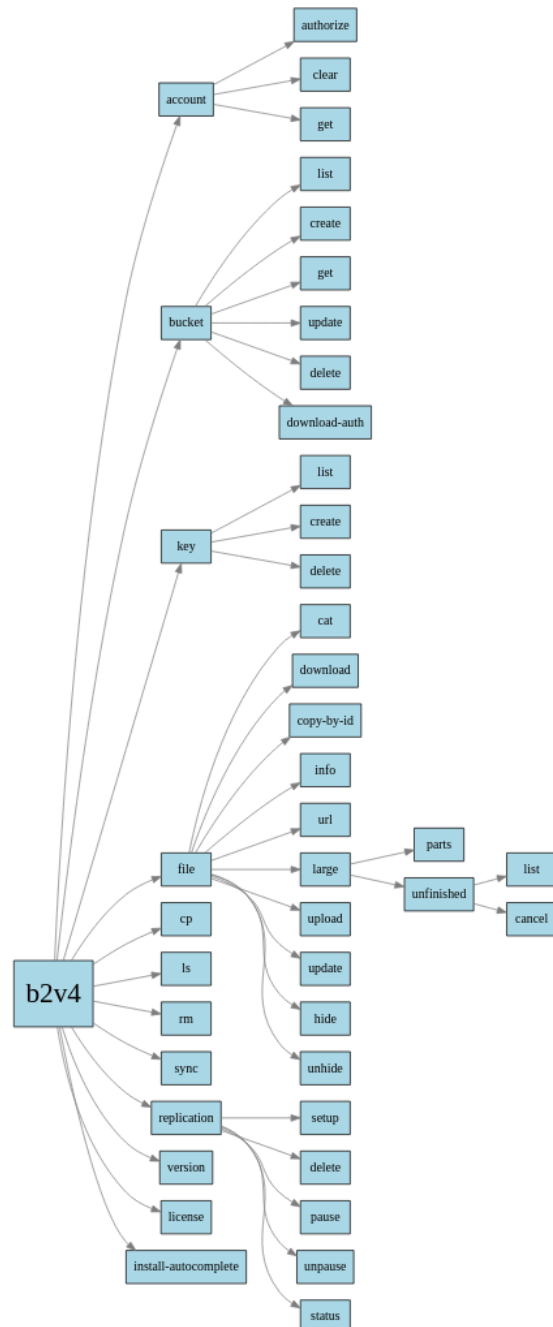


B2 Command-Line Tool (API behavior)

This covers calls made in the B2 Command-Line Tool (CLI) beyond the list of commands in the installation articles.

- Categories and subcommands (v4+)
- b2_uri (v4+)
- b2v4 & using b2v3 (v4+)
- b2 authorize-account
- b2 cancel-all-unfinished-large-files
- b2 cancel-large-file
- b2 clear-account
- b2 copy-file-by-id
- b2 create-bucket
- b2 create-key
- b2 delete-bucket
- b2 delete-file-version
- b2 delete-key
- b2 download-file-by-id
- b2 download-file-by-name
- b2 get-account-info
- b2 get-bucket
- b2 get-file-info
- b2 get-download-auth
- b2 get-download-url-with-auth
- b2 hide-file
- b2 list-buckets
- b2 list-keys
- b2 list-parts
- b2 list-unfinished-large-files
- b2 ls
- b2 make-url
- b2 make-friendly-url
- b2 rm
- b2 sync
 - Deletions at the destination:
 - Sync Exclusions
 - REGEX Formatting
 - File Comparisons
 - Example of compareVersions/compareThreshold
 - Server-Side Encryption
- b2 update-bucket
- b2 upload-file
- b2 update-file-legal-hold
- b2 update-file-retention

New subcommand structure



04/30/24 - New version 4.0+! New Changes! Note, all of the old commands STILL work as normal!!! PHEW!

As of 04/30/24, CLI V4 has been released and made some changes to the CLI we all love. These are the major changes:

- Command categories were added which added a second method of enacting the commands.

- For example, `b2 authorize-account` is now under the `account` category. The newly added command to authorize in the CLI is now `b2 account authorize` where “account” is the category and “authorize” is the subcommand.
- All old commands still work, but will just give a warning “WARNING: `list-buckets` command is deprecated. Use `bucket list` instead.”
- Both `b2 ls` and `b2 rm` no longer accept `bucketName` by itself. These will now only accept the `B2_URI` which means the whole command would look something like this: `b2 ls b2://bucketName`. There are also other commands (when using the new subcommand format) that will only accept `B2_URI`.
- All optional flag arguments have been changed from **camelCase** to **kebab-case**.
 - For example, `--lifecycleRules` is now `--lifecycle-rules`.

Full doc here https://docs.google.com/document/d/1_6mT84Od377MS8mplJnsCO4Xa9_k0OdY3UACt0VzAKE/edit

Categories and subcommands (v4+)

Because the CLI has grown with many newly added features, as of CLI v4+, commands are now categorized so that it's easier to navigate. These are the new categories:

b2 account	Account management subcommands.
b2 bucket	Bucket management subcommands.
b2 file	File management subcommands.
b2 install-autocomplete	Install autocomplete for supported shells.
b2 key	Application keys management subcommands.
b2 license	Print the license information for this tool.
b2 ls	List files in a given folder.
b2 replication	Replication rule management subcommands.
b2 rm	Remove a "folder" or a set of files matching a pattern.
b2 sync	Copy multiple files from source to destination.
b2 version	Print the version number of this tool.

For more information and a list of subcommands for a given category, enter the category (such as `b2 account`) into terminal with B2 CLI v4+ installed.

b2_uri (v4+)

As of CLI v4+, some of the new commands require a `B2_URI` or Uniform Resource Identifier. This makes it sound more enigmatic than it really is, but all it does is it formats a file name or file ID as a network path by placing either `b2://` in front of a bucketPath or `b2id://` in front of a file ID.

Bucket path example: If we want to list files in a bucket, it won't work with `b2 ls bucketName`. v4+ wants to see `b2 ls b2://bucketName`. If targeting a bucket path, it would be `b2://bucketName/folder`.

File example: If we want to remove a file from a bucket, it won't work with `b2 rm fileId`. v4+ wants to see `b2 rm b2id://fileId`. If targeting a file by name, it would be `b2://bucketName/fileName`.

b2v4 & using b2v3 (v4+)

`b2` now points to `b2v4`. While it points to `b2v4`, **kebab-case** arguments are required for both new and old format commands. The old **camelCase** will not work at all when using the new subcommand format. The **b2_uri** requirements will apply only when using the new subcommand format or `ls/rm`.

If we use `b2v3` instead, the **kebab-case** and **b2_uri** requirements will no longer apply and we are able to use **camelCase** and **bucketPath/fileId**. Interestingly, when using `b2v3` we are able to use the new subcommand format with the old **camelCase** and **bucketPath/fileId**; and vice versa.

<code>b2 update-bucket --bucketInfo '{"matt":"cool"}'</code> <code>myBucket allPrivate</code> ❌	Using <code>b2v4</code> , this does NOT work because <code>--bucketInfo</code> is in camelCase. It should be <code>b2 --bucket-info</code>
<code>b2v3 update-bucket --bucketInfo '{"pizza":"yummy"}'</code> <code>myBucket allPrivate</code> ✅	Using <code>b2v3</code> , this does work!
<code>b2 rm myBucket</code> ❌	Using <code>b2v4</code> , this does NOT work because <code>myBucket</code> is not formatted as <code>b2://myBucket</code> or a <code>B2_URI</code>
<code>b2v3 rm myBucket</code> ✅	Using <code>b2v3</code> , this does work!

*Below, the individual arguments will have optional portions of the call which will be in [blue] and required portions of the call.

Required portions of the call will be position and order specific.

All calls will have `[-h]` which brings up small guide on how to use the command in the CLI.

b2 authorize-account

```
b2 authorize-account [-h] [applicationKeyId] [applicationKey]
```

Category: account

Subcommand: authorize

Full command: `b2 account authorize [applicationKeyId] [applicationKey]`

Required information:

- keyID and application key
 - Can be either Master or created key.

Notes:

There are two ways to call this command:

1. Type in the call with all of the arguments and press enter to authenticate.
2. Type in the call without any of the arguments and press enter. The CLI will ask you to input each argument individually.
 - a. When inputting the `applicationKey`, it will not be displayed in the terminal or command window.
 - b. This method is useful for if you are sharing your screen for others and you do not want to them see your `applicationKey`.

