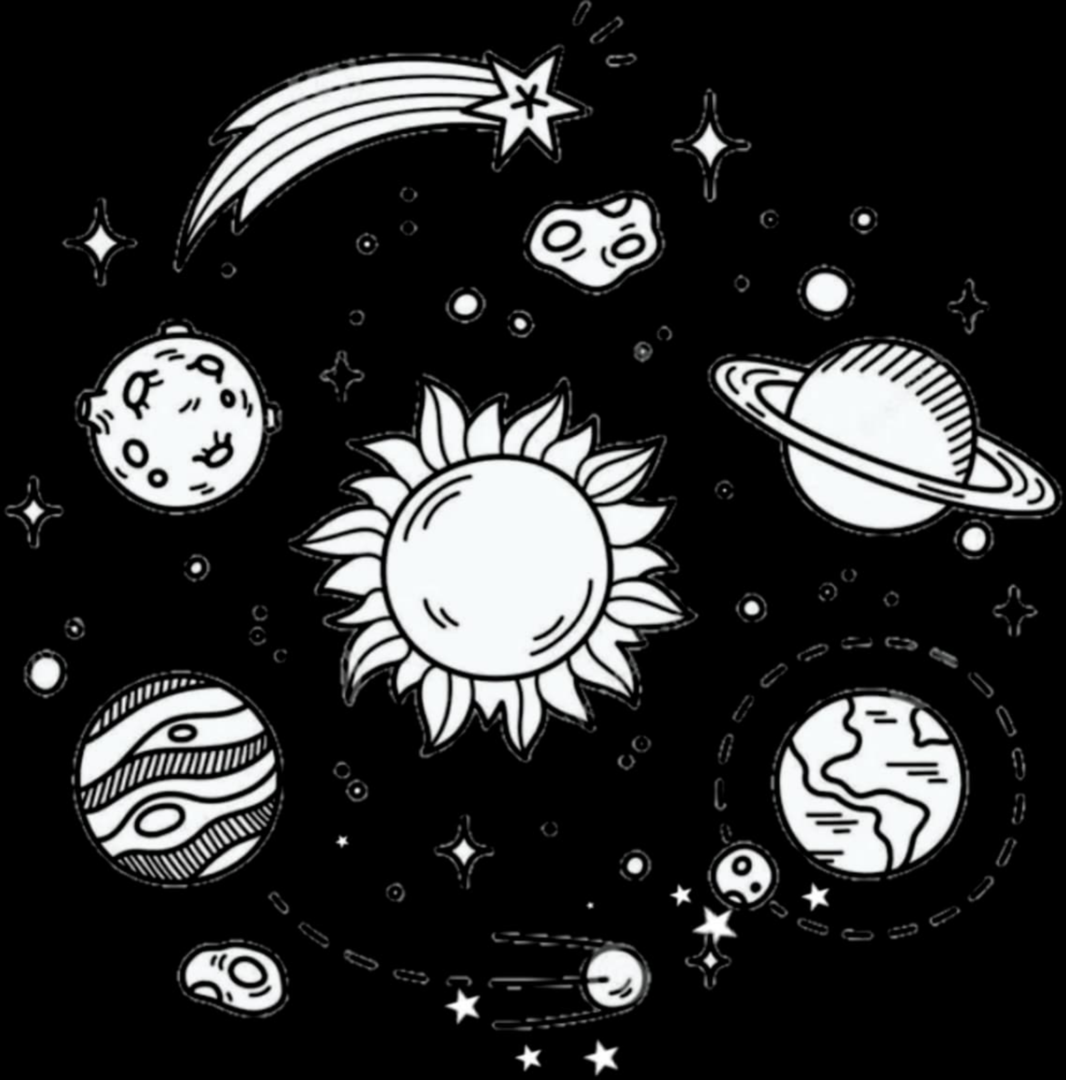
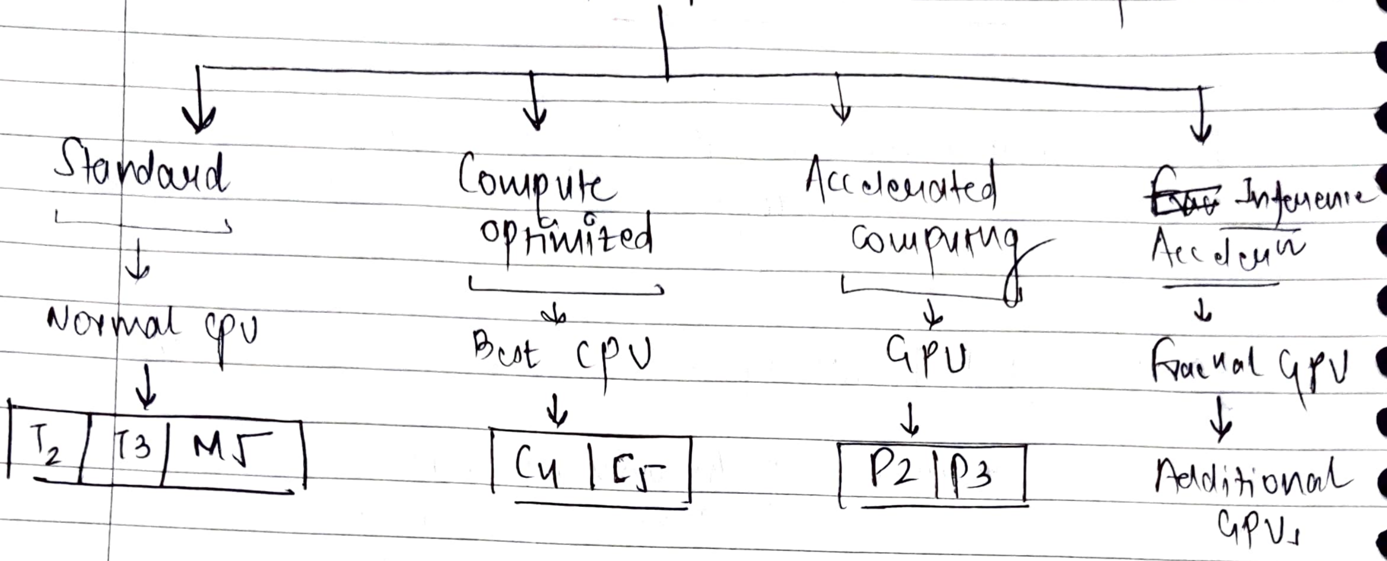


