

# Remote Procedure Call (RPC) – Cloud-Based Calculator Application

---

## 1. Aim

To implement a **Remote Procedure Call (RPC)** based calculator application where the server is hosted on a **cloud platform (AWS EC2)** and the client remotely invokes arithmetic procedures.

---

## 2. Objective

- To understand RPC-based communication in distributed systems
  - To implement client–server interaction using RPC
  - To deploy the RPC server on a cloud environment
  - To verify remote procedure execution and result transfer
- 

## 3. System Requirements

### Hardware

- Computer with minimum 4 GB RAM
- Internet connection

### Software

- Operating System: Ubuntu (Server), Windows / Kali Linux (Client)
  - Programming Language: Python
  - Cloud Platform: AWS EC2
- 

## 4. RPC Architecture

- RPC follows a **procedure-oriented client–server model**
  - The client invokes remote procedures as normal function calls
  - The server executes the procedure and returns the result
  - Communication is handled transparently over the network
-

## **5. RPC Implementation Details**

### **5.1 Remote Procedures**

The server provides the following remote procedures:

- Addition
- Subtraction
- Multiplication
- Division

### **5.2 Working Principle**

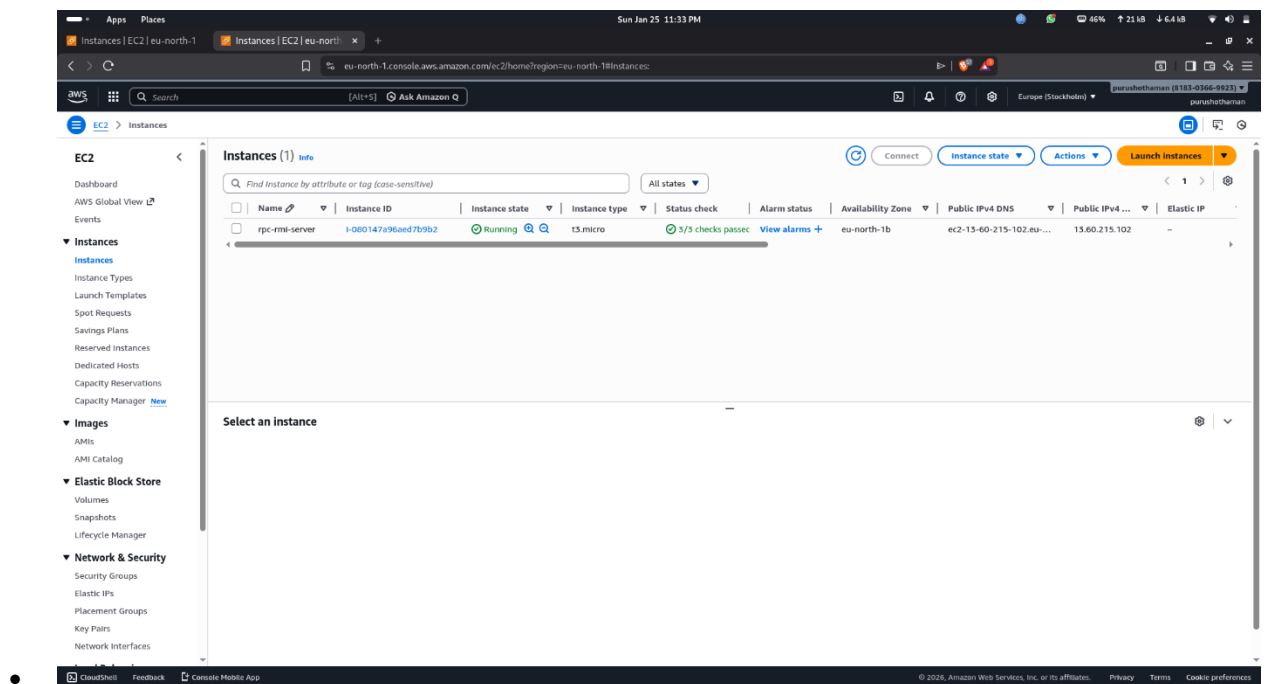
1. RPC server defines arithmetic procedures
2. Server listens on a specific port
3. Client connects using server IP address and port number
4. Client invokes procedures remotely
5. Server processes the request and returns the result