When successful, it will state "Using `https://api.backblazeb2.com`"

If it fails, it will tell you *"ERROR: unable to authorize account: Invalid authorization token. Server said: (bad_auth_token)"*.

```
> b2 authorize-account
```

```
> Backblaze application key ID:  
0002afc49h8bd640000000098
```

```
> Backblaze application key:
```

b2 cancel-all-unfinished-large-files

```
b2 cancel-all-unfinished-large-files [-h] bucketName
```

Category: file

Subcommand: large unfinished cancel

Full command: b2 file unfinished cancel b2://bucketName

Required information:

- `bucketName` - Name of the bucket whose unfinished large files will be canceled.
 - When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

This will list the fileId for all unfinished large files and then will cancel and delete all parts of them.

Output

```
> b2 cancel-all-unfinished-large-files matt01  
  
> 4_ze22a0fbef7h67g558bad0619_f214a4bffa2db52b2_d20231024_m215033_c000_v0001411_t0046_u01698184233469  
canceled
```

b2 cancel-large-file

```
b2 cancel-large-file [-h] fileId
```

Category: file

Subcommand: large unfinished cancel

Full command: b2 file large unfinished cancel b2://fileId

Require information:

- `fileId` - The fileId of the unfinished large file that will be canceled.
 - When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

This will list the fileId for the unfinished large file and then will cancel and delete all parts of it.

Output

```
> b2 cancel-large-file  
  
4_ze22a0fbef7h67g558bad0619_f214a4bffa2db52b2_d20231024_m215033_c000_v0001411_t0046_u01698184233469  
  
> 4_ze22a0fbef7h67g558bad0619_f214a4bffa2db52b2_d20231024_m215033_c000_v0001411_t0046_u01698184233469  
canceled
```

b2 clear-account

```
b2 clear-account [-h]
```

Category: account

Subcommand: clear

Full command: b2 account clear

No required information.

Notes:

This will just invalidate the current session's authorization token. This is essentially logging out of the b2 cli.

Output:

There will be no response from the cli.

b2 copy-file-by-id

```
b2 copy-file-by-id [-h] [--metadata-directive {copy,replace}] [--content-type CONTENTTYPE] [--range RANGE] [--info INFO] [--destination-server-side-encryption {SSE-B2}] [--destination-server-side-encryptionAlgorithm {AES256}] sourceFileId destinationBucketName b2FileName
```

Category: file

Subcommand: copy-by-id

Full command: b2 file copy-by-id sourceFileId destinationBucketName b2FileName

Required information:

- `sourceFileId` - File ID for the file stored in bucket.
 - This can be found with `b2 ls` with the optional `--long` argument.
- `destinationBucketName` - Name of destination bucket in the same account.
- `b2FileName` - User chosen file name as it will be stored in B2.
 - Does not have to match the original.

Notes:

This call can be used for a file in one bucket to create a copy in the same or another bucket in the same account. It cannot be used to move a file from one account to another.

The copy process happens server-side. No downloading and reuploading necessary.

When entering the `b2FileName`, a prefix can be added as a folder path. If the folder path does not exist yet, it will be created when the call is sent.

Optional arguments:

The option `--metadata-directive {copy,replace}` will allow you to copy or replace the metadata

- By default, it copies the file info and content-type. You can replace those by setting the `--metadata-directive {copy,replace}` to `replace`.
 - For this call, `--content-type CONTENTTYPE` and `--info INFO` should only be provided (and **MUST** be provided) if `--metadata-directive {copy,replace}` is set to `replace`.
 - `--content-type CONTENTTYPE` - This is to set the content-type to something other than the original metadata. If the content-type is to stay the same, it should still be stated in the call. Here is a list of possible content-types:

<https://www.backblaze.com/b2/docs/content-types.html>

- `--info INFO` - These are variables and values chosen and defined by the user.
 - This is to be formatted as `VARIABLE=VALUE`. Example below.
 - If more than one `VARIABLE=VALUE` is to be specified, and additional `--info` entry must be made. Example below.
- If left out, this choice will just default to `copy`.

The option `--range RANGE` allows you to specify a part of a file to be copied.

- `RANGE` should be a pair of numbers (number of bytes) separated by a comma.
- The “beginning of the file” would be 0, so if you want to copy the first half of the file of a 100,000 byte file, you would need to set `--range 0,50000`
 - It will error out if the first number is larger than the second.
 - It does not have to start at the beginning of the file. It can be something like `10,51000`

The option `--destination-server-side-encryption {SSE-B2}` will set Server-Side Encryption for the file at the file level.

- This does not require Default Server-Side Encryption enabled on the bucket.
- This will default to setting `--destination-server-side-encryptionAlgorithm {AES256}` to AES256, so this is not a required argument to include when setting `--destination-server-side-encryption SSE-B2`.

Output: The call will give back information similar to `b2 get-file-info`.

```
> b2 copy-file-by-id
4_zf2badf6h86e47c788b7d0620_f11833e97a5bb9317_d20230621_m232936_c000_v0001060_t0000_u01687390176565_matt01
newfolder/text.txt

> {
  "accountId": "2afc49h8bd64",
  "action": "copy",
  "bucketId": "e22a0fbef7h67g558bad0619",
  "contentMd5": "6503216a7c6d19102c8a56102b462e8d",
  "contentSha1": "88cb53f8908cac3a65659b857d5ee555efc2bd4a",
  "contentType": "text/plain",
  "fileId":
"4_ze22a0fbef7h67g558bad0619_f107dc98ce4522b5c_d20231024_m223001_c000_v0001410_t0003_u01698186601940",
  "fileInfo": {
    "src_last_modified_millis": "1686776127568"
  },
  "fileName": "newfolder/text.txt",
  "fileRetention": {
    "mode": null,
    "retainUntilTimestamp": null
  },
}
```

```
"legalHold": null,

"replicationStatus": null,

"serverSideEncryption": {

  "mode": "none"

},

"size": 8582,

"uploadTimestamp": 1698186601940

}
```

b2 create-bucket

```
b2 create-bucket [-h] [--bucket-info BUCKETINFO] [--cors-rules CORSRULES] [--lifecycle-rules LIFECYCLERULES] [--default-server-side-encryption {SSE-B2,none}] [--default-server-side-encryption-algorithm {AES256}] bucketName bucketType
```

Category: bucket

Subcommand: create

Full command: b2 bucket create bucketName bucketType

Required information:

- `bucketName` - Your chosen bucket name.
 - The bucket name must follow the normal bucket [guidelines](#).
- `bucketType` - Choice of bucket type.
 - Public
 - Applications will **not** need an authorization token to download from bucket.
 - Private
 - Application will need an authorization token to download from bucket.

Notes:

In the B2 docs for the CLI, the required `bucketType` argument show `[allPublic | allPrivate]` instead and is wrapped in square brackets, but the choice between `allPublic` and `allPrivate` is required. You must choose one or the other and not both.

Optional arguments:

The following optional arguments need values in JSON format. There are some rules for JSON formatting.

- If adding more than one VARIABLE:VALUE pair, it must be separated with a comma.
- If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

`--bucket-info BUCKETINFO` - These are user chosen and user defined values. They can be named anything and be anything the user wants.

BUCKETINFO Format:

The `BUCKETINFO` part must be replaced with a JSON formatted as `{“VARIABLE”: “VALUE”}`

If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

Example:

```
b2 create-bucket --bucket-info '{"people": "cool","pizza": "yes"}' mattclibucket2 allPublic
```

For more information: [🔥 Cloud Storage Buckets](#)

[\[--cors-rules CORSRULES\]](#) - This is to set CORS Rules (Cross-Origin Resource Sharing).

CORSRULES Format:

The [CORSRULES](#) part must be replaced with a JSON formatted as `'[{"VARIABLE": "VALUE", "VARIABLE2": "VALUE2"}]'`

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [] in the wrapping.

If a value should be nonspecific and refer to "all" (such as all Origins or all Files), then the wild card * can be used. It would look like this:

`"VARIABLE": "*"`

Example:

```
b2 create-bucket --cors-rules '[{"corsRuleName": "downloadFromAnyOrigin", "allowedOrigins": ["https"], "allowedHeaders": ["range"], "allowedOperations": ["b2_download_file_by_id", "b2_download_file_by_name"], "exposeHeaders": ["x-bz-content-sha1"], "maxAgeSeconds": 3600}]' mattybucket allPublic
```

For more CORS Rules information: [🔥 Cloud Storage Cross-Origin Resource Sharing Rules](#)

[\[--lifecycle-rules LIFECYCLERULES\]](#) - This is to set Lifecycle Rules.

LIFECYCLERULES Format:

The [LIFECYCLERULES](#) part must be replaced with a JSON formatted as `'[{"VARIABLE": "VALUE", "VARIABLE2": "VALUE2"}]'`

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [] in the wrapping.

