NAME : Naviya Dharshini
Roll no : 23AD083

DAY -3

CLOUDWATCH , LAMBDA,SAGEMAKER

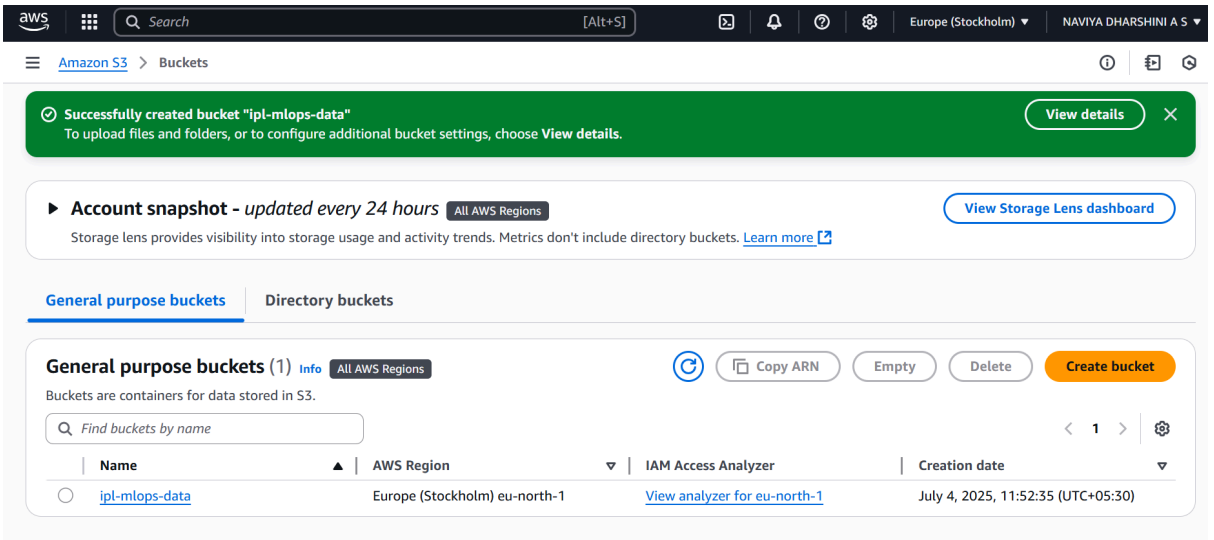
TECH STACK:

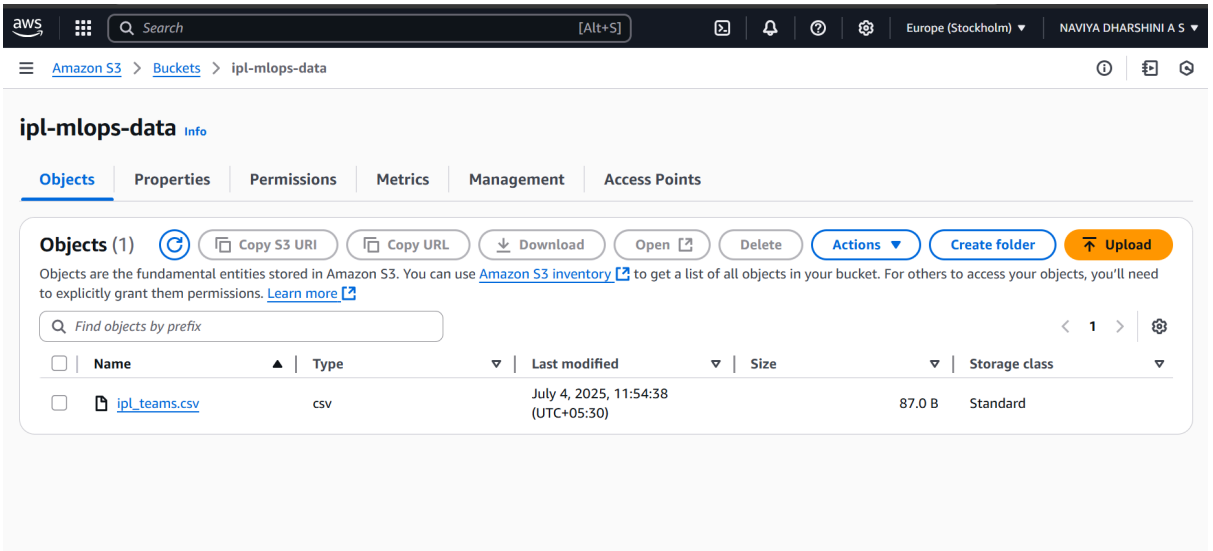
- S3: Storage for raw & processed data
- Lambda: Preprocessing & Inference
- SageMaker Notebook: Model training
- API Gateway: Serve predictions
- IAM: Permissions & roles
- CloudWatch: Debug logs

