Vim Kitty Navigator

This plugin is a port of Chris Toomey's vim-tmux-navigator plugin. When combined with a set of kitty key bindings and kittens, the plugin will allow you to navigate seamlessly between vim and kitty splits using a consistent set of hotkeys.

Important

This plugin requires kitty v0.30.0 or higher.

Usage

This plugin provides the following mappings which allow you to move between Vim panes and kitty splits seamlessly.

- `<ctrl-h>` → Left
- `<ctrl-j>` → Down
- `<ctrl-k>` → Up
- `<ctrl-l>` → Right

If you want to use alternate key mappings, see the configuration section below.

Installation

Vim

If you don't have a preferred installation method, I recommend using vim-plug. Assuming you have `vim-plug` installed and configured, the following steps will install the plugin:

Add the following line to your `~/.vimrc` file

```
Plug 'knubie/vim-kitty-navigator'
```

Then run

```
:PlugInstall
```

kitty

To configure the kitty side of this customization there are three parts:

1. Add `pass_keys.py` and `get_layout.py` kittens Move both `pass_keys.py` and `get_layout.py` kittens to the `~/.config/kitty/` directory.

This can be done manually or with a `vim-plug` post-update hook:

```
Plug 'knubie/vim-kitty-navigator', {'do': 'cp ./*.py ~/.config/kitty/'}
```

The `pass_keys.py` kitten is used to intercept keybindings defined in your kitty conf and "pass" them through to vim when it is focused. The `get_layout.py` kitten is used to check whether the current kitty tab is in `stack` layout mode so that it can prevent accidentally navigating to a hidden stack window.

2. Add this snippet to kitty.conf Add the following to your `~/.config/kitty/kitty.conf` file:

```
map ctrl+j kitten pass_keys.py bottom ctrl+j
map ctrl+k kitten pass_keys.py top    ctrl+k
map ctrl+h kitten pass_keys.py left   ctrl+h
map ctrl+l kitten pass_keys.py right  ctrl+l
```

By default `vim-kitty-navigator` uses the name of the current foreground process to detect when it is in a (neo)vim session or not. If that doesn't work, (or if you want to support applications other than vim) you can supply a regular expression as a third optional argument to the `pass_keys.py` call in your `kitty.conf` file to match the process name.

```
map ctrl+j kitten pass_keys.py bottom ctrl+j "^.* - nvim$"
map ctrl+k kitten pass_keys.py top    ctrl+k "^.* - nvim$"
map ctrl+h kitten pass_keys.py left   ctrl+h "^.* - nvim$"
map ctrl+l kitten pass_keys.py right  ctrl+l "^.* - nvim$"
```

3. Make kitty listen to control messages Start kitty with the `listen-on` option so that vim can send commands to it.

```
# For linux only:
kitty -o allow_remote_control=yes --single-instance --listen-on unix:@mykitty

# Other unix systems:
kitty -o allow_remote_control=yes --single-instance --listen-on unix:/tmp/mykitty
```

or if you don't want to start kitty with above mentioned command, simply add below configuration in your `kitty.conf` file.

```
# For linux only:
allow_remote_control yes
listen_on unix:@mykitty

# Other unix systems:
allow_remote_control yes
listen_on unix:/tmp/mykitty
```

Tip

After updating kitty.conf, close kitty completely and restart. Kitty does not support enabling `allow_remote_control` on configuration reload.

You can provide a kitty remote control password by setting the variable `g:kitty_navigator_password` to the desired kitty password, e.g.:

```
let g:kitty_navigator_password = "my_vim_password"
```

Tip

Mac users can learn more about command line options in kitty, from this link.

Configuration

Custom Key Bindings

If you don't want the plugin to create any mappings, you can use the five provided functions to define your own custom maps. You will need to define custom mappings in your `~/.vimrc` as well as update the bindings in kitty to match.

