```
Configuring SNS for CloudWatch Alarms
>>> import boto
>>> sns = boto.connect_sns()
>>> sns.create_topic('paws_cloudwatch')
{u'CreateTopicResponse': {u'ResponseMetadata': {u'RequestId':
u'73721b87da0e-11e0-99a4-59769425d805'},

u'CreateTopicResult': {u'TopicArn':
u'arn:aws:sns:useast-1:084307701560:paws_cloudwatch'}}}
>>>
topic_arn = _['CreateTopicResponse']['CreateTopicResult']['TopicArn']
>>>
topic_arn
u'arn:aws:sns:us-east-1:084307701560:paws_cloudwatch'
>>>
sns.subscribe(topic_arn, 'email', 'suemail@email.org')
{u'SubscribeResponse':
{u'SubscribeResult': {u'SubscriptionArn': u'pending confirmation'},

u'ResponseMetadata': {u'RequestId': u'd4a846fd-da0e-11e0-bcf1-37db33647dea'}}}
>>>

 Creating a CloudWatch Alarm
>>> import boto
>>> cw = boto.connect_cloudwatch()
>>> my_metrics = cw.list_metrics(dimensions={'InstanceId':'i-76894c16'})
>>> my_metrics
[Metric:DiskReadOps, Metric:NetworkOut, Metric:NetworkIn, Metric:DiskReadBytes,
Metric:CPUUtilization, Metric:DiskWriteBytes, Metric:DiskWriteOps]
>>> metric = my_metrics[4]
>>> metric
Metric:CPUUtilization
>>> alarm = metric.create_alarm(name='CPUAlarm', comparison='>', threshold=80.0,
period=60,
evaluation_periods=2, statistic='Average',
alarm_actions=['arn:aws:sns:us-east-1:084307701560:paws_cloudwatch'],
ok_actions=['arn:aws:sns:us-east-1:084307701560:paws_cloudwatch'])
>>> alarm
MetricAlarm:CPUAlarm[CPUUtilization(Average) GreaterThanThreshold 80.0]
>>> alarm.set_state('ALARM', 'Testing my alarm', '100')
True
>>> alarm.describe_history()


Easy Email Notifications
import os
import boto
def easy_alarm(instance_id,
               alarm_name,
               email_addresses,
               metric_name,
               comparison,
               threshold,
               period,
               eval_periods,
               statistics):
    """
    Create a CloudWatch alarm for a given instance.  You can choose
    the metric and dimension you want to associate with the alarm
    and you can provide a list of email addresses that will be
    notified when the alarm fires.
    instance_id     The unique identifier of the instance you wish to
                    monitoring.
    alarm_name      A short but meaningful name for your alarm.
    email_addresses A list of email addresses that you want to
                    have notified when the alarm fires.
    metric_name     The name of the Metric you want to be notified
                    about.  Valid values are:
                    DiskReadBytes|DiskWriteBytes|
                    DiskReadOps|DiskWriteOps|
                    NetworkIn|NetworkOut|
                    CPUUtilization
    comparison      The comparison operator.  Valid values are:
```

```
                    >= | > | < | <=
    threshold       The threshold value that the metric will
                    be compared against.
        period          The granularity of the returned data.
                    Minimum value is 60 (seconds) and valid values
                    must be multiples of 60.
    eval_periods    The number of periods over which the alarm
                    must be measured before triggering notification.
    statistics      The statistic to apply.  Valid values are:
                    SampleCount | Average | Sum | Minimum | Maximum
    """
    # Create a connection to the required services
    ec2 = boto.connect_ec2()
    sns = boto.connect_sns()
    cw = boto.connect_cloudwatch()
