**NETWORK ARCHITECTURE – I**

FALL - 2016

Project Report

Part I - GENI/Socket programming Warm-up

Submitted by:

Name: **Anusha Malineni**

Student ID: **16233382**

**Initial Setup**

**Introduction**

In this part, we will develop and deploy a simple TCP and UDP Client and server program in GENI. This consist of a simple exchange of messages between client and server and File transfer.

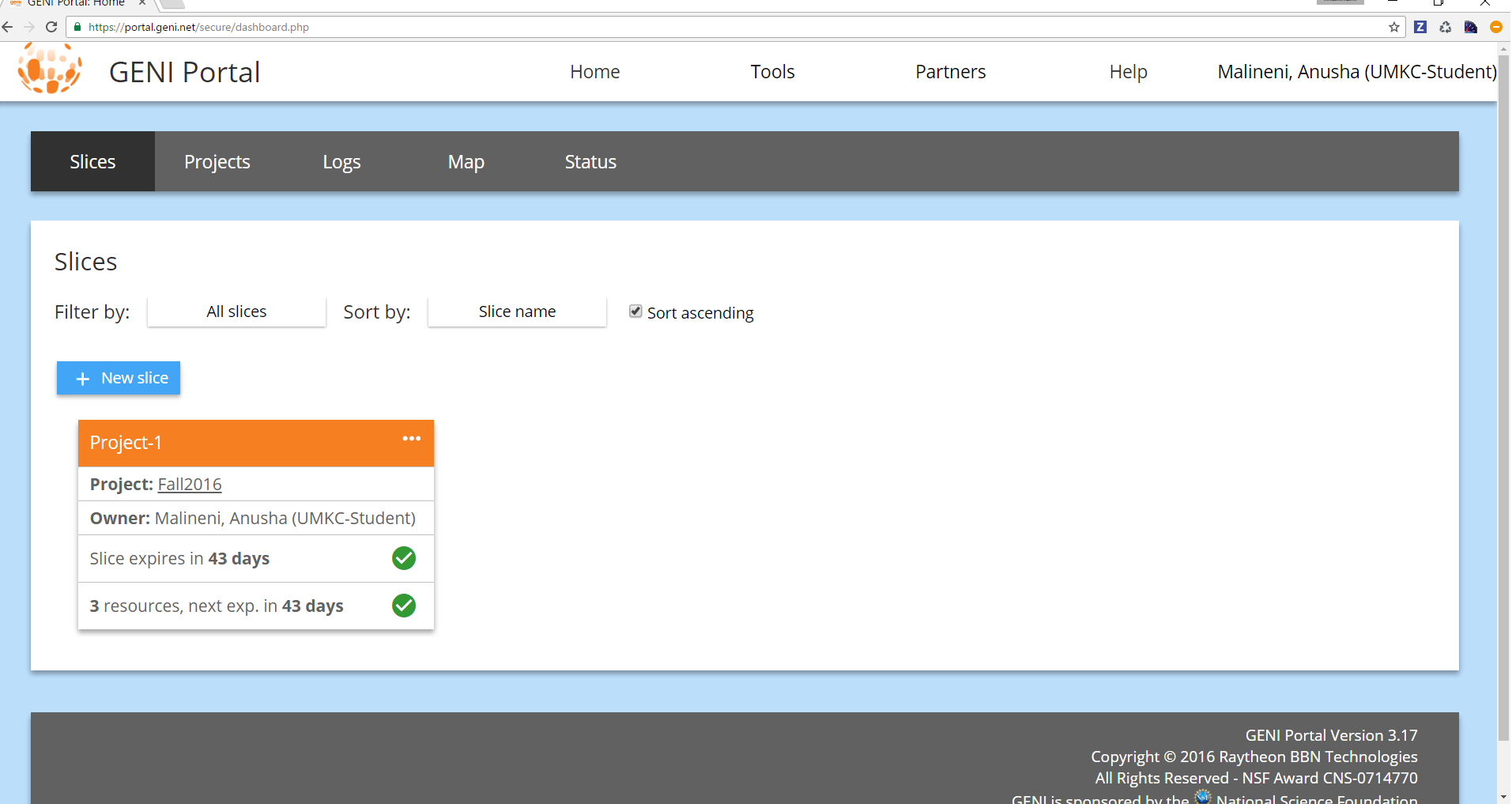
**Initial Setup**

* Creation of GENI account
* Creation of slice
* Allocation of Resource

**Step 1**

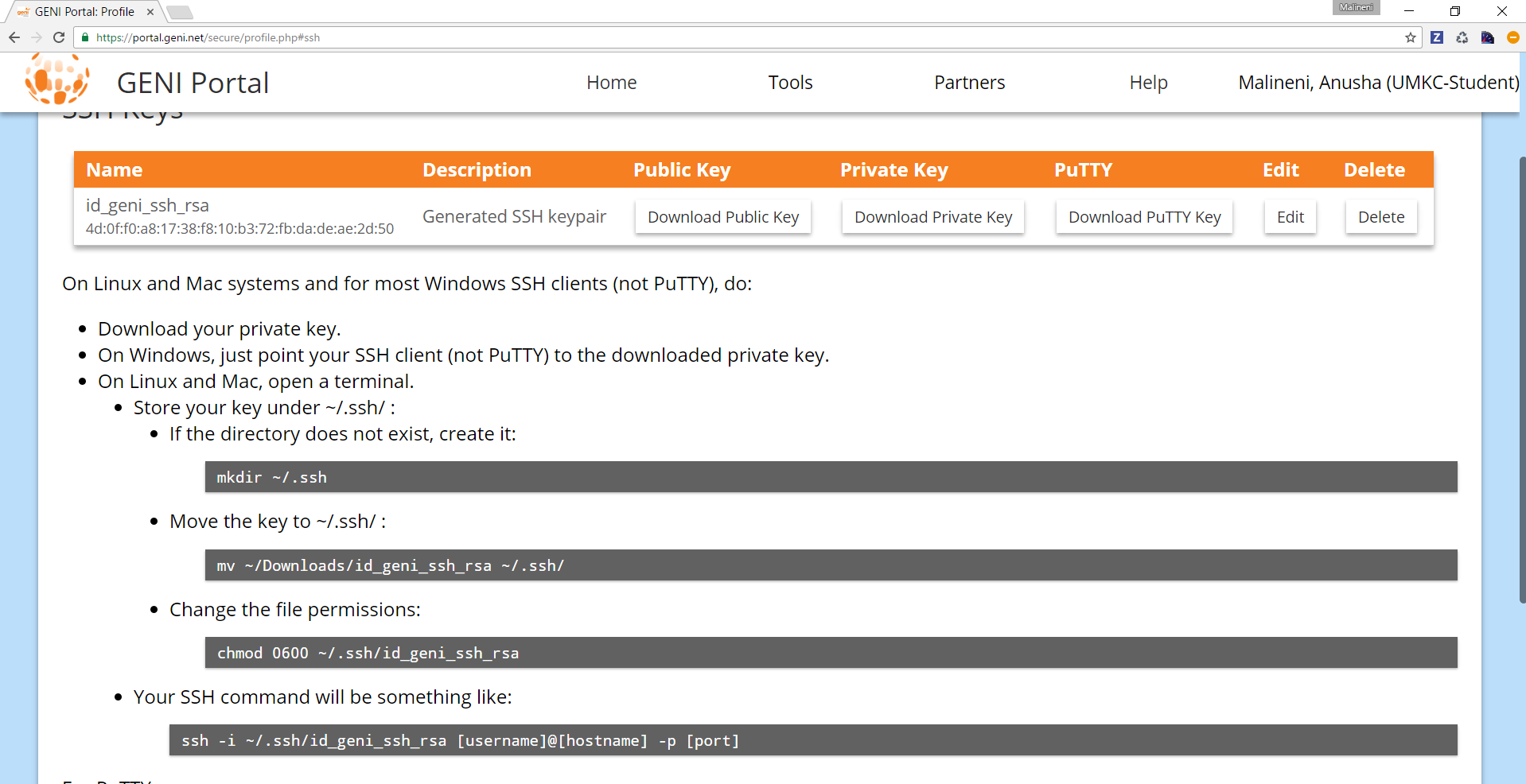
Initially, join a new or an existing project. After the approval from the project lead we can then be part of the project.





