## 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.resource('ec2')
instances = ec2.instances.all()
```

**# Print raw list of instances**

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

**# Print list of instances by name tag**

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

**# Print list of instances by state**

```
states = (
  { 'Code' : 0, 'state' : 'pending' },
  { 'Code' : 16, 'state' : 'running' },
  { 'Code' : 32, 'state' : 'shutting-down' },
  { 'Code' : 48, 'state' : 'terminated' },
  { 'Code' : 64, 'state' : 'stopping' },
  { 'Code' : 80, 'state' : 'stopped' },
)
instance_states = {
  'pending': [],
  'running': [],
  'shuttingdown' : [],
  'terminated' : [],
  'stopping' : [],
  'stopped' : [],
}
for instance in instances:
  instance_name=[tag['Value'] for tag in instance.tags if tag.get('Key') == 'Name']
  instance_states[instance.state['Name']].append( instance_name[0] )
for i in instance_states:
  instance_states[i].sort()
  print(i + ' instances\n--')
  for name in instance_states[i]:
print( name )
  print('')
```

## 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-3d787d57
# IDs of the Security Groups to be assign to the VM
security_group_ids = []
# Single subnet id in which VM will be started
subnet_id =
# Key file that will be used to decrypt the administrator password
keyfile_name =
# ARN of the IAM instance profile to be assigned to the VM
iam_profile_arn = 'arn:aws:iam::123456789012:instance-profile1/myprofile
# VM instance type / instance size
instance_type = 't2.micro'
# Disk drive capacity in GB
disk_size = 120
# UserData. Code block to execute on first instance launch
user_data = ' '
```

**# Create the instance**

```
ec2.create_instances(
  ImageId = ami_id,
  MinCount = 1,
  MaxCount = 1,
  KeyName = keyfile_name,
  SecurityGroupIds = security_group_ids,
  InstanceType = instance_type,
  BlockDeviceMappings = [
    {
    'DeviceName': '/dev/sda1',
    'Ebs': {
```

By **wtranmer**
cheatography.com/wtranmer/

Not published yet.
Last updated 18th April, 2016.

## Creating Instances (cont)

```
      'VolumeSize': disk_size,
      'DeleteOnTermination': True,
      'VolumeType': 'gp2',
    }
  ],
  IamInstanceProfile = {
    'Arn': arn_profile
  },
  SubnetId = subnet_id,
  UserData = user_data
)
```

By **wtranmer**
cheatography.com/wtranmer/

Not published yet.
Last updated 18th April, 2016.