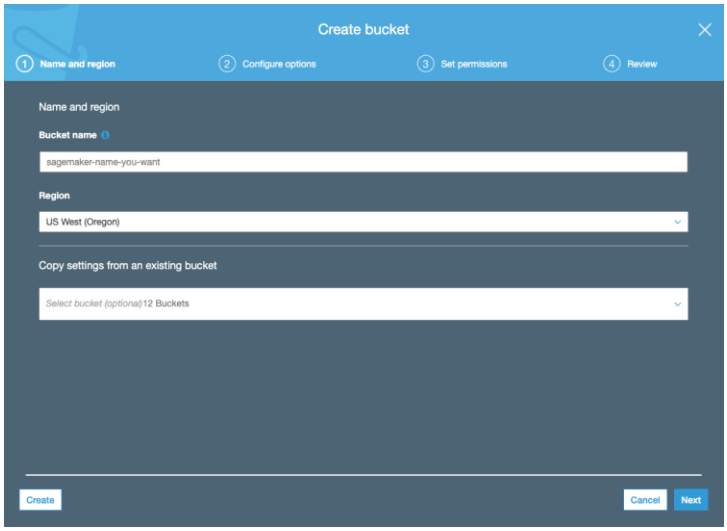
Inference Backend– Step-by-Step Implementation

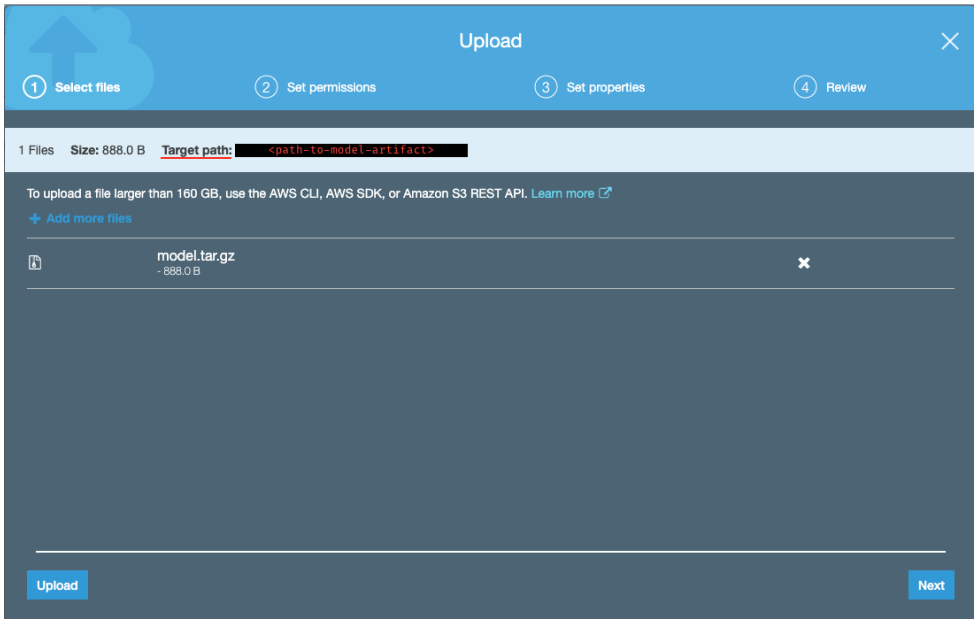
1. We will start by uploading our model artifact to S3.

- Log in to your AWS account and navigate to the S3 homepage.
- On the top left corner, click + Create bucket .
- On the window that opens:
    - Bucket name: you name your bucket whatever you wish, but if you include "sagemaker" in the name it will save you a configuration step later on.
    - Region: Choose the region in which you wish to host your model.
    - Leave the rest as is.
    - On the bottom left corner, click Create.



- Click on the name of the bucket you just created.
- On the top left corner, click + Create folder .
- Name the folder "Model" and hit Save .
- Click on the folder that was just created.
- From the window that appeared, click Add files, and select the model.tar.gz provided in the .zip file.

- Lastly make note of the Target path display at the top of the window.

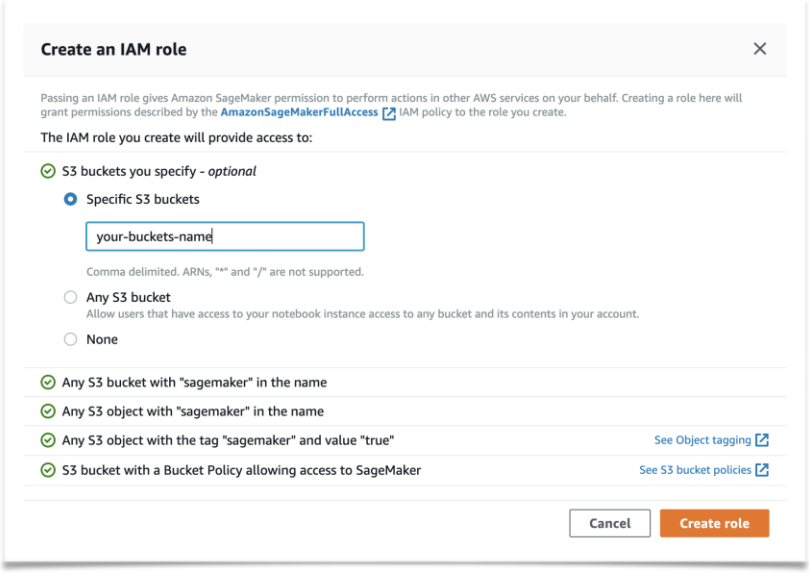


- On the bottom right corner, click **Upload**.

2. Now that our model artifact has been uploaded to S3, we will turn it into an actual model.

- First open your AWS and navigate to the Amazon SageMaker console.
- On the left side of the page, select the Models link under the Inference tab.
- In the Model settings cell, configure:
    - Model name: Choose any name you wish, but make note of it as we will need it later.
    - IAM role: From the drop down menu select Create a new role.
        - This option will create a role with permissions described by the AmazonSageMakerFullAccess IAM policy.

- If the bucket in which you hosted the model artifact contains “sagemaker” in its name, the just click . Otherwise make sure Specific S3 buckets is selected from the radio menu, and enter your bucket’s name in the text field.



- Click **Create role** .

- Now configure the Container Definition cell:

  - For the container input options select the first option on the list, Provide model artifacts and inference image location.

  - Now under the Provide model artifacts and inference image tab:

    - Location of inference code image: 763104351884.dkr.ecr.us-west-2.amazonaws.com/mxnet-inference:1.4.1-gpu-py2

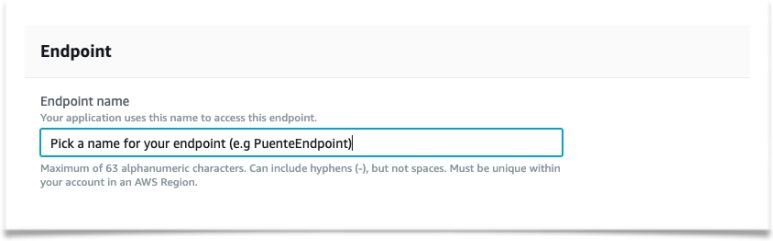
    - Location of model artifacts: Enter the path to the model.tar.gz file you uploaded to your bucket. Make sure that the path ends with /model.tar.gz

- Leave everything else as is, and click **Create model** .

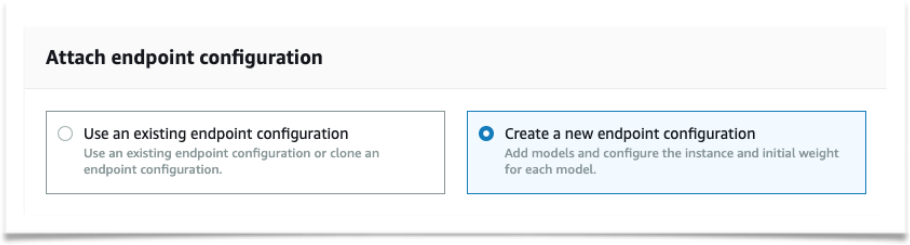
3. Next we will host the model we created using **Amazon SageMaker**.

- On the left hand side of the page, select the Endpoints link under the Inference tab.
- On the top right corner click 

.
- Configure the Cells as follows:
    - For the endpoint name, you may choose any name for the endpoint, but make sure you save it somewhere as we will need this name later to set up our lambda function.

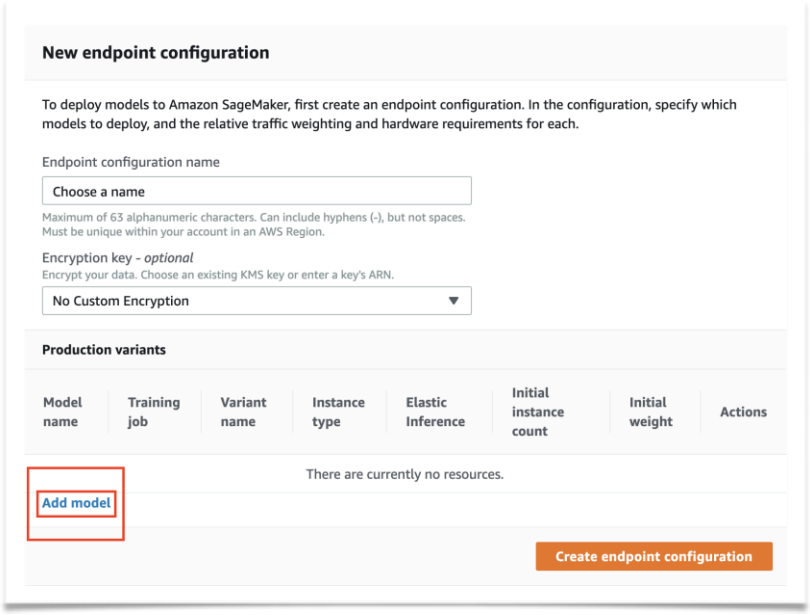


    - In the next cell select Create a new endpoint configuration.



    - For the new endpoint configuration:
        - Endpoint configuration name: Choose any name you wish.

- Encryption key: Leave this on the default option of “No Custom Encryption”.

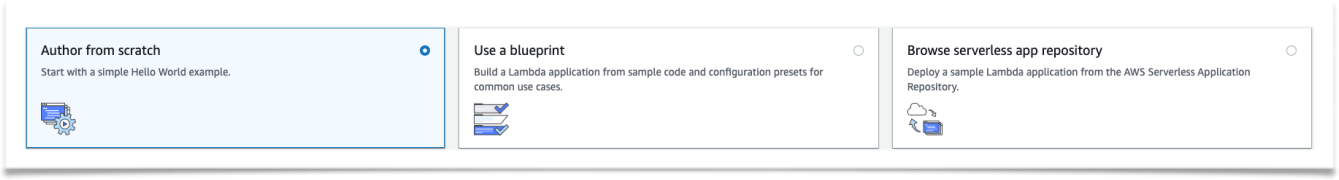


- Now click Add Model.
- From the menu that popped up, select the model we created in the last step and hit **Save**.
- Once you have selected your model, under the actions column you can click edit and choose the instance type you want to host your model on. For the original PoC we used the ml.t2.medium instance.
- Click **Create endpoint configuration**.
- Lastly you can use the Tags section to tag your endpoint. This will make it easier to group together the resources that belong to this PoC so you can tear down the application or keep track of its costs later on.

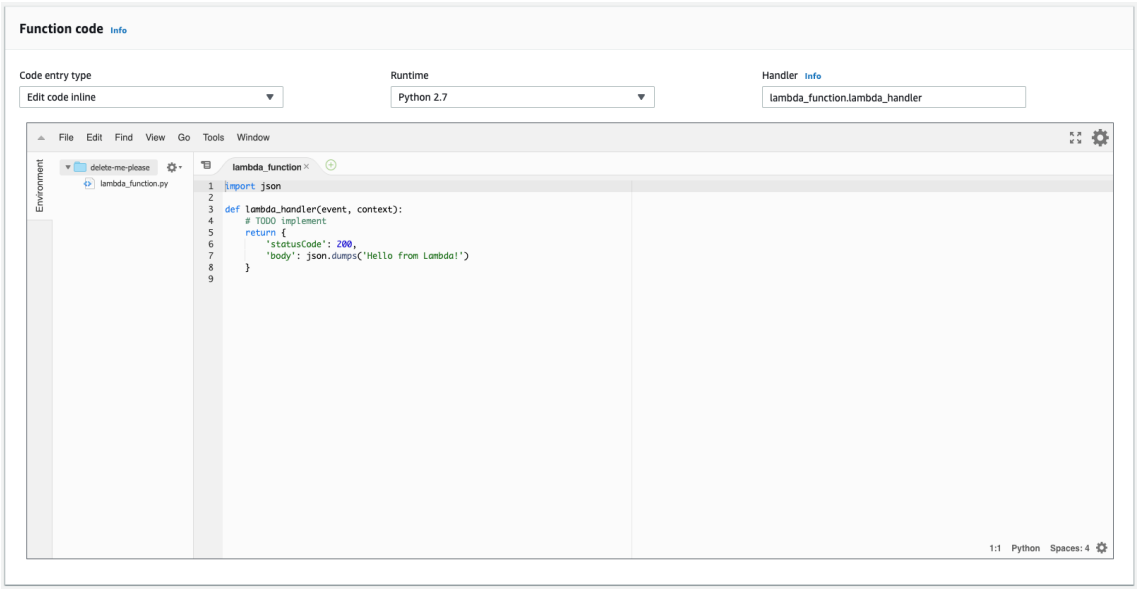
- Click **Create endpoint**.

4. Now we need to create a lambda function to invoke the endpoint we just created.

- Navigate to the AWS Lambda console.
- Click **Create function**.



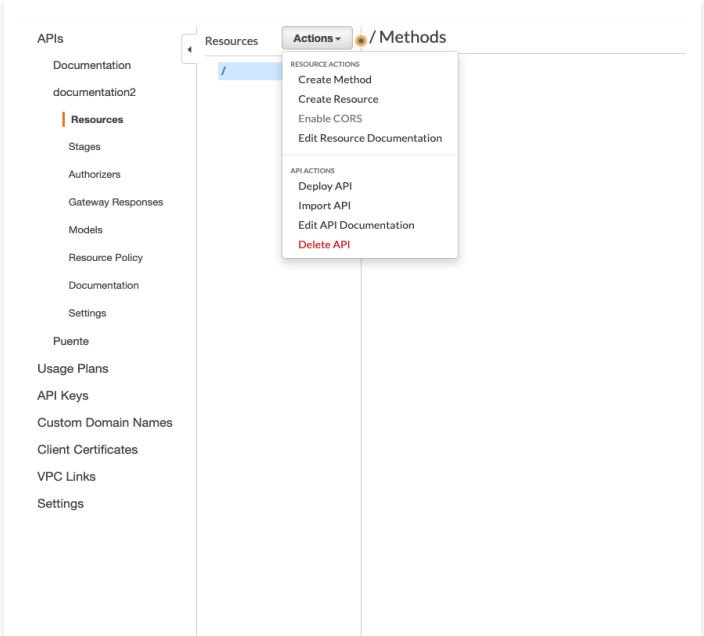
- From the radio menu at the top of the page, select Author from scratch.
- Configure:
    - Function name: Any name you wish.
    - Runtime: Python 2.7
    - Expand the choose or create permissions tab.
        - From the drop down menu, ensure that Create a new role with basic Lambda permissions is selected.
    - Click **Create function**.
    - Scroll down to the Function code cell.
        - Highlight everything in the editor and replace it with the code provided in the .zip .



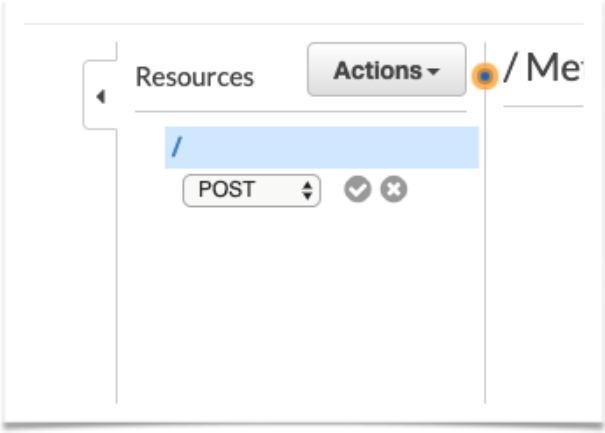
- Scroll to the top of the page, click Save.

5. Lastly we will use API Gateway to create an API to be called from code.

- Navigate to the Amazon API Gateway console
- Choose + Create API.
- On the page that comes up, configure:
    - Protocol: REST
    - Create new API: New API
    - API Name: choose a name
    - Description: Optional
    - Endpoint Type: Regional
- Click Create API.
- From the drop down menu labeled actions, select Create Method.



- A drop down menu will appear, from it select Post then click the checkmark button.



- The section to the right of Post will be populated with some options to configure:
    - Integration type: Lambda Function
    - Use Lambda Proxy Integration: Check
    - Lambda Region: Choose the region which you created the lambda function in.
    - Lambda Function: Start to type out the name of lambda function and a drop down menu should appear. Select the lambda function you created in the last section.
- Click Save.
- A window will appear stating that “You are about to give API Gateway permission to invoke your lambda function”.
- Choose okay.
- From the actions menu, choose deploy API.
- On the window that comes up, configure:
    - For deployment stage: choose “[New Stage]”
    - For Stage Name: choose a name (e.g. Dev)
    - For Stage description: Optional

    - For Deployment description: Optional

- Click Deploy.

- Once the deployment is done, you can navigate to the dashboard link on the left hand side menu under the name of your API. This is the URL you will use to contact the endpoint.