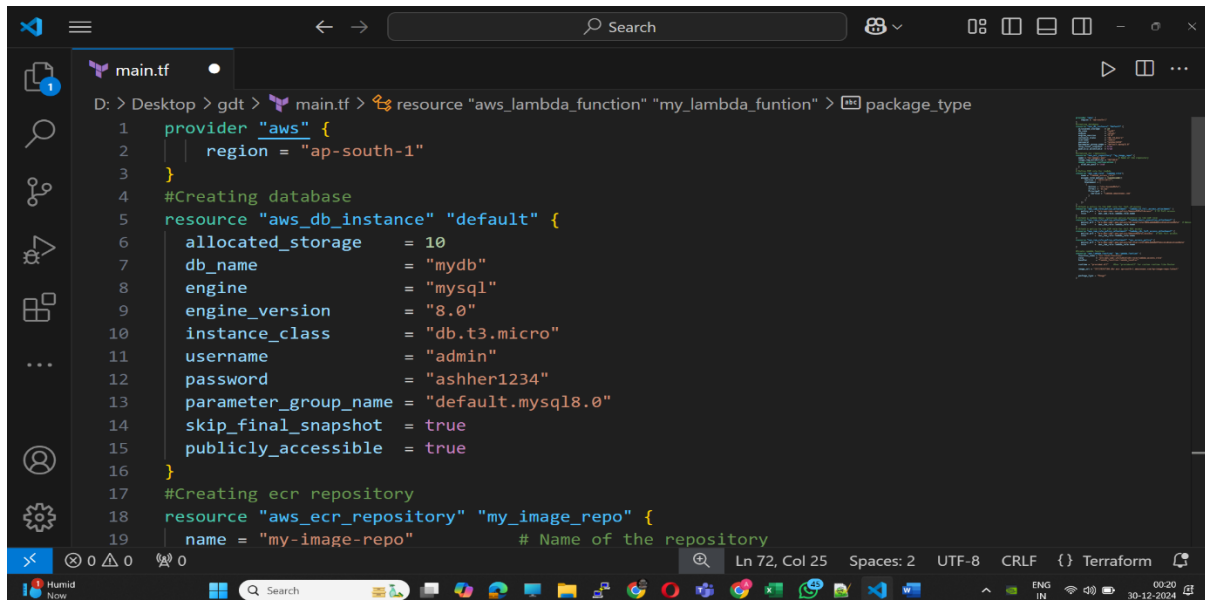
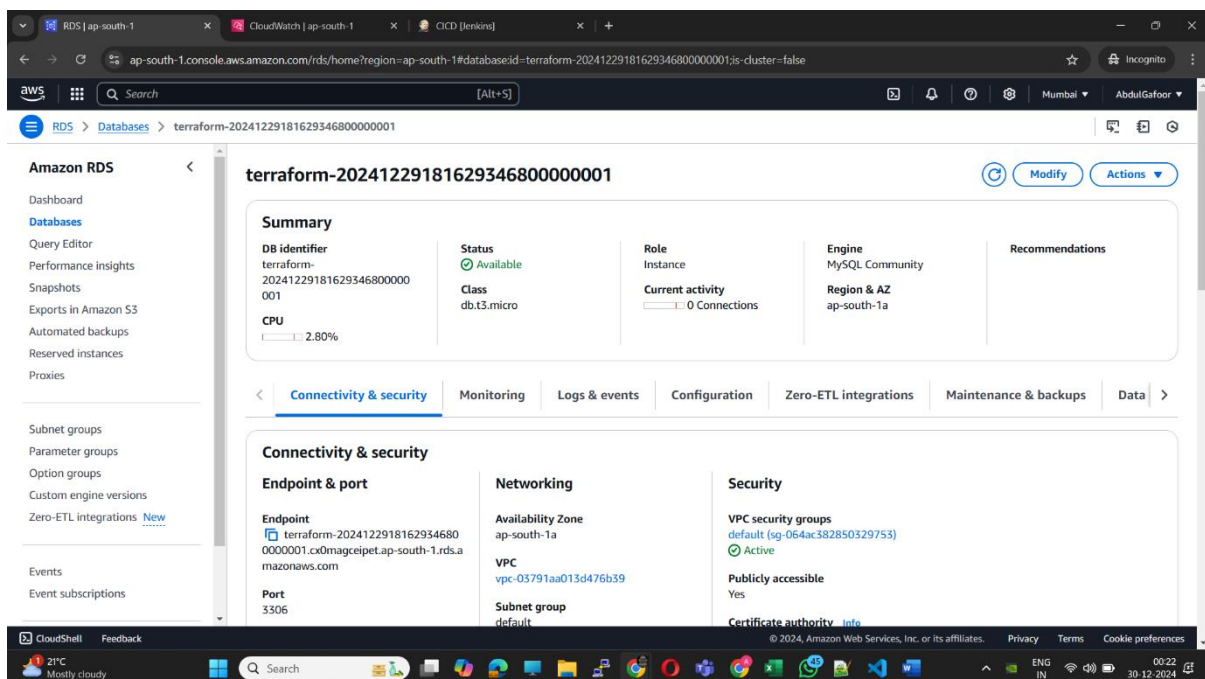


