The Backup System: Manual & Detail

Content

Deploy

Deploy the backup system:

- Deploy the backup client on servers which have data to backup.
- Deploy the backup server on servers which store backup data.
- One backup client can backup data to multi- backup servers.
- Administrator configures which backup client can connect to which server.

Prerequisite:

- git, cron, rsync, ssh
- gitosis: https://github.com/cee1/gitosis-hack.git
- sendmail or https://raw.github.com/cee1/cee1.archive/master/utilities/mailSender.py (which depends on pyinotify)

Deploy backup server

1. Make sure .../lib/python2.7/dist-packages/gitosis-0.2-py2.7.egg/gitosis/template/admin/hooks/post-update is executable:

```
chmod a+x  python2.7/dist-packages/gitosis-0.2-
py2.7.egg/gitosis/template/admin/hooks/post-update
```

2. cd /path/to/backupsystem/server
3. run setup.sh

```
bash setup.sh URL_of_this_machine [path install to where]
```

4. setup.sh will create(or reuse) a special user “**backupsrv**” with “*path install to where”* as its home directory.
5. setup.sh will also ask a few questions:
   - /path/to/sendmail: use this “sendmail” to notify administrator.
   - The email addresses that will receive notifications.
   - Administrator's public key: used to initialize gitosis control repo.

6. Other adjustment
   - Users of Arch Linux need to edit /etc/cron.d/backupserver (enable the line for Arch).
   - Modify ~**backupsrv**/scripts/backupconfig.sh if needed.

Deploy backup client

1. cd /path/to/backupsystem/client
2. run setup.sh

```
bash setup.sh URL_of_this_machine [path install to where]
```

3. setup.sh will create(or reuse) a special user “**backupclient**” with “*path install to where”* as its home directory.
4. setup.sh will also ask a few questions:
   - /path/to/sendmail: use this “sendmail” to notify administrator.
   - The email addresses that will receive notifications.
   - The URL of default backup server.
5. Other adjustment
   - Users of Arch Linux need to edit /etc/cron.d/backupserver (enable the line for Arch).
   - Modify ~**backupclient**/scripts/backupconfig.sh, specify the paths which need to backup.
   - Add user backupclient to related groups to ensure it can access data which need to backup.

Make a tunnel between backupclient and backupserver

1. Permit backup client to send request to backup server. It is achieved by gitosis:

```
# Administrator's machine
git clone ssh://backupsrv@backupserver_URL/gitosis-admin.git backupserver-admin.git


# Adds backup client's public key and modify gitosis.conf
# commit & push
```

2. On backup client: import the public key of backup server:

```
cat id_rsa.pub >>  ~backupclient/.ssh/authorized_keys
```

3. Backup server & client: remember host fingerprint of each other, i.e. leave a record in “.ssh/known_hosts”.

```
# On server as backupsrv, press 'yes'
ssh backupclient@backupclient_URL


# On  client as backupclient, press 'yes'
ssh backupsrv@backupserver_URL
```

Doing backup

On backupclient:

- ~backupclient/scripts/backup_it
  Backup individual module, usage:

```
backup_it <module> backupserver

# module may be one of “git”, “trac”, “wiki”, “wordpress” and
“directories”

# backupserver may be an empty string which denotes the
default backup server
```

- ~backupclient/scripts/backup_all
  Do a full backup(include all *modules* above).

