CS553 Programming Assignment#3

CloudKon clone with Amazon EC2, S3, SQS and DynamoDB

1)Introduction

A distributed task execution framework similar to CloudKon was implemented on Amazon EC2 using SQS and DynamoDB. The framework has two components. One client to push tasks into the SQS and one worker to retrieve works from the SQS and execute them. The Assignment was done using Python

2) Design

2.1) Client

The client was implemented in python and the tool can be run using the command

Python client.py <Instruction Queue> <Work_Load_File>

The Work_Load_File is generated by a command line tool

Python gnrtr.py <number of instructions> <instruction>

So if the tool is run as python gnrtr.py 10 sleep 10000 will generate a file wrkr with 1o sleep 10000 instructions

A unique id is generated in the client program and this id is send to the SQS by the client

The client generates unique id for a session using a counter. While putting the instruction into the SQS the id is appended to the message

2.2) Local Back End Worker

The Local Back End Worker is run using the command

Python local.py <number of threads>

The tool creates a pool of specified number of threads. The tool reads the instructions from the work load file and is put into a queue. The threads take these instructions from this queue, executes them and puts the status in the Response queue. In the end the status of the Response queue is checked

2.3) Remote Back End Worker

The worker tool is run using the command

Python wrkr.py <Instruction Queue> <DynamoDB> <Response Queue>

The worker takes the tasks out of the Instruction Queue, executes them, updates them in DynamoDB and Response Queue. The message is cut into two pieces. The first piece has the id and this is checked with the ids available in dynamodb. If it is not there then the instruction is executed deleted from instruction queue and then updated to dynamo db and response queue

Then we have a tool to check the response Queue which is run using the command

Python chk.py <Response Queue>

This will check the content of the Response Queue and check the status of the completed instructions

2.4) Duplicate Tasks:

Dynamo DB was used to avoid duplicate tasks. The worker after fetching values from the Instruction Queue checks whether the task is there in Dynamo DB. If it is there that means the instruction has already been executed. If it is a new task the task is executed and updated in response queue and DynamoDB

2.5)Animoto Clone

The animoto generator generates links into a file wrkr. The animoto client takes these values and is populated into the SQS. The worker takes these links from the SQS and then downloads these images. After downloading ffmpeg is used to make a video with this images

2)Manual

Step 1: Starting Instances, SQS and DynamoDB

This can be done from the local

Python strt_inst.py <number of Instances> <name of Instruction SQS> <name of Response SQS> <name of DynamoDB>

This tool will start the specified number of instances along with 2 SQS and 1 DynamoDB with the name specified

There is another tool to delete the instances and Queues

It is run as follows

Python terminate.py

This will terminate all the running instances and queues in the account. This is run just in case something goes wrong and u need to start making instances and queues again

17 instances, 1DynamoDB and 2 SQS are made using this tool

All the public DNS of instances are updated in the file worker

Step 2: Transferring Files

The following files has to be transferred to the client using scp

client.py

wrkr.py

gnrtr.py

local.py

sender

worker

chk.py

animoto.py

animoto_client.py

animoto_wrkr.py

Step 3: Transferring files to the workers

Password less ssh is set up on every instance

Python-boto is installed on every instance

Then the script sender is used to push wrkr.py and animoto_wrkr.py to all the workers

Step 4: Generating instructions

The Work_Load_File is generated by a command line tool

Python gnrtr.py <number of instructions> <instruction>

So if the tool is run as python gnrtr.py 10 sleep 10000 will generate a file wrkr with 1o sleep 10000 instructions

Step 5: Running the client

The client is run using the command

Python client.py <Instruction Queue> <Work_Load_File>

The client will take values from the work load file, generates the ids , combines the id and the message and pushes them to the Instruction Queue

Step 6: Running the Remote Workers

All the workers in the file worker is run using pssh at the same time

pssh -h wrkr -t 100000000 -l ubuntu -A -i "python wrkr.py <Instruction Queue> <DynamoDB> <Response Queue>

The worker is run using the command

Python wrkr.py <Instruction Queue> <DynamoDB> <Response Queue>

The workers will run till the instruction Queue is empty

The Response Queue is checked using the tool check.py

It is run using the following command

Python chk.py <Response Queue >

This will check the number of success messages in the given response queue

Step 7: Running the Local Worker

The worker is run using the following command

Python local.py <no of threads>

This will execute the number of instructions in the Work_Load_File and execute them using the number of threads specified

Step 8: Checking the Response Queue

The response queue is checked using the command

Python chk.py <Response Queue>

This will check the number of success messages in the response queue and this number is displayed

Step 9: Generating Links

Animoto.py is run as follows

Python animoto.py <n>

Where n is the number of workers

Step 10: Running the Animoto client

Animoto client is run as follows

Python animoto_client.py <instruction queue><Work load file>

This will make the client load the instructions form the work load file to the instruction queue

Step 11: Running the Animoto Worker

All the workers in the file worker is run using pssh at the same time

pssh -h wrkr -t 100000000 -l ubuntu -A -i "python animoto_wrkr.py <Instruction Queue> <DynamoDB> <Response Queue>

The worker is run using the command

Python animoto_wrkr.py <Instruction Queue> <DynamoDB> <Response Queue>

The workers will run till the instruction Queue is empty

3) Performance Evaluation

3.1) Efficiency

Remote:

Table showing the time taken

workers/Instruction	**10s x 10**	**1s x 100**	**10ms x 1000**
1	100.7198	103.5925	40.60159
2	100.7125	105.2598	40.31233
4	100.6499	105.1859	44.01281
8	100.8699	117.2482	64.98935
16	100.6964	104.3814	41.19066

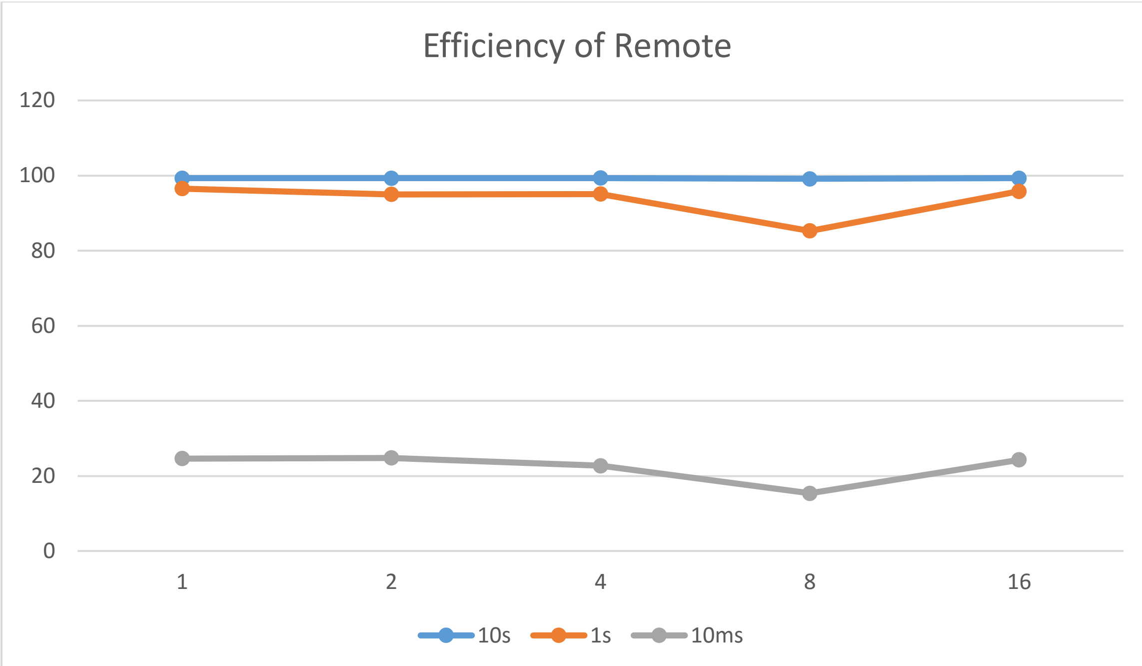
Table showing idle time taken

workers/Instruction	**10s x 10**	**1s x 100**	**10ms x 1000**
1	100	100	10
2	100	100	10
4	100	100	10
8	100	100	10
16	100	100	10

The efficiency is calculated as Idle time * 100/Time taken

Table showing idle time taken

workers/Instruction	**10s x 10**	**1s x 100**	**10ms x 1000**
1	99.28531	96.53208	24.62958
2	99.29259	95.00303	24.8063
4	99.35434	95.06974	22.72066
8	99.13758	85.28919	15.38714
16	99.30842	95.80252	24.27735



It can be observed that the Efficiency remains almost constant and we can conclude that the Cloudkon is scalable

It can also be observed that the efficiency is very less for 10ms instructions. This could be because of the high number of instructions being fetched and written into the db and SQS. So it can be concluded that as the number of instructions increases the efficiency decreases but is highly scalable

Local:

Table showing the time taken

Threads/Instructions	**10s x 10**	**1s x 100**	**10ms x 1000**
1	100.104	100.1499	10.357
2	200.2079	200.298	20.722
4	400.4077	400.6006	41.469
8	800.829	801.1957	82.8923
16	1601.652	1602.389	165.774

Table showing idle time taken

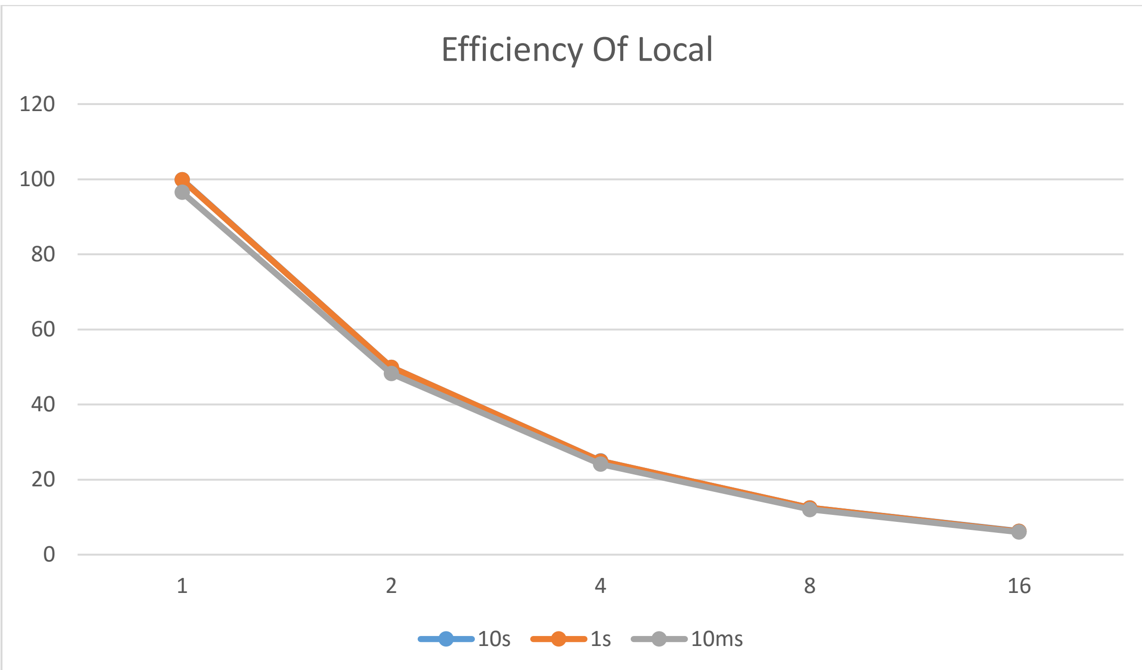
Threads/Instructions	**10s x 10**	**1s x 100**	**10ms x 1000**
1	100	100	10
2	100	100	10
4	100	100	10
8	100	100	10

16	100	100	10

The efficiency is calculated as Idle time * 100/Time taken

Table showing idle time taken

Threads/Instructions	**10s x 10**	**1s x 100**	**10ms x 1000**
1	99.89611	99.85032	96.55306
2	49.94809	49.92562	48.25789
4	24.97454	24.96252	24.1144
8	12.48706	12.48135	12.06385
16	6.243553	6.240682	6.032309



It can be noted that the efficiency in local is not dependent on the number of instructions submitted

And it can also be noted that the efficiency decreases as the number of threads increases. This is because of the reason that this experiment was done a t2.micro system that has only one cpu and one core. So using threads will slow down the system instead if increasing the efficiency

3.2) Throughput

Remote:

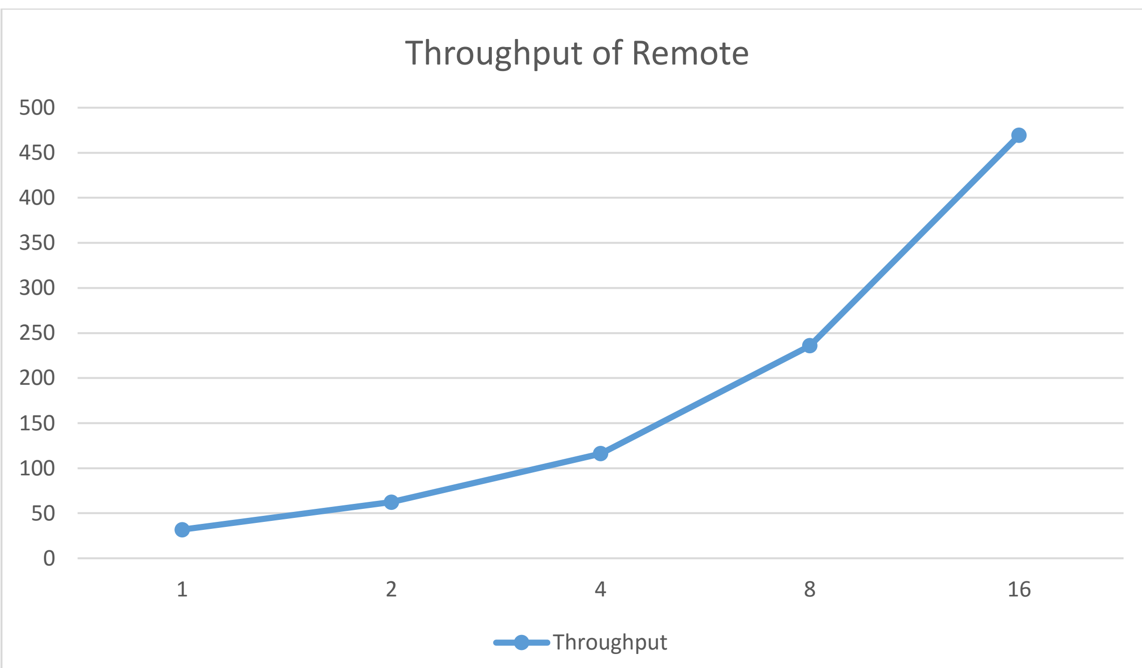
Local:

Time Taken for Executing 10k Sleep 0 Tasks is as follows

Worker/Instructions	10s x 10
1	315.326
2	160.7
4	86.131
8	42.4
16	21.303

Throughput is calculated as no of instructions/ Time taken

Workers/ Instructions	10s x 10
1	31.71321
2	62.22775
4	116.1022
8	235.8491
16	469.4175



It can be noted that the throughput increases as the number of workers are increased. This will show that the system is truly distributional and the total work is being efficiently divided and thus adding more workers is going to increase the processing. Thus we can conclude that the system is scalable

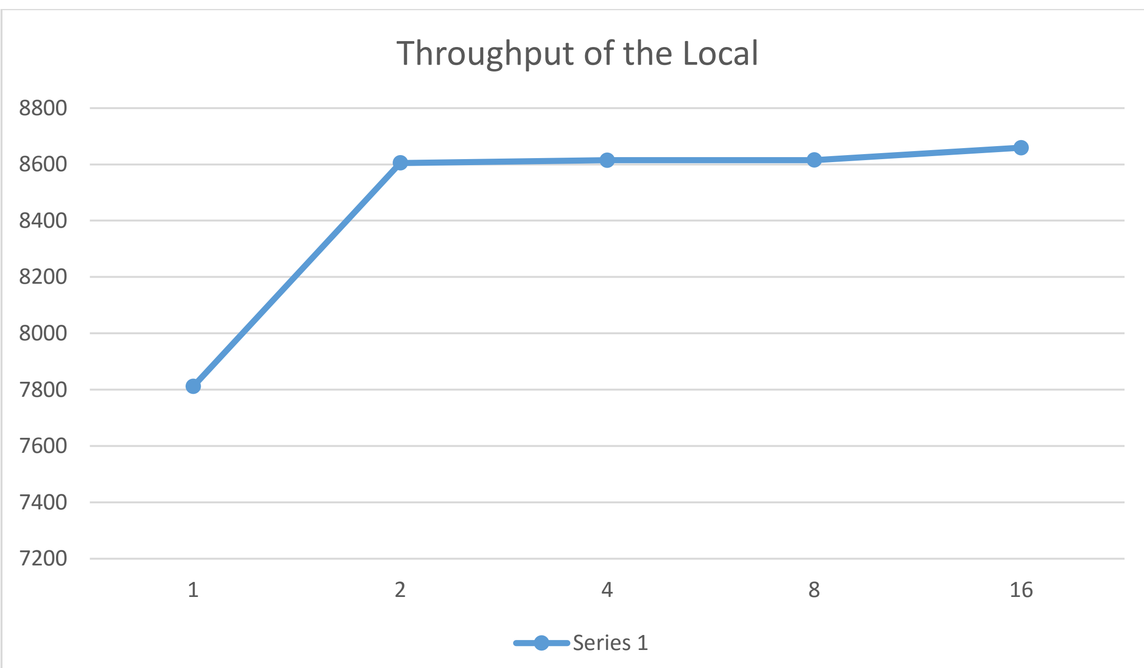
Local:

Time Taken for Executing 10k Sleep 0 Tasks is as follows

Threads/Instructions	10s x 10
1	12.801
2	11.6211
4	11.6087
8	11.6079
16	11.549

Throughput is calculated as no of instructions/ Time taken

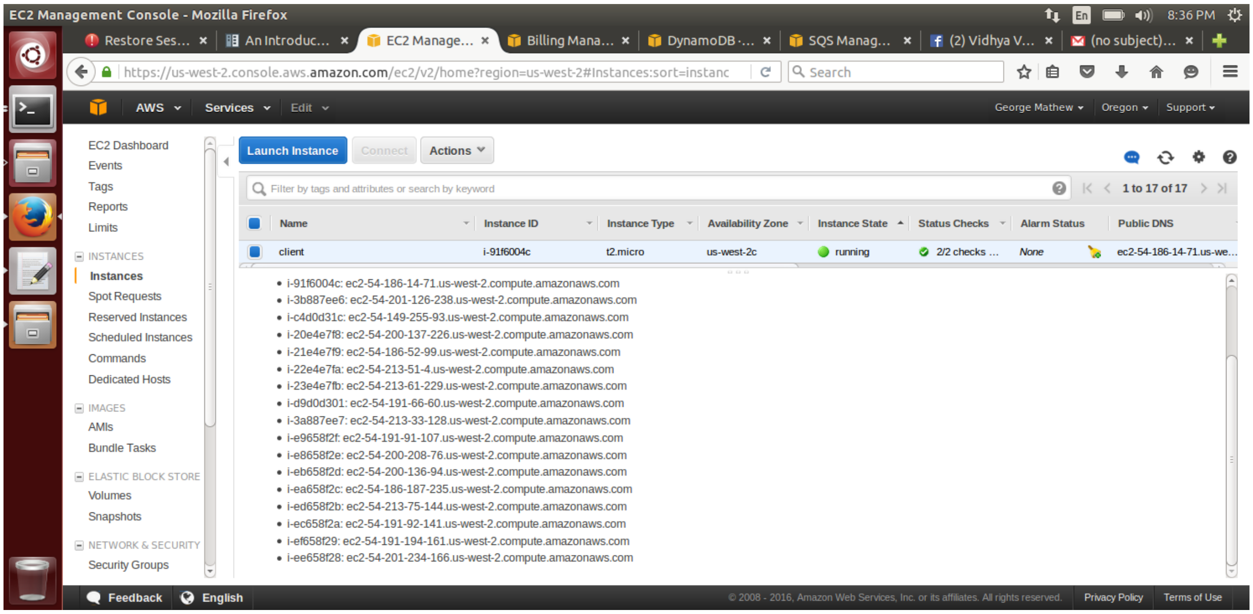
Threads/Instructions	10s x 10
1	7811.89
2	8605.037
4	8614.23
8	8614.823
16	8658.758



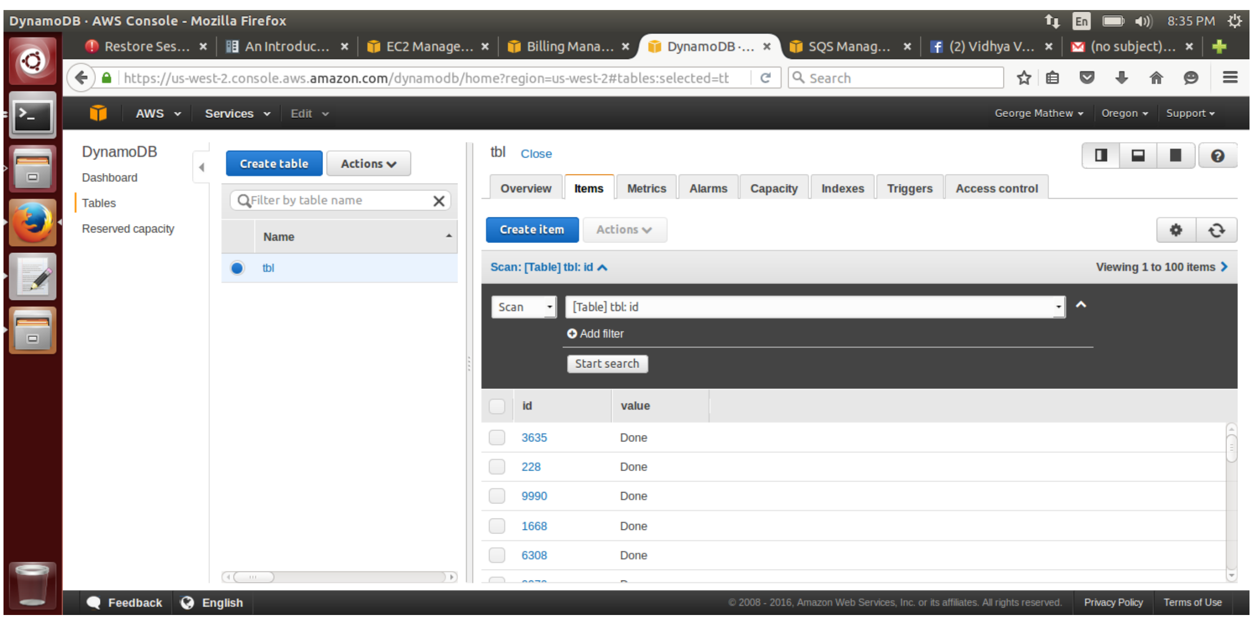
There is an increase in throughput when there are two threads but after that the throughput remains more or less the same. So it can be concluded that in a t2.micro system two threads can handle instruction read and write better than one thread. In case of one thread the resources are not being fully utilized

