VASAVI COLLEGE OF ENGINEERING

AUTONOMOUS

(Affiliated to Osmania University)

Hyderabad- 500 031.

DEPARTMENT OF : Computer Science and Engineering

NAME OF THE LABORATORY : DSCC LAB

Name : ___________________ Roll No : 1602-19-733-____ Page No: ___

Lab Experiment

Hosting a Static Website

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab to launch your lab.

   A **Start Lab** panel opens, and it displays the lab status.

2. Wait until the **Start Lab** panel displays the message *Lab status: ready*, then close the panel by choosing the **X**.
3. At the top of these instructions, choose AWS.

   This action opens the AWS Management Console in a new browser tab. The system automatically logs you in.

4. Arrange the **AWS Management Console** tab so that it displays alongside these instructions. Ideally, you will have both browser tabs open at the same time so that you can follow the lab steps more easily.

   Do not change the Region unless specifically instructed to do so.

Task 1: Creating a bucket in Amazon S3

In this task, you will create an S3 bucket and configure it for static website hosting.

In the **AWS Management Console**, on the **Services** menu, choose **S3**.

5. Choose **Create bucket**

   An S3 bucket name is globally unique, and the namespace is shared by all AWS accounts. After you create a bucket, the name of that bucket cannot be used by another AWS account in any AWS Region unless you delete the bucket.

   Thus, for this lab, you will use a bucket name that includes a random number, such as: *website-123*

6. For **Bucket name**, enter: `website-<123>` (replace <*123*> with a random number)

   Public access to buckets is blocked by default. Because the files in your static website will need to be accessible through the internet, you must permit public access.

   - Verify the **AWS Region** is set to **us-east-1** (if it is not, choose the us-east-1 Region)
7. In the **Object Ownership** section, select **ACLs enabled**, then verify **Bucket owner preferred** is selected.

8. Clear **Block all public access**, then select the box that states **I acknowledge that the current settings may result in this bucket and the objects within becoming public**.
9. Choose **Create bucket**.

   You can use tags to add additional information to a bucket, such as a project code, cost centre, or owner.

10. Choose the name of your new bucket.
11. Choose the **Properties** tab.
12. Scroll to the **Tags** panel.
13. Choose **Edit** then **Add tag** and enter:

- **Key:** Department
- **Value:** Marketing

14. Choose **Save changes** to save the tag.

    Next, you will configure the bucket for static website hosting.

15. Stay in the **Properties** console.
16. Scroll to the **Static website hosting** panel.
17. Choose **Edit**
18. Configure the following settings:
    - **Static web hosting:** Enable
    - **Hosting type:** Host a static website
    - **Index document:** index.html
      - **Note**: You must enter this value, even though it is already displayed.
    - **Error document:** error.html
19. Choose **Save changes**
20. In the **Static website hosting** panel, choose the link under **Bucket website endpoint**.

    You will receive a *403 Forbidden* message because the bucket permissions have not been configured yet. Keep this tab open in your web browser so that you can return to it later.

    Your bucket has now been configured to host a static website.

Task 2: Uploading content to your bucket

In this task, you will upload the files that will serve as your static website to the bucket.

21. Right-click each of these links and download the files to your computer:

    Ensure that each file keeps the same file name, including the extension.

    - index.html

- script.js
- style.css

22. Return to the Amazon S3 console and in the website-<123> bucket you created earlier, choose the **Objects** tab.
23. Choose **Upload.**
24. Choose **Add files**
25. Locate and select the three files that you downloaded.
26. If prompted, choose I acknowledge that existing objects with the same name will be overwritten.
27. Choose **Upload.**

    Your files are uploaded to the bucket.

    - Choose **Close**

Task 3: Enabling access to the objects

Objects that are stored in Amazon S3 are private by default. This ensures that your organization's data remains secure.

In this task, you will make the uploaded objects publicly accessible.

First, confirm that the objects are currently private.

28. Return to the browser tab that showed the *403 Forbidden* message.
29. Refresh the webpage

    You should still see a *403 Forbidden* message.

    Analysis: This response is expected! This message indicates that your static website is being hosted by Amazon S3, but that the content is private.

    You can make Amazon S3 objects public through two different ways:

    - To make either a whole bucket public, or a specific directory in a bucket public, use a *bucket policy*.
    - To make individual objects in a bucket public, use an *access control list (ACL)*.
30. Return to the web browser tab with the Amazon S3 console (but do not close the website tab).
31. Select all three objects.
32. In the **Actions** menu, choose **Make public via ACL**.

    A list of the three objects is displayed.

33. Choose **Make public**

Your static website is now publicly accessible.

34. Return to the web browser tab that has the *403 Forbidden* message.
35. Refresh the webpage.

You should now see the static website that is being hosted by Amazon S3.

Task 4: Updating the website

You can change the website by editing the HTML file and uploading it again to the S3 bucket.

36. On your computer, load the **index.html** file into a text editor (for example, Notepad or TextEdit).
37. Find the text **Served from Amazon S3** and replace it with Created by <YOUR-NAME>, substituting your name for <*YOUR-NAME*> (for example, *Created by Jane*).
38. Save the file.
39. Return to the Amazon S3 console and upload the **index.html** file that you just edited.
40. Select **index.html** and use the **Actions** menu to choose the **Make public via ACL** option again.
41. Return to the web browser tab with the static website and refresh the page.

    Your name should now be on the page.

Your static website is now accessible on the internet. Because it is hosted on Amazon S3, the website has high availability and can serve high volumes of traffic without using any servers.

You can also use your own domain name to direct users to a static website that is hosted on Amazon S3. To accomplish this, you could use the Amazon Route 53 Domain Name System (DNS) service in combination with Amazon S3.

Submitting your work

42. At the top of these instructions, choose **Submit** to record your progress and when prompted, choose **Yes**.
43. If the results don't display after a couple of minutes, return to the top of these instructions, and choose Grades
44. To find detailed feedback on your work, choose Details followed by **View Submission Report**.

Lab complete

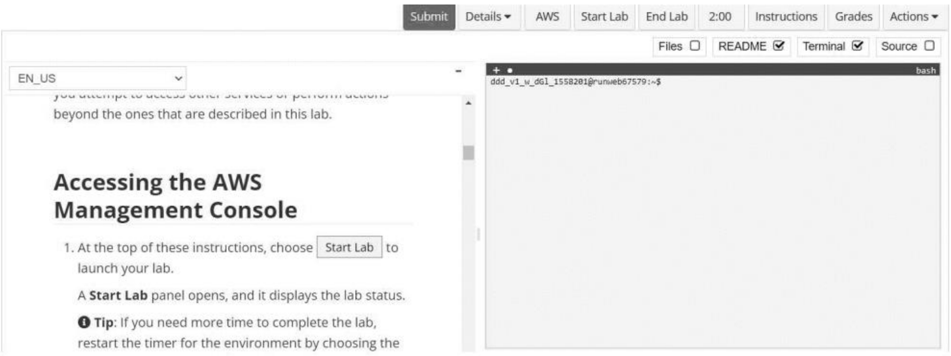
45. Choose End Lab at the top of this page, and then select **Yes** to confirm that you want to end the lab.

    A panel indicates that *DELETE has been initiated... You may close this message box now.*

46. Select the **X** in the top right corner to close the panel.

OUTPUT SCREENSHOTS:

Instructions Panel



you attempt to access other services or perform actions beyond the ones that are described in this lab.

Accessing the AWS Management Console

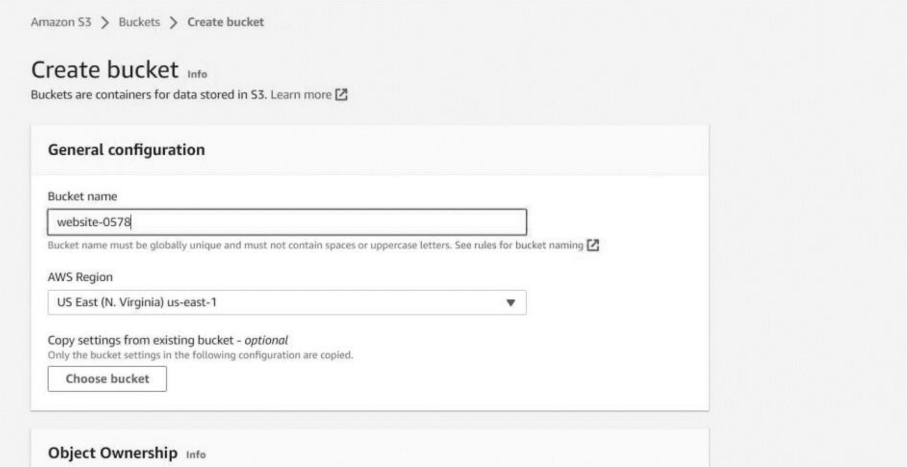
1. At the top of these instructions, choose Start Lab to launch your lab.

   A **Start Lab** panel opens, and it displays the lab status.

   Tip: If you need more time to complete the lab, restart the timer for the environment by choosing the

```
ddd_v1_w_dGl_1558201@runweb67579:~$
```

Bucket Creation



Create bucket Info

Buckets are containers for data stored in S3. Learn more

General configuration

Bucket name: website-0578

Bucket name must be globally unique and must not contain spaces or uppercase letters. See rules for bucket naming

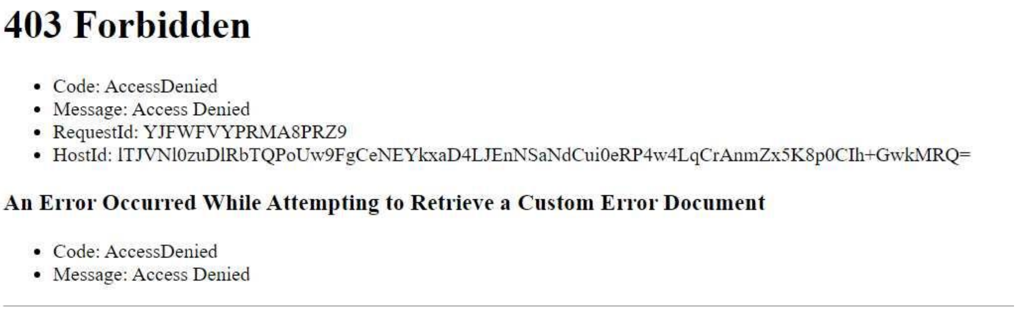
AWS Region: US East (N. Virginia) us-east-1

Copy settings from existing bucket - *optional*

Only the bucket settings in the following configuration are copied.

Object Ownership Info

Forbidden Error



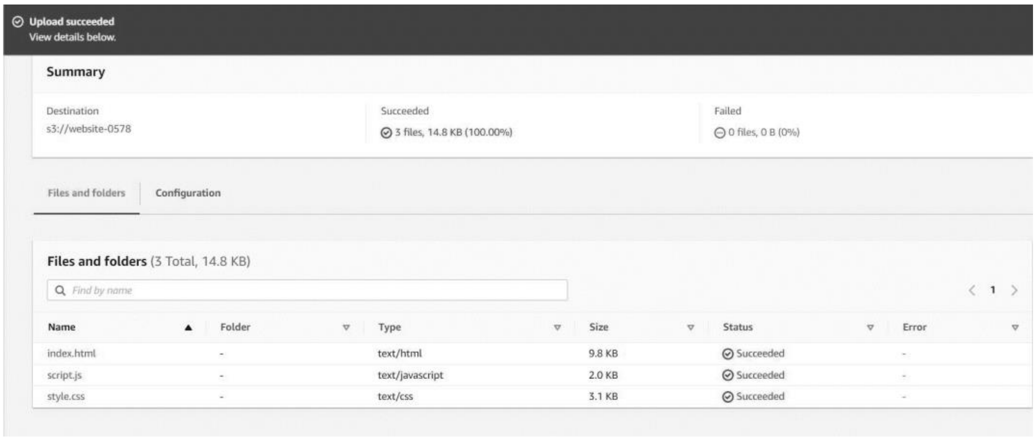
403 Forbidden

- Code: AccessDenied
- Message: Access Denied
- RequestId: YJFWFVYPRMA8PRZ9
- HostId: lTJVNl0zuDlRbTQPoUw9FgCeNEYkxaD4LJEnNSaNdCui0eRP4w4LqCrAnmZx5K8p0CIh+GwkMRQ=

An Error Occurred While Attempting to Retrieve a Custom Error Document

- Code: AccessDenied
- Message: Access Denied

Uploading Files



Upload succeeded
View details below.

Summary

Destination
s3://website-0578

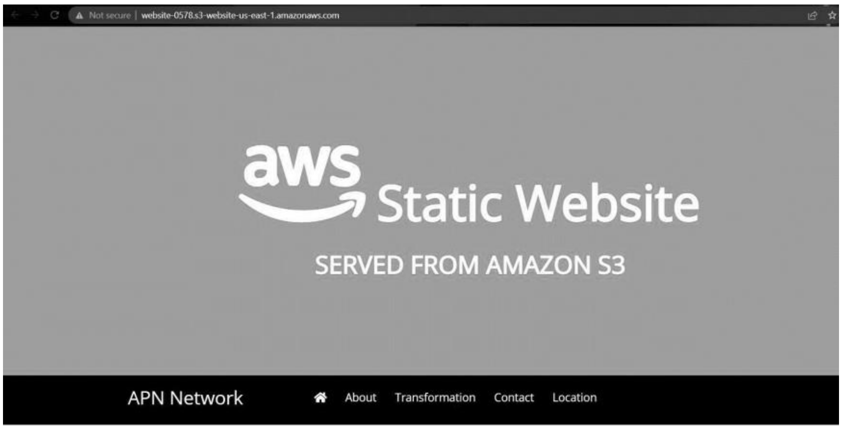
Succeeded
3 files, 14.8 KB (100.00%)

Failed
0 files, 0 B (0%)

Files and folders (3 Total, 14.8 KB)

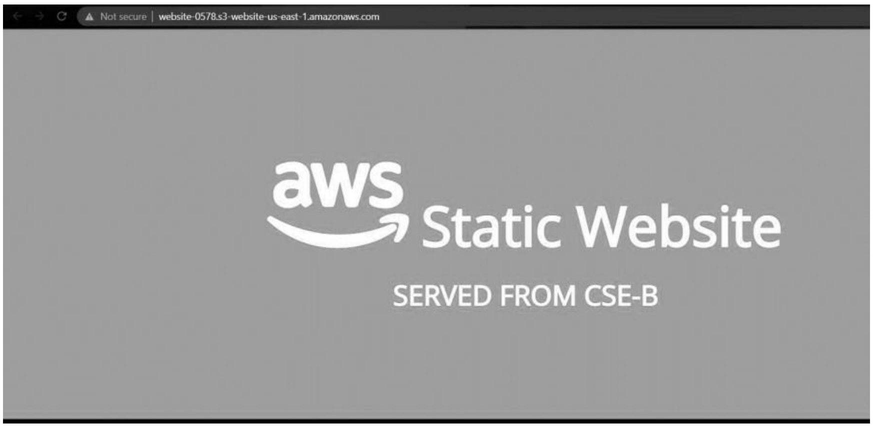
Name ▲	Folder ▽	Type ▽	Size ▽	Status ▽	Error ▽
index.html	-	text/html	9.8 KB	Succeeded	-
script.js	-	text/javascript	2.0 KB	Succeeded	-
style.css	-	text/css	3.1 KB	Succeeded	-

Static Website-1



Not secure | website-0578.s3-website-us-east-1.amazonaws.com

Static Website-2 (Updated)



Not secure | website-0578.s3-website-us-east-1.amazonaws.com

Lab Experiment

Introducing Amazon Elastic File System (Amazon EFS)

Accessing the AWS Management Console

1. At the top of these instructions, choose Start Lab to launch your lab.

   A **Start Lab** panel opens, and it displays the lab status.

