OSUOSL Mirror Sync Documentation

Release 1.0.0

Lance Albertson

March 17, 2014

CHAPTER 1

Adding a new Project

1.1 Synopsis

Adds a new project mirror to our ftp infrastructure

1.2 Intro/Background/Info

Mirroring projects is fun and easy

1.3 Detailed Procedure

1.3.1 First, choose which array to put the project:

- We have two arrays: /data/ftp/.1 and /data/ftp/.2
- Make sure there is enough disk space for your project
- If you expect the project to use a lot of bandwidth, put it on the less used array (look at iostat)

In cfengine

1.3.2 For local master (account on ftp-osl to upload files):

- add trigger permissions in cf.mirror:

  `/data/trigger/set/$project mode=2775 owner=$project group=trigger`

- add `files/data/mirror/bin/update-master/$project`:

```
#!/bin/bash

echo "We are master"
```

1.3.3 For remote mirrors (update via rsync):

- add project cronjob to `files/etc/cron.d/mirror/update.master`
- add rsync line to `files/data/mirror/bin/update-master/$project`:

```
#!/bin/bash

rsync -avH --delete projecthost::module/ /data/ftp/.$array/$project/
```

1.3.4 For all mirrors:

- add symlink in cf.mirror:

  ```
  /data/ftp/pub/$project -> /data/ftp/.1/$project
  ```

- if wanted, add vhost config: `files/etc/apache2/vhosts.d/mirror/$project.osuosl.org.conf`
- commit

On ftp-osl

1.3.5 For local master:

```
useradd -m $project
mkdir /data/ftp/.$array/$project
chown $project:$project /data/ftp/.$array/$project
ln -sf /data/ftp/pub/$project /home/$project/data
```

- put ssh keys in /home/$project/.ssh/authorized_keys
- add trigger script /home/$project/trigger-$project

```
#!/bin/bash

/data/mirror/bin/trigger-set $project
```

- add `/home/$project/README`

```
Your data is in /data/ftp/.$array/$project/
The symlink "data" links there for your convenience.

To signal the server to push your data to the other servers, run the
trigger-$project script.
```

1.3.6 For remote mirror:

- after committing/pushing changes in git, run `cfexecd -F -q`
- run the update script `/data/mirror/bin/run-update <project> --email` - this will sync the project, and trigger the slaves to sync with the master

On ftp-{osl,nyc,chi}

- mkdir `/data/ftp/.$array/$project`
- run `cfexecd -F -q`
- reload apache, `/etc/init.d/apache2 reload`

Email message

You should now be able to ssh to ftp-osl.osuosl.org with the username $project and the key that you provided. See the README file in your home directory on that server for instructions on how to upload files.

CHAPTER 2

FTP Emergency Docs

This doc will describe a variety of potential emergency situations we may run into with the ftp cluster. Its mostly to be used as a potential guide in case they happen.

2.1 FTP Array Failure

In the eventual but hopefully unlikely event that an array of disks completely fails due to too many disks failing, this might be what to do. This is much easier to work around if the machine is one of the slaves (nyc/chi), however in the case of ftp-osl going down it gets even more complicated.

2.1.1 ftp-osl failure

Some things to consider:

- This machine acts as master mirror for some projects
- All slaves sync from this host
- While we rebuild ftp-osl, all new syncs will likely be stopped to ensure data integrity

With this in mind, we have to consider a few list of actions. The TL;DR version of what we should probably do is:

- Stop all cronjobs on all ftp hosts to mitigate any potential data loss (simply stopping cron is probably ok)
- Take ftp-osl out of DNS rotation
- Shutdown http/xinetd. We might want to restrict ssh access too
- Notify the outage via hosting list and other methods (may even need a news post on our website)
- Ensure the integrity of the data on the other two slave machines for the same partition (basically make sure each repo is the same size or close to it)
- Designate which slave to use to re-copy data back to ftp-osl, likely chi but nyc might be faster
- Rebuild the disk array and mkfs.xfs it, etc. It's probably easier to just recreate the array at that point
- Start an rsync from slave node (this will likely take DAYS depending on the load and such).
- Ensure everything looks sane
- Re-enable cronjobs and do some tests
- Add back into rotation
- Send out another announcement

CHAPTER 3

Moving projects to a different array

Copy the files to the new array on the slave servers

```
rsync -av /data/ftp/.$old_array/$project/ /data/ftp/.$new_array/$project/
ln -sf ../.$new_array/$project /data/ftp/pub/$project
rm -rf /data/ftp/.$old_array/$project/
```

Copy the files to the new array on ftp-osl

```
rsync -av /data/ftp/.$old_array/$project/ /data/ftp/.$new_array/$project/
ln -sf ../.$new_array/$project /data/ftp/pub/$project
rm -rf /data/ftp/.$old_array/$project/
```

Change update-master sync script to sync to the new location, and update symlinks in cfengine

```
vi infra/files/data/mirror/bin/update-master/$project
vi infra/cfengine/inputs/cf.mirror
```

CHAPTER 4

FTP Scripts

4.1 Summary

This page is part of the *FTP Infrastructure* documentation. Please visit that page for an overview.

There are five categories of scripts:

- *Sync Data from Upstream*
- *Sync Between Master and Slave*
- *Control Downloads*
- *Gather Statistics*
- *ftpsync*

Each is explained here.

4.2 Types of scripts

4.2.1 Sync Data from Upstream

- `/data/mirror/bin/run-update`
  - Wrapper for scripts to sync and lock.
  - If you need to force an update to a specific project, please do the following on ftp-osl:

```
$ /data/mirror/bin/run-update <project name> --email
```

- `/data/mirror/bin/update-master/*`
  - Run by `run-update` to actually sync the data for each tree.
  - Should not be run manually (use `run-update` instead).

