How to: Backup Your Web Hosting (According to the 3–2–1 Principle)

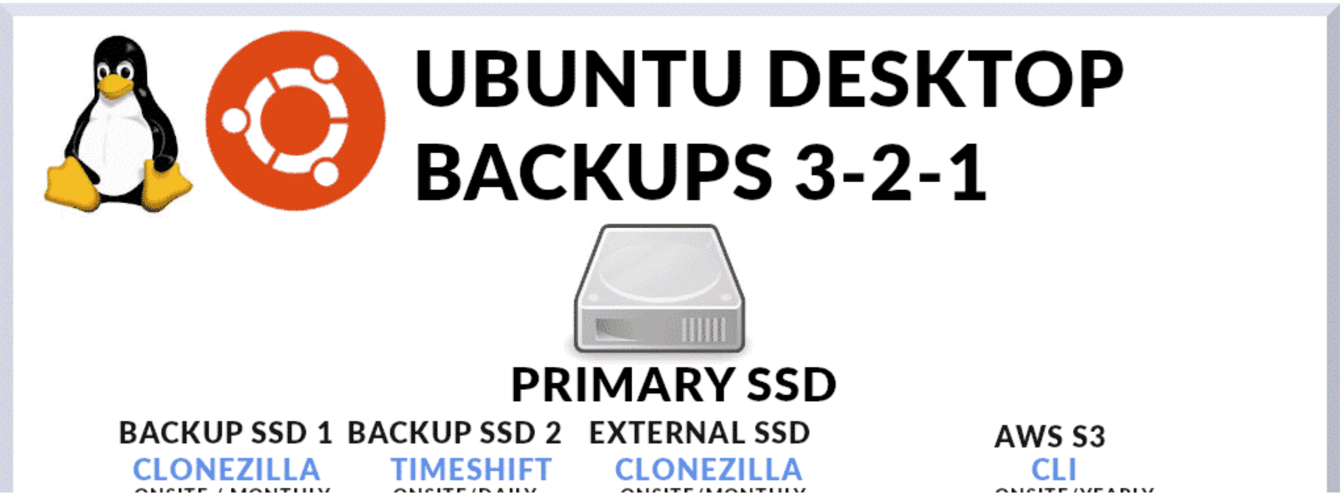
Daniel Rosehill
Apr 20 · 5 min read

In my writings about backups I have emphasized the importance of adhering to the 3–2–1 principle across all your data sources.

That is:

- Keeping **three** copies of any mission-critical data you hold (note: that means the extant copy plus *two* backups).
- Keeping the **two** backup copies on different storage media.
- Ensuring that **one** of those copies is stored off-site.

3-2-1: A Common-Sense Approach For Backing Up Ubuntu - And Keeping It In Good Order
If using Ubuntu, backups are a topic that you should give at least occasional thought to. I use Timeshift as my primary…
linuxhint.com



(You might notice a slight redundancy between the second two points — if one of your two backups has to be stored off-site then it's going to be on different storage media than your first copy).

I also pointed out that if you're concerned about protecting your data then you really need to protect it *everywhere* that it lives — ie, not just the data that lives on your local network. For your typical home computer user that might operate a website and some cloud applications that would includes things like:

— Your web hosting account. The most complete way to back this up is to backup all the Cpanels it contains.

— All data stored on SaaS services, in the cloud, that can be exported for backup (ideally automatically). It's easiest to start with the big tranches (G Suite, Office 365) and then work down to the smaller stuff (do you keep task lists on Todoist? Key accounting documentation in Wave? Ideally you'll want to include all these services in your backup strategy too.)

