Btrfs hands on: My first experiments with a new Linux file system

Summary: *Variously called "B-Tree" or "Better" or even "Butter", here's what I found when I started to explore this 'fast-moving target' of a file system.*

By J.A. Watson for Jamie's Mostly Linux Stuff | November 28, 2013 -- 12:09 GMT (04:09 PST)

Btrfs is a new file system for Linux, one that is still very much in development. Although I wouldn't exactly describe it as "experimental" any more, it is, as stated in the Wiki at kernel.org (https://btrfs.wiki.kernel.org/index.php/Main_Page) , "a fast-moving target".

It has also been said publicly that the basic format and structure of the filesystem should now be stable; it would only be changed in the future if some overriding reason or need is found.

The point of all this should be clear — it is still very early days, and it is not recommended to use btrfs in critical systems of any kind.

I leave it to the reader to decide how critical their systems are; for my own purposes, I will be using btrfs on several systems that I use as testbeds, some of which I carry with me and use for normal work on a daily basis, so it will get a "real" test, but I will not be using it on the primary systems that my partner and I use for home/work/business activities.

So, with that out of the way, now let's get to the point. I have no intention of trying to repeat or even paraphrase the information from their Wiki about the reasons behind Btrfs, the complete list of advantages it offers over other filesystems, or the disadvantages, limitations or preferred situations or scenarios for its use.

I will say that I became interested in Btrfs for the following reasons:

- It is obviously becoming increasingly important to the future of Linux systems, and I want to keep current with such developments
- It is possible to dynamically resize mounted filesystems
- filesystems can span physical volumes, with optional RAID support
- physical volumes can be added to and removed from mounted filesystems
- compression option
- snapshots (including read-only and read-write) and seeds

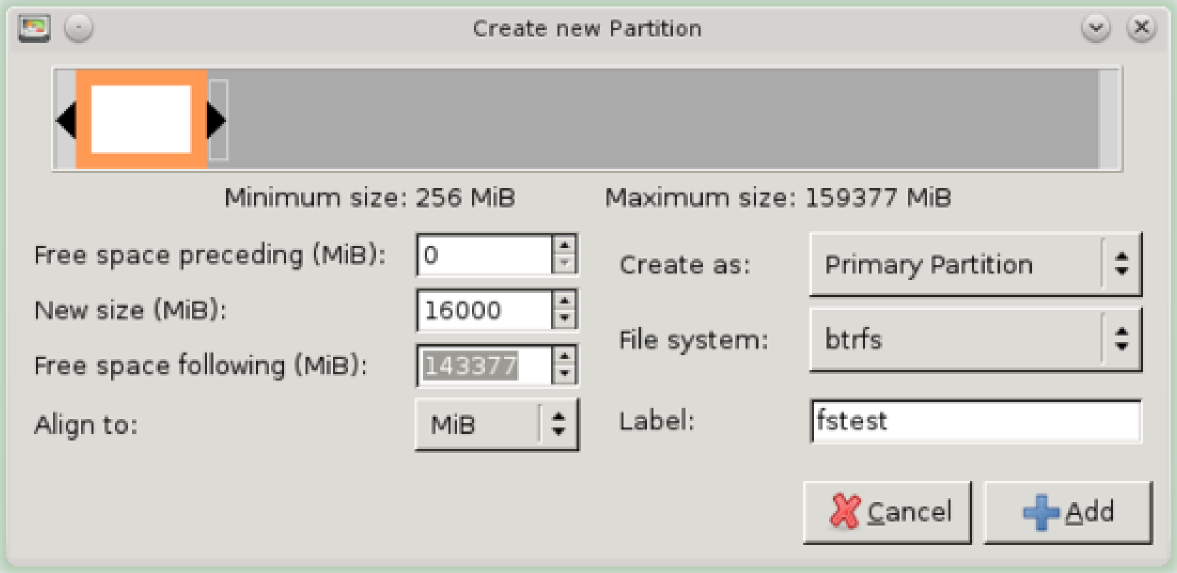
This is only a very short list, especially compared to the detailed list included in the Wiki mentioned above, but these are all issues that I have had to deal with, or suffered from not being able to deal with, over the years.

There have been a few other developments in file systems over the years which looked like they might address some or all of these issues (xfs and zfs come to mind), but for one reason or another (usually non-technical, unfortunately) they haven't become generally viable alternatives. I have been watching btrfs for some time, and had told myself that when it seemed stable enough and well enough integrated and supported in Linux, I would give it a try. As will be shown below, the latest openSuSE and Fedora releases seem to have reached that point.

So the first step is to get some kind of btrfs filesystem. There are two ways to do this — create one (duh), or convert one (huh? really? cool!). The simplest example would be creating a new btrfs filesystem within a specified disk partition, for example:

```
mkfs.btrfs /dev/sda16
```

Of course, this is the "old school" CLI approach, and the same thing can be done using most of the modern GUI disk management utilities, such as *gparted*:



Minimum size: 256 MiB Maximum size: 159377 MiB

Free space preceding (MiB): 0
New size (MiB): 16000
Free space following (MiB): 143377
Align to: MiB
Create as: Primary Partition
File system: btrfs
Label: fstest

creating a btrfs filesystem in gparted

Alternatively, one can convert an existing ext3 or ext4 filesystem to btrfs.

