IT3061 – Massive Data Processing and Cloud Computing
Year 3, Semester 2
Practical Sheet 5

Cloud storage

- This practical focus on different storage services in AWS and Azure. They are,
  AWS - Elastic Block Storage (EBS), Elastic File System (EFS) and S3
  Azure – Azure Disks, Azure Files, Azure Blobs

1. Creating an Amazon EBS volume

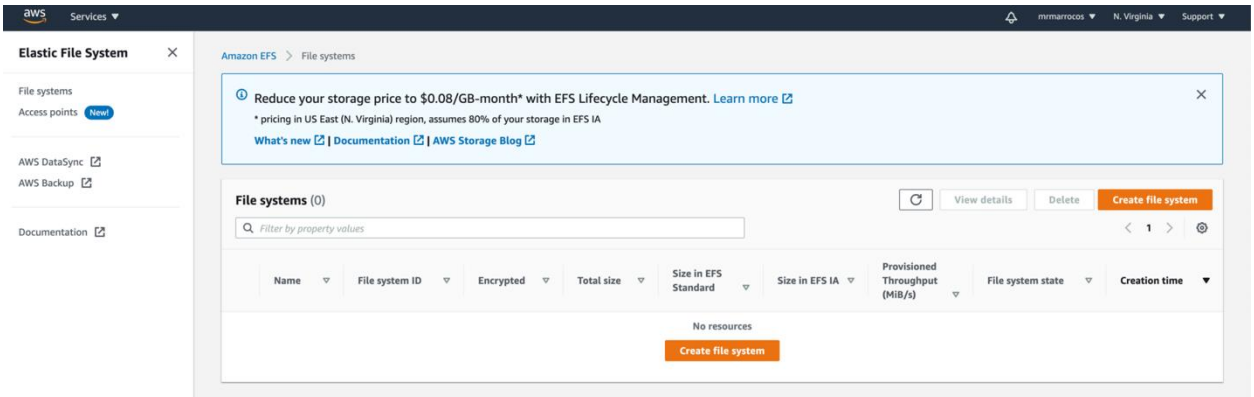
To create an empty EBS volume using the console,

1. Open the Amazon EC2 console.
2. From the navigation bar, select the Region in which you would like to create your volume. This choice is important because some Amazon EC2 resources can be shared between Regions, while others can't.
3. In the navigation pane, choose ELASTIC BLOCK STORE, Volumes.
4. Choose Create Volume.
5. For Volume Type, choose a volume type.
6. For Size, enter the size of the volume, in GiB.
7. For IOPS, enter the maximum number of input/output operations per second (IOPS) that the volume should provide. You can specify IOPS only for gp3, io1, and io2 volumes.
8. For Throughput, enter the throughput that the volume should provide, in MiB/s. You can specify throughput only for gp3 volumes.
9. For Availability Zone, choose the Availability Zone in which to create the volume. An EBS volume must be attached to an EC2 instance that is in the same Availability Zone as the volume.
10. (Optional) If the instance type supports EBS encryption and you want to encrypt the volume, select Encrypt this volume and choose a CMK. If encryption by default is enabled in this Region, EBS encryption is enabled and the default CMK for EBS encryption is chosen. You can choose a different CMK from Master Key or paste the full ARN of any key that you can access.
11. (Optional) Choose Create additional tags to add tags to the volume. For each tag, provide a tag key and a tag value.
12. Choose Create Volume. The volume is ready for use when the volume status is Available.

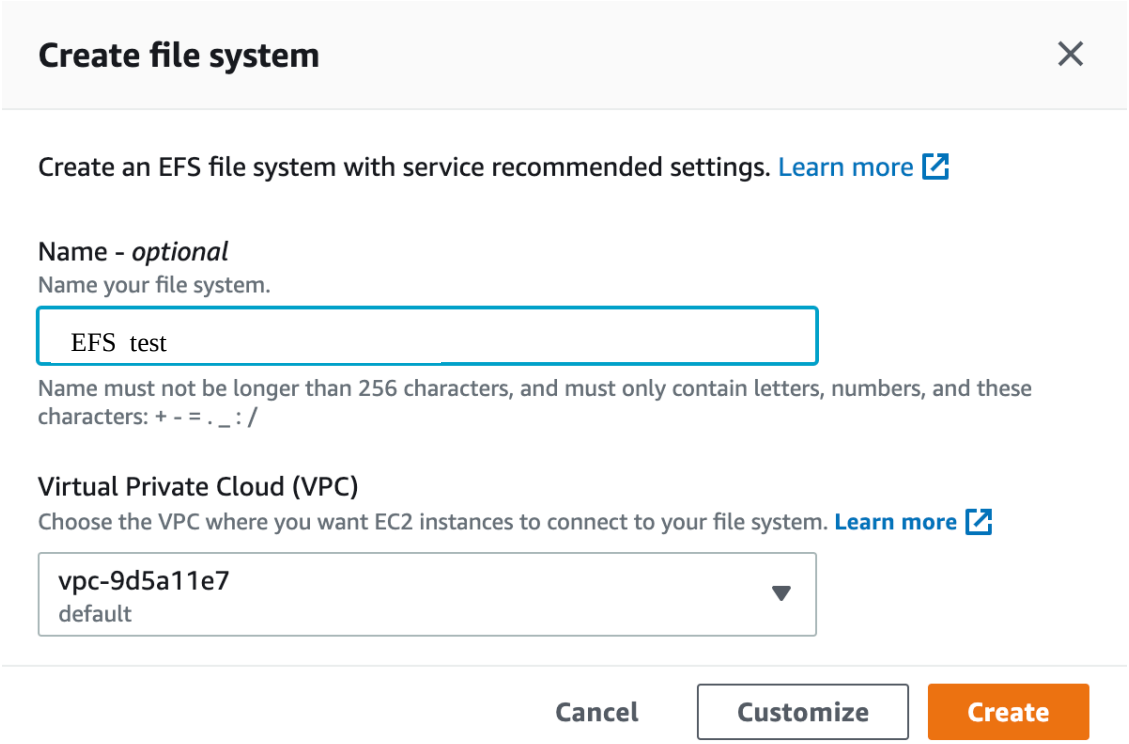
13. To use your new volume, attach it to an instance, format it, and mount it.

