

# **AWS Starter Kit**

[Introduction](#)

[Overview of Different Services](#)

[Setup](#)

[S3](#)

[EC2](#)

[Lambda](#)

# Introduction

Welcome to Hack@Brown's cloud starter kit! We will be using Amazon Web Services (AWS) as our cloud platform. This guide will take you through three different services and explain what they are and how they can be used.

*Note: There are other cloud services besides AWS such as Google Cloud, Microsoft Azure, IBM Cloud, etc., but for the sake of this tutorial, we will be using AWS.*

## Overview

This is a quick overview of the three services to give you an idea of what each one is about.

### S3

- S3 is an online, bulk storage services for all types of files that are accessible from most devices
- Provides reliable, durable storage that you can access from anywhere via the internet
- Allows you to store and retrieve any amount of data, like images, videos, or backups, as objects

### EC2

- EC2 is a virtual computing environment, known as *instances*. It is essentially a virtual server that is run on Amazon's cloud resources
- Launch virtual machines with custom configurations as needed, without upfront hardware investments
- Run any application, from simple websites to complex machine learning algorithms, on scalable virtual servers

- 

### Lambda

- AWS Lambda is a compute service that lets you run code without provisioning or managing servers
- Automatically runs your code when triggered by specific events, like HTTP requests or file uploads
- Automatically scales to handle any number of requests, making it ideal for unpredictable or variable workloads

## Setup

The first step is to create a free account on AWS. Follow this [link](#) and the steps that they outline to set up a free AWS account.

*Note: It will ask you for a payment method, but it will not charge you as long as you do not exceed the free tier usage policy. Read more about the free tier usage policy [here](#).*

# S3

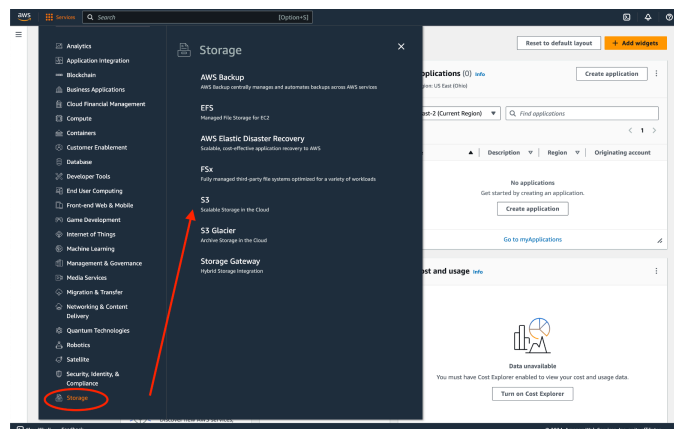
## What is S3?

AWS S3 is a simple storage service. Generally, it is an online, bulk storage service for all types of files that are accessible from most devices. In Amazon's words, S3 is a "simple web service that you can use to store and retrieve any amount of data, at any time, from anywhere on the web. It gives any user access to the same highly scalable, reliable, fast, inexpensive infrastructure that Amazon uses to run its own global network of websites. The service aims to maximize benefits of scale and to pass those benefits on to users."

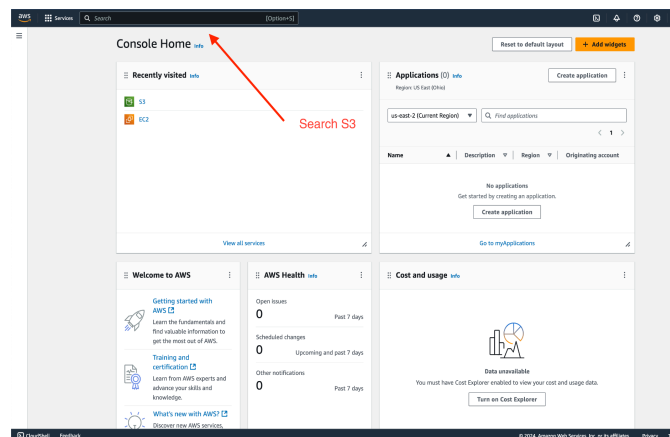
The free tier of S3 includes 5GB of Amazon S3 standard storage, 20,000 get requests, 2,000 put requests, and 15GB of data transfer out each month for one year.

## Navigating the AWS console

You can access S3 under Storage when you click on their Services tab. Alternatively, you could directly search for S3 in their search bar!



*Services → Storage → S3*



*Searching S3 with the search bar*

Once you get to the S3 Console, you'll notice that you have the option to create buckets. The lowest, or root level folders, are called buckets. Within buckets, you can store objects or subfolders, which can also store objects. Objects are just files, and can be stored in these buckets or subfolders.