Browser backup data

On backupserver, ~backupsrv/data/backupclient_URL/

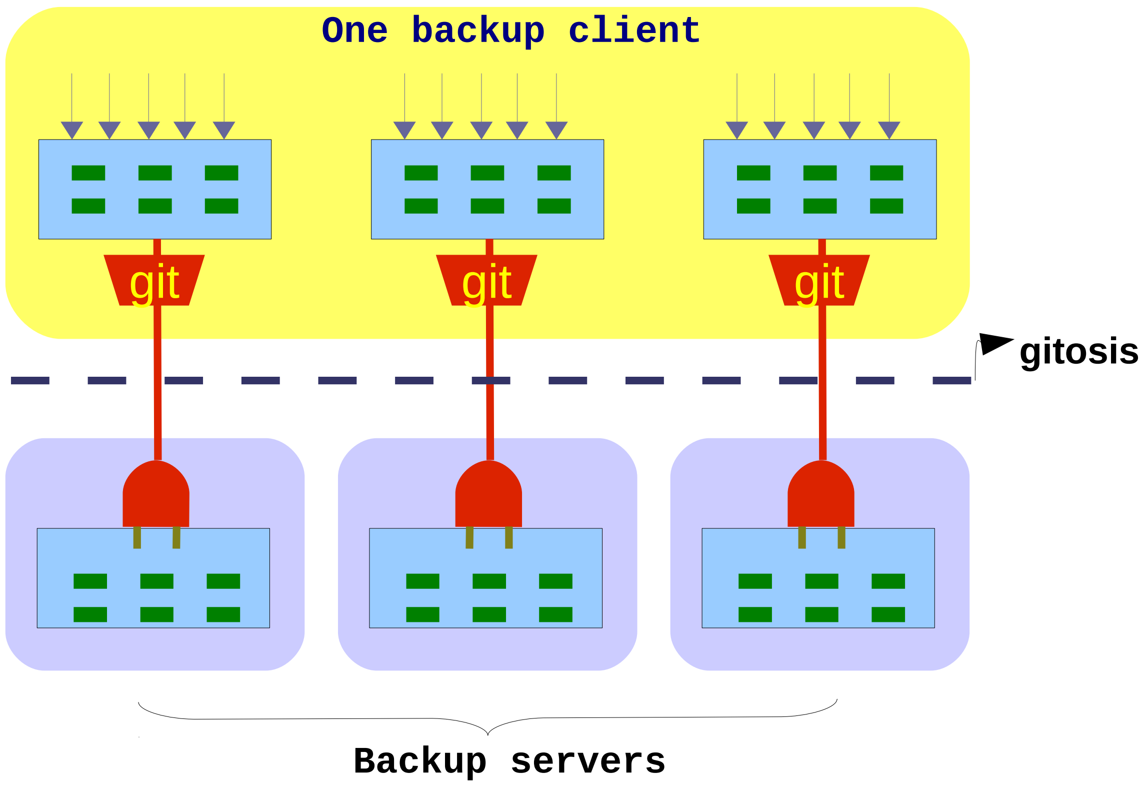
- git:
  backed up git repos.
- rsync:
  backed up directories.
- sftp:
  backed up archives, using ~backupsrv/scripts/bkdb to browser them:

```
cd ~backupsrv/data/backupclient_URL/sftp

~backupsrv/scripts/bkdb ls

Base path: "/home/services/backup/data/backupclient_URL/sftp"

* [0] /home/services/trac

* [1] /home/services/wiki

* [2] /home/services/wordpress

[0] /home/services/trac:

     2013-09-05 01:35:30 [0]


[1] /home/services/wiki:

     2013-09-05 01:48:41 [1]


[2] /home/services/wordpress:

     2013-09-05 01:48:43 [2]


# export backed up archives

~backupsrv/scripts/bkdb export <dest_dir> <data_index>
[datetime_range]
```

Detail

Overview

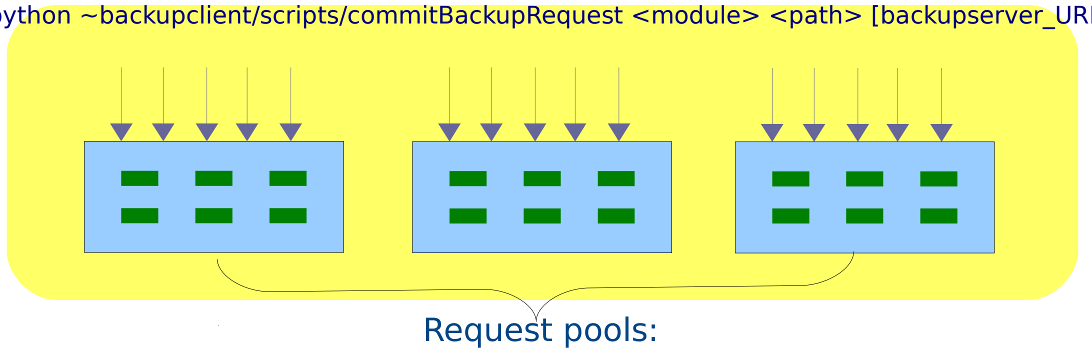
- gitosis controlled git as a control line.
  Backup clients send request through this control line.
- sftp(...) as a data line.
  Backup servers retrieve data to backup through this data line.