## 6. Cloud Deployment

- RPC server is hosted on **AWS EC2**
- Public IP address is used for client access
- Required ports are enabled in the EC2 security group

Attach the following screenshots:

- AWS EC2 instance running
- Security group inbound rule configuration



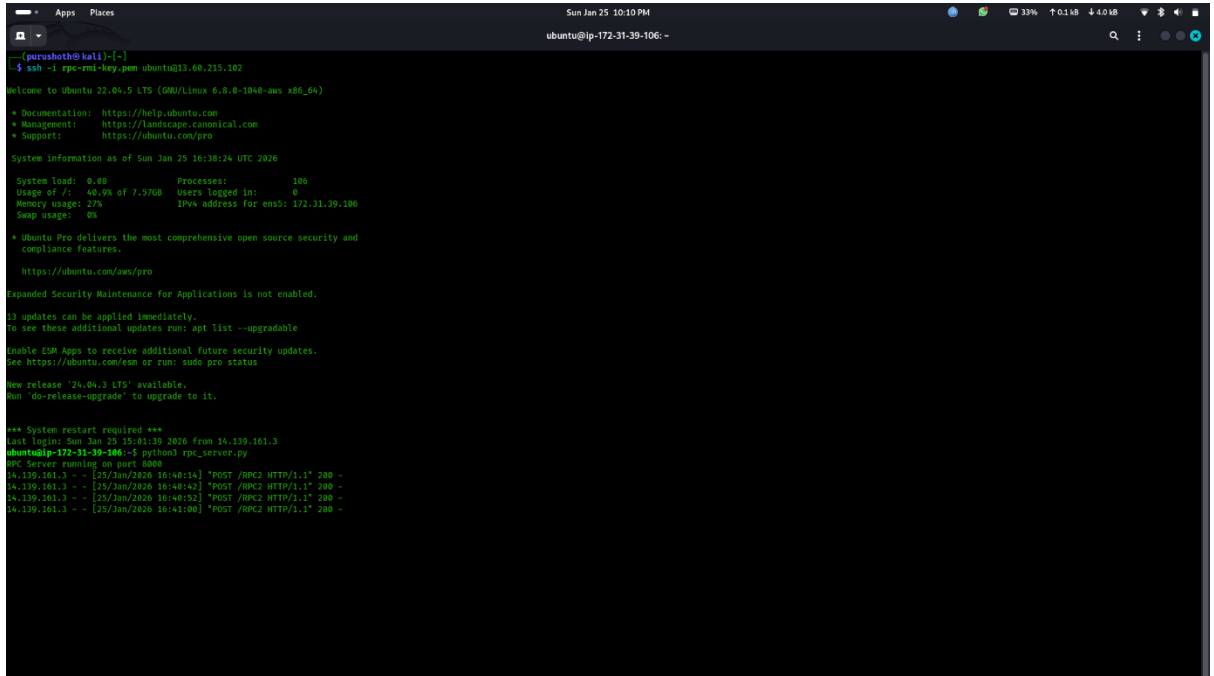
---

## 7. Error Handling

- Invalid inputs are handled safely
  - Division by zero is checked and prevented
  - Network-related exceptions are handled gracefully
-