## Creating a Bucket

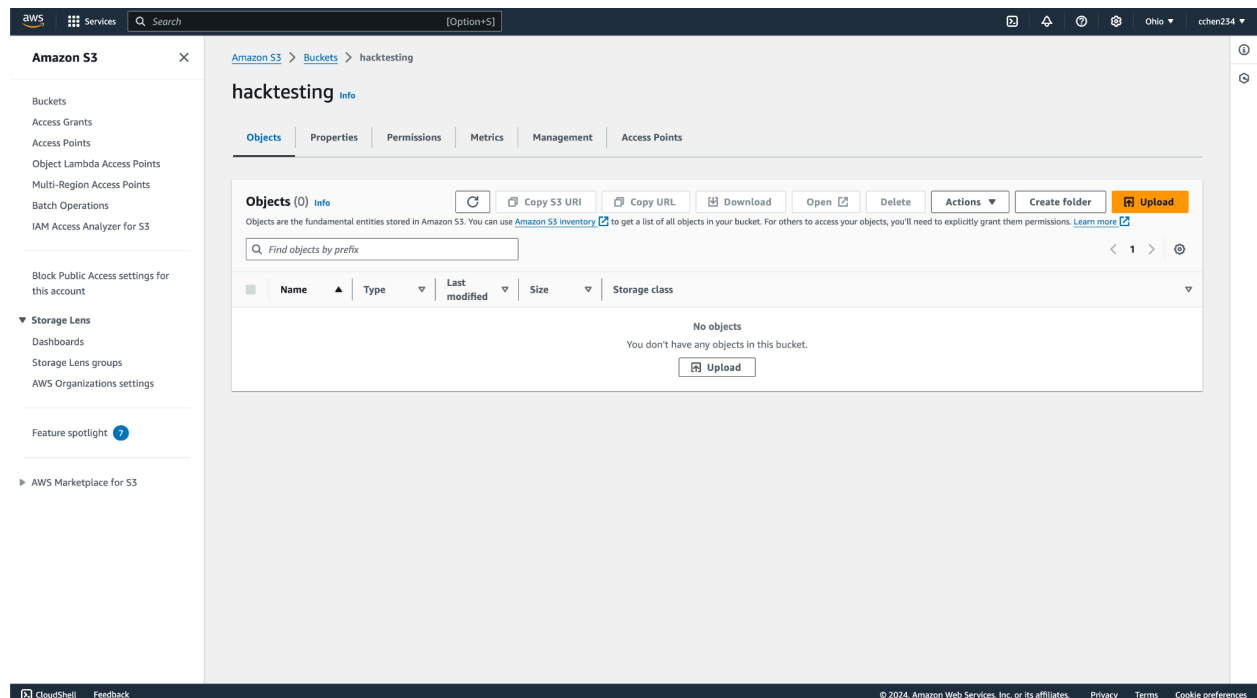
Bucket names must

- Be 3 to 62 characters long
- Be unique across ALL of AWS
- Only contain lowercase letters, numbers, and hyphens
- Not have the format of an IP address (e.g., 123.456.7.8)

When you create a bucket from the S3 console, you may or may not be prompted to specify the geographic region that it will occupy. Any data uploaded to that bucket will be stored in a data center in that region. The best practice is to select the region that is located physically closest to where you are. However, if you are creating a service to provide data to customers in a different region, you should create the bucket in that region.

## Creating folders, uploading objects

Once you have created a bucket, you can click on the bucket name in your console to create or upload files and objects.



In the top right corner of the middle console, you can choose to either create a folder (which you may want to do for more organization/subdivision of content) or to immediately upload an object. Folders do not need to follow the same naming conventions as buckets. You can think of the bucket as a file directory. You can add files to the bucket as you want, or create more subdirectories within the root directory!

## Object Durability and Availability

Durability is the percent chance over one year that a file stored in S3 will NOT be lost, corrupted, or deleted. Availability is the percentage over a one year time period that an object stored in S3 will be accessible (able to be downloaded, modified, etc.).

## Viewing Properties

When you click on a bucket, in your new bucket console, you have the option to view the properties of the bucket at the top dashboard. The top section of Properties displays general information about the bucket, including its name, region, creation date, and owner. However, there are other important properties that are worth discussing.

## Storage Classes

Storage classes are the general classification assigned to an object in S3. Each storage class differs in storage cost, object availability, object durability, and frequency of access to the object. Each object is assigned the standard class by default, but you can choose between Standard, RRS, and S3-IA in the 'Intelligent-Tiering Archive configurations' section of properties. Here are the different storage classes:

- **Standard:** Although it is the default class, it is the most expensive. It has 99.999999999% object durability and 99.99% object availability. Best for objects that you frequently access (on a daily basis) or objects that are extremely important and need high durability
- **Reduced Redundancy Storage (RRS):** Designed for non-critical, reproducible objects and is less expensive than Standard. This is because it only has 99.99% object durability, though it also has 99.9% object availability. Best for backups or files that can be reproduced
- **Infrequent Access (S3-IA):** Designed for objects that are not accessed as frequently, but need to be immediately available when you do need access. Like Standard, it has 99.999999999% durability, but only 99.90% availability. It is less expensive than both Standard and RRS. Best for objects that you need infrequently, but need a guarantee of quick access when you need it
- **Glacier:** Designed for long-term storage, and as the name implies, it may take hours for objects stored in Glacier to be retrieved. It has 99.999999999% durability and is the cheapest S3 storage class. Best for large, archival storage. To move an object to the Glacier class, you need to use object life cycles. The change to Glacier may take 1-2 days

## Object Lifecycles

Object lifecycles are a set of rules that automate the migration of an object's storage classes to a different storage class (or deletion), based on a specific time interval. By using an object lifecycle, you can automate the process of changing a file's storage class to meet your usage needs (that you know will change overtime) in order to minimize your S3 storage costs.

- To create a lifecycle policy, navigate to the bucket's management page (in the same dashboard as the properties)
- Follow the lifecycle management console to create new lifecycles or edit existing ones

## Permissions

Allows you to control who can view, access, and use specific buckets and objects. Navigate to the bucket's permission page (in the same dashboard as the properties). From here, you can change your bucket policies, block public access (bucket settings), change object ownership, edit the general access control list (ACL), and edit cross-origin resource sharing (CORS).

## Object Versioning

This allows you to keep track of and store all old/new version of an object so that you can access and use an older version if you need to. You could think of this as an analogous version of revision history in Google Docs!

- Versioning is either ON or OFF (default). Once versioning is on, you can “suspend” versioning, but not fully turn it off. Suspending versioning only prevents versioning going forward, so all previous objects with version will still maintain their older versions. Versioning can only be set on the bucket level and applies to all objects in the bucket
- To enable versioning, navigate to the bucket's properties and find the bucket versioning console. By default, versioning is off, so you can press the “Edit” button and click “Enable” to enable it
- You can reupload the object without worrying about moving the older version. You will need to specify the storage class of that object again, or else the new version will be set to Standard by default. If you suspend versioning and upload then reupload a new object, the older versions will not be saved
- To view the version of an object, in the bucket or folder where the object is contained, select the object by clicking in the horizontal space to the right of the object (without clicking on its name). In the top middle bar to the right of “Versions,” select “Show.” Now you can see a list of the previous versions of that object

