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

[] download the .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
- Amazon ECR
- Docker Pipeline

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)

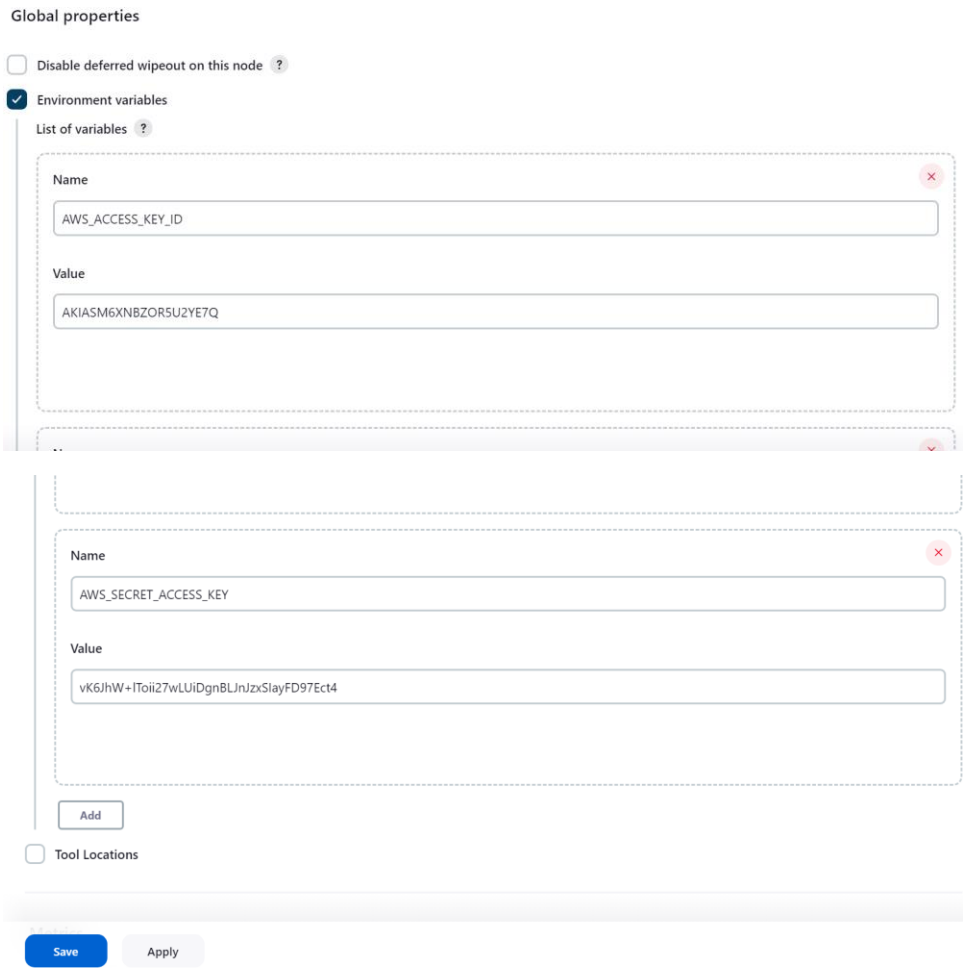
[] in **value** AKIASM6XNBZOR5U2YE7 (I gave access key id what I created in aws)

[] 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) <u>Code for to push docker image to ECR</u> (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

2) Amazon EC2

3) 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

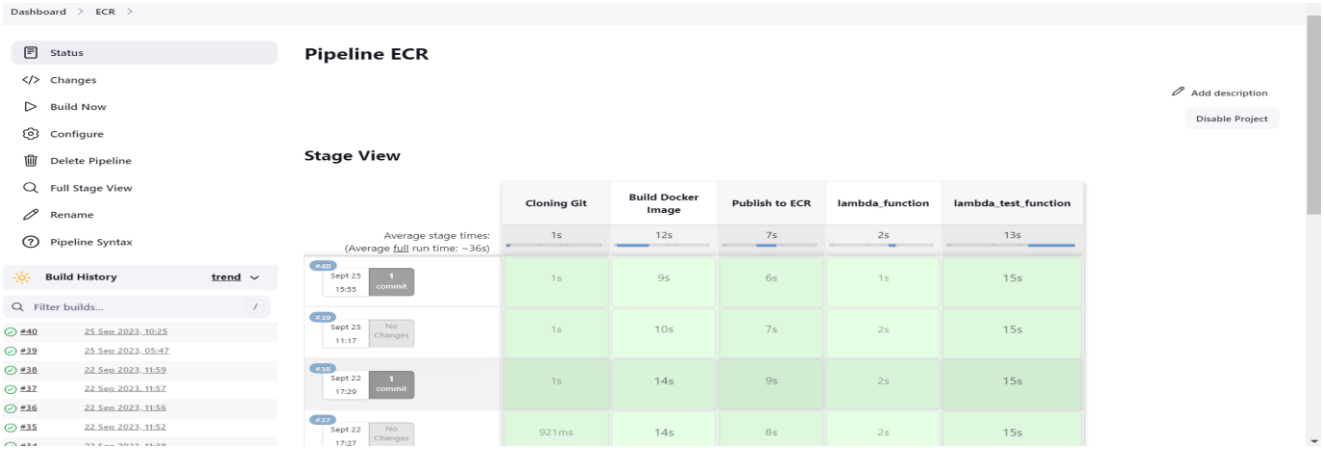
```
pipeline {
   agent any
   environment {
      AWS_ACCESS_KEY_ID="${env.AWS_ACCESS_KEY_ID}"
      AWS_SECRET_ACCESS_KEY="${env.AWS_SECRET_ACCESS_KEY}"
      AWS_REGION=('us-east-1')

   }
   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"
```

