Amazon S3 Storage Providers

The S3 backend can be used with a number of different providers:

AWS S3	Home Config
Alibaba Cloud (Aliyun) Object Storage System (OSS)	Home Config
Ceph	Home Config
DigitalOcean Spaces	Home Config
Dreamhost	Home Config
IBM COS S3	Home Config
Minio	Home Config
Scaleway	Home Config
StackPath	Home Config
Tencent Cloud Object Storage (COS)	Home Config
Wasabi	Home Config

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`.

Once you have made a remote (see the provider specific section above) you can use it like this:

See all buckets

```
rclone lsd remote:
```

Make 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
```

AWS S3

Here is an example of making an s3 configuration. First run

```
rclone config
```

This will guide you through an interactive setup process.

```
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
name> remote
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Amazon S3 Compliant Storage Providers including AWS, Ceph, Dreamhost, IBM COS, Minio, ar
   \ "s3"
[snip]
Storage> s3
Choose your S3 provider.
Choose a number from below, or type in your own value
 1 / Amazon Web Services (AWS) S3
   \ "AWS"
 2 / Ceph Object Storage
   \ "Ceph"
 3 / Digital Ocean Spaces
   \ "DigitalOcean"
 4 / Dreamhost DreamObjects
   \ "Dreamhost"
 5 / IBM COS S3
   \ "IBMCOS"
 6 / Minio Object Storage
   \ "Minio"
 7 / Wasabi Object Storage
   \ "Wasabi"
 8 / Any other S3 compatible provider
   \ "Other"
provider> 1
Get AWS credentials from runtime (environment variables or EC2/ECS meta data if no env vars).
Choose a number from below, or type in your own value
 1 / Enter AWS credentials in the next step
   \ "false"
 2 / Get AWS credentials from the environment (env vars or IAM)
   \ "true"
env_auth> 1
AWS Access Key ID - leave blank for anonymous access or runtime credentials.
access_key_id> XXX
AWS Secret Access Key (password) - leave blank for anonymous access or runtime credentials.
secret_access_key> YYY
Region to connect to.
Choose a number from below, or type in your own value
   / The default endpoint - a good choice if you are unsure.
 1 | US Region, Northern Virginia, or Pacific Northwest.
   | Leave location constraint empty.
   \ "us-east-1"
   / US East (Ohio) Region
 2 | Needs location constraint us-east-2.
```

```
   \ "us-east-2"
   / US West (Oregon) Region
 3 | Needs location constraint us-west-2.
   \ "us-west-2"
   / US West (Northern California) Region
 4 | Needs location constraint us-west-1.
   \ "us-west-1"
   / Canada (Central) Region
 5 | Needs location constraint ca-central-1.
   \ "ca-central-1"
   / EU (Ireland) Region
 6 | Needs location constraint EU or eu-west-1.
   \ "eu-west-1"
   / EU (London) Region
 7 | Needs location constraint eu-west-2.
   \ "eu-west-2"
   / EU (Frankfurt) Region
 8 | Needs location constraint eu-central-1.
   \ "eu-central-1"
   / Asia Pacific (Singapore) Region
 9 | Needs location constraint ap-southeast-1.
   \ "ap-southeast-1"
   / Asia Pacific (Sydney) Region
10 | Needs location constraint ap-southeast-2.
   \ "ap-southeast-2"
   / Asia Pacific (Tokyo) Region
11 | Needs location constraint ap-northeast-1.
   \ "ap-northeast-1"
   / Asia Pacific (Seoul)
12 | Needs location constraint ap-northeast-2.
   \ "ap-northeast-2"
   / Asia Pacific (Mumbai)
13 | Needs location constraint ap-south-1.
   \ "ap-south-1"
   / Asia Pacific (Hong Kong) Region
14 | Needs location constraint ap-east-1.
   \ "ap-east-1"
   / South America (Sao Paulo) Region
15 | Needs location constraint sa-east-1.
   \ "sa-east-1"
region> 1
Endpoint for S3 API.
Leave blank if using AWS to use the default endpoint for the region.
endpoint>
Location constraint - must be set to match the Region. Used when creating buckets only.
Choose a number from below, or type in your own value
 1 / Empty for US Region, Northern Virginia, or Pacific Northwest.
   \ ""
 2 / US East (Ohio) Region.
   \ "us-east-2"
 3 / US West (Oregon) Region.
```

```
   \ "us-west-2"
 4 / US West (Northern California) Region.
   \ "us-west-1"
 5 / Canada (Central) Region.
   \ "ca-central-1"
 6 / EU (Ireland) Region.
   \ "eu-west-1"
 7 / EU (London) Region.
   \ "eu-west-2"
 8 / EU Region.
   \ "EU"
 9 / Asia Pacific (Singapore) Region.
   \ "ap-southeast-1"
10 / Asia Pacific (Sydney) Region.
   \ "ap-southeast-2"
11 / Asia Pacific (Tokyo) Region.
   \ "ap-northeast-1"
12 / Asia Pacific (Seoul)
   \ "ap-northeast-2"
13 / Asia Pacific (Mumbai)
   \ "ap-south-1"
14 / Asia Pacific (Hong Kong)
   \ "ap-east-1"
15 / South America (Sao Paulo) Region.
   \ "sa-east-1"
location_constraint> 1
Canned ACL used when creating buckets and/or storing objects in S3.
For more info visit https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html#canned-
Choose a number from below, or type in your own value
 1 / Owner gets FULL_CONTROL. No one else has access rights (default).
   \ "private"
 2 / Owner gets FULL_CONTROL. The AllUsers group gets READ access.
   \ "public-read"
   / Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access.
 3 | Granting this on a bucket is generally not recommended.
   \ "public-read-write"
 4 / Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access.
   \ "authenticated-read"
   / Object owner gets FULL_CONTROL. Bucket owner gets READ access.
 5 | If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.
   \ "bucket-owner-read"
   / Both the object owner and the bucket owner get FULL_CONTROL over the object.
 6 | If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.
   \ "bucket-owner-full-control"
acl> 1
The server-side encryption algorithm used when storing this object in S3.
Choose a number from below, or type in your own value
 1 / None
   \ ""
 2 / AES256
   \ "AES256"
```

```
server_side_encryption> 1
The storage class to use when storing objects in S3.
Choose a number from below, or type in your own value
 1 / Default
   \ ""
 2 / Standard storage class
   \ "STANDARD"
 3 / Reduced redundancy storage class
   \ "REDUCED_REDUNDANCY"
 4 / Standard Infrequent Access storage class
   \ "STANDARD_IA"
 5 / One Zone Infrequent Access storage class
   \ "ONEZONE_IA"
 6 / Glacier storage class
   \ "GLACIER"
 7 / Glacier Deep Archive storage class
   \ "DEEP_ARCHIVE"
 8 / Intelligent-Tiering storage class
   \ "INTELLIGENT_TIERING"
storage_class> 1
Remote config
--------------------
[remote]
type = s3
provider = AWS
env_auth = false
access_key_id = XXX
secret_access_key = YYY
region = us-east-1
endpoint =
location_constraint =
acl = private
server_side_encryption =
storage_class =
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d>
```

Modified time

The modified time is stored as metadata on the object as `X-Amz-Meta-Mtime` as floating point since the epoch accurate to 1 ns.

If the modification time needs to be updated rclone will attempt to perform a server side copy to update the modification if the object can be copied in a single part. In the case the object is larger than 5Gb or is in Glacier or Glacier Deep Archive storage the object will be uploaded rather than copied.

Note that reading this from the object takes an additional `HEAD` request as the metadata isn't returned in object listings.

Reducing costs

Avoiding HEAD requests to read the modification time

By default rclone will use the modification time of objects stored in S3 for syncing. This is stored in object metadata which unfortunately takes an extra HEAD request to read which can be expensive (in time and money).

The modification time is used by default for all operations that require checking the time a file was last updated. It allows rclone to treat the remote more like a true filesystem, but it is inefficient on S3 because it requires an extra API call to retrieve the metadata.

The extra API calls can be avoided when syncing (using `rclone sync` or `rclone copy`) in a few different ways, each with its own tradeoffs.

- `--size-only`
  - Only checks the size of files.
  - Uses no extra transactions.
  - If the file doesn't change size then rclone won't detect it has changed.
  - `rclone sync --size-only /path/to/source s3:bucket`
- `--checksum`
  - Checks the size and MD5 checksum of files.
  - Uses no extra transactions.
  - The most accurate detection of changes possible.
  - Will cause the source to read an MD5 checksum which, if it is a local disk, will cause lots of disk activity.
  - If the source and destination are both S3 this is the **recommended** flag to use for maximum efficiency.
  - `rclone sync --checksum /path/to/source s3:bucket`
- `--update --use-server-modtime`
  - Uses no extra transactions.
  - Modification time becomes the time the object was uploaded.
  - For many operations this is sufficient to determine if it needs uploading.
  - Using `--update` along with `--use-server-modtime`, avoids the extra API call and uploads files whose local modification time is newer than the time it was last uploaded.
  - Files created with timestamps in the past will be missed by the sync.
  - `rclone sync --update --use-server-modtime /path/to/source s3:bucket`

These flags can and should be used in combination with `--fast-list` - see below.

If using `rclone mount` or any command using the VFS (eg `rclone serve`) commands then you might want to consider using the VFS flag`--no-modtime` which will stop rclone reading the modification time for every object. You could also use `--use-server-modtime` if you are happy with the modification times of the objects being the time of upload.

Avoiding GET requests to read directory listings

Rclone's default directory traversal is to process each directory individually. This takes one API call per directory. Using the `--fast-list`flag will read all info about the the objects into memory first using a smaller number of API calls (one per 1000 objects). See the rclone docs for more details.