If a value should be nonspecific and refer to "all" (such as all Origins or all Files), then the wild card * can be used. It would look like this:

`"VARIABLE": "*"`

Example:

```
b2 create-bucket --lifecycle-rules '[{"daysFromHidingToDeleting": 1, "daysFromUploadingToHiding": null, "fileNamePrefix": "chunks/"}]' mattybox allPublic
```

For more Lifecycle Rules information: [🔥 Cloud Storage Lifecycle Rules](#)

Default Server-Side Encryption:

The option [\[--default-server-side-encryption {SSE-B2,none}\]](#) will set Server-Side Encryption for the bucket.

- If the value chosen is [SSE-B2](#) it will enable Server-Side Encryption for the bucket. This will not encrypt non-encrypted objects that are already in the bucket.
- If the value chosen is [none](#) it will disable Server-Side Encryption for the bucket. This will not decrypt encrypted objects that are already in the bucket.

The option [\[--default-server-side-encryption-algorithm {AES256}\]](#) doesn't have an effect on the call as enabling Server-Side Encryption with [--default-server-side-encryption SSE-B2](#) will set [--default-server-side-encryption-algorithm](#) to [AES256](#) automatically.

Example:


```
b2 create-bucket --default-server-side-encryption SSE-B2 machu-bux2 allPublic
```

For more information: [Server-Side Encryption](#)

Output: The call will give back the brand new created bucket ID. Adding the optional arguments will not change the output.

```
> b2 create-bucket newestbucket allPublic  
> 82dacf2ef3578j688bbd0619
```

b2 create-key

```
b2 create-key [-h] [--bucket BUCKET] [--name-prefix NAMEPREFIX] [--duration DURATION] keyName capabilities
```

Category: key

Subcommand: create

Full command: b2 key create keyName capabilities

Required information:

- `keyName` - Name of the key.
 - Key names are required, can contain letters, numbers, and "-", and are limited to 100 characters.
- `capabilities` - User chosen capabilities (aka permissions) for the application key.
 - Multiple capabilities can be listed separated by a comma and no space.

Notes:

When making this call, the current application key must possess read AND write capabilities in order to make this call. Although the specific permission is called `writeKeys`, it is a permission specific to being both read and write.

Permissions:

The list of possible permissions as stated on [🔥 Create an Application Key Within Your Backblaze Account](#) is not the full list of permissions that you can list in this call. You are able to list any and all possible permissions ranging from ones required for object lock, server-side encryption as well as `listAllBucketNames` which is required to use the S3 compatible API when creating an application key restricted to a single bucket with `--bucket BUCKET`.

Optional arguments:

`--bucket BUCKET` - Like the above, the list of possible permissions as stated on [🔥 Create an Application Key Within Your Backblaze Account](#) is not the full list of permissions that you can list in this call.

`--name-prefix NAMEPREFIX` restricts file access to files whose names start with the prefix or file path.

`--duration DURATION` will set a timer for the application key that starts after the key is created. When the timer runs out, the key will be deleted and will no longer be usable. If this is left out of the call, the key will not expire and will not be deleted until the user manually deletes it through the CLI or the web UI.

Output: The call will give back both the keyId and applicationKey. Optional arguments will not change the output of this call.

```
> b2 create-key newKeyName readBucketEncryption,listBuckets  
> 0002afc49h8bd640000000099 K000UvRbZVvINferEzKFtWLn/u0vRUQ
```

b2 delete-bucket

```
b2 delete-bucket [-h] bucketName
```

Category: bucket

Subcommand: delete

Full command: b2 bucket delete bucketName

Required information:

- bucketName - Name of the bucket that will be deleted.

Notes:

The bucket that will be deleted must be empty and free of any files.

The placeholder file called ".DS_Store" left over from the cli command `b2 sync` with optional `--delete` and `--allow-empty-source` will count as a file that prevents the bucket from being deleted. It can easily be deleted in the web UI or with the `b2 delete-file-version` so that you can delete the bucket.

Output: None

b2 delete-file-version

```
b2 delete-file-version [-h] [fileName] fileId
```

Category: rm

Subcommand: N/A

Full command: b2 rm B2_URI

Required information:

- b2id://fileId - File ID for the file that will be deleted.
 - This can be found with `b2 ls` with the optional `--long` argument.
 - When using the subcommand version, this requires the B2_URI, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

If there is a specific version of a file that you want to delete, each version of the same file will have its own unique fileId.

Optional arguments:

Specifying the [fileName] is more efficient than leaving it out. If you omit the [fileName], it requires an initial query to B2 to get the file name, before making the call to delete the file. This extra query requires the readFiles capability.

Output:

Optional arguments will not change the output of this call.

Output

```
> b2 delete-file-version
4_zf2badf6h86e47c788b7d0620_f101cb0cc78b397aa_d20230621_m232929_c000_v0001057_t0010_u01687390169931
> {
  "action": "delete",
```

```
"fileId":  
"4_zf2badf6h86e47c788b7d0620_f101cb0cc78b397aa_d20230621_m232929_c000_v0001057_t0010_u01687390169931",  
  
"fileName": "test.txt"  
  
}
```

b2 delete-key

```
b2 delete-key [-h] applicationKeyId
```

Category: key

Subcommand: delete

Full command: b2 key delete applicationKeyId

Required information:

- `applicationKeyId` - The keyId from the pair you want to delete.

Output:

The keyId will be returned if the call is successful.

Output

```
> b2 delete-key 0002afc49h8bd640000000099  
  
> 0002afc49h8bd640000000099
```

b2 download-file-by-id

```
b2 download-file-by-id [-h] [--no-progress] fileId localFileName
```

Category: file

Subcommand: download

Full command: b2 file download b2id://fileId localFileName

Required information:

- `fileId` - File ID for the file that will be downloaded.
 - This can be found with `b2 ls` with the optional `[-long]` argument.
 - When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.
- `localFileName` - User chosen local file name.
 - Does not have to match the original.

Notes:

This call will download the file to the location that the CLI is currently in. However, a full file path plus file name can take the place of `localFileName`.

When entering the `b2FileName` , a prefix can be added as a folder path if the file is nested in one or more subfolders.

When entering the `localFileName` a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

The argument `[--no-progress]` will simply do the same thing but will not show a progress bar for the download.

Output:

The output will give back a progress bar and will give back information about the uploaded file if successful.

Output

```
> b2 download-file-by-id
4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768
download.pdf

> File name:      test.pdf

> File id:
4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768

> File size:      11940

> Content type:   application/pdf

> Content sha1:   1b8ee61ab12c1bfdda4c4bd9a9155bebb90ac25b

> Encryption:    none

> Retention:      none

> Legal hold:     <unset>

> INFO src_last_modified_millis: 1684449070014

> Checksum matches

> download.pdf:  0%|                                     | 0.00/11.9k [00:00<?, ?
B/s]Download finished

> download.pdf: 100%|████████████████████████████████████████████████████████████████████████████████| 11.9k/11.9k
[00:00<00:00, 6.25MB/s]
```

b2 download-file-by-name

```
b2 download-file-by-name [-h] [--no-progress] bucketName b2FileName localFileName
```

Category: file

Subcommand: download

Full command: b2 file download b2://bucketName/file.txt localFileName

Required information:

- `bucketName` - Name of the bucket that the file will be downloaded from.
- `b2FileName` - The full path plus file name that will be downloaded.

- When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.
- `localFileName` - User chosen local file name.
 - Does not have to match the original.

Notes:

This call will download the file to the location that the CLI is currently in. However, a full file path can be added as a prefix to the `localFileName`.

When entering the `b2FileName`, a prefix can be added as a folder path if the file is nested in one or more subfolders.

When entering the `localFileName` a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

The argument `[--no-progress]` will simply do the same thing but will not show a progress bar for the download.

Output:

The output will give back a progress bar and will give back information about the uploaded file if successful.

Output

```
> b2 download-file-by-name matt00 test.pdf downloaded-test.pdf

> File name:      test.pdf

> File id:
4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768

> File size:      11940

> Content type:   application/pdf

> Content sha1:   1b8ee61ab12c1bfd4c4bd9a9155bebb90ac25b

> Encryption:    none

> Retention:      none

> Legal hold:     <unset>

> INFO_src_last_modified_millis: 1684449070014

> Checksum matches

> downloaded-test.pdf: 0%|                                     | 0.00/11.9k [00:00<?, ?
B/s]Download finished

> downloaded-test.pdf: 100%|████████████████████████████████████████████████████████████████████████████████| 11.9k/11.9k
[00:00<00:00, 7.03MB/s]
```

b2 get-account-info

b2 clear-account `[-h]`

Category: account

Subcommand: get

Full command: b2 account get

No required information.

Notes:

Despite the name of the call, this will not give back any information about the Backblaze account. This will actually only give back information about the applicationKey, the current session's authentication token, its permissions, apiUrl and downloadUrl.

Output

```
> b2 get-account-info

> {
  "accountAuthToken": "4_0002afc49h8bd640000000098_01afe461_043c23_acct_Cyf4AJ0Y1oyx-fe85qEx4T-yZw8=",
  "accountFilePath": "/Users/matto/.b2_account_info",
  "accountId": "2afc49h8bd64",
  "allowed": {
    "bucketId": null,
    "bucketName": null,
    "capabilities": [
      "deleteBuckets",
      "listKeys",
      "writeFiles",
      "readBuckets",
      "writeBuckets",
      "deleteFiles",
      "writeBucketEncryption",
      "readBucketReplications",
      "bypassGovernance",
      "writeBucketRetentions",
      "readFiles",
      "writeFileRetentions",
      "writeFileLegalHolds",
      "readBucketRetentions",
      "shareFiles",
      "readFileLegalHolds",
      "readFileRetentions",
      "readBucketEncryption",
      "listFiles",
      "listBuckets",
      "writeBucketReplications",
```

```
"deleteKeys",
  "writeKeys"
],
"namePrefix": null
},
"apiUrl": "https://api000.backblazeb2.com",
"applicationKey": "K000W7HWN+4Dnh2QV4t04jzwjC1Et4",
"applicationKeyId": "0002afc49h8bd640000000098",
"downloadUrl": "https://f000.backblazeb2.com",
"isMasterKey": false
}
```

b2 get-bucket

```
b2 get-bucket [-h] [--show-size] bucketName
```

Category: bucket

Subcommand: get

Full command: b2 bucket get bucketName

Required information:

- `bucketName` - Name of bucket you want to get information for.

Notes:

This is a lot like `b2 get-file-info` in that it returns information about the bucket.

Optional arguments:

`--show-size` - This will add `totalSize` to the bottom of the list and will display the total size of the bucket in bytes.

Output:

Output

```
> b2 get-bucket --show-size matt00
> {
  "accountId": "2afe93h8bd60",
  "bucketId": "f2badf6h86e47c788b7d0620",
  "bucketInfo": {},
  "bucketName": "matt00",
  "bucketType": "allPublic",
  "corsRules": [],
```

```
"defaultRetention": {  
  "mode": null  
},  
"default-server-side-encryption": {  
  "algorithm": "AES256",  
  "mode": "SSE-B2"  
},  
"fileCount": 5,  
"isFileLockEnabled": false,  
"lifecycleRules": [],  
"options": [  
  "s3"  
],  
"replication": {  
  "asReplicationDestination": null,  
  "asReplicationSource": null  
},  
"revision": 4,  
"totalSize": 73357  
}
```

b2 get-file-info

```
b2 get-file-info [-h] fileId
```

Category: file

Subcommand: info

Full command: b2 file info B2_URI

Required information:

- `fileId` - File ID for the file that will be deleted.
 - This can be found with `b2 ls` with the optional `[-long]` argument.
 - When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

This will give general information about the file stored in B2.

Output


```
> b2 get-file-info
4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768

> {
  "accountId": "2afe93h8bd60",
  "action": "upload",
  "bucketId": "f2badf6h86e47c788b7d0620",
  "contentMd5": "e73bc22b0d02385b6666f26aca3e18e9",
  "contentSha1": "1b8ee61ab12c1bfdda4c4bd9a9155bebb90ac25b",
  "contentType": "application/pdf",
  "fileId":
"4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768",
  "fileInfo": {
    "src_last_modified_millis": "1684449070014"
  },
  "fileName": "test.pdf",
  "fileRetention": {
    "mode": null,
    "retainUntilTimestamp": null
  },
  "legalHold": null,
  "replicationStatus": null,
  "serverSideEncryption": {
    "mode": "none"
  },
  "size": 11940,
  "uploadTimestamp": 1684449072768
}
```

b2 get-download-auth

```
b2 get-download-auth [-h] [--prefix PREFIX] [--duration DURATION] bucketName
```

Category: bucket

Subcommand: get-download-auth

Full command: b2 bucket get-download-auth bucketName

Required information:

- bucketName - Name of the bucket that the authorization token is for.

Notes:

This will return an authorization token that can be passed to `b2_download_file_by_name` ([link](#); not the CLI version) through the `Authorization` header and allow another user or machine to download a file or files from their bucket.

Optional Arguments:

`[--prefix PREFIX]` - You can add a file path in the bucket that will restrict the authorization token to only download files from that specific folder path.

- The prefix does not have to match an existing folder or set of folders within the bucket. If done this way, the call will still return an authorization token that is only allowed to download objects within the non-existent folder or set of folders. In other words, this token will not have access to any objects in the bucket if the prefix does not exist.

`[--duration DURATION]` - Specifies how long the auth token is valid for in seconds.

Output:

All variations of this call will only return the authorization token.

Output

```
> b2 get-download-auth matt00  
  
>  
3_20231025173515_b079b9f2597899ddf29282f9_e541ba014635ac036d13866be53fbf1346b30c26_000_20231026173515_0000_  
dnld
```

b2 get-download-url-with-auth

```
b2 get-download-url-with-auth [-h] [--duration DURATION] bucketName fileName
```

Category: file

Subcommand: url

Full command: b2 file url --with-auth B2_URI

Required information:

- `bucketName` - Name of bucket the url will download the file from.
- `fileName` - Name of the file including file prefix/path if applicable.
- When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

Useful for allowing a user to download a file from a private bucket.

Similar to `[--prefix PREFIX]` in `b2 get-download-auth`, the `fileName` does not need to exist for the CLI to return a url. When the URL is accessed in an attempt to download a file that **does not exist** in the bucket but is specified with `fileName`, the browser will give back a 404 error.

If a previous version of a file is to be downloaded, that version ID should be used for

Optional Argument:

`[--duration DURATION]` - Specifies how long the authorized URL is valid for in seconds.

Output:

All variations of this call will only return the authorized download URL.

Output

```
> b2 get-download-url-with-auth matt00 test.pdf
> https://f000.backblazeb2.com/file/matt00/test.pdf?
Authorization=3_20231025174124_28f6451265ee90c54365d4c7_9bd37f99ccc04f191a7103665c5e408b3285886c_000_20231026
174124_0008_dnlld
```

b2 hide-file

```
b2 hide-file [-h] bucketName fileName
```

Category: file

Subcommand: hide

Full command: b2 file hide B2_URI

Required information:

- `bucketName` - Name of the bucket the file to be hidden is stored in.
- `fileName` - Name of the file including file prefix/path if applicable.
- When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

This will download and then upload the file and will be considered a different version of the same file. The original will still exist.

Output

```
> b2 hide-file matt00 upload.txt
> {
  "accountId": "2afe93h8bd60",
  "action": "hide",
  "bucketId": "f2badf6h86e47c788b7d0620",
  "contentMd5": "d41d8cd98f00b204e9800998ecf8427e",
  "contentSha1": "da39a3ee5e6b4b0d3255bfef95601890afd80709",
  "contentType": "application/x-bz-hide-marker",
  "fileId":
  "4_zf2badf6h86e47c788b7d0620_f423c6c3663006f0a_d20231025_m174808_c000_v7007000_t0000_u01698256088071",
  "fileInfo": {},
  "fileName": "upload.txt",
  "fileRetention": {
    "mode": null,
    "retainUntilTimestamp": null
  },
}
```

```
"legalHold": null,

"replicationStatus": null,

"serverSideEncryption": {

  "mode": "none"

},

"size": 0,

"uploadTimestamp": 1698256088071

}
```

b2 list-buckets

b2 list-buckets [-h] [--json]

Category: bucket

Subcommand: list

Full command: b2 bucket list

No required information.

Notes:

This will list the `bucketId` , `bucketType` , and `bucketName`

This is the same as `b2 ls` .

Optional arguments:

The CLI version has an optional argument `--json` which formats the list of buckets in a machine-readable output and will also include more bucket information.

Output

```
> b2 list-buckets

> f2badf6h86f45g788b7d0620 allPublic awesomebucket

> 9bae30056aq238ec85b90a14 allPublic nicerbucket

> 5b9ef0550li203ec85b90a14 allPublic nicestbucket
```

b2 list-keys

b2 list-keys [-h] [--long]

Category: key

Subcommand: list

Full command: b2 key list

No required information.

