

High Performance and Distributed Computing for Big Data

Unit 3: AWS - S3

Ferran Aran Domingo ferran.aran@udl.cat

Universitat Rovira i Virgili and Universitat de Lleida

Today's lecture

- AWS S3: Storing files in the cloud

Today's lecture

- AWS S3: Storing files in the cloud
 - Installing AWS CLI

Today's lecture

- AWS S3: Storing files in the cloud
 - Installing AWS CLI
 - Configuring AWS credentials

Today's lecture

- AWS S3: Storing files in the cloud
 - Installing AWS CLI
 - Configuring AWS credentials
 - Creating an S3 bucket

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

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

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

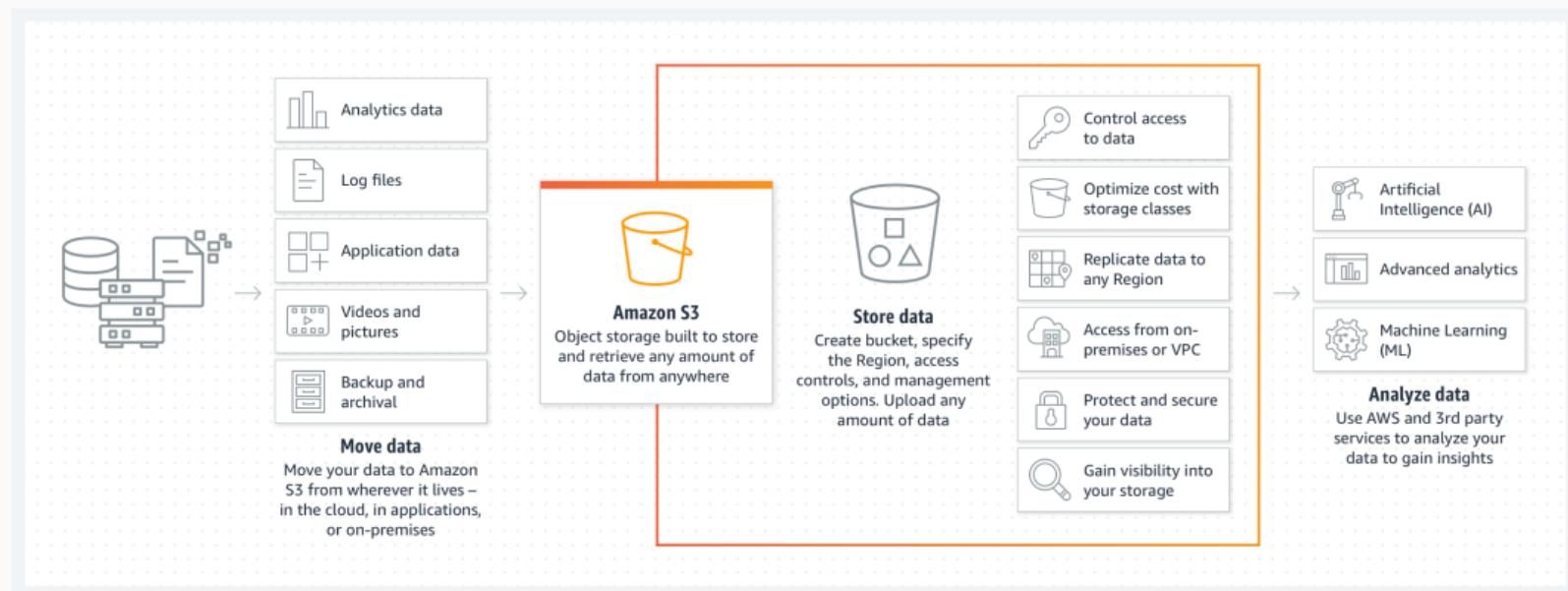
Today's lecture

- 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.

The screenshot shows the AWS CLI Getting Started guide on a computer screen. A red arrow labeled '1' points to the 'Windows' section under 'Topics'. Another red arrow points to the command line interface at the bottom of the page.

This topic describes how to install or update the latest release of the AWS Command Line Interface (AWS CLI) on supported operating systems. For information on the latest releases of AWS CLI, see the [AWS CLI version 2 Changelog](#) on GitHub.

To install a past release of the AWS CLI, see [Installing past releases of the AWS CLI version 2](#). For uninstall instructions, see [Uninstalling the AWS CLI version 2](#).

Important
AWS CLI versions 1 and 2 use the same `aws` command name. If you previously installed AWS CLI version 1, see [Migration guide for the AWS CLI version 2](#).

Topics

- [AWS CLI install and update instructions](#)
- [Troubleshooting AWS CLI install and uninstall errors](#)
- [Next steps](#)

AWS CLI install and update instructions

For installation instructions, expand the section for your operating system.

▶ [Linux](#)

▶ [macOS](#)

▼ [Windows](#) 1

Install and update requirements

- We support the AWS CLI on Microsoft-supported versions of 64-bit Windows.
- Admin rights to install software

Install or update the AWS CLI

To update your current installation of AWS CLI on Windows, download a new installer each time you update to overwrite previous versions. AWS CLI is updated regularly. To see when the latest version was released, see the [AWS CLI version 2 Changelog](#) on GitHub.

1. Download and run the AWS CLI MSI installer for Windows (64-bit):
<https://awscli.amazonaws.com/AWSCLIV2.msi>

Alternatively, you can run the `msiexec` command to run the MSI installer.

```
C:\> msiexec /i https://awscli.amazonaws.com/AWSCLIV2.msi
```

For various parameters that can be used with `msiexec`, see [msiexec](#) on the Microsoft Docs website. For example, you can use the `/qn` flag for a silent installation.

