

Efficiently Backup a WordPress website using 3-2-1 Strategy

Learn how to protect your WordPress site from data loss with the 3-2-1 backup strategy. Discover efficient methods to secure your content, themes, and plugins.

Contents

01	Introduction	3
02	Prerequisites	3
03	Create a Full WordPress Backup	4
04	Implement the 3-2-1 WordPress Backup Strategy	8
05	Automate the WordPress 3-2-1 Backup Strategy	15
06	Conclusion	19
07	Next Steps	19

Introduction

Building a reliable and good-ranking WordPress site is a time consuming task that takes a lot of human resources to achieve. However, losing WordPress data is as fast as a single site crash, security attack, or server failure. This means you can lose a 10-year-old high-ranking WordPress site in 10 minutes. To solve this, WordPress backups ensure that you can recover your site in all situations.

A 3-2-1 backup strategy is a data backup and recovery method that creates multiple copies of your WordPress site to ensure safety and availability if a data loss or site crashes occurs. Unlike regular backups, a 3-2-1 strategy involves abiding by multiple storage methods to keep your WordPress backups recoverable and up to date.

This article explains how you can efficiently back up a WordPress website using a 3-2-1 strategy. You are to create multiple files and database copies, store them on different media, and automate the procedure to keep your WordPress backups up to date.

Prerequisites

Before you begin, be sure to:

- Deploy a [OneClick WordPress server](#) using the Vultr marketplace application

If you have an existing WordPress instance, make sure you have terminal access to perform backup tasks

- Use [SSH](#), access the WordPress server as a [non-root user with sudo privileges](#)
- Switch to the non-root sudo user account. For example `webadmin`

```
# su webadmin
```

> This article uses the example values `webadmin` and the WordPress domain `example`. Replace the values with your actual WordPress server details

Create a Full WordPress Backup

A full WordPress backup consists of the data application files and a copy of the full database including the tables and table data. In this section, perform a full WordPress backup to implement a 3-2-1 strategy as described below.

Back Up WordPress Files

1. Find and switch to the WordPress web root directory. For example `/var/www/html/`

```
$ cd /var/www/html
```

If you're unsure of the WordPress directory, view your web server virtual host configuration. For example, for Nginx, check the `/etc/nginx/sites-available/` directory

```
$ cd /etc/nginx/sites-available
```

Run the following command to search your WordPress domain name in all available files and find the exact configuration file

```
$ grep -r "example.com" /etc/nginx/sites-available
```

View the WordPress configuration file and take note of the `root` directory path

```
root /var/www/example.com;
```

2. Switch to the directory

```
$ cd /var/www/example.com
```

3. List the directory and verify that all WordPress files are available

```
$ ls -l
```

Output:

```
-rw-r--r--  1 www-data www-data  7211 Aug  8 20:26 wp-activate.php
drwxr-xr-x  9 www-data www-data  4096 Aug  8 20:26 wp-admin
-rw-r--r--  1 www-data www-data   351 May  9 03:17 wp-blog-header.php
-rw-r--r--  1 www-data www-data  2323 Aug  8 20:26 wp-comments-post.php
-rw-r--r--  1 www-data www-data  3360 Jul 27 14:42 wp-config.php
-rw-r--r--  1 www-data www-data  3013 May  9 03:17 wp-config-sample.php
drwxr-xr-x 10 www-data www-data  4096 Aug 27 20:53 wp-content
-rw-r--r--  1 www-data www-data  5638 Aug  8 20:26 wp-cron.php
drwxr-xr-x 27 www-data www-data 16384 Aug  8 20:26 wp-includes
```

4. To make a backup of the WordPress files. Switch to the parent directory

```
$ cd ..
```

5. Copy the WordPress files to a separate directory such as your user home with a different filename such as `wp-files-backup`

```
$ sudo cp -Rp example.com /home/webadmin/wp-files-backup
```

The above command makes a copy of all WordPress files and preserves all permissions to your user home directory with the new directory name `wp-files-backup`. To uniquely identify your backup file, add the server timestamp to the filename while copying the files as below:

6. Switch to your user home directory and verify that the WordPress files are available

```
$ cd /home/webadmin/
```

7. List files in the directory

```
$ ls -l
```

Output:

```
drwxr-xr-x 7 www-data www-data 4096 Aug 27 20:53 wp-files-backup
```

8. To safely store the WordPress files backup, archive the directory using a compression format such as gunzip to create a `.tar.gz` file

```
$ tar czvf wp-files-backup.tar.gz wp-files-backup
```

When you archive and compress the WordPress files backup directory, the file size reduces, and file safety improves with little to no changes in the archive format unless extracted.

