### **Step 1: Set Up the Server**

#### **1. Choose a Cloud Provider**

In this project i used AWS cloud provider

#### **2. Launched an EC2 Instance**

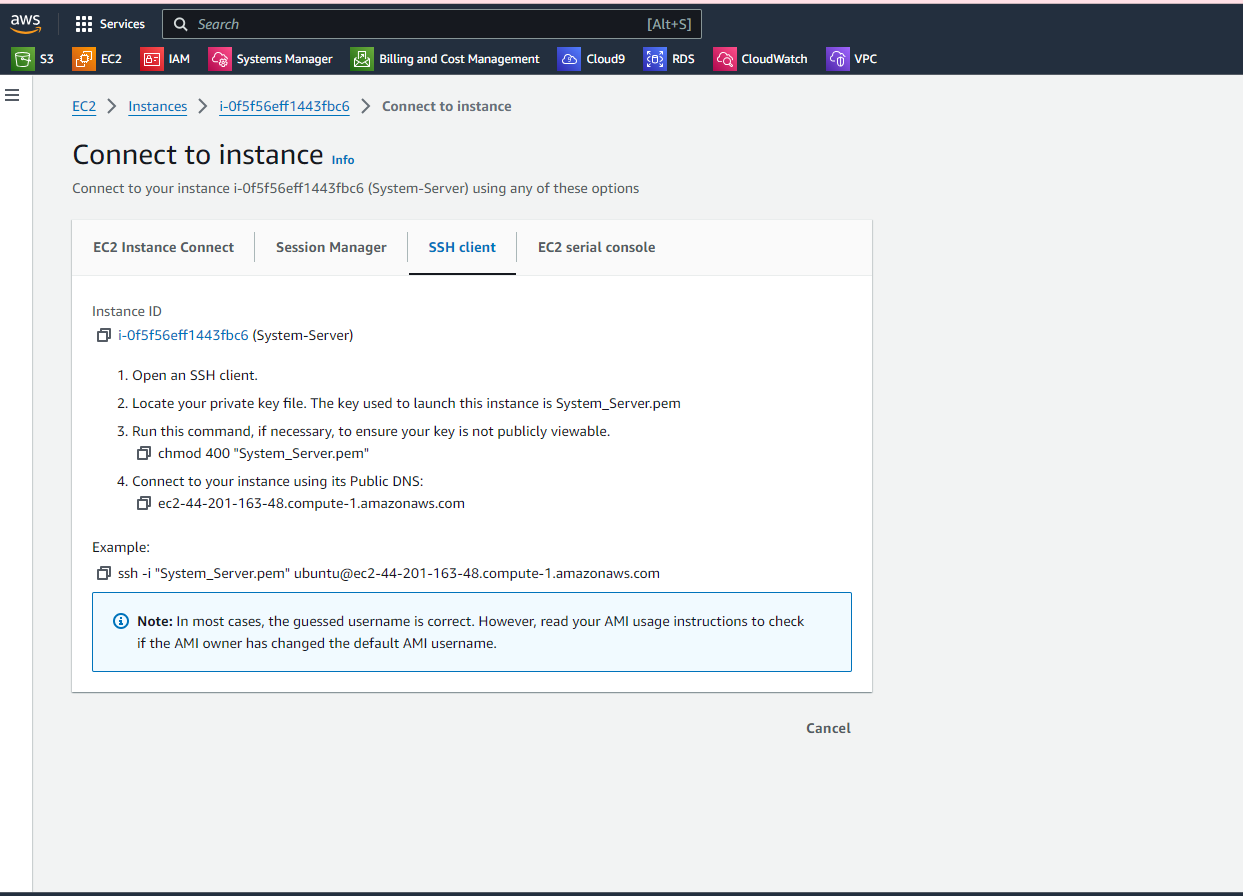
1. Navigated to the AWS Management Console.
2. Navigate to the EC2 Dashboard and click "Launch Instance".
3. Choose an Amazon Machine Image (AMI). You can use the free tier eligible "Ubuntu Server 20.04 LTS".
4. Choose an Instance Type. The t2.micro instance type is free tier eligible.
5. Configure Instance Details. You can keep the defaults.
6. Add Storage. The default is 8 GB, which is usually sufficient for basic setups.
7. Add Tags (optional).
8. Configure Security Group. Create a new security group and add the following rules:
   * SSH (port 22) from your IP address.
   * HTTP (port 80) from anywhere (optional if you plan to host a web server).
9. Review and Launch. You will be prompted to create or select a key pair. This key pair will be used to SSH into your instance.

#### **3 SSH into the Instance**

1. Download the key pair (.pem file) and save it in a secure location.
2. Change the permissions of the key pair file:

**chmod 400 path/to/your-key-pair.pem**

4. SSH into my instance:



### **5. Created the Backup Script**

Created a bash script named backup.sh. This script will perform the backup, upload it to S3, and send email notifications at each stage.

1. In the file named backup.sh wrote the below code, now let's break it down
2. **#!/bin/bash**

The shebang line specifies that the script should be run using the bash shell.

