[1. Introduction 2](#_Toc486878357)

[2. Virtualbox setup 3](#_Toc486878358)

[3. Ubuntu server setup 3](#_Toc486878359)

[3.1. Ubuntu server installation 3](#_Toc486878360)

[3.2. Ubuntu server configuration 6](#_Toc486878361)

[3.3. Python & Ansible installation 8](#_Toc486878362)

[4. Setting up the first vMX 8](#_Toc486878363)

[5. Building vMX clones 11](#_Toc486878364)

[6. Initial vMX configuration 13](#_Toc486878365)

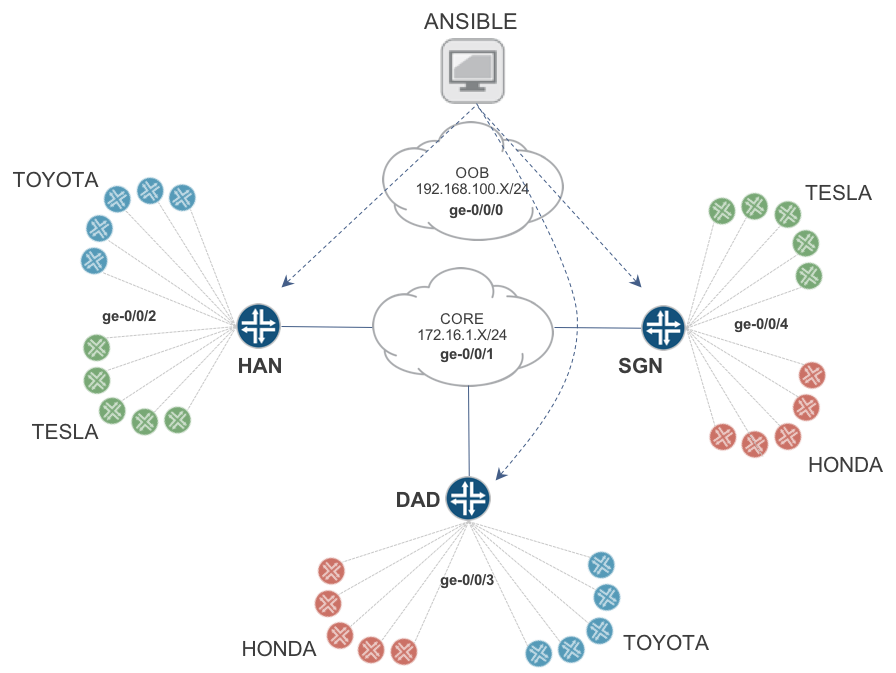
# 

# 

# Introduction

This document describes the initial lab setup used for the Ansible training. In this training, we will use Ansible to configure and validate an L3VPN network across 3 PE routers. The below lab setup with 3 vMX (PE) routers and 1 Ubuntu (Ansible) server will be used for the training. The vMX routers used will run an optimized custom image that uses only 2GB RAM/vMX (1GB for RE, 1GB for FPC). This image is meant only for lab setups and helps in setting up the complete topology on an SE laptop with 16GB RAM. Virtual Box is used as the virtualization platform.

Each vMX will be setup with 2 interfaces, ge-0/0/0 for connectivity to the Ansible server and local machine & ge-0/0/1 for connectivity between the vMXs. So, let’s get started. The Ansible server will also have 2 interfaces, one connecting to the vMXs and other to the internet via NAT.



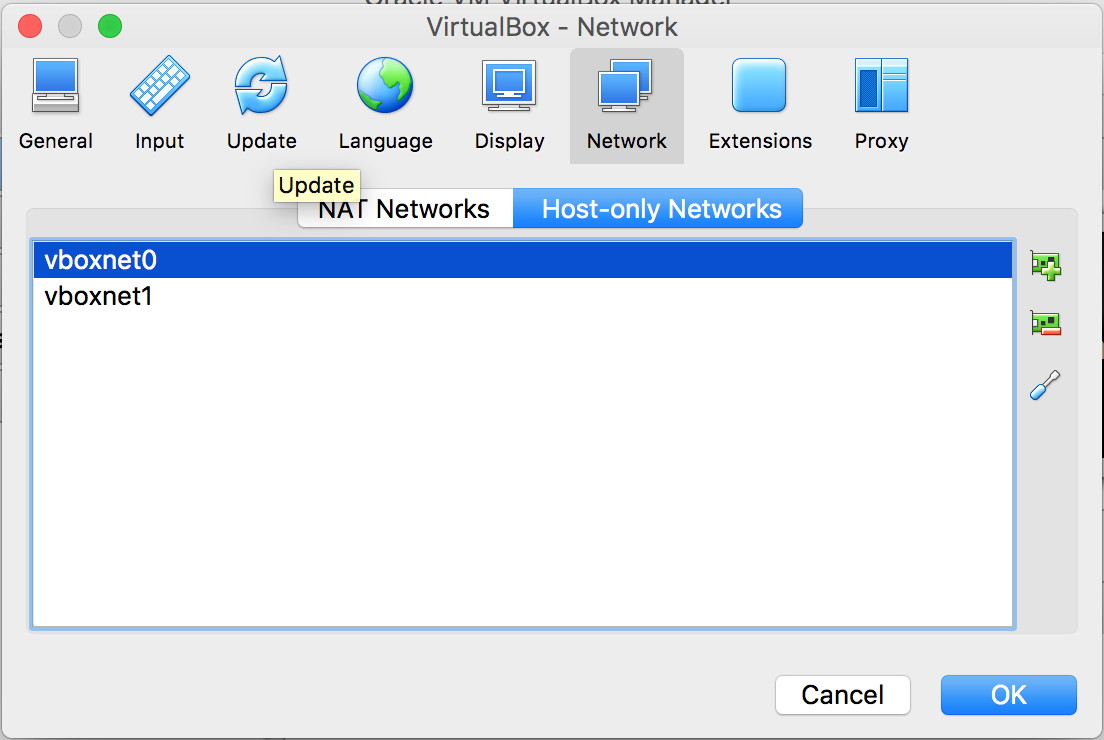
**Figure 1: Lab setup**

# Virtualbox setup

Install Virtual Box and VirtualBox extension pack:

<https://www.virtualbox.org/wiki/Downloads>

Under Virtualbox > Preferences > Network > Host-only networks > Add ‘vboxnet0’ interface

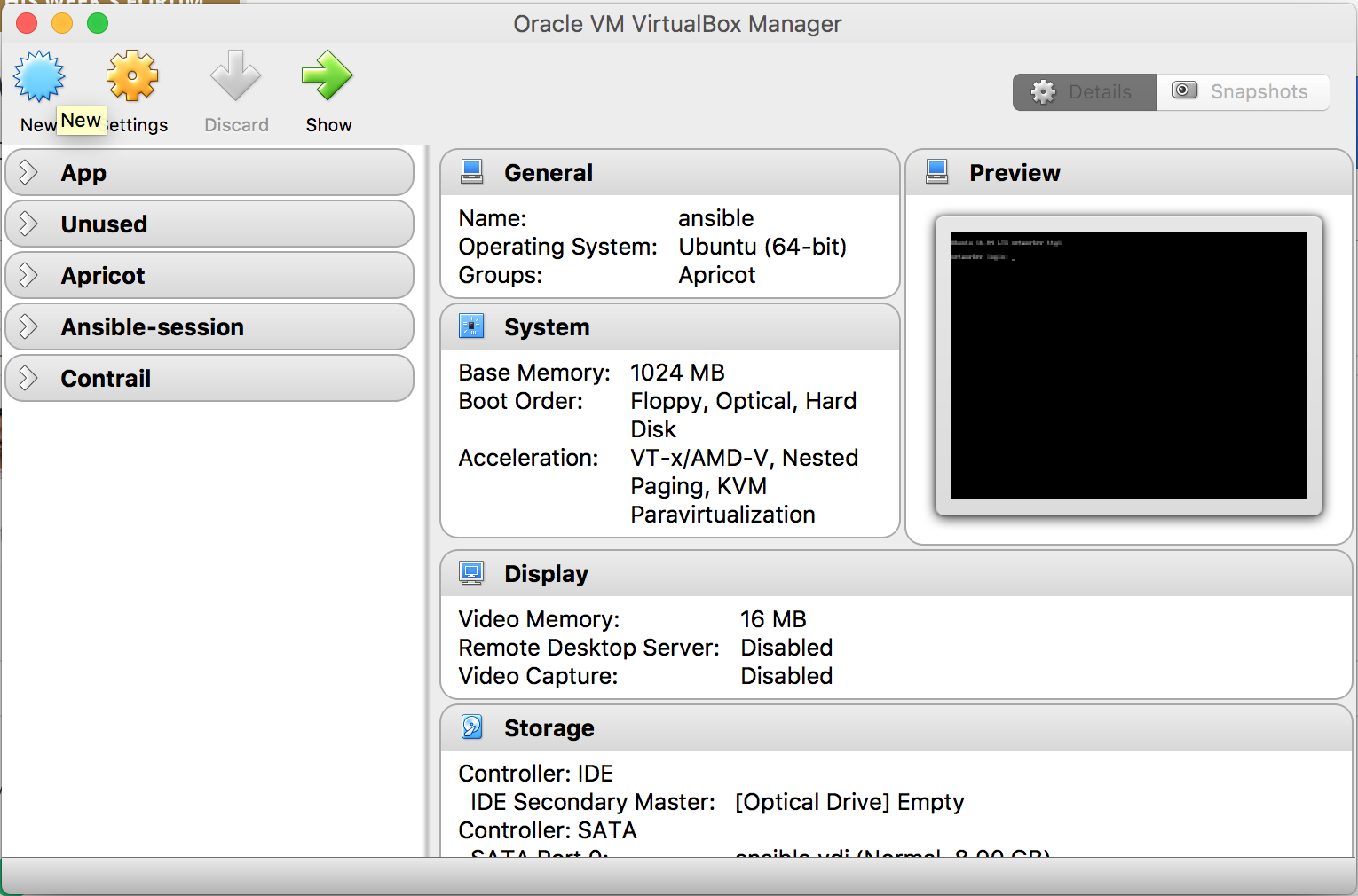