   Tip: If you need more time to complete the lab, restart the timer for the environment by choosing the Start Lab button again.

2. Wait until the **Start Lab** panel displays the message *Lab status: ready*, then close the panel by choosing the **X**.
3. At the top of these instructions, choose AWS.
4. Arrange the **AWS Management Console** tab so that it displays alongside these instructions. Ideally, you will have both browser tabs open at the same time so that you can follow the lab steps more easily.

Task 1: Creating a security group to access your EFS file system

5. In the **AWS Management Console**, on the Services menu, choose **EC2**.
6. In the navigation pane on the left, choose **Security Groups**.
7. Copy the **Security group ID** of the *EFSClient* security group to your text editor.

   The Group ID should look similar to *sg-03727965651b6659b*.

8. Choose Create security group then configure:
   - **Security group name:** EFS Mount Target
   - **Description:** Inbound NFS access from EFS clients
   - **VPC:** *Lab VPC*
9. Under the **Inbound rules** section, choose **Add rule** then configure:
   - **Type:** *NFS*
   - **Source:**
     - *Custom*
     - In the *Custom* box, paste the security group's **Security group ID** that you copied to your text editor
   - Choose Create security group.

Task 2: Creating an EFS file system

10. On the Services menu, choose **EFS**.
11. Choose Create file system
12. In the **Create file system** window, choose **Customize**

13. On **Step 1**:
    - Uncheck Enable automatic backups.
    - **Lifecycle management:** Select *None*
    - In the **Tags** section, configure:
        - **Key:** Name
        - **Value:** My First EFS File System
14. Choose Next
15. For **VPC**, select *Lab VPC*.
16. Detach the default security group from each *Availability Zone* mount target by choosing the check box on each default security group.
17. Attach the **EFS Mount Target** security group to each *Availability Zone* mount target by:

- Selecting each **Security groups** check box.
- Choosing **EFS Mount Target**

A mount target is created for each subnet

18. Choose Next
19. On **Step 3**, choose Next
20. On **Step 4:**

- Review your configuration.
- Choose Create

Proceed to the next step after the **Mount target state** for each mount target changes to *Available*. Choose the screen refresh button after 2–3 minutes to check its progress.

Task 3: Connecting to your EC2 instance via SSH

In this task, you will connect to your EC2 instance by using Secure Shell (SSH).

21. Above these instructions that you are currently reading, choose the Details dropdown menu, and then select Show

A **Credentials** window opens.

22. Choose the **Download PPK** button and save the **labsuser.ppk** file.

Note: Typically, your browser saves the file to the **Downloads** directory.

23. Note the **EC2PublicIP** address if it is displayed.
24. Exit the **Details** panel by choosing the **X**.
25. To use SSH to access the EC2 instance, you must use ***PuTTY***. If you do not have PuTTY installed on your computer, download PuTTY.
26. Open **putty.exe**.
27. To keep the PuTTY session open for a longer period of time, configure the PuTTY timeout:

- Choose **Connection**
- **Seconds between keepalives**: 30

28. Configure your PuTTY session by using the following settings.

- Choose **Session**
- **Host Name (or IP address):** Paste the **EC2PublicIP** for the instance you noted earlier
    - Alternatively, return to the Amazon EC2 console and choose **Instances**
    - Select the instance you want to connect to
    - In the *Description* tab, copy the **IPv4 Public IP** value
- Back in PuTTY, in the **Connection** list, expand **SSH**
- Choose **Auth** (but don't expand it)
- Choose **Browse**
- Browse to the *labsuser.ppk* file that you downloaded, select it, and choose **Open**
- Choose **Open** again

29. To trust and connect to the host, choose **Yes**.
30. When you are prompted with **login as**, enter: `ec2-user.`

    This action connects you to the EC2 instance.

Task 4: Creating a new directory and mounting the EFS file system

31. In your SSH session, make a new directory by entering `sudo mkdir efs`
32. Back in the **AWS Management Console**, on the Services menu, choose **EFS**.
33. Choose **My First EFS File System**.
34. In the **Amazon EFS Console**, on the top right corner of the page, choose Attach to open the Amazon EC2 mount instructions.
35. Copy the entire command in the **Using the NFS client** section.

    The mount command should look similar to this example:

    ```
    sudo mount -t nfs4 -o nfsvers=4.1,rsize=1048576,wsize=1048576,hard,timeo=600,retrans=2,noresvport fs-bce57914.efs.us-west-2.amazonaws.com:/ efs
    ```

    The provided `sudo mount...` command uses the default Linux mount options.

36. In your Linux SSH session, mount your Amazon EFS file system by:
    - Pasting the command
    - Pressing ENTER

37. Get a full summary of the available and used disk space usage by entering:

    ```
    sudo df -hT
    ```

Task 5: Examining the performance behavior of your new EFS file system

38. Examine the write performance characteristics of your file system by entering:

```
sudo fio --name=fio-efs --filesize=10G --filename=./efs/fio-efs-test.img --bs=1M --nrfiles=1 --direct=1 --sync=0 --rw=write --iodepth=200 --ioengine=libaio
```

Monitoring performance by using Amazon CloudWatch

39. In the **AWS Management Console**, on the Services menu, choose **CloudWatch**.
40. In the navigation pane on the left, choose **Metrics**.
41. In the **All-metrics** tab, choose **EFS**.
42. Choose **File System Metrics**.
43. Select the row that has the **PermittedThroughput** Metric Name.

    You might need to wait 2–3 minutes and refresh the screen several times before all available metrics, including **PermittedThroughput**, calculate and populate.

44. On the graph, choose and drag around the data line. If you do not see the line graph, adjust the time range of the graph to display the period during which you ran the `fio` command.

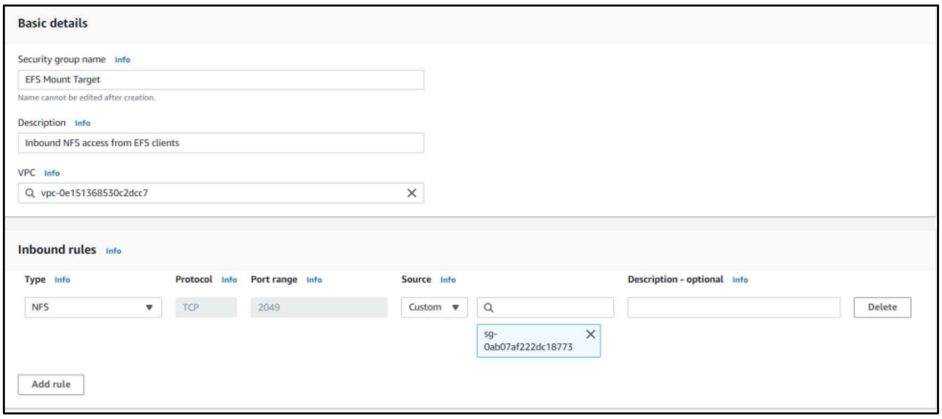
45. Pause your pointer on the data line in the graph. The value should be *105M*.
46. In the **All-metrics** tab, *uncheck* the box for **PermittedThroughput**.
47. Select the check box for **DataWriteIOBytes**.

    If you do not see *DataWriteIOBytes* in the list of metrics, use the **File System Metrics** search to find it.

48. Choose the **Graphed metrics** tab.
49. On the **Statistics** column, select **Sum**.
50. On the **Period** column, select **1 Minute**.
51. Pause your pointer on the peak of the line graph. Take this number (in bytes) and divide it by the duration in seconds (60 seconds). The result gives you the write throughput (B/s) of your file system during your test.

OUTPUT SCREENSHOTS:

Task-1: Creating a security group to access your EFS file system



Basic details

Security group name Info
EFS Mount Target
Name cannot be edited after creation.

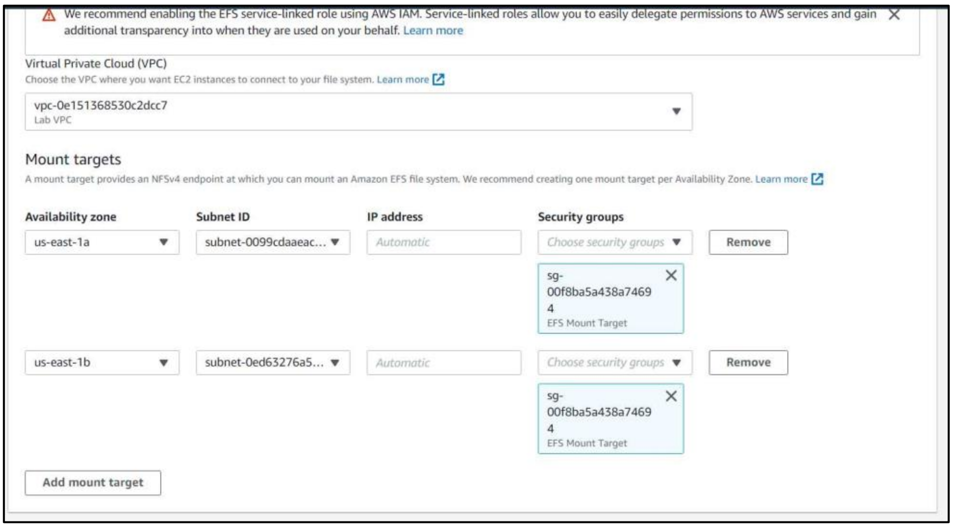
Description Info
Inbound NFS access from EFS clients

VPC Info
vpc-0e151368530c2dcc7

Inbound rules Info

Type	Protocol	Port range	Source		Description - optional
NFS	TCP	2049	Custom	sg-0ab07af222dc18773	

Task-2: Creating an EFS file system

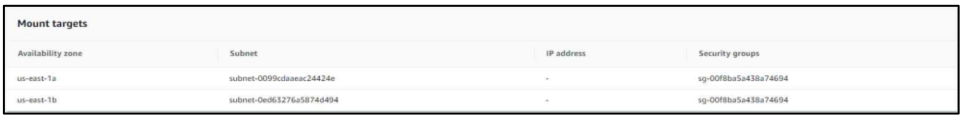


We recommend enabling the EFS service-linked role using AWS IAM. Service-linked roles allow you to easily delegate permissions to AWS services and gain additional transparency into when they are used on your behalf. Learn more

Virtual Private Cloud (VPC)
Choose the VPC where you want EC2 instances to connect to your file system. Learn more
vpc-0e151368530c2dcc7
Lab VPC

Mount targets
A mount target provides an NFSv4 endpoint at which you can mount an Amazon EFS file system. We recommend creating one mount target per Availability Zone. Learn more

Availability zone	Subnet ID	IP address	Security groups
us-east-1a	subnet-0099cdaaeac...	Automatic	sg-00f8ba5a438a74694 EFS Mount Target
us-east-1b	subnet-0ed63276a5...	Automatic	sg-00f8ba5a438a74694 EFS Mount Target

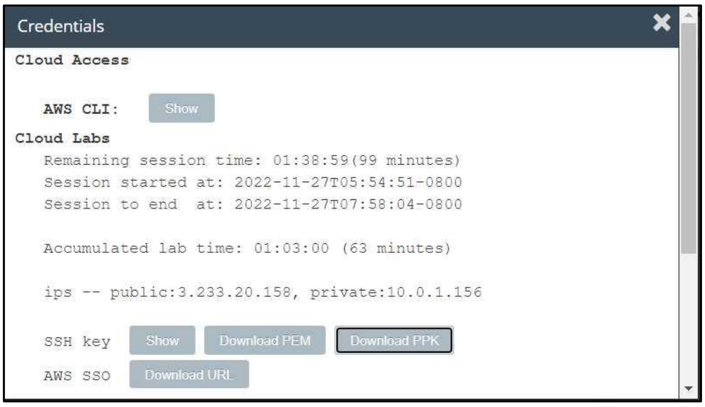


Mount targets

Availability zone	Subnet	IP address	Security groups
us-east-1a	subnet-0099cdaaeac24424e	-	sg-00f8ba5a438a74694
us-east-1b	subnet-0ed63276a5874d494	-	sg-00f8ba5a438a74694

Task-3:

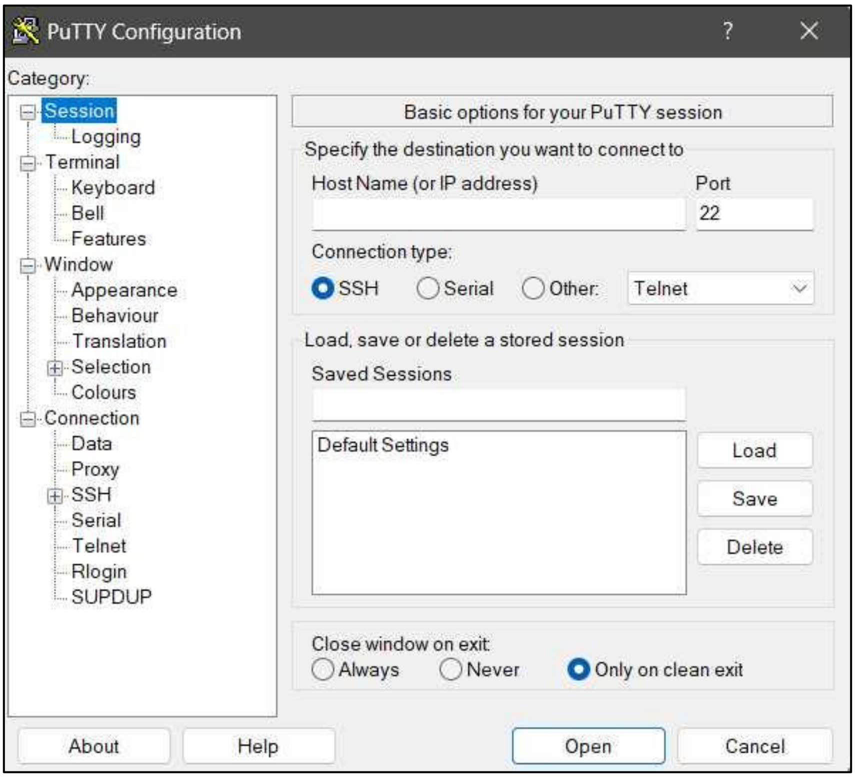
Credentials Tab



Credentials

```
Cloud Access

  AWS CLI:  Show
Cloud Labs
  Remaining session time: 01:38:59(99 minutes)
  Session started at: 2022-11-27T05:54:51-0800
  Session to end  at: 2022-11-27T07:58:04-0800

  Accumulated lab time: 01:03:00 (63 minutes)

  ips -- public:3.233.20.158, private:10.0.1.156

  SSH key  Show  Download PEM  Download PPK
  AWS SSO  Download URL
```