Vim Add the following to your `~/.vimrc` to define your custom maps:

```
let g:kitty_navigator_no_mappings = 1

nnoremap <silent> {Left-Mapping} :KittyNavigateLeft<cr>
nnoremap <silent> {Down-Mapping} :KittyNavigateDown<cr>
nnoremap <silent> {Up-Mapping} :KittyNavigateUp<cr>
nnoremap <silent> {Right-Mapping} :KittyNavigateRight<cr>
```

Note

Each instance of `{Left-Mapping}` or `{Down-Mapping}` must be replaced in the above code with the desired mapping. Ie, the mapping for `<ctrl-h>` => Left would be created with `nnoremap <silent> <c-h> :KittyNavigateLeft<cr>`.

Navigating In Stacked Layout

By default `vim-kitty-navigator` prevents navigating to "hidden" windows while in stacked layout. This is to prevent accidentally switching to a window that is "hidden" behind the current window. The default behavior can be overridden by setting the `g:kitty_navigator_enable_stack_layout` variable to `1` in your `~/.vimrc`

```
let g:kitty_navigator_enable_stack_layout = 1
```

Warning

Breaking changes

The latest version of this plugin requires kitty v0.30.0 or higher. This version introduced a new option to `kitty @ focus-window` that allows focusing a neighboring window.

Troubleshooting

vim<->vim navigation

If you are not able to navigate around vim, try the following:

1. Make sure you are using the latest version of Kitty. (If you are on MacOS, make sure the kitty in your `/Applications` folder is the right version as well.)
2. Make sure you are using the latest commit of `vim-kitty-navigator` (both within `~/.config/kitty` and the vim plugin).
3. Add a print statement in `pass_keys.py` between line 7 and 8 like this:
   ```
   def is_window_vim(window, vim_id):
    fp = window.child.foreground_processes
    print(fp)
    return any(re.search(vim_id, p['cmdline'][0], re.I) for p in fp)
   ```
4. Then run kitty in a debug mode:
   `kitty --debug-keyboard`
5. when the new window is opened, open up vim and some splits and try navigating around. When navigating your vim splits you should see some output similar to this:
   ```
   KeyPress matched action: kitten
   [{'pid': 97247, 'cmdline': ['nvim', '.'], 'cwd': '/Users/matt/.config/kitty'}]
   ```
   The `'cmdline': ['nvim', '.']` part will tell us the title of the vim window that we're using to match against in the script. Double check the regex in `pass_keys.py`, or the regex you passed in to `kitty.confg` with that title to make sure they match.

vim->kitty navigation

If you are able to go from kitty -> vim but not vim -> kitty, try the following:

1. Make sure you are using the latest version of Kitty. (If you are on MacOS, make sure the kitty in your `/Applications` folder is the right version as well.)
2. Make sure you are using the latest commit of `vim-kitty-navigator` (both within `~/.config/kitty` and the vim plugin).
3. Make sure other plugins are not overriding the key bindings. To test this, split your kitty window and open `vim` in the left pane. Run `:KittyNavigateRight`. If it moves focus to the right-side kitty window, it's likely another plugin is changing the `ctrl-l` mapping.
4. Try manually setting the mappings:

   `~/.vimrc`

   ```
   let g:kitty_navigator_no_mappings = 1
   ```

```
nnoremap <silent> {Left-Mapping} :KittyNavigateLeft<cr>
nnoremap <silent> {Down-Mapping} :KittyNavigateDown<cr>
nnoremap <silent> {Up-Mapping} :KittyNavigateUp<cr>
nnoremap <silent> {Right-Mapping} :KittyNavigateRight<cr>
```

If you use `neovim` and `init.lua`:

`~/.config/nvim/init.lua`

```
if os.getenv("TERM") == "xterm-kitty" then
    vim.g.kitty_navigator_no_mappings = 1
    vim.g.tmux_navigator_no_mappings = 1

    vim.api.nvim_set_keymap('n', 'C-h', ':KittyNavigateLeft <CR>', { noremap = true, si
    vim.api.nvim_set_keymap('n', 'C-j', ':KittyNavigateDown <CR>', { noremap = true, si
    vim.api.nvim_set_keymap('n', 'C-k', ':KittyNavigateUp <CR>', { noremap = true, sile
    vim.api.nvim_set_keymap('n', 'C-l', ':KittyNavigateRight <CR>', { noremap = true, s
end
```