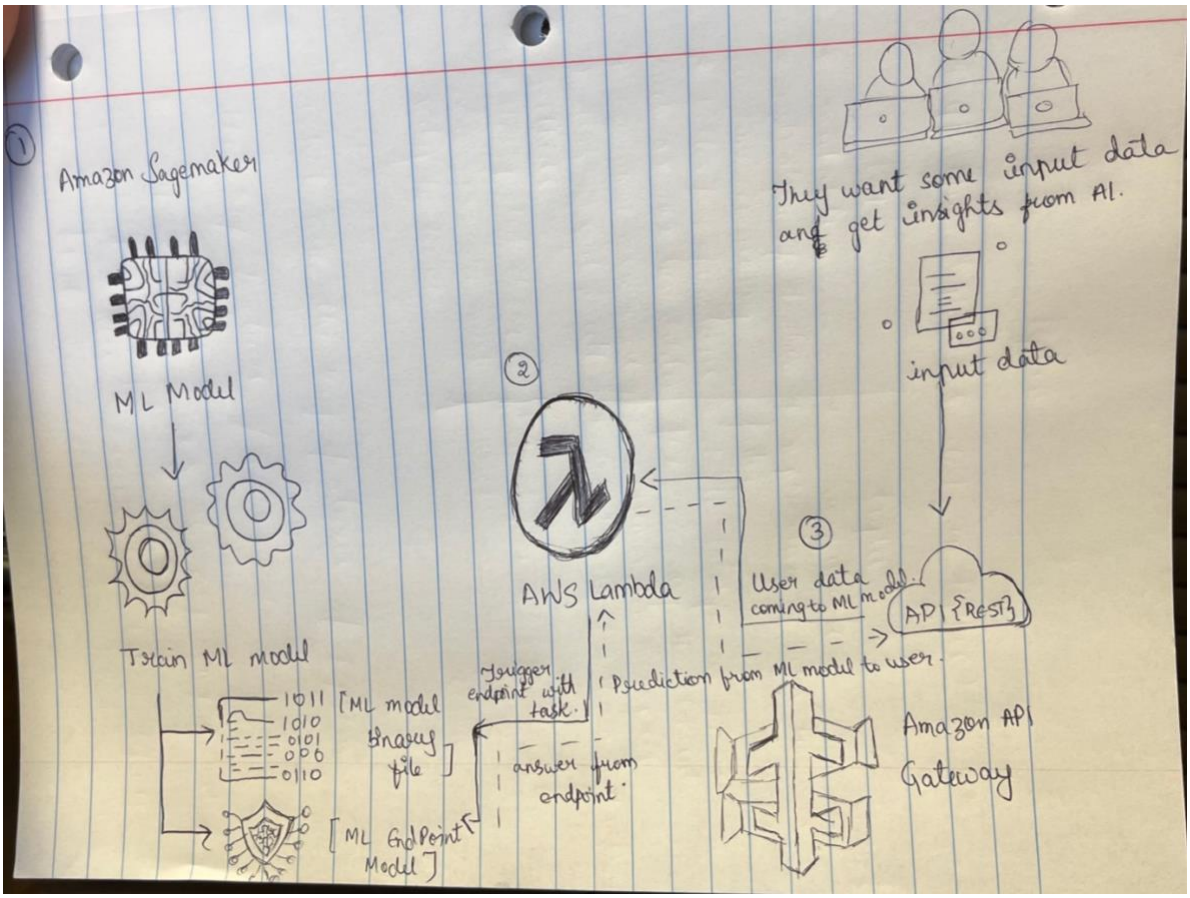
Assignment 4 Report

Architecture:

This is the architecture of the AWS Sage maker and the process that involves with creating, training and building the model. Also creating the Amazon Lambda function along with the API.



In this architecture, it shows us that it will follow certain steps:

- Firstly, we will have the Amazon Sage maker where we will be creating the AWS training model and also run the training model after creating it.
- The training model will be of two types:
    1. Machine Learning Binary File
    2. Machine Learning End point
- Then we will be having the AWS Lambda function created.
- The AWS lambda function will be dealing with the endpoint by triggering the endpoint with a task, and get a proper response from the Endpoint.
- The AWS lambda function will also be used in the API Gateway.
- The REST API which we have selected will be used to send the user data which is coming from the Machine Learning model which I have created.
- That is leading to the prediction from the Machine Learning model to the user as a response.

- We want some insights from the AI, where I have given input data.
- This is the architecture of the AWS Sage maker and the process that involves with creating, training and building the model.

Citation:
https://www.youtube.com/watch?v=stD47vPDadI

Dealing with each step in my Project:

Creating the model and training it:

- To train the model, I have followed the steps of a link which I have got when I browsed it.
- Now, before implementing the code, I created a notebook instance in the Amazon Sage maker.
- The amazon sage maker is used to train, build and deploy the models at scale.
- In this, I have created the notebook instance and gave a name to the notebook instance.
- After I have given name as the ML mode, I have selected the medium as well as T2 medium.
- Then, next comes the IAM Role, where I have created a new role which will be supporting the S3 bucket.
- After all these steps, I have finally created the notebook instance named ml-model.
- I was successful in creating the notebook instance but had to wait a couple of minutes while it was getting activated.
- After activation, I opened the Jupiter where the notebook is empty.
- I have named the notebook and implemented the data code inside it, I have named the folder as data.

- By using the reference, I have done the code in a step wise manner but ended up getting little errors and was involved in rectifying them.

- I have then completed creating the model and trained it.
- I have used the data first to upload the dataset and then have dealt with the prediction and a code for the prediction by using the reference I have cited.

```
from __future__ import print_function

import argparse
import os
import pandas as pd

from sklearn import tree
from sklearn.externals import joblib


if __name__ == '__main__':
    parser = argparse.ArgumentParser()
    # Sagemaker specific arguments. Defaults are set in the environment variables.

    #Saves Checkpoints and graphs
    parser.add_argument('--output-data-dir', type=str, default=os.environ['SM_OUTPUT_DATA_DIR'])

    #Save model artifacts
    parser.add_argument('--model-dir', type=str, default=os.environ['SM_MODEL_DIR'])

    #Train data
    parser.add_argument('--train', type=str, default=os.environ['SM_CHANNEL_TRAIN'])

    args = parser.parse_args()

    file = os.path.join(args.train, "50_Startups.csv")
    dataset = pd.read_csv(file, engine="python")

    # labels are in the first column
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, 4].values

    # Encoding categorical data
    from sklearn.preprocessing import LabelEncoder, OneHotEncoder
```

jupyter Startup prediction.py a minute ago

Logout

File Edit View Language Python

```
    parser.add_argument('--train', type=str, default=os.environ['SM_CHANNEL_TRAIN'])

    args = parser.parse_args()

    file = os.path.join(args.train, "50_Startups.csv")
    dataset = pd.read_csv(file, engine="python")

    # labels are in the first column
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, 4].values

    # Encoding categorical data
    from sklearn.preprocessing import LabelEncoder, OneHotEncoder
    labelencoder = LabelEncoder()
    X[:, 3] = labelencoder.fit_transform(X[:, 3])
    onehotencoder = OneHotEncoder(categorical_features = [3])
    X = onehotencoder.fit_transform(X).toarray()

    # Avoiding the Dummy Variable Trap
    X = X[:, 1:]

    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
    from sklearn.linear_model import LinearRegression
    regressor = LinearRegression()
    regressor.fit(X_train, y_train)

    # Print the coefficients of the trained classifier, and save the coefficients
    joblib.dump(regressor, os.path.join(args.model_dir, "model.joblib"))


def model_fn(model_dir):
    """Deserialized and return fitted model

    Note that this should have the same name as the serialized model in the main method
    """
    regressor = joblib.load(os.path.join(model_dir, "model.joblib"))
    return regressor
```

