🧰 Guia Único (Markdown) — Rocky Linux 10 + KVM/libvirt + LVM (RAW) + Bridge `br0` (instalação via **SSH**)

> Este runbook monta um **servidor de virtualização** no seu Lenovo ThinkCentre M710q com **Rocky Linux 10**, usando **um único SSD/SATA de 500 GB**, **storage LVM (RAW)** para máximo desempenho e **rede em bridge (`br0`)** para IP real da LAN. A criação das VMs será **100% via SSH/console** (sem GUI), com `virt-install --location` + `virsh console`.

🗺️ Sumário

⚙️ Pré-requisitos e BIOS

- Habilite **Intel VT-x** e **VT-d** na BIOS; mantenha **UEFI**.
- (Opcional) Desative o **Secure Boot** se for usar módulos de terceiros.
- No host, valide:

```
lscpu | grep -i virtualization     # deve mostrar VT-x
```

🧱 Layout de disco & LVM

> Host = Rocky 10 já instalado no **/dev/sdb**. Vamos reservar o **restante do disco** para as VMs como **LVM RAW** (melhor I/O que qcow2).

1. **Descobrir espaço livre** no final de `/dev/sdb`:

```
parted -s /dev/sdb unit GiB print free
```

> Anote o **Start** do último “Free Space” (ex.: `188GiB`). Usaremos abaixo como `<INICIO_LIVRE>`.

2. **Criar partição LVM das VMs** (sdb4) ocupando todo o livre:

```
parted -s /dev/sdb mkpart primary <INICIO_LIVRE> 100%
parted -s /dev/sdb set 4 lvm on
parted -s /dev/sdb name 4 pv_vms
partprobe /dev/sdb
```

> Alternativa à prova de erro (requer `dnf -y install gdisk`):

```
sgdisk -n 4:0:0 -t 4:8E00 -c 4:pv_vms /dev/sdb
```

3. **PV + VG das VMs**:

```
pvcreate /dev/sdb4
vgcreate vg_vms /dev/sdb4
```

4. **TRIM no LVM** (recomendado):

```
vim /etc/lvm/lvm.conf
# defina:
# issue_discards = 1
vgchange -ay
```

🔧 Pilha de virtualização

1. **Instalar pacotes**:

```
dnf -y groupinstall "Virtualization"
dnf -y install libvirtd virt-install virt-viewer libvirt-daemon-driver-storage
libvirt-daemon-driver-storage-logical
```

2. **Ativar libvirtd**:

```
systemctl enable --now libvirtd
usermod -aG  libvirt $USER   # depois faça logout/login
virt-host-validate
```

3. *(Opcional)* **Nested KVM** no host (Intel):

```
printf "options kvm_intel nested=1\n" > /etc/modprobe.d/kvm_intel.conf
systemctl stop libvirtd
modprobe -r kvm_intel || true; modprobe -r kvm || true
modprobe kvm_intel
systemctl start libvirtd
cat /sys/module/kvm_intel/parameters/nested   # Y
```

Bridge `br0` (IP real)

> Sua interface física é `enp0s31f6`. Faça os comandos abaixo **no host**. Execute localmente ou com acesso out-of-band (a rede cai por instantes).

1. **Identificar conexão ativa**:

```
nmcli --fields NAME,DEVICE connection show --active
# anote o nome da conexão atual -> <OLD>
```

2. **Criar a bridge (DHCP)**:

```
nmcli connection add type bridge ifname br0 con-name br0 ipv4.method auto
ipv6.method ignore
nmcli connection modify br0 bridge.stp no bridge.forward-delay 0
nmcli connection add type bridge-slave ifname enp0s31f6 master br0
nmcli connection modify "<OLD>" connection.autoconnect no
nmcli connection down "<OLD>"; nmcli connection up br0
ip -br a | egrep 'br0|enp0s31f6'  # br0 deve ter IP (ex.: 192.168.31.230/24)
```

> Para **IP fixo**, troque `ipv4.method auto` por `ipv4.method manual` e informe `ipv4.addresses`, `ipv4.gateway` e `ipv4.dns`.

