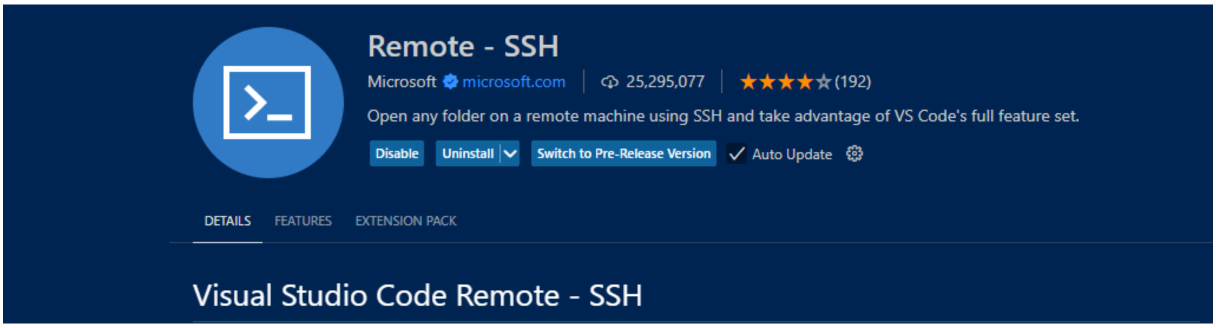
Connecting Linux server with VScode (not Local Terminal)

Xiong Bi

February 2025

1 download plugin from Extensions



Remote - SSH

Microsoft microsoft.com | 25,295,077 | ★★★★☆ (192)

Open any folder on a remote machine using SSH and take advantage of VS Code's full feature set.

Visual Studio Code Remote - SSH

Figure 1: Plugin name

2 click the Remote Explorer

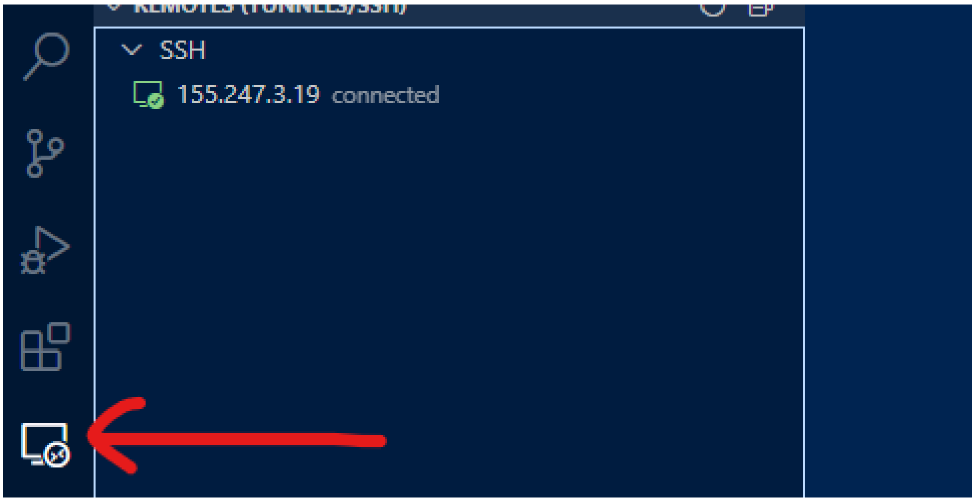


Figure 2: Icon of Remote Explorer

3 click the plus sign

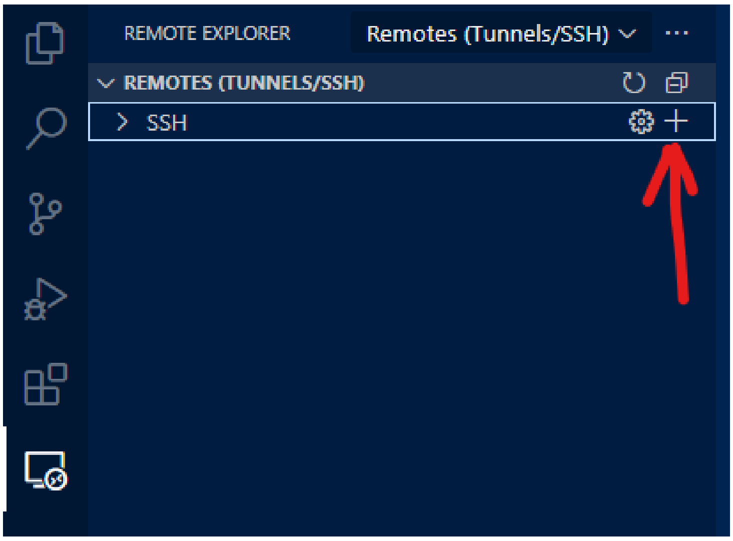
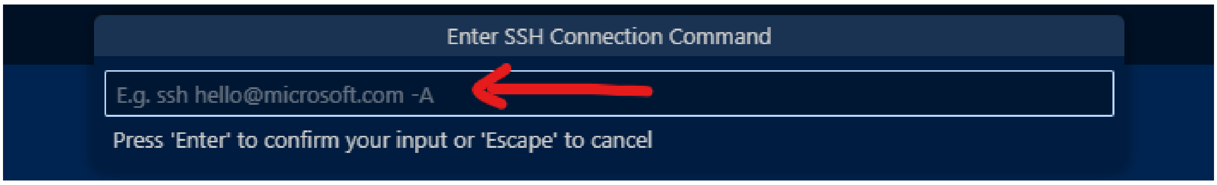