3)Screenshots

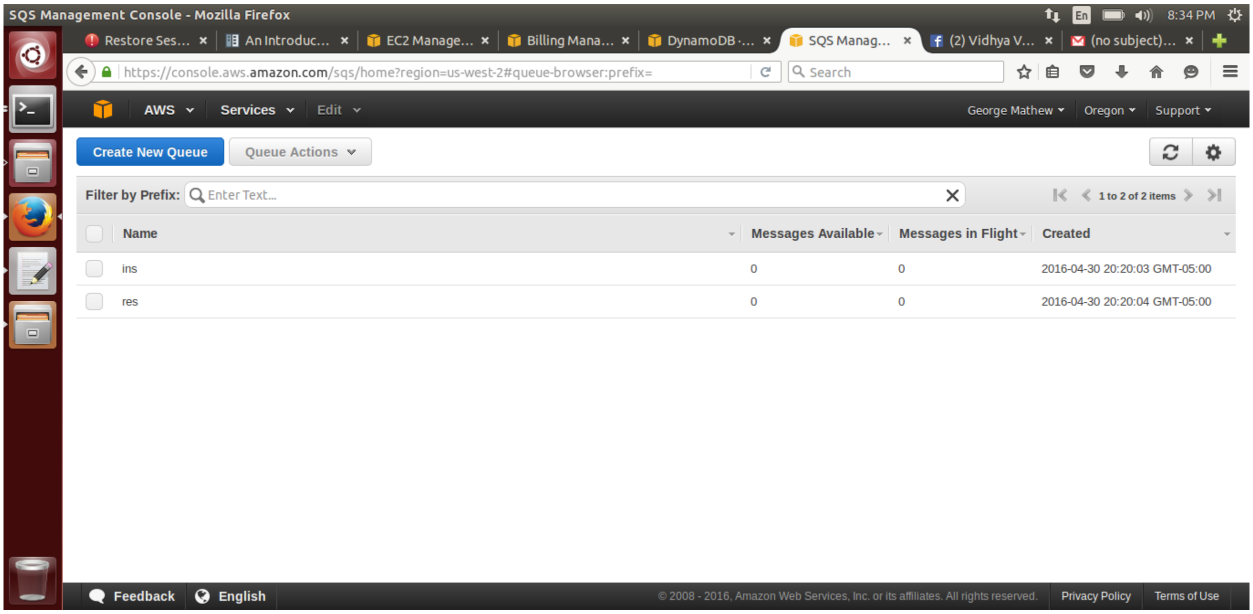
16 worker Framework



DynamoDB

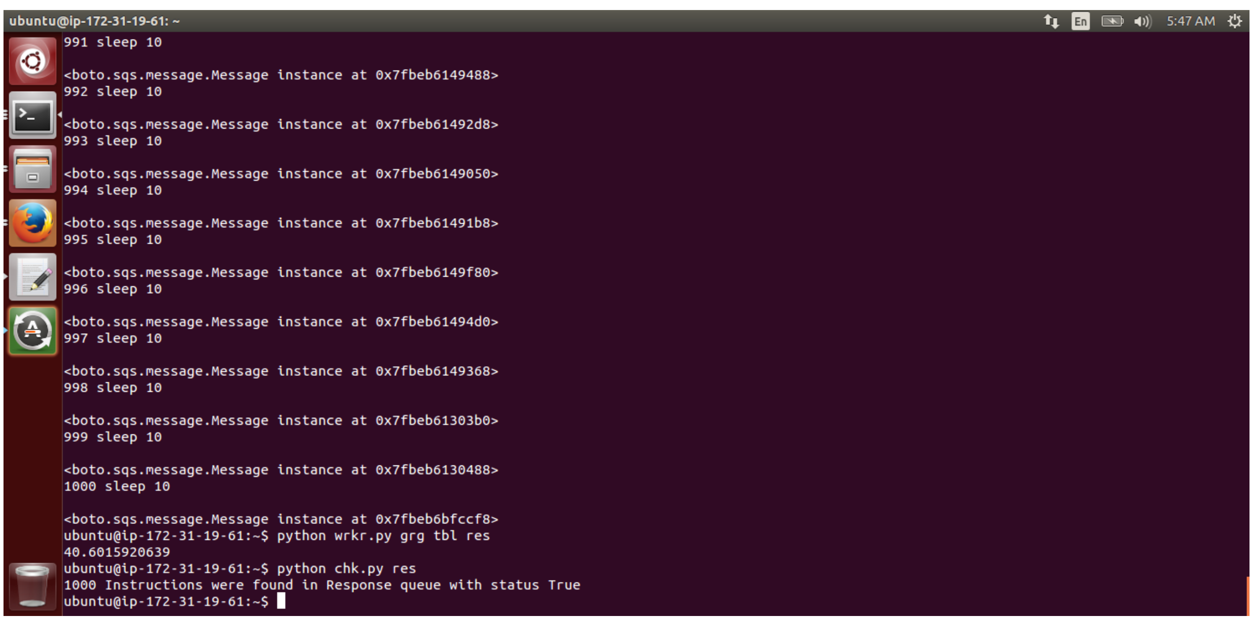


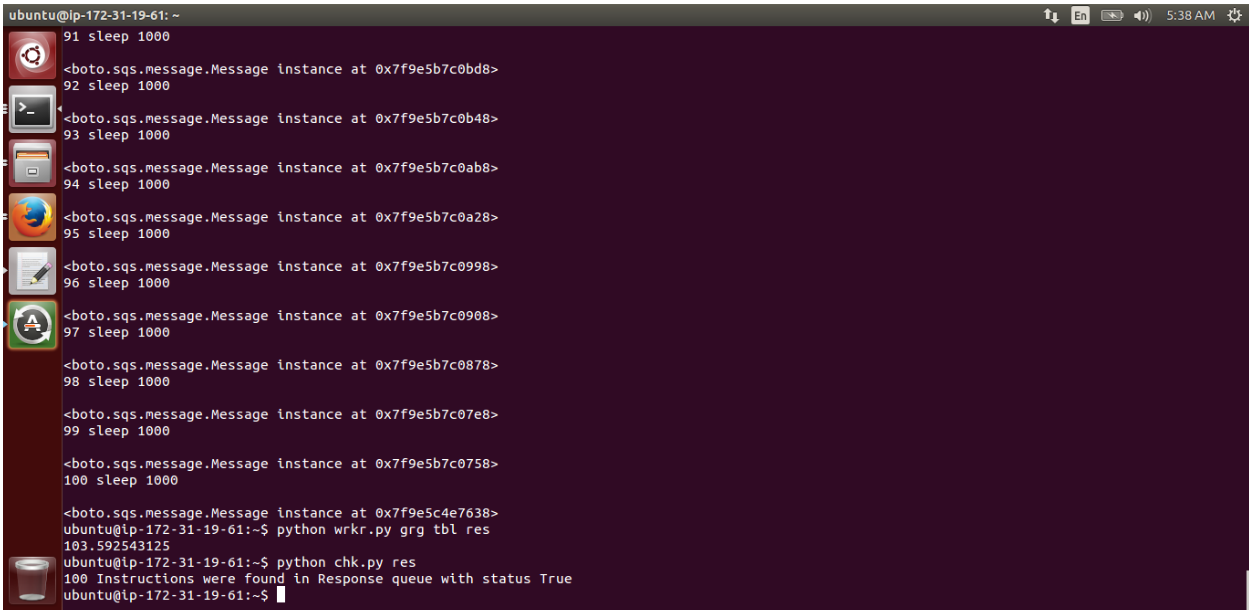
SQS

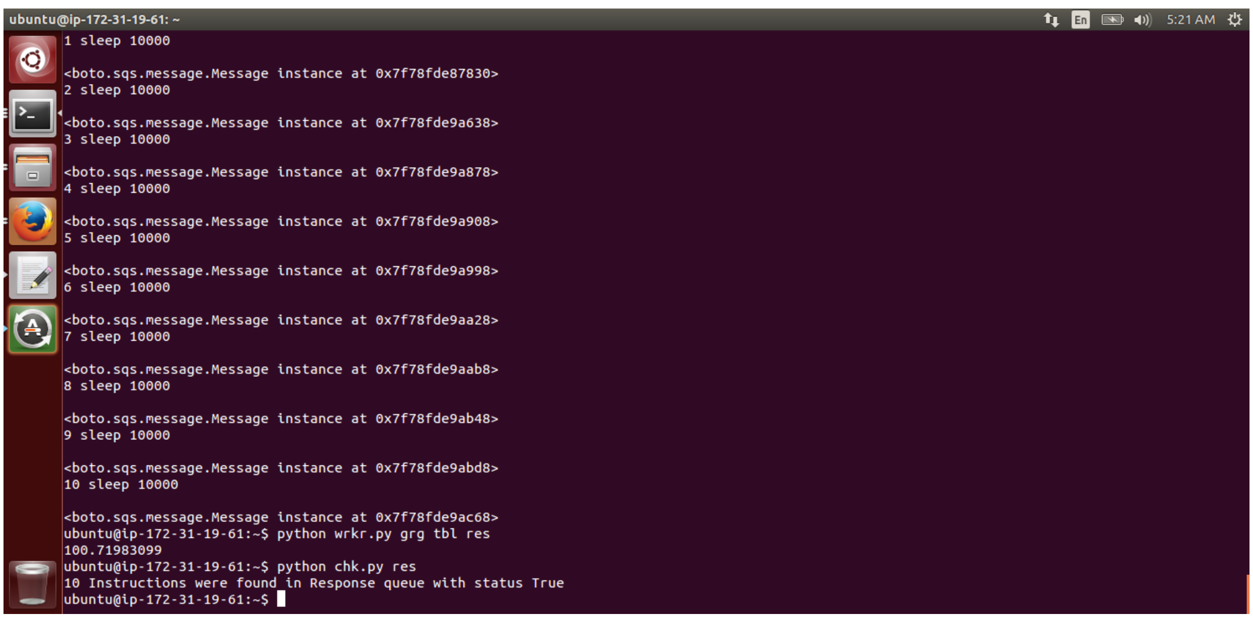


Experiments:

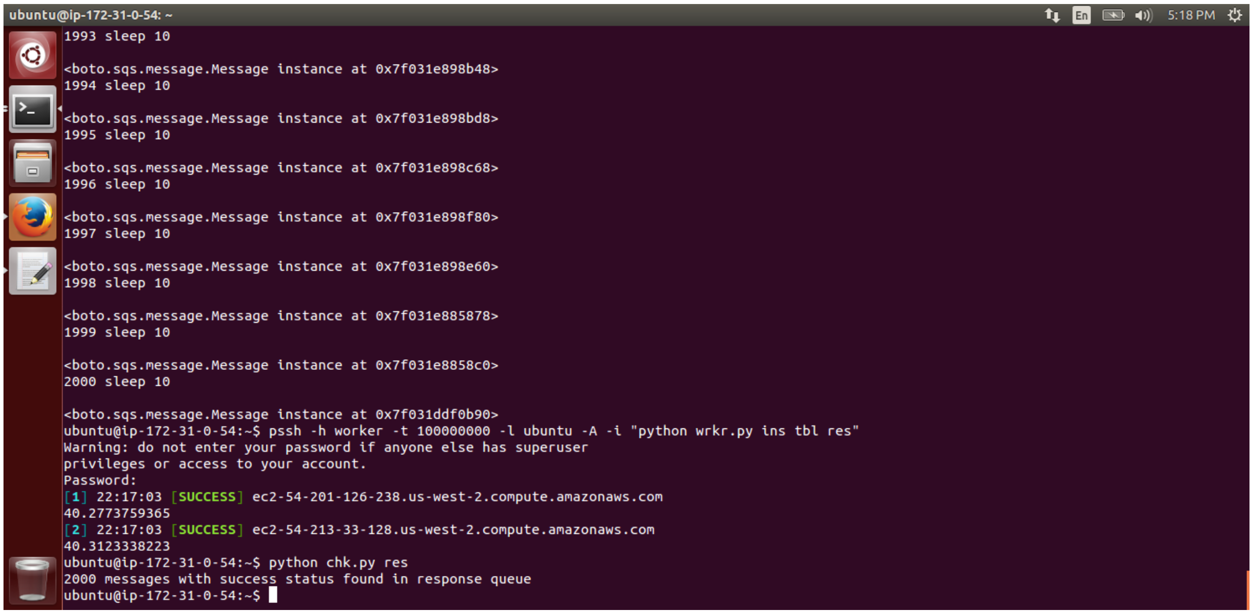
1 worker

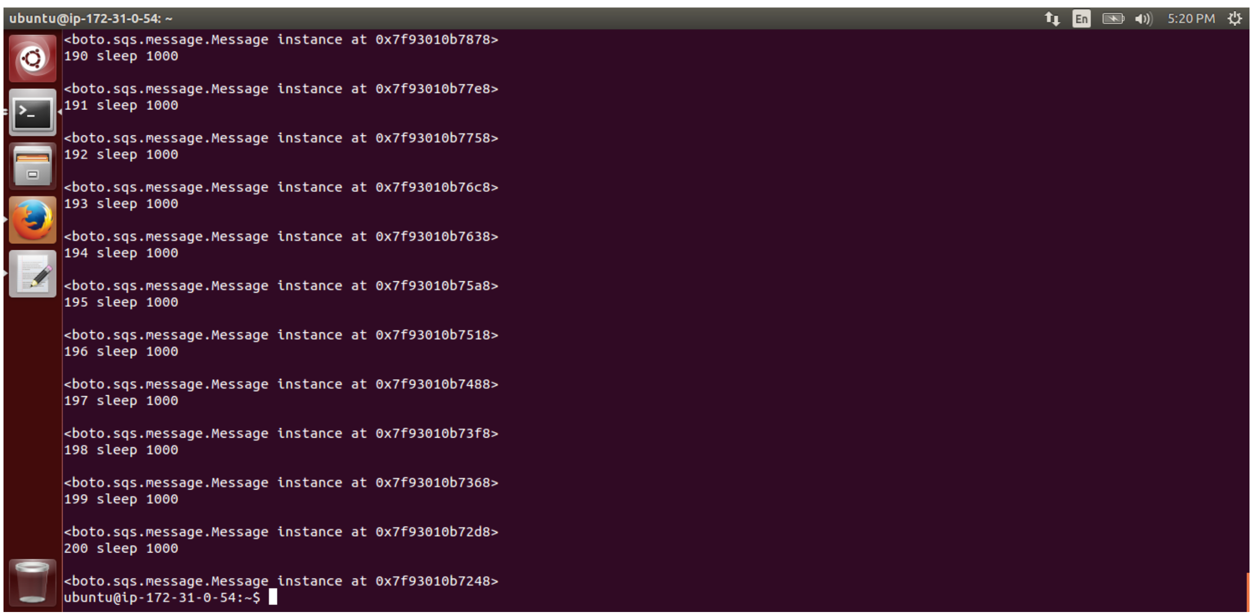


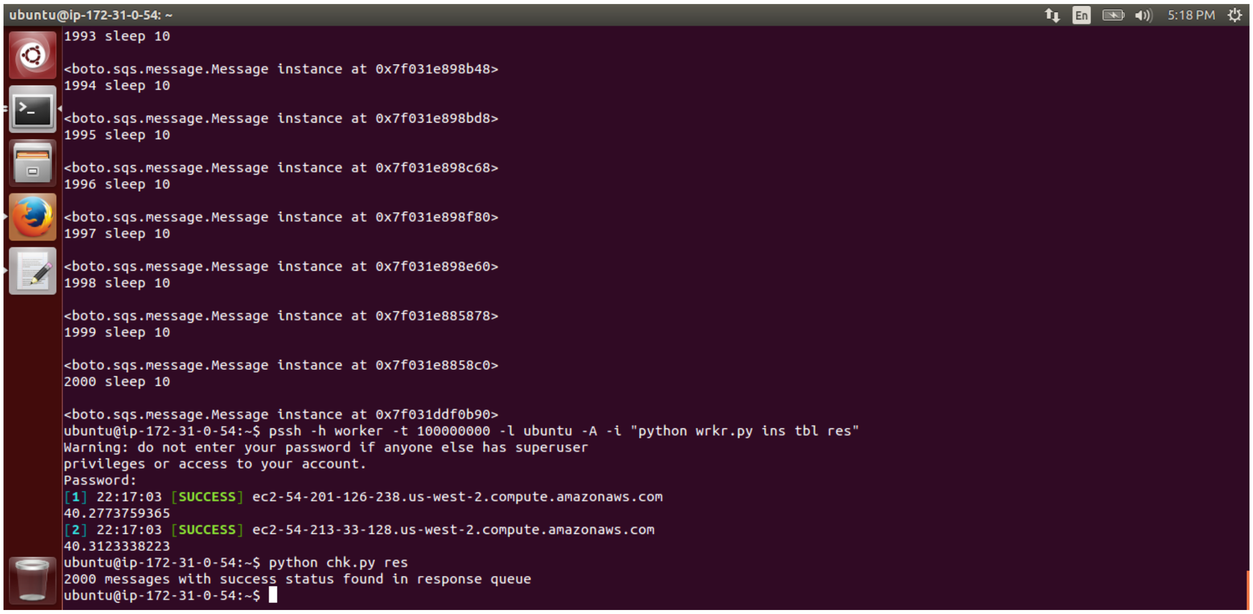

ubuntu@ip-172-31-19-61: ~
En
5:38 AM
91 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0bd8>
92 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0b48>
93 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0ab8>
94 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0a28>
95 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0998>
96 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0908>
97 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0878>
98 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c07e8>
99 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5b7c0758>
100 sleep 1000
<boto.sqs.message.Message instance at 0x7f9e5c4e7638>
ubuntu@ip-172-31-19-61:~$ python wrkr.py grg tbl res
103.592543125
ubuntu@ip-172-31-19-61:~$ python chk.py res
100 Instructions were found in Response queue with status True
ubuntu@ip-172-31-19-61:~$


ubuntu@ip-172-31-19-61: ~
En
5:21 AM
1 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde87830>
2 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9a638>
3 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9a878>
4 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9a908>
5 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9a998>
6 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9aa28>
7 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9aab8>
8 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9ab48>
9 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9abd8>
10 sleep 10000
<boto.sqs.message.Message instance at 0x7f78fde9ac68>
ubuntu@ip-172-31-19-61:~$ python wrkr.py grg tbl res
100.71983099
ubuntu@ip-172-31-19-61:~$ python chk.py res
10 Instructions were found in Response queue with status True
ubuntu@ip-172-31-19-61:~$

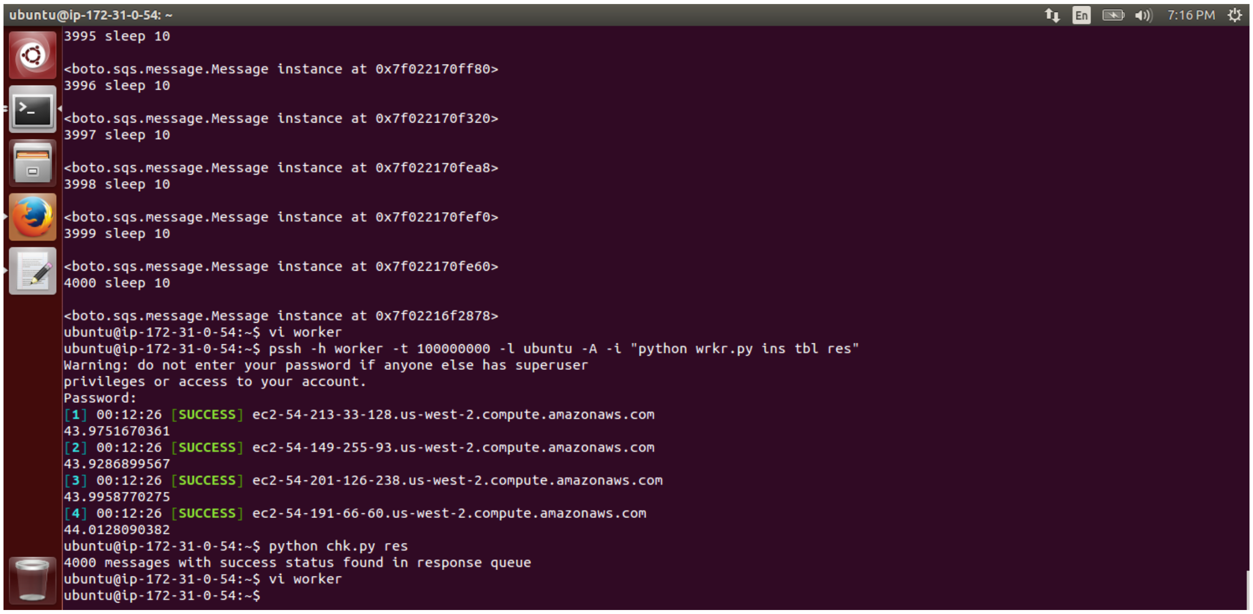
2 Workers


ubuntu@ip-172-31-0-54: ~
5:18 PM
1993 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898b48>
1994 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898bd8>
1995 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898c68>
1996 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898f80>
1997 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898e60>
1998 sleep 10
<boto.sqs.message.Message instance at 0x7f031e885878>
1999 sleep 10
<boto.sqs.message.Message instance at 0x7f031e8858c0>
2000 sleep 10
<boto.sqs.message.Message instance at 0x7f031ddf0b90>
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 22:17:03 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
40.2773759365
[2] 22:17:03 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
40.3123338223
ubuntu@ip-172-31-0-54:~$ python chk.py res
2000 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$

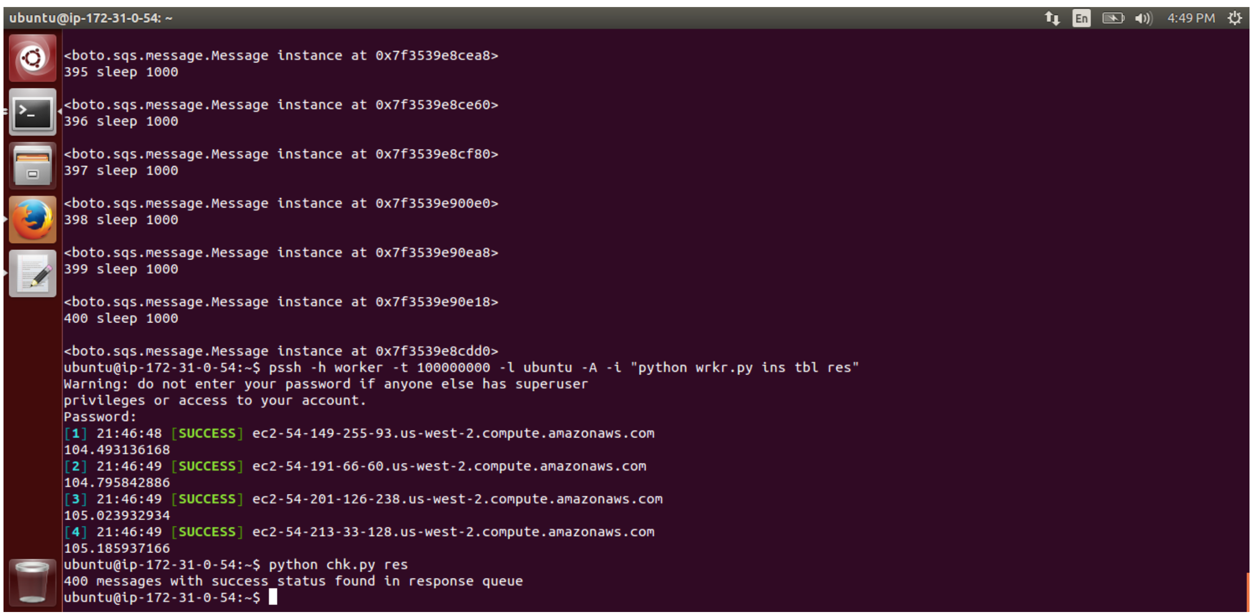

ubuntu@ip-172-31-0-54: ~
5:20 PM
<boto.sqs.message.Message instance at 0x7f93010b7878>
190 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b77e8>
191 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b7758>
192 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b76c8>
193 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b7638>
194 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b75a8>
195 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b7518>
196 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b7488>
197 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b73f8>
198 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b7368>
199 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b72d8>
200 sleep 1000
<boto.sqs.message.Message instance at 0x7f93010b7248>
ubuntu@ip-172-31-0-54:~$


ubuntu@ip-172-31-0-54: ~
5:18 PM
1993 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898b48>
1994 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898bd8>
1995 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898c68>
1996 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898f80>
1997 sleep 10
<boto.sqs.message.Message instance at 0x7f031e898e60>
1998 sleep 10
<boto.sqs.message.Message instance at 0x7f031e885878>
1999 sleep 10
<boto.sqs.message.Message instance at 0x7f031e8858c0>
2000 sleep 10
<boto.sqs.message.Message instance at 0x7f031ddf0b90>
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 22:17:03 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
40.2773759365
[2] 22:17:03 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
40.3123338223
ubuntu@ip-172-31-0-54:~$ python chk.py res
2000 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$

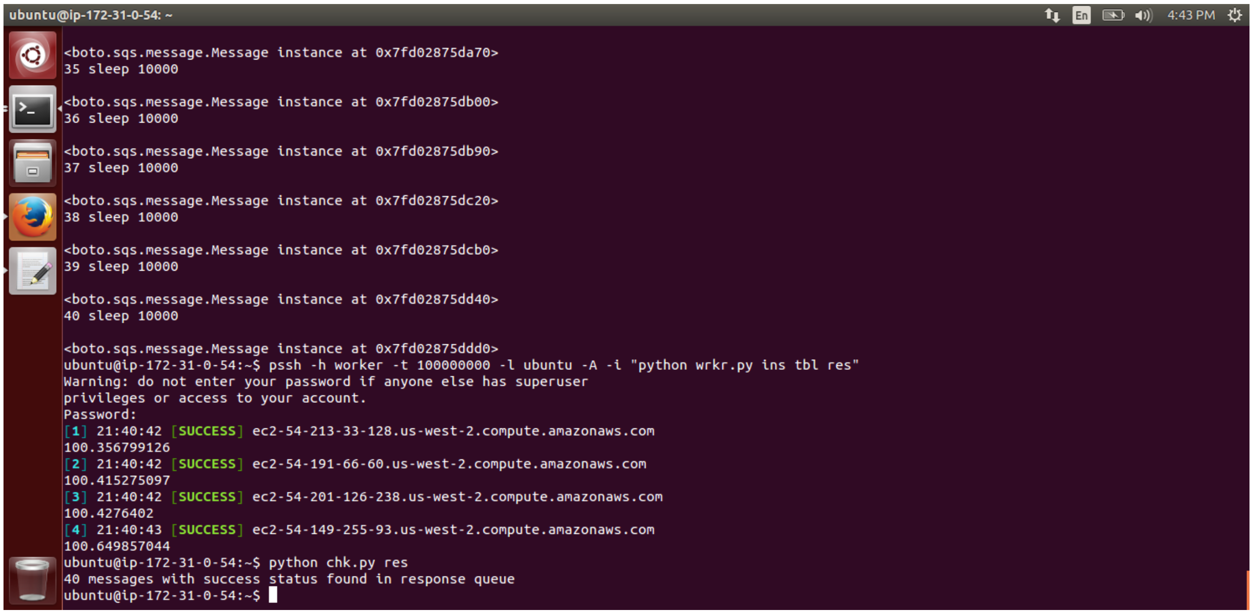
4 workers:



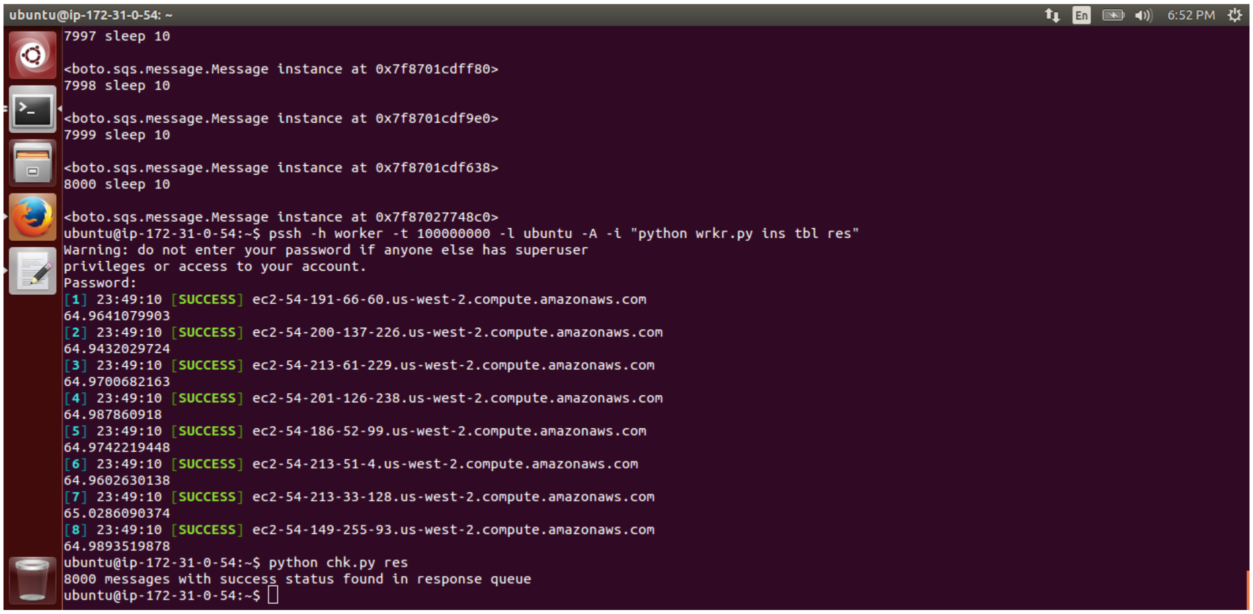
```
ubuntu@ip-172-31-0-54: ~                                        7:16 PM
3995 sleep 10

<boto.sqs.message.Message instance at 0x7f022170ff80>
3996 sleep 10

<boto.sqs.message.Message instance at 0x7f022170f320>
3997 sleep 10

<boto.sqs.message.Message instance at 0x7f022170fea8>
3998 sleep 10

<boto.sqs.message.Message instance at 0x7f022170fef0>
3999 sleep 10

<boto.sqs.message.Message instance at 0x7f022170fe60>
4000 sleep 10

<boto.sqs.message.Message instance at 0x7f02216f2878>
ubuntu@ip-172-31-0-54:~$ vi worker
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 00:12:26 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
43.9751670361
[2] 00:12:26 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
43.9286899567
[3] 00:12:26 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
43.9958770275
[4] 00:12:26 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
44.0128090382
ubuntu@ip-172-31-0-54:~$ python chk.py res
4000 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$ vi worker
ubuntu@ip-172-31-0-54:~$
```

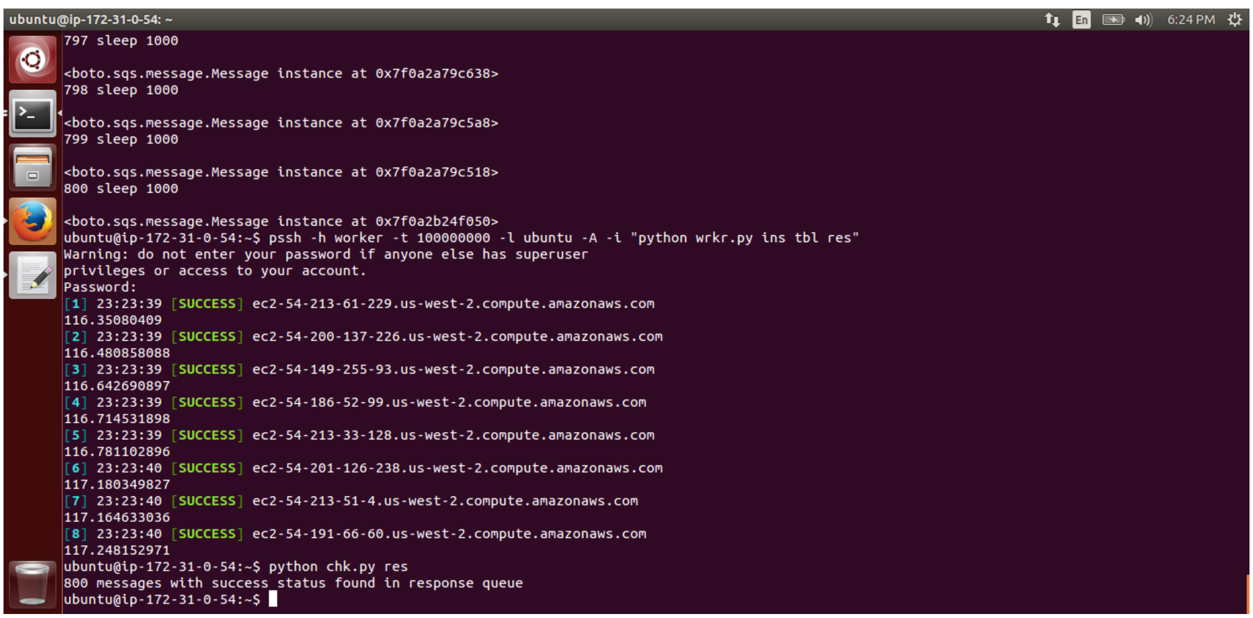


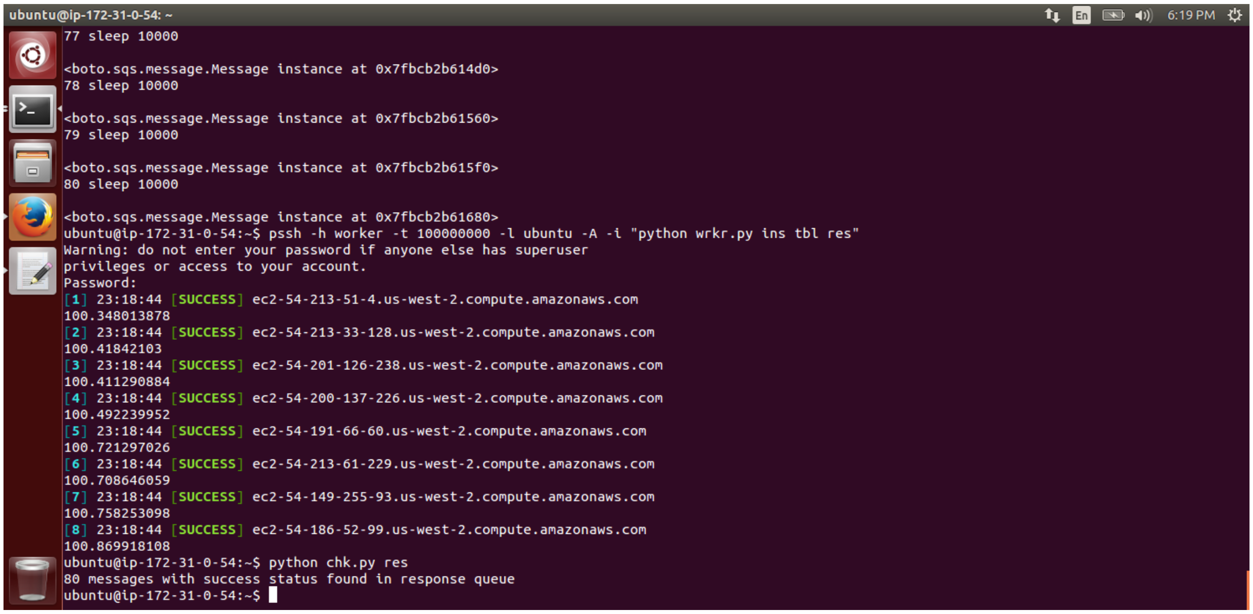
```
ubuntu@ip-172-31-0-54: ~                                        4:49 PM

<boto.sqs.message.Message instance at 0x7f3539e8cea8>
395 sleep 1000

<boto.sqs.message.Message instance at 0x7f3539e8ce60>
396 sleep 1000

<boto.sqs.message.Message instance at 0x7f3539e8cf80>
397 sleep 1000

<boto.sqs.message.Message instance at 0x7f3539e900e0>
398 sleep 1000

<boto.sqs.message.Message instance at 0x7f3539e90ea8>
399 sleep 1000

<boto.sqs.message.Message instance at 0x7f3539e90e18>
400 sleep 1000

<boto.sqs.message.Message instance at 0x7f3539e8cdd0>
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 21:46:48 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
104.493136168
[2] 21:46:49 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
104.795842886
[3] 21:46:49 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
105.023932934
[4] 21:46:49 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
105.185937166
ubuntu@ip-172-31-0-54:~$ python chk.py res
400 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$
```


ubuntu@ip-172-31-0-54: ~
4:43 PM
<boto.sqs.message.Message instance at 0x7fd02875da70>
35 sleep 10000
<boto.sqs.message.Message instance at 0x7fd02875db00>
36 sleep 10000
<boto.sqs.message.Message instance at 0x7fd02875db90>
37 sleep 10000
<boto.sqs.message.Message instance at 0x7fd02875dc20>
38 sleep 10000
<boto.sqs.message.Message instance at 0x7fd02875dcb0>
39 sleep 10000
<boto.sqs.message.Message instance at 0x7fd02875dd40>
40 sleep 10000
<boto.sqs.message.Message instance at 0x7fd02875ddd0>
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 21:40:42 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
100.356799126
[2] 21:40:42 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
100.415275097
[3] 21:40:42 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
100.4276402
[4] 21:40:43 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
100.649857044
ubuntu@ip-172-31-0-54:~$ python chk.py res
40 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$