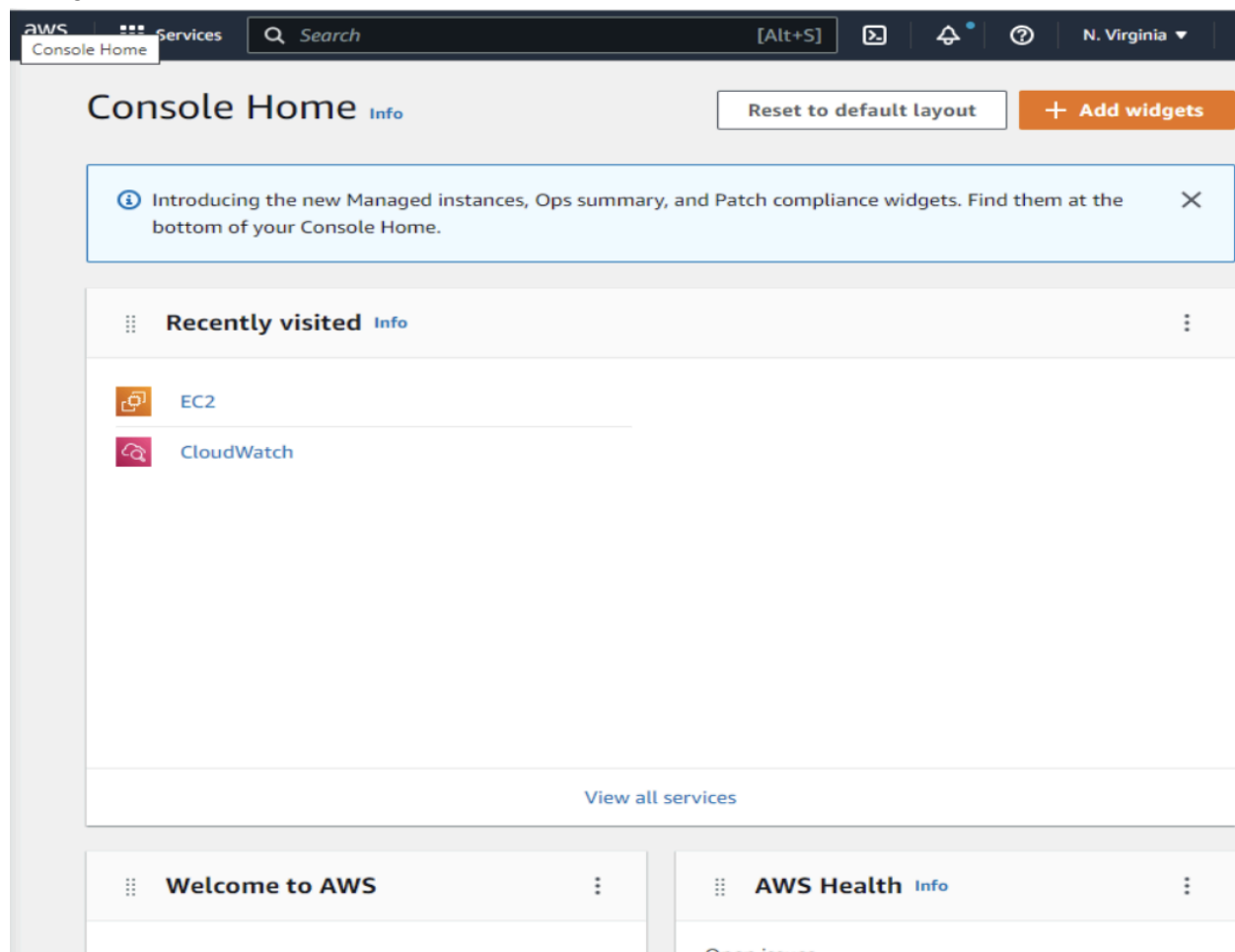
## EC2

### What is EC2?

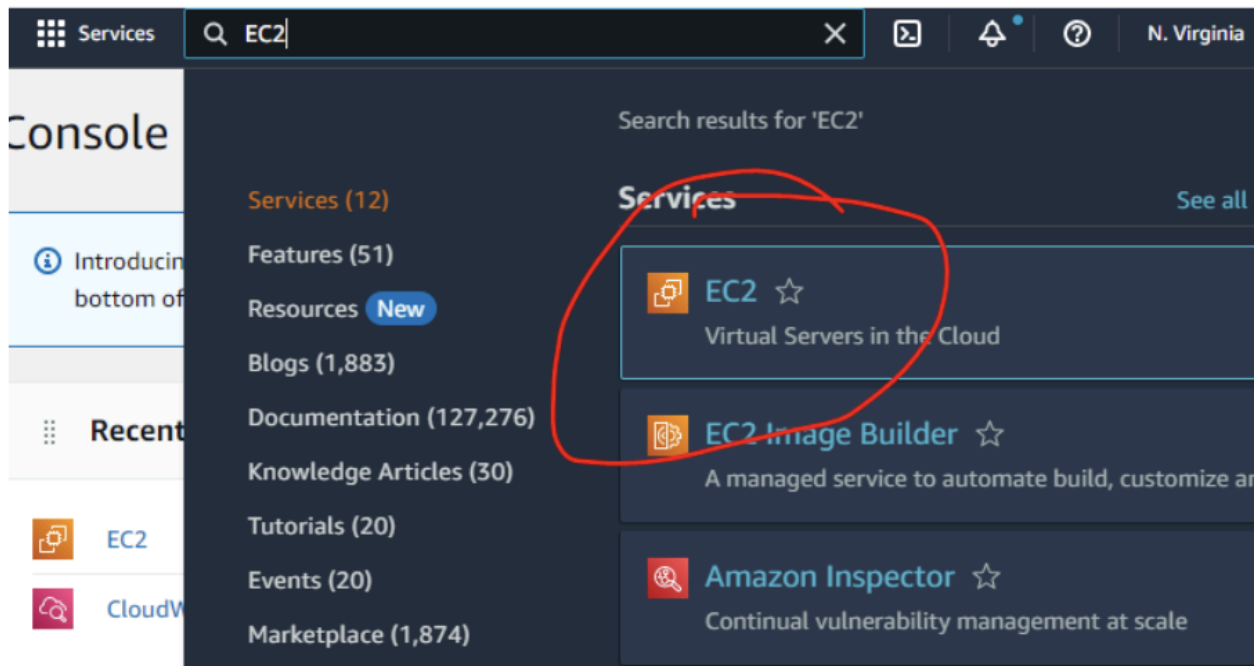
AWS EC2 (Elastic Compute Cloud) provides virtual computing environments called instances. These instances function as virtual servers that run on Amazon's cloud infrastructure, allowing you to launch and manage applications without needing physical hardware. You can choose from various instance types, which offer different configurations of CPU, memory, storage, and networking capabilities, depending on your specific needs. To make setup easier, EC2 provides Amazon Machine Images (AMIs)—preconfigured templates that contain everything your server needs, including the operating system, libraries, and additional software. EC2 offers flexibility in scaling and customizing resources, whether you need a small server for testing or a large one for production workloads.

