Declaration	
Questions in this exercise are intentionally complex and could be convoluted or confusing. This is by design and to simulate real life situations where customers seldom give crystal clear requirements and ask unambiguous questions.	
I have read the above statement and agree to these conditions	
I AGREE	**ADEKUNLE ADEYINKA ADEGBIE**
	<Enter your name above this line to indicate that you are in agreement>

Instructions
Every screenshot requested in this workbook is compulsory and carries 1 point
Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and severe penalty will be imposed.
All screenshots must be in the order mentioned under "Expected Screenshots" for every step
DO NOT WAIT UNTIL THE LAST MINUTE. The program office will not extend the project submission deadline under any circumstances.
The file should be renamed in the format BATCH_FIRSTNAME_LASTNAME_PROJECT1. For example: PGPCCMAY18_VIJAY_DWIVEDI_PROJECT1.pdf

Resource Clean Up
Cloud is always pay per use model and all resources/services that we consume are chargeable. Cleaning up when you've completed your lab or project is always necessary. This is true whether you're doing a lab or implementing a project at your workplace.
After completing the lab, make sure to delete each resource created in reverse chronological order.

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.

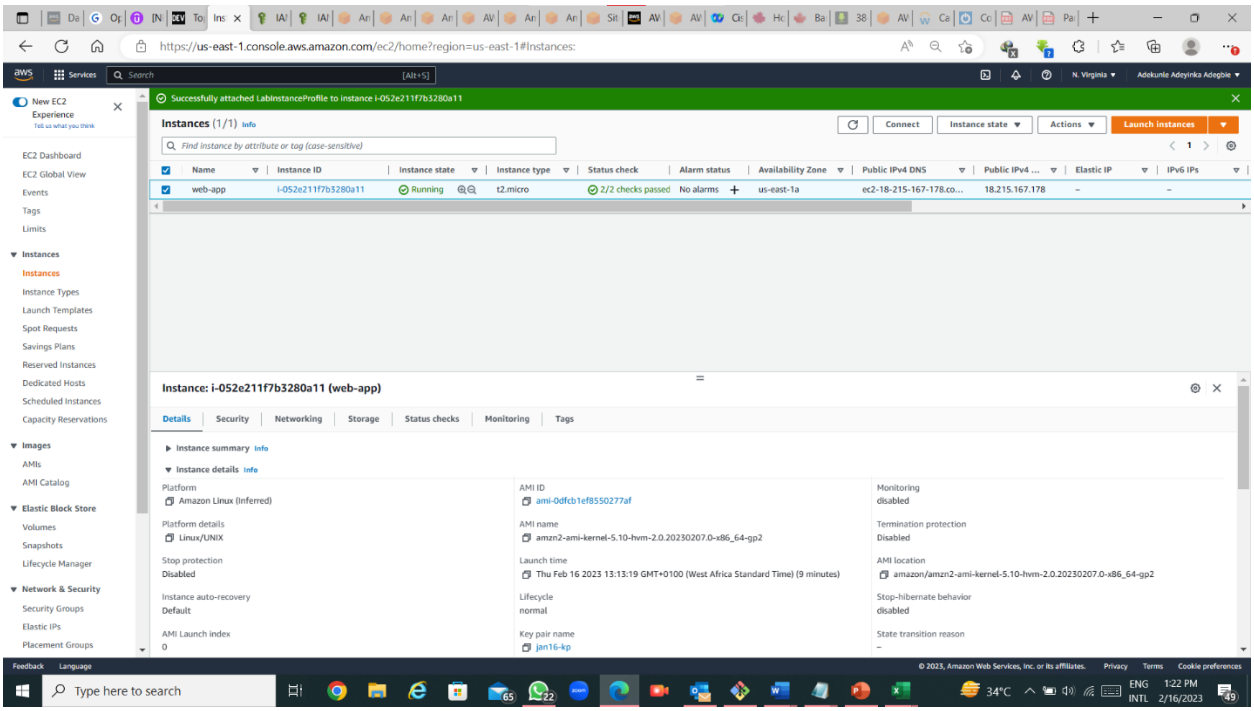
Architecture diagram

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.

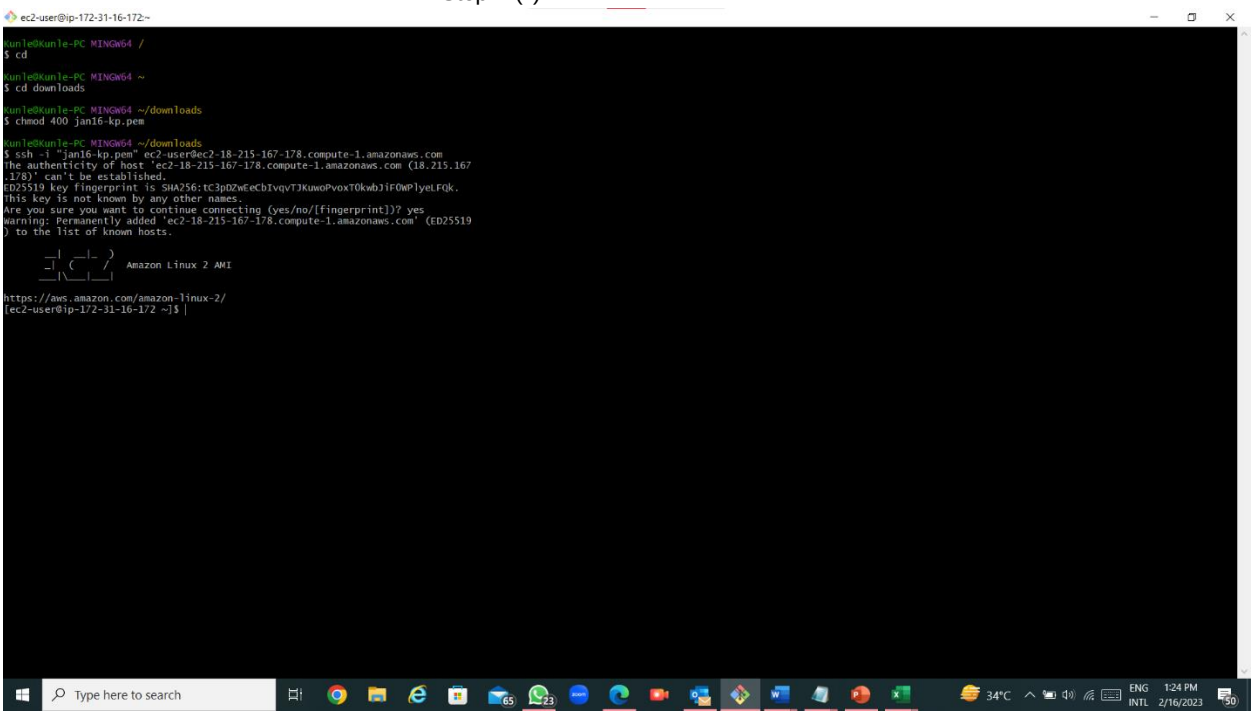
Step 1 : Docker Image creation

Step number	A
Step name	Creation of Docker image
Instructions	**1)** Create an EC2 instance using the Amazon Linux 2 AMI in the default VPC. 2) Attach the role “LabInstanceProfile” to the instance created above 3) Download the file OCI.zip provided with this workbook and copy it to the EC2 instance using the scp command scp -i <pem file name> ./OCI.zip ec2-user@<public IP of instance>:/home/ec2-user (Ensure that the file OCI.zip and the pem file are in the same folder before running this command) 4) Login to the instance using SSH and run the following commands to set up the environment *sudo yum update* *sudo yum install unzip* *sudo unzip OCI.zip* *sudo amazon-linux-extras install docker* *sudo service docker start* *sudo usermod -a -G docker ec2-user* **(At this point, log out of the instance and log in again to ensure that the above command works. Then continue with the rest of the commands)** *sudo yum install awscli -y* *aws configure* **Skip the access key and secret access key fields by pressing the Enter key. Enter the region as** **us-east-1 and format as json** 5) Run the below command to create the Docker image *sudo docker build -t lambda_ecr .* 6) Run the below command to verify the creation of the image *sudo docker images*
Expected screenshots	1)Building the Docker image 3) List of the created image

