

| | |
|---|---|
| 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> |

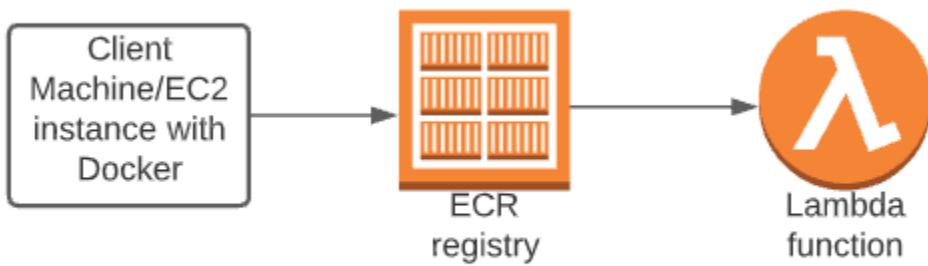
| |
|--|
| Instructions |
| Every screenshot requested in this workbook is compulsory and carries 1 point |
| Your AWS account ID must be clearly visible in every screenshot using the AWS console; missing id or using someone else's id is not permitted. Such cases will be considered as plagiarism and 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 | <ol style="list-style-type: none">1) Create an EC2 instance using the Amazon Linux 2 AMI in the default VPC.2) Attach the role “LabInstanceProfile” to the instance created above3) Download the file OCI.zip provided with this workbook and copy it to the EC2 instance using the scp command <code>scp -i <pem file name> ./OCI.zip ec2-user@<public IP of instance>:/home/ec2-user</code> (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 <code>sudo yum update</code> <code>sudo yum install unzip</code> <code>sudo unzip OCI.zip</code> <code>sudo amazon-linux-extras install docker</code> <code>sudo service docker start</code> <code>sudo usermod -a -G docker ec2-user</code> (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) <code>sudo yum install awscli -y</code> <code>aws configure</code> Skip the access key and secret access key fields by pressing the Enter key. Enter the region as us-east-1 and format as json5) Run the below command to create the Docker image <code>sudo docker build -t lambda_ecr .</code>6) Run the below command to verify the creation of the image <code>sudo docker images</code> |

Expected screenshots 1)Building the Docker image 3) List of the created image

Step 1-(i)

The screenshot shows the AWS Management Console interface for the EC2 service. The main title bar says "Successfully attached LabInstanceProfile to instance i-052e211f7b3280a11". The left sidebar has sections for New EC2 Experience, EC2 Dashboard, EC2 Global View, Events, Tags, Limits, Instances (selected), Instance Types, Launch Templates, Spot Requests, Savings Plans, Reserved Instances, Dedicated Hosts, Scheduled Instances, Capacity Reservations, Images, AMIs, AMI Catalog, Elastic Block Store, Volumes, Snapshots, Lifecycle Manager, Network & Security, Security Groups, Elastic IPs, Placement Groups, and Feedback. The main content area shows a table titled "Instances (1 / 1) Info" with one row for "web-app" (i-052e211f7b3280a11). The instance is running, t2.micro, with 2/2 checks passed, no alarms, in us-east-1a, with a Public IPv4 DNS of ec2-18-215-167-178.co... and a Public IPv4 IP of 18.215.167.178. The bottom status bar shows the date as 2/16/2023 and the time as 12:22 PM.

Step 1-(ii)

The screenshot shows a Windows terminal window titled "cmd" with the command prompt "ec2-user@ip-172-31-16-172~". The user is attempting to SSH into the instance using the command "ssh -i "jan16-kp.pem" ec2-user@ec2-18-215-167-178.compute-1.amazonaws.com". The terminal displays the host key fingerprint and asks if the user wants to continue connecting. The user responds with "yes". The terminal also shows the Amazon Linux 2 AMI logo. The bottom status bar shows the date as 2/16/2023 and the time as 12:24 PM.

```
ec2-user@ip-172-31-16-172~
$ ssh -i "jan16-kp.pem" ec2-user@ec2-18-215-167-178.compute-1.amazonaws.com
The authenticity of host 'ec2-18-215-167-178.compute-1.amazonaws.com (18.215.167.178)' can't be established.
ECDSA key fingerprint is SHA256:tc2jpdXwfeCbiVqvTJKuwoPvoxT0kwbJlFOwPlyelFQk.
This key is not known by any other names.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
warning: Permanently added 'ec2-18-215-167-178.compute-1.amazonaws.com' (ED25519)
) to the list of known hosts.

Amazon Linux 2 AMI

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-172-31-16-172 ~]$ |
```

Step 1-(iii)

```
sunile@Kunile-PC MINGW64 /  
$ cd  
sunile@Kunile-PC MINGW64 ~  
$ cd downloads  
sunile@Kunile-PC MINGW64 ~/downloads  
$ chmod 400 jan16-kp.pem  
sunile@Kunile-PC MINGW64 ~/downloads  
$ ssh -i "jan16-kp.pem" ec2-user@ec2-18-215-167-178.compute-1.amazonaws.com  
The authenticity of host "ec2-18-215-167-178.compute-1.amazonaws.com (18.215.167.178)" can't be established.  
RSA key fingerprint is SHA256:TcTp02wFeCbiVqvTJKuwoPvoxT0kwbjF0wP1yeLFQk.  
This key is not known to any other names.  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
warning: Permanently added 'ec2-18-215-167-178.compute-1.amazonaws.com' (ED25519)  
) to the list of known hosts.  
_ _|_ / Amazon Linux 2 AMI  
_ \_ |_  
https://aws.amazon.com/amazon-linux-2/  
[ec2-user@ip-172-31-16-172 ~]$ ip a  
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default  
    qlen 1000  
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00  
    inet 127.0.0.1/8 scope host lo  
        valid_lft forever preferred_lft forever  
        inet6 ::1/128 scope host  
            valid_lft forever preferred_lft forever  
2: eth0: <NO-CARRIER,BROADCAST,MULTICAST,UP,LOWER_UP> mtu 9001 qdisc pfifo_fast state UP group default qlen 1000  
    link/ether 0a:c7:02:8a:14:27 brd ff:ff:ff:ff:ff:ff  
    inet 172.31.16.172/24 brd 172.31.31.255 scope global dynamic eth0  
        valid_lft 86397sec expires 86427sec  
        inet6 fe80::4c7:14ff:fe8a:1427/64 scope link  
            valid_lft forever preferred_lft forever  
[ec2-user@ip-172-31-16-172 ~]$ exit  
logout  
Connection to ec2-18-215-167-178.compute-1.amazonaws.com closed.  
sunile@Kunile-PC MINGW64 ~/downloads  
$ scp -i jan16-kp.pem ./OCI.zip ec2-user@18.215.167.178:/home/ec2-user  
The authenticity of host "18.215.167.178" can't be established.  
RSA key fingerprint is SHA256:TcTp02wFeCbiVqvTJKuwoPvoxT0kwbjF0wP1yeLFQk.  
This host key is known by the following other names/addresses:  
  ~/.ssh/known_hosts:81: ec2-18-215-167-178.compute-1.amazonaws.com  
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes  
warning: Permanently added '18.215.167.178' (ED25519) to the list of known hosts  
OCI.zip          100% 1595      7.2KB/s   00:00  
sunile@Kunile-PC MINGW64 ~/downloads  
$
```

Step 1-(iv)

```
ec2-user@ip-172-31-16-172 ~$  
_stable ]  
18 libreoffice           available [ _stable ]  
19 gimp                  available [ =5.0.6.2_15 =5.3.6.1 _stable ]  
20 docker-latest         available [ =2.8.22 ]  
21 mate-desktop-polkit  available [ _1.12.1 =18.03.1 =18.06.1 =18.09.9 _stable ]  
22 Granђинаgokl1.3     available [ _1.19.0 =1.20.0 _stable ]  
23 tomcat8.5             available [ _1.13.29 =1.3.32 =1.3.34 _stable ]  
24 openepel              available [ _stable ]  
25 testing               available [ =1.0 _stable ]  
26 ecs                  available [ =stable ]  
27 corretto8             available [ _1.8.0.212 =1.8.0.202 =1.8.0.202 =1.8.0.222 =1.8.0.232  
                                         _1.8.0.242 _stable ]  
28 goolang1.11           available [ _stable ]  
29 squid                available [ =1.11.3 =1.11.11 =1.11.13 _stable ]  
30 squid                available [ =4 _stable ]  
32 fusenet2.10           available [ _stable ]  
33 java-openjdk11        available [ _1.8.0.242 _stable ]  
34 lynis                available [ =stable ]  
36 GCC                  available [ =stable ]  
37 nmap                 available [ =stable ]  
38 nginxx               available [ =stable ]  
39 ruby2.6               available [ =stable ]  
40 modc                available [ =stable ]  
41 postgresql11          available [ =stable ]  
43 livepatch            available [ =stable ]  
44 python3.8             available [ =stable ]  
45 haproxy2              available [ =stable ]  
46 collectd              available [ =stable ]  
47 open-nitro-enclaves-cli available [ =stable ]  
48 R4                  available [ =stable ]  
_ kernel-5.4             available [ =stable ]  
50 selinux-ng            available [ =stable ]  
51 php8.0               available [ =stable ]  
52 curl7.69              available [ =stable ]  
53 unbound1.13           available [ =stable ]  
54 mariadb10.5            available [ =stable ]  
55 kernel-5.10/latest    enabled [ =stable ]  
56 curl7.68              available [ =stable ]  
57 ruby3.0               available [ =stable ]  
58 postgresql12           available [ =stable ]  
59 postgresql13           available [ =stable ]  
60 mock2                available [ =stable ]  
61 kernel-5.4.85          available [ =stable ]  
62 kernel-5.15             available [ =stable ]  
63 postgresql14           available [ =stable ]  
64 firefox               available [ =stable ]  
65 lustre                available [ =stable ]  
66 gnutls               available [ =stable ]  
67 awscli               available [ =stable ]  
[ec2-user@ip-172-31-16-172 ~]$
```

Step 1-(v)

```
ec2-user@ip-172-31-16-172:~$ /81eddb6f342: Pull complete
lambda: Pull complete
libcurl: Pull complete
libcurl-c82c61d: Pull complete
libcurl-3018634: Pull complete
Digests: sha256:84630610c68e7c97384bc6e10f5490ab7b8398c30cdfffefal39ae20c3407cda
Status: Downloaded newer image for python:alpine
--> /81eddb6f342
Step 7/4 : COPY ./content .
--> 11048fffcfa3
Step 8/4 : RUN pip install -r requirements.txt
--> Running in 02a891b788b8
Collecting boto3>=1.26.72-py3-none-any.whl (132 kB)
  Downloading boto3-1.26.72-py3-none-any.whl (132 kB) 132.7/132.7 kB 4.0 MB/s eta 0:00:00
Collecting requests>=2.25.0
  Downloading requests-2.28.2-py3-none-any.whl (62 kB) 62.8/62.8 kB 12.2 MB/s eta 0:00:00
Collecting botocore<1.30.0,>=1.29.72
  Downloading botocore-1.29.72-py3-none-any.whl (10.4 kB) 10.4/10.4 kB 58.0 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.9.7
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB) 20.0/20.0 kB 16.6 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 16.6 MB/s eta 0:00:00
Collecting charset-normalizer<4,>2
  Downloading charset_normalizer-3.0.1-cp311-musllinux_1_1_x86_64.whl (190 kB) 190.6/190.6 kB 32.7 MB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB) 61.5/61.5 kB 14.2 MB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.14-py2.py3-none-any.whl (140 kB) 140.6/140.6 kB 21.1 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB) 155.3/155.3 kB 26.2 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 37.6 MB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB) 11.0/11.0 kB 14.2 MB/s eta 0:00:00
Installing collected packages: charset-normalizer, urllib3, six, jmespath, idna, certifi, requests, python-dateutil, botocore, s3transfer, boto3
Successfully installed boto3-1.26.72 botocore-1.29.72 certifi-2022.12.7 charset-normalizer-3.0.1 idna-3.4 jmespath-1.0.1 python-dateutil-2.8.2 requests-2.28.2 s3transfer-0.6.0 six-1.16.0 urllib3-1.26.14
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: https://pip.pypa.io/warnings/dev

[notice] A new release of pip available: 22.3.1 => 23.0
[notice] To update, run: pip install --upgrade pip
Removing intermediate container 02a891b788b8
--> 9e9203e3315a
Step 4/4 : CMD python3 bootstrap.py
--> Running in 4034edfd6093
Removing intermediate container 4034edfd6093
--> 876818be91f6
Successfully built 876818be91f6
Successfully tagged lambda_ecr:latest
[ec2-user@ip-172-31-16-172 ~]$
```



Step 1-(vi)

```
ec2-user@ip-172-31-16-172:~$ /81eddb6f342
Step 7/4 : COPY ./content .
--> 11048fffcfa3
Step 8/4 : RUN pip install -r requirements.txt
--> Running in 02a891b788b8
Collecting boto3>=1.16.19
  Downloading boto3-1.26.72-py3-none-any.whl (132 kB) 132.7/132.7 kB 4.0 MB/s eta 0:00:00
Collecting requests>=2.25.0
  Downloading requests-2.28.2-py3-none-any.whl (62 kB) 62.8/62.8 kB 12.2 MB/s eta 0:00:00
Collecting botocore<1.30.0,>=1.29.72
  Downloading botocore-1.29.72-py3-none-any.whl (10.4 kB) 10.4/10.4 kB 58.0 MB/s eta 0:00:00
Collecting jmespath<2.0.0,>=0.9.7.1
  Downloading jmespath-1.0.1-py3-none-any.whl (20 kB) 20.0/20.0 kB 16.6 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 16.6 MB/s eta 0:00:00
Collecting charset-normalizer<4,>2
  Downloading charset_normalizer-3.0.1-cp311-musllinux_1_1_x86_64.whl (190 kB) 190.6/190.6 kB 32.7 MB/s eta 0:00:00
Collecting idna<4,>=2.5
  Downloading idna-3.4-py3-none-any.whl (61 kB) 61.5/61.5 kB 14.2 MB/s eta 0:00:00
Collecting urllib3<1.27,>=1.21.1
  Downloading urllib3-1.26.14-py2.py3-none-any.whl (140 kB) 140.6/140.6 kB 21.1 MB/s eta 0:00:00
Collecting certifi>=2017.4.17
  Downloading certifi-2022.12.7-py3-none-any.whl (155 kB) 155.3/155.3 kB 26.2 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 37.6 MB/s eta 0:00:00
Collecting six>=1.5
  Downloading six-1.16.0-py2.py3-none-any.whl (11 kB) 11.0/11.0 kB 14.2 MB/s eta 0:00:00
Installing collected packages: charset-normalizer, urllib3, six, jmespath, idna, certifi, requests, python-dateutil, botocore, s3transfer, boto3
Successfully installed boto3-1.26.72 botocore-1.29.72 certifi-2022.12.7 charset-normalizer-3.0.1 idna-3.4 jmespath-1.0.1 python-dateutil-2.8.2 requests-2.28.2 s3transfer-0.6.0 six-1.16.0 urllib3-1.26.14
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: https://pip.pypa.io/warnings/dev

[notice] A new release of pip available: 22.3.1 => 23.0
[notice] To update, run: pip install --upgrade pip
Removing intermediate container 02a891b788b8
--> 9e9203e3315a
Step 4/4 : CMD python3 bootstrap.py
--> Running in 4034edfd6093
Removing intermediate container 4034edfd6093
--> 876818be91f6
Successfully built 876818be91f6
Successfully tagged lambda_ecr:latest
[ec2-user@ip-172-31-16-172 ~]$
```



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 <xxxxxx.dkr.ecr.us-east-1.amazonaws.com>
```

```
docker tag lambda_ecr_image:latest <xxxxxx.dkr.ecr.us-east-  
1.amazonaws.com/lambda_ecr>:latest
```

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

- Expected
- 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)

The screenshot shows the AWS ECR console with a success message: "Successfully created repository lambda_ecr". The URL is https://us-east-1.console.aws.amazon.com/ecr/repositories?region=us-east-1. The page displays a table of private repositories, including "lambda_ecr" which was created on February 16, 2023, at 13:41:40 (UTC+01). The repository has a size of 51.7MB, is disabled, and uses AES-256 encryption.

Step 2-(ii)

