

Getting started with tmux and vim

Abhishek Vishwakarma
vabhishek.me@gmail.com

agenda

- editors & keyboards
- vim
 - intro & history
 - installation & playing with it
 - configuring – making it cool & productive
 - plugins using pathogen package manager
 - resources to learn more
- tmux
 - intro to terminal multiplexers & why to use it
 - Installation & exploring the features
 - configuring it for easiness
 - exploring the possibilities
- Bonus – Pair Programming Using tmux

editors

- sublime
- atom
- vs code
- intellij

downsides:

- require installation
- not much portable
- only GUI Based
- takes time to load/start

But they are awesome.

Amazing features out-of-the-box.

Perfect as primary text editor.

keyboards

QWERTY

- One keystroke to frequently used characters . / [] ; ' \
- Two keystrokes characters with easy access () { } < > :
- One keystroke to type numbers

Why not use the keyboard the way it is designed?

my goal

vim as your secondary text editor
and prevent you from touching mouse

vim is not just a text editor
it's a way of thinking while coding
it's an art

Developer Survey Results 2018



Overview

Developer Profile

Technology

I. Most Popular Technologies

II. Most Loved, Dreaded, and
Wanted

**III. Development Environments
and Tools**

IV. Top Paying Technologies

V. Correlated Technologies

VI. Technology and Society

Work

Community

Methodology

Back to top

Visual Studio Code
34.9%

Visual Studio
34.3%

Notepad++
34.2%

Sublime Text
28.9%

Vim
25.8%

IntelliJ
24.9%

Android Studio
19.3%

Eclipse
18.9%

Atom
18.0%

PyCharm
12.0%

Xcode
10.6%

intro & history

- vim is a '**highly configurable**' '**text editor**' for '**efficiently**' creating and changing '**any kind of text**'.
- It is an improved version of the vi editor distributed with most UNIX systems
- Incredibly Powerful – Can increasing your productivity many times.
- Steep learning curve – takes years to master

History ->

“People think that computer science is the art of geniuses but the actual reality is the opposite, just many people doing things that build on each other, like a wall of mini stones.”

– Donald Knuth

history

- 'ed' (1971) → em → Bill Joy 'ex' (1976)
→ vi (1978) → stevie → vim → nvim
- written by – Bram Moolenaar
- first released in 1991
- Bram Moolenaar bought an Amiga and wanted to use Vi but there was no good Vi for Amiga. Started with the best that was available (a program called Stevie) and started improving it. Later it grew into VIM.
- Over time Vim has grown constantly
- We are now to using Vim 8.1

Installation

ubuntu: (using apt)

```
sudo add-apt-repository ppa:jonathonf/vim;  
sudo apt update;  
sudo apt install vim;
```

mac: (using homebrew)

```
brew install vim --with-override-system-vi
```

other's just do a google search

let's finally learn vim

and be a superdev



I Am Developer

@iamdeveloper



Following

I've been using Vim for about 2 years now, mostly because I can't figure out how to exit it.

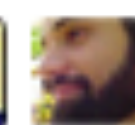
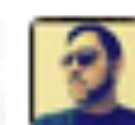
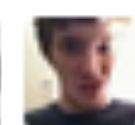
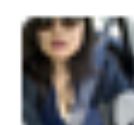
[↩ Reply](#) [↻ Retweet](#) [★ Favorite](#) [⋮ More](#)

RETWEETS

4,846

FAVORITES

2,105



4:56 AM - 18 Feb 2014

Google

how to quit smoking

how to quit **smoking**

how to quit **facebook**

how to quit **sugar**

how to quit **vim**

:q!

modes

- vim provides a modal user interface : this means that the result of pressing any key on the keyboard may differ depending on which mode is active at the time. it's vital to know which mode is active and how to switch between vim's modes.
 - normal mode - default mode [esc]
 - insert mode - type here [i]
 - visual mode - select here [v]

Moving the cursor

**** To move the cursor, press the h,j,k,l ****

```
      ^  
      k  
  
  < h      l >  
  
      j  
      v
```

Hint:

The h key is at the left and moves left.

The l key is at the right and moves right.

