DSCI 525 - Web and Cloud Computing

Milestone 2: Your team is planning to migrate to the cloud. AWS gave 400($100 each) to your team to support this. As part of this initiative, your team needs to set up a server in the cloud, a collaborative environment for your team, and later move your data to the cloud. After that, your team can wrangle the data in preparation for machine learning.

Milestone 2 checklist

You will have mainly 2 tasks. Here is the checklist...

- To set up a collaborative environment
  - Setup your EC2 instance with JupyterHub.
  - Install all necessary things needed in your UNIX server (amazon ec2 instance).
  - Set up your S3 bucket.
  - Move the data that you wrangled in your last milestone to s3.
  - To move data from s3.
- Wrangle the data in preparation for machine learning
  - Get the data from S3 in your notebook and make data ready for machine learning.

Keep in mind:

- *All services you use are in region us-west-2.*
- *Don't store anything in these servers or storage that represents your identity as a student (like your student ID number) .*
- *Use only default VPC and subnet.*
- *No IP addresses are visible when you provide the screenshot.*
- *You do proper budgeting so that you don't run out of credits.*
- *We want one single notebook for grading, and it's up to your discretion on how you do it.* ***So only one person in your group needs to spin up a big instance and a `t2.xlarge` is of decent size.***
- _Please stop the instance when not in use. This can save you some bucks, but it's again up to you and how you budget your money. Maybe stop it if you or your team won't use it for the next 5 hours?
- *Your AWS lab will shut down after 3 hours 30 min. When you start it again, your AWS credentials (****access key****,****secret****, and* ***session token****) will change, and you want to update your credentials file with the new one.*
- *Say something went wrong and you want to spin up another EC2 instance, then make sure you terminate the previous one.*
- *We will be choosing the storage to be `Delete on Termination` , which means that stored data in your instance will be lost upon termination. Make sure you save any data to S3 and download the notebooks to your laptop so that next time you have your jupyterHub in a different instance, you can upload your notebook there.*

Outside of Milestone: *If you are working as an individual just to practice setting up EC2 instances, make sure you select* `t2.Large` *instance (not anything bigger than that as it can cost you money). I strongly recommend you spin up your own instance and experiment with the s3 bucket in doing something (there are many things that we learned and practical work from additional instructions and video series) to get comfortable with AWS. But we won't be looking at it for a grading purpose.*

NOTE: Everything you want for this notebook is discussed in lecture 3, lecture 4, and setup instructions.

