

# Network Architecture-1

## Project-1

Fall 2016



Submitted by:

Moulika Chadalavada (16234180)

## 1. Introduction

This project mainly aims on working with GENI. TCP Client and Server programs are developed to send messages and file transfer using GENI.

**Software's Used:** GENI, Putty, FileZilla, Java

## 2. GENI and Putty

GENI (Global Environment for Network Innovations) provides a virtual laboratory for networking and distributed systems research and education. It is well suited for exploring networks at scale, thereby promoting innovations in network science, security, services and applications.

Initially account is created in GENI and added into Project. SSH keys are downloaded from GENI for authentication.

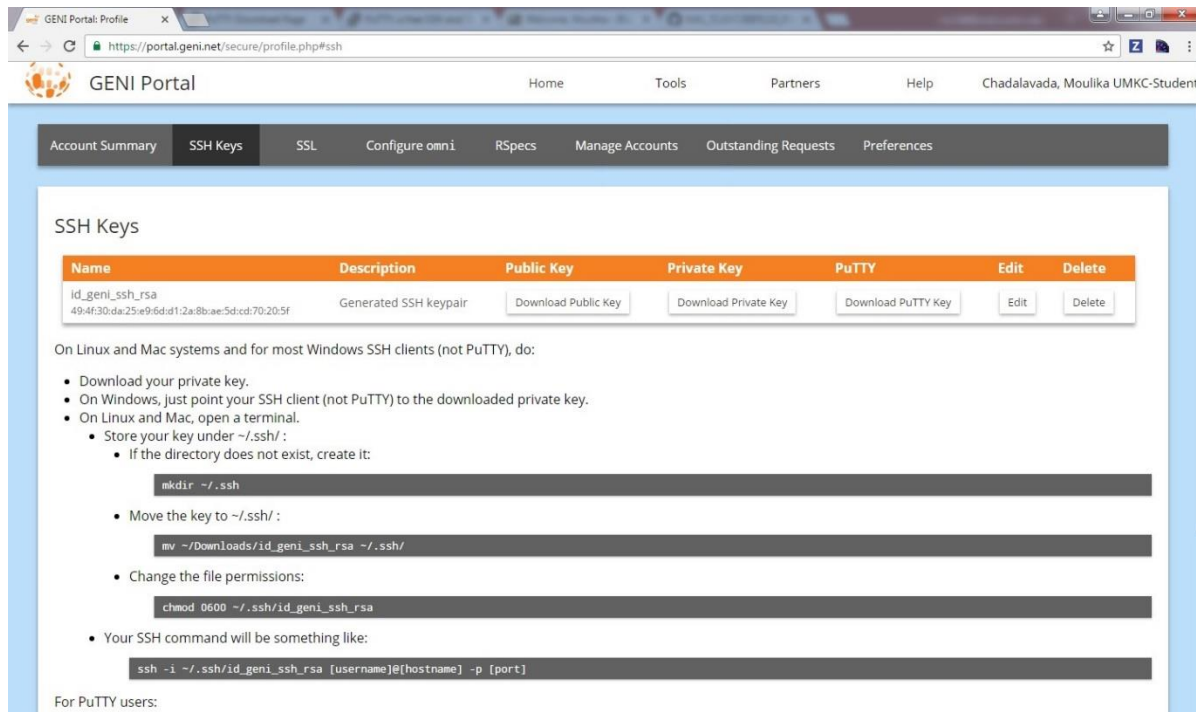


Fig 1: SSH Keys

Once added into project Slice '**Project-NA-1**' is created two resources **Client** and **Server**, and established connection between the resources using Link.

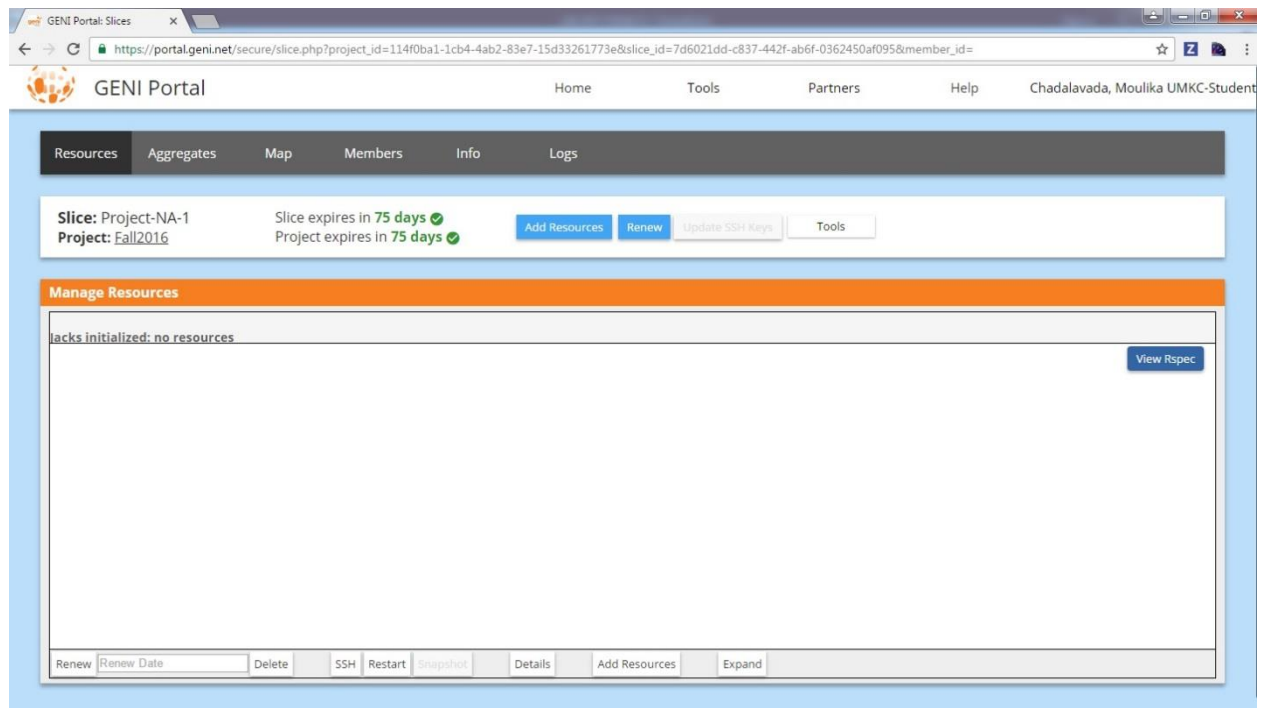


Fig 2: Slice creation

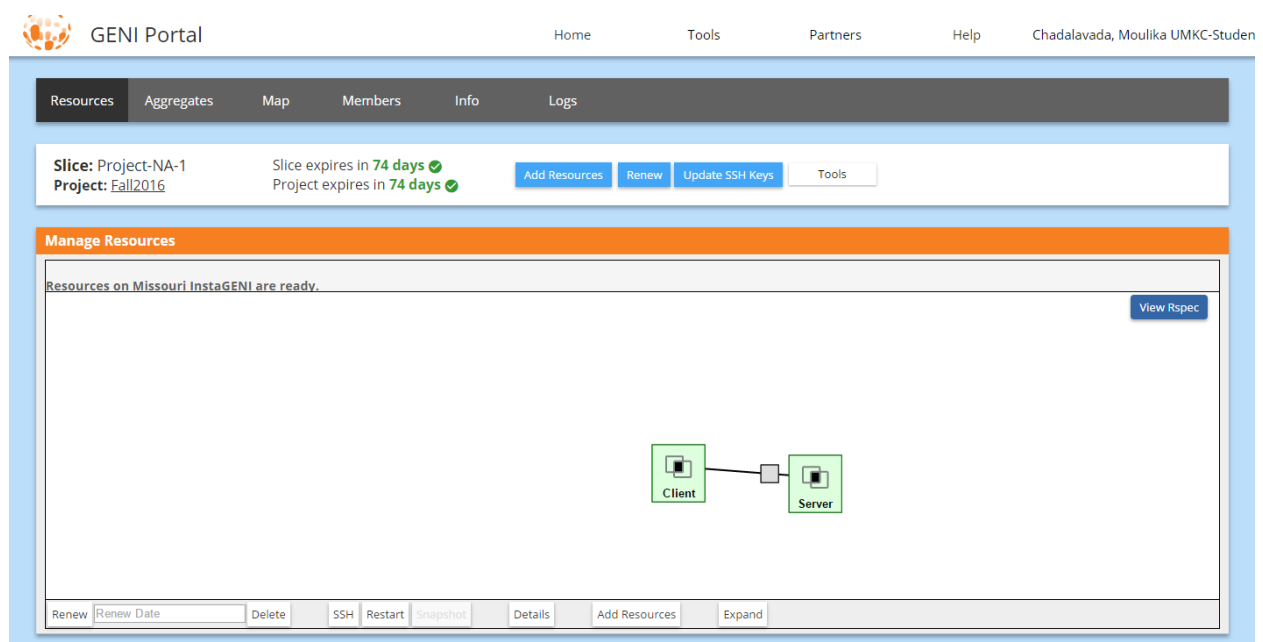


Fig 3: 2 Resources in Slice

The screenshot shows the GENI Portal interface. At the top, there is a navigation bar with links: Home, Tools, Partners, Help, and Chadalavada, Moulika UMKC-Student. Below this is a secondary navigation bar with links: Resources, Aggregates, Map, Members, Info, and Logs. The main content area displays information for 'Slice: Project-NA-1' and 'Project: Fall2016'. It indicates that the slice expires in 75 days and the project expires in 75 days, both with green checkmarks. There are buttons for 'Add Resources', 'Renew', 'Update SSH Keys', and 'Tools'. Below this is a section titled 'Manage Resources' with a sub-header 'Resources on Missouri InstaGENI are ready.' and a 'View Rspec' button. A modal window is open showing details for a 'Client' resource. The modal includes a 'Name' field with the value 'Client', an 'SSH to' field with a list of IP addresses, a 'Node Type' dropdown set to 'Other...', and a 'Hardware Type' dropdown set to 'default-vm'. At the bottom of the modal, there are buttons for 'Renew', 'Renew Date', 'Delete', 'SSH', 'Restart', 'Snapshot', 'Details', 'Add Resources', and 'Expand'. A diagram in the background shows a 'Client' node connected to a 'Server' node.

Fig 4: Client details

The screenshot shows the GENI Portal interface, similar to the previous one. The main content area displays information for 'Slice: Project-NA-1' and 'Project: Fall2016'. It indicates that the slice expires in 75 days and the project expires in 75 days, both with green checkmarks. There are buttons for 'Add Resources', 'Renew', 'Update SSH Keys', and 'Tools'. Below this is a section titled 'Manage Resources' with a sub-header 'Resources on Missouri InstaGENI are ready.' and a 'View Rspec' button. A modal window is open showing details for a 'Server' resource. The modal includes a 'Name' field with the value 'Server', an 'SSH to' field with a list of IP addresses, a 'Node Type' dropdown set to 'Other...', and a 'Hardware Type' dropdown set to 'default-vm'. At the bottom of the modal, there are buttons for 'Renew', 'Renew Date', 'Delete', 'SSH', 'Restart', 'Snapshot', 'Details', 'Add Resources', and 'Expand'. A diagram in the background shows a 'Client' node connected to a 'Server' node.

Fig 5: Server Details

Home → Project Fall2016 → Slice Project-NA-1 → Resources on Slice Project-NA-1

Resources on slice: Project-NA-1

Queried 1 of 1 aggregates.

[Refresh All Details](#) [Refresh Status Only](#)

Status	Aggregate
READY	Missouri InstaGENI

Aggregate Missouri InstaGENI's Resources:

Node #1:

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	Client	pc3	2016-12-30T23:59:59.000Z	default-vm	Client.Project-NA-1.ch-geni-net.instageni.rnet.missouri.edu

Login

```
ssh hhcc77@pc3.instageni.rnet.missouri.edu -p 39738
ssh chciyb@pc3.instageni.rnet.missouri.edu -p 39738
ssh rzc6d@pc3.instageni.rnet.missouri.edu -p 39738
ssh mc7d@pc3.instageni.rnet.missouri.edu -p 39738
```

Interfaces

Interface	MAC	Layer 3
interface-1	pc3:1a0	02b07e4f9750

Node #2:

Status	Client ID	Component ID	Expiration	Type	Hostname
READY	Server	pc3	2016-12-30T23:59:59.000Z	default-vm	Server.Project-NA-1.ch-geni-net.instageni.rnet.missouri.edu

Login

```
ssh hhcc77@pc3.instageni.rnet.missouri.edu -p 39739
ssh chciyb@pc3.instageni.rnet.missouri.edu -p 39739
ssh rzc6d@pc3.instageni.rnet.missouri.edu -p 39739
ssh mc7d@pc3.instageni.rnet.missouri.edu -p 39739
```

Interfaces

Interface	MAC	Layer 3
interface-0	pc3:1a0	02feaf5f604

Link #1:

Client ID	Endpoint #0	Endpoint #1
link-0	interface-0	interface-1

Fig 6: Resource Details

Once the resources are created in Slice and SSH keys are downloaded, in **puttygen** private key is generated using Putty key are is downloaded from GENI.



Fig 7: Putty Key Generator

Once Private key is generated in putty, logged in to two VM's for Client and Server with its respective IP Address and Port Number. Before logging in Private key is loaded under SSH -> Auth as shown below in Fig 9.

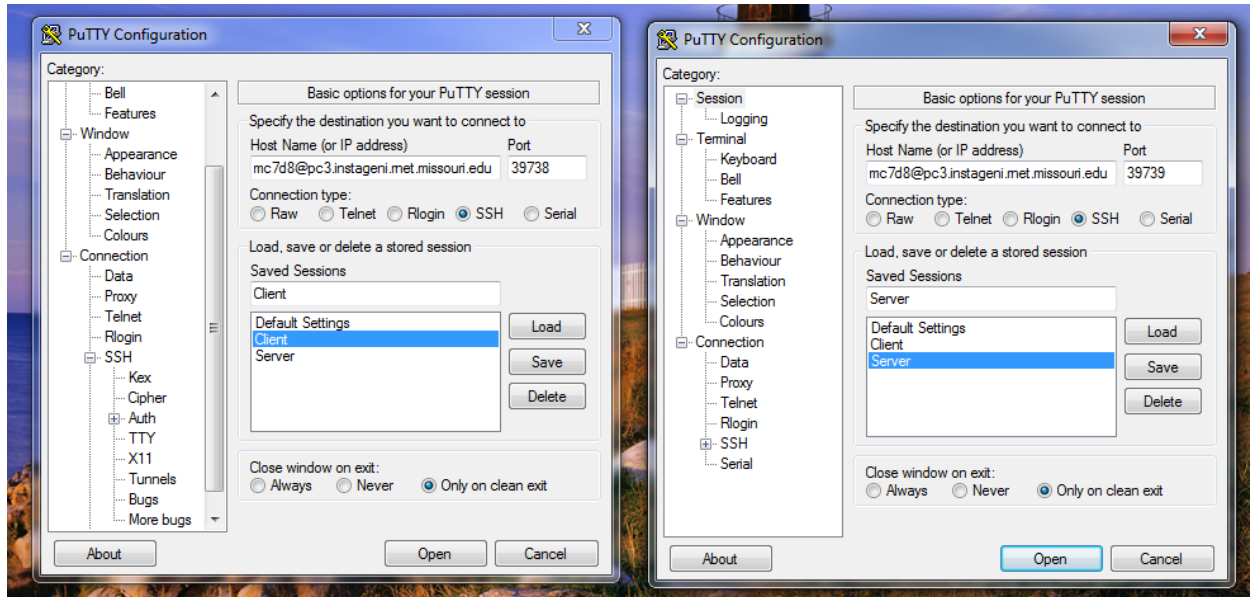


Fig 8: Putty Client and Server Login

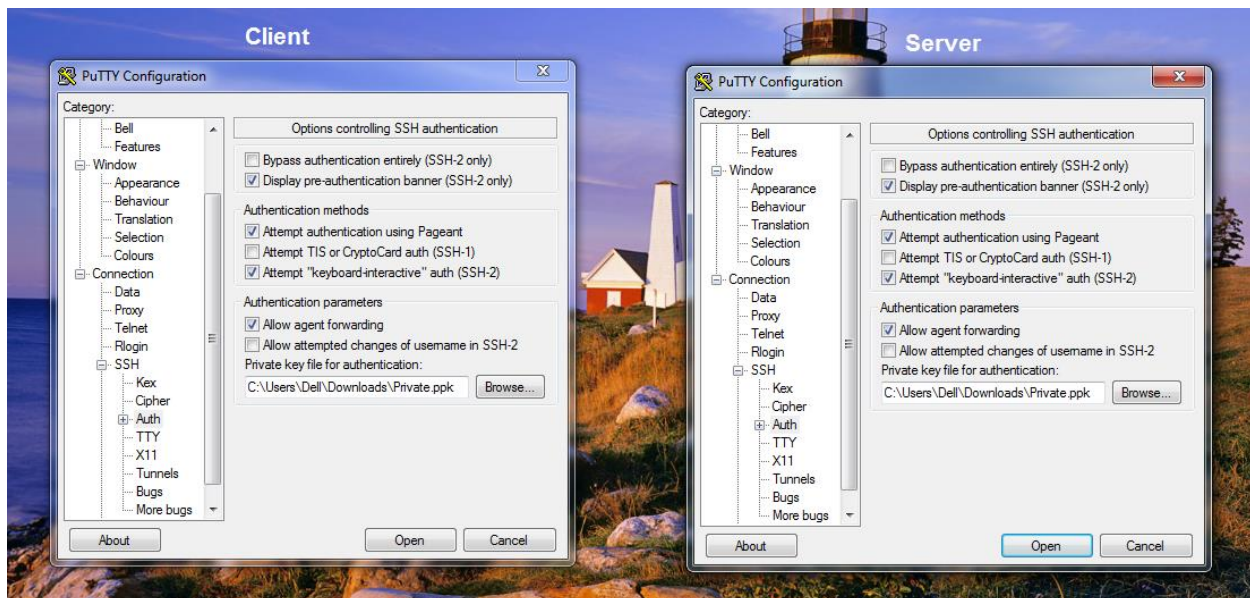


Fig 9: Client and Server Private Key Loading



Once private key is loaded and credentials are provided session is logged in successfully.

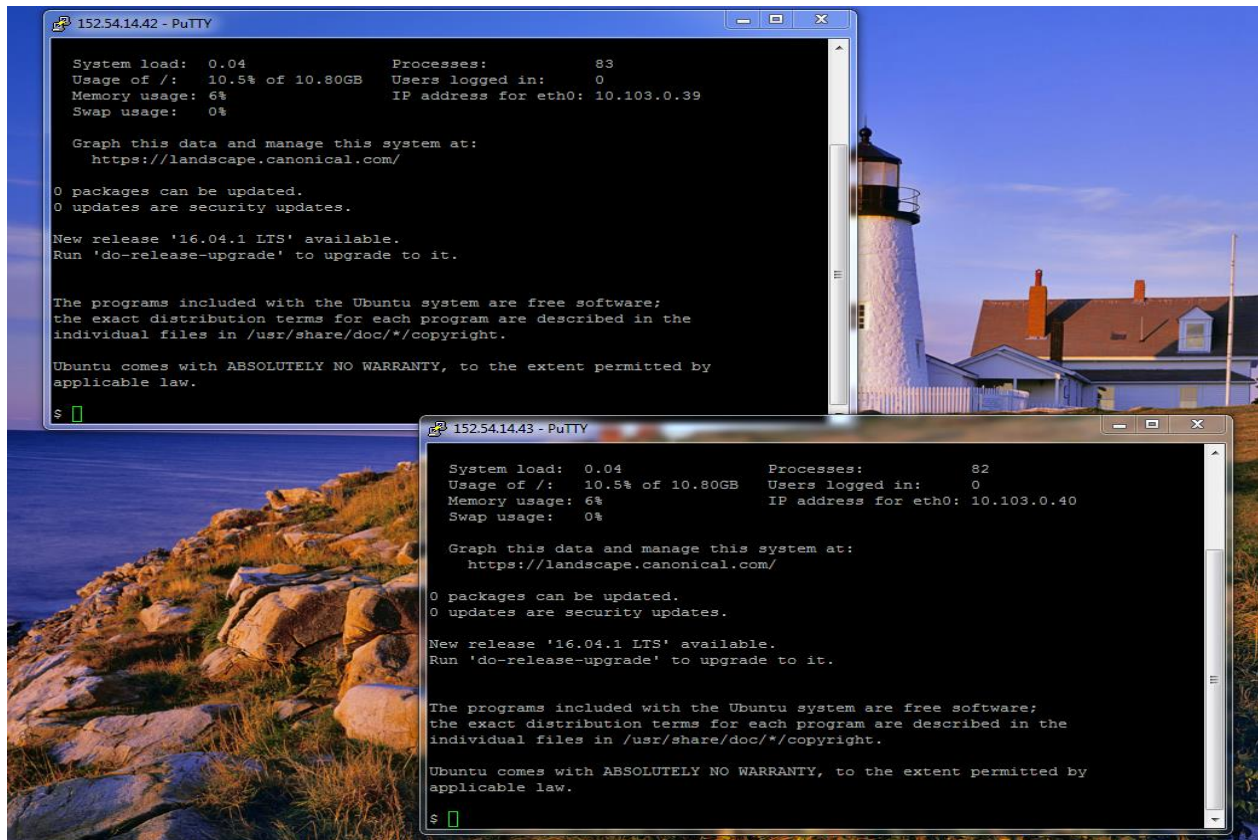


Fig 10: Server and Client Login

**FileZilla** Client is a fast and reliable cross-platform FTP, FTPS and SFTP client with lots of useful features and an intuitive graphical user interface. Using FileZilla the file structure on Client and Server can be viewed.

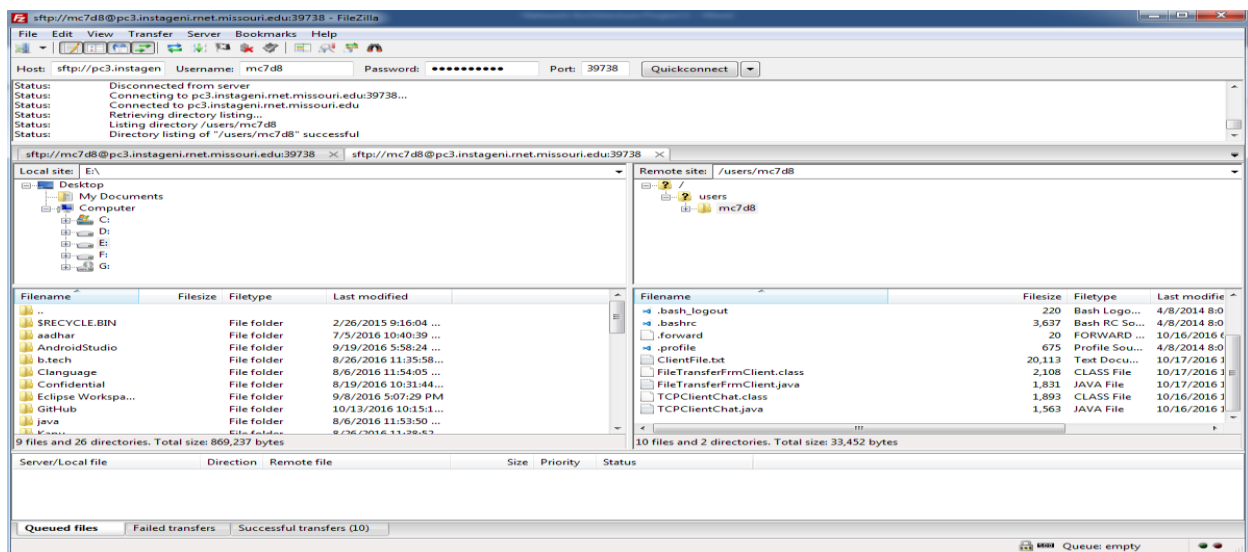


Fig 11: FileZilla

### 3. Requirements

Develop and deploy a simple TCP client and server programs on GENI. Show the screenshots of simple message exchanges.

#### 3.1 Question-1

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'.

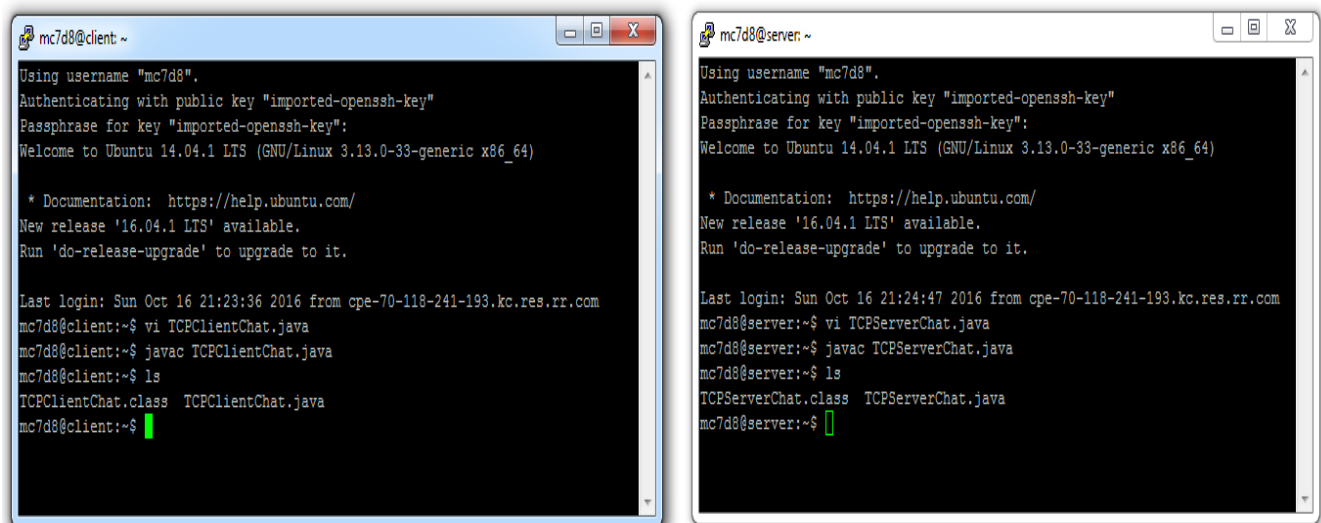
#### Solution:

In this query connection is built between client and server, messages are sent from Client to Server and vice-versa.

TCPClientChat.java file is created on Client in which the connection is made to Server using

```
clientSocket= new Socket("Server.Project-NA-1.ch-geni-net.instageni.rnet.missouri.edu", 12789);
```

TCPServerChat.java file is created which receives messages from Client and responds back.



```
mc7d8@client:~
Using username "mc7d8".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-33-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:23:36 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@client:~$ vi TCPClientChat.java
mc7d8@client:~$ javac TCPClientChat.java
mc7d8@client:~$ ls
TCPClientChat.class  TCPClientChat.java
mc7d8@client:~$
```

```
mc7d8@server:~
Using username "mc7d8".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-33-generic x86_64)

 * Documentation:  https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:24:47 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@server:~$ vi TCPServerChat.java
mc7d8@server:~$ javac TCPServerChat.java
mc7d8@server:~$ ls
TCPServerChat.class  TCPServerChat.java
mc7d8@server:~$
```

Fig 12: Client & Server Program Execution

Once the java files are compiled and the class file is executed, Client and Server connections are started as shown below



```

mc7d8@client:~
Using username "mc7d8".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-33-generic x86_64)

 * Documentation: https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:23:36 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@client:~$ vi TCPClientChat.java
mc7d8@client:~$ javac TCPClientChat.java
mc7d8@client:~$ ls
TCPClientChat.class  TCPClientChat.java
mc7d8@client:~$ java TCPClientChat
Client Started Running
*****
Enter message:
█

mc7d8@server:~
Using username "mc7d8".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-33-generic x86_64)

 * Documentation: https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:24:47 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@server:~$ vi TCPServerChat.java
mc7d8@server:~$ javac TCPServerChat.java
mc7d8@server:~$ ls
TCPServerChat.class  TCPServerChat.java
mc7d8@server:~$ java TCPServerChat
Server Started Running
*****
█

```

Fig 13: Client &amp; Server Connection Started

Client sends message, 'Hello from Client- Moulika Chadalavada' and server receives the message as shown below.

```

mc7d8@client:~
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-33-generic x86_64)

 * Documentation: https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:23:36 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@client:~$ vi TCPClientChat.java
mc7d8@client:~$ javac TCPClientChat.java
mc7d8@client:~$ ls
TCPClientChat.class  TCPClientChat.java
mc7d8@client:~$ java TCPClientChat
Client Started Running
*****
Enter message:
Hello from Client- Moulika Chadalavada
Message from Server: Hello from Server- Moulika Chadalavada
Enter message :
Hello, How are you??
Message from Server: Hello, How are you??
Enter message :
Hello from Client- Subhash
Message from Server: Hello from Server- Subhash
Enter message :
█

mc7d8@server:~
Using username "mc7d8".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-33-generic x86_64)

 * Documentation: https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:24:47 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@server:~$ vi TCPServerChat.java
mc7d8@server:~$ javac TCPServerChat.java
mc7d8@server:~$ ls
TCPServerChat.class  TCPServerChat.java
mc7d8@server:~$ java TCPServerChat
Server Started Running
*****
Message Received from Client : Hello from Client- Moulika Chadalavada
Hello Moulika Chadalavada
Message Received from Client : Hello, How are you??
Message Received from Client : Hello from Client- Subhash
Hello Subhash
█

```

Fig 14: Client &amp; Server Chat

Once chat between Client and Server is completed, Clients sends 'Bye from Client- Moulika Chadalavada' which is received from Server, and server sends back 'Bye from Server- Moulika Chadalavada' where the connection between Client and Server gets terminated.

```

mc7d8@client:~
* Documentation: https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:23:36 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@client:~$ vi TCPClientChat.java
mc7d8@client:~$ javac TCPClientChat.java
mc7d8@client:~$ ls
TCPClientChat.class  TCPClientChat.java
mc7d8@client:~$ java TCPClientChat
Client Started Running
*****
Enter message:
Hello from Client- Moulika Chadalavada
Message from Server: Hello from Server- Moulika Chadalavada
Enter message :
Hello, How are you??
Message from Server: Hello, How are you??
Enter message :
Hello from Client- Subhash
Message from Server: Hello from Server- Subhash
Enter message :
Bye from Client- Moulika Chadalavada
Message from Server: Bye from Server- Moulika Chadalavada
mc7d8@client:~$

mc7d8@server:~
Using username "mc7d8".
Authenticating with public key "imported-openssh-key"
Passphrase for key "imported-openssh-key":
Welcome to Ubuntu 14.04.1 LTS (GNU/Linux 3.13.0-33-generic x86_64)

* Documentation: https://help.ubuntu.com/
New release '16.04.1 LTS' available.
Run 'do-release-upgrade' to upgrade to it.

Last login: Sun Oct 16 21:24:47 2016 from cpe-70-118-241-193.kc.res.rr.com
mc7d8@server:~$ vi TCPServerChat.java
mc7d8@server:~$ javac TCPServerChat.java
mc7d8@server:~$ ls
TCPServerChat.class  TCPServerChat.java
mc7d8@server:~$ java TCPServerChat
Server Started Running
*****
Message Received from Client : Hello from Client- Moulika Chadalavada
Hello Moulika Chadalavada
Message Received from Client : Hello, How are you??
Message Received from Client : Hello from Client- Subhash
Hello Subhash
Message Received from Client : Bye from Client- Moulika Chadalavada
mc7d8@server:~$

```

Fig 15: End Client &amp; Server Connection

### 3.2 Question-2

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 (e.g. '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.

#### Solution:

In this part of query, the file has to be transferred from Client to Server which is >10KB and when a line is entered in Server the same has to be appended in Client file.

Initially **ClientFile.txt** (Client) and **ServerFile.txt** (Server) are created. In ClientFile.txt the text is copied and saved. In server '**ServerFileTransfer.java**' is executed where Server connection is built and receives file from Client and stores it in 'ServerFile.txt' (Fig 15), also if line is entered in Server the same is sent to Client. client '**FileTransferFrmClient.java**' is executed which build connection with server and transfers file to Server. In return the line received from Server is appended to 'ClientFile.txt'. (Fig 16)

```

mc7d8@server:~
mc7d8@server:~$ vi ServerFile.txt
mc7d8@server:~$ vi ServerFileTransfer.java
mc7d8@server:~$ javac ServerFileTransfer.java
mc7d8@server:~$ ls
ServerFileTransfer.class  ServerFile.txt  TCPServerChat.java
ServerFileTransfer.java  TCPServerChat.class
mc7d8@server:~$ java ServerFileTransfer
Server Started Running
*****

mc7d8@client:~
mc7d8@client:~$ vi ClientFile.txt
mc7d8@client:~$ vi FileTransferFrmClient.java
mc7d8@client:~$ javac FileTransferFrmClient.java
mc7d8@client:~$ ls
ClientFile.txt  FileTransferFrmClient.java  TCPClientChat.java
FileTransferFrmClient.class  TCPClientChat.class
mc7d8@client:~$

```

Fig 16: Server &amp; Client connection established

```

mc7d8@client:~$ ls
ClientFile.txt      FileTransferFrmClient.java  TCPClientChat.java
FileTransferFrmClient.class  TCPClientChat.class
mc7d8@client:~$ clear
mc7d8@client:~$ ls
ClientFile.txt      FileTransferFrmClient.java  TCPClientChat.java
FileTransferFrmClient.class  TCPClientChat.class
mc7d8@client:~$ java FileTransferFrmClient
Client Started Running
*****
----- Sending File from Client -----
***** File is sent successfully to Server *****

INTRODUCTION
Hello World is often used by developers to familiarize themselves with new concepts by building a simple program. This tutorial aims to achieve a similar purpose by getting practitioners started with Hadoop and HDP. We will use an Internet of Things (IoT) use case to build your first HDP application.

This tutorial describes how to refine data for a Trucking IoT Data Discovery (aka IoT Discovery) use case using the Hortonworks Data Platform. The IoT Discovery use cases involves vehicles, devices and people moving across a map or similar surface. Your analysis is targeted to linking location information with your analytic data.

For our tutorial we are looking at a use case where we have a truck fleet. Each truck has been equipped to log location and event data. These events are streamed back to a datacenter where we will be processing the data. The company wants to use this data to better understand risk.

Here is the video of Analyzing Geolocation Data to show you what you'll be doing in this tutorial.

PRE-REQUISITES:
Downloaded and Installed Hortonworks Sandbox
Before entering hello HDP labs, we highly recommend you go through Learning the Ropes of the Hortonworks Sandbox to become familiar with the Sandbox in a VM and the Ambari Interface.
Data Set Used: Geolocation.zip
Optional: Hortonworks ODBC driver installed and configured - see the tutorial on installing the ODBC driver for Windows or OS X. Refer to Installing and Configuring the Hortonworks ODBC driver on Windows 7 Installing and Configuring the Hortonworks ODBC driver on Mac OS X
In this tutorial, the Hortonworks Sandbox is installed on an Oracle VirtualBox virtual machine (VM) - your screens may be different.

TUTORIAL OVERVIEW
In this tutorial, we will provide the collected geolocation and truck data. We will import this data into HDFS and build derived tables in Hive. Then we will process the data using Pig, Hive and Spark. The processed data is then visualized using Apache Zeppelin.

To refine and analyze Geolocation data, we will:
Review some Hadoop Fundamentals

```

Fig 17: Sending File from Client to Server

```

mc7d8@client:~$
Pig fits in through its data flow strengths where it takes on the tasks of bringing data into Apache Hadoop and working with it to get it into the form for querying.
A good overview of how this works is in Alan Gates posting on the Yahoo Developer blog titled Pig and Hive at Yahoo!.
From a technical point of view, both Pig and Hive are feature complete, so you can do tasks in either tool.
However, you will find one tool or the other will be preferred by the different groups that have to use Apache Hadoop.
The good part is they have a choice and both tools work together.

OUR DATA PROCESSING TASK
We are going to do the same data processing task as we just did with Pig in the previous tutorial.
We have several files of truck driver statistics and we are going to bring them into Hive and do some simple computing with them.
We are going to compute the sum of hours and miles logged driven by a truck driver for an year.

Once we have the sum of hours and miles logged, we will extend the script to translate a driver id field into the name of the drivers by joining two different tables.

This is line added from server
***** File successfully received from Server *****
mc7d8@client:~$

mc7d8@server:~$
g SQL for querying data.
Pig fits in through its data flow strengths where it takes on the tasks of bringing data into Apache Hadoop and working with it to get it into the form for querying.
A good overview of how this works is in Alan Gates posting on the Yahoo Developer blog titled Pig and Hive at Yahoo!.
From a technical point of view, both Pig and Hive are feature complete, so you can do tasks in either tool.
However, you will find one tool or the other will be preferred by the different groups that have to use Apache Hadoop.
The good part is they have a choice and both tools work together.

OUR DATA PROCESSING TASK
We are going to do the same data processing task as we just did with Pig in the previous tutorial.
We have several files of truck driver statistics and we are going to bring them into Hive and do some simple computing with them.
We are going to compute the sum of hours and miles logged driven by a truck driver for an year.

Once we have the sum of hours and miles logged, we will extend the script to translate a driver id field into the name of the drivers by joining two different tables.

Sending File from Server...
File transfer completed successfully
mc7d8@server:~$

```

Fig 18: File Appending in Client

Initially blank ServerFile.txt is created in Server, but once File transfer process is completed the text from ClientFile.txt is copied to Server as shown below.

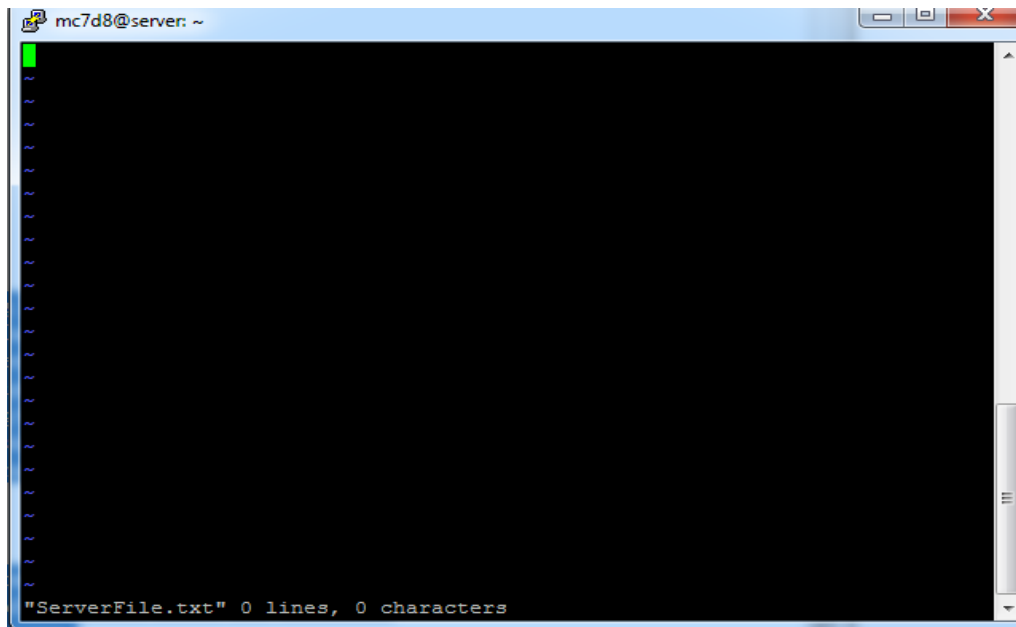


Fig 19: Blank Server File

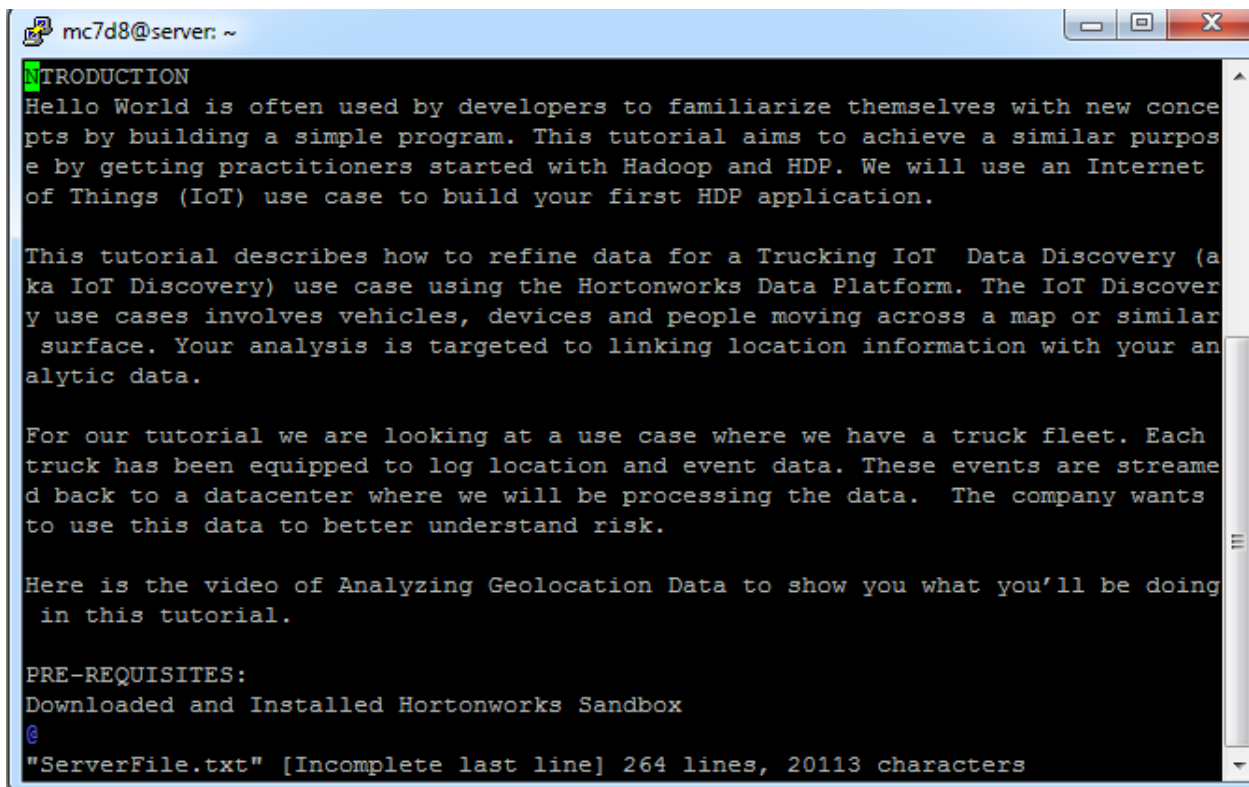


Fig 20: ServerFile.txt after file transfer

## 4. References

<https://portal.geni.net/secure/dashboard.php>

<http://stackoverflow.com/>

<http://www.putty.org/>

<https://filezilla-project.org/>