**Dsps Assignment 1**

Sagi levi: 308213446

Daniel Abutbul: 313843062

**Installation**

Run the localapp:

* java -jar localApp.jar inputName outputName n

or, if you want this localapp to be the last one, that will kill the Manager:

* java -jar localApp.jar inputName outputName n terminate

when:

* **inputName** are: the input you enter to the manager when the format is “Job URL” when Job is ToImage/ToText/ToHtml
* **output**: is the name of the HTML resulting by the program.
* **n**: is the number of reviews each worker will process.
* **Terminate**: telling the local app to kill the manager after finishing processing his Jobs.

**Program Explain**

**Terms:**

* **Task –** It’s the tiny pieces of the Job we get from the Local.
* **Personal queue:** each local application has a personal SQS queue to communicate with the manager.
* **Manager queue:** is the main queue of the program. Each new local application sends throw it his personal queue,and Job.
* **Jars bucket:** main bucket for containing the jars: Manager, Worker.
* **Locals bucket:** all the local app input are upload to this bucket, and the manager download form there the inputs to process.
* **Workers Queue In:** messages from workers to the manager – finish terminating message or Task that was processed.
* **Workers Queue Out:** messages from the manger to the workers – terminate message or Task to process.

**Local app:**

1. **Local App Runner:**
   1. The local app create the “local’s Bucket” and upload his inputFile to s3, and create for himself a personal queue.
   2. The local will check if the manger is alive. If not, he will create him, by creating Ec2 instance , If the manager was alive, he send to his queue the URL to the file and URL to his personal queue.
   3. the local will upload another input file, if needed and will wait for Done message from the manger.
   4. After "done" arrived, the local will download The output from S3, and will parse it into HTML, and save it in the current directory.
   5. If terminate **wasn't** one of the arguments, local will finish.
   6. Else, a "terminate" message will be sent to the manager, telling the manager to terminate nicely. And start waiting for "done" message from the manager.
   7. After it's arrived, we know that all workers, queues are deleted, and he will exit.
2. **Out Req**
   1. This thread are listen to the local app keyboard and ask if he wants to enter another input , if yes then we send a new msg to the Manager Queue with the new input, and count the new task he enter. If not then this thread is terminate and only the local App Runner continue to run and wait for the output file.

**Manager:**

1. The manger will create a thread Pool that will execute handle new local app , and listener to the workers.
2. **Local Req Handler**
   1. This Class are receive message from the manager queue
   2. And, For each message:
      1. If it's a terminate message, he set the Manager Runner to terminate, and will stop the loop (terminating himself),and delete the main queue of the manager
      2. Else, it's a new clients message containing his personal queue address and the URL of the input file in s3. So, he handle the message by uploading another workers if needed (depend of the N the local send and how much workers we have alive).
   3. After handling all the messages he got, he received another msges and then he yield and give control to the other thread that runs the workers.
3. **WorkerReqHandler**
   1. This class are listen to the WorkersToManagerQueue
   2. He get all task that the workers finish and process it. If the all of the specify client request were completed , then he sent to this client that he done this task and url to the output file in s3
4. **ManagerRunner**
   1. This class create all the first Initalize of the manager. He creates the queue for the workers (in and out) and create Thread pool that handle all the things in the system(handle new local app and handle workers)
5. **Worker:**
   1. While not terminate:
      1. Received messages from the manger.
      2. For each message:
         1. If it a terminate message, send to the manager that he is finished, and exiting.
         2. Else, download the task URL and performed the operation that was requested. In case of error of downloading the file or convert it , the worker send the manager the error that was occurred. Otherwise he will upload the new file to s3 and send the URL to the Manager.

**Usage**

* in the same directory where the project located, the local can save all the inputs he want.
* at the end of the process, the summary file (.html) will be in the same directory.

**Additional data**

* the ami we use is: ami-0626266604b6dbaf4
* how much time its took on samle-input-1: **12 minutes**
* how much time its took on samle-input-2: 4 minutes
* what n we used: N = 3
* Security? We use IAM ROLE for all the instances we uploaded, and set the credentials as global variable in there Img.
* Scalability – we used SQS, S3,Ec2 that are Scalability, all the files are stroed in s3, and not on the computer .The works save the converted file and delete the file. The manager saves on his memory Table of the clients .When he finished with this client we delete any information we saved in the memory.
* Persistence : if works dies, The manager checks the works list ,and upload a new one if necessary. If the worker take a long time to process the msg , because we set the visibility of the messages to be high enough , if another worker accidently manage to take the job, In the manager we wrote a function that count this to one, although two workers will send the converted PDF .
* We used in the manager with thread pool so he will give enough attention To new locals , and for the workers messages.
* Every client queue named by his Ip, so we run with two locals from two different computers.
* Do you understand how the system works? Do a full run using pen and paper, draw the different parts and the communication that happens between them.
* The manager terminated process , we don’t get another requests from new local, he inform the workers they need to terminated, not before they finishes all the msg.
* The job is equals to all the workers . every worker that finish a job, upload the output and go to receive another messages