

Installing a Distributed Database Server - MySQL Active Passive Replication

Contents

Step 1. Create an Ubuntu 22.04 Virtual Machine in Oracle VirtualBox	2
Step 2: Configure your Network Interface Cards (NICs)	9
Step 3 (Optional). Install a secure shell server to be able to remotely connect to your VM:	14
Step 4: Install MySQL on the primary DBMS server.....	15
Step 5: Configure the Active Primary MySQL Server	18
Step 6: Clone your Ubuntu VM to Make a Secondary Ubuntu VM.....	19
Step 7: Configure and Active Passive MySQL Cluster	20
Step 8: Test the Configuration	36
Step 9 (Optional): Synchronizing A Replication Database Out of Sync	37

Step 1. Create an Ubuntu 22.04 Virtual Machine in Oracle VirtualBox

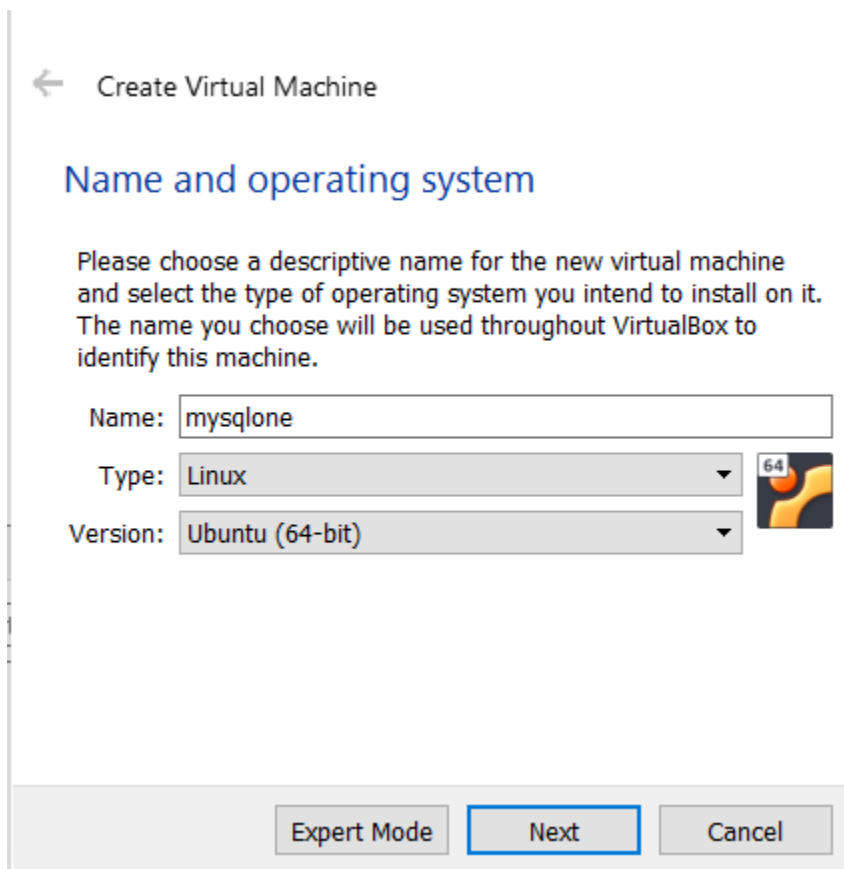
In this lab we are going to create a MySQL distributed database environment using two (2) Ubuntu Virtual Machines (VMs). While only a small prototype, the lab helps build skills required for distributed computing beyond databases, from networking to system administration.

The first step is to download Oracle VirtualBox: <https://www.virtualbox.org/>

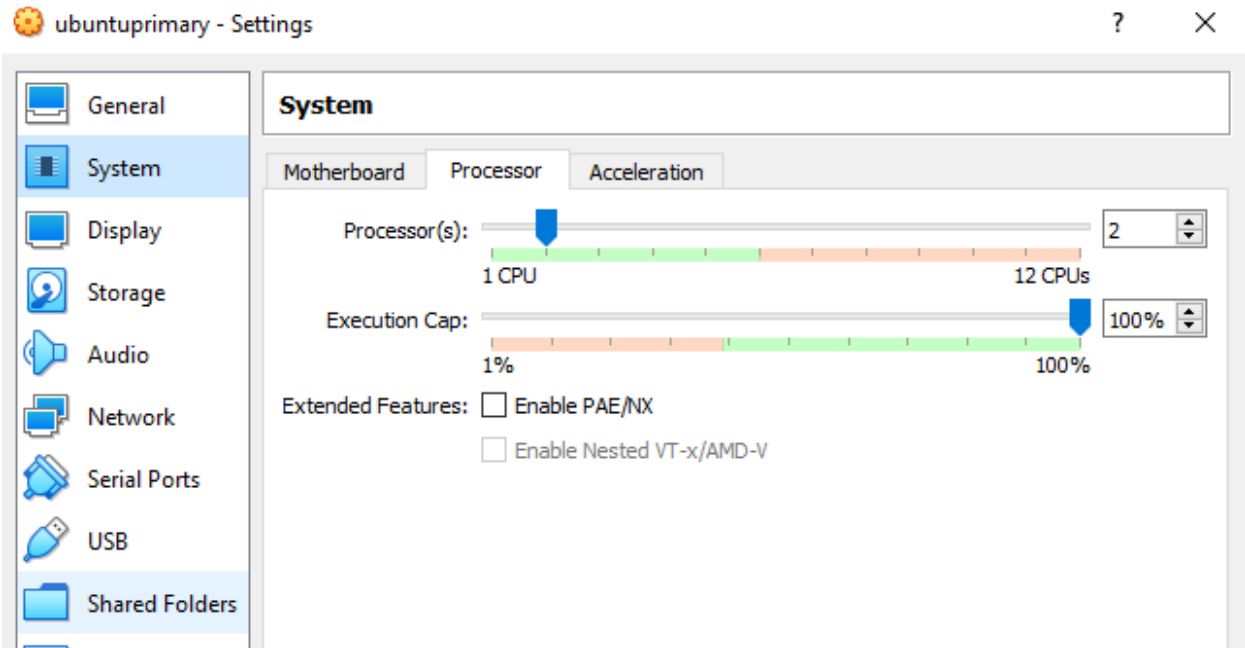
Next, download Ubuntu 22.04.1 LTS - <https://ubuntu.com/download/server>

Download the manual server installation for **Ubuntu Server Option 2.**

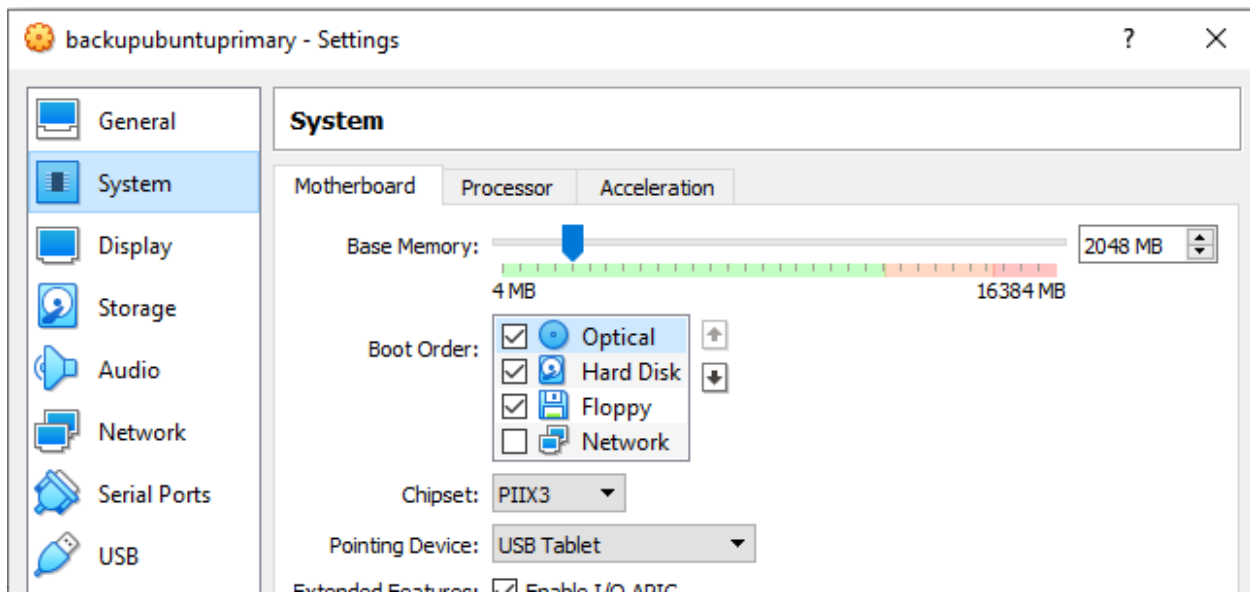
Create a new VM in Oracle VirtualBox with 2 GBs of memory or more and 2 CPU cores.