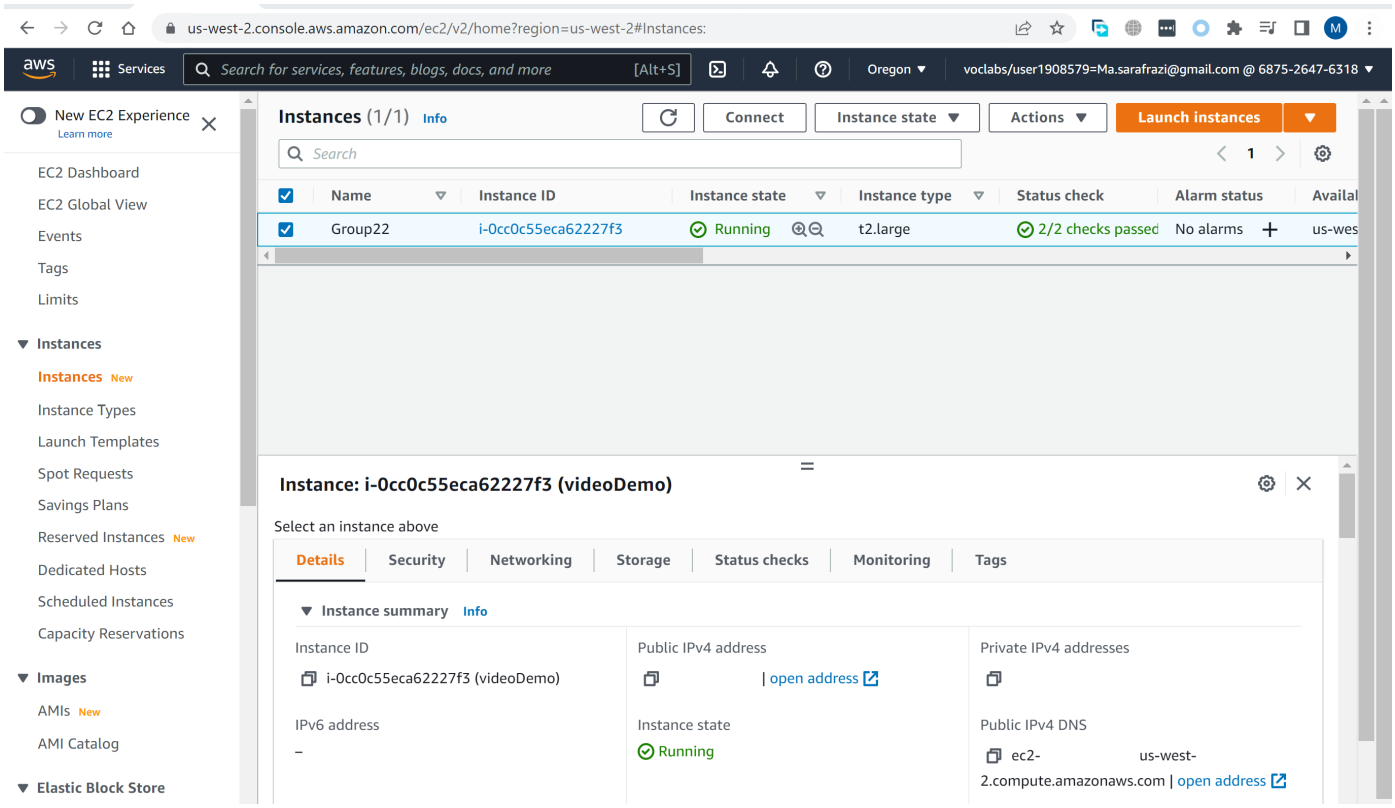
1. Setup your EC2 instance

rubric={correctness:20}

Please attach this screen shots from your group for grading.

> https://github.com/UBC-MDS/dsci_525_group_22/blob/main/screenshots/instance_EC2.png

In [11]:
```
from IPython.display import display, Image
display(Image(filename='../screenshots/instance_EC2.png'))
```



2. Setup your JupyterHub

rubric={correctness:20}

Please attach this screen shots from your group for grading

I want to see all the group members here in this screenshot

> https://github.com/UBC-MDS/dsci_525_group_22/blob/main/screenshots/Jupyter_SS.png

In [3]:
```
display(Image(filename='../screenshots/Jupyter_SS.png'))
```

Admin

User	Admin	Last Activity	Running (2)		
Add Users			Start All	Stop All	Shutdown Hub
gfairbrother	admin	a minute ago	stop server	edit user	
mahsa	admin	3 minutes ago	stop server	edit user	delete user
mwang	admin	Never	start server	edit user	delete user
rpandey	admin	Never	start server	edit user	delete user

Displaying users 1 - 4 of 4

3. Setup the server

rubric={correctness:20}

3.1) Add your team members to EC2 instance.

3.2) Setup a common data folder to download data, and this folder should be accessible by all users in the JupyterHub.

3.3)(***OPTIONAL***) Setup a sharing notebook environment.

3.4) Install and configure AWS CLI.

Please attach this screen shots from your group for grading

Make sure you mask the IP address refer here.

> https://github.com/UBC-MDS/dsci_525_group_22/blob/main/screenshots/Shared_SS.png

In [4]:
```
display(Image(filename='../screenshots/Shared_SS.png'))
```

```
jupyter-gfairbrother@ip-                :/srv/data/shared$ ls -ld .
drwxrwxrwx 2 root root 4096 Apr  7 03:14 .
```

4. Get the data what we wrangled in our first milestone.

You have to install the packages that are needed. Refer this TLJH document.Refer `pip` section.

Don't forget to add option -E. This way, all packages that you install will be available to other users in your JupyterHub. These packages you must install and install other packages needed for your wrangling.

```
sudo -E pip install pandas
sudo -E pip install pyarrow
sudo -E pip install s3fs
```

As in the last milestone, we looked at getting the data transferred from Python to R, and we have different solutions. Henceforth, I uploaded the parquet file format, which we can use moving forward.

In [5]:
```
import re
import os
import glob
import zipfile
import requests
from urllib.request import urlretrieve
```

```
import json
import pandas as pd
```

Rememeber here we gave the folder that we created in Step 3.2 as we made it available for all the users in a group.

