

**ADVANCES IN DATA SCIENCE/ARCHITECTURE**

**ASSIGNMENT-1**

**REPORT**

**BY**

**MEGHA SINGH**

**SNEHA MALSHETTI**

**Index**

|  |  |
| --- | --- |
| **Sno.** | **Topic** |
| 1. | Overview |
| 2. | Some of the Installations and softwares used for creating the project: |
| 3. | Data Ingestion |
| 4. | Exploratory Data Analysis |
| 5. | Data Wrangling |
| 6. | Automation on Celery |
| 7. | Automation on Cloud(AWS Batch) |
| 8. | Docker and Celery |
| 9. | Citations |

**Overview :**

There is a huge pool of data around the world. Tera bytes and even picobytes of data are collected everyday by companies who need this data to process it and analyse it to make future predictions on this data. Not all the data that comes in is properly formatted. In fact, some data is so poorly formatted it becomes more of a challenge to read the data into a csv file or any other system-readable or human-readable format.

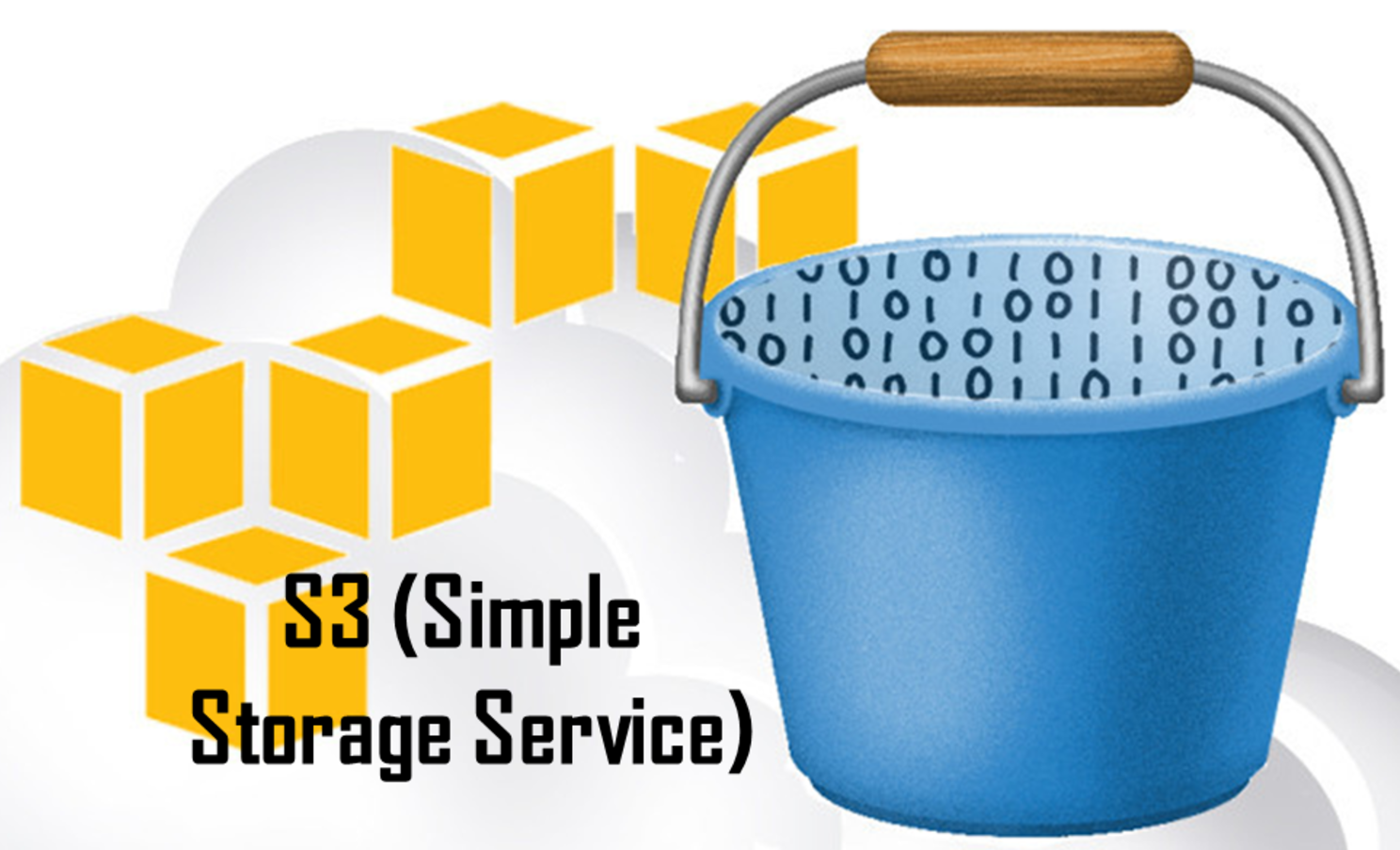
This data needs to be pulled in from more than one systems in scenarios where there aren’t many API’s supporting this import. Hence we use languages like R and Python to convert this data pushed from various systems using standard file formats like JSON and csv to read and manipulate data the way we want to visualize it.

The data that we spend so much space, time and money for, is then used to visualize the trends using various visualization graphs and table format. But for this the data needs to be cleaned first. The processes involved are Feature Selections and Data Wrangling. Feature Selection involves selecting what features you want to keep in the data set you have, with a justifiable reasoning of why this set of data has to be kept and why the other set has to be discard. The reasoning depends completely on the kind of analytics you want to perform. After Feature Selection we perform Data Wrangling which keeps only what is required and manipulates the rest of the data as desired.

And then comes data visualization to see if we missed and missing values or outliers in the data wrangling and maybe perform it again and then comes the part where we automate this entire process using pipelines like (Make, Celery, Airflow) so we can scale this process for larger data sets and finally schedule it so it doesn’t need any human intervention to start the process everyday, unless there is an error.

**Some of the Installations and softwares used for creating the project:**



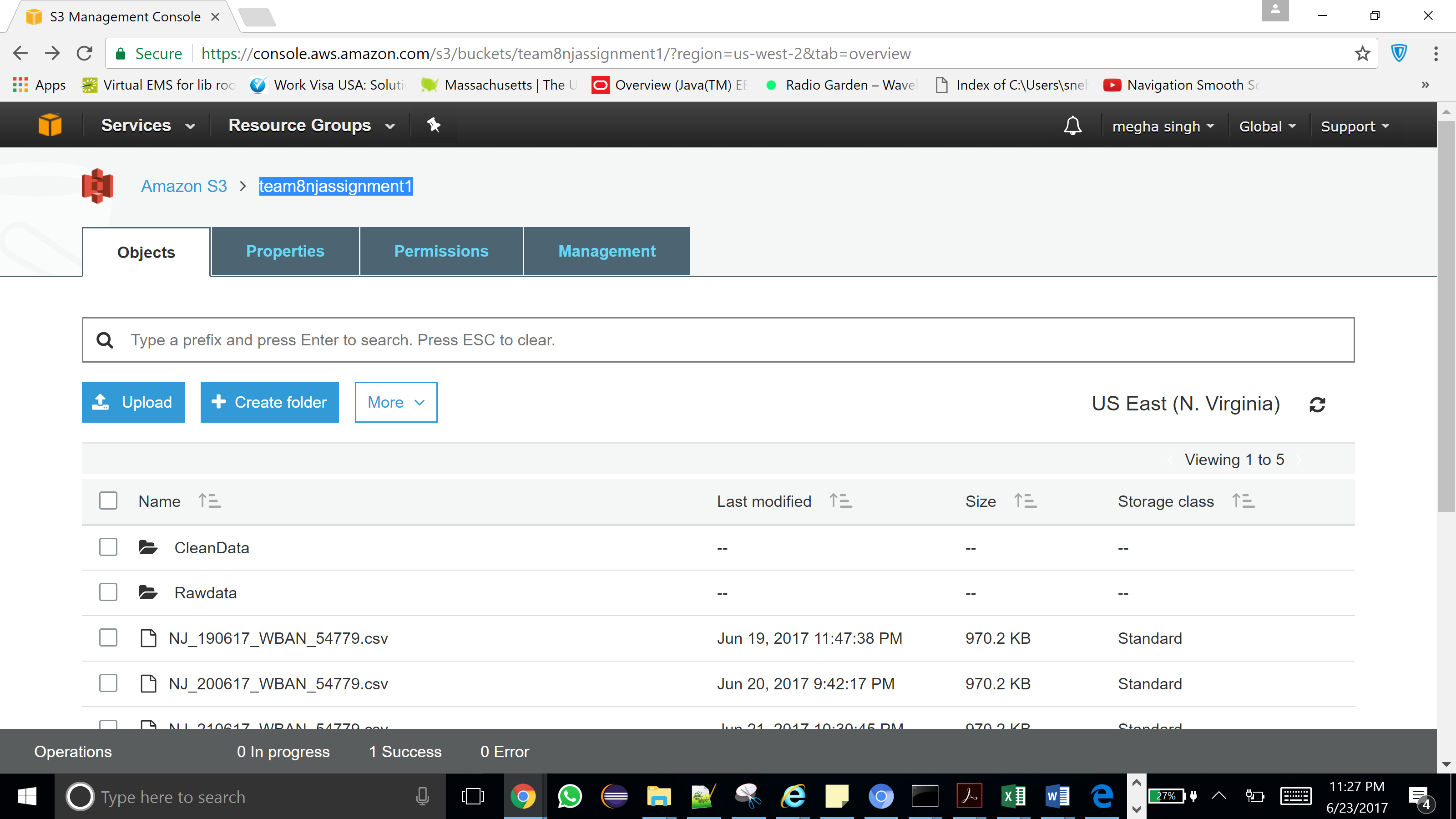
**Data Ingestion :**

For Data Ingestion, we had to first of all install Anaconda and run the code in jupyter notebook environment which supports Python 3 . Python 3 hoverver is not compatible with many tools like docker and Xamp. The latest and most compatible versions of Python are 2.6.x nd 2.7.x. This was one issue we faced during out assignment. Another compatibility issue was with python shell and pithon notebook and ipython. All3 have a few syntax changes and hence this added to the version compatibility issues. However, having said that, the most seemingly easy to work with tool was the jupyter notebook because it had most of the libraries installed in the environment so we didn’t hve to install pandas or numpy libraries or even matplotlib kind of libraries.

