Configuring Object Storage as Primary Storage

- Differences from External Storage
  - Performance Implications
  - Data Backup and Recovery Implications
- Configuration
  - OpenStack Swift
  - Simple Storage Service (S3)
  - Microsoft Azure Blob Storage
- Multibucket Object Store
- Multibucket Object Store with per Bucket configuration overrides
- Multi-instance Object Store
- S3 SSE-C encryption support

Nextcloud allows to configure object storages like OpenStack Swift or Amazon Simple Storage Service (S3) or any compatible S3-implementation (e.g. Minio or Ceph Object Gateway) as primary storage replacing the default storage of files.

By default, files are stored in `nextcloud/data` or another directory configured in the `config.php` of your Nextcloud instance. This data directory might still be used for compatibility reasons)

Differences from External Storage

When an object store is used as Primary Storage, Nextcloud requires exclusive access over the bucket being used. All metadata (filenames, directory structures, etc) is stored in Nextcloud and not in the object store. The metadata is only stored in the database and the object store only holds the file content by unique identifier.

Performance Implications

Because of this, object stores configured as Primary Storage usually perform better than when using the same object store via the External Storage support application, but the downside is being unable to access the files from outside of Nextcloud. This makes using an object store as Primary Storage distinct from using an object store via External Storage.

Data Backup and Recovery Implications

One impact of using an object store as Primary Storage is that your data backup strategy needs to incorporate this. **Your data is no longer stored on your Nextcloud server, but your files are also no longer accessible by simply bypassing your Nextcloud server and accessing your object store directly.**

Configuration

Primary object stores need to be configured in `config.php` by specifying the objectstore backend and any backend specific configuration.

> “□□
>
> Configuring a primary object store on an existing Nextcloud instance will make all existing files on the instance inaccessible.

The configuration has the following structure:

```
'objectstore' => [
    'class' => 'Object\\Storage\\Backend\\Class',
    'arguments' => [
        ...
    ],
],
```

OpenStack Swift

The OpenStack Swift backend mounts a container on an OpenStack Object Storage server into the virtual filesystem.

The class to be used is `OCFilesObjectStoreSwift`

Both openstack v2 and v3 authentication are supported,

V2 Authentication:

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\Swift',
    'arguments' => [
        'username' => 'username',
        'password' => 'Secr3tPaSSWoRdt7',
        // the container to store the data in
        'bucket' => 'nextcloud',
        'autocreate' => true,
        'region' => 'RegionOne',
        // The Identity / Keystone endpoint
        'url' => 'http://example.com/v2.0',
        // optional on some swift implementations
        'tenantName' => 'username',
        'serviceName' => 'swift',
        // The Interface / url Type, optional
        'urlType' => 'internal'
    ],
],
```

V3 Authentication:

```
'objectstore' => [
    'class' => 'OC\\Files\\ObjectStore\\Swift',
    'arguments' => [
        'autocreate' => true,
        'user' => [
            'name' => 'UserName',
            'password' => 'Secr3tPaSSWoRdt7',
            'domain' => [
                'name' => 'Default',
            ],
        ],
        'scope' => [
            'project' => [
                'name' => 'TenantName',
                'domain' => [
```

```
                    'name' => 'Default',
                ],
            ],
        ],
        'serviceName' => 'swift',
        'region' => 'regionOne',
        'url' => 'http://example.com/v3',
        'bucket' => 'nextcloud',
    ],
],
```

Simple Storage Service (S3)

The Simple Storage Service (S3) backend mounts a bucket on an Amazon S3 object storage or compatible implementation (e.g. Minio or Ceph Object Gateway) into the virtual filesystem.

The class to be used is `OCFilesObjectStoreS3`

Amazon-hosted S3:

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\S3',
    'arguments' => [
        'bucket' => 'my-nextcloud-store',
        'region' => 'us-east-1',
        'key' => 'EJ39ITYZEUH5BGWDRUFY',
        'secret' => 'M5MrXTRjkyMaxXPe2FRXMTfTfbKEnZCu+7uRTVSj',
    ],
],
```

Non-Amazon hosted S3:

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\S3',
    'arguments' => [
        'bucket' => 'my-nextcloud-store',
        'hostname' => 's3.example.com',
        'key' => 'EJ39ITYZEUH5BGWDRUFY',
        'secret' => 'M5MrXTRjkyMaxXPe2FRXMTfTfbKEnZCu+7uRTVSj',
        'port' => 8443,
```

```
        // required for some non-Amazon S3 implementations
        'use_path_style' => true,
    ],
],
```

Minimum required parameters are:

- `bucket` [Note: Even if non-Amazon hosted, bucket names must meet AWS S3 naming requirements regardless of what your S3 provider/platform considers acceptable - i.e. no underscores]
- `key`
- `secret`

> “□□
>
> You will *probably* need to specify additional parameters beyond these, unless the default values (see below) exactly match your situation. In particular, your `region` (if Amazon hosted) or `hostname` (if non-Amazon hosted).

Optional parameters most commonly needing adjustment (and their defaults values if left unconfigured):

- `region` defaults to `eu-west-1`
- `storageClass` defaults to `STANDARD`
- `hostname` defaults to `s3.REGION.amazonaws.com` [Note: If using this parameter (non-Amazon), specify the generic S3 endpoint hostname, **not** the hostname that contains your bucket name]
- `use_ssl` defaults to `true`

Optional parameters sometimes needing adjustment:

- `use_path_style` defaults to `false`
- `port` defaults to `443`
- `sse_c_key` has no default

Optional parameters less commonly needing adjustment:

- `concurrency` defaults to `5` [Note: This defines the maximum number of concurrent multipart uploads]
- `proxy` defaults to `false`
- `connect_timeout` defaults to `5` [Note: the connection timeout is set in seconds, but decimal precision can be used for subsecond accuracy (for example, 4.2 for 4200 milliseconds)]
- `timeout` defaults to `15`

- `uploadPartSize` defaults to `524288000`
- `putSizeLimit` defaults to `104857600`
- `useMultipartCopy` defaults to `true`
- `copySizeLimit` defaults to `5242880000`
- `legacy_auth` has no default
- `version` defaults to `latest`
- `verify_bucket_exists` defaults to `true` [Note: Setting this to `false` *after* confirming the bucket has been created may provide a performance benefit, but may not be possible in multibucket scenarios.]

If you are using Amazon S3: the `region` parameter is required unless you're happy with the default of `eu-west-1`. There is no need to override the `hostname` or `port`. And `storageClass` only needs to be modified if you're using a different configuration at AWS. Lastly, `use_path_style` is rarely required with Amazon, but some legacy Amazon datacenters may require it.

If you using a non-Amazon hosted S3 store: you will need to set the `hostname` parameter (and can ignore the `region` parameter). You may need to use `use_path_style` if your non-Amazon S3 store does *not* support requests like `https://bucket.hostname.domain/`. Setting `use_path_style` to true configures the S3 client to make requests like `https://hostname.domain/bucket` instead.

Microsoft Azure Blob Storage

The Azure Blob Storage backend mounts a container on Microsoft's Azure Blob Storage into the virtual filesystem.

The class to be used is `OCFilesObjectStoreAzure`

```
'objectstore' => [
    'class' => '\\OC\\Files\\ObjectStore\\Azure',
    'arguments' => [
        'container' => 'nextcloud',
        'autocreate' => true,
        'account_name' => 'account_name',
        'account_key' => 'xxxxxxxxxx'
    ],
],
```

Multibucket Object Store

It's possible to configure Nextcloud to distribute the data over multiple buckets for scalability purposes.

To setup multiple buckets, set `'multibucket => true'` in the object store configuration in `config.php`:

```
'objectstore' => [
    'class' => 'Object\\Storage\\Backend\\Class',
    'arguments' => [
        'multibucket' => true,
        // optional, defaults to 64
        'num_buckets' => 64,
        // will be postfixed by an integer in the range from 0 to (num_nuckets-1)
        'bucket' => 'nextcloud_',
        ...
    ],
],
```

Multibucket object store backend maps every user to a range of buckets and saves all files for that user in their corresponding bucket.

> “□□
>
> While it is possible to change the number of buckets used by an existing Nextcloud instance, the user-to-buckets mapping is only created once, so only newly created users will be mapped to the updated range of buckets.

You can find out more information about upscaling with object storage and Nextcloud in the Nextcloud customer portal.

Multibucket Object Store with per Bucket configuration overrides

When using an Object Store with `'multibucket => true'` it is possible to configure overrides for all config options per bucket:

```
'objectstore' => [
    'class' => 'Object\\Storage\\Backend\\Class',
    'arguments' => [
        'multibucket' => true,
```

```
        'bucket' => 'nextcloud_',
        'perBucket' => [
            'nextcloud_1' => [
                'port' => 9999,
            ],
        ],
    ],
],
```

This can be useful for example if you want to configure credentials per bucket that is used by a Team folder. A script for provisioning new Team folders this way could look like this (first make sure the bucket exists with those credentials):

```
occ config:system:set --type=string --value=KEYVALUE objectstore arguments perBucket
BUCKETNAME key
occ config:system:set --type=string --value=SECRETVALUE objectstore arguments perBucket
BUCKETNAME secret
occ groupfolders:create --bucket BUCKETNAME TEAMFOLDERNAME
```

The credentials must be set before the new Team folder is created.

Multi-instance Object Store

It's possible to configure Nextcloud to distribute the data over multiple object store instances for further scaling and gradual migration.

To setup multiple buckets, set `'objectstore'` to an array of named configurations configuration in `config.php` and set the `'default'` to the name of the configuration to use for newly created users:

```
'objectstore' => [
    'default' => 'server2',
    'root' => 'server1',
    'server1' => [
        'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => [
            'hostname' => 's3-server1.example.com',
            'bucket' => 's1_nextcloud',
            ...
        ],
    ],
```

```
    'server2' => [
    'class' => 'Object\\Storage\\Backend\\Class',
        'arguments' => [
            'multibucket' => true,
            'hostname' => 's3-server2.example.com',
            'bucket' => 's2_nextcloud_',
            ...
        ],
    ],
],
```

> “□□
>
> Bucket names must be unique between all configured object store instances.

Newly created users will be mapped to the object store instance set in `default`. Files that are not part of the users storage are put in the `root` instance, or in the `default` instance if no `root` instance is configured.

In the above example, if `server2` is starting to run low on capacity, an admin can setup and configure a new `server3` and change the `default` to `server3`. Than any newly created user will have their files put on `server3`.

> “□□
>
> As with multibucket object store, the user-to-instance mapping is only created once, so only newly created users will be mapped to the new default instance.

It is possible to mix different object store backends and multibucket and non-multibucket in a multi-instance configuration.

S3 SSE-C encryption support

Nextcloud supports server side encryption, also known as SSE-C, with compatible S3 bucket provider. The encryption and decryption happens on the S3 bucket side with a key provided by the Nextcloud server.

The key can be specified with the `sse_c_key` parameter which needs to be provided as a base64 encoded string with a maximum length of 32 bytes. A random key could be generated using the the following command:

```
openssl rand 32 | base64
```

The following example shows how to configure the S3 object store with SSE-C encryption support in the objectstore section of the Nextcloud config.php file:

```
'objectstore' => [
    array (
        'class' => 'OC\\Files\\ObjectStore\\S3',
        'arguments' =>
        array (
            'bucket' => 'nextcloud',
            'key' => 'nextcloud',
            'secret' => 'nextcloud',
            'hostname' => 's3',
            'port' => '443',
            'use_ssl' => true,
            'use_path_style' => true,
            'autocreate' => true,
            'verify_bucket_exists' => true,
            'sse_c_key' => 'o9d3Q9tHcPMv6TIpH53MSXaUmY91YheZRwuIhwCFRSs=',
        ),
    );
],
```

□□□ #1
□ cocobook-sync □□□ 2018-08-10 03:12:43 UTC
□ cocobook-sync □□□ 2026-05-06 16:34:01 UTC