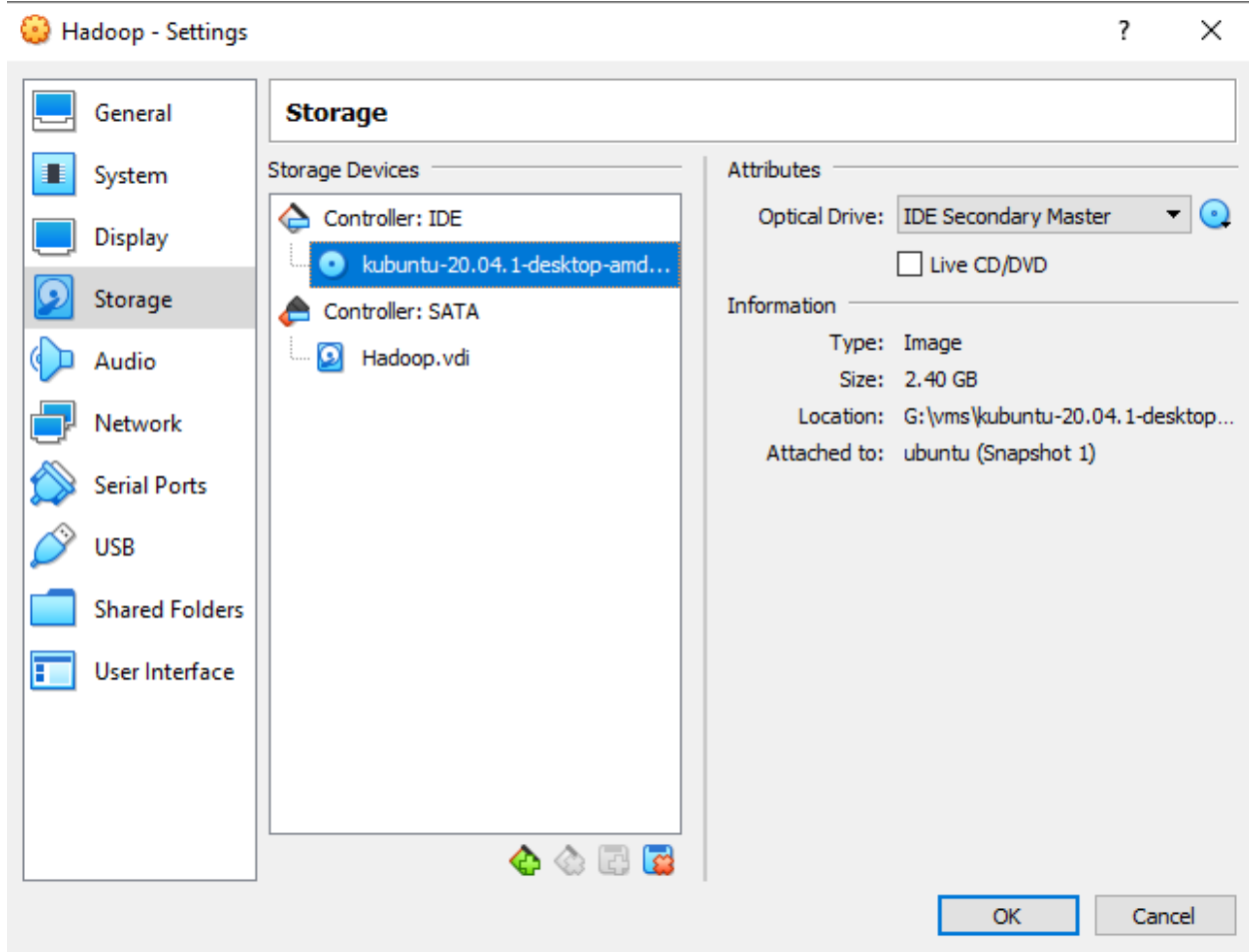
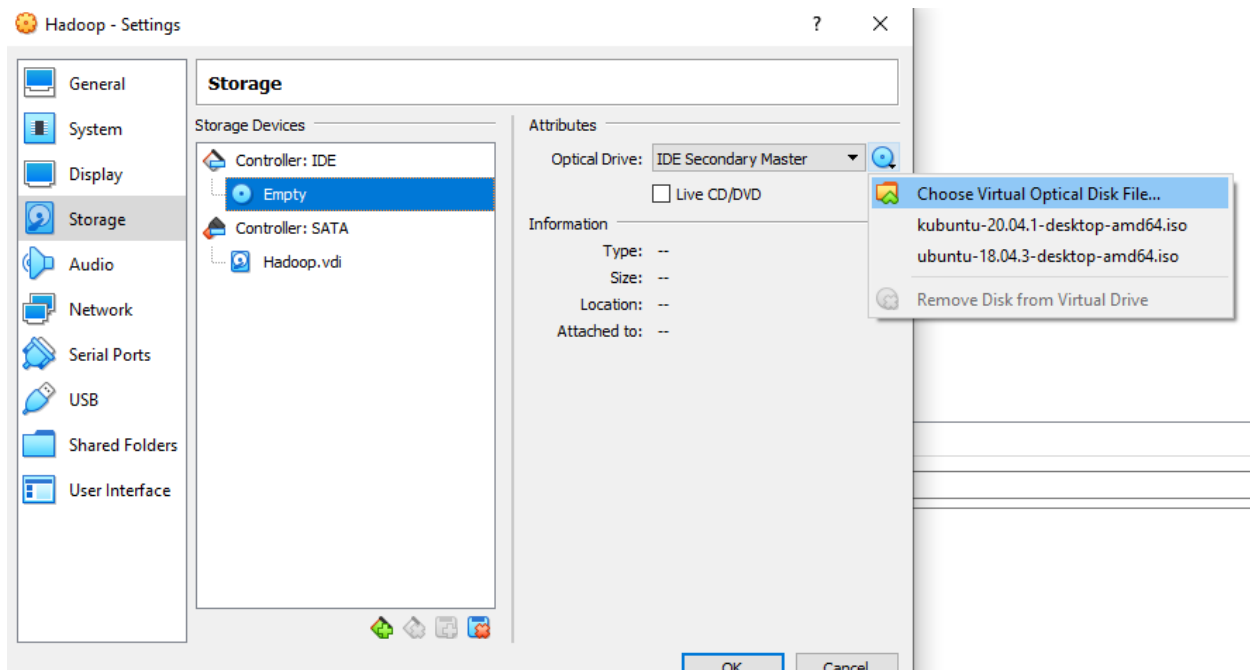
Make sure the Type is Linux and Version is Ubuntu (64-bit). Click next, next, next and leave the defaults for all other steps when creating the new virtual machine. Once you are finished, go to the settings of the VM.



Make sure your CD / optical is set to boot first.

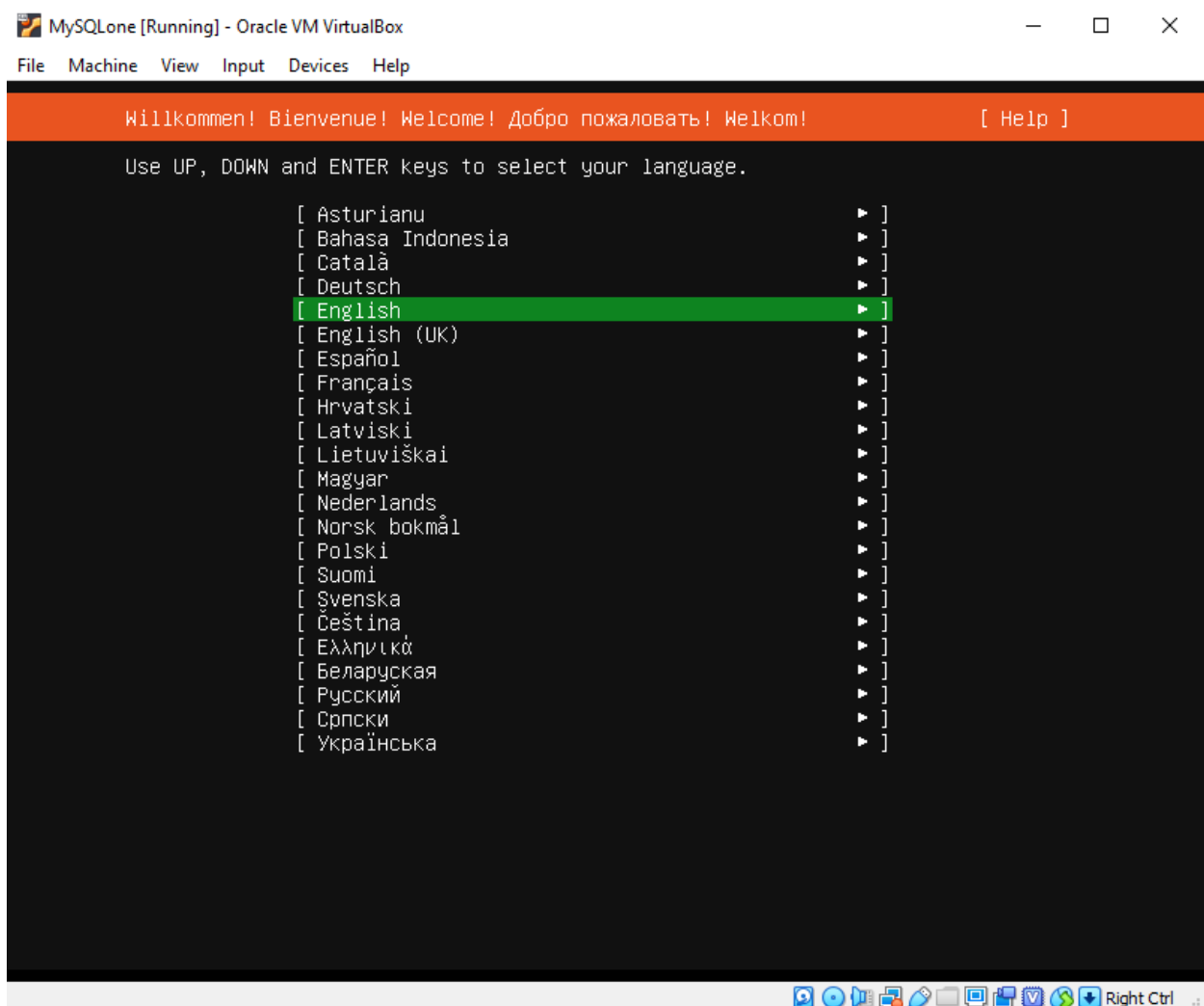


Next, go to storage and select your Ubuntu 20.04 ISO image:

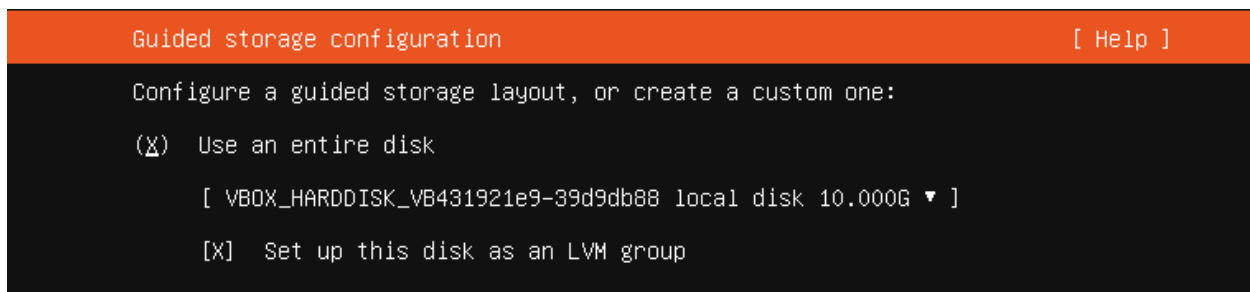


Start the VM and the Ubuntu installation should start. You can leave all the settings as default during the Ubuntu 20 installation. It will start and installation wizard, simply keep everything the same and proceed through the entire installation.

The installation should look something like this:



Keep hitting enter, leaving everything the same. For the storage configuration, use the entire disk.



You can use your TAB key on the keyboard to tab between options. When it asks if you want to confirm the destructive action and/or create the new disk partition, click yes or continue.

Storage configuration [Help]

FILE SYSTEM SUMMARY

MOUNT POINT	SIZE	TYPE	DEVICE TYPE
[/	8.996G	new ext4	new LVM logical volume ▶]
[/boot	1.000G	new ext4	new partition of local disk ▶]

AVAILABLE DEVICES

Confirm destructive action

Selecting Continue below will begin the installation process and result in the loss of data on the disks selected to be formatted.

You will not be able to return to this or a previous screen once the installation has started.

Are you sure you want to continue?

[No]

[Continue]

Use a simple password that you will not forget! Use your First Name and Last Name when setting up your username, like this (replace Amy Smith with your real name):

Profile setup [Help]

Enter the username and password you will use to log in to the system. You can configure SSH access on the next screen but a password is still needed for sudo.

