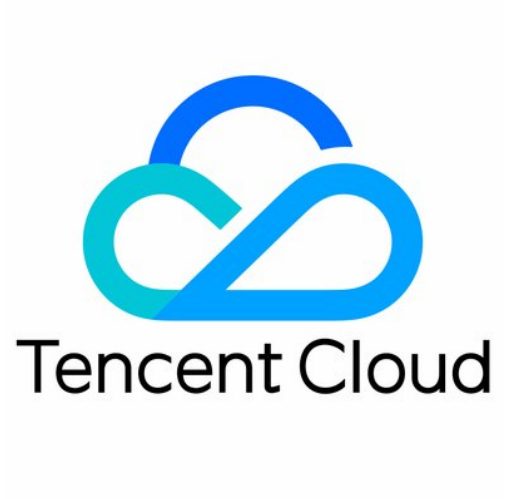
Cloud Block Storage

Best Practices

Product Documentation



Contents

Best Practices
Measuring Cloud Disk Performance

Last updated：2022-09-29 17:31:51

Important Notes

alarm

- This document uses the FIO test tool. **To avoid damaging important system files, do not perform any FIO test on the system disk**.
- **To avoid data corruption due to damaged metadata of the underlying file system, do not perform stress tests on the business data disk. Instead, use the cloud disk with no business data stored for tests and create a snapshot in advance to protect your data.**
- Ensure the `/etc/fstab` file configuration items **do not contain** the mounting configuration of the disk to be tested. Otherwise, CVM may fail to launch.

Metrics

Tencent Cloud CBS devices vary in performance and price by type. For more information, see Cloud Disk Types. Because different applications have different workloads, if the number of I/O requests is low, the cloud disk may not play its full performance.
The following metrics are generally used to measure the performance of a cloud disk:

- IOPS: Read/write count per second. IOPS is determined by the underlying drive type of the storage device.
- Throughput: Read/written data volume per second, in MB/s.
- Latency: Time from I/O operation sending to receiving, in microseconds.

Test Tool

FIO is a tool for testing disk performance. It is used to perform stress test and verification on hardware. This document uses FIO as an example.
We recommend that you use FIO together with libaio's I/O engine to perform the test. Install FIO and libaio with reference to Tool Installation.

Recommended Test Objects

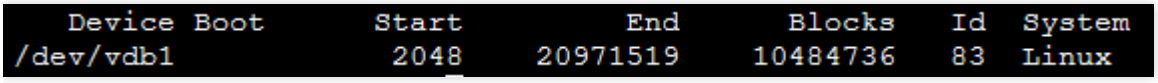
- We recommend that you perform FIO test on empty disks that do not store important data, and re-create the file system after completing the test..
- When testing disk performance, we recommend that you directly test raw data disks (such as /dev/vdb).
- When testing file system performance, we recommend that you specify the specific file (such as /data/file) for testing.

Tool Installation

1. Log in to the CVM as instructed in Log in to Linux Instance Using Standard Login Method. Here, take the CVM running CentOS 7.6 OS as an example.
2. Run the following command to check whether the cloud disk is 4KiB-aligned.

```
fdisk -lu
```

As shown below, if the Start value in the command output is divisible by 8, then the disk is 4KiB-aligned. Otherwise, complete 4KiB alignment before testing.



3. Run the following commands in sequence to install the testing tools, FIO and libaio.

```
yum install libaio -y
```

```
yum install libaio-devel -y
```

```
yum install fio -y
```

Once completed, start testing the cloud disk performance as instructed in the test example below.

Test Example

The testing formulas for different scenarios are basically the same, except the `rw` , `iodepth` , and `bs` (block size) parameters. For example, the optimal `iodepth` for each workload is different as it depends on the sensitivity of your application to the IOPS and latency.

Parameters:

Parameter	Description	Sample value
bs	Block size of each request, which can be 4 KB, 8 KB, or 16 KB.	4k
ioengine	I/O engine. We recommend that you use Linux's async I/O engine.	libaio
iodepth	Queue depth of an I/O request.	1
direct	Specifies direct mode. • True (1) indicates that the O_DIRECT identifier is specified, the I/O cache will be ignored, and data will be written directly. • False (0) indicates that the O_DIRECT identifier is not specified. The default is True (1).	1
rw	Read and write mode. Valid values include read, write, randread, randwrite, randrw, and rw, readwrite.	read
time_based	Specifies that the time mode is used. As long as FIO runs based on the time, it is unnecessary to set this parameter.	N/A
runtime	Specifies the test duration, which is the FIO runtime.	600
refill_buffers	FIO will refill the I/O buffer at every submission. The default setting is to fill the I/O buffer only at the start and reuse the data.	N/A
norandommap	When performing random I/O operations, FIO overwrites every block of the file. If this parameter is set, a new offset will be selected without viewing the I/O history.	N/A
randrepeat	Specifies whether the random sequence is repeatable. True (1) indicates that the random sequence is repeatable. False (0) indicates that the random sequence is not repeatable. The default value is True (1).	0
group_reporting	When multiple jobs are concurrent, statistics for the entire group are printed.	N/A
`name`	Name of the job.	fio-read
size	Address space of the I/O test.	100 GB
filename	Test object, which is the name of the disk to be tested.	/dev/sdb

Common use cases are as follows:

Show All

bs = 4k iodepth = 1: Random read/write test, which can reflect the disk latency.

展开&收起

alarm

- **To avoid damaging important files in the system, do not perform FIO test on the system disk.**
- **To avoid data corruption due to damaged metadata of the underlying file system, do not perform stress tests on the business data disk. Instead, use the cloud disk with no business data stored for tests and create a snapshot in advance to protect your data.**

Run the following command to test the random read latency of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=1 -direct=1 -rw=randread -time_based -runtim
e=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randr
ead-lat --size=10G -filename=/dev/vdb
```

Run the following command to test the random write latency of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=1 -direct=1 -rw=randwrite -time_based -runti
me=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-rand
write-lat --size=10G -filename=/dev/vdb
```

Run the following command to test the random hybrid read and write latency performance of an SSD cloud disk:

```
fio --bs=4k --ioengine=libaio --iodpth=1 --direct=1 --rw=randrw --time_based --ru
ntime=100 --refill_buffers --norandommap --randrepeat=0 --group_reporting -rw --s
ize=1G --filename=/dev/vdb
```

The following figure shows the command output:

```
fio-read: (g=0): rw=randrw, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=libaio, iodepth=1
fio-3.1
Starting 1 process
Jobs: 1 (f=1): [m(1)][100.0%][r=3411KiB/s,w=3603KiB/s][r=852,w=900 IOPS][eta 00m:00s]
fio-read: (groupid=0, jobs=1): err= 0: pid=2377: Thu Jun 13 18:23:47 2019
   read: IOPS=880, BW=3523KiB/s (3607kB/s)(344MiB/100001msec)
    slat (nsec): min=2905, max=62479, avg=5254.61, stdev=2075.46
    clat (usec): min=205, max=6921, avg=463.65, stdev=259.48
     lat (usec): min=209, max=6925, avg=469.13, stdev=259.56
    clat percentiles (usec):
     |  1.00th=[  245],  5.00th=[  269], 10.00th=[  293], 20.00th=[  375],
     | 30.00th=[  400], 40.00th=[  416], 50.00th=[  437], 60.00th=[  457],
     | 70.00th=[  478], 80.00th=[  498], 90.00th=[  545], 95.00th=[  619],
     | 99.00th=[ 2057], 99.50th=[ 2376], 99.90th=[ 3294], 99.95th=[ 4015],
     | 99.99th=[ 6259]
   bw (  KiB/s): min= 2168, max= 4024, per=100.00%, avg=3522.64, stdev=310.95, samples=200
   iops        : min=  542, max= 1006, avg=880.66, stdev=77.74, samples=200
  write: IOPS=877, BW=3511KiB/s (3595kB/s)(343MiB/100001msec)
    slat (nsec): min=2981, max=58808, avg=5377.71, stdev=2079.36
    clat (usec): min=421, max=10492, avg=659.10, stdev=219.96
     lat (usec): min=428, max=10496, avg=664.70, stdev=220.05
    clat percentiles (usec):
     |  1.00th=[  490],  5.00th=[  523], 10.00th=[  545], 20.00th=[  562],
     | 30.00th=[  578], 40.00th=[  594], 50.00th=[  611], 60.00th=[  635],
     | 70.00th=[  660], 80.00th=[  693], 90.00th=[  783], 95.00th=[  914],
     | 99.00th=[ 1516], 99.50th=[ 1926], 99.90th=[ 3261], 99.95th=[ 3982],
     | 99.99th=[ 5342]
   bw (  KiB/s): min= 2296, max= 4008, per=100.00%, avg=3510.84, stdev=305.46, samples=200
   iops        : min=  574, max= 1002, avg=877.71, stdev=76.36, samples=200
  lat (usec)   : 250=0.76%, 500=40.51%, 750=51.04%, 1000=5.03%
  lat (msec)   : 2=1.90%, 4=0.71%, 10=0.05%, 20=0.01%
  cpu          : usr=0.50%, sys=1.52%, ctx=175841, majf=0, minf=29
  IO depths    : 1=100.0%, 2=0.0%, 4=0.0%, 8=0.0%, 16=0.0%, 32=0.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
```

bs = 128k iodepth = 32: Sequential read/write test, which can reflect the disk throughput.

展开&收起

alarm

- **To avoid damaging important files in the system, do not perform FIO test on the system disk.**
- **To avoid data corruption due to damaged metadata of the underlying file system, do not perform stress tests on the business data disk. Instead, use the cloud disk with no business data stored for tests and create a snapshot in advance to protect your data.**

Run the following command to test the sequential read throughput bandwidth:

```
fio -bs=128k -ioengine=libaio -iodepth=32 -direct=1 -rw=read -time_based -runtime
=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-read-t
hroughput --size=10G -filename=/dev/vdb
```

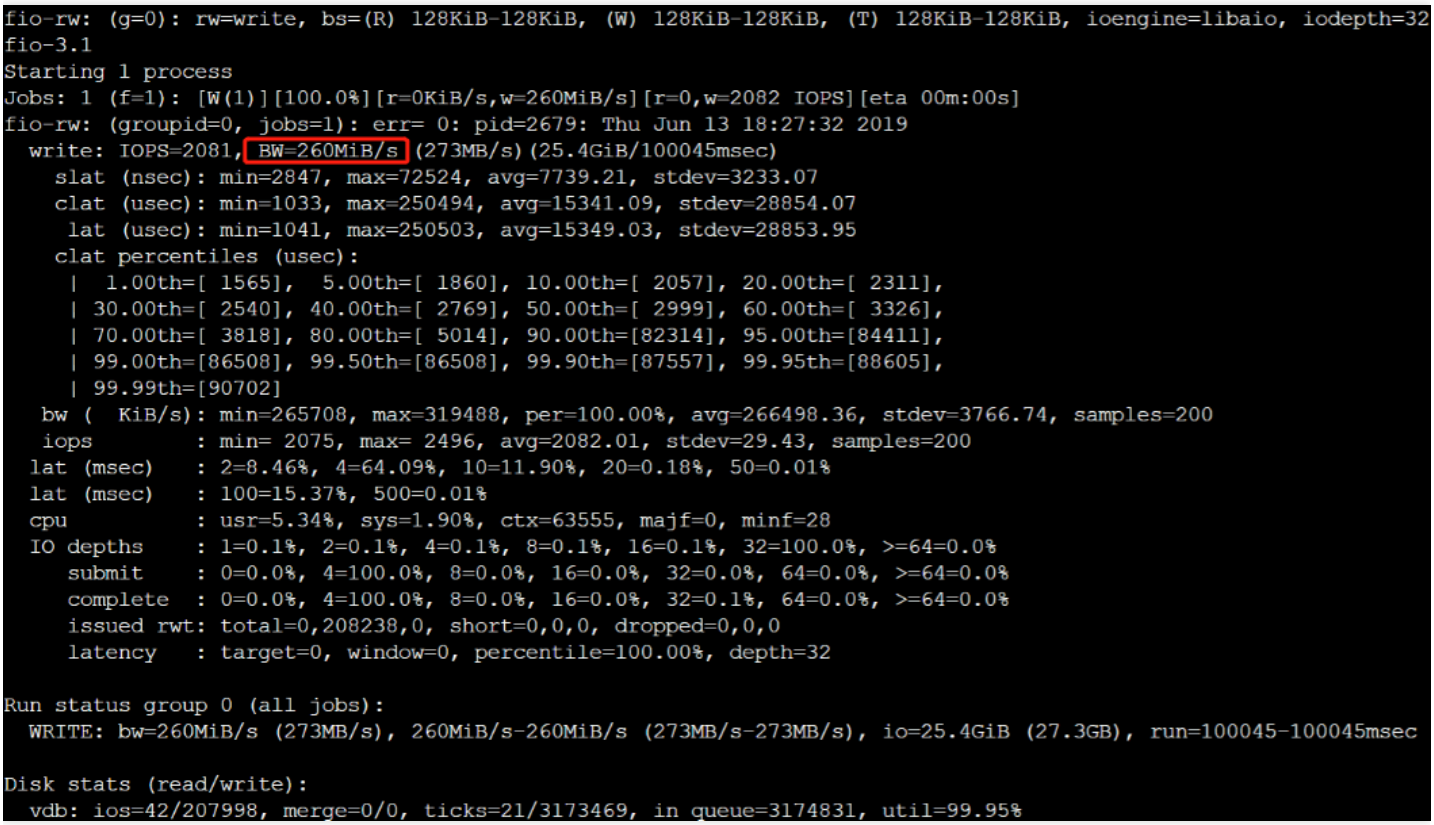
Run the following command to test the sequential write throughput bandwidth:

```
fio -bs=128k -ioengine=libaio -iodepth=32 -direct=1 -rw=write -time_based -runtim
e=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-write
-throughput --size=10G -filename=/dev/vdb
```

Run the following command to test the sequential read throughput performance of an SSD cloud disk:

```
fio --bs=128k --ioengine=libaio --iodepth=32 --direct=1 --rw=read --time_based --
runtime=100 --refill_buffers --norandommap --randrepeat=0 --group_reporting --nam
e=fio-rw --size=1G --filename=/dev/vdb
```

The following figure shows the command output:



bs = 4k iodepth = 32: Random read/write test, which can reflect the disk IOPS.

展开&收起

alarm

- **To avoid damaging important files in the system, do not perform FIO test on the system disk.**

- **To avoid data corruption due to damaged metadata of the underlying file system, do not perform stress tests on the business data disk. Instead, use the cloud disk with no business data stored for tests and create a snapshot in advance to protect your data.**

Run the following command to test the random read IOPS of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=32 -direct=1 -rw=randread -time_based -runti
me=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-rand
read-iops --size=10G -filename=/dev/vdb
```

Run the following command to test the random write IOPS of the disk:

```
fio -bs=4k -ioengine=libaio -iodepth=32 -direct=1 -rw=randwrite -time_based -runt
ime=600 -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-ran
dwrite-iops --size=10G -filename=/dev/vdb
```

Test the random read IOPS performance of an SSD cloud disk. The following figure shows the command output.

```
[root@VM_16_21_centos ~]# fio -bs=4k -ioengine=libaio -iodepth=32 -direct=1 -rw=randread -time_based
 -runtime=300  -refill_buffers -norandommap -randrepeat=0 -group_reporting -name=fio-randread  --siz
e=100G -filename=/dev/vdc
fio-randread: (g=0): rw=randread, bs=(R) 4096B-4096B, (W) 4096B-4096B, (T) 4096B-4096B, ioengine=lib
aio, iodepth=32
fio-3.1
Starting 1 process
Jobs: 1 (f=1): [r(1)][100.0%][r=18.8MiB/s,w=0KiB/s][r=4804,w=0 IOPS][eta 00m:00s]
fio-randread: (groupid=0, jobs=1): err= 0: pid=2689: Tue Jul 16 13:39:33 2019
   read: IOPS=4800, BW=18.8MiB/s (19.7MB/s)(5626MiB/300017msec)
    slat (usec): min=2, max=3183, avg= 7.55, stdev=14.34
    clat (usec): min=209, max=777668, avg=6656.04, stdev=45385.49
     lat (usec): min=371, max=777673, avg=6664.08, stdev=45385.46
    clat percentiles (usec):
     |  1.00th=[   635],  5.00th=[   832], 10.00th=[   938], 20.00th=[  1090],
     | 30.00th=[  1237], 40.00th=[  1352], 50.00th=[  1467], 60.00th=[  1582],
     | 70.00th=[  1713], 80.00th=[  1991], 90.00th=[  9372], 95.00th=[ 19006],
     | 99.00th=[ 38536], 99.50th=[341836], 99.90th=[734004], 99.95th=[750781],
     | 99.99th=[767558]
   bw (  KiB/s): min=  256, max=38424, per=100.00%, avg=19202.16, stdev=13626.10, samples=600
   iops        : min=   64, max= 9606, avg=4800.52, stdev=3406.52, samples=600
  lat (usec)   : 250=0.01%, 500=0.12%, 750=2.64%, 1000=10.86%
  lat (msec)   : 2=66.45%, 4=5.87%, 10=4.59%, 20=6.49%, 50=2.32%
  lat (msec)   : 100=0.01%, 500=0.36%, 750=0.26%, 1000=0.05%
  cpu          : usr=1.38%, sys=5.00%, ctx=233328, majf=0, minf=63
  IO depths    : 1=0.1%, 2=0.1%, 4=0.1%, 8=0.1%, 16=0.1%, 32=100.0%, >=64=0.0%
     submit    : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.0%, 64=0.0%, >=64=0.0%
     complete  : 0=0.0%, 4=100.0%, 8=0.0%, 16=0.0%, 32=0.1%, 64=0.0%, >=64=0.0%
     issued rwt: total=1440148,0,0, short=0,0,0, dropped=0,0,0
     latency   : target=0, window=0, percentile=100.00%, depth=32

Run status group 0 (all jobs):
   READ: bw=18.8MiB/s (19.7MB/s), 18.8MiB/s-18.8MiB/s (19.7MB/s-19.7MB/s), io=5626MiB (5899MB), run=
300017-300017msec

Disk stats (read/write):
  vdc: ios=1440052/0, merge=0/0, ticks=9478008/0, in_queue=9485217, util=99.98%
```

Building LVM Logic Volumes with Multiple Elastic Cloud Disks

Last updated：2023-12-22 10:50:53

Introduction to LVM

Logical Volume Manager (LVM) creates a logical layer over your disks or partitions to divide them into physical extent (PE) units with the same size. This categorizes disks or partitions into a volume group (VG), on which you can create a logical volume (LV), and then create file systems on the LV.
Different from direct disk partitioning, LVM allows elastic scaling of file system.
The file system is not limited by the size of a physical disk. Instead, it can be distributed among multiple disks:
For example, you can purchase three 4-TB elastic cloud disks, and use LVM to create a massive file system nearly 12 TB.
You can resize the LVs dynamically instead of repartitioning your disks.
When the LVM VG capacity cannot meet your needs, you can purchase a elastic cloud disk, attach it to your CVM instance, and add it to the LVM VG to expand capacity.

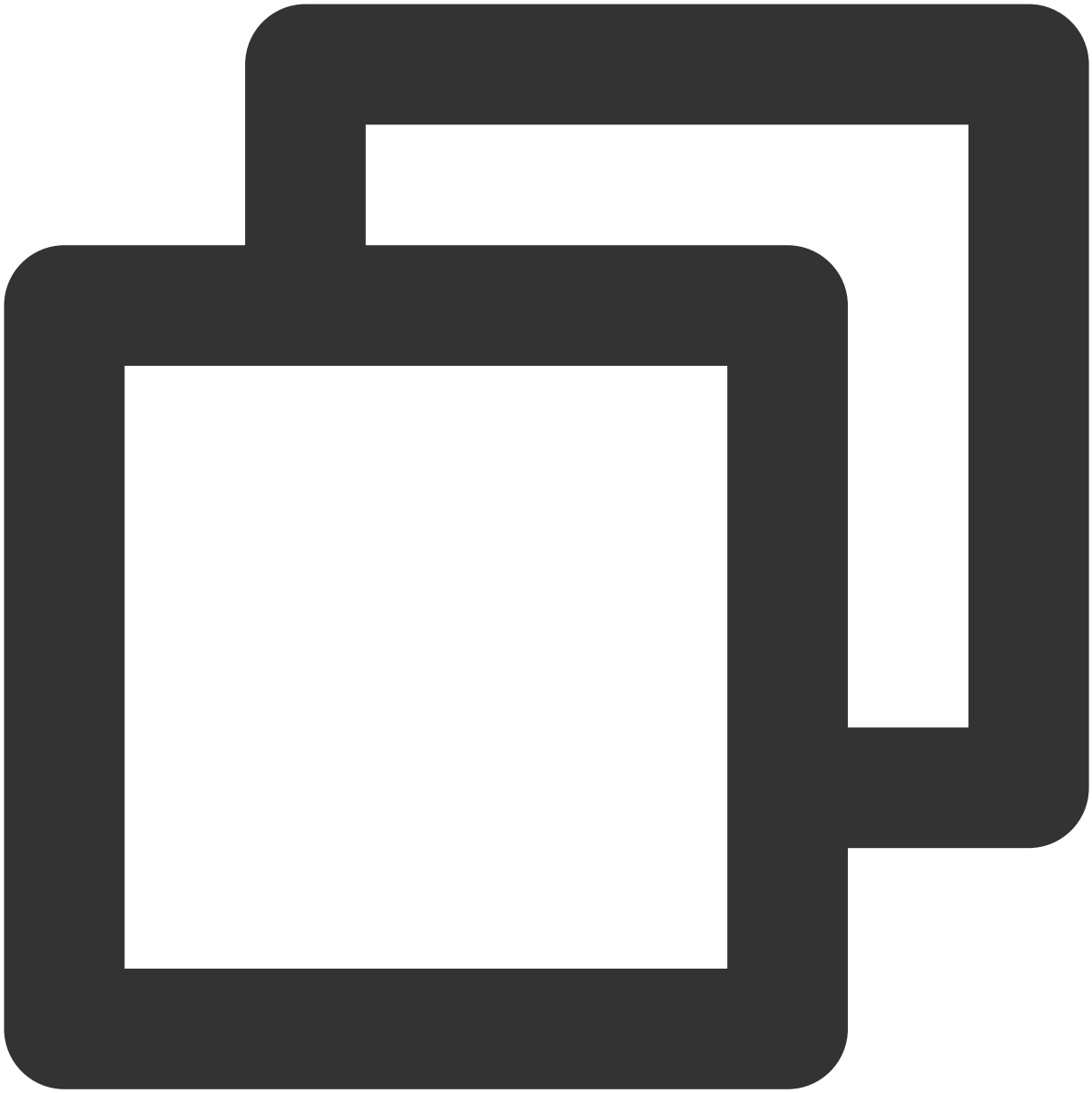
Building LVM

Note:
The following example uses three elastic cloud disks to create a dynamically resizable file system through LVM.

```
[root@VM_63_126_centos ~]# fdisk -l | grep vd | grep -v vda | grep -v vdb
Disk /dev/vdc: 10.7 GB, 10737418240 bytes
Disk /dev/vdd: 10.7 GB, 10737418240 bytes
Disk /dev/vde: 10.7 GB, 10737418240 bytes
```

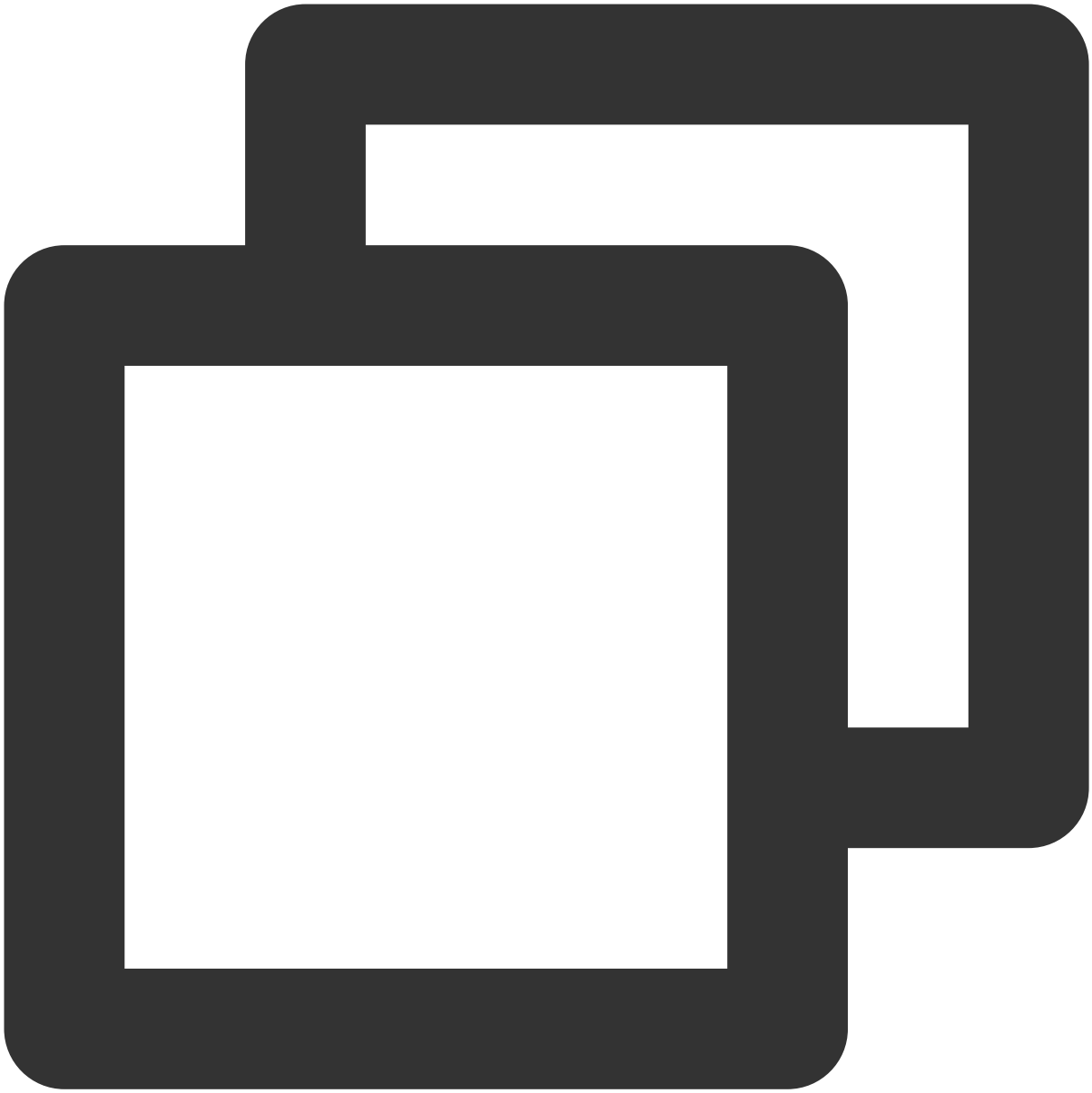
Step 1: Create a physical volume (PV)

1. Log in to the Linux instance using Web Shell as the root user.
2. Run the following command to create a PV.



```
pvcreate <disk path 1> ... <disk path N>
```

Take `/dev/vdc` , `/dev/vdd` and `/dev/vde` as an example, then run:

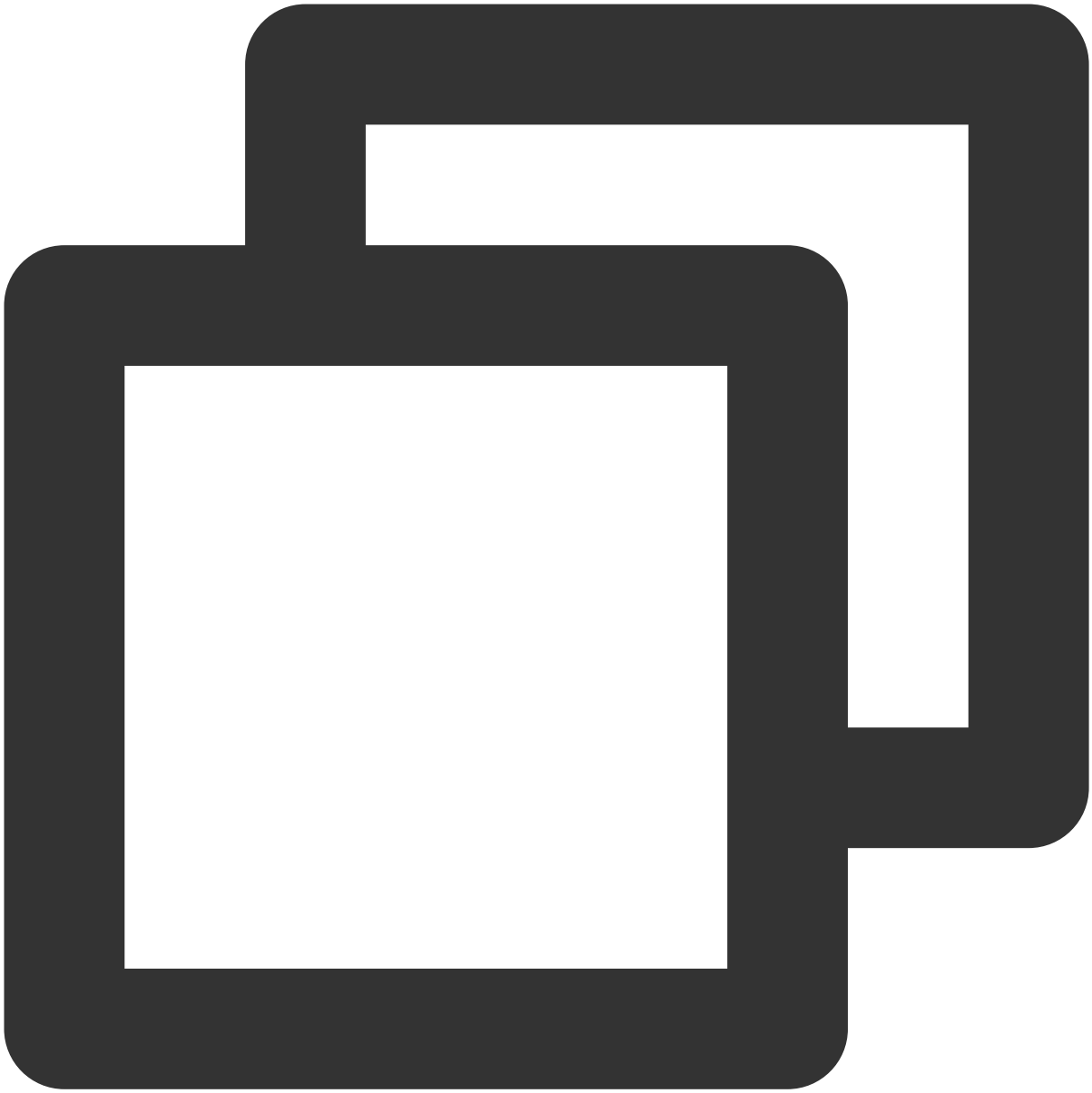


```
pvcreate /dev/vdc /dev/vdd /dev/vde
```

The following figure shows the command output when the creation is successful:

```
[root@VM_63_126_centos ~]# pvcreate /dev/vdc /dev/vdd /dev/vde
  Physical volume "/dev/vdc" successfully created
  Physical volume "/dev/vdd" successfully created
  Physical volume "/dev/vde" successfully created
```

3. Run the following command to view physical volumes of the system.

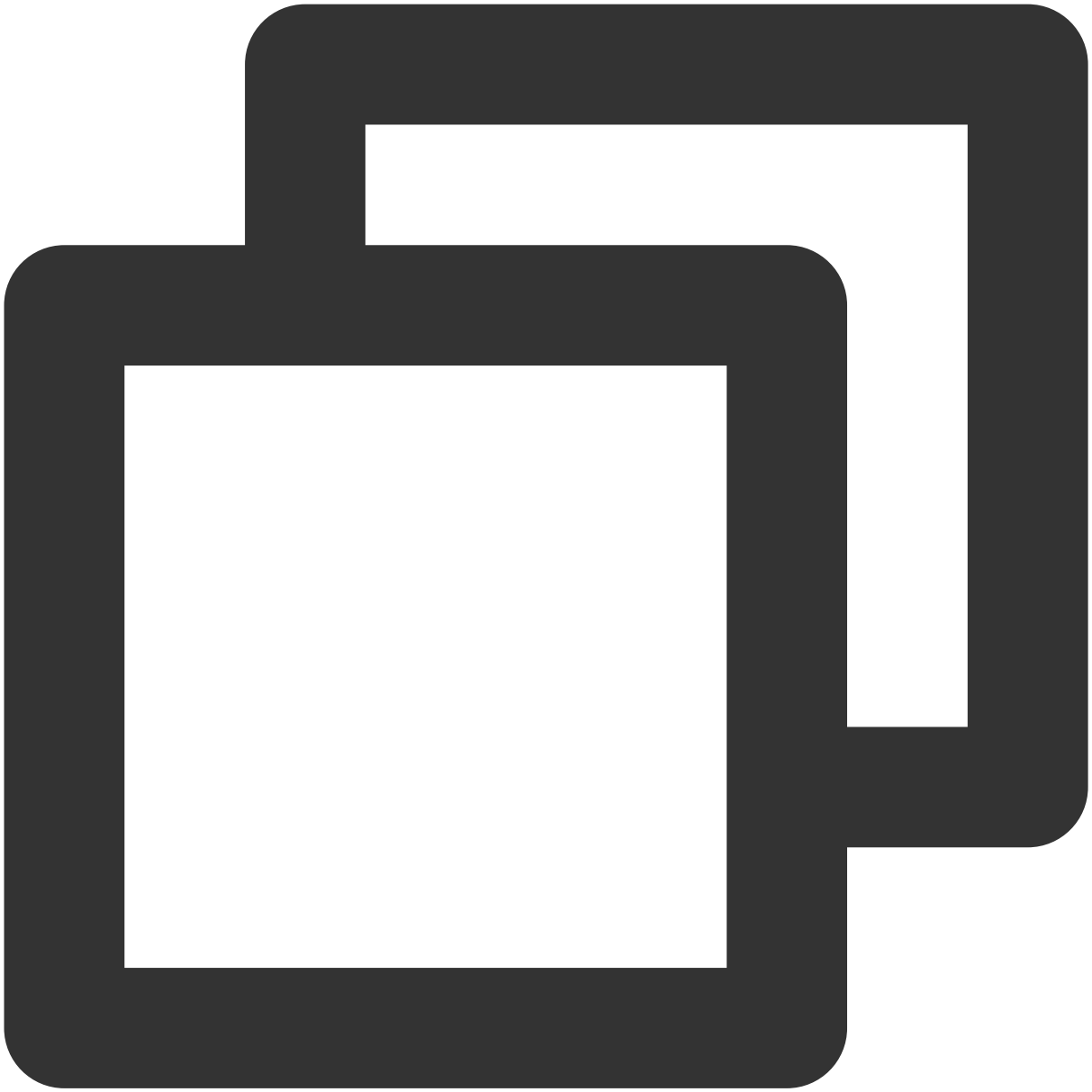


```
lvmdiskscan | grep LVM
```

```
[root@VM_63_126_centos ~]# lvmdiskscan  | grep LVM
  /dev/vdc   [      10.00 GiB] LVM physical volume
  /dev/vdd   [      10.00 GiB] LVM physical volume
  /dev/vde   [      10.00 GiB] LVM physical volume
  3 LVM physical volume whole disks
  0 LVM physical volumes
```

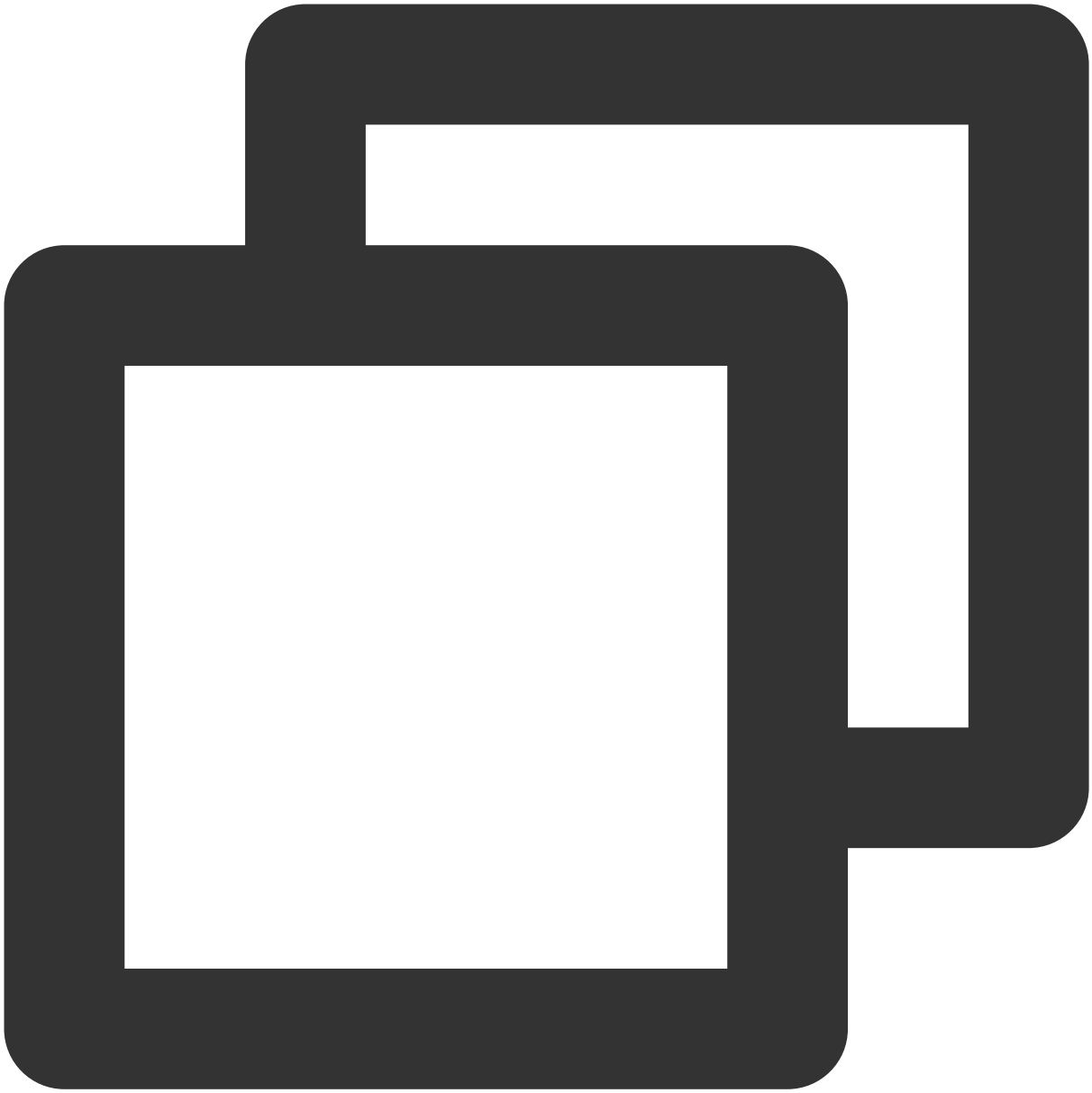
Step 2: Create a volume group (VG)

1. Run the following command to create a VG.



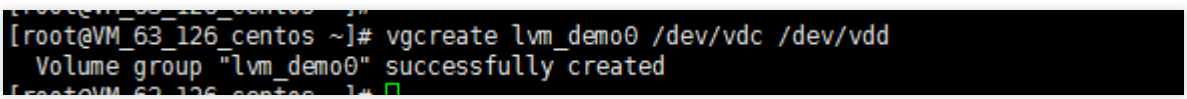
```
vgcreate [-s <specified PE size>] <VG name> <PV path>.
```

Assume you want to create a VG named "lvm_demo0", then run:



```
vgcreate lvm_demo0 /dev/vdc /dev/vdd
```

The following figure shows the command output when the creation is successful:

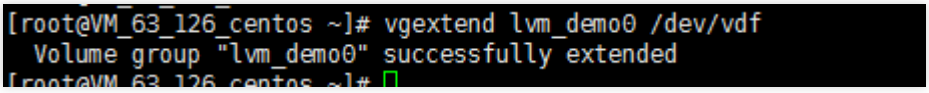


When **Volume group"** <VG name> **"successfully created** is displayed, the VG has been created.

Then you can run the following commands to add a new PV to the VG.

```
vgextend VG name New PV path
```

The following figure shows the command output when the operation is successful:

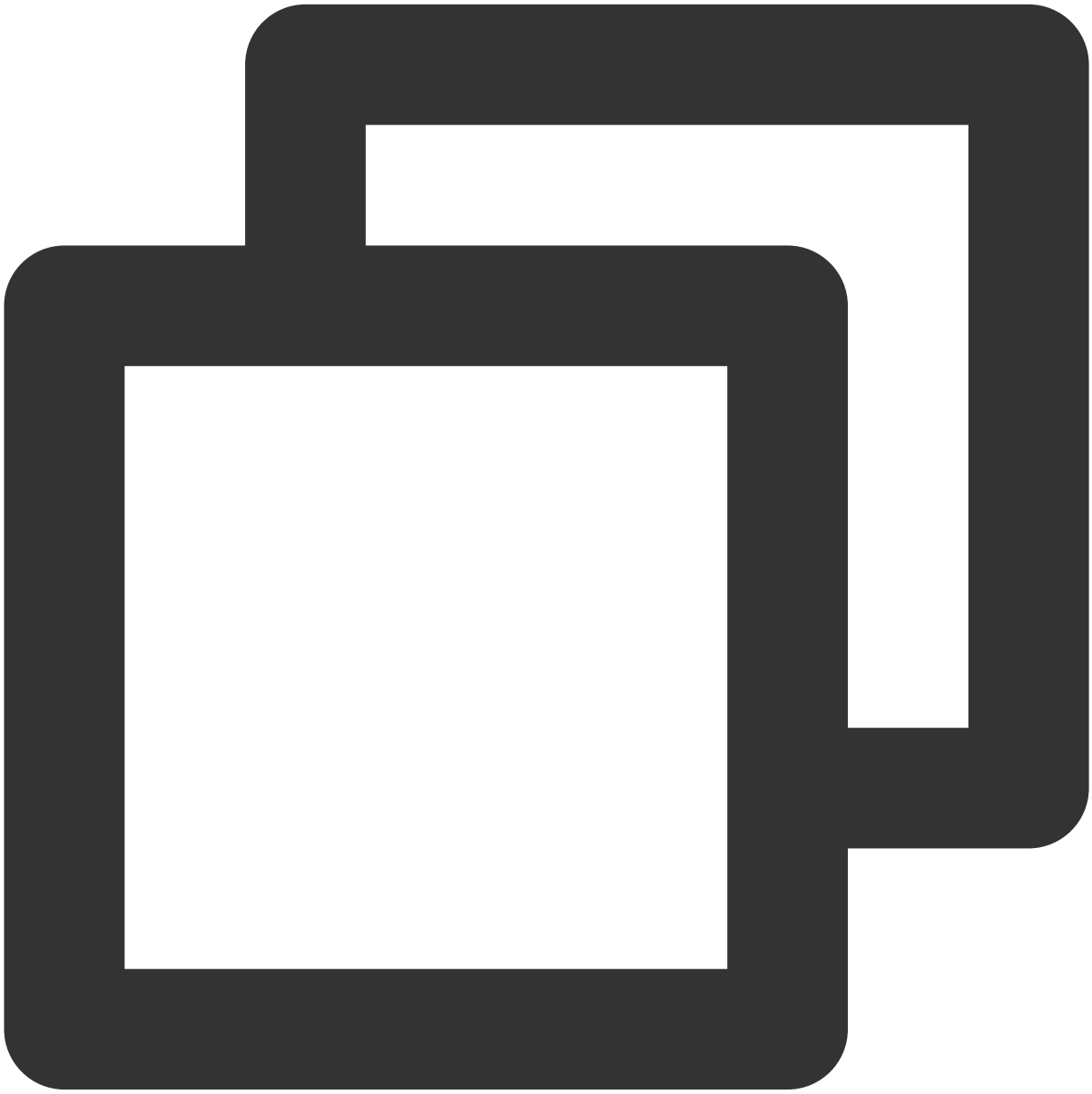


After the VG is created, you can run `vgs` , `vgdisplay` or other commands to query information about VGs of the system, as shown below:

```
[root@VM_63_126_centos ~]# vgs
  VG       #PV #LV #SN Attr   VSize  VFree
  lvm_demo   3   0   0 wz--n- 29.99g 29.99g
```

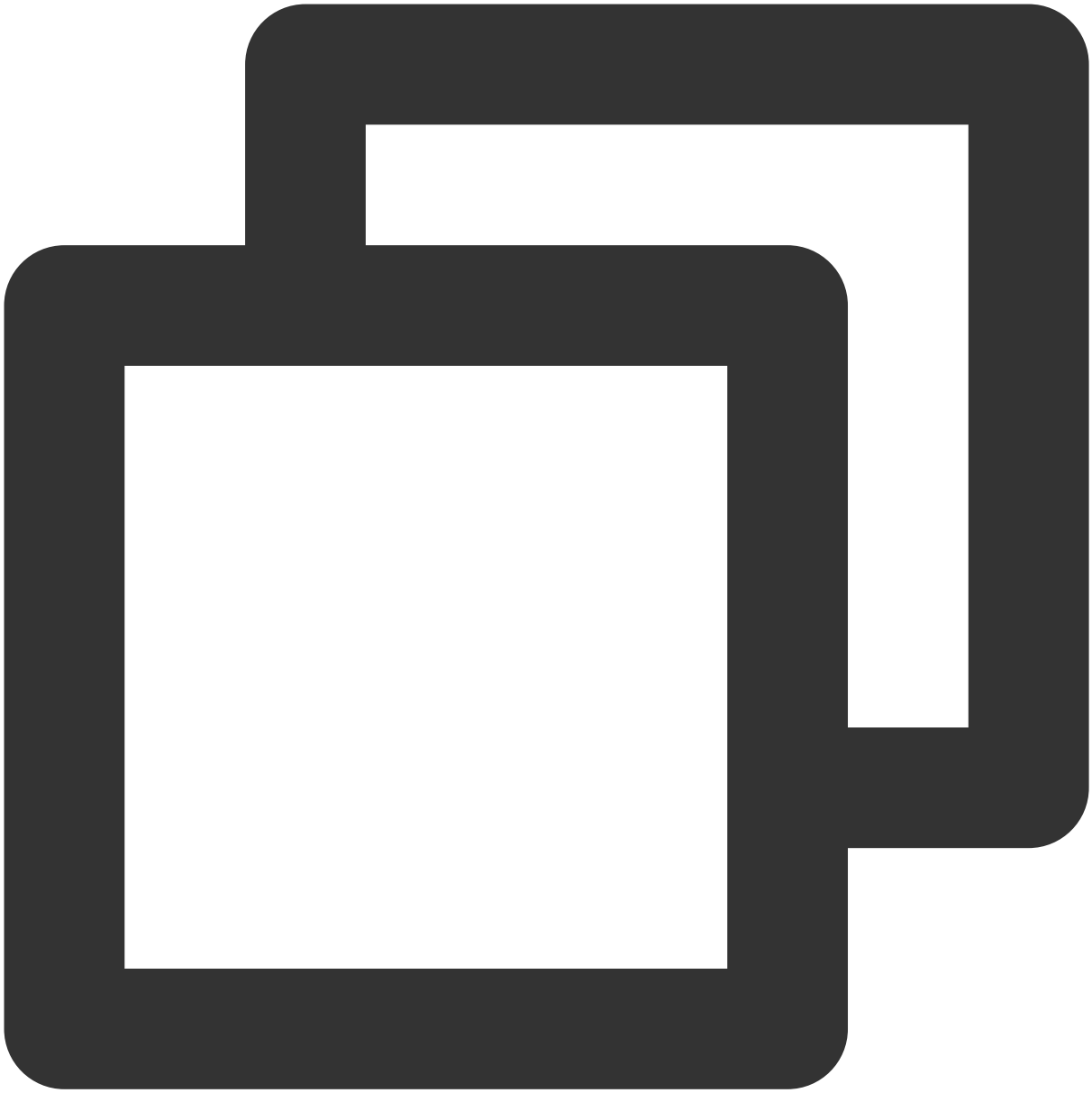
Step 3: Create a logical volume (LV)

1. Run the following command to create a LV.



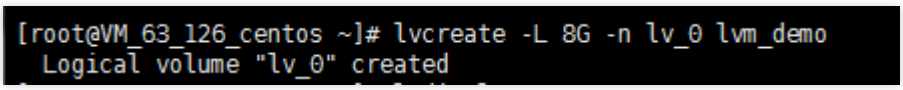
```
lvcreate [-L <LV size>][-n <LV name>] <VG name>.
```

Assume you want to create an 8 GB logical volume named "lv_0", then run



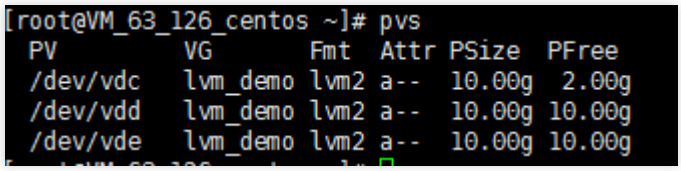
```
lvcreate -L 8G -n lv_0 lvm_demo0
```

The following figure shows the command output when the creation is successful:



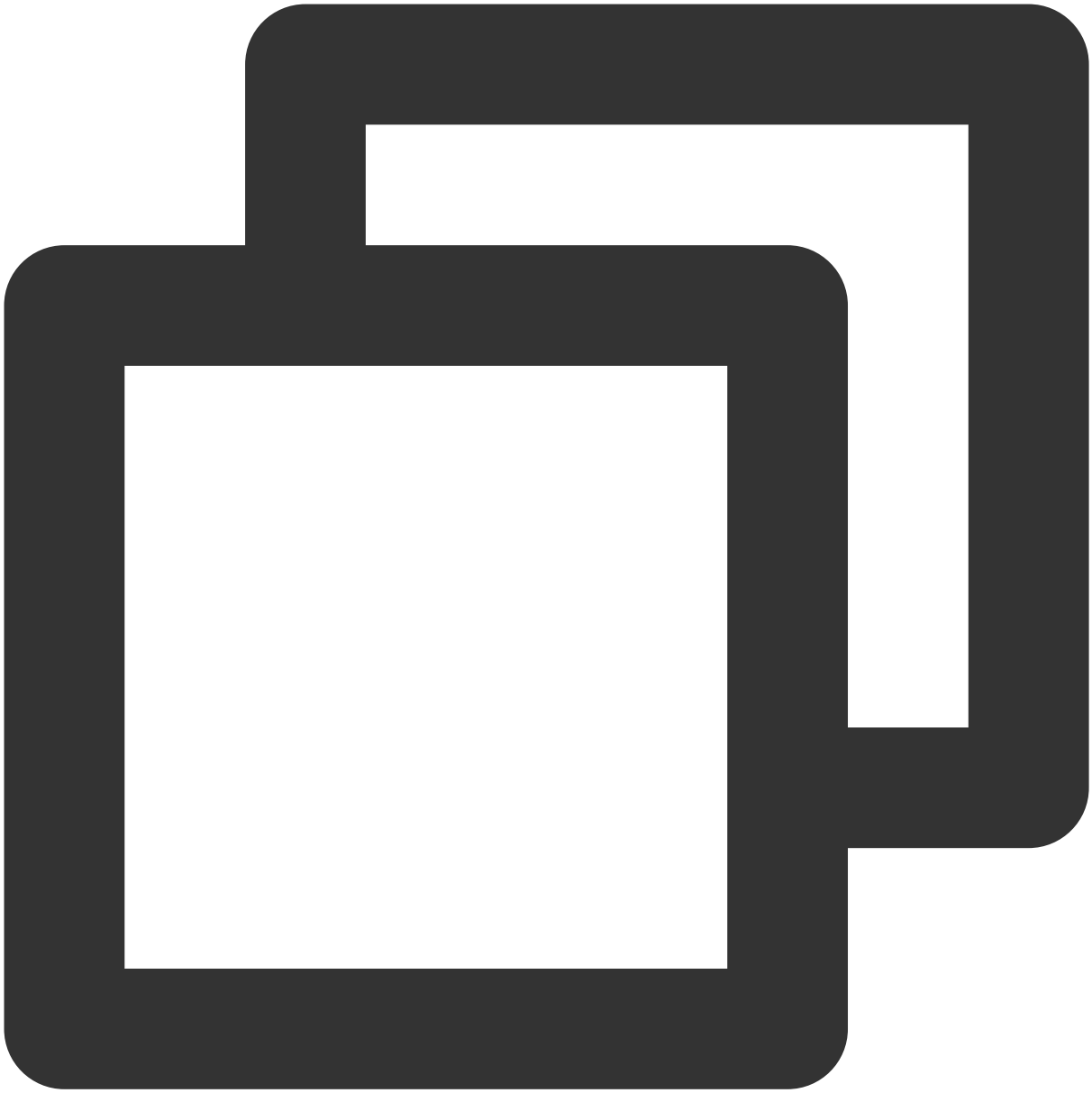
Note:

Run the `pvs` command. You can see that only the capacity of the `/dev/vdc` disk is reduced by 8 GB, as shown below:



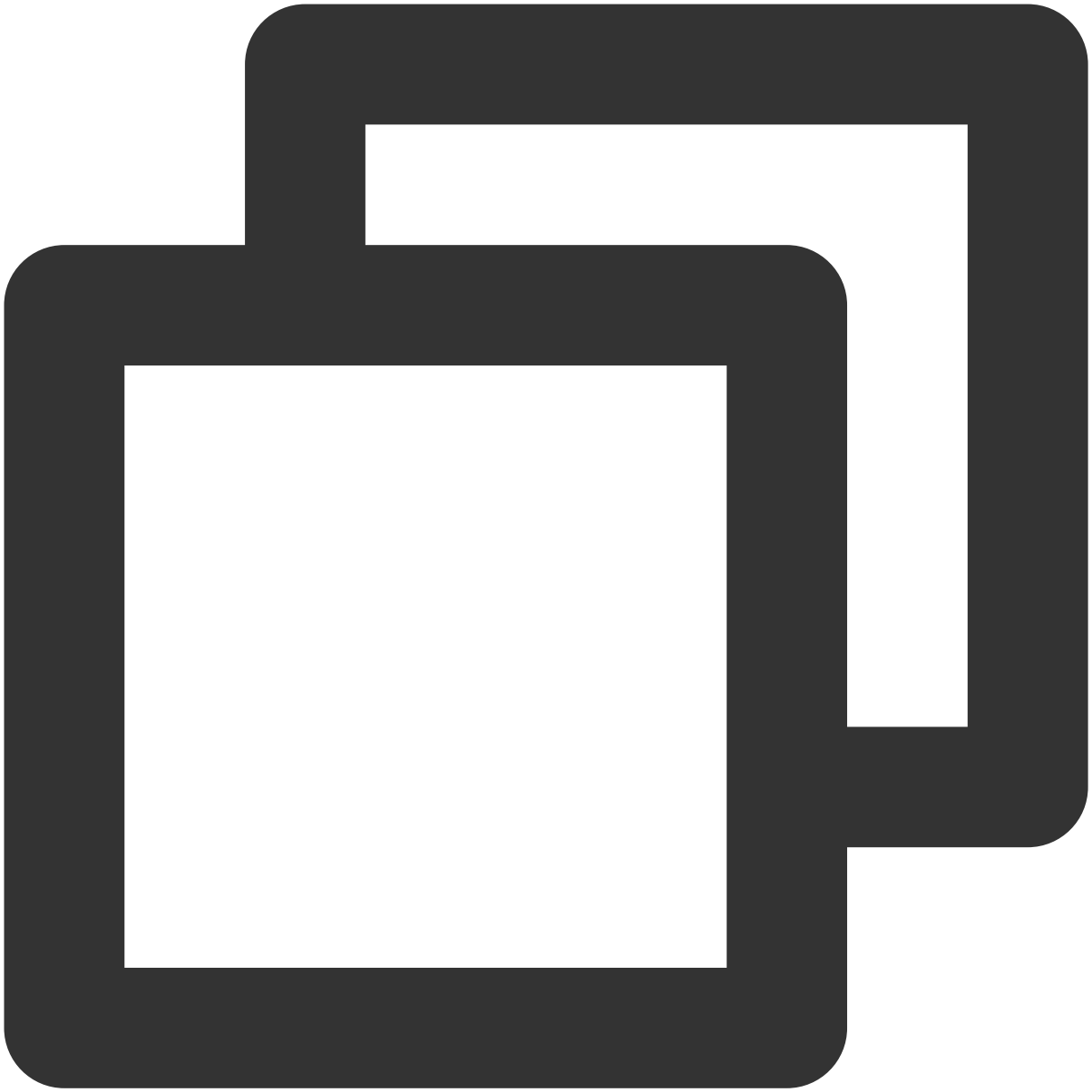
Step 4: Create and mount a file system

1. Run the following command to create a file system on an existing LV.



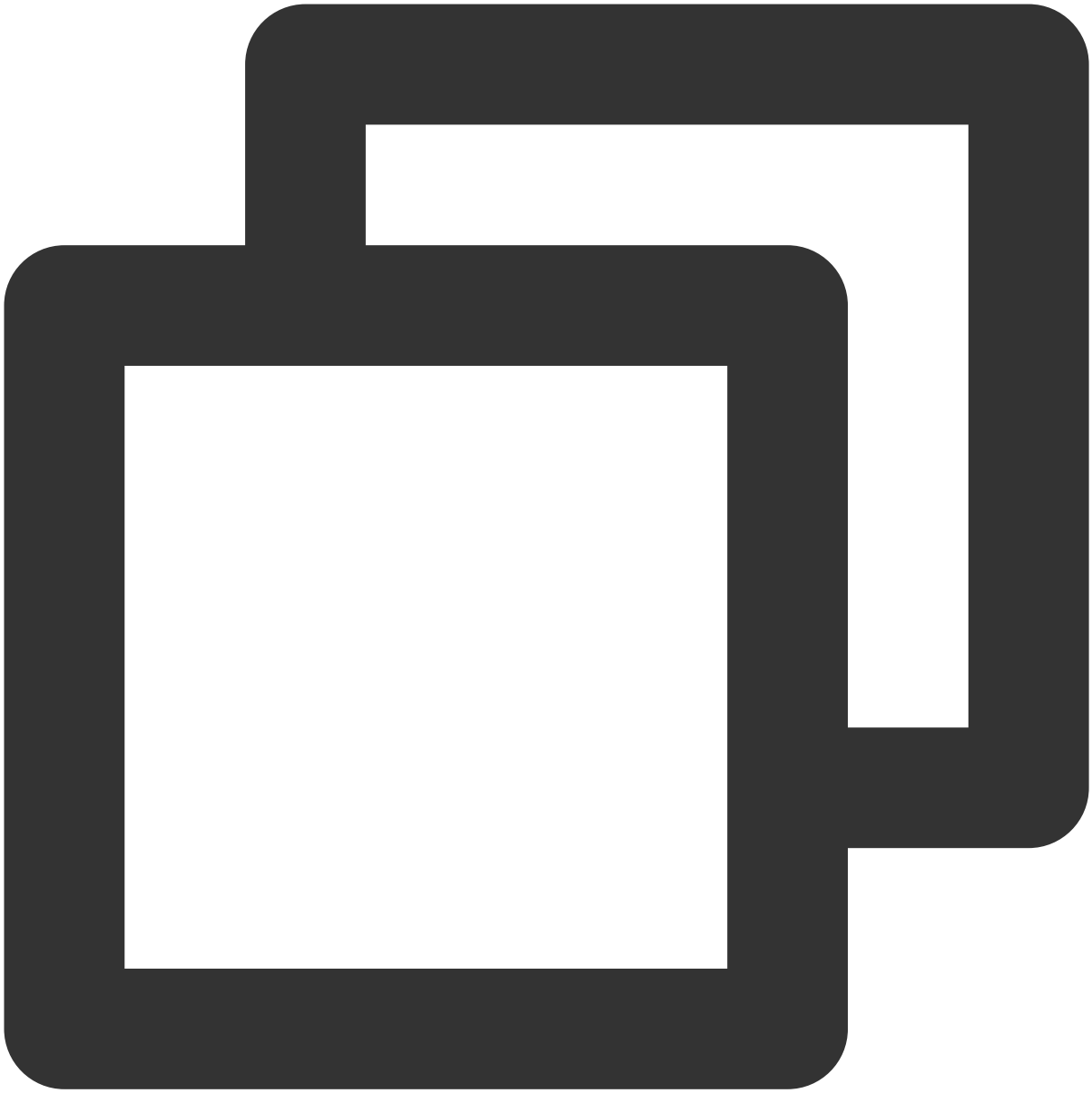
```
mkfs.ext3 /dev/lvm_demo0/lv_0
```

2. Run the following command to create the mount point `/vg0` .



```
mkdir /vg0
```

3. Run the following command to mount the file system.



```
mount /dev/lvm_demo0/lv_0 /vg0
```

The following figure shows the command output when the mount is successful:

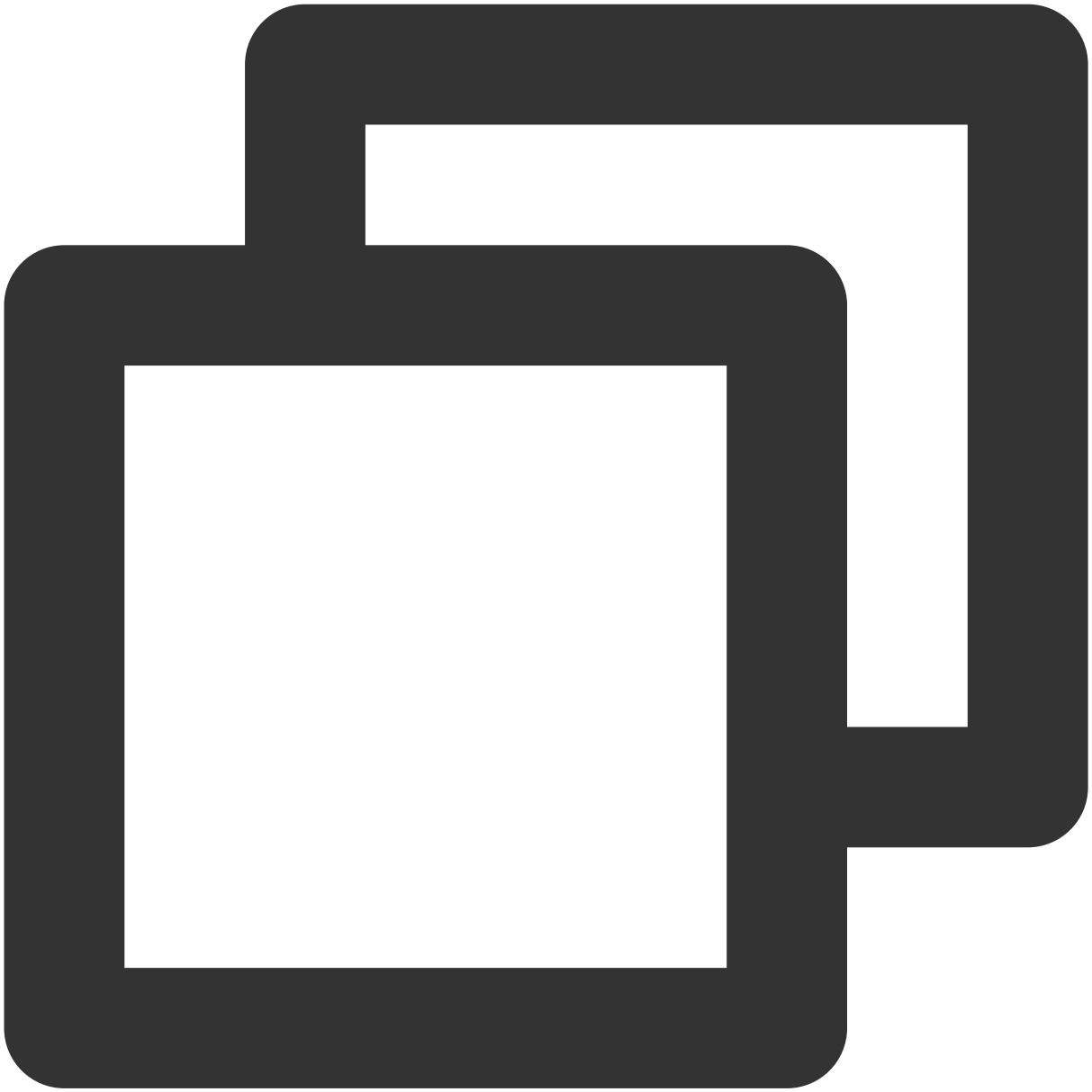
```
[root@VM_63_126_centos ~]# mount | grep lvm
/dev/mapper/lvm_demo-lv_0 on /root/vg0 type ext3 (rw)
```

Step 5: Resize the logical volume and file system dynamically

Note:

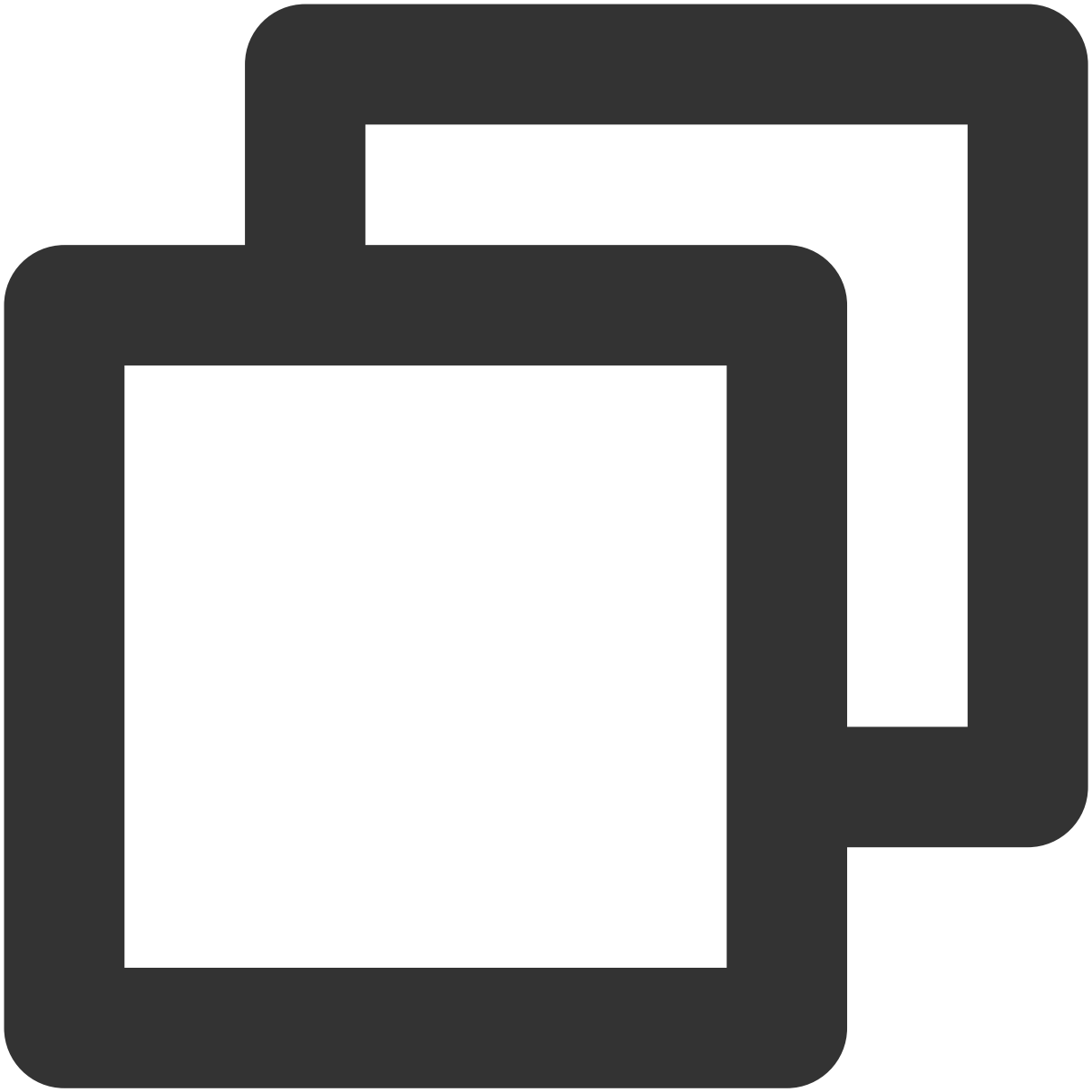
LVs can be extended dynamically only when the VG capacity is not used up. After increasing the LV capacity, you need to extend the file system created on this LV.

1. Run the following command to extend the LV.