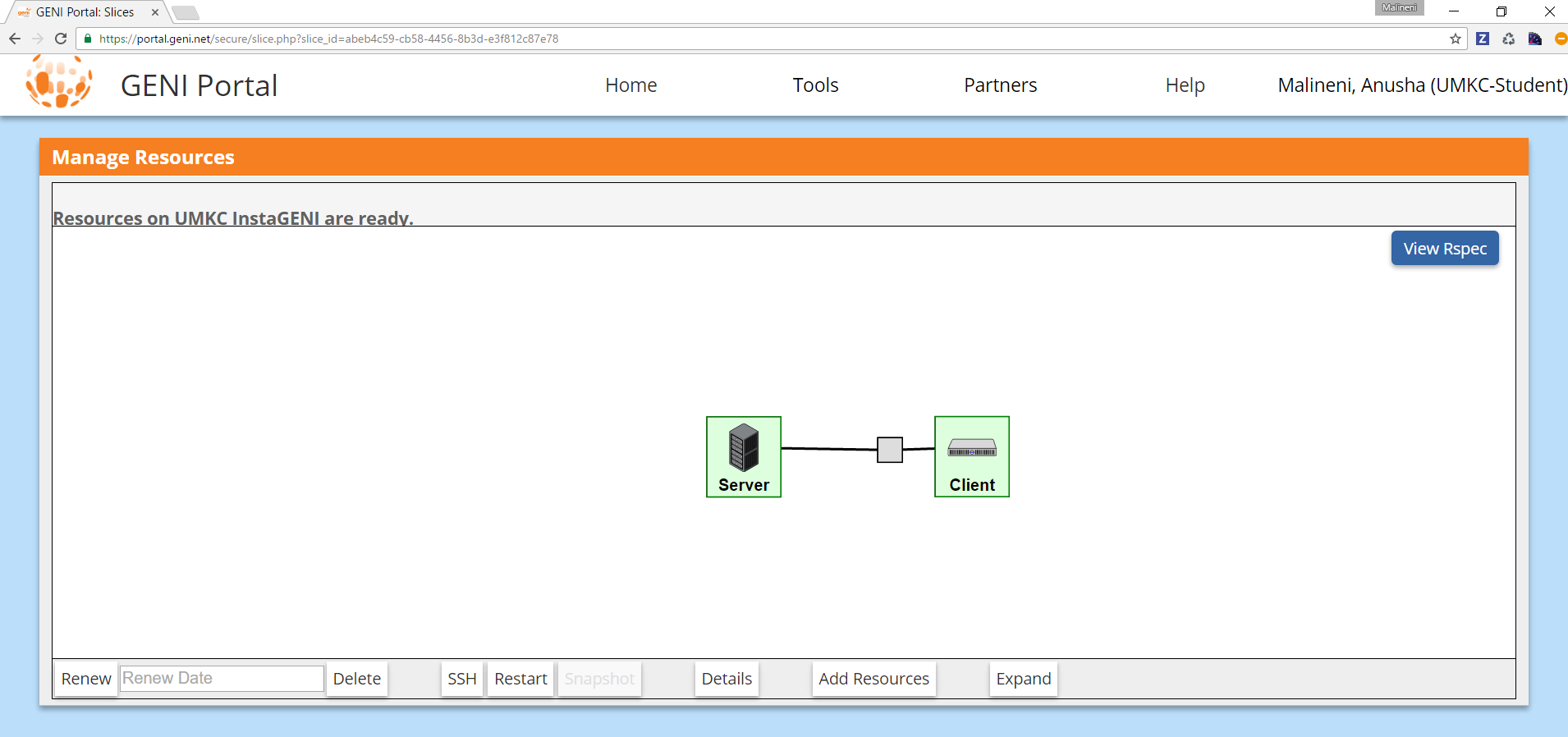
**Step 2**

After logging to the Geni portal SSH keys are downloaded for authentication purpose.

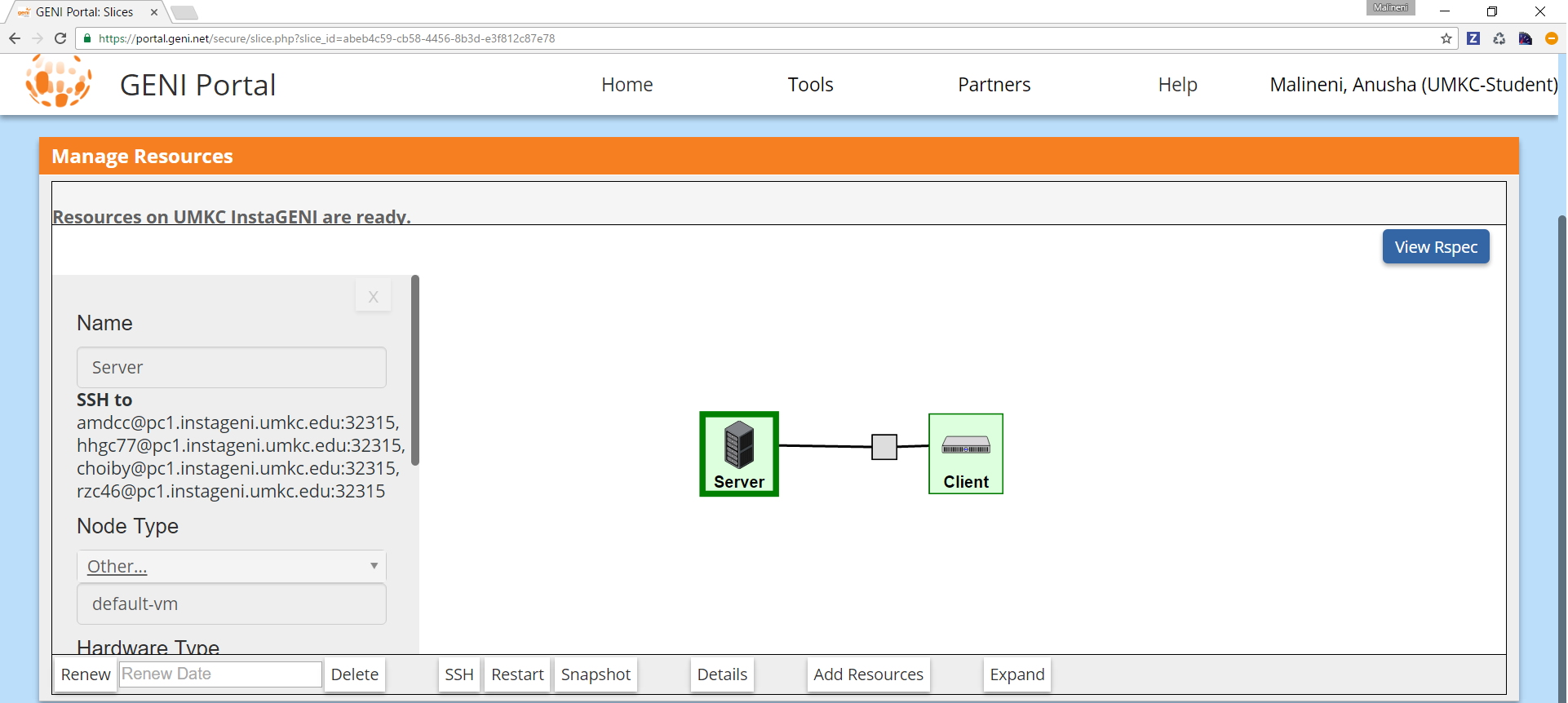
****

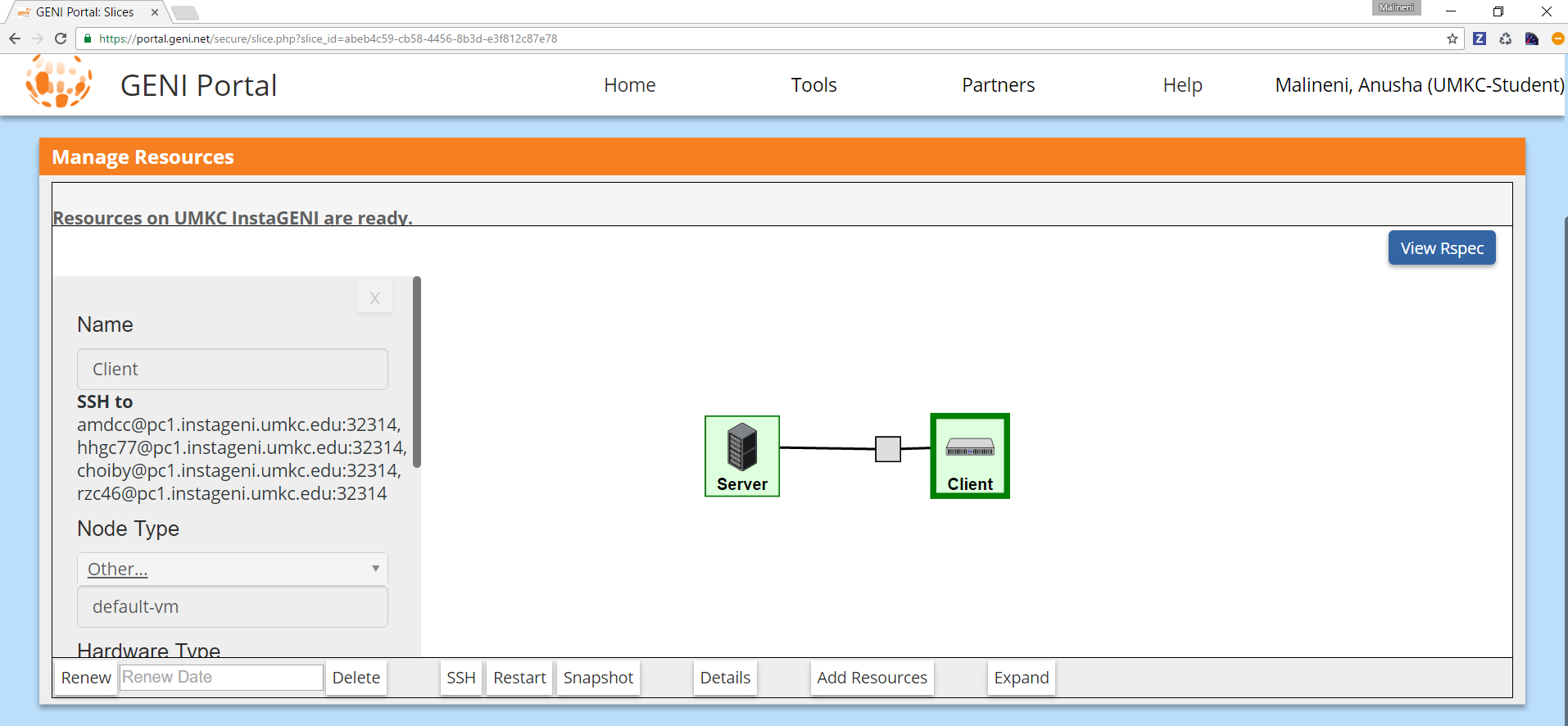
**Step 3**

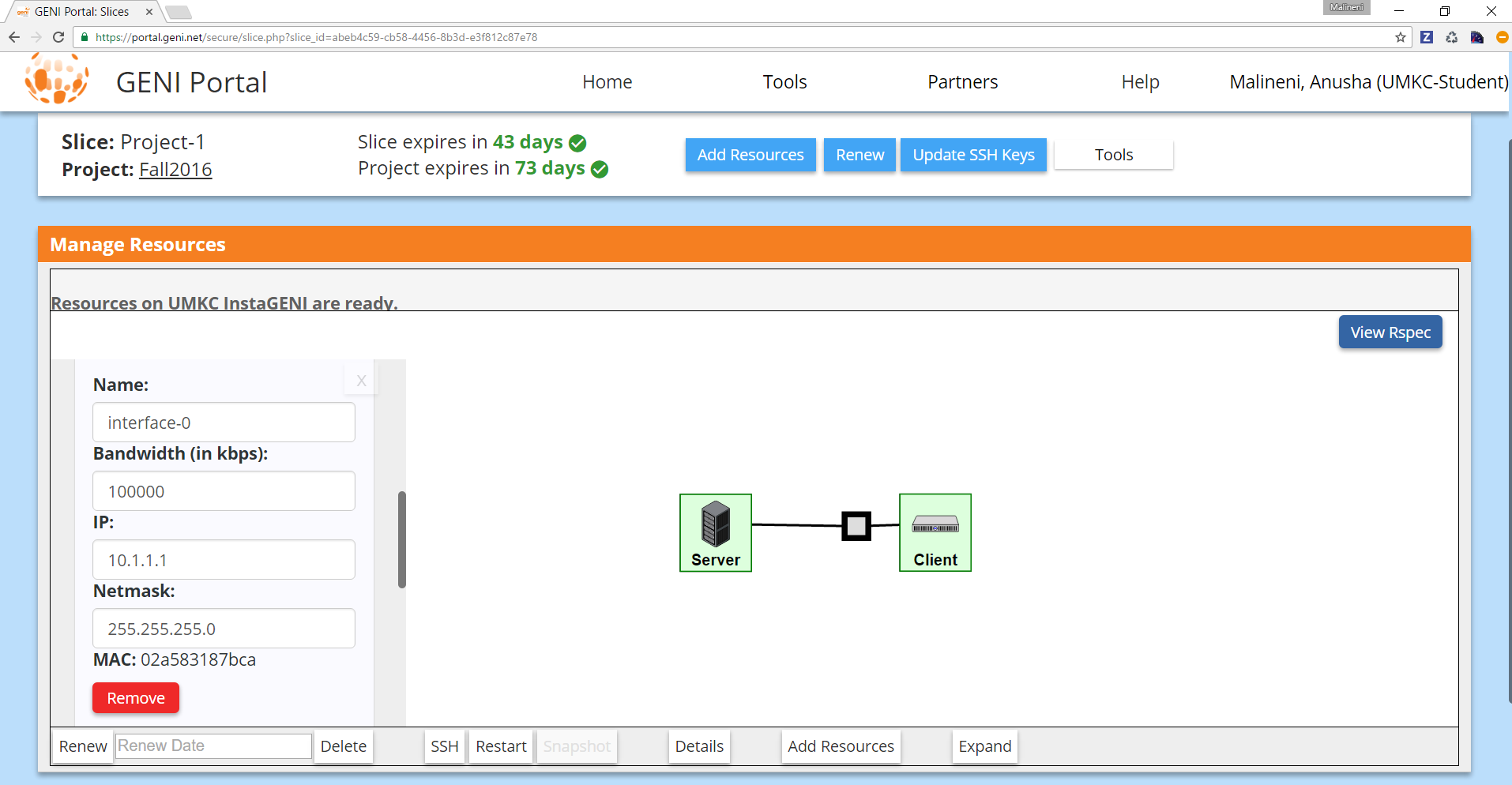
After a slice “Project-1” has been created we need to add resources by adding two VM’S namely server and client and establish a connection between them.

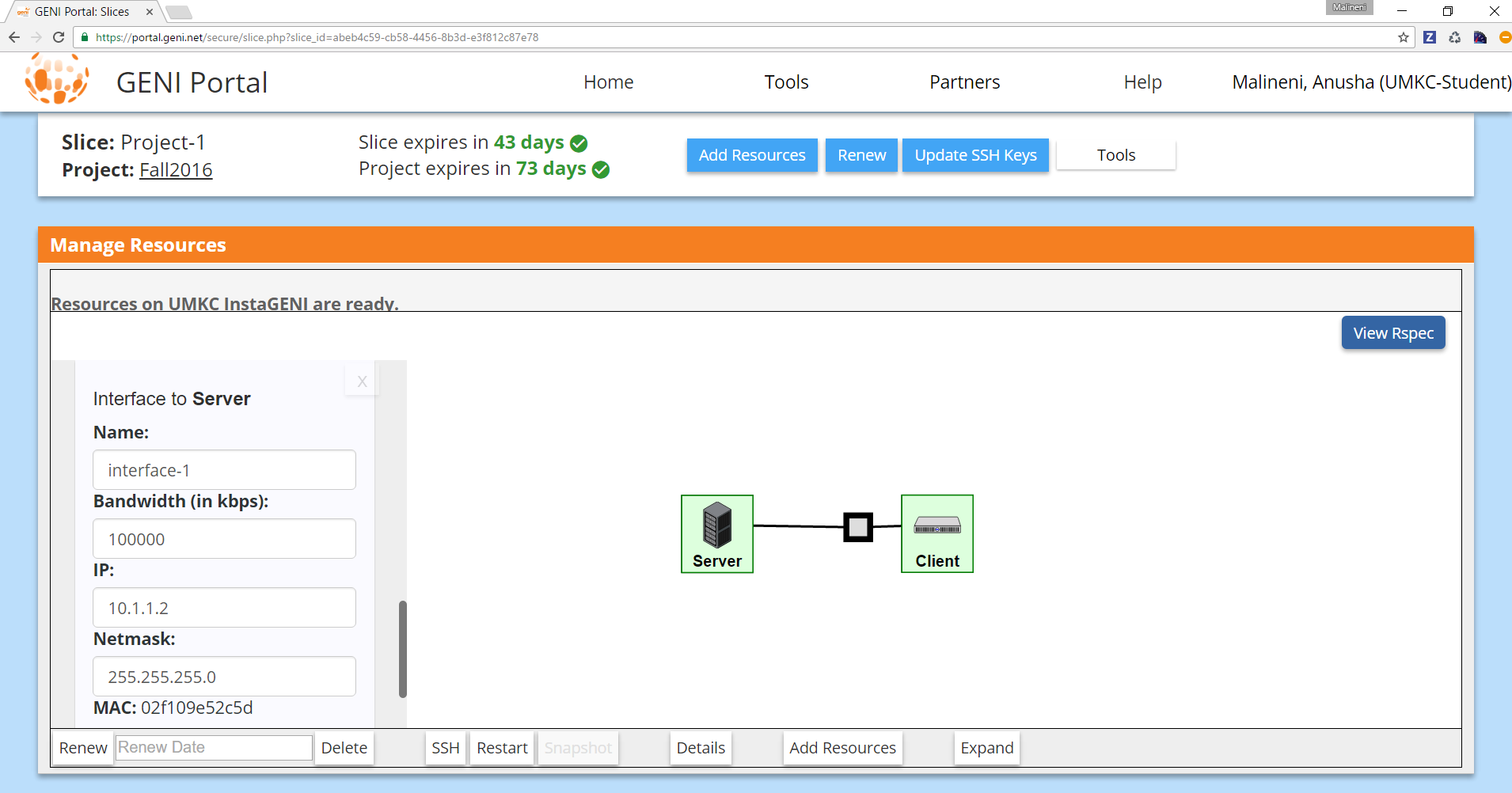


The names of VM’s are changed by clicking on the VM’s and naming them accordingly.

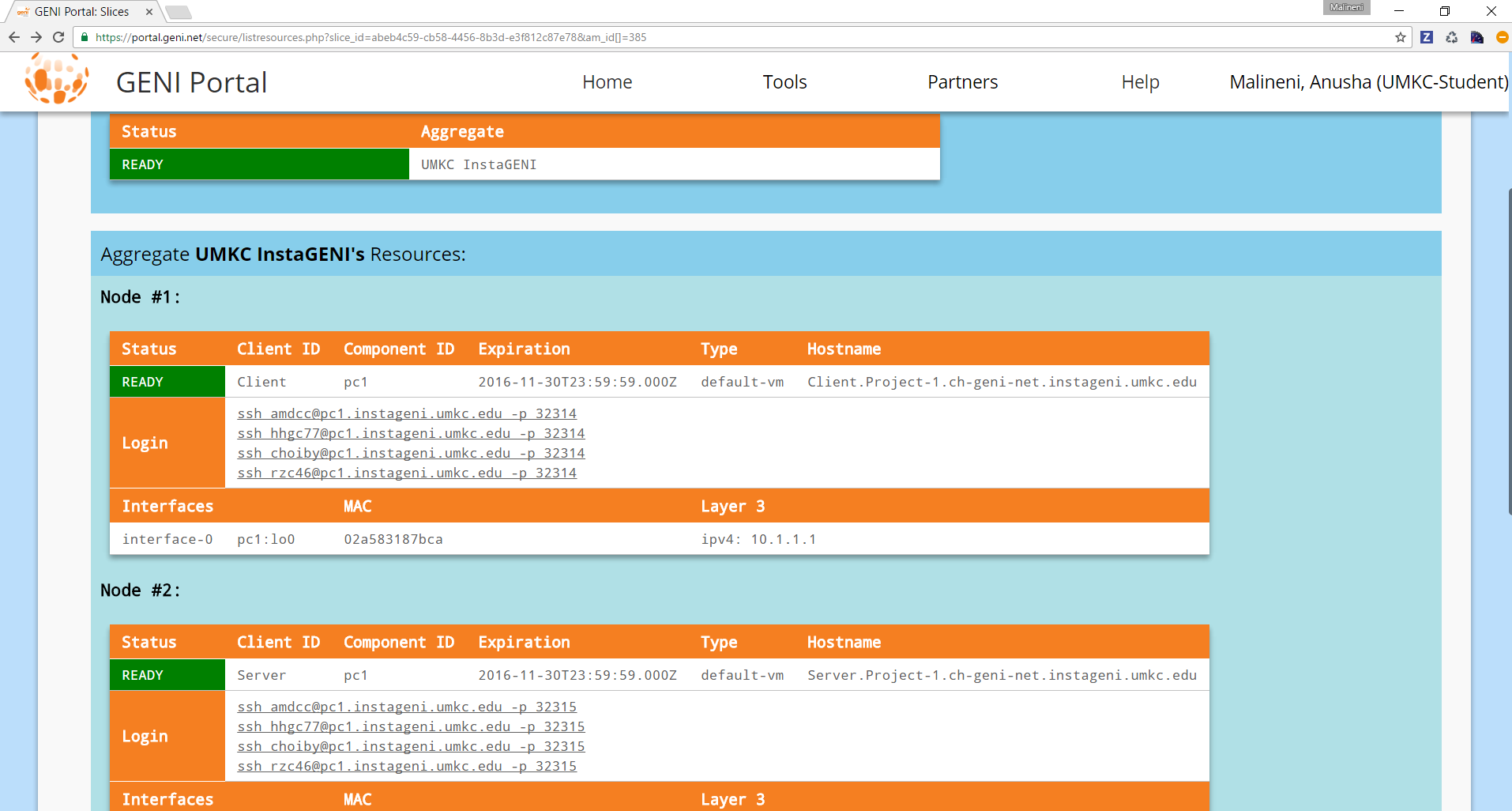








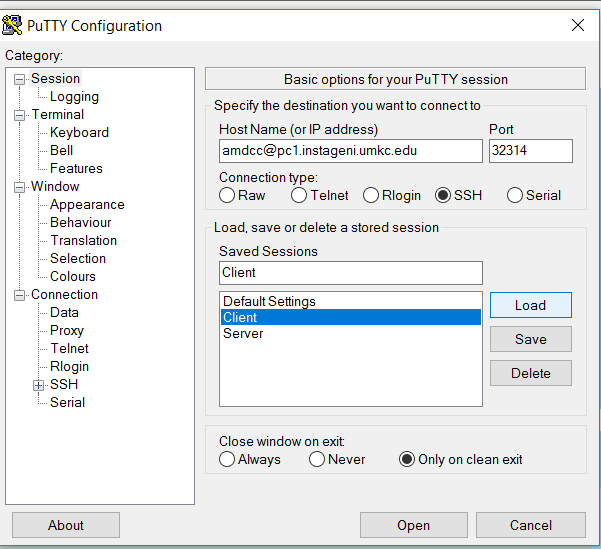
**Status of the resources in GENI portal.**

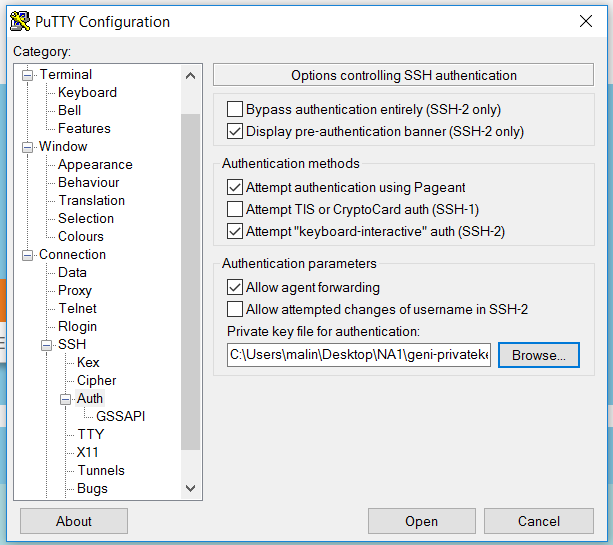


**Step 4**

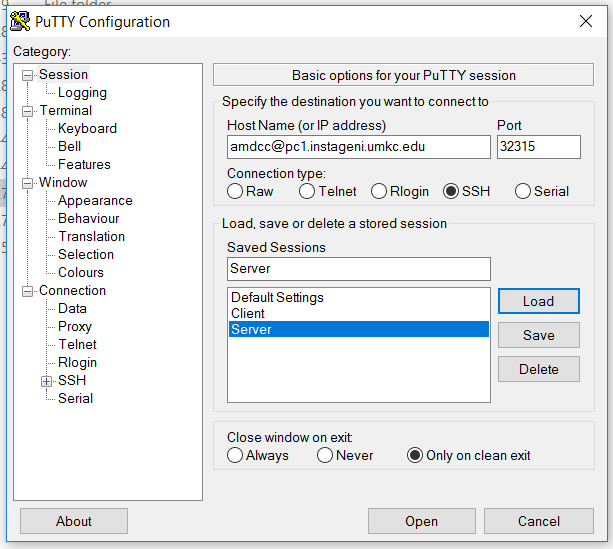
Load both client resources and server resources to putty using private key generation through putty gen.

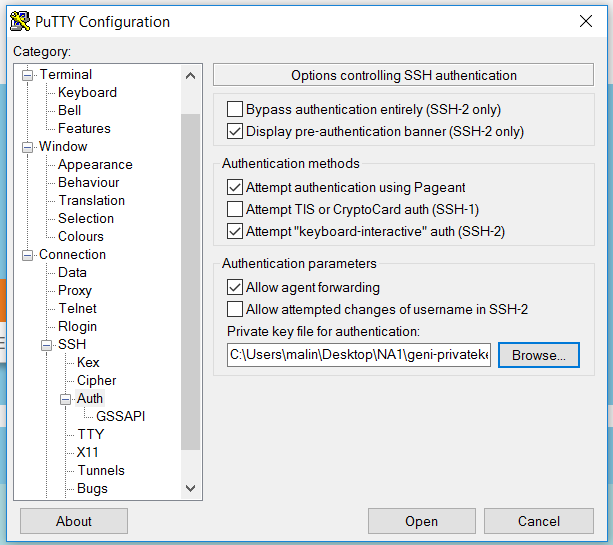
**For Client:**





**For Server:**





**a) Start from client message ‘Hello from Client-your names’ and server responses with ‘Hello from Server-your names’. Then messages from each side are echoed to each other. The program quit the program with typing ‘Bye from Client-your name’ and ‘Bye from Server-your name’.**

Part (a): Client – Server Communication using TCP (Transmission Control Protocol):

1. Enable the Client and Server window by using the Putty file with user ID’s and logging into host numbers which are obtained in the details while creating the slice as mentioned above.

2. Insert the java files containing the program required to enable the communication between the Client and Server using the command vi “filename.java”.

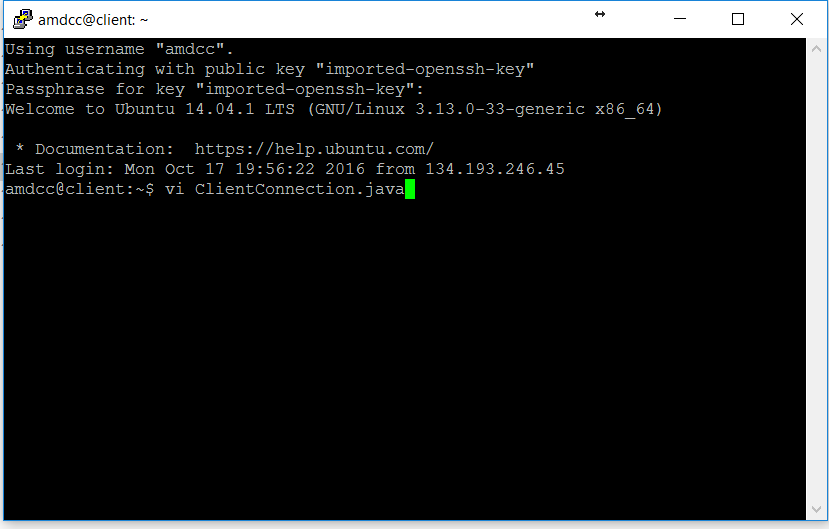
3. Here, we have used ClientConnection and ServerConnection as class names for Client and Server respectively.

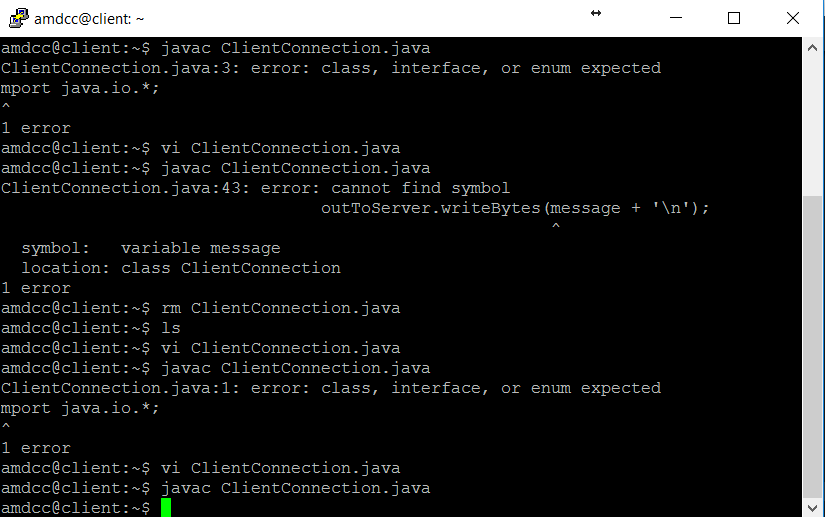
4. Compile the programs using the command “javac filename.java”.

5. Run the class files using the command “java classfilename”.

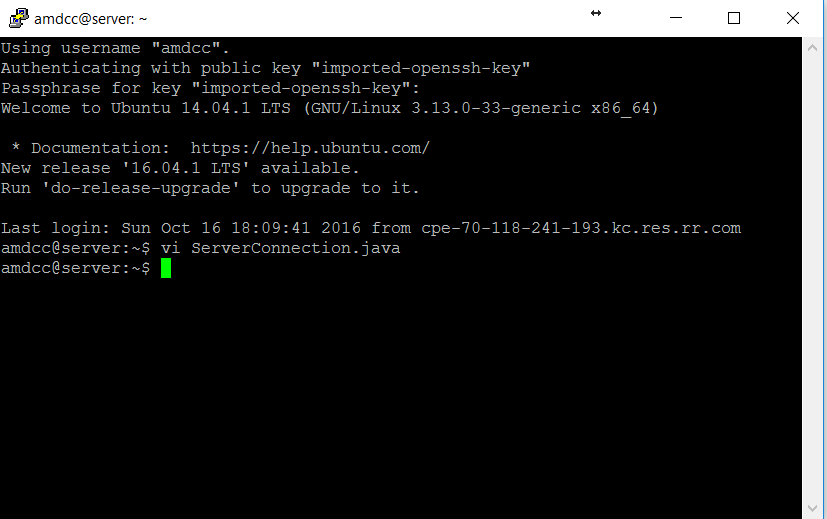
6. We then continue with the communication between the Client and Server as following:

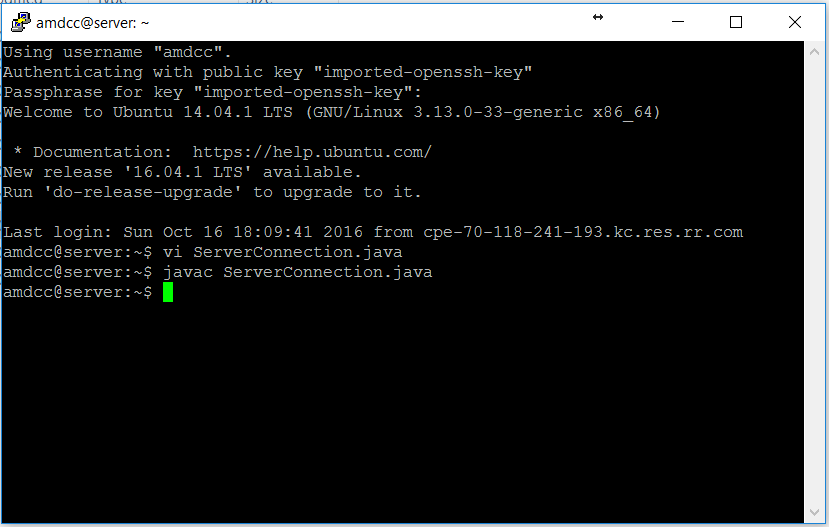
**For Client:**





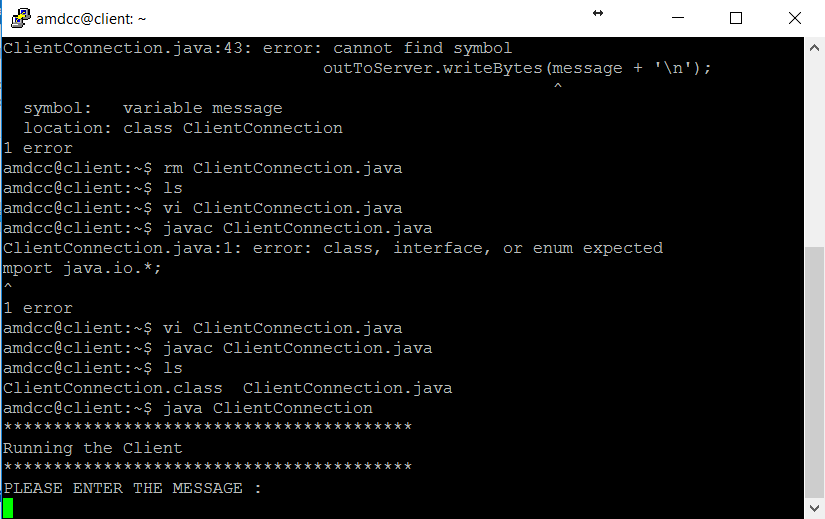
**For Server:**



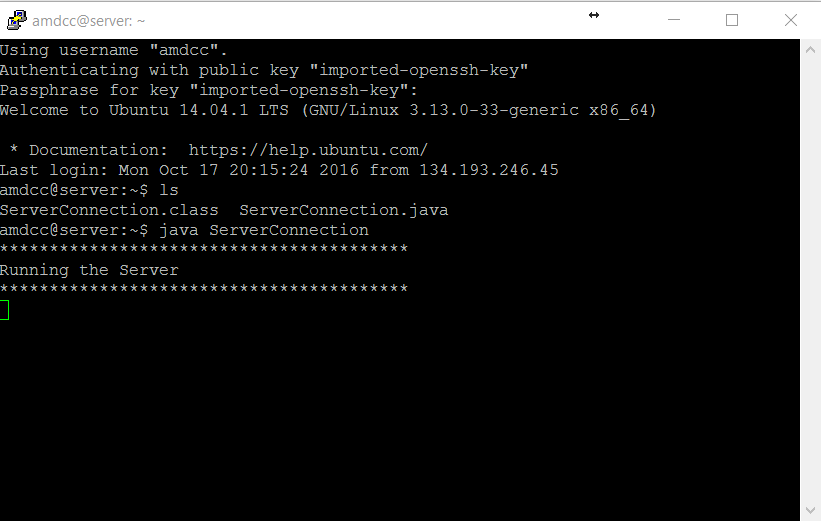


**Estabishing connection between server and client:**

**At Client:**

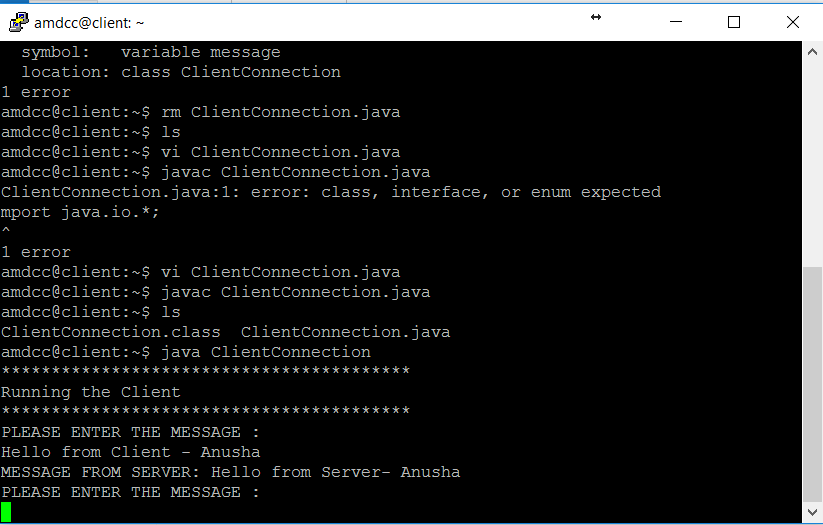


**At Server:**

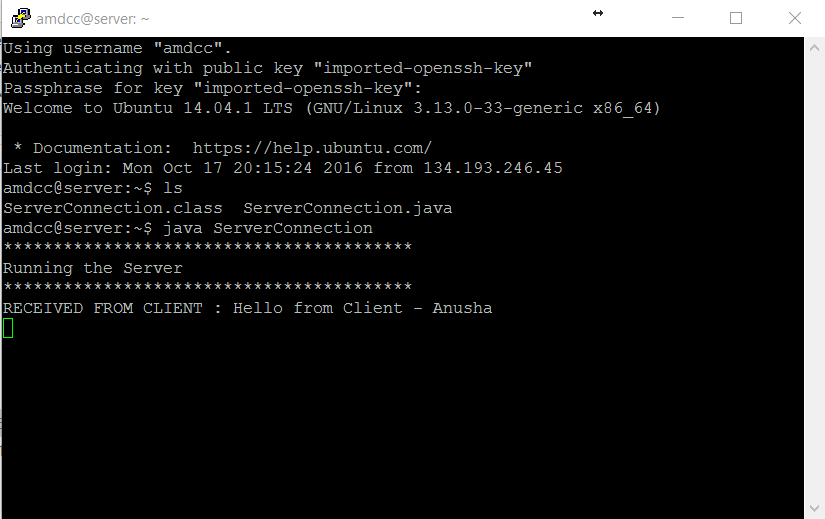


**Flow of messages from server and client:**

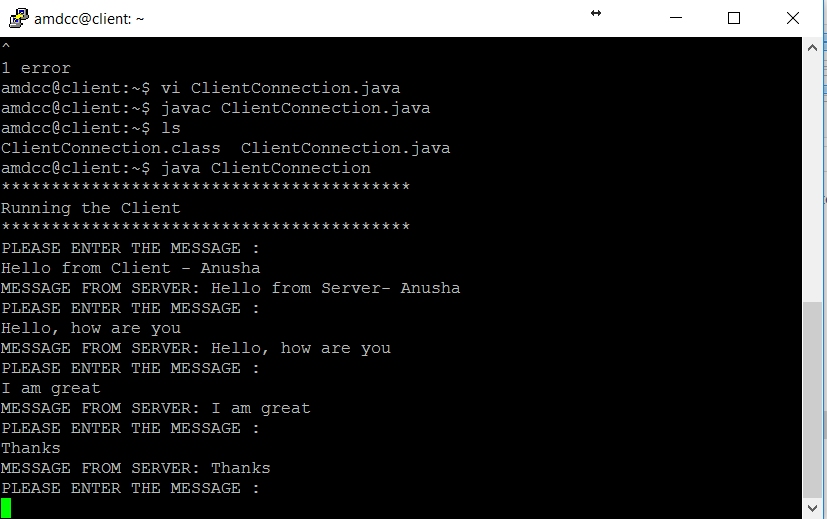
**At client:**



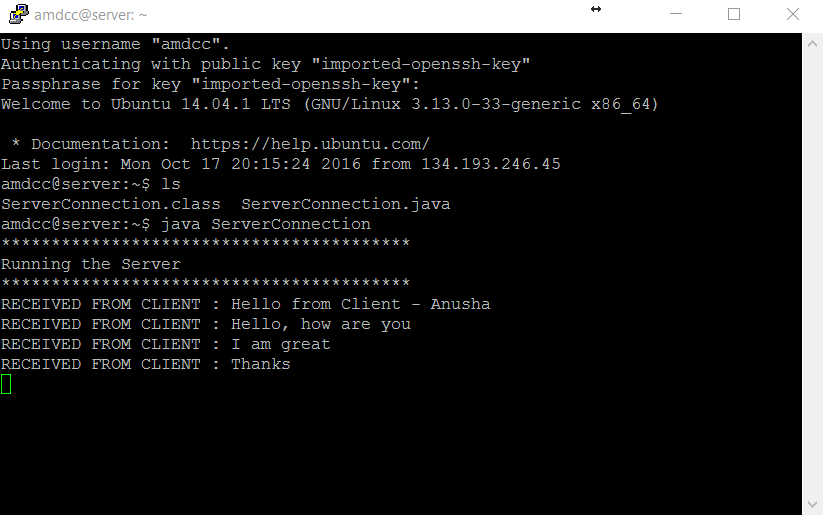
**At server:**



**At client:**

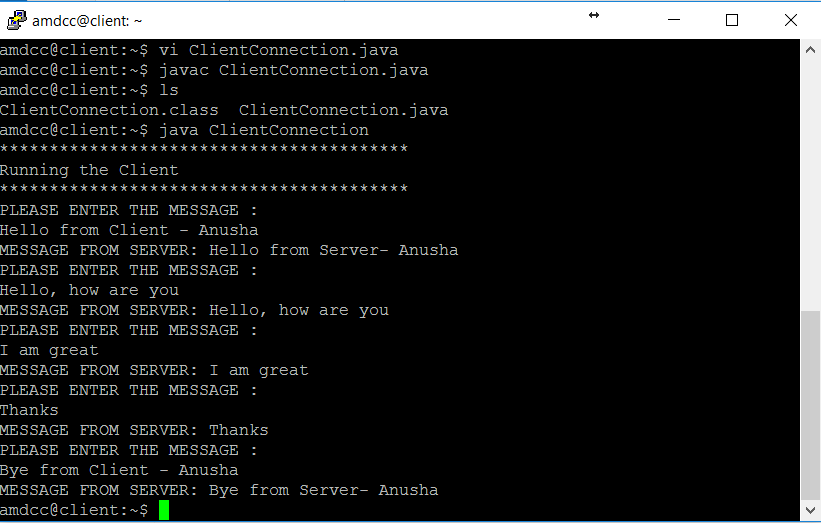


**At server:**

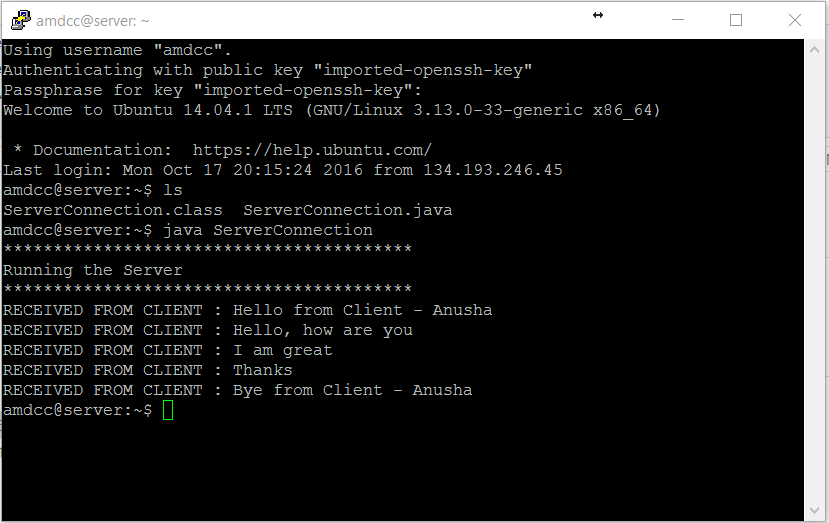


**Terminating connection between server and client:**

**At client:**



**At server:**



**(b) A client sends a text file (> 10KB) to a server.**

**Server prints the file on the screen,**

**Server saves the file in a local system,**

**Server appends one more line (eg. ‘This is an added line from a server) to the file,**

**And send the updated file back to the client.**

**Client shows the file on the screen after it fully receives the file.**

Part b: Client – Server Communication using UDP (User Datagram Protocol) used for Transfer and Receive file:

1. Enable the Client and Server window by using the Putty file with user ID’s and logging into host numbers which are obtained in the details while creating the slice as mentioned above.

2. Insert the programs required for the transfer of the file from Client to Server using the command vi filename.java

3. Here, we have used “FileTransferFromClient” and “FileTransferFromServer” for client and server respectively.

4. Compile the programs using the command “javac filename.java”.

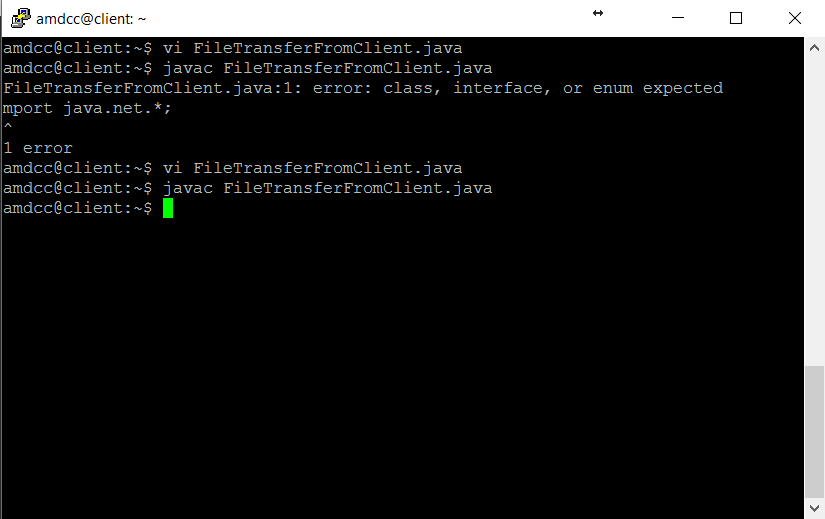
5. Create a text file in the client window with content.

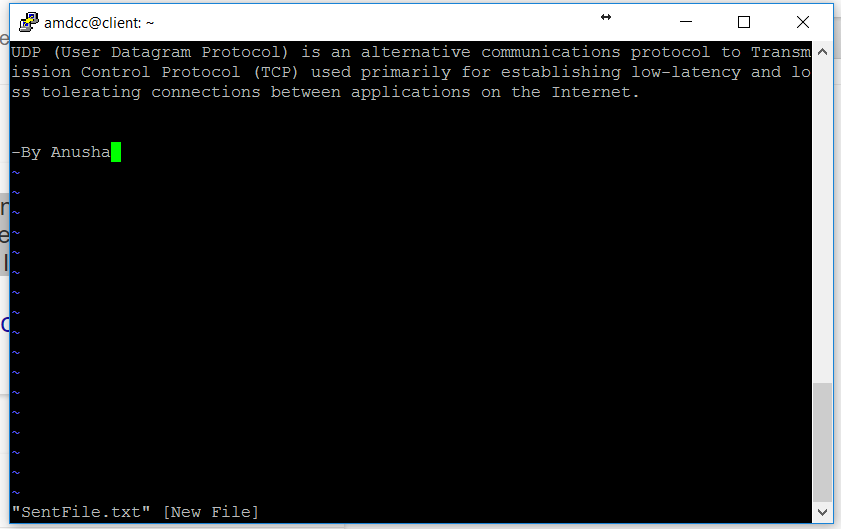
6. Create a plain text file in the server window.

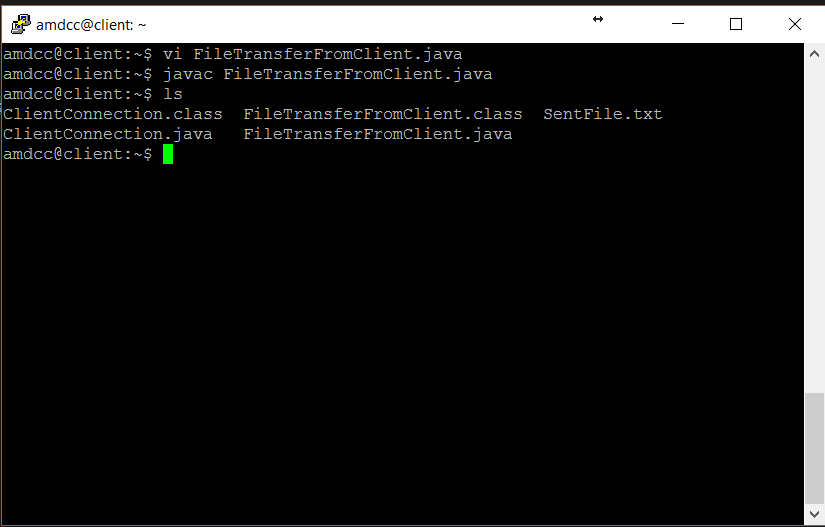
7. Run the class files using the command “java classfilename” in server first and then in client.

8. This transfers the file from client to server and the server appends an extra line to the file and sends it to the client which is displayed on the client window.