Putty Config



PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

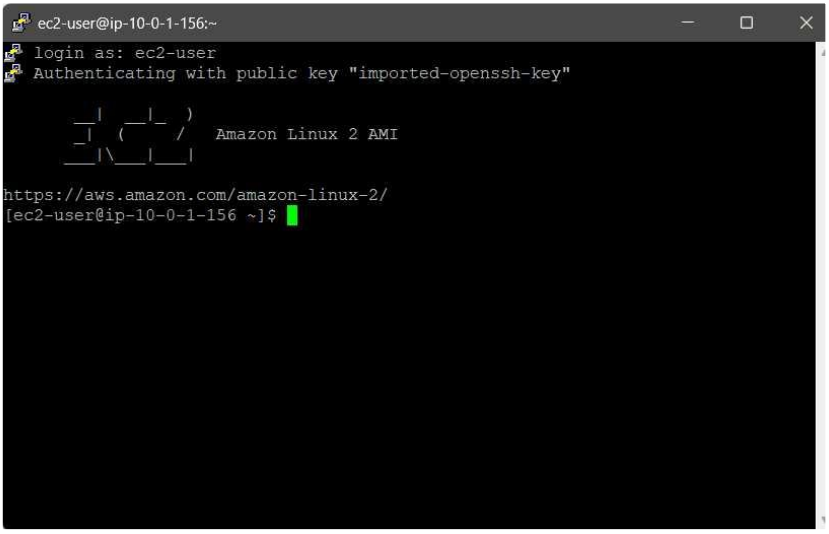
Connection type: ● SSH ○ Serial ○ Other: Telnet

Load, save or delete a stored session

Saved Sessions:

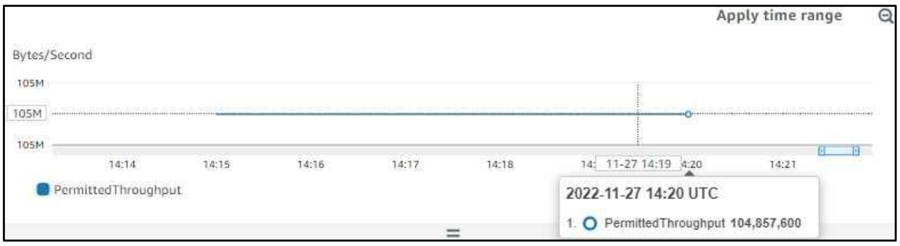
Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit



```
login as: ec2-user
Authenticating with public key "imported-openssh-key"

       __|  __|_  )
       _|  (     /   Amazon Linux 2 AMI
      ___|\___|___|

https://aws.amazon.com/amazon-linux-2/
[ec2-user@ip-10-0-1-156 ~]$
```

Task-5: Examining the performance behavior of your new EFS file system



LAB PROGRAM

Experiment: Deploying a Node.js Web Application on AWS

HARDWARE REQUIREMENTS: Core I5 Processor, 4 GB RAM, 40GB HDD

SOFTWARE REQUIREMENTS: Amazon AWS, EC2, VS Code/Eclipse, Node, NPM, GIT, Putty

Description:

Node.js is a JavaScript runtime environment that allows one to run JS on the server. It is built on the open-source V8 JavaScript engine used in Chrome and written in C++ which executes JS in a standalone environment.

In this experiment, we clone a Nodejs application from GITHUB and deploy this application on to Amazon EC2 instance, make it available over Amazon AWS URI.

Steps to configure EC2 Instance :

1. Create an EC2 instance and Launch it:

Choose amazon Ec2 instance machine image as Ubuntu 18.04 64 bit with type of micro.

(Login to AwsAcademy,

LMS-Dashboard - AWS Academy Learner Lab – Educator

Click on Modules

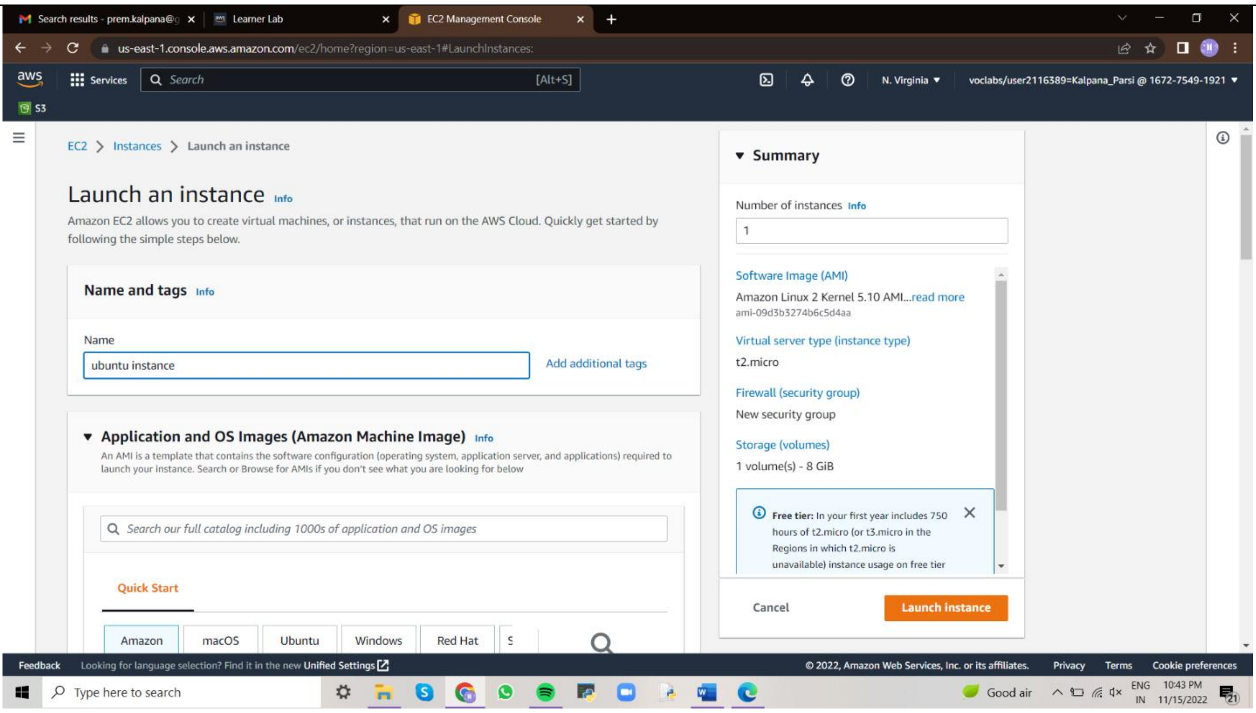
Click on Learner Lab

Click on Start Lab

Click on AWS

Services – EC2

EC2 – Instances – Launch an instance


Search results - prem.kalpana@
Learner Lab
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search
[Alt+S]
N. Virginia ▼
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921 ▼
S3
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
Good air
ENG IN
10:43 PM 11/15/2022

EC2 > Instances > Launch an 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: ubuntu instance

Add additional tags

▼ 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

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon | macOS | Ubuntu | Windows | Red Hat | S

▼ Summary

Number of instances Info: 1

Software Image (AMI)
Amazon Linux 2 Kernel 5.10 AMI...read more
ami-09d3b3274b6c5d4aa

Virtual server type (instance type)
t2.micro

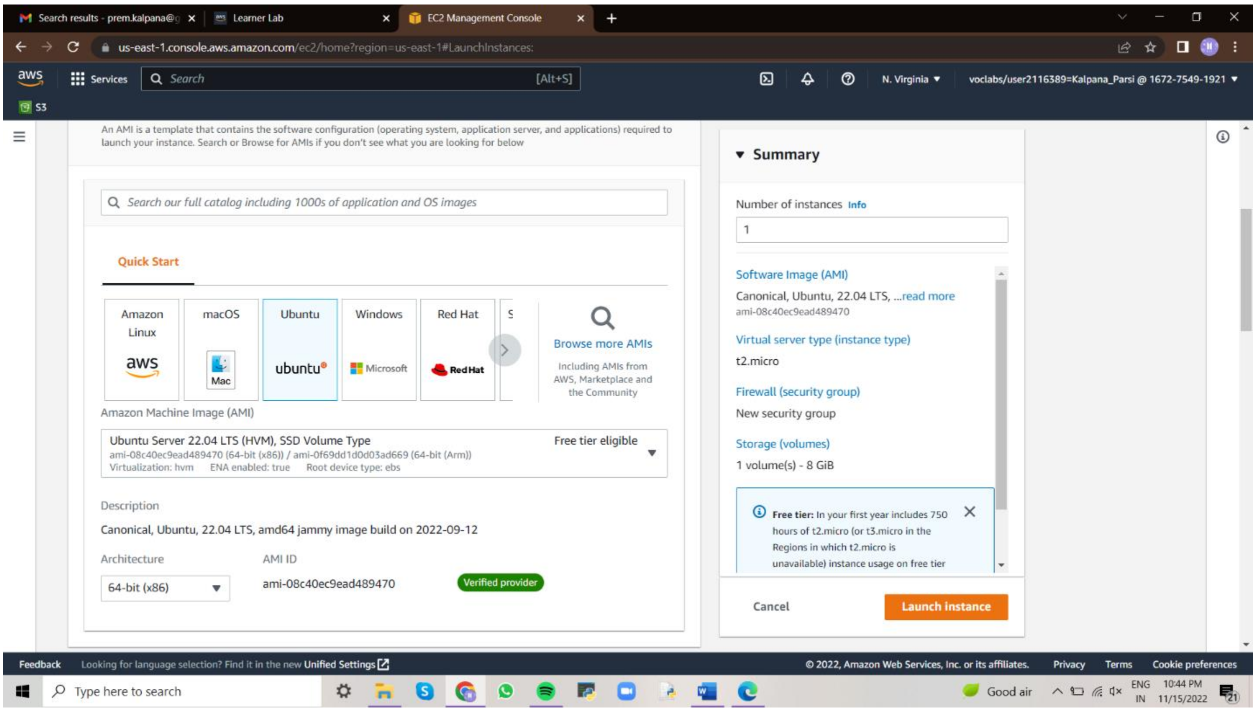
Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier

Cancel | Launch instance

Select Amazon Ubuntu


Search results - prem.kalpana@
Learner Lab
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search
[Alt+S]
N. Virginia ▼
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921 ▼
S3
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
Good air
ENG IN
10:44 PM 11/15/2022

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

Search our full catalog including 1000s of application and OS images

Quick Start

Amazon Linux (aws) | macOS (Mac) | Ubuntu (ubuntu) | Windows (Microsoft) | Red Hat (Red Hat) | S

Browse more AMIs
Including AMIs from AWS, Marketplace and the Community

Amazon Machine Image (AMI)

Ubuntu Server 22.04 LTS (HVM), SSD Volume Type — Free tier eligible
ami-08c40ec9ead489470 (64-bit (x86)) / ami-0f69dd1d0d03ad669 (64-bit (Arm))
Virtualization: hvm ENA enabled: true Root device type: ebs

Description
Canonical, Ubuntu, 22.04 LTS, amd64 jammy image build on 2022-09-12

Architecture: 64-bit (x86) ▼

AMI ID: ami-08c40ec9ead489470 Verified provider

▼ Summary

Number of instances Info: 1

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...read more
ami-08c40ec9ead489470

Virtual server type (instance type)
t2.micro

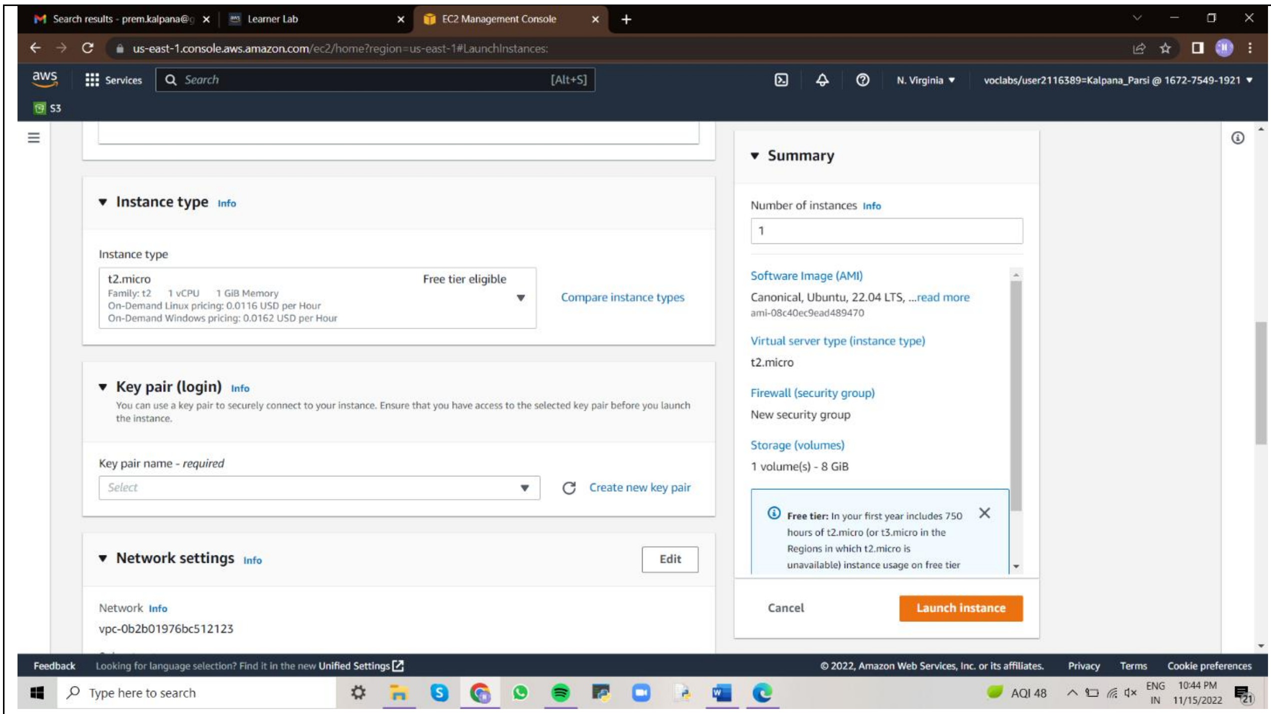
Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier

Cancel | Launch instance

Instance type - t2.micro)


Search results - prem.kalpana@g
Learner Lab
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921
S3
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
AQI 48
ENG IN
10:44 PM 11/15/2022

Instance type Info

Instance type

t2.micro Free tier eligible
Family: t2 1 vCPU 1 GiB Memory
On-Demand Linux pricing: 0.0116 USD per Hour
On-Demand Windows pricing: 0.0162 USD per Hour

Compare instance types

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

Network settings Info Edit

Network Info

vpc-0b2b01976bc512123

Summary

Number of instances Info

1

Software Image (AMI)

Canonical, Ubuntu, 22.04 LTS, ...read more

ami-08c40ec9ead489470

Virtual server type (instance type)

t2.micro

Firewall (security group)

