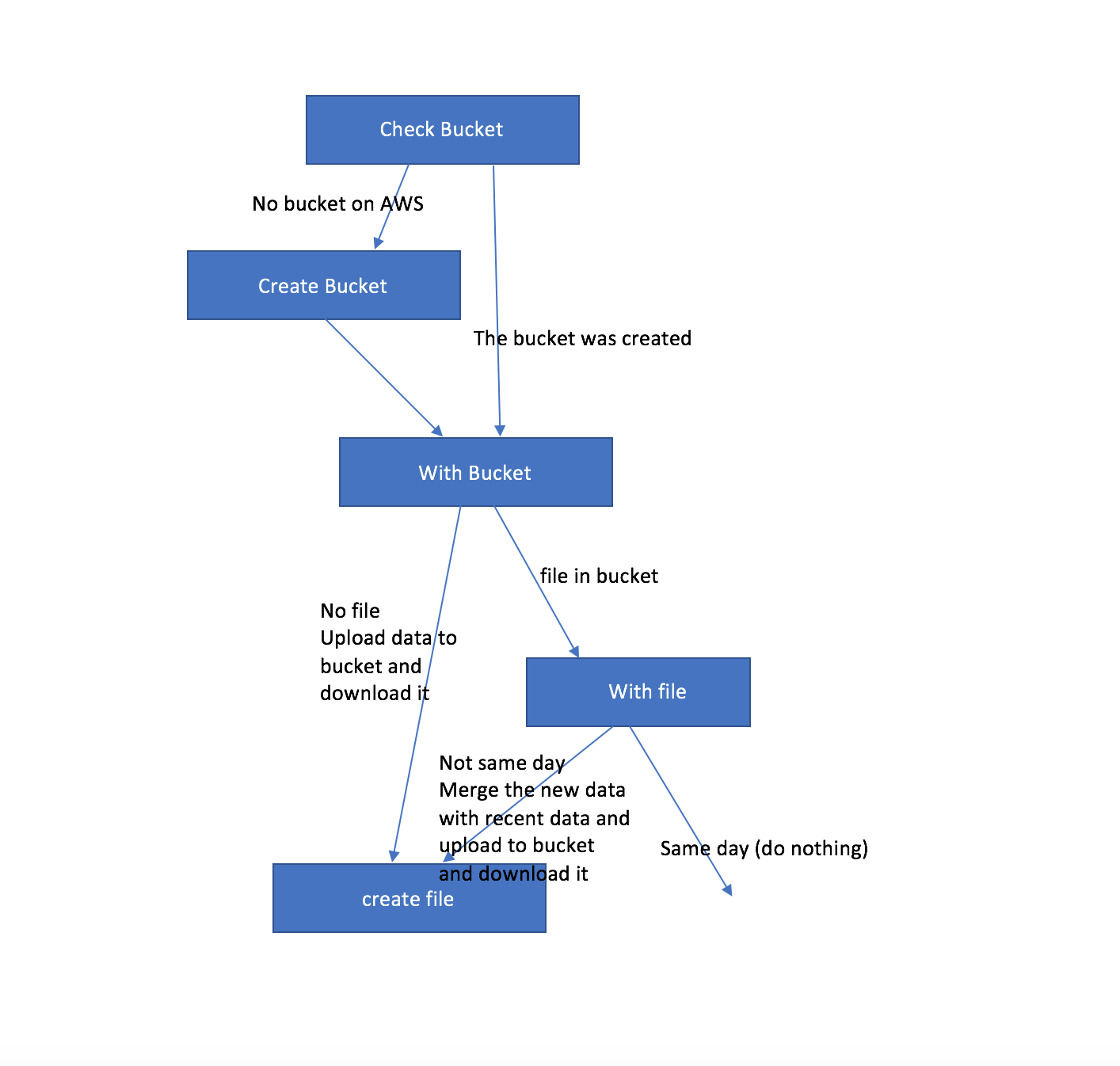
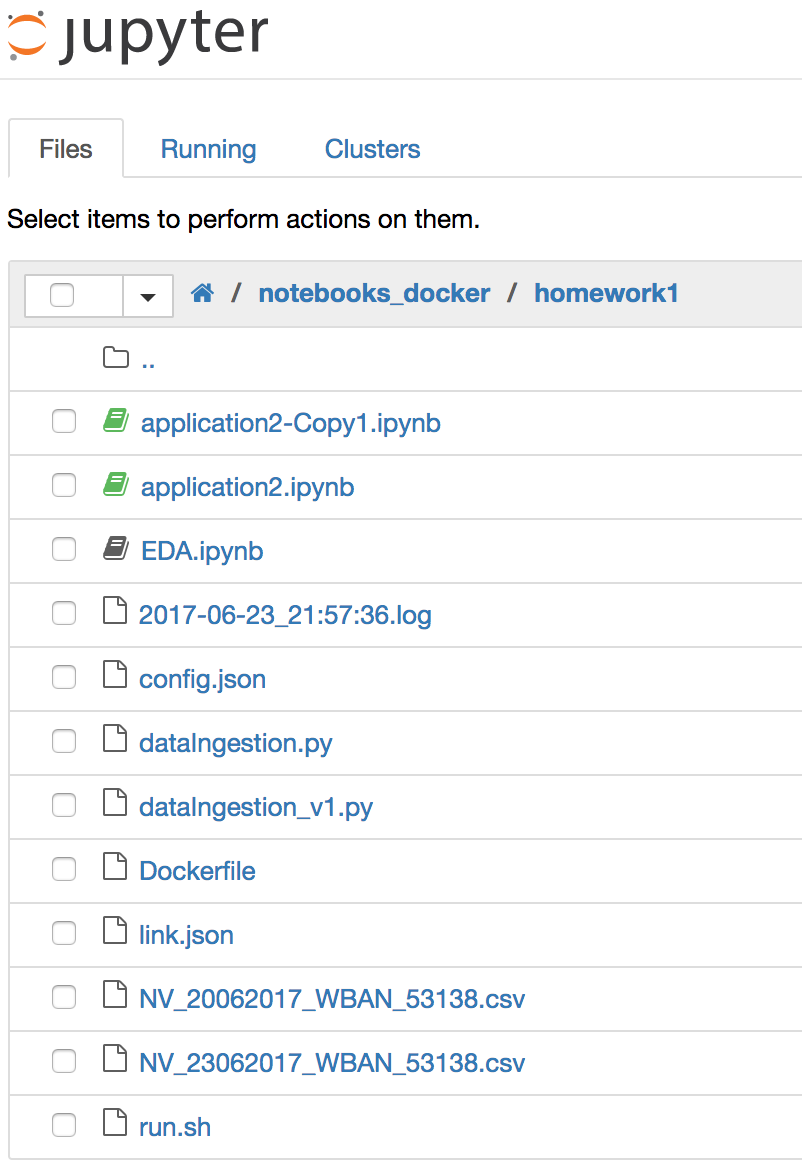