Installing the AWS CLI

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

The screenshot shows the AWS Command Line Interface User Guide for Version 2. The main content area discusses installing or updating the AWS CLI. A red box highlights an important note: "AWS CLI versions 1 and 2 use the same aws command name. If you previously installed AWS CLI version 1, see Migration guide for the AWS CLI version 2." Below this note, a Windows PowerShell window is displayed, showing the command `msieexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi` being run. A progress bar in the background indicates the installer is preparing to install.

This topic describes how to install or update the latest release of the AWS Command Line Interface (AWS CLI) on supported operating systems. For information on the latest releases of AWS CLI, see the [AWS CLI version 2 Changelog](#) on GitHub.

To install a past release of the AWS CLI, see [Installing past releases of the AWS CLI version 2](#). For uninstall instructions, see [Uninstalling the AWS CLI version 2](#).

Important

AWS CLI versions 1 and 2 use the same `aws` command name. If you previously installed AWS CLI version 1, see [Migration guide for the AWS CLI version 2](#).

Topics

- [AWS CLI install and update instructions](#)
- [Troubleshooting AWS CLI installations](#)
- [Next steps](#)

AWS CLI install and update instructions

For installation instructions, expand the section for your operating system:

- ▶ [Linux](#)
- ▶ [macOS](#)
- ▼ [Windows](#)

Install and update requirements

- We support the AWS CLI on Microsoft-supported versions of 64-bit Windows.
- Admin rights to install software

Install or update the AWS CLI

To update your current installation of AWS CLI on Windows, download a new installer each time you update to overwrite previous versions. AWS CLI is updated

Installing the AWS CLI

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

The screenshot shows the AWS Command Line Interface documentation page for macOS. The left sidebar contains links for the AWS CLI interface, including 'Install/Update'. The main content area has a heading 'Install and update requirements' with two bullet points. Below it is a 'macOS version support matrix' table:

AWS CLI version	Supported macOS version
2.21.0 – current	11+
2.17.0 – 2.20.0	10.15+
2.0.0 – 2.16.12	10.14 and below

Under 'Install or update the AWS CLI', there are three tabs: 'GUI installer', 'Command line installer - All users' (which is selected), and 'Command line - Current user'. The 'Command line installer - All users' tab contains instructions for installing the AWS CLI for all users. It includes a code block with two lines of bash commands:

```
$ curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg"  
$ sudo installer -pkg AWSCLIV2.pkg -target /
```

Below the code block is a 'copy' icon (a small square with a double-headed horizontal arrow).

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.

The screenshot shows the AWS Academy Learner Lab interface. The left sidebar contains navigation links: Account, Dashboard, Courses, Calendar, Inbox, History, and Help. The main content area shows a terminal window with the command 'eee_W_4181718@runweb163543:~\$'. To the right is a 'Learner Lab' sidebar with links to Environment Overview, Environment Navigation, Access the AWS Management Console, Region restriction, Service usage and other restrictions, Using the terminal in the browser, Running AWS CLI commands, Using the AWS SDK for Python, Preserving your budget, Accessing EC2 Instances, SSH Access to EC2 Instances, SSH Access from Windows, and SSH Access from a Mac.

aws academy

ALLv2EN... > Modules > AWS Acad... > Launch AWS Academy Learner Lab

Home AWS Used \$1.5 of \$50 01:03 Start Lab End Lab AWS Details Readme Reset

Account Dashboard Courses Calendar Inbox History Help

Modules

eee_W_4181718@runweb163543:~\$

EN-US

Learner Lab

[Environment Overview](#)
[Environment Navigation](#)
[Access the AWS Management Console](#)
[Region restriction](#)
[Service usage and other restrictions](#)
[Using the terminal in the browser](#)
[Running AWS CLI commands](#)
[Using the AWS SDK for Python](#)
[Preserving your budget](#)
[Accessing EC2 Instances](#)
[SSH Access to EC2 Instances](#)
[SSH Access from Windows](#)
[SSH Access from a Mac](#)

Configuring the AWS Credentials

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

The screenshot shows the AWS Academy Learner Lab interface. On the left, there's a sidebar with navigation links: Home, Modules (which is selected and highlighted in blue), Discussions, Grades, and Lucid (Whiteboard). The main content area has a breadcrumb trail: ALLv2EN-... > Modules > AWS Acad... > Launch AWS Academy Learner Lab. At the top right, there are several buttons: AWS (green circle), Used \$1.6 of \$50, 01:01, Start Lab, End Lab, AWS Details (highlighted with a red circle labeled '1'), Readme, Reset, and Close. A terminal window shows a command-line session: `eee_W_4181718@runweb163552:~$`. To the right of the terminal is a 'Cloud Access' panel with a 'Show' button (highlighted with a red circle labeled '2'). Below it is a 'Cloud Labs' section with session details: Remaining session time: 01:00:53(61 minutes), Session started at: 2025-03-03T08:48:13-0800, Session to end at: 2025-03-03T12:48:13-0800, and Accumulated lab time: 1 day 05:41:00 (1781 minutes).

Configuring the AWS Credentials

You'll see some text containing your credentials.