## Go Digital Technology Consulting LLP- TASK

Step 1: To mention provider AWS and then created Mysql Database.



```
1 provider "aws" {
2   | region = "ap-south-1"
3 }
4 #Creating database
5 resource "aws_db_instance" "default" {
6   allocated_storage = 10
7   db_name           = "mydb"
8   engine            = "mysql"
9   engine_version    = "8.0"
10  instance_class     = "db.t3.micro"
11  username           = "admin"
12  password           = "ashher1234"
13  parameter_group_name = "default.mysql8.0"
14  skip_final_snapshot = true
15  publicly_accessible = true
16 }
17 #Creating ecr repository
18 resource "aws_ecr_repository" "my_image_repo" {
19   name = "my-image-repo" # Name of the repository
20 }
```

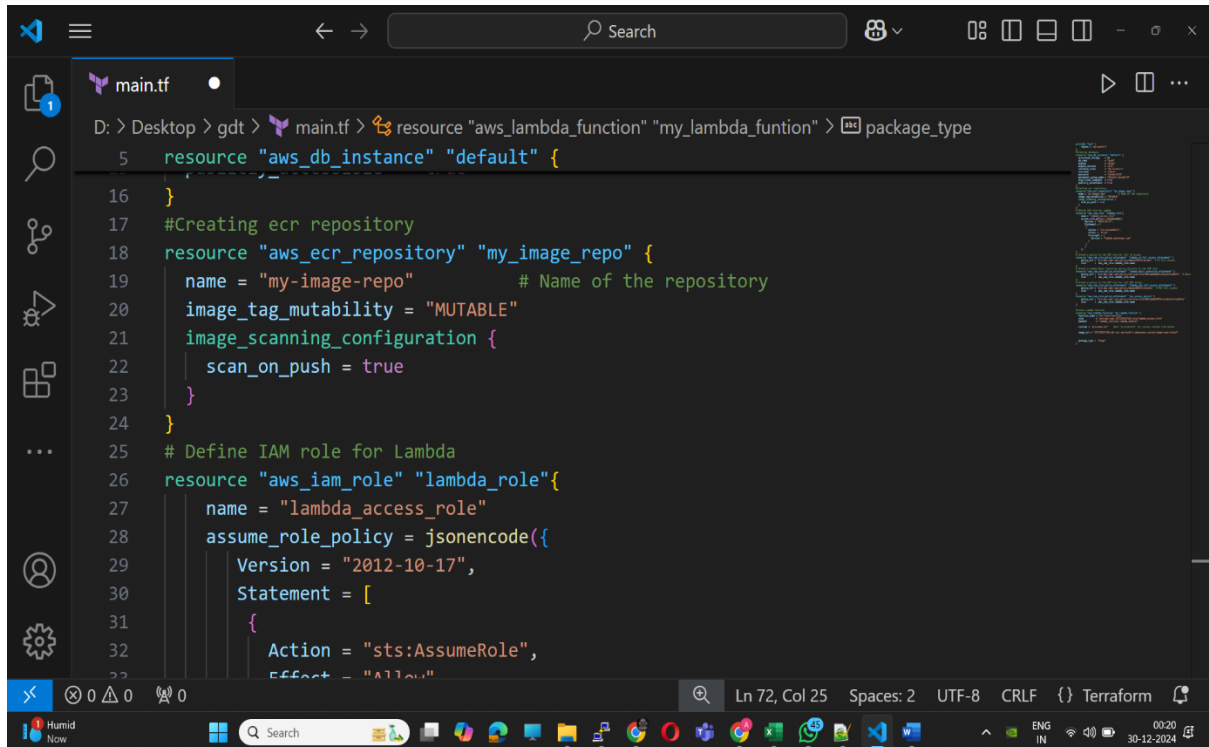


The screenshot shows the Amazon RDS console for a MySQL database instance. The instance is named 'terraform-20241229181629346800000001' and is in the 'ap-south-1' region. The instance is currently 'Available' and is using the 'db.t3.micro' class. The engine is 'MySQL Community' and the region is 'ap-south-1a'. The instance has 0 connections and is publicly accessible. The endpoint is 'terraform-20241229181629346800000001.cx0macejpet.ap-south-1.rds.amazonaws.com' and the port is 3306. The instance is in the 'ap-south-1a' availability zone and is using the 'vpc-03791aa013d476b39' VPC. The instance is using the 'default' subnet group. The instance is using the 'default' VPC security group. The instance is using the 'default' certificate authority.

