#golang #vietnam

MEET UP #5

Vim as Go Editor

Thach Le
thach@dwarvesf.com

Agenda

1. Vim introduction
2. Vim usages
3. Vim as go editor
4. Vim tour

Vim introduction

Programming editors

1. Sublime Text
2. Emacs
3. Atom
4. Vscode
5. Vim
6. Webstorm

What is Vim?

1. Text editor
2. Stands for **Vi Improved**
3. Designed for command line interface(CLI).
4. Can be used as IDE by tons of plugins
5. Can be extensively customized

Why Vim?

1. Friend of terminal-users
2. Keyboard-base, mouse-free interface
3. Everywhere
4. Amazing extensibility
5. Lightweight and fast
6. Semantic commands
7. Impression

Vim usages

Vim setup

1. Install Vim
2. Install Vim plugins manager
3. Configure Vim plugins

Install Vim

Install with home brew:

```
brew install Vim
```

Create file .vimrc

```
touch ~/.vimrc
```

Install Vim plugins manager

Vundle/pathogen/vpm/Plug

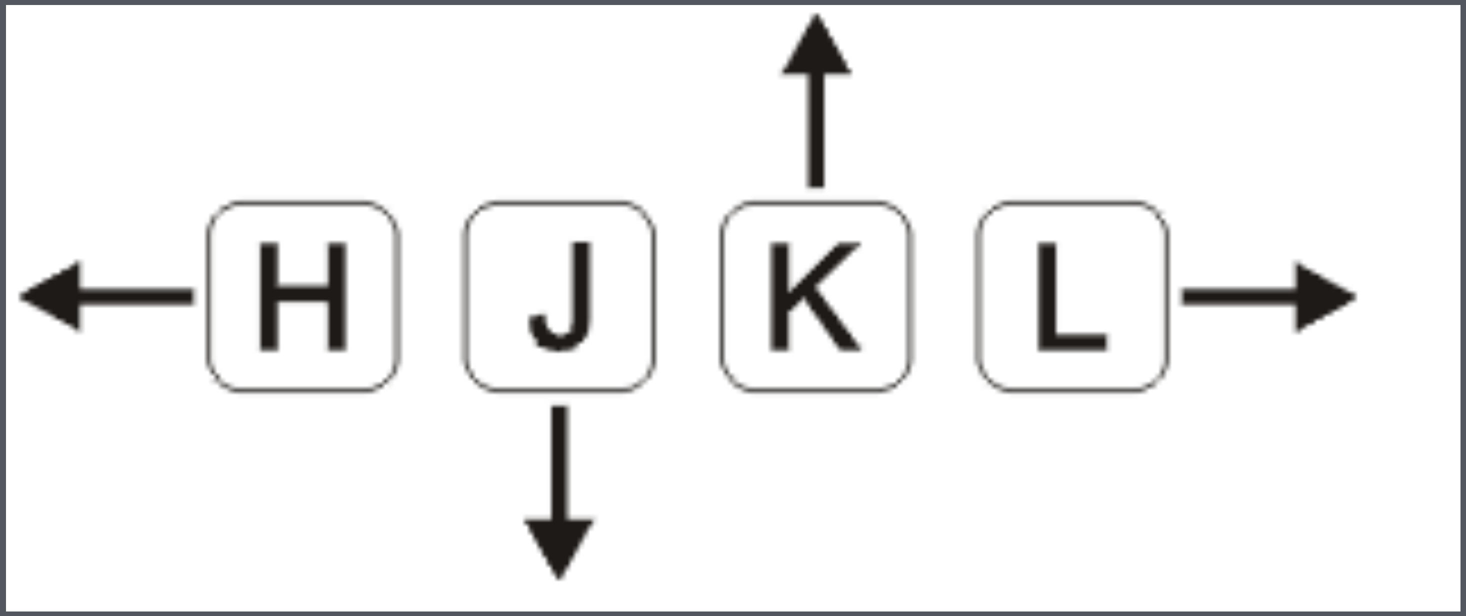
```
git clone https://github.com/VundleVim/Vundle.vim.git ~/.vim/bundle/Vundle.vim
```