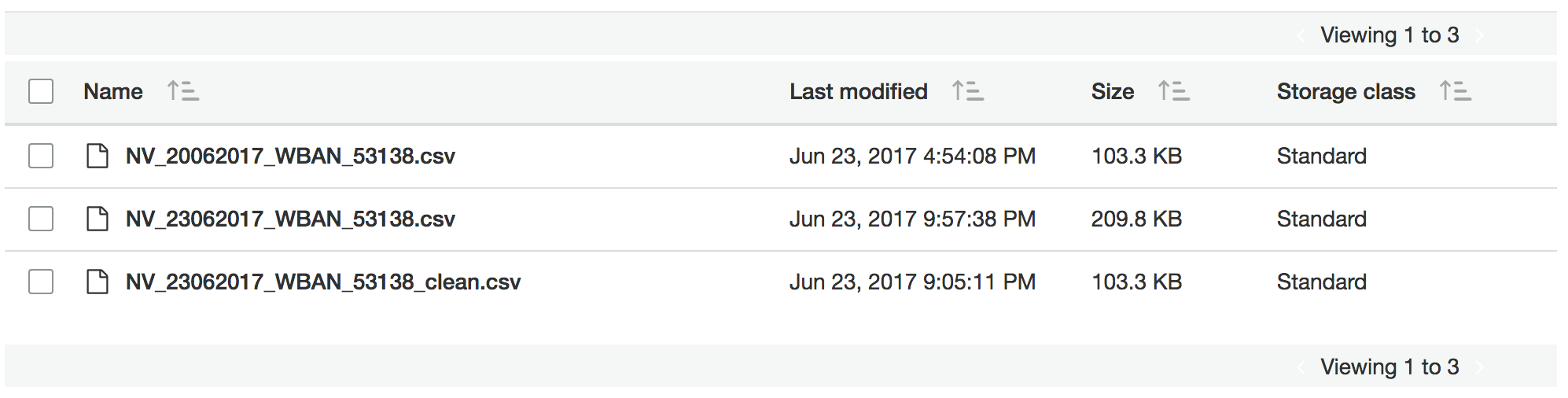
**Part1. Data Ingestion**



First, we check if a bucket exists on S3. If not, then we would create a bucket. And then we check if the file of that day exists in that bucket. If not, then we append the latest file in the bucket with the data of the current date to create the file of the current date and then download the file.



This is the directory under Jupyter. First, run “run.sh” to execute “dataIngestion.py”, which utilizes “config.json” to retrieve the weather data to local and then upload the file onto an S3 bucket. You can see “NV\_20062017\_WBAN\_53138.csv” and “NV\_23062017\_WBAN\_53138.csv” as the data files we got. And all the files along with their links are stored in “link.json”. The log file stores every piece of information that we output during the execution of the python file.



The figure above shows the S3 bucket after our code is run. The first two files are the raw data and the second one is created by appending the data of 06/23/2017 to the first file.

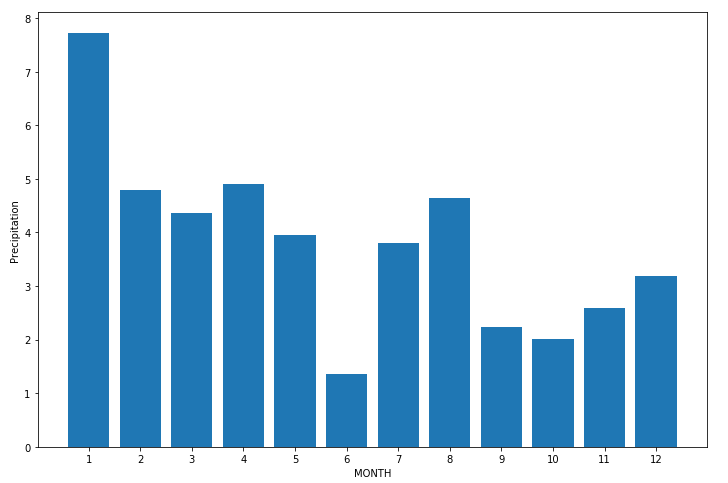
**Dockerize Image**

Under the local directory “/Users/XunPeng/Desktop/ADS/docker/notebooks\_docker/homework1”, there is a “Dockerfile”. In the docker file, write codes to copy certain files to the docker image. Then, run “docker build -t homework1 .” to create a local docker image “homework1”. Then run “docker run homework1” to run this docker image locally.

**EDA**

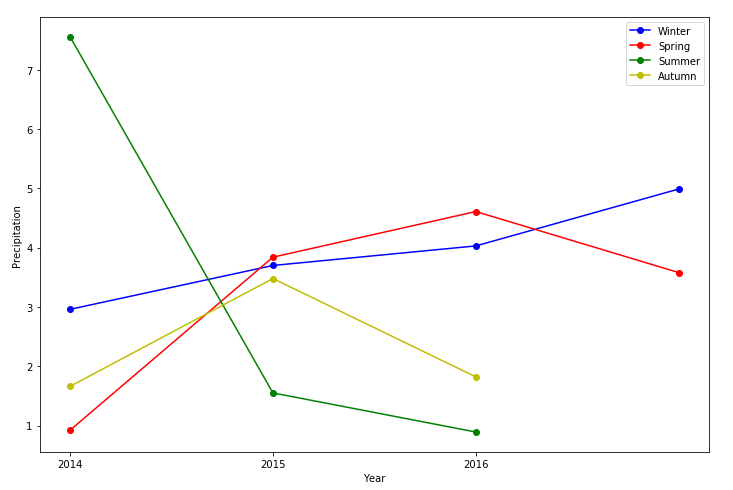
Please see below as the sample visualizations and analysis from *rawDataEDA.ipynb.* We would like to mention that we only have values in the following columns in the dataset of the first weather station in Nevada: temperature, precipitation, wind speed, and sunrise & sunset. That limits our potential of data analysis. Yet we came up with more than 10 visualizations that can give valuable insights about how the data is shaped.

