Executing Python Boto3 Scripts from Local Windows Machine

- We have an IAM User called boto3Admin for two AWS Accounts and also created programmatic access keys for these two AWS Accounts with respect to boto3Admin IAM User

- What Are AWS Programmatic Access Keys?
  - Programmatic Access Keys = Access Key ID + Secret Access Key
  - These are like a username and password that allow a script or application to interact with AWS services without logging in to the AWS Console
  - Finally, AWS Programmatic Access Keys are used to authenticate Python Boto3 scripts to perform actions on your AWS account

- There are different ways to provide these credentials to your Python Boto3 scripts from a local Windows machine, they are:
  - Using AWS CLI Configuration to pass AWS Programmatic Access Keys
  - Using a Custom AWS Programmatic Access Keys File
  - Using AWS Programmatic Access Keys in the Script
  - Using Environment Variables to pass AWS Programmatic Access Keys

Using AWS CLI Configuration to pass AWS Credentials

- We already installed aws cli v2 on Windows
- Configure Credentials using : aws configure --profile dev
- Note: This store credentials and config details in:
  - C:\Users\<YourUsername>\.aws\credentials
  - C:\Users\<YourUsername>\.aws\config

  - This is the standard and secure practice to execute or develop Python Boto3 Scripts from Local Machine
  - Easy to switch between different AWS profiles (e.g., dev, prod)

Using Custom Credentials File

- Here, we have to Create aws credentials.ini file instead of aws configure command as below

```
[dev]
aws_access_key_id = DEV_ACCESS_KEY
aws_secret_access_key = DEV_SECRET_KEY
region = us-east-1

[prod]
aws_access_key_id = PROD_ACCESS_KEY
aws_secret_access_key = PROD_SECRET_KEY
region = us-west-2
```

- Notes:
  - No dependency on AWS CLI
  - Easy to switch between different AWS profiles (e.g., dev, prod)

Using AWS Credentials Directly in the Script

- Its very straightforward – simply we will pass programmatic access keys directly in python boto3 script
- ⚠ Important Security Note:
  - Do NOT use this method in production environments.
  - If the script is accidentally pushed to GitHub or shared then someone could misuse your AWS account badly
  - 🛑 Credentials hardcoded in code are difficult to rotate and manage securely.

Using Environment Variables to Pass AWS Credentials

- We will Set environment variables like AWS_ACCESS_KEY_ID, AWS_SECRET_ACCESS_KEY and AWS_DEFAULT_REGION in your system or terminal session
- Setting Credentials as Environment Variables on Windows:
  - On PowerShell:
    - $env:AWS_ACCESS_KEY_ID = "your-access-key-id"
    - $env:AWS_SECRET_ACCESS_KEY = "your-secret-access-key"
    - $env:AWS_DEFAULT_REGION = "us-east-1“

  - On Command Prompt:
    - set AWS_ACCESS_KEY_ID=your-access-key-id
    - set AWS_SECRET_ACCESS_KEY=your-secret-access-key
    - set AWS_DEFAULT_REGION=us-east-1

- Notes:
  - Keeps credentials out of source code
  - Works well for temporary sessions or CI/CD pipelines
  - Safer than hardcoding credentials
  - ✕ Not persisted across terminal restarts (unless added to system variables)