**How To Create Rocky Linux 8 Vagrant Boxes Using Packer**

In this guide we will look at the steps that you can use to generate Rocky Linux 8 Vagrant Boxes Using Packer templates. Vagrant is an open source solution that allows you to build and maintain portable virtual software development environments for your Virtualization Providers – KVM, VirtualBox, Hyper-V, VMware, Docker Containers, and AWS. We’ll use [Packer](https://www.packer.io/), which is is a tool for creating identical machine images for multiple platforms from a single source configuration.

For this task we’ll use [Bento project](https://github.com/chef/bento) which encapsulates Packer templates for building [Vagrant](https://www.vagrantup.com/) base boxes. This simplifies Vagrant Boxes creation without the need to write your own templates; which is time consuming and difficult. For some templates the Vagrant Boxes have been build for you and published to the [bento org](https://app.vagrantup.com/bento) on Vagrant Cloud.

So follow the next steps to create your own Rocky Linux 8 Vagrant Boxes Using Packer and Bento Project templates.

**Step 1: Install Packer and Vagrant Tools**

The first step is installation of Packer and Vagrant which are dependencies for the next sections. The good news is that HashiCorp now maintains the package repositories for various Linux Distributions.

**Install Packer and Vagrant on Ubuntu / Debian**

Install required tools for repository addition into your system:

sudo apt update

sudo apt install wget apt-transport-https gnupg2

Import repository GPG key:

curl -fsSL https://apt.releases.hashicorp.com/gpg | sudo apt-key add -

Add HashiCorp APT repository by running the commands below in your terminal:

sudo apt-add-repository "deb [arch=amd64] https://apt.releases.hashicorp.com $(lsb\_release -cs) main"

sudo apt update

Once the repository is added install Packager:

$ sudo apt install packer

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following NEW packages will be installed:

packer

0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.

Need to get 31.7 MB of archives.

After this operation, 145 MB of additional disk space will be used.

Get:1 https://apt.releases.hashicorp.com focal/main amd64 packer amd64 1.7.3 [31.7 MB]

Fetched 31.7 MB in 1s (43.0 MB/s)

Selecting previously unselected package packer.

(Reading database ... 64995 files and directories currently installed.)

Preparing to unpack .../packer\_1.7.3\_amd64.deb ...

Unpacking packer (1.7.3) ...

Setting up packer (1.7.3) ...

Do the same for Vagrant:

$ sudo apt install vagrant

Reading package lists... Done

Building dependency tree

Reading state information... Done

The following NEW packages will be installed:

vagrant

0 upgraded, 1 newly installed, 0 to remove and 4 not upgraded.

Need to get 40.9 MB of archives.

After this operation, 115 MB of additional disk space will be used.

Get:1 https://apt.releases.hashicorp.com focal/main amd64 vagrant amd64 2.2.16 [40.9 MB]

Fetched 40.9 MB in 1s (64.4 MB/s)

Selecting previously unselected package vagrant.

(Reading database ... 64998 files and directories currently installed.)

Preparing to unpack .../vagrant\_2.2.16\_amd64.deb ...

Unpacking vagrant (2.2.16) ...

Setting up vagrant (2.2.16) ...

Confirm installation by checking software versions:

$ packer version

Packer v1.7.3

$ vagrant version

Installed Version: 2.2.16

Latest Version: 2.2.16

**Install Packer and Vagrant on CentOS / Fedora / RHEL / Amazon Linux**

Add the repository and install the packages:

**CentOS / RHEL**:

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/RHEL/hashicorp.repo

sudo yum -y install packer vagrant

For **Fedora** run the commands below:

sudo dnf install -y dnf-plugins-core

sudo dnf config-manager --add-repo https://rpm.releases.hashicorp.com/fedora/hashicorp.repo

sudo dnf -y install packer vagrant

**Amazon Linux:**

sudo yum install -y yum-utils

sudo yum-config-manager --add-repo https://rpm.releases.hashicorp.com/AmazonLinux/hashicorp.repo

sudo yum -y install packer vagrant

**Install Vagrant and Packer on macOS**

For macOS users the installation is done as shown below:

brew tap hashicorp/tap

brew install hashicorp/tap/packer

brew install vagrant

**Step 2: Install Virtualization providers of your choice**

You have to install at least one of the following virtualization providers:

* [VirtualBox](https://www.virtualbox.org/)
* [VMware Fusion](https://www.vmware.com/products/fusion.html)
* [VMware Workstation](https://www.vmware.com/products/workstation-pro.html)
* [Parallels Desktop](https://www.parallels.com/products/desktop/) also requires [Parallels Virtualization SDK](https://www.parallels.com/products/desktop/download/)
* [KVM](https://www.linux-kvm.org/page/Main_Page)
* [Hyper-V](https://docs.microsoft.com/en-us/virtualization/hyper-v-on-windows/about/)

You can start with the Virtualization provider for which you want to run Vagrant machine on.

**Step 3: Clone Bento Project on Github**

Install git tool if doesn’t exist in your machine:

***# CentOS / RHEL / Fedora***

sudo yum -y install git

*#* ***Ubuntu / Debian***

sudo apt update

sudo apt install git

***# macOS***

brew install git

Once git package is installed use it to download Bento source from Github:

git clone https://github.com/chef/bento.git

The folder *bento* has a number of files and directories inside:

$ ls -1 bento/

bento.gemspec

bin

builds

builds.yml

CHANGELOG.md

CODE\_OF\_CONDUCT.md

Gemfile

lib

LICENSE

MAINTAINERS.md

NOTICE.md

packer\_templates

Rakefile

README.md

test\_templates

**Step 4: Generate Rocky Linux 8 Vagrant Box for your Virtualization Provider**

Packer template for Rocky Linux is located in the *bento/packer\_templates/rockylinux/* directory. Let’s switch to this dir:

cd bento/packer\_templates/rockylinux/

As of this article writing the available template is for Rocky Linux 8.5.

$ ls

rockylinux-8.5-x86\_64.json

You can then build Rocky Linux 8 box for only Virtualization Provider:

*# Virtualbox provider*

packer build -only=virtualbox-iso rockylinux-8.5-x86\_64.json

*# VMware*

packer build -only=vmware-iso rockylinux-8.5-x86\_64.json

*# Parallels*

packer build -only=parallels-iso rockylinux-8.5-x86\_64.json

*# KVM*

packer build -only=qemu rockylinux-8.5-x86\_64.json

**VirtualBox Example**

Here is the output of Vagrant box creation with VirtualBox provider:

virtualbox-iso: output will be in this color.

==> virtualbox-iso: Retrieving Guest additions

==> virtualbox-iso: Trying /Applications/VirtualBox.app/Contents/MacOS/VBoxGuestAdditions.iso

==> virtualbox-iso: Trying /Applications/VirtualBox.app/Contents/MacOS/VBoxGuestAdditions.iso

==> virtualbox-iso: /Applications/VirtualBox.app/Contents/MacOS/VBoxGuestAdditions.iso => /Applications/VirtualBox.app/Contents/MacOS/VBoxGuestAdditions.iso

==> virtualbox-iso: Retrieving ISO

==> virtualbox-iso: Trying http://download.rockylinux.org/pub/rocky/8.5/isos/x86\_64/Rocky-8.5-x86\_64-dvd1.iso

==> virtualbox-iso: Trying http://download.rockylinux.org/pub/rocky/8.5/isos/x86\_64/Rocky-8.5-x86\_64-dvd1.iso?checksum=sha256%3A4be83f5edf28209ce5caa06995c1c3fc5112d0d260b9e8c1cc2fecd384abcee0

virtualbox-iso: Rocky-8.5-x86\_64-dvd1.iso 376.07 MiB / 9.23 GiB [==>--------------------------------------------------------------------] 3.98% 33m08s

Once the ISO file is downloaded an VM instance is created that is later converted to a box file.

==> virtualbox-iso: Starting HTTP server on port 8793

==> virtualbox-iso: Creating virtual machine...

==> virtualbox-iso: Creating hard drive ../../builds/packer-rockylinux-8.5-x86\_64-virtualbox/rockylinux-8.5-x86\_64.vdi with size 65536 MiB...

==> virtualbox-iso: Mounting ISOs...

virtualbox-iso: Mounting boot ISO...

