Roll No. 45 Exam Seat No._____

VIVEKANANDEDUCATION SOCIETY'S INSTITUTE OF TECHNOLOGY

HashuAdvani Memorial Complex, Collector's Colony, R. C. Marg,
Chembur, Mumbai – 400074. Contact No. 02261532532

Since 1962

CERTIFICATE

Certified that Mr./Miss Pushkar Prasad Sane

of SY MCA / A has

satisfactorily completed a course of the necessary experiments in

Distributed System and Cloud Computing Lab under my supervision

in the Institute of Technology in the academic year 2024 - 2025.

Principal Head of Department

Lab In-charge Subject Teacher

V.E.S. Institute of Technology, Collector Colony,
Chembur, Mumbai, Maharashtra 400047
Department of M.C.A

INDEX

Sr. No.	Contents	Date of Preparation	Date of Submission	Marks	Faculty Sign
1	To develop a multi-client chat server application where multiple clients chat with each other concurrently, where the messages sent by different clients are first communicated to the server and then the server, on behalf of the source client, communicates the messages to the appropriate destination client.	05-08-2024	12-08-2024		
2	a. Implement a Server calculator containing ADD(),MUL(),SUB() etc. b. Implement a Date Time Server containing date() and time().	25-08-2024	26-08-2024		
3	a. To find date and time using Remote Method Invocation (Use stubs and skeletons). b. Implementation of equation solver using Remote Method Invocation (RMI).	02-09-2024	05-09-2024		
4	Implementation of Remote Method	09-09-2024	12-09-2024		

	Communication using JDBC and RMI.				
5	Implementation of mutual exclusion using the token ring algorithm.	07-10-2024	14-10-2024		
6	To develop an application using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array. (Binary Search)	14-10-2024	17-10-2024		
7	Description and Implementation of Identity Management in Amazon Web Services.	21-10-2024	21-10-2024		
8	Continuous Assessment 1	09-10-2024	11-10-2024		
9	Continuous Assessment 2	23-10-2024	24-10-2024		

Final Grade	**Instructor Signature**

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 1**	
Title of Lab Assignment: Remote Process Communication		
DOP: 05-08-2024	**DOS: 12-08-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 1

Aim:

To develop a multi-client chat server application where multiple clients chat with each other concurrently, where the messages sent by different clients are first communicated to the server and then the server, on behalf of the source client, communicates the messages to the appropriate destination client.

Description:

A multi-client chat server application where multiple clients chat with each other concurrently. The messages sent by different clients are first communicated to the server and then the server, on behalf of the source client, communicates the messages to the appropriate destination client.

Code:

Server.java

```
package message;
import java.io.*;
import java.net.*;
import java.util.*;

public class server {
   private static List<ClientHandler> clients = new ArrayList<>();
   private static String privateRecipient = null;

   public static void main(String[] args) {
      try {
         ServerSocket serverSocket = new ServerSocket(9999);
         System.out.println("Server is running and listening on port 9999...");

         while (true) {
            Socket clientSocket = serverSocket.accept();
            System.out.println("New client connected");

            ClientHandler clientHandler = new ClientHandler(clientSocket);
            clients.add(clientHandler);
```

```
            clientHandler.start();
        }
    } catch (IOException e) {
        e.printStackTrace();
    }
}
static class ClientHandler extends Thread {
    private Socket clientSocket;
    private PrintWriter writer;
    private String name;

    public ClientHandler(Socket socket) {
        this.clientSocket = socket;
    }

    public void run() {
        try {
                                    BufferedReader reader = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
            writer = new PrintWriter(clientSocket.getOutputStream(), true);

            System.out.print("Enter your name: ");
            name = reader.readLine();
            writer.println("Welcome, " + name + "!");

            String message;
            while ((message = reader.readLine()) != null) {
                if (message.startsWith("/private")) {
                    String[] parts = message.split(" ", 2);
                    String recipient = parts[1];
                    setPrivateRecipient(recipient);
                    writer.println("Chatting privately with " + recipient);
                } else if (message.equals("/offprivate")) {
                    setPrivateRecipient(null);
                    writer.println("Chatting with everyone");
                } else {
```

```
                if (isInPrivateMode()) {
                    sendPrivateMessage(name, privateRecipient, message);
                } else {
                    broadcastMessage(name + ": " + message);
                }
            }
        }
        clients.remove(this);
        writer.close();
        reader.close();
        clientSocket.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}

private void broadcastMessage(String message) {
    for (ClientHandler client : clients) {
        client.writer.println(message);
    }
}

private void sendPrivateMessage(String sender, String recipient, String message) {
    for (ClientHandler client : clients) {
        if (client.name.equals(recipient)) {
            client.writer.println("[Private from " + sender + "]: " + message);
        }
    }
}

private synchronized boolean isInPrivateMode() {
    return privateRecipient != null;
}

private synchronized void setPrivateRecipient(String recipient) {
    privateRecipient = recipient;
```

```
        }
    }
}
```

Client.java

```
package message;
import java.io.*;
import java.net.*;
import java.util.Scanner;

public class client {
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        System.out.print("Enter your name: ");
        String name = scanner.nextLine();

        try {
            Socket socket = new Socket("localhost", 9999);
                                    BufferedReader reader = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
            PrintWriter writer = new PrintWriter(socket.getOutputStream(), true);

            Thread readThread = new Thread(() -> {
                try {
                    String message;
                    while ((message = reader.readLine()) != null) {
                        System.out.println(message);
                    }
                } catch (IOException e) {
                    e.printStackTrace();
                }
            });
            readThread.start();

            Thread writeThread = new Thread(() -> {
                try {
```

```
            Scanner inputScanner = new Scanner(System.in);
            String input;
            while (true) {
                input = inputScanner.nextLine();
                writer.println(input);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    });
    writeThread.start();
    writer.println(name);
    readThread.join();
    writeThread.join();

    reader.close();
    writer.close();
    socket.close();
  } catch (IOException | InterruptedException e) {
    e.printStackTrace();
  }
 }
}
```

Output:

```
Enter your name: Anish
Welcome, Anish!
What time is the train today?
Anish: What time is the train today?
Pushkar: The train has a fixed time everyday, you are late.
```

```
Enter your name: Pushkar
Welcome, Pushkar!
The train has a fixed time everyday, you are late.
Pushkar: The train has a fixed time everyday, you are late.
```

Conclusion:

Successfully demonstrated Remote Process Communication.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 2**	
Title of Lab Assignment: Remote Procedure call		
DOP: 25s-08-2024	**DOS: 26-08-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 2

Aim:

1. Implement a Server calculator containing Add(), Mul(), Sub() etc.
2. Implement a Date Time Server containing date() and time()

Description:

Implementation of Remote Procedure Call Concept using datagram: This application will demonstrate the remote procedure communication.

Theory:

Remote Procedure Call (RPC) is a method used in distributed computing to enable a program to request the execution of a procedure or subroutine on a remote server as if it were a local call. This abstraction simplifies the complexity of network communications by allowing the client to invoke a remote procedure transparently. RPC involves defining procedures on the server side that the client can call using a standardized interface. When the client invokes the procedure, the request is transmitted over the network, and the server executes the procedure and sends back the result. This model allows developers to write distributed systems as if all components were running on the same machine, streamlining the development of networked applications.

Datagram-based communication, particularly using the User Datagram Protocol (UDP), is a connectionless and lightweight method of sending data across a network. In this approach, data is transmitted in discrete packets called datagrams, each of which is routed independently. Unlike connection-oriented protocols like TCP, UDP does not establish a persistent connection between the sender and receiver, nor does it guarantee that packets will arrive in the correct order or without duplication. This can lead to occasional packet loss or out-of-order delivery, but the low overhead associated with UDP can significantly reduce latency and improve performance for applications that can tolerate such issues. This makes UDP ideal for real-time applications, such as streaming or online gaming, where speed is critical and occasional loss of data is acceptable.

Implementing RPC using datagrams involves sending requests and receiving responses through UDP datagrams. In this setup, the client encodes a procedure call and its parameters into a datagram and sends it to the server. The server receives the datagram, decodes the request, executes the procedure, and sends the result back to the client in a response datagram. This method benefits from the low latency and reduced overhead of datagram communication, making it suitable for applications where performance is more

critical than guaranteed delivery. However, the inherent unreliability of datagram-based communication requires applications to handle potential packet loss, duplication, or order issues. Thus, while this approach can offer significant performance benefits, it is essential to ensure that the application logic can manage these reliability challenges effectively.

Code:

Implement a Server calculator containing Add(), Mul(), Sub() etc.

CalculatorServer.java

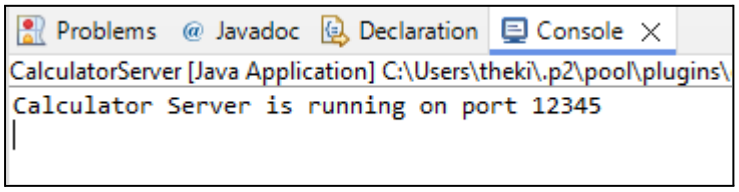
```
package prac2;
import java.io.*;
import java.net.*;

public class CalculatorServer {
	private static final int PORT = 12345;
	public static void main(String[] args) {
		try (ServerSocket serverSocket = new ServerSocket(PORT)) {
			System.out.println("Calculator Server is running on port " + PORT);
			while (true) {
				try (Socket clientSocket = serverSocket.accept();
					BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
					PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true)) {

					String request = in.readLine();
					System.out.println("Received request: " + request);

					String response = processRequest(request);
					out.println(response);
				} catch (IOException e) {
					System.err.println("Error handling client: " +
e.getMessage());
				}
			}
		} catch (IOException e) {
			System.err.println("Server exception: " + e.getMessage());
```

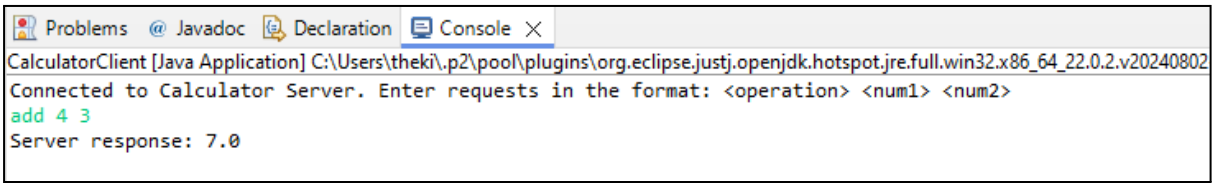
```
                }
        }

        private static String processRequest(String request) {
                String[] parts = request.split(" ");
                if (parts.length != 3) {
                        return "Invalid request. Format should be: <operation> <num1>
<num2>";
                }

                String operation = parts[0];
                double num1;
                double num2;
                try {
                        num1 = Double.parseDouble(parts[1]);
                        num2 = Double.parseDouble(parts[2]);
                } catch (NumberFormatException e) {
                        return "Invalid numbers. Please provide valid numbers.";
                }
                switch (operation.toLowerCase()) {
                case "add":
                        return String.valueOf(num1 + num2);
                case "sub":
                        return String.valueOf(num1 - num2);
                case "mul":
                        return String.valueOf(num1 * num2);
                case "div":
                        if (num2 == 0) {
                                return "Cannot divide by zero.";
                        }
                        return String.valueOf(num1 / num2);
                default:
                        return "Unknown operation. Use add, sub, mul, or div.";
                }
        }
}
```

CalculatorClient.java

```
package prac2;
import java.io.*;
import java.net.*;

public class CalculatorClient {
	private static final String SERVER_ADDRESS = "localhost";
	private static final int PORT = 12345;
	public static void main(String[] args) {
		try (Socket socket = new Socket(SERVER_ADDRESS, PORT);
				BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
				PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);
				BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in))) {

			System.out
					.println("Connected to Calculator Server. Enter requests
in the format: <operation> <num1> <num2>");
			String request;
			while ((request = userInput.readLine()) != null) {
				out.println(request);
				String response = in.readLine();
				System.out.println("Server response: " + response);
			}
		} catch (IOException e) {
			System.err.println("Client exception: " + e.getMessage());
		}
	}
}
```

Output:





Implement a Date Time Server containing date() and time().

DateTimeServer.java

```
package DateTimeServer;
import java.io.*;
import java.net.*;
import java.text.SimpleDateFormat;
import java.util.Date;

public class DateTimeServer {
	private static final int PORT = 222;
	public static void main(String[] args) {
		try (ServerSocket serverSocket = new ServerSocket(PORT)) {
			System.out.println("Date Time Server is running on port " + PORT);
			while (true) {
				try (Socket clientSocket = serverSocket.accept();
					BufferedReader in = new BufferedReader(new
InputStreamReader(clientSocket.getInputStream()));
					PrintWriter out = new
PrintWriter(clientSocket.getOutputStream(), true)) {

					String request = in.readLine();
					System.out.println("Received request: " + request);
					String response = handleRequest(request);
					out.println(response);
				} catch (IOException e) {
```

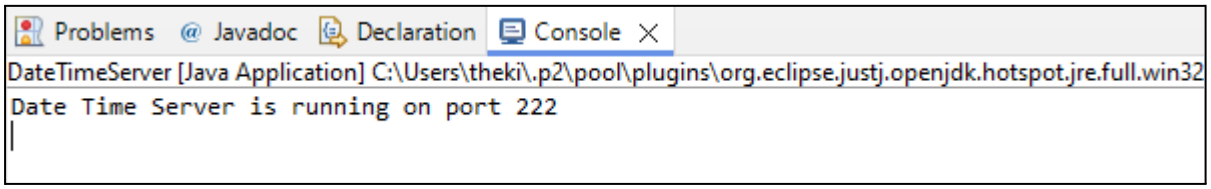
```
                                        System.err.println("Error handling client: " +
e.getMessage());
                                }
                        }
                } catch (IOException e) {
                        System.err.println("Server exception: " + e.getMessage());
                }
        }

        private static String handleRequest(String request) {
                String response;
                if ("date".equalsIgnoreCase(request)) {
                        response = getCurrentDate();
                } else if ("time".equalsIgnoreCase(request)) {
                        response = getCurrentTime();
                } else {
                        response = "Invalid request. Use 'date' or 'time'.";
                }
                return response;
        }

        private static String getCurrentDate() {
                SimpleDateFormat dateFormat = new SimpleDateFormat("yyyy-MM-dd");
                return "Current date: " + dateFormat.format(new Date());
        }

        private static String getCurrentTime() {
                SimpleDateFormat timeFormat = new SimpleDateFormat("HH:mm:ss");
                return "Current time: " + timeFormat.format(new Date());
        }
}
```

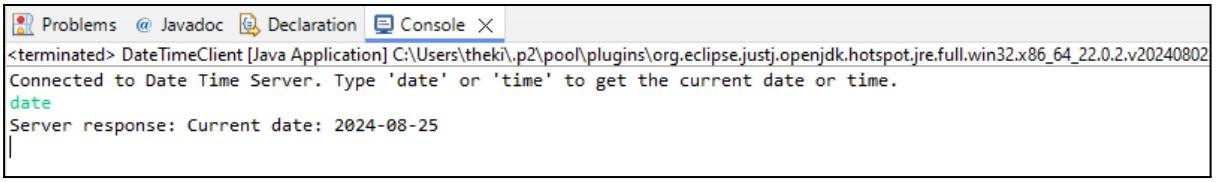
DateTimeClient.java

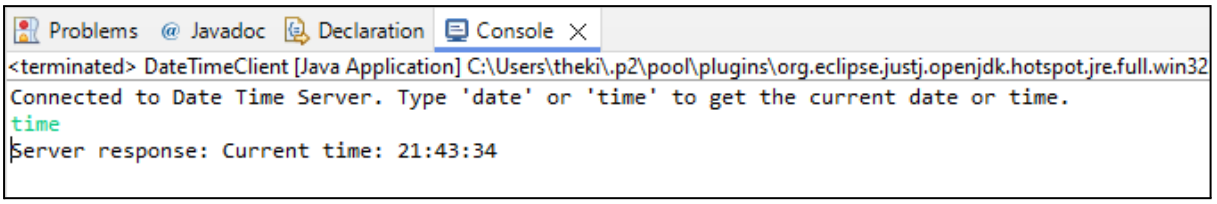
```
package DateTimeServer;
import java.io.*;
import java.net.*;
```

```
public class DateTimeClient {
	private static final String SERVER_ADDRESS = "localhost";
	private static final int PORT = 222;
	public static void main(String[] args) {
		try (Socket socket = new Socket(SERVER_ADDRESS, PORT);
				BufferedReader in = new BufferedReader(new
InputStreamReader(socket.getInputStream()));
				PrintWriter out = new PrintWriter(socket.getOutputStream(),
true);
				BufferedReader userInput = new BufferedReader(new
InputStreamReader(System.in))) {
			System.out.println("Connected to Date Time Server. Type 'date' or
'time' to get the current date or time.");
			String request = userInput.readLine();
			out.println(request);
			String response = in.readLine();
			System.out.println("Server response: " + response);
		} catch (IOException e) {
			System.err.println("Client exception: " + e.getMessage());
		}
	}
}
```

Output:







Conclusion:

The implementation of the RPC-based server and Date Time Server illustrates the efficient use of socket programming to manage remote procedure calls. It effectively handles client requests and ensures reliable communication, showcasing the benefits of distributed computing in networked applications.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 3**	
Title of Lab Assignment: Remote Method Invocation		
DOP: 02-09-2024	**DOS: 05-09-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 3

Aim:

1. To find date and time using Remote Method Invocation (Use stubs and skeletons).
2. Implementation of equation solver using Remote Method Invocation (RMI).

Description:

1. Create a client and server application where the client invokes methods via an interface. These methods are implemented on the server side. Create the necessary STUBS and SKELETONS. Retrieve time and date function from server to client. This program should display server date and time. (Use the concept of JDBC and RMI for accessing multiple data access objects).
2. Equation solver. The client should provide an equation to the server through an interface. The server will solve the expression given by the client.
   (a-b)2 = a2 –2ab + b2;
   If a = 5 and b = 2 then return value = 52 – 2.5.2 + 22 = 9

Theory:

1. **RMI (Remote Method Invocation)**
   Remote Method Invocation (RMI) is a Java API that allows an object running in one Java Virtual Machine (JVM) to invoke methods on an object running in another JVM. This is often referred to as remote communication. RMI enables the construction of distributed applications, where different parts of the application can be located on different machines.
   RMI abstracts much of the complexity of network programming, allowing developers to focus on the application logic rather than the details of network communication.

2. **Stub**
   A stub is a client-side proxy that represents the remote object. When a client wants to invoke a method on a remote object, it actually calls the corresponding method on the stub. The stub then handles the communication over the network, forwarding the call to the actual remote object on the server.

Key Responsibilities of the Stub:

- Marshalling: The stub converts (serializes) the method arguments into a format that can be transmitted over the network.
- Sending the Request: The stub sends the method invocation request to the server via the network.
- Receiving the Response: After the server processes the request and sends back a response, the stub receives it.
- Unmarshalling: The stub converts (deserializes) the response back into a form that the client can use.

3. Skeleton

The skeleton is the server-side counterpart to the stub. In older versions of Java (pre-Java 1.2), the skeleton was responsible for receiving method invocation requests from the stub, unmarshalling the parameters, invoking the appropriate method on the remote object, and sending the result back to the stub.

Key Responsibilities of the Skeleton:

- Receiving the Request: The skeleton listens for incoming method invocation requests from the client.
- Unmarshalling the Request: The skeleton deserializes the method arguments sent by the stub.
- Invoking the Method: The skeleton invokes the corresponding method on the actual remote object on the server.
- Marshalling the Response: The skeleton serializes the result and sends it back to the stub.

Code:

1. **Client and server application where the client invokes methods via an interface.**

   DateTimeService.java

```
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface DateTimeService extends Remote {
        String getDateTime() throws RemoteException;
}
```

DateTimeServiceImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.text.SimpleDateFormat;
import java.util.Date;
public class DateTimeServiceImpl extends UnicastRemoteObject implements DateTimeService {
	protected DateTimeServiceImpl() throws RemoteException {
		super();
	}

	@Override
	public String getDateTime() throws RemoteException {
		SimpleDateFormat sdf = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
		return sdf.format(new Date());
	}
}
```

DateTimeServer.java

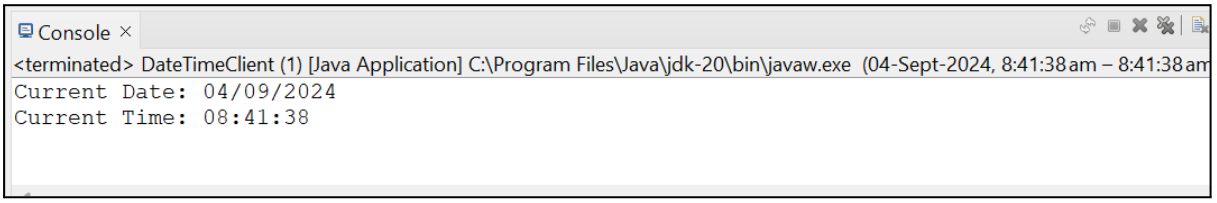
```
import java.rmi.Naming;
import java.rmi.RemoteException;
import java.rmi.registry.LocateRegistry;

public class DateTimeServer {
	public static void main(String[] args) {
		try {
			// Create and export a remote object
			DateTimeServiceImpl obj = new DateTimeServiceImpl();

			// Create and start the RMI registry on port 1099
			LocateRegistry.createRegistry(1099);

			// Bind the remote object in the registry
			Naming.rebind("DateTimeService", obj);
```

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

DateTimeClient.java

```
import java.rmi.Naming;
import java.rmi.RemoteException;

public class DateTimeClient {
    public static void main(String[] args) {
        try {
            // Lookup the remote object
            DateTimeService service = (DateTimeService) Naming.lookup("rmi://localhost/DateTimeService");
            // Invoke the remote method
            String dateTime = service.getDateTime();
            System.out.println("Current Date and Time from Server: " + dateTime);
        } catch (Exception e) {
            System.err.println("Client exception: " + e.toString());
            e.printStackTrace();
        }
    }
}
```

Output:



Code: Implementation of equation solver using Remote Method Invocation (RMI).

EquationSolverInterface.java:

```
package server;
import java.rmi.Remote;
import java.rmi.RemoteException;
public interface EquationSolverInterface extends Remote {
  int solveEquation(String equation, int a, int b, int c) throws RemoteException;
}
```

EquationSolverImpl.java:

```
package server;
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
public class EquationSolverImpl extends UnicastRemoteObject implements EquationSolverInterface {
  protected EquationSolverImpl() throws RemoteException {
    super();
  }
  @Override
  public int solveEquation(String equation, int a, int b, int c) throws RemoteException {
    int result = 0;
    switch (equation) {
      case "(a+b)^2":
        result = (int) (Math.pow(a + b, 2));
        break;
      case "(a-b)^2":
        result = (int) (Math.pow(a, 2) - 2 * a * b + Math.pow(b, 2));
        break;
      case "(a+b+c)^2":
        result = (int) (Math.pow(a + b + c, 2));
```

```
            break;
        case "(a+b)^3":
            result = (int) (Math.pow(a + b, 3));
            break;
        default:
            throw new RemoteException("Equation not supported.");
    }
    return result;
  }
}
```

EquationSolverServer.java:

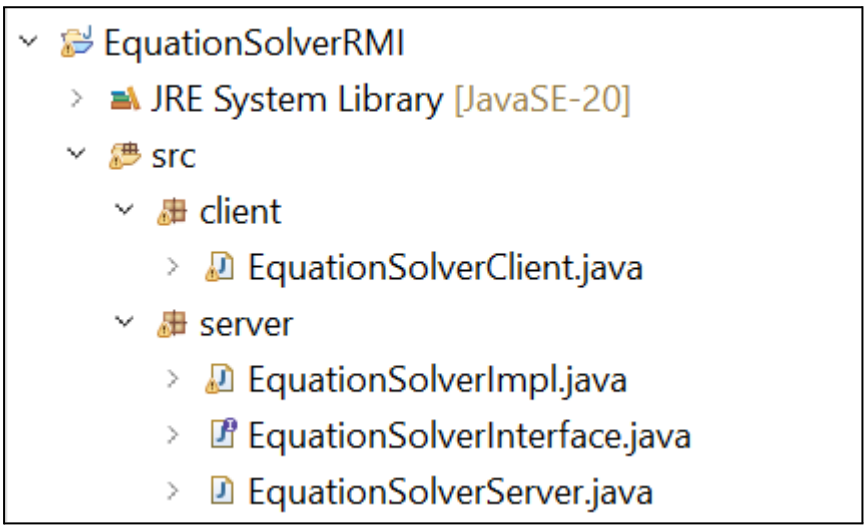
```
package server;
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
public class EquationSolverServer {
  public static void main(String[] args) {
    try {
      LocateRegistry.createRegistry(1099);
      EquationSolverImpl obj = new EquationSolverImpl();
      Naming.rebind("rmi://localhost/EquationSolverServer", obj);
      System.out.println("Server started, waiting for client to enter equations...");
    } catch (Exception e) {
      e.printStackTrace();
    }
  }
}
```

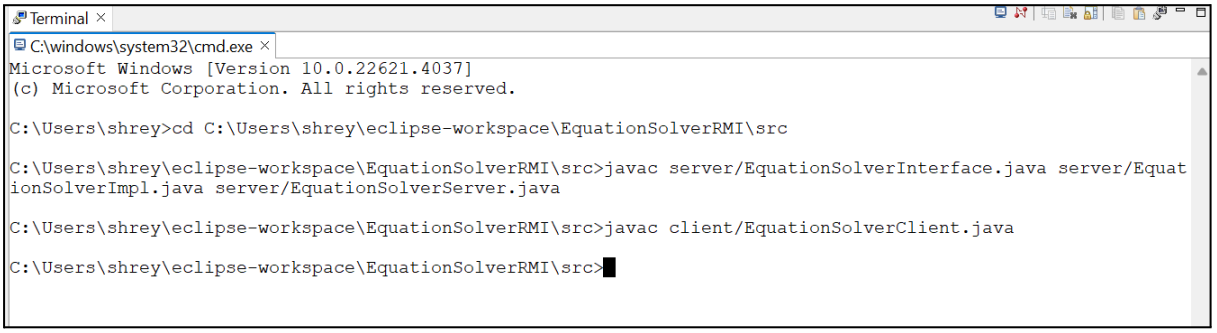
EquationSolverClient.java:

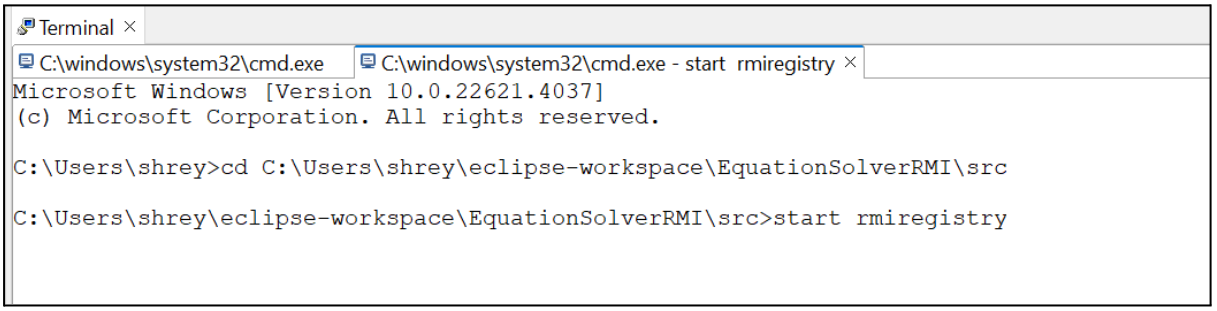
```
package client;
import server.EquationSolverInterface;
import java.rmi.Naming;
import java.util.Scanner;
public class EquationSolverClient {
  public static void main(String[] args) {
    try {
```

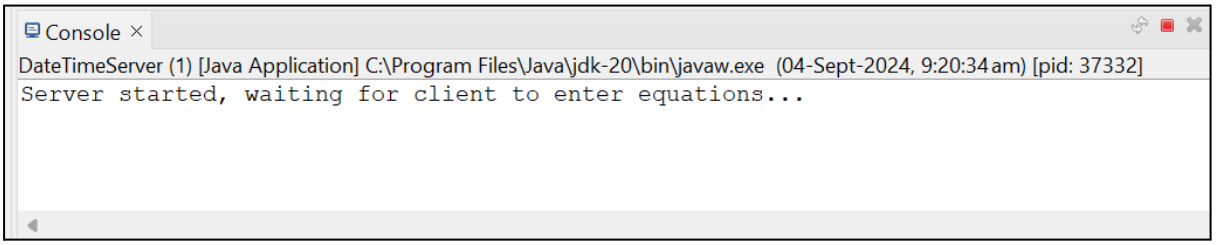
```
                EquationSolverInterface obj = (EquationSolverInterface)
Naming.lookup("rmi://localhost/EquationSolverServer");
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter equation: ");
        String equation = sc.nextLine();
        System.out.print("Enter value for a: ");
        int a = sc.nextInt();
        System.out.print("Enter value for b: ");
        int b = sc.nextInt();
        int c = 0;
        if (equation.contains("c")) {
            System.out.print("Enter value for c: ");
            c = sc.nextInt();
        }
        int result = obj.solveEquation(equation, a, b, c);
        System.out.println("Answer: " + result);
    } catch (Exception e) {
        e.printStackTrace();
    }
  }
}
```

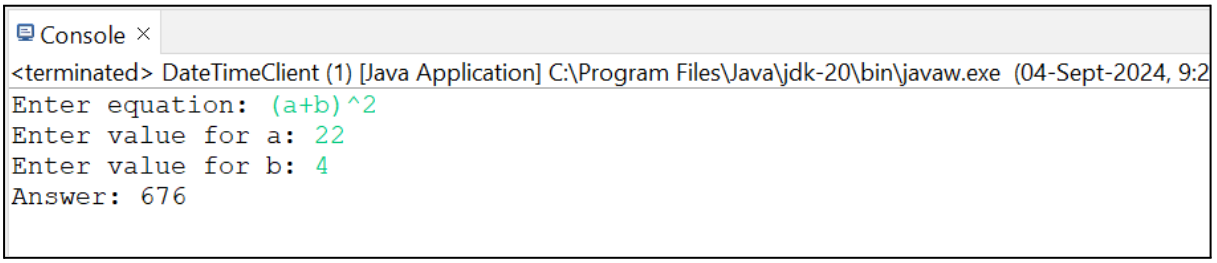
Output:

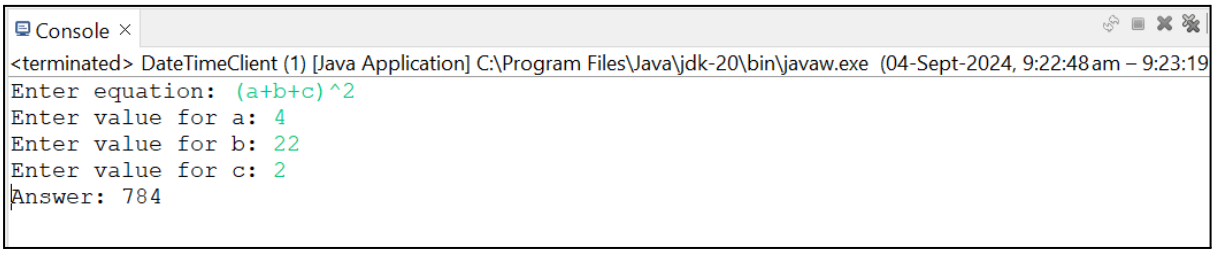












Conclusion: Implemented finding date and time and solving equations using Remote Method Invocation successfully.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 4**	
Title of Lab Assignment: Implementation of Remote Method Communication using JDBC and RMI.		
DOP: 09-09-2024	**DOS: 12-09-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 4

Aim: Implementation of Remote Method Communication using JDBC and RMI.

Description: Make use of JDBC and RMI for accessing multiple data access objects.

Theory:

- **Java Database Connectivity (JDBC)**

  Java Database Connectivity (JDBC): JDBC is an API in Java that enables applications to interact with relational databases. It allows Java programs to execute SQL queries, retrieve results, and manage database connections. JDBC acts as a bridge between the Java application and the database, providing a standard method for connecting to and operating on databases like MySQL, Oracle, and others.

- **Steps to Connect Java JDBC**

  To connect Java to a database using JDBC, the following steps are typically followed:

  1. Load the JDBC driver class using Class.forName.
  2. Establish a connection to the database using DriverManager.getConnection.
  3. Create a Statement or PreparedStatement object to execute SQL queries.
  4. Execute the query and process the results using ResultSet.
  5. Finally, close the ResultSet, Statement, and Connection objects to release resources.

- **RMI (Remote Method Invocation)**

  RMI is a Java API that allows objects residing in different Java Virtual Machines (JVMs) to communicate with each other. It enables the invocation of methods on a remote object as if it were a local object. RMI abstracts the complexities of remote communication, making it easy to build distributed applications where methods can be invoked across a network.

- **Stub**

  In RMI, a stub is a client-side proxy object that represents the remote object. When a client invokes a method on the stub, the stub handles the communication with the remote server, sending the method call over the network, and receiving the response, making the remote method invocation appear seamless to the client.

- **Skeleton**

  A skeleton was a server-side component that acted as a gateway between the RMI runtime and the actual remote object implementation. It received incoming requests from the stub, deserialized the parameters, invoked the corresponding method on the remote object, and sent back the results. In later versions of Java, the skeleton functionality was incorporated into the RMI runtime, making the skeleton class unnecessary.

  Add the following code in pom.xml for both questions:

```
<project xmlns="http://maven.apache.org/POM/4.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://maven.apache.org/POM/4.0.0
https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>DSCC4</groupId>
  <artifactId>DSCC4</artifactId>
  <version>0.0.1-SNAPSHOT</version>

   <dependencies>
      <dependency>
         <groupId>mysql</groupId>
         <artifactId>mysql-connector-java</artifactId>
         <version>8.0.33</version>
      </dependency>
   </dependencies>
</project>
```

1. Using MySQL, create a Library database. Create table Book (Book_id, Book_name, Book_author) and retrieve the Book information from the Library database.

   Code:

   SQL Code:

```
CREATE DATABASE Library;
USE Library;
CREATE TABLE Book (
   Book_id INT AUTO_INCREMENT PRIMARY KEY,
   Book_name VARCHAR(100) NOT NULL,
```

```
    Book_author VARCHAR(100)
);
INSERT INTO Book (Book_name, Book_author)
VALUES
('To Kill a Mockingbird', 'Harper Lee'),
('1984', 'George Orwell'),
('The Great Gatsby', 'F. Scott Fitzgerald');
```

Book.java

```
import java.io.Serializable;
public class Book implements Serializable {
    private int id;
    private String name;
    private String author;
    public Book(int id, String name, String author) {
        this.id = id;
        this.name = name;
        this.author = author;
    }
    // Getters and toString method
    public int getId() {
        return id;
    }
    public String getName() {
        return name;
    }
    public String getAuthor() {
        return author;
    }
    @Override
    public String toString() {
        return "Book{" +
                "id=" + id +
                ", name='" + name + '\'' +
                ", author='" + author + '\'' +
                '}';
```

```
    }
}
```

LibraryClient.java

```
import java.rmi.Naming;
import java.util.List;
public class LibraryClient {
    public static void main(String[] args) {
        try {
            LibraryService libraryService = (LibraryService)
Naming.lookup("rmi://localhost:1099/LibraryService");
            List<Book> books = libraryService.getBooks();
            for (Book book : books) {
                System.out.println(book);
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

LibraryServer.java

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
public class LibraryServer {
    public static void main(String[] args) {
        try {
            LocateRegistry.createRegistry(1099); // Default RMI port
            LibraryService libraryService = new LibraryServiceImpl();
            Naming.rebind("rmi://localhost:1099/LibraryService", libraryService);
            System.out.println("Library RMI Server is running...");
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

LibraryService.java

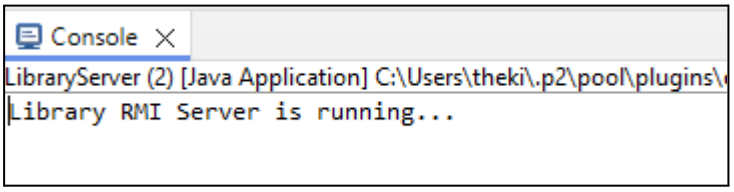
```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
public interface LibraryService extends Remote {
    List<Book> getBooks() throws RemoteException;
}
```

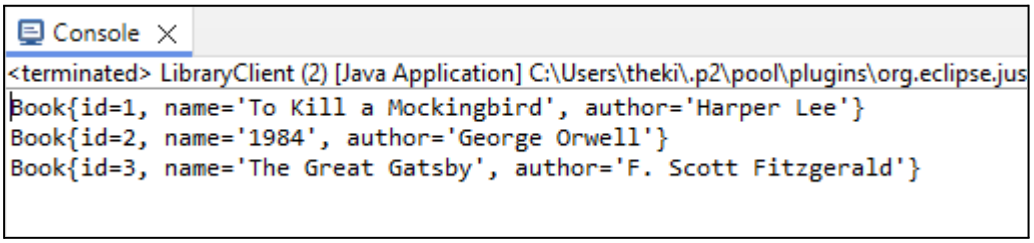
LibraryServiceImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
public class LibraryServiceImpl extends UnicastRemoteObject implements
LibraryService {
    protected LibraryServiceImpl() throws RemoteException {
        super();
    }
    @Override
    public List<Book> getBooks() throws RemoteException {
        List<Book> books = new ArrayList<>();
        try {
            // Connect to the MySQL database
            Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Library", "root", "");
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM Book");
            // Retrieve book information
            while (rs.next()) {
                int id = rs.getInt("Book_id");
                String name = rs.getString("Book_name");
                String author = rs.getString("Book_author");
```

```
                books.add(new Book(id, name, author));
            }
            conn.close();
        } catch (Exception e) {
            e.printStackTrace();
        }
        return books;
    }
}
```

Output:





2. Using MySQL, create an Electric_Bill database. Create table Bill. (bill_id, consumer_name, bill_due_date, bill_amount) and retrieve the Bill information from the Electril_Bill database.

Code:

SQL Code:

```
CREATE DATABASE Electric_Bill;
USE Electric_Bill;
CREATE TABLE Bill (
    bill_id INT AUTO_INCREMENT PRIMARY KEY,
    consumer_name VARCHAR(100) NOT NULL,
    bill_due_date DATE NOT NULL,
    bill_amount DECIMAL(10, 2) NOT NULL
);
INSERT INTO Bill (consumer_name, bill_due_date, bill_amount)
VALUES
('John Doe', '2024-09-30', 150.75),
```

```
('Jane Smith', '2024-10-15', 250.5),
('Alice Johnson', '2024-10-20', 325.4);
```

Bill.java

```
import java.io.Serializable;
public class Bill implements Serializable {
	private int billId;
	private String consumerName;
	private String billDueDate;
	private double billAmount;
	public Bill(int billId, String consumerName, String billDueDate, double
billAmount) {
		this.billId = billId;
		this.consumerName = consumerName;
		this.billDueDate = billDueDate;
		this.billAmount = billAmount;
	}
	// Getters and toString method
	public int getBillId() {
		return billId;
	}
	public String getConsumerName() {
		return consumerName;
	}
	public String getBillDueDate() {
		return billDueDate;
	}
	public double getBillAmount() {
		return billAmount;
	}
	@Override
	public String toString() {
		return "Bill{" + "billId=" + billId + ", consumerName='" +
consumerName + '\'' + ", billDueDate='" + billDueDate
				+ '\'' + ", billAmount=" + billAmount + '}';
	}
```

```
}
```

ElectricBillClient.java

```
import java.rmi.Naming;
import java.util.List;
public class ElectricBillClient {
	public static void main(String[] args) {
		try {
			ElectricBillService billService = (ElectricBillService) Naming
				.lookup("rmi://localhost:1099/ElectricBillService");
			List<Bill> bills = billService.getBills();
			for (Bill bill : bills) {
				System.out.println(bill);
			}
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

ElectricBillServer.java

```
import java.rmi.Naming;
import java.rmi.registry.LocateRegistry;
public class ElectricBillServer {
	public static void main(String[] args) {
		try {
			LocateRegistry.createRegistry(1099); // Default RMI port
			ElectricBillService billService = new ElectricBillServiceImpl();
			Naming.rebind("rmi://localhost:1099/ElectricBillService",
billService);
			System.out.println("Electric Bill RMI Server is running...");
		} catch (Exception e) {
			e.printStackTrace();
		}
	}
}
```

ElectricBillService.java

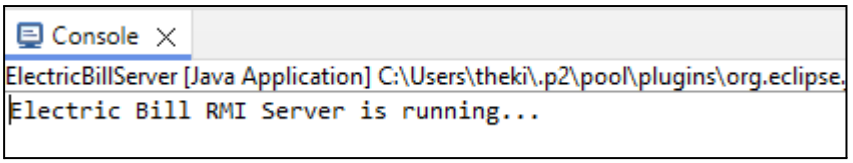
```
import java.rmi.Remote;
import java.rmi.RemoteException;
import java.util.List;
public interface ElectricBillService extends Remote {
	List<Bill> getBills() throws RemoteException;
}
```

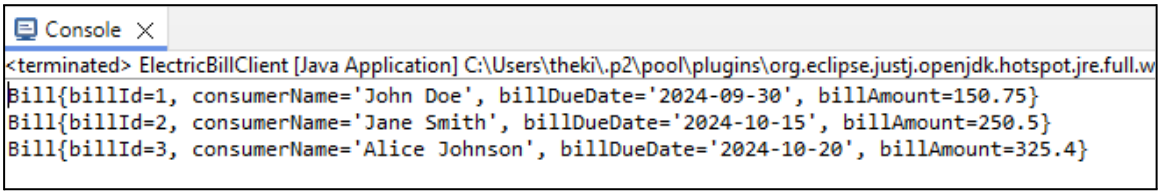
ElectricBillServiceImpl.java

```
import java.rmi.RemoteException;
import java.rmi.server.UnicastRemoteObject;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;
import java.util.ArrayList;
import java.util.List;
public class ElectricBillServiceImpl extends UnicastRemoteObject implements
ElectricBillService {
	protected ElectricBillServiceImpl() throws RemoteException {
		super();
	}
	@Override
	public List<Bill> getBills() throws RemoteException {
		List<Bill> bills = new ArrayList<>();
		try {
			// Connect to the MySQL database using XAMPP's settings
			Connection conn =
DriverManager.getConnection("jdbc:mysql://localhost:3306/Electric_Bill", "root", "");
			Statement stmt = conn.createStatement();
			ResultSet rs = stmt.executeQuery("SELECT * FROM Bill");
			// Retrieve bill information
			while (rs.next()) {
				int id = rs.getInt("bill_id");
				String consumerName =
rs.getString("consumer_name");
```

```
                                String dueDate = rs.getString("bill_due_date");
                                double amount = rs.getDouble("bill_amount");
                                bills.add(new Bill(id, consumerName, dueDate,
amount));
                        }
                        conn.close();
                } catch (Exception e) {
                        e.printStackTrace();
                }
                return bills;
        }
}
```

Output:





Conclusion:

Successfully demonstrated the Implementation of Remote Method Communication using JDBC and RMI.

Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 5**	
Title of Lab Assignment: Implementation of mutual exclusion using the token ring algorithm.		
DOP: 07-10-2024	**DOS: 14-10-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**

Practical No. 5

Aim: Implementation of mutual exclusion using the token ring algorithm.

Theory:

The Token Ring algorithm is a method used in distributed systems to achieve mutual exclusion, allowing multiple nodes (processes) to safely access shared resources without conflict. The algorithm is based on the concept of a token, a special control message that is passed around the nodes in a network topology arranged in a logical ring.

Key Concepts:

1. Token: A unique token circulates among the nodes in the ring. Only the node holding the token can enter its critical section (the part of the code that accesses shared resources).
2. Ring Topology: The nodes are logically arranged in a circular manner, where each node is connected to two neighbors. The token is passed sequentially from one node to its immediate neighbor.
3. Critical Section: This is the part of the code where shared resources are accessed. Only one node can be in its critical section at any time, which prevents race conditions and ensures data integrity.

Algorithm Steps:

1. Initialization: At the start, one node is assigned the token. This node can enter its critical section when it needs to access shared resources.
2. Requesting Access: If a node wants to enter its critical section, it waits until it receives the token. If it does not have the token, it continues to wait.
3. Entering Critical Section: When a node receives the token, it enters its critical section, performs its operations, and then prepares to release the token.
4. Releasing the Token: After completing its work in the critical section, the node releases the token to the next node in the ring. This enables the next node to access the critical section.
5. Passing the Token: The token continues to circulate around the ring, ensuring that each node gets a chance to access the critical section in a fair manner.

Advantages:

1. Fairness: Each node gets an opportunity to access the critical section.
2. Deadlock-Free: The algorithm prevents deadlock as there's always a token available for some node.
3. Simplicity: The logic of token passing is straightforward and easy to implement.

Disadvantages:

1. Token Loss: If a node fails and the token is lost, the system can be deadlocked. Recovery mechanisms are needed.
2. Latency: The time to wait for the token can lead to increased latency, especially in large networks.
3. Single Point of Failure: The system depends on the continuous circulation of the token, making it vulnerable to failures.

Applications:

The Token Ring algorithm is widely used in networking protocols and distributed systems, such as LANs (Local Area Networks), where multiple devices need to share the same communication channel while ensuring data integrity and avoiding conflicts.

Code:

```
import java.util.Random;
class TokenRingNode extends Thread {
    private final int id;
    private final TokenRing ring;
    private boolean hasToken;

    public TokenRingNode(int id, TokenRing ring) {
        this.id = id;
        this.ring = ring;
        this.hasToken = false;
    }

    @Override
    public void run() {
        while (true) {
            if (hasToken) {
```

```
            enterCriticalSection();
            releaseToken();
            break; // Exit after using the token
        }

        try {
            Thread.sleep(new Random().nextInt(1000));
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
    }
}

public void enterCriticalSection() {
    System.out.println("Node " + id + " entering critical section.");
    try {
        Thread.sleep(new Random().nextInt(1000) + 500);
    } catch (InterruptedException e) {
        Thread.currentThread().interrupt();
    }
    System.out.println("Node " + id + " leaving critical section.");
}

public void releaseToken() {
    hasToken = false;
    System.out.println("Node " + id + " releasing token.");
    int nextNodeId = (id + 1) % ring.getSize();
    ring.getNode(nextNodeId).receiveToken();
}

public void receiveToken() {
    if (!hasToken) {
        hasToken = true;
        System.out.println("Node " + id + " received token.");
        run(); // Start requesting access
    }
```

```
    }

    public void requestToken() {
        if (hasToken) {
            return;
        }
        try {
            Thread.sleep(new Random().nextInt(1000)); // Simulate delay
        } catch (InterruptedException e) {
            Thread.currentThread().interrupt();
        }
        run();
    }

    public void setHasToken(boolean hasToken) {
        this.hasToken = hasToken;
    }
}

class TokenRing {
    private final TokenRingNode[] nodes;
    public TokenRing(int numNodes) {
        nodes = new TokenRingNode[numNodes];
        for (int i = 0; i < numNodes; i++) {
            nodes[i] = new TokenRingNode(i, this);
        }
        nodes[0].setHasToken(true);
    }

    public void start() {
        for (TokenRingNode node : nodes) {
            node.start();
        }
    }
```

```
    public int getSize() {
        return nodes.length;
    }

    public TokenRingNode getNode(int index) {
        return nodes[index];
    }
}

public class token {
    public static void main(String[] args) {
        int numNodes = 5;
        TokenRing ring = new TokenRing(numNodes);
        ring.start();
    }
}
```

Output:

```
<terminated> token [Java Application] C:\Users\Exam\.p2\pool\plugins\org.eclipse.justj.o
Node 0 entering critical section.
Node 0 leaving critical section.
Node 0 releasing token.
Node 1 received token.
Node 1 entering critical section.
Node 1 leaving critical section.
Node 1 releasing token.
Node 2 received token.
Node 2 entering critical section.
Node 2 entering critical section.
Node 2 leaving critical section.
Node 2 releasing token.
Node 3 received token.
Node 3 entering critical section.
Node 3 entering critical section.
Node 3 leaving critical section.
Node 3 releasing token.
Node 4 received token.
Node 4 entering critical section.
Node 2 leaving critical section.
Node 2 releasing token.
Node 3 received token.
Node 3 entering critical section.
Node 3 leaving critical section.
Node 3 releasing token.
Node 4 entering critical section.
Node 3 leaving critical section.
Node 3 releasing token.
Node 4 leaving critical section.
```

Conclusion:

The Token Ring algorithm is a fundamental method for achieving mutual exclusion in distributed systems. Its reliance on a unique token to control access to critical sections makes it an efficient and effective solution for managing shared resources in a synchronized manner.

Name of Student: Pushkar Sane		
Roll Number: 45		**Lab Assignment No: 6**
Title of LAB Assignment: To develop application using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array. (Binary Search)		
DOP: 14-10-2024		**DOS: 17-10-2024**
CO Mapped: CO6	**PO Mapped: PO1, PO2, PO3, PO7, PO9, PSO1**	**Signature:**

Practical No. 6

Aim: To develop application using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array. (Binary Search)

Description:

Platform as a Service (PaaS) is a cloud computing service model that provides a platform allowing developers to build, run, and manage applications without worrying about the underlying infrastructure. PaaS supplies hardware and software tools over the internet, typically requiring the user to manage the applications but not the servers or storage. It simplifies the application development process by offering pre-built infrastructure and deployment environments, allowing developers to focus on coding rather than managing the back-end systems.

Key Features of PaaS:

1. **Development Frameworks**: Provides frameworks to help developers create cloud-native applications quickly.
2. **Middleware**: PaaS comes with middleware that integrates with databases and other services.
3. **Infrastructure Management**: It automates much of the infrastructure management, such as load balancing and scaling.
4. **Development Tools**: Offers built-in tools like version control systems, debugging, and testing tools.
5. **Scalability**: PaaS platforms scale applications automatically based on the traffic load.

Example: Google App Engine

Google App Engine (GAE) is a well-known PaaS offering by Google that enables developers to build and host web applications on Google-managed infrastructure. It abstracts much of the infrastructure complexity, allowing developers to focus solely on writing code.

Features of Google App Engine:

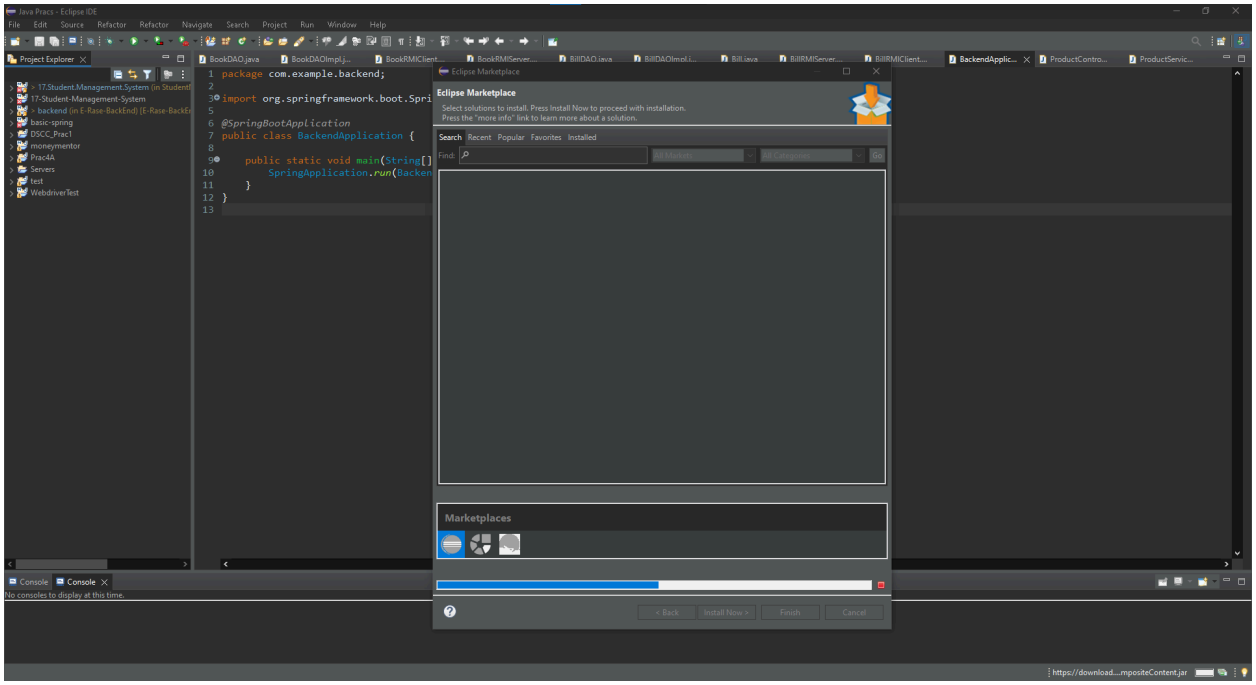
1. **Automatic Scalability**: GAE automatically scales applications up or down based on traffic.
2. **Managed Platform**: It handles server management, such as configuration, load balancing, and scaling, for the user.
3. **Multiple Languages**: Supports various programming languages like Java, Python, Go, Node.js, PHP, and Ruby.
4. **Integrated Google Services**: Offers easy integration with other Google Cloud services such as Datastore, Cloud SQL, and Cloud Storage.
5. **Flexible Deployment**: GAE supports both standard and flexible environments, allowing developers to choose between sandboxed or container-based environments.

Google App Engine is ideal for building scalable web applications and services, eliminating the need to manage infrastructure manually while leveraging Google's extensive cloud resources.

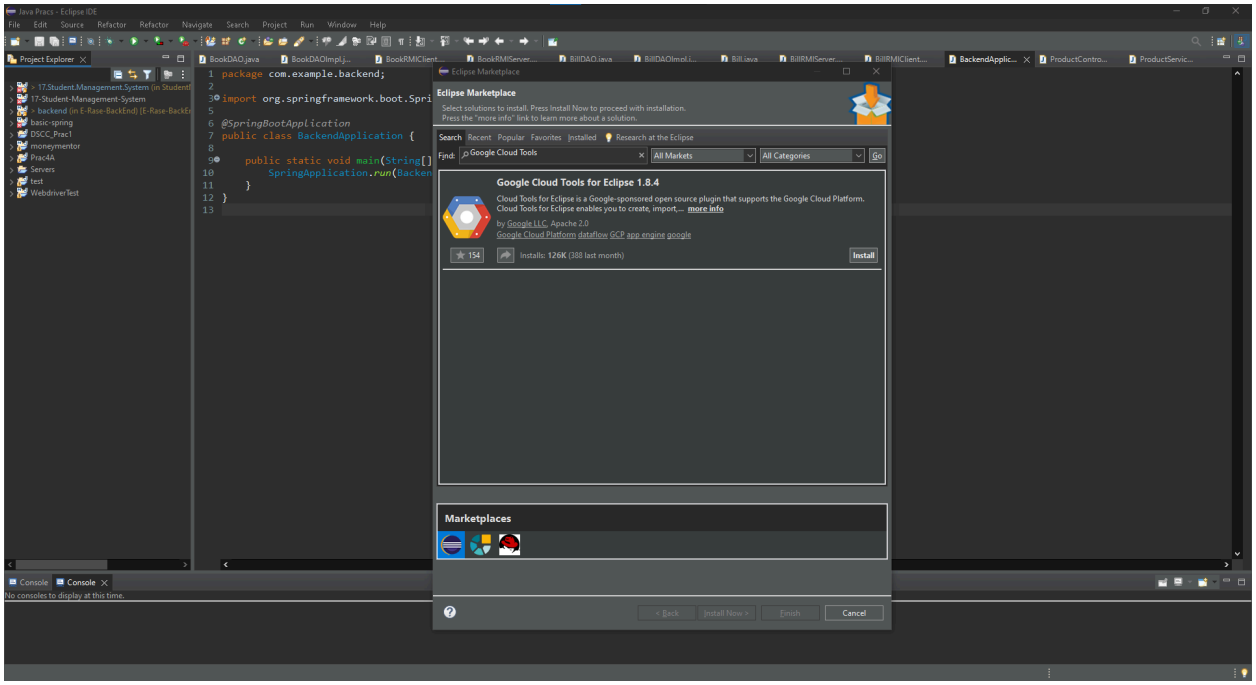
To develop application using Google App Engine by using Eclipse IDE. Find the position of a target value within a sorted integer array. (Binary Search)

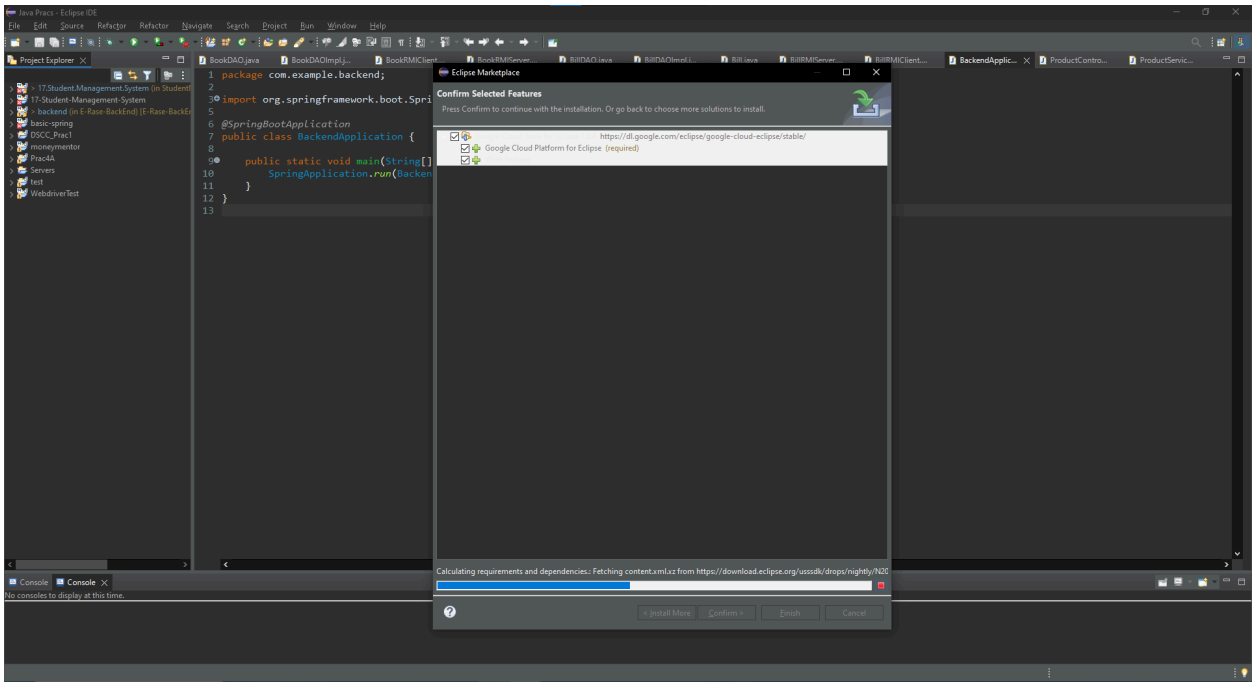
Step 1: Set Up Your Google App Engine Project

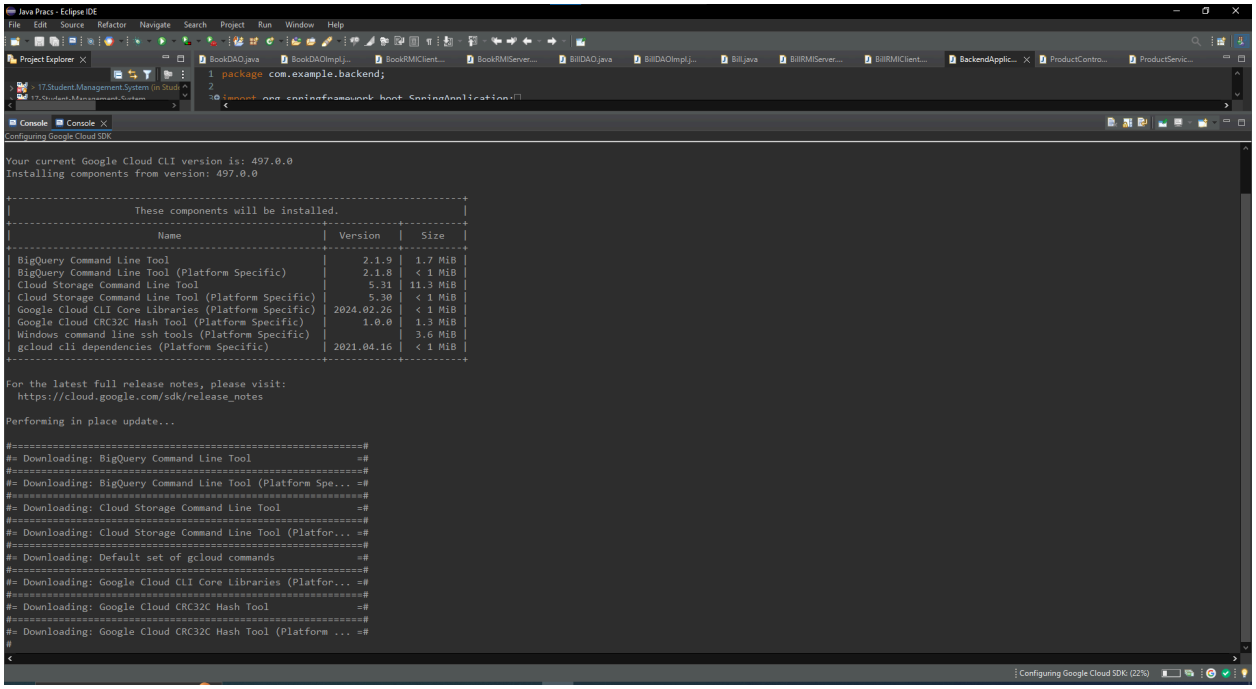
1. **Install Eclipse IDE**: Ensure you have Eclipse IDE installed. You can download it from the Eclipse website.
2. **Install Google Cloud Tools for Eclipse**: This plugin helps to develop and deploy applications to Google App Engine.
    - Go to Help -> Eclipse Marketplace.



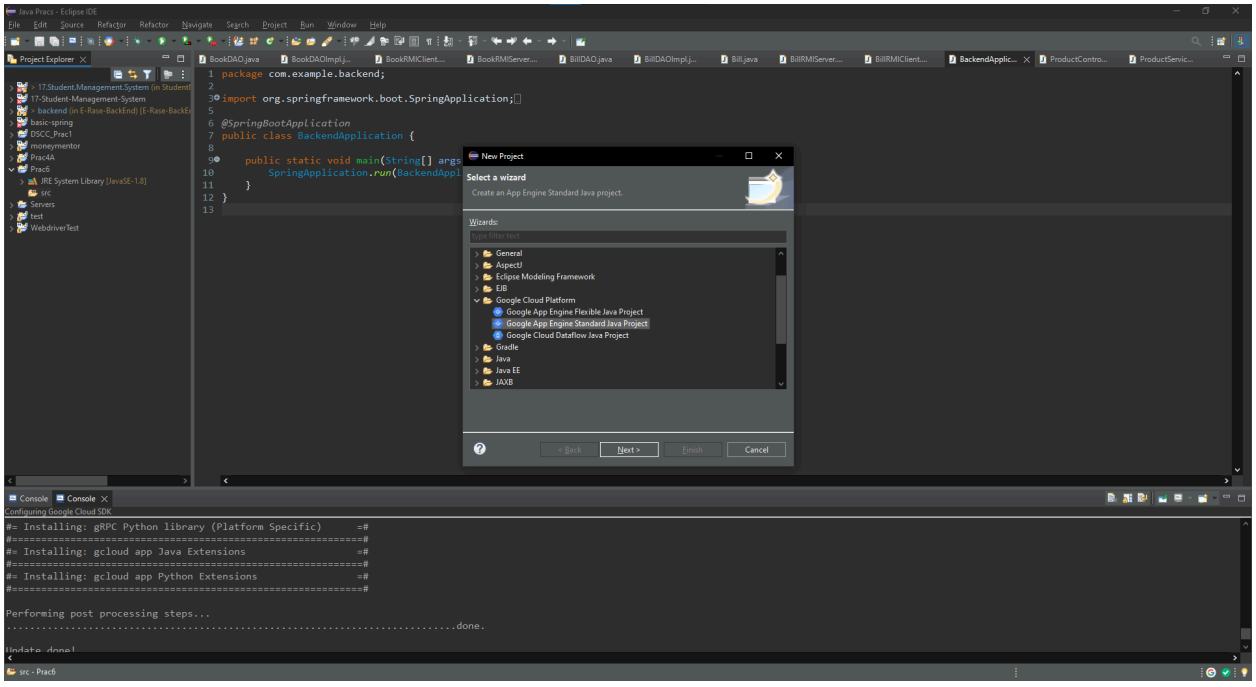
- Search for "Google Cloud Tools" and install it.

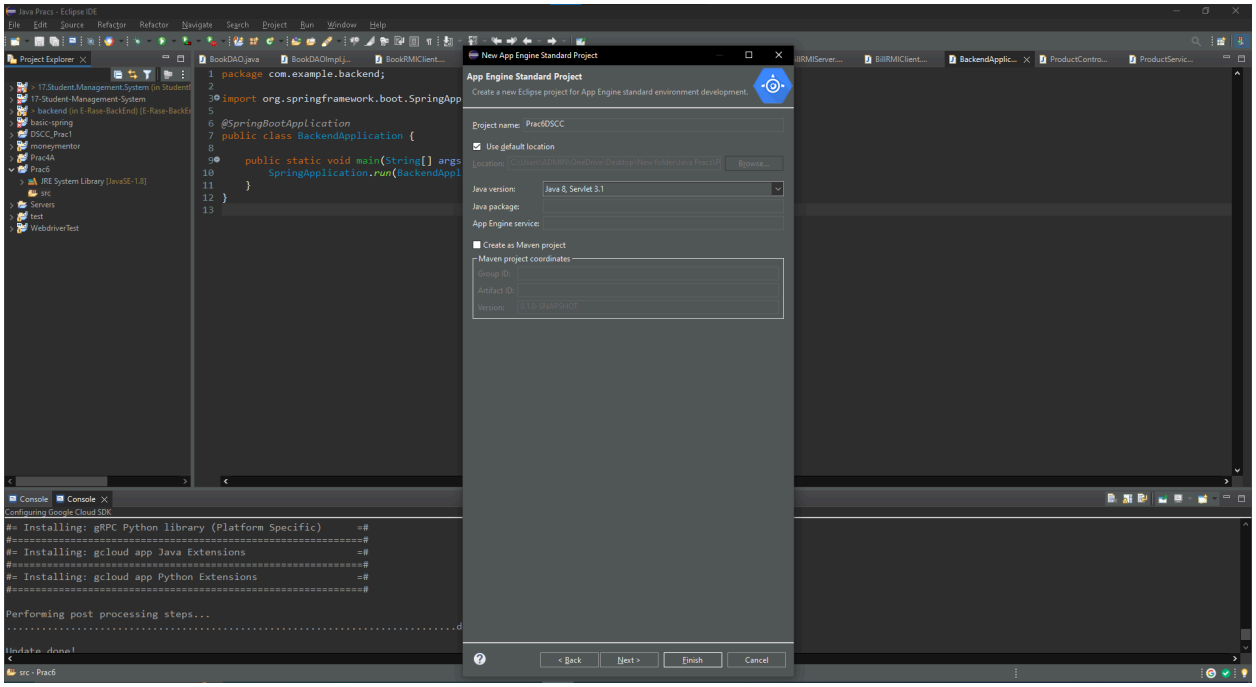






3. **Create a New Project**:
    - Click on File -> New -> Project.
    - Select Google Cloud Platform -> Google App Engine Standard Java Project.

