AWS CERTIFIED SOLUTION ARCHITECT – ASSOCIATE LAB GUIDE

CONTENTS:

LAB5: Creating EBS volumes, Attaching, Increasing Volume size (Disk Upgrade), Detaching and Deleting Volumes

- **5a. Create an SSD EBS volume**
- **5b. Attach the created volume to an EC2 instance**
- **5c. Disk Upgrade: Increasing volume size**
- **5c. Detach the volume from the running instance and delete it**

LAB 6: Taking Snapshots, creating custom private AMIs and creating EBS volumes, AMIs from snapshots, and launching instances out of them

- **6a. Taking Snapshots from EBS volumes**
- **6b. Creating EBS volumes from Snapshots**
- **6c. Creating AMIs from Snapshots**
- **6d. Launching an instance from custom AMI and de-registering AMI**
- **6e. Deleting the created Snapshot**
- **6f. Creating custom AMIs from running instances and launching instances out of custom AMI**

LAB7: Creating IAM users, groups, roles and working with the policies

- **7a. Create IAM users with AWS managed policy**
- **7b. Create an IAM group with AWS managed policy**
- **7c. Adding Users to the group**
- **7d. Create a Role**
- **7e. Create a customer managed custom policy using Policy generator**
- **7f. Attach the policy created to a user**

LAB8: S3 bucket creation with public access, uploading objects and static web-hosting

- **8a. Create an S3 bucket with public access**
- **8b. Upload an object in the S3 bucket**
- **8c. S3 static web-hosting**

LAB 1: Working with an EC2 instance

1a. Creating an EC2 Linux instance

In this lab, you will be launching a Linux EC2 instance which will be later used as a Web Server

Step 1: Login to AWS Management console and select **EC2** from **Services**

Step 2: Select **Launch Instance** option in the **EC2 dashboard**

Step 3: Select an 'Amazon Linux AMI' in the **choose AMI** step and click **Next**

Step 4: Choose an **Instance Type** = **'t2.micro'** and click **Next**

Step 5: In Configure instance details page, Select the **default VPC** in **Network** field and an associated **default subnet** to launch and click **Next**

Step 6: Select the storage **size** of EBS Root volume in **Add Storage** section and click **Next**

Step 7: Enter a valid tag in **Add Tags** section and click **Next**

Step 8: Create a new Security Group with a valid name and description in **Configure Security Group** section and **add** the below **rules** and click **Review and launch**

Type: SSH; Port: 22; Source: Anywhere

Type: HTTP; Port: 80; Source: Anywhere

Step 9: Review the launch details and then select **Launch**

Step 10: Create a **New Key Pair**, **download** it and select **Launch instances**

Step 11: Navigate to EC2 console to **check** if the instance has been launched and is **running** successfully

1b. Starting, Stopping, Rebooting and Terminating the EC2 instance

Step 1: We can **Stop** the **running** instances by selecting the running instance and then clicking on **Actions -> Stop**

Step 2: We can **Reboot** the **running** instances by selecting the running instance and then clicking on **Actions -> Reboot**

Step 3: We can **Terminate** the **running** instances by selecting the running instance and then clicking on **Actions -> Terminate**

Step 4: Similarly, we can **Start** or **Terminate** the stopped instances by selecting the instance and then clicking on **Actions -> Start** or **Terminate**

LAB2: Logging to EC2 instance using CLI for Linux & putty for windows, Update yum, install apache httpd web service and host a sample web page

2a. Logging in to the Linux EC2 instance via Putty for Windows machines

Step 1: Download and install **putty tools** for Windows from https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html

Step 2: We will use **PuttyGen** to convert the **.pem** Private key to **.ppk** format (putty private key) to use it with **Putty -** ssh client for Windows

