Umair-khurshid
Posted on Aug 24, 2024

4

Using Packer and Proxmox to Build Templates

#proxmox #packer #tutorial #homelab

Automating the deployment of virtual machines (VMs) is an essential part of modern IT infrastructure. Packer and Proxmox are two powerful tools that, when combined, offer an effective solution for automated VM image creation. This article explores why using Packer with Proxmox is beneficial, as well as the practical implementation of this combination.

Why use Packer for Proxmox?

- **Reproducibility and consistency**: Packer lets you define all the configurations of a VM image in a file in HCL or JSON format. By using Packer, you guarantee reproducibility in the image creation process, ensuring consistency in your deployments.
- **Cross-platform support**: Packer supports multiple vendors, and Proxmox is just one of them. This flexibility allows you to create images that can be deployed on different types of platform, whether on-premise or in the cloud.
- **Optimisation of resources**: You can automate the creation of lightweight, optimised images, eliminating unnecessary components and applying configurations specific to your environment.
- **Time savings**: Automating the image creation process significantly reduces the time needed to prepare VMs. You can automate tasks such as software installation, network configuration and many others, speeding up the deployment cycle.

How to use Packer on Proxmox?

Install Packer on your development machine and configure it to use the Proxmox provider by specifying the connection details (IP address of the server where the image will be created, username, password). For installation, I can only suggest that you visit the official website.

There are then 3 steps in the Packer image creation process:

1. Creation of Packer model definition and automated OS configuration files
2. Create a Packer configuration file in HCL or JSON format describing the steps required to create your Proxmox image.
3. Specify parameters such as operating system type, provisioning scripts and Proxmox-specific parameters.

Operating systems can be configured autonomously and dynamically, without having to click or enter commands. To do this, the use of configuration files must be taken into account and maintained in line with changing requirements.

Build execution:
Start the building process with Packer. Packer will connect to your Proxmox host, create a temporary VM, apply configurations and will transform it to a template. This template will be the source for your deployment, the template to clone.

Validation:

- Check the generated image by deploying it on a Proxmox VM.
- Distribute the image to other Proxmox hosts or save it for future use.

My use case and my configuration

I use Packer in my homelab to generate templates (Debian 12 and Ubuntu 22.04) which I then re-use via Open Tofu to build the VMs I need. To make customization easier, I use variables as much as possible. So the base file is static and only the variables (in a separate file) need to be modified as required.

I have already uploaded the Debian 12 ISO image to storage on my Proxmox host. It is possible to use Packer to download the ISO image from the Internet and send it to your Proxmox host. I didn't choose this solution because I wanted to limit web access and set myself up for full "self-hosting".

To make the connection between Proxmox and Packer, I created a user in the "pve" realm, then I created a token for this user and finally assigned rights to the token API:

- In the "Datacenter" tab, go to the "Permissions" section and click on "Users" and the "Add" button. Enter the necessary information (no password required) then save the form. Remember the name of this user.
- Still in the "Permissions" section, click on "API Token" and the "Add" button. Select the user previously created, give the token a name (to remember) and continue with the form; a secret will then be displayed and must be stored in a safe place. This secret is required to operate Proxmox with Packer.
- Finally, go back to the "Permissions" section, then click on the "Add - API Token Permission" button: select the user with the API ID you just created, and add the following rights to it:
- *Path*: "/"
- *Role* : "PVEAdmin" (it's a lot, but there's no choice. Alternatively, you can create your own role with the necessary rights...)
- *Propagate*: tick the box (useful for having permissions over the entire infrastructure - machines, storage, network)

In the folder where my files are stored, I have created an "autoinstall" subfolder, to put the preseed file that allows Debian to be installed automatically.

```
~/git/homelab/packer$ tree
.
├── autoinstall
│   ├── preseed.cfg
├── customdeb12.pkrvars.hcl
└── debian12.pkr.hcl
```

