Swift

Swift refers to OpenStack Object Storage. Commercial implementations of that being:

- Rackspace Cloud Files
- Memset Memstore
- OVH Object Storage
- Oracle Cloud Storage
- IBM Bluemix Cloud ObjectStorage Swift

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

Here is an example of making a swift 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 / OpenStack Swift (Rackspace Cloud Files, Memset Memstore, OVH)
   \ "swift"
[snip]
Storage> swift
Get swift credentials from environment variables in standard OpenStack form.
Choose a number from below, or type in your own value
 1 / Enter swift credentials in the next step
   \ "false"
 2 / Get swift credentials from environment vars. Leave other fields blank if using this.
   \ "true"
env_auth> true
User name to log in (OS_USERNAME).
user>
API key or password (OS_PASSWORD).
key>
Authentication URL for server (OS_AUTH_URL).
Choose a number from below, or type in your own value
 1 / Rackspace US
   \ "https://auth.api.rackspacecloud.com/v1.0"
 2 / Rackspace UK
   \ "https://lon.auth.api.rackspacecloud.com/v1.0"
 3 / Rackspace v2
   \ "https://identity.api.rackspacecloud.com/v2.0"
 4 / Memset Memstore UK
   \ "https://auth.storage.memset.com/v1.0"
 5 / Memset Memstore UK v2
   \ "https://auth.storage.memset.com/v2.0"
 6 / OVH
   \ "https://auth.cloud.ovh.net/v3"
auth>
User ID to log in - optional - most swift systems use user and leave this blank (v3 auth) (OS
user_id>
User domain - optional (v3 auth) (OS_USER_DOMAIN_NAME)
domain>
Tenant name - optional for v1 auth, this or tenant_id required otherwise (OS_TENANT_NAME or O
tenant>
Tenant ID - optional for v1 auth, this or tenant required otherwise (OS_TENANT_ID)
tenant_id>
Tenant domain - optional (v3 auth) (OS_PROJECT_DOMAIN_NAME)
tenant_domain>
Region name - optional (OS_REGION_NAME)
region>
```

```
Storage URL - optional (OS_STORAGE_URL)
storage_url>
Auth Token from alternate authentication - optional (OS_AUTH_TOKEN)
auth_token>
AuthVersion - optional - set to (1,2,3) if your auth URL has no version (ST_AUTH_VERSION)
auth_version>
Endpoint type to choose from the service catalogue (OS_ENDPOINT_TYPE)
Choose a number from below, or type in your own value
 1 / Public (default, choose this if not sure)
   \ "public"
 2 / Internal (use internal service net)
   \ "internal"
 3 / Admin
   \ "admin"
endpoint_type>
Remote config
--------------------
[test]
env_auth = true
user =
key =
auth =
user_id =
domain =
tenant =
tenant_id =
tenant_domain =
region =
storage_url =
auth_token =
auth_version =
endpoint_type =
--------------------
y) Yes this is OK
e) Edit this remote
d) Delete this remote
y/e/d> y
```

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

See all containers

```
rclone lsd remote:
```

Make a new container

```
rclone mkdir remote:container
```

List the contents of a container

```
rclone ls remote:container
```

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

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

Configuration from an OpenStack credentials file

An OpenStack credentials file typically looks something something like this (without the comments)

```
export OS_AUTH_URL=https://a.provider.net/v2.0
export OS_TENANT_ID=ffffffffffffffffffffffffffffffff
export OS_TENANT_NAME="1234567890123456"
export OS_USERNAME="123abc567xy"
echo "Please enter your OpenStack Password: "
read -sr OS_PASSWORD_INPUT
export OS_PASSWORD=$OS_PASSWORD_INPUT
export OS_REGION_NAME="SBG1"
if [ -z "$OS_REGION_NAME" ]; then unset OS_REGION_NAME; fi
```

The config file needs to look something like this where `$OS_USERNAME` represents the value of the `OS_USERNAME` variable - `123abc567xy` in the example above.

