Creating Linux virtual machine templates with Packer

date: Aug 14, 2021

tags: homelab linux packer proxmox virtual machine virtualbox vm template

categories: Homelab Proxmox Virtual Machines

On this page

Build Automated Machine Images

packer

You could set up each virtual machine by mounting the installation iso image to your virtual machine's virtual disk drive and proceeding through the installer. In fact this is what I recommend for new users, but I will be using a software known as Packer to create Linux templates that I can use in VirtualBox or even other hypervisors.

Templates can be found on my public github repo.

Getting Started

To create a virtual machine, you need a CPU that supports virtualization and hypervisor software to emulate computer hardware within software. I recommend starting with VirtualBox since this is an open-source hypervisor that you can install on Windows, macOS, or Linux.

You need to make sure that virtualization is enabled in your motherboard BIOS.

On my Windows PC, I have already installed VirtualBox and Packer with Chocolatey. If you would like to know how to install VirtualBox or other Windows software easily, check out my previous post on setting up a Windows system.

Key components

Packer makes it easier to perform unattended/automated installations of operating systems focused on virtual platforms but the community has also extended Packer to build Raspberry Pi images.

In order to proceed with Packer you need:

- Operating system Installation media
- Appropriate installation configuration template:
  - Debian has something called preseed
  - Red Hat based has something called kickstart
  - Windows supports something called answer files
- Packer template(s)
- (Optional) provisioning scripts (bash, PowerShell, ansible, etc.)

Installing Packer

Packer on Windows

I would also recommend installing this software with Chocolatey. If you are not familiar with Chocolatey, I suggest checking out my previous post. Once you have Chocolatey installed you can install Rufus with one command:

```
1 choco install packer
```

Packer on macOS

I would recommend installing Packer with homebrew on macOS. Once you have homebrew installed, you can install Packer with one command:

```
1 brew install packer
```

Packer on Linux

I would recommend installing Packer with your distribution's package manager if possible. Once you add the Hashicorp repo to your system you can also install other tools like Terraform. Otherwise, you can download the latest compiled binary from the website.

Building a Proxmox template with Packer

Proxmox is a Linux distribution where you can run virtual machines and containers and there is a nice web application for managing those resources. Check out my previous post for more information about Proxmox. You will need to use the default root credentials for Proxmox or create an API user with appropriate permissions.

Packer can have a bit of a steep learning curve if you are not familiar with JSON files. In fact, the makers of Packer now recommend you use their own template language HCL. If you are just getting started, I recommend using an existing set of Packer templates and provisioning scripts.

Proxmox API user

Enter the following commands to create an API user for Packer from the Proxmox host's shell:

```
pveum user add packerapi@pve -comment "Packer API user"
# Create a password for the new user
pveum passwd packerapi@pve
# Create a role with the appropriate permissions
pveum role add Packer -privs "Datastore.AllocateSpace Sys.Modify VM.Config.Disk VM.Config.CPU VM.Config.Memory VM.Config.Options VM.Allocate VM.Audit VM.Console VM.Con
# Assign the packer user the Packer role
pveum acl modify / -user packerapi@pve -role Packer
```

Example with a Debian template

Packer supports creating Proxmox templates from .iso images or existing virtual machines. This example will use a Debian .iso image to create a Proxmox VM template for creating additional virtual machines. Make a directory on your computer for the template files like this:

```
.
├── http
│   ├── deb10
│   │   └── preseed.cfg
├── pmox-deb10.json
└── scripts
    ├── deb10-seal.sh
```

The `http` subdirectory is for presenting the unattended installation files to Proxmox. The `scripts` subdirectory contains a script to randomize the machine UUID so the template can be reused.

Here is the unattended installation config that works well for Debian. The settings here are for the United States Eastern time zone.

http/deb10/preseed.cfg

```
choose-mirror-bin mirror/http/proxy string
d-i apt-setup/use_mirror boolean true
d-i base-installer/kernel/override-image string linux-server
d-i clock-setup/utc boolean true
d-i clock-setup/utc-auto boolean true
d-i finish-install/reboot_in_progress note
d-i grub-installer/only_debian boolean true
d-i grub-installer/with_other_os boolean true
d-i keymap select us
d-i mirror/country string manual
d-i mirror/http/directory string /debian
d-i mirror/http/hostname string ftp.us.debian.org
d-i mirror/http/proxy string
d-i partman-auto-lvm/guided_size string max
d-i partman-auto/choose_recipe select atomic
d-i partman-auto/method string lvm
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true
d-i partman-lvm/device_remove_lvm boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
d-i partman/confirm_write_new_label boolean true
d-i passwd/root-login boolean false
d-i passwd/root-password-again password debian
d-i passwd/root-password password debian
d-i passwd/user-fullname string debian
d-i passwd/user-uid string 1000
d-i passwd/user-password password debian
d-i passwd/user-password-again password debian
d-i passwd/username string debian
d-i pkgsel/include string sudo bzip2 acpid cryptsetup zlib1g-dev wget curl dkms make nfs-common net-tools vim git qemu-guest-agent
d-i pkgsel/install-language-support boolean false
d-i pkgsel/update-policy select none
d-i pkgsel/upgrade select full-upgrade
d-i time/zone string America/New_York
d-i user-setup/allow-password-weak boolean true
d-i user-setup/encrypt-home boolean false
d-i preseed/late_command string sed -i '/^deb cdrom:/s/^/#/' /target/etc/apt/sources.list
apt-cdrom-setup apt-setup/cdrom/set-first boolean false
apt-mirror-setup apt-setup/use_mirror boolean true
popularity-contest popularity-contest/participate boolean false
tasksel tasksel/first multiselect standard, ssh-server
```