2. Creating an Amazon EFS

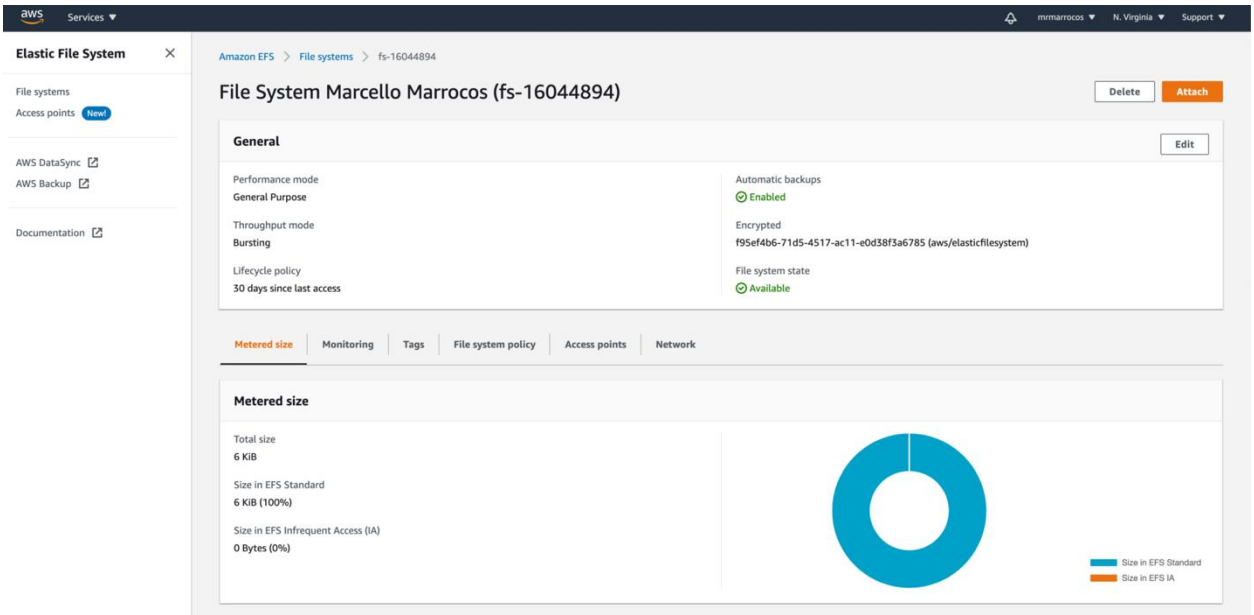
To create an EFS, you need to access the menu Services -> Storage -> EFS -> File System.



Click on the button "Create file system" to open the dialog. The field "name" is optional but required to select the VPC (Virtual Private Cloud) where this file system will be available.



After hitting the "Create" button, your file system will be available within a few seconds.



When clicking on your file system ID or file system name, it takes you to the details page of your EFS.

2.1 Attaching an EFS to an EC2 Instance

Now the EFS is created and need to attach it to an EC2 instance and start using it.

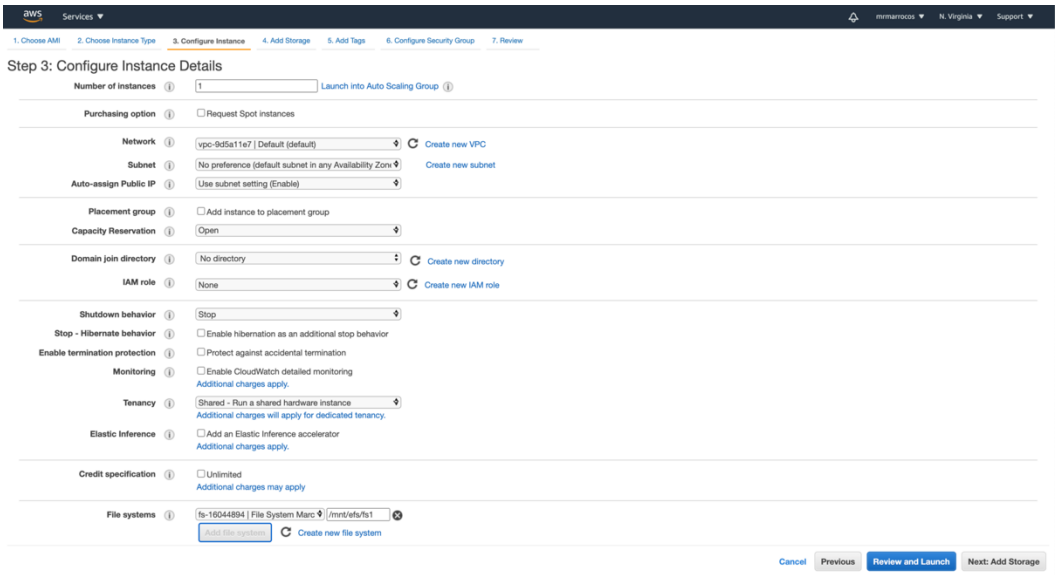
You have two ways to mount the EFS:

- At the moment of launching a new instance.
- On a running instance, using bash commands with the help of the amazon-efs-utils library.