**Precipitation By Month**



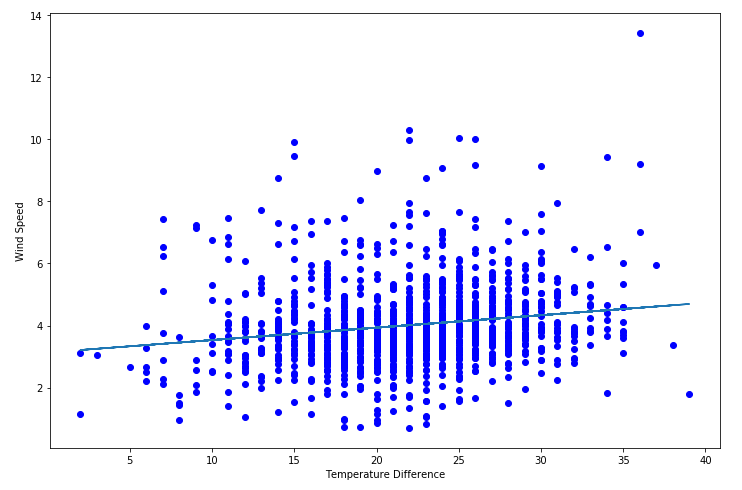
This bar chart was achieved by first grouping the precipitation data by month and getting the accumulated value. The figure above indicates that the precipitation is generally higher in winter and lower in summer.

**Precipitation By Season**



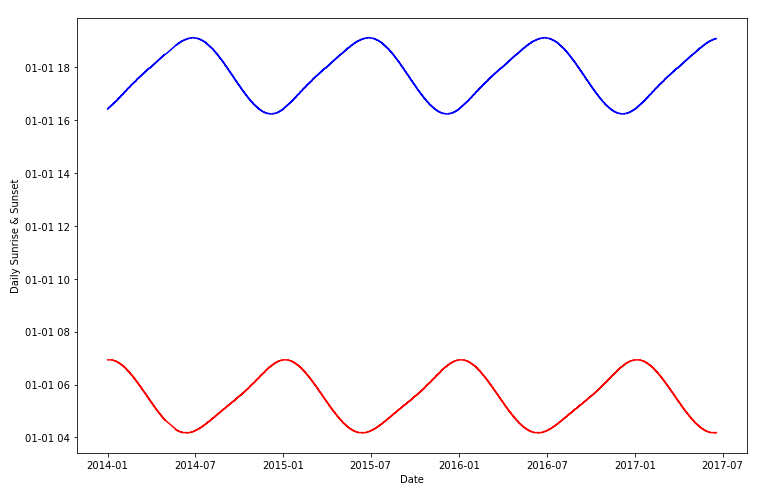
This line charts were achieved by first grouping the precipitation data by season and year and getting the accumulated value. Please refer to *rawDataEDA.ipynb* for the code that determines the four seasons. The figure above indicates that the precipitation in 2014 is high in summer and low in other 3 seasons. While the precipitation in summer decreases dramatically in 2015 and 2016.

