Creating a VM Debian 12 Image for arm64 architecture targeting QEMU & Vagrant usage with QEMU and Packer on MacOS

Posted: Sep 6, 2024

Intro

In this post, we'll walk through the process of creating an unattended installation of Debian 12 for arm64 architecture using QEMU and Packer on an M1 Mac.

Packer is a community tool from Hashicorp that standardizes and automates the process of building system and container images for multiple platforms.

The goal here is to generate a QEMU disk image, which can be used in a lab environment, leveraging the MacOS HVF (Hypervisor Framework) for optimal performance. Additionally, we'll configure the image with passwordless sudo for ease in post-installation and day to day tasks, which is practical for lab use.

Let's dive into the Packer setup and automation with HVF acceleration enabled.

Packerfile

Here is the Packerfile `debian.pkr.hcl`:

```
# Packer plugins section
packer {
  required_plugins {
    qemu = {
      source  = "github.com/hashicorp/qemu"
      version = "~> 1"
    }
    vagrant = {
      source  = "github.com/hashicorp/vagrant"
      version = "~> 1"
    }
  }
}

# Variables Section
variable "boot_wait" {
  type    = string
  default = "5s"
}
variable "disk_size" {
  type    = string
  default = "5G"
}
variable "iso" {
  type    = string
  default = "debian-12.5.0-arm64-DVD-1.iso"
}
variable "arch" {
  type    = string
  default = "a64" # a64 for aarch64 , amd for x86_64
}
variable "checksum" {
  type    = string
  default = "de7c838d24d1664d06671df545471f6ed84d945874060c7318d4b2424a480d10"
}

locals {
  name    = "debian"
  version = "12.5"
  url     = "./${var.iso}"
  #url     = "https://cdimage.debian.org/debian-cd/current/arm64/iso-dvd/${var.iso}"
  vm_name = "${local.name}-${local.version}"
  arch    = "arm64"

  vars = {
    vm_name    = local.name
    domain = "local"
    user = {
      name     = "vagrant"
      password = "vagrant"
      }
    root = {
      password = "vagrant"
      }
  }
}

source "qemu" "debian" {
  vm_name               = local.vm_name
  qemu_binary           = "qemu-system-aarch64"
  machine_type          = "virt"
  accelerator           = "hvf"
  cpus                  = 2
  memory                = 2048
  boot_wait             = "${var.boot_wait}"
  boot_key_interval     = "10ms"
  net_device            = "e1000"
  disk_interface        = "virtio"
  format                = "qcow2"
  disk_size             = "${var.disk_size}"
  headless              = true
 # The preseed file is generated from a template and served by packer
  http_content            = {
        "/preseed.cfg" = templatefile("${path.root}/preseed.pkrtpl", local.vars )
      }


# The boot command enter the menu to modify the boot args to allow for usage of the preseeding file.
  boot_command = [
    "<wait5>e",
    "<leftCtrlOn>k<leftCtrlOff><leftCtrlOn>k<leftCtrlOff><leftCtrlOn>k<leftCtrlOff><leftCtrlOn>k<leftCtrlOff><leftCtrlOn>k<leftCtrlOff><leftCtrlOn>k<leftCtrlOff><leftCtrlOn>k<leftCtrlOff><leftCtrlOn>k<leftCtrlOff>",
    "linux /install.${var.arch}/vmlinuz<spacebar>",
    "auto=true<spacebar>",
    "preseed/url=http://{{ .HTTPIP }}:{{ .HTTPPort }}/preseed.cfg<spacebar>",
    "vga=788 noprompt<spacebar>",
    "hostname=${local.vars.vm_name}<spacebar>",
    "domain=${local.vars.domain}<spacebar>",
    "console=ttyS0,115200<spacebar>",
    "BOOT_DEBUG=2<spacebar>",
    "DEBCONF_DEBUG=5<enter>",
    "initrd /install.${var.arch}/initrd.gz<enter>",
    "<leftCtrlOn>x<leftCtrlOff>",
  ]
  iso_url               = local.url
  iso_checksum          = var.checksum
  qemuargs=[
      ["-cpu", "host"],
      ["-bios", "/opt/homebrew/share/qemu/edk2-aarch64-code.fd"],
      ["-boot", "strict=off"],
      ["-device", "qemu-xhci"],
      ["-audio", "none"],
      ["-serial", "file:./serial-output.txt"],
      ["-device", "usb-kbd"],
      ["-device", "virtio-gpu-pci"],
  ]
  shutdown_command = "echo 'packer'|sudo systemctl poweroff "
  ssh_password     = local.vars.user.password
  ssh_port         = 22
  ssh_timeout      = "5m"
  ssh_username     = local.vars.user.name
}

build {
  name = local.vm_name
  sources = ["source.qemu.debian"]

  # Pre-build task to remove the output directory if it exists
  provisioner "shell-local" {
    inline = ["rm -rf output-debian"]
    only = ["before_build"]
  }

  provisioner "file" {
    source      = "scripts/vagrant.pub"
    destination = "/tmp/vagrant.pub"
  }

  provisioner "shell" {
    execute_command = "echo 'packer'|{{ .Vars }} sudo -S -E bash '{{ .Path }}'"
    scripts         = ["scripts/cleanup.sh", "scripts/vagrant.sh"]
  }

      # Once the provisionning is done we package it for Vagrant
  post-processor "vagrant" {
    keep_input_artifact = true
    output              = "../box/${local.vars.vm_name}-vagrant.box"
  }
}
```

