[Ubuntu20.04 Development Environment]

0. Desktop Background

```
$ gsettings set org.gnome.desktop.background primary-color '#3D5571'
```

1. Default tools

```
$ sudo apt update && sudo apt upgrade && sudo apt install build-essential make cmake clang node-typescript libdbus-1-dev libssl-dev cargo gdebi python3-pip ppa-purge openssh-server git curl screen net-tools pm-utils tldr screenfetch htop tree gnome-tweak-tool gnome-shell-extensions vim uim uim-byeoru valgrind terminator python python3.9
```

2. 한글입력

```
$ sudo apt install fcitx-hangul
```

0. Setting > Region & Language 에서 Korean 추가.

1. Language Support > Keyboard input method system > uim 을 선택해 적용합니다.

2. Activities에서 uim 검색하여 진입하면 uim-pref-gtk UI 가 나옵니다. 여기서 Specify defuault IM 을 활성화한 뒤, Default Input Method 를 Byeoru 로 설정합니다.

3. 좌측 메뉴에서 Byeoru key binding 1 을 선택해 [Byeoru] on 과 [Byeoru] off 에 Multi_key 를 인식시켜야 합니다. Grab... 을 눌렀을 때, Alt_key 가 잡힌다면 4번을 거쳐야 합니다. 정상적으로 Multi_key 가 인식된다면 5번으로 넘어 갑니다.

4. Tweaks > Keyboard & Mouse > Compose Key 를 활성화해 Right Alt 를 선택해줍니다. 다시 3번으로 돌아가 오른쪽 Alt를 눌러 주면 Multi_key 로 인식이 될 것입니다.

5. Multi_key 를 인식해 ON/OFF 에 할당했다면, Apply 를 눌러 적용합니다. 적용되었다면 로그아웃 및 로그인 하여 오른쪽 아래에 uim ui가 뜨는지 확인합니다.

6. Activities 에서 uim-pref-gtk 검색후 Toolbar > Display > Never 로 설정하여 ui 삭제

3. zsh

```
$ sudo apt install zsh
$ chsh -s /usr/bin/zsh
$ wget https://raw.githubusercontent.com/robbyrussell/oh-my-zsh/master/tools/install.sh -O - | sh

$ git clone https://github.com/zsh-users/zsh-syntax-highlighting.git ${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/plugins/zsh-syntax-highlighting

$ git clone git://github.com/zsh-users/zsh-autosuggestions $ZSH_CUSTOM/plugins/zsh-autosuggestions
```

4. prompt

4.1. pure

```
$ mkdir -p "$HOME/.zsh"
$ git clone https://github.com/sindresorhus/pure.git "$HOME/.zsh/pure"
```

4.2. powerlevel10k (good)

```
$ git clone --depth=1 https://github.com/romkatv/powerlevel10k.git
${ZSH_CUSTOM:-~/.oh-my-zsh/custom}/themes/powerlevel10k
```

```
$ p10k configure   (재설정시)
```

5. neovim

```
$ sudo apt-get install neovim
```

6. font-meslo-lg-nerd-font

```
$ wget
https://github.com/ryanoasis/nerd-fonts/releases/download/v2.1.0/DroidSansMono.zip
$ unzip DroidSansMono.zip -d ~/.fonts
$ fc-cache -fv
```

or

```
$ git clone https://github.com/ryanoasis/nerd-fonts.git
$ cd nerd-fonts
$ ./install.sh
$ fc-cache -fv
```

7. ctags & cscope

```
$ sudo apt-get install ctags cscope
```

8. spacevim

```
$ curl -sLf https://spacevim.org/install.sh | bash
$ cd ~/.SpaceVim/bundle/vimproc.vim
$ git pull
$ make
```