```

root@ip-172-31-16-172:/home/ec2-user
Successfully built 876818be91f6
successfully tagged lambda_ecr:latest
[ec2-user@ip-172-31-16-172 ~]$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
lambda_ecr latest 876818be91f6 About a minute ago 51.7MB
lambda_ecr 85da6554f50c 5 days ago 51.7MB
[ec2-user@ip-172-31-16-172 ~]$ 
[ec2-user@ip-172-31-16-172 ~]$ 
[ec2-user@ip-172-31-16-172 ~]$ ls -ali
[ec2-user@ip-172-31-16-172 ~]$ 
[ec2-user@ip-172-31-16-172 ~]$ ls -ali
total 24
13241975 drwx----- 6 ec2-user ec2-user 169 Feb 16 12:35 .
4194789 drwxr-xr-x 3 root root 22 Feb 16 12:13 ..
4542465 drwxrwxr-x 2 ec2-user ec2-user 20 Feb 16 12:35 .aws
13241907 -rw-r--r-- 1 ec2-user ec2-user 10 Jul 15 2020 .bash_history
13241976 -rw-r--r-- 1 ec2-user ec2-user 18 Jul 15 2020 .bash_logout
13241977 -rw-r--r-- 1 ec2-user ec2-user 193 Jul 15 2020 .bash_profile
13241978 -rw-r--r-- 1 ec2-user ec2-user 231 Jul 15 2020 .bashrc
13242510 drwxr-xr-x 2 root root 64 Oct 11 2021 .cache
13242509 -rw-r--r-- 1 root root 105 Oct 11 2021 Dockerfile
13242508 -rw-r--r-- 1 ec2-user ec2-user 1595 Feb 16 12:27 Oc1.zip
394383 drwx----- 3 ec2-user root 25 Feb 16 12:31 .cache
13242510 drwxr-xr-x 2 root root 64 Oct 11 2021 content
13242508 -rw-r--r-- 1 ec2-user ec2-user 1007 Feb 16 12:27 Oc1.zip
13242508 -rw-r--r-- 1 ec2-user ec2-user 1595 Feb 16 12:27 Oc1.zip
394383 drwx----- 2 ec2-user ec2-user 29 Feb 16 12:13 .ssh
[ec2-user@ip-172-31-16-172 ~]$ sudo su
[ec2-user@ip-172-31-16-172 ~]$ vi Dockerfile
[ec2-user@ip-172-31-16-172 ~]$ docker build -t lambda_ecr .
Sending build context to Docker daemon 598.5kB
Step 1/4 : FROM python:alpine
--> 85da6554f50c
Step 2/4 : RUN pip install -r requirements.txt
--> Using cache
--> 110486ffff3ca
Step 3/4 : RUN pip install -r requirements.txt
--> Using cache
--> 876818be91f6
Step 4/4 : CMD python3 bootstrap.py
--> Using cache
--> 876818be91f6
successfully built 876818be91f6
successfully tagged lambda_ecr:latest
[ec2-user@ip-172-31-16-172 ec2-user]#

```

Step 2-(iii)

```
root@ip-172-31-16-172:home/ec2-user
[ec2-user@ip-172-31-16-172 ~]$ sudo docker images
REPOSITORY TAG IMAGE ID CREATED SIZE
lambda_ecr latest 876818be1f16 About a minute ago 152MB
python alpine 85da6554f50c 5 days ago 51.7MB

[ec2-user@ip-172-31-16-172 ~]$ 
[ec2-user@ip-172-31-16-172 ~]$ 
[ec2-user@ip-172-31-16-172 ~]$ 
[ec2-user@ip-172-31-16-172 ~]$ 
[ec2-user@ip-172-31-16-172 ~]$ ls -ali
total 24
13241973 drwx----- 6 ec2-user ec2-user 169 Feb 16 12:35 .
4194789 -rwxr--r-x 3 root root 22 Feb 16 12:13 ..
4542465 drwxrwxr-x 2 ec2-user ec2-user 20 Feb 16 12:35 .aws
13242507 -rwx----- 1 ec2-user ec2-user 18 Feb 16 12:25 .bash_history
13242508 -rwx----- 1 ec2-user ec2-user 18 Feb 16 12:25 .bash_logout
13241077 -rwxr--r-- 1 ec2-user ec2-user 193 Jul 15 2020 .bash_profile
13241978 -rwxr--r-- 1 ec2-user ec2-user 231 Jul 15 2020 .bashrc
394386 drwxrwxr-x 3 ec2-user root 25 Feb 16 12:31 .cache
13242509 -rwxr--r-- 1 ec2-user ec2-user 103 Feb 16 12:21 Dockerfile
13242508 -rwxr--r-- 1 ec2-user ec2-user 1595 Feb 16 12:27 OCI.zip
394383 drwxr----- 2 ec2-user ec2-user 29 Feb 16 12:13 .ssh
[ec2-user@ip-172-31-16-172 ~]$ touch Dockerfile
[ec2-user@ip-172-31-16-172 ~]$ sudo chmod 777 Dockerfile
[ec2-user@ip-172-31-16-172 ~]$ sudo touch Dockerfile
[ec2-user@ip-172-31-16-172 ~]$ ls -ali
total 24
13241973 drwx----- 6 ec2-user ec2-user 169 Feb 16 12:35 .
4194789 -rwxr--r-x 3 root root 22 Feb 16 12:13 ..
4542465 drwxrwxr-x 2 ec2-user ec2-user 20 Feb 16 12:35 .aws
13242507 -rwx----- 1 ec2-user ec2-user 10 Feb 16 12:25 .bash_history
13241976 -rwxr--r-- 1 ec2-user ec2-user 18 Jul 15 2020 .bash_logout
13242508 -rwx----- 1 ec2-user ec2-user 193 Jul 15 2020 .bash_profile
13241078 -rwxr--r-- 1 ec2-user ec2-user 193 Jul 15 2020 .bashrc
394386 drwxrwxr-x 3 ec2-user root 25 Feb 16 12:31 .cache
13242510 drwxrwxr-x 2 root root 105 Feb 16 12:44 Dockerfile
13242509 -rwxr--r-- 1 ec2-user ec2-user 1595 Feb 16 12:27 OCI.zip
394383 drwxr----- 2 ec2-user ec2-user 29 Feb 16 12:13 .ssh
[ec2-user@ip-172-31-16-172 ~]$ sudo su
[root@ip-172-31-16-172 ec2-user]# vi Dockerfile
[root@ip-172-31-16-172 ec2-user]# docker build -t lambda_ecr .
Sending build context to Docker daemon 598.5kB
Step 1/4 : FROM python:alpine
--> 85da6554f50c
Step 2/4 : COPY ./content .
--> Using cache
--> 9e9203c3315a
Step 3/4 : RUN pip install -r requirements.txt
--> Using cache
--> 9e9203c3315a
Step 4/4 : CMD python bootstrap.py
--> Using cache
--> 876818be1f16
Successfully built 876818be1f16
Successfully tagged lambda_ecr:latest
[root@ip-172-31-16-172 ec2-user]# docker tag lambda_ecr:latest 380732321839.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
[root@ip-172-31-16-172 ec2-user]#
```

Step 2-(iv)