### Launching an Instance (GUIDE)

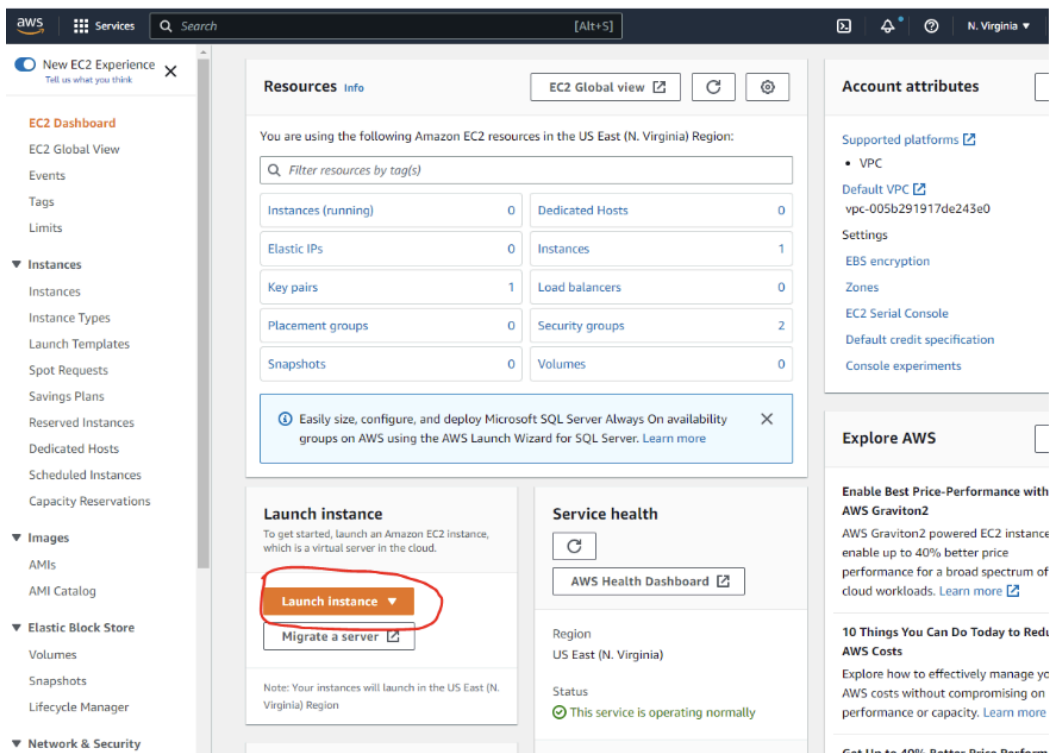
First go to the AWS console:



Then type “EC2” in the search bar at the top and navigate to that page.



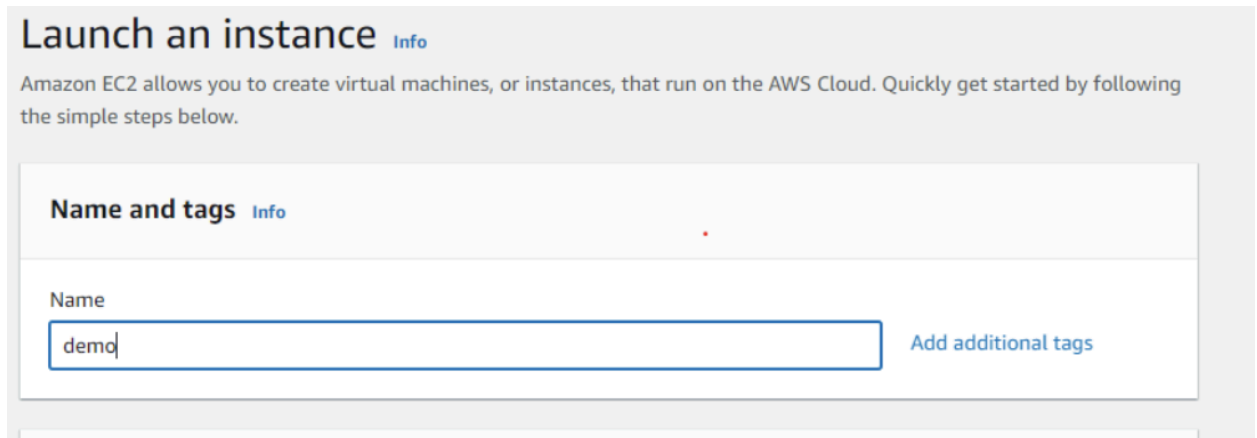
Then click on the “Launch Instance” button.





You will then be taken to a form to fill out to launch an instance. Here are a few steps for filling out the form:

- 1) The first thing that will need to be filled out is the name of your instance



## Launch an instance [Info](#)

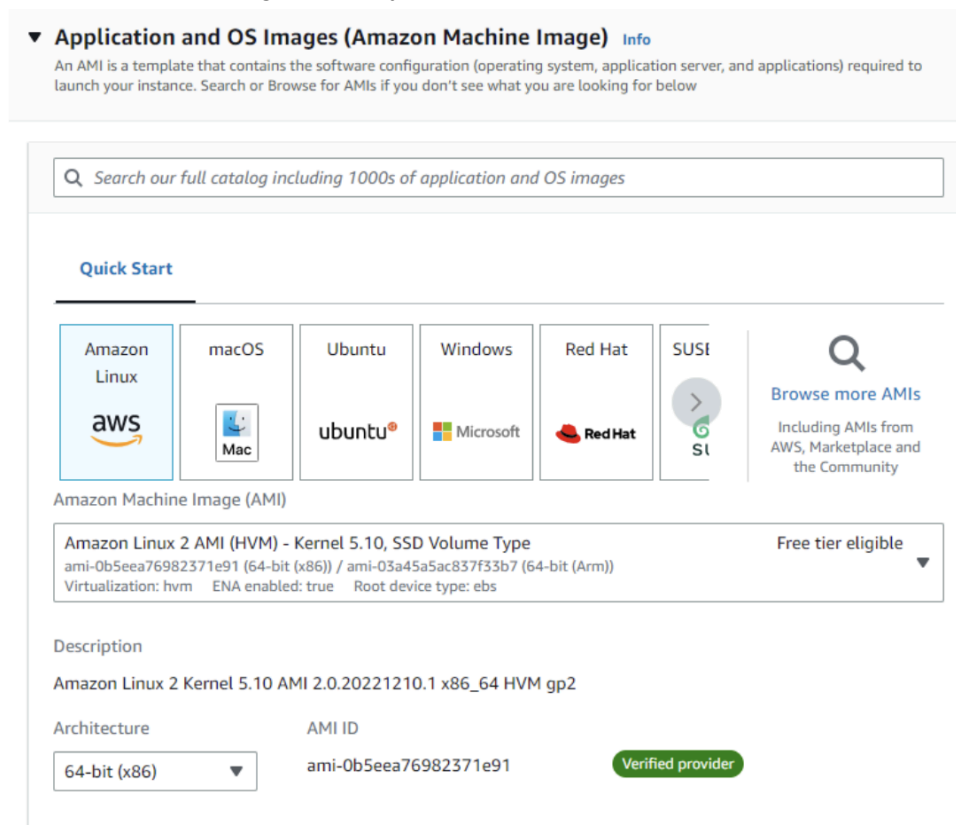
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.

