3.ROS2 development environment

Theoretically, in ROS2, you only need Notepad to write basic ROS2 programs. However, in order to improve development efficiency, you can install an integrated development environment. It is recommended to use: vscode

In addition, the terminal and git are also often used, which will be explained here.

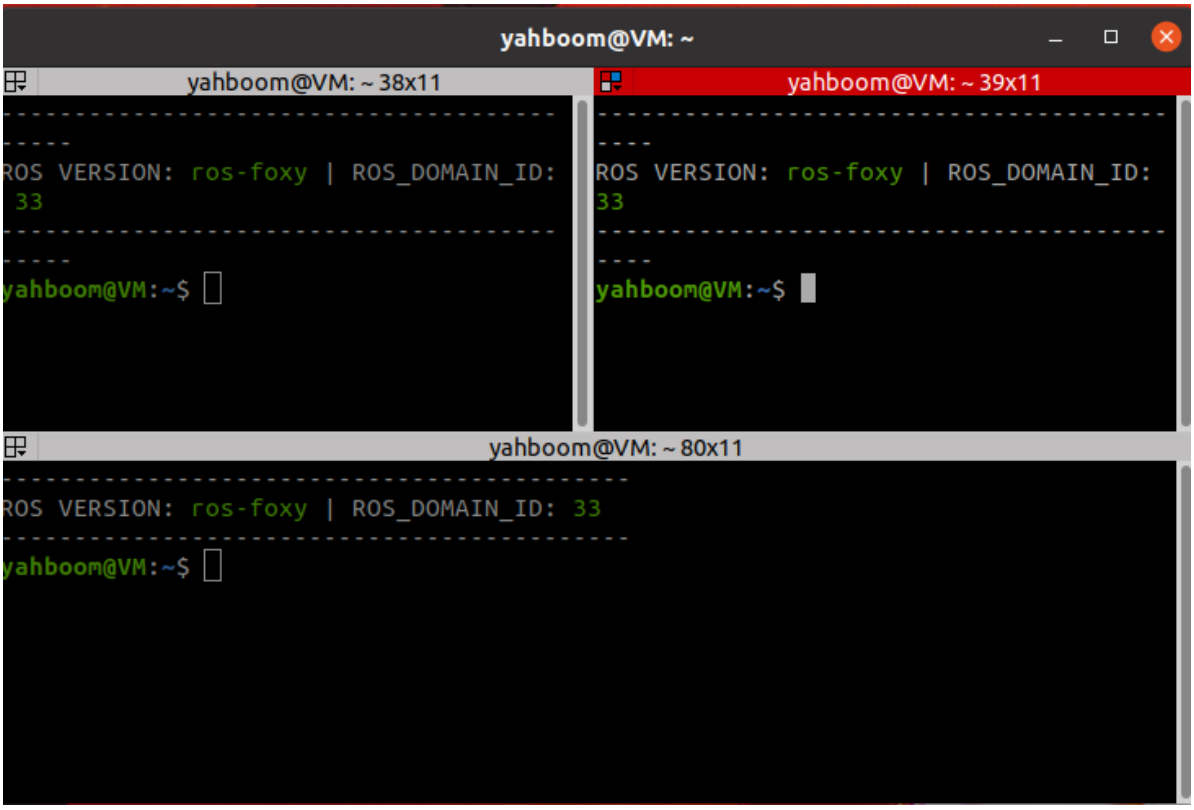
1. Use vscode

Since the tutorial cases in this chapter will be demonstrated in a docker container, here is how to configure vscode to access docker for development.

If the reader does not need to develop in docker and develops directly on the car, just ignore the content in the tutorial about operating docker with vscode. The other steps are basically the same.

2. Use the terminal

In ROS2, you need to use the terminal frequently. We recommend a relatively easy-to-use terminal: Terminator. The effect is as follows:



1. Installation

```
sudo apt install terminator
```

2. Start up

```
Shortcut key `Ctrl+Alt+T` to start
```

3. Commonly used shortcut keys for Terminator

Regarding operations within the same tag:

```
Alt+Up                          //Move to the terminal above
Alt+Down                        //Move to the terminal below
Alt+Left                        //Move to the terminal on the left
Alt+Right                       //Move to the terminal on the right
Ctrl+Shift+O                    //Split horizon terminal
Ctrl+Shift+E                    //Split terminal vertically
Ctrl+Shift+Right                //Move splitter bar to right in vertically split
terminal
Ctrl+Shift+Left                 //Move splitter bar left in vertically split
terminal
Ctrl+Shift+Up                   //Move splitter bar up in horizontally split
terminal
Ctrl+Shift+Down                 //Move splitter bar down in horizontally split
terminal
Ctrl+Shift+S                    //Hide/show scrollbar
Ctrl+Shift+F                    //search
Ctrl+Shift+C                    //Copy selected content to clipboard
Ctrl+Shift+V                    //Paste the contents of the clipboard here
Ctrl+Shift+W                    //Close current terminal
Ctrl+Shift+Q                    //Exit the current window and all terminals in
the current window will be closed.
Ctrl+Shift+X                    //Maximize the display of the current terminal
Ctrl+Shift+Z                    //Maximize the display of the current terminal
and make the font larger
Ctrl+Shift+N or Ctrl+Tab        //Move to next terminal
Ctrl+Shift+P or Ctrl+Shift+Tab  //Crtl+Shift+Tab Move to a previous terminal
```