At the launch of a new instance

The easiest way is to configure your EFS when launching a new EC2 instance.

In the "File systems" section, you can click on the "Add file system" button and select the EFS that you previously created.



Once you login into your EC2 instance, go to the path where you mounted, in this case, /mnt/efs/fs1, and type "pwd" to see the mounted drive:

On a running instance

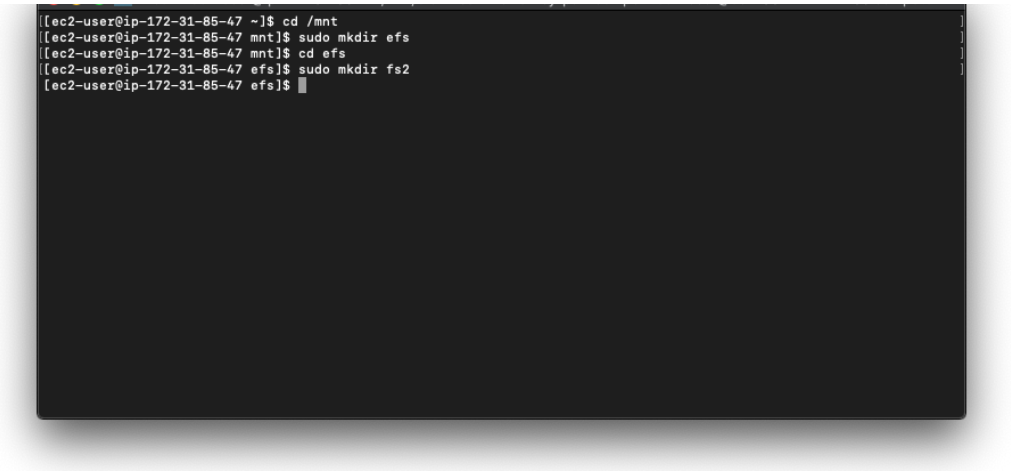
It is easier to mount your EFS launching a new instance, however, you can mount a file system at any time. For that, have an EC2 instance up and running and connect to it via SSH.

Connected to the instance, you need to create a directory where you will mount the EFS. For instance, create the folder structure mnt/efs/fs2 with the following commands:

```
sudo mkdir efs
```

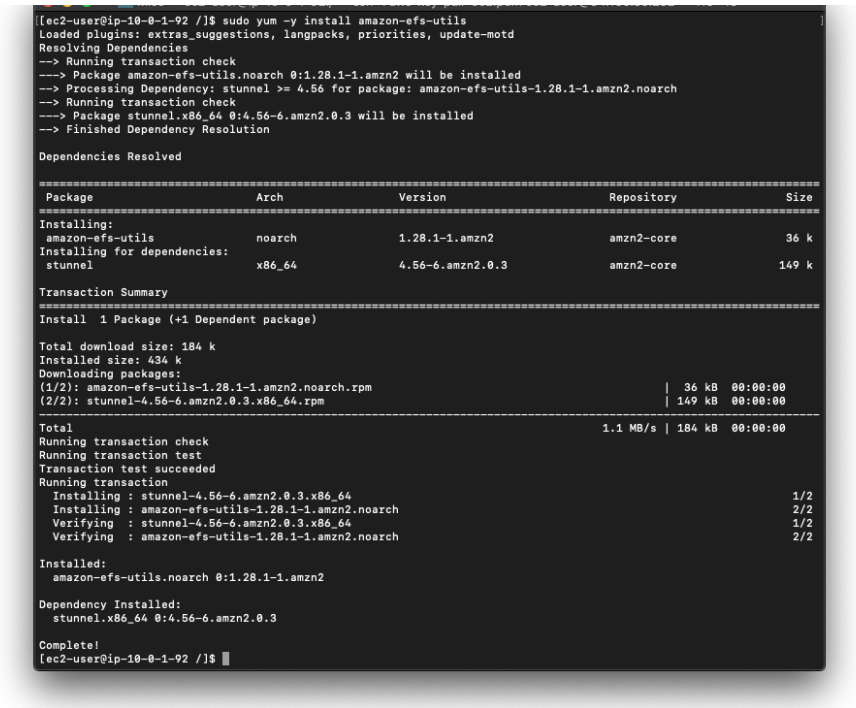
```
cd efs
```

```
sudo mkdir fs2
```



Now, you need to install the amazon efs utils library, which will allow us to run the connection command and mount the EFS. To move on with the installation of this library, run the following command:

```
sudo yum install -y amazon-efs-utils
```

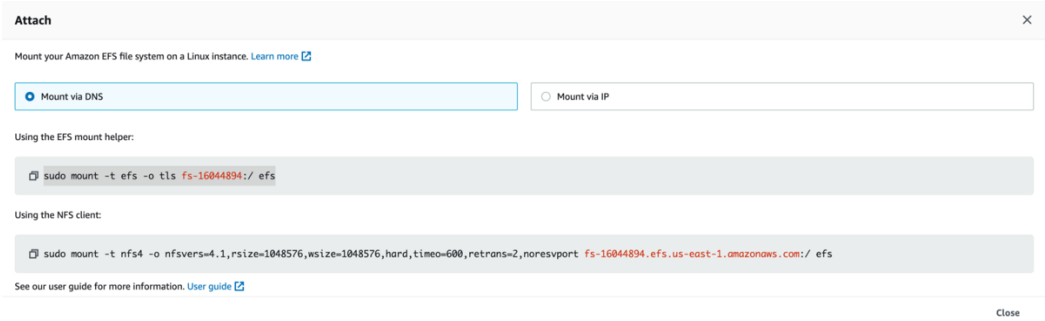


```
[ec2-user@ip-10-0-1-92 /]$ sudo yum -y install amazon-efs-utils
Loaded plugins: extras_suggestions, langpacks, priorities, update-motd
Resolving Dependencies
--> Running transaction check
---> Package amazon-efs-utils.noarch 0:1.28.1-1.amzn2 will be installed
--> Processing Dependency: stunnel >= 4.56 for package: amazon-efs-utils-1.28.1-1.amzn2.noarch
--> Running transaction check
---> Package stunnel.x86_64 0:4.56-6.amzn2.0.3 will be installed
--> Finished Dependency Resolution

Dependencies Resolved

================================================================================================================
 Package                       Arch                Version                       Repository               Size
================================================================================================================
Installing:
 amazon-efs-utils              noarch              1.28.1-1.amzn2                amzn2-core                36 k
Installing for dependencies:
 stunnel                       x86_64              4.56-6.amzn2.0.3              amzn2-core               149 k

Transaction Summary
================================================================================================================
Install  1 Package (+1 Dependent package)

Total download size: 184 k
Installed size: 434 k
Downloading packages:
(1/2): amazon-efs-utils-1.28.1-1.amzn2.noarch.rpm                                         |  36 kB  00:00:00
(2/2): stunnel-4.56-6.amzn2.0.3.x86_64.rpm                                                | 149 kB  00:00:00
----------------------------------------------------------------------------------------------------------------
Total                                                                                 1.1 MB/s | 184 kB  00:00:00
Running transaction check
Running transaction test
Transaction test succeeded
Running transaction
  Installing : stunnel-4.56-6.amzn2.0.3.x86_64                                                             1/2
  Installing : amazon-efs-utils-1.28.1-1.amzn2.noarch                                                      2/2
  Verifying  : stunnel-4.56-6.amzn2.0.3.x86_64                                                             1/2
  Verifying  : amazon-efs-utils-1.28.1-1.amzn2.noarch                                                      2/2

Installed:
  amazon-efs-utils.noarch 0:1.28.1-1.amzn2

Dependency Installed:
  stunnel.x86_64 0:4.56-6.amzn2.0.3

Complete!
[ec2-user@ip-10-0-1-92 /]$
```

After a successful installation, it is time to get the information needed to build the connection. The id of the EFS is required and part of the command to mount the unit.

Back to AWS console, access the file system that you created, and click on the button "Attach." Note that this can be a little misleading because the action will not attach the EFS to your instance, but provide you the full command to connect.

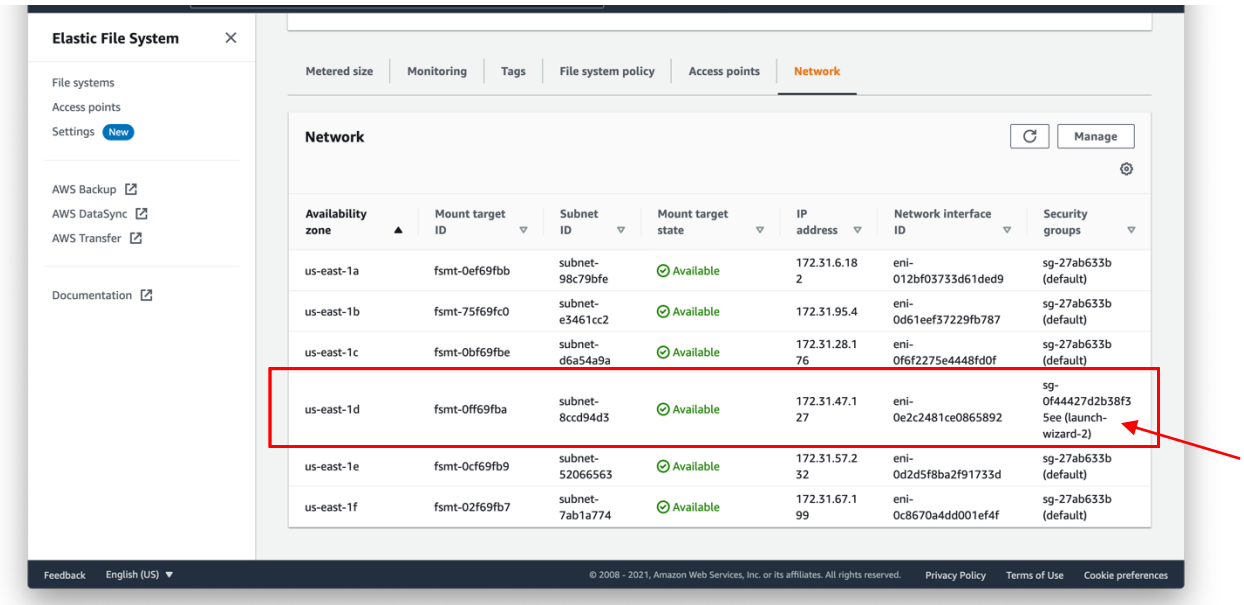
Unfortunately, there is no automatic way to click and mount it, so copy the command from "using the EFS mount helper" field.

