**Mongo DB Configuration**

1. **Download latest oracle virtual box**

<https://filehippo.com/download_virtualbox/>

* Once downloaded the file click on the installer and complete the installation.

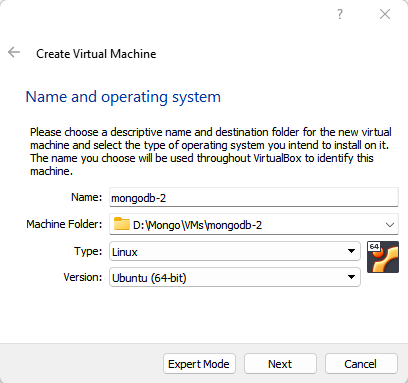
1. **Download Ubuntu**

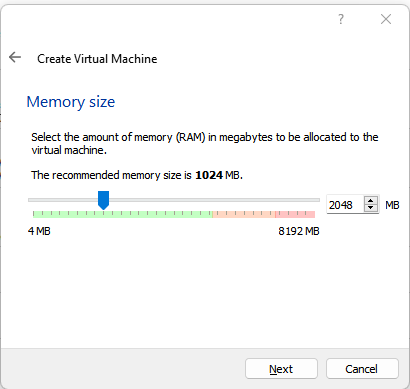
<https://www.osboxes.org/ubuntu/>

Download the below VirtualBox (VDI) 64bit

Ubuntu 22.04 Jammy Jellyfish

1. **Setup VM Box**
2. Search for Oracle VM virtualbox from Windows start menu and launch it.
3. Click on new and create virtual machine





**4. Fix VirtualBox UUID Errors**

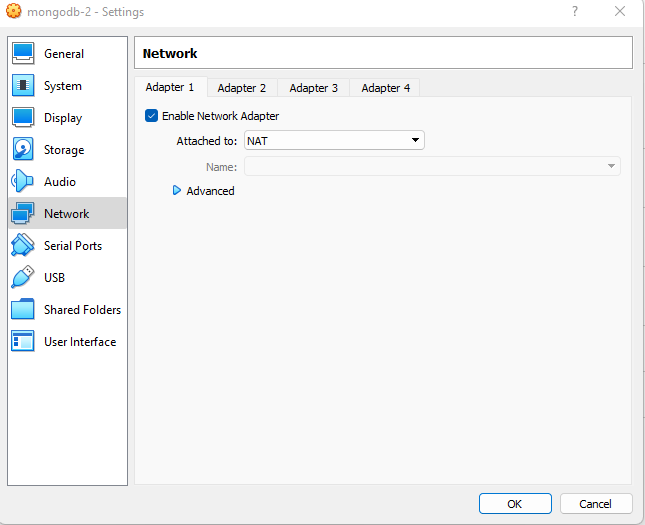
C:\ProgramFiles\Oracle\VirtualBox>

VBOXMANAGE.EXE internalcommands sethduuid D:\Mongo\VMs\mongodb-2\mongodbdisk2.vdi

UUID changed to: e5baa3c0-f4b3-49ec-b980-3ebb4e39570d

**5 Add 2 network cards.**

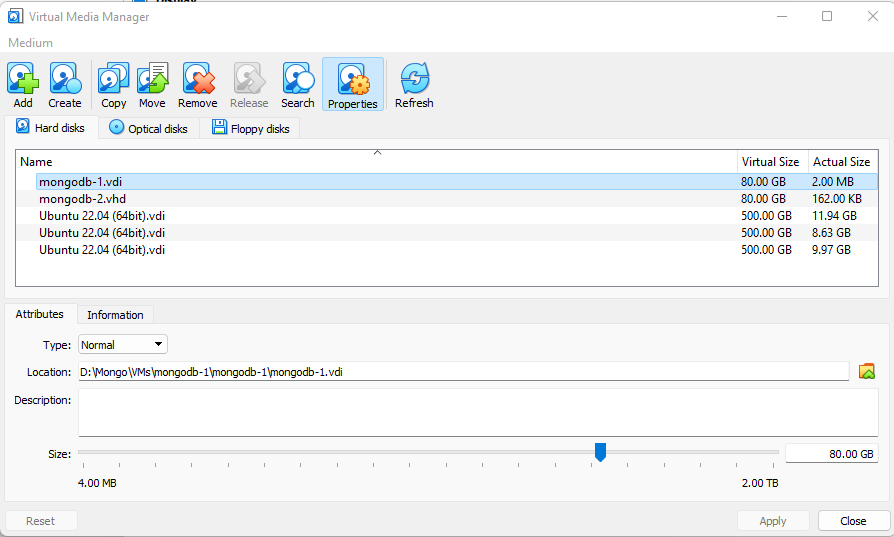
1. One NAT for network connectivity to the VM box
2. Second for local windows connectivity. To open Putty session



