fzf - command-line fuzzy finder

Make your workflow matchier

Olaf Alders

April 2021

Playing Along

Examples are at https://github.com/oalders/fzf-demo. See the README for examples on how to build and use the Docker container.

Problem: It can be hard to find a thing

fzf helps you find things, like files

but it matches on other things as well.

- `kill` integration
- `ssh` integration
- `vim` integration
- `VSCode` integration
- `unset` environment variables
- `history` search via `ctrl-r`
- `source`

Installation

- `git`
- Package managers
- `homebrew`
- As a `vim` plugin

Wait, install via `vim`?

```
Plug 'junegunn/fzf', { 'dir': '~/.fzf', 'do': './install --all' }
Plug 'junegunn/fzf.vim'
```

Now, `fzf` will be installed for you via `:PlugInstall` and will add the necessary `source` to your shell config.

Can I do this without a vim session?

Install via command line:

```
vim +'PlugInstall --sync' +qa
```

Add the above command to your dot files in order to automate your `fzf` installation.

Usage

- `command **<tab>`
- Use `<tab>` to select multiple lines
- fuzzy matching enabled by default
- Prefix a word with `'` to get an exact (case-insensitive) match

  ```
  kill -9 **<tab>
  > `macdown
  ```

  The above will select all processes running `MacDown`
- Prefix a word with `!` to exclude a match
- `<return>` to exit and pass selections to your command

More Advanced Matching

- `.json$`
- `^lab`
- `^lab | ^bin`

Handy Shortcuts

Emulate `tig`

```
git log --oneline | fzf --multi --preview 'git show {+1}'
```

vim

- `:Files ~`
- `:GFiles` (`git ls-files`)
- `:GFiles?` (`git status`)
- `:Buffers`
- `:Colors`
- `:Lines`
- `:History`
- `:History:`
- `:Commits` (requires Fugitive.vim)
- `:Commands`
- `:Filetypes`
- `:Rg`

vim fullscreen

- Add a trailing `!` to your command: `:GFiles!`

vim preview windows

- Toggle preview window via `ctrl-/`

Grep Repository (via ripgrep)

```
:Rg some text
```

Customizing fzf

```
export FZF_DEFAULT_COMMAND='fd --type f --hidden --follow --exclude .git'
export FZF_DEFAULT_OPTS='--multi --pointer ">>"'
```

Wrapping fzf

```
f () {
    fzf --bind='ctrl-/:toggle-preview' --preview "bat --style=numbers --color=always --line-range :500 {}" "$@"
}
```

Starting at an arbitrary directory

```
find shared/ -type f|fzf
```

Custom choosers

Simple: `shared/bash/simple-chooser.sh`

More flexible: `shared/bash/commands.sh`

Wrappers

Wrapping 3rd party tools: `shared/bash/tmux.sh`

Add completion to existing tools:

`prove` wrapper: `shared/bash/prove-wrapper.sh`

Custom vim commands

```
" Example without a preview window:
command! GShow
  \ call fzf#run({'source': 'git diff-tree --no-commit-id --name-only HEAD~1', 'sink': 'e'})

command! -bang GShowP
  \ call fzf#run(
  \   fzf#vim#with_preview(
  \     fzf#wrap({ 'source': 'git diff-tree --no-commit-id --name-only HEAD~1' }, <bang>0)
  \   )
  \ )
```

mm-psql

```
# mm-psql
_fzf_complete_mm-psql() {
  _fzf_complete --reverse --prompt="mm-psql> " -- "$@" < <(
    echo gi-primary
    echo gi-standby
    echo log-primary
    echo log-standby
    echo mm-primary
    echo mm-standby
    echo monitoring-primary
    echo monitoring-standby
    echo monitoring-local-primary
    echo monitoring-local-standby
  )
}

_fzf_complete_mm-psql_post() {
    awk '{print $1}'
}

[ -n "$BASH" ] && complete -F _fzf_complete_mm-psql -o default -o bashdefault mm-psql
```

prove-this

```
# prove-this
_fzf_complete_prove-this() {
  _fzf_complete --reverse --multi --prompt="prove-this> " -- "$@" < <(
      find t/lib -type f | grep TestFor
  )
}

_fzf_complete_prove-this_post() {
    awk '{print $1}'
}

[ -n "$BASH" ] && complete -F _fzf_complete_prove-this -o default -o bashdefault prove-this
```

mm

```
mm() {
    SELECTION=$(cat ~/local-dot-files/mm.txt | fzf --reverse --no-multi)
    COMMAND=$(echo "$SELECTION" | cut -d'#' -f2-)

    echo "Running $COMMAND"
    eval "$COMMAND"
}
```

Contents of `mm.txt`:

```
1) Install Perl deps    # ./dev/bin/cpan/install-from-cpanfile --sudo
2) Submit Work          # ./dev/bin/general/submit-work.pl
3) Submit One-off       # ./dev/bin/general/submit-work.pl --create
4) Pg Migrations        # ./dev/bin/pg/run-all-migrations
5) Interactive Rebase   #   git rebase -i origin/main
6) mm-psql              #   mm-psql mm-primary
7) update-backend       #   cd ~/mm_website/ansible && time sudo ./roles/ansible/files/update-backend
8) dev-website          #   yarn && ./dev/bin/web/dev-website -ckpf
```