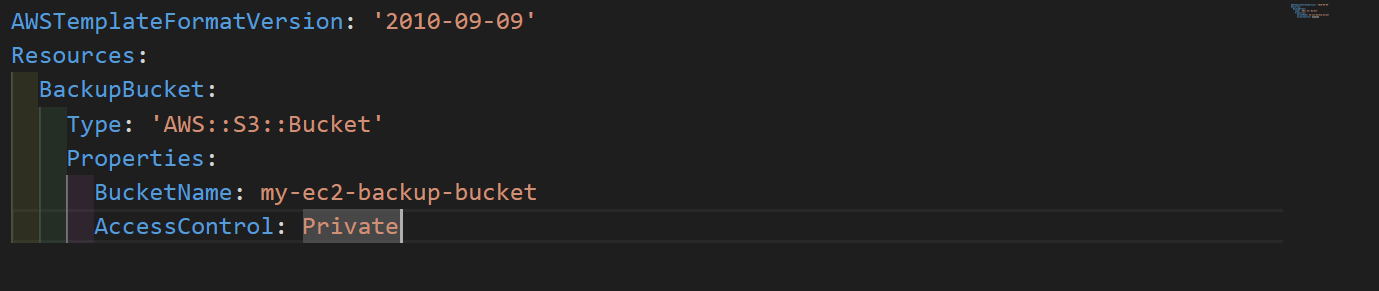
**STEP 1 CREATE AN S3 BUCKET.**

Use Terraform or cloud formation to create an S3 bucket where the backups will be stored. You can use any text editor of your choice.

Cloud formation template for creating S3 bucket.



Terraform template for creating an S3 bucket



These two templates create an S3 bucket and set its access to private.

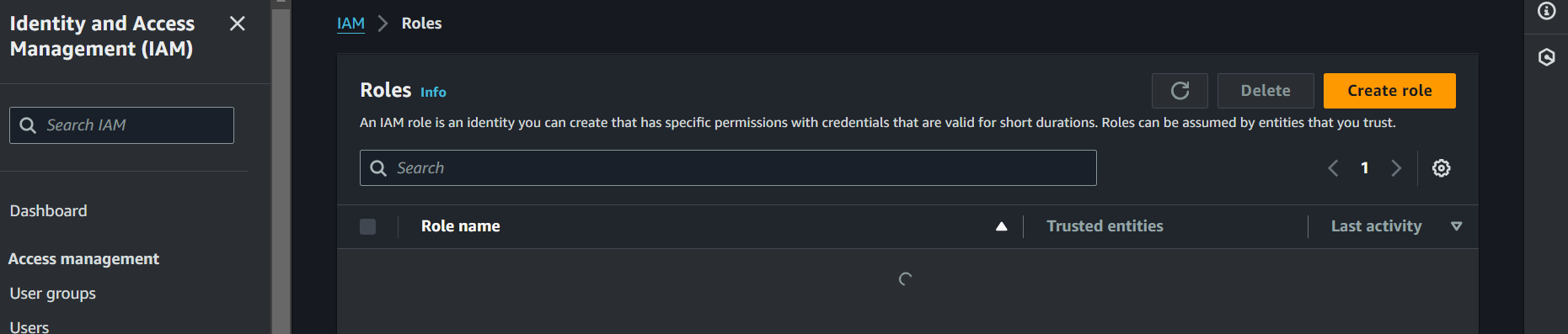
**STEP 2: IAM ROLE FOR EC2**

You create an IAM role that grants EC2 instances permissions to take backups and upload them to S3.

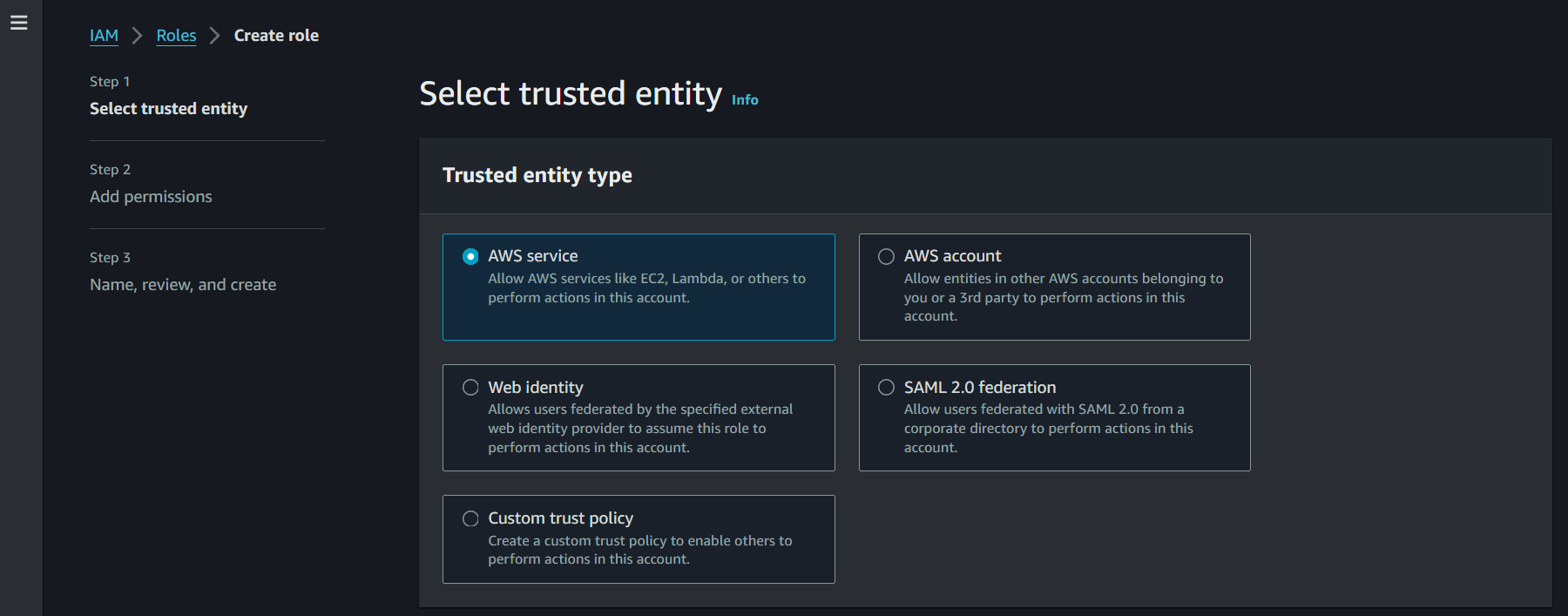
1. Open the IAM console in the AWS Management Console.



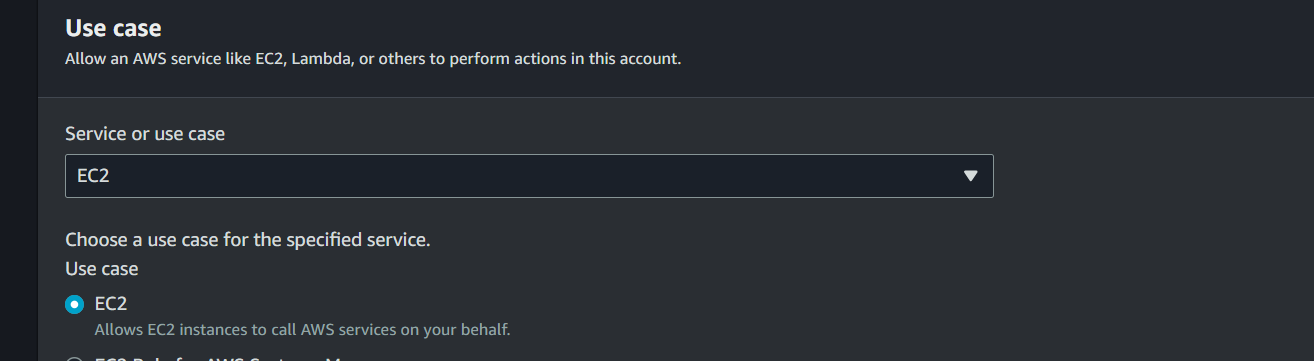
In the navigation pane, choose "Roles" and "Create role".