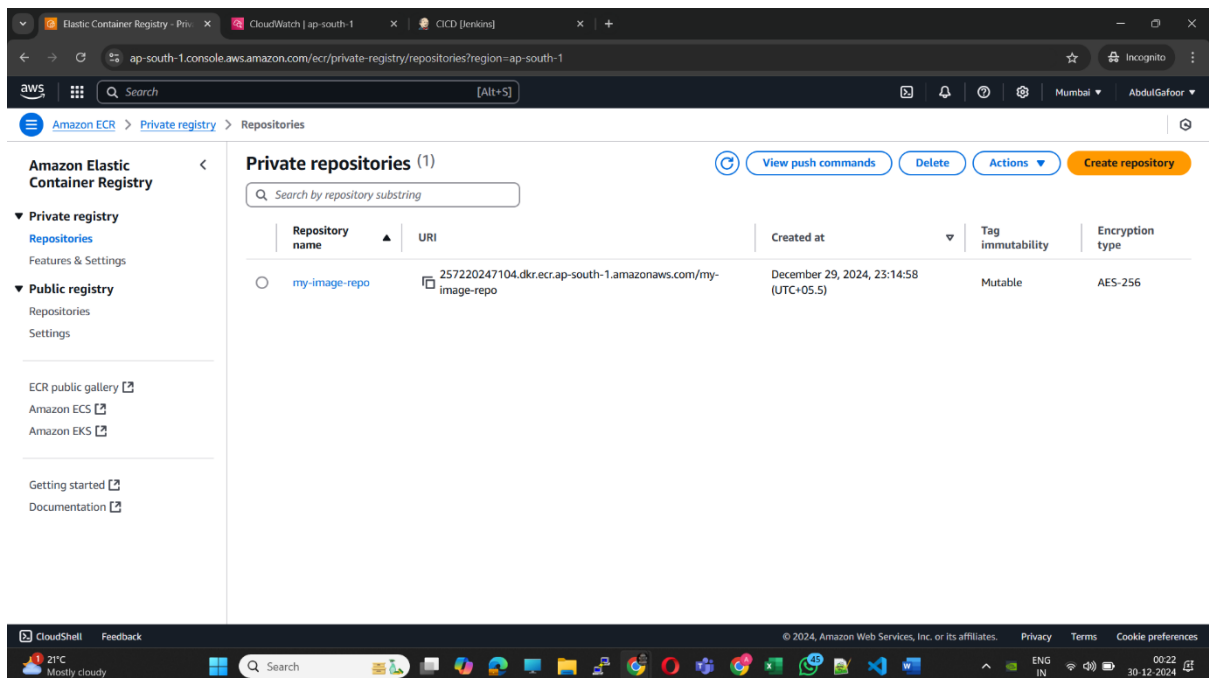
Summary				
DB identifier	terraform-20241229181629346800000001	Status	Available	Role
CPU	2.80%	Class	db.t3.micro	Instance
		Current activity	0 Connections	
		Engine	MySQL Community	
		Region & AZ	ap-south-1a	

Connectivity & security		
Endpoint & port	Networking	Security
Endpoint	Availability Zone	VPC security groups
terraform-20241229181629346800000001.cx0macejpet.ap-south-1.rds.amazonaws.com	ap-south-1a	default (sg-064ac382850329753)
Port	VPC	Active
3306	vpc-03791aa013d476b39	Publicly accessible
	Subnet group	Yes
	default	Certificate authority

## Step 2: Created ECR Repository



```
main.tf
D: > Desktop > gdt > main.tf > resource "aws_lambda_function" "my_lambda_funtion" > package_type
5  resource "aws_db_instance" "default" {
16 }
17 #Creating ecr repository
18 resource "aws_ecr_repository" "my_image_repo" {
19     name = "my-image-repo" # Name of the repository
20     image_tag_mutability = "mutable"
21     image_scanning_configuration {
22         scan_on_push = true
23     }
24 }
25 # Define IAM role for Lambda
26 resource "aws_iam_role" "lambda_role" {
27     name = "lambda_access_role"
28     assume_role_policy = jsonencode({
29         Version = "2012-10-17",
30         Statement = [
31             {
32                 Action = "sts:AssumeRole",
33                 Effect = "Allow",
34                 Principal = {
35                     AWS = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole"
36                 }
37             }
38         ]
39     })
40 }
```



