Start Lab 02:00:00

Deep Learning on AWS Lab 3 - Deploying a Trained Model for Prediction on AWS Lambda

2 hours 10 Credits ★★★★½ Rate Lab

In this lab, you will deploy a pre-trained model for production using AWS Lambda. The model is stored in Amazon Simple Storage Service (Amazon S3) and will be downloaded to the Lambda function for predictions. You will upload images to a static website hosted on Amazon S3. The website will call AWS Lambda using Amazon API Gateway. AWS Lambda will pre-process the image and generate predictions for the image based on probabilities. These probabilities are predictions of what the model thinks the image is.

Objectives

After completing this lab, you will be able to:

- Create an AWS Lambda function to load a model for prediction

- Launch a static website for prediction
- Evaluate your predictions

Prerequisites

This lab requires:

- Access to a notebook computer with Wi-Fi running Microsoft Windows, Mac OS X, or Linux (Ubuntu, SuSE, or Red Hat). The Qwiklabs lab environment is not accessible using an iPad or tablet device, but you can use these devices to access the student guide.
- For Microsoft Windows users: administrator access to the computer
- An internet browser such as Chrome, Firefox, or Internet Explorer 9 (previous versions of Internet Explorer are not supported).

Duration

This lab takes approximately **30 minutes**.

Access the AWS Management Console

1. At the top of the lab page, launch the lab by clicking **Start Lab**

   A status bar shows the progress of the lab environment creation process. The AWS Management Console is accessible during lab resource creation, but your AWS resources may not be fully available until the process is complete.

   Note This process can take up to 12 minutes. Do not exit or refresh your browser during this time.

2. When the provisioning process is complete, click **Open console**

3. Log in to the console:

   - For **IAM user name**, type `awsstudent`

- For **Password**, copy and paste the **Password** value from the left side of the lab page

- Click **Sign In**

4. At the top-right corner of the console, make sure the AWS Region is the same as the **Region** displayed on the left side of the lab page.

   ⚠ Only use the Region indicated on the lab page. Do not change to a different Region during this lab.

Task 1: Set up AWS Lambda with Amazon API Gateway

In this task, you will set up AWS Lambda with Amazon API Gateway to push images to the model.

5. In the AWS Management Console, on the **Services** menu, click **Lambda**.

6. Click **Create a function**

7. Select **Use a blueprint**.

8. In the search box, enter `microservice`

9. Click **microservice-http-endpoint-python** for python3.7 api-gateway.

   Note The version displays below each result.

10. Click **Configure**.

11. In the **Basic information** section, configure the following:

12. **Function name:** `ModelDeployment`

13. **Execution role:** *Use an existing role*

13. **Execution role:** *Use an existing role*

14. **Existing role:** *ModelDeploymentRole*

15. In the **API Gateway trigger** section, configure the following:

16. **API:** *Create an API*

17. **Security:** *Open*

18. Click ▸ **Additional settings**, and configure the following:

    - **API name:** `ModelDeployment`

    - **API type:** *REST API*

    - **Deployment stage:** `prod`

    The **Lambda function code** section is pre-populated from the blueprint. The code demonstrates a simple HTTP endpoint using API Gateway.

14. At the bottom of the page, click **Create function**

    Now, you will replace the sample function with one written specifically for your lab environment.

15. Scroll down to the **Function code** section, and configure the following settings:

16. **Code entry type:** *Upload a file from Amazon S3*

17. **Amazon S3 link URL:** Copy and paste the **ZipFile** value from the left side of the lab page

18. Scroll down to the **Basic settings** section.

19. Click Edit and configure the following settings:

20. **Description:** `A simple backend for model prediction with a RESTful API endpoint using`

21. **Memory (MB):** Set to the maximum value (this increases both memory and CPU allocation)

22. **Timeout:** 1 min

23. Click **Save**

24. At the top-right of the page, click **Save**

    Now, you will configure the trigger that activates the function.

20. Scroll up to the **Designer** section at the top of the page, and click **API Gateway**.

21. Scroll down to the **API Gateway** section.

22. Copy the **API endpoint** value, and save it in a text file for later reference.

    Note It will look similar to *https://ypmoszusdk.execute-api.us-west-2.amazonaws.com/prod/ModelDeployment*. This will be referred to as the **PredictURL** in your website code later in the lab.

23. To test your Lambda function, click **Test** at the top of the page.

24. For **Event name**, enter `Pets`

25. Paste the following test event into the editor, *replacing the existing content*:

```
{
  "url": "https://images-na.ssl-images-amazon.com/images/G/01/img15/pet-
products/small-
tiles/23695_pets_vertical_store_dogs_small_tile_8._CB312176604_.jpg"
}
```

    The test asks the model to predict the contents of this picture:

26. At the bottom of the window, click **Create**

27. Click **Test**.

28. Scroll up to the **Execution result** section (above the function chart) and expand ▸ **Details**

    The result returned by your function execution should be similar to the following:

```
{
  "body": "probability=0.714828, class=n02088364 beagle ,
probability=0.144921, class=n02089867 Walker hound, Walker foxhound ,
probability=0.086121, class=n02089973 English foxhound ,
probability=0.021632, class=n04409515 tennis ball ,
probability=0.013204, class=n02088632 bluetick , \n",
  "headers": {
    "Access-Control-Allow-Origin": "*",
    "content-type": "application/json"
  },
  "statusCode": 200
}
```

    This result contains a list of predictions of the picture contents. Notice that the highest probability prediction is *beagle*.

Task 2: Upload your static website to Amazon S3

In this task, you will configure your static website.

29. Right-click the following link, and download the file to your computer: config.js

30. Open the **config.js** file in a text editor.

30. Open the **config.js** file in a text editor.

31. In the file, replace the following fields with the values shown on the left side of the lab page:

    - **region**
    - **upload_bucket_name**
    - **identity_pool_id**

32. In the file, replace the **predict_url** field with the URL you saved earlier in the lab from the API Gateway configuration in Lambda.

33. Save the changes to the **config.js** file.

34. In the AWS Management Console, on the **Services** menu, click **S3**.

35. Click the bucket the includes the word **hosting**. This bucket will host the static website.

36. Navigate to the **js** folder.

37. Click **Upload**.

38. Add the **config.js** file to the dialog box.

39. Click **Upload**.

40. Click the **config.js** file.

41. Click **Make public**.

    This configuration file will be used when predicting images.

Task 3: Predict images using AWS Lambda

In this task, you will use the model to predict image content.

42. In a new browser tab, copy and paste **WebsiteURL** from the left side of the lab page. The static website displays.

43. On the static website, click **Browse**.

44. Upload any JPEG image you have on your computer.

    💬 If you do not have any JPEG images, download one from a website.

45. For predictions, click **Analyze**.

    You should see the top five predictions for the uploaded image. The model used is based on this paper: Deep Residual Learning for Image Recognition.

Lab complete

Congratulations! You have completed this lab. To clean up your lab environment, do the following:

46. Log out of the AWS Management Console by clicking **awsstudent** at the top of the console, and then clicking **Sign Out**.

47. End the lab session in Qwiklabs by clicking **End Lab**.

Chat