Distributed System and Cloud Computing

A Project Report Submitted in Fulfillment

of the Degree of

MASTER

In

COMPUTER APPLICATION

Year 2022-2023

By

Mr. Gupta Sudhir Prahlad Sabitree

(Seat No-806061)

(Application Id-171010)

Under the Guidance of

Prof. Trupti Rongare

Institute of Distance and Open Learning

Vidya Nagari, Kalina, Santacruz East – 400098.

University of Mumbai

PCP Center

Satish Pradhan Dnyanasadhana College, Thane.

Institute of Distance and Open Learning

Vidya Nagari, Kalina, Santacruz East – 400098.

CERTIFICATE

This is to certify that, this project report entitled **"(Distributed System and Cloud Computing)"** is a record of work carried out by **Mr. Gupta Sudhir Prahlad Sabitree (Seat no-806061),** student of **MCA semester-III** class and is submitted to University of Mumbai, in partial fulfillment of the requirement for the award of the degree of **Master in Computer Application**. The project report has been approved.

Guide	External Examiner	Coordinator – M.C.A

ACKNOWLEDGMENT

After the completion of this work, words are not enough to express my feelings about all those who helped me to reach my goal; feeling above this is my indebtedness to the almighty for providing me this moment in my life.

It's a great pleasure and moment of immense satisfaction for me to express my profound gratitude to my Practical guide, **Asst. Prof. Trupti Rongare.** whose constant encouragement enabled me to work enthusiastically. Her perpetual motivation, patience and excellent expertise in discussion during progress of dissertation work have benefited me to an extent, which is beyond expression. Her depth and breadth of knowledge of Engineering field made me realize that theoretical knowledge always help to develop efficient operational software, which is a blend of all core subjects of the field. The completion of this project would not have been possible without her encouragement, patient guidance and constant support.

I would like to thank all staff members for their valuable cooperation and permitting me to work in the computer labs.

Special thanks to my colleagues and friends for providing me useful comments, suggestions and continuous encouragement.

Finally, I thanks my family members, for their support and endurance during this work.

Mr. Gupta Sudhir Prahlad Sabitree

(**Seat No:- 806061**)

Declaration

I declare that this written submission represents my ideas in my own words and where other's ideas or words have been included, I have adequately cited and referenced the original sources. I also declare that I have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in my submission. I understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

(Signature)

Mr. Gupta Sudhir Prahlad Gupta

Seat No-806061

Date:

Place:

INDEX

Practical No. 1

Aim: Write a program to develop multi-client server application, where multiple clients chat with each other concurrently.

Code:

MultithreadedSocketServer.java

```
import java.net.*;

public class MultithreadedSocketServer {

   public static void main(String[] args) throws Exception {

      try {

         ServerSocket server = new ServerSocket(8888);

         int counter = 0;

         System.out.println("Server Started ....");

         while (true) {

            counter++;

            Socket serverClient = server.accept(); // server accept the client connection request

            System.out.println(" >> " + "Client No:" + counter + " started!");

            ServerClientThread sct = new ServerClientThread(serverClient, counter); // send the request to
a separate thread

            sct.start();

         }

      } catch (Exception e) {

         System.out.println(e);

      }

   }

}
```

ServerClientThread.java

```
import java.net.*;
```

```
import java.io.*;
class ServerClientThread extends Thread {
   Socket serverClient;
   int clientNo;
   int squre;
   ServerClientThread(Socket inSocket, int counter) {
      serverClient = inSocket;
      clientNo = counter;
   }
   public void run() {
      try {
         DataInputStream inStream = new DataInputStream(serverClient.getInputStream());
         DataOutputStream outStream = new DataOutputStream(serverClient.getOutputStream());
         String clientMessage = "", serverMessage = "";
         while (!clientMessage.equals("bye")) {
            clientMessage = inStream.readUTF();
            System.out.println("From Client-" + clientNo+": Number is :" + clientMessage);
            squre = Integer.parseInt(clientMessage) *
                  Integer.parseInt(clientMessage);
            serverMessage = "From Server to Client-"+clientNo+ " Square of " +
                  clientMessage + " is " + squre;
            outStream.writeUTF(serverMessage);
            outStream.flush();
         }
         inStream.close();
         outStream.close();
         serverClient.close();
      } catch (Exception ex) {
```

```
            System.out.println(ex);
        } finally {
            System.out.println("Client -" + clientNo + " exit!! ");
        }
    }
}
```

TCPClient.java

```
import java.net.*;
import java.io.*;
public class TCPClient {
    public static void main(String[] args) throws Exception {
        try {
            Socket socket = new Socket("127.0.0.1", 8888);
            DataInputStream inStream = new DataInputStream(socket.getInputStream());
            DataOutputStream outStream = new DataOutputStream(socket.getOutputStream());
            BufferedReader br = new BufferedReader(new InputStreamReader(System.in));
            String clientMessage = "", serverMessage = "";
            while (!clientMessage.equals("bye")) {
                System.out.println("Enter number :");
                clientMessage = br.readLine();
                outStream.writeUTF(clientMessage);
                outStream.flush();
                serverMessage = inStream.readUTF();
                System.out.println(serverMessage);
            }
            outStream.close();
            outStream.close();
```

```
        socket.close();
      } catch (Exception e) {
        System.out.println(e);
      }
    }
}
```

Output:

```
Microsoft Windows [Version 10.0.22000.1696]
(c) Microsoft Corporation. All rights reserved.

C:\MCA\DSCC\P1>javac ServerClientThread.java

C:\MCA\DSCC\P1>javac MultithreadedSocketServer.java

C:\MCA\DSCC\P1>java MultithreadedSocketServer
Server Started ....
 >> Client No:1 started!
From Client-1: Number is :007
From Client-1: Number is :9
From Client-1: Number is :15
```

```
Microsoft Windows [Version 10.0.22000.1696]
(c) Microsoft Corporation. All rights reserved.

C:\MCA\DSCC\P1>javac TCPClient.java

C:\MCA\DSCC\P1>java TCPClient
Enter number :
007
From Server to Client-1 Square of 007 is 49
Enter number :
9
From Server to Client-1 Square of 9 is 81
Enter number :
15
From Server to Client-1 Square of 15 is 225
Enter number :
```

Practical No. 2

Aim: Write a program to implement a server calculator using RPC concept.

Code:

RPCServer.java

```
import java.util.*;
import java.net.*;
class RPCServer
{
DatagramSocket ds;
DatagramPacket dp;
String str,methodName,result;
int val1,val2;
RPCServer()
{
try
{
ds=new DatagramSocket(1200);
byte b[]=new byte[4096];
while(true)
{
dp=new DatagramPacket(b,b.length);
ds.receive(dp);
str=new String(dp.getData(),0,dp.getLength());
if(str.equalsIgnoreCase("q"))
{
System.exit(1);
}
else
```

```
{
StringTokenizer st = new StringTokenizer(str," ");
int i=0;
while(st.hasMoreTokens())
{
String token=st.nextToken();
methodName=token;
val1 = Integer.parseInt(st.nextToken());
val2 = Integer.parseInt(st.nextToken());
}
}
System.out.println(str);
InetAddress ia = InetAddress.getLocalHost();
if(methodName.equalsIgnoreCase("add"))
{
result= "" + add(val1,val2);
}
else if(methodName.equalsIgnoreCase("sub"))
{
result= "" + sub(val1,val2);
}
else if(methodName.equalsIgnoreCase("mul"))
{
result= "" + mul(val1,val2);
}
else if(methodName.equalsIgnoreCase("div"))
{
result= "" + div(val1,val2);
```

```
}
byte b1[]=result.getBytes();
DatagramSocket ds1 = new DatagramSocket();
DatagramPacket dp1 = new
DatagramPacket(b1,b1.length,InetAddress.getLocalHost(), 1300);
System.out.println("result : "+result+"\n");
ds1.send(dp1);
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
public int add(int val1, int val2)
{
return val1+val2;
}
public int sub(int val3, int val4)
{
return val3-val4;
}
public int mul(int val3, int val4)
{
return val3*val4;
}
public int div(int val3, int val4)
{
```

```
return val3/val4;
}
public static void main(String[] args)
{
new RPCServer();
}
}
```

Client-side java file:

RPCClient.java

```
import java.io.*;
import java.net.*;
class RPCClient
{
RPCClient()
{
try
{
InetAddress ia = InetAddress.getLocalHost();
DatagramSocket ds = new DatagramSocket();
DatagramSocket ds1 = new DatagramSocket(1300);
System.out.println("\nRPC Client\n");
System.out.println("Enter method name and parameter like add 3 4\n");
while (true)
{
BufferedReader br = new BufferedReader(new Remote Procedure Call
InputStreamReader(System.in));
String str = br.readLine();
```

```
byte b[] = str.getBytes();
DatagramPacket dp = new
DatagramPacket(b,b.length,ia,1200);
ds.send(dp);
dp = new DatagramPacket(b,b.length);
ds1.receive(dp);
String s = new String(dp.getData(),0,dp.getLength());
System.out.println("\nResult = " + s + "\n");
}
}
catch (Exception e)
{
e.printStackTrace();
}
}
public static void main(String[] args)
{
new RPCClient();
}
}
```

Output :

Command Prompt - java RPCServer

```
E:\MSc_My_Work\Distributed Systems\Practs\Pract4\Calculator>javac RPCServer.java

E:\MSc_My_Work\Distributed Systems\Practs\Pract4\Calculator>java RPCServer
sub 10 8
result : 2

mul 15 2
result : 30

add 20 3
result : 23

div 10 2
result : 5
```

C:\Windows\System32\cmd.exe - java RPCClient

```
E:\MSc_My_Work\Distributed Systems\Practs\Pract4\Calculator>java RPCClient
RPC Client
Enter method name and parameter like add 3 4
sub 10 8
Result = 2
mul 15 2
Result = 30
add 20 3
Result = 23
div 10 2
Result = 5
```

Practical No. 3 (a)

Aim : Demonstrate a sample RMI Java application.

Code :

Hello.java :-

```
import java.rmi.Remote;
import java.rmi.RemoteException;
// Creating Remote interface for our application
public interface Hello extends Remote {
 void printMsg() throws RemoteException;
}
```

ImplExample.java

```
public class ImplExample implements Hello {

 // Implementing the interface method
 public void printMsg() {
 System.out.println("This is an example RMI program");
 }
}
```

Server.java :-

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Server extends ImplExample {
 public Server() {}
 public static void main(String args[]) {
```

```
 try {
 // Instantiating the implementation class
 ImplExample obj = new ImplExample();

 // Exporting the object of implementation class
 // (here we are exporting the remote object to the stub)
 Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

 // Binding the remote object (stub) in the registry
 Registry registry = LocateRegistry.getRegistry();

 registry.bind("Hello", stub);
 System.err.println("Server ready");
 } catch (Exception e) {
 System.err.println("Server exception: " + e.toString());
 e.printStackTrace();
 }
 }
}
```

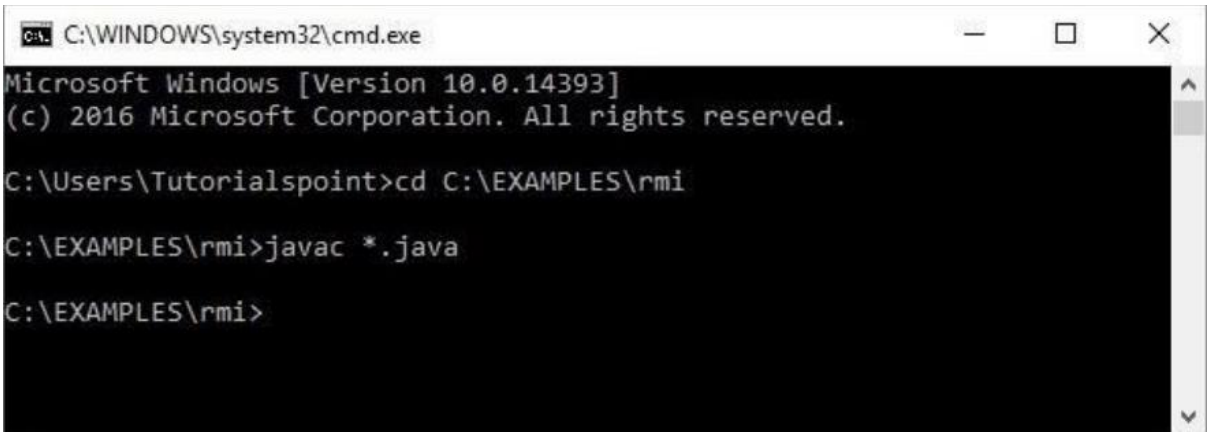
RMI client program :

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
public class Client {
 private Client() {}
 public static void main(String[] args) {
 try {
 // Getting the registry
 Registry registry = LocateRegistry.getRegistry(null);
```

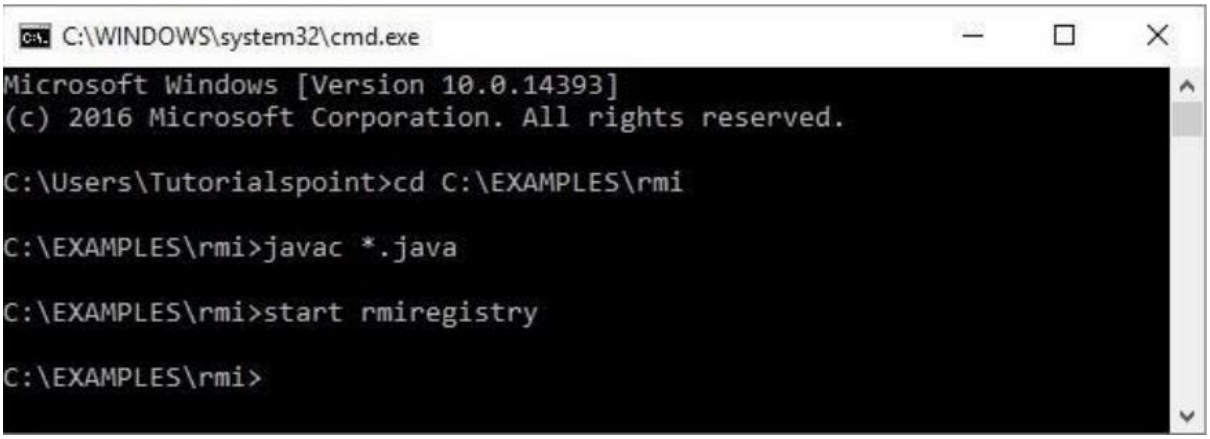
```
// Looking up the registry for the remote object
Hello stub = (Hello) registry.lookup("Hello");
// Calling the remote method using the obtained object
stub.printMsg();


// System.out.println("Remote method invoked");
} catch (Exception e){
System.err.println("Client exception: " + e.toString()); e.printStackTrace();
}
}}
```

Output :-

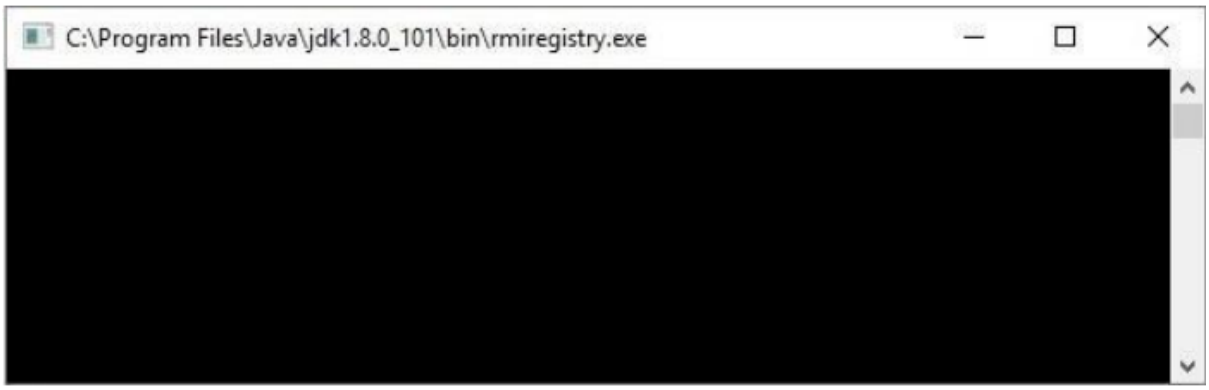
Javac *.java



Start rmiregistry

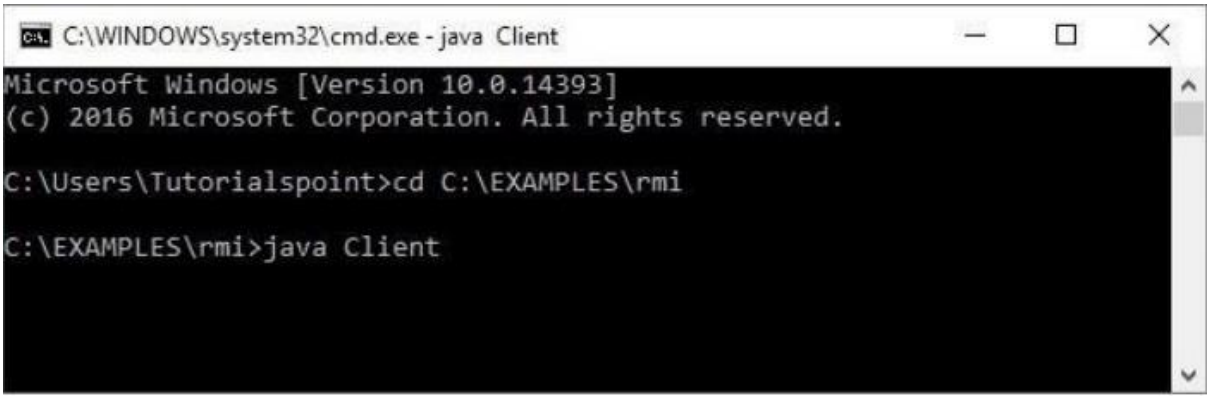


This will start an rmi registry on a separate window as shown below :-



Java Server

Java Client



Final Output

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>javac *.java

C:\EXAMPLES\rmi>start rmiregistry

C:\EXAMPLES\rmi>java Server
Server ready
This is an example RMI program
```

Practical No. 3 (b)

Aim : Demonstrate an how a client program can retrieve the records of a table in MySQL database residing on the server

Code :

Student.java

```
public class Student implements java.io.Serializable {
private int id, percent;
private String name, branch, email;
public int getId() {
return id;
}
public String getName() {
 return name;
 }
public String getBranch() {
return branch;
}
public int getPercent() {
 return percent;
 }
 public String getEmail() {
 return email;
 }
 public void setID(int id) {
 this.id = id;
 }
 public void setName(String name) {
 this.name = name;
```

```
 }
 public void setBranch(String branch) {
 this.branch = branch;
 }
 public void setPercent(int percent) {
 this.percent = percent;
 }
 public void setEmail(String email) {
 this.email = email;
 }
}
```

Creating Hello interface :

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.*;
public interface Hello extends Remote {
public List getStudents() throws Exception;
 }
```