The screenshot shows a user interface for managing AWS credentials. At the top, there's a header with resource usage ("Used \$2.1 of \$50"), time ("03:46"), and navigation links ("Start Lab", "End Lab", "AWS Details", "Readme", "Reset", and a close button). Below this is a modal window titled "Cloud Access". Inside the modal, under the heading "AWS CLI:", there is a red underline over the instruction "Copy and paste the following into ~/.aws/credentials". A scrollable text area displays the following configuration:

```
[default]
aws_access_key_id=ASIA2CKYVHJA03I2XRZ3
aws_secret_access_key=dudm/D3hbusUkb6iqzEXFcVXjjQkghDy3+AL
E0a3
aws_session_token=IQoJb3JpZ2luX2VjELf//////////wEaCXVzLXdI
c3QtMiJGMEQCIE00uLiVzFVLhv5EbzPZj8hPqPJGou84lGfMOezfiSP0Ai
B+S2vrdPaz4TFMEYXwe1e0Hr/C05m0B3N72MZHQ1n0DyqzAgj
////////////////8BEAEaDDY5MjIxMjU0NjExMiIMCDFeGSKhijJKhgHKocCd
6djgX83VD2PG83XsoMK972gcLP6P8T0ZvujIY4RToS2uqbjAW/2cNwdpuK
yh8LX0zAjI00CzpWr+M5HrC/7vUPZSANOnVQdJvhBrXZ+ln8SzjTuBFq1
zD6bRHnV7iHhA8naNoXE4eNAM0C09HQ6JmfVAoPTQ3nGDPGKEfIPmG6KzI
R3H0vmuZSvsRPb4MctQxk/x4m1d4KuJ6lDeEWobNnd7fq1kvFfi+C/Gvc5
```

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.

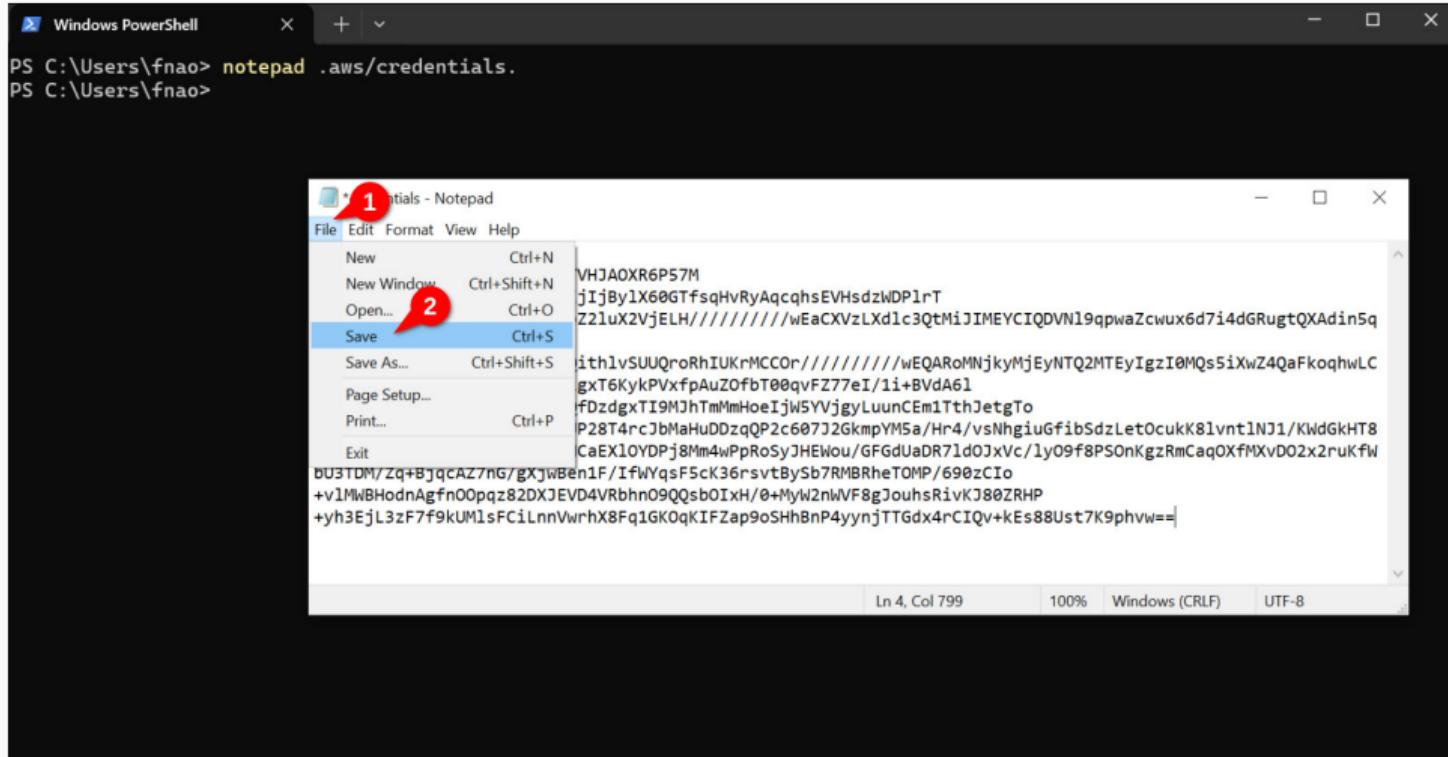
The screenshot shows a browser window for the AWS Academy Cloud Access lab. At the top, there are navigation buttons: 03:42, Start Lab, End Lab, AWS Details, Readme, Reset, and Close. Below these, a section titled "Cloud Access" displays the AWS CLI configuration. A red box highlights the configuration text, and a red arrow points from the "Select & copy" button above it to the highlighted area. The configuration text is as follows:

```
[default]
aws_access_key_id=ASIA2CKYVHJA0XR6P57M
aws_secret_access_key=lu/KjIjBylx60GTfsqHvRyAqcqhsEVHsdzWD
PlrT
aws_session_token=IQoJb3JpZ2luX2VjELH//////////wEaCXvzLXd1
c3QTMijIMEYCIQDVNl9qpwaZcwux6d714d6RugtQXAAdin5q+ISZR2QBNzg
IhAJI+r19NrLmD6ZsymReGFw7kafQithlvSUUQroRhIUkrMCC0
r//////////wEQAroMNjkyMjEyNTQMTExyIgZI0MQs51xwZ4QaFkoqhwlC
gtQ10HefvpNEbPF2F97MwfHzA1gxT6KykPVxfpAuZ0fbT00qvFZ77eI/11
+BVdA6l+VM51baBaPYQJW5cyug1rzQAAqfDzdgxTI9MJhTmMmHoeIjW5YV
jgyLuunCEm1ThJettGTo+hayojoLoSTXkmgWqduLqS7+M4UP2BT4rcJbMaH
uDDzqQP2c607J2GkmpYM5a/Hr4/vsNhgju6fibSdzLetOcukK8lvn1NJ
1/KwdGkHT8E3GA3nbJ7rZvQbfGsvSRwRuHHCaEXl0YDPj8Mm4wPpRoSyJ
HEWou/GFGduAdR7ldoJxvC/lY09f8PSOnKgzRmCaq0XfMXvD02x2ruKfwb
U3TDM/Zq+BjqcAZ7n6/gXjwBen1F/IfwYqsF5cK36rvst/Bs7RMRBReTO
MP/690zC1o+v1MwBHodnAgfn00pqz82DXJEVD4VRbhn09QQsb0IxH/0+My
W2nWVF8gJuohsR1vKJ80ZRHP+yh3EjL3zF7f9kUMlsFC1LnnVwrhX8Fq1G
K0qKIFZap9oSHhBnPyyynjTTGdx4rCIQv+kEs88Ust7K9phvw=
```

At the bottom of the window, the "Cloud Labs" section shows a remaining session time of 03:49:58 (230 minutes).

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"
}

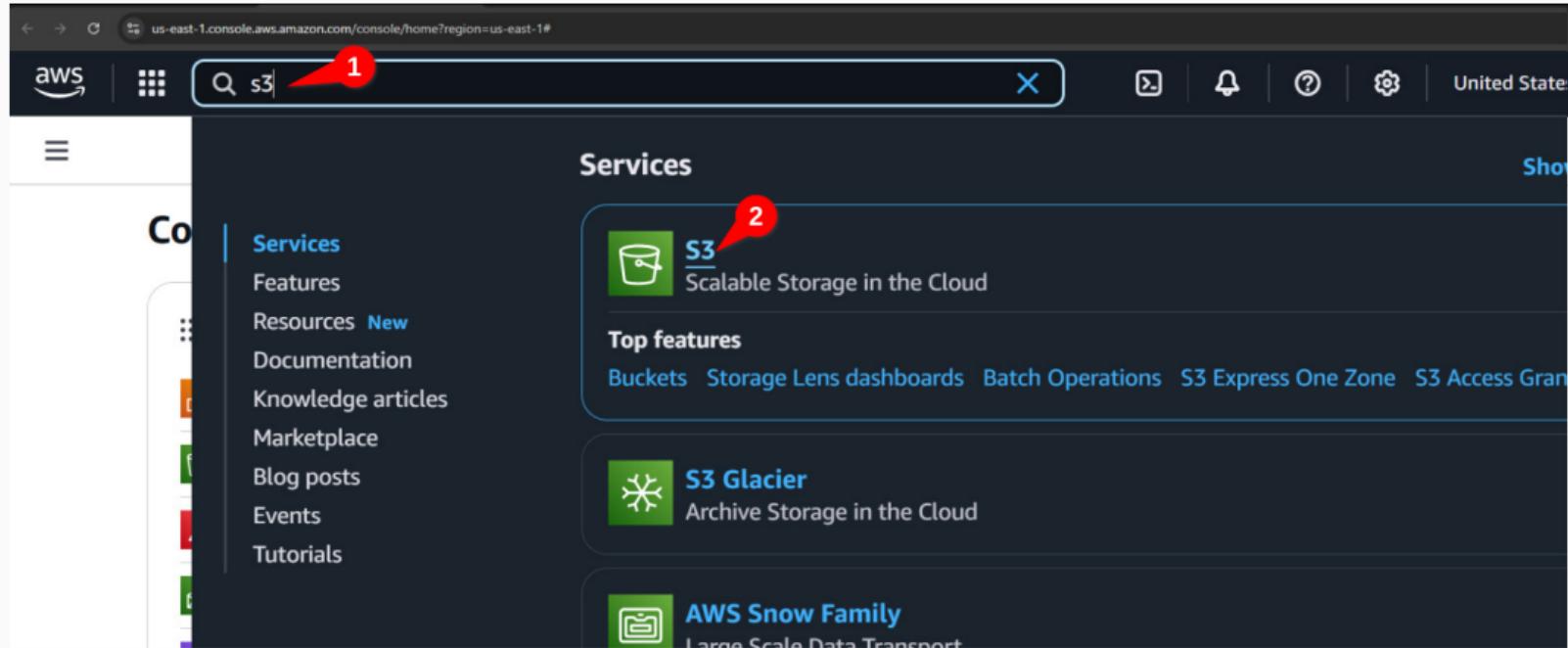
PS C:\Users\fnao>
```

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**.

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with various navigation links under 'Amazon S3'. The main area displays an 'Account snapshot - updated every 24 hours' section with a link to 'View Storage Lens dashboard'. Below this, there are two tabs: 'General purpose buckets' (which is selected) and 'Directory buckets'. Under the 'General purpose buckets' tab, it says '(1)' and there's a table with one row. The table has columns for 'Name' and 'AWS Region'. The single entry is 'ferran-test123' in 'US East (N. Virginia) us-east-1'. There are buttons for 'Copy ARN', 'Empty', 'Delete', and 'Create bucket'. A red circle with the number '1' is drawn around the 'Create bucket' button. At the bottom right of the table, there are navigation arrows and a settings gear icon.

Name	AWS Region	IAM Access Analyzer
ferran-test123	US East (N. Virginia) us-east-1	View analyzer for us-east-1

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.

The screenshot shows the 'Create bucket' configuration page in the AWS Management Console. The top navigation bar includes the AWS logo, search bar, and account information ('United States (N. Virginia)'). Below the navigation is a breadcrumb trail: 'Amazon S3 > Buckets > Create bucket'. On the left, a sidebar titled 'General configuration' lists 'AWS Region' (set to 'US East (N. Virginia) us-east-1') and 'Bucket type' (with 'General purpose' selected). The main configuration area has a blue border around the 'Bucket name' field, which contains the value 'data-ferran-aran'. A red arrow points to this field. Below the field, a note states: 'Bucket name must be unique within the global namespace and follow the bucket naming rules.' A link 'See rules for bucket naming' is provided. There is also a 'Copy settings from existing bucket - optional' section with a 'Choose bucket' button and a note about copied settings. At the bottom, there is an 'Object Ownership' section.

Create bucket Info

Buckets are containers for data stored in S3.

General configuration

AWS Region
US East (N. Virginia) us-east-1

Bucket type Info

General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name Info

`data-ferran-aran`

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional

Only the bucket settings in the following configuration are copied.

Choose bucket

Format: s3://bucket/prefix

Object Ownership

Creating a S3 bucket

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

The screenshot shows the 'Create bucket' wizard interface. At the top, there's a navigation bar with 'Amazon S3 > Buckets > Create bucket'. Below it, there's an 'Add tag' button. The main section is titled 'Default encryption' with an 'Info' link. It says 'Server-side encryption is automatically applied to new objects stored in this bucket.' Under 'Encryption type', the 'Server-side encryption with Amazon S3 managed keys (SSE-S3)' option is selected (radio button is checked). There are also other options: 'Server-side encryption with AWS Key Management Service keys (SSE-KMS)' and 'Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)'. A note below DSSE-KMS mentions secure pricing details. The next section is 'Bucket Key', which notes using an SSE-KMS key reduces costs. It has 'Disable' and 'Enable' options, with 'Enable' selected. A 'Learn more' link is provided. At the bottom, a note says 'After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.' On the right side, there are 'Cancel' and 'Create bucket' buttons. A red arrow points to the 'Create bucket' button.