### Step 3: Created Role for Lambda so that we can attach permissions to it to access S3 and RDS

```
main.tf
D: > Desktop > gdt > main.tf > resource "aws_lambda_function" "my_lambda_function" > package_type
18 resource "aws_ecr_repository" "my_image_repo" {
24 }
25 # Define IAM role for Lambda
26 resource "aws_iam_role" "lambda_role" {
27   name = "lambda_access_role"
28   assume_role_policy = jsonencode({
29     Version = "2012-10-17",
30     Statement = [
31       {
32         Action = "sts:AssumeRole",
33         Effect = "Allow",
34         Principal = {
35           Service = "lambda.amazonaws.com"
36         }
37       }
38     ]
39   })
40 }
41 # Attach a policy to the IAM role for full s3 access
```

us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/roles/details/lambda\_access\_role?section=permissions

## Identity and Access Management (IAM)

### lambda\_access\_role

**Summary**

Creation date: December 29, 2024, 23:46 (UTC+05:30)

Last activity: 16 minutes ago

ARN: `arn:aws:iam::257220247104:role/lambda_access_role`

Maximum session duration: 1 hour

**Permissions policies (4)**

You can attach up to 10 managed policies.

Policy name	Type	Attached entities
<input type="checkbox"/> <a href="#">AmazonRDSFullAccess</a>	AWS managed	2
<input type="checkbox"/> <a href="#">AmazonS3FullAccess</a>	AWS managed	2
<input type="checkbox"/> <a href="#">AWSLambdaBasicExecutionRole</a>	AWS managed	1

#### Step 4: Attaching the policies to IAM role required for Lambda to access S3 and RDS.

```
main.tf
D: > Desktop > gdt > main.tf > resource "aws_lambda_function" "my_lambda_funtion" > package_type
40 }
41 # Attach a policy to the IAM role for full s3 access
42 resource "aws_iam_role_policy_attachment" "lambda_s3_full_access_attachment" {
43     policy_arn = "arn:aws:iam::aws:policy/AmazonS3FullAccess" # S3 Full access
44     role       = aws_iam_role.lambda_role.name
45 }
46 # Attach a Lambda Basic Execution policy directly to the IAM role
47 resource "aws_iam_role_policy_attachment" "lambda_basic_execution_attachment" {
48     policy_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaBasicExecutionRole" #
49     role       = aws_iam_role.lambda_role.name
50 }
51 # Attach a policy to the IAM role for full RDS access
52 resource "aws_iam_role_policy_attachment" "lambda_rds_full_access_attachment" {
53     policy_arn = "arn:aws:iam::aws:policy/AmazonRDSFullAccess" # RDS Full access
54     role       = aws_iam_role.lambda_role.name
55 }
56 resource "aws_iam_role_policy_attachment" "vpc_access_policy" {
57     policy_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
58     role       = aws_iam_role.lambda_role.name
59 }
```

us-east-1.console.aws.amazon.com/iam/home?region=ap-south-1#/roles/details/lambda\_access\_role?section=permissions

**Identity and Access Management (IAM)**

Creation date: December 29, 2024, 23:46 (UTC+05:30)

