## Requirements:

**In AWS**

**1)** Create **user**

[] go to **IAM**, select **user** and select **create user**, give **username**

**for that user create inline policy .**

[] select user.

[] 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

**For that user create access key and secrete access key (AWS CLI)**

[] select security credentials

[] scroll down

[] select **create access key**

[] select **command line interface**

[] select confirm

[] next

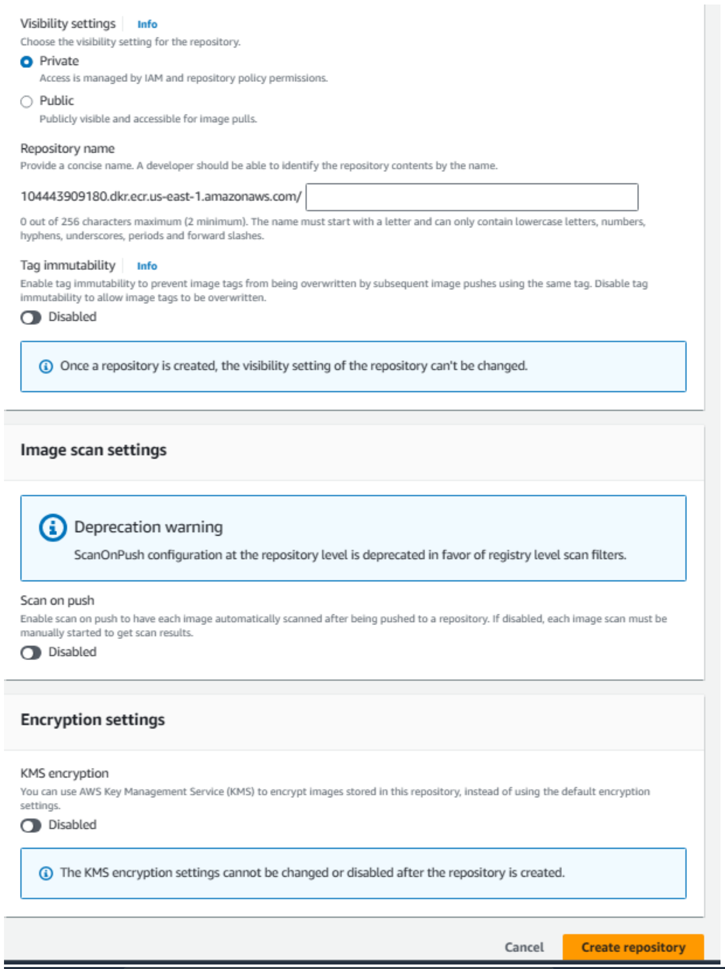
[] type description

[] create access key

[] download the .csv there we get keys

**2)**create **ECR**

**[]** go to ECR, select create repository and mention repo name.


Visibility settings Info

Choose the visibility setting for the repository.

[✓ Private]
Access is managed by IAM and repository policy permissions.

Public
Publicly visible and accessible for image pulls.

Repository name

Provide a concise name. A developer should be able to identify the repository contents by the name.

104443909180.dkr.ecr.us-east-1.amazonaws.com/

0 out of 256 characters maximum (2 minimum). The name must start with a letter and can only contain lowercase letters, numbers, hyphens, underscores, periods and forward slashes.

Tag immutability Info

Enable tag immutability to prevent image tags from being overwritten by subsequent image pushes using the same tag. Disable tag immutability to allow image tags to be overwritten.

Disabled

Once a repository is created, the visibility setting of the repository can't be changed.

Image scan settings

Deprecation warning

ScanOnPush configuration at the repository level is deprecated in favor of registry level scan filters.

Scan on push

Enable scan on push to have each image automatically scanned after being pushed to a repository. If disabled, each image scan must be manually started to get scan results.