```
rclone sync --fast-list --checksum /path/to/source s3:bucket
```

`--fast-list` trades off API transactions for memory use. As a rough guide rclone uses 1k of memory per object stored, so using `--fast-list` on a sync of a million objects will use roughly 1 GB of RAM.

If you are only copying a small number of files into a big repository then using `--no-traverse` is a good idea. This finds objects directly instead of through directory listings. You can do a "top-up" sync very cheaply by using `--max-age` and `--no-traverse` to copy only recent files, eg

```
rclone copy --min-age 24h --no-traverse /path/to/source s3:bucket
```

You'd then do a full `rclone sync` less often.

Note that `--fast-list` isn't required in the top-up sync.

Hashes

For small objects which weren't uploaded as multipart uploads (objects sized below `--s3-upload-cutoff` if uploaded with rclone) rclone uses the `ETag:` header as an MD5 checksum.

However for objects which were uploaded as multipart uploads or with server side encryption (SSE-AWS or SSE-C) the `ETag` header is no longer the MD5 sum of the data, so rclone adds an additional piece of metadata `X-Amz-Meta-Md5chksum` which is a base64 encoded MD5 hash (in the same format as is required for `Content-MD5`).

For large objects, calculating this hash can take some time so the addition of this hash can be disabled with `--s3-disable-checksum`. This will mean that these objects do not have an MD5 checksum.

Note that reading this from the object takes an additional `HEAD` request as the metadata isn't returned in object listings.

Cleanup

If you run `rclone cleanup s3:bucket` then it will remove all pending multipart uploads older than 24 hours. You can use the `-i` flag to see exactly what it will do. If you want more control over the expiry date then run `rclone backend cleanup s3:bucket -o max-age=1h` to expire all uploads older than one hour. You can use `rclone backend list-multipart-uploads s3:bucket` to see the pending multipart uploads.

Restricted filename characters

S3 allows any valid UTF-8 string as a key.

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

The following characters are replaced since these are problematic when dealing with the REST API:

Character	Value	Replacement
NUL	0x00	␀
/	0x2F	／

The encoding will also encode these file names as they don't seem to work with the SDK properly:

File name	Replacement
.	．
..	．．

Multipart uploads

rclone supports multipart uploads with S3 which means that it can upload files bigger than 5GB.

Note that files uploaded *both* with multipart upload *and* through crypt remotes do not have MD5 sums.

rclone switches from single part uploads to multipart uploads at the point specified by `--s3-upload-cutoff`. This can be a maximum of 5GB and a minimum of 0 (ie always upload multipart files).

The chunk sizes used in the multipart upload are specified by `--s3-chunk-size` and the number of chunks uploaded concurrently is specified by `--s3-upload-concurrency`.

Multipart uploads will use `--transfers` * `--s3-upload-concurrency` * `--s3-chunk-size` extra memory. Single part uploads to not use extra memory.

Single part transfers can be faster than multipart transfers or slower depending on your latency from S3 - the more latency, the more likely single part transfers will be faster.

Increasing `--s3-upload-concurrency` will increase throughput (8 would be a sensible value) and increasing `--s3-chunk-size` also increases throughput (16M would be sensible). Increasing either of these will use more memory. The default values are high enough to gain most of the possible performance without using too much memory.

Buckets and Regions

With Amazon S3 you can list buckets (`rclone lsd`) using any region, but you can only access the content of a bucket from the region it was created in. If you attempt to access a bucket from the wrong region, you will get an error, `incorrect region, the bucket is not in 'XXX' region`.

Authentication

There are a number of ways to supply `rclone` with a set of AWS credentials, with and without using the environment.

The different authentication methods are tried in this order:

- Directly in the rclone configuration file (`env_auth = false` in the config file):
    - `access_key_id` and `secret_access_key` are required.

    - `session_token` can be optionally set when using AWS STS.
- Runtime configuration (`env_auth = true` in the config file):
    - Export the following environment variables before running `rclone`:
        - Access Key ID: `AWS_ACCESS_KEY_ID` or `AWS_ACCESS_KEY`
        - Secret Access Key: `AWS_SECRET_ACCESS_KEY` or `AWS_SECRET_KEY`
        - Session Token: `AWS_SESSION_TOKEN` (optional)
    - Or, use a named profile:
        - Profile files are standard files used by AWS CLI tools
        - By default it will use the profile in your home directory (e.g. `~/.aws/credentials` on unix based systems) file and the "default" profile, to change set these environment variables:
            - `AWS_SHARED_CREDENTIALS_FILE` to control which file.
            - `AWS_PROFILE` to control which profile to use.
    - Or, run `rclone` in an ECS task with an IAM role (AWS only).
    - Or, run `rclone` on an EC2 instance with an IAM role (AWS only).
    - Or, run `rclone` in an EKS pod with an IAM role that is associated with a service account (AWS only).

If none of these option actually end up providing `rclone` with AWS credentials then S3 interaction will be non-authenticated (see below).

S3 Permissions

When using the `sync` subcommand of `rclone` the following minimum permissions are required to be available on the bucket being written to:

- `ListBucket`
- `DeleteObject`
- `GetObject`
- `PutObject`
- `PutObjectACL`

When using the `lsd` subcommand, the `ListAllMyBuckets` permission is required.

Example policy:

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Effect": "Allow",
            "Principal": {
                "AWS": "arn:aws:iam::USER_SID:user/USER_NAME"
            },
            "Action": [
                "s3:ListBucket",
                "s3:DeleteObject",
                "s3:GetObject",
                "s3:PutObject",
                "s3:PutObjectAcl"
            ],
            "Resource": [
              "arn:aws:s3:::BUCKET_NAME/*",
              "arn:aws:s3:::BUCKET_NAME"
            ]
        },
        {
            "Effect": "Allow",
            "Action": "s3:ListAllMyBuckets",
            "Resource": "arn:aws:s3:::*"
        }
    ]
}
```

Notes on above:

1. This is a policy that can be used when creating bucket. It assumes that `USER_NAME` has been created.
2. The Resource entry must include both resource ARNs, as one implies the bucket and the other implies the bucket's objects.

For reference, here's an Ansible script that will generate one or more buckets that will work with `rclone sync`.

Key Management System (KMS)

If you are using server-side encryption with KMS then you must make sure rclone is configured with `server_side_encryption = aws:kms`otherwise you will find you can't transfer small objects - these will create checksum errors.

Glacier and Glacier Deep Archive

You can upload objects using the glacier storage class or transition them to glacier using a lifecycle policy. The bucket can still be synced or copied into normally, but if rclone tries to access data from the glacier storage class you will see an error like below.

```
2017/09/11 19:07:43 Failed to sync: failed to open source object: Object in GLACIER, restore
```

In this case you need to restore the object(s) in question before using rclone.

Note that rclone only speaks the S3 API it does not speak the Glacier Vault API, so rclone cannot directly access Glacier Vaults.

Standard Options

Here are the standard options specific to s3 (Amazon S3 Compliant Storage Providers including AWS, Alibaba, Ceph, Digital Ocean, Dreamhost, IBM COS, Minio, and Tencent COS).

--s3-provider

Choose your S3 provider.

- Config: provider
- Env Var: RCLONE_S3_PROVIDER
- Type: string
- Default: ""
- Examples:
  - "AWS"
    - Amazon Web Services (AWS) S3
  - "Alibaba"
    - Alibaba Cloud Object Storage System (OSS) formerly Aliyun
  - "Ceph"
    - Ceph Object Storage
  - "DigitalOcean"
    - Digital Ocean Spaces
  - "Dreamhost"
    - Dreamhost DreamObjects
  - "IBMCOS"
    - IBM COS S3
  - "Minio"
    - Minio Object Storage
  - "Netease"
    - Netease Object Storage (NOS)
  - "Scaleway"
    - Scaleway Object Storage
  - "StackPath"
    - StackPath Object Storage
  - "TencentCOS"
    - Tencent Cloud Object Storage (COS)
  - "Wasabi"
    - Wasabi Object Storage
  - "Other"
    - Any other S3 compatible provider

--s3-env-auth

Get AWS credentials from runtime (environment variables or EC2/ECS meta data if no env vars). Only applies if access_key_id and secret_access_key is blank.

- Config: env_auth
- Env Var: RCLONE_S3_ENV_AUTH
- Type: bool
- Default: false
- Examples:
    - "false"
        - Enter AWS credentials in the next step
    - "true"
        - Get AWS credentials from the environment (env vars or IAM)

--s3-access-key-id

AWS Access Key ID. Leave blank for anonymous access or runtime credentials.

- Config: access_key_id
- Env Var: RCLONE_S3_ACCESS_KEY_ID
- Type: string
- Default: ""

--s3-secret-access-key

AWS Secret Access Key (password) Leave blank for anonymous access or runtime credentials.

- Config: secret_access_key
- Env Var: RCLONE_S3_SECRET_ACCESS_KEY
- Type: string
- Default: ""

--s3-region

Region to connect to.

- Config: region
- Env Var: RCLONE_S3_REGION
- Type: string
- Default: ""
- Examples:
    - "us-east-1"
        - The default endpoint - a good choice if you are unsure.
        - US Region, Northern Virginia, or Pacific Northwest.
        - Leave location constraint empty.
    - "us-east-2"
        - US East (Ohio) Region
        - Needs location constraint us-east-2.
    - "us-west-1"
        - US West (Northern California) Region
        - Needs location constraint us-west-1.