Storage Pool LVM no libvirt

> Faremos o libvirt “enxergar” o **VG** `vg_vms` como um **pool lógico**.

```
virsh pool-define-as vms logical --source-name vg_vms --target /dev/vg_vms
virsh pool-start vms
virsh pool-autostart vms
virsh pool-list --all   # vms -> active, autostart yes
```

Criar VM via **SSH** (modo texto)

> Vamos criar a VM **rocky10_openshift** em **LVM RAW** (60 GB), topologia **6 vCPU (1×3×2)**, **16 GB RAM**, **bridge** `br0`, **OSINFO** `linux2024`, instalação **TUI** via `--location` + **serial**.

1. **Criar LV da VM**:

```
lvcreate -L 60G -n rocky10_openshift vg_vms
virsh pool-refresh vms
virsh vol-list vms
```

2. **Rodar o** `virt-install` **(modo texto/console)**:

```
virt-install --name rocky10_openshift \
  --virt-type kvm \
  --vcpus 6,sockets=1,cores=3,threads=2 \
  --cpu host-passthrough,cache.mode=passthrough \
  --memory 16000 \
  --osinfo detect=on,name=linux2024 \
  --iothreads 2 \
  --controller type=scsi,model=virtio-scsi \
  --disk
path=/dev/vg_vms/rocky10_openshift,format=raw,bus=scsi,cache=none,io=native,disc
ard=unmap \
  --network bridge=br0,model=virtio \
  --graphics none \
  --location /var/lib/libvirt/images/iso/Rocky-10.0-x86_64-minimal.iso \
  --extra-args 'inst.text console=ttyS0,115200n8' \
  --boot uefi
```

> **Sem DHCP?** Use IP fixo no instalador (ajuste interface `ens3` se necessário):

```
--extra-args 'inst.text console=ttyS0,115200n8
ip=192.168.31.240::192.168.31.1:255.255.255.0:rocky10:ens3:none
nameserver=1.1.1.1'
```

3. **Abrir o console do instalador** (em outra aba/SSH):

```
virsh console rocky10_openshift
# para sair do console: Ctrl + ]
```

4. **No Anaconda (TUI)**:

- Idioma/teclado.
- **Installation Destination** → disco `/dev/sda` (o LV RAW).
- **Network & Hostname** → ON (DHCP) ou IP fixo.
- **Root password** e usuário (ex.: jmsalles1 com sudo).

- **Begin Installation** → aguarde → **Reboot**.

> Se o console “sumir” no reboot, reconecte com `virsh console rocky10_openshift`.

📦 Pós-instalação do guest

Dentro da VM (via `virsh console` ou SSH):

```
sudo dnf -y install qemu-guest-agent
sudo systemctl enable --now qemu-guest-agent
sudo systemctl enable --now serial-getty@ttyS0.service
sudo grubby --update-kernel=ALL --args='console=ttyS0,115200n8'
sudo dnf -y update
```

No host, confirme IP da VM:

```
virsh domifaddr rocky10_openshift
```

🚀 Tuning de desempenho (host & VMs)

🖥️ Host

```
# tuned + ksmtuned
dnf -y install tuned ksmtuned
systemctl enable --now tuned ksmtuned
tuned-adm profile virtual-host

# TRIM semanal
systemctl enable --now fstrim.timer
fstrim -v /

# sysctl (latência e flush):
cat > /etc/sysctl.d/99-virt.conf <<'EOF'
vm.swappiness = 10
vm.dirty_background_ratio = 5
vm.dirty_ratio = 20
EOF
sysctl --system
```

🧠 Hugepages (host + VM)

Host (ex.: ~8 GB):

```
```

```
cat > /etc/sysctl.d/98-hugepages.conf <<'EOF'
vm.nr_hugepages = 4096
EOF
sysctl --system
```