Your name: Amy Smith

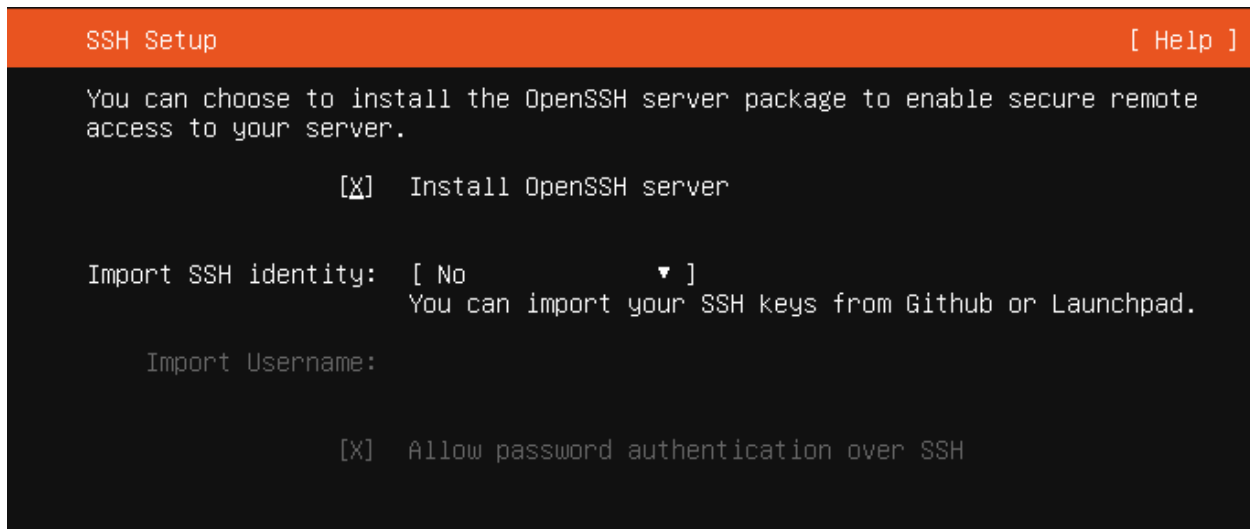
Your server's name: amysmithsqlone
The name it uses when it talks to other computers.

Pick a username: amysmith

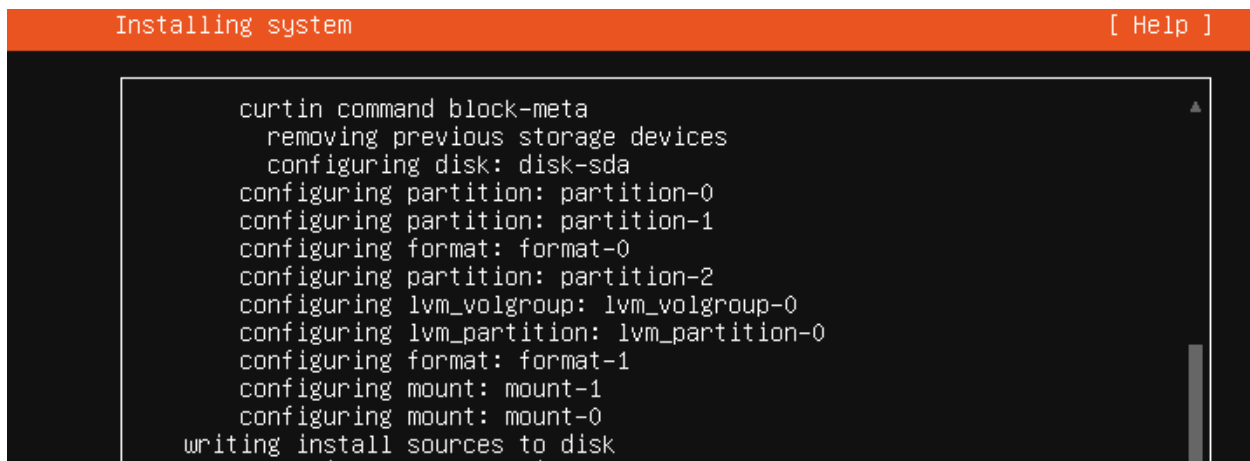
Choose a password: *****

Confirm your password: *****_

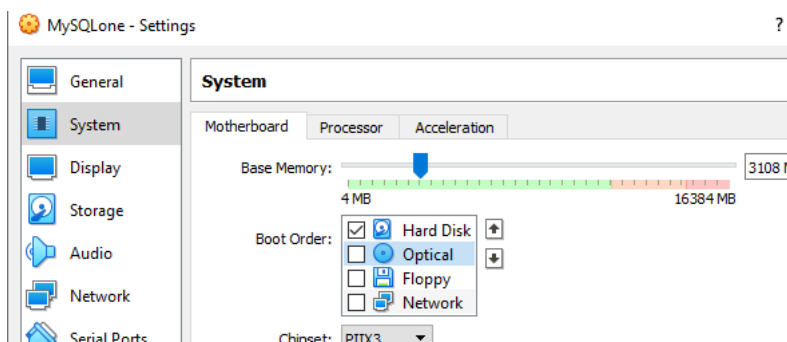
If asked to install SSH, click YES so that you do not have to install it manually:



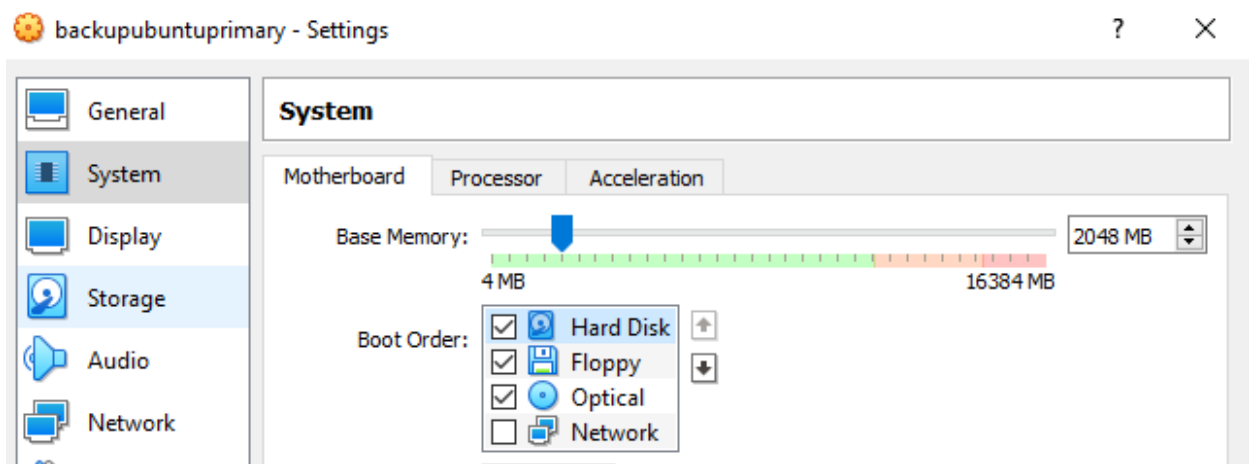
It may ask to install popular server features next, feel free to install other packages if necessary but you can leave this blank/empty as we are going to install MySQL separately. It should install the OS if everything was set as the default like this:



When the installation finishes, highlight the reboot now option to reboot your VM. Close your VM and **change your boot settings so the Hard Disk boots FIRST** rather than the CD:



Once installed, change the boot settings on the VM so that the hard disk boots first rather than the Optical/the Ubuntu install ISO. Otherwise, your VM will want to install the ISO again.



Start your new Ubuntu VM!

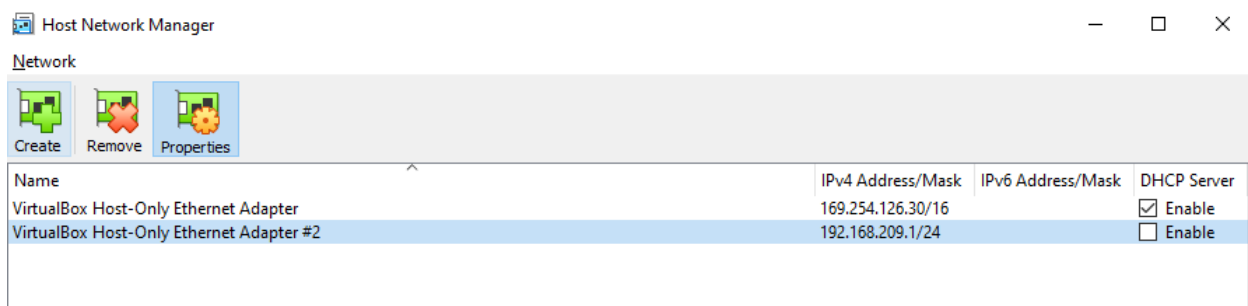
Step 2: Configure your Network Interface Cards (NICs)

A critical aspect of distributed database servers is ensuring they have IP addresses that allow them to communicate with each other. Virtual Box has two different network settings that you need to be aware of and set properly.

Host-Only mode is the setting we will be using when the distributed databases need to be connected.

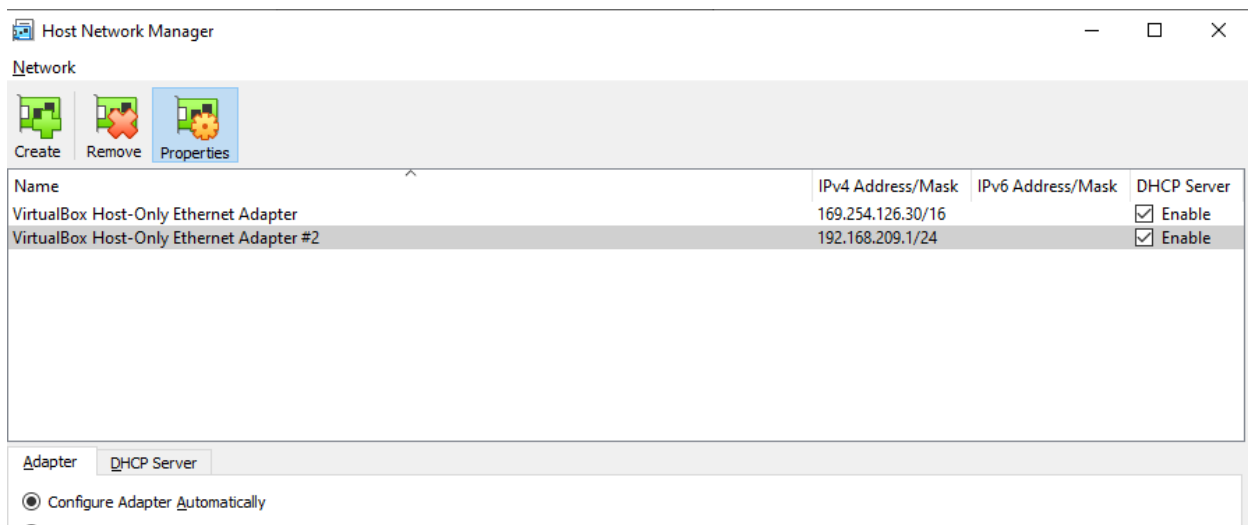
NAT or Bridged mode are settings necessary for your servers to connect to the Internet.

In VirtualBox click on File in the Menu and go to the Host Network Manager:



In host network manager, click “Create” to create a new NIC. It will ask you whether to give it Windows administrative privileges to create a new NIC, you need to answer YES to give it permissions to create the new NIC.

In the new NIC settings, set DHCP server to Enable. And, Configure Adapter Automatically. Your settings should mirror this screenshot:



Next, click on the DHCP server tab above “Configure Adapter Automatically”. Set your IP addresses using the 10.100.100.0/24 subnet. Mirror the settings in the screenshot:

Host Network Manager

Network

Create Remove Properties

Name	IPv4 Address/Mask	IPv6 Address/Mask	DHCP Server
VirtualBox Host-Only Ethernet Adapter	169.254.126.30/16		<input checked="" type="checkbox"/> Enable
VirtualBox Host-Only Ethernet Adapter #2	192.168.209.1/24		<input checked="" type="checkbox"/> Enable

Adapter DHCP Server

☒ Enable Server

Server Address: 10.100.100.1

Server Mask: 255.255.255.0

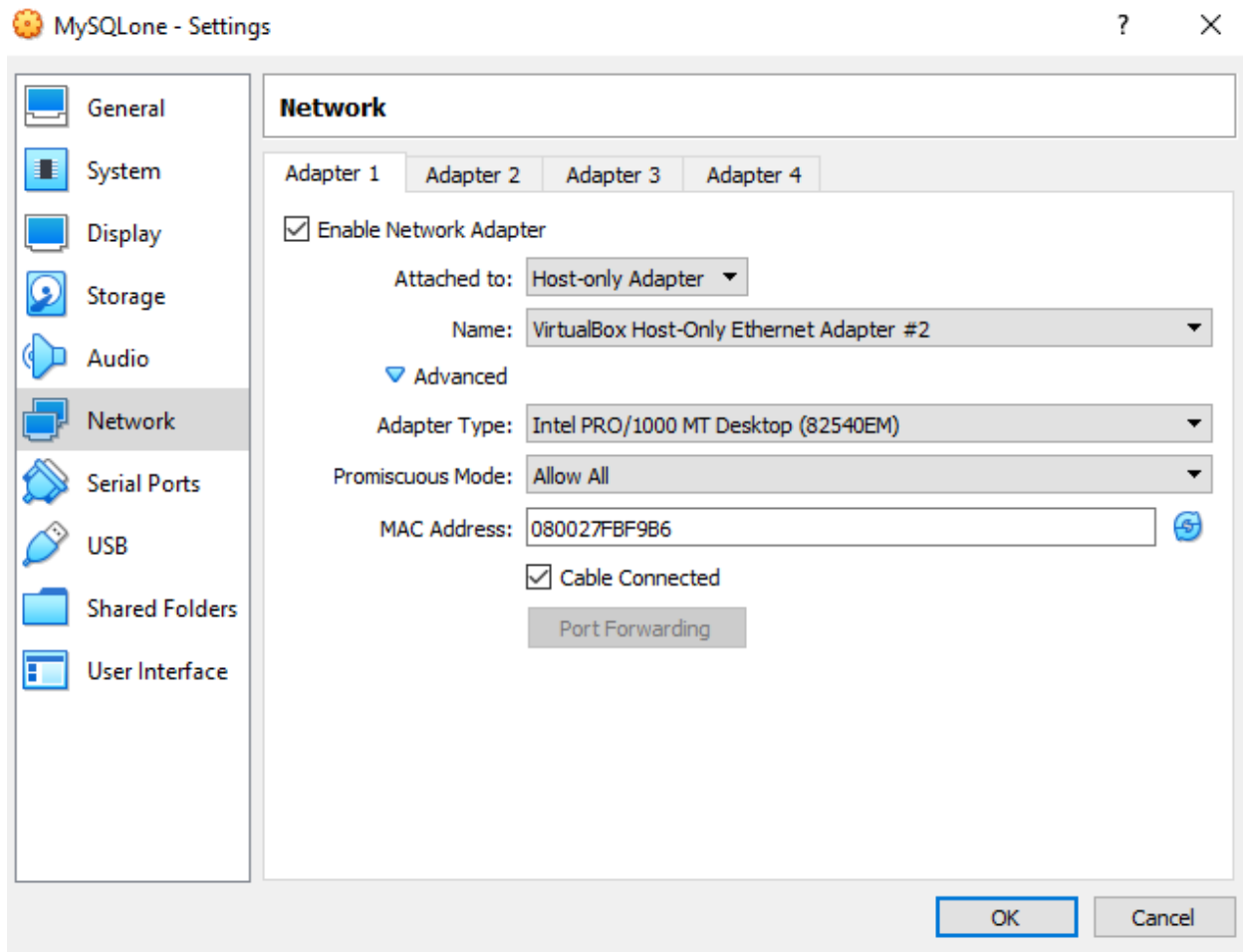
Lower Address Bound: 10.100.100.2

Upper Address Bound: 10.100.100.254

Reset Apply Close

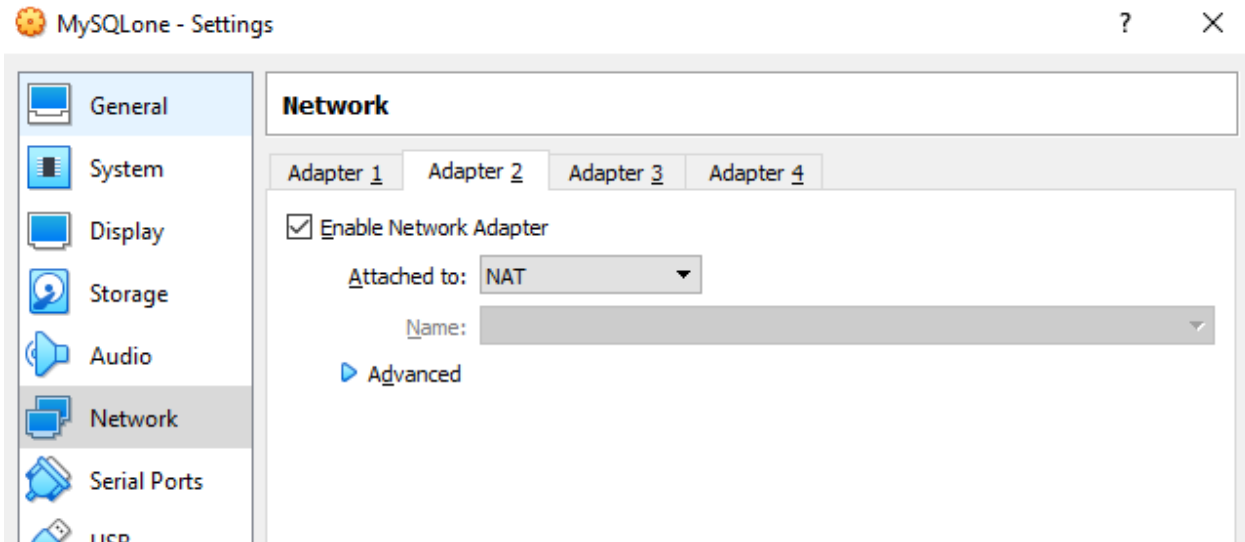
Apply the settings and verify by going back into Host Network Manager.

Next go the settings of your VM, go to Network, and change Adapter 1 to the new host only adapter you just created:



In the Adapter 1 screenshot you should see Adapter 1 is enabled, set to Host-only Adapter and Promiscuous Mode is set to “Allow All” to allow the MySQL secondary servers to connect to each other as well as your host OS. NOTE: This is insecure and should only be setting for testing purposes. You can set it back to deny later if preferred.

Click on Adapter 2 on the top tab and set Adapter 2 to Enable and NAT, like this:



Adapter 2 will allow a connection to the Internet and only needs to be enabled when you are installing new software using the package manager.

Start the new VM and login with your username and password that you configured from the Ubuntu installation:

```
amysmithsqlone login:amysmith
Password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-70-generic x86_64)
```

A hash symbol in front of the prompt means you are logged in as the root account, which has full administrative privileges. Otherwise, you need to use the “sudo” command to run commands in Linux as root. You can switch users and sudo as root when necessary using the command “sudo su -” like this:

```
amysmith@amysmithsqlone:~$ sudo su -
[sudo] password for amysmith:
root@amysmithsqlone:~# whoami
root
root@amysmithsqlone:~# exit
logout
amysmith@amysmithsqlone:~$ whoami
amysmith
amysmith@amysmithsqlone:~$
```

“Exit” allows you to logout as a user while “whoami” tells you whom you are logged in as.

Next check to see what your IP address is using “ip a”:

```

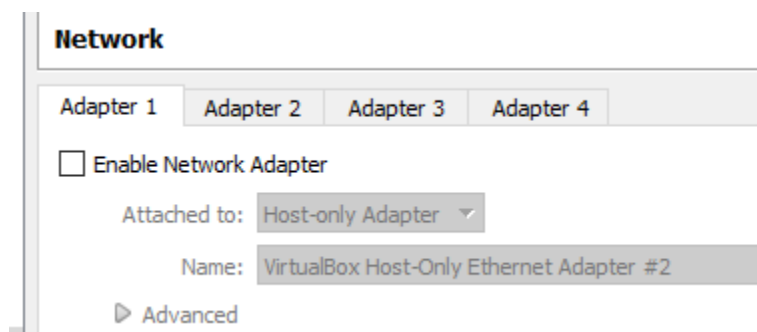
amysmith@amysmithsqlone:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state U
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc f
0
    link/ether 08:00:27:fb:f9:b6 brd ff:ff:ff:ff:ff:ff
    inet 10.100.100.2/24 brd 10.100.100.255 scope global dyna
        valid_lft 745sec preferred_lft 745sec
    inet6 fe80::a00:27ff:fe9b:f9b6/64 scope link
        valid_lft forever preferred_lft forever
3: enp0s8: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state D0
    link/ether 08:00:27:ac:89:aa brd ff:ff:ff:ff:ff:ff
amysmith@amysmithsqlone:~$

```

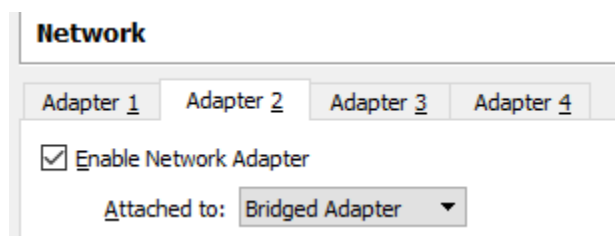
In this screenshot, the IP address is “10.100.100.2” under enp0s3. The loop back address is typically 127.0.0.1. This is a common address used by the host only and is not an IP addressed used to communicate with other servers and/or the Internet.

Next, we are going to shut down the VM and configure our NIC card to use NAT only or bridged mode. Use the command “sudo shutdown now” to shut down your VM from the OS. It is best to always shutdown your system using the operating system rather than virtual box just like in Windows or MacOS. Using virtual box is like the power button on your laptop, you do not want to use the power button but instead the OS menu to shut down a system cleanly.

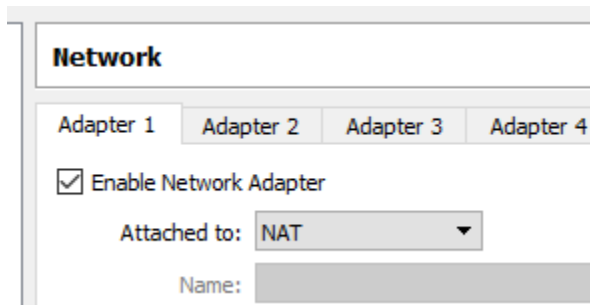
Disable your Host-only adapter:



Enable your NAT or Bridged Adapter on Adapter 2:



Start your VM again. Identify if your IP address changed using the “ip a” command. NOTE, if your computer does not support two (2) NIC cards, you may need to change Adapter 1 from Host-only to NAT. You should be able to have both NICs working at the same time but some machines can only have one NIC card and thus you must change between NAT and Host-Only as necessary in the lab.



Step 3 (Optional). Install a secure shell server to be able to remotely connect to your VM:

```
sudo apt install openssh-server openssh-client -y
```

Make sure it is started:

```
sudo systemctl status ssh
```

```
sudo systemctl start ssh
```

Disable UFW if necessary:

```
sudo ufw disable
```

```
sudo apt-get install rsync
```

Create the SSH key pair in **your home directory**:

```
cd /home/bobsmith
```

```
ssh-keygen -t rsa -P "" -f ~/.ssh/id_rsa
```

Place the public key in authorized_keys in the ssh directory:

```
cat ~/.ssh/id_rsa.pub >> ~/.ssh/authorized_keys
```

Set the permissions for your user to 0600 (this is crucial to have correct):

```
chmod 0600 ~/.ssh/authorized_keys
```

Attempt to SSH to your remote host:

```
ssh localhost
```

If everything is working, you should have been able to connect.

Step 4: Install MySQL on the primary DBMS server

Use the “ping” command to check if you can access the Internet from your VM. If your NIC is configured properly, you should receive replies from the server you ping, like this:

```
amysmith@amysmithsqlone:~$ ping www.yahoo.com
PING new-fp-shed.wg1.b.yahoo.com (74.6.231.19) 56(84) bytes of data.
64 bytes from media-router-fp71.canary.media.vip.ne1.yahoo.com (74.6.231.19):
=51.8 ms
64 bytes from media-router-fp71.canary.media.vip.ne1.yahoo.com (74.6.231.19):
=50.7 ms
64 bytes from media-router-fp71.canary.media.vip.ne1.yahoo.com (74.6.231.19):
=50.2 ms
```

With your network settings set to NAT or bridge mode (You must have internet connectivity for this step to work), update the package manager and install MySQL using these commands:

`sudo apt-get update`

```
amysmith@amysmithsqlone:~$ sudo apt-get update
[sudo] password for amysmith:
Hit:1 http://us.archive.ubuntu.com/ubuntu focal InRelease
Get:2 http://us.archive.ubuntu.com/ubuntu focal-updates InRelease [114 kB]
Get:3 http://us.archive.ubuntu.com/ubuntu focal-backports InRelease [101 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu focal-security InRelease [109 kB]
Fetched 324 kB in 1s (411 kB/s)
Reading package lists... Done
```

`sudo apt-get install mysql-server`

When it asks, do you want to continue put in an upper case Y and hit enter:

```
amysmith@amysmithsqlone:~$ sudo apt-get install mysql-server
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libi
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mec
  mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common m
  mysql-server-core-8.0
Suggested packages:
  libdata-dump-perl libipc-sharedcache-perl libwww-perl mailx tinyd
The following NEW packages will be installed:
  libcgi-fast-perl libcgi-pm-perl libencode-locale-perl libevent-core-
  libevent-pthreads-2.1-7 libfcgi-perl libhtml-parser-perl libhtml-
  libhtml-template-perl libhttp-date-perl libhttp-message-perl libi
  liblwp-mediatypes-perl libmecab2 libtimedate-perl liburi-perl mec
  mecab-utils mysql-client-8.0 mysql-client-core-8.0 mysql-common m
  mysql-server-core-8.0
0 upgraded, 25 newly installed, 0 to remove and 38 not upgraded.
Need to get 30.9 MB of archives.
After this operation, 250 MB of additional disk space will be used.
Do you want to continue? [Y/n] Y
```

If it installed successfully you should see that it is running, like this:

```
done!
update-alternatives: using /var/lib/mecab/dic/ipadic-utf8 to provide /
b-dictionary) in auto mode
Setting up libhtml-parser-perl (3.72-5) ...
Setting up libhttp-message-perl (6.22-1) ...
Setting up mysql-server-8.0 (8.0.23-0ubuntu0.20.04.1) ...
update-alternatives: using /etc/mysql/mysql.cnf to provide /etc/mysql/
Renaming removed key_buffer and myisam-recover options (if present)
mysqld will log errors to /var/log/mysql/error.log
mysqld is running as pid 1805
```

Next, we are going to install the “locate” command so that we can find files easily.

```
sudo apt-get install locate
```

For locate to work, you must update its database. Any time you install new software or make changes, you need to update the database to find the new files. “sudo updatedb” accomplishes this:


```
amysmith@amysmithsqlone:~$ sudo apt-get install locate
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  locate
0 upgraded, 1 newly installed, 0 to remove and 38 not up
Need to get 85.3 kB of archives.
After this operation, 265 kB of additional disk space wi
Get:1 http://us.archive.ubuntu.com/ubuntu focal/universe
Fetched 85.3 kB in 0s (267 kB/s)
Selecting previously unselected package locate.
(Reading database ... 71848 files and directories curren
Preparing to unpack .../locate_4.7.0-1ubuntu1_amd64.deb
Unpacking locate (4.7.0-1ubuntu1) ...
Setting up locate (4.7.0-1ubuntu1) ...
Processing triggers for man-db (2.9.1-1) ...
amysmith@amysmithsqlone:~$ sudo updatedb
amysmith@amysmithsqlone:~$ _
```

Step 5: Configure the Active Primary MySQL Server

You can use the “locate” command to find files if your files are in different locations than the lab. For example, the first file we need to edit is “mysqld.cnf”. Thus, try to find this file:

```
sudo updatedb
```

```
locate mysqld.cnf
```

```
amysmith@amysmithsqlone:~$ locate mysqld.cnf
/etc/mysql/mysql.conf.d/mysqld.cnf
amysmith@amysmithsqlone:~$
```

On my VM, the mysqld.cnf file exists in the directory “/etc/mysql/mysql.conf.d/”

The first step is to configure the primary MySQL server. Go to the directory where you mysqld.cnf file exists, make a copy of the file, and edit it. The “cd” command changes directories, the “ls” commands list file directory contents, and the “cp” command copies files. Follow the steps in the screenshot:

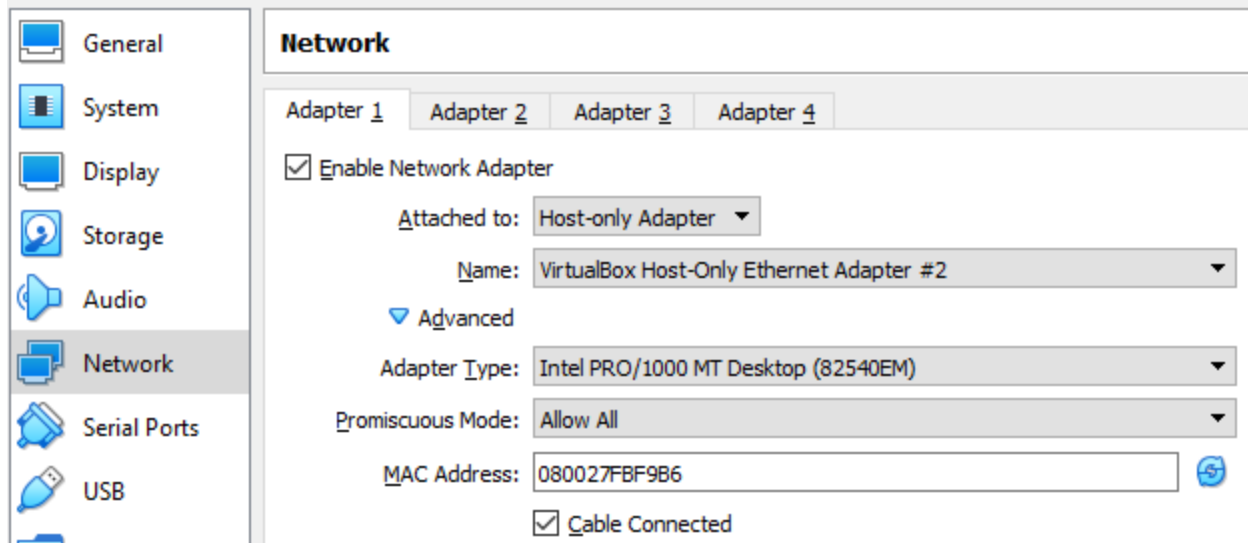
```
amysmith@amysmithsqlone:/etc/mysql/mysql.conf.d$ cd /etc/mysql/mysql.conf.d/
amysmith@amysmithsqlone:/etc/mysql/mysql.conf.d$ ls -lar
total 16
-rw-r--r-- 1 root root 2220 Nov 26 12:03 mysqld.cnf
-rw-r--r-- 1 root root 132 Nov 6 2019 mysql.cnf
drwxr-xr-x 4 root root 4096 Apr 4 15:48 ..
drwxr-xr-x 2 root root 4096 Apr 4 15:48 .
amysmith@amysmithsqlone:/etc/mysql/mysql.conf.d$ sudo cp mysqld.cnf ./mysqld.cnf
amysmith@amysmithsqlone:/etc/mysql/mysql.conf.d$ ls -lar
total 20
-rw-r--r-- 1 root root 2220 Apr 4 15:59 mysqld.cnf.backup
-rw-r--r-- 1 root root 2220 Nov 26 12:03 mysqld.cnf
-rw-r--r-- 1 root root 132 Nov 6 2019 mysql.cnf
drwxr-xr-x 4 root root 4096 Apr 4 15:48 ..
drwxr-xr-x 2 root root 4096 Apr 4 15:59 .
amysmith@amysmithsqlone:/etc/mysql/mysql.conf.d$ _
```

Next, we are going to use a file editor called “nano” to edit the mysqld.cnf file, which configures MySQL server.

```
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

Once you look through the file on your keyboard simultaneously use the keys CTRL and X to exit nano. CTRL + X.

We need to go back into Host-only mode now that we are done installing applications. Shutdown ubuntu using “sudo shutdown now” and go to the network settings of your VM. Change your NIC back to host-only mode:



NOTE, from this point forward know that if you need to install software, you need your NIC to be in NAT or bridge mode. However, for the MySQL distributed servers to talk to each other, you need to be in host-only mode.

Step 6: Clone your Ubuntu VM to Make a Secondary Ubuntu VM

Next, we are going to do a full clone of our Ubuntu VM in VirtualBox. Right click on your VM in VirtualBox and click “clone” to clone the VM. Rename the new VM with a “two” or secondary name as it will be the secondary server in the MySQL distributed cluster. Ensure the new VM has new MAC addresses:

← Clone Virtual Machine

New machine name and path

Please choose a name and optionally a folder for the new virtual machine. The new machine will be a clone of the machine **MySQLone**.

Name:

Path:

MAC Address Policy:

Additional Options: ☐ Keep Disk Names

☐ Keep Hardware UUIDs

Expert Mode

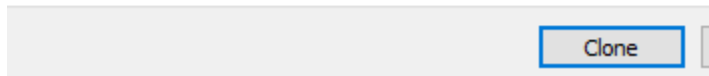
Next

Cancel

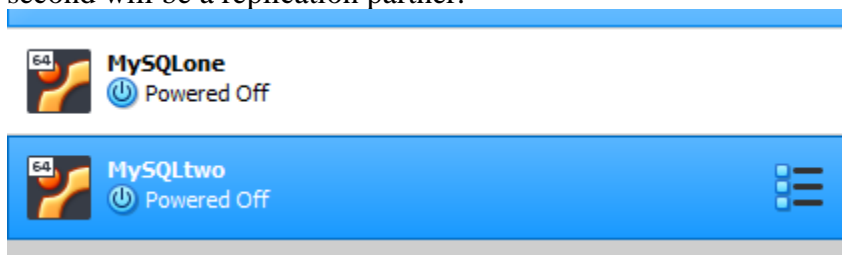
Perform a Full clone:

If you create a **Linked clone** then a new snapshot will be created in the virtual machine as part of the cloning process.

- ☒ Full clone
☐ Linked clone



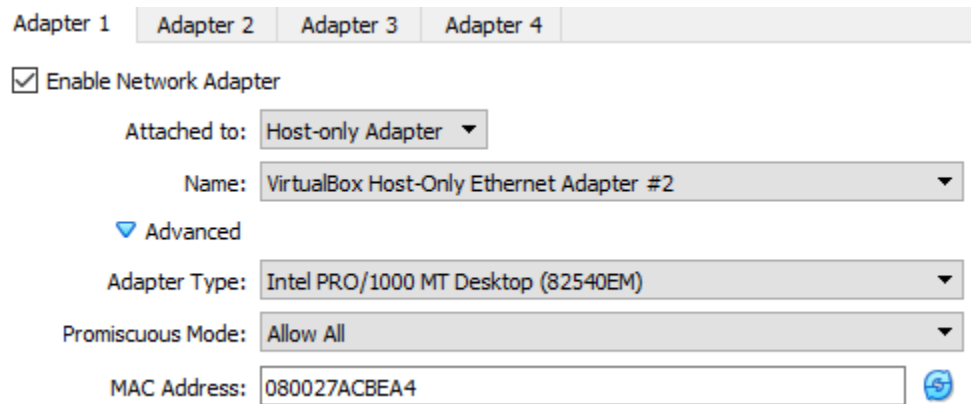
You should now have two MySQL VMs. One will be your primary database server and the second will be a replication partner.



Step 7: Configure and Active Passive MySQL Cluster

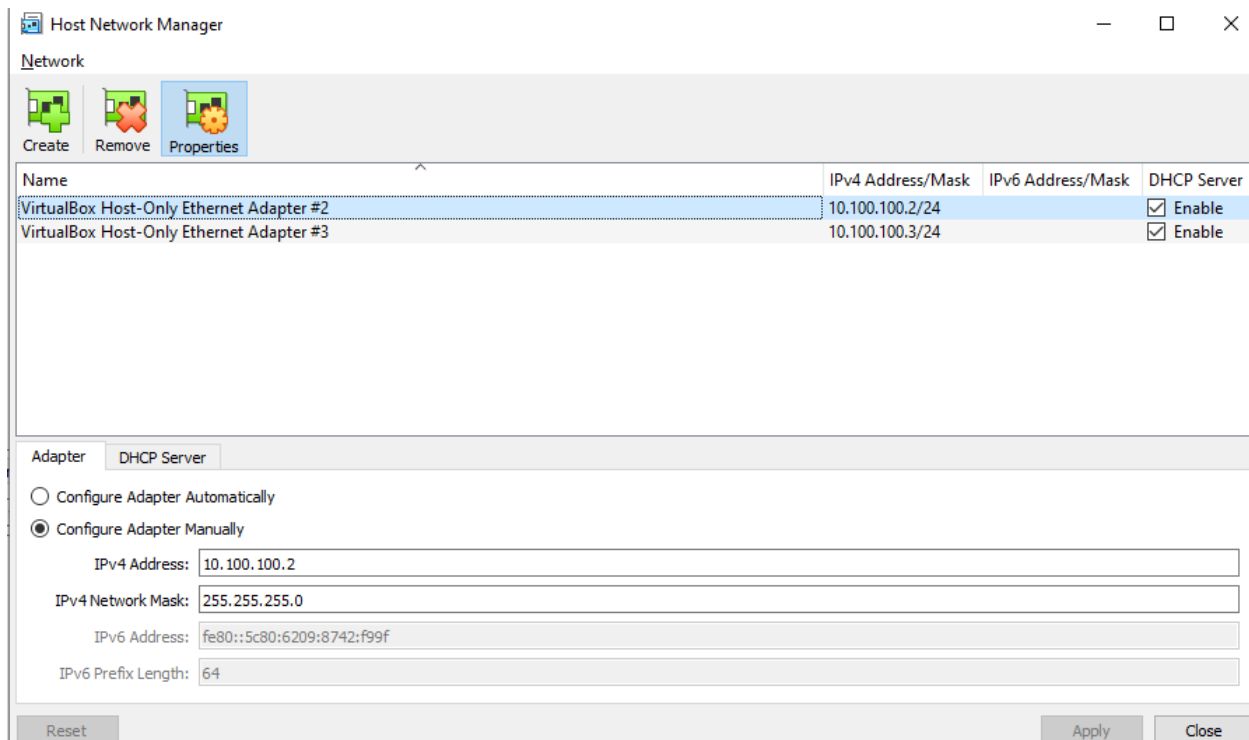
NOTE: From this point forward, you need to configure settings on BOTH your MySQL VMs.

Ensure BOTH VMs are in host only mode in their network settings. Start the VMs and determine their respective IP addresses. Make sure they have different IP addresses. If they do not, shut down your second VM and refresh the MAC address on the NIC, like this:



The screenshot shows the 'Adapter 1' tab in the VirtualBox Network Adapter settings. The 'Enable Network Adapter' checkbox is checked. The 'Attached to' dropdown is set to 'Host-only Adapter'. The 'Name' dropdown is set to 'VirtualBox Host-Only Ethernet Adapter #2'. The 'Advanced' section is expanded, showing 'Adapter Type' as 'Intel PRO/1000 MT Desktop (82540EM)', 'Promiscuous Mode' as 'Allow All', and 'MAC Address' as '080027ACBEA4'. A blue recycle icon is visible next to the MAC address field.

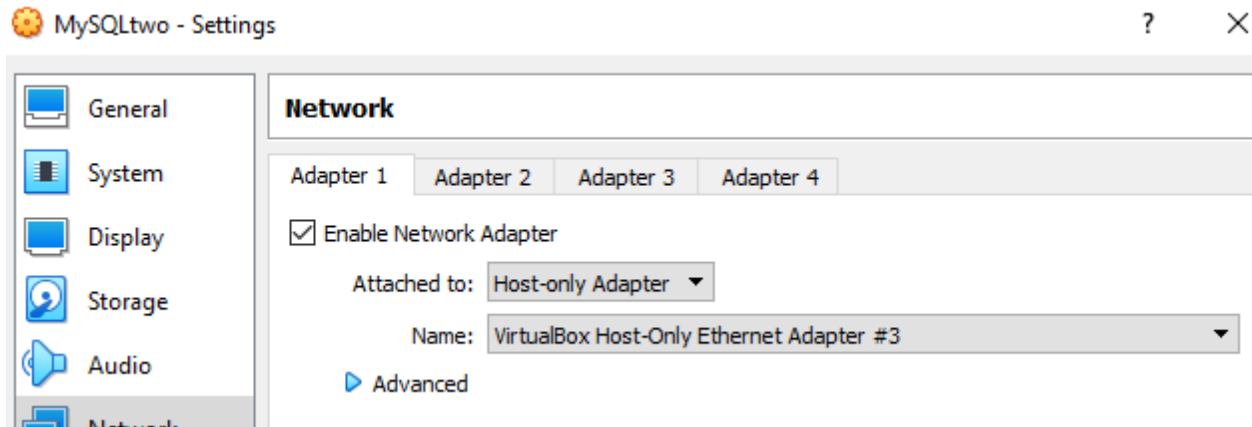
The blue recycle icon next to the MAC address will change it. If the IP addresses are still not the same, you can setup two different NICs in the Host Network Manager by configuring the adapters manually. **On the DHCP server tab, DISABLE DHCP.**



The screenshot shows the 'Host Network Manager' window. The 'Network' tab is active, displaying a list of network adapters. The 'VirtualBox Host-Only Ethernet Adapter #2' is selected. Below the list, the 'DHCP Server' tab is active, showing options to 'Configure Adapter Automatically' or 'Configure Adapter Manually'. The 'Configure Adapter Manually' option is selected, and the fields for 'IPv4 Address', 'IPv4 Network Mask', 'IPv6 Address', and 'IPv6 Prefix Length' are visible. The 'Reset', 'Apply', and 'Close' buttons are at the bottom.

Name	IPv4 Address/Mask	IPv6 Address/Mask	DHCP Server
VirtualBox Host-Only Ethernet Adapter #2	10.100.100.2/24		<input checked="" type="checkbox"/> Enable
VirtualBox Host-Only Ethernet Adapter #3	10.100.100.3/24		<input checked="" type="checkbox"/> Enable

Remember to change the network settings of each VM to user their new NIC:



When they have different IP addresses, ping the servers using their respective IP addresses from each other to make sure they can communicate with one another:

Go to the `/etc/netplan/` directory:

```
cd /etc/netplan/
```

Make a copy / backup of your `init.yaml` file in the `netplan` directory in case you need to restore your configuration for DHCP:

```
amysmith@amysmithsqlone:/etc/netplan$ cd /etc/netplan
amysmith@amysmithsqlone:/etc/netplan$ ls
00-installer-config.yaml
amysmith@amysmithsqlone:/etc/netplan$ sudo cp 00-installer-config.yaml ./00-installer-config.yaml.BACKUP
[sudo] password for amysmith:
Sorry, try again.
[sudo] password for amysmith:
amysmith@amysmithsqlone:/etc/netplan$ ls
00-installer-config.yaml 00-installer-config.yaml.BACKUP
amysmith@amysmithsqlone:/etc/netplan$ _
```

Use the “`ls`” command to check the `/etc/netplan` directory for the backup file:

```
ls -la
```

If copied successfully you should see the copy you made

We now need to set the static IP addresses on BOTH database servers. Switch to the Root user:

Sudo su -

Use the nano file editor. Use Ctrl + O to write to the file or Ctrl + X to exit out of the file

Sudo nano 50-cloud-init.yaml

Or

Sudo nano 00-installer-config.yaml

Netplan is very picky, it does not allow tabs. Use 2 spaces or 4 spaces. Enter spaces manually with the spacebar. If necessary, use backspace to assure every line starts without a tab. Add the configuration below for a static IP address:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [10.100.100.2/24]
      gateway4: 10.100.100.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

Once you are ready to apply the changes, use netplan apply:

```
$ sudo netplan apply
```

In case you run into some issues execute:

```
$ sudo netplan --debug apply
```

Now, change the IP address on your second database server to a **different IP address** such as 10.100.100.3.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s3:
      dhcp4: no
      addresses: [10.100.100.3/24]
      gateway4: 10.100.100.1
      nameservers:
        addresses: [8.8.8.8, 8.8.4.4]
```

MySQLone [Running] - Oracle VM VirtualBox	MySQLtwo [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help	File Machine View Input Devices Help
GNU nano 4.8	GNU nano 4.8
# This is the network config written by network: ethernets: enp0s3: dhcp4: no addresses: [10.100.100.2/24] gateway4: 10.100.100.1 nameservers: addresses: [8.8.8.8,1.1.1.1]_ version: 2	# This is the network config written by network: ethernets: enp0s3: dhcp4: no addresses: [10.100.100.3/24] gateway4: 10.100.100.1 nameservers: addresses: [8.8.8.8,1.1.1.1] version: 2_

To apply the settings you need to run the command:

Sudo netplan apply

There should not be any errors when running “sudo netplan apply”. Otherwise, you have a spacing or other issue in your yaml file.

Configure your /etc/hosts files to include the IP addresses and host names of your servers:

Sudo nano /etc/hosts

MySQLone [Running] - Oracle VM VirtualBox	MySQLtwo [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help	File Machine View Input Devices Help
GNU nano 4.8	GNU nano 4.8 /etc/hosts
127.0.0.1 localhost	127.0.0.1 localhost
10.100.100.2 amysmithsqlone	10.100.100.3 amysmithsqltwo
10.100.100.3 amysmithsqltwo_	10.100.100.2 amysmithsqlone
# The following lines are desirable for IPv6 capable hosts	# The following lines are desirable for IPv6 capable hosts
::1 ip6-localhost ip6-loopback	::1 ip6-localhost ip6-loopback
fe00::0 ip6-localnet	fe00::0 ip6-localnet
ff00::0 ip6-mcastprefix	ff00::0 ip6-mcastprefix
ff02::1 ip6-allnodes	ff02::1 ip6-allnodes
ff02::2 ip6-allrouters	ff02::2 ip6-allrouters

Make sure the hostnames have distinct names.

Sudo /etc/hostname

Change the secondary server to a new hostname from the first server.

Once both servers have separate IP addresses they should be able to ping each other. You should see successful replies from each host. If /etc/hosts is configured properly you can also ping them by hostnames in addition to their IP addresses:

```
MySQLOne [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
amysmith@amysmithsqlone:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_c
    link/ether 08:00:27:58:ee:e3 brd ff:ff:ff:ff:ff:ff
    inet 10.100.100.2/24 brd 10.100.100.255 scope global enp0s3
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe58:eee3/64 scope link
        valid_lft forever preferred_lft forever
amysmith@amysmithsqlone:~$ ping 10.100.100.3
PING 10.100.100.3 (10.100.100.3) 56(84) bytes of data.
 64 bytes from 10.100.100.3: icmp_seq=1 ttl=64 time=0.286 ms
 64 bytes from 10.100.100.3: icmp_seq=2 ttl=64 time=0.326 ms
 64 bytes from 10.100.100.3: icmp_seq=3 ttl=64 time=0.351 ms
 64 bytes from 10.100.100.3: icmp_seq=4 ttl=64 time=0.332 ms
^C
--- 10.100.100.3 ping statistics ---
 4 packets transmitted, 4 received, 0% packet loss, time 3074ms
rtt min/avg/max/mdev = 0.286/0.323/0.351/0.023 ms
amysmith@amysmithsqlone:~$ _

MySQLtwo [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
amysmith@amysmithsqltwo:~$ ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKN
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: enp0s3: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_c
    link/ether 08:00:27:3b:e8:ce brd ff:ff:ff:ff:ff:ff
    inet 10.100.100.3/24 brd 10.100.100.255 scope global enp0
        valid_lft forever preferred_lft forever
    inet6 fe80::a00:27ff:fe3b:e8ce/64 scope link
        valid_lft forever preferred_lft forever
amysmith@amysmithsqltwo:~$ ping 10.100.100.2
PING 10.100.100.2 (10.100.100.2) 56(84) bytes of data.
 64 bytes from 10.100.100.2: icmp_seq=1 ttl=64 time=0.322 ms
 64 bytes from 10.100.100.2: icmp_seq=2 ttl=64 time=0.282 ms
 64 bytes from 10.100.100.2: icmp_seq=3 ttl=64 time=0.375 ms
 64 bytes from 10.100.100.2: icmp_seq=4 ttl=64 time=0.396 ms
^C
--- 10.100.100.2 ping statistics ---
 4 packets transmitted, 4 received, 0% packet loss, time 3079ms
rtt min/avg/max/mdev = 0.282/0.343/0.396/0.044 ms
amysmith@amysmithsqltwo:~$
```

Once your servers IP addresses are configured, it is time to configure `mysqld.cnf` to include their respective IP addresses.

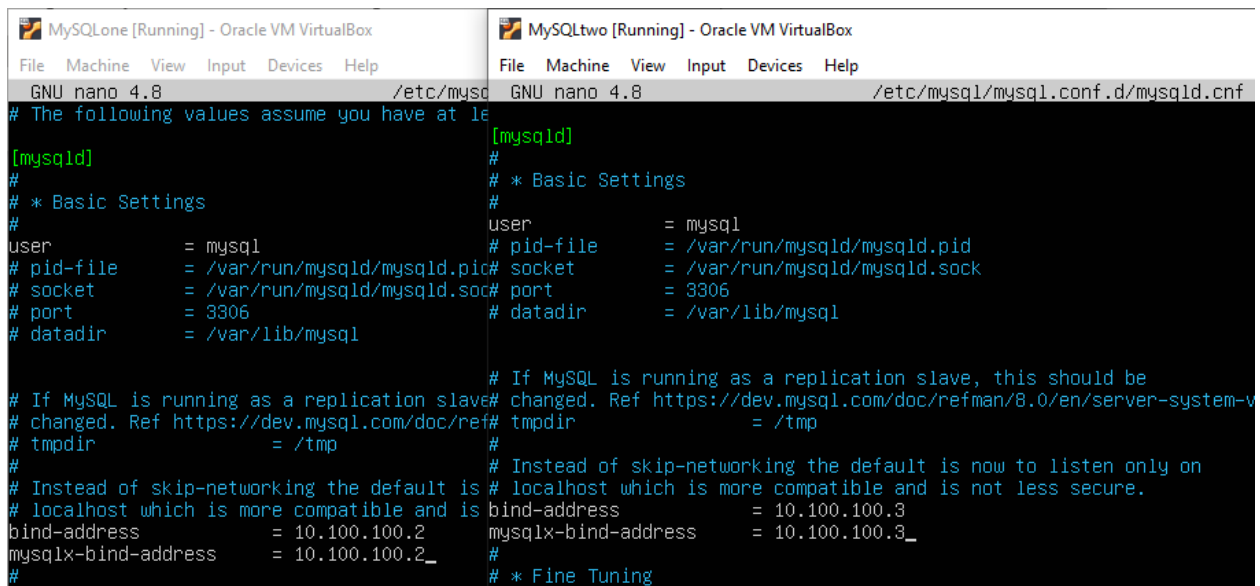
Locate `mysqld.cnf`

The latter locates the directory where your `mysqld.cnf` file resides. Next, edit the file:

`Sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf`

Add the correct IP addresses for each MySQL server:

```
bind-address      = 10.100.100.2
server-id         = 1
log_bin           = /var/log/mysql/mysql-bin.log
```



```
MySQLOne [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 4.8 /etc/mysql/my.cnf
# The following values assume you have at least one server running in a
# replication configuration
[mysqld]
#
# * Basic Settings
#
user                = mysql
# pid-file           = /var/run/mysql/mysql.pid
# socket             = /var/run/mysql/mysql.sock
# port               = 3306
# datadir            = /var/lib/mysql

# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar-tmpdir
# tmpdir             = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address        = 10.100.100.2
mysqlx-bind-address = 10.100.100.2_
#
# * Fine Tuning

MySQLtwo [Running] - Oracle VM VirtualBox
File Machine View Input Devices Help
GNU nano 4.8 /etc/mysql/mysql.conf.d/mysqld.cnf
[mysqld]
#
# * Basic Settings
#
user                = mysql
# pid-file           = /var/run/mysql/mysql.pid
# socket             = /var/run/mysql/mysql.sock
# port               = 3306
# datadir            = /var/lib/mysql

# If MySQL is running as a replication slave, this should be
# changed. Ref https://dev.mysql.com/doc/refman/8.0/en/server-system-variables.html#sysvar-tmpdir
# tmpdir             = /tmp
#
# Instead of skip-networking the default is now to listen only on
# localhost which is more compatible and is not less secure.
bind-address        = 10.100.100.3
mysqlx-bind-address = 10.100.100.3_
#
# * Fine Tuning
```

Set the primary MySQL server id = 1 and the secondary server to server id = 2, like this:

```
# The following can be used as easy to replicate the following can be used as easy to
# note: if you are setting up a replication # note: if you are setting up a replica
# other settings you may need to change # other settings you may need to
server-id = 1 server-id = 2_
# log_bin = /var/log/mysql/ log_bin = /var/log/mysql/
# binlog_expire_logs_seconds = 2592000 # binlog_expire_logs_seconds = 2592000
max_binlog_size = 100M max_binlog_size = 100M
# binlog_do_db = include_database_r# binlog_do_db = include_database_r
```

An important step is to change your server uid. When you clone virtual machines, the MySQL uid can stay the same. To fix this problem, remove the auto.cnf file:

locate auto.cnf

The latter should locate the file directory location of auto.cnf. Remove this file:

```
cp /var/lib/mysql/auto.cnf /var/lib/mysql/auto.cnf.BACKUP
```

```
rm /var/lib/mysql/auto.cnf
```

After your configuration is correct and saved restart the MySQL service:

```
sudo systemctl restart mysql
```

```
sudo systemctl status mysql
```

Disable any firewalls and/or add proper rules for MySQL to connect:

```
Sudo ufw disable
```

After the restart, the status should show mysql running on both servers.

```
amysmith@amysmithsqlone:~$ sudo systemctl restart mysql
amysmith@amysmithsqlone:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
  Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
  Active: active (running) since 2023-10-10 18:34:00; 1s ago
  Process: 1109 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=0)
  Main PID: 1136 (mysqld)
  Status: "Server is operational"
```

```
amysmith@amysmithsqltwo:~$ sudo systemctl restart mysql
amysmith@amysmithsqltwo:~$ sudo systemctl status mysql
● mysql.service - MySQL Community Server
  Loaded: loaded (/lib/systemd/system/mysql.service; enabled; vendor preset: enabled)
  Active: active (running) since 2023-10-10 18:34:15 UTC; 20s ago
  Process: 1130 ExecStartPre=/usr/share/mysql/mysql-systemd-start pre (code=0)
  Main PID: 1156 (mysqld)
  Status: "Server is operational"
```

Also, this is where it is nice to use SSH to remotely connect to your servers. The command to start ssh is:

```
Service sshd start
```

```
Service sshd status
```

```
Sudo ufw allow ssh
```

Or
Sudo ufw disable

To disable the firewall if it is blocking connections.

Once it is running, you can download an SSH client called “Putty” – Download Putty and connect to the IP address of each VM. Once SSHed you can copy and paste commands much faster.

Installing mysql

```
sudo apt update
sudo apt install mysql-server
sudo systemctl start mysql.service
```

Check to ensure MySQL is running

```
sudo systemctl status mysql
```

Login to the MySQL CLI:

```
sudo mysql -u
```

```
show databases;
quit;
```

On the PRIMARY MySQL node, we need to change the configuration settings.
We can modify it directly but it is best to go to the directory and make a backup.

```
parallelize
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf
```

```
cd /etc/mysql/mysql.conf.d/
```

Make a backup copy of mysqld.cnf:

```
sudo cp mysqld.cnf ./mysqld.cnf.BACKUP
```

Next modify the cnf configuration file for mysql.

```
sudo nano mysqld.cnf
```

Change the following lines. Here are there approximate locations:

```
# line 31: change to listen all
bind-address          = 0.0.0.0
mysqlx-bind-address   = 0.0.0.0
```

```
# line 73: uncomment and change to a unique ID for DB replication
server-id             = 10
```

```
# line 74: uncomment
log_bin               = /var/log/mysql/mysql-bin.log
```

```
# add to the end the enable clone plugin:
plugin-load=mysql_clone.so
plugin-load-add=mysql_clone.so
clone_valid_donor_list=michaelhartdone:3306
```

```
sudo systemctl restart mysql
sudo mysql -u root
```

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is
Server version:

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
# create replication user (set any password for [password] section)
mysql> create user 'cluster'@'%' identified by 'password';
Query OK, 0 rows affected (0.10 sec)
```

```
mysql> grant replication slave on *.* to cluster@'%';
Query OK, 0 rows affected (0.05 sec)
```

```
# create clone user
mysql> create user 'clone'@'%' identified by 'password';
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> grant backup_admin on *.* to 'clone'@'%';
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> flush privileges;
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> exit
Bye
```

If you need to change a users password:

```
ALTER USER 'root'@'localhost' IDENTIFIED BY 'MyNewPass';
ALTER USER 'cluster'@'%' identified by 'oracle';
```

On the SECONDARY MySQL node, change the settings for replication:
sudo nano /etc/mysql/mysql.conf.d/mysqld.cnf

Approximate line numbers:

line 31: change bind address to listen all

bind-address = 0.0.0.0

mysqlx-bind-address = 0.0.0.0

line 73: uncomment and change to a DIFFERENT ID THAN the Primary MySQL server

server-id = 11

line 74: uncomment

log_bin = /var/log/mysql/mysql-bin.log

make the secondary MySQL node read only

read_only=1

define your secondary node hostname

report-host=michaelhartdtwo

define relay logs

relay-log=/var/log/mysql/node01-relay-bin

relay-log-index=/var/log/mysql/node01-relay-bin

Add plugin code

plugin-load=mysql_clone.so

plugin-load-add=mysql_clone.so

Add permissions to clone for PRIMARY MySQL node

clone_valid_donor_list=michaelhardone:3306

Next, restart MySQL and check the status to ensure no errors were made in the conf:

```
sudo systemctl restart mysql  
sudo systemctl status mysql
```

Next, login to the MySQL CLI:

```
sudo mysql -u root
```

```
Welcome to the MySQL monitor.  Commands end with ; or \g.  
Your MySQL connection id is  
Server version:
```

```
Copyright (c) 2000, 2020, Oracle and/or its affiliates. All rights reserved.
```

```
Oracle is a registered trademark of Oracle Corporation and/or its  
affiliates. Other names may be trademarks of their respective  
owners.
```

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.
```

```
# create a clone user with an password easy to remember  
mysql> create user 'clone'@'%' identified by 'password';  
Query OK, 0 rows affected (0.07 sec)
```

```
mysql> grant clone_admin on *.* to 'clone'@'%';  
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> grant BACKUP_ADMIN on *.* to 'clone'@'%';  
Query OK, 0 rows affected (0.06 sec)
```

```
mysql> flush privileges;  
Query OK, 0 rows affected (0.00 sec)
```

```
mysql> exit  
Bye
```

Next, SSH to the secondary database node. This is your backup mysql database server.

```
sudo systemctl start mysql  
sudo systemctl status mysql
```


Next, login to the MySQL CLI:

```
sudo mysql -u root
```

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is
Server version:

Copyright (c) 2000, 2022, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

```
# Set the global clone_valid_donor_list to you PRIMARY DB server = Primary Host IP  
address:port  
mysql> set global clone_valid_donor_list = '10.100.100.12:3306';  
Query OK, 0 rows affected (0.00 sec)
```

```
# start Clone  
mysql> clone instance from clone_user@10.100.100.12:3306 identified by 'password';  
Query OK, 0 rows affected (0.49 sec)
```

Here are examples:

```
CLONE INSTANCE FROM 'clone'@'10.100.100.51':3306 IDENTIFIED BY 'oracle';  
CLONE INSTANCE FROM 'clone'@'michaelhardone':3306 IDENTIFIED BY 'oracle';
```

```
# Check the clone status
```

```
mysql> select ID,STATE,SOURCE,DESTINATION,BINLOG_FILE,BINLOG_POSITION  
from performance_schema.clone_status;
```

ID	STATE	SOURCE	DESTINATION	BINLOG_FILE	BINLOG_POSITION
1	Completed	michaelhardone:3306	LOCAL INSTANCE	mysql-bin.000002	

```
1 row in set (0.00 sec)
```

On the PRIMARY server check your MySQL status:
SHOW MASTER STATUS\G

Make note of these details as they are necessary to start a secondary MySQL instance.

Next, start your SECONDARY mysql server as a replication partner:

```
# replication settings
# master_host    ⇒ Primary Host IP address
# master_user    ⇒ replication user
# master_password ⇒ replication user's password
# master_ssl     ⇒ enable SSL connection
# master_log_file ⇒ specify [BINLOG_FILE] value it is displayed above
# master_log_pos  ⇒ specify [BINLOG_POSITION] value it is displayed above
mysql> change master to
master_host='michaelhartdone',
master_user='cluster',
master_password='password',
master_ssl=1,
master_log_file='mysql-bin.000003',
master_log_pos=1089;
Query OK, 0 rows affected, 9 warnings (0.04 sec)

# start replication
mysql> start slave;
Query OK, 0 rows affected, 1 warning (0.02 sec)

# show status
mysql> show slave status\G
```

Once started, it is important to check whether replication is working properly again:

```
mysql> SHOW SLAVE STATUS \G
```

Give it a few minutes, check again and assure no errors exist. Now leave the MySQL shell...

```
mysql> quit;
```

... and check the log file:

```
# grep mysql /var/log/syslog  
# cat /var/log/mysql/error.log
```

You should see something like this if everything is working:

```
2019-10-25T00:51:59.827224Z 2 [Note] Slave I/O thread for channel "": connected to master  
'replica@192.168.36.129:3306',replication started in log 'mysql-bin.000001' at position 1100  
If either the log file or replication status is in a failed state, you need to fix replication BEFORE  
creating any new databases or tables. Or else, the replication will not synchronize properly. If  
this happens, you need to jump down to the instructions below to re-sync the master-slave  
databases.
```

Step 8: Test the Configuration

At this point, you should have a working MySQL replication server environment. The primary MySQL server keeps a copy of the database and all changes. Changes should only be made on the primary MySQL server. If the secondary server becomes out of sync, you have to back up the primary MySQL server and restore it to the secondary server. Once restored, you can re-establish the replication partners.

To verify that everything works as expected, we will create a new database on the master server:

```
sudo mysql
```

```
mysql> CREATE DATABASE replication_database;
```

Login to the secondary MySQL shell:

```
sudo mysql
```

List the databases:

```
mysql> SHOW DATABASES;
```

The database you created on the master server should be replicated on the secondary server:

```
+-----+
| Database          |
+-----+
| information_schema |
| mysql             |
| performance_schema |
| replication_database; |
| sys               |
+-----+
```

Go ahead and use the new database, create a table, and put a record in the table to see if it replicates.

```
mysql> USE replication_database;
```

```
mysql> CREATE TABLE replication (  
id INT(10) AUTO_INCREMENT PRIMARY KEY,  
name VARCHAR(50)  
);
```

Now, insert a record into the replication table and perform a select * from the table on the non-active database server to see if the data is replicating!

Once everything is working take a screenshot of

1. the # cat /var/log/mysql/error.log files on both servers open
2. A screenshot of mysql> SHOW SLAVE STATUS \G
3. the show SELECT * statements running on both the master and slave servers.
4. Submit the screenshots to D2L for credit.

You should be finished if everything is working! Regardless, there are additional steps below that are necessary if your two database servers ever get out of sync.

Step 9 (Optional): Synchronizing A Replication Database Out of Sync

----- Note, you only need to attempt these next steps if replication fails to reinitialize

The primary MySQL server keeps a copy of the database and all changes. Changes should only be made on the primary MySQL server. If the secondary server gets out of sync, you have to backup the primary MySQL server and restore it to the secondary server.

If replication fails at any point you may need to restore the replication partner that failed.

First, it is important to check whether replication is working properly:

```
mysql> SHOW SLAVE STATUS \G  
mysql> SHOW SLAVE STATUS \G  
***** 1. row *****  
Slave_IO_State: Waiting for master to send event  
Master_Host: 1.2.3.4  
Master_User: slave_user  
Master_Port: 3306  
Connect_Retry: 60
```

```
Master_Log_File: mysql-bin.001079
Read_Master_Log_Pos: 447560366
Relay_Log_File: slave-relay.000130
Relay_Log_Pos: 225644062
Relay_Master_Log_File: mysql-bin.001079
Slave_IO_Running: Yes
Slave_SQL_Running: Yes
Replicate_Do_DB: mydb
Replicate_Ignore_DB:
```

You can now exist from the MySQL CLI.

```
mysql> quit;
```

... and check the log file:

```
grep mysql /var/log/syslog
cat /var/log/mysql/error.log
```

If either the log file or replication status is in a failed state, you need to fix replication.

On the secondary server:

First, we need to stop secondary MySQL server. Login to MySQL server and execute following command.

```
mysql> STOP SLAVE;
```

On the primary Server:

After stopping the secondary go to primary server and reset the master state using following command.

```
mysql> RESET MASTER;
```

Add a read lock on the database:

```
mysql> FLUSH TABLES WITH READ LOCK;
```

Take a backup of database is being replicated using following command.

```
# mysqldump --all-databases > db_backup.sql
```

```
#quit
```

Now start your SSHD server so that you can transfer the file

```
# service ssh start  
>sudo service ssh start (if you are not logged in as root)
```

You can use WinSCP to transfer the backup file to the secondary MySQL server:

<https://winscp.net/eng/download.php>

Once installed, open WinSCP and connect to the different hosts to transfer files back and forth.

Or, copy the contents of the dump file via VirtualBox by sharing your clipboard.

On the secondary Server:

Restore database backup taken from the primary server on the secondary server using following command.

```
# mysql -u root -p mydb < mydb-dump.sql
```

OR simply:

```
# mysql < db_backup.sql
```

Login to mysql and execute following commands to reset slave state also.

```
mysql> RESET SLAVE;
```

Reset the slave to the master log file position. From the master database server:
SHOW MASTER STATUS \G

Document the mysql-bin position on the master.

Go back to the secondary/slave mysql server and change the mysql bin position to the master.

```
mysql> CHANGE MASTER TO MASTER_LOG_FILE='mysql-bin.000001',  
MASTER_LOG_POS=154;
```

After taking a backup and restoring it you can unlock the tables on the primary server again:

```
mysql> UNLOCK TABLES;
```

Then start the secondary MySQL server again:

```
mysql> START SLAVE;
```

Now your replication should be re-synchronized:

```
mysql> SHOW SLAVE STATUS \G
```