From now on, here are the files needed to create a Debian 12 image with Packer, for Proxmox:
debian12.pkr.hcl" file: this is the file that defines the Packer requirements for our use case, the basis of our model, the connection to the Proxmox server and the variables.

```
// debian12.pkr.hcl
packer {
  required_plugins {
    name = {
      version = "1.1.6"
      source  = "github.com/hashicorp/proxmox"
    }
  }
}
variable "bios_type" {
  type = string
}
variable "boot_command" {
  type = string
}
variable "boot_wait" {
  type = string
}
variable "bridge_firewall" {
  type    = bool
  default = false
}
variable "bridge_name" {
  type = string
}
variable "cloud_init" {
  type = bool
}
variable "iso_file" {
  type = string
}
variable "iso_storage_pool" {
  type    = string
  default = "local"
}
variable "machine_default_type" {
  type    = string
  default = "pc"
}
variable "network_model" {
  type    = string
  default = "virtio"
}
variable "os_type" {
  type    = string
  default = "l26"
}
variable "proxmox_api_token_id" {
  type = string
}
variable "proxmox_api_token_secret" {
  type      = string
  sensitive = true
}
variable "proxmox_api_url" {
  type = string
}
variable "proxmox_node" {
  type = string
}
variable "qemu_agent_activation" {
  type    = bool
  default = true
}
variable "scsi_controller_type" {
  type = string
}
variable "ssh_timeout" {
  type = string
}
variable "tags" {
  type = string
}
variable "io_thread" {
  type = bool
}
variable "cpu_type" {
  type    = string
  default = "kvm64"
}
variable "vm_info" {
  type = string
```

```
}
variable "disk_discard" {
  type    = bool
  default = true
}
variable "disk_format" {
  type    = string
  default = "qcow2"
}
variable "disk_size" {
  type    = string
  default = "16G"
}
variable "disk_type" {
  type    = string
  default = "scsi"
}
variable "nb_core" {
  type    = number
  default = 1
}
variable "nb_cpu" {
  type    = number
  default = 1
}
variable "nb_ram" {
  type    = number
  default = 1024
}
variable "ssh_username" {
  type = string
}
variable "ssh_password" {
  type = string
}
variable "ssh_handshake_attempts" {
  type = number
}
variable "storage_pool" {
  type    = string
  default = "local-lvm"
}
variable "vm_id" {
  type    = number
  default = 99999
}
variable "vm_name" {
  type = string
}
locals {
  packer_timestamp = formatdate("YYYYMMDD-hhmm", timestamp())
}
source "proxmox-iso" "debian12" {
  bios                     = "${var.bios_type}"
  boot_command             = ["${var.boot_command}"]
  boot_wait                = "${var.boot_wait}"
  cloud_init               = "${var.cloud_init}"
  cloud_init_storage_pool  = "${var.storage_pool}"
  communicator             = "ssh"
  cores                    = "${var.nb_core}"
  cpu_type                 = "${var.cpu_type}"
  http_directory           = "autoinstall"
  insecure_skip_tls_verify = true
  iso_file                 = "${var.iso_file}"
  machine                  = "${var.machine_default_type}"
  memory                   = "${var.nb_ram}"
  node                     = "${var.proxmox_node}"
  os                       = "${var.os_type}"
  proxmox_url              = "${var.proxmox_api_url}"
  qemu_agent               = "${var.qemu_agent_activation}"
  scsi_controller          = "${var.scsi_controller_type}"
  sockets                  = "${var.nb_cpu}"
  ssh_handshake_attempts   = "${var.ssh_handshake_attempts}"
  ssh_pty                  = true
  ssh_timeout              = "${var.ssh_timeout}"
  ssh_username             = "${var.ssh_username}"
  ssh_password             = "${var.ssh_password}"
  tags                     = "${var.tags}"
  template_description     = "${var.vm_info} - ${local.packer_timestamp}"
  token                    = "${var.proxmox_api_token_secret}"
  unmount_iso              = true
  username                 = "${var.proxmox_api_token_id}"
  vm_id                    = "${var.vm_id}"
  vm_name                  = "${var.vm_name}"
  disks {
    discard      = "${var.disk_discard}"
    disk_size    = "${var.disk_size}"
    format       = "${var.disk_format}"
    io_thread    = "${var.io_thread}"
    storage_pool = "${var.storage_pool}"
    type         = "${var.disk_type}"
  }
  network_adapters {
    bridge   = "${var.bridge_name}"
    firewall = "${var.bridge_firewall}"
    model    = "${var.network_model}"
  }
}
build {
  sources = ["source.proxmox-iso.debian12"]
}
```