VM (`virsh edit rocky10_openshift` → dentro de `<memoryBacking>`):

```
<memoryBacking><hugepages/></memoryBacking>
```

Disco & filas (VM)

`virsh edit rocky10_openshift` — garantir:

```
<iothreads>2</iothreads>
<devices>
  <controller type='scsi' model='virtio-scsi'>
    <driver queues='4'/>
  </controller>
  <disk type='block' device='disk'>
    <driver name='qemu' type='raw' cache='none' io='native' iothread='1'
discard='unmap'/>
    <source dev='/dev/vg_vms/rocky10_openshift'/>
    <target dev='sda' bus='scsi'/>
  </disk>
</devices>
```

NIC multiqueue (guest)

Dentro da VM (ajuste `ens3` se outro nome):

```
sudo ethtool -L ens3 combined 4
```

> (Opcional) persistir com service systemd.

Pinagem leve (opcional)

`virsh edit rocky10_openshift`:

```
<cputune>
  <vcpupin vcpu='0' cpuset='0'/>
  <vcpupin vcpu='1' cpuset='4'/>
  <emulatorpin cpuset='1'/>
  <iothreadpin iothread='1' cpuset='2'/>
</cputune>
```

🛠 Operações do dia a dia

Volume/disco

```
virsh vol-list vms
virsh vol-path --pool vms rocky10_openshift
```

Anexar 2º disco (20 GB)

```
lvcreate -L 20G -n rocky10_data vg_vms
virsh attach-disk rocky10_openshift /dev/vg_vms/rocky10_data vdb \
  --targetbus scsi --cache none --subdriver raw --live --persistent
```

Desanexar

```
virsh detach-disk rocky10_openshift vdb --live --persistent
```

Aumentar disco root (+20 GB)

```
lvextend -r -L +20G /dev/vg_vms/rocky10_openshift
# Em XFS dentro do guest, se precisar:
# sudo xfs_growfs /
```

Snapshot LVM (a quente)

```
lvcreate -s -n snap_r10 -L 8G /dev/vg_vms/rocky10_openshift
# copie o snapshot para backup externo…
lvremove -f /dev/vg_vms/snap_r10
```

🩹 Verificações úteis

```
# Host
pvs && vgs && lvs -o+devices
virsh pool-list --all
ip -br a | egrep 'br0|enp0s31f6'

# VM
virsh domifaddr rocky10_openshift
```

```
virsh domblklist rocky10_openshift
virsh dumpxml rocky10_openshift | sed -n '/<cpu/,/<\/cpu>/p'
```

🚑 Troubleshooting rápido

- `Falha ao obter MTU em 'br0'` → A bridge não existe. Refaça a bridge `br0`.
- `--osinfo` **obrigatório** → Use `--osinfo detect=on,name=linux2024` (ou liste: `virt-install --osinfo list | grep -i rocky`).
- **Sem saída no console** → Use `--location ... --extra-args 'inst.text console=ttyS0,115200n8'` e conecte com `virsh console`.
- **Topologia inválida** → Garanta `vcpus = sockets × cores × threads`. Para 6 vCPU: `--vcpus 6,sockets=1,cores=3,threads=2`.
- **Pool LVM não vê LVs** → `virsh pool-refresh vms`. Valide `pvs/vgs/lvs`.
- **Network NAT** `default` (se quiser usar) → crie `default.xml` em `/etc/libvirt/qemu/networks/` e `virsh net-define/default/net-start`.

☑ Pronto!

Com esse tutorial, seu host Rocky 10 está **otimizado** e você consegue **instalar e operar VMs via SSH** com **desempenho alto** usando **LVM RAW** e **bridge** `br0`. Se quiser, posso gerar um **XML completo** da `rocky10_openshift` com todos os blocos de performance prontos para colar via `virsh edit`.