While importing the sage maker and implementing the code and also deploying it, this is the code:

jupyter Untitled Last Checkpoint: a few seconds ago (autosaved)

Logout

File Edit View Insert Cell Kernel Widgets Help Not Trusted conda_python3

Run Code nbdiff

```
In [1]: import sagemaker
        from sagemaker import get_execution_role

In [2]: sagemaker_session = sagemaker.Session()

In [3]: # Get a SageMaker-compatible role used by this Notebook Instance.
        role = get_execution_role()

In [4]: role
Out[4]: 'arn:aws:iam::520123161164:role/service-role/AmazonSageMaker-ExecutionRole-20221122T200971'

In [5]: train_input = sagemaker_session.upload_data("data")
```

```
In [1]: from sagemaker.sklearn.estimator import SKLearn

        script_path = 'startup_prediction.py'

        sklearn = SKLearn(
            entry_point=script_path,
            instance_type="ml.m4.xlarge",
            framework_version="0.20.0",
            py_version="py3",
            role=role,
            sagemaker_session=sagemaker_session)
```

```
In [2]: sklearn.fit({'train': train_input})
```

```
In [7]: deployment = sklearn.deploy(initial_instance_count=1, instance_type='ml.m4.xlarge')
        !pip install sklearn
```

```
In [4]: deployment.endpoint
```

```
In [5]: deployment.predict([[1,0,50000,25000,40000]])
```

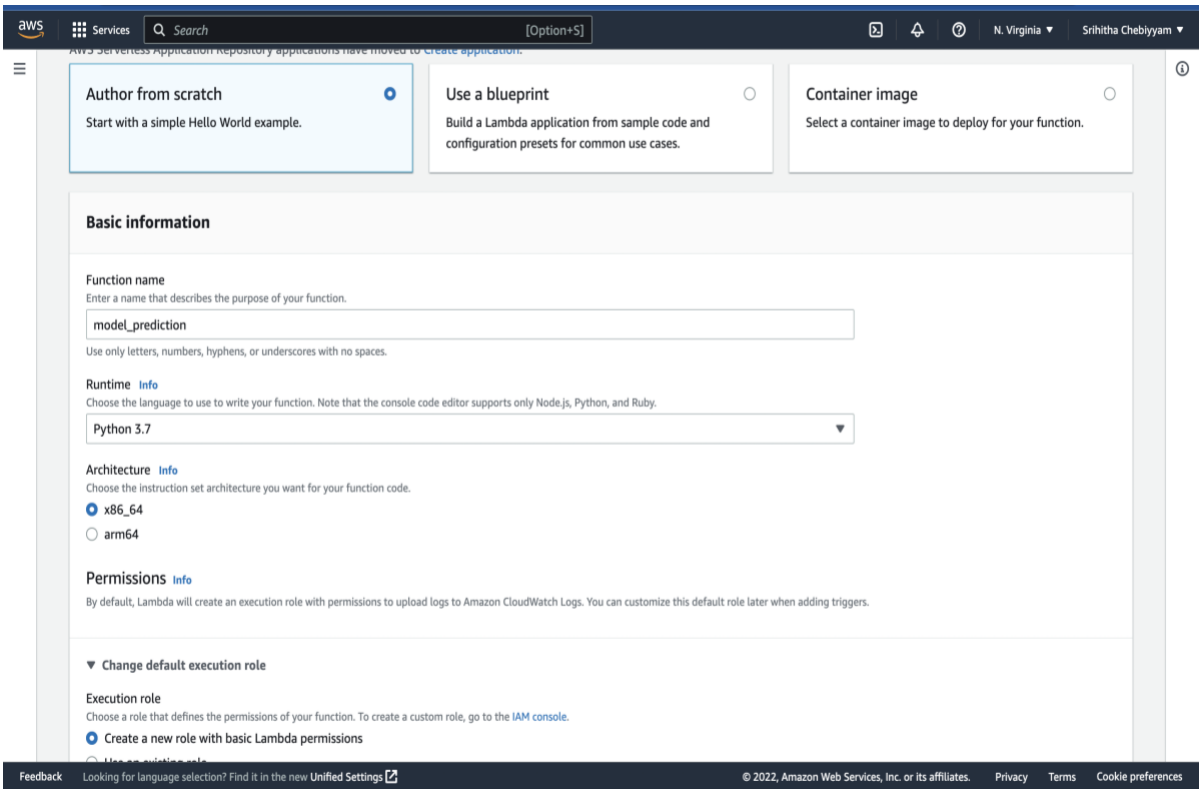
Building an Endpoint:

- To create an endpoint, I went to the inference in the AWS Sage maker.
- Since we have the code for endpoint written already in building and training the model, I can create an endpoint easily.
- After clicking on the inference, you get the endpoint which I have created in the code, then I started creating Lambda.

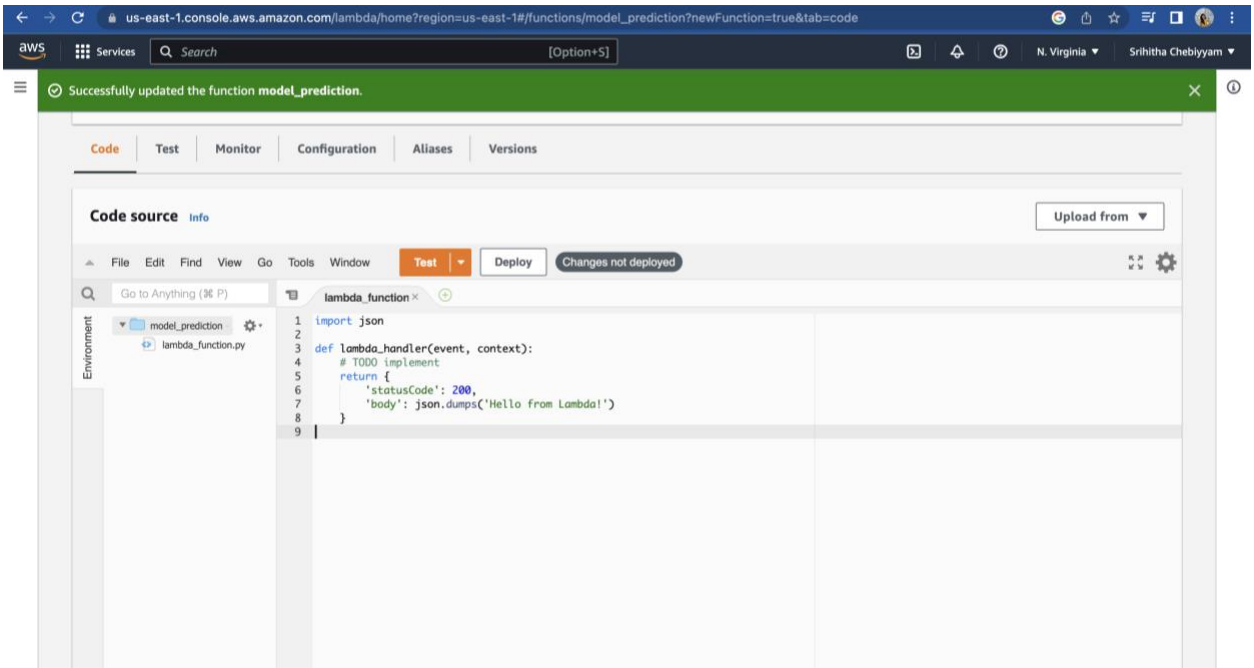
Creating Lambda Function:

- The reason for creating the Lambda function is that it will trigger the endpoint with a task.
- Now, here I have clicked Lambda on the search, and clicked on a new tab.
- Then I have clicked on create function to create a lambda function.

