## AWS CLI TOOL AND BOTO3

- Install `python3` using `yum`

```
sudo yum install python3
```

### Using Python pip

- `pip` is the package installer for Python. You can use pip to install packages from the `Python Package Index`.
- To check Python Version use `python3 --version`
- Using the `pip` command, install the AWS CLI and Boto3:

```
sudo pip3 install awscli boto3 -U
```

- We can confirm the packages are installed by checking the version of the AWS CLI tool and loading the boto3 library

```
aws --version
```

- To run a Python Command without going into Python Shell

```
$ python3 -c "import boto3"
$ sudo python3 -m pip install boto3
```

- If there is no error, package is present. If there is any error for above command, revisit the installation of boto3 package using `pip`

---

## Permissions to access the Resources in AWS Account

**EC2 Instance Execution**

- Go to `IAM > Create Role for EC2 service > Assign Service Specific Policies to this Role (e.g AmazonEC2FullAccess , AmazonS3FullAccess, AmazonRDSFullAccess)` > `Attach this Role to EC2 Instance`

> OR

**Local Machine Execution**

- Navigate to `IAM > Add User > Select Programmatic Access > Attach Service Specific Permissions to this User (e.g AmazonEC2FullAccess , AmazonS3FullAccess, AmazonRDSFullAccess)`
- Now that we have a user and credentials, we can finally configure the scripting environment with the AWS CLI tool
- Open any command line utility : For Windows : `Open Git Bash`

```
aws configure
```

- You'll be prompted for the AWS access key ID, AWS secret access key, default region name, and default output format.
- Enter the access key ID and secret access key from the user creation step
- For the default region name, enter a suitable region. The region you enter will determine the location where any resources created by your script will be located.
- Now that our environment is all configured, lets test with aws cli tool.

```
$ aws ec2 describe-instances
```

- If you already have instances running, output of above command will be the details of those instances. If not, you should see an empty response. If there are any errors, walk through the previous steps to see if anything was overlooked or entered incorrectly, particularly the access key ID and secret access key.

## Python Code Execution with AWS Services

**Python boto3 List EC2 Instances**

- First, we'll import the `boto3` library. Using the library, we'll create an ice client object. This is like a handle to the EC2 console that we can use in our script to print the instance ID and state.

> Note : Save the below code as <filename.py> and execute using `python filename.py`

```
#!/usr/bin/env python
import boto3
```

```
#Create a client object connection with ec2 service
ec2 = boto3.client('ec2')

# Execute a function call to describe instances present in aws account
ec2_dict=ec2.describe_instances()
print("ec2_dict type is",type(ec2_dict))
print("ec2_dict is",ec2_dict)
```

**Python boto3 List S3 Buckets Information**

```
import boto3
#Create a client object connection with ec2 service
s3 = boto3.client('s3')
# Execute a function call to get list of S3 buckets in aws account
bucket_dict=s3.list_buckets()
print("Type of bucket_dict is",type(bucket_dict))
bucket_list=bucket_dict['Buckets']
print("Type of bucket_list is",type(bucket_list))
for bucket_info in bucket_list:
    print("Type of bucket_info is",type(bucket_info))
    print("Bucket Name is ",bucket_info['Name'])
```