User Manual

1) Generate the Workload File :

- *bash FileCreate.sh*
  - this bash file will generate the workload files with the specific no. of tasks(ex. 10 ms with 1 thread = 1000*1 Tasks).
  - Folder 1worker, 2worker …..16 worker contains files 10, 1000, 10000 for 10ms, 1 sec and 10 sec respectively times.

2) Local Worker Execution :

- Compile and Run :
  - *Javac Client.java*
  - *Java Client -s Local -t N -w FileName*
    N= No. of threads
    FileName= Name of the workload File
- This execution will send the task in the queue and worker will start reading the tasks one after another.
- Worker tasks does parallel adding of bit 0 or 1 as per the tasks execution in the response queue.
- After finishing the execution, total time for the execution is printed.

3) Remote Worker Execution :

- Send the Client.java and Worker.java in two different EC2 t2.micro instance.
- Create two queues in the Client.java and two in Worker.java. RequestQueue has the tasks and response queue has the Task Status.
- DyanmoDb is instantiated in both Client and Worker files.
- Now, Compile and run Client.java
  - *javac Client.java*
  - *java Client -s RequestQueue -w FileName*
    where, RequestQueueu= SQS Task Carrying queue
    FileName=WorkLoad File
- On successful execution of the command, line by line data is taken from the Workload file and sent in the queue.
- Now, Compile and run Worker.java
  - *javac Worker.java*
  - *java Worker -s ResponseQueue -t N*
    where, ResponseQueue= SQS Task Status queue
    N=No. Of instances to be launched
- First, Worker checks the taskid in the DyanamoDb(Table Name :Task-Table). If the taskid matches with the taskid in the DyanmoDB table, then it will delete the task from the RequestQueue and put the value 0 in the ResponseQueue.
- If the taskid does not exist in the DyanamoDB table, then that task is added in the

DyanmoDB table and then the task is executed. Meanwhile, the task status 0 set in the ResponseQueue.

- The key feature of the Worker is that once it is started it will never stop. It will keep on pulling the data from the client with the infinite loop pulling functionality.
- Parallelly, the ResponseQueue will keep on sending the task status back to client.
- If at the client side if the total task send matches with the no. of task status in ResponseQueue then the total execution time will be printed with the Task completion status.