### Name and tags [Info](#)

Name

[Add additional tags](#)

- 2) Then you will be asked to choose an Amazon Image machine. For this tutorial, we will be using the Amazon Linux image because it is free tier eligible. There are other images that are free as well that you can look into, but please be sure to choose the free tiers in order to not be charged money



### ▼ Application and OS Images (Amazon Machine Image) [Info](#)

An AMI is a template that contains the software configuration (operating system, application server, and applications) required to launch your instance. Search or Browse for AMIs if you don't see what you are looking for below

#### Quick Start

Amazon Linux  
aws

macOS  
Mac

Ubuntu  
ubuntu

Windows  
Microsoft

Red Hat  
Red Hat

SUSE  
SUSE

[Browse more AMIs](#)  
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Amazon Linux 2 AMI (HVM) - Kernel 5.10, SSD Volume Type  
ami-0b5eea76982371e91 (64-bit (x86)) / ami-03a45a5ac837f33b7 (64-bit (Arm))  
Virtualization: hvm ENA enabled: true Root device type: ebs

Free tier eligible ▼

Description

Amazon Linux 2 Kernel 5.10 AMI 2.0.20221210.1 x86\_64 HVM gp2

Architecture	AMI ID
64-bit (x86) ▼	ami-0b5eea76982371e91 <span>Verified provider</span>

- 3) For instance type, we can use the **t2.micro** because it is also in the free tier
- 4) We will need to specify a key pair (login) in order to connect to your instance later on
  - a) Click on "Create a new key pair"

▼ **Key pair (login)** [Info](#)

You can use a key pair to securely connect to your instance. Ensure that you have access to the selected key pair before you launch the instance.

Key pair name - *required*

Select ▼

⌂ [Create new key pair](#)

- b) Create a new name for your pair, select “RSA,” and choose either the **.pem** or the **.ppk** format, depending on what software you have

**Create key pair** ✕

**Key pair name**  
Key pairs allow you to connect to your instance securely.

Enter key pair name

The name can include up to 255 ASCII characters. It can't include leading or trailing spaces.

**Key pair type**

☒ **RSA**  
RSA encrypted private and public key pair

☐ **ED25519**  
ED25519 encrypted private and public key pair

**Private key file format**

☒ **.pem**  
For use with OpenSSH

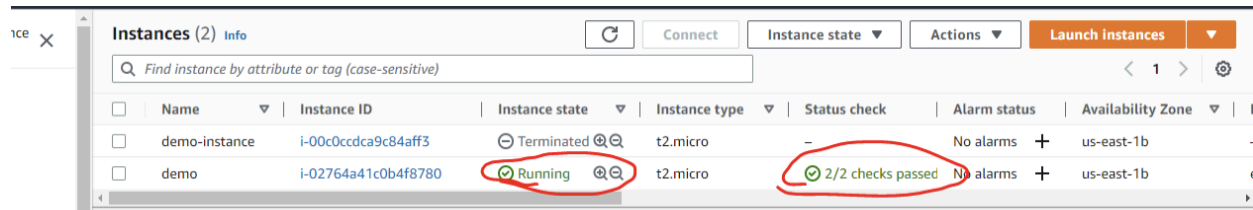
☐ **.ppk**  
For use with PuTTY

⚠ When prompted, store the private key in a secure and accessible location on your computer. **You will need it later to connect to your instance.** [Learn more](#) ↗

Cancel **Create key pair**

- 5) For network settings, you can leave everything as it is. The service will automatically create a new security group that will allow SSH traffic from any IP address
- 6) You can adjust the storage as long as it remains within the free tier
- 7) Finally, click “Launch Instance”

It will take some time for the instance to get up and running. You can view all your instances by clicking on the instances tab on the left side navigation panel.



	Name	Instance ID	Instance state	Instance type	Status check	Alarm status	Availability Zone
<input type="checkbox"/>	demo-instance	i-00c0ccdca9c84aff3	Terminated	t2.micro	-	No alarms	us-east-1b
<input type="checkbox"/>	demo	i-02764a41c0b4f8780	Running	t2.micro	2/2 checks passed	No alarms	us-east-1b

When you see that the instance is running and all the checks have passed, then it is ready to be used!

### Connecting to your instance:

- Depending on which OS you have (Windows / MacOS / etc.)
  - There are different ways to connect your instance
  - For windows, please follow this [documentation](#)
  - For other devices, please follow this [documentation](#)

### Terminate/Stopping your instance:

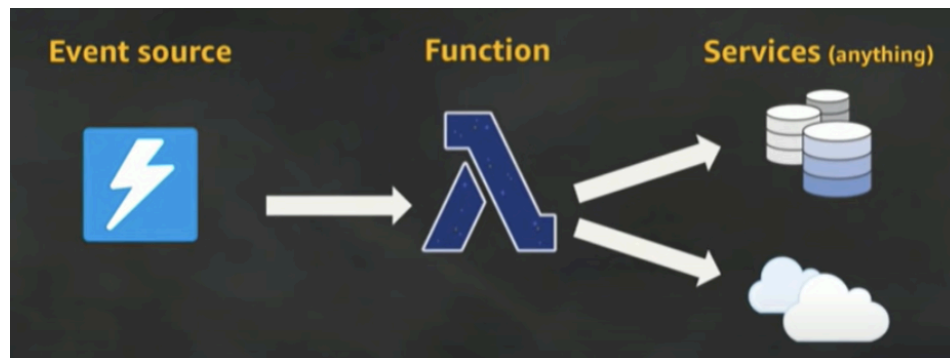
When you are done using the instance, make sure to delete it or stop it so it does not keep running and potentially cost money!

