

How to use Rclone with Vultr Object Storage

Learn how to configure and use Rclone with Vultr Object Storage for efficient file synchronization, backup, and management across your cloud storage systems.

Contents

01	Introduction	3
02	Prerequisites	3
03	Configuration	3
04	Testing	4
05	Sync Two Buckets	5
06	Sync Two Different Locations	6
07	Troubleshooting	7
08	More Information	8

Introduction

Rclone is a command-line program to manage files on cloud storage, with cloud equivalents to Unix commands like `rsync`, `cp`, `mv`, `mount`, `ls`, and more. This guide describes how to configure rclone for Vultr Object Storage.

Prerequisites

- Install rclone for your platform, either [manually](#), with [Chocolatey on Windows](#), or [Homebrew on macOS and Linux](#).
- Create a Vultr Object Storage [bucket](#). This guide uses the example `testbucket`.

The rclone commands are cross-platform. This guide shows examples for macOS in zsh, which are compatible with Linux in bash. Windows users will make minor command adjustments for PowerShell or CMD.

Configuration

Locate your rclone configuration file location. This example path below comes from a macOS installation.

```
$ rclone config file
Configuration file doesn't exist, but rclone will use this path:
/Users/example/.config/rclone/rclone.conf
```

Create a configuration file.

```
$ nano /Users/example/.config/rclone/rclone.conf
```

Paste the following to the file. Replace `access_key_id` and `secret_access_key` with your values.

```
[vultr-ewr1]
type = s3
provider = Vultr
env_auth = false
access_key_id = YOUR_ACCESS_KEY
secret_access_key = YOUR_SECRET_ACCESS_KEY
region =
endpoint = https://ewr1.vultrobjects.com
location_constraint =
acl = private
server_side_encryption =
storage_class =
```

The `acl =` line sets the default file permission.

- **private** : Owner gets **full control**. No one else has access. This permission is the default.
- **public-read** : Owner gets **full control**. Everyone has **read** access.

In general, use **private** unless you share files via URL. [Read more about s3 permissions in the rclone documentation.](#)

Testing

Create a test folder and change to it.

```
$ mkdir tmp
$ cd tmp
```

Assuming the `tmp` folder is empty, create three test files.

```
$ echo "hello rclone" > test1.txt
$ echo "hello rclone" > test2.txt
$ echo "hello rclone" > test3.txt
```

Copy one file to the bucket.

```
$ rclone copy test1.txt vultr-ewr1:testbucket
```

List the bucket. Notice that test1.txt exists in the bucket and is 13 bytes.

```
$ rclone ls vultr-ewr1:testbucket
13 test1.txt
```

Delete `test1.txt` from the local system.

```
$ rm test1.txt
```

List the bucket. Notice that test1.txt still exists.

```
$ rclone ls vultr-ewr1:testbucket
13 test1.txt
```

Sync the entire folder to the bucket.

```
$ rclone sync . vultr-ewr1:testbucket
```

List the bucket. Notice that rclone synced the bucket with the local system by removing `test1.txt` and adding the other two files.

```
$ rclone ls vultr-ewr1:testbucket
13 test2.txt
13 test3.txt
```

Sync Two Buckets

In addition to cloning local files to object Storage, rclone can work with two different object storage buckets. Assuming you had two buckets in the `vultr-ewr1` object store shown previously, you'd do this:

```
$ rclone sync vultr-ewr1:my_bucket1 vultr-ewr1:my_bucket2
```

If you have multiple stores with different keys, declare them in `rclone.conf` with different labels. You can also use different permissions in the `acl=` line of each.

Here are two different object stores in the New Jersey location with different permissions.

```
[vultr-ewr1-private]
type = s3
provider = Vultr
env_auth = false
access_key_id = SAMPLE_KEY1
secret_access_key = SAMPLE_SECRET_KEY1
region =
endpoint = https://ewr1.vultrobjects.com
location_constraint =
acl = private
server_side_encryption =
storage_class =

[vultr-ewr1-public]
type = s3
provider = Vultr
env_auth = false
access_key_id = SAMPLE_KEY2
secret_access_key = SAMPLE_SECRET_KEY2
region =
endpoint = https://ewr1.vultrobjects.com
location_constraint =
acl = public-read
server_side_encryption =
storage_class =
```

To sync `bucket1-private` with `bucket2-public`, use rclone like this:

```
$ rclone sync vultr-ewr1-private:my_private_bucket vultr-ewr1-public:my_public_bucket
```

Sync Two Different Locations

You can also sync buckets between locations. For example, here is a sample configuration with New Jersey (ewr1) and Amsterdam (ams1).

```
[ewr1]
type = s3
```

```
provider = Vultr
env_auth = false
access_key_id = SAMPLE_EWR1_ACCESS_KEY
secret_access_key = SAMPLE_EWR1_SECRET_KEY
region =
endpoint = https://ewr1.vultrobjects.com
location_constraint =
acl = private
server_side_encryption =
storage_class =

[ams1]
type = s3
provider = Vultr
env_auth = false
access_key_id = SAMPLE_AMS1_ACCESS_KEY
secret_access_key = SAMPLE_AMS1_SECRET_KEY
region =
endpoint = https://ams1.vultrobjects.com
location_constraint =
acl = public-read
server_side_encryption =
storage_class =
```

To sync files between the locations, you'd use rclone like this:

```
$ rclone sync ewr1:my_new_jersey_bucket ams1:my_amsterdam_bucket
```

You'll find a full list of our locations and hostnames in our [Object Storage documentation](#).

Troubleshooting

If you encounter HTTP error codes 429, 503, or 504, you may have hit the [Vultr Object Storage rate limit](#). If you encounter this issue, we recommend the following command line options.

- `--tpslimit 100`
 - [Limit transaction per second to 100](#).

- `--transfers 1`
 - [This prevents parallel file transfers.](#)

More Information

Please refer to the [rclone documentation](#) for more information.



VULTR