```
root@ip-172-31-16-172:~/home/ec2-user
total 24
13241975 drwx----- 6 ec2-user ec2-user 169 Feb 16 12:35 .
4194789 drwxr-xr-x 3 root root 22 Feb 16 12:13 ..
4542465 drwxrwxr-x 2 ec2-user ec2-user 20 Feb 16 12:35 .aws
13242507 -rw-r--r-- 1 ec2-user ec2-user 10 Feb 16 12:25 .bash_history
13241976 -rw-r--r-- 1 ec2-user ec2-user 18 Jul 15 2020 .bash_logout
13241977 -rw-r--r-- 1 ec2-user ec2-user 193 Jul 15 2020 .bash_profile
13241978 -rw-r--r-- 1 ec2-user ec2-user 231 Jul 15 2020 .bashrc
13242510 -rw-r--r-- 1 root root 8 Feb 16 12:35 .cache
13242509 -rw-r--r-- 1 root root 105 Oct 11 2021 .content
13242508 -rw-r--r-- 1 ec2-user ec2-user 1595 Feb 16 12:27 .oci.zip
394381 drwx----- 3 ec2-user ec2-user 25 Feb 16 12:31 .dockercfg
[ec2-user@ip-172-31-16-172 ~]$ touch Dockerfile
touch: cannot touch 'Dockerfile': Permission denied
[ec2-user@ip-172-31-16-172 ~]$ sudo touch Dockerfile
[ec2-user@ip-172-31-16-172 ~]$ ls -al
total 24
13241975 drwx----- 6 ec2-user ec2-user 169 Feb 16 12:35 .
4194789 drwxr-xr-x 3 root root 22 Feb 16 12:13 ..
4542465 drwxrwxr-x 2 ec2-user ec2-user 20 Feb 16 12:35 .aws
13242507 -rw-r--r-- 1 ec2-user ec2-user 10 Feb 16 12:25 .bash_history
13241976 -rw-r--r-- 1 ec2-user ec2-user 183 Jul 15 2020 .bash_logout
13241977 -rw-r--r-- 1 ec2-user ec2-user 193 Jul 15 2020 .bash_profile
13241978 -rw-r--r-- 1 ec2-user ec2-user 231 Jul 15 2020 .bashrc
394386 drwx----- 3 ec2-user root 25 Feb 16 12:31 .cache
13242510 -rw-r--r-- 1 root root 64 Oct 11 2021 .content
13242509 -rw-r--r-- 1 root root 105 Oct 11 2021 .dockercfg
13242508 -rw-r--r-- 1 root root 1595 Feb 16 12:27 .oci.zip
394383 drwx----- 2 ec2-user ec2-user 29 Feb 16 12:13 .ssh
[ec2-user@ip-172-31-16-172 ~]$ sudo su
[root@ip-172-31-16-172 ~]# vi Dockerfile
[ec2-user@ip-172-31-16-172 ~]# docker build -t lambda_ecr .
Sending build context to Docker daemon 598.5kB
Step 1/4 : FROM python:alpine
--> 85da654f50c
Step 2/4 : COPY requirements.txt /content .
--> 11048ff7fc3ca
Step 3/4 : RUN pip install -r requirements.txt
--> Using cache
--> 87681be91f6
Step 4/4 : CMD python bootstrap.py
--> Using cache
--> 87681be91f6
Successfully built 87681be91f6
[ec2-user@ip-172-31-16-172 ~]# docker tag lambda_ecr:latest 380732321839.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
[ec2-user@ip-172-31-16-172 ~]# aws ecr get-login-password --region us-east-1
[ec2-user@ip-172-31-16-172 ~]# docker login --username AWS --password-stash 380732321839.dkr.ecr.us-east-1.amazonaws.com
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning. See
https://docs.docker.com/engine/reference/commandline/login/#credentials-store

Login Successed
[root@ip-172-31-16-172 ~]#
```

Step 2-(v)

```

root@ip-172-31-16-172:/home/ec2-user
394383 drwx--- 2 ec2-user ec2-user 29 Feb 16 12:13 .ssh
[ec2-user@ip-172-31-16-172 ~]# touch Dockerfile
touch: cannot touch 'Dockerfile': Permission denied
[ec2-user@ip-172-31-16-172 ~]# sudo touch Dockerfile
[ec2-user@ip-172-31-16-172 ~]# ls -al
total 598
13241972 drwx----- 6 ec2-user ec2-user 169 Feb 16 12:35 .
4194789 drwxr-xr-x 3 root root 22 Feb 16 12:13 ..
4542465 drwxrwxr-x 2 ec2-user ec2-user 20 Feb 16 12:35 .aws
13242507 -rw----- 1 ec2-user ec2-user 10 Feb 16 12:25 .bash_history
13241976 -rw-r--r-- 1 ec2-user ec2-user 18 Jul 15 2020 .bash_logout
13241977 -rw-r--r-- 1 ec2-user ec2-user 31 Jul 15 2020 .bashrc
13241978 -rw-r--r-- 1 ec2-user ec2-user 231 Jul 15 2020 .bashrc
394386 drwx---- 3 ec2-user root 25 Feb 16 12:31 .cache
13742510 drwxr-xr-x 2 root root 64 Oct 11 2021 content
13241969 -rw-r--r-- 1 root root 30 Feb 16 12:44 Dockerfile
13242508 -rw-r--r-- 1 ec2-user ec2-user 1599 Feb 16 12:25 EC1.zip
394383 drwx--- 2 ec2-user ec2-user 29 Feb 16 12:13 .ssh
[ec2-user@ip-172-31-16-172 ~]# sudo su
[root@ip-172-31-16-172 ec2-user]# vi Dockerfile
[root@ip-172-31-16-172 ec2-user]# docker build -t lambda_ecr .
Sending build context to Docker daemon 598.5kB
Step 1/4 : FROM python:alpine
--> 85da554f50c
Step 2/4 : COPY ./content .
--> 110486fff3ca
Step 3/4 : RUN pip install -r requirements.txt
--> Using cache
--> 110486fff3ca
Step 4/4 : CMD python3 bootstrap.py
--> Using cache
--> 876818be91f6
Successfully built 876818be91f6
Step 5/5 : TAG latest
--> 20162c03d83: Tagged
[root@ip-172-31-16-172 ec2-user]# docker tag lambda_ecr:latest 380732321839.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
[root@ip-172-31-16-172 ec2-user]# aws ecr get-login-password --region us-east-1
docker login --username AWS --password $(aws ecr get-login-password --region us-east-1)
Amazon ECR
WARNING! Your password will be stored unencrypted in /root/.docker/config.json.
Configure a credential helper to remove this warning.
See https://docs.docker.com/engine/reference/commandline/login/#credentials-store
Login Succeeded
[root@ip-172-31-16-172 ec2-user]# docker push 380732321839.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr:latest
The push refers to repository [380732321839.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr]
d8...: Pushed
20162c03d83: Pushed
e2cd8f647283: Pushed
8ab058a1a5ad: Pushed
8b058a1a5ad: Pushed
8c173a500bd5: Pushed
f7cd9720fcfe: Pushed
7cd52847ad77: Pushed
latest: digest: sha256:86fffb51079335c79c5864705fdd0693ada547d28a17f4cac4e1dc4bf957cd47 size: 1787
[root@ip-172-31-16-172 ec2-user]#

```

Step 2-(vi)

https://us-east-1.console.aws.amazon.com/ecr/repositories/private/380732321839/lambda_ecr?region=us-east-1

| Image tag | Artifact type | Pushed at | Size (MB) | Image URI | Digest | Scan status | Vulnerabilities |
|-----------|---------------|--------------------------------------|-----------|--------------------------|---|-------------|-----------------|