Disabled

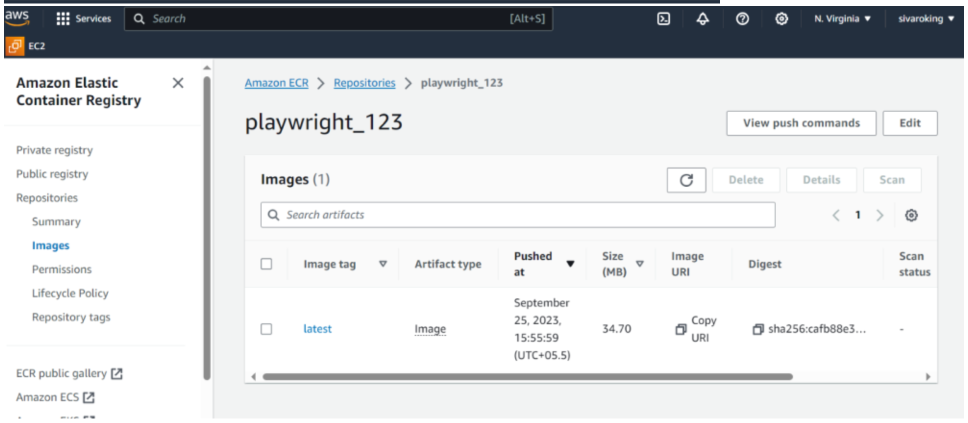
Encryption settings

KMS encryption

You can use AWS Key Management Service (KMS) to encrypt images stored in this repository, instead of using the default encryption settings.

Disabled

The KMS encryption settings cannot be changed or disabled after the repository is created.



playwright_123

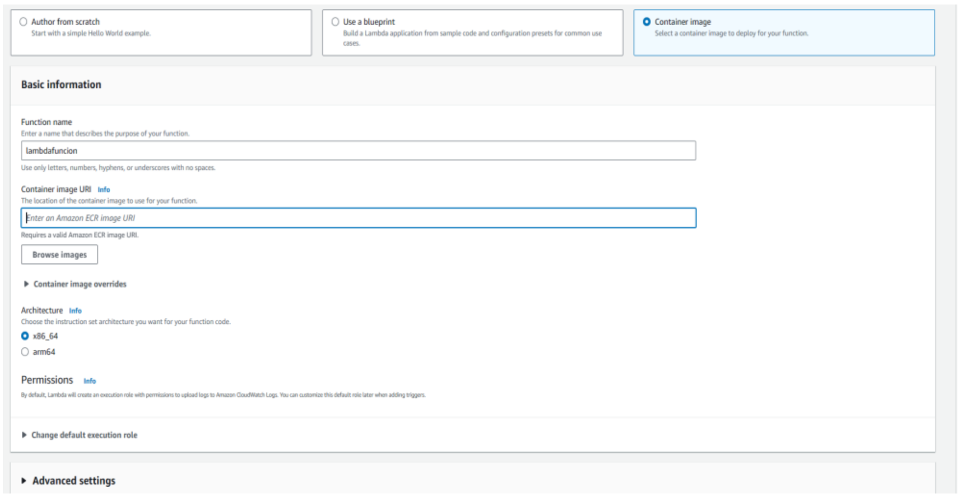
Images (1)

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

## 3)create lambda function.

**[] to create lambda function we need ECR image URI, then only we can create lambda functions.**

[] go to **lambda function** and select **create function**
[] select **container image**.
[] **function name**.
[] copy and paste **ECR image URI,** (ECR repo URI is different and ECR image URI is different)



Author from scratch [✓ Container image]
Use a blueprint

**Basic information**

Function name
Enter a name that describes the purpose of your function.
lambdafuncion
Use only letters, numbers, hyphens, or underscores with no spaces.

Container image URI
The location of the container image to use for your function.
*Enter an Amazon ECR image URI*
Requires a valid Amazon ECR image URI.

Architecture
Choose the instruction set architecture you want for your function code.
[✓ x86_64] arm64

Permissions
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

Start with a simple Hello World example.
Build a Lambda application from sample code and configuration presets for common use cases.
Select a container image to deploy for your function.
Info
Browse images
Container image overrides
Change default execution role
Advanced settings

---------------------------------------------------------------------------------------------------------

## <u>In server:</u>

**[]** install **Jenkins**, **Docker**, **GIT** and install **AWSCLI** install version 2 (**apt-get install awscli).**

**[]** check version

**[] chmod 777 /var/run/docker.sock**

**[] aws configure ------>** to give AWS access and secret key in server and region.

---------------------------------------------------------------------------------------------------------

## In Jenkins:

[] add **GitHub** credentials.

username and PAT

[] add **AWS access and secrete access key**.