aws

INTRO TO AWS SAGEMAKER

LAYMAN NOTES
IN A TERRIBLE HANDWRITING
(DIVYANSHU VYAS)

→ Sagemaker offers different kinds of instances based on CPU, GPU & additional size parameters.



Naming convention : C5 .2x large

C → instance type (compute optimized)

5 → hardware version

large → size.

* Sagemaker Free Tier → ① Development = 250 Hrs t2.medium t3.medium

② Train = 50 Hrs/month m4.xlarge | m5.xlarge

③ Deploy = 125 Hrs m4xlarge | m5.xlarge

* Sagemaker Instance Pricing is a fn of Time & disk space required by instance for training.

Kinds of Deployment + Inferences

① Real-Time Inference : Very low (<1sec) latency desired

└→ Deploy on multiple instances + Fractional GPUs.

② Input → Run on Batch (entire data) → Output Type
(Not Real time, high latency is Okay!)

└→ Batch Transform

↳ lower costs in total since total Run time is less.

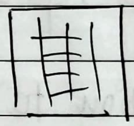
DATA FORMATS supported by Sagemaker (for training).

- Training Data format :- CSV | JSON
  - RecordIO | Parquet
  - LibSVM

└→ Data needs to be stored in S3

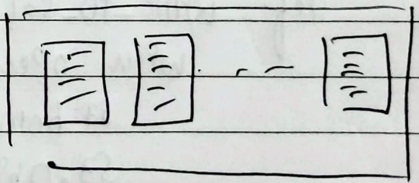
- Inference → CSV | JSON | RecordIO

Training data storage styles →



in 1 file

OR



split across several files in a folder.

Modes of Data (Training) transfer from S3 to Train Instance

① File Mode	② Pipe Mode
- entire file copied to instance	- streaming
- Needs more space	- efficient
→ slow	- regular inflow

What is boto3

↳ its an Aws provided library for interacting with Aws services.

↳ Sagemaker library - to interact with sagemaker services.

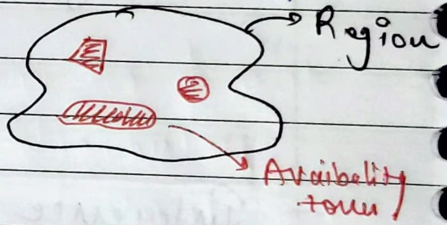
LAB : DATA Inflow Outflow

① create a df using pandas.

② df.to_csv('demo.csv', index=False)

↳ Stores file in the Notebook Instance folder only.

③ Now writing to S3



- Files are called "OBJECTS" in S3.
- File name == Key Name in S3.
- Files are replicated across 3+ Availability Zones in same Region.

Helper function to write to S3 Bucket ↓

```
def write_to_s3(filename, bucket, key):
    with open(filename, 'rb') as f:
        s3=boto3.Session().resource('s3').Bucket(bucket)
        s3.Object(key).upload_fileobj(f).
```

- Creating A Python S3 Variable (to interact with S3 codewise)

```
>> s3 = boto3.Session().resource('s3')
```

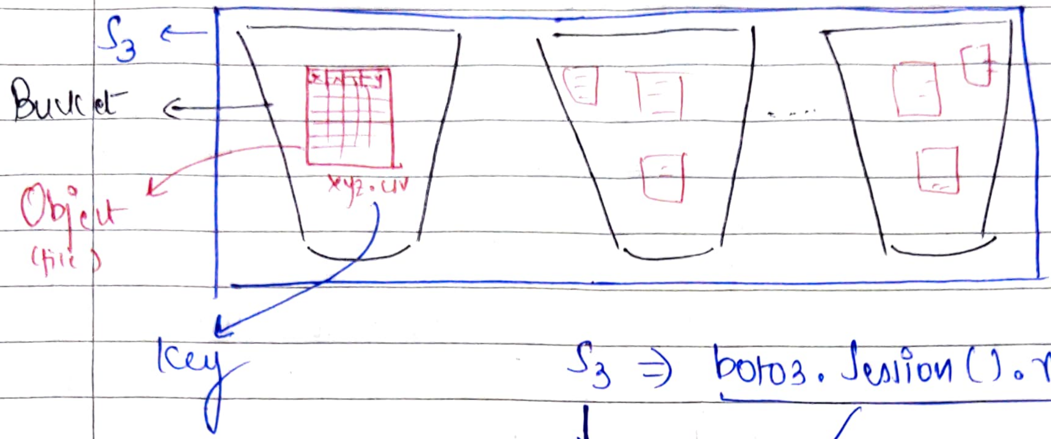
- List All bucket names in S3

```
[buck.name for buck in s3.buckets.all()]
```

- Listing All files in a given bucket.

→ bucket_obj = s3.Bucket(bucket_name)

→ [obj.key for obj in buck_obj.Objects.all()].



S3 ⇒ boto3.Session().resource('s3')

↓

Particular Bucket ⇒ ⌴ .Bucket("dv-ml-bucket")

↓

Particular file ? ⌴ .Object(filename)

⌴ .get()

- Saving a (CSV file) that you're working on to CSV in S3

Pandas DF

→ from io import StringIO

→ csv_buffer = StringIO()

→ df.to_csv(csv_buffer)

→ output_file_object = s3.Object('bucketname', 'output.csv')

→ output_file_object.put(Body = csv_buffer.getvalue())

Homework (Need Based) : RecordIO format