Debian VM Template

Purpose

These instructions are for standing up a Debian 13.4 VM to be used as a template in an isolated manufacturing network.

It will be prepopulated with packages we anticipate needing:

- NGINX
- SubVersion
- net-tools
- Docker
- nano
- python
- sqlite

> NOTE: This process creates a template VM that is scrubbed of any machine-ID, host SSH keys, etc.
> These values are created each time the template is started. And, must be reset, for the VM to be a viable template, again.
>
> So. If you do make changes to the template VM after creation, you will have to rerun the cleanup and template reset scripts.
>
> To do so, follow the steps near bottom of this page, starting in the Cleanup section.

Setup

Update Host

```
apt update
```

Install sudo:

```
apt install sudo
```

Add a user:

```
usermod -aG sudo yourusername
```

Add user to sudoers:

As root, open /etc/sudoers.

Add this entry to the bottom:

```
yourusername ALL=(ALL:ALL) ALL
```

Package Install

Update the base image:

```
sudo apt update
sudo apt upgrade -y
```

Install core admin and networking tools:

```
sudo apt install -y \
  openssh-server sudo ca-certificates curl wget gnupg lsb-release \
  nano vim less bash-completion locales tzdata \
  iproute2 net-tools dnsutils iputils-ping traceroute tcpdump nmap netcat-openbsd \
  htop iotop lsof psmisc strace procps sysstat \
  jq tree file unzip zip tar rsync dos2unix \
  apache2-utils openssl \
  chrony \
  git make build-essential python3 python3-pip python3-venv \
  nfs-common cifs-utils \
  parted gdisk smartmontools acl \
  tmux screen ncdu \
  sqlite3
```

Install network tools:

```
sudo apt install -y mtr-tiny socat
```

Install certificate debugging tools:

```
sudo apt install -y ssl-cert
```

Install http API debugging tools:

```
sudo apt install -y httpie
```

Install process debugging tools:

```
sudo apt install -y dstat
```

Install NGINX

```
sudo apt install -y nginx
sudo systemctl enable nginx
sudo systemctl start nginx
```

Install Docker

First remove conflicting packages:

```
sudo apt remove -y docker.io docker-doc docker-compose podman-docker containerd runc || true
```

Add Docker repo and key:

```
sudo install -m 0755 -d /etc/apt/keyrings

curl -fsSL https://download.docker.com/linux/debian/gpg | \
  sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

sudo chmod a+r /etc/apt/keyrings/docker.gpg

echo \
  "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
  https://download.docker.com/linux/debian \
  $(. /etc/os-release && echo $VERSION_CODENAME) stable" | \
  sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
```

Install Docker:

```
sudo apt update
sudo apt install -y \
  docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin
```

Enable Docker:

```
sudo systemctl enable docker
sudo systemctl start docker
```

Allow user to run Docker without sudo:

```
sudo usermod -aG docker $USER
```

Add offline package management for Docker:

```
sudo apt install -y dpkg-dev
```

SubVersion Client

Installing the SVN client is straightforward.

NOTE: It was actually installed as offline packages, after the VM was moved to the target network. The instructions, here, are to describe how it would have been installed, before the move.

See this page for details of how it was installed as an offline package: Debian: Offline Package Installation

To install the subversion client, use this:

```
sudo apt update
sudo apt install subversion
```

Once installed, you can confirm it with this:

```
svn --version
```

If successful it will display the help file with client version data.

.NET Runtime

This requires a couple steps.

Do this, first:

```
wget https://packages.microsoft.com/config/debian/13/packages-microsoft-prod.deb -O packages-microsoft-prod.deb


sudo dpkg -i packages-microsoft-prod.deb
```

```
rm packages-microsoft-prod.deb

sudo apt update
```

Install the runtime:

```
sudo apt install -y dotnet-runtime-8.0
```

Install the aspnet runtime:

```
sudo apt install -y aspnetcore-runtime-8.0
```

Install the .NET sdk:

```
sudo apt install -y dotnet-sdk-8.0
```

Verify the installed version:

```
dotnet --info
```

Have it list the runtimes, as well:

```
dotnet --list-runtimes
```

Do a quick hello world, to test the runtime.

```
mkdir ./dotnet-test && cd ./dotnet-test

dotnet new console
dotnet run
```

Validation Checks

```
systemctl status nginx
systemctl status docker
docker run hello-world
```

VM Templating Setup

Follow the instructions on this page, to modify the VM for templating:

Doing so, allows the VM to deploy, without any chance of colliding machine-Id, host SSH keys, and such.

Cleanup

Once the above is done, and you've followed the VM templating setup steps, you can do any final cleanup, with this:

> NOTE: This step is included as a means to reduce the VM size for copying it across networks.
> You may not have to accomplish this each time you update the template VM.
> You can probably skip this step, if just updating the template VM, where it sits.

```
sudo apt autoremove --purge -y
sudo apt clean
sudo rm -rf /var/lib/apt/lists/*
sudo journalctl --vacuum-time=1d
sudo rm -rf /tmp/* /var/tmp/*
```

Template Reset Script

Once the VM image size is reduced, you are ready to scrub it as a template.

Previous steps installed a template reset script, here:

```
/etc/template/reset.sh
```

Run it each time you shutdown the template VM, to reset needed info, for cloning.

> NOTE: This is what you will execute, each time you had to start up the template VM, to make modifications to it.
> Follow this command with a PowerOff command. See PowerOff section.

It can be executed with this:

```
sudo /etc/template/reset.sh
```

The script contents are here: Template VM Reset Script

Zero Out Free Space

To make the VM smaller, this needs to be done.

>

> NOTE: This step is included as a means to reduce the VM size for copying it across networks.
> You may not have to accomplish this each time you update the template VM.
> You can probably skip this step, if just updating the template VM, where it sits.

```
sudo dd if=/dev/zero of=/EMPTY bs=1M status=progress || true
sudo rm /EMPTY
sync
```

Power Off

Instead of a shutdown, you should execute a power off with this:

```
sudo poweroff
```

VM Export

The easiest way to move the VM to another datacenter, is by turning it into an OVA.

Follow the instructions on this page, to convert it to an OVA: VMWare Export OVA

Further Packages

You will inevitably need to install packages, after the VM has been deployed inside the isolated network.

See this page for details of how we installed the SubVersion client as an offline package: Debian: Offline Package Installation

> NOTE: Each time you make changes to the template VM, that requires starting it up, which creates fresh machine-id host SSH keys, etc.
> So, you have to clear those out, before the VM is a viable template, again.
> Start in the Cleanup section, above, and follow steps, to make the VM a viable template, again.

Revision #29
Created 24 April 2026 22:05:47 by glwhite
Updated 28 April 2026 13:12:20 by glwhite