

# Assignment - 1: Creating VMS and Socket programming

Name: Bhargavi Rengarajan

Student ID: 862467935

## 1. Initial Setup: Creating VMS in cloud lab

- Created two Virtual Machines (VMs) using **VirtualBox** or **CloudLab**.
- Configured them to be on the same network (NAT/Bridged mode) so they can communicate.
- Assigned one VM to act as the **server** and the other as the **client**.

The Utah cluster suffered a power outage last night that has left some resources unavailable. [News](#)

▼ Your experiment is scheduled to start later

Name:	bharu21-239184
State:	scheduled
Profile:	small-lan
Creator:	bharu21
Project:	Assignment
Created:	Jan 23, 2025 11:50 AM
Scheduled:	Jan 23, 2025 12:00 PM
Expires:	Jan 24, 2025 4:00 AM (in 16 hours)

[Portal Log](#) [Share](#) [Save Parameters](#) [Copy](#) [Terminate](#)

Rspect Bindings

by clicking on them in the topology, and choosing the 'shell' menu option.  
Use 'sudo' to run root commands.

```
</instructions>
</rspec_tour>
<link client_id="link-1">
  <interface_ref client_id="vm0:eth1"/>
  <interface_ref client_id="vm1:eth1"/>
</link>
<component_manager name="urn:publicid:IDN#wisc.cloudlab.us+authority+cm"/>
<node client_id="vm0" exclusive="true" component_manager_id="urn:publicid:IDN#wisc.cloudlab.us+authority+cm">
  <sliver_type name="emulab-xen">
    <disk_image name="urn:publicid:IDN#emulab.net+image+emulab-ops/UBUNTU20-64-STD"/>
  </sliver_type>
  <interface client_id="vm0:eth1"/>
</node>
```

Topology View List View Manifest Graphs Portstats Bindings vm1 vm0

ID	Node	Type	Cluster	Status	Startup	Image	SSH command (if you provided your own key)	
vm0	c220g2-010622vm-1	pcvm	Wisc	ready	n/a	emulab-ops/UBUNTU20-64-STD	ssh -p 26810 bharu21@c220g2-010622.wisc.cloudlab.us	<input type="checkbox"/>
vm1	c220g2-010622vm-2	pcvm	Wisc	ready	n/a	emulab-ops/UBUNTU20-64-STD	ssh -p 26811 bharu21@c220g2-010622.wisc.cloudlab.us	<input type="checkbox"/>
c220g2-010622	c220g2-010622	c220g2	Wisc	n/a	n/a	n/a	ssh bharu21@c220g2-010622.wisc.cloudlab.us	<input type="checkbox"/>

## 2. Server Side output

Created a server program Node.js to:

- Bind to a specific IP address and port (e.g., **0.0.0.0:8080**).
- Wait for incoming connections from clients.
- Send the file (**mydata.txt**) to the client.

### i) Server Program

```

const net = require('net');
const fs = require('fs');

const PORT = 8080;

// Create the server
const server = net.createServer((socket) => {
  console.log('Client connected.');
```

  

```

  // Open the file to read
  const fileStream = fs.createReadStream('mydata.txt');
```

  

```

  // Pipe the file stream to the socket
  fileStream.pipe(socket);

  fileStream.on('end', () => {
    console.log('File sent successfully.');
```

  

```

    socket.end();
  });
  socket.on('close', () => {
    console.log('Connection closed.');
```

  

```

  });
  socket.on('error', (err) => {
    console.error('Socket error:', err.message);
  });
});

// Start the server
server.listen(PORT, () => {
  console.log(`Server is listening on port ${PORT}`);
});

```

### **i) Application running:**

- Ran the server program (**server.js**) on the Server VM.
- Observed the following logs:

- Confirmation of the server listening on **PORT 8080**.
- Logs showing when a client connects and when the file transfer is complete.

```
bharu21@vm0:~$ sha256sum mydata.txt
9248a4b3f78c8f2fa9885cab305402f759d88f34456a8357bab3821f4d8bc403 mydata.txt
bharu21@vm0:~$ node server
Server is listening on port 8080
Client connected.
File sent successfully.
Connection closed.
```

## II) Checksum (Server):

```
Topology View List View Manifest Graphs Portstats Bindings vm1@ vm0@
bharu21@vm0:~$ sha256sum mydata.txt
9248a4b3f78c8f2fa9885cab305402f759d88f34456a8357bab3821f4d8bc403 mydata.txt
bharu21@vm0:~$
```

The checksum value

9248a4b3f78c8f2fa9885cab305402f759d88f34456a8357bab3821f4d8bc403

### 3. Client output:

Created a client program to:

- Connect to the server using its IP address and port (**172.17.34.1:8080**).
- Receive the file sent by the server.
- Save the file as **mydata\_client\_copy.txt** on the client VM's local filesystem.

#### i) Client program

```
const net = require('net');
const fs = require('fs');
```

```
const PORT = 8080;
const SERVER_IP = '172.17.34.1'; // Replace with the IP address of VM0
```

```
// Create a socket to connect to the server
```

```
const client = net.createConnection({ port: PORT, host: SERVER_IP }, ()
=> {
```

```
console.log('Connected to server.');
```

```
// Open a writable stream to save the received file
const fileStream = fs.createWriteStream('mydata_client_copy.txt');
```

```
// Pipe the socket data into the writable stream
client.pipe(fileStream);
```

```
fileStream.on('finish', () => {
  console.log('File received and saved as mydata_client_copy.txt');
  client.end();
});
```

```
client.on('error', (err) => {
  console.error('Connection error:', err.message);
});
```

```
client.on('end', () => {
  console.log('Disconnected from server.');
```

```
});
```

## **ii) Application running:**

Ran the client program (**client.js**) on the Client VM.  
Observed the following logs:

- Confirmation of connection to the server.
- Logs showing the file was received and saved as **mydata\_client\_copy.txt**.
- Disconnection from the server after the transfer.

Topology View List View Manifest Graphs Portstats Bindings vm1 vm0

```
bharu21@vm1:~$ ls
client.js
bharu21@vm1:~$ node client
Connected to server.
Disconnected from server.
File received and saved as mydata_client_copy.txt
bharu21@vm1:~$ ls
client.js  mydata_client_copy.txt
bharu21@vm1:~$ sha256sum mydata_client_copy.txt
```

### iii) Checksum (Client):

Topology View List View Manifest Graphs Portstats Bindings vm1 vm0

```
bharu21@vm1:~$ ls
client.js
bharu21@vm1:~$ node client
Connected to server.
Disconnected from server.
File received and saved as mydata_client_copy.txt
bharu21@vm1:~$ ls
client.js  mydata_client_copy.txt
bharu21@vm1:~$ sha256sum mydata_client_copy.txt
9248a4b3f78c8f2fa9885cab305402f759d88f34456a8357bab3821f4d8bc403  mydata_client_copy.txt
bharu21@vm1:~$
```

### Checksum:

9248a4b3f78c8f2fa9885cab305402f759d88f34456a8357bab3821f4d8bc403

### RESULT:

The client and server gives the same check sum. Hence the give program executes and complies with the assignment.