Installing Vagrant on Ubuntu

Vagrant is a command-line tool for building and managing virtual machines.

By default, Vagrant can provision machines on top of VirtualBox, Hyper-V, and Docker. Other providers such as Libvirt (KVM), VMware and AWS can be installed via the Vagrant plugin system.

Vagrant is typically used by developers to set up a development environment that works across multiple operating systems.

We will provision the virtual machines on top of VirtualBox.

```
$ sudo apt update
$ sudo apt install virtualbox
$ sudo apt install qemu libvirt-daemon-system libvirt-clients libxslt-dev
libxml2-dev libvirt-dev zlib1g-dev ruby-dev ruby-libvirt ebtables dnsmasq-
base
```

At the time of writing this article, the latest stable version of Vagrant is version 2.2.9. Visit the Vagrant downloads page to see if there is a new version of Vagrant available.

```
$ curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -
$ wget
https://releases.hashicorp.com/vagrant/2.2.18/vagrant_2.2.18_linux_amd64.zi
p
$ unzip  vagrant_2.2.18_linux_amd64.zip
$ sudo mv vagrant /usr/local/bin
```

To verify that the installation was successful, run the following command that will print the Vagrant version:

```
$ vagrant --version
```

Now, install vagrant-libvirt plugin using command:

```
$ vagrant plugin install vagrant-libvirt
```

You also need to install vagrant-mutate plugin which converts vagrant boxes to work with different providers.

```
$ vagrant plugin install vagrant-mutate
$ sudo apt-get install libarchive-tools
```

Getting Started with Vagrant

Creating a Vagrant project is as simple as setting up the project root directory and defining a Vagrantfile.

Run the following commands to create the directory and cd into it with:

Alternatively you can tell Vagrant to permanently use libvirt as default provider by adding the following environment variable.

```
$ export VAGRANT_DEFAULT_PROVIDER=libvirt
```

Next, run the following command to start the virtual machine:

```
$ vagrant up --provider=libvirt


$ mkdir ~/my-vagrant-project
$ cd ~/my-vagrant-project
```

Create a Vagrantfile with content similar to below:

```
# -*- mode: ruby -*-
# vi: set ft=ruby :

ENV['VAGRANT_DEFAULT_PROVIDER'] = 'libvirt'


Vagrant.configure("2") do |config|

##### DEFINE VM #####
config.vm.define "cent-01" do |config|
config.vm.hostname = "cent-01"
config.vm.box = "centos/7"
config.vm.box_check_update = false
config.vm.network "private_network", ip: "192.168.18.9"
config.vm.provider :libvirt do |v|
    v.memory = 1024
    end
end
end
```

In this example, we’re going to use a ready template. Let add CentOS 7 and CentOS 6 boxes.

```
$ vagrant box add centos/7 --provider=libvirt
==> box: Loading metadata for box 'centos/7'
box: URL: https://vagrantcloud.com/centos/7
==> box: Adding box 'centos/7' (v1902.01) for provider: libvirt

$ vagrant box add centos/8 --provider=libvirt
```

Add Ubuntu 20.04 Vagrant box:

```
$ vagrant box add generic/ubuntu2004 --provider=libvirt
```

Check the list of boxes presents locally.

```
$ vagrant box list
```

Verify if VM is running in **Libvirt KVM**

You can verify if the CentOS 7 VM is really running in Libvirt KVM provider from Virsh command line interface.

```
$ virsh list
```

Vagrant will create a Linux bridge on the host system.

```
$ brctl show virbr1
```

To ssh to the VM, use vagrant ssh command.

```
$ vagrant ssh
```

To output .ssh/config valid syntax for connecting to this environment via ssh, run ssh-config command. You'll need to place provided output under ~/.ssh/config directory to ssh.

```
$ vagrant ssh-config
$ vagrant ssh-config >>~/.ssh/config
```

Then use ssh command to log in with name configured above:

```
$ ssh cent-01
Last login: Fri Apr 19 07:40:42 2019 from 192.168.121.1
[vagrant@cent-01 ~]$
```

To shut down the VM, run:

```
$ vagrant halt
==> cent-01: Halting domain…
```

To set VM to its initial state by cleaning all data, use vagrant destroy:

```
$ vagrant destroy
```