AWS CLI

> **AWS CLI is a command-line tool that lets you control AWS services directly from your Windows terminal using text commands.**

Why it matters?

- ✅ Automate tasks (upload files to S3, launch EC2, etc.)
  - e.g., backup files daily
- ✅ Run scripts instead of clicking the AWS console
- ✅ Use AWS with Python (`boto3`) more securely
- ✅ **Start/stop servers** (EC2 instances).
- ✅ **Manage security permissions** (IAM users).
- ✅ **And 200+ other AWS services!**

✅1: Install AWS CLI on Windows

Installing or updating to the latest version of the AWS CLI - AWS Command Line Interface

The AWS CLI is an open source tool built using the AWS SDK for Python (Boto) that provides commands for interacting with AWS services. With minimal configuration, you can start using all of the functionality provided by the AWS Management Console from your favorite terminal program.

📦 https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

▼ **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 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.

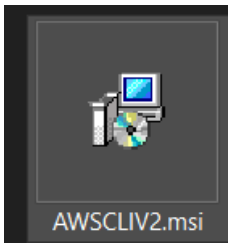
   ```
   C:\> msiexec.exe /i https://awscli.amazonaws.com/AWSCLIV2.msi
   ```

   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.

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

2. To confirm the installation, open the **Start** menu, search for `cmd` to open a command prompt window, and at the command prompt use the `aws --version` command.

   ```
   C:\> aws --version
   ```

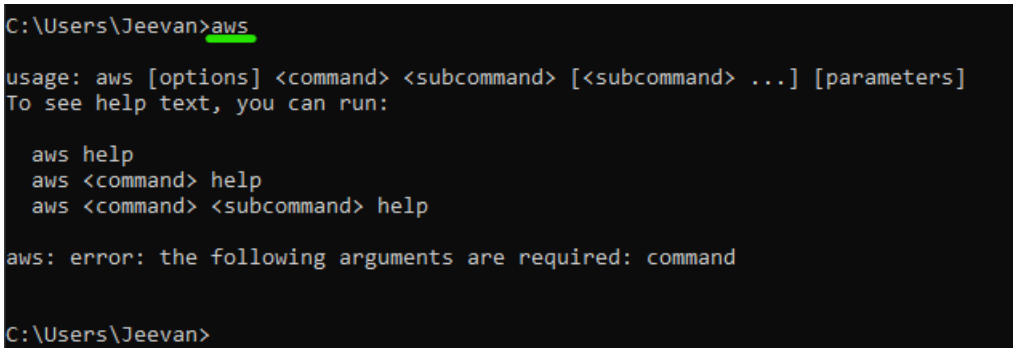


- Install

Verify Installation:

- In command prompt type:

```
aws
```



```
aws --version
```

```
C:\Users\Jeevan>aws --version
aws-cli/2.27.38 Python/3.13.4 Windows/10 exe/AMD64
```

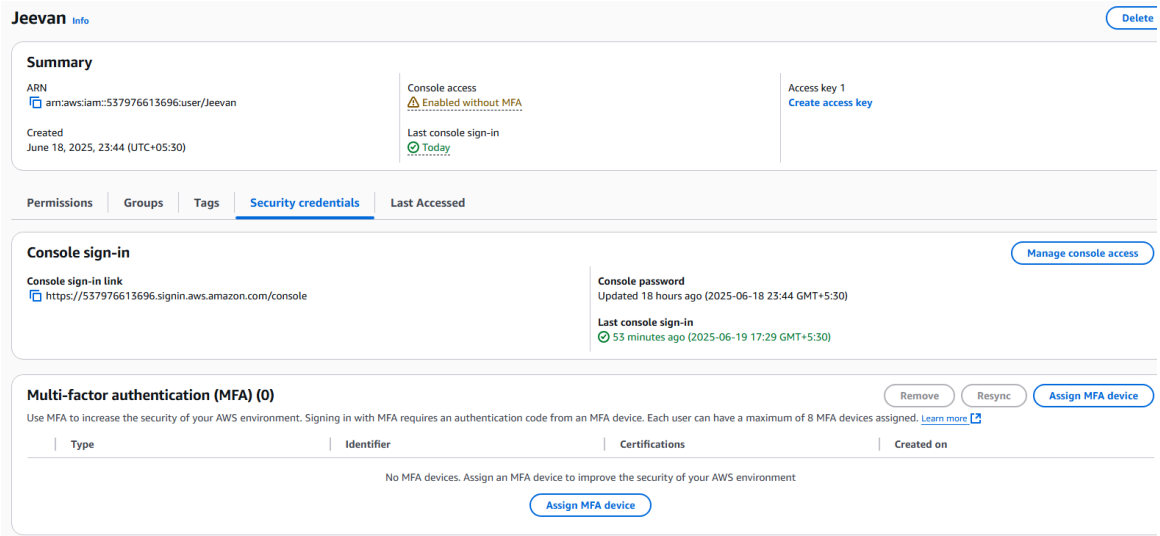
2. Configure AWS CLI

- Now, connect your CLI to your AWS account using your **IAM user's credentials**.