Debian Preseed file

A Debian preseed file is a configuration file used to automate the installation process of Debian-based operating systems. It contains predefined answers to installation questions, allowing for unattended or minimally attended installations of Debian or Ubuntu systems.

This preseed file is templated with Packer (`preseed.pkrtpl`). Feel free to read it and modify it according to your need (language, user & root password, etc).

Note the late_command part that activated the sudo without password and as well correct an EFI issue for the boot later with qemu.

```
# Source: https://www.debian.org/releases/bookworm/example-preseed.txt
 # Source: https://www.debian.org/releases/bookworm/example-preseed.txt

d-i debconf/priority select critical

### Localization
# Preseeding only locale sets language, country and locale.
d-i debian-installer/locale string en_US

# Keyboard selection.
d-i keyboard-configuration/xkb-keymap select fr-latin9
d-i keyboard-configuration/variant select France
d-i console-setup/charmap select UTF-8
```

```
d-i console-setup/layout select France
d-i console-keymaps-at/keymap select fr-latin9
d-i debian-installer/keymap string fr-latin9

### Network configuration

# netcfg will choose an interface that has link if possible. This makes it
# skip displaying a list if there is more than one interface.
d-i netcfg/choose_interface select auto

# Any hostname and domain names assigned from dhcp take precedence over
# values set here. However, setting the values still prevents the questions
# from being shown, even if values come from dhcp.
d-i netcfg/get_domain string ${domain}

# If you want to force a hostname, regardless of what either the DHCP
# server returns or what the reverse DNS entry for the IP is, uncomment
# and adjust the following line.
d-i netcfg/hostname string ${vm_name}
d-i netcfg/get_hostname string ${vm_name}

# Disable that annoying WEP key dialog.
d-i netcfg/wireless_wep string

### Mirror settings
# If you select ftp, the mirror/country string does not need to be set.
#d-i mirror/protocol string ftp
d-i mirror/http/hostname string http://deb.debian.org/debian/
d-i mirror/http/hostname string http://security.debian.org/debian-security

### Account setup

# Root password, either in clear text
d-i passwd/root-password password ${root.password}
d-i passwd/root-password-again password ${root.password}

# Normal user's password, either in clear text
d-i passwd/user-fullname string ${user.name}
d-i passwd/username string ${user.name}
d-i passwd/user-password password ${user.password}
d-i passwd/user-password-again password ${user.password}
d-i user-setup/allow-password-weak boolean true
#d-i user-setup/encrypt-home boolean false
d-i passwd/user-default-groups string si audio cdrom video admin sudo vagrant
d-i passwd/user-uid string 1010

### Clock and time zone setup
# Controls whether or not the hardware clock is set to UTC.
d-i clock-setup/utc boolean true

# You may set this to any valid setting for $TZ; see the contents of
# /usr/share/zoneinfo/ for valid values.
d-i time/zone string Europe/Paris
d-i time/zone select Europe/Paris

# Controls whether to use NTP to set the clock during the install
d-i clock-setup/ntp boolean true

### Partitioning

# In addition, you'll need to specify the method to use.
# The presently available methods are:
# - regular: use the usual partition types for your architecture
# - lvm:     use LVM to partition the disk
# - crypto:  use LVM within an encrypted partition
d-i partman-auto/method string lvm

# You can define the amount of space that will be used for the LVM volume
# group. It can either be a size with its unit (eg. 20 GB), a percentage of
# free space or the 'max' keyword.
d-i partman-auto-lvm/guided_size string max

# If one of the disks that are going to be automatically partitioned
# contains an old LVM configuration, the user will normally receive a
# warning. This can be preseeded away...
d-i partman-lvm/device_remove_lvm boolean true
# The same applies to pre-existing software RAID array:
d-i partman-md/device_remove_md boolean true
# And the same goes for the confirmation to write the lvm partitions.
d-i partman-lvm/confirm boolean true
d-i partman-lvm/confirm_nooverwrite boolean true

# You can choose one of the three predefined partitioning recipes:
# - atomic: all files in one partition
# - home:   separate /home partition
# - multi:  separate /home, /var, and /tmp partitions
d-i partman-auto/choose_recipe select atomic

# This makes partman automatically partition without confirmation, provided
# that you told it what to do using one of the methods above.
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

# This makes partman automatically partition without confirmation.
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

# Force UEFI booting ('BIOS compatibility' will be lost). Default: false.
d-i partman-efi/non_efi_system boolean true

# Ensure the partition table is GPT - this is required for EFI
d-i partman-partitioning/choose_label select gpt
d-i partman-partitioning/default_label string gpt


### Base system installation

### Package selection
tasksel tasksel/first multiselect standard, ssh-server

# Whether to upgrade packages after debootstrap.
# Allowed values: none, safe-upgrade, full-upgrade
d-i pkgsel/upgrade select safe-upgrade
d-i pkgsel/include string sudo vim htop

# Some versions of the installer can report back on what software you have
# installed, and what software you use. The default is not to report back,
# but sending reports helps the project determine what software is most
# popular and include it on CDs.
popularity-contest popularity-contest/participate boolean false

### Boot loader installation

# Due notably to potential USB sticks, the location of the MBR can not be
# determined safely in general, so this needs to be specified:
#d-i grub-installer/bootdev  string /dev/vda

# This one makes grub-installer install to the MBR if it also finds some other
# OS, which is less safe as it might not be able to boot that other OS.
d-i grub-installer/with_other_os boolean true

# To install to the first device (assuming it is not a USB stick):
d-i grub-installer/bootdev string default

### Finishing up the installation
# During installations from serial console, the regular virtual consoles
# (VT1-VT6) are normally disabled in /etc/inittab. Uncomment the next
# line to prevent this.
d-i finish-install/keep-consoles boolean true

# Avoid that last message about the install being complete.
d-i finish-install/reboot_in_progress note

# This will prevent the installer from ejecting the CD during the reboot,
# which is useful in some situations.
#d-i cdrom-detect/eject boolean false
# Prevent asking for more installation media
d-i apt-setup/cdrom/set-first boolean false

# This command is run just before the install finishes, but when there is
# still a usable /target directory. You can chroot to /target and use it
# directly, or use the apt-install and in-target commands to easily install
# packages and run commands in the target system.
d-i preseed/late_command string apt install netplan.io ; echo '${user.name} ALL=(ALL) NOPASSWD: ALL' > /target/etc/sudoers.d/${user.name} ; in-target chmod 440 /etc/sudoers.d/${user.name} ; mkdir target/boot/efi/EFI/boot ; cp target/boot/efi/EFI/debian/grubaa64.efi target/boot/efi/EFI/boot/bootaa64.e
```