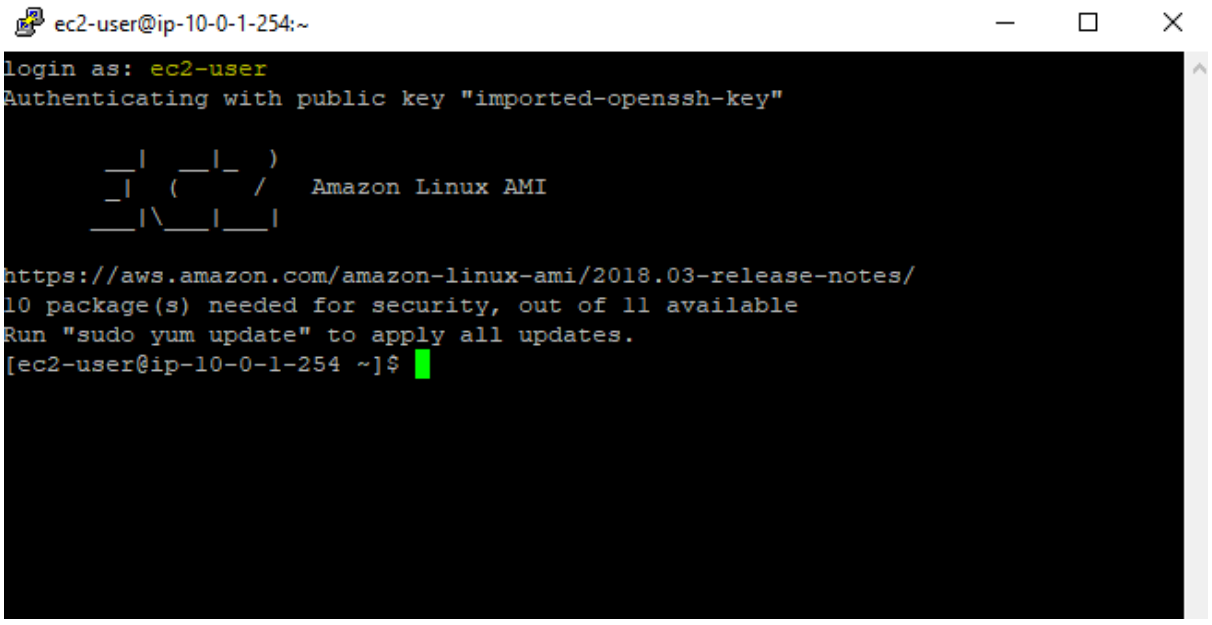
2a. Open PuttyGen and select **Load** option and load the **.pem key** and click **open**

2b. Once the pem key is loaded in PuttyGen , Select **save private key** option and click on **Yes** when prompted and save the .ppk file

Step 3: Open **Putty** app to ssh login in to the previously launched Linux server using Instance's Public IP in **Host Name** field and **Port: 22**

Step 4: Navigate and expand **SSH**, Click on **Auth** and **Browse** the .ppk file and click **Open**

Step 5: Select **Yes** when prompted and Enter Login as **ec2-user** in the CLI



2b. Logging in to the EC2 instance via terminal for Linux machine

For **Linux based machines**, Follow the steps from

https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/AccessingInstancesLinux.html

Step 1: In a command-line shell, change directories to the **location of the private key file** that you downloaded when you launched the instance

Step 2: Use the following command to **set the permissions** of your private key file so that only you can read it.

```
chmod 400 /path/my-key-pair.pem
```

If you do not set these permissions, then you cannot connect to your instance using this key pair. For more information, see Error: Unprotected Private Key File.

Step 3: Use the ssh command to connect to the instance. You specify the private key (.pem) file and *user_name@public_dns_name*. For example, if you used Amazon Linux 2 or the Amazon Linux AMI, the user name is ec2-user.

```
ssh -i /path/my-key-pair.pem ec2-user@ec2-198-51-100-1.compute-1.amazonaws.com
```

When are prompted for a response: Type **yes** and click **enter**

2c. Updating yum, installing apache httpd web server and hosting a sample web page

Step 1: Update the yum repository using "**sudo yum update –y**" command

```
[ec2-user@ip-10-0-1-254 ~]$ sudo yum update -y
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main
amzn-updates
Resolving Dependencies
--> Running transaction check
---> Package amazon-ssm-agent.x86_64 0:2.3.68.0-1.amzn1 will be updated
---> Package amazon-ssm-agent.x86_64 0:2.3.274.0-1.amzn1 will be an update
---> Package curl.x86_64 0:7.53.1-16.84.amzn1 will be updated
---> Package curl.x86_64 0:7.53.1-16.85.amzn1 will be an update
---> Package glibc.x86_64 0:2.17-222.173.amzn1 will be updated
---> Package glibc.x86_64 0:2.17-260.175.amzn1 will be an update
---> Package glibc-common.x86_64 0:2.17-222.173.amzn1 will be updated
---> Package glibc-common.x86_64 0:2.17-260.175.amzn1 will be an update
---> Package kernel.x86_64 0:4.14.88-72.73.amzn1 will be installed
---> Package kernel-tools.x86_64 0:4.14.77-70.59.amzn1 will be updated
---> Package kernel-tools.x86_64 0:4.14.88-72.73.amzn1 will be an update
---> Package libcurl.x86_64 0:7.53.1-16.84.amzn1 will be updated
```


Step 2: Install Apache httpd web service with "**sudo yum install httpd -y**" command

```
[ec2-user@ip-10-0-1-254 ~]$ sudo yum install httpd -y
Loaded plugins: priorities, update-motd, upgrade-helper
amzn-main                                                | 2.1 kB     00:00
amzn-updates                                             | 2.5 kB     00:00
Resolving Dependencies
--> Running transaction check
---> Package httpd.x86_64 0:2.2.34-1.16.amzn1 will be installed
--> Processing Dependency: httpd-tools = 2.2.34-1.16.amzn1 for package: httpd-2.
2.34-1.16.amzn1.x86_64
--> Processing Dependency: apr-util-ldap for package: httpd-2.2.34-1.16.amzn1.x8
6_64
```

Step 3: check the **status** and **start the httpd service** if it's in stopped state

"sudo service httpd status"

"sudo service httpd start"

```
[ec2-user@ip-10-0-1-254 ~]$ sudo service httpd status
httpd is stopped
[ec2-user@ip-10-0-1-254 ~]$ sudo service httpd start
Starting httpd: httpd: apr_sockaddr_info_get() failed for ip-10-0-1-254
httpd: Could not reliably determine the server's fully qualified domain name, us
ing 127.0.0.1 for ServerName
                                                           [  OK  ]
[ec2-user@ip-10-0-1-254 ~]$
```

Step 4: Navigate to **/var/www/html** directory and create an **index.html** file with below contents

```
[ec2-user@ip-10-0-1-254 ~]$ cd /var/www/html/
[ec2-user@ip-10-0-1-254 html]$ vi index.html
```

sudo su

cd /var/www/html

vi index.html

<head>

<title> favourites / bookmark title goes here </title>

</head>

<body bgcolor="white" text="blue">

<h1> My first page </h1>

This is the landing web page and type anything you want in here

</body>

</html>

Step 10: Press **Esc** key and type **:wq** and click enter to save changes to the file

Step 11: Restart httpd service **'sudo service httpd restart'**

```
[ec2-user@ip-10-0-1-254 html]$ sudo service httpd restart
Stopping httpd:                                            [  OK  ]
Starting httpd: httpd: apr_sockaddr_info_get() failed for ip-10-0-1-254
httpd: Could not reliably determine the server's fully qualified domain name, using 127.0.0.1 for ServerName
                                                           [  OK  ]
[ec2-user@ip-10-0-1-254 html]$
```

Step 12: Open a **web browser** and Enter the **Public IP** of the instance to view the Web Page

My first page

This is my first web page and I can type anything I want in here :)

LAB3: Creating VPC, Subnets, IGW, Route table and launching an instance in the Public Subnet

3a. Create a non-default custom VPC

Step 1: Login to AWS Management console and select **VPC** from **Services**

Step 2: Select **VPCs** in the VPC Dashboard and Click on **'Create VPC'**

Step 3: Give a **Name Tag** (for e.g. Test VPC) for your reference, Enter the **IPv4 CIDR block** details (for e.g. 10.0.0.0/16) and click on **Create**

Step 4: Once the VPC is created, we can view the same under **'Your VPCs'** section in VPC console

Creating a VPC creates a default DHCP options set, Main Route table and a default NACL

