User Manual

Falkon/Animoto clone with Amazon Cloud Computing Services

Contents

1. Introduction

This user manual is prepared in order to guide a user to use the system. The manual explains in detail about each module, their requirements and the running environment.

2. Architecture

2.1. The Client

The client is implemented in python. To run the client, run the following command :

python client.py -s <IPADDRESS:PORT> -w <WORKLOADFILE>

The IP Adress and port are given by the scheduler. The workload file contains the tasks separated by new line character.

2.2. The Front-End Scheduler

The scheduler program is implemented in php. We have tested this program with php version – PHP 5.4.34. In order to run local workers, one should have php installed on the instance/machine. The scheduler follows different interfaces for local and remote workers.

2.2.1. Local Back-End Worker

- The scheduler follows below interface for local workers :

  scheduler <PORT> lw <NUMTHREADS>

- To run the scheduler in local mode, run below command :

  php schedule.php <PORT> lw <NUMTHREADS>

2.2.2. Remote Back-End Worker

- Install "AWS SDK for PHP" on the instance having the scheduler.
- In code's bootstrap process, need to explicitly require the bootstrap file from Version 1 of the SDK as shown below.

  require '/path/to/sdk.class.php'

- The scheduler follows below interface for local workers :

  scheduler <PORT> rw 0

- To autoload the file that is capable of autoloading all of the classes in any of the libraries that it downloads, need to add the following line to code's bootstrap process:

  require '/path/to/sdk/vendor/autoload.php'

- To run the scheduler in remote mode, run below command :

  php schedule.php <PORT> rw 0

2.3. Local Back-End Worker

The local back-end worker does not require to be launched or run separately. The code is included with the scheduler, and running is managed by the scheduler.Ã§

2.4. Remote Back-End Worker

2.4.1. Before Compilation

Follow below steps before compiling the program.

1. Create two SQS Queue. (Request Queue , Response Queue)
2. Create two DynamoDB tables
    - Task Table with "task-Id" column.
    - Instance Table with "Inatance-Id" and "Time-Out" column.

Update the Main Class as follow:

1. Change the "accesskey" and "secretKey" in BasicAWSCredential.
2. Change the value of "REQUEST-QUEUENAME" and "INSTANCE-TABLE-NAME" with the name of your own Request Queue and Instance Table.
3. Change the URL of Request Queue. ("reqQueueUrl" variable)
4. Change the value of "RESPONSE-QUEUE" to your response queue.

Update the Task Class as follow:

1. Change the "APP-NAME" variable with the name of python animoto program.
2. Change the value of "JAVA-PORT" and "PYTHON-PORT"
3. Pllace animoto program in home.
4. Change the value of "TABLE-NAME" with the name of your Task table.

2.4.2. Compile

Comile with javac and the Amazon SDK library or use Amazon AWS Enabled Eclipse. Our compiled JAR file (with the values of our Amazon accounts):

https://www.dropbox.com/s/k4fdnuy7yriwkxw/Remote/%20Worker.jar?dl=0

2.4.3. Terminal (For running)

Run the program with command :

java -jar worker.jar

2.5. Dynamic Provisioning

2.5.1. Before Compilation

Update the Instance Class as follow:

1. Change the value of "TABLE-NAME" to your own Instance table.
2. Create a file with the name of "imageid.txt" and put the AMI ID of the instance on it. "ami-3d50120d".

Update the Main Class as follow:

1. Change the "accesskey" and "secretKey" in BasicAWSCredential.
2. Change the value of "REQUEST-QUEUENAME" and "INSTANCE-TABLE-NAME" to your own names.

2.5.2. Compile

Comile with javac and the Amazon SDK library or use Amazon AWS Enabled Eclipse. Our compiled JAR file (with the values of our Amazon accounts):

https://www.dropbox.com/s/b12ck4w0yhs7ebp/dynamic-provision.jar?dl=0

2.5.3. Terminal (For running)

To run in smart dynamic allocation of workers based on the size of request queue, run following command:

java -jar dynamic-provision.jar To statically allocate a worker, run following command:

java -jar dynamic-provision.jar <time-out>

2.6. Animoto Clone

The main code for animoto clone is implemented in Python. To run the program on the worker instance, install the AWS Python SWK. The program requires the AWS access key and secure key, which should be declared in the code. The program also requires the S3 bucket name and access to store the created video. The video is created with help of ffmpeg, which should be installed on the instance as well. The running of this code is managed by the remote worker. To make it work along with the remote worker, save the python file - animoto.py in the same directory as the remote-worker jar file. This program will download images and creates video, which are stored on local disk, so keep enough space to run the program.

3. Task Execution

To execute any tasks using this system, follow below sequence:

1. Run the front-end scheduler in local or remote mode.[on VM as per the prog definition]
2. Run the dynamic provision module.[on VM as per the prog definition]

3. Run the remote worker module.[on VM as per the prog definition]
4. Run the client program.[on local machine as per the prog definition]
5. The scheduler will submit the jobs and return the response to the client.