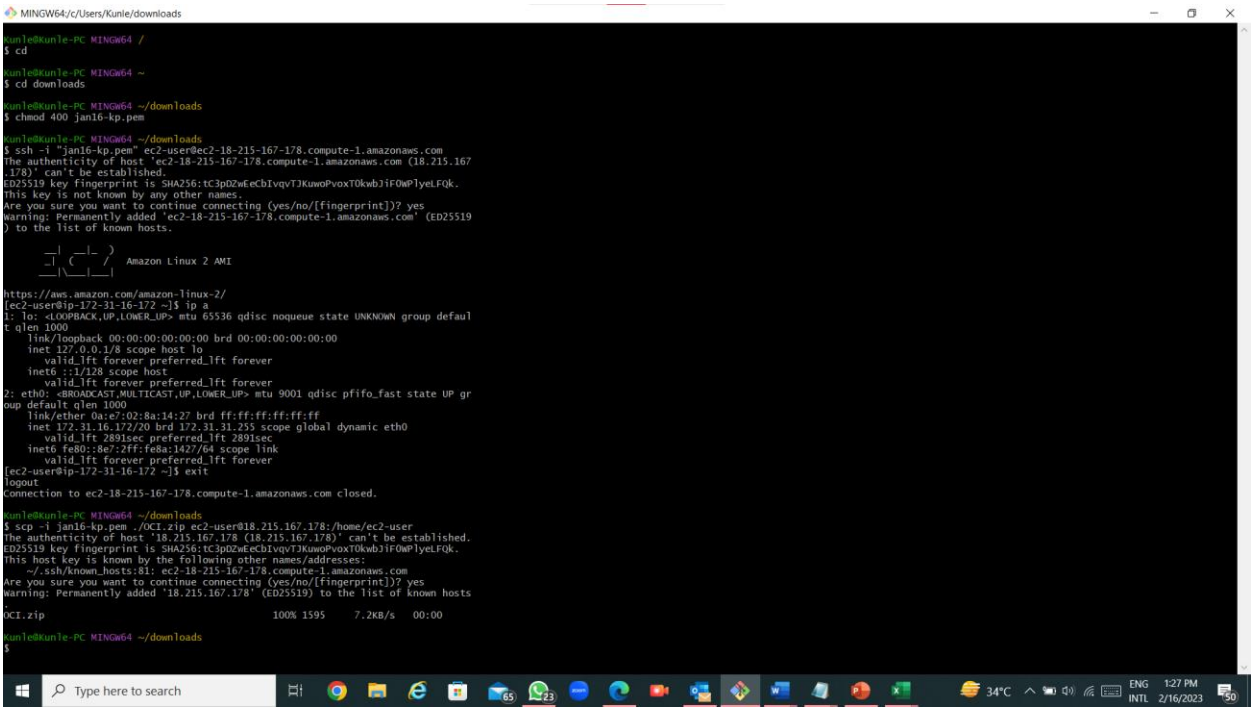
Step 1-(i)



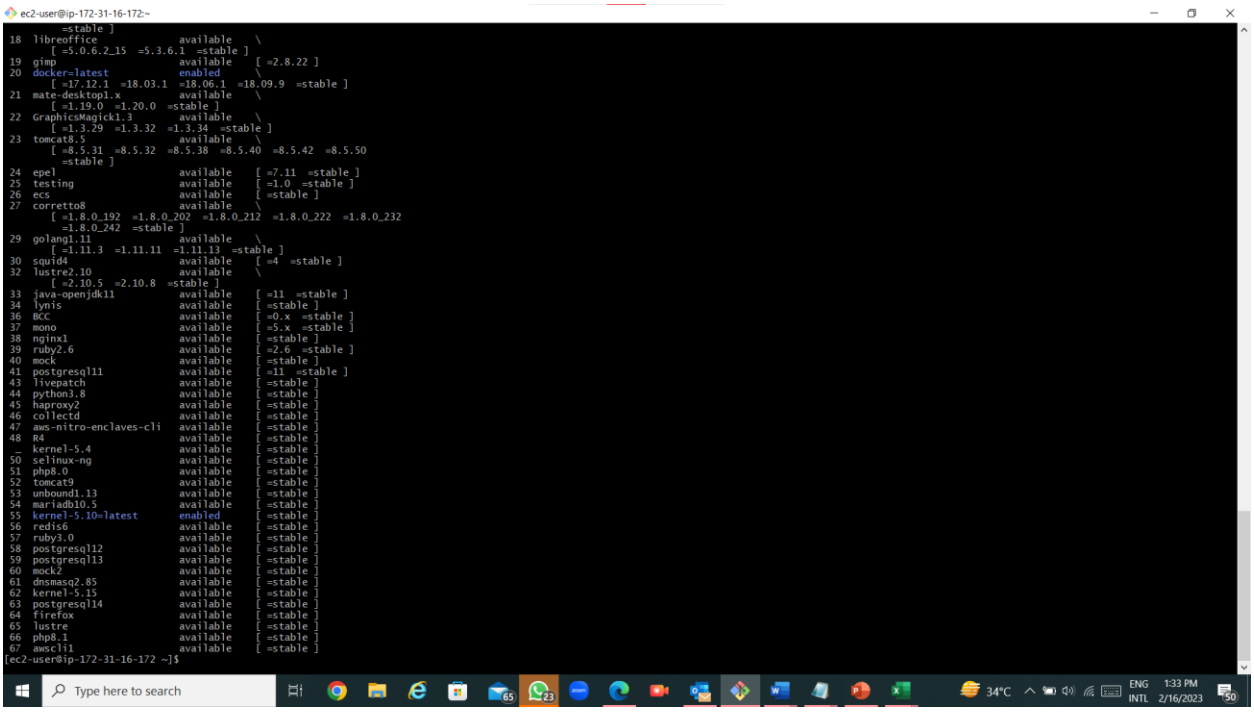
Step 1-(ii)



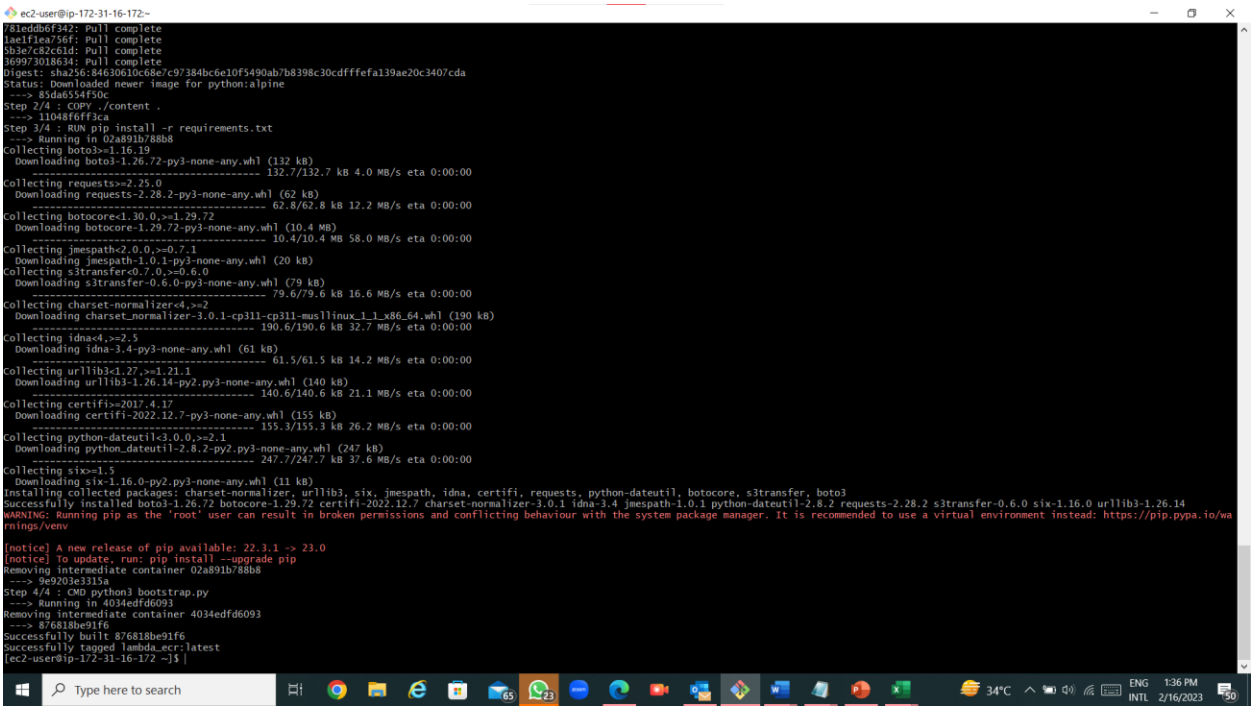
Step 1-(iii)