Launching the build

Once both files are present, you can launch the build with :

```
packer build debian.pkr.hcl
```

Here is the output expected:

```
debian-12.5.qemu.debian: output will be in this color.

==> debian-12.5.qemu.debian: Retrieving ISO
==> debian-12.5.qemu.debian: Trying ./debian-12.5.0-arm64-DVD-1.iso
==> debian-12.5.qemu.debian: Trying ./debian-12.5.0-arm64-DVD-1.iso?checksum=sha256%3Ade7c838d24d1664d06671df545471f6ed84d945874060c7318d4b2424a480d10
==> debian-12.5.qemu.debian: ./debian-12.5.0-arm64-DVD-1.iso?checksum=sha256%3Ade7c838d24d1664d06671df545471f6ed84d945874060c7318d4b2424a480d10 => /Users/to108637/DevProjects/udp-sessionlist-lab/packer/debian-12.5.0-arm64-DVD-1.iso
==> debian-12.5.qemu.debian: Starting HTTP server on port 8982
==> debian-12.5.qemu.debian: Found port for communicator (SSH, WinRM, etc): 3381.
==> debian-12.5.qemu.debian: Creating temporary RSA SSH key for instance...
==> debian-12.5.qemu.debian: Looking for available port between 5900 and 6000 on 127.0.0.1
==> debian-12.5.qemu.debian: Starting VM, booting from CD-ROM
    debian-12.5.qemu.debian: The VM will be run headless, without a GUI. If you want to
    debian-12.5.qemu.debian: view the screen of the VM, connect via VNC without a password to
    debian-12.5.qemu.debian: vnc://127.0.0.1:5910
==> debian-12.5.qemu.debian: Overriding default Qemu arguments with qemuargs template option...
==> debian-12.5.qemu.debian: Waiting 5s for boot...
==> debian-12.5.qemu.debian: Connecting to VM via VNC (127.0.0.1:5910)
==> debian-12.5.qemu.debian: Typing the boot commands over VNC...
    debian-12.5.qemu.debian: Not using a NetBridge -- skipping StepWaitGuestAddress
==> debian-12.5.qemu.debian: Using SSH communicator to connect: 127.0.0.1
==> debian-12.5.qemu.debian: Waiting for SSH to become available...
==> debian-12.5.qemu.debian: Connected to SSH!
==> debian-12.5.qemu.debian: Uploading scripts/vagrant.pub => /tmp/vagrant.pub
    debian-12.5.qemu.debian: vagrant.pub 519 B / 519 B [========] 100.00% 0s
==> debian-12.5.qemu.debian: Provisioning with shell script: scripts/cleanup.sh
    debian-12.5.qemu.debian: Reading package lists...
    debian-12.5.qemu.debian: Building dependency tree...
    debian-12.5.qemu.debian: Reading state information...
    debian-12.5.qemu.debian: 0 upgraded, 0 newly installed, 0 to remove and 0 not upgraded.
==> debian-12.5.qemu.debian: Provisioning with shell script: scripts/vagrant.sh
    debian-12.5.qemu.debian: Added PubKeyAuthentication yes to /etc/ssh/sshd_config
    debian-12.5.qemu.debian: Added AuthorizedKeysFile %h/.ssh/authorized_keys to /etc/ssh/sshd_config
    debian-12.5.qemu.debian: Added PermitEmptyPasswords no to /etc/ssh/sshd_config
    debian-12.5.qemu.debian: Added PasswordAuthentication no to /etc/ssh/sshd_config
    debian-12.5.qemu.debian: Reading package lists...