Important! There are a variety of limitations and restrictions on using btrfs for the root (/) and/or boot (/boot) filesystems.

Do not go charging off and convert these kinds of critical filesystems to btrfs, because you will almost certainly be disappointed with the consequences. The kinds of things one might want to convert to btrfs are your home, work, data and other such additional filesystems.

Also, anyone doing this should make sure they have adequate backups of whatever they convert — not only because btrfs is still in development, but also because they might well make a mistake, or get surprised by something, and end up deleting, corrupting or otherwise losing the data in a converted filesystem.

It's possible to convert an existing ext3/ext4 filesystem to btrfs. The filesystem has to be unmounted, and should be checked (fsck) first. The conversion command is simple:

```
btrfs-convert /dev/sdXX
```

Replace the *XX* in the example above with the partition to convert. This will convert the filesystem to btrfs, and it will make a backup copy of the original partition which will be stored in a subvolume of the converted filesystem.

At this point I have a btrfs filesystem, either a newly-created emtpy one, or one which contains the data converted from one of your existing ext3/ext4 filesystems. I could go on from here and learn about things like subvolumes, snapshots, growing, shrinking and adding partitions and devices. But my interest is in creating complete btrfs-based systems, so I am going to press on with that. I will return to a discussion of most of those other features and capabilities in a subsequent post.

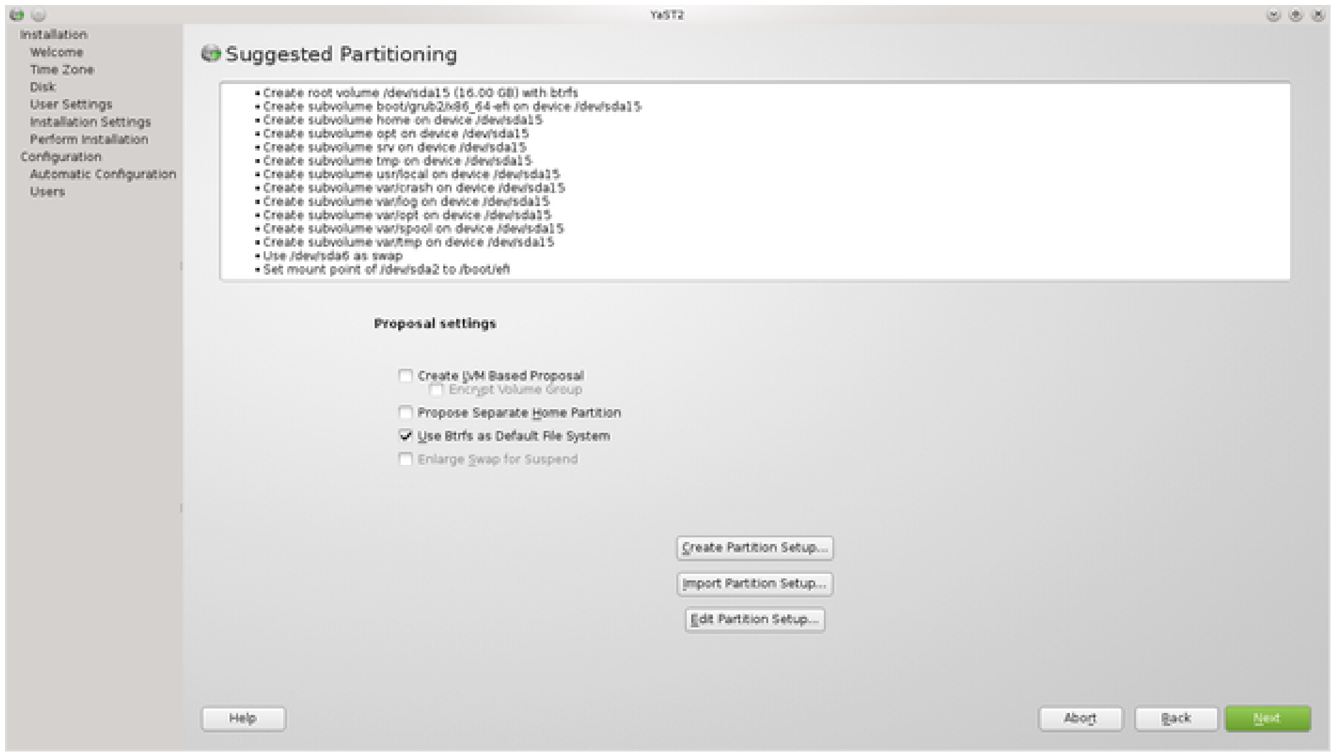
According to the btrfs Wiki, at least the following Linux distributions include support for btrfs filesystems: Arch, Debian, Fedora, Gentoo, openSuSE and Ubuntu.

I know that there are others by this time, including of course a lot of variations and derivatives of the above list, and of course there are commercial distributions including SUSE, Red Hat and Oracle. I happen to be right in the middle of new releases from openSuSE (13.1 just out) and Fedora (20 about to come out), so I decided to use them for testing.