1. **The Configuration**

SOURCE\_DIR: The directory containing the data to be backed up.

BACKUP\_DIR: The directory where the backup file will be stored temporarily.

BACKEDUP\_FILE: The name of the backup file, which includes the current date for uniqueness.

DEST\_BUCKET: The S3 bucket where the backup will be uploaded.

ADMIN\_EMAIL: The email address to receive notifications about the backup process.

1. **Email Notification Function**

* send\_email: A function to send email notifications. It uses the mutt email client.
* subject: The subject of the email.
* message: The body of the email.

1. **Notify Backup Start**

This sends an email notification indicating that the backup has started

1. **Create the Backup**

tar -czf: Creates a compressed archive (.tar.gz) of the source directory.

$?: Checks the exit status of the tar command. If it is 0, the command was successful.

1. **Upload to S3 and Notify**

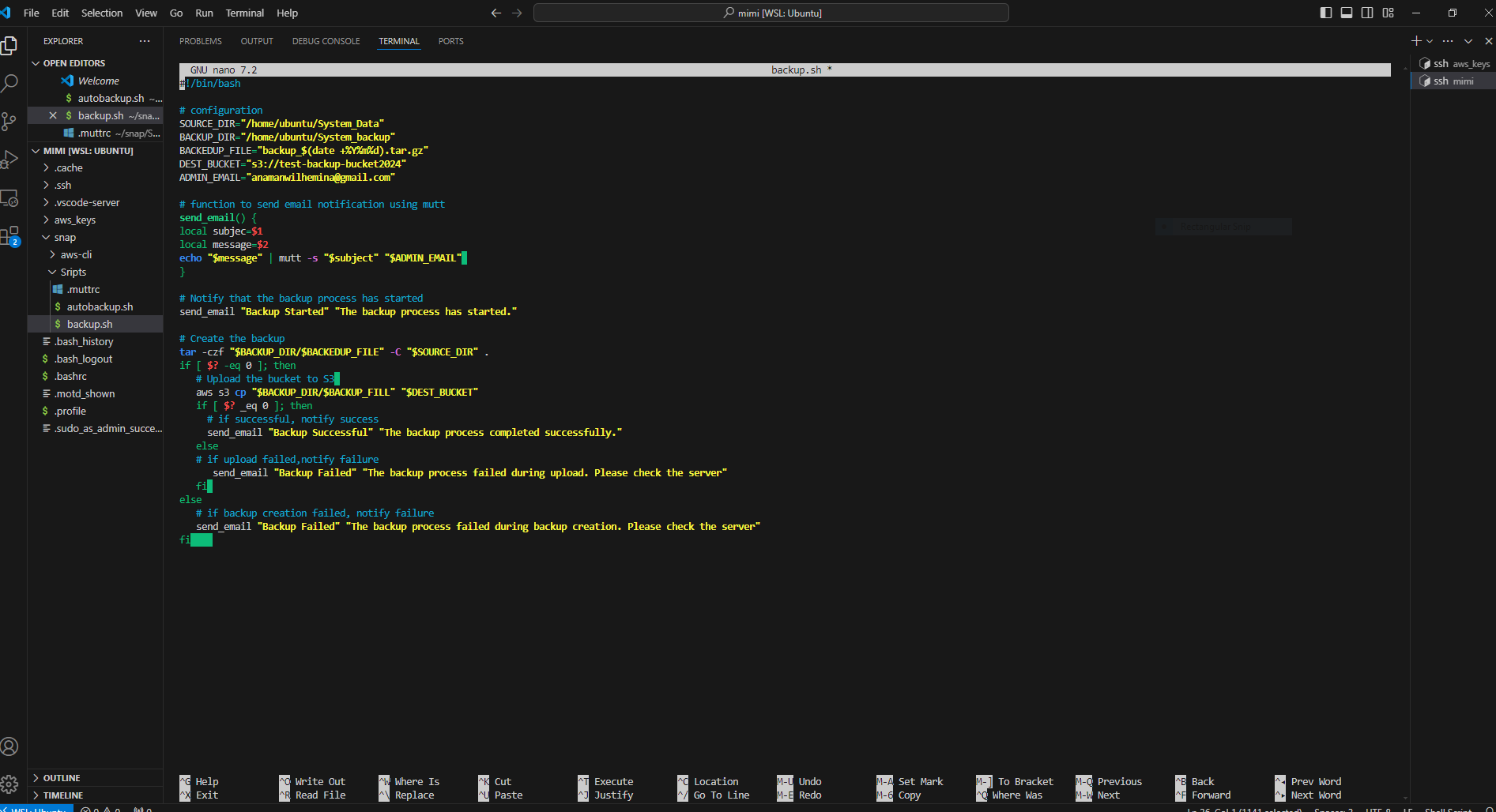
**aws s3 cp**: Uploads the backup file to the specified S3 bucket.

If the upload is successful, it sends an email notification indicating success. If it fails, it sends an email notification indicating failure.