ImplExample.java

```
import java.sql.*;
import java.util.*;
// Implementing the remote interface
public class ImplExample implements Hello {

 // Implementing the interface method
 public List<Student> getStudents() throws Exception {
 List<Student> list = new ArrayList<Student>();
```

```
// JDBC driver name and database URL
String JDBC_DRIVER = "com.mysql.jdbc.Driver";
String DB_URL = "jdbc:mysql://localhost:3306/details";

// Database credentials
String USER = "myuser";
String PASS = "password";
Connection conn = null;
Statement stmt = null;
//Register JDBC driver
Class.forName("com.mysql.jdbc.Driver");
//Open a connection
System.out.println("Connecting to a selected database...");
conn = DriverManager.getConnection(DB_URL, USER, PASS);
System.out.println("Connected database successfully...");
//Execute a query
System.out.println("Creating statement...");

stmt = conn.createStatement();
String sql = "SELECT * FROM student_data";
ResultSet rs = stmt.executeQuery(sql);

//Extract data from result set
while(rs.next()) {
// Retrieve by column name
int id = rs.getInt("id");

String name = rs.getString("name");
```

```
 String branch = rs.getString("branch");

 int percent = rs.getInt("percentage");
 String email = rs.getString("email");

 // Setting the values
 Student student = new Student();
 student.setID(id);
 student.setName(name);
 student.setBranch(branch);
 student.setPercent(percent);
 student.setEmail(email);
 list.add(student);
 }
 rs.close();
 return list;
 }
}
```

Server Program:

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Server extends ImplExample {
 public Server() {}
 public static void main(String args[]) {
 try {
 // Instantiating the implementation class
```

```
 ImplExample obj = new ImplExample();

 // Exporting the object of implementation class (
 here we are exporting the remote object to the stub)
 Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

 // Binding the remote object (stub) in the registry
 Registry registry = LocateRegistry.getRegistry();

 registry.bind("Hello", stub);
 System.err.println("Server ready");
 } catch (Exception e) {
 System.err.println("Server exception: " + e.toString());
 e.printStackTrace();
 }
 }
}
```

Client Program:

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.*;
public class Client {
 private Client() {}
 public static void main(String[] args)throws Exception {
 try {
 // Getting the registry
 Registry registry = LocateRegistry.getRegistry(null);
```

```
// Looking up the registry for the remote object
Hello stub = (Hello) registry.lookup("Hello");

// Calling the remote method using the obtained object
List<Student> list = (List)stub.getStudents();
for (Student s:list)v {

// System.out.println("bc "+s.getBranch());
System.out.println("ID: " + s.getId());
System.out.println("name: " + s.getName());
System.out.println("branch: " + s.getBranch());
System.out.println("percent: " + s.getPercent());
System.out.println("email: " + s.getEmail());
}
// System.out.println(list);
} catch (Exception e) {
System.err.println("Client exception: " + e.toString());
e.printStackTrace();
}
}
}
```

Output :-

Javac *.java

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>javac *.java

C:\EXAMPLES\rmi>
```

start rmiregistry

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>javac *.java

C:\EXAMPLES\rmi>start rmiregistry

C:\EXAMPLES\rmi>
```

Java Server

```
C:\WINDOWS\system32\cmd.exe - java Server

C:\EXAMPLES\rmi>java Server
Server ready
```

java Client

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.14393]
(c) 2016 Microsoft Corporation. All rights reserved.

C:\Users\Tutorialspoint>cd C:\EXAMPLES\rmi

C:\EXAMPLES\rmi>java Client
ID: 1
name: Ram
branch: IT
percent: 85
email: ram123@gmail.com
ID: 2
name: Rahim
branch: EEE
percent: 95
email: rahim123@gmail.com
ID: 3
name: Robert
branch: ECE
percent: 90
email: robert123@gmail.com

C:\EXAMPLES\rmi>
```

Practical No. 4 (a)

Aim : Ceating RMI Database Application. Using MySQL create Library database.

Code :

Library.java

```
public class Library implements java.io.Serializable {
 private int BookID;
 private String BookName,BookAuthor;

 public int getBookId() {
 return BookID;
 }
 public String getBookName() {
 return BookName;
 }
 public String getBookAuthor() {
 return BookAuthor;
 }

 public void setBookID(int BookID) {
 this.BookID =BookID;
 }
 public void setBookName(String BookName) {
 this.BookName =BookName;
 }
 public void setBookAuthor(String BookAuthor) {
 this.BookAuthor = BookAuthor;
 }
}
```

Hello.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.*;
// Creating Remote interface for our application
public interface Hello extends Remote
{
 public List<Library> getBookInfo() throws Exception;
}
```

ImplExample.java

```
import java.sql. *;
import java.util.*;
// Implementing the remote interface
public class ImplExample implements Hello {

 // Implementing the interface method
 public List<Library> getBookInfo() throws Exception {
 List<Library> list = new ArrayList<Library>();

 // JDBC driver name and database URL
 String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
 String DB_URL = "jdbc:mysql://localhost:3306/Library";
// Database credentials
 String USER = "root";
 String PASS = "system";
```

```
Connection conn = null;
Statement stmt = null;

//Register JDBC driver
Class.forName("com.mysql.cj.jdbc.Driver").newInstance ();
//Open a connection
System.out.println("Connecting to a selected database...");
conn = DriverManager.getConnection(DB_URL, USER, PASS);
System.out.println("Connected database successfully...");

//Execute a query
System.out.println("Creating statement...");

stmt = conn.createStatement();
String sql = "SELECT * FROM Book";
ResultSet rs = stmt.executeQuery(sql);

//Extract data from result set
while(rs.next()) {
// Retrieve by column name
int id = rs.getInt("BookID");
String name = rs.getString("BookName");
String author = rs.getString("BookAuthor");
// Setting the values
Library info = new Library();
info.setBookID(id);
info.setBookName(name);
```

```
 info.setBookAuthor(author);
 list.add(info);
 }
 rs.close();
 return list;
 }
}
```

Server.java

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Server extends ImplExample {
 public Server() {}
 public static void main(String args[]) {
 try {
 // Instantiating the implementation class
 ImplExample obj = new ImplExample();

 // Exporting the object of implementation class (here we are
exporting the remote object to the stub)
 Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

 // Binding the remote object (stub) in the registry
Registry registry = LocateRegistry.createRegistry(6666);
 registry.rebind("bookinfo",stub);
 System.err.println("Server ready");
```

```
 } catch (Exception e) {
System.err.println("Server exception: " + e.toString()); e.printStackTrace();
}
 }
 }
```

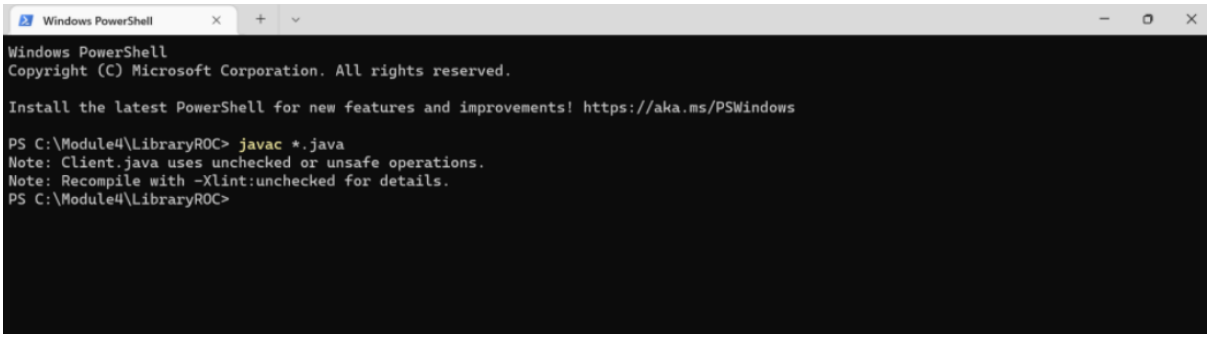
Client.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.*;
public class Client {
 private Client() {}
 public static void main(String[] args)throws Exception {
 try {
 // Getting the registry
 Registry registry =
LocateRegistry.getRegistry("192.168.0.101",6666);

 // Looking up the registry for the remote object
 Hello stub = (Hello) registry.lookup("bookinfo");
 // Calling the remote method using the obtained object
 List<Library> list = (List)stub.getBookInfo();
 for (Library l:list)
{
 System.out.println("Book ID: " + l.getBookId());
 System.out.println("Book Name: " + l.getBookName());
 System.out.println("Book Author: " + l.getBookAuthor());
 System.out.println("----------------------------------------------");
 }
```

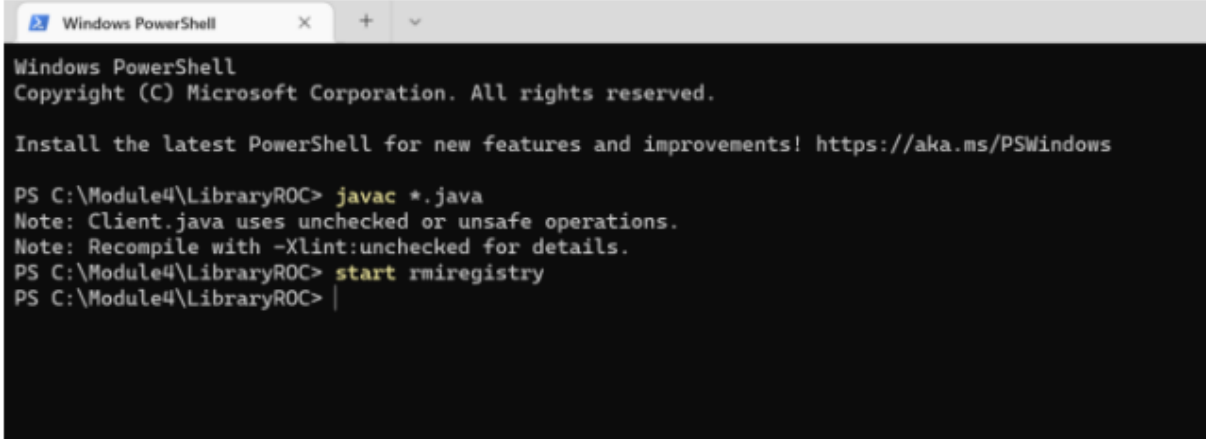
```
 } catch (Exception e) {
 System.err.println("Client exception: " + e.toString());
 e.printStackTrace();
 }
 }
}
```

Output :

javac *.java



start rmiregistry



java Server

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Module4\LibraryROC> javac *.java
Note: Client.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Module4\LibraryROC> start rmiregistry
PS C:\Module4\LibraryROC> java Server
Server ready
```

java Client

```
Windows PowerShell
PS C:\Module4\LibraryROC> java Client
Book ID: 1001
Book Name: Introduction to c
Book Author: Goutam Biswa
--------------------------------------------
Book ID: 1002
Book Name: Python track
Book Author: Mike Vanier
--------------------------------------------
Book ID: 1003
Book Name: Perl Programming
Book Author: Nathan Clement
--------------------------------------------
PS C:\Module4\LibraryROC>
```

Practical No. 4 (b)

Aim : Ceating RMI Database Application. Using MySQL create ElecrticBill database.

Code :

Electric.java

```
public class Electric implements java.io.Serializable {
 private float BillAmount;
 private String CustomerName,BillDueDate;

 public float getBillAmount() {
 return BillAmount;
 }
 public String getCustomerName() {
 return CustomerName;
 }
 public String getBillDueDate() {
 return BillDueDate;
 }
 public void setBillAmount(float BillAmount) {
 this.BillAmount =BillAmount;
 }
 public void setCustomerName(String CustomerName) {
 this.CustomerName =CustomerName;
 }
 public void setBillDueDate(String BillDueDate) {
 this.BillDueDate = BillDueDate;
 }
}
```

Hello.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.*;
// Creating Remote interface for our application
public interface Hello extends Remote
{
 public List<Electric> getBillInfo() throws Exception;
}
```

3) Developing the Implementation Class (Remote Object):

Create a class and implement the above created interface.

Here we are implementing the getBillInfo() method of the Remote interface. When you invoke this method, it retrieves the records of a table named Bill. Sets these values to the Electric class using its setter methods, adds it to a list object and returns that list.

ImplExample.java

```
import java.sql. *;
import java.util.*;

// Implementing the remote interface
public class ImplExample implements Hello {

 // Implementing the interface method
 public List<Electric> getBillInfo() throws Exception {
 List<Electric> list = new ArrayList<Electric>();

 // JDBC driver name and database URL
 String JDBC_DRIVER = "com.mysql.cj.jdbc.Driver";
```

```
String DB_URL = "jdbc:mysql://localhost:3306/electricbill";

// Database credentials
String USER = "root";
String PASS = "system";

Connection conn = null;
Statement stmt = null;

//Register JDBC driver
Class.forName("com.mysql.cj.jdbc.Driver").newInstance ();
//Open a connection
System.out.println("Connecting to a selected database...");
conn = DriverManager.getConnection(DB_URL, USER, PASS);
System.out.println("Connected database successfully...");

//Execute a query
System.out.println("Creating statement...");

stmt = conn.createStatement();
String sql = "SELECT * FROM Bill";
ResultSet rs = stmt.executeQuery(sql);
//Extract data from result set
while(rs.next()) {
// Retrieve by column name
float amount = rs.getFloat("BillAmount");
String name = rs.getString("CustomerName");
String Date = rs.getString("BillDueDate");
```

```
 // Setting the values
 Electric info = new Electric();
 info.setBillAmount(amount);
 info.setCustomerName(name);
 info.setBillDueDate(Date);
 list.add(info);
 }
 rs.close();
 return list;
 }
}
```

Server.java

```
import java.rmi.registry.Registry;
import java.rmi.registry.LocateRegistry;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class Server extends ImplExample {
 public Server() {}
 public static void main(String args[]) {
 try {
 // Instantiating the implementation class
 ImplExample obj = new ImplExample();

 // Exporting the object of implementation class (here we are
exporting the remote object to the stub)
 Hello stub = (Hello) UnicastRemoteObject.exportObject(obj, 0);

 // Binding the remote object (stub) in the registry
```

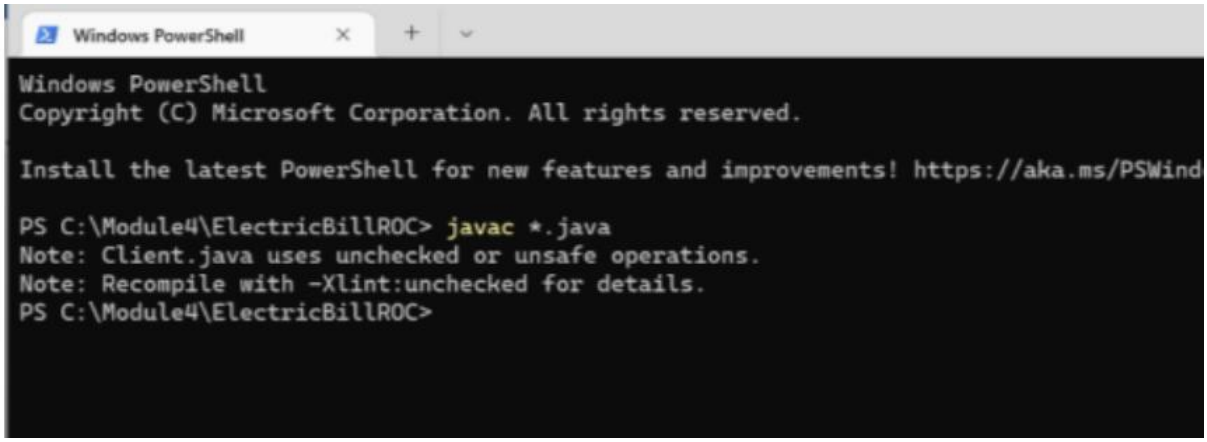
```
 Registry registry = LocateRegistry.createRegistry(6666);
 registry.rebind("billinfo",stub);
 System.err.println("Server ready");
 } catch (Exception e) {
 System.err.println("Server exception: " + e.toString());
 e.printStackTrace();
 }
 }
}
```

Client.java

```
import java.rmi.registry.LocateRegistry;
import java.rmi.registry.Registry;
import java.util.*;
public class Client {
 private Client() {}
 public static void main(String[] args)throws Exception {
 try {
 // Getting the registry
 Registry registry =
LocateRegistry.getRegistry("192.168.0.102",6666);

 // Looking up the registry for the remote object
 Hello stub = (Hello) registry.lookup("billinfo");
 // Calling the remote method using the obtained object
 List<Electric> list = (List)stub.getBillInfo();
 for (Electric l:list)
```

```
{
 System.out.println("Customer name: " + l.getCustomerName());
 System.out.println("Bill Due Date: " + l.getBillDueDate());
 System.out.println("Bill Amount: " + l.getBillAmount());
 System.out.println("----------------------------------------------");
 }
 // System.out.println(list);
 } catch (Exception e) {
 System.err.println("Client exception: " + e.toString());
 e.printStackTrace();
 }
 }
}
```

javac *.java



start rmiregistry

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Module4\ElectricBillROC> javac *.java
Note: Client.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Module4\ElectricBillROC> start rmiregistry
PS C:\Module4\ElectricBillROC>
```

java Server

```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Module4\ElectricBillROC> javac *.java
Note: Client.java uses unchecked or unsafe operations.
Note: Recompile with -Xlint:unchecked for details.
PS C:\Module4\ElectricBillROC> start rmiregistry
PS C:\Module4\ElectricBillROC> java Server
Server ready
```

java Client

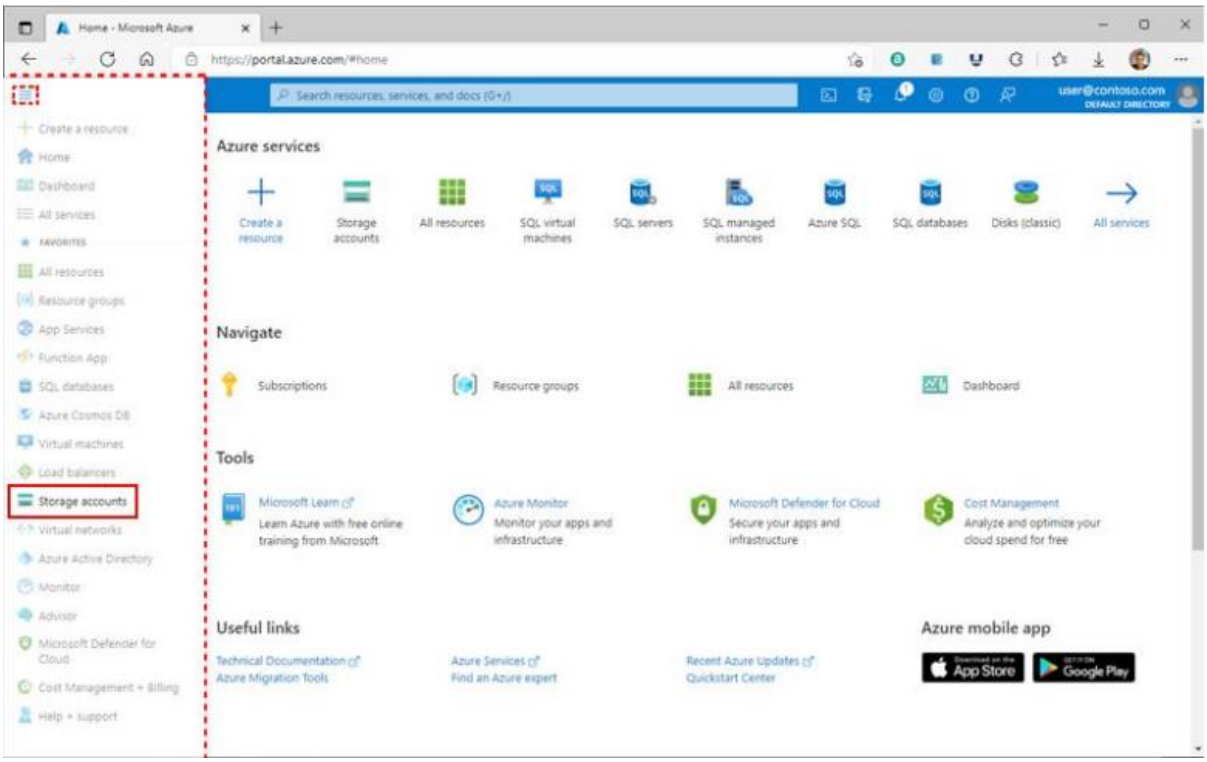
```
Windows PowerShell
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Module4\ElectricBillROC> java Client
Customer name: XYZ
Bill Due Date: 12-Aug-2020
Bill Amount: 2000.0
---------------------------------------------
Customer name: PQR
Bill Due Date: 22-Aug-2020
Bill Amount: 1500.0
---------------------------------------------
Customer name: LMN
Bill Due Date: 23-Sep-2020
Bill Amount: 2500.0
---------------------------------------------
Customer name: ABC
Bill Due Date: 14-Oct-2020
Bill Amount: 3000.0
---------------------------------------------
PS C:\Module4\ElectricBillROC>
```

Practical No. 5 (a)

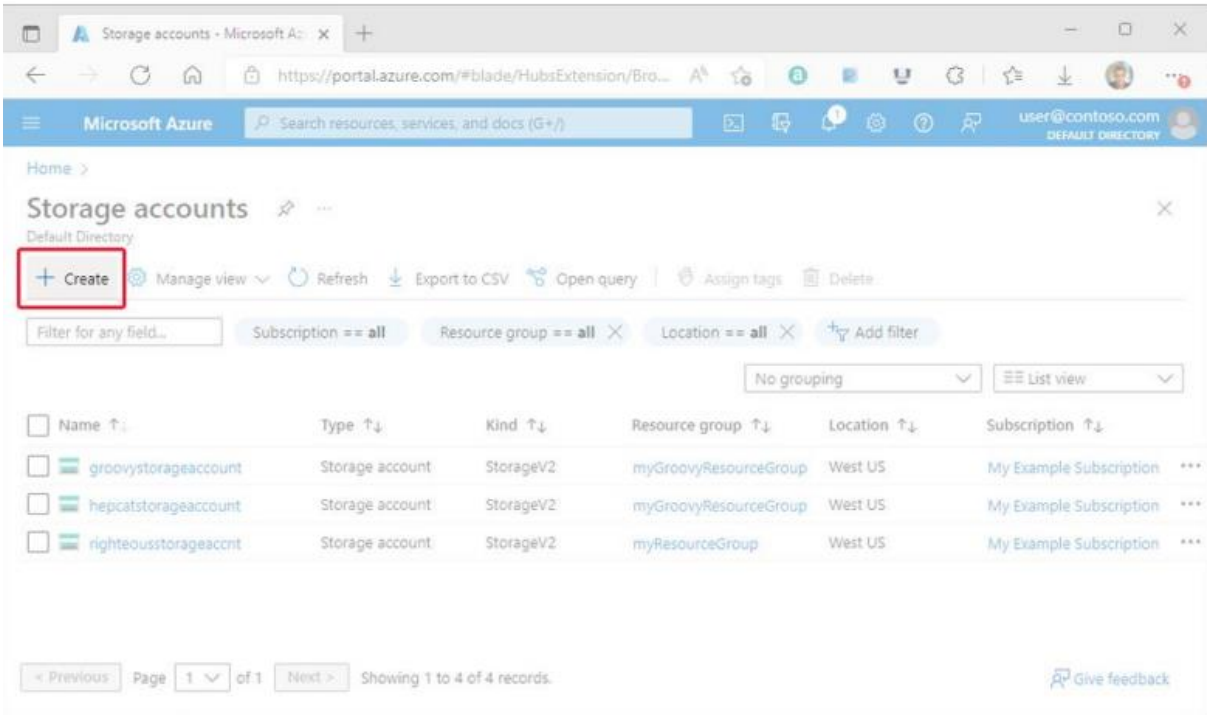
Aim : Infrastructure as a service (IaaS) Create a storage account.

To create an Azure storage account with the Azure portal, follow these steps:

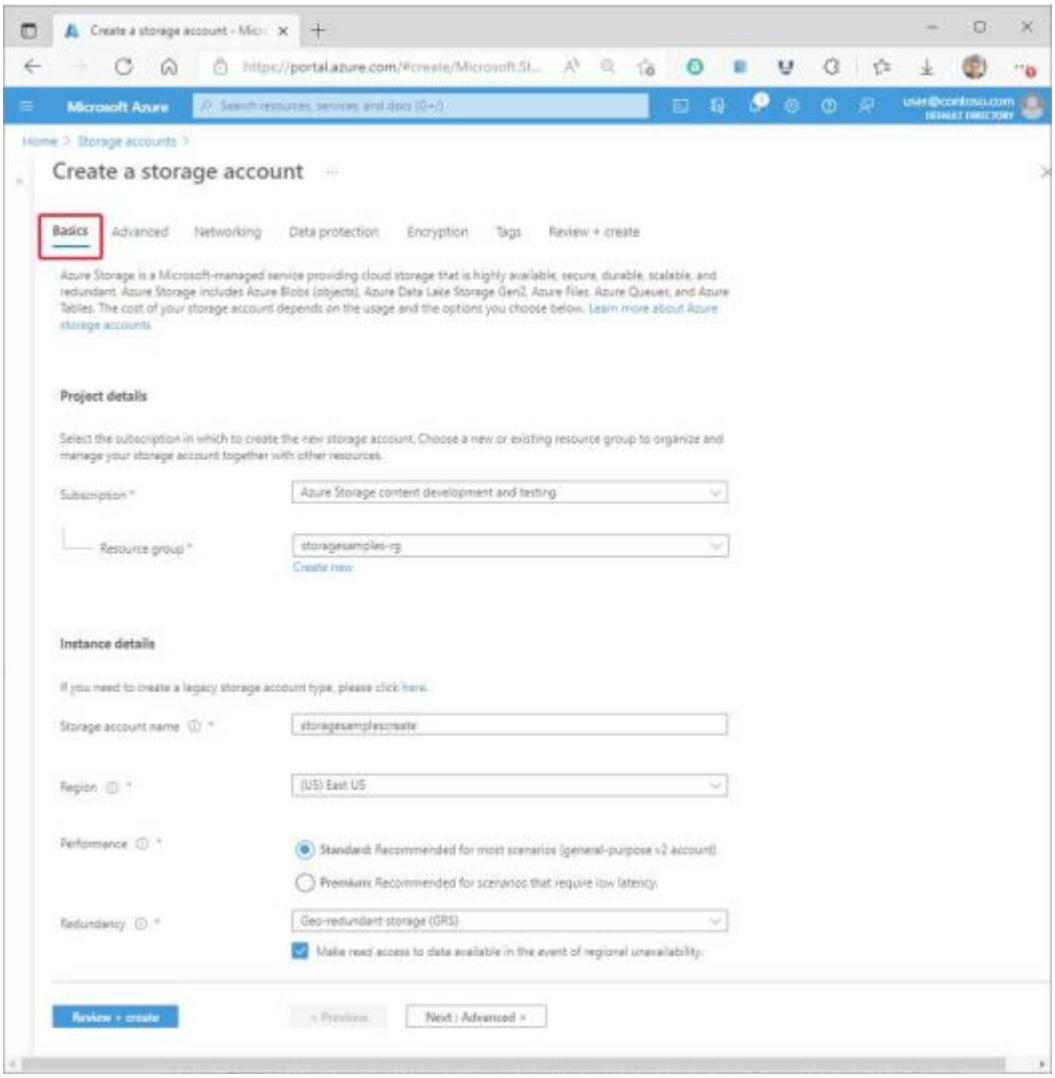
1. From the left portal menu, select Storage accounts to display a list of your storage accounts. If the portal menu isn't visible, click the menu button to toggle it on



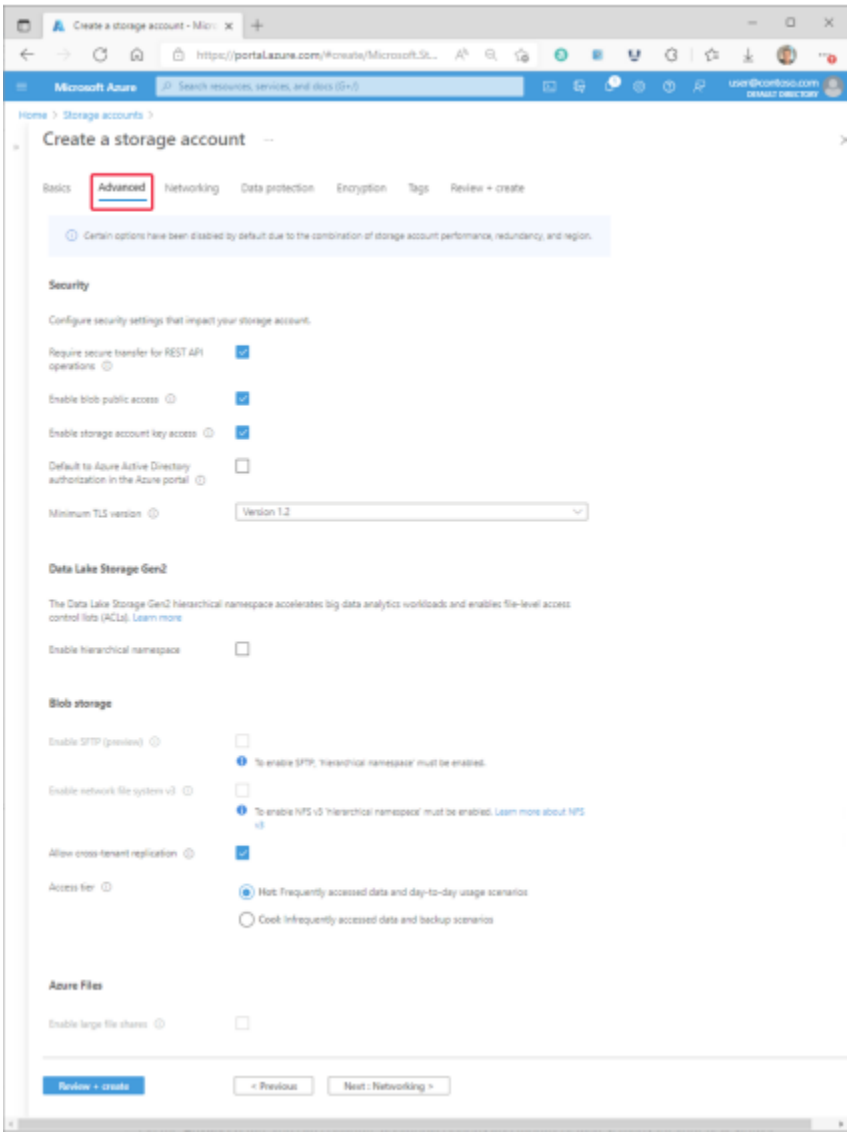
2. On the Storage accounts page, select Create.



The following image shows a standard configuration of the basic properties for a new storage account



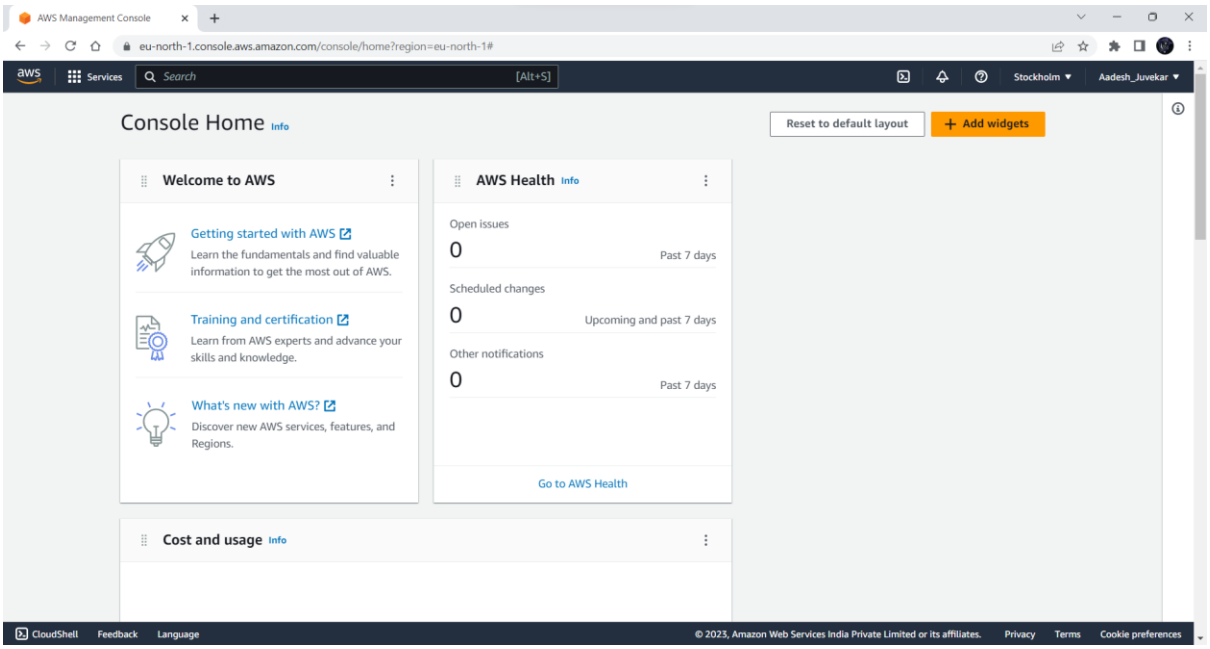
The following image shows a standard configuration of the advanced properties for a new storage account.


Create a storage account
Basics
Advanced
Networking
Data protection
Encryption
Tags
Review + create
Security
Data Lake Storage Gen2
Blob storage
Azure Files
Review + create
< Previous
Next : Networking >

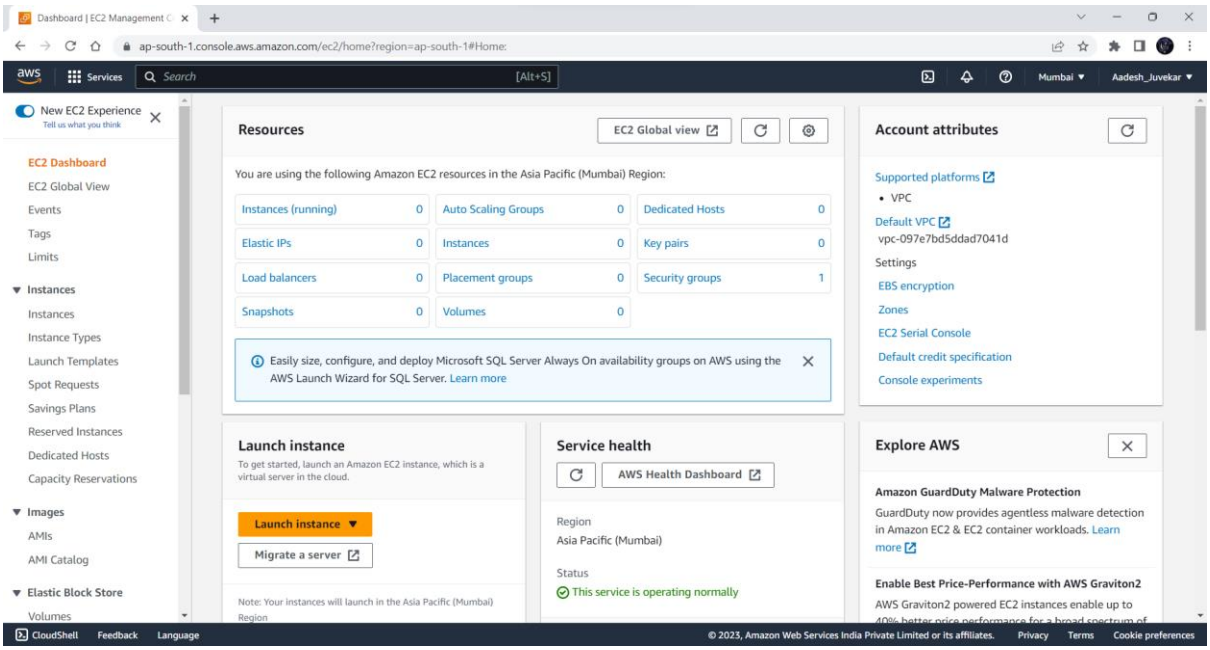
Practical No. 5 (b)

Aim : Create and EC2 instance with Windows and host an HTML Page

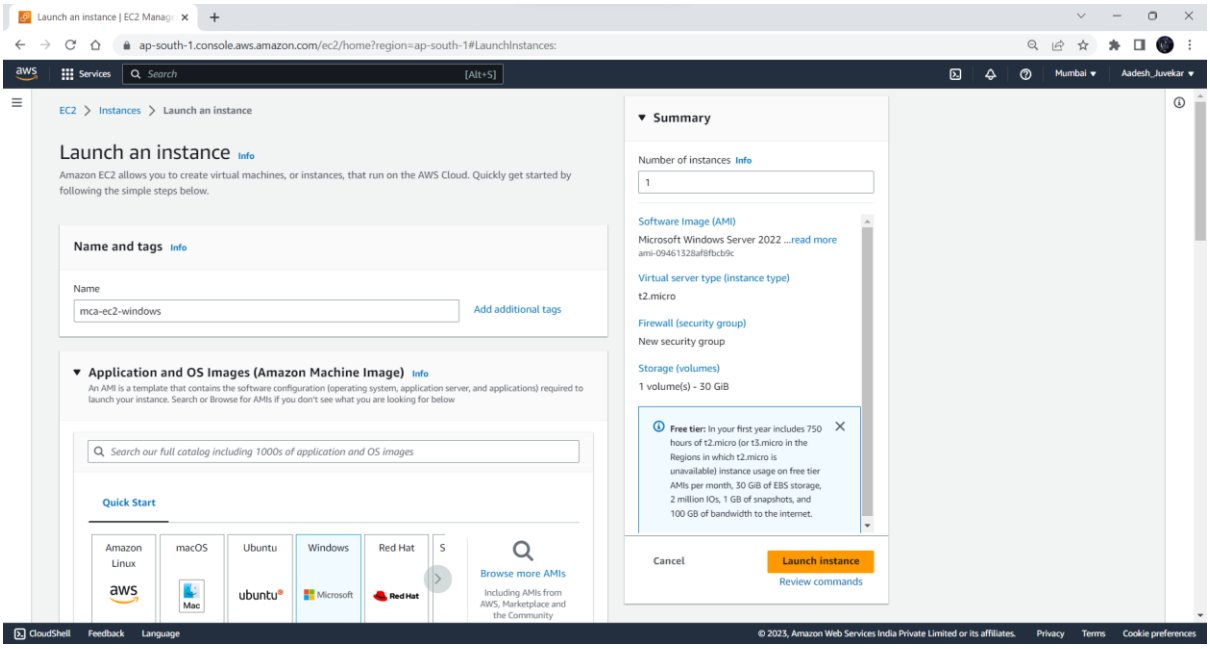
Step 1: Login to AWS Management Console https://console.aws.amazon.com/



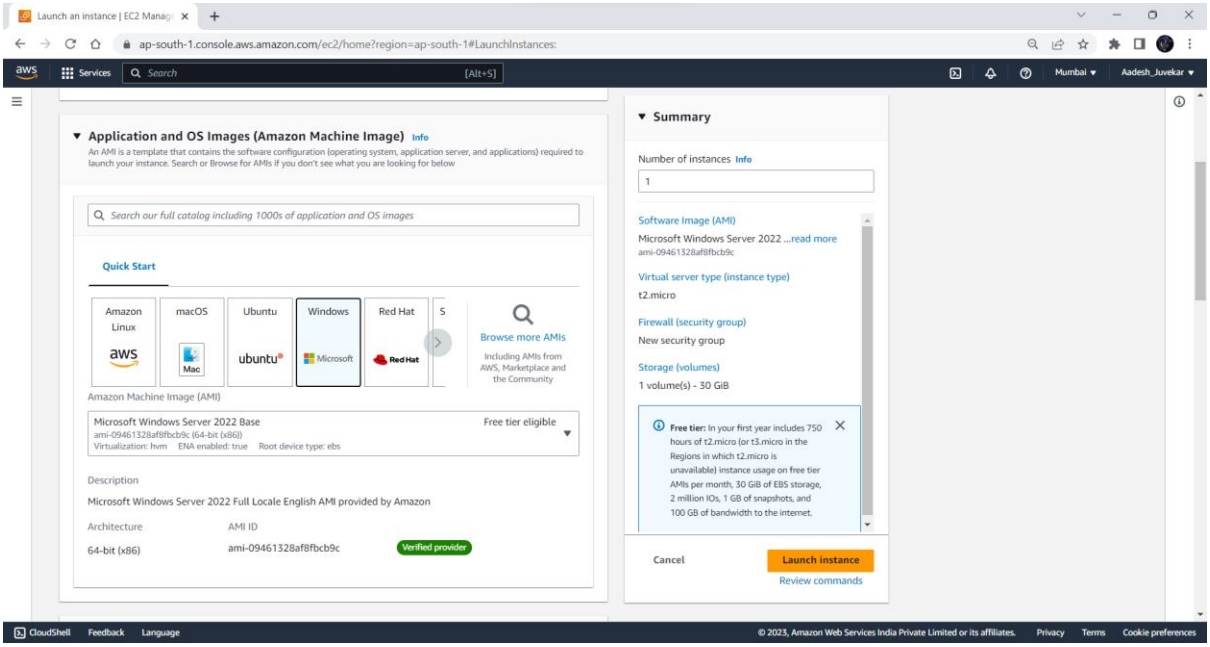
Step 2: Navigate to EC2 https://console.aws.amazon.com/ec2



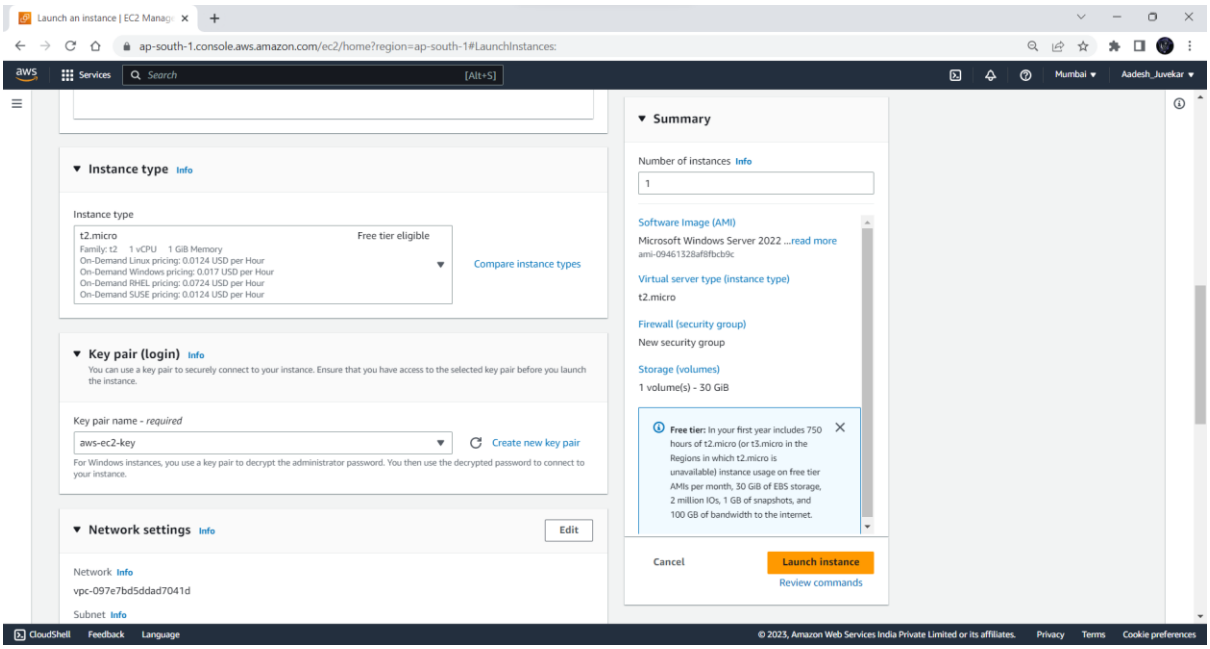
Step 3: Click on Launch Instance to create a new EC2 Instance



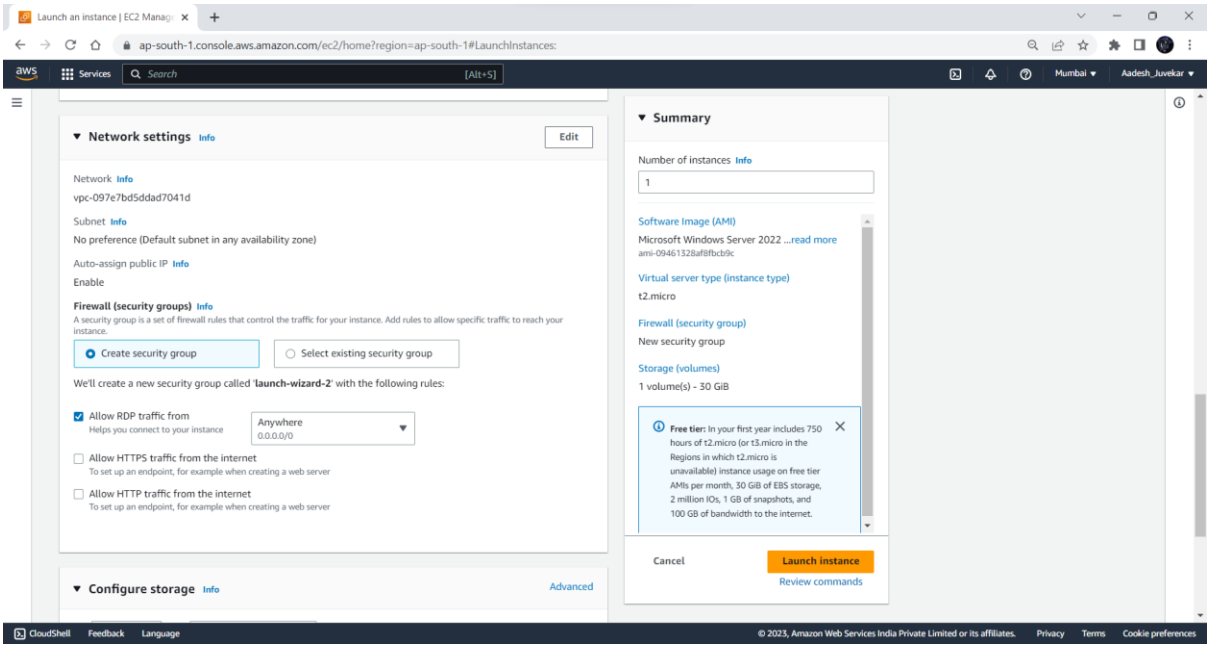
Step 4: Select Windows OS Image and select version and architecture



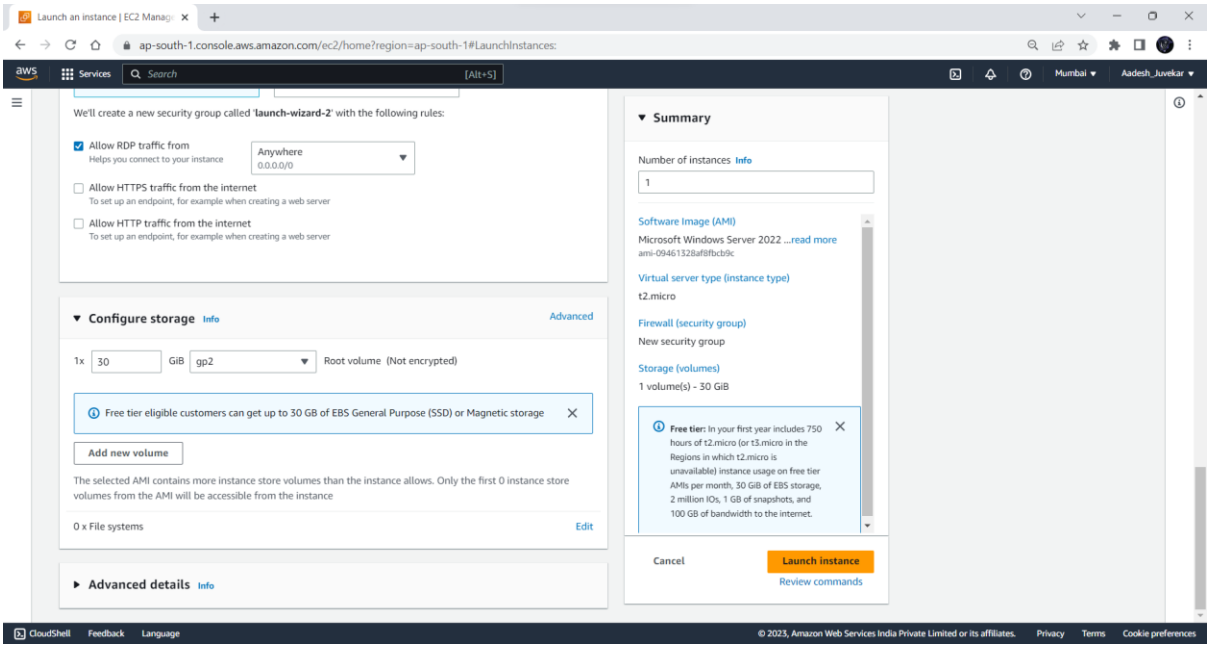
Step 5: Select Instance type and select existing Key-Pair

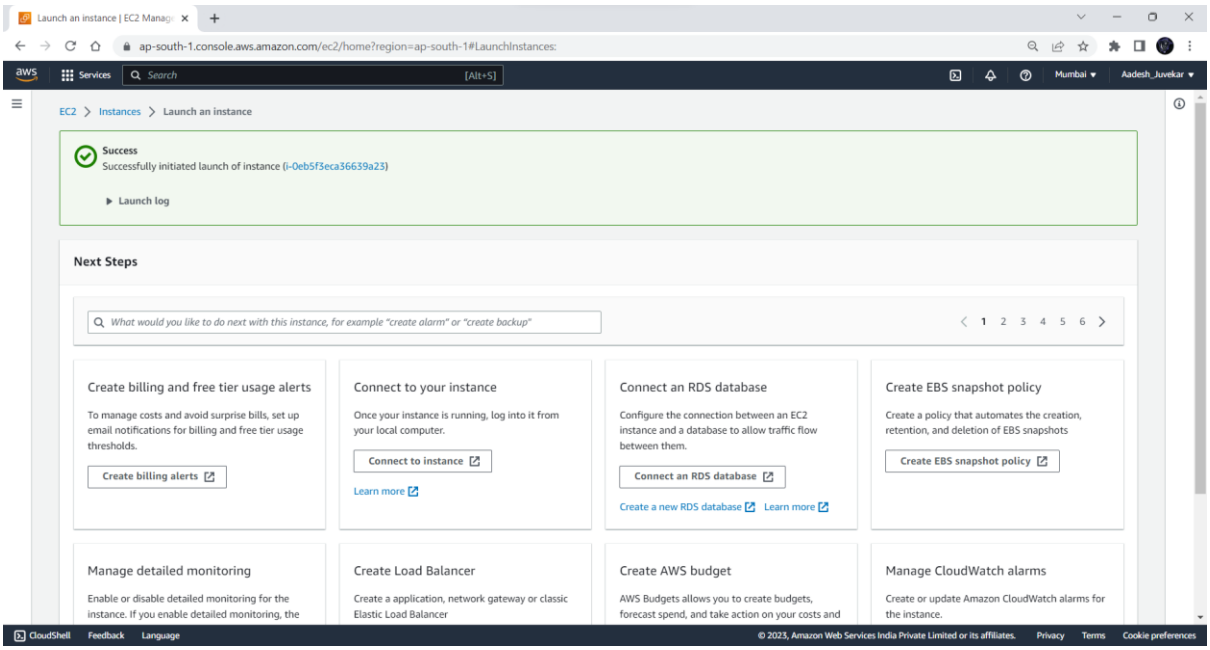


Step 6: Configure Network Settings and Security groups policies

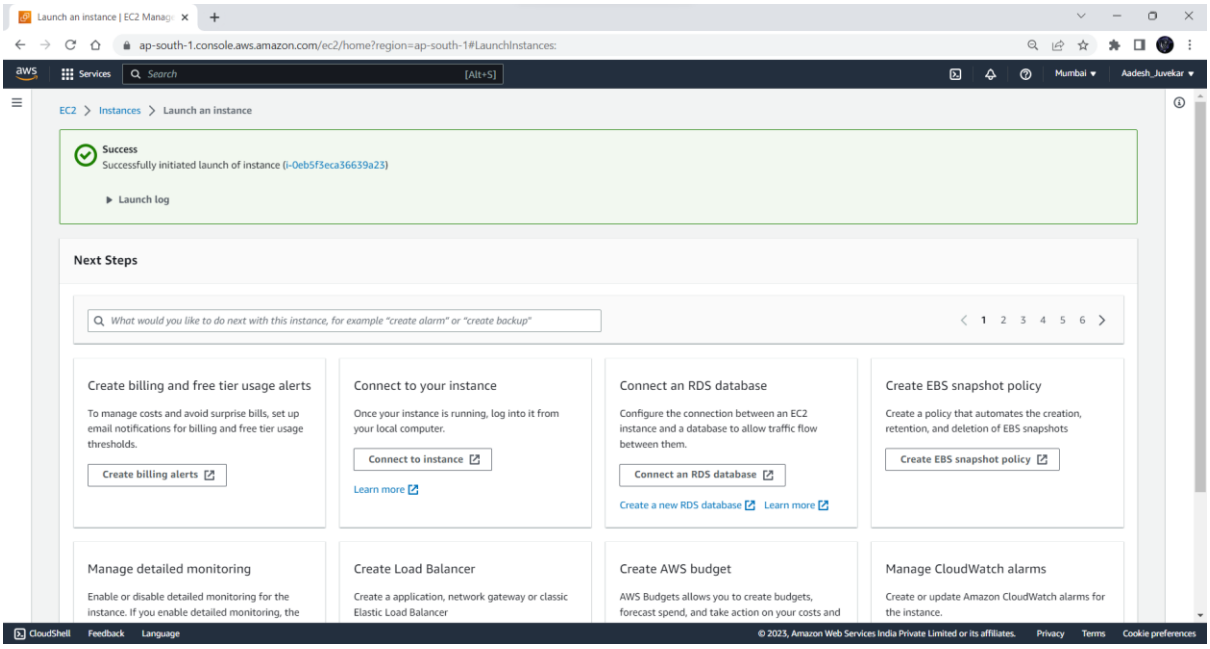


Step 7: Configure storage and then click on Launch Instance to create EC2 instance.

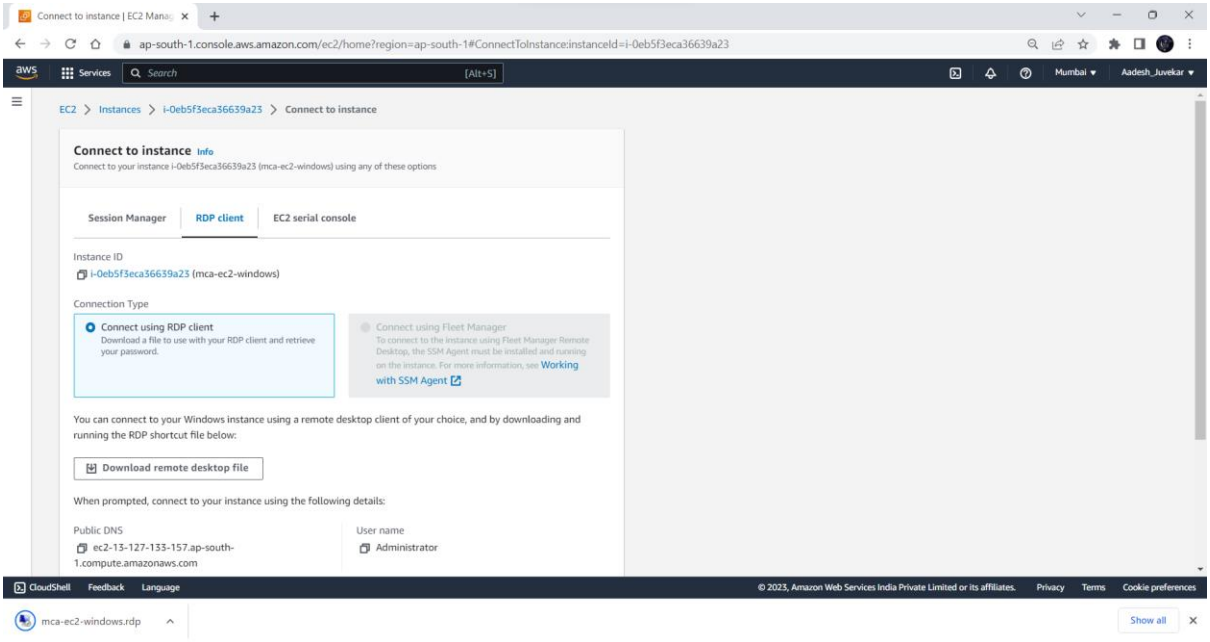




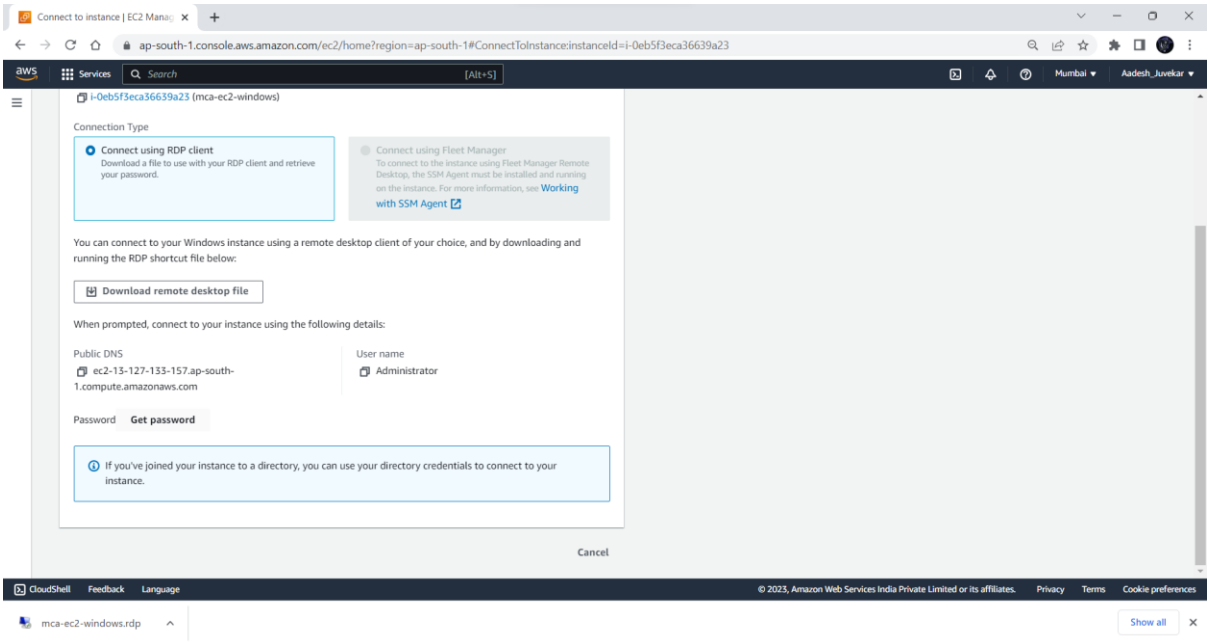
Step 8: Navigate to EC2 instances and select the newly created instance and click on Connect.

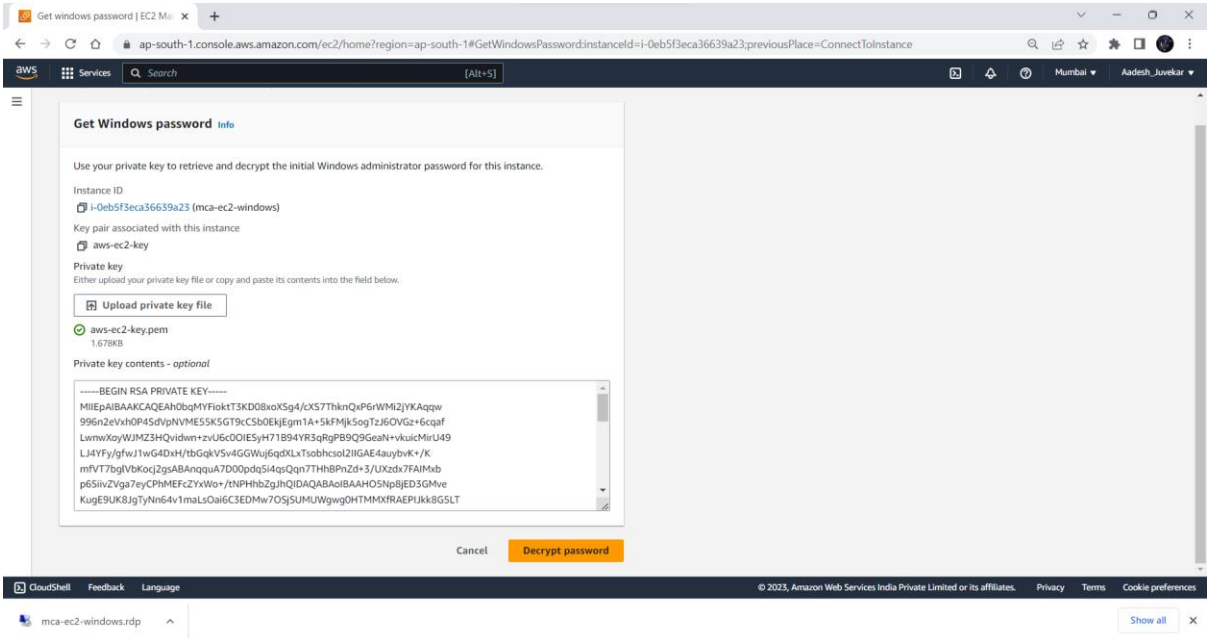


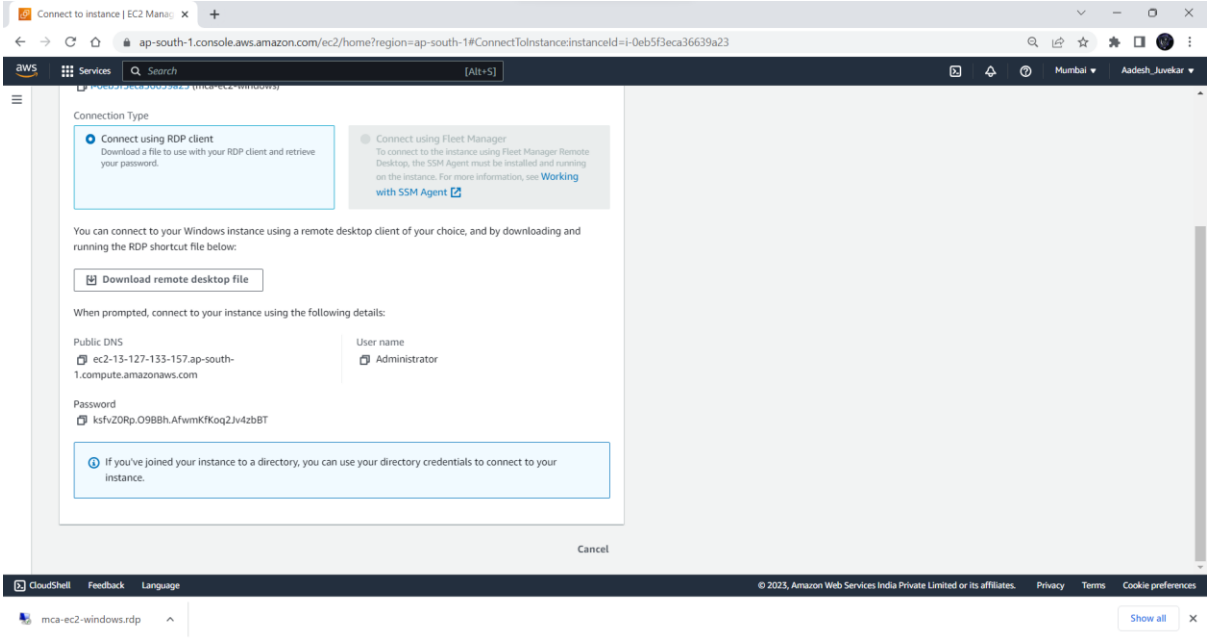
Step 9: Select RDP Client and download Remote Desktop File



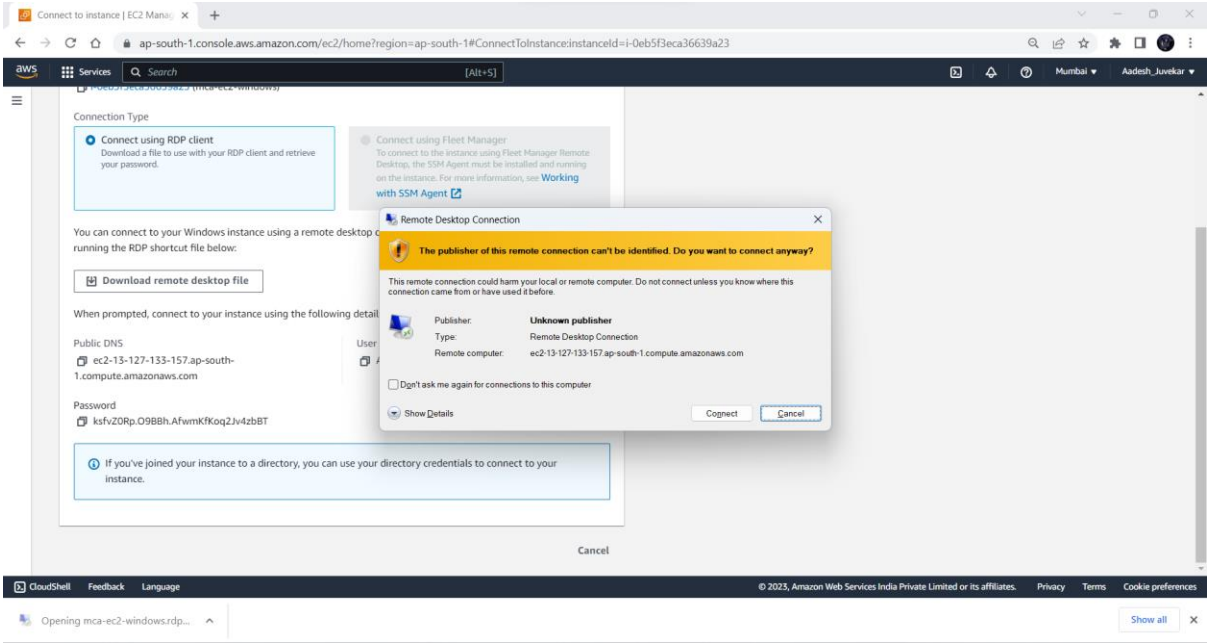
Step 10: Click on Get Password and upload the private key file we used while creating instance and click on Decrypt Password.



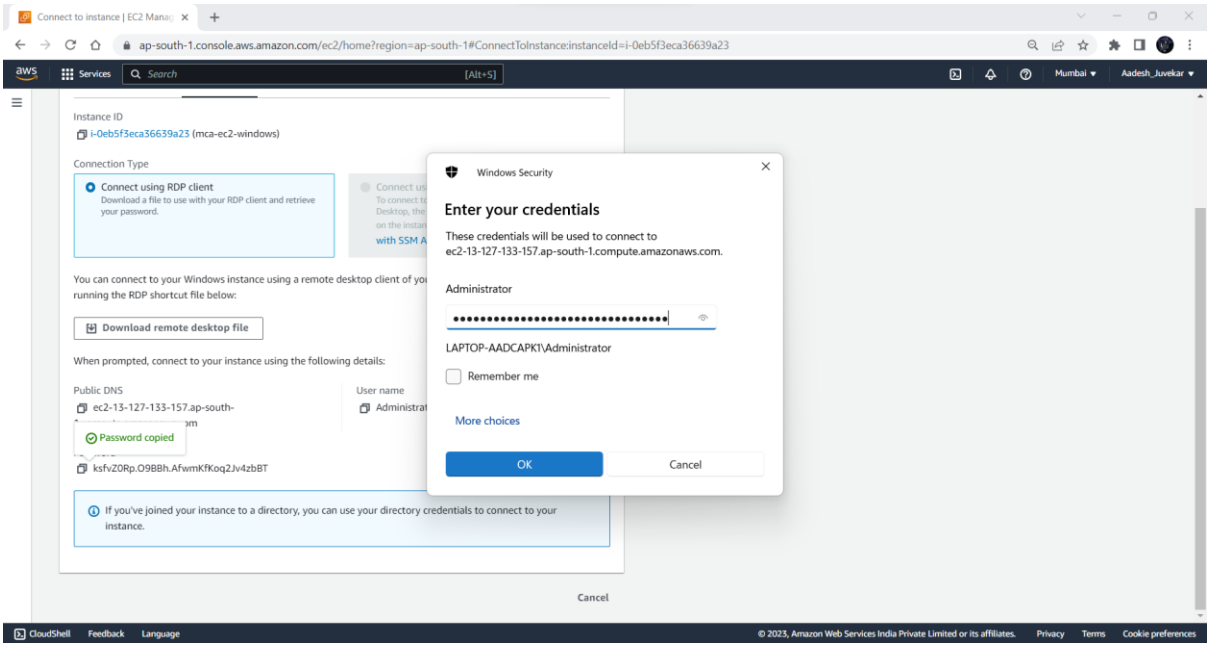


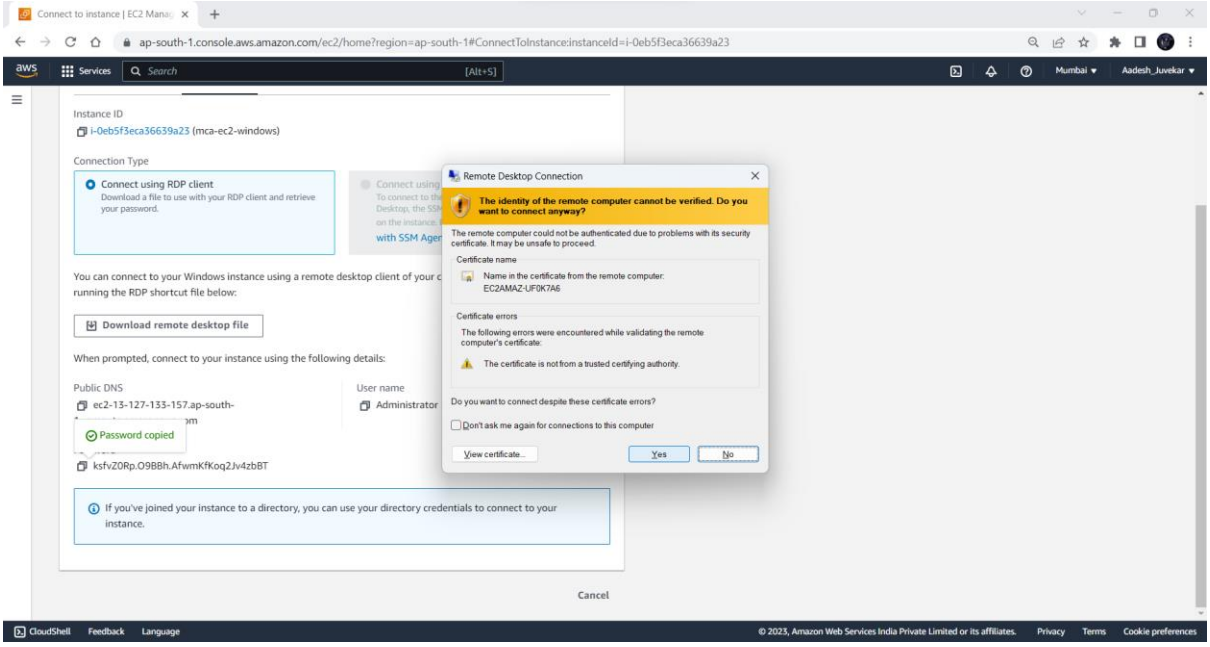


Step 11: Open the Remote Desktop File and click on connect

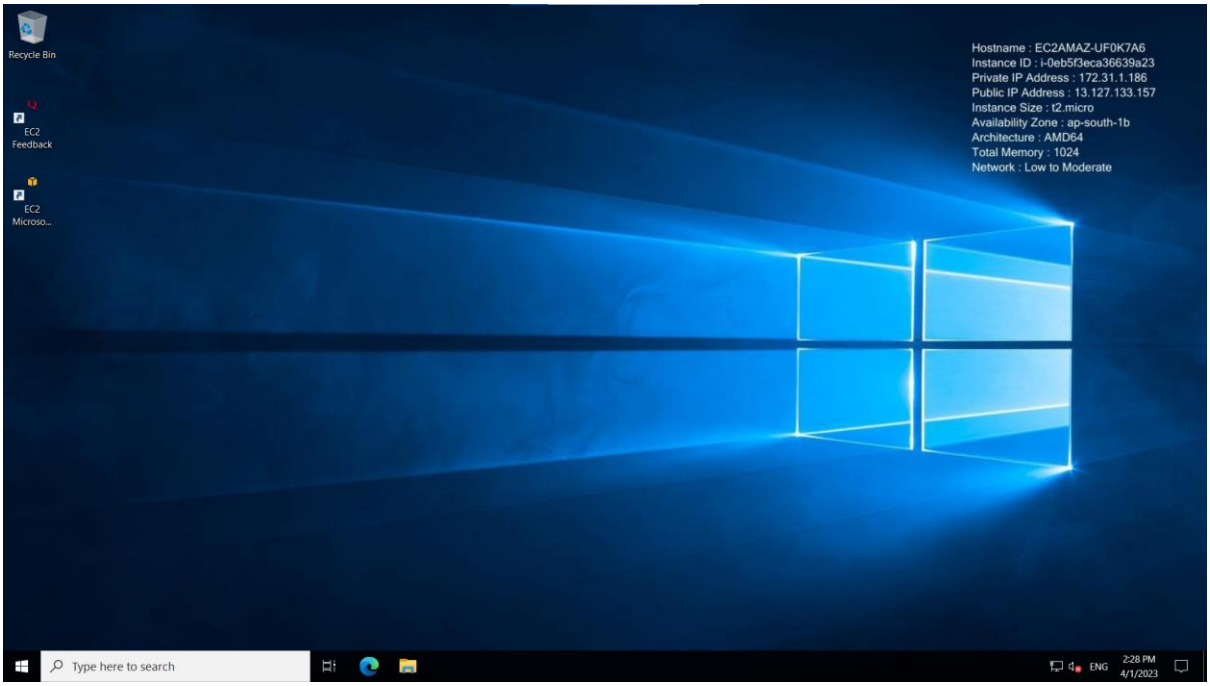


Step 12: Enter the given password and click on OK and then click on Yes





Step 13: We have successfully connected to EC2 instance



Practical No. 6

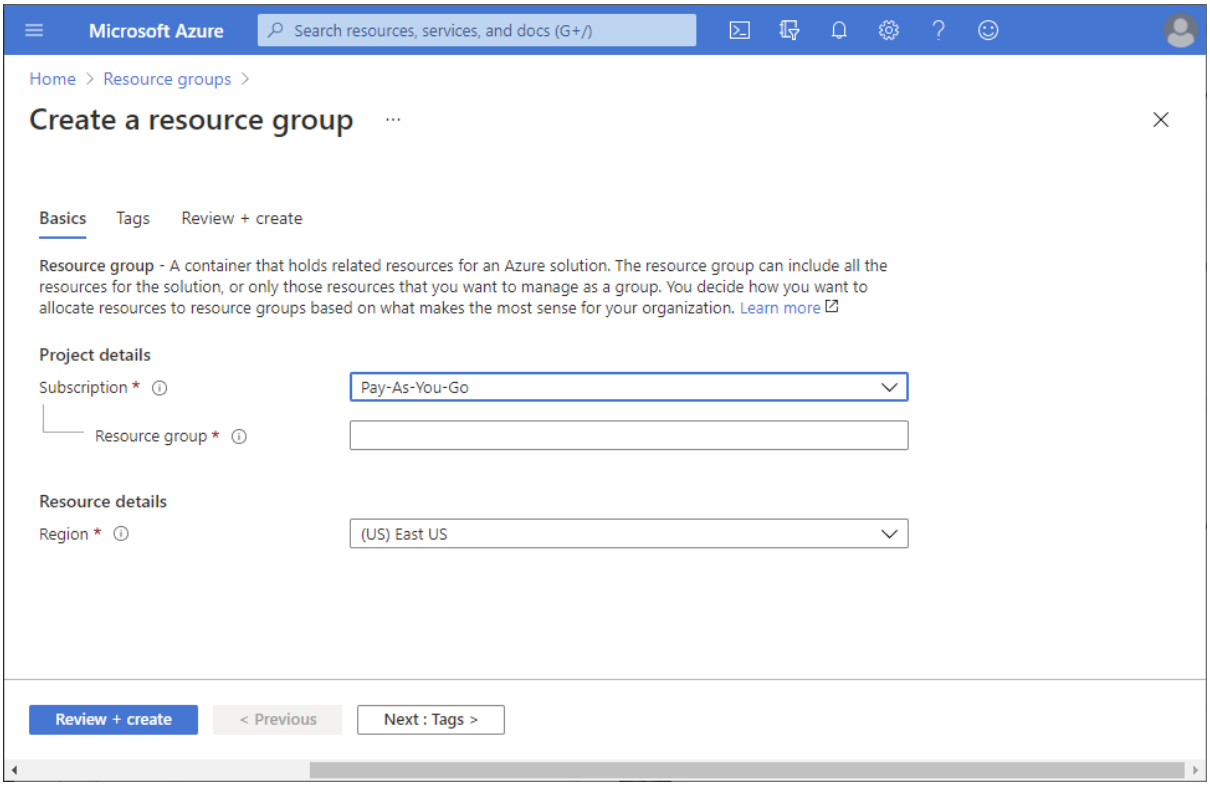
Aim : Create IAM AND USER ASSIGN ROLE.

Sign in to Azure

Step1 : Sign in to the Azure portal at https://portal.azure.com.

Create a resource group

- In the navigation list, click Resource groups.
- Click New to open the Create a resource group page.

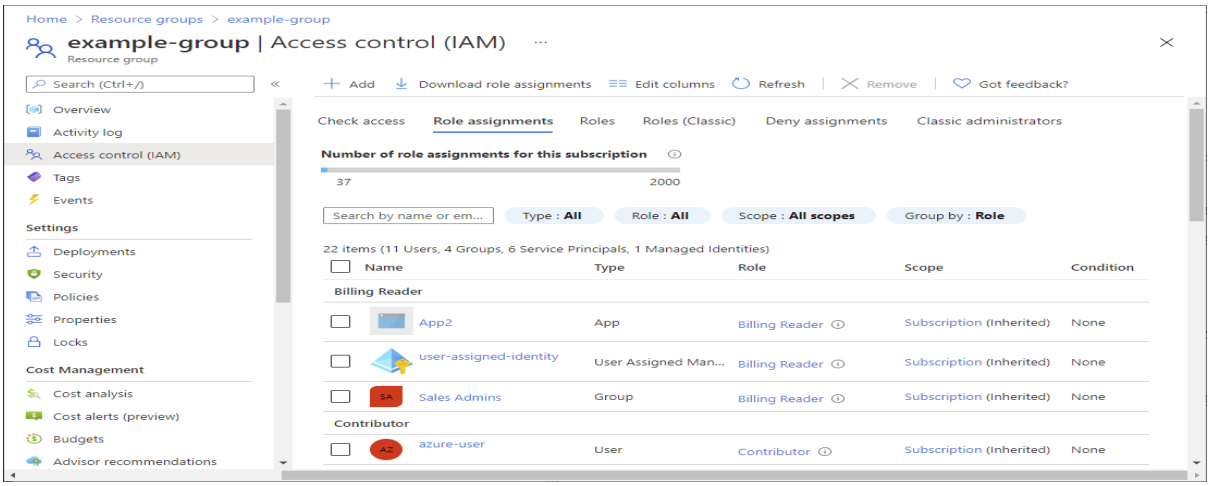


- Select a subscription.
- For Resource group name, enter example-group or another name.
- Click Review + create and then click Create to create the resource group.
- Click Refresh to refresh the list of resource groups.
- The new resource group appears in your resource groups list.

Step 2: Grant access

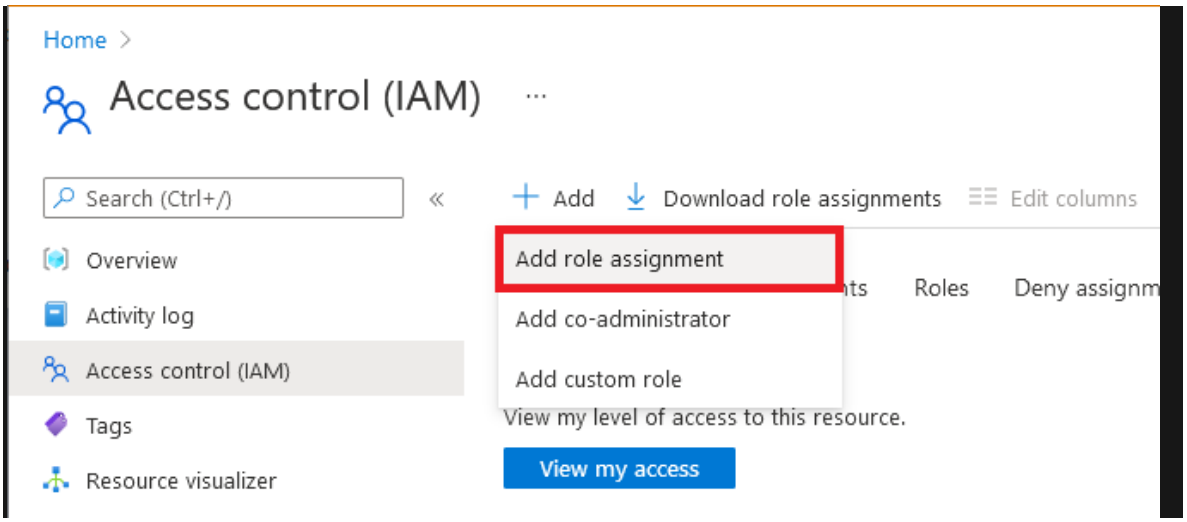
- In Azure RBAC, to grant access, you assign an Azure role.
- In the list of Resource groups, open the new example-group resource group.
- In the navigation menu, click Access control (IAM).

- Click the Role assignments tab to see the current list of role assignments.

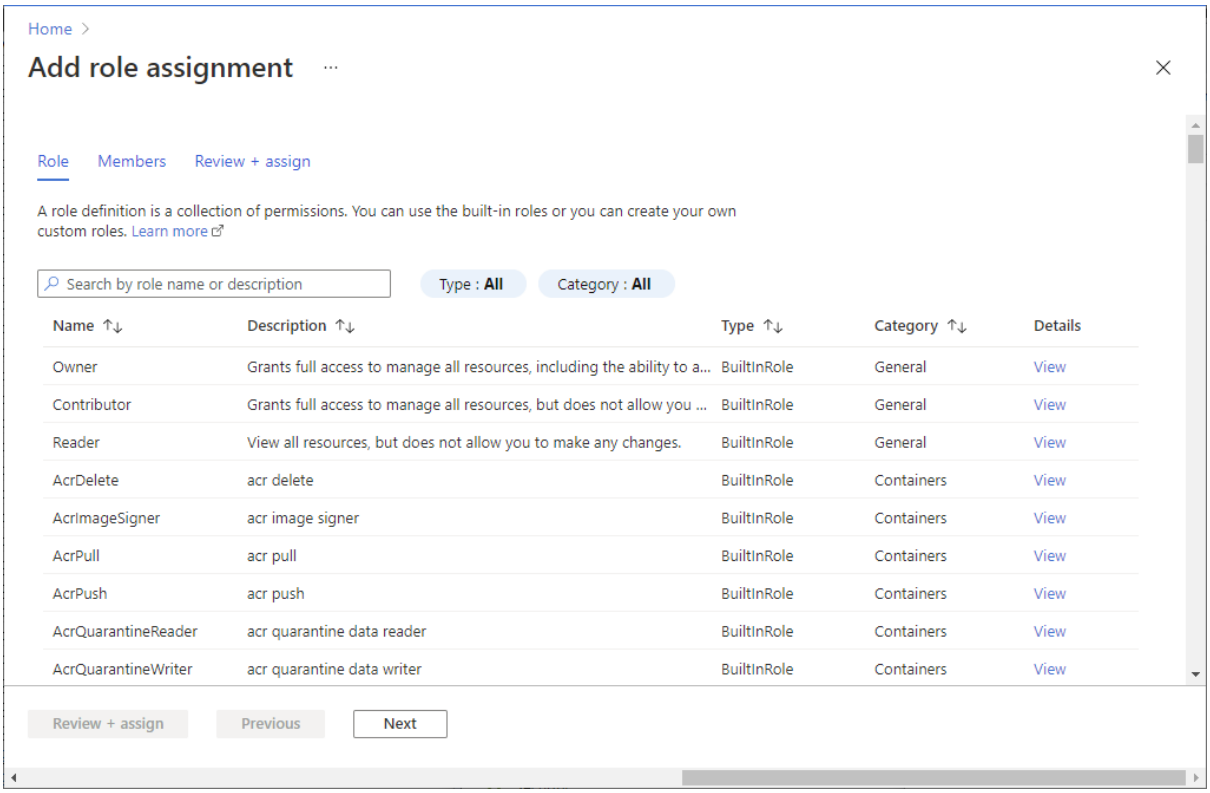


- Click Add > Add role assignment.

If you don't have permissions to assign roles, the Add role assignment option will be disabled.



On the Role tab, select the Virtual Machine Contributor role.

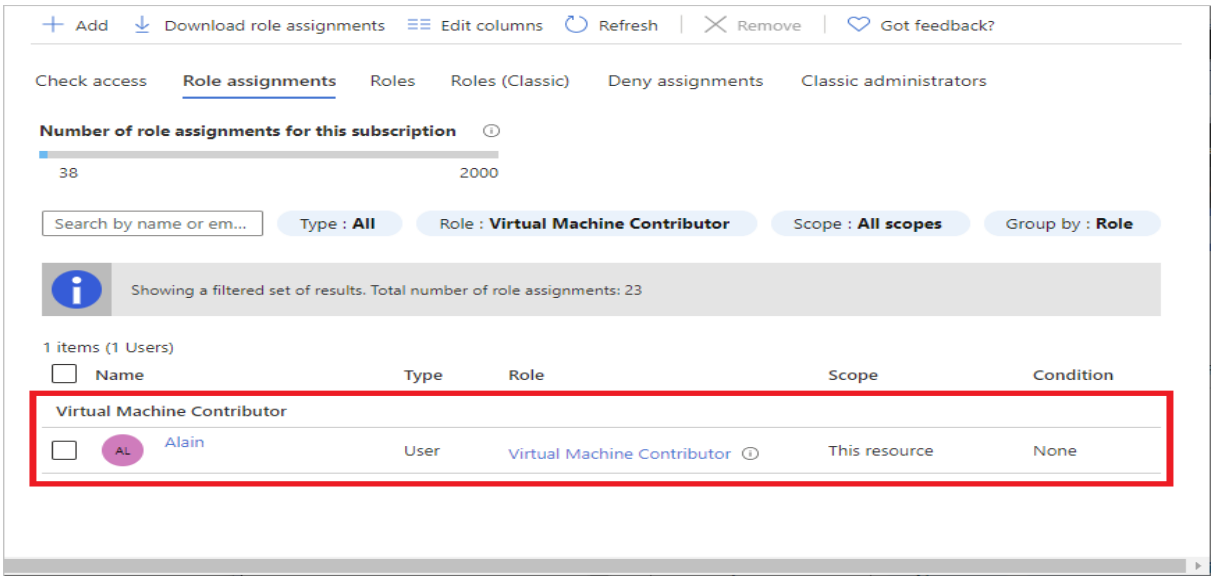


On the Members tab, select yourself or another user.

On the Review + assign tab, review the role assignment settings.

Click Review + assign to assign the role.

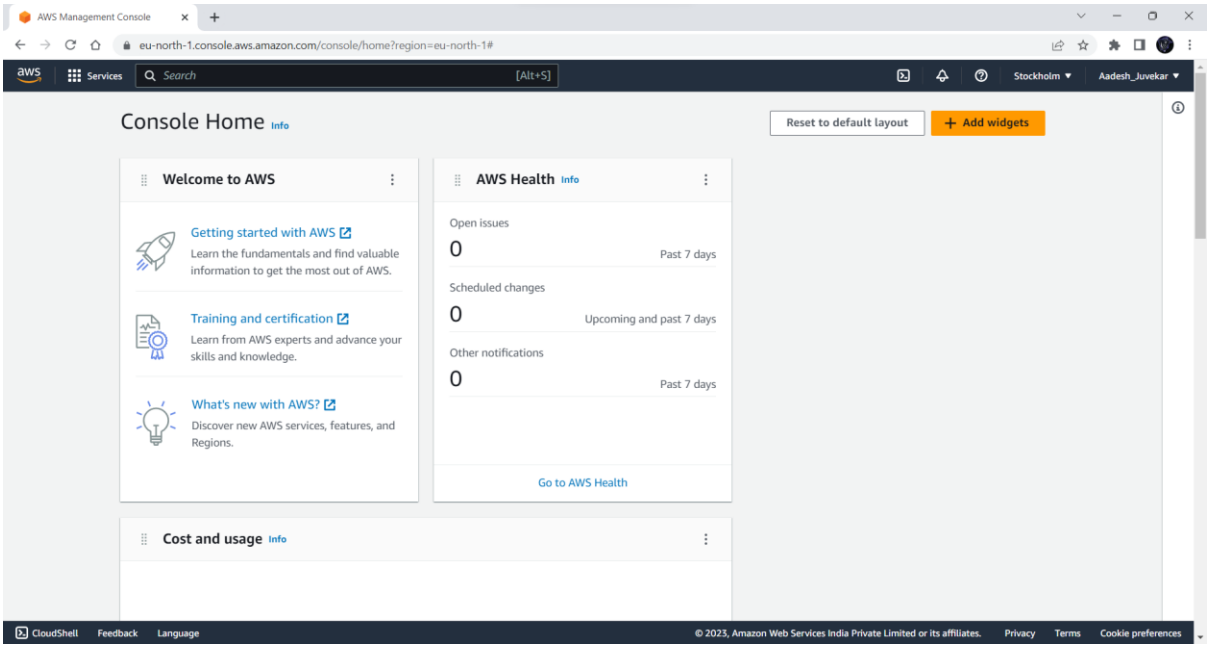
After a few moments, the user is assigned the Virtual Machine Contributor role at the example-group resource group scope.



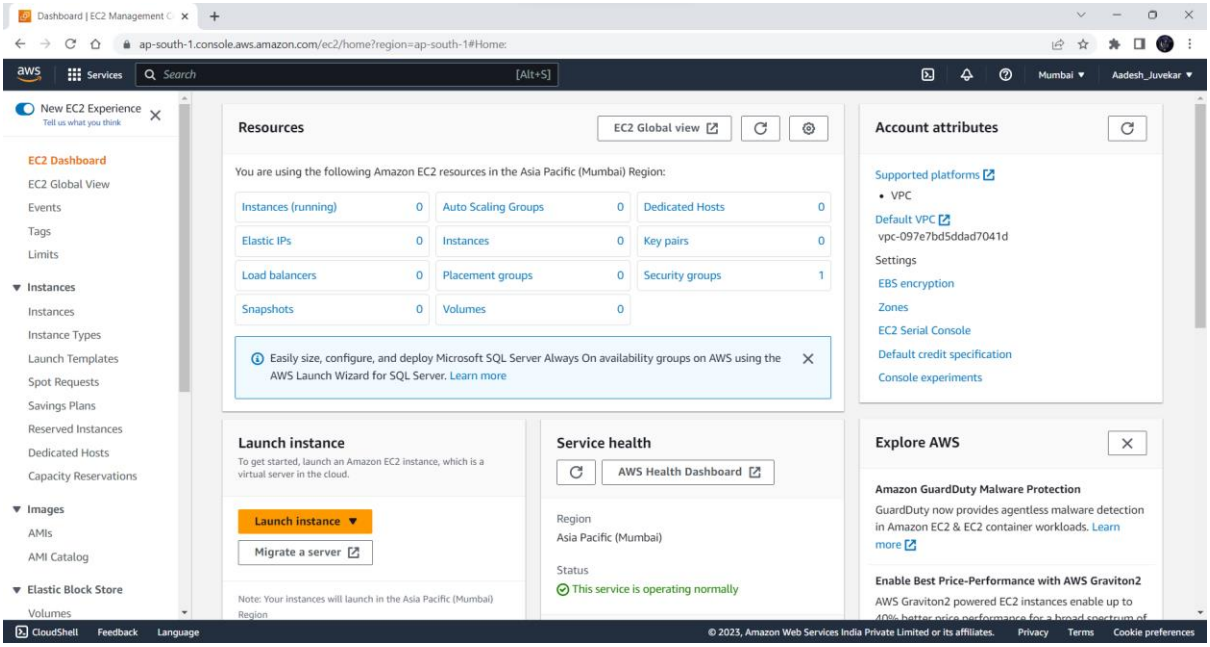
Practical No. 7

Aim : Create and EC2 instance with Windows and host Dynamically changing the background color of a webpage on each click.

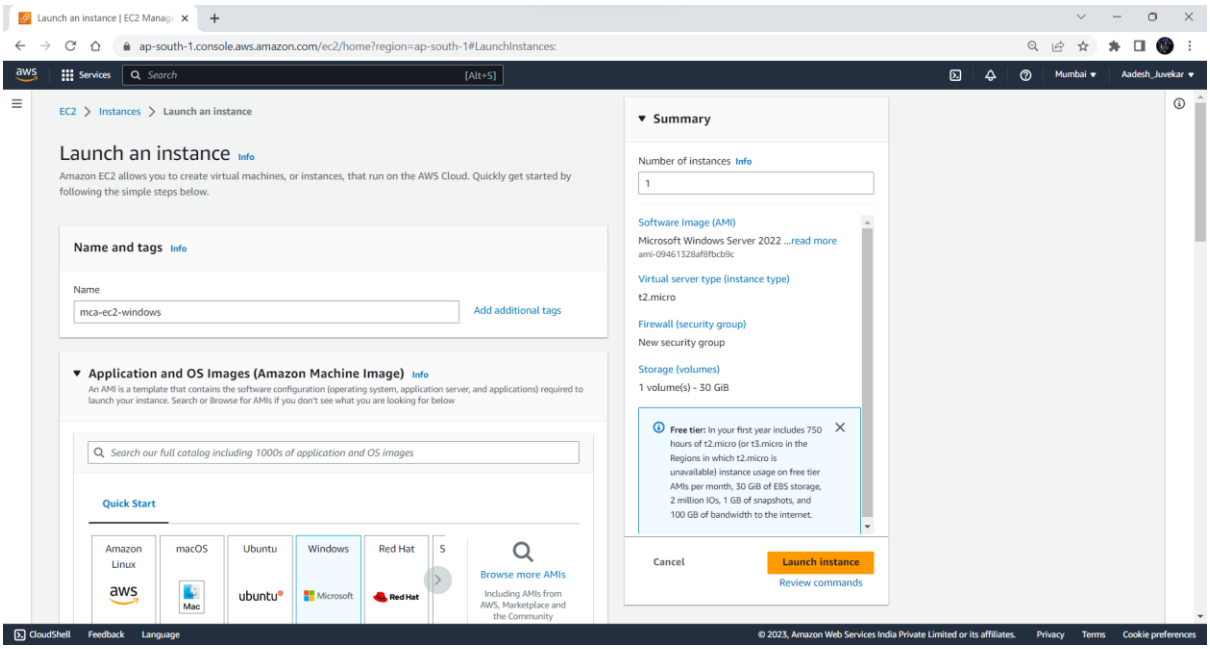
Step 1: Login to AWS Management Console https://console.aws.amazon.com/



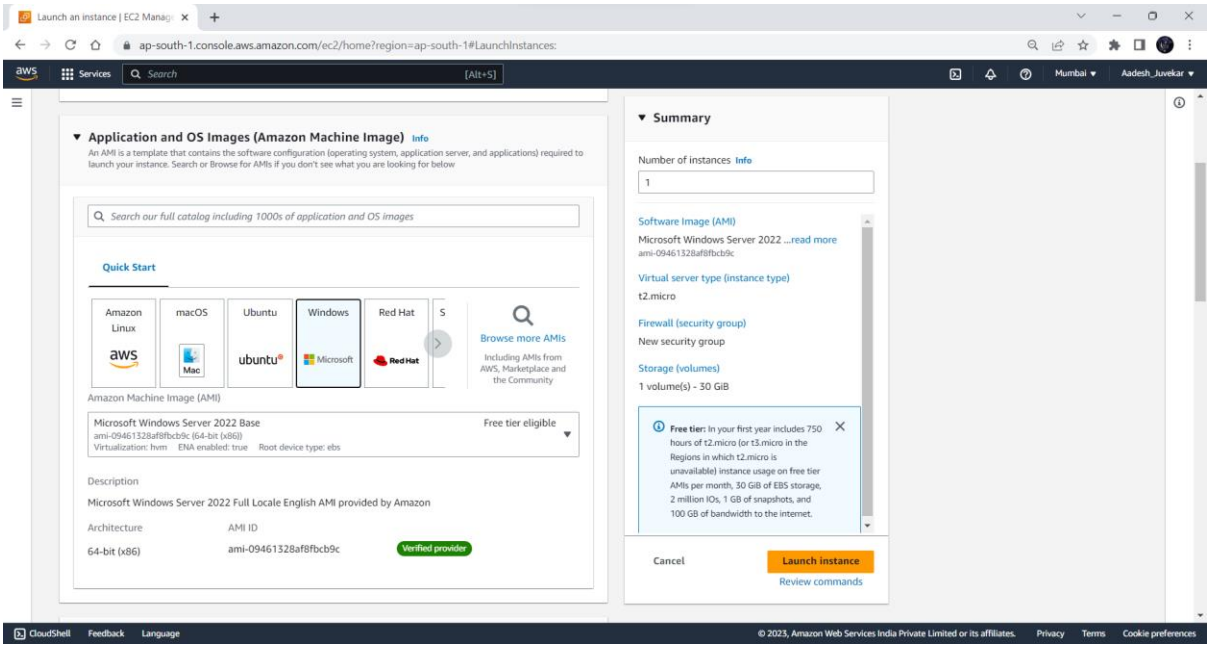
Step 2: Navigate to EC2 https://console.aws.amazon.com/ec2



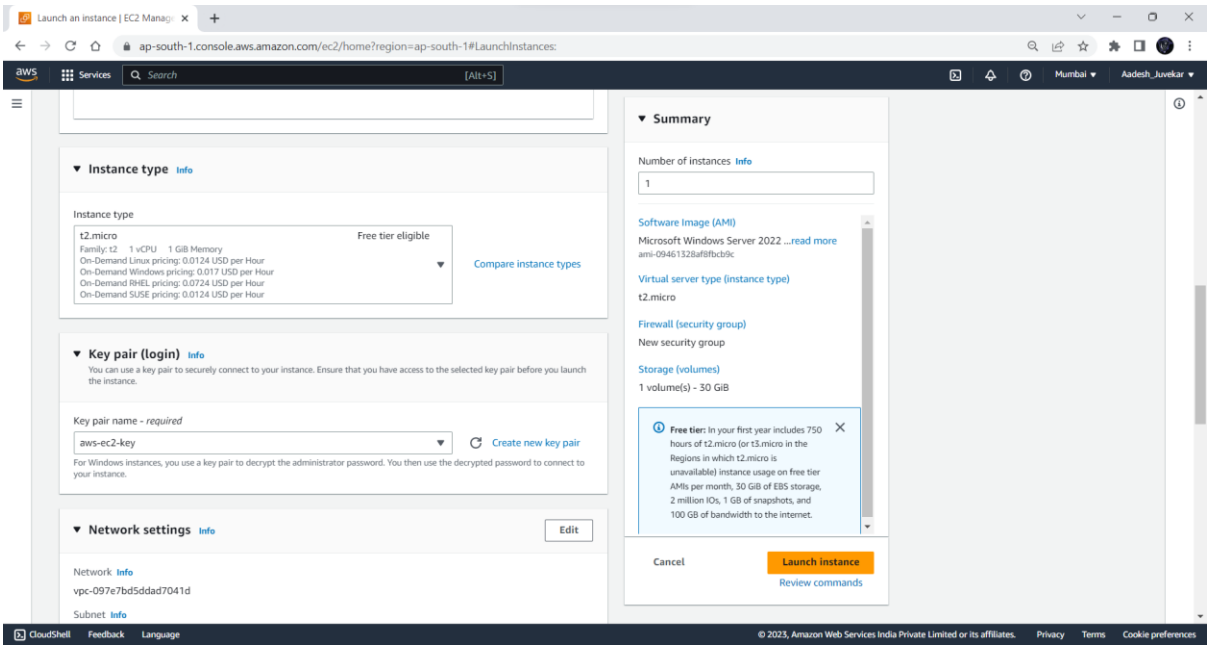
Step 3: Click on Launch Instance to create a new EC2 Instance



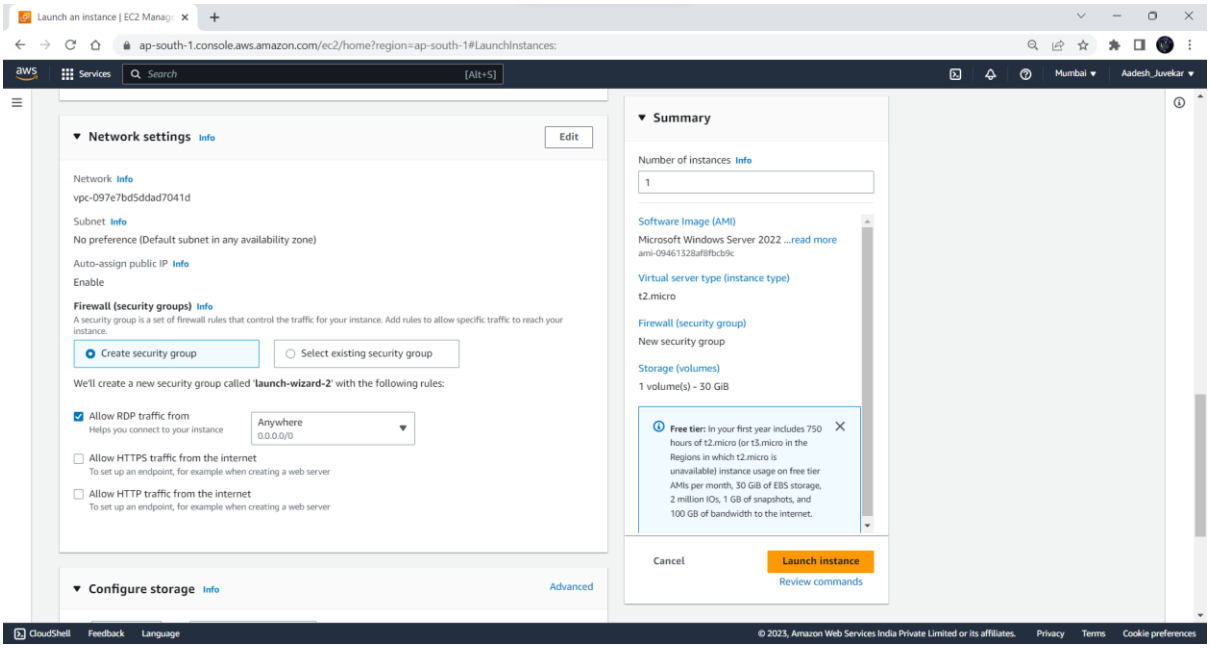
Step 4: Select Windows OS Image and select version and architecture



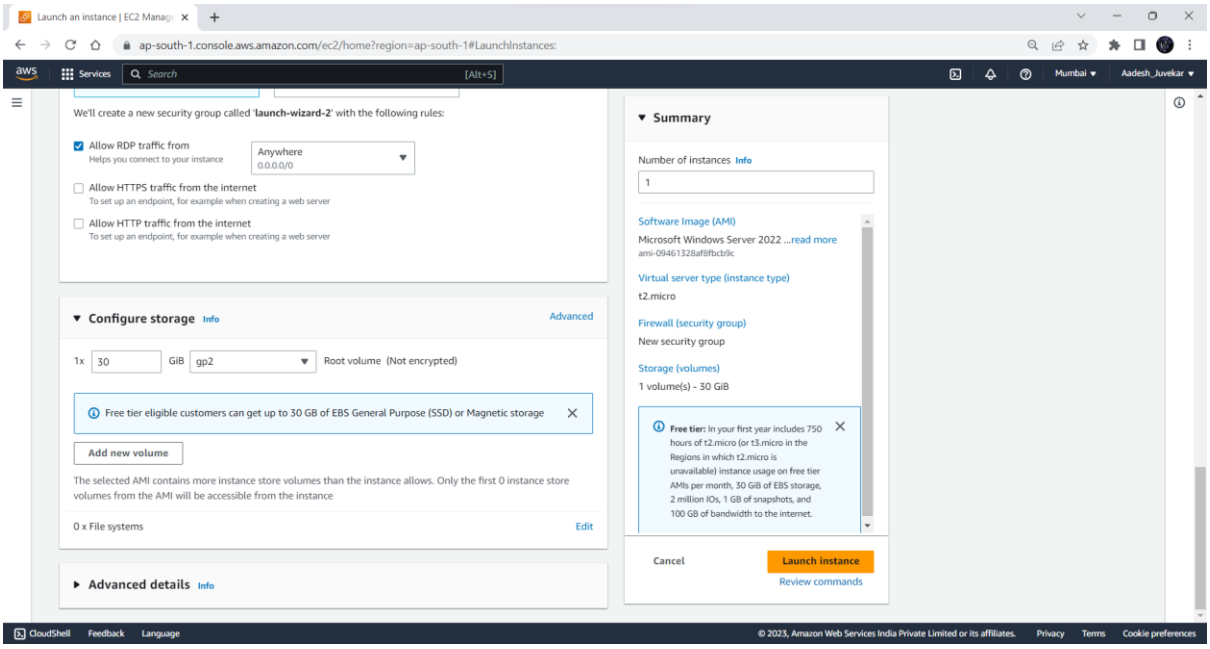
Step 5: Select Instance type and select existing Key-Pair

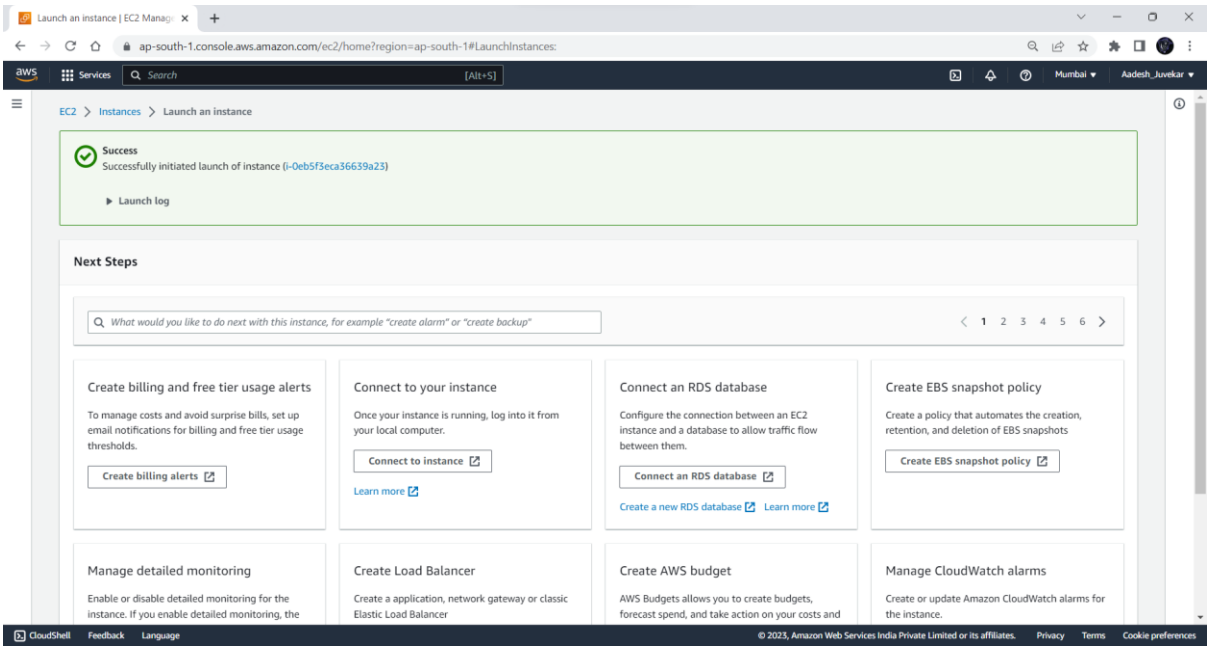


Step 6: Configure Network Settings and Security groups policies

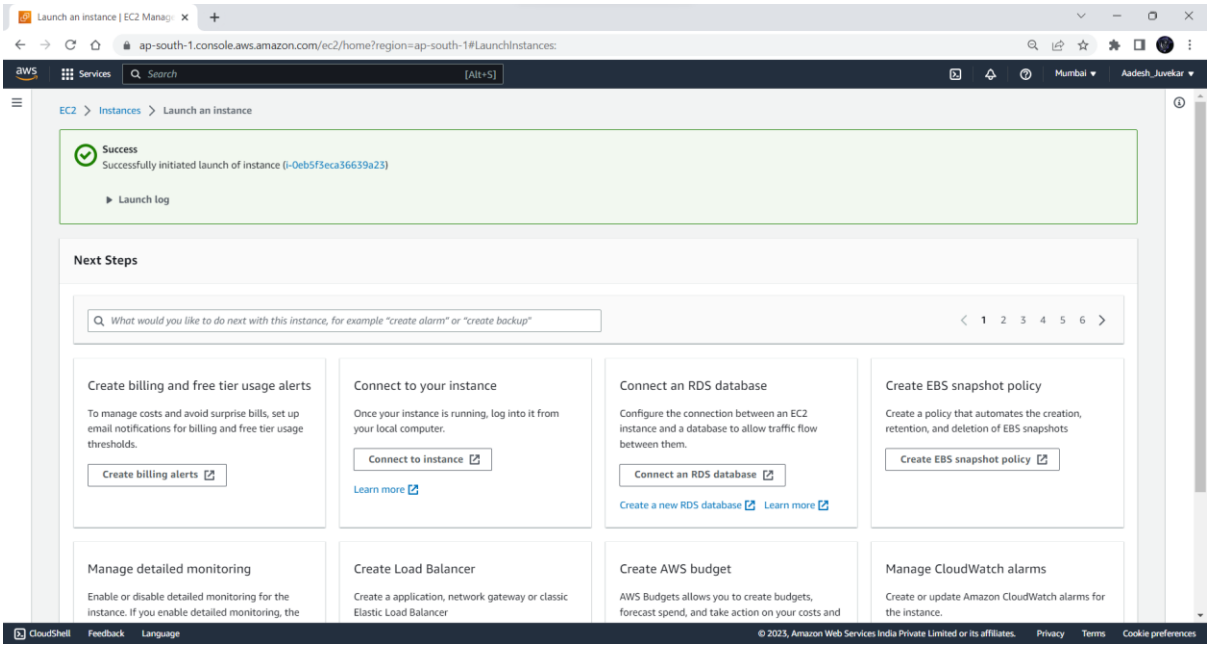


Step 7: Configure storage and then click on Launch Instance to create EC2 instance.

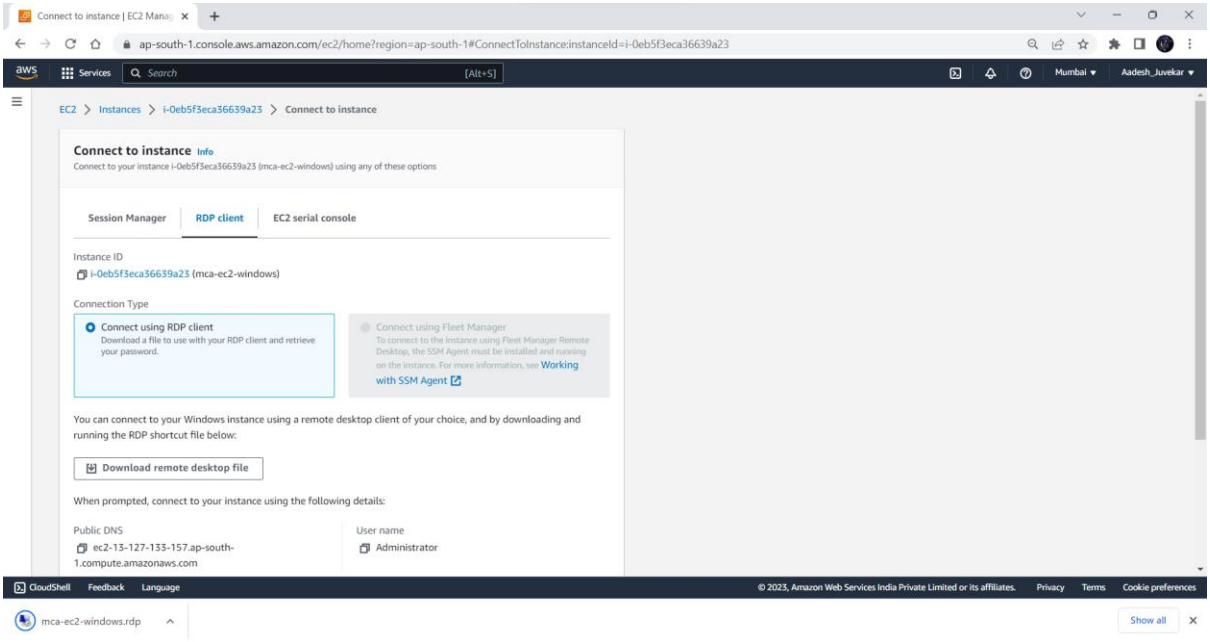




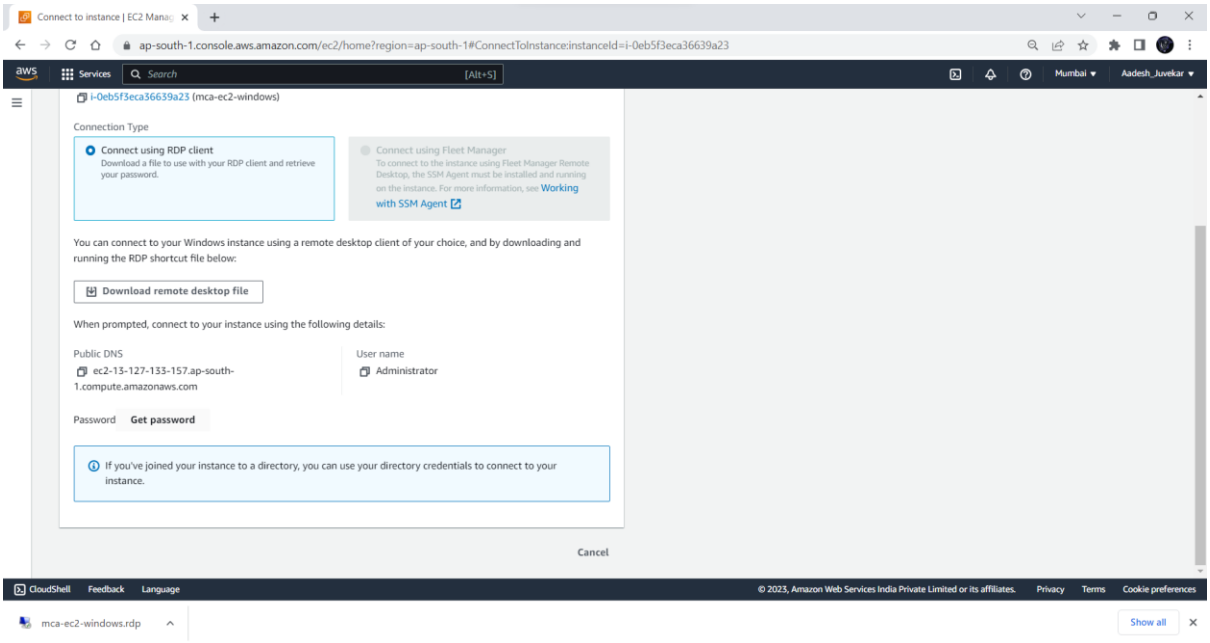
Step 8: Navigate to EC2 instances and select the newly created instance and click on Connect.

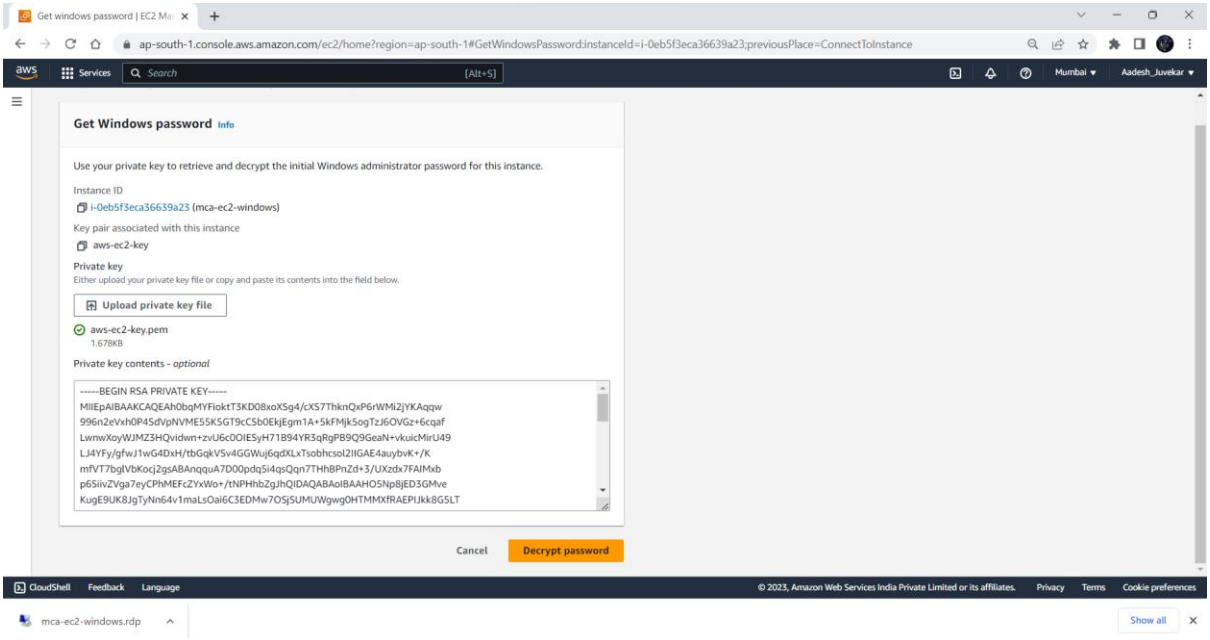


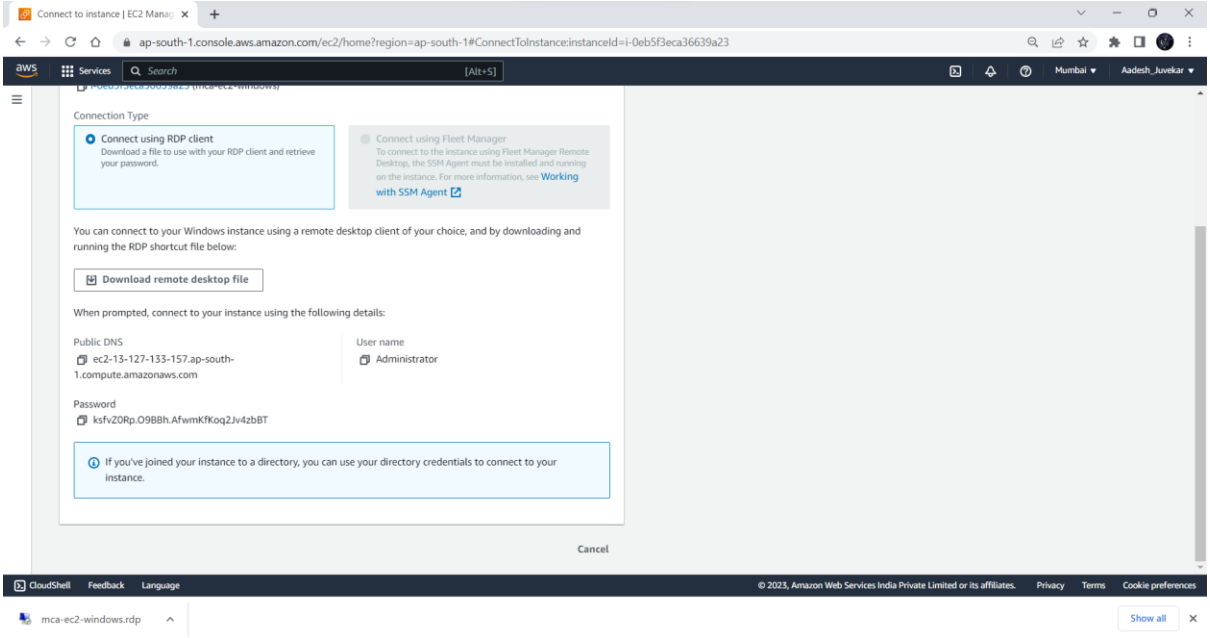
Step 9: Select RDP Client and download Remote Desktop File



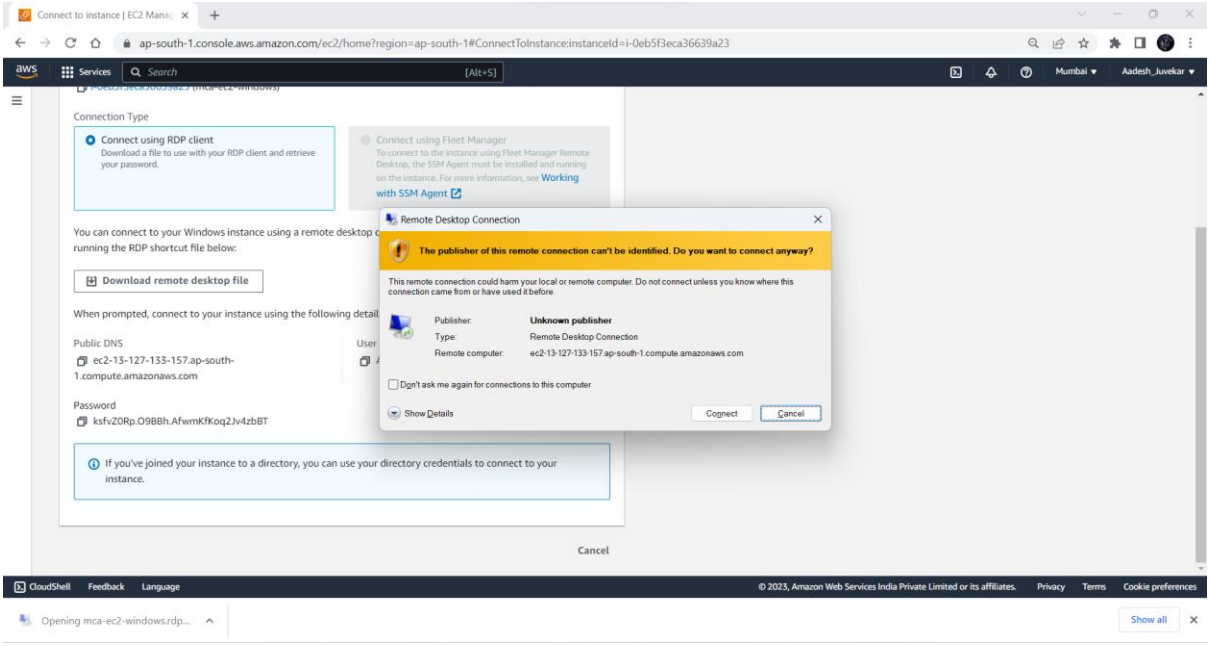
Step 10: Click on Get Password and upload the private key file we used while creating instance and click on Decrypt Password.



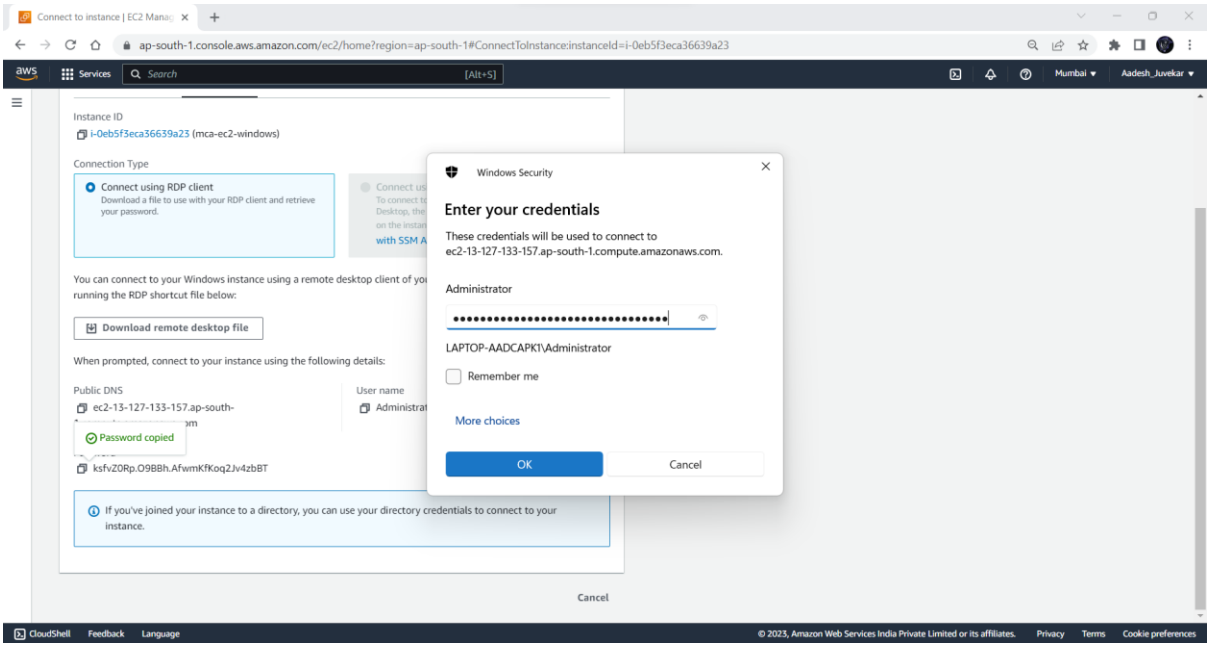


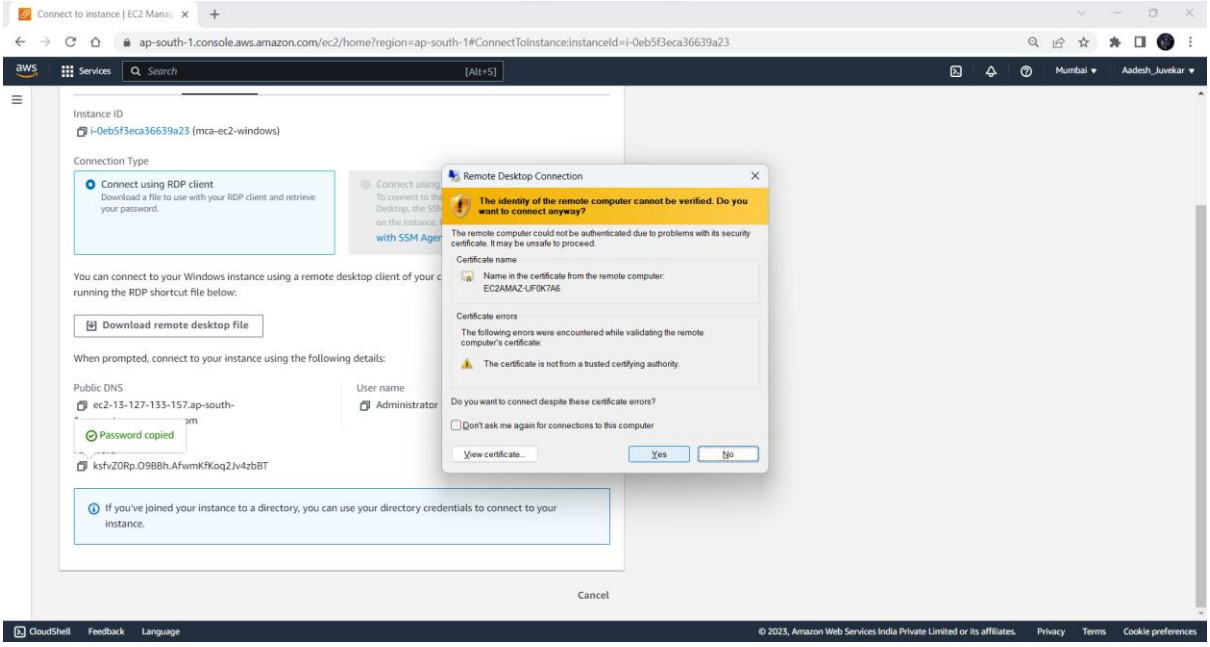


Step 11: Open the Remote Desktop File and click on connect

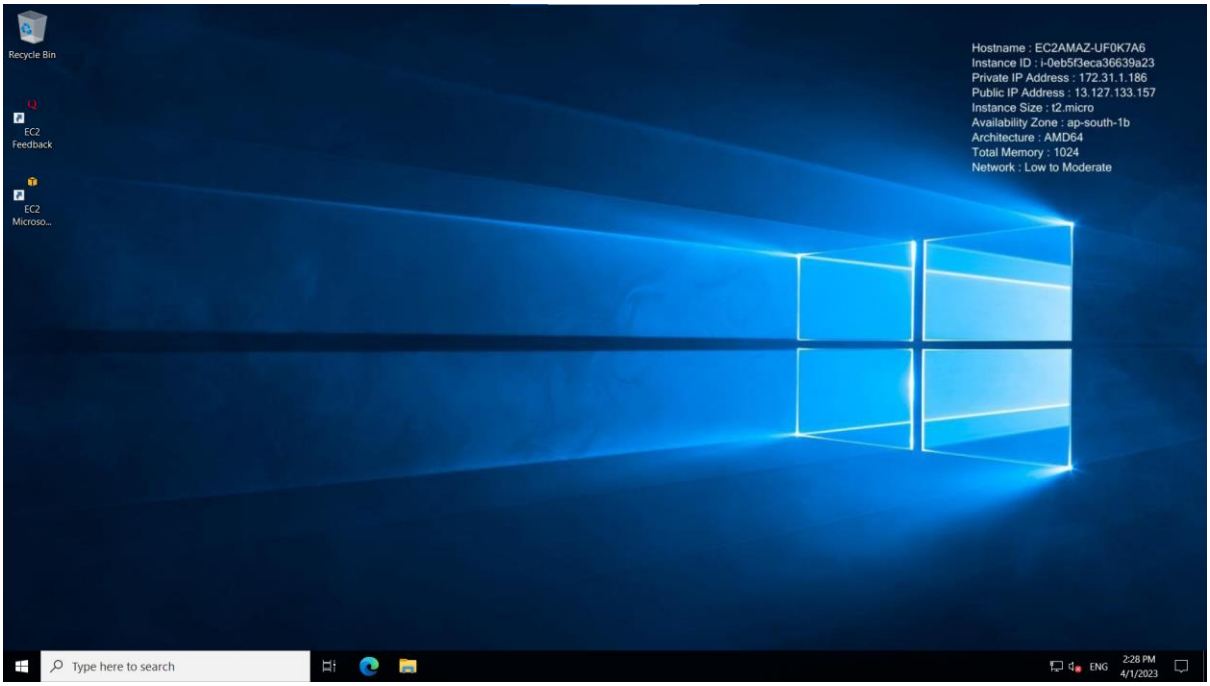


Step 12: Enter the given password and click on OK and then click on Yes





Step 13: We have successfully connected to EC2 instance



Step 14: Create an index.html file and open it in browser.