- "us-west-2"
  - US West (Oregon) Region
  - Needs location constraint us-west-2.
- "ca-central-1"
  - Canada (Central) Region
  - Needs location constraint ca-central-1.
- "eu-west-1"
  - EU (Ireland) Region
  - Needs location constraint EU or eu-west-1.
- "eu-west-2"
  - EU (London) Region
  - Needs location constraint eu-west-2.
- "eu-west-3"
  - EU (Paris) Region
  - Needs location constraint eu-west-3.
- "eu-north-1"
  - EU (Stockholm) Region
  - Needs location constraint eu-north-1.
- "eu-south-1"
  - EU (Milan) Region
  - Needs location constraint eu-south-1.
- "eu-central-1"
  - EU (Frankfurt) Region
  - Needs location constraint eu-central-1.
- "ap-southeast-1"
  - Asia Pacific (Singapore) Region
  - Needs location constraint ap-southeast-1.
- "ap-southeast-2"
  - Asia Pacific (Sydney) Region
  - Needs location constraint ap-southeast-2.
- "ap-northeast-1"
  - Asia Pacific (Tokyo) Region
  - Needs location constraint ap-northeast-1.
- "ap-northeast-2"
  - Asia Pacific (Seoul)
  - Needs location constraint ap-northeast-2.
- "ap-northeast-3"
  - Asia Pacific (Osaka-Local)
  - Needs location constraint ap-northeast-3.
- "ap-south-1"
  - Asia Pacific (Mumbai)
  - Needs location constraint ap-south-1.
- "ap-east-1"
  - Asia Pacific (Hong Kong) Region
  - Needs location constraint ap-east-1.
- "sa-east-1"
  - South America (Sao Paulo) Region
  - Needs location constraint sa-east-1.
- "me-south-1"
  - Middle East (Bahrain) Region
  - Needs location constraint me-south-1.
- "af-south-1"
  - Africa (Cape Town) Region
  - Needs location constraint af-south-1.

    - "cn-north-1"
        - China (Beijing) Region
        - Needs location constraint cn-north-1.
    - "cn-northwest-1"
        - China (Ningxia) Region
        - Needs location constraint cn-northwest-1.
    - "us-gov-east-1"
        - AWS GovCloud (US-East) Region
        - Needs location constraint us-gov-east-1.
    - "us-gov-west-1"
        - AWS GovCloud (US) Region
        - Needs location constraint us-gov-west-1.

--s3-region

Region to connect to.

- Config: region
- Env Var: RCLONE_S3_REGION
- Type: string
- Default: ""
- Examples:
    - "nl-ams"
        - Amsterdam, The Netherlands
    - "fr-par"
        - Paris, France

--s3-region

Region to connect to. Leave blank if you are using an S3 clone and you don't have a region.

- Config: region
- Env Var: RCLONE_S3_REGION
- Type: string
- Default: ""
- Examples:
    - ""
        - Use this if unsure. Will use v4 signatures and an empty region.
    - "other-v2-signature"
        - Use this only if v4 signatures don't work, e.g. pre Jewel/v10 CEPH.

--s3-endpoint

Endpoint for S3 API. Leave blank if using AWS to use the default endpoint for the region.

- Config: endpoint
- Env Var: RCLONE_S3_ENDPOINT
- Type: string
- Default: ""

--s3-endpoint

Endpoint for IBM COS S3 API. Specify if using an IBM COS On Premise.

- Config: endpoint
- Env Var: RCLONE_S3_ENDPOINT
- Type: string
- Default: ""
- Examples:
    - "s3.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region Endpoint
    - "s3.dal.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region Dallas Endpoint
    - "s3.wdc.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region Washington DC Endpoint
    - "s3.sjc.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region San Jose Endpoint
    - "s3.private.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region Private Endpoint
    - "s3.private.dal.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region Dallas Private Endpoint
    - "s3.private.wdc.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region Washington DC Private Endpoint
    - "s3.private.sjc.us.cloud-object-storage.appdomain.cloud"
        - US Cross Region San Jose Private Endpoint
    - "s3.us-east.cloud-object-storage.appdomain.cloud"
        - US Region East Endpoint
    - "s3.private.us-east.cloud-object-storage.appdomain.cloud"
        - US Region East Private Endpoint
    - "s3.us-south.cloud-object-storage.appdomain.cloud"
        - US Region South Endpoint
    - "s3.private.us-south.cloud-object-storage.appdomain.cloud"
        - US Region South Private Endpoint
    - "s3.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Endpoint
    - "s3.fra.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Frankfurt Endpoint
    - "s3.mil.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Milan Endpoint
    - "s3.ams.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Amsterdam Endpoint
    - "s3.private.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Private Endpoint
    - "s3.private.fra.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Frankfurt Private Endpoint
    - "s3.private.mil.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Milan Private Endpoint
    - "s3.private.ams.eu.cloud-object-storage.appdomain.cloud"
        - EU Cross Region Amsterdam Private Endpoint
    - "s3.eu-gb.cloud-object-storage.appdomain.cloud"
        - Great Britain Endpoint
    - "s3.private.eu-gb.cloud-object-storage.appdomain.cloud"
        - Great Britain Private Endpoint

- "s3.eu-de.cloud-object-storage.appdomain.cloud"
  - EU Region DE Endpoint
- "s3.private.eu-de.cloud-object-storage.appdomain.cloud"
  - EU Region DE Private Endpoint
- "s3.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional Endpoint
- "s3.tok.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional Tokyo Endpoint
- "s3.hkg.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional HongKong Endpoint
- "s3.seo.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional Seoul Endpoint
- "s3.private.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional Private Endpoint
- "s3.private.tok.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional Tokyo Private Endpoint
- "s3.private.hkg.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional HongKong Private Endpoint
- "s3.private.seo.ap.cloud-object-storage.appdomain.cloud"
  - APAC Cross Regional Seoul Private Endpoint
- "s3.jp-tok.cloud-object-storage.appdomain.cloud"
  - APAC Region Japan Endpoint
- "s3.private.jp-tok.cloud-object-storage.appdomain.cloud"
  - APAC Region Japan Private Endpoint
- "s3.au-syd.cloud-object-storage.appdomain.cloud"
  - APAC Region Australia Endpoint
- "s3.private.au-syd.cloud-object-storage.appdomain.cloud"
  - APAC Region Australia Private Endpoint
- "s3.ams03.cloud-object-storage.appdomain.cloud"
  - Amsterdam Single Site Endpoint
- "s3.private.ams03.cloud-object-storage.appdomain.cloud"
  - Amsterdam Single Site Private Endpoint
- "s3.che01.cloud-object-storage.appdomain.cloud"
  - Chennai Single Site Endpoint
- "s3.private.che01.cloud-object-storage.appdomain.cloud"
  - Chennai Single Site Private Endpoint
- "s3.mel01.cloud-object-storage.appdomain.cloud"
  - Melbourne Single Site Endpoint
- "s3.private.mel01.cloud-object-storage.appdomain.cloud"
  - Melbourne Single Site Private Endpoint
- "s3.osl01.cloud-object-storage.appdomain.cloud"
  - Oslo Single Site Endpoint
- "s3.private.osl01.cloud-object-storage.appdomain.cloud"
  - Oslo Single Site Private Endpoint
- "s3.tor01.cloud-object-storage.appdomain.cloud"
  - Toronto Single Site Endpoint
- "s3.private.tor01.cloud-object-storage.appdomain.cloud"
  - Toronto Single Site Private Endpoint
- "s3.seo01.cloud-object-storage.appdomain.cloud"
  - Seoul Single Site Endpoint
- "s3.private.seo01.cloud-object-storage.appdomain.cloud"
  - Seoul Single Site Private Endpoint
- "s3.mon01.cloud-object-storage.appdomain.cloud"
  - Montreal Single Site Endpoint

    - "s3.private.mon01.cloud-object-storage.appdomain.cloud"
        - Montreal Single Site Private Endpoint
    - "s3.mex01.cloud-object-storage.appdomain.cloud"
        - Mexico Single Site Endpoint
    - "s3.private.mex01.cloud-object-storage.appdomain.cloud"
        - Mexico Single Site Private Endpoint
    - "s3.sjc04.cloud-object-storage.appdomain.cloud"
        - San Jose Single Site Endpoint
    - "s3.private.sjc04.cloud-object-storage.appdomain.cloud"
        - San Jose Single Site Private Endpoint
    - "s3.mil01.cloud-object-storage.appdomain.cloud"
        - Milan Single Site Endpoint
    - "s3.private.mil01.cloud-object-storage.appdomain.cloud"
        - Milan Single Site Private Endpoint
    - "s3.hkg02.cloud-object-storage.appdomain.cloud"
        - Hong Kong Single Site Endpoint
    - "s3.private.hkg02.cloud-object-storage.appdomain.cloud"
        - Hong Kong Single Site Private Endpoint
    - "s3.par01.cloud-object-storage.appdomain.cloud"
        - Paris Single Site Endpoint
    - "s3.private.par01.cloud-object-storage.appdomain.cloud"
        - Paris Single Site Private Endpoint
    - "s3.sng01.cloud-object-storage.appdomain.cloud"
        - Singapore Single Site Endpoint
    - "s3.private.sng01.cloud-object-storage.appdomain.cloud"
        - Singapore Single Site Private Endpoint

--s3-endpoint

Endpoint for OSS API.

