

## S3

## Basic Modules to import

```
from boto.s3.connection import S3Connection
from boto.s3.key import Key
import boto
```

## Basic Operations

## Connecting:

```
c = S3Connection('<AWS_KEY_ID>', '<AWS_SECRET_KEY>' [,region,...])
c = boto.connect_s3()
```

## Creating a bucket:

```
c.create_bucket('<bucket-name>')
```

## Getting a bucket:

```
b = c.get_bucket('<bucket-name>')
```

## Deleting a bucket:

```
c.delete_bucket(b)
```

## Getting a bucket object:

```
k = Key(b)
k.key = 'object-name'
```

## Downloading said object to file:

```
k.get_contents_to_filename('<filename>')
```

## Downloading string data:

```
k.get_contents_as_string()
```

## Creating a new object key:

```
k = b.new_key('<key-name>')
```

## Uploading file to bucket:

```
k.key = 'object-name'
k.set_contents_from_filename(<path_to_file>)
```

## Uploading from string:

```
k.set_contents_from_string('<string>')
```

## Other Operations:

## Setting Access Controls (bucket-wide):

```
b.set_acl('public-read')
# or any of 'private', 'public-read-write', 'authenticated-read'
```

## Object-specific Access Control:

```
b.set_acl('private', 'confidential.txt')
```

## Setting Object Metadata:

```
k.set_metadata('meta1', 'This is the first metadata value')
k.set_metadata('meta2', 'This is the second metadata value')
```

## Getting Object Metadata:

```
k.get_metadata('meta1')
'This is the first metadata value'
```

## SQS

## Basic Imports:

```
from boto.sqs.connection import SQSConnection
from boto.sqs.message import Message
import boto
```

## Basic Operations:

## Connecting:

```
c = SQSConnection(AWS_KEY_ID, AWS_SECRET_KEY[, region])
c = boto.connect_sqs()
```

## Creating a queue:

```
q = c.create_queue('<que_name>'[,visibility_timeout])
```

## Listing all queues in region:

```
qs = c.get_all_queues([prefix='<prefix>'])
```

## Getting a specific queue:

```
q = c.get_queue('<queue_name>')
```

## Writing messages:

```
m = Message()
m.set_body('<body_text>')
res = q.write(m)
```

## Reading Messages:

```
rs = q.get_messages([num_messages,...])
mbody = rs[0].get_body()
```

## Deleting Messages:

```
q.delete_message(m)
```

## Emptying a queue:

```
q.clear() #use carefully
```

## Deleting (Empty) queues:

```
c.delete_queue(q)
```

# boto Cheat Sheet

## EC2

### Basic Imports:

```
from boto.ec2.connection import EC2Connection
import boto
```

### Basic Operations:

#### Connecting:

```
c = EC2Connection('<AWS_KEY_ID>', '<AWS_SECRET_KEY>',[, region])
c = boto.connect_ec2()
```

#### Getting all reservations within a region:

```
rsv = c.get_all_instances([instance_ids,...])
```

#### Get all instances within reservations:

```
for r in rsv:
```

```
    ins = r.instances
```

#### Get specific instance (with known id):

```
ins = c.get_all_instances(instance_ids=['<instance_id>'])[0]
```

#### Launching Instances:

```
c.run_instances('<ami-image-id>',[key_name, instance_type, ...])
```

#### Stopping Instances:

```
c.stop_instances([instance_ids, force])
```

#### Terminating Instances:

```
c.terminate_instances([instance_ids])
```

### Instance Operations:

#### Starting an instance:

```
ins.start()
```

#### Stopping an instance:

```
ins.stop()
```

#### Rebooting an instance:

```
ins.reboot()
```

#### Terminating an instance:

```
ins.terminate()
```

#### Getting instance attributes:

```
ins.get_attribute('<attribute>')['<attribute>']
```

#### Setting instance attributes:

```
ins.modify_attribute('<attr_name>', <attr_value>)
# Valid attribute names: instanceType|kernel|ramdisk|userData|
# disableApiTermination|instanceInitiatedShutdownBehavior|
# rootDeviceName|blockDeviceMapping|sourceDestCheck
```

## DynamoDB

### Basic Imports:

```
import boto
```

### Basic Operations:

#### Connecting:

```
c = boto.connect_dynamodb('<YOUR_AWS_KEY_ID>', '<YOUR_AWS_SECRET_KEY>'
    [, region, ...])
c = boto.connect_dynamodb()
```

#### Creating table schemata:

```
sch = c.create_schema('<hash_key_name>', '<hash_key_proto_value>'
    [, '<range_key_name>', '<range_key_proto_value>'])
```

#### Creating a table:

```
t = c.create_table('<name>', <schema>, <read_units>, <write_units>)
```

#### Listing all tables in region:

```
l = c.list_tables()
```

#### Getting a specific table:

```
t = c.get_table('<table_name>')
```

#### Describing tables:

```
c.describe_table('<table_name>')
```

#### Deleting tables:

```
t.delete() #nukes table and items within, use carefully.
```

### Item Operations:

#### Adding items:

```
data = {'field_name': <value>, 'another_field': 'another value'}
```

```
item = t.new_item(hash_key_name=<value>, attrs=<data>
```

```
    [, range_key=<value>])
```

```
item.put() #item is not committed until this is executed
```

#### Retrieving items:

```
it = t.get_item(hash_key=<value> [, range_key=<value>])
```

#### Updating items:

```
it['field_name'] = <new_value>
```

```
it.put()
```

#### Deleting items:

```
it.delete()
```