

EUROfusion LTDSF S3 tutorial - short version

- 1. Quick start guide
 - 1.1. Make sure you have s3cmd installed on your computer
 - 1.2. Configure s3cmd config file according to this template
 - 1.3. Check if configuration works
 - 1.4. Copy data from local filesystem into s3 bucket
 - 1.5. Copy data from s3 bucket into local filesystem
 - 1.6. Listing data in s3 bucket
 - 1.7. Checking size of data stored on s3
 - 1.8. More info about s3cmd usage
- 2. What is S3?
 - 2.1. S3 Buckets
 - 2.2. S3 Objects
 - 2.2.1. S3 object key structure and differences from filesystem
 - 2.3. S3 Clients and API Compatibility
 - 2.4. S3 API limitations
 - 2.4.1. Command line clients
 - 2.4.1.1. s3cmd
 - 2.4.1.2. rclone
 - 2.4.1.3. AWS cli
 - 2.4.2. API libraries
 - 2.4.3. S3 Bucket policies vs Legacy ACL system
- 3. Performance
 - 3.1. Optimizing S3 access
 - 3.2. Optimizing multipart upload
 - 3.2.1. s3cmd multipart upload options
 - 3.2.2. rclone multipart upload options
 - 3.3. Performance limitations per bucket
 - 3.4. Bucket object count limits
- 4. Preparing CLI clients
 - 4.1. s3cmd
 - 4.1.1. s3cmd with multiple accounts
 - 4.2. Rclone
 - 4.2.1. Multiple accounts with rclone
 - 4.3. AWS cli
 - 4.3.1. Multiple accounts with AWS cli
- 5. Basic operations (data upload and download)
 - 5.1. Creating example directory structure used in this tutorial
 - 5.2. Example: Listing available buckets
 - 5.2.1. s3cmd
 - 5.3. Example: Listing bucket content
 - 5.3.1. s3cmd
 - 5.4. Example: Uploading file to bucket
 - 5.4.1. s3cmd
 - 5.5. Example: Uploading directory structure with data
 - 5.5.1. s3cmd
 - 5.6. Example: Downloading data from bucket
 - 5.6.1. s3cmd
 - 5.7. Example: Copying data inside s3
 - 5.7.1. s3cmd
 - 5.8. Example: Measuring data usage on a buckets
 - 5.8.1. s3cmd
 - 5.9. Example: Deleting data
 - 5.9.1. s3cmd
 - 5.10. Operations on large files (Multipart upload)
 - 5.10.1. Configuring multipart upload chunk size
 - 5.10.1.1. s3cmd
 - 5.10.1.2. rclone
 - 5.10.1.3. AWS cli
 - 5.10.2. Multipart upload example
 - 5.10.2.1. s3cmd
 - 5.10.2.2. rclone
 - 5.10.3. Listing and cleaning up unfinished multipart uploads with s3cmd
 - 5.10.4. Dealing with unfinished multipart uploads using rclone

- 5.11. Example: backing up \${HOME} to s3
 - 5.11.1. s3cmd
 - 5.11.2. rclone
 - 6. Bucket and Object permissions in S3
 - 6.1. Bucket Policies
 - 6.1.1. Bucket policy compatibility
 - 6.1.2. How users are identified in bucket policies
 - 6.2. Example: Checking bucket policy status
 - 6.2.1. s3cmd
 - 6.2.2. AWS cli
 - 6.3. Example: Setting bucket policy
 - 6.3.1. s3cmd
 - 6.3.2. AWS cli
 - 6.4. Example: Deleting bucket-policy
 - 6.4.1. s3cmd
 - 6.4.2. AWS cli
 - 6.5. Example: Giving other user Read Only access to your bucket
 - 6.6. Example: Giving other user Read Only access to part of your bucket
 - 6.7. Example: Giving other user Write Only access to part of your bucket
 - 6.8. Example: Giving other user Read Write List Delete access to part of your bucket
 - 6.9. Example: Making part of your bucket Public by enabling unauthenticated download

1. Quick start guide

This section shows only most basic commands without explanation. We recommend you read the whole tutorial to get better understanding of how s3 works.

1.1. Make sure you have s3cmd installed on your computer

1.2. Configure s3cmd config file according to this template

Default s3cmd config file path \${HOME}/.s3cfg

Configuration template

```
[default]
use_https = True
storage_class = STANDARD
host_base = https://eurofusion-2.s3.storage.psnc.pl
host_bucket = https://eurofusion-2.s3.storage.psnc.pl
bucket_location = eurofusion-2
access_key = <<< ACCESS_KEY PROVIDED BY ADMINISTRATOR >>>
secret_key = <<< SECRET_KEY PROVIDED BY ADMINISTRATOR >>>
```

1.3. Check if configuration works

If your config is correct you should see list of buckets

```
s3cmd ls
2026-03-23 11:45 s3://tutorial-bucket
2026-03-20 12:08 s3://g2bpalak
2026-03-20 12:03 s3://g2bpogo
2026-03-20 10:10 s3://g2dfigat
2026-03-20 12:09 s3://g2jwasik
2026-03-20 12:10 s3://g2knizn
2026-03-20 12:01 s3://g2michal
```

It works \o/

1.4. Copy data from local filesystem into s3 bucket

```
s3cmd sync dir1 s3://tutorial-bucket/
```

1.5. Copy data from s3 bucket into local filesystem

```
s3cmd sync s3://tutorial-bucket/ data_downloaded_from_s3/
```

1.6. Listing data in s3 bucket

Non recursive listing

```
s3cmd ls s3://tutorial-bucket/
```

Recursive listing

```
s3cmd ls -r s3://tutorial-bucket/
```

1.7. Checking size of data stored on s3

WARNING, THIS OPERATION CAN BE VERY TIME AND RESOURCE INTENSIVE ON BUCKETS WITH A LOT OF OBJECTS

```
s3cmd du -H s3://tutorial-bucket/
```

1.8. More info about s3cmd usage

You can get more information about s3cmd in the section 4 of this tutorial or in official s3cmd [manpage](#)

2. What is S3?

S3 is an object storage http API. It's a method of storing and sharing data without using a filesystem.

S3 abstracts files to Key Value pairs.

S3 is not a filesystem and should not be used as such. **There are no POSIX guarantees!**

Filesystem style access to s3 is not supported and is discouraged.

2.1. S3 Buckets

Data in S3 is organized in buckets which are like big directories or lists of objects.

Files in s3 are represented as objects. They look similar to files but are not always equivalent.

Buckets maintain tree like filesystem object hierarchies making easy to store conventionally organized datasets.

2.2. S3 Objects

Files are represented in S3 as objects stored in buckets. Each object consists of its Key, its Value (data) and metadata.

Object keys can contain / character in the keys string. These / prefix delimiters act like pseudo directories in s3. They allow us to list keys having common prefix without actually having directory objects in S3 API. It allows you to copy tree of files into s3 bucket without losing its tree like structure.

2.2.1. S3 object key structure and differences from filesystem

Example object from `s3cmd ls -r s3://tutorial-bucket` command output

```
s3cmd ls -r s3://tutorial-bucket
2026-03-23 14:08      0 s3://tutorial-bucket/aws-cli/dir1/dir13/dir131/file-0
```

In this case I've listed content of **bucket** `tutorial-bucket`

This bucket contains object with **key** `aws-cli/dir1/dir13/dir131/file-0`

Many tools and libraries glue bucket name and object key together for simplicity `tutorial-bucket/aws-cli/dir1/dir13/dir131/file-0`

On a normal filesystem we would have said that `file-0` is in a directory structure `aws-cli/dir1/dir13/dir131`

But in case of s3 these are not directories.

Directories do not exist here. Object key is a string!!! `"aws-cli/dir1/dir13/dir131/file-0"`

/ characters in object key are for our convenience. Listing partial path like `tutorial-bucket/aws-cli/dir1/` would act as string filter giving us all object with matching prefix

This is why if our test bucket contained this one object, deleting it would also "delete" "pseudo directories" in object key.

There was never real directory structure in our bucket path `aws-cli/dir1/dir13/dir131` won't exist without object `aws-cli/dir1/dir13/dir131/file-0` inside it

2.3. S3 Clients and API Compatibility

Official S3 API specifications available on [AWS website](#). In case of S3 provided by PSNC we are using Ceph RADOS Gateway S3 implementation which has slightly less functionality than official AWS version. Detailed compatibility of this s3 instance can be found here on [Ceph website](#).

Main functional difference between Official and Ceph implementation is that version we are using doesn't support [Virtual hosting of buckets](#). Some clients and libraries will require you to configure alternative "path style request" in configuration.

2.4. S3 API limitations

S3 Bucket names are unique in the s3 instance. (bucketnames on `eurofusion-2.s3.storage.psnc.pl` are unique and cannot repeat)

S3 API is not a filesystem and doesn't have POSIX guarantees (for example file locking)

S3 API doesn't have real directories

S3 API Object key has 1024 Byte length limit. Character limit in key can shrink when using extended UTF characters (not recommended)

S3 API object don't support partial writes. Each write has to overwrite whole object.

S3 API has single file upload limit of 5GiB

S3 API single file size limit can be overcome by using multipart upload method

S3 API has multipart max upload limit of 10000 parts

S3 API maximum multipart file size limit is close to 45TiB but **creating such big objects is strongly discouraged**.

2.4.1. Command line clients

We recommend 3 linux command line programs for working with S3 API. Each of them have different strengths and weaknesses.

2.4.1.1. s3cmd

This is the most user friendly and feature complete tool in our opinion. It allows for basic file upload and download. It also allows you to configure bucket policies and lifecycle policies.

2.4.1.2. rclone

This is the best tool for data transfer from and out of s3. `rc lone sync` and `rc lone copy` can transfer many files in parallel making it good for bulk data transfer.

Rclone doesn't have tools for configuring bucket policy or lifecycle policy.

2.4.1.3. AWS cli

Most feature complete but hard to use tool.

This is official AWS command line tool for interacting with S3. AWS is hostile to non AWS s3 implementations and can be sometimes hard to use.

2.4.2. API libraries

S3 client library exists for almost any mainstream programming language. If your project requires it you can interact with S3 API directly using one of these libraries.

Most popular libraries are first party libraries provided by AWS and minio S3 client library.

We do not provide programming support when it comes to use of these libraries

2.4.3. S3 Bucket policies vs Legacy ACL system

In s3 API exist 2 kinds of permission mechanisms

Legacy ACL system and bucket policies

Legacy ACL system is very complicated, troublesome to troubleshoot and applies permissions per object

Bucket-Policies are applied at bucket level, apply to specific prefixes and are much easier to work with.

Legacy ACL system is not guaranteed to work in the future versions and we strongly discourage it's use.

3. Performance

3.1. Optimizing S3 access

LTDSF S3 is a HDD based storage system. It performs the best with sequential read and write workloads.

Uploading few larger files is easier on the system.

Uploading many small files is harder on the system.

Storing and listing huge amounts of objects in a single bucket is also difficult for the system and is discouraged.

We encourage users to package their data in a way which will work best with these limitations.

3.2. Optimizing multipart upload

Different s3 clients have different s3 multipart upload default values.

3.2.1. s3cmd multipart upload options

For example s3cmd default multipart configuration looks like this

```
enable_multipart = True
multipart_chunk_size_mb = 15
multipart_max_chunks = 10000
```

It means that any file larger than 15MiB will be transferred via multipart upload method and not simple 1 request upload.

It also limits use to maximum file size of 146GiB

In some cases you may want to increase size data send to s3 in a single request to achieve better upload performance or upload larger files.

To achieve it you have to have to increase **multipart_chunk_size_mb** value. We would recommend **128** or even **256**.

Raising it further is probably counter productive and may cause issues in case of network timeouts.

3.2.2. rclone multipart upload options

Rclone with default configuration options will switch multipart upload method if file you are trying to upload is over 200MiB

You can change that limit by setting **upload_cutoff** in configuration file

By default rclone will transfer multipart uploads in 5MiB requests. This limits you to about 48GiB max file size.

You can configure size of multipart requests by setting **chunk_size** in your rclone configuration to raise this limit.

Raising this limit may improve upload speed in some cases and help you upload larger files.

Example rclone config with customized multipart options

```
[s3-account]
type = s3
provider = Ceph
access_key_id = XXXXXXXX
secret_access_key = XXXXXXXX
region = eurofusion-2
endpoint = eurofusion-2.s3.storage.psnc.pl
location_constraint = eurofusion-2
acl = bucket-owner-full-control
no_check_bucket = true
upload_cutoff = 256Mi
chunk_size = 128Mi
```

(You have to configure these options per rclone remote, setting are not config wide)

3.3. Performance limitations per bucket

To maintain fair and usable service for all users buckets have ratelimit on operations per second.

3.4. Bucket object count limits

To maintain fair and usable service for all users buckets have limit of 1 000 000 buckets per bucket.

4. Preparing CLI clients

4.1. s3cmd

If you already have working .s3cfg file you can just copy it into ~/.s3cfg

If you need to configure new account

To configure s3cmd to use your s3 credential start with `s3cmd --configure`

Program will ask you to input:

Access Key: (provided by admins)

Secret Key: (provided by admins)

Default s3 region: (in this case eurofusion-2)

S3 Endpoint: (in this case just [eurofusion-2.s3.storage.psnc.pl](#))

DNS-style bucket+.....: (in this case just [eurofusion-2.s3.storage.psnc.pl](#))

Use HTTPS protocol [Yes]: (Say Yes to enable HTTPS)

Program will also ask you about Encryption password and GPG binary location.

This is for client side encryption and completely optional.

If you decide to use this encryption method and loose your encryption key your data will be unrecoverable!!!

HTTP Proxy server name: (this is optional and you don't have to configure anything in this field)

Example s3cmd configuration process.

```
s3cmd --configure
```

Enter new values or accept defaults in brackets with Enter.
Refer to user manual for detailed description of all options.

Access key and Secret key are your identifiers for Amazon S3. Leave them empty for using the env variables.

Access Key: XXXXXXXX

Secret Key: XXXXXXXX

Default Region [US]: eurofusion-2

Use "s3.amazonaws.com" for S3 Endpoint and not modify it to the target Amazon S3.

S3 Endpoint [s3.amazonaws.com]: eurofusion-2.s3.storage.psnc.pl

Use "%(bucket)s.s3.amazonaws.com" to the target Amazon S3. "%(bucket)s" and "%(location)s" vars can be used if the target S3 system supports dns based buckets.

DNS-style bucket+hostname:port template for accessing a bucket [% (bucket)s.s3.amazonaws.com]: eurofusion-2.s3.storage.psnc.pl

Encryption password is used to protect your files from reading by unauthorized persons while in transfer to S3

Encryption password:

Path to GPG program [/usr/local/bin/gpg]:

When using secure HTTPS protocol all communication with Amazon S3 servers is protected from 3rd party eavesdropping. This method is slower than plain HTTP, and can only be proxied with Python 2.7 or newer

Use HTTPS protocol [Yes]: Yes

On some networks all internet access must go through a HTTP proxy.

Try setting it here if you can't connect to S3 directly

HTTP Proxy server name:

New settings:

Access Key: XXXXXXXX

Secret Key: XXXXXXXX

Default Region: eurofusion-2

S3 Endpoint: eurofusion-2.s3.storage.psnc.pl

DNS-style bucket+hostname:port template for accessing a bucket: eurofusion-2.s3.storage.psnc.pl

Encryption password:

Path to GPG program: /opt/homebrew/bin/gpg

Use HTTPS protocol: True

HTTP Proxy server name:

HTTP Proxy server port: 0

Test access with supplied credentials? [Y/n] yes

Please wait, attempting to list all buckets...

Success. Your access key and secret key worked fine :-)

Now verifying that encryption works...

Not configured. Never mind.

Save settings? [y/N] y

Configuration saved to '/home/user/.s3cfg'

4.1.1. s3cmd with multiple accounts

If you have multiple s3 accounts you store configuration for them in multiple s3cmd config files.

For example using command `s3cmd --config ~/.s3cfg-second-account --configure` will initiate configuration but result will be saved to `~/.s3cfg-second-account`.

You can then add `--config ~/.s3cfg-second-account` to any s3cmd command to use account and configuration from that config file.

4.2. Rclone

Rclone configuration is usually stored in `~/.config/rclone/rclone.conf`

Rclone can store configuration for multiple accounts in this file.

Accounts configured in rclone are called "remote" in rclone nomenclature. (remote means it's some instance of remote storage rclone is connecting to)
In this tutorial we are naming our remote S3-account for simplicity.

Important configuration

We recommend setting `acl = bucket-owner-full-control` in rclone configuration. This prevents many legacy ACL issues when user is writing to a shared bucket.

`no_check_bucket = true` **configuration option may be required for some rclone operations to work correctly**

You can add your account to this `~/.config/rclone/rclone.conf` file using this template (Replace all fields in `<<>>` arrows)

```
[<<RCLONE ACCOUNT NAME>>]
type = s3
provider = Ceph
access_key_id = <<YOUR ACCESS KEY>>
secret_access_key = <<YOUR SECRET ACCESS KEY>>
region = eurofusion-2
endpoint = eurofusion-2.s3.storage.psnc.pl
location_constraint = eurofusion-2
acl = bucket-owner-full-control
no_check_bucket = true
```

Or you can use rclone built-in configuration manager to add an account.
Here is example of how to add account to rclone via configuration manager:

```
$ ~ rclone config
Current remotes:

Name                Type
====                ==

e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> n

Enter name for new remote.
name> s3-account

Option Storage.
Type of storage to configure.
Choose a number from below, or type in your own value.
 1 / 1Fichier
   \ (fichier)
 2 / Akamai NetStorage
   \ (netstorage)
 3 / Alias for an existing remote
   \ (alias)
 4 / Amazon S3 Compliant Storage Providers including AWS, Alibaba, ArvanCloud, BizflyCloud, Ceph, ChinaMobile, Cloudflare, Cubbit, DigitalOcean, Dreamhost, Exaba, FileLu, FlashBlade, GCS, Hetzner, HuaweiOBS, IBMCOS, IDrive, Intercolo, IONOS, Leviia, Liara, Linode, LyveCloud, Magalu, Mega, Minio, Netease, Outscale, OVHcloud, Petabox, Qiniu, Rabata, RackCorp, Rclone, Scaleway, SeaweedFS, Selectel, Servercore, SpectraLogic, StackPath, Storj, Synology, TencentCOS, Wasabi, Zata, Other
   \ (s3)
 5 / Backblaze B2
   \ (b2)
...
67 / seafile
   \ (seafile)
Storage> s3
```

```

Option provider.
Choose your S3 provider.
Choose a number from below, or type in your own value.
Press Enter to leave empty.
 1 / Amazon Web Services (AWS) S3
   \ (AWS)
...
 5 / Ceph Object Storage
   \ (Ceph)
...
47 / Any other S3 compatible provider
   \ (Other)
provider> Ceph

Option 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.
Choose a number from below, or type in your own boolean value (true or false).
Press Enter for the default (false).
 1 / Enter AWS credentials in the next step.
   \ (false)
 2 / Get AWS credentials from the environment (env vars or IAM).
   \ (true)
env_auth>

Option access_key_id.
AWS Access Key ID.
Leave blank for anonymous access or runtime credentials.
Enter a value. Press Enter to leave empty.
access_key_id> XXXXX

Option secret_access_key.
AWS Secret Access Key (password).
Leave blank for anonymous access or runtime credentials.
Enter a value. Press Enter to leave empty.
secret_access_key> XXXXX

Option region.
Region to connect to.
Leave blank if you are using an S3 clone and you don't have a region.
Choose a number from below, or type in your own value.
Press Enter to leave empty.
 / Use this if unsure.
 1 | Will use v4 signatures and an empty region.
   \ ( )
 / Use this only if v4 signatures don't work.
 2 | E.g. pre Jewel/v10 CEPH.
   \ (other-v2-signature)
region> eurofusion-2

Option endpoint.
Endpoint for S3 API.
Required when using an S3 clone.
Enter a value. Press Enter to leave empty.
endpoint> eurofusion-2.s3.storage.psn.pl

Option location_constraint.
Location constraint - must be set to match the Region.
Leave blank if not sure. Used when creating buckets only.
Enter a value. Press Enter to leave empty.
location_constraint> eurofusion-2

Option 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.
If the acl is an empty string then no X-Amz-Acl: header is added and
the default (private) will be used.

```

Choose a number from below, or type in your own value.
 Press Enter to leave empty.

```

/ Owner gets FULL_CONTROL.
1 | No one else has access rights (default).
  \ (private)
/ Owner gets FULL_CONTROL.
2 | The AllUsers group gets READ access.
  \ (public-read)
/ Owner gets FULL_CONTROL.
3 | The AllUsers group gets READ and WRITE access.
  | Granting this on a bucket is generally not recommended.
  \ (public-read-write)
/ Owner gets FULL_CONTROL.
4 | The AuthenticatedUsers group gets READ access.
  \ (authenticated-read)
/ Object owner gets FULL_CONTROL.
5 | Bucket owner gets READ access.
  | 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> bucket-owner-full-control

```

Option server_side_encryption.
 The server-side encryption algorithm used when storing this object in S3.
 Choose a number from below, or type in your own value.
 Press Enter to leave empty.

```

1 / None
  \ ()
2 / AES256
  \ (AES256)
3 / aws:kms
  \ (aws:kms)
server_side_encryption>

```

Option sse_kms_key_id.
 If using KMS ID you must provide the ARN of Key.
 Choose a number from below, or type in your own value.
 Press Enter to leave empty.

```

1 / None
  \ ()
2 / arn:aws:kms:*
  \ (arn:aws:kms:us-east-1:*)
sse_kms_key_id>

```

Edit advanced config?
 y) Yes
 n) No (default)
 y/n> n

Configuration complete.
 Options:

```

- type: s3
- provider: Ceph
- access_key_id: SNP9V2Y37XX52Q6DFZSX
- secret_access_key: S0ISLTPbDMmoJa7u3LYirUpoypEkgilC5f5gWaHE
- region: eurofusion-2
- endpoint: eurofusion-2.s3.storage.psnc.pl
- location_constraint: eurofusion-2
- acl: bucket-owner-full-control
Keep this "s3-account" remote?
y) Yes this is OK (default)
e) Edit this remote
d) Delete this remote
y/e/d> y

```

Current remotes:

Name	Type
=====	=====

```
s3-account          s3

e) Edit existing remote
n) New remote
d) Delete remote
r) Rename remote
c) Copy remote
s) Set configuration password
q) Quit config
e/n/d/r/c/s/q> q
```

4.2.1. Multiple accounts with rclone

Rclone config file `~/.config/rclone/rclone.conf` can store multiple "remotes" with different configurations.

You can create different remotes with different configurations for the same account.

You can add them via configuration manager or via text editor.

Example of how 2 remotes for the same account would look in the config file.

Notice that second remote has configured `chunk_size` value.

```
[REMOTE-1]
type = s3
provider = Ceph
access_key_id = <<YOUR ACCESS KEY>>
secret_access_key = <<YOUR SECRET ACCESS KEY>>
region = eurofusion-2
endpoint = eurofusion-2.s3.storage.psnc.pl
location_constraint = eurofusion-2
acl = bucket-owner-full-control
no_check_bucket = true

[REMOTE-2]
type = s3
provider = Ceph
access_key_id = <<YOUR ACCESS KEY>>
secret_access_key = <<YOUR SECRET ACCESS KEY>>
region = eurofusion-2
endpoint = eurofusion-2.s3.storage.psnc.pl
location_constraint = eurofusion-2
acl = bucket-owner-full-control
no_check_bucket = true
chunk_size = 128MiB
```

4.3. AWS cli

Configuring AWS cli client

```
$ ~ aws configure
AWS Access Key ID [None]: XXXXX
AWS Secret Access Key [None]: XXXXX
Default region name [None]: eurofusion-2
Default output format [None]:
```

Warning:

Setting `endpoint_url` in `~/.aws/config` has no effect. (AWS is annoying and doesn't support people using their tools with non AWS services)

While using `aws` command line program you have to always include `--endpoint https://eurofusion-2.s3.storage.psnc.pl` as an argument.

For example:

```
aws s3 --endpoint https://eurofusion-2.s3.storage.psnc.pl ls
```

4.3.1. Multiple accounts with AWS cli

AWS cli can store multiple accounts in it's configuration files.

You can configure additional accounts by using `--profile <<PROFILE_NAME>>` switch

For example:

```
# configure second profile
aws configure --profile second-profile
# list buckets using second profile
aws s3 --profile second-profile --endpoint https://eurofusion-2.s3.storage.psnc.pl ls
```

5. Basic operations (data upload and download)

In this section we will use default s3cmd and AWS cli configuration files.

Rclone will use remote s3-account we created in section 3.2.

In this test I'm working as `tutorial-bucket` user which has only access to it's own bucket `tutorial-bucket`

5.1. Creating example directory structure used in this tutorial

You can create example directory and file structure I will be using in the tutorial with this commands.

You just have to execute these 2 commands in your shell. (tested on bash-5.3)

```
mkdir -p dir1/dir12 dir1/dir13/dir131
touch touch ./dir1/file{0..9} ./dir1/dir13/dir131/file-{0..9}
```

It will create this structure

```
tree dir1
dir1
  dir12
  dir13
    dir131
      file-0
      file-1
      file-2
      file-3
      file-4
      file-5
      file-6
      file-7
      file-8
      file-9
  file0
  file1
  file2
  file3
  file4
  file5
  file6
  file7
  file8
  file9
```

5.2. Example: Listing available buckets

Warning: Just because you can see a bucket doesn't mean you have access to it's content.

5.2.1. s3cmd

```
s3cmd ls
2026-03-23 11:45 s3://tutorial-bucket
2026-03-20 12:08 s3://g2bpalak
2026-03-20 12:03 s3://g2bpogo
2026-03-20 10:10 s3://g2dfigat
2026-03-20 12:09 s3://g2jwasik
2026-03-20 12:10 s3://g2knizn
2026-03-20 12:01 s3://g2michal
```

5.3. Example: Listing bucket content

5.3.1. s3cmd

Listing objects in s3 bucket

```
s3cmd ls s3://tutorial-bucket
DIR s3://tutorial-bucket/pseudo-directory/
2026-03-23 13:36 154 s3://tutorial-bucket/s3-tutorial.sh
```

Listing objects in pseudo-directory

```
s3cmd ls s3://tutorial-bucket/pseudo-directory/
2026-03-23 13:36 154 s3://tutorial-bucket/pseudo-directory/s3-tutorial.sh
```

Recursive object listing in s3 bucket

```
s3cmd ls -r s3://tutorial-bucket
2026-03-23 13:36 154 s3://tutorial-bucket/pseudo-directory/s3-tutorial.sh
2026-03-23 13:36 154 s3://tutorial-bucket/s3-tutorial.sh
```

5.4. Example: Uploading file to bucket

5.4.1. s3cmd

Uploading single file to s3 bucket

```
s3cmd put s3-tutorial.sh s3://tutorial-bucket/
upload: 's3-tutorial.sh' -> 's3://tutorial-bucket/s3-tutorial.sh' [1 of 1]
154 of 154 100% in 0s 2.14 KB/s done
```

Uploading file to pseudo-directory inside a bucket and changing the filename

```
s3cmd put s3-tutorial.sh s3://tutorial-bucket/s3cmd/changed-name.sh
upload: 's3-tutorial.sh' -> 's3://tutorial-bucket/s3cmd/changed-name.sh' [1 of 1]
154 of 154 100% in 0s 2.26 KB/s done
```

5.5. Example: Uploading directory structure with data

In this section we will upload tree of multiple directories to S3 bucket

NOTICE empty dir1/dir12 will not be preserved since s3 doesn't have directory objects.

If you want to keep dir1/dir12 represented on s3 bucket, you will have to create empty file in it. (Just like in git repository)

Our example data structure:

```
dir1/  
  dir12  
  dir13  
    dir131  
      file-0  
      file-1  
      file-2  
      file-3  
      file-4  
      file-5  
      file-6  
      file-7  
      file-8  
      file-9  
  file-0  
  file-1  
  file-2  
  file-3  
  file-4  
  file-5  
  file-6  
  file-7  
  file-8  
  file-9
```

5.5.1. s3cmd

Uploading directory structure to s3 bucket

```
s3cmd sync ./dir1/ s3://tutorial-bucket/
upload: './dir1/dir13/dir131/file-0' -> 's3://tutorial-bucket/dir13/dir131/file-0' [1 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-1' -> 's3://tutorial-bucket/dir13/dir131/file-1' [2 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-2' -> 's3://tutorial-bucket/dir13/dir131/file-2' [3 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-3' -> 's3://tutorial-bucket/dir13/dir131/file-3' [4 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-4' -> 's3://tutorial-bucket/dir13/dir131/file-4' [5 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-5' -> 's3://tutorial-bucket/dir13/dir131/file-5' [6 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-6' -> 's3://tutorial-bucket/dir13/dir131/file-6' [7 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-7' -> 's3://tutorial-bucket/dir13/dir131/file-7' [8 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-8' -> 's3://tutorial-bucket/dir13/dir131/file-8' [9 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/dir13/dir131/file-9' -> 's3://tutorial-bucket/dir13/dir131/file-9' [10 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-0' -> 's3://tutorial-bucket/file-0' [11 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-1' -> 's3://tutorial-bucket/file-1' [12 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-2' -> 's3://tutorial-bucket/file-2' [13 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-3' -> 's3://tutorial-bucket/file-3' [14 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-4' -> 's3://tutorial-bucket/file-4' [15 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-5' -> 's3://tutorial-bucket/file-5' [16 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-6' -> 's3://tutorial-bucket/file-6' [17 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-7' -> 's3://tutorial-bucket/file-7' [18 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-8' -> 's3://tutorial-bucket/file-8' [19 of 20]
0 of 0 0% in 0s 0.00 B/s done
upload: './dir1/file-9' -> 's3://tutorial-bucket/file-9' [20 of 20]
0 of 0 0% in 0s 0.00 B/s done
```

Listing our s3 bucket to check structure of uploaded data

```
s3cmd ls -r s3://tutorial-bucket
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-0
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-1
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-2
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-3
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-4
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-5
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-6
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-7
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-8
2026-03-23 13:59 0 s3://tutorial-bucket/dir13/dir131/file-9
2026-03-23 13:59 0 s3://tutorial-bucket/file-0
2026-03-23 13:59 0 s3://tutorial-bucket/file-1
2026-03-23 13:59 0 s3://tutorial-bucket/file-2
2026-03-23 13:59 0 s3://tutorial-bucket/file-3
2026-03-23 13:59 0 s3://tutorial-bucket/file-4
2026-03-23 13:59 0 s3://tutorial-bucket/file-5
2026-03-23 13:59 0 s3://tutorial-bucket/file-6
2026-03-23 13:59 0 s3://tutorial-bucket/file-7
2026-03-23 13:59 0 s3://tutorial-bucket/file-8
2026-03-23 13:59 0 s3://tutorial-bucket/file-9
```

5.6. Example: Downloading data from bucket

5.6.1. s3cmd

S3cmd sync command works more or less like unix cp shell command.

Here it copies pseudo-r dir1 from bucket tutorial-bucket into local directory ./dir2

```
s3cmd sync s3://tutorial-bucket/dir1 ./dir2
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-0' -> './dir2/dir1/dir13/dir131/file-0' [1 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-1' -> './dir2/dir1/dir13/dir131/file-1' [2 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-2' -> './dir2/dir1/dir13/dir131/file-2' [3 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-3' -> './dir2/dir1/dir13/dir131/file-3' [4 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-4' -> './dir2/dir1/dir13/dir131/file-4' [5 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-5' -> './dir2/dir1/dir13/dir131/file-5' [6 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-6' -> './dir2/dir1/dir13/dir131/file-6' [7 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-7' -> './dir2/dir1/dir13/dir131/file-7' [8 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-8' -> './dir2/dir1/dir13/dir131/file-8' [9 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/dir13/dir131/file-9' -> './dir2/dir1/dir13/dir131/file-9' [10 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-0' -> './dir2/dir1/file-0' [11 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-1' -> './dir2/dir1/file-1' [12 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-2' -> './dir2/dir1/file-2' [13 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-3' -> './dir2/dir1/file-3' [14 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-4' -> './dir2/dir1/file-4' [15 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-5' -> './dir2/dir1/file-5' [16 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-6' -> './dir2/dir1/file-6' [17 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-7' -> './dir2/dir1/file-7' [18 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-8' -> './dir2/dir1/file-8' [19 of 20]
0 of 0 0% in 0s 0.00 B/s done
download: 's3://tutorial-bucket/dir1/file-9' -> './dir2/dir1/file-9' [20 of 20]
0 of 0 0% in 0s 0.00 B/s done
```

Checking file structure of downloaded data

```
tree dir2/
dir2/
  dir1
    dir13
      dir131
        file-0
        file-1
        file-2
        file-3
        file-4
        file-5
        file-6
        file-7
        file-8
        file-9
    file-0
    file-1
    file-2
    file-3
    file-4
    file-5
    file-6
    file-7
    file-8
    file-9
```

5.7. Example: Copying data inside s3

In most cases copying data between prefixes or buckets in the same s3 instance happens in server-side mode.

This means that for simplicity objects are not sent to the client and back to the server, but copying takes place on the server and client is just notified about progress.

5.7.1. s3cmd

In case of s3cmd cp command keep in mind that you have add trailing / characters to indicate that you are copying data between prefixes/pseudo directories.

As we can see in this example, copy was performed server-side without sending data to the user

```
s3cmd cp --recursive s3://tutorial-bucket/dir1/ s3://tutorial-bucket/dir2/
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-0' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-0' [1 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-1' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-1' [2 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-2' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-2' [3 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-3' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-3' [4 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-4' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-4' [5 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-5' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-5' [6 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-6' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-6' [7 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-7' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-7' [8 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-8' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-8' [9 of 20]
remote copy: 's3://tutorial-bucket/dir1/dir13/dir131/file-9' -> 's3://tutorial-bucket/dir2/dir13/dir131/file-9' [10 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-0' -> 's3://tutorial-bucket/dir2/file-0' [11 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-1' -> 's3://tutorial-bucket/dir2/file-1' [12 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-2' -> 's3://tutorial-bucket/dir2/file-2' [13 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-3' -> 's3://tutorial-bucket/dir2/file-3' [14 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-4' -> 's3://tutorial-bucket/dir2/file-4' [15 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-5' -> 's3://tutorial-bucket/dir2/file-5' [16 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-6' -> 's3://tutorial-bucket/dir2/file-6' [17 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-7' -> 's3://tutorial-bucket/dir2/file-7' [18 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-8' -> 's3://tutorial-bucket/dir2/file-8' [19 of 20]
remote copy: 's3://tutorial-bucket/dir1/file-9' -> 's3://tutorial-bucket/dir2/file-9' [20 of 20]
```

Result

```
s3cmd ls -r s3://tutorial-bucket/dir2/
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-0
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-1
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-2
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-3
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-4
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-5
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-6
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-7
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-8
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/dir13/dir131/file-9
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-0
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-1
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-2
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-3
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-4
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-5
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-6
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-7
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-8
2026-03-26 12:24      0 s3://tutorial-bucket/dir2/file-9
```

5.8. Example: Measuring data usage on a buckets

5.8.1. s3cmd

**CHECKING BUCKET SIZE REQUIRES CLIENT TO LIST AND CHECK METADATA OF EVERY OBJECT
IT IS TIME CONSUMING AND RESOURCE INTENSIVE FOR LARGE BUCKETS**

Listing test bucket

this bucket has 40 files with 0 byte content (all files are empty)

```
s3cmd du -H s3://tutorial-bucket
0      40 objects s3://tutorial-bucket/
```

After uploading some more data bucket size changes.

```
s3cmd du -H s3://tutorial-bucket
128M   41 objects s3://tutorial-bucket/
```

5.9. Example: Deleting data

```
s3cmd ls s3://tutorial-bucket/examples/delete/
DIR     s3://tutorial-bucket/examples/delete/aws-cli/
DIR     s3://tutorial-bucket/examples/delete/rc1one/
DIR     s3://tutorial-bucket/examples/delete/s3cmd/
```

5.9.1. s3cmd

```
s3cmd rm -r s3://tutorial-bucket/examples/delete/s3cmd
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-0'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-1'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-2'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-3'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-4'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-5'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-6'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-7'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-8'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/dir13/dir131/file-9'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-0'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-1'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-2'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-3'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-4'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-5'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-6'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-7'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-8'
delete: 's3://tutorial-bucket/examples/delete/s3cmd/dir1/file-9'
```

Result pseudo-directory was deleted

```
s3cmd ls s3://tutorial-bucket/examples/delete/
DIR     s3://tutorial-bucket/examples/delete/aws-cli/
DIR     s3://tutorial-bucket/examples/delete/rc1one/
```

5.10. Operations on large files (Multipart upload)

S3 API is HTTP based and is limited by HTTP request sizes. To overcome these limitations and improve performance S3 implements multipart uploads. Multipart uploads allow you to upload files larger than 5120MiB by uploading them in parts. Multipart upload can have a maximum of 10000 parts and each part size can be up to 5120MiB in size.

Uploading objects this big is strongly discouraged

5.10.1. Configuring multipart upload chunk size

Default multipart size differs between S3 clients. In most cases you don't need to worry about changing it.

In case you need to change it:

5.10.1.1. s3cmd

You configure it in command line via `--multipart-chunk-size-mb=128`

5.10.1.2. rclone

You can configure multipart upload size via command line argument `--s3-chunk-size` or in configuration file with `chunk_size = 128MiB` [Link to detailed rclone documentation](#)

Rclone can also upload multiple multipart chunks in parallel (for example chunks 1, 2, 3, 4 can be uploaded at the same time) [Link to detailed rclone documentation](#)

This can be configured with `--s3-upload-concurrency`. This option can sometimes be used to speedup upload speed, but doesn't have the same performance impact as raising chunk size.

5.10.1.3. AWS cli

You can configure multipart upload size via in `~/.aws/config` using variable `multipart_chunksize` [Link to detailed aws cli documentation](#)

5.10.2. Multipart upload example

Let's quickly create a 300MiB test file

```
dd if=/dev/urandom of=./big_file.bin bs=1M count=300 status=progress
```

5.10.2.1. s3cmd

We upload file using normal put command

```
s3cmd -c ~/.s3cfg put big_file.bin s3://tutorial-bucket/size/
```

In command output we see that file was transferred in 20 15MiB requests

```

upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 1 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 40.48 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 2 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 53.85 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 3 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 52.84 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 4 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 53.65 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 5 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 54.46 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 6 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 57.71 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 7 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 58.50 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 8 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 56.70 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 9 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 58.45 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 10 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 58.43 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 11 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 51.30 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 12 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 53.78 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 13 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 54.95 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 14 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 57.19 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 15 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 59.36 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 16 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 54.16 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 17 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 57.89 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 18 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 55.68 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 19 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 58.07 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 20 of 20, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 55.89 MB/s done

```

We can configure size of the chunks in which file is uploaded to s3.
We can quickly overwrite the default size with this command.

```
echo 'multipart_chunk_size_mb = 128' >> ~/.s3cfg
```

We can retry uploading the same file

```
s3cmd put big_file.bin s3://tutorial-bucket/size/
```

Result

```

upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 1 of 3, 128MB] [1 of 1]
134217728 of 134217728 100% in 2s 44.74 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 2 of 3, 128MB] [1 of 1]
134217728 of 134217728 100% in 2s 60.67 MB/s done
upload: 'big_file.bin' -> 's3://tutorial-bucket/size/big_file.bin' [part 3 of 3, 44MB] [1 of 1]
46137344 of 46137344 100% in 0s 61.31 MB/s done

```

Our file was uploaded in 3 chunks. 2 were 128MiB long. Tailing chunk is of course smaller.

5.10.2.2. rclone

By default rclone won't show you in terminal output how big it's configured multipart upload chunk size is
If you add -vv debug option to rclone arguments you can see chunk sizes used during upload.

Example from rclone debug

```
2026/04/13 11:55:25 DEBUG : big_file.bin: multipart upload: starting chunk 51 size 5Mi offset 255Mi/300Mi
```

Uploading our 300MiB file with default 5MiB chunk setting took me 19 seconds and achieved transfer transfer speed of 15,4MiB/s

Changing multipart chunk size to 100MiB and uploading the same file new settings took only 6,7 seconds and achieved transfer transfer speed of 44MiB/s

5.10.3. Listing and cleaning up unfinished multipart uploads with s3cmd

In case you interrupt multipart upload or it fails to finish for some reason your partially uploaded object will remain in your s3 bucket

Unfinished multipart uploads count towards your Quota usage

Example: interrupted multipart upload

```
s3cmd put files/128MiB s3://tutorial-bucket/size/4
upload: 'files/128MiB' -> 's3://tutorial-bucket/size/4' [part 1 of 9, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 19.70 MB/s done
upload: 'files/128MiB' -> 's3://tutorial-bucket/size/4' [part 2 of 9, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 21.28 MB/s done
upload: 'files/128MiB' -> 's3://tutorial-bucket/size/4' [part 3 of 9, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 17.46 MB/s done
upload: 'files/128MiB' -> 's3://tutorial-bucket/size/4' [part 4 of 9, 15MB] [1 of 1]
15728640 of 15728640 100% in 0s 22.87 MB/s done
upload: 'files/128MiB' -> 's3://tutorial-bucket/size/4' [part 5 of 9, 15MB] [1 of 1]
65536 of 15728640 0% in 0s 3.99 MB/s^CERROR:
Upload of 'files/128MiB' part 5 failed. Use
/usr/bin/s3cmd abortmp s3://tutorial-bucket/size/4 2~nXJaeaNSGin_TbIPnghIWYShQFVFJHa
to abort the upload, or
/usr/bin/s3cmd --upload-id 2~nXJaeaNSGin_TbIPnghIWYShQFVFJHa put ...
to continue the upload.
See ya!
```

s3cmd gives you 2 options, to resume upload or abort it.

Let's list our unfinished multipart uploads

```
s3cmd listmp s3://tutorial-bucket/size/4 '2~nXJaeaNSGin_TbIPnghIWYShQFVFJHa'
LastModified          PartNumber      ETag             Size
2026-03-26T12:57:08.568Z    1              "9e6ead96034cf8dbeaea0f36185d1eea"    15728640
2026-03-26T12:57:09.285Z    2              "7e28cc663c25d9fab279e91e7bbc9abb"    15728640
2026-03-26T12:57:10.149Z    3              "3a8be8bd692cc6265fad05820b3165e7"    15728640
2026-03-26T12:57:10.806Z    4              "d43a0f706556332ea5d60e88d7a60d55"    15728640
```

We can delete this unfinished multipart upload with this command

```
s3cmd abortmp s3://tutorial-bucket/size/4 '2~nXJaeaNSGin_TbIPnghIWYShQFVFJHa'
s3://tutorial-bucket/size/4
```

Our multipart is deleted

```
s3cmd listmp s3://tutorial-bucket/size/4 '2~nXJaeaNSGin_TbIPnghIWYShQFVFJHa'  
ERROR: S3 error: 404 (NoSuchKey)
```

5.10.4. Dealing with unfinished multipart uploads using rclone

Rclone can list all unfinished multipart uploads on our bucket

```
rclone backend list-multipart-uploads s3-account:tutorial-bucket  
{  
  "tutorial-bucket": [  
    {  
      "ChecksumAlgorithm": "",  
      "ChecksumType": "",  
      "Initiated": "2026-03-26T13:03:24.132Z",  
      "Initiator": {  
        "DisplayName": "eurofusion-2",  
        "ID": "RGW88505570706778033"  
      },  
      "Key": "size/4",  
      "Owner": {  
        "DisplayName": "eurofusion-2",  
        "ID": "RGW88505570706778033"  
      },  
      "StorageClass": "STANDARD",  
      "UploadId": "2~43kQsy_Ky3UZs2cNDe6MTdFa0a90Wpm"  
    },  
  ],  
}
```

`rclone cleanup s3-account:bucketname` will remove all unfinished s3 uploads older than 24h

```
rclone cleanup -P s3-account:tutorial-bucket
```

You can also manually delete unfinished multipart uploads immediately

```
rclone backend cleanup s3-account:tutorial-bucket -o max-age=0h
```

Now there are no unfinished uploads in our bucket

```
rclone backend list-multipart-uploads s3-account:tutorial-bucket  
{  
  "tutorial-bucket": []  
}
```

5.11. Example: backing up \${HOME} to s3

In this example we will backup `${HOME}` directory to s3 bucket

We will create a fake `${HOME}` directory for this test. We wouldn't want to backup real data in this example.

Creating fake `${HOME}` data structure

```
mkdir -p ./fake_home/dir{1,2,3,4}
touch ./fake_home/dir1/file-{1..6} ./fake_home/dir2/file-2{0..4} ./fake_home/dir3/.keep_dir
#copy s3cmd into our fake_home
cp ~/.s3cfg ./fake_home/
#backup original ${HOME} variable
BACKUP_HOME="${HOME}"
#changen ${HOME} variable
export HOME=${PWD}/fake_homecd ${HOME}
```

It should generate this data structure in fake home

```
$ ~ -> tree
.
├── dir1
│   ├── file-1
│   ├── file-2
│   ├── file-3
│   ├── file-4
│   ├── file-5
│   └── file-6
├── dir2
│   ├── file-20
│   ├── file-21
│   ├── file-22
│   ├── file-23
│   └── file-24
├── dir3
└── dir4
```

You can later reset value of your `${HOME}` variable by: logging in again via ssh, restarting terminal windows or using backup variable

```
export HOME=${BACKUP_HOME}
```

5.11.1. s3cmd

We will use `s3cmd sync --no-delete-removed` to backup `${HOME}` directory.

```

$ ~ -> s3cmd sync --no-delete-removed ${HOME}/ s3://tutorial-bucket/backup/
upload: '/Users/ev/tmp/s3-tutorial/fake_home/.s3cfg' -> 's3://tutorial-bucket/backup/.s3cfg' [1 of 13]
 2431 of 2431 100% in 0s 144.76 KB/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir1/file-1' -> 's3://tutorial-bucket/backup/dir1/file-1' [2 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir1/file-2' -> 's3://tutorial-bucket/backup/dir1/file-2' [3 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir1/file-3' -> 's3://tutorial-bucket/backup/dir1/file-3' [4 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir1/file-4' -> 's3://tutorial-bucket/backup/dir1/file-4' [5 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir1/file-5' -> 's3://tutorial-bucket/backup/dir1/file-5' [6 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir1/file-6' -> 's3://tutorial-bucket/backup/dir1/file-6' [7 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir2/file-20' -> 's3://tutorial-bucket/backup/dir2/file-20' [8 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir2/file-21' -> 's3://tutorial-bucket/backup/dir2/file-21' [9 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir2/file-22' -> 's3://tutorial-bucket/backup/dir2/file-22' [10 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir2/file-23' -> 's3://tutorial-bucket/backup/dir2/file-23' [11 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir2/file-24' -> 's3://tutorial-bucket/backup/dir2/file-24' [12 of 13]
 0 of 0 0% in 0s 0.00 B/s done
upload: '/Users/ev/tmp/s3-tutorial/fake_home/dir3/.keep_dir' -> 's3://tutorial-bucket/backup/dir3/.keep_dir' [13 of 13]
 0 of 0 0% in 0s 0.00 B/s done
Done. Uploaded 2431 bytes in 1.0 seconds, 2.37 KB/s.

```

Resulting data on s3 bucket

```

$ ~ -> s3cmd ls -r s3://tutorial-bucket/backup/
2026-04-13 09:29      2431 s3://tutorial-bucket/backup/.s3cfg
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-1
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-2
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-3
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-4
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-5
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-6
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-20
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-21
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-22
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-23
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-24
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir3/.keep_dir

```

We can delete some of our data in fake home directory

```
$ ~ -> rm -rf dir2/
$ ~ -> tree
.
├── dir1
│   ├── file-1
│   ├── file-2
│   ├── file-3
│   ├── file-4
│   ├── file-5
│   └── file-6
├── dir3
└── dir4

4 directories, 6 files
```

We can run our backup command again

```
$ ~ -> s3cmd sync --no-delete-removed ${HOME}/ s3://tutorial-bucket/backup/
$ ~ ->
```

s3cmd didn't delete any of our data from backup

```
$ ~ -> s3cmd ls -r s3://tutorial-bucket/backup/
2026-04-13 09:29      2431 s3://tutorial-bucket/backup/.s3cfg
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-1
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-2
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-3
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-4
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-5
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-6
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-20
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-21
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-22
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-23
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir2/file-24
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir3/.keep_dir
```

We can change the flag to allow of deletion of backed up files

```
$ ~ -> s3cmd sync --delete-removed ${HOME}/ s3://tutorial-bucket/backup/
delete: 's3://tutorial-bucket/backup/dir2/file-20'
delete: 's3://tutorial-bucket/backup/dir2/file-21'
delete: 's3://tutorial-bucket/backup/dir2/file-22'
delete: 's3://tutorial-bucket/backup/dir2/file-23'
delete: 's3://tutorial-bucket/backup/dir2/file-24'
```

We can check our bucket content again

```
$ ~ -> s3cmd ls -r s3://tutorial-bucket/backup/
2026-04-13 09:29      2431 s3://tutorial-bucket/backup/.s3cfg
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-1
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-2
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-3
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-4
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-5
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir1/file-6
2026-04-13 09:29          0 s3://tutorial-bucket/backup/dir3/.keep_dir
```

5.11.2. rclone

We will use `rclone copy` command because it copies data to a bucket without deleting objects which are no longer in the `${HOME}` directory.

Our `${HOME}` directory looks like this

```
tree $HOME
/home/s3-tutorial/dir1
  dir12
  dir13
    dir131
      file-0
      file-1
      file-2
      file-3
      file-4
      file-5
      file-6
      file-7
      file-8
      file-9
  file-0
  file-1
  file-2
  file-3
  file-4
  file-5
  file-6
  file-7
  file-8
  file-9
```

We backup our data to s3 bucket

```
rcclone copy -P ${HOME}/ s3-account:tutorial-bucket/home-backup/
Transferred:          0 B / 0 B, -, 0 B/s, ETA -
Checks:              0 / 0, -, Listed 23
Transferred:         20 / 20, 100%
Elapsed time:         0.2s
```

We can now check how data structure looks on s3 bucket

```
rcclone ls s3-account:tutorial-bucket/home-backup/
0 dir13/dir131/file-0
0 dir13/dir131/file-1
0 dir13/dir131/file-2
0 dir13/dir131/file-3
0 dir13/dir131/file-4
0 dir13/dir131/file-5
0 dir13/dir131/file-6
0 dir13/dir131/file-7
0 dir13/dir131/file-8
0 dir13/dir131/file-9
0 file-0
0 file-1
0 file-2
0 file-3
0 file-4
0 file-5
0 file-6
0 file-7
0 file-8
0 file-9
```

We can also test how ``rcclone copy`` will behave once some `${HOME}` directory is deleted.

Deleting some files

```
rm -rf ${HOME}/dir13
```

Running rclone check again

```
rclone-mac copy -P ${HOME}/ s3-account:tutorial-bucket/home-backup/  
Transferred:          0 B / 0 B, -, 0 B/s, ETA -  
Checks:              10 / 10, 100%, Listed 22  
Elapsed time:        0.1s
```

We can see that no data was deleted on s3 bucket

```
rclone-mac ls s3-account:tutorial-bucket/home-backup/  
0 dir13/dir131/file-0  
0 dir13/dir131/file-1  
0 dir13/dir131/file-2  
0 dir13/dir131/file-3  
0 dir13/dir131/file-4  
0 dir13/dir131/file-5  
0 dir13/dir131/file-6  
0 dir13/dir131/file-7  
0 dir13/dir131/file-8  
0 dir13/dir131/file-9  
0 file-0  
0 file-1  
0 file-2  
0 file-3  
0 file-4  
0 file-5  
0 file-6  
0 file-7  
0 file-8  
0 file-9
```

Testing what happens when file gets modified

Overwriting file content

```
dd if=/dev/urandom of=${HOME}/file-0 bs=1M count=1
```

Running rclone copy again

```
rclone-mac copy -P ${HOME}/ s3-account:tutorial-bucket/home-backup/  
Transferred:          1 MiB / 1 MiB, 100%, 0 B/s, ETA -  
Checks:              10 / 10, 100%, Listed 22  
Transferred:          1 / 1, 100%  
Elapsed time:        0.4s
```

Only changed file was backed up to s3

Listing new bucket content

```

rclone-mac ls s3-account:tutorial-bucket/home-backup/
0 dir13/dir131/file-0
0 dir13/dir131/file-1
0 dir13/dir131/file-2
0 dir13/dir131/file-3
0 dir13/dir131/file-4
0 dir13/dir131/file-5
0 dir13/dir131/file-6
0 dir13/dir131/file-7
0 dir13/dir131/file-8
0 dir13/dir131/file-9
1048576 file-0
0 file-1
0 file-2
0 file-3
0 file-4
0 file-5
0 file-6
0 file-7
0 file-8
0 file-9

```

As we can see, file-0 size changed because it was overwritten.

6. Bucket and Object permissions in S3

6.1. Bucket Policies

In case you need to give some user access to your data you do it using bucket policies and applying it to your bucket.

In this tutorial we will show you few useful bucket policy examples.

Subject of bucket policies is very well explained on official [AWS website](#).

6.1.1. Bucket policy compatibility

Keep in mind that in case of this service, we are using different implementation of s3 API and not every advanced AWS s3 bucket policy option will be available. You can find list of supported bucket policy action in [official ceph documentation](#).

6.1.2. How users are identified in bucket policies

In bucket policies users are identified by ARM string.

It should be provided to you with your s3 keys by administrators

In most cases it will look like this `arn:aws:iam::RGW88505570706778033:user/user-1`

Last part of the ARN string usually show username

Middle part RGWXXXXXXXXXXXXXXXXX is not guaranteed to be the same for different users

If you want to give some user access to your data, you need to get their ARM string and user it in bucket policy.

6.2. Example: Checking bucket policy status

6.2.1. s3cmd

In this case our bucket has no bucket policy configured

```
s3cmd info s3://tutorial-bucket
s3://tutorial-bucket/ (bucket):
  Location:    eurofusion-2
  Payer:       BucketOwner
  Ownership:   none
  Versioning:  none
  Expiration rule: none
  Block Public Access: none
  Policy:      none
  CORS:        none
  ACL:         eurofusion-2: FULL_CONTROL
```

6.2.2. AWS cli

Checking state of bucket-policy

```
aws --endpoint https://eurofusion-2.s3.storage.psnc.pl s3api get-bucket-policy-status --bucket tutorial-bucket
{
  "PolicyStatus": {
    "IsPublic": false
  }
}
```

Checking bucket policy (in this case it's empty)

```
aws --endpoint https://eurofusion-2.s3.storage.psnc.pl s3api get-bucket-policy --bucket tutorial-bucket

aws: [ERROR]: An error occurred (NoSuchBucketPolicy) when calling the GetBucketPolicy operation: The bucket
policy does not exist

Additional error details:
BucketName: tutorial-bucket
```

6.3. Example: Setting bucket policy

In this section we will configure bucket-policy

Bucket policy file content:

```
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/user-1"]},
    "Action": "*",
    "Resource": [
      "arn:aws:s3:::bucket-user-1",
      "arn:aws:s3:::bucket-user-1/*"
    ]
  }]
}
```

6.3.1. s3cmd

```
s3cmd setpolicy ./bucket-policy.json s3://tutorial-bucket
s3://tutorial-bucket/: Policy updated
```

Result

```
s3cmd info s3://tutorial-bucket
s3://tutorial-bucket/ (bucket):
  Location:   eurofusion-2
  Payer:      BucketOwner
  Ownership:  none
  Versioning: none
  Expiration rule: none
  Block Public Access: none
  Policy:     {
"Version": "2012-10-17",
"Statement": [{
  "Effect": "Allow",
  "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/user-1"]},
  "Action": "*",
  "Resource": [
    "arn:aws:s3:::bucket-user-1",
    "arn:aws:s3:::bucket-user-1/*"
  ]
}]
}

CORS:      none
ACL:       eurofusion-2: FULL_CONTROL
```

6.3.2. AWS cli

Command exits with no success message

```
aws --endpoint https://eurofusion-2.s3.storage.psnc.pl s3api put-bucket-policy --bucket tutorial-bucket --
policy file://./bucket-policy.json
```

Result

```
aws --endpoint https://eurofusion-2.s3.storage.psnc.pl s3api get-bucket-policy --bucket tutorial-bucket --
output text
{
  "Version": "2012-10-17",
  "Statement": [{
    "Effect": "Allow",
    "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/user-1"]},
    "Action": "*",
    "Resource": [
      "arn:aws:s3:::bucket-user-1",
      "arn:aws:s3:::bucket-user-1/*"
    ]
  }]
}
```

6.4. Example: Deleting bucket-policy

6.4.1. s3cmd

```
s3cmd delpolicy s3://tutorial-bucket
s3://tutorial-bucket/: Policy deleted
```

Result (bucket policy is gone)

```
s3://tutorial-bucket/ (bucket):
Location:  eurofusion-2
Payer:     BucketOwner
Ownership: none
Versioning: none
Expiration rule: none
Block Public Access: none
Policy:    none
CORS:      none
ACL:       eurofusion-2: FULL_CONTROL
```

6.4.2. AWS cli

```
aws --endpoint https://eurofusion-2.s3.storage.psnc.pl s3api delete-bucket-policy --bucket tutorial-bucket
```

Result (bucket-policy was deleted)

```
aws --endpoint https://eurofusion-2.s3.storage.psnc.pl s3api get-bucket-policy --bucket tutorial-bucket --
output text

aws: [ERROR]: An error occurred (NoSuchBucketPolicy) when calling the GetBucketPolicy operation: The bucket
policy does not exist

Additional error details:
BucketName: tutorial-bucket
```

6.5. Example: Giving other user Read Only access to your bucket

In this example we give ARN **arn:aws:iam::RGW88505570706778033:user/ltids-s3-training-2** read only and list access to the whole bucket **ltids-s3-training-01**

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListSpecificPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/ltids-s3-training-02"]},
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::ltids-s3-training-01"
    },
    {
      "Sid": "AllowGetObjectsFromSpecificPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/ltids-s3-training-02"]},
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3:::ltids-s3-training-01/*"
    }
  ]
}
```

Testing result

User **ltids-training-02** can list all files in bucket **s3://ltids-training-01**

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg ls -r s3://ltds-s3-training-01
2026-04-13 07:42      1048576  s3://ltds-s3-training-01/prefix-1/1MiB
```

6.6. Example: Giving other user Read Only access to part of your bucket

In this example we give ARN `arn:aws:iam::RGW88505570706778033:user/ltds-s3-training-2` read only and list access to prefix `prefix-1` from bucket `ltds-s3-training-01`

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListSpecificPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/ltds-s3-training-02"]},
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3::ltds-s3-training-01",
      "Condition": {
        "StringLike": {
          "s3:prefix": [
            "prefix-1/*",
            "prefix-1"
          ]
        }
      }
    },
    {
      "Sid": "AllowGetObjectsFromSpecificPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/ltds-s3-training-02"]},
      "Action": [
        "s3:GetObject"
      ],
      "Resource": "arn:aws:s3::ltds-s3-training-01/prefix-1/*"
    }
  ]
}
```

Testing permissions

user `ltds-s3-training-02` cannot list the whole bucket

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg ls -r s3://ltds-s3-training-01
ERROR: Access to bucket 'ltds-s3-training-01' was denied
ERROR: S3 error: 403 (AccessDenied)
```

User `ltds-s3-training-02` can list only objects in specified `prefix-1`

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg ls -r s3://ltds-s3-training-01/prefix-1
2026-04-13 07:42      1048576  s3://ltds-s3-training-01/prefix-1/1MiB
```

6.7. Example: Giving other user Write Only access to part of your bucket

In this example we give ARN `arn:aws:iam::RGW88505570706778033:user/ltds-s3-training-02` write to prefix `prefix-1` from bucket `ltds-s3-training-01`

This bucket-policy allows ARN to List and Uploads objects but not delete or download them.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListOnlyPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/user-1"]},
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3::your-bucket-name",
      "Condition": {
        "StringLike": { "s3:prefix": [ "prefix-1/*", "prefix-1" ] }
      }
    },
    {
      "Sid": "AllowPutObjectsInPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/user-1"]},
      "Action": [
        "s3:PutObject"
      ],
      "Resource": "arn:aws:s3::your-bucket-name/prefix-1/*"
    }
  ]
}
```

Testing policy

User **ltds-s3-training-02** cannot list whole bucket

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg ls -r s3://ltds-s3-training-01
ERROR: Access to bucket 'ltds-s3-training-01' was denied
ERROR: S3 error: 403 (AccessDenied)
```

User **ltds-s3-training-02** can list prefix-1

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg ls -r s3://ltds-s3-training-01/prefix-1
2026-04-13 07:42      1048576  s3://ltds-s3-training-01/prefix-1/1MiB
```

User **ltds-s3-training-02** can upload files only to **prefix-1**

Upload to **prefix-2** is denied

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg put example_files/1MiB s3://ltds-s3-training-01/prefix-2/file-2.1MiB
upload: 'example_files/1MiB' -> 's3://ltds-s3-training-01/prefix-2/file-2.1MiB' [1 of 1]
1048576 of 1048576 100% in 0s 2.29 MB/s done
ERROR: S3 error: 403 (AccessDenied)
```

Upload to **prefix-1** is allowed

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg put example_files/1MiB s3://ltds-s3-training-01/prefix-1/file-2.1MiB
upload: 'example_files/1MiB' -> 's3://ltds-s3-training-01/prefix-1/file-2.1MiB' [1 of 1]
1048576 of 1048576 100% in 0s 1194.98 KB/s done
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg ls -r s3://ltds-s3-training-01/prefix-1
2026-04-13 07:42      1048576  s3://ltds-s3-training-01/prefix-1/1MiB
2026-04-13 07:55      1048576  s3://ltds-s3-training-01/prefix-1/file-2.1MiB
```

6.8. Example: Giving other user Read Write List Delete access to part of your bucket

In this example we give ARN `arn:aws:iam::RGW88505570706778033:user/ltids-s3-training-02` write to prefix `prefix-1` from bucket `ltids-s3-training-01`

This bucket-policy allows ARN to Download, Upload List and Delete objects from prefix.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "AllowListOnlyPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/ltids-s3-training-02"]},
      "Action": "s3:ListBucket",
      "Resource": "arn:aws:s3:::ltids-s3-training-01",
      "Condition": {
        "StringLike": { "s3:prefix": [ "prefix-1/*", "prefix-1" ] }
      }
    },
    {
      "Sid": "AllowReadWriteObjectsInPrefix",
      "Effect": "Allow",
      "Principal": {"AWS": ["arn:aws:iam::RGW88505570706778033:user/ltids-s3-training-02"]},
      "Action": [
        "s3:GetObject",
        "s3:PutObject",
        "s3:DeleteObject"
      ],
      "Resource": "arn:aws:s3:::ltids-s3-training-01/prefix-1/*"
    }
  ]
}
```

Testing

User `ltids-s3-training-02` can list `prefix-1`

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltids-training-02.cfg ls -r s3://ltids-s3-training-01/prefix-1
2026-04-13 07:42      1048576  s3://ltids-s3-training-01/prefix-1/1MiB
```

User `ltids-s3-training-02` can download objects from `prefix-1`

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltids-training-02.cfg get s3://ltids-s3-training-01/prefix-1/1MiB
download: 's3://ltids-s3-training-01/prefix-1/1MiB' -> './1MiB' [1 of 1]
1048576 of 1048576 100% in 0s 2.80 MB/s done
```

User `ltids-s3-training-02` can download objects from `prefix-1`

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltids-training-02.cfg put example_files/1MiB s3://ltids-s3-training-01/prefix-1/file-3.1MiB
upload: 'example_files/1MiB' -> 's3://ltids-s3-training-01/prefix-1/file-3.1MiB' [1 of 1]
1048576 of 1048576 100% in 0s 1373.95 KB/s done
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltids-training-02.cfg ls -r s3://ltids-s3-training-01/prefix-1
2026-04-13 07:42      1048576  s3://ltids-s3-training-01/prefix-1/1MiB
2026-04-13 08:04      1048576  s3://ltids-s3-training-01/prefix-1/file-3.1MiB
```

User `ltids-s3-training-02` can delete objects from `prefix-1`

```
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg rm s3://ltds-s3-training-01/prefix-1/file-3.1MiB
delete: 's3://ltds-s3-training-01/prefix-1/file-3.1MiB'
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-02.cfg ls -r s3://ltds-s3-training-01/prefix-1
2026-04-13 07:42      1048576  s3://ltds-s3-training-01/prefix-1/1MiB
```

6.9. Example: Making part of your bucket Public by enabling unauthenticated download

In this case anyone with a URL to a bucket will be able to download object from this specific path.

For example object **s3://ltds-s3-training-01/public-prefix/file-0.txt** Will be downloadable from URL <https://eurofusion-2.s3.storage.psync.pl/ltds-s3-training-01/public-prefix/file-0.txt>

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "PublicReadPrefix",
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::ltds-s3-training-01/public-prefix/*"
    }
  ]
}
```

Testing

Uploading file to public-prefix

```
$ ~/tmp/s3-tutorial -> echo Hello World > file-0.txt
$ ~/tmp/s3-tutorial -> s3cmd -c s3cmd_configs/s3cfg-ltds-training-01.cfg put file-0.txt s3://ltds-s3-training-01/public-prefix/file-0.txt
upload: 'file-0.txt' -> 's3://ltds-s3-training-01/public-prefix/file-0.txt' [1 of 1]
12 of 12 100% in 0s 224.66 B/s done
```

Downloading file using unauthenticated client

```
curl https://eurofusion-2.s3.storage.psync.pl/ltds-s3-training-01/public-prefix/file-0.txt
Hello World
```

File can also be viewed in the browser

