**AWS Automation with Python Boto3 and Lambda Functions**

**[PART - 1] Introduction**

**Aim:** Learn how to automate AWS common tasks using boto3 and Lambda Functions.

**Objective of this course:**

1. Cover the core concepts of boto3 and Lambda.
2. Understand boto3 and Lambda concepts with real-time scenarios.
3. Running boto3 scripts on your local machine and triggering lambda functions.
4. By the end of this course you will get the knowledge to apply different concepts of boto3 and Lambda for different AWS Services.

**Pre-requisites:**

What do you need for this course?

1. AWS Account - It is great if you have a free tier account.
2. Good if you have basic Knowledge on AWS Services and Python (Not mandatory)
3. Knowledge on Any Python IDE (Not mandatory)

**Note:** All the videos will be uploaded to **“AWS Automation with Python Boto3”** playlist.

**[PART - 2] Introduction to Boto3**

**Introduction to Boto3**

* Boto3 is the name of the Python SDK/Library/Module/API for AWS.
* Boto3 allows us to directly create, update, and delete AWS services from our Python scripts.
* Boto3 is built on the top of the botocore module.

We have to Install boto3 to work with AWS Services using Python Scripts.

**How to install boto3?**

**Python-2.x:**  pip install boto3

**Python-3.x:**  pip3 install boto3

**Install Python and Boto3 on Windows Machine.**

* Python-3.7.4
* Go to www.python.org
* Set Paths for python and pip3
* Install boto3 using  pip3 install boto3

**Install Python and Boto3 on Linux Machine.**

yum install gcc openssl-devel bzip2-devel libffi-devel

cd /usr/src

wget https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tgz

tar xzf Python-3.7.4.tgz

cd Python-3.7.4

./configure --enable-optimizations

make altinstall

cd /usr/local/bin/

./python3.7 --version

./pip3.7 --version

pwd

ln -s /usr/local/bin/python3.7 /bin/python3

python3 --version

ln -s /usr/local/bin/pip3.7 /bin/pip3

pip3 --version

pip3 install boto3

**[PART 3] Boto3 Environment Setup**

* Configure credentials of your AWS account on using aws cli commands.
* **What is AWS CLI?**

The AWS Command Line Interface (AWS CLI) is **a unified tool to manage your AWS services**. With just one tool to download and configure, you can control multiple AWS services from the command line and automate them through scripts.

* **Downloading AWS CLI:** https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html/
* Login to AWS Management Console and create a new user with programmatic access and provide **AdministratorAccess.**
* Configure root/IAM user access-keys/credentials using:

aws configure (Works fine but creates DEFAULT profile)

OR use different profiles for different environments like this:

aws configure --profile dev

aws configure --profile qa

aws configure --profile prod

First Automation Script to list all the IAM Users in your Account.

* First let us do it manually and see in the AWS Management Console.

**Manual Steps:**

**Step1:** Get AWS Management Console (https://aws.amazon.com/console/)

**Step2:** Get IAM Console

       In IAM Console we have many options to select like,

**- Users**

       - Groups

       - Roles

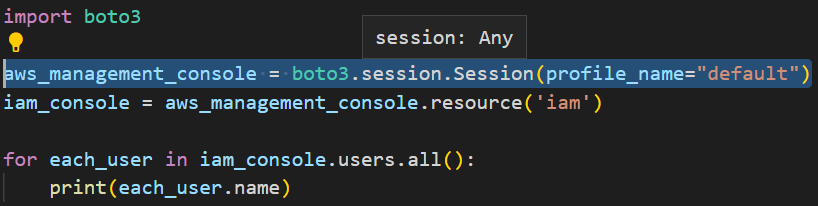
       - Policies etc...

* Link to the code: <https://github.com/yeshwanthlm/Boto3-Course-YouTube/tree/main/Project-1>

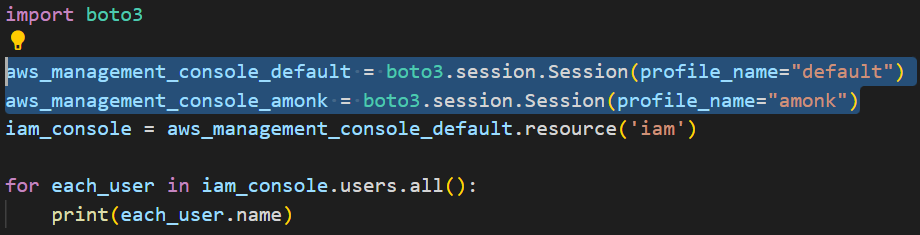
**[PART 4] Concepts of Boto3**

* **Session**
* **Resource**
* **Client**
* Meta
* Collections
* Waiters
* Paginators

