# AWS Deployment Guide for Serverless News Aggregator

This guide will walk you through deploying your news aggregator application on AWS using a cost-effective, serverless architecture.

## Architecture Overview

* **Frontend:** index.html is hosted as a static website on **Amazon S3**.
* **Database:** A **Amazon DynamoDB** table stores all the news articles.
* **Backend API:** An **AWS Lambda** function (api\_handler\_lambda) provides the data to the frontend, exposed via **Amazon API Gateway**.
* **Scheduled Fetcher:** A second **AWS Lambda** function (fetch\_news\_lambda) runs on an hourly schedule using **Amazon EventBridge** to fetch and store news.

## Deployment Steps

### Step 1: Create a DynamoDB Table

1. Navigate to the **DynamoDB** service in the AWS Console.
2. Click **Create table**.
3. **Table name:** NewsArticles
4. **Partition key:** link (Type: String). This will be the unique URL of the news article.
5. Leave all other settings as default and click **Create table**.

### Step 2: Create an IAM Role for Lambda

Your Lambda functions need permission to interact with other AWS services.

1. Navigate to the **IAM** service.
2. Go to **Roles** and click **Create role**.
3. **Trusted entity type:** Select **AWS service**.
4. **Use case:** Select **Lambda**. Click **Next**.
5. On the "Add permissions" page, search for and add the following two policies:
   * AWSLambdaBasicExecutionRole (for logging)
   * AmazonDynamoDBFullAccess (for simplicity; in production, you would create a more restrictive custom policy).
6. Click **Next**.
7. **Role name:** LambdaNewsAggregatorRole.
8. Click **Create role**.

### Step 3: Create the News Fetcher Lambda

This function will fetch news from RSS feeds and store it in DynamoDB.

1. Navigate to the **Lambda** service.
2. Click **Create function**.
3. Select **Author from scratch**.
4. **Function name:** FetchNewsFunction.
5. **Runtime:** **Python 3.9**.
6. **Architecture:** x86\_64.
7. **Permissions:** Expand "Change default execution role", select **Use an existing role**, and choose the LambdaNewsAggregatorRole you created.
8. Click **Create function**.
9. **Package the Code:**
   * The feedparser library is not included in the Lambda environment. You must package it with your code.
   * On your local machine, create a new folder (e.g., fetcher-package).
   * Save the fetch\_news\_lambda.py code into this folder.
   * Open your terminal, navigate into the folder, and run: pip install feedparser -t .
   * Zip the *contents* of the fetcher-package folder. **Do not zip the folder itself.**
10. **Upload and Configure:**
    * In the FetchNewsFunction console, under **Code source**, click **Upload from** and select **.zip file**. Upload your zip file.
    * Go to the **Configuration** > **General configuration** tab and click **Edit**.
    * Set the **Timeout** to **1 minute** (fetching can sometimes be slow). Click **Save**.
    * Go to the **Configuration** > **Environment variables** tab. Click **Edit**.
    * Add a new variable:
      + **Key:** DYNAMODB\_TABLE
      + **Value:** NewsArticles
    * Click **Save**.
11. **Set up the Schedule (Trigger):**
    * In the function overview, click **Add trigger**.
    * Select **EventBridge (CloudWatch Events)**.
    * Choose **Create a new rule**.
    * **Rule name:** HourlyNewsFetch.
    * **Schedule expression:** rate(1 hour).
    * Click **Add**.

### Step 4: Create the API Handler Lambda

This function will serve the news to your frontend.

1. Follow the same steps 1-8 from "Step 3" to create another Lambda function.
2. **Function name:** ApiHandlerFunction.
3. **Runtime:** **Python 3.9**.
4. **Execution role:** Use the existing LambdaNewsAggregatorRole.
5. **Upload Code:** Copy the code from api\_handler\_lambda.py and paste it directly into the inline code editor in the Lambda console. Click **Deploy**.
6. **Add Environment Variable:** Just like before, add an environment variable DYNAMODB\_TABLE with the value NewsArticles.

### Step 5: Create the API Gateway

1. Navigate to the **API Gateway** service.
2. Find the **REST API** box and click **Build**.
3. Select **New API**.
4. **API name:** NewsAggregatorAPI.
5. Leave other settings as default and click **Create API**.
6. **Create Resources and Methods:**
   * Click the **Actions** dropdown and select **Create Resource**.
   * **Resource Name:** news. Click **Create Resource**.
   * With the /news resource selected, click **Actions** and **Create Method**.
   * Select **GET** from the dropdown and click the checkmark.
   * **Integration type:** **Lambda Function**.
   * Check **Use Lambda Proxy integration**.
   * **Lambda Function:** Start typing ApiHandlerFunction and select it.
   * Click **Save**.
   * Repeat this process to create another resource:
   * **Resource Name:** track-view.
   * **Method:** **POST**.
   * Integrate it with the same ApiHandlerFunction.
7. **Enable CORS:**
   * Select the /news resource. Click **Actions** and **Enable CORS**.
   * Click **Enable CORS and replace existing CORS headers**.
   * Repeat for the /track-view resource.
8. **Deploy the API:**
   * Click **Actions** and **Deploy API**.
   * **Deployment stage:** **New Stage**.
   * **Stage name:** prod.
   * Click **Deploy**.
   * You will be taken to the stage editor. Copy the **Invoke URL**. This is your API\_BASE\_URL.

### Step 6: Host the Frontend on S3

1. Navigate to the **S3** service.
2. Click **Create bucket**.
3. **Bucket name:** Choose a globally unique name (e.g., my-unique-news-aggregator-app).
4. **Region:** Choose the same region as your other resources.
5. **Uncheck "Block all public access"** and acknowledge the warning. Click **Create bucket**.
6. **Configure for Website Hosting:**
   * Go into your new bucket and select the **Properties** tab.
   * Scroll down to **Static website hosting** and click **Edit**.
   * Select **Enable**.
   * **Index document:** index.html.
   * Click **Save changes**.
7. **Set Bucket Policy:**
   * Go to the **Permissions** tab.
   * Under **Bucket policy**, click **Edit**.
   * Paste the following policy, replacing YOUR\_BUCKET\_NAME with your actual bucket name.

{  
 "Version": "2012-10-17",  
 "Statement": [  
 {  
 "Sid": "PublicReadGetObject",  
 "Effect": "Allow",  
 "Principal": "\*",  
 "Action": "s3:GetObject",  
 "Resource": "arn:aws:s3:::YOUR\_BUCKET\_NAME/\*"  
 }  
 ]  
}

* + Click **Save changes**.

1. **Upload the Frontend:**
   * First, edit your index.html file locally. Replace the placeholder YOUR\_API\_GATEWAY\_URL with the **Invoke URL** you copied from API Gateway.
   * Go to the **Objects** tab of your bucket.
   * Click **Upload**, then **Add files**, and select your modified index.html.
   * Click **Upload**.

### Step 7: Launch!

Go back to the S3 bucket's **Properties** tab and find the **Static website hosting** section again. Your website URL will be listed there. Click it to see your live news aggregator!

**Note:** The first time you load the site, it might be empty. The fetcher Lambda needs to run first. You can trigger it manually by going to its Lambda console page, clicking the **Test** tab, configuring a dummy test event, and clicking **Test**. After a minute, refresh your website.