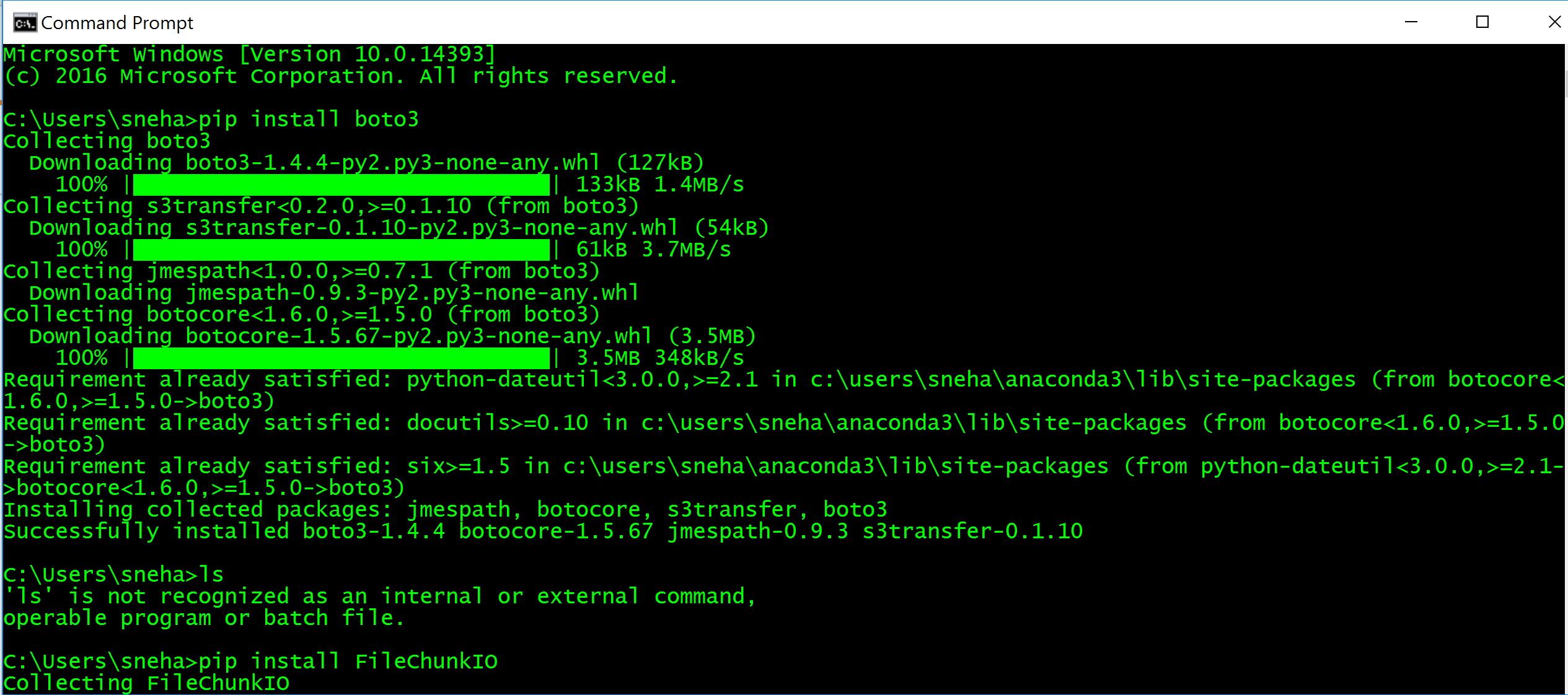
To connect with S3 we had to install Boto and Boto3 since boto doesn’t work in python3 invironment and hence a lot of issues were faced. A lot of times the S3 bucket removes access from buckets and hence there is a lot of trouble with that as well. It seems to change permissions without notice, which is not easy to catch and the code doesn’t run since all the data that comes in is from the AWS S3 Bucket. In S3 Bucket we cannot duplicate names so incase there is a name already on S3, we need to create a different name this is to keep all bucket names on the cloud unique.

The S3 Bucket we created : [team8njassignment1](https://console.aws.amazon.com/s3/)

S3Bucket:



Boto Installation :



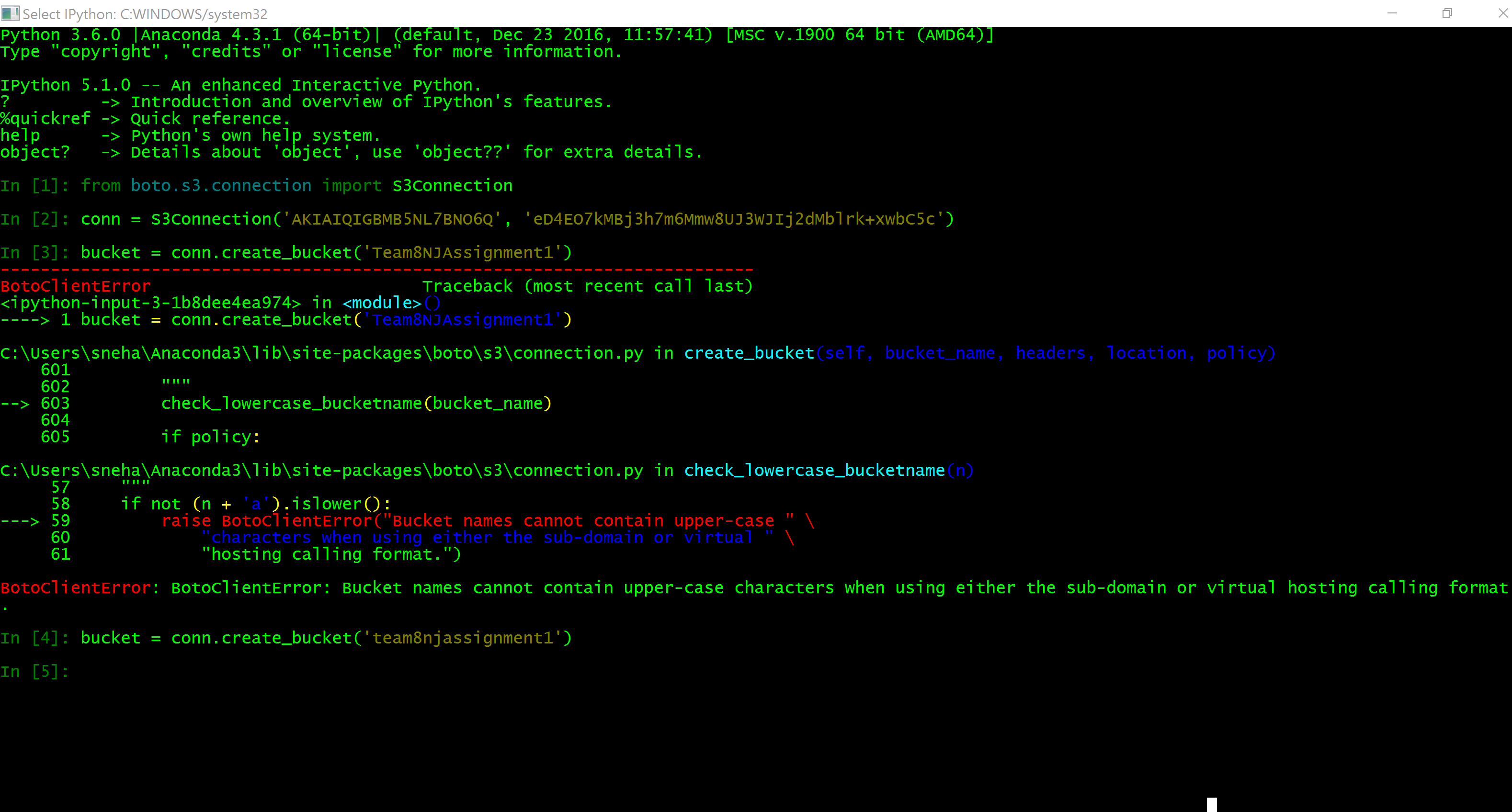
A few Boto commands that are same for Boto3 and Boto:

In [1]: from boto.s3.connection import S3Connection

In [2]: conn = S3Connection('AKIAIQIGBMB5NL7BNO6Q', 'eD4EO7kMBj3h7m6Mmw8UJ3WJIj2dMblrk+xwbC5c')

In [3]: bucket = conn.create\_bucket('Team8NJAssignment1')

In [4]: bucket = conn.create\_bucket('team8njassignment1')



**Exploratory Data Analysis :**

The following are a few plots that we thought would be useful in summarizing the entire year’s Temperature or humidity and other factors. This also helps in Data Analysis. And to make future predictions based on models. The visualization of data also gives us an insight of whether the data has gone bad with a few outliers and missing values or strings in the numeric fields.

Hence it is necessary that we carry out data cleaning activities in the raw data with techniques like data wrangling using R or Python.

**Data Wrangling :**

Data wrangling is the process of manipulating data to get reasonable values. This helps in data visualization as well because wrangled data is good for data visualization since it has less anomalies than that when working with Raw data.

**Automation using Pipelines and Cloud:**

The entire process of data extraction/retrieval and processing and analysing can be automated by using pipelines. This can also be scheduled to avoid human intervention and human errors. This helps scale the process for any amount of data and any type of data coming in. The other way is to automate it on the cloud and automating it on cloud is much easier since we have a management console on AWS and we do not have to type in commands and it runs on both Mac and Windows environment.