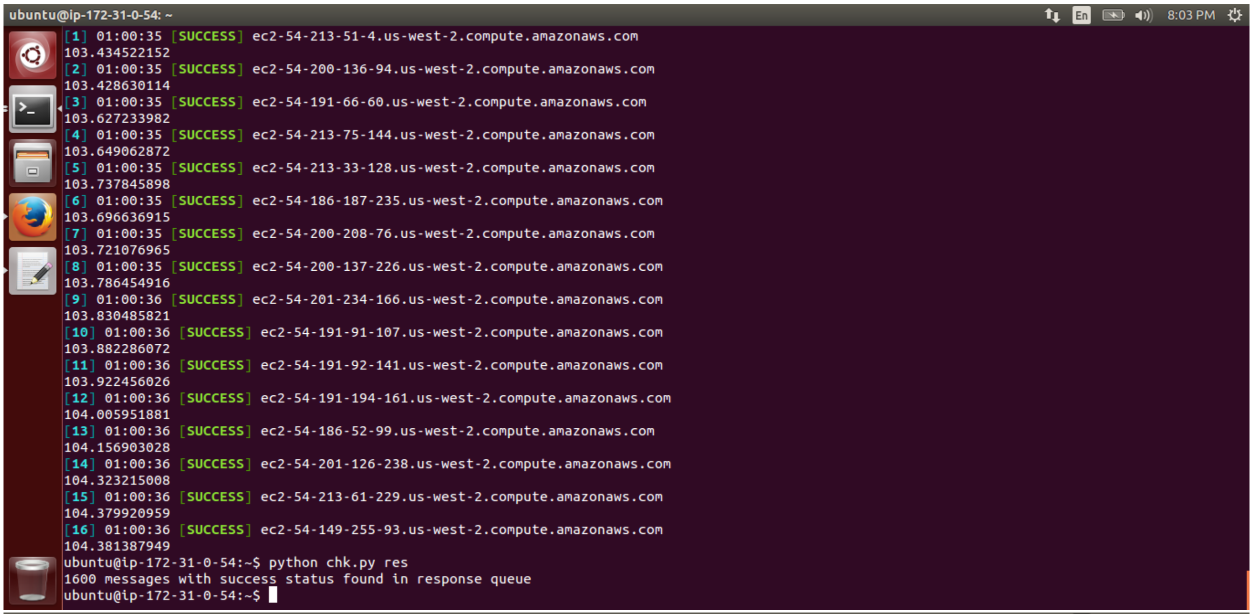
8 workers

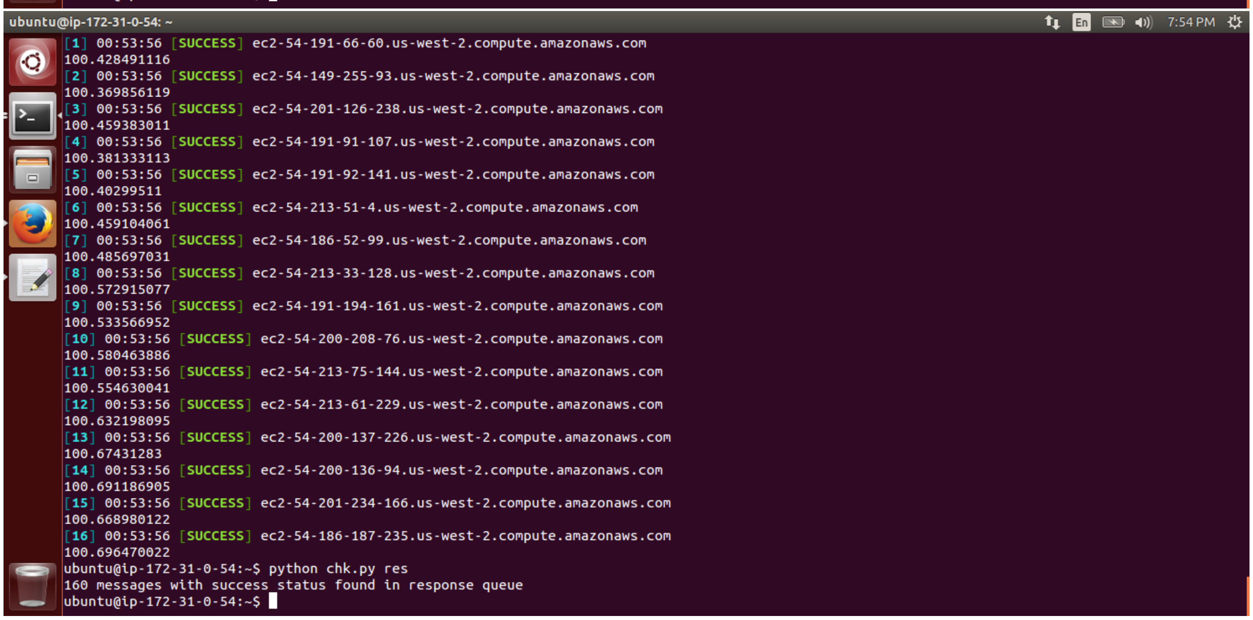

ubuntu@ip-172-31-0-54: ~
6:52 PM
7997 sleep 10
<boto.sqs.message.Message instance at 0x7f8701cdff80>
7998 sleep 10
<boto.sqs.message.Message instance at 0x7f8701cdf9e0>
7999 sleep 10
<boto.sqs.message.Message instance at 0x7f8701cdf638>
8000 sleep 10
<boto.sqs.message.Message instance at 0x7f87027748c0>
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 23:49:10 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
64.9641079903
[2] 23:49:10 [SUCCESS] ec2-54-200-137-226.us-west-2.compute.amazonaws.com
64.9432029724
[3] 23:49:10 [SUCCESS] ec2-54-213-61-229.us-west-2.compute.amazonaws.com
64.9700682163
[4] 23:49:10 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
64.987860918
[5] 23:49:10 [SUCCESS] ec2-54-186-52-99.us-west-2.compute.amazonaws.com
64.9742219448
[6] 23:49:10 [SUCCESS] ec2-54-213-51-4.us-west-2.compute.amazonaws.com
64.9602630138
[7] 23:49:10 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
65.0286090374
[8] 23:49:10 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
64.9893519878
ubuntu@ip-172-31-0-54:~$ python chk.py res
8000 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$

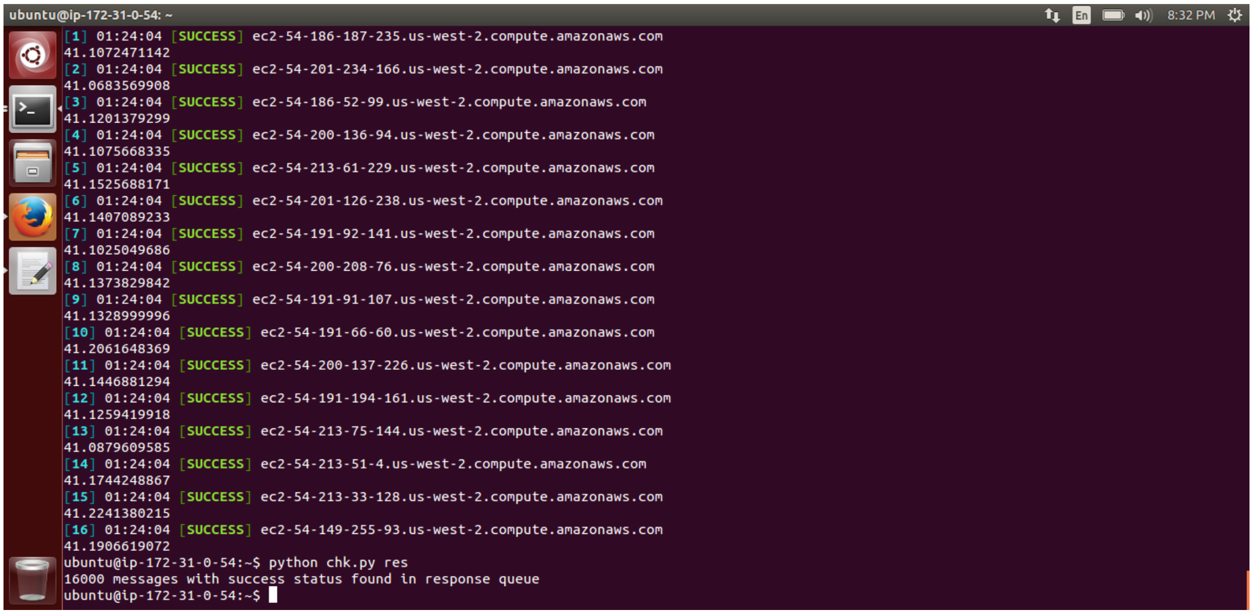

ubuntu@ip-172-31-0-54: ~
6:24 PM
797 sleep 1000
<boto.sqs.message.Message instance at 0x7f0a2a79c638>
798 sleep 1000
<boto.sqs.message.Message instance at 0x7f0a2a79c5a8>
799 sleep 1000
<boto.sqs.message.Message instance at 0x7f0a2a79c518>
800 sleep 1000
<boto.sqs.message.Message instance at 0x7f0a2b24f050>
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 23:23:39 [SUCCESS] ec2-54-213-61-229.us-west-2.compute.amazonaws.com
116.35080409
[2] 23:23:39 [SUCCESS] ec2-54-200-137-226.us-west-2.compute.amazonaws.com
116.480858088
[3] 23:23:39 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
116.642690897
[4] 23:23:39 [SUCCESS] ec2-54-186-52-99.us-west-2.compute.amazonaws.com
116.714531898
[5] 23:23:39 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
116.781102896
[6] 23:23:40 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
117.180349827
[7] 23:23:40 [SUCCESS] ec2-54-213-51-4.us-west-2.compute.amazonaws.com
117.164633036
[8] 23:23:40 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
117.248152971
ubuntu@ip-172-31-0-54:~$ python chk.py res
800 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$


ubuntu@ip-172-31-0-54: ~
6:19 PM
77 sleep 10000
<boto.sqs.message.Message instance at 0x7fbcb2b614d0>
78 sleep 10000
<boto.sqs.message.Message instance at 0x7fbcb2b61560>
79 sleep 10000
<boto.sqs.message.Message instance at 0x7fbcb2b615f0>
80 sleep 10000
<boto.sqs.message.Message instance at 0x7fbcb2b61680>
ubuntu@ip-172-31-0-54:~$ pssh -h worker -t 100000000 -l ubuntu -A -i "python wrkr.py ins tbl res"
Warning: do not enter your password if anyone else has superuser
privileges or access to your account.
Password:
[1] 23:18:44 [SUCCESS] ec2-54-213-51-4.us-west-2.compute.amazonaws.com
100.348013878
[2] 23:18:44 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
100.41842103
[3] 23:18:44 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
100.411290884
[4] 23:18:44 [SUCCESS] ec2-54-200-137-226.us-west-2.compute.amazonaws.com
100.492239952
[5] 23:18:44 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
100.721297026
[6] 23:18:44 [SUCCESS] ec2-54-213-61-229.us-west-2.compute.amazonaws.com
100.708646059
[7] 23:18:44 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
100.758253098
[8] 23:18:44 [SUCCESS] ec2-54-186-52-99.us-west-2.compute.amazonaws.com
100.869918108
ubuntu@ip-172-31-0-54:~$ python chk.py res
80 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$

