SageMaker Hosting a Model

1. Create **SageMakerInvokeEndPoint** policy:[1]

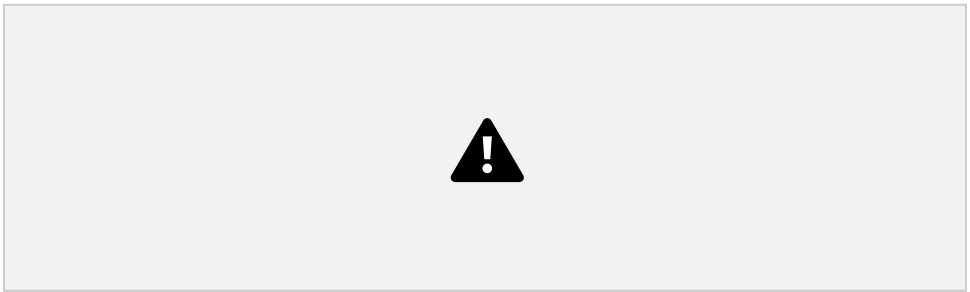
a. AWS Console > IAM > Policies > Create Policy.
b. Service = SageMaker
c. Actions = Read
d. Resources = All Resources
e. Name = “SageMakerInvokeEndPoint”
f. No tags
g. Create Policy

2. Create **ml_user_predict** user:[2]

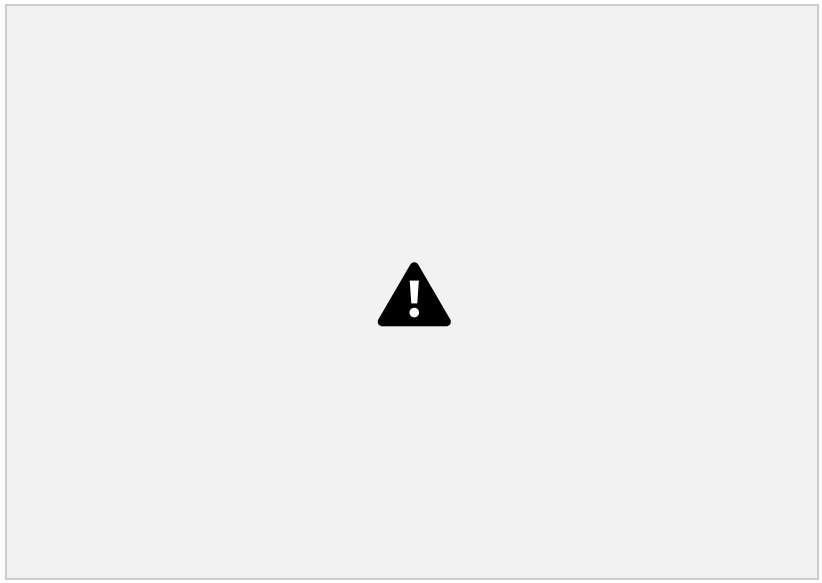
a. AWS Console > IAM > Users > Add User > User name = “ml_user_predict”
b. Access Type = Programmatic Access
c. Attach existing policies directly > SageMakerInvokeEndPoint > Check box
d. Attach existing policies directly > AmazonS3ReadOnlyAccess > Check box
e. No tags.
f. Create user
g. Download .csv > Open .csv on your local machine using any text editor or IDE.

3. Create **Notebook Instance**:[3]

a. AWS Console > SageMaker
b. Notebook > Notebook Instances > Create Notebook Instance
c. Notebook Instance Name = “xgboost-hosted-model”
d. Permissions and encryption > IAM Role > Drop down and select **Create A New Role**



[1] Section 1: 9. Lab - Configure IAM Users, Setup Command Line Interface (CLI)
[2] Section 1: 9. Lab - Configure IAM Users, Setup Command Line Interface (CLI)
[3] Section 2: 16. Lab - S3 Bucket Setup

- 

e. Create role
f. Create Notebook Instance

4. Create **Bucket**:[4]

a. AWS Console > S3 > Buckets > Create Bucket
b. Bucket name = “xgboost-hosted-model-bucket”
c. Create Bucket

5. Download the **Datasets** needed for cleaning:[5]

a. Go to the competition
   - https://www.kaggle.com/c/bike-sharing-demand/data
b. Login > Rules > Accept Competition
c. Data > Download All

6. Upload the **Datasets** needed for cleaning:

a. Unzip the two datasets onto your local machine:
   - You should see both a test.csv and train.csv
b. AWS Console > Amazon SageMaker > Notebook > Notebook Instances
c. Click “Open Jupyter” for the “xgboost-hosted-model” instance
d. Upload the two files test.csv and train.csv
e. 

