AWS Storage Compression

Using AWS S3, Lambda, DynamoDB, and EC2

CS 7125

Fall

By: Dave Boggs

Professor: Dr. Shi

Contents

[Introduction 1](#_Toc57760991)

[Service Summaries 2](#_Toc57760992)

[S3 2](#_Toc57760993)

[EC2 2](#_Toc57760994)

[DynamoDB 2](#_Toc57760995)

[Lambda 3](#_Toc57760996)

[Application Workflow 3](#_Toc57760997)

[Installation Instructions 4](#_Toc57760998)

[Source Code 4](#_Toc57760999)

[Create an EC2 Instance 4](#_Toc57761000)

[Create an S3 Bucket 8](#_Toc57761001)

[Create a DynamoDB Table 9](#_Toc57761002)

[Create Lambda Function 10](#_Toc57761003)

[Using the Application 20](#_Toc57761004)

# Introduction

S3compression is an application used to minimize AWS S3 storage utilization with the intent of keeping storage fees down. The primary purpose of this project is to demonstrate some of the features of certain AWS services as well as how to leverage them. This project utilizes the S3, EC2, DynamoDB, and Lambda services.

# Service Summaries

## S3

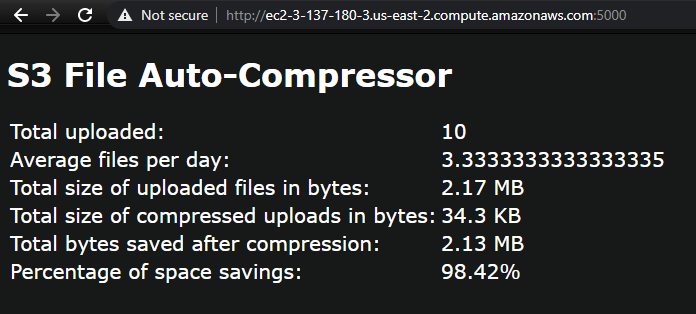
S3 (Simple Storage Service) is a serverless object storage solution. For each AWS account multiple buckets can be created. Buckets are collections of objects, where objects are the files stored in the buckets. Files of any type can be stored in buckets, and they can be up to 5 TB in size. Bucket contents can be managed (uploaded, downloaded, deleted, etc.) from the AWS Management Console or through various APIs.

This project interacts with S3 through the python boto3 library.

## EC2

EC2 (Elastic Compute Cloud) allows for the creation and management of cloud server instances. Cloud instances can have a flexible amount and variety of resources allocated.

This project can leverage an EC2 instance for serving the application statistics site.



## DynamoDB

DynamoDB is a NoSQL Key-Value database. The rows in the database do not all need to have the same dimensions or fields. Each row can contain different types of data including files, images, code, and more. The strength of DynamoDB is the query response time which is usually less than 10ms per request for reads. The use cases that are best for this sort of database paradigm games and other applications that have many read requests that require a fast response time.

This project uses DynamoDB to store records of compression tasks that can be consumed by s3Compression’s statistics site.

## Lambda

AWS Lambda runs code in response to events that occur in many AWS services. Triggers for various events can be tracked, and when the correct conditions are met Lambda will run user-defined code known as lambda functions. Lambda functions can be written in most popular programming languages. Lambda provides the benefit of not managing a server instance in order to run an app. When it is time to run a lambda function it is automatically assigned to an Amazon-managed server instance to execute.

Lambda provides the core workflow functionality for this application.

# Application Workflow

Lambda monitors for object creation events on S3. By default, the application only monitors events for one bucket, and the name of the bucket must be defined in the lambda function. When an object creation event is triggered Lambda calls a specific method defined in the lambda function. The Lambda function will determine which file was uploaded, and decide if it needs to be compressed. By default, all files are compressed (into a zip file) except for files with a .zip extension. The file compression takes place in the memory space of the lambda function. Once the compression is complete a zip file is saved to the S3 bucket, and the previously-uploaded file is deleted.

For each compression task the lambda function creates a new row in a DynamoDB table. The row tracks the file name, the date of the task, the original object size before compression, and the compressed object size.

An EC2 instance runs an app server that pulls information from the DynamoDB table and generates statistics. The statistics can be viewed through a web browser by accessing a specific URL (served by the EC2 instance).

# Installation Instructions

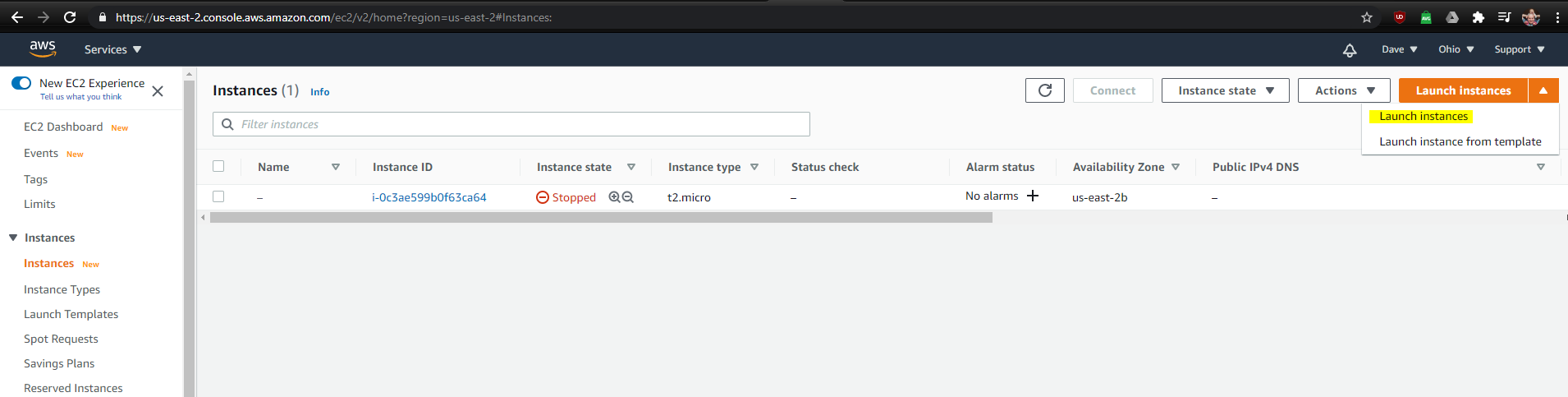
## Source Code

All source code is available from the project’s GitHub repository:

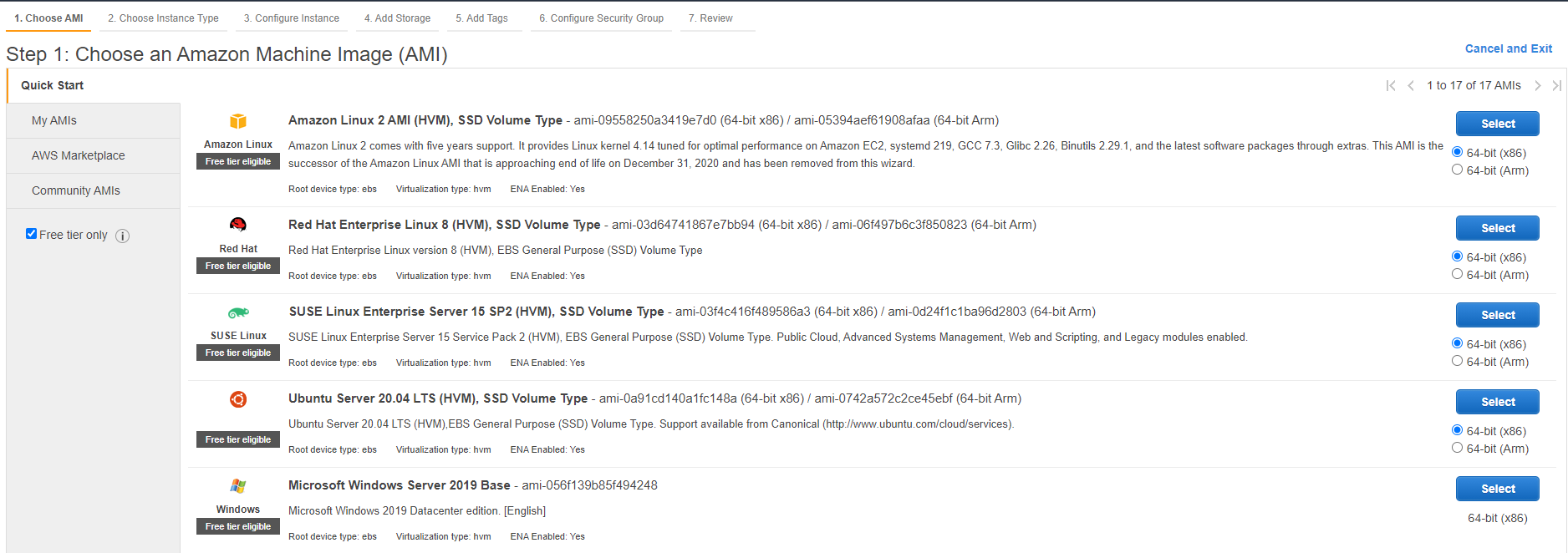
<https://github.com/dboggs0/s3Compression.git>

## Create an EC2 Instance

From the EC2 Management Console select Launch Instances, and then Launch Instances on the drop-down.

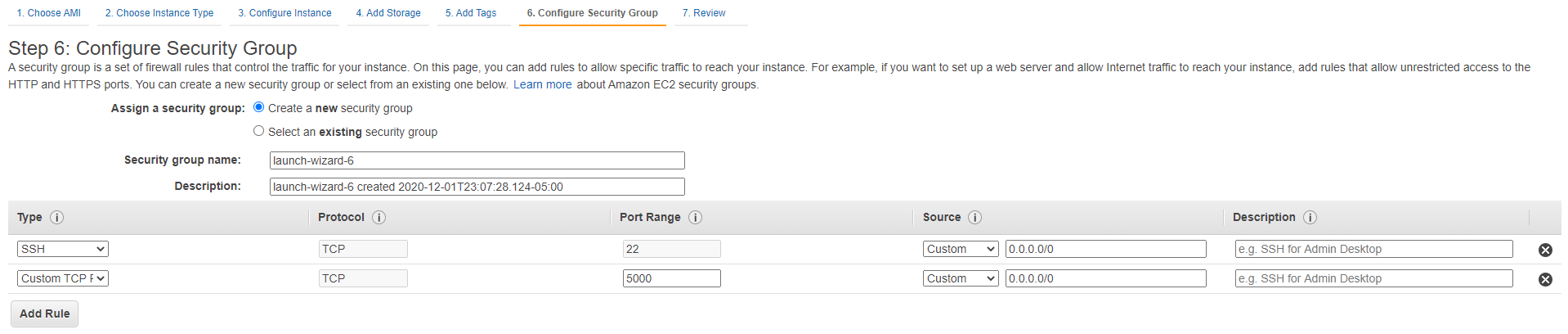


Choose an AMI. Any free-tier-eligible Linux or Windows AMI should work, but the initial development was done on Ubuntu Server 20.04.



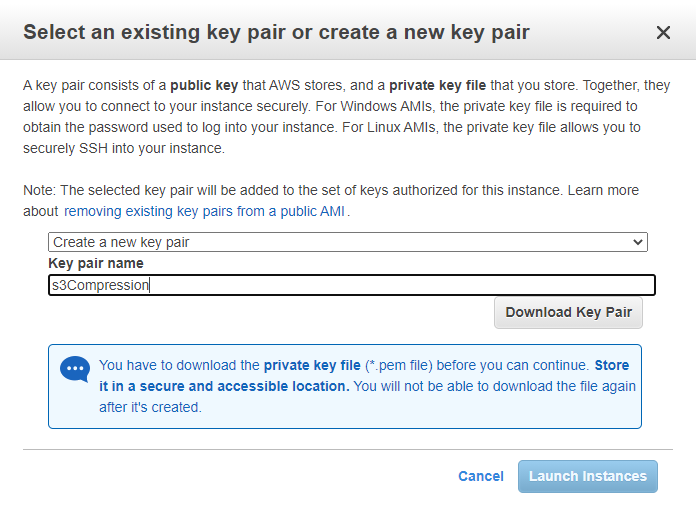
Select the instance type. The only free-tier-eligible instance type is t2.micro, which is suitable to run this application.

Configure Security Group. Select the port over which the statistics site will be accessed. By default, the application will run on TCP port 5000. Setting the source field to “0.0.0.0/0” opens access to any WAN-connected host.



Click Review and Launch, then Launch.

Generate and download a key pair.



Full instructions on how to access the new instance using the key pair are available at the following link:

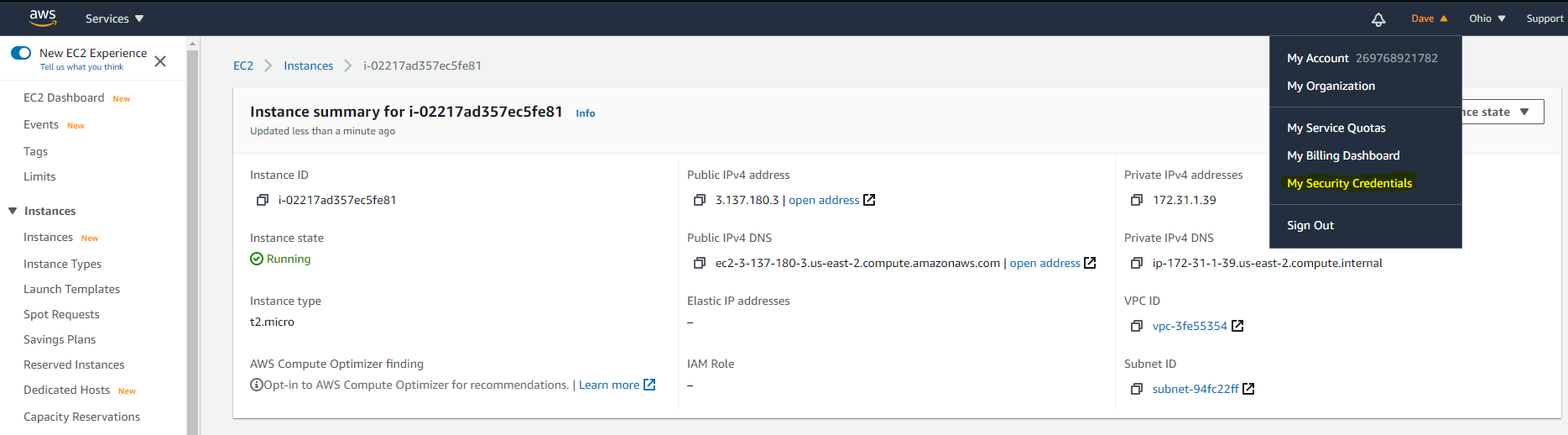
<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-key-pairs.html>