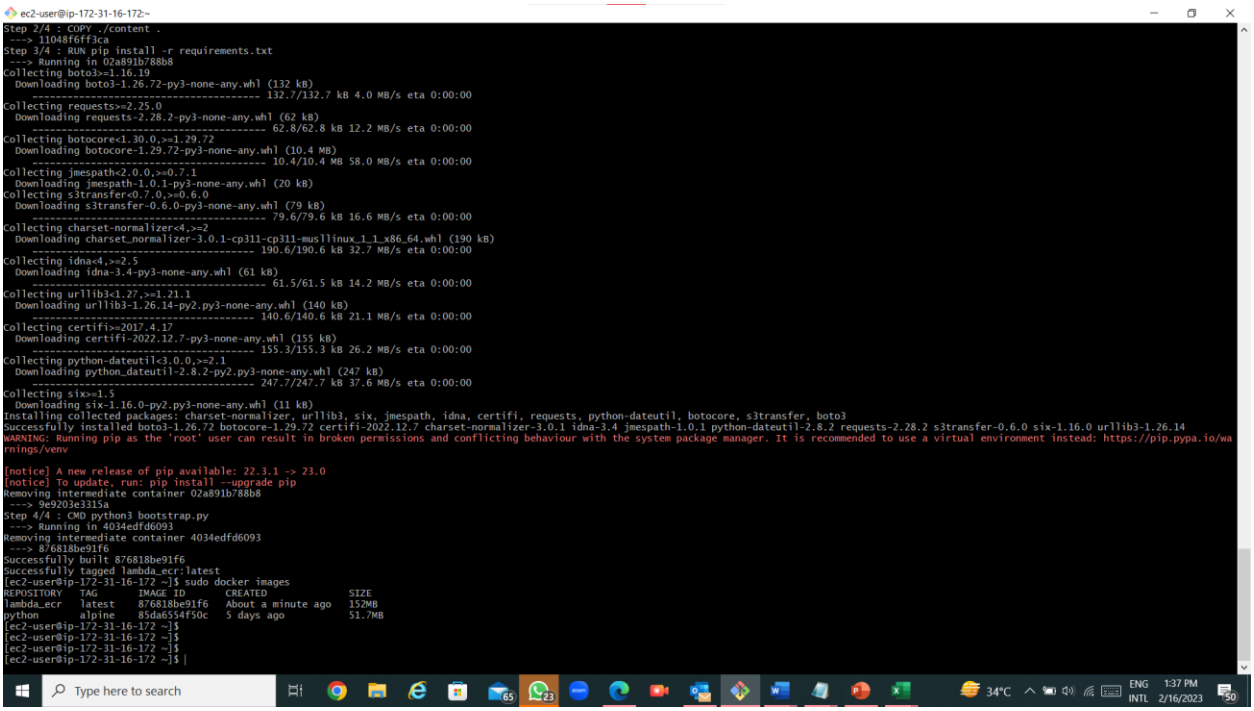
Step 1-(iv)



Step 1-(v)



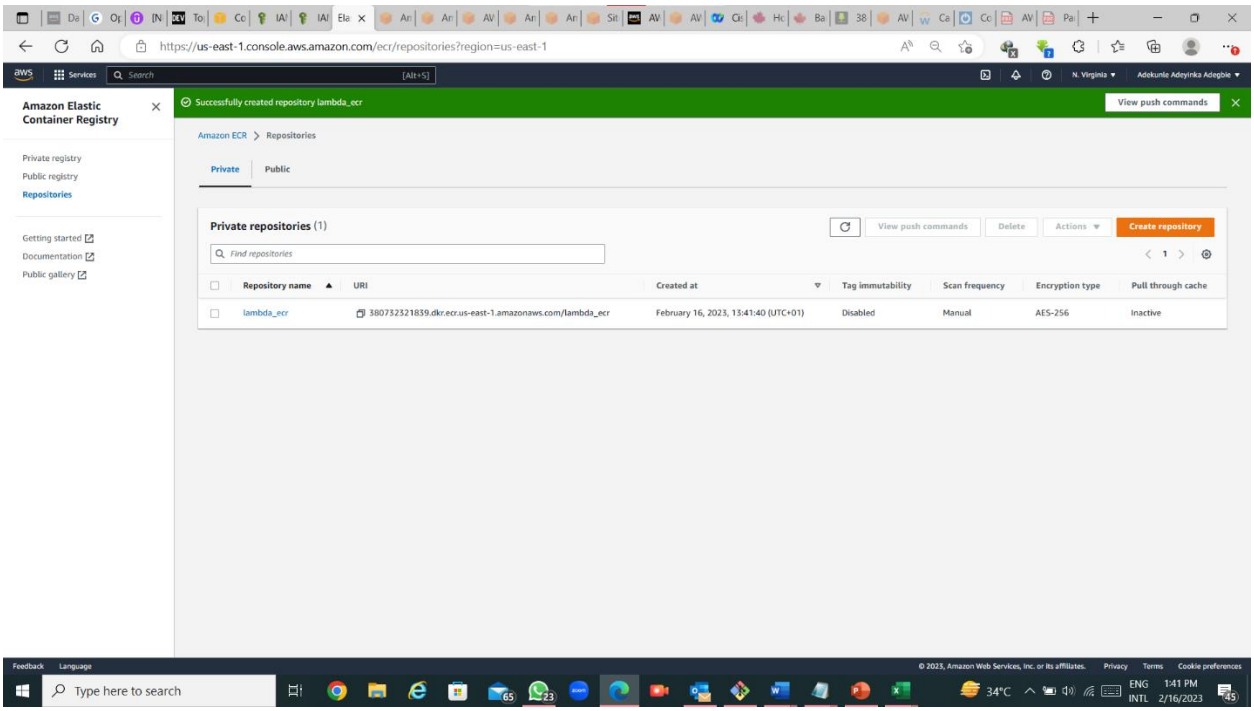
Step 1-(vi)



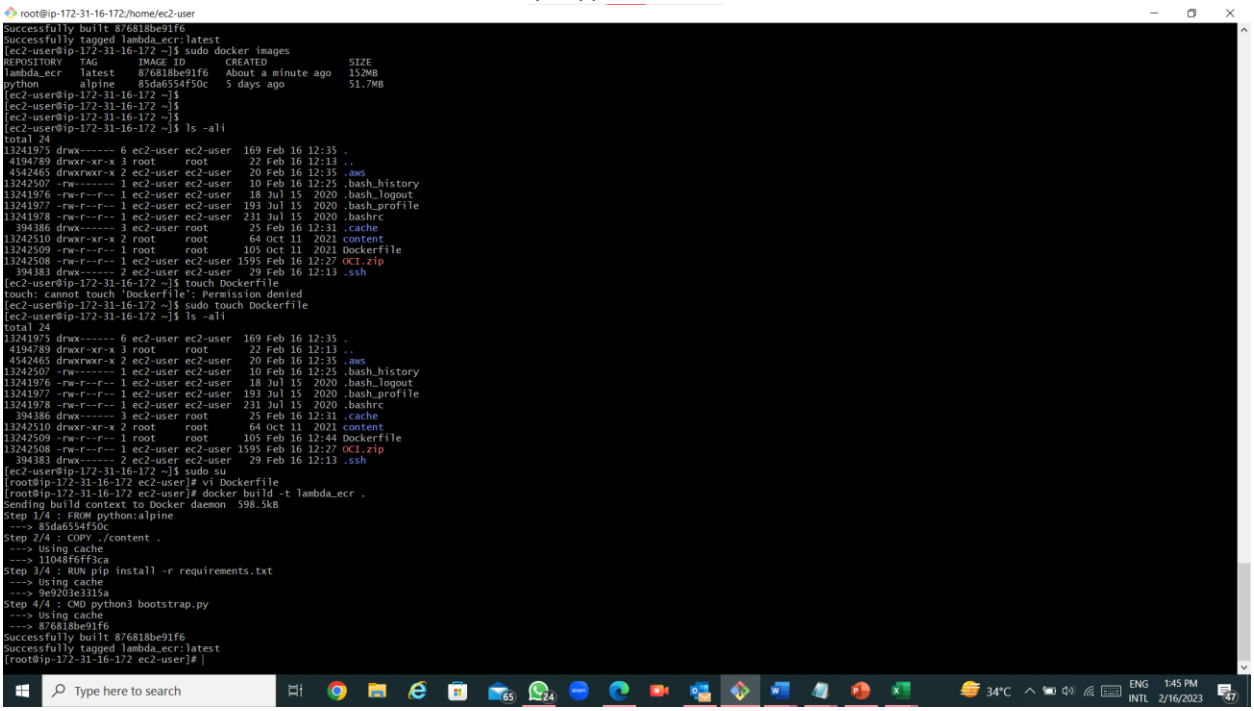
Step 2: Create ECR repository and upload image to ECR

Step number	a
Step name	Creating the ECR repository
Instructions	1) Go to the ECR service on the AWS console 2) Select the Repositories from the left pane 3) Create a new private repository named **lambda_ecr** with the default settings
Step number	B
Step name	Image upload to ECR
Instructions	1) Once the repository is created, select the repository and then click on “View push commands“ on the top right 2) From the pop up screen which appears, run commands 1, 3 and 4 after logging into the EC2 instance created above. Note that command 2 was already executed in the previous step when the image was created. For reference, the commands will be in the format shown below: **aws ecr get-login-password --region us-east-1 | docker login --username AWS --password-stdin <xxxxxxx.dkr.ecr.us-east-1.amazonaws.com>** **docker tag lambda_ecr_image:latest <xxxxxxx.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr>:latest** **docker push <xxxxxxx.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr>:latest**
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

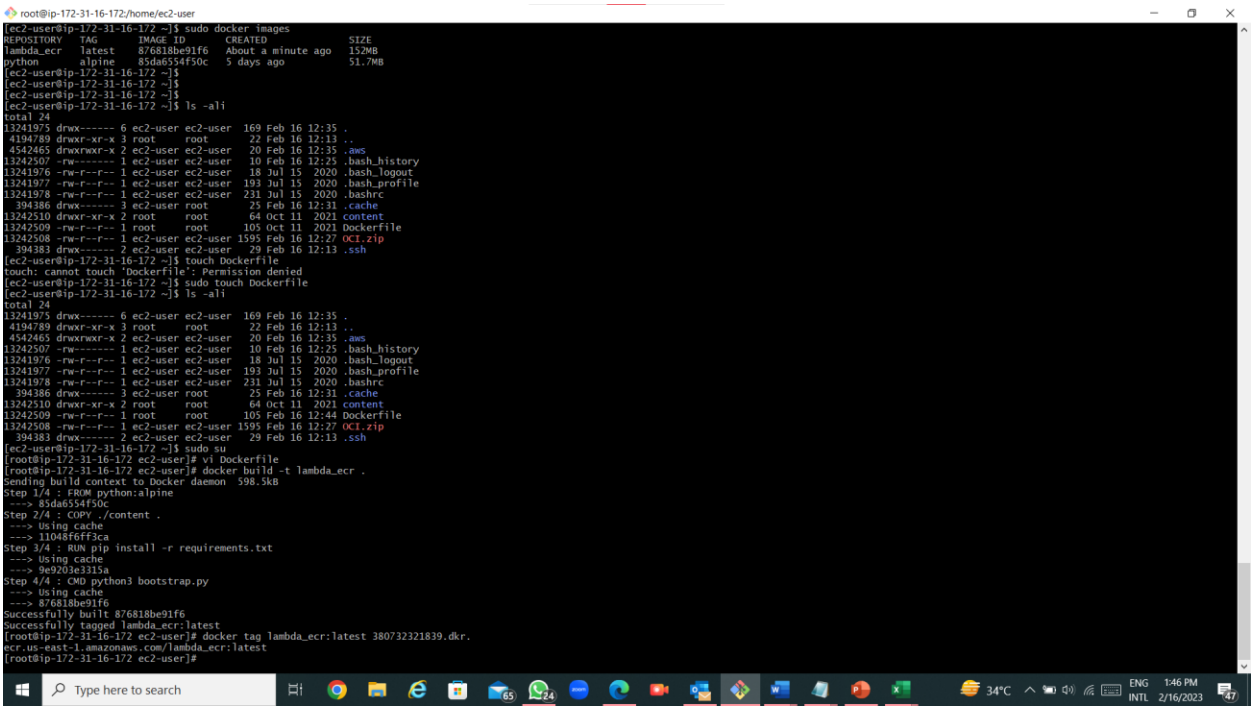
Step 2-(i)



Step 2-(ii)



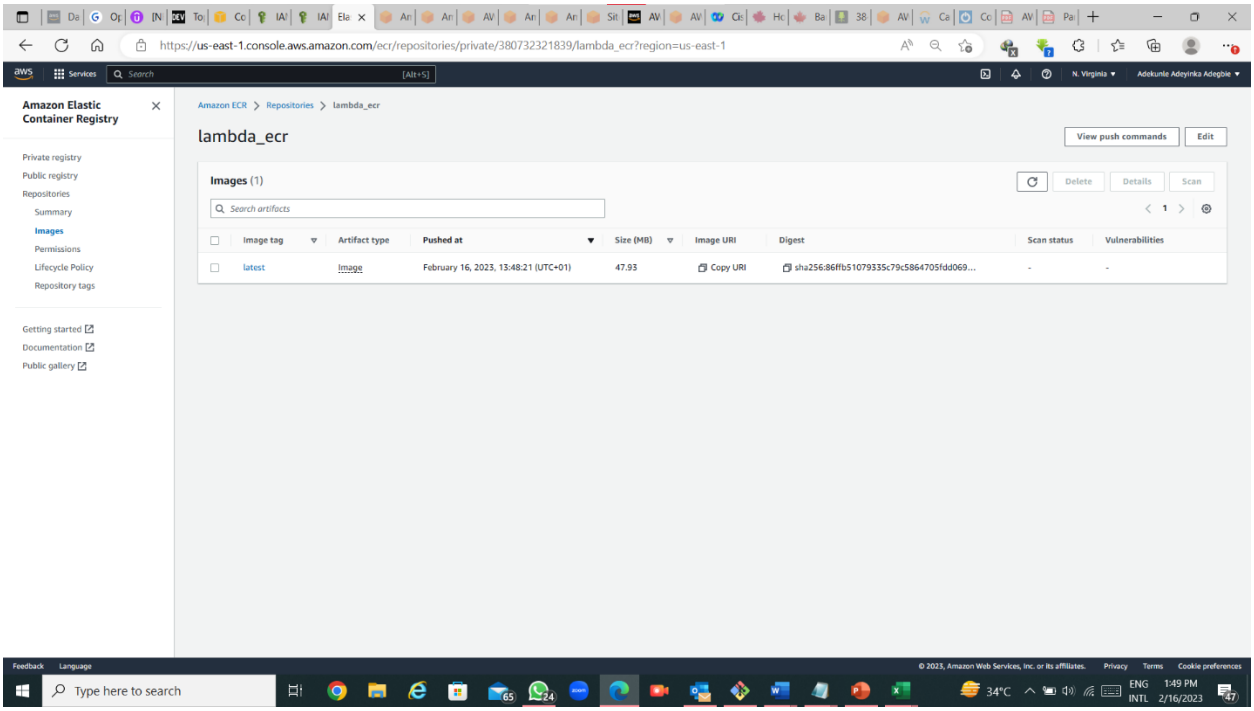
Step 2-(iii)



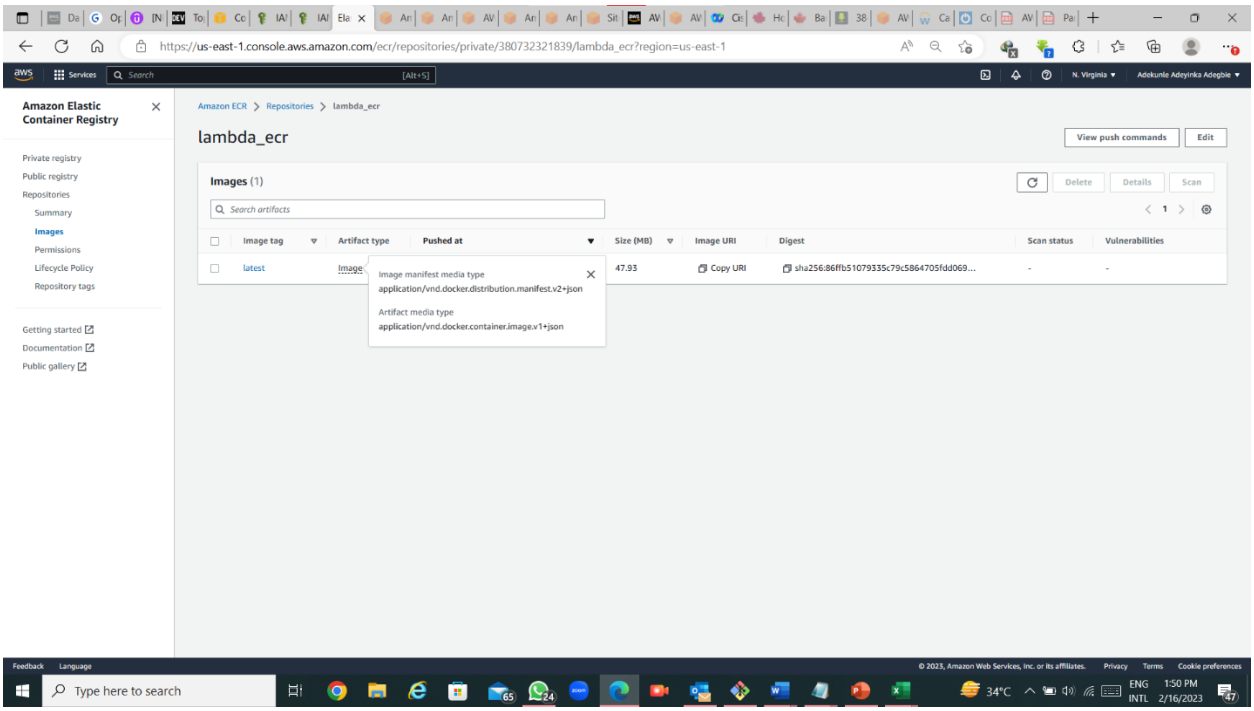
Step 2-(iv)

Step 2-(v)

