CloudKon Clone with Amazon EC2, S3, SQS and DynamoDB

Design Document

Shalin A. Chopra

5/1/2016

DESIGN DOCUMENT

1. Local Back-end Workers:

- The code for Task Execution Framework using Local Workers is written in Python Programming Language.
- The aim of this code is to implement an in-memory Queue and depict the behaviour of SQS, which will hold the tasks which are to be executed.
- Here the tasks to be executed are “sleep <time in millis>” tasks.
- Here, we have a Client which reads a workload file and every task from this workload file is then put onto the Queue for Workers to fetch.
- As we consider a Local Scenario, both the client and worker are on the same machine, we introduce multiple workers by using threads.
- A thread-pool is being created which acts as multiple workers. Every worker takes a task from the Tasks Queue and executes it. The execution response is noted back in Response Queue which notes 0 for success and 1 for failure. This continues until all the tasks are executed.
- Here for performance evaluation we have threads from 1,2,4,8 and 16 to measure Throughput and Efficiency.

2. Remote Back-end Workers:

- The code for Remote Back-end Workers is also written in Python Programming language.
- In this implementation we need to use AWS services like EC2 Instances (t2.micro) which acts as Client and Workers, SQS which is the Queue for storing the Tasks as Messages and DynamoDB for removing duplicate messages or tasks.
- The tasks to be executed are the same sleep tasks as executed before in Local Worker Scenario.
- The client and workers are on different machines here and the worker keeps on polling to the queue for tasks.
- The Client uses the SQS to store tasks from the workload file and to pick the response logs from the SQS Resonse Queue.
- The job of the worker here is to fetch the tasks from the Queue execute it and write the response to SQS Response Queue, the task from the task queue is deleted by the worker.
- Since, SQS is Asynchronous there might be chances that the same message/task is executed again. To deal with this duplicate issue we introduce DynamoDB.
- The role of DynamoDB here is to avoid duplicate tasks to be executed. It stores the Tasks which are executed.

- We introduce a Key/Value pair format for the Task where each task has its own Key i.e. TaskID and Value i.e. Tasks. When a task is picked for execution it is first checked with dynamoDB if it is present, if present the task is skipped marking it as duplicate, if not present it is added into the table and then executed.
- Every task to be executed has to first go through the dynamodb table to check whether it is already executed or not.
- There might be cases that there aren't many Tasks in the Queue compared to number of workers running. So, to keep the worker in execution it is given a sleep of 1 sec and then asked to poll back to the Queue.
- After the final task is executed, all the workers check the Queue for message/task, a wait time of 10 sec is given to all after which the worker terminates.
- Here for performance evaluation we have Workers (instances) from 1,2,4,8 and 16 to measure Throughput and Efficiency.
- The DynamoDB's Read and Write Throughput capacities are set to 20 and 50 respectively.

3. <u>Animoto:</u>

- The code for Animoto involves the use of S3 as the storage medium for storing the videos that are made using 60 images.
- Here I have used ffmpeg as the tool for conversion of 60 images to 1 minute video in .mkv format.
- Firstly, we have a total of 160 Jobs, each job consisting of 60 image url's. I have stored all the job file name in a Animoto_job.txt file. Every file has 60 urls for the images to be downloaded and converted.
- The SQS is used to store the Jobs.
- Then, we take the images download it using wget, as image names are if longer format we rename them to image<id>.jpg.
- Further, these images are taken and converted to an appropriate format of video.
- After converting the video, this video is then stored in S3.
- For storing in S3 a bucket needs to be created and this bucket needs to have a name which is unique all-over else exception is returned also the bucket name should be entirely lowercase.
- Since we have large videos in MB's we upload them in S3 using FileChunkIO library, which send file in chunks and the merges them together.
- After the video is put onto S3, a URL to the video is sent as a response to the Response Queue.
- Finally after all the 160 videos are made, the Response is written onto a Log file having all the urls. These urls are only valid for 10 minutes after which these URL's become inaccessible.
- There are multiple Workers used from 1,2,4,8 and 16 to make these videos.

Trade-offs made:

- As seen when evaluating for performance, the SQS queue is slow at times, this can be improved by introducing horizontal scaling and sending messages as Batch Messages (Batching).
- There were at times too many Workers and few elements in the Queue, which caused the worker to go slow as they didn't have many tasks to process. This can improved with Dynamic Provisioning with the help of Cloud Watch, release those workers when there isn't huge workload and include more when the workload increases.

MANUAL

Folder Structure:

PROG3_CHOPRA_SHALIN

- ➢ TaskExecutionFramework
  - ▪ Executables
  - ▪ Workloads
  - ▪ ResponseLogs
  - ▪ filechunkio

Executables: Contains all the code and scripts.
Workloads: Contains different workload types for Throughput and Efficiency and Animoto URL's
ResponseLogs: The Log files of the response Queue, all the logs will be stored here.
filechunkio: A python package.

Scripts:

1. LaunchAWSServices.py <service to Launch> it includes (t2m, sqs, dydb, s3, del)
2. GenerateSleepTasks.py it includes code for generating workload for Throughput and Efficiency.
3. Put the AWS Credentials in config.json file

Configurations and Installations:

1. sudo apt-get update
   # for python
2. wget https://repo.continuum.io/miniconda/Miniconda3-latest-Linux-x86_64.sh
3. bash Miniconda3-latest-Linux-x86_64.sh , and restart the terminal
4. sudo apt-get remove --purge ffmpeg **# for Animoto video**
5. sudo apt-add-repository ppa:mc3man/trusty-media
6. sudo apt-get update
7. sudo apt-get install ffmpeg

How to run the code:

1. The Executable folder contains all the code implementation, to launch the t2.micro instance just execute the LaunchAWSServices.py file as follows:

   ```
   python LaunchAWSServices.py t2m
   ```

   this will ask for number of instances to launch and then launch them.
2. For Local worker case,

   ```
   python client.py –s LOCAL –w <Name of Workload File> -t <Threads>
   ```

3. For Remote Worker, Launching SQS, DynamoDB,

   ```
   python LaunchAWSSerivces.py sqs
   python LaunchAWSSerivces.py dydb
   ```

4. For Client,

   ```
   python client_SQS.py –s TasksQueue –w <Name of Workload File>
   ```

5. For Remote-Worker,

   ```
   python RemoteWorker.py –s TasksQueue –t <N: Threads>
   ```

6. To purge SQS and delete DyanmoDB service:

   ```
   python LaunchAWSServices.py del
   ```

7. Animoto Client,

   ```
   python client_SQS.py –s TasksQueue –w Animoto_Jobs.txt
   ```

8. Launch S3,

   ```
   python LaunchAWSServices.py s3
   ```

9. Animoto Worker,

   ```
   python worker_animoto.py –s TasksQueue –t <N: Threads>
   ```

CloudKon Clone with Amazon EC2, SQS, S3 and DynamoDB

Performance Evaluation

Shalin A. Chopra

5/1/2016

PERFORMANCE EVALUATION

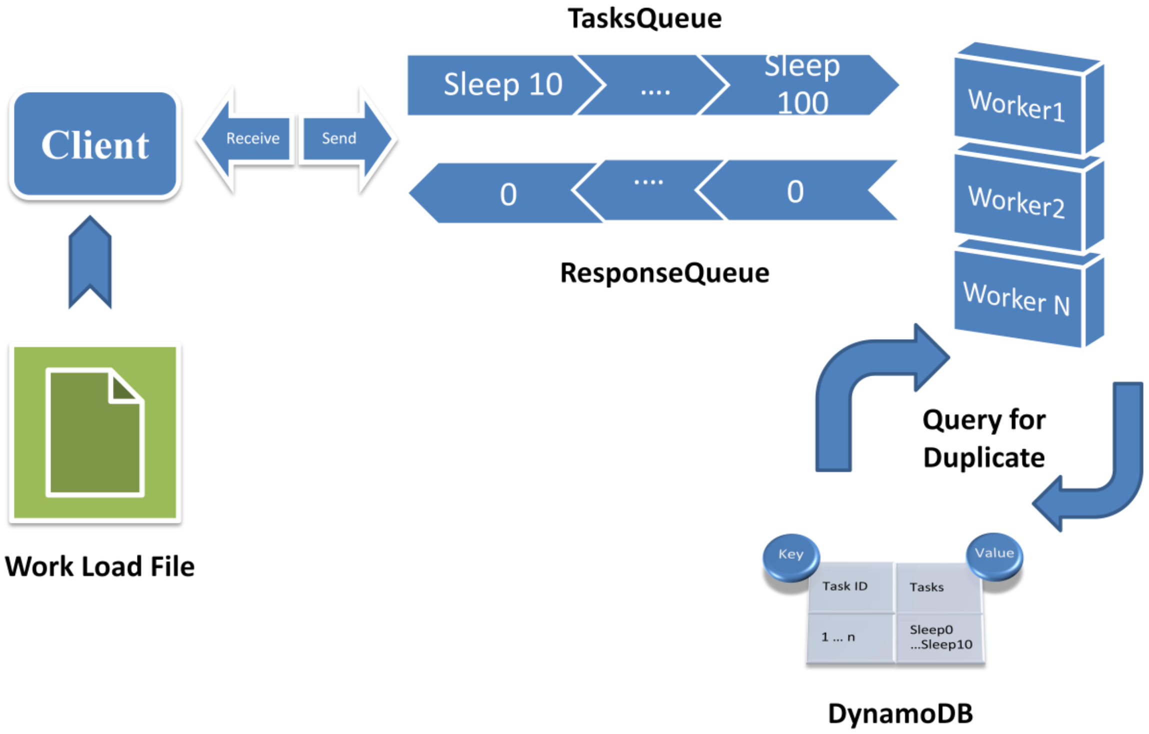
This document presents with the performance evaluation for:

1. **Local Back-end Worker**
2. **Remote Back-end Worker**
3. **Animoto Video**

The diagram below shows the overview of Task Execution Framework for my implementation. It consists of a Client, 2 Queues (SQS/ in-memory), Workers (Remote/ Local) and DynamoDB (Remote Workers).

The client is presented with a workload file which has tasks (sleep tasks, Image lists file location). The client puts the jobs onto the SQS tasks queue. The worker executes the jobs and writes the response back to response queue. DynamoDB interaction is with worker to check for already executed tasks.
There is S3 component involved for storing the videos in case of Animoto Clone for converting images to video.

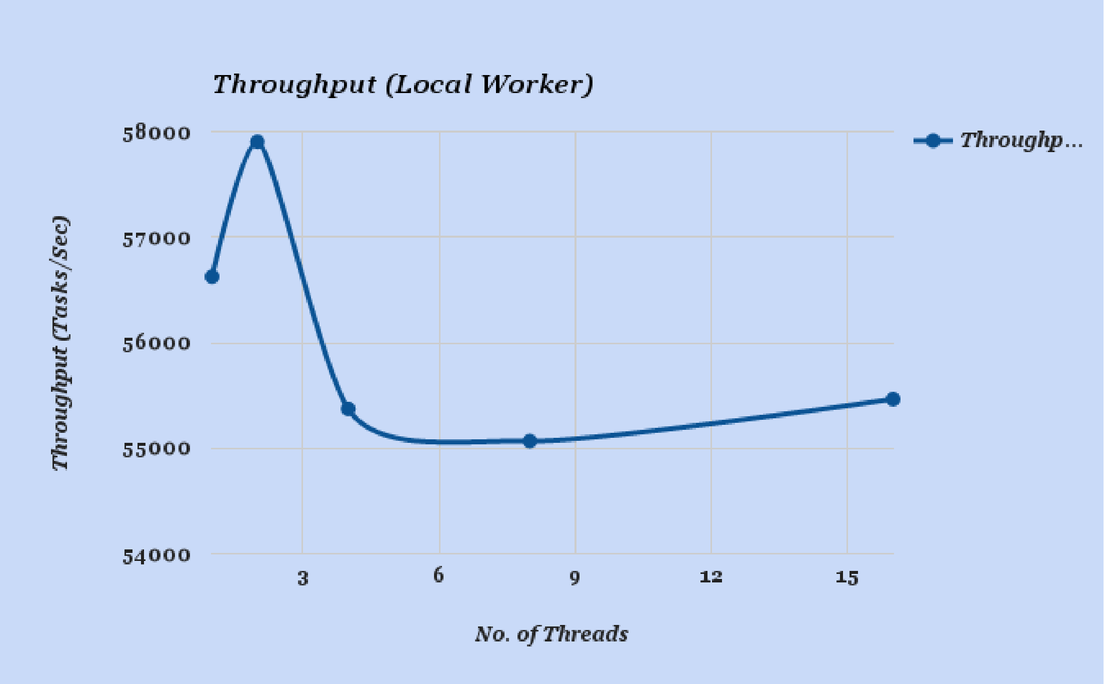
The instance used for evaluation purpose is t2.micro, linux version: ubutnu and region is us-east-1 (N. Virginia)



1. Local Back-end Worker:

The graphs and tables shows the Throughput and Efficiency of the system:

Throughput:



Workers/ Threads	Tasks (sleep 0)	Time (sec)	Throughput (Tasks/sec)
1	1000000	17.66	56625.14
2	1000000	17.27	57903.88
4	1000000	18.06	55370.99
8	1000000	18.16	55066.08
16	1000000	18.03	55463.12

For throughput calculation we have 10^6 operations which are sleep 0. Here, the threads act as local worker who execute these tasks. When we increase the number of threads:

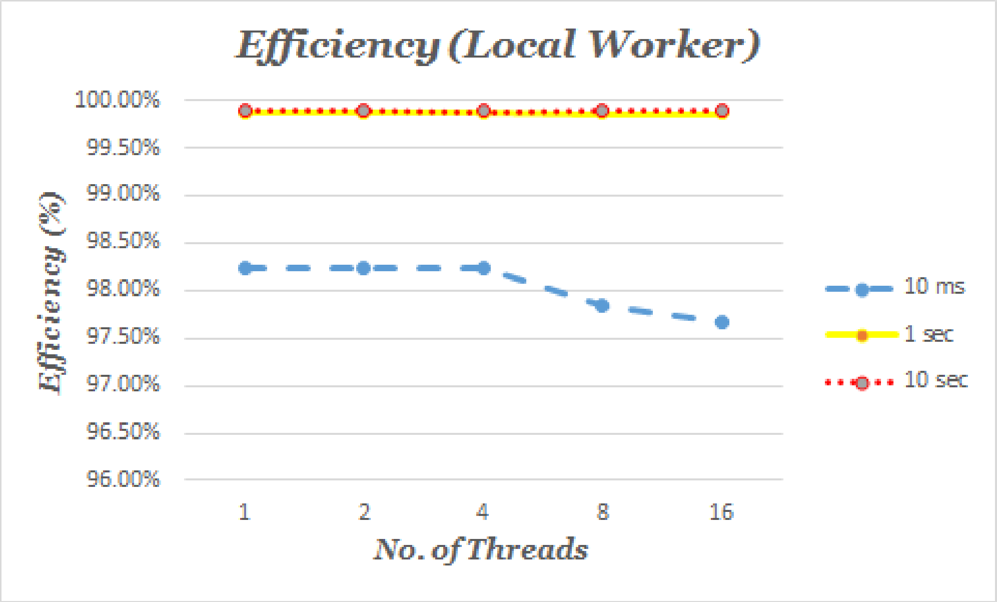
a. Thread = 1, 2

As we increase the number of threads we can observe that the there is an increase in Throughput. Maximum throughput is achieved when thread = 2 i.e. 57903 tasks/sec.

b. Thread = 4, 8, 16

In this case, the throughput decreases when compared to thread 1and 2, this is because now large number of threads are requesting the access to queue and Queue being thread safe, if there are concurrent requests made one might have to wait for receiving the tasks. As we can see from the graph, the throughput tends to increase gradually when increasing the threads from 4 to 16.

Efficiency:



No. of Threads	Task (10 ms)	Task (1 sec)	Task (10 sec)
1	98.23%	99.88%	99.90%
2	98.23%	99.88%	99.90%
4	98.23%	99.88%	99.89%
8	97.85%	99.87%	99.90%
16	97.66%	99.87%	99.90%

For efficiency we have aggregated workload for the threads. We have the following number of tasks for different sleep tasks.

Sleep Time	Number of Tasks per Thread
10 ms	1000
1 sec	100
10 sec	10

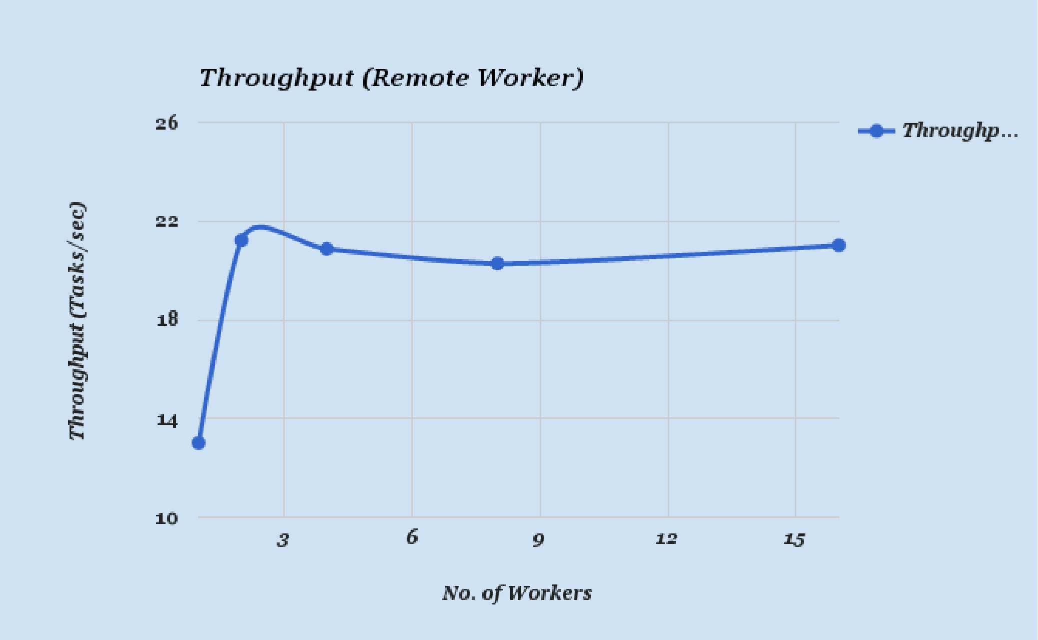
As we can the efficiency for 10 ms is low when compared to efficiency for 1 sec and 10 sec tasks. For 10 ms average efficiency is about 98%, while in case of 1 sec and 10 sec tasks it is about 99.88% and 99.9% respectively.

The efficiency for 10 ms tasks is less because 10 ms is a too small sleep time, which happens too fast and hence there are more concurrent requests to the queue which causes it to slow down. In case of 1 sec and 10 sec, the threads sleep for a significant amount of time which causes them to perform well and give higher efficiency.

2. Remote Back-end Worker:

The graphs and tables shows the Throughput and Efficiency of the system:

Throughput:



Workers	Tasks (sleep 0)	Time (sec)	Throughput (Tasks/sec)
1	10000	768.49	13.01
2	10000	471.32	21.22
4	10000	479.23	20.87
8	10000	493.2	20.28
16	10000	475.81	21.02

For throughput calculation we have 10^4 operations which are sleep 0. Here, the workers act as remote worker each as an instance who executes these tasks. When we increase the number of workers:

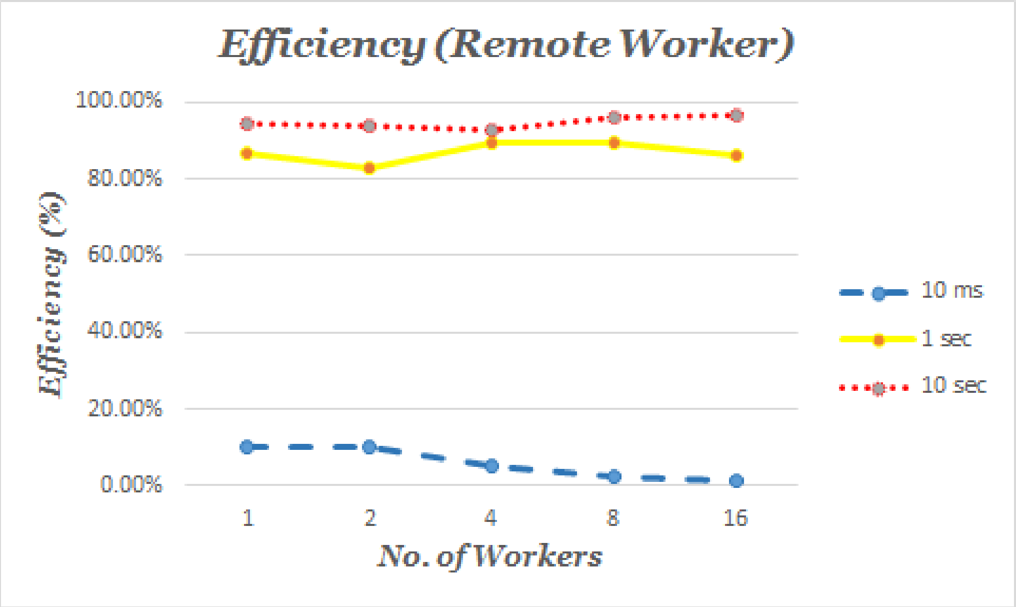
a. Worker = 1, 2

When we increase the number of workers we can depict that the there is an increase in Throughput. Maximum throughput is achieved when worker = 2 i.e. 21.22 tasks/sec. When compared with Local Workers we can see that local worker being in-memory and is thus faster, while in case of remote workers we have Network communication latency with the SQS and DynamoDB. As we have to communicate with DynamoDB for duplicates check and insert into table if not a duplicate value. This takes a lot of time.

b. Worker = 4, 8, 16

In this case, the throughput decreases slightly and then it increases gradually when compared to thread 1and 2, this is because now large number of threads are requesting the access to SQS Queue. Now the same overhead is spread across all the workers. As we can see from the graph, the throughput tends to increase gradually when increasing the threads from 4 to 16.

Efficiency:



Workers	Task (10 ms)	Task (1 sec)	Task (10 sec)
1	10.36%	86.64%	94.30%
2	9.99%	82.95%	93.76%
4	5.19%	89.33%	92.96%
8	2.67%	89.43%	95.81%
16	2.32%	86.16%	96.50%

For efficiency we have aggregated workload for the threads. We have the following number of tasks for different sleep tasks.

Sleep Time	Number of Tasks per Thread
10 ms	1000
1 sec	100
10 sec	10

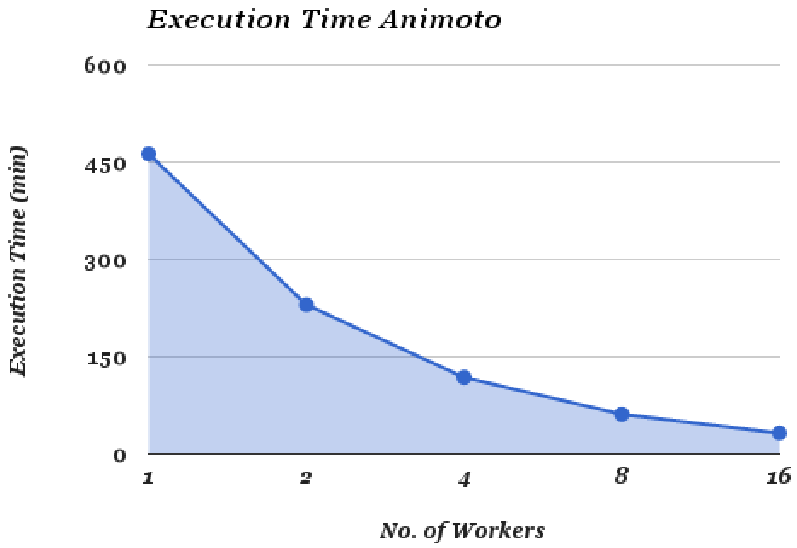
As we can the efficiency for 10 ms is very low when compared to efficiency for 1 sec and 10 sec tasks. For 10 ms average efficiency is about 6%, while in case of 1 sec and 10 sec tasks it is about 85% and 94% respectively.

The efficiency for 10 ms tasks is less because 10 ms is a too small sleep time, which happens too fast and hence there are more concurrent requests to the SQS queue which causes it to slow down. The request in SQS gets completed very fast for large number of workers and they just wait (poll) for more tasks to the Queue which doesn't have many tasks. Half the time is spent polling to the queue as its get empty very fast for 10ms case. Thus, the efficiency is so low. In case of 1 sec and 10 sec tasks, the workers sleep for a significant amount of time which causes them to perform well and give higher efficiency. As the amount of time spent by the workers for sleep helps the SQS queue to load more tasks.

Thus, we can say that for larger sleep tasks the system performs very well in terms of efficiency for all the number of workers from 1 to 16. The efficiency tends to increase gradually.

3. Animoto Clone:

The following graphs and table shows the Animoto Clone Execution time:



Workers	Execution Time (min)
1	463
2	230
4	118
8	61
16	32

I have taken 160 Jobs into consideration each of 60 urls. The images are fetched using wget and are compressed to 45 MB, the video then formed after compression is of 21 MB.

The time taken for downloading images is about 160~175 secs, which takes most of the time for downloads.

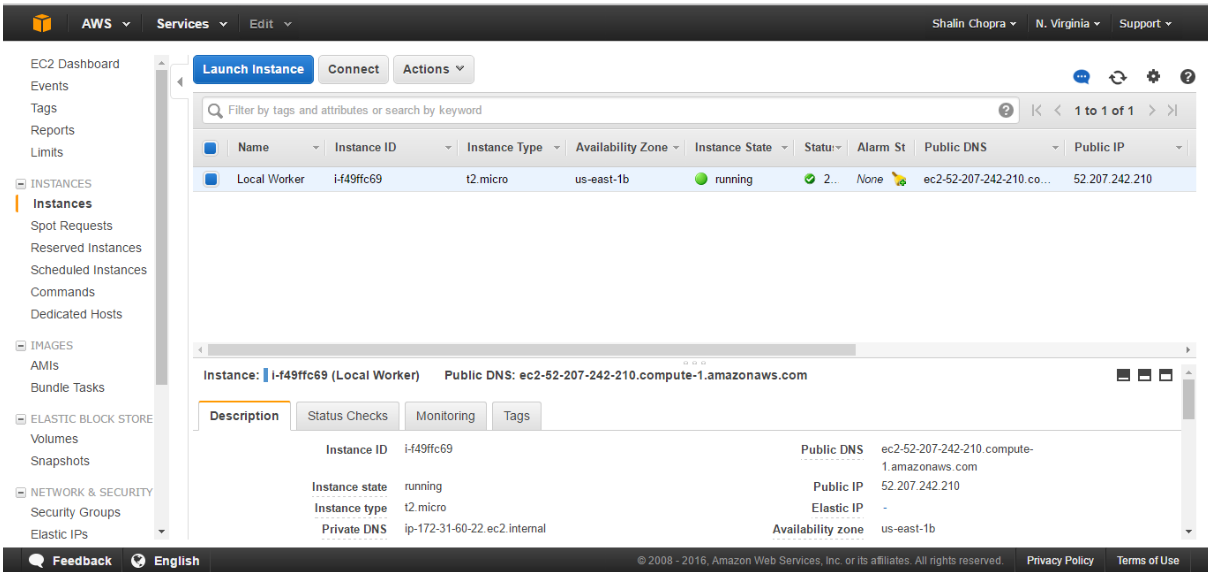
The above graph shows the execution time for 160 videos formation. As we can see that for a single worker to do 160 jobs it takes around 8 hours (including download time, S3 storage) which is a lot for a single worker to do having more load induced onto a single worker. Then we introduce multiple workers for jobs and achieve a faster time in execution as we now have multiple workers simultaneously converting them to videos. Time taken by 16 workers is about 32 minutes.