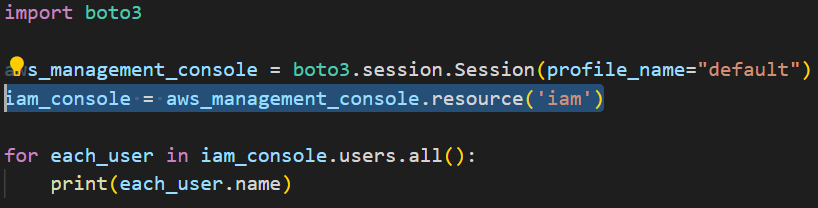
**Session:**



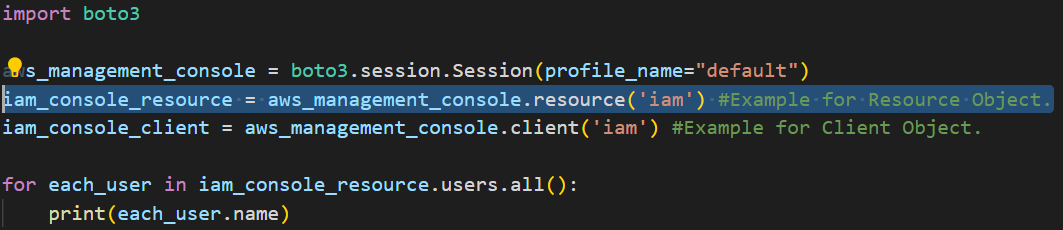
* In simple words, it is just the **AWS Management Console**.
* Store configuration information (Credentials of Default user etc…).
* Allows us to create Service, Clients and Resources.
* It creates a default session for us when we need it.
* We can create multiple sessions in the same script!



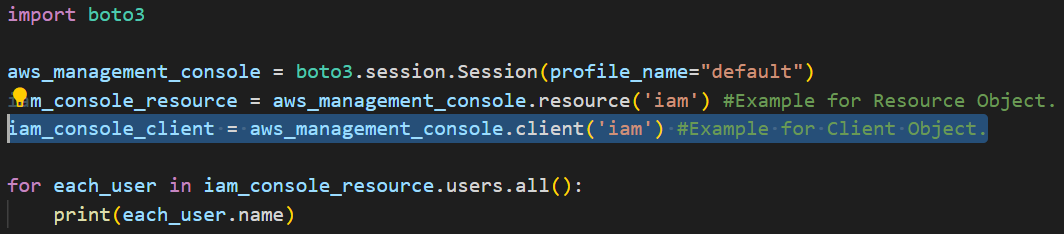
**Resource and Client:**

****

* We can create particular **AWS Service console** examples: IAM Console, EC2 Console etc…
* You can create an AWS Service console from your Session object.
* Region name can be specified after the Profile name.

**Example for Resource Object:**

**Example for Client Object:**

****

**Should I choose Resource or Client? 🤔**

* You can choose anyone depending on your use case.
* Resource is **Higher Level Object** oriented service access.
* Resource objects are only available for a few AWS Services.
* Let us check which AWS Service has a Resource Object!!!  - DEM😉
* **['cloudformation', 'cloudwatch', 'dynamodb', 'ec2', 'glacier', 'iam', 'opsworks', 's3', 'sns', 'sqs'] - Resource Object Available.**
* Client is **Low Level Service Access**.
* In simple terms, the output/response in case of Client will be in Dictionary, which needs more effort in implementing boto3 scripts.
* Whereas Resource is an object, we can use simple (.) operation.
* All operations that we see in AWS Management Console can be done in Client whereas Resource does not guarantee you that. Some operations may not be supported.
* If we do not have some operations in Resource we can enter into Client by using the **“Meta”** concept. Let us talk about this later! 😉
* Let us see how much effort is needed for both Resource and Client. - DEM😀

**[PART 5] Boto3 Script with Boto3 Documentation (Only Demo)**

**Example 1: List all the IAM users in AWS Account using client objects.**

**Example 2: List all the running EC2 Instances in your AWS Account using client objects.**

**Example 3: List all the IAM users in AWS Account using resource objects.**

['cloudformation', 'cloudwatch', 'dynamodb', 'ec2', 'glacier', 'iam', 'opsworks', 's3', 'sns', 'sqs'] - Resource Object Available.

**Example 4:** **List all the running EC2 Instances in your AWS Account using resource objects.**