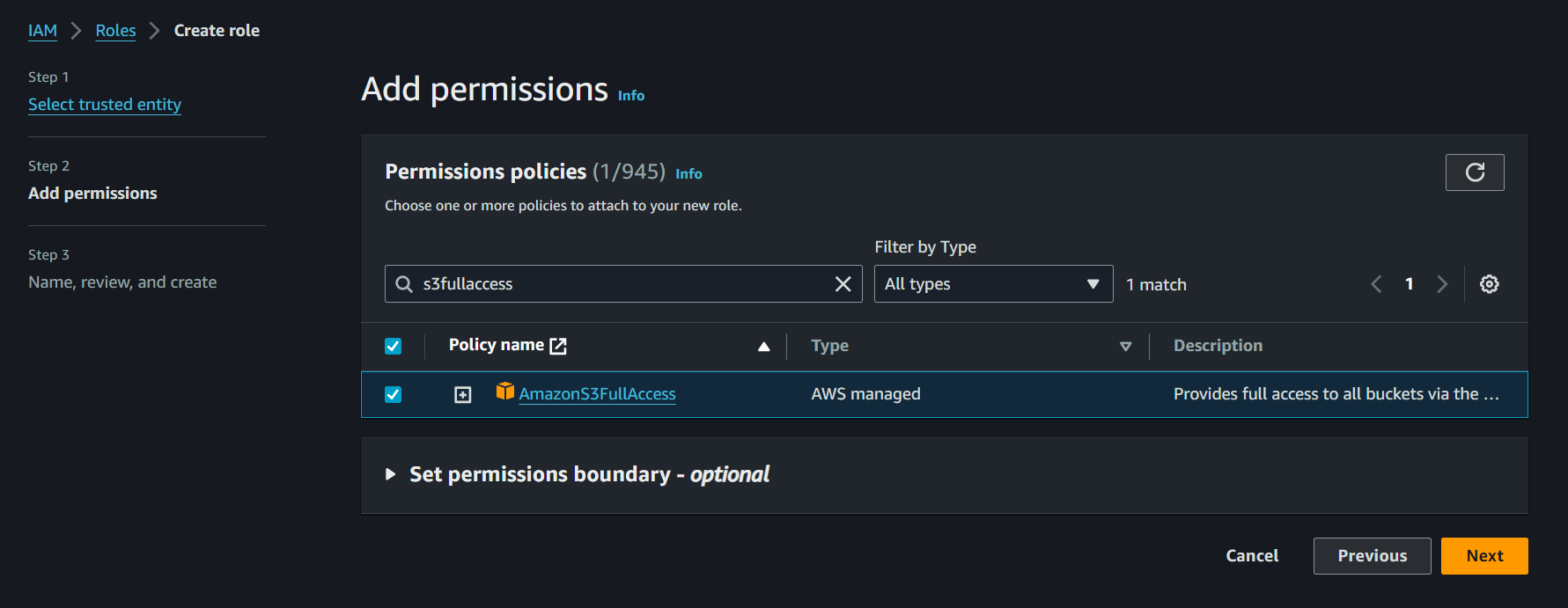
1. Select "EC2" as the service which will use this role, then choose "Next: Permissions".