```
lvextend [-L +/- <scale capacity>] <LV path>
```

Assume you want to scale up the capacity of LV named "lv_0" by 4 GB, then run



```
lvextend -L +4G /dev/lvm_demo0/lv_0
```

The following figure shows the command output when the scaling is successful:

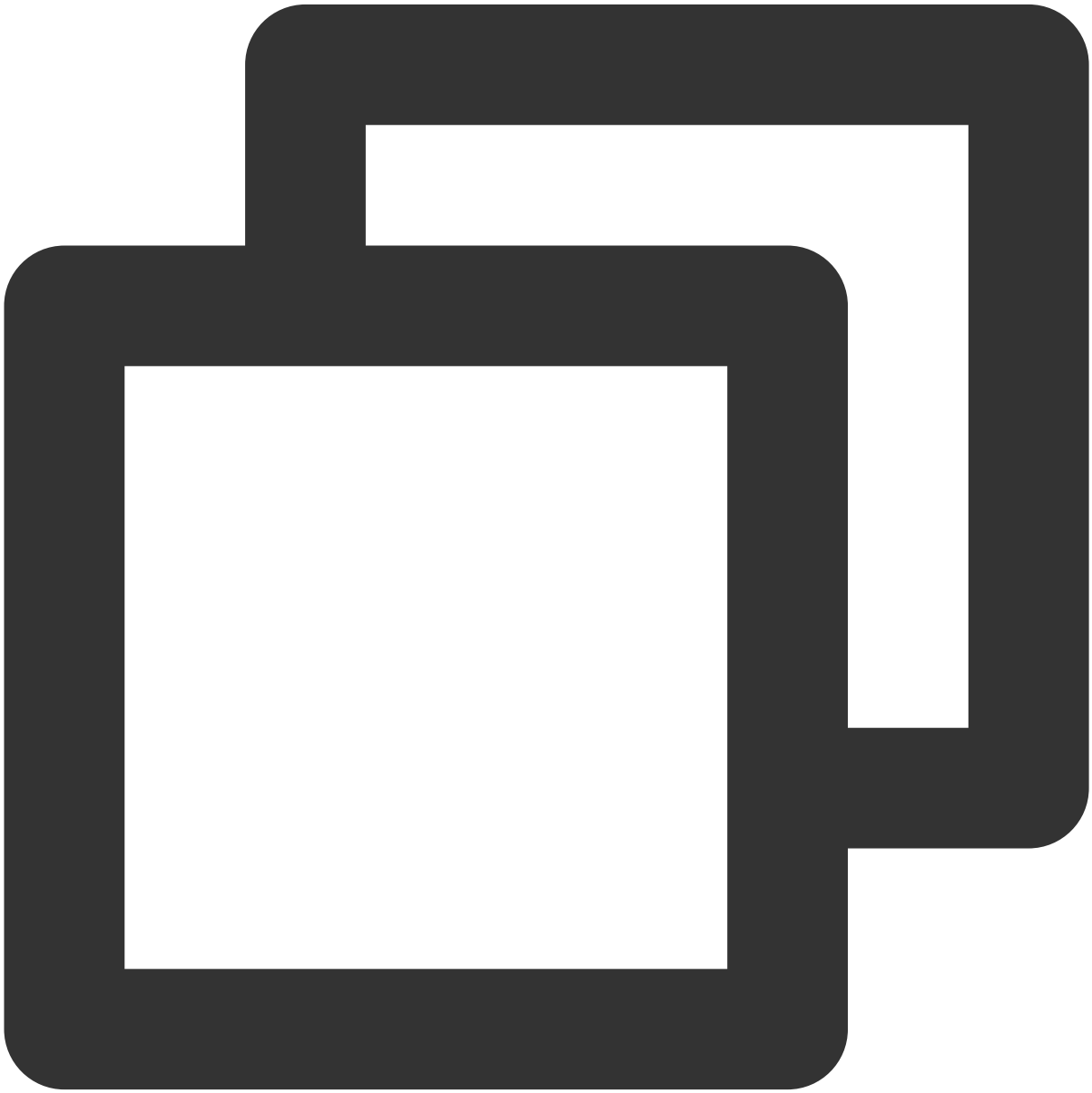
```
[root@VM_63_126_centos vg0]# lvextend -L +4G /dev/lvm_demo/lv_0
  Size of logical volume lvm_demo/lv_0 changed from 8.00 GiB (2048 extents) to 12.00 GiB (3072 e
  Logical volume lv_0 successfully resized
```

Note:

Run the `pvs` command. You can find that the `/dev/vdc` disk capacity has been fully used, and the `/dev/vdd` disk capacity has been used by 2 GB, as shown below:

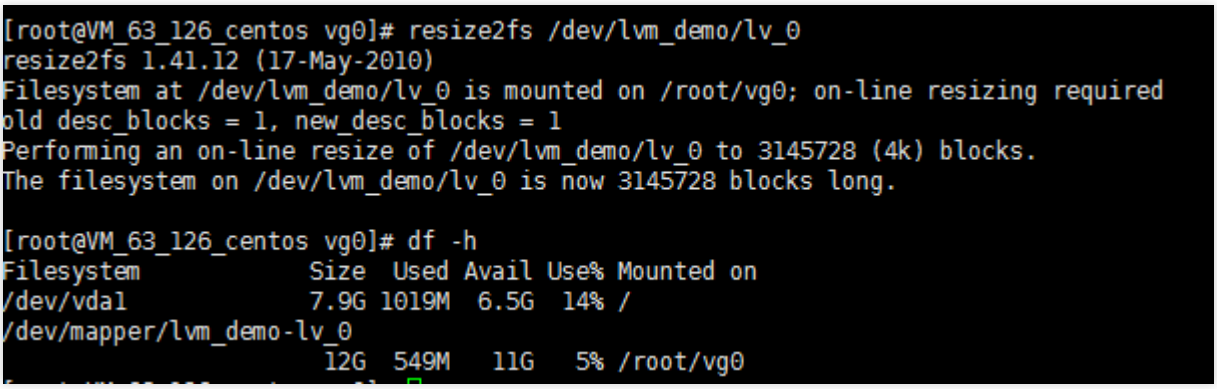
```
[root@VM_63_126_centos vg0]# pvs
  PV         VG       Fmt  Attr PSize  PFree
  /dev/vdc   lvm_demo lvm2 a--  10.00g     0
  /dev/vdd   lvm_demo lvm2 a--  10.00g  7.99g
  /dev/vde   lvm_demo lvm2 a--  10.00g 10.00g
```

2. Run the following command to extend the file system.

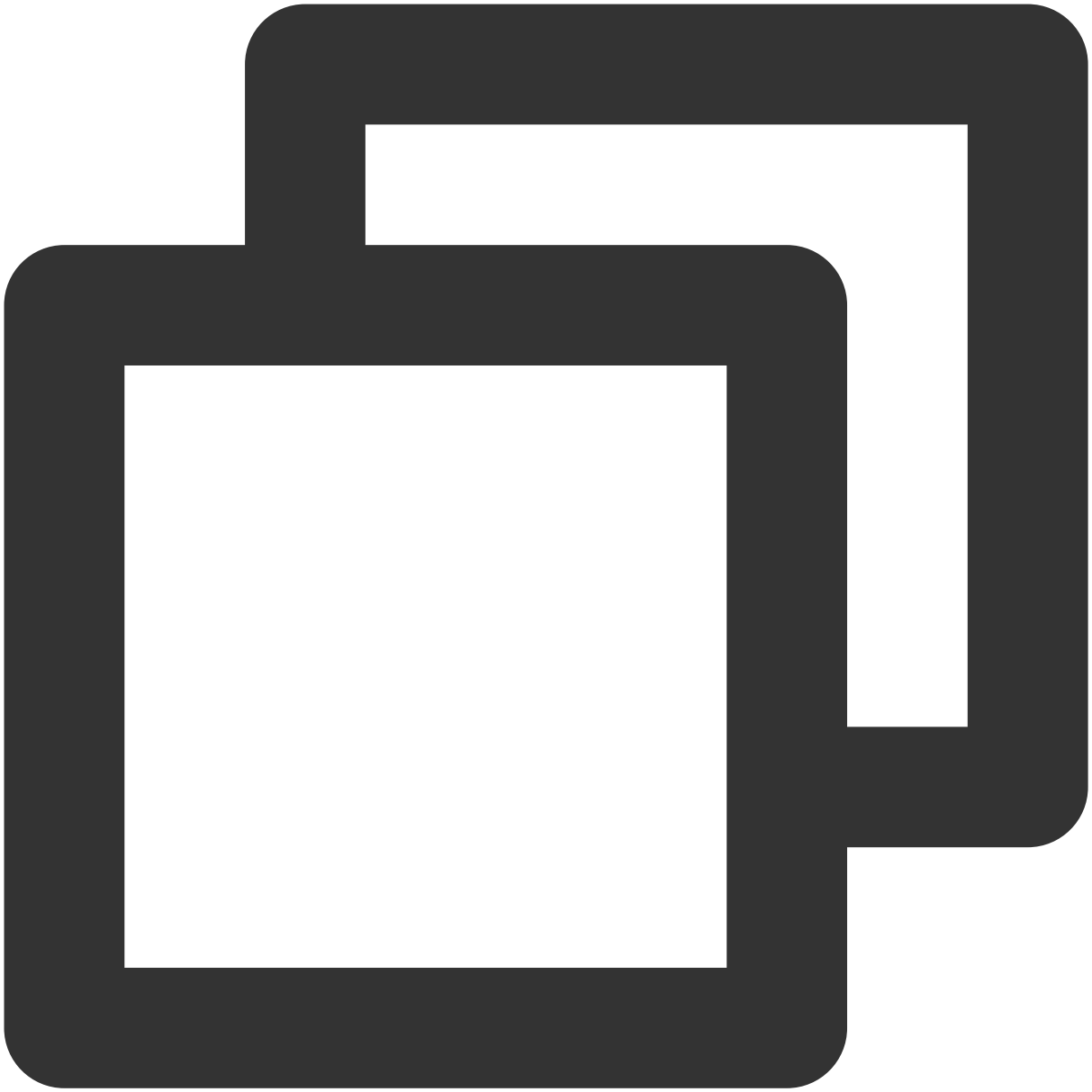


```
resize2fs /dev/lvm_demo0/lv_0
```

The following figure shows the command output when the scaling is successful:



After the extension, run the following command to check whether the LV capacity is 12 GB.