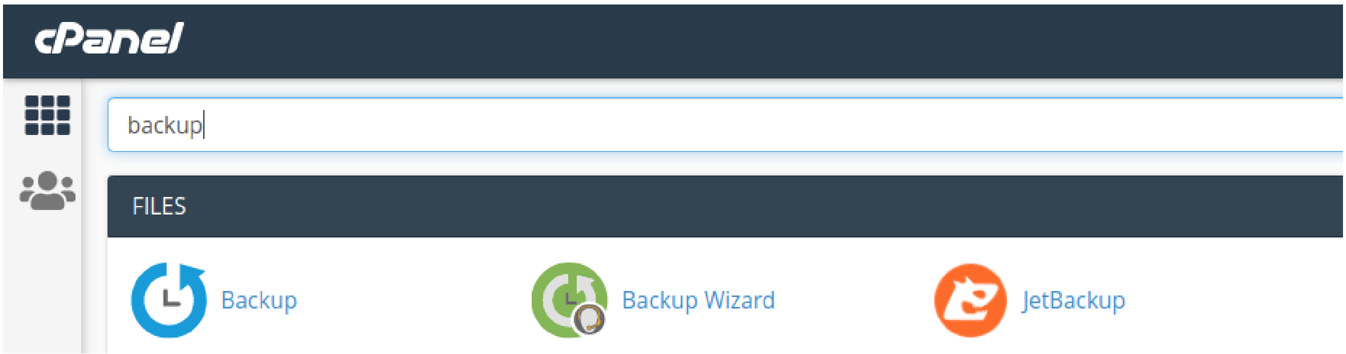
Remember that it's safe to assume that any major SaaS player — and certainly monoliths like AWS — are already backing up your data with redundancy that you could only dream about. But that doesn't mean that you shouldn't play it safe and keep a local copy too. It thankfully happens extraordinarily rarely, but even professionally managed web services *have* lost user data. For the small cost of S3 storage and an external hard drive, this is a cheap insurance policy to hedge against that loss.

In this post, I'll demonstrate the steps needed to take a *full* (not incremental) backup of your Cpanel and how to get it into two cloud storage locations so that your hosting backup meets the 3–2–1 best practice.

I suggest doing this twice a year and retaining two previous (full) backups. It isn't beautiful. It isn't particularly. But it works. And — in the unlikely event that your web host disintegrates into a sea of internet mush overnight — it might just save your sites.

· · ·

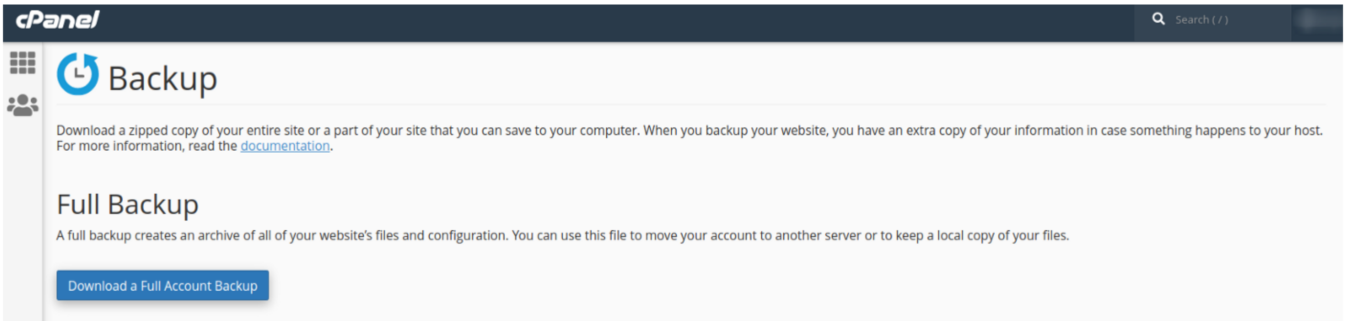
1: Generate a Full Backup In Cpanel



Your exact Cpanel configuration will vary depending on who you're hosting with. This is mine. My host offers JetBackup — which is excellent for rolling the whole Cpanel or parts of it to previous restore points. But for the purpose of keeping a copy of the whole Cpanel for backup purposes I prefer to use the native tool (called simply "Backup").

I take hosting backups with the idea that I might — one day — need to restore the sites, from scratch, on another web server operated by another host. For that reason, I need to capture everything I need to make my websites work. That includes:

- Files
- MySQL databases
- Email addresses, forwarders, and DNS records



Backup

Download a zipped copy of your entire site or a part of your site that you can save to your computer. When you backup your website, you have an extra copy of your information in case something happens to your host. For more information, read the documentation.

Full Backup

A full backup creates an archive of all of your website's files and configuration. You can use this file to move your account to another server or to keep a local copy of your files.

The "full account backup" will capture all these components.

