QEMU Machine emulator and virtualizer

START HERE

`qemu-img create -f qcow2 <disk.qcow2> 20G`
Make a virtual disk

`qemu-system-x86_64 -cdrom <install.iso> -hda <disk.qcow2> -m 2G -boot d`
Install an OS

`qemu-system-x86_64 -accel kvm -cpu host -m 4G -smp 4 -hda <disk.qcow2>`
Run it fast

`-nic user,hostfwd=::2222-:22`
SSH into the guest

`-nographic`
Run in the terminal

`Ctrl+a x`
Quit a -nographic guest

`qemu-img info <disk>`
Size, format, backing

`-snapshot`
Throw away changes

CONCEPTS

One binary per CPU architecture (qemu-system-x86_64, -aarch64, ...); without -accel kvm the guest is emulated and slow.

Host Linux, /dev/kvm
- **qemu-system-*** one process per VM
  - **Machine** -machine, -cpu, -m, -smp
    - **Devices** -drive, -netdev, -device

Options are order-independent but -netdev and -chardev ids must match the -device or -serial that uses them.

The guest's disk is a file on the host; qcow2 grows on demand and supports snapshots and backing files.

RUN A VM

`qemu-system-x86_64 -hda <disk>`
Boot from a disk image

`-accel kvm`
Hardware acceleration (Linux)
Needs access to /dev/kvm. -enable-kvm is the older spelling of the same thing.

`-cpu host`
Pass the host CPU model through

`-m <size>`
Guest RAM, e.g. 2G

`-smp <n>`
Number of virtual CPUs

`-name <vm>`
Name the VM (window, process)

`-machine q35`
Pick a modern chipset

`-machine help`
List machine types

`-daemonize`
Detach into the background

`-loadvm <tag>`
Start from a saved snapshot

DISKS AND BOOT

`-drive file=<disk>,format=qcow2,if=virtio`
Attach a virtio disk

`-cdrom <iso>`
Attach an optical image

`-boot d`
Boot from the CD first

DISKS AND BOOT (cont.)

`-boot order=<cdn>`
Set boot device order

`-snapshot`
Discard disk writes on exit
Writes go to temporary files; press Ctrl+a s in the console or run commit in the monitor to keep them.

`-drive file=<disk>,cache=none,aio=native`
Direct I/O for speed

`-drive if=pflash,format=raw,readonly=on,file=<OVMF_CODE.fd>`
UEFI firmware (OVMF)

`-kernel <bzImage> -append "<cmdline>"`
Boot a kernel directly

DISPLAY AND CONSOLE

`-nographic`
No window; serial on the terminal

`-display gtk`
Choose the display backend

`-display vnc=:<n>`
VNC server on port 5900+n

`-vga virtio`
Pick the virtual graphics card

`-serial mon:stdio`
Serial and monitor on the terminal

`-monitor stdio`
Monitor on the terminal

`-qmp unix:<sock>,server,nowait`
Machine-readable control socket

NETWORK

`-nic user`
Simple NAT, no setup (default)

`-nic user,hostfwd=::<host>-:<guest>`
Forward a host port to the guest

`-nic user,model=virtio-net-pci`
Choose the NIC model

`-nic tap,script=no,ifname=<tap>`
Attach to a host tap device

`-nic bridge,br=<br0>`
Join a host bridge (needs helper)

`-netdev user,id=<n0>,hostfwd=tcp::<host>-:<guest>`
Same, long form

NETWORK (cont.)

`-device virtio-net-pci,netdev=<n0>`
Pair a NIC with a -netdev

DEVICES AND SHARING

`-device help`
List available devices

`-device usb-host,vendorid=<vid>,productid=<pid>`
Pass a host USB device through

`-virtfs local,path=<dir>,mount_tag=<tag>,security_model=mapped-xattr`
Share a folder with the guest (9p)

QEMU-IMG

`qemu-img create -f qcow2 <disk> <size>`
New disk image

`qemu-img info <disk>`
Show format and sizes

`qemu-img convert -O qcow2 <in> <out>`
Convert between formats

`qemu-img convert -c -p -O qcow2 <in> <out>`
Compress with progress

`qemu-img resize <disk> +<size>`
Grow an image
Shrinking needs --shrink and will destroy data past the new end unless the guest filesystem was shrunk first.

`qemu-img create -f qcow2 -b <base> -F qcow2 <overlay>`
Overlay on a backing file

`qemu-img commit <overlay>`
Merge overlay into its base

`qemu-img check <disk>`
Check for corruption

`qemu-img snapshot -l <disk>`
List internal snapshots

`qemu-img snapshot -c <tag> <disk>`
Create a snapshot

`qemu-img snapshot -a <tag> <disk>`
Restore a snapshot

`qemu-img convert -f raw -O qcow2 <in.img> <out.qcow2>`
Raw to qcow2

Never run qemu-img write commands on an image a running VM uses; info -U is the read-only exception.

QEMU-IMG (cont.)

Common image formats

-f / -O	Use for
`raw`	Plain bytes, fastest, no features
`qcow2`	Sparse, snapshots, backing files
`vmdk`	VMware images
`vdi`	VirtualBox images
`vhdx`	Hyper-V images

MONITOR

`Ctrl+a c`
Switch console / monitor (mon:stdio)

`Ctrl+a x`
Exit the emulator

`info status`
Running or paused?

`info block`
Show attached drives

`stop / cont`
Pause / resume the guest

`savevm <tag>`
Save full VM state (needs qcow2)

`loadvm <tag>`
Restore VM state

`system_powerdown`
Send an ACPI power button press

`quit`
Kill the VM now

`change <dev> <file>`
Swap the medium in a drive

RECIPES

Install an OS

```
qemu-img create -f qcow2
vm.qcow2 20G
qemu-system-x86_64 \
  -accel kvm -cpu host -m 4G \
  -drive
file=vm.qcow2,if=virtio \
  -cdrom <install.iso> -boot d
```

Headless VM with SSH

```
qemu-system-x86_64 -nographic \
  -accel kvm -cpu host -m 2G \
  -drive
file=vm.qcow2,if=virtio \
  -nic user,hostfwd=::2222-:22
# then, from the host
ssh -p 2222 <user>@localhost
```

Disposable test run

```
qemu-img create -f qcow2 \
  -b base.qcow2 -F qcow2
t.qcow2
qemu-system-x86_64 -accel kvm \
  -m 2G -drive
file=t.qcow2,if=virtio
rm t.qcow2
```

Boot a kernel in the terminal

```
qemu-system-x86_64 -nographic \
  -accel kvm -m 1G \
  -kernel <bzImage> \
  -append "console=ttyS0 root=/
dev/vda" \
  -drive file=<rootfs>,format=
raw,if=virtio
```