3b. Create two Subnets inside the VPC

Step 1: Navigate to **Subnets** section in VPC console

Step 2: To create a new subnet, click **'Create Subnet'**

Step 3: Enter the values for **Name tag** (for e.g. Subnet1)**,** Select the previously created **VPC**, enter a valid **IPv4 CIDR block** (for e.g. 10.0.1.0/24) and click on **Create**

Step 4: Similarly, create another Subnet with the following details:

Name tag (for e.g. Subnet2)**,** Select the previously created **VPC**, enter a valid **IPv4 CIDR block** (for e.g. 10.0.2.0/24) and click on **Create**

Step 5: We can view the created Subnets under **Subnets** section in VPC console

3c. Create an Internet Gateway and attach it to the VPC

Step 1: Navigate to **Internet Gateways** section in VPC console

Step 2: Create an Internet Gateway by clicking **'Create Internet gateway'**

Step 3: Enter a valid **Name tag** and click **Create**

Step 4: We can view the newly created IGW under **'Internet Gateways'** section as state **detached** in VPC console

Step 5: We will now attach the newly created IGW to the previously created VPC by selecting the IGW and select **Actions -> Attach to VPC**

Step 6: Select the previously created **VPC** and click **Attach**

Step 7: We can view the attached IGW under **'Internet Gateways'** section as state **attached** in VPC console

3d. Create a Public Route table with route to IGW

By default, Main Route table created with VPC is Private, so we will now create a Public Route table to be associated with Public Subnet

Step 1: Navigate to **'Route Tables'** section under VPC Console

Step 2: Select **'Create route table'**

Step 3: Enter a valid **Name tag** (for e.g. Public RT), select the **VPC** and click **Create**

Step 4: Once the Route table is created, we will now add a route to previously created IGW for Internet access by selecting the Route table and **Actions -> Edit routes**

Step 5: Add a public route to previously created IGW and click **'Save routes'**

Destination: 0.0.0.0 and **Target: IGW ID**

3e. Associate the Public RT with a Subnet to make it Public Subnet

Once the routes are updated, we will now associate the newly created Public RT with the Subnet to make it Public Subnet

Step 1: Navigate to **'Route Tables'** section under VPC Console

Step 2: Select the RT and click **Actions -> Edit subnet associations**

Step 3: Select the subnet which you want to make Public (for e.g. Subnet1) and click **Save**

We can now navigate to EC2 console and launch an EC2 instance (**Ref. Lab 1**) in **Newly created Custom VPC and subnet**.

Select the newly created VPC and Public subnet in **step 3**, so that you can access the instance from the internet.

LAB4: Allocating, Associating, Disassociating and Releasing Elastic IP

4a. Allocating an Elastic IP to the AWS Account

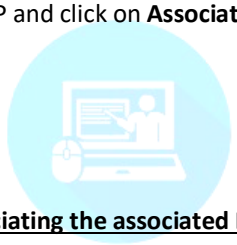
Step 1: Navigate to **Elastic IPs** section in the EC2 console and Click on '**Allocate new address'**

Step 2: Check **'Amazon pool'** and click on **Allocate**

4b. Associating the allocated Elastic IP with an EC2 instance

Step 1: Select the Allocated Elastic IP and click on **Actions -> Associate address**

Step 2: Check **Resource type** as **Instance** and select the instance id of the instance which you want to associate the Elastic IP and click on **Associate**



4c. Disassociating the associated Elastic IP from the EC2 instance

Step 1: To disassociate an Elastic IP, navigate to **Elastic IPs** section in the **EC2 console** and select the associated Elastic IP

Step 2: Click on **Actions -> Disassociate address -> Disassociate**

4d. Releasing the disassociated Elastic IP

Step 1: To release an Elastic IP, navigate to **Elastic IPs** section in the **EC2 console** and select the disassociated Elastic IP

Step 2: Select the Elastic IP and click on **Actions -> Release addresses -> Release**

LAB5: Creating EBS volumes, Attaching, Increasing Volume size (Disk Upgrade), Detaching and Deleting Volumes

We will now create a secondary EBS volume, attach it to a running instance, modify volume size and detach it from the same

5a. Create an SSD EBS volume

Step 1: Navigate to **Volumes** section in EC2 console and select **'Create volume'**