Upgrade the OS packages

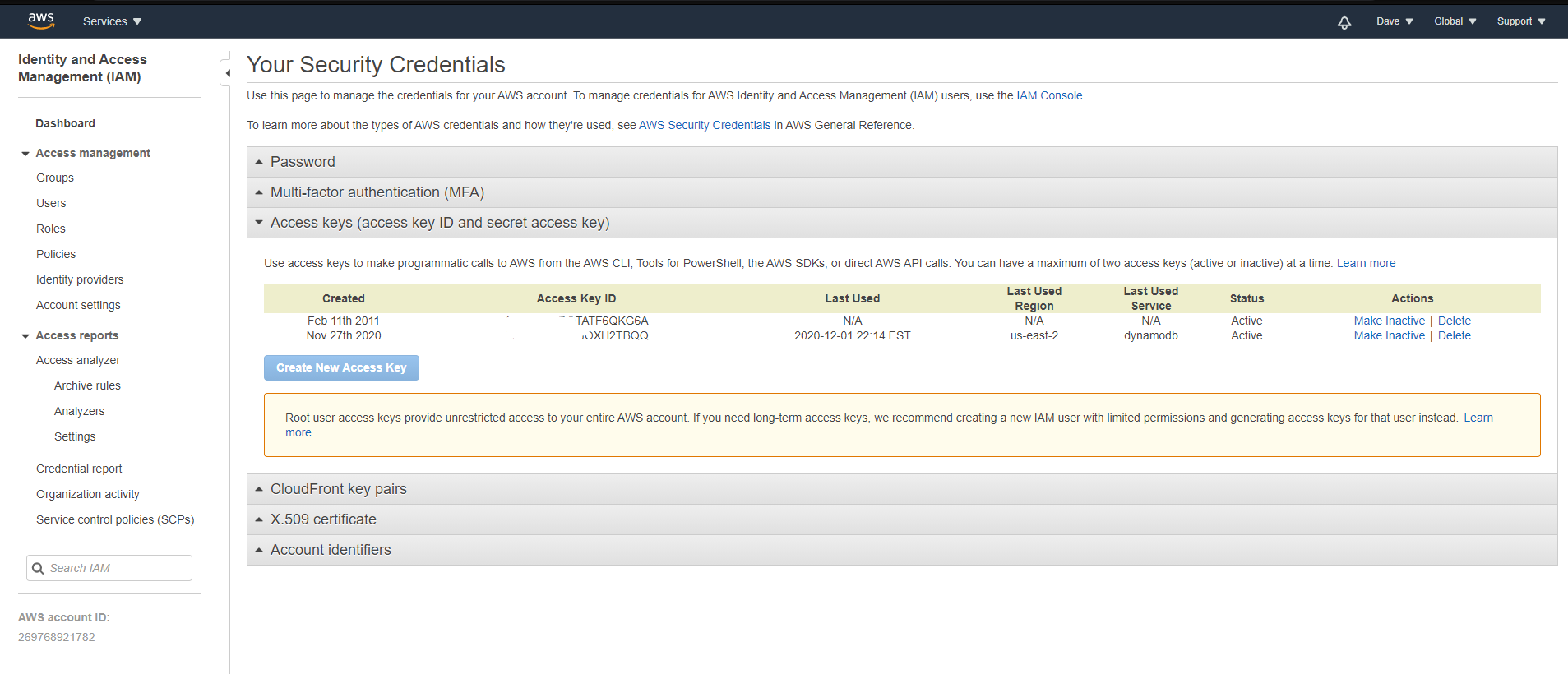
*$ sudo apt update*

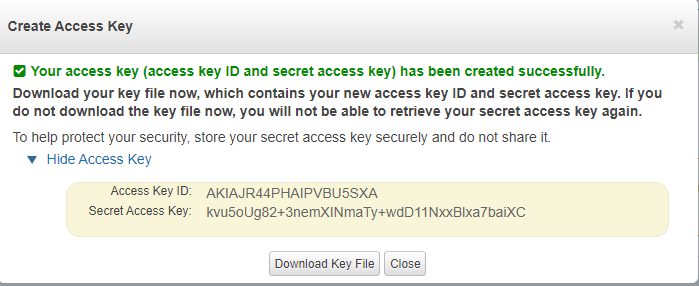
*$ sudo apt upgrade*

This process will take a few minutes. While it progresses create an AWS Security Credential.



Click Create New Access Key





Download the key file. Once the key file is downloaded it cannot be downloaded a second time. Make sure to keep the downloaded file somewhere safe.

Make note of the Access Key ID, and Secret Access Key.

Go back to the EC2 Instance Console.

Make sure that zip is installed. If not, it can be installed with the following command (Ubuntu):

*$ apt install zip*

Install the AWS CLI.

<https://docs.aws.amazon.com/cli/latest/userguide/install-cliv2-linux.html#cliv2-linux-install>

*$ curl "https://awscli.amazonaws.com/awscli-exe-linux-x86\_64-2.0.30.zip" -o "awscliv2.zip"*

*$ unzip awscliv2.zip*

*$ sudo ./aws/install*

Add your access key to the AWS CLI.

*$ aws configure*

*AWS Access Key ID [None]: \*\*\*…\*TBQQ*

*AWS Secret Access Key [None]: \*\*\*…\*K0a5A1*

*Default region name [None]: us-east-2*

*Default output format [None]:*

For a list of region codes refer here:

<https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-regions-availability-zones.html#concepts-available-regions>

Clone the GitHub repository.

*$ git clone https://github.com/dboggs0/s3Compression.git*

Install Python 3.6 or higher.

*$ sudo apt install python*

Install virtualenv.

*$ sudo apt install virtualenv*

Create a virtual environment for the project.

*$ virtualenv s3Compression*

Activate the virtual environment.

*$ cd s3Compression/bin*

*$ . activate*

Install the project dependencies.

*$ cd ..*

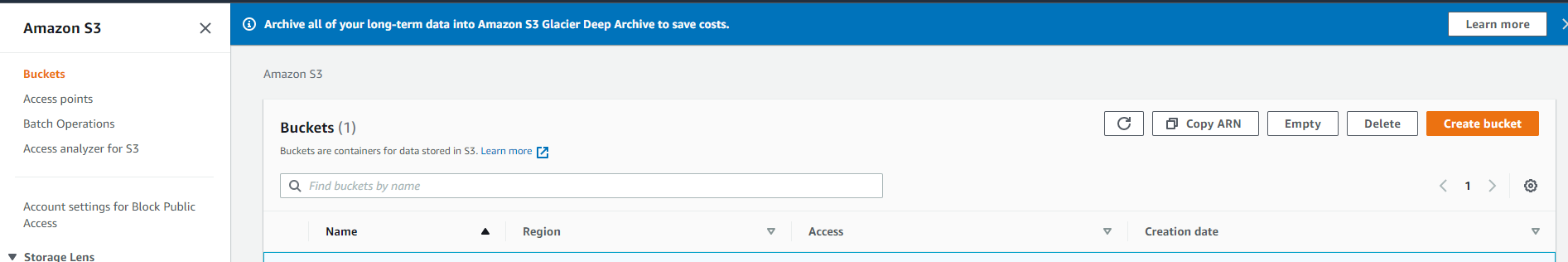
*$ pip install -t requirements.txt*

The Statistics Site is now installed. It will be started later.

## Create an S3 Bucket

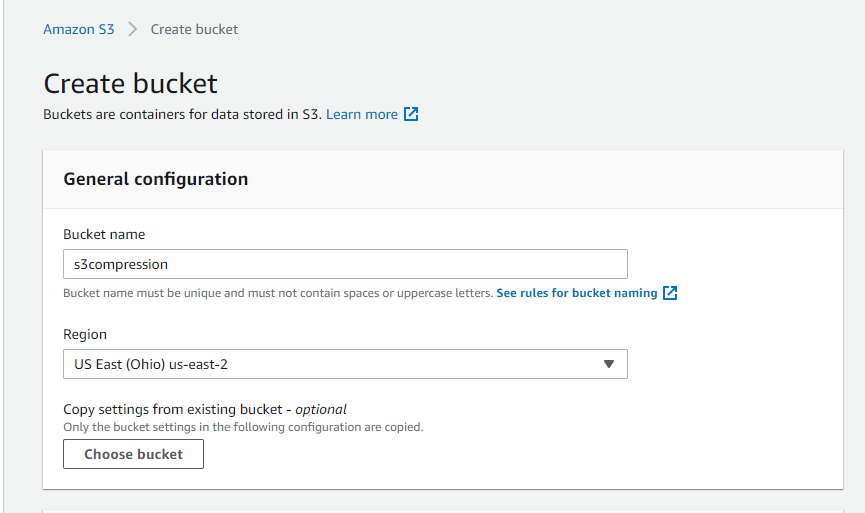
Go to S3 in the AWS Management Console.

