SKYLABLE

LibreS3 Manual

Contents

Chapter 1

Introduction

LibreS3 is a robust Open Source implementation of the Amazon S3 service, supporting a subset of the S3 REST API.

Standard S3 client libraries and tools can be used to access it (for example s3cmd, Python-boto, DragonDisk, etc.).

LibreS3 uses Skylable S^X as the storage backend, which provides data deduplication and replication.

1.1 Useful links

- `http://lists.skylable.com`
- `https://bugzilla.skylable.com`
- `http://www.skylable.com/docs/`
- `http://www.skylable.com/products/libres3`

CHAPTER 2

INSTALLATION

LibreS3 is regularly tested on Linux and FreeBSD. We recommend using the binary packages from http://www.skylable.com/download if your platform is supported.

2.1 DOCKER

Stable releases of LibreS3 are available as tags on the Docker hub, e.g.:

```
docker pull skylable/libres3:release-1.2
```

Latest master is always available as:

```
docker pull skylable/libres3:latest
```

Follow the instructions on the Docker hub for configuring and running this container: http://hub.docker.com/r/skylable/libres3

2.2 BINARY PACKAGES

DEBIAN WHEEZY AND JESSIE

Add the following entry to /etc/apt/sources.list.d/skylable.list:

```
deb http://cdn.skylable.com/debian wheezy main
```

then run the following commands:

```
# curl 'https://pgp.mit.edu/pks/lookup?op=get&search=0x5377E192B7BC1D2E' | sudo apt-key add
      -
# apt-get install libres3
```

CENTOS 6/7

Create the file /etc/yum.repos.d/skylable-sx.repo with this content:

```
[skylable-sx]
name=Skylable SX
baseurl=http://cdn.skylable.com/centos/$releasever/$basearch
enabled=1
gpgcheck=0
```

then execute:

```
# yum install libres3
```

Fedora 21+

Create the file `/etc/yum.repos.d/skylable-sx.repo` with this content:

```
[skylable-sx]
name=Skylable SX
baseurl=http://cdn.skylable.com/fedora/$releasever/$basearch
enabled=1
gpgcheck=0
```

then execute:

```
# yum install libres3
```

2.3 Source code

On most Unix platforms you can compile LibreS3 from source. You will need the following packages to be installed together with their development versions:

- OCaml (>= 3.12.1)
- camlp4 (matching your OCaml compiler version)
- OpenSSL
- PCRE C library
- GNU Make and m4
- Zlib
- pkg-config

For example, on Debian run:

```
# apt-get install ocaml-native-compilers camlp4-extra libssl-dev libpcre3-dev zlib1g-dev\
 pkg-config make m4
```

On Fedora run:

```
# yum install ocaml /usr/bin/camlp4of /usr/bin/camlp4rf /usr/bin/camlp4 openssl-devel\
 pcre-devel zlib-devel pkgconfig make m4 ncurses-devel
```

Compilation

Follow the standard installation procedure to install LibreS3 into the default location (`/usr/local`):

```
$ ./configure && make && make check
# make install
```

The rest of the manual assumes that LibreS3 was installed from a binary package, so some paths may be different.

Note: On OpenSolaris/OmniOS you need some additional flags:

```
$ CPPFLAGS=-m64 ./configure --destdir=${DESTDIR} && make && make check
# make install
```

CHAPTER 3

CONFIGURATION

3.1 REQUIREMENTS

LibreS3 by default listens on ports 8008 and 8443, which need to be available on a given IP address.

LibreS3 connects to the S^X cluster via `HTTP(S)`. You can run LibreS3 and S^X on the same or different hosts.

LibreS3 doesn't store state on the local filesystem, when it does need to share data with other LibreS3 instances (e.g. multipart uploads) it will use an S^X volume for that purpose. You can run any number of LibreS3 instances, to provide load-balancing or failover.

LibreS3 1.2+ requires at least SX version 2.0 for all the features to work correctly. `libres3_setup` will check if the SX cluster is running a supported version.