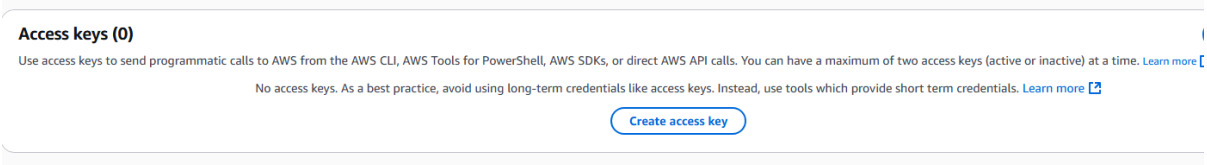
> 💡 **You will need your IAM user's Access Key ID and Secret Access Key.**

Get Access Key + Secret Key:

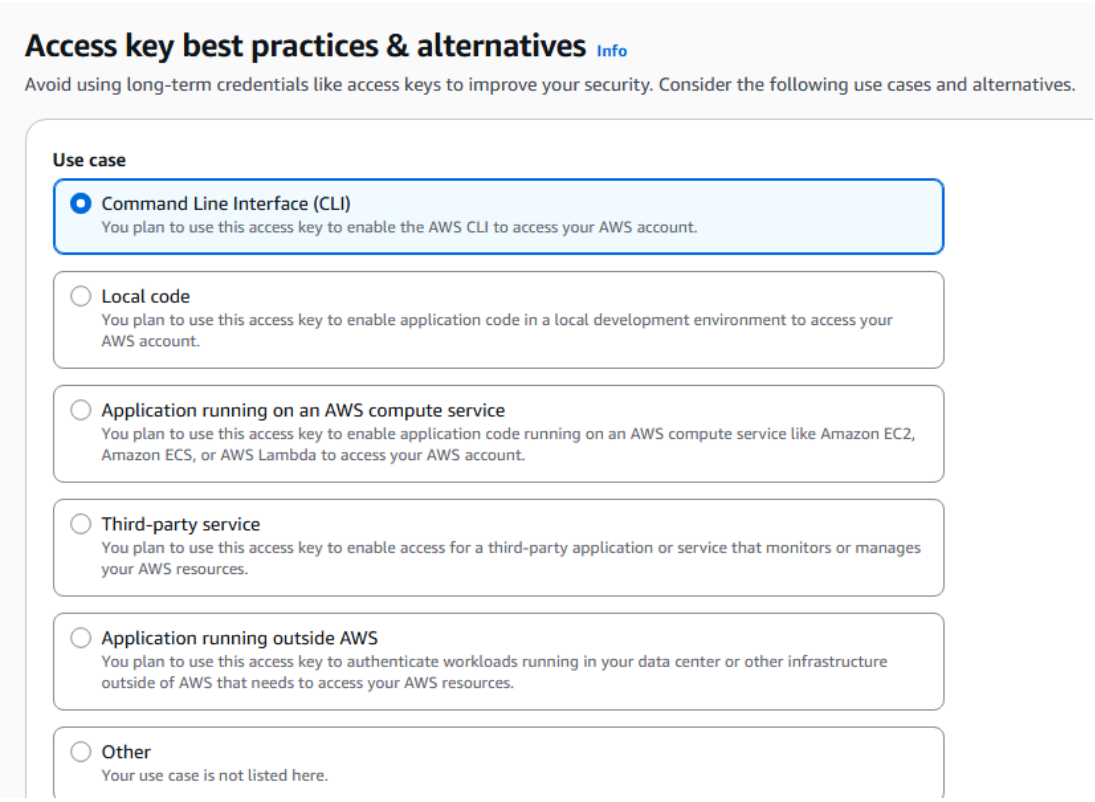
1. Login to AWS Console **(with your IAM user)**
2. Go to:
   - Services → **IAM**
   - Click **Users**
   - Click your username
   - Go to **"Security Credentials" tab**