# Ubuntu server setup

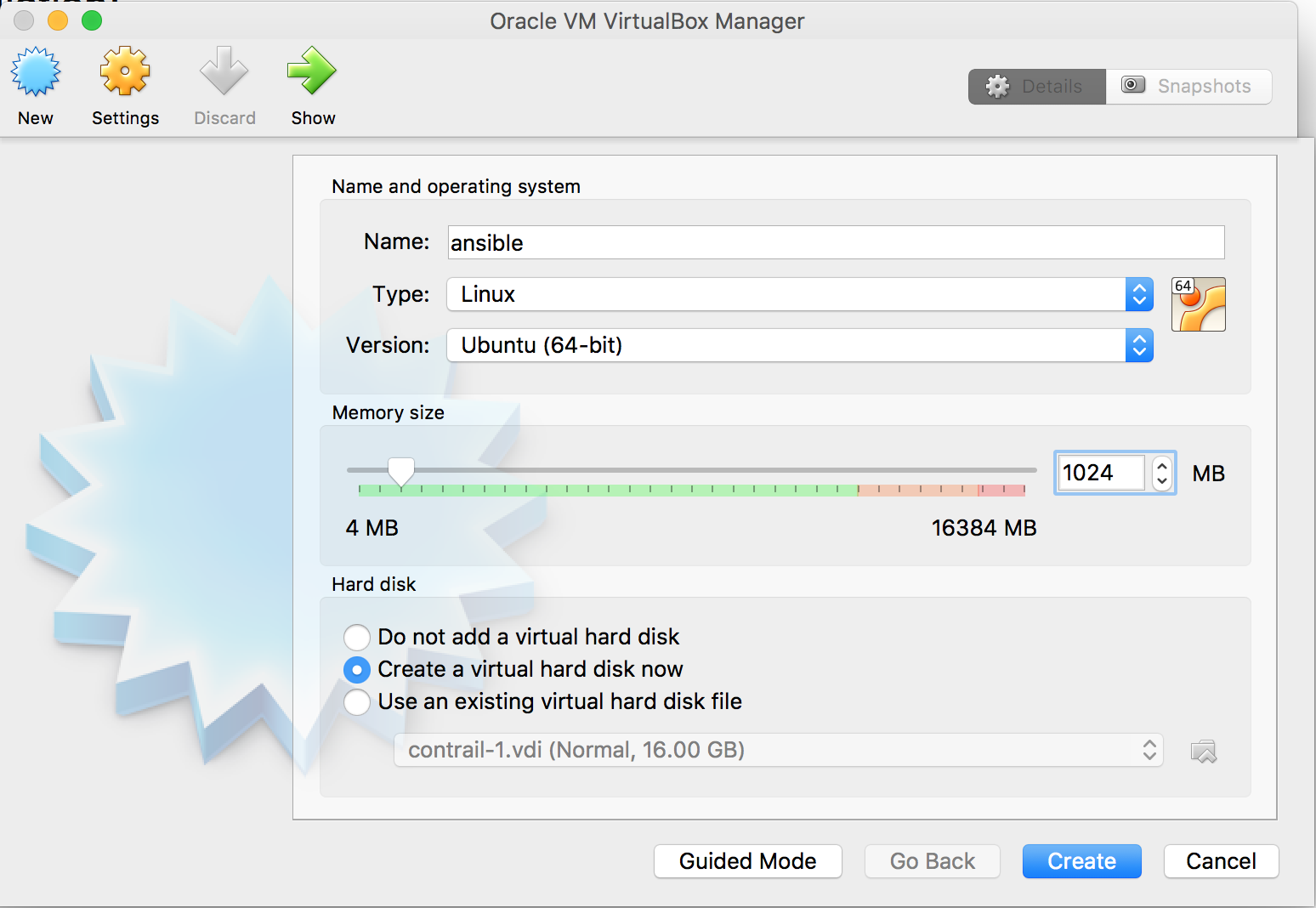
# Ubuntu server installation

Setup an Ubuntu 16.04 image in Virtual Box

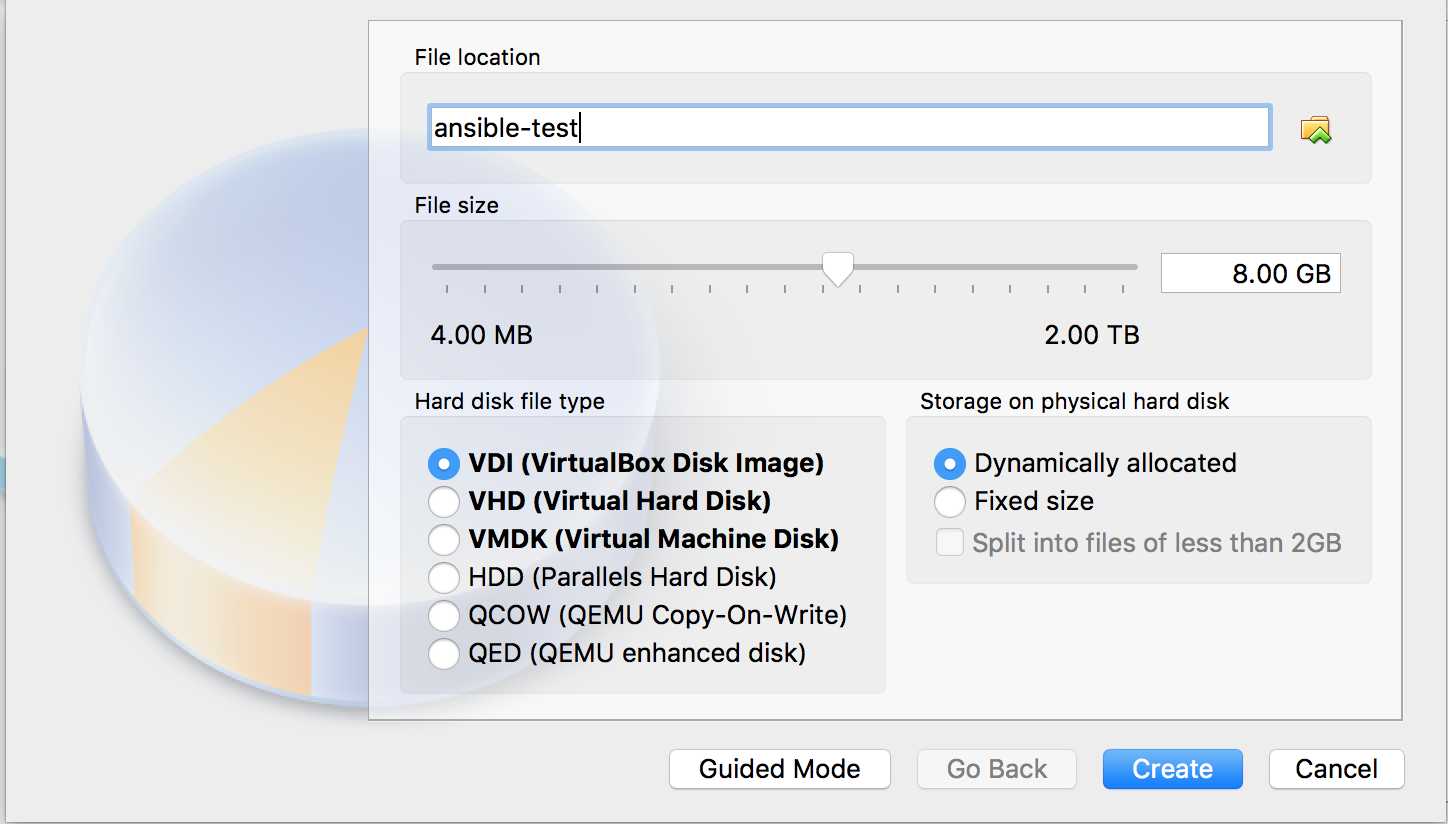
Virtualbox > New



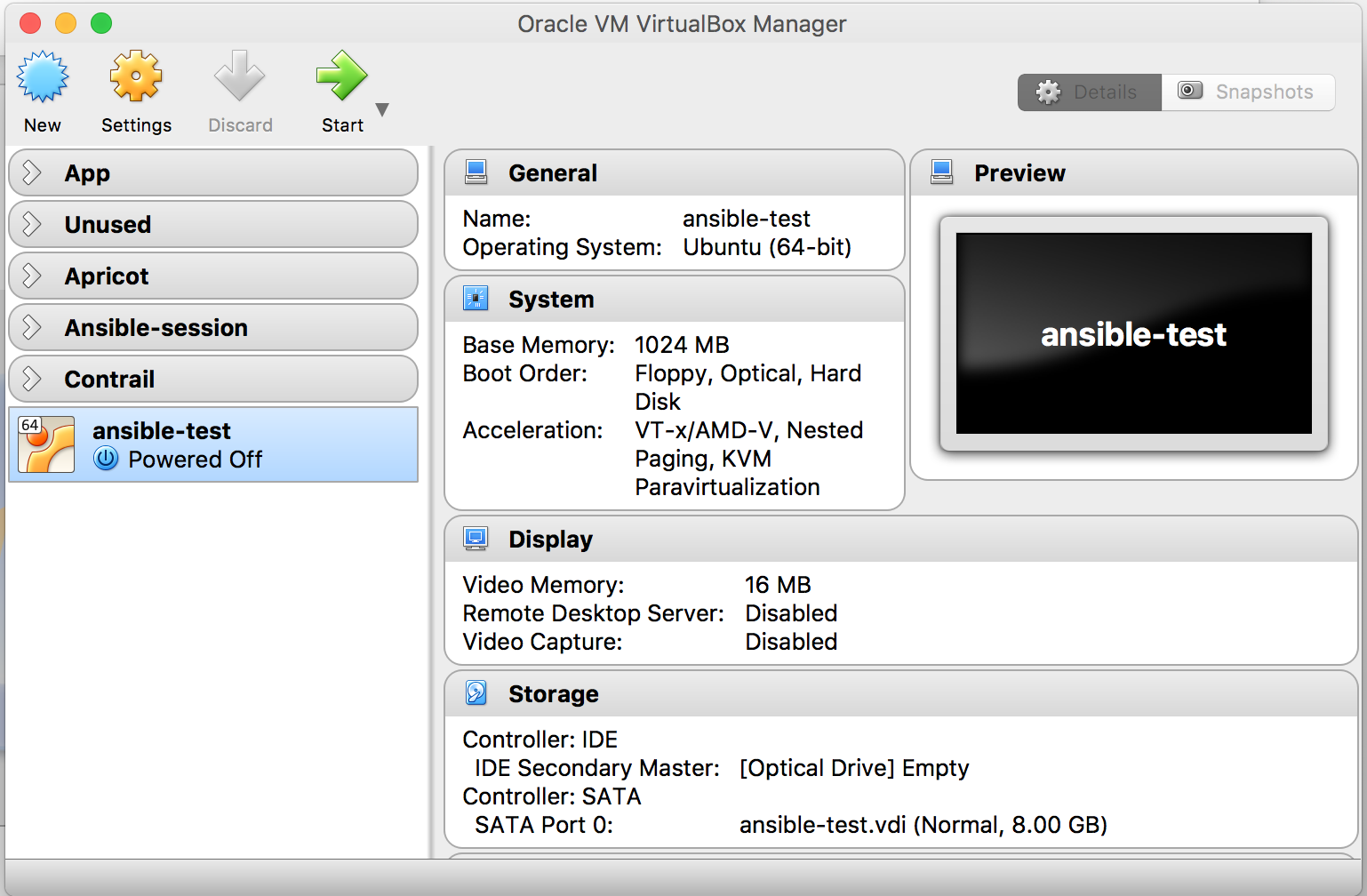
Define Name, Type, Version & Memory



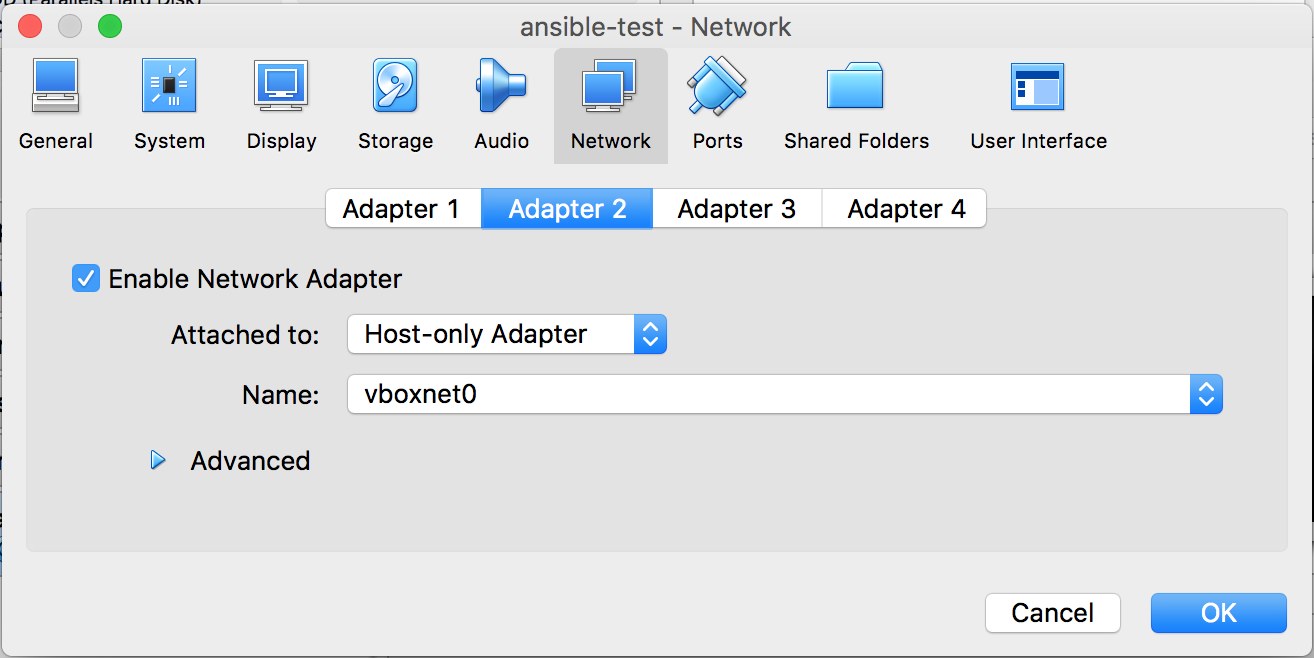
Define the HDD space:



Once done, you should see the new VM in the list. Select the VM and go to Settings:



Under Network, go to Adapter 2. Enable it and Attach to ‘Host-only adapter’



Start the VM to begin Ubuntu server installation:

* Most settings are default
* Set your username and password
* Set the correct timezone
* Install openssh server as a part of the installation

# Ubuntu server configuration

* + 1. Run an update

sudo apt-get update

On the Ubuntu server, we will setup 2 interfaces, first is the default NAT interface that will be used for communication to the internet. The second interface (that we will setup next), will be used for communication with the local machine and the vMX routers. This second interface is the host-only adapter that we added in VBox for this VM.

* + 1. Setup the 2nd interface
       1. Check the name of the 2nd interface

ifconfig –a

* + - 1. Edit the /etc/network/interfaces file to add a static IP for the 2nd interface

sudo vi /etc/network/interfaces

* + - 1. Make the following entries in the file:

auto enp0s10

iface enp0s10 inet static

address 192.168.100.206

netmask 255.255.254.0

dns-nameservers 8.8.8.8 8.8.4.4

* + - 1. Bring up the interface

sudo ifup enp0s10

* + 1. Setup password-less SSH
       1. On local Mac: Generate ssh keys if required

nitinvig-mbp:.ssh nitinvig$ ssh-keygen

Generating public/private rsa key pair.

Enter file in which to save the key (/Users/nitinvig/.ssh/id\_rsa):

/Users/nitinvig/.ssh/id\_rsa already exists.

Overwrite (y/n)? n

* + - 1. On Ubuntu:

mkdir .ssh

touch .ssh/authorized\_keys

chmod 0600 .ssh/authorized\_keys

* + - 1. On local mac:

cat ~/.ssh/id\_rsa.pub | ssh automation@192.168.100.207 'cat >> .ssh/authorized\_keys'

* + 1. Setup ratom & tree on Ubuntu:

sudo curl -o /usr/local/bin/rmate <https://raw.githubusercontent.com/aurora/rmate/master/rmate>

sudo chmod +x /usr/local/bin/rmate

sudo mv /usr/local/bin/rmate /usr/local/bin/ratom

sudo apt-get update && sudo apt-get install

sudo apt install tree

* + 1. On local MAC: Update /etc/hosts

192.168.100.209 ansible-ws

* + 1. On local MAC: Update .ssh/config (for default user & ratom ports)

Host ansible-ws

RemoteForward 52698 localhost:52698

User automation

# 

# 

# Python & Ansible installation

* + 1. Install python

sudo apt-get install python

sudo apt-get install python-pip

pip install --upgrade pip

* + 1. Install packages for cryptography dependencies

sudo apt-get install build-essential libssl-dev libffi-dev python-dev

* + 1. Install Ansible, PyEZ, JSNAPy and Ansible galaxy Juniper module

sudo pip install ansible

ansible —version

sudo pip install junos-eznc

sudo pip install jsnapy

sudo pip install jxmlease

sudo ansible-galaxy install Juniper.junos

* + 1. (not required currently) If you need the git version of jsnapy

sudo pip install -U git+https://github.com/Juniper/jsnapy.git

# Setting up the first vMX

* 1. Create VMs for vMX as per Rusdy’s blog

<https://junipernetworks.sharepoint.com/portals/hub/personal/rkrisman/Running-a-Custom-vMX-Instance-on-Your-Laptop>

<https://junipernetworks.sharepoint.com/portals/hub/personal/rkrisman/Running-a-Custom-vMX-Instance-on-Your-Laptop>

Once you have the 2 VMs setup for RE & FPC, we will setup the network interfaces. This vMX VM uses 3 interface adapters internally as a part of the setup. We will now add 2 additional interfaces for our lab. First interface will be a ‘host-only’ adapter used for communication with the local machine and Ansible Ubuntu server. Second interface will be used for setting up connectivity between the vMXs.

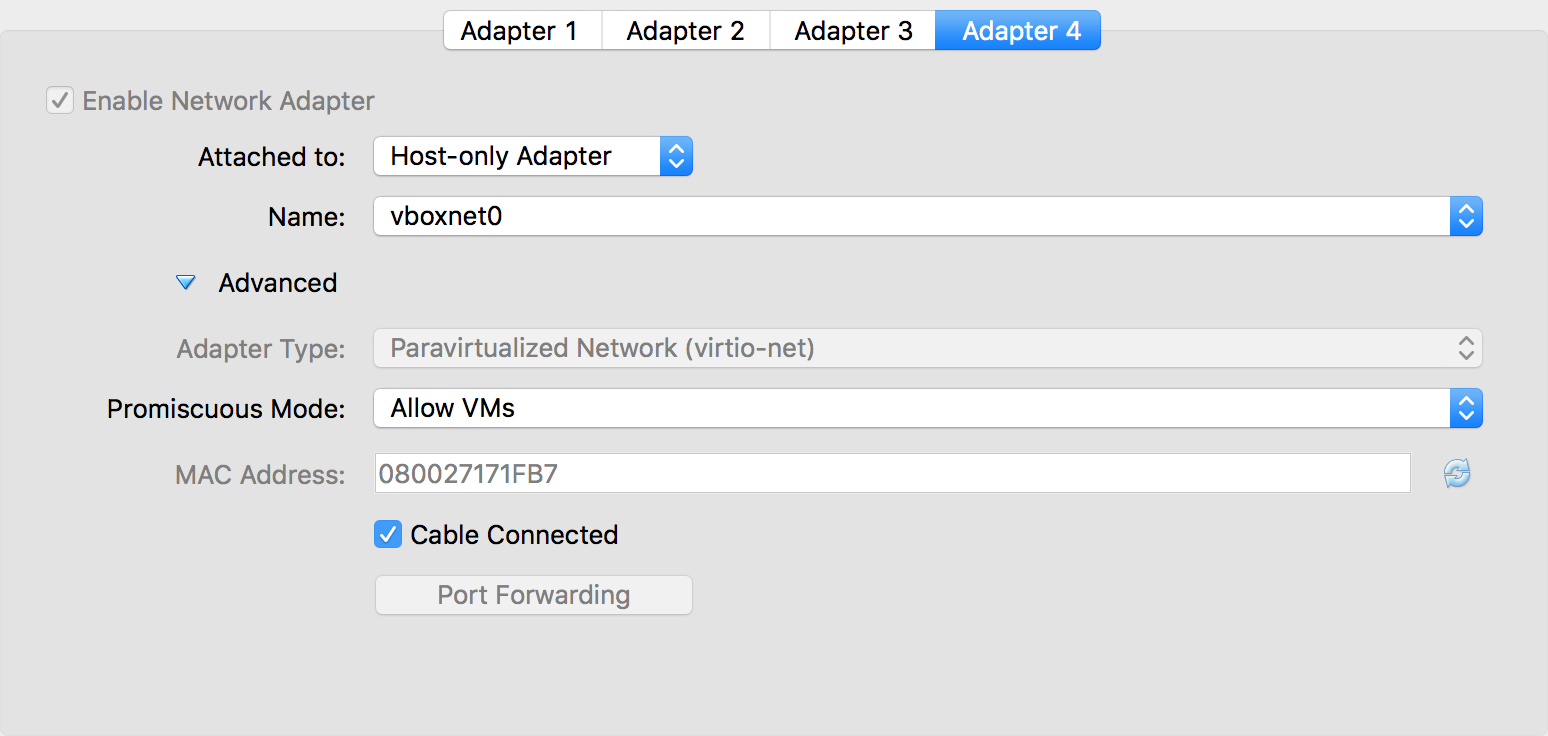
* 1. Setup additional network interfaces
     1. Setting up interfaces in Virtual Box
        1. **On Virtual Box**, select the RE0 VM > Settings > Network > Adapter 4

Attached to: Host-only

Name: vboxnet0

Adapter Type: virtio-net

Promiscuous Mode: Allow VMs



* + - 1. Follow same process on the FPC0 VM
      2. Setting up Adapter 5:

The Virtual Box GUI allows for only upto 4 adapters to be configured. Since this is the 5th adapter on the vMX (3 for internal use + 2 for our lab setup), we will need to set it up using CLI.

**On the Host machine** CLI, manually add the N/W adapter 5 for both the RE & FPC VMs:

VBoxManage modifyvm vmx3-re0 --nic5 intnet

VBoxManage modifyvm vmx3-re0 --intnet5 "spnet"

VBoxManage modifyvm vmx3-re0 --nicpromisc5 allow-vms

VBoxManage modifyvm vmx3-re0 --nictype5 virtio

VBoxManage modifyvm vmx3-fpc0 --nic5 intnet

VBoxManage modifyvm vmx3-fpc0 --intnet5 "spnet"

VBoxManage modifyvm vmx3-fpc0 --nicpromisc5 allow-vms

VBoxManage modifyvm vmx3-fpc0 --nictype5 virtio

\*vmx3-re0 and vmx3-fpc0 are the names of the 2 VMs.

* + - 1. Confirm that the network adapters are correctly added:

nitinvig-mbp:.ssh nitinvig$ vboxmanage showvminfo vmx3-re0 | grep NIC

NIC 1: MAC: 080027F61026, Attachment: Bridged Interface 'en0: Wi-Fi (AirPort)', Cable connected: on, Trace: off (file: none), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none

NIC 2: MAC: 080027777C45, Attachment: Internal Network 'VMX3-BACKPLANE', Cable connected: on, Trace: off (file: none), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none

NIC 3: MAC: 080027411CEC, Attachment: Internal Network 'UNUSED', Cable connected: on, Trace: off (file: none), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none

NIC 4: MAC: 080027171FB7, Attachment: Host-only Interface 'vboxnet0', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: allow-vms, Bandwidth group: none

**NIC 5: MAC: 0800273D3BD0, Attachment: Internal Network 'spnet', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: allow-vms, Bandwidth group: none**