A cleanup script to make the VM template have a new UUID:

scripts/deb10-seal.sh

```
#!/bin/bash -eux
apt-get autoremove -y
apt-get update
> /etc/machine-id
rm /var/lib/dbus/machine-id
ln -s /etc/machine-id /var/lib/dbus/machine-id
```

Here is an example of the Packer template in JSON format:

pmox-deb10.json

```
{
    "variables": {
        "username": "api@pve",
        "password": "password",
        "pmox_url": "https://proxmox.url:8006/api2/json",
        "guest_hostname": "packer-deb10",
        "ssh_user": "debian",
        "ssh_pass": "debian",
        "iso_location": "https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-10.9.0-amd64-xfce-CD-1.iso",
        "checksum": "sha256:6e507be9fd35c8a7c6be00aefa5b550ed3d8641432b2ae533295f4bb5246642b"
    },
    "builders": [
        {
            "type": "proxmox",
            "proxmox_url": "{{ user `pmox_url`}}",
            "insecure_skip_tls_verify": true,
            "username": "{{ user `username`}}",
            "password": "{{ user `password`}}",
            "node": "pve",
            "network_adapters": [
                {
                    "bridge": "vmbr0",
                    "model": "virtio"
                }
            ],
            "disks": [
                {
                    "type": "scsi",
                    "disk_size": "64G",
                    "storage_pool": "local-lvm",
                    "storage_pool_type": "lvm-thin",
                    "format": "raw"
                }
            ],
            "cores": 2,
            "sockets": 1,
            "memory": 2048,
            "os": "l26",
            "qemu_agent": true,
            "scsi_controller": "virtio-scsi-single",
            "iso_url": "{{ user `iso_location` }}",
            "iso_checksum": "{{ user `checksum` }}",
            "iso_storage_pool": "local",
            "http_directory": "http",
            "boot_wait": "10s",
            "boot_command": [
                "<esc><wait>",
                "install <wait>",
                " preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/deb10/preseed.cfg <wait>",
```

```
                "debian-installer=en_US.UTF-8 <wait>",
                "auto <wait>",
                "locale=en_US.UTF-8 <wait>",
                "kbd-chooser/method=us <wait>",
                "keyboard-configuration/xkb-keymap=us <wait>",
                "netcfg/get_hostname={{ user `guest_hostname` }} <wait>",
                "netcfg/get_domain=localdomain <wait>",
                "fb=false <wait>",
                "debconf/frontend=noninteractive <wait>",
                "console-setup/ask_detect=false <wait>",
                "console-keymaps-at/keymap=us <wait>",
                "grub-installer/bootdev=/dev/sda <wait>",
                "<enter><wait>"
            ],
            "ssh_username": "{{ user `ssh_user` }}",
            "ssh_timeout": "15m",
            "ssh_password": "{{ user `ssh_pass`}}",
            "unmount_iso": true,
            "template_name": "{{ user `guest_hostname` }}",
            "template_description": "Debian 10 Template created by packer"
        }
    ],
    "provisioners": [
        {
            "type": "shell",
            "execute_command": "echo '{{ user `ssh_pass`}}' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
            "script": "scripts/deb10-seal.sh"
        }
    ]
}
```

Once these files are assembled, the template can be created with the `packer build` command. I recommend always running `packer validate` before running the build.

Example end of output for a successful build:

```
==> proxmox: Converting VM to template
Build 'proxmox' finished after 8 minutes 8 seconds.
==> Wait completed after 8 minutes 8 seconds
==> Builds finished. The artifacts of successful builds are:
--> proxmox: A template was created
```

Building a VirtualBox template with Packer

Packer can also build a template for VirtualBox. Packer will boot a VM, install the OS, run any configuration scripts you add, shutdown the VM, and convert it to an OVF template. You can use that template to create new Virtual Machines that already have the Operating System installed.

Here is an example to create a Debian Linux template:

```
.
├── http
│   ├── deb10
│   │   └── preseed.cfg
├── vbox-deb10.json
└── scripts
    ├── deb10-seal.sh
```

http/deb10/preseed.cfg

```
choose-mirror-bin mirror/http/proxy string
d-i apt-setup/use_mirror boolean true
d-i base-installer/kernel/override-image string linux-server
d-i clock-setup/utc boolean true
d-i clock-setup/utc-auto boolean true
d-i finish-install/reboot_in_progress note
d-i grub-installer/only_debian boolean true
d-i grub-installer/with_other_os boolean true
d-i keymap select us
d-i mirror/country string manual
d-i mirror/http/directory string /debian
d-i mirror/http/hostname string ftp.us.debian.org
d-i mirror/http/proxy string
d-i partman-auto-lvm/guided_size string max
d-i partman-auto/choose_recipe select atomic
d-i partman-auto/method string lvm
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true
d-i partman-lvm/device_remove_lvm boolean true
d-i partman/choose_partition select finish
d-i partman/confirm boolean true
d-i partman/confirm_nooverwrite boolean true
d-i partman/confirm_write_new_label boolean true
d-i passwd/root-login boolean false
d-i passwd/root-password-again password debian
d-i passwd/root-password password debian
d-i passwd/user-fullname string debian
d-i passwd/user-uid string 1000
d-i passwd/user-password password debian
d-i passwd/user-password-again password debian
d-i passwd/username string debian
d-i pkgsel/include string sudo bzip2 acpid cryptsetup zlib1g-dev wget curl dkms make nfs-common net-tools vim git
d-i pkgsel/install-language-support boolean false
d-i pkgsel/update-policy select none
d-i pkgsel/upgrade select full-upgrade
d-i preseed/early_command string sed -i \
  '/in-target/idiscover(){/sbin/discover|grep -v VirtualBox;}' \
  /usr/lib/pre-pkgsel.d/20install-hwpackages
d-i time/zone string America/New_York
d-i user-setup/allow-password-weak boolean true
d-i user-setup/encrypt-home boolean false
d-i preseed/late_command string sed -i '/^deb cdrom:/s/^/#/' /target/etc/apt/sources.list
apt-cdrom-setup apt-setup/cdrom/set-first boolean false
apt-mirror-setup apt-setup/use_mirror boolean true
popularity-contest popularity-contest/participate boolean false
tasksel tasksel/first multiselect standard, ssh-server
```

---

vbox-deb10.json

```
{
    "variables": {
        "guest_hostname": "packer-deb10",
        "password": "debian",
        "iso_location": "https://cdimage.debian.org/debian-cd/current/amd64/iso-cd/debian-10.9.0-amd64-xfce-CD-1.iso",
        "checksum": "sha256:6e507be9fd35c8a7c6be00aefa5b550ed3d8641432b2ae533295f4bb5246642b"
    },
    "provisioners": [
      {
        "type": "shell",
        "execute_command": "echo '{{ user `password` }}' | {{.Vars}} sudo -S -E bash '{{.Path}}'",
        "script": "scripts/deb10-seal.sh"
      }
    ],
    "builders": [
      {
        "type": "virtualbox-iso",
        "boot_command": [
          "<esc><wait>",
          "install <wait>",
          " preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/deb10/preseed.cfg <wait>",
          "debian-installer=en_US.UTF-8 <wait>",
          "auto <wait>",
          "locale=en_US.UTF-8 <wait>",
          "kbd-chooser/method=us <wait>",
          "keyboard-configuration/xkb-keymap=us <wait>",
          "netcfg/get_hostname={{ .Name }} <wait>",
          "netcfg/get_domain=localdomain <wait>",
          "fb=false <wait>",
          "debconf/frontend=noninteractive <wait>",
          "console-setup/ask_detect=false <wait>",
```

```
          "console-keymaps-at/keymap=us <wait>",
          "grub-installer/bootdev=/dev/sda <wait>",
          "<enter><wait>"
        ],
        "boot_wait": "5s",
        "disk_size": 81920,
        "guest_os_type": "Debian_64",
        "headless": true,
        "http_directory": "http",
        "iso_urls": [
          "debian-10.6.0-amd64-xfce-CD-1.iso",
          "{{ user `iso_location` }}"
        ],
        "iso_checksum": "{{ user `checksum` }}",
        "ssh_username": "{{ user `password` }}",
        "ssh_password": "{{ user `password` }}",
        "ssh_port": 22,
        "ssh_wait_timeout": "1800s",
        "shutdown_command": "echo '{{ user `password` }}'|sudo -S shutdown -P now",
        "guest_additions_path": "VBoxGuestAdditions_{{.Version}}.iso",
        "virtualbox_version_file": ".vbox_version",
        "vm_name": "{{ user `guest_hostname` }}",
        "vboxmanage": [
          [
            "modifyvm",
            "{{.Name}}",
            "--memory",
            "1024"
          ],
          [
            "modifyvm",
            "{{.Name}}",
            "--cpus",
            "1"
          ]
        ]
      }
    ]
  }
```

scripts/deb10-seal.sh

```
#!/bin/bash -eux
apt-get autoremove -y
apt-get update
> /etc/machine-id
rm /var/lib/dbus/machine-id
ln -s /etc/machine-id /var/lib/dbus/machine-id
```

Once these files are assembled, the template can be created with the `packer build` command. I recommend always running `packer validate` before running the build.

Example end of output for a successful build:

```
Build 'virtualbox-iso' finished after 5 minutes 11 seconds.
==> Wait completed after 5 minutes 11 seconds
==> Builds finished. The artifacts of successful builds are:
--> virtualbox-iso: VM files in directory: output-virtualbox-iso
```

Packer will create an OVF file and a virtual machine hard disk in the specified directory. This template can be used to create new virtual machines.