```
df -h
```

Expanding MBR Cloud Disks to Greater Than 2 TB

Last updated：2023-12-22 10:51:35

Overview

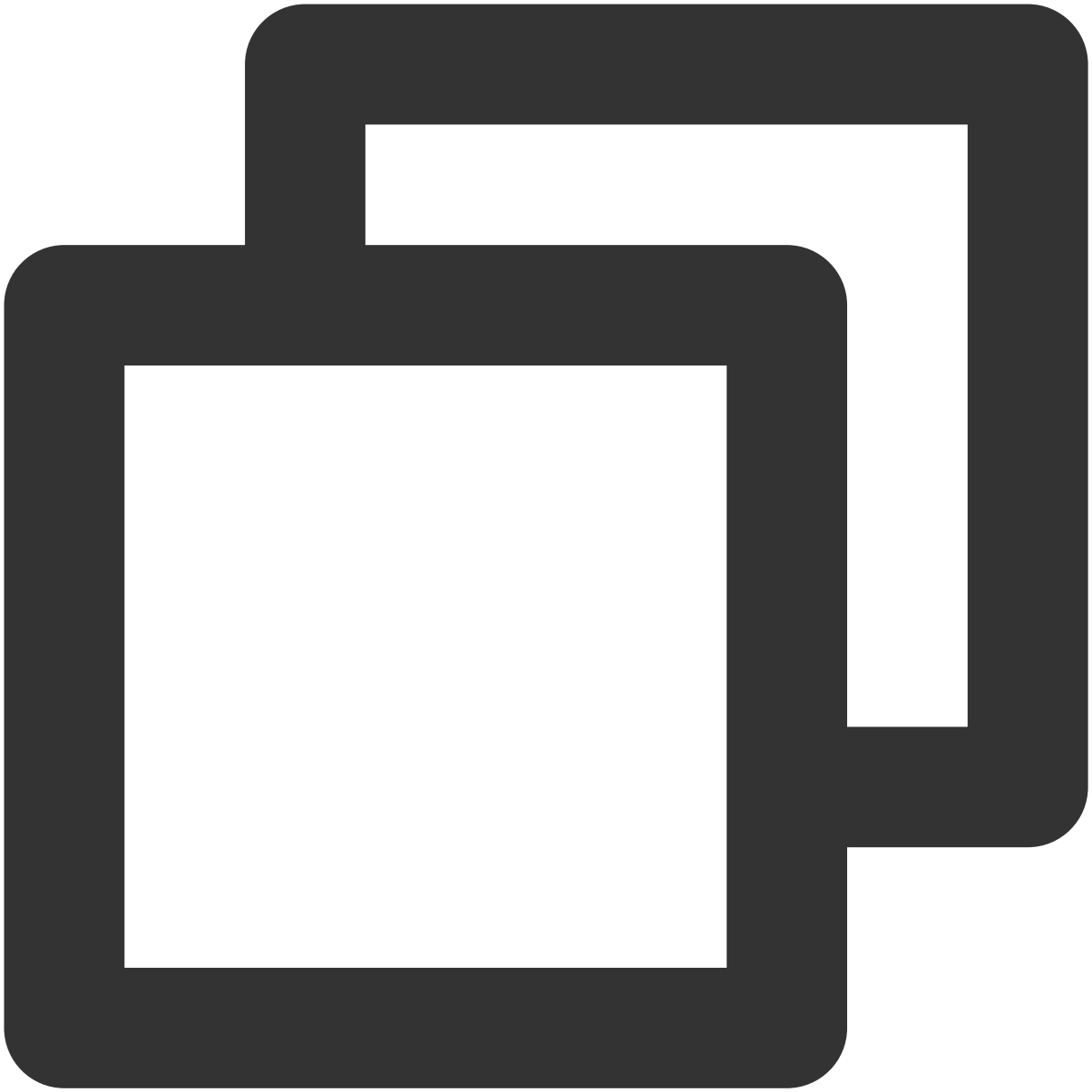
When your cloud disk has an MBR partition with a created file system and has been expanded to greater than 2 TB, the file system cannot be expanded to greater than 2 TB. This document describes how to convert the MBR partition to the GPT partition to implement the expansion.

Notes

To convert the partition format, you need to replace the original partition. The original partition data will not be deleted in normal cases. However, as the original partition needs to be unmounted, online businesses will be affected. Maloperation may cause data losses or exceptions. Proceed with caution. Create a snapshot of the cloud disk for data backup. For detailed directions, see Creating Snapshots. If data is lost due to maloperation, you can roll back the data for restoration.

Directions

1. Log in to a Linux instance using standard login method.
2. Run the following command to check whether the partition format is MBR.

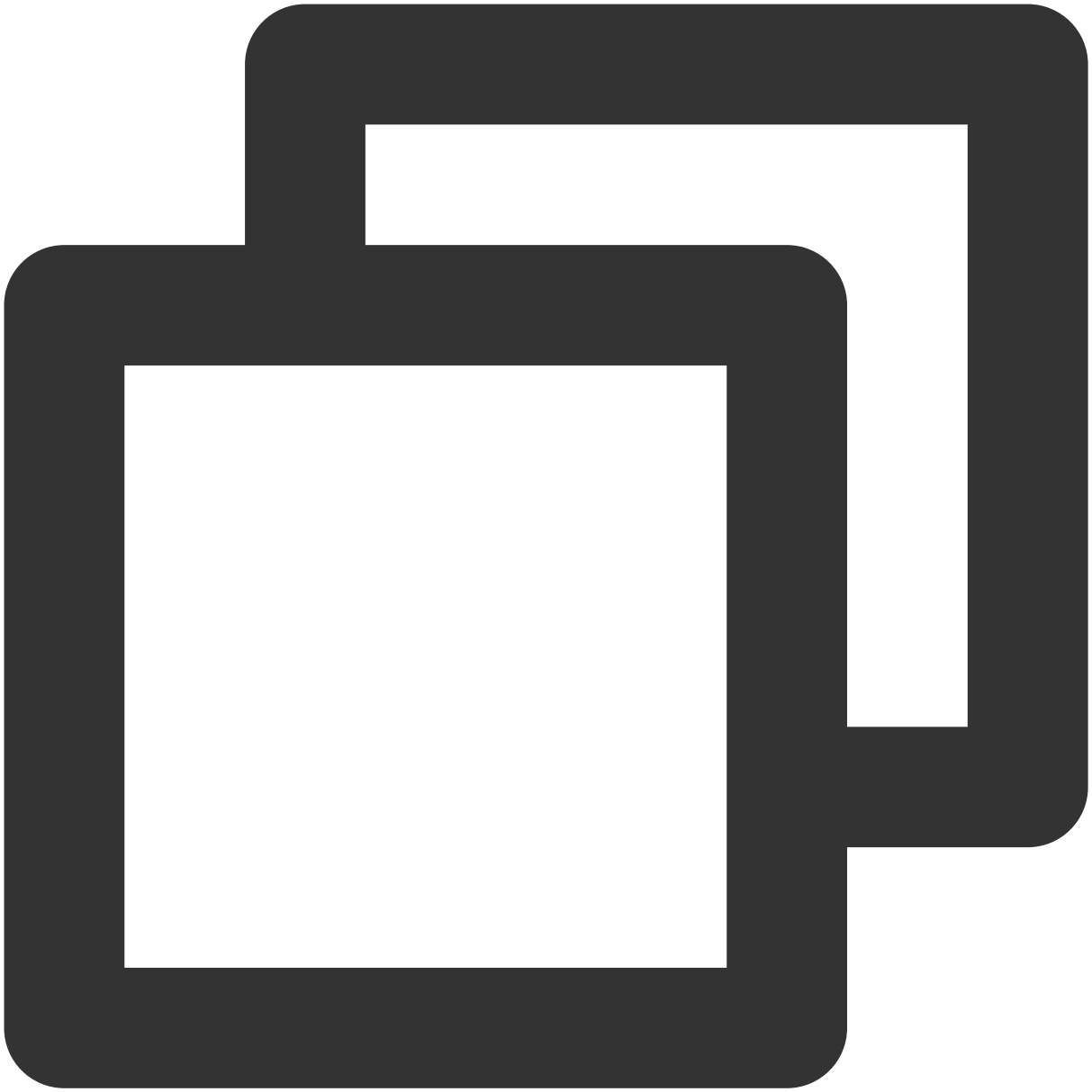


```
fdisk -l
```

If the following result is shown (which may vary by operation system), the partition format is MBR.

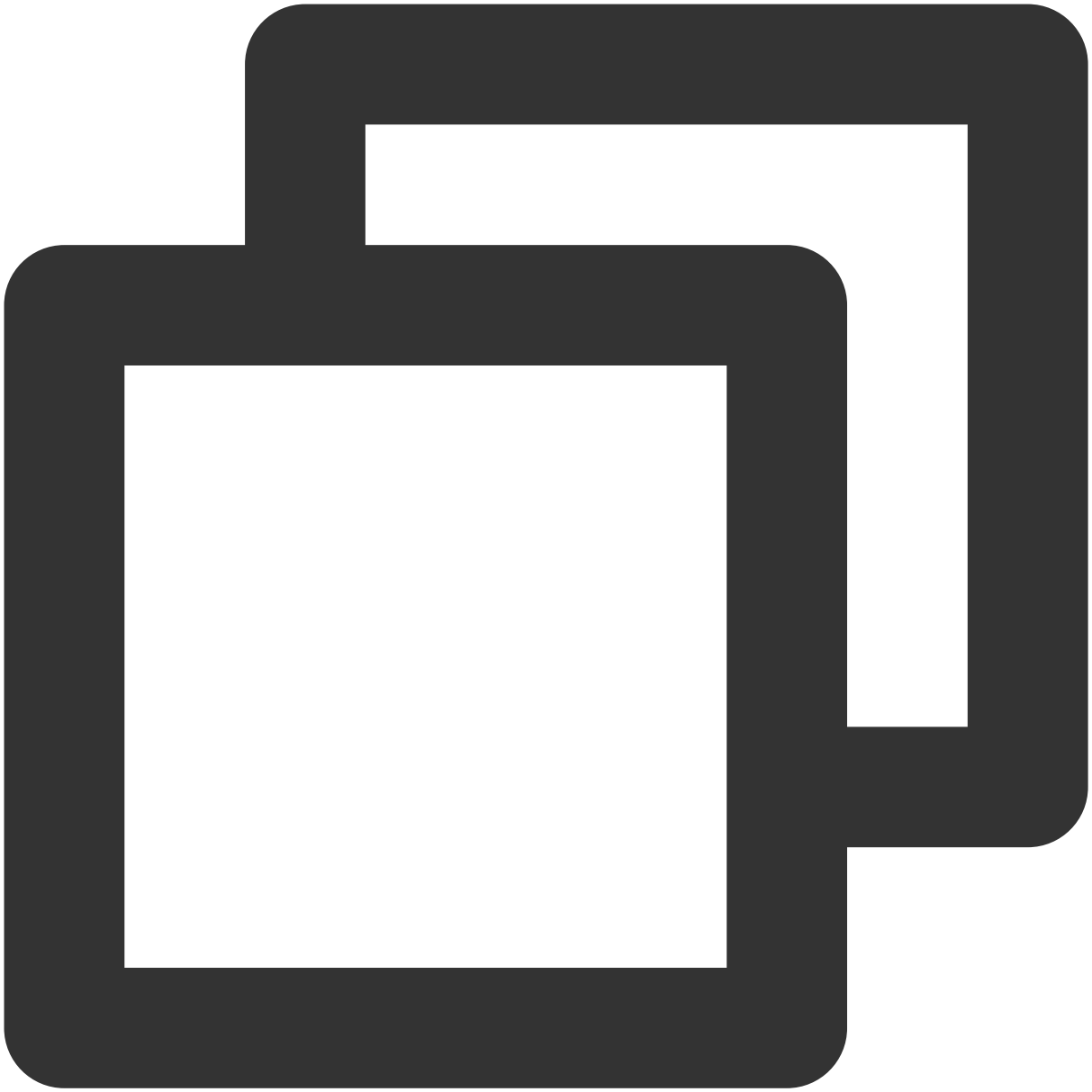
```
[root@VM-0-3-centos ~]# fdisk -l
Disk /dev/vda: 50 GiB, 53687091200 bytes, 104857600 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0xe609e297

Device     Boot Start       End   Sectors Size Id Type
/dev/vda1        2048 104857566 104855519  50G 83 Linux


Disk /dev/vdb: 2 TiB, 2147483648000 bytes, 4194304000 sectors
Units: sectors of 1 * 512 = 512 bytes
Sector size (logical/physical): 512 bytes / 512 bytes
I/O size (minimum/optimal): 512 bytes / 512 bytes
Disklabel type: dos
Disk identifier: 0x048787f2

Device     Boot Start       End   Sectors Size Id Type
/dev/vdb1        2048 104857599 104855552  50G 83 Linux
```

3. Run the following command to unmount the partition.



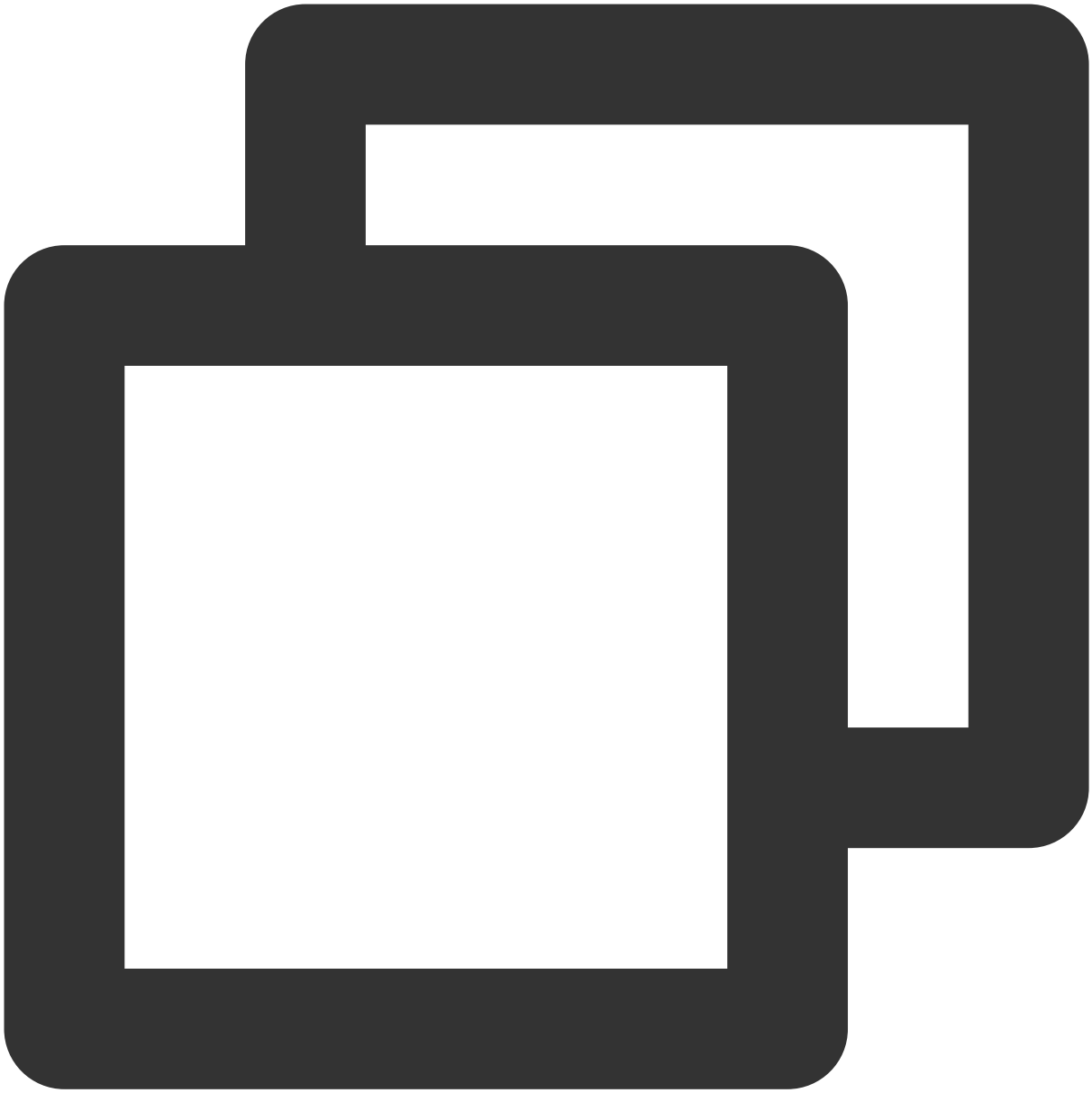
```
umount <Mount point>
```

Taking the `/data` mount point as an example, run the following command:



```
umount /data
```

4. Run the following command to view the unmount result.

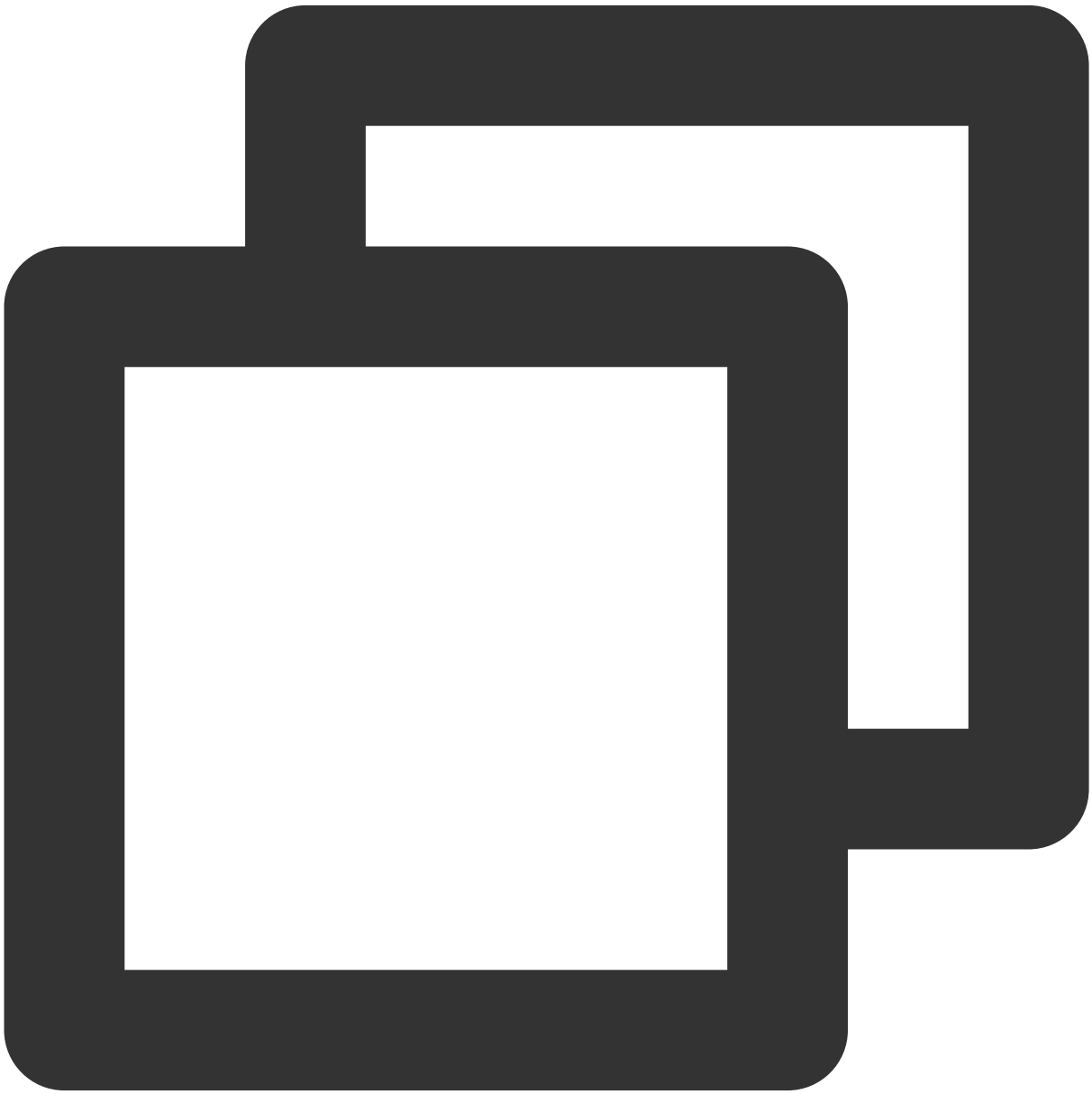


```
lsblk
```

If the `MOUNTPOINT` of the original partition is empty, the unmount is successful. This document takes the `/dev/vdb1` partition as an example. Below is the returned result.

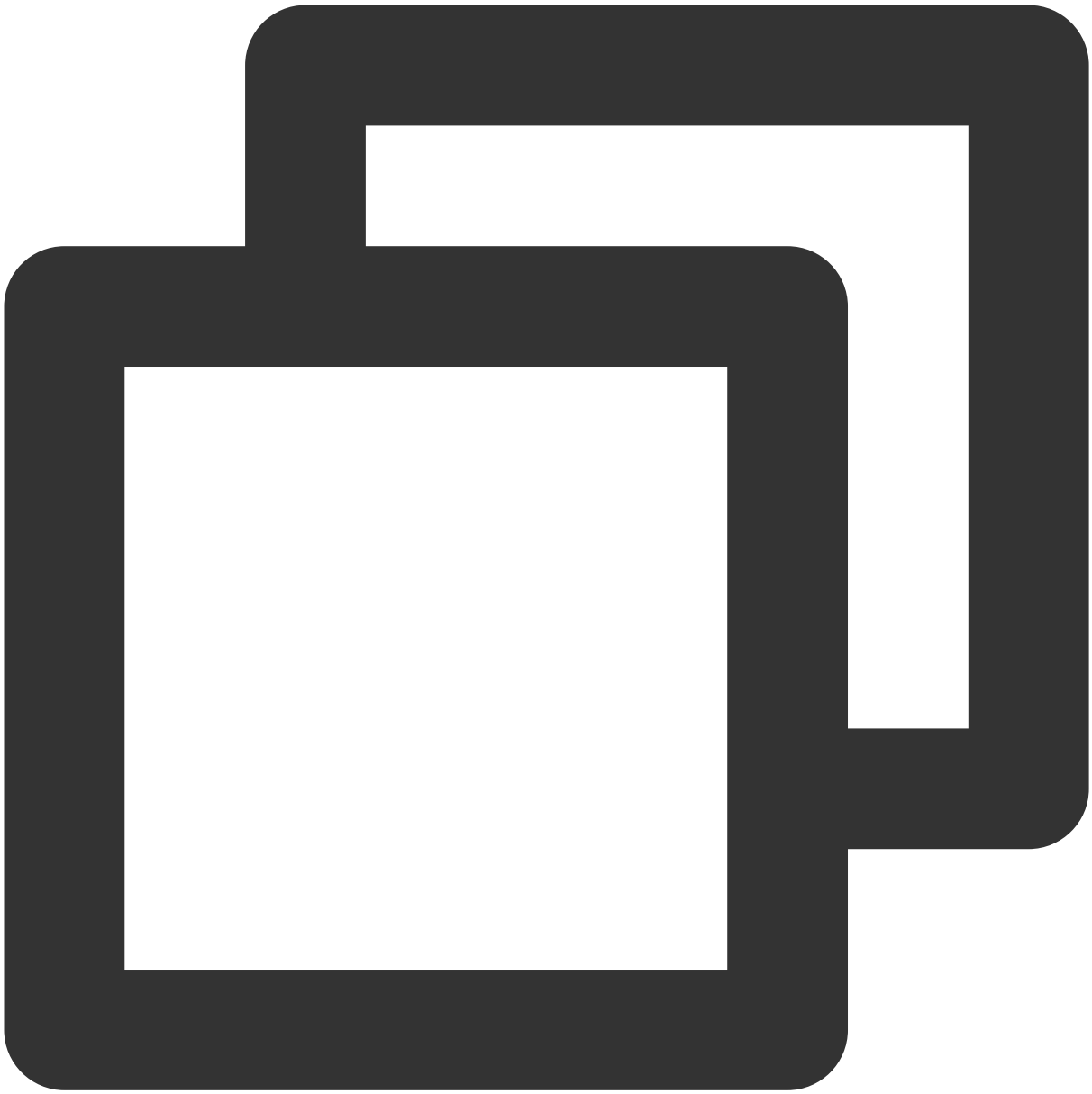
```
[root@VM-0-3-centos ~]# lsblk
NAME    MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sr0      11:0    1 184.1M  0 rom
vda     253:0    0    50G  0 disk
└─vda1  253:1    0    50G  0 part /
vdb     253:16   0     2T  0 disk
└─vdb1  253:17   0    50G  0 part
```

5. Run the following command to use the parted partition tool.



```
parted <Disk path>
```

Taking the disk path `/dev/vdb` as an example, run the following command:



```
parted /dev/vdb
```

6. Enter `p` and press **Enter** to view the current partition information. Below is the returned information:

```
[root@VM-5-94-centos ~]# parted /dev/vdb
GNU Parted 3.2
Using /dev/vdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) p
Model: Virtio Block Device (virtblk)
Disk /dev/vdb: 2201GB
Sector size (logical/physical): 512B/512B
Partition Table: msdos
Disk Flags:

Number  Start   End     Size    Type     File system  Flags
 1      1049kB  53.7GB  53.7GB  primary  ext4
```

7. Enter `rm partition number` and press **Enter** to delete the last partition to be replaced.

In this example, there is only one partition, so you can enter `rm 1` and press **Enter** to delete partition 1.

8. Enter `p` and press **Enter** to view the current partition information. Check whether the last partition has been deleted.

9. Enter `mklabel GPT` and press **Enter** to create a partition in GPT format.

10. Enter `Yes` and press **Enter**.

```
(parted) rm 1
(parted) mklabel GPT
Warning: The existing disk label on /dev/vdb will be destroyed and all data on this disk
Yes/No? Yes
```

11. Enter `mkpart primary 2048s 100%` and press **Enter** to create the partition.

Here, `2048s` indicates the initial disk capacity, and `100%` indicates the maximum disk capacity. This is for reference only. You can choose the number of disk partitions and their capacities based on your business needs.

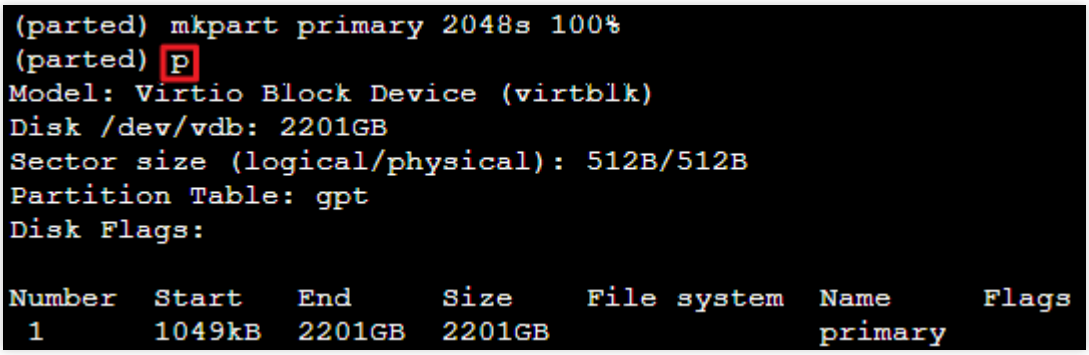
Note:

Data may be lost in the following cases:

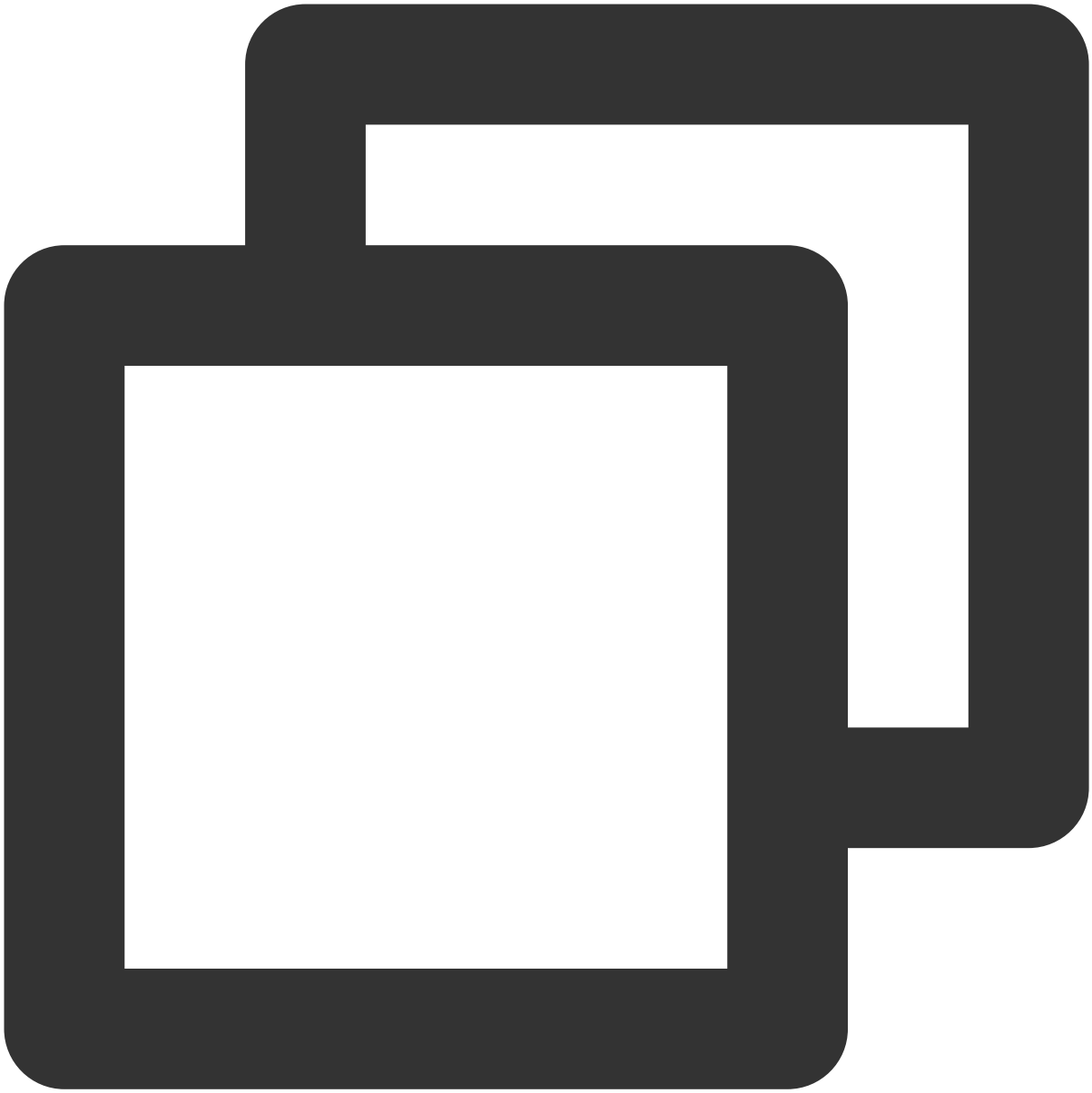
The configured initial capacity differs from the original partition capacity.

The configured maximum capacity is smaller than the original partition capacity before the expansion.

12. Enter `p` and press **Enter** to check whether the partition has been replaced successfully. If the following result is shown, the replacement is successful:

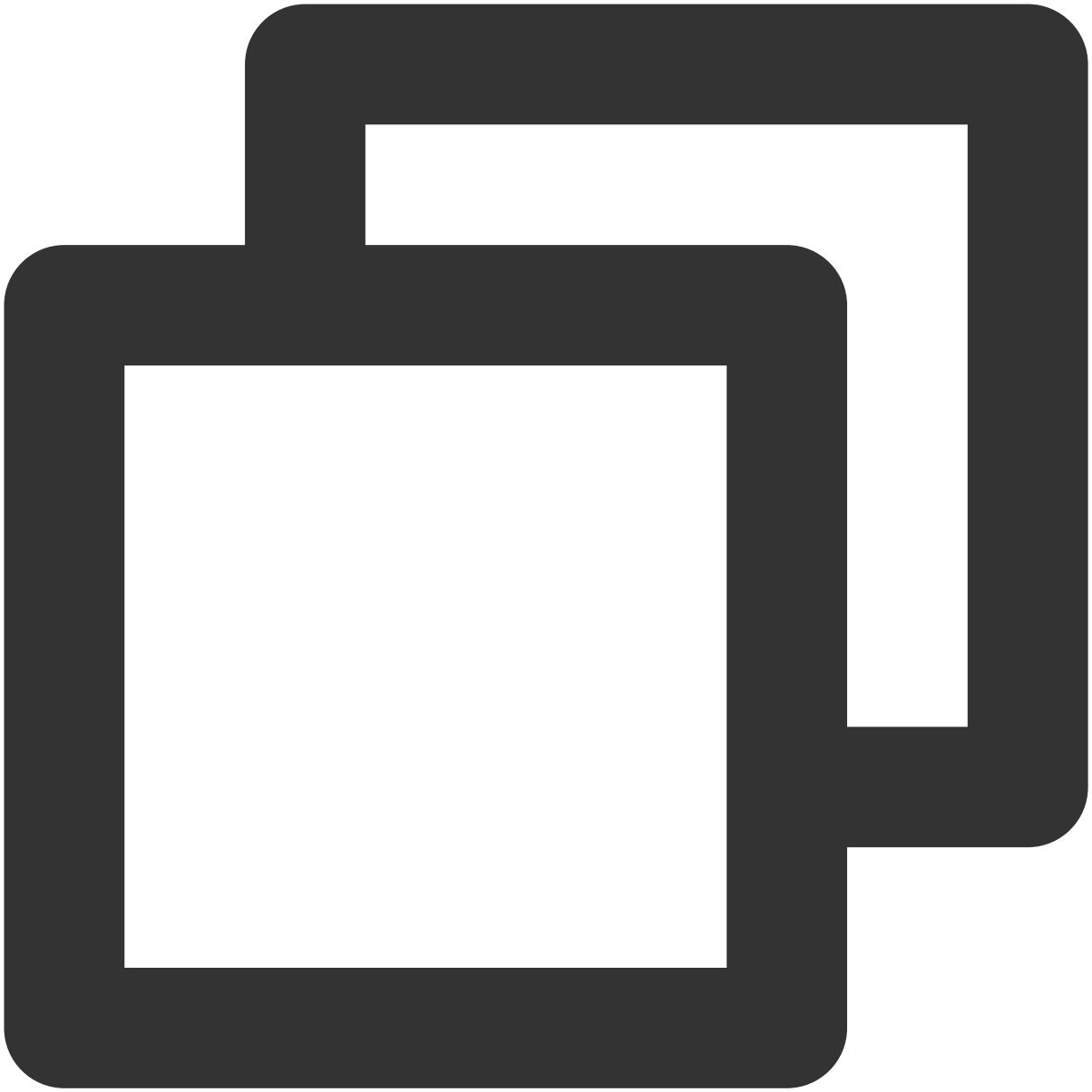


13. Enter `q` and press **Enter** to exit the parted partition tool.
14. Run the following command to mount the partition.