Step 2: Select the **Volume type** as **General Purpose SSD**, enter volume **Size**, Choose **Availability Zone** (it should be similar to the AZ of launched Instance) and Click on **'create volume'**

Step 3: Once the volume is created successfully, it will be in **available** state in **Volumes** tab

5b. Attach the created volume to an EC2 instance

Step 1: Select the available volume in the **Volumes** section of EC2 console

Step 2: Click on **Actions -> Attach volume**

Step 3: Select the **Instance** details, enter a device name in **Device** field (e.g. **/dev/sdb** for **b** drive) and click on **Attach**

Step 4: Once the volume has been attached successfully to an instance, we can see the volume state as **in use** in **Volumes** section of EC2 console

Step 5: To verify it from the backend, Login to the ec2 instance and run 'lsblk' command to view the attached volume

```
[ec2-user@ip-10-0-1-254 ~]$ lsblk
NAME    MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
xvda    202:0    0   8G  0 disk
└─xvda1 202:1    0   8G  0 part /
xvdb    202:16   0   5G  0 disk
```

5c. Disk Upgrade: Increasing volume size

AWS lets you upgrade the attached volume on the fly without detaching it or rebooting the EC2 instance

Step 1: Select the EBS volume/disk to be upgraded in the **Volumes** section of the EC2 console

Step 2: Click on **Actions -> Modify Volume**

Step 3: Enter the desired **size** and click on **Modify**

Note: EBS volumes once upgraded/increased cannot be downgraded/decreased

5d. Detach the volume from the running instance and delete it

Step 1: To detach an attached volume, select the volume in **Volumes** section of EC2 console

Step 2: Click on **Actions -> Detach volume**

Step 3: To delete the detached volume, select the detached volume in available state and click on **Actions -> Delete Volume** and confirm deletion

LAB 6: Taking Snapshots, creating custom private AMIs and creating EBS volumes, AMIs from snapshots, and launching instances out of them

6a. Taking Snapshots from EBS volumes

We will now create Snapshots from EBS volumes which acts as a backup of the disk and create EBS volumes from the same Snapshots

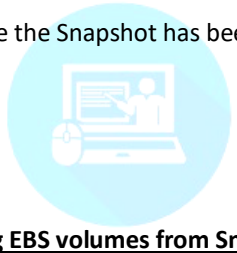
Step 1: Navigate to **Volumes** section in EC2 console and select the root volume of the EC2 instance

Step 2: Click on **Actions -> Create Snapshot**

Step 3: Fill in the **Description** for reference and click on **Create Snapshot**

Step 4: Navigate to **Snapshots** section in EC2 console to verify the snapshot creation

Step 5: Once the Snapshot has been created successfully, you can see the status as **completed**



6b. Creating EBS volumes from Snapshots

Step 1: To create a volume from the snapshot, navigate to the **Snapshots** section in EC2 console and select the snapshot

Step 2: Click on **Actions -> Create volume**

Step 3: Modify the volume size if needed (increase only) and choose the Availability zone details and click on **'Create volume'**

6c. Creating AMIs from Snapshots

Step 1: Similarly, to create an **AMI** from the snapshot, navigate to **Snapshots** section in EC2 console and select the snapshot of **root** volume

Note: AMIs can be created only from snapshot of root volumes as it needs root device mappings

Step 2: select the root volume's snapshot and click on **Actions -> Create Image**

Step 3: Give a **Name, Description** and click on **Create**

Step 4: Navigate to **AMIs** section in EC2 console

Step 5: We can view the newly created custom private AMI under **available** status

6d. Launching an instance from custom AMI and de-registering AMI

Step 1: Navigate to **AMIs** section in EC2 console

Step 2: We can launch a new instance from the AMI created by selecting the AMI and clicking **Actions -> Launch** and follow the instructions from **Lab 1**

Step 3: To delete or Deregister an AMI, select the AMI and click on **Actions -> Deregister**

Step 4: Deregistering the AMI wouldn't delete the associated Snapshot, make sure you delete the Snapshot which was created with the AMI **manually (Ref 5e)**

6e. Deleting the created Snapshot

Step 1: Navigate to **Snapshots** section in EC2 console

Step 2: Select the snapshot to be deleted and click on **Actions -> Delete**

6f. Creating custom AMIs from running instances and launching instances out of custom AMI