The backup process

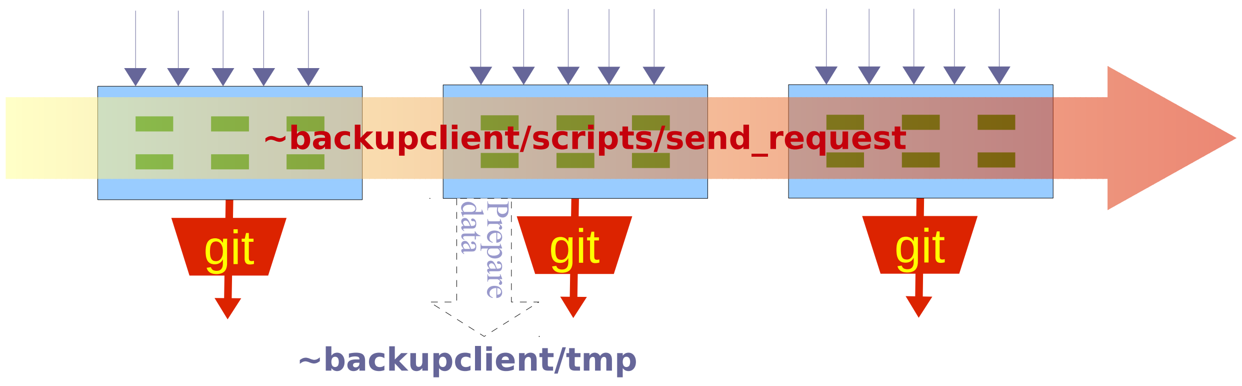
1. Backup client creates requests in request pool(s)

```
python ~backupclient/scripts/commitBackupRequest <module> <path> [backupserver_URL]
```

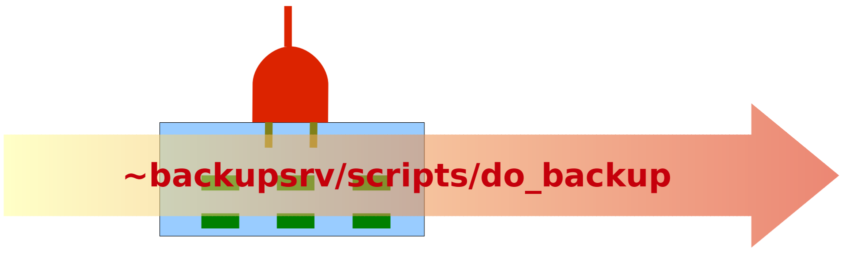


~backupclient/requests/Backupserver1_URL
~backupclient/requests/Backupserver2_URL
~backupclient/requests/Backupserver3_URL

2. A job in cron will check request pool(s) ever 5mins:
   - For some requests, it will prepare the data to backup, e.g. export database, create tar archive, etc.
   - The backup request will be sent to server through a gitosis controlled git line.



3. Backup server: the **post-update** hook will extract requests under ~backupsrv/TODO/backupclient_URL
4. A job in cron will check TODO pool(s) ever 5 mins:
   - It will retrieve the data to backup by git/sftp/rsync ...



Extend the backup system

New backup form can be added to the backup system as follows:

- On client:
  1. Add an entry in **client/backup_it** in form of:

```
commitBackupRequest <module> <path> <backupserver_URL>
```

     Note: variables like <*path*> can be placed in **client/backupconfig.sh**
  2. If the new "module" relies upon an EXISTING "cmd", add a "module to cmd" map in **client/backupconfig.py**
  3. If the new "module" relies upon a new "cmd", it needs to be added:
     - Add the new "cmd" in **client/cmds/**
     - Add a "module to cmd" map in **client/backupconfig.py**
     - Add the path to **normal_sources** array in client/setup.sh

     Note:

- New “cmd” may receive arguments, the “prototype” is specified in *client/backupconfig.py*
- In the end of new “cmd”, it should call “add_queue” with:
  1) base64 encoded <path> as the name;
  2) a backup request in form of “<transfer_method> <URL>”

- `On server:`
  1. Do nothing
  2. Or if a new “cmd” of client relies upon a new transfer method:
     - Add a “cmd” for this new transfer method in **server/cmds**
     - Add a case in **server/do_backup**
     - Add the path to **normal_sources** array in server/setup.sh

State of the backup requests

- request queue of client
  - ✔ `i*: incomlete, the request is creating.`
  - ✔ `C*: complete, the request is created, but hasn't processed.`
  - ✔ `P*: processing`
  - ✔ `F*: finished`
- TODO queue of server
  - ✔ `RQ*: files of this type reside at .incomplete dir, which will then be renamed & moved to its parent dir(i.e. TODO queue)`
  - ✔ `C*: the request hasn't processed yet`
  - ✔ `P*: processing`
  - ✔ `F*: finished`

Known problems

1. In case of the “control line” disconnected suddenly, git will not return a non-zero code, hence no way to detect this failure.