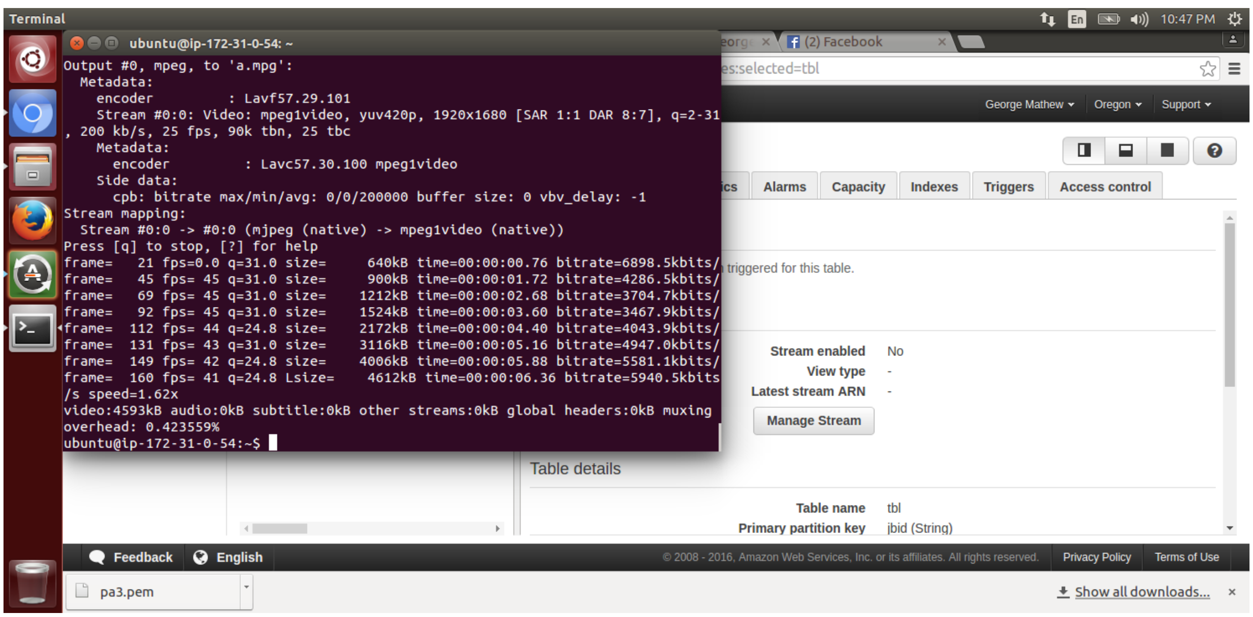
16 workers

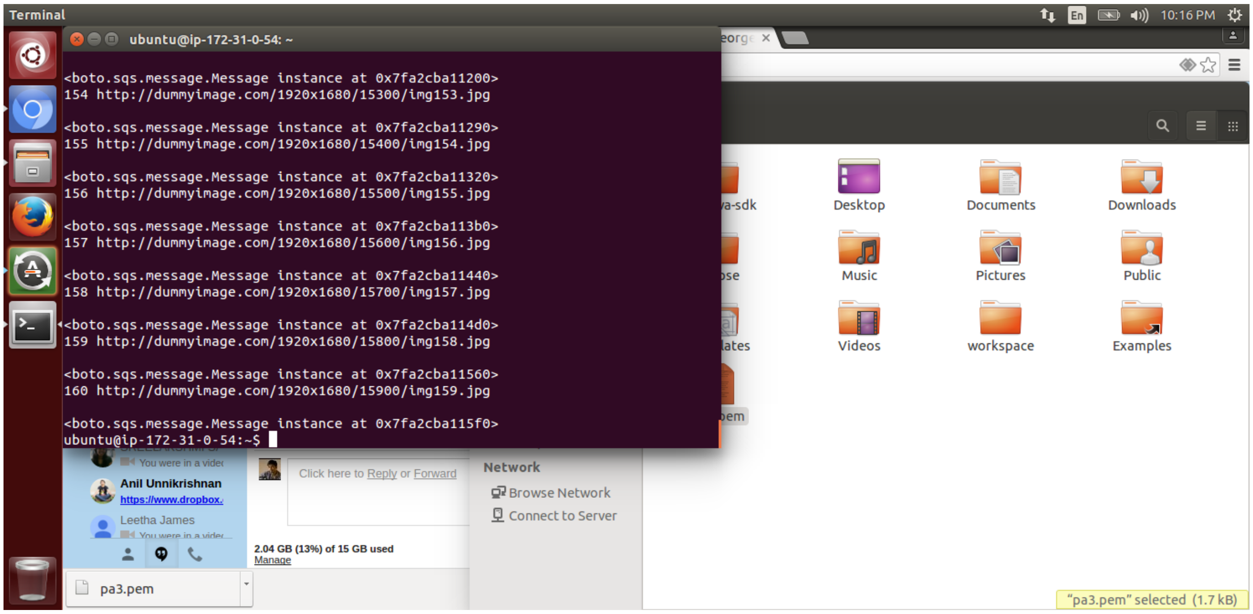

ubuntu@ip-172-31-0-54: ~
8:03 PM
[1] 01:00:35 [SUCCESS] ec2-54-213-51-4.us-west-2.compute.amazonaws.com
103.434522152
[2] 01:00:35 [SUCCESS] ec2-54-200-136-94.us-west-2.compute.amazonaws.com
103.428630114
[3] 01:00:35 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
103.627233982
[4] 01:00:35 [SUCCESS] ec2-54-213-75-144.us-west-2.compute.amazonaws.com
103.649062872
[5] 01:00:35 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
103.737845898
[6] 01:00:35 [SUCCESS] ec2-54-186-187-235.us-west-2.compute.amazonaws.com
103.696636915
[7] 01:00:35 [SUCCESS] ec2-54-200-208-76.us-west-2.compute.amazonaws.com
103.721076965
[8] 01:00:35 [SUCCESS] ec2-54-200-137-226.us-west-2.compute.amazonaws.com
103.786454916
[9] 01:00:36 [SUCCESS] ec2-54-201-234-166.us-west-2.compute.amazonaws.com
103.830485821
[10] 01:00:36 [SUCCESS] ec2-54-191-91-107.us-west-2.compute.amazonaws.com
103.882286072
[11] 01:00:36 [SUCCESS] ec2-54-191-92-141.us-west-2.compute.amazonaws.com
103.922456026
[12] 01:00:36 [SUCCESS] ec2-54-191-194-161.us-west-2.compute.amazonaws.com
104.005951881
[13] 01:00:36 [SUCCESS] ec2-54-186-52-99.us-west-2.compute.amazonaws.com
104.156903028
[14] 01:00:36 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
104.323215008
[15] 01:00:36 [SUCCESS] ec2-54-213-61-229.us-west-2.compute.amazonaws.com
104.379920959
[16] 01:00:36 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
104.381387949
ubuntu@ip-172-31-0-54:~$ python chk.py res
1600 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$


ubuntu@ip-172-31-0-54: ~
7:54 PM
[1] 00:53:56 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
100.428491116
[2] 00:53:56 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
100.369856119
[3] 00:53:56 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
100.459383011
[4] 00:53:56 [SUCCESS] ec2-54-191-91-107.us-west-2.compute.amazonaws.com
100.381333113
[5] 00:53:56 [SUCCESS] ec2-54-191-92-141.us-west-2.compute.amazonaws.com
100.40299511
[6] 00:53:56 [SUCCESS] ec2-54-213-51-4.us-west-2.compute.amazonaws.com
100.459104061
[7] 00:53:56 [SUCCESS] ec2-54-186-52-99.us-west-2.compute.amazonaws.com
100.485697031
[8] 00:53:56 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
100.572915077
[9] 00:53:56 [SUCCESS] ec2-54-191-194-161.us-west-2.compute.amazonaws.com
100.533566952
[10] 00:53:56 [SUCCESS] ec2-54-200-208-76.us-west-2.compute.amazonaws.com
100.580463886
[11] 00:53:56 [SUCCESS] ec2-54-213-75-144.us-west-2.compute.amazonaws.com
100.554630041
[12] 00:53:56 [SUCCESS] ec2-54-213-61-229.us-west-2.compute.amazonaws.com
100.632198095
[13] 00:53:56 [SUCCESS] ec2-54-200-137-226.us-west-2.compute.amazonaws.com
100.67431283
[14] 00:53:56 [SUCCESS] ec2-54-200-136-94.us-west-2.compute.amazonaws.com
100.691186905
[15] 00:53:56 [SUCCESS] ec2-54-201-234-166.us-west-2.compute.amazonaws.com
100.668980122
[16] 00:53:56 [SUCCESS] ec2-54-186-187-235.us-west-2.compute.amazonaws.com
100.696470022
ubuntu@ip-172-31-0-54:~$ python chk.py res
160 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$

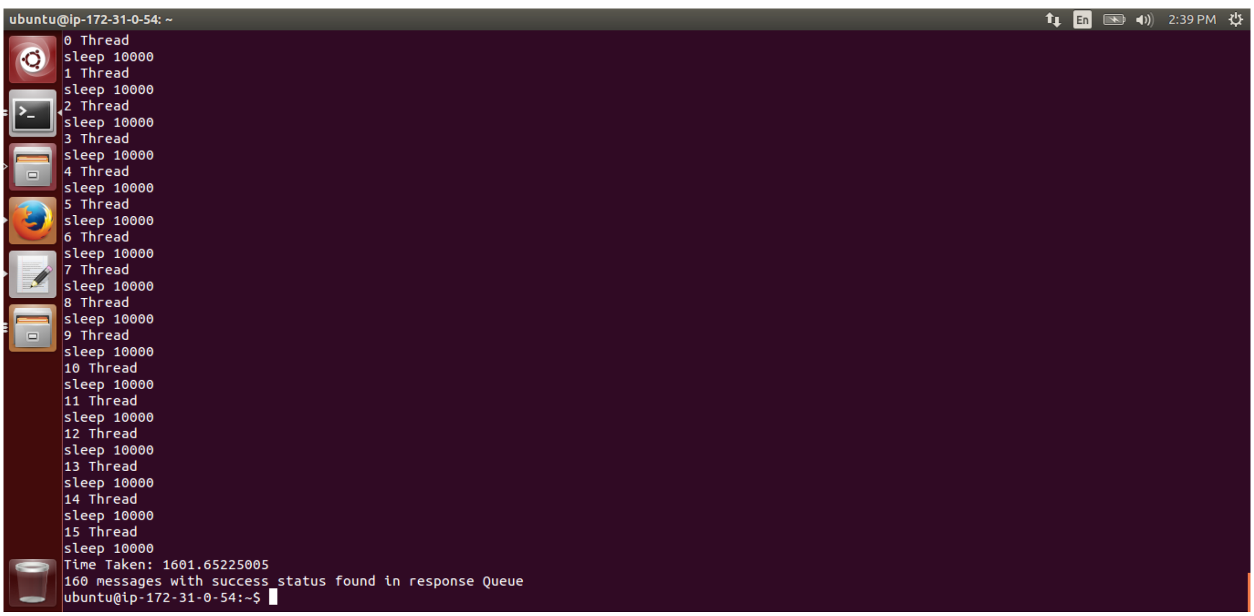

ubuntu@ip-172-31-0-54: ~
8:32 PM
[1] 01:24:04 [SUCCESS] ec2-54-186-187-235.us-west-2.compute.amazonaws.com
41.1072471142
[2] 01:24:04 [SUCCESS] ec2-54-201-234-166.us-west-2.compute.amazonaws.com
41.0683569908
[3] 01:24:04 [SUCCESS] ec2-54-186-52-99.us-west-2.compute.amazonaws.com
41.1201379299
[4] 01:24:04 [SUCCESS] ec2-54-200-136-94.us-west-2.compute.amazonaws.com
41.1075668335
[5] 01:24:04 [SUCCESS] ec2-54-213-61-229.us-west-2.compute.amazonaws.com
41.1525688171
[6] 01:24:04 [SUCCESS] ec2-54-201-126-238.us-west-2.compute.amazonaws.com
41.1407089233
[7] 01:24:04 [SUCCESS] ec2-54-191-92-141.us-west-2.compute.amazonaws.com
41.1025049686
[8] 01:24:04 [SUCCESS] ec2-54-200-208-76.us-west-2.compute.amazonaws.com
41.1373829842
[9] 01:24:04 [SUCCESS] ec2-54-191-91-107.us-west-2.compute.amazonaws.com
41.1328999996
[10] 01:24:04 [SUCCESS] ec2-54-191-66-60.us-west-2.compute.amazonaws.com
41.2061648369
[11] 01:24:04 [SUCCESS] ec2-54-200-137-226.us-west-2.compute.amazonaws.com
41.1446881294
[12] 01:24:04 [SUCCESS] ec2-54-191-194-161.us-west-2.compute.amazonaws.com
41.1259419918
[13] 01:24:04 [SUCCESS] ec2-54-213-75-144.us-west-2.compute.amazonaws.com
41.0879609585
[14] 01:24:04 [SUCCESS] ec2-54-213-51-4.us-west-2.compute.amazonaws.com
41.1744248867
[15] 01:24:04 [SUCCESS] ec2-54-213-33-128.us-west-2.compute.amazonaws.com
41.2241380215
[16] 01:24:04 [SUCCESS] ec2-54-149-255-93.us-west-2.compute.amazonaws.com
41.1906619072
ubuntu@ip-172-31-0-54:~$ python chk.py res
16000 messages with success status found in response queue
ubuntu@ip-172-31-0-54:~$

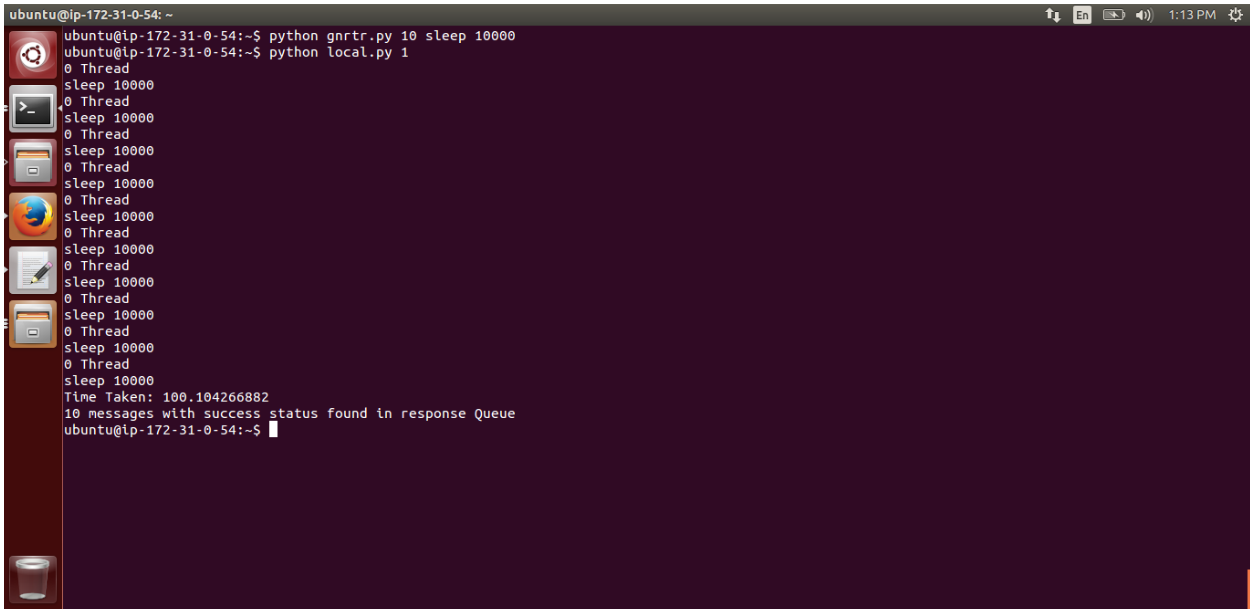
Animoto

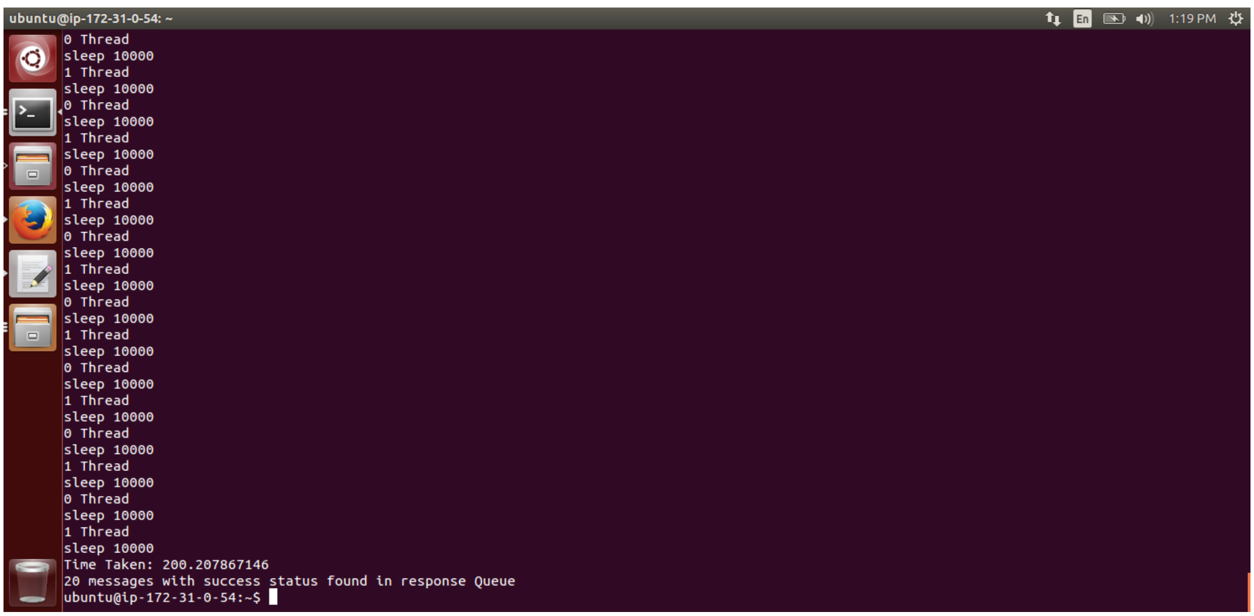

Terminal
10:47 PM
ubuntu@ip-172-31-0-54: ~
Output #0, mpeg, to 'a.mpg':
  Metadata:
    encoder   : Lavf57.29.101
    Stream #0:0: Video: mpeg1video, yuv420p, 1920x1680 [SAR 1:1 DAR 8:7], q=2-31, 200 kb/s, 25 fps, 90k tbn, 25 tbc
    Metadata:
      encoder   : Lavc57.30.100 mpeg1video
    Side data:
      cpb: bitrate max/min/avg: 0/0/200000 buffer size: 0 vbv_delay: -1
Stream mapping:
  Stream #0:0 -> #0:0 (mjpeg (native) -> mpeg1video (native))
Press [q] to stop, [?] for help
frame=   21 fps=0.0 q=31.0 size=   640kB time=00:00:00.76 bitrate=6898.5kbits/
frame=   45 fps= 45 q=31.0 size=   900kB time=00:00:01.72 bitrate=4286.5kbits/
frame=   69 fps= 45 q=31.0 size=   1212kB time=00:00:02.68 bitrate=3704.7kbits/
frame=   92 fps= 45 q=31.0 size=   1524kB time=00:00:03.60 bitrate=3467.9kbits/
frame= 112 fps= 44 q=24.8 size=   2172kB time=00:00:04.40 bitrate=4043.9kbits/
frame= 131 fps= 43 q=31.0 size=   3116kB time=00:00:05.16 bitrate=4947.0kbits/
frame= 149 fps= 42 q=24.8 size=   4006kB time=00:00:05.88 bitrate=5581.1kbits/
frame= 160 fps= 41 q=24.8 Lsize=   4612kB time=00:00:06.36 bitrate=5940.5kbits/s speed=1.62x
video:4593kB audio:0kB subtitle:0kB other streams:0kB global headers:0kB muxing overhead: 0.423559%
ubuntu@ip-172-31-0-54:~$
(2) Facebook
George Mathew
Oregon
Support
Alarms
Capacity
Indexes
Triggers
Access control
triggered for this table.
Stream enabled No
View type -
Latest stream ARN -
Manage Stream
Table details
Table name tbl
Primary partition key jbid (String)
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.
Privacy Policy
Terms of Use
pa3.pem
Show all downloads...

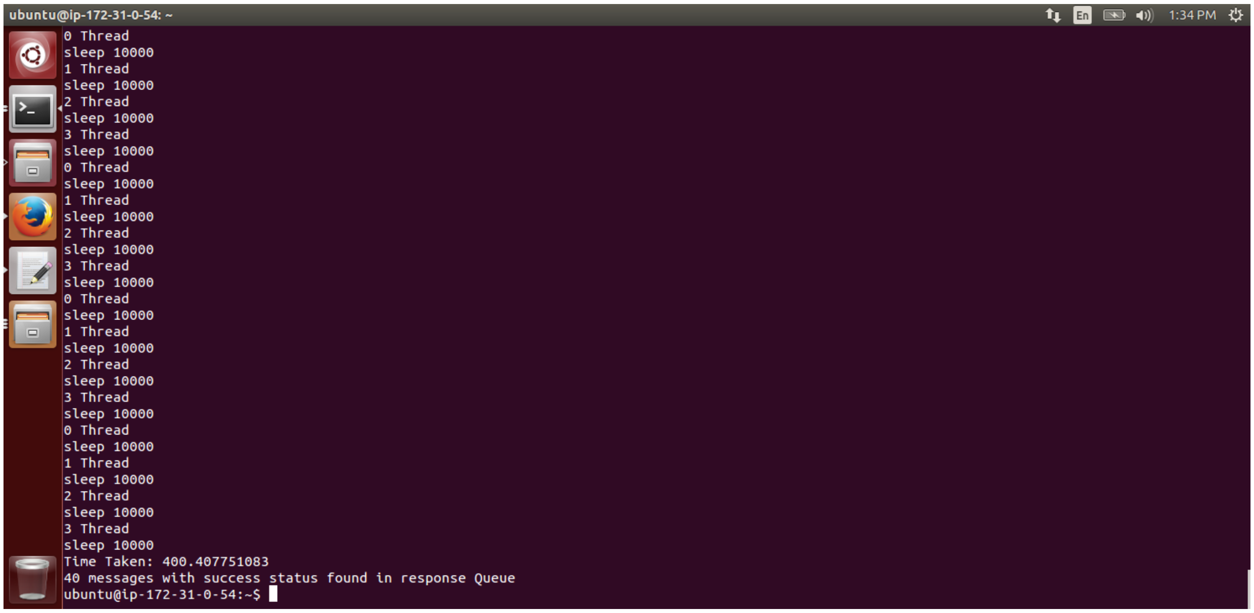

Terminal
10:16 PM
ubuntu@ip-172-31-0-54: ~
<boto.sqs.message.Message instance at 0x7fa2cba11200>
154 http://dummyimage.com/1920x1680/15300/img153.jpg
<boto.sqs.message.Message instance at 0x7fa2cba11290>
155 http://dummyimage.com/1920x1680/15400/img154.jpg
<boto.sqs.message.Message instance at 0x7fa2cba11320>
156 http://dummyimage.com/1920x1680/15500/img155.jpg
<boto.sqs.message.Message instance at 0x7fa2cba113b0>
157 http://dummyimage.com/1920x1680/15600/img156.jpg
<boto.sqs.message.Message instance at 0x7fa2cba11440>
158 http://dummyimage.com/1920x1680/15700/img157.jpg
<boto.sqs.message.Message instance at 0x7fa2cba114d0>
159 http://dummyimage.com/1920x1680/15800/img158.jpg
<boto.sqs.message.Message instance at 0x7fa2cba11560>
160 http://dummyimage.com/1920x1680/15900/img159.jpg
<boto.sqs.message.Message instance at 0x7fa2cba115f0>
ubuntu@ip-172-31-0-54:~$
Desktop
Documents
Downloads
Music
Pictures
Public
Videos
workspace
Examples
Network
Browse Network
Connect to Server
Anil Unnikrishnan
Leetha James
Click here to Reply or Forward
2.04 GB (13%) of 15 GB used
Manage
pa3.pem
"pa3.pem" selected (1.7 kB)

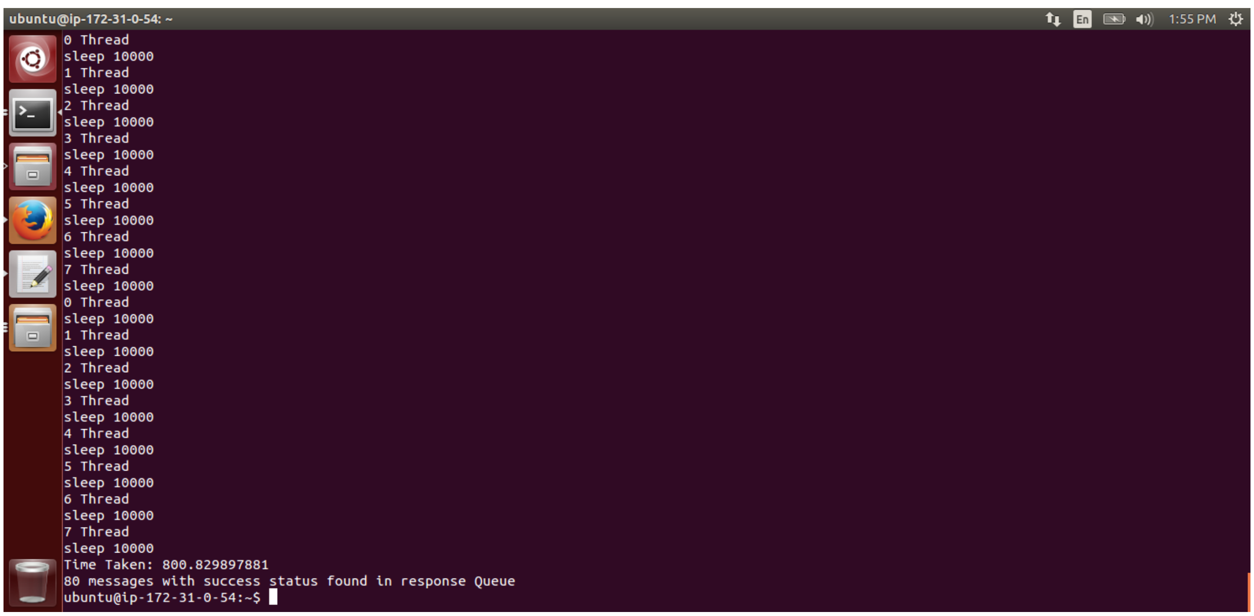
Local


ubuntu@ip-172-31-0-54: ~
2:39 PM
0 Thread
sleep 10000
1 Thread
sleep 10000
2 Thread
sleep 10000
3 Thread
sleep 10000
4 Thread
sleep 10000
5 Thread
sleep 10000
6 Thread
sleep 10000
7 Thread
sleep 10000
8 Thread
sleep 10000
9 Thread
sleep 10000
10 Thread
sleep 10000
11 Thread
sleep 10000
12 Thread
sleep 10000
13 Thread
sleep 10000
14 Thread
sleep 10000
15 Thread
sleep 10000
Time Taken: 1601.65225005
160 messages with success status found in response Queue
ubuntu@ip-172-31-0-54:~$


ubuntu@ip-172-31-0-54: ~
En
1:13 PM
ubuntu@ip-172-31-0-54:~$ python gnrtr.py 10 sleep 10000
ubuntu@ip-172-31-0-54:~$ python local.py 1
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
0 Thread
sleep 10000
Time Taken: 100.104266882
10 messages with success status found in response Queue
ubuntu@ip-172-31-0-54:~$


ubuntu@ip-172-31-0-54: ~
En
1:19 PM
0 Thread
sleep 10000
1 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
Time Taken: 200.207867146
20 messages with success status found in response Queue
ubuntu@ip-172-31-0-54:~$


ubuntu@ip-172-31-0-54: ~
1:34 PM
0 Thread
sleep 10000
1 Thread
sleep 10000
2 Thread
sleep 10000
3 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
2 Thread
sleep 10000
3 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
2 Thread
sleep 10000
3 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
2 Thread
sleep 10000
3 Thread
sleep 10000
Time Taken: 400.407751083
40 messages with success status found in response Queue
ubuntu@ip-172-31-0-54:~$


ubuntu@ip-172-31-0-54: ~
1:55 PM
0 Thread
sleep 10000
1 Thread
sleep 10000
2 Thread
sleep 10000
3 Thread
sleep 10000
4 Thread
sleep 10000
5 Thread
sleep 10000
6 Thread
sleep 10000
7 Thread
sleep 10000
0 Thread
sleep 10000
1 Thread
sleep 10000
2 Thread
sleep 10000
3 Thread
sleep 10000
4 Thread
sleep 10000
5 Thread
sleep 10000
6 Thread
sleep 10000
7 Thread
sleep 10000
Time Taken: 800.829897881
80 messages with success status found in response Queue
ubuntu@ip-172-31-0-54:~$