- Config: endpoint
- Env Var: RCLONE_S3_ENDPOINT
- Type: string
- Default: ""
- Examples:
    - "oss-cn-hangzhou.aliyuncs.com"
        - East China 1 (Hangzhou)
    - "oss-cn-shanghai.aliyuncs.com"
        - East China 2 (Shanghai)
    - "oss-cn-qingdao.aliyuncs.com"
        - North China 1 (Qingdao)
    - "oss-cn-beijing.aliyuncs.com"
        - North China 2 (Beijing)
    - "oss-cn-zhangjiakou.aliyuncs.com"
        - North China 3 (Zhangjiakou)
    - "oss-cn-huhehaote.aliyuncs.com"
        - North China 5 (Huhehaote)
    - "oss-cn-shenzhen.aliyuncs.com"
        - South China 1 (Shenzhen)
    - "oss-cn-hongkong.aliyuncs.com"
        - Hong Kong (Hong Kong)
    - "oss-us-west-1.aliyuncs.com"

        - US West 1 (Silicon Valley)
    - "oss-us-east-1.aliyuncs.com"
        - US East 1 (Virginia)
    - "oss-ap-southeast-1.aliyuncs.com"
        - Southeast Asia Southeast 1 (Singapore)
    - "oss-ap-southeast-2.aliyuncs.com"
        - Asia Pacific Southeast 2 (Sydney)
    - "oss-ap-southeast-3.aliyuncs.com"
        - Southeast Asia Southeast 3 (Kuala Lumpur)
    - "oss-ap-southeast-5.aliyuncs.com"
        - Asia Pacific Southeast 5 (Jakarta)
    - "oss-ap-northeast-1.aliyuncs.com"
        - Asia Pacific Northeast 1 (Japan)
    - "oss-ap-south-1.aliyuncs.com"
        - Asia Pacific South 1 (Mumbai)
    - "oss-eu-central-1.aliyuncs.com"
        - Central Europe 1 (Frankfurt)
    - "oss-eu-west-1.aliyuncs.com"
        - West Europe (London)
    - "oss-me-east-1.aliyuncs.com"
        - Middle East 1 (Dubai)

--s3-endpoint

Endpoint for Scaleway Object Storage.

- Config: endpoint
- Env Var: RCLONE_S3_ENDPOINT
- Type: string
- Default: ""
- Examples:
    - "s3.nl-ams.scw.cloud"
        - Amsterdam Endpoint
    - "s3.fr-par.scw.cloud"
        - Paris Endpoint

--s3-endpoint

Endpoint for StackPath Object Storage.

- Config: endpoint
- Env Var: RCLONE_S3_ENDPOINT
- Type: string
- Default: ""
- Examples:
    - "s3.us-east-2.stackpathstorage.com"
        - US East Endpoint
    - "s3.us-west-1.stackpathstorage.com"
        - US West Endpoint
    - "s3.eu-central-1.stackpathstorage.com"
        - EU Endpoint

--s3-endpoint

Endpoint for Tencent COS API.

- Config: endpoint
- Env Var: RCLONE_S3_ENDPOINT
- Type: string
- Default: ""
- Examples:
    - "cos.ap-beijing.myqcloud.com"
        - Beijing Region.
    - "cos.ap-nanjing.myqcloud.com"
        - Nanjing Region.
    - "cos.ap-shanghai.myqcloud.com"
        - Shanghai Region.
    - "cos.ap-guangzhou.myqcloud.com"
        - Guangzhou Region.
    - "cos.ap-nanjing.myqcloud.com"
        - Nanjing Region.
    - "cos.ap-chengdu.myqcloud.com"
        - Chengdu Region.
    - "cos.ap-chongqing.myqcloud.com"
        - Chongqing Region.
    - "cos.ap-hongkong.myqcloud.com"
        - Hong Kong (China) Region.
    - "cos.ap-singapore.myqcloud.com"
        - Singapore Region.
    - "cos.ap-mumbai.myqcloud.com"
        - Mumbai Region.
    - "cos.ap-seoul.myqcloud.com"
        - Seoul Region.
    - "cos.ap-bangkok.myqcloud.com"
        - Bangkok Region.
    - "cos.ap-tokyo.myqcloud.com"
        - Tokyo Region.
    - "cos.na-siliconvalley.myqcloud.com"
        - Silicon Valley Region.
    - "cos.na-ashburn.myqcloud.com"
        - Virginia Region.
    - "cos.na-toronto.myqcloud.com"
        - Toronto Region.
    - "cos.eu-frankfurt.myqcloud.com"
        - Frankfurt Region.
    - "cos.eu-moscow.myqcloud.com"
        - Moscow Region.
    - "cos.accelerate.myqcloud.com"
        - Use Tencent COS Accelerate Endpoint.

--s3-endpoint

Endpoint for S3 API. Required when using an S3 clone.

- Config: endpoint
- Env Var: RCLONE_S3_ENDPOINT
- Type: string
- Default: ""
- Examples:
    - "objects-us-east-1.dream.io"
        - Dream Objects endpoint
    - "nyc3.digitaloceanspaces.com"
        - Digital Ocean Spaces New York 3
    - "ams3.digitaloceanspaces.com"
        - Digital Ocean Spaces Amsterdam 3
    - "sgp1.digitaloceanspaces.com"
        - Digital Ocean Spaces Singapore 1
    - "s3.wasabisys.com"
        - Wasabi US East endpoint
    - "s3.us-west-1.wasabisys.com"
        - Wasabi US West endpoint
    - "s3.eu-central-1.wasabisys.com"
        - Wasabi EU Central endpoint

--s3-location-constraint

Location constraint - must be set to match the Region. Used when creating buckets only.

- Config: location_constraint
- Env Var: RCLONE_S3_LOCATION_CONSTRAINT
- Type: string
- Default: ""
- Examples:
    - ""
        - Empty for US Region, Northern Virginia, or Pacific Northwest.
    - "us-east-2"
        - US East (Ohio) Region.
    - "us-west-1"
        - US West (Northern California) Region.
    - "us-west-2"
        - US West (Oregon) Region.
    - "ca-central-1"
        - Canada (Central) Region.
    - "eu-west-1"
        - EU (Ireland) Region.
    - "eu-west-2"
        - EU (London) Region.
    - "eu-west-3"
        - EU (Paris) Region.
    - "eu-north-1"
        - EU (Stockholm) Region.
    - "eu-south-1"
        - EU (Milan) Region.
    - "EU"
        - EU Region.
    - "ap-southeast-1"
        - Asia Pacific (Singapore) Region.

    - "ap-southeast-2"
        - Asia Pacific (Sydney) Region.
    - "ap-northeast-1"
        - Asia Pacific (Tokyo) Region.
    - "ap-northeast-2"
        - Asia Pacific (Seoul) Region.
    - "ap-northeast-3"
        - Asia Pacific (Osaka-Local) Region.
    - "ap-south-1"
        - Asia Pacific (Mumbai) Region.
    - "ap-east-1"
        - Asia Pacific (Hong Kong) Region.
    - "sa-east-1"
        - South America (Sao Paulo) Region.
    - "me-south-1"
        - Middle East (Bahrain) Region.
    - "af-south-1"
        - Africa (Cape Town) Region.
    - "cn-north-1"
        - China (Beijing) Region
    - "cn-northwest-1"
        - China (Ningxia) Region.
    - "us-gov-east-1"
        - AWS GovCloud (US-East) Region.
    - "us-gov-west-1"
        - AWS GovCloud (US) Region.

--s3-location-constraint

Location constraint - must match endpoint when using IBM Cloud Public. For on-prem COS, do not make a selection from this list, hit enter

- Config: location_constraint
- Env Var: RCLONE_S3_LOCATION_CONSTRAINT
- Type: string
- Default: ""
- Examples:
    - "us-standard"
        - US Cross Region Standard
    - "us-vault"
        - US Cross Region Vault
    - "us-cold"
        - US Cross Region Cold
    - "us-flex"
        - US Cross Region Flex
    - "us-east-standard"
        - US East Region Standard
    - "us-east-vault"
        - US East Region Vault
    - "us-east-cold"
        - US East Region Cold
    - "us-east-flex"
        - US East Region Flex

- "us-south-standard"
  - US South Region Standard
- "us-south-vault"
  - US South Region Vault
- "us-south-cold"
  - US South Region Cold
- "us-south-flex"
  - US South Region Flex
- "eu-standard"
  - EU Cross Region Standard
- "eu-vault"
  - EU Cross Region Vault
- "eu-cold"
  - EU Cross Region Cold
- "eu-flex"
  - EU Cross Region Flex
- "eu-gb-standard"
  - Great Britain Standard
- "eu-gb-vault"
  - Great Britain Vault
- "eu-gb-cold"
  - Great Britain Cold
- "eu-gb-flex"
  - Great Britain Flex
- "ap-standard"
  - APAC Standard
- "ap-vault"
  - APAC Vault
- "ap-cold"
  - APAC Cold
- "ap-flex"
  - APAC Flex
- "mel01-standard"
  - Melbourne Standard
- "mel01-vault"
  - Melbourne Vault
- "mel01-cold"
  - Melbourne Cold
- "mel01-flex"
  - Melbourne Flex
- "tor01-standard"
  - Toronto Standard
- "tor01-vault"
  - Toronto Vault
- "tor01-cold"
  - Toronto Cold
- "tor01-flex"
  - Toronto Flex

--s3-location-constraint

Location constraint - must be set to match the Region. Leave blank if not sure. Used when creating buckets only.

- Config: location_constraint
- Env Var: RCLONE_S3_LOCATION_CONSTRAINT
- Type: string
- Default: ""

--s3-acl

Canned ACL used when creating buckets and storing or copying objects.

This ACL is used for creating objects and if bucket_acl isn't set, for creating buckets too.

For more info visit https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html#canned-acl

Note that this ACL is applied when server-side copying objects as S3 doesn't copy the ACL from the source but rather writes a fresh one.

