**Server Documentation**

## Step 1: Setting up Servers on VCENTER

Download vSphere Client & Install from a web browser

Use the IP address: 10.10.7.30

Chris has the username and password for the Client. He won’t give the credential out to students.

Once we are in, we are able to view/configure servers that have been created.

As of yet nothing has been created.

Following a series of steps I was able to created and deployed a Virtual Linux Machine (Ubuntu 14.04 LTS) as the 1st Server.

Ram: 4GB

## Step 2: Install Prerequisite Packages

Once the installation of the server completed, I navigated over to the find bar so I can open up a terminal.

I issued the following commands in the terminal.

sudo apt-get update – updates the repositories of the system

sudo apt-get upgrade – installs newer versions packages for the system

sudo apt-get install openssh - install ssh on the server. With this installed you will be able to access your server with programs like Putty. I will cover this later in the documentation.

sudo apt-get install x11vnc – install vnc if you wish to access your server with a GUI.

## Step 3: Network Requirements

Once the system repositories are up to date, with the help of the Update and Upgrade commands, we can go ahead to configure and setup the network requirements for the servers. I assigned an IP address, Subnet mask, gateway, and DNS 1 & 2 to the 1st server that I created.

Chris cloned the server each student had created into the desired number of servers that each group wanted. He also gave each IT student a range of IP address they can use for their servers. range of IP addresses that was assigned to me is 10.10.7.160 – 165. We only have Two Servers, one for the Database and application. I chose IP address 10.10.7.161 and 162 ans assigned it to the 2 Servers.

Server 1 (**Database**) (**Application**) has an IP address: 10.10.7.161, Subnet Mask: 255.255.255.0, Gateway: 10.10.7.1, DNS 10.12.1.11, DNS 10.12.1.12.

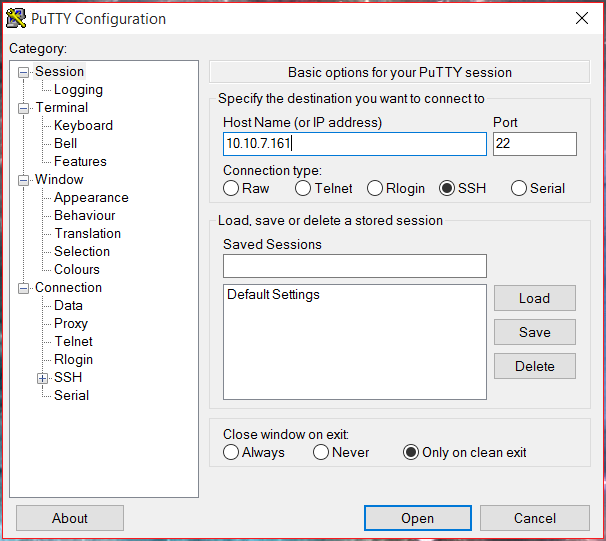
After getting the server configured and running, Chris clone 2 more Servers, an Application server and a backup server. The IP address changes for the 2 new servers, but the rest of the network requirements information stays the same.

Server 2 (**Backup**) has an IP address: 10.10.7.162, Subnet Mask: 255.255.255.0, Gateway: 10.10.7.1, DNS 10.12.1.11, DNS 10.12.1.12.

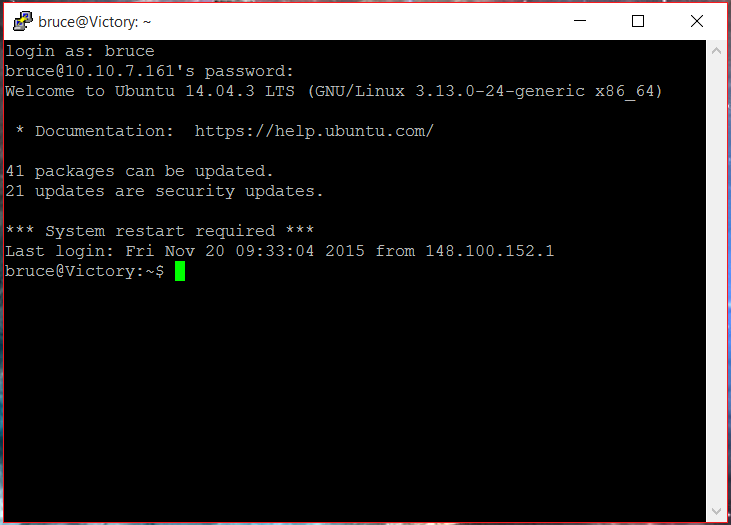
We first started with 3 servers but ended up using 2 servers. The group agreed to have both the Database and Application (front-end) run on one server and have another server to backup the data if an unexpected outcome surfaces. The reason why we had a turnaround was due to a few problems we ran into with our Database Server. The migration to 2 servers was a blessing although we had to recreate our database on the Application server. There are a lot of pros when it comes to hosting both the Database and Application on the same server. There is a better turn around time, very unlikely to have lost of failure, and reduction of traffic and improving response time. Connecting our Database to the Application and having them work hand in hand is far better and efficient on the same server than the decision to have them on 2 separate servers.

