**Explanation of implementation**

The purpose is to find how many logs generated for each city on spesific time. For example:

If we had following log file :

2019-04-16-21:03:10 DEBUG   Tokyo   Hello from Tokyo

2019-04-16-21:03:10 DEBUG   Beijing Hello from Beijing

2019-04-16-21:03:10 INFO    Beijing Hello from Beijing

2019-04-16-21:03:11 WARN    Moskow  Hello from Moskow

2019-04-16-21:03:11 FATAL   Beijing Hello from Beijing

2019-04-16-21:03:11 ERROR   Moskow  Hello from Moskow

2019-04-16-21:03:11 FATAL   Moskow  Hello from Moskow

2019-04-16-21:03:12 WARN    Beijing Hello from Beijing

2019-04-16-21:03:10 Tokyo -> 1

2019-04-16-21:03:10 Beijing -> 2

2019-04-16-21:03:11 Moskow -> 3

2019-04-16-21:03:11 Beijing -> 1

2019-04-16-21:03:12 Beijing -> 1

For that purpose my method is to group cities with time column and count the element of these groups.

You can find explanation of elements that created for that purpose.

## Loggenerator.sh

Script generates log which will be read by Producer. Generating period is 0.25 second as default. Script also controls size of log and it creates backup log time prefixed backup log when size of current log exceeds 2MB.

Running Method:

- chmod +x loggenerator.sh (only first run)

- ./loggenerator.sh

### Producer Jar

Jar file takes log path as argument and returns it a JSON formated document. It also adds key to this file to calculate how many logs came from a city on x time. Outputs its data to city-logs-input topic.

Example : Time : 2019-04-16-21:16:40, City: Tokyo -> Key : 2019-04-16-21:16:40Tokyo

This method helps us grouping logs by city and the time.

Running Method:

- Should be run after loggenerator.sh

- java -jar city-count-producer-1.0-SNAPSHOT-jar-with-dependencies.jar [path\_of\_log(where loggenerator.sh runs)]

### Count App Jar

Jar file does the calculation job using KafkaStreams. It aggregates the count value grouping by the key value which comes from producer. Outputs its data to city-logs-output topic.

Due to streaming calculation process does not work as expected. Whenever new data is written to log file count value is inreases. So occurness of city being updated every time new data is written to log file.

Running Method:

- Should be run after city-count-producer-1.0-SNAPSHOT-jar-with-dependencies.jar

- java -jar city-log-count-app-1.0-SNAPSHOT-jar-with-dependencies.jar

Consumer Jar

This jar takes database user name and its password as argument and inserts the data to database which is previously written to city-logs-output topic.

Running Method:

- Should be run after city-log-count-app-1.0-SNAPSHOT-jar-with-dependencies.jar

- java -jar city-log-count-consumer-1.0-SNAPSHOT-jar-with-dependencies.jar [dbUser] [dbPassword]

After running all these files our system gets logs calculates occurness of cities on logs and writes the result to database.

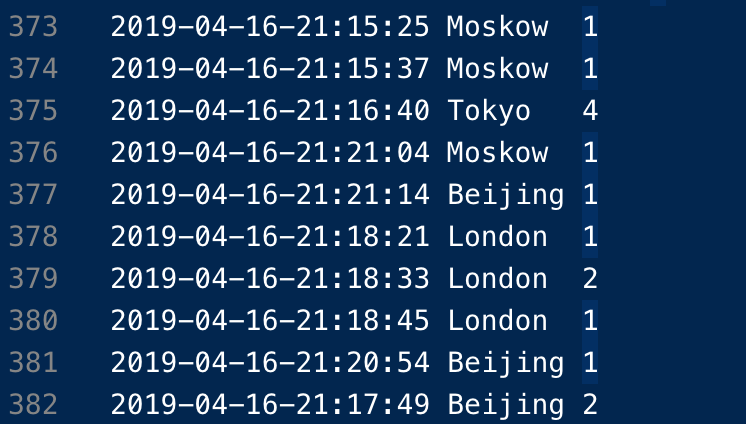
Calculation process result can be verified using following methods:

#### Verification Method

**Log file side :**

* Route to log file path and run following command:
* cat current\_log.log | awk -F '\t' '{printf $0"\t"$1$3"\n";}' | awk -F '\t' '{keys[$5]++;time[$5]=$1;city[$5]=$3;count[$5]+=1;}END{for(key in keys){printf time[key]"\t"city[key]"\t"count[key]"\n";}}'> result.txt

This command basicly groups cities and time column and counts their occurness which is our purpose. When result.txt opened in an text editor counts of cities can be seen. Result will be like following:



This result basically means 4 rows of log came from the city of Tokyo on the time 2019-04-16-21:16:40 .

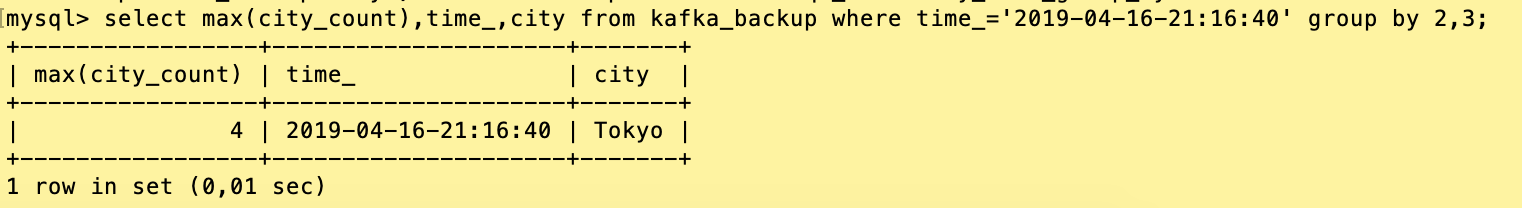
This step our first step of verifiying.

Note : If you don’t use Unix based operating system like Linux or MacOS. GitBash will help you running AWK command fort his step.

**Database side :** For this side MySQL is used. On MySQL just run following command:

* select max(city\_count),time\_,city from kafka\_backup where time\_='2019-04-16-21:16:40' group by 2,3;

This command is also does the same job that we executed on previous step. And result will get us the maximum count of the city and time group. Again it means how many times logs are counted from the spesific city and a spesific time.



It can be clearly seen that the calculation is done correctly but this count value will increase whenever new log came from the same city on exact the same time.

**Dependencies**

**- java > 1.8**

**- Kafka v**0.11.0.1

- **MySQL v5.7 :predefined** **structure like following:**

*create database kafkabackup;*

*use kafkabackup;*

*create table kafka\_backup (time\_ varchar(30), log\_level varchar(20), city varchar(20),log\_message varchar(100));*