```
[remote]
type = swift
user = $OS_USERNAME
key = $OS_PASSWORD
auth = $OS_AUTH_URL
tenant = $OS_TENANT_NAME
```

Note that you may (or may not) need to set `region` too - try without first.

Configuration from the environment

If you prefer you can configure rclone to use swift using a standard set of OpenStack environment variables.

When you run through the config, make sure you choose `true` for `env_auth` and leave everything else blank.

rclone will then set any empty config parameters from the environment using standard OpenStack environment variables. There is a list of the variables in the docs for the swift library.

Using an alternate authentication method

If your OpenStack installation uses a non-standard authentication method that might not be yet supported by rclone or the underlying swift library, you can authenticate externally (e.g. calling manually the `openstack` commands to get a token). Then, you just need to pass the two configuration variables `auth_token` and `storage_url`. If they are both provided, the other variables are ignored. rclone will not try to authenticate but instead assume it is already authenticated and use these two variables to access the OpenStack installation.

Using rclone without a config file

You can use rclone with swift without a config file, if desired, like this:

```
source openstack-credentials-file
export RCLONE_CONFIG_MYREMOTE_TYPE=swift
export RCLONE_CONFIG_MYREMOTE_ENV_AUTH=true
rclone lsd myremote:
```

--fast-list

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

--update and --use-server-modtime

As noted below, the modified time is stored on metadata on the object. It 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 because it requires an extra API call to retrieve the metadata.

For many operations, the time the object was last uploaded to the remote is sufficient to determine if it is "dirty". By using `--update` along with `--use-server-modtime`, you can avoid the extra API call and simply upload files whose local modtime is newer than the time it was last uploaded.

Standard Options

Here are the standard options specific to swift (OpenStack Swift (Rackspace Cloud Files, Memset Memstore, OVH)).

--swift-env-auth

Get swift credentials from environment variables in standard OpenStack form.

- Config: env_auth
- Env Var: RCLONE_SWIFT_ENV_AUTH
- Type: bool
- Default: false
- Examples:
    - "false"

        - Enter swift credentials in the next step
    - "true"
        - Get swift credentials from environment vars. Leave other fields blank if using this.

--swift-user

User name to log in (OS_USERNAME).

- Config: user
- Env Var: RCLONE_SWIFT_USER
- Type: string
- Default: ""

--swift-key

API key or password (OS_PASSWORD).

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

--swift-auth

Authentication URL for server (OS_AUTH_URL).

- Config: auth
- Env Var: RCLONE_SWIFT_AUTH
- Type: string
- Default: ""
- Examples:
    - "https://auth.api.rackspacecloud.com/v1.0"
        - Rackspace US
    - "https://lon.auth.api.rackspacecloud.com/v1.0"
        - Rackspace UK
    - "https://identity.api.rackspacecloud.com/v2.0"
        - Rackspace v2
    - "https://auth.storage.memset.com/v1.0"
        - Memset Memstore UK
    - "https://auth.storage.memset.com/v2.0"
        - Memset Memstore UK v2
    - "https://auth.cloud.ovh.net/v3"
        - OVH

--swift-user-id

User ID to log in - optional - most swift systems use user and leave this blank (v3 auth) (OS_USER_ID).

- Config: user_id

- Env Var: RCLONE_SWIFT_USER_ID
- Type: string
- Default: ""

--swift-domain

User domain - optional (v3 auth) (OS_USER_DOMAIN_NAME)

- Config: domain
- Env Var: RCLONE_SWIFT_DOMAIN
- Type: string
- Default: ""

--swift-tenant

Tenant name - optional for v1 auth, this or tenant_id required otherwise (OS_TENANT_NAME or OS_PROJECT_NAME)

- Config: tenant
- Env Var: RCLONE_SWIFT_TENANT
- Type: string
- Default: ""

--swift-tenant-id

Tenant ID - optional for v1 auth, this or tenant required otherwise (OS_TENANT_ID)

- Config: tenant_id
- Env Var: RCLONE_SWIFT_TENANT_ID
- Type: string
- Default: ""

--swift-tenant-domain

Tenant domain - optional (v3 auth) (OS_PROJECT_DOMAIN_NAME)

- Config: tenant_domain
- Env Var: RCLONE_SWIFT_TENANT_DOMAIN
- Type: string
- Default: ""

--swift-region

Region name - optional (OS_REGION_NAME)

- Config: region
- Env Var: RCLONE_SWIFT_REGION
- Type: string
- Default: ""

--swift-storage-url