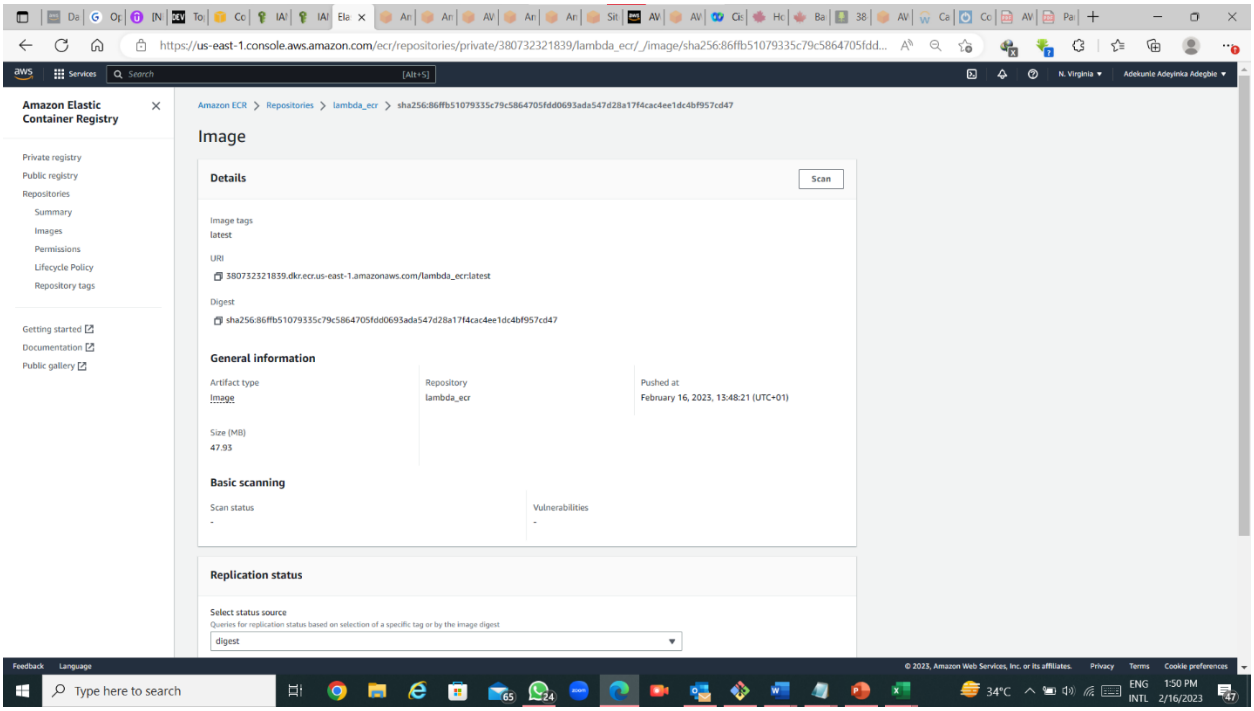
Step 2-(vi)



Step 2-(vii)



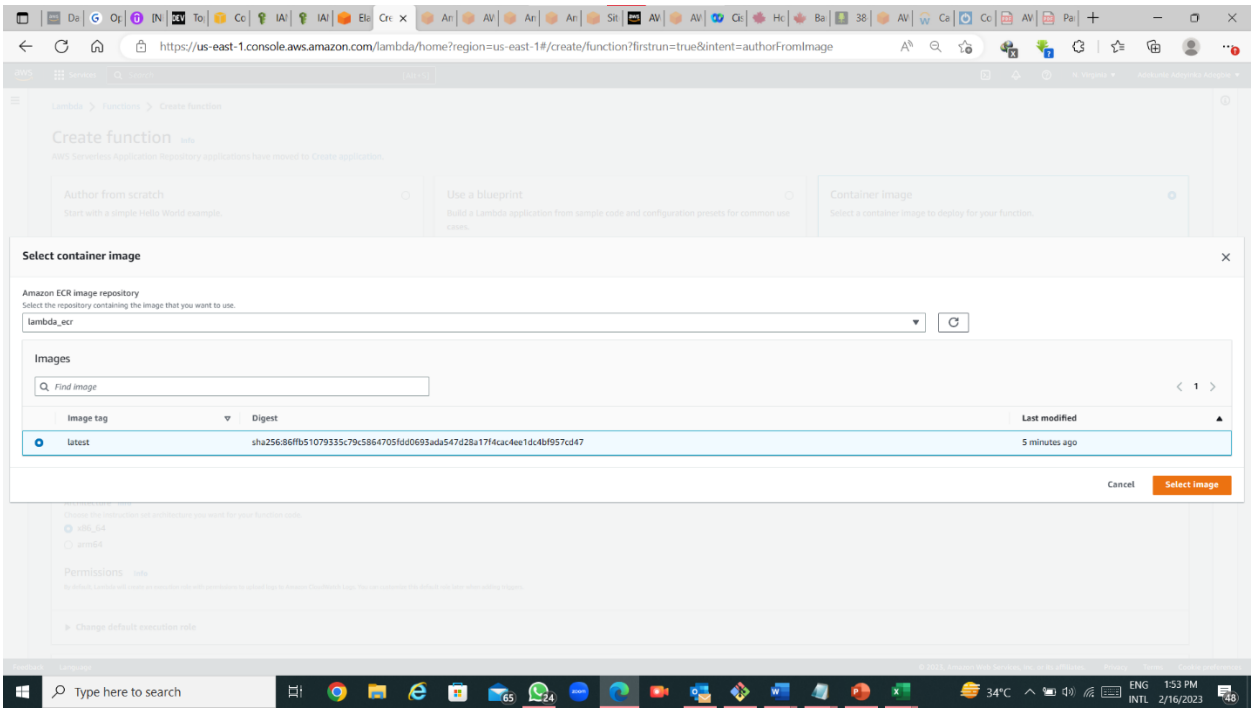
Step 2-(viii)



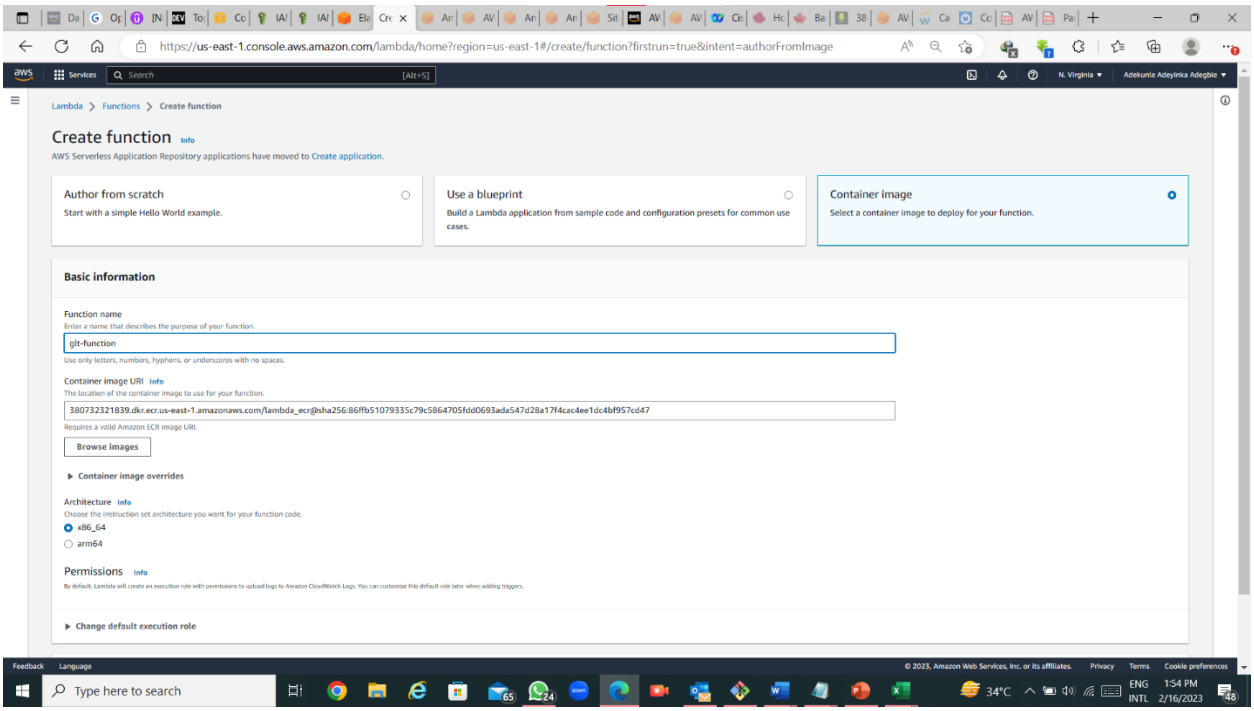
Step 3: Creation of Lambda function to test the image

Step number	A
Step name	Create the Lambda function and test the image
Instructions	1) Navigate to the AWS Lambda service using the AWS Console 2) Click on **Create Function** 3) Under Create Function page select the 'Container image' option and enter a function name of your choice 4) For 'Container image URI' Click on "Browse Images" and select the repository and the image 5) Use the existing IAM role – LabRole. 6) Click on Create 7) Wait a few minutes for the function to be created 8) Test the function with the default "Hello World" test to see the result.
Expected screenshots	1) Container image selection 2) Execution role selection 3) Created function 4)Test result of function

Step 3-(i)