1. **Attach Ubuntu as Storage which is downloaded from the osboxes.org**

Open Virtual Media Manager - > Oracle VM VIrtual Manager - > File -> Virtual Media Manager

Click on add and browse for the downloaded Ubuntu storage



Start Virutal machine

Username: osboxes

Password: osboxes.org

Gust Tools: Installed

Keyboard Layout: US (Qwerty)

VMware Compatibility: Version 10+

sudo su - root

Password - osboxes.org

Change password for root if you would like

passwd root

root password - root123

1. **Setting Up and Securing SSH on Ubuntu 22.04**

<https://serverastra.com/docs/Tutorials/Setting-Up-and-Securing-SSH-on-Ubuntu-22.04%3A-A-Comprehensive-Guide>

**Prerequisites**

An Ubuntu 22.04 system with root privileges

**Step 1: Update the System**

Before you start, make sure your system is up to date. Open a terminal window and run the following commands:

sudo apt update

sudo apt upgrade

**Step 2: Install the OpenSSH Server**

To enable SSH on your Ubuntu system, you'll need to install the OpenSSH server. Run the following command in the terminal window:

sudo apt install openssh-server

**Step 3: Configure Firewall Rules for SSH**

By default, Ubuntu comes with the Uncomplicated Firewall (UFW) to manage firewall rules. First, check the status of UFW by running:

sudo ufw status

If UFW is inactive, enable it with the following command:

sudo ufw enable

Next, allow inbound SSH connections by adding a new firewall rule:

sudo ufw allow ssh

**Step 4: Change the Default SSH Port**

In this example, we'll change the default SSH port from 22 to 33556. You can choose a different port number if you prefer, but make sure it's not already in use by another service.

First, create a backup of the original SSH configuration file:

sudo cp /etc/ssh/sshd\_config /etc/ssh/sshd\_config.bak

Next, open the SSH configuration file using a text editor like nano:

Find the line that starts with #Port 22 and change it to the desired port number. Remove the # at the beginning of the line to uncomment it. For example:

Port 22

Save the file and exit the text editor.

**Step 5: Configure Firewall Rules for the Custom SSH Port**

Update the firewall rules to allow inbound connections on the custom SSH port. First, delete the previous SSH rule:

sudo ufw delete allow ssh

Then, add a new rule for the custom SSH port:

sudo ufw allow 33556/tcp

**Step 6: Restart the SSH Service**

Restart the SSH service to apply the changes:

sudo systemctl restart ssh

**Step 7: Verify the Configuration**

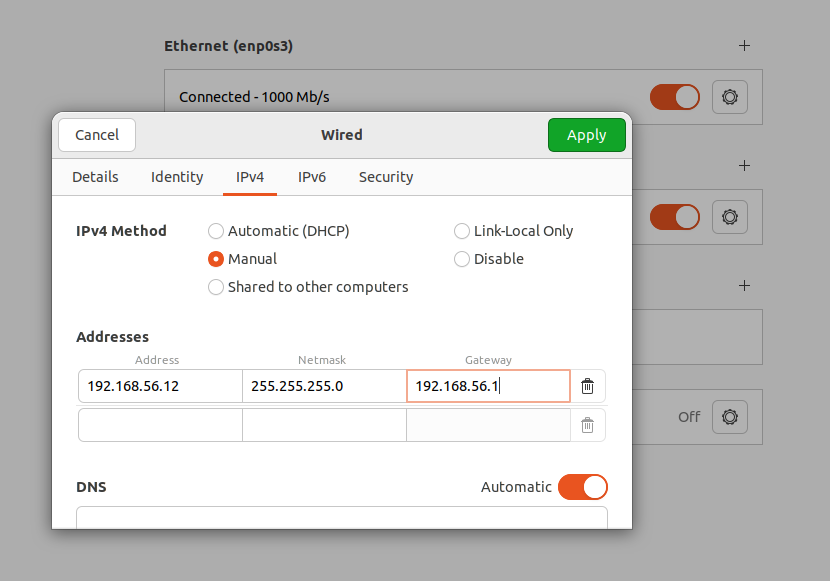
To verify that SSH has been enabled and the new SSH port and firewall rules have been configured correctly, run the following command in the terminal window:

sudo systemctl status ssh

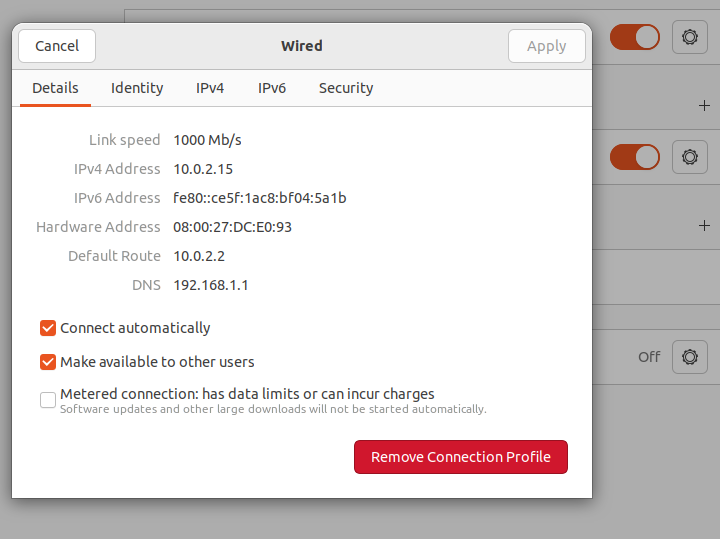
If the configuration was successful, you should see information about the SSH service running with the new port number.

1. **Network settings**

Change the IPV4 settings as below and restart the VM to take the effect. This Host only network is for connecting putty sessions from Windows laptop.



Leave the NAT related network details as default. This network is for connecting internet to the VM box.



1. **Update hostname and /etc/hosts file**

root@osboxes:~# cat /etc/hostname

#osboxes

mongodb-2

root@osboxes:~#

root@osboxes:~# cat /etc/hosts

127.0.0.1 localhost

127.0.1.1 osboxes

# The following lines are desirable for IPv6 capable hosts

::1 ip6-localhost ip6-loopback

fe00::0 ip6-localnet

ff00::0 ip6-mcastprefix

ff02::1 ip6-allnodes

ff02::2 ip6-allrouters

192.168.56.10 mongodb-1

192.168.56.11 mongodb-2

192.168.56.12 mongodb-3

root@osboxes:~#

1. **Install MongoDB in Ubuntu**

**Follow the below link for the latest instructions to install mongodb**

**<https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/>**

### **Reload local package database**

sudo apt-get update

### **Install the MongoDB packages[IMG_256](https://www.mongodb.com/docs/manual/tutorial/install-mongodb-on-ubuntu/#install-the-mongodb-packages)**

sudo apt-get install -y mongodb-org

If you installed through the package manager, the data directory /var/lib/mongodb and the log directory /var/log/mongodb are created during the installation.

#### Start MongoDB.

sudo systemctl start mongod

#### Verify that MongoDB has started successfully.

sudo systemctl status mongod

You can optionally ensure that MongoDB will start following a system reboot by issuing the following command:

sudo systemctl enable mongod

#### Stop MongoDB.

sudo systemctl stop mongod

#### Restart MongoDB.

sudo systemctl restart mongod

### **Setting up a new MongoDB Replica Set**

You can set up a new Replica Set in MongoDB using the following steps:

#### **[Step 1:Updating Each Server’s Firewall Configurations with UFW](https://www.digitalocean.com/community/tutorials/how-to-configure-a-mongodb-replica-set-on-ubuntu-20-04" \l "step-2-updating-each-server-s-firewall-configurations-with-ufw)**

On each server, run the following ufw command to allow  access to port 27017

ufw allow from 192.168.56.10 to any port 27017

ufw allow from 192.168.56.11 to any port 27017

ufw allow from 192.168.56.12 to any port 27017

#### **Step 2: Starting the mongod Instances on all the servers**

To start the mongod instance, specify the port value for your Mongo instance along with the path to your MongoDB installation on your system. Use the following command to enable your mongod instance:

mongod --port 27017 --dbpath /var/lib/mongodb --logpath /var/log/mongodb/mongod.log --replSet replicaSet1 --bind\_ip\_all --fork

You can also connect them by using the following lines of code:

mongosh –host mongodb-2 –port 27017

mongosh –host mongodb-3 –port 27017

#### **[Step 3 — Enabling Replication in Each Server’s MongoDB Configuration File](https://www.digitalocean.com/community/tutorials/how-to-configure-a-mongodb-replica-set-on-ubuntu-20-04" \l "step-3-enabling-replication-in-each-server-s-mongodb-configuration-file)**

root@mongodb-2:~# **cat /etc/mongod.conf**

# mongod.conf

# for documentation of all options, see:

# http://docs.mongodb.org/manual/reference/configuration-options/

# Where and how to store data.

storage:

dbPath: /var/lib/mongodb

# engine:

# wiredTiger:

# where to write logging data.

systemLog:

destination: file

logAppend: true

path: /var/log/mongodb/mongod.log

# network interfaces

net:

port: 27017

**bindIp: 127.0.0.1,mongodb-2**

# how the process runs

processManagement:

timeZoneInfo: /usr/share/zoneinfo

#security:

#operationProfiling:

replication:

replSetName: "replicaSet1"

#sharding:

## Enterprise-Only Options:

#auditLog:

root@mongodb-2:~#

#### **Step 4: Enabling Replication in MongoDB**

Once you’ve made all the configurations, you now need to open the Mongo Shell with your primary instance and use initiate command as follows:

config = {\_id: "replicaSet1", members:[

{\_id: 0, host: 'mongodb-1:27017'},

{\_id: 1, host: 'mongodb-2:27017'},

{\_id: 2, host: 'mongodb-3:27017'}]

}

rs.initiate(config)

if the replication is already initialized , use **rs.reconfig()** to reconfigure the replication set.

