**Docker image to AWS ECR:**

**[] we need create ECR and IAM users and iam police permission and iam we need create access key**

**To create access key and secret key**

[] in iam

[] select users

[] select which user want

[] select security credentials

[] scroll down

[] select **create access key**

[] select **command line interface**

[] select confirm

[] next

[] type description

[] create access key

[] downloadthe .csv there we get keys / we can copy and paste the keys

**we need give ECR full access permission for docker push**

[]Open the AWS Management Console and navigate(search) to the IAM service.  
  
[] Locate and select the IAM user to which you want to attach the policy.  
  
[] In the user scroll down to the **"Permissions"** section.  
  
 [] in **“add permissions”** Click on the **"Add inline policy"** button .  
  
 In the policy editor, choose the **"JSON"** tab to enter the policy code.  
  
 Replace the existing policy code with the JSON code provided earlier

{"Version": "2012-10-17",

"Statement": [

{

"Effect": "Allow",

"Action":

["ecr:\*"],

"Resource": "\*"}]}

[] next

[] Provide a name for the policy in the **"Name"** field.  
[] Click on **"Review policy"** to verify the policy details.  
[]Finally, click on **"Create policy"** or **"Attach policy"** to attach the policy to the IAM user or role

**------------------------------------------------------------------------------------------------------**

**In server**

**[] sudo su -**

**[] apt-get update**

**[] apt-get install awscli**

**[] apt install awscli**

**[] apt install dnf**

We need to install plugins

* [CloudBees AWS Credentials](https://plugins.jenkins.io/aws-credentials/)
* [Amazon ECR](https://plugins.jenkins.io/amazon-ecr/)
* [Docker Pipeline](https://plugins.jenkins.io/docker-workflow/)

we need set up access key and secret key

**[]** login to j**enkins 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+lToii27wLUiDgnBLJnJzxSIayFD97Ect4 (I gave secret access key id what I created in aws )

**[]** save





**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**

1) AWS lambda plugin

1. Amazon EC2
2. 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'

}

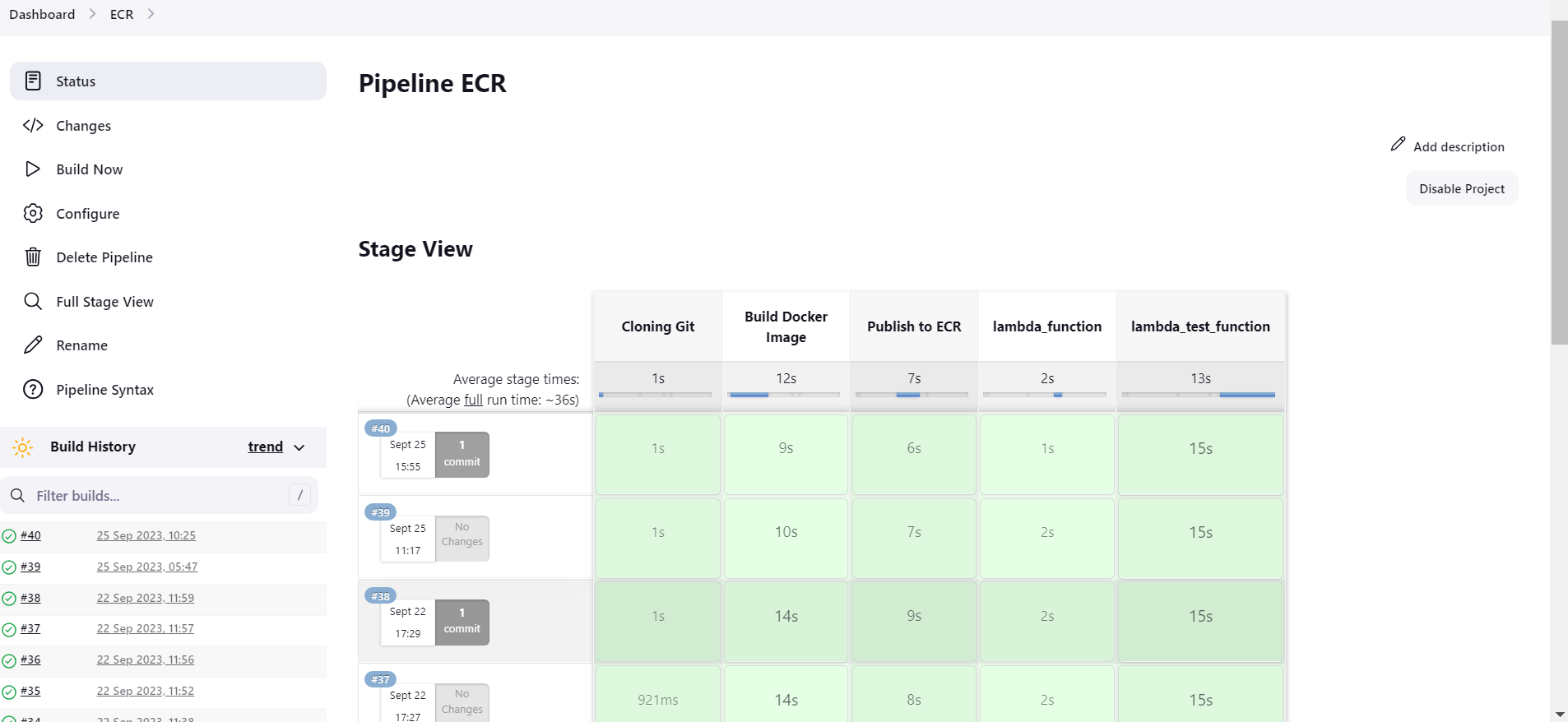
}

}

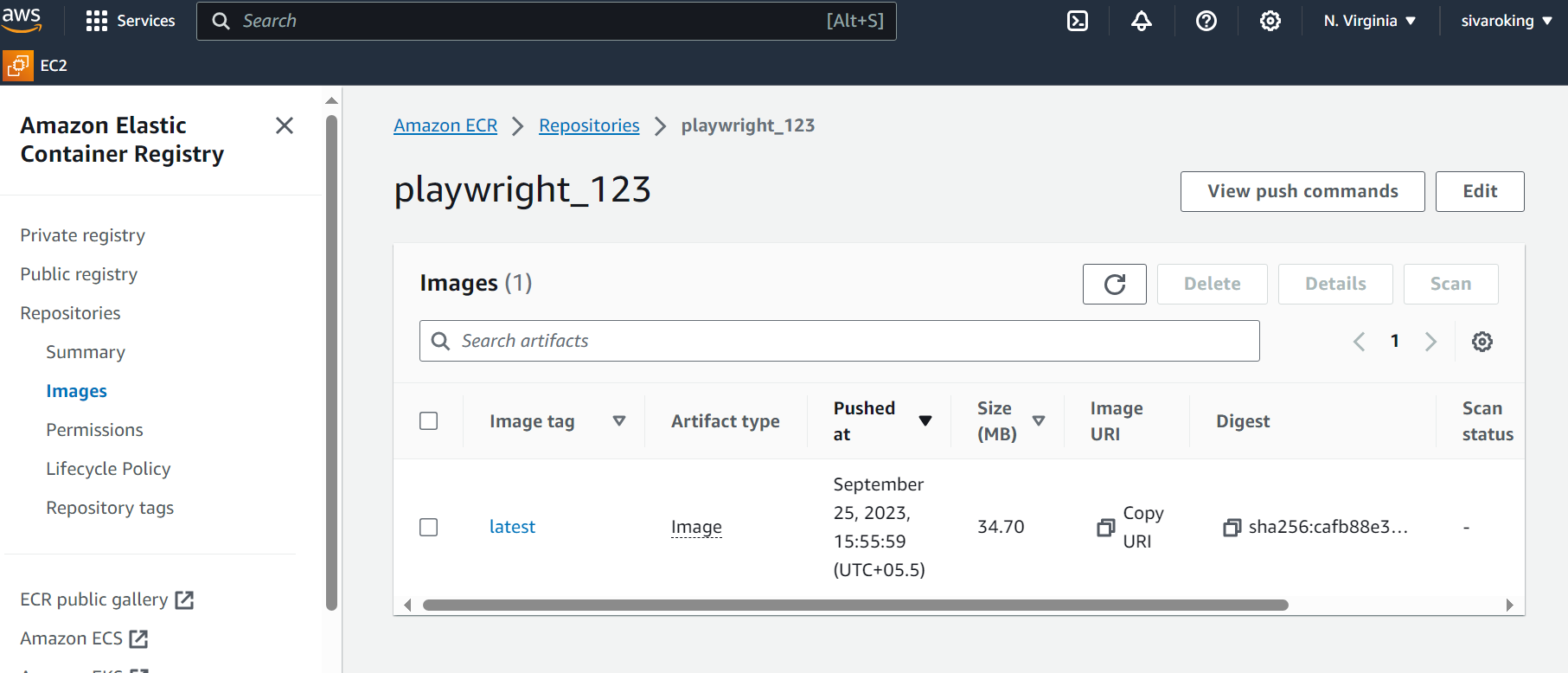
}

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

**Jenkins output:**

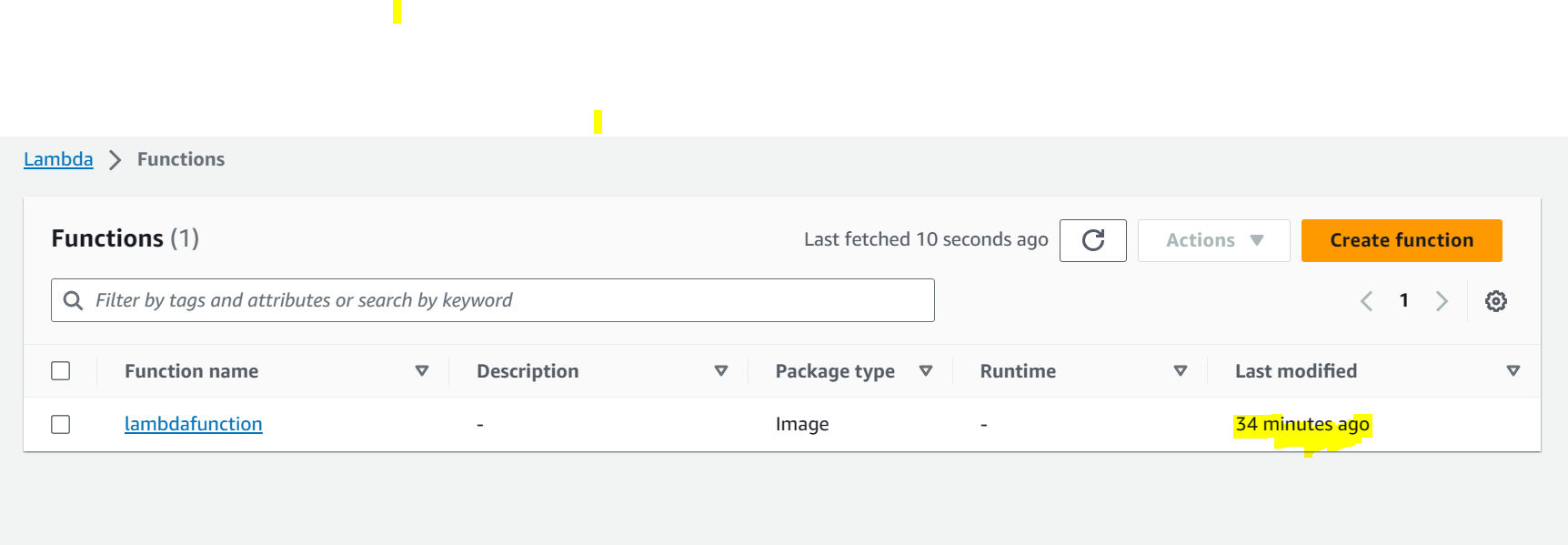