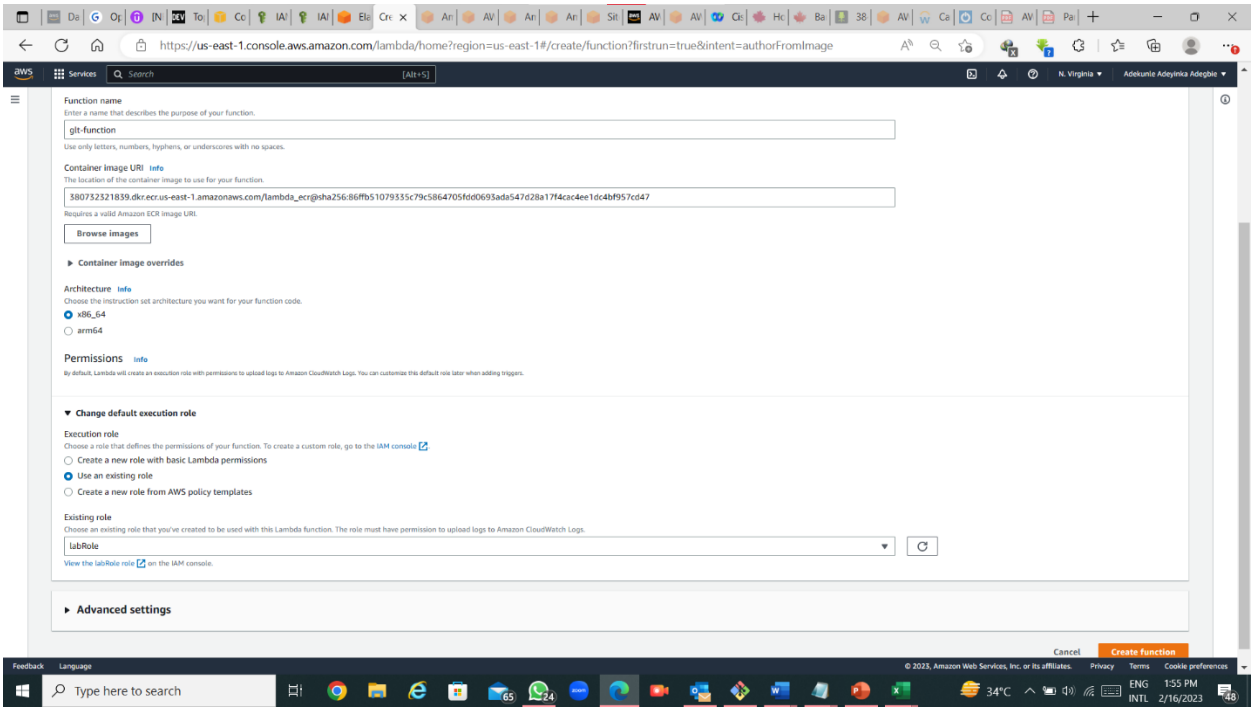


Step 3-(ii)

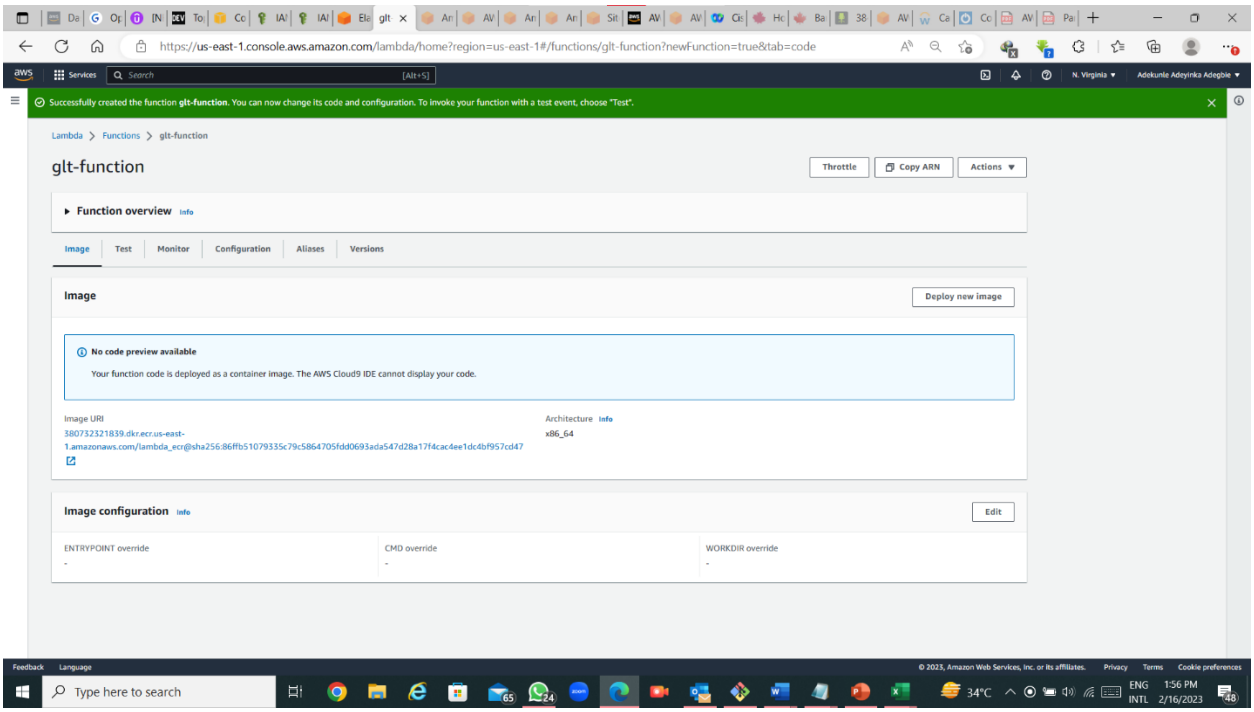


.

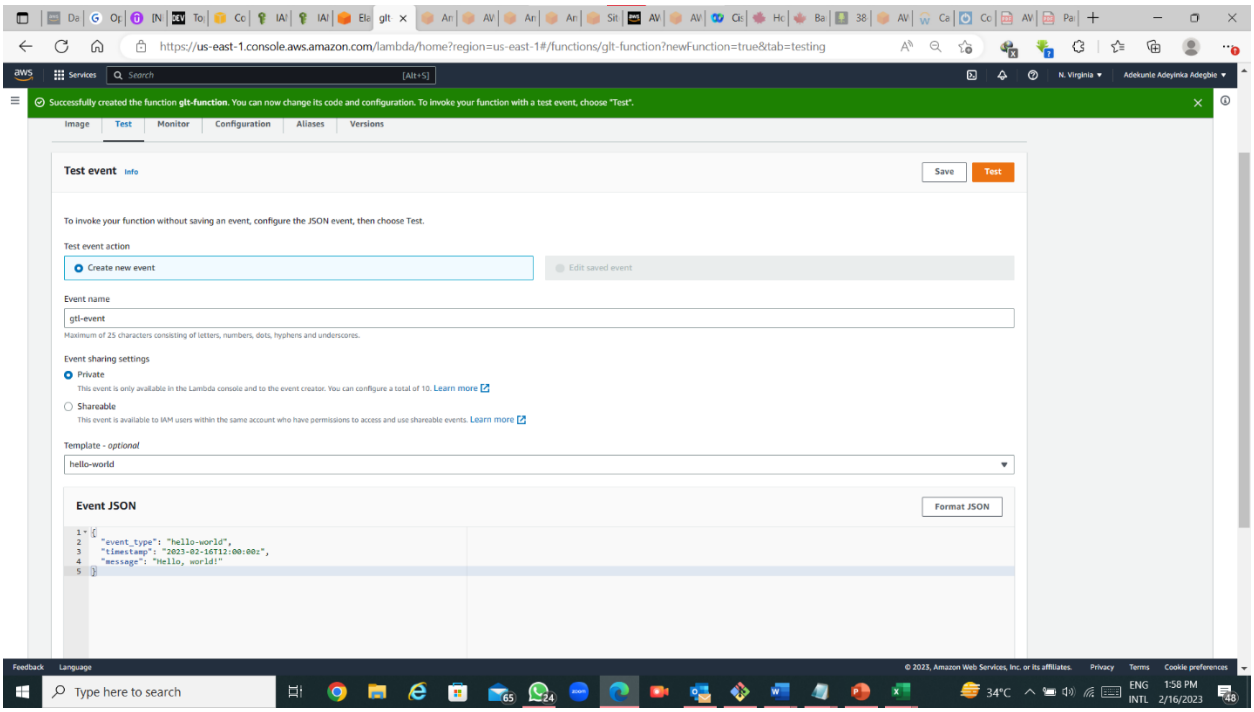
Step 3-(iii)