==> virtualbox-iso: Creating forwarded port mapping for communicator (SSH, WinRM, etc) (host port 3183)

==> virtualbox-iso: Starting the virtual machine...

==> virtualbox-iso: Waiting 5s for boot...

==> virtualbox-iso: Typing the boot command...

==> virtualbox-iso: Using ssh communicator to connect: 127.0.0.1

==> virtualbox-iso: Waiting for SSH to become available...

Screenshot:

A VM instance created on VirtualBox but deleted after box generation.

Since the installer uses kickstart to automate the installation no user action is required:

Expected output from a successful build.

....

==> virtualbox-iso: ++ readlink -f /dev/disk/by-uuid/6a6aa6d9-8698-4e77-93b8-55809cf8e3a6

==> virtualbox-iso: + swappart=/dev/sda1

==> virtualbox-iso: + /sbin/swapoff /dev/sda1

==> virtualbox-iso: + dd if=/dev/zero of=/dev/sda1 bs=1M

==> virtualbox-iso: dd: error writing '/dev/sda1': No space left on device

==> virtualbox-iso: 2197+0 records in

==> virtualbox-iso: 2196+0 records out

virtualbox-iso: dd exit code 1 is suppressed

==> virtualbox-iso: 2302672896 bytes (2.3 GB, 2.1 GiB) copied, 1.51517 s, 1.5 GB/s

==> virtualbox-iso: + echo 'dd exit code 1 is suppressed'

==> virtualbox-iso: + /sbin/mkswap -U 6a6aa6d9-8698-4e77-93b8-55809cf8e3a6 /dev/sda1

virtualbox-iso: Setting up swapspace version 1, size = 2.1 GiB (2302668800 bytes)

virtualbox-iso: no label, UUID=6a6aa6d9-8698-4e77-93b8-55809cf8e3a6

==> virtualbox-iso: + sync

==> virtualbox-iso: Gracefully halting virtual machine...

==> virtualbox-iso: Preparing to export machine...

virtualbox-iso: Deleting forwarded port mapping for the communicator (SSH, WinRM, etc) (host port 3183)

==> virtualbox-iso: Exporting virtual machine...

virtualbox-iso: Executing: export rockylinux-8.5-x86\_64 --output ../../builds/packer-rockylinux-8.5-x86\_64-virtualbox/rockylinux-8.5-x86\_64.ovf

==> virtualbox-iso: Cleaning up floppy disk...

==> virtualbox-iso: Deregistering and deleting VM...

==> virtualbox-iso: Running post-processor: vagrant

==> virtualbox-iso (vagrant): Creating a dummy Vagrant box to ensure the host system can create one correctly

==> virtualbox-iso (vagrant): Creating Vagrant box for 'virtualbox' provider

virtualbox-iso (vagrant): Copying from artifact: ../../builds/packer-rockylinux-8.5-x86\_64-virtualbox/rockylinux-8.5-x86\_64-disk001.vmdk

virtualbox-iso (vagrant): Copying from artifact: ../../builds/packer-rockylinux-8.5-x86\_64-virtualbox/rockylinux-8.5-x86\_64.ovf

virtualbox-iso (vagrant): Renaming the OVF to box.ovf...

virtualbox-iso (vagrant): Compressing: Vagrantfile

virtualbox-iso (vagrant): Compressing: box.ovf

virtualbox-iso (vagrant): Compressing: metadata.json

virtualbox-iso (vagrant): Compressing: rockylinux-8.5-x86\_64-disk001.vmdk

Build 'virtualbox-iso' finished after 12 minutes 15 milliseconds.

==> Wait completed after 12 minutes 15 milliseconds

==> Builds finished. The artifacts of successful builds are:

--> virtualbox-iso: 'virtualbox' provider box: ../../builds/rockylinux-8.5.virtualbox.box

List builds directory:

$ ls ../../builds/

rockylinux-8.5.virtualbox.box uploaded

Import the Vagrant Box

$vagrant box add rockylinux-8.5 file://../../builds/rockylinux-8.5.virtualbox.box

==> box: Box file was not detected as metadata. Adding it directly...