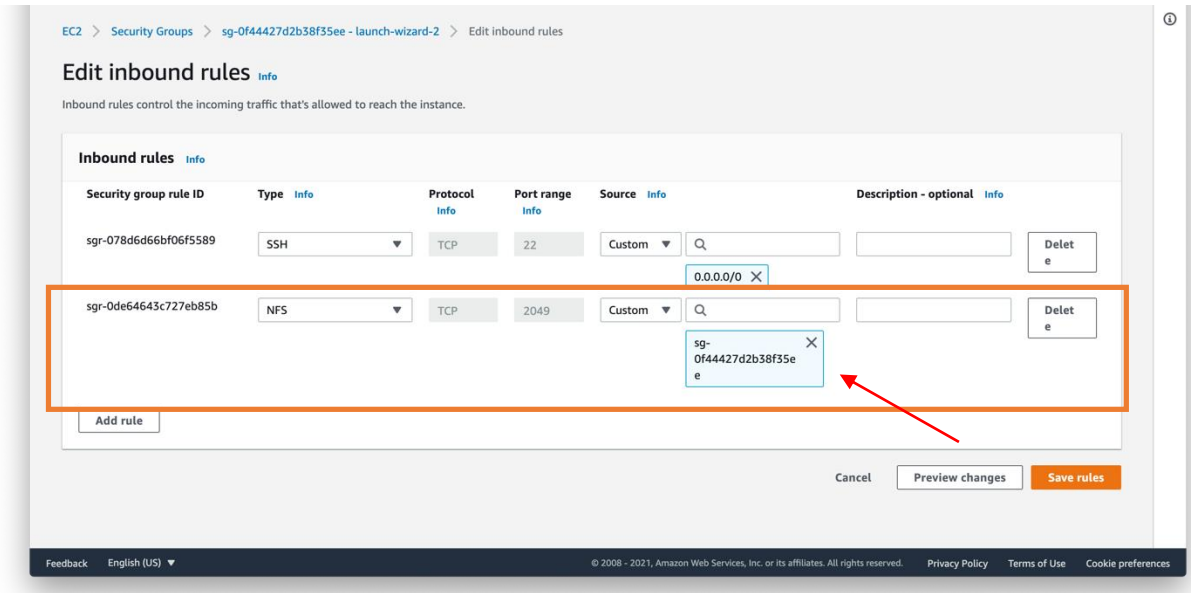


Then, back to the EC2 instance and execute the command, which is similar to the following — note that the last parameter is the path that you created to mount the EFS:

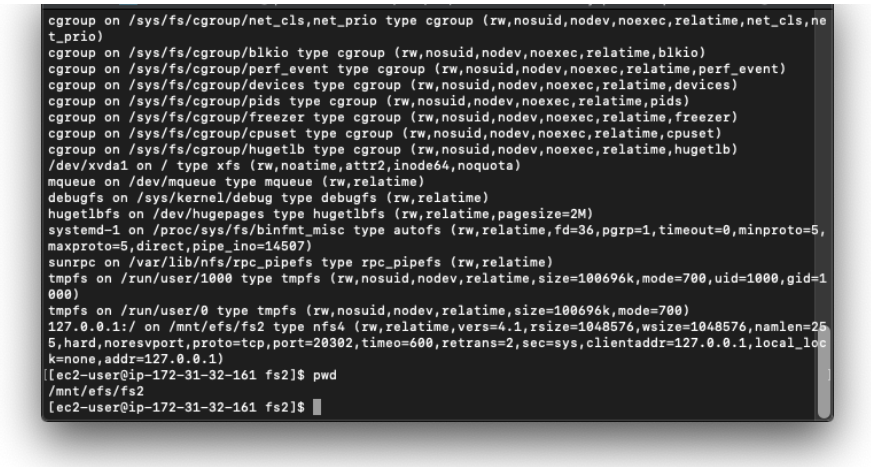
```
sudo mount -t efs -o tls fs-c12341234:/ /mnt/efs/fs2
```

Note – Add the following inbound rule to the security group. Make sure to set the correct security group for the availability zone.





After successfully mounting the EFS, no message will be displayed. But you can validate with the command "mount":



```
cgroup on /sys/fs/cgroup/net_cls,net_prio type cgroup (rw,nosuid,nodev,noexec,relatime,net_cls,ne
t_prio)
cgroup on /sys/fs/cgroup/blkio type cgroup (rw,nosuid,nodev,noexec,relatime,blkio)
cgroup on /sys/fs/cgroup/perf_event type cgroup (rw,nosuid,nodev,noexec,relatime,perf_event)
cgroup on /sys/fs/cgroup/devices type cgroup (rw,nosuid,nodev,noexec,relatime,devices)
cgroup on /sys/fs/cgroup/pids type cgroup (rw,nosuid,nodev,noexec,relatime,pids)
cgroup on /sys/fs/cgroup/freezer type cgroup (rw,nosuid,nodev,noexec,relatime,freezer)
cgroup on /sys/fs/cgroup/cpuset type cgroup (rw,nosuid,nodev,noexec,relatime,cpuset)
cgroup on /sys/fs/cgroup/hugetlb type cgroup (rw,nosuid,nodev,noexec,relatime,hugetlb)
/dev/xvda1 on / type xfs (rw,noatime,attr2,inode64,noquota)
mqueue on /dev/mqueue type mqueue (rw,relatime)
debugfs on /sys/kernel/debug type debugfs (rw,relatime)
hugetlbfs on /dev/hugepages type hugetlbfs (rw,relatime,pagesize=2M)
systemd-1 on /proc/sys/fs/binfmt_misc type autofs (rw,relatime,fd=36,pgrp=1,timeout=0,minproto=5,
maxproto=5,direct,pipe_ino=14507)
sunrpc on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
tmpfs on /run/user/1000 type tmpfs (rw,nosuid,nodev,relatime,size=100696k,mode=700,uid=1000,gid=1
000)
tmpfs on /run/user/0 type tmpfs (rw,nosuid,nodev,relatime,size=100696k,mode=700)
127.0.0.1:/ on /mnt/efs/fs2 type nfs4 (rw,relatime,vers=4.1,rsize=1048576,wsize=1048576,namlen=25
5,hard,noresvport,proto=tcp,port=20302,timeo=600,retrans=2,sec=sys,clientaddr=127.0.0.1,local_loc
k=none,addr=127.0.0.1)
[[ec2-user@ip-172-31-32-161 fs2]$ pwd
/mnt/efs/fs2
[ec2-user@ip-172-31-32-161 fs2]$
```