Last activity: 16 minutes ago

ARN: arn:aws:iam:257220247104:role/lambda\_access\_role

Maximum session duration: 1 hour

**Permissions** | Trust relationships | Tags | Last Accessed | Revoke sessions

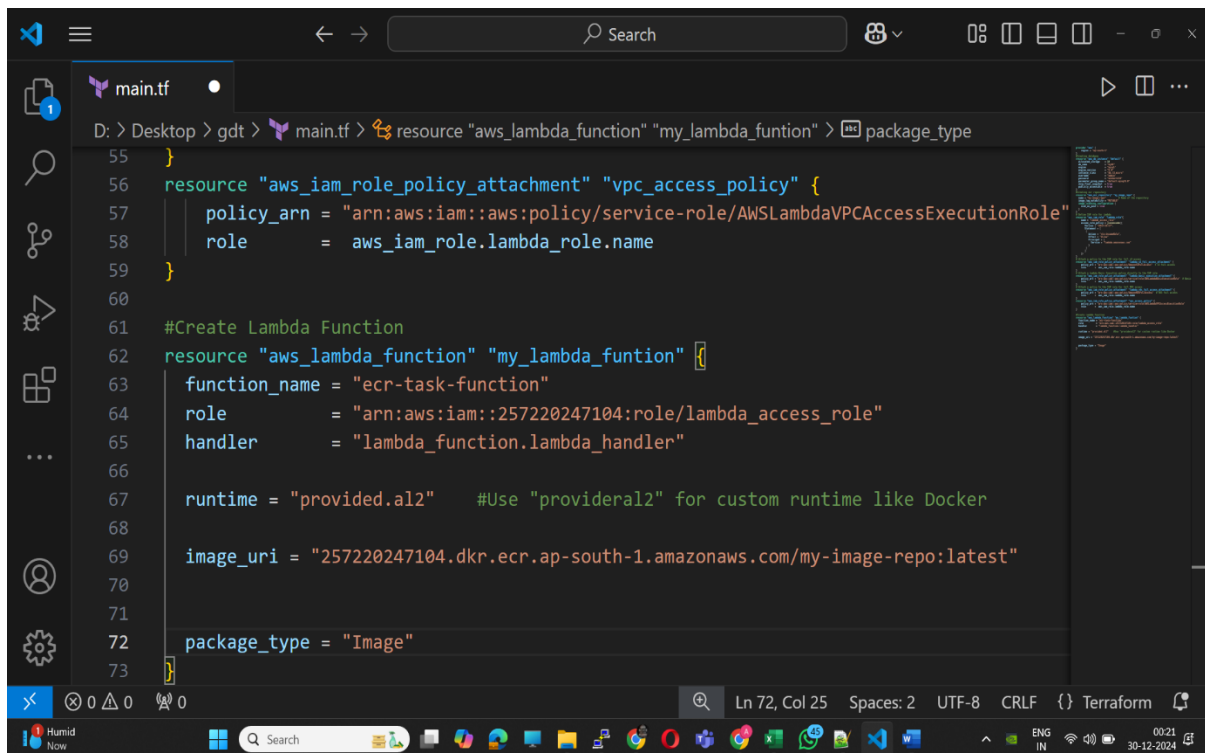
**Permissions policies (4)** Info

You can attach up to 10 managed policies.

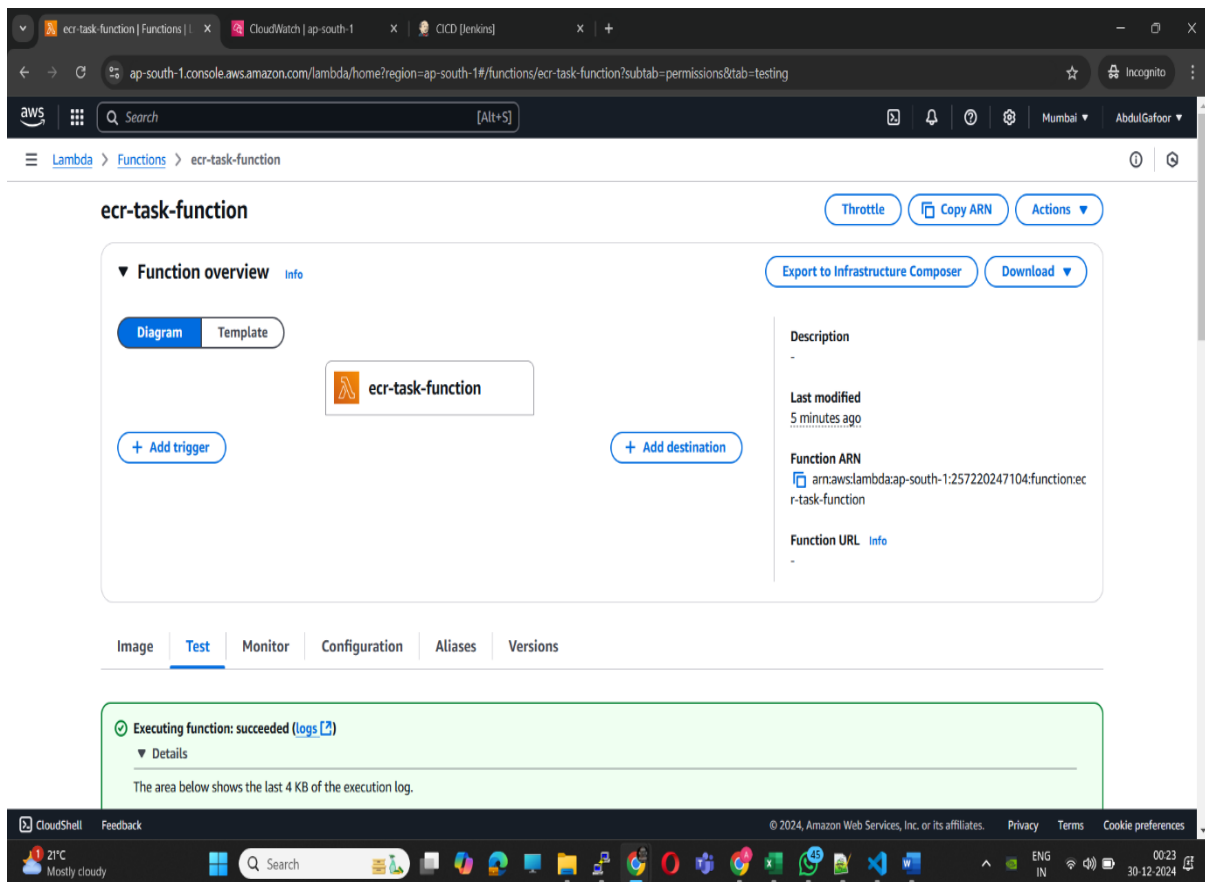
Policy name	Type	Attached entities
<input type="checkbox"/> AmazonRDSFullAccess	AWS managed	2
<input type="checkbox"/> AmazonS3FullAccess	AWS managed	2
<input type="checkbox"/> AWSLambdaBasicExecutionRole	AWS managed	1
<input type="checkbox"/> AWSLambdaVPCAccessExecutionRole	AWS managed	2