    # Make sure the instance in question exists and
    # is being monitored with CloudWatch.
    rs = ec2.get_all_instances(filters={'instance-id', instance_id})
    if len(rs) != 1:
        raise ValueError('Unable to find instance: %s' % instance_id)
    instance = rs[0].instances[0]
    instance.monitor()

    # Create the SNS Topic
    topic_name = 'CWAlarm-%s' % alarm_name
    print 'Creating SNS topic: %s' % topic_name
    response = sns.create_topic(topic_name)
    topic_arn = response['CreateTopicResponse']['CreateTopicResult']['TopicArn']
    print 'Topic ARN: %s' % topic_arn
    # Subscribe the email addresses to SNS Topic
    for addr in email_addresses:
        print 'Subscribing %s to Topic %s' % (addr, topic_arn)
        sns.subscribe(topic_arn, 'email', addr)
    # Now find the Metric we want to be notified about
    metric = cw.list_metrics(dimensions={'InstanceId':instance_id},
                             metric_name=metric_name)[0]
    print 'Found: %s' % metric
    # Now create Alarm for the metric
    print 'Creating alarm'
    alarm = metric.create_alarm(name=alarm_name, comparison=comparison,
                                threshold=threshold, period=period,
                                evaluationn_periods=eval_periods,
                                statistics=statistics,
                                alarm_actions=[topic_arn],
                                ok_actions=[topic_arn])



    A Simple User-Data Script
>>> from ec2_launch_instance import launch_instance
>>> script = """#!/bin/sh
... echo "Hello World.  The time is now $(date -R)!" | tee /root/output.txt
... """
>>> instance, cmdshell = launch_instance(user_data=script)
Security Group: paws already authorized
waiting for instance
.
.
.
.
done
SSH Connection refused, will retry in 5 seconds
>>> cmdshell.shell()
       __|  __|_  )
       _|  (     /   Amazon Linux AMI
      ___|\___|___|
See /usr/share/doc/system-release/ for latest release notes.
No packages needed for security; 1 packages available
[ec2-user@domU-12-31-39-00-E4-53 ~]# sudo su
[ec2-user@domU-12-31-39-00-E4-53 ~]# cd /root
[ec2-user@domU-12-31-39-00-E4-53 ~]# ls
output.txt
[ec2-user@domU-12-31-39-00-E4-53 ~]# cat output.txt
```

```
Hello World.  The time is now Wed, 21 Sep 2011 23:53:51 +0000!
[ec2-user@domU-12-31-39-00-E4-53 ~]# exit
[ec2-user@domU-12-31-39-00-E4-53 ~]$ logout
*** EOF
>>> instance.terminate()
>>>

 Creating a Restricted User with IAM
import boto
# Create a restricted user using IAM
# The following JSON policy was generated using the
# AWS Policy Generator app.
# http://awspolicygen.s3.amazonaws.com/policygen.html
policy_json = """{
  "Statement": [
      "Sid": "Stmt1316576423630",
      "Action": [
        "cloudwatch:PutMetricData"
      ],
      "Effect": "Allow",
      "Resource": "*"
      }
  ]
}"""
def create_restricted_user(user_name):
    """
    Create a new user in this account.  The user will be
    restricted by the JSON policy document above.
    This function returns a tuple containing the access key
    and secret key for the new account.
    user_name The name of the new user.
    """
    iam = boto.connect_iam()
    user = iam.create_user(user_name)
    keys = iam.create_access_key(user_name)
    response = iam.put_user_policy(user_name,
                                   'CloudWatchPutMetricData',
                                   policy_json)
    fp = open('boto.cfg', 'w')
    fp.write('[Credentials]\n')
    fp.write('aws_access_key_id = %s\n' % keys.access_key_id)
    fp.write('aws_secret_access_key = %s\n' % keys.secret_access_key)
    fp.close()


Custom Metric for Disk Usage

import boto
import time
import datetime
import os
def main():
    cw = boto.connect_cloudwatch()
    md = boto.utils.get_instance_metadata()
    now = datetime.datetime.utcnow()
    stats = os.statvfs('/')

        total = float(stats.f_blocks * stats.f_bsize)
    available = float(stats.f_bavail * stats.f_bsize)
    percent_used = int(100 - 100 * (available / total))
    print 'metric_disk_usage: %d' % percent_used
    cw.put_metric_data(namespace='PAWS',
                       name='DiskUsage',
                       value=percent_used,
                       timestamp=now,
                       unit='Percent',
                       dimensions=[{'InstanceId' : md['instance-id']}])
if __name__ == "__main__":
    main()
```