Sagemaker

Amazon SageMaker is a widely used cloud machine-learning platform and is defined as a platform that enables developers to create, train, and deploy machine learning (ML) models in the cloud

- The Amazon SageMaker Studio provides a single, web-based visual interface where all Machine Learning development steps can be performed
- At the most basic level, SageMaker provides Jupyter notebooks like interface. You can use these notebooks for building, training and deploying ML models.
    - What is the advantage of using SageMaker notebooks instead of local or notebooks hosted on an EC2 server somewhere? Well, SageMaker lets you decide the type of machine you prefer so you don't need to manage any complex AMIs or security groups — this makes it very easy to get started. SageMaker also provides access to GPUs and big machines with high amounts of RAM that might not be possible on a local setup.

- Yet another advantage is how you can use Amazon's own pre-built models that have been highly optimized to run on AWS services. These models come pre-built and you do not need to do much to build and check the model. You can use prebuilt XGBoost or LDA or PCA or Seq to Seq models — all these are available via high level Python SDK called sagemaker
    - One great thing about SageMaker is its modular design. If you prefer to train elsewhere and just use SageMaker for deployment, you can do that. If you just prefer to train your model and use its hyper-parameter tuning capability you can do that as well.
- It delivers High Performance.
- great integration with other AWS services like Dynamo DB or S3
- a very big reason to use sagemaker is that if your data is stored in s3 and you train or explore in colab there will be significant cost of data retrival, but within AWS services this is not the case.

Steps:

1. **Create a Noteboook Instance:**
    - Open the SageMaker console (https://console.aws.amazon.com/sagemaker/).
    - Navigate to Notebook instances and click Create notebook instance.
    - Choose a suitable instance type (consider factors like model size, complexity, and expected traffic).
    - Specify a name, IAM role (with S3 access permissions), and security group.

    - Optionally, configure networking, lifecycle settings, and storage.
    - Click Create.
2. **Start the Notebook Instance:**
    - Under Notebook instances, locate your instance and click the Start button.
    - Once started, access the Jupyter notebook by clicking the link.
3. **Prepare the Model:**
    - Save your trained model in a compatible format (e.g., TensorFlow .pb, PyTorch .pt, XGBoost model file).
4. **Upload the Model to S3:**
    - Create an S3 bucket in your region.
    - Using the notebook instance, upload the compressed model archive to the bucket.
5. **Create an Endpoint:**
    - From the notebook deploy the model created with sagemaker model estimator

Testing the Deployed Model:

- Once the endpoint is in service (active), use the provided invocation endpoint URL and appropriate request format (JSON, CSV, etc.) to send test data to the model.
- Verify that the model returns the expected predictions or outputs using the boto3 api

Code for accessing sagemaker endpoint outside of sagemaker

```
tst=test data that you have

runtime =
boto3.Session().client('sagemaker-runtime',region_name='us-east-1',
                  aws_access_key_id= 'your aws_access_key_id',
                  aws_secret_access_key='your aws_secret_access_key' )


response = runtime.invoke_endpoint(
    EndpointName='your endpoint name', ContentType="text/csv", Body=tst
)
result = response["Body"].read().decode("ascii")
print("Predicted Class Probabilities: {}.".format(result))
```