Common Disk Operations in Linux

On this page are several mini-howto's for accomplishing certain disk-related tasks in Linux, similar to the ones you've seen in the Project Primer. You will find the instructions for the experiments to run on the next page. We recommend that you keep both pages open in separate tabs or windows.

Since most of these commands require root access, you might find it easier to run the following once: sudo su - or alternatively you can prepend sudo to each command.

Show available drives

```
parted -l
```

/dev/xvda1 – this is the OS partition

/dev/xvdb – this is the first Ephemeral (instance store) drive

/dev/xvdc – this is the second Ephemeral (instance store) drive

Note: ramdisks will not show up in the output of parted.

Create and format a partition

```
parted /dev/xvdX mklabel gpt
parted /dev/xvdX mkpart db ext4 0% 10G
mkfs.ext4 /dev/xvdX1
```

Parted, the partition editor, is used to manipulate the partition table on block devices. It has replaced **fdisk** and supports newer features like GUID Partition Tables (GPT), which are required for volumes > 2TB. You can also run **parted** with no parameters to get interactive mode.

If you have a small volume (EBS or Ramdisk), it's easier to just format the whole device directly, without creating a partition table (note the missing '1'):

```
mkfs.ext4 /dev/xvdX
```

Mount a volume

```
mkdir /storage/mountpoint
mount /dev/yourdevice /storage/mountpoint
```

mountpoint is your choice, but you have to use a different name for each filesystem that is mounted at the same time. (/storage/ is arbitrary as well – most Linux mount extra drives under /mnt, but this automatically mounts first ephemeral at boot on EC2 images). *yourdevice* is the device or partition that you've previously formatted, such as xvdf, mdop1, etc.

mount

Running mount without parameters shows all mountpoints. Use this to verify that your mount was successful.

```
cd /storage/mountpoint
cp --a /home/mysql_backup/* .
chown mysql:mysql /storage/mountpoint
```

You should copy the database files from backup to the newly mounted partition before running sysbench.

Bind a mounted volume to multiple mount-points

```
service mysql stop
mount --bind /storage/mountpoint /var/lib/mysql
service mysql start
```

mount with the --bind option will attach one filesystem to multiple locations simultaneously. All changes are seen immediately at both locations because they are the same exact files, just extra paths to reach them. We use this trick so that you don't have to edit your MySQL my.cnf configuration file for each new benchmark.

Running sysbench

SysBench has multiple benchmarks, we are going to be running the OLTP (online transaction processing) MySQL benchmark. This command should be run as the "ubuntu" user, unlike the ones run above as root. The sysbench MySQL database has already been populated with 5 million records (~1.5GB) on the AMI (this was done by running "sysbench prepare") and those files backed up from

```
/var/lib/mysql/ to /home/mysql_backup/.
cd ~/sysbench-0.5/sysbench/

export RUN_NAME=yourRun1
export MYSQL_SERVER=yourServer
./sysbench --test=tests/db/oltp.lua --mysql-host=$MYSQL_SERVER --mysql-user=sysbench --mysql-password=project3 --oltp-table-size=5000000 --num-threads=16 --max-requests=0 --max-time=300 --report-interval=5 --oltp-read-only=on run | tee $RUN_NAME.out
```

- Change *yourRun1* to whatever you like, make sure to give each output a different name for when you are collecting results.
- Change *yourServer* either to **localhost** if running the test locally, or to the EC2 hostname of your server instance if running remotely.

In this mode, SysBench will run continuously for 5 minutes, outputting a result every 5 seconds. (the **tee** command outputs to a file and STDOUT simultaneously) After it completes, you can optionally run the following:

```
echo "$RUN_NAME" > ~/$RUN_NAME.csv
tail -n +11 $RUN_NAME.out | head -n 60 | awk '{print $6}' | tr -d ',' >> ~/$RUN_NAME.csv
mv $RUN_NAME.out ~/
```

The above will extract the individual Transactions-Per-Second (tps) values from each line and output into a .csv file. We encourage you to look at the outputs in Excel or similar, to get an idea of how consistent (or inconsistent) the performance is. We are only asking for the average tps values on the checkpoint quiz however (which you can get by sum/divide the .csv file, or from the "transactions:" line of the summary.

Unmount a volume

```
service mysql stop
umount /var/lib/mysql
umount /storage/your-mountpoint
```

optionally,

```
rmdir /storage/your-mountpoint
```

If you get an error trying to unmount, check that none of your shells current directory (**pwd**) is under the mountpoint. Also quit nmon, because that creates a lock on each volume while it's running. If none of that works, you can check open files with the following:

```
lsof | grep /storage/your-mountpoint
```

Create a RAMDISK

```
mkdir /storage/tmpfs
mount -t tmpfs -o size=2001M tmpfs /storage/tmpfs
dd if=/dev/zero of=/storage/tmpfs/ramblock bs=1M count=2000
mkfs.ext4 /storage/tmpfs/ramblock
```

(answer Y)

```
mkdir /storage/ramblock/
mount -o loop /storage/tmpfs/ramblock /storage/ramblock/
```

This is a multi-step process, first to create a tmpfs (which is it's own simple filesystem), then create a contiguous file on that, format it and re-mount it as a loopback device. The reason we create the ext4 partition is so we can be consistent with all of the other tests. We want a comparison of device performance, not filesystem performance. In addition to tmpfs (most popular), Linux also provides ramfs (no size limit so this could keep growing and crash the system if you aren't careful) and the older **`/dev/ram*`** devices (default 16) which requires a kernel boot parameter to change the size.

Then continue with "mount a volume" and "bind" instructions from above, using /storage/ramblock as your device.

Create a software RAID0 device

```
mdadm --create /dev/md0 --level=0 --chunk=256 --raid-devices=2 /dev/xvdb
/dev/xvdc
```

Creates a RAID0 (stripe) of 2 drives, in this example xvdb & c (replace these with your devices). When creating a 4-drive stripe, change --raid-devices=4 and list all 4 drives.

```
cat /proc/mdstat
```

Shows the status of software RAID devices

```
mdadm --stop /dev/md0
```

Destroys the raid device (has to be unmounted first).

"Warm Up" EBS volume

Amazon uses an "on-demand" approach to EBS volumes, so only blocks that are actively being used are brought to the Instance. There can be between a 5% and 50% reduction in performance for the very *first* access of a particular block, but afterwards it is normal. One way of doing this "warm up" with a Linux instance is to read (or write) all of the blocks using dd:

```
dd if=/dev/xvdX of=/dev/null bs=256k
```

(/dev/null is a "sink" - think of it as a black hole, nothing comes out.)