The j key looks like a down arrow.

basic commands

\$ vim <filename or path> - opens vim

:w	- write file to disk
:q	- quit
:q!	- quit forcefully
:wq	- save and close
:wq!	- save and close forcefully
:edit <filename>	- open file

basic keys

0	- beginning of line
I	- beginning with insert mode
\$	- end of line
A	- end with insert mode
gg	- first line
G	- last line
w	- move word by word <forward>
b	- move word by word <backward>
e	- end of a word
a	- append
s	- delete current character and append
c<action>	- change (delete and activate insert mode)
o	- create a new line below and activate insert mode
O	- create a new line above and activate insert mode

basic keys

y<action>	- yank / copy
yy	- yank current line
p	- paste
<numbers>	- repeat that many times e.g d2w
x	- delete character under the cursor
r <char> character	- replace current character with some other
dd	- delete current line
>>	- indent current line
<<	- shift-left current line
=	- auto indent
u	- undo
^ r	- redo
~	- swap case

in-between commands

- Vim understands the structure of these well-formed patterns, and it allows us to operate on the regions of text that they delimit.
- Text objects define regions of text by structure. With only a couple of keystrokes, we can use these to select or operate on a chunk of text.

Sample text = "<title> change this </title>"

Just run 'cit' and it will change the text inside tags

'cit' 'cat' 'yit' 'ci{' 'ci[' etc

marks and move

Vim's marks allow us to jump quickly to locations of interest within a document. We can set marks manually, but Vim also keeps track of certain points of interest for us automatically.

`m{a-zA-Z}`

- set a mark

``a`

- move to exact mark position

`'a`

- move to marked line

Tips: Handy Pair

`mm`

- mark it

``m`

- move to it

find / search

f <char>	- goto char <forward>
F <char>	- goto char <backward>
*	- search current word under the cursor
/<text>	- search some text
n	- next match
N	- previous match
:s/<find>/<replace>/g	- find and replace globally

the dot command

- the dot command lets us repeat the last change.
 - the most powerful and versatile command in Vim.
-
- ➔ Write something like ``youtube-dl -f mp4`` and then press `.`
 - ➔ x something and then press `.`

visual mode

- `v` – turns on visual mode
- `V` – selects current line
- `ggVG` – selects whole file
- `<esc>` – switch to normal mode
- `o` – move cursor to boundaries to extend

there are lot of commands
which makes it easy
to traverse the whole file
no matter what language you use

bash output to vim

- get output from any bash command into vim

```
:.! <command> <parameters>
```

e.g

```
:.! figlet ILUGD
```


buffers

- We can load multiple files during an editing session. Vim lets us manage them using the buffer list. (files are stored on the disk, whereas buffers exist in memory.)

<code>:ls</code>	- list all the buffers
<code>:buffer <N></code>	- open N (number) buffer
<code>:bdelete <N></code>	- delete N buffer

registers

- vim's registers are simply containers that hold text. They can be used in the manner of a clipboard for cutting, copying, and pasting text, or they can be used to record a macro by saving a sequence of keystrokes.

We can specify which register we want to use by prefixing the command with "{register}.

If we don't specify a register, then Vim will use the unnamed register.

"{register} command

e.g "a diw - cut and save the word in 'a register'

"b diw - cut some other word and save in 'b register'

"a p - paste from a register

macros

- Vim offers more than one way to repeat changes. Using macros, we can record any number of keystrokes into a register and then play them back.

`q{a-zA-Z}` - record a macro

`@{a-zA-Z}` - play a macro

macros

- here's a story:

- File

```
abhishek, vishwakarma, 1234567890, delhi
aaron, swartz, 0987654321, us
bram, moolenar, 1243568790, america
.....
```

- Convert to

```
{
  "firstname": "bram",
  "lastname": "moolenar",
  "phone": "+91(12345)67890"
},
.....
```

convert the csv file to json with filtering

let's do it live

tabs

- Vim's tab pages can be used to partition work into different workspaces.

