

Step 7: Create Function URL

1. Go to the **Configuration** tab.
2. Under the left-side menu, click **Function URL**.
3. Click **Create function URL**.
4. For **Auth type**, choose **None**.
5. Click **Save**.

Assignment 15

Step 8: Test the URL

1. Once the Function URL is created, click on it.
 2. A new browser tab opens, showing your Lambda function output (e.g., "Welcome from Sneha!").
- ⚠ If you see an error, ensure your function code returns a valid HTTP response.

✔ Step 9: Delete Resources

1. Go back to **Configuration > Function URL** and **delete the URL**.
2. Then return to the **Lambda dashboard**, select your function, and click **Delete**.

🔗 Expected Output

- After deployment and testing, your function should return: "Welcome from Sneha!"
- You should be able to view this output via the Test button and directly from the Function URL.

The screenshot shows the AWS Lambda console interface. At the top, a green banner indicates "Your changes have been saved." Below this, the breadcrumb navigation shows "Lambda > Functions > func01". The main navigation tabs include "Code", "Test", "Monitor", "Configuration" (which is selected), "Aliases", and "Versions". On the left sidebar, the "Function URL" option is highlighted under the "General configuration" section. The main content area displays the "Function URL" configuration for "func01". It includes a status message: "Your function URL is public. Anyone with the URL can access your function." Below this, a table shows the configuration details: "Function URL" is "https://6ncw4huzzuqekxv4iuv5ivagy40ggvef.lambda-url.us-east-1.on.aws/", "Auth type" is "NONE", and "Invoke mode" is "BUFFERED". It also shows "Creation time" and "Last modified" as "11 seconds ago". At the bottom, it states "CORS (Not enabled)". On the right side, there is a "Tutorials" section titled "Create a simple web app" with a description and a "Start tutorial" button. The footer of the console shows "CloudShell", "Feedback", and copyright information for Amazon Web Services.

Step 4: Modify the Code

1. Wait for the function page to load. You'll be taken to the function dashboard.
2. Under the **Code** tab, locate and open the index.mjs or main file (for Python, it might be lambda_function.py).
3. Replace any occurrence of the word “**lambda**” with “**sneha**” in the sample code.

Example (Node.js):

```
export const handler = async (event) => {  
  const response = {  
    statusCode: 200,  
    body: JSON.stringify('Welcome from Sneha!'),  
  };  
  return response;  
};
```

4. Click **File > Save** to save the code

Step 5: Create and Run a Test Event

1. Click on the **Test** button (top-right).
2. Select “**Create new test event.**”
3. Give it an **Event name**, e.g., eve1.
4. Leave the default JSON data as is (you don't need to change anything).
5. Click **Save**.
6. Now click **Test** to execute the Lambda function.

Note: If you don't see your message change (e.g., “sneha”), it means you haven't deployed the latest code yet.

Step 6: Deploy and Re-Test

1. Click the **Deploy** button to apply your code changes.
2. Click **Test** again to see the updated result.

The screenshot shows the AWS Lambda console interface. At the top, there's a navigation bar with the AWS logo, a search bar, and various utility icons. Below this, the breadcrumb trail indicates the current location: Lambda > Functions > func01. The main content area has several tabs: Code, Test (which is selected), Monitor, Configuration, Aliases, and Versions. A green notification banner at the top of the Test tab states "Executing function: succeeded (logs [2])" with a link to view details. Below this, the "Test event" section provides instructions on how to invoke the function. It includes buttons for "Delete", "CloudWatch Logs Live Tail", "Save", and "Test". Under "Test event action", there are two radio buttons: "Create new event" and "Edit saved event" (which is selected). The "Event name" field contains the value "eve1". Below the event name, there's a section for "Event JSON" with a "Format JSON" button. The bottom of the console shows a snippet of the JSON event data: {"key1": "value1"}. On the right side of the console, there's a sidebar with "Info" and "Tutorials" tabs. The "Tutorials" tab is active, showing a tutorial titled "Create a simple web app" with a list of steps: "Build a simple web app, consisting of a Lambda function with a function URL that outputs a webpage" and "Invoke your function through its function URL". A "Start tutorial" button is at the bottom of the tutorial card. The footer of the console displays the URL: https://us-east-1.console.aws.amazon.com/lambda/home?region=us-east-1#/functions/func01?tab=testing, the copyright notice "© 2025, Amazon Web Services, Inc. or its affiliates.", and links for "Privacy", "Terms", and "Cookie preferences".

Assignment 15

Create Serverless Computing Service using AWS Lambda

🔧 Objective

To create and deploy a simple AWS Lambda function that prints a custom welcome message — demonstrating serverless computing on AWS.

🚀 Part 1: Creating the Lambda Function

Step 1: Open Lambda Service

1. Log in to your AWS Console: <https://aws.amazon.com/console/>
2. In the **Search bar**, type **Lambda** and click on it.

Explanation:

AWS Lambda lets you run code without managing servers. You only focus on writing the function logic.

The screenshot shows the AWS Lambda console interface. At the top, a green notification bar states: "Successfully created the function func01. You can now change its code and configuration. To invoke your function with a test event, choose 'Test'." Below this, the "func01" function overview is displayed. It includes tabs for "Diagram" and "Template", a visual representation of the function with its layers, and buttons for "Add trigger" and "Add destination". On the right, a "Description" panel shows the function's last modified time (13 seconds ago), its ARN (arn:aws:lambda:us-east-1:52167398133:func01), and its URL. A sidebar on the right offers a tutorial titled "Create a simple web app" with a "Start tutorial" button. The bottom of the console shows the "CloudShell" button and a footer with copyright information and links to Privacy, Terms, and Cookie preferences.

Step 2: Create the Function

1. Click on the “Create function” button.
2. Select “Author from scratch.”

Step 3: Set Function Details

- **Function name:** e.g., func_x1
 - **Runtime:** Choose **Python 3.9** or any preferred runtime (Node.js, etc.)
- Tip:** The runtime determines what programming language your Lambda function will use.
3. Scroll down and leave all other settings as **default**.
 4. Click **Create function**.

This screenshot shows the AWS Lambda console with the "func01" function selected. The left sidebar contains navigation icons and buttons for "Deploy (Ctrl+Shift+U)" and "Test (Ctrl+Shift+I)". The main area is split into two panes. The top pane shows the Python code for the lambda_handler function:

```
1 import json
2
3 def lambda_handler(event, context):
4     # TODO implement
5     return {
6         'statusCode': 200,
7         'body': json.dumps('Hello from sndeiv!')}
8
9
```

 The bottom pane shows the "Execution Results" for a test event named "eve1", indicating a "Status: Succeeded" and displaying the response:

```
{
  "statusCode": 200,
  "body": "\"Hello from sndeiv!\""
}
```

 A right sidebar provides a tutorial for "Create a simple web app". The bottom of the console shows the "Amazon Q" logo and system information like "Ln 3, Col 11" and "Python".