9. To apply above steps

```
$ vim ~/.zshrc
```

```
#ZSH_THEME="robbyrussell"
ZSH_THEME="powerlevel10k/powerlevel10k"


plugins=(
  git
  zsh-syntax-highlighting
  zsh-autosuggestions
```

```
)
# pure prompt
#fpath+=$HOME/.zsh/pure
#autoload -U promptinit; promptinit
#prompt pure

# neovim
alias vim="nvim"
alias vi="nvim"
alias vimdiff="nvim -d"
export EDITOR=/usr/local/bin/nvim

# cscope
export CSCOPE_DB=~/Playground/dev/tesla/src/cscope.out
```

```
$ source ~/.zshrc
```

```
$ vim ~/.vim/init.vim
```

```
" ctags
" set tags=./tags,tags;$HOME
set tags=~/Playground/dev/tesla/src/tags,tags;$HOME

" cscope
if has("cscope")
    set csprg=/usr/local/bin/cscope
    set csto=0
    set cst
    set nocsverb
    if filereadable("~/Workspace/rufus_next/src/cscope.out")
      cs add ~/Playground/dev/tesla/src/cscope.out
    endif

    if $CSCOPE_DB != ""
        cs add $CSCOPE_DB
    endif

    set csverb
    set cscopeverbose
    " cscope/vim key mappings
    " 's' symbol: find all references to the token under cursor
    " 'g' global: find global definition(s) of the token under cursor
    " 'c' calls:  find all calls to the function name under cursor
    " 't' text:   find all instances of the text under cursor
    " 'e' egrep:  egrep search for the word under cursor
    " 'f' file:   open the filename under cursor
    " 'i' includes: find files that include the filename under cursor
```

```
    " 'd' called: find functions that function under cursor calls
    nmap <C-\>s :cs find s <C-R>=expand("<cword>")<CR><CR>
    nmap <C-\>g :cs find g <C-R>=expand("<cword>")<CR><CR>
    nmap <C-\>c :cs find c <C-R>=expand("<cword>")<CR><CR>
    nmap <C-\>t :cs find t <C-R>=expand("<cword>")<CR><CR>
    nmap <C-\>e :cs find e <C-R>=expand("<cword>")<CR><CR>
    nmap <C-\>f :cs find f <C-R>=expand("<cword>")<CR><CR>
    nmap <C-\>i :cs find i <C-R>=expand("<cword>")<CR><CR>
    nmap <C-\>d :cs find d <C-R>=expand("<cword>")<CR><CR>

    " CTRL-space <C-@> search and split horizonal window
    nmap <C-@>s :scs find s <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@>g :scs find g <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@>c :scs find c <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@>t :scs find t <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@>e :scs find e <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@>f :scs find f <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@>i :scs find i <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@>d :scs find d <C-R>=expand("<cword>")<CR><CR>

    " CTRL-space CTRL-space vertical split
    nmap <C-@><C-@>s :vert scs find s <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@><C-@>g :vert scs find g <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@><C-@>c :vert scs find c <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@><C-@>t :vert scs find t <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@><C-@>e :vert scs find e <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@><C-@>f :vert scs find f <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@><C-@>i :vert scs find i <C-R>=expand("<cword>")<CR><CR>
    nmap <C-@><C-@>d :vert scs find d <C-R>=expand("<cword>")<CR><CR>

    " key map timeout
    "set notimeout
    " Or
    "set timetouteln=4000
    "set ttimeout
    "set ttimeoutlen=100
endif
```

* Error
connection을 못찾을 경우 cscope.files가 생성된 곳에서 cscope를 터미널에서 한번 실행을 한다.
이후 vi에서 :cs show로 connection을 찾을수 있다.

10. clang-formant

```
$ sudo apt-get install clang-format
$ vim .SpaceVim/init.vim or /etc/vim/vimrc
```

```
" python for osx SpaveVim (osx only)
let g:python_host_prog = '/usr/bin/python'
let g:python2_host_prog = '/usr/bin/python'
let g:python3_host_prog = '/usr/bin/python3'

" clang-format for ubuntu vim
map <C-K> :py3f /usr/share/clang/clang-format-10.0/clang-format.py<cr>
imap <C-K> <c-o>:py3f /usr/share/clang/clang-format-10.0/clang-format.py<cr>
```

10.1. usage
$ clang-format -style=google -dump-config > .clang-format

10.2. neovim에서 python, python3 에러날 경우 처리
$ sudo easy_install pip (pip 설치안된 경우)
$ pip install pynvim
$ pip3 install pynvim

10.3. SpaceVim (NeoVim) 에서 QuickFix window 없애는 법
10.3.1. ale 플러그인 설치
설치는 10.3.2. 라인 추가후 vim 실행시 ale 에러가 발생하면서 자동으로 플러그인이 설치된다.
이후 vim 실행시 ale가 설치되었으므로 에러 메세지는 없어짐.

10.3.2. .SpaveVim의 init.vim 추가

```
" disable QuickFix window
let g:ale_set_quickfix = 0
let g:ale_keep_list_window_open = 0
```

10.3.3. .SpaceVim.d의 init.toml 추가

```
[options]
    # ale enable to disable QuickFix window
    enable_neomake = false
    enable_ale     = true
```

10.6. neomake로 팝업이 발생되면 https://spacevim.org/layers/checkers/ 에서 하기 두가지 방법을 사용해 본다.

```
SPC t s                    Normal                toggle syntax
```

```
SPC e c                     Normal                  clear errors
```

11. Github

```
$ git config --global user.name "jaej"
$ git config --global user.email "jaej.dev@gmail.com"
$ git config --global color.ui true
$ git config --global core.editor vim

$ ssh-keygen -t rsa -C "jaej.dev@gmail.com"
$ cat ~/.ssh/id_rsa.pub
$ ssh -T https://github.com/jaej-dev
```

12. Valgrind

12.1 external debuginfo

```
$ valgrind --leak-check=full --trace-children=yes --show-leak-kinds=all
--track-origins=yes --extra-debuginfo-path="/symbol_path/syms"
--allow-mismatched-debuginfo=yes --verbose --log-file=valgrind-out.txt
./binary_name
```

12.2. normal

```
valgrind --leak-check=full --trace-children=yes --show-leak-kinds=all
--track-origins=yes --verbose --log-file=valgrind-out.txt ./binary_name
```