# **Temperature Difference and Wind Speed**



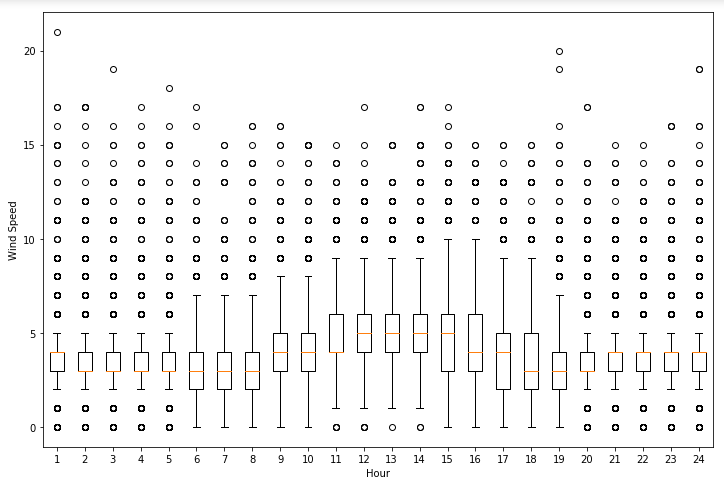
The figure above is the scatter plot that visualizes the correlation between temperature difference and wind speed in the NV weather dataset. Generally, it shows the positive correlation between the two attributes. But the correlation is not that strong.

**Sunrise and Sunset Line Chart**



We simply converted the sunrise and sunset columns to datetime type and plotted the line charts. The figure above indicates that the trend of sunrise and sunset time above the 3 and a half years time span is fluent and shows strong pattern. We can see that the peaks appear at the summer solstice and autumn solstice.