Step 1: Navigate to **instances** section in EC2 console

Step 2: Select the running instance and click on **Actions -> Image -> Create Image**

Step 3: Enter a **Name, Description** and click on **'Create Image'**

Step 4: Navigate to **AMIs** section in EC2 console

Step 5: We can view the newly created custom private AMI under **available** status

Step 6: We can launch a new instance from the AMI created by selecting the AMI and clicking **Actions -> Launch** and follow the instructions from **Lab 1**

Step 7: To delete or Deregister an AMI, select the AMI and click on **Actions -> Deregister**

Step 8: Deregistering the AMI wouldn't delete the associated Snapshot, make sure you delete the Snapshot which was created with the AMI **manually (Ref 5e)**

LAB7: Creating IAM users, groups, roles and working with the policies

7a. Create IAM users with AWS managed policy

Step 1: Navigate to **IAM** in **Resources** tab of AWS console

Step 2: To create users, navigate to **Users** in IAM console

Step 3: Select **Add user**

We will create two users 'User1' and 'User2' here for AWS console access

Step 4: Enter 2 usernames for 2 users in **User name** field and check **Access type** as AWS Management Console access and click **Next: Permissions**

We will now attach AWS managed EC2 Read Only Policy to the users

Step 5: Choose "**Attach existing policies directly**" and search for ''**AmazonEC2ReadOnlyAccess"** policy, check and click **Next: Tags**

Step 6: Add tags if needed and click **Next**

Step 7: Review the details and then Click **"Create Users"**

Step 8: Once the users are created, we can share the login details to respective users via email or login through the newly created users to verify if they have granted EC2 read permissions

7b. Create an IAM group with AWS managed policy

We will now create a group and attach a Group policy

Step 1: To create IAM Group, navigate to **Groups** in IAM console

Step 2: Click on **'Create New Group'**

Step 3: Enter a group name (e.g. Developers) and click **Next step**

Step 4: We will now Attach **AmazonEC2fullAccess** policy to the group and click **Next step -> Create Group**

7c. Adding Users to the group

Once the group is created, we will now add previously created users to the group.

Step 1: Navigate to **Groups** in IAM console and select the previously created group

Step 2: Select **Add Users to Group** in Users tab

Step 3: Select the users created previously and Click '**Add users'**

This will add the users to the group and the users will now have the group permissions

7d. Create a Role

Similar to Users, we can now create Service role, which can be used by AWS resources to integrate/access with other AWS resources

Step 1: Navigate to **Roles** in IAM console

Step 2: Click on **'Create role'**

We will now Create a service role for EC2 to access S3

Step 3: Select **AWS service** and then **Choose the service that will use this rule** as **EC2** and click **Next: Permissions**

Step 4: We will choose **'AmazonS3FullAccess**' policy and click **Next**

Step 5: Add tags if needed and then click **Review**

Step 6: Enter a **Role name** and Description for reference, then click **Create role**

This will create a new EC2 service role which can be selected while launching the EC2 instances to provide full access to S3 without having to manually configure with User credentials.

7e. Create a customer managed custom policy using Policy generator

We will now create a Customer managed custom policy using Policy generator

Step 1: Navigate to **Policies** in IAM console

Step 2: Click on **Create Policy**

Step 3: Switch to **Deny Permissions** and select

Service -> EC2

Actions -> Write -> TerminateInstances

Resources -> All resources and click **Review policy**

Step 4: Enter a valid **Name** and click **Create policy**

We have now created a Custom policy to deny EC2 termination, we will now attach this policy to a user

7f. Attach the policy created to a user

Step 1: Navigate to **Users** in IAM console

Step 2: Select a user (e.g. User1) and click on **Add Permissions**

Step 3: Choose the previously created customer managed policy by selecting **Attach existing policies directly** option and click **Next: Review -> Add permissions**

If you switch to User1 and try to terminate an instance, it will be denied even though User1 has a EC2 full access group permission attached, due to explicitly assigned deny customer managed policy. Explicit deny will always take precedence over allow rules.