DNS ZONE ENTRY

S3 buckets require a wildcard `A` record pointing to the IP address (1.2.3.4 below) of the host running LibreS3, for example:

```
*.libres3.example.com. A 1.2.3.4
libres3.example.com. A 1.2.3.4
```

If you don't have control over the DNS you'll have to modify the `/etc/hosts` file of each client machine and add a line for each bucket you want to access:

```
libres3.example.com 1.2.3.4
bucket1.libres3.example.com 1.2.3.4
bucket2.libres3.example.com 1.2.3.4
```

You can also add `CNAME` for virtual hosting buckets:

```
bucket.example.net CNAME bucket.example.net.libres3.example.com.
```

You can of course just add directly `A` or `AAAA` entries that point directly to the LibreS3 server, it only matters that clients send the desired bucket name in the `Host:` header.

3.2 SETTING UP A LIBRES3 NODE

Setting up LibreS3 is as simple as running the interactive tool `libres3_setup`. If you provide the path to an existing S^X cluster configuration file created by `sxsetup`, most of the settings will be done automatically.

Please make sure the default volume replica count setting is less or equal to the number of nodes in the S^X cluster.

In the examples below we assume that you have an S^X cluster already up and running and you want to deploy LibreS3 on `libres3.example.com`.

Example setup with sxsetup.conf:

```
# libres3_setup --sxsetup-conf /etc/sxserver/sxsetup.conf
Successfully loaded SX configuration from '/etc/sxserver/sxsetup.conf'

S3 (DNS) name: libres3.example.com

Locking LibreS3 private settings on SX cluster: sx://admin@192.168.1.192:443/
libres3_setup: main: locking cluster for changes

Generating default SSL certificate and key in /etc/ssl/certs/libres3.pem and /etc/ssl/
     private/libres3.key
Generating a 2048 bit RSA private key
.....+++
..+++
writing new private key to '/etc/ssl/private/libres3.key'
-----
Uploaded generated SSL certificate and key to SX cluster
libres3_setup: main: unlocking cluster for changes
Settings completed

S3 HTTPS port: 8443

S3 HTTP port: 8008

Default volume size [use K, M, G and T suffixes]: 100G

Default volume replica count: 1

Generating '/etc/libres3/libres3.conf'
File '/etc/libres3/libres3.conf' already exists, overwriting
libres3_setup: main: locking cluster for changes
libres3_setup: main: Updating cluster metadata
libres3_setup: main: unlocking cluster for changes
volume_size = 100G
s3_https_port = 8443
s3_http_port = 8008
s3_host = libres3.example.com
replica_count = 1
allow_volume_create_any_user = true
Updating '/etc/libres3/libres3.sample.s3cfg'
Generating '/etc/libres3/libres3.sample.boto'
Updating '/etc/libres3/libres3-insecure.sample.s3cfg'
Generating '/etc/libres3/libres3-insecure.sample.boto'

Do you want to start LibreS3 now? [Y/n] Y
[....] Restarting LibreS3: libres3No libres3_ocsigen found running; none killed.

Loading configuration from /etc/libres3/libres3.conf
Waiting for server to start (5s) ... OK
```

Example without sxsetup.conf: [1]

```
# libres3_setup
Admin key or path to key-file: 0DPiKuNIrrVmD8IUCuw1hQxNqZfJ0hlBUgyckAolodd4C/4r4ecY3QAA

SX server IP/DNS name: sx.example.com

SX server HTTPS port: 443

Run as user: nobody

Run as group: nogroup
```

[1] you can use `sxadm node --info /var/lib/sxserver/storage/` on the S^X node to find out the required information

```
S3 (DNS) name: libres3.example.com

Locking LibreS3 private settings on SX cluster: sx://admin@192.168.1.192:443/
libres3_setup: main: locking cluster for changes

Generating default SSL certificate and key in /etc/ssl/certs/libres3.pem and /etc/ssl/
     private/libres3.key
Generating a 2048 bit RSA private key
.....+++
..+++
writing new private key to '/etc/ssl/private/libres3.key'
-----
Uploaded generated SSL certificate and key to SX cluster
libres3_setup: main: unlocking cluster for changes
Settings completed

S3 HTTPS port: 8443

S3 HTTP port: 8008

Default volume size [use K, M, G and T suffixes]: 100G

Default volume replica count: 1

Generating '/etc/libres3/libres3.conf'
File '/etc/libres3/libres3.conf' already exists, overwriting
libres3_setup: main: locking cluster for changes
libres3_setup: main: Updating cluster metadata
libres3_setup: main: unlocking cluster for changes
volume_size = 100G
s3_https_port = 8443
s3_http_port = 8008
s3_host = libres3.example.com
replica_count = 1
allow_volume_create_any_user = true
Updating '/etc/libres3/libres3.sample.s3cfg'
Generating '/etc/libres3/libres3.sample.boto'
Updating '/etc/libres3/libres3-insecure.sample.s3cfg'
Generating '/etc/libres3/libres3-insecure.sample.boto'

Do you want to start LibreS3 now? [Y/n] Y
[....] Restarting LibreS3: libres3No libres3_ocsigen found running; none killed.

Loading configuration from /etc/libres3/libres3.conf
Waiting for server to start (5s) ... OK
```

You can use `--batch` and provide all settings on the command line, see `libres3_setup --help` for the full list of options.

To start/stop LibreS3[2]:

```
# libres3 start
Starting LibreS3
LibreS3 started successfully
# libres3 status
--- LibreS3 STATUS ---
LibreS3 is running (PID 28245)

--- LibreS3 INFO ---
SSL private key: /etc/ssl/private/libres3.key
LibreS3 logs: /var/log/libres3/
# libres3 stop
Loading configuration from /etc/libres3/libres3.conf
Sending TERM to PID 28245 ...
Waiting for PID 28245 ...
```

If the server doesn't start, please check the log files for details.

That's it — your LibreS3 cloud storage is already up and running! You can now connect to it with your favorite S3 client.

[2]LibreS3 and S^X will communicate using TLS by default. For debugging purposes you can configure S^X with `sxsetup --no-ssl`

3.3 Advanced settings

Internal IP addresses

If you've configured S^X with public and internal IP addresses LibreS3 can only access S^X via its public IP address, so make sure it is reachable from LibreS3 and that you set the public IP address of S^X in `sx_host`.

libres3.conf

When you run `libres3_setup` it will generate a file `/etc/libres3/libres3.conf` and store settings in SX cluster metadata.

LibreS3 tries to be compatible with the S3 protocol by default, however sometimes there are features outside of the S3 protocol that would be useful. These can be configured in `libres3.conf` and all have the `allow_` prefix:

allow_list_all_volumes (default=true) only volumes owned by you are listed by default, see Listing buckets on page 17.

allow_public_bucket_index (default=false) by default you can't list a bucket or subdirectory from a browser, see Directory indexing on page 21.

allow_volume_create_any_user (default=true) by default any authenticated user can create a bucket that doesn't exist yet. In S^X only admin users can perform such an action, and you can configure LibreS3 to provide that behaviour, however some S3 applications may not behave correctly anymore.

There are more settings that tune various aspects of LibreS3, an up-to-date list[3] can always be found in the generated `libres3.conf`: each configuration option has a comment briefly describing its purpose, and settings that are disabled by default are commented out.

You can customize `libres3.conf` and restart LibreS3 for the changes to take effect. However if you are running multiple LibreS3 nodes a more convenient way to update the settings on the entire cluster is:

```
# libres3_setup --update key=value sx://admin@sx.example.com
```

And then run `libres3 reload` on each node. This will store the LibreS3 settings in SX cluster metadata, which LibreS3 reads on startup or when it receives a SIGHUP signal.

For example to enable public bucket indexing:

```
# libres3_setup --update allow_public_bucket_index=true sx://admin@sx.example.com
Previous configuration:
	volume_size = 10G
	s3_http_port = 8008
	s3_host = libres3.example.com
	replica_count = 2
	allow_volume_create_any_user = true
	allow_public_bucket_index = false

libres3_setup: main: locking cluster for changes
libres3_setup: main: Updating cluster metadata
libres3_setup: main: unlocking cluster for changes

New configuration:
	volume_size = 10G
	s3_http_port = 8008
```

[3] this list doesn't include settings used for debugging/tuning low-level aspects

```
        s3_host = libres3.example.com
        replica_count = 2
        allow_volume_create_any_user = true
        allow_public_bucket_index = true
```

SSL CERTIFICATES

When you run `libres3_setup` on the first node a self-signed wildcard SSL certificate is generated, and uploaded to the SX cluster private settings (accessible only to admin users). On the next invocation of `libres3_setup` it will download the SSL certificate and key from the SX cluster (if available), to ensure that the same certificate is used on all LibreS3 nodes.