New security group

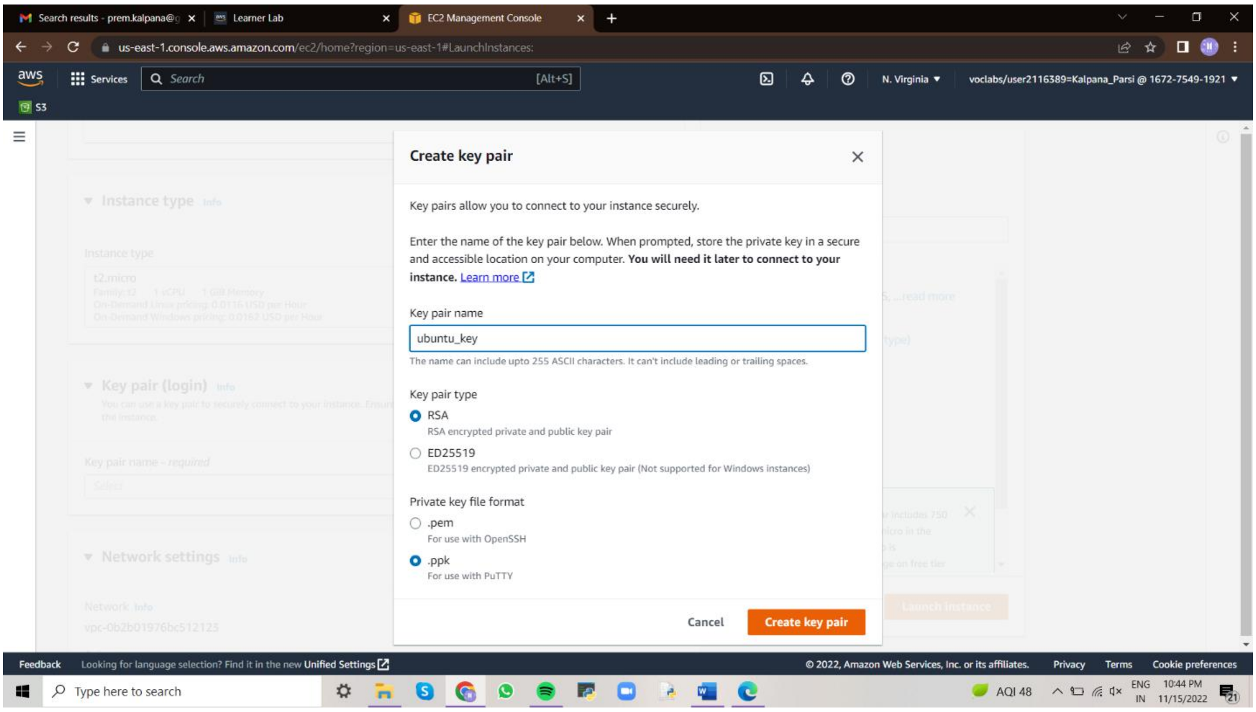
Storage (volumes)

1 volume(s) - 8 GiB

Free tier: In your first year includes 750 hours of t2.micro (or t3.micro in the Regions in which t2.micro is unavailable) instance usage on free tier

Cancel Launch instance

Create new key pair – Save the key pair as .ppk (to work with putty)


Search results - prem.kalpana@g
Learner Lab
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921
S3
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
Type here to search
AQI 48
ENG IN
10:44 PM 11/15/2022

Create key pair

Key pairs allow you to connect to your instance securely.

Enter the name of the key pair below. 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

Key pair name

ubuntu_key

The name can include upto 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 (Not supported for Windows instances)

Private key file format

○ .pem
For use with OpenSSH

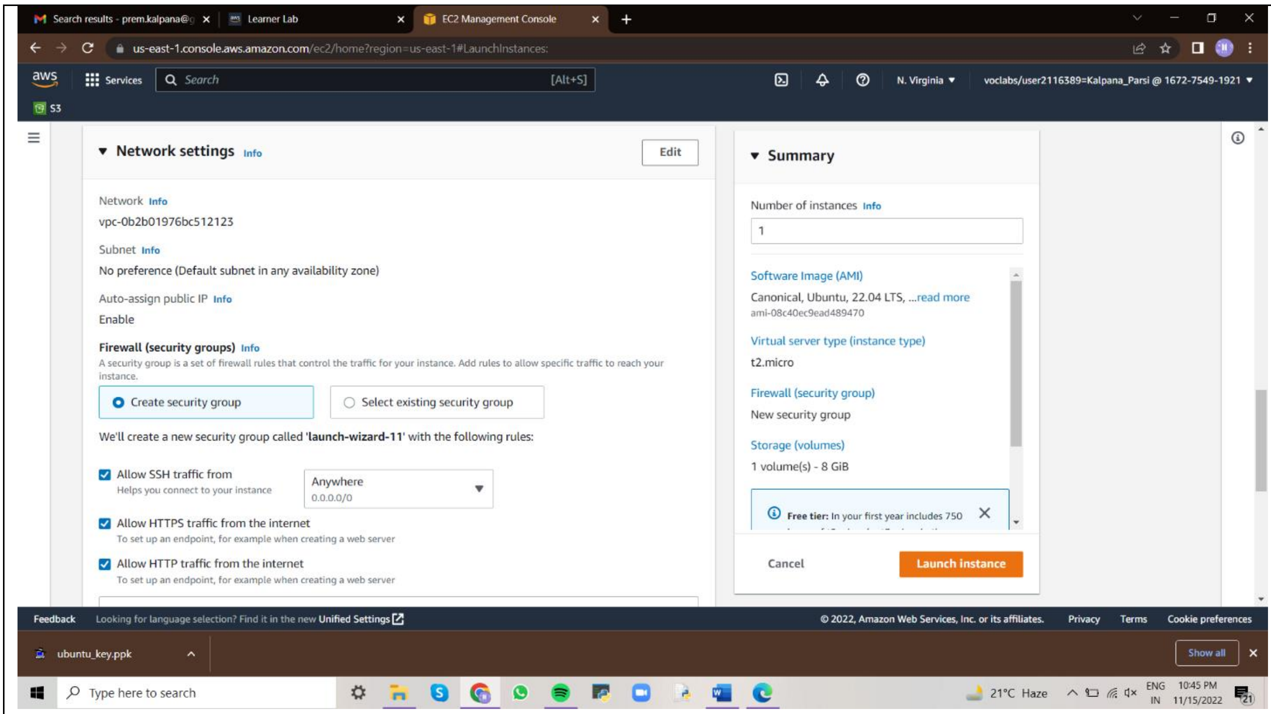
● .ppk
For use with PuTTY

Cancel Create key pair

Next Add storage

Next configure Security Group – Create security group.

In this step we need to allow http and https requests to access from any group.


Search results - prem.kalpana@g
Learner Lab
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921
S3
Edit
Cancel
Launch instance
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ubuntu_key.ppk
Show all
Type here to search
21°C Haze
ENG IN
10:45 PM 11/15/2022

▼ Network settings Info

Network Info
vpc-0b2b01976bc512123

Subnet Info
No preference (Default subnet in any availability zone)

Auto-assign public IP Info
Enable

Firewall (security groups) Info
A security group is a set of firewall rules that control the traffic for your instance. Add rules to allow specific traffic to reach your instance.

● Create security group ○ Select existing security group

We'll create a new security group called 'launch-wizard-11' with the following rules:

☑ Allow SSH traffic from — Anywhere 0.0.0.0/0
Helps you connect to your instance

☑ Allow HTTPS traffic from the internet
To set up an endpoint, for example when creating a web server

☑ Allow HTTP traffic from the internet
To set up an endpoint, for example when creating a web server

▼ Summary

Number of instances Info
1

Software Image (AMI)
Canonical, Ubuntu, 22.04 LTS, ...read more
ami-08c40ec9ead489470

Virtual server type (instance type)
t2.micro

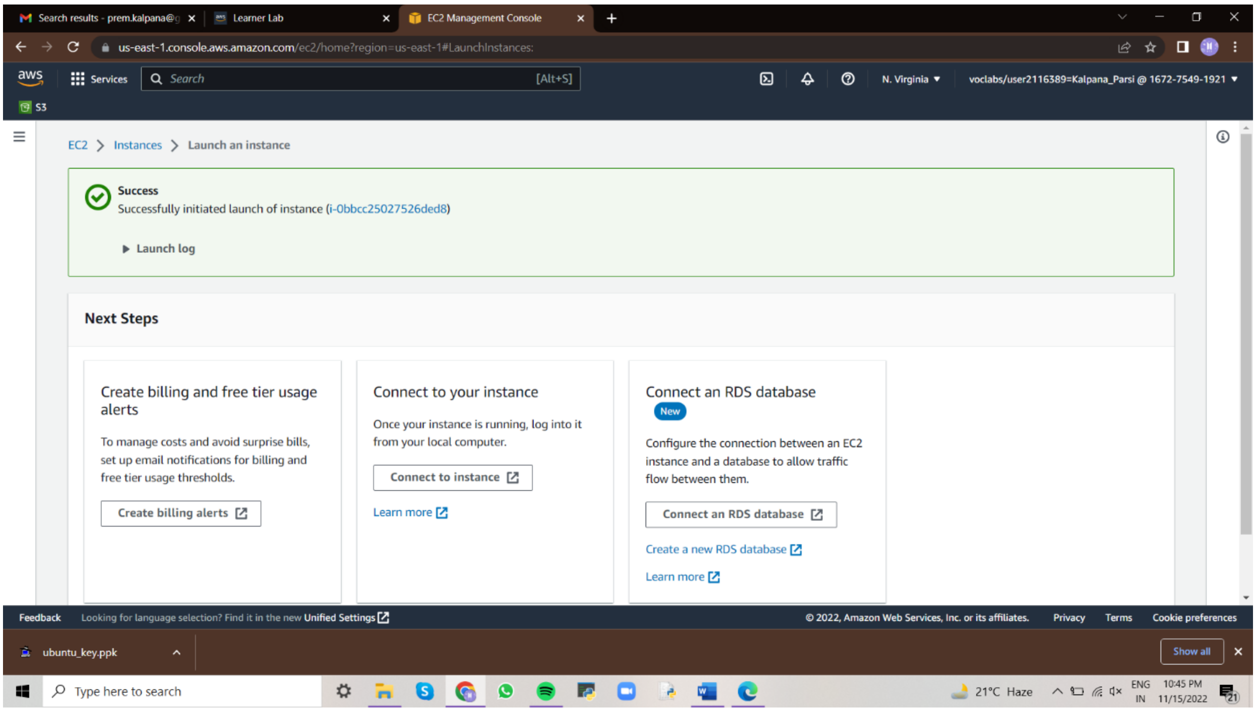
Firewall (security group)
New security group

Storage (volumes)
1 volume(s) - 8 GiB

Free tier: In your first year includes 750

Finally click on Launch instance.

We can see instance is launched successfully.


Search results - prem.kalpana@g
Learner Lab
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921
S3
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ubuntu_key.ppk
Show all
Type here to search
21°C Haze
ENG IN
10:45 PM 11/15/2022

EC2 > Instances > Launch an instance

Success
Successfully initiated launch of instance (i-0bbcc25027526ded8)

▶ Launch log

Next Steps

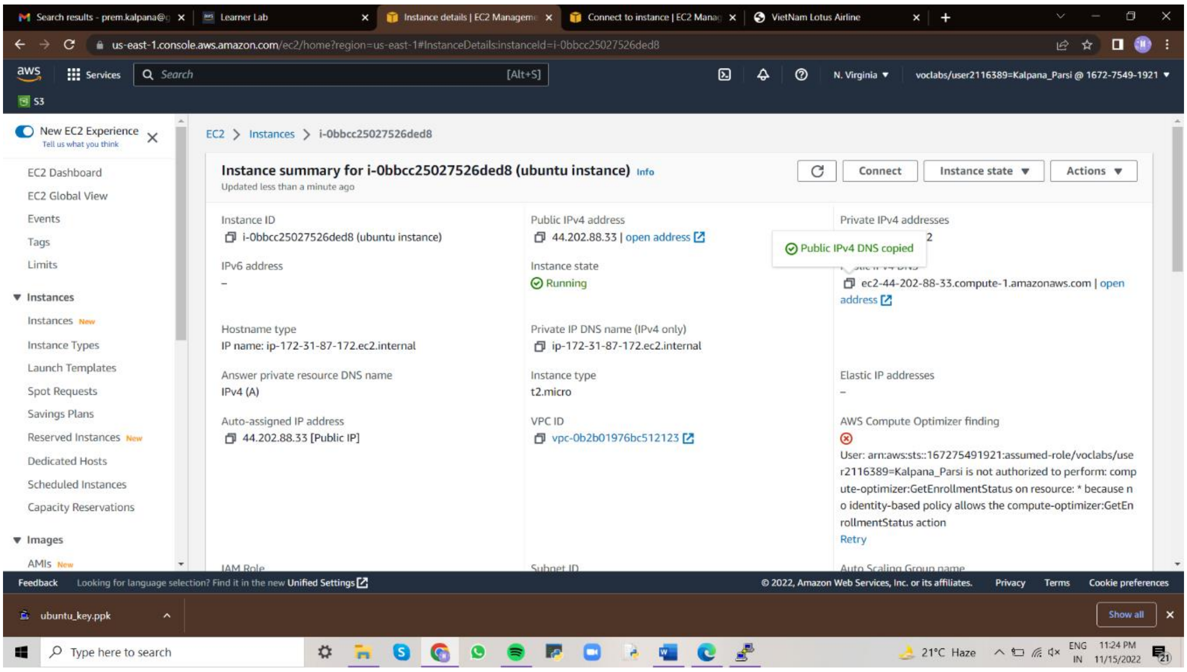
Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database New
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

When the instance state is running , it indicates that your instance was created successfully.

Copy the public DNS of your Instance. You can access different app running on your instance at a different port.



Instance summary for i-0bbcc25027526ded8 (ubuntu instance) Info
Updated less than a minute ago

Instance ID
i-0bbcc25027526ded8 (ubuntu instance)

Public IPv4 address
44.202.88.33 | open address

Private IPv4 addresses
2

Public IPv4 DNS copied

IPv6 address
–

Instance state
Running

Public IPv4 DNS
ec2-44-202-88-33.compute-1.amazonaws.com | open address

Hostname type
IP name: ip-172-31-87-172.ec2.internal

Private IP DNS name (IPv4 only)
ip-172-31-87-172.ec2.internal

Answer private resource DNS name
IPv4 (A)

Instance type
t2.micro

Elastic IP addresses
–

Auto-assigned IP address
44.202.88.33 [Public IP]

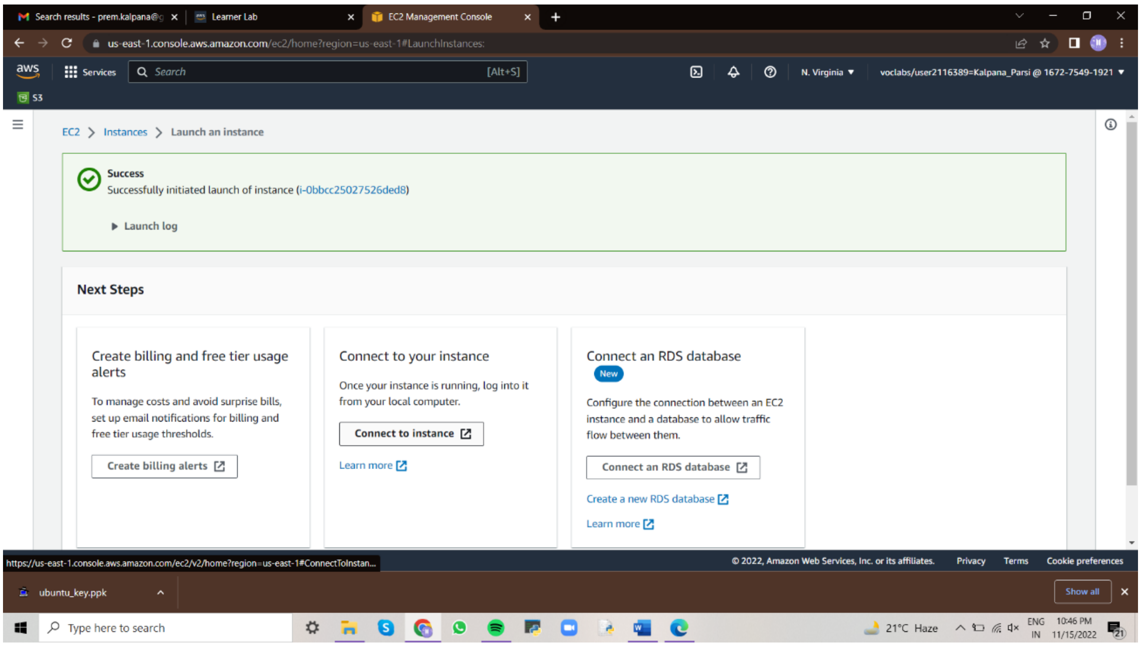
VPC ID
vpc-0b2b01976bc512123

AWS Compute Optimizer finding
User: arn:aws:sts::167275491921:assumed-role/voclabs/user2116389=Kalpana_Parsi is not authorized to perform: compute-optimizer:GetEnrollmentStatus on resource: * because no identity-based policy allows the compute-optimizer:GetEnrollmentStatus action
Retry

IAM Role
Subnet ID
Auto Scaling Group name

2. Connect to your Instance:

Click on launch instance then it shows popup window giving details how to connect to your instance.



