Backup or Clone SD Card

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

Find SD 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:

```
NAME   FSTYPE   SIZE MOUNTPOINT
sda           232,9G
├─sda1 ntfs     100M
└─sda2 ext4   162,9G /
sdb            29,7G
├─sdb1 vfat    41,8M /media/sd/boot
└─sdb2 f2fs    28,9G /media/sd/06a48ea6-a145-48cd-b021-026c2887f3db
```

Backup using dd

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 using files archive

This approach consist of the following steps:

- Retrieve and save the partition table structure of the SD card
- Mount and save the files for each partition in an individual archive
- Create the partitions on the new SD card
- Mount and restore the files for each partition in an individual archive

Use the method above to figure out the block device of the SD card. Let's assume that it is still connected to /dev/sdb. A typical Raspbian disc image has 2 partitions: sdb1 (boot) and sdb2 (root) as shown below:

```
NAME   FSTYPE   SIZE MOUNTPOINT
sda           232,9G
├─sda1 ntfs     100M
└─sda2 ext4   162,9G /
sdb            29,7G
├─sdb1 vfat    41,8M /media/sd/boot
└─sdb2 f2fs    28,9G /media/sd/06a48ea6-a145-48cd-b021-026c2887f3db
```

Save Partition Table Structure

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

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

This results in a text file with similar content as shown below. Note that partition alignment on SD cards should be 4M (8192 x 512 byte).

```
cat part_table

label: dos
label-id: 0xefe3bd3a
device: /dev/sdb
unit: sectors

/dev/sdb1 : start=        8192, size=       85622, type=c
/dev/sdb2 : start=       94208, size=    60657664, type=83
```

Backup the partition data

Now I prefer to make some more usable mount point names. Mount the boot partition off the Raspberry Pi, (/dev/sdb1 with vfat file system) to the temporary directory /tmp/source_boot. And second: Mount the root partition off the Raspberry Pi, (/dev/sdb2 with a 28,9 GB f2fs file system) to the temporary directory /tmp/source_root.

```
   sudo umount /dev/sdb1
   sudo umount /dev/sdb2
   sudo mkdir /tmp/source_boot
   sudo mkdir /tmp/source_root
   sudo mount -t vfat /dev/sdb1 /tmp/source_boot
   sudo mount -t f2fs /dev/sdb2 /tmp/source_root
```

Now, the SD card mount is much better readable:

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

NAME   FSTYPE        SIZE MOUNTPOINT       LABEL
sda               232,9G
├─sda1 ntfs          100M                  System Reserved
```

```
└─sda2 ext4        162,9G /
sdb                 29,7G
├─sdb1 vfat         41,8M /tmp/source_boot boot
└─sdb2 f2fs         28,9G /tmp/source_root
```

First make a compressed tar archive of the boot partition:

```
   sudo tar -zcvf boot.tgz /tmp/source_boot
```

Second make a compressed tar archive of the root partition:

```
   sudo tar -zcvf root.tgz /tmp/source_root
```

Restore Partition Table Structure

Restore the partition table/structure to the new SD card using the file we saved.

```
sudo sfdisk /dev/sdb < part_table
```

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

```
sudo umount /dev/sdb1
sudo mkfs.vfat -F 32 -n boot /dev/sdb1
sudo umount /dev/sdb2
sudo mkfs.f2fs -l root /dev/sdb2
```

Restore the partition data

Let's mount both partitions and restore the files back on the volumes. It is important that

```
sudo mount -t vfat /dev/sdb1 /tmp/source_boot
sudo mount -t f2fs /dev/sdb2 /tmp/source_root
```

Now restore the files to the partitions.

```
cd /
sudo tar -zxvf /tmp/boot.tgz
sudo tar -zxvf /tmp/root.tgz
```