# Proxmox Debian 12 Docker Template Manual

## Overview

This guide covers how to build a Debian 12 “gold image” VM on Proxmox, prepare it with Docker and QEMU Guest Agent, convert it into a reusable template, and then clone it. It includes both \*\*ready-to-use scripts\*\* and \*\*manual command sequences\*\* for when you need finer control.

---

## 🔹 Section 1. Host: Build Base VM from qcow2

### Script: `build\_deb12\_docker\_base.sh`

# Creates a Debian 12 base VM with VGA + serial consoles and cloud-init enabled.

#!/usr/bin/env bash  
  
set -euo pipefail  
  
VMID=9000  
NAME=deb12-docker-base  
BRIDGE=vmbr0  
MEM=2048  
CORES=2  
DISK\_SIZE=20G  
ISO\_QCOW\_PATH=/var/lib/vz/template/iso/debian-12-genericcloud-amd64.qcow2  
STORAGE=local-lvm  
CIUSER=dockeruser  
CIPASS=TempPassw0rd!  
  
qm create $VMID --name $NAME --ostype l26 --memory $MEM --cores $CORES --net0 virtio,bridge=$BRIDGE  
  
qm importdisk $VMID $ISO\_QCOW\_PATH $STORAGE  
  
DISK\_REF=$(qm config $VMID | awk '/^unused[0-9]+:/ {print $2; exit}')  
qm set $VMID --scsihw virtio-scsi-pci --scsi0 $DISK\_REF  
qm set $VMID --ide2 ${STORAGE}:cloudinit  
qm set $VMID --boot c --bootdisk scsi0  
qm set $VMID --vga std --serial0 socket --agent enabled=1  
qm resize $VMID scsi0 $DISK\_SIZE || true  
qm set $VMID --ciuser $CIUSER --cipassword $CIPASS --ipconfig0 ip=dhcp  
qm cloudinit update $VMID  
  
echo "[✓] VM $VMID ($NAME) ready to boot."

### Manual Equivalent

qm create 9000 --name deb12-docker-base --ostype l26 --memory 2048 --cores 2 --net0 virtio,bridge=vmbr0  
qm importdisk 9000 /var/lib/vz/template/iso/debian-12-genericcloud-amd64.qcow2 local-lvm  
qm set 9000 --scsihw virtio-scsi-pci --scsi0 local-lvm:vm-9000-disk-0  
qm set 9000 --ide2 local-lvm:cloudinit  
qm set 9000 --boot c --bootdisk scsi0  
qm set 9000 --vga std --serial0 socket --agent enabled=1  
qm resize 9000 scsi0 20G  
qm set 9000 --ciuser dockeruser --cipassword 'TempPassw0rd!' --ipconfig0 ip=dhcp  
qm cloudinit update 9000

---

## 🔹 Section 2. Guest: Prepare Debian (inside VM)

### Script: `prep\_guest\_deb12.sh`

# Inside the Debian 12 VM: enable serial login + install QEMU Guest Agent and Docker

# Serial login

# Guest agent

# Docker

#!/usr/bin/env bash  
  
set -euo pipefail  
export DEBIAN\_FRONTEND=noninteractive  
  
systemctl enable serial-getty@ttyS0.service  
systemctl start serial-getty@ttyS0.service  
  
apt-get update  
apt-get install -y qemu-guest-agent  
systemctl start qemu-guest-agent || true  
systemctl enable --now qemu-guest-agent.socket || true  
  
apt-get install -y ca-certificates curl gnupg lsb-release  
mkdir -m 0755 -p /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/debian/gpg | gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \  
https://download.docker.com/linux/debian $(lsb\_release -cs) stable" \  
 > /etc/apt/sources.list.d/docker.list  
  
apt-get update  
apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin  
  
systemctl enable --now docker  
usermod -aG docker dockeruser

### Manual Equivalent

sudo systemctl enable serial-getty@ttyS0.service  
sudo systemctl start serial-getty@ttyS0.service  
  
sudo apt-get update  
sudo apt-get install -y qemu-guest-agent  
sudo systemctl start qemu-guest-agent  
sudo systemctl enable --now qemu-guest-agent.socket  
  
sudo apt-get install -y ca-certificates curl gnupg lsb-release  
sudo mkdir -m 0755 -p /etc/apt/keyrings  
curl -fsSL https://download.docker.com/linux/debian/gpg | \  
 sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg  
echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \  
https://download.docker.com/linux/debian $(lsb\_release -cs) stable" | \  
 sudo tee /etc/apt/sources.list.d/docker.list  
sudo apt-get update  
sudo apt-get install -y docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-compose-plugin  
sudo systemctl enable --now docker  
sudo usermod -aG docker dockeruser

---

## 🔹 Section 3. Host: Clone Helper

### Script: `clone\_from\_template.sh`

# Clone from template 9000 and set per-VM creds

#!/usr/bin/env bash  
  
set -euo pipefail  
  
TEMPLATE\_VMID=9000  
NEWID=$1  
NEWNAME=$2  
PUBKEY=$3  
CIPASS=${4:-BetterPassw0rd!}  
NEWDISK=${5:-}  
  
qm clone $TEMPLATE\_VMID $NEWID --name $NEWNAME --full  
  
if [[ -n "$NEWDISK" && "$NEWDISK" != "-" ]]; then  
 qm resize $NEWID scsi0 $NEWDISK  
fi  
  
if [[ "$CIPASS" != "-" ]]; then  
 qm set $NEWID --ciuser dockeruser --cipassword "$CIPASS"  
else  
 qm set $NEWID --ciuser dockeruser  
fi  
  
qm set $NEWID --sshkey "$PUBKEY"  
qm cloudinit update $NEWID  
qm start $NEWID

### Manual Equivalent

qm clone 9000 101 --name docker01 --full  
qm resize 101 scsi0 40G # optional  
qm set 101 --ciuser dockeruser --cipassword 'BetterPassw0rd!'  
qm set 101 --sshkey ~/.ssh/id\_ed25519.pub  
qm cloudinit update 101  
qm start 101

### Post-clone FS growth (if root FS didn’t expand)

# OR sudo xfs\_growfs / # xfs

sudo apt-get install -y cloud-guest-utils  
lsblk  
sudo growpart /dev/sda 1  
sudo resize2fs /dev/sda1 # ext4

### Reset cloud-init (if you want to reapply settings)

sudo cloud-init clean  
sudo rm -rf /var/lib/cloud  
sudo reboot

---

## 🔹 Section 4. SSH Key Management

- \*\*Per-clone\*\*: inject one or more public keys with:

```bash

qm set <vmid> --sshkey /root/combined\_keys.pub

qm cloudinit update <vmid>

```

- File can contain multiple public keys (one per line).

---

# ✅ Summary Workflow

1. Build base VM (`build\_deb12\_docker\_base.sh` or manual).

2. Boot once, prep guest (`prep\_guest\_deb12.sh`).

3. Shut down → `qm template 9000`.

4. Clone (`clone\_from\_template.sh` or manual).

5. Optionally resize disk and grow FS.

6. Inject per-clone SSH keys.