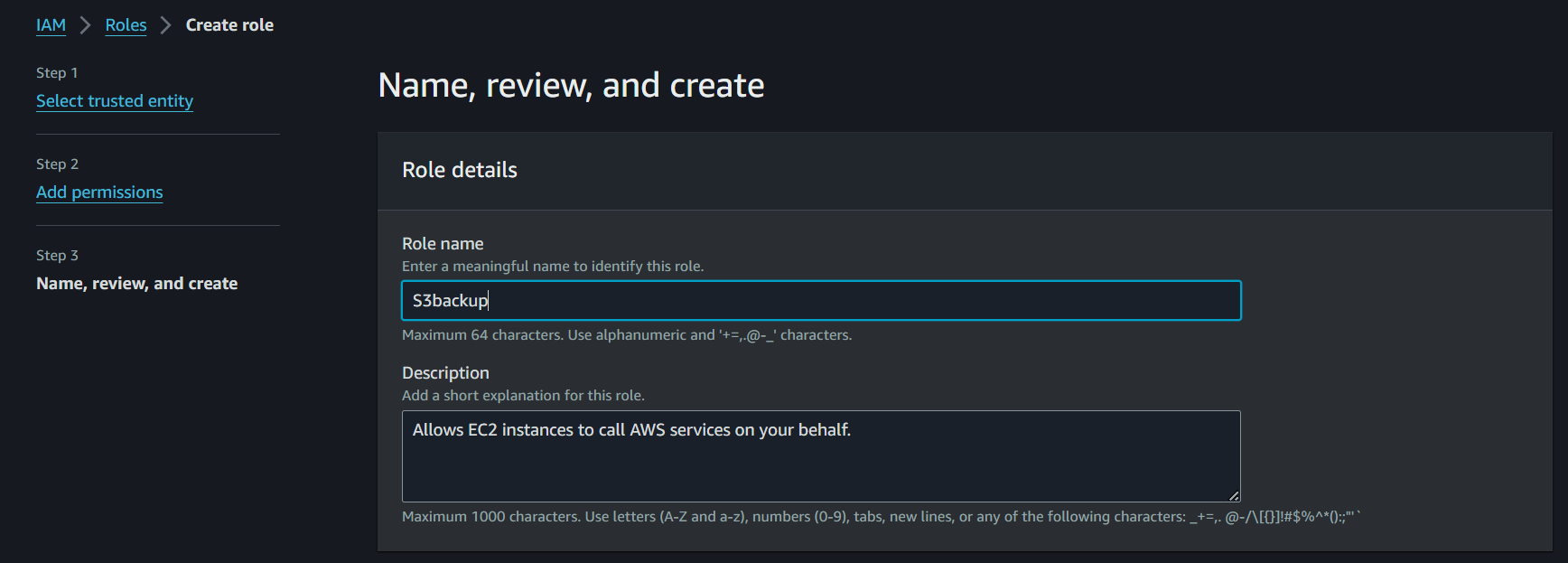
1. In the use case select EC2



1. Type S3FullAccess in the search bar and attach this policy to the role: AmazonS3FullAccess

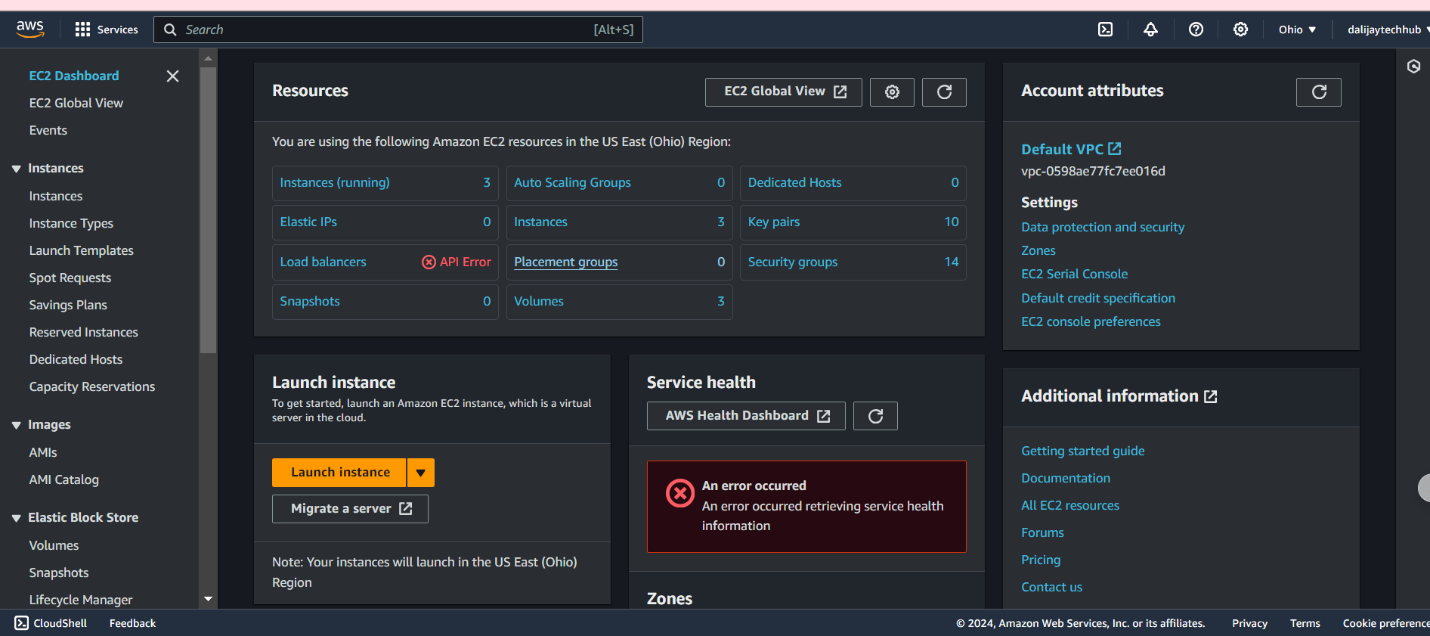


1. Give a name to the role and click on the create button.



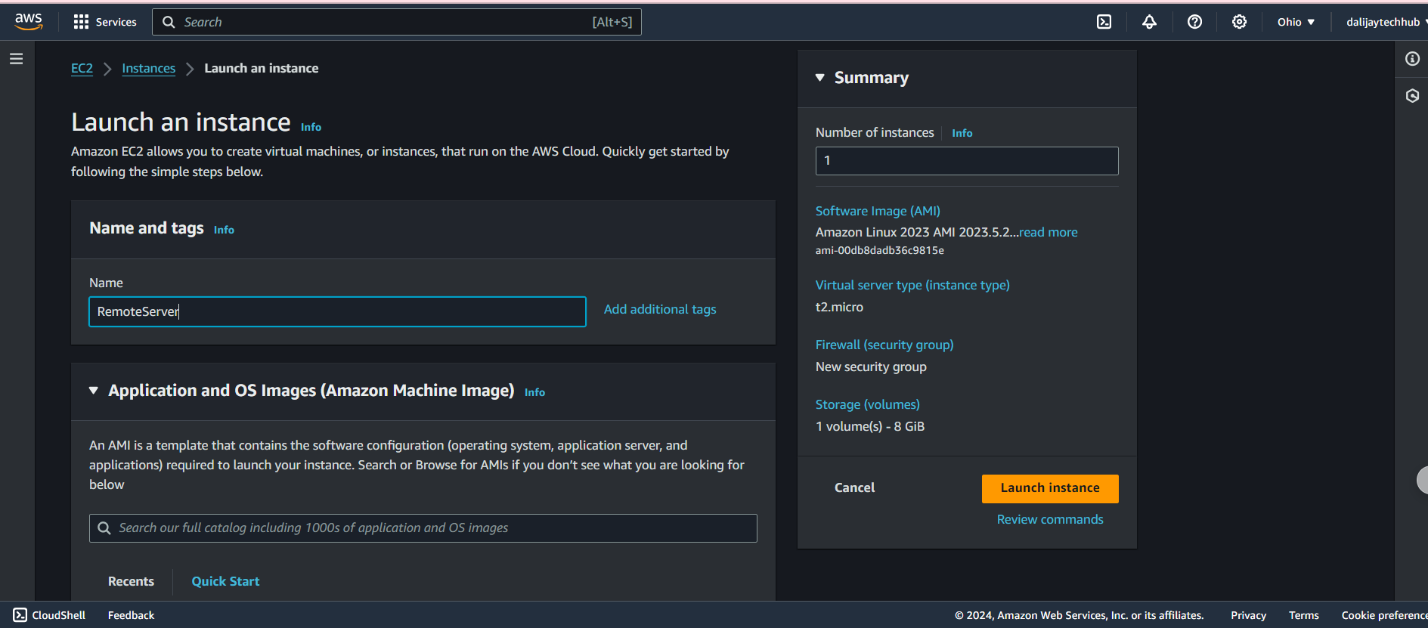
**STEP 3: CREATE AN EC2 INSTANCE AND ATTACH THE IAM ROLE FOR THE EC2 INSTANCE.**

1. Before you begin, ensure you have access to the AWS Management Console by signing in with your AWS account credentials. If you don't have an AWS account, create one by following the simple sign-up process on the AWS website.
2. After signing in to the AWS Management Console, navigate to the EC2 dashboard by typing "EC2" in the services search bar at the top of the page and selecting "EC2" from the search results. This will take you to the EC2 console, where you can manage your instances and perform various EC2-related tasks.



1. Launch a New Instance:

In the EC2 dashboard, initiate the instance creation process by clicking the Launch Instance button as shown above. This will guide you through a series of steps to configure your instance according to your specific requirements. Upon clicking the Launch Instance button, you will be redirected to a page that resembles the one shown below, where you can begin configuring your instance.



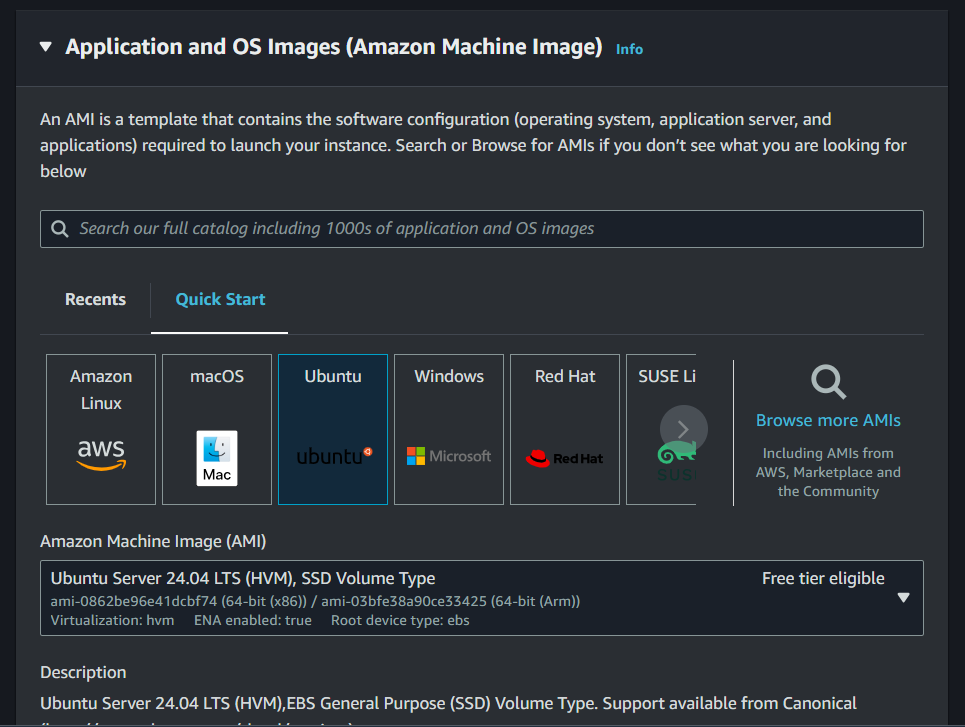
1. Assign a Name to Your EC2 Instance:

Assign a descriptive name to your EC2 instance to simplify the identification and management of your resources. Choosing a meaningful name will make it easier to distinguish your instance from others in the future. For this example, enter “RemoteServer” as the name for your EC2 server. This will help you quickly identify your instance in the EC2 console.

