NASMan Documentation

Release 0.1.0.dev0

Ben Cole

April 28, 2015

Contents

Django ZFS NAS management system. Primarily for browsing snapshots for file restoration

CHAPTER 1

Contents

1.1 Installation

NASMan is a Docker application. To install it you will need to have docker and docker-compose installed on your host system

NASMan requires access to ZFS in order to be useful. The main docker container runs in privileged mode. This allows the container and the app to access the ZFS filesystems on the host

1.1.1 Prerequisites

- Docker
- Docker compose
- ZFS (for now, will make this optional later)

1.1.2 Installation

Clone the project, then from inside the project directory

```
$ docker-compose up -d
$ docker exec nasman_web_1 python manage.py migrate
$ docker exec nasman_web_1 python manage.py sitetree_resync_apps
```

Your installation will now be running and listening on port 8000 of the host.

At the moment it is using the Django runserver, which is discouraged from being used in production for performance issues, however due to the use case of NASMan, it seems unlikely to be an issue. Feel free to raise an issue on github of you encounter any problems.

1.2 Developer Documentation

This covers code style, planned features, and architectural decisions for reference

1.2.1 Code Style

- PEP8 always

1.2.2 File recovery feature

The primary function of this project, is to allow convenient recovery of files from ZFS snapshots.

- Search for files across "live" filesystem(s) and snapshots
  - Done for live filesystem
- Browse filesystems and snapshot
  - Done for live filesystem
- Recover files from snapshots to live filesystem

Todo

- Indexing snapshots

Indexing snapshots

- NASMan will eventually handle snapshot creation and rotation. A call to a view will create a snapshot, optionally recursively and rotate old ones.
- When a snapshot is created, we fire a task to index it. Likewise when one is deleted, remove it from the index.
- Indexing doesn't have to do anything fancy like hash files.

1.2.3 Planned Features

As the primary use case is for ZFS based storage servers there are many other useful possibilities that could be developed later:

- Disk space usage analyser - find large files/folders
- Duplicate Finder
- System health dashboard
- Backup to multiple locations

1.2.4 Duplicate Finder

A good way to free up space is to find duplicate files. There are numerous ways to do this; hashing, filesize comparison, filename comparison etc.

I intend to do something a little unique. My use case is remove duplicate HD video files (left over from transcoding etc) These are likely to have different filesizes and hashes (as the content will differ). But they should have at least similar metadata.

The plan therefore is to do the following:

- Extract metadata from files whilst indexing them
- Store the metadata in Xapian faceted
- Use facets to determine duplicates

1.2.5 System health dashboard

As ZFS storage servers tend to be tucked away they're often forgotten about until something breaks.

There aren't many (any?) good, simple monitoring tools "that just work" out the box

- Monitor hardware - temp, disk health etc
- Monitor pool(s) - disk space, faulted disks
- Notifications

1.2.6 Backup to multiple locations

Using zfs send and receive commands pools can be easily backed up whilst online. There are other tools that do this, but as NASMan will manage snapshots it makes sense for it to manage backups.

Possible backup options include:

- Mirroring pool to local or remote pool
- Incremental backup
- A customisable history of snapshots
- Backups piped to xz/tar for compression
- Backups piped to GnuPG or OpenSSL for encryption
- Backup to a file to be stored on non-zfs filesystem
  - Cloud, DVD, Bluray, dumb hard drive, etc.
- Combinations of the above

Perhaps even having access to cloud providers within the app to do this automatically.

CHAPTER 2

Indices and tables

- *genindex*
- *modindex*
- *search*