Backup or Clone HDD

There are more then enough approaches to backup or clone a SD card. I prefer to use one of the following two approaches:

1. Creating a raw diskimage using dd
    - Relatively easy
    - Makes a diskimage of the complete diskspace, including not used / empty disk space
    - Results in a large image.
    - Relatively slow.
    - Can only be restored on disk with identical of larger size
2. Creating a file and partition based disk image
    - Takes a bit more steps
    - Makes a diskimage of only the used diskspace, including only used data (files).
    - Results in a small image.
    - Relatively fast.
    - Can be restored on smaller disks

GPT vs MBR

GPT stands for Globally Unique Identifier (GUID) Partition Table, which is a standard for the layout of the partition table on a physical hard disk. GPT is gradually replacing the old MBR (Master Boot Record) standard, which has some limitations that only supports hard disk up to 2TB in size, and up to four primary partitions.

- GPT supports hard drive lager than 2TB.
- Windows can only boot from GPT on a UEFI-based computers.
- Unlike MBR, GPT hard disk allows you to create primary partitions up to 128 due to system limitations.
- GPT disks only support 64-bit Windows system while MBR disks support both 32-bit and 64-bit Windows

Identify Partition Table type using gdisk (or fdisk):

```
# gdisk -l /dev/sda
```

Output should show something similar to:

```
Partition table scan:
  MBR: protective
  BSD: not present
  APM: not present
  GPT: present
Found valid GPT with protective MBR; using GPT.
```

Or with fdisk:

```
# fdisk -l /dev/sda
```

For MBR partition types, output should show something similar to:

```
Disk /dev/sda: 953,9 GiB, 1024209543168 bytes, 2000409264 sectors
Disk model: Samsung SSD 850
* * *
Disklabel type: dos
```

For GPT partition types, output should show something similar to:

```
Disk /dev/sda: 953,9 GiB, 1024209543168 bytes, 2000409264 sectors
Disk model: Samsung SSD 850
* * *
Disklabel type: gpt
```

Be aware of UUID

UUIDs are not hardware-specific but stored in the partition's filesystem. That means cloning a disk or partition with dd will result in the same UUID. However recreating the partitions manually on the new disk (e.g. smaller disk), will result in new UUID. This could result in problems when booting the new disk uses UUID in fstab.

```
cat /etc/fstab
# /etc/fstab: static file system information.
# <file system> <mount point>   <type>  <options>       <dump>  <pass>
# / was on /dev/sda3 during installation
UUID=b2fa29ee-670f-4d44-becc-d9ec368d4a41 /               ext4
noatime,nodiratime,errors=remount-ro 0       1
```

This can be solved by either changing the entries in fstab to the old style: /dev/dbX, Or by getting the new UUID and update fstab accordingy:

```
# blkid /dev/sdb1
/dev/sdb1: UUID="34628ffd-58e6-4a58-9b4d-533719305931" TYPE="ext4"
PARTUUID="fa64ccff-01"
```

MBR Explained

A master boot record (MBR) is the 512-byte boot sector that is the first sector of a partitioned data storage device of a hard disk. The total size is 512 bytes (446 + 64 + 2 = 512). Where,

```
  446 bytes – Bootstrap.
  64 bytes – Partition table.
  2 bytes – Signature.
```

Backup MBR

Next command will copy 512 bytes (MBR) from disk sdX to a backup file:

```
# dd if=/dev/sdX of=mbr.backup.img bs=512 count=1
```

Restore MBR

For disks with identical partitions and size:

Use 512 bytes to overwrite or restore your /dev/sdX the full MBR (which contains both boot code and the drive’s partition table) with the contents of mbr.backup.file.

```
# dd if=/mbr.backup.img of=/dev/sdX bs=512 count=1
```

For disks with different partitions and size:

Use 446 bytes to overwrite or restore your /dev/sdX MBR boot code only with the contents of mbr.backup.file. This will preserve the partitioning schema.

```
# dd if=/mbr.backup.img of=/dev/sdX bs=446 count=1
```