Search results - prem.kalpana@
Learner Lab
EC2 Management Console
us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#LaunchInstances:
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921
S3
EC2 > Instances > Launch an instance
https://us-east-1.console.aws.amazon.com/ec2/v2/home?region=us-east-1#ConnectToInstan...
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ubuntu_key.ppk
Show all
Type here to search
21°C Haze
ENG IN
10:46 PM 11/15/2022

Success
Successfully initiated launch of instance (i-0bbcc25027526ded8)

Launch log

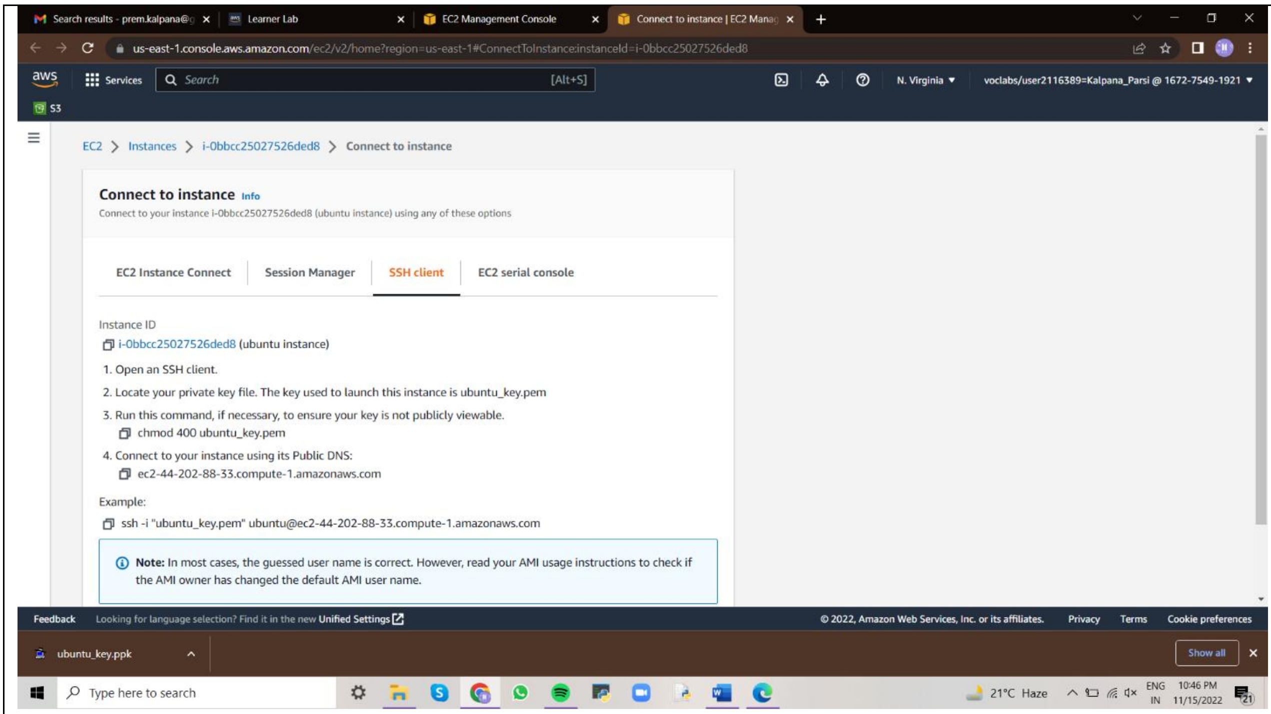
Next Steps

Create billing and free tier usage alerts
To manage costs and avoid surprise bills, set up email notifications for billing and free tier usage thresholds.
Create billing alerts

Connect to your instance
Once your instance is running, log into it from your local computer.
Connect to instance
Learn more

Connect an RDS database New
Configure the connection between an EC2 instance and a database to allow traffic flow between them.
Connect an RDS database
Create a new RDS database
Learn more

To open SSH client and If we are in windows platform we need to launch the instance with the help of putty soft.


Search results - prem.kalpana@g
Learner Lab
EC2 Management Console
Connect to instance | EC2 Manag
us-east-1.console.aws.amazon.com/ec2/v2/home?region=us-east-1#ConnectToInstance:instanceId=i-0bbcc25027526ded8
aws
Services
Search
[Alt+S]
N. Virginia
voclabs/user2116389=Kalpana_Parsi @ 1672-7549-1921
S3
EC2 > Instances > i-0bbcc25027526ded8 > Connect to instance
Feedback
Looking for language selection? Find it in the new Unified Settings
© 2022, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences
ubuntu_key.ppk
Show all
Type here to search
21°C Haze
ENG IN
10:46 PM
11/15/2022

Connect to instance Info

Connect to your instance i-0bbcc25027526ded8 (ubuntu instance) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID

i-0bbcc25027526ded8 (ubuntu instance)

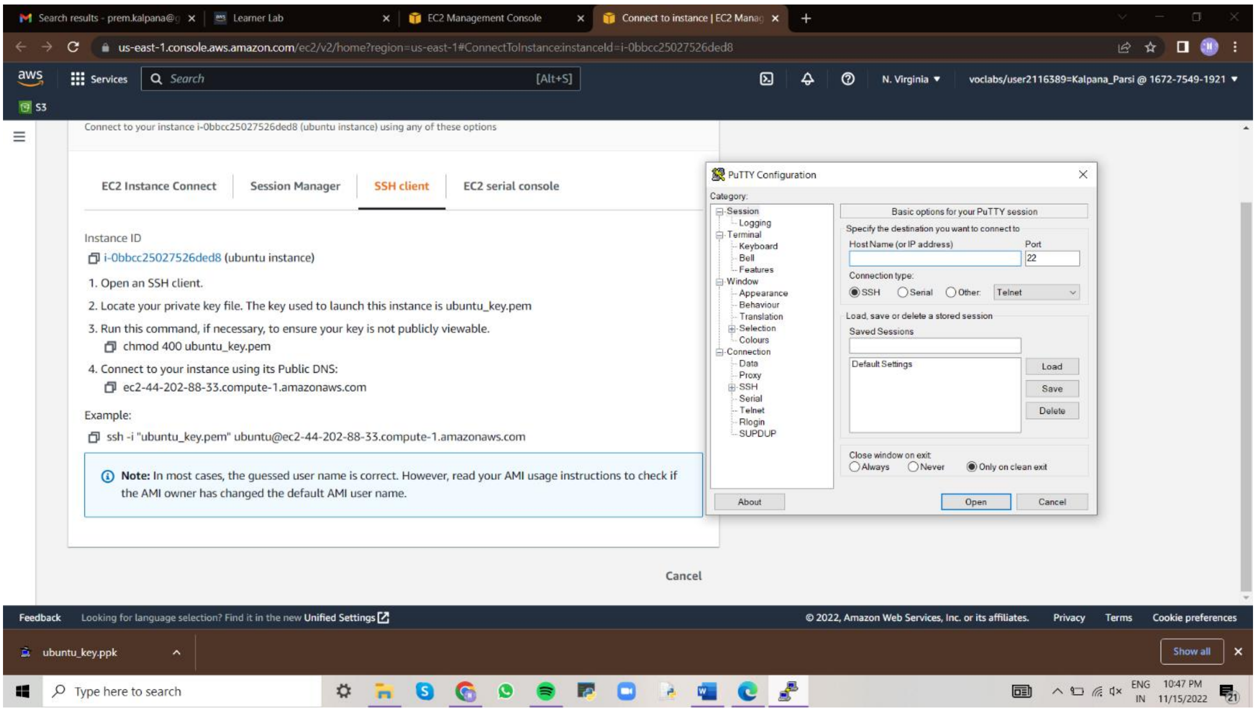
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ubuntu_key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   ```
   chmod 400 ubuntu_key.pem
   ```
4. Connect to your instance using its Public DNS:
   ```
   ec2-44-202-88-33.compute-1.amazonaws.com
   ```

Example:

```
ssh -i "ubuntu_key.pem" ubuntu@ec2-44-202-88-33.compute-1.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

Open Putty



Connect to your instance i-0bbcc25027526ded8 (ubuntu instance) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID

i-0bbcc25027526ded8 (ubuntu instance)

1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ubuntu_key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
   ```
   chmod 400 ubuntu_key.pem
   ```
4. Connect to your instance using its Public DNS:
   ```
   ec2-44-202-88-33.compute-1.amazonaws.com
   ```

Example:

```
ssh -i "ubuntu_key.pem" ubuntu@ec2-44-202-88-33.compute-1.amazonaws.com
```

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

PuTTY Configuration

Basic options for your PuTTY session

Specify the destination you want to connect to

Host Name (or IP address):

Port: 22

Connection type: ● SSH ○ Serial ○ Other: Telnet

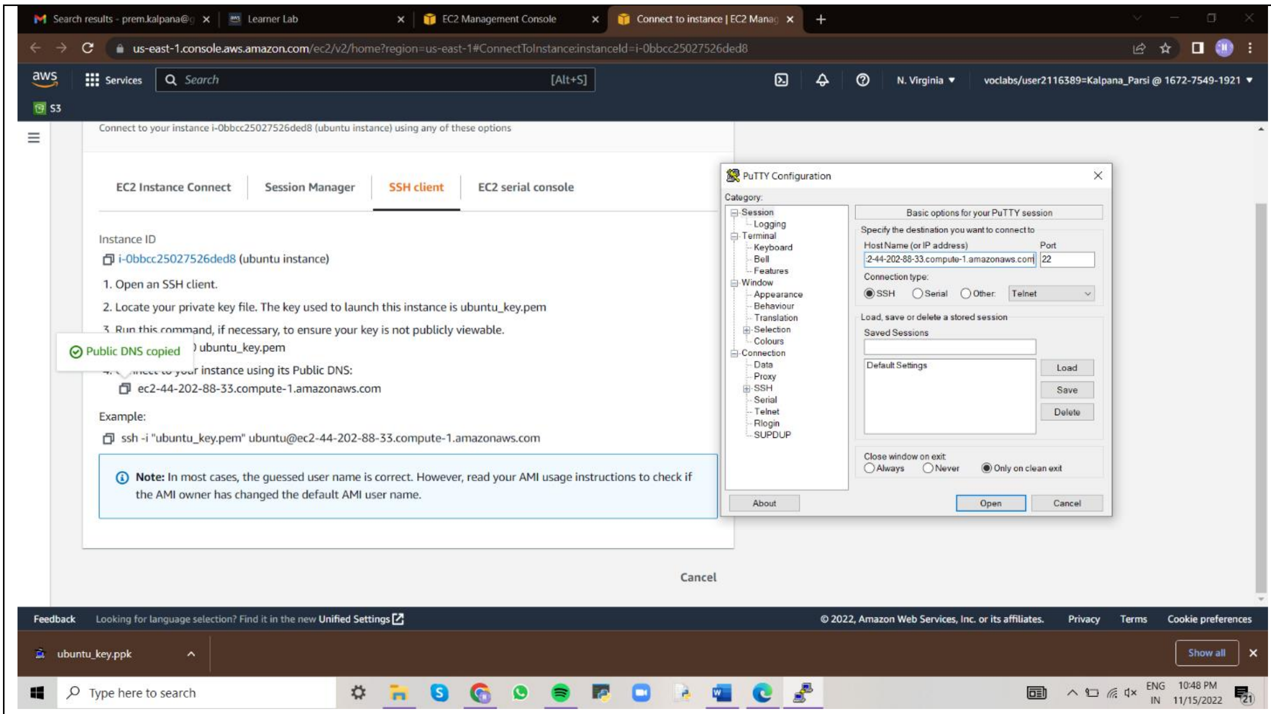
Load, save or delete a stored session

Saved Sessions

Default Settings

Close window on exit: ○ Always ○ Never ● Only on clean exit

Enter the Public DNS of your Instance in Host Name(IP address)



us-east-1.console.aws.amazon.com/ec2/v2/home?region=us-east-1#ConnectToInstance:instanceId=i-0bbcc25027526ded8

Connect to your instance i-0bbcc25027526ded8 (ubuntu instance) using any of these options

EC2 Instance Connect | Session Manager | SSH client | EC2 serial console

Instance ID
i-0bbcc25027526ded8 (ubuntu instance)

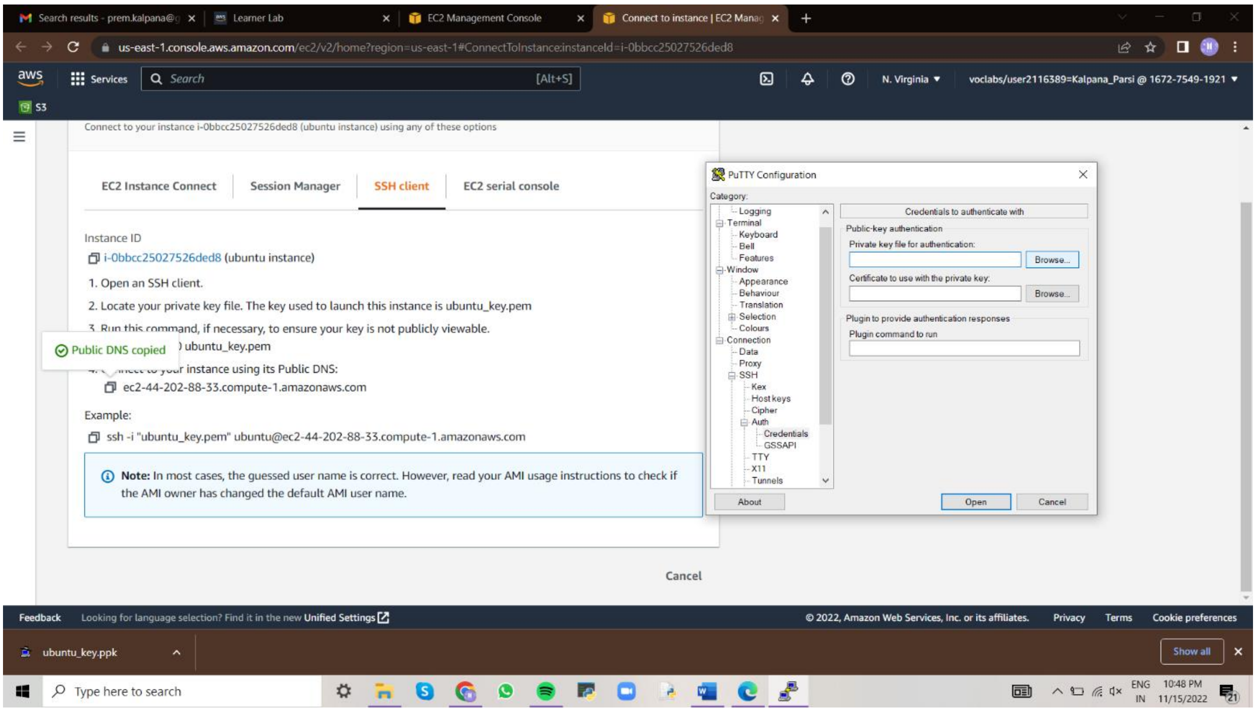
1. Open an SSH client.
2. Locate your private key file. The key used to launch this instance is ubuntu_key.pem
3. Run this command, if necessary, to ensure your key is not publicly viewable.
 ubuntu_key.pem
4. Connect to your instance using its Public DNS:
 ec2-44-202-88-33.compute-1.amazonaws.com

Example:
ssh -i "ubuntu_key.pem" ubuntu@ec2-44-202-88-33.compute-1.amazonaws.com

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

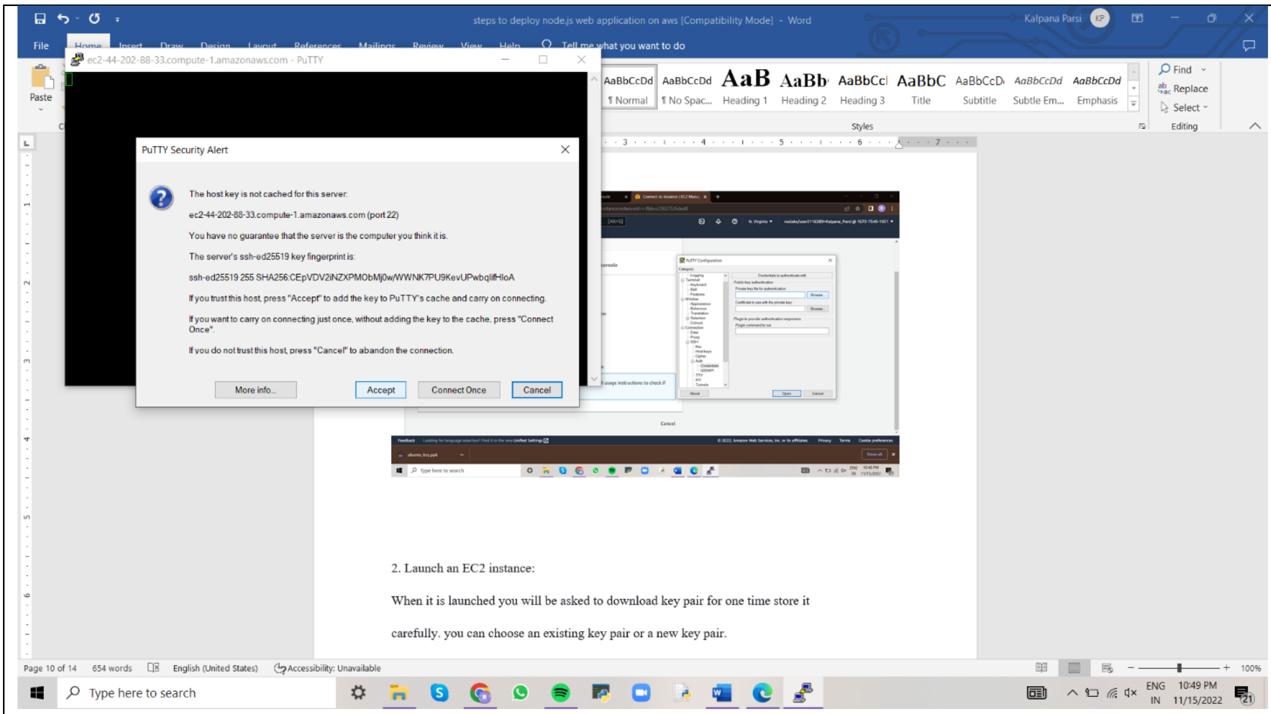
PuTTY Configuration — Basic options for your PuTTY session
Host Name (or IP address): 2-44-202-88-33.compute-1.amazonaws.com
Port: 22
Connection type: ● SSH ○ Serial ○ Other: Telnet
Saved Sessions: Default Settings
Close window on exit: ○ Always ○ Never ● Only on clean exit

Click on Connection – SSH – Auth – Credentials –



PuTTY Configuration — Credentials to authenticate with
Public-key authentication
Private key file for authentication:
Certificate to use with the private key:
Plugin to provide authentication responses
Plugin command to run

Private key for Authentication - Browse - select the .ppk which was downloaded when EC2 instance is created



PuTTY Security Alert

The host key is not cached for this server:

ec2-44-202-88-33.compute-1.amazonaws.com (port 22)

You have no guarantee that the server is the computer you think it is.

The server's ssh-ed25519 key fingerprint is:

ssh-ed25519 255 SHA256:CEpVDV2iNZXPMObMj0w/WWNK7PU9KevUPwbqlifHloA

If you trust this host, press "Accept" to add the key to PuTTY's cache and carry on connecting.

If you want to carry on connecting just once, without adding the key to the cache, press "Connect Once".

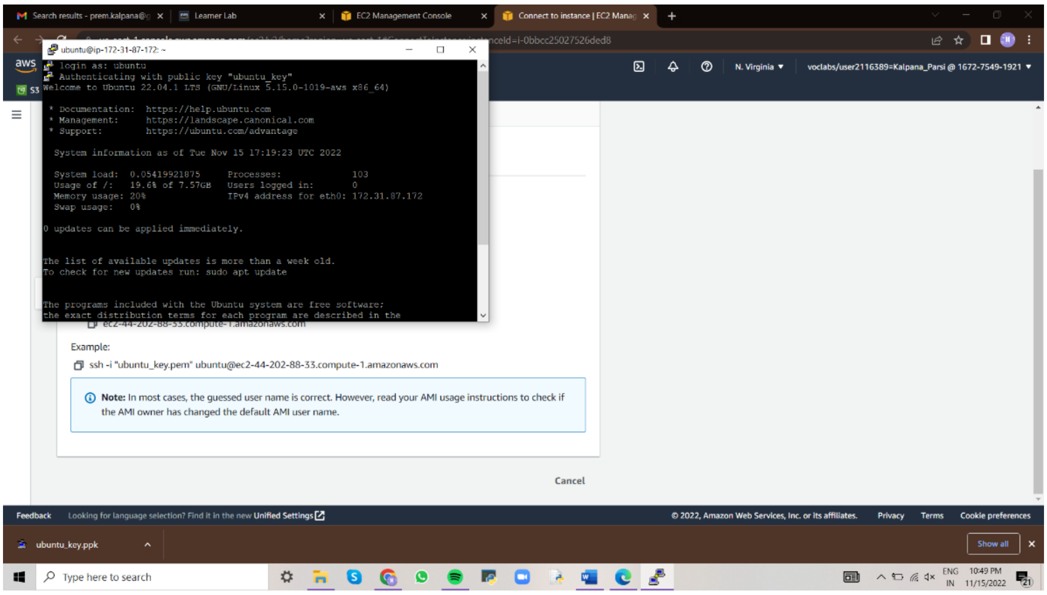
If you do not trust this host, press "Cancel" to abandon the connection.

2. Launch an EC2 instance:

When it is launched you will be asked to download key pair for one time store it carefully. you can choose an existing key pair or a new key pair.

Once entered, it will ask you to confirm, click on Accept

Once it is opened login as ubuntu



```
login as: ubuntu
Authenticating with public key "ubuntu_key"
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1019-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Nov 15 17:19:23 UTC 2022

  System load:  0.05419921875     Processes:             103
  Usage of /:   19.6% of 7.57GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for eth0: 172.31.87.172
  Swap usage:   0%

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
```

Example:

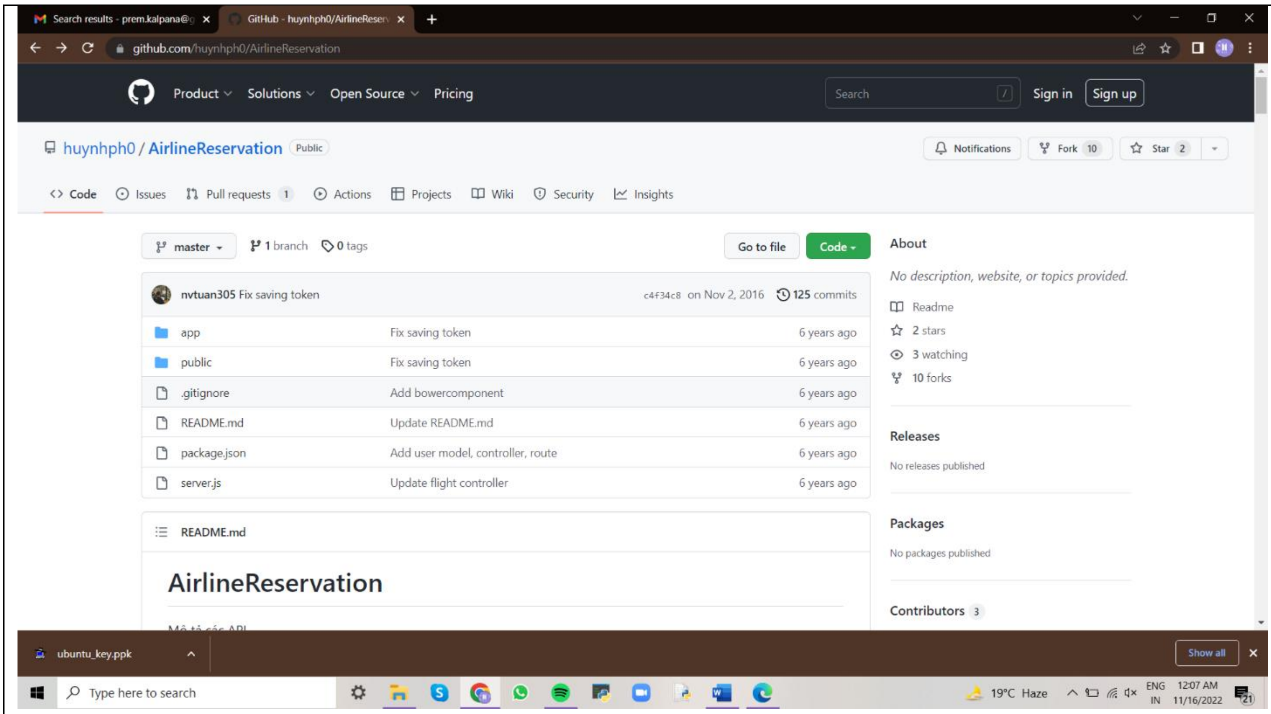
ssh -i "ubuntu_key.pem" ubuntu@ec2-44-202-88-33.compute-1.amazonaws.com

Note: In most cases, the guessed user name is correct. However, read your AMI usage instructions to check if the AMI owner has changed the default AMI user name.

mkdir demo

cd demo

git clone https://github.com/hoanghuynh1995/AirlineReservation



nvtuan305 Fix saving token   c4f34c8 on Nov 2, 2016   125 commits

app	Fix saving token	6 years ago
public	Fix saving token	6 years ago
.gitignore	Add bowercomponent	6 years ago
README.md	Update README.md	6 years ago
package.json	Add user model, controller, route	6 years ago
server.js	Update flight controller	6 years ago

README.md

AirlineReservation

Mô tả các API

About

No description, website, or topics provided.

Readme

2 stars

3 watching

10 forks

Releases

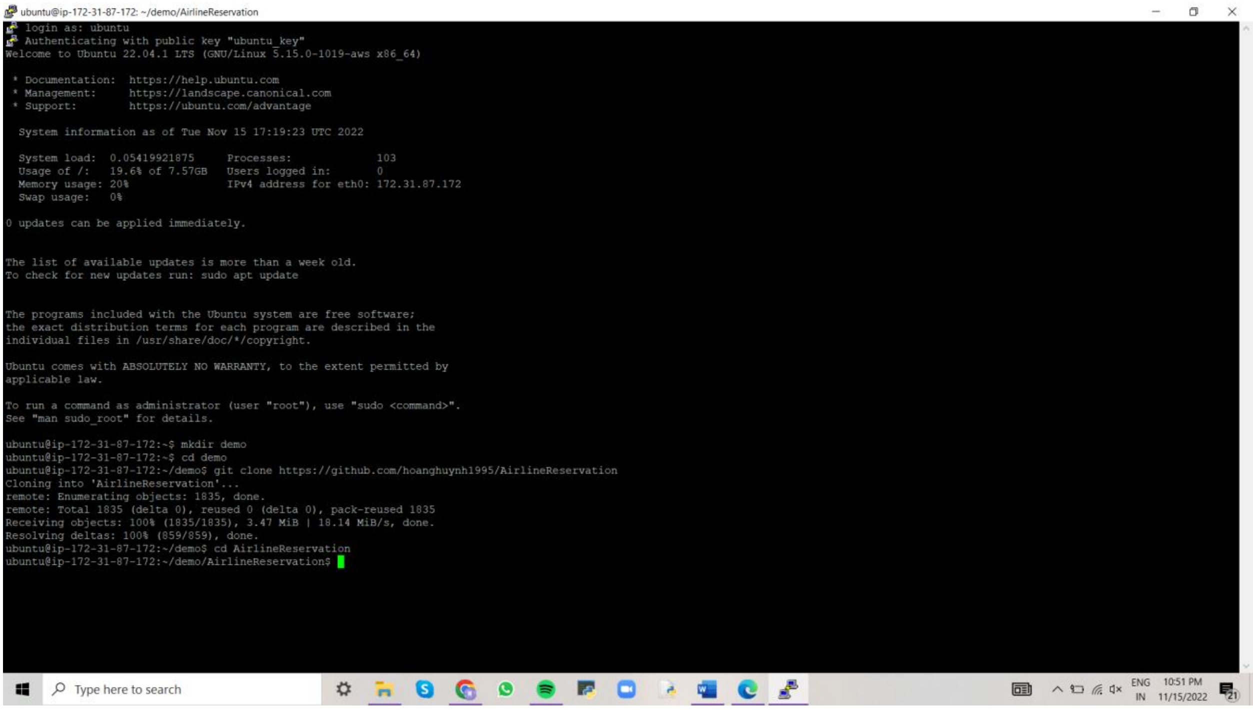
No releases published

Packages

No packages published

Contributors 3

cd AirlineReservation



```
login as: ubuntu
Authenticating with public key "ubuntu_key"
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1019-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Tue Nov 15 17:19:23 UTC 2022

  System load:  0.05419921875     Processes:             103
  Usage of /:   19.6% of 7.57GB   Users logged in:       0
  Memory usage: 20%               IPv4 address for eth0: 172.31.87.172
  Swap usage:   0%

0 updates can be applied immediately.


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.

ubuntu@ip-172-31-87-172:~$ mkdir demo
ubuntu@ip-172-31-87-172:~$ cd demo
ubuntu@ip-172-31-87-172:~/demo$ git clone https://github.com/hoanghuynh1995/AirlineReservation
Cloning into 'AirlineReservation'...
remote: Enumerating objects: 1835, done.
remote: Total 1835 (delta 0), reused 0 (delta 0), pack-reused 1835
Receiving objects: 100% (1835/1835), 3.47 MiB | 18.14 MiB/s, done.
Resolving deltas: 100% (859/859), done.
ubuntu@ip-172-31-87-172:~/demo$ cd AirlineReservation
ubuntu@ip-172-31-87-172:~/demo/AirlineReservation$
```

sudo apt-get update //to download package information from all configured sources

sudo apt-get install npm

//to install Node.js on ubuntu, we must first install npm (node package manager) select Yes

Ok

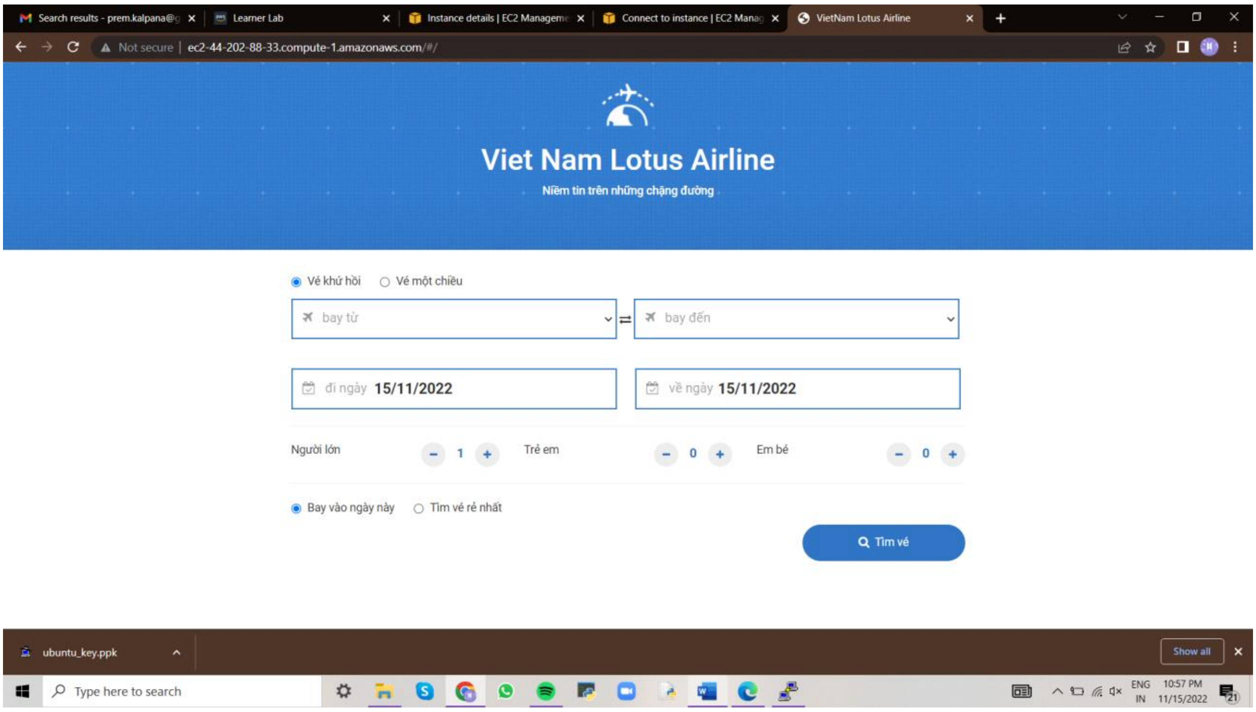
npm install

sudo apt-get install nodejs //to install Node.js on ubuntu

open server.js file using vi editor and change the port no to 80, and save file and exit

sudo node server.js

Copy public DNS of your instance in new tab and view the deployed web application.



Not secure | ec2-44-202-88-33.compute-1.amazonaws.com/#/

Viet Nam Lotus Airline

Niềm tin trên những chặng đường

● Vé khứ hồi ○ Vé một chiều

bay từ ⇄ bay đến

đi ngày 15/11/2022

về ngày 15/11/2022

Người lớn − 1 + Trẻ em − 0 + Em bé − 0 +

● Bay vào ngày này ○ Tìm vé rẻ nhất

Lab Program

C program for Three address code generation.

Code:

three.l

```
%{
#include "y.tab.h"
extern char yyval;
%}
number [0-9]+
letter [a-zA-Z]+
%%
{number} {yylval.sym=(char)yytext[0];return number;}
{letter} {yylval.sym=(char)yytext[0]; return letter; }
\n {return 0;}
.  {return yytext[0];}
%%
```

Three.y

```
%{
#include<stdio.h>
#include<string.h>
int nIndex=0;
struct Intercode
{
char operand1;
char operand2;
char opera;
};
%}
```

```
%union
{
char sym;
}
%token <sym> letter number
%type  <sym> expr
%left '-' '+'
%right '*' '/'
%%
statement: letter '=' expr ';' { addtotable((char)$1,(char)$3,'=' ); }
            | expr ;
            ;
expr: expr '+' expr  { $$=addtotable((char)$1,(char)$3,'+');}
        | expr '-'  expr { $$=addtotable((char)$1,(char)$3,'-');}
        | expr '*'  expr { $$=addtotable((char)$1,(char)$3, '*');}
        | expr '/'  expr { $$=addtotable((char)$1,(char)$3,'/');}
        | '(' expr ')' { $$= (char)$2;}
        |  number  { $$= (char)$1;}
        | letter { $$= (char)$1;}
%%
yyerror(char *s)
{
printf("%s",s);
exit (0);
}
struct Intercode code[20];
char temp = 'A';
int f=0;
char addtotable(char operand1, char operand2,char opera)
{
```

```
if(f!=0)
        temp++;
code[nIndex].operand1 = operand1;
code[nIndex].operand2 = operand2;
code[nIndex].opera = opera;
nIndex++;
f++;
return temp;
}
threeaddresscode()
{
int nCnt=0;
char temp='A';
printf("\n\n\t three address codes\n\n");
while(nCnt<nIndex)
{
printf("%c:=\t",temp);
if (isalpha(code[nCnt].operand1))
printf("%c\t", code[nCnt].operand1);
else
printf("%c\t",temp);
printf("%c\t", code[nCnt].opera);
if (isalpha(code[nCnt].operand2))
printf("%c\t", code[nCnt].operand2);
else
printf("%c\t",temp);
printf("\n");
nCnt++;
temp++;
}}
```

```
main()
{
printf("enter expression");
yyparse();
threeaddresscode();
}
yywrap()
{
return 1;
}
```

Output:

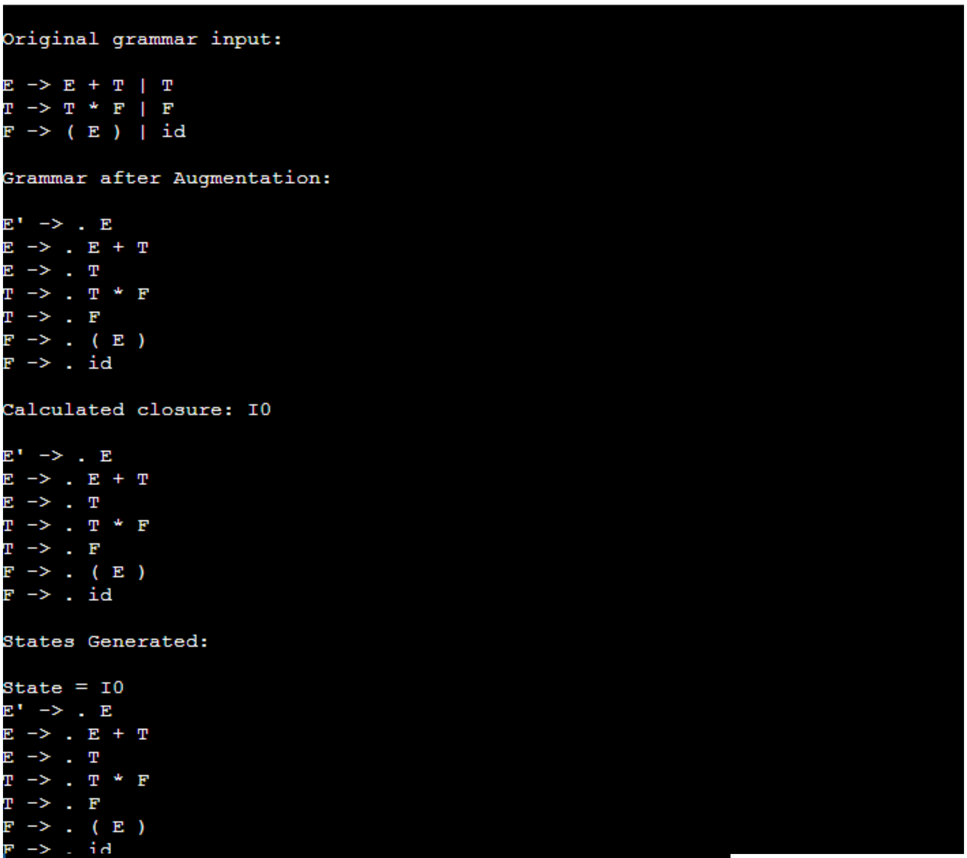
```
enter expression (a*b)+(c*d)
        three address codes


B:=    a      *      b
C:=    c      *      d
D:=    B      +      C
```

LAB PROGRAMS

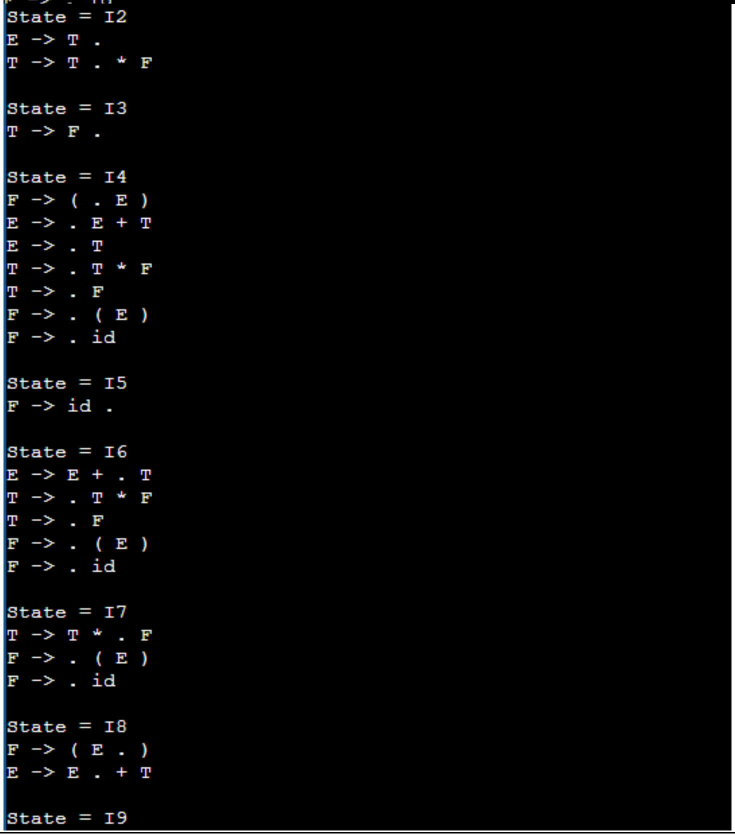
Implement SLR parser.

```
import copy
def grammarAugmentation(rules, nonterm_userdef,
                        start_symbol):
    newRules = []
    newChar = start_symbol + "'"
    while (newChar in nonterm_userdef):
        newChar += "'"
    newRules.append([newChar,
                    ['.', start_symbol]])
    for rule in rules:
        k = rule.split("->")
        lhs = k[0].strip()
        rhs = k[1].strip()
        multirhs = rhs.split('|')
        for rhs1 in multirhs:
            rhs1 = rhs1.strip().split()
            rhs1.insert(0, '.')
            newRules.append([lhs, rhs1])
    return newRules
def findClosure(input_state, dotSymbol):
    global start_symbol, \
        separatedRulesList, \
        statesDict
    closureSet = []
    if dotSymbol == start_symbol:
        for rule in separatedRulesList:
            if rule[0] == dotSymbol:
                closureSet.append(rule)
    else:
        closureSet = input_state
    prevLen = -1
    while prevLen != len(closureSet):
        prevLen = len(closureSet)
        tempClosureSet = []
        for rule in closureSet:
            indexOfDot = rule[1].index('.')
            if rule[1][-1] != '.':
                dotPointsHere = rule[1][indexOfDot + 1]
                for in_rule in separatedRulesList:
                    if dotPointsHere == in_rule[0] and \
                            in_rule not in tempClosureSet:
                        tempClosureSet.append(in_rule)
        for rule in tempClosureSet:
            if rule not in closureSet:
                closureSet.append(rule)
    return closureSet
def compute_GOTO(state):
    global statesDict, stateCount
    generateStatesFor = []
    for rule in statesDict[state]:
        if rule[1][-1] != '.':
            indexOfDot = rule[1].index('.')
            dotPointsHere = rule[1][indexOfDot + 1]
            if dotPointsHere not in generateStatesFor:
                generateStatesFor.append(dotPointsHere)
    if len(generateStatesFor) != 0:
        for symbol in generateStatesFor:
            GOTO(state, symbol)
    return
def GOTO(state, charNextToDot):
    global statesDict, stateCount, stateMap
    newState = []
    for rule in statesDict[state]:
        indexOfDot = rule[1].index('.')
        if rule[1][-1] != '.':
            if rule[1][indexOfDot + 1] == \
                    charNextToDot:
                shiftedRule = copy.deepcopy(rule)
                shiftedRule[1][indexOfDot] = \
                    shiftedRule[1][indexOfDot + 1]
                shiftedRule[1][indexOfDot + 1] = '.'
                newState.append(shiftedRule)
    addClosureRules = []
    for rule in newState:
        indexDot = rule[1].index('.')
        if rule[1][-1] != '.':
            closureRes = \
                findClosure(newState, rule[1][indexDot + 1])
            for rule in closureRes:
                if rule not in addClosureRules \
                        and rule not in newState:
                    addClosureRules.append(rule)
    for rule in addClosureRules:
        newState.append(rule)
    stateExists = -1
    for state_num in statesDict:
        if statesDict[state_num] == newState:
            stateExists = state_num
            break
    if stateExists == -1:
        stateCount += 1
        statesDict[stateCount] = newState
        stateMap[(state, charNextToDot)] = stateCount
    else:
        stateMap[(state, charNextToDot)] = stateExists
    return
def generateStates(statesDict):
    prev_len = -1
    called_GOTO_on = []
    while (len(statesDict) != prev_len):
        prev_len = len(statesDict)
        keys = list(statesDict.keys())
        for key in keys:
            if key not in called_GOTO_on:
                called_GOTO_on.append(key)
                compute_GOTO(key)
    return
def first(rule):
    global rules, nonterm_userdef, \
        term_userdef, diction, firsts
    if len(rule) != 0 and (rule is not None):
        if rule[0] in term_userdef:
            return rule[0]
        elif rule[0] == '#':
            return '#'
    if len(rule) != 0:
        if rule[0] in list(diction.keys()):
            fres = []
            rhs_rules = diction[rule[0]]
            for itr in rhs_rules:
                indivRes = first(itr)
                if type(indivRes) is list:
                    for i in indivRes:
                        fres.append(i)
                else:
                    fres.append(indivRes)
            if '#' not in fres:
                return fres
            else:
                newList = []
                fres.remove('#')
                if len(rule) > 1:
                    ansNew = first(rule[1:])
                    if ansNew != None:
                        if type(ansNew) is list:
                            newList = fres + ansNew
                        else:
                            newList = fres + [ansNew]
                    else:
                        newList = fres
                    return newList
                fres.append('#')
                return fres
def follow(nt):
    global start_symbol, rules, nonterm_userdef, \
        term_userdef, diction, firsts, follows
    solset = set()
    if nt == start_symbol:
        solset.add('$')
    for curNT in diction:
        rhs = diction[curNT]
        for subrule in rhs:
            if nt in subrule:
                while nt in subrule:
                    index_nt = subrule.index(nt)
                    subrule = subrule[index_nt + 1:]
                    if len(subrule) != 0:
                        res = first(subrule)
                        if '#' in res:
                            newList = []
                            res.remove('#')
                            ansNew = follow(curNT)
                            if ansNew != None:
                                if type(ansNew) is list:
                                    newList = res + ansNew
                                else:
                                    newList = res + [ansNew]
                            else:
                                newList = res
                            res = newList
                    else:
                        if nt != curNT:
                            res = follow(curNT)
                    if res is not None:
                        if type(res) is list:
                            for g in res:
                                solset.add(g)
                        else:
                            solset.add(res)
    return list(solset)
def createParseTable(statesDict, stateMap, T, NT):
    global separatedRulesList, diction
    rows = list(statesDict.keys())
    cols = T+['$']+NT
    Table = []
    tempRow = []
    for y in range(len(cols)):
        tempRow.append('')
    for x in range(len(rows)):
        Table.append(copy.deepcopy(tempRow))
    for entry in stateMap:
        state = entry[0]
        symbol = entry[1]
        # get index
        a = rows.index(state)
        b = cols.index(symbol)
        if symbol in NT:
            Table[a][b] = Table[a][b]\
                + f"{stateMap[entry]} "
        elif symbol in T:
            Table[a][b] = Table[a][b]\
                + f"S{stateMap[entry]} "
    numbered = {}
    key_count = 0
    for rule in separatedRulesList:
        tempRule = copy.deepcopy(rule)
        tempRule[1].remove('.')
        numbered[key_count] = tempRule
        key_count += 1
    addedR = f"{separatedRulesList[0][0]} -> " \
        f"{separatedRulesList[0][1][1]}"
    rules.insert(0, addedR)
    for rule in rules:
        k = rule.split("->")
        k[0] = k[0].strip()
        k[1] = k[1].strip()
        rhs = k[1]
        multirhs = rhs.split('|')
        for i in range(len(multirhs)):
            multirhs[i] = multirhs[i].strip()
            multirhs[i] = multirhs[i].split()
        diction[k[0]] = multirhs
    for stateno in statesDict:
        for rule in statesDict[stateno]:
            if rule[1][-1] == '.':
                temp2 = copy.deepcopy(rule)
                temp2[1].remove('.')
                for key in numbered:
                    if numbered[key] == temp2:
                        follow_result = follow(rule[0])
                        for col in follow_result:
                            index = cols.index(col)
                            if key == 0:
                                Table[stateno][index] = "Accept"
                            else:
                                Table[stateno][index] =\
                                    Table[stateno][index]+f"R{key} "
    print("\nSLR(1) parsing table:\n")
    frmt = "{:>8}" * len(cols)
    print(" ", frmt.format(*cols), "\n")
    ptr = 0
    j = 0
    for y in Table:
        frmt1 = "{:>8}" * len(y)
        print(f"{{:>3}} {frmt1.format(*y)}"
            .format('I'+str(j)))
        j += 1
def printResult(rules):
    for rule in rules:
        print(f"{rule[0]} ->"
            f" {' '.join(rule[1])}")
def printAllGOTO(diction):
    for itr in diction:
        print(f"GOTO ( I{itr[0]} ,"
                              f" {itr[1]} ) = I{stateMap[itr]}")
rules = ["E -> E + T | T",
                    "T -> T * F | F",
                    "F -> ( E ) | id"
                    ]
nonterm_userdef = ['E', 'T', 'F']
term_userdef = ['id', '+', '*', '(', ')']
start_symbol = nonterm_userdef[0]
print("\nOriginal grammar input:\n")
for y in rules:
          print(y)
print("\nGrammar after Augmentation: \n")
separatedRulesList = \
          grammarAugmentation(rules,
                                                  nonterm_userdef,
                                                  start_symbol)
printResult(separatedRulesList)
start_symbol = separatedRulesList[0][0]
print("\nCalculated closure: I0\n")
I0 = findClosure(0, start_symbol)
printResult(I0)
statesDict = {}
stateMap = {}
statesDict[0] = I0
stateCount = 0
generateStates(statesDict)
print("\nStates Generated: \n")
for st in statesDict:
          print(f"State = I{st}")
          printResult(statesDict[st])
          print()
print("Result of GOTO computation:\n")
printAllGOTO(stateMap)
diction = {}
createParseTable(statesDict, stateMap,
                                        term_userdef,
                                        nonterm_userdef)
```

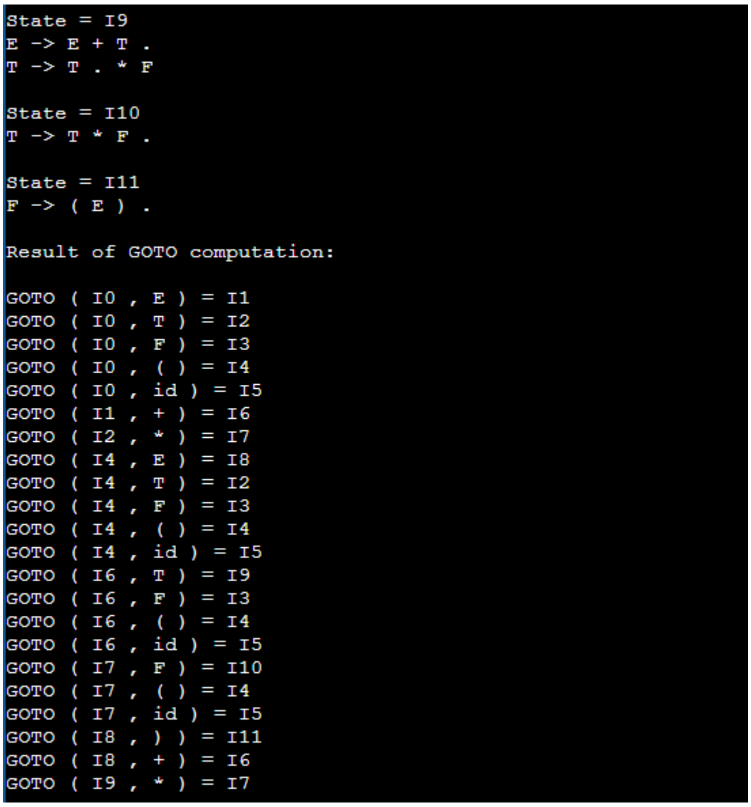
OUTPUT:



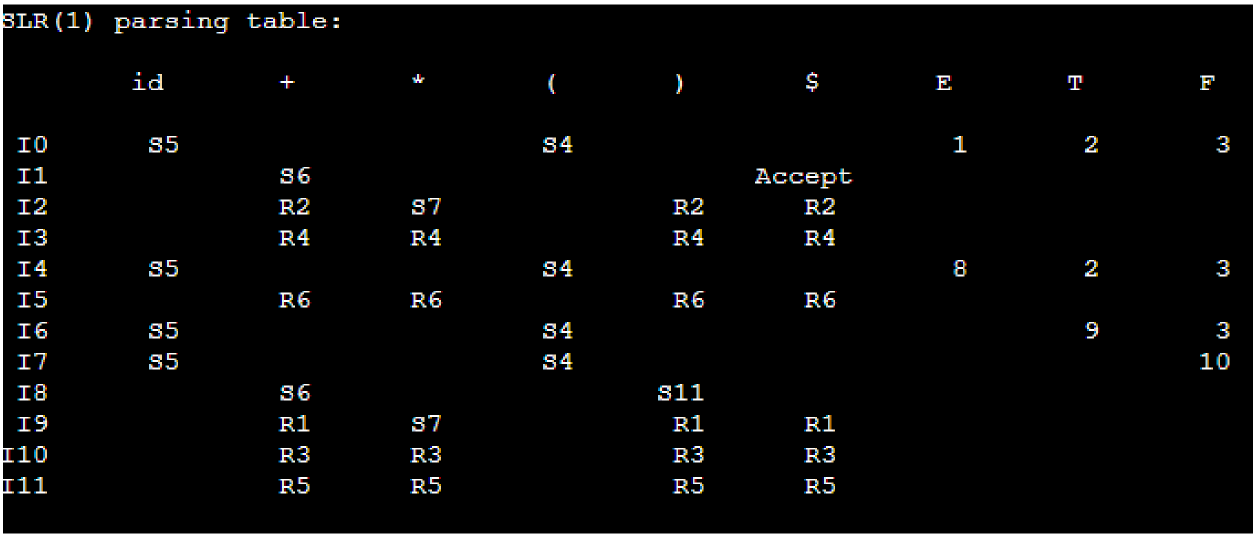
```
Original grammar input:

E -> E + T | T
T -> T * F | F
F -> ( E ) | id

Grammar after Augmentation:

E' -> . E
E -> . E + T
E -> . T
T -> . T * F
T -> . F
F -> . ( E )
F -> . id

Calculated closure: I0

E' -> . E
E -> . E + T
E -> . T
T -> . T * F
T -> . F
F -> . ( E )
F -> . id

States Generated:

State = I0
E' -> . E
E -> . E + T
E -> . T
T -> . T * F
T -> . F
F -> . ( E )
F -> . id
```



```
State = I2
E -> T .
T -> T . * F

State = I3
T -> F .

State = I4
F -> ( . E )
E -> . E + T
E -> . T
T -> . T * F
T -> . F
F -> . ( E )
F -> . id

State = I5
F -> id .

State = I6
E -> E + . T
T -> . T * F
T -> . F
F -> . ( E )
F -> . id

State = I7
T -> T * . F
F -> . ( E )
F -> . id

State = I8
F -> ( E . )
E -> E . + T

State = I9
```



```
State = I9
E -> E + T .
T -> T . * F

State = I10
T -> T * F .

State = I11
F -> ( E ) .

Result of GOTO computation:

GOTO ( I0 , E ) = I1
GOTO ( I0 , T ) = I2
GOTO ( I0 , F ) = I3
GOTO ( I0 , ( ) = I4
GOTO ( I0 , id ) = I5
GOTO ( I1 , + ) = I6
GOTO ( I2 , * ) = I7
GOTO ( I4 , E ) = I8
GOTO ( I4 , T ) = I2
GOTO ( I4 , F ) = I3
GOTO ( I4 , ( ) = I4
GOTO ( I4 , id ) = I5
GOTO ( I6 , T ) = I9
GOTO ( I6 , F ) = I3
GOTO ( I6 , ( ) = I4
GOTO ( I6 , id ) = I5
GOTO ( I7 , F ) = I10
GOTO ( I7 , ( ) = I4
GOTO ( I7 , id ) = I5
GOTO ( I8 , ) ) = I11
GOTO ( I8 , + ) = I6
GOTO ( I9 , * ) = I7
```



```
SLR(1) parsing table:

        id       +       *        (       )       $       E       T       F

 I0      S5                      S4                        1       2       3
 I1              S6                           Accept
 I2              R2      S7               R2      R2
 I3              R4      R4               R4      R4
 I4      S5                      S4                        8       2       3
 I5              R6      R6               R6      R6
 I6      S5                      S4                                9       3
 I7      S5                      S4                                       10
 I8              S6                     S11
 I9              R1      S7               R1      R1
I10              R3      R3               R3      R3
I11              R5      R5               R5      R5
```

Implement parser generator using YACC(calculator)

Lex program:

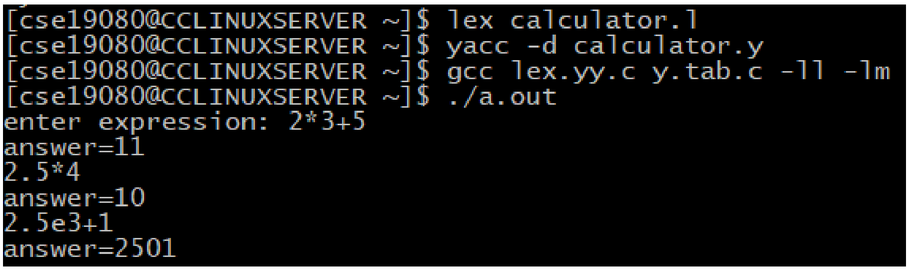
```
%{
#include &quot;y.tab.h&quot;
#include&lt;math.h&gt;
%}
%%
([0-9]+|([0-9]*\.[0-9]+)([eE][-+]?[0-9]+)?) {yylval.dval=atof(yytext);
return NUMBER;
}
log|LOG {return LOG;}
ln {return nLOG;}
sin|SIN {return SINE;}
cos|COS {return COS;}
tan|TAN {return TAN;}
mem {return MEM;}
[\t];
\$; {return 0;}
\n|. {return yytext[0];}
%%
```

yacc program

```
%{
#include&lt;stdio.h&gt;
#include&lt;math.h&gt;
double memvar;
%}
%union
{

double dval;
}
%token&lt;dval&gt;NUMBER
%token&lt;dval&gt;MEM
%token LOG SINE nLOG COS TAN
%left &#39;-&#39;&#39;+&#39;
%left &#39;*&#39;&#39;/&#39;
%right &#39;^&#39;
%left LOG SINE nLOG COS TAN
%nonassoc UMINUS
%type&lt;dval&gt; expression
%%
start: statement &#39;\n&#39;
|start statement &#39;\n&#39;
;
statement: MEM&#39;=&#39;expression { memvar=$3;}
|expression {printf(&quot;answer=%g\n&quot;,$1);}
;
expression:expression&#39;+&#39;expression {$$=$1+$3;}
|expression&#39;-&#39;expression {$$=$1+$3;}
```

```
|expression&#39;*&#39;expression {$$=$1*$3;}
|expression&#39;/&#39;expression
{
if($3==0)
yyerror(&quot;divide by zero&quot;);
else
$$=$1/$3;}
|expression&#39;^&#39;expression {$$=pow($1,$3);}
;
expression: &#39;-&#39; expression %prec UMINUS {$$=-$2;}
|&#39;(&#39;expression&#39;)&#39; {$$=$2;}
|LOG expression {$$=log($2)/log(10);}
|nLOG expression {$$=log($2);}
|SINE expression {$$=sin($2*3.14159/180);}
|COS expression {$$=cos($2*3.14159/180);}
|TAN expression {$$=tan($2*3.14159/180);}
|NUMBER { $$ = $1;}
|MEM {$$=memvar;}
;
%%
main()
{
printf(&quot;enter expression:&quot;);
yyparse();
}
int yyerror(char *error)

{
fprintf(stderr,&quot;%s\n&quot;,error);
}
yywrap()
{ return 1;
}
```

OUTPUT:



```
[cse19080@CCLINUXSERVER ~]$ lex calculator.l
[cse19080@CCLINUXSERVER ~]$ yacc -d calculator.y
[cse19080@CCLINUXSERVER ~]$ gcc lex.yy.c y.tab.c -ll -lm
[cse19080@CCLINUXSERVER ~]$ ./a.out
enter expression: 2*3+5
answer=11
2.5*4
answer=10
2.5e3+1
answer=2501
```