gitzone

tg(x)

October 31, 2013

Contents

1 About

Gitzone is a git-based zone file management tool for BIND. Users can update their zones in a git repository then during a push the zone files are checked, updated & reloaded from git receive hooks. If there's an error in a file being pushed then the push is rejected, thus only correct files are stored on the server. Gitzone is written in Perl.

Gitzone-shell is similar to git-shell but it restricts the user to the zones repository and provides some additional commands for dynamic DNS updates & SSH key management. Gitzone-shell and the Gitzone-install scripts are written in Zsh shell script.

2 Installation (semi-automatic)

First install Bind9 (not covered by this documentation).

Then install all scripts in the prefix /bin path and /libexec

```
# make install
```

Once the binaries are in place, to enable gitzone for a user there is a relatively simple script: gitzone-install. Usage synopsis:

```
# gitzone-install username id_rsa.pub
```

This script assumes that a user with 'username' (first argument) already exists: anyone with access to this user will be in control of gitzone, since access is managed via ssh authentication.

Second argument is the first public ssh key which will have write permissions to change zones (more keys can be added later).

If you intend to use the dynamic DNS feature via the gitzone-shell, then you'd better create a specific user only for gitzone.

Once ready, run the script with all the arguments in place.

Then create /etc/bind/repos/${user}.conf and put inside:

```
zone "domain.com" {
        type master;
        notify yes;
        file "/var/cache/bind/$user/domain.com";
        allow-transfer { transfer; };
};
```

Where 'domain.com' is the first domain you are managing with gitzone. There can be more domains and for each of them the above configuration section must be created.

Now clone the gitzone repository from another user that has access to the ssh secret key configured in gitzone-install. The git url will be composed of the hostname of the machine where is has been installed and the username chosen:

```
 git clone username@ns.myown.net:zones/username gitzone-admin
```

The command above will clone the new gitzone repository into a directory gitzone-admin. If you aren't familiar with git, this is a good time to go study it.

Create a file named 'domain.com' inside gitzone-admin and fill it in as a canonical DNS zone file for bind9. Then add, commit and push:

```
cd gitzone-admin; vim domain.com
 (edit the zone file according to bind9 documentation)
git add domain.com
git commit -m "initial zone commit for domain.com"
git push
```

If the domain.com file contains any errors, gitzone will not accept the push and will report an error among the screen messages.

If all went well, restart the bind9 daemon and you'll see that the zone for domain.com is served by your new DNS. One can check using nslookup.

Gitzone can be installed on multiple users on the same machine, this way there can be different admins (or groups of admins) for different zones all on the same machine.

3 Installation (in close detail)

- set PREFIX in Makefile and make sure the paths in the hooks are correct, then

```
# make install
```

- edit path settings in gitzone-shell
- create users with ssh access and set their shell to gitzone-shell
- create a zones repo for each user and set receive.denyCurrentBranch to ignore, this allows pushing to a checked out repository. The checked out files are used for incrementing serials and validating the zones with named-checkzone.

```
# mkdir -p ~$user/zones
# cd ~$user/zones
# git init $user
# cd $user
# git config receive.denyCurrentBranch ignore
# cd .git/hooks
# ln -s /usr/libexec/gitzone/pre-receive
# ln -s /usr/libexec/gitzone/post-receive
```

- if you want to use a repository locally add these hooks as well / instead:

```
# ln -s /usr/libexec/gitzone/pre-commit
# ln -s /usr/libexec/gitzone/post-commit
```

- create a .gitconfig for each user that contains user name & user email (used for auto increment commits):

```
# git config -f ~$user/.gitconfig user.name $user
# git config -f ~$user/.gitconfig user.email "$user@ns.example.com"
```

- add ssh keys to ~$user/.ssh/authorized_keys and enable ssh key editing if desired:

```
# touch ~$user/.ssh/authorized_keys_edit_allowed
```

- make sure the user's HOME directory has correct permissions:

```
# chown -R $user:users ~$user
```

- edit the settings in gitzone.conf
- create a directory for each user in $zone_dir and chown them to the users, this will contain a clone of the user's repository, the zone files here should be included in named.conf.

```
# cd $zone_dir
# mkdir $user
# chown $user:$group $user
```

- edit named.conf
  - set directory in options to $zone_dir, this is needed to make relative file names work in $INCLUDE:

```
options {
  directory "/var/named";
  // ...
}
```

  - put user zone configuration in a separate file for each user and include them:

```
include "/etc/bind/repos/user1.conf";
include "/etc/bind/repos/user2.conf";
include "/etc/bind/repos/user3.conf";
```

4 Usage

4.1 Git repository

To make changes to the zones you need to clone the git repository, edit the files, commit the changes and finally push the changes to the server. If you use the auto increment feature you also need to pull after a push as the receive hooks on the server make commits to the repository during a push.

```
% git clone ns.example.net:zones/$user zones
% # or if you're using gitzone-shell you can use any path:
% git clone ns.example.net:zones
% cd zones
% # edit files
% git add .
% git commit -m 'commit msg'
% git push origin && git pull
```

4.2 SSH commands

The following SSH commands are provided by gitzone-shell:

- `update-record <filename> <record>`: updates the IP address of the first matched record in the given file to the SSH client's IP address.

  ```
  % ssh ns.example.net update-record example.net somehost IN A
  ```

- SSH key management commands, to use these `touch .ssh/authorized_keys_edit_enabled` in the users' home directories.
  - `list-keys`: list added ssh keys

    ```
    % ssh ns.example.net list-keys
    ```

  - `add-key`: add a new ssh key

    ```
    % ssh ns.example.net add-key ‘cat id_rsa.pub‘
    ```

    or only allow one specific command:

    ```
    % ssh ns.example.net add-key ’command="update-record example.net somehost IN A"’
    ```

  - `del-key`: delete an ssh key from the config

    ```
    % ssh ns.example.net del-key user@somewhere
    ```

4.3 Dynamic DNS

In order to do automatic dynamic DNS updates, create an SSH key without a password and use the add-key command to add it with a command= parameter which has an update-record command in it, see the example in the previous section. This way the host doing the updates does not have access to the git repository as it is restricted to the specified command only. Then all you have to do to update your IP is:

```
% ssh ns.example.net
```

Run this command whenever the IP changes or the interface comes up.

4.3.1 Debian, Ubuntu

On Debian-like systems you can use a post-up command in `/etc/network/interfaces`.

4.3.2 Gentoo

On Gentoo you can put a postup() function in `/etc/conf.d/net`.

4.4 Zone files

There are a few keywords you can use in the zone files:

- ;AUTO_INCREMENT after a serial number to automatically increment it during a push. If the number is 10 digits and starts with 20 it's treated as a date. e.g.:

```
example.net.  IN  SOA  ns1.example.net. hostmaster.example.net. (
                       2011013101  ;AUTO_INCREMENT
                       1d 2h 4w 2d )
```

- $INCLUDE can be used to include other files from the repository, the file names should be prefixed with the user name
- ;INCLUDED_BY on the first line of a file indicates what other files include this file. When this file is committed & pushed all the other files listed after ;INCLUDED_BY are reloaded as well.

  E.g. if you have the following files in the repository then a change in example-common would result in the reload of both example.net & example.org:

  – example.net:

  ```
  ...
  $INCLUDE username/example-common example.net.
  ```

  – example.org:

  ```
  ...
  $INCLUDE username/example-common example.org.
  ```

  – example-common:

  ```
  ;INCLUDED_BY example.net example.org
  ...
  ```