Notes:

This will list all `keyIds` & name of the keys created in account. This will NOT list applicationKeys

Optional argument:

The argument `[-long]`, on top of listing all keyIds, will also list bucket restriction, prefix restriction, duration, as well as all capabilities/permissions.

Output:

Output

```
> b2 list-keys  
  
> 000be09ks5c59a40000000007  oldkeyname  
  
> 000be09ks5c59a40000000008  newkeyname
```

b2 list-parts

```
b2 list-parts [-h] largeFileId
```

Category: file

Subcommand: large parts

Full command: b2 file large parts `B2_URI`

Required information:

- `largeFileId` - File ID of the large file whose individual parts you want to list.
 - This can be found with `b2 list-unfinished-large-files`.
 - When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

This will only work for a large file upload that has not been finished or canceled.

This will list the `total number of parts`, `contentLegnth` (size in bites), and the `sha1` of the large file.

Output

```
> b2 list-parts  
  
4_ze22a0fbef7h67g558bad0619_f214a4bffa2db52b2_d20231024_m215033_c000_v0001411_t0046_u01698184233469  
  
> 3  9715200 30aef4f7bbacd26f169503b735284f66245e7943
```

b2 list-unfinished-large-files

```
b2 list-unfinished-large-files [-h] bucketName
```

Category: file

Subcommand: large unfinished list

Full command: `b2 file large unfinished list b2://bucketName`

Required information:

- `bucketName` - Name of the bucket whose unfinished large files will be listed.
- When using the subcommand version, this requires the `b2://bucketName`.

Notes:

This will list the `fileId`, `fileName`, `contentType`, and `lastModified` for each unfinished large file.

`lastModified` is an integer if milliseconds since “epoch”.

Epoch dates & timestamps:

A date in epoch format is an integer that represents milliseconds from January 1, 1970 to a specified date.

For example, if you want a file to be locked until January 1, 2024, you would set `--retain-until 1704096000` where 1704096000 is the number of milliseconds from January 1, 1970 to January 1, 2024. [This](#) is a useful tool to calculate a date converted to epoch.

Output

```
> b2 list-unfinished-large-files matt00

> 4_zf2badf6h86e47c788b7d0620_f20769d9737d1c5d7_d20231026_m225826_c000_v0001069_t0003_u01698361106883
largefile application/octet-stream large_file_sha1=fd7c5327c68fc94b62dc9f58fc1cdb3c8c01258
src_last_modified_millis=1698184182559
```

b2 ls

`b2 ls [-h] [--long] [--json] [--versions] [--recursive] [--prefix] [b2://bucketName] [folderName]`

Required information:

Can be sent alone to list buckets in an account. If listing files in a bucket or bucket path:

- `b2://bucketName` - Name of the bucket whose contents will be listed.
 - As of version 4.0, this call will only accept the `B2_URI` / `b2://bucketName`.
- `[folderName]` - If there are contents nested in “folders”, those folder paths will be needed as well.

Notes:

With `b2://bucketName` alone, this call will list the contents at the top level of the bucket.

With `b2://bucketName` & `folderName`, this call will list all contents in the included folder name.

Without `b2://bucketName`, this call will list all buckets in the account. Similar to `b2 list-buckets` or `b2 bucket list`.

Optional arguments:

If there are desired contents nested in a folder, the optional arguments `--recursive` and/or `[folderName]` will need to be included in the call.

- When making the call with `--recursive` without a specified `[folderName]`, it will list everything nested in each folder at the top level.
- When making the call with `[folderName]`, it must either be a single folder name or a full folder path. It will list just the contents in the top level of the specified folder or ending folder of the full folder path.
- When making the call with both `--recursive` and `[folderName]`, it will list the contents at the top level and everything nested in the specified folder or the ending folder of the full folder path.
- If included, the `[folderName]` must be *after* `b2://bucketName`

If you need further information about the contents in the bucket or bucket/folder, include the argument `--long`. This will display file ID, upload date/time, file size and file name.

If you need to list files along with file versions in a bucket, `--long` & `--versions` can be used for the call. By itself, `--versions` will only list the file names with no further information.

▼ Bucket contents for the examples below:

- matt-cli-bucket
 - /example/
 - animals/
 - dog.jpeg
 - pasta.jpeg

Output

```
> b2 ls b2://matt00
> newfolder/
> test.pdf
> test.txt
```

```
> b2 ls b2://matt00 newfolder
> newfolder/test.pdf
> newfolder/test.txt
```

```
> b2 ls --recursive b2://matt00
> newfolder/test.pdf
> newfolder/test.txt
> test.pdf
> test.txt
```

```
> b2 ls --long b2://matt00
>
- - - - 0 newfolder/
> 4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768 upload
2023-05-18 22:31:12 11940 test.pdf
> 4_zf2badf6h86e47c788b7d0620_f11833e97a5bb9317_d20230621_m232936_c000_v0001060_t0000_u01687390176565
upload 2023-06-21 23:29:36 8582 test.txt
```

b2 make-url

b2 make-url `-h` fileId

Category: file

Subcommand: url

Full command: `b2 file url b2id://fileId`

Required information:

- `fileId` - File ID for the file that the URL will link to.
- When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

This will create a download link to the file with the `fileId` in the URL. The URL will be the same as the “Native URL” in the file info when browsing the bucket through the web UI. [Here](#) is an example.

If the bucket is private, this will successfully create a download link to the file but the link cannot be accessed without authorization. [Here](#) is an example of what you would see if you try to access a file from a private bucket.

Output:

Output

```
> b2 make-url
4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768

https://f000.backblazeb2.com/b2api/v2/b2_download_file_by_id?
fileId=4_zf2badf6h86e47c788b7d0620_f1099ecdffc36d9c0_d20230518_m223112_c000_v0001061_t0030_u01684449072768
```

b2 make-friendly-url

`b2 make-friendly-url [-h] bucketName fileName`

Category: file

Subcommand: url

Full command: `b2 file url b2://bucketName/fileName`

Required information:

- `bucketName` - Name of bucket that is storing the file to share.
- `fileName` - Name of the file including file prefix/path if applicable.
- When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.

Notes:

This will create a download link to the file with the file `bucketName` and `fileName` in the URL. The URL will be the same as the “Friendly URL” in the file info when browsing the bucket through the web UI. [Here](#) is an example.

If the bucket is private, this will successfully create a download link to the file but the link cannot be accessed without authorization. [Here](#) is an example of what you would see if you try to access a file from a private bucket.

Output:

Output

```
> b2 make-friendly-url matt00 test.txt

> https://f000.backblazeb2.com/file/matt00/test.txt
```


b2 rm

```
b2 rm [-h] [--bypass-governance] [--dry-run] [--queue-size QUEUE_SIZE] [--no-progress] [--fail-fast] [--threads THREADS] [--versions] [-r]
[--with-wildcard] [--include FILTERS] [--exclude FILTERS] B2_URI
```

⚠ **Use with caution as it has odd mechanism***

Required information:

- `b2://bucketName` - Name of the bucket to delete items from.
 - When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt`, `b2id://fileId` or `b2://bucketName`.
- `[folderName]` - Folder path if deleting a specific folder in the bucket.

Notes:

This will remove all files in the top level of a bucket if no `folderName` is specified. If a `folderName` is specified, it will delete all files in that folder.

* If `b2 rm` targets a location containing a file with versions the API will delete **only** the most recently uploaded version of that file. The rest of the versions will remain.

* Using `b2 rm` with `--recursive` & `--versions` will erase **everything** in the bucket. Be careful using this.

* For buckets storing an extremely large amount of files, use life cycle rules to delete everything instead.

Optional Arguments:

`--recursive` will expand the command to also target child folders.

`--versions` will expand the command to also target older versions of files, if they exist.

`--fail-fast` will stop the whole execution the moment the command is unable to delete a file for one reason or another.

Output

```
> b2 rm b2://matt00
```

```
> No response when successful. This will only delete the most recent version of each file in the top level of the bucket and won't
target anything in child folders.
```

b2 sync

```
b2 sync [-h] [--no-progress] [--dry-run] [--allow-empty-source] [--exclude-all-symlinks] [--threads THREADS] [--compare-versions
{none,modTime,size}] [--compare-threshold MILLIS] [--exclude-regex REGEX] [--include-regex REGEX] [--exclude-dir-regex REGEX] [--
exclude-if-modified-after TIMESTAMP] [--destination-server-side-encryption {SSE-B2}] [--destination-server-side-encryptionAlgorithm
{AES256}] [--skip-newer | --replace-newer] [--delete | --keep-days DAYS] source destination
```

Required information:

- `source` - The source folder that contents will be copied from.
- `destination` - The destination folder to which the contents will be copied.