[4] Section 2: 16. Lab - S3 Bucket Setup
[5] Section 7: 61. Lab - Data Preparation Bike Rental Regression

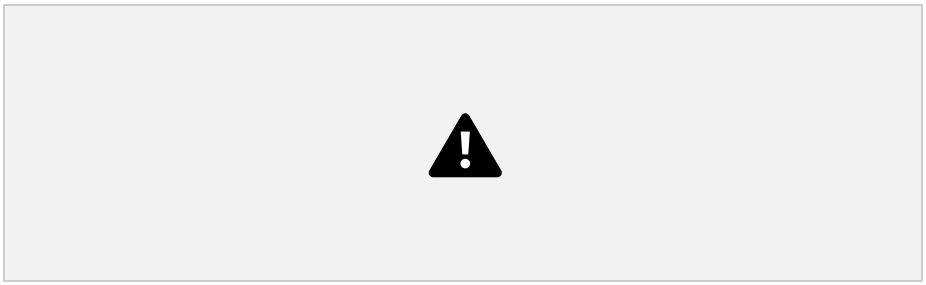
7. Clean **Datasets** needed to train the model:[6]

a. AWS Console > Amazon SageMaker > Notebook > Notebook Instances
b. Click “Open Jupyter” for the “xgboost-hosted-model” instance
c. Upload the **bikerental_data_preparation_rev1.ipynb** notebook
    - Udemy course resource directory: /AmazonSageMakerCourse-master/xgboost/BikeSharingRegression/bikerental_data_preparation_rev1.ipynb
    - Google Drive Link to notebook
d. Open the notebook
e. Cell > Run All
f. Close the notebook tab

8. Train **XGBoost Model** & create **Endpoint Configuration**:[7]

a. AWS Console > Amazon SageMaker > Notebook > Notebook Instances
b. Click “Open Jupyter” for the “xgboost-hosted-model” instance
c. Upload the **xgboost_cloud_training_template.ipynb** notebook
    - From the Udemy course, confirm that this is NOT the iris classification notebook of the same name.
    - Udemy course resource directory: /AmazonSageMakerCourse-master/xgboost/BikeSharingRegression/sdk1.7/xgboost_cloud_training_template.ipynb
    - Google Drive Link to notebook
d. Open the notebook
e. Cell > Run All
f. Close the notebook tab
g. **NOTE**:
    - If you run this notebook more than once, this cell:
    - 
    - Will give the error:
    -
    - This is because you’ve already made the Endpoint Configuration, even though the Endpoint itself isn’t yet running. To see this go to: AWS Console > Amazon SageMaker > Inference > Endpoint Configuration

[6] Section 7: 61. Lab - Data Preparation Bike Rental Regression
[7] Section 7: 65. Lab - How to train using SageMaker’s built-in XGBoost Algorithm



- To actually run the endpoint, you need to create a new endpoint using that configuration. To do this go to: AWS Console > Amazon SageMaker > Inference > Endpoints.
- Create Endpoint > Endpoint name = “xgboost-bikerental-v1” (Or, I use the convention of the same name as the endpoint configuration.)
- Use An Existing Configuration
- Check the box to the “xgboost-bikerental-v1” configuration > Select endpoint configuration”
- Create Endpoint
- Never run that cell again. You can now delete the cell. Continue running the remainder of the notebook or just don’t run this notebook again. It’s served its purpose.

9. Examine your **Endpoint,** once the previous step is complete:

a. AWS Console > Amazon SageMaker > Inference > Endpoints

b. Leave the endpoint running for now but delete it when you’re done with this project (which is the final step of this tutorial).

10. Make predictions on the **XGBoost Model**:[8]

a. **NOTE** - You don’t actually need to run this step to make a hosted model but it’s useful to know how to hit a deployed endpoint using a notebook.
b. AWS Console > Amazon SageMaker > Notebook > Notebook Instances
c. Click “Open Jupyter” for the “xgboost-hosted-model” instance
d. Upload the **xgboost_cloud_prediction_template.ipynb** notebook
    - From the Udemy course, confirm that this is NOT the iris classification notebook of the same name.

[8] Section 7: 67. Lab - How to run predictions against an existing endpoint

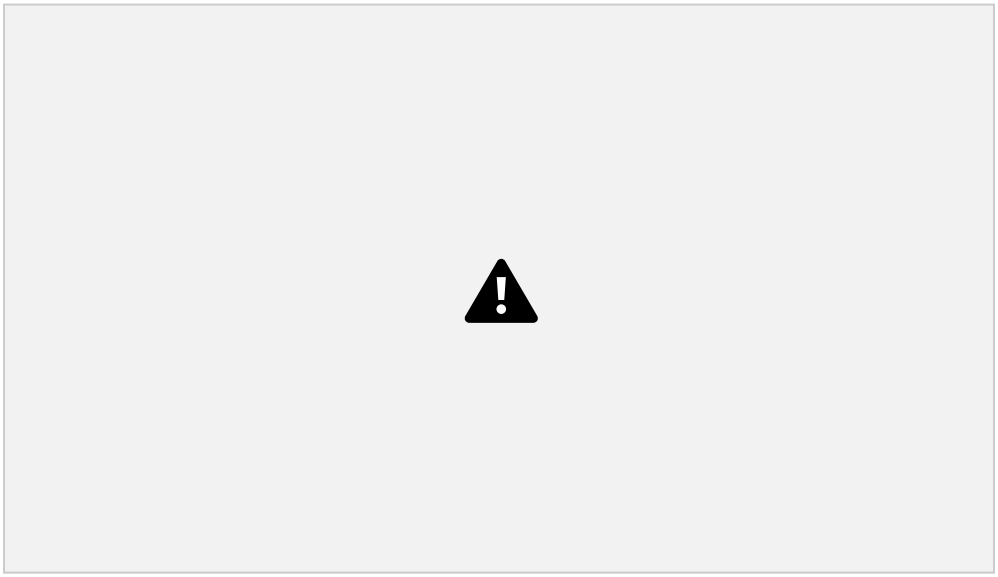
- Udemy course resource directory: /AmazonSageMakerCourse-master/xgboost/BikeSharingRegression/sdk1.7/xgboost_cloud_prediction_template.ipynb
- Google Drive Link to notebook

e. Open the notebook tab
f. Cell > Run All
g. Close the notebook tab

11. Create an IAM role for your **Lambda function**:[9]

a. AWS Console > IAM > Roles > Create Role.
b. AWS Service > Lambda > Next.
c. Assign the permission “SageMakerInvokeEndPoint” > Check the box
d. Assign the permission “AWSLambdaBasicExecutionRole” > Check the box.
e. Role name = “lambda_sagemaker_invoke_endpoint”

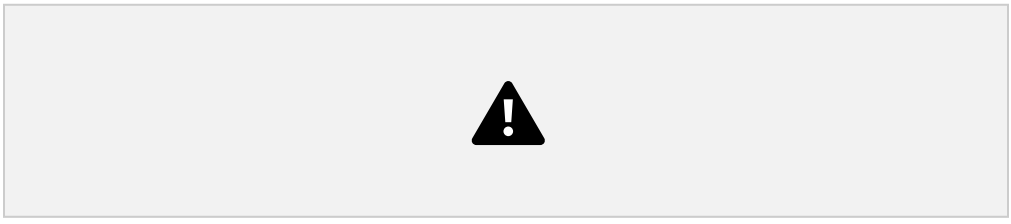
12. Create a **Lambda function**:[10]

a. AWS Console > Lambda > Create Function
b. Author From Scratch
c. Function name = “bikerental_prediction”
d. Runtime = Python3.9 (or latest supported)
e. Change default execution role > Use existing role > lambda_sagemaker_invoke_endpoint
f. Create function
g. Scroll down > Copy and paste the from this file:
   - Google Drive Link to Code
   - Should look like this:
   - 

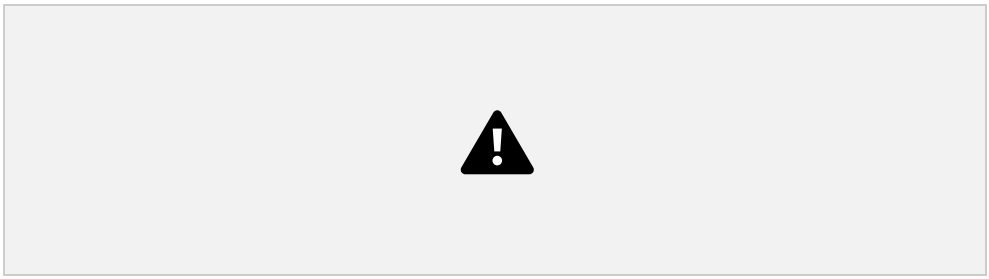
[9] Section 8: 85. Lab - Microservice - Lambda to Endpoint
[10] Section 8: 85. Lab - Microservice - Lambda to Endpoint

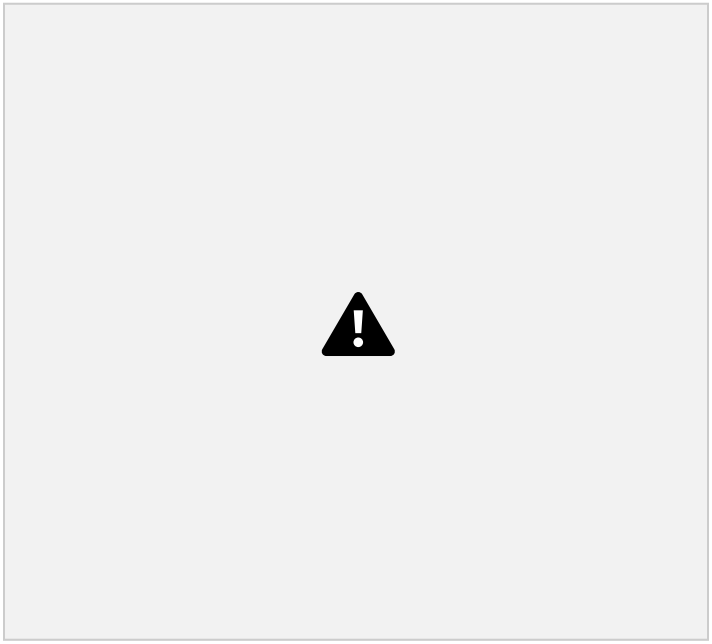
h. Configuration tab > Environment Variables > Edit > Add Environment Variable
i. Key = “ENDPOINT_NAME”
j. Value = “xgboost-bikerental-v1”
k. Save
    - Should look like this:
    - 
l. Click **DEPLOY**
    - 
    -
m.

13. Create an **API Gateway** to the endpoint:[11]

a. AWS Console > API > APIs > “Create API”
b. Locate “Rest API” > Build
c. Click REST (non private)
d. Click New API
e. API name = “BikeRentalPrediction”
f. Create API
g. Drop down “Actions” > Create Method > POST
    - Should look like this:
    - 
h. Integration Type = Lambda Function
i. Lambda Function = “bikerental_prediction” (From above)
    - Add permission

[11] Section 8: 87. Lab - API Gateway, Lambda, Endpoint

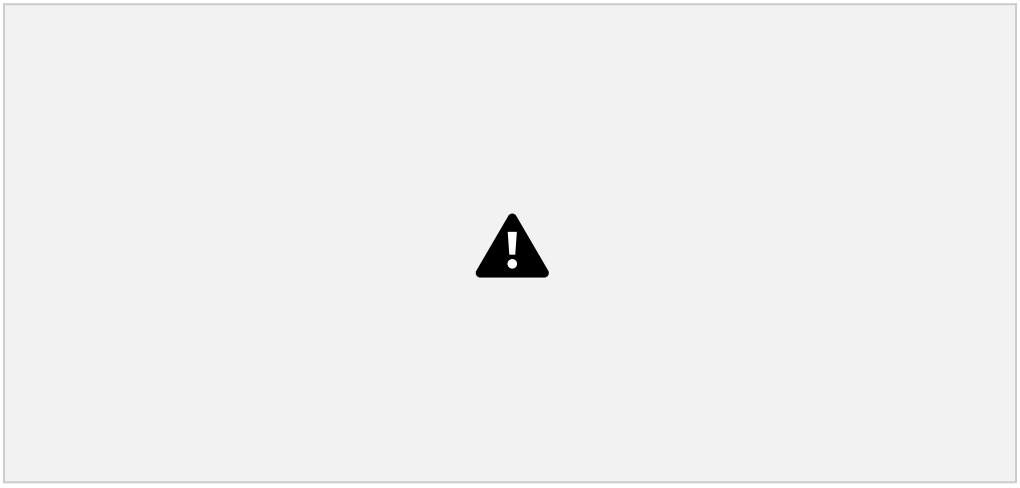
- 

14. Deploy the API Gateway:
    a. Continuing from steps above,
    b. Drop down “Actions” > Deploy API
        - Should look like this:
        - 
    c. Deployment Stage > New Stage
    d. Stage name = “beta”
    e. Deploy
    f. Invoke URL is the url we need for testing
        - 

15. Test on Postman
    a. URL = the url from the step above
    b. POST
    c. Body > Raw
    d. Copy and paste the code below into the body

```
{
     "instances": [
         {
             "features": [
                 "2012-12-19 17:00:00",
                 4,
                 0,
                 1,
                 1,
                 16.4,
                 20.455,
                 50,
                 26.0027
             ]
         }
    ]
}
```

e. Send

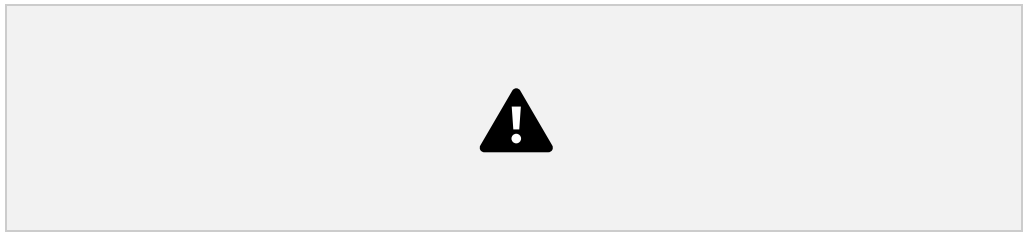
f. 

16. **DELETE THE ENDPOINT** to prevent charges

a. AWS Console > Amazon SageMaker > Inference > Endpoints

- 

b. Check the box on the right > Drop down Actions > Delete > Confirm delete

- 

17. **Recreate** the endpoint for the future

a. AWS Console > Amazon SageMaker > Inference > Endpoints
b. Create endpoint
c. To actually run the endpoint, you need to create a new endpoint using that configuration. To do this go to: AWS Console > Amazon SageMaker > Inference > Endpoints.
d. Create Endpoint > Endpoint name = “xgboost-bikerental-v1” (Or, I use the convention of the same name as the endpoint configuration.)
e. Use An Existing Configuration
f. Check the box to the “xgboost-bikerental-v1” configuration > Select endpoint configuration”
g. Create Endpoint
h. **NOTE!!!** :
    - Don’t forget to **delete** the endpoint when you’re done!

Adding Dependencies (Libraries) To Your Endpoint

18. Adding Dependencies to your Lambda Function[12]:

a. In order to add dependencies to your lambda function (you know, the thing at the top of your python code where you write “import requests”) you’ll need to upload a zip file into your lambda **which could reset all your code** if you don’t copy your existing code when explained below.
b. Open a command prompt and create a `my-sourcecode-function` project directory. For example, on macOS:
    - `mkdir my-sourcecode-function`
c. Navigate to the `my-sourcecode-function` project directory.
    - `cd my-sourcecode-function`
d. Copy the contents of the following sample Python code and save it in a new file named `lambda_function.py` or copy your existing lambda_function.py code:
    - ```
      import requests
      def lambda_handler(event, context):
          response = requests.get("https://www.example.com/")
      ```

[12] https://docs.aws.amazon.com/lambda/latest/dg/python-package.html
      ```

```
    print(response.text)
    return response.text
```

e. Your directory structure should look like this:
   - ```
     my-sourcecode-function$
     | lambda_function.py
     ```
f. Install the requests library to a new package directory.
   - `pip install --target ./package requests`
   - `pip install --target ./package pandas`
g. Create a deployment package with the installed library at the root.
   - ```
     cd package
     zip -r ../my-deployment-package.zip .
     ```
h. This generates a `my-deployment-package.zip` file in your project directory. The command produces the following output:
   - ```
     adding: chardet/ (stored 0%)
     adding: chardet/enums.py (deflated 58%)
     ...
     ```
i. Add the `lambda_function.py` file to the root of the zip file.
   - ```
     cd ..
     zip -g my-deployment-package.zip lambda_function.py
     ```
j. Upload the zip file to the lambda function