1. **Notify Backup Creation Failure**

If the backup creation fails, it sends an email notification indicating failure.

backup.sh File



**6**.**Set Up the Cron Job**

Edit the crontab file to schedule the backup script to run every Friday after working hours

Edit the crontab file, use the crontab -e command:

Head over to Crontab Guru to calculate the cron schedule expression for every Friday after work and added it to the last line **30 5 \* \* 5 /home/ubuntu/backup.sh**. This entry schedules the execution of a script located at /home/ubuntu/backup.sh at 5:30 AM every Friday.

**Minute (m)**: The minute of the hour when the task should run (0-59).

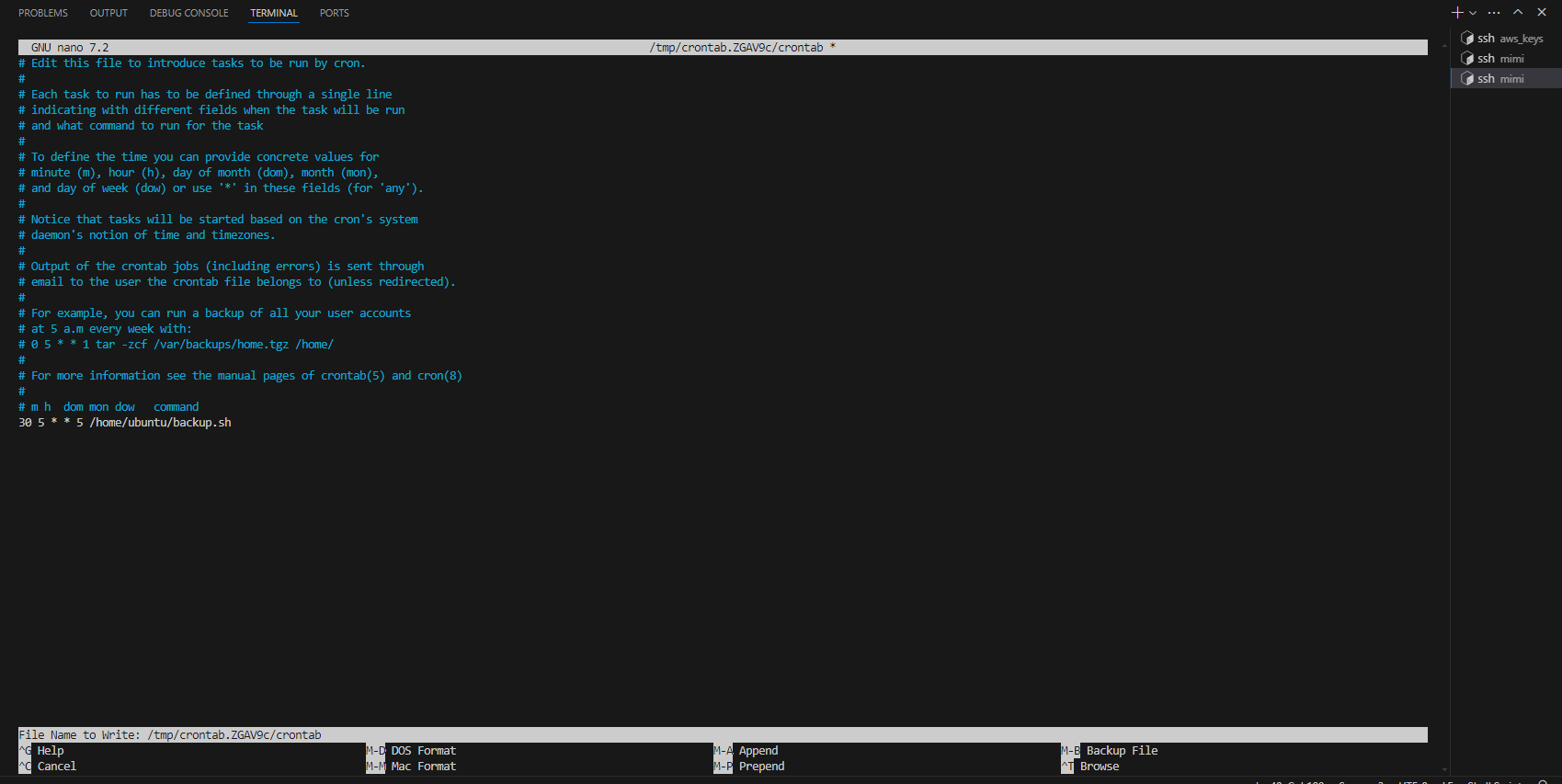
**Hour (h)**: The hour of the day when the task should run (0-23).

**Day of Month (dom)**: The day of the month when the task should run (1-31).

**Month (mon)**: The month when the task should run (1-12 or names like Jan, Feb).

**Day of Week (dow)**: The day of the week when the task should run (0-7, where both 0 and 7 represent Sunday, or names like Sun, Mon).