In [6]:
```
# Necessary metadata
article_id = 14226968  # this is the unique identifier of the article on figshare
url = f"https://api.figshare.com/v2/articles/{article_id}"
headers = {"Content-Type": "application/json"}
output_directory = "/srv/data/shared/"
```

In [7]:
```
response = requests.request("GET", url, headers=headers)
data = json.loads(response.text)  # this contains all the articles data, feel free to cl
files = data["files"]             # this is just the data about the files, which is wha
files
```

Out[7]:

```
[{'id': 26844650,
  'name': 'allyears.csv.zip',
  'size': 2405908113,
  'is_link_only': False,
  'download_url': 'https://ndownloader.figshare.com/files/26844650',
  'supplied_md5': '9e046ac05ecd2c32a256a47dd1098b81',
  'computed_md5': '9e046ac05ecd2c32a256a47dd1098b81'},
 {'id': 26863682,
  'name': 'individual_years.zip',
  'size': 1896206676,
  'is_link_only': False,
  'download_url': 'https://ndownloader.figshare.com/files/26863682',
  'supplied_md5': '921da748974b07b2a70bbfcc04535a77',
  'computed_md5': '921da748974b07b2a70bbfcc04535a77'},
 {'id': 27515426,
  'name': 'combined_model_data.csv.zip',
  'size': 821308997,
  'is_link_only': False,
  'download_url': 'https://ndownloader.figshare.com/files/27515426',
  'supplied_md5': '7638434c44a7d29cbb29fe200b4fd65d',
  'computed_md5': '7638434c44a7d29cbb29fe200b4fd65d'},
 {'id': 27520682,
  'name': 'combined_model_data_parti.parquet.zip',
  'size': 519743915,
  'is_link_only': False,
  'download_url': 'https://ndownloader.figshare.com/files/27520682',
  'supplied_md5': '02f4e3df8d16580a02291de225072689',
  'computed_md5': '02f4e3df8d16580a02291de225072689'},
 {'id': 27520808,
  'name': 'combined_model_data.parquet',
  'size': 565872005,
  'is_link_only': False,
  'download_url': 'https://ndownloader.figshare.com/files/27520808',
  'supplied_md5': 'ae63699ab21ffa8006559c6afbcd2271',
  'computed_md5': 'ae63699ab21ffa8006559c6afbcd2271'}]
```


In [8]:
```
files_to_dl = ["combined_model_data_parti.parquet.zip"]  ## Please download the partitio
for file in files:
    if file["name"] in files_to_dl:
        os.makedirs(output_directory, exist_ok=True)
        urlretrieve(file["download_url"], output_directory + file["name"])
```

In [9]:
```
with zipfile.ZipFile(os.path.join(output_directory, "combined_model_data_parti.parquet.z
    f.extractall(output_directory)
```

5. Setup your S3 bucket and move data

rubric={correctness:20}

5.1) Create a bucket name should be mds-s3-xxx. Replace xxx with your "groupnumber".

5.2) Create your first folder called "output".

5.3) Move the "observed_daily_rainfall_SYD.csv" file from the Milestone1 data folder to your s3 bucket from your local computer.

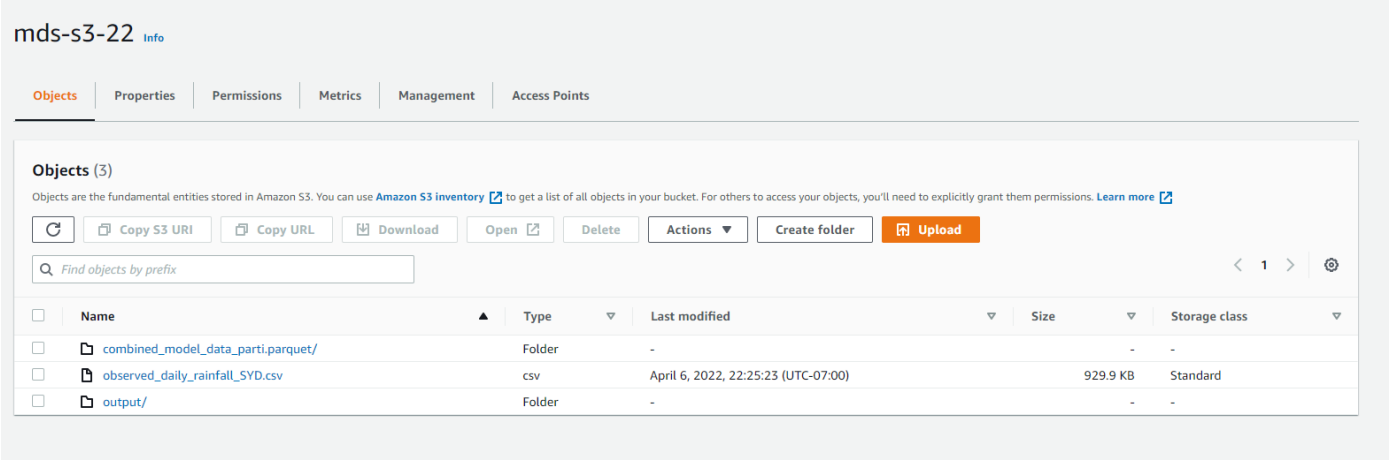
5.4) Moving the parquet file we downloaded(combined_model_data_parti.parquet) in step 4 to S3 using the cli what we installed in step 3.4.

