Building a Serverless Application Using Step Functions, API Gateway, Lambda, and S3 in AWS

Building a Serverless Application Using Step Functions, API Gateway, Lambda, and S3 in AWS

Introduction

In this hands-on lab, you will create a fully working serverless reminder application using S3, Lambda, API Gateway, Step Functions, Simple Email Service, and Simple Notification Service. By the end of the lab, you will feel more comfortable architecting and implementing serverless solutions within AWS.

Solution

Log in to the AWS Management Console using the credentials provided on the lab instructions page. Be sure to use an incognito or private browser window to ensure you're using the lab account rather than your own.

Note: Make sure you're in the N. Virginia (`us-east-1`) region throughout the lab.

Open the lab's GitHub repository in a second tab. All of the code needed to complete this lab is available there.

Objective 1: Verify an Email Address in Simple Email Service (SES)

1. Open Simple Email Service in a new tab by typing *Simple Email Service* into the search bar at the top of the screen. Right-click **Amazon Simple Email Service** from the drop-down menu, and select **Open Link in New Tab**.
2. In the Amazon Simple Email Service (SES) console, Click the three horizontal line icon in the top left and choose **Configuration - Identities**.
3. Click **Create identity**.
4. Choose **Email address**, enter your personal email address (or an email address you created specifically for this hands-on lab). Click **Create identity**. (Note outlook accounts do not seem to work well with this lab but gmail does).
5. In a new browser tab or email client, navigate to your email, open the Amazon SES verification email, and click the provided link.

   > **Note:** Check your spam mail if you do not see it in your inbox.

   You should see a **Congratulations!** page that confirms you successfully verified your email address.
6. Return to the AWS SES console tab.

   You should now see **Verified** under **Identity status**. (**Note:** You may need to refresh the page.)

Objective 2: Create the Email Lambda Function

1. In the GitHub repo GitHub Repo. Download the files as a .zip and expand this on your machine. Using a text editor such as VSCode open the file, `email_reminder.py` .
2. Return to your first tab, and in the AWS Management Console, navigate to Lambda by typing *Lambda* in the search bar at the top of the screen. Select **Lambda** from the drop-down menu.
3. In the AWS Lambda console, click **Create function**.

4. Leave the **Author from scratch** option selected, and then set the following values:
    - **Function name:** *email*
    - **Runtime: Python 3.12**
5. Expand **Change default execution role** by clicking the arrow next to it.
6. Below **Execution role**, select **Use an existing role**.
7. Once you select that, click into the empty drop-down menu that appears below **Existing role**.
8. Select **LambdaRuntimeRole** from the drop-down menu.
9. Click **Create function** at the bottom of the page.
10. Scroll down to **Code source** and from the **Environment** file list on the left, click `lambda_function.py` to display the function code.
11. Delete all of the provided code.
12. Return to your visual editor with the `email_reminder.py` file from the GitHub repo open.
13. Copy all of the code to your clipboard.
14. Return to the Lambda console in your first tab, and paste the copied code into `lambda_function.py`.
15. Within the `lambda_function` code, on line 3, delete the `YOUR_SES_VERIFIED_EMAIL` placeholder, and type in the email you just used, making sure to leave the single quotes around it.
16. Click **Deploy** in the **Code source** menu bar above the code.

    Your changes have now been deployed.
17. Scroll up on the same page to the **Function Overview** section, and copy the **Function ARN** by clicking the copy icon next to it, and paste it into a text file for later use in the lab.

Objective 3: Create a Step Function State Machine

1. Open the AWS Step Functions console in a new tab by typing *Step Functions* into the search bar at the top of the screen. Right-click **Step Functions** from

the drop-down menu, and select **Open Link in New Tab**.