==> box: Adding box 'rockylinux-8.5' (v0) for provider:

box: Unpacking necessary files from: file:///Users/jkmutai/Downloads/bento/builds/rockylinux-8.5.virtualbox.box

==> box: Successfully added box 'rockylinux-8.5' (v0) for 'virtualbox'!

Confirm the Box is available:

$ vagrant box list| grep rockylinux-8.5

rockylinux-8.5 (virtualbox, 0)

**VMware Box creation**

Successful box creation on VMware:

....

vmware-iso (vagrant): Compressing: disk-s001.vmdk

vmware-iso (vagrant): Compressing: disk-s002.vmdk

vmware-iso (vagrant): Compressing: disk-s003.vmdk

vmware-iso (vagrant): Compressing: disk-s004.vmdk

vmware-iso (vagrant): Compressing: disk-s005.vmdk

vmware-iso (vagrant): Compressing: disk-s006.vmdk

vmware-iso (vagrant): Compressing: disk-s007.vmdk

vmware-iso (vagrant): Compressing: disk-s008.vmdk

vmware-iso (vagrant): Compressing: disk-s009.vmdk

vmware-iso (vagrant): Compressing: disk-s010.vmdk

vmware-iso (vagrant): Compressing: disk-s011.vmdk

vmware-iso (vagrant): Compressing: disk-s012.vmdk

vmware-iso (vagrant): Compressing: disk-s013.vmdk

vmware-iso (vagrant): Compressing: disk-s014.vmdk

vmware-iso (vagrant): Compressing: disk-s015.vmdk

vmware-iso (vagrant): Compressing: disk-s016.vmdk

vmware-iso (vagrant): Compressing: disk-s017.vmdk

vmware-iso (vagrant): Compressing: disk.vmdk

vmware-iso (vagrant): Compressing: metadata.json

vmware-iso (vagrant): Compressing: rockylinux-8.5-x86\_64.nvram

vmware-iso (vagrant): Compressing: rockylinux-8.5-x86\_64.vmsd

vmware-iso (vagrant): Compressing: rockylinux-8.5-x86\_64.vmx

vmware-iso (vagrant): Compressing: rockylinux-8.5-x86\_64.vmxf

Build 'vmware-iso' finished after 12 minutes 31 seconds.

==> Wait completed after 12 minutes 31 seconds

==> Builds finished. The artifacts of successful builds are:

--> vmware-iso: 'vmware' provider box: ../../builds/rockylinux-8.5.vmware.box

You can import the box in similar way:

$ vagrant box add rockylinux-8.5 file://../../builds/rockylinux-8.5.vmware.box

==> box: Box file was not detected as metadata. Adding it directly...

==> box: Adding box 'rockylinux-8.5' (v0) for provider:

box: Unpacking necessary files from: file:///Users/jkmutai/Downloads/bento/builds/rockylinux-8.5.vmware.box

==> box: Successfully added box 'rockylinux-8.5' (v0) for 'vmware\_desktop'!

Check if box import was successful:

$ vagrant box list| grep rockylinux-8.5

rockylinux-8.5 (*virtualbox*, 0)

rockylinux-8.5 (*vmware\_desktop*, 0)

**Step 5: Create VM instance using Vagrant Box created**

We can then create a Vagrantfile for our box:

mkdir -p ~/vagrantboxes/rocky-linux

cd ~/vagrantboxes/rocky-linux

Create a *Vagrantfile* file:

vim Vagrantfile

For **VirtualBox** provider paste below contents:

# -\*- mode: ruby -\*-

# vi: set ft=ruby :

ENV['VAGRANT\_DEFAULT\_PROVIDER'] = 'virtualbox'

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

config.vm.box = "rockylinux-8.5"

config.vm.hostname = "*rocky-linux-8*"

config.vm.box\_check\_update = false

config.vm.provider "virtualbox" do |vm|

vm.name = "*rocky-linux-8*"

vm.memory = "*2048*"

vm.cpus = *2*

# Display the VirtualBox GUI when booting the machine

vm.gui = false

end

end

Create the vagrant environment using *vagrant up* command:

$ vagrant up

Bringing machine 'rockylinux-8' up with 'virtualbox' provider...

==> rockylinux-8: Importing base box 'rockylinux-8.5'...

==> rockylinux-8: Matching MAC address for NAT networking...

==> rockylinux-8: Setting the name of the VM: rocky-linux-8\_1625237224117\_71022

==> rockylinux-8: Clearing any previously set network interfaces...

==> rockylinux-8: Preparing network interfaces based on configuration...

rockylinux-8: Adapter 1: nat

==> rockylinux-8: Forwarding ports...

rockylinux-8: 22 (guest) => 2222 (host) (adapter 1)

==> rockylinux-8: Running 'pre-boot' VM customizations...

==> rockylinux-8: Booting VM...

==> rockylinux-8: Waiting for machine to boot. This may take a few minutes...

rockylinux-8: SSH address: 127.0.0.1:2222

rockylinux-8: SSH username: vagrant

rockylinux-8: SSH auth method: private key

rockylinux-8:

rockylinux-8: Vagrant insecure key detected. Vagrant will automatically replace

rockylinux-8: this with a newly generated keypair for better security.

rockylinux-8:

rockylinux-8: Inserting generated public key within guest...

rockylinux-8: Removing insecure key from the guest if it's present...

rockylinux-8: Key inserted! Disconnecting and reconnecting using new SSH key...

==> rockylinux-8: Machine booted and ready!

==> rockylinux-8: Checking for guest additions in VM...

==> rockylinux-8: Setting hostname...

==> rockylinux-8: Mounting shared folders...

rockylinux-8: /vagrant => /Users/jkmutai/myhacks/vagrant/rocky-linux

Access Vagrant environment VM Shell:

$ vagrant ssh

This system is built by the Bento project by Chef Software

More information can be found at https://github.com/chef/bento

[vagrant@rocky-linux-8 ~]$ cat /etc/redhat-release

Rocky Linux release 8.5 (Green Obsidian)

You can also generate SSH Config:

$ vagrant ssh-config rockylinux-8

Host rockylinux-8

HostName 127.0.0.1

User vagrant

Port 2222

UserKnownHostsFile /dev/null

StrictHostKeyChecking no

PasswordAuthentication no

IdentityFile /Users/jmutai/vagrantboxes/rocky-linux/.vagrant/machines/rockylinux-8/virtualbox/private\_key

IdentitiesOnly yes

LogLevel FATAL

These can be added to **~/.ssh/config** file

To suspends the the rocky-linux-8 vagrant environment run:

vagrant suspend

When done you can destroy vagrant environment anytime with the command:

$ **vagrant destroy**

rockylinux-8: Are you sure you want to destroy the 'rockylinux-8' VM? [y/N] y

==> rockylinux-8: Forcing shutdown of VM...

==> rockylinux-8: Destroying VM and associated drives...

**Video Courses to help you Learn Linux Administration:**

* [Linux Mastery: Master the Linux Command Line in 11.5 Hours](https://click.linksynergy.com/deeplink?id=2sb89XJC/EQ&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Flinux-mastery%2F)
* [Complete Linux Training Course to Get Your Dream IT Job 2021](https://click.linksynergy.com/deeplink?id=2sb89XJC/EQ&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Fcomplete-linux-training-course-to-get-your-dream-it-job%2F)
* [Learn Linux in 5 Days and Level Up Your Career](https://click.linksynergy.com/deeplink?id=2sb89XJC/EQ&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Flearn-linux-in-5-days%2F)
* [Linux Administration Bootcamp: Go from Beginner to Advanced](https://click.linksynergy.com/deeplink?id=2sb89XJC/EQ&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Flinux-administration-bootcamp%2F)
* [Complete Linux Bash Shell Scripting with Real Life Examples](https://click.linksynergy.com/deeplink?id=2sb89XJC/EQ&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Flinux-bash-shell-scripting-through-real-life-examples%2F)
* [Linux Shell Scripting: A Project-Based Approach to Learning](https://click.linksynergy.com/deeplink?id=2sb89XJC/EQ&mid=39197&murl=https%3A%2F%2Fwww.udemy.com%2Fcourse%2Flinux-shell-scripting-projects%2F)

**Conclusion**

Vagrant simplifies developer actions required to have a running instance to test application code. By using Chef Bento templates we were able to quickly create our own Box and started a Vagrant environment from the box generated. We hope this article was helpful. You can drop a comment as feedback or with any issue encountered.