Version Control and Backup

Version Control

Version Control is performed using git with a gitolite installation on the Salt Master on its own separate EBS partition at /mnt/snapshot/. Standard ssh-based access controls are used. This was done to make snapshots smaller and more reliable.

For access to a repository, please supply the CommIT technical team with your preferred SSH key. This public key will be imported to gitolite by placing the key in `/home/git/.gitolite/keydir` on the Salt Master(54.244.127.183).

```
sudo su
cd /home/git/.gitolite/keydir/
cp /home/ndk/newkey.pub ./ndk.pub
chown git:git ndk.pub
chmod 400 ndk.pub
su - git
./gitolite compile; ./gitolite trigger POST_COMPILE
logout
logout
```

On the client machine, you may need to add the following to `~/.ssh/config`:

```
Host 54.244.127.183
        User git
        IdentityFile /home/ndk/.ssh/id_dsa.pub
```

Finally, verify access works and list the repositories you have access to:

```
ssh git@54.244.127.183
```

And then get working:

```
git clone git@54.244.127.183:db-name
```

I built it without a gitolite-admin repo because it was much more straightforward to me. We can do gitolite's own configuration versioning in commit-config.

Three things are version controlled and each has its own repo to keep permissions separated:

1) Configuration files (`commit-config`)
2) CommIT CPR customization (`commit-cpr`)
3) CommIT IdP/uApprove GUI customization (`commit-web-shib`)

```
phlogios:~ ndk$ ssh git@saltmaster
X11 forwarding request failed on channel 0
PTY allocation request failed on channel 0
hello ndk, this is git@ip-10-238-42-29 running gitolite3 v3.6-16-g4fefd3f on git 1.7.9.5

 R W        commit-config
 R W        commit-cpr
 R W        commit-web-shib
Connection to saltmaster closed.
```

Repo configuration is done in /home/git/.gitolite/conf/gitolite.conf. This configuration itself is version-controlled within the commit-config

To add a new user, put their certificate in the `keydir` and recompile gitolite:

```
cp /home/ndk/.ssh/authorized_keys /home/git/.gitolite/keydir/ndk.pub
chown -R git:git /home/git/.gitolite/keydir/
su git
cd ~; ./gitolite compile; ./gitolite trigger POST_COMPILE
```

Backup

Most servers and resources within the CommIT deployment architecture are effectively expendable. If a working node fails, replacing it with a new working node using Salt is the first preference and default course of action.

Four pieces of the architecture do require comprehensive backup, though, because they do manage important data:

1) Salt Master
2) gitolite repo (on the Salt Master atm)
3) rsyslogd server's received logs
4) PostgreSQL database

Internet2 built a custom installer for our licensed backup solution, CrashPlan PROe, for us. This does live backup as a daemon running as `backupuser` in our environment with the following default paths and invocation.

Important directories:
Installation:
/home/backupuser/crashplan
Logs:
/home/backupuser/crashplan/log
Default archive location:
/home/backupuser/manifest

Start Scripts:
/home/backupuser/crashplan/bin/CrashPlanEngine start|stop
/home/backupuser/crashplan/bin/CrashPlanDesktop

Periodic snapshots are taken of instances and of the database by using pg_dump, and these snapshots are then archived in S3 or CrashPlan. Running backups are kept by CrashPlan.

CrashPlan does not chase symlinks.

PostgreSQL backups

In order to ensure that no user data is lost and because databases are different animals, separate strategies are used to backup the PostgreSQL database:

1. Add an additional replica.
2. Ensure that cpruser cannot drop and delete tables and that no entity is using the admin user for anything.
3. Use pg_dump to backup the database.

(master on .148 CPR node responsible for backup currently)