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	MOHAMED EL HASSNAOUI
	<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 PGPCCJUN24_MOAHMED_ELHASSNAOUI_OPTION2. 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.

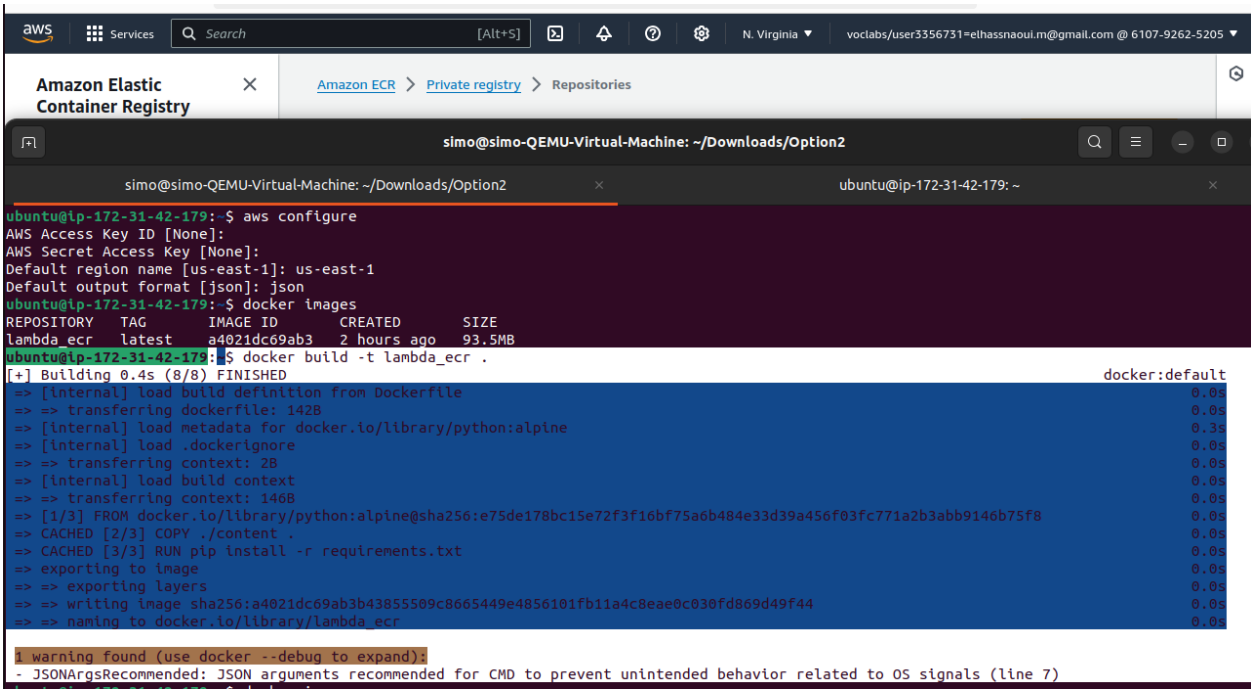
Architecture diagram

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

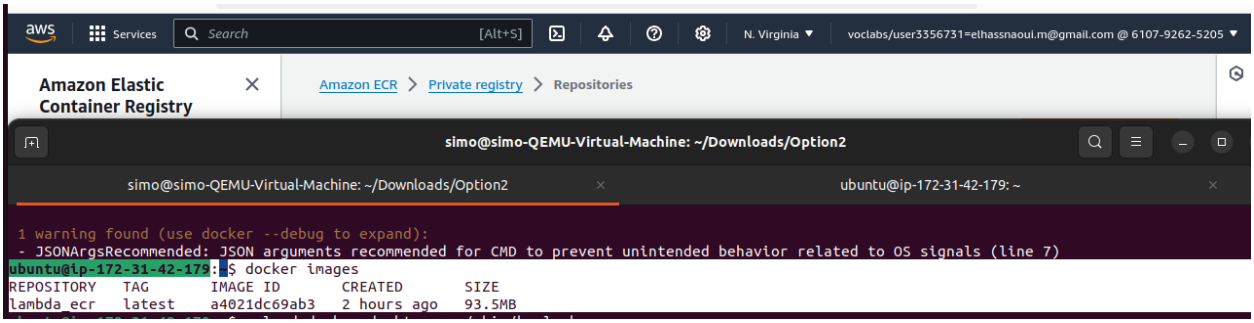
Step number	a
Step name	Creation of Docker image
Instructions	1) Create an EC2 instance using the Ubuntu 22.04 in the default VPC. 2) Attach the role “LabInstanceProfile” to the instance created above. This role is already present on your AWS Academy account. Note : For personal AWS accounts, create a new IAM role for EC2 and add the policy “EC2ContainerRegistryFullAccess” to it. 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 ubuntu@<public IP of instance>:/home/ubuntu (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 wget https://d6opu47qoi4ee.cloudfront.net/dockerinstallscript.sh bash ./dockerinstallscript.sh sudo apt install unzip unzip OCI.zip 5) **(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 apt 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** 6) Run the below command to create the Docker image *docker build -t lambda_ecr .* 7) Run the below command to verify the creation of the image *docker images*
Expected screenshots	1) Building the Docker image 2) List of the created images