Figure 3: Location of plus sign

4 Input: ssh username@serverIP



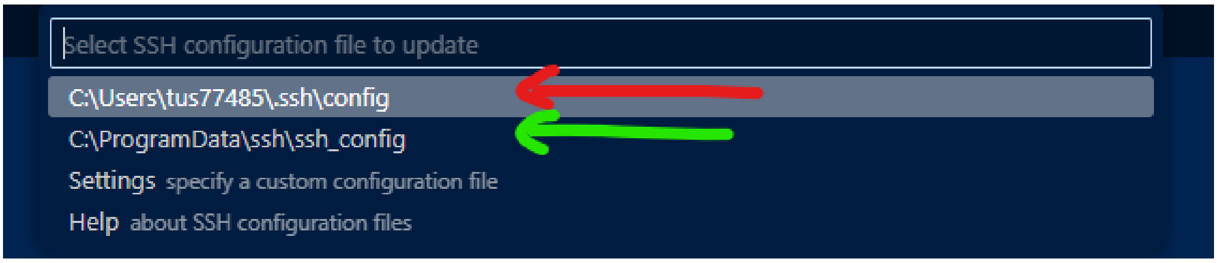
Enter SSH Connection Command

E.g. ssh hello@microsoft.com -A

Press 'Enter' to confirm your input or 'Escape' to cancel

Figure 4: Location of input, and then press Enter

5 save ssh configuration



Select SSH configuration file to update

C:\Users\tus77485\.ssh\config

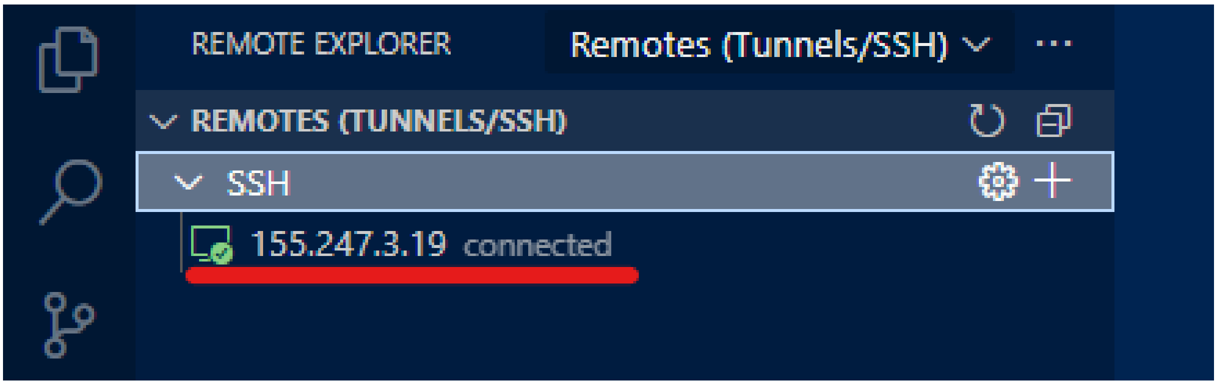
C:\ProgramData\ssh\ssh_config

Settings specify a custom configuration file

Help about SSH configuration files

Figure 5: Save ssh configuration to myself user (Red arrow, don't effect other users and recommend), to all users (Green arrow, not recommend)

6 after clicking red arrow



REMOTE EXPLORER

Remotes (Tunnels/SSH)

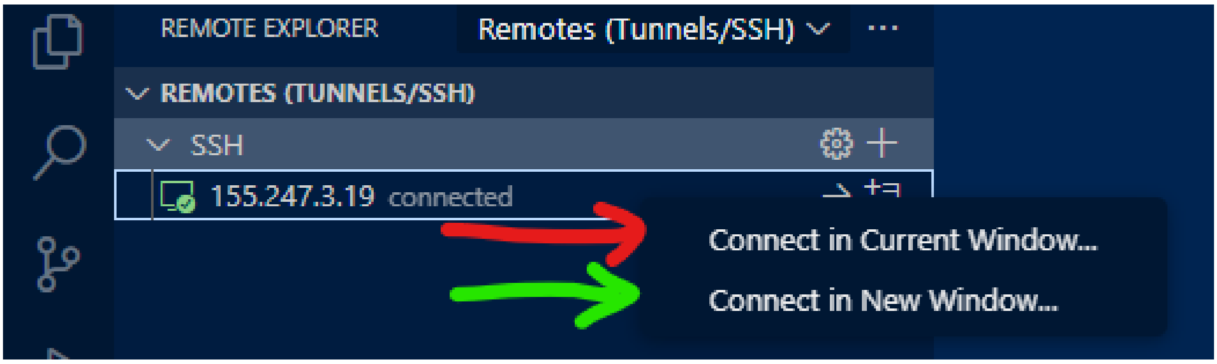
REMOTES (TUNNELS/SSH)

SSH

155.247.3.19 connected

Figure 6: Display serverIP address (i.e., red underlined text)

7 r-click serverIP address



REMOTE EXPLORER

Remotes (Tunnels/SSH)

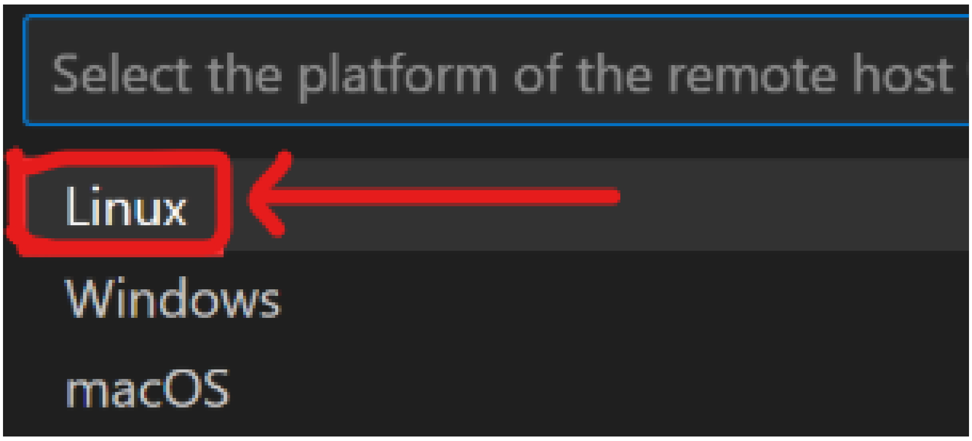
REMOTES (TUNNELS/SSH)