## 8. Output

- RPC server running on AWS EC2



```
--(purushoth@kali)-[~]
$ ssh -i rpc-rmi-key.pem ubuntu@13.68.215.182
Welcome to Ubuntu 22.04.5 LTS (GNU/Linux 6.8.0-1040-aws x86_64)

+ Documentation: https://help.ubuntu.com
+ Management: https://landscape.canonical.com
+ Support: https://ubuntu.com/pro

System information as of Sun Jan 25 16:38:24 UTC 2026

System load: 0.00 Processes: 100
Usage of /: 40.9% of 7.57GB Users logged in: 0
Memory usage: 27% IP address for ens5: 172.31.39.106
Swap usage: 0%

+ Ubuntu Pro delivers the most comprehensive open source security and
compliance features.

https://ubuntu.com/aws/pro

Expanded Security Maintenance for Applications is not enabled.

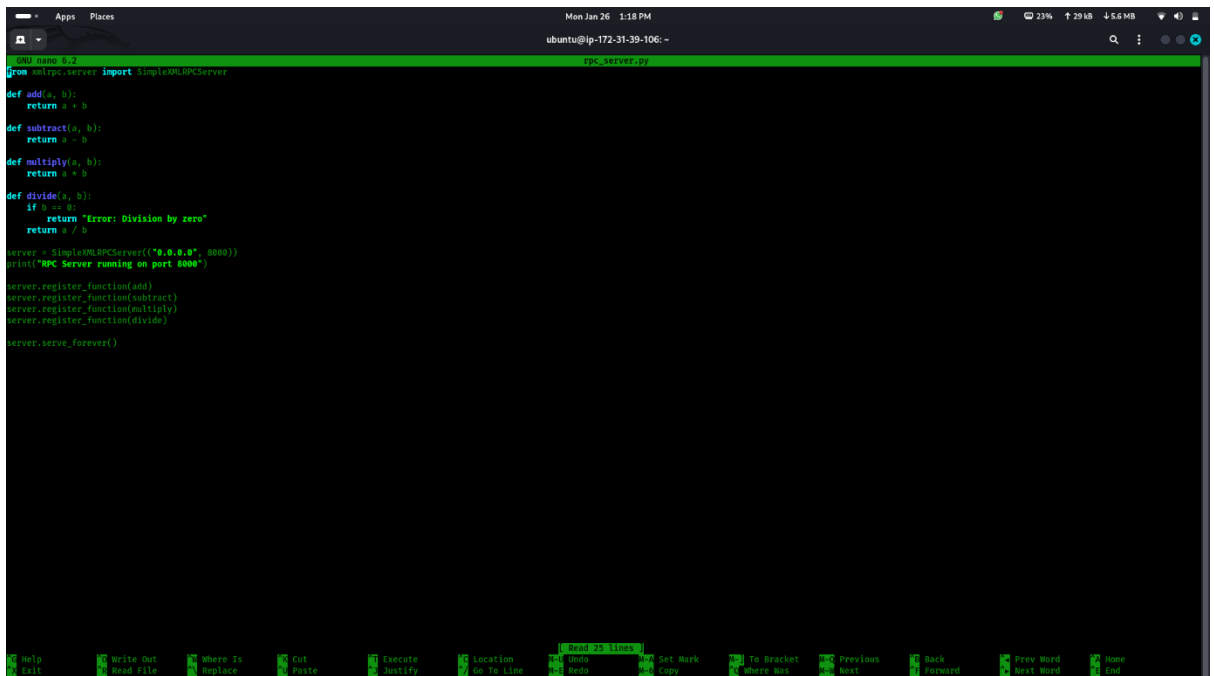
All updates can be applied immediately.
To see these additional updates run: apt list --upgradable

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status

New release '24.04.3 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

*** System restart required ***
Last login: Sun Jan 25 15:01:39 2026 from 14.139.161.3
ubuntu@ip-172-31-39-106:~$ python3 rpc_server.py
RPC Server running on port 8000
14.139.161.3 - - [25/Jan/2026 16:40:16] "POST /RPC2 HTTP/1.1" 200 -
14.139.161.3 - - [25/Jan/2026 16:40:42] "POST /RPC2 HTTP/1.1" 200 -
14.139.161.3 - - [25/Jan/2026 16:40:52] "POST /RPC2 HTTP/1.1" 200 -
14.139.161.3 - - [25/Jan/2026 16:41:00] "POST /RPC2 HTTP/1.1" 200 -
```

- Client invoking remote procedures



```
#!/usr/bin/env python3
from xmlrpc.server import SimpleXMLRPCServer

def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

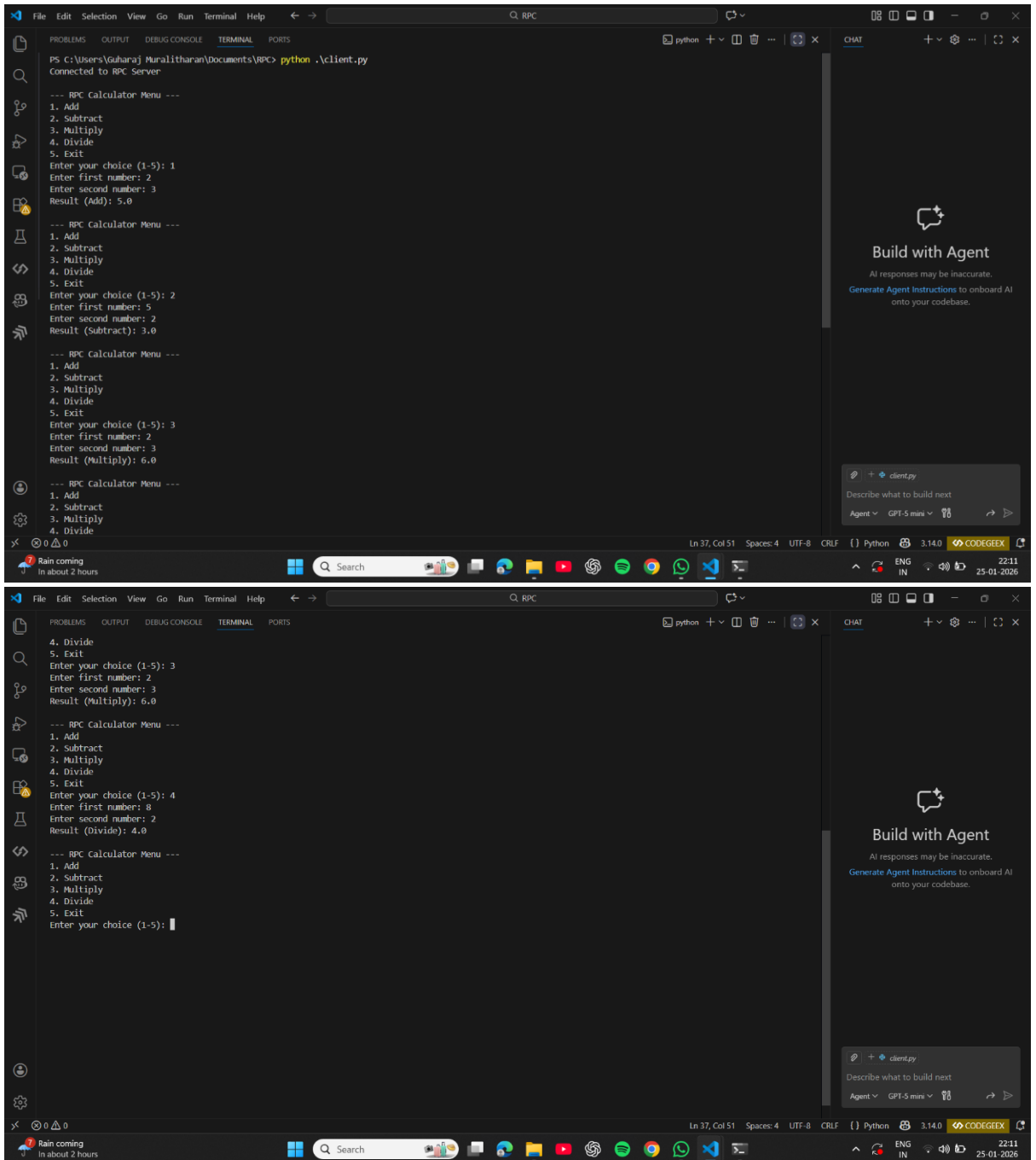
def divide(a, b):
    if b == 0:
        return "Error: Division by zero"
    return a / b

server = SimpleXMLRPCServer(("0.0.0.0", 8000))
print("RPC Server running on port 8000")

server.register_function(add)
server.register_function(subtract)
server.register_function(multiply)
server.register_function(divide)

server.serve_forever()
```

- **Correct arithmetic results displayed**



## 9. Result

The RPC-based calculator application was successfully implemented. The client remotely invoked procedures hosted on the cloud server and received correct computation results.

## **10. Conclusion**

This experiment demonstrated how RPC simplifies distributed computing by allowing remote functions to be invoked as local procedures. Deploying the server on AWS EC2 provided practical experience with cloud-based distributed systems