High Performance and Distributed Computing for Big Data

Unit 3: Cloud Systems and Big Data Management

Session 4 - AWS S3

Francesc Solsona Tehas francesc.solsona@udl.cat

Universitat Rovira i Virgili and Universitat de Lleida

Today's lecture

- AWS S3: Storing files in the cloud
  - Installing AWS CLI
  - Configuring AWS credentials
  - Creating an S3 bucket
  - Syncing a local directory to upload to a bucket
  - Loading files from the bucket to the notebook from python
  - Writing files from the notebook to the bucket from python
  - Syncinc a local directory to download from a bucket

S3 - Storing files in the cloud

What is S3?

Amazon S3 (Simple Storage Service) is an object storage service that offers industry-leading scalability, data availability, security, and performance. It is designed to store and retrieve any amount of data from anywhere on the web.

S3 - Buckets and Objects

- **Buckets**: In Amazon S3, a bucket is a unique container for objects. Every object is stored within a bucket.
  - The bucket name must be globally unique across all existing bucket names in Amazon S3.
  - Buckets are used to store and organize objects.
- **Objects**: Objects are the fundamental entities within a bucket. They consist of data and metadata.
  - The information within an object is stored as a **key-value** pair.
  - The key, which is a unique identifier, is used to organize objects within the bucket. It is often formatted as a prefix to the object name.

For example, we can create a bucket called hdcb-{your-name} and store objects organized by session or other criteria.

```
data-{your-name}/
    project1/data.csv
    project2/data.csv
```

Terms

- **Key** = *prefix* + *object name*
  - **prefix** = project1/ or project2/
  - **object name** = data.csv

S3 Pricing

Amazon S3 pricing is based on five factors:

1. **Storage Class**: The cost depends on the storage class used (Standard, Intelligent-Tiering, One Zone-IA, etc.).
2. **Storage**: The total volume of data stored per month.
3. **Requests**: The number and type of requests made.
4. **Data Transfer**: The cost of transferring data can vary by region and is also affected by whether data is transferred in or out.
5. **Management & Replication**: Additional features like data replication or management operations can also affect the cost.

For detailed information, you can refer to the Amazon S3 Pricing Page.

AWS CLI

What is the AWS CLI?

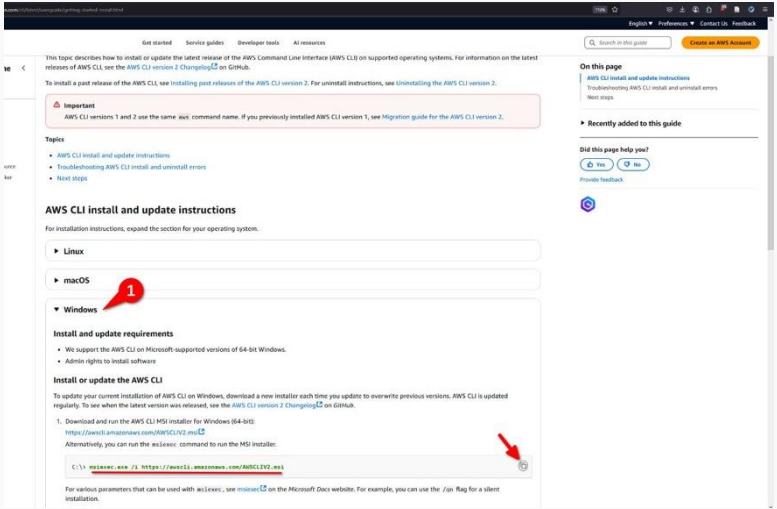
The AWS CLI is a tool that allows you to interact with AWS services from the command line. It is a powerful tool that can be used to automate tasks and manage your AWS resources.

```
[ec2-user@ip-172-31-86-82 ~]$ aws

usage: aws [options] <command> <subcommand> [<subcommand> ...] [parameters]
To see help text, you can run:

  aws help
  aws <command> help
  aws <command> <subcommand> help

aws: error: the following arguments are required: command

[ec2-user@ip-172-31-86-82 ~]$
```

This is the base command which by itself doesn't do anything. What we are going to do now is to configure the AWS CLI with our credentials so we can later run commands that can access other AWS resources like S3.

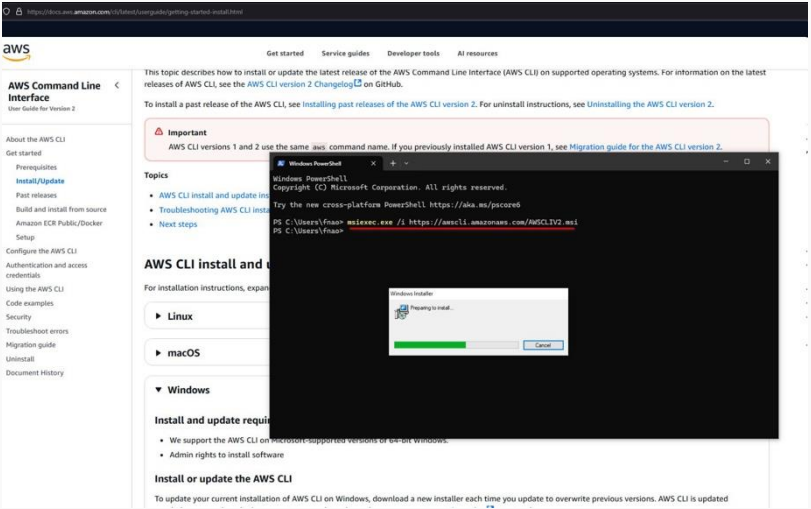
Installing the AWS CLI

Visit the AWS CLI installation guide to install the AWS CLI on your machine.