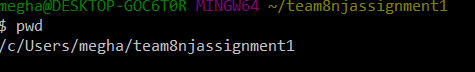
**DOCKER and Celery:**

A Docker *image* is a read-only template used to create and launch a Docker *container*.

Docker File

1. Create a working directory” team8njassignment1” for docker image and then create a docker file inside it.





1. Create a docker File

* touch Dockerfile



Docker file created

1. Now Configure you dockerfile to run python script file and shell (dataingestion.py and )

Edit the Dockerfile using command

* Cat Dockerfile



1. For dockerfile

FROM ubuntu

RUN apt-get update

RUN apt-get update update apt-get install php5

# Install Python.

FROM python: 3

RUN mkdir -p /usr/src/team8njassignment1

WORKDIR /usr/src/team8njassignment1

COPY \*.py \*.json \*.sh /usr/src/team8njassignment1

RUN pip install jupyter notebook

RUN pip install boto3

RUN pip install python-louvain

RUN pip install numpy

RUN pip install matplotlib

RUN pip install pandas

RUN pip install ipython

ADD run.sh /

RUN chmod +x /run.sh

#ENTRYPOINT ["/run.sh"]

ENTRYPOINT ["python", "/usr/src/team8njassignment1/dataingestion.ny"]

This instruction is similar to the COPY instruction with few added features like remote URL support in the source field and local-only tar extraction. But if you don’t need a extra features, [it is suggested](https://docs.docker.com/articles/dockerfile_best-practices/#add-or-copy) to use COPY as it is more readable.

Example:

ADD http://www.site.com/downloads/sample.tar.xz /usr/src

EXPOSE

While running your service in the container you may want your container to listen on specified ports. The EXPOSE instruction helps you do this.

Example:

EXPOSE 6456

#### ENTRYPOINT

You can use this instruction to set the primary command for the image.

For example, if you have installed only one application in your image and want it to run whenever the image is executed, ENTRYPOINT is the instruction for you.

Note: arguments are optional, and you can pass them during the runtime with something like docker run <image-name>.

Also, all the elements specified using CMD will be overridden, except the arguments. They will be passed to the command specified in ENTRYPOINT.

Example:

CMD "Hello World!"

ENTRYPOINT echo

#### WORKDIR

This is used to set the currently active directory for other instructions such as RUN, CMD, ENTRYPOINT, COPY and ADD.

Note that if relative path is provided, the next WORKDIR instruction will take it as relative to the path of previous WORKDIR instruction.

Example:

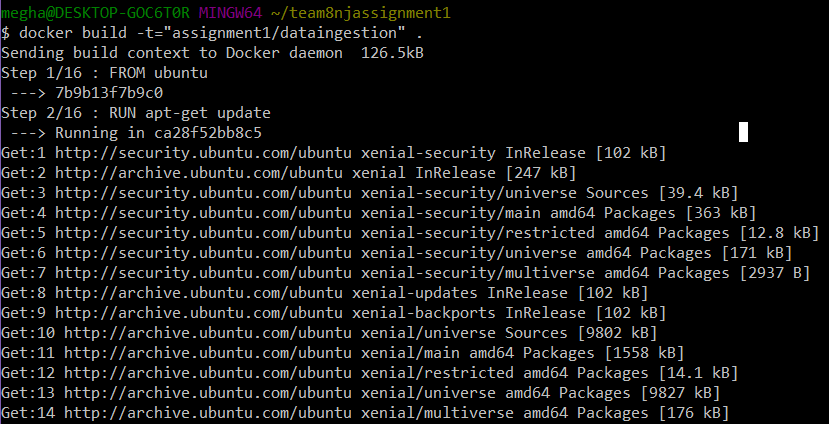
WORKDIR /user

WORKDIR home

RUN pwd

CREATE a docker image using dockerfile

* docker build -t =”assignment1/dataingestion” .



Docker image is created.

