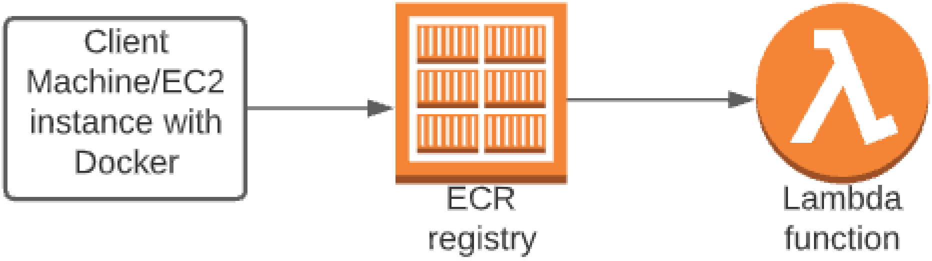
Scenario

The introduction of Lambda support for OCI container images provides customers with more choices when it comes to packaging formats. Developers can now choose to take advantage of the event-driven runtime model and cost-savings advantages of AWS Lambda, while taking advantage of the predictability and control offered by a container-based development and deployment cycle.

Architecture diagram

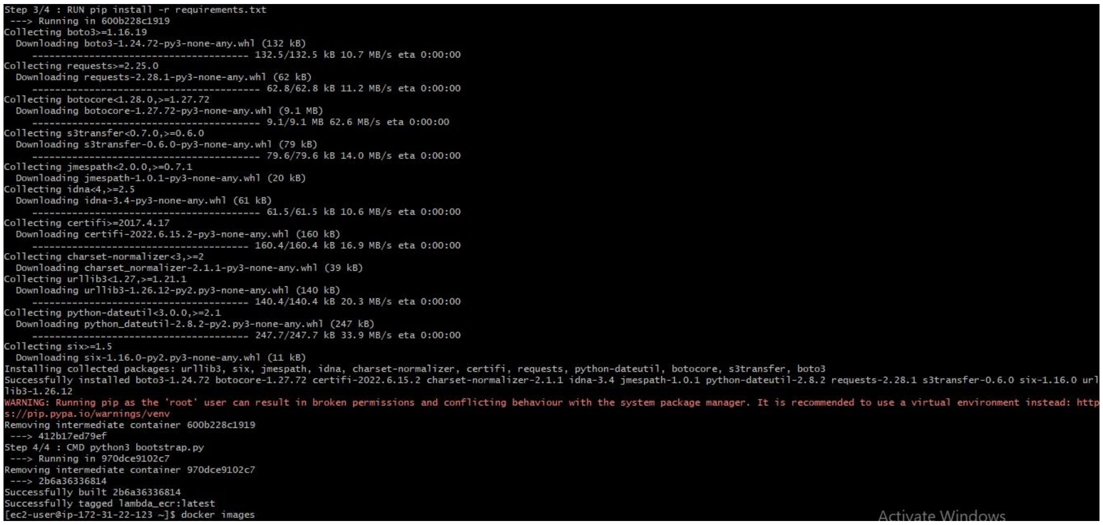


Architecture Implementation	
1	Download the Dockerfile and the app code folder provided with this workbook
2	Package the web application as a Docker image running on Alpine with Python
3	Create an ECR repository and login to it.
4	Build the image with the downloaded dockerfile and the support files
5	Tag the image appropriately and push it into the ECR repository.
6	Create a Lambda function with the image in ECR.

