Mapping RBD Images Persistently

The `rbdmap` service can automatically map and unmap RBD images to devices when booting and shutting down the system. This service looks for the mapped images with their credentials in the `/etc/ceph/rbdmap` file. The service mounts and unmounts the RBD images using their mount points as they appear in the `/etc/fstab` file.

The following steps configure `rbdmap` to persistently map and unmap an RBD image that already contains a file system:

1. Create the mount point for the file system.

2. Create a single-line entry in the `/etc/ceph/rbdmap` RBD map file. This entry must specify the name of the RBD pool and image. It must also reference the Cephx user who has read/write permissions to access the image and the corresponding key-ring file. Ensure that the key-ring file for the Cephx user exists on the client system.

3. Create an entry for the RBD in the `/etc/fstab` file on the client system. The name of the block device has the following form:

   ```
   /dev/rbd/pool_name/image_name
   ```

   Specify the `noauto` mount option, because the `rbdmap` service, not the Linux fstab routines, handles the mounting of the file system.

4. Confirm that the block device mapping works. Use the `rbdmap map` command to mount the devices. Use the `rbdmap unmap` command to unmount them.

5. Enable the `rbdmap systemd` service.

Refer to `rbdmap`(8) for more information.

Accessing Ceph Storage with librbd-based Clients

The `librbd` library provides direct access to RBD images for user space applications. It inherits the capabilities of `librados` to map data blocks to objects in the Ceph Object Store, and implements the ability to access RBD images and create snapshots and clones.

Cloud and virtualization solutions, such as OpenStack and `libvirt`, use `librbd` to provide RBD images as block devices to cloud instances and the virtual machines that they manage. For example, RBD images can store QEMU virtual machine images. Using the RBD clone feature, virtualized containers can boot a virtual machine without copying the boot image. The copy-on-write (COW) mechanism copies data from the parent to the clone when it writes to an unallocated object within the clone. The copy-on-read (COR) mechanism copies data from the parent to the clone when it reads from an unallocated object within the clone.