Find HDD block device

Fist step is to figure out which device is our SD Card. Use the 'lsblk' to list all the block devices currently connected to the system:

```
sudo lsblk -o NAME,FSTYPE,SIZE,MOUNTPOINT,LABEL
```

This should give an output similar to the following, illustrating that the SD card is connected to /dev/sdb:

```
AME   FSTYPE   SIZE MOUNTPOINT             LABEL
sda           232,9G
├─sda1 ntfs     100M                        System Reserved
├─sda2 ntfs    69,9G                        WINDOWS 7
├─sda3 ext4   153,7G /
└─sda4 swap     9,2G [SWAP]
sdb            29,8G
├─sdb1 ext4    28,9G
└─sdb2 swap     948M
```

Backup using dd

This will only work for restoring to a larger or identical size disk. Backup image will be very large as well.

Create Backup Image

```
dd bs=4M if=/dev/sdb | gzip > image.gz
```

Restore Backup Image

```
gzip -dc image.gz | dd bs=4M of=/dev/sdb
sync
```

Monitor Progress

Using dd for large SD cards could be quite lengthy. E.g. a 32Gb card could take up to 20 minutes to read. Use the following steps to watch the progress. First, find out the process id of the dd process by running the following in the new virtual terminal.

```
$ pgrep -l '^dd$'
8789 dd
$
```

This shows that the running dd proces has proces Id 8789. The dd process will print out the current statistics when it receives an user signal (USR1). To send the USR1 signal to the dd process:

```
$ kill -USR1  8789
$
```

Note that as soon as the USR1 signal is detected, dd will print out the current statistics to its STDERR.

```
$ dd if=/dev/random of=/dev/null bs=1K count=100
0+14 records in
0+14 records out
204 bytes (204 B) copied, 24.92 seconds, 0.0 kB/s
```

After reporting the status, dd will resume copying. You can repeat the above kill command any time you want to see the interim statistics. Alternatively, you can use the watch command to execute kill at a set interval.

```
$ watch -n 10 kill -USR1 8789
```

Backup HDD using files archive

This approach will work for restoring to smaller disks and will result in a minimal sized image. It consist of the following steps:

Backup:

- Save the partition table structure of the SD card
- Save the MBR
- Mount and save the files for each partition in an individual archive

Restore:

- Create the partitions on the new HDD card
- Restore the MBR to new HDD card
- Mount and restore the files in the partitions

Use **lsblk** to figure out the block device of the HDD card.

Save Partition Table Structure

Save the partition table/structure of the current SD card to a file.

```
sudo sfdisk -d /dev/sdX > part_table
```

Save the MBR

```
# dd if=/dev/sdX of=mbr.backup.img bs=512 count=1
```

Backup the partition data

For each of the partitions do (use the correct fs-type):

```
# umount /dev/sdX1
# mkdir /tmp/source_sdX1
# mount -t ext4 /dev/sdX1 /tmp/source_sdX1
# tar -zcvf image-sdX1.tgz /tmp/source_sdX1
```

Restore Partition Table Structure

Restore the partition table/structure to the new HDD. If the disks have identical size, we can use the saved part_table file. If not we need to create the partitions manually (e.g. with gparted or fdisk).

```
sudo sfdisk /dev/sdX < part_table
```

Now we have the empty partitions that needs to be formated.

```
sudo umount /dev/sdbX
sudo mkfs.ext4 -l root /dev/sdbX
```

Restore the MBR

See explanation on MBR above. For disks with identical partitions and size:

```
# dd if=/mbr.backup.img of=/dev/sdX bs=512 count=1
```

For disks with different partitions and size:

```
# dd if=/mbr.backup.img of=/dev/sdX bs=446 count=1
```

Restore the partition data

Let's mount both partitions and restore the files back on the volumes. The absolute path is stored in the tar file and used to restore the files. It is therefore important that the partition is mounted in the same location as it was done while making the backup. And that you execute at /.

```
# mount -t ext4 /dev/sdX1 /tmp/source_sdX1
```

Now restore the files to the partitions.

```
cd /
sudo tar -zxvf image-sdX1.tgz
```