# Lambda

## What is Lambda?

- AWS Lambda is a compute service that lets you run code without provisioning or managing servers
- With Lambda, you can run code for virtually any type of application or backend service. All you need to do is supply your code in one of the languages that Lambda supports
- You organize your code into Lambda functions. Lambda runs your function only when needed and scales automatically, from a few requests per day to thousands per second

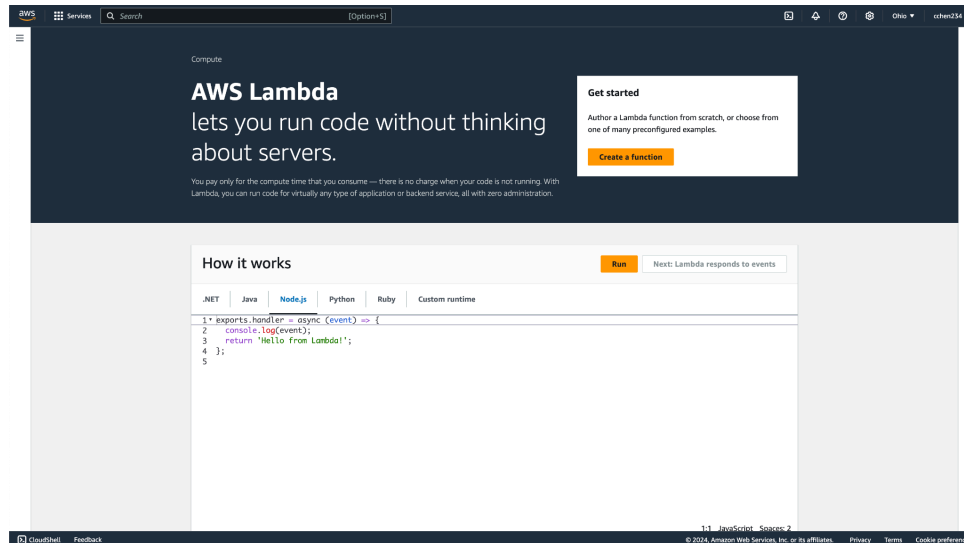
AWS Lambda works by executing functions in response to event triggers. These events can come from a variety of sources, such as a file upload to Amazon S3, an API call through API Gateway, or changes in a database. Once triggered, Lambda automatically runs the code you have specified in your function, without the need to manage or provision servers. The code execution is fully managed by AWS, scaling automatically based on the number of incoming events. After the code completes its task, Lambda returns the output to the designated destination, such as an S3 bucket, an API response, or even another AWS service. This makes Lambda ideal for automating tasks, handling real-time data processing, or building serverless applications.



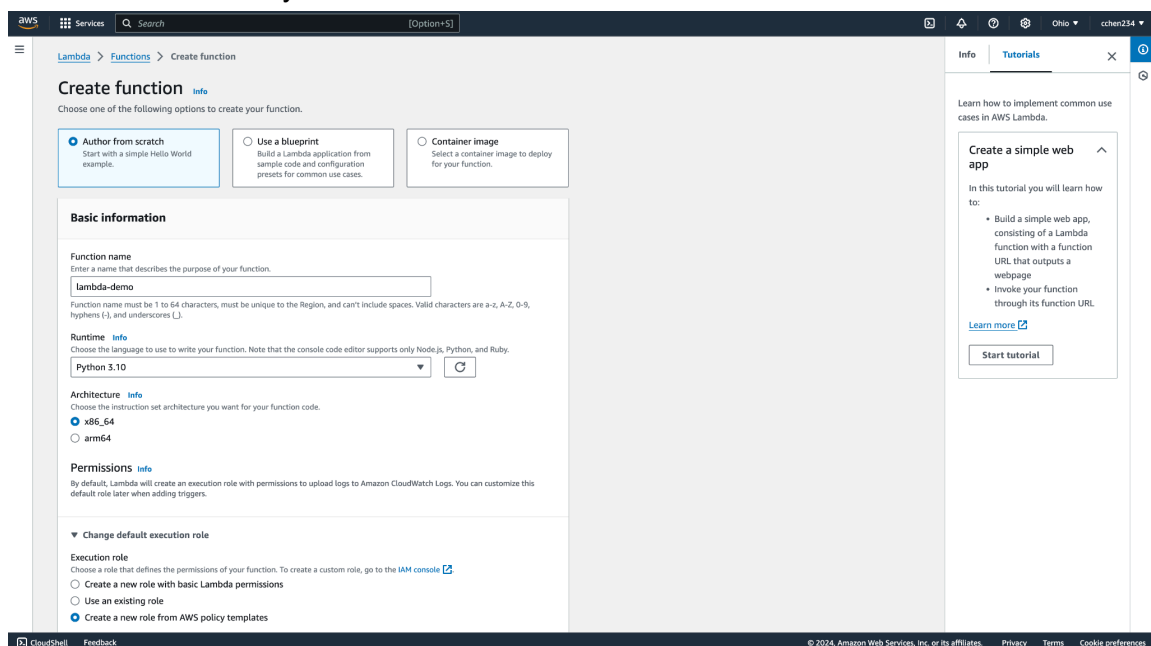
## How to use Lambda:

In this demo, we will be explaining how to trigger a Lambda function by uploading a file to an S3 bucket.

First, go to the Lambda console by searching “Lambda” in the search bar at the top. Then click on “Create a Function” on the homepage.



We will now demo how to create a custom Lambda function from scratch. Feel free to follow the tutorials that Amazon may list on the side as well!



- 1) First, give your function a name, for example, "lambda-demo"
- 2) Then, select your coding environment under Runtime. For the sake of this demo, we will select Python 3.10 .
- 3) Then, select your architecture
- 4) Since we are going to need to access a S3 bucket, we are going to need to create a new permission group

### ▼ Change default execution role

#### Execution role

Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

- ☐ Create a new role with basic Lambda permissions
- ☐ Use an existing role
- ☒ Create a new role from AWS policy templates

**i** Role creation might take a few minutes. Please do not delete the role or edit the trust or permissions policies in this role.

#### Role name

Enter a name for your new role.

myRoleName

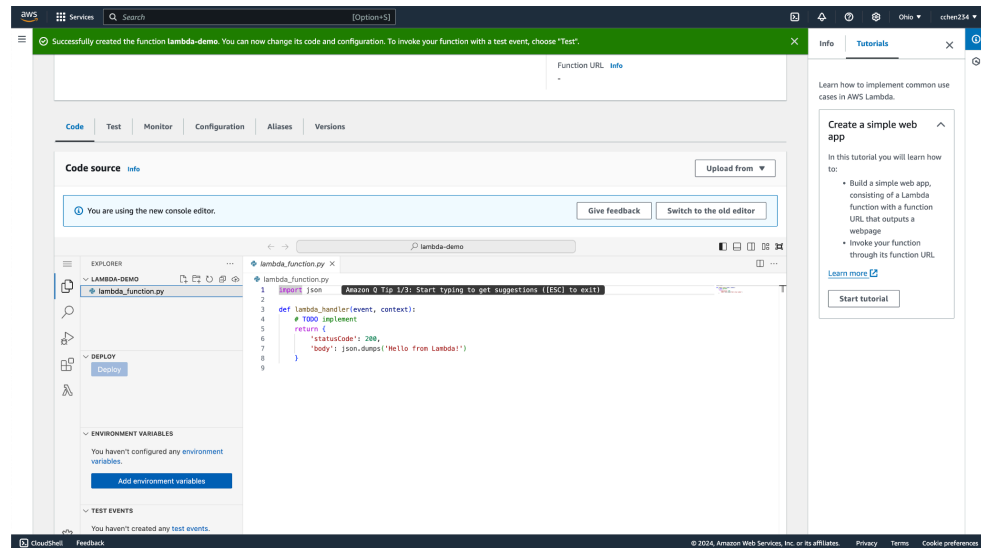
Use only letters, numbers, hyphens, or underscores with no spaces.