Click “Create Bucket”



Name the bucket.

Make sure the Region is the same as the one for the EC2 instance.

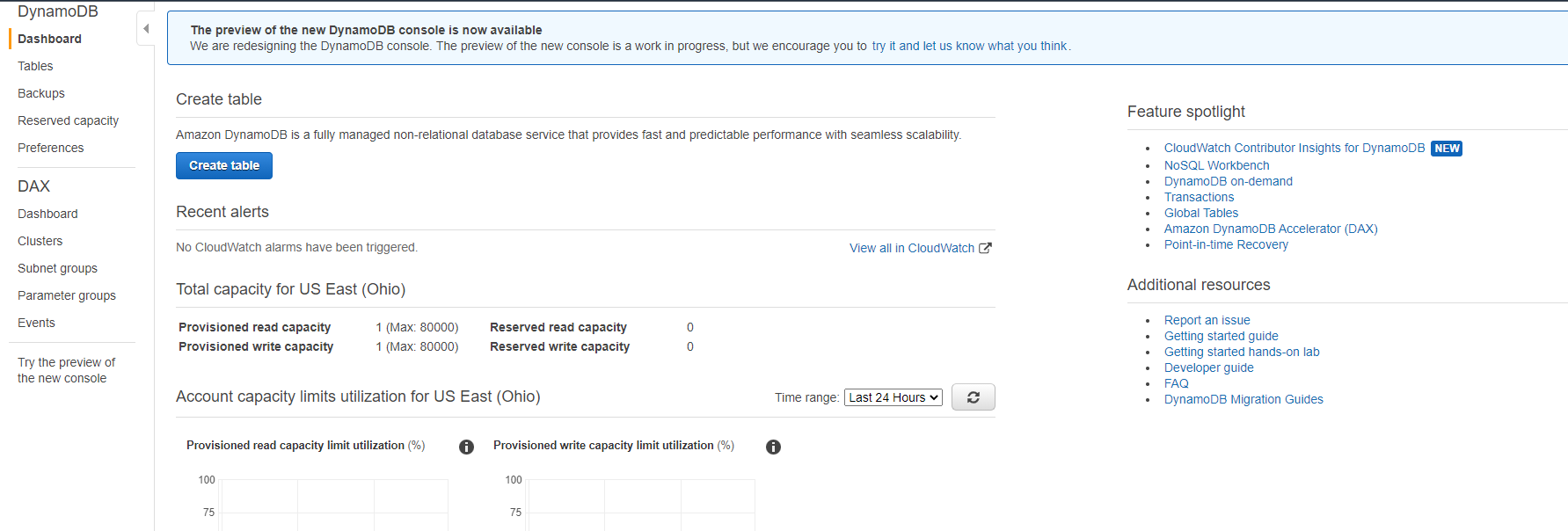


The default name for the bucket in this project is s3compression. If you want to use a different one the code will need to be updated.

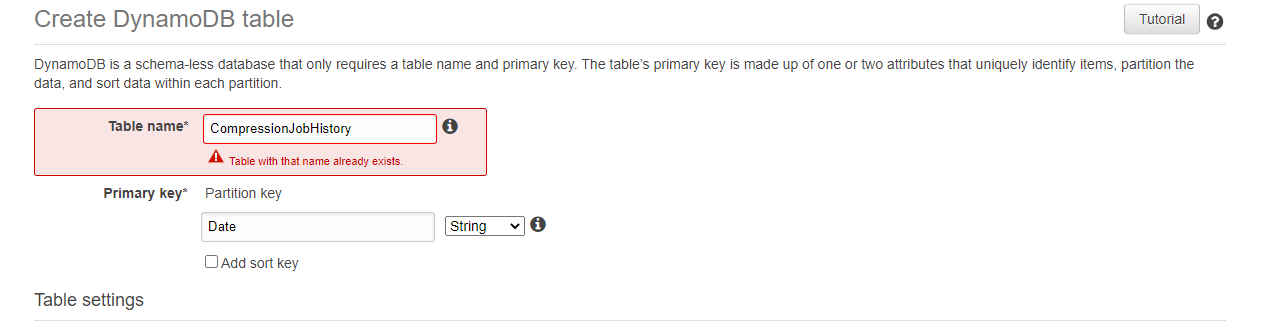
## Create a DynamoDB Table

Go to DynamoDB in the AWS Management Console.

Click “Create table”



The default table name for this project is CompressionJobHistory. To use the default table name fill out the form as follows:



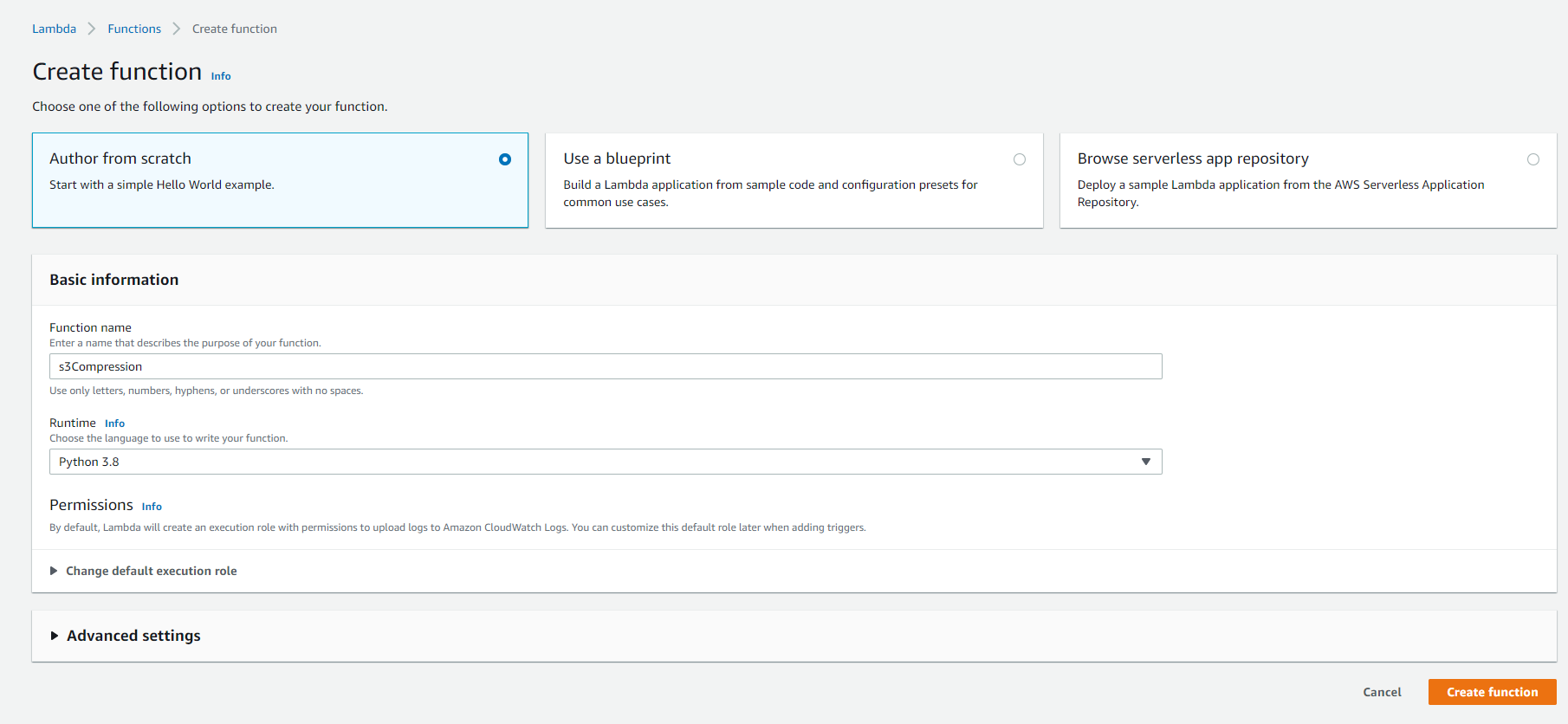
In order to use a different table name the project code must be modified.

## Create Lambda Function

Go to Lambda in the AWS Management Console.

Click “Create function”

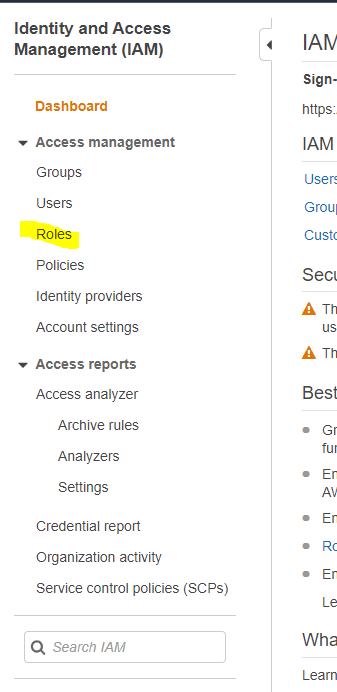
Select “Author from scratch”. The name of the function does not matter. Select the Python 3.8 runtime.



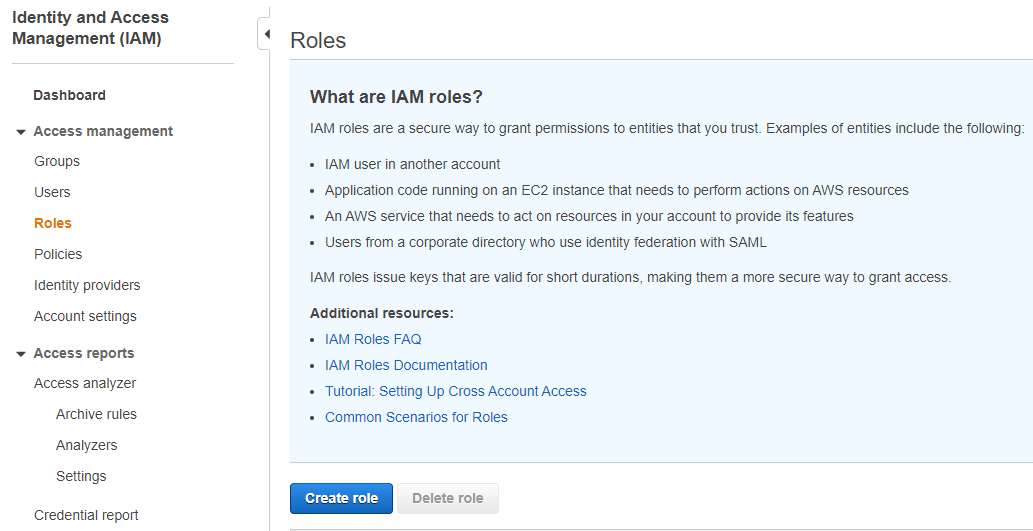
The lamba function configuration is not yet complete, but before proceeding further permissions must be added to the lambda function.

Go to IAM in the AWS Management Console.

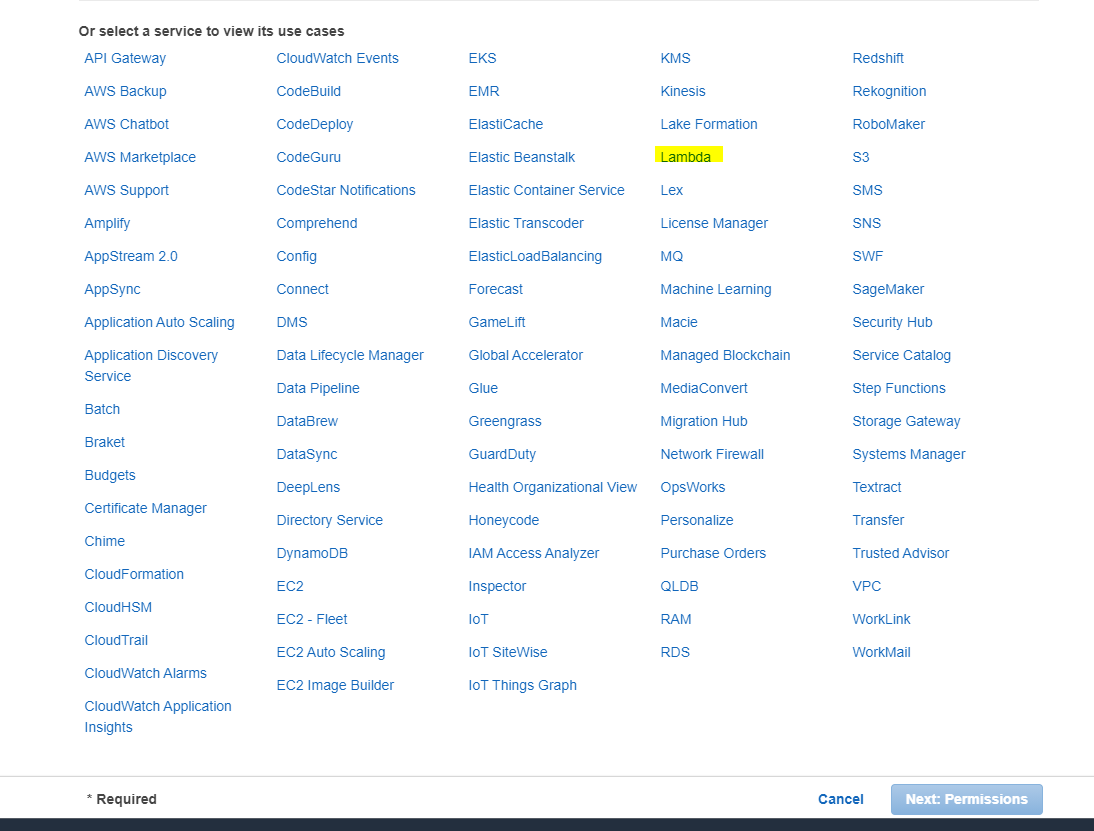
Select Roles on the navigation tree.



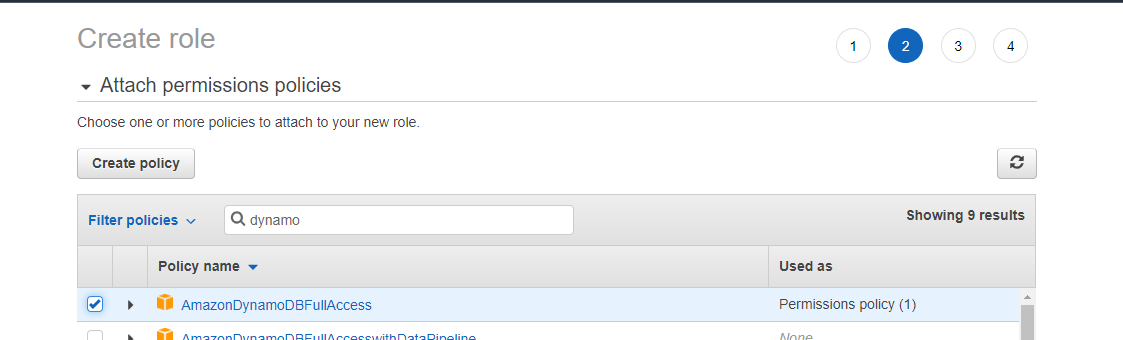
Click “Create role”.