```
mount <Partition path> <Mount point>
```

Taking the `/dev/vdb1` partition path and `/data` mount point as an example, run the following command:



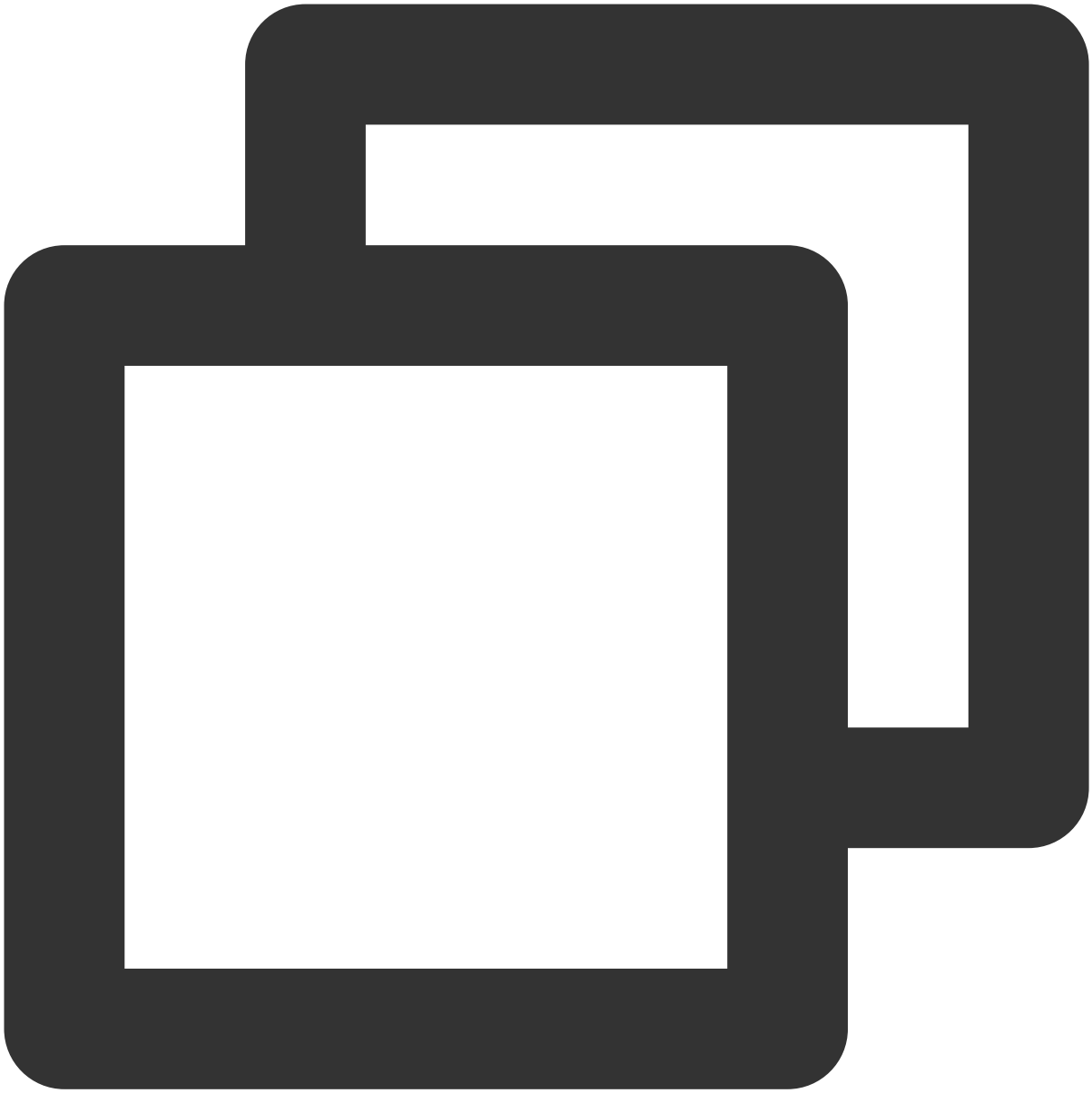
```
mount /dev/vdb1 /data
```

15. Run the command to extend the file system.

Extending the EXT file system

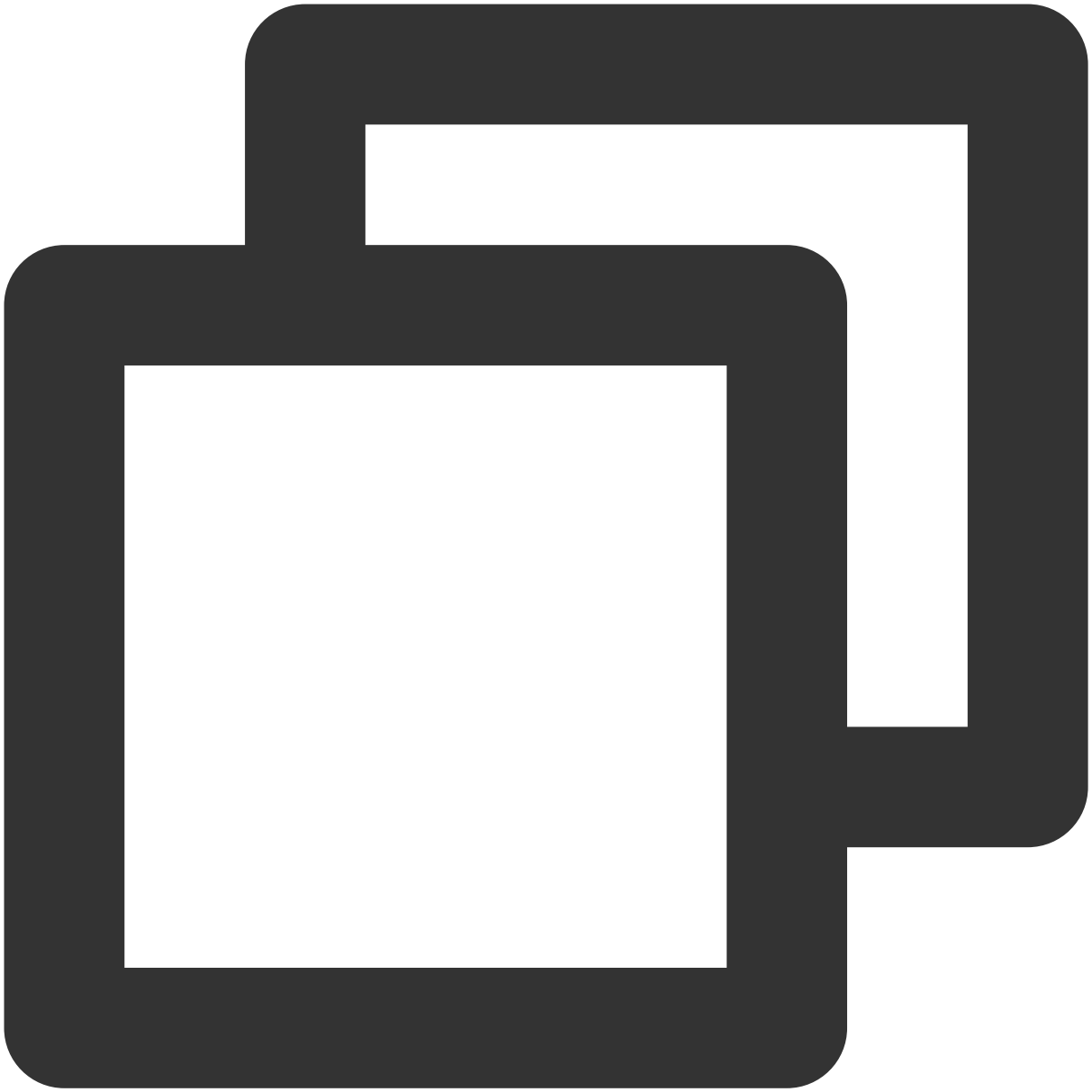
Extending the XFS file system

Run the following command to extend the EXT file system.



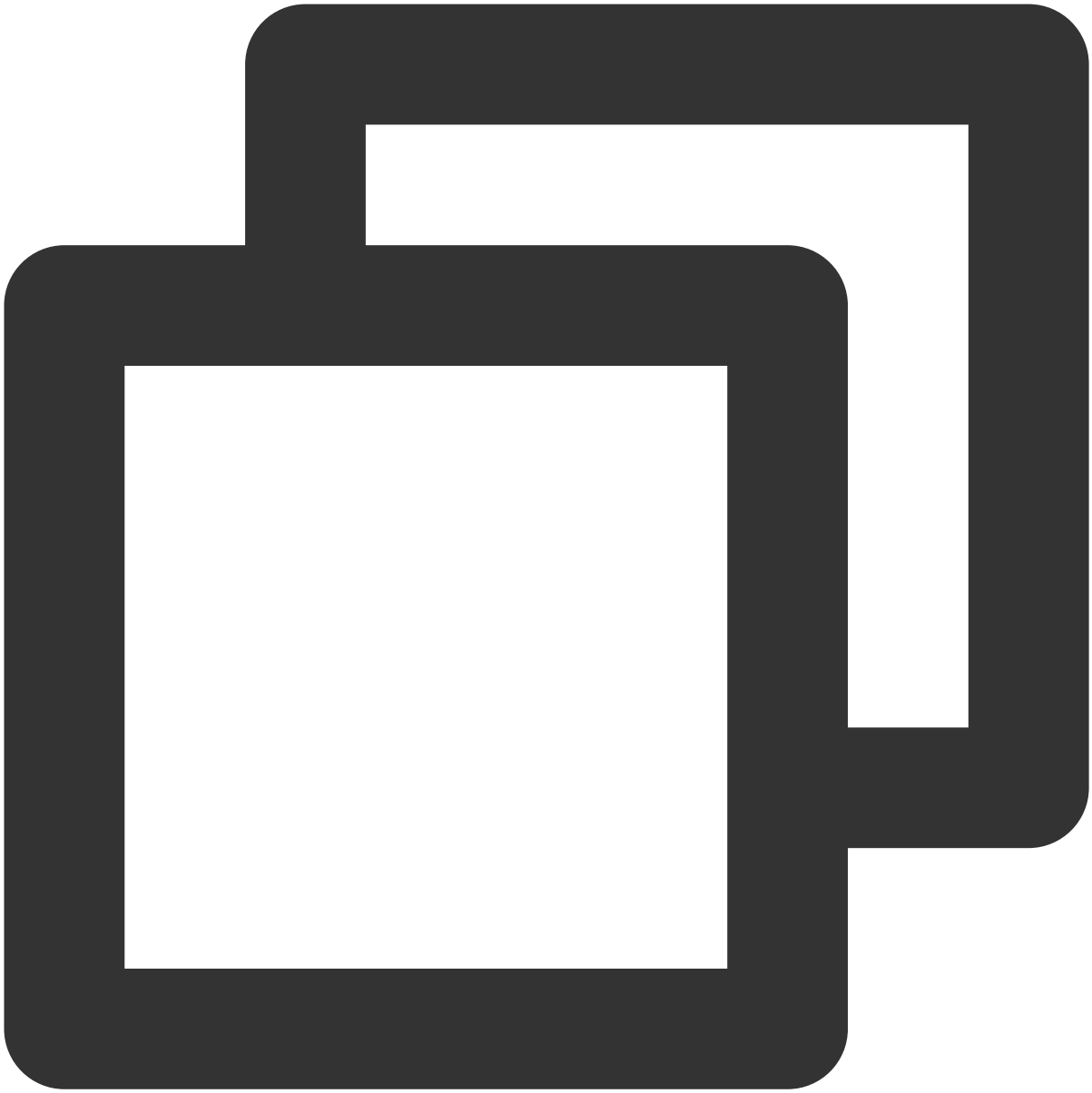
```
resize2fs /dev/corresponding partition
```

Taking the `/dev/vdb1` partition path as an example, run the following command:



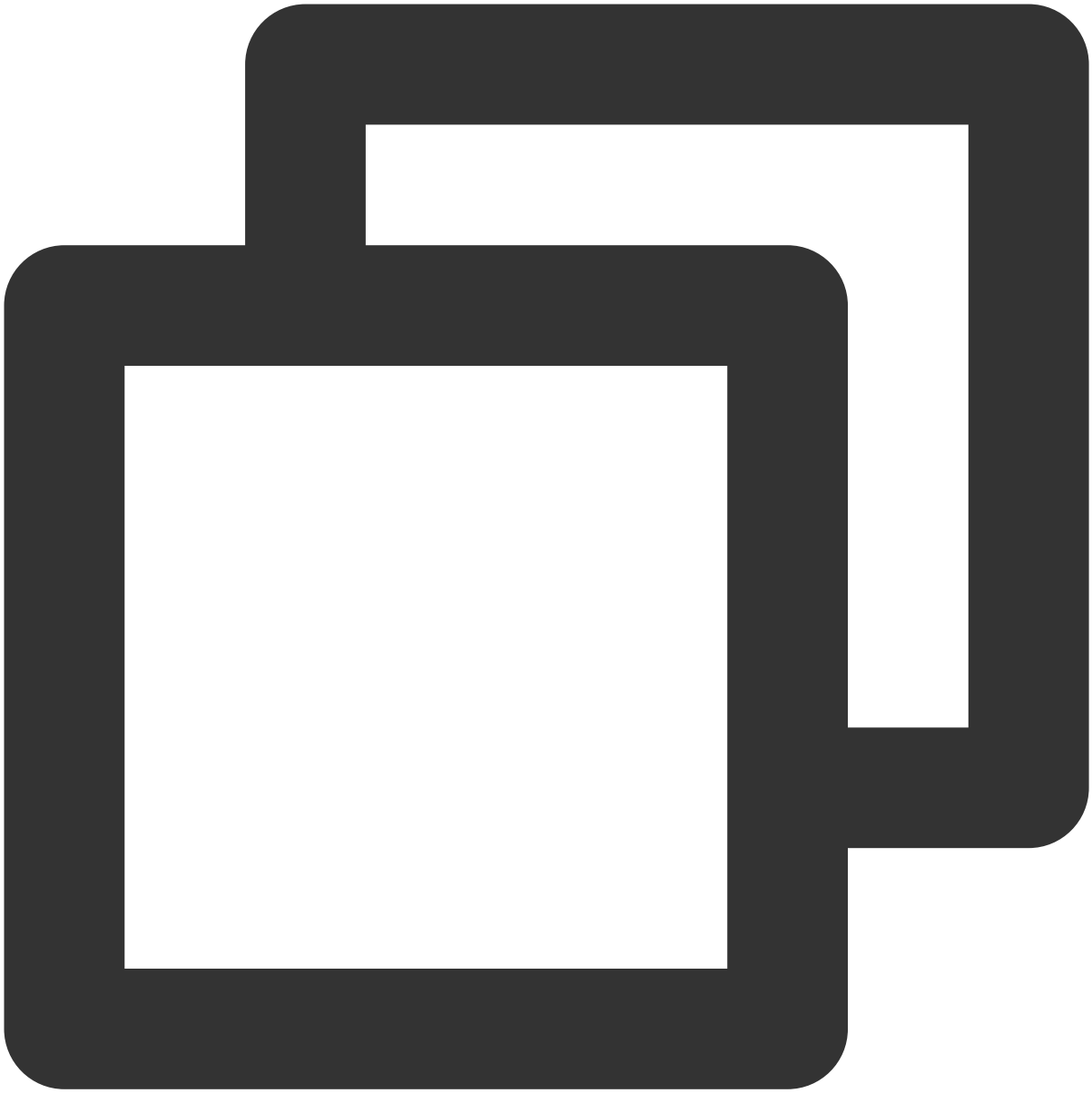
```
resize2fs /dev/vdb1
```

Run the following command to extend the XFS file system.



```
xfs_growfs /dev/corresponding partition
```

Taking the `/dev/vdb1` partition path as an example, run the following command:



```
xfs_growfs /dev/vdb1
```

16. Set partition auto-mounting as instructed in Initializing Cloud Disks (≥2 TB).

At this point, you have converted the MBR partition to the GPT partition. You can run the `df -h` command to view the partition information.