3) **<Insert screenshot for a(1) here -** Building the Docker image



>

4) **<Insert screenshot for a(2) here:** List of the created images

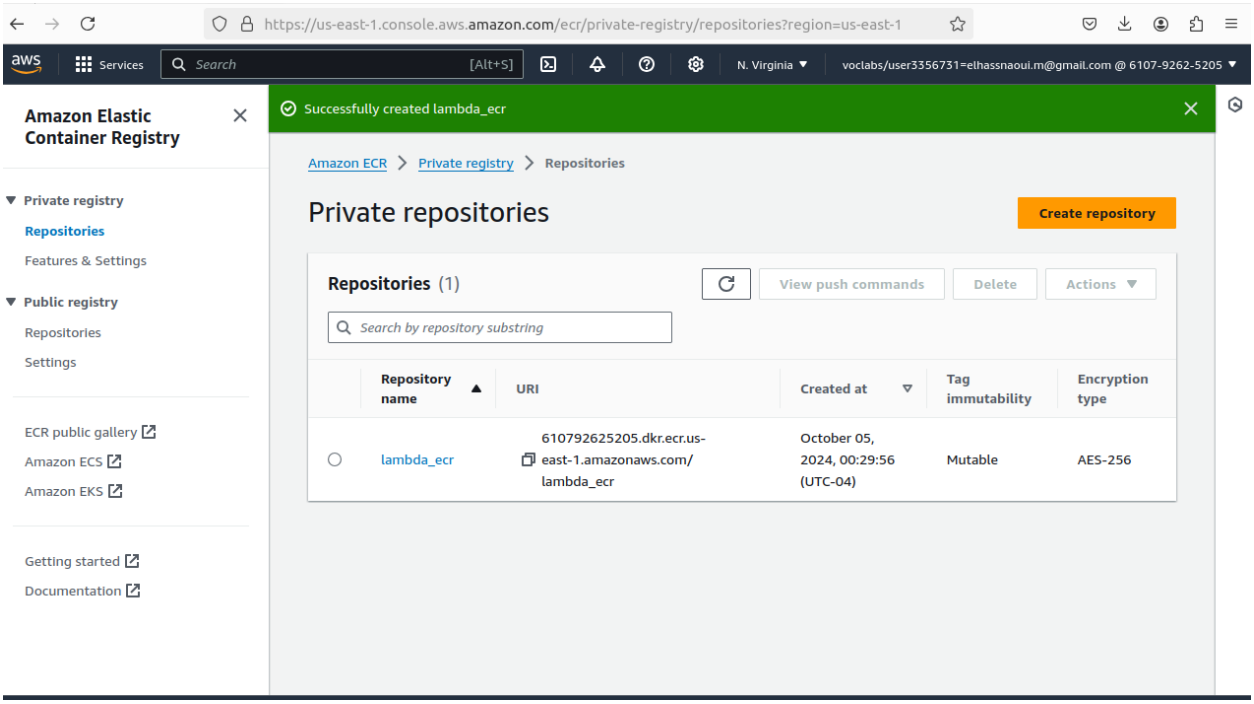


>

Step 2: Create ECR repository and upload image to ECR

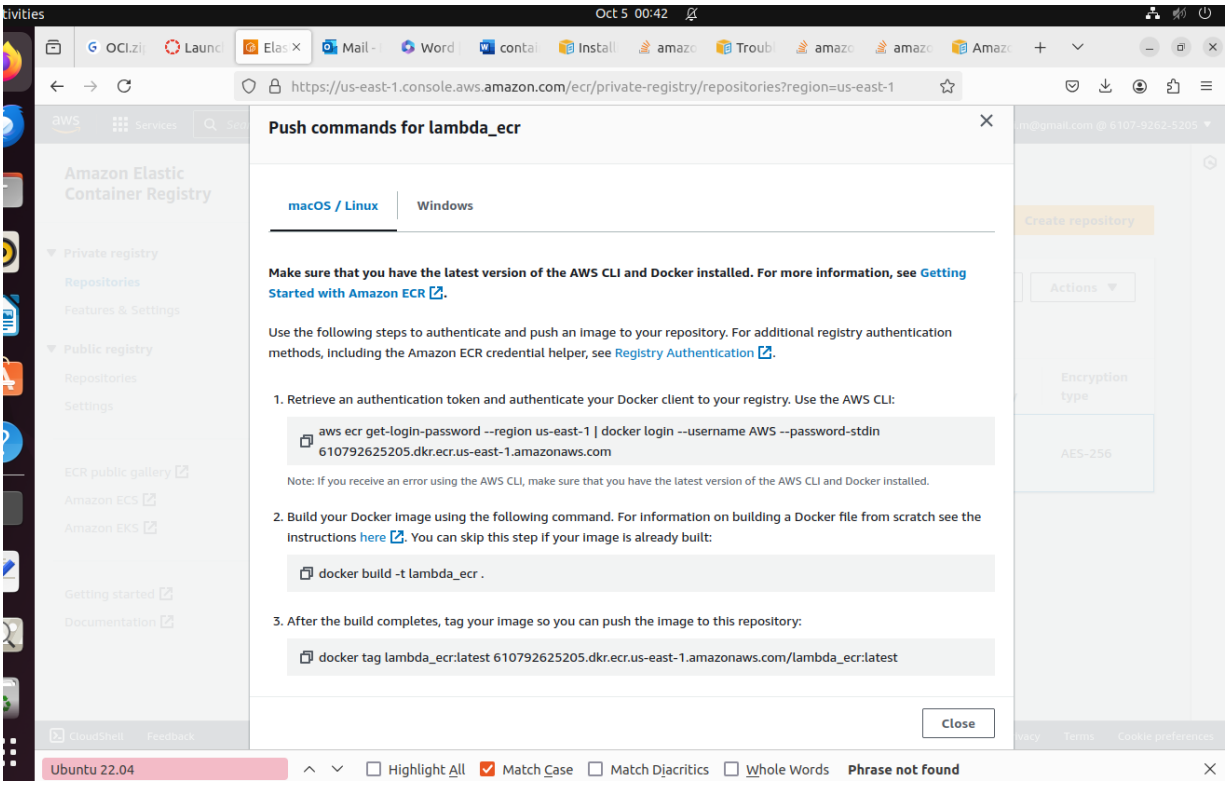
Step number	a
Step name	Creating the ECR repository
Instructions	1) Go to the ECR service on the AWS console 2) Select the Repositories from the left pane 3) Create a new private repository named **lambda_ecr** with the default settings
Step number	b
Step name	Image upload to ECR
Instructions	1) Once 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