The videos are stored in s3, and a URL is written in the Logs. The screen shots below will show the steps taken to achieve animoto kind of implementation.

Find attached a video for the same, Animoto_video009.mkv

SCREEN SHOTS:

1. Local Worker Instance:



Name	Instance ID	Instance Type	Availability Zone	Instance State	Statu	Alarm St	Public DNS	Public IP
Local Worker	i-f49ffc69	t2.micro	us-east-1b	running	2..	None	ec2-52-207-242-210.co...	52.207.242.210

Instance: i-f49ffc69 (Local Worker) Public DNS: ec2-52-207-242-210.compute-1.amazonaws.com

Instance ID i-f49ffc69

Instance state running

Instance type t2.micro

Private DNS ip-172-31-60-22.ec2.internal

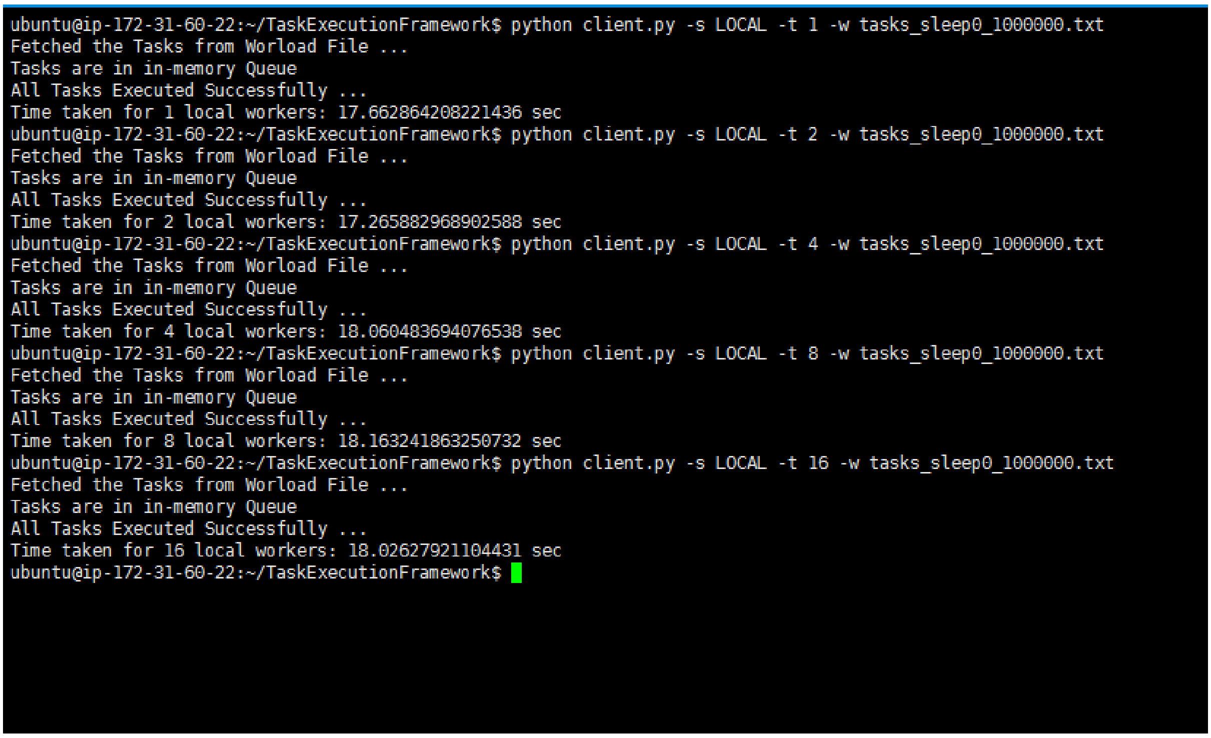
Public DNS ec2-52-207-242-210.compute-1.amazonaws.com

Public IP 52.207.242.210

Elastic IP -

Availability zone us-east-1b

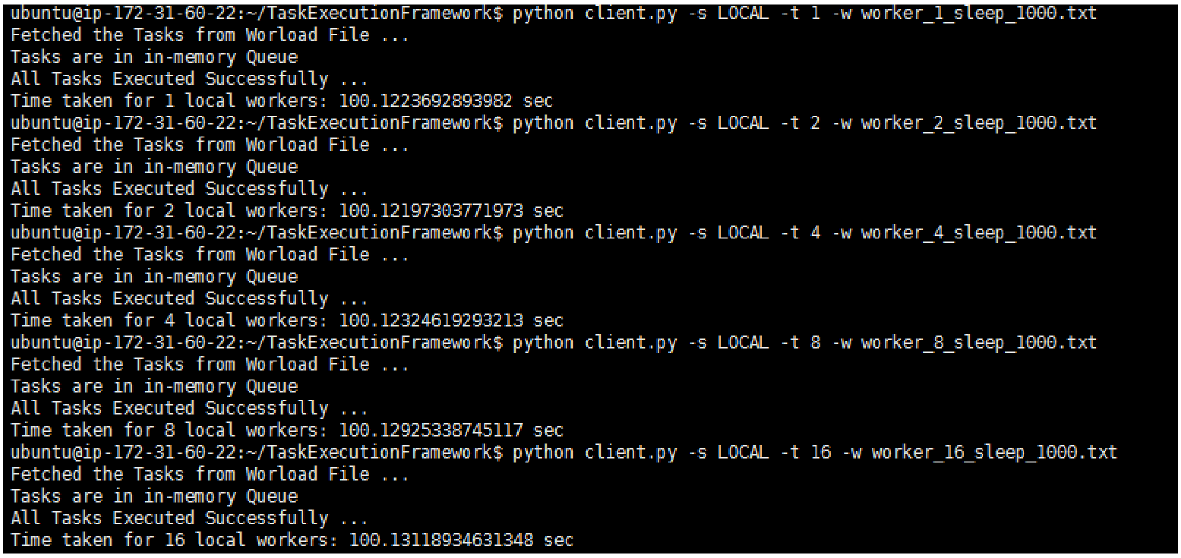
2. Throughput (Sleep 0 Tasks) Evaluations



```
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 1 -w tasks_sleep0_1000000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 1 local workers: 17.662864208221436 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 2 -w tasks_sleep0_1000000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 2 local workers: 17.265882968902588 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 4 -w tasks_sleep0_1000000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 4 local workers: 18.060483694076538 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 8 -w tasks_sleep0_1000000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 8 local workers: 18.163241863250732 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 16 -w tasks_sleep0_1000000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 16 local workers: 18.02627921104431 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$
```

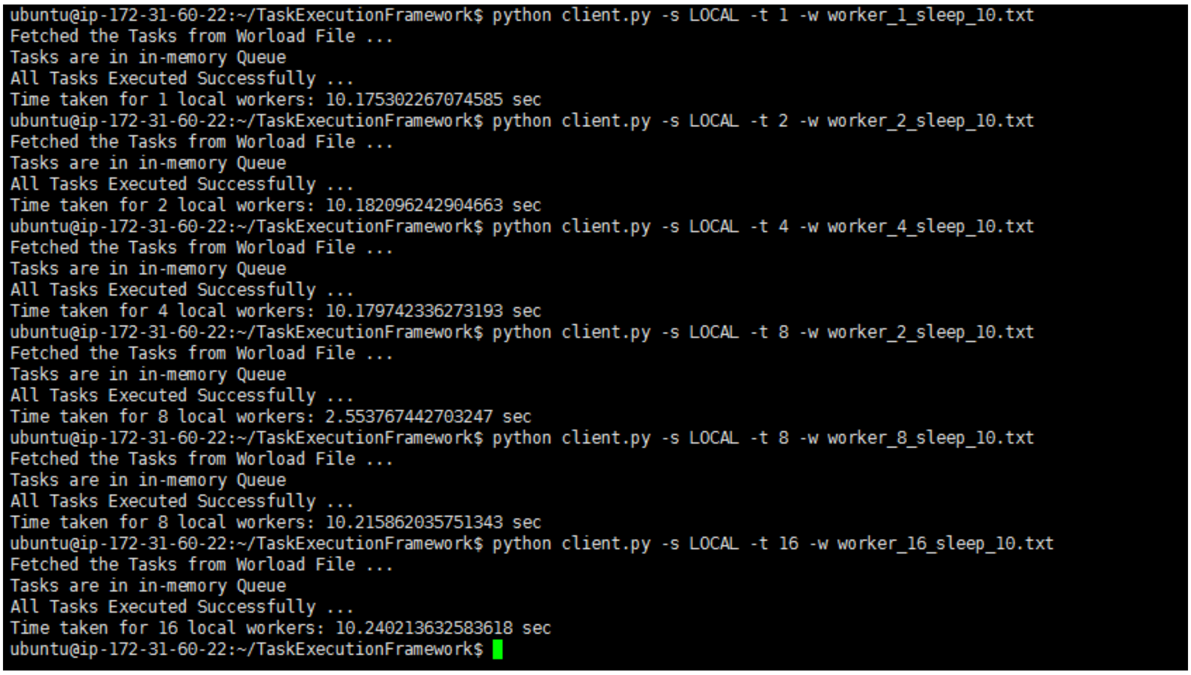
3. Efficiency:

Sleep 1sec:



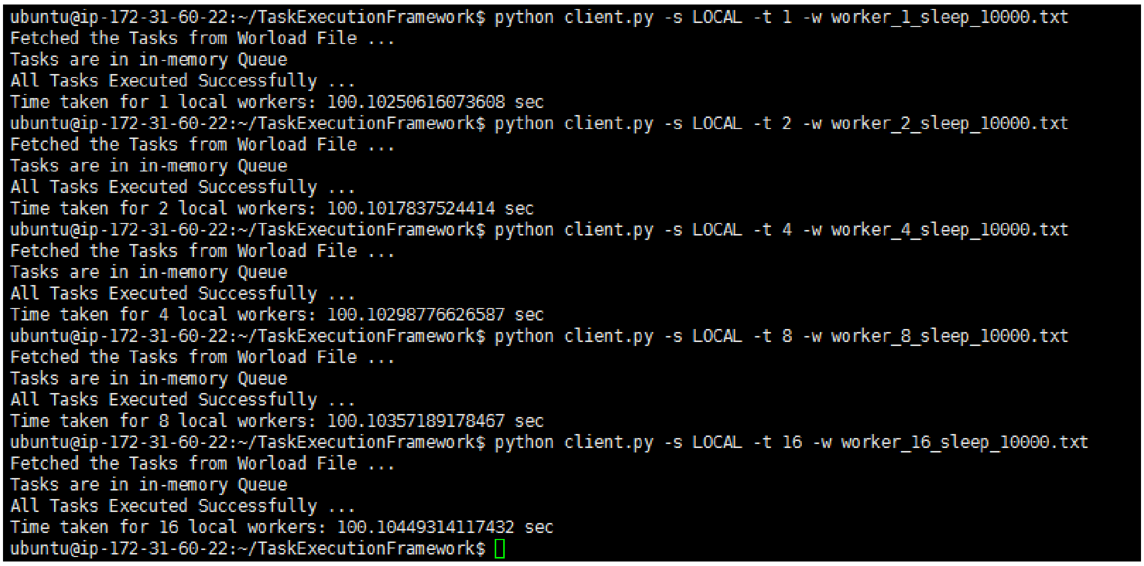
```
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 1 -w worker_1_sleep_1000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 1 local workers: 100.1223692893982 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 2 -w worker_2_sleep_1000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 2 local workers: 100.12197303771973 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 4 -w worker_4_sleep_1000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 4 local workers: 100.12324619293213 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 8 -w worker_8_sleep_1000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 8 local workers: 100.12925338745117 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 16 -w worker_16_sleep_1000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 16 local workers: 100.13118934631348 sec
```

Sleep 10 ms:



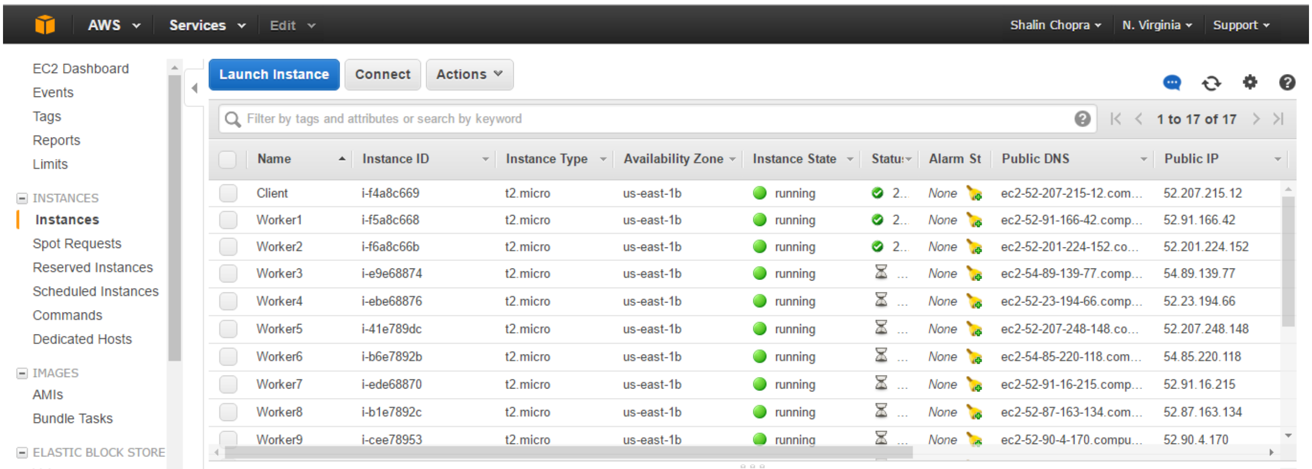
```
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 1 -w worker_1_sleep_10.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 1 local workers: 10.175302267074585 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 2 -w worker_2_sleep_10.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 2 local workers: 10.182096242904663 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 4 -w worker_4_sleep_10.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 4 local workers: 10.179742336273193 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 8 -w worker_2_sleep_10.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 8 local workers: 2.553767442703247 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 8 -w worker_8_sleep_10.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 8 local workers: 10.215862035751343 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 16 -w worker_16_sleep_10.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 16 local workers: 10.240213632583618 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ 
```

Sleep 10 sec:

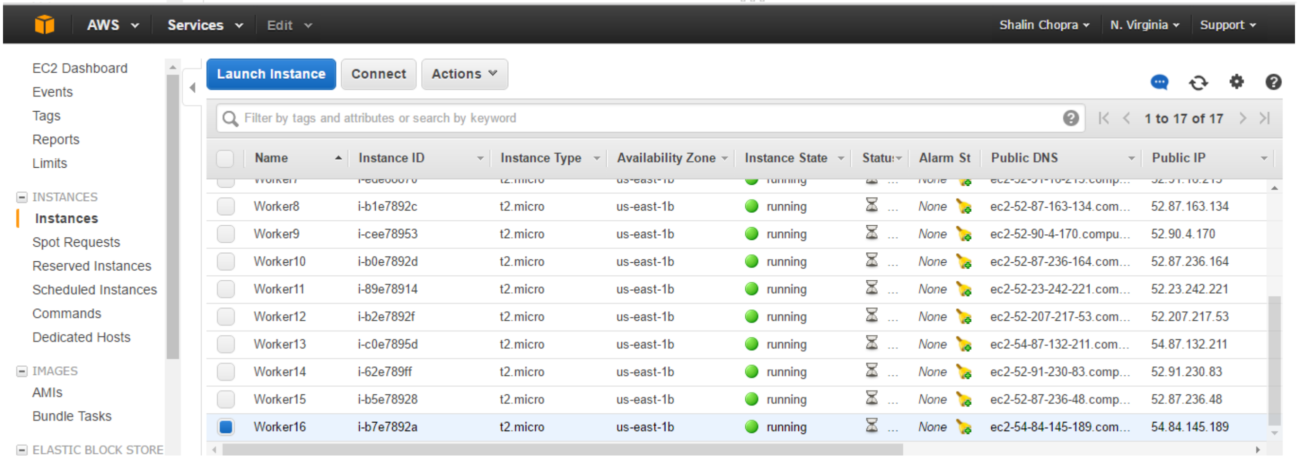


```
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 1 -w worker_1_sleep_10000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 1 local workers: 100.10250616073608 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 2 -w worker_2_sleep_10000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 2 local workers: 100.1017837524414 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 4 -w worker_4_sleep_10000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 4 local workers: 100.10298776626587 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 8 -w worker_8_sleep_10000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 8 local workers: 100.10357189178467 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ python client.py -s LOCAL -t 16 -w worker_16_sleep_10000.txt
Fetched the Tasks from Worload File ...
Tasks are in in-memory Queue
All Tasks Executed Successfully ...
Time taken for 16 local workers: 100.10449314117432 sec
ubuntu@ip-172-31-60-22:~/TaskExecutionFramework$ 
```

4. Remote Worker Instances:



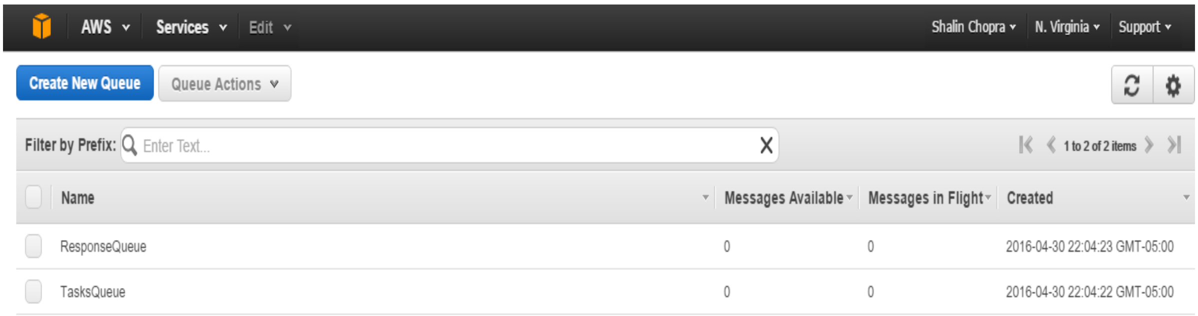
□	Name	Instance ID	Instance Type	Availability Zone	Instance State	Statu:	Alarm St	Public DNS	Public IP
□	Client	i-f4a8c669	t2.micro	us-east-1b	● running	✓ 2...	None	ec2-52-207-215-12.com...	52.207.215.12
□	Worker1	i-f5a8c668	t2.micro	us-east-1b	● running	✓ 2...	None	ec2-52-91-166-42.comp...	52.91.166.42
□	Worker2	i-f6a8c66b	t2.micro	us-east-1b	● running	✓ 2...	None	ec2-52-201-224-152.co...	52.201.224.152
□	Worker3	i-e9e68874	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-54-89-139-77.comp...	54.89.139.77
□	Worker4	i-ebe68876	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-23-194-66.comp...	52.23.194.66
□	Worker5	i-41e789dc	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-207-248-148.co...	52.207.248.148
□	Worker6	i-b6e7892b	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-54-85-220-118.com...	54.85.220.118
□	Worker7	i-ede68870	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-91-16-215.comp...	52.91.16.215
□	Worker8	i-b1e7892c	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-87-163-134.com...	52.87.163.134
□	Worker9	i-cee78953	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-90-4-170.compu...	52.90.4.170



□	Name	Instance ID	Instance Type	Availability Zone	Instance State	Statu:	Alarm St	Public DNS	Public IP
□	Worker8	i-b1e7892c	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-87-163-134.com...	52.87.163.134
□	Worker9	i-cee78953	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-90-4-170.compu...	52.90.4.170
□	Worker10	i-b0e7892d	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-87-236-164.com...	52.87.236.164
□	Worker11	i-89e78914	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-23-242-221.com...	52.23.242.221
□	Worker12	i-b2e7892f	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-207-217-53.com...	52.207.217.53
□	Worker13	i-c0e7895d	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-54-87-132-211.com...	54.87.132.211
□	Worker14	i-62e789ff	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-91-230-83.comp...	52.91.230.83
□	Worker15	i-b5e78928	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-52-87-236-48.comp...	52.87.236.48
■	Worker16	i-b7e7892a	t2.micro	us-east-1b	● running	⌛ ...	None	ec2-54-84-145-189.com...	54.84.145.189

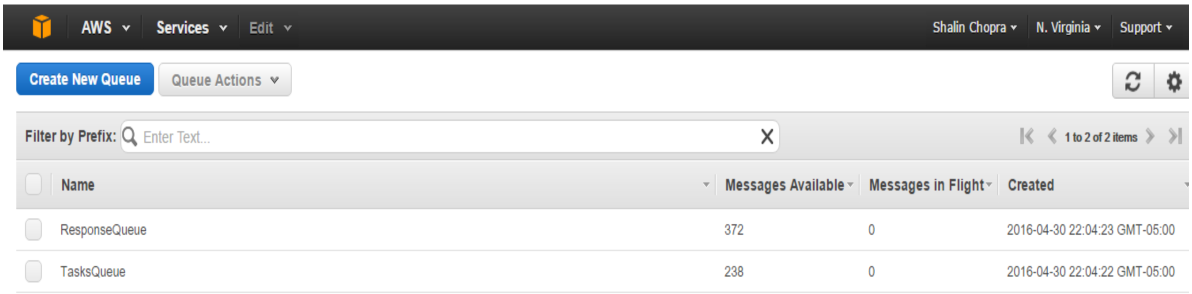
5. SQS

Creation of SQS:



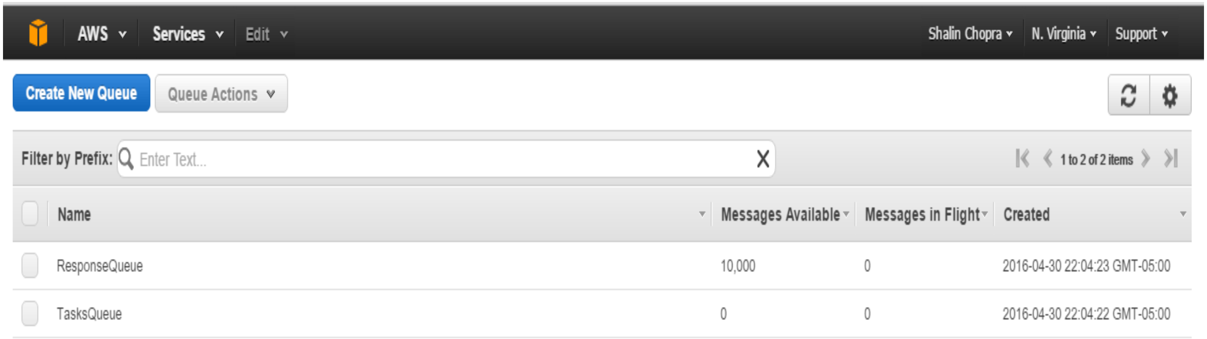
Name	Messages Available	Messages in Flight	Created
ResponseQueue	0	0	2016-04-30 22:04:23 GMT-05:00
TasksQueue	0	0	2016-04-30 22:04:22 GMT-05:00

Message/Tasks/Job in Transit:



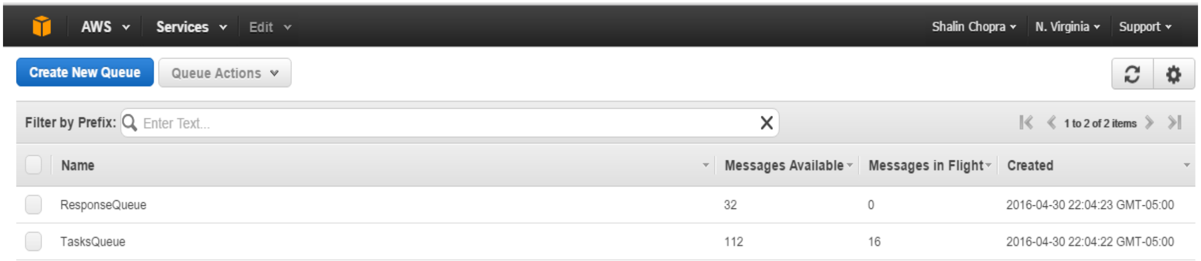
Name	Messages Available	Messages in Flight	Created
ResponseQueue	372	0	2016-04-30 22:04:23 GMT-05:00
TasksQueue	238	0	2016-04-30 22:04:22 GMT-05:00

Completion of Tasks from TasksQueue all Responses Stored in Response Queue:



Name	Messages Available	Messages in Flight	Created
ResponseQueue	10,000	0	2016-04-30 22:04:23 GMT-05:00
TasksQueue	0	0	2016-04-30 22:04:22 GMT-05:00

Tasks/Message in transit (ex. 16 workers):