Step 1 : Docker Image creation

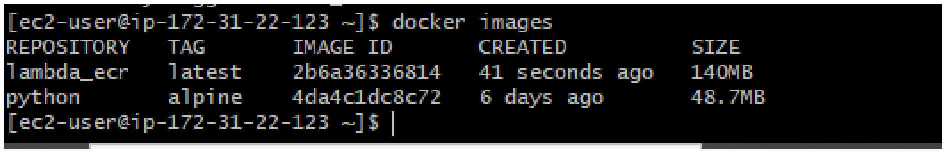
Step number	a
Step name	Creation of Docker image
Expected screenshots	1)Building the Docker image 2) List of the created image

Building Docker Image:



```
Step 3/4 : RUN pip install -r requirements.txt
 ---> Running in 600b228c1919
Collecting boto3>=1.16.19
  Downloading boto3-1.24.72-py3-none-any.whl (132 kB)
     ---------------------------------------- 132.5/132.5 kB 10.7 MB/s eta 0:00:00
Collecting requests>=2.25.0
  Downloading requests-2.28.1-py3-none-any.whl (62 kB)
     ---------------------------------------- 62.8/62.8 kB 11.2 MB/s eta 0:00:00
Collecting botocore<1.28.0,>=1.27.72
  Downloading botocore-1.27.72-py3-none-any.whl (9.1 MB)
     ---------------------------------------- 9.1/9.1 MB 62.6 MB/s eta 0:00:00
Collecting s3transfer<0.7.0,>=0.6.0
  Downloading s3transfer-0.6.0-py3-none-any.whl (79 kB)
     ---------------------------------------- 79.6/79.6 kB 14.0 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB)
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB)
     ---------------------------------------- 61.5/61.5 kB 10.6 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.6.15.2-py3-none-any.whl (160 kB)
     ---------------------------------------- 160.4/160.4 kB 16.9 MB/s eta 0:00:00
Collecting charset-normalizer<3,>=2
  Downloading charset_normalizer-2.1.1-py3-none-any.whl (39 kB)
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.12-py2.py3-none-any.whl (140 kB)
     ---------------------------------------- 140.4/140.4 kB 20.3 MB/s eta 0:00:00
Collecting python-dateutil<3.0.0,>=2.1
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ---------------------------------------- 247.7/247.7 kB 33.9 MB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB)
Installing collected packages: urllib3, six, jmespath, idna, charset-normalizer, certifi, requests, python-dateutil, botocore, s3transfer, boto3
Successfully installed boto3-1.24.72 botocore-1.27.72 certifi-2022.6.15.2 charset-normalizer-2.1.1 idna-3.4 jmespath-1.0.1 python-dateutil-2.8.2 requests-2.28.1 s3transfer-0.6.0 six-1.16.0 url
lib3-1.26.12
WARNING: Running pip as the 'root' user can result in broken permissions and conflicting behaviour with the system package manager. It is recommended to use a virtual environment instead: http
s://pip.pypa.io/warnings/venv
Removing intermediate container 600b228c1919
 ---> 412b17ed79ef
Step 4/4 : CMD python3 bootstrap.py
 ---> Running in 970dce9102c7
Removing intermediate container 970dce9102c7
 ---> 2b6a36336814
Successfully built 2b6a36336814
Successfully tagged lambda_ecr:latest
[ec2-user@ip-172-31-22-123 ~]$ docker images
```

List of Created Image:

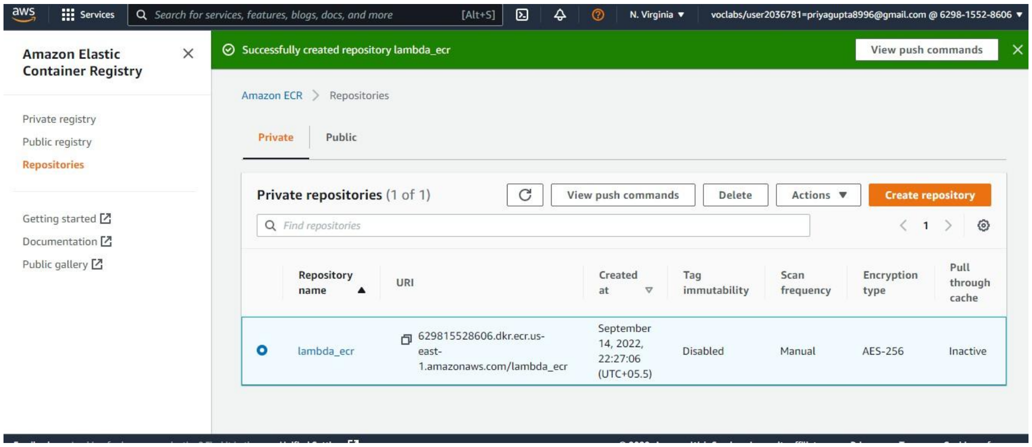


```
[ec2-user@ip-172-31-22-123 ~]$ docker images
REPOSITORY   TAG       IMAGE ID       CREATED          SIZE
lambda_ecr   latest    2b6a36336814   41 seconds ago   140MB
python       alpine    4da4c1dc8c72   6 days ago       48.7MB
[ec2-user@ip-172-31-22-123 ~]$ |
```

