目录

[vim 1](#_Toc142641334)

[VS CODE 1](#_Toc142641335)

[Installing Extension: Remote Development 1](#_Toc142641336)

[WebStorm 3](#_Toc142641337)

[PyCharm 4](#_Toc142641338)

[Eclipse IDE 4](#_Toc142641339)

# vim

1. 光標在屏幕文本中的移動既可以用箭頭鍵

2. 欲進入vim編輯器(從命令行提示符)，請輸入︰vim 文件名 <回車>

3. 欲退出vim編輯器，請輸入以下命令放棄所有修改︰

<ESC> :q! <回車>

或者輸入以下命令保存所有修改︰

<ESC> :wq <回車>

4. 在正常模式下刪除光標所在位置的字符，請按︰ x

5. 在正常模式下要在光標所在位置開始插入文本，請按︰

i 輸入必要文本 <ESC>

# VS CODE

Code 命令

$ code --help

## Installing Extension: Remote Development

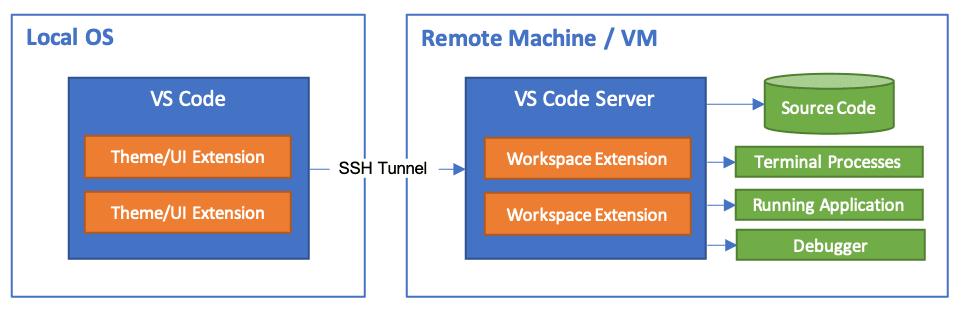
* open any folder in a container, on a remote machine, or in the Windows Subsystem for Linux (WSL) and take advantage of VS Code's full feature set.
* No source code needs to be on your local machine since Remote Development uses tools/runtimes and runs commands/extensions directly on the remote machine.
* Debug an application running somewhere else such as directly inside a container, in WSL, or on a remote machine in the cloud so that everything feels like it does when you run locally.

Remote Development extension pack includes four extensions:

* Remote - SSH - Connect to any location by opening folders on a remote machine/VM using SSH.
* Dev Containers - Work with a separate toolchain or container based application by opening any folder mounted into or inside a container.

**Local VS CODE to connect VM via Remote -SSH**

<https://code.visualstudio.com/docs/remote/ssh>



.ssh> ssh-keygen -t rsa -b 4096 -f dev-vm-ssh # after VM is created

.ssh>scp -p dev-vm-ssh.pub vagrant@192.168.200.3:/home/vagrant/.ssh/authorized\_keys

~/.ssh$ chmod 600 authorized\_keys

~$ chmod 700 .ssh

VS code, ctrl+shift+x -> install extension: Remote – SSH

ssh -> connect to host -> Add New SSH Host: ssh [vagrant@192.168.200.3](mailto:vagrant@192.168.200.3)

.ssh/config:

Host 192.168.200.3

HostName 192.168.200.3

User vagrant

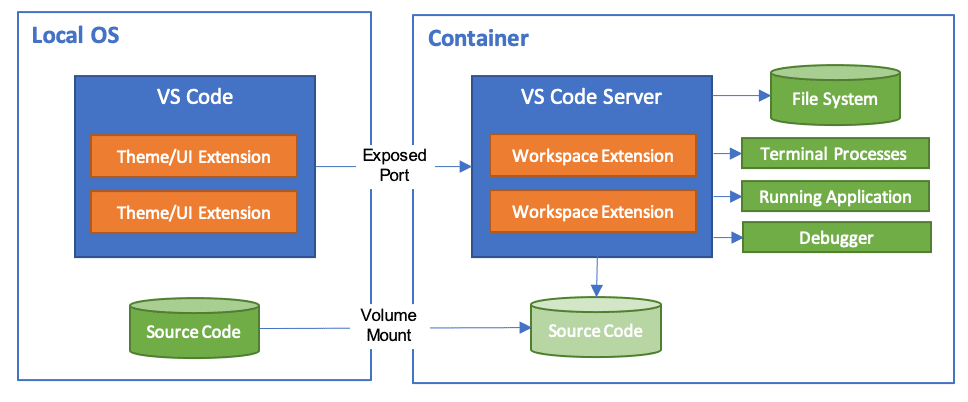
IdentityFile ~/.ssh/dev-vm-ssh

Refresh and ssh connect, then open folder from VM

// Debugging on the SSH host

Once you are connected to a remote host, you can use VS Code's debugger in the same way you would when running the application locally. For example, if you select a launch configuration in launch.json and start debugging (F5), the application will start on remote host and attach the debugger to it.