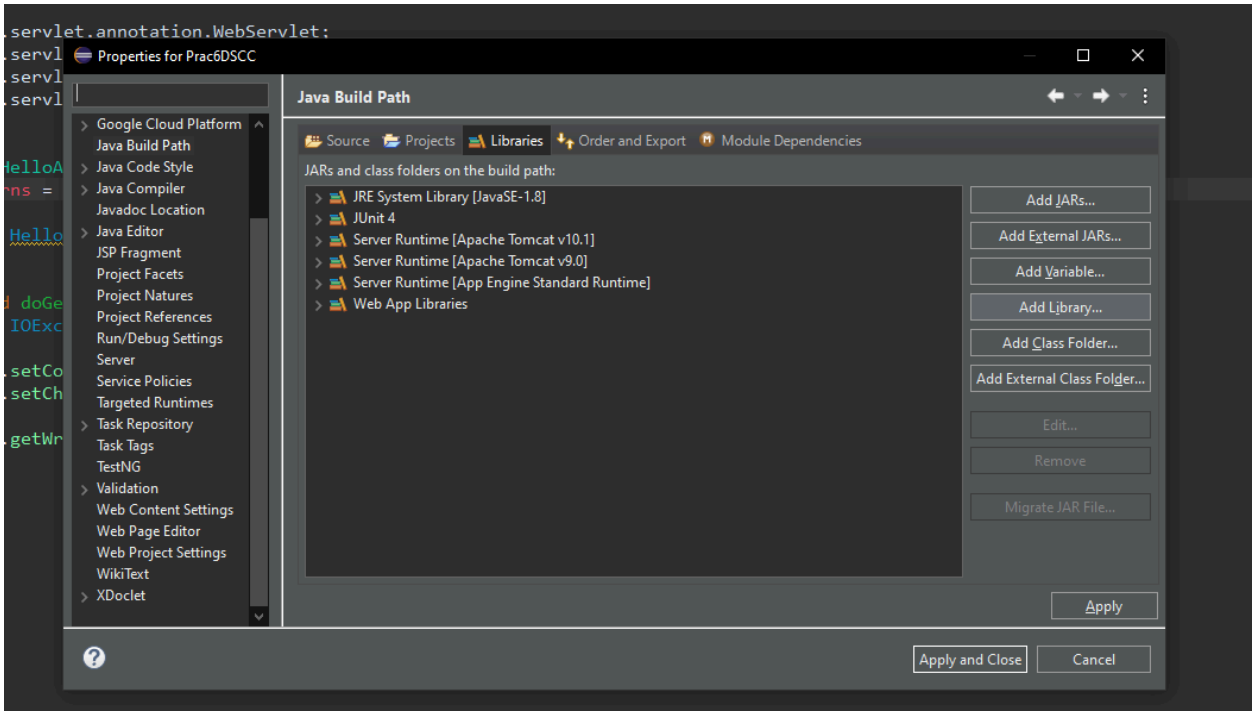




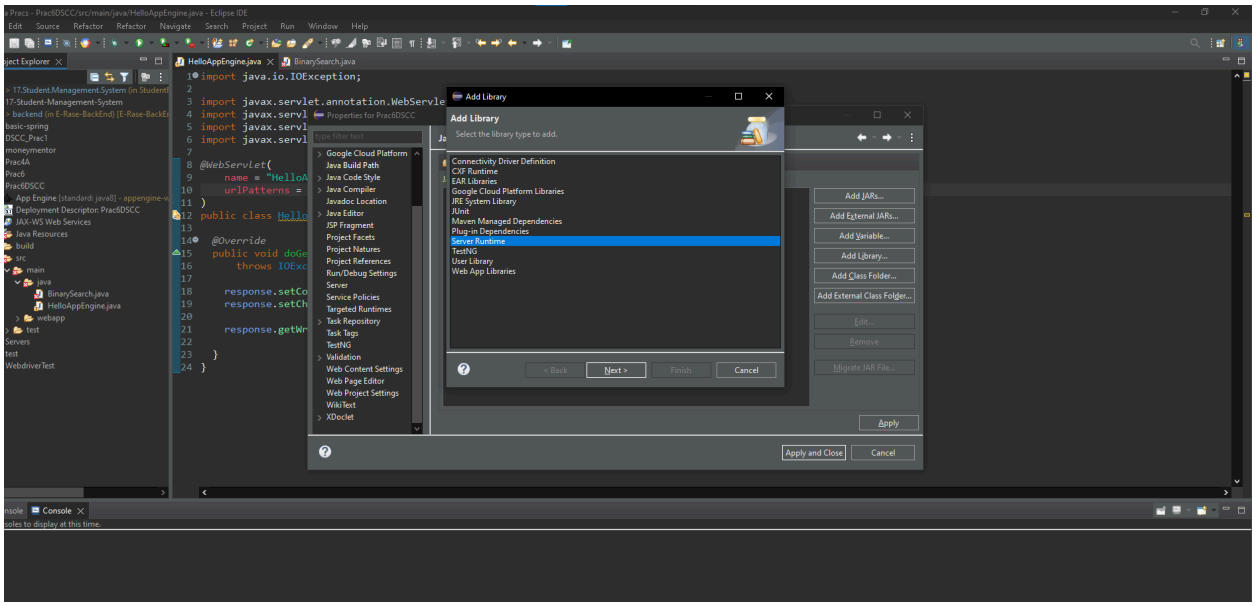
- Click Next, provide a project name (e.g., BinarySearchApp), and configure settings as required.

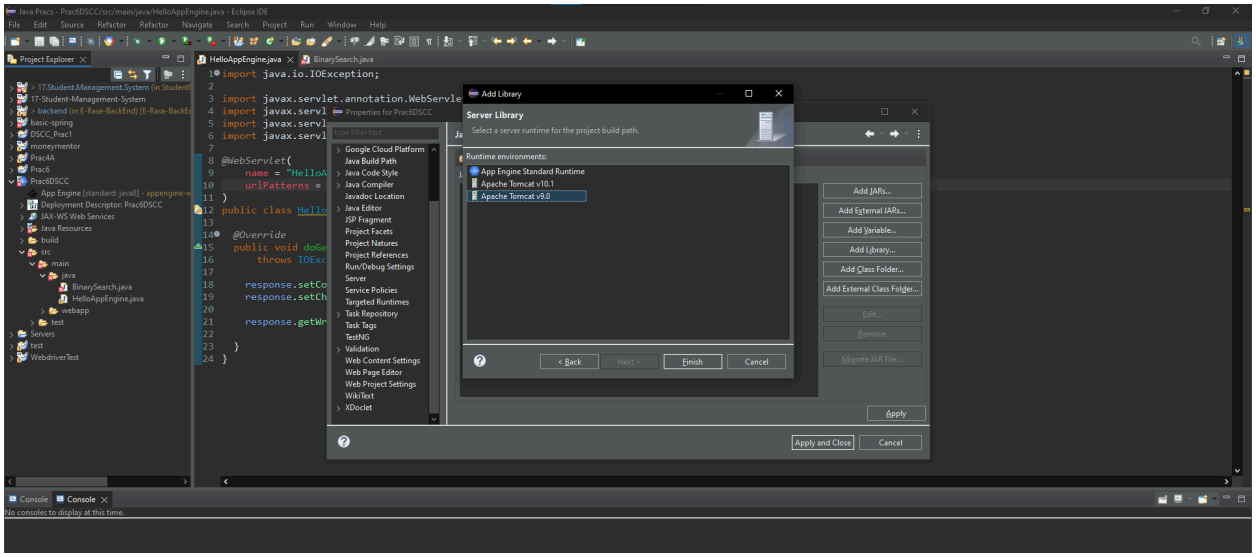
Add Servlet API to Your Project:

- Right-click on your project in the Project Explorer in Eclipse.
- Select **Properties**.
- Go to **Java Build Path** and then click on the **Libraries** tab.



- Click **Add Library**.
- Choose **Server Runtime** (or search for **Servlet API** if available). Choose Tomcat v9.0 and click on Finish





Step 2: Implement the Binary Search Algorithm

1. **Create a new Java class**:
    - Right-click on src/main/java and select New -> Class.
    - Name the class (e.g., BinarySearch).
2. **Implement the binary search method**: Here's a simple implementation of the binary search algorithm:

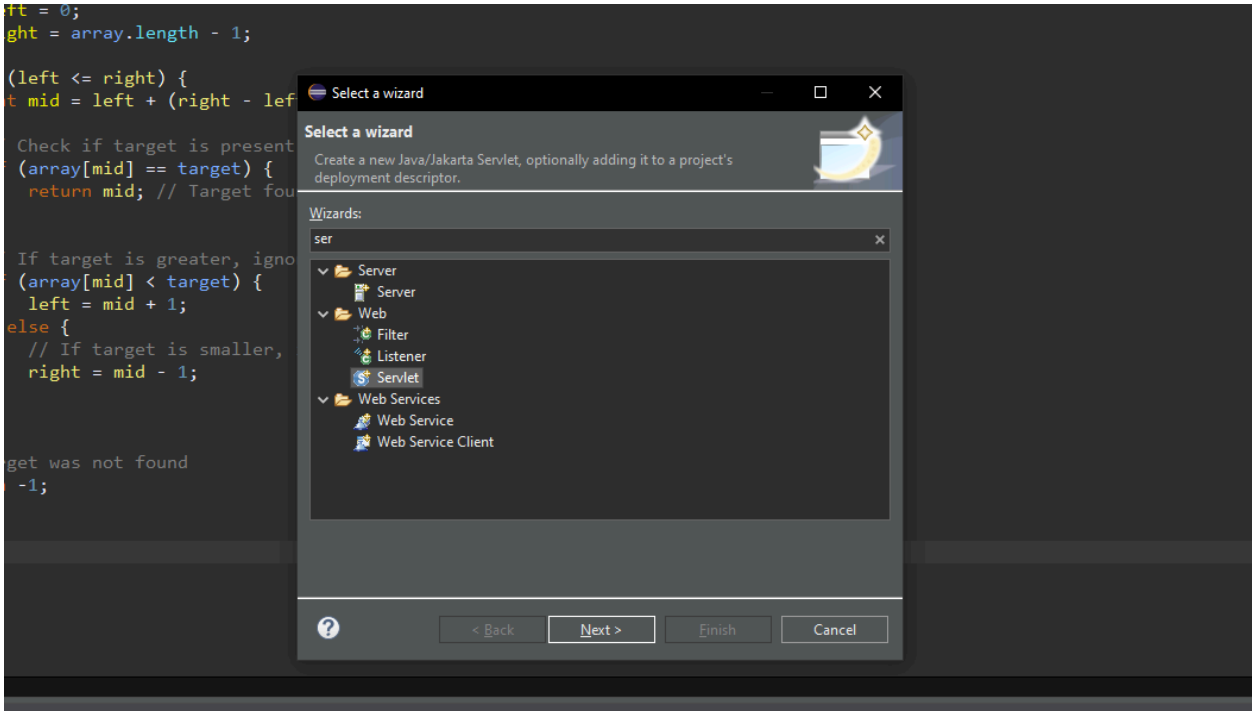
BinarySearch.java

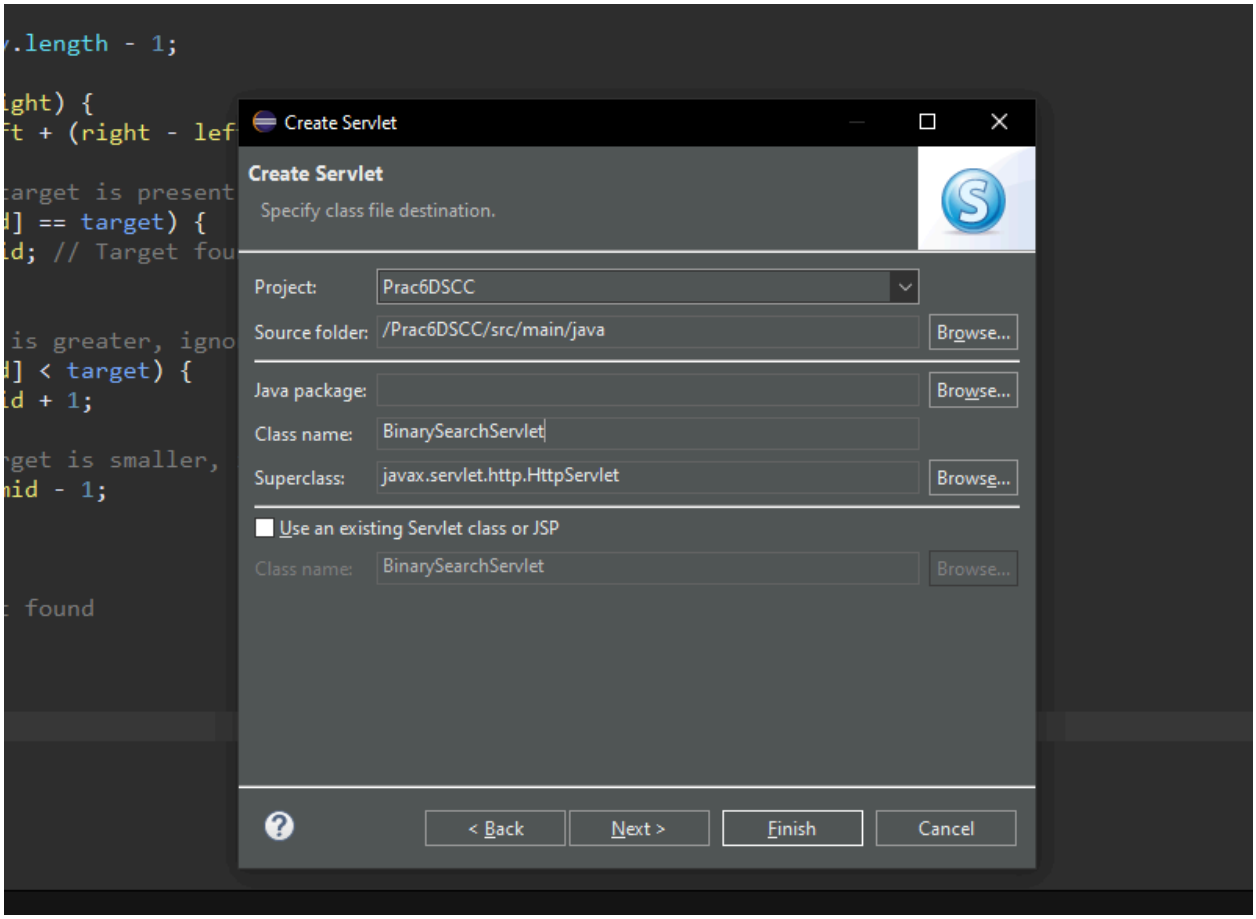
```
public class BinarySearch {

    public static int binarySearch(int[] array, int target) {
        int left = 0;
        int right = array.length - 1;

        while (left <= right) {
            int mid = left + (right - left) / 2;

            // Check if target is present at mid
            if (array[mid] == target) {
```

```
            return mid; // Target found at index mid
        }

        // If target is greater, ignore left half
        if (array[mid] < target) {
            left = mid + 1;
        } else {
            // If target is smaller, ignore right half
            right = mid - 1;
        }
    }

    // Target was not found
    return -1;
  }
}
```

Step 3: Create a Servlet to Handle Requests

1. **Create a new Servlet**:
    - Right-click on src/main/java and select New -> Servlet.
    - Name the servlet (e.g., BinarySearchServlet).


ft = 0;
ght = array.length - 1;
(left <= right) {
t mid = left + (right - lef
Check if target is present
(array[mid] == target) {
return mid; // Target fou
If target is greater, igno
(array[mid] < target) {
left = mid + 1;
else {
// If target is smaller,
right = mid - 1;
get was not found
-1;
Select a wizard
Select a wizard
Create a new Java/Jakarta Servlet, optionally adding it to a project's deployment descriptor.
Wizards:
ser
Server
Server
Web
Filter
Listener
Servlet
Web Services
Web Service
Web Service Client
< Back
Next >
Finish
Cancel



2. **Implement the servlet to handle HTTP requests**: Here's an example of how to set up the servlet:

BinarySearchServlet.java

```
import java.io.IOException;
import javax.servlet.ServletException;
import javax.servlet.annotation.WebServlet;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

@WebServlet("/search")
public class BinarySearchServlet extends HttpServlet {
```

```
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {

        // Get parameters
        String[] arrayStr = request.getParameterValues("array");
        int target = Integer.parseInt(request.getParameter("target"));

        // Convert String array to int array
        int[] array = new int[arrayStr.length];
        for (int i = 0; i < arrayStr.length; i++) {
            array[i] = Integer.parseInt(arrayStr[i]);
        }

        // Perform binary search
        int result = BinarySearch.binarySearch(array, target);

        // Set the result in the response
        response.setContentType("text/plain");
        if (result != -1) {
            response.getWriter().println("Target found at index: " + result);
        } else {
            response.getWriter().println("Target not found.");
        }
    }
}
```

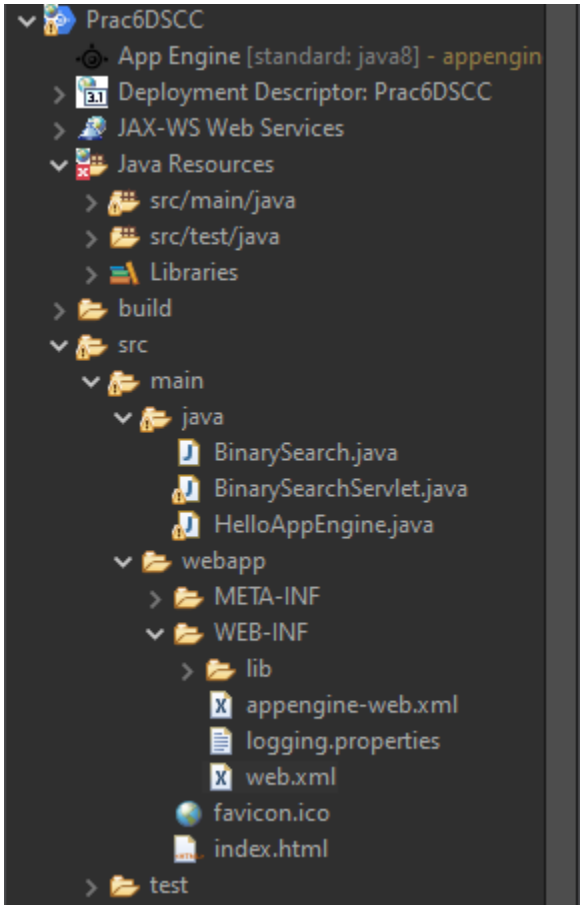
Step 4: Deploy the Application

1. **Configure Deployment Descriptor**: Make sure to define your servlet in the web.xml file under src/main/webapp/WEB-INF/.

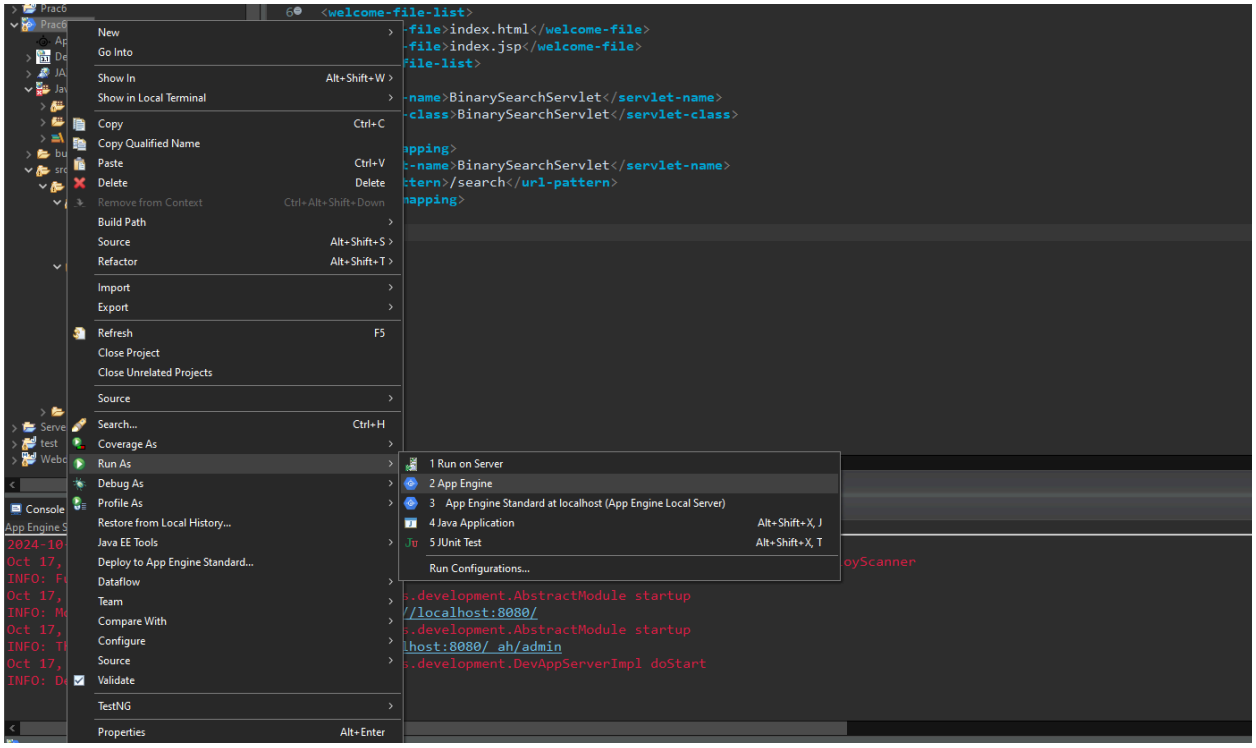
web.xml

```
<?xml version="1.0" encoding="utf-8"?>
<web-app xmlns="http://xmlns.jcp.org/xml/ns/javaee"
        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
                                                xsi:schemaLocation="http://xmlns.jcp.org/xml/ns/javaee
http://xmlns.jcp.org/xml/ns/javaee/web-app_3_1.xsd"
        version="3.1">
  <welcome-file-list>
    <welcome-file>index.html</welcome-file>
    <welcome-file>index.jsp</welcome-file>
  </welcome-file-list>
  <servlet>
    <servlet-name>BinarySearchServlet</servlet-name>
    <servlet-class>your.package.name.BinarySearchServlet</servlet-class>
  </servlet>
  <servlet-mapping>
          <servlet-name>BinarySearchServlet</servlet-name>
          <url-pattern>/search</url-pattern>
  </servlet-mapping>
</web-app>
```

Folder Structure

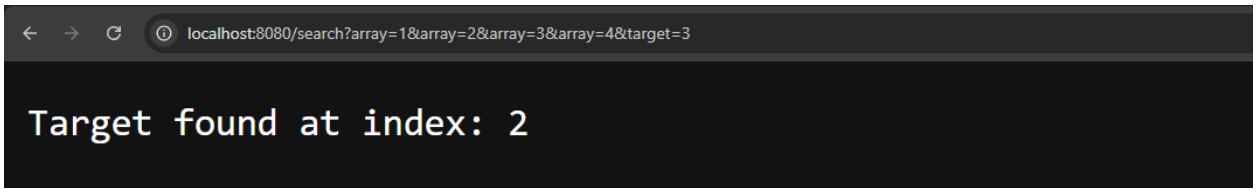


2. **Run the application**:
   - Right-click on your project in the Project Explorer.
   - Select **Run as > App Engine**.



Once the application start running, On browser URL, search for

http://localhost:8080/search?array=1&array=2&array=3&array=4&target=3



Conclusion:

In this practical, we successfully developed a web application using Google App Engine in Eclipse IDE that implements a binary search algorithm to find the position of a target value within a sorted integer array. By setting up the Google Cloud Tools plugin, writing the binary search logic, and configuring servlets to handle HTTP requests, we demonstrated how to efficiently search through data using a fundamental algorithm in a cloud environment. Additionally, resolving issues such as missing dependencies (e.g., Servlet API) further reinforced the importance of proper project configuration when working with web applications.

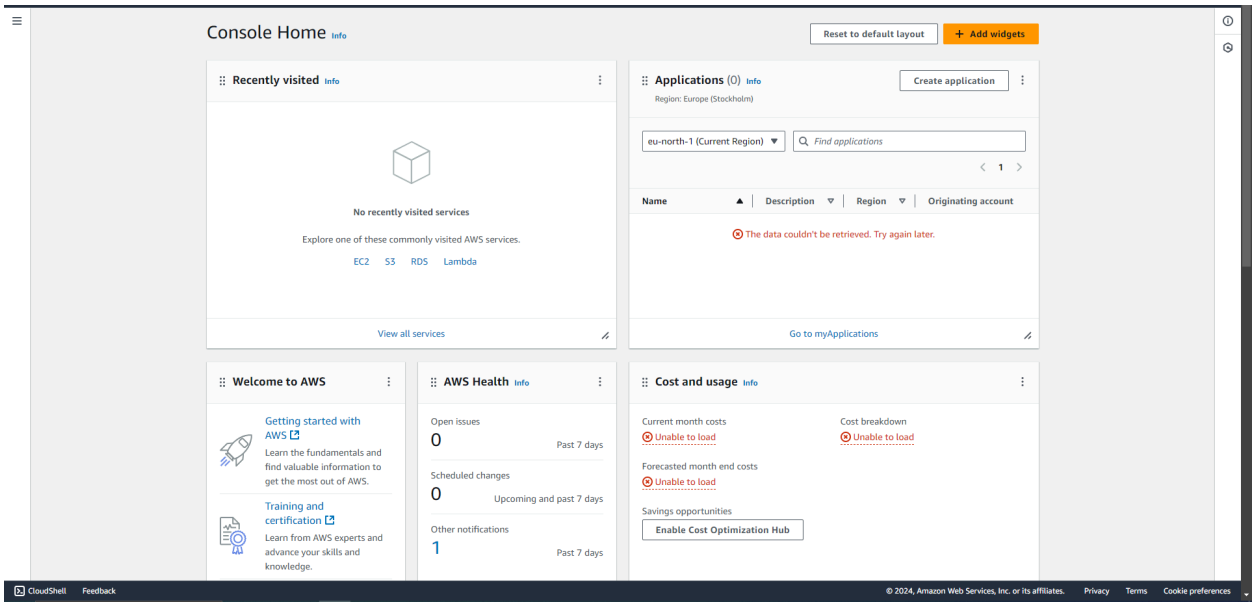
Name of Student: Pushkar Sane		
Roll Number: 45	**Lab Assignment Number: 7**	
Title of Lab Assignment: Identity Access Management		
DOP: 21-10-2024	**DOS: 21-10-2024**	
CO Mapped: **CO5**	**PO Mapped:** **PO1, PO2, PO3, PO7, PO9, PSO1**	**Signature:**

Practical No. 7

Aim: Identity Access Management

AWS

Amazon Web Services (AWS)



Amazon Web Services (AWS) is a comprehensive and widely adopted cloud computing platform offered by Amazon. It provides a variety of cloud services, enabling businesses and individuals to access computing power, storage, and other functionalities on-demand via the internet.

AWS Security

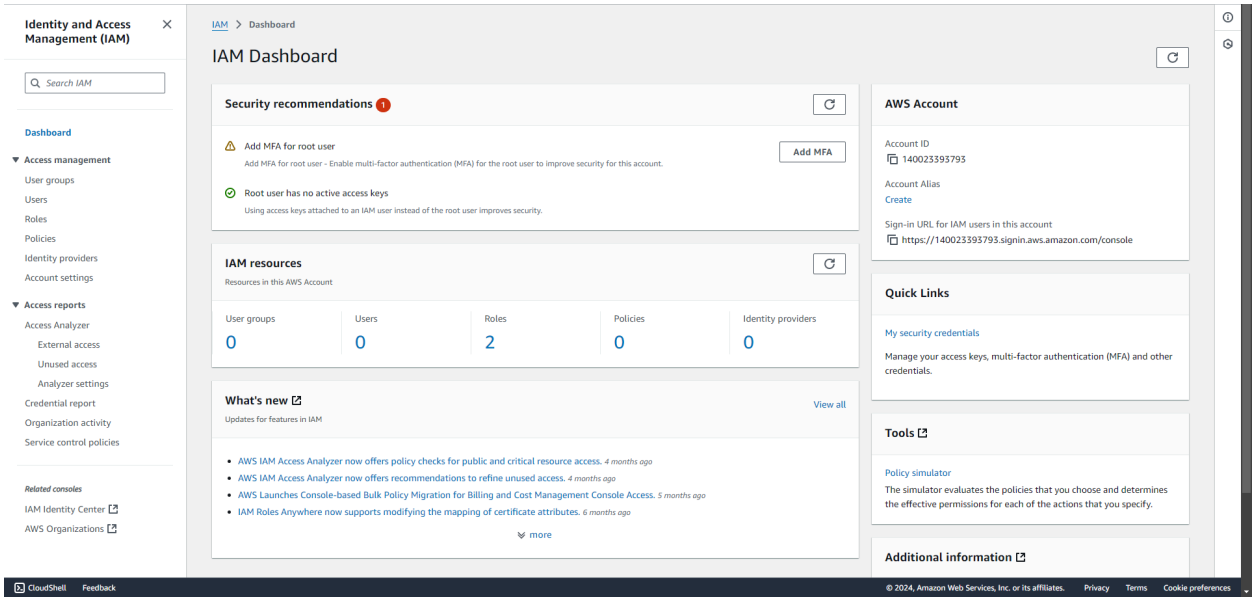
AWS Security is a comprehensive framework that includes tools, policies, and best practices to protect customer data, applications, and services within the AWS cloud environment. AWS adopts a **Shared Responsibility Model**, where:

- **AWS's Responsibilities**: AWS manages the security of the cloud infrastructure, which includes hardware, software, networking, and facilities. This involves maintaining the physical security of data centers, patching underlying infrastructure, and ensuring compliance with various security standards.
- **Customer's Responsibilities**: Customers are responsible for securing their applications, managing access to their data, configuring security controls (like IAM), and ensuring compliance with relevant regulations.

AWS provides various security services and features, including:

- **Identity and Access Management (IAM)**: Manage access to AWS services and resources.
- **AWS Shield and AWS WAF**: Protect applications against Distributed Denial of Service (DDoS) attacks and web exploits.
- **AWS Key Management Service (KMS)**: Manage cryptographic keys for data encryption.
- **Amazon CloudTrail**: Track user activity and API usage for auditing purposes.

What is IAM?



Identity and Access Management (IAM) is a critical service within AWS that allows administrators to control user access to AWS services and resources securely. IAM provides the following capabilities:

- **User Management**: Create and manage AWS users and their associated access permissions.
- **Access Control**: Define who can access what resources and under what conditions.
- **Policy Management**: Create policies that dictate permissions for actions on specific resources.

IAM ensures that users have the minimum level of access necessary to perform their jobs (principle of least privilege), which helps reduce the risk of unauthorized access.

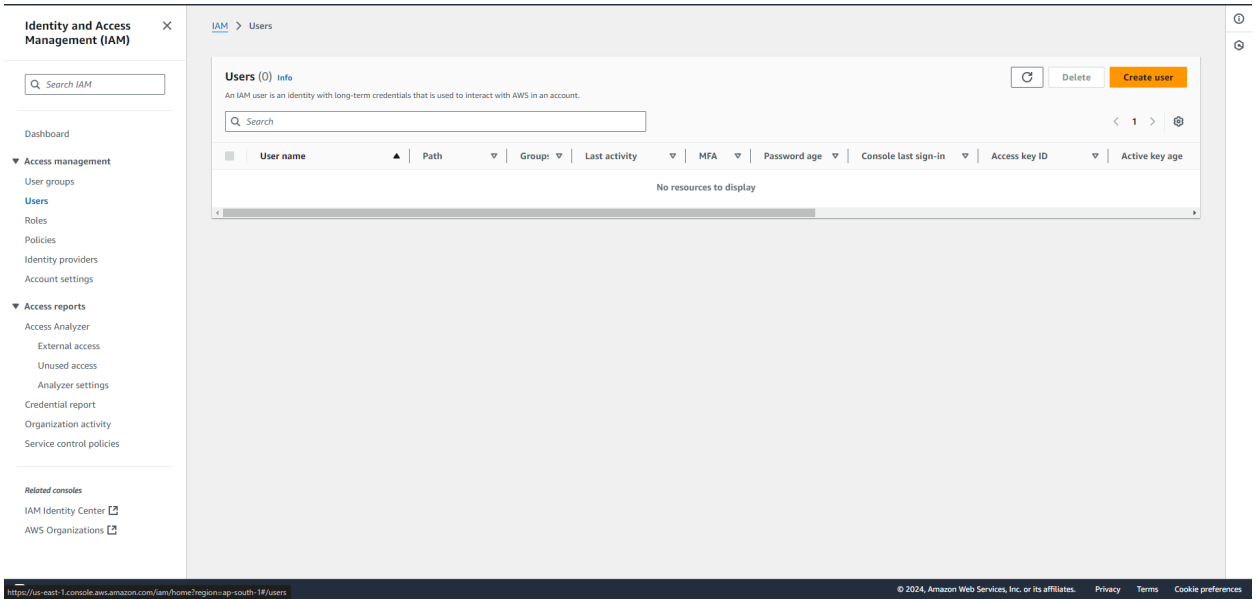
How Does IAM Work?

IAM operates through several key processes and concepts:

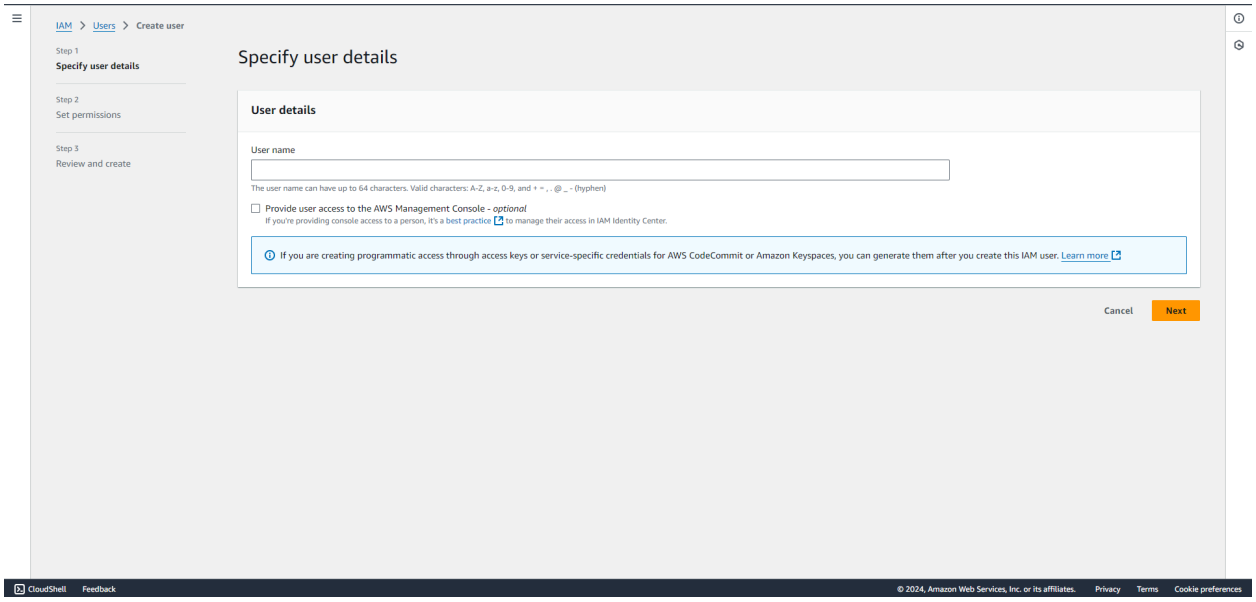
1. **User Creation**: Administrators can create IAM users, each with unique security credentials (username and password) to access AWS resources.
2. **Groups**: Users can be organized into groups to simplify permission management. Instead of assigning permissions to each user individually, permissions can be assigned to a group, which then applies to all users in that group.
3. **Policies**: IAM uses policies to grant or restrict access to resources. Policies are JSON documents that specify:
    - **Effect**: Whether the policy allows or denies access.
    - **Action**: The specific actions (e.g., s3:PutObject, ec2:StartInstances) the policy permits or denies.
    - **Resource**: The specific AWS resources (e.g., Amazon S3 buckets, EC2 instances) to which the actions apply.
4. **Roles**: Roles are IAM identities that have specific permissions but are not associated with a specific user. Instead, roles can be assumed by users, applications, or AWS services, enabling temporary access without needing to manage long-term credentials.
5. **Authentication and Authorization**: When a user or application requests access to an AWS resource, IAM first authenticates the identity (verifying the provided credentials). Then, it checks the permissions associated with that identity against the requested action and resource to determine if access should be granted or denied.

Create a New IAM User

1. **Select Users**: In the left navigation pane, click on **Users**.

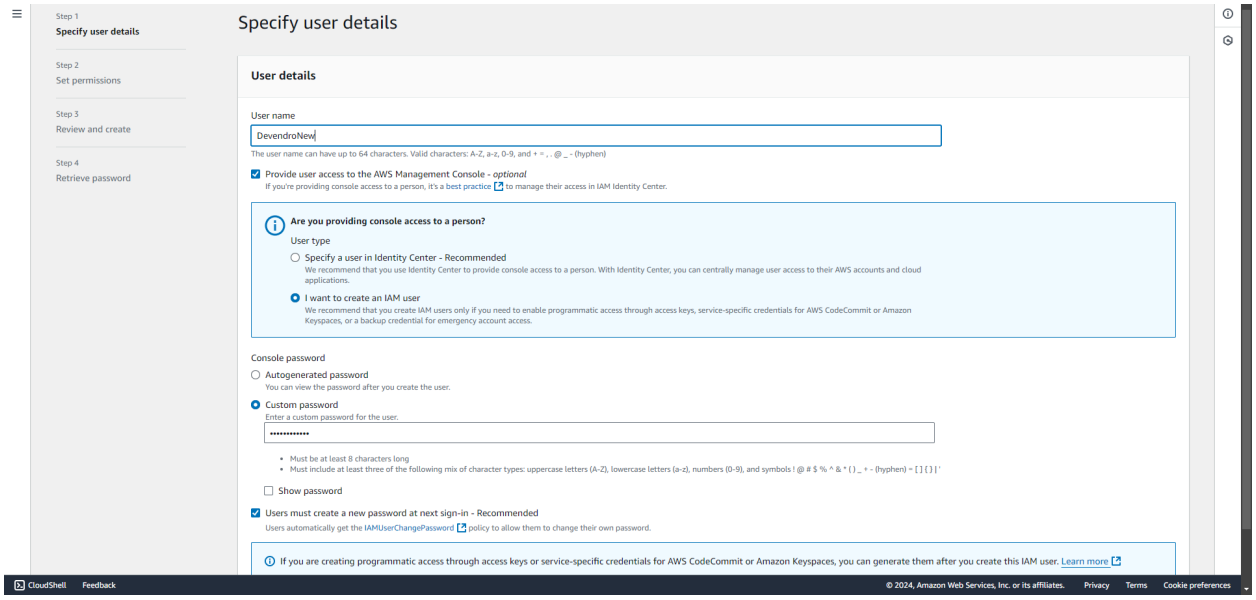


2. **Add User**: Click the **Create user** button.



3. **Enter User Details**:

- **User Name**: Provide a unique username for the new user.
- **Access Type**: Choose the type of access:
  - **Programmatic access**: Provides an Access Key ID and Secret Access Key for AWS CLI, SDK, and APIs.
  - **AWS Management Console access**: Allows the user to sign in to the console. If selected, set a password (either autogenerate or create a custom one).

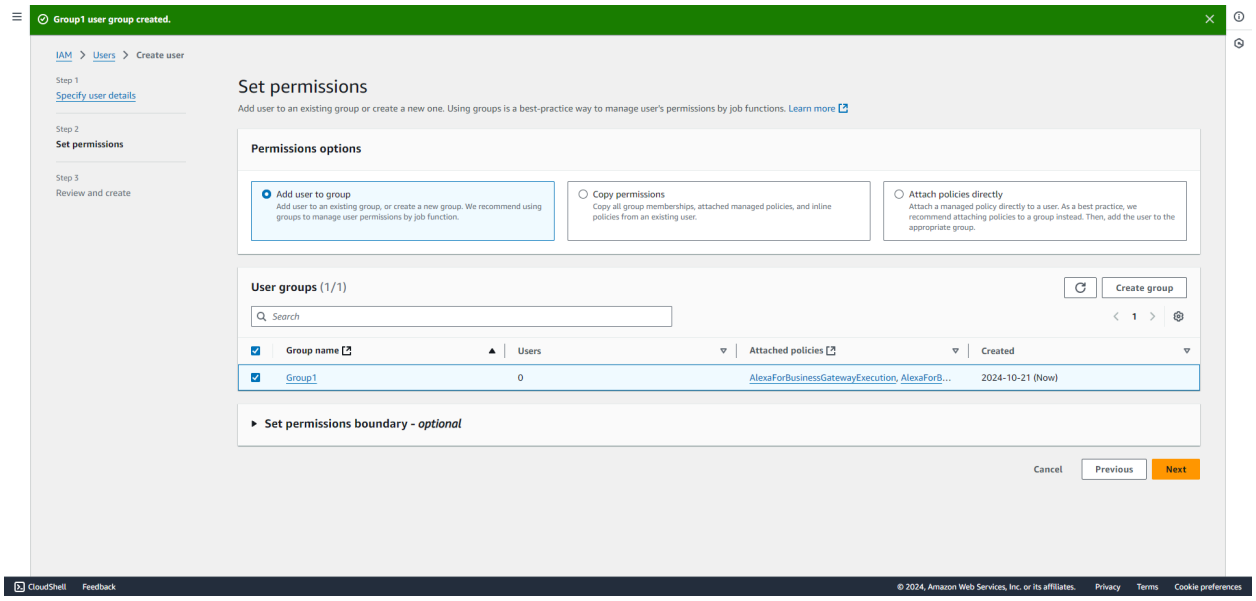


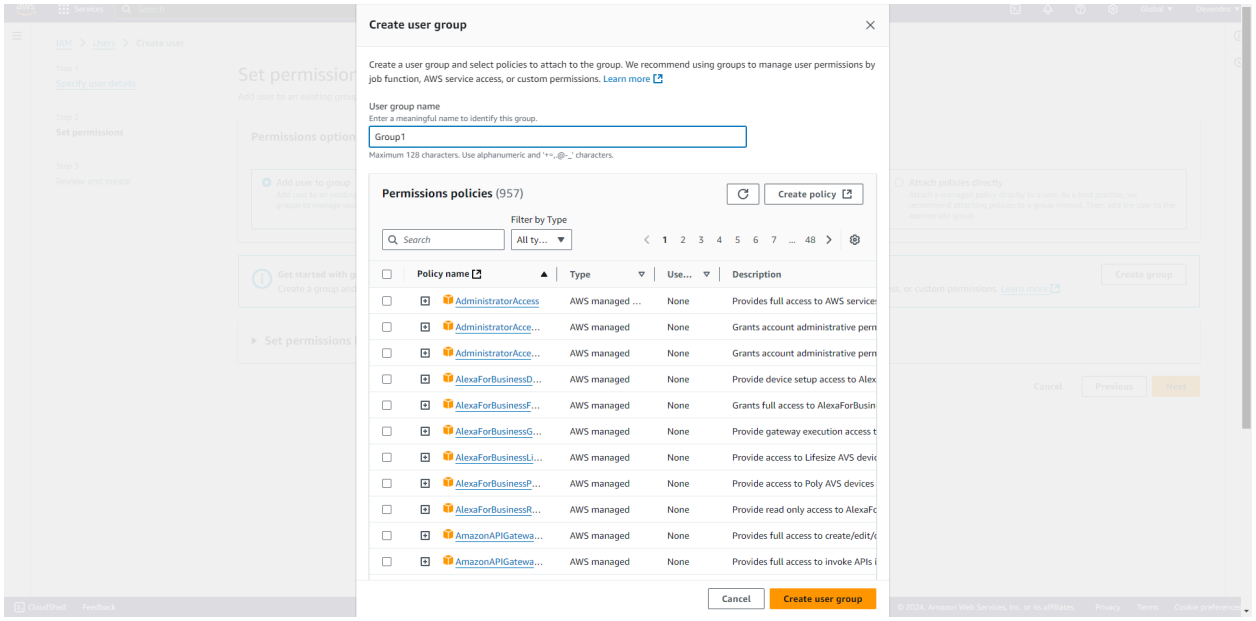
4. **Click Next**: After filling in the details, click the **Next: Permissions** button.

Step 4: Set Permissions for the User

1. **Attach Existing Policies**:
   - Choose from existing policies to assign specific permissions. You can filter policies based on service or type.
   - Select the appropriate policies based on the user's role (e.g., Administrator, Read-only, etc.).
2. **Create Group** (Optional):
   - You can create a group and assign policies to that group, then add the user to the group.

- To do this, click on **Create group**, enter a group name, select policies, and click **Create group**.
- Select the group you created to add the user to that group.





3. **Add Inline Policy** (Optional):

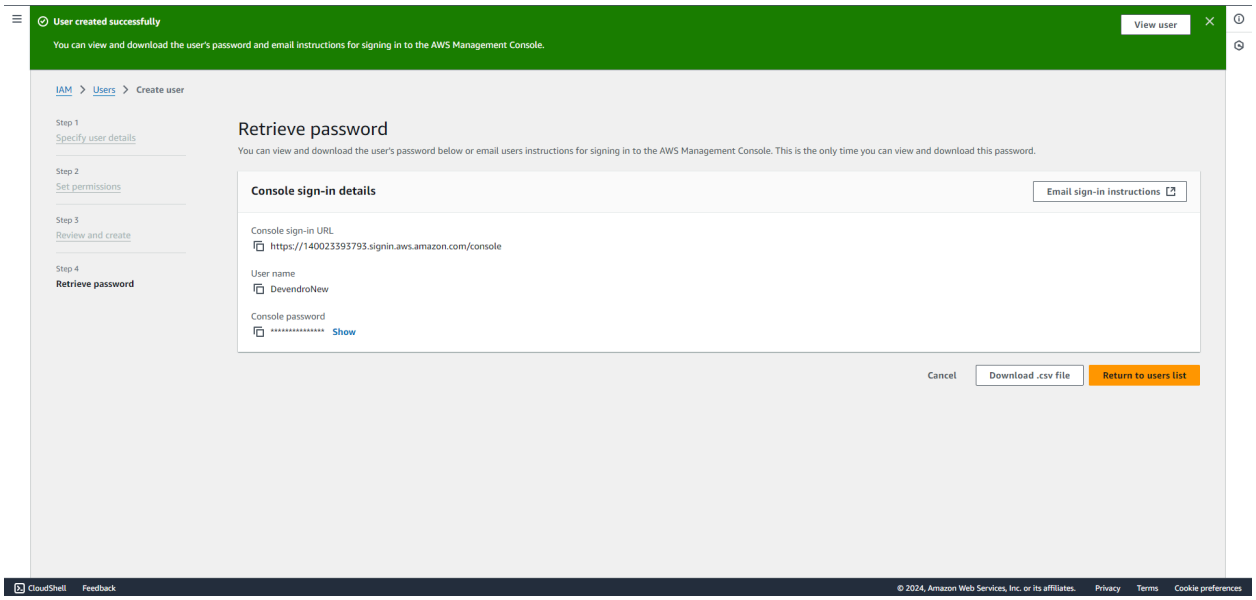
- If you need specific permissions not covered by existing policies, you can create an inline policy.
- Click **Next: Tags** to proceed.

Step 5: (Optional) Add Tags

1. **Add Tags**: Tags are key-value pairs used for organizing and managing IAM resources. You can add tags to the user for easier identification.
2. **Click Next**: After adding tags (if any), click **Next: Review**.

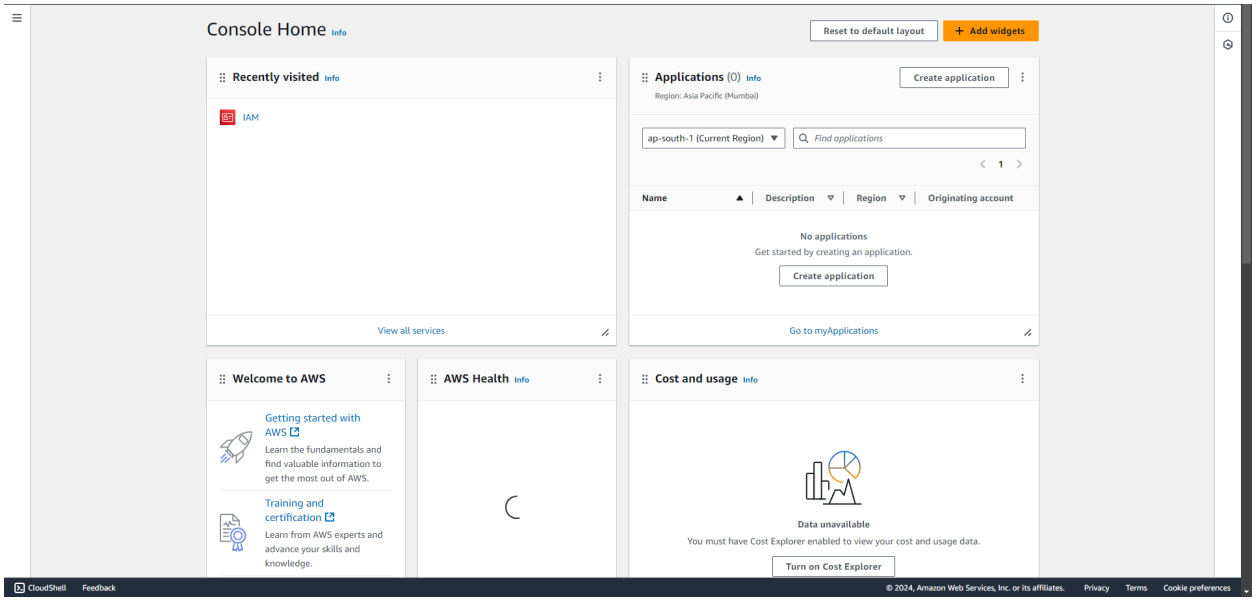
Step 6: Review and Create User

1. **Review Settings**: Check the user details, permissions, and tags.
2. **Create User**: Click the **Create user** button.



Step 7: Sign In as the New User

1. **Sign-In Link**: To log in to the AWS Management Console as the new user, use the sign-in link provided on the confirmation page or generate it from the IAM dashboard.
2. **Enter Credentials**: Use the new username and password (if console access was enabled) to sign in.



Features of IAM

IAM offers numerous features that enhance security and streamline management:

1. **Granular Permissions**: IAM policies allow administrators to specify detailed permissions at various levels (individual actions on specific resources), providing fine-grained control over access.
2. **Centralized Management**: IAM provides a unified interface to manage users, groups, roles, and permissions across all AWS services.
3. **Role-Based Access Control**: The ability to create roles simplifies permission management, especially in scenarios where temporary access is needed (e.g., granting AWS Lambda functions access to S3).
4. **Multi-Factor Authentication (MFA)**: MFA can be enforced to enhance security, requiring users to authenticate using a second factor in addition to their password.
5. **Temporary Credentials**: IAM roles allow the provision of temporary security credentials for users and applications, reducing the risks associated with long-term credentials.
6. **Policy Versioning**: IAM supports versioning of policies, enabling administrators to update permissions while preserving older versions for auditing or rollback purposes.
7. **Audit and Monitoring**: AWS CloudTrail can track IAM actions, allowing organizations to monitor access and changes to IAM settings, providing visibility into who accessed what resources and when.

Multi-Factor Authentication (MFA)

Multi-Factor Authentication (MFA) is a crucial security feature in IAM that adds an extra layer of protection. MFA requires users to provide two or more verification factors when logging in, significantly reducing the risk of unauthorized access.

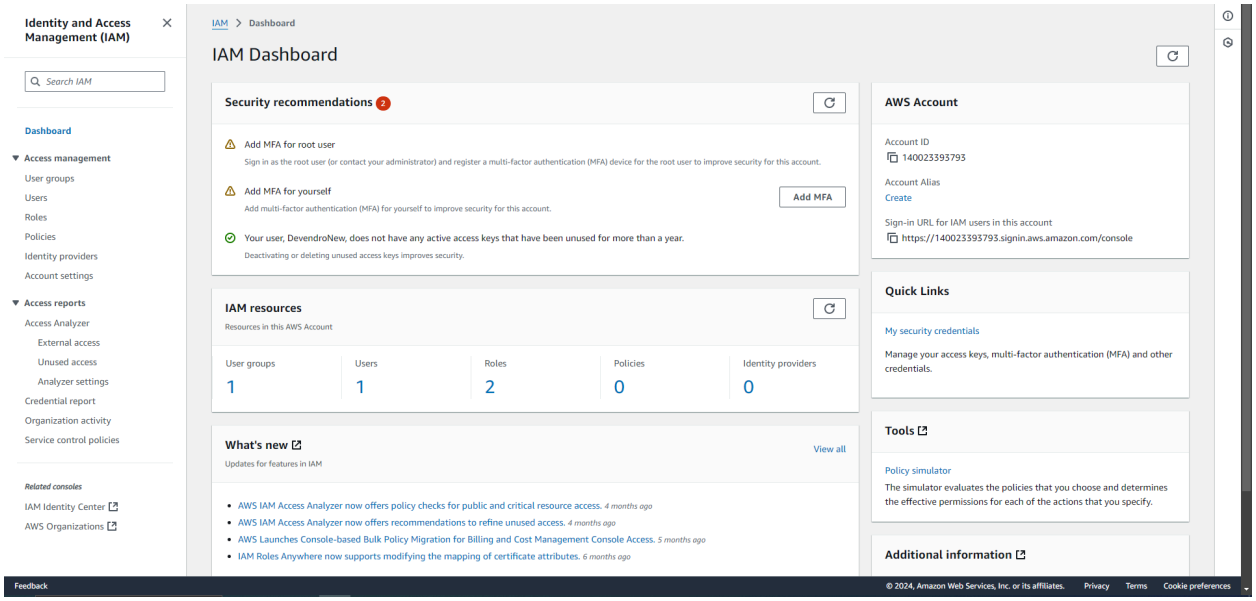
How MFA Works:

1. **MFA Device Options**:
    - **Virtual MFA**: Users can install a virtual MFA application (e.g., Google Authenticator, AWS MFA) on their mobile device. The app generates a time-based one-time password (TOTP).
    - **Hardware MFA**: Physical devices (e.g., YubiKey) can be used to generate authentication codes.
    - **SMS MFA**: Users can receive a one-time code via SMS. While convenient, this method is less secure than the others.
2. **Enabling MFA**: Administrators can enforce MFA for specific users or groups. When MFA is enabled for an IAM user, the user must configure their MFA device and use it during the login process.
3. **Authentication Process**:
    - During login, the user enters their username and password.
    - After successful password entry, they are prompted to enter the MFA code generated by their MFA device.
    - The code is time-sensitive (typically valid for 30 seconds), requiring users to enter the latest code.
4. **Role Assumption with MFA**: IAM roles can also require MFA for access. When a user assumes a role, they must provide the MFA code along with their request, ensuring that only authorized users can access sensitive resources.
5. **Use Cases for MFA**:
    - Accessing AWS Management Console.
    - Using AWS CLI or SDKs for programmatic access.
    - Assuming IAM roles for sensitive operations.

How to Enable MFA

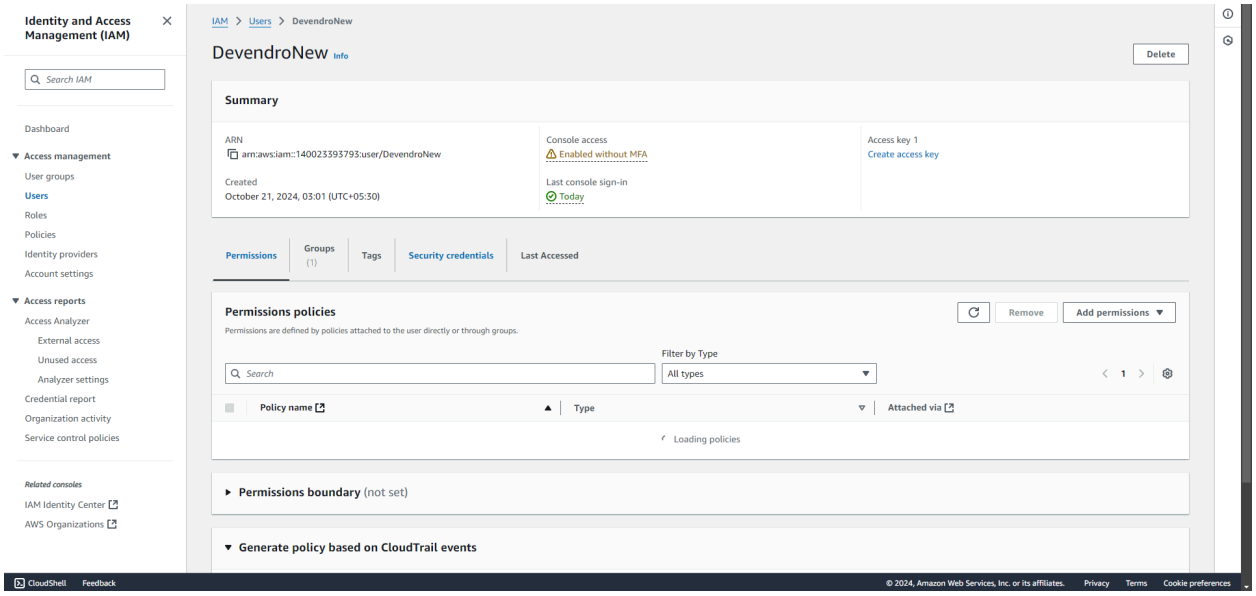
Step 1: Access the IAM Dashboard

1. **Navigate to IAM**: In the AWS Management Console, type "IAM" in the search bar and select **IAM** from the results.
2. **IAM Dashboard**: You will see the IAM dashboard where you can manage users, groups, roles, and policies.



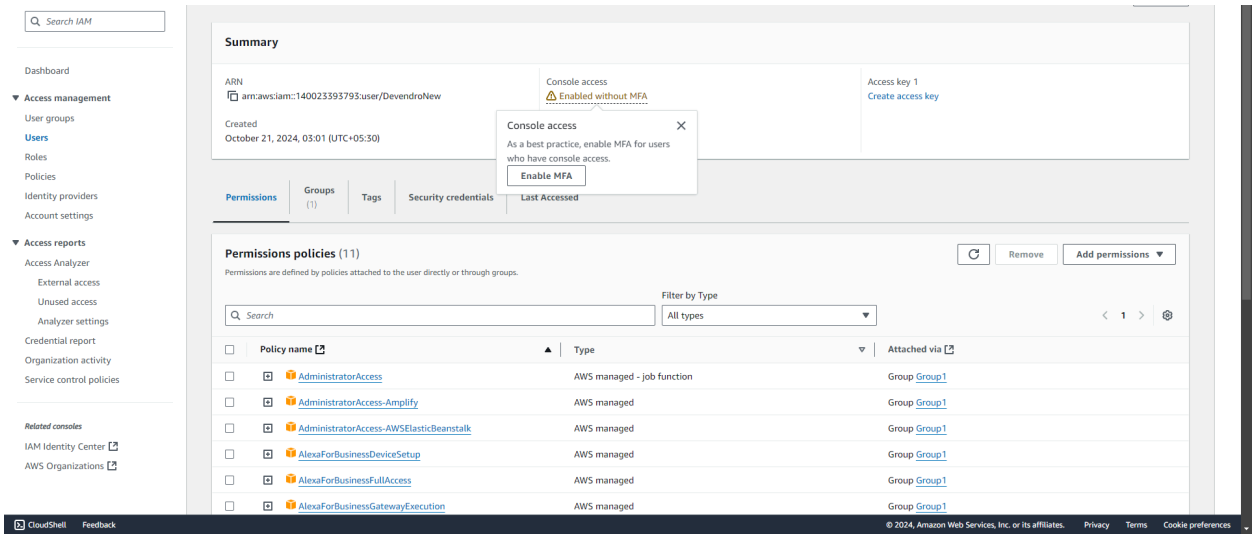
Step 2: Select the User

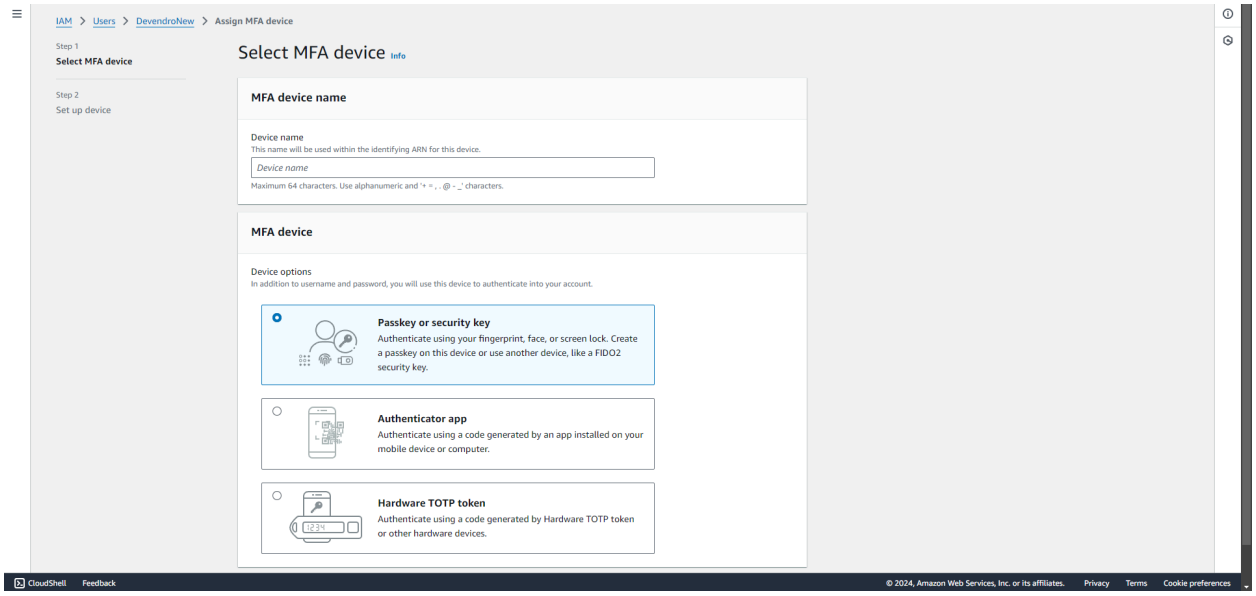
1. **Choose Users**: In the left navigation pane, click on **Users**.
2. **Select the User**: Click on the name of the user for whom you want to enable MFA.



Step 3: Enable MFA

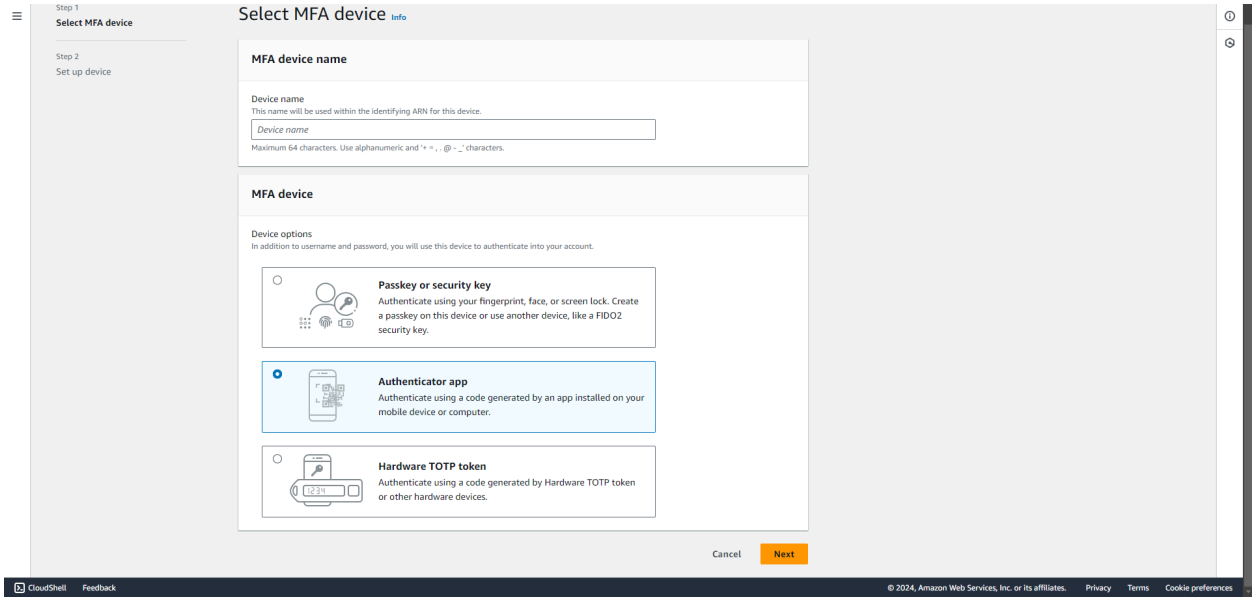
1. **Security Credentials Tab**: Navigate to the **Security credentials** tab for the selected user.
2. **Manage MFA Device**: Find the **Multi-Factor Authentication (MFA)** section and click on **Manage** next to it.





Step 4: Choose MFA Device Type

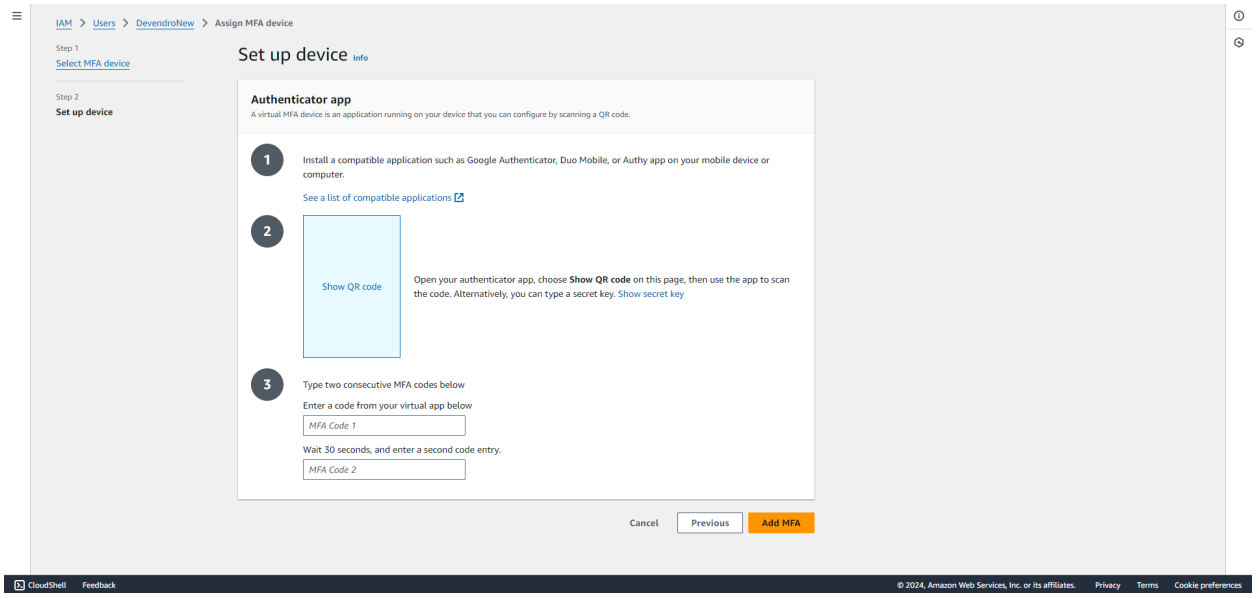
1. **Select MFA Device Type**: You will be presented with different MFA device options. Choose one of the following:
    - **Virtual MFA device**: This option allows you to use an authenticator app (like Google Authenticator, Authy, or AWS Virtual MFA).
    - **U2F security key**: If you have a hardware security key (like YubiKey), you can select this option.
    - **SMS MFA**: You can receive a one-time code via SMS to your mobile phone (less recommended due to security concerns).
2. **Click Next**: After selecting the MFA device type, click **Next**.



Step 5: Configure the MFA Device

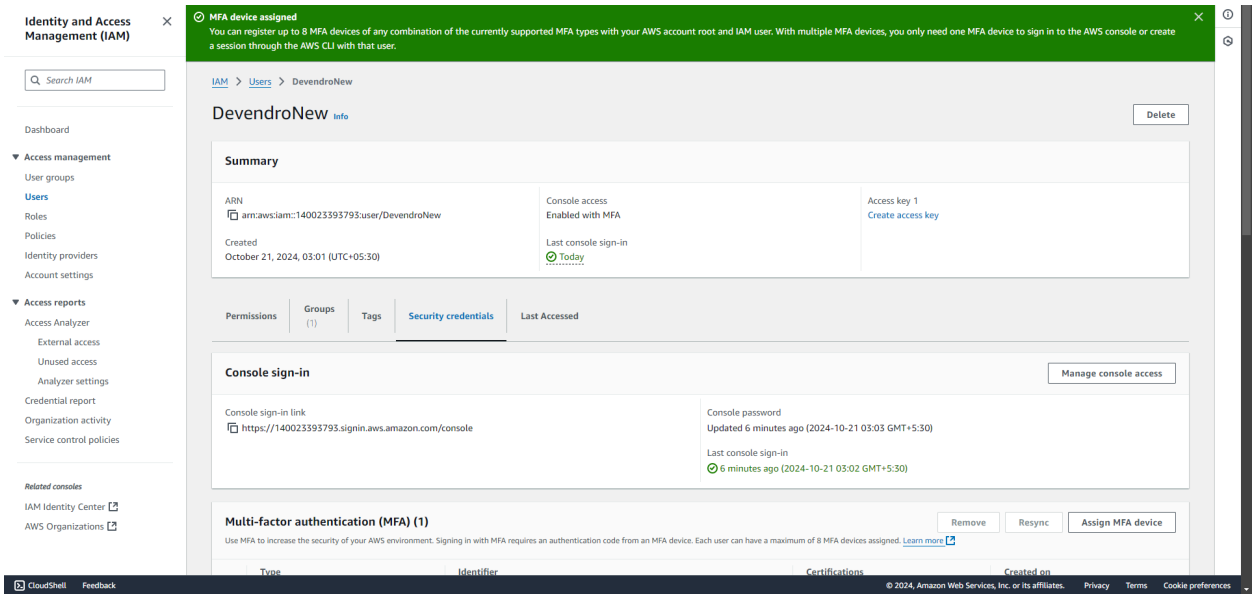
For Virtual MFA Device:

1. **Install Authenticator App**: Ensure you have an authenticator app installed on your smartphone.
2. **Scan QR Code**:
    - You will see a QR code on the screen. Open your authenticator app and scan the QR code.
    - Alternatively, you can enter the provided secret key manually into your app.
3. **Enter MFA Codes**:
    - In the IAM console, enter two consecutive one-time passwords (OTPs) generated by your authenticator app.
    - These codes must be entered within a short time frame (usually within 30 seconds).
4. **Click Assign MFA**: Once both codes are validated, click **Assign MFA** to complete the setup.



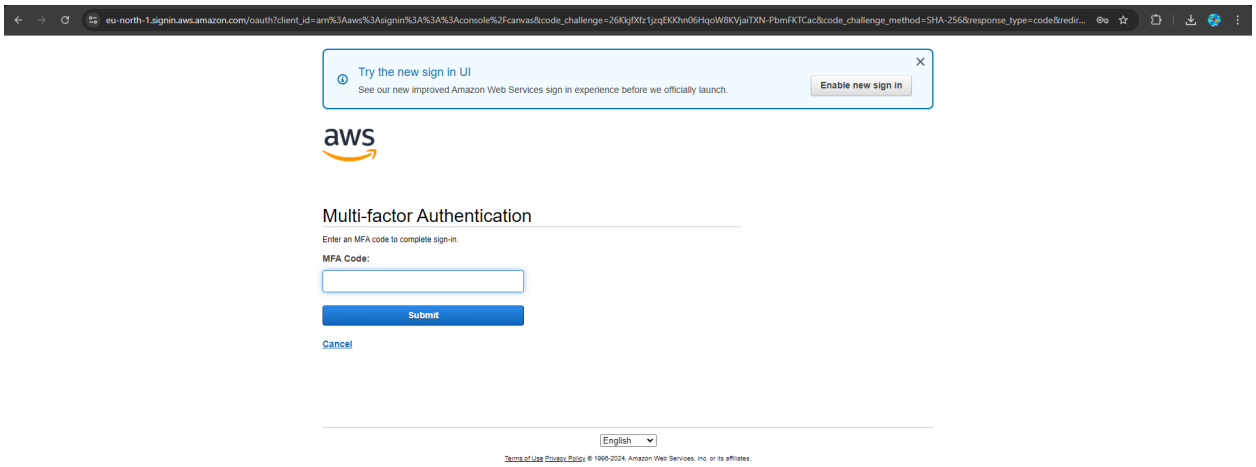
Step 6: Confirmation

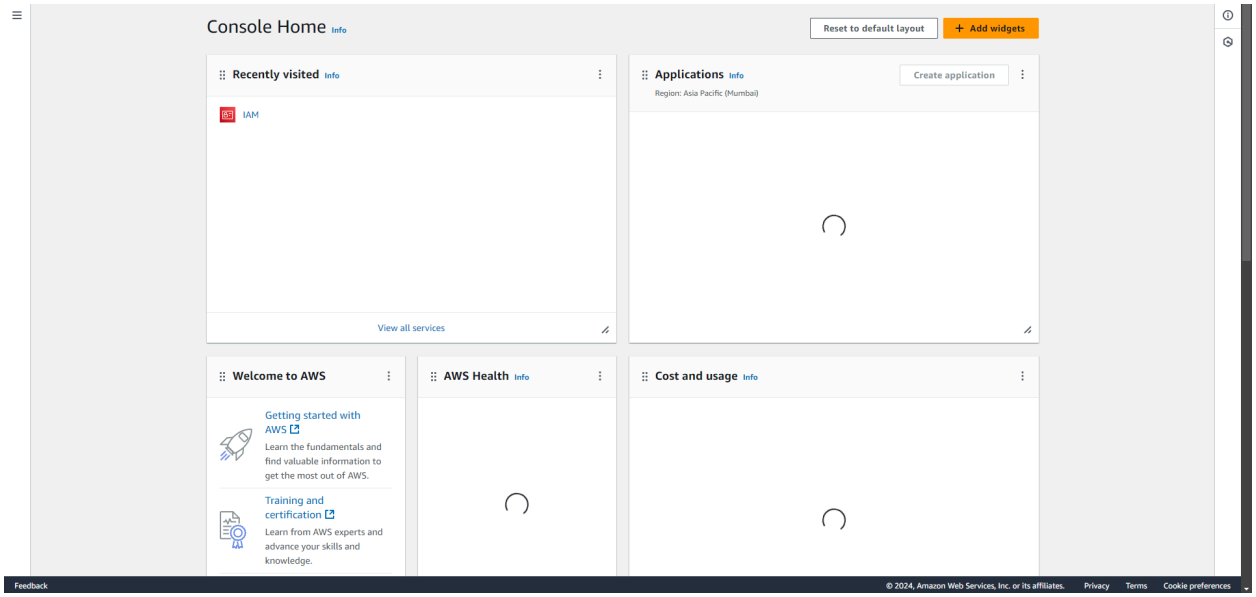
- **Success Message**: Once MFA is successfully enabled, you'll see a confirmation message indicating that MFA has been configured for the user.



Step 7: Sign In Using MFA

1. **Sign Out and Sign In Again**: Log out of the AWS Management Console and attempt to sign in again with the IAM user.
2. **Enter MFA Code**: After entering the username and password, you will be prompted to enter the MFA code generated by your authenticator app or received via SMS.
3. **Access AWS Resources**: Once the correct MFA code is entered, you will gain access to the AWS Management Console.





Conclusion:

AWS Security, particularly through IAM, plays a vital role in protecting cloud resources. By leveraging IAM's features, organizations can implement stringent access controls, enhance security with Multi-Factor Authentication, and ensure that users have appropriate permissions based on their roles. This robust framework not only helps safeguard sensitive data but also streamlines the management of user access across the AWS ecosystem, aligning with best practices for cloud security.

Name of Student: Pushkar Sane			
Roll Number: 45		**CA Assignment No: 1**	
Title of CA Assignment: Cloud Services.			
DOP: 09-10-2024		**DOS: 11-10-2024**	
CO Mapped:	**PO Mapped:**	**Signature:**	**Marks:**

CA Assignment No. 1

1. Amazon EC2

a. Create an Amazon Web Services account

b. Log into the AWS account and open the Amazon EC2 console. Click on 'Launch instance' button to create your own instance

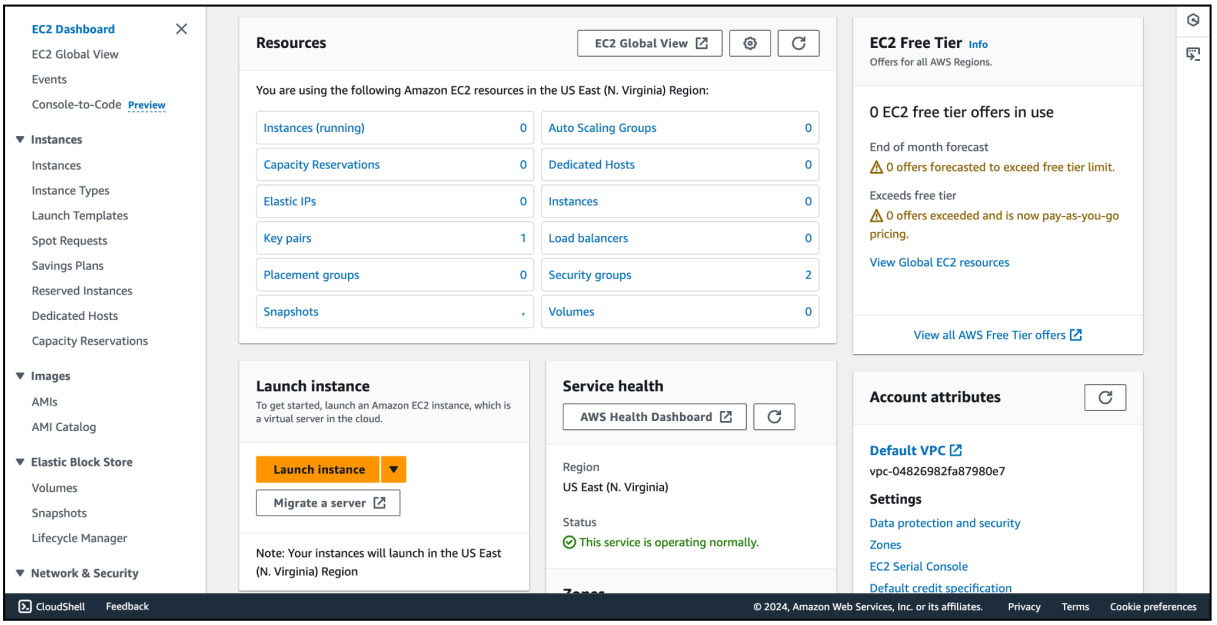
c. Select an 'Ubuntu server'. In the create instance wizard select t1.micro instance type. Proceed with the instance creation wizard with default settings and launch the instance.

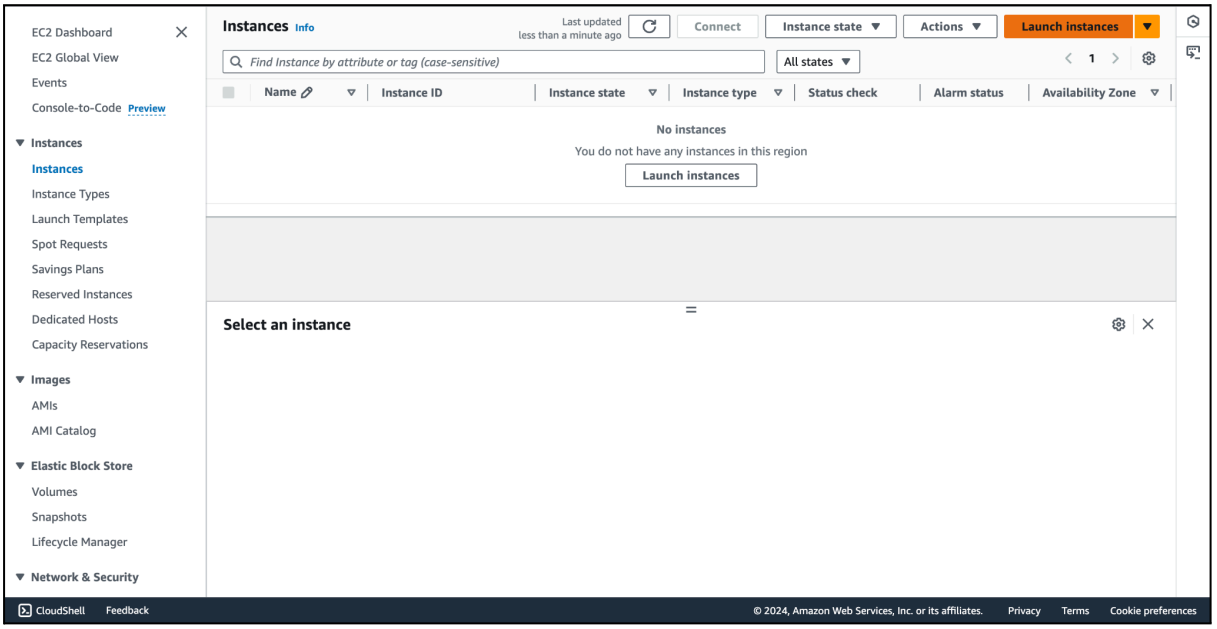
d. View the instance status in the console and wait till the status becomes running. When the status becomes running, note the public DNS of the instance from the console.

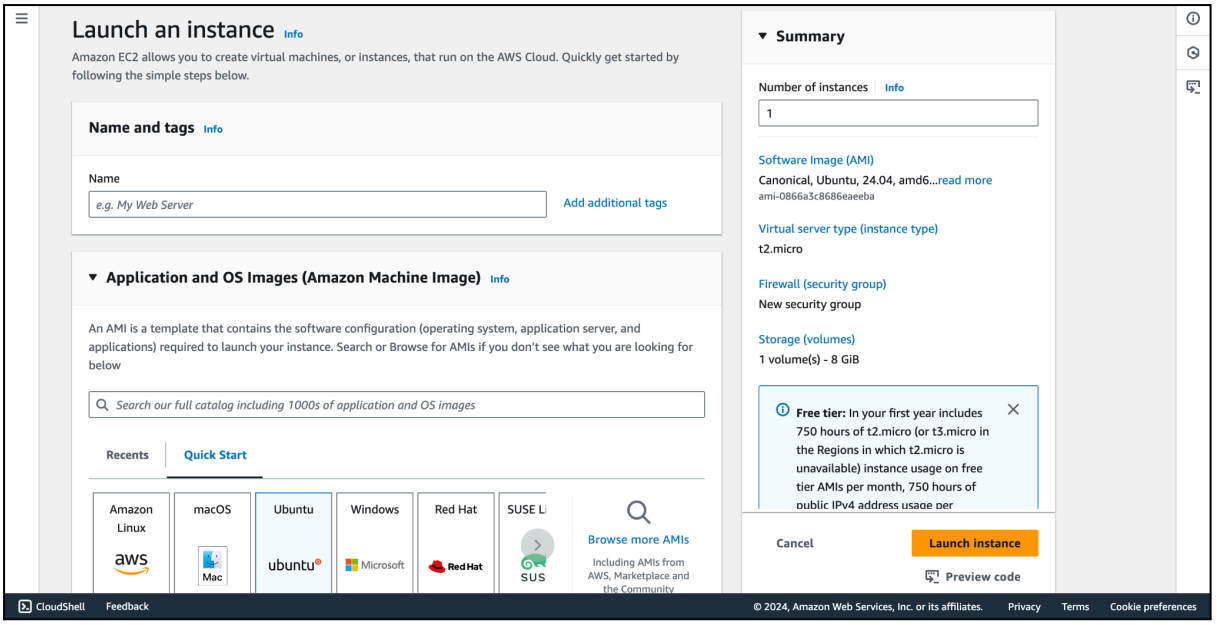
e. After connecting to EC2 instance, install apache server using following Command:

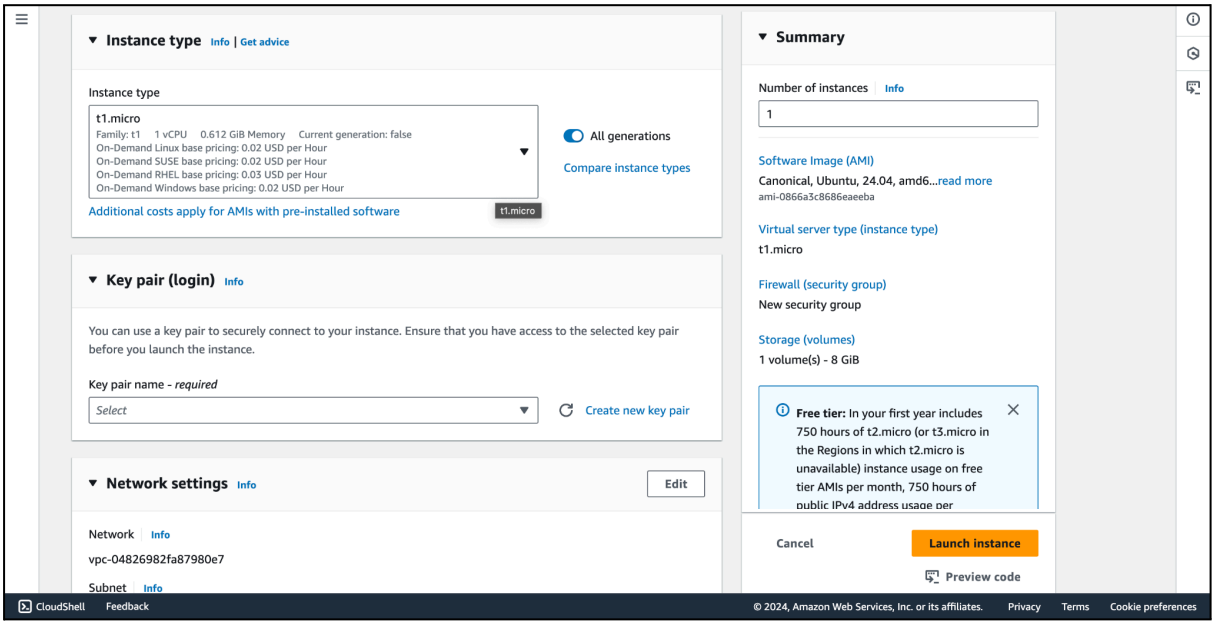
sudo apt-get install apache2

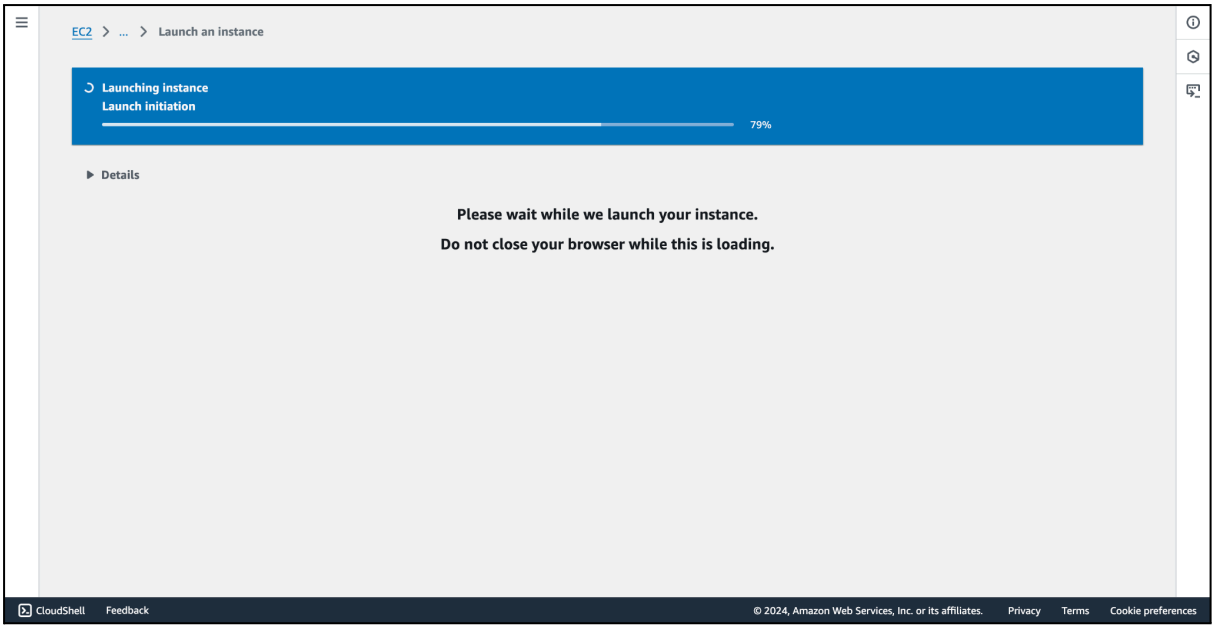
Procedure:

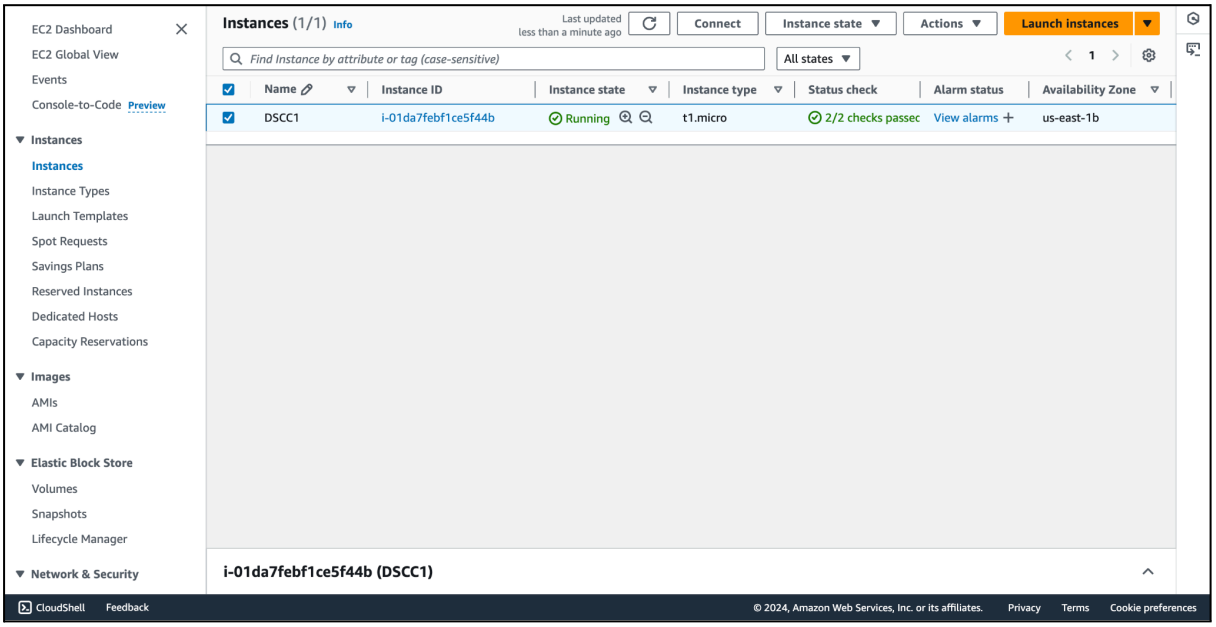


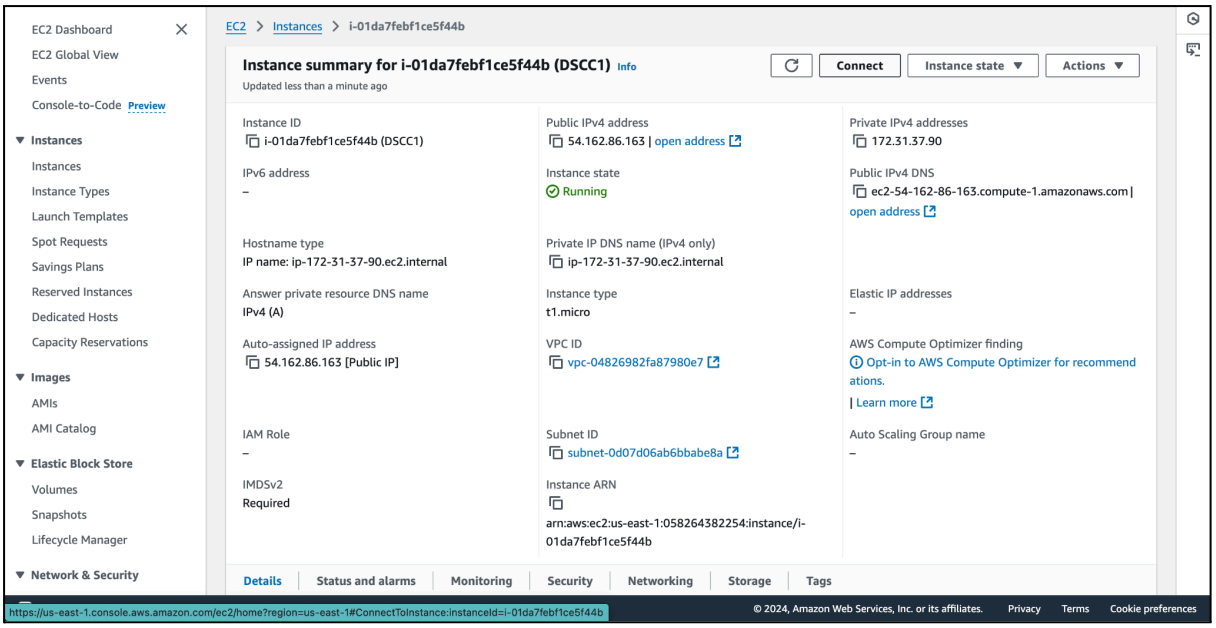

EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Instances Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
No instances
You do not have any instances in this region
Launch instances
Select an instance
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

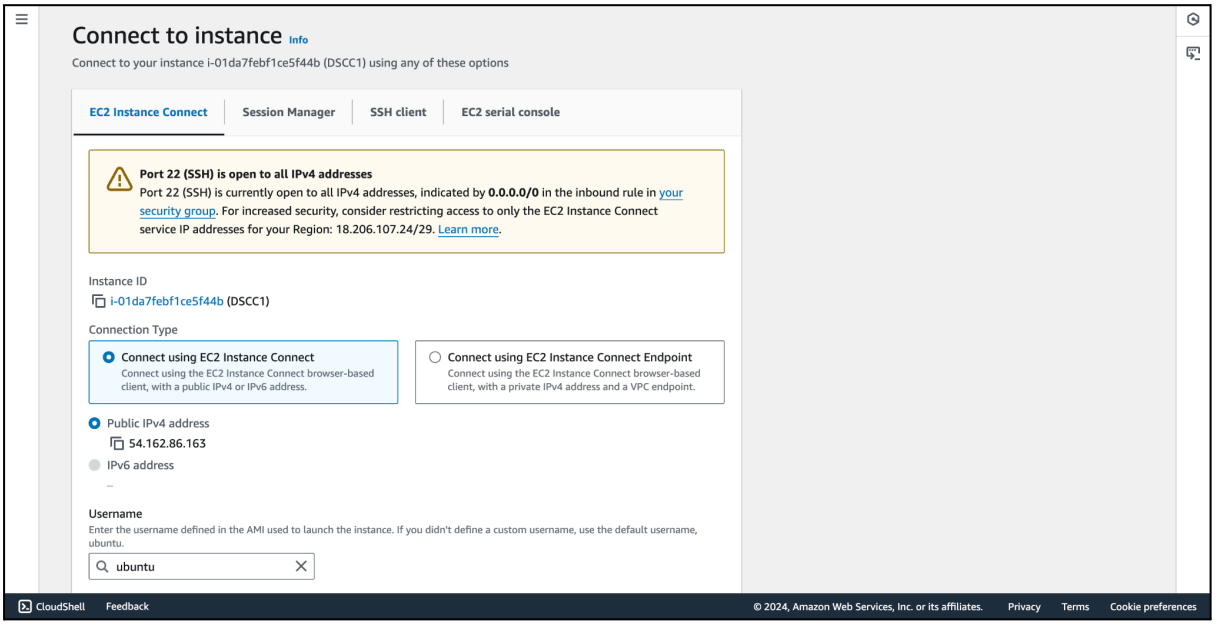

Launch an instance Info
Amazon EC2 allows you to create virtual machines, or instances, that run on the AWS Cloud. Quickly get started by following the simple steps below.
Name and tags Info
Name
e.g. My Web Server
Add additional tags
Application and OS Images (Amazon Machine Image) Info
An AMI is a 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
Recents
Quick Start
Amazon Linux
macOS
Ubuntu
Windows
Red Hat
SUSE Li
Browse more AMIs
Including AMIs from AWS, Marketplace and the Community
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0866a3c8686eaeeba
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 AMIs per month, 750 hours of public IPv4 address usage per
Cancel
Launch instance
Preview code
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

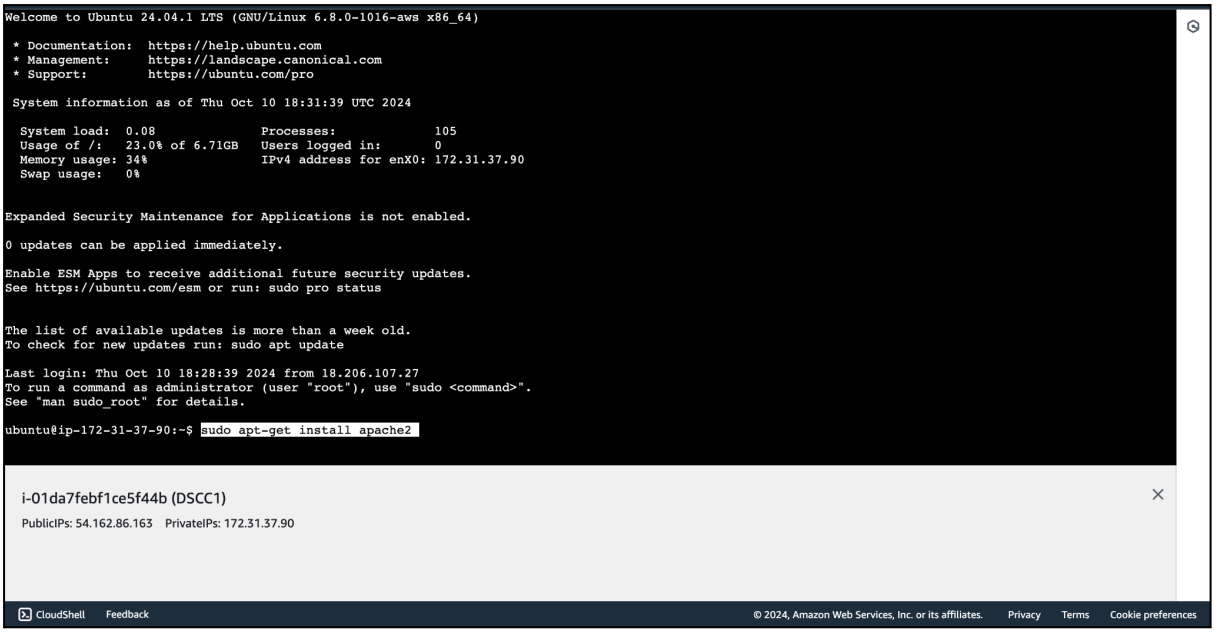

Instance type Info | Get advice
Instance type
t1.micro
Family: t1 1 vCPU 0.612 GiB Memory Current generation: false
On-Demand Linux base pricing: 0.02 USD per Hour
On-Demand SUSE base pricing: 0.02 USD per Hour
On-Demand RHEL base pricing: 0.03 USD per Hour
On-Demand Windows base pricing: 0.02 USD per Hour
All generations
Compare instance types
Additional costs apply for AMIs with pre-installed software
t1.micro
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-04826982fa87980e7
Subnet Info
Summary
Number of instances Info
1
Software Image (AMI)
Canonical, Ubuntu, 24.04, amd6...read more
ami-0866a3c8686eaeeba
Virtual server type (instance type)
t1.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 AMIs per month, 750 hours of public IPv4 address usage per
Cancel
Launch instance
Preview code
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

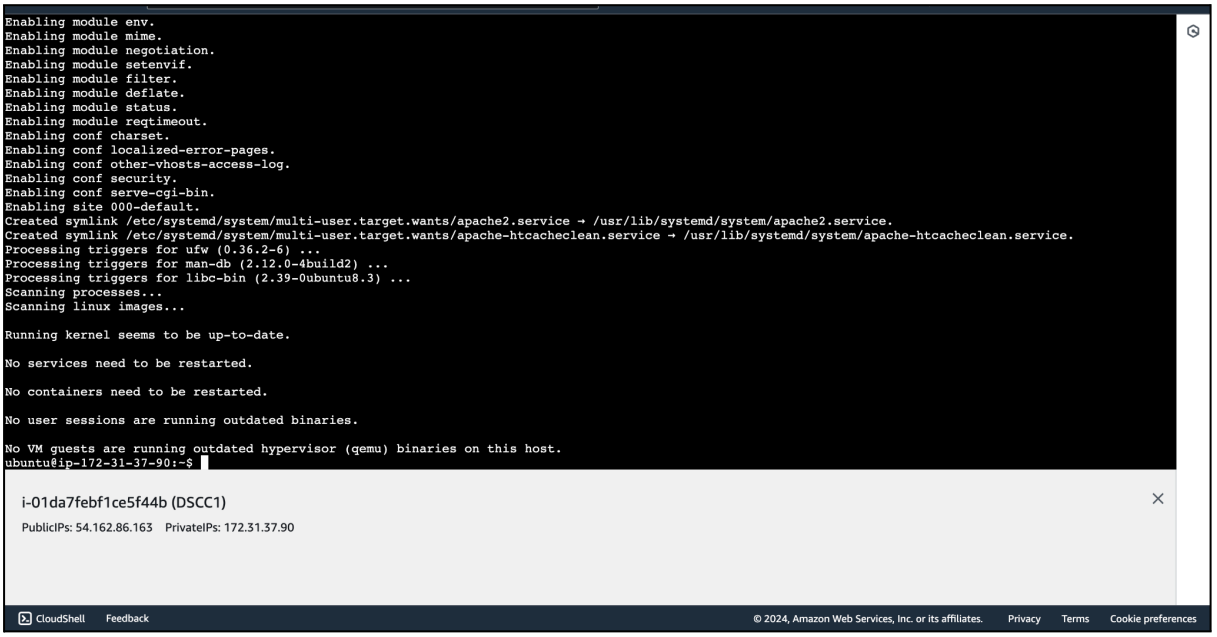

EC2 > ... > Launch an instance
Launching instance
Launch initiation
79%
Details
Please wait while we launch your instance.
Do not close your browser while this is loading.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
Instances (1/1) Info
Last updated less than a minute ago
Connect
Instance state
Actions
Launch instances
Find Instance by attribute or tag (case-sensitive)
All states
Name
Instance ID
Instance state
Instance type
Status check
Alarm status
Availability Zone
DSCC1
i-01da7febf1ce5f44b
Running
t1.micro
2/2 checks passec
View alarms
us-east-1b
i-01da7febf1ce5f44b (DSCC1)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


EC2 Dashboard
EC2 Global View
Events
Console-to-Code Preview
Instances
Instances
Instance Types
Launch Templates
Spot Requests
Savings Plans
Reserved Instances
Dedicated Hosts
Capacity Reservations
Images
AMIs
AMI Catalog
Elastic Block Store
Volumes
Snapshots
Lifecycle Manager
Network & Security
EC2 > Instances > i-01da7febf1ce5f44b
Instance summary for i-01da7febf1ce5f44b (DSCC1) Info
Updated less than a minute ago
Connect
Instance state
Actions
Instance ID
i-01da7febf1ce5f44b (DSCC1)
IPv6 address
–
Hostname type
IP name: ip-172-31-37-90.ec2.internal
Answer private resource DNS name
IPv4 (A)
Auto-assigned IP address
54.162.86.163 [Public IP]
IAM Role
–
IMDSv2
Required
Public IPv4 address
54.162.86.163 | open address
Instance state
Running
Private IP DNS name (IPv4 only)
ip-172-31-37-90.ec2.internal
Instance type
t1.micro
VPC ID
vpc-04826982fa87980e7
Subnet ID
subnet-0d07d06ab6bbabe8a
Instance ARN
arn:aws:ec2:us-east-1:058264382254:instance/i-01da7febf1ce5f44b
Private IPv4 addresses
172.31.37.90
Public IPv4 DNS
ec2-54-162-86-163.compute-1.amazonaws.com | open address
Elastic IP addresses
–
AWS Compute Optimizer finding
Opt-in to AWS Compute Optimizer for recommendations.
| Learn more
Auto Scaling Group name
–
Details
Status and alarms
Monitoring
Security
Networking
Storage
Tags
https://us-east-1.console.aws.amazon.com/ec2/home?region=us-east-1#ConnectToInstance:instanceId=i-01da7febf1ce5f44b
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

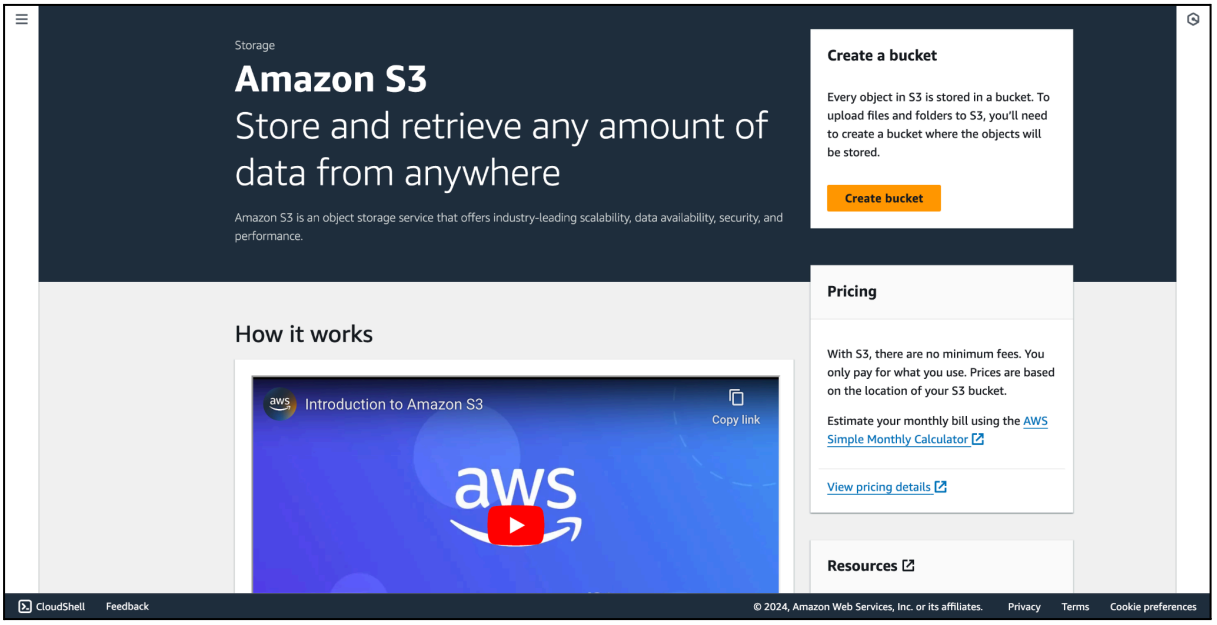

Connect to instance Info
Connect to your instance i-01da7febf1ce5f44b (DSCC1) using any of these options
EC2 Instance Connect
Session Manager
SSH client
EC2 serial console
Port 22 (SSH) is open to all IPv4 addresses
Port 22 (SSH) is currently open to all IPv4 addresses, indicated by 0.0.0.0/0 in the inbound rule in your security group. For increased security, consider restricting access to only the EC2 Instance Connect service IP addresses for your Region: 18.206.107.24/29. Learn more.
Instance ID
i-01da7febf1ce5f44b (DSCC1)
Connection Type
Connect using EC2 Instance Connect
Connect using the EC2 Instance Connect browser-based client, with a public IPv4 or IPv6 address.
Connect using EC2 Instance Connect Endpoint
Connect using the EC2 Instance Connect browser-based client, with a private IPv4 address and a VPC endpoint.
Public IPv4 address
54.162.86.163
IPv6 address
Username
Enter the username defined in the AMI used to launch the instance. If you didn't define a custom username, use the default username, ubuntu.
ubuntu
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

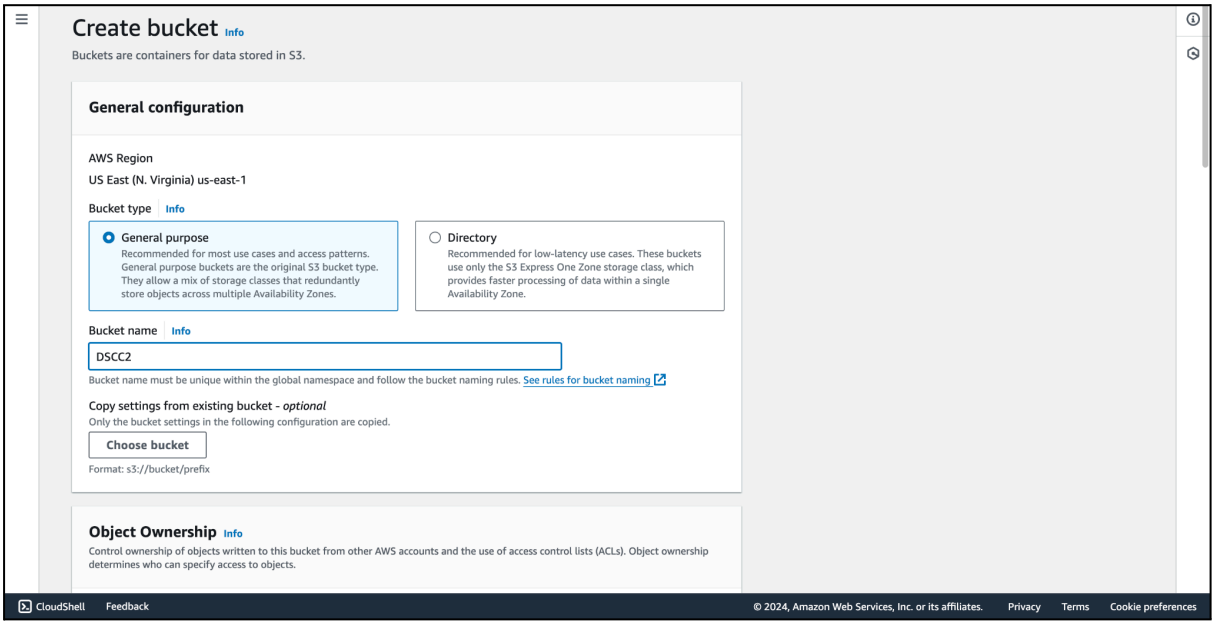

Welcome to Ubuntu 24.04.1 LTS (GNU/Linux 6.8.0-1016-aws x86_64)
* Documentation: https://help.ubuntu.com
* Management: https://landscape.canonical.com
* Support: https://ubuntu.com/pro
System information as of Thu Oct 10 18:31:39 UTC 2024
System load: 0.08
Usage of /: 23.0% of 6.71GB
Memory usage: 34%
Swap usage: 0%
Processes: 105
Users logged in: 0
IPv4 address for enX0: 172.31.37.90
Expanded Security Maintenance for Applications is not enabled.
0 updates can be applied immediately.
Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status
The list of available updates is more than a week old.
To check for new updates run: sudo apt update
Last login: Thu Oct 10 18:28:39 2024 from 18.206.107.27
To run a command as administrator (user "root"), use "sudo <command>".
See "man sudo_root" for details.
ubuntu@ip-172-31-37-90:~$ sudo apt-get install apache2
i-01da7febf1ce5f44b (DSCC1)
PublicIPs: 54.162.86.163 PrivateIPs: 172.31.37.90
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

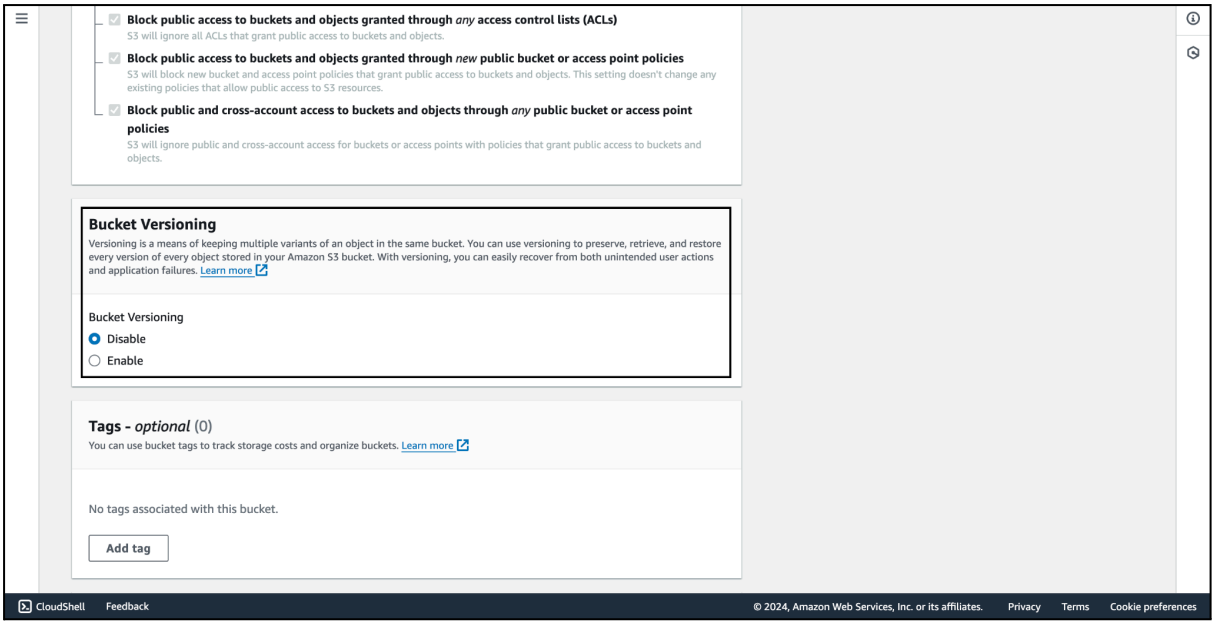

Enabling module env.
Enabling module mime.
Enabling module negotiation.
Enabling module setenvif.
Enabling module filter.
Enabling module deflate.
Enabling module status.
Enabling module reqtimeout.
Enabling conf charset.
Enabling conf localized-error-pages.
Enabling conf other-vhosts-access-log.
Enabling conf security.
Enabling conf serve-cgi-bin.
Enabling site 000-default.
Created symlink /etc/systemd/system/multi-user.target.wants/apache2.service → /usr/lib/systemd/system/apache2.service.
Created symlink /etc/systemd/system/multi-user.target.wants/apache-htcacheclean.service → /usr/lib/systemd/system/apache-htcacheclean.service.
Processing triggers for ufw (0.36.2-6) ...
Processing triggers for man-db (2.12.0-4build2) ...
Processing triggers for libc-bin (2.39-0ubuntu8.3) ...
Scanning processes...
Scanning linux images...
Running kernel seems to be up-to-date.
No services need to be restarted.
No containers need to be restarted.
No user sessions are running outdated binaries.
No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@ip-172-31-37-90:~$
i-01da7febf1ce5f44b (DSCC1)
PublicIPs: 54.162.86.163 PrivateIPs: 172.31.37.90
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

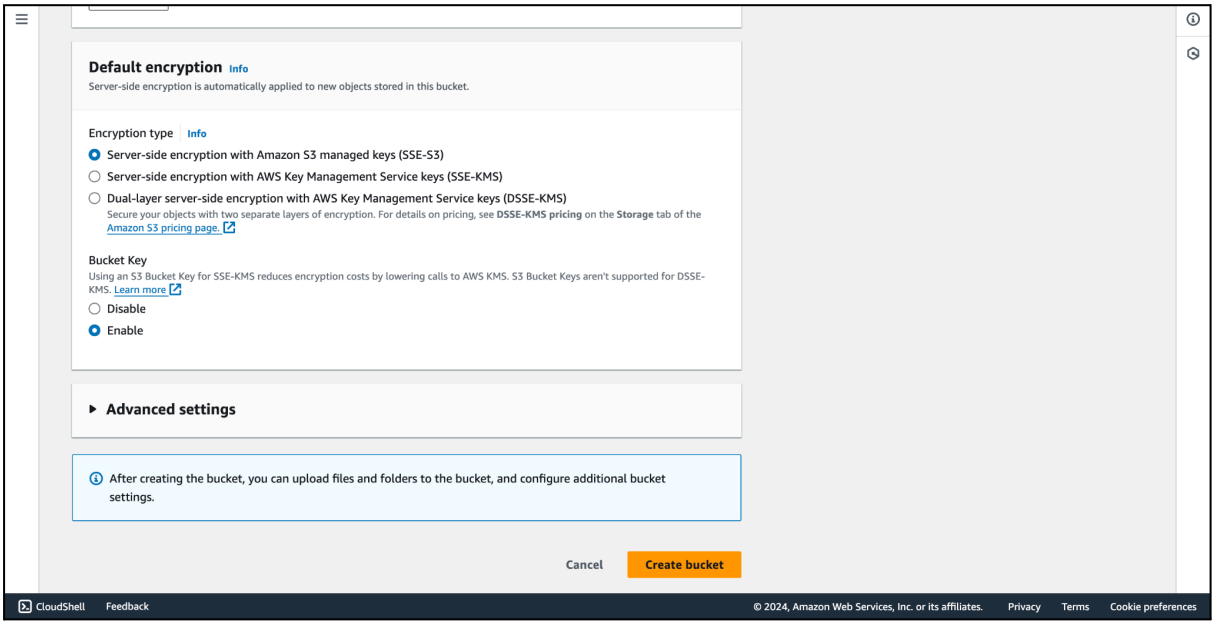
2. **Amazon S3**
    a. Open Amazon S3 console, create a new bucket and upload any two files from PC to bucket (disable Versioning)
    b. Create another bucket with unique name and create folder in this bucket
    c. Upload similar files (mentioned in step 1) to this folder
    d. Copy one file from first bucket to new bucket (step 2)
    e. Show all the versions of files in new folder
    f. Delete old version of the file and check if it is visible again
    g. Empty the first folder then delete it.

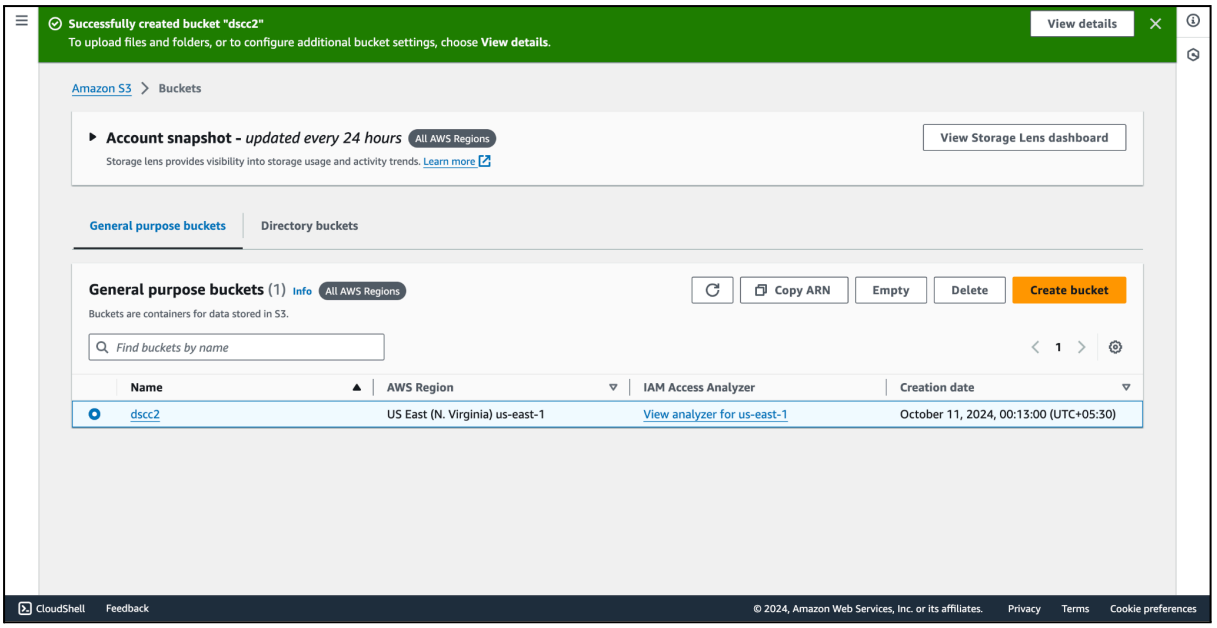
Procedure:

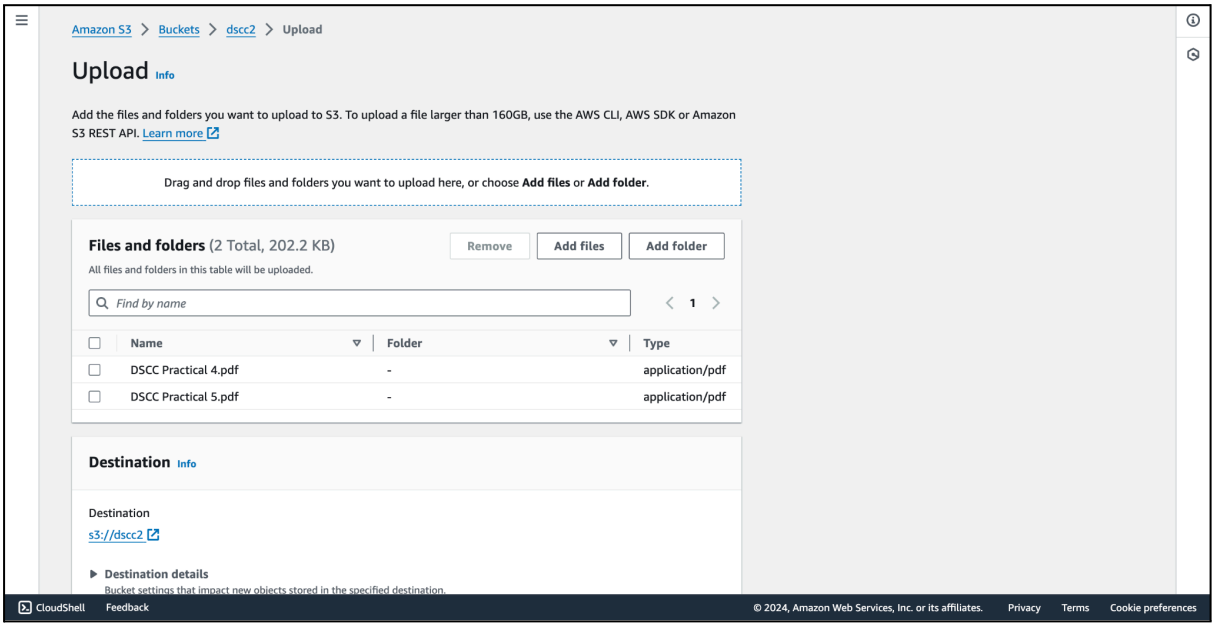


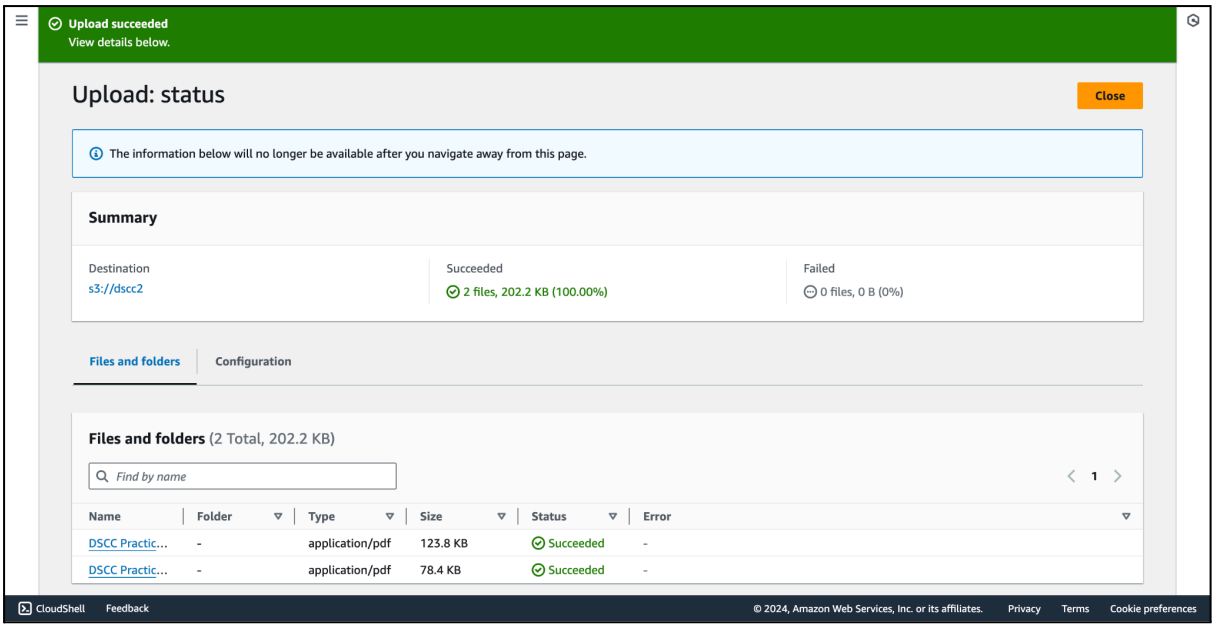

Create bucket Info
Buckets are containers for data stored in S3.
General configuration
AWS Region
US East (N. Virginia) us-east-1
Bucket type Info
General purpose
Recommended for most use cases and access patterns. General purpose buckets are the original S3 bucket type. They allow a mix of storage classes that redundantly store objects across multiple Availability Zones.
Directory
Recommended for low-latency use cases. These buckets use only the S3 Express One Zone storage class, which provides faster processing of data within a single Availability Zone.
Bucket name Info
DSCC2
Bucket name must be unique within the global namespace and follow the bucket naming rules. See rules for bucket naming
Copy settings from existing bucket - optional
Only the bucket settings in the following configuration are copied.
Choose bucket
Format: s3://bucket/prefix
Object Ownership Info
Control ownership of objects written to this bucket from other AWS accounts and the use of access control lists (ACLs). Object ownership determines who can specify access to objects.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

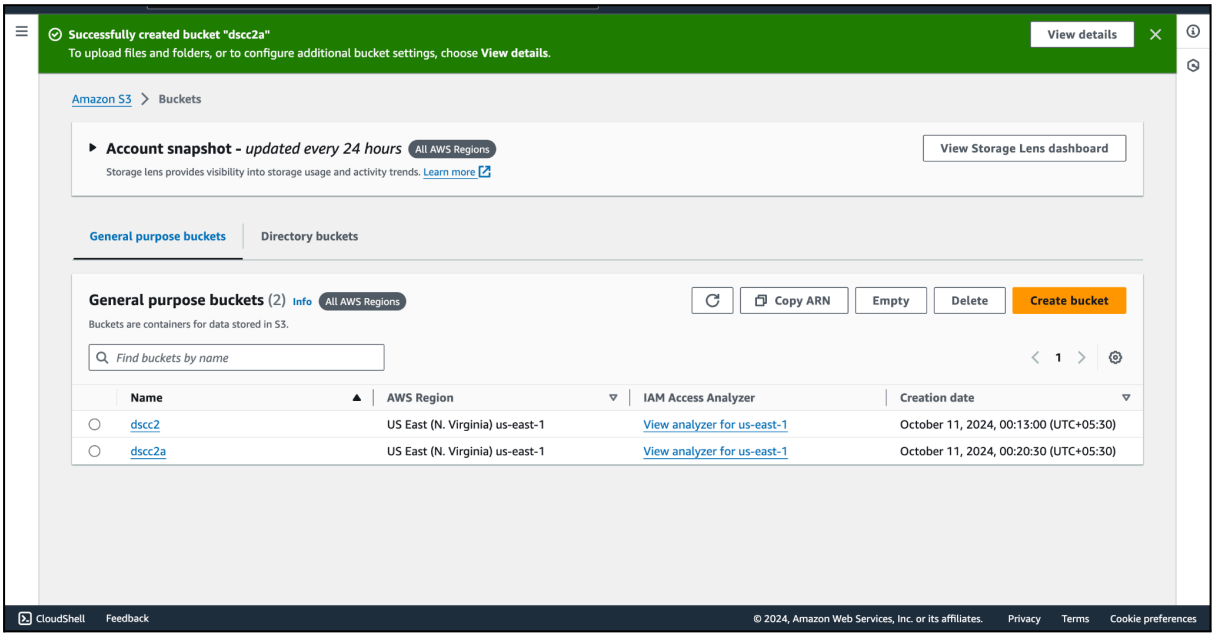

Block public access to buckets and objects granted through any access control lists (ACLs)
S3 will ignore all ACLs that grant public access to buckets and objects.
Block public access to buckets and objects granted through new public bucket or access point policies
S3 will block new bucket and access point policies that grant public access to buckets and objects. This setting doesn't change any existing policies that allow public access to S3 resources.
Block public and cross-account access to buckets and objects through any public bucket or access point policies
S3 will ignore public and cross-account access for buckets or access points with policies that grant public access to buckets and objects.
Bucket Versioning
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more
Bucket Versioning
Disable
Enable
Tags - optional (0)
You can use bucket tags to track storage costs and organize buckets. Learn more
No tags associated with this bucket.
Add tag
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

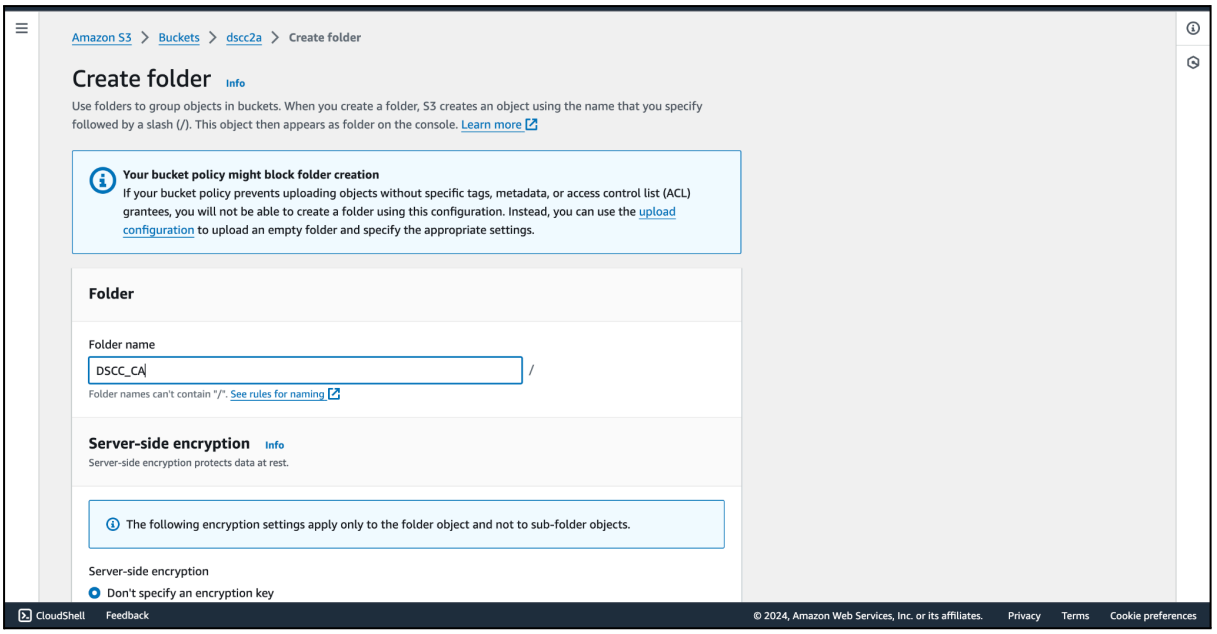

Default encryption Info
Server-side encryption is automatically applied to new objects stored in this bucket.
Encryption type Info
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Server-side encryption with AWS Key Management Service keys (SSE-KMS)
Dual-layer server-side encryption with AWS Key Management Service keys (DSSE-KMS)
Secure your objects with two separate layers of encryption. For details on pricing, see DSSE-KMS pricing on the Storage tab of the Amazon S3 pricing page.
Bucket Key
Using an S3 Bucket Key for SSE-KMS reduces encryption costs by lowering calls to AWS KMS. S3 Bucket Keys aren't supported for DSSE-KMS. Learn more
Disable
Enable
Advanced settings
After creating the bucket, you can upload files and folders to the bucket, and configure additional bucket settings.
Cancel
Create bucket
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

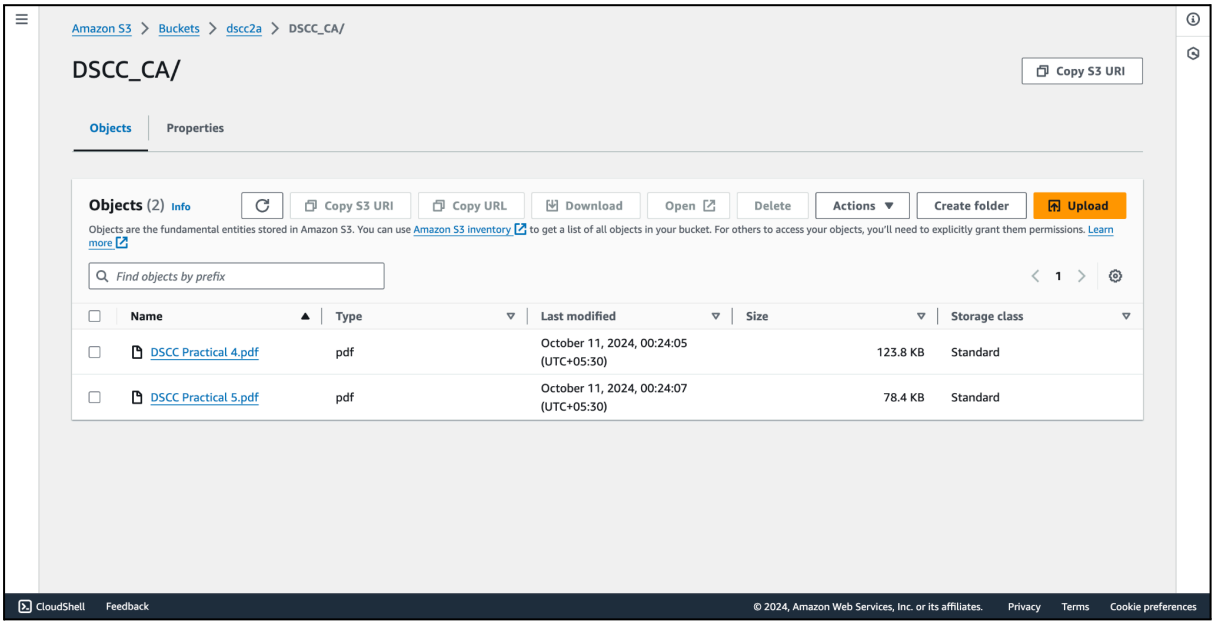

Successfully created bucket "dscc2"
To upload files and folders, or to configure additional bucket settings, choose View details.
View details
Amazon S3 > Buckets
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (1) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
dscc2
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 11, 2024, 00:13:00 (UTC+05:30)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

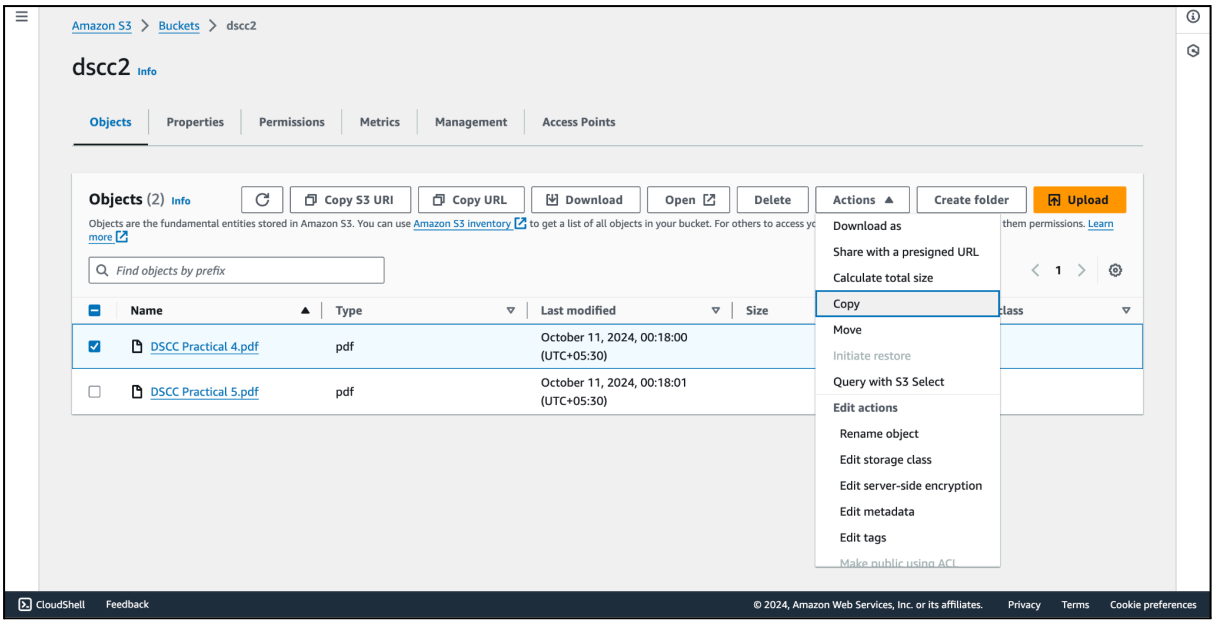

Amazon S3 > Buckets > dscc2 > Upload
Upload Info
Add the files and folders you want to upload to S3. To upload a file larger than 160GB, use the AWS CLI, AWS SDK or Amazon S3 REST API. Learn more
Drag and drop files and folders you want to upload here, or choose Add files or Add folder.
Files and folders (2 Total, 202.2 KB)
Remove
Add files
Add folder
All files and folders in this table will be uploaded.
Find by name
1
Name
Folder
Type
DSCC Practical 4.pdf
-
application/pdf
DSCC Practical 5.pdf
-
application/pdf
Destination Info
Destination
s3://dscc2
Destination details
Bucket settings that impact new objects stored in the specified destination.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

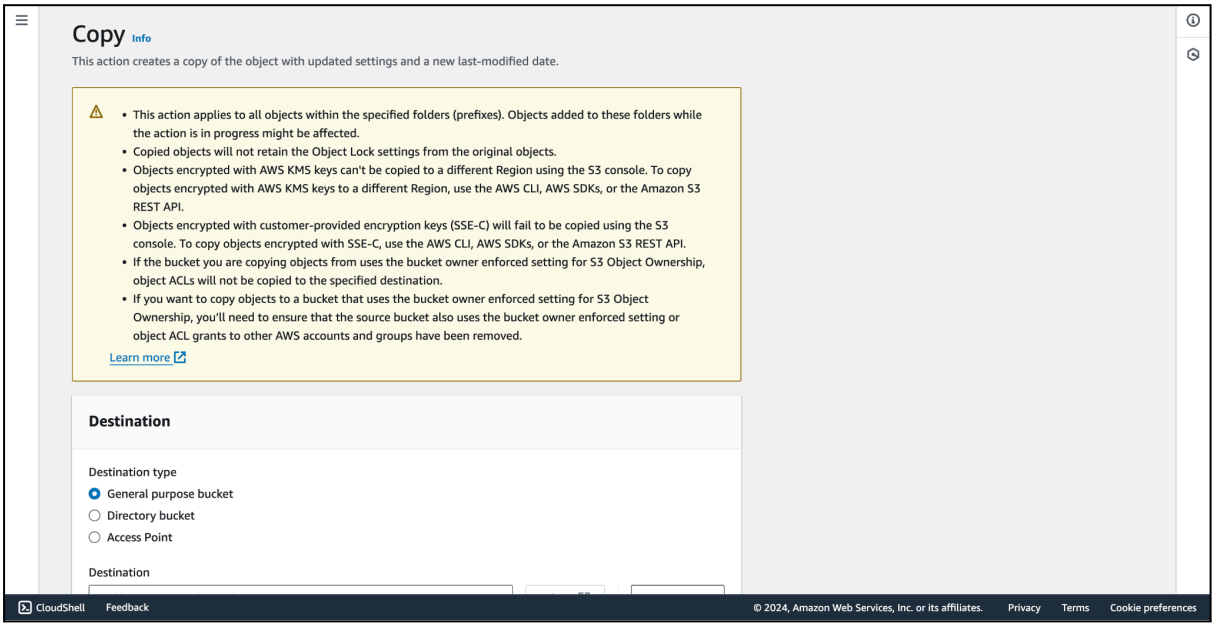

Upload succeeded
View details below.
Upload: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Destination
s3://dscc2
Succeeded
2 files, 202.2 KB (100.00%)
Failed
0 files, 0 B (0%)
Files and folders
Configuration
Files and folders (2 Total, 202.2 KB)
Find by name
1
Name
Folder
Type
Size
Status
Error
DSCC Practic...
-
application/pdf
123.8 KB
Succeeded
-
DSCC Practic...
-
application/pdf
78.4 KB
Succeeded
-
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

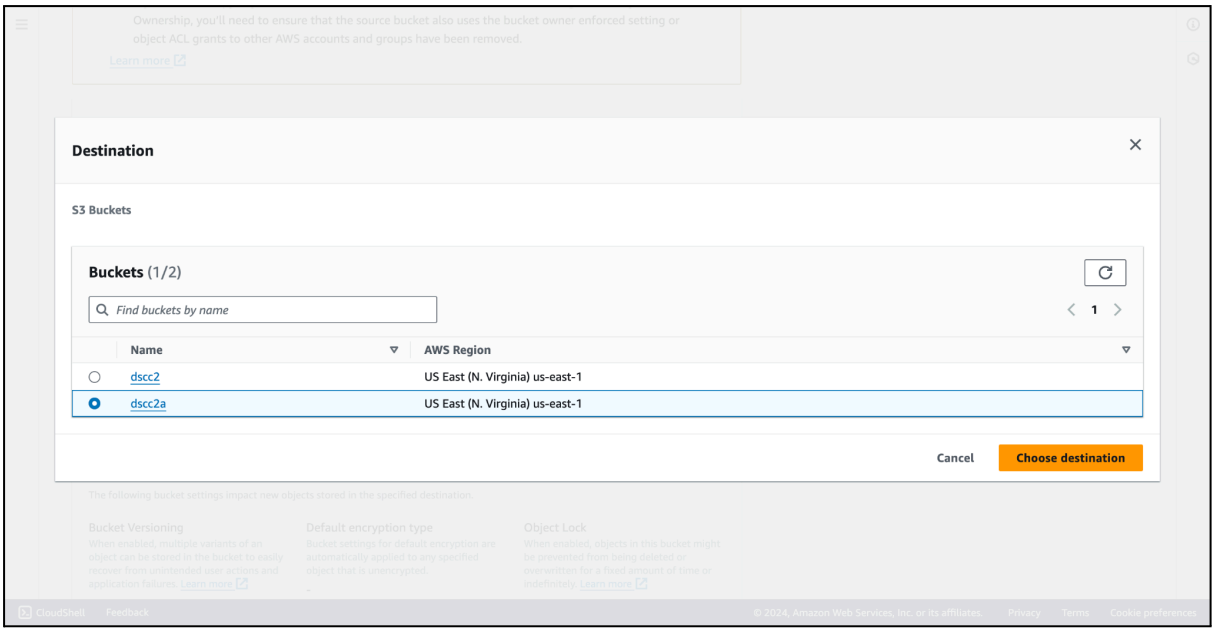

Successfully created bucket "dscc2a"
To upload files and folders, or to configure additional bucket settings, choose View details.
View details
Amazon S3 > Buckets
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (2) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
dscc2
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 11, 2024, 00:13:00 (UTC+05:30)
dscc2a
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 11, 2024, 00:20:30 (UTC+05:30)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

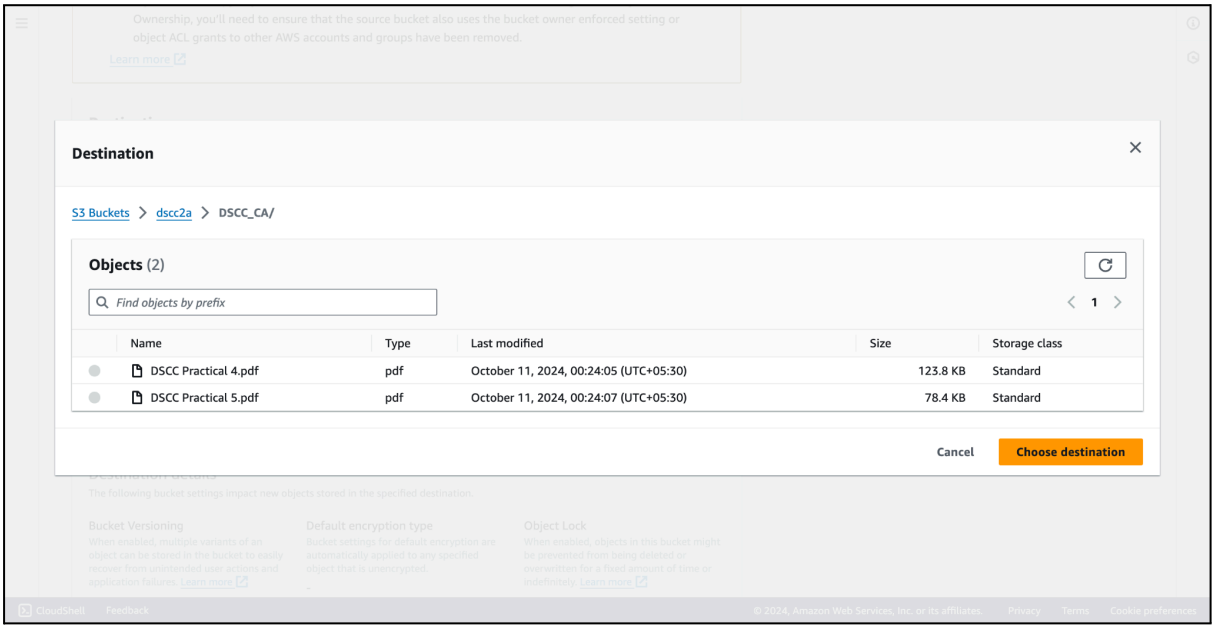

Amazon S3 > Buckets > dscc2a > Create folder
Create folder Info
Use folders to group objects in buckets. When you create a folder, S3 creates an object using the name that you specify followed by a slash (/). This object then appears as folder on the console. Learn more
Your bucket policy might block folder creation
If your bucket policy prevents uploading objects without specific tags, metadata, or access control list (ACL) grantees, you will not be able to create a folder using this configuration. Instead, you can use the upload configuration to upload an empty folder and specify the appropriate settings.
Folder
Folder name
DSCC_CA
/
Folder names can't contain "/". See rules for naming
Server-side encryption Info
Server-side encryption protects data at rest.
The following encryption settings apply only to the folder object and not to sub-folder objects.
Server-side encryption
Don't specify an encryption key
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

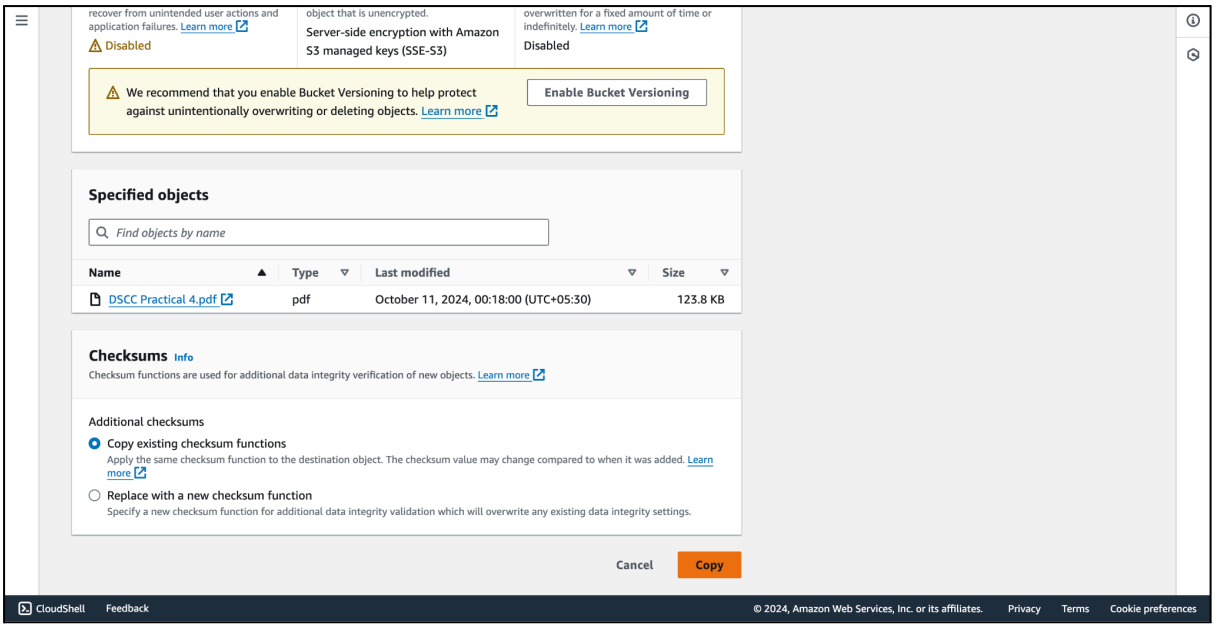

Amazon S3 > Buckets > dscc2a > DSCC_CA/
DSCC_CA/
Copy S3 URI
Objects
Properties
Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Name
Type
Last modified
Size
Storage class
DSCC Practical 4.pdf
pdf
October 11, 2024, 00:24:05 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 5.pdf
pdf
October 11, 2024, 00:24:07 (UTC+05:30)
78.4 KB
Standard
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

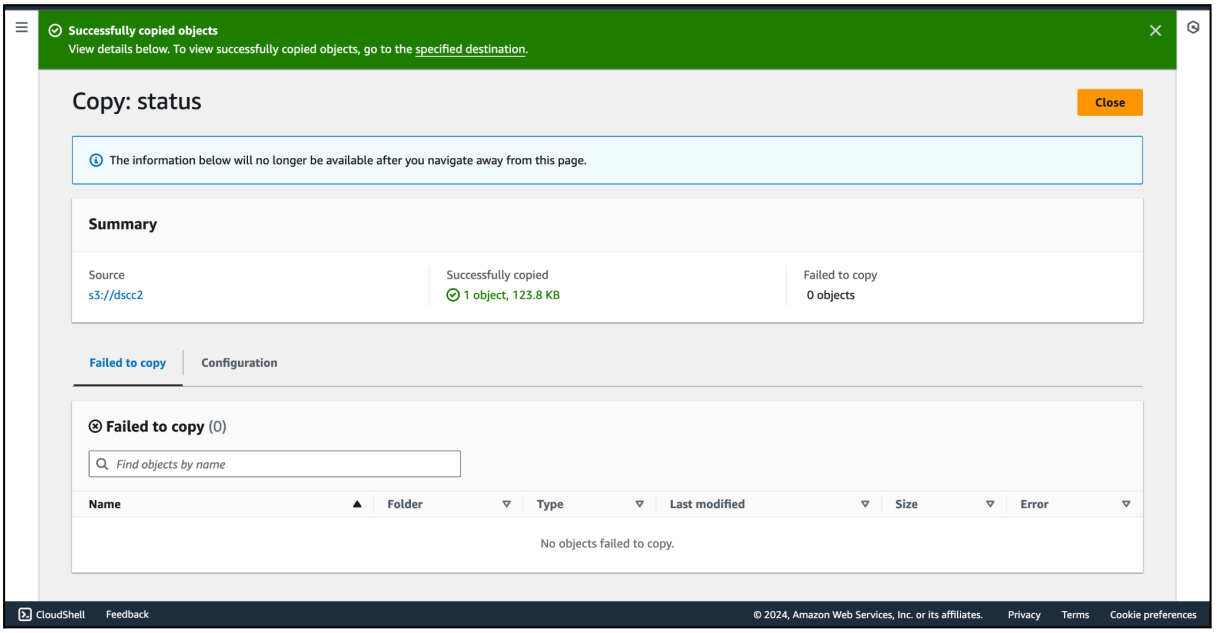

Amazon S3 > Buckets > dscc2
dscc2 Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Download as
Share with a presigned URL
Calculate total size
Copy
Move
Initiate restore
Query with S3 Select
Edit actions
Rename object
Edit storage class
Edit server-side encryption
Edit metadata
Edit tags
Find objects by prefix
Name
Type
Last modified
Size
DSCC Practical 4.pdf
pdf
October 11, 2024, 00:18:00 (UTC+05:30)
DSCC Practical 5.pdf
pdf
October 11, 2024, 00:18:01 (UTC+05:30)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

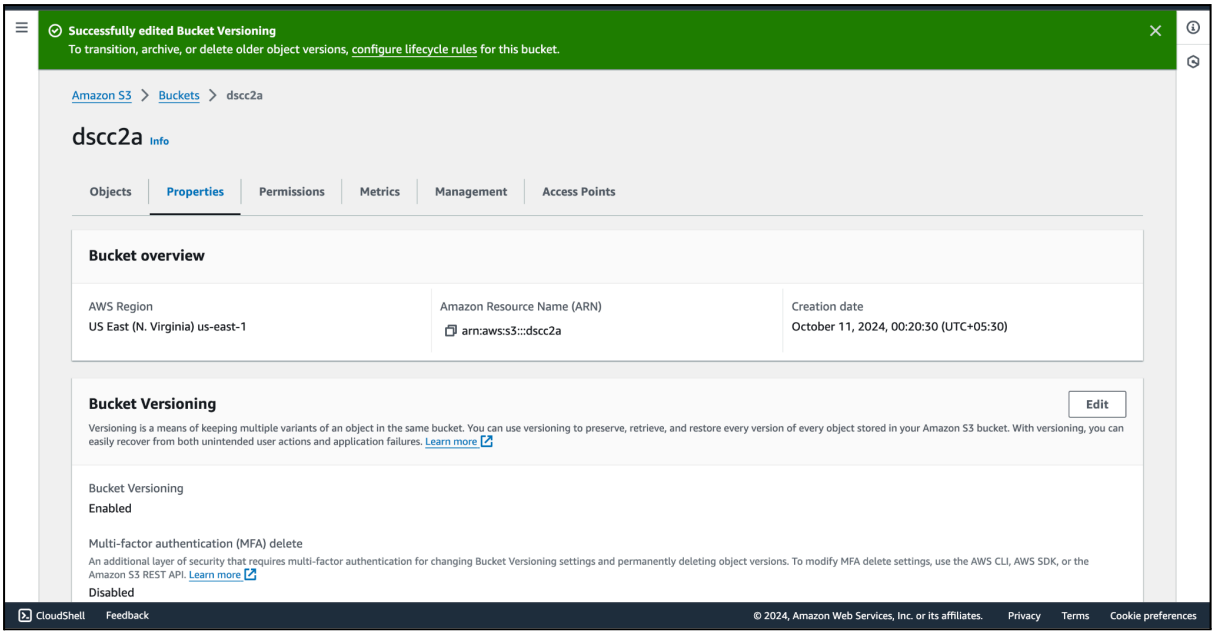

Copy Info
This action creates a copy of the object with updated settings and a new last-modified date.
• This action applies to all objects within the specified folders (prefixes). Objects added to these folders while the action is in progress might be affected.
• Copied objects will not retain the Object Lock settings from the original objects.
• Objects encrypted with AWS KMS keys can't be copied to a different Region using the S3 console. To copy objects encrypted with AWS KMS keys to a different Region, use the AWS CLI, AWS SDKs, or the Amazon S3 REST API.
• Objects encrypted with customer-provided encryption keys (SSE-C) will fail to be copied using the S3 console. To copy objects encrypted with SSE-C, use the AWS CLI, AWS SDKs, or the Amazon S3 REST API.
• If the bucket you are copying objects from uses the bucket owner enforced setting for S3 Object Ownership, object ACLs will not be copied to the specified destination.
• If you want to copy objects to a bucket that uses the bucket owner enforced setting for S3 Object Ownership, you'll need to ensure that the source bucket also uses the bucket owner enforced setting or object ACL grants to other AWS accounts and groups have been removed.
Learn more
Destination
Destination type
General purpose bucket
Directory bucket
Access Point
Destination
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

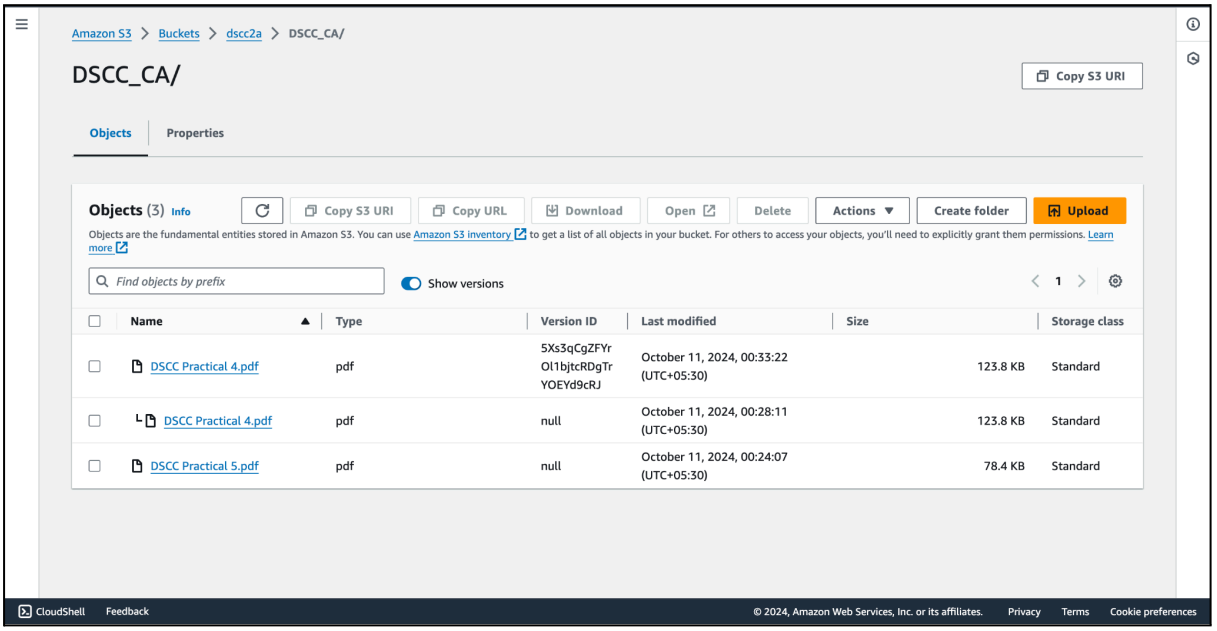

Destination
S3 Buckets
Buckets (1/2)
Find buckets by name
Name
AWS Region
dscc2
US East (N. Virginia) us-east-1
dscc2a
US East (N. Virginia) us-east-1
Cancel
Choose destination

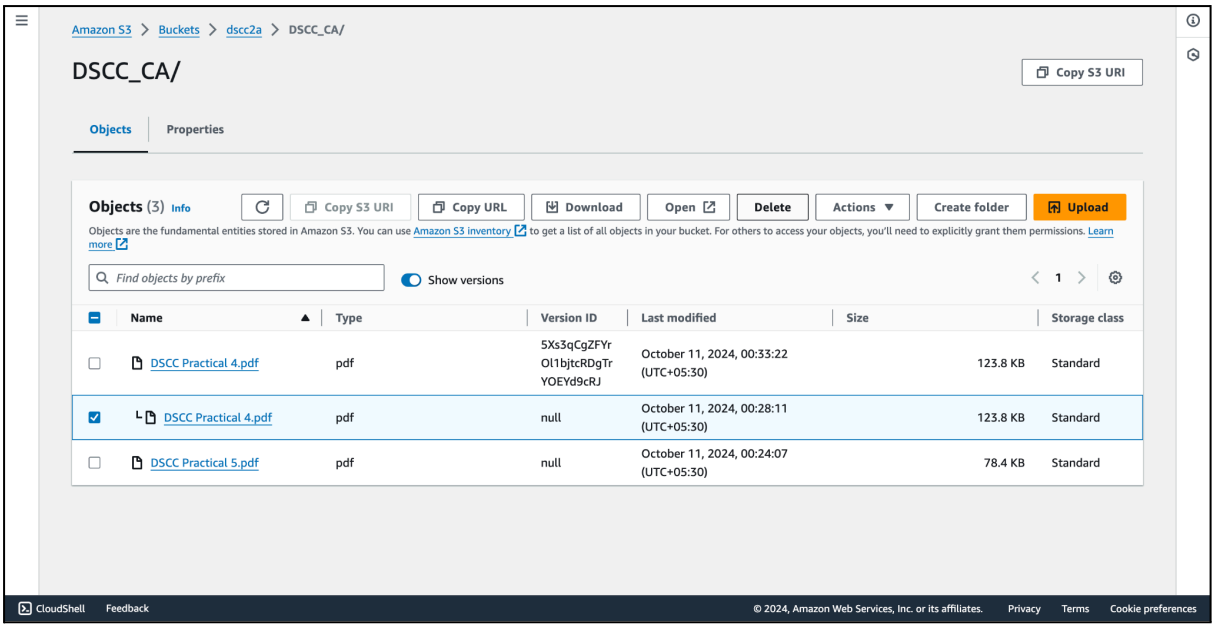

Destination
S3 Buckets > dscc2a > DSCC_CA/
Objects (2)
Find objects by prefix
Name
Type
Last modified
Size
Storage class
DSCC Practical 4.pdf
pdf
October 11, 2024, 00:24:05 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 5.pdf
pdf
October 11, 2024, 00:24:07 (UTC+05:30)
78.4 KB
Standard
Cancel
Choose destination

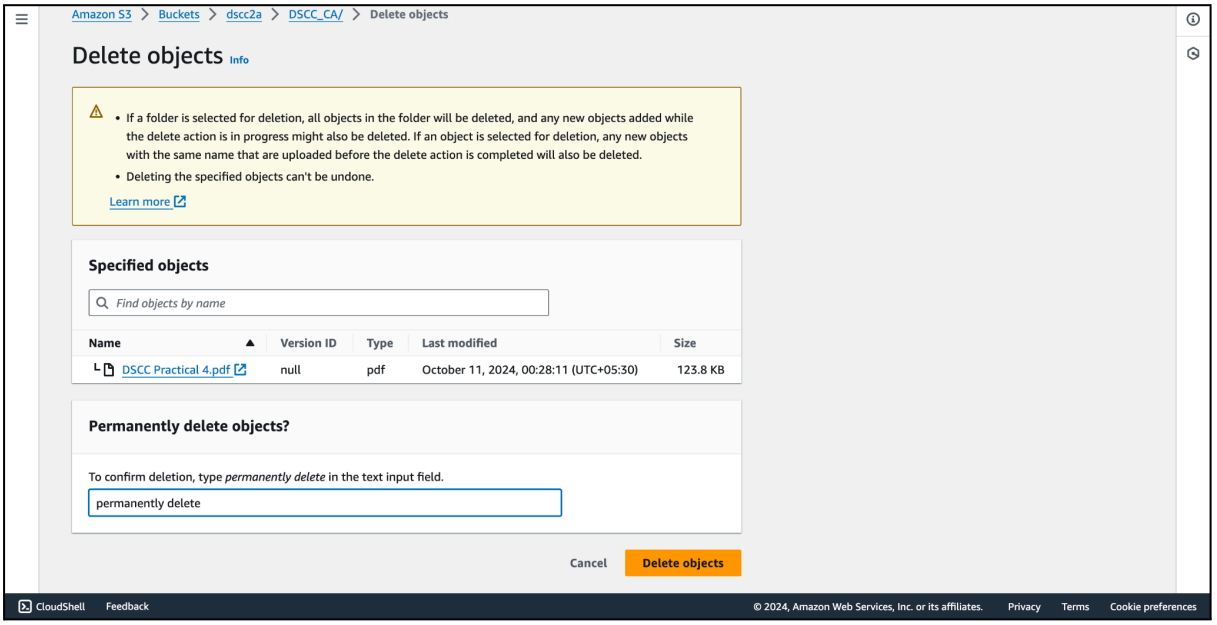

Disabled
Server-side encryption with Amazon S3 managed keys (SSE-S3)
Disabled
We recommend that you enable Bucket Versioning to help protect against unintentionally overwriting or deleting objects. Learn more
Enable Bucket Versioning
Specified objects
Find objects by name
Name
Type
Last modified
Size
DSCC Practical 4.pdf
pdf
October 11, 2024, 00:18:00 (UTC+05:30)
123.8 KB
Checksums Info
Checksum functions are used for additional data integrity verification of new objects. Learn more
Additional checksums
Copy existing checksum functions
Apply the same checksum function to the destination object. The checksum value may change compared to when it was added. Learn more
Replace with a new checksum function
Specify a new checksum function for additional data integrity validation which will overwrite any existing data integrity settings.
Cancel
Copy
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

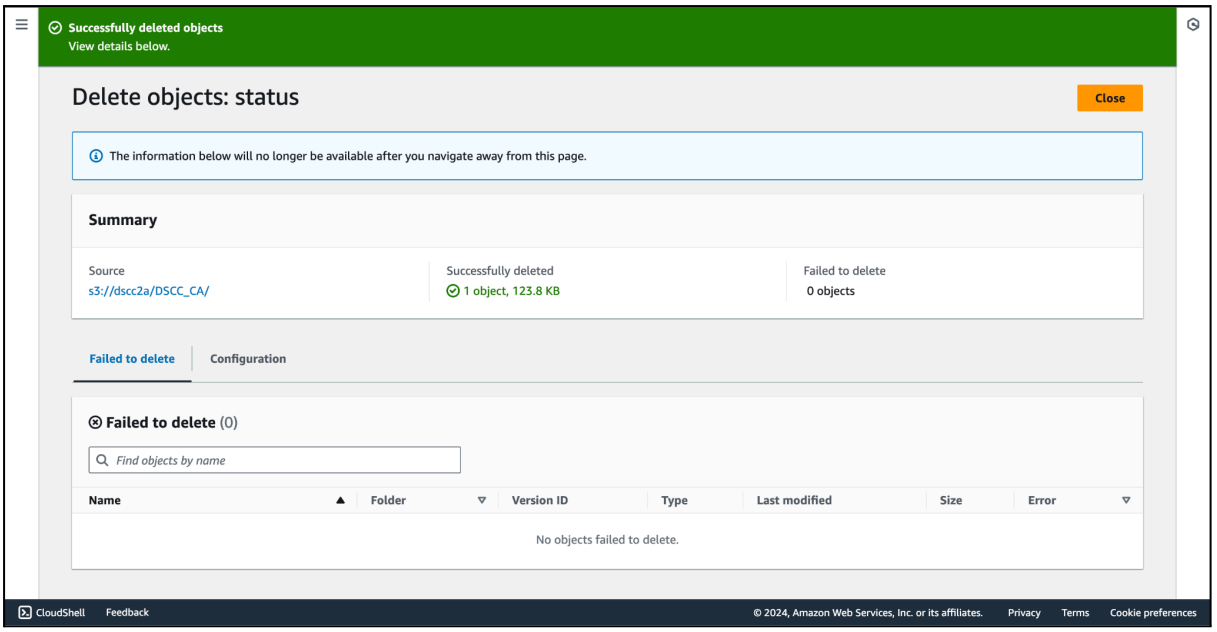

Successfully copied objects
View details below. To view successfully copied objects, go to the specified destination.
Copy: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Source
s3://dscc2
Successfully copied
1 object, 123.8 KB
Failed to copy
0 objects
Failed to copy
Configuration
Failed to copy (0)
Find objects by name
Name
Folder
Type
Last modified
Size
Error
No objects failed to copy.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

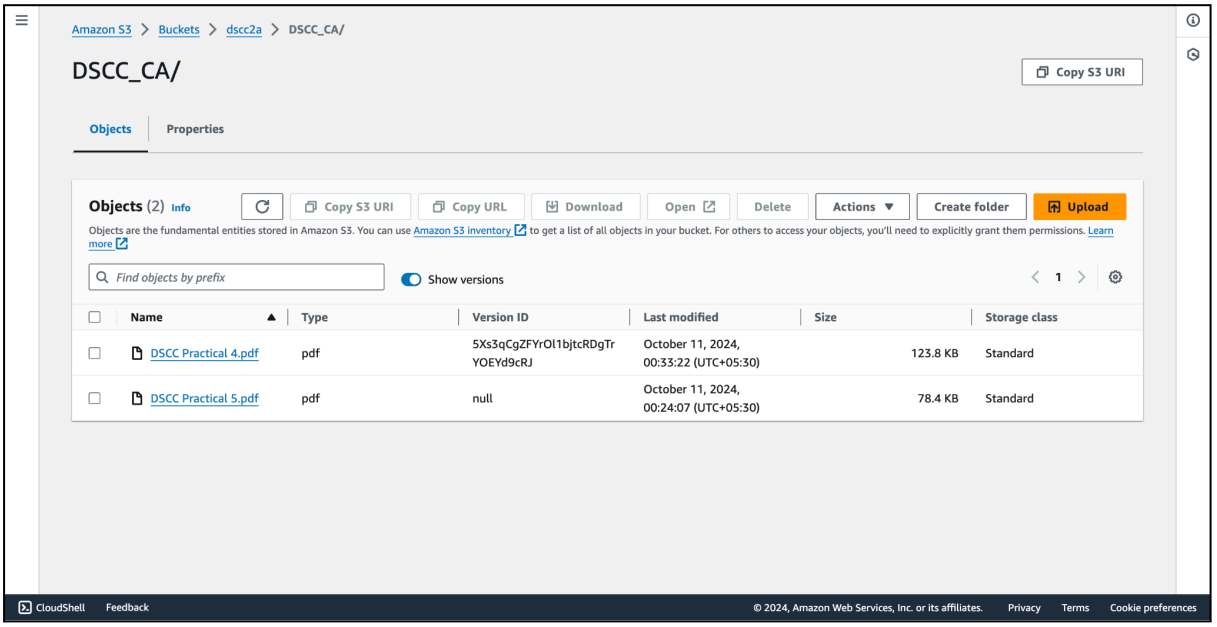

Successfully edited Bucket Versioning
To transition, archive, or delete older object versions, configure lifecycle rules for this bucket.
Amazon S3
Buckets
dscc2a
dscc2a
Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Bucket overview
AWS Region
US East (N. Virginia) us-east-1
Amazon Resource Name (ARN)
arn:aws:s3:::dscc2a
Creation date
October 11, 2024, 00:20:30 (UTC+05:30)
Bucket Versioning
Edit
Versioning is a means of keeping multiple variants of an object in the same bucket. You can use versioning to preserve, retrieve, and restore every version of every object stored in your Amazon S3 bucket. With versioning, you can easily recover from both unintended user actions and application failures. Learn more
Bucket Versioning
Enabled
Multi-factor authentication (MFA) delete
An additional layer of security that requires multi-factor authentication for changing Bucket Versioning settings and permanently deleting object versions. To modify MFA delete settings, use the AWS CLI, AWS SDK, or the Amazon S3 REST API. Learn more
Disabled
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

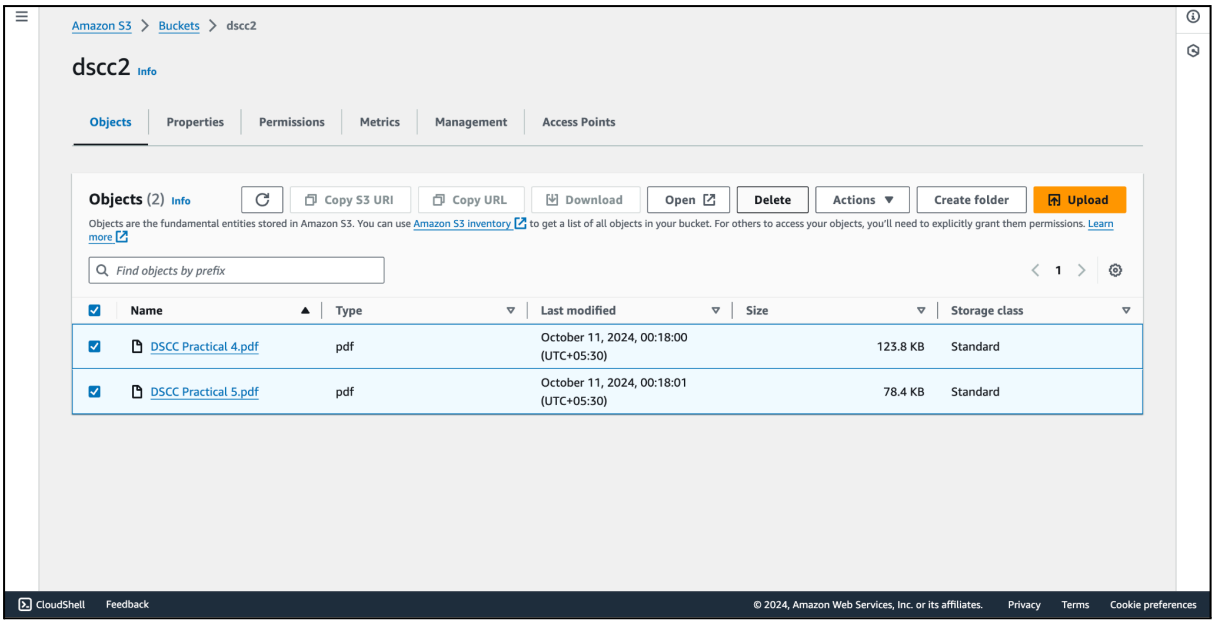

Amazon S3 > Buckets > dscc2a > DSCC_CA/
DSCC_CA/
Copy S3 URI
Objects
Properties
Objects (3) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
1
Name
Type
Version ID
Last modified
Size
Storage class
DSCC Practical 4.pdf
pdf
5Xs3qCgZFYrOl1bjtcRDgTrYOEYd9cRJ
October 11, 2024, 00:33:22 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 4.pdf
pdf
null
October 11, 2024, 00:28:11 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 5.pdf
pdf
null
October 11, 2024, 00:24:07 (UTC+05:30)
78.4 KB
Standard
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

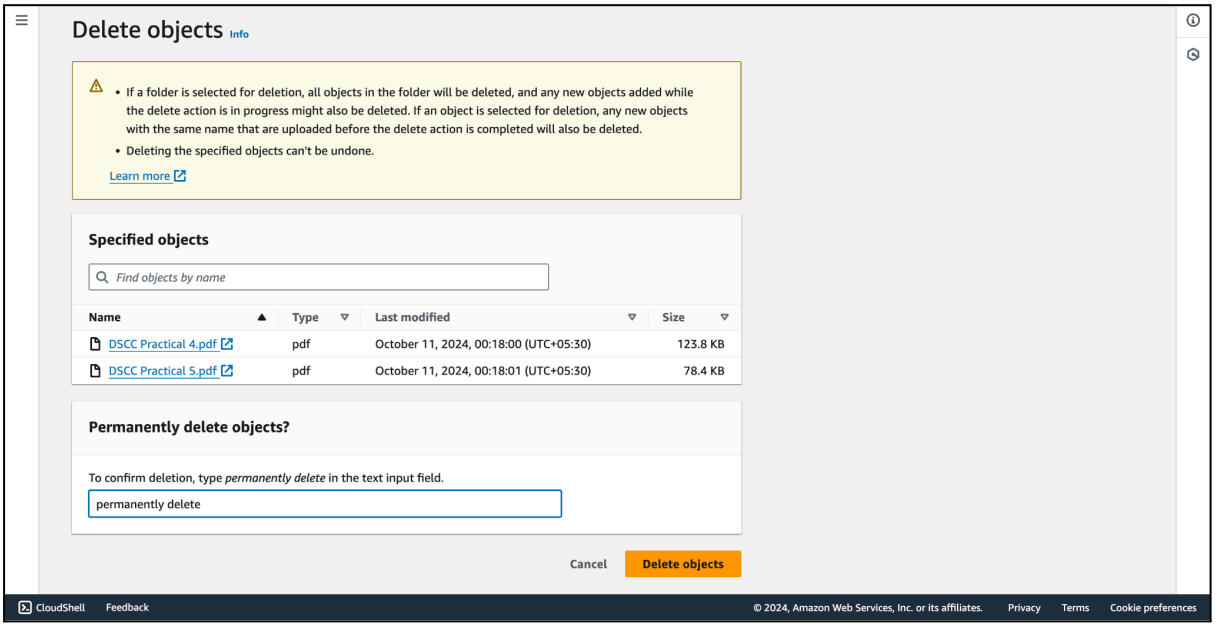

Amazon S3 > Buckets > dscc2a > DSCC_CA/
DSCC_CA/
Copy S3 URI
Objects
Properties
Objects (3) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
1
Name
Type
Version ID
Last modified
Size
Storage class
DSCC Practical 4.pdf
pdf
5Xs3qCgZFYrOl1bjtcRDgTrYOEYd9cRJ
October 11, 2024, 00:33:22 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 4.pdf
pdf
null
October 11, 2024, 00:28:11 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 5.pdf
pdf
null
October 11, 2024, 00:24:07 (UTC+05:30)
78.4 KB
Standard
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

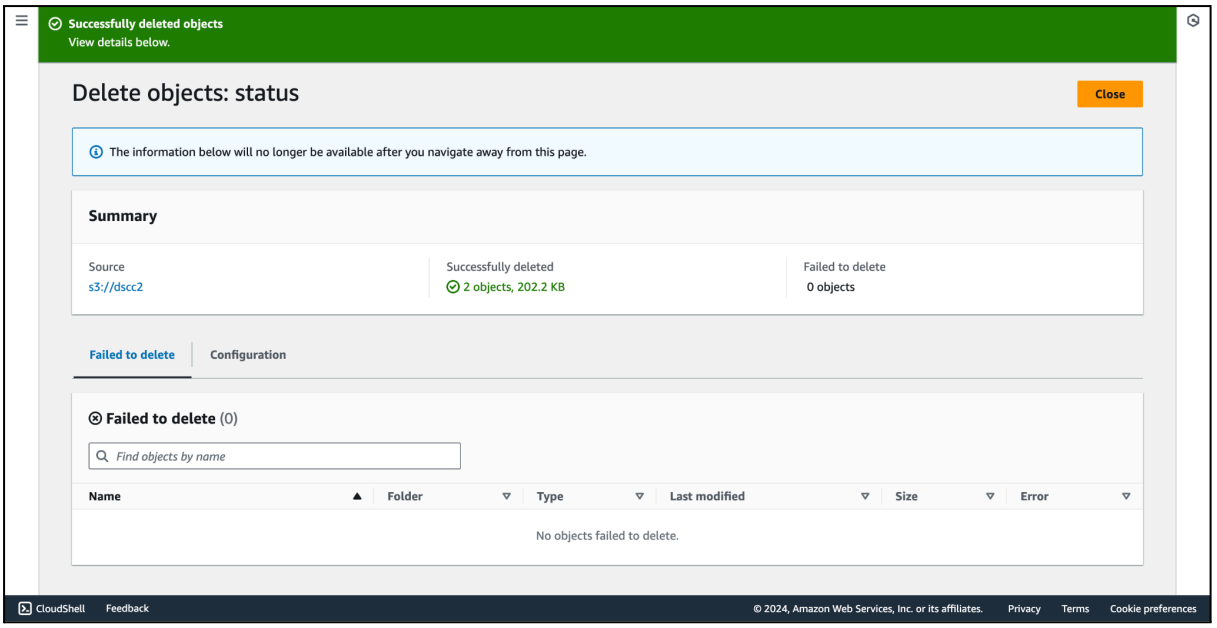

Amazon S3 > Buckets > dscc2a > DSCC_CA/ > Delete objects
Delete objects Info
If a folder is selected for deletion, all objects in the folder will be deleted, and any new objects added while the delete action is in progress might also be deleted. If an object is selected for deletion, any new objects with the same name that are uploaded before the delete action is completed will also be deleted.
Deleting the specified objects can't be undone.
Learn more
Specified objects
Find objects by name
Name
Version ID
Type
Last modified
Size
DSCC Practical 4.pdf
null
pdf
October 11, 2024, 00:28:11 (UTC+05:30)
123.8 KB
Permanently delete objects?
To confirm deletion, type permanently delete in the text input field.
permanently delete
Cancel
Delete objects
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

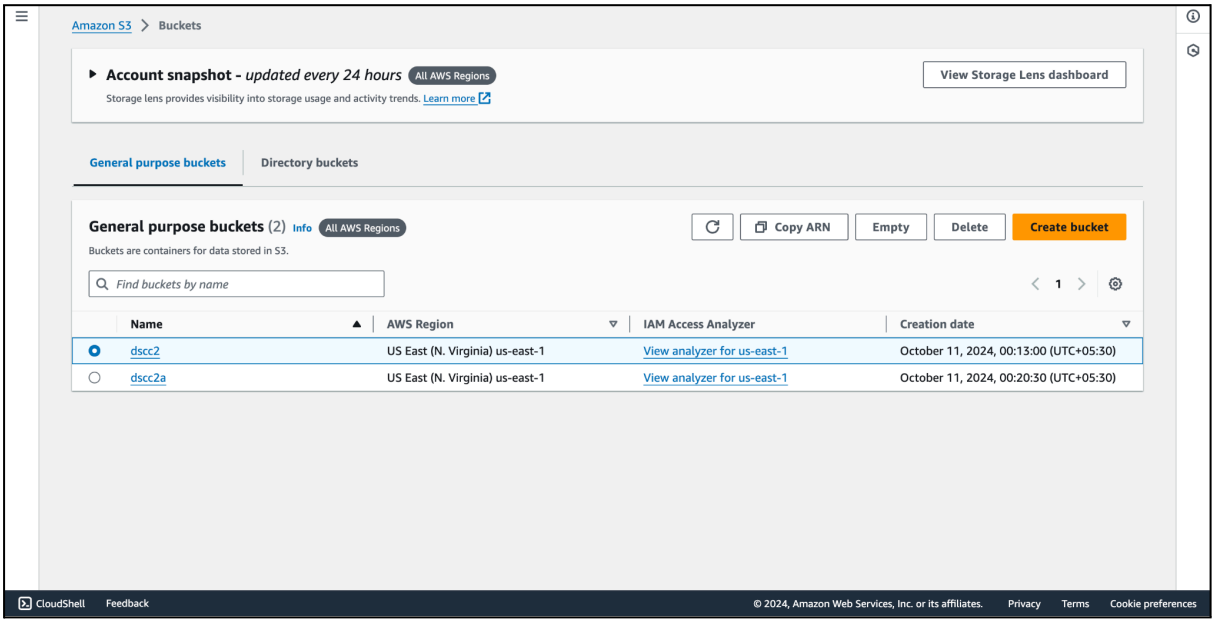

Successfully deleted objects
View details below.
Delete objects: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Source
s3://dscc2a/DSCC_CA/
Successfully deleted
1 object, 123.8 KB
Failed to delete
0 objects
Failed to delete
Configuration
Failed to delete (0)
Find objects by name
Name
Folder
Version ID
Type
Last modified
Size
Error
No objects failed to delete.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

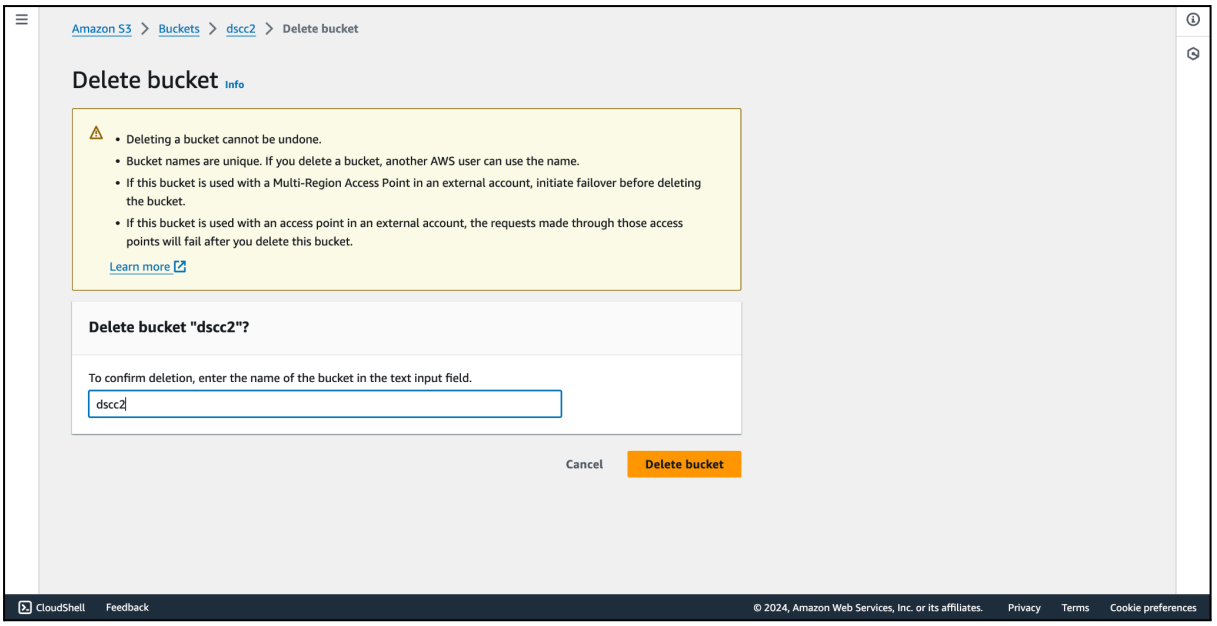

Amazon S3 > Buckets > dscc2a > DSCC_CA/
DSCC_CA/
Copy S3 URI
Objects
Properties
Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
Show versions
1
Name
Type
Version ID
Last modified
Size
Storage class
DSCC Practical 4.pdf
pdf
5Xs3qCgZFYrOl1bjtcRDgTr YOEYd9cRJ
October 11, 2024, 00:33:22 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 5.pdf
pdf
null
October 11, 2024, 00:24:07 (UTC+05:30)
78.4 KB
Standard
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences

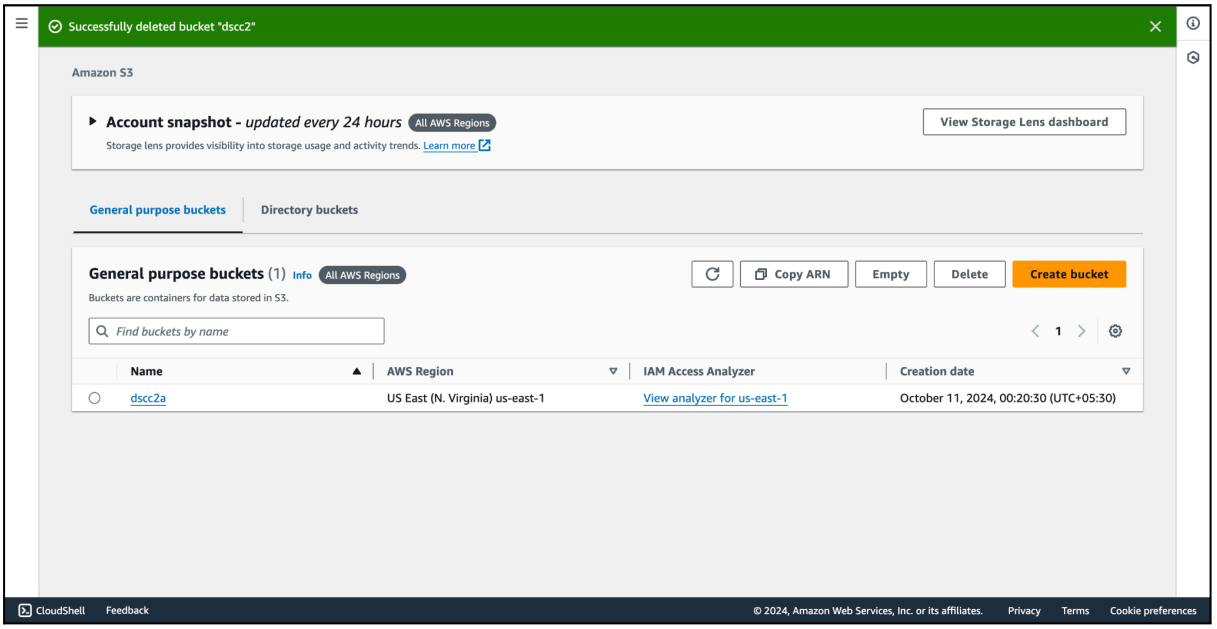

Amazon S3 > Buckets > dscc2
dscc2 Info
Objects
Properties
Permissions
Metrics
Management
Access Points
Objects (2) Info
Copy S3 URI
Copy URL
Download
Open
Delete
Actions
Create folder
Upload
Objects are the fundamental entities stored in Amazon S3. You can use Amazon S3 inventory to get a list of all objects in your bucket. For others to access your objects, you'll need to explicitly grant them permissions. Learn more
Find objects by prefix
1
Name
Type
Last modified
Size
Storage class
DSCC Practical 4.pdf
pdf
October 11, 2024, 00:18:00 (UTC+05:30)
123.8 KB
Standard
DSCC Practical 5.pdf
pdf
October 11, 2024, 00:18:01 (UTC+05:30)
78.4 KB
Standard
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Delete objects Info
If a folder is selected for deletion, all objects in the folder will be deleted, and any new objects added while the delete action is in progress might also be deleted. If an object is selected for deletion, any new objects with the same name that are uploaded before the delete action is completed will also be deleted.
Deleting the specified objects can't be undone.
Learn more
Specified objects
Find objects by name
Name
Type
Last modified
Size
DSCC Practical 4.pdf
pdf
October 11, 2024, 00:18:00 (UTC+05:30)
123.8 KB
DSCC Practical 5.pdf
pdf
October 11, 2024, 00:18:01 (UTC+05:30)
78.4 KB
Permanently delete objects?
To confirm deletion, type permanently delete in the text input field.
permanently delete
Cancel
Delete objects
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Successfully deleted objects
View details below.
Delete objects: status
Close
The information below will no longer be available after you navigate away from this page.
Summary
Source
s3://dscc2
Successfully deleted
2 objects, 202.2 KB
Failed to delete
0 objects
Failed to delete
Configuration
Failed to delete (0)
Find objects by name
Name
Folder
Type
Last modified
Size
Error
No objects failed to delete.
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Amazon S3 > Buckets
Account snapshot - updated every 24 hours
All AWS Regions
Storage lens provides visibility into storage usage and activity trends. Learn more
View Storage Lens dashboard
General purpose buckets
Directory buckets
General purpose buckets (2) Info
All AWS Regions
Buckets are containers for data stored in S3.
Copy ARN
Empty
Delete
Create bucket
Find buckets by name
Name
AWS Region
IAM Access Analyzer
Creation date
dscc2
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 11, 2024, 00:13:00 (UTC+05:30)
dscc2a
US East (N. Virginia) us-east-1
View analyzer for us-east-1
October 11, 2024, 00:20:30 (UTC+05:30)
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences


Amazon S3 > Buckets > dscc2 > Delete bucket
Delete bucket Info
Deleting a bucket cannot be undone.
Bucket names are unique. If you delete a bucket, another AWS user can use the name.
If this bucket is used with a Multi-Region Access Point in an external account, initiate failover before deleting the bucket.
If this bucket is used with an access point in an external account, the requests made through those access points will fail after you delete this bucket.
Learn more
Delete bucket "dscc2"?
To confirm deletion, enter the name of the bucket in the text input field.
dscc2
Cancel
Delete bucket
CloudShell
Feedback
© 2024, Amazon Web Services, Inc. or its affiliates.
Privacy
Terms
Cookie preferences



Conclusion:

In conclusion, the assignment on AWS account creation and the management of EC2 and S3 instances provided hands-on experience with fundamental AWS services. Creating an AWS account laid the groundwork for accessing and utilizing AWS resources, while setting up EC2 instances demonstrated the ability to deploy and manage virtual servers in the cloud. The exploration of S3 buckets involved practical exercises in object storage, including bucket creation, file uploads, and version management. This exercise underscored the flexibility and scalability of AWS services, highlighting their essential role in modern cloud computing and data management. Through this assignment, we gained valuable skills in cloud infrastructure management and data storage, which are crucial for leveraging AWS's robust capabilities.

Name of Student: Pushkar Sane			
Roll Number: 45		**CA Assignment No:2**	
Title of CA Assignment: Case study on Windows Azure Platform Appliance.			
DOP: 23-10-2024		**DOS: 24-10-2024**	
CO Mapped: **CO5**	**PO Mapped:** **PO3, PO5, PO7, PO12, PSO1, PSO2**	**Signature:**	**Marks:**

Continuous Assessment 2

Aim: Case study on Windows Azure Platform Appliance-Coca-Cola.

Coca-Cola's Use of Microsoft Azure: A Comprehensive Case Study

Coca-Cola's collaboration with Microsoft Azure is part of a broader digital transformation strategy aimed at improving operational efficiency, enhancing employee experiences, and streamlining customer engagement.

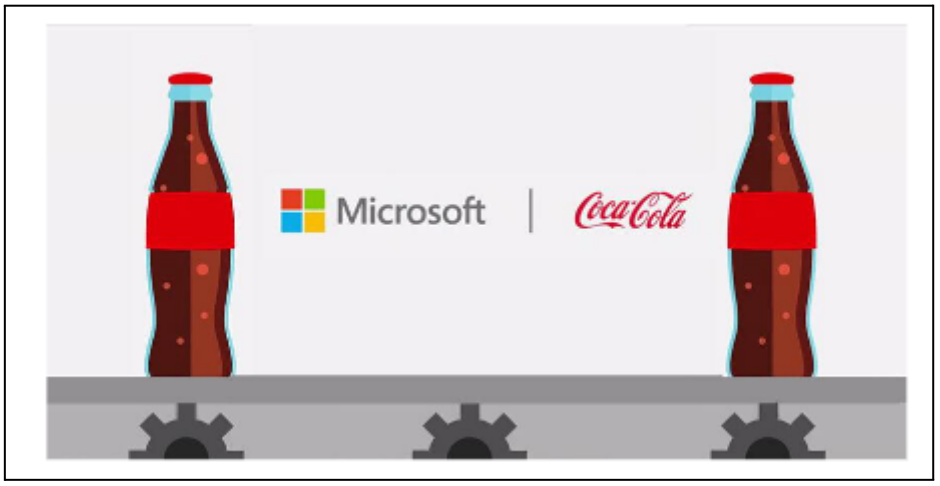
In 2020, Coca-Cola partnered with Microsoft to adopt Azure as its preferred cloud platform, alongside solutions like Dynamics 365, Microsoft 365, and the Power Platform. This allowed the company to standardize its business operations, improve decision-making through AI-driven insights, and enhance employee productivity globally. For instance, Coca-Cola has been using Dynamics 365 to improve call center operations, providing managers with real-time dashboards to monitor performance metrics and employee satisfaction.

The cloud partnership has also enabled Coca-Cola to host global virtual events, such as town halls, using Microsoft 365 Live Events, enhancing collaboration across the organization. Microsoft Teams has played a key role in connecting Coca-Cola's distributed workforce, especially during remote working periods. With Azure and Microsoft 365, Coca-Cola is delivering personalized employee experiences, replacing fragmented systems, and further integrating AI to optimize its business functions.

This transformation also extends to supply chain management and marketing, where Coca-Cola has explored generative AI tools in Azure OpenAI Service to streamline operations, improve customer experiences, and foster innovation.

Coca-Cola has made a $1.1 billion commitment to use Microsoft's cloud and artificial intelligence (AI) services. The contract is part of a five-year agreement that will see Microsoft support Coca-Cola's core technology strategy systemwide. The $1.1bn commitment will see Microsoft becoming Coca-Cola's preferred cloud and AI provider.

Coca-Cola will migrate all of its applications to Microsoft Azure. For the last year, the company has been using generative AI and has already been using Azure OpenAI service for its marketing, manufacturing, and supply chain business functions



The two companies will continue to experiment with the Azure OpenAI Service, including seeing how Copilot could help with workplace productivity. Coca-Cola is currently seeing how it can improve customer experiences and streamline operations.

“This new agreement builds on the success of Coca-Cola’s partnership strategy with Microsoft, showing our commitment to ongoing digital transformation,” said John Murphy, president and chief financial officer of The Coca-Cola Company. “Our partnership with Microsoft has grown exponentially, from the $250 million agreement we initially announced in 2020 to $1.1 billion today.”

Neeraj Tolmare, senior vice president and global CIO for The Coca-Cola Company added: “Our expanded partnership with Microsoft is an important next chapter in Coca-Cola’s journey toward a digital-first enterprise powered by emerging technologies,” “Microsoft’s capabilities help accelerate our adoption of AI to create incremental enterprise value.”
Coca-Cola previously had a data center in Atlanta, where the company is headquartered. In 2009, it was in discussions with HP to outsource its data center operations. The company

sold its Atlanta data center in 2017 to Rackhouse as part of its cloud migration plan. At that time, Coca-Cola was a customer of Amazon Web Services and Google Cloud.

1. **Modernizing Employee Experiences and Operations**

   Coca-Cola embarked on a journey to overhaul its global IT infrastructure by partnering with Microsoft to standardize operations on Microsoft Azure, Microsoft 365, Dynamics 365, and the Power Platform.

   Key Outcomes:

   a. Centralized Operations: By moving to Azure, Coca-Cola standardized its fragmented IT systems across different regions, enabling a seamless global collaboration platform.
   b. Employee Empowerment: Through the use of Microsoft 365 and Microsoft Teams, Coca-Cola enabled real-time collaboration, breaking down geographic and departmental silos, and enhancing communication within the organization.
      i. Use of Teams: The company deployed Teams to manage remote work during the COVID-19 pandemic. Virtual town halls, global meetings, and collaborative initiatives became streamlined.
      ii. Power Apps for Efficiency: Employees were empowered to build low-code apps using Power Apps, simplifying internal workflows and improving productivity

   AI-Driven Insights in Operations: With Dynamics 365 and AI-powered insights, Coca-Cola optimized its call center and field operations. Real-time dashboards allowed managers to monitor performance metrics and customer satisfaction, leading to more data-driven decision-making.

   <u>Case Study Example:</u>

   Coca-Cola used Microsoft 365 Live Events for a global virtual town hall in April 2020, during the peak of the pandemic, enabling thousands of employees to connect securely and interact in real-time

2. **Data Analytics and AI for Supply Chain Optimization**

Coca-Cola uses Azure AI and Machine Learning tools to transform its supply chain, providing enhanced insights that help reduce costs, optimize logistics, and improve product availability.

Key Applications:

a. Inventory Management: By using Azure's AI services, Coca-Cola can predict demand, reduce excess inventory, and ensure timely product delivery, enhancing the overall supply chain efficiency.

b. IoT Integration: Coca-Cola deployed IoT-enabled vending machines that use Azure's data analytics capabilities to gather real-time information about customer preferences, sales patterns, and equipment maintenance. This data allows the company to optimize the placement of machines, streamline maintenance, and boost customer satisfaction.

Example:

In Coca-Cola's European bottling operations, Azure Machine Learning was utilized to predict equipment maintenance needs, reducing downtime and lowering operational costs.

3. **Sustainability and Innovation with Azure Cloud**

   Coca-Cola has also committed to leveraging Azure's infrastructure to advance sustainability goals. The company uses Azure to track and report on key environmental metrics, including water usage, packaging waste, and carbon footprint.

   Examples:

   a. Water Stewardship: Azure's cloud services enable Coca-Cola to monitor water usage across its bottling plants worldwide, using AI to optimize water recycling and minimize waste.
   b. Sustainable Packaging: Azure's data services help Coca-Cola assess the impact of its packaging materials and production methods, aligning with the company's broader environmental sustainability goals.

4. **Customer Engagement and Marketing Innovation**

   Coca-Cola leverages Azure AI and the Power Platform to deliver personalized customer experiences, from marketing campaigns to sales strategies.

   Key Technologies:

   a. Azure OpenAI Service: Coca-Cola explores generative AI to create more personalized marketing campaigns and improve customer interactions across digital platforms. The data-driven insights allow for targeted promotions and enhanced customer loyalty programs.
   b. Customer Insights: Coca-Cola uses Azure to analyze customer behavior and preferences, helping the company tailor its product offerings and marketing strategies accordingly.

Example:

By analyzing data from vending machines, Coca-Cola tailors its product offerings to meet local preferences, ensuring that the right products are available in the right locations, increasing sales and customer satisfaction.

5. **Enhancing Security and Compliance**

   Azure provides Coca-Cola with top-tier security features, including Azure Active Directory and Advanced Threat Protection, to ensure data privacy and security across all operations.

   Key Highlights:

   a. Data Security: Coca-Cola uses Azure to protect sensitive customer and operational data while adhering to global compliance standards such as GDPR.
   b. Advanced Threat Protection: By leveraging Azure's security tools, Coca-Cola ensures that its global digital infrastructure remains secure, preventing cyberattacks and safeguarding its operations.

Conclusion: A Digital-First Future for Coca-Cola

The Coca-Cola and Microsoft Azure partnership represents a blueprint for successful digital transformation at scale. By integrating cloud computing, AI, and data analytics into its core operations, Coca-Cola has successfully optimized supply chain management, enhanced customer engagement, and empowered employees to innovate and collaborate seamlessly across the globe. The partnership reflects Coca-Cola's commitment to staying at the forefront of technology while driving sustainable growth.

References:

1. https://www.technologyrecord.com/article/microsoft-helps-coca-cola-transform-employee-experiences
2. https://www.datacenterdynamics.com/en/news/microsoft-wins-11bn-cloud-and-ai-contract-from-coca-cola/
3. https://news.microsoft.com/2024/04/23/the-coca-cola-company-and-microsoft-announce-five-year-strategic-partnership-to-accelerate-cloud-and-generative-ai-initiatives/