

Using the QCOW2 disk format in Proxmox

4 4sysops.com/archives/using-the-qcow2-disk-format-in-proxmox

June 6, 2023

Surender Kumar Tue, Jun 6 2023 virtualization, storage, proxmox 3

QCOW2 is a virtual disk image format used by Proxmox Virtual Environment (Proxmox VE), an open-source virtualization management platform. In Proxmox VE, QCOW2 is the default virtual disk image format for virtual machines (VMs).

Surender Kumar

Surender Kumar has over 15 years of experience in server and network administration. His fields of interest are Windows Servers, Active Directory, PowerShell, Web Servers, Networking, Linux, Virtualization, Docker, and Kubernetes. He loves writing for [his blog](#).

What is QCOW2?

QCOW2 stands for QEMU Copy-On-Write version 2, which is essentially a file format for disk images used by Quick Emulator (QEMU). QEMU is open-source virtualization software commonly used with a kernel-based virtual machine (KVM). In Proxmox, both KVM and QEMU work together as a complete virtualization platform.

Below are a few characteristics of the QCOW disk image format:

- The initial file size is small
- Ability to grow as new data is added
- Snapshot capability
- Supports Zlib-based compression
- Supports AES encryption to encrypt data at rest

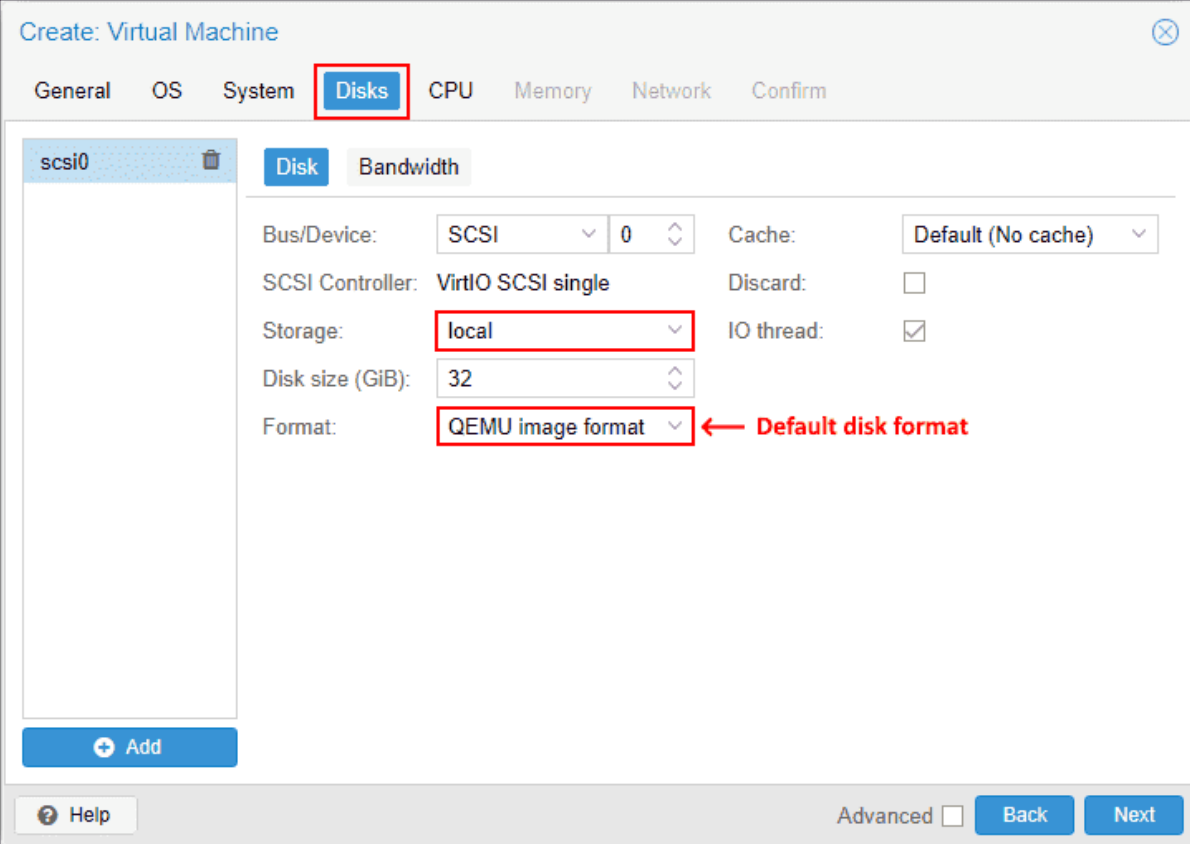
QCOW2 vs. raw disk?

With the QCOW2 disk format, mapping between logical and physical blocks is needed, which makes it slightly slower compared to a raw disk. The raw disk, on the other hand, preallocates the entire space to the disk file, so you get a bigger disk file; however, since there is no additional mapping overhead, you get better performance. People tend to get performance improvement by enabling the write cache, but doing so also increases the risk of data loss. Fortunately, the performance of QCOW2 is now significantly improved, so you may want to disable the write cache to reduce the data loss risk. If you do, remember that the QCOW2 format is more sophisticated than the raw format.

In a nutshell, both the QCOW2 and raw disk formats have pros and cons. You can choose QCOW2 if you need more features, but if I/O performance is your main priority, choose a raw disk format. If you have to choose a raw disk format, remember to use a Proxmox storage type that supports snapshots.

Create a QCOW2 disk

In Proxmox, when a new VM is created in file-based storage, the QCOW2 disk format is automatically selected. See the screenshot below for reference:



The screenshot shows the 'Create: Virtual Machine' window in Proxmox, specifically the 'Disks' tab. The 'Format' dropdown menu is set to 'QEMU image format', which is highlighted with a red box and a red arrow pointing to it with the text 'Default disk format'. Other settings include 'Storage: local', 'Disk size (GiB): 32', and 'Cache: Default (No cache)'. The 'Disks' tab is also highlighted with a red box.

Proxmox uses the QCOW2 disk format by default when using file based storage

Thus, unless you manually change the disk format when creating a new VM or adding a new disk to an existing VM, the disk format is QCOW2, by default, provided that you are storing it in file-based storage. The next section explains how to convert a virtual disk to QCOW2.

Convert and import a disk

When you use a tool like the VMware vCenter Converter to migrate your VMs, you get virtual disks in the *vmdk* or *raw* format. In addition, there are many virtual appliances available in *vmdk* or *raw* format. So, whenever you want to import such disks into Proxmox, you can use the *qm disk import* command, as shown below:

```
qm disk import <vmid> <source_path> <destination_storage> [options]
```

Recently, I migrated a Metasploitable VM from Hyper-V to Proxmox using the following commands:

```
# create a new VM without a virtual disk
```

```
qm create 108 --name metasploitable --memory 2048 --sockets 1 --cores 2 --net0  
virtio,bridge=vmbr0
```

```
# import a vmdk disk as qcow2 format in proxmox
```

```
qm disk import 108 /mnt/metasploitable.vmdk local --format qcow2
```

```
# attach the qcow2 disk and configure the boot order
```

```
qm set 108 --scsi0 local:108/vm-108-disk-0.qcow2,discard=on --boot order='scsi0;net0'
```

```
# start the new VM
```

```
qm start 108
```



Import and convert a virtual disk to QCOW2 using the Proxmox command line

The main command to note here is the `qm disk import 108 /mnt/metasploitable.vmdk local --format qcow2`, which converts and imports a `vmdk` disk in the QCOW2 format from the `/mnt` directory to VM ID `108`. Similarly, you can use the `qemu-img convert` command to convert a disk from one format to another, as shown in the command below:

```
qemu-img convert -f raw /mnt/usb/windows-10.vmdk -O qcow2 /var/lib/vz/images/109/vm-109-disk-0.qcow2
```

The `-f` option specifies the source disk format, and the `-O` option specifies the output disk format. The above command converts a `raw` disk located in `/mnt/usb` directory to the QCOW2 format in the `/var/lib/vz/images/109` directory on the Proxmox server. You can then create a new VM in Proxmox by attaching the new QCOW2 disk.

