Cloud Deployment Process

1. **Import relevant libraries**

```
## import libarary
import os
import pandas as pd
import numpy as np

#import google cloud library
from google.cloud import bigquery
from google.cloud import storage
from google.cloud import aiplatform
import db_dtypes

## sklearn module
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, fbeta_score
from sklearn.preprocessing import LabelEncoder
import pickle
```

- bigquery : connect with BigQuery
- storage : connect with Google Cloud Storage
- aiplatform : connect with Vertex AI
- db_dbtypes : allows proper representation of data types unique to BigQuery in Python

2. **Authenticate credentials to big query with JSON file**

```
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "trial_bigq.json"
```

3. **Upload data file**

```
project_id = 'dtidsus'
dataset_id = 'capstone'
table_id = 'data_telco_customer_churn'
region = 'us-central1'
bucket_name = 'modul4'
blob_name = 'sekar/data_telco_customer_churn.csv'
```

```
client = bigquery.Client(project='dtidsus')
```

```
try :
    storage_client = storage.Client(project=project_id)
    bucket = storage_client.get_bucket(bucket_name) # Add bucket name
    data_capstone = bucket.blob('sekar/data_telco_customer_churn.csv')
    data_capstone.upload_from_filename('data/data_telco_customer_churn.csv')

    print ("Uploading succeeded")
except:
    raise TypeError("An exception occurred")
```

4. **Access data from BigQuery**

```
try :
    storage_client = storage.Client(project=project_id)
    bucket = storage_client.get_bucket(bucket_name) # Add bucket name
    data_capstone = bucket.blob('sekar/data_telco_customer_churn.csv')
    data_capstone.download_from_filename('data/data_telco_customer_churn.csv')

    print ("Downloading succeeded")
except:
    raise TypeError("An exception occurred")
```

```
Downloading model succeeded
```

```
client = bigquery.Client('dtidsus')
```

```
query_job = client.query(f"""select * from {dataset_id}.{table_id}""")
```

```
df = query_job.result().to_dataframe()
```

5. **Upload model to Google Cloud Storage**

```
try :
    storage_client = storage.Client(project=project_id)
    bucket = storage_client.get_bucket(bucket_name) # Add bucket name
    blob_model = bucket.blob('sekar/model/model.pkl')
    blob_model.upload_from_filename('model.pkl')

    print ("Uploading model succeeded")
except:
    raise TypeError("An exception occurred")
```

```
Uploading model succeeded
```

6. **Authenticate credentials to ai platform with JSON file**

```
os.environ['GOOGLE_APPLICATION_CREDENTIALS'] = "dev_trial.json"
```

7. **Uploading model to Vertex AI**

```
aiplatform.init(project='dtidsus', location='us-central1')

model = aiplatform.Model.upload(
    display_name='sekar_model_000',
    artifact_uri="gs://modul4/sekar/model/",
    serving_container_image_uri="us-docker.pkg.dev/vertex-ai/prediction/sklearn-cpu.1-2:latest",
)

model.wait()
```

```
Creating Model
Create Model backing LRO: projects/41965541199/locations/us-central1/models/6296970162482446336/operations/1990588183439474688
Model created. Resource name: projects/41965541199/locations/us-central1/models/6296970162482446336@1
To use this Model in another session:
model = aiplatform.Model('projects/41965541199/locations/us-central1/models/6296970162482446336@1')
```

6296970162482446336@1 is the unique number to access your model

8. **Creating Endpoint**

```
endpoint = aiplatform.Endpoint.create(
    display_name="sekar-endpoint-000",
    project='dtidsus',
    location='us-central1',
)
```

```
Creating Endpoint
Create Endpoint backing LRO: projects/41965541199/locations/us-central1/endpoints/8605192514858647552/operations/1211465447904378880
Endpoint created. Resource name: projects/41965541199/locations/us-central1/endpoints/8605192514858647552
To use this Endpoint in another session:
endpoint = aiplatform.Endpoint('projects/41965541199/locations/us-central1/endpoints/8605192514858647552')
```

9. **Endpoint Deployment**

```
min_replica_count: int = 1
max_replica_count: int = 1
```

```
endpoint.deploy(
    model=model,
    deployed_model_display_name='sekar_model_000',
    machine_type='e2-standard-2',
    min_replica_count=min_replica_count,
    max_replica_count=max_replica_count,
    sync=True,
)
```

10. Use endpoint to make prediction

```
## predict your data with online prediction here
PROJECT_ID = 'dtidsus'
ENDPOINT_ID = "projects/41965541199/locations/us-central1/endpoints/endpoint_number"
REGION = 'us-central1'
```

```
predict_inst = ['Yes', 5, 'No', 'No', 'DSL', 'Yes', 'Yes', 'One year', 'No', 20]
```

```
aiplatform.init(project=PROJECT_ID, location=REGION)
endpoint = aiplatform.Endpoint(ENDPOINT_ID)
prediction = endpoint.predict(instances=[predict_inst])

print("PREDICTION:", prediction)
```