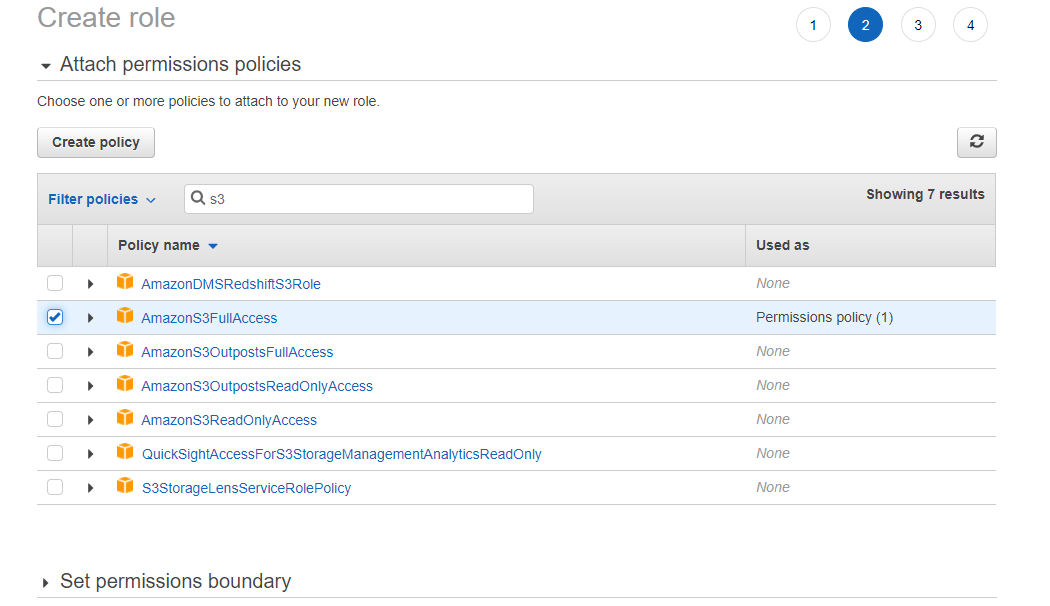
Select Lambda from the list of services.



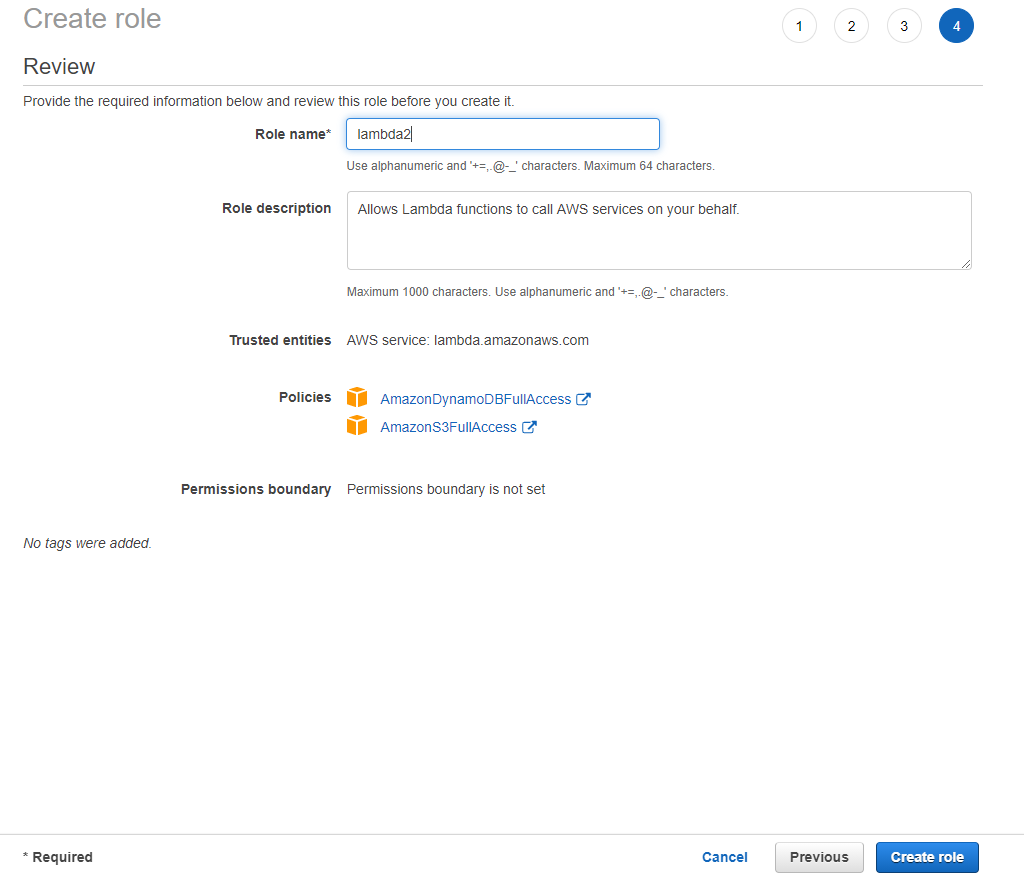
Add AmazonDynamoDBFullAccess



Add AmazonS3FullAccess



Proceed through the wizard until the Review page. Name the role. The name does not matter

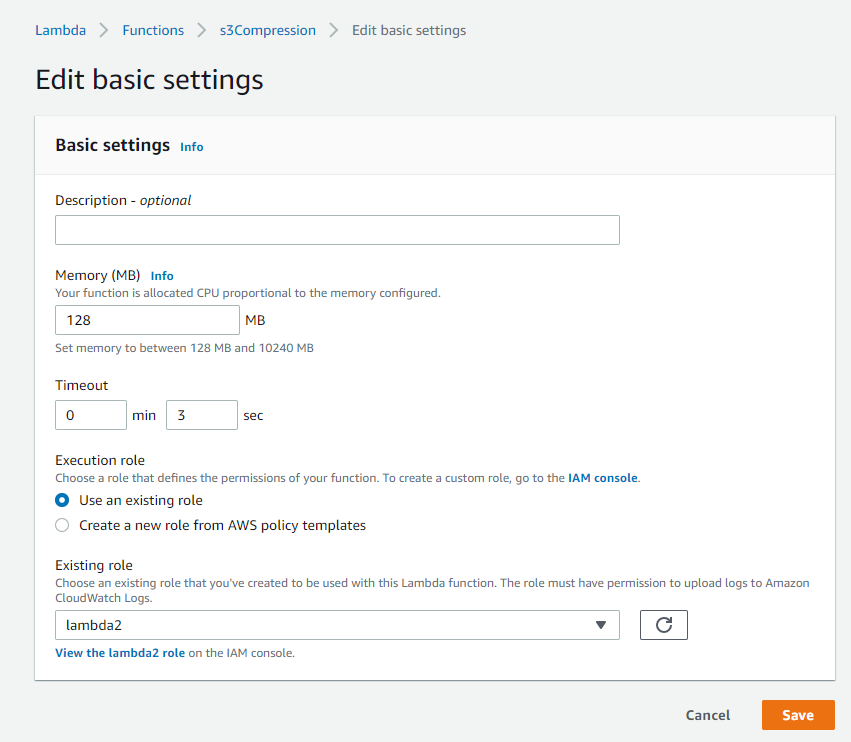


Go to Lambda in the AWS Management Console.