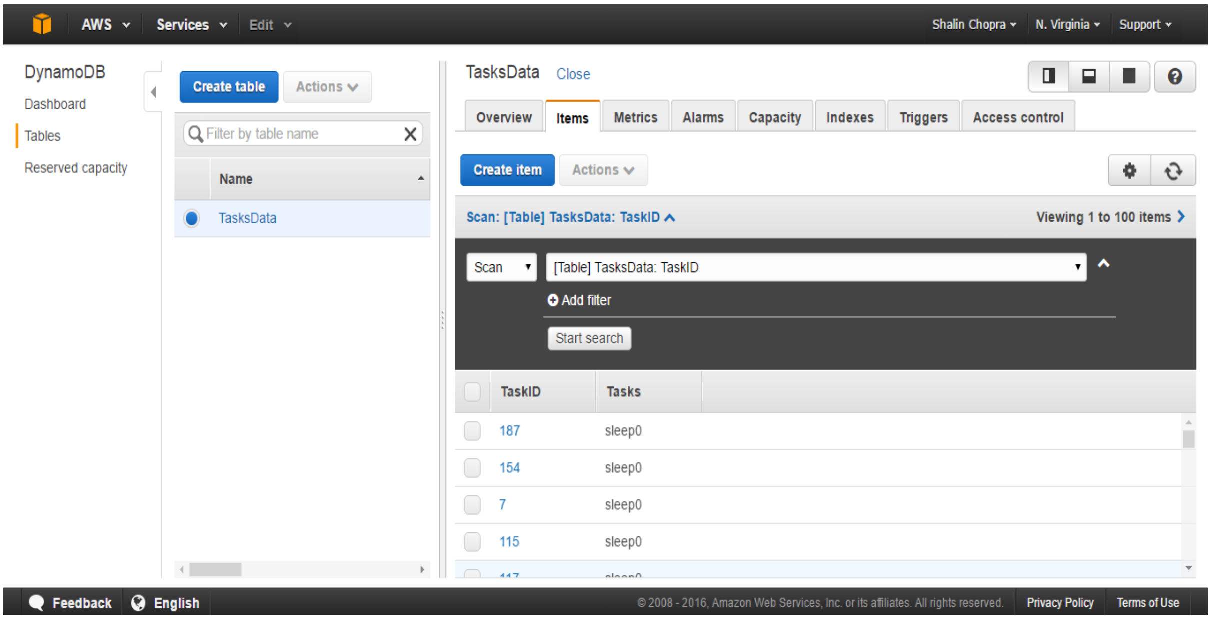


Name	Messages Available	Messages in Flight	Created
ResponseQueue	32	0	2016-04-30 22:04:23 GMT-05:00
TasksQueue	112	16	2016-04-30 22:04:22 GMT-05:00

6. DynamoDB

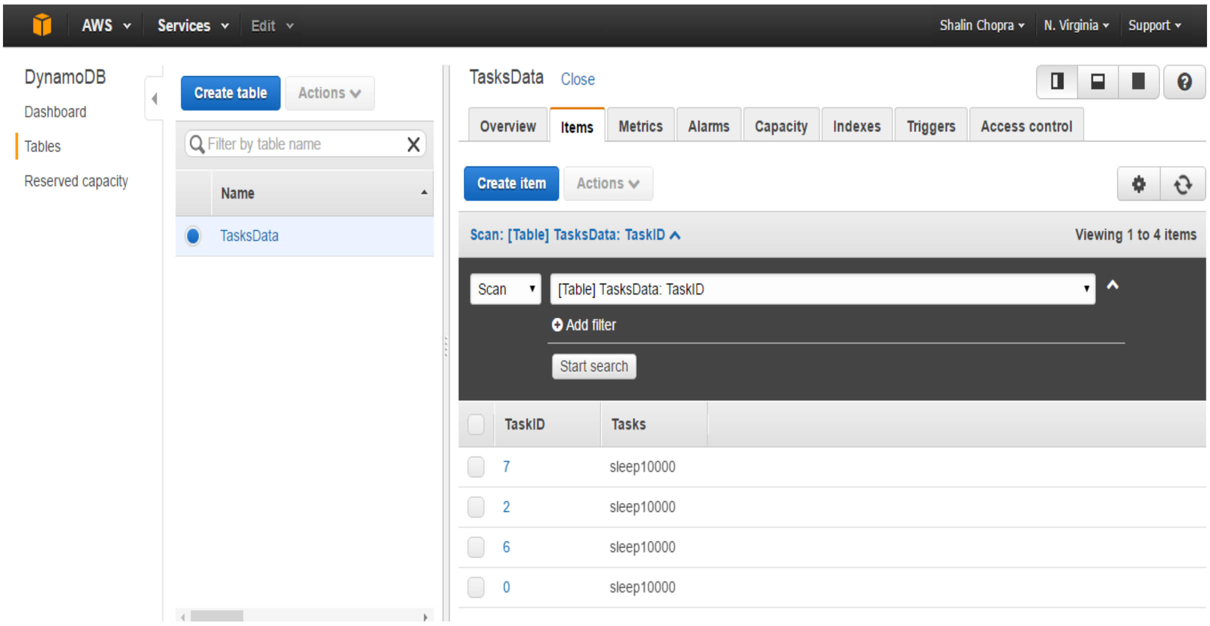
Creation of Table and insertion of tasks:

For Throughput:



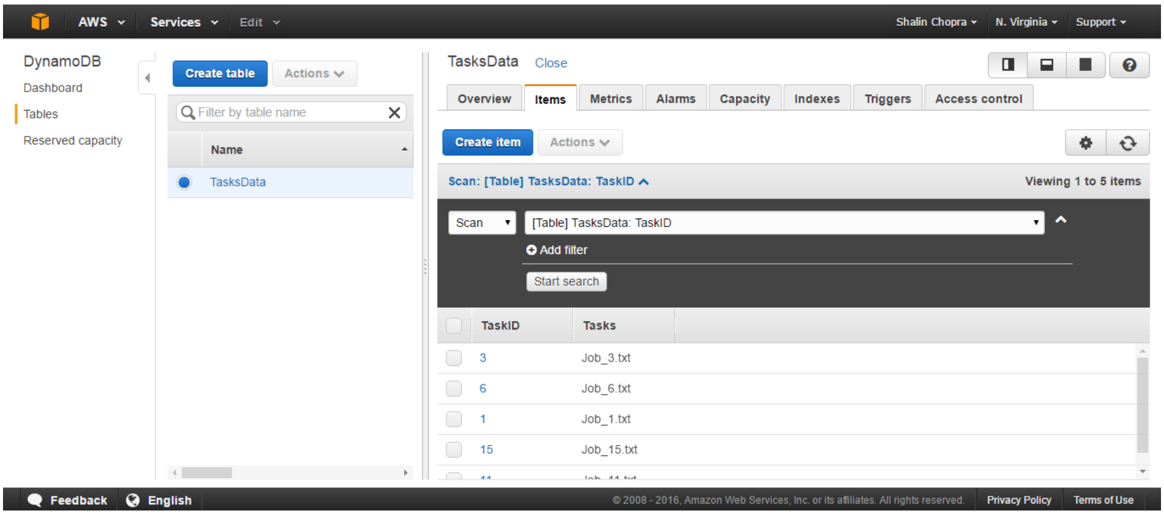
TaskID	Tasks
187	sleep0
154	sleep0
7	sleep0
115	sleep0

For Efficiency (Sleep 10 sec):



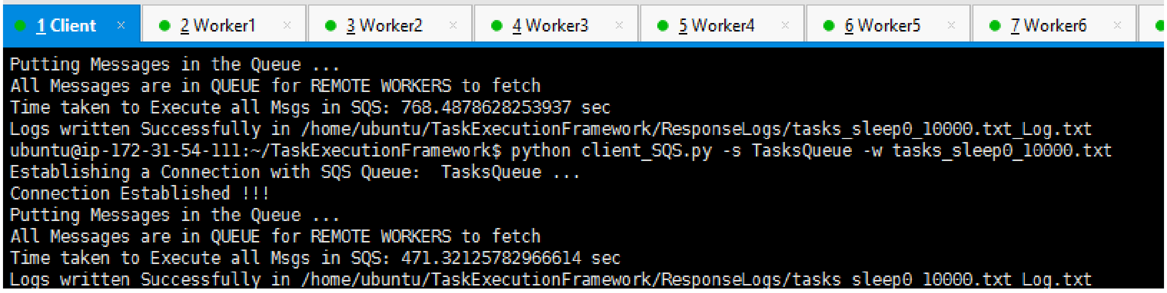
TaskID	Tasks
7	sleep10000
2	sleep10000
6	sleep10000
0	sleep10000

For Animoto Jobs (in Execution DynamoDB):

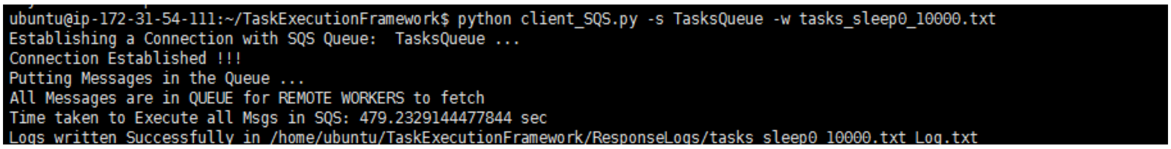


TaskID	Tasks
3	Job_3.txt
6	Job_6.txt
1	Job_1.txt
15	Job_15.txt
11	Job_11.txt

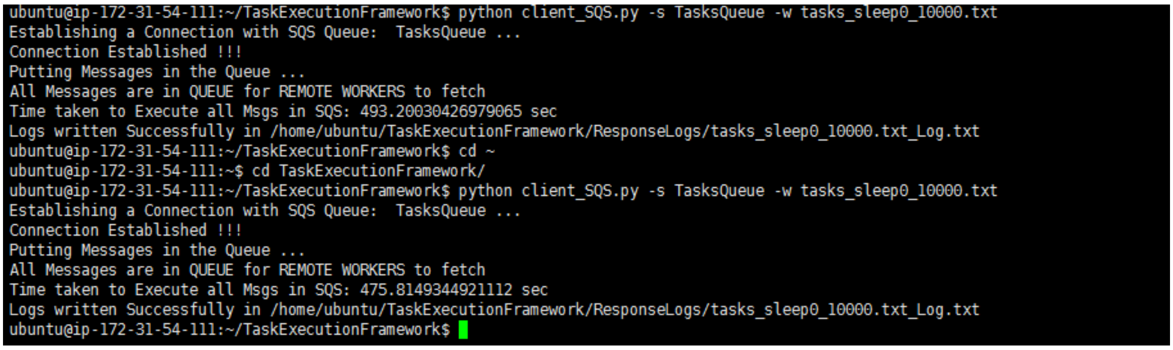
7. Throughput Remote Workers (for 1 to 16 Workers):



```
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 768.4878628253937 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/tasks_sleep0_10000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w tasks_sleep0_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 471.32125782966614 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/tasks_sleep0_10000.txt_Log.txt
```

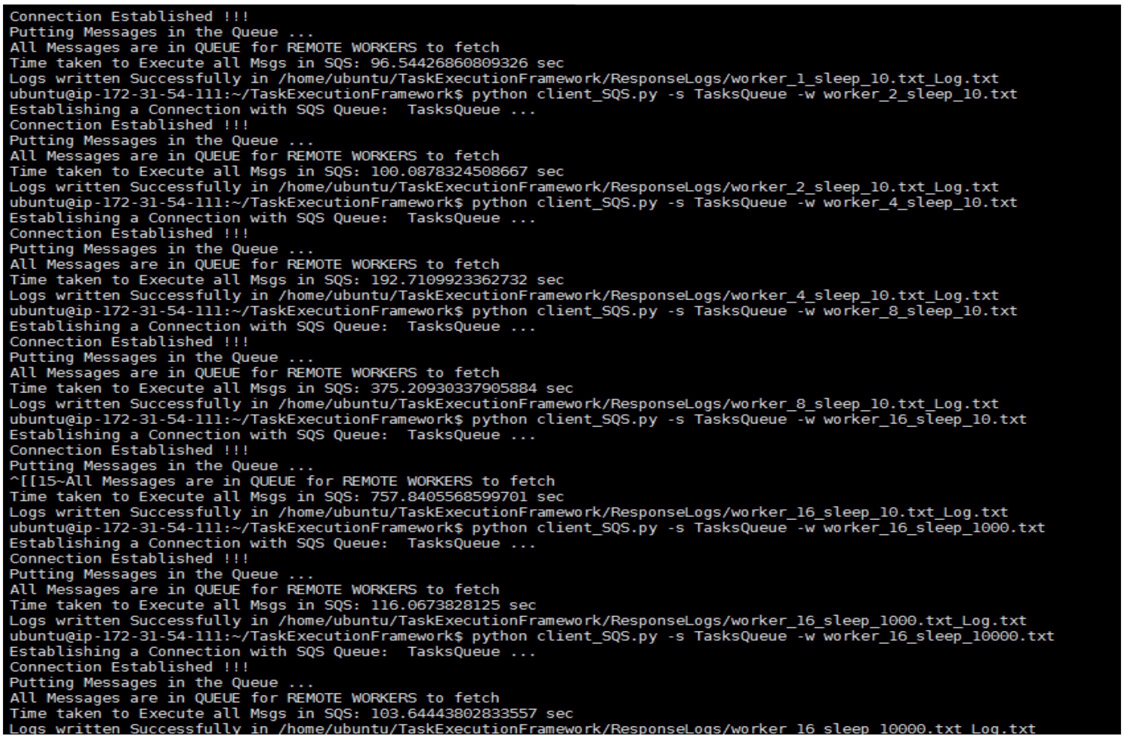


```
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w tasks_sleep0_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 479.2329144477844 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/tasks_sleep0_10000.txt_Log.txt
```

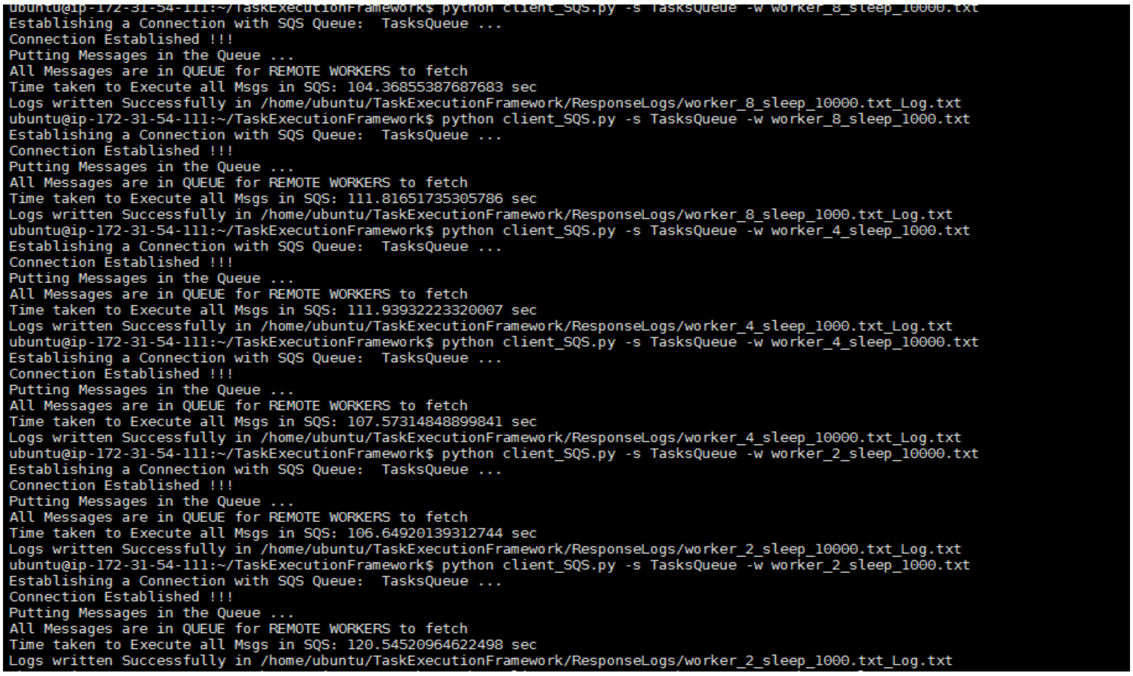


```
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w tasks_sleep0_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 493.20030426979065 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/tasks_sleep0_10000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ cd ~
ubuntu@ip-172-31-54-111:~$ cd TaskExecutionFramework/
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w tasks_sleep0_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 475.8149344921112 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/tasks_sleep0_10000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ 
```

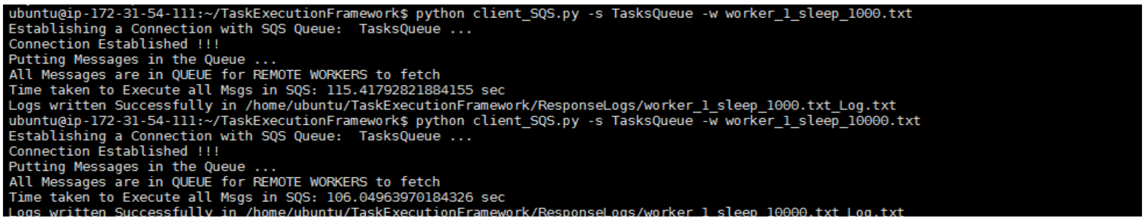
8. Efficiency:



```
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 96.54426860809326 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_1_sleep_10.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_2_sleep_10.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 100.0878324508667 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_2_sleep_10.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_4_sleep_10.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 192.7109923362732 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_4_sleep_10.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_8_sleep_10.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 375.20930337905884 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_8_sleep_10.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_16_sleep_10.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
^[[15~All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 757.8405568599701 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_16_sleep_10.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_16_sleep_1000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 116.0673828125 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_16_sleep_1000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_16_sleep_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 103.64443802833557 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_16_sleep_10000.txt_Log.txt
```

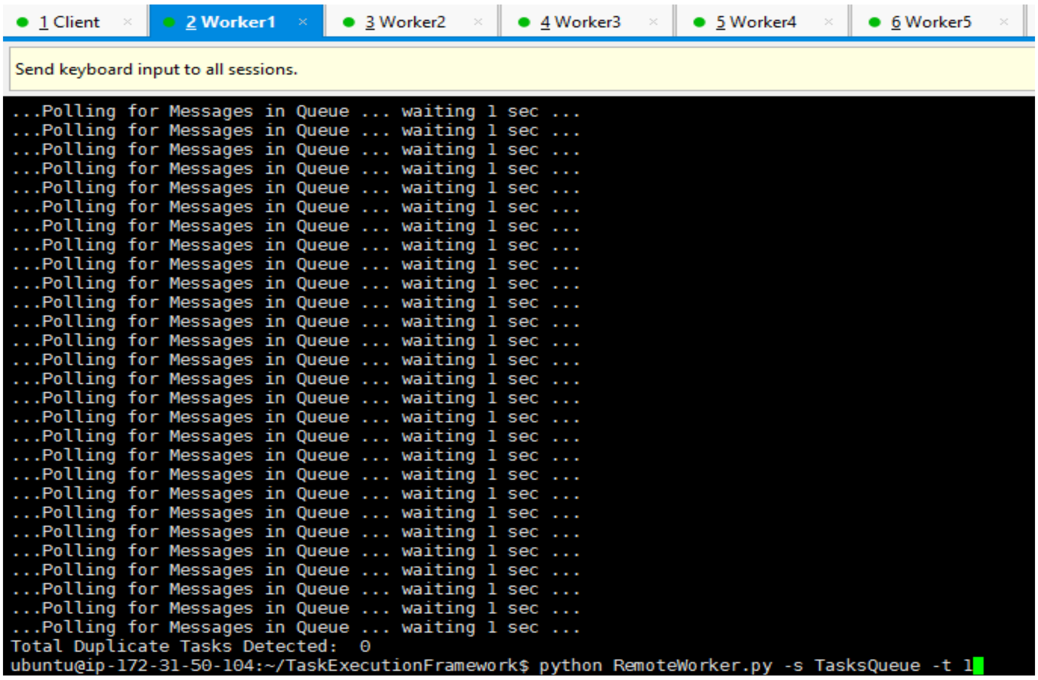


```
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_8_sleep_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 104.36855387687683 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_8_sleep_10000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_8_sleep_1000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 111.81651735305786 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_8_sleep_1000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_4_sleep_1000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 111.93932223320007 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_4_sleep_1000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_4_sleep_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 107.57314848899841 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_4_sleep_10000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_2_sleep_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 106.64920139312744 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_2_sleep_10000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_2_sleep_1000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 120.54520964622498 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_2_sleep_1000.txt_Log.txt
```



```
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_1_sleep_1000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 115.41792821884155 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_1_sleep_1000.txt_Log.txt
ubuntu@ip-172-31-54-111:~/TaskExecutionFramework$ python client_SQS.py -s TasksQueue -w worker_1_sleep_10000.txt
Establishing a Connection with SQS Queue:  TasksQueue ...
Connection Established !!!
Putting Messages in the Queue ...
All Messages are in QUEUE for REMOTE WORKERS to fetch
Time taken to Execute all Msgs in SQS: 106.04963970184326 sec
Logs written Successfully in /home/ubuntu/TaskExecutionFramework/ResponseLogs/worker_1_sleep_10000.txt_Log.txt
```

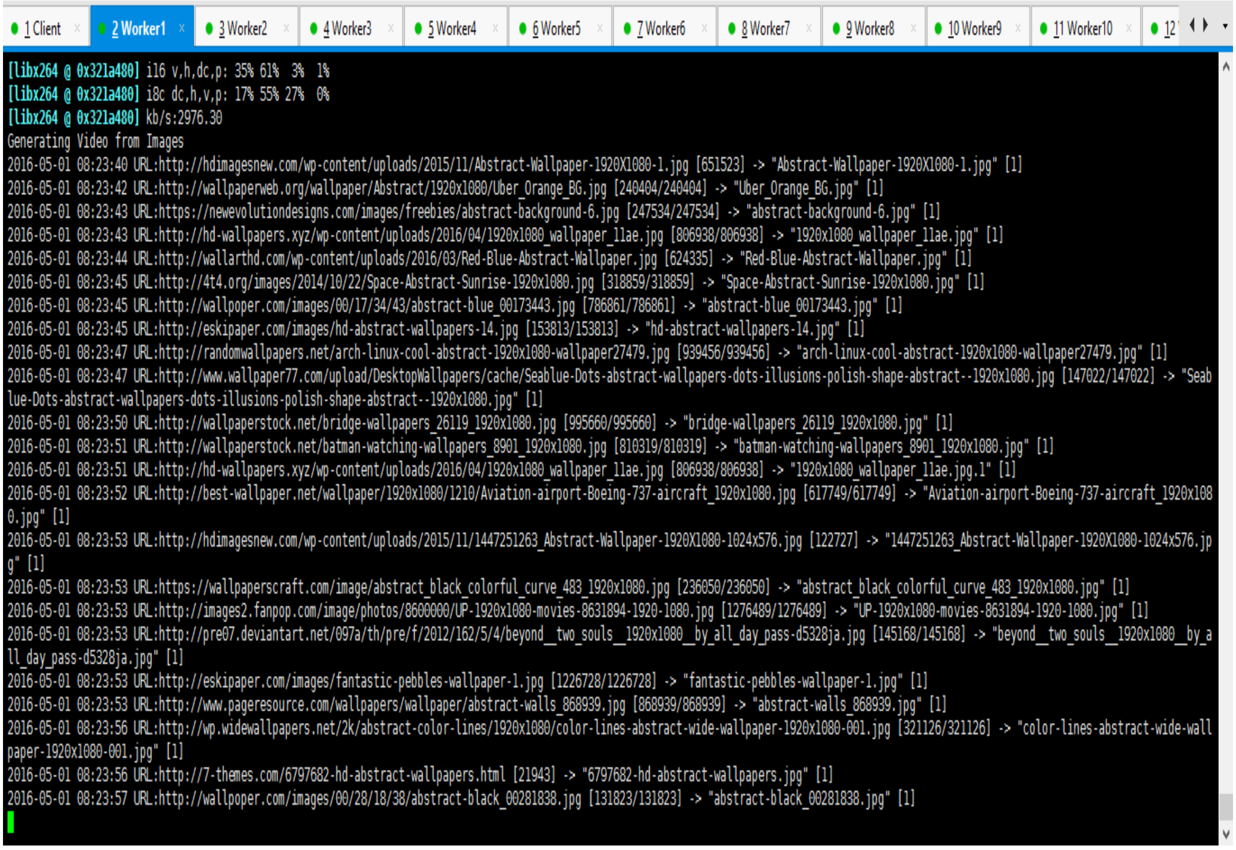
Polling of messages:



```
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
...Polling for Messages in Queue ... waiting 1 sec ...
Total Duplicate Tasks Detected:  0
ubuntu@ip-172-31-50-104:~/TaskExecutionFramework$ python RemoteWorker.py -s TasksQueue -t 1
```

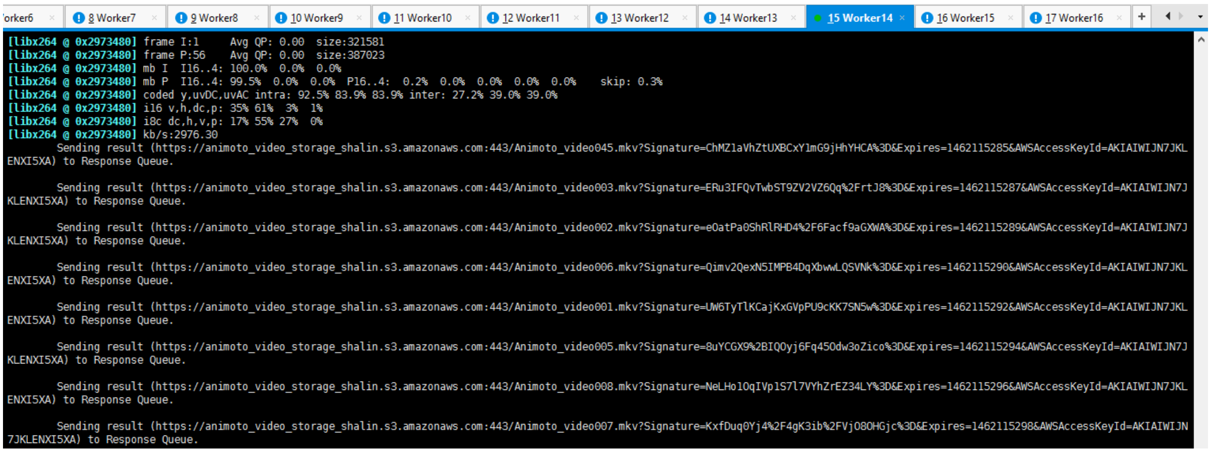
9. Animoto Clone:

Downloading Images:



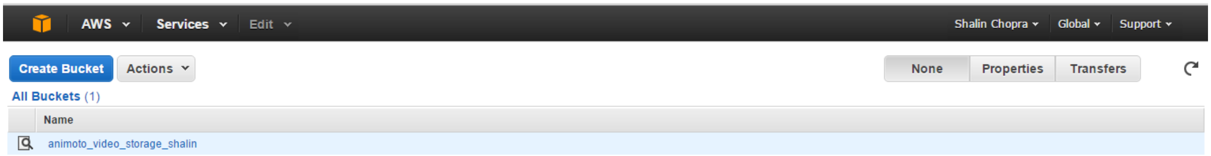
```
[libx264 @ 0x321a480] i16 v,h,dc,p: 35% 61%  3%  1%
[libx264 @ 0x321a480] i8c dc,h,v,p: 17% 55% 27%  0%
[libx264 @ 0x321a480] kb/s:2976.30
Generating Video from Images
2016-05-01 08:23:40 URL:http://hdimagesnew.com/wp-content/uploads/2015/11/Abstract-Wallpaper-1920X1080-1.jpg [651523] -> "Abstract-Wallpaper-1920X1080-1.jpg" [1]
2016-05-01 08:23:42 URL:http://wallpaperweb.org/wallpaper/Abstract/1920x1080/Uber_Orange_BG.jpg [240404/240404] -> "Uber_Orange_BG.jpg" [1]
2016-05-01 08:23:43 URL:https://newevolutiondesigns.com/images/freebies/abstract-background-6.jpg [247534/247534] -> "abstract-background-6.jpg" [1]
2016-05-01 08:23:43 URL:http://hd-wallpapers.xyz/wp-content/uploads/2016/04/1920x1080_wallpaper_11ae.jpg [806938/806938] -> "1920x1080_wallpaper_11ae.jpg" [1]
2016-05-01 08:23:44 URL:http://wallarthd.com/wp-content/uploads/2016/03/Red-Blue-Abstract-Wallpaper.jpg [624335] -> "Red-Blue-Abstract-Wallpaper.jpg" [1]
2016-05-01 08:23:45 URL:http://4t4.org/images/2014/10/22/Space-Abstract-Sunrise-1920x1080.jpg [318859/318859] -> "Space-Abstract-Sunrise-1920x1080.jpg" [1]
2016-05-01 08:23:45 URL:http://wallpoper.com/images/00/17/34/43/abstract-blue_00173443.jpg [786861/786861] -> "abstract-blue_00173443.jpg" [1]
2016-05-01 08:23:45 URL:http://eskipaper.com/images/hd-abstract-wallpapers-14.jpg [153813/153813] -> "hd-abstract-wallpapers-14.jpg" [1]
2016-05-01 08:23:47 URL:http://randomwallpapers.net/arch-linux-cool-abstract-1920x1080-wallpaper27479.jpg [939456/939456] -> "arch-linux-cool-abstract-1920x1080-wallpaper27479.jpg" [1]
2016-05-01 08:23:47 URL:http://www.wallpaper77.com/upload/DesktopWallpapers/cache/Seablue-Dots-abstract-wallpapers-dots-illusions-polish-shape-abstract--1920x1080.jpg [147022/147022] -> "Seab
lue-Dots-abstract-wallpapers-dots-illusions-polish-shape-abstract--1920x1080.jpg" [1]
2016-05-01 08:23:50 URL:http://wallpaperstock.net/bridge-wallpapers_26119_1920x1080.jpg [995660/995660] -> "bridge-wallpapers_26119_1920x1080.jpg" [1]
2016-05-01 08:23:51 URL:http://wallpaperstock.net/batman-watching-wallpapers_8901_1920x1080.jpg [810319/810319] -> "batman-watching-wallpapers_8901_1920x1080.jpg" [1]
2016-05-01 08:23:51 URL:http://hd-wallpapers.xyz/wp-content/uploads/2016/04/1920x1080_wallpaper_11ae.jpg [806938/806938] -> "1920x1080_wallpaper_11ae.jpg.1" [1]
2016-05-01 08:23:52 URL:http://best-wallpaper.net/wallpaper/1920x1080/1210/Aviation-airport-Boeing-737-aircraft_1920x1080.jpg [617749/617749] -> "Aviation-airport-Boeing-737-aircraft_1920x108
0.jpg" [1]
2016-05-01 08:23:53 URL:http://hdimagesnew.com/wp-content/uploads/2015/11/1447251263_Abstract-Wallpaper-1920X1080-1024x576.jpg [122727] -> "1447251263_Abstract-Wallpaper-1920X1080-1024x576.jp
g" [1]
2016-05-01 08:23:53 URL:https://wallpaperscraft.com/image/abstract_black_colorful_curve_483_1920x1080.jpg [236050/236050] -> "abstract_black_colorful_curve_483_1920x1080.jpg" [1]
2016-05-01 08:23:53 URL:http://images2.fanpop.com/image/photos/8600000/UP-1920x1080-movies-8631894-1920-1080.jpg [1276489/1276489] -> "UP-1920x1080-movies-8631894-1920-1080.jpg" [1]
2016-05-01 08:23:53 URL:http://pre07.deviantart.net/097a/th/pre/f/2012/162/5/4/beyond__two_souls__1920x1080__by_all_day_pass-d5328ja.jpg [145168/145168] -> "beyond__two_souls__1920x1080__by_a
ll_day_pass-d5328ja.jpg" [1]
2016-05-01 08:23:53 URL:http://eskipaper.com/images/fantastic-pebbles-wallpaper-1.jpg [1226728/1226728] -> "fantastic-pebbles-wallpaper-1.jpg" [1]
2016-05-01 08:23:53 URL:http://www.pageresource.com/wallpapers/wallpaper/abstract-walls_868939.jpg [868939/868939] -> "abstract-walls_868939.jpg" [1]
2016-05-01 08:23:56 URL:http://wp.widewallpapers.net/2k/abstract-color-lines/1920x1080/color-lines-abstract-wide-wallpaper-1920x1080-001.jpg [321126/321126] -> "color-lines-abstract-wide-wall
paper-1920x1080-001.jpg" [1]
2016-05-01 08:23:56 URL:http://7-themes.com/6797682-hd-abstract-wallpapers.html [21943] -> "6797682-hd-abstract-wallpapers.jpg" [1]
2016-05-01 08:23:57 URL:http://wallpoper.com/images/00/28/18/38/abstract-black_00281838.jpg [131823/131823] -> "abstract-black_00281838.jpg" [1]
```

Worker sending URL to response queue:



```
[libx264 @ 0x2973480] frame I:1     Avg QP: 0.00  size:321581
[libx264 @ 0x2973480] frame P:56    Avg QP: 0.00  size:387023
[libx264 @ 0x2973480] mb I  I16..4: 100.0%  0.0%  0.0%
[libx264 @ 0x2973480] mb P  I16..4: 99.5%  0.0%  0.0%  P16..4:  0.2%  0.0%  0.0%  0.0%  0.0%    skip: 0.3%
[libx264 @ 0x2973480] coded y,uvDC,uvAC intra: 92.5% 83.9% 83.9% inter: 27.2% 39.0% 39.0%
[libx264 @ 0x2973480] i16 v,h,dc,p: 35% 61%  3%  1%
[libx264 @ 0x2973480] i8c dc,h,v,p: 17% 55% 27%  0%
[libx264 @ 0x2973480] kb/s:2976.30
        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video045.mkv?Signature=ChMZ1aVhZtUXBCxY1mG9jHhYHCA%3D&Expires=1462115285&AWSAccessKeyId=AKIAIWIJN7JKL
ENXI5XA) to Response Queue.

        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video003.mkv?Signature=ERu3IFQvTwbST9ZV2VZ6Qq%2FrtJ8%3D&Expires=1462115287&AWSAccessKeyId=AKIAIWIJN7J
KLENXI5XA) to Response Queue.

        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video002.mkv?Signature=eOatPa0ShRlRHD4%2F6Facf9aGXWA%3D&Expires=1462115289&AWSAccessKeyId=AKIAIWIJN7J
KLENXI5XA) to Response Queue.

        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video006.mkv?Signature=Qimv2QexN5IMPB4DqXbwwLQSVNk%3D&Expires=1462115290&AWSAccessKeyId=AKIAIWIJN7JKL
ENXI5XA) to Response Queue.

        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video001.mkv?Signature=UW6TyTlKCajKxGVpPU9cKK7SN5w%3D&Expires=1462115292&AWSAccessKeyId=AKIAIWIJN7JKL
ENXI5XA) to Response Queue.

        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video005.mkv?Signature=8uYCGX9%2BIQOyj6Fq45Odw3oZico%3D&Expires=1462115294&AWSAccessKeyId=AKIAIWIJN7J
KLENXI5XA) to Response Queue.

        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video008.mkv?Signature=NeLHo1OqIVp1S7l7VYhZrEZ34LY%3D&Expires=1462115296&AWSAccessKeyId=AKIAIWIJN7JKL
ENXI5XA) to Response Queue.

        Sending result (https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video007.mkv?Signature=KxfDuq0Yj4%2F4gK3ib%2FVjO8OHGjc%3D&Expires=1462115298&AWSAccessKeyId=AKIAIWIJN
7JKLENXI5XA) to Response Queue.
```

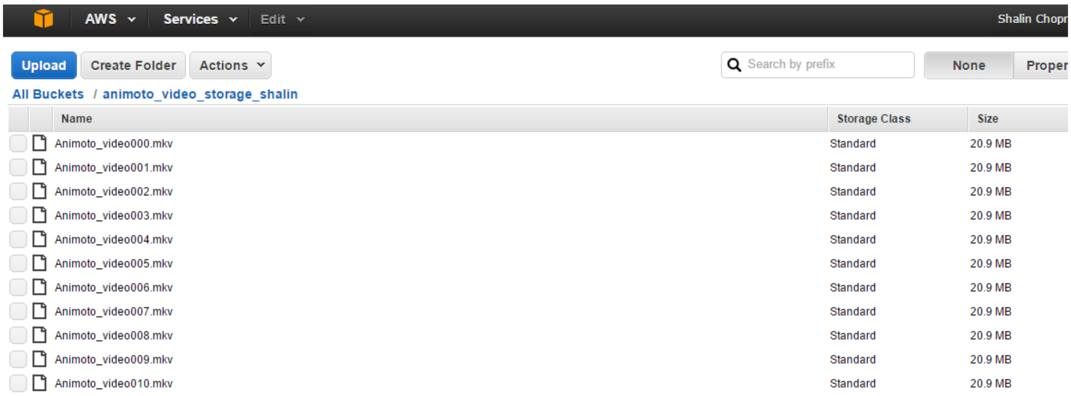
Animoto Bucket:



All Buckets (1)

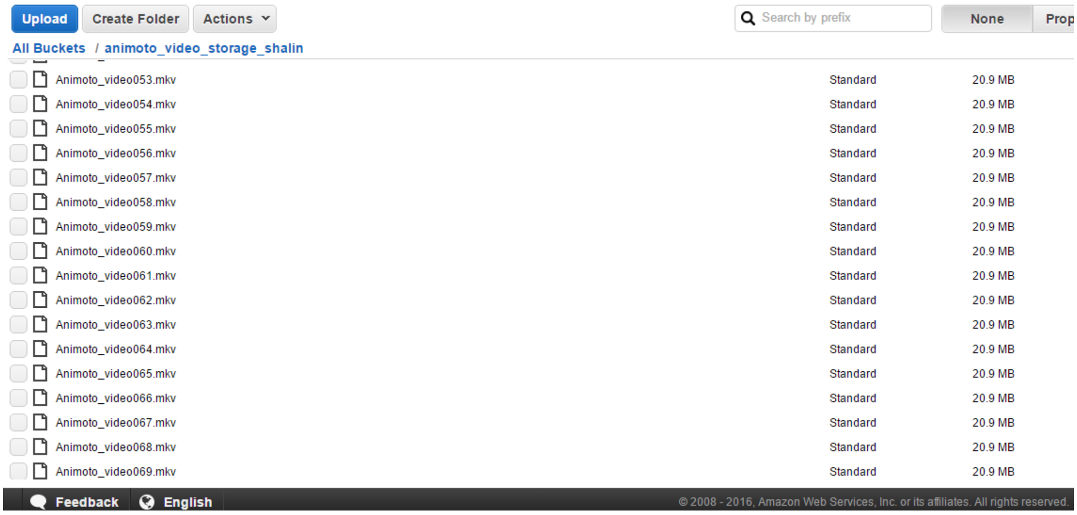
Name
animoto_video_storage_shalin

Storing Images in S3:



All Buckets / animoto_video_storage_shalin

Name	Storage Class	Size
Animoto_video000.mkv	Standard	20.9 MB
Animoto_video001.mkv	Standard	20.9 MB
Animoto_video002.mkv	Standard	20.9 MB
Animoto_video003.mkv	Standard	20.9 MB
Animoto_video004.mkv	Standard	20.9 MB
Animoto_video005.mkv	Standard	20.9 MB
Animoto_video006.mkv	Standard	20.9 MB
Animoto_video007.mkv	Standard	20.9 MB
Animoto_video008.mkv	Standard	20.9 MB
Animoto_video009.mkv	Standard	20.9 MB
Animoto_video010.mkv	Standard	20.9 MB

