Backblaze B2

B2 is Backblaze's cloud storage system.

Paths are specified as `remote:bucket` (or `remote:` for the `lsd` command.) You may put subdirectories in too, e.g. `remote:bucket/path/to/dir`.

Here is an example of making a b2 configuration. First run

```
rclone config
```

This will guide you through an interactive setup process. To authenticate you will either need your Account ID (a short hex number) and Master Application Key (a long hex number) OR an Application Key, which is the recommended method. See below for further details on generating and using an Application Key.

```
No remotes found - make a new one
n) New remote
q) Quit config
n/q> n
name> remote
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Backblaze B2
   \ "b2"
[snip]
Storage> b2
Account ID or Application Key ID
account> 123456789abc
Application Key
key> 0123456789abcdef0123456789abcdef0123456789
Endpoint for the service - leave blank normally.
endpoint>
Remote config
--------------------
[remote]
account = 123456789abc
key = 0123456789abcdef0123456789abcdef0123456789
endpoint =
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

This remote is called `remote` and can now be used like this

See all buckets

```
rclone lsd remote:
```

Create a new bucket

```
rclone mkdir remote:bucket
```

List the contents of a bucket

```
rclone ls remote:bucket
```

Sync `/home/local/directory` to the remote bucket, deleting any excess files in the bucket.

```
rclone sync -i /home/local/directory remote:bucket
```

Application Keys

B2 supports multiple Application Keys for different access permission to B2 Buckets.

You can use these with rclone too; you will need to use rclone version 1.43 or later.

Follow Backblaze's docs to create an Application Key with the required permission and add the `applicationKeyId` as the `account` and the `Application Key` itself as the `key`.

Note that you must put the *applicationKeyId* as the `account` – you can't use the master Account ID. If you try then B2 will return 401 errors.

--fast-list

This remote supports `--fast-list` which allows you to use fewer transactions in exchange for more memory. See the rclone docs for more details.

Modified time

The modified time is stored as metadata on the object as `X-Bz-Info-src_last_modified_millis` as milliseconds since 1970-01-01 in the Backblaze standard. Other tools should be able to use this as a modified time.

Modified times are used in syncing and are fully supported. Note that if a modification time needs to be updated on an object then it will create a new version of the object.

Restricted filename characters

In addition to the default restricted characters set the following characters are also replaced:

Character	Value	Replacement
\	0x5C	\

Invalid UTF-8 bytes will also be replaced, as they can't be used in JSON strings.

Note that in 2020-05 Backblaze started allowing \ characters in file names. Rclone hasn't changed its encoding as this could cause syncs to re-transfer files. If you want rclone not to replace \ then see the `--b2-encoding` flag below and remove the `BackSlash` from the string. This can be set in the config.

SHA1 checksums

The SHA1 checksums of the files are checked on upload and download and will be used in the syncing process.

Large files (bigger than the limit in `--b2-upload-cutoff`) which are uploaded in chunks will store their SHA1 on the object as `X-Bz-Info-large_file_sha1` as recommended by Backblaze.

For a large file to be uploaded with an SHA1 checksum, the source needs to support SHA1 checksums. The local disk supports SHA1 checksums so large file transfers from local disk will have an SHA1. See the overview for exactly which remotes support SHA1.

Sources which don't support SHA1, in particular `crypt` will upload large files without SHA1 checksums. This may be fixed in the future (see #1767).

Files sizes below `--b2-upload-cutoff` will always have an SHA1 regardless of the source.

Transfers

Backblaze recommends that you do lots of transfers simultaneously for maximum speed. In tests from my SSD equipped laptop the optimum setting is about `--transfers 32` though higher numbers may be used for a slight speed improvement. The optimum number for you may vary depending on your hardware, how big the files are, how much you want to load your computer, etc. The default of `--transfers 4` is definitely too low for Backblaze B2 though.

Note that uploading big files (bigger than 200 MB by default) will use a 96 MB RAM buffer by default. There can be at most `--transfers`of these in use at any moment, so this sets the upper limit on the memory used.

Versions

When rclone uploads a new version of a file it creates a new version of it. Likewise when you delete a file, the old version will be marked hidden and still be available. Conversely, you may opt in to a "hard delete" of files with the `--b2-hard-delete` flag which would permanently remove the file instead of hiding it.

Old versions of files, where available, are visible using the `--b2-versions` flag.

NB Note that `--b2-versions` does not work with crypt at the moment #1627. Using --backup-dir with rclone is the recommended way of working around this.

If you wish to remove all the old versions then you can use the `rclone cleanup remote:bucket` command which will delete all the old versions of files, leaving the current ones intact. You can also supply a path and only old versions under that path will be deleted, e.g. `rclone cleanup remote:bucket/path/to/stuff`.

Note that `cleanup` will remove partially uploaded files from the bucket if they are more than a day old.

When you `purge` a bucket, the current and the old versions will be deleted then the bucket will be deleted.

However `delete` will cause the current versions of the files to become hidden old versions.

Here is a session showing the listing and retrieval of an old version followed by a `cleanup` of the old versions.

Show current version and all the versions with `--b2-versions` flag.

```
$ rclone -q ls b2:cleanup-test
        9 one.txt

$ rclone -q --b2-versions ls b2:cleanup-test
        9 one.txt
        8 one-v2016-07-04-141032-000.txt
       16 one-v2016-07-04-141003-000.txt
       15 one-v2016-07-02-155621-000.txt
```

Retrieve an old version

```
$ rclone -q --b2-versions copy b2:cleanup-test/one-v2016-07-04-141003-000.txt /tmp

$ ls -l /tmp/one-v2016-07-04-141003-000.txt
-rw-rw-r-- 1 ncw ncw 16 Jul  2 17:46 /tmp/one-v2016-07-04-141003-000.txt
```

Clean up all the old versions and show that they've gone.

```
$ rclone -q cleanup b2:cleanup-test

$ rclone -q ls b2:cleanup-test
        9 one.txt

$ rclone -q --b2-versions ls b2:cleanup-test
        9 one.txt
```

Data usage

It is useful to know how many requests are sent to the server in different scenarios.

All copy commands send the following 4 requests:

```
/b2api/v1/b2_authorize_account
/b2api/v1/b2_create_bucket
/b2api/v1/b2_list_buckets
/b2api/v1/b2_list_file_names
```

The `b2_list_file_names` request will be sent once for every 1k files in the remote path, providing the checksum and modification time of the listed files. As of version 1.33 issue #818 causes extra requests to be sent when using B2 with Crypt. When a copy operation does not require any files to be uploaded, no more requests will be sent.

Uploading files that do not require chunking, will send 2 requests per file upload:

```
/b2api/v1/b2_get_upload_url
/b2api/v1/b2_upload_file/
```

Uploading files requiring chunking, will send 2 requests (one each to start and finish the upload) and another 2 requests for each chunk:

```
/b2api/v1/b2_start_large_file
/b2api/v1/b2_get_upload_part_url
/b2api/v1/b2_upload_part/
/b2api/v1/b2_finish_large_file
```

Versions

Versions can be viewed with the `--b2-versions` flag. When it is set rclone will show and act on older versions of files. For example

Listing without `--b2-versions`

```
$ rclone -q ls b2:cleanup-test
        9 one.txt
```

And with

```
$ rclone -q --b2-versions ls b2:cleanup-test
        9 one.txt
        8 one-v2016-07-04-141032-000.txt
       16 one-v2016-07-04-141003-000.txt
       15 one-v2016-07-02-155621-000.txt
```

Showing that the current version is unchanged but older versions can be seen. These have the UTC date that they were uploaded to the server to the nearest millisecond appended to them.

Note that when using `--b2-versions` no file write operations are permitted, so you can't upload files or delete them.

B2 and rclone link

Rclone supports generating file share links for private B2 buckets. They can either be for a file for example:

```
./rclone link B2:bucket/path/to/file.txt
https://f002.backblazeb2.com/file/bucket/path/to/file.txt?Authorization=xxxxxxxx
```

or if run on a directory you will get:

```
./rclone link B2:bucket/path
https://f002.backblazeb2.com/file/bucket/path?Authorization=xxxxxxxx
```

you can then use the authorization token (the part of the url from the `?Authorization=` on) on any file path under that directory. For example:

```
https://f002.backblazeb2.com/file/bucket/path/to/file1?Authorization=xxxxxxxx
https://f002.backblazeb2.com/file/bucket/path/file2?Authorization=xxxxxxxx
https://f002.backblazeb2.com/file/bucket/path/folder/file3?Authorization=xxxxxxxx
```

Standard Options

Here are the standard options specific to b2 (Backblaze B2).

--b2-account

Account ID or Application Key ID

- Config: account
- Env Var: RCLONE_B2_ACCOUNT
- Type: string
- Default: ""

--b2-key

Application Key

- Config: key
- Env Var: RCLONE_B2_KEY
- Type: string
- Default: ""

--b2-hard-delete

Permanently delete files on remote removal, otherwise hide files.

- Config: hard_delete
- Env Var: RCLONE_B2_HARD_DELETE
- Type: bool
- Default: false

Advanced Options

Here are the advanced options specific to b2 (Backblaze B2).

--b2-endpoint

Endpoint for the service. Leave blank normally.

- Config: endpoint
- Env Var: RCLONE_B2_ENDPOINT
- Type: string
- Default: ""

--b2-test-mode

A flag string for X-Bz-Test-Mode header for debugging.

This is for debugging purposes only. Setting it to one of the strings below will cause b2 to return specific errors:

- "fail_some_uploads"
- "expire_some_account_authorization_tokens"
- "force_cap_exceeded"

These will be set in the "X-Bz-Test-Mode" header which is documented in the b2 integrations checklist.

- Config: test_mode
- Env Var: RCLONE_B2_TEST_MODE
- Type: string
- Default: ""

--b2-versions

Include old versions in directory listings. Note that when using this no file write operations are permitted, so you can't upload files or delete them.

- Config: versions
- Env Var: RCLONE_B2_VERSIONS
- Type: bool
- Default: false

--b2-upload-cutoff

Cutoff for switching to chunked upload.

Files above this size will be uploaded in chunks of "--b2-chunk-size".

This value should be set no larger than 4.657GiB (== 5GB).

- Config: upload_cutoff
- Env Var: RCLONE_B2_UPLOAD_CUTOFF
- Type: SizeSuffix
- Default: 200M

--b2-copy-cutoff

Cutoff for switching to multipart copy

Any files larger than this that need to be server-side copied will be copied in chunks of this size.

The minimum is 0 and the maximum is 4.6GB.

- Config: copy_cutoff
- Env Var: RCLONE_B2_COPY_CUTOFF
- Type: SizeSuffix
- Default: 4G

--b2-chunk-size

Upload chunk size. Must fit in memory.

When uploading large files, chunk the file into this size. Note that these chunks are buffered in memory and there might a maximum of "--transfers" chunks in progress at once. 5,000,000 Bytes is the minimum size.

- Config: chunk_size
- Env Var: RCLONE_B2_CHUNK_SIZE
- Type: SizeSuffix
- Default: 96M

--b2-disable-checksum

Disable checksums for large (> upload cutoff) files

Normally rclone will calculate the SHA1 checksum of the input before uploading it so it can add it to metadata on the object. This is great for data integrity checking but can cause long delays for large files to start uploading.

- Config: disable_checksum
- Env Var: RCLONE_B2_DISABLE_CHECKSUM
- Type: bool
- Default: false

--b2-download-url

Custom endpoint for downloads.

This is usually set to a Cloudflare CDN URL as Backblaze offers free egress for data downloaded through the Cloudflare network. Rclone works with private buckets by sending an "Authorization" header. If the custom endpoint rewrites the requests for authentication, e.g., in Cloudflare Workers,

this header needs to be handled properly. Leave blank if you want to use the endpoint provided by Backblaze.

- Config: download_url
- Env Var: RCLONE_B2_DOWNLOAD_URL
- Type: string
- Default: ""

--b2-download-auth-duration

Time before the authorization token will expire in s or suffix ms|s|m|h|d.

The duration before the download authorization token will expire. The minimum value is 1 second. The maximum value is one week.

- Config: download_auth_duration
- Env Var: RCLONE_B2_DOWNLOAD_AUTH_DURATION
- Type: Duration
- Default: 1w

--b2-memory-pool-flush-time

How often internal memory buffer pools will be flushed. Uploads which requires additional buffers (f.e multipart) will use memory pool for allocations. This option controls how often unused buffers will be removed from the pool.

- Config: memory_pool_flush_time
- Env Var: RCLONE_B2_MEMORY_POOL_FLUSH_TIME
- Type: Duration
- Default: 1m0s

--b2-memory-pool-use-mmap

Whether to use mmap buffers in internal memory pool.

- Config: memory_pool_use_mmap
- Env Var: RCLONE_B2_MEMORY_POOL_USE_MMAP
- Type: bool
- Default: false

--b2-encoding

This sets the encoding for the backend.

See: the encoding section in the overview for more info.

- Config: encoding
- Env Var: RCLONE_B2_ENCODING
- Type: MultiEncoder
- Default: Slash,BackSlash,Del,Ctl,InvalidUtf8,Dot

Limitations

`rclone about` is not supported by the B2 backend. Backends without this capability cannot determine free space for an rclone mount or use policy `mfs` (most free space) as a member of an rclone union remote.