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

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

**[] add**

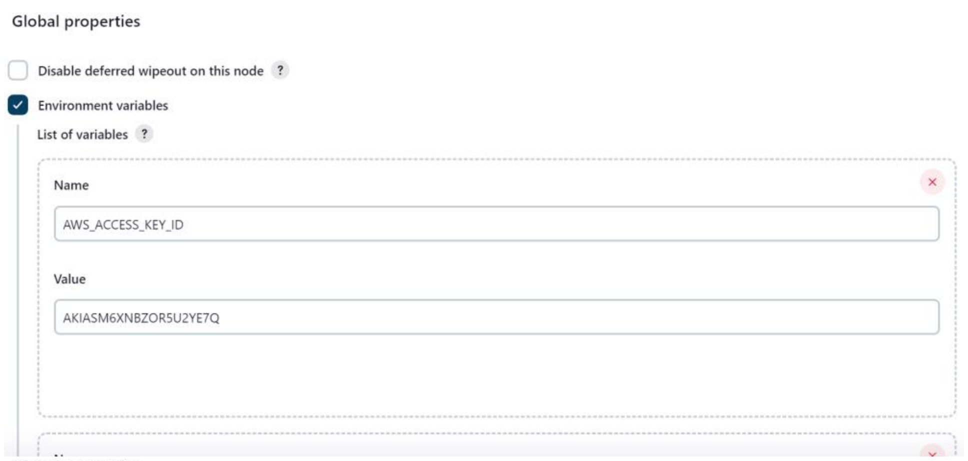
**[]** in **name** AWS_ACCESS_KEY_ID (we can give whatever)

**[]** in **value** AKIASM6XNBZOR5U2YE7 (I give access key id)

**[]** add

**[]** in **name** AWS_SECRET_ACCESS_KEY )

**[]** in **value** vK6JhW+lToii27wLUiDgnBLJnJzxSIayFD97Ect4 (I give secret access key id what I created in aws )


Global properties

- [ ] Disable deferred wipeout on this node
- [✓] Environment variables

List of variables

Name: AWS_ACCESS_KEY_ID

Value: AKIASM6XNBZOR5U2YE7Q

Global properties

- [ ] Disable deferred wipeout on this node
- [✓] Environment variables

List of variables

Name: AWS_ACCESS_KEY_ID

Value: AKIASM6XNBZOR5U2YE7Q

**Install plugins: (related plugins)**

- AWS Credentials

- Amazon ECR
- Docker Pipeline
- AWS lambda plugin
- Amazon EC2
- Docker
- GitHub