4.2.2 Sync Between Master and Slave

This system of triggering runs on a cron job once per minute, so in theory changes to ftp-osl are propagated very quickly out to the other ftp servers.

- `/data/mirror/bin/run-update` - Same use as for pulling from upstream, but pulls from master.

- `/data/mirror/bin/trigger-set`
  - Puts a trigger file in `/data/trigger/set/tree_name/` on master.
  - The trigger indicates that the master is up-to-date and the slaves can begin pulling data.
- `/data/mirror/bin/trigger-run`
  - Used by the slave servers to check for new triggers set on the master.
  - If it tries to remove a trigger that has already been removed, will send an error e-mail. It is safe to ignore this e-mail unless a deluge of errors occurs. Network latency is usually the cause.
- `/data/mirror/bin/trigger-scan`
  - Copies set triggers from master to slaves.
- `/usr/local/sbin/mkchroot`
  - Creates and sets up chroot used by trigger system.

4.2.3 Control Downloads

- `/usr/local/sbin/i2-check`
  - Check if a particular IP address is on the I2 list.
- `/usr/local/sbin/i2-check.pl`
  - A much faster version of the bash script.
- `/usr/local/sbin/i2-fill`
  - Update the mysql database of IP addresses from the I2 list.
- `/usr/local/sbin/ip*`
  - Manage temporary iptables blacklist.
  - Accepts any number of IPs/netblocks as arguments.
  - Only affects local box.
  - Won't affect normal firewall; uses mangle table.

4.2.4 Gather Statistics

- `/data/mirror/bin/check-size`
  - Goes through trees to find size of each.
  - Info stored in a log file, processed by fir.
- `/data/mirror/bin/gen-header`
  - Run manually to update fancy header used on http indexes.
  - Since the output files are in cfengine, need to update those files at the same time.
- `/data/mirror/bin/gen-footer`
  - Run every minute to update bandwidth bar.
- `/usr/local/sbin/check-apt-mirror`
  - Runs md5sum on every package in specified ubuntu or debian tree.

  - Takes about half a day to run.
- `/usr/local/sbin/logs-*` *Deprecated??*
  - Manage apache and ftp logs instead of using logrotate
  - Run by cron jobs.

4.2.5 ftpsync

- `/data/mirror/ftpsync/bin/ftpsync`
  - syncs files with a 2-stage sync to avoid breaking
- `/data/mirror/ftpsync/bin/runmirrors`
  - triggers slaves
- `/data/mirror/etc/ftpsync-$project.conf`
  - config file for ftpsync to sync `$project`
- `/data/mirror/etc/runmirrors-$project.conf`
  - config file for runmirrors to push `$project`
- `/data/mirror/etc/$project.mirror`
  - config file listing slaves to push
- see http://www.debian.org/mirror/push_mirroring for more info

CHAPTER 5

Summary

The FTP mirrors are fairly complex, especially when it comes to the way data is kept in sync and monitored. There are many scripts that are scattered about that makes the magic happen. Hopefully this documentation reveals the truth behind the magic.

CHAPTER 6

Introduction

The nature of our system allows some flexibility to the services that we can offer. We can serve as the master mirror for small projects, but also as a part of a large project's mirror infrastructure. Each set of data is referred to as a 'tree'; each project may have one or more trees.

Note: This project has been tightly integrated into the OSL mirror infrastructure and has a lot of hard coded references to hosts, paths and other bits.

CHAPTER 7

Installation

As it stands now, the system is currently deployed using the OSL internal CFEngine configuration management system. Documentation will be updated to reflect how it should be installed externally.

Most of the files reside in `/data/mirror` on the nodes while some scripts are located in `/usr/local/sbin`. In addition there are other services such as xinetd, bwbar, etc that need to be configured outside these directories or the current repository.

CHAPTER 8

Hardware

Currently there are three servers in the system:

- ftp-osl
  - Located in Corvallis, OR
  - Master mirror
  - HP Proliant DL385 G5 server
  - 8G RAM
  - Two data arrays
    - 3T (RAID 6 + spare) msa70 25x 146G 2.5” SAS
    - 3T (RAID 6 + spare) msa70 25x 146G 2.5” SAS
- ftp-chi
  - Located in Chicago, IL, courtesy of TDS
  - Slave mirror
  - HP Proliant DL385 G5 server
  - 8G RAM
  - Two data arrays
    - 3T (RAID 6 + spare) msa70 25x 146G 2.5” SAS
    - 3T (RAID 6 + spare) msa70 25x 146G 2.5” SAS
- ftp-nyc
  - Located in New York, NY, courtesy of TDS
  - Slave mirror
  - HP Proliant DL385 G5 server
  - 8G RAM
    - 3T (RAID 6 + spare) msa70 25x 146G 2.5” SAS
    - 3T (RAID 6 + spare) msa70 25x 146G 2.5” SAS

CHAPTER 9

Software

The scripts used to manage the servers fall into several categories:

- Sync from upstream.
- Sync from the master mirror to the slaves.
- Control distribution of downloads.
- Gather statistics on the status of the data.

CHAPTER 10

External Links

Note: Some of these links are restricted to OSL only networks currently.

- Mirror Sync Status: The sync status of every tree on each mirror.
- Mirror Tree Sizes: The size of every tree. Used to monitor changes in array usage.
- FTP Map: A graphical overview of the bandwidth usage of each server.
- Awstats: Some projects have stats compiled for downloads of their software hosted by us.

CHAPTER 11

Indices and tables

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