Notes:

This call can be used to upload or download files in from a source to a destination. It will work in a few directions.

- ✓ From bucket/folder path to local folder.
 - This will be a download of the bucket/folder path contents to a local folder.

- ✓ From local folder to bucket/folder path.
 - This will be an upload of the local folder contents to a bucket/folder path.
- ✓ Additionally, it will work from one bucket/folder to another bucket/folder in the same account.
- ✗ This will NOT work for a local folder to another local folder.

One of the locations have to be a local folder and the other must be a b2 location. For the B2 location, since this call can work either to or from a bucket, the path must start with `b2://` to denote a network path to the bucket. This is also called a `B2_URI`.

When entering the `destination`, a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

This call has a LOT of optional arguments, so these will be split into several sections.

`--no-progress` will simply do the same thing but will not show a progress for the upload or download.

`--dry-run` will simulate the syncing process without actually uploading or downloading data.

`--exclude-all-symlinks` will skip symlinks when syncing from a local source.

Deletions at the destination:

`--delete` | `--keep-days DAYS`

- `--delete` will delete whatever data already exists in the destination folder. If this argument is not included, the contents of the source will be copied and **added** to the destination.
- `--keep-days DAYS` will delete **versions** of any file older than the specified age in DAYS.

`--allow-empty-source` will respect an empty folder and the contents from the destination will be deleted if the source is indeed empty.

These first two are a helpful way to clear a bucket entirely. In which case, you can sync an empty folder to a bucket with the options `--allow-empty-source` and `--delete`. More information on using `b2 sync` to delete the contents of a bucket can be found here: <https://help.backblaze.com/hc/en-us/articles/225556127-How-Can-I-Easily-Delete-All-Files-in-a-Bucket->

When you use these two optional arguments to delete stuff in a folder or the bucket itself, it will also upload a placeholder file called `“.DS_Store”`.

`--skip-newer` | `--replace-newer` - Files at the `source` that have a newer modification time are always copied to the `destination`. If the destination file is newer, the default is to report an error and stop.

- `--skip-newer` will tell the sync job to ignore files with a newer modification time at the `destination`.
- `--replace-newer` will tell the sync job to replace files with a newer modification time at the `destination` with the files with an older modification time at the `source`.

To make the `destination` EXACTLY match the `source`, you can use the option `--replace-newer` and `--delete`.

```
b2 sync --delete --replace-newer source destination
```

To make the destination match the source, but retain previous versions for 30 days:

```
b2 sync --keep-days 30 --replace-newer source destination
```

Sync Exclusions

`--exclude-regex REGEX` can be used to specify a “regular expression” to exclude certain files, file types or folders from being uploaded or downloaded. The regular expression should be formatted a certain way.

`--include-regex REGEX` can be used to have `--exclude-regex` make exceptions for the rules specified in its REGEX.

`--exclude-dir-regex REGEX` - Folders excluded by `--exclude-dir-regex` will not be included even if it matches a REGEX specified by `--include-regex`.

REGEX Formatting

macOS	<p>Files</p> <pre>'(./file.txt)'</pre> <pre>'(./file1.txt) (./file2.txt)'</pre> <p>To target file by extension:</p> <pre>'(./*.txt)'</pre> <p>To also target files/files by extension at the top level of the folder</p> <pre>'(./**file.txt)'</pre> <pre>'(./**.*.txt)'</pre>	<p>Folders</p> <pre>'(./folder)'</pre> <pre>'(./folder1) (./folder2)'</pre> <p>To also target items at the top level of the folder</p> <pre>'(./**folder)'</pre>
Windows	<p>Files</p> <pre>'(.*\file.txt)'</pre> <pre>'(.*\file1.txt) (.*\file2.txt)'</pre> <p>To target file by extension:</p> <pre>'(.*\.txt)'</pre> <p>To also target files/files by extension at the top level of the folder</p> <pre>'(.**file.txt)'</pre> <pre>'(.**.*.txt)'</pre>	<p>Folders</p> <pre>'(.*\folder)'</pre> <pre>'(.*\folder1) (.*\folder2)'</pre> <p>To also target items at the top level of the folder</p> <pre>'(.***folder)'</pre>

Notes	<ul style="list-style-type: none">• Folders and files can be specified in the same REGEX input.• Folder and file names can be partial and everything with that partial name will be targeted.• By default, REGEX will not target folders or files in the top level of the source location you are syncing.• In order to have REGEX work for folders or files in the top level of the source location, there are a couple ways to achieve this. Some examples:<ul style="list-style-type: none">◦ Targeting the top level items per REGEX item Adding the wildcard right after the forward slash will make the sync not target those items at the top level of the source as well as other places the REGEX shows up in the source. <pre>'(./**cars) (./**.*.DS_Store) (./**.*food)'</pre> will target ALL folders called <code>cars</code> and ALL files called <code>.DS_Store</code>. That goes for everything at the top level as well. Since <code>(./**.*food)</code> does not have the <code>*</code> right after the forward slash, it will not be excluded if it is at the top level of the source. <pre>'(./**.*cat.jpg)'</pre> will target ALL <code>cat.jpg</code> files (both top and sub level).
-------	--

'(.*cat.jpg)' will target ONLY sub level cat.jpg files.

- **Targeting *ONLY* the top level items for the entire REGEX input**

Adding square brackets to the REGEX wrapping will make the sync target every item in the REGEX at the top level of the source and will not target the matching items in subfolders. This method does not require any of the REGEX items to have the * wildcard.

'[(.*cars)|(.*DS_Store)|(.*food)]' will target only TOP LEVEL folders called *cars*, TOP LEVEL files called *.DS_Store* and TOP LEVEL folders called *food*.

`--exclude-if-modified-after` **TIMESTAMP** - You can specify `--exclude-if-modified-after` to selectively ignore file versions (including hide markers) which were synced after given time (for local source) or ignore only specific file versions (for b2 source). Ignored files or file versions will not be taken for consideration during sync. The time should be given as a seconds timestamp (e.g. "1367900664") If you need milliseconds precision, put it after the decimal point (e.g. "1367900664.152")

File Comparisons

By default, a file is the same if the name and modification time are the same as a file that already exists in the destination. If a file is the same, it will not be synced.

`--compare-versions {none,modTime,size}` will tell the sync job to determine whether a file is the same based on your choice for this optional argument.

- none: Comparison using the file name only
- modTime: Comparison using the modification time (default)
- size: Comparison using the file size

`--compare-threshold` **MILLIS** is to be used when `--compare-versions` is set to either `modTime` or `size`. This will tell `--compare-versions` to do a fuzzy comparison of files within a threshold of all files that already exist in a bucket based on what `--compare-versions` is set to. If a file's modification time or size is within what is set with `--compare-threshold`, it will be considered the same and will not be synced.


Example of compareVersions/compareThreshold

We have a bucket with a file dog.jpeg and it is 10 KB or 10000 Bytes plus a few other files that don't change. If the original file is changed and is now 10.05 KB or 10050 Bytes, a sync job with `b2 sync` will determine that this file is different because of its difference in size.

If this 50 byte difference is not big enough for the user to want to upload via `b2 sync`, it can be ignored with `--compare-versions` if it is set to `size` and with `--compare-threshold` if the threshold is set to **100** (bytes). Example of the call is below.

Server-Side Encryption

`--destination-server-side-encryption {SSE-B2}` - When the destination is a B2 bucket, this will encrypt all new files that get uploaded to it. This will not encrypt anything that already exists in the bucket.

 If the intention is to encrypt everything in the bucket with this call, it is better to creating a new bucket and then syncing the local folder to the new bucket. Or even a new folder within the bucket and resyncing the local source to the new folder. Everything newly uploaded will be encrypted.

`--destination-server-side-encryptionAlgorithm {AES256}` - This option is not necessary as calling the sync job with `--destination-server-side-encryption`

Note: The output of the sync job when using `--destination-server-side-encryption` will look exactly the same as a standard sync job.