nitinvig-mbp:.ssh nitinvig$ vboxmanage showvminfo vmx3-fpc0 | grep NIC

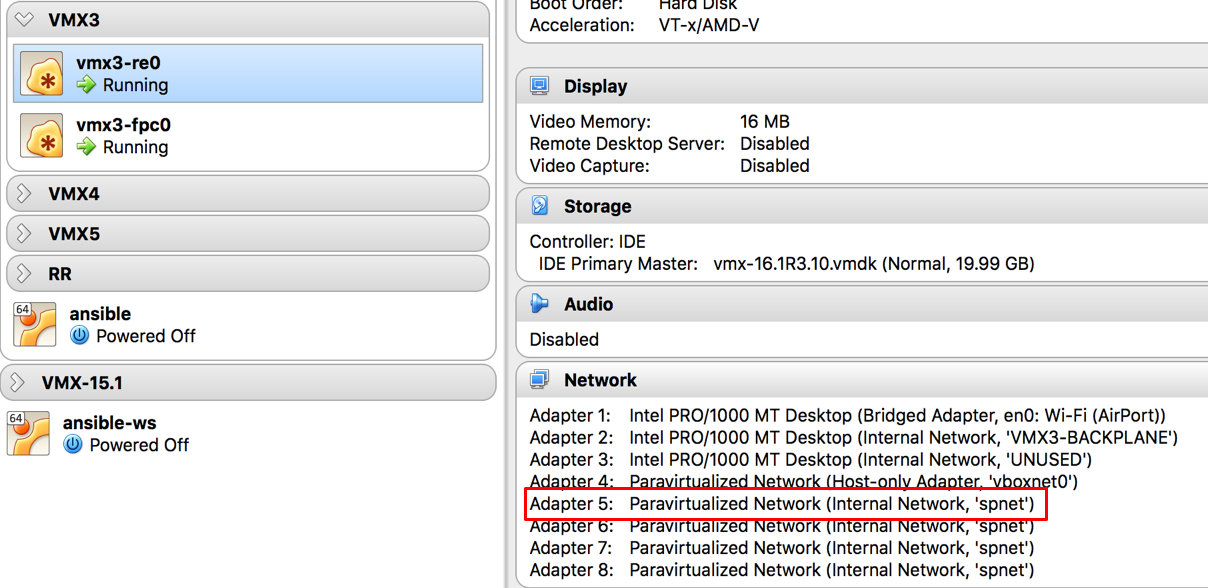
NIC 1: MAC: 0800275F7251, Attachment: Bridged Interface 'en0: Wi-Fi (AirPort)', Cable connected: on, Trace: off (file: none), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none

NIC 2: MAC: 08002745F182, Attachment: Internal Network 'VMX3-BACKPLANE', Cable connected: on, Trace: off (file: none), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none

NIC 3: MAC: 0800270CD469, Attachment: Internal Network 'UNUSED', Cable connected: on, Trace: off (file: none), Type: 82540EM, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: deny, Bandwidth group: none

NIC 4: MAC: 08002701F6EB, Attachment: Host-only Interface 'vboxnet0', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: allow-vms, Bandwidth group: none

**NIC 5: MAC: 0800276E069E, Attachment: Internal Network 'spnet', Cable connected: on, Trace: off (file: none), Type: virtio, Reported speed: 0 Mbps, Boot priority: 0, Promisc Policy: allow-vms, Bandwidth group: none**



* 1. Boot the RE & FPC VMs

Wait for the RE-FPC communication to get established. You should see the Virtual FPC come up on the vMX.

automation@HAN> show chassis hardware

Hardware inventory:

Item             Version  Part number  Serial number     Description

Chassis                                VMXcac5           MX960

Midplane

Routing Engine 0                                         RE-VMX

CB 0                                                     VMX SCB

CB 1                                                     VMX SCB

FPC 0                                                    Virtual FPC

  CPU            Rev. 1.0 RIOT         123XYZ987

  MIC 0                                                  Virtual 20x 1GE(LAN) SFP

    PIC 0                 BUILTIN      BUILTIN           Virtual 10x 1GE(LAN) SFP

    PIC 1                 BUILTIN      BUILTIN           Virtual 10x 1GE(LAN) SFP

  MIC 1                                                  Virtual 4x 10GE(LAN) XFP

    PIC 2                 BUILTIN      BUILTIN           Virtual 2x 10GE(LAN) XFP

    PIC 3                 BUILTIN      BUILTIN           Virtual 2x 10GE(LAN) XFP

* 1. Mapping the em interface to ge-0/0/x

On the vMX, Configure the em3 & em4 inteface MAC addresses on the ge-0/0/0 & ge-0/0/1 interfaces respectively:

nitin@HAN# run show interfaces em3 | match address

Current address: 08:00:27:17:1f:b7, Hardware address: 08:00:27:17:1f:b7

[edit]

nitin@HAN# show interfaces ge-0/0/0

mac 08:00:27:17:1f:b7;

nitin@HAN# run show interfaces em4 | match address

Current address: 08:00:27:3D:3B:D0, Hardware address: 08:00:27:3D:3B:D0

[edit]

nitin@HAN# show interfaces ge-0/0/1

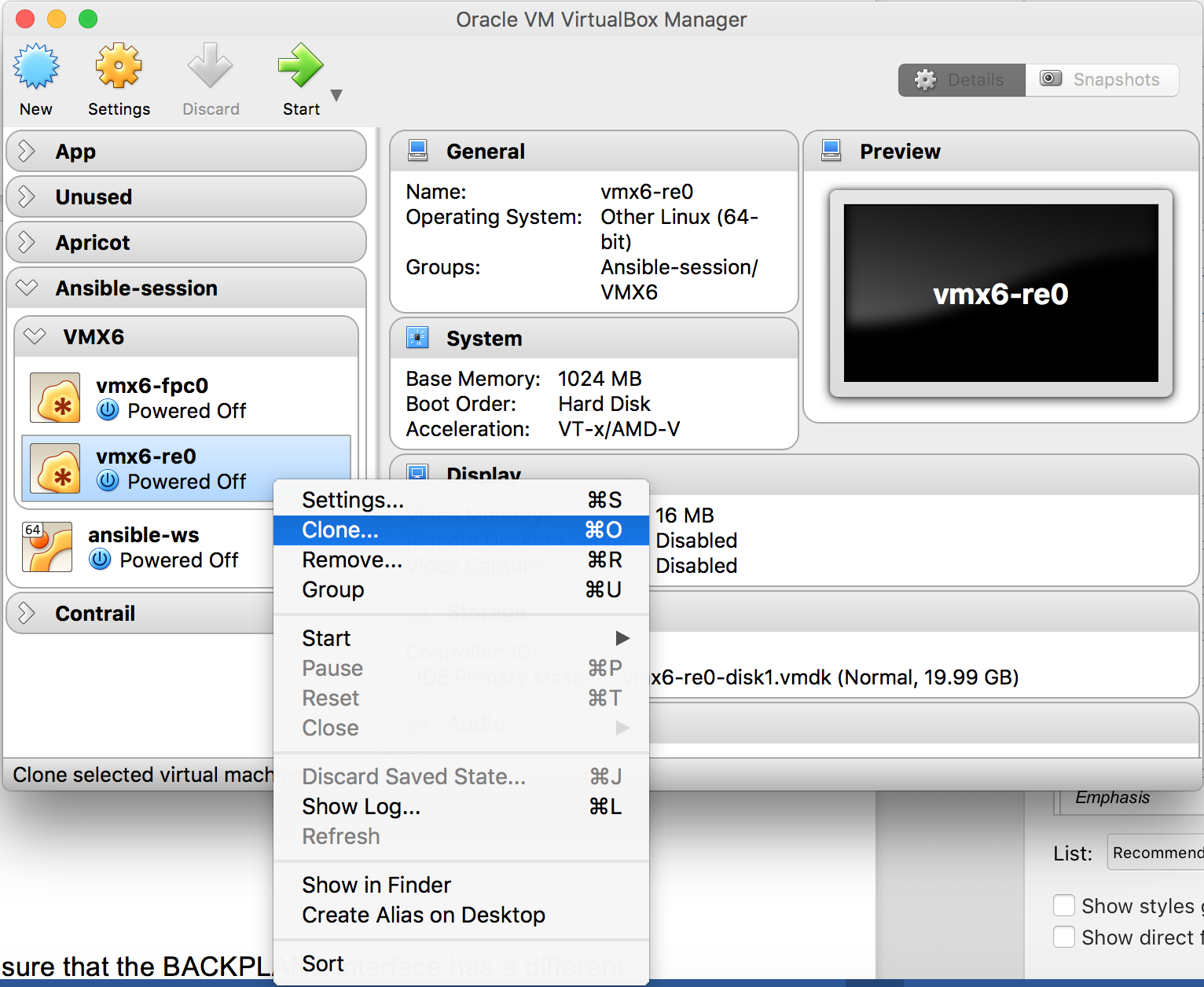
mac 08:00:27:3D:3B:D0;