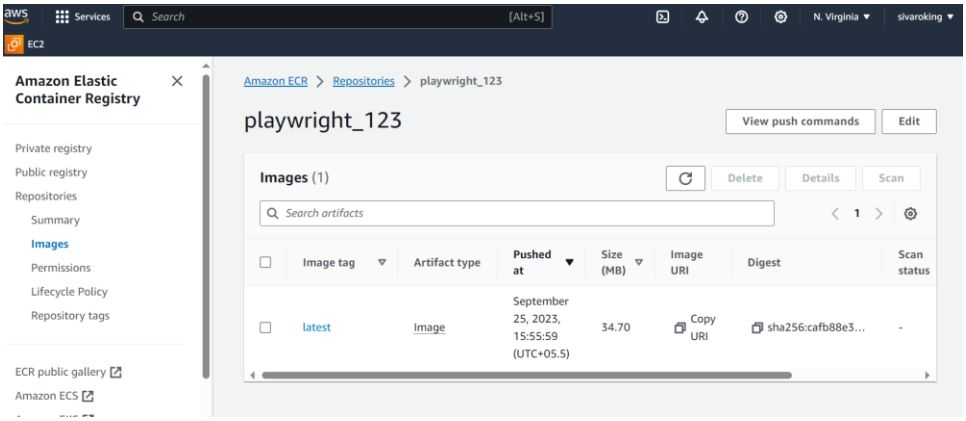
```
                }

            }

    }

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.

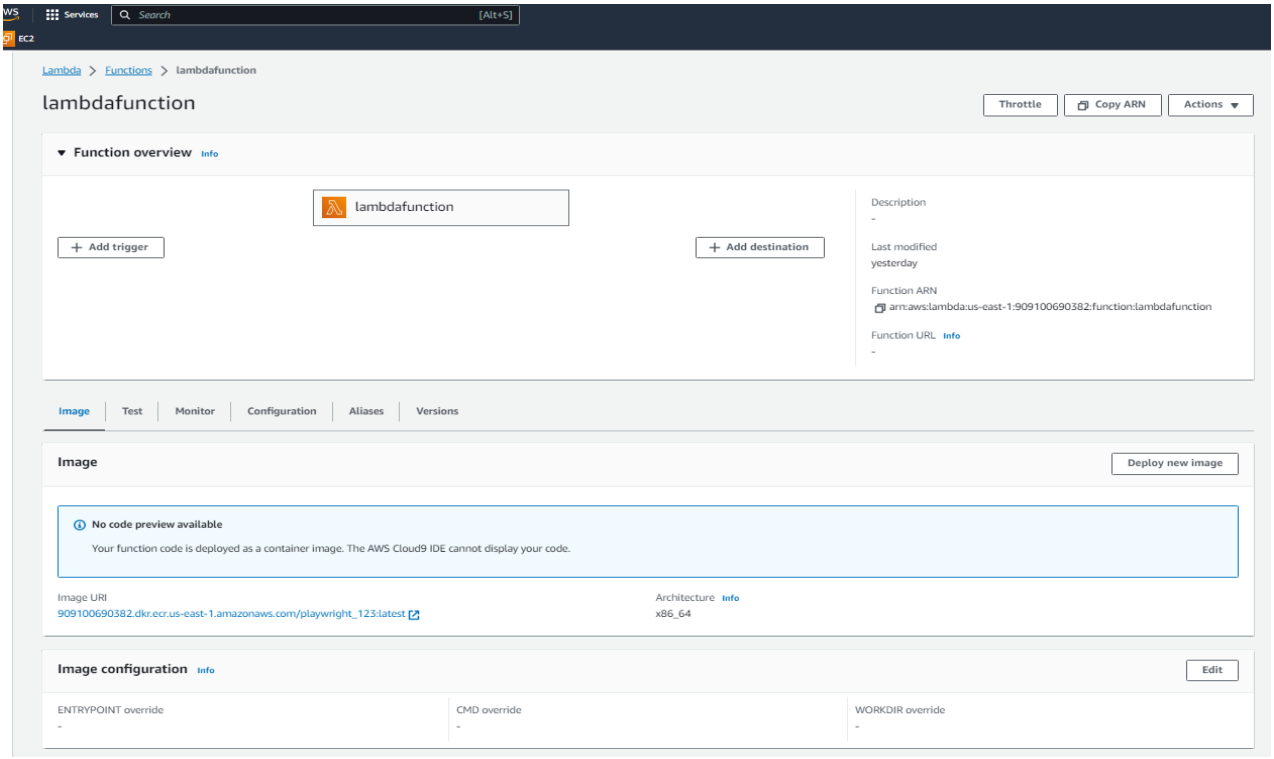
Jenkins output:



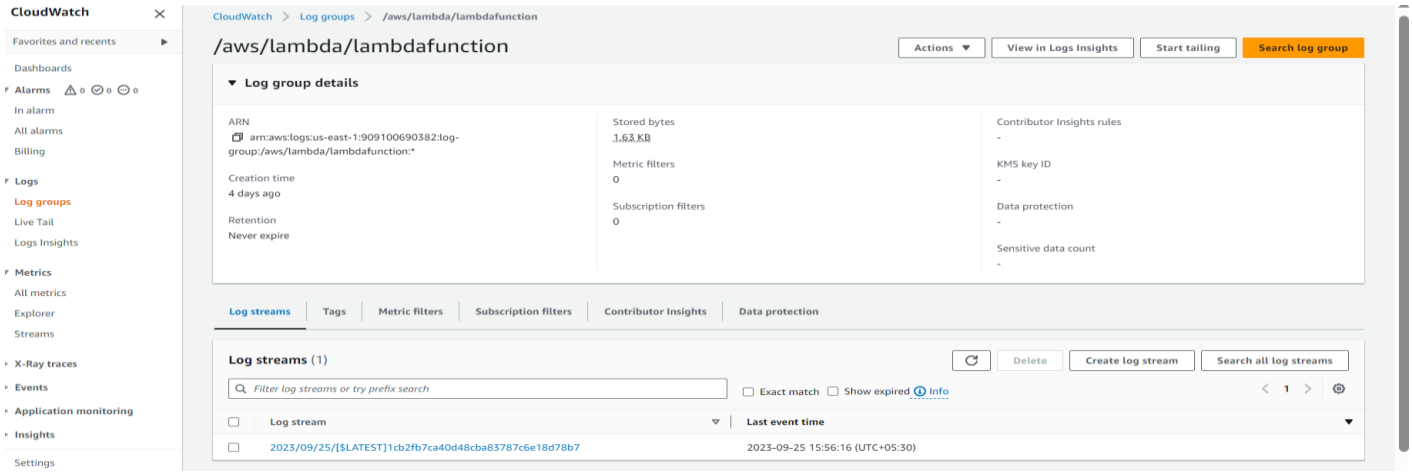
ECR image



Lambda functions



Cloud watch log:


CloudWatch
Favorites and recents
Dashboards
Alarms 0 0 0
In alarm
All alarms
Billing
Logs
Log groups
Live Tail
Logs Insights
Metrics
All metrics
Explorer
Streams
X-Ray traces
Events
Application monitoring
Insights
Settings
CloudWatch > Log groups > /aws/lambda/lambdafunction
/aws/lambda/lambdafunction
Actions
View in Logs Insights
Start tailing
Search log group
Log group details
ARN
arn:aws:logs:us-east-1:909100690382:log-group:/aws/lambda/lambdafunction:*
Creation time
4 days ago
Retention
Never expire
Stored bytes
1.63 KB
Metric filters
0
Subscription filters
0
Contributor Insights rules
-
KMS key ID
-
Data protection
-
Sensitive data count
-
Log streams
Tags
Metric filters
Subscription filters
Contributor Insights
Data protection
Log streams (1)
Delete
Create log stream
Search all log streams
Filter log streams or try prefix search
Exact match
Show expired
Info
1
Log stream
Last event time
2023/09/25/[$LATEST]1cb2fb7ca40d48cba83787c6e18d78b7
2023-09-25 15:56:16 (UTC+05:30)