- Config: acl
- Env Var: RCLONE_S3_ACL
- Type: string
- Default: ""
- Examples:
  - "default"
    - Owner gets Full_CONTROL. No one else has access rights (default).
  - "private"
    - Owner gets FULL_CONTROL. No one else has access rights (default).
  - "public-read"
    - Owner gets FULL_CONTROL. The AllUsers group gets READ access.
  - "public-read-write"
    - Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access.
    - Granting this on a bucket is generally not recommended.
  - "authenticated-read"
    - Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access.
  - "bucket-owner-read"
    - Object owner gets FULL_CONTROL. Bucket owner gets READ access.
    - If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.
  - "bucket-owner-full-control"
    - Both the object owner and the bucket owner get FULL_CONTROL over the object.
    - If you specify this canned ACL when creating a bucket, Amazon S3 ignores it.
  - "private"
    - Owner gets FULL_CONTROL. No one else has access rights (default). This acl is available on IBM Cloud (Infra), IBM Cloud (Storage), On-Premise COS
  - "public-read"
    - Owner gets FULL_CONTROL. The AllUsers group gets READ access. This acl is available on IBM Cloud (Infra), IBM Cloud (Storage), On-Premise IBM COS
  - "public-read-write"
    - Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access. This acl is available on IBM Cloud (Infra), On-Premise IBM COS
  - "authenticated-read"
    - Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access. Not supported on Buckets. This acl is available on IBM Cloud (Infra) and On-Premise IBM COS

--s3-server-side-encryption

The server-side encryption algorithm used when storing this object in S3.

- Config: server_side_encryption
- Env Var: RCLONE_S3_SERVER_SIDE_ENCRYPTION
- Type: string
- Default: ""
- Examples:
  - ""
    - None
  - "AES256"
    - AES256
  - "aws:kms"
    - aws:kms

--s3-sse-kms-key-id

If using KMS ID you must provide the ARN of Key.

- Config: sse_kms_key_id
- Env Var: RCLONE_S3_SSE_KMS_KEY_ID
- Type: string
- Default: ""
- Examples:
  - ""
    - None
  - "arn:aws:kms:us-east-1:*"
    - arn:aws:kms:*

--s3-storage-class

The storage class to use when storing new objects in S3.

- Config: storage_class
- Env Var: RCLONE_S3_STORAGE_CLASS
- Type: string
- Default: ""
- Examples:
  - ""
    - Default
  - "STANDARD"
    - Standard storage class
  - "REDUCED_REDUNDANCY"
    - Reduced redundancy storage class
  - "STANDARD_IA"
    - Standard Infrequent Access storage class
  - "ONEZONE_IA"
    - One Zone Infrequent Access storage class
  - "GLACIER"
    - Glacier storage class
  - "DEEP_ARCHIVE"
    - Glacier Deep Archive storage class

    - "INTELLIGENT_TIERING"
        - Intelligent-Tiering storage class

--s3-storage-class

The storage class to use when storing new objects in OSS.

- Config: storage_class
- Env Var: RCLONE_S3_STORAGE_CLASS
- Type: string
- Default: ""
- Examples:
    - ""
        - Default
    - "STANDARD"
        - Standard storage class
    - "GLACIER"
        - Archive storage mode.
    - "STANDARD_IA"
        - Infrequent access storage mode.

--s3-storage-class

The storage class to use when storing new objects in Tencent COS.

- Config: storage_class
- Env Var: RCLONE_S3_STORAGE_CLASS
- Type: string
- Default: ""
- Examples:
    - ""
        - Default
    - "STANDARD"
        - Standard storage class
    - "ARCHIVE"
        - Archive storage mode.
    - "STANDARD_IA"
        - Infrequent access storage mode.

--s3-storage-class

The storage class to use when storing new objects in S3.

- Config: storage_class
- Env Var: RCLONE_S3_STORAGE_CLASS
- Type: string
- Default: ""
- Examples:
    - ""
        - Default
    - "STANDARD"

        - The Standard class for any upload; suitable for on-demand content like streaming or CDN.
    - "GLACIER"
        - Archived storage; prices are lower, but it needs to be restored first to be accessed.

Advanced Options

Here are the advanced options specific to s3 (Amazon S3 Compliant Storage Providers including AWS, Alibaba, Ceph, Digital Ocean, Dreamhost, IBM COS, Minio, and Tencent COS).

--s3-bucket-acl

Canned ACL used when creating buckets.

For more info visit https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html#canned-acl

Note that this ACL is applied when only when creating buckets. If it isn't set then "acl" is used instead.

- Config: bucket_acl
- Env Var: RCLONE_S3_BUCKET_ACL
- Type: string
- Default: ""
- Examples:
    - "private"
        - Owner gets FULL_CONTROL. No one else has access rights (default).
    - "public-read"
        - Owner gets FULL_CONTROL. The AllUsers group gets READ access.
    - "public-read-write"
        - Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access.
        - Granting this on a bucket is generally not recommended.
    - "authenticated-read"
        - Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access.

--s3-requester-pays

Enables requester pays option when interacting with S3 bucket.

- Config: requester_pays
- Env Var: RCLONE_S3_REQUESTER_PAYS
- Type: bool
- Default: false

--s3-sse-customer-algorithm

If using SSE-C, the server-side encryption algorithm used when storing this object in S3.

- Config: sse_customer_algorithm
- Env Var: RCLONE_S3_SSE_CUSTOMER_ALGORITHM
- Type: string
- Default: ""

- Examples:
  - ""
    - None
  - "AES256"
    - AES256

--s3-sse-customer-key

If using SSE-C you must provide the secret encryption key used to encrypt/decrypt your data.

- Config: sse_customer_key
- Env Var: RCLONE_S3_SSE_CUSTOMER_KEY
- Type: string
- Default: ""
- Examples:
  - ""
    - None

--s3-sse-customer-key-md5

If using SSE-C you may provide the secret encryption key MD5 checksum (optional).

If you leave it blank, this is calculated automatically from the sse_customer_key provided.

- Config: sse_customer_key_md5
- Env Var: RCLONE_S3_SSE_CUSTOMER_KEY_MD5
- Type: string
- Default: ""
- Examples:
  - ""
    - None

--s3-upload-cutoff

Cutoff for switching to chunked upload

Any files larger than this will be uploaded in chunks of chunk_size. The minimum is 0 and the maximum is 5GB.

- Config: upload_cutoff
- Env Var: RCLONE_S3_UPLOAD_CUTOFF
- Type: SizeSuffix
- Default: 200M

--s3-chunk-size

Chunk size to use for uploading.

When uploading files larger than upload_cutoff or files with unknown size (e.g. from "rclone rcat" or uploaded with "rclone mount" or google photos or google docs) they will be uploaded as multipart uploads using this chunk size.

Note that "--s3-upload-concurrency" chunks of this size are buffered in memory per transfer.

If you are transferring large files over high-speed links and you have enough memory, then increasing this will speed up the transfers.

Rclone will automatically increase the chunk size when uploading a large file of known size to stay below the 10,000 chunks limit.

Files of unknown size are uploaded with the configured chunk_size. Since the default chunk size is 5MB and there can be at most 10,000 chunks, this means that by default the maximum size of a file you can stream upload is 48GB. If you wish to stream upload larger files then you will need to increase chunk_size.

- Config: chunk_size
- Env Var: RCLONE_S3_CHUNK_SIZE
- Type: SizeSuffix
- Default: 5M

--s3-max-upload-parts

Maximum number of parts in a multipart upload.

This option defines the maximum number of multipart chunks to use when doing a multipart upload.

This can be useful if a service does not support the AWS S3 specification of 10,000 chunks.

Rclone will automatically increase the chunk size when uploading a large file of a known size to stay below this number of chunks limit.

- Config: max_upload_parts
- Env Var: RCLONE_S3_MAX_UPLOAD_PARTS
- Type: int
- Default: 10000

--s3-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 5GB.

- Config: copy_cutoff
- Env Var: RCLONE_S3_COPY_CUTOFF
- Type: SizeSuffix
- Default: 4.656G

--s3-disable-checksum

Don't store MD5 checksum with object metadata

Normally rclone will calculate the MD5 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_S3_DISABLE_CHECKSUM
- Type: bool
- Default: false

--s3-shared-credentials-file

Path to the shared credentials file

If env_auth = true then rclone can use a shared credentials file.

If this variable is empty rclone will look for the "AWS_SHARED_CREDENTIALS_FILE" env variable. If the env value is empty it will default to the current user's home directory.

```
Linux/OSX: "$HOME/.aws/credentials"
Windows:   "%USERPROFILE%\.aws\credentials"
```

- Config: shared_credentials_file
- Env Var: RCLONE_S3_SHARED_CREDENTIALS_FILE
- Type: string
- Default: ""

--s3-profile

Profile to use in the shared credentials file

If env_auth = true then rclone can use a shared credentials file. This variable controls which profile is used in that file.

If empty it will default to the environment variable "AWS_PROFILE" or "default" if that environment variable is also not set.

- Config: profile
- Env Var: RCLONE_S3_PROFILE
- Type: string
- Default: ""

--s3-session-token

An AWS session token

- Config: session_token
- Env Var: RCLONE_S3_SESSION_TOKEN
- Type: string
- Default: ""

--s3-upload-concurrency

Concurrency for multipart uploads.

This is the number of chunks of the same file that are uploaded concurrently.

If you are uploading small numbers of large files over high-speed links and these uploads do not fully utilize your bandwidth, then increasing this may help to speed up the transfers.

- Config: upload_concurrency
- Env Var: RCLONE_S3_UPLOAD_CONCURRENCY
- Type: int
- Default: 4

--s3-force-path-style

If true use path style access if false use virtual hosted style.

If this is true (the default) then rclone will use path style access, if false then rclone will use virtual path style. See the AWS S3 docs for more info.

Some providers (e.g. AWS, Aliyun OSS, Netease COS, or Tencent COS) require this set to false - rclone will do this automatically based on the provider setting.

- Config: force_path_style
- Env Var: RCLONE_S3_FORCE_PATH_STYLE
- Type: bool
- Default: true

--s3-v2-auth

If true use v2 authentication.

If this is false (the default) then rclone will use v4 authentication. If it is set then rclone will use v2 authentication.

Use this only if v4 signatures don't work, e.g. pre Jewel/v10 CEPH.

- Config: v2_auth
- Env Var: RCLONE_S3_V2_AUTH
- Type: bool
- Default: false

--s3-use-accelerate-endpoint

If true use the AWS S3 accelerated endpoint.