The line "127.0.0.1>/ on /mnt/efs/fs2…" represents the EFS mounted on this EC2 Instance.

Deleting the EFS

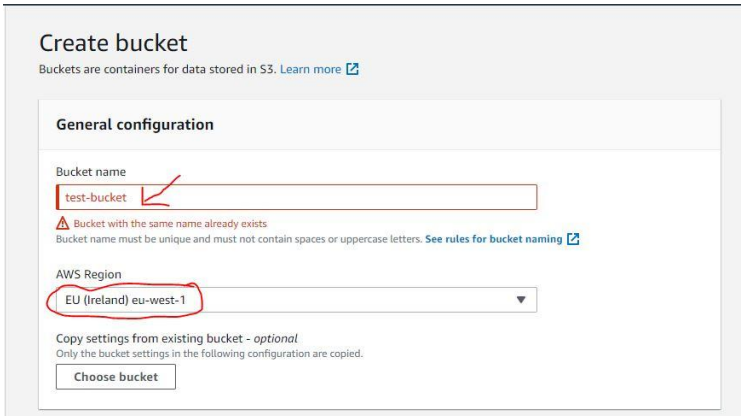
To delete the EFS, you can the File System details page and click on the "Delete" button.

Note that even if you have the EFS mounted into EC2 instances, you will be able to delete the EFS with no specific warning. So be careful when taking this action.

3. Creating an Amazon S3 bucket

To create s3 bucket, you need to access the menu Services -> Storage -> S3.
Once you click on S3 in above step, it will lead you to S3 dashboard. Here you can see the list of all your bucket across regions and create new bucket as well.

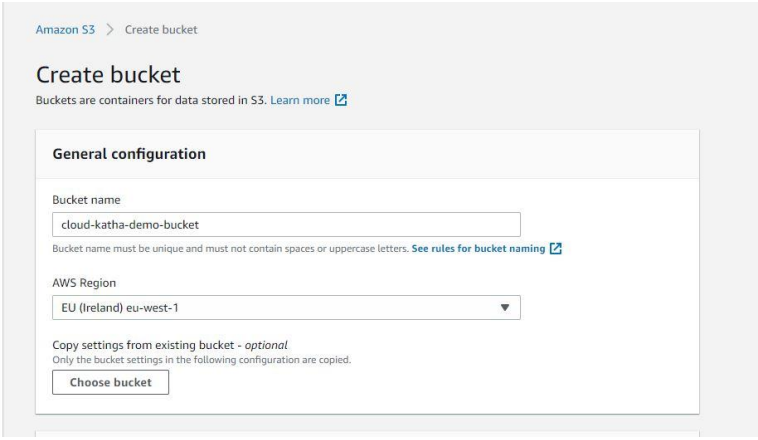
Click on **Create bucket.**



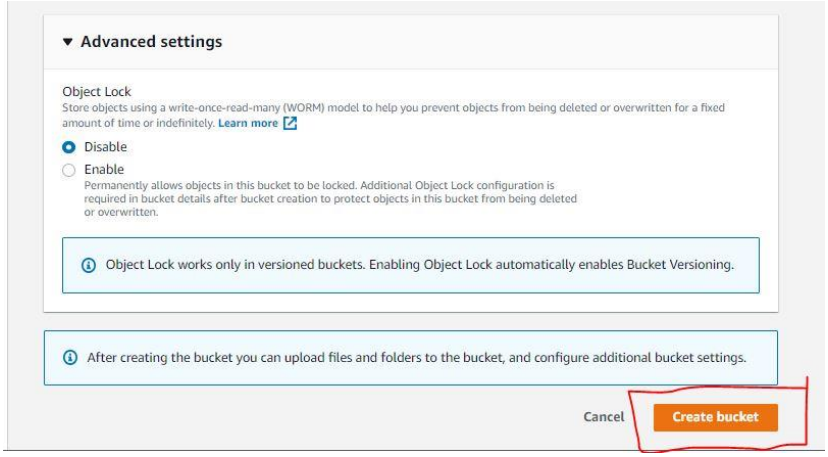
Provide a **unique** name to your bucket or else you will get "***Bucket with same name already exists***" error as above.

Reason is, **S3 bucket names are globally unique**. One you create a bucket with name "xyz" no one else in the world can create a bucket with the same name even in any other region or account until the bucket is deleted.

After the name, choose a region in which you would like your bucket to be created. You can choose any region of your choice, preferably near to your customer location to have optimum latency.



After putting the name and region, leave all other details to default and scroll down the page to find create Bucket button. Click on ***Create bucket***.

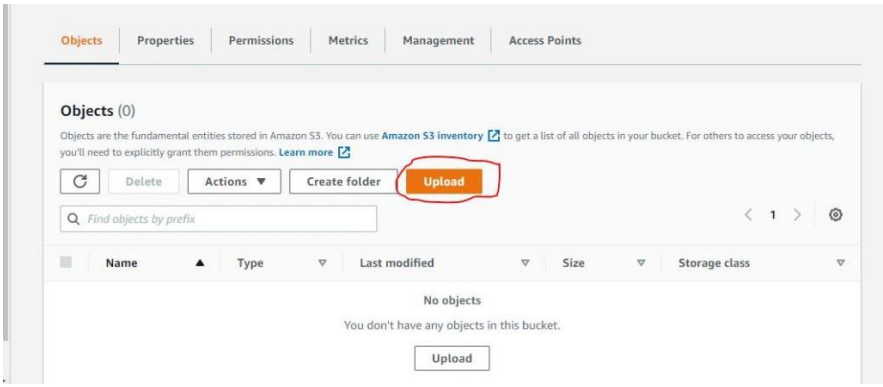


3.1 Upload an Object

Now, you can see your created S3 bucket in the list of buckets.

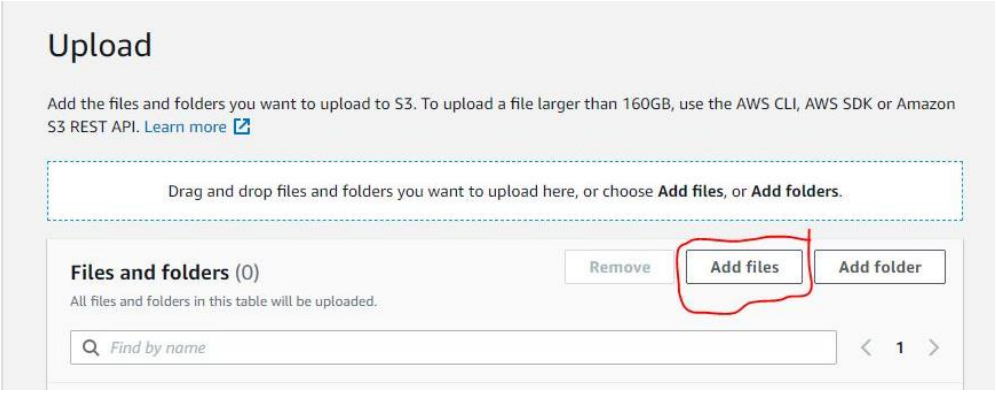
Create a simple text file named **demo.txt**

Click on the **bucket name** link to navigate inside the bucket. Once inside, you can upload your file.



Click on ***Upload.***

It will lead you to below screen where you can add files or even folders to upload in the bucket.



Click on ***Add files.***

Select file from your local system. Once file is loaded, Scroll down the page and click on ***Upload.***

3.2 View the Object

At this point of time, we have created an S3 bucket and uploaded a simple text file into it.

Now, it's time to view the uploaded file.

In the above screen, with the success message You can see your uploaded file in the Files and Folders section.

Click on **demo.txt**. Then you will see object overview screen.

Click on the ***Object URL*** to view the object.

P.S. – By default, all Amazon **S3** buckets and **objects** are private so on clicking above URL you will get similar error as below.

```
<Error>
<Code>AccessDenied</Code>
<Message>Access Denied</Message>
<RequestId>9X41902RKTPWRJY9</RequestId>
<HostId>thTGy7XWX8RAWbkbA7kCZWhYEZ4ENJhosN0M108EdQZ4273r4KTSVjWoTIJG/IPGsaS9Mfw3bYg=</HostId>
</Error>
```

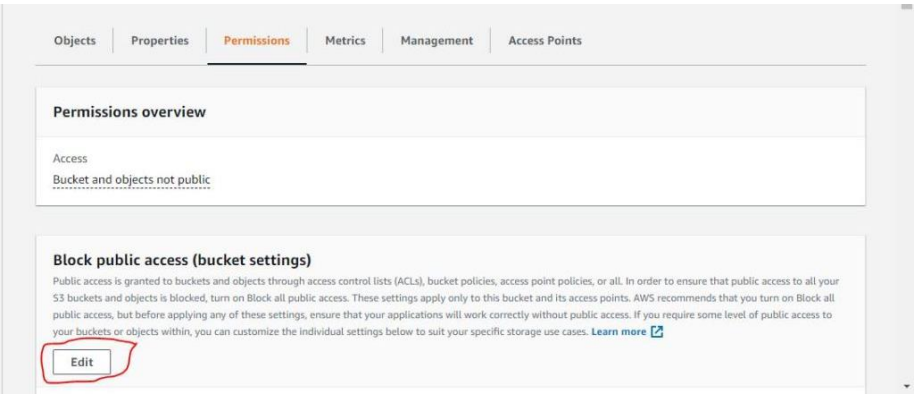
And the error is so obvious because the object is not publicly accessible. That's why you get **access denied**.

To fix this error follow below tasks.