1. Select an Amazon Machine Image (AMI) type:

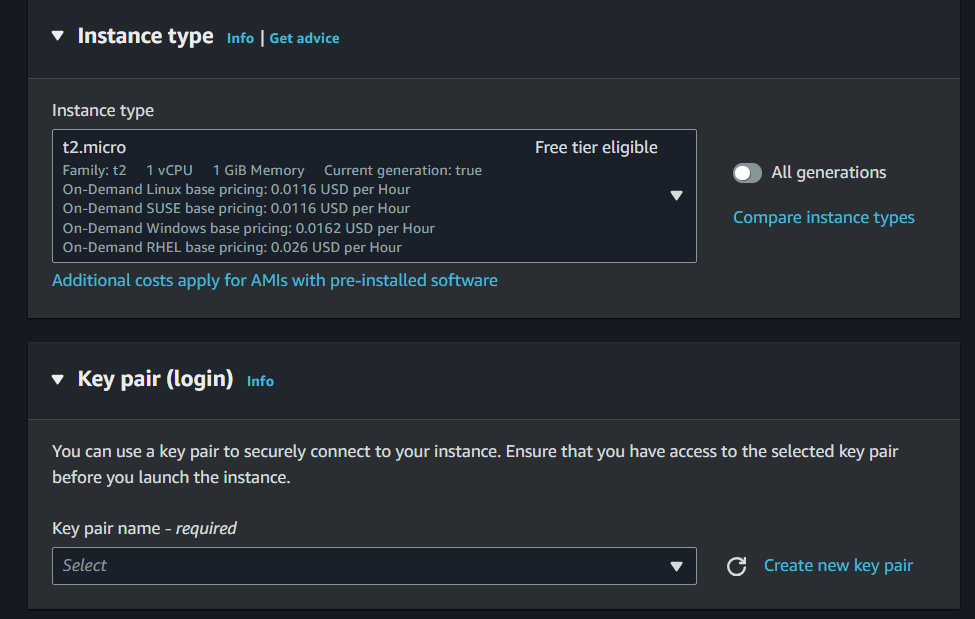
Choose an Amazon Machine Image (AMI), that contains the operating system and software required for your instance. AWS and the AWS Marketplace offer a diverse range of AMIs to suit various needs. Browse the available options and select the AMI that best fits your requirements. In this example, select Ubuntu as the operating system for your instance. This will provide a stable and secure foundation for your EC2 server.

1. Select an Instance type



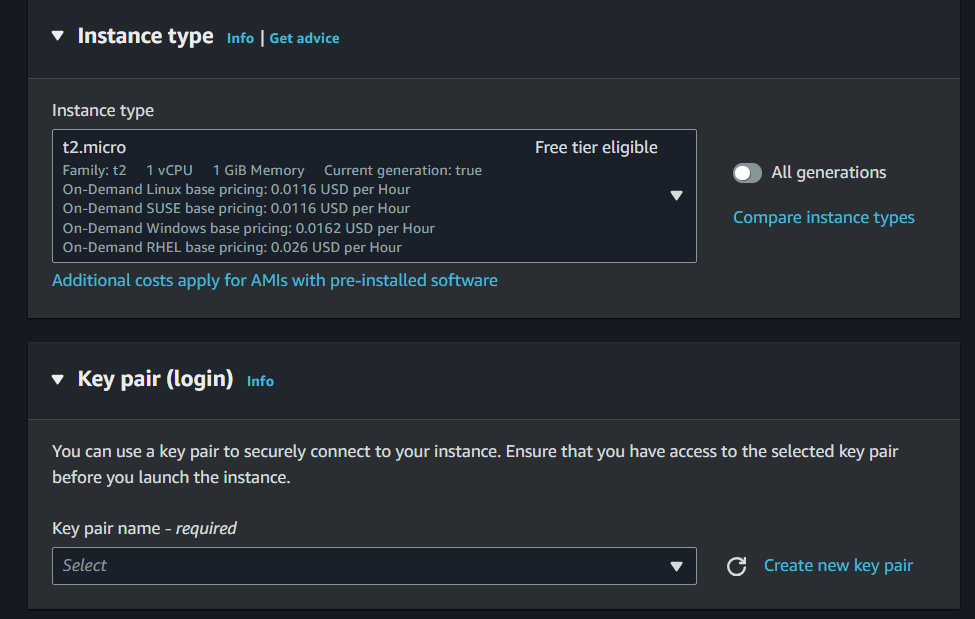
ype:

This determines the hardware specifications of your host computer by choosing an instance type. AWS provides a variety of instance types, each with different computing, memory, and storage capacities. Assess your workload requirements and select the instance type that best aligns with your needs. In this scenario select t2 micro.

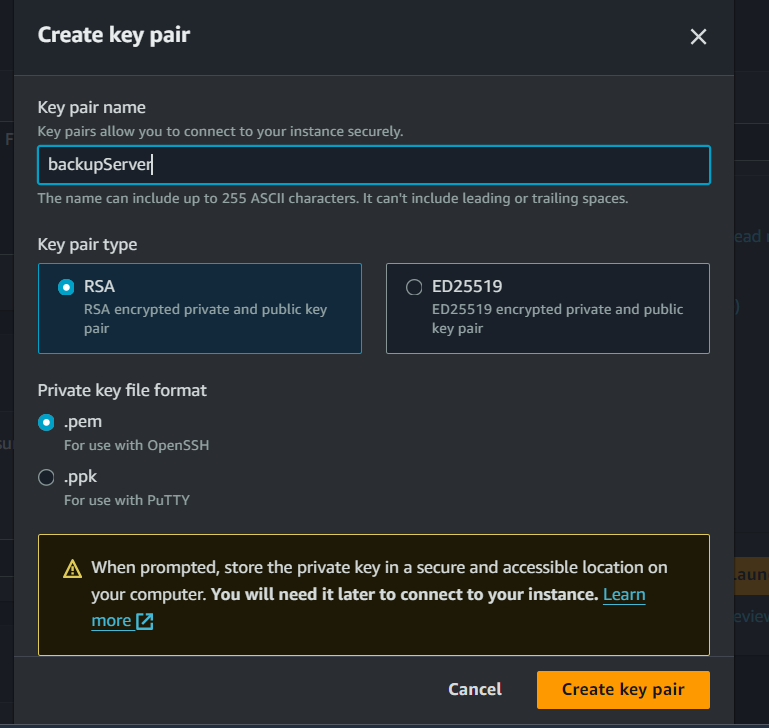


1. Create a Key Pair

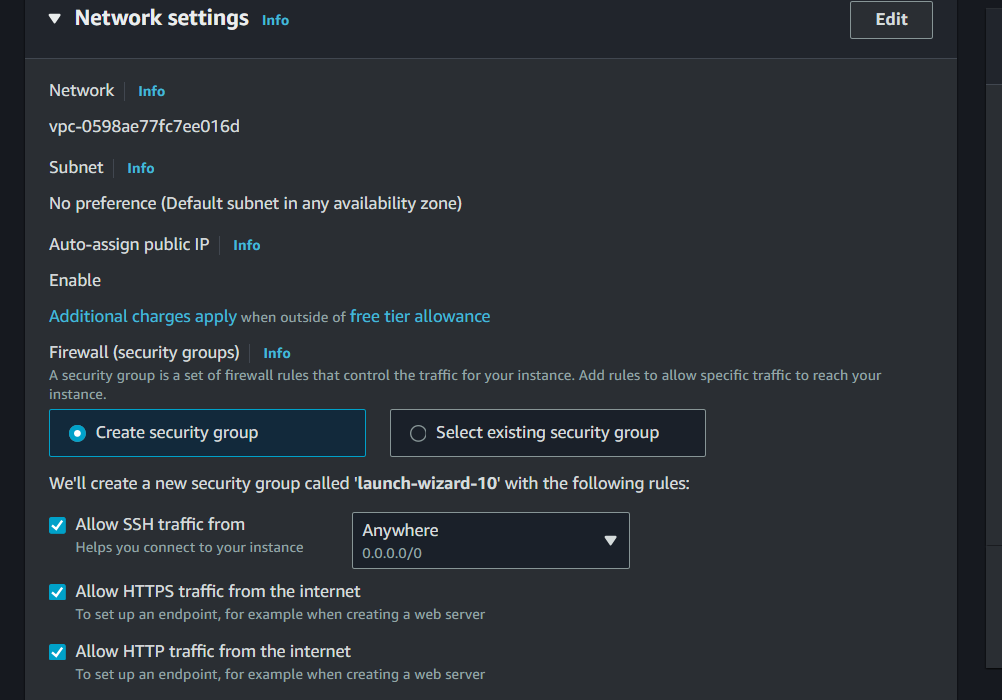
To securely access your EC2 instance from your local machine, you need to create a key pair. A key pair consists of a public Key: Stored by AWS, this key is used to authenticate your instance, and a private Key: Downloaded to your local machine, this key is used to establish a secure connection with your EC2 instance. By creating a key pair, you'll be able to securely access your instance and protect it from unauthorized access. Follow the prompts to create a new key pair by clicking on create new key pair as shown in the picture or use an existing one by clicking on the select arrow. Be sure to download and save the private key securely, as you won't be able to retrieve it again.