When you talk about a distribution "including btrfs support", there are three very general levels possible:

- Include the *btrfs-tools* package. This will give you the ability to create, convert, examine and manipulate btrfs filesystems.
- Include support for btrfs in the operating system installer utility. This means that the root filesystem can be btrfs, and the installer takes responsibility for making sure that you can boot it somehow (see the next point)
- Include support for btrfs in the bootloader (grub, lilo or whatever). This means that your entire system can be btrfs all the way through, but it is not strictly necessary; as we have learned when other new filesystems have come along, you can keep a separate /boot partition of type ext4 (or less), and use that with non-btrfs compatible bootloaders while having everything else (including the root filesystem) btrfs.

openSuSE 13.1 is a btrfs through-and-through system, including boot capability, and creating a btrfs system is as easy as checking the appropriate box in the disk partitioning screen:



- Create root volume /dev/sda15 (16.00 GB) with btrfs
- Create subvolume boot/grub2/x86_64-efi on device /dev/sda15
- Create subvolume home on device /dev/sda15
- Create subvolume opt on device /dev/sda15
- Create subvolume srv on device /dev/sda15
- Create subvolume tmp on device /dev/sda15
- Create subvolume usr/local on device /dev/sda15
- Create subvolume var/crash on device /dev/sda15
- Create subvolume var/log on device /dev/sda15
- Create subvolume var/opt on device /dev/sda15
- Create subvolume var/spool on device /dev/sda15
- Create subvolume var/tmp on device /dev/sda15
- Use /dev/sda6 as swap
- Set mount point of /dev/sda2 to /boot/efi

Proposal settings

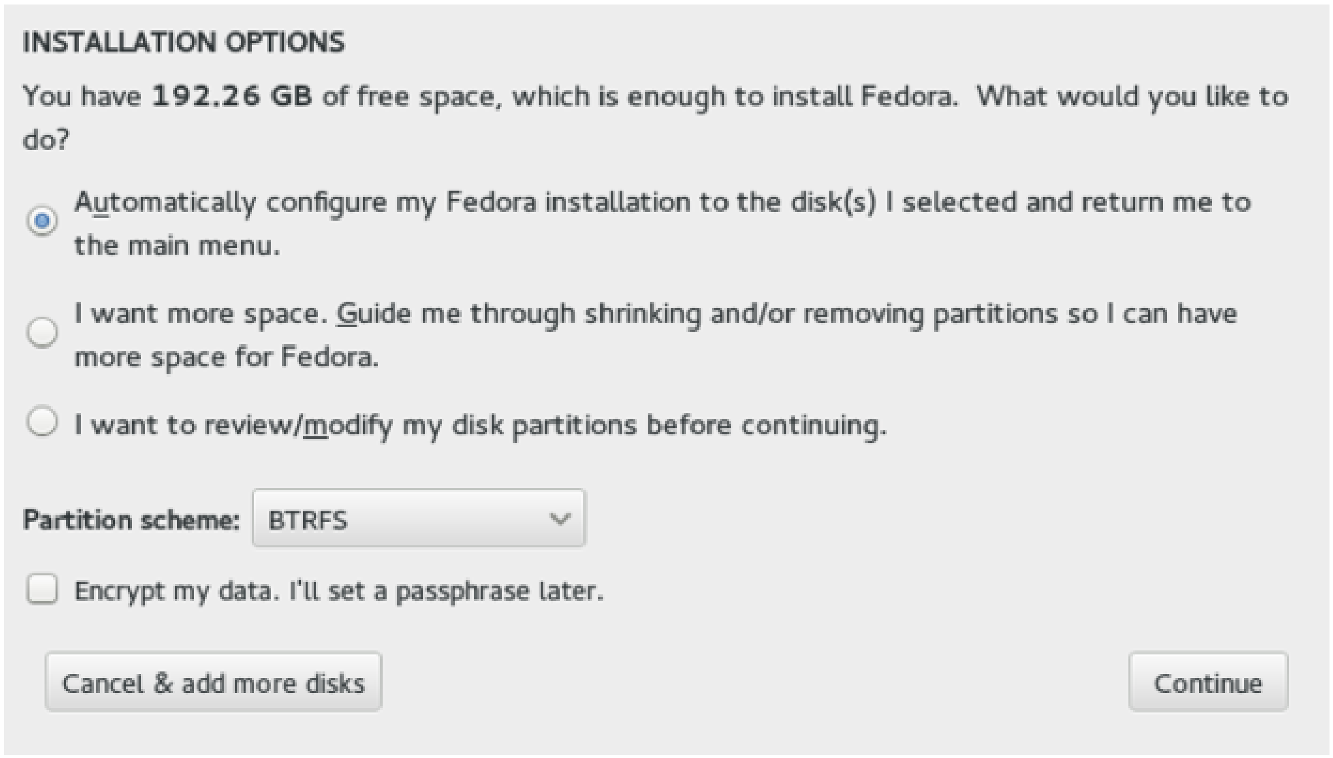
- ☐ Create LVM Based Proposal
  - ☐ Encrypt Volume Group
- ☐ Propose Separate Home Partition
- ☑ Use Btrfs as Default File System
- ☐ Enlarge Swap for Suspend

openSuSE installer creating a btrfs system

For simplicity here I have removed the separate /home partition. The window shows the actions which will be taken to create this layout, it is always a good idea to review this information before proceeding.

If for some reason I was deleting, overwriting or otherwise destroying an existing partition, it will be shown there in red. This looks like a pretty scary list of actions, especially compared to my normal ext4 selection, which produces one or two lines here, but on closer inspection it is pretty simple. It will create only one new partition (/dev/sda15), and then the next 11 lines are just subpartitions to set up the various parts of the root filesystem. The final two lines configure openSuSE to use the existing swap and EFI boot partitions. This is actually so simple, and so easy to do, that I was sceptical at first that it would work, but it really does, and very well.

Fedora 20 is a btrfs root with ext4 /boot configuration, and it requires a bit more manipulation to achieve it. After selecting the disk for installation, I get the **Installation Options** screen shown below. In this screen, change the **Partition Scheme** from the default *LVM* to *BTRFS*. If I were willing to accept the defaults for everything else (file system size and layout), I could just go ahead and let the installer run. I prefer to make some adjustments to the defaults — most importantly to use the existing swap and EFI boot partitions - so I also check the "*I want to review/modify...*" option on this screen.


INSTALLATION OPTIONS

You have **192.26 GB** of free space, which is enough to install Fedora. What would you like to do?

[✓ Automatically configure my Fedora installation to the disk(s) I selected and return me to the main menu.]

I want more space. Guide me through shrinking and/or removing partitions so I can have more space for Fedora.

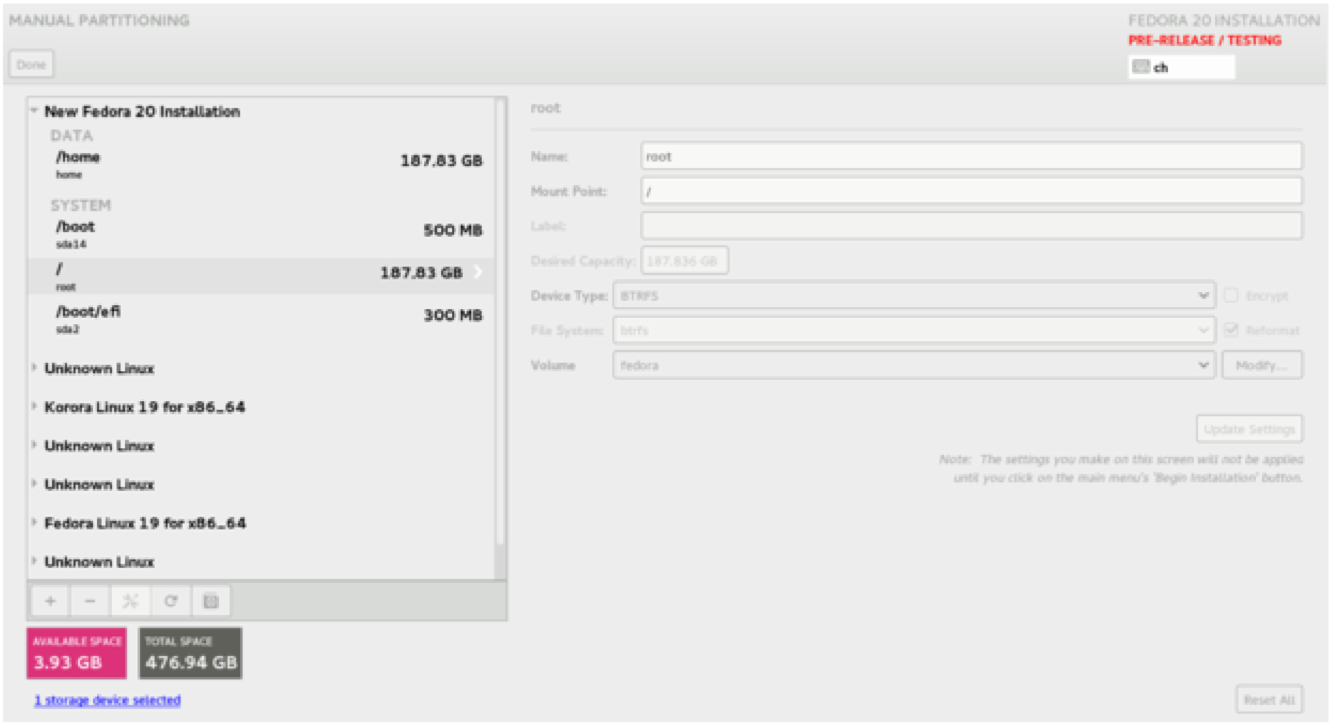
I want to review/modify my disk partitions before continuing.

Partition scheme: BTRFS

Encrypt my data. I'll set a passphrase later.

Fedora anaconda - selecting btrfs

Now I am in the **Manual Partitioning** screen. Here I have removed the swap area that anaconda wanted to create (don't worry, it will find the existing swap partition), and I have redirected the EFI boot to the existing partition as well. For our purposes the more relevant and interesting parts are that anaconda will create a btrfs filesystem (by default using all available space on the disk), and put both root and home in it, and it has created an ext4 /boot partition.


MANUAL PARTITIONING

FEDORA 20 INSTALLATION
PRE-RELEASE / TESTING

New Fedora 20 Installation

DATA
/home — home — 187.83 GB

SYSTEM
/boot — sda14 — 500 MB
/ — root — 187.83 GB
/boot/efi — sda2 — 300 MB

Unknown Linux
Korora Linux 19 for x86_64
Unknown Linux
Unknown Linux
Fedora Linux 19 for x86_64
Unknown Linux

AVAILABLE SPACE 3.93 GB
TOTAL SPACE 476.94 GB

1 storage device selected

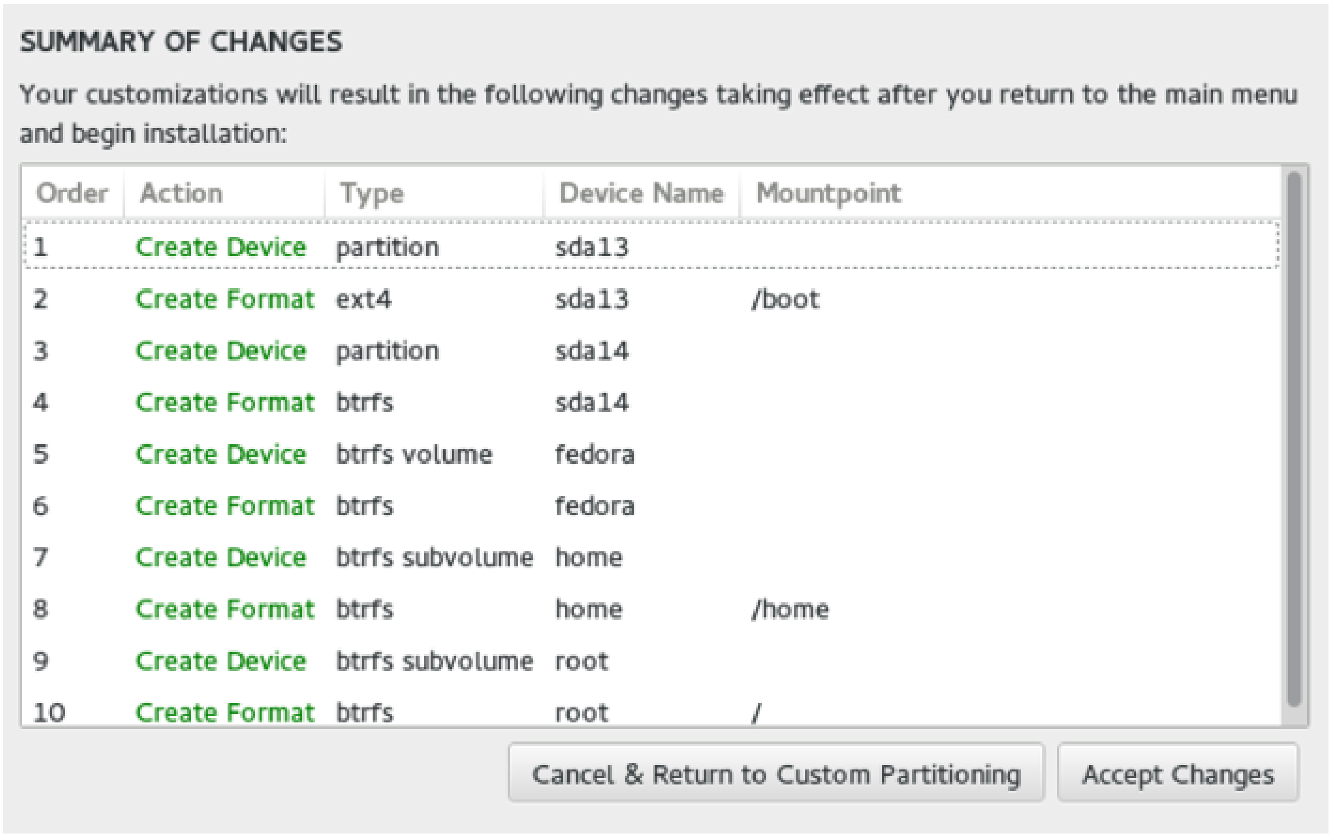
root

Name:	root
Mount Point:	/
Label:	
Desired Capacity:	187.836 GB
Device Type:	BTRFS [ ] Encrypt
File System:	btrfs [✓ Reformat]
Volume	fedora

Note: The settings you make on this screen will not be applied until you click on the main menu's 'Begin Installation' button.

Fedora anaconda disk layout for btrfs creation

If I didn't want btrfs to use all of the free space on the disk, I could **Modify** and then set a fixed size for it. When I click **Done**, anaconda shows me the actions that it will take. Again, it is a good idea to review these to make sure it is not doing something unintended.


SUMMARY OF CHANGES

Your customizations will result in the following changes taking effect after you return to the main menu and begin installation:

Order	Action	Type	Device Name	Mountpoint
1	Create Device	partition	sda13	
2	Create Format	ext4	sda13	/boot
3	Create Device	partition	sda14	
4	Create Format	btrfs	sda14	
5	Create Device	btrfs volume	fedora	
6	Create Format	btrfs	fedora	
7	Create Device	btrfs subvolume	home	
8	Create Format	btrfs	home	/home
9	Create Device	btrfs subvolume	root	
10	Create Format	btrfs	root	/

Fedora anaconda actions to create a btrfs system

No surprises here — create an ext4 partition for boot and a btrfs partition for root, and further subpartition that for root and home. Let anaconda run through the installation, and everything completes normally.

Once the installation is completed (either or both of the above), I reboot to the installed system and what do I see? Not much, at least for the casual observer. It all looks, feels and works like your typical Linux system. It takes a bit of investigation to see the differences. For example, you can use the *mount* command, and mixed in with a bunch of other lines you will see this:

```
/dev/sda15 on / type btrfs (rw,realtime,space_cache)
```

Aha. Now that we know the name of the root partition, we can use the *btrfs* utility to examine it:

```
linux-6569:/home/jw # btrfs filesystem show /dev/sda15

    Label: 'openSuSE_btrfs'  uuid: 00e10875-995a-4bc6-a36b-7c823474cc85

    Total devices 1 FS bytes used 4.76GiB

    devid    1 size 16.00GiB used 7.04GiB path /dev/sda15

Btrfs v0.20-rc1+20131031
```

OK, that's a bit better... we can see that it is a btrfs filesystem, with a fixed size of 16GB and currently used 4.76GB. But that is still not all the information available, and I remember that the openSuSE installer actions said that it was going to create a bunch of subvolumes within the btrfs filesystem. So see those, we can use the *btrfs subvolumes* command:

```
btrfs subvolume list /home

ID 256 gen 40 top level 5 path boot/grub2/x86_64-efi

ID 258 gen 145 top level 5 path home

ID 259 gen 19 top level 5 path opt

ID 260 gen 20 top level 5 path srv

ID 261 gen 142 top level 5 path tmp

ID 262 gen 49 top level 5 path usr/local

ID 263 gen 29 top level 5 path var/crash

ID 264 gen 144 top level 5 path var/log

ID 265 gen 30 top level 5 path var/opt

ID 266 gen 145 top level 5 path var/spool

ID 267 gen 144 top level 5 path var/tmp

ID 272 gen 94 top level 5 path .snapshots

ID 273 gen 50 top level 5 path .snapshots/1/snapshot

ID 274 gen 51 top level 5 path .snapshots/2/snapshot
```

```
    ID 275 gen 61 top level 5 path .snapshots/3/snapshot

    ID 276 gen 62 top level 5 path .snapshots/4/snapshot

    ID 278 gen 76 top level 5 path .snapshots/5/snapshot

    ID 279 gen 80 top level 5 path .snapshots/6/snapshot

    ID 280 gen 84 top level 5 path .snapshots/7/snapshot

    ID 281 gen 93 top level 5 path .snapshots/8/snapshot
```

Ah, there they all are... and a few extras at the end, which appear to be automatic snapshots that have been made along the way.

So, I'm about ready to call it a day. So far I have been able to create (or convert from ext3 or ext4) a btrfs filesystem, and I have been able to install both openSuSE 13.1 and Fedora 20 with btrfs as the root filesystem. From here, in the next post, I will go on and look at some of the more interesting characteristics and capabilities of btrfs.

Further reading

- Hands on with openSuSE 13.1: Another outstanding release (http://www.zdnet.com/hands-on-with-opensuse-13-1-another-outstanding-release-7000023392/)
- Using Linux to manage my keyboard and mouse with Logitech's Unifying receiver (http://www.zdnet.com/using-linux-to-manage-my-keyboard-and-mouse-with-logitechs-unifying-receiver-7000023041/)
- Testing a new monitor with OpenSuSE, Fedora, Linux Mint, Ubuntu...and Windows 7 (http://www.zdnet.com/testing-a-new-monitor-with-opensuse-fedora-linux-mint-ubuntu-and-windows-7-7000022947/)

Topics: Linux, Open Source, Operating Systems

J.A. Watson

About J.A. Watson

I started working with what we called "analog computers" in aircraft maintenance with the United States Air Force in 1970. After finishing military service and returning to university, I was introduced to microprocessors and machine language programming on Intel 4040 processors. After that I also worked on, operated and programmed Digital Equipment Corporation PDP-8, PDP-11 (/45 and /70) and VAX minicomputers. I was involved with the first wave of Unix-based microcomputers, in the early '80s. I have been working in software development, operation, installation and support since then.

Talkback

Excellent Article

Look forward to it continuing.

Alan Smithie
28 November, 2013 12:27

Reply 8 Votes

Interesting, I'll have to poke around and try to convert my ext4 over...

My eyes want to pronounce it as butterfast because it doesn't seem like a butterface.

Max™
28 November, 2013 12:47

Reply 1 Vote

Bear in mind

You'll need a separate boot partition if you convert whichever partition currently has /boot on.

It also depends on distro - i've been ising btrfs as my default FS on opensuse sinse 12.1 landed two years ago with only one issue (the original 12.1 release cause painfully slow read speeds from my disk - write was fine, but launching an application like firefox was taking 40+seconds on a quad core, the issue went away with an update.)

However BRFS has been adopted at a much slower rate by other distros, for example i probably wouldn't configure an ubuntu instal woth BTRFS

MarknWill
28 November, 2013 16:20

Reply 1 Vote

Specifically mentioned in the article

A couple of things that I was very careful and very specific about in the original post:

- Not all distributions support btrfs for the boot partition. Some use a separate boot of a different type. Some won't work at all.

- I just installed openSuSE 13.1, entirely with btrfs, including the boot directory, with no problems. It works just fine. I was surprised too. I am not aware that any other distribution is capable of this yet, but that might just be a lack of knowledge on my part.

Thanks for reading and commenting.

jw

j.a.watson@...
28 November, 2013 16:32

Reply Vote

/boot on btrfs in Fedora

It nearly works, and it may be fixed for F20 Final. There's just one bug with it, which is currently blocking F20 Final: https://bugzilla.redhat.com/show_bug.cgi?id=864198

If you really want to do it right now there are ways to work around the problem, but it doesn't work cleanly yet.

AdamWill
28 November, 2013 19:52

Reply 1 Vote

Thanks Adam

Hi Adam, thanks for the comment - I almost said in the article that if I was lucky you would come by and fill us in on what Fedora was doing in this area. If it is done in the final release I will definitely include that when I write about it. Personally I don't find the ext4 /boot configuration to be any trouble, and it is an amusing example of the flexibility we have - a btrfs root (/), then mount an ext4 /boot, and if you're on a UEFI system you also mount a FAT /boot/efi. It all works, and /boot and/or /boot/efi are certainly not getting much disk activity once the system is up and running.

Thanks for reading and commenting, as always.

jw

j.a.watson@...
28 November, 2013 20:36

Reply Vote

btrfs root

Ubuntu's been capable of a btrfs-only install since 12.04 at least, possibly further back. There are some papercuts, which I mention in another comment, but it does "just work" in general, even including booting directly to a btrfs-raid root with no separate /boot.

I'd be pretty surprised if the same wasn't true of Debian.

Jim Salter
30 November, 2013 01:47

Reply *1 Vote*

Well, I'll probably leave my / as ext4...

At least until I do another fresh install, already have a different partition I could use for mbr when I swap / over, but for now I'm just thinking of getting my /home swapped over in preparation.

Max™
29 November, 2013 01:48

Reply *Vote*

btrfs and ubuntu (and stuff)

Btrfs and ubuntu go together quite nicely, actually - more nicely (in my opinion) than what's described with the current state of OpenSUSE and Fedora. You can pick btrfs instead of ext4 directly as your root filesystem at install time, and it Just Works - no need for a separate /boot running ext4, or any other silliness.

However. I did mention papercuts - as stated in the article, btrfs is a "rapidly moving target". Which means you want a NEW, NEW kernel, and a NEW, NEW copy of btrfs-tools. So, if you start with Ubuntu 12.04.3-LTS, the first thing you'll want to do is get a 3.11 kernel:

```
apt-get install linux-generic-lts-saucy linux-tools-lts-saucy
```

Next, you'll want to get a much newer copy of btrfs-tools - which, unfortunately, for the moment means grabbing them from Debian Sid, as even the version in Ubuntu Saucy is a few years old.

```
wget http://http.us.debian.org/debian/pool/main/b/btrfs-tools/btrfs-tools_0.19+20130705-3_amd64.deb
dpkg -i btrfs-tools_0.19+20130705-3_amd64.deb
apt-get -f install
```

Now, there's an irritating bug in GRUB2 when you have a btrfs / and no separate /boot that complains about sparse files and invites you to press a key - you don't even need to PRESS the key; the system will boot after several nerve-wracking seconds whether or not you press a key... but it still sucks, and it can camouflage other un-btrfs-related boot problems you might have, so let's fix it. nano /etc/grub.d/00_header, look for the line that says (in part) "save_env recordfail; fi; fi" and comment it out. Now do update-grub, and you won't get the spurious (and bogus) message when you boot anymore.

You can even boot directly to a btrfs-raid0, btrfs-raid1, btrfs-raid5, btrfs-raid6, or btrfs-raid10 root! Just do you normal install to a single btrfs partition, while creating empty partitions of the same size on your other drives; then once the system is installed, if - for example - you have four drives, and you installed to /dev/sda1 and you have identical unused partitions on sda through sdd, and you want to do raid10...

```
btrfs dev add /dev/sdb1 /
btrfs dev add /dev/sdc1 /
btrfs dev add /dev/sdd1 /
btrfs balance start -dconvert=raid10 -mconvert=raid10 /
```

And presto, in a few moments (during which the system remains USABLE!) the root filesystem will, while mounted, magically change itself from a single drive to a four-drive RAID10 array. You can even REbalance it from RAID10 to RAID5 or RAID6 later, if you like!

