Implementation of Remote Procedure Call (RPC) in a Cloud Environment Using Python

NAME: NAVINESHARAN S

ROLL NUMBER: 2023115015

---

1. Introduction

Remote Procedure Call (RPC) is a communication protocol that allows a program to execute a procedure on a remote server as if it were a local function call. RPC hides the complexities of network communication and enables distributed systems to interact seamlessly.

In this experiment, RPC is implemented using Python's XML-RPC library. The server hosting the remote procedures is deployed in a cloud environment, and the client invokes those procedures remotely over the internet.

---

2. Objective

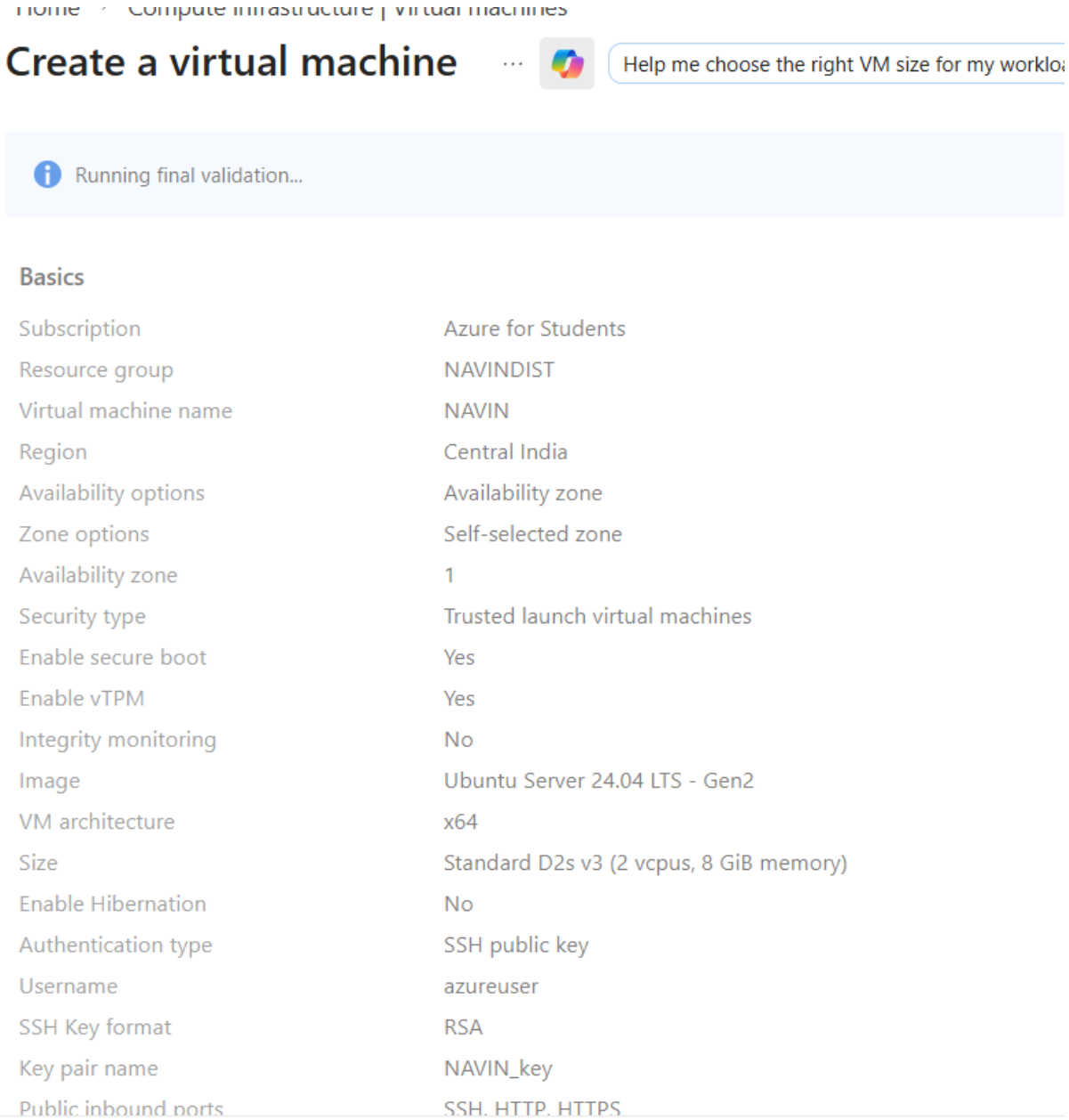
- To implement RPC using Python
- To host the RPC server in a cloud environment
- To allow a client to remotely invoke procedures from the cloud server
- To receive correct results from remote execution