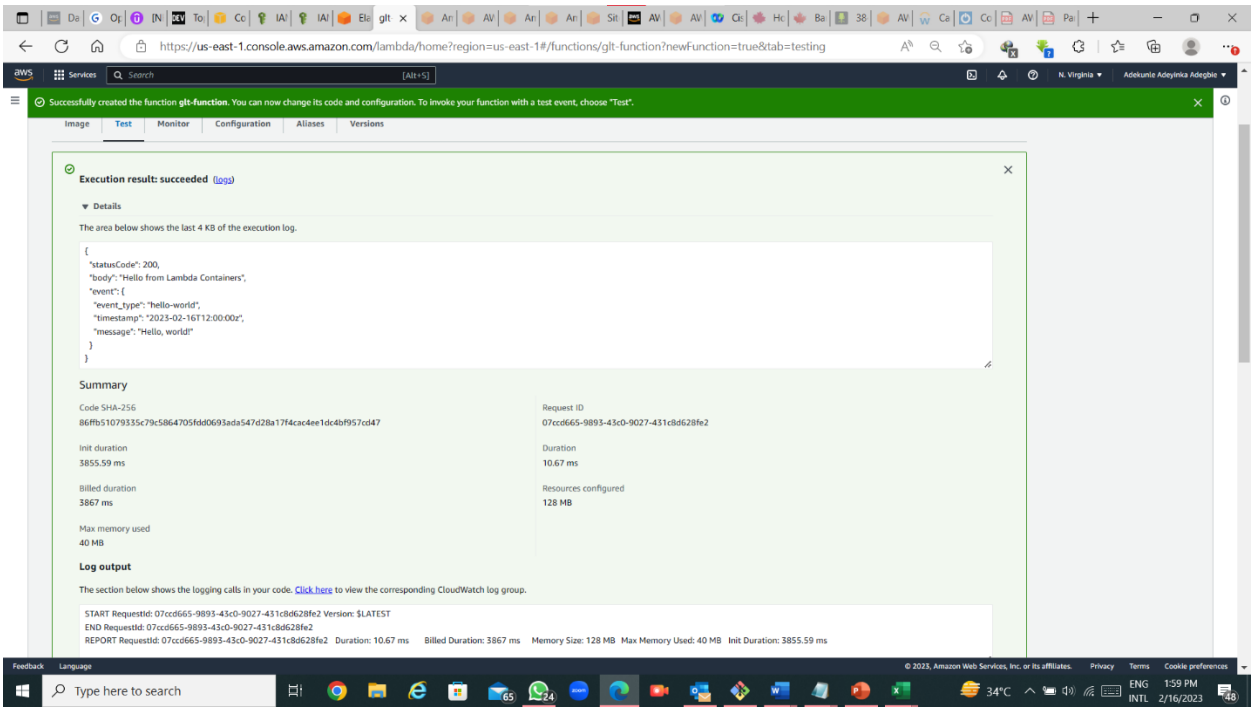
Step 3-(iv)



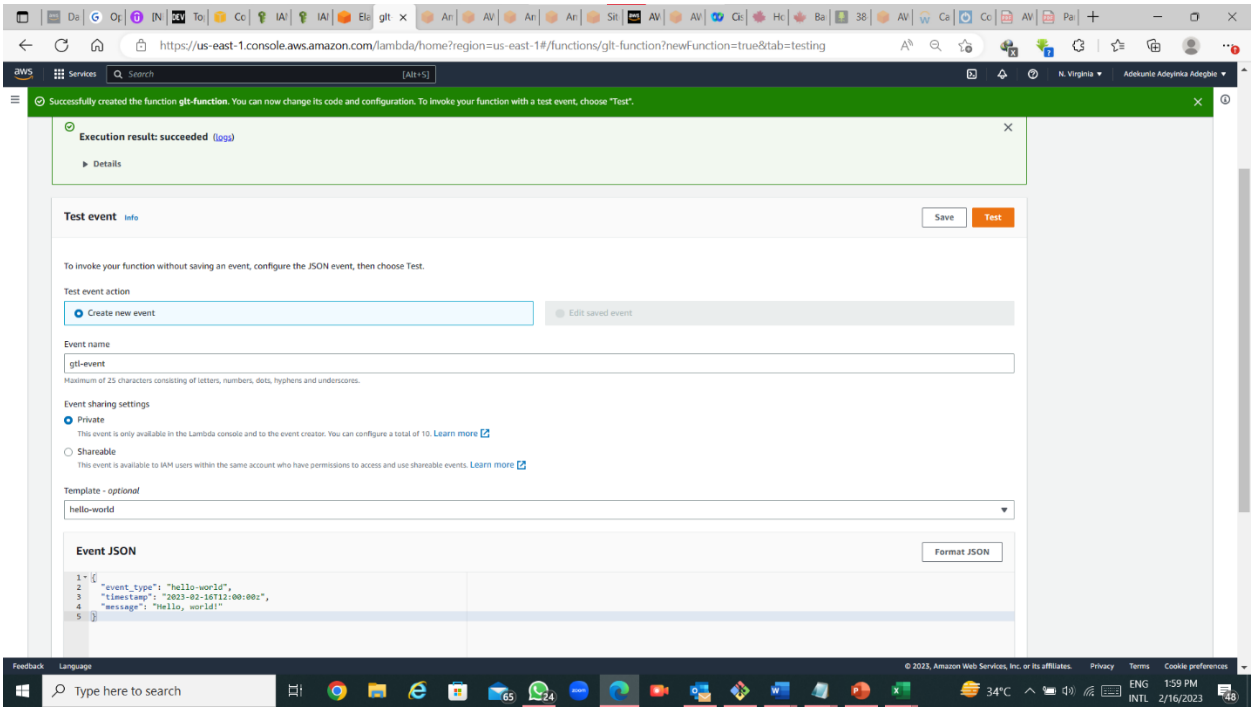
Step 3-(v)



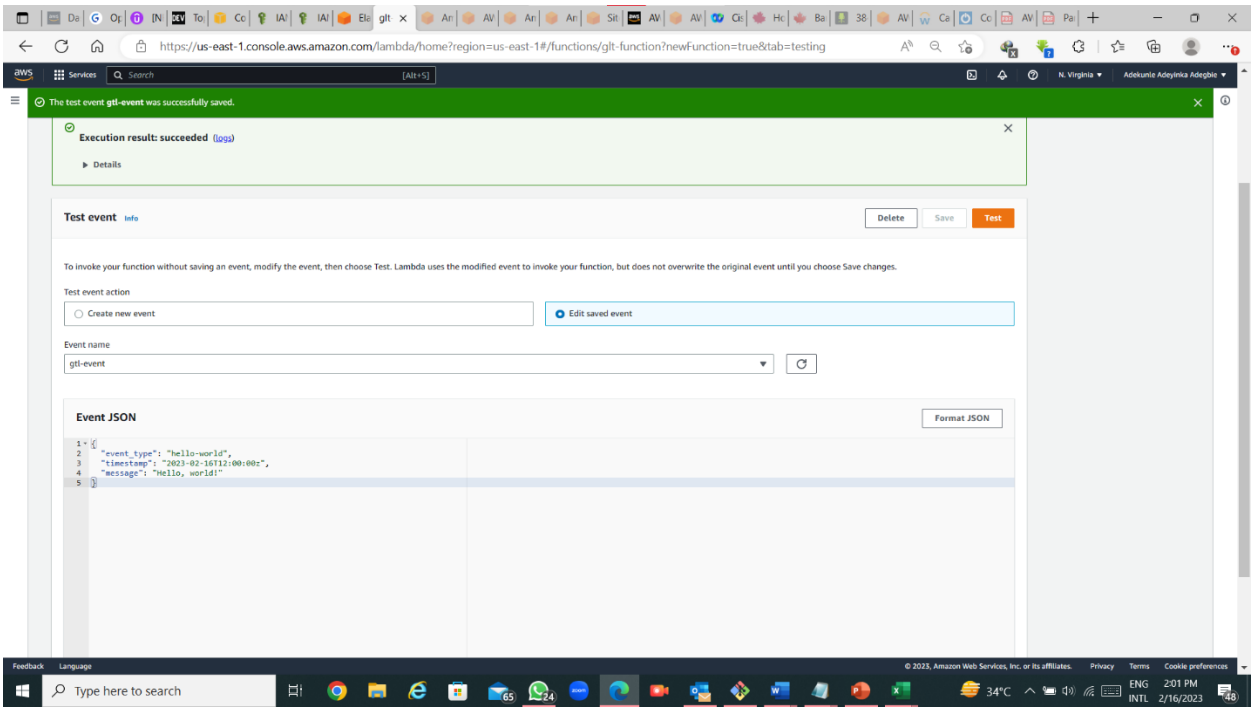
Step 3-(vi)