Regarding the operations between each label:

```
F11                             //Full screen switch
Ctrl+Shift+T                    //Open a new label
Ctrl+PageDown                   //Move to next tab
Ctrl+PageUp                     //Move to previous tab
Ctrl+Shift+PageDown             //Swap the current label with the next label
Ctrl+Shift+PageUp               //Swap the current label with its previous label
Ctrl+Plus (+)                   //Increase font size
Ctrl+Minus (-)                  //Reduce font size
Ctrl+Zero (0)                   //Restore font to original size
Ctrl+Shift+R                    //Reset terminal status
Ctrl+Shift+G                    //Reset the terminal state and clear the screen
Super+g                         //Bind all terminals so that one input can be
input to all terminals
```

```
Super+Shift+G                   //Unbind
Super+t                         //Bind all terminals of the current label, and
content entered into one terminal will be automatically entered into other
terminals.
Super+Shift+T                   //Unbind
Ctrl+Shift+I                    //Open a window. The new window uses the same
process as the original window.
Super+i                         //Open a new window. The new window uses a
different process than the original window.
```

3. Use git

3.1. Installation

In daily work, because it is team collaboration and involves version management, git is an unavoidable skill. git is a free and open source distributed version control system. Install git under Ubuntu:

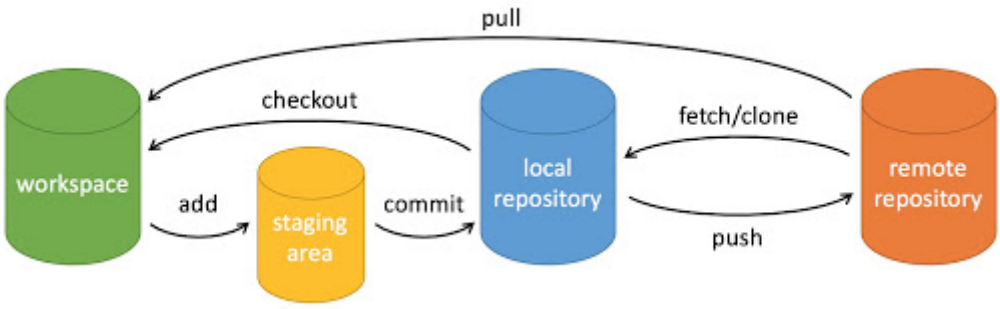
```
sudo apt install git
```

3.2. Git basic operations

Git's job is to create and save snapshots of your project and compare them to subsequent snapshots.

This chapter introduces the commands for creating and submitting snapshots of your project.

The following 6 commands are commonly used in Git: **git clone**, **git push**, **git add**, **git commit**, **git checkout**, **git pull** , we will introduce it in detail later.



Description:

- workspace
- staging area
- local repository
- remote repository

A simple operation step:

```
$ git init
$ git add .
$ git commit
```

- git init - Initialize the repository.
- git add . - Add files to the staging area.

- git commit - Add the contents of the staging area to the repository.

3.2.1. Create repository command

The following table lists the git commands for creating a repository:

Command	Description
`git init`	Initialize the repository
`git clone`	Copy a remote repository, which is to download a project.

3.2.2. Submission and modification

Git's job is to create and save snapshots of your project and compare them to subsequent snapshots.

The following table lists the commands for creating and committing snapshots of your project:

Command	Description
`git add`	Add files to the staging area
`git status`	View the current status of the warehouse and display changed files.
`git diff`	Compare the differences between files, that is, the differences between the staging area and the work area.
`git commit`	Submit the staging area to the local warehouse.
`git reset`	Fallback version.
`git rm`	Delete files from the staging area and workspace.
`git mv`	Move or rename workspace files.
`git checkout`	Branch switching.
`git switch （Git 2.23 Version introduction）`	Switching branches more clearly.
`git restore （Git 2.23 Version introduction）`	Revert or undo changes made to a file.

3.2.3. Submit log

Command	Description
`git log`	View historical submission records
`git blame <file>`	View the historical modification records of the specified file in list form

3.2.4. Remote operation

Command	Description
`git remote`	Remote warehouse operations
`git fetch`	Get the code base from remote
`git pull`	Download remote code and merge
`git push`	Upload remote code and merge

For more information about the use of git tools, you can enter: `git --help` in the terminal to view the help documentation.