Please attach this screen shots from your group for grading

Make sure it has 3 objects.

> https://github.com/UBC-MDS/dsci_525_group_22/blob/main/screenshots/S3.png

In [10]:
```
display(Image(filename='../screenshots/S3.png'))
```



6. Wrangle the data in preparation for machine learning

rubric={correctness:20}

Our data currently covers all of NSW, but say that our client wants us to create a machine learning model to predict rainfall over Sydney only. There's a bit of wrangling that needs to be done for that:

1. We need to query our data for only the rows that contain information covering Sydney
2. We need to wrangle our data into a format suitable for training a machine learning model. That will require pivoting, resampling, grouping, etc.

To train an ML algorithm we need it to look like this:

	model-1_rainfall	model-2_rainfall	model-3_rainfall	...	observed_rainfall
0	0.12	0.43	0.35	...	0.31
1	1.22	0.91	1.68	...	1.34
2	0.68	0.29	0.41	...	0.57

6.1) Get the data from s3 (`combined_model_data_parti.parquet` and `observed_daily_rainfall_SYD.csv`)

6.2) First query for Sydney data and then drop the lat and lon columns (we don't need them).

```
syd_lat = -33.86
syd_lon = 151.21
```

Expected shape `(1150049, 2)` .

6.3) Save this processed file to s3 for later use:

Save as a csv file `ml_data_SYD.csv` to `s3://mds-s3-xxx/output/` expected shape `(46020,26)` - This includes all the models as columns and also adding additional column `Observed` loaded from `observed_daily_rainfall_SYD.csv` from s3.

In [1]:
```
import pandas as pd
import os

# Define file path
s3_bucket = "s3://mds-s3-22/"
model_file = os.path.join(s3_bucket, "combined_model_data_parti.parquet")
obs_file = os.path.join(s3_bucket, "observed_daily_rainfall_SYD.csv")
out_file = os.path.join(s3_bucket, "output", "ml_data_SYD.csv")
```

In [2]:
```
# Read data from S3 and apply filter
columns = ["time", "rain (mm/day)", "model"]
model_df = pd.read_parquet(
    model_file,
    filters=[
        [
            ("lat_min", "<=", -33.86),
            ("lat_max", ">=", -33.86),
            ("lon_min", "<=", 151.21),
            ("lon_max", ">=", 151.21),
        ]
    ],
    columns=columns,
)
# set index to time
model_df = model_df.set_index("time")

# Read obs file
obs_df = pd.read_csv(obs_file, parse_dates=["time"], index_col="time")

# print shape of model df
print(model_df.shape)

# change format of index to only date
model_df.index = model_df.index.date
obs_df.index = obs_df.index.date
```

```
(1150049, 2)
```

In [3]:
```
# change name of index to time
model_df.index.name = "time"
obs_df.index.name = "time"

# pivot table to get the rainfall for each day and model
```

```
model_df = model_df.pivot_table(index="time", columns="model", values="rain (mm/day)")

# print shape of model df
print(model_df.shape)

# show model df
model_df
```

```
(46020, 25)
```

Out[3]:

model	ACCESS-CM2	ACCESS-ESM1-5	AWI-ESM-1-1-LR	BCC-CSM2-MR	BCC-ESM1	CMCC-CM2-HR4	CMCC-CM2-SR5	CMCC-ESM2	CanE
time									
1889-01-01	0.040427	1.814552	3.557934e+01	4.268112e+00	1.107466e-03	1.141054e+01	3.322009e-08	2.668800	1.32
1889-01-02	0.073777	0.303965	4.596520e+00	1.190141e+00	1.015323e-04	4.014984e+00	1.312700e+00	0.946211	2.78
1889-01-03	0.232656	0.019976	5.927467e+00	1.003845e-09	1.760345e-05	9.660565e+00	9.103720e+00	0.431999	0.00
1889-01-04	0.911319	13.623777	8.029624e+00	8.225225e-02	1.808932e-01	3.951528e+00	1.317160e+01	0.368693	0.01
1889-01-05	0.698013	0.021048	2.132686e+00	2.496841e+00	4.708019e-09	2.766362e+00	1.822940e+01	0.339267	0.00
...	...	...	...	...	...	...	...	...	
2014-12-27	0.033748	0.123476	1.451179e+00	3.852845e+01	2.061717e-03	8.179260e-09	1.171263e-02	0.090786	59.89
2014-12-28	0.094198	2.645496	4.249335e+01	5.833801e-01	5.939502e-09	8.146937e-01	4.938899e-01	0.000000	0.51
2014-12-29	0.005964	3.041667	2.898325e+00	9.359547e-02	2.000051e-08	2.532205e-01	1.306046e+00	0.000002	37.16
2014-12-30	0.000028	1.131412	2.516381e-01	1.715028e-01	7.191735e-05	8.169252e-02	1.722262e-01	0.788577	7.36
2014-12-31	0.532747	2.370896	1.047835e-13	4.437736e+00	2.863683e-01	6.343592e+00	6.368303e-01	0.442130	0.30

46020 rows × 25 columns

In [4]:

```
# rename observed rainfall to observed
obs_df.rename(columns={"rain (mm/day)": "observed"}, inplace=True)

# merge model and obs df
df = pd.merge(model_df, obs_df, how="left", left_index=True, right_index=True)

# print shape of df
print("Shape of final df: ", df.shape)

# show df
df
```

```
Shape of final df:  (46020, 26)
```

Out[4]:

time	ACCESS-CM2	ACCESS-ESM1-5	AWI-ESM-1-1-LR	BCC-CSM2-MR	BCC-ESM1	CMCC-CM2-HR4	CMCC-CM2-SR5	CMCC-ESM2	CanES
1889-01-01	0.040427	1.814552	3.557934e+01	4.268112e+00	1.107466e-03	1.141054e+01	3.322009e-08	2.668800	1.321
1889-01-02	0.073777	0.303965	4.596520e+00	1.190141e+00	1.015323e-04	4.014984e+00	1.312700e+00	0.946211	2.788
1889-01-03	0.232656	0.019976	5.927467e+00	1.003845e-09	1.760345e-05	9.660565e+00	9.103720e+00	0.431999	0.003
1889-01-04	0.911319	13.623777	8.029624e+00	8.225225e-02	1.808932e-01	3.951528e+00	1.317160e+01	0.368693	0.013
1889-01-05	0.698013	0.021048	2.132686e+00	2.496841e+00	4.708019e-09	2.766362e+00	1.822940e+01	0.339267	0.002
...	...	...	...	...	...	...	...	...	
2014-12-27	0.033748	0.123476	1.451179e+00	3.852845e+01	2.061717e-03	8.179260e-09	1.171263e-02	0.090786	59.895
2014-12-28	0.094198	2.645496	4.249335e+01	5.833801e-01	5.939502e-09	8.146937e-01	4.938899e-01	0.000000	0.512
2014-12-29	0.005964	3.041667	2.898325e+00	9.359547e-02	2.000051e-08	2.532205e-01	1.306046e+00	0.000002	37.169
2014-12-30	0.000028	1.131412	2.516381e-01	1.715028e-01	7.191735e-05	8.169252e-02	1.722262e-01	0.788577	7.361
2014-12-31	0.532747	2.370896	1.047835e-13	4.437736e+00	2.863683e-01	6.343592e+00	6.368303e-01	0.442130	0.306

46020 rows × 26 columns

In [5]:

```
# write df to csv
df.to_csv(out_file)
print(f"Successfully written to csv file: {out_file}")
```

```
Successfully written to csv file: s3://mds-s3-22/output/ml_data_SYD.csv
```

How the final file format looks like https://github.ubc.ca/mds-2021-22/DSCI_525_web-cloud-comp_students/blob/master/release/milestone2/image/finaloutput.png

Shape `(46020,26 )`

> https://github.com/UBC-MDS/dsci_525_group_22/blob/main/screenshots/6_data_wrangle.png

(***OPTIONAL***) If you are interested in doing some benchmarking!! How much time it took to read..

- Parquet file from your local disk ?
- Parquet file from s3 ?
- CSV file from s3 ? For that, upload the CSV file (`combined_model_data.csv`)to S3 and try to read it instead of parquet.

In [ ]: