

# HomeLab Project

Imran Hakim Roslan  
imrh@protonmail.com

January 30, 2021

## 1 Introduction

A virtual environment for computing infrastructure using Qemu-KVM hypervisor. Using the hypervisor, hardware-assisted virtualization is made possible thus providing various form of IT infrastructure as for instance software defined networking and data center facility. The hypervisor is hosted on Fedora 32 Linux.

In this project, a relational database management system MariaDB, is deployed in virtualized Debian 10 Linux operating system on-top of Qemu-KVM hypervisor. The hypervisor then is being managed by libvirt API as the orchestration layer.

## 2 Hardware Configurations

The Qemu-KVM hypervisor is set up on Fedora 32 Linux operating system machine with an Intel processor i3-2100 providing 2 compute cores with hyperthreading. The system has been provided with total of 10 GB of RAM. \*Note: The motherboard has issue with current dual-channel memory RAM configuration.



Figure 1: Host hypervisor system

To utilize KVM hardware-assisted virtualization, Intel VT-x has been toggled-on in the motherboard BIOS. This provide device hardware emulation for the guest operating system to operate on.

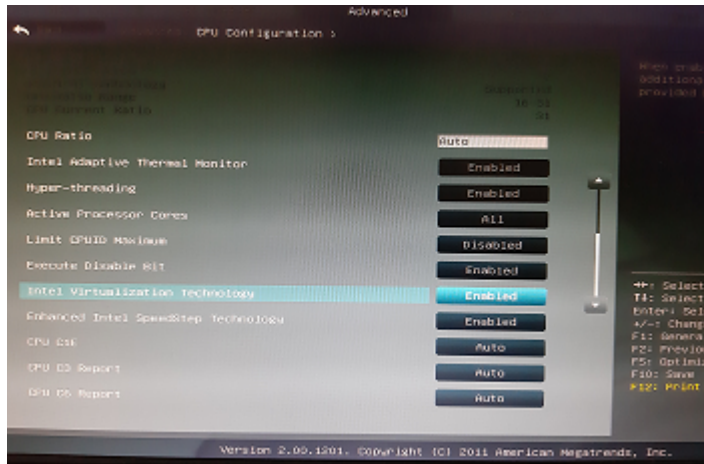
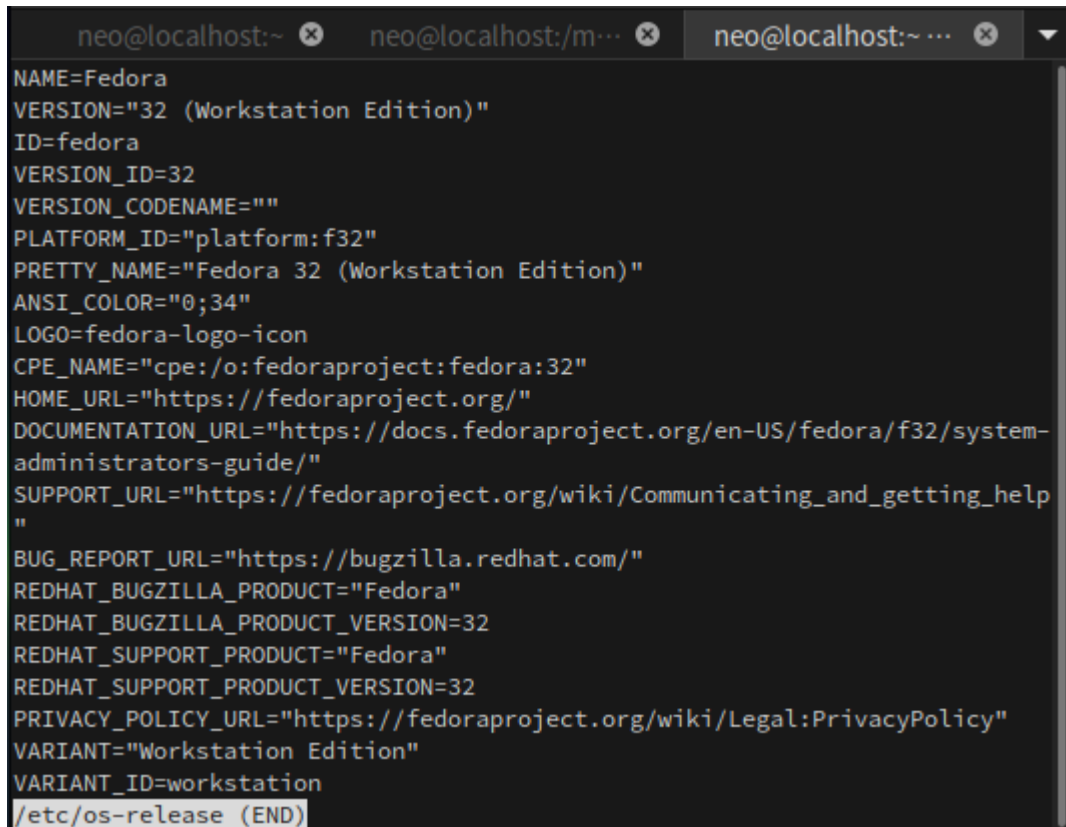


Figure 2: Enabling Intel VT-x in BIOS

### 3 Software Configurations

Host operating system is based on Fedora 32 Linux distro. To manage the hypervisor, libvirt virtualization API is used to coordinate the virtual platform. The libvirt API served as orchestration layer to control how the virtualized environment being structured. This project uses libvirt API to manage the networking of guest virtual machine.

A terminal window with three tabs, all showing 'neo@localhost:~'. The active tab displays the output of the 'cat /etc/os-release' command. The output lists various system identifiers and URLs for Fedora 32 (Workstation Edition).

```
NAME=Fedora
VERSION="32 (Workstation Edition)"
ID=fedora
VERSION_ID=32
VERSION_CODENAME=""
PLATFORM_ID="platform:f32"
PRETTY_NAME="Fedora 32 (Workstation Edition)"
ANSI_COLOR="0;34"
LOGO=fedora-logo-icon
CPE_NAME="cpe:/o:fedoraproject:fedora:32"
HOME_URL="https://fedoraproject.org/"
DOCUMENTATION_URL="https://docs.fedoraproject.org/en-US/fedora/f32/system-administrators-guide/"
SUPPORT_URL="https://fedoraproject.org/wiki/Communicating_and_getting_help"
BUG_REPORT_URL="https://bugzilla.redhat.com/"
REDHAT_BUGZILLA_PRODUCT="Fedora"
REDHAT_BUGZILLA_PRODUCT_VERSION=32
REDHAT_SUPPORT_PRODUCT="Fedora"
REDHAT_SUPPORT_PRODUCT_VERSION=32
PRIVACY_POLICY_URL="https://fedoraproject.org/wiki/Legal:PrivacyPolicy"
VARIANT="Workstation Edition"
VARIANT_ID=workstation
/etc/os-release (END)
```

Figure 3: Host Operating System

## 4 Storage

For this project, the guest operating system storage disk creation is done by using `qemu-img` command and stored in local storage in the host.

## 5 Networking

The host machine is operated with 1 Realtek network interface card (NIC) shipped with the motherboard and extra add-on 4 Intel NIC that was plugged in through PCIe slot. A router that act as a network gateway is connected to the host machine. The connection is wired on interface 1 and interface 2 of LAN from the TP-Link AX50 router to Realtek NIC and Intel NIC 'enp4s0f1' interface respectively. Static IP has been set on the Realtek NIC at 192.168.1.15 with network mask 255.255.255.0.



Figure 4: Host network interface cards



Figure 5: TP-Link AX50 router

## 6 Deployment Procedure

### 6.1 Setup Virtual Machine Storage Disk

A storage disk sized 20 GB stored in `/mnt/B/VM/debian10_mariadb/` in host local machine is created using `qemu-img`.

```
[neo@localhost ~]$ qemu-img create -f raw -o size=20G /mnt/B/VM/debian10_mariadb/debian10_mariadb_server.img
Formatting '/mnt/B/VM/debian10_mariadb/debian10_mariadb_server.img', fmt=raw size=21474836480
```

Figure 6: Creating a raw storage disk for guest operating system

### 6.2 Setup a Network for Guest Operating System

Setting up a network interface for the Debian 10 guest operating system (OS) using `macvtap` Qemu-KVM network type for easy interfacing with MariaDB database that is to be installed later on in the guest OS.

Method procedure for the guest network interface is being discussed below:

1. Create an XML file named "macvtap-enp4s0f1.xml".
2. Insert code listing in Figure 7 to define the networking interface for the guest OS. The `macvtap` is forwarded to 'enp4s0f1' Intel NIC interface connected to AX50 LAN 2 interface.

```
<network>
  <name>macvtap-enp4s0f1</name>
  <forward mode="bridge">
    <interface dev="enp4s0f1"/>
  </forward>
</network>
```

Figure 7: XML file for the guest OS network definition

3. Using `libvirt` to define the XML file as a network interface for the guest OS to connect with (Figure 8):

```
[root@localhost ~]# virsh net-define /mnt/B/VM/debian10_mariadb/macvtap-enp4s0f1.xml
Network macvtap-enp4s0f1 defined from /mnt/B/VM/debian10_mariadb/macvtap-enp4s0f1.xml
```

Figure 8: Defining the network using `libvirt` API

4. Again using `libvirt` API to start the network device interface.

```
[root@localhost ~]# virsh net-start macvtap-enp4s0f1
Network macvtap-enp4s0f1 started
```

Figure 9: Start the macvtap network interface

## 6.3 Installing The Virtual Machine

The virtual machine (VM) will be installed with `virt-install` command. The Debian 10 VM is equipped with 1 vcpu, 1024 Bytes of RAM, and assigned with the disk and network interface created earlier. The installation procedure is illustrated below:

1. Using `virt-install` to install the guest OS virtual machine.

```
[root@localhost ~]# virt-install \
> --name debian_mariadb_server \
> --ram 1024 \
> --disk path=/mnt/B/VM/debian10_mariadb/debian10_mariadb_server.img \
> --vcpus 1 \
> --os-type Linux \
> --os-variant debian10 \
> --network network:macvtap-enp4s0f1,model=virtio \
> --graphics vnc,port=5999 \
> --console pty,target_type=serial \
> --cdrom /mnt/A/20201019_1723_Backups/ISO/debian-10.4.0-amd64-netinst.iso

Starting install...
```