9. Long list files and verify that the `wp-files-backup.tar.gz` is available in your directory

```
$ ls -l
```

Back Up the WordPress Database

1. View your WordPress `wp-config.php` file

```
$ cat wp-files-backup/wp-config.php
```

2. Find the following section

```
// ** Database settings - You can get this info from your web host ** //  
/** The name of the database for WordPress */  
define( 'DB_NAME', 'wordpressdb' );  
  
/** Database username */  
define( 'DB_USER', 'wpuser' );  
  
/** Database password */  
define( 'DB_PASSWORD', 'user-password' );
```

```
/** Database hostname */  
define( 'DB_HOST', 'localhost' );
```

Keep note of the database name in the `DB_NAME` directive, the database user, and password used to connect WordPress to your database.

3. Log in to your MySQL database server using the WordPress database user and password

```
$ mysql -u wpuser -p
```

When prompted, enter the database user password

4. Verify that the WordPress database is available

```
mysql> SHOW DATABASES;
```

5. Exit the MySQL console

```
mysql> EXIT
```

6. Using `mysqldump`, export the WordPress database to a readable file such as `wpdatabase.sql`

```
$ mysqldump -u wpuser -p wordpressdb > wpdatabase.sql
```

7. When successful, list your directory files and verify that a new `wpdatabase.sql` file is available

```
$ ls -l
```

8. To avoid making changes to the file, archive and compress the file to a format such as `.tar.gz`

```
$ tar czvf wpdatabase.tar.gz wpdatabase.sql
```

9. Verify that the database archive file is available in your directory

```
$ ls
```

10. To organize your full WordPress backup files, create a new directory named `fullwpbackup`

```
$ mkdir -p fullwpbackup
```

11. Move all WordPress backup archive files to the directory

```
$ mv *.tar.gz fullwpbackup/
```

12. List files and verify that only archive files are available in the directory

```
$ ls fullwpbackup/
```

Perform WordPress Backups using a Plugin

To perform WordPress backups using your administrator dashboard, install a compatible plugin and use it to create a full backup file that includes the database and WordPress files. Many backup plugins are available in the WordPress repository, depending on your choice, visit the following resources to perform backups using a WordPress plugin.

- [How to Back Up WordPress to Vultr Object Storage with UpdraftPlus](#)
- [How to Migrate your WordPress Site to Vultr with BackupBuddy](#)
- [How to Migrate your WordPress Site to Vultr with Duplicator](#)

Implement the 3-2-1 WordPress Backup Strategy

A 3-2-1 backup strategy follows the standard below:

- 3 different backup copies
- 2 different storage media types
- 1 off-site or offline copy of the backup files

By following the strategy correctly, your WordPress backup files are safe from any possible form of data corruption. In case your WordPress site crashes or suffers a massive security breach, you can roll back changes to the last stable state and work out solutions to prevent a similar scenario from happening again.

In this section, implement the 3-2-1 WordPress backup storage on your server as described in the steps below.

Generate 3 different WordPress Backup Copies

To implement the 3 backup storage types method, save multiple copies of the backup files and apply different compression mechanisms. For example, keep a single `.tar.gz` copy, a `.zip` variant, and a non-archived backup copy as described in the following steps.

1. Switch to your user home directory

```
$ cd /home/webadmin/
```

2. Long list files and verify that your `wpdatabase.sql` file, `fullwpbackup` and `wp-files-backup` directories are available

```
$ ls -l
```

Output:

```
drwxrwxr-x 2 webadmin webadmin    4096 Aug 29 16:40 fullwpbackup
-rw-rw-r-- 1 webadmin webadmin 19499776 Aug 29 16:39 wpdatabase.sql
drwxr-xr-x 7 webadmin webadmin    4096 Aug 27 20:53 wp-files-backup
```

3. Create the first full backup archive `.tar.gz` archive file

```
$ tar czvf fullwpbackup.tar.gz fullwpbackup
```

The `fullwpbackup.tar.gz` includes files with the same `.tar.gz` archive format. To apply a different format, it must match the files to avoid any forms of data corruption

4. Create a second full backup archive using the `.zip` format, `wpdbatabase.sql` file and `wp-files-backup` directory

```
$ zip -r fullwpbackup.zip wpdatabase.sql wp-files-backup
```

5. Long list files to verify that both `.tar.gz` and `.zip` WordPress archive files are available in the directory

```
$ ls -l
```

Output:

```
-rw-rw-r-- 1 webadmin webadmin 113366047 Aug 29 16:53 fullwpbackup.tar.gz
-rw-rw-r-- 1 webadmin webadmin 127738982 Aug 29 16:46 fullwpbackup.zip
```

6. The original `wp-files-backup` and `wpdbatabase.sql` apply as the third backup copies. To create another archive, use a different format such as `tar.xz`

```
$ tar cvf 3rdwpbackup.tar wp-files-backup wpdatabase.sql
```

Compress the file

```
$ xz 3rdwpbackup.tar
```

You have implemented three WordPress backup copies through which you can recover your WordPress site in case of any failure. To safeguard the copies, separate them into different storage volumes.