---

3. Cloud Environment Setup

- Cloud Platform: **Microsoft Azure Virtual Machine**
- Server OS: **Ubuntu Linux**
- Client OS: **Windows**
- Programming Language: **Python 3**
- Library Used: xmlrpc.server, xmlrpc.client

The RPC server runs on an Azure VM with a public IP, while the client runs locally and communicates through the internet using TCP/IP.



---

4. RPC Architecture

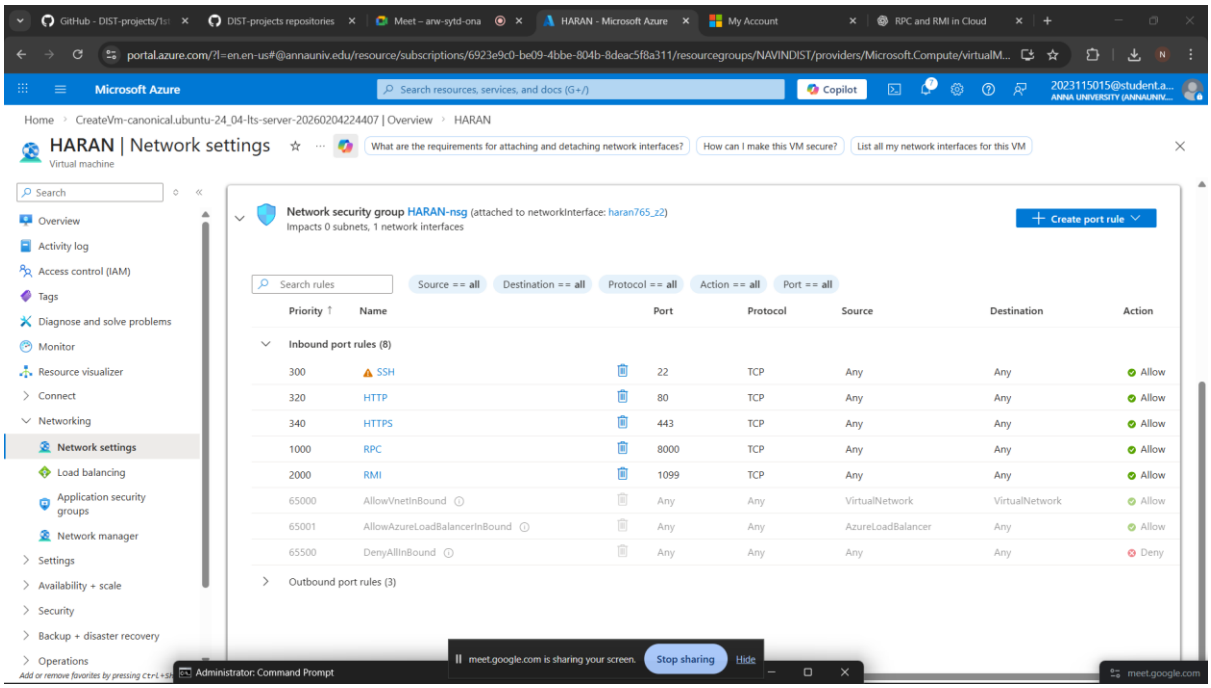
The client sends a request to invoke a procedure, the server executes it, and the result is sent back to the client.

---

5. Port Selection and Configuration

- **Port Used**: 8000
- Purpose:
    - Used by XML-RPC server to listen for client requests

Azure Network Security Group (NSG) is configured to allow inbound TCP traffic on port 8000 so that the client can access the RPC service hosted in the cloud.



