Backup or Clone SSD

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 (MBR disk)
    - 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

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
```

Using files archives (GPT disk)

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 disk
- Mount and save the files for each partition in an individual archive

Restore:

- Create the partitions on the new disk
- Mount and restore the files in the partitions

Use **lsblk** to figure out the block device of the disk.

Save Partition Table Structure

Save the GPT partition table/structure of the disk to a file.

```
sgdisk --backup=filename /dev/sda
```

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 disk. 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).

```
sgdisk --load-backup=filename /dev/sdb
sgdisk -G /dev/sdb
```

The option G (–randomize-guids)Randomize the disk's GUID and all partitions' unique GUIDs (but not their partition type code GUIDs). This function may be used after cloning a disk in order to render all GUIDs once again unique. Now we have the empty partitions that needs to be formated.

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

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
```

Using files archives (MBR disk)

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 disk
- Save the MBR
- Mount and save the files for each partition in an individual archive

Restore:

- Create the partitions on the new disk
- 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 disk.

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
```