CPTC10 Export — VM Access Reference

CPTC10-Export • qcow2 images for Proxmox • generated 2026-05-30

All exported VMs share a single standardized credential. **Password: `ExportControlled1337`** for every host. Linux hosts log in as `root`; Windows hosts as `Administrator`.

VM NAME	IP ADDRESS	ADMIN LOGIN	ADMIN PASSWORD	OS	OPENSTACK SPEC
PROD NETWORK — 10.0.1.0/24					
dockercompute	`10.0.1.5`	`root`	`ExportControlled1337`	Ubuntu 22.04	`xxlarge · 180 GB`
flakead	`10.0.1.6`	`Administrator`	`ExportControlled1337`	Win Server 2022	`large · 120 GB`
DEV NETWORK — 10.0.2.0/24					
dockercompute-dev	`10.0.2.5`	`root`	`ExportControlled1337`	Ubuntu 22.04	`xlarge · 180 GB`
networkdebug	`10.0.2.250`	`root`	`ExportControlled1337`	Ubuntu 22.04	`medium · 180 GB`
VDI NETWORK — 10.0.254.0/24					
win01	`10.0.254.101`	`Administrator`	`ExportControlled1337`	Win Server 2022	`large · 120 GB`
kali01	`10.0.254.201`	`root`	`ExportControlled1337`	Kali Linux	`large · 256 GB`
ns01	`10.0.254.253`	`root`	`ExportControlled1337`	Ubuntu 22.04	`small · 50 GB`

IP addresses are the original competition scheme (network base + host last-octet). Under Proxmox + DHCP these are the **intended** addresses — pin them with DHCP reservations (see below) or treat the table as a name reference and look up the leased IP.

Hardware — match the OpenStack flavor. These images are built and deployed on OpenStack. Size each Proxmox VM to the **OpenStack Spec** above: the `flavor` (instance_size) sets vCPU/RAM and the disk is the volume size. Look up the flavor's exact vCPU/RAM in your OpenStack flavor catalog and set Proxmox `--cores`/`--memory` to match. Under-provisioning relative to the flavor can break services that were tuned for it.

Creating these VMs in Proxmox (qcow2 import)

1. Import each qcow2 as a disk

1. Copy the `.qcow2` file to a Proxmox node (e.g. `scp host.qcow2 root@pve:/var/lib/vz/template/`).
2. Create an empty VM shell to get a VMID — via the GUI (*Create VM*, detach the disk) or CLI:

```
# VMID 110 — set --cores/--memory to match the OpenStack flavor (see table)
qm create 110 --name kali01 --cores <flavor vCPU> --memory <flavor MB> \
  --scsihw virtio-scsi-single --ostype l26 --agent enabled=1 \
  --net0 virtio,bridge=vmbr0
```

3. Import the qcow2 into your storage (e.g. `local-lvm`) and attach it as the boot disk:

```
qm importdisk 110 /var/lib/vz/template/kali01.qcow2 local-lvm
qm set 110 --scsi0 local-lvm:vm-110-disk-0
qm set 110 --boot order=scsi0
```

4. The disk size comes in with the qcow2; grow it to the flavor's disk if needed: `qm resize 110 scsi0 256G`.

5. Start it: `qm start 110`, then open the **Console** (noVNC) to confirm boot.

2. Disk bus & VirtIO drivers

- These images are built and run on OpenStack (KVM/QEMU + VirtIO), so the VirtIO storage and NIC drivers are **already baked into every guest — Windows included**. Attach the disk on the VirtIO bus and it boots directly; no driver injection needed.
- Attach as **VirtIO SCSI** (`--scsi0` + `--scsihw virtio-scsi-single`, as above), or match OpenStack's layout with **virtio-blk** (`qm set 110 --virtio0 local-lvm:vm-110-disk-0 --boot order=virtio0` → `/dev/vda`). Both boot.
- **Windows** (flakead, win01): also pass `--ostype win11` and a display `--vga std`; the VirtIO NIC binds automatically.
- **Firmware:** these are BIOS/MBR images — keep the Proxmox default `SeaBIOS` + `i440fx`. If a guest won't boot, it may be UEFI: switch the VM to `OVMF (UEFI)`.

3. Networking — DHCP

- Put `net0` on the bridge/VLAN that reaches your DHCP server (e.g. `virtio,bridge=vmbr0,tag=10`); the guests are set to DHCP and request an address on boot.
- Get each VM's NIC MAC for the reservation: `qm config <vmid> | grep net`.

4. pfSense static DHCP mappings

Pin each VM to its table IP. GUI: **Services** → **DHCP Server** → the interface tab serving that subnet → **Static Mappings** → **Add**. Required: **MAC Address** (from `qm config`); also set **IP Address** and **Hostname**.

```
Prod DHCP interface (10.0.1.0/24)
  dockercompute      MAC=<vm-mac>  IP=10.0.1.5      Hostname=dockercompute
  flakead            MAC=<vm-mac>  IP=10.0.1.6      Hostname=flakead

Dev DHCP interface (10.0.2.0/24)
  dockercompute-dev  MAC=<vm-mac>  IP=10.0.2.5      Hostname=dockercompute-dev
  networkdebug       MAC=<vm-mac>  IP=10.0.2.250    Hostname=networkdebug

VDI DHCP interface (10.0.254.0/24)
  win01              MAC=<vm-mac>  IP=10.0.254.101  Hostname=win01
  kali01             MAC=<vm-mac>  IP=10.0.254.201  Hostname=kali01
  ns01               MAC=<vm-mac>  IP=10.0.254.253  Hostname=ns01
```

Equivalent on-disk entry (ISC DHCP backend) — add one `<staticmap>` per VM inside the matching `<dhcpd>` interface block of `/conf/config.xml` (Diagnostics → Edit File), then reload. On the **Kea** backend, use the GUI instead.

```
<staticmap>
  <mac>aa:bb:cc:dd:ee:ff</mac>
  <ipaddr>10.0.1.5</ipaddr>
  <hostname>dockercompute</hostname>
</staticmap>
```

5. Connect

- **Linux** (root): `ssh root@<ip>` — password auth is enabled.
- **Windows** (Administrator): RDP to `<ip>:3389` — NLA disabled, firewall off.