**Wind Speed By Hour**

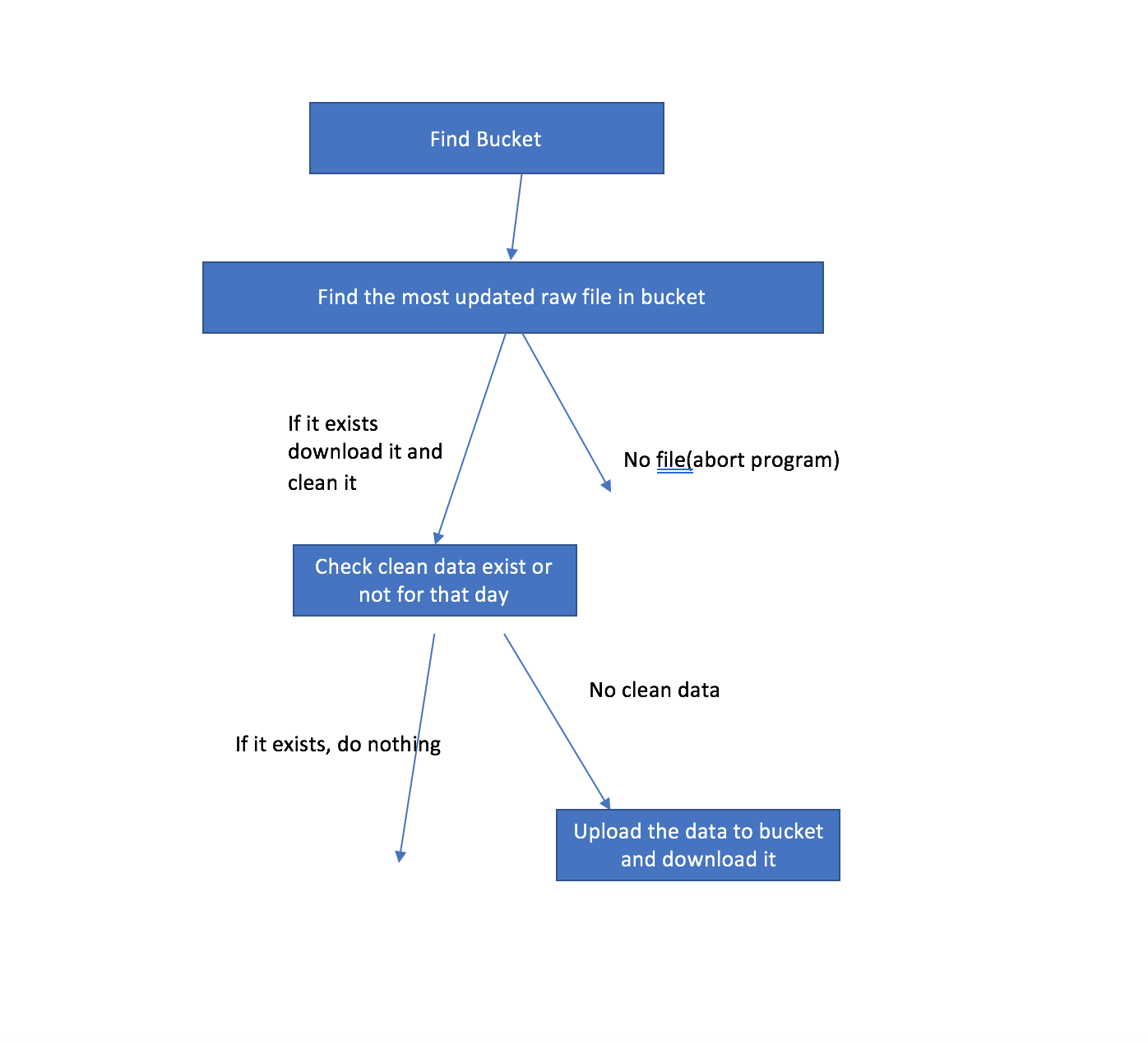


To be able to create this box plot, we aggregated the wind speed data by hour and created a list containing all the wind speed data for the 24 hours of a day. The figure above indicates that the wind speed is higher in noon time than in the morning and evening. And the wind speed is more sparsely distributed in noon time.

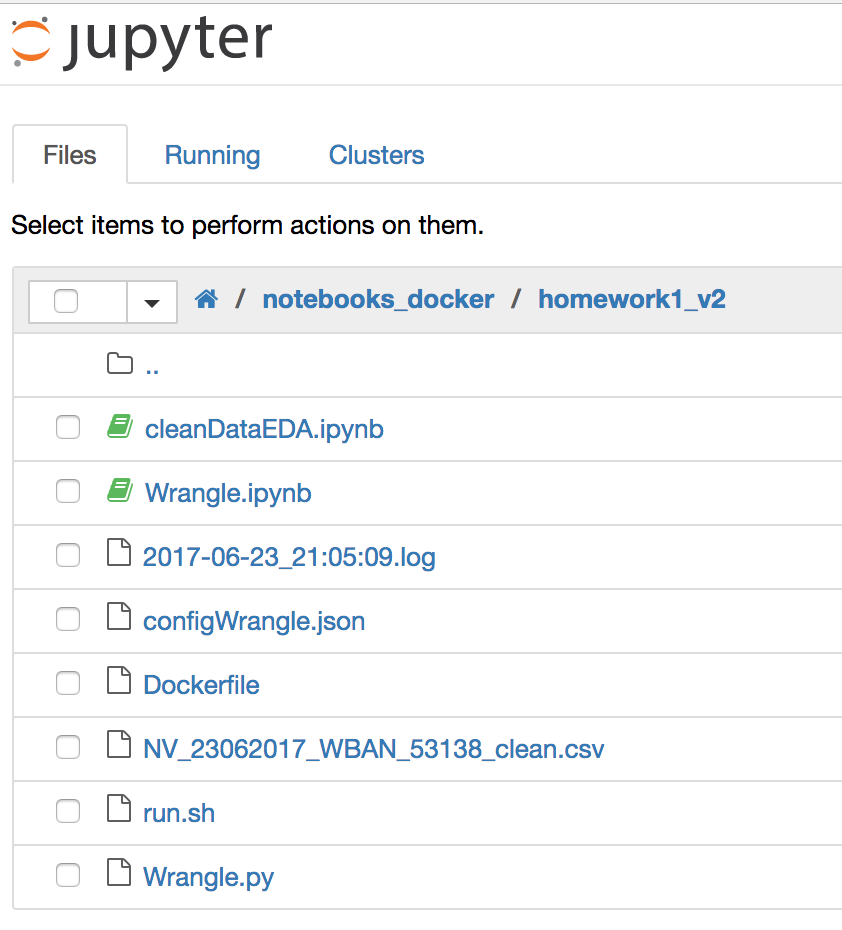
**Wrangling**

There are abnormal values in the dataset. We wrote code to automatically convert abnormal data like ‘54s’ to numeric values like ‘54’ and then wrote the clean data back to a csv file. As to outlier and missing value analysis, we think that we are dealing with the real-time data, so there does not exist missing outlier values. Therefore, we did not do that part.