<Insert screenshot for a(1) here - Creating the ECR repository



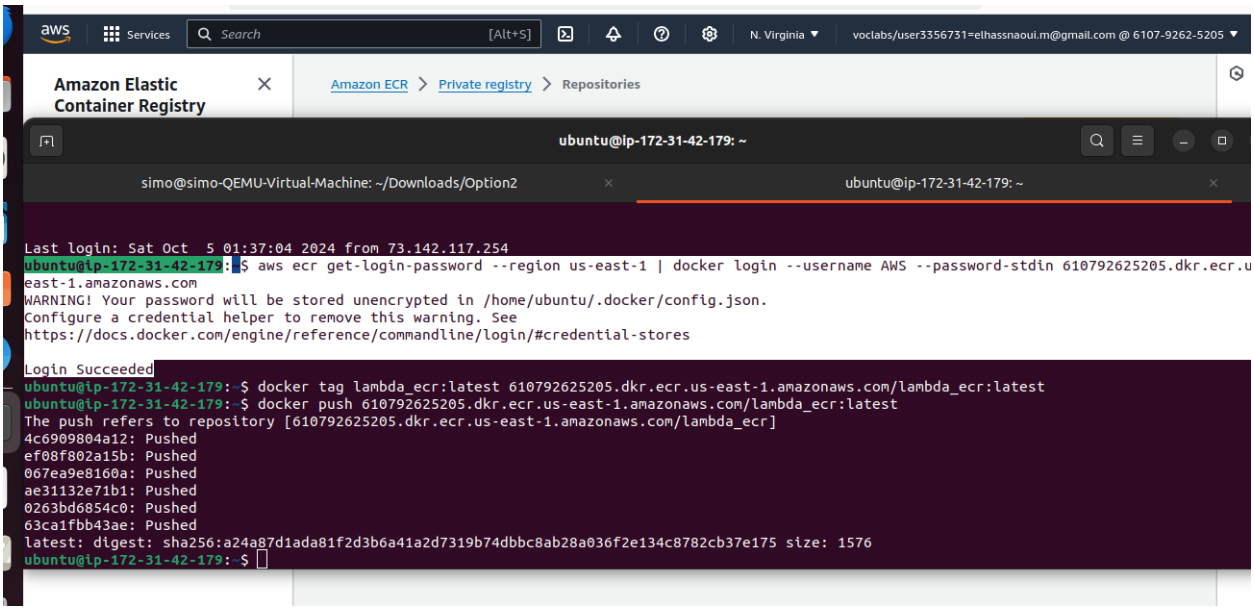
>

<Insert screenshot for b(1) here - View push commands



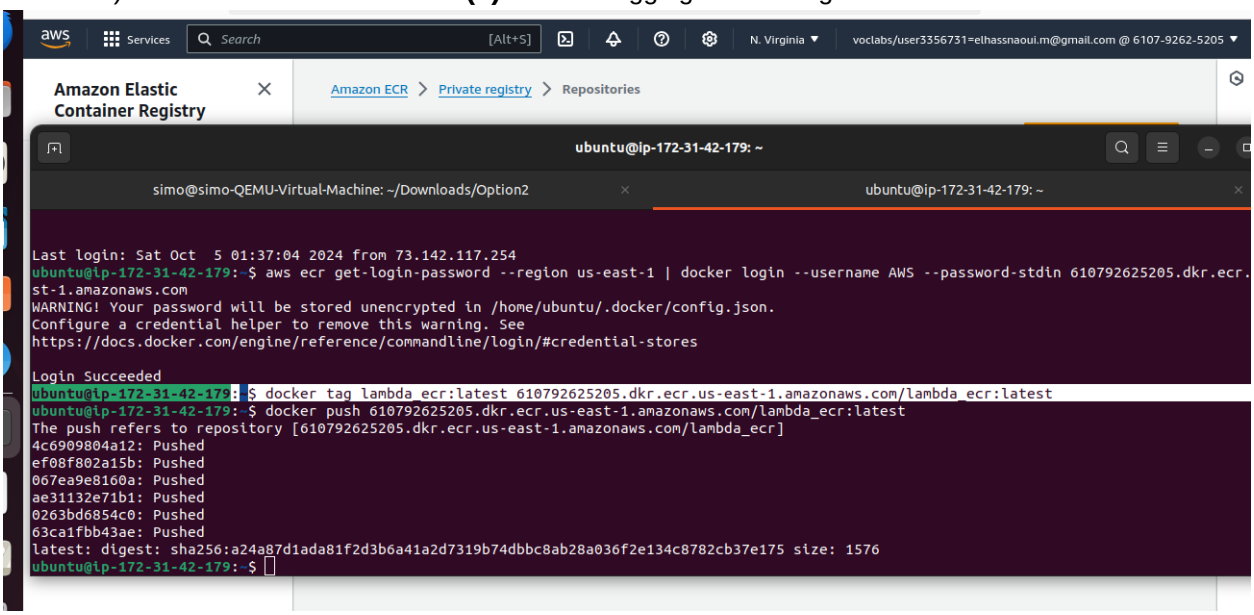
>

<Insert screenshot for b(2) here - Login Succeeded