- Similar to creating a model, I have given a name to the function.

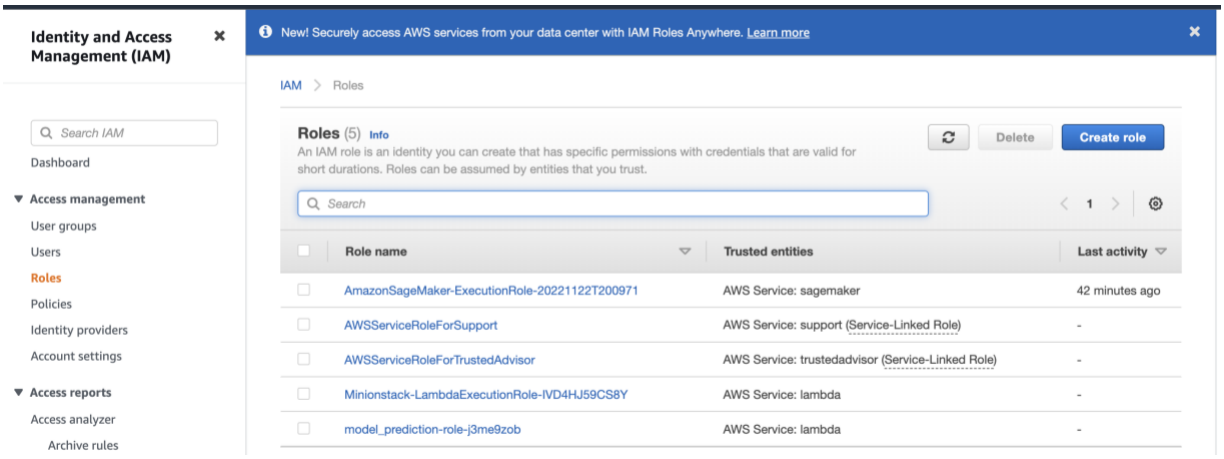


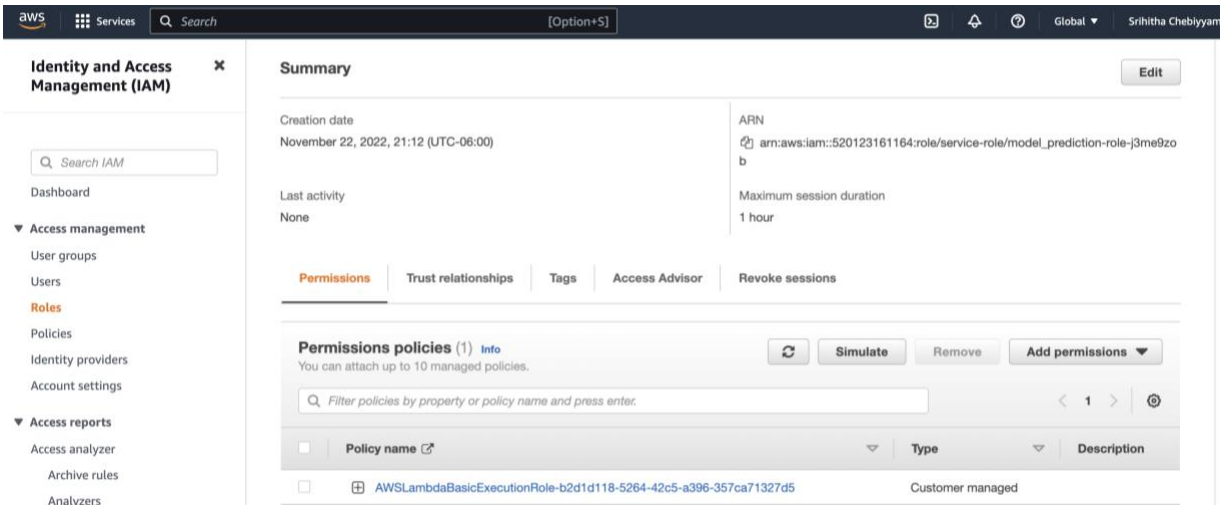
- After creating the lambda function, I have written a code for the lambda function.
- I have inserted the code which I have written into the lambda function.



- Then, I have checked the permissions, went to the IAM and clicked on roles, and selected the one which is suitable and allows the lambda function.

- I have given permission as full access permission to the sage maker in order to run the code.

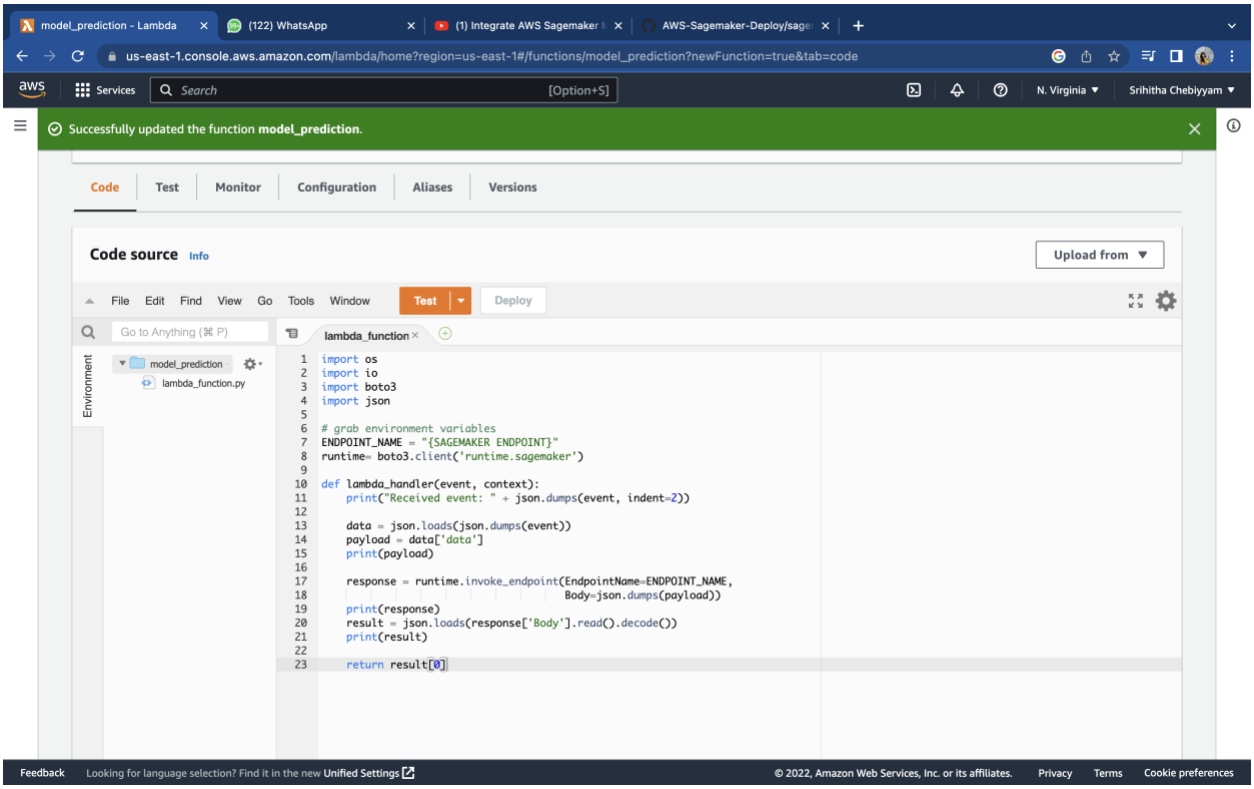




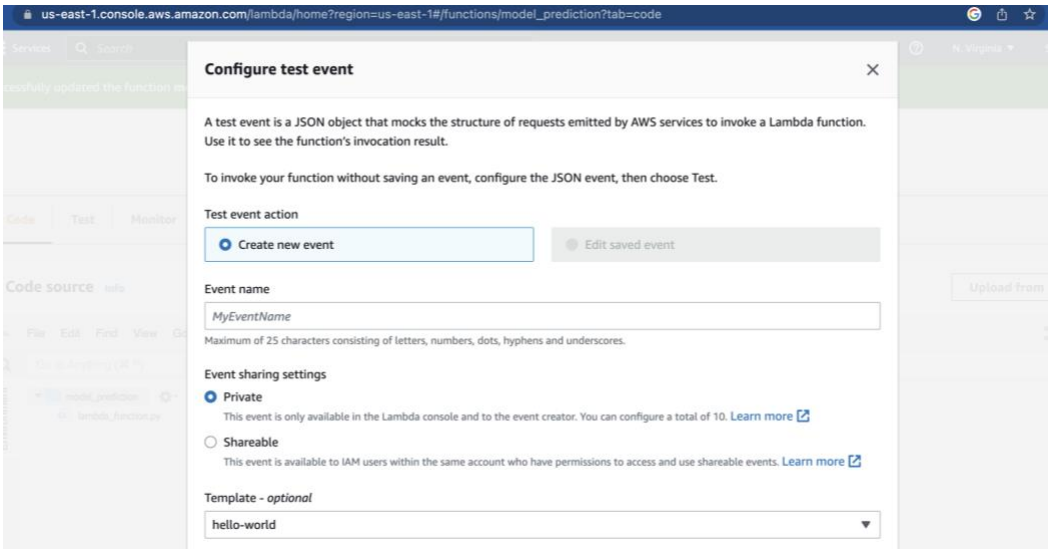
- After inserting the code into the lambda function, I will have to deploy the function, and click on deploy.

- Here, in the code, for the endpoint name, we have to give an environment variable, thus, I have clicked on the environment variable and taken one from that.

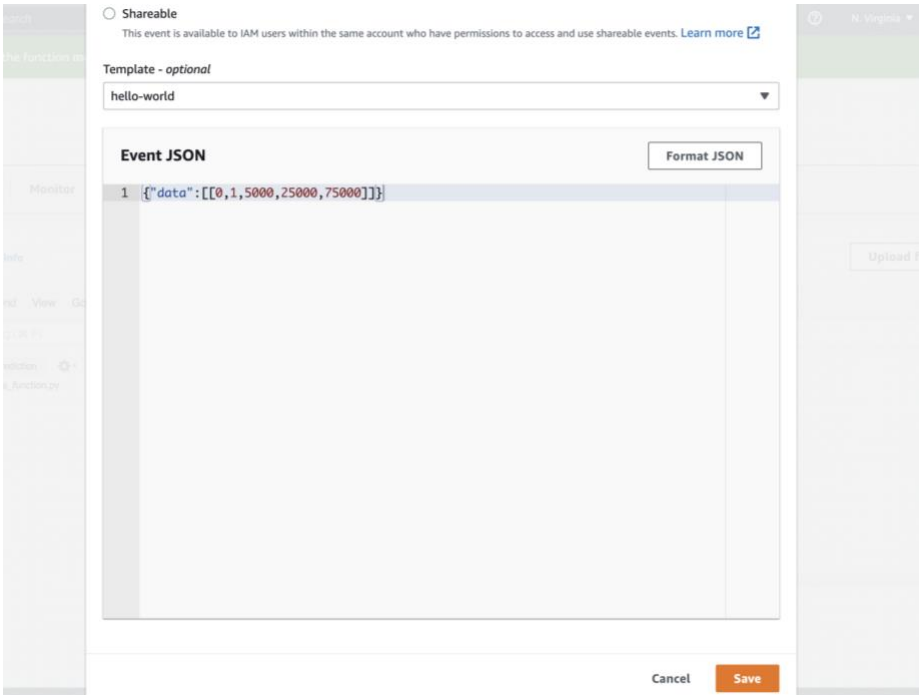
- I have clicked on the suitable permission, there is this option called the policy where I have clicked on it and copied and pasted my code in it.
- In the code, there is a line which allows the lambda function in the sage maker. If there is no code in the line which gives permission to the lambda function, then the lambda function will not be able to enter the AWS sage maker.
- Later by editing the policy and saving it, I have given permission for the sage maker to enter the code and save the policy.



- This is the code for the lambda function which I have edited in the model prediction.
- Then, I have started to test the model.

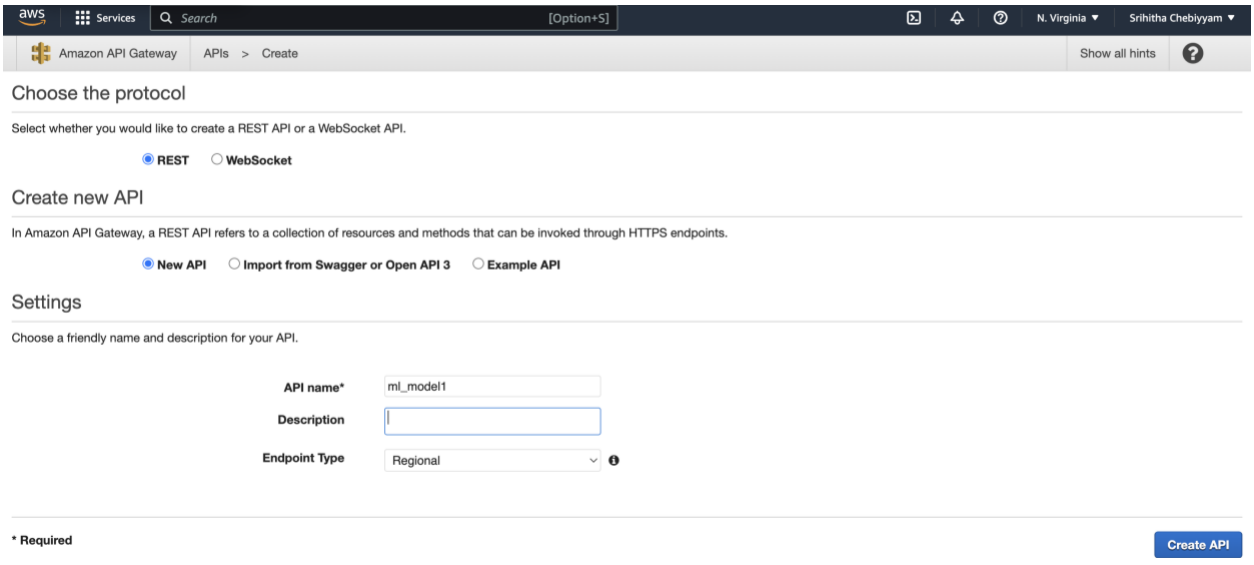


- I have configured the code and saved it.