Celery task scheduling

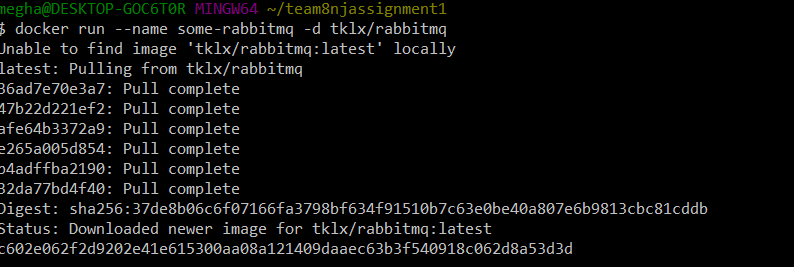
[Celery](http://www.celeryproject.org/) is an asynchronous task queue/job queue based on distributed message passing. It is focused on real-time operation, but supports scheduling as well. The execution units, called tasks, are executed concurrently on a single or more worker servers using multiprocessing, Eventlet, or gevent. Tasks can execute asynchronously (in the background) or synchronously (wait until ready). Celery is used in production systems to process millions of tasks a day.

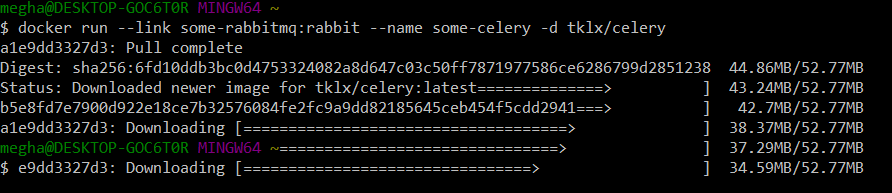
[Celery](http://www.celeryproject.org/), a python library which sits on top of RabbitMQ and provides workers to execute tasks.

### Start a celery worker utilizing RabbitMQ as broker

$ docker run --name some-rabbitmq -d tklx/rabbitmq

$ docker run --link some-rabbitmq:rabbit --name some-celery -d tklx/celery



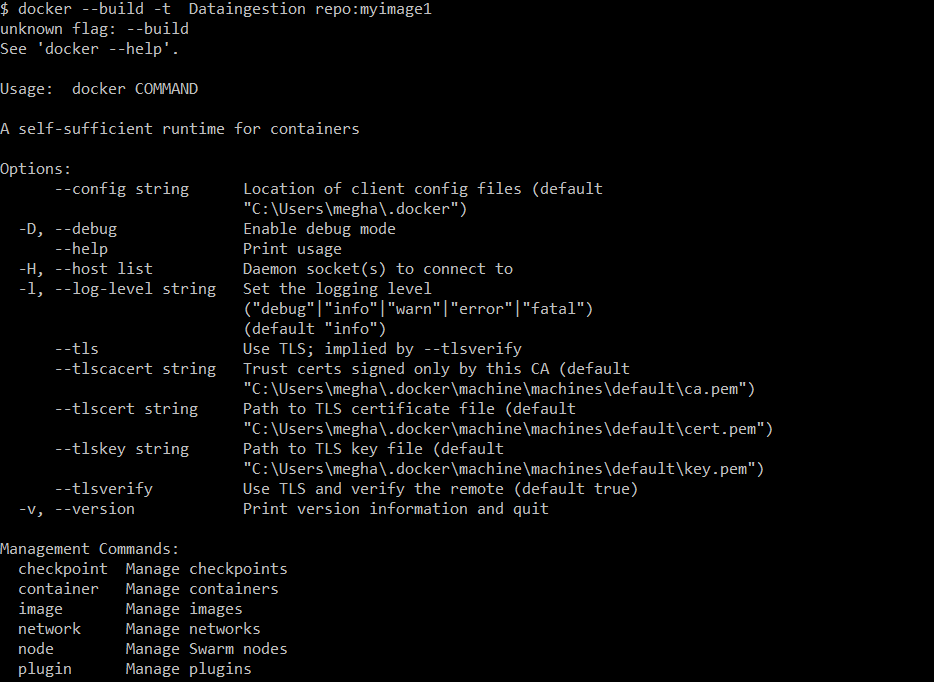
 

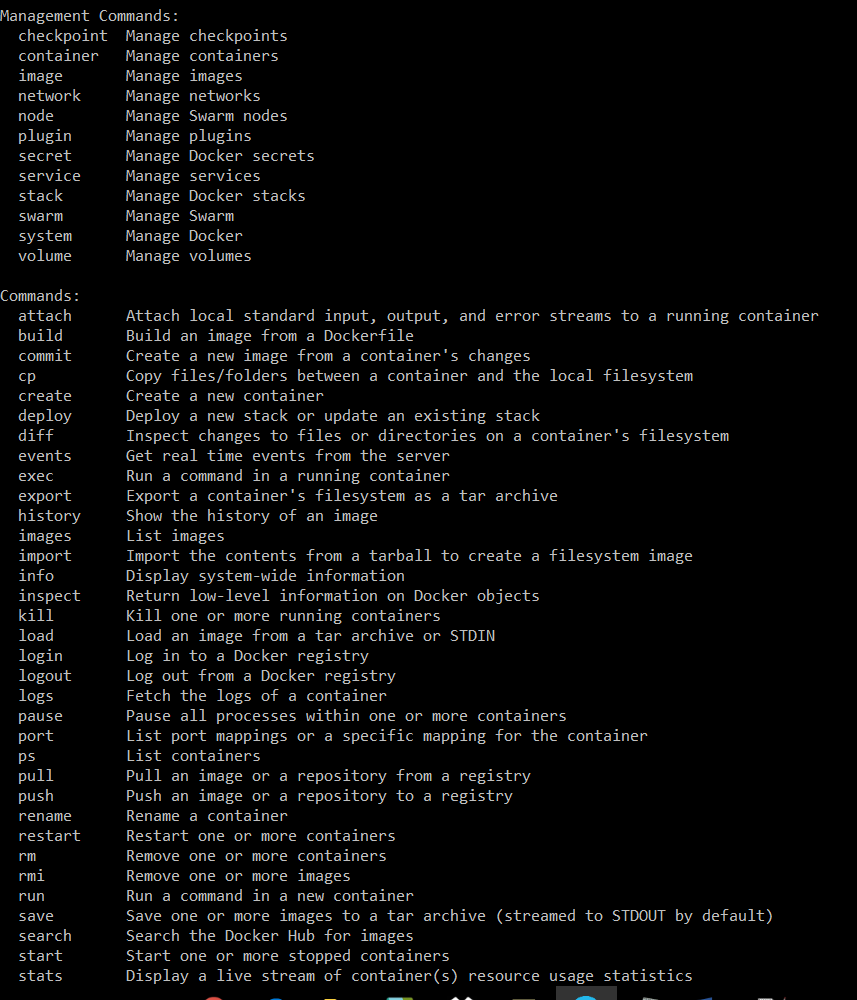
<http://rapidsms.readthedocs.io/en/develop/topics/celery.html>

A [task](http://docs.celeryproject.org/en/latest/userguide/tasks.html) is just a Python function. You can think of scheduling a task as a time-delayed call to the function. For example, you might ask Celery to call your function task1 with arguments (1, 3,3) after five minutes. Or you could have your function batchjob called every night at midnight.

When a task is ready to be run, Celery puts it on a [queue](http://docs.celeryproject.org/en/latest/getting-started/introduction.html#what-is-a-task-queue), a list of tasks that are ready to be run. You can have many queues, but we’ll assume a single queue here for simplicity.

Putting a task on a queue just adds it to a to-do list, so to speak. In order for the task to be executed, some other process, called a *worker*, has to be watching that queue for tasks. When it sees tasks on the queue, it’ll pull off the first and execute it, then go back to wait for more. You can have many workers, possibly on many different servers, but we’ll assume a single worker for now.





**Citations**

<https://www.ibm.com/developerworks/community/blogs/jfp/entry/using_ipython_notebooks_in_docker_containers_on_windows?lang=en>

boto.cloudhackers.com/en/latest/ref/s3.html

<http://www.slashroot.in/dockerfile-tutorial-building-docker-images-for-containers>

<https://stackoverflow.com>