Store WordPress backups on 2 Different Storage Volumes

When your server crashes, any attached secondary storage volumes can function normally and you can re-attach them to your server when troubleshooting is complete. In this section, attach a Vultr block storage volume

to your server and store your WordPress backups to two different volumes as described below.

- Deploy a [Vultr Block Storage volume](#)
- [Attach the Volume to your server](#) as `backupdisk` in the `/media/` directory

1. Switch to the Vultr Block Storage volume directory

```
$ cd /media/backupdisk/
```

2. Create a new file `hellobackup.txt` to verify that you can write to the disk

```
$ touch hellobackup.txt
```

3. When created, switch back to your user home directory

```
$ cd /home/webadmin/
```

4. Create a new directory `WP-Backup`

```
$ mkdir -p WP-Backup
```

5. Copy the WordPress `.tar.gz` and `.zip` archive files to the directory

```
$ cp *.tar.gz *.zip WP-Backup/
```

6. Copy `wp-files-backup` and `wpdatabase.sql` to the directory

```
$ cp -r wp-files-backup/ wpdatabase.sql WP-Backup/
```

7. List the directory files and verify that all copies are available

```
$ ls WP-Backup/
```

Output:

```
fullwpbackup.tar.gz fullwpbackup.zip wpdatabase.sql wp-files-backup
```

8. Copy the WP-Backup directory to the Vultr block storage volume

```
$ cp -r WP-Backup /media/backupdisk
```

To append the server timestamp on the directory name, adjust the command to:

```
$ cp -r WP-Backup /media/backupdisk/WP-Backup$(date +%Y%m%d')
```

9. List the volume files and verify that the directory is available

```
$ ls /media/backupdisk
```

You have applied the two storage volumes backup method. The WordPress backup files are available on your server storage volume and the attached Vultr Block Storage volume. In case of any server failure, you can recover WordPress files by accessing the block storage volume.

Keep 1 Off-Site WordPress backup Copy

Computing instances often crash in a batch, in case your hosted server and block storage volumes fail, you can recover your WordPress site using a local backup file. Or, you can apply an S3-compatible Vultr Object Storage instance to work as an off-site destination. In this case, your WordPress backup files become available in different locations than your server.

Depending on your WordPress backup file size, implement the off-site storage method as described in the steps below.

Download WordPress backup files to your Computer

You can download your WordPress backup files to your computer using various file transfer methods such as FTP, SFTP, or SCP. When the transfer is successful, a local copy of your WordPress backup files is available on your computer and you can recover it by re-uploading the files to the server.

1. Open a new terminal window

2. Establish an SFTP connection to your server

```
$ sftp webadmin@SERVER-IP
```

3. List files in the working directory

```
> ls
```

Verify that the `WP-Backup` directory is available

4. Download the `WP-Backup` directory to your computer

```
> get -r WP-Backup
```

5. Monitor the download progress, when complete, end the SFTP connection

```
> exit
```

6. Using your file explorer, find and open the `WP-Backup` directory on your computer

7. Verify that all WordPress backup copies are available

Upload WordPress backup files to Vultr Object Storage

Vultr Object Storage is an S3-compatible instance that is fully independent from your server infrastructure. You can create buckets, and upload your WordPress backup files for recovery in case of any server failure. Implement the off-site backup method with Vultr Object Storage as described in the steps below.

- Deploy a [Vultr Object Storage instance](#) and copy the **Access Key** and **Secret Key** from the instance dashboard.
- Install s3cmd on the server

```
$ sudo apt install s3cmd
```

- [Configure s3cmd with your Vultr Object Storage endpoint](#), **Access** and **Secret** keys

1. In your server SSH session, list the available buckets linked to your Vultr Object storage

```
$ s3cmd ls
```

If no configuration is available, run `s3cmd --configure` to connect to your Vultr Object Storage

```
$ s3cmd --configure
```