Amazon S3 > Buckets > Create bucket

Add tag

Default encryption [Info](#)

Server-side encryption is automatically applied to new objects stored in this bucket.

Encryption type [Info](#)

Server-side encryption with Amazon S3 managed keys (SSE-S3)

Server-side encryption with AWS Key Management Service keys (SSE-KMS)

Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see [DSSE-KMS pricing](#) on the Storage tab of the [Amazon S3 pricing page](#).

Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. [Learn more](#)

Disable

Enable

▶ Advanced settings

After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.

Cancel **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.

The screenshot shows the AWS S3 Buckets dashboard. At the top, there is a green success message: "Successfully created bucket 'data-ferran-aran'. To upload files and folders, or to configure additional bucket settings, choose View details." Below this, there is an "Account snapshot - updated every 24 hours" section with a "View Storage Lens dashboard" button. The main area displays two tabs: "General purpose buckets" (selected) and "Directory buckets". Under "General purpose buckets", there is a table with one item:

Name	AWS Region	IAM Access Analyzer	Creation date
data-ferran-aran	US East (N. Virginia) us-east-1	View analyzer for us-east-1	March 4, 2025, 17:39:58 (UTC+01:00)

Actions for this bucket include: Copy ARN, Empty, Delete, and Create bucket.

Uploading a file to the bucket

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.

Sync with a bucket

Syncing a local directory to upload to a bucket

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 folder. 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.

The screenshot shows the AWS S3 dashboard. On the left, there's a sidebar with various options like General purpose buckets, Directory buckets, Table buckets, etc. The main area has a heading 'Account snapshot - updated every 24 hours' with a link to 'All AWS Regions'. Below it, a note says 'Storage lens provides visibility into storage usage and activity trends. Metrics don't include directory buckets.' with a 'Learn more' link. There are two tabs: 'General purpose buckets' (which is selected) and 'Directory buckets'. Under 'General purpose buckets', there's a sub-section titled 'General purpose buckets (2)' with a link to 'Info' and 'All AWS Regions'. It says 'Buckets are containers for data stored in S3.' Below this is a search bar with the placeholder 'Find buckets by name'. A table lists two buckets:

Name	AWS Region	IAM Access Analyzer
data-ferran-aran	US East (N. Virginia) us-east-1	View analyzer for us-east-1

Inspecting the bucket from the AWS Console

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

The screenshot shows the AWS S3 console interface. On the left, there's a sidebar with 'Amazon S3' navigation and a 'General purpose buckets' section containing links for Directory buckets, Table buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, and IAM Access Analyzer for S3. Below that is a link to 'Block Public Access settings for this account'. The main area is titled 'data-ferran-aran' and has tabs for Objects, Metadata, Properties, Permissions, Metrics, Management, and Access Points. The 'Objects' tab is selected. It displays a list of objects with one item: 'my-dataset.txt'. The file is a text file ('txt') last modified on March 4, 2025, at 18:09:54 (UTC+01:00). To the right of the object list are buttons for Copy S3 URI, Copy URL, Download, and Open.