**Crontab image**

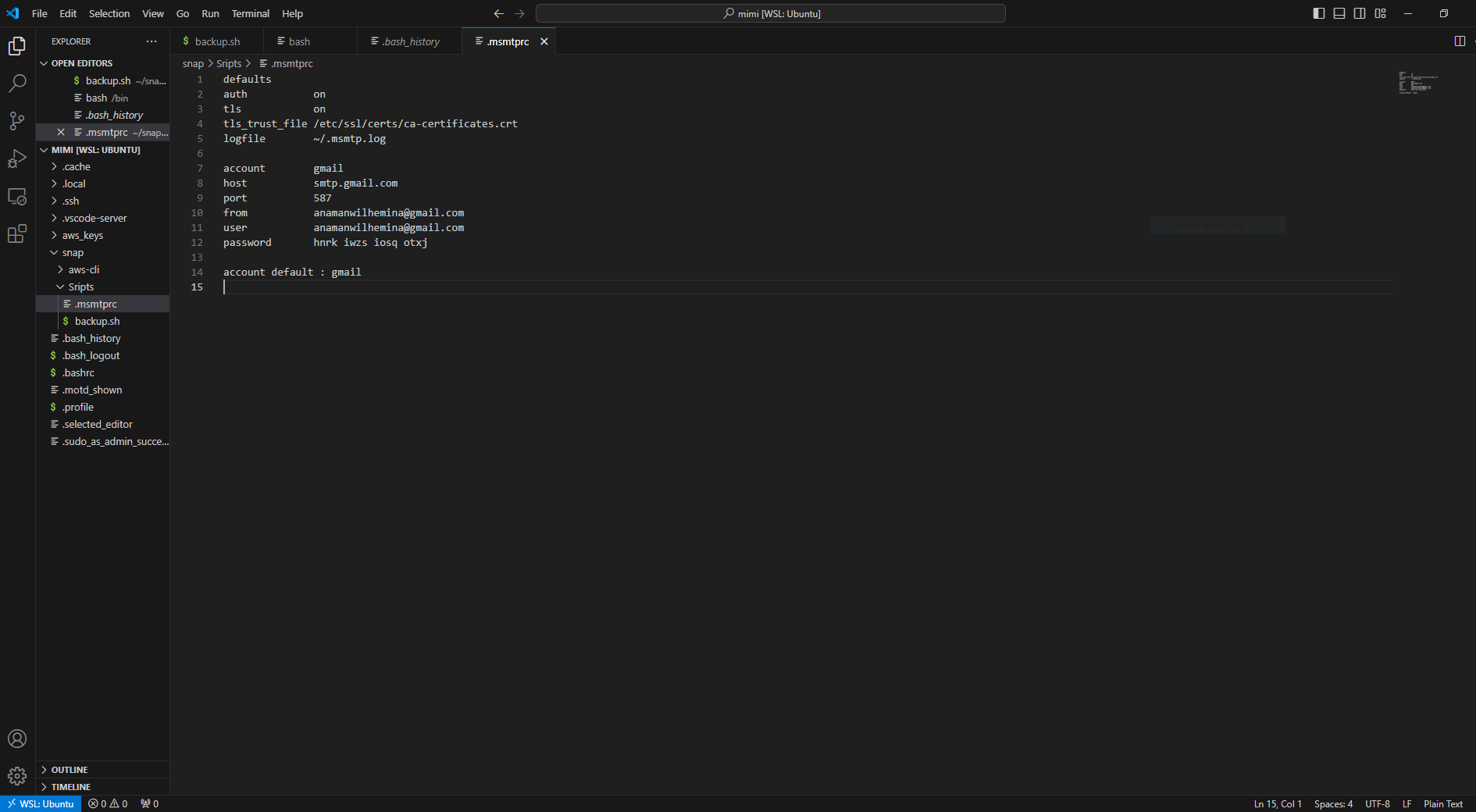


7. Set up an email client to handle the email notification , msmtp is a lightweight and easy-to-use SMTP client that can send emails using an external SMTP server. It is commonly used in scripts and command-line operations to send emails programmatically. I installed using these commands in the terminal **sudo apt update** and then **sudo apt** install msmtp . after installation was successful i created another

