Setup

To use boto3, you must first install python. There are a number of distributions available; among the most popular are Anaconda3, ActivePython, and PyPy.

I prefer Anaconda3 because of the large number of pre-installed packages. This does however also mean it is one of the largest distributions. But as the saying goes, "It is better to have and not need than to need and not have."

Once you have installed your Python of choice, use the **pip** installer to install boto3.

```
pip install boto3
```

Once boto3 is installed, install the Amazon AWS CLI tools and run **aws configure** to set your credentials and default region.

```
aws configure
```

Now you are ready to roll.

Services

boto3 includes access to almost all of the AWS services. To interact with these services, you create a resource or client object that connects to a particular service. Then you can use the service using boto3's api for that object.

For instance, to create a new EC2 instance, you will create a resource object that is connected to 'ec2.' Then with that object, you will call a function to create the instance and pass the appropriate parameters to the function.

Querying EC2

Connect to EC2

To create an EC2 resource object

```
import boto3
ec2 = boto3.r es our ce( 'ec2')
instances = ec2.in sta nce s.all
()
```

Print raw list of instances

```
for instance in instances:
    print( instance )
```

Print list of instances by name tag

```
for instance in instances:
```

Querying EC2 (cont)

```
    print( [tag[' Value'] for tag in instance.tags if tag.get('Key')
```

Print list of instances by state

```
states = (
   { 'Code' : 0, 'state' : 'pending' },
   { 'Code' : 16, 'state' : 'running' },
   { 'Code' : 32, 'state' : 'shutt ing down' },
   { 'Code' : 48, 'state' : 'termi nated' },
   { 'Code' : 64, 'state' : 'stopping' },
   { 'Code' : 80, 'state' : 'stopped' },
)
instan ce_ states = {
    'pe nding': [],
    'ru nning': [],
    'sh utt ing  down' : [],
    'te rmi nated' : [],
    'st opping' : [],
    'st opped' : [],
}
for instance in instances:
    ins tan ce_ nam e=[ tag ['V alue'] for tag in instance.tags if[tag
 == 'Name']
    ins tan ce_ sta tes [in sta nce.st ate ['N ame']].a ppe nd(ins tan
for i in instan ce_ states:
    ins tan ce_ sta tes [i].sort()
    print(i + ' instan ces \n--')
    for name in instan ce_ sta tes[i]: print( name
    pri nt('')
```

Creating Instances

Gather instance details

```
# ID of the AMI (Amazon Machine Image) that the VM will use for OS
`# This is the ami for Window Server 2012 R2 64-bit
ami_id = ami-3d 787d57
# IDs of the Security Groups to be assign to the VM
securi ty_ gro up_ids = []
# Single subnet id in which VM will be started
subnet_id =
```

Creating Instances (cont)

```
# Key for the host that will be used (Key de
keyfil e_name =
# ARN of the IAM instance profile
iam_pr ofi le_arn = 'arn:a ws: iam
file l/m ypr ofile
# VM instance type / instance size
instan ce_type = 't2.micro'
# Disk drive capacity in GB
disk_size = 120
# UserData. Code block to execute
user_data = ' '
```

Create the instance

```
ec2.cr eat e_i nst ances(
    ImageId = ami_id,
    Min Count = 1,
    Max Count = 1,
    KeyName = keyfil e_name,
    Sec uri tyG roupIds = securi t
    Ins tan ceType = instan ce_ ty
    Blo ckD evi ceM appings =[
        {
            'De vic eName': '/d ev/sda
            'Ebs': {
                'Vo lum eSize': disk_s
                'De let eOn Ter min at
                'Vo lum eType': 'gp2',
            }
        }
    ],
    Iam Ins tan ceP rofile = {
        'Arn': arn_pr ofile
    }
    Sub netId = subnet_id,
    Use rData = user_data
)
```