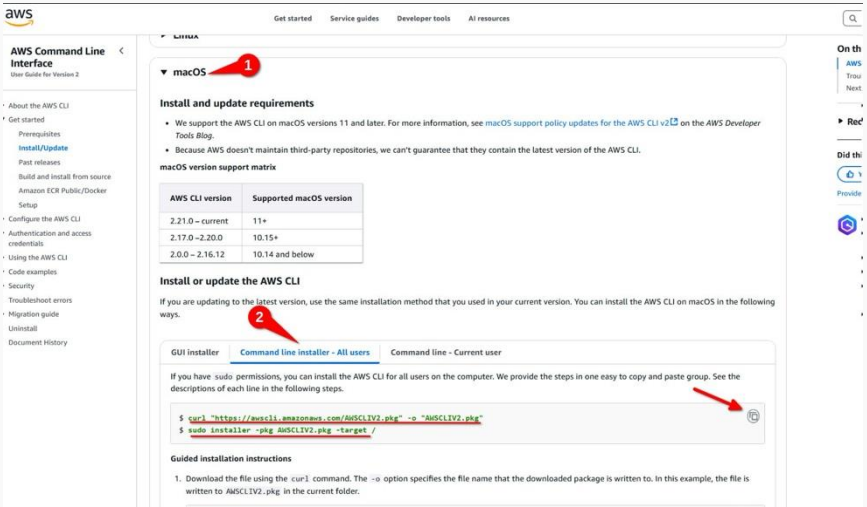
Installing the AWS CLI

Paste the command on the terminal and wait for the installation to complete.



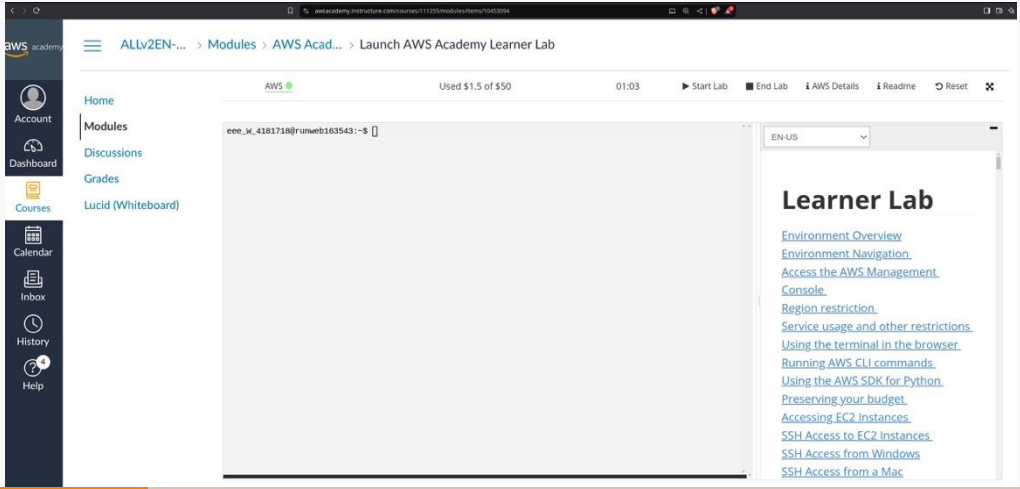
Installing the AWS CLI

Or if you are on MacOS, go to the MacOS section and also paste the corresponding commands on your terminal.



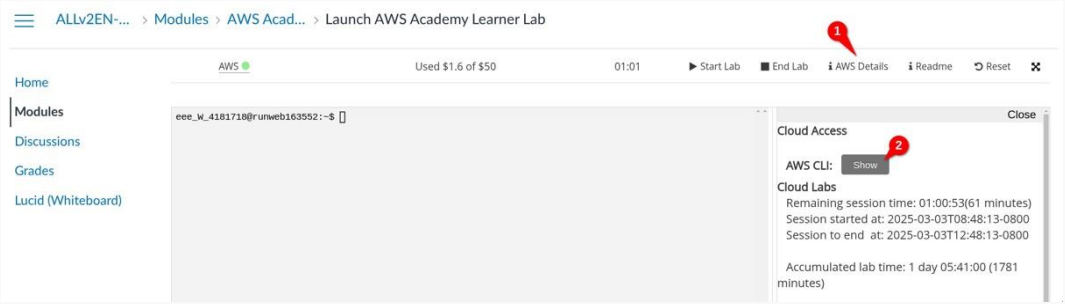
Configuring the AWS Credentials

We're now going to visit the Learner Lab page on the AWS Academy website to get our credentials. You have *this guide* and *this guide* available on the subject's *website* to help you with setting up AWS. Wait until the lab loads and you see the page below.



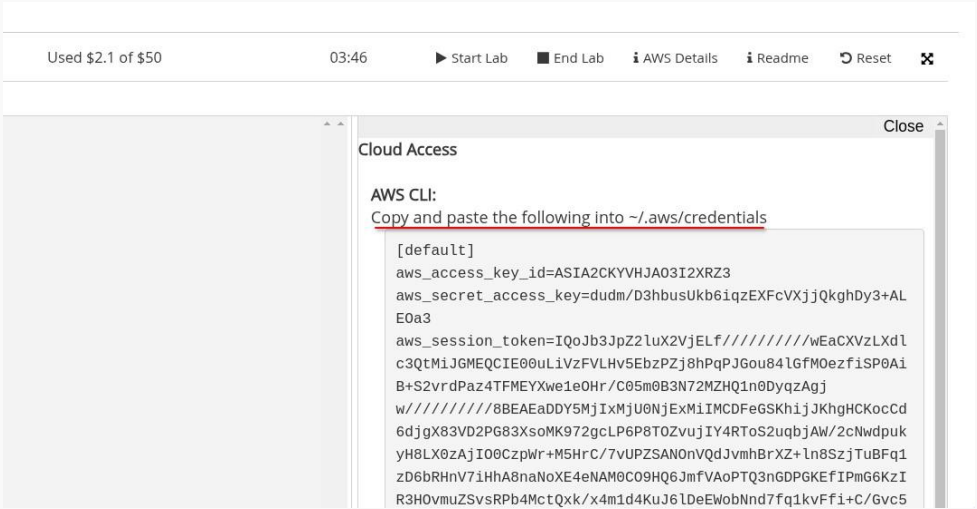
Configuring the AWS Credentials

Now click on `AWS Details` and then on `Show` to reveal your credentials.



Configuring the AWS Credentials

You'll see some text containing your credentials.



Configuring the AWS Credentials

Next we need to paste that text to a file called `credentials` inside the `.aws` folder in our home directory.