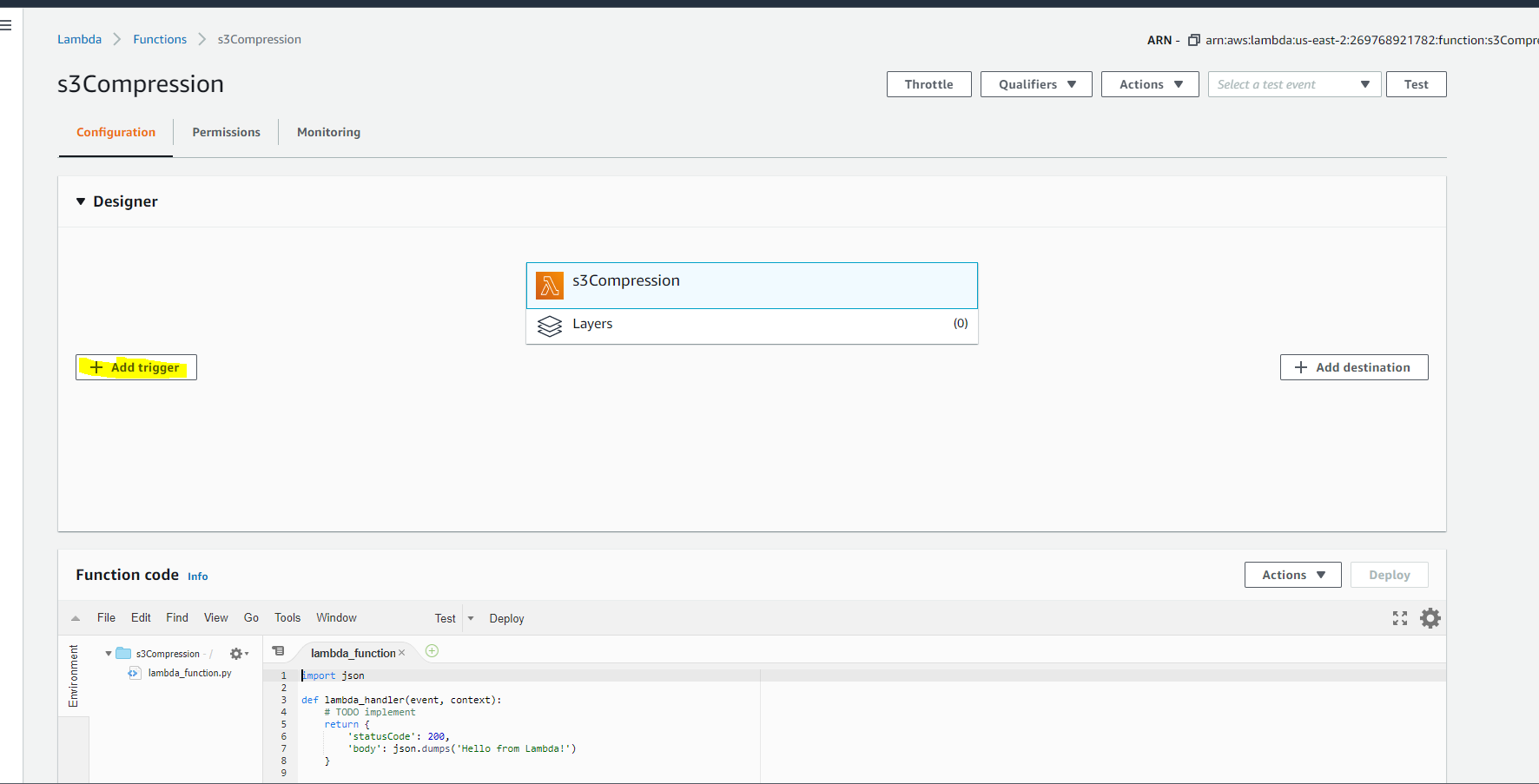
Open your lamda function.

Select Permissions.

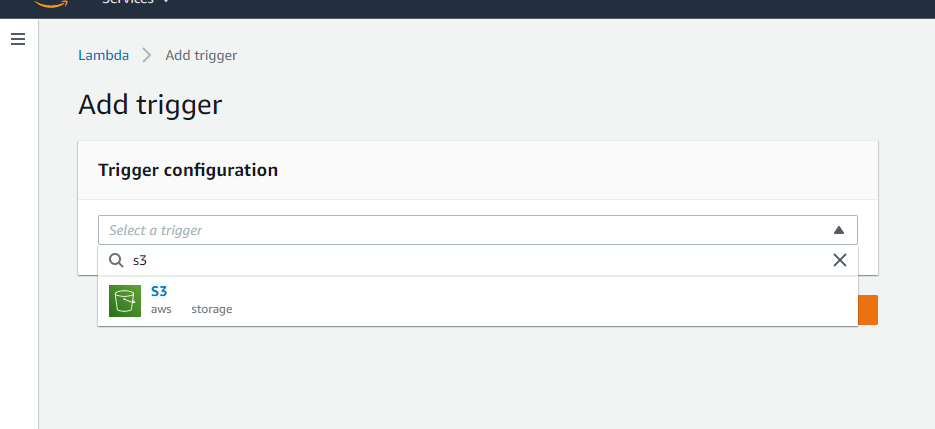
Select the security role that was created in IAM.



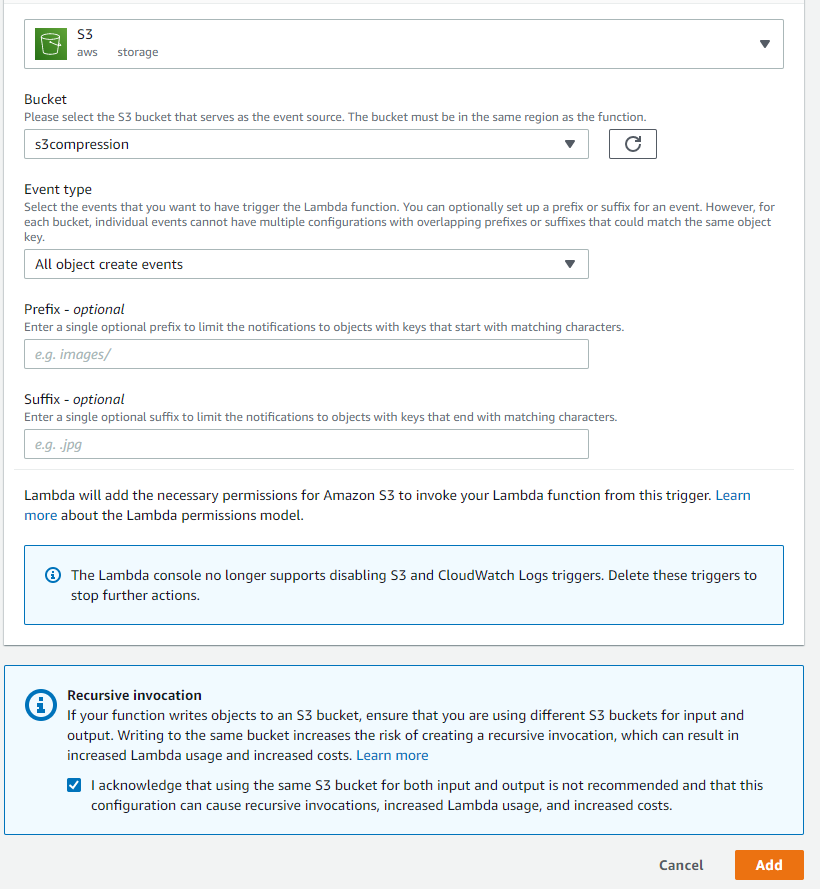
Click “Add trigger”



Select S3.

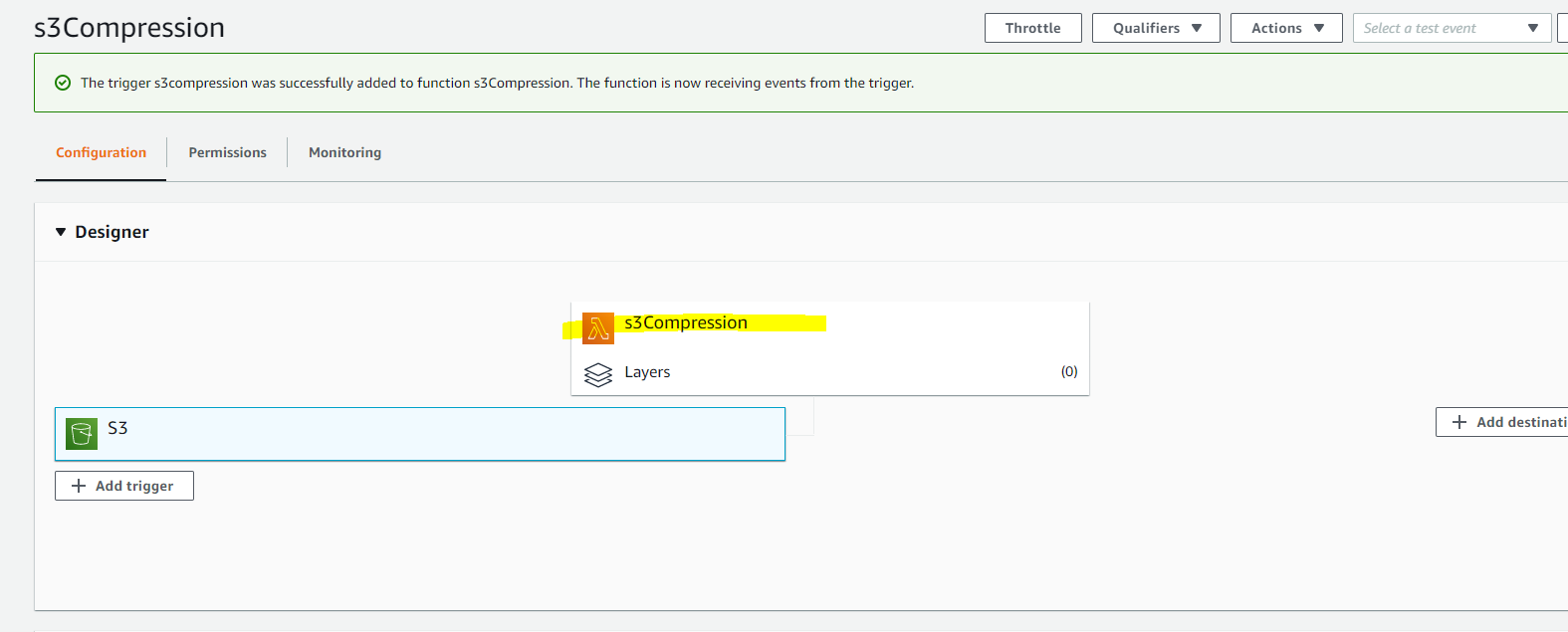


Fill out the trigger configuration as follows:



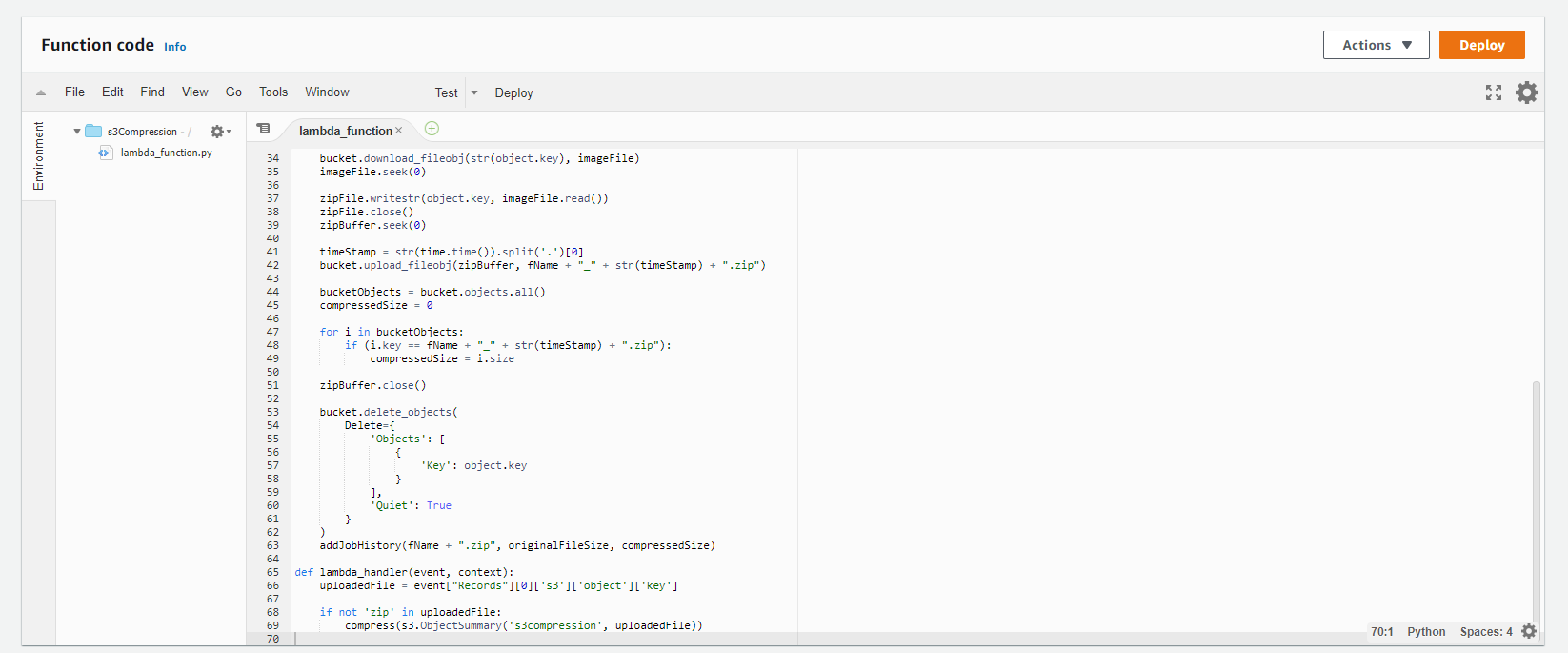
Make sure to check the acknowledgement at the bottom. Be careful when modifying the project code. Recursive invocations can cost a lot of money very quickly.

Click on the Lambda symbol in the Designer.



Open the lambdaFunction.py in a text editor to copy the code. It is in the src directory of the project.

Copy the code and past it into the Function code section of the page.



Click Deploy.

At this point the statistic site can be started.

On the EC2 Instance run the following commands.

*$ export FLASK\_APP=”routes.py”*

*$ flask run –host=0.0.0.0*

The installation is complete.

# Using the Application

Upload any file to S3. As long as it is not a zip file the Lambda function will trigger and compress it.

To access the statistics site, get the external IP or DNS name of the EC2 instance, put it in the address bar of a web browser and add “:5000” to the end. The site does not auto-refresh. Refresh the browser page periodically for more current statistics.