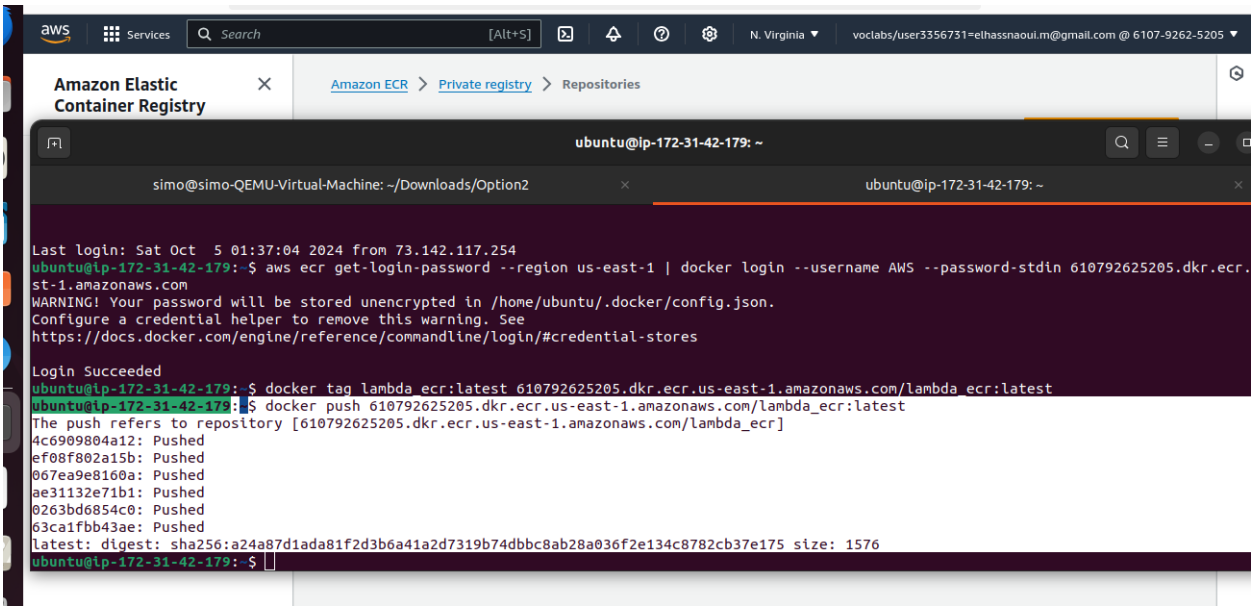
>

7) **<Insert screenshot for b(3) here:** Tagging of the image



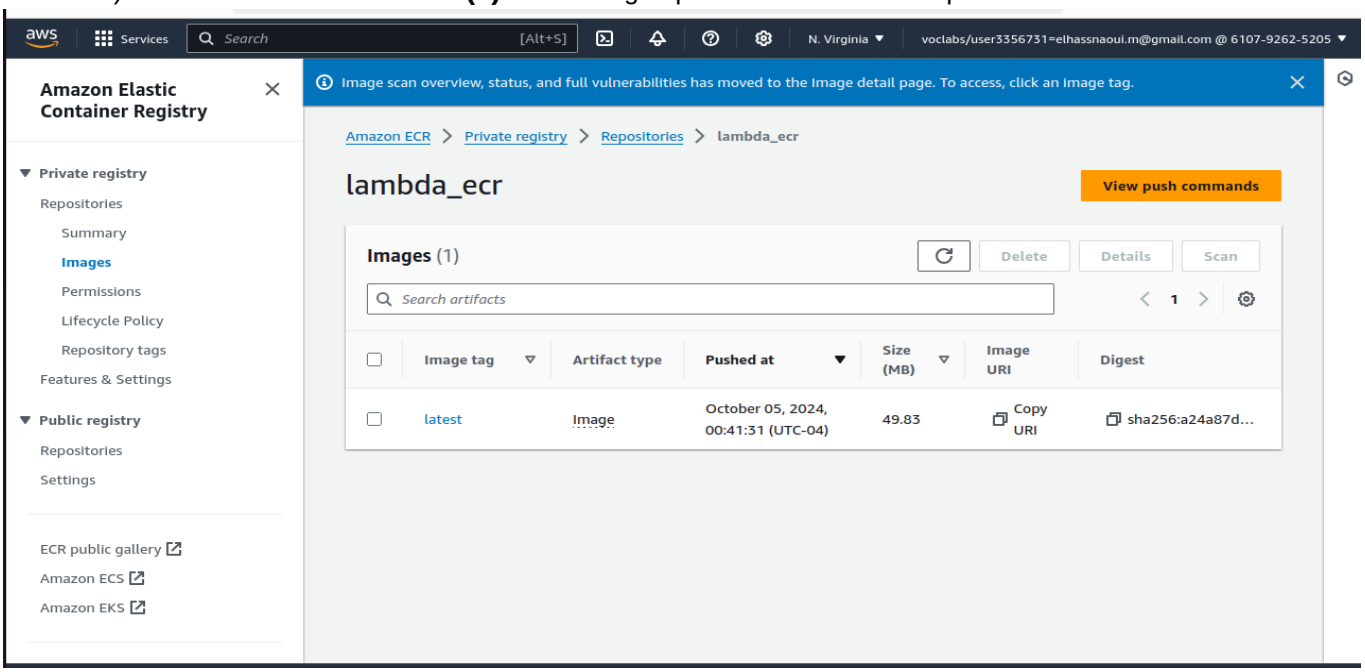
>

8) **<Insert screenshot for b(4) here:** Pushing of image to ECR



>