Make sure the `.aws` folder exists on your local machine by running `mkdir .aws` command (remember if it throws an error there's nothing to worry about, it just means the folder already exists). Now we are going to create the `credentials` file inside the `.aws` folder. Run the following command:

```
notepad .aws/credentials.
```

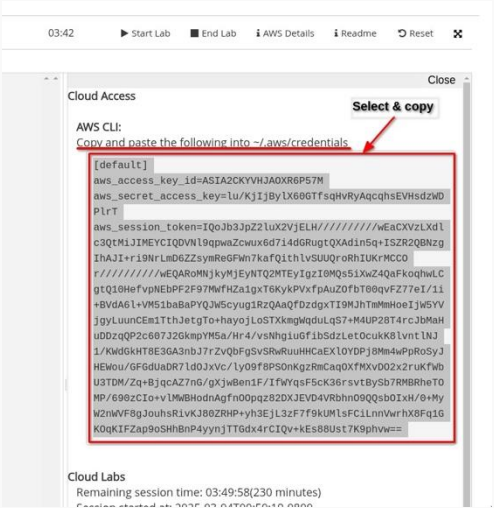
or for MacOS users:

```
open .aws/credentials.
```

This will open a text editor where you can write the credentials.

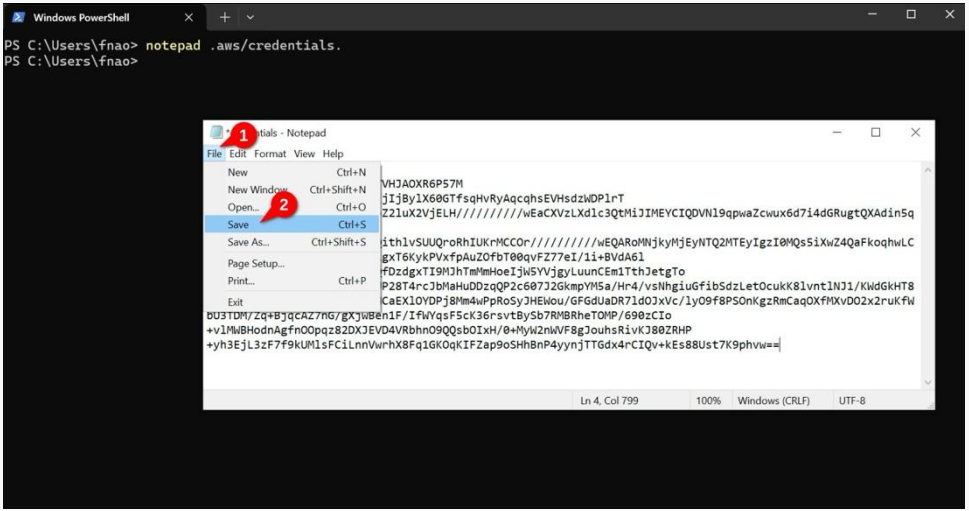
Configuring the AWS Credentials

Go back to the AWS Academy website and copy the text containing your credentials.



Configuring the AWS Credentials

Now go back to the text editor. Paste the credentials and save the file. You can now exit the text editor.



Configuring the AWS Credentials

We can check the contents of the file using `cat`:

```
PS C:\Users\fnao> cat .aws\credentials
[default]
aws_access_key_id=ASIA2CKYVHJAOXR6P57M
aws_secret_access_key=lu/KjIjBylX60GTfsqHvRyAqcqhsEVHsdzWDPlrT
aws_session_token=IQoJb3JpZ2...
PS C:\Users\fnao>
```

Configuring the AWS Credentials

To test if the configuration was successful, run `aws sts get-caller-identity` and you should see something like this:

```
PS C:\Users\fnao> aws sts get-caller-identity
{
    "UserId": "AROA2CKYVHJALK46ZMHVM:user3869188=Ferran_Aran_Test",
    "Account": "692212546112",
    "Arn":  "arn:aws:sts::692212546112:assumed-role/voclabs/user3869188=Ferran_Aran_Test"
}

PS C:\Users\fnao>
```

Great! We can now run AWS CLI commands on our local machine to manage our AWS resources. For example, we can upload files to an S3 bucket.

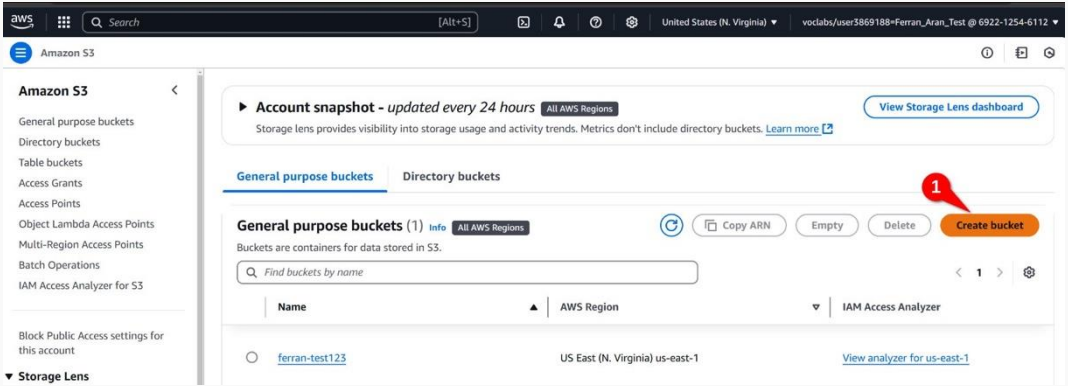
S3 Buckets

Creating a S3 bucket

Use the searchbar to head to the S3 service.

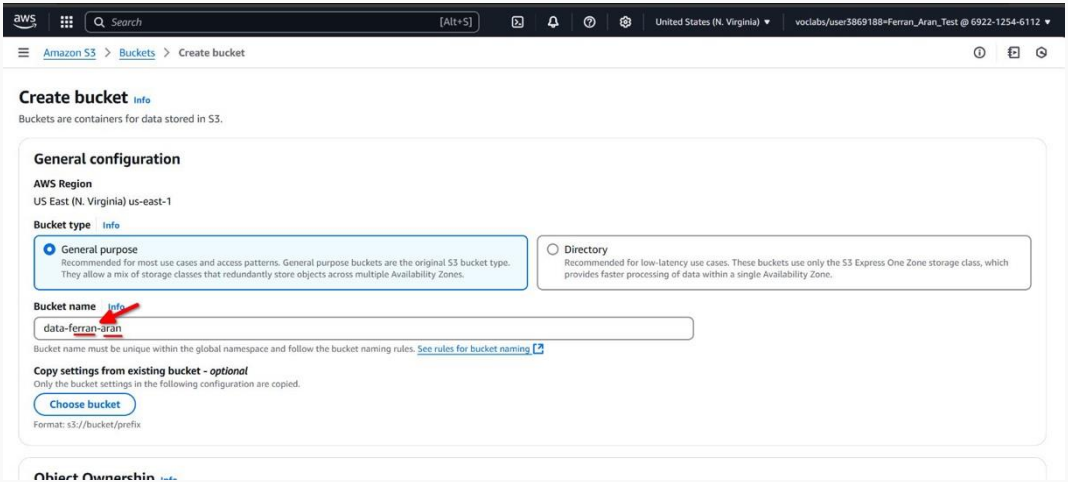
Creating a S3 bucket

Click on `Create bucket`.