Output:

```
> b2 sync /Users/matto/Desktop/files b2://mattdata/synctest

> upload .DS_Store

> upload soda.jpg

> upload water.jpg
```

Further examples	Output
<p><code>b2 sync --dry-run --exclude-regex '(*Screen Shot)(*.DS_Store)'</code></p> <p><code>/Users/matto/Desktop/synctest b2://mattybucket/synctest7</code></p> <p>This excludes objects with a specific file path. Can be full or partial paths. Also works for files.</p>	<pre>matto@MacBook-Pro ~ % b2 sync --dryRun --excludeRegex '(*Screen Shot)(*.DS_Store)' /Users/matto/Desktop/synctest b2://mattybucket/synctest7 upload .DS_Store upload example/animals/dog.jpeg upload example/food/pasta.jpeg</pre>
<p>The destination in this example has a couple files that are unchanged, and a file called <i>bread.jpeg</i> that is 40,100 bytes. The source currently has that <i>bread.jpeg</i> file but it was modified and is now 39,300 bytes.</p> <p>If we run:</p> <ul style="list-style-type: none"> <code>b2 sync --dry-run /Users/matto/Desktop/files\ 2 b2://himatt/test0</code> <p>It will want to upload the modified <i>bread.jpeg</i> since it is different in size.</p>	<pre>matto@MacBook-Pro ~ % b2 sync --dryRun /Users/matto/Desktop/files\ 2 b2://himatt/test0 upload bread.jpeg</pre>
<p>However, if you use <code>--compare-versions</code> and <code>--compare-threshold</code> this difference can be ignored and the modified <i>bread.jpeg</i> file will not be uploaded. The difference between the files is 800 bytes, so we can set the threshold to 1000 bytes so that the newer and smaller version is not accounted for in the sync job.</p> <p>The call looks like this:</p> <ul style="list-style-type: none"> <code>b2 sync --dry-run --compare-versions size --compare-threshold 1000 /Users/matto/Desktop/files\ 2 b2://himatt/test0</code> <p>Notice that the sync job no longer wants to pick up the modified <i>bread.jpeg</i>.</p>	<pre>matto@MacBook-Pro ~ % b2 sync --dryRun --compareVersions size --compareThreshold 1000 /Users/matto/Desktop/files\ 2 b2://himatt/test0 matto@MacBook-Pro ~ %</pre>

b2 update-bucket

```
b2 update-bucket [-h] [--bucket-info BUCKETINFO] [--cors-rules CORSRULES] [--lifecycle-rules LIFECYCLERULES] [--default-server-side-encryption {SSE-B2,none}] [--default-server-side-encryption-algorithm {AES256}] bucketName bucketType
```

Category:

Subcommand:

Full command:

Required information:

- `bucketName` - Name of the bucket that will be updated.
- `bucketType` - Choice of bucket type.
 - Public
 - `allPublic` - Applications will not need an authorization token to download from bucket.

- Private
 - `allPrivate` - Application will need an authorization token to download from bucket.

Notes:

In the B2 docs for the CLI, the required `bucketType` argument show `[allPublic | allPrivate]` instead and is wrapped in square brackets, but the choice between `allPublic` and `allPrivate` is required. You must choose one or the other and not both.

Optional arguments:

The following optional arguments need values in JSON format. There are some rules for JSON formatting.

- If adding more than one VARIABLE:VALUE pair, it must be separated with a comma.
- If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

`[--bucket-info BUCKETINFO]` - These are user chosen and user defined values. They can be named anything and be anything the user wants.

BUCKETINFO Format:

The `BUCKETINFO` part must be replaced with a JSON formatted as `{["VARIABLE": "VALUE"]}`

If a VALUE is to be treated as an integer (number), it shouldn't be in double quotes. If a VALUE is to be treated as a string (text), it should be in double quotes

For more information: [!\[\]\(d5d7044e5caf6907399af2dced8d6ff8_img.jpg\) Cloud Storage Buckets](#)

`[--cors-rules CORSRULES]` - This is to set CORS Rules (Cross-Origin Resource Sharing).

CORSRULES Format:

The `CORSRULES` part must be replaced with a JSON formatted as `{[["VARIABLE": "VALUE", "VARIABLE2": "VALUE2"]]}`

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [] in the wrapping.

If a value should be nonspecific and refer to "all" (such as all Origins or all Files), then the wild card * can be used. It would look like this:

`"VARIABLE": "*"`

For `allowedHeaders`, `allowedOrigins` and `allowedOperations`, the variables must be wrapped in ["]. Like this: `"allowedHeaders": ["range"]`

For more CORS Rules information: [!\[\]\(097cdd6c9c875b64d9b8c9a2409491c4_img.jpg\) Cloud Storage Cross-Origin Resource Sharing Rules](#)

`[--lifecycle-rules LIFECYCLERULES]` - This is to set Lifecycle Rules. For more information: [!\[\]\(f9f168a9979beed8b01f8750d577d508_img.jpg\) Cloud Storage Lifecycle Rules](#)

LIFECYCLERULES Format:

The `LIFECYCLERULES` part must be replaced with a JSON formatted as `{[["VARIABLE": "VALUE", "VARIABLE2": "VALUE2"]]}`

Since this JSON requires multiple VARIABLE:VALUE pairs, it must have those square braces [] in the wrapping.

If a value should be nonspecific and refer to "all" (such as all Origins or all Files), then the wild card * can be used. It would look like this:

`"VARIABLE": "*"`

For more Lifecycle Rules information: [!\[\]\(608bfbc50031d613907ec08333d4afc7_img.jpg\) Cloud Storage Lifecycle Rules](#)

Default Server-Side Encryption:

The option `--default-server-side-encryption {SSE-B2,none}` will set Server-Side Encryption for the bucket.

- If the value chosen is `SSE-B2` it will enable Server-Side Encryption for the bucket. This will not encrypt non-encrypted objects that are already in the bucket.
- If the value chosen is `none` it will disable Server-Side Encryption for the bucket. This will not decrypt encrypted objects that are already in the bucket.

The option `--default-server-side-encryption-algorithm {AES256}` doesn't have an effect on the call as enabling Server-Side Encryption with `--default-server-side-encryption SSE-B2` will set `--default-server-side-encryption-algorithm` to `AES256` automatically.

For more information: [Server-Side Encryption](#)

Output:

Input

```
b2 update-bucket himatt allPublic
```

```
> b2 update-bucket himatt allPublic
> {
  "accountId": "2afe93h8bd60",
  "bucketId": "327a4f1ee3e42c586bcd0619",
  "bucketInfo": {
    "bucketcoowner": "kellie",
    "bucketowner": "matt"
  },
  "bucketName": "himatt",
  "bucketType": "allPublic",
  "corsRules": [
    {
      "allowedHeaders": [
        "range"
      ],
      "allowedOperations": [
        "b2_download_file_by_id",
        "b2_download_file_by_name"
      ],
      "allowedOrigins": [
        "https"
      ],
      "corsRuleName": "downloadFromAnyOrigin",
      "exposeHeaders": [
        "x-bz-content-sha1"
      ],
```

```

    "maxAgeSeconds": 3600

  },

],

"defaultRetention": {

  "mode": null

},

"default-server-side-encryption": {

  "mode": null

},

"isFileLockEnabled": false,

"lifecycleRules": [],

"options": [],

"replication": {

  "asReplicationDestination": null,

  "asReplicationSource": null

},

"revision": 19

}

```

Further examples:

```
> b2 update-bucket --bucket-info '{"people": "cool", "pizza": "yes"}' mattclibucket2 allPublic
```

> Does the same as above but adds bucketInfo.

```
> "bucketInfo": {

  "bucketcoowner": "kellie",

  "bucketowner": "matt"

},
```

```
> b2 update-bucket --cors-rules '[{"corsRuleName": "downloadFromAnyOrigin", "allowedOrigins": ["https"], "allowedHeaders": ["range"], "allowedOperations": ["b2_download_file_by_id", "b2_download_file_by_name"], "exposeHeaders": ["x-bz-content-sha1"], "maxAgeSeconds": 3600}]' mattybucke1 allPublic
```

> Does the same as above but adds cors rules.

```
> "corsRules": [

  {

    "allowedHeaders": [

      "range"

    ],

    "allowedOperations": [

      "b2_download_file_by_id",
```



```

        "b2_download_file_by_name"

    ],

    "allowedOrigins": [

        "https"

    ],

    "corsRuleName": "downloadFromAnyOrigin",

    "exposeHeaders": [

        "x-bz-content-sha1"

    ],

    "maxAgeSeconds": 3600

    }

    ],

```

```

> b2 update-bucket --lifecycle-rules '[{"daysFromHidingToDeleting": 1, "daysFromUploadingToHiding": null, "fileNamePrefix":
"chunks/"}]' mattybux allPublic

```

> Does the same as above but adds lifecycle rules.

```

> "lifecycleRules": [

    {

        "daysFromHidingToDeleting": 1,

        "daysFromUploadingToHiding": null,

        "fileNamePrefix": "chunks/"

    }

    ],

```

```

> b2 update-bucket --default-server-side-encryption SSE-B2 machu-bux2 allPublic

```

> Does the same as above but sets server side encryption.

```

> "defaultServerSideEncryption": {

    "algorithm": "AES256",

    "mode": "SSE-B2"

    },

```

b2 upload-file

```

b2 upload-file [-h] [--no-progress] [--quiet] [--content-type CONTENTTYPE] [--min-part-size MINPARTSIZE] [--sha1 SHA1] [--threads
THREADS] [--info INFO] [--destination-server-side-encryption {SSE-B2}] [--destination-server-side-encryptionAlgorithm {AES256}]
bucketName localFilePath b2FileName

```

Category: file

Subcommand: upload

Full command: `b2 file upload bucketName localFilePath b2FileName`

Required information:

- `bucketName` - Name of the bucket that a file will be uploaded to.
- `localFilePath` - Full file path + file name of the file that will be uploaded.
- `b2FileName` - User chosen file name.
 - Does not have to match the original.

