RBD – MANAGE RADOS BLOCK DEVICE (RBD) IMAGES

SYNOPSIS

rbd [-c *ceph.conf*] [-m *monaddr*] [–cluster *cluster-name*] [-p | –pool *pool*] [*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).

--cluster `cluster-name`
Use different cluster name as compared to default cluster name *ceph*.

-p `pool-name`, **--pool** `pool-name`
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-id`
Specifies which object layout to use. The default is 2.

- format 1 - (deprecated) 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 and kernel since version 3.11 (except for striping). This adds support for cloning and is more easily extensible to allow more features in the future.

-s `size-in-M/G/T`, **--size** `size-in-M/G/T`
Specifies the size of the new rbd image or the new size of the existing rbd image in M/G/T. If no suffix is given, unit M is assumed.

--object-size `size-in-B/K/M`
Specifies the object size in B/K/M. Object size will be rounded up the nearest power of two; if no suffix is given, unit B is assumed. The default object size is 4M, smallest is 4K and maximum is 32M.

--stripe-unit `size-in-B/K/M`
Specifies the stripe unit size in B/K/M. If no suffix is given, unit B is assumed. 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.

--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.

--keyfile `filename`
Specifies a file containing the secret key of `--id user` to use with the map command. This option is overridden by `--keyring` if the latter is also specified.

--shared `lock-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.

-o `krbd-options`, **--options** `krbd-options`
Specifies which options to use when mapping or unmapping an image via the rbd kernel driver. krbd-options is a comma-separated list of options (similar to mount(8) mount options). See kernel rbd (krbd) options section below for more details.

--read-only
Map the image read-only. Equivalent to -o ro.

--image-feature `feature-name`
Specifies which RBD format 2 feature should be enabled when creating an image. Multiple features can be enabled by repeating this option multiple times. The following features are supported:

- layering: layering support
- striping: striping v2 support
- exclusive-lock: exclusive locking support
- object-map: object map support (requires exclusive-lock)
- fast-diff: fast diff calculations (requires object-map)
- deep-flatten: snapshot flatten support
- journaling: journaled IO support (requires exclusive-lock)
- data-pool: erasure coded pool support

--image-shared
Specifies that the image will be used concurrently by multiple clients. This will disable features that are dependent upon exclusive ownership of the image.

--whole-object
Specifies that the diff should be limited to the extents of a full object instead of showing intra-object deltas. When the object map feature is enabled on an image, limiting the diff to the object extents will dramatically improve performance since the differences can be computed by examining the in-memory object map instead of querying RADOS for each object within the image.

--limit
Specifies the limit for the number of snapshots permitted.

COMMANDS

bench –io-type <read | write | readwrite | rw> [–io-size *size-in-B/K/M/G/T*] [–io-threads *num-ios-in-flight*] [–io-total *size-in-B/K/M/G/T*] [–io-pattern seq | rand] [–rw-mix-read *read proportion in readwrite*] *image-spec*
Generate a series of IOs to the image and measure the IO throughput and latency. If no suffix is given, unit B is assumed for both –io-size and –io-total. Defaults are: –io-size 4096, –io-threads 16, –io-total 1G, –io-pattern seq, –rw-mix-read 50.

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

This requires image format 2.

clone [–object-size *size-in-B/K/M*] [–stripe-unit *size-in-B/K/M* –stripe-count *num*] [–image-feature *feature-name*] [–image-shared] *parent-snap-spec child-image-spec*

Will create a clone (copy-on-write child) of the parent snapshot. Object size will be identical to that of the parent image unless specified. Size will be the same as the parent snapshot. The –stripe-unit and –stripe-count arguments are optional, but must be used together.

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

cp (*src-image-spec* | *src-snap-spec*) *dest-image-spec*

Copy the content of a src-image into the newly created dest-image. dest-image will have the same size, object size, and image format as src-image.

create (-s | –size *size-in-M/G/T*) [–image-format *format-id*] [–object-size *size-in-B/K/M*] [–stripe-unit *size-in-B/K/M* –stripe-count *num*] [–image-feature *feature-name*]… [–image-shared] *image-spec*

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.

deep cp (*src-image-spec* | *src-snap-spec*) *dest-image-spec*

Deep copy the content of a src-image into the newly created dest-image. Dest-image will have the same size, object size, image format, and snapshots as src-image.

device list [-t | –device-type *device-type*] [–format plain | json | xml] –pretty-format

Show the rbd images that are mapped via the rbd kernel module (default) or other supported device.

device map [-t | –device-type *device-type*] [–read-only] [–exclusive] [-o | –options *device-options*] *image-spec* | *snap-spec*

Map the specified image to a block device via the rbd kernel module (default) or other supported device (*nbd* on Linux or *ggate* on FreeBSD).

The –options argument is a comma separated list of device type specific options (opt1,opt2=val,…).

device unmap [-t | –device-type *device-type*] [-o | –options *device-options*] *image-spec* | *snap-spec* | *device-path*

Unmap the block device that was mapped via the rbd kernel module (default) or other supported device.

The –options argument is a comma separated list of device type specific options (opt1,opt2=val,…).

diff [–from-snap *snap-name*] [–whole-object] *image-spec* | *snap-spec*

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.

du [-p | –pool *pool-name*] [*image-spec* | *snap-spec*]

Will calculate the provisioned and actual disk usage of all images and associated snapshots within the specified pool. It can also be used against individual images and snapshots.

If the RBD fast-diff feature is not enabled on images, this operation will require querying the OSDs for every potential object within the image.

export [–export-format *format (1 or 2)*] (*image-spec* | *snap-spec*) [*dest-path*]

Export image to dest path (use - for stdout). The –export-format accepts ‘1’ or ‘2’ currently. Format 2 allow us to export not only the content of image, but also the snapshots and other properties, such as image_order, features.

export-diff [–from-snap *snap-name*] [–whole-object] (*image-spec* | *snap-spec*) *dest-path*

Export 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.

feature disable *image-spec feature-name*…

Disable the specified feature on the specified image. Multiple features can be specified.

feature enable *image-spec feature-name*…

Enable the specified feature on the specified image. Multiple features can be specified.

flatten *image-spec*

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 image format 2.

group create *group-spec*

Create a group.

group image add *group-spec image-spec*
Add an image to a group.

group image list *group-spec*
List images in a group.

group image remove *group-spec image-spec*
Remove an image from a group.

group ls [-p | –pool *pool-name*]
List rbd groups.

group rename *src-group-spec dest-group-spec*
Rename a group. Note: rename across pools is not supported.

group rm *group-spec*
Delete a group.

group snap create *group-snap-spec*
Make a snapshot of a group.

group snap list *group-spec*
List snapshots of a group.

group snap rm *group-snap-spec*
Remove a snapshot from a group.

group snap rename *group-snap-spec snap-name*
Rename group’s snapshot.

image-meta get *image-spec key*
Get metadata value with the key.

image-meta list *image-spec*
Show metadata held on the image. The first column is the key and the second column is the value.

image-meta remove *image-spec key*
Remove metadata key with the value.

image-meta set *image-spec key value*
Set metadata key with the value. They will displayed in *image-meta list*.

import [–export-format *format (1 or 2)*] [–image-format *format-id*] [–object-size *size-in-B/K/M*] [–stripe-unit *size-in-B/K/M* –stripe-count *num*] [–image-feature *feature-name*]… [–image-shared] *src-path* [*image-spec*]
Create 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 (object size).

The –stripe-unit and –stripe-count arguments are optional, but must be used together.

The –export-format accepts ‘1’ or ‘2’ currently. Format 2 allow us to import not only the content of image, but also the snapshots and other properties, such as image_order, features.

import-diff *src-path image-spec*
Import 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.

info *image-spec* | *snap-spec*
Will dump information (such as size and object size) 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.

journal client disconnect *journal-spec*
Flag image journal client as disconnected.

journal export [–verbose] [–no-error] *src-journal-spec path-name*
Export image journal to path (use - for stdout). It can be make a bakcup of the image journal especially before attempting dangerous operations.

Note that this command may not always work if the journal is badly corrupted.

journal import [–verbose] [–no-error] *path-name dest-journal-spec*
Import image journal from path (use - for stdin).

journal info *journal-spec*

Show information about image journal.

journal inspect [–verbose] *journal-spec*
Inspect and report image journal for structural errors.

journal reset *journal-spec*
Reset image journal.

journal status *journal-spec*
Show status of image journal.

lock add [–shared *lock-tag*] *image-spec 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 ls *image-spec*
Show locks held on the image. The first column is the locker to use with the *lock remove* command.

lock rm *image-spec lock-id locker*
Release a lock on an image. The lock id and locker are as output by lock ls.