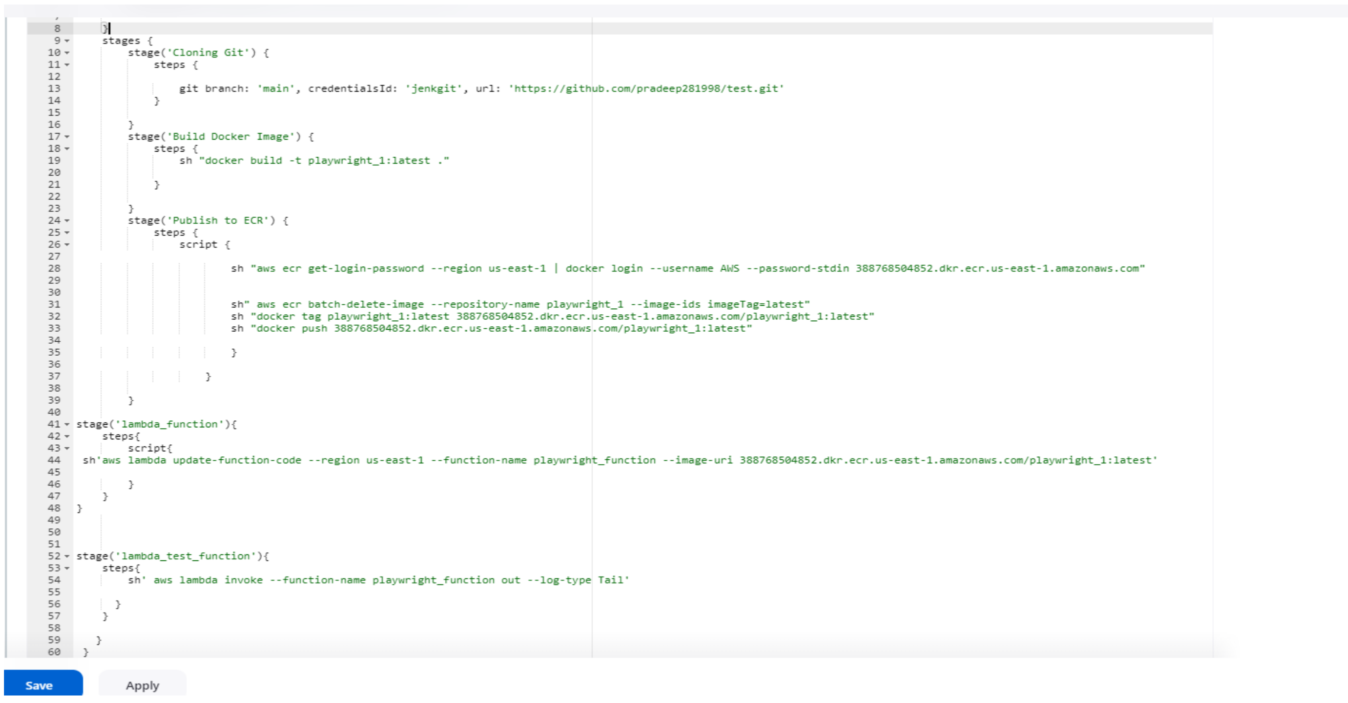
## <u>**Pipeline code**</u>

Stage 1: git clone
Stage 2: docker image build

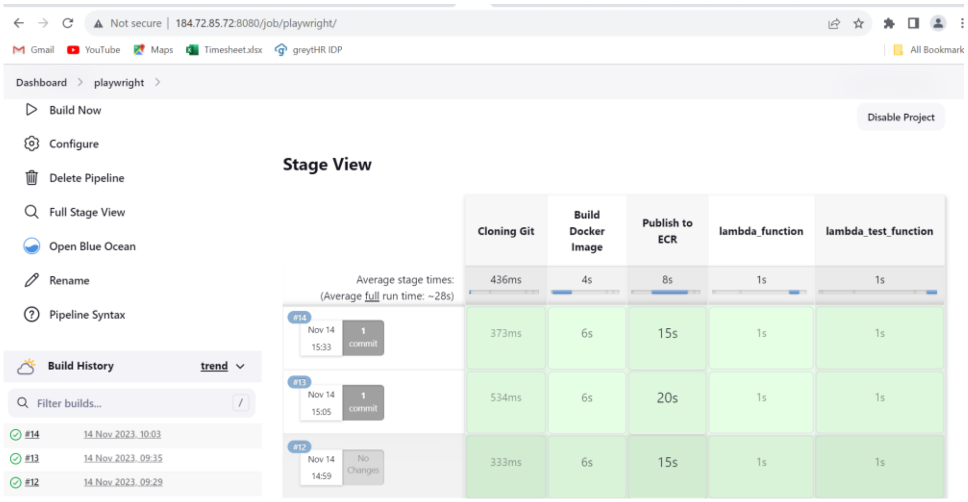
Stage 3: ECR login and push docker image to ECR repo.

Stage 4: pushing ERC image to lambda function.

Stage 5: lambda test and AWS CloudWatch log creation.



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

                git branch: 'main', credentialsId: 'jenkgit', url: 'https://github.com/pradeep281998/test.git'
            }

        }
        stage('Build Docker Image') {
            steps {
                sh "docker build -t playwright_1:latest ."