**Developing inside a Container (refer to `simulation-mid-position-ventilation-lib`)**



Workspace files are mounted from the local file system or copied or cloned into the container. Extensions are installed and run inside the container, where they have full access to the tools, platform, and file system.

open any folder inside (or mounted into) a container. A devcontainer.json file in your project tells VS Code how to access (or create) a development container with a well-defined tool and runtime stack.

Install VS Code, Docker

$ docker --version

// run build container

$ docker container run --tty --interactive --network=host

--volume $REPO\_ROOT:/home/builder/host-data \

--volume $HOME/.conan:/home/builder/.conan \

--workdir=/home/builder/host-data \

<artifactor-ip>/common-images/centos\_07/c-compiler-image.gcc10:edge \

--name <container-name> \

/bin/bash

// lets you run Visual Studio Code inside a Docker container.

Install extension: Dev Containers

The extension starts (or attaches to) a development container running a well defined tool and runtime stack. Workspace files can be mounted into the container from the local file system, or copied or cloned into it once the container is running. Extensions are installed and run inside the container where they have full access to the tools, platform, and file system.

// enter into container

**Vscode -> docker -> containers: <container-name> -> attach vscode**

Open folder: /home/builder/host-data

Install extension: C/C++, CMake Tools

Note: latest vscode extension python only support >python3.7, install down-graded python-extension manually `ms-python.python-2022.6.0.vsix` (refer to <https://blog.csdn.net/Judikator/article/details/125781089> )

copy it to docker and install manually from \*vsix

Run&Debug -> …

# WebStorm

Search everywhere 

Search for project elements

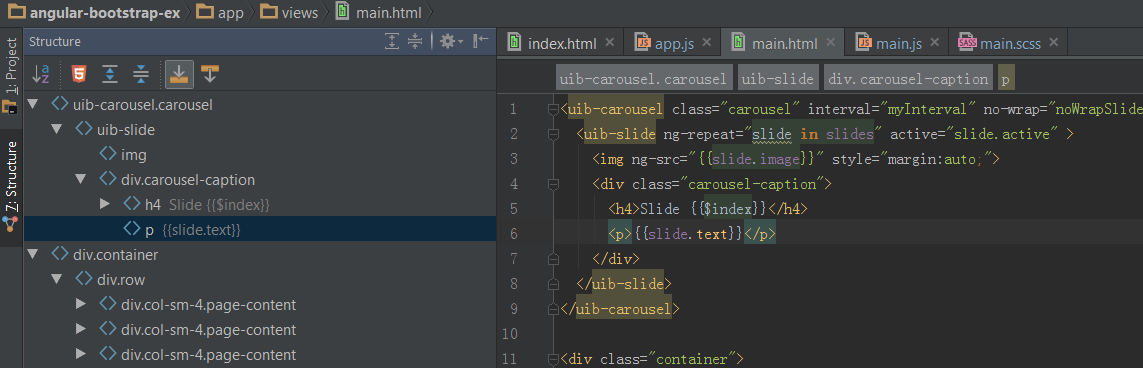
Ctrl + n for class

Ctrl + shift + n for a file

Ctrl + shift + alt + n for a symbol

Search for code elements ctrl + f

To view file structure view Alt + 7



Navigate across the IDE ctrl + tab

Edit and write code

Code completion ctrl + space



Auto-create a new method that you have not yet declared

Intellij idea

How to add junit4 test?

<https://www.jetbrains.com/help/idea/2016.2/creating-tests.html>

将鼠标放到editor里的类名，按Alt+Enter会弹出Create Test, Create subclass

# PyCharm

File -> settings -> Project -> Project Interpreter

默认是系统默认的python 即/usr/bin/python 一般是python2

也可以改为/usr/bin/python3

甚至可以改为virutalenv下的env/bin/python

甚至是远端的python执行环境

这样就可以debug

# Eclipse IDE

1. eclipse中,把java函数代码折叠/展开

首先在eclipse 中开启设置代码折叠功能

a. windows->perferences->General->Editors->Structured Text Editors

可以看到Enable folding选项，打上勾就可以使用代码折叠功能，但还要在具体的语言中设置。

b.windows->perferences->Java->Editors->Folding

可以看到Enable folding选项，打上勾就可以使用代码折叠功能。

其次使用快捷键

下面你就可以用如下快捷键在你的java class 中 折叠或者展开你的代码了.

代码折叠的快捷键，默认是：

Ctrl+Shift+Numpad\_Divede(小键盘的/号)

Ctrl+Shift+Numpad\_Multiply(小键盘的\*号)

笔记本没小键盘，于是改成：

Ctrl+Shift+-

Ctrl+Shift+=

eclipse中，把python函数代码折叠/展开

Ctrl + 9/0

2. override父类API

右击 -> Source -> override/implement methods

3. java注释快捷键

ctrl + shift + /

ctrl + shift + \