<u>Task1:</u> *<u>Unblock Public Access on Bucket Level</u>*

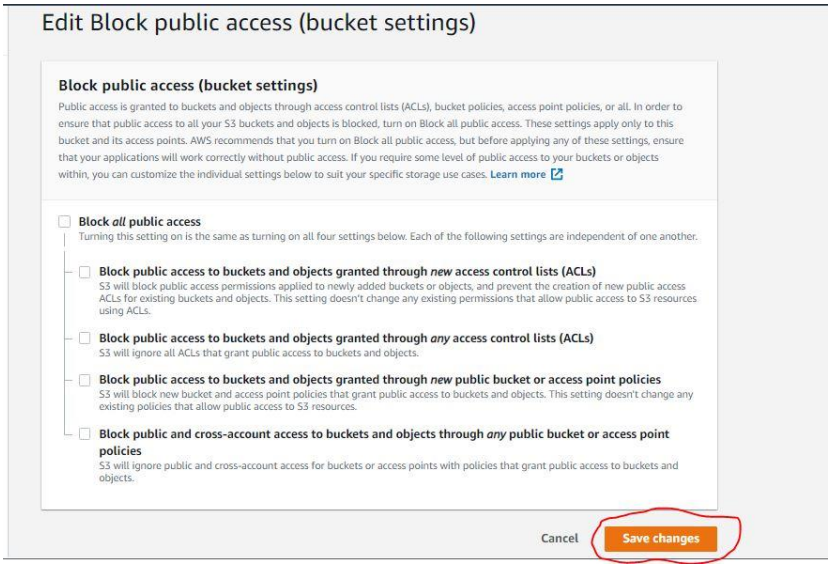
A newly created bucket is always private by default and all objects belonging to the bucket is private. So, unblock that setting on bucket level first.

Click on the **Permissions** tab on your bucket like below.

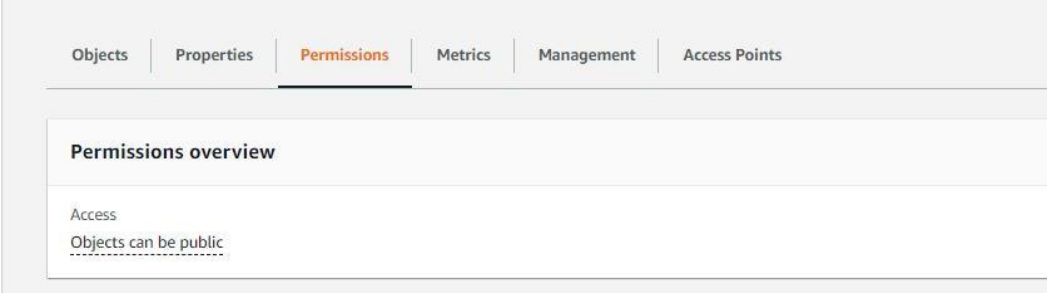


Click on ***Edit***.

Uncheck **Block *all* public access** checkbox like below.



Click on ***Save changes.*** Now as you can see below, objects can be public.



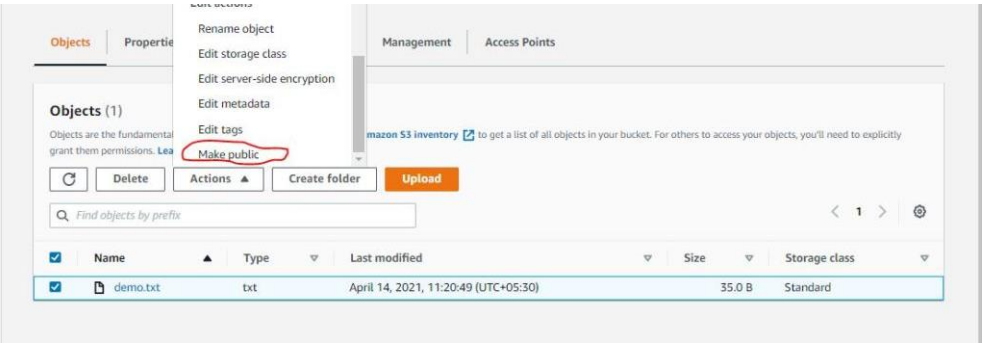
Note: Here notice that it says objects can be public and not as objects are public

Well, it means that now you can use various mechanism like **bucket policy** or **Access Control List** to allow public access on your object.

Task 2: Allow Public Access on Bucket.

Let's update the object's ***ACL*** to allow public read. You can do that from console using ***Make public*** action.

Select the object you would like to make publicly accessible.



Click on ***Actions*** drop-down and click ***Make public.*** Confirm the dialog box and your object is public now and you can view it publicly.

Go to Object -> Click on Object Name -> Click on Object URL.

Note: Please note that making an object public using this way makes only that specific object public and all other objects permission is unaffected.

4. Azure Disks

Use the following reference to learn about creating and using Azure Disks in Azure Portal.

https://docs.microsoft.com/en-us/azure-stack/user/azure-stack-manage-vm-disks?view=azs-2206&tabs=az1%2Caz2%2Caz3%2Caz4%2Caz5%2Caz6%2Caz7%2Caz8

5. Azure Files

Use the following reference to learn about creating and using an Azure file share in Azure Portal.

https://docs.microsoft.com/en-us/azure/storage/files/storage-how-to-use-files-portal?tabs=azure-portal

6. Azure Blobs

Use the following reference to learn about creating an Azure Storage table in the Azure portal.

https://docs.microsoft.com/en-us/azure/storage/blobs/storage-quickstart-blobs-portal