Synchronizing plugins with git submodules and pathogen

Oct 12, 2010

If you use Vim on muliple machines, it can be difficult to keep your configuration files synchronized across them. One solution is to put your dotfiles under version control. In this episode, I demonstrate how to keep your vimrc and plugins synchronized using git submodules and the pathogen plugin.

Keep your dotfiles in git

The following instructions assume that your home directory contains a `.vimrc` file, a `.vim` directory and a `.gvimrc` file (optional).

Move the `.vimrc` and `.gvimrc` files into the `.vim` directory:

```
mv .vimrc ~/.vim/vimrc
mv .gvimrc ~/.vim/gvimrc
```

Create symbolic links so that `~/.vimrc` points to the `~/.vim/vimrc` file:

```
ln -s ~/.vim/vimrc ~/.vimrc
ln -s ~/.vim/gvimrc ~/.gvimrc
```

Change to the `.vim` directory, and initialize it as a git repository:

```
cd ~/.vim
git init
```

Create a `README` file, and paste installation instructions into it (see example README).

Add all files, and make an initial commit:

```
git add .
git commit -m "Initial commit"
```

I suggest publishing your dotvim files to github: it's really easy to set up an account, and they host open source projects for free. In the video, I demonstrate how to publish a git repository to github.

Keep your plugins in git

The traditional method for installing Vim plugins is to copy each script that is distributed with the plugin into the corresponding `.vim` subdirectory. For example, if you wanted to install Fugitive.vim (a git wrapper for Vim), you would copy the documentation file into `.vim/doc`, and copy the plugin file into `.vim/plugin`. You could then check these in to your git repository, and they could be syncronised across machines as easily as the rest of your configuration files. But you lose something by doing this. The Fugitive plugin itself is kept under version control with git. It would be much better if you could keep it that way.

Pathogen.vim

The pathogen plugin makes it possible to cleanly install plugins as a bundle. Rather than having to place all of your plugins side by side in the same directory, you can keep all of the files for each individual plugin together in one directory. This makes installation more straightforward, and also simplifies the tasks of upgrading and even removing a plugin if you decide you no longer need it.

To install Pathogen, download the script and place it in your `.vim/autoload` directory (if the directory doesn't exist, you'll have to create it).

There are a couple of lines that you should add to your .vimrc file to activate pathogen.

```
call pathogen#runtime_append_all_bundles()
```

It is essential that these lines are called before enabling filetype detection, so I would recommend putting them at the top of your vimrc file.

Install plugins as submodules

With pathogen installed, it's now possible to keep the files for each plugin together, which means that every plugin can be kept in its own git repository. The best way to do this is to use git submodules, which are designed especially for the purpose of keeping git repositories within a git repository.

To install the fugitive plugin as a git submodule, take the following steps:

```
cd ~/.vim
mkdir ~/.vim/bundle
git submodule add http://github.com/tpope/vim-fugitive.git bundle/fugitive
git add .
git commit -m "Install Fugitive.vim bundle as a submodule."
```

Installing your Vim environment on another machine

Once your vim configuration is under version control, it's quite straightforward to import your settings to any machine that has git installed. If you followed the instructions above to put your vimrc and plugins in a `dotvim` directory, then you can follow these steps to synchronise them to another machine:

```
cd ~
git clone http://github.com/username/dotvim.git ~/.vim
ln -s ~/.vim/vimrc ~/.vimrc
ln -s ~/.vim/gvimrc ~/.gvimrc
cd ~/.vim
git submodule init
git submodule update
```

As Marcin Kulik points out in the comments below, the last two git commands can be rolled in to one: `git submodule update --init`.

Upgrading a plugin bundle

At some point in the future, the fugitive plugin might be updated. To fetch the latest changes, go into the fugitive repository, and pull the latest version:

```
cd ~/.vim/bundle/fugitive
git pull origin master
```

Upgrading all bundled plugins

You can use the `foreach` command to execute any shell script in from the root of all submodule directories. To update to the latest version of each plugin bundle, run the following:

```
git submodule foreach git pull origin master
```

Further reading

- github - free git hosting for open source projects
- GitCasts screencast on git submodules
- Pathogen.vim - allows Vim plugins to be installed as bundles
- Fugitive.vim - a git wrapper for Vim
- git-submodule(1) Manual Page

Updates

Matt noted in the comments that when you follow this method, generating helptags dirties the submodule's git repository tree. Several other people chimed in with suggestions on how to fix this. Nils Haldenwang has written a blog post describing a simple fix, which just involves adding the line `ignore = dirty` to the .gitmodules file for each submodule that reports a dirty tree when you run `git status`. Go and read Nils's blog post, which goes into a bit more detail.