# Computer Assignment III

Course: Applied Cloud Computing
Course-ID:
1TD265

Instructors:
Andreas Hellander, Andreas.Hellander@it.uu.se
Salman Toor, Salman.Toor@it.uu.se

*The following assignment can take anything from 10-20h to complete, depending on prior knowledge.*

**Task 1**

This task is to be completed by everyone and is a requirement for a passing grade on this assignment (1 point).

In this assignment, you will build a prototype system to analyze a dataset of ~5.000.000 Twitter tweets collected beforehand using Twitter's datastream API. The tweets are available in the public container 'tweets' in RegionOne in SMOG, and the dataset consists of 18 files containing line-separated tweet entries. Every second line is a blank line.

Each tweet is a JSON document http://en.wikipedia.org/wiki/JSON. JSON is one of the standard Markup formats used on the Web. While it would not be considered your typical "scientific data", familiarity with JSON quickly becomes essential in applied cloud computing and data analytics. For the specific case of Twitter tweets, you can read about the possible fields in the JSON documents here: https://dev.twitter.com/overview/api/tweets

This particular Twitter dataset was collected by filtering the stream of tweets to store those containing the Swedish pronouns "han", "hon", "den", "det", "denna", "denne" and the gender neutral, new pronoun "hen". Your task is now to construct a compute-service based on the Distributed Task Queue Celery, using RabbitMQ as the broker, capable of analyzing the dataset on demand. In particular, as an example of an analysis, your solution should count the total number of mentions of the above mentioned pronouns in the dataset.

*Minimal Requirement for a passing grade on Task 1:*
To complete Task 1, your solution should minimally comprise of:

- A backend comprising of Celery/RabbitMQ deployed on a single VM (one Celery worker)

- This service should be exposed to an end-user via a simple REST API, capable (only) of returning a JSON document containing the pronouns and counts. For a very simple example of a REST API, see the Flask app from Lab1).

- A Visualization of the frequencies (number of mentions normalized by the total number of unique tweets) of the pronouns, for example as a bar chart. In the analysis, only unique tweets should be taken into account, i.e. ‘retweets’ should be disregarded.

- Upload a short report to the student portal, describing how you designed your solution, and displaying e.g. a screenshot of the REST API call showing the returned result. Also include the visualization (see above point), and your answer to the theory questions.

*Practical advice:*

- Break down your work into smaller components:
    - Subject you code to version control from the beginning. Plan for VMs to be short-lived.
    - Start by deploying a simple, single node deployment of Celery/RabbitMQ by completing the following this tutorial https://www.digitalocean.com/community/tutorials/how-to-use-celery-with-rabbitmq-to-queue-tasks-on-an-ubuntu-vps

    - Download a single datafile from the Swift bucket, extract only a small portion (use e.g. a Python script to read only the first line) and learn how to use the ‘json’ module in order to parse the tweet documents

.

    - Develop and test the code to extract the pronoun from the twitter message on a small subset of the data, and external to the Celery system (use a simple Python script)

Useful links:
http://celery.readthedocs.org/en/latest/getting-started/first-steps-with-celery.html

http://celery.readthedocs.org/en/latest/getting-started/brokers/rabbitmq.html#starting-stopping-the-rabbitmq-server

**Task 2**

This task is not a requirement to pass the assignment, but can give up to 1 point, counting towards higher grades (see grading table).

In this task, we will look more into the performance of the system we just developed. Conduct a weak scaling analysis of your solution, using up to 8 workers of size m1.medium. For this, you will need to extend your backend system to a distributed setup with multiple Celery workers connecting to the same broker. Feel free to suggest/try other chunk-sizes of the data (currently, the individual files are ~500MB each). If you do so, please create another bucket with your modified dataset, don't ever write to the public tweets container. Report your findings and discuss your results.

Although not a requirement to complete the task, it would of course be elegant and amusing with a Web-frontend that is capable of displaying the incremental result of the analysis to the end-user as Celery tasks complete (only attempt this if you have previous experience with Web-frontend development, and is familiar with e.g. Django, jQuery or similar).

*Practical advice:*

- In Celery 3.1 the default 'guest' user in RabbitMQ is only allowed to connect on localhost. For a distributed setup, you need to create a separate user and vhost using 'rabbitmqctl' (see the Celery docs)
- Plan how you intend to conduct the analysis and how to collect timing results.
- Use 'Flower' to get a visual confirmation that workers are connecting
- Instead of manually starting workers, use your knowledge from Lab2 to automate the task of adding/removing VMs/workers to the pool. (This is not a requirement to pass this task, but might save you time in the end)