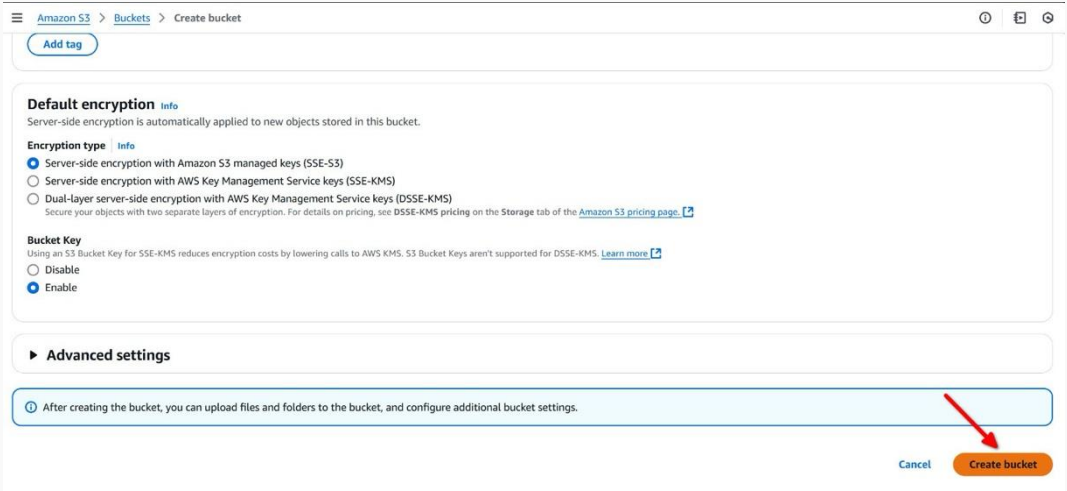
Creating a S3 bucket

Name the bucket `data-{your-name}`. For example I will be naming it `data-ferran-aran`. **Bucket names have to be unique across all of AWS**.



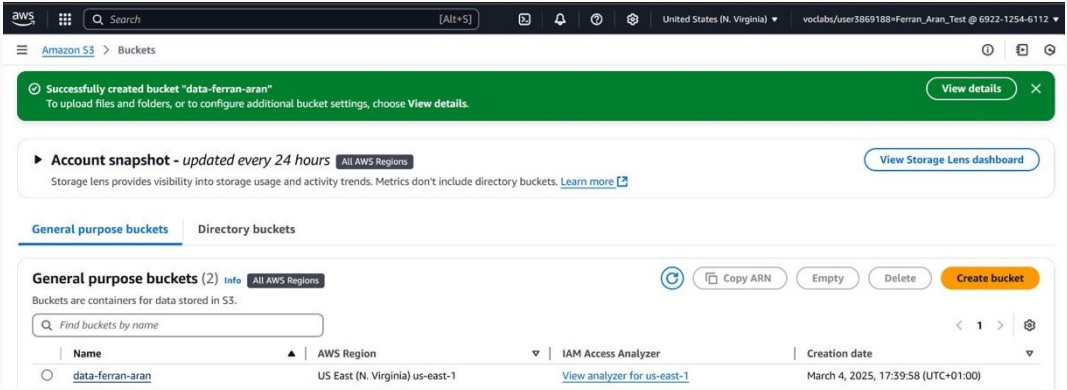
Creating a S3 bucket

Leave everything else as default, scroll all the way down and click on `Create bucket`.



Creating a S3 bucket

You will now be headed to the S3 dashboard. You should see a success message and the bucket you just created.



Uploading a file to the bucket

From the S3 dashboard

1. Click on the bucket.
2. Create a folder and name it `test`.
3. Click on the folder.
4. Click on Upload.
5. Click on Add files and select an example file.
6. Click Upload.

Our bucket **data-{your-name}/** now contains one file. It is stored in **s3://data-{your-name}/test**. This file is private by default. Only the owner can access them.

From the AWS CLI

- aws s3 cp <local-file-path> s3://<bucket-name>/<optional-target-name>

Example (upload photo.jpg to a bucket named my-bucket):

aws s3 cp photo.jpg s3://my-bucket/

Sync with a bucket

Syncing a local directory to upload to a bucket

From the AWS CLI

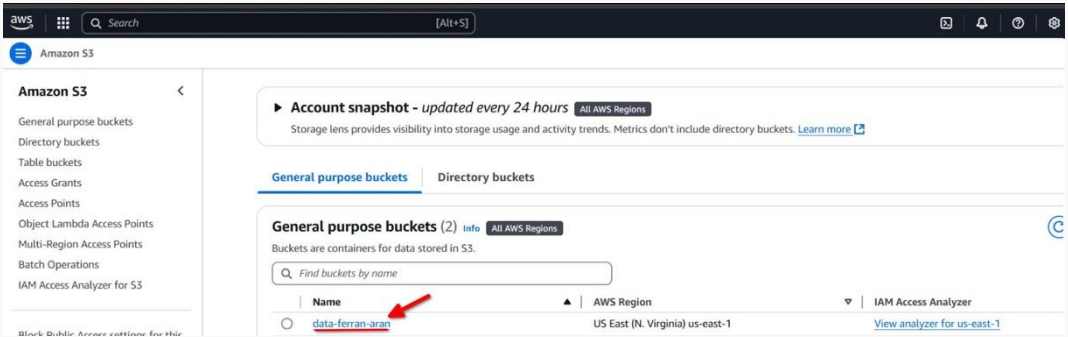
1. Go to your desktop and create a folder named `data`.
2. Create a file named `my-dataset.txt` and write some text in it. Save it.
3. Open a terminal and navigate to the Desktop directory. Use `cd Desktop`.
4. Run `aws s3 sync data s3://data-{your-name}` to upload the files to the bucket. In my case this is the result:

```
PS C:\Users\fnao\Desktop> aws s3 sync data s3://data-ferran-aran
upload: data\my-dataset.txt to s3://data-ferran-aran/my-dataset.txt
```

That's it! You have now uploaded a file to the bucket!