To reset the SSL certificate and allow generating a new one:

```
# sxadm cluster --set-param libres3_private= sx://admin@sx.example.com
```

CHAPTER 4

CLIENT CONFIGURATION

4.1 AUTHENTICATION

S3 clients require that you provide an “Access key” and a “Secret key”. The “Access key” is your S^X username, and your “Secret key” is your S^X token (key).

In case you are accessing your S^X cluster using a username and password, you need to find out your S^X token by running `sxinit` if you haven't already done so:

```
$ sxinit sx://username@clustername
$ cat $HOME/.sx/clustername/auth/username
```

4.2 S3CMD

You can use the generated s3cfg config file [1] or configure s3cmd from scratch. Below we assume that your LibreS3 is running on `libres3.example.com` and it supports TLS. The important s3cmd configuration settings are:

```
use_https True
host_base libres3.example.com:8443
host_bucket %(bucket)s.libres3.example.com:8443
access_key <your-sx-username>
secret_key <your-sx-key>
ca_certs_file s/etc/ssl/certs/libres3.pem
```

If you don't use TLS, please use the port 8008 instead of 8443, and set `use_https` to `False`. Once you've configured s3cmd check that it properly connects to LibreS3:

```
$ s3cmd ls --debug 2>&1 | grep host
```

Supported s3cmd commands:

Bucket `mb`, `rb`, `ls`, `la`, `du`, `info`, `setpolicy`, `delpolicy`, `multipart`

Object `put`, `get`, `del`, `rm`, `sync`, `info`, `cp`, `modify`, `mv`, `abortmp`, `listmp`, `sign`, `signurl`

ENCRYPTED FILES

You can set `gpg_passphrase` in `.s3cfg` and use `s3cmd --encrypt` to upload/-download encrypted files [2].

[1] `/etc/libres3/libres3.sample.s3cfg`

[2] Accessing files on S^X volumes that use the aes filter is not supported, because this would require decrypting the files by LibreS3 server-side, and encrypted volumes in S^X are meant to be used with client-side encryption

Caveats

Certificate verification changed in Python version 2.7.9+, and you'll need s3cmd version 1.5.1 or newer to match. The .s3cfg must contain a ca_certs_file entry pointing to the certificate of the LibreS3 server, otherwise certificate verification (and thus HTTPS connections) will fail.

Note that wildcard TLS certificates only match one level, hence you should avoid using bucket names which contain dots.

For example a certificate for *.s3.example.com is:

- valid for a.s3.example.com
- NOT valid for a.b.s3.example.com

4.3 Python-boto

S3 clients using Python boto are configured in ~/.boto, or you can use the generated[3] file. A typical configuration is:

```
[Credentials]
aws_access_key_id=<your-sx-username>
aws_secret_access_key=<your-sx-key>
s3_host=libres3.example.com
s3_port=8443
[Boto]
is_secure = True
ca_certificates_file=/etc/libres3/libres3.pem
https_validate_certificates=True
```

Note that setting s3_host will override the hostname you give to applications on the command-line. If you are using an application that allows setting the S3 hostname on the command-line, you might want to use that instead.

Note that old version of python-boto require the port to be on s3_host instead of s3_port.

Caveats

Certificate verification changed in Python version 2.7.9+, and you'll need to add ca_certificates_file and https_validate_certificates=True entries in the [Boto] section of your .boto file. Otherwise python-boto applications will keep trying to reconnect, and eventually fail with:

```
ssl.SSLError: [SSL: CERTIFICATE_VERIFY_FAILED] certificate verify failed (_ssl.c:581)
```

The libres3_setup generated python-boto configuration files already have the necessary entries since LibreS3 version 1.1.

4.4 s3fs-fuse

S3FS can be used to provide a FUSE-based file system backed by LibreS3.

You must specify the S^X username and access token in ~/.passwd-s3fs, the URL for your LibreS3 server with -o url, and the certificate used by LibreS3 in CURL_CA_BUNDLE:

```
$ cat >~/.passwd-s3fs <<EOF
admin:0DPiKuNIrrVmD8IUCuw1hQxNqZfJ0hlBUgyckAolodd4C/4r4ecY3QAA
EOF
```

[3] /etc/libres3/libres3.sample.boto

```
$ chmod 0600 ~/.passwd-s3fs
$ mkdir ~/libres3-vol1
$ CURL_CA_BUNDLE=/etc/ssl/certs/libres3.pem s3fs -o url=https://libres3.example.com:8443
     vol1 ~/libres3-vol1 -o uid=1000
```

If you want to access LibreS3 unencrypted for debugging purposes then:

```
$ s3fs -o url=http://libres3.example.com:8008 vol1 ~/libres3-vol1 -o uid=1000
```

Caveats

If s3fs fails to connect to LibreS3 it quits without an error message, and the next time you access the mountpoint you get an error, including trying to start s3fs again:

```
$ touch ~/libres3-vol1/x
touch: cannot touch '/home/USER/libres3-vol1/x': Transport endpoint is not connected
$ s3fs ...
s3fs: unable to access MOUNTPOINT /home/USER/libres3-vol1: Transport endpoint is not
     connected
$ fusermount -u ~/libres3-vol1
$ s3fs ...
```

You'll have to manually unmount and it is useful to run `s3fs` in the foreground to see the actual error message:

```
$ fusermount -u ~/libres3-vol1
$ CURL_CA_BUNDLE=/etc/ssl/certs/libres3.pem s3fs -o url=https://libres3.example.com:8443
     vol1 ~/libres3-vol1 -o uid=1000 -f
[...]
```

Common errors are using the wrong user/token in `~/.passwd-s3fs`, or s3fs reading the wrong passwd-s3fs (from path/etc for example), or that you don't have permissions to access the volume, etc.

4.5 DragonDisk

Add an account configured like this:

Provider Other S3 compatible service

Service endpoint `libres3.example.com` (`s3_host` from `libres3.conf`)

Access key your S^X username

Secret key your S^X secret token

HTTP port 8008 (`s3_http_port` from `libres3.conf`)

HTTPS port 8443 (`s3_https_port` from `libres3.conf`)

4.6 SDKs

The recommended way to pass the credentials to the AWS SDKs is to store them in the file `~/.aws/credentials`:

```
[default]
aws_access_key_id=<your-sx-username>
aws_secret_access_key=<your-sx-key>
```

PHP

After you install the SDK (with `composer require aws/aws-sdk-php`) you need to set the endpoint and SSL certificate location to point to LibreS3:

```
require 'vendor/autoload.php';
use Aws\S3\S3Client;

$endpoint='libres3.example.com';

$s3Client = new Aws\S3\S3Client([
    'version' => 'latest',
    'region' => 'us-east-1',
// 'debug' => true,
   'endpoint' => "https://$endpoint:8443",
   'http' => ['verify' => '/etc/ssl/certs/libres3.pem']
]);
```

Java

Import LibreS3's certificate into Java's store (replace with actual) as root:

```
# export JAVA_HOME=/usr/lib/jvm/java-8-openjdk-amd64
# keytool -storepass changeit -import -alias libres3 -keystore $JAVA_HOME/jre/lib/security/
     cacerts -file /etc/ssl/certs/libres3.pem
```

You can use a `build.gradle` file like this:

```
group 'com.example'
version '1.0-SNAPSHOT'

buildscript {
    repositories {
        mavenCentral()
    }

    dependencies {
        classpath "io.spring.gradle:dependency-management-plugin:0.5.4.RELEASE"
    }

}

apply plugin: "io.spring.dependency-management"
apply plugin: 'java'

sourceCompatibility = 1.5

repositories {
    mavenCentral()
}

dependencyManagement {
    imports {
        mavenBom 'com.amazonaws:aws-java-sdk-bom:1.10.49'
    }
}

dependencies {
    compile 'com.amazonaws:aws-java-sdk-s3', 'log4j:log4j:1.2.17'
    testCompile group: 'junit', name: 'junit', version: '4.11'
}

jar {
    from { configurations.compile.collect { it.isDirectory() ? it : zipTree(it) } }
    manifest {
        attributes 'Main-Class': 'com.example.Main'
    }
}
```

Then you have to set the endpoint on the S3 client object:

```
  AWSCredentials credentials = null;
  try {
    credentials = new ProfileCredentialsProvider().getCredentials();
  } catch (Exception e) {
    // ...
  }

  System.setProperty(SDKGlobalConfiguration.ENFORCE_S3_SIGV4_SYSTEM_PROPERTY, "true");
  AmazonS3 s3 = new AmazonS3Client(credentials);
  s3.setEndpoint("https://libres3.example.com:8443/");
```

Boto3

To connect to the HTTP endpoint (plain text – for testing purposes only!):

```
endpoint = "http://libres3.example.com:8008"
self.session = botocore.session.get_session()
self.region = 'us-east-1'
self.client = self.session.create_client('s3',
                                     region_name=self.region,
                                     endpoint_url=endpoint)
self.client.meta.events.unregister('before-sign.s3', fix_s3_host)
```

CHAPTER 5

HIGHLIGHTED FEATURES

5.1 VIRTUAL HOSTING

Buckets in LibreS3 can be accessed in three ways:

path-style `libres3.example.com/bucket/object`

A direct access to the object, which works even if the bucket name is not DNS compatible.

virtual-hosted subdomain style `bucket.libres3.example.com/object`

Requires that bucket names are DNS compatible and a DNS/hosts entry for each bucket or a wildcard DNS record (see DNS zone entry on page 6)

virtual-hosted FQDN `bucket.example.net/object`

Equivalent to `bucket.example.net.libres3.example.com`.

The client should send a `Host: bucket.example.net` header, which usually requires a `CNAME` DNS entry, see DNS zone entry on page 6. Requires that bucket names are DNS compatible, and that you use your FQDN as the bucket name, for example:

```
s3cmd mb s3://bucket.example.net
```

The latter can be useful for public buckets, see Virtual hosting for public buckets on page 20.

5.2 STREAMING AND RANGE-REQUESTS

LibreS3 supports streaming files from S^X by downloading blocks as they are needed (streaming a large file begins as soon as the first batch of blocks is downloaded by LibreS3).

It is also possible to begin streaming a file from an arbitrary position, and retrieve less than the entire file, i.e. range requests.

This works both for authenticated (`s3cmd get --continue`) and public requests (`wget --continue`), see Streaming videos on page 21.

5.3 Access logs

Since version `1.1` LibreS3 uses the Apache/NCSA Combined Log Format for its `access.log` file, for example:

```
196.168.1.2 - edwin [06/Jul/2015:08:38:20 +0200] "GET /volpublic/?policy HTTP/1.1" 200 190
     "-" "Boto/2.34.0 Python/2.7.9 Linux/3.16.0-4-amd64"
192.168.1.2 - - [06/Jul/2015:09:58:26 +0200] "GET /slider_logo.png HTTP/1.1" 200 11436 "
     http://volpublic.libres3.example.com:8008/" "Mozilla/5.0 (X11; Linux x86_64; rv:38.0)
     Gecko/20100101 Firefox/38.0 Iceweasel/38.1.0"
```

The first line is an example for an authenticated request, and the second line is an example for an anonymous request (using directory indexing according to the referer).

The format is:

192.168.1.2 IP address of client

- always unavailable (the `identd` identity)

edwin name of the authenticated S^X user. If the HTTP result code is `401` or `5xx` then it cannot be trusted because the user may not be authenticated yet

[06/Jul/2015:09:58:26 +0200] the server's local date, time and timezone

"GET /volpublic/?policy HTTP/1.1" the request method, path and HTTP version

200 the HTTP result code

190 the size of the reply body sent to the client. Note that if the connection is closed/reset then this shows the truncated size, not the size that would have been sent. This is shown as - when the body is empty.

"-" no HTTP referer available

"Boto/2.34.0 Python/2.7.9 Linux/3.16.0-4-amd64" User-Agent of the client (not all S3 clients send a user-agent, for example `s3cmd` doesn't)

Access.log entries are written asynchronously, after the entire reply body is sent to the client.

5.4 Listing buckets

`s3cmd ls s3://` shows only the buckets that are owned by your user, what is consistent with S3's behaviour. In case you have access to some buckets in read or write mode and would like to use them with s3cmd or other clients, you can enable listing of all accessible buckets by adding the following to `libres3.conf` and restarting LibreS3:

```
allow_list_all_volumes=true
```

5.5 IPv6 support

LibreS3 supports IPv6 communication with S^X version `1.2` or newer (note that an S^X node can have either an IPv4 or an IPv6 public address, but not both at the same time). LibreS3 can be reached either by IPv4 or IPv6, depending on how your DNS wildcard record is set up, and whether your S3 client and OS support IPv6.

5.6 Quota support

Each volume in S^X has a maximum size (its quota). There is no equivalent concept in S3, thus you need to specify the default volume size (and replica count) for buckets created through S3 in `libres3.conf` using the `volume_size` field. This is done automatically if you used `libres3_setup` to configure LibreS3.

In case a new file would exceed the S^X volume's quota, LibreS3 should report an HTTP 413 error. The quota errors will be reported for both normal and multipart uploads, however in the case of multipart ones, the quota error will only be reported after all parts have been uploaded to LibreS3.

LibreS3 creates an internal volume for each user to store the parts uploaded during a multipart upload. This volume's name is prefixed by `libres3-` and is created with the size specified in the `volume_size` field and replica count 1. You have to ensure that the size is sufficient to hold all in-progress (and interrupted) multipart uploads for a user, otherwise one may run out of space during multipart upload even if the destination volume has plenty of space available.

Interrupted multipart uploads are not automatically removed (because one can resume by providing the upload ID), to clean them first run `s3cmd multipart` on each bucket in turn to find out the objects and upload IDs, and `s3cmd abortmp` on each upload ID in turn to actually delete it:

```
$ s3cmd multipart -c s3://bucket
$ s3cmd abortmp s3://bucket/object Id
```

CHAPTER 6

PUBLIC ACCESS TO OBJECTS

6.1 SIGNED URLS

To create a signed URL valid for 24 hours for retrieving the object x from the bucket vol1:

```
# s3cmd -c /etc/libres3/libres3.sample.s3cfg signurl s3://vol1/x ‘date -d ’24 hours’ +%s‘ |
      sed -e ’s/http:/https:/’
https://vol1.libres3.example.com:8443/x?AWSAccessKeyId=admin&Expires=1421780366&Signature=
     uT1uzWiRKOcr%2F459zjLvmWoMTSg%3D
```

The URL can be pasted into a browser, and used with curl or wget to download the file without additional authorization. After 24 hours the URL expires and can’t be used anymore.

There is also a python script as a sample for writing your applications that would generate signed URLs:

```
$ git clone http://git.skylable.com/experimental
$ cd experimental/s3genlink
# BOTO_CONFIG=/etc/libres3/libres3.sample.boto ./s3genlink.py vol1/x
```

6.2 PUBLIC READ-ONLY ACCESS TO OBJECTS

Signing URLs can be inconvenient if you want to make all objects in a bucket publicly accessible. In this case you can enable public access to all objects in a bucket by setting a bucket policy (replace `volpublic` with the name of your bucket):

```
$ cat >anon.json <<EOF
{"Version":"2012-10-17",
 "Statement" : [{ "Sid":"AddPerm",
    "Effect": "Allow", "Principal": "*",
    "Action": ["s3:GetObject"],
    "Resource":["arn:aws:s3:::volpublic/*"]}]}
EOF
# s3cmd -c /etc/libres3/libres3.sample.s3cfg setpolicy anon.json s3://volpublic
# s3cmd -c /etc/libres3/libres3.sample.s3cfg info s3://volpublic
[...]
$ wget volpublic.libres3.example.com:8008/x
$ wget libres3.example.com:8008/volpublic/x
```

To disable public access just delete the policy:

```
# s3cmd -c /etc/libres3/libres3.sample.s3cfg delpolicy s3://volpublic
```

Note that public/anonymous users can only download individual objects — they are not allowed to list the bucket’s contents.

There are some scripts to help managing public buckets:

```
$ git clone http://git.skylable.com/experimental
$ cd experimental/s3publicvol
# BOTO_CONFIG=/etc/libres3/libres3.sample.boto ./s3publicvol.py volpublic/x
Using access key id: admin
Turning on public access for bucket volpublic
# BOTO_CONFIG=/etc/libres3/libres3.sample.boto ./s3policy.py volpublic/x
Using access key id: admin
Bucket volpublic has an access policy:
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": "*",
      "Action": "s3:GetObject",
      "Resource": "arn:aws:s3:::volpublic/*"
    }
  ]
}
# BOTO_CONFIG=/etc/libres3/libres3.sample.boto ./s3privatevol.py volpublic/x
Using access key id: admin
Turning off public access for bucket volpublic
# BOTO_CONFIG=/etc/libres3/libres3.sample.boto ./s3policy.py volpublic/x
Using access key id: admin
Bucket volpublic is private
```

Note that using ACLs to set individual objects as publicly accessible is not supported.

6.3 Setting Content-Type, Cache-Control and other headers

When uploading an object the following headers can be set:

`Cache-Control`, `Content-Disposition`, `Content-Encoding`, `Expires`, and `Content-Type`.

When downloading an object (either authenticated or from a public bucket), LibreS3 will send these customized header values if they are set. Therefore an object's Content-Type on download will match the value used on upload via LibreS3. If you use `s3cmd` to upload or copy your files the `Content-Type` will be determined correctly in most cases, but you can override it with `--mime-type=`. For files uploaded via S^X the `Content-Type` is guessed automatically based on file extension.

You can modify the headers of already uploaded files:

```
s3cmd modify --add-header 'Cache-Control: max-age=86400' s3://bucket/object
```

If you want to set a header for all currently existing objects in a bucket:

```
s3cmd modify --add-header 'Cache-Control: max-age=86400' s3://bucket --recursive
```

6.4 Virtual hosting for public buckets

Using a FQDN as bucket name (see Virtual hosting on page 16) can be useful if you want to host a website through HTTP. All public bucket features are supported with virtual hosted buckets.

Note that when using TLS LibreS3 will still serve its self-signed wildcard certificate for `*.libres3.example.com` even with virtual hosted buckets, so this is not suitable for serving a website over TLS.

6.5 Streaming videos

Using LibreS3's range request support[1], a web browser or media player can play videos directly from a LibreS3 public bucket, and the clients can efficiently seek to a particular position in the video!

Note that the file must be fully uploaded and its size known before streaming can begin, so this won't work for live video streams.

6.6 Directory indexing

A public bucket only allows accessing files directly by name, if you try to access the root of the bucket or a "directory" then you get an error message in XML.

Although the S3 API doesn't provide this feature directly (static website mode would allow a static `index.html` only), you can enable directory listing for public buckets in LibreS3 by adding the following to `libres3.conf` and restarting LibreS3 (please make sure all LibreS3 instances get updated):

```
allow_public_bucket_index=true
```

As long as you access the buckets using virtual hosted style (see Virtual hosting for public buckets on page 20) you will be able to navigate the directory listing of the bucket directly from your web browser by just opening the root of the bucket:

`http://volpublic.libres3.example.com:8008/`

6.7 Query parameters

LibreS3 ignores any query parameters in GET that are not part of a known S3 API (`acl`,`cors`,...).

Some websites use a query parameter in the URL of images/scripts accessed from an HTML file to make sure that the new/up-to-date version is used instead of an old cached version. LibreS3 supports that, however it would probably be more reliable to change the filename of those assets each time they are modified (e.g. by embedding its hash in the name).

6.8 Using caching proxy or web accelerator

LibreS3 sends an `ETag` header for each file based on the coresponding S^X file's revision. This uniquely identifies the file across the cluster and is guaranteed to change every time you upload a new file. A `Last-Modified` header is also sent for files.

These two headers should already be sufficient to use a caching proxy or a web accelerator (such as Varnish), however for better results you should also set Cache-Control headers with a `max-age`, see Setting Content-Type, Cache-Control and other headers on page 20.

[1] see Streaming and range-requests on page 16

CHAPTER 7

TROUBLESHOOTING

7.1 LIBRES3_REPORT

If you face a problem connecting to your LibreS3 server, please run `libres3_report` which generates a file `libres3-report-*.log` in the current directory, containing details about LibreS3's build and runtime configuration, and logs.

By default the secret key is removed from the output, but names and IP addresses are not anonymized to make debugging easier. You can anonymize the logged information by running `libres3_report --anonymize` which replaces each name and IP address with a hash.

7.2 LOGS

You can read the logs directly at `/var/log/libres3/*.log`.

To enable logging of full HTTP requests run the following commands:

```
# export LIBRES3_DEBUG=1
# libres3 restart
```

To enable debugging of the LibreS3 and S^X communication:

```
# libres3 stop
# libres3_ocsigen --foreground --debug
```

7.3 FURTHER INFORMATION

For more information and FAQ please visit http://www.skylable.com/docs/.

If you can't find your solution there, please subscribe to our mailing list at http://lists.skylable.com and post about your issues.