2. In the AWS Step Functions console tab, click **Get started**.
3. In the pop-up window, click **Create your own** to create your own workflow from scratch.
4. On the new page, click on **Code** and delete all of the provided code.
5. Open the `step-function-template.json` file to your text editor.
6. Copy all of the code to your clipboard.
7. Return to the Step Functions console tab, and paste the code on the left side of the screen.
8. On line 23, replace the `EMAIL_REMINDER_ARN` placeholder value with the copied ARN for `email` that you copied over onto a text file earlier, being sure to leave the double quotes around the ARN.
9. Click on the **Config** button at the top.
10. Leave the default name **MyStateMachine**.
11. Under **Permissions**, click the dropdown for the Execution role and select **RoleForStepFunction** under **Existing roles**.
12. Scroll to the top and click on **Create**.

    Your state machine should now be created.
13. On the **MyStateMachine** page, under **Details**, copy the ARN to your clipboard or text file. (If you are still in the State machine configuration page click on exit at the top)

Objective 4: Create the `api_handler` Lambda Function

1. Scroll up to the navigation line at the top of the page, click **Lambda** to return to the main Lambda console.
2. Click **Create function**.
3. Leave the **Author from scratch** option selected, and then set the following values:
    - **Function name:** *api_handler*
    - **Runtime: Python 3.12**

4. Expand **Change default execution role** by clicking the arrow next to it.
5. Below **Execution role**, select **Use an existing role**.
6. Once you select that, click into the empty drop-down menu that appears below **Existing role**.
7. Select **LambdaRuntimeRole** from the drop-down menu.
8. Click **Create function** at the bottom of the page.
9. Scroll down to **Code source** and from the **Environment** file list on the left, click `lambda_function.py` to display the function code.
10. Delete all of the provided code.
11. Open the `api_handler.py` file from the files you downloaded into your text editor..
12. Copy all of the code to your clipboard.
13. Return to the AWS Lambda console in your first tab, and paste the copied code in to the `lambda_function.py` function.
14. On line 6 copy the **MyStateMachine** arn from your clipboard or text editor making sure to leave ' each side of the arn.
15. Click **Deploy**. Keep this tab open.

Objective 5: Create the API Gateway

1. Open the Amazon API Gateway console in a new tab by typing *API Gateway* into the search bar at the top of the screen. Right-click **API Gateway** from the drop-down menu, and select **Open Link in New Tab**.
2. In your Amazon API Gateway console tab, scroll down to **REST API (the one that does not say Private)**, and then select **Build**.
3. Under **API Details**, select **New API**.
4. Then configure the following values:
    - **API name:** *reminders*
    - Leave **Description** blank.
    - **Endpoint Type:** *Regional*
5. Click **Create API** at the bottom of the screen.

You should now be in your API Gateway, and you will see that you don't have any methods defined for the resource.

6. You should now see the Resources page. Click **Create Resource**.
7. Under **Resource Name**, enter *reminders*.
8. Select the checkbox next to **CORS**.
9. Click **Create Resource** at the bottom of the page.
10. You should be back at the Resources page again.
11. Click **Create method**.
12. In the dropdown that appears under **Method type**, select **POST** and set the following values:
    - **Integration type: Lambda Function**
    - Enable **Lambda Proxy integration** by clicking the toggle.
    - **Lambda Region: us-east-1**
    - **Lambda Function:** Start typing, and then select, *api_handler*.
13. Click **Create method** at the bottom of the page.
14. Back at the Resources page, click **Deploy API** at the top.
15. In the **Deploy API** pop-up window, set the following values:
    - **Deployment stage: [New Stage]**
    - **Stage name:** *prod*
16. Click **Deploy**.

> **Note:** You may ignore any Web Application Firewall (WAF) permissions warning messages if received after deployment.

17. Copy the invoke url into your clipboard or text editor for use in the next objective.

Objective 6: Create and Test the Static S3 Website

1. Within the files you downloaded at the start of this lab, navigate to the static website folder.
2. Open the `formlogic.js` file in a text editor.

3. On line 5 of `formlogic.js`, delete the `UPDATETOYOURINVOKEURLENDPOINT` placeholder. Paste in the **Invoke URL** you previously copied from API Gateway, then append the following at the end of the url **/reminders**.
4. Save the local file.
5. Return to your first AWS Management Console tab.
6. Open the Amazon S3 console in a new tab by typing *S3* into the search bar at the top of the screen. Right-click **S3** from the drop-down menu, and select **Open Link in New Tab**.
7. In the Amazon S3 console, click **Create bucket**.
8. On the **Create bucket** page, under **Bucket Type** leave *General Purpose* selected and then for **Bucket name**, enter a globally unique bucket name.
9. Under **Object Ownership**, select **ACLs enabled**, and ensure **Bucket owner preferred** is selected.
10. Uncheck **Block *all* public access**.
11. Under **Block *all* public access**, select **I acknowledge that the current settings might result in this bucket and the objects within becoming public.**
12. Leave the other defaults, scroll down, and click **Create bucket**.
13. Select the new bucket under **Buckets** to open it.
14. On the bucket page, click **Upload**.
15. On the **Upload** page, click **Add files** and select all of your local files from the `static_website` folder, and click **Open**. Or, drag all of your local files from the `static_website` folder into the **Drag and drop** box under **Upload**.
    - `cat.png`
    - `error.html`
    - `formlogic.js`
    - `IMG_0991.jpg`
    - `index.html`
    - `main.css`
16. Once all of the files have been added, scroll down, and click **Permissions** to expand the access options.

17. Under **Predefined ACLs**, select **Grant public-read access**.
18. Under **Grant public-read access**, select **I understand the risk of granting public-read access to the specified objects.**
19. Click **Upload** at the bottom of the page.
20. Once you see the uploads have succeeded, click **Close** in the top right-hand corner of the page.
21. Click on the **Properties** tab under the bucket name.
22. Scroll down to **Static website hosting**, and click **Edit**.
23. On the **Edit static website hosting** page, under **Static website hosting**, select **Enable**.
24. Set the following values:
    - **Index document:** *index.html*
    - **Error document:** *error.html*
25. Click **Save changes** at the bottom of the page.
26. Scroll down again to **Static website hosting**, and click the URL below **Bucket website endpoint** to access the webpage.
27. Once you are on the static website, to test the service's functionality, set the following values:
    - **Seconds to wait:** *1*
    - **Message:** *Hello!*
    - ***someone@something.com***: Your personal email address (This has to be the same one you verified with Simple Email Service earlier.)
28. Under **Reminder Type**, select **email**.

    You should see **Looks ok. But check the result below!** above the **Required sections**, and you should see `{"Status":"Success"}` at the bottom of the page.
29. Navigate to your AWS Step Functions console tab.
30. In the **Executions** section, click on the refresh icon.

    You should see at least one execution displayed.
31. Click on one of the executions.

32. Scroll down to **Graph view** to view the event's visual workflow.
33. In a new browser tab or email client, navigate to your email. You should now see a reminder email from the service.

> **Note**: Check your spam folder if you do not see it in your inbox.

Conclusion

Congratulations — you've completed this hands-on lab!

Tools

Lab Diagram Instant Terminal

Credentials

How do I connect?

AWS Account

Username

cloud_user

Password

C)bzEXDbv3]1|Z8bS4i%

Open Link in Incognito Window

Additional Resources

Log in to the live AWS environment using the credentials provided. Make sure you're in the N. Virginia (`us-east-1`) region throughout the lab.

All of the resources needed to complete this lab are available from this GitHub repo.

Learning Objectives

0 of 6 completed

Optional: Run progress checks to confirm you've completed the objectives

- Validate email with Simple Email Service (SES)
- Configure Email Reminder Lambda Function
- Create Step Function State Machine
- Configure api_handler Lambda Function
- Create Rest API
- Create Static Website Running on S3 and Test