

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

The screenshot shows a web interface for managing tool locations. It features a form with two input fields: 'Name' and 'Value'. The 'Name' field contains the text 'AWS\_SECRET\_ACCESS\_KEY' and the 'Value' field contains a long alphanumeric string 'vk6JhW+IToii27wLUIDgnBLjnJzSlayFD97Ect4'. Below the form is an 'Add' button and a checkbox labeled 'Tool Locations'. At the bottom of the page, there are 'Save' and 'Apply' buttons.

**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 **paste 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 testing lambda functions 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'

    }
}

}

}

```

=====

### **With testing lambda functions 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'  
  
  }  
}  
  
}
```

**After testing the lambda functions, AWS CloudWatch log groups created automatically.**

**Jenkins output:**



Dashboard > ECR >

Status

</> Changes

▶ Build Now

⚙️ Configure

🗑️ Delete Pipeline

🔍 Full Stage View

✎ Rename

❓ Pipeline Syntax

Build History

trend

Filter builds...

#40

25.Sen.2023.10:25

#39

25.Sen.2023.05:47

#38

22.Sen.2023.11:59

#37

22.Sen.2023.11:52

#36

22.Sen.2023.11:56

#35

22.Sen.2023.11:52

#34

22.Sen.2023.11:59

Pipeline ECR

Add description

Disable Project

Stage View

Average stage times:  
(Average full run time: ~36s)

	Cloning Git	Build Docker Image	Publish to ECR	lambda_function	lambda_test_function
#40 Sept 23 15:55 1 commit	1s	9s	6s	1s	15s
#39 Sept 25 11:17 No Changes	1s	10s	7s	2s	15s
#38 Sept 22 17:29 1 commit	1s	14s	9s	2s	15s
#37 Sept 22 17:27 No Changes	921ms	14s	8s	2s	15s

## ECR image

RWS

Services

Search

[Alt+S]

N. Virginia

slvaroking

EC2

Amazon Elastic Container Registry

Private registry

Public registry

Repositories

Summary

Images

Permissions

Lifecycle Policy

Repository tags

ECR public gallery

Amazon ECS

Amazon ECR > Repositories > playwright\_123

playwright\_123

View push commands

Edit

Images (1)

Search artifacts

< 1 >

	Image tag	Artifact type	Pushed at	Size (MB)	Image URI	Digest	Scan status
<input type="checkbox"/>	latest	Image	September 25, 2023, 15:55:59 (UTC+05.5)	34.70	Copy URI	sha256:cafb88e3...	-

## Lambda functions

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with 'WS', 'Services', a search bar, and a keyboard shortcut '[Alt+S]'. Below the navigation bar, the breadcrumb trail is 'Lambda > Functions > lambdafunction'. The main header area displays 'lambdafunction' with buttons for 'Throttle', 'Copy ARN', and 'Actions'. The 'Function overview' tab is active, showing a diagram of the function with an 'Add trigger' button. To the right, a metadata panel lists: Description (-), Last modified yesterday, Function ARN 'arn:aws:lambda:us-east-1:909100690382:function:lambdafunction', and Function URL with an 'Info' link. Below the overview, there are tabs for 'Image', 'Test', 'Monitor', 'Configuration', 'Aliases', and 'Versions'. The 'Image' tab is selected, showing a message: 'No code preview available. Your function code is deployed as a container image. The AWS Cloud9 IDE cannot display your code.' It also displays the 'Image URI' '909100690382.dkr.ecr.us-east-1.amazonaws.com/playwright\_123:latest' and 'Architecture' 'x86\_64'. At the bottom, the 'Image configuration' tab is visible, showing 'ENTRYPOINT override' (-), 'CMD override' (-), and 'WORKDIR override' (-), with an 'Edit' button.

## Cloud watch log:

The screenshot shows the AWS CloudWatch console. On the left is a sidebar with navigation options: 'CloudWatch', 'Favorites and recents', 'Dashboards', 'Alarms', 'In alarm', 'All alarms', 'Billing', 'Logs', 'Log groups' (selected), 'Live Tail', 'Logs Insights', 'Metrics', 'All metrics', 'Explorer', 'Streams', 'X-Ray traces', 'Events', 'Application monitoring', 'Insights', and 'Settings'. The main content area has a breadcrumb trail 'CloudWatch > Log groups > /aws/lambda/lambdafunction'. The header shows the path '/aws/lambda/lambdafunction' and buttons for 'Actions', 'View in Logs Insights', 'Start tailing', and 'Search log group'. The 'Log group details' section is expanded, displaying a table with the following information: ARN 'arn:aws:logs:us-east-1:909100690382:log-group:/aws/lambda/lambdafunction:\*', Stored bytes '1.63 KB', Contributor Insights rules (-), Creation time '4 days ago', Metric filters '0', KMS key ID (-), Retention 'Never expire', Subscription filters '0', Data protection (-), and Sensitive data count (-). Below this, there are tabs for 'Log streams', 'Tags', 'Metric filters', 'Subscription filters', 'Contributor Insights', and 'Data protection'. The 'Log streams' tab is active, showing a search bar 'Filter log streams or try prefix search' and buttons for 'Exact match', 'Show expired', 'Info', 'Create log stream', and 'Search all log streams'. A table lists log streams, with one entry: '2023/09/25/[\$LATEST]1cb2fb7ca40d48c8a83787c6e18d78b7' with a 'Last event time' of '2023-09-25 15:56:16 (UTC+05:30)'.