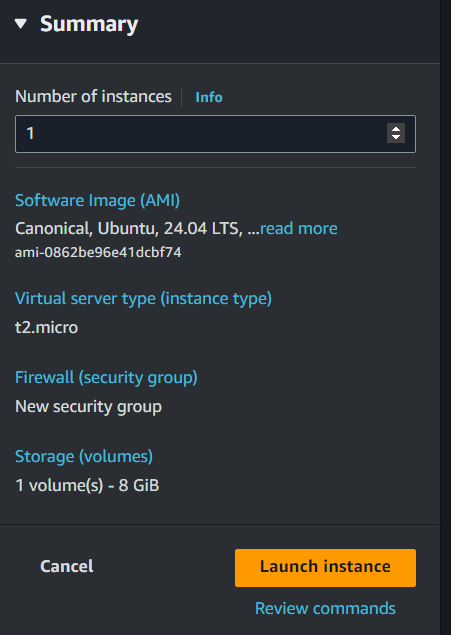
Type a name for your key pair and click on Create key pair. In this example type backupServer.



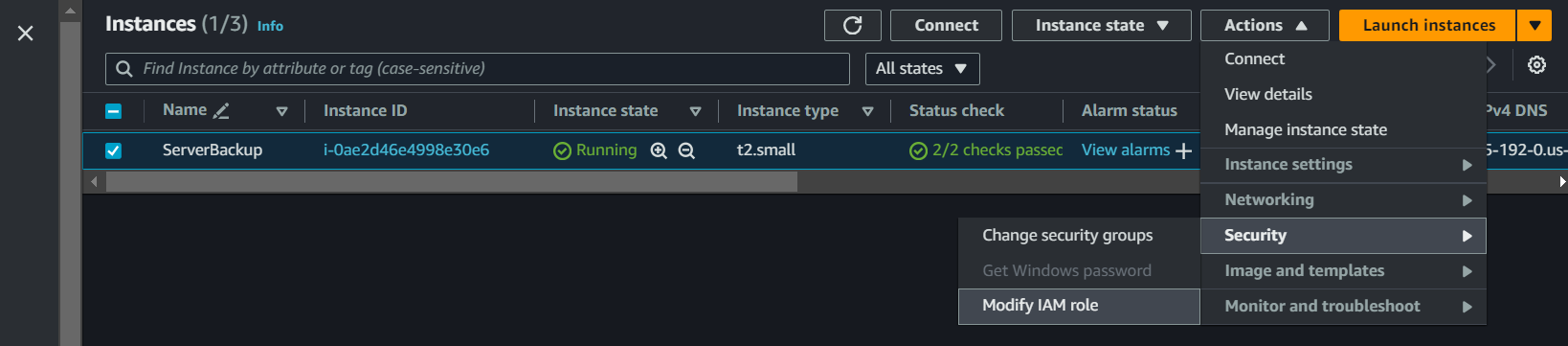
1. Configure Instance Details, by allowing http, https, and ssh in the network setting as shown in the diagram below. This will enable you to SSH into your EC2 instance.



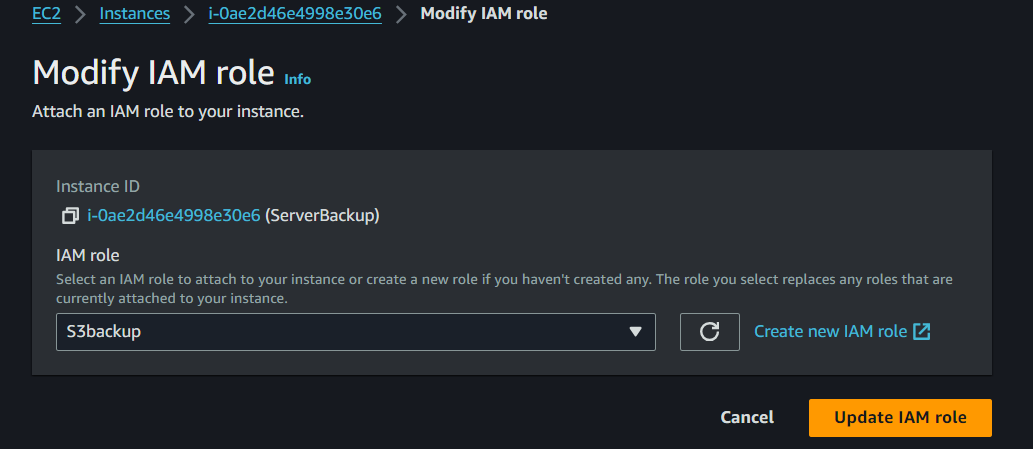
1. **Review and Launch**  
   Before launching the instance, review the configuration details to ensure everything is set up correctly. Click on the lunch instance button.



1. Go to an EC2 instance dashboard and click EC2 instance go to Actions and Security finally click Modify IAM Role.

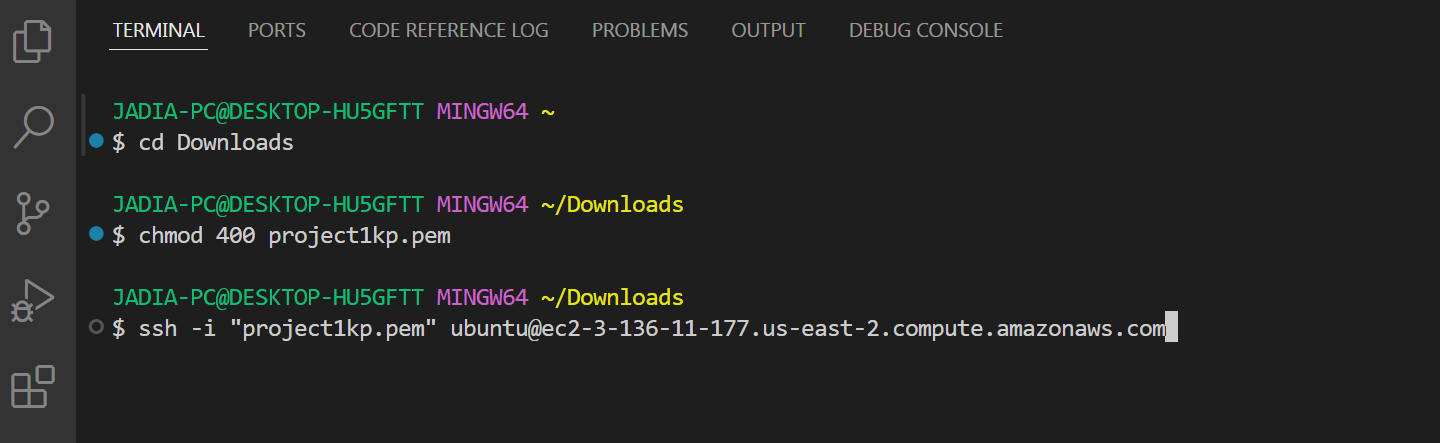


1. Next, Modify the IAM role, click the previously created IAM Role, and click Update the IAM role.

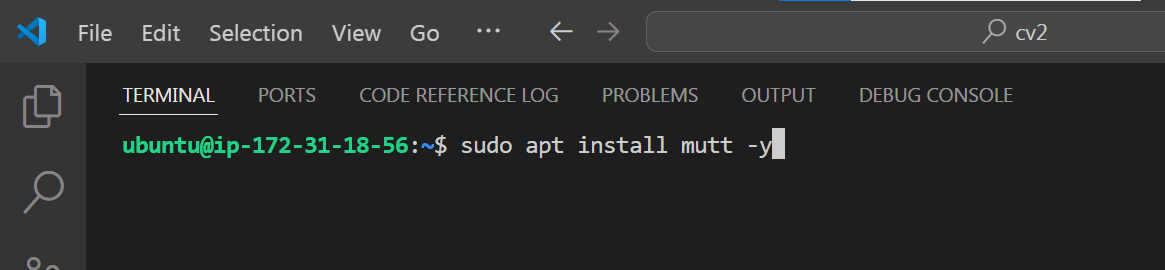


1. **4TH STEP - CONNECT EC2 INSTANCE AND INSTALL AWS CLI**

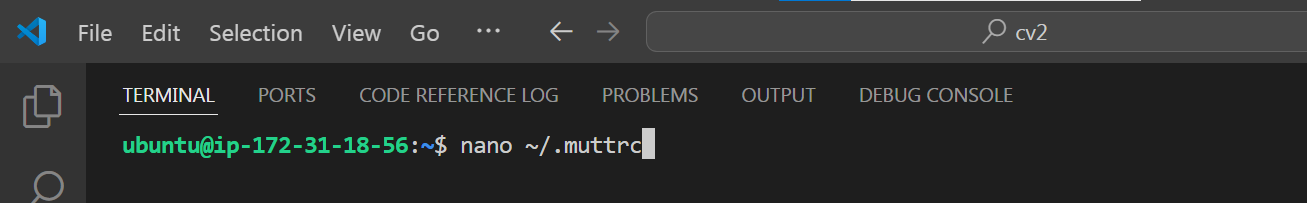
Step 4 - Connect to the EC2 Instance by typing the following command.