9) **<Insert screenshot for b(5) here** Image uploaded on the ECR repo

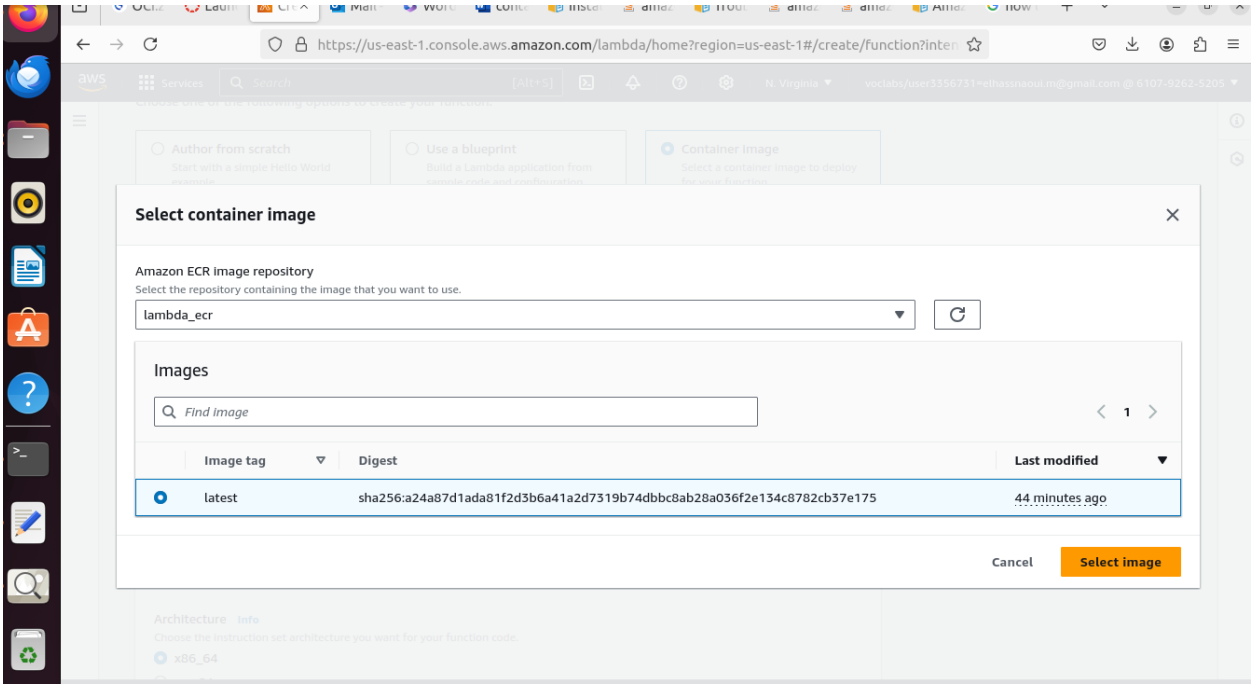


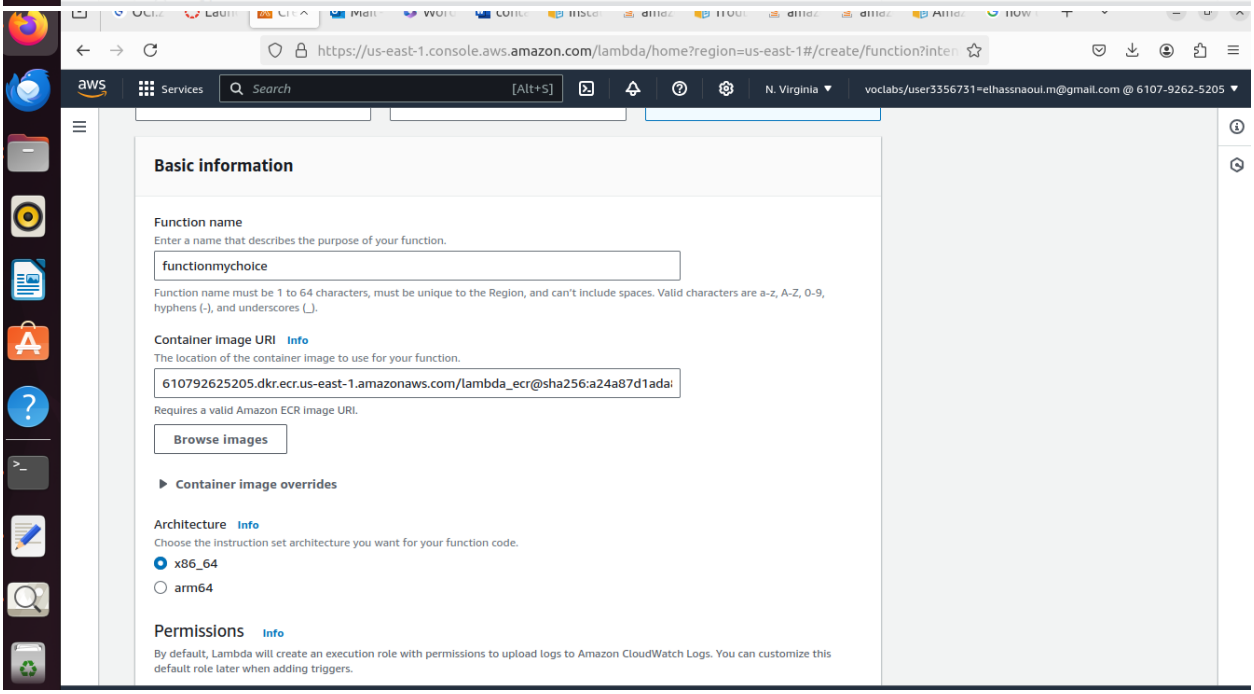
>

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. This role is already present on your AWS Academy accounts. Note : For personal AWS