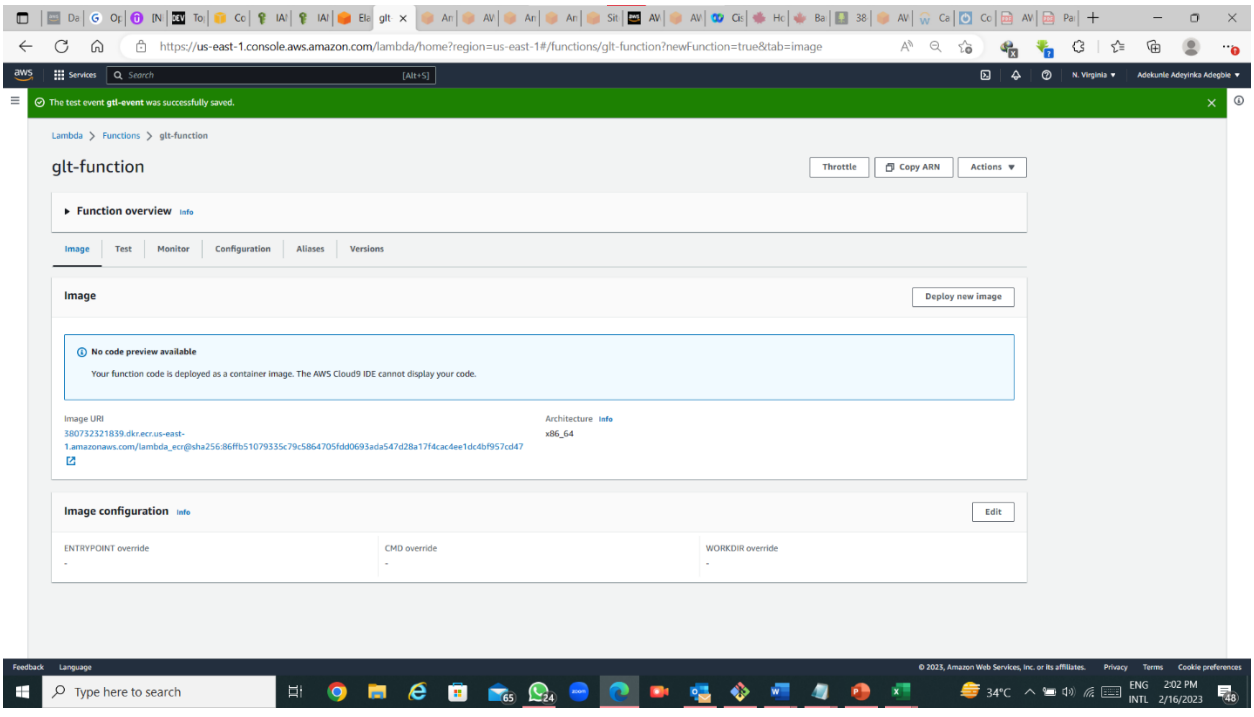
Step 3-(vii)



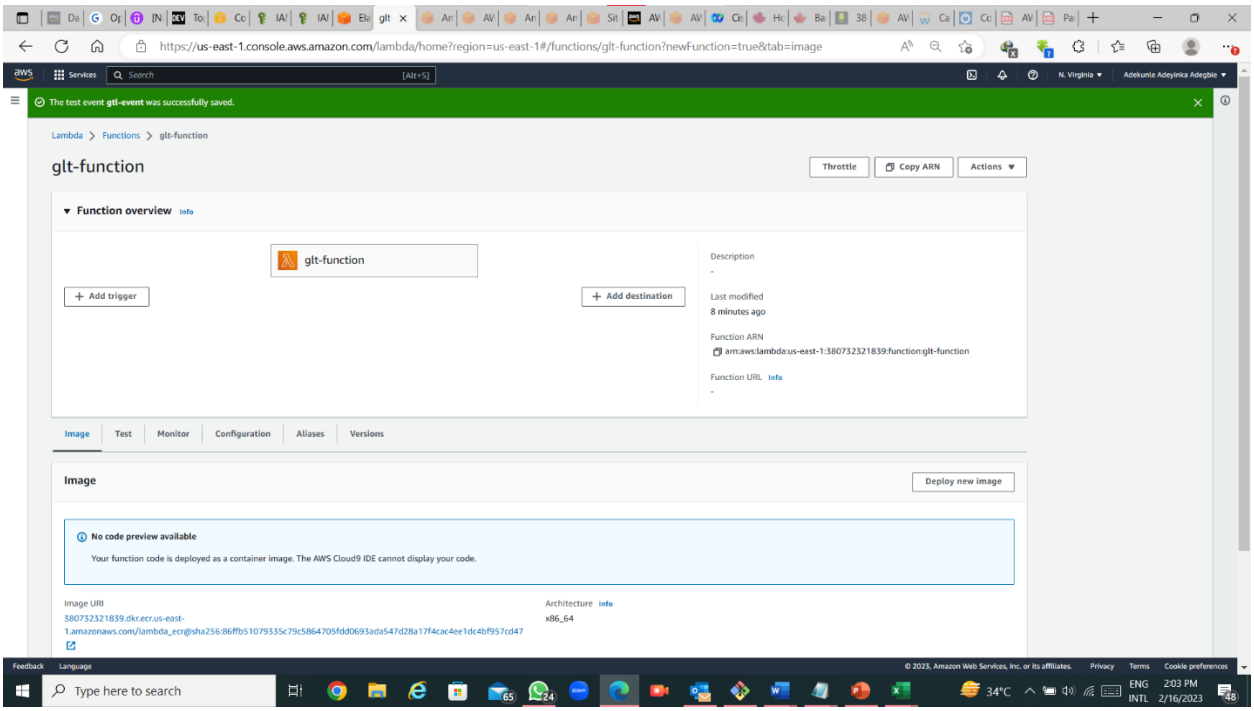
Step 3-(viii)



Step 3-(ix)



Step 3-(x)


https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/glt-function?newFunction=true&tab=image
The test event gtl-event was successfully saved.
Lambda > Functions > glt-function
glt-function
Throttle
Copy ARN
Actions
Function overview Info
glt-function
Add trigger
Add destination
Description
-
Last modified
8 minutes ago
Function ARN
arn:aws:lambda:us-east-1:380732321839:function:glt-function
Function URL Info
-
Image
Test
Monitor
Configuration
Aliases
Versions
Image
Deploy new image
No code preview available
Your function code is deployed as a container image. The AWS Cloud9 IDE cannot display your code.
Image URI
380732321839.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr@sha256:86ffb51079335c79c5864705fdd0693ada547d28a17f4cac4ee1dc4bf957cd47
Architecture Info
x86_64
Feedback
Language
© 2023, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
34°C
ENG
INTL
2:03 PM
2/16/2023

Answer the following questions

Q1 How long does a container stay in the running state if it is not manually halted?

a) As long as the container's PID 1 is running

b) Has a set timeout after which it pauses

c) Until its container is expunged

d) Docker daemon process scheduler decides on load

Enter your answer here | A

Q2 Which of the following best illustrates the relationship between an image and a container?

a) Executable and its hard link

b) Executable and process

c) Parent and child process

d) Many to one

Enter your answer here | B

Q3 What is the maximum amount of RAM a container can consume if the memory flag is not used?

a) 8GiB

b) 32GiB

c) None of these

d) As much as the host instance has free

Enter your answer here | B

Q4 Which of the following will happen in the same Docker image is pushed to Docker Hub multiple times with different tags

a) Dockerhub will refuse to upload the image

b) The layers in the first image (if unchanged) will be reused in subsequent pushes

c) Dockerhub will merge the images

d) The same image cannot have multiple tags

Enter your answer here | B

Q5 Which of the following will run a Docker container in interactive mode?

a) -v

b) -it

c) -b

d) -u

Enter your answer here | B

Q6 How would data persistence be handled in a container environment set up for autoscaling?

Data persistence in a container environment set up for autoscaling can be a challenge because containers are designed to be ephemeral, meaning that they can be created and destroyed dynamically as needed. Therefore, if you have persistent data that needs to be stored across container instances, you will need to implement a strategy for managing that data.

One approach is to use external storage solutions that are decoupled from the container environment. This can include solutions like network-attached storage (NAS), storage area networks (SAN), or cloud-based object storage. By using an external storage solution, you can keep your data separate from your containers, and ensure that it is available to all container instances in the autoscaling group.

Another approach is to use container-specific solutions for data persistence, such as container volumes or data-only containers. These solutions allow you to store data within the container environment and make it available to all container instances in the autoscaling group. However, it is important to note that this approach can increase complexity and potentially create data consistency issues, especially when multiple containers need to write to the same volume simultaneously.

In addition to data persistence, it's also important to consider data replication and synchronization in an autoscaling container environment. This can involve implementing solutions such as database clustering, or replication, to ensure that data is consistently available across all container instances. It may also be necessary to implement solutions for data backups and disaster recovery, to protect against data loss in the event of a failure or outage.

Q7 Why is this statement false? "Docker is the only popular choice for microservices deployment".

The statement "Docker is the only popular choice for microservices deployment" is false because there are other popular choices for microservices deployment, in addition to Docker. While Docker is a widely used containerization platform and is often associated with microservices deployment, it is not the only option available.

Some other popular choices for microservices deployment include:

1. Kubernetes: Kubernetes is an open-source container orchestration platform that can be used to manage and deploy microservices across a distributed infrastructure.
2. Apache Mesos: Apache Mesos is a cluster manager that provides resource isolation and sharing across distributed applications, making it well-suited for microservices deployment.
3. AWS Elastic Beanstalk: AWS Elastic Beanstalk is a cloud-based platform that enables developers to deploy, manage, and scale web applications and services, including microservices.
4. Google App Engine: Google App Engine is a platform-as-a-service (PaaS) offering that enables developers to build and deploy applications, including microservices, on Google's cloud infrastructure.
5. Microsoft Azure Service Fabric: Microsoft Azure Service Fabric is a distributed systems platform that provides a foundation for building and deploying microservices-based applications.

These are just a few examples of other popular choices for microservices deployment that are available. The choice of platform will depend on factors such as the specific requirements of the application, the organization's existing infrastructure and tools, and the preferences and expertise of the development team.

Grades distribution	
MCQs	5 (1 point each)
Subjective questions	11 points (5+6)
Implementation screenshots	24 points (2 point each)
Total	40 points