If you have a virtual appliance in Open Virtualization Appliance (OVA) format, you first need to extract it using the `tar` command, since OVA is essentially an archive containing an Open Virtualization Format (OVF) file and some other files. You can then import the OVF file with the help of the `qm importovf` command, as shown below:

```
tar xvf <ova_file_path>
```

```
qm importovf <unused_vm_id> <ovf_file_path> <storage> --format qcow2
```



Import a virtual appliance packaged as OVF in Proxmox

Again, remember that the `--format qcow2` option is important, or the disk will be imported as `raw`. If you have a VM in Proxmox that you want to move to a different storage, check out [this post](#).

Shrink a QCOW2 disk manually

After using a QCOW2 disk for some time, the file size becomes bigger than the actual data stored in it. This happens because the guest OS marks the files as deleted, but the unused blocks aren't automatically emptied. In this situation, you might need to manually

shrink the QCOW2 disk.

Before doing anything, shut down the original VM, and make sure you have a good backup of the virtual disk file. To locate the virtual disk file, you can use the Proxmox VE Storage Manager (*pvesm*) command line tool, as shown below:

```
# view the Proxmox storage status
```

```
pvesm status
```

```
# list storage
```

```
pvesm list local
```

```
# locate the path of a virtual disk file
```

```
pvesm path local:107/vm-107-disk-3.qcow2
```

```
# duplicate the original virtual disk file
```

```
cp /var/lib/vz/images/107/vm-107-disk-3.qcow2 /var/lib/vz/images/107/vm-107-disk-3.qcow2.bak
```

■

Locate and back up a QCOW2 disk on the Proxmox server

You can see that the current file size is 24 GB. Once you have the backup file, you can move the original QCOW2 disk to another storage.

```
mv vm-107-disk-3.qcow2 ~/orig
```

To shrink the QCOW2 file, you can use the *qemu-img convert* command, as shown below:

```
# shrink the virtual disk file without compression
```

```
qemu-img convert -O qcow2 vm-107-disk-3.qcow2.bak vm-107-disk-3.qcow2
```

```
# shrink the virtual disk file with compression
```

```
qemu-img convert -O qcow2 -c vm-107-disk-3.qcow2.bak vm-107-disk-3.qcow2
```

—

Shrink a QCOW2 disk with compression on the Proxmox server

The first command shrinks a QCOW2 file without compression, and the second command shrinks it with compression. Compression takes longer but results in a much smaller file.

—

Viewing the QCOW2 file size after shrinking

To obtain more information about the virtual disk, use the `qemu-img info` command, as shown below:

```
qemu-img info vm-107-disk-3.qcow2
```

—

View the detailed information about the virtual disk

You can see that the virtual disk size is 64 GiB, whereas the actual QCOW2 file size is 18.5 GiB. When compression is done, start your VM. If everything works as expected, you can delete the backup files to reclaim the free space.

Enable TRIM support

Manual compression works when there are just a few disk images. When there are a lot of VMs, you need a way to trim the unused blocks automatically. This is where trim (also known as `fstrim` or `discard`) comes into the picture. Note that trim requires SSD or a storage type that supports thin provisioning (e.g., `thin-lvm` and `ZFS`). Since Proxmox allows you to enable SSD emulation, you do not necessarily need an SSD-backed storage. Furthermore, for the trim operation to work effectively, you need to enable the *Discard* option on the VM disk, as shown in the screenshot below:

—

Enable the discard option on a VM disk in Proxmox

Additionally, you also need to make sure that trim is enabled in the guest OS so that it can send the information about unused blocks to the underlying storage. Fortunately, most modern operating systems (including Linux and Windows) have trim enabled by default. To check it on Windows, run the following command in an elevated command prompt:

```
fsutil behavior query disableddeletenotify
```

—

Check whether trim support is enabled on Windows

If you see 0 in the output, you're all set. To enable it, run the following command:

```
fsutil behavior set disableddeletenotify 0
```

Similarly, the same thing on Linux can be accomplished with the `fstrim -av` command. The `systemctl status fstrim.timer` command shows the status of the `fstrim` service and the configured discard timer on a Linux system.

—

Checking the `fstrim` timer on Linux

Configuring trim support is a one-time thing. Once it is configured correctly, you no longer need to worry about manually shrinking and optimizing the virtual disks. Everything will be taken care of automatically for you.

That was all for this post. You just learned about the QCOW2 disk image format and how to convert/import a virtual disk from various formats to the Proxmox server.