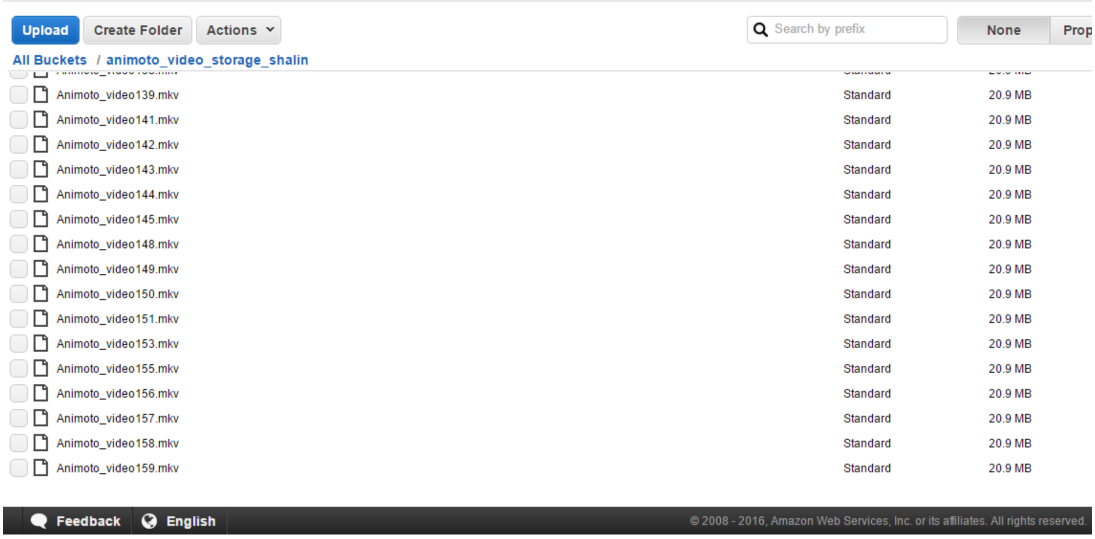


All Buckets / animoto_video_storage_shalin

Animoto_video053.mkv	Standard	20.9 MB
Animoto_video054.mkv	Standard	20.9 MB
Animoto_video055.mkv	Standard	20.9 MB
Animoto_video056.mkv	Standard	20.9 MB
Animoto_video057.mkv	Standard	20.9 MB
Animoto_video058.mkv	Standard	20.9 MB
Animoto_video059.mkv	Standard	20.9 MB
Animoto_video060.mkv	Standard	20.9 MB
Animoto_video061.mkv	Standard	20.9 MB
Animoto_video062.mkv	Standard	20.9 MB
Animoto_video063.mkv	Standard	20.9 MB
Animoto_video064.mkv	Standard	20.9 MB
Animoto_video065.mkv	Standard	20.9 MB
Animoto_video066.mkv	Standard	20.9 MB
Animoto_video067.mkv	Standard	20.9 MB
Animoto_video068.mkv	Standard	20.9 MB
Animoto_video069.mkv	Standard	20.9 MB

© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

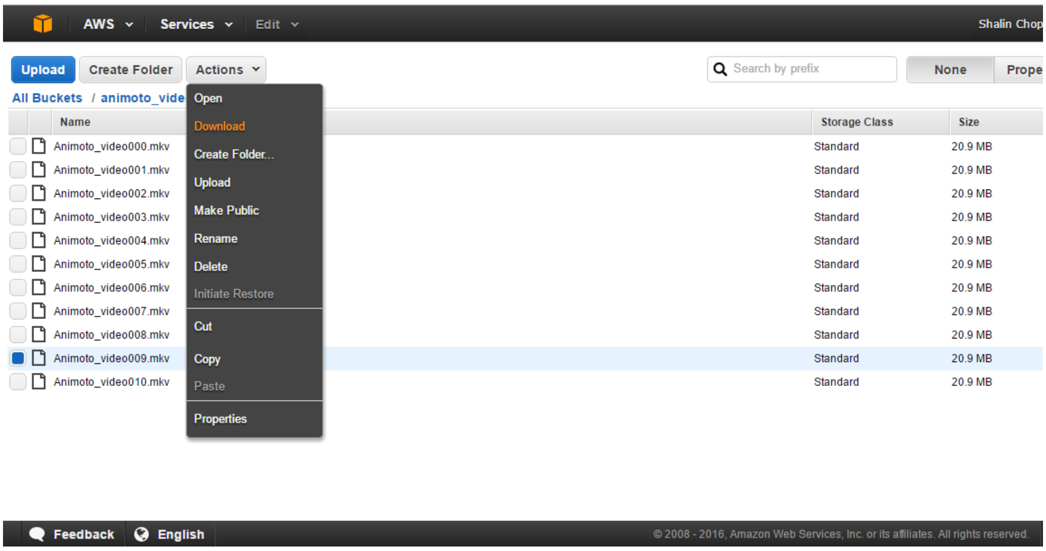
Some last tasks in execution:


Upload
Create Folder
Actions
Search by prefix
None
Prop
Feedback
English
© 2008 - 2016, Amazon Web Services, Inc. or its affiliates. All rights reserved.

All Buckets / animoto_video_storage_shalin

Animoto_video139.mkv	Standard	20.9 MB
Animoto_video141.mkv	Standard	20.9 MB
Animoto_video142.mkv	Standard	20.9 MB
Animoto_video143.mkv	Standard	20.9 MB
Animoto_video144.mkv	Standard	20.9 MB
Animoto_video145.mkv	Standard	20.9 MB
Animoto_video148.mkv	Standard	20.9 MB
Animoto_video149.mkv	Standard	20.9 MB
Animoto_video150.mkv	Standard	20.9 MB
Animoto_video151.mkv	Standard	20.9 MB
Animoto_video153.mkv	Standard	20.9 MB
Animoto_video155.mkv	Standard	20.9 MB
Animoto_video156.mkv	Standard	20.9 MB
Animoto_video157.mkv	Standard	20.9 MB
Animoto_video158.mkv	Standard	20.9 MB
Animoto_video159.mkv	Standard	20.9 MB

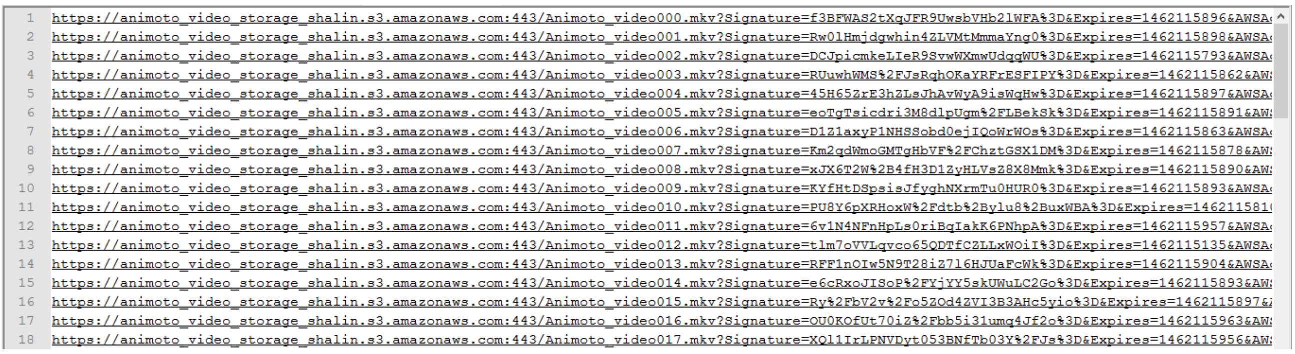
Downloading Video:



All Buckets / animoto_vide

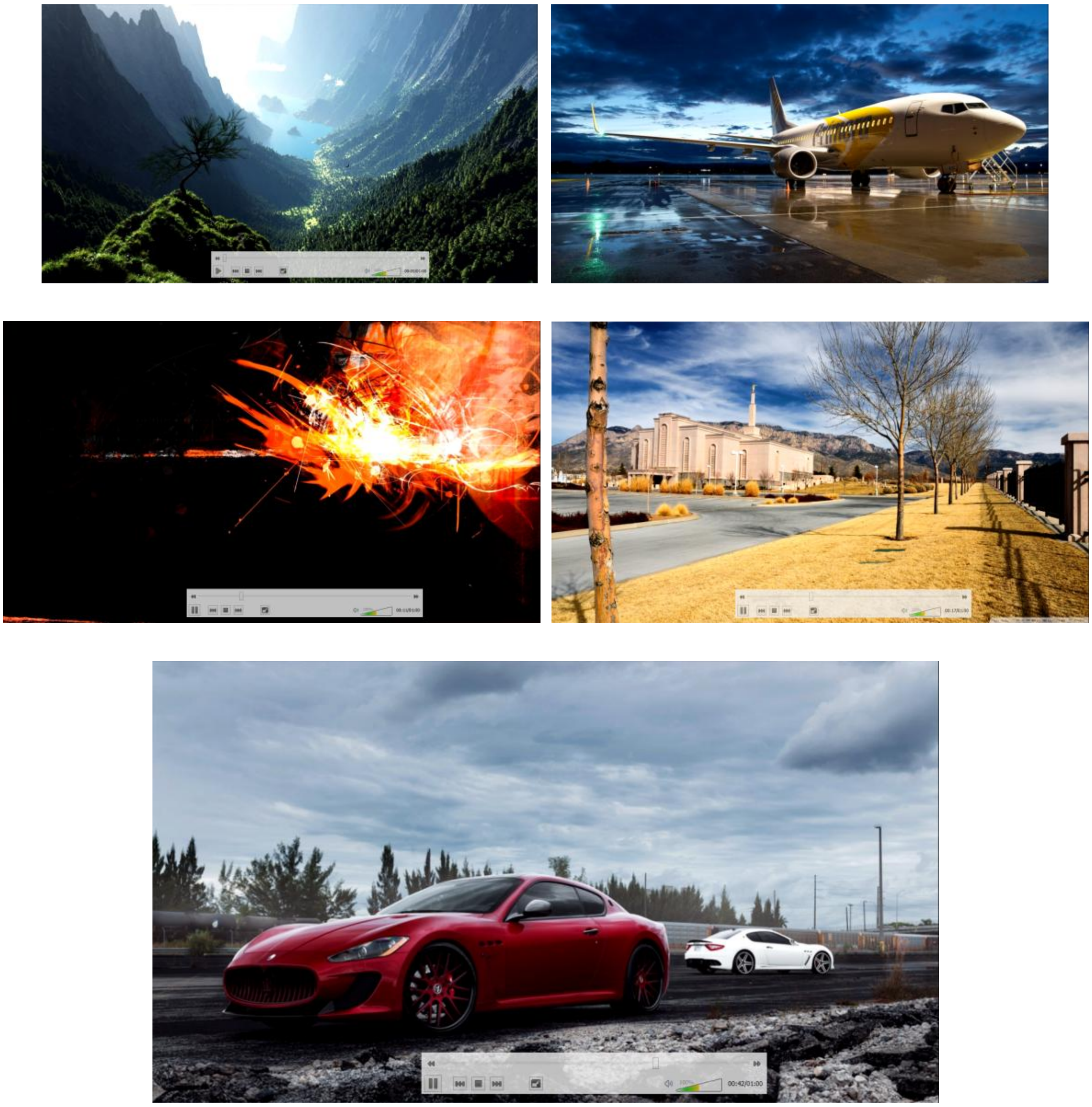
Name	Storage Class	Size
Animoto_video000.mkv	Standard	20.9 MB
Animoto_video001.mkv	Standard	20.9 MB
Animoto_video002.mkv	Standard	20.9 MB
Animoto_video003.mkv	Standard	20.9 MB
Animoto_video004.mkv	Standard	20.9 MB
Animoto_video005.mkv	Standard	20.9 MB
Animoto_video006.mkv	Standard	20.9 MB
Animoto_video007.mkv	Standard	20.9 MB
Animoto_video008.mkv	Standard	20.9 MB
Animoto_video009.mkv	Standard	20.9 MB
Animoto_video010.mkv	Standard	20.9 MB

Video URLs in Log File:



```
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video000.mkv?Signature=f3BFWAS2tXqJFR9UwsbVHb2lWFA%3D&Expires=1462115896&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video001.mkv?Signature=Rw0lHmjdgwhin4ZLVMtMmmaYng0%3D&Expires=1462115898&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video002.mkv?Signature=DCJpicmkeLIeR9SvwWXmwUdqqWU%3D&Expires=1462115793&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video003.mkv?Signature=RUuwhWMS%2FJsRqhOKaYRFrESFIPY%3D&Expires=1462115862&AW
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video004.mkv?Signature=45H65ZrE3hZLsJhAvWyA9isWqHw%3D&Expires=1462115897&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video005.mkv?Signature=eoTgTsicdri3M8dlpUgm%2FLBekSk%3D&Expires=1462115891&AW
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video006.mkv?Signature=D1Z1axyP1NHSSobd0ejIQoWrWOs%3D&Expires=1462115863&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video007.mkv?Signature=Km2qdWmoGMTgHbVF%2FChztGSX1DM%3D&Expires=1462115878&AW
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video008.mkv?Signature=xJX6T2W%2B4fH3D1ZyHLVsZ8X8Mmk%3D&Expires=1462115890&AW
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video009.mkv?Signature=KYfHtDSpsisJfyghNXrmTu0HUR0%3D&Expires=1462115893&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video010.mkv?Signature=PU8Y6pXRHoxW%2Fdtb%2Bylu8%2BuxWBA%3D&Expires=146211581
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video011.mkv?Signature=6v1N4NFnHpLs0riBqIakK6PNhpA%3D&Expires=1462115957&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video012.mkv?Signature=tlm7oVVLqvco65QDTfCZLLxWOiI%3D&Expires=1462115135&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video013.mkv?Signature=RFF1nOIw5N9T28iZ7l6HJUaFcWk%3D&Expires=1462115904&AWSA
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video014.mkv?Signature=e6cRxoJISoP%2FYjYY5skUWuLC2Go%3D&Expires=1462115893&AW
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video015.mkv?Signature=Ry%2FbV2v%2Fo5ZOd4ZVI3B3AHc5yio%3D&Expires=1462115897&
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video016.mkv?Signature=OU0KOfUt70iZ%2Fbb5i31umq4Jf2o%3D&Expires=1462115963&AW
https://animoto_video_storage_shalin.s3.amazonaws.com:443/Animoto_video017.mkv?Signature=XQl1IrLPNVDyt053BNfTb03Y%2FJs%3D&Expires=1462115956&AW
```

Downloaded Video Snaps:



References:

- http://datasys.cs.iit.edu/publications/2014_CCGrid14_CloudKon.pdf
- http://boto.cloudhackers.com/en/latest/sqs_tut.html
- http://boto.cloudhackers.com/en/latest/dynamodb_tut.html
- http://boto.cloudhackers.com/en/latest/s3_tut.html
- http://www.ffmpeg.org/faq.html#How-do-I-encode-single-pictures-into-movies_003f