Step 2: Create ECR repository and upload image to ECR

Step number	a
Step name	Creating the ECR repository
Step number	b
Step name	Image upload to ECR
Expected screenshots	1) Creation of Repository 2) View push commands 3) Login Succeeded 4) Tagging of the image 5) Pushing of image to ECR 6) Image uploaded on the ECR repo

Creation of Repository

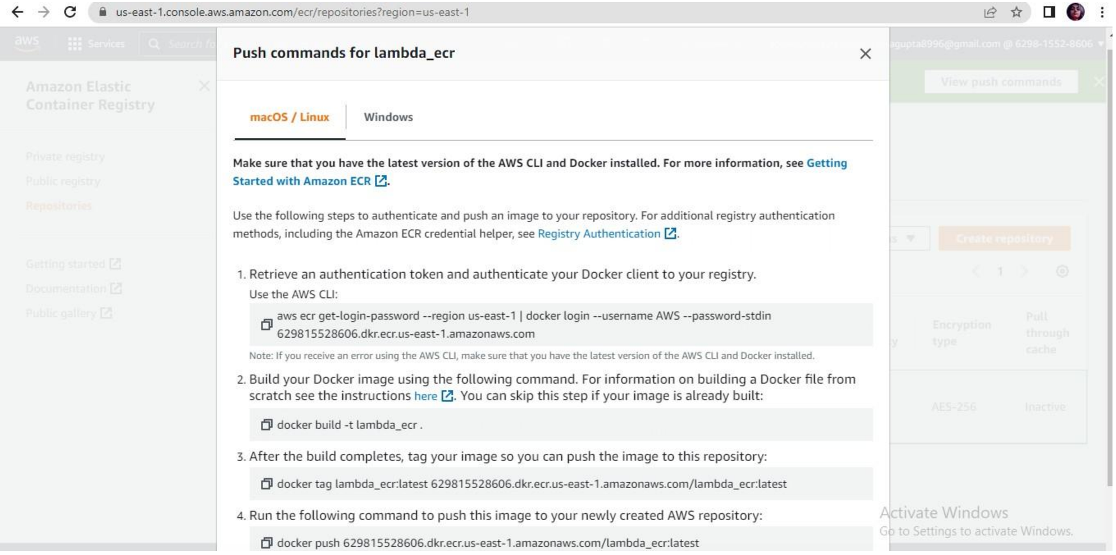


Successfully created repository lambda_ecr

Private repositories (1 of 1)

	Repository name ▲	URI	Created at ▽	Tag immutability	Scan frequency	Encryption type	Pull through cache
●	lambda_ecr	629815528606.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr	September 14, 2022, 22:27:06 (UTC+05.5)	Disabled	Manual	AES-256	Inactive

View push commands


us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1
macOS / Linux
Windows
Activate Windows
Go to Settings to activate Windows.

Push commands for lambda_ecr

Make sure that you have the latest version of the AWS CLI and Docker installed. For more information, see Getting Started with Amazon ECR.