SSH

155.247.3.19 connected

Figure 7: Select a window for connecting (both Red and Green arrow are OK)

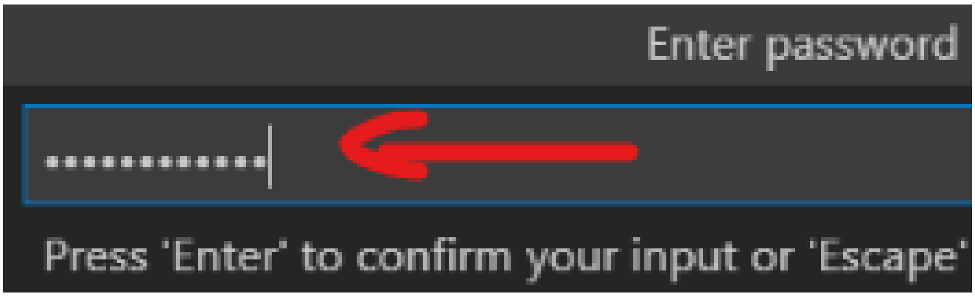
8 Select a server type



Select the platform of the remote host

Figure 8: I select the Linux

9 Input: password



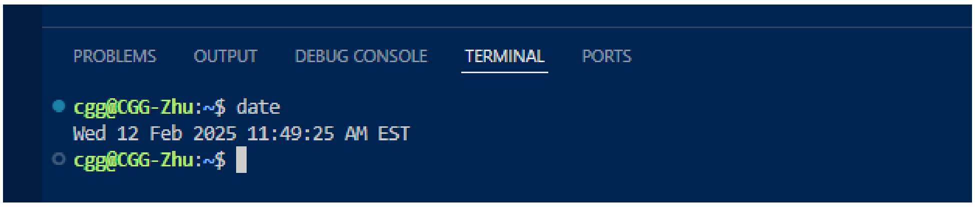
Enter password

••••••••••••

Press 'Enter' to confirm your input or 'Escape'

Figure 9: Input password, and then press Enter

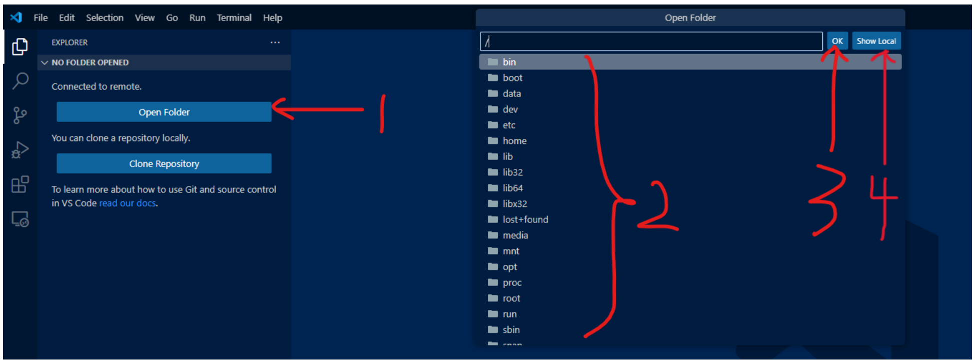
10 Successfully entered the server



```
cgg@CGG-Zhu:~$ date
Wed 12 Feb 2025 11:49:25 AM EST
cgg@CGG-Zhu:~$ 
```

Figure 10: Verify it in the Terminal

11 Open folders



Connected to remote.

You can clone a repository locally.

To learn more about how to use Git and source control in VS Code read our docs.

Figure 11: First step: click the Open folder (i.e., Red arrow 1). Second step: select a folder from 2, and then click the Ok (i.e., Red arrow 3) to open folder of server. If you want to open Local folder, only skip 2 & 3 and directly click Show Local (i.e., Red arrow 4).

12 My choice

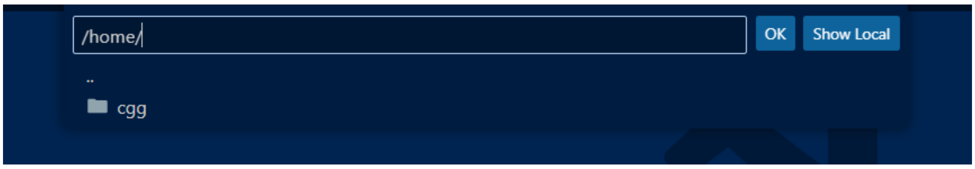


Figure 12: I firstly select server's folder, i.e., home folder. And after clicking OK, there are two images as follow.