            }

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

                        sh "aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 388768504852.dkr.ecr.us-east-1.amazonaws.com"


                        sh" aws ecr batch-delete-image --repository-name playwright_1 --image-ids imageTag=latest"
                        sh "docker tag playwright_1:latest 388768504852.dkr.ecr.us-east-1.amazonaws.com/playwright_1:latest"
                        sh "docker push 388768504852.dkr.ecr.us-east-1.amazonaws.com/playwright_1:latest"

                        }

                    }

        }

stage('lambda_function'){
    steps{
        script{
 sh'aws lambda update-function-code --region us-east-1 --function-name playwright_function --image-uri 388768504852.dkr.ecr.us-east-1.amazonaws.com/playwright_1:latest'

        }
    }
}



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

      }
    }

   }
 }
```



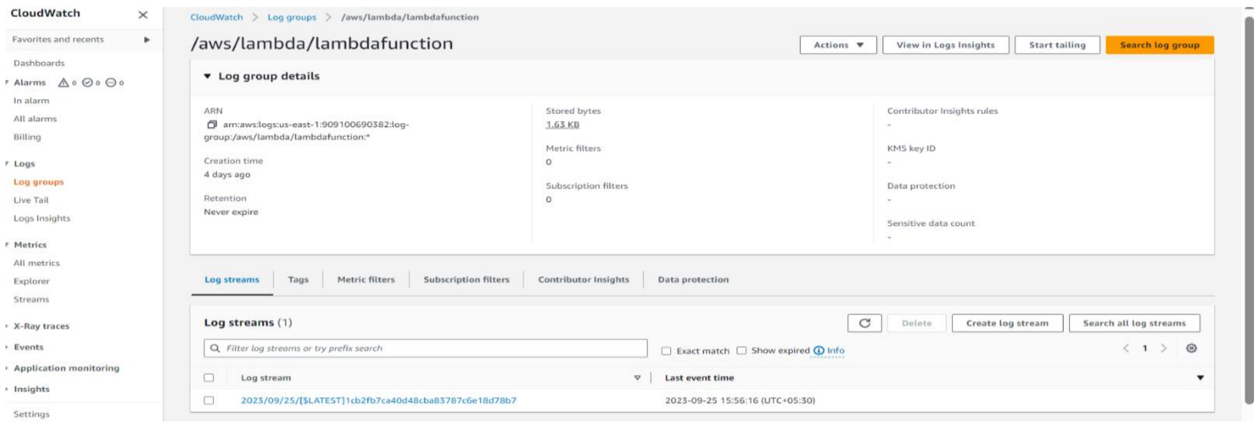
#14 14 Nov 2023, 10:03
#13 14 Nov 2023, 09:35
#12 14 Nov 2023, 09:29

Stage View

| | Cloning Git | Build Docker Image | Publish to ECR | lambda_function | lambda_test_function |
|---|---|---|---|---|---|
| Average stage times: (Average full run time: ~28s) | 436ms | 4s | 8s | 1s | 1s |
| #14 Nov 14 15:33 1 commit | 373ms | 6s | 15s | 1s | 1s |
| #13 Nov 14 15:05 1 commit | 534ms | 6s | 20s | 1s | 1s |
| #12 Nov 14 14:59 No Changes | 333ms | 6s | 15s | 1s | 1s |

## In AWS:

After building the pipeline code we will get AWS CloudWatch log.



/aws/lambda/lambdafunction

Log group details

| | | |
|---|---|---|
| ARN<br>arn:aws:logs:us-east-1:909100690382:log-group:/aws/lambda/lambdafunction:* | Stored bytes<br>1.63 KB | Contributor Insights rules<br>- |
| Creation time<br>4 days ago | Metric filters<br>0 | KMS key ID<br>- |
| Retention<br>Never expire | Subscription filters<br>0 | Data protection<br>- |
| | | Sensitive data count<br>- |

Log streams (1)

| Log stream | Last event time |
|---|---|
| 2023/09/25/[$LATEST]1cb2fb7ca40d48cba83787c6e18d78b7 | 2023-09-25 15:56:16 (UTC+05:30) |