**Serverless Data Pipeline**

**Creating S3 Buckets**

To create a serverless data pipeline using AWS Lambda, S3, and IAM, the process starts by configuring two S3 buckets: one for the source data (**data-source-bucket-1),** where raw CSV files are uploaded, and one for the destination (**target-destination-bucket-1**), where the aggregated results will be stored.

A screenshot of a computer

Description automatically generated

**Creating IAM Roles**

After creating the S3 buckets, an IAM (Identity and Access Management) role is created. The role called **pipeline-role** is configured with the necessary permissions to allow the Lambda function to read from the source bucket, write to the destination bucket, and log processing events to CloudWatch.

A screenshot of a computer

Description automatically generated

**Lambda Function**

Once the IAM role is set up, the Lambda function called **mylambdafunction** is created. The function is written in Python and leverages the boto3 library to interact with AWS S3. The function is designed to trigger whenever a new CSV file is uploaded to the source bucket. This is accomplished by configuring an event notification on the source bucket that triggers the Lambda function upon file upload. The Lambda function is responsible for reading the incoming CSV file, processing the data, aggregating salary information by location (i.e., summing the salary for each unique location), and then writing the results to the destination bucket. The file is processed using Python’s built-in csv module, which allows the function to read and write CSV data easily. Once all the lambda functions and triggers are created, a python code is written and then deployed. Hence, once the code is deployed, every csv file uploaded on the source bucket is going to trigger a lambda function which will then read the file and the aggregates the data and then stored to the destination file.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

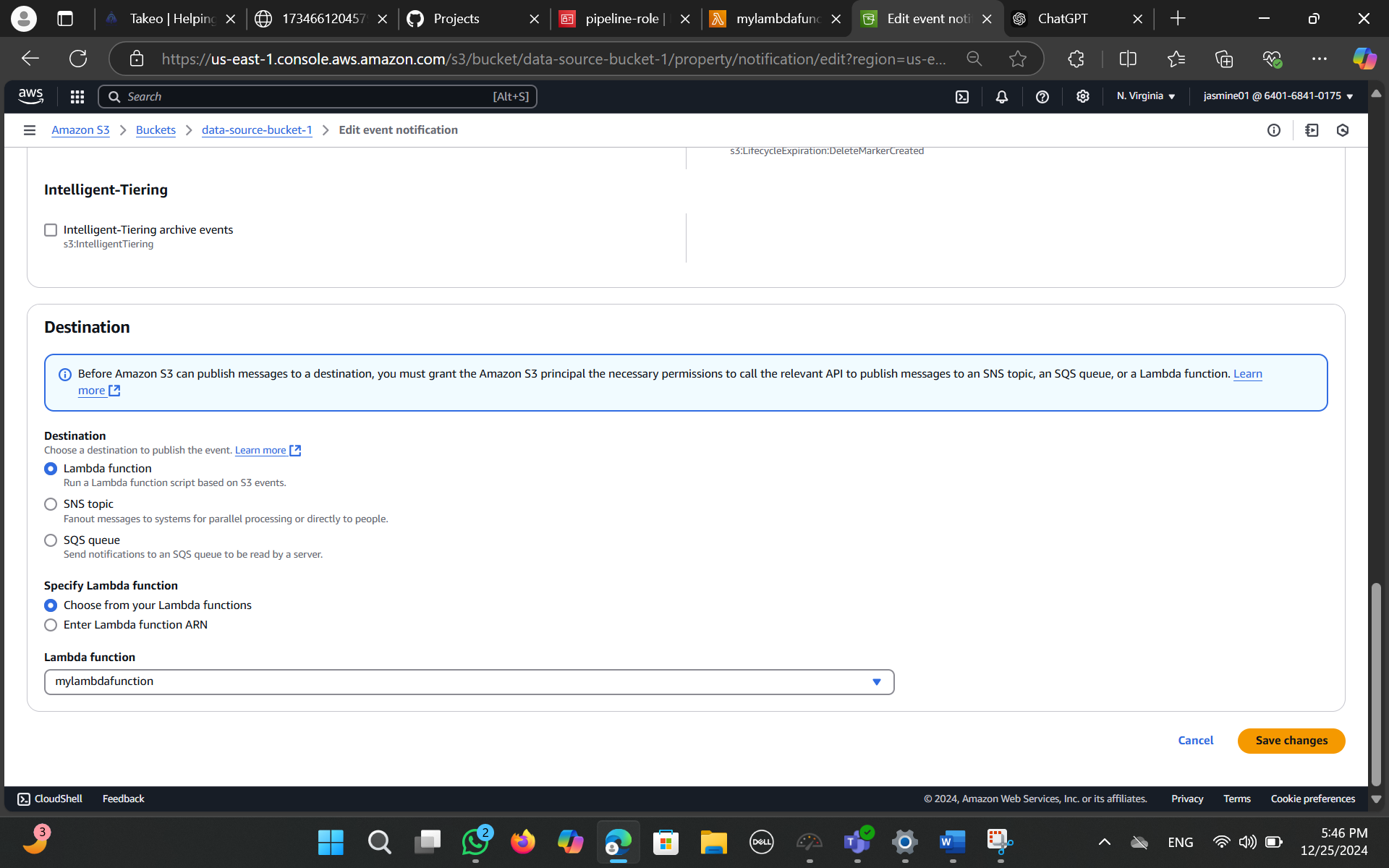
Description automatically generated

**Configuring Event Notification on S3 bucket**

After configuring the Lambda function, the next step is to set up the event notification on the source S3 bucket. This ensures that every time a new file is uploaded to the bucket, the Lambda function is triggered. The function processes the uploaded file, aggregates the salary data, and writes the output to the destination bucket. The function logs key details of the process, such as the file being processed and any errors encountered, to CloudWatch for monitoring purposes. This makes it easy to track the success or failure of the pipeline.

A screenshot of a computer

Description automatically generated



**Uploading a .csv file**

The final step involves testing the solution to ensure that the entire pipeline works as expected. To do this, a CSV file is uploaded to the source bucket. The Lambda function is automatically triggered, reads the file, performs the aggregation of salaries by location, and stores the output in the destination bucket. Once the process is complete, the results can be verified by checking the destination bucket for the aggregated CSV file. If any errors occur during processing, they will be captured in the CloudWatch logs, allowing for quick debugging and resolution.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

In summary, this serverless data pipeline relies on AWS Lambda for processing, S3 for storage, and IAM for security. By leveraging these AWS services, the pipeline automates the process of aggregating employee salary data in real time, storing the results in a destination S3 bucket for further use.