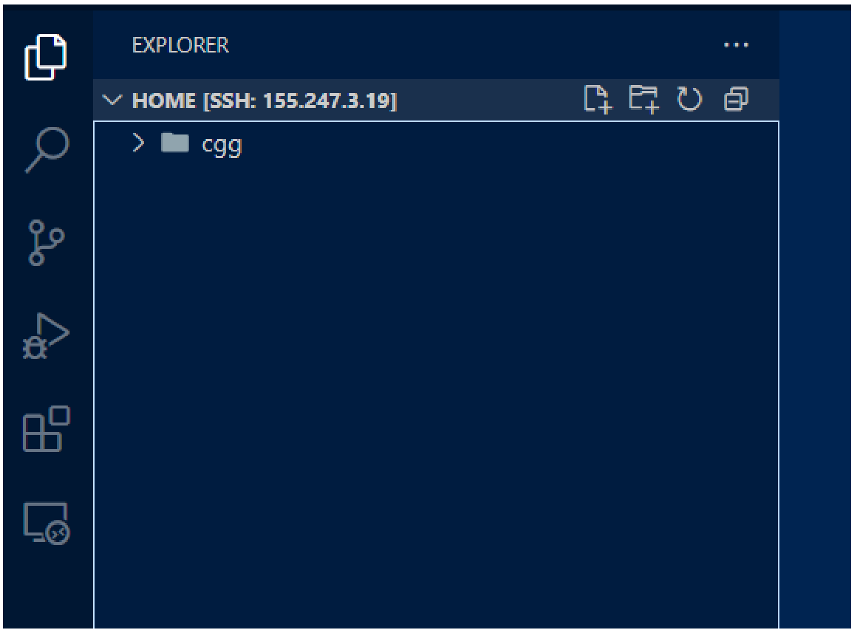


Figure 13: Image 1

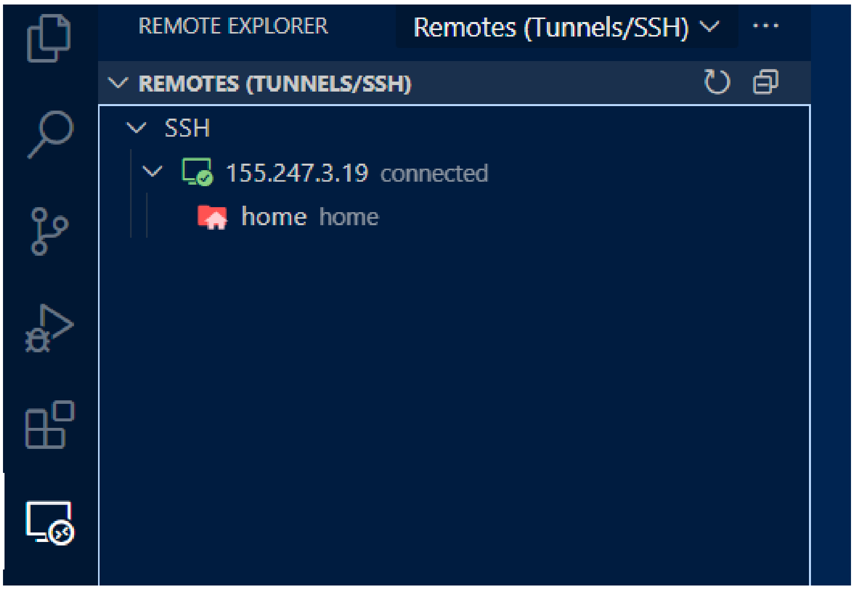
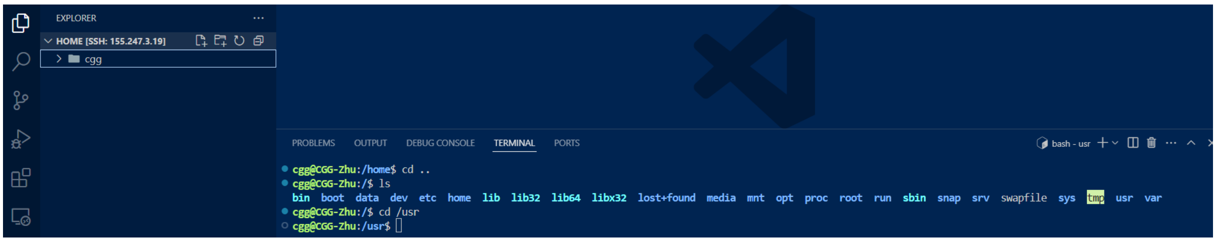


Figure 14: Image 2

13 change folder

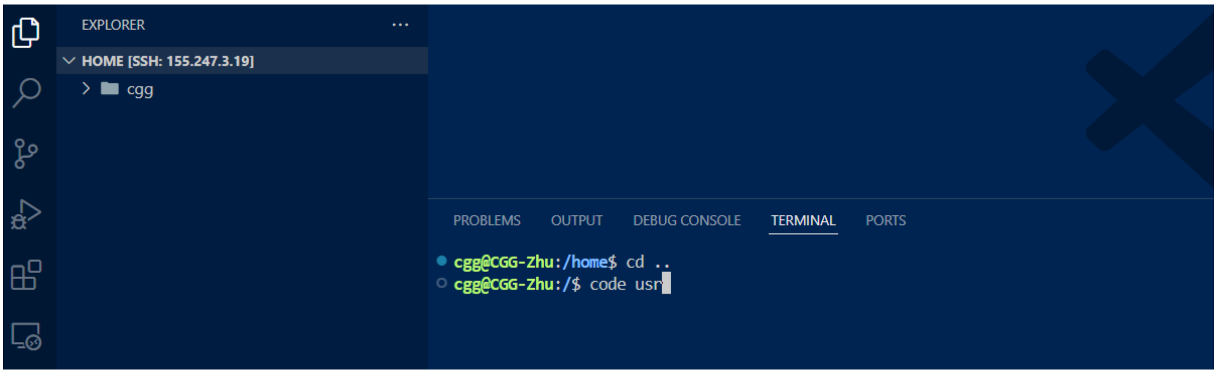
Approach 1



```
cgg@CGG-Zhu:/home$ cd ..
cgg@CGG-Zhu:/$ ls
bin  boot  data  dev  etc  home  lib  lib32  lib64  libx32  lost+found  media  mnt  opt  proc  root  run  sbin  snap  srv  swapfile  sys  tmp  usr  var
cgg@CGG-Zhu:/$ cd /usr
cgg@CGG-Zhu:/usr$ 
```

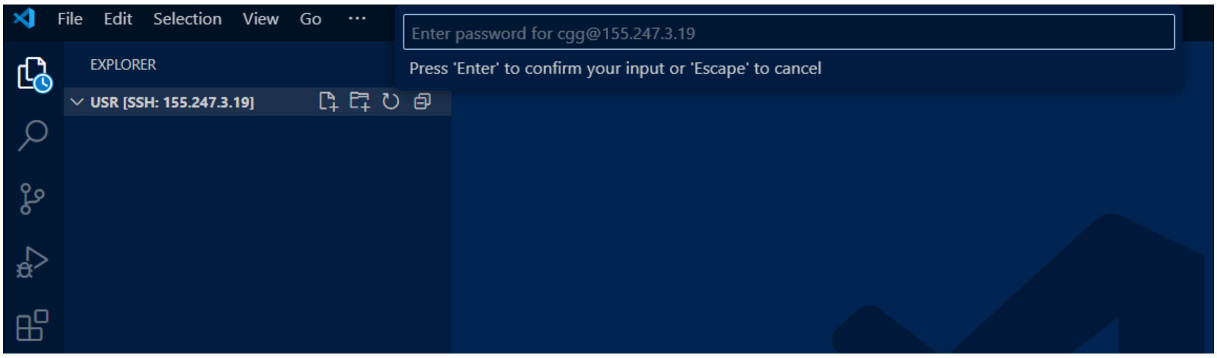
Figure 15: Input some common commands in the Terminal to enter the server's other folder, such as *cd ..* etc. Note: HOME[SSH: 155.247.3.19] doesn't update to USR[SSH: 155.247.3.19] automatically. But it doesn't matter.