accounts, create a new IAM role for EC2 and add the policy "EC2ContainerRegistryFullAccess" to it. 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

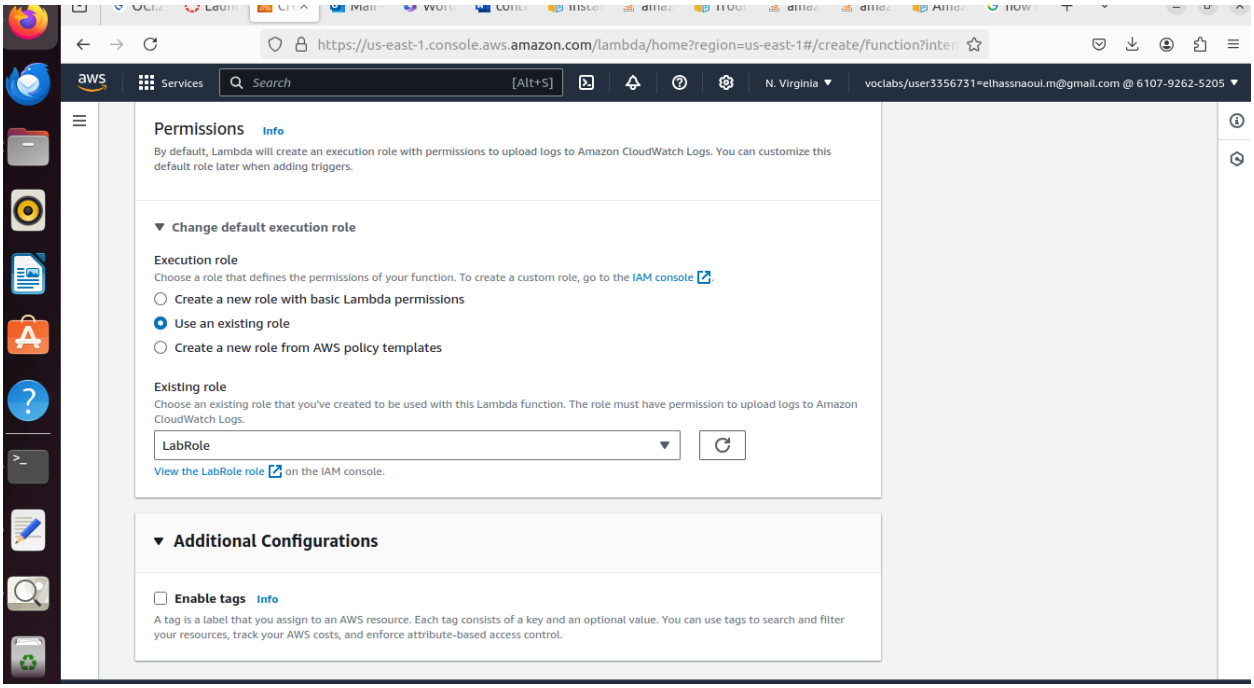
<Insert screenshot for a(1) here: Container image selection