LAB8: S3 bucket creation with public access, uploading objects and static web-hosting

8a. Create an S3 bucket with public access

Step 1: Navigate to **S3** in **Resources tab** of AWS console

Step 2: lick on **Create bucket**

Step 3: Enter a Globally Unique **Bucket name,** select **Region** and click **Next**

Step 4: Enter a valid tag and click **Next**

Step 5: Make the Bucket public by **Unchecking** all the **permissions** and click **Next -> Create Bucket**

8b. Upload an object in the S3 bucket

Once the bucket has been created successfully, we will now upload an image object into the Bucket

Step 1: Select the newly created bucket in S3 console

Step 2: Click **Upload** and browse an image (for e.g. earth.jpg)

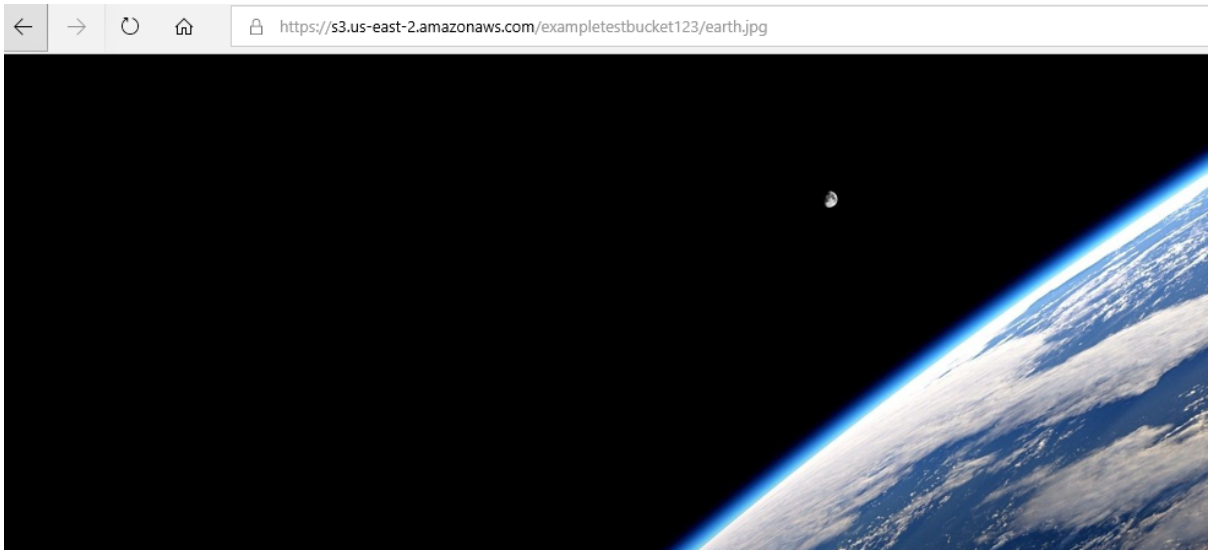
Step 3: After selecting the file and click **Next**

Step 4: We will now provide Public read access to this object by selecting **'Grant public read access to this object(s)'** in **Manage public permissions** and click **Next**

Step 5: Choose the **standard** storage class and click **Next -> Upload**

Step 6: Once the image object has been uploaded successfully, we can view the object by selecting the object and obtaining the **Object URL** from **object overview** tab.

Open the URL in the web browser to access the object:



8c. S3 static web-hosting

We will now take a step ahead and Enable Static web hosting for this bucket.

Step 1: Select the bucket and navigate to the Bucket **properties** and click on **'Static website hosting'**

Step 2: Select '**Use this bucket to host a website'** and type **'index.html'** in **Index document** and click **Save**. Make sure you copy the '**Endpoint'** before saving.

Step 3: We will now create an **index.html** file with below details and Upload it to the same bucket with public access

```
<!DOCTYPE html>
<html>
<head>
<title>Example S3 </title>
</head>
<body>
<h1> S3 WebHosting </h1>
<img src="earth.jpg" alt="Earth">
</body>
</html>
</html>
```

Here <img src = > Value could be an Image URL or an image name in the same bucket uploaded earlier.

Step 4: Once we have uploaded the index.html file successfully, we can now access the static website from the Bucket **Endpoint** copied earlier in any web browser:

S3 WebHosting