Approach 2



```
cgg@CGG-Zhu:/home$ cd ..
cgg@CGG-Zhu:/$ code usr
```

Figure 16: Input *cd ..* to return to the previous directory, and then input *code usr* and press Enter to enter the server's other folder, i.e., usr folder. See the following images.



Enter password for cgg@155.247.3.19

Press 'Enter' to confirm your input or 'Escape' to cancel

Figure 17: After pressing Enter, you can see HOME[SSH: 155.247.3.19] updates to USR[SSH: 155.247.3.19] automatically (Because this is a new window). And then input password to enter usr folder.

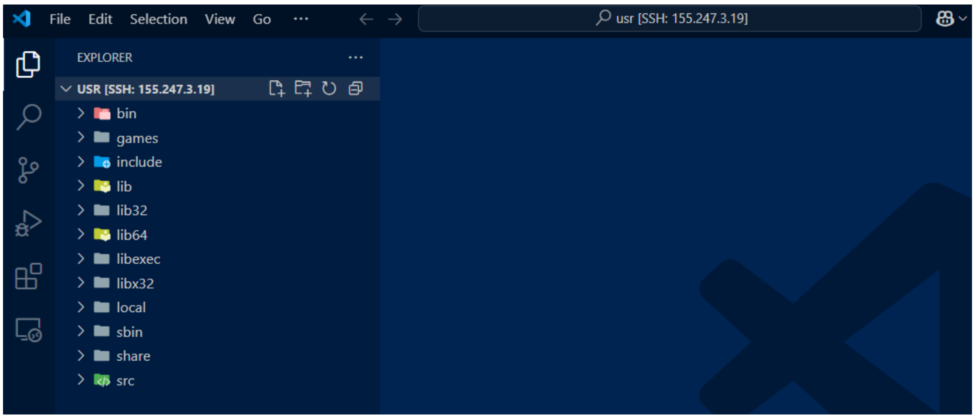
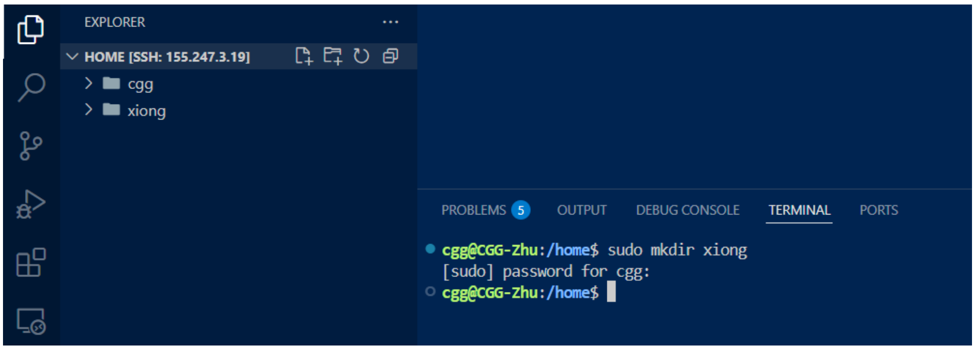


Figure 18: Finally, I entered the usr folder in a new window

14 create my folder

Usually create your own folder under home folder to save your data and files.

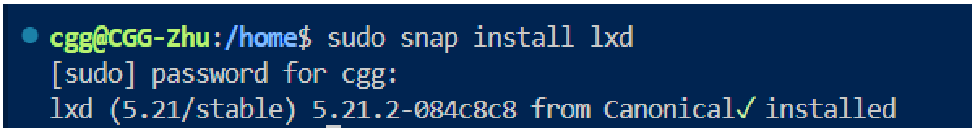


```
cgg@CGG-Zhu:/home$ sudo mkdir xiong
[sudo] password for cgg:
cgg@CGG-Zhu:/home$
```

Figure 19: I created my own folder under home folder. If you want to delete this folder, you can enter *sudo rm -r xiong* in the Terminal.

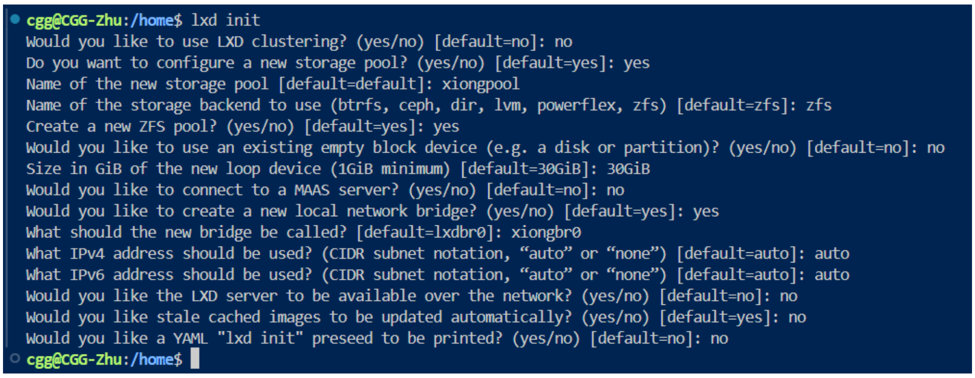
15 create my container

Desktop:
(1) Virtual Box: not used (mainly use Virtual Box to cretae VM in the Desktop).
Server:
(1) Docker: not used (because it cannot install other Ubuntu version; it only uses the server's Ubuntu version to run container).
(2) LXC: not used (because it is so old; use its updated version – LXD)
(3) LXD: recommended
LXD advantages:
(1) everyone can create their own container, and then work independently.
(2) everyone can install some packages & Ubuntu version they want in their own container.
(3) container created by LXD can directly use the server's GPU.
you can learn more about LXC, LXD, Docker from some blogs.



```
cgg@CGG-Zhu:/home$ sudo snap install lxd
[sudo] password for cgg:
lxd (5.21/stable) 5.21.2-084c8c8 from Canonical✓ installed
```

