Portainer Demo: Proxmox Clone + Cloud-Init + Docker Compose

# Contents

- portainer-userdata.yaml — cloud-init user-data that writes a Portainer compose stack and systemd unit.

- clone\_portainer.sh — host-side helper to clone from your Docker-ready template and bind the user-data.

# Prereqs

• Docker-ready Debian 12 template at VMID 9000 with QEMU guest agent enabled.

• Snippets storage on Proxmox at /var/lib/vz/snippets.

• Your SSH public key on the host (e.g., ~/.ssh/id\_ed25519.pub).

# Install

Copy the files:

sudo mkdir -p /var/lib/vz/snippets  
sudo cp portainer-userdata.yaml /var/lib/vz/snippets/  
sudo cp clone\_portainer.sh /root/  
sudo chmod +x /root/clone\_portainer.sh

# Clone & Boot

sudo /root/clone\_portainer.sh 220 portainer-demo ~/.ssh/id\_ed25519.pub 'BetterPassw0rd!'

Then visit https://<VM-IP>:9443 and complete Portainer’s first-time setup.

# Full: portainer-userdata.yaml

#cloud-config  
package\_update: false  
package\_upgrade: false  
  
write\_files:  
 - path: /opt/portainer/docker-compose.yml  
 permissions: '0644'  
 content: |  
 services:  
 portainer:  
 image: portainer/portainer-ce:latest  
 container\_name: portainer  
 restart: unless-stopped  
 ports:  
 - "9443:9443" # HTTPS UI  
 - "8000:8000" # Edge agent (optional)  
 volumes:  
 - /var/run/docker.sock:/var/run/docker.sock  
 - portainer\_data:/data  
 volumes:  
 portainer\_data:  
  
 - path: /etc/systemd/system/compose-portainer.service  
 permissions: '0644'  
 content: |  
 [Unit]  
 Description=Portainer via Docker Compose  
 After=docker.service network-online.target  
 Wants=docker.service network-online.target  
  
 [Service]  
 Type=oneshot  
 RemainAfterExit=yes  
 WorkingDirectory=/opt/portainer  
 ExecStart=/usr/bin/docker compose up -d  
 ExecStop=/usr/bin/docker compose down  
 TimeoutStartSec=0  
  
 [Install]  
 WantedBy=multi-user.target  
  
runcmd:  
 - [ mkdir, -p, /opt/portainer ]  
 - [ /bin/systemctl, daemon-reload ]  
 - [ /bin/systemctl, enable, --now, compose-portainer.service ]

# Full: clone\_portainer.sh

#!/usr/bin/env bash  
# clone\_portainer.sh: Clone from Docker-ready template and auto-bootstrap Portainer via cloud-init.  
# Usage: ./clone\_portainer.sh <new\_vmid> <name> <pubkey\_path> [password|-]  
set -euo pipefail  
  
TEMPLATE\_VMID=${TEMPLATE\_VMID:-9000}  
SNIPPETS\_DIR=${SNIPPETS\_DIR:-/var/lib/vz/snippets}  
USERDATA\_FILE=${USERDATA\_FILE:-portainer-userdata.yaml}  
CIUSER=${CIUSER:-dockeruser}  
  
NEWID=${1:?Usage: $0 <new\_vmid> <name> <pubkey\_path> [password|-]}  
NEWNAME=${2:?}  
PUBKEY=${3:?}  
CIPASS=${4:-"BetterPassw0rd!"}  
  
[[ -f "$PUBKEY" ]] || { echo "Public key not found: $PUBKEY"; exit 1; }  
[[ -f "$SNIPPETS\_DIR/$USERDATA\_FILE" ]] || { echo "User-data snippet not found: $SNIPPETS\_DIR/$USERDATA\_FILE"; exit 1; }  
  
echo "[\*] Cloning template $TEMPLATE\_VMID -> $NEWID ($NEWNAME)..."  
qm clone "$TEMPLATE\_VMID" "$NEWID" --name "$NEWNAME" --full  
  
# Optional: resize disk BEFORE first boot (uncomment to use)  
# qm resize "$NEWID" scsi0 30G  
  
echo "[\*] Configuring cloud-init (user, password/keys, DHCP, user-data)..."  
if [[ "$CIPASS" != "-" ]]; then  
 qm set "$NEWID" --ciuser "$CIUSER" --cipassword "$CIPASS"  
else  
 qm set "$NEWID" --ciuser "$CIUSER"  
fi  
qm set "$NEWID" --sshkey "$PUBKEY"  
qm set "$NEWID" --ipconfig0 ip=dhcp  
qm set "$NEWID" --cicustom "user=local:snippets/${USERDATA\_FILE}"  
qm cloudinit update "$NEWID"  
  
echo "[\*] Starting VM..."  
qm start "$NEWID"  
  
cat <<'EOS'  
  
[✓] VM started. On first boot, cloud-init writes /opt/portainer/docker-compose.yml,  
creates a systemd unit, and runs "docker compose up -d".  
  
Next steps:  
 1) Get the VM IP from the Proxmox GUI (Summary tab) once the QEMU agent reports it.  
 2) Open https://<VM-IP>:9443 in your browser (Portainer CE).  
 3) Create the initial admin user in the Portainer UI.  
  
Troubleshooting:  
 - Check service logs: sudo journalctl -u compose-portainer -n 200 --no-pager  
 - Check Docker status: docker ps  
 - If cloud-init didn't run: sudo cloud-init status --long  
EOS