Alpenhorn

Preliminaries

1. Start the DB and enter the first container

   ```
   docker-compose up -d db
   docker-compose run host_1 bash -l
   ```

2. Initialize Alpenhorn database

   ```
   alpenhorn init
   alpenhorn create_group group_1
   alpenhorn create_node --auto_import=yes node_1 /data host_1 group_1
   alpenhorn mount --user root --address host_1 node_1
   ```

Importing files

3. Start the service

   ```
   docker-compose up host_1
   ```

Note files being imported. (Could also do `docker-compose logs host_1 | head -30`.)

4. Client

   ```
   docker-compose exec host_1 bash -l
   alpenhorn status
   ```

Two files on host_1

Start more nodes

5. Create storage

   ```
   alpenhorn create_group group_2
   alpenhorn create_node --auto_import=yes node_2 /data host_2 group_2
   alpenhorn mount --user root --address host_2 --hostname host_2 node_2

   alpenhorn create_group group_3
   alpenhorn create_node --auto_import=yes node_3 /data host_3 group_3
   alpenhorn mount --user root --address host_3 --hostname host_3 node_3
   ```

6. Start the service

   ```
   docker-compose up -d host_1
   docker-compose up host_2
   docker-compose up host_3
   ```

Note no files are being imported.

Syncing files

7. Client

   ```
   docker-compose exec host_2 bash -l
   alpenhorn verify
   find /data
   ```

No files on host_2

8. Start a sync

   ```
   alpenhorn sync node_1 group_2 --show_files
   ```

   Note the server copying

   ```
   find /data
   ```

   The files are here!

9. Sync an explicit acquisition

   ```
   docker-compose exec host_3 bash -l
   find /data
   alpenhorn sync node_1 group_3 --acq=12345678T000000Z_inst_zab --show_files
   ```

   Note the server copying

10. Client

    ```
    alpenhorn status
    alpenhorn verify node_2
    alpenhorn verify node_3
    ```

Two files on host_2, one file on host_3.

Add a new file

11. create a new file

    ```
    docker-compose exec host_2 bash -l
    echo foo bar > /data/12345678T000000Z_inst_zab/jim.out
    ```

12. see the new file get synced

    ```
    alpenhorn sync --acq 12345678T000000Z_inst_zab node_2 group_3 --show_files
    alpenhorn status
    docker-compose up host_3
    ```

    Highlight "transferring file … jim.out"

13. See how many files are present

    ```
    alpenhorn status
    ```

Dealing with corruption

14. Modify a copy of a file:

    ```
    alpenhorn status
    echo bla >> /data/12345678T000000Z_inst_zab/jim.out
    alpenhorn status
    ```

    Notice in the output of the service:

    ```
    > INFO >> Checking file "/data/12345678T000000Z_inst_zab/jim.out" on node "node_2".
    > ERROR >> File is corrupted!
    > INFO >> Updating file copy status [id=6].
    ```

15. Repair with a sync from a known good copy:

    ```
    alpenhorn sync --acq 12345678T000000Z_inst_zab node_1 group_2 --show_files
    alpenhorn status
    ```

Cleaning

16. Remove unneeded files with `clean`:

    ```
    alpenhorn clean --now node_2
    ```

    Observe only two are removed: “Too few backups to delete 2017/03/21/acq_xy1_45678901T000000Z_inst_zab/acq_data/x_123_1_data/raw/acq_123_1.zxc”. This is because this acquisition was never synced to node_3

Transport disks

17. Create storage

    ```
    alpenhorn create_group transport
    alpenhorn create_node --storage_type=T t1 /mnt/t1 host_1 transport
    ```

18. Make it available

    ```
    alpenhorn mount --address=host_1 t1
    ```

19. Copy to transport for a target destination:

    ```
    alpenhorn sync node_1 transport --target group_3 --show_files
    ```

    Note files being synced

```
alpenhorn status
find /mnt/t1
alpenhorn unmount t1
```

20. On the other side (host_3)…

    ```
    alpenhorn mount --address=host_3 t1
    alpenhorn sync t1 group_3 --show_files
    ```

    Note files being synced

    ```
    alpenhorn status
    alpenhorn clean --now t1
    alpenhorn status
    find /mnt/t1
    ```