Figure 20: 1. Install container manager – lxd



```
cgg@CGG-Zhu:/home$ lxd init
Would you like to use LXD clustering? (yes/no) [default=no]: no
Do you want to configure a new storage pool? (yes/no) [default=yes]: yes
Name of the new storage pool [default=default]: xiongpool
Name of the storage backend to use (btrfs, ceph, dir, lvm, powerflex, zfs) [default=zfs]: zfs
Create a new ZFS pool? (yes/no) [default=yes]: yes
Would you like to use an existing empty block device (e.g. a disk or partition)? (yes/no) [default=no]: no
Size in GiB of the new loop device (1GiB minimum) [default=30GiB]: 30GiB
Would you like to connect to a MAAS server? (yes/no) [default=no]: no
Would you like to create a new local network bridge? (yes/no) [default=yes]: yes
What should the new bridge be called? [default=lxdbr0]: xiongbr0
What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: auto
What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: auto
Would you like the LXD server to be available over the network? (yes/no) [default=no]: no
Would you like stale cached images to be updated automatically? (yes/no) [default=yes]: no
Would you like a YAML "lxd init" preseed to be printed? (yes/no) [default=no]: no
cgg@CGG-Zhu:/home$
```

Figure 21: 2. Initialize lxd (i.e., configure storage pool, network etc.)

Note:
(1) Would you like to use LXD clustering? (yes/no) [default=no]: no
yes – one server
no – multiple servers
(2) Do you want to configure a new storage pool? (yes/no) [default=yes]: yes
no – existing an old storage pool and you want to use the old storage pool
yes – no storage pool
(3) Name of the new storage pool [default=default]: xiongpool
it's up to you
(4) Name of the storage backend to use (btrfs, ceph, dir, lvm, powerflex, zfs) [default=zfs]: zfs
zfs – suitable for most scenarios, especially multiple containers
(5) Create a new ZFS pool? (yes/no) [default=yes]: yes
no – existing an old ZFS storage pool and you want to use the old ZFS storage pool
yes – no ZFS storage pool
(6) Would you like to use an existing empty block device (e.g. a disk or partition)? (yes/no) [default=no]: no
(7) Size in GiB of the new loop device (1GiB minimum) [default=30GiB]: 30GiB
If there is not enough space later, you can expand the storage pool.
(8) Would you like to connect to a MAAS server? (yes/no) [default=no]: no
yes – used for large-scale physical server management (only used in enterprises or data centers)
no – we select
(9) Would you like to create a new local network bridge? (yes/no) [default=yes]: yes
yes – container can connect to the Internet (such as download software, etc.)
no – cannot connect to the Internet
(10) What should the new bridge be called? [default=lxdbr0]: xiongbr0
it's up to you
(11) What IPv4 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: auto
auto – LXD automatically assign an IPv4 address

(12) What IPv6 address should be used? (CIDR subnet notation, “auto” or “none”) [default=auto]: auto

auto – LXD automatically assign an IPv6 address

(13) Would you like the LXD server to be available over the network? (yes/no) [default=no]: no

yes – LXD container can be accessed and managed remotely over the network.

no – LXD containers can only be accessed and managed locally.

Local management here means connecting to the server with SSH first, and then entering the server’s LXD container.

Remote management means skipping the step of connecting to the server with SSH and directly entering the server’s LXD container (which is risky).

(14) Would you like stale cached images to be updated automatically? (yes/no) [default=yes]: no

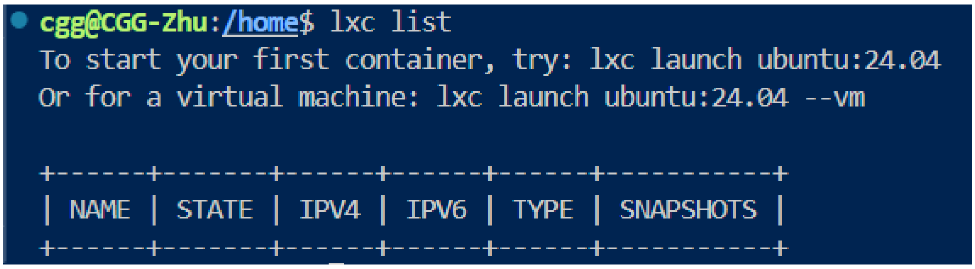
yes – LXD automatically maintains the latest images

no – prevent changes to the experimental environment, as the experiment may depend on a specific version

(15) Would you like a YAML ”lxd init” preseed to be printed? (yes/no) [default=no]: no

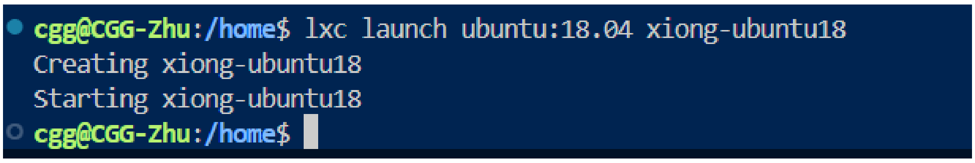
There is no need to save and print the detailed configuration information of the LXD container this time.

Next time, just follow my above Note to configure LXD container information on other computers.