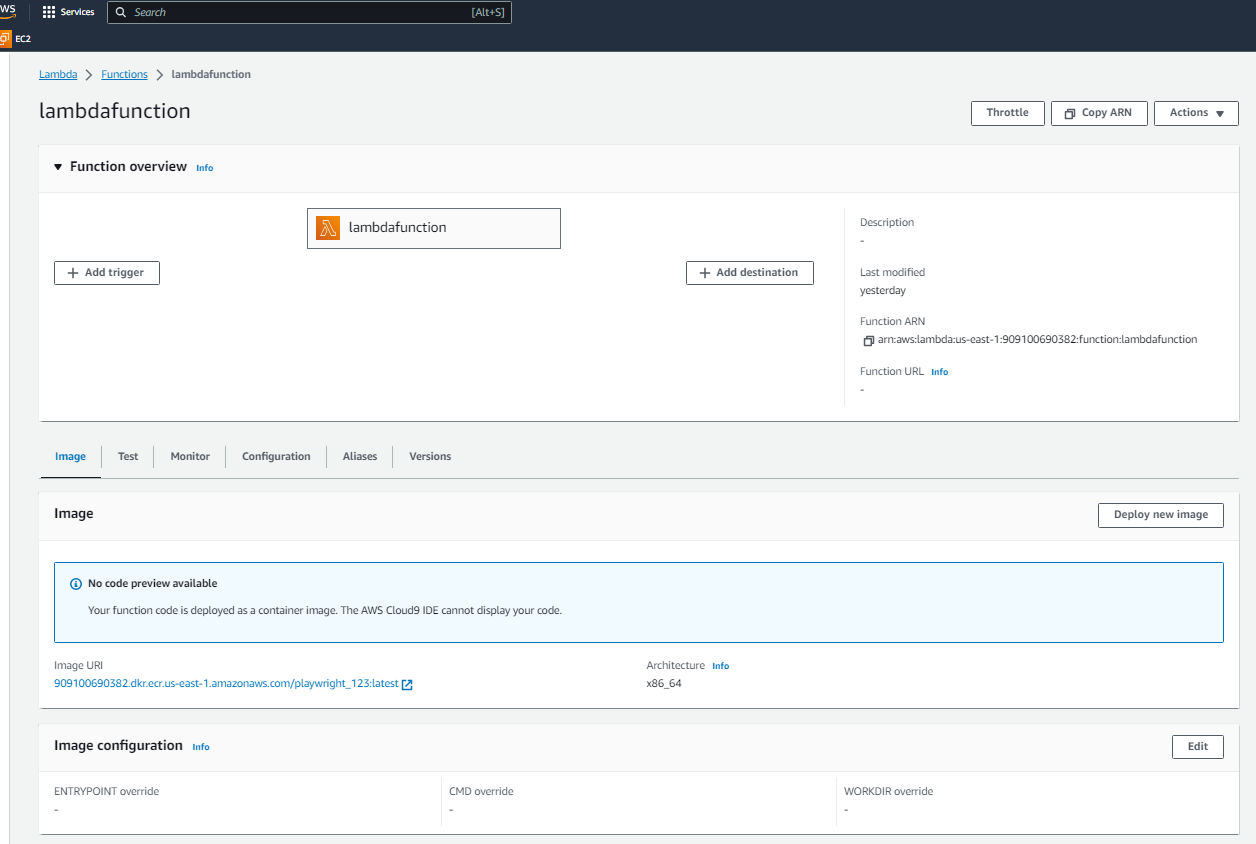
**ECR image**



**Lambda functions**

**Latest image will update in lambda function**





**Cloud watch log:**

