PERSOONLIJKE VIM-CONFIGURATIE BEHEREN

Op verschillende machines dezelfde VIM-configuratie draaien?

We gaan naar aanleiding van het opzetten van een plug-in manager (Linux Magazine november 2017) de VIM-configuratie universeler maken. Dit zorgt ervoor dat de setup eenvoudig te gebruiken is op verschillende machines. **Paul Reemeijer**

Het is geweldig om verschillende plug-ins in gebruik te hebben voor VIM. Het maakt de editor een stuk aangenamer, maar om dit op iedere machine keer op keer in te stellen en de plug-ins of bundels bij te houden, zorgt voor te veel werk. Het fijnst om mee te starten is een werkplek waar je de VIM-editor op de juiste en persoonlijk ideale manier hebt ingericht.

In het artikel wordt gebruik gemaakt van GitHub als centrale plek. Er zal niet in detail worden besproken hoe je toegang krijgt tot een centrale Git repository. Oplossingen zijn onder andere een lokale GIT-server, GitHub of GitLab. Het configureren van de lokale Git cliënt configuratie is tevens buiten de scope van dit artikel.

CREËER DE REPOSITORY

Log in op GitHub en maak een nieuwe repository aan. Het mag iedere willekeurige naam zijn. Voor het juiste verloop van het artikel gebruikte ik de naam: vim-persoonlijke-configuratie. Als je op GitHub een nieuwe repository aanmaakt, worden een aantal voorbeelden gegeven hoe je gebruik maakt van de nieuw gecreëerde repository. Later in het artikel gaan we het toepassen.

.VIM DIRECTORY

Heb je de .vim directory al onder je homefolder? Verplaats deze naar een andere plek als referentie voor je al gekozen plug-ins. Voer het volgende uit om een nieuwe .vim directory te maken:

```
cd ~
mkdir .vim
cd .vim
mkdir autoload bundle
```

Het is een optie om je bestaande .vim directory onder Git te brengen. Deze optie is nu niet geschikt om verder in dit artikel te behandelen.

VIMRC

Je persoonlijke vimrc verplaatsen naar de .vim directory. Dat is de volgende stap. Sinds de release van Vim 7.4 is het een optie om vimrc te plaatsen in de .vim directory. Deze setting is te vinden in de help van VIM (maar niet in de man van VIM). Deze stap zorgt ervoor dat alles van VIM in de .vim directory aanwezig is. In deze listing gaan we het punt waarmee het bestand start, verwijderen en het bestand op de juiste plek zetten:

```
cd ~
cp .vimrc ~/.vim/vimrc
rm .vimrc
```

Heb je geen vimrc configuratie bestand? In **Afbeelding 1** staat een minimale vimrc configuratie voor het volgen van de rest van het artikel.

Een README.md aanmaken is niet noodzakelijk. Wel is het netjes en krijg je hierover een melding als je de webinterface opent op GitHub van je repository.

GIT REPOSITORY

Om de gehele configuratie van VIM onder Git beheer te brengen, gaan we de Git repository initialiseren. Dit doe je door in de directory ~/.vim te staan en het commando `git init` uit te voeren.

Nu is het tijd voor 'the magic' met Git submodules. Deze techniek zorgt ervoor dat we Git repostories kunnen hebben in onze eigen Git repository. In het vorige artikel werd een plug-in toegevoegd door een GitHub repository uit te checken in de bundle directory. Voor het updaten van een plug-in is dit een handige optie. Dit blijft nog steeds mogelijk met het gebruiken van submodules.

SUBMODULES

De Pathogen plug-in manager was geplaatst onder de autoload directory. Dit laten we zo, alleen maken we er een Git submodule van. In

17 lines (14 sloc) | 493 Bytes — Raw | Blame | History

```
" Set syntax on
syntax on

" == pathogen; plugin manager ==
" https://github.com/tpope/vim-pathogen
" runtime bundle/vim-pathogen/autoload/pathogen.vim
execute pathogen#infect()
filetype plugin indent on

" == NERDTree ==
" Always show the NERDTree
" autocmd VimEnter * NERDTree
" Toggling NERDtree, On/off with CTRL-n
map <C-n> :NERDTreeToggle<CR>
" Close NERDTree if there is no open file
autocmd bufenter * if (winnr("$") == 1 && exists("b:NERDTree") && b:NERDTree.isTabTree()) | q | endif
```

Afbeelding 1

LISTING 1

```
cd ~/.vim
git submodule add https://github.com/tpope/vim-pathogen.git autoload/vim-pathogen
cd autoload
ln -s vim-pathogen/autoload/pathogen.vim pathogen.vim
```

LISTING 2

```
cd ~/.vim
git submodule add https://github.com/scrooloose/nerdtree.git bundle/nerdtree
```

```
[submodule "autoload/vim-pathogen"]
        path = autoload/vim-pathogen
        url = https://github.com/tpope/vim-pathogen.git
[submodule "bundle/nerdtree"]
        path = bundle/nerdtree
        url = https://github.com/scrooloose/nerdtree.gi
~
".gitmodules" 6 regels --100%--
```

Afbeelding 2

Listing 1 ga je eerst in de .vim directory staan, plaats je met het commando git submodule Pathogen op je juiste plek en als laatste maak je een link naar je juiste autoload bestand.

Voor de NERDtree plug-in voeren we het volgende uit (zie **Listing 2**). Voor je overige gekozen plug-ins voer je hetzelfde uit.

In de .vim directory staat nu een .gitmodules bestand. Hierin is de informatie weggeschreven van de toegevoegde submodules, zie **Afbeelding 2**. Voeg de wijzigingen toe door `git add .` uit te voeren.

CENTRALE GIT REPOSITORY TOEVOEGEN

We hebben aan de lokale Git repository nog niet verteld waar onze centrale plek is. Dit doen we met het volgende commando. Vervang [username] met je eigen gebruikersnaam van GitHub (zie **Listing 3**).

Of het toevoegen gelukt is, controleer je met het volgende commando:

```
`git remote -v`
```

Om de wijzigingen te schrijven naar de centrale repository voeren we eerst het volgende commando uitvoeren:

```
`git commit`
```

Alle regels die je op je scherm ziet met een # worden niet getoond op GitHub. Een regel zonder # wordt gebruikt als input voor de commit en wordt getoond op GitHub.

Nu als laatste het volgende commando:

```
`git push -u origin master`
```

Door het commando zal worden gevraagd om een gebruikersnaam en wachtwoord. Vul deze in en kijk op de webinterface van GitHub wat er allemaal is gebeurd.

DE ANDERE WERKPLEK

Pak nu een schone, andere werkplek waar je je persoonlijke VIM configuratie op wilt hebben. Voor het gemak kan dit ook een virtuele machine zijn of de plek waar je de configuratie op hebt gemaakt door de .vim directory naar een andere locatie te verplaatsen.

Nu check je de git repository uit door eerst in je homefolder te gaan staan en de checkout direct naar de juiste directory te schrijven (zie **Listing 4**).

Initieel alle geregistreerde submodules binnenhalen, dat gaat als volgt:

```
cd ~/.vim
git submodule update --init --recursive
```

UPDATEN VAN SUBMODULES

Deze laatste stap laat je zien hoe je het best je submodules naar de laatste versie brengt en deze naar je centrale plek, op GitHub, te sturen.

```
1 cd ~/.vim
2 git pull
3 git submodule update --init
4 git submodule update --remote
5 git add .
6 git commit
7 git push
```

Bovenstaande Listing zijn de stappen die nodig zijn voor het updaten van de aanwezige submodules. Regel 5 en verder is nodig bij daadwerkelijke updates om zo de updates toe te voegen aan je centrale Git repository. Regel 2 voer je uit om er zeker van te zijn dat je met de laatste versie werkt van je Git repository.

TEMPLATE

Voor dit artikel is een schone GitHub repository opgezet samen met de Pathogen manager en NERDtree plug-in. Dit helpt je direct op weg om gebruik te maken van de omschreven set-up. Deze kloon je op de juiste plek in je home directory door gebruik te maken van **Listing 5**.

Bestaat er een .vim directory, maak dan een back-up. Zorg er wel voor dat je een back-up maakt van je eigen .vim directory. Kopieer je eigen vimrc of wijzig de aanwezige vimrc. Voeg je zelfgekozen plug-ins toe, zoals wordt beschreven in **Listing 2** voor de NERDtree plug-in.

SLOT

Deze setup is getest op verschillende Linux distributies met Vim >7.4 en op een MAC met Vim 8 aan boord. GVIM zal tevens netjes gebruik maken van deze set-up. De hier gekozen oplossing, door de configuratie op een centrale plek op te slaan, is voor meerdere softwareproducten toe te passen. Git Submodules zijn ook op veel en verschillende manieren toe te passen, maar op internet bestaan wel waarschuwingen. Submodules zorgt namelijk ook voor complexiteit, vooral bij het gebruik van branches.

LISTING 3

```
`git remote add origin https://github.com/[gebruikersnaam]/vim-persoonlijke-configuratie.git`
```

LISTING 4

```
cd ~
git clone https://github.com/[gebruikernaam]/vim-persoonlijke-configuratie.git .vim
```

LISTING 5

```
cd ~
git clone https://github.com/preemeijer/vim-persoonlijke-configuratie.git .vim
```

LINKS

https://git-scm.com/book/en/v2/Git-Tools-Submodules