<code>:tabnew <filename></code>	- new tab
<code>^wT</code>	- move current window to tab
<code>:tabclose</code>	- close tab
<code>:tabonly</code>	- close all other tabs
<code>:tabnext</code> or <code>gt</code>	- next tab
<code>:tabprevious</code> or <code>gT</code>	- prev tab

windows

- When Vim starts up, it contains a single window. We can divide this window horizontally/vertically

<code>^ws</code>	- split horizontally
<code>^wv</code>	- vertically
<code>^w<h j k l></code>	- move between panes
<code>^ww</code>	- cycle through panes
<code>^wc</code>	- close pane
<code>^w_</code>	- max height
<code>^w </code>	- max width
<code>^w=</code>	- equal size

- You can split the window and open another buffer if you like.

a todo list app

- Open vim ~/.bashrc (or ~/.bash_profile for mac users)
- go to bottom (G) and add (o)
alias todo=~/.todo.md;
alias todo-show="cat ~/.todo.md";
- save the file and exit (:wq!)
- reload the bashrc config
source ~/.bashrc
- use your todo list app - add anything

```
# Todo  
[ ] Do this  
[ ] Do that
```


configuring vim
and
installing plugins
using pathogen package manager

.vimrc file

- vim loads initial configurations from ~/.vimrc file
- you can edit it to customize your vim experience
- Open: <https://github.com/vabhishek-me/talks>
- Goto: [talks/tmux-vim-ilugd/vim/.vimrc](#)
- Copy paste the vimrc content to your ~/.vimrc

let's see what the
configurations are

pathogen package manager

- Open : <https://github.com/tpope/vim-pathogen>
- Follow the installation steps

plugins

- Open : <https://vimawesome.com/>
- It has collection of best plugins
- Let's have a look at these plugins:
 - NerdTree
 - vim-surround
 - youcompleteme

NeoVim

the future of vim

- Open : <https://neovim.io/>
- Neovim is an extension of Vim.
- Goals:
 - Optimize out of the box, for new users but especially regular users
 - Keep the core small and fast
 - Leverage ongoing Vim development: harmony
 - Consistent user experience across platforms
 - Target all platforms supported by libuv

Easiest way to learn VIM








```
root@s:~# apt-get remove vim
```

```
root@s:~# apt-get install nano
```

```
root@s:~# ln -s /usr/bin/nano /usr/bin/vim
```

Q & A for VIM

2 Minutes Break



How to Use



tmux

the Terminal Multiplexer

How to do this?

```
iammrddollar > ... > ilugd > figlet ILUGD
```

iammrddollar ... > ilugd

iammrdollar ... > ilugd

iammrdollar ... > ilugd

iammrddollar ... > ilugd

iammrdollar ... > ilugd

iammrdollar ... > ilugd

iammrddollar ... > ilugd

iammrddollar ... > ilugd

tmux

- tmux is a terminal multiplexer i.e., tmux lets you tile window panes in a command-line environment. Similar to GNU's screen.
- It gives you an IDE like experience when using with VIM
- Let's Install:
 - `sudo apt-get install tmux [ubuntu]`
 - `brew install tmux [macos]`

basic commands

`$ tmux new -s [session-name]` - start tmux session

<code>^b</code>	- prefix key
<code><prefix> "</code>	- split pane horizontally
<code><prefix> %</code>	- split pane vertically
<code><prefix> q</code>	- see pane numbers
<code><prefix> <arrows></code>	- move between panes
<code><prefix> t</code>	- clock
<code><prefix> c</code>	- new window
<code><prefix> <number></code>	- move to window
<code><prefix> d</code>	- detach

`$ tmux ls` - list all tmux sessions

`$ tmux a -t [session]` - attach to session

Hands On

.tmux.conf

customize the look and feel of tmux

- tmux loads initial configurations from ~/.tmux.conf
- edit it to customize your tmux look and functionality
- Open: <https://github.com/vabhishek-me/talks>
- Goto: [talks/tmux-vim-ilugd/tmux/.tmux.conf](#)
- Copy paste to your .tmux.conf

Tips

- You can use VIM + TMUX as a full IDE for most of your development work.
- You can use tmux to work on remote servers with ssh.
- Use Powerline (<https://github.com/powerline/powerline>) to customize your status bar and terminal prompt.
- With tmux, you can even do pair programming (we will see later)

Q & A for tmux

2 Minutes Break

Bonus

Doing pair programming
using tmux

pair programming

- Install tmate from (<https://tmate.io/>)
- tmate is a wrapper around tmux which offers pair programming capability.
- Let's get started

Thank You

Abhishek Vishwakarma

Twitter : [@vabhishek_me](https://twitter.com/vabhishek_me)

Github: github.com/vabhishek-me