Notes:

Unlike the normal B2 docs call, the CLI version of this call will not require the *authorization token* nor the *upload URL* input. It will handle the `b2 get-upload-url` call and plug in the URL and upload auth token automatically.

When entering the `b2FileName`, a prefix can be added as a folder path. If the folder path does not exist yet, a new folder or new set of folders will be created when the call is sent.

Optional arguments:

`--no-progress` will simply do the same thing but will not show a progress bar for the download.

`--quiet` will do the same thing but will not list either of the URLs for the object in the bucket.

`--content-type CONTENTTYPE` will set the content-type for the file uploaded. If this is left out of the upload call, it will default according to the file's extension.

`--sha1 SHA1` allows you to tell the call the sha1 of the file to be uploaded. Without this argument, the call will automatically calculate it for you. This is useful to make the processing and computing for many upload calls take less time in total. If making this call one at a time, it's easier to let the call do it for you.

`--info INFO` - These are variables and values chosen and defined by the user.

- This is to be formatted as `VARIABLE=VALUE`. Example below.
- If more than one `VARIABLE=VALUE` is to be specified, and additional `--info` entry must be made. Example below.

Large Files:

Large files can be from 5MB to 10TB.

If the file being uploaded is 200MB or larger, the CLI will default to treating the file as a large file and upload it in multiple threads. If not specified by the optional `--threads THREADS`, the CLI will default to 10 threads. Using the optional `--threads THREADS` argument on a file smaller than 200MB will not have an effect and the CLI will ignore that argument. You can set the size of the parts with `--min-part-size MINPARTSIZE`. If left out, the CLI will choose the "recommendedPartSize" for you.

A file (larger than 200MB) will be broken up into parts and each part will be uploaded on its own thread to reduce the amount of time taken to upload the file in whole.

When uploading a large file in parts, the CLI will not look any different as it does when uploading a small file under 200MB.

- ▼ If you want to test uploading a large file but don't have one large enough.

You can create a dummy file of any size by running this command:

```
mkfile -n <size> <file name>
```

Example:

```
mkfile -n 100M TestFile100M
```

Output: The call will give back information similar to `b2 get-file-info`.

```
> b2 upload-file mattwower /Users/matto/Desktop/notes.txt notes.txt
```

```

> /Users/matto/Desktop/ps2.txt: 100%| 265/265 [00:01<00:00, 196B/s]URL by
file name: https://f000.backblazeb2.com/file/mattwower/notes.txt

> URL by fileId: https://f000.backblazeb2.com/b2api/v2/b2_download_file_by_id?
fileId=4_z020a0ffe8375rhd97b9d0619_f11844e0d831c3323_d20240529_m224048_c000_v0001412_t0042_u01717022448458

> {
  "accountId": "2afe93h8bd60",
  "action": "upload",
  "bucketId": "020a0ffe8375rhd97b9d0619",
  "contentMd5": "a13248167c43623aa4af3bf4dd62f993",
  "contentSha1": "fc89e9acb65d987312889ee5ecdcd064e0e46b47",
  "contentType": "text/plain",
  "fileId":
"4_z020a0ffe8375rhd97b9d0619_f11844e0d831c3323_d20240529_m224048_c000_v0001412_t0042_u01717022448458",
  "fileInfo": {
    "src_last_modified_millis": "1713904475821"
  },
  "fileName": "notes.txt",
  "fileRetention": {
    "mode": null,
    "retainUntilTimestamp": null
  },
  "legalHold": null,
  "replicationStatus": null,
  "serverSideEncryption": {
    "algorithm": "AES256",
    "mode": "SSE-B2"
  },
  "size": 265,
  "uploadTimestamp": 1717022448458
}

>/Users/matto/Desktop/notes.txt: 100%| 265/265 [00:01<00:00, 175B/s]

```

```

> b2 upload-file -info hello=matt -info matt=hello machu-bucket /Users/matto/Desktop/files\ 4/animals/dog.jpeg pic3.jpeg

> Does the same as above but adds info to the uploaded file that can be passed back to the user when info on the file is requested.

> "fileInfo": {
  "hello": "matt"
},

```

b2 update-file-legal-hold

```
b2 update-file-legal-hold [-h] [fileName] fileId {on,off}
```

Category: file

Subcommand: update

Full command: b2 file update --legal-hold {on,off} B2_URI

Required information:

- b2id://fileId - File ID of the file we will toggle legal hold for.
 - When using the subcommand version, this requires the B2_URI, e.g. b2://bucketName/file.txt or b2id://fileId.
- {on,off} - User's choice for if legal hold should be on or off.

Notes:

This call will toggle [Object Lock](#) on a specified file and it can't be deleted until it is disabled.

In order to use this command, the bucket itself needs to have fileLockEnabled=true. Required capabilities for the application key are: writeFileLegalHolds and (if file name is not provided) readFiles.

Optional arguments:

Specifying the [fileName] is more efficient than leaving it out. If you omit the [fileName], it requires an initial query to B2 to get the file name, before making the call to delete the file. This extra query requires the capability bypassGovernance.

Output:

If the API accepts the call and is successful, it will return nothing.

If the API doesn't accept the call or fails, it will return the error and prompt that pertains to the issue (example below)

Input

```
> b2 update-file-legal-hold 4_z521acf4e63289j687b5d0619_f111ad32d4d6d1cd7_d20210624_m202933_c000_v0001076_t0026
on

> Since a successful call doesn't pass anything back, the example below is from b2 get-file-info .

> ...

  "legalHold": "on",

  ...
```

b2 update-file-retention

```
b2 update-file-retention [-h] [--retain-until TIMESTAMP] [--bypass-governance] [fileName] fileId {governance,compliance,none}
```

Category: file

Subcommand: update

Full command: b2 file update --bypass-governance B2_URI {governance,compliance,none}

Required information:

- b2id://fileId - File ID for the file we want to update file retention for.

- When using the subcommand version, this requires the `B2_URI`, e.g. `b2://bucketName/file.txt` or `b2id://fileId`.
- {governance,compliance,none} - Choose between `governance`, `compliance` or `none`.

Notes:

This call will set a certain amount of time of [Object Lock](#) for a specified file and it can't be deleted for that amount of time.

In order to use this command, the bucket itself needs to have `fileLockEnabled=true`. Required capabilities for the application key are: `writeFileLegalHolds` and (if file name is not provided) `readFiles`.

File retention is synonymous to "Object Lock" "Immutability" and "File Lock"

Optional arguments:

`[--retain-until TIMESTAMP]` will lock the file from being deleted from a bucket until the specified time. The timestamp must be an integer representing milliseconds since "epoch".

In addition to the above, if the type of file retention is set to `governance`, in order to disable or shorten file retention, the user must be using an application key with the capability `bypassGovernance` and the call must also pass the `[--bypass-governance]` argument in the call.

If the type of file retention is set to `compliance`, the user will not be able to remove or shorten file retention for the file. The file must reach the end of the retention period set before it can be deleted.

Epoch dates & timestamps:

A date in epoch format is an integer that represents milliseconds from January 1, 1970 to a specified date.

For example, if you want a file to be locked until January 1, 2024, you would set `--retain-until 1704096000` where 1704096000 is the number of milliseconds from January 1, 1970 to January 1, 2024. [This](#) is a useful tool to calculate a date converted to epoch.

Output:

If the API accepts the call and is successful, it will return nothing.

If the API doesn't accept the call or fails, it will return the error and prompt that pertains to the issue

Input

```
> b2 update-file-retention --retain-until 3704096000000
4_z521acf4e63289j687b5d0619_f111ad32d4d6d1cd7_d20210624_m202933_c000_v0001076_t0026 governance

> Since a successful call doesn't pass anything back, the example below is from b2 get-file-info after. This will

> "fileRetention": {

    "mode": "governance",

    "retainUntilTimestamp": 3704096000000

  },
```