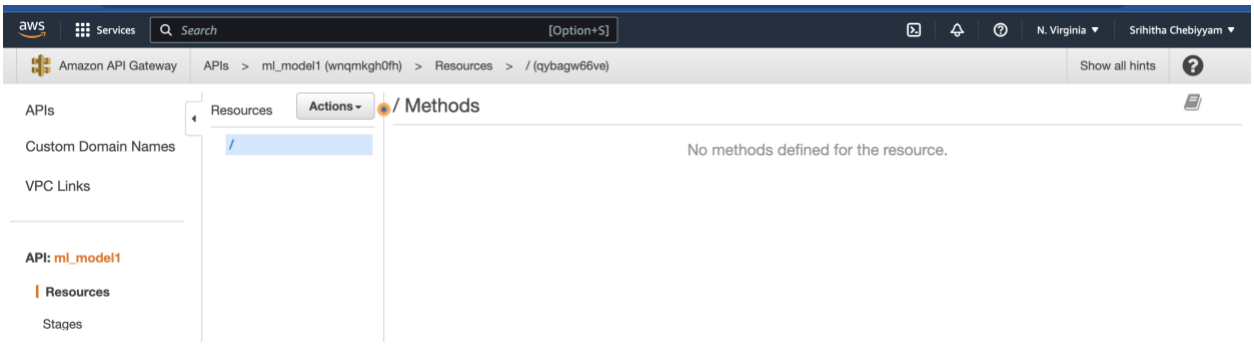
- Then, I got a response after I have tested it and also got a request ID.

Creating an API Gateway:

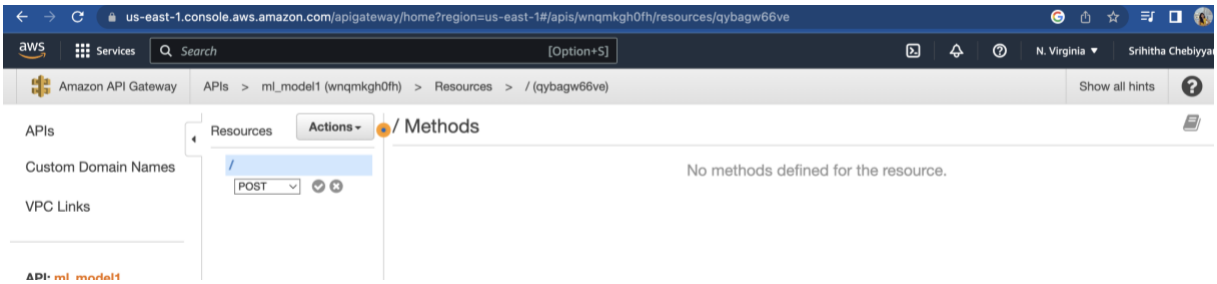
- For creating a API Gateway, I have to search API in the amazon sage maker, and create a new API for the Machine Learning Model.
- Now, by creating the API, the user model from the REST API will be coming to the lambda function.
- 

- The prediction from the lambda function will be delivered to the API gateway as a response.

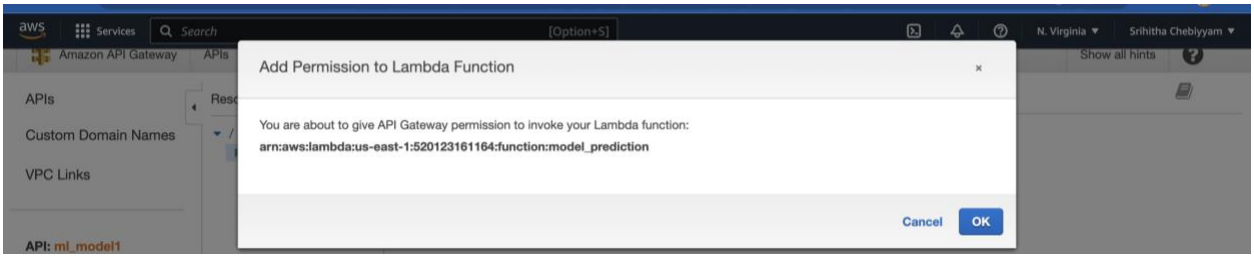
- I have created and built a REST API. Then I have implemented the code for building the REST API.

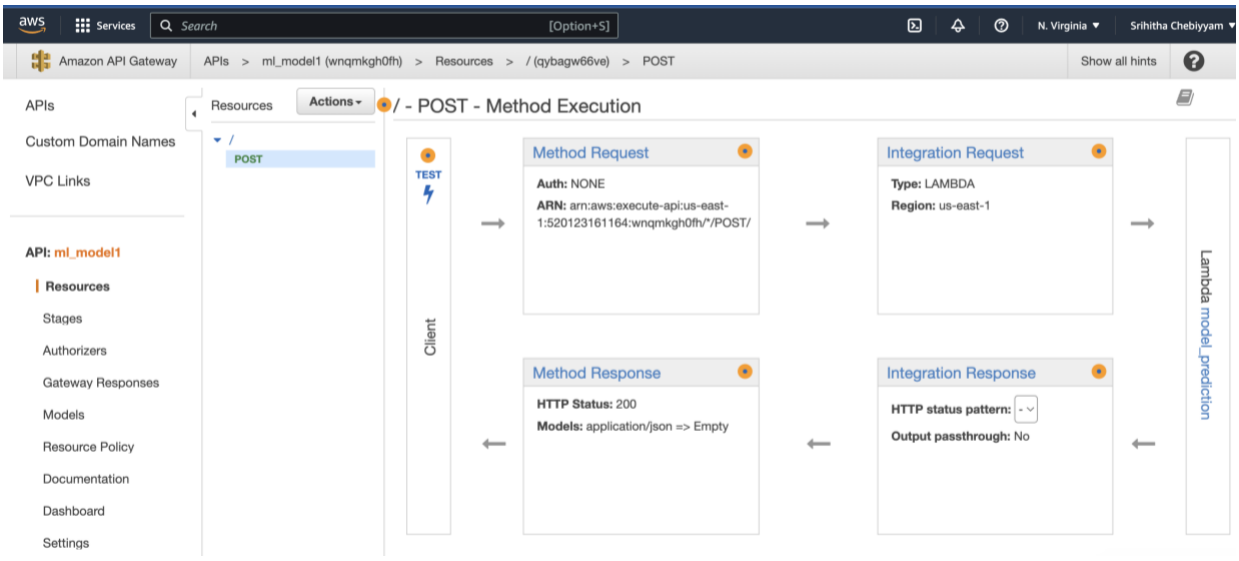


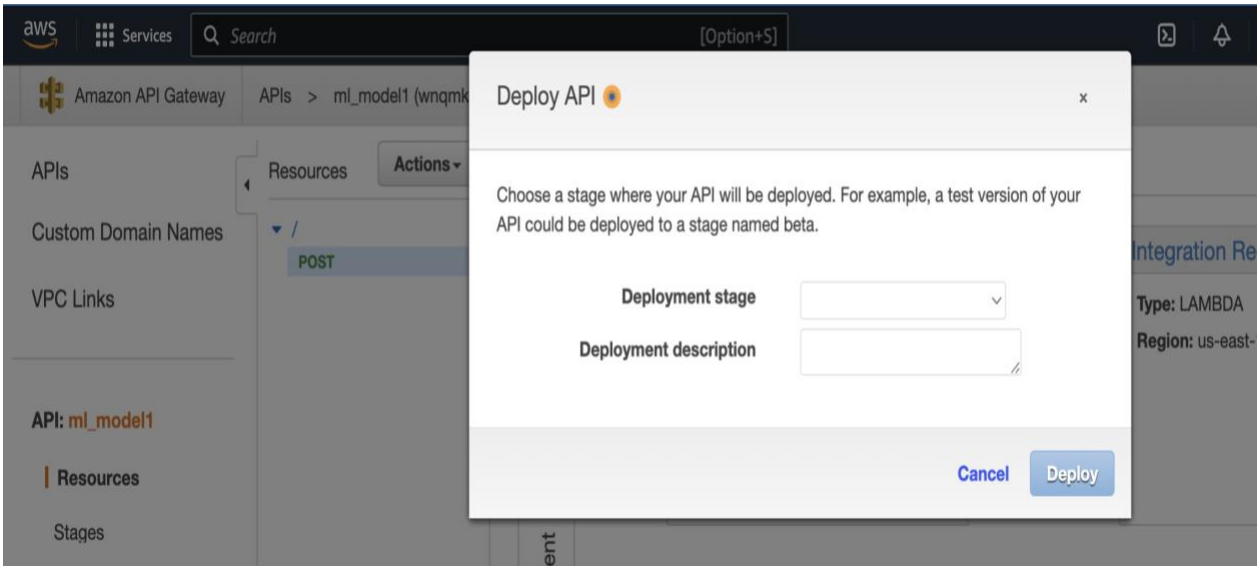
- Similar to the lambda function and endpoint, I have given a name to the API function.
- After giving a name to the API, I have created a new resource.
- Later on, after the creation of a resource, I have created the method inside the resource. The method which I am planning to use is created inside the resource.
- The method type which I have been using is the POST method for a proper functioning.



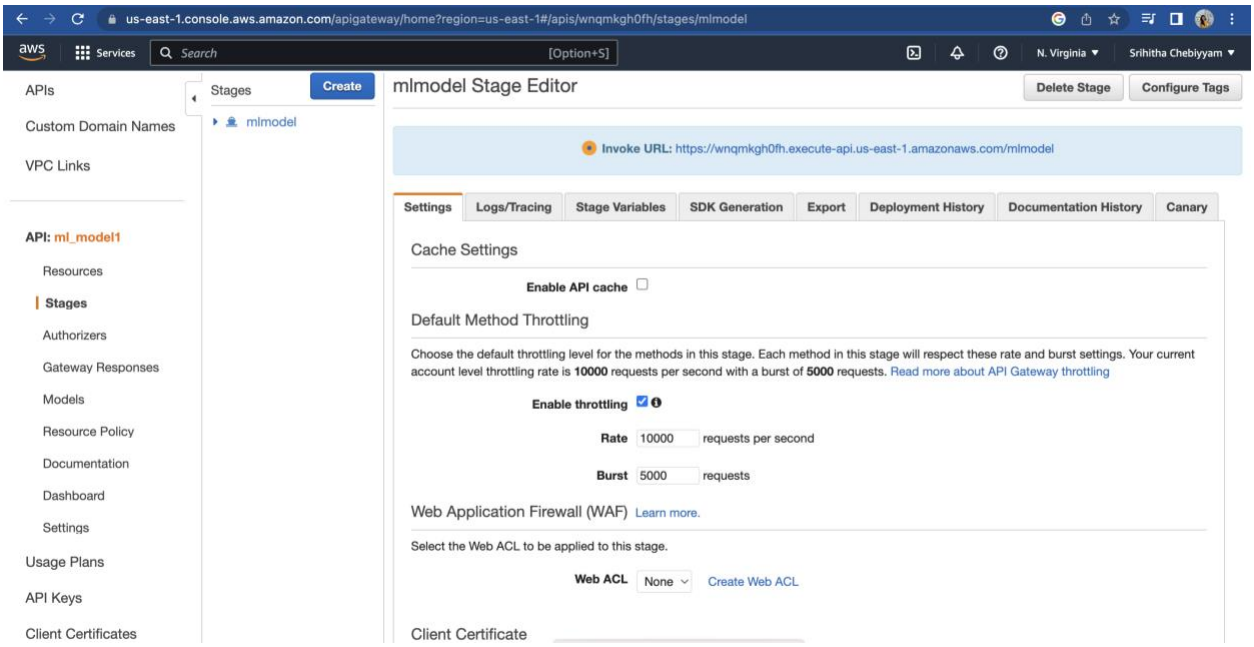
-
- In the integration type, I have selected the lambda function.
- You have to specify the type of lambda function, and save the lambda function. After that, you will have to add permission and test it.
-

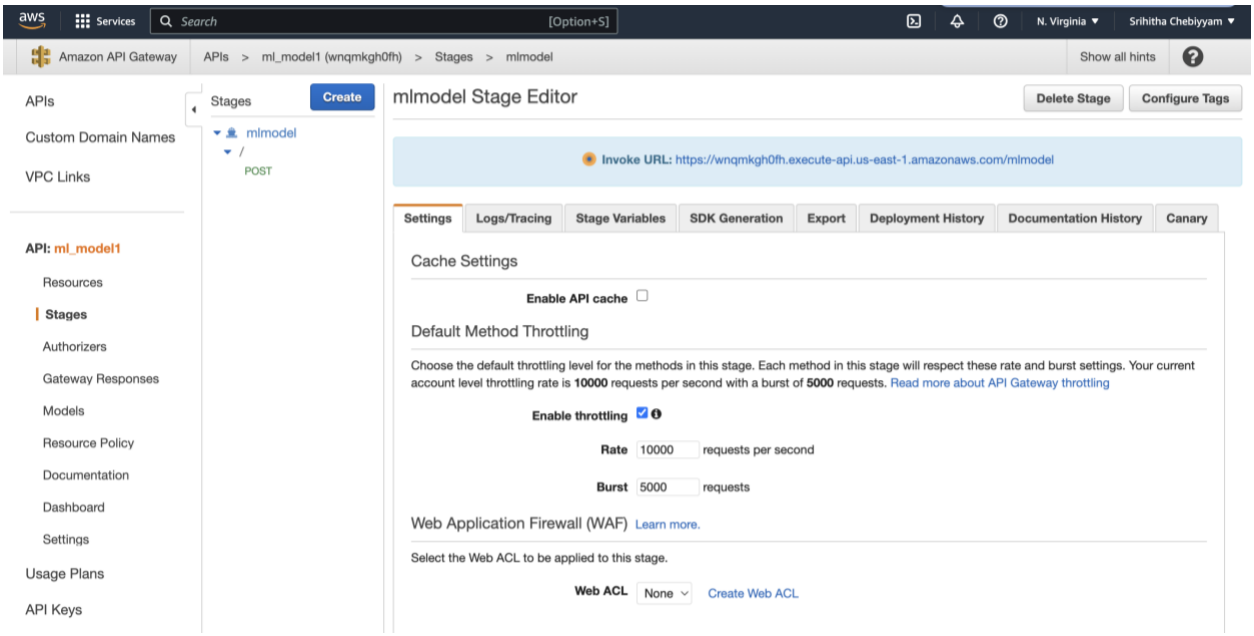


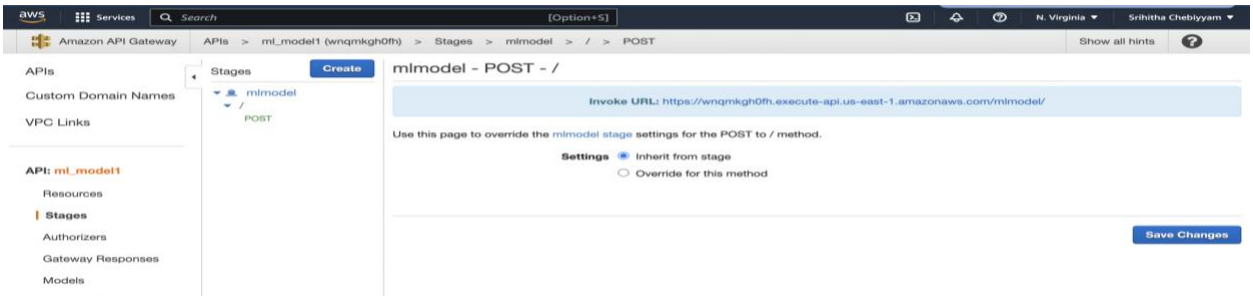




- I have given the deployment stage as new stage and given the name of the deployment stage name as mlmodel.


us-east-1.console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/wnqmkgh0fh/stages/mlmodel
Services
Search
[Option+S]
N. Virginia
Srihitha Chebiyyam
APIs
Custom Domain Names
VPC Links
API: ml_model1
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Dashboard
Settings
Usage Plans
API Keys
Client Certificates
Stages
Create
mlmodel
mlmodel Stage Editor
Delete Stage
Configure Tags
Invoke URL: https://wnqmkgh0fh.execute-api.us-east-1.amazonaws.com/mlmodel
Settings
Logs/Tracing
Stage Variables
SDK Generation
Export
Deployment History
Documentation History
Canary
Cache Settings
Enable API cache
Default Method Throttling
Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is 10000 requests per second with a burst of 5000 requests. Read more about API Gateway throttling
Enable throttling
Rate 10000 requests per second
Burst 5000 requests
Web Application Firewall (WAF) Learn more.
Select the Web ACL to be applied to this stage.
Web ACL None Create Web ACL
Client Certificate

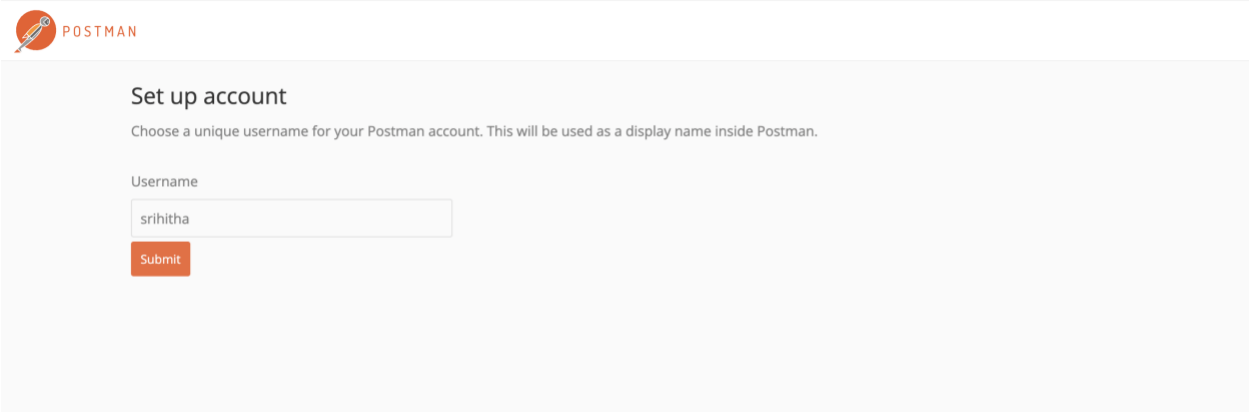

Services
Search
[Option+S]
N. Virginia
Srihitha Chebiyyam
Amazon API Gateway
APIs > ml_model1 (wnqmkgh0fh) > Stages > mlmodel
Show all hints
APIs
Custom Domain Names
VPC Links
API: ml_model1
Resources
Stages
Authorizers
Gateway Responses
Models
Resource Policy
Documentation
Dashboard
Settings
Usage Plans
API Keys
Stages
Create
mlmodel
/
POST
mlmodel Stage Editor
Delete Stage
Configure Tags
Invoke URL: https://wnqmkgh0fh.execute-api.us-east-1.amazonaws.com/mlmodel
Settings
Logs/Tracing
Stage Variables
SDK Generation
Export
Deployment History
Documentation History
Canary
Cache Settings
Enable API cache
Default Method Throttling
Choose the default throttling level for the methods in this stage. Each method in this stage will respect these rate and burst settings. Your current account level throttling rate is 10000 requests per second with a burst of 5000 requests. Read more about API Gateway throttling
Enable throttling
Rate 10000 requests per second
Burst 5000 requests
Web Application Firewall (WAF) Learn more.
Select the Web ACL to be applied to this stage.
Web ACL None Create Web ACL


Services
Search
[Option+S]
N. Virginia
Srihitha Chebiyyam
Amazon API Gateway
APIs > ml_model1 (wnqmkgh0fh) > Stages > mlmodel > / > POST
Show all hints
APIs
Custom Domain Names
VPC Links
API: ml_model1
Resources
Stages
Authorizers
Gateway Responses
Models
Stages
Create
mlmodel
/
POST
mlmodel - POST - /
Invoke URL: https://wnqmkgh0fh.execute-api.us-east-1.amazonaws.com/mlmodel/
Use this page to override the mlmodel stage settings for the POST to / method.
Settings
Inherit from stage
Override for this method
Save Changes

Invoke URL: https://wnqmkgh0fh.execute-api.us-east-1.amazonaws.com/mlmodel/

Testing the Architecture:

- Finally after implementing all the steps, I have finally started to test the API.
- I have tested the API by putting in the code in the block of request body of the postman app.
- I have first created my ID and then started with the testing and executing the architecture.



- In the request body, I have specified the data and also the actual values that are there.
- You will have to enter the same values as the values that we have used in the model training because to get the right architecture and the right output, taking the right values are also very important.
- When I test the architecture, I have entered the values properly, and the predictions were also right.
- After entering the data and the values in the request body, I tested the architecture and got a latency and the status in a good manner.
- This is how I have dealt with the steps of the project.