See: AWS S3 Transfer acceleration

- Config: use_accelerate_endpoint
- Env Var: RCLONE_S3_USE_ACCELERATE_ENDPOINT

- Type: bool
- Default: false

--s3-leave-parts-on-error

If true avoid calling abort upload on a failure, leaving all successfully uploaded parts on S3 for manual recovery.

It should be set to true for resuming uploads across different sessions.

WARNING: Storing parts of an incomplete multipart upload counts towards space usage on S3 and will add additional costs if not cleaned up.

- Config: leave_parts_on_error
- Env Var: RCLONE_S3_LEAVE_PARTS_ON_ERROR
- Type: bool
- Default: false

--s3-list-chunk

Size of listing chunk (response list for each ListObject S3 request).

This option is also known as "MaxKeys", "max-items", or "page-size" from the AWS S3 specification. Most services truncate the response list to 1000 objects even if requested more than that. In AWS S3 this is a global maximum and cannot be changed, see AWS S3. In Ceph, this can be increased with the "rgw list buckets max chunk" option.

- Config: list_chunk
- Env Var: RCLONE_S3_LIST_CHUNK
- Type: int
- Default: 1000

--s3-no-check-bucket

If set, don't attempt to check the bucket exists or create it

This can be useful when trying to minimise the number of transactions rclone does if you know the bucket exists already.

It can also be needed if the user you are using does not have bucket creation permissions. Before v1.52.0 this would have passed silently due to a bug.

- Config: no_check_bucket
- Env Var: RCLONE_S3_NO_CHECK_BUCKET
- Type: bool
- Default: false

--s3-no-head

If set, don't HEAD uploaded objects to check integrity

This can be useful when trying to minimise the number of transactions rclone does.

Setting it means that if rclone receives a 200 OK message after uploading an object with PUT then it will assume that it got uploaded properly.

In particular it will assume:

- the metadata, including modtime, storage class and content type was as uploaded
- the size was as uploaded

It reads the following items from the response for a single part PUT:

- the MD5SUM
- The uploaded date

For multipart uploads these items aren't read.

If an source object of unknown length is uploaded then rclone **will** do a HEAD request.

Setting this flag increases the chance for undetected upload failures, in particular an incorrect size, so it isn't recommended for normal operation. In practice the chance of an undetected upload failure is very small even with this flag.

- Config: no_head
- Env Var: RCLONE_S3_NO_HEAD
- Type: bool
- Default: false

--s3-encoding

This sets the encoding for the backend.

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

- Config: encoding
- Env Var: RCLONE_S3_ENCODING
- Type: MultiEncoder
- Default: Slash,InvalidUtf8,Dot

--s3-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_S3_MEMORY_POOL_FLUSH_TIME
- Type: Duration
- Default: 1m0s

--s3-memory-pool-use-mmap

Whether to use mmap buffers in internal memory pool.

- Config: memory_pool_use_mmap
- Env Var: RCLONE_S3_MEMORY_POOL_USE_MMAP
- Type: bool
- Default: false

--s3-disable-http2

Disable usage of http2 for S3 backends

There is currently an unsolved issue with the s3 (specifically minio) backend and HTTP/2. HTTP/2 is enabled by default for the s3 backend but can be disabled here. When the issue is solved this flag will be removed.

See: https://github.com/rclone/rclone/issues/4673, https://github.com/rclone/rclone/issues/3631

- Config: disable_http2
- Env Var: RCLONE_S3_DISABLE_HTTP2
- Type: bool
- Default: false

Backend commands

Here are the commands specific to the s3 backend.

Run them with

```
rclone backend COMMAND remote:
```

The help below will explain what arguments each command takes.

See the "rclone backend" command for more info on how to pass options and arguments.

These can be run on a running backend using the rc command backend/command.

restore

Restore objects from GLACIER to normal storage

```
rclone backend restore remote: [options] [<arguments>+]
```

This command can be used to restore one or more objects from GLACIER to normal storage.

Usage Examples:

```
rclone backend restore s3:bucket/path/to/object [-o priority=PRIORITY] [-o lifetime=DAYS]
rclone backend restore s3:bucket/path/to/directory [-o priority=PRIORITY] [-o lifetime=DAYS]
rclone backend restore s3:bucket [-o priority=PRIORITY] [-o lifetime=DAYS]
```

This flag also obeys the filters. Test first with -i/--interactive or --dry-run flags

```
rclone -i backend restore --include "*.txt" s3:bucket/path -o priority=Standard
```

All the objects shown will be marked for restore, then

```
rclone backend restore --include "*.txt" s3:bucket/path -o priority=Standard
```

It returns a list of status dictionaries with Remote and Status keys. The Status will be OK if it was successful or an error message if not.

```
[
    {
        "Status": "OK",
        "Path": "test.txt"
    },
    {
        "Status": "OK",
        "Path": "test/file4.txt"
    }
]
```

Options:

- "description": The optional description for the job.
- "lifetime": Lifetime of the active copy in days
- "priority": Priority of restore: Standard|Expedited|Bulk

list-multipart-uploads

List the unfinished multipart uploads

```
rclone backend list-multipart-uploads remote: [options] [<arguments>+]
```

This command lists the unfinished multipart uploads in JSON format.

```
rclone backend list-multipart s3:bucket/path/to/object
```

It returns a dictionary of buckets with values as lists of unfinished multipart uploads.

You can call it with no bucket in which case it lists all bucket, with a bucket or with a bucket and path.

```
{
  "rclone": [
    {
      "Initiated": "2020-06-26T14:20:36Z",
      "Initiator": {
        "DisplayName": "XXX",
        "ID": "arn:aws:iam::XXX:user/XXX"
      },
      "Key": "KEY",
      "Owner": {
        "DisplayName": null,
        "ID": "XXX"
      },
      "StorageClass": "STANDARD",
      "UploadId": "XXX"
    }
  ],
  "rclone-1000files": [],
  "rclone-dst": []
}
```


cleanup

Remove unfinished multipart uploads.

```
rclone backend cleanup remote: [options] [<arguments>+]
```

This command removes unfinished multipart uploads of age greater than max-age which defaults to 24 hours.

Note that you can use -i/--dry-run with this command to see what it would do.

```
rclone backend cleanup s3:bucket/path/to/object
rclone backend cleanup -o max-age=7w s3:bucket/path/to/object
```

Durations are parsed as per the rest of rclone, 2h, 7d, 7w etc.

Options:

- "max-age": Max age of upload to delete

Anonymous access to public buckets

If you want to use rclone to access a public bucket, configure with a blank `access_key_id` and `secret_access_key`. Your config should end up looking like this:

```
[anons3]
type = s3
provider = AWS
env_auth = false
access_key_id =
secret_access_key =
region = us-east-1
endpoint =
location_constraint =
acl = private
server_side_encryption =
storage_class =
```

Then use it as normal with the name of the public bucket, e.g.

```
rclone lsd anons3:1000genomes
```

You will be able to list and copy data but not upload it.

Ceph

Ceph is an open source unified, distributed storage system designed for excellent performance, reliability and scalability. It has an S3 compatible object storage interface.

To use rclone with Ceph, configure as above but leave the region blank and set the endpoint. You should end up with something like this in your config:

```
[ceph]
type = s3
provider = Ceph
env_auth = false
access_key_id = XXX
secret_access_key = YYY
region =
endpoint = https://ceph.endpoint.example.com
location_constraint =
acl =
server_side_encryption =
storage_class =
```

If you are using an older version of CEPH, e.g. 10.2.x Jewel, then you may need to supply the parameter `--s3-upload-cutoff 0` or put this in the config file as `upload_cutoff 0` to work around a bug which causes uploading of small files to fail.

Note also that Ceph sometimes puts `/` in the passwords it gives users. If you read the secret access key using the command line tools you will get a JSON blob with the `/` escaped as `\/`. Make sure you only write `/` in the secret access key.

Eg the dump from Ceph looks something like this (irrelevant keys removed).

```
{
    "user_id": "xxx",
    "display_name": "xxxx",
    "keys": [
        {
            "user": "xxx",
            "access_key": "xxxxxx",
            "secret_key": "xxxxxx\/xxxx"
        }
    ],
}
```

Because this is a json dump, it is encoding the `/` as `\/`, so if you use the secret key as `xxxxxx/xxxx` it will work fine.

Dreamhost

Dreamhost DreamObjects is an object storage system based on CEPH.

To use rclone with Dreamhost, configure as above but leave the region blank and set the endpoint. You should end up with something like this in your config:

```
[dreamobjects]
type = s3
provider = DreamHost
env_auth = false
access_key_id = your_access_key
secret_access_key = your_secret_key
region =
endpoint = objects-us-west-1.dream.io
location_constraint =
acl = private
server_side_encryption =
storage_class =
```

DigitalOcean Spaces

Spaces is an S3-interoperable object storage service from cloud provider DigitalOcean.

To connect to DigitalOcean Spaces you will need an access key and secret key. These can be retrieved on the "Applications & API" page of the DigitalOcean control panel. They will be needed when prompted by `rclone config` for your `access_key_id` and `secret_access_key`.

When prompted for a `region` or `location_constraint`, press enter to use the default value. The region must be included in the `endpoint`setting (e.g. `nyc3.digitaloceanspaces.com`). The default values can be used for other settings.