That’s it. Our vMX is now ready for the lab.

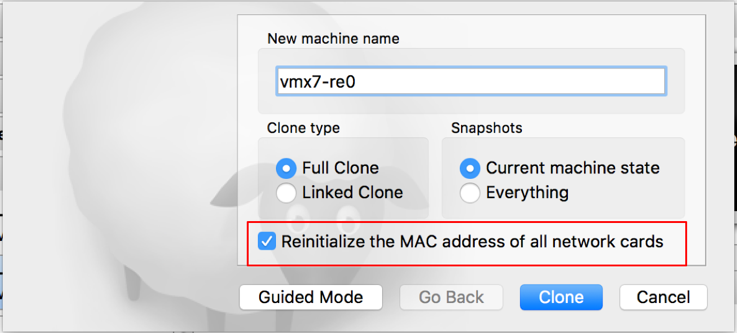
# Building vMX clones

Ok, so we are not quite done yet. Since our lab setup requires 3 vMXs, we are going to create the other 2 by cloning the vMX that we just created and make some changes to it. Both the RE & FPC VMs need to be cloned individually.

* 1. Cloning the RE VM
     1. Right-click on the RE VM and select Clone:

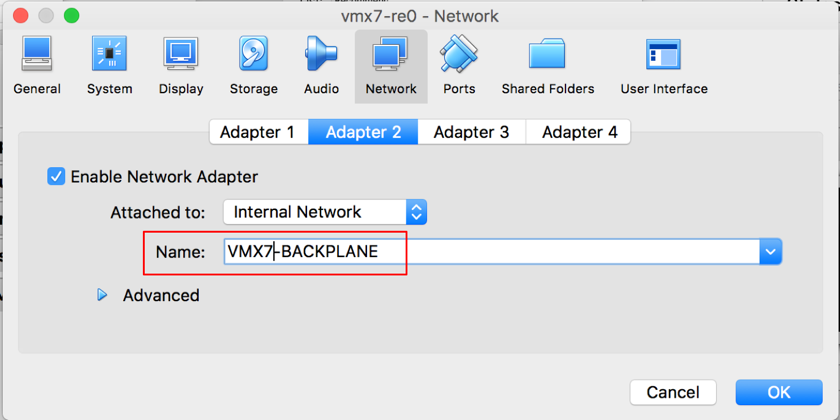


* + 1. Give the VM a new name & reinitialize the MAC addresses:



* + 1. On the cloned VM, go to Settings > Network > Adapter 2. Give a unique name for the backplane network adapter.

**NOTE:** Every vMX must have a unique backplane interface name. Each vMX’s RE & FPC VMs share the same backplane interface name. This is important to ensure that RE & FPC communication for each vMX is isolated.



* 1. Repeat the same process for the FPC0 VM.
  2. Boot the RE & FPC VMs.
  3. Follow Step 6.4 to map the emX and ge-0/0/x interface MAC address

Repeat the same process for the 3rd VM. This completes the vMX lab setup. You should now have 3 VMX instances with 2 interfaces each.

# Initial vMX configuration

Each vMX will be setup with some basic configuration via console to communicate with the Ansible server. In a production network, these can also be configure via Ansible & PyEZ using support for console. However, for this lab we will set this up manually on each vMX.

Configure the following tasks for each vMX:

* Root authentication
* Assign an IP address to ge-0/0/0 interface in the same network as the Ansible server (Step 4.2.3)
* Configure ssh & netconf ssh services
* Create a login user: automation/automation123

automation@HAN> show configuration system

root-authentication {

    encrypted-password "$5$yHNaE1Id$W8NEIrd/nlgq/BesMRwmNW6g5mQXg5h0QIw3ms/r.N0"; ## SECRET-DATA

}

login {

    user automation {

        uid 1001;

        class super-user;

        authentication {

            encrypted-password "$5$H6BGyUsX$lOL4ZUYUihaGZeOm/m8ZAYBJu2MyT1U9sVajN1E3fs5"; ## SECRET-DATA

        }

    }

}

services {

    ssh;

    netconf {

        ssh;

    }

}

automation@HAN> show configuration interfaces ge-0/0/0

mac 08:00:27:17:1f:b7;

unit 0 {

    family inet {

        address 192.168.100.223/24;

    }

}