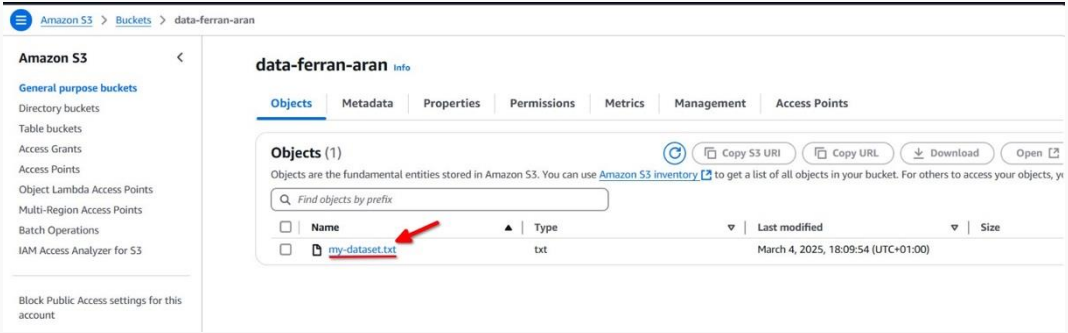
Inspecting the bucket from the AWS Console

Go to the S3 dashboard on AWS and click on the bucket we just created.



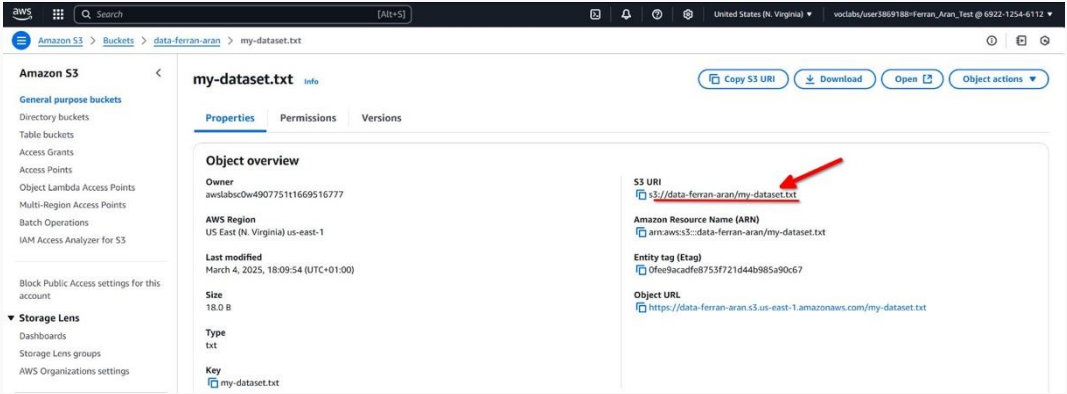
Inspecting the bucket from the AWS Console

You will see the contents of the bucket, in this case the file `my-dataset.txt`.



Inspecting the bucket from the AWS Console

Click on the file to see further details.



Working with S3 from python

Configuring AWS credentials on an EC2 instance

The first thing we'll have to do is to connect to our EC2 instance we configured during last session. More information on last session can be found *here (Session 3)*.

Once we are connected through SSH on a remote terminal, we'll need to configure AWS Credentials similarly to how we did on our local machine. But this time we will have to use a terminal editor.

If this steps get confusing I suggest checking *this guide* on the subject's website for a more detailed explanation.

Configuring AWS credentials on an EC2 instance

EC2 machines come with AWS CLI already installed so we won't have to worry about that.

Remember we first need to make sure the `.aws` folder exists in the home directory of the user we are using. If it doesn't exist we can create it by running `mkdir .aws`.

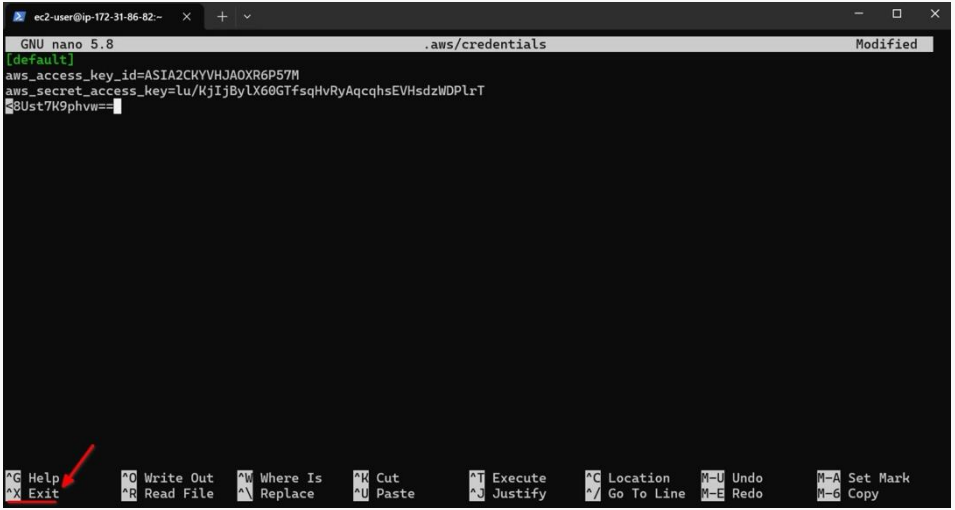
Next we need to create the `credentials` file inside the `.aws` folder. We can do this by running `nano .aws/credentials`.

The nano text editor will open and we can paste the credentials we copied from the AWS Academy website.