Jim Salter
30 November, 2013 01:02

Reply Vote

Butterfast?

Why not just call it what it was meant to be called in the first place: "Better File System." ;-)

Steve Kasian
5 April, 2014 08:11

Reply Vote

Great! Thank you!

I am very curious about the new possibilities of btrfs in real life..... Especially your next article is exactly what I was looking for. :-)

pjotr123
28 November, 2013 13:58

Reply 1 Vote

Just about everything

It's easier to say what btrfs _can't_ do, really. For a start, it does everything you're used to using LVM and mdraid for - it does logical volumes and disk spanning.

AdamWill
28 November, 2013 19:52

Reply 1 Vote

Boot with btrfs

[quote] Important! There are a variety of limitations and restrictions on using btrfs for the root (/) and/or boot (/boot) filesystems.

Do not go charging off and convert these kinds of critical filesystems to btrfs, because you will almost certainly be disappointed with the consequences. The kinds of things one might want to convert to btrfs are your home, work, data and other such additional filesystems.

Also, anyone doing this should make sure they have adequate backups of whatever they convert — not only because btrfs is still in development, but also because they might well make a mistake, or get surprised by something, and end up deleting, corrupting or otherwise losing the data in a converted filesystem. [/quote]

Great article!
Good thing you mentioned btrfs can't be used as FS of root or /boot because I did just last week and it just gives critical error and won't boot, for some strange reasons. So I reinstalled a different FS (ext3 or ext4 if I remember correctly) and everything becomes normal again. I am not so sure about the advantages of btrfs as what you've mentioned, but those are typical features of any journaling FS, including NTFS of windows.

Martmarty
28 November, 2013 16:11

Reply 1 Vote

Can't wait for Chapter 2...

Thank you.
BTRFS barely gets mentioned in a lot of Linux heavy-duty books (at least, from the perspective of 2-3 years ago, when I formally studied Linux).
Here you are, saying, "Hey, folks. There's a new file system out there which has great potential. Follow along with me; look over my shoulder as I investigate, and make mistakes as well as progress. Then we'll all know more about btrfs."

Absolutely great stuff, as is very typical of you.

Warmest regards...

"Teaching by example is not simply one way to teach. It's the only way."
--Albert Einstein

thestars
29 November, 2013 16:01

Reply 1 Vote

BTRFS has been in Suse for a while now

I used to use OpenSuse as my primary system, but since then I've seen 3 iterations of Mint including 15.

Suse gave me the option of a BTRFS system as a clean install from a live CD (it was the only option for it, no upgrade or conversion...) so out of interest I tried it out on a brand new HDD. It wasnt difficult at all, installed cleanly and rebooted into it with no problems. I managed to make a separate Home partition with no trouble as well and everything worked as I expected until I tried to move my old EXT3 home directory into it and then all hell broke loose.
It didnt like cross-volume links at all, and although Suse itself worked correctly, everything I had written or modified crashed and burned. I wound up reinstalling the system as EXT4 and the problems resolved themselves. But I never did like Suse all that much and settled on Mint, which as far as I know doesnt support BTRFS (It might now, but I havent looked and I happen to like EXT4 because its very hard to break.)

Nice comprehensive article, particularly for the examples that show how BTRFS handles because I've been referring to it as 'butterfingers'...

Thanks for experimenting, I just gave it a 'meh' and went back to EXTn which I knew was trouble free if a little harder to mess about with. Just one question though, would it be possible to convert an EXT4 boot partition to BTRFS from a Suse live CD, or would this break Grub? I've got a few distros spread over a 1TB drive using the same home partition but I no longer dualboot XP having moved it into VMWare (where it is getting rather dusty I might add!)

SiO2
29 November, 2013 23:17

Reply 1 Vote

Benefits? Huh?

Other than the six vague bullet-points in the article, what are the real, day-to-day and long-term benefits of Btrfs over, say EXT4? I would have liked the article to say at least a little more about the major pros of Btrfs rather than just how to create the filesystem.
Yes, I know there's going to be a follow-up article that "look(s) at some of the more interesting characteristics and capabilities of Btrfs", but to get us interested in reading the sequel it would have been nice to dedicate at least a decent-size paragraph to WHY Btrfs is something we should consider. A "teaser" if you will...
As for pronouncing the acronym:- I like "Butter F S". Sounds slick, tasty and breakfast-y. Like a melting pat of butter on hot morning toast. Mmmm.
I have heard "butter face", but it sounds too much like "butt or face" which either way sounds like some Adult Swim cartoon character or spinoff of Robot Chicken.
If there are some concrete performance/security/etc. benefits I will definitely give it a try, though not necessarily on my /root or /boot. I'd start with my /foot. :P

Steve I.
18 January, 2014 07:57

Reply 1 Vote

Bitrot

Supposedly this file system and ZFS are the first practical filesystems to maintain data integrity down to the last bit. I don't want to rush headlong into unstable software but it's heartbreaking to find bitrot data I so carefully backed up.

http://arstechnica.com/information-technology/2014/01/bitrot-and-atomic-cows-inside-next-gen-filesystems/

Theta-G
8 April, 2014 10:38

Reply Vote