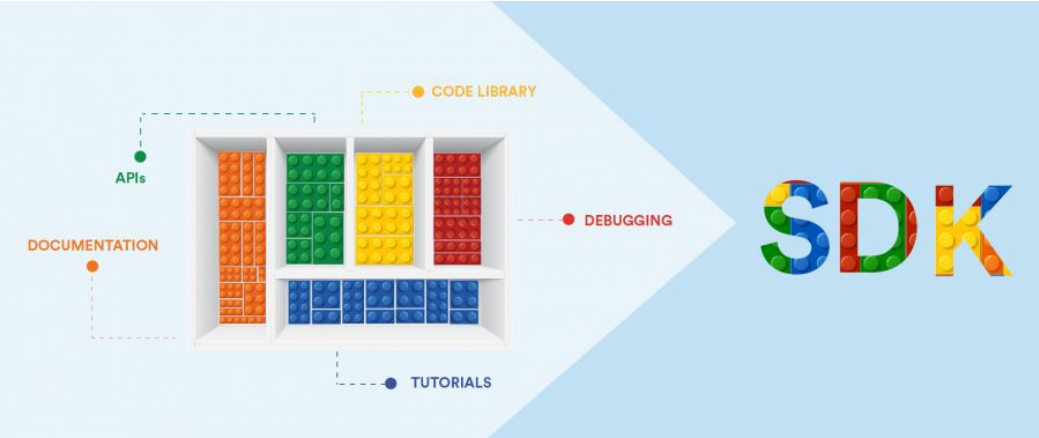
AWS Boto3

CLARUSWAY
WAY TO REINVENT YOURSELF

Boto3

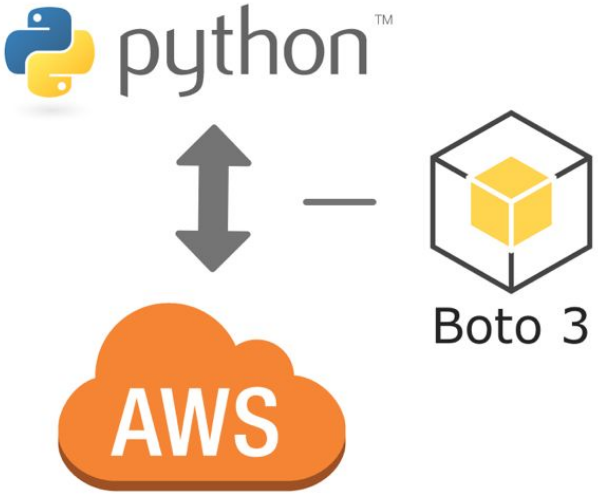
What is SDK?



SDK stands for software development kit or devkit for short. It's a set of software tools and programs used by developers to create applications for specific platforms. SDK tools will include a range of things, including libraries, documentation, code samples, processes, and guides that developers can use and integrate into their own apps. SDKs are designed to be used for specific platforms or programming languages.

Boto3

What is Boto3?



Boto3 is the **AWS SDK** (Software Development Kit) for **Python**. It enables you to create, update, and delete AWS resources with your **Python scripts**. It is basically a **Python library**.

Boto3

- Boto3 runs on top of botocore, which is foundation for AWS CLI
- A session initiates the connectivity to AWS services. A default session uses the default credential profile(e.g. ~/.aws/credentials, or assume your EC2 using IAM instance profile)

Default session

```
import boto3

# Using the default session
sqs = boto3.client('sqs')
s3 = boto3.resource('s3')
```

Custom session

```
import boto3
import boto3.session

# Create your own session
my_session = boto3.session.Session()
```

Boto3

- Client vs Resource
    - Client:
        - Low-level AWS service access
        - Exposes botocore client to the developer
        - Supports all AWS service operations

Example

```
s3 = boto3.client('s3')
```

Boto3

Example

```
import boto3


    client = boto3.client('s3')
    response = client.list_objects(Bucket='example')
    for content in response['cont']:
        obj_dict = client.get_object(Bucket='example', Key=cont['Key'])
        print(cont['Key'], obj_dict['LastModified'])
```

Boto3

- Resource:

  - Higher-level, object oriented API
  - Exposes sub processes of AWS resources
  - Does not provide 100% coverage of AWS API

  Example

```
s3 = boto3.resource('s3')
```

Boto3

Example

```
import boto3


    s3 = boto3.resource('s3')
    bucket = s3.Bucket('example')
    for obj in bucket.objects.all():
        print(obj.key, obj.last_modified)
```

Boto3

Clients vs Resources

To summarize, resources are higher-level abstractions of AWS services compared to clients. Resources are the recommended pattern to use boto3 as you don’t have to worry about a lot of the underlying details when interacting with AWS services. As a result, code written with Resources tends to be simpler.

However, Resources aren’t available for all AWS services. In such cases, there is no other choice but to use a Client instead.

Boto3

Boto3 Labs

Lab1:

- Install boto3 on Amazon Linux 2 ec2
- Configure boto3 access
- Test AWS access by listing S3 buckets, creating s3 bucket and pushing text file into s3

Lab2:

- Launch ec2 instance
- Resize an instance
- Stop/Start/Terminate an instance

Boto3

Boto3 Labs

Amazon Linux 2 - Install Python and boto3

```
yum install python3
curl -O https://bootstrap.pypa.io/get-pip.py
python3 get-pip.py
pip3 install boto3
```

Boto3

Using Boto3 List S3 Buckets

```
import boto3

# Use Amazon S3
s3 = boto3.resource('s3')

# Print out all bucket names
for bucket in s3.buckets.all():
    print(bucket.name)
```

Boto3

Using Boto3 Create & List S3 Buckets

```
import boto3

# Use Amazon S3
s3 = boto3.resource('s3')

# Create a new bucket
s3.create_bucket(Bucket='xxxxxxx-boto3-bucket')

# Print out all bucket names
for bucket in s3.buckets.all():
    print(bucket.name)
```

Boto3

Using Boto3 Upload Files To S3 Buckets

```
import boto3

# Use Amazon S3
s3 = boto3.resource('s3')

# Upload a new file
data = open('test.txt', 'rb')
s3.Bucket('xxxxxxxx-boto3-bucket').put_object(Key='test.txt', Body=data)
```

Boto3

Instance codes

0 : pending
16 : running
32 : shutting-down
48 : terminated
64 : stopping
80 : stopped

Boto3

Launch instances

```
import boto3
ec2 = boto3.resource('ec2')

# create a new EC2 instance
instances = ec2.create_instances(
    ImageId='ami-xxxxxxxxxxxxx', <=update instance id
    MinCount=1,
    MaxCount=1,
    InstanceType='t2.micro',
    KeyName='your keypair without .pem'
 )
```

Boto3

Stop instance

```
import boto3
ec2 = boto3.resource('ec2')
ec2.Instance('your InstanceID').stop()
```

Boto3

Start instance

```
import boto3
ec2 = boto3.resource('ec2')
ec2.Instance('your InstanceID').start()
```

Boto3

Terminate instance

```
import boto3
ec2 = boto3.resource('ec2')
ec2.Instance('your InstanceID').terminate()
```

THANKS!

Any questions?

You can find me at:

- @sumod
- sumod@clarusway.com