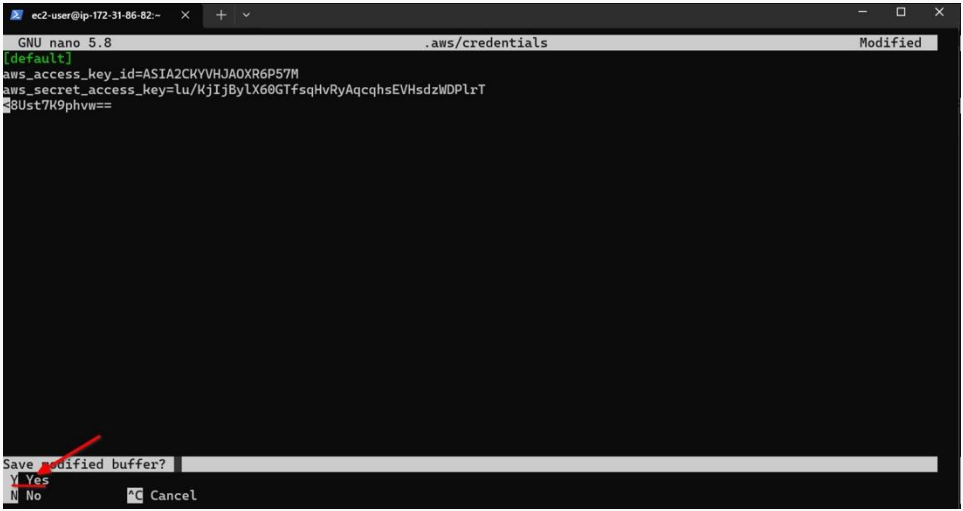
To save the file we can press `Ctrl + X` and then `Y` and finally `Enter`. See the following screenshots of the process.

Configuring AWS credentials on an EC2 instance

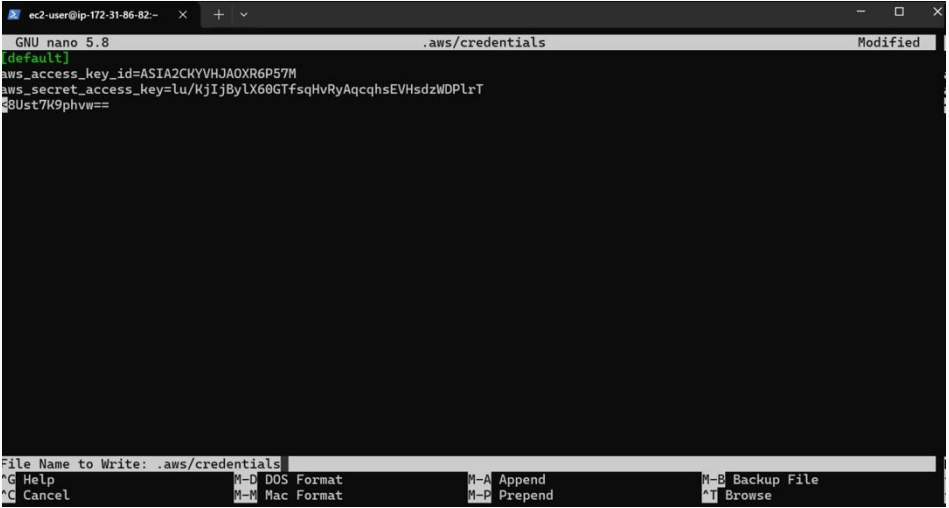
Configuring AWS credentials on an EC2 instance



Configuring AWS credentials on an EC2 instance



Configuring AWS credentials on an EC2 instance



Configuring AWS credentials on an EC2 instance

We can now test if the configuration was successful by running `aws sts get-caller-identity`.

If the configuration was successful we should see something like this:

```
{
    "UserId": "AROA2CKYVHJALK46ZMHVM:user3869188=Ferran_Aran_Test",
    "Account": "692212546112",
    "Arn":  "arn:aws:sts::692212546112:assumed-role/voclabs/user3869188=Ferran_Aran_Test"
}
```

Reading files from S3 with python

We are going to use the *boto3 python library* to access and write S3 files from a jupyter notebook. This library allows Python developers to write software that makes use of services like Amazon S3 on AWS.

We'll need to first activate one of the environment we created during the last session. For example, I will be using `project1` environment. Follow the steps below to activate the environment and install `boto3`:

```
cd project1
source .project1/bin/activate
pip install boto3
```

Reading files from S3 with python

Once that is done, launch the jupyter server with the command below:

```
jupyter notebook --no-browser --port=8888 --ip=0.0.0.0
```

Remember on last session we saw the steps to access the jupyter notebook from our local machine. If you need a refresher you can check *Session 3* on the subject's website.

Reading files from S3 with python

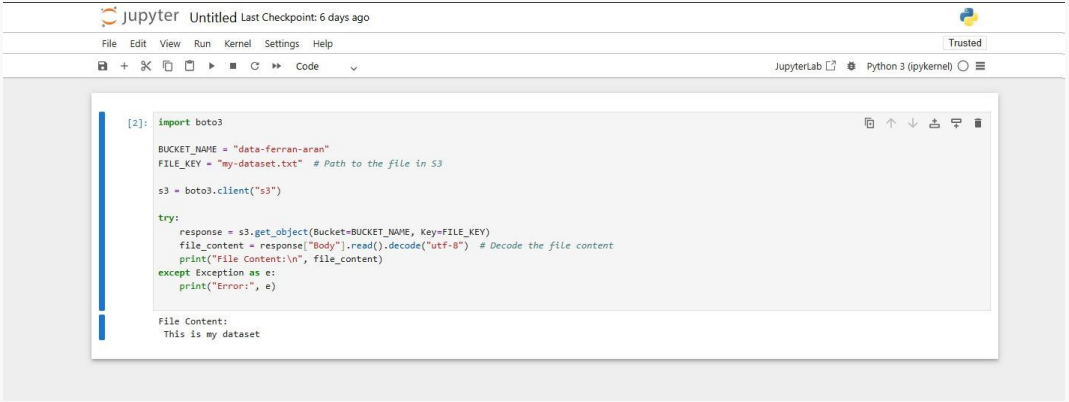
Open a jupyter notebook and paste the following code:

```
import boto3

DATA_BUCKET_NAME = "data-your-name"
DATA_FILE_NAME = "my-dataset.txt"  # Path to the file in S3

s3 = boto3.client("s3")

response = s3.get_object(Bucket=DATA_BUCKET_NAME, Key=DATA_FILE_NAME)
file_content = response["Body"].read().decode("utf-8")  # Decode the file content
print("File Content:\n", file_content)
```

Be careful to replace `data-your-name` with the name of the bucket you created and `my-dataset.txt` with the name of the file you uploaded.

Reading files from S3 with python

If everything is correct you should see the content of the file printed in the notebook.



RECAP: Accessing the file from the notebook

To access the file from the notebook we have used our aws-credentials for the current user (owner of the bucket).

Be aware!
Someone could log into your jupyter instance in the browser and access the file using your credentials.

What to do?
In real life, we would use IAM roles to give the notebook the necessary permissions to access the file.

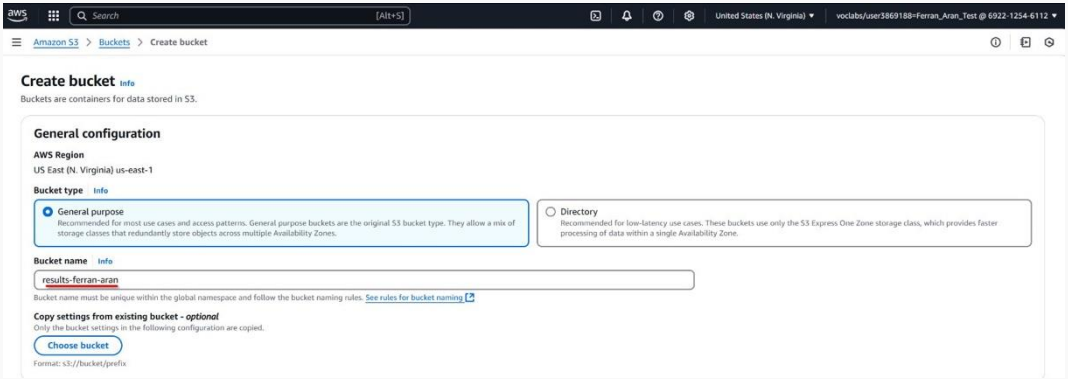
But, in AWS Educate, we can not use IAM roles.

Another option
Make the file public and access it without credentials :)

Writing files to S3 with python

We can also write files to S3 using boto3. Lets first create another bucket to store the files we will write from the notebook. This one we will call `results-{your-name}`.

Leave everything as default like before and scroll all the way down to click on `Create bucket`.



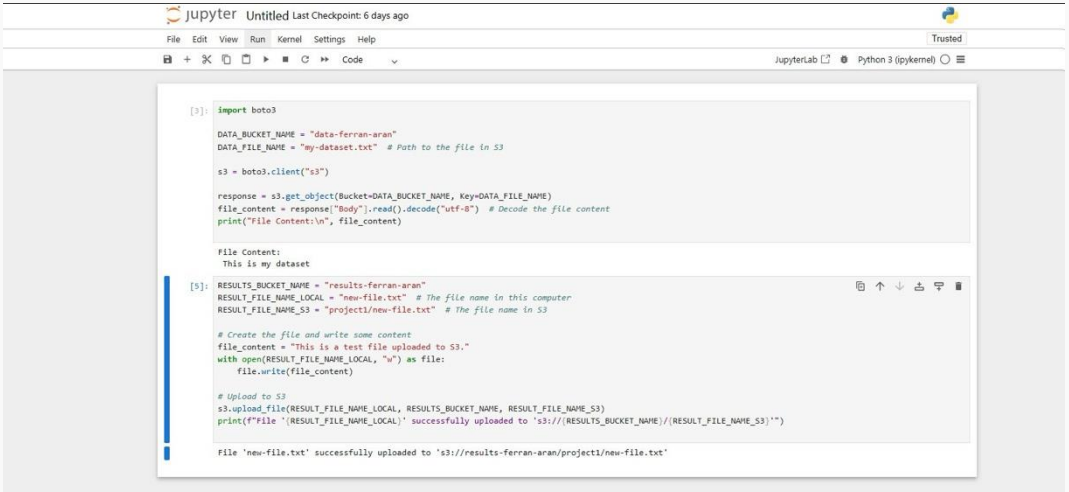
Writing files to S3 with python

We're now going to use the following code to write a file to the bucket we just created:

```
RESULTS_BUCKET_NAME = "results-your-name"
RESULT_FILE_NAME_LOCAL = "new-file.txt"  # The file name in this computer
RESULT_FILE_NAME_S3 = "project1/new-file.txt"  # The file name in S3

# Create the file and write some content file_content =
"This is a test file uploaded to S3." with
open(RESULT_FILE_NAME_LOCAL, "w") as file:
    file.write(file_content)

# Upload to S3
s3.upload_file(RESULT_FILE_NAME_LOCAL, RESULTS_BUCKET_NAME, RESULT_FILE_NAME_S3)
print(f"File '{RESULT_FILE_NAME_LOCAL}' successfully uploaded to 's3://{RESULTS_BUCKET_NAME}/
{RESULT_FILE_NAME_S3} ' ")
```

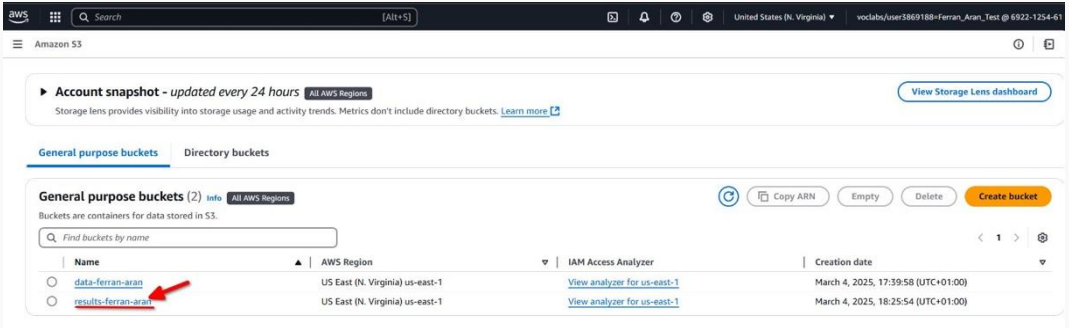
Writing files to S3 with python

If everything is correct you should see the following:



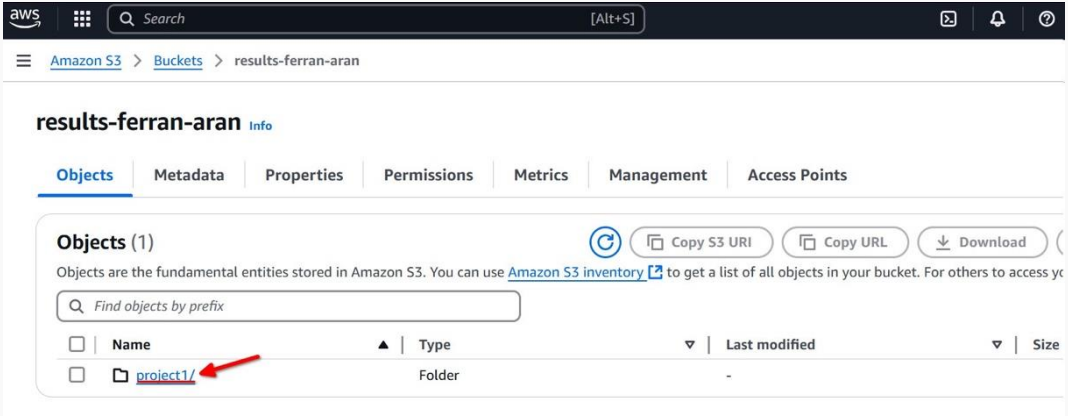
Inspecting the bucket from the AWS Console

We can now navigate to the S3 dashboard and click on the `results-{your-name}` bucket.



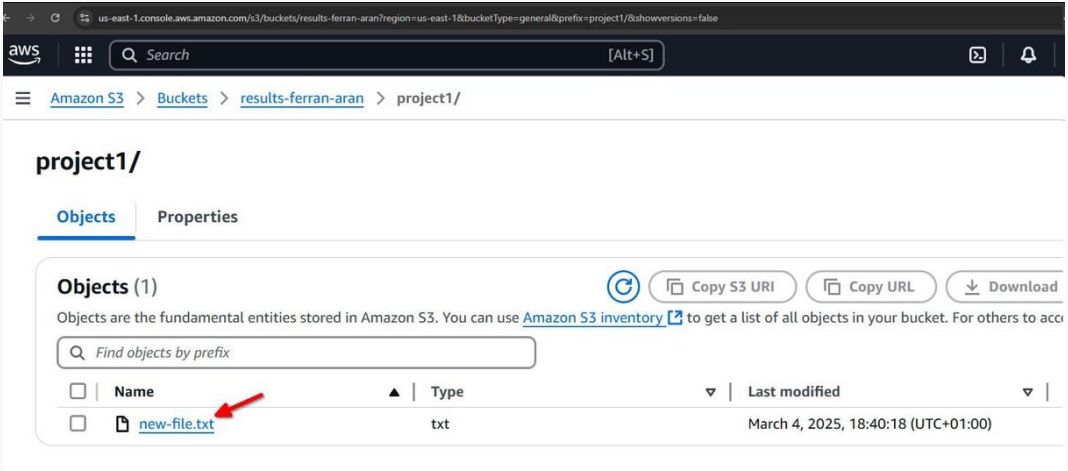
Inspecting the bucket from the AWS Console

Inside the bucket we should see the folder we just created, click on it.



Inspecting the bucket from the AWS Console

And now we should see the file we just uploaded.



Sync with a bucket

Syncing a local directory to download from a bucket

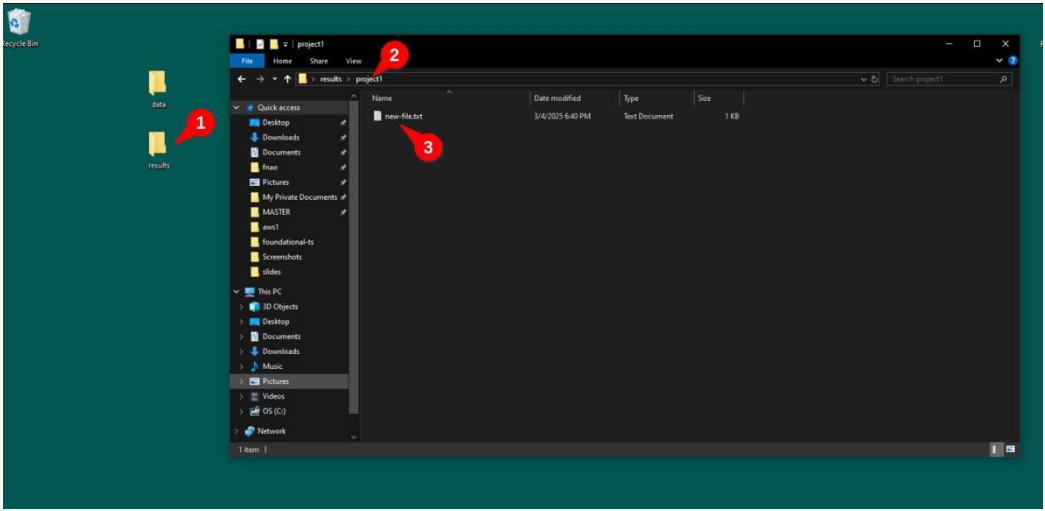
From the AWS CLI

1. Create a new folder on your desktop and name it `results`.
2. Open a terminal and navigate to the folder. Use `cd Desktop`.
3. Run `aws s3 sync s3://results-{your-name} results` to download the files from the bucket.
   In my case this is the result:

```
PS C:\Users\fnao\Desktop> aws s3 sync s3://results-ferran-aran results
download: s3://results-ferran-aran/project1/new-file.txt to results\project1\new-file.txt
PS C:\Users\fnao\Desktop>
```

Syncing a local directory to download from a bucket

We can use the File Explorer to navigate to the folder and see the file we just downloaded.



Recap

Recap - Today's contents

Today we have learned how to:

- Install the AWS CLI
- Configure AWS credentials
- Create an S3 bucket
- Sync a local directory to download from a bucket
- Sync a local directory to upload to a bucket
- Load files from the bucket to the notebook from python
- Write files from the notebook to the bucket from python

Recap - Use cases

- With the setup we have done today we ended up with a bucket on which we can **upload datasets that we have to work on**.
- This datasets can be **accessed from any machine** with internet connection and the necessary permissions.
- We also have a resulsts bucket which we can sync with our machine so **we have the results of our projects available locally**.

Recap - Subject's website

- Remember there is a *website* with useful information related to the subject.
- It has recently been updated with information about past sessions.
- Two new guides have been added to *Get started with AWS* and *Set up your Lab for every session*.