Storage URL - optional (OS_STORAGE_URL)

- Config: storage_url
- Env Var: RCLONE_SWIFT_STORAGE_URL
- Type: string
- Default: ""

--swift-auth-token

Auth Token from alternate authentication - optional (OS_AUTH_TOKEN)

- Config: auth_token
- Env Var: RCLONE_SWIFT_AUTH_TOKEN
- Type: string
- Default: ""

--swift-application-credential-id

Application Credential ID (OS_APPLICATION_CREDENTIAL_ID)

- Config: application_credential_id
- Env Var: RCLONE_SWIFT_APPLICATION_CREDENTIAL_ID
- Type: string
- Default: ""

--swift-application-credential-name

Application Credential Name (OS_APPLICATION_CREDENTIAL_NAME)

- Config: application_credential_name
- Env Var: RCLONE_SWIFT_APPLICATION_CREDENTIAL_NAME
- Type: string
- Default: ""

--swift-application-credential-secret

Application Credential Secret (OS_APPLICATION_CREDENTIAL_SECRET)

- Config: application_credential_secret
- Env Var: RCLONE_SWIFT_APPLICATION_CREDENTIAL_SECRET
- Type: string
- Default: ""

--swift-auth-version

AuthVersion - optional - set to (1,2,3) if your auth URL has no version (ST_AUTH_VERSION)

- Config: auth_version
- Env Var: RCLONE_SWIFT_AUTH_VERSION
- Type: int
- Default: 0

--swift-endpoint-type

Endpoint type to choose from the service catalogue (OS_ENDPOINT_TYPE)

- Config: endpoint_type
- Env Var: RCLONE_SWIFT_ENDPOINT_TYPE
- Type: string
- Default: "public"
- Examples:
  - "public"
    - Public (default, choose this if not sure)
  - "internal"
    - Internal (use internal service net)
  - "admin"
    - Admin

--swift-storage-policy

The storage policy to use when creating a new container

This applies the specified storage policy when creating a new container. The policy cannot be changed afterwards. The allowed configuration values and their meaning depend on your Swift storage provider.

- Config: storage_policy
- Env Var: RCLONE_SWIFT_STORAGE_POLICY
- Type: string
- Default: ""
- Examples:
  - ""
    - Default
  - "pcs"
    - OVH Public Cloud Storage
  - "pca"
    - OVH Public Cloud Archive

Advanced Options

Here are the advanced options specific to swift (OpenStack Swift (Rackspace Cloud Files, Memset Memstore, OVH)).

--swift-leave-parts-on-error

If true avoid calling abort upload on a failure. It should be set to true for resuming uploads across different sessions.

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

--swift-chunk-size

Above this size files will be chunked into a _segments container.

Above this size files will be chunked into a _segments container. The default for this is 5GB which is its maximum value.

- Config: chunk_size
- Env Var: RCLONE_SWIFT_CHUNK_SIZE
- Type: SizeSuffix
- Default: 5G

--swift-no-chunk

Don't chunk files during streaming upload.

When doing streaming uploads (e.g. using rcat or mount) setting this flag will cause the swift backend to not upload chunked files.

This will limit the maximum upload size to 5GB. However non chunked files are easier to deal with and have an MD5SUM.

Rclone will still chunk files bigger than chunk_size when doing normal copy operations.

- Config: no_chunk
- Env Var: RCLONE_SWIFT_NO_CHUNK
- Type: bool
- Default: false

--swift-encoding

This sets the encoding for the backend.

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

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

Modified time

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

This is a de facto standard (used in the official python-swiftclient amongst others) for storing the modification time for an object.

Restricted filename characters

Character	Value	Replacement
NUL	0x00	␀
/	0x2F	/

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

Limitations

The Swift API doesn't return a correct MD5SUM for segmented files (Dynamic or Static Large Objects) so rclone won't check or use the MD5SUM for these.

Troubleshooting

Rclone gives Failed to create file system for "remote:": Bad Request

Due to an oddity of the underlying swift library, it gives a "Bad Request" error rather than a more sensible error when the authentication fails for Swift.

So this most likely means your username / password is wrong. You can investigate further with the `--dump-bodies` flag.

This may also be caused by specifying the region when you shouldn't have (e.g. OVH).

Rclone gives Failed to create file system: Response didn't have storage url and auth token

This is most likely caused by forgetting to specify your tenant when setting up a swift remote.