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.

merge-diff *first-diff-path second-diff-path merged-diff-path*
Merge two continuous incremental diffs of an image into one single diff. The first diff's end snapshot must be equal with the second diff's start snapshot. The first diff could be - for stdin, and merged diff could be - for stdout, which enables multiple diff files to be merged using something like 'rbd merge-diff first second - | rbd merge-diff - third result'. Note this command currently only support the source incremental diff with stripe_count == 1

mirror image demote *image-spec*
Demote a primary image to non-primary for RBD mirroring.

mirror image disable [–force] *image-spec*
Disable RBD mirroring for an image. If the mirroring is configured in `image` mode for the image's pool, then it can be explicitly disabled mirroring for each image within the pool.

mirror image enable *image-spec*
Enable RBD mirroring for an image. If the mirroring is configured in `image` mode for the image's pool, then it can be explicitly enabled mirroring for each image within the pool.

This requires the RBD journaling feature is enabled.

mirror image promote [–force] *image-spec*
Promote a non-primary image to primary for RBD mirroring.

mirror image resync *image-spec*
Force resync to primary image for RBD mirroring.

mirror image status *image-spec*
Show RBD mirroring status for an image.

mirror pool demote [*pool-name*]
Demote all primary images within a pool to non-primary. Every mirroring enabled image will demoted in the pool.

mirror pool disable [*pool-name*]
Disable RBD mirroring by default within a pool. When mirroring is disabled on a pool in this way, mirroring will also be disabled on any images (within the pool) for which mirroring was enabled explicitly.

mirror pool enable [*pool-name*] *mode*
Enable RBD mirroring by default within a pool. The mirroring mode can either be `pool` or `image`. If configured in `pool` mode, all images in the pool with the journaling feature enabled are mirrored. If configured in `image` mode, mirroring needs to be explicitly enabled (by `mirror image enable` command) on each image.

mirror pool info [*pool-name*]
Show information about the pool mirroring configuration. It includes mirroring mode, peer UUID, remote cluster name, and remote client name.

mirror pool peer add [*pool-name*] *remote-cluster-spec*
Add a mirroring peer to a pool. *remote-cluster-spec* is [*remote client name*@]*remote cluster name*.

The default for *remote client name* is "client.admin".

This requires mirroring mode is enabled.

mirror pool peer remove [*pool-name*] *uuid*
Remove a mirroring peer from a pool. The peer uuid is available from `mirror pool info` command.

mirror pool peer set [*pool-name*] *uuid key value*
Update mirroring peer settings. The key can be either `client` or `cluster`, and the value is corresponding to remote client name or remote cluster name.

mirror pool promote [–force] [*pool-name*]
Promote all non-primary images within a pool to primary. Every mirroring enabled image will promoted in the pool.

mirror pool status [–verbose] [*pool-name*]
Show status for all mirrored images in the pool. With –verbose, also show additionally output status details for every mirroring image in the pool.

mv *src-image-spec dest-image-spec*
Rename an image. Note: rename across pools is not supported.

object-map check *image-spec* | *snap-spec*
Verify the object map is correct.

object-map rebuild *image-spec* | *snap-spec*
Rebuild an invalid object map for the specified image. An image snapshot can be specified to rebuild an invalid object map for a snapshot.

pool init [*pool-name*] [–force]
Initialize pool for use by RBD. Newly created pools must initialized prior to use.

resize (-s | –size *size-in-M/G/T*) [–allow-shrink] *image-spec*
Resize rbd image. The size parameter also needs to be specified. The –allow-shrink option lets the size be reduced.

rm *image-spec*
Delete an rbd image (including all data blocks). If the image has snapshots, this fails and nothing is deleted.

snap create *snap-spec*
Create a new snapshot. Requires the snapshot name parameter specified.

snap limit clear *image-spec*
Remove any previously set limit on the number of snapshots allowed on an image.

snap limit set [–limit] *limit image-spec*
Set a limit for the number of snapshots allowed on an image.

snap ls *image-spec*
Dump the list of snapshots inside a specific image.

snap protect *snap-spec*
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 image format 2.

snap purge *image-spec*
Remove all unprotected snapshots from an image.

snap rename *src-snap-spec dest-snap-spec*
Rename a snapshot. Note: rename across pools and images is not supported.

snap rm [–force] *snap-spec*
Remove the specified snapshot.

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

snap unprotect *snap-spec*
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 image format 2.

status *image-spec*
Show the status of the image, including which clients have it open.

trash ls [*pool-name*]
List all entries from trash.

trash mv *image-spec*
Move an image to the trash. Images, even ones actively in-use by clones, can be moved to the trash and deleted at a later time.

trash purge [*pool-name*]
Remove all expired images from trash.

trash restore *image-id*
Restore an image from trash.

trash rm *image-id*
Delete an image from trash. If image deferment time has not expired you can not removed it unless use force. But an actively in-use by clones or has snapshots can not be removed.

watch *image-spec*
Watch events on image.

