage = docker.build "lamda_ply:latest"

                // }
                sh "docker build -t lamda_ply:latest ."
            }
        }
        stage('Publish to ECR') {
            steps {

```

```

    script {
        sh "aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin 165271113309.dkr.ecr.us-east-1.amazonaws.com"

        sh" aws ecr batch-delete-image --repository-name lamda_ply --image-
ids imageTag=latest"

        //sh" aws ecr batch-delete-image --repository-name lamda_ply --image-
ids imageTag=latest"

        sh"docker tag lamda_ply:latest 165271113309.dkr.ecr.us-east-
1.amazonaws.com/lamda_ply:latest"

        sh "docker push 165271113309.dkr.ecr.us-east-
1.amazonaws.com/lamda_ply:latest"

    }

}

}

}

}

}

```

=====

Update the ecr image to lambda function

[] select create lambda and select container image and past ecr uri

```
[] sh "aws lambda update-function-code --region ${AWS_DEFAULT_REGION} -
-function-name ${LAMBDA_FUNCTION_NAME} --image-uri
```



```
${AWS_ACCOUNT_ID}.dkr.ecr.${AWS_DEFAULT_REGION}.amazonaws.com/${IMAGE_
REPO_NAME}:${IMAGE_TAG}"
```

[] We need to install plugins

- 2) AWS lambda plugin
- 3) Amazon EC2
- 4) CloudBees AWS Credentials Plugin

```
pipeline {
  agent any
  environment {
    AWS_ACCESS_KEY_ID="${env.AWS_ACCESS_KEY_ID}"
    AWS_SECRET_ACCESS_KEY="${env.AWS_SECRET_ACCESS_KEY}"
    AWS_REGION=('us-east-1')
  }
  stages {
    stage('Cloning Git') {
      steps {
        git credentialsId: 'PATH', url:
'https://github.com/kanchana08/Dockerfile_python.git'
      }
    }
  }
}
```

```
stage('Build Docker Image') {
  steps {
    sh "docker build -t playwright_123:latest ."

  }

}

stage('Publish to ECR') {
  steps {
    script {

      sh "aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin 909100690382.dkr.ecr.us-east-1.amazonaws.com"

      //sh" aws ecr batch-delete-image --repository-name playwright_123 --
image-ids imageTag=latest"

      sh" aws ecr batch-delete-image --repository-name playwright_123 --
image-ids imageTag=latest"

      sh "docker tag playwright_123:latest 909100690382.dkr.ecr.us-east-
1.amazonaws.com/playwright_123:latest"

      sh "docker push 909100690382.dkr.ecr.us-east-
1.amazonaws.com/playwright_123:latest"

    }

  }

}
```

```

stage('lambda_function'){
    steps{
        script{
sh'aws lambda update-function-code --region us-east-1 --function-name
lambdafunction --image-uri 909100690382.dkr.ecr.us-east-
1.amazonaws.com/playwright_123:latest'

        }
    }
}
}

}

```

=====

With output stage

```

pipeline {
    agent any
    environment {
        AWS_ACCESS_KEY_ID="${env.AWS_ACCESS_KEY_ID}"
        AWS_SECRET_ACCESS_KEY="${env.AWS_SECRET_ACCESS_KEY}"
        AWS_REGION=('us-east-1')

    }
    stages {
        stage('Cloning Git') {
            steps {

```

```

        git credentialsId: 'PATH', url:
'https://github.com/kanchana08/Dockerfile_python.git'

    }

}

stage('Build Docker Image') {
    steps {
        sh "docker build -t playwright_123:latest ."

    }

}

stage('Publish to ECR') {
    steps {
        script {

            sh "aws ecr get-login-password --region us-east-1 | docker login --
username AWS --password-stdin 909100690382.dkr.ecr.us-east-1.amazonaws.com"

            //sh" aws ecr batch-delete-image --repository-name playwright_123 --
image-ids imageTag=latest"

            sh" aws ecr batch-delete-image --repository-name playwright_123 --
image-ids imageTag=latest"

            sh "docker tag playwright_123:latest 909100690382.dkr.ecr.us-east-
1.amazonaws.com/playwright_123:latest"

            sh "docker push 909100690382.dkr.ecr.us-east-
1.amazonaws.com/playwright_123:latest"

```

```

    }

}

stage('lambda_function'){
    steps{
        script{
sh'aws lambda update-function-code --region us-east-1 --function-name
lambdafunction --image-uri 909100690382.dkr.ecr.us-east-
1.amazonaws.com/playwright_123:latest'

        }
    }
}

stage('lambda_test_function'){
    steps{
        sh' aws lambda invoke --function-name lambdafunction out --log-type Tail'

    }
}

```

}

}