✅ STEP 1: S3 Setup

Buckets You'll Use

- ipl-mlops-data – Raw and processed data
- Optional: football-mlops-data (if you clone for football later)





Files to Upload

- ipl_data.csv → Format:

csv

CopyEdit

team,ipl_wins,ipl_team_score

CSK,5,3000

MI,5,3200

RCB,0,2900

LSG,6,3000 <-- Add this line manually for Exercise 1

🧠 STEP 2: IPLDataPreProcessor Lambda

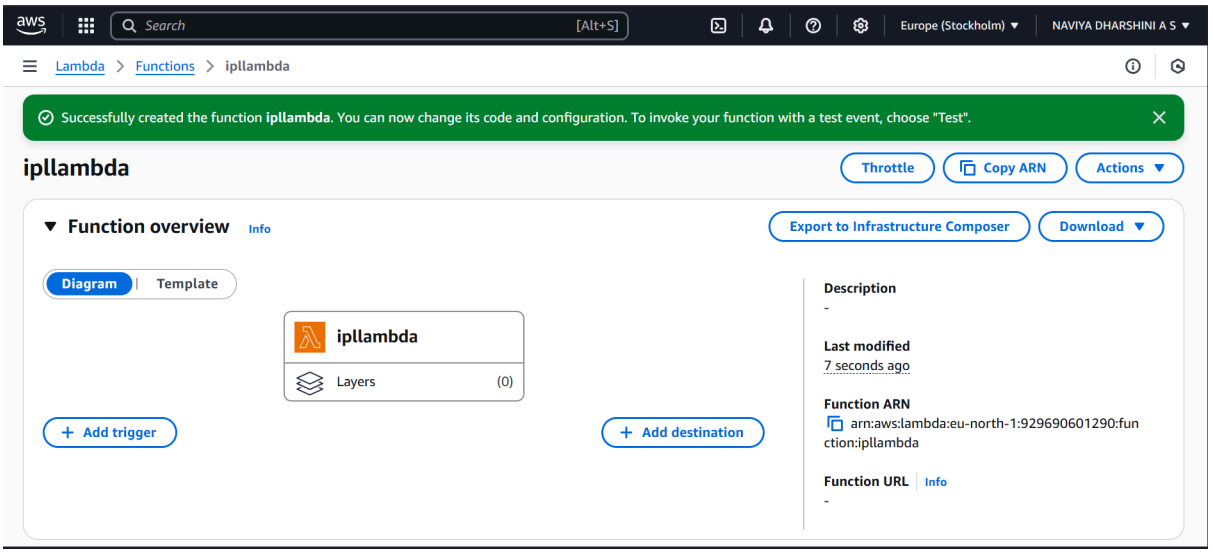
Code :

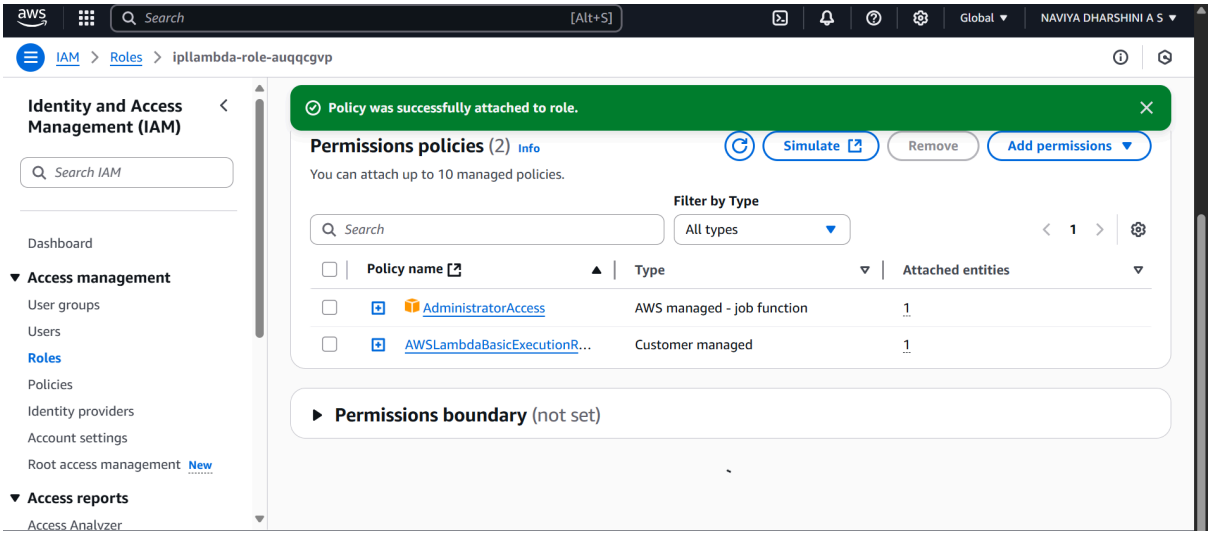
```
import json
import boto3
import os
import csv
from io import StringIO

s3_client = boto3.client('s3')

def lambda_handler(event, context):
    bucket_name = os.environ.get('S3_BUCKET_NAME')
    input_file_key = 'ipl_teams.csv'
    output_file_key = 'processed_ipl_data.csv'

    print(f"Reading {input_file_key} from bucket {bucket_name}")

    try:
        response = s3_client.get_object(Bucket=bucket_name,
Key=input_file_key)
        csv_content = response['Body'].read().decode('utf-8')
```

```
        # Use StringIO to treat the string as a file for csv.reader
        csv_file = StringIO(csv_content)
        reader = csv.reader(csv_file)
        header = next(reader) # Read header row
        data = list(reader)  # Read remaining data rows

        # Add 'can_win' to header
        header.append('can_win')

        processed_rows = []
        for row in data:
            ipl_wins = int(row[1]) # ipl_wins is the second column (index 1)
            ipl_team_score = int(row[2]) # ipl_team_score is the third column
(index 2)

            # Simple preprocessing logic
            can_win = 1 if (ipl_wins >= 3 and ipl_team_score >= 2800) else 0
            new_row = row + [str(can_win)] # Append the new value
            processed_rows.append(new_row)

        # Prepare content for writing back to S3
        output_csv_file = StringIO()
        writer = csv.writer(output_csv_file)
        writer.writerow(header) # Write header
        writer.writerows(processed_rows) # Write processed rows
        processed_csv_content = output_csv_file.getvalue()

        s3_client.put_object(Bucket=bucket_name, Key=output_file_key,
Body=processed_csv_content)

        print(f"Successfully processed {input_file_key} and saved to
{output_file_key} in {bucket_name}")

        return {
            'statusCode': 200,
            'body': json.dumps(f'Successfully processed data and stored in
s3://{bucket_name}/{output_file_key}')
        }
    except Exception as e:
        print(f"Error processing data: {e}")
        return {
```

```
        'statusCode': 500,
        'body': json.dumps(f'Error processing data: {str(e)}')
    }
```

Output :





- Role & Permissions:
  - IAM Role: IPLDataPreProcessor-role-xxxx
  - Attach: AmazonS3FullAccess (Edit this in Exercise 3)
  - Trust policy allows Lambda to assume this role.

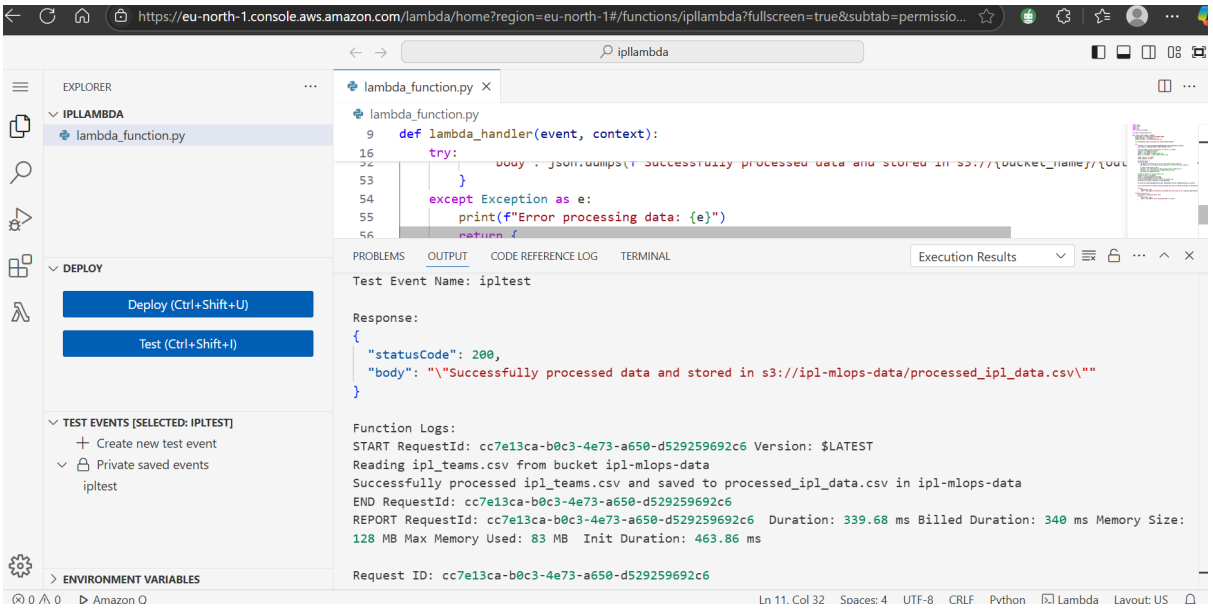
- Python Code (lambda_function.py)

python

CopyEdit

```
import boto3

import csv

import io


def lambda_handler(event, context):

    s3 = boto3.client('s3')

    bucket = 'ipl-mlops-data'

    input_key = 'ipl_data.csv'

    output_key = 'processed_ipl_data.csv'


    # Download raw CSV

    raw_obj = s3.get_object(Bucket=bucket, Key=input_key)

    raw_data = raw_obj['Body'].read().decode('utf-8').splitlines()


    reader = csv.DictReader(raw_data)

    output = []


    for row in reader:

        team = row['team']

        wins = int(row['ipl_wins'])

        score = int(row['ipl_team_score'])
```

```
        can_win = 1 if wins >= 3 and score >= 2800 else 0


        output.append({'team': team, 'ipl_wins': wins, 'ipl_team_score': score, 'can_win': can_win})


    # Write back processed CSV

    output_io = io.StringIO()

    writer = csv.DictWriter(output_io, fieldnames=['team', 'ipl_wins', 'ipl_team_score', 'can_win'])

    writer.writeheader()

    for row in output:

        writer.writerow(row)


    s3.put_object(Bucket=bucket, Key=output_key, Body=output_io.getvalue())

    return {'statusCode': 200, 'body': 'Preprocessing completed'}
```



📓 STEP 3: SageMaker Notebook – Train Your Model

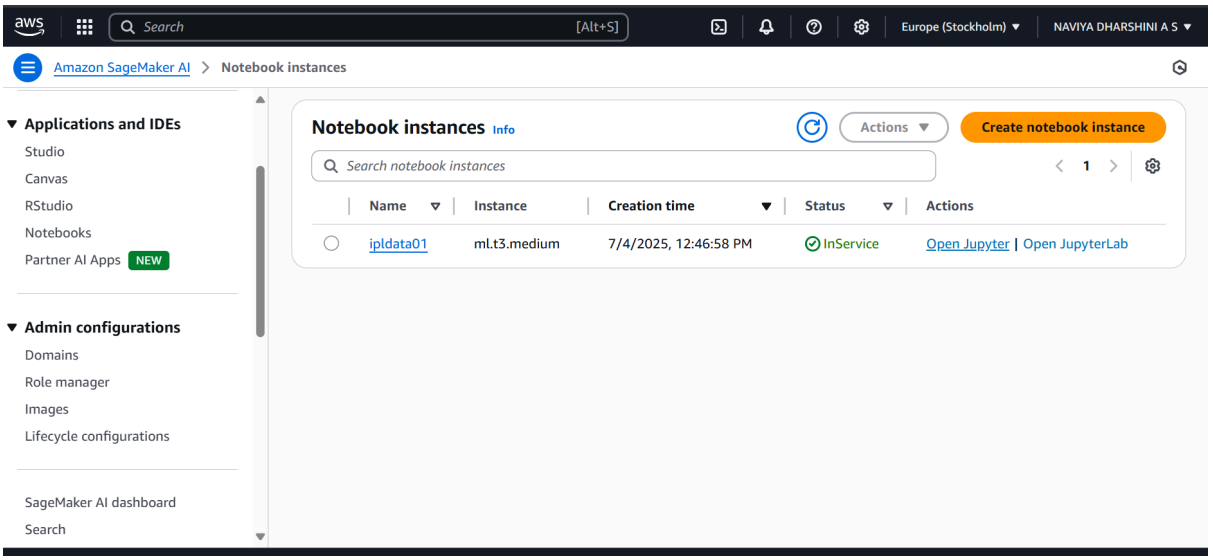
Code :

```
# Line 1: Install scikit-learn if it's not already available in the environment
!pip install scikit-learn
!pip install pandas

import pandas as pd
import boto3
import io
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import joblib # For saving/loading models

# Line 1: Define your S3 bucket name
bucket_name = 'ipl-mlops-data' # REPLACE with your S3 bucket name
# Line 2: Define the key for the processed data file
processed_data_key = 'processed_ipl_data.csv'
# Line 3: Define the key for saving the trained model
model_key = 'ipl_winner_predictor.joblib'

# Line 4: Initialize S3 client
s3 = boto3.client('s3')

print(f"Downloading {processed_data_key} from {bucket_name}...")
# Line 5: Get the object from S3
obj = s3.get_object(Bucket=bucket_name, Key=processed_data_key)
# Line 6: Read the object body and decode it
body = obj['Body'].read().decode('utf-8')
# Line 7: Use io.StringIO to read the string content as a CSV file
df = pd.read_csv(io.StringIO(body))

print("Data loaded successfully:")
print(df.head())
print(df.info())

# Line 1: Define features (X) and target (y)
X = df[['ipl_wins', 'ipl_team_score']] # Features for prediction
y = df['can_win'] # Target variable

print("Features (X) head:")
print(X.head())
print("Target (y) head:")
```

```
print(y.head())

# Line 2: Split data into training and testing sets (80% train, 20% test)
# random_state for reproducibility
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

print(f"Training data shape: {X_train.shape}")
print(f"Testing data shape: {X_test.shape}")

# Line 1: Initialize the Logistic Regression model
model = LogisticRegression(random_state=42)

print("Training model...")
# Line 2: Train the model using the training data
model.fit(X_train, y_train)

print("Model training complete.")

# Line 3: Evaluate the model on the test set
accuracy = model.score(X_test, y_test)
print(f"Model accuracy on test set: {accuracy:.2f}")

# Line 1: Save the trained model locally as a joblib file
local_model_path = '/tmp/ipl_winner_predictor.joblib' # SageMaker notebook instances have
/tmp for temporary storage
joblib.dump(model, local_model_path)

print(f"Model saved locally to {local_model_path}")

# Line 2: Upload the saved model to S3
s3.upload_file(local_model_path, bucket_name, model_key)

print(f"Model uploaded to s3://{bucket_name}/{model_key}")
```

Output:

```
  from pandas.core.computation.check import NUMEXPR_INSTALLED
Downloading processed_ipl_data.csv from ipl-mlops-data...
Data loaded successfully:
  ipl_team  ipl_wins  ipl_team_score  can_win
0      CSK         4            2850        1
1       MI         5            2900        1
2      RCB         0            2700        0
3      SRH         2            2750        0
4      KKR         3            2800        1
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5 entries, 0 to 4
Data columns (total 4 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   ipl_team        5 non-null      object
 1   ipl_wins        5 non-null      int64
 2   ipl_team_score  5 non-null      int64
 3   can_win         5 non-null      int64
dtypes: int64(3), object(1)
memory usage: 288.0+ bytes
None
Features (X) head:
   ipl_wins  ipl_team_score
0         4            2850
1         5            2900
2         0            2700
3         2            2750
4         3            2800
Target (y) head:
0    1
1    1
2    0
3    0
4    1
Name: can_win, dtype: int64
Training data shape: (4, 2)
Testing data shape: (1, 2)
Training model...
Model training complete.
Model accuracy on test set: 1.00
```



Setup:

1. Start a new notebook instance (e.g., ipl-sagemaker-notebook).
2. Attach IAM Role with AmazonS3FullAccess and AmazonSageMakerFullAccess.

Notebook Code Outline (Python):

python

CopyEdit

```
import pandas as pd

import boto3

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

import joblib


# Load processed CSV from S3

bucket = 'ipl-mlops-data'

key = 'processed_ipl_data.csv'

s3 = boto3.client('s3')

obj = s3.get_object(Bucket=bucket, Key=key)

df = pd.read_csv(obj['Body'])


# Model training

X = df[['ipl_wins', 'ipl_team_score']]

y = df['can_win']

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42)


clf = RandomForestClassifier()

clf.fit(X_train, y_train)
```

```
# Save and upload model

joblib.dump(clf, 'ipl_model.joblib')

s3.upload_file('ipl_model.joblib', bucket, 'ipl_model.joblib')
```

🧠 STEP 4: IPLPredictionInference Lambda

🔹 Permissions:

- Add AmazonS3ReadOnlyAccess

🔹 Python Code:

python

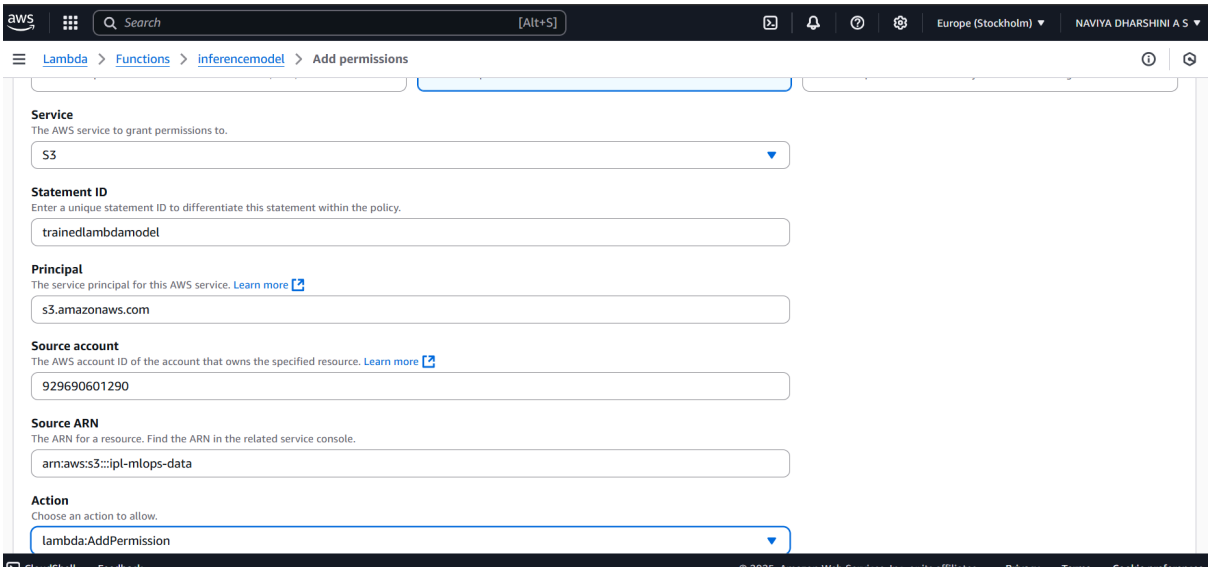
CopyEdit

```
import boto3

import joblib

import json

import os

import pandas as pd

from io import BytesIO



def lambda_handler(event, context):

    try:

        body = json.loads(event['body'])  # Exercise 2: Break it by renaming to `body_typo` for debugging



        ipl_wins = body['ipl_wins']

        ipl_team_score = body['ipl_team_score']



        # Load model from S3

        s3 = boto3.client('s3')

        bucket = 'ipl-mlops-data'

        key = 'ipl_model.joblib'

        model_obj = s3.get_object(Bucket=bucket, Key=key)

        model = joblib.load(BytesIO(model_obj['Body'].read()))
```

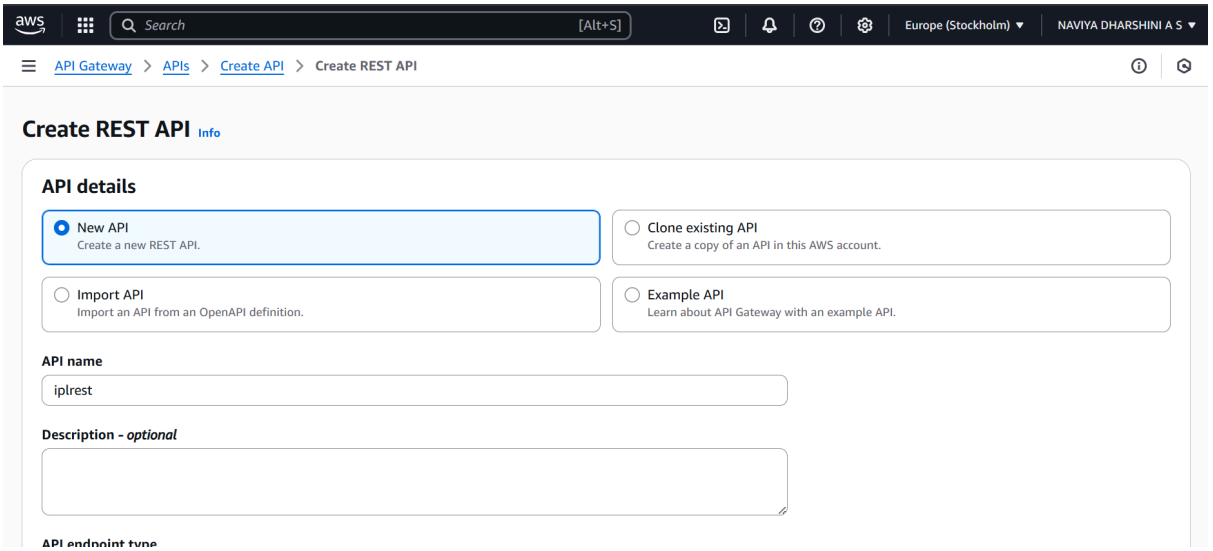
```
        # Make prediction

        input_df = pd.DataFrame([[ipl_wins, ipl_team_score]], columns=['ipl_wins',
'ipl_team_score'])

        prediction = model.predict(input_df)[0]



        return {

            'statusCode': 200,

            'body': json.dumps({'can_win': int(prediction)})

        }



    except Exception as e:

        return {

            'statusCode': 500,

            'body': f"Error: {str(e)}"

        }
```

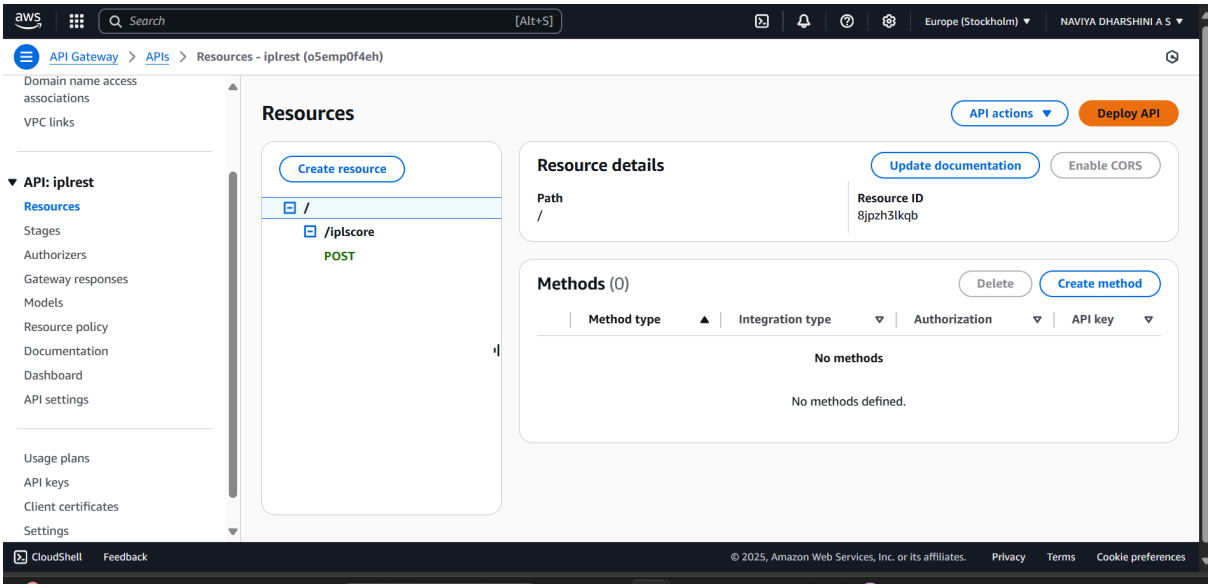
STEP 4 .B) CREATE TRAINED MODEL - INFRENCE MODEL 👍