The output {‘ok’:1} indicates that a MongoDB instance has been successfully added to the Replica Set.

replicaSet1 [direct: primary] test> rs.reconfig(config)

{

**ok: 1,**

'$clusterTime': {

clusterTime: Timestamp({ t: 1697130162, i: 1 }),

signature: {

hash: Binary.createFromBase64("AAAAAAAAAAAAAAAAAAAAAAAAAAA=", 0),

keyId: Long("0")

}

},

operationTime: Timestamp({ t: 1697130162, i: 1 })

}

To check the status of the replication, you can use the status command as follows:

**rs.status()**

The status command will produce the following output if the MongoDB instances have been added successfully:

replicaSet1 [direct: primary] test> rs.status()

{

set: 'replicaSet1',

date: ISODate("2023-10-12T17:03:09.516Z"),

myState: 1,

term: Long("6"),

syncSourceHost: '',

syncSourceId: -1,

heartbeatIntervalMillis: Long("2000"),

majorityVoteCount: 2,

writeMajorityCount: 2,

votingMembersCount: 3,

writableVotingMembersCount: 3,

optimes: {

….

{

\_id: 0,

name: 'mongodb-1:27017',

health: 1,

state: 1,

stateStr: 'PRIMARY',

{

\_id: 1,

name: 'mongodb-2:27017',

health: 1,

state: 2,

stateStr: 'SECONDARY',

uptime: 26,

{

\_id: 2,

name: 'mongodb-3:27017',

health: 1,

state: 2,

stateStr: 'SECONDARY',

replicaSet1 [direct: primary] test> **rs.isMaster()**

{

topologyVersion: {

processId: ObjectId("65282637f1d8fae6b51bee8b"),

counter: Long("9")

},

hosts: [ 'mongodb-1:27017', 'mongodb-2:27017', 'mongodb-3:27017' ],

setName: 'replicaSet1',

setVersion: 4,

ismaster: true,

secondary: false,

**primary: 'mongodb-1:27017',**

**me: 'mongodb-1:27017',**

#### **Step 6: Testing the Replication Process**

You can test the process by adding a document in the primary node. If replication is working properly, the document will automatically be copied into the secondary node.

First, connect to the primary node and add a document using the insertOne command as follows:

use carData

db.persons.insertOne({name:"Max",age:29,salary:30000})

db.cars.insertOne({model:"BMW", price:40000, owner:ObjectId("5f074743a950295538eca6b6")})

db.cars.findOne().owner

**Connect one of the secondary node and check if the data replicated.**

replicaSet1 [direct: secondary] carData>

db.getMongo().setReadPref('primaryPreferred')

replicaSet1 [direct: secondary] carData> db.cars.find();

[

{

\_id: ObjectId("652830b8cdb02420d70e0ea4"),

model: 'BMW',

price: 40000,

owner: ObjectId("5f074743a950295538eca6b6")

}

]

**Check the status of the replication Status**

replicaSet1 [direct: primary] test> rs.printSecondaryReplicationInfo()

source: mongodb-2:27017

{

syncedTo: 'Thu Oct 12 2023 13:37:00 GMT-0400 (Eastern Daylight Time)',

replLag: '0 secs (0 hrs) behind the primary '

}

---

source: mongodb-3:27017

{

syncedTo: 'Thu Oct 12 2023 13:37:00 GMT-0400 (Eastern Daylight Time)',

replLag: '0 secs (0 hrs) behind the primary '

}

replicaSet1 [direct: primary] test> rs.printReplicationInfo()

actual oplog size

'9954.731689453125 MB'

---

configured oplog size

'9954.731689453125 MB'

---

log length start to end

'27280 secs (7.58 hrs)'

---

oplog first event time

'Thu Oct 12 2023 06:03:00 GMT-0400 (Eastern Daylight Time)'

---

oplog last event time

'Thu Oct 12 2023 13:37:40 GMT-0400 (Eastern Daylight Time)'

---

now

'Thu Oct 12 2023 13:37:44 GMT-0400 (Eastern Daylight Time)'