#### Policy templates - optional [Info](#)

Choose one or more policy templates.

Amazon S3 object read-only permissions **X**  
S3


- a) Click on the “Change default execution role” dropdown
- b) Select “Create a new role from AWS policy templates”
- c) Then, give the role a name
- d) Select “Amazon S3 object read-only permissions”
- e) Press create function
- 5) You will be brought to a page that looks like this



- 6) To add the S3 trigger, press the “add trigger” button

- 7) You will be brought to a separate page
- Choose "S3" as the source event
  - Select a bucket (if you do not have a bucket, please create one)
  - Select the event type "POST"
  - You can also add prefixes if you have a subfolder within your bucket or suffixes if you want to specify file types that trigger the lambda function
  - Press "Create"

### Trigger configuration [Info](#)

 **S3**  
aws storage

**Bucket**  
Please select the S3 bucket that serves as the event source. The bucket must be in the same region as the function.

✕ ↺



Bucket region: us-east-1


**Event type**  
Select the events that you want to have trigger the Lambda function. You can optionally set up a prefix or suffix for an event. However, for each bucket, individual events cannot have multiple configurations with overlapping prefixes or suffixes that could match the same object key.

**Prefix - optional**  
Enter a single optional prefix to limit the notifications to objects with keys that start with matching characters.

✔ The trigger lambda-s3-hab-demo was successfully added to function lambda-demo. The function is now receiving events from the trigger.

### ▼ Function overview [Info](#)


 **lambda-demo**  
 Layers (0)

 **S3**  
+ Add trigger

+ Add destination

**Description**  
-

**Last modified**  
9 minutes ago

**Function ARN**  
 arn:aws:lambda:us-east-1:7188517524:lambda-demo

**Function URL** [Info](#)  
-

- 8) Before we go on, it is important to understand what an S3 event actually looks like. Below is the information that gets sent to Lambda when an S3 event is processed. The

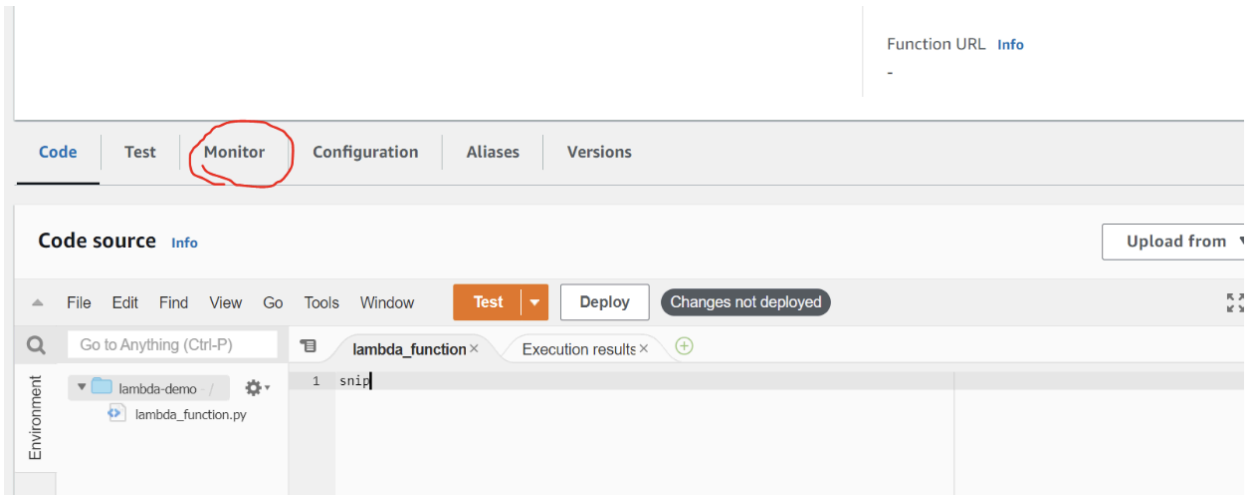
fields that will be important to us are the bucket name and key. The bucket name will be the bucket that has the uploaded file and the key will actually be the name of the uploaded file.

```
"Records": [
  {
    "eventVersion": "2.0",
    "eventSource": "aws:s3",
    "awsRegion": "us-east-1",
    "eventTime": "1970-01-01T00:00:00.000Z",
    "eventName": "ObjectCreated:Put",
    "userIdentity": {
      "principalId": "EXAMPLE"
    },
    "requestParameters": {
      "sourceIPAddress": "127.0.0.1"
    },
    "responseElements": {
      "x-amz-request-id": "EXAMPLE123456789",
      "x-amz-id-2":
"EXAMPLE123/5678abcdefghijklmbdaisawesome/mnopqrstuvwxyzABCDEFGH"
    },
    "s3": {
      "s3SchemaVersion": "1.0",
      "configurationId": "testConfigRule",
      "bucket": {
        "name": "example-bucket",
        "ownerIdentity": {
          "principalId": "EXAMPLE"
        },
        "arn": "arn:aws:s3:::example-bucket"
      },
      "object": {
        "key": "test%2Fkey",
        "size": 1024,
        "eTag": "0123456789abcdef0123456789abcdef",
        "sequencer": "0A1B2C3D4E5F678901"
      }
    }
  }
]
```