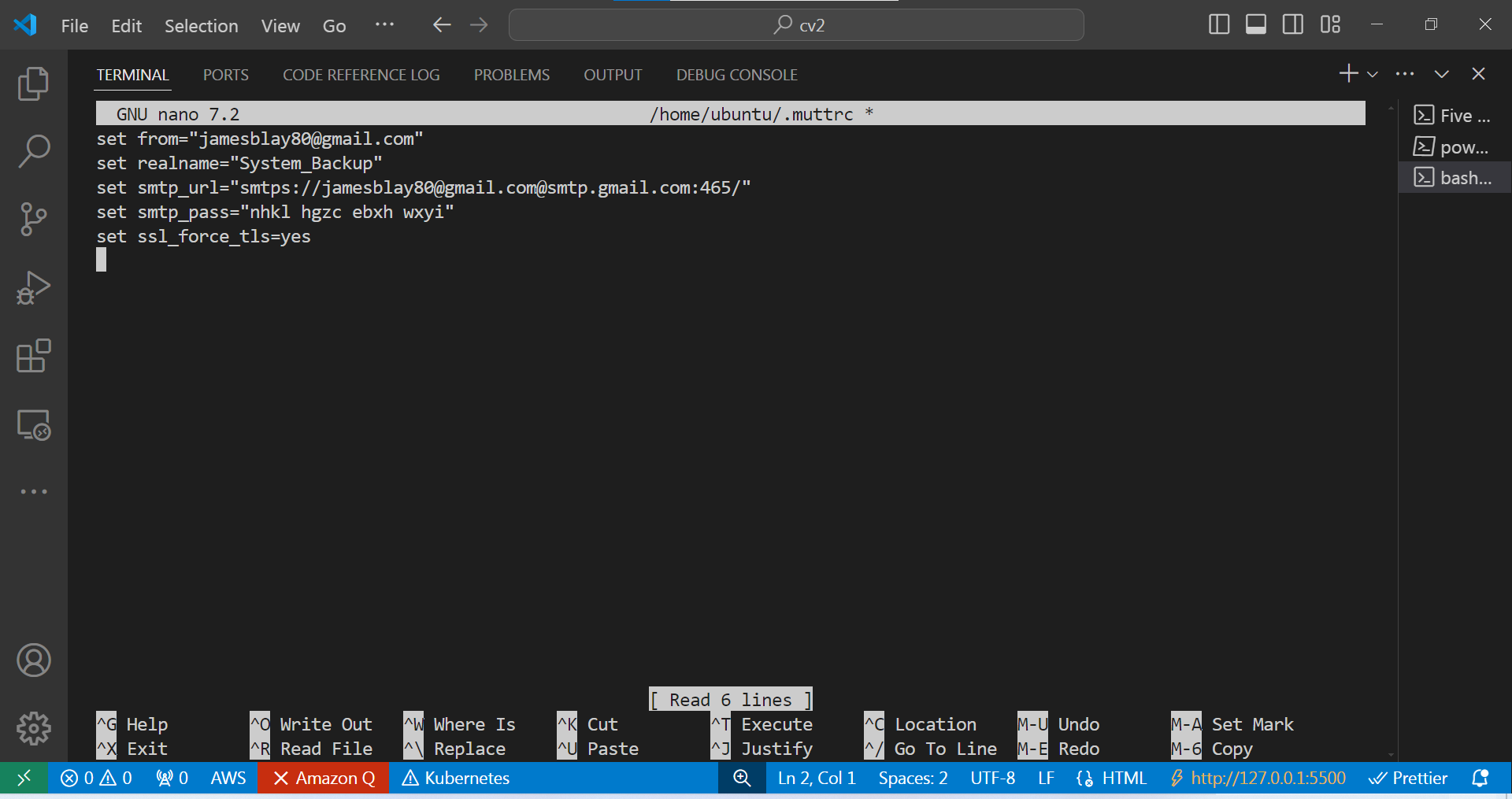


Step 5: Install the mail utilities.



Step 6: Set up your mail server which will enable you to receive email notification at every stage of the backup process. Type the following command as shown in the diagram below.





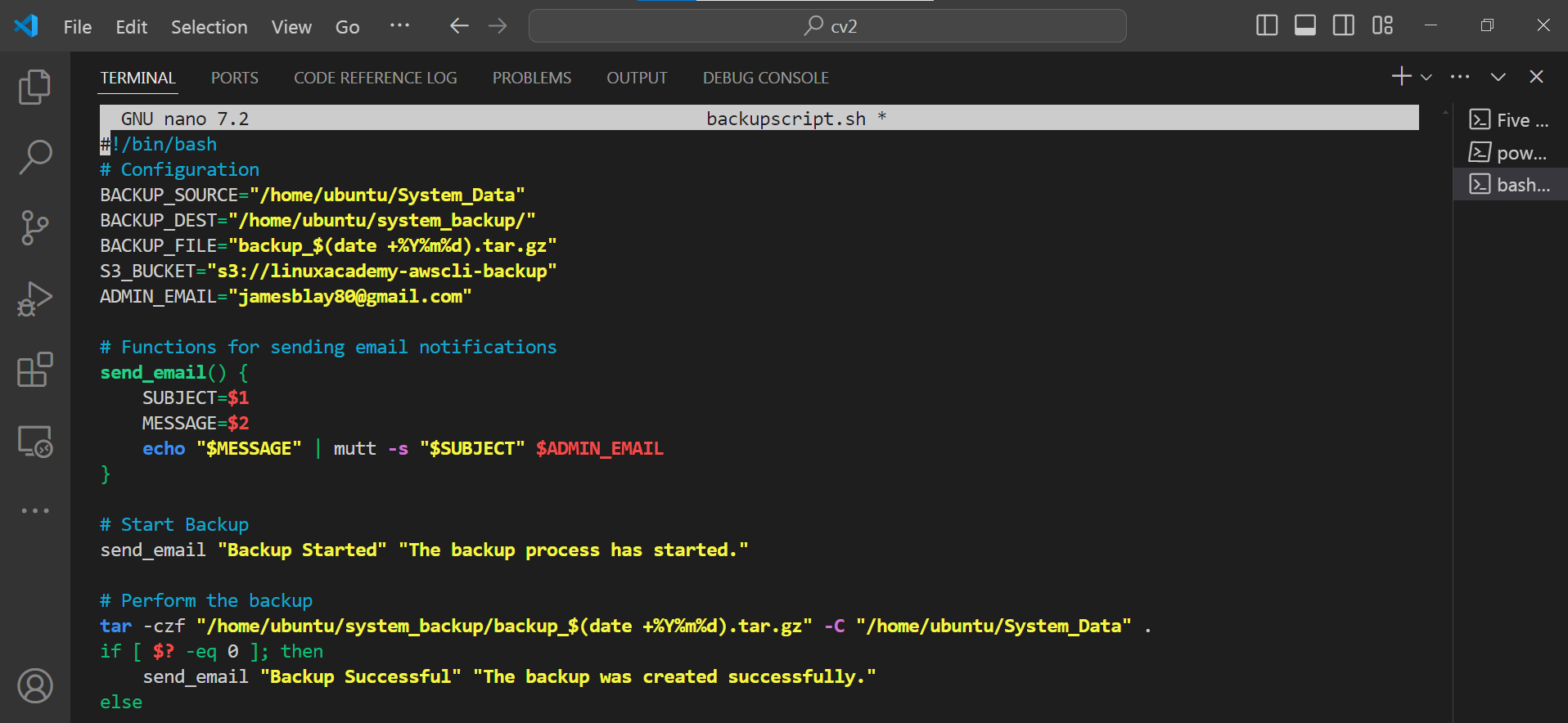
Step 7: Configure your mail server by typing the following command. Replace the email address with your email and the set smtp\_pass replace it with your app password. For more information on how to set Google app password click on this link. <https://www.youtube.com/watch?v=J4CtP1MBtOE>

When you finish the configuration click on CTRL + O and hint enter on the keyboard to save the configuration and CTRL + X to exit from the window.