**Permissions boundary (not set)**

Step 5: Created Lambda Function with image uri and attached the role to Lambda Function.



```
55 }
56 resource "aws_iam_role_policy_attachment" "vpc_access_policy" {
57     policy_arn = "arn:aws:iam::aws:policy/service-role/AWSLambdaVPCAccessExecutionRole"
58     role       = aws_iam_role.lambda_role.name
59 }
60
61 #Create Lambda Function
62 resource "aws_lambda_function" "my_lambda_function" {
63     function_name = "ecr-task-function"
64     role          = "arn:aws:iam::257220247104:role/lambda_access_role"
65     handler       = "lambda_function.lambda_handler"
66
67     runtime = "provided.al2" #Use "provided.al2" for custom runtime like Docker
68
69     image_uri = "257220247104.dkr.ecr.ap-south-1.amazonaws.com/my-image-repo:latest"
70
71
72     package_type = "Image"
73 }
```



Step 6: Created Jenkins CI/CD Pipeline to automate all the stages including Cloning of Git Repo, Building Docker Image, Pushing image to ECR, Creating all resources, etc.

The image displays two screenshots of the Jenkins CI/CD Pipeline configuration interface, showing the 'Script' stage configuration.

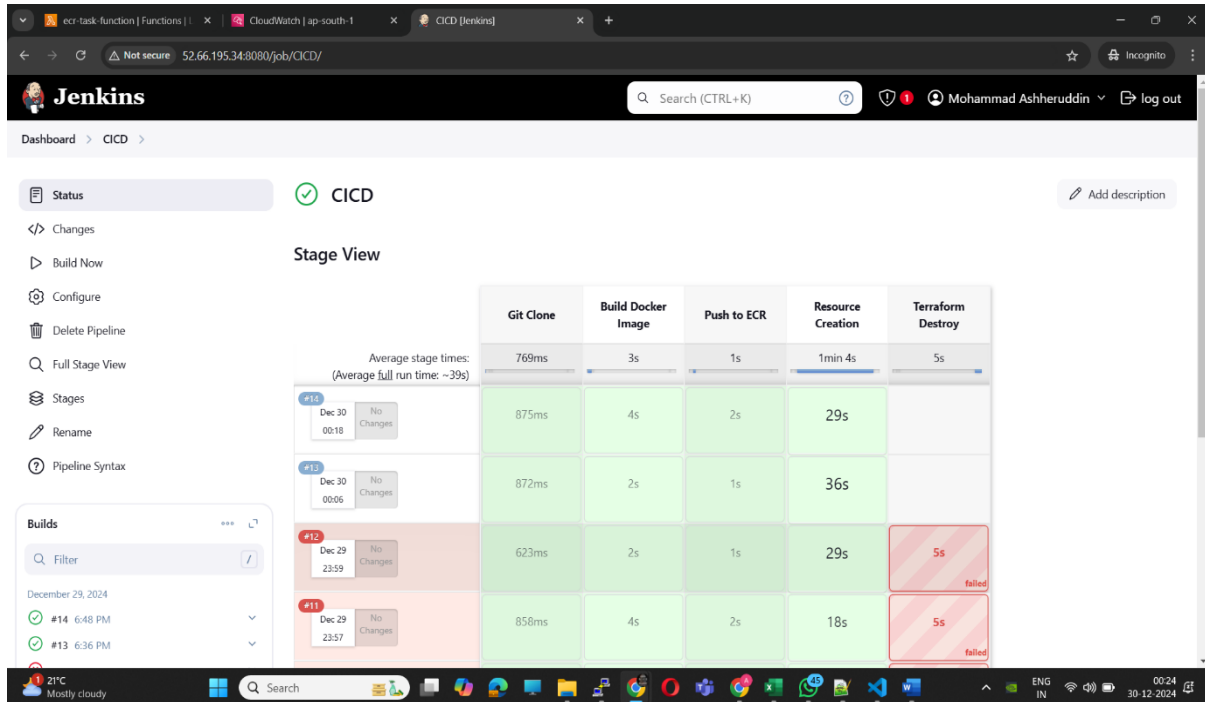
**Top Screenshot:** The 'Script' stage is configured with the following pipeline script:

```
1 pipeline {
2   agent any
3
4   stages {
5     stage('Git Clone') {
6       steps {
7         git branch: 'main', changelog: false, poll: false, url: 'https://github.com/Ashher05/task.git'
8       }
9     }
10
11    stage('Build Docker Image') {
12      steps {
13        sh 'docker build -t my-image-repo:latest .'
14      }
15    }
16
17    stage('Push to ECR') {
18      steps {
19        withCredentials([
20          [
21            $class: 'AmazonWebServicesCredentialsBinding',
22            credentialsId: '4ebe76b2-2204-4506-a395-4ab60568df57',
23            accessKeyVariable: 'AWS_ACCESS_KEY_ID',
24            secretKeyVariable: 'AWS_SECRET_ACCESS_KEY'
25          ]
26        ]) {
27          sh """
28            aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 257220247104.dkr.ecr.ap-south-1.amazonaws.com
29            docker tag my-image-repo:latest 257220247104.dkr.ecr.ap-south-1.amazonaws.com/my-image-repo:latest
30            docker push 257220247104.dkr.ecr.ap-south-1.amazonaws.com/my-image-repo:latest
31          """
32        }
33      }
34    }
35  }
36}
```