IMAGE, SNAP, GROUP AND JOURNAL SPECS

image-spec is [*pool-name*/]*image-name*
snap-spec is [*pool-name*/]*image-name*@*snap-name*
group-spec is [*pool-name*/]*group-name*
group-snap-spec is [*pool-name*/]*group-name*@*snap-name*
journal-spec is [*pool-name*/]*journal-name*

The default for *pool-name* is “rbd”. If an image name contains a slash character (‘/’), *pool-name* is required.

The *journal-name* is *image-id*.

You may specify each name individually, using –pool, –image and –snap options, but this is discouraged in favor of the above spec syntax.

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:

`object-size`
The size of objects we stripe over is a power of two. It will be rounded up the nearest power of two. The default object size is 4 MB, smallest is 4K and maximum is 32M.

`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. 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.

KERNEL RBD (KRBD) OPTIONS

Most of these options are useful mainly for debugging and benchmarking. The default values are set in the kernel and may therefore depend on the version of the running kernel.

Per client instance *rbd device map* options:

- fsid=aaaaaaaa-bbbb-cccc-dddd-eeeeeeeeeeee - FSID that should be assumed by the client.
- ip=a.b.c.d[:p] - IP and, optionally, port the client should use.
- share - Enable sharing of client instances with other mappings (default).

- noshare - Disable sharing of client instances with other mappings.
- crc - Enable CRC32C checksumming for data writes (default).
- nocrc - Disable CRC32C checksumming for data writes.
- cephx_require_signatures - Require cephx message signing (since 3.19, default).
- nocephx_require_signatures - Don't require cephx message signing (since 3.19).
- tcp_nodelay - Disable Nagle's algorithm on client sockets (since 4.0, default).
- notcp_nodelay - Enable Nagle's algorithm on client sockets (since 4.0).
- cephx_sign_messages - Enable message signing (since 4.4, default).
- nocephx_sign_messages - Disable message signing (since 4.4).
- mount_timeout=x - A timeout on various steps in *rbd device map* and *rbd device unmap* sequences (default is 60 seconds). In particular, since 4.2 this can be used to ensure that *rbd device unmap* eventually times out when there is no network connection to a cluster.
- osdkeepalive=x - OSD keepalive timeout (default is 5 seconds).
- osd_idle_ttl=x - OSD idle TTL (default is 60 seconds).

Per mapping (block device) *rbd device map* options:

- rw - Map the image read-write (default).
- ro - Map the image read-only. Equivalent to –read-only.
- queue_depth=x - queue depth (since 4.2, default is 128 requests).
- lock_on_read - Acquire exclusive lock on reads, in addition to writes and discards (since 4.9).
- exclusive - Disable automatic exclusive lock transitions (since 4.12).

rbd device unmap options:

- force - Force the unmapping of a block device that is open (since 4.9). The driver will wait for running requests to complete and then unmap; requests sent to the driver after initiating the unmap will be failed.

EXAMPLES

To create a new rbd image that is 100 GB:

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

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

```
rbd create mypool/myimage --size 102400 --object-size 8M
```

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

To map an image via the kernel with different cluster name other than default *ceph*:

```
rbd device map mypool/myimage --cluster cluster-name
```

To unmap an image:

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

To create an image and a clone from it:

```
rbd import --image-format 2 image mypool/parent
rbd snap create mypool/parent@snap
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 create mypool/myimage --size 102400 --stripe-unit 65536B --stripe-count 16
```

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

```
rbd export mypool/myimage@snap /tmp/img
rbd import --image-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
```

To list images from trash:

```
rbd trash ls mypool
```

To defer delete an image (use *–expires-at* to set expiration time, default is now):

```
rbd trash mv mypool/myimage --expires-at "tomorrow"
```

To delete an image from trash (be careful!):

```
rbd trash rm mypool/myimage-id
```

To force delete an image from trash (be careful!):

```
rbd trash rm mypool/myimage-id  --force
```

To restore an image from trash:

```
rbd trash restore mypool/myimage-id
```

To restore an image from trash and rename it:

```
rbd trash restore mypool/myimage-id --image mynewimage
```

AVAILABILITY

rbd is part of Ceph, a massively scalable, open-source, distributed storage system. Please refer to the Ceph documentation at http://ceph.com/docs for more information.

SEE ALSO

ceph(8), rados(8)