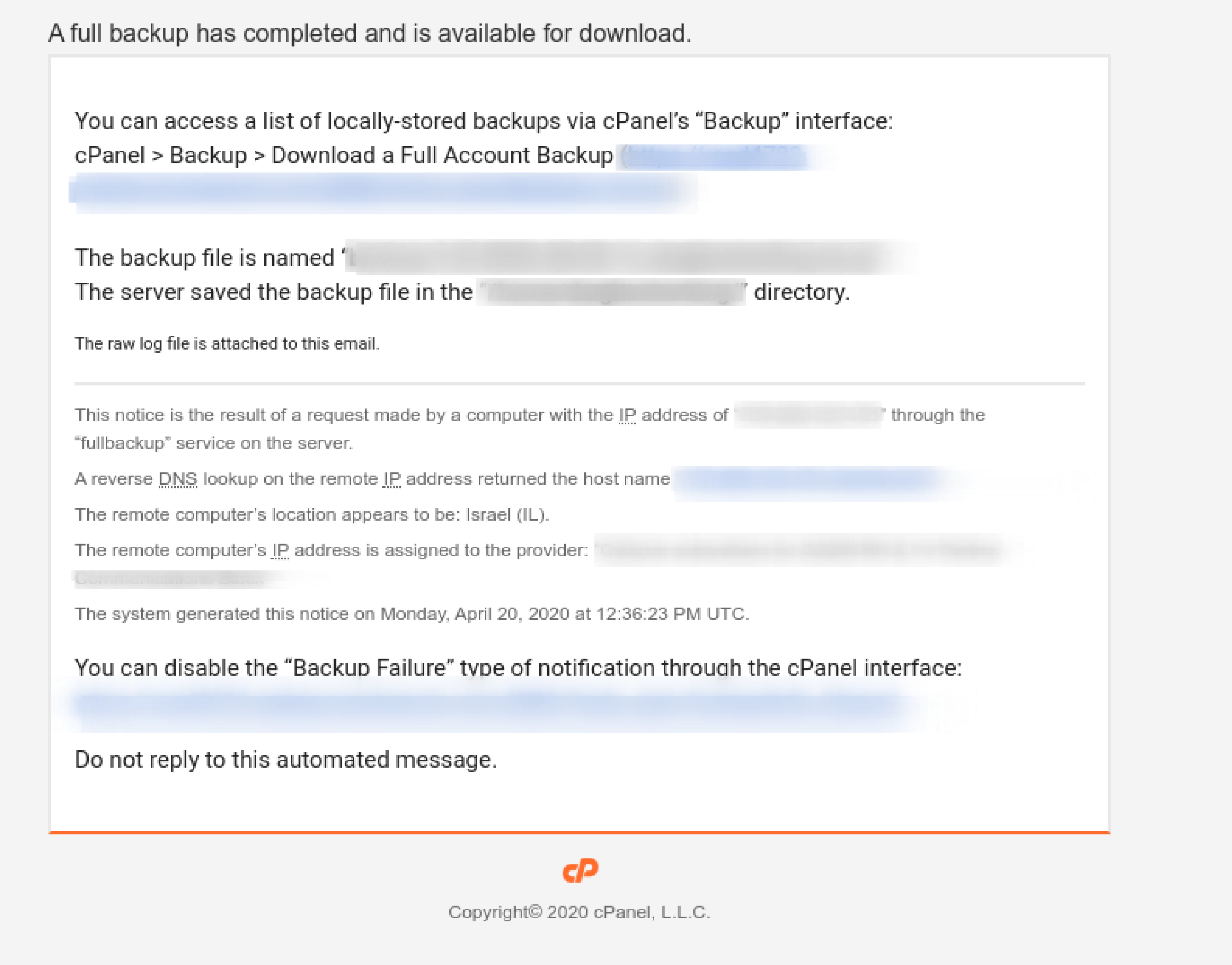
Cpanel's backup creator gives you the option to backup the Cpanel on the folder itself or to run a cross-internet backup over FTP or SCP.

Because I'm copying my backup onto an external hard drive anyway I usually just download the backup and then upload to S3 over DragonDisk or a CLI.

. . .

2: Wait for the Email Notification

Once Cpanel has finished packaging up your archive you will receive an email notification like this:



A full backup has completed and is available for download.

You can access a list of locally-stored backups via cPanel's "Backup" interface:
cPanel > Backup > Download a Full Account Backup [redacted]

The backup file is named "[redacted]
The server saved the backup file in the "[redacted]" directory.

The raw log file is attached to this email.

This notice is the result of a request made by a computer with the IP address of "[redacted]" through the "fullbackup" service on the server.

A reverse DNS lookup on the remote IP address returned the host name [redacted]

The remote computer's location appears to be: Israel (IL).

The remote computer's IP address is assigned to the provider: "[redacted]

The system generated this notice on Monday, April 20, 2020 at 12:36:23 PM UTC.

You can disable the "Backup Failure" type of notification through the cPanel interface:
[redacted]

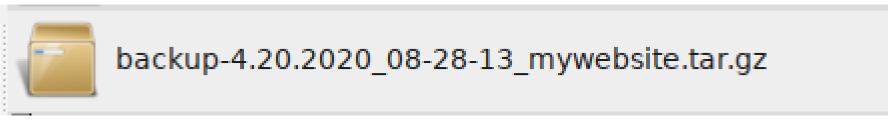
Do not reply to this automated message.