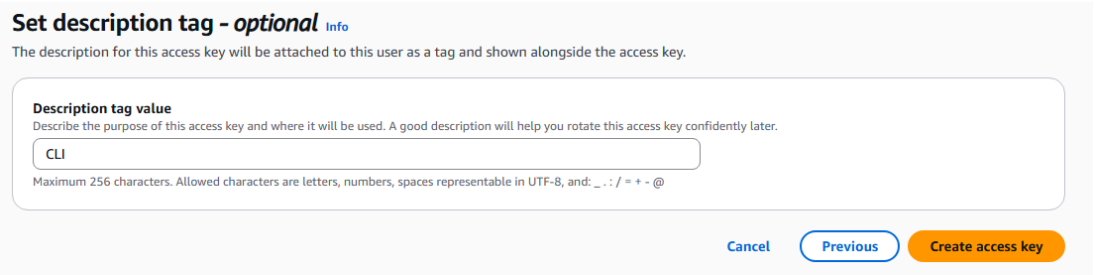
   - Under **Access Keys**, click **Create Access Key**



- Select CLI



Set description tag



- Download the **.csv file**

⚠️ These are **like your ATM card + PIN**. Don't share or hardcode them anywhere.

💡 **!!If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.**

Configure the CLI:

```
aws configure
```

You'll be asked for:

Prompt	What to Enter
AWS Access Key ID	Paste from your `.csv`
AWS Secret Access Key	Paste from your `.csv`
Default region	e.g., `us-east-1` or `ap-south-1` (for Mumbai)
Default output format	Use `json` (or `table` if you prefer)

Check:

```
C:\Users\Jeevan>aws iam list-users
{
    "Users": [
        {
            "Path": "/",
            "UserName": "Jeevan",
            "UserId": "AIDAX2QPHVNAINW37RPK4",
            "Arn": "arn:aws:iam::537976613696:user/Jeevan",
            "CreateDate": "2025-06-18T18:14:52+00:00",
            "PasswordLastUsed": "2025-06-19T11:59:56+00:00"
        }
    ]
}
```

🌰 Real-World Example

Task: Upload a folder (`C:\Photos`) to S3 bucket `my-travel-pics`.

GUI Method:

- Open S3 → Click bucket → Click "Upload" → Select files → Wait... 😴

AWS CLI Method:

```
aws s3 sync C:\Photos s3://my-travel-pics
```

Done in seconds! ⚡

🚫 Common Mistakes to Avoid

Mistake	Fix
❌ Using root user's access keys	🚫 NEVER generate access keys for root — always use an IAM user
❌ Sharing `.csv` file or hardcoding keys	Use environment variables or IAM roles instead
❌ Forgetting to install Python	Not needed for AWS CLI v2 — it bundles its own Python version
❌ Wrong region during `aws configure`	Use `ap-south-1` if you are in India (Mumbai region)

Simple Real-Life Examples (Python/Data Workflows with CLI):

Prerequisite: You need an S3 bucket. If you don't have one, you can create it with the CLI too!

Create an S3 Bucket (if you don't have one):

```
aws s3 mb s3://my-unique-data-science-bucket-2025 --region ap-south-1
```

- `mb` stands for "make bucket."
- **Important:** S3 bucket names must be globally unique across ALL of AWS. So, pick something truly unique!

1. **List S3 Buckets:**

   ```
   aws s3 ls
   ```

   - This will list all S3 buckets in your configured region.

2. **Upload a File to S3:**
   - Create a simple text file on your desktop, e.g., `my_data.txt`, with some content.
   - Open your terminal and navigate to your desktop: `cd C:\Users\YourUsername\Desktop`
   - Then, upload the file:

     ```
     aws s3 cp my_data.txt s3://my-unique-data-science-bucket-2025/data/
     ```

     - `cp` stands for "copy."
     - This copies `my_data.txt` from your local machine into a folder called `data` inside your S3 bucket.

3. **List Contents of an S3 Bucket/Folder:**Bash

```
aws s3 ls s3://my-unique-data-science-bucket-2025/data/
```

4. **Download a File from S3:**

```
aws s3 cp s3://my-unique-data-science-bucket-2025/data/my_data.txt downloaded_data.txt
```

   - This downloads `my_data.txt` from S3 and saves it as `downloaded_data.txt` in your current local directory.

5. **Using `boto3` with AWS CLI Configuration:**
   - The best part is that when you configure the AWS CLI, `boto3` (the Python SDK) automatically uses those same credentials and region!
   - Create a Python file (`s3_test.py`):

```
import boto3

s3_client = boto3.client('s3')
bucket_name = 'my-unique-data-science-bucket-2025' # Use your bucket name

try:
    response = s3_client.list_objects_v2(Bucket=bucket_name)
    if 'Contents' in response:
        print(f"Contents of {bucket_name}:")
        for obj in response['Contents']:
            print(f"- {obj['Key']}")
    else:
        print(f"Bucket {bucket_name} is empty or does not exist.")
except Exception as e:
    print(f"Error listing bucket: {e}")
```

   - Run it from your terminal: `python s3_test.py`
   - You'll see it list the `my_data.txt` file you uploaded, **without you putting any credentials in the Python code!** This is the power of shared configuration.