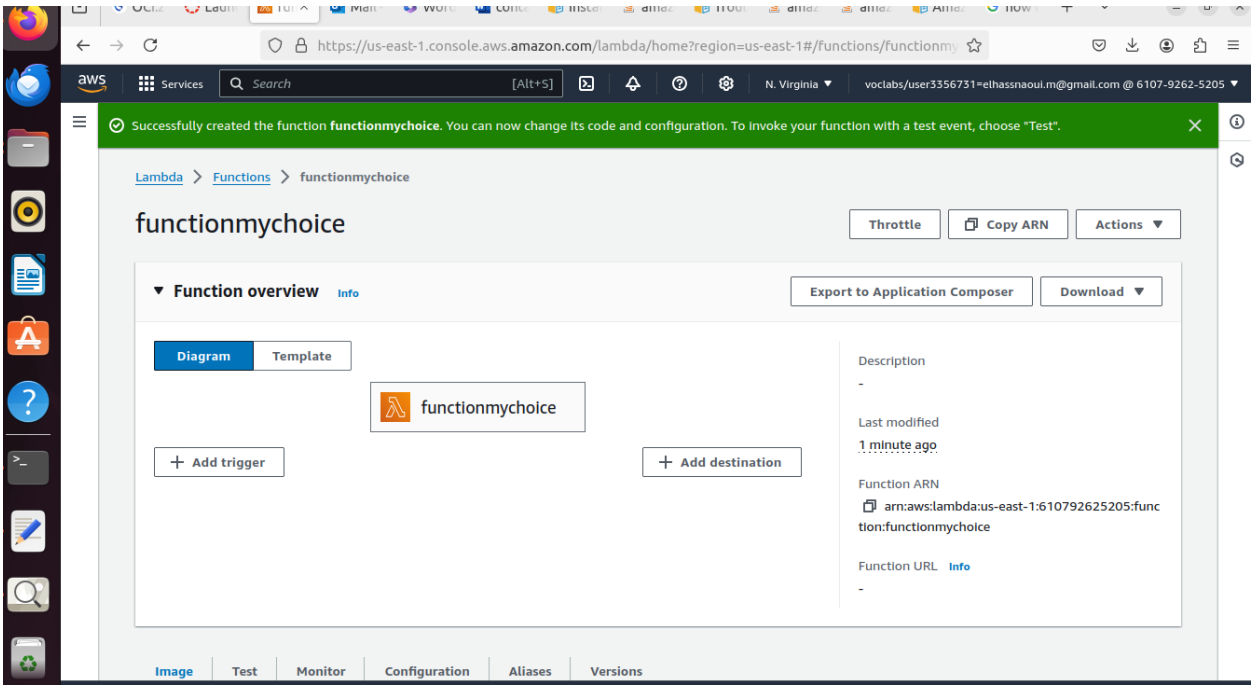
>

<Insert screenshot for a(2) here: Execution role selection



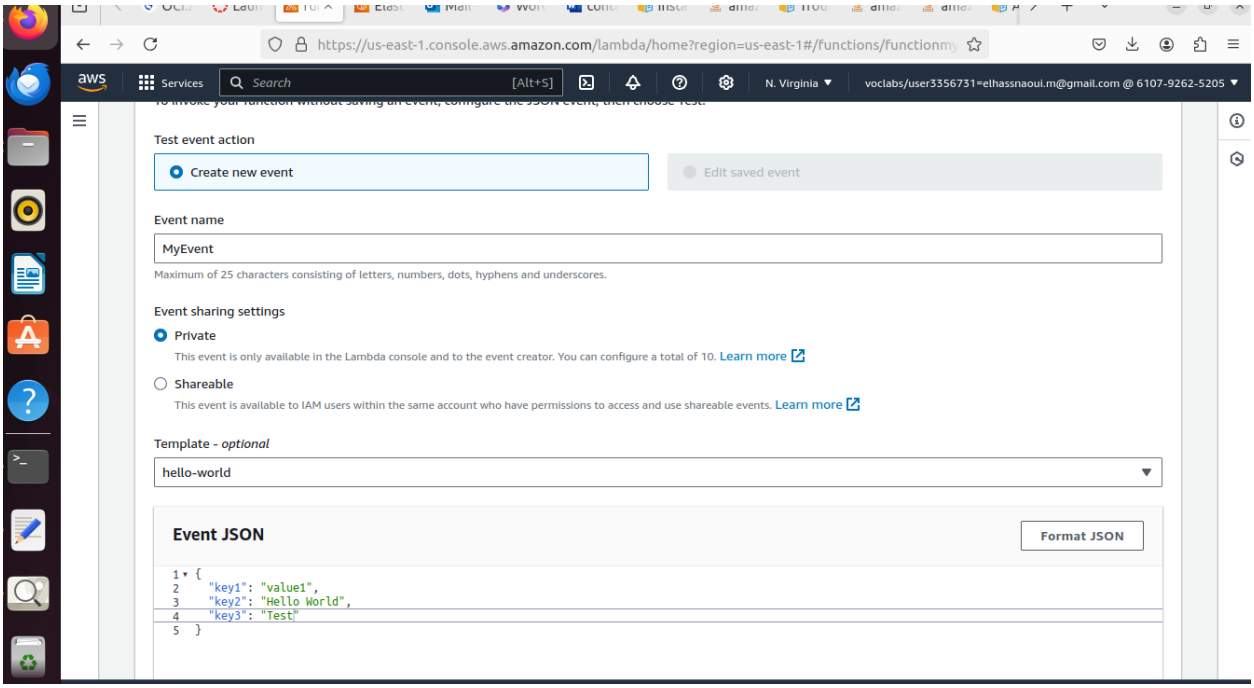
>

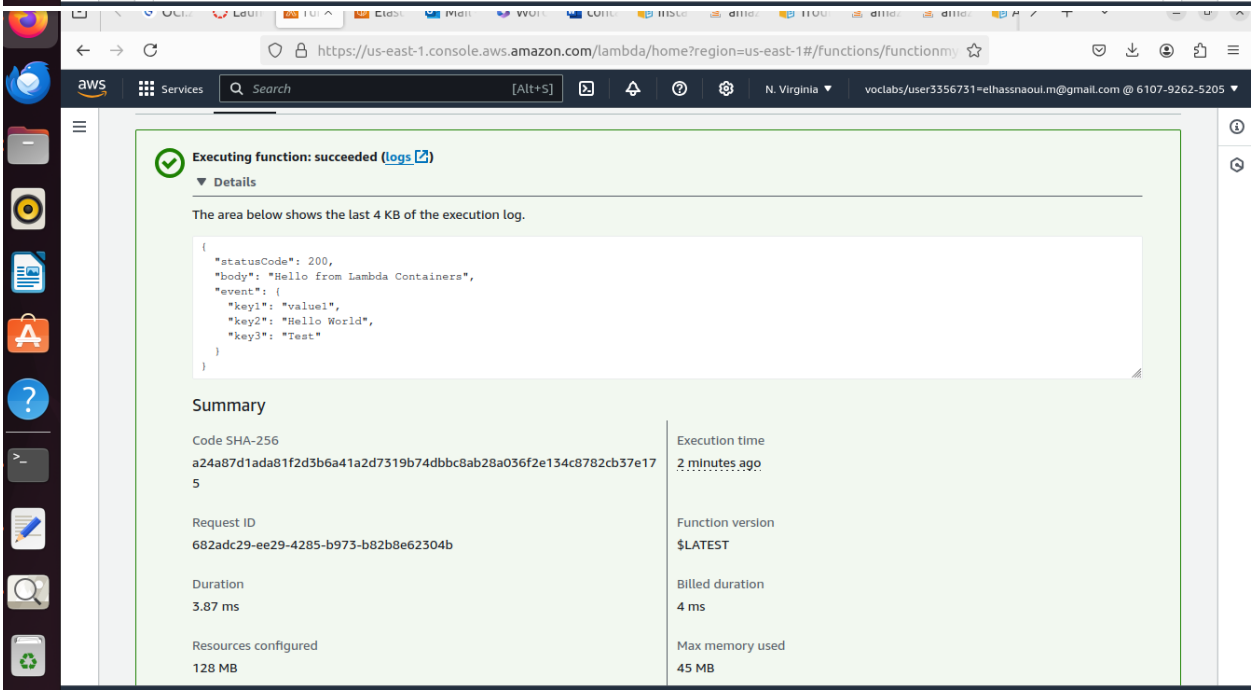
<Insert screenshot for a(3) here: Created function



>

<Insert screenshot for a(4) here: Test result of function





>

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

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) Executable and process

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

d) As much as the host instance has free

Q4 Which of the following will happen if 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) The layers in the first image (if unchanged) will be reused in subsequent pushes

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) -it

Q6 How should data persistence be handled in a container environment configured for autoscaling?

Data Persistence in Autoscaling Container Environments

Ensuring data persistence in autoscaling containerized environments requires selecting the appropriate storage mechanisms. The following approaches address different use cases:

8) **Volumes**: Docker volumes or Amazon EBS (Elastic Block Store) volumes provide persistent storage at the container level. These options ensure data retention across container restarts and are suitable for stateful applications.
9) **Managed Persistent Storage Services**: AWS services such as Elastic File System (EFS) or FSx offer scalable, managed file systems. These are ideal for scenarios where multiple containers need to access shared storage, ensuring high availability and data durability.
10) **Database Services**: Managed database solutions like Amazon RDS, Aurora, and DynamoDB offer automated scaling and high availability for structured data. These services handle operational aspects, such as backups and failover, making them a robust choice for relational or NoSQL data persistence.
11) **State Management Tools**: Kubernetes StatefulSets and similar orchestration mechanisms provide stateful container management. These tools maintain persistent storage and network identities for pods, ensuring consistency and stability in data handling across pod rescheduling or scaling events.
12) **Application-Level Persistence**: In-memory databases (e.g., Redis) or file-based persistence can be used for non-critical data where short-term retention or performance is a priority. This approach may not provide long-term durability but can optimize response times for transient data.

Selection Criteria: The appropriate data persistence strategy should be chosen based on key factors such as data sensitivity, performance requirements, scalability needs, and cost-efficiency. Aligning the storage solution with the application's architecture and operational demands is critical for maintaining performance and ensuring reliable data availability.

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

The statement is false because **Docker is not the only popular choice** for microservices deployment. Other widely-used containerization and orchestration tools include **Kubernetes**, **Podman**, and **OpenShift**. Additionally, **serverless platforms** like AWS Lambda and **container services** like AWS Fargate and Azure Container Instances are also popular for deploying microservices, providing alternatives to Docker.

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