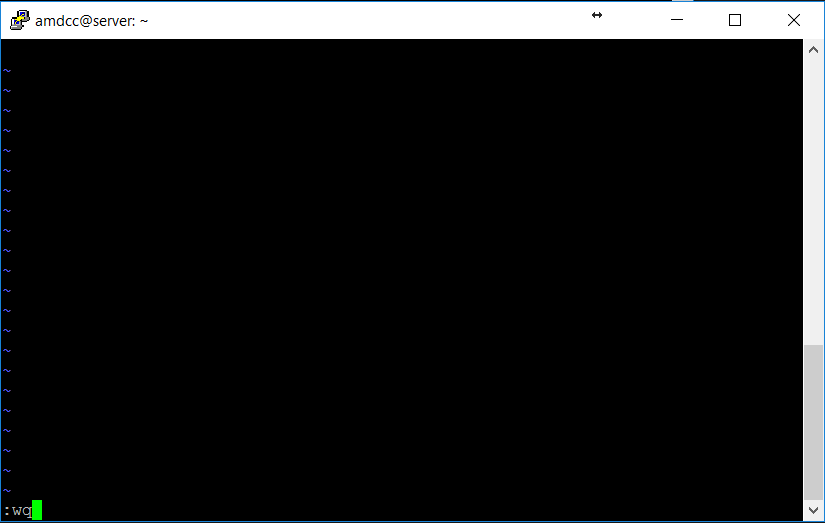
**At client side:**

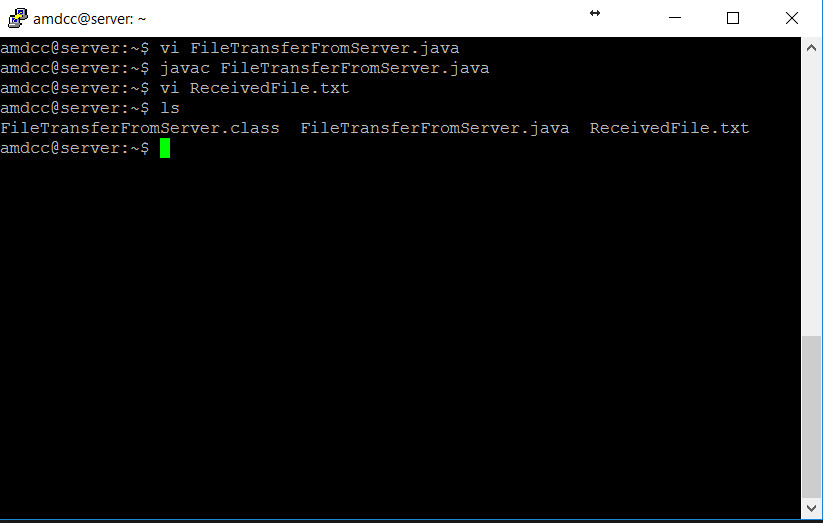


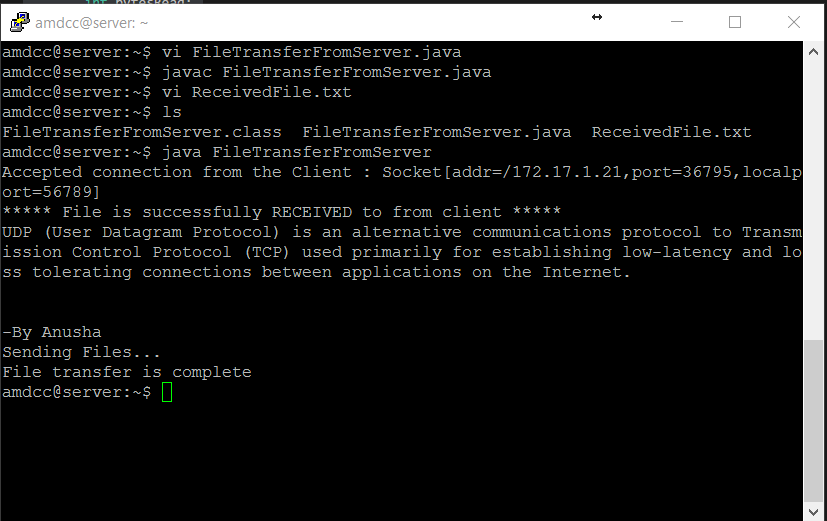


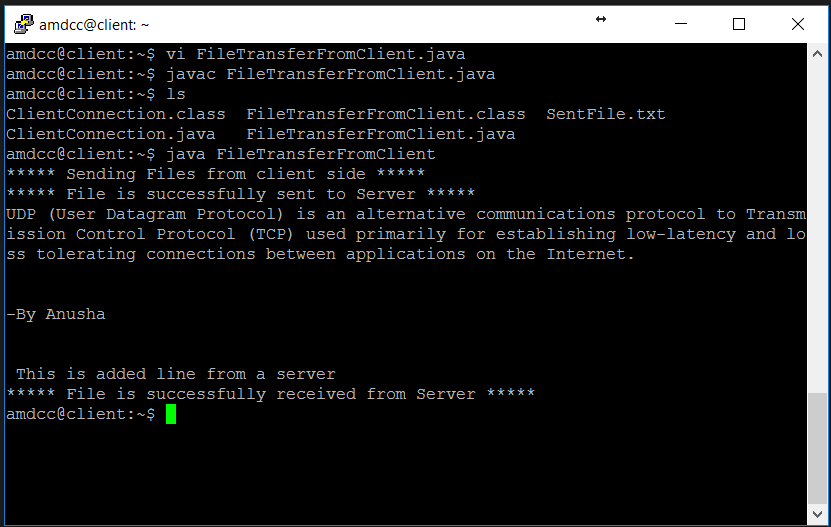


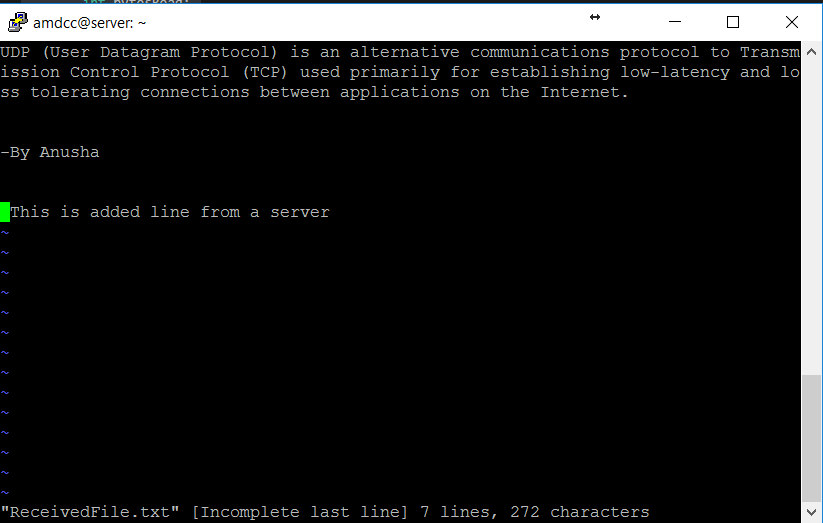
**At server side:**

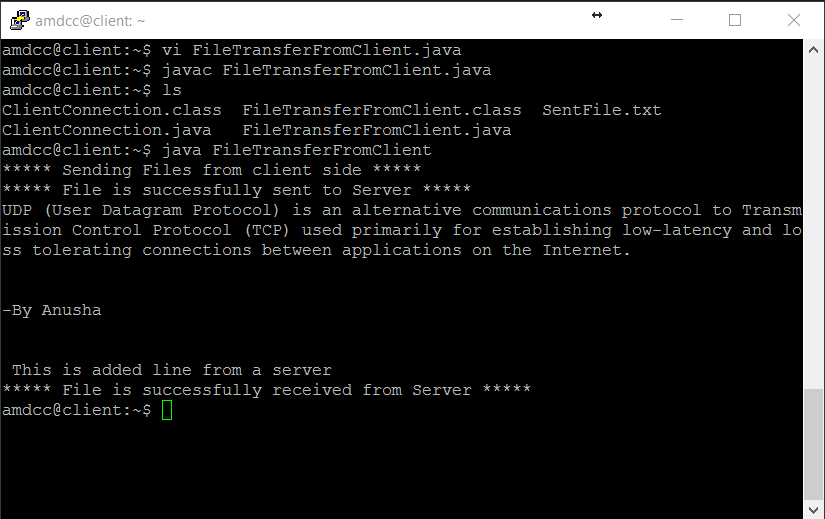












The Server appended a line and sent the file back to the client which is displayed on the client window stating that it received the file which was been sent by the client in previous steps.