| latest | Image | February 16, 2023, 13:48:21 (UTC+01) | 47.93 | Copy URI | sha256:86fffb51079335c79c5864705fdd069... | - | - |

Step 2-(vii)

The screenshot shows the AWS ECR console with the URL https://us-east-1.console.aws.amazon.com/ecr/repositories/private/380732321839/lambda_ecr?region=us-east-1. The page displays the 'lambda_ecr' repository under the 'Images' section. It shows one image entry: 'latest' (Image type, size 47.93 MB, URI sha256:86ff51079335c79c5864705fdd0693ada547d28a17f4cac4ee1dc4bf957cd47). There are buttons for View push commands, Delete, Details, and Scan.

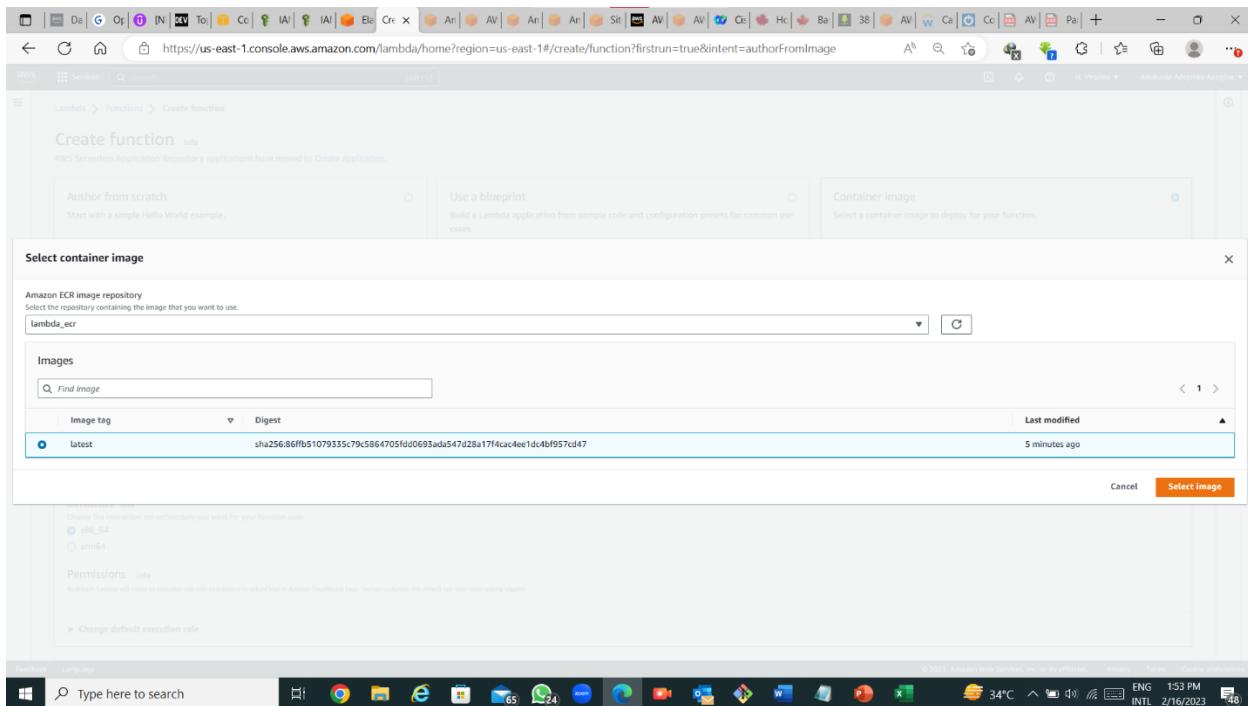
Step 2-(viii)

The screenshot shows the AWS ECR console with the URL https://us-east-1.console.aws.amazon.com/ecr/repositories/private/380732321839/lambda_ecr/_image/sha256:86ff51079335c79c5864705fdd0693ada547d28a17f4cac4ee1dc4bf957cd47. The page displays the details of the image 'sha256:86ff51079335c79c5864705fdd0693ada547d28a17f4cac4ee1dc4bf957cd47'. It includes sections for Details, General information (Artifact type: Image, Repository: lambda_ecr, Size: 47.93 MB), Basic scanning (Scan status: -, Vulnerabilities: -), and Replication status (Select status source: digest). The page is part of a Windows desktop environment.

Step 3: Creation of Lambda function to test the image

| | |
|----------------------|--|
| Step number | A |
| Step name | Create the Lambda function and test the image |
| Instructions | <ol style="list-style-type: none"> 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)



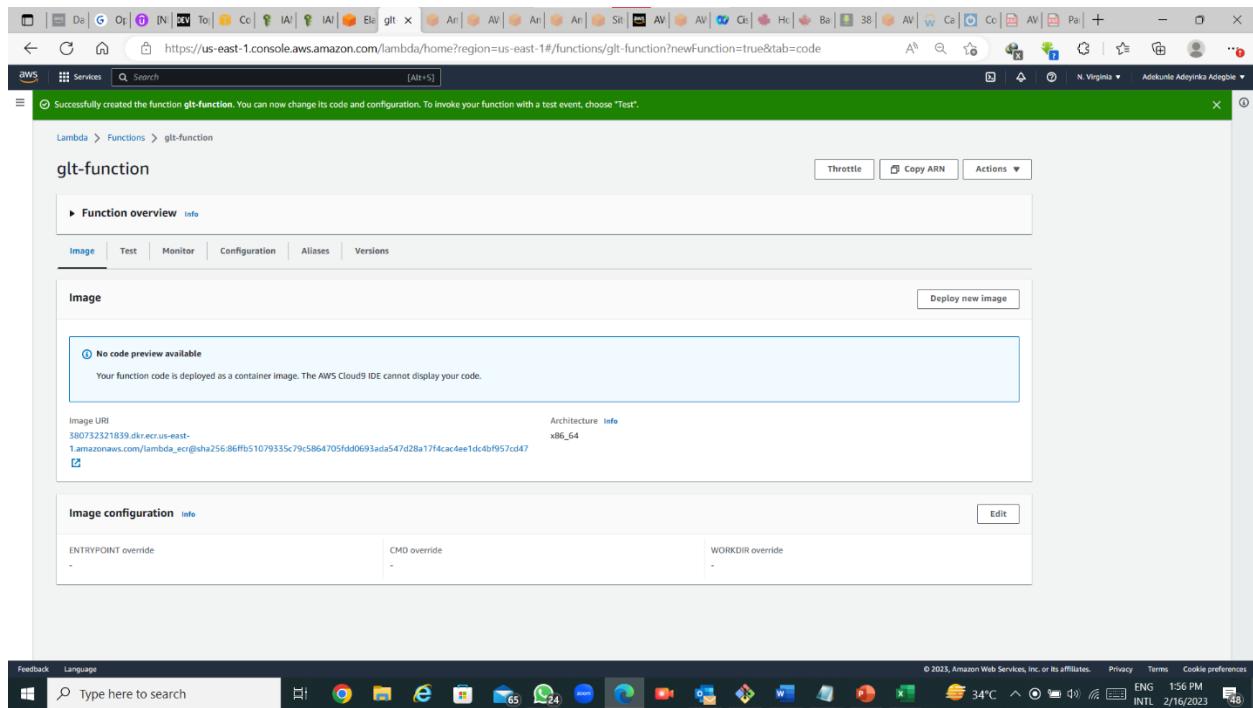
Step 3-(ii)

The screenshot shows the AWS Lambda 'Create function' wizard. The 'Container image' tab is selected. In the 'Basic information' section, the function name is 'git-function'. The 'Container image URI' field contains '38072321859.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr@sha256:86ffb51079335c79c5864705fd0693ada547d28a17f4cac4ee1dc4bf957cd47'. The 'Architecture' section shows 'x86_64' selected. The 'Permissions' section indicates that Lambda will create an execution role with CloudWatch Logs permission. The 'Change default execution role' section is collapsed.

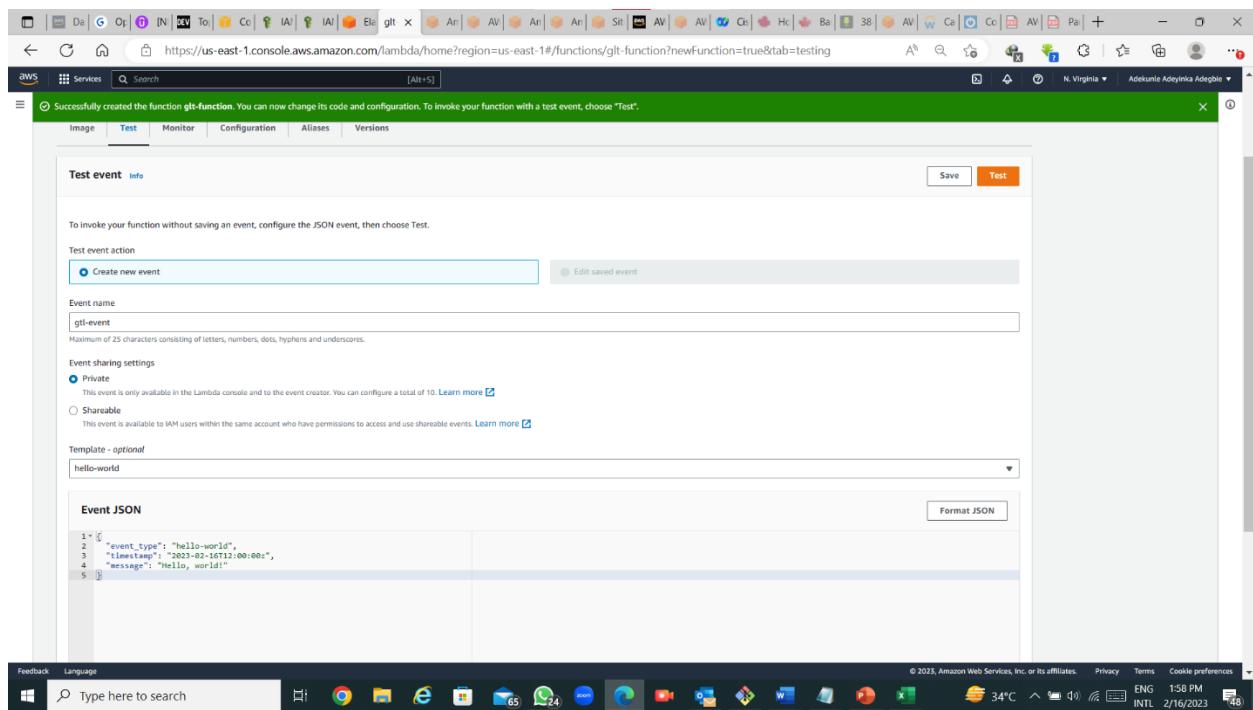
Step 3-(iii)

The screenshot shows the AWS Lambda 'Create function' wizard with the 'Change default execution role' section expanded. Under 'Execution role', 'Use an existing role' is selected, and 'labRole' is chosen from the dropdown. The 'Existing role' dropdown also lists 'labRole'. The 'Advanced settings' section is collapsed.

Step 3-(iv)



Step 3-(v)



Step 3-(vi)

The screenshot shows the AWS Lambda console with the URL <https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/glt-function?newFunction=true&tab=testing>. A modal window titled "Execution result: succeeded" is displayed, showing the log output of a test event. The log content is as follows:

```
{ "statusCode": 200, "body": "Hello from Lambda Containers", "event": { "event_type": "hello-world", "timestamp": "2023-02-16T12:00:00Z", "message": "Hello, world!" } }
```

The modal also displays summary statistics: Request ID 07cc6d65-9893-43c0-9027-431c8d628fe2, Duration 10.67 ms, Resources configured 128 MB, and Max memory used 40 MB. Below the modal, the Lambda function configuration page is visible.

Step 3-(vii)

The screenshot shows the AWS Lambda console with the URL <https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/glt-function?newFunction=true&tab=testing>. A modal window titled "Test event" is open, showing the configuration for a new test event named "glt-event". The "Event name" field contains "glt-event". Under "Event sharing settings", the "Private" option is selected. The "Template - optional" dropdown is set to "hello-world". The "Event JSON" field contains the following JSON:

```
1- [ 2- { 3- "event_type": "hello-world", 4- "timestamp": "2023-02-16T12:00:00Z", 5- "message": "Hello, world!" } ]
```

The modal has "Save" and "Test" buttons at the top right. The Lambda function configuration page is visible in the background.

Step 3-(viii)

The screenshot shows the AWS Lambda console interface. A modal window at the top indicates that the test event 'gtl-event' was successfully saved. Below it, the 'Test event' configuration screen is visible. It includes fields for 'Event name' (set to 'gtl-event'), an 'Event JSON' section containing a sample event object, and a 'Test' button. The status bar at the bottom shows the date and time as 2/16/2023 2:01 PM.

Step 3-(ix)

The screenshot shows the AWS Lambda console interface. The 'gtl-function' function is selected. The 'Image' tab is active, showing the function code is deployed as a container image. The 'Image URI' is listed as 580732321839.dkr.ecr.us-east-1.amazonaws.com/lambda_ecr@sha256:86ff51079335c79:5864705fd0693eda547d28a17f4cac4ee1dc4b957cd47. The 'Architecture' is listed as x86_64. The 'Image configuration' section shows the 'ENTRYPOINT override' field set to '-' and the 'CMD override' field also set to '-'. The status bar at the bottom shows the date and time as 2/16/2023 2:02 PM.

Step 3-(x)

The screenshot shows the AWS Lambda console interface. The URL in the address bar is <https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/glt-function?newFunction=true&tab=image>. The browser tab is titled "glt-function". The main content area displays the "Function overview" for the "glt-function" Lambda function. The function name is "glt-function", and it is described as a container image function. It was last modified 8 minutes ago. The ARN is `arn:aws:lambda:us-east-1:380732321859:function:glt-function`. The Function URL is listed as `amaz...f957cd47`. Below this, there are tabs for "Image", "Test", "Monitor", "Configuration", "Aliases", and "Versions". The "Image" tab is selected. A note indicates "No code preview available" because the function code is deployed as a container image. The "Architecture" section shows "x86_64". At the bottom of the page, there is a search bar, a taskbar with various icons, and a status bar showing the date and time.

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 |