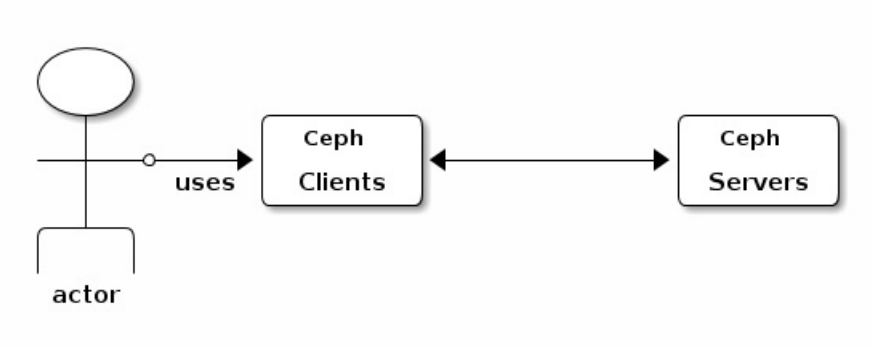
USER MANAGEMENT

This document describes Ceph Client users, and their authentication and authorization with the Ceph Storage Cluster. Users are either individuals or system actors such as applications, which use Ceph clients to interact with the Ceph Storage Cluster daemons.



When Ceph runs with authentication and authorization enabled (enabled by default), you must specify a user name and a keyring containing the secret key of the specified user (usually via the command line). If you do not specify a user name, Ceph will use `client.admin` as the default user name. If you do not specify a keyring, Ceph will look for a keyring via the `keyring` setting in the Ceph configuration. For example, if you execute the `ceph health` command without specifying a user or keyring:

```
ceph health
```

Ceph interprets the command like this:

```
ceph -n client.admin --keyring=/etc/ceph/ceph.client.admin.keyring health
```

Alternatively, you may use the `CEPH_ARGS` environment variable to avoid re-entry of the user name and secret.

For details on configuring the Ceph Storage Cluster to use authentication, see Cephx Config Reference. For details on the architecture of Cephx, see Architecture - High Availability Authentication.

BACKGROUND

Irrespective of the type of Ceph client (e.g., Block Device, Object Storage, Filesystem, native API, etc.), Ceph stores all data as objects within pools. Ceph users must have access to pools in order to read and write data. Additionally, Ceph users must have execute permissions to use Ceph's administrative commands. The following concepts will help you understand Ceph user management.

USER

A user is either an individual or a system actor such as an application. Creating users allows you to control who (or what) can access your Ceph Storage Cluster, its pools, and the data within pools.

Ceph has the notion of a `type` of user. For the purposes of user management, the type will always be `client`. Ceph identifies users in period (.) delimited form consisting of the user type and the user ID: for example, `TYPE.ID`, `client.admin`, or `client.user1`. The reason for user typing is that Ceph Monitors, OSDs, and Metadata Servers also use the Cephx protocol, but they are not clients. Distinguishing the user type helps to distinguish between client users and other users–streamlining access control, user monitoring and traceability.

Sometimes Ceph's user type may seem confusing, because the Ceph command line allows you to specify a user with or without the type, depending upon your command line usage. If you specify `--user` or `--id`, you can omit the type. So `client.user1` can be entered simply as `user1`. If you specify `--name` or `-n`, you must specify the type and name, such as `client.user1`. We recommend using the type and name as a best practice wherever possible.

Note: A Ceph Storage Cluster user is not the same as a Ceph Object Storage user or a Ceph Filesystem user. The Ceph Object Gateway uses a Ceph Storage Cluster user to communicate between the gateway daemon and the storage cluster, but the gateway has its own user management functionality for end users. The Ceph Filesystem uses POSIX semantics. The

user space associated with the Ceph Filesystem is not the same as a Ceph Storage Cluster user.

AUTHORIZATION (CAPABILITIES)

Ceph uses the term “capabilities” (caps) to describe authorizing an authenticated user to exercise the functionality of the monitors, OSDs and metadata servers. Capabilities can also restrict access to data within a pool, a namespace within a pool, or a set of pools based on their application tags. A Ceph administrative user sets a user’s capabilities when creating or updating a user.

Capability syntax follows the form:

```
{daemon-type} '{cap-spec}[, {cap-spec} ...]'
```

- **Monitor Caps:** Monitor capabilities include `r`, `w`, `x` access settings or `profile {name}`. For example:

  ```
  mon 'allow {access-spec}'

  mon 'profile {name}'
  ```

  The `{access-spec}` syntax is as follows:

  ```
  * | all | [r][w][x]
  ```

- **OSD Caps:** OSD capabilities include `r`, `w`, `x`, `class-read`, `class-write` access settings or `profile {name}`. Additionally, OSD capabilities also allow for pool and namespace settings.

  ```
  osd 'allow {access-spec} [{match-spec}]'

  osd 'profile {name} [pool={pool-name} [namespace={namespace-name}]]'
  ```

  The `{access-spec}` syntax is either of the following:

  ```
  * | all | [r][w][x] [class-read] [class-write]

  class {class name} [{method name}]
  ```

  The optional `{match-spec}` syntax is either of the following:

  ```
  pool={pool-name} [namespace={namespace-name}] [object_prefix {prefix}]

  [namespace={namespace-name}] tag {application} {key}={value}
  ```

- **Metadata Server Caps:** For administrators, use `allow *`. For all other users, such as CephFS clients, consult CephFS Client Capabilities

Note: The Ceph Object Gateway daemon (`radosgw`) is a client of the Ceph Storage Cluster, so it is not represented as a Ceph Storage Cluster daemon type.

The following entries describe each access capability.

`allow`

Description: Precedes access settings for a daemon. Implies `rw` for MDS only.

`r`

Description: Gives the user read access. Required with monitors to retrieve the CRUSH map.

`w`

Description: Gives the user write access to objects.

`x`

Description: Gives the user the capability to call class methods (i.e., both read and write) and to conduct `auth` operations on monitors.

`class-read`

Descriptions: Gives the user the capability to call class read methods. Subset of `x`.

`class-write`

Description: Gives the user the capability to call class write methods. Subset of `x`.

`*`, `all`

Description: Gives the user read, write and execute permissions for a particular daemon/pool, and the ability to execute admin commands.

The following entries describe valid capability profiles:

`profile osd` (Monitor only)

Description: Gives a user permissions to connect as an OSD to other OSDs or monitors. Conferred on OSDs to enable OSDs to handle replication heartbeat traffic and status reporting.

`profile mds` (Monitor only)

Description: Gives a user permissions to connect as a MDS to other MDSs or monitors.

`profile bootstrap-osd` (Monitor only)

Description: Gives a user permissions to bootstrap an OSD. Conferred on deployment tools such as `ceph-volume`, `ceph-deploy`, etc. so that they have permissions to add keys, etc. when bootstrapping an OSD.

`profile bootstrap-mds` (Monitor only)

Description: Gives a user permissions to bootstrap a metadata server. Conferred on deployment tools such as `ceph-deploy`, etc. so they have permissions to add keys, etc. when bootstrapping a metadata server.

`profile rbd` (Monitor and OSD)

Description: Gives a user permissions to manipulate RBD images. When used as a Monitor cap, it provides the minimal privileges required by an RBD client application. When used as an OSD cap, it provides read-write access to an RBD client application.

`profile rbd-read-only` (OSD only)

Description: Gives a user read-only permissions to RBD images.

POOL

A pool is a logical partition where users store data. In Ceph deployments, it is common to create a pool as a logical partition for similar types of data. For example, when deploying Ceph as a backend for OpenStack, a typical deployment would have pools for volumes, images, backups and virtual machines, and users such as `client.glance`, `client.cinder`, etc.

APPLICATION TAGS

Access may be restricted to specific pools as defined by their application metadata. The * wildcard may be used for the key argument, the `value` argument, or both. `all` is a synony for *.

NAMESPACE

Objects within a pool can be associated to a namespace–a logical group of objects within the pool. A user's access to a pool can be associated with a namespace such that reads and writes by the user take place only within the namespace. Objects written to a namespace within the pool can only be accessed by users who have access to the namespace.

Note: Namespaces are primarily useful for applications written on top of `librados` where the logical grouping can alleviate the need to create different pools. Ceph Object Gateway (from `luminous`) uses namespaces for various metadata objects.

The rationale for namespaces is that pools can be a computationally expensive method of segregating data sets for the purposes of authorizing separate sets of users. For example, a pool should have ~100 placement groups per OSD. So an exemplary cluster with 1000 OSDs would have 100,000 placement groups for one pool. Each pool would create another 100,000 placement groups in the exemplary cluster. By contrast, writing an object to a namespace simply associates the namespace to the object name with out the computational overhead of a separate pool. Rather than creating a separate pool for a user or set of users, you may use a namespace. **Note:** Only available using `librados` at this time.

Access may be restricted to specific RADOS namespaces using the `namespace` capability. Limited globbing of namespaces is supported; if the last character of the specified namespace is *, then access is granted to any namespace starting with the provided argument.

MANAGING USERS

User management functionality provides Ceph Storage Cluster administrators with the ability to create, update and delete users directly in the Ceph Storage Cluster.

When you create or delete users in the Ceph Storage Cluster, you may need to distribute keys to clients so that they can be added to keyrings. See Keyring Management for details.

LIST USERS

To list the users in your cluster, execute the following:

```
ceph auth ls
```

Ceph will list out all users in your cluster. For example, in a two-node exemplary cluster, `ceph auth ls` will output something that looks like this:

```
installed auth entries:

osd.0
        key: AQCvCbtToC6MDhAATtuT70Sl+DymPCfDSsyV4w==
        caps: [mon] allow profile osd
        caps: [osd] allow *
osd.1
        key: AQC4CbtTCFJBChAAVq5spj0ff4eHZICxIOVZeA==
        caps: [mon] allow profile osd
        caps: [osd] allow *
client.admin
        key: AQBHCbtT6APDHhAA5W00cBchwkQjh3dkKsyPjw==
        caps: [mds] allow
        caps: [mon] allow *
        caps: [osd] allow *
client.bootstrap-mds
        key: AQBICbtTOK9uGBAAdbe5zcIGHZL3T/u2g6EBww==
        caps: [mon] allow profile bootstrap-mds
client.bootstrap-osd
        key: AQBHCbtT4GxqORAADE5u7RkpCN/oo4e5W0uBtw==
        caps: [mon] allow profile bootstrap-osd
```

Note that the `TYPE.ID` notation for users applies such that `osd.0` is a user of type `osd` and its ID is `0`, `client.admin` is a user of type `client` and its ID is `admin` (i.e., the default `client.admin` user). Note also that each entry has a `key: <value>` entry, and one or more `caps:` entries.

You may use the `-o {filename}` option with `ceph auth ls` to save the output to a file.

GET A USER

To retrieve a specific user, key and capabilities, execute the following:

```
ceph auth get {TYPE.ID}
```

For example:

```
ceph auth get client.admin
```

You may also use the `-o {filename}` option with `ceph auth get` to save the output to a file. Developers may also execute the following:

```
ceph auth export {TYPE.ID}
```

The `auth export` command is identical to `auth get`, but also prints out the internal `auid`, which is not relevant to end users.

ADD A USER

Adding a user creates a username (i.e., `TYPE.ID`), a secret key and any capabilities included in the command you use to create the user.

A user's key enables the user to authenticate with the Ceph Storage Cluster. The user's capabilities authorize the user to read, write, or execute on Ceph monitors (`mon`), Ceph OSDs (`osd`) or Ceph Metadata Servers (`mds`).

There are a few ways to add a user:

- `ceph auth add`: This command is the canonical way to add a user. It will create the user, generate a key and add any specified capabilities.
- `ceph auth get-or-create`: This command is often the most convenient way to create a user, because it returns a keyfile format with the user name (in brackets) and the key. If the user already exists, this command simply returns the user name and key in the keyfile format. You may use the `-o {filename}` option to save the output to a file.
- `ceph auth get-or-create-key`: This command is a convenient way to create a user and return the user's key (only). This is useful for clients that need the key only (e.g., libvirt). If the user already exists, this command simply returns the key. You may use the `-o {filename}` option to save the output to a file.

When creating client users, you may create a user with no capabilities. A user with no capabilities is useless beyond mere authentication, because the client cannot retrieve the cluster map from the monitor. However, you can create a user with no capabilities if you wish to defer adding capabilities later using the `ceph auth caps` command.

A typical user has at least read capabilities on the Ceph monitor and read and write capability on Ceph OSDs. Additionally, a user's OSD permissions are often restricted to accessing a particular pool.

```
ceph auth add client.john mon 'allow r' osd 'allow rw pool=liverpool'
ceph auth get-or-create client.paul mon 'allow r' osd 'allow rw pool=liverpool'
ceph auth get-or-create client.george mon 'allow r' osd 'allow rw pool=liverpool' -o george.k
ceph auth get-or-create-key client.ringo mon 'allow r' osd 'allow rw pool=liverpool' -o ringo
```

Important: If you provide a user with capabilities to OSDs, but you DO NOT restrict access to particular pools, the user will have access to ALL pools in the cluster!

MODIFY USER CAPABILITIES

The `ceph auth caps` command allows you to specify a user and change the user's capabilities. Setting new capabilities will overwrite current capabilities. To view current capabilities run `ceph auth get USERTYPE.USERID`. To add capabilities, you should also specify the existing capabilities when using the form:

```
ceph auth caps USERTYPE.USERID {daemon} 'allow [r|w|x|*|...] [pool={pool-name}] [namespace={n
```

For example:

```
ceph auth get client.john
ceph auth caps client.john mon 'allow r' osd 'allow rw pool=liverpool'
ceph auth caps client.paul mon 'allow rw' osd 'allow rwx pool=liverpool'
ceph auth caps client.brian-manager mon 'allow *' osd 'allow *'
```

To remove a capability, you may reset the capability. If you want the user to have no access to a particular daemon that was previously set, specify an empty string. For example:

```
ceph auth caps client.ringo mon ' ' osd ' '
```

See Authorization (Capabilities) for additional details on capabilities.

DELETE A USER

To delete a user, use `ceph auth del`:

```
ceph auth del {TYPE}.{ID}
```

Where `{TYPE}` is one of `client`, `osd`, `mon`, or `mds`, and `{ID}` is the user name or ID of the daemon.

PRINT A USER'S KEY

To print a user's authentication key to standard output, execute the following:

```
ceph auth print-key {TYPE}.{ID}
```

Where `{TYPE}` is one of `client`, `osd`, `mon`, or `mds`, and `{ID}` is the user name or ID of the daemon.

Printing a user's key is useful when you need to populate client software with a user's key (e.g., libvirt).

```
mount -t ceph serverhost:/ mountpoint -o name=client.user,secret=`ceph auth print-key client.
```

IMPORT A USER(S)

To import one or more users, use `ceph auth import` and specify a keyring:

```
ceph auth import -i /path/to/keyring
```

For example:

```
sudo ceph auth import -i /etc/ceph/ceph.keyring
```

Note: The ceph storage cluster will add new users, their keys and their capabilities and will update existing users, their keys and their capabilities.

KEYRING MANAGEMENT

When you access Ceph via a Ceph client, the Ceph client will look for a local keyring. Ceph presets the `keyring` setting with the following four keyring names by default so you don't have to set them in your Ceph configuration file unless you want to override the defaults (not recommended):

- /etc/ceph/$cluster.$name.keyring
- /etc/ceph/$cluster.keyring
- /etc/ceph/keyring
- /etc/ceph/keyring.bin

The $cluster metavariable is your Ceph cluster name as defined by the name of the Ceph configuration file (i.e., ceph.conf means the cluster name is ceph; thus, ceph.keyring). The $name metavariable is the user type and user ID (e.g., client.admin; thus, ceph.client.admin.keyring).

Note: When executing commands that read or write to /etc/ceph, you may need to use sudo to execute the command as root.

After you create a user (e.g., client.ringo), you must get the key and add it to a keyring on a Ceph client so that the user can access the Ceph Storage Cluster.

The User Management section details how to list, get, add, modify and delete users directly in the Ceph Storage Cluster. However, Ceph also provides the ceph-authtool utility to allow you to manage keyrings from a Ceph client.

CREATE A KEYRING

When you use the procedures in the Managing Users section to create users, you need to provide user keys to the Ceph client(s) so that the Ceph client can retrieve the key for the specified user and authenticate with the Ceph Storage Cluster. Ceph Clients access keyrings to lookup a user name and retrieve the user's key.

The ceph-authtool utility allows you to create a keyring. To create an empty keyring, use --create-keyring or -C. For example:

```
ceph-authtool --create-keyring /path/to/keyring
```

When creating a keyring with multiple users, we recommend using the cluster name (e.g., $cluster.keyring) for the keyring filename and saving it in the /etc/ceph directory so that the keyring configuration default setting will pick up the filename without requiring you to specify it in the local copy of your Ceph configuration file. For example, create ceph.keyring by executing the following:

```
sudo ceph-authtool -C /etc/ceph/ceph.keyring
```

When creating a keyring with a single user, we recommend using the cluster name, the user type and the user name and saving it in the /etc/ceph directory. For example, ceph.client.admin.keyring for the client.admin user.

To create a keyring in /etc/ceph, you must do so as root. This means the file will have rw permissions for the root user only, which is appropriate when the keyring contains administrator keys. However, if you intend to use the keyring for a particular user or group of users, ensure that you execute chown or chmod to establish appropriate keyring ownership and access.

ADD A USER TO A KEYRING

When you Add a User to the Ceph Storage Cluster, you can use the Get a User procedure to retrieve a user, key and capabilities and save the user to a keyring.

When you only want to use one user per keyring, the Get a User procedure with the -o option will save the output in the keyring file format. For example, to create a keyring for the client.admin user, execute the following:

```
sudo ceph auth get client.admin -o /etc/ceph/ceph.client.admin.keyring
```

Notice that we use the recommended file format for an individual user.

When you want to import users to a keyring, you can use ceph-authtool to specify the destination keyring and the source keyring. For example:

```
sudo ceph-authtool /etc/ceph/ceph.keyring --import-keyring /etc/ceph/ceph.client.admin.keyrin
```

CREATE A USER

Ceph provides the Add a User function to create a user directly in the Ceph Storage Cluster. However, you can also create a user, keys and capabilities directly on a Ceph client keyring. Then, you can import the user to the Ceph Storage Cluster. For example:

```
sudo ceph-authtool -n client.ringo --cap osd 'allow rwx' --cap mon 'allow rwx' /etc/ceph/ceph
```

See Authorization (Capabilities) for additional details on capabilities.

You can also create a keyring and add a new user to the keyring simultaneously. For example:

```
sudo ceph-authtool -C /etc/ceph/ceph.keyring -n client.ringo --cap osd 'allow rwx' --cap mon
```

In the foregoing scenarios, the new user `client.ringo` is only in the keyring. To add the new user to the Ceph Storage Cluster, you must still add the new user to the Ceph Storage Cluster.

```
sudo ceph auth add client.ringo -i /etc/ceph/ceph.keyring
```

MODIFY A USER

To modify the capabilities of a user record in a keyring, specify the keyring, and the user followed by the capabilities. For example:

```
sudo ceph-authtool /etc/ceph/ceph.keyring -n client.ringo --cap osd 'allow rwx' --cap mon 'al
```

To update the user to the Ceph Storage Cluster, you must update the user in the keyring to the user entry in the the Ceph Storage Cluster.

```
sudo ceph auth import -i /etc/ceph/ceph.keyring
```

See Import a User(s) for details on updating a Ceph Storage Cluster user from a keyring.

You may also Modify User Capabilities directly in the cluster, store the results to a keyring file; then, import the keyring into your main `ceph.keyring` file.

COMMAND LINE USAGE

Ceph supports the following usage for user name and secret:

`--id` | `--user`

Description: Ceph identifies users with a type and an ID (e.g., `TYPE.ID` or `client.admin`, `client.user1`). The `id`, `name` and `-n` options enable you to specify the ID portion of the user name (e.g., `admin`, `user1`, `foo`, etc.). You can specify the user with the `--id` and omit the type. For example, to specify user `client.foo` enter the following:

```
ceph --id foo --keyring /path/to/keyring health
ceph --user foo --keyring /path/to/keyring health
```

`--name` | `-n`

Description: Ceph identifies users with a type and an ID (e.g., `TYPE.ID` or `client.admin`, `client.user1`). The `--name` and `-n` options enables you to specify the fully qualified user name. You must specify the user

type (typically `client`) with the user ID. For example:

```
ceph --name client.foo --keyring /path/to/keyring health
ceph -n client.foo --keyring /path/to/keyring health
```

`--keyring`

Description: The path to the keyring containing one or more user name and secret. The `--secret` option provides the same functionality, but it does not work with Ceph RADOS Gateway, which uses `--secret` for another purpose. You may retrieve a keyring with `ceph auth get-or-create` and store it locally. This is a preferred approach, because you can switch user names without switching the keyring path. For example:

```
sudo rbd map --id foo --keyring /path/to/keyring mypool/myimage
```

LIMITATIONS

The cephx protocol authenticates Ceph clients and servers to each other. It is not intended to handle authentication of human users or application programs run on their behalf. If that effect is required to handle your access control needs, you must have another mechanism, which is likely to be specific to the front end used to access the Ceph object store. This other mechanism has the role of ensuring that only acceptable users and programs are able to run on the machine that Ceph will permit to access its object store.

The keys used to authenticate Ceph clients and servers are typically stored in a plain text file with appropriate permissions in a trusted host.

Important: Storing keys in plaintext files has security shortcomings, but they are difficult to avoid, given the basic authentication methods Ceph uses in the background. Those setting up Ceph systems should be aware of these shortcomings.

In particular, arbitrary user machines, especially portable machines, should not be configured to interact directly with Ceph, since that mode of use would require the storage of a plaintext authentication key on an insecure machine. Anyone who stole that machine or obtained surreptitious access to it could obtain the key that will allow them to authenticate their own machines to Ceph.

Rather than permitting potentially insecure machines to access a Ceph object store directly, users should be required to sign in to a trusted machine in your environment using a method that provides sufficient security for your purposes. That trusted machine will store the plaintext Ceph keys for the human users. A future version of Ceph may address these particular authentication issues more fully.

At the moment, none of the Ceph authentication protocols provide secrecy for messages in transit. Thus, an eavesdropper on the wire can hear and understand all data sent between clients and servers in Ceph, even if it cannot create or alter them. Further, Ceph does not include options to encrypt user data in the object store. Users can hand-encrypt and store their own data in the Ceph object store, of course, but Ceph provides no features to perform object encryption itself. Those storing sensitive data in Ceph should consider encrypting their data before providing it to the Ceph system.