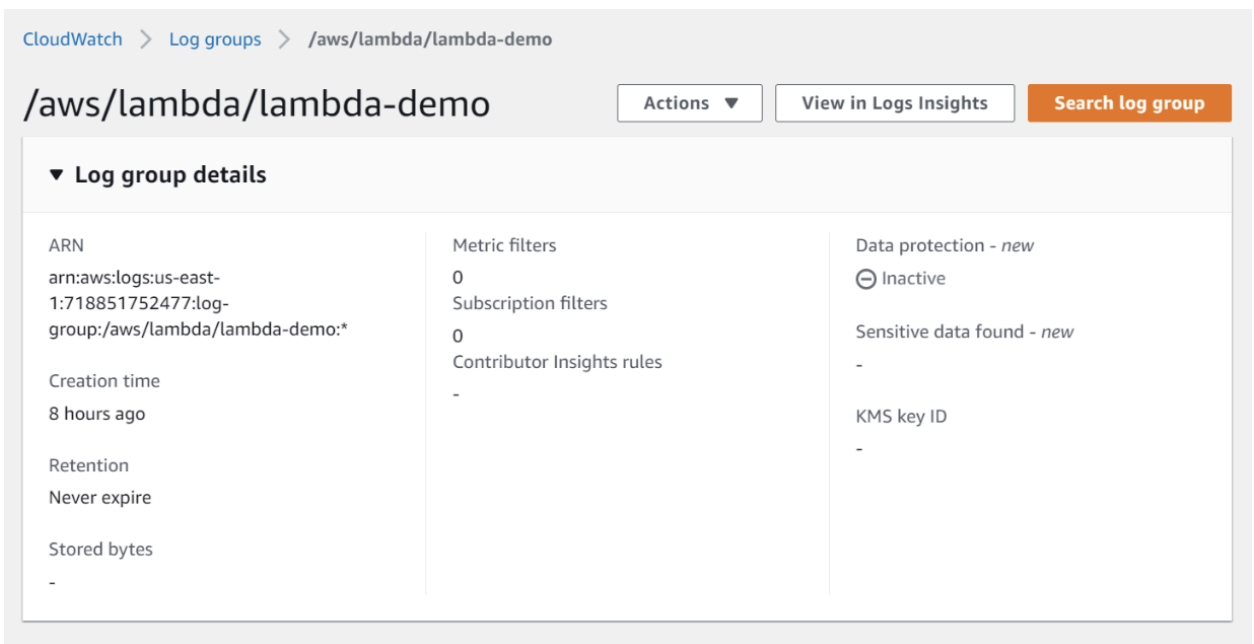
- 9) For the code section, here is the Python code for processing the csv file that will be uploaded
- 10) Next, here is a link to the csv file to download



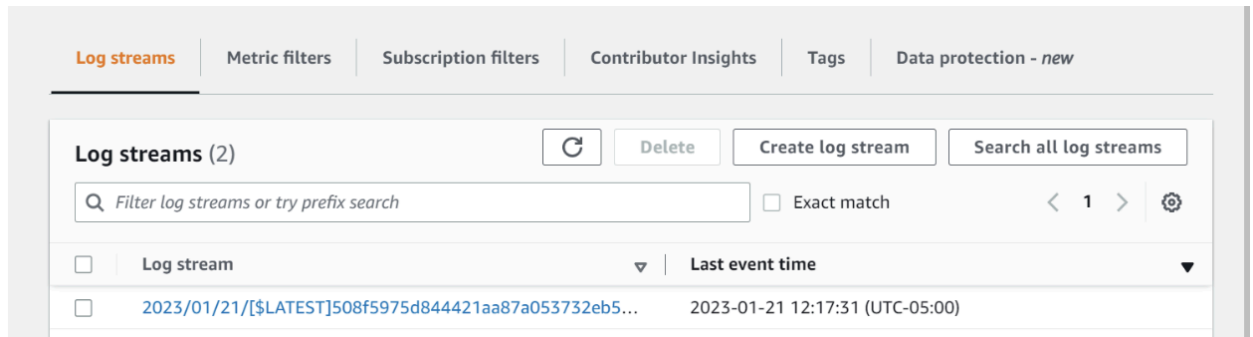
- a) After downloading the link, go to the S3 using the search bar and upload the file into the bucket that you selected for the trigger
- 11) Then, after you uploaded the file go back to Lambda, and at the bottom of the page, you should see a monitor tab



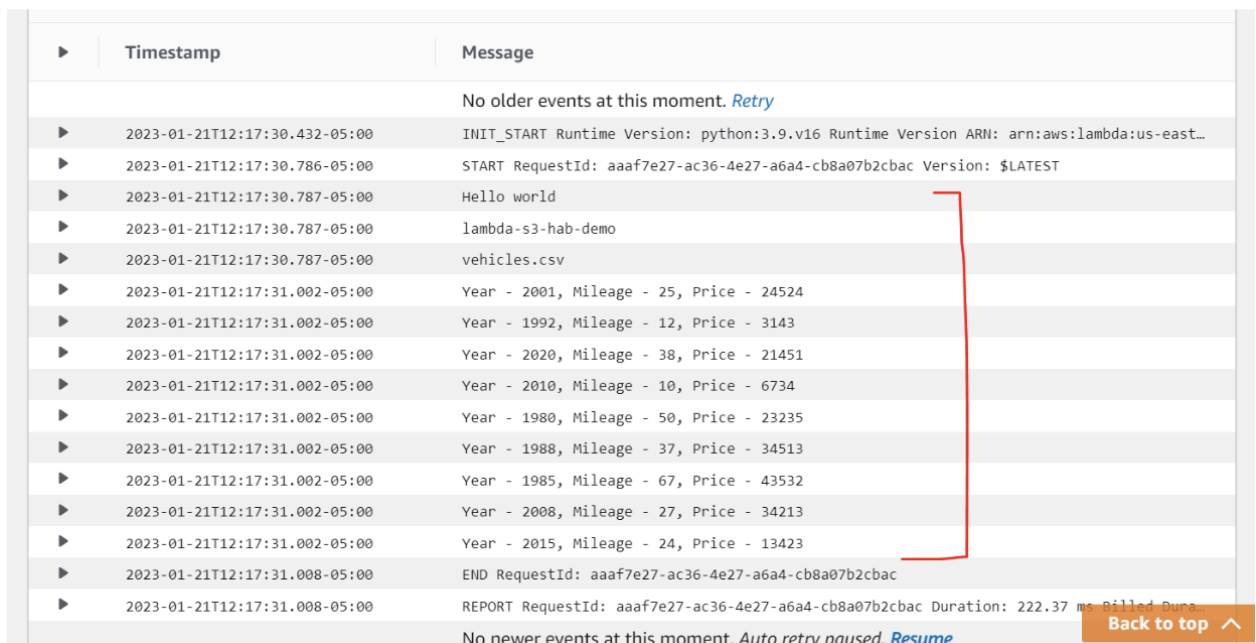
- 12) Click on this tab and then click on the button that says “View Cloudwatch Logs”
- 13) It will take you to a page that looks like this



- 14) At the very bottom of the page, you will see a log streams tab. Click on the most recent log stream which should have been triggered by the S3 event



- 15) You should see something like this in the logs, where the output of the data processing is printed in the logs, meaning that the Lambda function was triggered and our data has been processed



Congrats! You finished our demo.

Please reach out to the Hack@Brown team if you have any questions on the content in this starter kit!