2. Create a new bucket named `wordpress-backups`

```
$ s3cmd mb s3://wordpress-backups
```

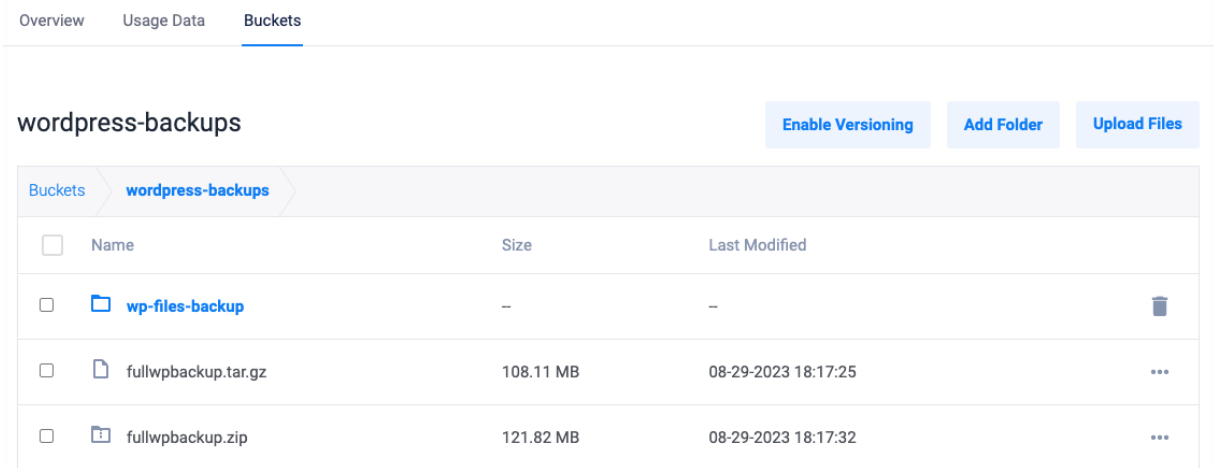
3. Upload the `WP-Backup` directory to the `wordpress-backups` bucket

```
$ s3cmd put -r WP-Backup/ s3://wordpress-backups/
```





4. When the upload is complete, list all objects in the `wordpress-backups` bucket to verify that the transfer is complete

```
$ s3cmd ls s3://wordpress-backups
```

Or, navigate to your Vultr Object Storage control panel and view the bucket objects



The screenshot shows the AWS S3 console interface for a bucket named 'wordpress-backups'. At the top, there are tabs for 'Overview', 'Usage Data', and 'Buckets'. Below the tabs, the bucket name 'wordpress-backups' is displayed, along with three buttons: 'Enable Versioning', 'Add Folder', and 'Upload Files'. A table lists the contents of the bucket:

<input type="checkbox"/>	Name	Size	Last Modified	
<input type="checkbox"/>	 wp-files-backup	-	-	
<input type="checkbox"/>	 fullwpbackup.tar.gz	108.11 MB	08-29-2023 18:17:25	...
<input type="checkbox"/>	 fullwpbackup.zip	121.82 MB	08-29-2023 18:17:32	...

Verify that the `WP-Backup` directory files are available in your `wordpress-backups` bucket

5. Delete the `WP-Backup` directory to free space on your main server storage volume

```
$ rm -r WP-Backup/
```

You have implemented the off-site backup method to your WordPress backup files. To further widen your off-site implementation, you can upload copies of your WordPress backup to public storage services such as Google Drive, Dropbox, Mega, among others.

Automate the WordPress 3-2-1 Backup Strategy

Running the WordPress 3-2-1 backup strategy is a time consuming process. To save time and automate the process, but keep monitoring any changes, use CronJob to schedule automatic tasks to backup your WordPress database, and files as described below.

1. Using a text editor such as `nano`, create a new `auto-backup.sh` script in your user home directory.

```
$ nano auto-backup.sh
```