**Bottom Screenshot:** The 'Script' stage is configured with the following pipeline script:

```
23
24     accessKeyVariable: 'AWS_ACCESS_KEY_ID',
25     secretKeyVariable: 'AWS_SECRET_ACCESS_KEY'
26   }) {
27     sh """
28       aws ecr get-login-password --region ap-south-1 | docker login --username AWS --password-stdin 257220247104.dkr.ecr.ap-south-1.amazonaws.com
29       docker tag my-image-repo:latest 257220247104.dkr.ecr.ap-south-1.amazonaws.com/my-image-repo:latest
30       docker push 257220247104.dkr.ecr.ap-south-1.amazonaws.com/my-image-repo:latest
31     """
32   }
33 }
34
35 stage('Resource Creation') {
36   steps {
37     withCredentials([
38       [
39         $class: 'AmazonWebServicesCredentialsBinding',
40         credentialsId: '4ebe76b2-2204-4506-a395-4ab60568df57',
41         accessKeyVariable: 'AWS_ACCESS_KEY_ID',
42         secretKeyVariable: 'AWS_SECRET_ACCESS_KEY'
43       ]
44     ]) {
45       script {
46         // Initialize Terraform
47         sh "terraform init"
48
49         // Create Terraform plan
50         sh "terraform plan -out=tfplan"
51
52         // Apply Terraform changes
53         sh "terraform apply -auto-approve tfplan"
54       }
55     }
56   }
57 }
58
59 }
60
61 }
```

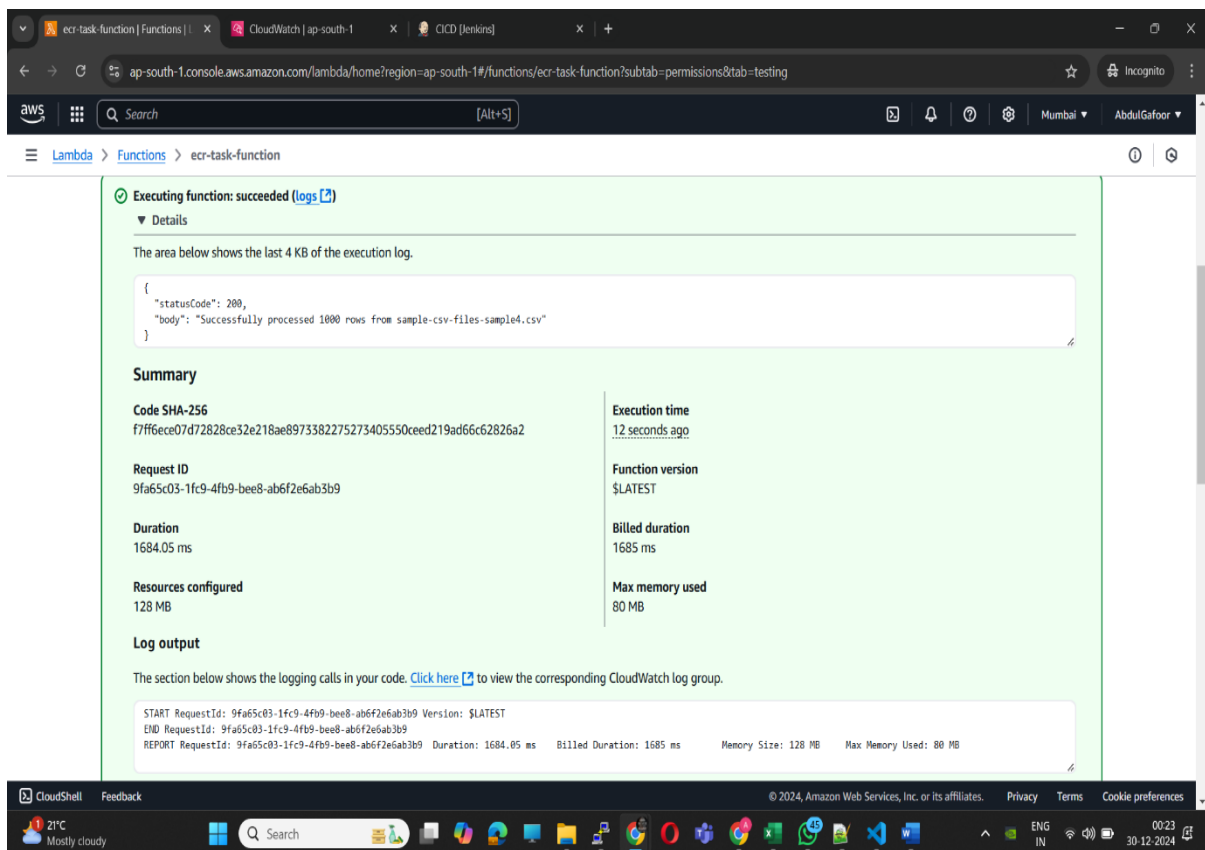
## Step 7: CICD Stages Completion



The screenshot shows the Jenkins web interface for a pipeline named 'CICD'. The pipeline is in a 'Completed' state, indicated by a green checkmark. The 'Stage View' displays a table of stages and their execution times. The stages are: Git Clone, Build Docker Image, Push to ECR, Resource Creation, and Terraform Destroy. The table shows the execution times for each stage across four builds. The first three builds are successful, while the fourth build failed at the 'Terraform Destroy' stage.

	Git Clone	Build Docker Image	Push to ECR	Resource Creation	Terraform Destroy
Average stage times: (Average full run time: ~39s)	769ms	3s	1s	1min 4s	5s
#16 Dec 30 00:18 No Changes	875ms	4s	2s	29s	
#15 Dec 30 00:06 No Changes	872ms	2s	1s	36s	
#12 Dec 29 23:59 No Changes	623ms	2s	1s	29s	5s failed
#11 Dec 29 23:57 No Changes	858ms	4s	2s	18s	5s failed

## Step 8: Final Outcome that our s3 data has been transferred to RDS in Lambda Function



The screenshot shows the AWS Lambda console for the 'ecr-task-function'. The function execution was successful. The execution log shows the status code 200 and the body: 'Successfully processed 1000 rows from sample-csv-files-sample4.csv'. The summary section provides details about the function execution, including the code SHA-256, request ID, duration, resources configured, and log output.

**Executing function: succeeded** (logs)

**Details**

The area below shows the last 4 KB of the execution log.

```
{
  "statusCode": 200,
  "body": "Successfully processed 1000 rows from sample-csv-files-sample4.csv"
}
```

**Summary**

<b>Code SHA-256</b> f7ff6ce07d72828ce32e218ae8973382275273405550ced219ad66c62826a2	<b>Execution time</b> 12 seconds ago
<b>Request ID</b> 9fa65c03-1fc9-4fb9-bee8-ab6f2e6ab3b9	<b>Function version</b> \$LATEST
<b>Duration</b> 1684.05 ms	<b>Billed duration</b> 1685 ms
<b>Resources configured</b> 128 MB	<b>Max memory used</b> 80 MB

**Log output**

The section below shows the logging calls in your code. [Click here](#) to view the corresponding CloudWatch log group.

```
START RequestId: 9fa65c03-1fc9-4fb9-bee8-ab6f2e6ab3b9 Version: $LATEST
END RequestId: 9fa65c03-1fc9-4fb9-bee8-ab6f2e6ab3b9
REPORT RequestId: 9fa65c03-1fc9-4fb9-bee8-ab6f2e6ab3b9 Duration: 1684.05 ms Billed Duration: 1685 ms Memory Size: 128 MB Max Memory Used: 80 MB
```

**THANK YOU**