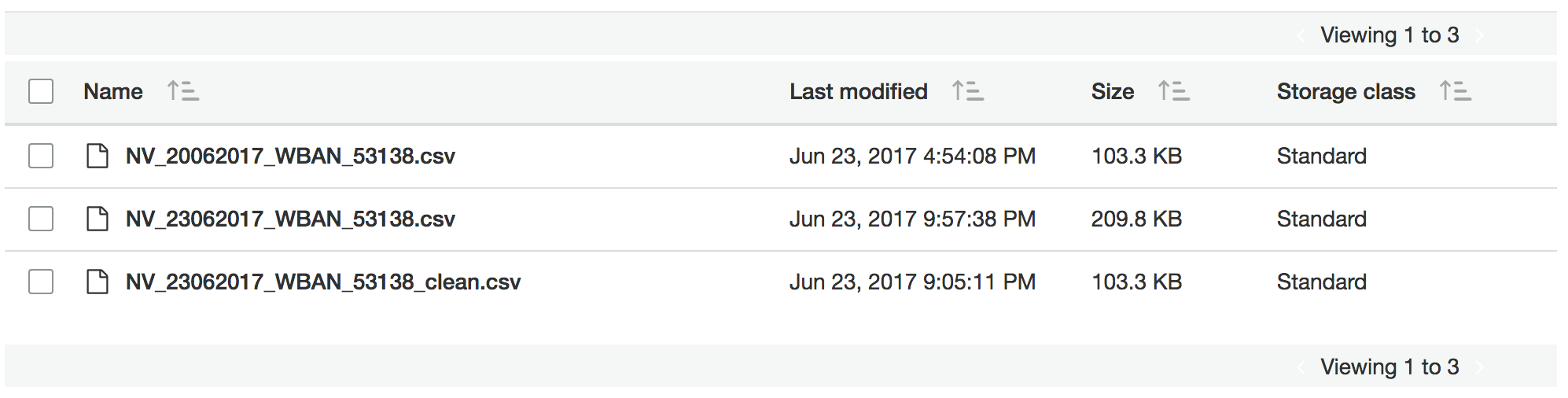
Please see the figure below as the process of uploading the cleaned data onto our S3 bucket.



First we would find the designated bucket. Then we would look for the latest raw file in the bucket. If we found it, we download it and clean it. Otherwise, the program is aborted. And after downloading and cleaning it, we then check if the cleaned data exists for that day. If not, then we upload the data to bucket and download it.