2. Add the following configurations to the file

```
#!/bin/bash

echo "##### This script implements the 3-2-1 WordPress backup strategy to keep
your backup files updated every once a week#####"

# The base Example User directory
base_dir="/home/webadmin/"

# Create the auto-backups directory if does not exist
if [ ! -d "$base_dir/auto-backups" ]; then
    mkdir "$base_dir/auto-backups"
fi

# Backup the WordPress file directory
timestamp=$(date +%Y%m%d)
backup_dir="$base_dir/auto-backups/wp-files-backup_$timestamp"
cp -r /var/www/example.com "$backup_dir"

# Backup the wordpressdb database
mysqldump -u wpuser -p wpuser-password wordpressdb > "$base_dir/auto-backups/
wpdatabase.sql"

# Switch to the auto-backups directory
cd "$base_dir/auto-backups"

# Delete the existing fullwpbackup directory and recreate it
if [ -d "fullwpbackup" ]; then
    rm -rf fullwpbackup
fi
mkdir fullwpbackup

# Create the latest tar.gz archive for the wp-files-backup directory
tar czf "fullwpbackup/wp-files-backup_$timestamp.tar.gz" "wp-files-
backup_$timestamp"

# Create the latest tar.gz archive for database.sql
tar czf "fullwpbackup/database_$timestamp.tar.gz" "wpdatabase.sql"
```

```
# Create the latest tar.gz archive for fullwpbackup with a server timestamp
tar czf "fullwpbackup/fullwpbackup_$(date +%Y%m%d%H%M%S).tar.gz" fullwpbackup

# Create a new .zip archive for wpdatabase.sql and wp-files-backup files
zip -r "fullwpbackup/wp-files-db-$(date +%Y%m%d%H%M%S).zip" "wp-files-backup_$(date +%Y%m%d%H%M%S)"
wpdatabase.sql

# List files in the auto-backups directory
ls -lh

# 2 different storage volumes

# Check if the Vultr Block Storage backupdisk directory is available
if [ ! -d "/media/backupdisk" ]; then
    echo "The Vultr Block Storage volume is not available with the /media/
backupdisk path, please attach a volume or apply the correct path"
    exit 1
fi

#Off-site storage

# Create the WP-Backup directory with the current server date stamp
backup_dir="/media/backupdisk/WP-Backup_$(date +%Y%m%d%H%M%S)"
mkdir "$backup_dir"

# Copy .tar.gz and .zip files to the WP-Backup directory
cp *.tar.gz *.zip "$backup_dir"

# Use s3cmd to upload WP-Backup directory to s3
s3cmd put -r "$backup_dir" s3://wordpress-backups/

echo "#####The WordPress 3-2-1 Backup process is successfully Completed!!!!!!
#####"

# Delete the WP-Backup directory
if [ -d "$base_dir/auto-backups/WP-Backup_$(date +%Y%m%d%H%M%S)" ]; then
    rm -rf "$base_dir/auto-backups/WP-Backup_$(date +%Y%m%d%H%M%S)"
fi

#OldFiles

# Delete files older than 2 weeks from the auto-backups directory
find "$base_dir/auto-backups" -type f -name "*.tar.gz" -mtime +14 -exec rm {} \;
find "$base_dir/auto-backups" -type f -name "*.zip" -mtime +14 -exec rm {} \;
```

Save and close the file

The above script creates the latest full WordPress backup that includes the CMS files and database. The full backup exports to the `auto-backups` directory, then, 2 archive format copies generate from the files to make 3 available copies.

The `# 2 different storage volumes` section exports the generated `WP-Backup` directory to your Vultr block storage volume and keeps the original copy available on your server.

In the `#off-site storage` section, the `s3cmd` tool uploads a copy of the `WP-Backup` directory to your Vultr Object Storage bucket. When the upload completes successfully without any error, the `WP-Backup` gets deleted from the main server storage to free up space.

Files older than 2 weeks are auto-deleted from the directory to create space for newer backups. To increase the age of deleted files, edit the `oldFiles` section values.

3. Run the above script to verify that it works correctly.

```
$ bash auto-backup.sh
```

4. When complete, view files in the `auto-backups` directory

```
$ ls auto-backups/
```

Verify that all files include the server time stamp to create unique backup files

5. To automate the above script to run every week, edit your `crontab` configuration

```
$ crontab -e
```

6. Add the following directive to the file

```
0 1 * * 0 /bin/bash /home/webadmin/auto-backup.sh
```

Save and close the file

The above Cron task runs the `auto-backup.sh` script every Sunday at 01:00. You can change the time and date to match a date with low WordPress site activity to create the latest weekly backup copy. For more information, visit the [how to use the Cron Task Scheduler](#) resource.

Conclusion

In this article, you implemented a WordPress 3-2-1 backup strategy that creates a good disaster recovery plan. Depending on your preferences, you must ensure that your WordPress backup files are up to date to clear any errors that may result from updates or security breaches on your site.

Next Steps

WordPress is a highly extensible CMS, to implement more solutions on your server, visit the following resources:

- [How to Migrate your WordPress Site to Vultr with Jetpack](#)
- [How to Migrate your WordPress Site to Vultr with UpdraftPlus](#)
- [Install WordPress on LAMP with Ubuntu 20.04 LTS](#)
- [Manage WordPress With WordOps](#)
- [How to Migrate WordPress to Vultr](#)
- [How to Migrate WordPress to Vultr without Downtime](#)
- [How to Deploy WordPress on Vultr Kubernetes Engine](#)



VULTR