Citation:

- **https://github.com/aws/amazon-sagemaker-examples/blob/main/sagemaker-python-sdk/pytorch_mnist/pytorch_mnist.ipynb**
- **https://www.youtube.com/watch?v=stD47vPDadI**
- **https://github.com/ScalarPy/AWS-Sagemaker-Deploy**
- **https://www.youtube.com/watch?v=BgougRcbYCE**

Limitations of my Project:

- There were many limitations I faced while I was doing the project.
- I was getting a lot of errors in the implementation of each of the code.
- When I tried to implement the training model and build it, I have got errors in the code and it took me a very long time to resolve it.
- I was even getting problems with the AWS sagemaker. Before I inserted my training code into the AWS Sagemaker, I have done my code in google colab, I got few errors, then when

I inserted the same code after rectification, I am getting errors in the code in the AWS sagemaker.

- While inserting the code into the AWS sagemaker, it costed some amount and a lot of amount was cut at the beginning.
- After realizing it, I immediately stopped running the code, and then started with execution of the code in google colab.
- While writing the lambda code and creating an API, I faced a issue in the code as well and creating it. There were few errors and I tried very hard to rectify it but couldn't do it so better.
- The overall process took a lot of time with a lot of errors and confusions. I have asked my peers for help and tried to complete the task.
- I first wanted to refactor the code, but then cited the references from where I have taken all the code and the values and also the architecture.

Code which I have written:

Prediction code:

```
from
__future__
import
print_function

        import argparse
        import os
        import pandas as pd

        from sklearn import tree
        from sklearn.externals import joblib



        if __name__ == '__main__':
            parser = argparse.ArgumentParser()
            # Sagemaker specific arguments. Defaults are set in the environment variables.

            #Saves Checkpoints and graphs
            parser.add_argument('--output-data-dir',                                    type=str,
        default=os.environ['SM_OUTPUT_DATA_DIR'])
```

```
    #Save model artifacts
    parser.add_argument('--model-dir', type=str, default=os.environ['SM_MODEL_DIR'])

    #Train data
    parser.add_argument('--train', type=str, default=os.environ['SM_CHANNEL_TRAIN'])

    args = parser.parse_args()

    file = os.path.join(args.train, "50_Startups.csv")
    dataset = pd.read_csv(file, engine="python")

    # labels are in the first column
    X = dataset.iloc[:, :-1].values
    y = dataset.iloc[:, 4].values

    # Encoding categorical data
    from sklearn.preprocessing import LabelEncoder, OneHotEncoder
    labelencoder = LabelEncoder()
    X[:, 3] = labelencoder.fit_transform(X[:, 3])
    onehotencoder = OneHotEncoder(categorical_features = [3])
    X = onehotencoder.fit_transform(X).toarray()

    # Avoiding the Dummy Variable Trap
    X = X[:, 1:]

    from sklearn.model_selection import train_test_split
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 0)
    from sklearn.linear_model import LinearRegression
    regressor = LinearRegression()
    regressor.fit(X_train, y_train)

    # Print the coefficients of the trained classifier, and save the coefficients
    joblib.dump(regressor, os.path.join(args.model_dir, "model.joblib"))



def model_fn(model_dir):
    """Deserialized and return fitted model
```

Note that this should have the same name as the serialized model in the main method

"""

regressor = joblib.load(os.path.join(model_dir, "model.joblib"))

return regressor

Training the model:

```
import sagemaker
from sagemaker import get_execution_role
sagemaker_session = sagemaker.Session()
# Get a SageMaker-compatible role used by this Notebook Instance.
role = get_execution_role()
role
Uploading the data for training:
train_input = sagemaker_session.upload_data("data")
train_input
Create sagemaker scikit estimator:
from sagemaker.sklearn.estimator import SKLearn

script_path = 'startup_prediction.py'

sklearn = SKLearn(
    entry_point=script_path,
    instance_type="ml.m4.xlarge",
    framework_version="0.20.0",
    py_version="py3",
    role=role,
    sagemaker_session=sagemaker_session)
Train SKlearn estimator on Startup data:
sklearn.fit({'train': train_input})
Deploy the model:
deployment              =              sklearn.deploy(initial_instance_count=1,
instance_type="ml.m4.xlarge")
deployment.endpoint
deployment.predict([[1,0,50000,25000,40000]])
```

Lambda function:

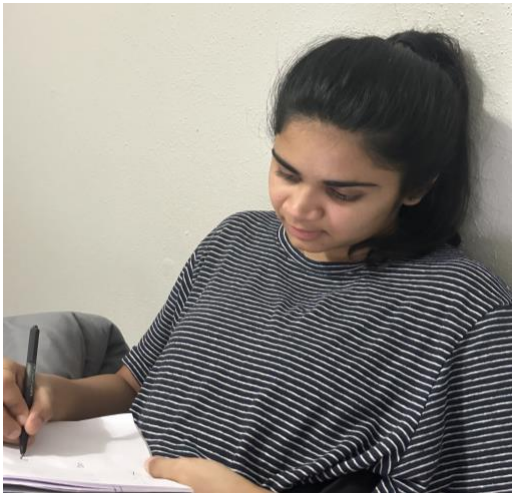
```
import
os
        import io
        import boto3
        import json


        # grab environment variables
```

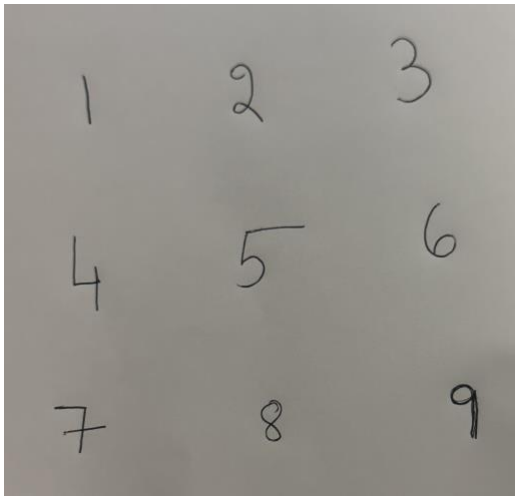
```
ENDPOINT_NAME = "{SAGEMAKER ENDPOINT}"
runtime= boto3.client('runtime.sagemaker')

def lambda_handler(event, context):
    print("Received event: " + json.dumps(event, indent=2))

    data = json.loads(json.dumps(event))
    payload = data['data']
    print(payload)

    response = runtime.invoke_endpoint(EndpointName=ENDPOINT_NAME,
                                       Body=json.dumps(payload))
    print(response)
    result = json.loads(response['Body'].read().decode())
    print(result)

    return result[0]
```

Working API:

I have terminated the API because it is costing a lot of money. I have got the output, but I have terminated it once after the execution because there are charges which are occurring too much and I might not be able to afford so much.

Images of me and my writing:





Citations for the project:

- https://github.com/aws/amazon-sagemaker-examples/blob/main/sagemaker-python-sdk/pytorch_mnist/pytorch_mnist.ipynb
- https://www.youtube.com/watch?v=stD47vPDadI
- https://github.com/ScalarPy/AWS-Sagemaker-Deploy
- https://www.youtube.com/watch?v=BgougRcbYCE