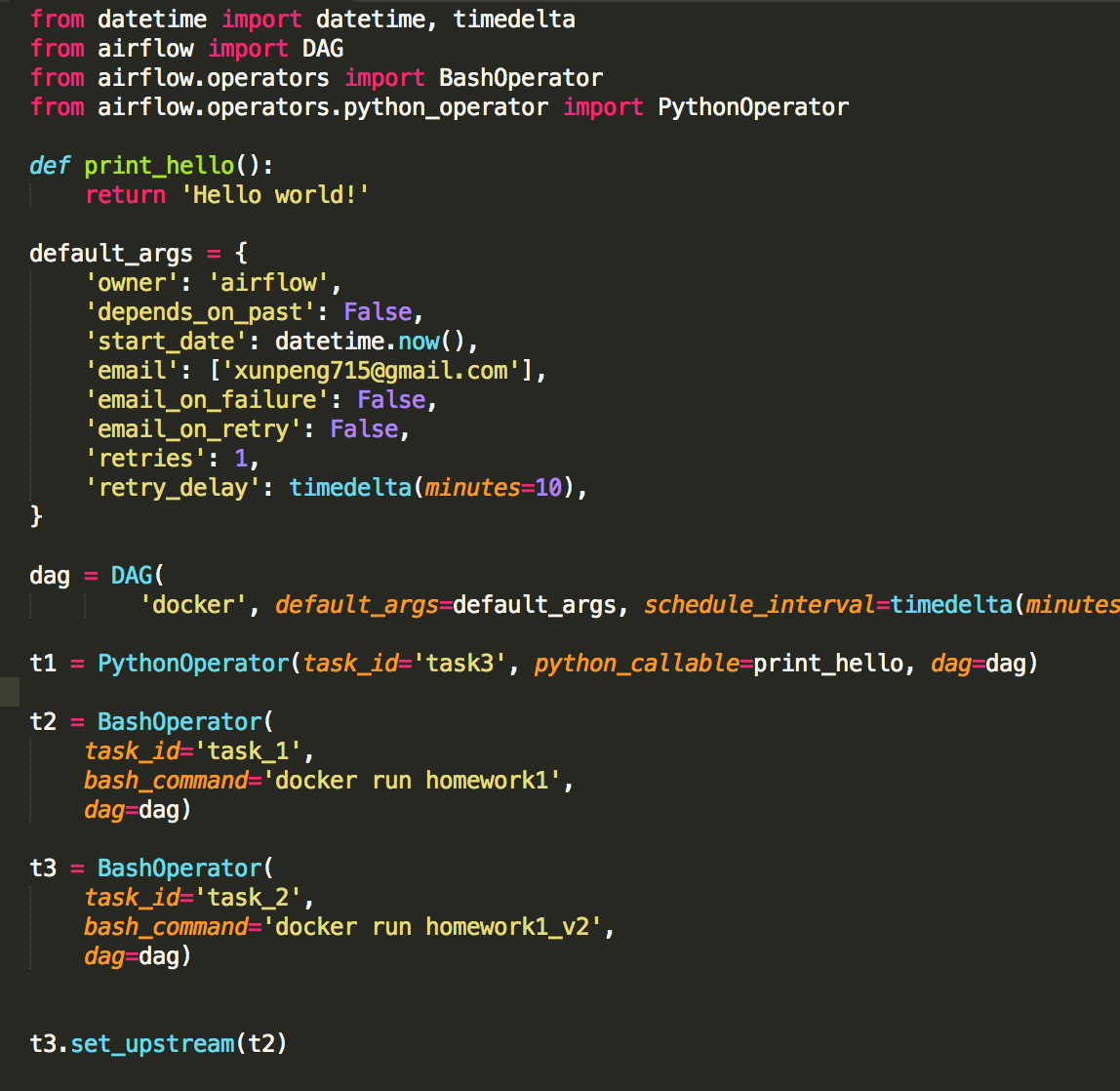


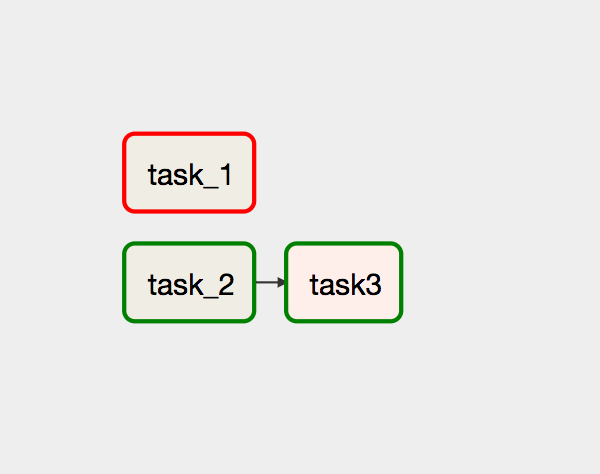
This is the directory under Jupyter. First, run “run.sh” to execute “Wrangle.py”, which utilizes wrangling techniques to clean the data and save to a new file locally, and then utilizes “configWrangle.json” to check and update the cleaned csv file on the S3 bucket. You can see “NV\_23062017\_WBAN\_53138\_clean.csv” as the data file after data cleansing. The log file stores every piece of information that we output during the execution of the python file.



The figure above shows the S3 bucket after our code is run. The last file is the csv file after data cleansing.

**Automate the task with pipelines(airflow)**

1. **install airflow and create workspace for airflow**
2. **create virtualenv directory for workspace**
3. **start the airflow web server**
4. **Under the tags directory I created “dockerScheduler.py” file to schedule two tasks of docker images**
5. **Python code to use BashOperator to schedule two docker images**
6. 
7. **We check the web UI to check that these two images running success according to schedule**



**Automation on the cloud**

**AWS Batch Job**

The configurations on AWS Batch is as follows:

* Image:
* Command: /bin/bash -c Ref::code
* Code: ./run.sh

**Send Email**

Please refer to wrangle.py for the code regarding email sending. We basically used SES to achieve that, which includes registering our email addresses on the SES service.