---

6. Server Implementation

The server performs the following:

1. Creates a class containing remote procedures (add, subtract, multiply, divide)
2. Registers the class instance with the XML-RPC server
3. Binds the server to port 8000
4. Waits continuously for client requests

The server executes procedures requested by the client and returns the result.

---

7. Client Implementation

The client performs the following:

1. Connects to the server using the cloud VM's public IP and port 8000
2. Calls remote procedures as if they are local functions
3. Receives the computed result from the server

---

8. Cloud Connection Establishment

- The client uses the server's **public IP address**
- Communication occurs using HTTP protocol over TCP
- Remote procedures are executed on the cloud server

This confirms distributed execution between local client and cloud server.

---

9. Output

Sample Output (Client Side):

Addition: 15

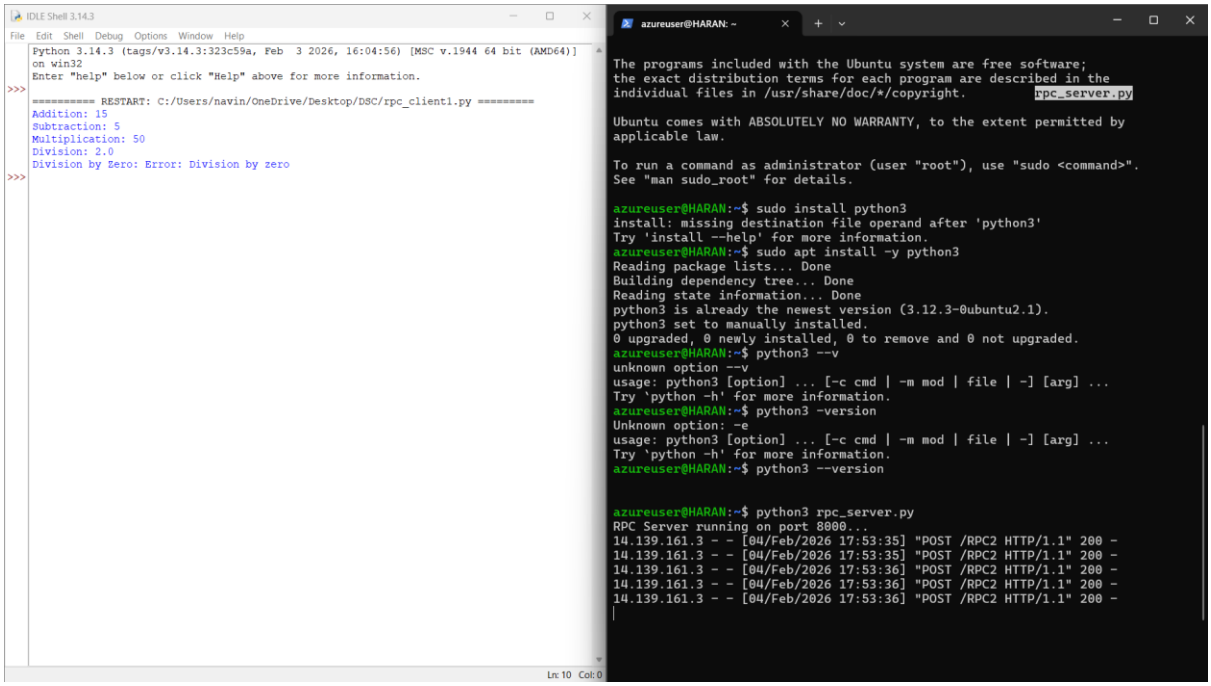
Subtraction: 5

Multiplication: 50

Division: 2.0

Division by Zero: Error: Division by zero

This proves that the procedures are executed on the cloud server and results are returned to the client.



---

10. Error Handling

- Division by zero handled using exception handling
- Network errors handled using Python exception mechanisms

- Server remains active even after multiple client requests

---

11. Conclusion

The RPC-based distributed application was successfully implemented using Python XML-RPC and deployed in a cloud environment. The client was able to remotely invoke procedures hosted on the cloud server and receive correct results. This demonstrates how RPC enables transparent communication between distributed systems over the internet.