Name	Type	Last modified	Size
my-dataset.txt	txt	March 4, 2025, 18:09:54 (UTC+01:00)	

Inspecting the bucket from the AWS Console

Click on the file to see further details.

The screenshot shows the AWS S3 console interface. On the left, there's a navigation sidebar with links for Amazon S3, General purpose buckets, Directory buckets, Table buckets, Access Grants, Access Points, Object Lambda Access Points, Multi-Region Access Points, Batch Operations, IAM Access Analyzer for S3, Block Public Access settings for this account, and Storage Lens (Dashboards, Storage Lens groups, AWS Organizations settings). The main content area shows a file named "my-dataset.txt" in the bucket "data-ferran-aran". The "Properties" tab is selected. The "Object overview" section contains the following details:

Attribute	Value
Owner	awslabsc0w4907751t1669516777
AWS Region	US East (N. Virginia) us-east-1
Last modified	March 4, 2025, 18:09:54 (UTC+01:00)
Size	18.0 B
Type	txt
Key	my-dataset.txt

On the right side, there are several actions: Copy S3 URI, Download, Open, and Object actions. A red arrow points to the "S3 URI" link, which is underlined and shows the value <s3://data-ferran-aran/my-dataset.txt>.

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](#)*.

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](#)*.

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.

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*](#).

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 these 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.

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`.

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`.

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.

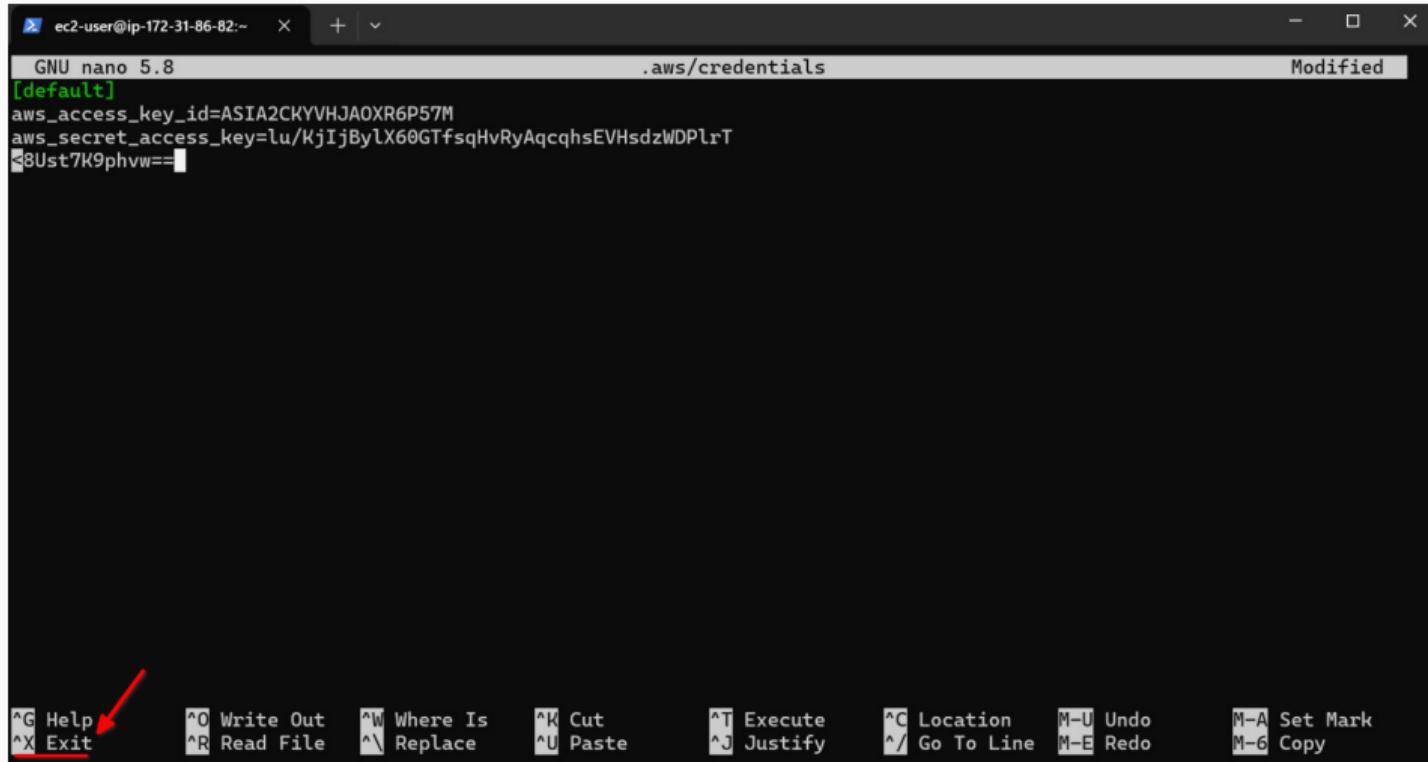
To save the file we can press `Ctrl + X` and then `Y` and finally `Enter`. See the following screenshots of the process.

Configuring AWS credentials on an EC2 instance

The screenshot shows a terminal window titled "ec2-user@ip-172-31-86-82:~". The window title bar also displays "GNU nano 5.8", the file path ".aws/credentials", and the status "Modified". The main area of the terminal is a black space where the AWS configuration file would be displayed. At the bottom of the screen, there is a menu bar with various keyboard shortcut commands for navigating and editing files.

^G Help	^O Write Out	^W Where Is	^K Cut	^T Execute	^C Location	M-U Undo	M-A Set Mark
^X Exit	^R Read File	^V Replace	^U Paste	^J Justify	^/ Go To Line	M-E Redo	M-C Copy

Configuring AWS credentials on an EC2 instance



GNU nano 5.8 .aws/credentials Modified
[default]
aws_access_key_id=ASIA2CKYVHJA0XR6P57M
aws_secret_access_key=lw/KjIjBylX60GTfsqHvRyAqcqhsEVHsdzWDPlrT
8Ust7K9phvw==

^G Help ^O Write Out ^W Where Is ^K Cut ^T Execute ^C Location M-U Undo
^X Exit ^R Read File ^\ Replace ^U Paste ^J Justify ^/ Go To Line M-E Redo M-A Set Mark
M-6 Copy

Configuring AWS credentials on an EC2 instance

The screenshot shows a terminal window titled "ec2-user@ip-172-31-86-82:~". Inside the terminal, the file ".aws/credentials" is open in nano editor. The file contains the following content:

```
[default]
aws_access_key_id=ASIA2CKYVHJA0XR6P57M
aws_secret_access_key=lu/KjIjBylX60GTfsqHvRyAqcqhsEVHsdzWDPlrT
S8Ust7k9phvw==
```

At the bottom of the terminal, a modal dialog box is displayed, asking "Save modified buffer?". The "Yes" option is highlighted with a red arrow.

Save modified buffer?
Y Yes
N No ^C Cancel

Configuring AWS credentials on an EC2 instance

The screenshot shows a terminal window titled "ec2-user@ip-172-31-86-82:~". The window displays a file named ".aws/credentials" using the "GNU nano 5.8" editor. The file contains the following content:

```
[default]
aws_access_key_id=ASIA2CKYVHJA0XR6P57M
aws_secret_access_key=lu/KjIjBylX60GTfsqHvRyAqcqhsEVHsdzWDPLrT
S8Ust7K9phvw==
```

The terminal interface includes a menu bar with "File", "Edit", "Search", "Help", and "Exit". At the bottom, there is a status bar with the text "File Name to Write: .aws/credentials" and a set of keyboard shortcuts:

^G Help	M-D DOS Format	M-A Append	M-B Backup File
^C Cancel	M-M Mac Format	M-P Prepend	^T Browse

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`.

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 be using 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, launc 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.

The screenshot shows a Jupyter Notebook interface with the following details:

- Title Bar:** jupyter Untitled Last Checkpoint: 6 days ago
- Kernel Status:** Trusted
- Toolbar:** File Edit View Run Kernel Settings Help, with icons for Cell, Insert Cell, Run Cell, Stop Cell, Kernel, Help, and Cell Type.
- Cell Header:** [2]:
- Code Content:**

```
import boto3

BUCKET_NAME = "data-ferran-aran"
FILE_KEY = "my-dataset.txt" # Path to the file in S3

s3 = boto3.client("s3")

try:
    response = s3.get_object(Bucket=BUCKET_NAME, Key=FILE_KEY)
    file_content = response["Body"].read().decode("utf-8") # Decode the file content
    print("File Content:\n", file_content)
except Exception as e:
    print("Error:", e)
```
- Output Content:**

```
File Content:
This is my dataset
```

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).

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.

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.

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 that 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`.

The screenshot shows the AWS Management Console interface for creating a new S3 bucket. The top navigation bar includes the AWS logo, a search bar, and a 'Create bucket' button. Below the navigation, the breadcrumb trail shows 'Amazon S3 > Buckets > Create bucket'. The main content area is titled 'Create bucket' with a 'General configuration' section.

General configuration

AWS Region: US East (N. Virginia) us-east-1

Bucket type: [Info](#)

General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.

Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.

Bucket name: [Info](#)
results-ferran-aran

Bucket name must be unique within the global namespace and follow the bucket naming rules. [See rules for bucket naming](#)

Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.

[Choose bucket](#)

Format: s3://bucket/prefix

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:

The screenshot shows a Jupyter Notebook interface with the title "jupyter Untitled Last Checkpoint: 6 days ago". The menu bar includes File, Edit, View, Run, Kernel, Settings, Help, and a Trusted button. The toolbar has icons for file operations like Open, Save, and Run.

The notebook contains two code cells:

Cell [3]:

```
import boto3

DATA_BUCKET_NAME = "data-ferran-aran"
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)
```

Output:

```
File Content:
This is my dataset
```

Cell [5]:

```
RESULTS_BUCKET_NAME = "results-ferran-aran"
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}'")
```

Output:

```
File 'new-file.txt' successfully uploaded to 's3://results-ferran-aran/project1/new-file.txt'
```

Inspecting the bucket from the AWS Console

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

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various icons. Below it, the main title is "Amazon S3". On the left, there are tabs for "General purpose buckets" (which is selected) and "Directory buckets". In the center, the heading "General purpose buckets (2)" is displayed, with "Info" and "All AWS Regions" buttons. A note below says "Buckets are containers for data stored in S3." To the right of the table are buttons for "Copy ARN", "Empty", "Delete", and "Create bucket". The table itself has columns for "Name", "AWS Region", "IAM Access Analyzer", and "Creation date". Two rows are listed:

Name	AWS Region	IAM Access Analyzer	Creation date
data-ferran-aran	US East (N. Virginia) us-east-1	View analyzer for us-east-1	March 4, 2025, 17:39:58 (UTC+01:00)
results-ferran-aran	US East (N. Virginia) us-east-1	View analyzer for us-east-1	March 4, 2025, 18:25:54 (UTC+01:00)

Inspecting the bucket from the AWS Console

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

The screenshot shows the AWS S3 console interface. At the top, there's a navigation bar with the AWS logo, a search bar containing 'Search [Alt+S]', and three small icons. Below the navigation bar, the path 'Amazon S3 > Buckets > results-ferran-aran' is displayed. The main title 'results-ferran-aran' is followed by an 'Info' link. Below the title, there are tabs for 'Objects', 'Metadata', 'Properties', 'Permissions', 'Metrics', 'Management', and 'Access Points'. The 'Objects' tab is currently selected. Under the 'Objects' tab, the heading 'Objects (1)' is shown. To the right of this heading are three buttons: 'Copy S3 URI', 'Copy URL', and 'Download'. Below the heading, a sub-instruction says 'Objects are the fundamental entities stored in Amazon S3. You can use [Amazon S3 inventory](#) to get a list of all objects in your bucket. For others to access yo...'. A search bar with the placeholder 'Find objects by prefix' is present. The main table lists one object: 'project1/' which is a 'Folder'. A red arrow points to the 'project1/' entry in the table. The table has columns for 'Name', 'Type', 'Last modified', and 'Size'.

Name	Type	Last modified	Size
project1/	Folder	-	-

Inspecting the bucket from the AWS Console

And now we should see the file we just uploaded.

The screenshot shows the AWS S3 console interface. The URL in the browser bar is `us-east-1.console.aws.amazon.com/s3/buckets/results-ferran-aran?region=us-east-1&bucketType=general&prefix=project1/&showversions=false`. The navigation path is `Amazon S3 > Buckets > results-ferran-aran > project1/`. The page title is `project1/`. The 'Objects' tab is selected, showing one object: `new-file.txt`. The object details are: Name: `new-file.txt`, Type: `txt`, Last modified: `March 4, 2025, 18:40:18 (UTC+01:00)`. There are buttons for `Copy S3 URI`, `Copy URL`, and `Download`. A search bar at the bottom left contains the placeholder `Find objects by prefix`.

Name	Type	Last modified
<code>new-file.txt</code>	<code>txt</code>	<code>March 4, 2025, 18:40:18 (UTC+01:00)</code>

Sync with a bucket

Syncing a local directory to download from a bucket

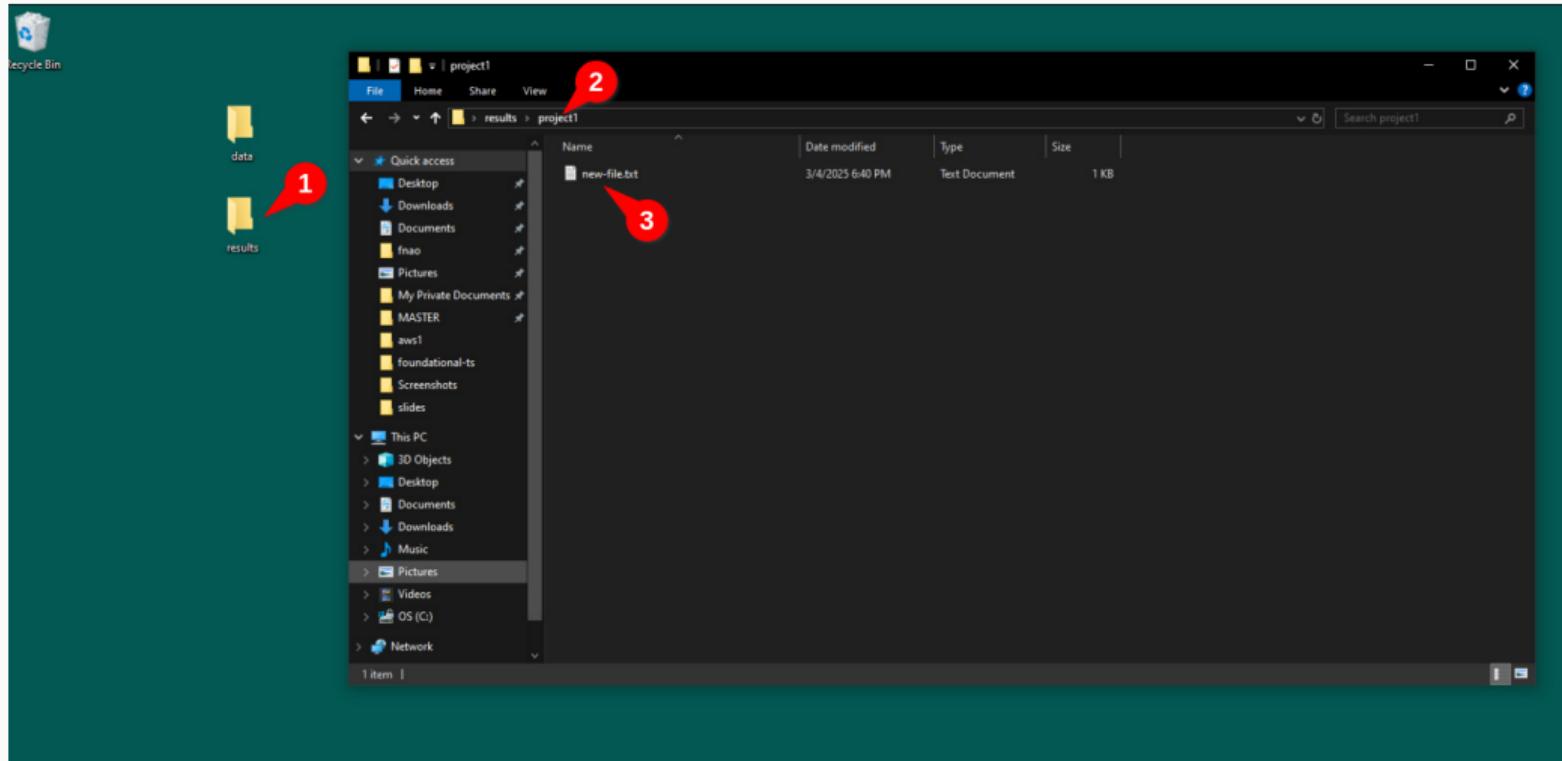
1. Create a new folder on your desktop and name it `results`.
2. Open a terminal and navigate to the folder. Use `cd Desktop`.
3. Run `aws s3 sync s3://results-{your-name} results` to download the files from the bucket.

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 results 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*.