Figure 10: Instantiate the virtual machine

2. The guest OS installation process will continue in virt-viewer.

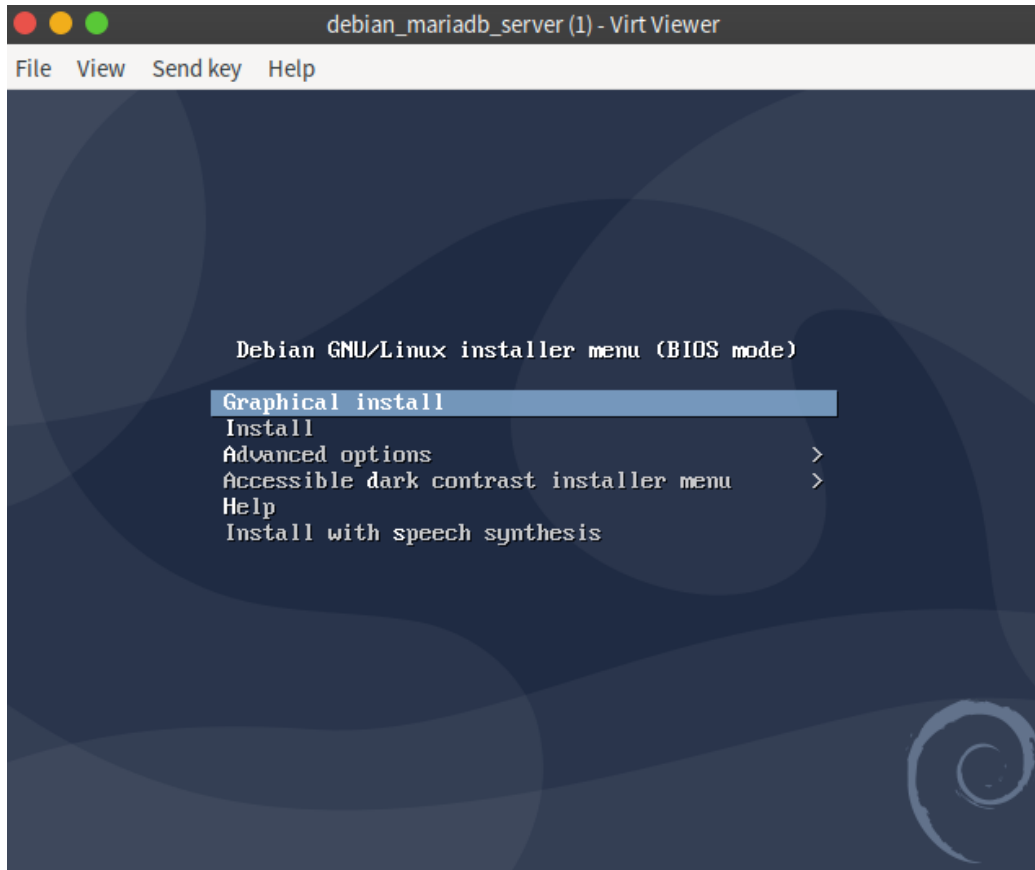


Figure 11: Installing the guest OS in virt-viewer

## 6.4 Deploy MariaDB Database Server

Install the MariaDB server using apt install command (Figure 12) in the guest OS and verify the database server is running (Figure 13).

```
sudo apt install mariadb-server
```

Figure 12: Installing MariaDB database server



```

neo-deb@deb-mariadb-server:~$ systemctl status mariadb.service
● mariadb.service - MariaDB 10.3.27 database server
   Loaded: loaded (/lib/systemd/system/mariadb.service; enabled; vendor preset: enabled)
   Active: active (running) since Wed 2021-01-27 22:52:43 PST; 4h 31min ago
     Docs: man:mysqld(8)
           https://mariadb.com/kb/en/library/systemd/
   Process: 334 ExecStartPre=/usr/bin/install -m 755 -o mysql -g root -d /var/run/mysqld (code=exi
   Process: 342 ExecStartPre=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=ex
   Process: 344 ExecStartPre=/bin/sh -c [ ! -e /usr/bin/galera_recovery ] && VAR= || VAR='cd /us
   Process: 422 ExecStartPost=/bin/sh -c systemctl unset-environment _WSREP_START_POSITION (code=e
   Process: 424 ExecStartPost=/etc/mysql/debian-start (code=exited, status=0/SUCCESS)
 Main PID: 391 (mysqld)
    Status: "Taking your SQL requests now..."
     Tasks: 31 (limit: 1149)
   Memory: 108.3M
    CGroup: /system.slice/mariadb.service
            └─391 /usr/sbin/mysqld
lines 1-16/16 (END)

```

Figure 13: Verify the MariaDB server is running

## 7 Appendix

### A Improvements

Various ways can be improved with the current virtualize environment such as:

1. Having a centralized storage server and connect to host or guest operating system using iSCSI connection.
2. Utilize a VPN using OpenVPN to make the network accessible from anywhere.
3. Deploy Ansible to automate virtual environment management.
4. Using pfSense to tighten up the network firewall.
5. Monitor and manage guest OS remotely by deploying kimchi.