Custom.pkvars.hcl" file: Variables are used to customise the model.

```
// custom.pkvars.hcl
bios_type                 = "seabios"
boot_command              = "<esc><wait>auto console-keymaps-at/keymap=fr console-setup/ask_detect=false debconf/frontend=noninteractive fb=false url=http://{{ .HTTPIP
boot_wait                 = "10s"
bridge_name               = "vmbr0"
bridge_firewall           = false
cloud_init                = true
cpu_type                  = "x86-64-v2-AES"
disk_discard              = true
disk_format               = "qcow2"
disk_size                 = "12G"
disk_type                 = "scsi"
iso_file                  = "local:iso/debian-12.4.0-amd64-netinst.iso"
machine_default_type      = "q35"
nb_core                   = 1
nb_cpu                    = 1
nb_ram                    = 2048
network_model             = "virtio"
io_thread                 = false
os_type                   = "l26"
proxmox_api_token_id      = "packbot@pve!nom_token"
proxmox_api_token_secret = "aaaaaa-aaaa-bbbb-cccc-123456789012"
proxmox_api_url           = "https://ip_proxmox:8006/api2/json"
proxmox_node              = "nom_hôte_proxmox"
qemu_agent_activation     = true
scsi_controller_type      = "virtio-scsi-pci"
ssh_handshake_attempts    = 6
ssh_timeout               = "35m"
ssh_username              = "umair"
ssh_password              = "umairpwd"
storage_pool              = "stoCeph"
tags                      = "template"
vm_id                     = 99998
vm_info                   = "Debian 12 Packer Template"
vm_name                   = "pckr-deb12"
```

A few notes about these variables:

- Don't forget to change the values of "bridge_name", "iso_file" (the name of the storage where the ISO image is and its name), "proxmox_", "ssh_*"...
- The "boot_command" variable is used to force the ISO to use the preseed file
- "cloud_init is installed and activated to make it easier for Open Tofu to configure VMs

I'll leave you to take the other variables into account and modify them to suit your needs.

"autoinstall/preseed.cfg" file: Debian installation configuration file, for zero-touch installation.

```
#_preseed_V1
d-i debian-installer/language string en
d-i debian-installer/country string EN
d-i debian-installer/locale string en_US.UTF-8
d-i localechooser/supported-locales multiselect en_US.UTF-8, en_US.UTF-8
d-i keyboard-configuration/xkb-keymap select en
d-i console-keymaps-at/keymap select en-latin9
d-i debian-installer/keymap string en-latin9
d-i netcfg/choose_interface select auto
d-i netcfg/link_wait_timeout string 5
d-i netcfg/dhcp_timeout string 60
d-i netcfg/get_hostname string pckr-deb12
d-i netcfg/get_domain string local.xyz.com
d-i netcfg/wireless_wep string
d-i hw-detect/load_firmware boolean false
d-i mirror/country string FR
d-i mirror/http/hostname string deb.debian.org
d-i mirror/http/directory string /debian
d-i mirror/http/proxy string
d-i passwd/root-login boolean true
d-i passwd/make-user boolean true
d-i passwd/root-password password umairpwd
d-i passwd/root-password-again password umairpwd
d-i passwd/user-fullname string umair
d-i passwd/username string umair
d-i passwd/user-password password umairpwd
d-i passwd/user-password-again password umairpwd
d-i clock-setup/utc boolean true
d-i time/zone string US/New York
d-i clock-setup/ntp boolean true
d-i clock-setup/ntp-server string  0.us.pool.ntp.org
d-i partman-auto/disk string /dev/sda
d-i partman-auto/method string lvm
d-i partman-auto-lvm/guided_size string max
d-i partman-lvm/device_remove_lvm boolean true
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true
d-i partman-auto/choose_recipe select multi
d-i partman-partitioning/confirm_write_new_label 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-md/confirm boolean true
d-i partman-partitioning/confirm_write_new_label 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/mount_style select uuid
d-i base-installer/install-recommends boolean false
d-i apt-setup/cdrom/set-first boolean false
d-i apt-setup/use_mirror boolean true
d-i apt-setup/security_host string security.debian.org
tasksel tasksel/first multiselect standard, ssh-server
d-i pkgsel/include string qemu-guest-agent sudo ca-certificates cloud-init
d-i pkgsel/upgrade select safe-upgrade
popularity-contest popularity-contest/participate boolean false
d-i grub-installer/only_debian boolean true
d-i grub-installer/with_other_os boolean false
d-i grub-installer/bootdev string default
d-i finish-install/reboot_in_progress note
d-i cdrom-detect/eject boolean true
```

With all the above files and the user configuration in Proxmox, you should be able to create a machine. The Packer commands to run are as follows:

```
# initialize the repository to retrieve:
packer init debian12.pkr.hcl
# validate Packer files:
packer validate -var-file=customdeb12.pkrvars.hcl debian12.pkr.hcl
# build the image, replacing the artefact if it already exists:
packer build -on-error=ask -force -var-file=customdeb12.pkrvars.hcl debian12.pkr.hcl
```

Top comments (0)