Use the following steps to authenticate and push an image to your repository. For additional registry authentication methods, including the Amazon ECR credential helper, see Registry Authentication.

1. Retrieve an authentication token and authenticate your Docker client to your registry.
   Use the AWS CLI:

```
aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin
629815528606.dkr.ecr.us-east-1.amazonaws.com
```

Note: If you receive an error using the AWS CLI, make sure that you have the latest version of the AWS CLI and Docker installed.

2. Build your Docker image using the following command. For information on building a Docker file from scratch see the instructions here. You can skip this step if your image is already built:

```
docker build -t lambda_ecr .
```

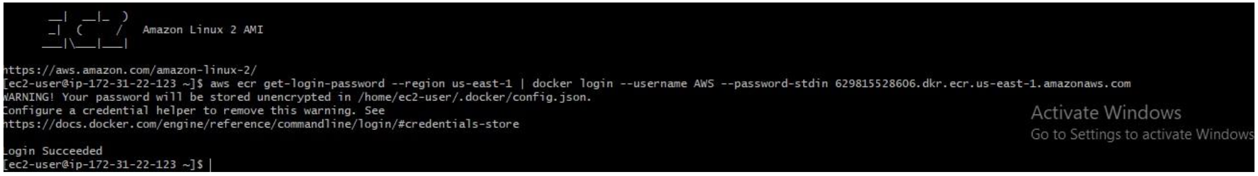
3. After the build completes, tag your image so you can push the image to this repository:

```
docker tag lambda_ecr:latest 629815528606.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
```

4. Run the following command to push this image to your newly created AWS repository:

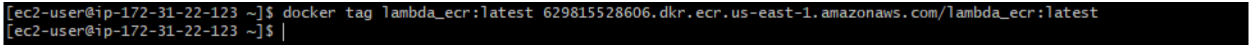
```
docker push 629815528606.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
```

Login Succeeded



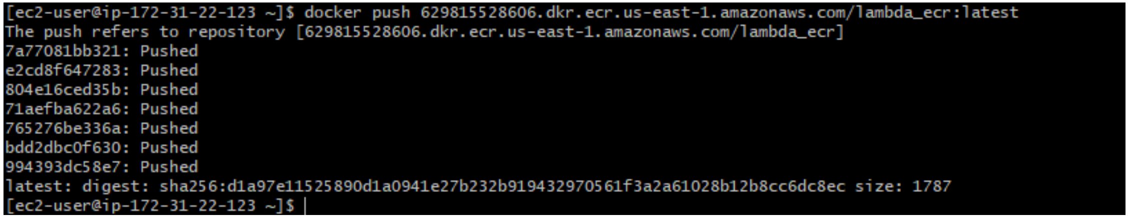
```
       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

ttps://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-22-123 ~]$ aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin 629815528606.dkr.ecr.us-east-1.amazonaws.com
ARNING! Your password will be stored unencrypted in /home/ec2-user/.docker/config.json.
onfigure a credential helper to remove this warning. See
ttps://docs.docker.com/engine/reference/commandline/login/#credentials-store

ogin Succeeded
[ec2-user@ip-172-31-22-123 ~]$ |
```

Tagging of the image



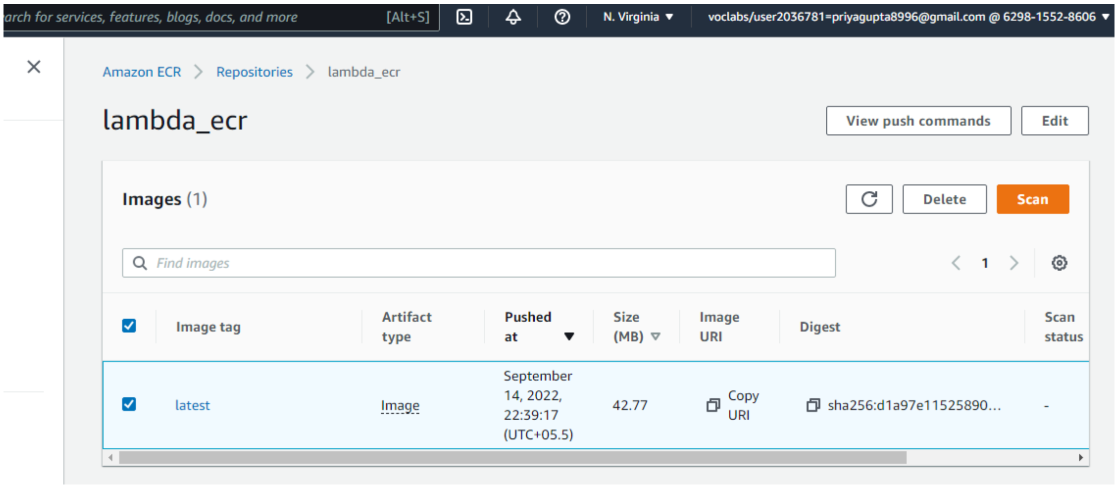
```
[ec2-user@ip-172-31-22-123 ~]$ docker tag lambda_ecr:latest 629815528606.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
[ec2-user@ip-172-31-22-123 ~]$ |
```

Pushing of image to ECR



```
[ec2-user@ip-172-31-22-123 ~]$ docker push 629815528606.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
The push refers to repository [629815528606.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr]
7a77081bb321: Pushed
e2cd8f647283: Pushed
804e16ced35b: Pushed
71aefba622a6: Pushed
765276be336a: Pushed
bdd2dbc0f630: Pushed
994393dc58e7: Pushed
latest: digest: sha256:d1a97e11525890d1a0941e27b232b919432970561f3a2a61028b12b8cc6dc8ec size: 1787
[ec2-user@ip-172-31-22-123 ~]$ |
```

Image uploaded on the ECR repository



arch for services, features, blogs, docs, and more
[Alt+S]
N. Virginia
voclabs/user2036781=priyagupta8996@gmail.com @ 6298-1552-8606
Amazon ECR > Repositories > lambda_ecr
View push commands
Edit
Delete
Scan
Find images
1

lambda_ecr

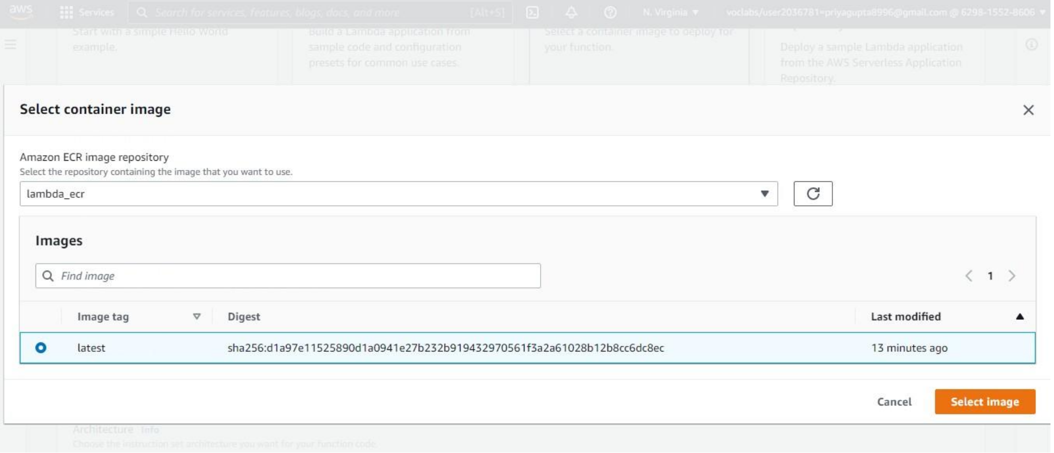
Images (1)

☑	Image tag	Artifact type	Pushed at ▼	Size (MB) ▽	Image URI	Digest	Scan status
☑	latest	Image	September 14, 2022, 22:39:17 (UTC+05.5)	42.77	Copy URI	sha256:d1a97e11525890...	-

Step 3: Creation of Lambda function to test the image

Step number	a
Step name	Create the Lambda function and test the image
Expected screenshots	1) Container image selection 2) Execution role selection 3) Created function 4)Test result of function

Container image selection



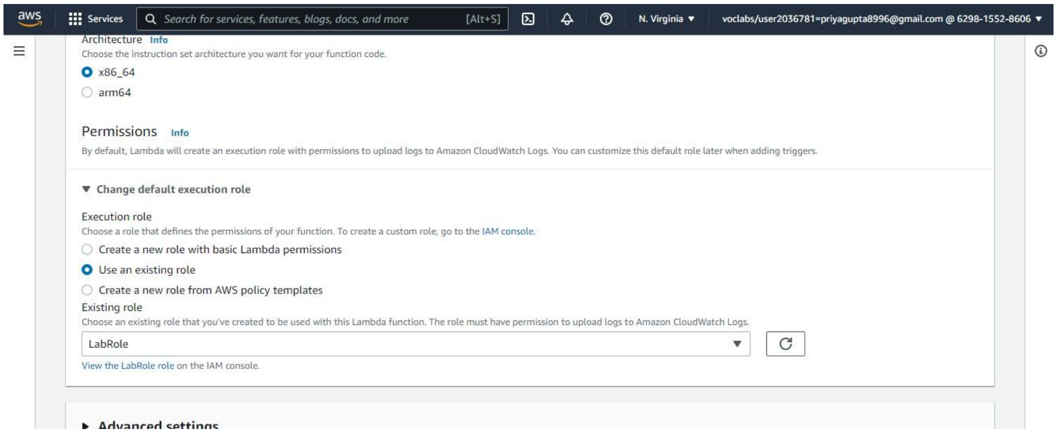
Select container image

Amazon ECR image repository
Select the repository containing the image that you want to use.
lambda_ecr

Images

	Image tag	Digest	Last modified
●	latest	sha256:d1a97e11525890d1a0941e27b232b919432970561f3a2a61028b12b8cc6dc8ec	13 minutes ago

Execution role selection


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia ▼
voclabs/user2036781=priyagupta8996@gmail.com @ 6298-1552-8606 ▼

Architecture Info

Choose the instruction set architecture you want for your function code.

● x86_64

○ arm64

Permissions Info

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.

▼ Change default execution role

Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the IAM console.

○ Create a new role with basic Lambda permissions

● Use an existing role

○ Create a new role from AWS policy templates

Existing role

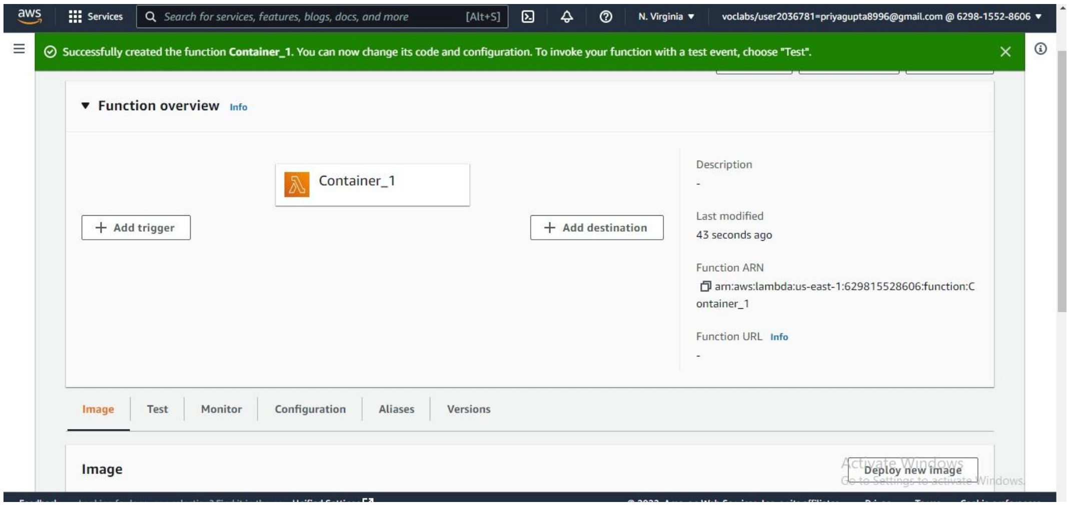
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

LabRole ▼

View the LabRole role on the IAM console.

▶ Advanced settings

Created function


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia ▼
voclabs/user2036781=priyagupta8996@gmail.com @ 6298-1552-8606 ▼
+ Add trigger
+ Add destination
Image
Test
Monitor
Configuration
Aliases
Versions
Deploy new image
Activate Windows
Go to Settings to activate Windows.

Successfully created the function **Container_1**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

▼ Function overview Info

Container_1

Description

-

Last modified

43 seconds ago

Function ARN

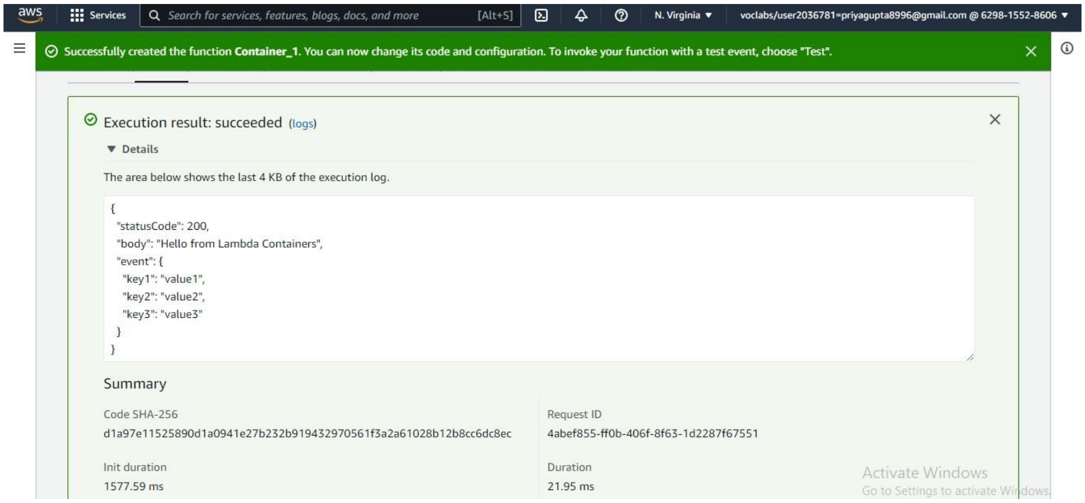
arn:aws:lambda:us-east-1:629815528606:function:Container_1

Function URL Info

-

Image

Test resultof function


aws
Services
Search for services, features, blogs, docs, and more
[Alt+S]
N. Virginia ▼
voclabs/user2036781=priyagupta8996@gmail.com @ 6298-1552-8606 ▼
Activate Windows
Go to Settings to activate Windows.

Successfully created the function **Container_1**. You can now change its code and configuration. To invoke your function with a test event, choose "Test".

Execution result: succeeded (logs)

▼ Details

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

```
{
  "statusCode": 200,
  "body": "Hello from Lambda Containers",
  "event": {
    "key1": "value1",
    "key2": "value2",
    "key3": "value3"
  }
}
```

Summary

Code SHA-256 d1a97e11525890d1a0941e27b232b919432970561f3a2a61028b12b8cc6dc8ec	Request ID 4abef855-ff0b-406f-8f63-1d2287f67551
Init duration 1577.59 ms	Duration 21.95 ms