Configure Vim plugins

1. Choose plugins from http://vimawesome.com/
2. Place it in **.vimrc**
3. Run **:PluginInstall** in Vim

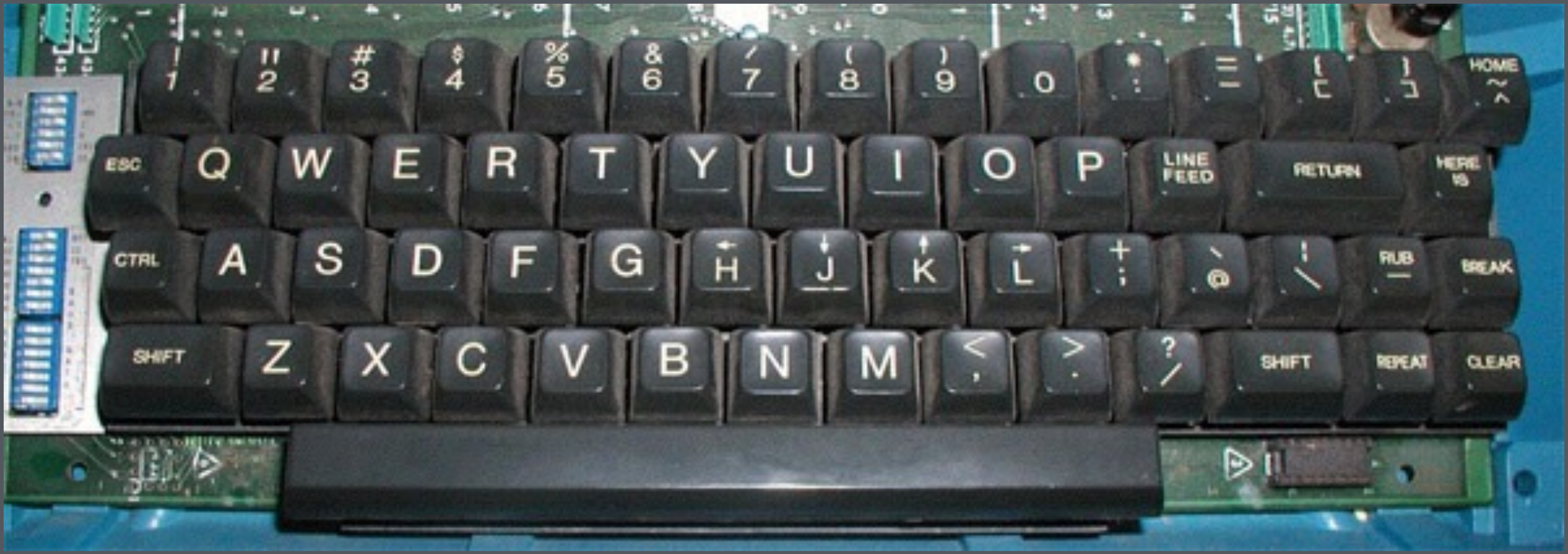
Basic Vim

Movement:



Basic Vim

ADM-3A - First computer terminals



! 1 " 2 # 3 $ 4 % 5 & 6 ' 7 (8) 9 0 * : = - { [}] HOME ~ ^
ESC Q W E R T Y U I O P LINE FEED RETURN HERE IS
CTRL A S D F G ← H ↓ J ↑ K → L + ; ` @ | \ RUB — BREAK
SHIFT Z X C V B N M < , > . ? / SHIFT REPEAT CLEAR

Basic Vim

Open file with vim:

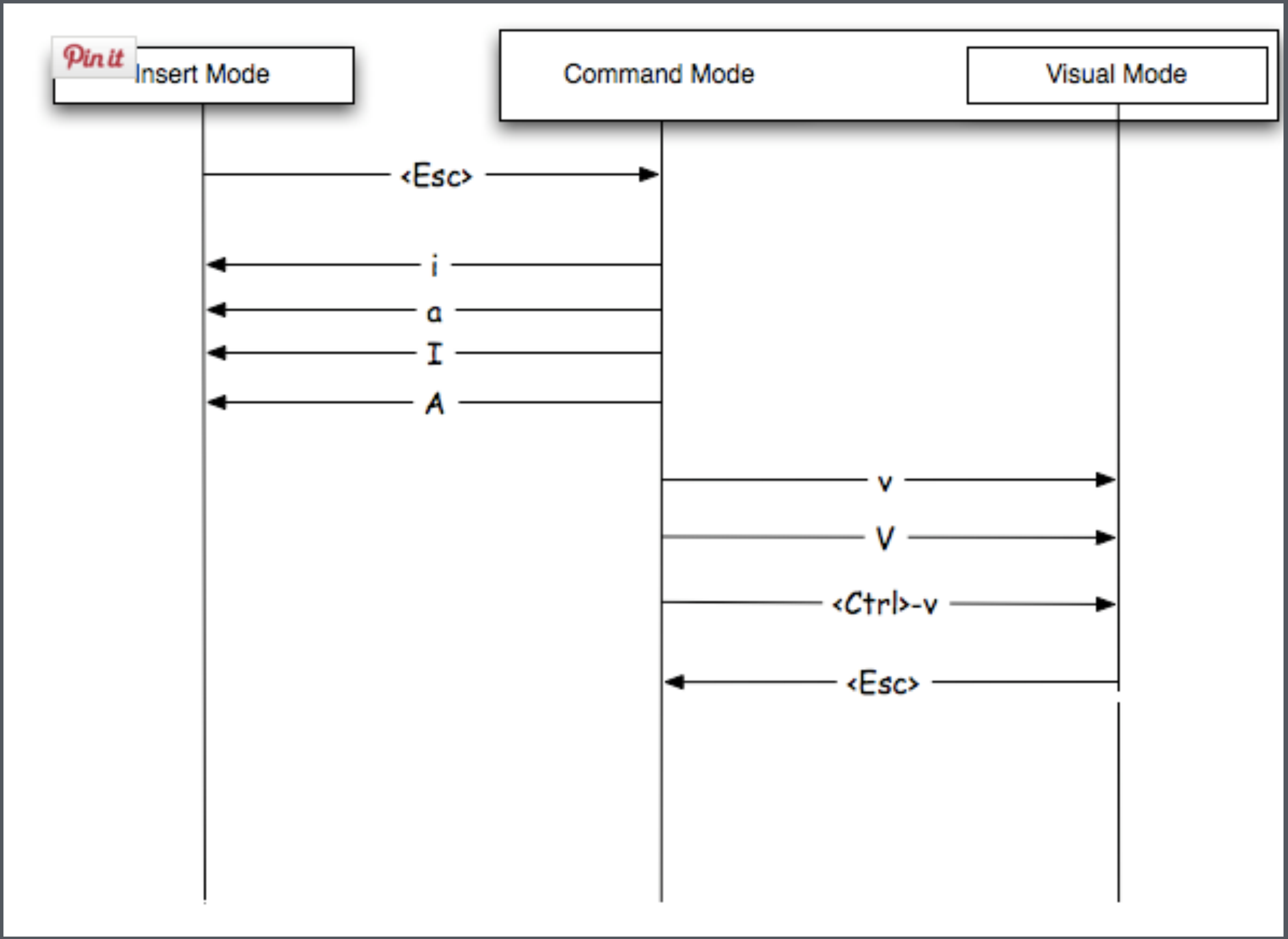
```
vim <file_name>
```

Open vim in current directory:

```
vim/vim .
```

Basic Vim

Vim is **modal** editor:



Command(Normal) mode

Into this mode when pressing **Esc**

All keystrokes are interpreted as commands

Examples: **:q   :w :GoTest**

Insert mode

Into this mode when pressing:

1. **a** - after the current position
2. **A** - last character of line
3. **i** - before current position
4. **I** - first character of line
5. **o** - under current line
6. **O** - above current line
7. **s** - delete char at current position
8. **S** - delete current line

Visual mode

Into this mode when pressing **v, V, Ctrl-V**

Using to select a piece of text, the only way to select a block of text - for further commands

Vim “languages”

Vim’s normal mode command is like a language.

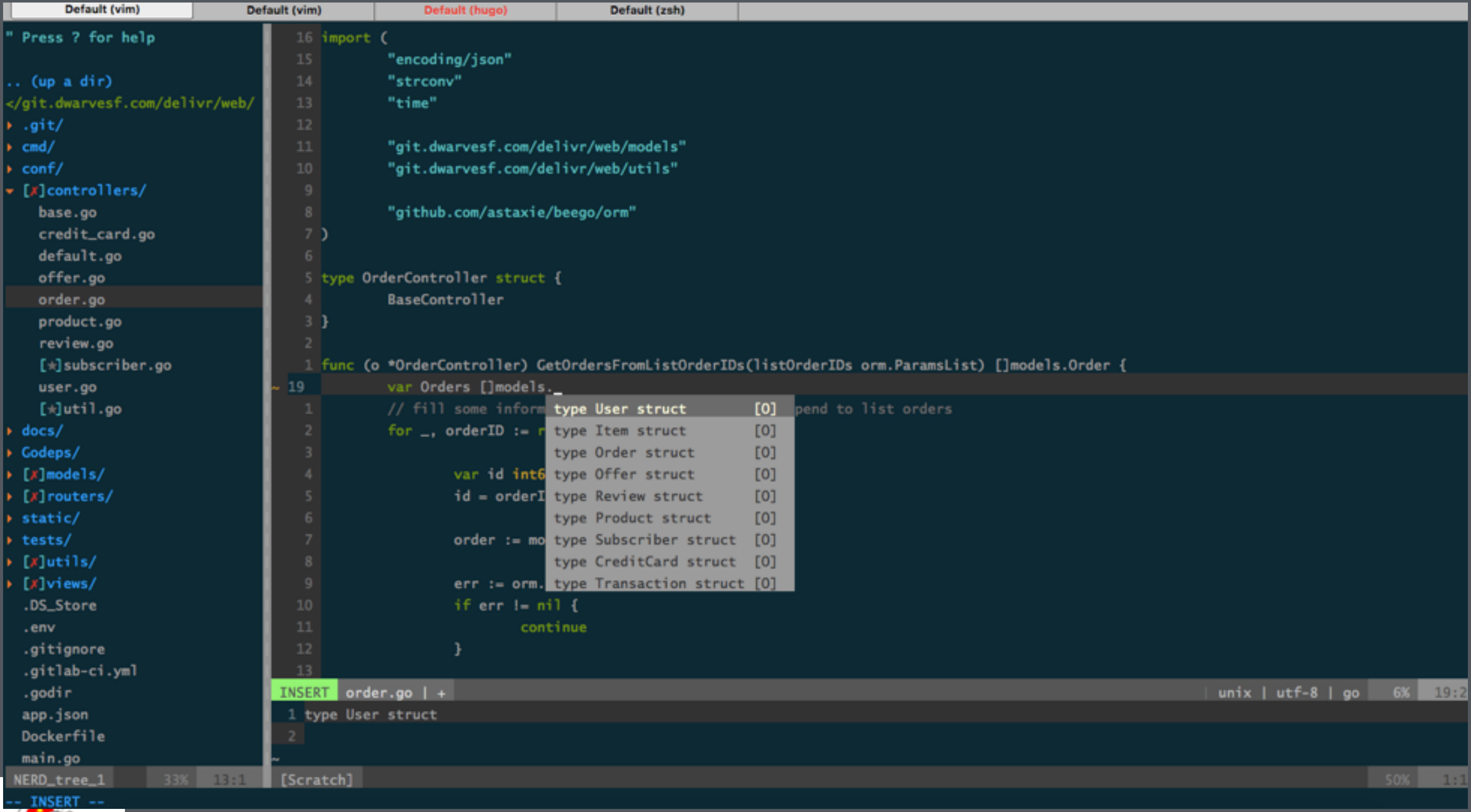
verb	d : delete
noun	w: word
adjective	i : inside

Vim as Go editor

Install plugins

1. vim-go
2. nerd tree
3. neocomplete
4. auto-pairs
5. gitv
6. nerdcommenter

Install Plugins



```
" Press ? for help

.. (up a dir)
</git.dwarvesf.com/delivr/web/
▸ .git/
▸ cmd/
▸ conf/
▾ [✗]controllers/
    base.go
    credit_card.go
    default.go
    offer.go
    order.go
    product.go
    review.go
    [✭]subscriber.go
    user.go
    [✭]util.go
▸ docs/
▸ Godeps/
▸ [✗]models/
▸ [✗]routers/
▸ static/
▸ tests/
▸ [✗]utils/
▸ [✗]views/
  .DS_Store
  .env
  .gitignore
  .gitlab-ci.yml
  .godir
  app.json
  Dockerfile
  main.go
NERD_tree_1                33%   13:1
-- INSERT --
```

```
import (
        "encoding/json"
        "strconv"
        "time"

        "git.dwarvesf.com/delivr/web/models"
        "git.dwarvesf.com/delivr/web/utils"

        "github.com/astaxie/beego/orm"
)

type OrderController struct {
        BaseController
}

func (o *OrderController) GetOrdersFromListOrderIDs(listOrderIDs orm.ParamsList) []models.Order {
        var Orders []models._
        // fill some inform type User struct        [O] pend to list orders
        for _, orderID := r type Item struct        [O]
                            type Order struct       [O]
                var id int6 type Offer struct       [O]
                id = orderI type Review struct      [O]
                            type Product struct     [O]
                order := mo type Subscriber struct  [O]
                            type CreditCard struct  [O]
                err := orm. type Transaction struct [O]
                if err != nil {
                        continue
                }

 INSERT  order.go | +                              unix | utf-8 | go   6%   19:2
type User struct

~
[Scratch]                                                                  50%   1:1
```

Vim’s mapping

When I press this key, I want you to do this action instead what you usually do

```
:map - x
```

=> When I press “-“, vim does action like “x” does.

Using Vim’s mapping

Some kinds of mapping:

- map: work in all modes
- nmap: work in normal mode
- imap: work in insert mode
- vmap: work in visual mode

Using Vim’s mapping

noremap = non-recursive map

```
:map a b
:map b c
=> press a the result is c

:noremap c d
=> press a the result is c
```

Example

```
nnoremap <silent> df :GoDef<cr>

inoremap <silent> pp pp.Println("")
```

Q & A

References

- http://devblog.dwarvesf.com/post/vim-go/
- http://farazdagi.com/blog/2015/vim-as-golang-ide/
- http://vimawesome.com/
- http://learnvimscriptthehardway.stevelosh.com/

Thanks for watching