# 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.

## Create the Lambda Functions

1. In the GitHub repo, click on the file, email\_reminder.py to open it.
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 3.8

1. Expand Change default execution role by clicking the arrow next to it.
2. Below Execution role, select Use an existing role.
3. Once you select that, click into the empty drop-down menu that appears below Existing role.
4. Select LambdaRuntimeRole from the drop-down menu.
5. Click Create function at the bottom of the page.
6. Scroll down to Code source and from the Environment file list on the left, click lambda\_function.py to display the function code.
7. Delete all of the provided code.
8. Return to your other tab with the email\_reminder.py file from the GitHub repo open.
9. Click Raw above the code to display the raw function code.
10. Copy all of the code to your clipboard.
11. Return to the Lambda console in your first tab, and paste the copied code into lambda\_function.py.

### 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 Create identity.
    3. On the Create identity page, under Identity type, select Email address.
    4. Under Email address, enter your personal email address (or an email address you created specifically for this hands-on lab).
    5. Scroll down and click Create identity.
    6. 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.

* + 1. Return to the AWS SES console tab.

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

### Finish the email Lambda Function Setup

1. Return to the Lambda console in your first tab.
2. 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.
3. Click Deploy in the Code source menu bar above the code.

Your changes have now been deployed.

1. 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.

### Create the sms Lambda Function

1. In 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:
4. Function name: sms
5. Runtime: Python 3.8
6. Expand Change default execution role by clicking the arrow next to it.
7. Below Execution role, select Use an existing role.
8. Once you select that, click into the empty drop-down menu that appears below Existing role.
9. Select LambdaRuntimeRole from the drop-down menu.
10. Click Create function at the bottom of the page.
11. Scroll down to Code source and from the Environment file list on the left, click lambda\_function.py to display the function code.
12. Delete all of the provided code.
13. Return to your GitHub repo tab, and click the back icon in your browser twice, to return to the main page of the repo.
14. Click the sms\_reminder.py file.
15. Click Raw to display the raw function code.
16. Copy all of the code to your clipboard.
17. Return to the AWS Lambda console in your first tab, and paste the copied code into lambda\_function.py.
18. Click Deploy. Keep this tab open.

### 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 3.8
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. Return to your GitHub repo tab, and click the back icon in your browser twice, to return to the main page of the repo.
12. Click the api\_handler.py file.
13. Click Raw above the code to display the raw function code.
14. Copy all of the code to your clipboard.
15. Return to the AWS Lambda console in your first tab, and paste the copied code in to lambda\_function.py.
16. Click Deploy. Keep this tab open.

### 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 line under the Review Hello World example, click here at the end of the sentence to create your own step function.

Note: If prompted with a Leave page? warning pop-up window, click Leave, and then click here on the line under Review Hello World example again.

1. On the Create state machine page, under Choose authoring method, select Write your workflow in code.
2. Under Type, leave Standard selected.
3. Scroll down to Definition and delete all of the provided code.
4. Return to your GitHub repo tab, and click the back icon in your browser twice, to return to the main page of the repo.
5. Click the step-function-template.json file to open it.
6. Click Raw above the code to display the raw function code.
7. Copy all of the code to your clipboard.
8. Return to the Step Functions console tab, and paste the code under Definition.
9. On lines 34 and 52, replace the EMAIL\_REMINDER\_ARN placeholder value with the copied ARN for email that you should have copied over onto a text file earlier, being sure to leave the double quotes around the ARN.
10. On lines 40 and 62, replace the TEXT\_REMINDER\_ARN placeholder value with the copied ARN for sms.

* Return to the Lambda console in your first tab.
* In the navigation line at the top of the page, click Functions.
* Under Functions, select sms.
* Under Function overview on the right, copy the Function ARN to your clipboard by clicking the copy icon next to it.
* Return to your Step Functions console tab.
* Replace the two instances of TEXT\_REMINDER\_ARN on lines 40 and 62, being sure to leave the double quotes around the ARN.

You should now see the updated function diagram show up to the right of the code.

1. Click Next at the bottom of the screen.
2. On the Specify details page, leave the default name MyStateMachine and under Permissions, select Choose an existing role.
3. Leave the auto-populated, RoleForStepFunction under Existing roles.
4. Scroll down, and click Create state machine.

Your state machine should now be created.

1. On the MyStateMachine page, under Details, copy the ARN to your clipboard.
2. Return to the Lambda console in your first tab.
3. In the navigation line at the top of the page, click Functions.
4. Under Functions, select the api\_handler function.
5. Scroll down to Code source and view the lambda\_function.py code.
6. On line 6, replace the STEP\_FUNCTION\_ARN placeholder with the ARN you just copied.
7. Click Deploy.

### 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. In the Create your first API pop-up window, click OK.
4. Under Choose the protocol, leave REST selected.
5. Under Create new API, select New API.
6. Under Settings, set the following values:

* API name: reminders
* Leave Description blank.
* Endpoint Type: Regional

1. 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.

1. Click on the / under Resources.
2. Click Actions > Create Resource.
3. On the New Child Resource page, next to Resource Name, enter reminders, which will also auto-populate /reminders as the Resource Path.
4. Select the checkbox next to Enable API Gateway CORS.
5. Click Create Resource at the bottom of the page.
6. Then, click /reminders, and click Actions > Create Method.
7. In the dropdown that appears under /reminders, select POST and click the adjacent checkmark icon.
8. Under /reminders - POST - Setup, set the following values:

* Integration type: Lambda Function
* Use Lambda Proxy integration: Select the checkbox.
* Lambda Region: us-east-1
* Lambda Function: Start typing, and then select, api\_handler.

1. Click Save at the bottom of the page
2. In the Add Permission to Lambda Function pop-up window, click OK.
3. Click Actions > Deploy API.
4. In the Deploy API pop-up window, set the following values:

* Deployment stage: [New Stage]
* Stage name: prod

1. Click Deploy.

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

1. At the top of the page, under prod Stage Editor, copy the Invoke URL and paste it into a text file for later use in the lab..

### Create and Test the Static S3 Website

1. Return to your GitHub repo tab, and click the back icon in your browser twice, to return to the main page of the repo.
2. Click Code in the top right above the files in the repo, and select Download ZIP to download all of the files in the repo.
3. Unzip and open the downloaded file, and then open your local static\_website folder.
4. Open the formlogic.js file in a text editor.
5. On line 5 of formlogic.js, delete the UPDATETOYOURINVOKEURLENDPOINT placeholder. Paste in the Invoke URL you previously copied from API Gateway, being sure to keep /reminders on the end of the string.
6. Save the local file.
7. Return to your first AWS Management Console tab.
8. 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.
9. In the Amazon S3 console, click Create bucket.
10. On the Create bucket page, under Bucket name, enter a globally unique bucket name.
11. Under Object Ownership, select ACLs enabled, and ensure Bucket owner preferred is selected.
12. Uncheck Block all public access.
13. Under Block all public access, select I acknowledge that the current settings might result in this bucket and the objects within becoming public.
14. Leave the other defaults, scroll down, and click Create bucket.
15. Select the new bucket under Buckets to open it.
16. On the bucket page, click Upload.
17. 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

1. Once all six of the files have been added, scroll down, and click Permissions to expand the access options.
2. Under Predefined ACLs, select Grant public-read access.
3. Under Grant public-read access, select I understand the risk of granting public-read access to the specified objects.
4. Click Upload at the bottom of the page.
5. Once you see the uploads have succeeded, click Close in the top right-hand corner of the page.
6. Click on the Properties tab under the bucket name.
7. Scroll down to Static website hosting, and click Edit.
8. On the Edit static website hosting page, under Static website hosting, select Enable.
9. Set the following values:

* Index document: index.html
* Error document: error.html

1. Click Save changes at the bottom of the page.
2. Scroll down again to Static website hosting, and click the URL below Bucket website endpoint to access the webpage.
3. 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.)

1. 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.

1. 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.

1. Navigate to your AWS Step Functions console tab.
2. In the Executions section, click on the refresh icon.

You should see at least one execution displayed.

1. Click on one of the executions.
2. Scroll down to Graph inspector to view the event's visual workflow.

## Conclusion

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