    debian-12.5.qemu.debian: Building dependency tree...
    debian-12.5.qemu.debian: Reading state information...
    debian-12.5.qemu.debian: The following additional packages will be installed:
    debian-12.5.qemu.debian:   binutils binutils-aarch64-linux-gnu binutils-common cpp cpp-12 dirmngr
```

```
[...]
    debian-12.5.qemu.debian:   rpcsvc-proto
    debian-12.5.qemu.debian: 0 upgraded, 86 newly installed, 0 to remove and 0 not upgraded.
    debian-12.5.qemu.debian: Need to get 0 B/88.5 MB of archives.
    debian-12.5.qemu.debian: After this operation, 358 MB of additional disk space will be used.
    debian-12.5.qemu.debian: Get:1 cdrom://[Debian GNU/Linux 12.5.0 _Bookworm_ - Official arm64 DVD Binary-1 with firmware 20240210-11:28] bookworm/main arm64 binutils-common arm64 2.40-2 [2,487 kB]
[...]
    debian-12.5.qemu.debian: Setting up linux-headers-6.1.0-18-arm64 (6.1.76-1) ...
    debian-12.5.qemu.debian: Setting up g++ (4:12.2.0-3) ...
    debian-12.5.qemu.debian: update-alternatives: using /usr/bin/g++ to provide /usr/bin/c++ (c++) in auto mode
    debian-12.5.qemu.debian: Setting up build-essential (12.9) ...
    debian-12.5.qemu.debian: Processing triggers for man-db (2.11.2-2) ...
    debian-12.5.qemu.debian: Processing triggers for libc-bin (2.36-9+deb12u4) ...
==> debian-12.5.qemu.debian: Gracefully halting virtual machine...
==> debian-12.5.qemu.debian: Converting hard drive...
==> debian-12.5.qemu.debian: Running post-processor:  (type vagrant)
==> debian-12.5.qemu.debian (vagrant): Creating a dummy Vagrant box to ensure the host system can create one correctly
==> debian-12.5.qemu.debian (vagrant): Creating Vagrant box for 'libvirt' provider
    debian-12.5.qemu.debian (vagrant): Copying from artifact: output-debian/debian-12.5
    debian-12.5.qemu.debian (vagrant): Compressing: Vagrantfile
    debian-12.5.qemu.debian (vagrant): Compressing: box_0.img
    debian-12.5.qemu.debian (vagrant): Compressing: metadata.json
Build 'debian-12.5.qemu.debian' finished after 2 minutes 48 seconds.

==> Wait completed after 2 minutes 48 seconds

==> Builds finished. The artifacts of successful builds are:
--> debian-12.5.qemu.debian: VM files in directory: output-debian
--> debian-12.5.qemu.debian: 'libvirt' provider box: ../box/debian-vagrant.box
```

References

https://gist.github.com/max-i-mil/f44e8e6f2416d88055fc2d0f36c6173b#2---manually-start-virtlogd

https://cloud.debian.org/images/cloud/bookworm/20240415-1718/

https://actuated.dev/blog/automate-packer-qemu-image-builds

https://sigmaris.info/blog/2019/04/automating-debian-install-qemu/

https://shantanoo-desai.github.io/posts/technology/packer-ubuntu-qemu/

- QEMU
- M1 Mac
- arm64
- HVF
- EFI Boot
- Packer
- Unattended Installation
- Automation
- Disk Image
- Debian 12
- Cloud Infrastructure
- Passwordless Sudo
- Virtualization
- Networking