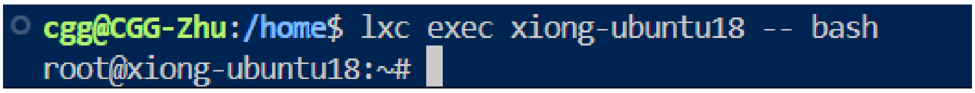
```
cgg@CGG-Zhu:/home$ lxc list
To start your first container, try: lxc launch ubuntu:24.04
Or for a virtual machine: lxc launch ubuntu:24.04 --vm

+------+-------+------+------+------+-----------+
| NAME | STATE | IPV4 | IPV6 | TYPE | SNAPSHOTS |
+------+-------+------+------+------+-----------+
```

Figure 22: 3. If you enter *lxc list* and the output is as shown in the figure — LXD has been successfully initialized and can run normally.



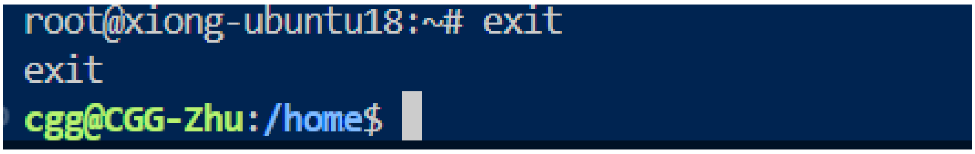
```
cgg@CGG-Zhu:/home$ lxc launch ubuntu:18.04 xiong-ubuntu18
Creating xiong-ubuntu18
Starting xiong-ubuntu18
cgg@CGG-Zhu:/home$
```

Figure 23: 4. Create LXD container. *ubuntu:18.04* is the version I want; I use *xiong-ubuntu18* as the name of my own container.



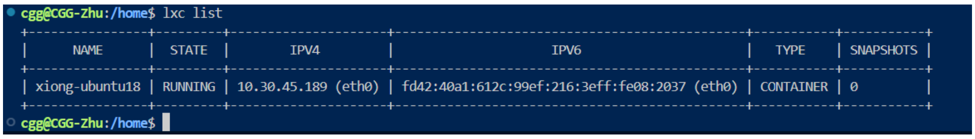
```
cgg@CGG-Zhu:/home$ lxc exec xiong-ubuntu18 -- bash
root@xiong-ubuntu18:~#
```

Figure 24: 5. use *lxc exec xiong-ubuntu18 – bash* to enter my own container



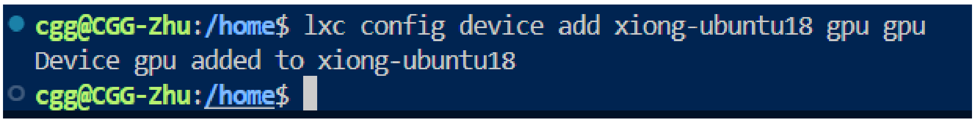
```
root@xiong-ubuntu18:~# exit
exit
cgg@CGG-Zhu:/home$
```

Figure 25: 6. exit my own container and return to the server – use *exit* command.



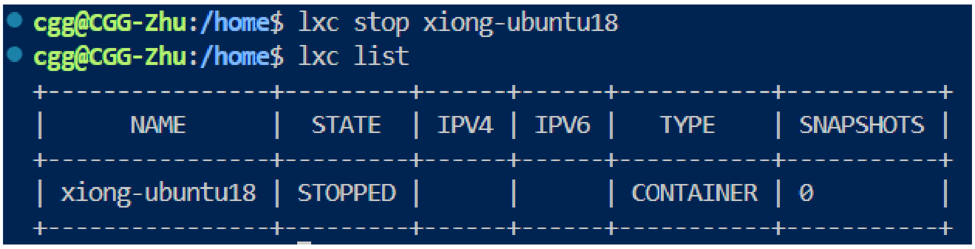
```
cgg@CGG-Zhu:/home$ lxc list
+----------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
|      NAME      |  STATE  |         IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+----------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| xiong-ubuntu18 | RUNNING | 10.30.45.189 (eth0) | fd42:40a1:612c:99ef:216:3eff:fe08:2037 (eth0) | CONTAINER | 0         |
+----------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
cgg@CGG-Zhu:/home$
```

Figure 26: 7. Re-enter *lxc list* to view the created container



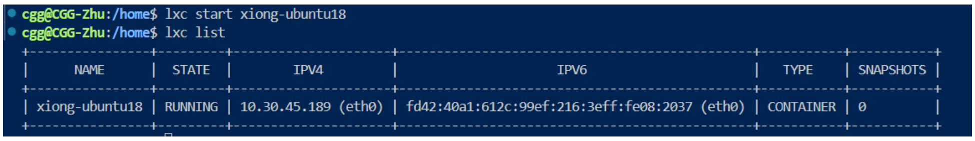
```
cgg@CGG-Zhu:/home$ lxc config device add xiong-ubuntu18 gpu gpu
Device gpu added to xiong-ubuntu18
cgg@CGG-Zhu:/home$
```

Figure 27: 8. enter *lxc config device add xiong-ubuntu18 gpu gpu* to allow my own container to use the server's GPU.



```
cgg@CGG-Zhu:/home$ lxc stop xiong-ubuntu18
cgg@CGG-Zhu:/home$ lxc list
+----------------+---------+------+------+-----------+-----------+
|      NAME      |  STATE  | IPV4 | IPV6 |   TYPE    | SNAPSHOTS |
+----------------+---------+------+------+-----------+-----------+
| xiong-ubuntu18 | STOPPED |      |      | CONTAINER | 0         |
+----------------+---------+------+------+-----------+-----------+
```

Figure 28: 9. If you don't want to use container, you can enter *lxc stop xiong-ubuntu18* to stop your own container. And then, you will see the *STATE* is changed to *STOPPED* after you enter *lxc list*.