## Step 6: Install Putty on PC

I had to install Putty on my personal machine. Putty is a program that helps a user get access to an operating system (OS) they created through a command-line interface terminal. They can do so with the help from the IP address assigned to the OS, and the default port number of 22.



Putty will then open up a command-line interface terminal that will ask for a username and password that was setup at the beginning of the creation of the server. Once the user inputs this information they will be granted access to the servers terminal.



(10.10.7.161)

**Database Server**

## Step 1: Install PostgreSQL

**(PostgreSQl is an open source relational database management system.)**

## The following commands will install PostgreSQL on the server. I did this to each server.

Ensuring the list of available packages is up to date before installing anything new.

apt-get -y update

Then it’s a matter of just running one command for installation via apt-get:

apt-get -y install postgresql postgresql-contrib

Let’s switch into that system user:

su – postgres

And then connect to the PostgreSQL terminal (in the postgres role):

psql

To get out of the PostgreSQl terminal

\q

## https://www.digitalocean.com/community/tutorials/how-to-install-and-use-postgresql-on-ubuntu-14-04

## Step 2: Install Git

**(Git is an open source controlled system designed to handle small to larger projects.)**

## The following commands will install Git on the server. I did this to each server.

Ensuring the list of available packages is up to date before installing anything new.

sudo apt-get update

sudo apt-get install build-essential libssl-dev libcurl4-gnutls-dev libexpat1-dev gettext unzip

Back on your Ubuntu 14.04 server, you can type wget and follow it by pasting the address you copied.

The URL that you copied may be different from mine:

wget https://github.com/weonyuan/Herby/archive/master.zip -o git.zip

Unzip the file that you downloaded and move into the resulting directory by typing:

unzip git .zip

cd git-\*

Now, you can make the package and install it by typing these two commands:

make prefix=/user/local all

sudo make prefix=/usr/local install

Now that you have git installed, if you want to upgrade to a later version, you can simply clone the repository and then build and install:

git clone github-windows://openRepo/https://github.com/weonyuan/Herby

This will create a new directory within your current directory where you can rebuild the package and reinstall the newer version, just like you did above. This will overwrite your older version with the new version:

make prefix=/user/local all

sudo make prefix=/usr/local install

How To SetUp Git

git config –-global user.name “Your Name”

git config –-global user.name “youremail@domain.com”

We can see all of the configuration items that have been set by typing:

git config –-list

**git configuration**

**user.name=Your Name**

**user.email=youremail@domain.com**

As you can see, this has a slightly different format. The information is stored in the configuration file, which you can optionally edit by hand with your text editor like this:

Nano ~/.gitconfig

**~/.gitconfig contents**

**[user]**

**name = Your Name**

**email = youremail@domain.com**

## <https://www.digitalocean.com/community/tutorials/how-to-install-git-on-ubuntu-14-04>

**Application Server**

## Step 1: Install Node

**(Node.js is a Javascript platform for server-side programming that allows users to build network applications quickly.)**

sudo apt-get update  
sudo apt-get install nodejs

sudo apt-get install npm

## Step 2: Install Nginx

**(Nginx is a Web Server that is great with performance and low memory usage.)**

Since this is our first interaction with the apt packaging system in this session, we should update our local package index before we begin so that we are using the most up-to-date information. Afterwards, we will install nginx:

sudo apt-get update  
sudo apt-get install nginx

## Start Nginx Service

sudo service nginx start

## Step 3: Install Curl

**(Curl is a command line tool for transferring data with URL syntax.)**

You can access the default Nginx landing page to confirm that the software is running properly by visiting your server's domain name or public IP address in your web browser, but first we have to install curl.

sudo apt-get install curl

<https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-14-04-lts>

## Step 3: Check / Connect to Nginx on the Web

Enter the following commands

**ip addr show eth0 | grep inet | awk '{ print $2; }' | sed 's/\/.\*$//'**

**curl http://icanhazip.com**

**When you have your Server's IP address or domain, enter it into your browser's address bar:**

**http://10.10.7.161**

## Stop Nginx Service

sudo service nginx stop

We can make sure that our web server will restart automatically when the server is rebooted by typing:

sudo update-rc.d nginx defaults

This should already be enabled by default, so you may see a message like this:

System start/stop links for /etc/init.d/nginx already exist.

https://www.digitalocean.com/community/tutorials/how-to-install-nginx-on-ubuntu-14-04-lts

(10.10.7.162)

**Backup Server**

## Step 1: Install RSnapShots

sudo apt-get update  
sudo apt-get install rsnapshots

In order to backup another server, your backup server will need to be able to connect through SSH to the server you wish to back up

To generate a public and private key run this command on your backup server:

sudo ssh-keygen -t rsa

We need to make sure we can SSH to another server from the Backup Server.

**sudo apt-get install sshpass**

**sshpass -p password ssh bruce@10.10.7.161**

**exit**

## Step 2: Configuring RSnapShots

After you have installed rsnapshot, you will need to edit the configuration file.

sudo nano /etc/rsnapshot.conf

Once in the configuration file you can move up with F7 and down with F8. Find the variable for each directory and make sure it corresponds to what’s below, if not edit it to match the variables.

**snapshot\_root /backup/**

**cmd\_ssh /usr/bin/ssh**

**cmd\_du /usr/bin/du**

We set these up under the “BACKUP INTERVALS” section of the configuration.

**#########################################  
# BACKUP INTERVALS #  
# Must be unique and in ascending order #  
# i.e. hourly, daily, weekly, etc. #  
#########################################  
  
retain hourly 6  
retain daily 7  
retain weekly 4  
retain monthly 3**

Another item that may need to be edited is the ssh\_args variable and backup home directory. **ssh\_args -p 25000**

**backup /home/ localhost/  
backup /etc/ localhost/**

**backup bruce@10.10.7.161:/home/ remote-droplet/**

## Step 3: Testing the Configuration

sudo rsnapshot configtest

Enter the commands below to set when you want to backup snapshot to be taken.

sudo rsnapshot -t daily

sudo rsnapshot daily

## Step 4: Automating the Process

sudo nano /etc/cron.d/rsnapshot

We’re going to remove the “#” character from the beginning of the scheduling section to activate these values.

**# This is a sample cron file for rsnapshot.  
# The values used correspond to the examples in /etc/rsnapshot.conf.  
# There you can also set the backup points and many other things.  
#  
# To activate this cron file you have to uncomment the lines below.  
# Feel free to adapt it to your needs.  
  
0 \*/4 \* \* \* root /usr/bin/rsnapshot hourly  
30 3 \* \* \* root /usr/bin/rsnapshot daily  
0 3 \* \* 1 root /usr/bin/rsnapshot weekly  
30 2 1 \* \* root /usr/bin/rsnapshot monthly**

These settings will run add a snapshot to the “hourly” directory within our “/backup/” directory every four hours, add a daily snapshot everyday at 3:30 am, add a “weekly” snapshot every Monday at 3:00 am, and add a “monthly” snapshot on the first of every month at 2:30 am.

## Step 5: Restoring the files from the Backup Server

Login to Backup.nixcraft.net.in, enter:

# ssh bruce@10.10.7.162.nixcraft.net.in

Cd to your rsnapshot directory:

# cd /.raid/rsnapshot

To list current snapshots, enter:

# ls -l

The daily backups for server1 is stored at /.raid/rsnapshot/daily.\*/server1/ directory. To see latest daily backup (daily.0), enter:

You can restore backups to server1 using the rsync as follows to restore /var/www/html/:

# cd /.raid/rsnapshot/daily.0/server1/var/www/html/

# rsync -avr \* [user@server1.nixcraft.net.in](mailto:user@server1.nixcraft.net.in):/var/www/html/

OR use the scp command to copy selected file such as /var/www/html/db.conf.php: