RBD – MANAGE RADOS BLOCK DEVICE (RBD) IMAGES

SYNOPSIS

rbd [-c *ceph.conf*] [-m *monaddr*] [-p | –pool *pool*] [–size *size*] [–order *bits*] [*command* ...]

DESCRIPTION

rbd is a utility for manipulating rados block device (RBD) images, used by the Linux rbd driver and the rbd storage driver for Qemu/KVM. RBD images are simple block devices that are striped over objects and stored in a RADOS object store. The size of the objects the image is striped over must be a power of two.

OPTIONS

-c `ceph.conf`, **--conf** `ceph.conf`
Use ceph.conf configuration file instead of the default /etc/ceph/ceph.conf to determine monitor addresses during startup.

-m `monaddress[:port]`
Connect to specified monitor (instead of looking through ceph.conf).

-p `pool`, **--pool** `pool`
Interact with the given pool. Required by most commands.

--no-progress
Do not output progress information (goes to standard error by default for some commands).

PARAMETERS

--image-format `format`
Specifies which object layout to use. The default is 1.

- format 1 - Use the original format for a new rbd image. This format is understood by all versions of librbd and the kernel rbd module, but does not support newer features like cloning.
- format 2 - Use the second rbd format, which is supported by librbd (but not the kernel rbd module) at this time. This adds support for cloning and is more easily extensible to allow more features in the future.

--size `size-in-mb`
Specifies the size (in megabytes) of the new rbd image.

--order `bits`
Specifies the object size expressed as a number of bits, such that the object size is `1 << order`. The default is 22 (4 MB).

--stripe-unit `size-in-bytes`
Specifies the stripe unit size in bytes. See striping section (below) for more details.

--stripe-count `num`
Specifies the number of objects to stripe over before looping back to the first object. See striping section (below) for more details.

--snap `snap`
Specifies the snapshot name for the specific operation.

--id `username`
Specifies the username (without the `client.` prefix) to use with the map command.

--keyfile `filename`
Specifies a file containing the secret to use with the map command. If not specified, `client.admin` will be used by default.

--keyring `filename`

Specifies a keyring file containing a secret for the specified user to use with the map command. If not specified, the default keyring locations will be searched.

--shared `tag`

Option for *lock add* that allows multiple clients to lock the same image if they use the same tag. The tag is an arbitrary string. This is useful for situations where an image must be open from more than one client at once, like during live migration of a virtual machine, or for use underneath a clustered filesystem.

--format `format`

Specifies output formatting (default: plain, json, xml)

--pretty-format

Make json or xml formatted output more human-readable.

COMMANDS

ls [-l | –long] [pool-name]

Will list all rbd images listed in the rbd_directory object. With -l, also show snapshots, and use longer-format output including size, parent (if clone), format, etc.

info [*image-name*]

Will dump information (such as size and order) about a specific rbd image. If image is a clone, information about its parent is also displayed. If a snapshot is specified, whether it is protected is shown as well.

create [*image-name*]

Will create a new rbd image. You must also specify the size via –size. The –stripe-unit and –stripe-count arguments are optional, but must be used together.

clone [*parent-snapname*] [*image-name*]

Will create a clone (copy-on-write child) of the parent snapshot. Object order will be identical to that of the parent image unless specified. Size will be the same as the parent snapshot.

The parent snapshot must be protected (see *rbd snap protect*). This requires format 2.

flatten [*image-name*]

If image is a clone, copy all shared blocks from the parent snapshot and make the child independent of the parent, severing the link between parent snap and child. The parent snapshot can be unprotected and deleted if it has no further dependent clones.

This requires format 2.

children [*image-name*]

List the clones of the image at the given snapshot. This checks every pool, and outputs the resulting poolname/imagename.

This requires format 2.

resize [*image-name*] [–allow-shrink]

Resizes rbd image. The size parameter also needs to be specified. The –allow-shrink option lets the size be reduced.

rm [*image-name*]

Deletes an rbd image (including all data blocks). If the image has snapshots, this fails and nothing is deleted.

export [*image-name*] [*dest-path*]

Exports image to dest path (use - for stdout).

import [*path*] [*dest-image*]

Creates a new image and imports its data from path (use - for stdin). The import operation will try to create sparse rbd images if possible. For import from stdin, the sparsification unit is the data block size of the destination image (1 << order).

export-diff [*image-name*] [*dest-path*] [–from-snap *snapname*]

Exports an incremental diff for an image to dest path (use - for stdout). If an initial snapshot is specified, only changes since that snapshot are included; otherwise, any regions of the image that contain data are included. The end snapshot is specified using the standard –snap option or @snap syntax (see below). The image diff format includes metadata about image size changes, and the start and end snapshots. It efficiently represents discarded or ‘zero’ regions of the image.

import-diff [*src-path*] [*image-name*]

Imports an incremental diff of an image and applies it to the current image. If the diff was generated relative to a start snapshot, we verify that snapshot already exists before continuing. If there was an end snapshot we verify it does not already exist before applying the changes, and create the snapshot when we are done.

diff [*image-name*] [–from-snap *snapname*]
Dump a list of byte extents in the image that have changed since the specified start snapshot, or since the image was created. Each output line includes the starting offset (in bytes), the length of the region (in bytes), and either ‘zero’ or ‘data’ to indicate whether the region is known to be zeros or may contain other data.

cp [*src-image*] [*dest-image*]
Copies the content of a src-image into the newly created dest-image. dest-image will have the same size, order, and format as src-image.

mv [*src-image*] [*dest-image*]
Renames an image. Note: rename across pools is not supported.

snap ls [*image-name*]
Dumps the list of snapshots inside a specific image.

snap create [*image-name*]
Creates a new snapshot. Requires the snapshot name parameter specified.

snap rollback [*image-name*]
Rollback image content to snapshot. This will iterate through the entire blocks array and update the data head content to the snapshotted version.

snap rm [*image-name*]
Removes the specified snapshot.

snap purge [*image-name*]
Removes all snapshots from an image.

snap protect [*image-name*]
Protect a snapshot from deletion, so that clones can be made of it (see *rbd clone*). Snapshots must be protected before clones are made; protection implies that there exist dependent cloned children that refer to this snapshot. *rbd clone* will fail on a nonprotected snapshot.

This requires format 2.

snap unprotect [*image-name*]
Unprotect a snapshot from deletion (undo *snap protect*). If cloned children remain, *snap unprotect* fails. (Note that clones may exist in different pools than the parent snapshot.)

This requires format 2.

map [*image-name*]
Maps the specified image to a block device via the rbd kernel module.

unmap [*device-path*]
Unmaps the block device that was mapped via the rbd kernel module.

showmapped
Show the rbd images that are mapped via the rbd kernel module.

lock list [*image-name*]
Show locks held on the image. The first column is the locker to use with the *lock remove* command.

lock add [*image-name*] [*lock-id*]
Lock an image. The lock-id is an arbitrary name for the user’s convenience. By default, this is an exclusive lock, meaning it will fail if the image is already locked. The –shared option changes this behavior. Note that locking does not affect any operation other than adding a lock. It does not protect an image from being deleted.

lock remove [*image-name*] [*lock-id*] [*locker*]
Release a lock on an image. The lock id and locker are as output by lock ls.

bench-write [*image-name*] –io-size [*io-size-in-bytes*] –io-threads [*num-ios-in-flight*] –io-total [*total-bytes-to-write*]
Generate a series of sequential writes to the image and measure the write throughput and latency. Defaults are: –io-size 4096, –io-threads 16, –io-total 1GB

IMAGE NAME

In addition to using the –pool and the –snap options, the image name can include both the pool name and the snapshot name.

The image name format is as follows:

```
[pool/]image-name[@snap]
```

Thus an image name that contains a slash character (‘/’) requires specifying the pool name explicitly.

STRIPING

RBD images are striped over many objects, which are then stored by the Ceph distributed object store (RADOS). As a result, read and write requests for the image are distributed across many nodes in the cluster, generally preventing any single node from becoming a bottleneck when individual images get large or busy.

The striping is controlled by three parameters:

order
The size of objects we stripe over is a power of two, specifially 2^[*order*] bytes. The default is 22, or 4 MB.

stripe_unit
Each [*stripe_unit*] contiguous bytes are stored adjacently in the same object, before we move on to the next object.

stripe_count
After we write [*stripe_unit*] bytes to [*stripe_count*] objects, we loop back to the initial object and write another stripe, until the object reaches its maximum size (as specified by [*order*]. At that point, we move on to the next [*stripe_count*] objects.

By default, [*stripe_unit*] is the same as the object size and [*stripe_count*] is 1. Specifying a different [*stripe_unit*] requires that the STRIPINGV2 feature be supported (added in Ceph v0.53) and format 2 images be used.

EXAMPLES

To create a new rbd image that is 100 GB:

```
rbd -p mypool create myimage --size 102400
```

or alternatively:

```
rbd create mypool/myimage --size 102400
```

To use a non-default object size (8 MB):

```
rbd create mypool/myimage --size 102400 --order 23
```

To delete an rbd image (be careful!):

```
rbd rm mypool/myimage
```

To create a new snapshot:

```
rbd snap create mypool/myimage@mysnap
```

To create a copy-on-write clone of a protected snapshot:

```
rbd clone mypool/myimage@mysnap otherpool/cloneimage
```

To see which clones of a snapshot exist:

```
rbd children mypool/myimage@mysnap
```

To delete a snapshot:

```
rbd snap rm mypool/myimage@mysnap
```

To map an image via the kernel with cephx enabled:

```
rbd map mypool/myimage --id admin --keyfile secretfile
```

To unmap an image:

```
rbd unmap /dev/rbd0
```

To create an image and a clone from it:

```
rbd import --format 2 image mypool/parent
rbd snap create --snap snapname mypool/parent
rbd snap protect mypool/parent@snap
rbd clone mypool/parent@snap otherpool/child
```

To create an image with a smaller stripe_unit (to better distribute small writes in some workloads):

```
rbd -p mypool create myimage --size 102400 --stripe-unit 65536 --stripe-count 16
```

To change an image from one format to another, export it and then import it as the desired format:

```
rbd export mypool/myimage@snap /tmp/img
rbd import --format 2 /tmp/img mypool/myimage2
```

To lock an image for exclusive use:

```
rbd lock add mypool/myimage mylockid
```

To release a lock:

```
rbd lock remove mypool/myimage mylockid client.2485
```

AVAILABILITY

rbd is part of the Ceph distributed file system. Please refer to the Ceph documentation at http://ceph.com/docs for more information.

SEE ALSO

ceph(8), rados(8)