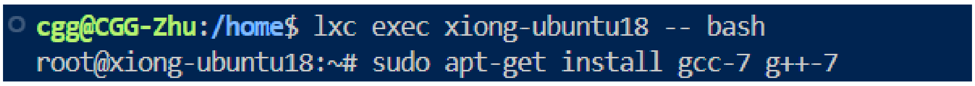
```
cgg@CGG-Zhu:/home$ lxc start xiong-ubuntu18
cgg@CGG-Zhu:/home$ lxc list
+----------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
|      NAME      |  STATE  |         IPV4         |                     IPV6                      |   TYPE    | SNAPSHOTS |
+----------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
| xiong-ubuntu18 | RUNNING | 10.30.45.189 (eth0) | fd42:40a1:612c:99ef:216:3eff:fe08:2037 (eth0) | CONTAINER | 0         |
+----------------+---------+----------------------+-----------------------------------------------+-----------+-----------+
```

Figure 29: 10. If you want to use container again, you can enter *lxc start xiong-ubuntu18* to use your own container. And then, you will see the *STATE* is changed to *RUNNING* after you enter *lxc list*.

11. If you want to delete container, you must sotp your own container firstly, and then you need to enter *lxc delete xiong-ubuntu18* to delete your own container. But I won't display it.

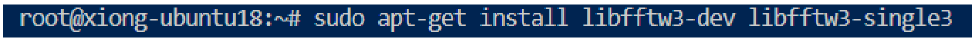
16 Install packages you want in your own container

For example:



```
cgg@CGG-Zhu:/home$ lxc exec xiong-ubuntu18 -- bash
root@xiong-ubuntu18:~# sudo apt-get install gcc-7 g++-7
```

Figure 30: Install gcc-7, g++-7



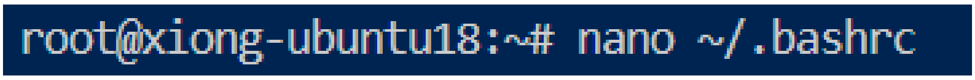
```
root@xiong-ubuntu18:~# sudo apt-get install libfftw3-dev libfftw3-single3
```

Figure 31: Install libfftw3-dev, libfftw3-single3

you can install any packages you want.

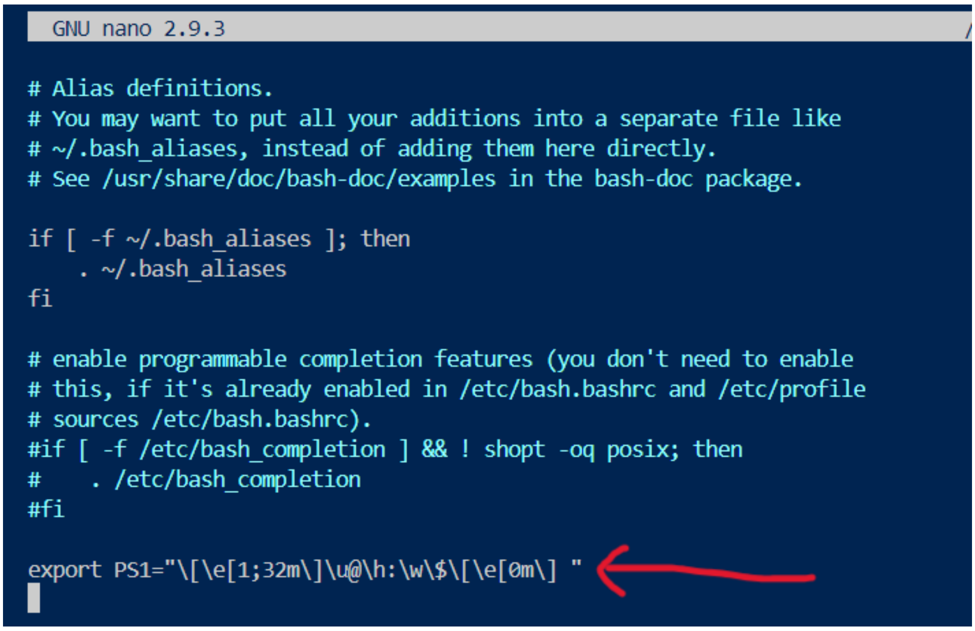
17 Highlight terminal prompt

If you want your own container's terminal prompt *root@xiong-ubuntu18* to be highlighted like the server's terminal prompt *cgg@CGG-Zhu*, see the following image:



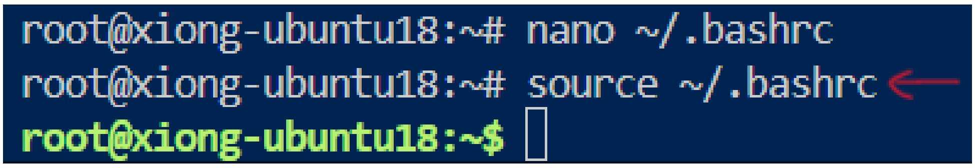
```
root@xiong-ubuntu18:~# nano ~/.bashrc
```

Figure 32: 1. enter the command as shown in your own container to edit the environmental variables. Note: you must open your own container firstly.



```
  GNU nano 2.9.3

# Alias definitions.
# You may want to put all your additions into a separate file like
# ~/.bash_aliases, instead of adding them here directly.
# See /usr/share/doc/bash-doc/examples in the bash-doc package.

if [ -f ~/.bash_aliases ]; then
    . ~/.bash_aliases
fi

# enable programmable completion features (you don't need to enable
# this, if it's already enabled in /etc/bash.bashrc and /etc/profile
# sources /etc/bash.bashrc).
#if [ -f /etc/bash_completion ] && ! shopt -oq posix; then
#    . /etc/bash_completion
#fi

export PS1="\[\e[1;32m\]\u@\h:\w\$\[\e[0m\] "
```

Figure 33: 2. enter the Red arrow command at the end firstly, and then click *Ctrl+x* (i.e., save edited), and enter *y* (i.e., exit environmental variables).



```
root@xiong-ubuntu18:~# nano ~/.bashrc
root@xiong-ubuntu18:~# source ~/.bashrc
root@xiong-ubuntu18:~$
```

Figure 34: 3. After you exited the environmental variables, enter the Red arrow command and press Enter to save edited permanently, you will see the color of *root@xiong-ubuntu18* is highlighted.

Finish!!!