Step 8: Now write the bash script to automate the backup process and transfer the backup file onto the S3 bucket by typing nano “backscript.sh”. You can use any editor either Vim, Nano, or GNU Emacs.



s



This is a Bash script that automates the backup of a system's data and uploads it to an Amazon S3 bucket. Here's a breakdown of the script:

Configuration

The script starts by defining several configuration variables:

BACKUP\_SOURCE: The directory to be backed up (/home/ubuntu/System\_Data)

BACKUP\_DEST: The directory where the backup file will be stored (/home/ubuntu/system\_backup/)

BACKUP\_FILE: The name of the backup file, which includes the current date (backup\_$(date +%Y%m%d).tar.gz)

S3\_BUCKET: The Amazon S3 bucket where the backup file will be uploaded (s3://linuxacademy-awscli-backup)

ADMIN\_EMAIL: The email address that will receive notifications about the backup process (jamesblay80@gmail.com)

Functions

The script defines a function send\_email that sends an email notification using the mutt command. The function takes two arguments: SUBJECT and MESSAGE.

Backup Process

The script then starts the backup process by sending an email notification with the subject "Backup Started".

The backup is performed using the tar command, which creates a compressed archive of the BACKUP\_SOURCE directory and stores it in the BACKUP\_DEST directory. The archive file name includes the current date.

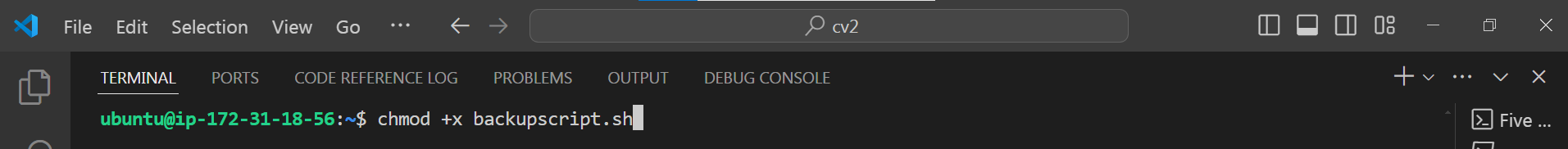
If the backup is successful, an email notification is sent with the subject "Backup Successful". If the backup fails, an email notification is sent with the subject "Backup Failed", and the script exits with a non-zero status code.

Upload to S3

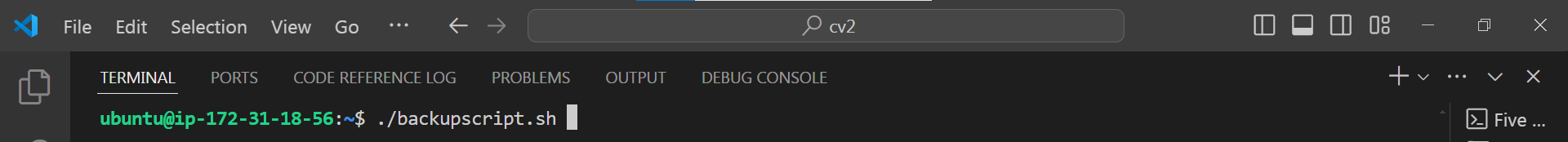
After the backup is complete, the script uploads the backup file to the specified S3 bucket using the aws s3 cp command.

If the upload is successful, an email notification is sent with the subject "S3 Upload Successful", and the backup file is deleted from the local system. If the upload fails, an email notification is sent with the subject "S3 Upload Failed", and the script exits with a non-zero status code.

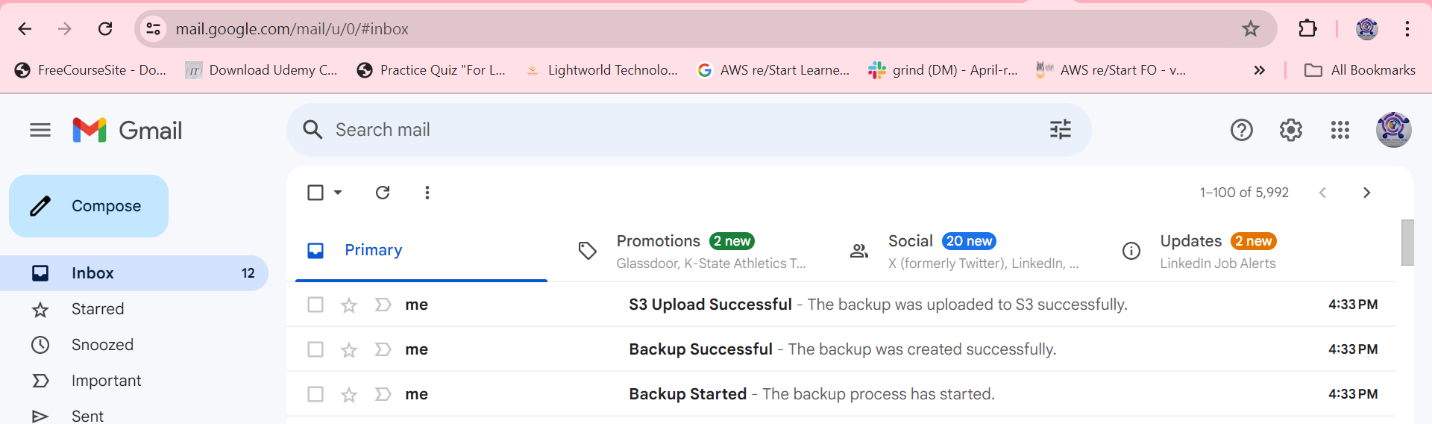
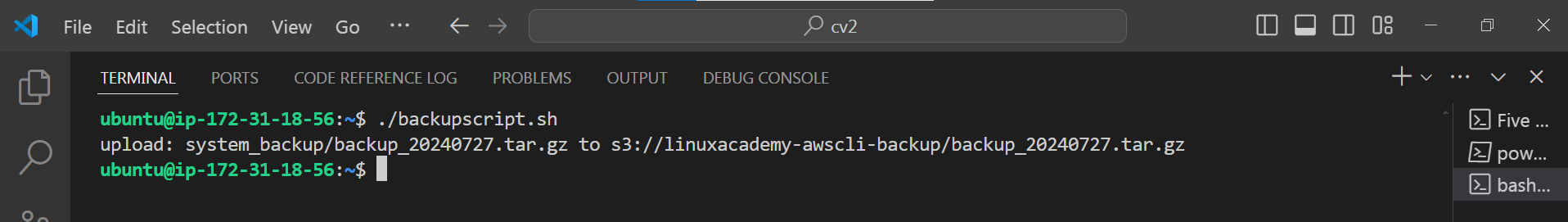
Step 10: Make your script file executable by typing the command shown below.

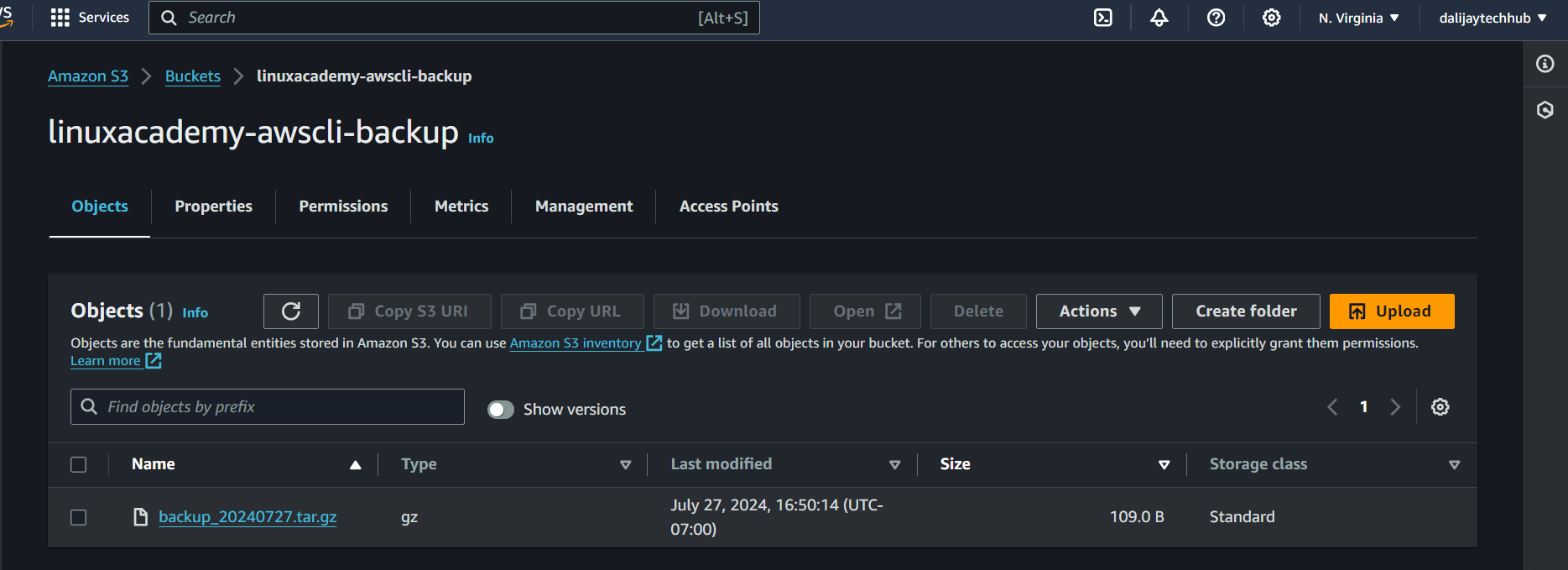


Step 11: Run the script manually to test if the script is working. Type this command below.

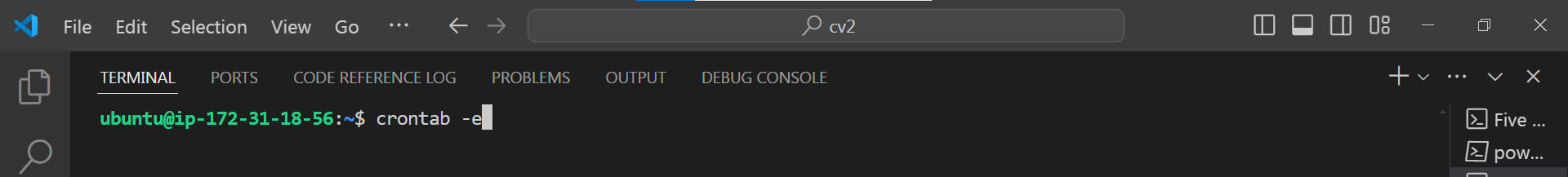
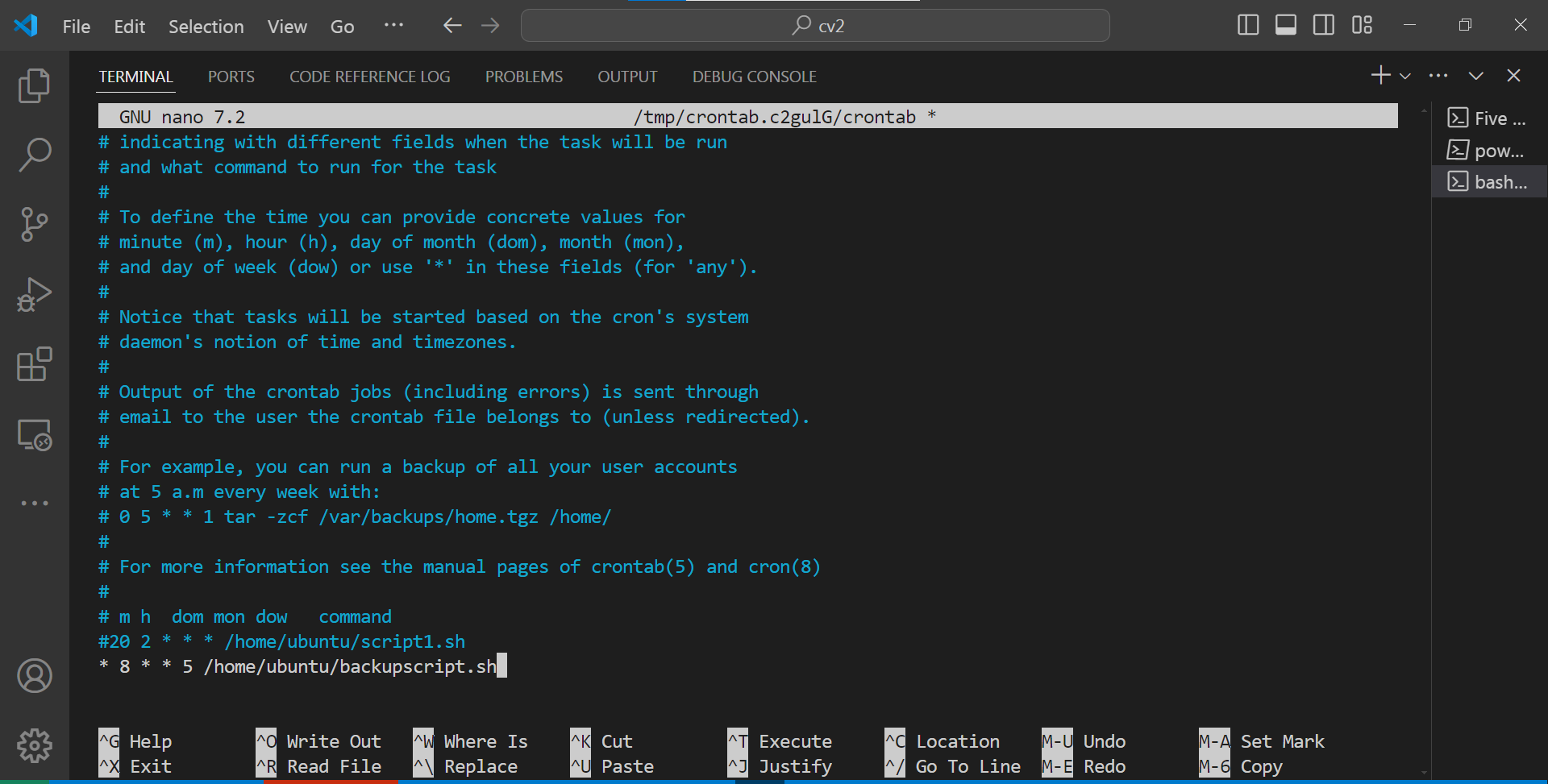


Step 12: When you run the script and it's successful, you will receive an email notification and a notification in the command line as shown below.





Step 13: Now lets set the cronjob to automate the whole backup process.



Here's a breakdown of the fields:

0: minute (0-59)

8: hour (0-23) - 8:00 pm is equivalent to 20:00 in a 24-hour format

\*: day of the month (1-31) - \* means any day

\*: month (1-12) - \* means any month

5: day of the week (0-6) - 5 means Friday (0 = Sunday, 1 = Monday, ..., 6 = Saturday)

/home/ubuntu/backupscript.sh: the command to run - replace with the actual path to your backup script

Save and exit the editor. The cronjob will be installed and run the backup script at 8:00 pm every Friday.

This setup automates taking EC2 instance backups and storing them in an S3 bucket using a bash script. Customise it according to your specific requirements and security best practices.