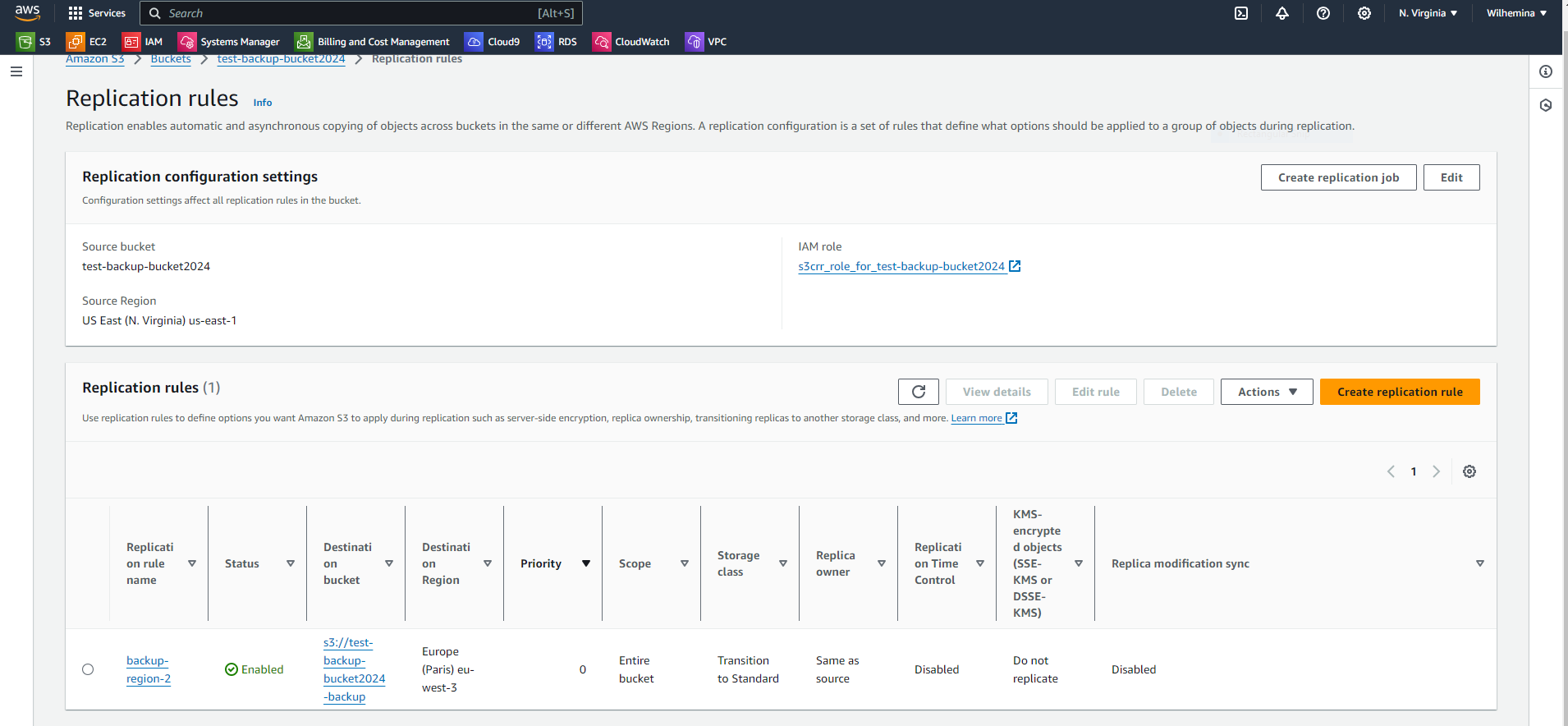
### **Explanation of Each Setting**

* **auth on**: Enables authentication, which is necessary for most SMTP servers to prevent unauthorized use.
* **tls on**: Enables TLS encryption, ensuring that the connection between your client and the SMTP server is secure.
* **tls\_trust\_file /etc/ssl/certs/ca-certificates.crt**: Specifies the location of the trusted certificates. This helps verify that the SMTP server is legitimate and prevents man-in-the-middle attacks.
* **logfile ~/.msmtp.log**: Logs all msmtp activities to the specified file, which helps in troubleshooting email delivery issues.
* **host smtp.gmail.com**: The SMTP server address for Gmail.
* **port 587**: A common port for submitting emails over a TLS-encrypted connection.
* **from anamanwilhemina@gmail.com**: The email address that will be shown as the sender of the email.
* **user anamanwilhemina@gmail.com**: The email account's username for authentication.
* **password hnrk iwzs iosq otxj**: The email account's password. (Note: It's recommended to use application-specific passwords or environment variables for security.)
* **account default : gmail**: Sets the specified account as the default, so it's used automatically if no other account is specified when sending an email.

**script with email client setup in the image below :**

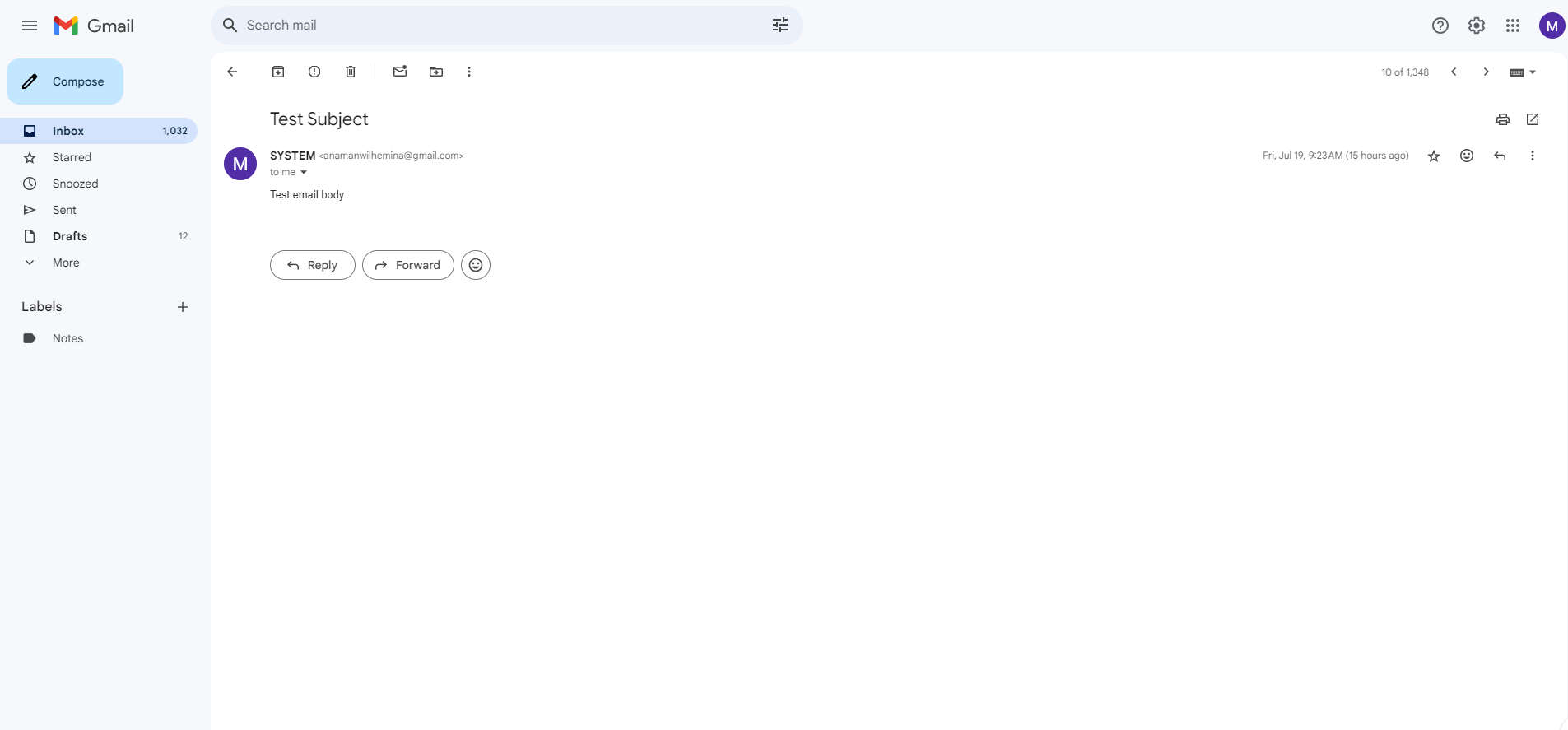


* At this stage I navigated to the AWS management console and two buckets in S3.
* Created the first bucket in the us-east-1 region , this bucket is to store uploads
* And the second bucket was created in Paris , this bucket is to store backups
* Created a replication rule for the primary bucket to automatically replicate data to the backed-up bucket , ensuring that i alway have up-to-date copies of your data.

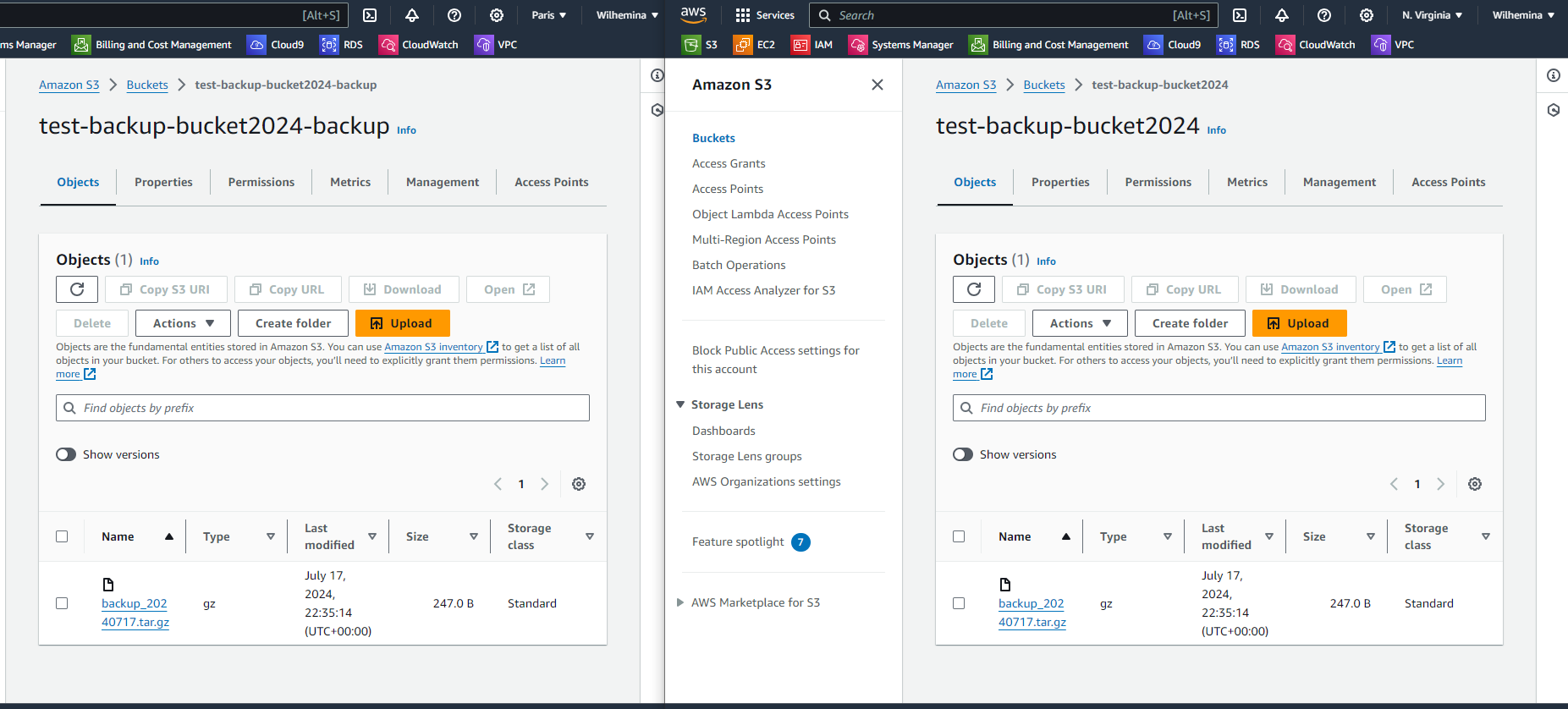


To test if my email client configuration was correct, i used this command to send a test email to my gmail :

**echo "This is a test email" | msmtp --debug --from=default -t anamanwilhemina@gmail.com**

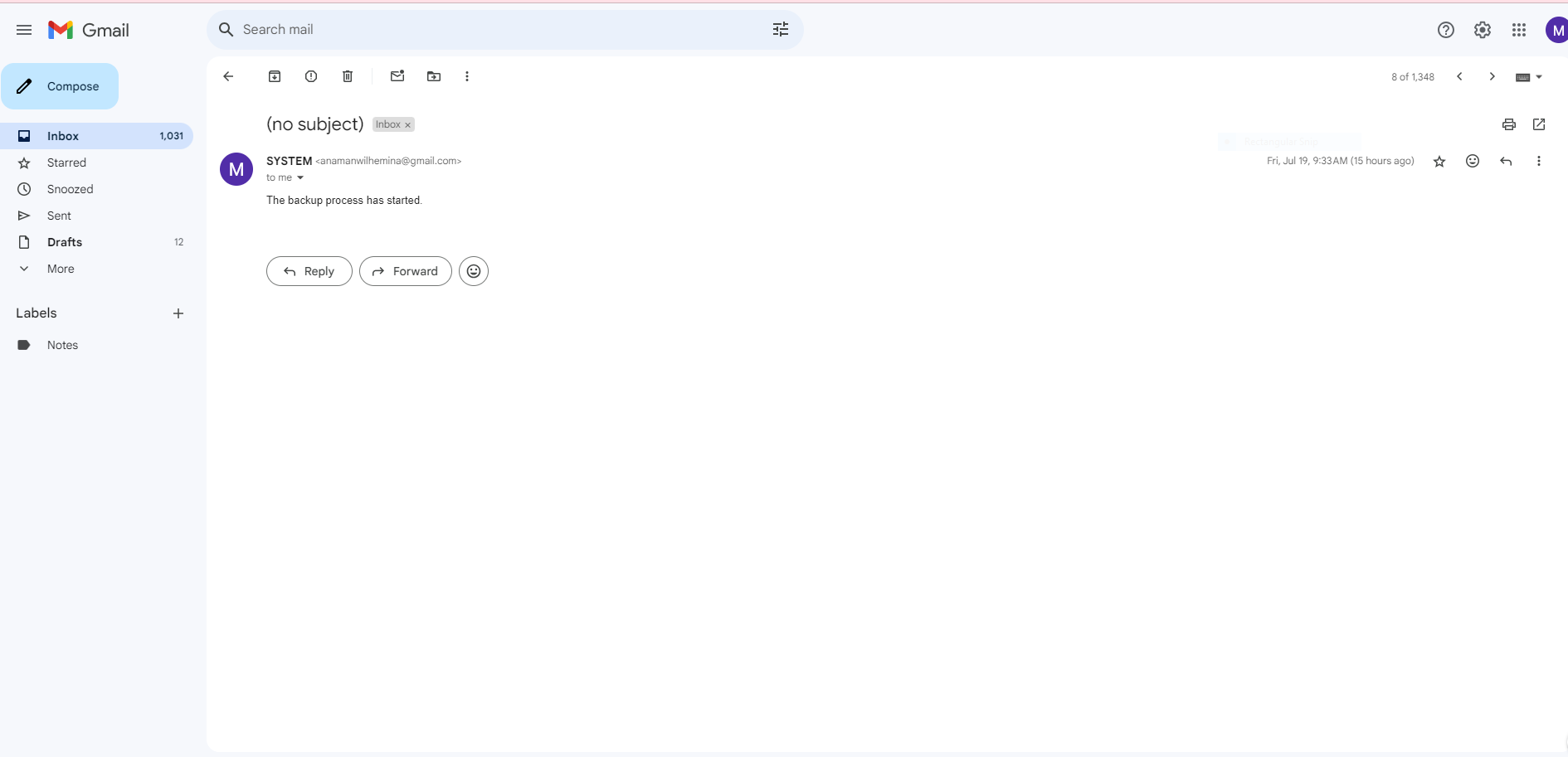
****

The above image shows the email client configuration was successful so I went ahead and tested the cron job using this command **./backup.sh**

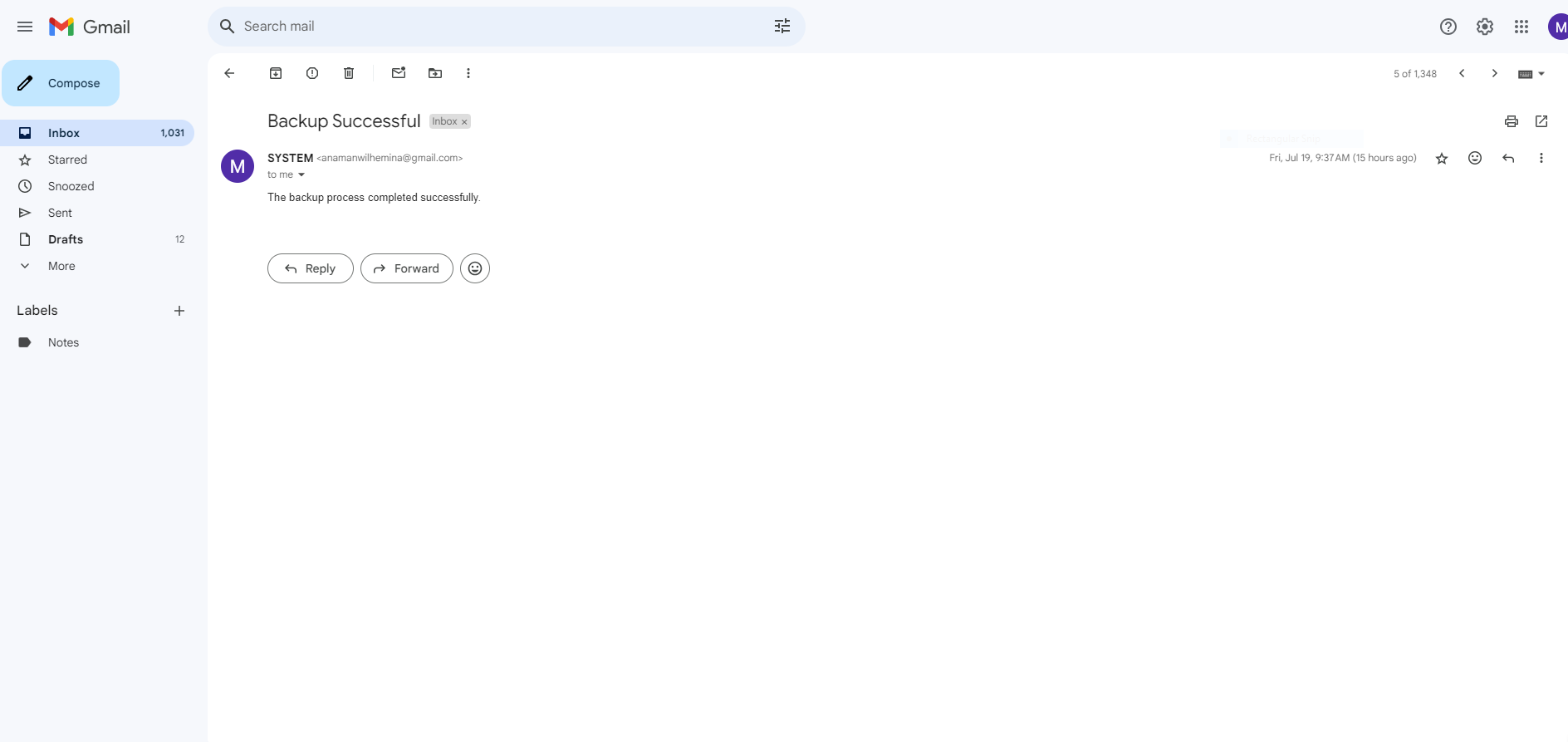
****

The image above indicates that the upload to both the primary bucket and the backed-up bucket is successful.

**Received an email when the backup process started**



**Received another email when backup was successful**

****

### 10. Implemented S3 Event Notification

To notify system admins of an object upload in the S3 bucket, i used S3 Event Notifications with Amazon SNS or

Using Amazon SNS:

Created an SNS Topic:

* + Created a new topic and named it Backup-Topic.
  + Created a subscription to this topic with the email addresses of my system admins.

Configured S3 Event Notifications:

* + Select my primary bucket.
  + Navigated to the "Properties" tab.
  + Scrolled down to "Event notifications" and created a new notification.
  + Configured the event to trigger on "All objects create events".
  + Select the SNS topic Backup-Topic as the destination.