Going through the whole process of creating a new remote by running `rclone config`, each prompt should be answered as shown below:

```
Storage> s3
env_auth> 1
access_key_id> YOUR_ACCESS_KEY
secret_access_key> YOUR_SECRET_KEY
region>
endpoint> nyc3.digitaloceanspaces.com
location_constraint>
acl>
storage_class>
```

The resulting configuration file should look like:

```
[spaces]
type = s3
provider = DigitalOcean
env_auth = false
access_key_id = YOUR_ACCESS_KEY
secret_access_key = YOUR_SECRET_KEY
region =
endpoint = nyc3.digitaloceanspaces.com
location_constraint =
acl =
server_side_encryption =
storage_class =
```

Once configured, you can create a new Space and begin copying files. For example:

```
rclone mkdir spaces:my-new-space
rclone copy /path/to/files spaces:my-new-space
```

IBM COS (S3)

Information stored with IBM Cloud Object Storage is encrypted and dispersed across multiple geographic locations, and accessed through an implementation of the S3 API. This service makes use of the distributed storage technologies provided by IBM's Cloud Object Storage System (formerly Cleversafe). For more information visit: (http://www.ibm.com/cloud/object-storage)

To configure access to IBM COS S3, follow the steps below:

1. Run rclone config and select n for a new remote.

```
        2018/02/14 14:13:11 NOTICE: Config file "C:\\Users\\a\\.config\\rclone\\rclone.conf"
        No remotes found - make a new one
        n) New remote
        s) Set configuration password
        q) Quit config
        n/s/q> n
```

2. Enter the name for the configuration

```
        name> <YOUR NAME>
```

3. Select "s3" storage.

```
Choose a number from below, or type in your own value
        1 / Alias for an existing remote
        \ "alias"
        2 / Amazon Drive
        \ "amazon cloud drive"
        3 / Amazon S3 Complaint Storage Providers (Dreamhost, Ceph, Minio, IBM COS)
        \ "s3"
        4 / Backblaze B2
        \ "b2"
[snip]
        23 / http Connection
    \ "http"
Storage> 3
```

4. Select IBM COS as the S3 Storage Provider.

```
Choose the S3 provider.
Choose a number from below, or type in your own value
         1 / Choose this option to configure Storage to AWS S3
           \ "AWS"
         2 / Choose this option to configure Storage to Ceph Systems
         \ "Ceph"
         3 /  Choose this option to configure Storage to Dreamhost
     \ "Dreamhost"
   4 / Choose this option to the configure Storage to IBM COS S3
         \ "IBMCOS"
         5 / Choose this option to the configure Storage to Minio
     \ "Minio"
         Provider>4
```

5. Enter the Access Key and Secret.

```
        AWS Access Key ID - leave blank for anonymous access or runtime credentials.
        access_key_id> <>
        AWS Secret Access Key (password) - leave blank for anonymous access or runtime creder
        secret_access_key> <>
```

6. Specify the endpoint for IBM COS. For Public IBM COS, choose from the option below. For On Premise IBM COS, enter an endpoint address.

```
        Endpoint for IBM COS S3 API.
        Specify if using an IBM COS On Premise.
        Choose a number from below, or type in your own value
         1 / US Cross Region Endpoint
           \ "s3-api.us-geo.objectstorage.softlayer.net"
         2 / US Cross Region Dallas Endpoint
           \ "s3-api.dal.us-geo.objectstorage.softlayer.net"
         3 / US Cross Region Washington DC Endpoint
           \ "s3-api.wdc-us-geo.objectstorage.softlayer.net"
         4 / US Cross Region San Jose Endpoint
           \ "s3-api.sjc-us-geo.objectstorage.softlayer.net"
         5 / US Cross Region Private Endpoint
           \ "s3-api.us-geo.objectstorage.service.networklayer.com"
         6 / US Cross Region Dallas Private Endpoint
           \ "s3-api.dal-us-geo.objectstorage.service.networklayer.com"
         7 / US Cross Region Washington DC Private Endpoint
           \ "s3-api.wdc-us-geo.objectstorage.service.networklayer.com"
         8 / US Cross Region San Jose Private Endpoint
           \ "s3-api.sjc-us-geo.objectstorage.service.networklayer.com"
         9 / US Region East Endpoint
           \ "s3.us-east.objectstorage.softlayer.net"
        10 / US Region East Private Endpoint
           \ "s3.us-east.objectstorage.service.networklayer.com"
        11 / US Region South Endpoint
[snip]
        34 / Toronto Single Site Private Endpoint
           \ "s3.tor01.objectstorage.service.networklayer.com"
        endpoint>1
```

7. Specify a IBM COS Location Constraint. The location constraint must match endpoint when using IBM Cloud Public. For on-prem COS, do not make a selection from this list, hit enter

```
         1 / US Cross Region Standard
           \ "us-standard"
         2 / US Cross Region Vault
           \ "us-vault"
         3 / US Cross Region Cold
           \ "us-cold"
         4 / US Cross Region Flex
           \ "us-flex"
         5 / US East Region Standard
           \ "us-east-standard"
         6 / US East Region Vault
           \ "us-east-vault"
         7 / US East Region Cold
           \ "us-east-cold"
         8 / US East Region Flex
           \ "us-east-flex"
         9 / US South Region Standard
           \ "us-south-standard"
        10 / US South Region Vault
           \ "us-south-vault"
[snip]
        32 / Toronto Flex
           \ "tor01-flex"
location_constraint>1
```

9. Specify a canned ACL. IBM Cloud (Storage) supports "public-read" and "private". IBM Cloud(Infra) supports all the canned ACLs. On-Premise COS supports all the canned ACLs.

```
Canned ACL used when creating buckets and/or storing objects in S3.
For more info visit https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html#canned-
Choose a number from below, or type in your own value
      1 / Owner gets FULL_CONTROL. No one else has access rights (default). This acl is avail
      \ "private"
      2  / Owner gets FULL_CONTROL. The AllUsers group gets READ access. This acl is availabl
      \ "public-read"
      3 / Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access. This acl is
      \ "public-read-write"
      4  / Owner gets FULL_CONTROL. The AuthenticatedUsers group gets READ access. Not suppor
      \ "authenticated-read"
acl> 1
```

12. Review the displayed configuration and accept to save the "remote" then quit. The config file should look like this

```
[xxx]
type = s3
Provider = IBMCOS
access_key_id = xxx
secret_access_key = yyy
endpoint = s3-api.us-geo.objectstorage.softlayer.net
location_constraint = us-standard
acl = private
```

13. Execute rclone commands

```
1)      Create a bucket.
        rclone mkdir IBM-COS-XREGION:newbucket
2)      List available buckets.
        rclone lsd IBM-COS-XREGION:
        -1 2017-11-08 21:16:22        -1 test
        -1 2018-02-14 20:16:39        -1 newbucket
3)      List contents of a bucket.
        rclone ls IBM-COS-XREGION:newbucket
        18685952 test.exe
4)      Copy a file from local to remote.
        rclone copy /Users/file.txt IBM-COS-XREGION:newbucket
5)      Copy a file from remote to local.
        rclone copy IBM-COS-XREGION:newbucket/file.txt .
6)      Delete a file on remote.
        rclone delete IBM-COS-XREGION:newbucket/file.txt
```

Minio

Minio is an object storage server built for cloud application developers and devops.

It is very easy to install and provides an S3 compatible server which can be used by rclone.

To use it, install Minio following the instructions here.

When it configures itself Minio will print something like this

```
Endpoint:  http://192.168.1.106:9000  http://172.23.0.1:9000
AccessKey: USWUXHGYZQYFYFFIT3RE
SecretKey: MOJRH0mkL1IPauahWITSVvyDrQbEEIwljvmxdq03
Region:    us-east-1
SQS ARNs:  arn:minio:sqs:us-east-1:1:redis arn:minio:sqs:us-east-1:2:redis

Browser Access:
   http://192.168.1.106:9000  http://172.23.0.1:9000

Command-line Access: https://docs.minio.io/docs/minio-client-quickstart-guide
   $ mc config host add myminio http://192.168.1.106:9000 USWUXHGYZQYFYFFIT3RE MOJRH0mkL1IPau

Object API (Amazon S3 compatible):
   Go:         https://docs.minio.io/docs/golang-client-quickstart-guide
   Java:       https://docs.minio.io/docs/java-client-quickstart-guide
   Python:     https://docs.minio.io/docs/python-client-quickstart-guide
   JavaScript: https://docs.minio.io/docs/javascript-client-quickstart-guide
   .NET:       https://docs.minio.io/docs/dotnet-client-quickstart-guide

Drive Capacity: 26 GiB Free, 165 GiB Total
```

These details need to go into `rclone config` like this. Note that it is important to put the region in as stated above.

```
env_auth> 1
access_key_id> USWUXHGYZQYFYFFIT3RE
secret_access_key> MOJRH0mkL1IPauahWITSVvyDrQbEEIwljvmxdq03
region> us-east-1
endpoint> http://192.168.1.106:9000
location_constraint>
server_side_encryption>
```

Which makes the config file look like this

```
[minio]
type = s3
provider = Minio
env_auth = false
access_key_id = USWUXHGYZQYFYFFIT3RE
secret_access_key = MOJRH0mkL1IPauahWITSVvyDrQbEEIwljvmxdq03
region = us-east-1
endpoint = http://192.168.1.106:9000
location_constraint =
server_side_encryption =
```

So once set up, for example to copy files into a bucket

```
rclone copy /path/to/files minio:bucket
```

Scaleway

Scaleway The Object Storage platform allows you to store anything from backups, logs and web assets to documents and photos. Files can be dropped from the Scaleway console or transferred through our API and CLI or using any S3-compatible tool.

Scaleway provides an S3 interface which can be configured for use with rclone like this:

```
[scaleway]
type = s3
provider = Scaleway
env_auth = false
endpoint = s3.nl-ams.scw.cloud
access_key_id = SCWXXXXXXXXXXXXXX
secret_access_key = 1111111-2222-3333-44444-55555555555555
region = nl-ams
location_constraint =
acl = private
server_side_encryption =
storage_class =
```

Wasabi

Wasabi is a cloud-based object storage service for a broad range of applications and use cases. Wasabi is designed for individuals and organizations that require a high-performance, reliable, and secure data storage infrastructure at minimal cost.

