Plugin dependency management with Nix

by kuuote

2024-11-23

About me

- Normally enjoying environment setup

- vim-lsp or nvim-lsp or coc.nvim

- vim-lsp or nvim-lsp or coc.nvim
- nvim-treesitter

- vim-lsp or nvim-lsp or coc.nvim
- nvim-treesitter
- fzf.vim or fzf-preview.vim or fzf-lua

- vim-lsp or nvim-lsp or coc.nvim
- nvim-treesitter
- fzf.vim or fzf-preview.vim or fzf-lua
- skkeleton

Q. Common factor in these plugins?

A. Have external dependencies.

For dependency management

- Use system package manager(APT, Homebrew, …)
- Build binary or library when plugin update
- Manual downloads some data

For dependency management

- Use system package manager(APT, Homebrew, …)
- Build binary or library when plugin update
- Manual downloads some data
- Very tired!!

For dependency management

- Use system package manager(APT, Homebrew, …)
- Build binary or library when plugin update
- Manual downloads some data
- Very tired!!
- Peoples want to native plugins(Vim script or Lua only)

There is

There is

Nix is all you need

What’s Nix?

- Declarative builds and deployments
  - Builds everything like pure function
  - Takes a unique approach
    - Immutable store and strictly sandbox
  - Perfect dependency management
    - Fundamentally, haven’t conflicts

Use package manager

- Nixpkgs is official package collection
  - Has sufficient packages and build cache. Can use easily
- Try package
  - `$ nix-shell -p deno` or `$ nix shell nixpkgs#deno`
- Install package to environment
  - `$ nix-env -iA nixpkgs.deno` or `$ nix profile install nixpkgs#deno`

Builds everything

- It’s no different than a global installation, right?
- Can do anything that can with nix mechanism
  - Input => Output

Vim plugins => Vim wrapper with vimrc

It is Nix expression generates Neovim with denops.vim:

```
{
  pkgs ? import <nixpkgs> { },
}:
pkgs.neovim.override {
  configure.customRC = ''
    set runtimepath+=${pkgs.vimPlugins.denops-vim}
    let g:denops#deno = '${pkgs.deno}/bin/deno'
  '';
}
```

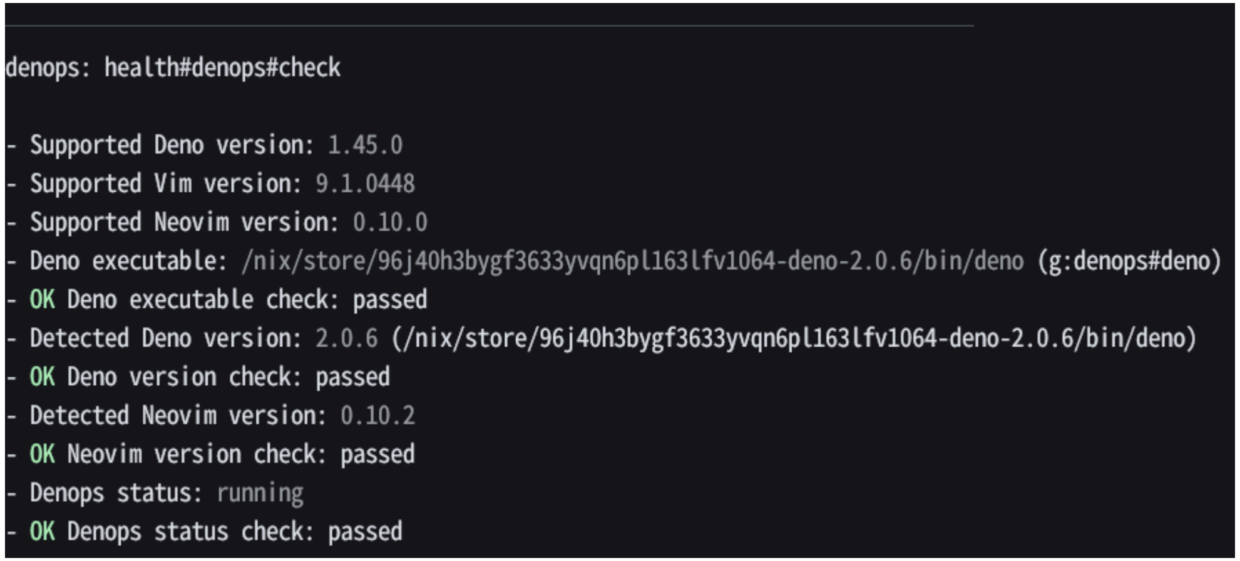
$ nix-build vim.nix

=> generated vimrc:

```
set rtp+=/nix/store/15plwi3r2004kfvjxkg1v7aklmvin26v-vimplugin-denops.vim-2024-11-08
let g:denops#deno = '/nix/store/96j40h3bygf3633yvqn6pl163lfv1064-deno-2.0.6/bin/deno'
```

- /nix/store/… is store path
  - Strictly separate packages

Result of :checkhealth



```
denops: health#denops#check

- Supported Deno version: 1.45.0
- Supported Vim version: 9.1.0448
- Supported Neovim version: 0.10.0
- Deno executable: /nix/store/96j40h3bygf3633yvqn6pl163lfv1064-deno-2.0.6/bin/deno (g:denops#deno)
- OK Deno executable check: passed
- Detected Deno version: 2.0.6 (/nix/store/96j40h3bygf3633yvqn6pl163lfv1064-deno-2.0.6/bin/deno)
- OK Deno version check: passed
- Detected Neovim version: 0.10.2
- OK Neovim version check: passed
- Denops status: running
- OK Denops status check: passed
```

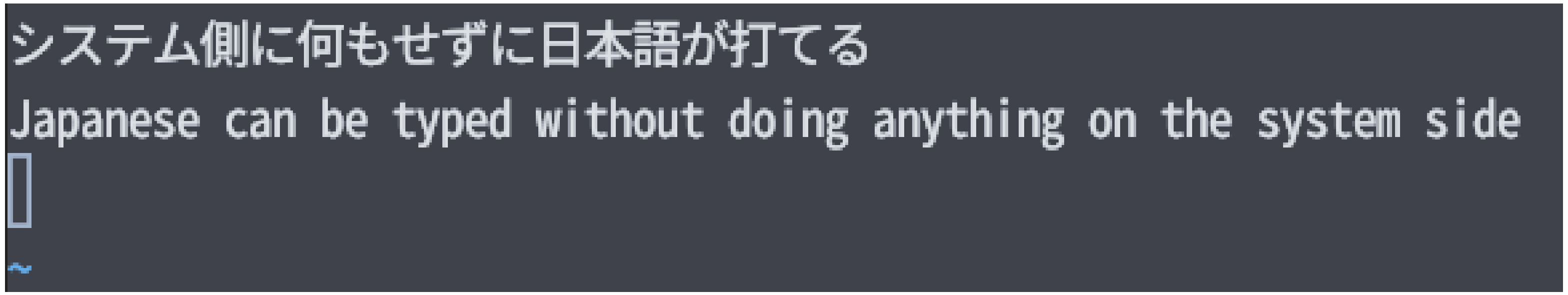
- Running only denops.vim is meaningless
- So add skkeleton
- and Neovim => Vim
  - denops.vim can used in both of Vim/Neovim
  - Can easy switch with Nix

```diff
@@ -3,4 +3,4 @@
 }:
-pkgs.neovim.override {
-  configure.customRC = ''
+pkgs.vim.customize {
+  vimrcConfig.customRC = ''
     set rtp+=${pkgs.vimPlugins.denops-vim}
```

- skkeleton isn’t include to Nixpkgs
- Almost plugins are GitHub repository
  - skkeleton is the same
- Call fetchFromGitHub function to use repository source

```nix
let
  skkeleton_source = pkgs.fetchFromGitHub {
    owner = "vim-skk";
    repo = "skkeleton";
    rev = "2cdd414c1bfa8c363505b8dfc9f50ef2f446ea61";
    hash = "sha256-7oTJGGkUb3K8nzcPqlJrm316ECmgswy/+N8cTQghv3k=";
  };
  skkeleton = pkgs.vimUtils.buildVimPlugin {
    name = "skkeleton";
    src = skkeleton_source;
  };
in
pkgs.vim.customize {
```

```diff
@@ -19,2 +19,7 @@ pkgs.vim.customize {
     let g:denops#deno = '${pkgs.deno}/bin/deno'
+    set rtp+=${skkeleton}
+    call skkeleton#config(#{globalDictionaries: [
+    \ ['${pkgs.skkDictionaries.l}/share/skk/SKK-JISYO.L', 'euc-jp'],
+    \ ]})
+    inoremap <C-j> <Plug>(skkeleton-enable)
   '';
```



```
システム側に何もせずに日本語が打てる
Japanese can be typed without doing anything on the system side

~
```

Appendix

- Q. Fetch plugin is very compilcated
  - A. Exists source expr generator e.g. nvfetcher
- Above contents are written detail at natsukium's zenn article
  - This is japanese only. sorry.
- vim-jp is active Nix community(#tech-nix channel)
  - https://vim-jp.org/docs/chat.html

Appendix2 - Easy to use nvim-treesitter

```
pkgs.neovim.override {
  configure.packages.ts.start = [ pkgs.vimPlugins.nvim-treesitter.withAllGrammars ];
  configure.customRC = ''
    packadd! nvim-treesitter
    lua << EOF
    require('nvim-treesitter.configs').setup {
      highlight = {
        enable = true
      }
    }
    EOF
  '';
};
```

Thank you for your attention