Copyright© 2020 cPanel, L.L.C.

. . .

3: Copy to Local Storage

To create a local backup simply copy the file, packaged as a .tar.gz archive, over to your external hard drive — or on an internal drive on your computer.



backup-4.20.2020_08-28-13_mywebsite.tar.gz

If you generated the backup files on your own server don't forgot to delete the files when you have finished downloading them!

4: Push to the Cloud

If you're using AWS to store your hosting backups on the cloud, then most of your backup archives are going to exceed the 5GB limit for what can be uploaded with one PUT operation. Therefore, a multipart uploading tool is required.

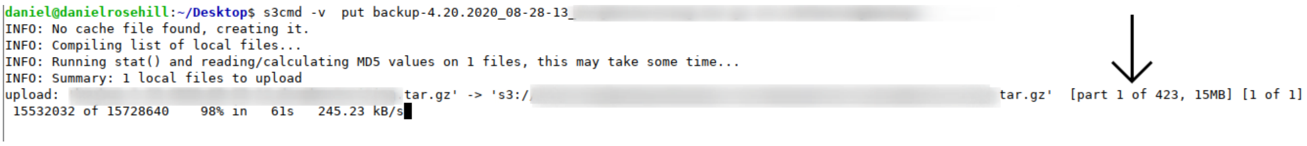
If you're using Linux Ubuntu the s3cmd CLI supports multipart uploads.

Refer to the program documentation for how to initiate a multipart upload. Or you can use:

```
s3cmd -v put mybackup.tar.gz s3://mybackupbucket
```

Because the command can take a long time to run, I suggest adding the verbosity operator (-v) so that you can keep abreast of what is happening.

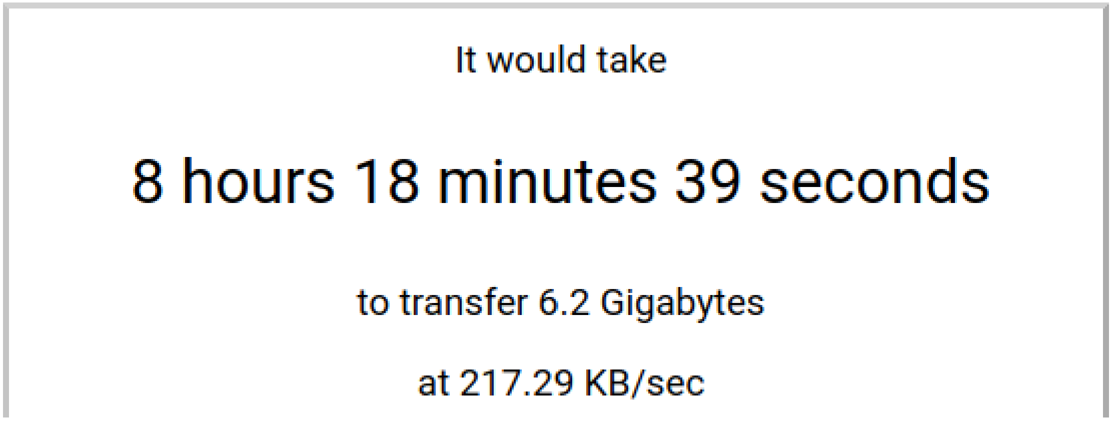
You may need to leave your computer running overnight. You can use this upload time calculator to compute how long the process should take.



```
daniel@danielrosehill:~/Desktop$ s3cmd -v  put backup-4.20.2020_08-28-13_[redacted]
INFO: No cache file found, creating it.
INFO: Compiling list of local files...
INFO: Running stat() and reading/calculating MD5 values on 1 files, this may take some time...
INFO: Summary: 1 local files to upload
upload: [redacted]tar.gz' -> 's3:/[redacted]tar.gz'  [part 1 of 423, 15MB] [1 of 1]
 15532032 of 15728640    98% in   61s   245.23 kB/s
```

As can be seen above, after I initiated the 'put' command the multipart uploader has divided my archive into 423 files.

And my upload time:



It would take

8 hours 18 minutes 39 seconds

to transfer 6.2 Gigabytes

at 217.29 KB/sec

The process might be long and a little tedious. But going through it twice a year is a great means of ensuring that keep safe copies of any websites and cloud data that you maintain.

Technology Remote Backups AWS Cloud