Wasabi provides an S3 interface which can be configured for use with rclone like this.

```
No remotes found - make a new one
n) New remote
s) Set configuration password
n/s> n
name> wasabi
Type of storage to configure.
Choose a number from below, or type in your own value
[snip]
XX / Amazon S3 (also Dreamhost, Ceph, Minio)
   \ "s3"
[snip]
Storage> s3
Get AWS credentials from runtime (environment variables or EC2/ECS meta data if no env vars).
Choose a number from below, or type in your own value
 1 / Enter AWS credentials in the next step
   \ "false"
 2 / Get AWS credentials from the environment (env vars or IAM)
   \ "true"
env_auth> 1
AWS Access Key ID - leave blank for anonymous access or runtime credentials.
access_key_id> YOURACCESSKEY
AWS Secret Access Key (password) - leave blank for anonymous access or runtime credentials.
secret_access_key> YOURSECRETACCESSKEY
Region to connect to.
Choose a number from below, or type in your own value
   / The default endpoint - a good choice if you are unsure.
 1 | US Region, Northern Virginia, or Pacific Northwest.
   | Leave location constraint empty.
   \ "us-east-1"
[snip]
region> us-east-1
Endpoint for S3 API.
Leave blank if using AWS to use the default endpoint for the region.
Specify if using an S3 clone such as Ceph.
endpoint> s3.wasabisys.com
Location constraint - must be set to match the Region. Used when creating buckets only.
Choose a number from below, or type in your own value
 1 / Empty for US Region, Northern Virginia, or Pacific Northwest.
   \ ""
[snip]
location_constraint>
Canned ACL used when creating buckets and/or storing objects in S3.
For more info visit https://docs.aws.amazon.com/AmazonS3/latest/dev/acl-overview.html#canned-
Choose a number from below, or type in your own value
 1 / Owner gets FULL_CONTROL. No one else has access rights (default).
   \ "private"
[snip]
acl>
The server-side encryption algorithm used when storing this object in S3.
Choose a number from below, or type in your own value
 1 / None
```

```
   \ ""
 2 / AES256
   \ "AES256"
server_side_encryption>
The storage class to use when storing objects in S3.
Choose a number from below, or type in your own value
 1 / Default
   \ ""
 2 / Standard storage class
   \ "STANDARD"
 3 / Reduced redundancy storage class
   \ "REDUCED_REDUNDANCY"
 4 / Standard Infrequent Access storage class
   \ "STANDARD_IA"
storage_class>
Remote config
--------------------
[wasabi]
env_auth = false
access_key_id = YOURACCESSKEY
secret_access_key = YOURSECRETACCESSKEY
region = us-east-1
endpoint = s3.wasabisys.com
location_constraint =
acl =
server_side_encryption =
storage_class =
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

This will leave the config file looking like this.

```
[wasabi]
type = s3
provider = Wasabi
env_auth = false
access_key_id = YOURACCESSKEY
secret_access_key = YOURSECRETACCESSKEY
region =
endpoint = s3.wasabisys.com
location_constraint =
acl =
server_side_encryption =
storage_class =
```

Alibaba OSS

Here is an example of making an Alibaba Cloud (Aliyun) OSS configuration. First run:

```
rclone config
```

This will guide you through an interactive setup process.

```
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
name> oss
Type of storage to configure.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
[snip]
 4 / Amazon S3 Compliant Storage Providers including AWS, Alibaba, Ceph, Digital Ocean, Dream
   \ "s3"
[snip]
Storage> s3
Choose your S3 provider.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
 1 / Amazon Web Services (AWS) S3
   \ "AWS"
 2 / Alibaba Cloud Object Storage System (OSS) formerly Aliyun
   \ "Alibaba"
 3 / Ceph Object Storage
   \ "Ceph"
[snip]
provider> Alibaba
Get AWS credentials from runtime (environment variables or EC2/ECS meta data if no env vars).
Only applies if access_key_id and secret_access_key is blank.
Enter a boolean value (true or false). Press Enter for the default ("false").
Choose a number from below, or type in your own value
 1 / Enter AWS credentials in the next step
   \ "false"
 2 / Get AWS credentials from the environment (env vars or IAM)
   \ "true"
env_auth> 1
AWS Access Key ID.
Leave blank for anonymous access or runtime credentials.
Enter a string value. Press Enter for the default ("").
access_key_id> accesskeyid
AWS Secret Access Key (password)
Leave blank for anonymous access or runtime credentials.
Enter a string value. Press Enter for the default ("").
secret_access_key> secretaccesskey
Endpoint for OSS API.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
 1 / East China 1 (Hangzhou)
   \ "oss-cn-hangzhou.aliyuncs.com"
 2 / East China 2 (Shanghai)
   \ "oss-cn-shanghai.aliyuncs.com"
 3 / North China 1 (Qingdao)
   \ "oss-cn-qingdao.aliyuncs.com"
```

```
[snip]
endpoint> 1
Canned ACL used when creating buckets and storing or copying objects.

Note that this ACL is applied when server-side copying objects as S3
doesn't copy the ACL from the source but rather writes a fresh one.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
 1 / Owner gets FULL_CONTROL. No one else has access rights (default).
   \ "private"
 2 / Owner gets FULL_CONTROL. The AllUsers group gets READ access.
   \ "public-read"
   / Owner gets FULL_CONTROL. The AllUsers group gets READ and WRITE access.
[snip]
acl> 1
The storage class to use when storing new objects in OSS.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
 1 / Default
   \ ""
 2 / Standard storage class
   \ "STANDARD"
 3 / Archive storage mode.
   \ "GLACIER"
 4 / Infrequent access storage mode.
   \ "STANDARD_IA"
storage_class> 1
Edit advanced config? (y/n)
y) Yes
n) No
y/n> n
Remote config
--------------------
[oss]
type = s3
provider = Alibaba
env_auth = false
access_key_id = accesskeyid
secret_access_key = secretaccesskey
endpoint = oss-cn-hangzhou.aliyuncs.com
acl = private
storage_class = Standard
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

Tencent COS

Tencent Cloud Object Storage (COS) is a distributed storage service offered by Tencent Cloud for unstructured data. It is secure, stable, massive, convenient, low-delay and low-cost.

To configure access to Tencent COS, follow the steps below:

1. Run `rclone config` and select `n` for a new remote.

```
rclone config
No remotes found - make a new one
n) New remote
s) Set configuration password
q) Quit config
n/s/q> n
```

2. Give the name of the configuration. For example, name it 'cos'.

```
name> cos
```

3. Select `s3` storage.

```
Choose a number from below, or type in your own value
1 / 1Fichier
   \ "fichier"
 2 / Alias for an existing remote
   \ "alias"
 3 / Amazon Drive
   \ "amazon cloud drive"
 4 / Amazon S3 Compliant Storage Providers including AWS, Alibaba, Ceph, Digital Ocean, Dream
   \ "s3"

[snip]
Storage> s3
```

4. Select `TencentCOS` provider.

```
Choose a number from below, or type in your own value
1 / Amazon Web Services (AWS) S3
   \ "AWS"
[snip]
11 / Tencent Cloud Object Storage (COS)
   \ "TencentCOS"
[snip]
provider> TencentCOS
```

5. Enter your SecretId and SecretKey of Tencent Cloud.

```
Get AWS credentials from runtime (environment variables or EC2/ECS meta data if no env vars).
Only applies if access_key_id and secret_access_key is blank.
Enter a boolean value (true or false). Press Enter for the default ("false").
Choose a number from below, or type in your own value
 1 / Enter AWS credentials in the next step
   \ "false"
 2 / Get AWS credentials from the environment (env vars or IAM)
   \ "true"
env_auth> 1
AWS Access Key ID.
Leave blank for anonymous access or runtime credentials.
Enter a string value. Press Enter for the default ("").
access_key_id> AKIDxxxxxxxxxx
AWS Secret Access Key (password)
Leave blank for anonymous access or runtime credentials.
Enter a string value. Press Enter for the default ("").
secret_access_key> xxxxxxxxxx
```

6. Select endpoint for Tencent COS. This is the standard endpoint for different region.

```
 1 / Beijing Region.
   \ "cos.ap-beijing.myqcloud.com"
 2 / Nanjing Region.
   \ "cos.ap-nanjing.myqcloud.com"
 3 / Shanghai Region.
   \ "cos.ap-shanghai.myqcloud.com"
 4 / Guangzhou Region.
   \ "cos.ap-guangzhou.myqcloud.com"
[snip]
endpoint> 4
```

7. Choose acl and storage class.

```
Note that this ACL is applied when server-side copying objects as S3
doesn't copy the ACL from the source but rather writes a fresh one.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
 1 / Owner gets Full_CONTROL. No one else has access rights (default).
   \ "default"
[snip]
acl> 1
The storage class to use when storing new objects in Tencent COS.
Enter a string value. Press Enter for the default ("").
Choose a number from below, or type in your own value
 1 / Default
   \ ""
[snip]
storage_class> 1
Edit advanced config? (y/n)
y) Yes
n) No (default)
y/n> n
Remote config
--------------------
[cos]
type = s3
provider = TencentCOS
env_auth = false
access_key_id = xxx
secret_access_key = xxx
endpoint = cos.ap-guangzhou.myqcloud.com
acl = default
--------------------
y) Yes this is OK (default)
e) Edit this remote
d) Delete this remote
y/e/d> y
Current remotes:

Name                 Type
====                 ====
cos                  s3
```

Netease NOS

For Netease NOS configure as per the configurator `rclone config` setting the provider `Netease`. This will automatically set`force_path_style = false` which is necessary for it to run properly.

Limitations

`rclone about` is not supported by the S3 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.