Google kills Measure
Measure, which allowed people to measure things in the real world, has now been discountinued. **http://dgit.in/Jul21-61**

Twitter NFTs with a twist
Twitter users have to reply to the Tweet inorder to get a chance of winning them. **http://dgit.in/Jul21-62**

Pop in BTRFS

Install Pop!_OS on BTRFS with Backups and Data Deduplication

Aaruni Kaushik | feedback@digit.in

BTRFS

BTRFS, often pronounced as Butter-FS is a modern copy-on-write filesystem for Linux systems. It brings to the table features like transparent compression, snapshots, subvolumes, multi device support, deduplication, incremental backups, and more. Chris Mason, the principal Btrfs author, stated that its goal was "to let [Linux] scale for the storage that will be available. Scaling is not just about addressing the storage but also means being able to administer and to manage it with a clean interface that lets people see what's being used and makes it more reliable".

POP!_OS

Pop!_OS is a Linux distribution maintained by System 76. It is derived from Linux Ubuntu, and adds many tweaks and quality of life improvements over Ubuntu. Particularly, it works like a charm on hardware shipped by System 76, and is also the only distribution which lets me use the NVIDIA dGPU on my laptop. Find out more about Pop!_OS on the official website at https://pop.system76.com. The guide is for pop 20.10, and if you user an older / newer version, "your experience may wary"

THE INGREDIENTS

For this particular article, we consider a computer with two storage drives of roughly equal performance. This could be two HDDs, or two SSDs, or even something else, as long as they are in the same performance class. You could apply this guide to a setup with a very fast drive and a much slower drive and it should still work, but the overall performance of the resulting setup would be abysmal. You could, however, also apply the guide to three or more drives of comparable speed.

We will use BTRFS' ability to pool disks together to combine the storage of both the drives, and present it to the OS as one large filesystem. As this setup is not supported by Pop!_OS' installer by default, we will need to install Pop on a configuration it is compatible with, and then move things around to be in line with our goals.

Note that much of the trickery involved is in getting the right subvolumes in place. If you already have a btrfs setup, you can simply add another disk to the pool without jumping through hoops.

The guide also assumes you are running a UEFI computer. Any recent computer should support UEFI, as do many hypervisors.

TL;DR

In very short, we take the following steps:

1. Boot Pop!_OS installer, enter the locale settings
2. Open gparted, partition one of your disks to have ESP, Recovery, Swap, and a BTRFS partition. On the other disk, create a single large partition without a filesystem
3. Install Pop using the BTRFS partition on the first disk as root, and the BTRFS partition on the second disk as home. When the installer asks to reboot, ***do not reboot!***
4. Mount the root partition somewhere, create a BTRFS subvolume named `@` for root, and move everything into it. Also create a subvolume for home named `@home`.
5. Add the large btrfs partition into the mountpoint, and balance your data across both drives
6. Bind mount `/dev, /dev/pts, /proc, /sys, and /run` into your mountpoint, and `chroot` into it. Update your `fstab` by hand. You can use `kernelstub` command line utility to add rootflags and update bootloader configuration.
7. Automount your drives via `fstab`, and check that there are no errors. Optionally add a timeout to the loader selection screen at `/boot/efi/loader/loader.conf`

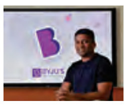
THE DETAILS

Pop Installer

Boot off your Pop install media. Before anything else, the installer asks you to confirm your locale information : language, keyboard layout, etc. Once you go through this process, it will present you the option to install or try the demo environment. At this point, we want to click on the Try Demo button, which will close the installer, and leave us in a full fledged Pop!_OS environment.

Disk Preparation

Use `lsblk` to find out what the names of your drives are. For HDDs, it should be something like `sda` and `sdb`, and for SSDs, it should be something like `nvme0n1` and `nvme1n1`. Remember which are the drives you want to dedicate to your install. If you find yourself confused with which drives you want to wipe, consider unplugging any drives you do not want to modify. In this guide, the first drive is `/dev/sda` of size 50

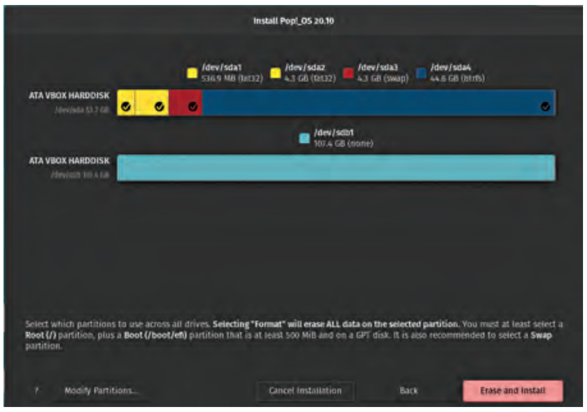


Byjus selling student data
salsken.ai has secured an exposed server that Byjus was using to sell private and sensitive data. **http://dgit.in/Jul21-63**

Android apps in Windows 11
Users can now sideload android applications directly into windows 11. **http://dgit.in/Jul21-64**

GiB, and the second drive is **/dev/sdb** of size 100 GiB.

Open GParted, and create a partition table of type GPT on both the drives. On one of the drives, add a FAT32 partition of atleast 512 MiB, a FAT32 partition of 4095 MiB, a SWAP partition at least as large as your physical RAM, and make a BTRFS partition in the remaining space. On the second drive, devote the entire space to a single large partition without a filesystem. For this,



Select which partitions to use across all drives. **Selecting "Format" will erase ALL data on the selected partition.** You must at least select a **Root (/)** partition, plus a **Boot (/boot/efi)** partition that is at least 500 MiB and on a GPT disk. It is also recommended to select a **Swap** partition.

The partitioning scheme

use the "cleared" filesystem option in gparted. It will change to "unknown" after successful partitioning. The computer used in this guide has 4 GiB of RAM, so we create a swap parition of 4 GiB.

Install Pop

Open the Pop!_OS installer. It will ask you for your locale settings again. This time, proceed by choosing custom install. On the next screen, choose to mark all the partitions you created appropriately. So, the small FAT32 is the /efi/boot partition, the big FAT32 partition is /recovery, the SWAP partition is Swap, the small BTRFS partition is /. For now, we ignore and the big partition. If you have been following this guide exactly, your disk selection screen should like like the following :
Click on Erase and Install to begin installation. You can click on the small terminal icon near the progress bar if you want to take a peek under the hood while the system installs. At the end of the install process, when it asks you to power off or reboot, do not proceed. Instead, simply ignore it for a while. You cannot minimize this, so you may want to go to a different virtual desktop if it helps **(Ctrl+Super+Down).**

BTRFS Configuration

In this step, we will setup two btrfs subvolumes: one to be mounted later as your root partition, and the other to be mounted as your home partition. Mount the root partition (the btrfs partition from your first drive) onto /mnt . Then, change your working directory to it, and create a btrfs subvolume. We will name this subvolume as @, following the naming convention used by the Ubuntu installer. Then, we move all the things created by the Pop installer into the new subvolume. After that, we also make a subvolume named **@home.**

```
sudo mount /dev/sda4 /mnt
cd /mnt
sudo btrfs subvolume create @
ls | grep -v @ | sudo xargs
mv -t @
sudo btrfs subvolume create @
home
```

At this point, if you run ls in the terminal, you should see only get the following one line output:

```
@ @home
```

If you see something different, an error has been made at some point, and you should start from the top.

Adding Second Device

We will now add the second device to the mount point. This will present to the OS as one big filesystem with the combined capacity of both the storage devices. We also run btrfs balance, which will try to balance the usage across both devices which make up the filesystem. The balance operation might take a while depending on your disk speed.

You can do this operation at anytime, even after installing the system and rebooting into it. But the more data you already have, the longer it will take to balance them across devices. You may also remove any device in the pool in the future via btrfs device remove, which will take care of reclaiming the data blocks from the device being removed onto the remaining devices.

```
sudo btrfs device add /dev/
sdb1 /mnt
sudo btrfs balance start /mnt
--full-balance
```

System Configuration Files

We have now significantly changed the state of the system from what the installer did. So, we need to tweak some configuration files to account for our changes. We need to change 4 files in particulary, two of which reside on the ESP. The files are

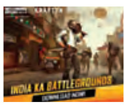
```
- /mnt/@/etc/kernelstub/
configuration
- /mnt/@/etc/fstab
- /mnt/@/boot/efi/loader/
loader.conf
- /mnt/@/efi/loader/entries/
Pop_OS-current.conf
```

We will change fstab, and loader.conf by hand, and use the kernelstub management utility to change the rest.

We will chroot into the newly installed OS. For this, we bind mount the required paths into /mnt.

```
cd /
sudo umount /mnt
sudo mount -o
defaults,subvol=@,ssd,
discard,noatime,space_cache,
compress=zstd,commit=120 /
```

Youtube web to redesign
YouTube is testing description redesign for web with more focus on comments.
http://dgit.in/jul21-65



BGMI launched on PlayStore
Battlegrounds Mobile India goes live on Play Store for Android Smartphones.
http://dgit.in/jul21-66

```
dev/sda4 /mnt
for i in /dev/dev/pts/proc/
sys/run; do sudo mount -B $i
/mnt$i; done
sudo cp /etc/resolv.conf /
mnt/etc/
sudo chroot /mnt
```

fstab

First, we edit the fstab

fstab is the configuration file which contains all the partitions which should be automatically mounted on boot. We change this file manually via the nano text editor. We want to add additional boot options to the last line, and then add another line for the home partition. As the UUID of the home partition needs to be the same as that of /, we will copy-paste the edited last line, and simply add `@home` instead of `@`. We want the following boot options: `defaults,space_cache,noatime,compress=zstd,commit=120,ssd,discard,autodefrag,subvol=Z`

For SSDs, you should omit adding `autodefrag`. For HDDs, you should omit adding `ssd,discard`.

Open the file with nano

```
nano /etc/fstab
```

In the last line, find the word `defaults`, and replace it with `defaults,space_cache,noatime,compress=zstd,commit=120,ssd,discard,autodefrag,subvol=@`. Remember to change the line depending on if you have HDDs or SSDs. Once you have made the changes, hit the **End** key to transport the cursor at the end of the line. Press the key combination **Alt+A** to set a mark, and hit the **Home** key to mark the entire line. Then press the key combination **Alt+6** to copy the line. Go to the next line, and paste the line by pressing the combination **Ctrl+U**. Then edit the **/** to become **/home**, and `subvol=@` to become `subvol=@home`. Note that the UUID for both the root partition and the home partition must match. The file should look like the following:

```
PARTUUID=aaaaaaaa-aaaa-aaaa-
aaaa-aaaaaaaa  /boot/efi
vfat  umask=0077  0  0
PARTUUID=bbbbbbbb-bbbb-bbbb-
bbbb-bbbbbbbb /recovery vfat
umask=0077 0 0
/dev/mapper/cryptswap  none
swap  defaults  0  0
UUID=yyyyyyyy-yyyy-yyyy-
yyyy-yyyyyyyyyyyy  /  btrfs
defaults,space_cache,noatime,
compress=zstd,commit=120,ssd,
discard,subvol=@  0  0
UUID=yyyyyyyy-yyyy-yyyy-yyyy-
yyyyyyyyyyyy  /home  btrfs
defaults,space_cache,noatime,
compress=zstd,commit=120,ssd,
discard,subvol=@home  0  0
```

We can check that the fstab is right by running the mount command: sudo mount -av. If the output shows errors of any kind, your system will fail to boot, and you shuold start over, or seek help. The output should look something like as below:

```
  /boot/efi : successfully
mounted
  /recovery : successfully
mounted
  none : ignored
  / : ignored
  /home : successfully
mounted
```

kernelstub

We now add changes to the kernelstub configuration. Kernelstub is the bootloader, the program which actually loads the Linux kernel and starts your OS. We indicate to the loader the subvolume where our root filesystem resides. We also update the init file to be double sure that our changes have propogated to all the required places.

```
  kernelstub -a
'rootflags=subvol=@' -l -s
  update-initramfs -c -k all
```

loader.conf

This file manages the settings of the bootloader screen. We add a timeout of 5 seconds, so its easier to boot into the firmware or access recovery options. This is an optional change, so you can skip this if you don't want a 5 second delay each time you start your computer.

```
echo "timeout 5" >> /boot/
efi/loader/loader.conf
```

Reboot

If you have reached this point, and not run into any errors, you can now reboot!

Step out of the chroot environment by typing exit at the terminal, and then close the terminal. Go back to the Pop!_OS installer, and hit "Restart Device". You shouold now boot into a shiny new install of Pop!_OS, with a side of butter! The first time boot will welcome you to your computer, and help you set up your User Account.

You now have a fully functional system, but continue reading for quality of life configuration like backups and data deduplication.

TIMESHIFT

Timeshift is a nifty tool which simplifies the backup process of your system. The utility is meant to roll back any breaking system changes by keeping versioned snapshots of your computer. It will save you from an accidental delete, or a weird update which breaks your environment. It will not save you from a failing disk on its own.

Note that snapshots on btrfs do not take much extra space, so you could have as many snapshots as required to fit into your backup strategy, as long as they're under a hundred. Its possible to have even more snapshots, but you'll start running into performance problems around then.

Also note that creating btrfs snapshots is an instant action that barely impacts system resources, but deleting snapshots is a bit more involved, and the system might feel a little sluggish for the few minutes it takes for btrfs to delete a snapshot.

Install Timeshift via APT as follows

```
sudo apt install timeshift
```

After a successful install, open Timeshift from the Activities menu on the top left, or by hitting the Super key. The first time, timeshift will ask for configuration. You may choose to expand the help dropdown for extra info. On the first screen, select btrfs. Hit Next. On the next screen, select the disk which has our root partition. It will be easy to identify, as it will have a size greater than the disk size. Since we added our second disk to the first filesystem, selecting one will also automatically select the other. Hit Next. In the next screen, you will be asked to setup a schedule for snapshots. You can tune this to your preferred settings. A sane starting point is 5 daily backups, 3 weekly backups, and 2 monthly backups. Its also a good idea to keep a few 'Boot' backups, which save a known working state of the computer 10 minutes after every time you start your computer. When you are done, click "Next".

In this section, enable both the options: to inlcude @home in the backup, and to enable BTRFS qgroups. The first option also backups your personal files and folders along with core system data. The second option gives you an accurate report of space used by each snapshot. Hit Next.

The final screen is a short conclusion of your settings and expected behaviour. Notice that the last point warns you that btrfs snapshots based backups will not save you from a disk failure, and on first glance suggest that you should use rsync based backups to avoid that. This is misleading, as rsync based backups on the same system will also not guard against disk failures. To move your btrfs based backups onto an external drive, you can use btrfs-send and btrfs-receive utilities.

Click on Finish to proceed to the main program interface. In this interface, you can hit "Create" to make your first snapshot. You can see how quick and easy it is to backup files via btrfs! Note how in the screenshot I have two backups of nearly 6 GB worth of data each, but their actual sizes are just 1.3 MB and 163.8 KB!

> WARNING: It is possible that when you fiddle about at Settings and return to the main window, it will warn you that you are out of space and backups will no longer be created. This is in error, and is because it is checking the wrong device, as our devices are pooled together via btrfs. Simply close and open Timeshift again to fix this.

timeshift-autosnap-apt

timeshift-autosnap-apt is an optional tool you can install in this setup. This tool takes a snapshot backup of your system every time the APT package manager is invoked : so, before install, uninstall, or update operation. This is a convenient way to have automatic backups in case an update or a new package break your system. First, we install the dependencies for the autosnap utility.

```
sudo apt install git make

  Then, we clone the git repo
to your computer, and compile
the utility.

  mkdir -p ~/.deb
  git clone https://github.
com/wmutschl/timeshift-
autosnap-apt ~/.deb/timeshift-
autosnap-apt
  cd ~/.deb/timeshift-
autosnap-apt
  sudo make install
```

Once installed, we tune the configuration file to our setup by running sudo nano /etc/timeshift-autosnap-apt.conf

We set snapshotBoot to false, maxSnapshots to 5, and updateGrub to false. Once your config file is tuned to your liking, save and exit nano. Your computer will now backup before any package changes done via apt!

DATA DEDUPLICATION

On btrfs, one can also deduplicate data. Deduplication here doesn't change the number of files you end up with, but saves on space by linking identical data, so its stored only once, and all the copies merely need to link to the original copy. Deduplication on btrfs happens on a filesystem block level, not just on a file level. So, if two files are not identical, but share some blocks of data, you still save space using the same strategy.

We use the tool duperemove for the process. The utility reads all of your files to compute checksums of each block of each file, and then submits them to the Linux kernel as candidates for deduplication. This is a safe method, so that none of your data is lost in the process.

Install dupermove from apt. Note that if you have timeshift-autosnap-apt installed, you might see much more output than you are used to. This is because the autosnapper will take a system backup after you are done downloading duperemove, but before installing it. In case you already have 5 autosnap backups, it will also delete the oldest backup. All this extra activity is printed to your terminal.

```
sudo apt install duperemove
```

Once you are done downloading, you can run duperemove. Its a good idea to let duperemove maintain a list of file checksums so that subsequent runs of the application are incremental, and thus, faster. You can add the invocation to a crontab so it runs on its own every once in a while, or run it manually when you think you have enough data duplicates that its turning into a problem.

```
  sudo duperemove -dhr
--hashfile=/.dedupehash /
```

This process may take a while, but will display a summary when its done. On even just a fresh install, I was able to save 28 MB! d