1. Same env

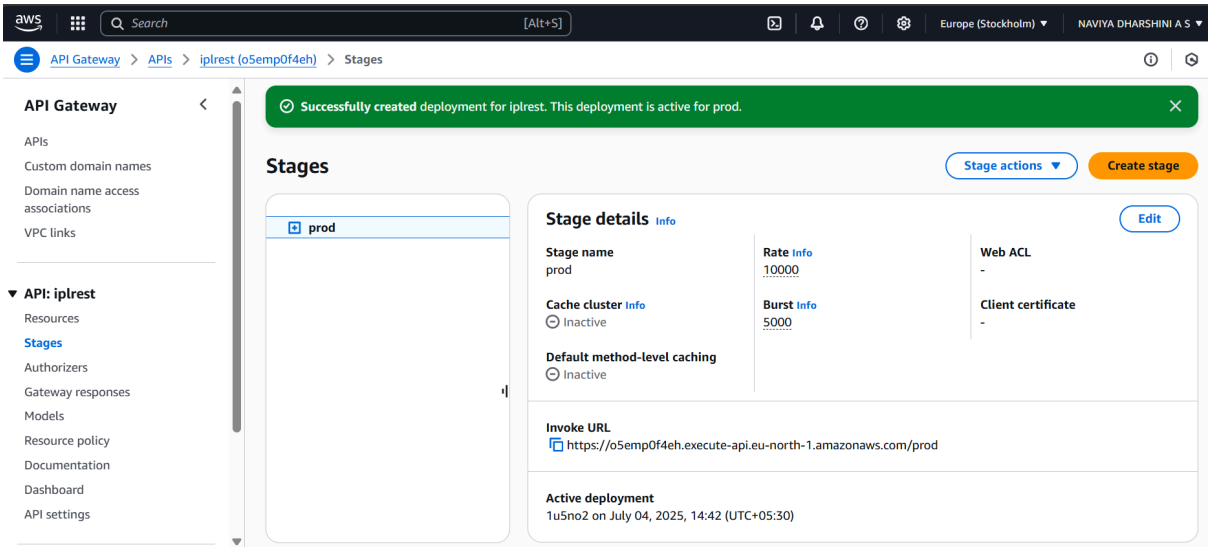


🌐 STEP 5: API Gateway Setup





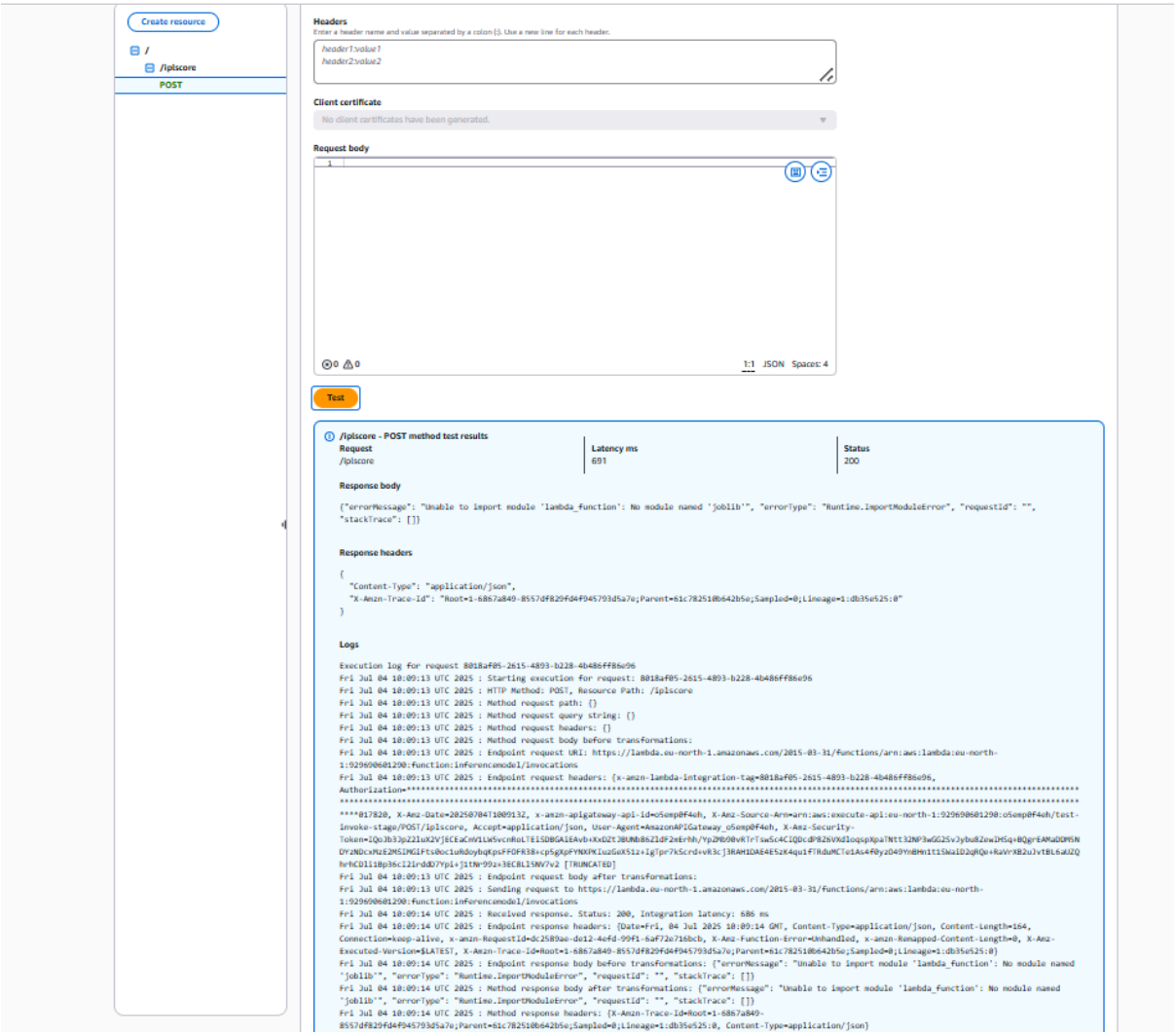
Stages deploy ;;



1. Create a new REST API.
2. POST method → integrate with IPLPredictionInference Lambda.
3. Deploy to a stage (e.g., prod).
4. Test the endpoint via Postman or CLI:

```
json

CopyEdit

POST /prod/predict

{

  "ipl_wins": 6,

  "ipl_team_score": 3000

}
```

🔎 STEP 6: Debugging with CloudWatch (Exercise 2)



Testing stage ..

- Logs auto-generated per Lambda run.
- Go to CloudWatch → Logs → Log Group for Lambda.
- Spot KeyError: 'body_typo' etc.
- Fix code → re-deploy → re-test.

🔐 STEP 7: IAM Access Denied Simulation (Exercise 3)

- Go to IAM → Roles → IPLDataPreProcessor-role.
- Edit inline S3 policy → Remove s3:PutObject access to your S3 bucket.
- Re-run Lambda → watch AccessDenied in CloudWatch logs.
- Revert permissions to fix.

🏈 STEP 8 (Optional): Clone for Football

- Duplicate all above setup using a new S3 bucket (football-mlops-workshop).
- CSV: team,games_played,goals_scored
- Change logic in processor/inference
- New model, SageMaker notebook, Lambda, and endpoint.

🚀 ENDGAME: Final Checklist

✅ ipl_data.csv updated
✅ PreProcessor Lambda triggered
✅ SageMaker model retrained
✅ Inference Lambda deployed
✅ API tested successfully
✅ IAM roles tested
✅ CloudWatch debugging practiced

🗃️ GitHub README (Sample)

md

CopyEdit

AWS MLOps - IPL Win Prediction

Built an end-to-end AWS pipeline using:

- 💾 S3 (raw + processed data)
- 🧠 Lambda (preprocess + inference)
- 📈 SageMaker (train model)
- 🌐 API Gateway (predictions)

How to Use

1. Upload IPL data to S3.
2. Trigger preprocess Lambda.
3. Run SageMaker training.
4. Call API to get predictions!

> Tested with: CSK, MI, LSG, RCB