

# Network System Capstone @CS.NYCU

2025.03.13 Lab2 Beamforming with NS3

Instructor: Kate Ching-Ju Lin (林靖茹)

Deadline: 2025.04.03 23:59

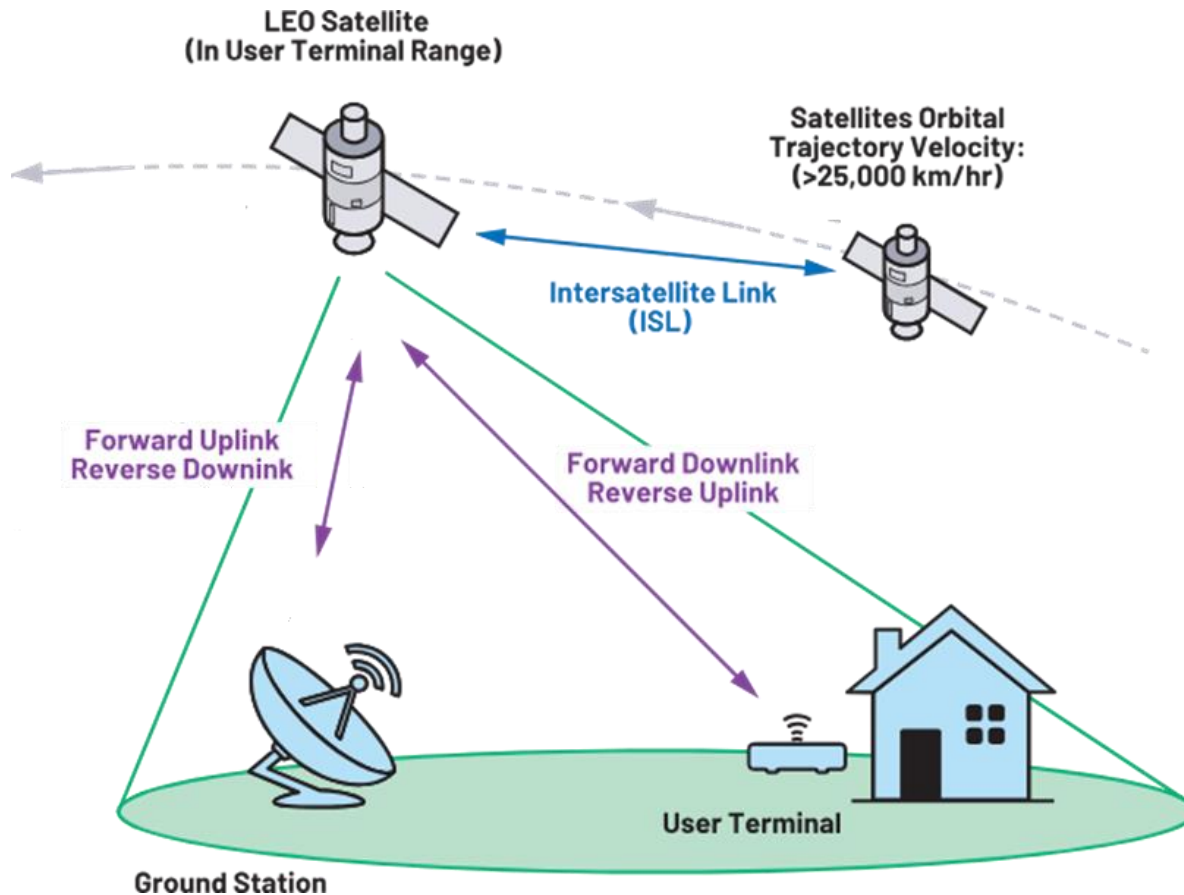
# Agenda

---

- Lab Overview
- Installation
  - VirtualBox
  - Ubuntu
  - NS3
  - LEO module
- Example Code Execution
  - Calculate end-to-end delay
  - Change the data rate and observe the results

# Lab Overview

- In this lab, we are going to write an NS3 program to simulate LEO communications



# Lab Overview

---

- **Limitation of the LEO module:**
  - Constant link data rate without considering path loss and Tx Gain
- **Goal of this lab:**
  - Leverage lab 1 to find the beamforming steering vector and the corresponding Tx gain
  - Read this Tx Gain and calculate the Rx power in NS3
  - Calculate the resulting SNR and data rate
  - Set the link data rate accordingly

# Lab Overview

---

- **Tasks (Week 1):**

- Install Virtual Box
- Install Ubuntu
- Install NS3
- Install LEO module
- Configure and test NS3/LEO module
- Execute the example code (calculate\_delay.cc)
- Modify the link data rate

# Lab Overview

---

- **Tasks (Week 2-3):**
  - Output node locations
  - Execute lab1 (bf.m) to find Tx Gain
  - Read Tx gain and calculate the Rx power in NS3
  - Calculate SNR and data rate
  - Modify the link data rate

# Installation – VirtualBox

---

- VirtualBox version: 7.X
- Ubuntu version: 22.04 LTS
- Minimum hardware requirements for VB
  - CPU: 8-core
  - RAM: 8GB
  - Storage: 30GB
  - Video memory: 128MB
- Notice: You can increase the hardware configuration if needed

# Installation – VirtualBox (1/13)

- Visit the [VirtualBox official download page](#)
- Download the “Platform Packages” for your OS

VirtualBox

Home Download Documentation Community

## Download VirtualBox

The VirtualBox Extension Pack is available for personal and educational use on this page under the PUEL license. The VirtualBox Extension Pack is also available under commercial or enterprise terms. By downloading, you agree to the terms and conditions of the respective license.

### VirtualBox Platform Packages

VirtualBox 7.1.6 platform packages

- Windows hosts
- macOS / Intel hosts
- macOS / Apple Silicon hosts
- Linux distributions
- Solaris hosts
- Solaris 11 IPS hosts

Platform packages are released under the terms of the [GPL version 3](#)

### VirtualBox Extension Pack

VirtualBox 7.1.6 Extension Pack

This VirtualBox Extension Pack Personal Use and Educational License governs your access to and use of the VirtualBox Extension Pack. It does not apply to the VirtualBox base package and/or its source code, which are licensed under version 3 of the GNU General Public License “GPL”.

See our [FAQ](#) for answers to common questions.

#### VirtualBox Extension Pack Personal Use and Educational License (PUEL)

[PUEL License FAQ](#) [PUEL License Text](#) [Accept and download](#)



# Installation – VirtualBox (2/13)

- Locate the downloaded installer (.exe file)
- Run the installer to finish the installation



# Installation – VirtualBox (3/13)

- Download image file (.iso)
  - Visit the [Ubuntu official download page](#)
  - Select Ubuntu Desktop 22.04 LTS

ubuntu<sup>®</sup> releases

Ubuntu 22.04.5 LTS (Jammy Jellyfish)

## Select an image

Ubuntu is distributed on three types of images described below.

### Desktop image

The desktop image allows you to try Ubuntu without changing your computer at all, and at your option to install it permanently later. This type of image is what most people will want to use. You will need at least 1024MiB of RAM to install from this image.

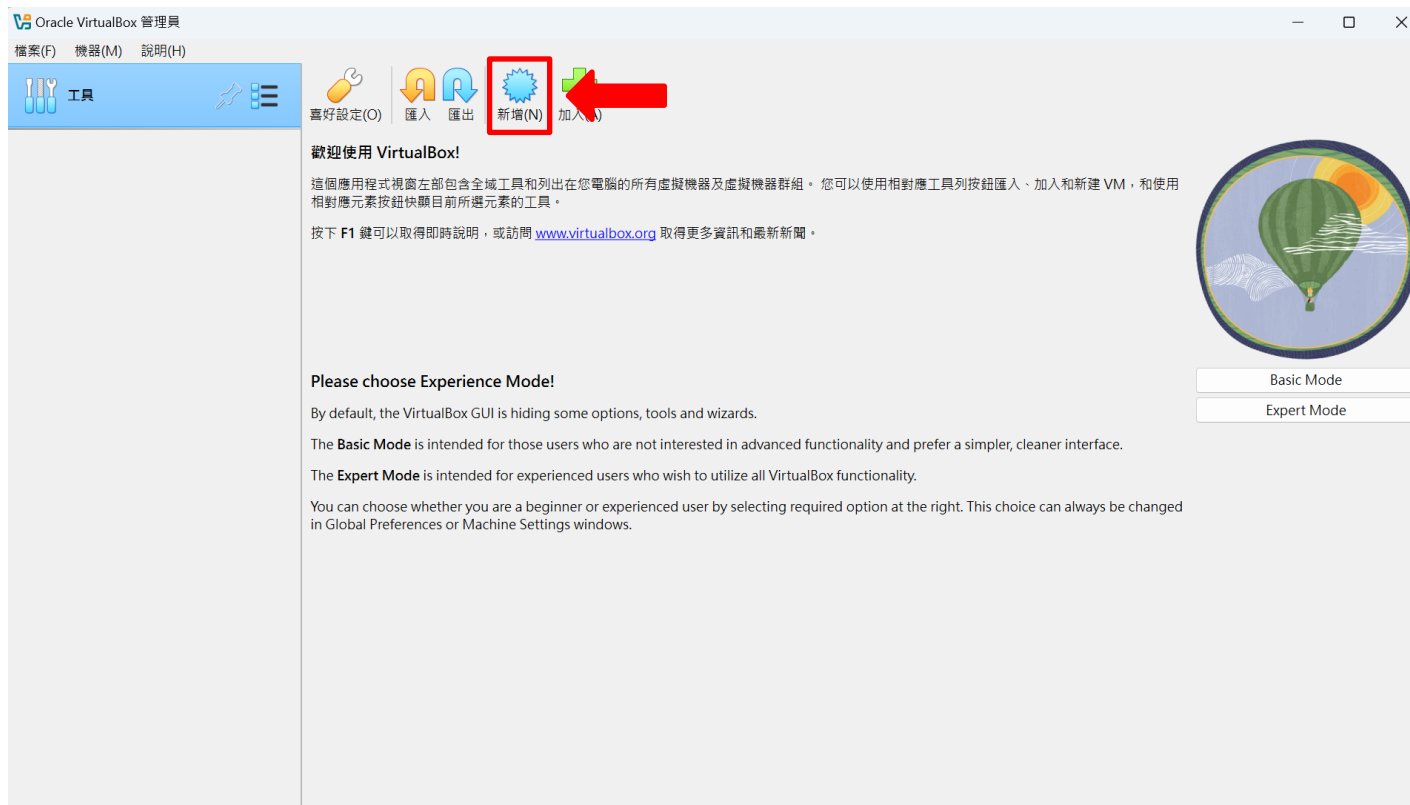
**64-bit PC (AMD64) desktop image**

Choose this if you have a computer based on the AMD64 or EM64T architecture (e.g., Athlon64, Opteron, EM64T Xeon, Core 2). Choose this if you are at all unsure.



# Installation – VirtualBox (4/13)

- Start VirtualBox and click “New”



# Installation – VirtualBox (5/13)

- Name the virtual machine
- Select the correct “Type”, “Subtype”, and “Version”

建立虛擬機器

### 虛擬機器名稱和作業系統

請為新的虛擬機器選擇描述性名稱和目的地資料夾。您選擇的名稱將在整個 VirtualBox 中使用，以識別這部電腦。此外，您可以選取用於安裝客體作業系統的 ISO 映像。

名稱(N): **Lab2** ← Change the name if you want ✓

資料夾(F): C:\Users\Annie\VirtualBox VMs

ISO 映像(I): <未選取> ✓

版本(V):

類型(T): **Linux** x64

Subtype: **Ubuntu**

版本(V): **Ubuntu (64-bit)**

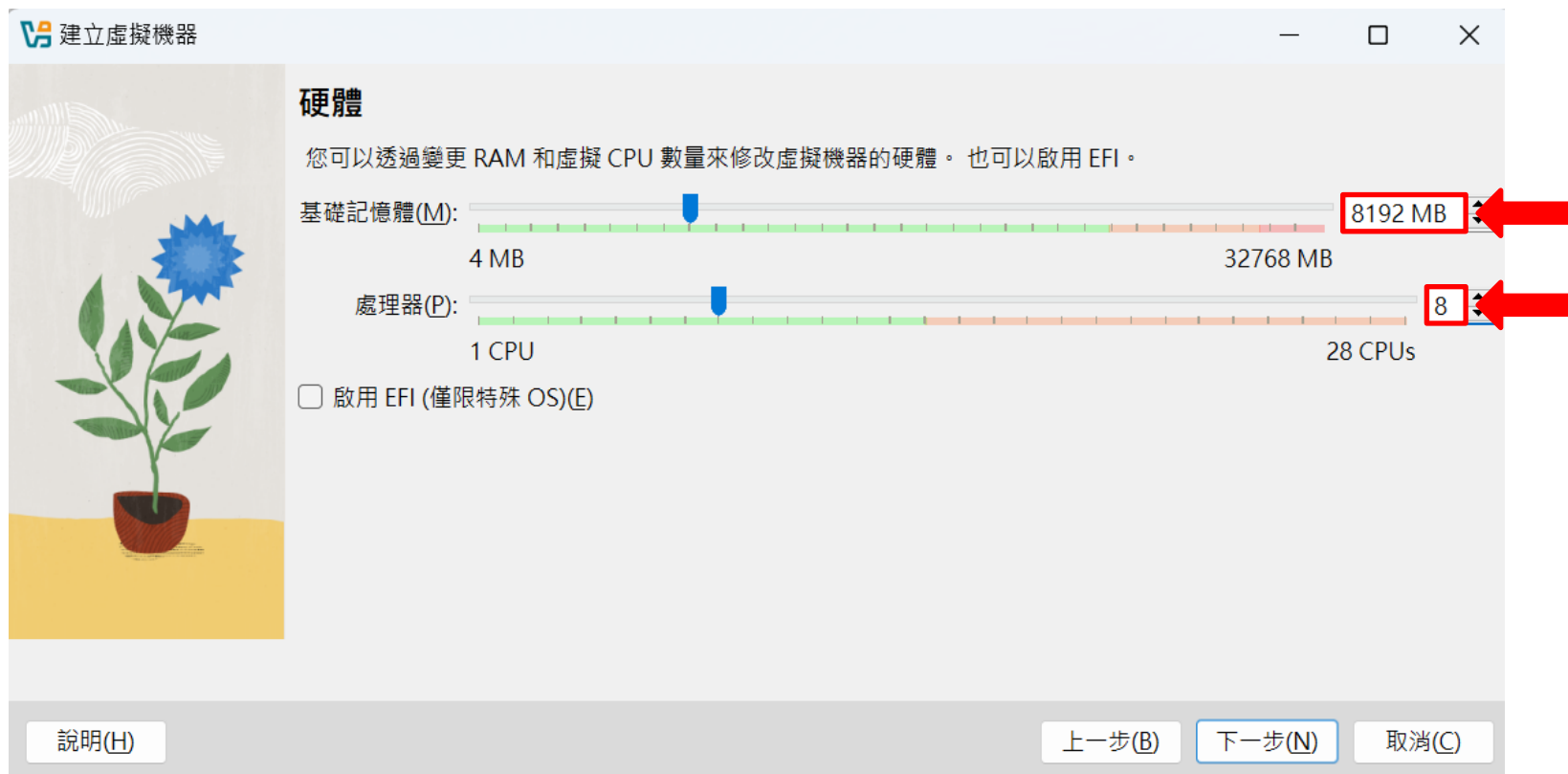
☐ 略過無人值守安裝(S)

未選取 ISO 映像，將需要手動安裝客體作業系統。

說明(H) 上一步(B) 下一步(N) 取消(C)

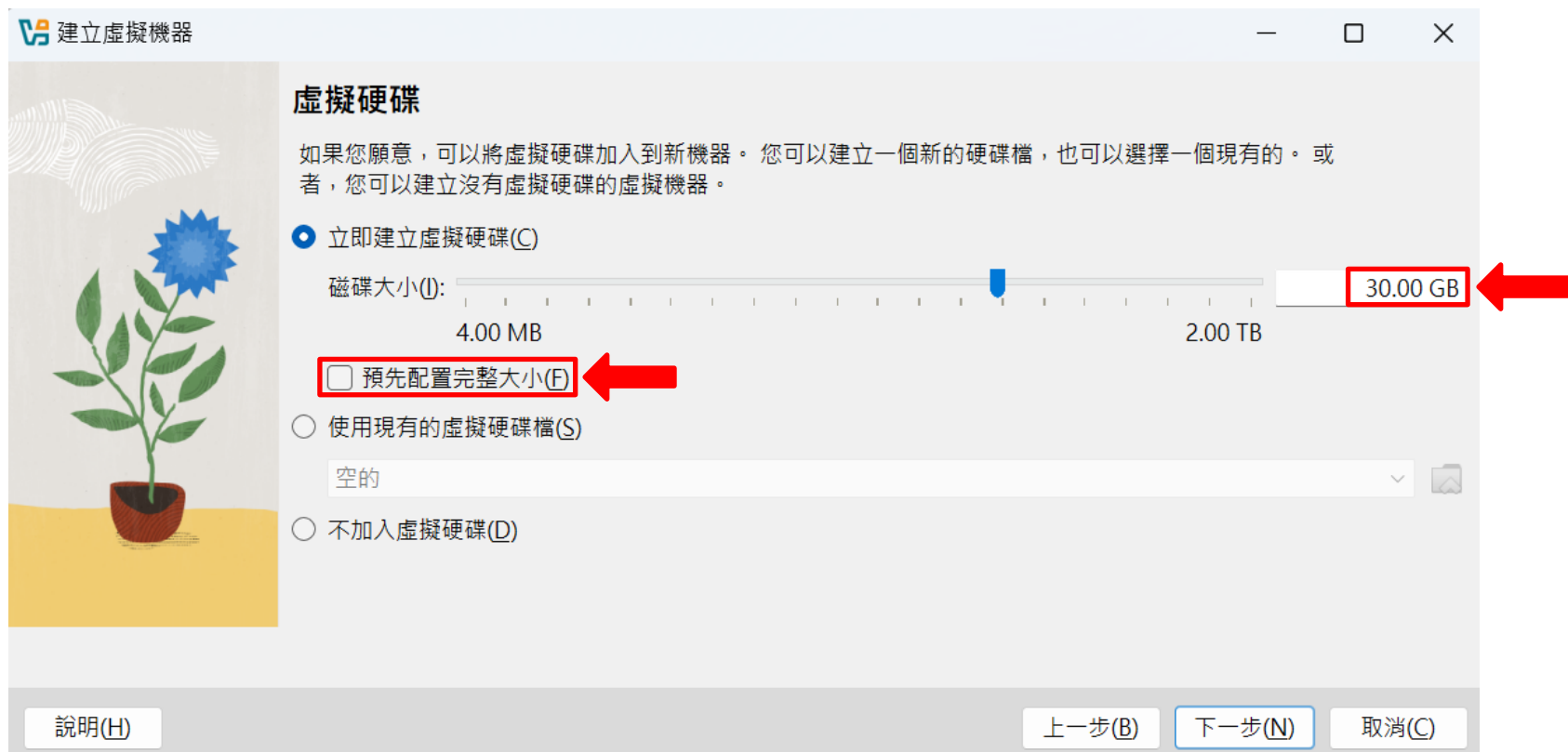
# Installation – VirtualBox (6/13)

- Set the memory size to 8192MB
- Allocate 8 CPU cores



# Installation – VirtualBox (7/13)

- Set the virtual hard disk size to 30GB



# Installation – VirtualBox (8/13)

- Check the final settings and click “Finish”



# Installation – VirtualBox (9/13)

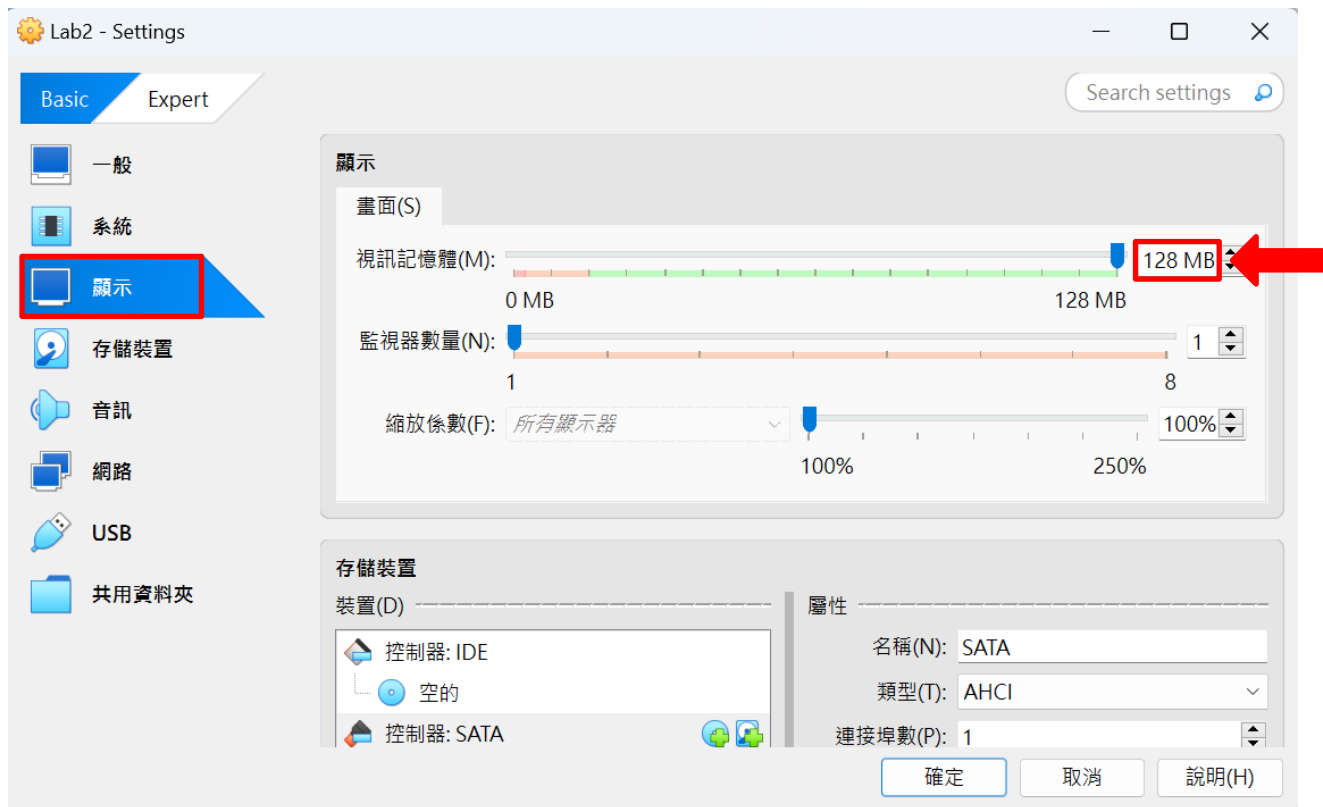
- Open “Settings”





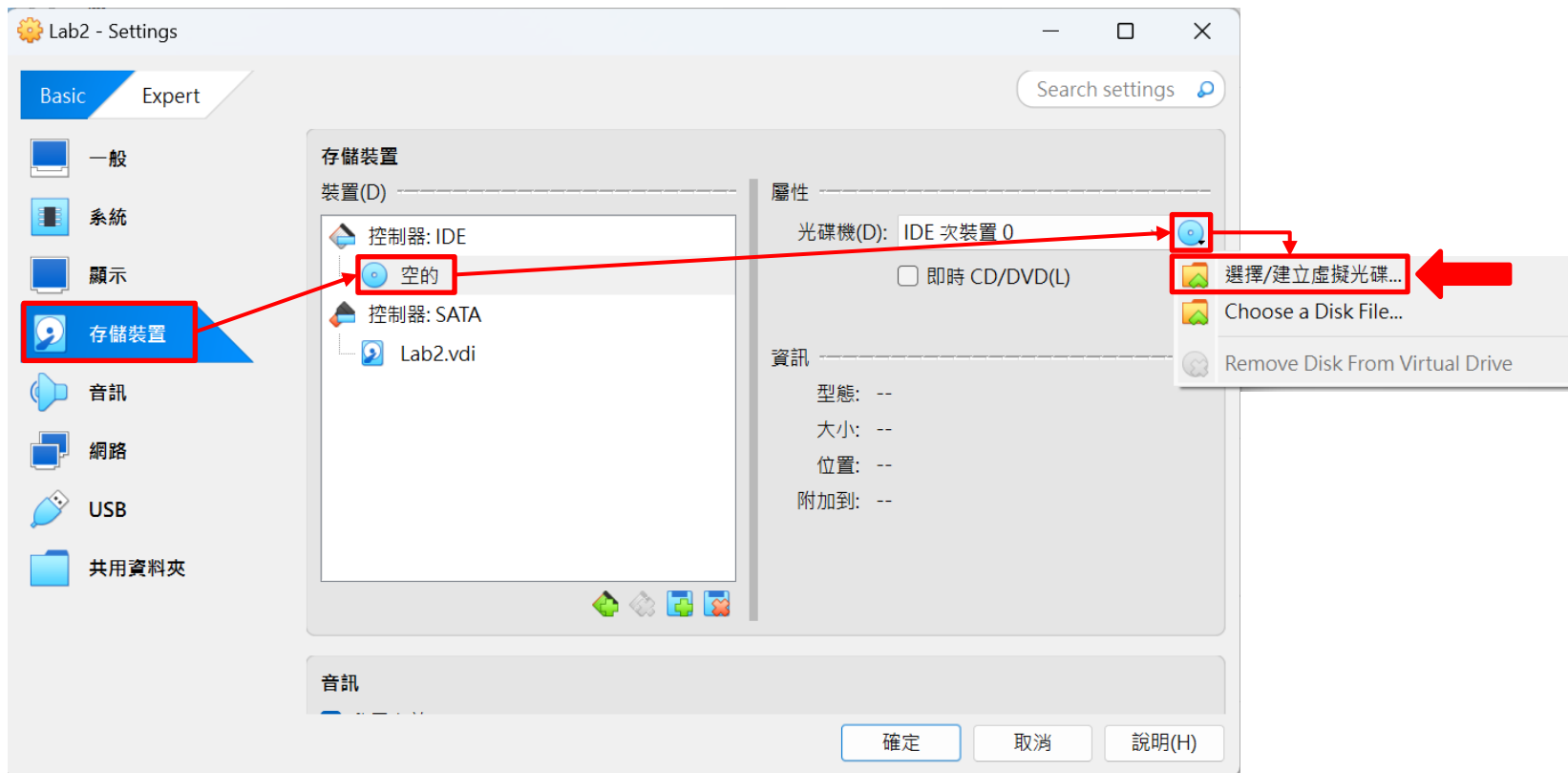
# Installation – VirtualBox (10/13)

- Go to Display > Screen > Video Memory
- Set the video memory size to **128MB**



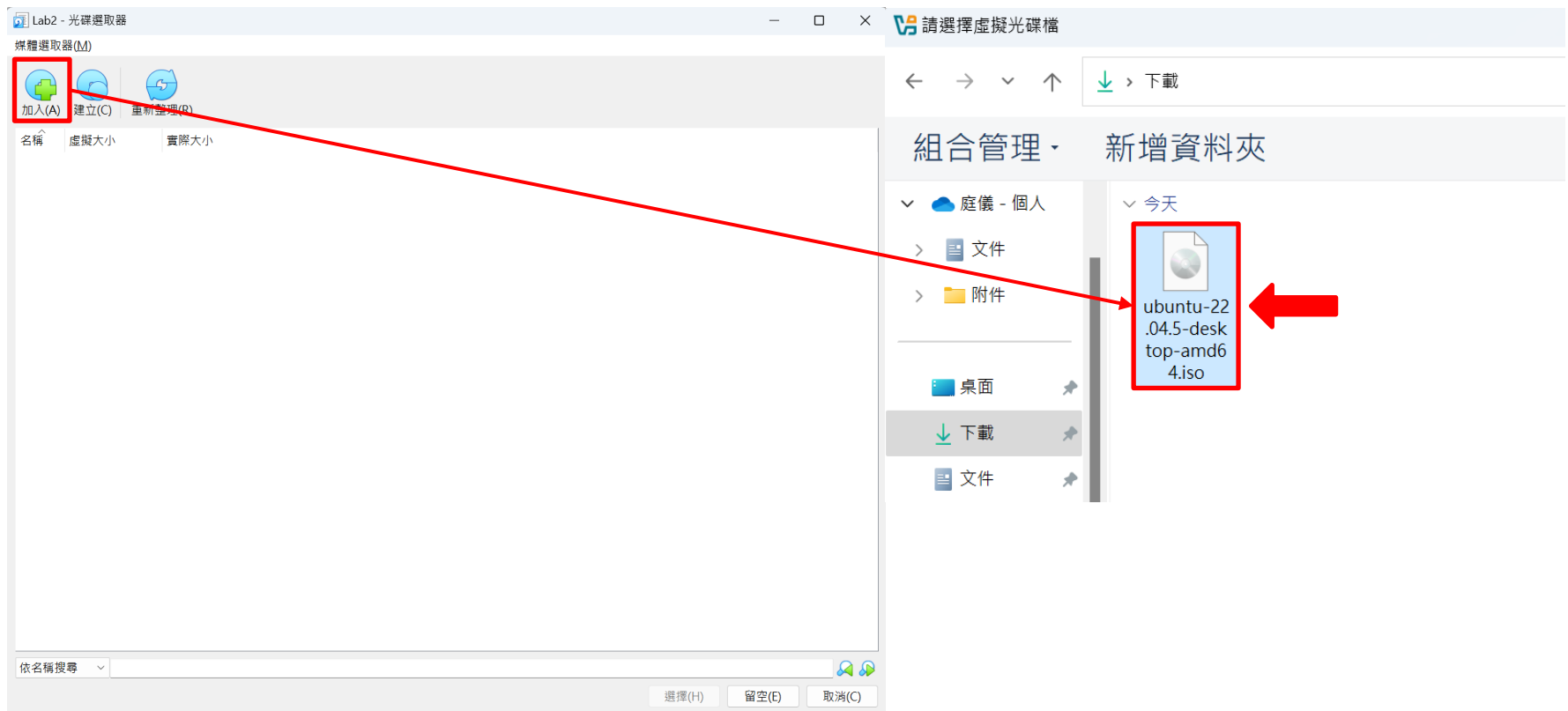
# Installation – VirtualBox (11/13)

- Go to Storage > Controller: IDE
- Create a virtual optical disk



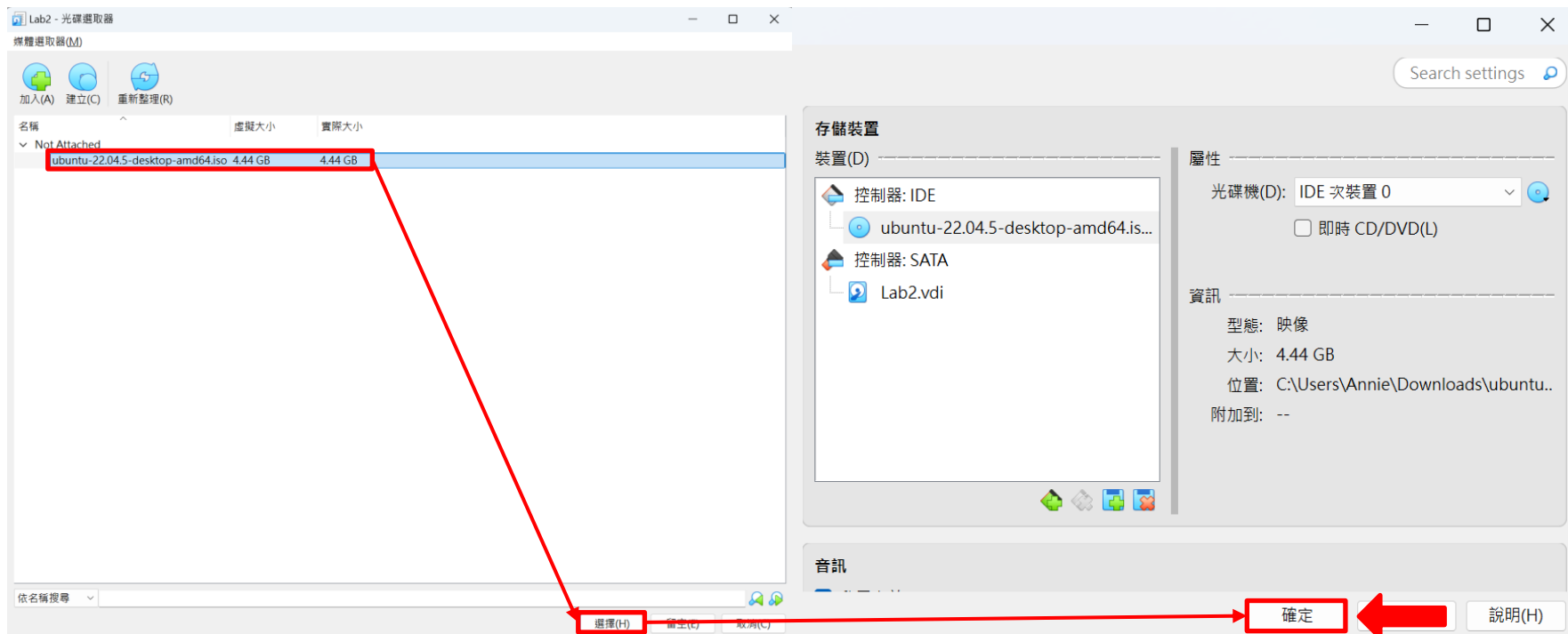
# Installation – VirtualBox (12/13)

- Click “Add” and select the downloaded image file (.iso)



# Installation – VirtualBox (13/13)

- Click “Choose”
- Review the final settings and click “OK”



# Installation – Ubuntu (1/12)

- Click “Start”



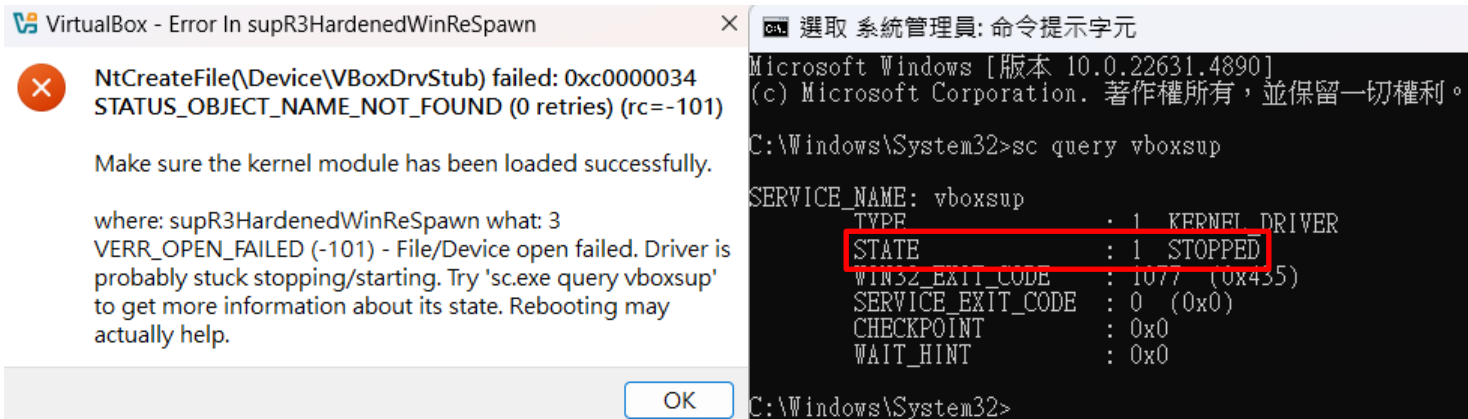
# VirtualBox Start Error

- If an error appears, refer to the solutions below; otherwise, skip this part
  - Open CMD with administrator privileges
  - Run this command

```
$ sc query vboxsup
```

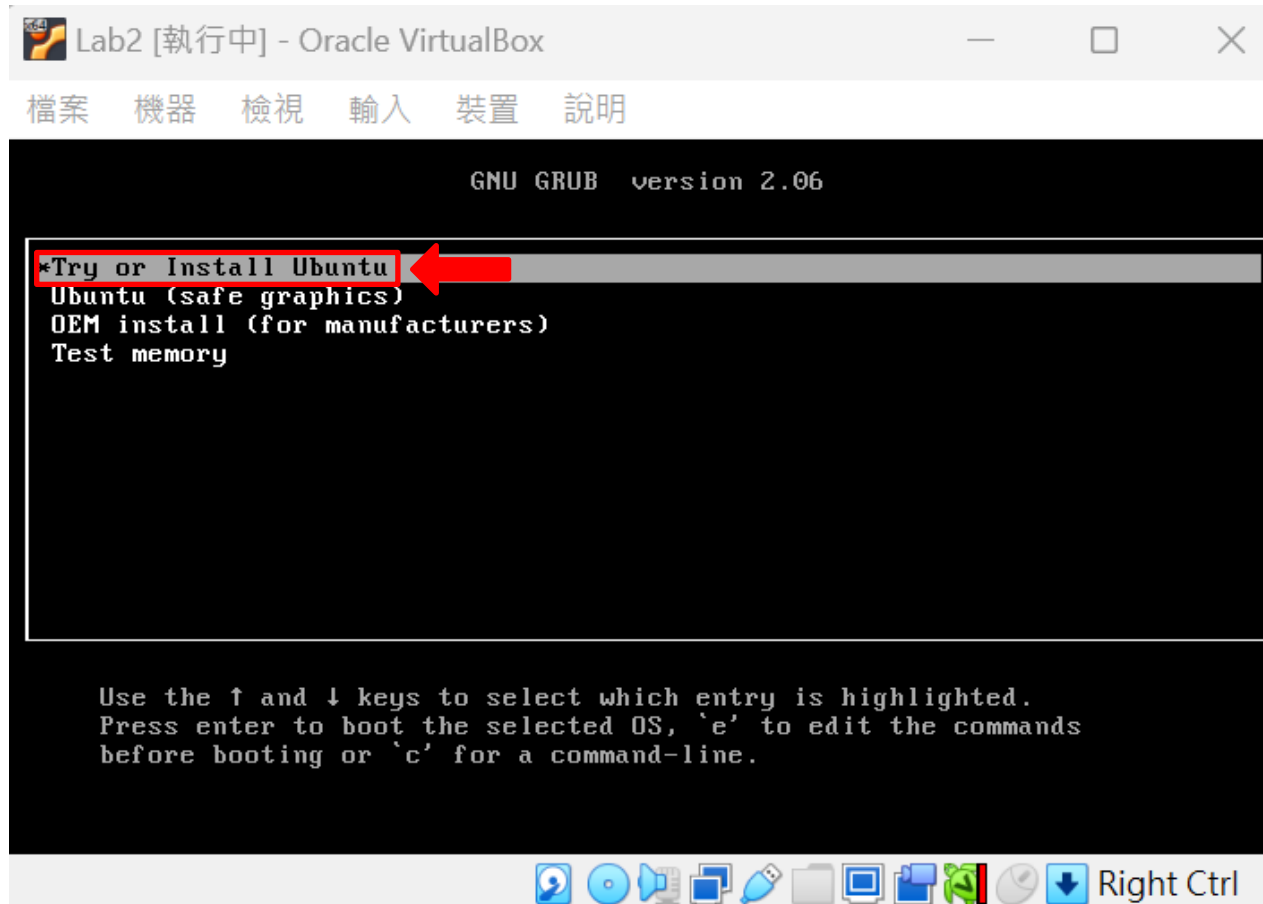
- If “STATE: STOPPED” appears, run this command

```
$ sc start vboxsup
```



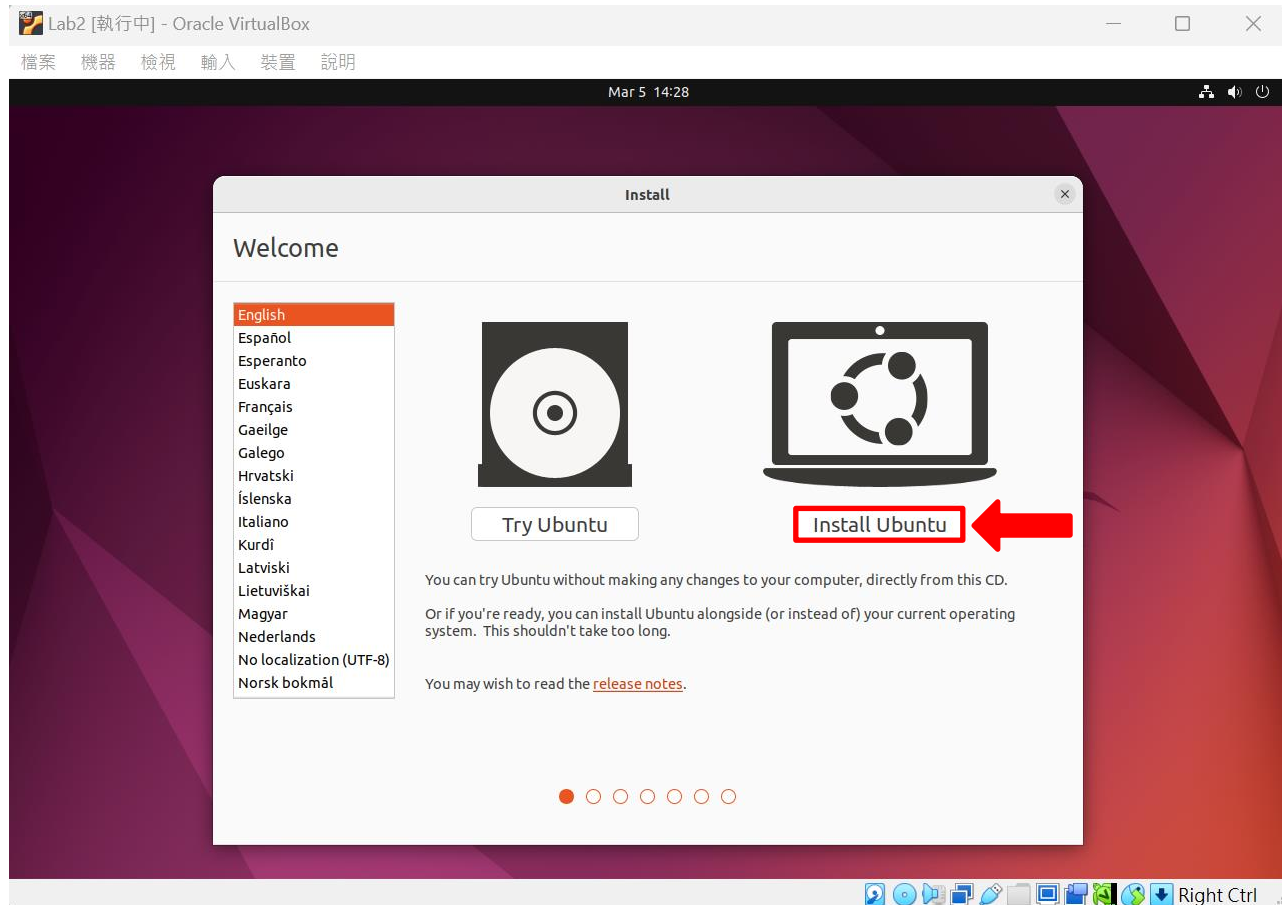
# Installation – Ubuntu (2/12)

- Select “Try or install Ubuntu” and then press “Enter” bottom



# Installation – Ubuntu (3/12)

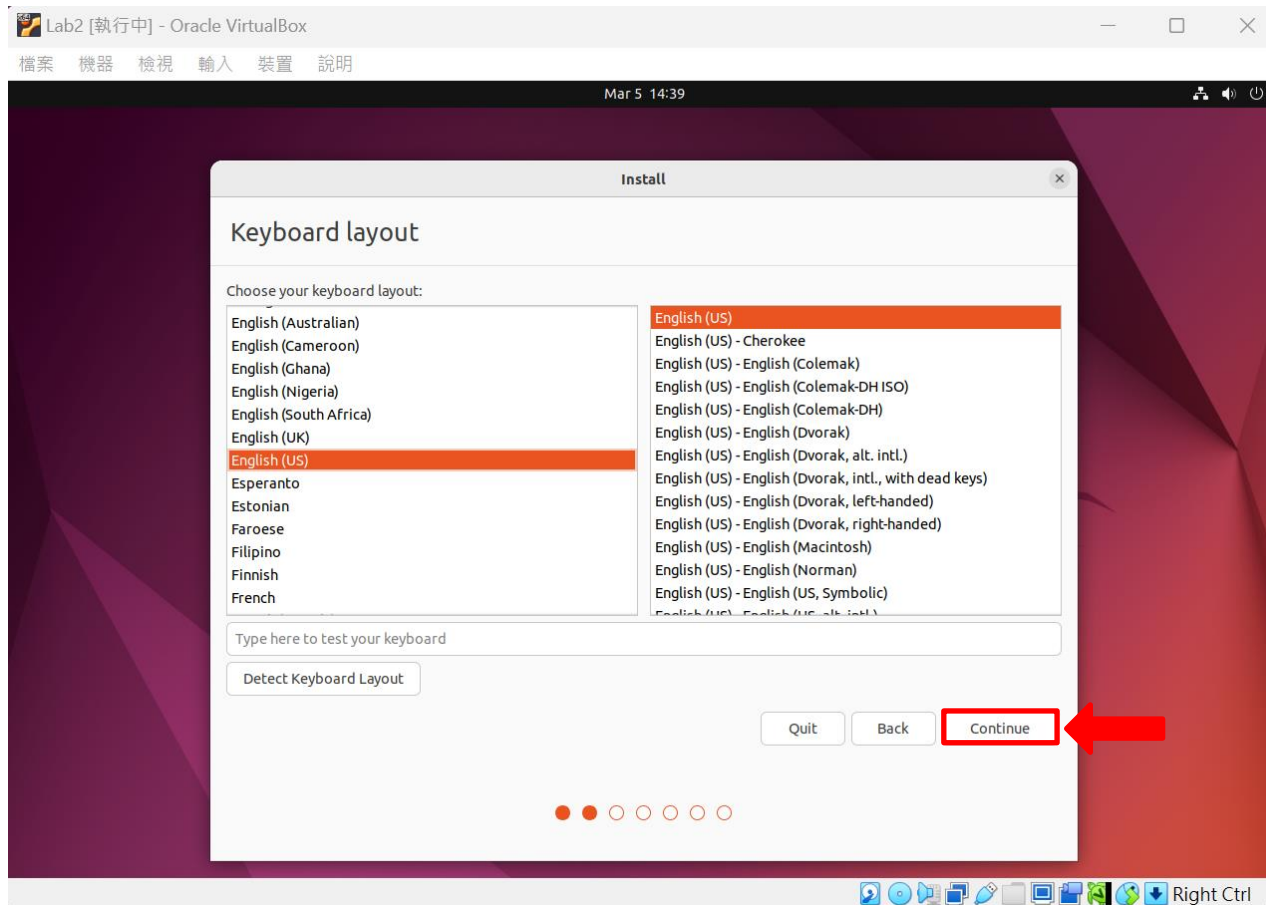
- Click “Install Ubuntu”





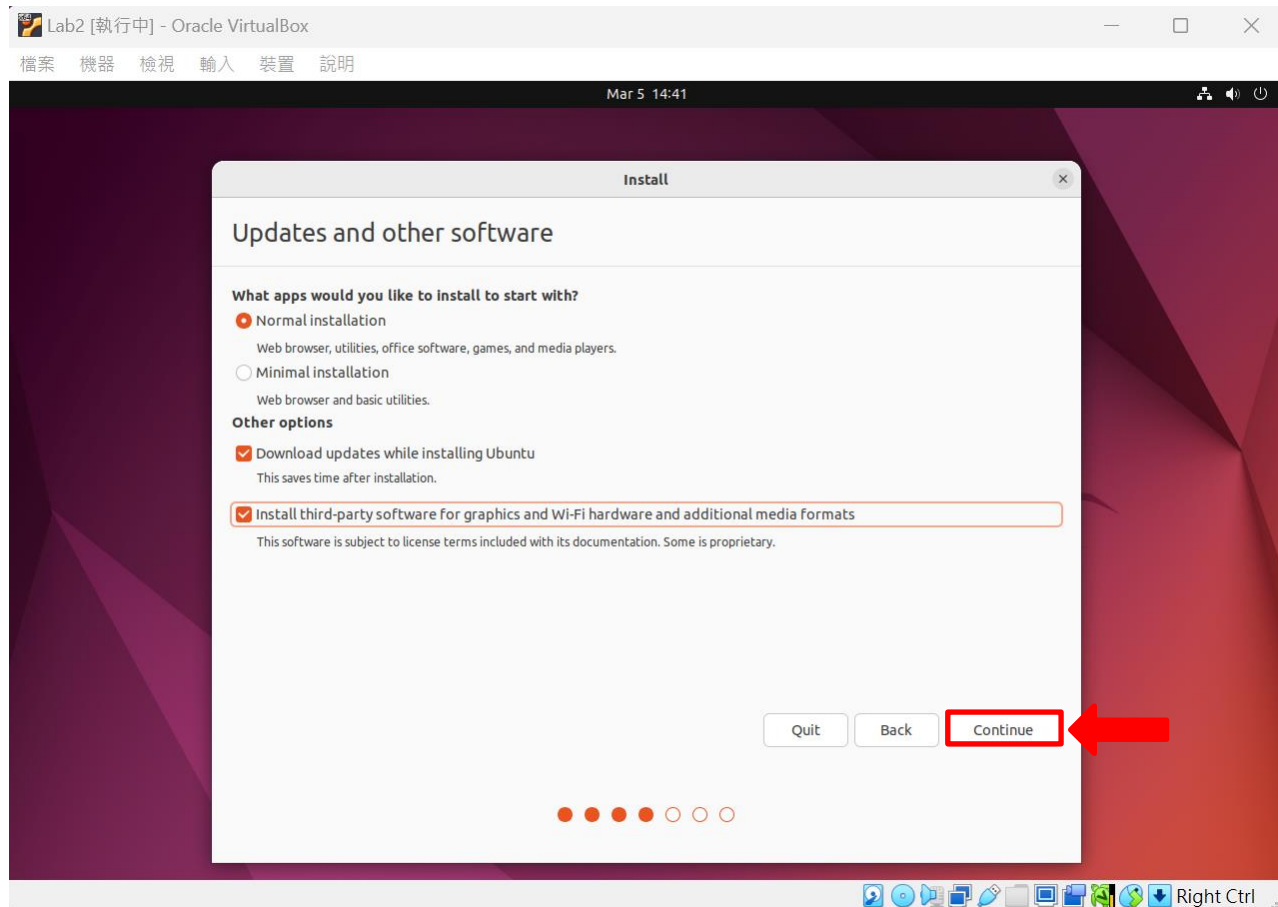
# Installation – Ubuntu (4/12)

- Set according to the image



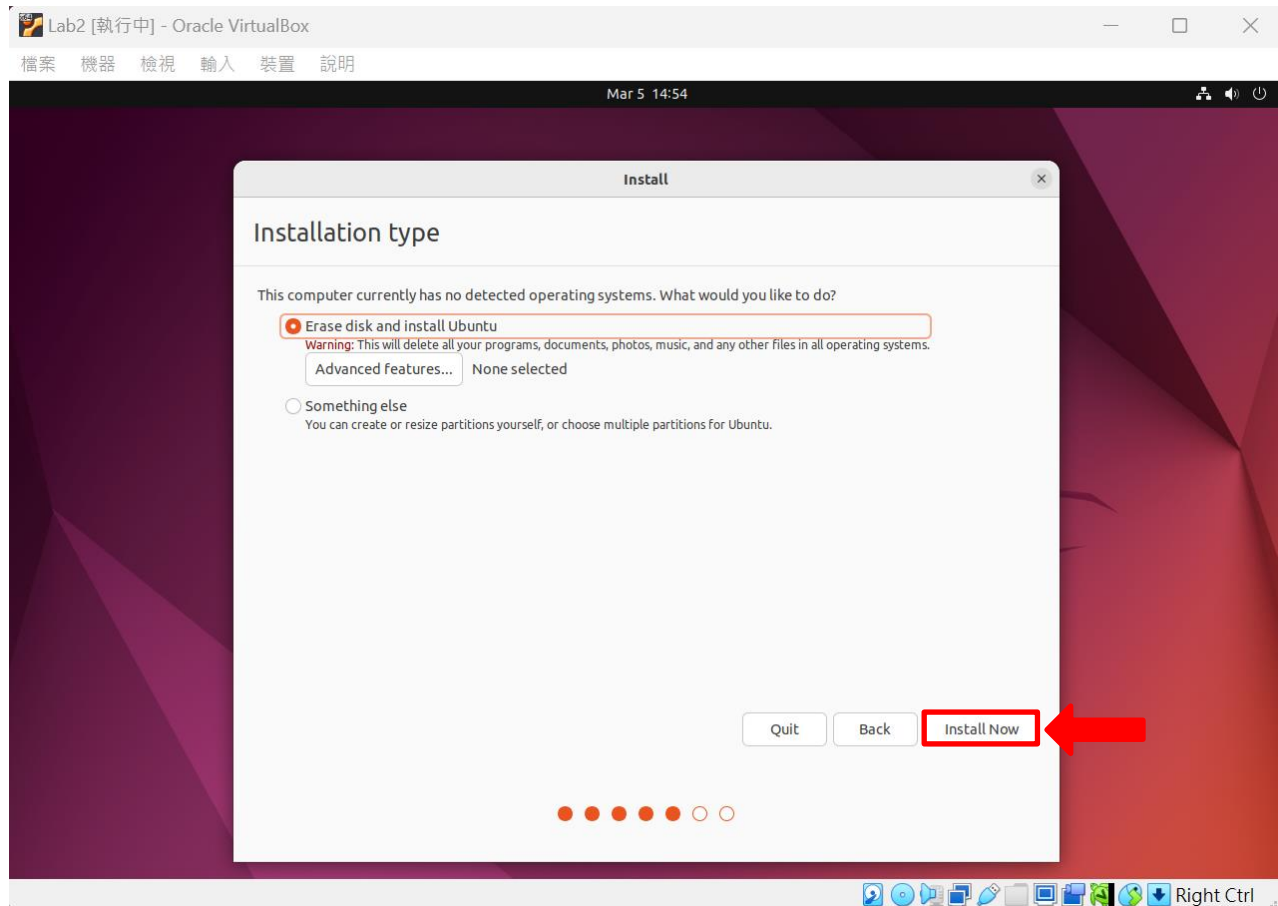
# Installation – Ubuntu (5/12)

- Set according to the image



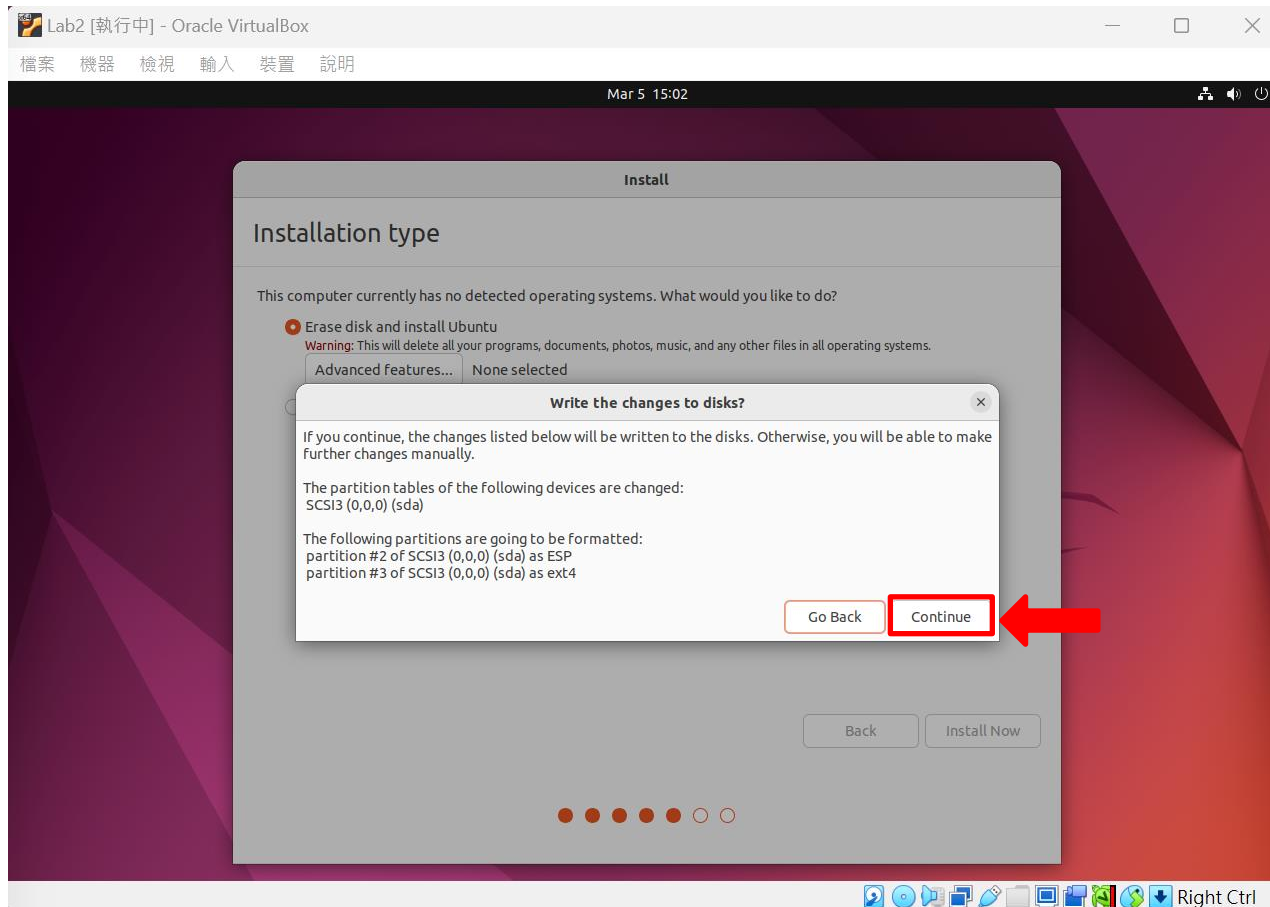
# Installation – Ubuntu (6/12)

- Set according to the image



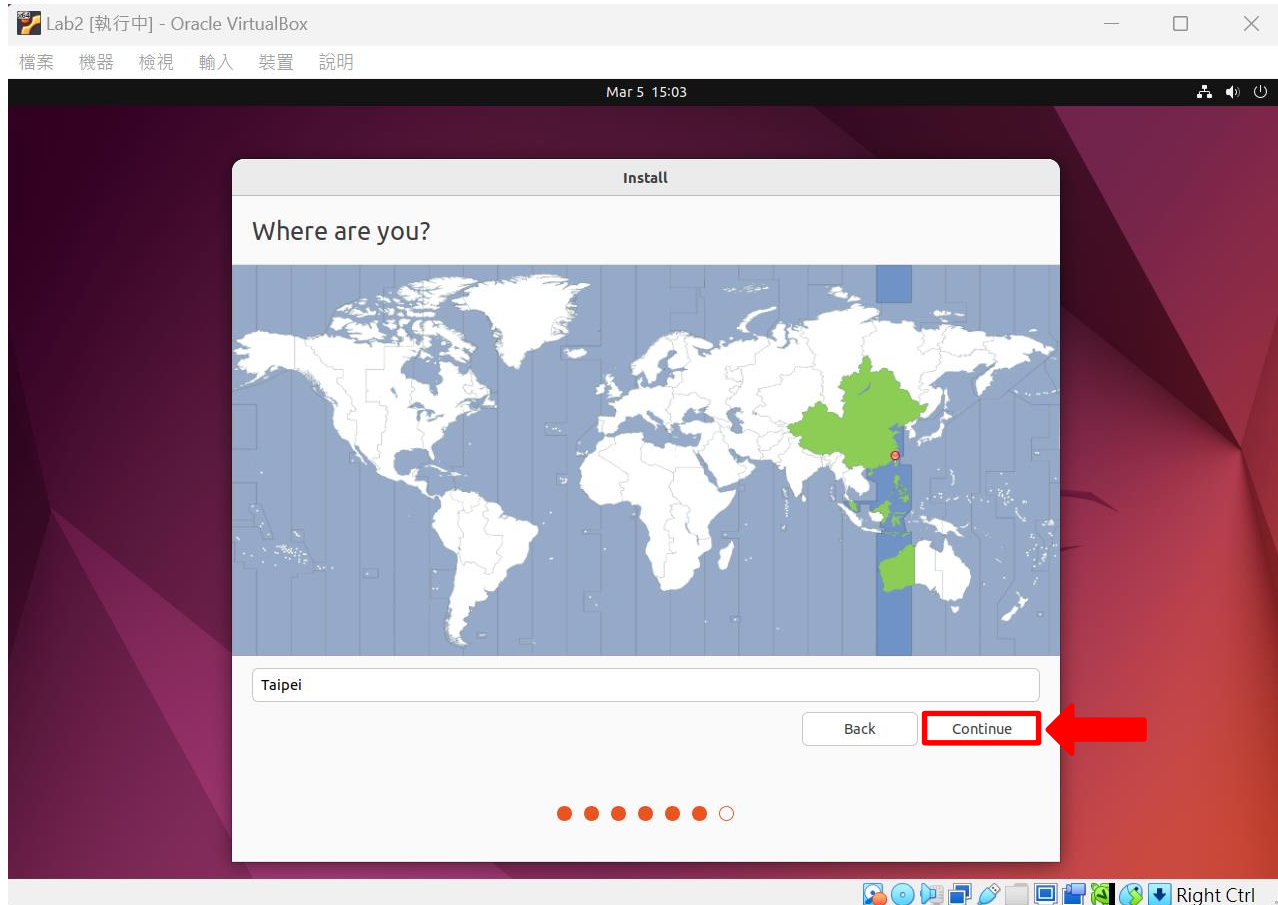
# Installation – Ubuntu (7/12)

- Set according to the image



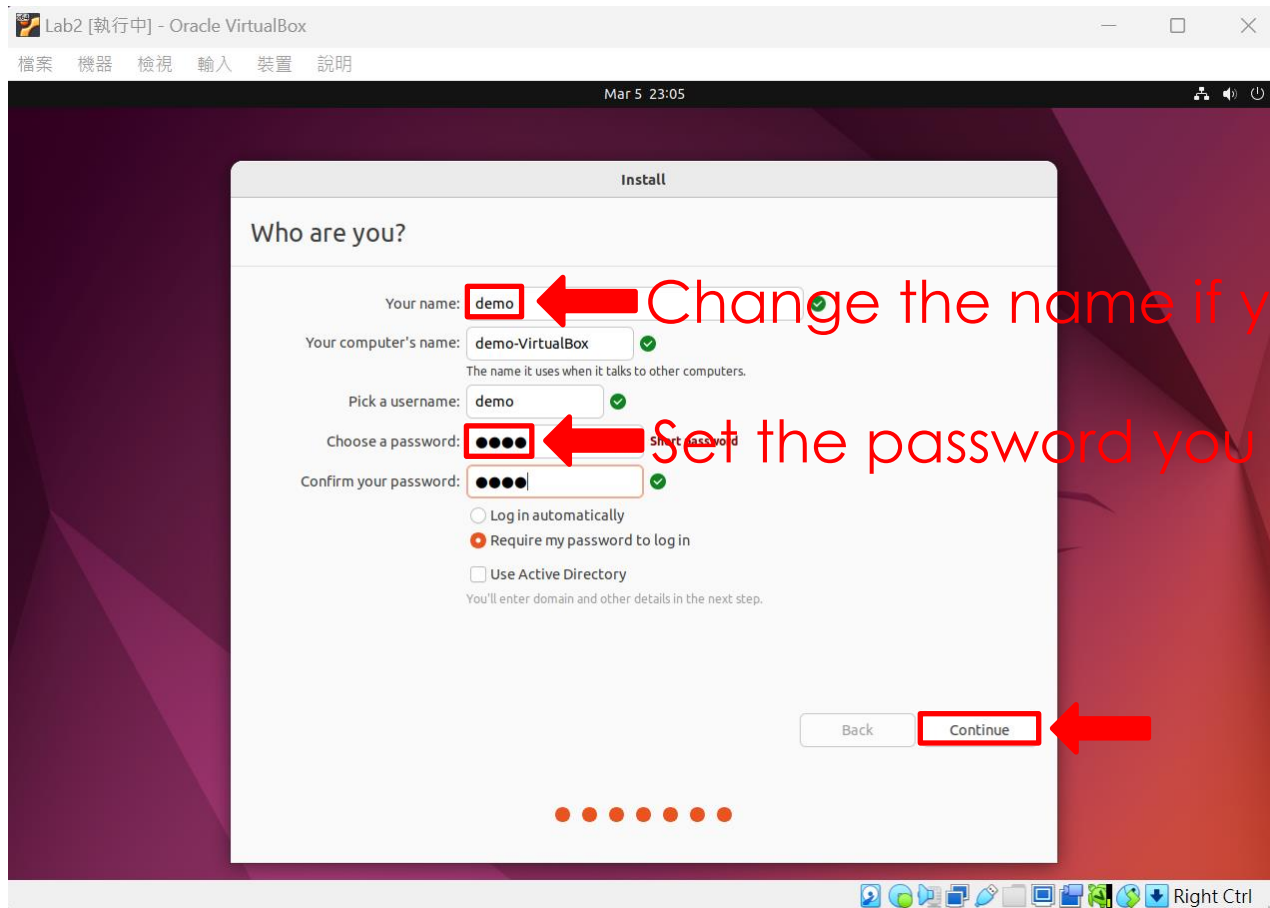
# Installation – Ubuntu (8/12)

- Set according to the image



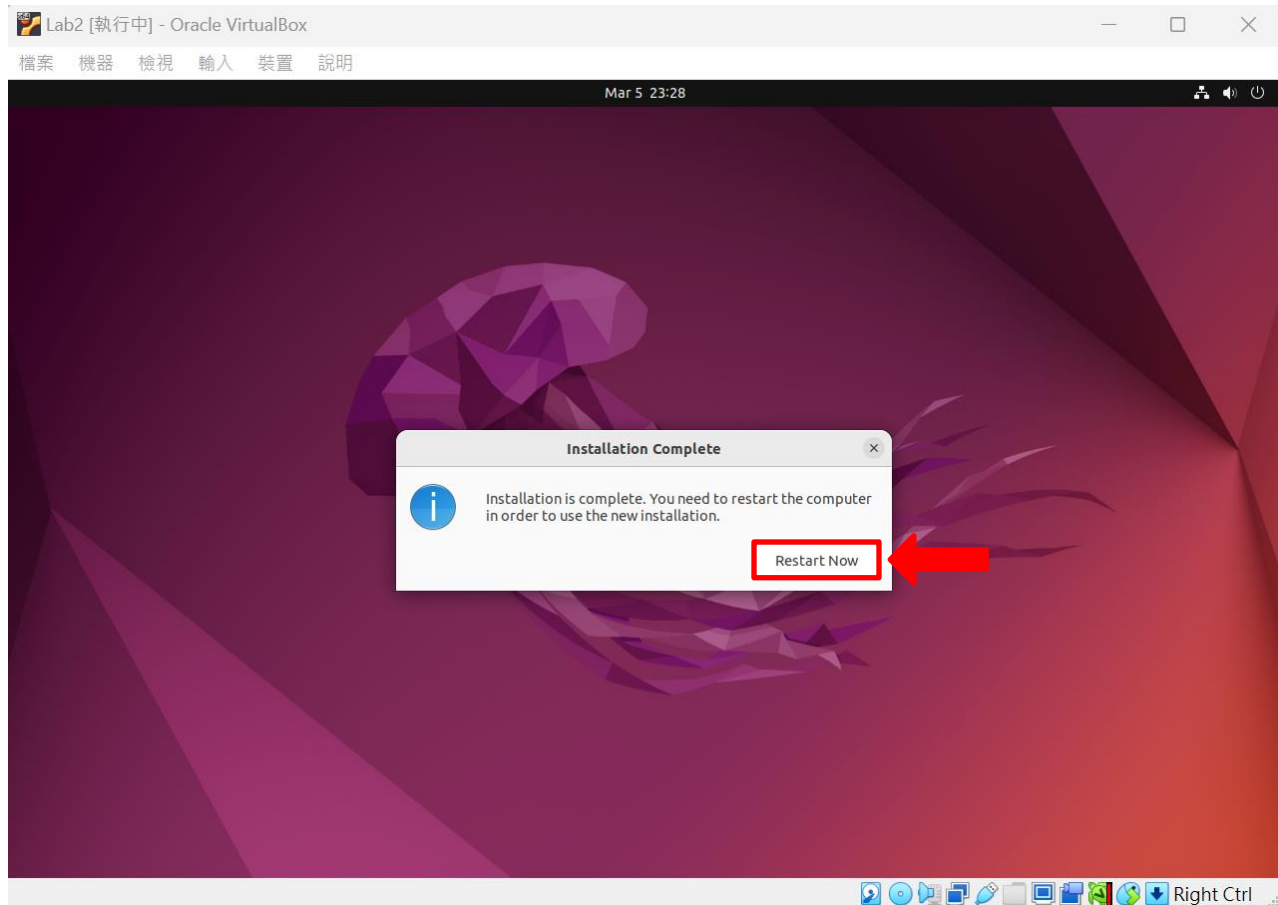
# Installation – Ubuntu (9/12)

- Set according to the image



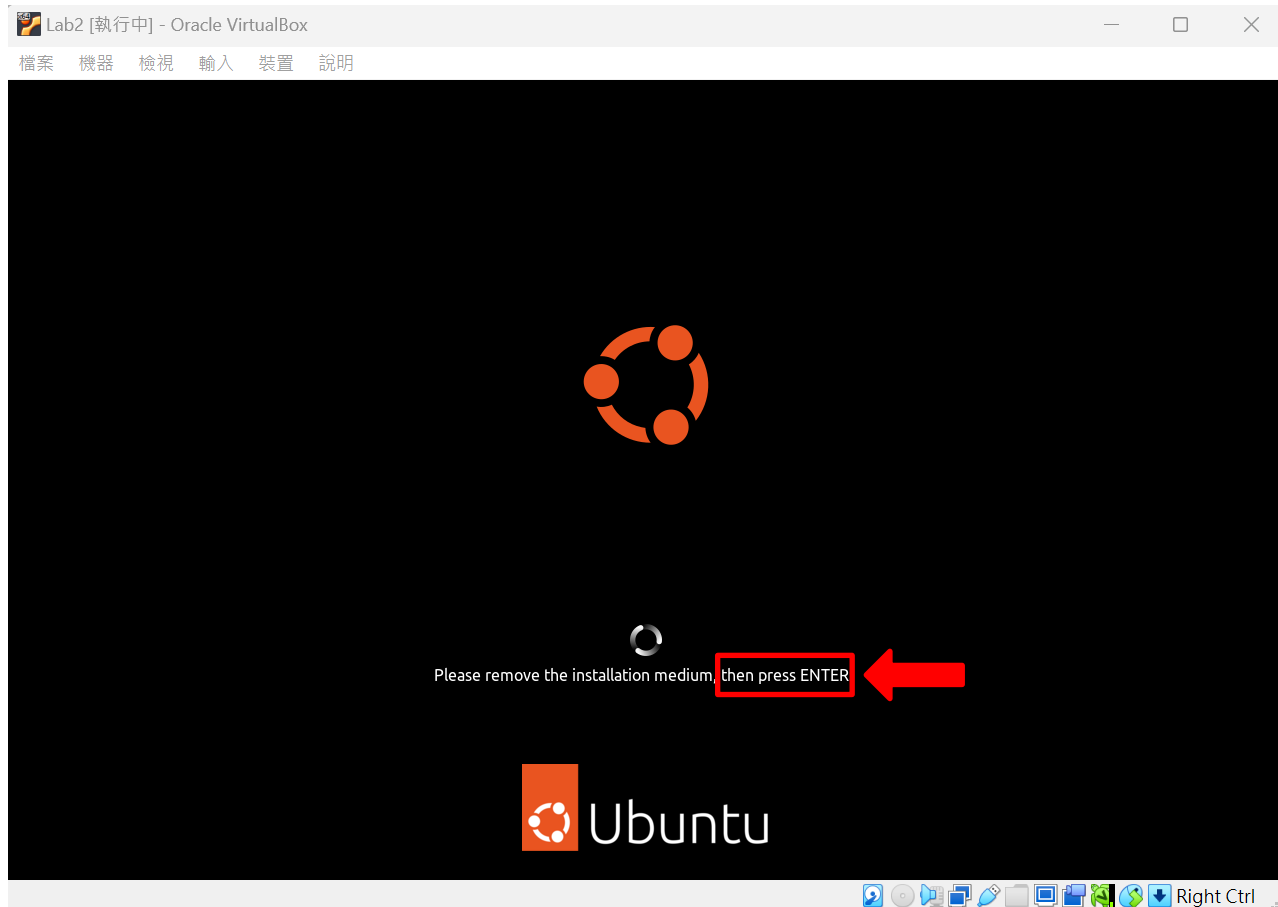
# Installation – Ubuntu (10/12)

- Click “Restart Now”



# Installation – Ubuntu (11/12)

- Press “Enter” to restart the VM

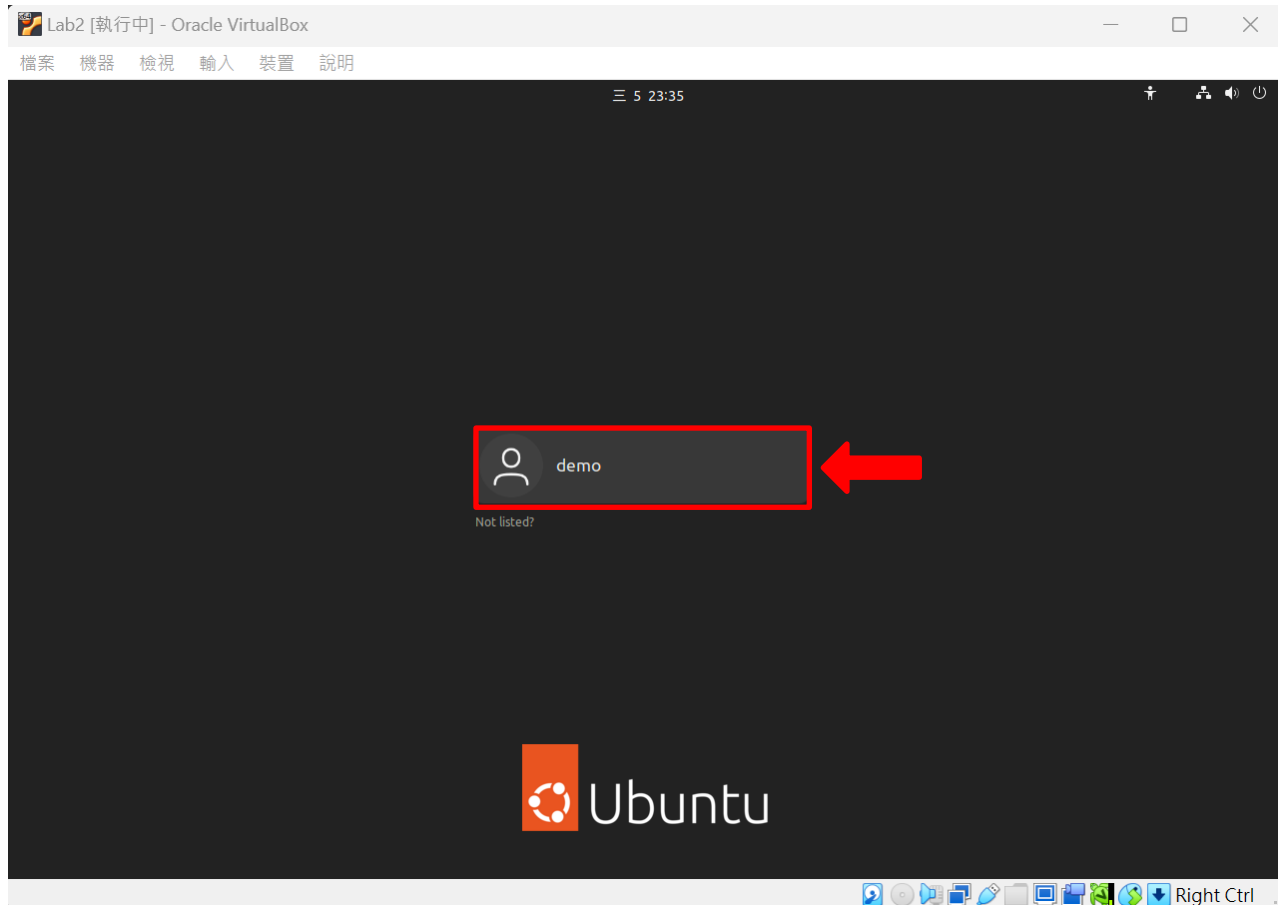




# Installation – Ubuntu (12/12)

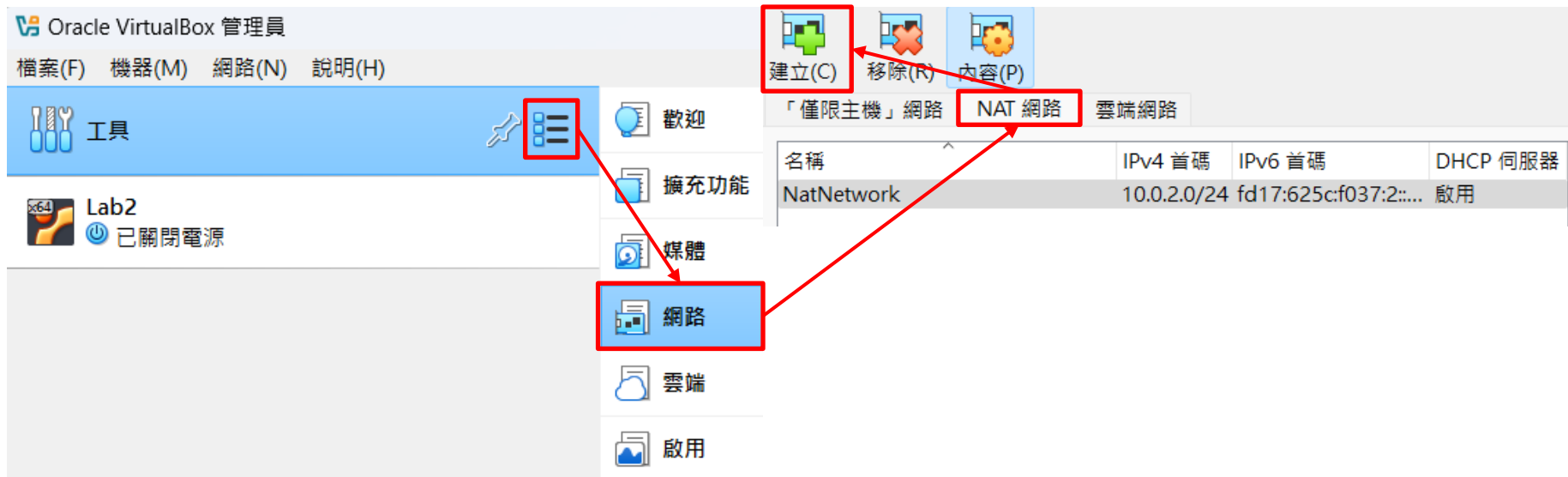
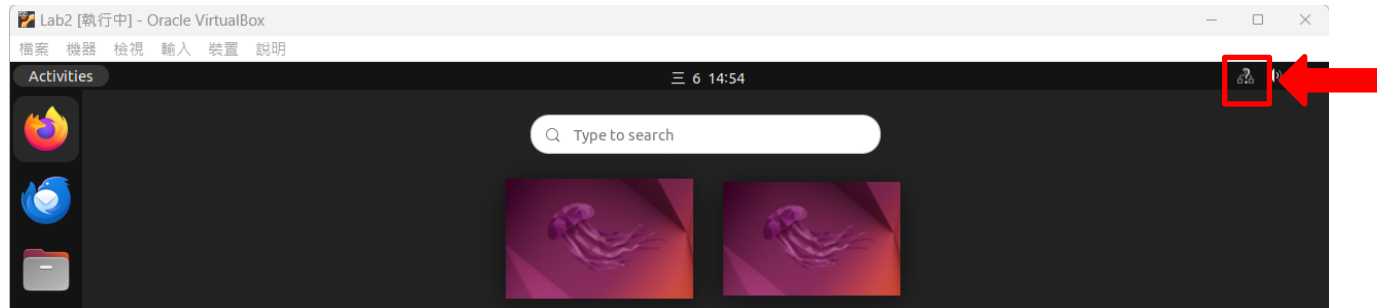
---

- Login your account



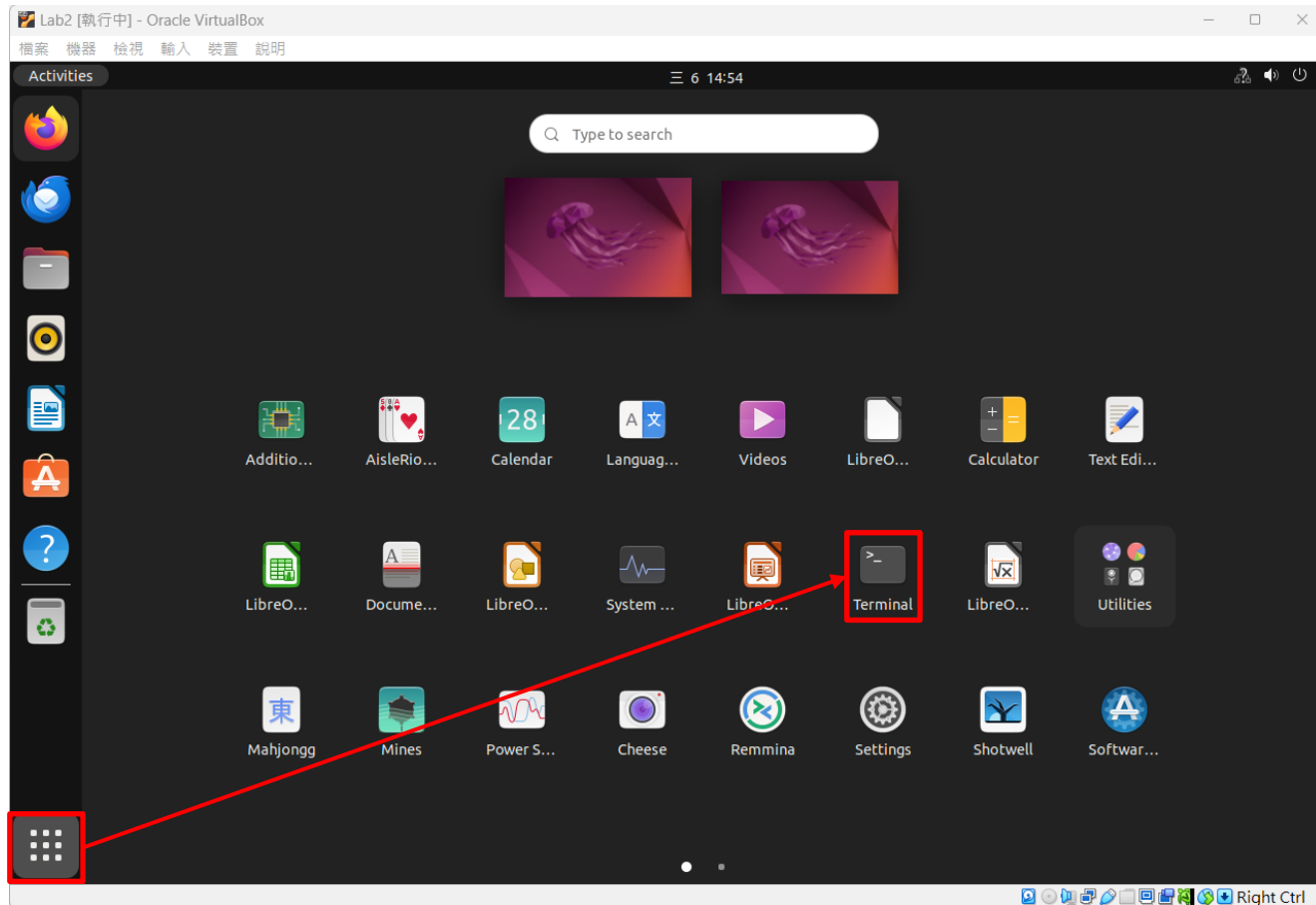
# Ubuntu Network Error

- If you have no internet, refer to the solutions [here](#); otherwise, skip this part



# Installation – NS3 (1/4)

- Open terminal



# Installation – NS3 (2/4)

---

- Update the system up to date

```
$ sudo apt update && sudo apt upgrade -y
```

- Install git

```
$ sudo apt install git -y
```

- Install GCC

```
$ sudo apt install gcc -y
```


- Install G++

```
$ sudo apt install build-essential -y
```

# Installation – NS3 (3/4)

- Install NS3 (3.35 version)
  - Make sure to install mpic++ package
  - [Official download website](#)
- Configure the build

```
$ ./waf configure --enable-examples --enable-tests  
--enable-mpi --disable-werror
```



```
---- Summary of optional NS-3 features:  
Build profile           : debug  
Build directory        :  
BRITe Integration      : not enabled (BRITe not enabled (see option --with-brite))  
DES Metrics event collection : not enabled (defaults to disabled)  
DPDK NetDevice         : not enabled (libdpdk not found, $RTE_SDK and/or $RTE_TARGET environment variable not set or incorrect)  
Emulation FdNetDevice  : enabled  
Examples               : enabled  
File descriptor NetDevice : enabled  
GNU Scientific Library (GSL) : not enabled (GSL not found)  
Gcrypt library         : not enabled (libgcrypt not found: you can use libgcrypt-config to find its location.)  
GtkConfigStore         : not enabled (library 'gtk+-3 >= 3.22' not found)  
MPI Support            : enabled  
NS-3 Click Integration : not enabled (nsclick not enabled (see option --with-nsclick))  
NS-3 OpenFlow Integration : not enabled (OpenFlow not enabled (see option --with-openflow))  
Netmap emulation FdNetDevice : not enabled (needs net/netmap_user.h)  
Network Simulation Cradle : not enabled (NSC not found (see option --with-nsc))  
PlanetLab FdNetDevice   : not enabled (PlanetLab operating system not detected (see option --force-planetlab))  
PyViz visualizer        : not enabled (Python Bindings are needed but not enabled)  
Python Bindings         : not enabled (The python found version is too low (2.3 required))  
Real Time Simulator     : enabled  
SQLite stats support    : not enabled (library 'sqlite3' and/or semaphore.h not found)  
Tap Bridge              : enabled  
Tap FdNetDevice         : enabled  
Tests                   : enabled  
Threading Primitives    : enabled  
Use sudo to set suid bit : not enabled (option --enable-sudo not selected)  
Xml2                    : not enabled (library 'libxml-2.0 >= 2.7' not found)  
'configure' finished successfully (1.098s)  
demo@demo-virtual-machine:~/ns-3-click/~/ns-3.35$
```

# Installation – NS3 (4/4)

- Build the NS3 module libraries and executables

```
$ ./waf build
```

```
'build' finished successfully (3m46.229s)
```

Modules built:

antenna	aodv	applications
bridge	buildings	config-store
core	csma	csma-layout
dsdv	dsrc	energy
fd-net-device	flow-monitor	internet
internet-apps	lr-wpan	lte
mesh	mobility	mpi
netanim	network	nix-vector-routing
olsr	point-to-point	point-to-point-layout
propagation	sixlowpan	spectrum
stats	tap-bridge	test (no Python)
topology-read	traffic-control	uan
virtual-net-device	wave	wifi
wimax		

Modules not built (see ns-3 tutorial for explanation):

brite	click	dpdk-net-device
openflow	visualizer	

```
demo@demo-VirtualBox:~/ns-3-allinone/ns-3.35$
```

# LEO Module Overview

---

- LEO module
  - [Official download website](#)
- Original LEO module exists some bugs
  - Redundant installation (epidemic)
  - Wrong latitude calculation
  - Wrong delay calculation

# Installation – LEO Module (1/2)

---

- Install LEO module

```
$ cd contrib/  
$ git clone https://github.com/NYCU-NETCAP2025/lab2-  
<GITHUB_ID>.git leo
```

- Config and build NS3 again

```
$ cd ..  
$ ./waf configure --enable-examples --enable-tests --  
enable-mpi --disable-werror  
$ ./waf build
```



# Installation – LEO Module (2/2)

- Execute example code

```
$ ./waf --run=leo-bulk-send
```

- Check the results are the same as below
  - Segmentation fault is normal :)

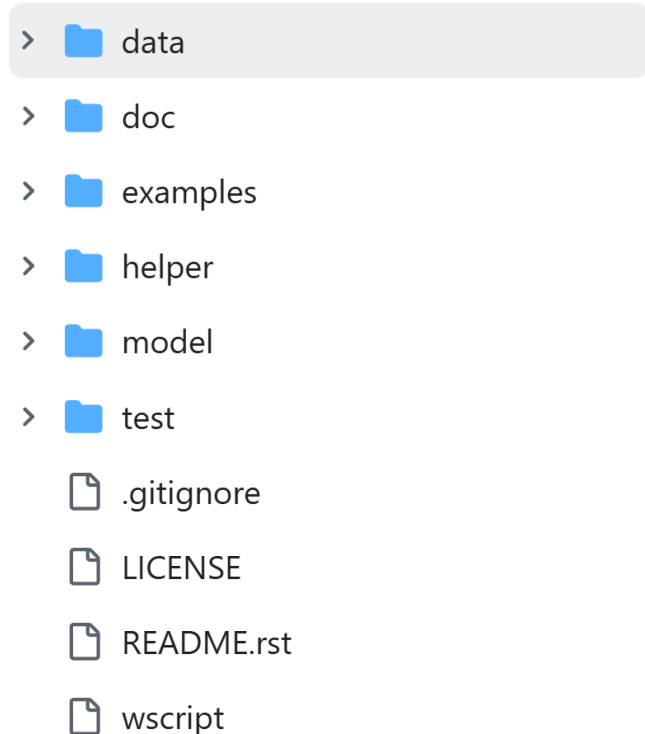
```
demo@demo-VirtualBox:~/ns-3-allinone/ns-3.35$ ./waf --run=leo-bulk-send
Waf: Entering directory `/home/demo/ns-3-allinone/ns-3.35/build'
Waf: Leaving directory `/home/demo/ns-3-allinone/ns-3.35/build'
Build commands will be stored in build/compile_commands.json
'build' finished successfully (0.416s)
ISL enabled
msg="Could not connect callback to /NodeList/*/Ns3::TcpL4Protocol/SocketList/*/Tx", +0.000000000s -1 file=../src/core/model/config.cc, line=925
terminate called without an active exception

ns3.35-leo-bulk-send-debug:48815 terminated with signal 6 at PC=7da08ec969fc SP=7ffdbbd11920. Backtrace:
/lib/x86_64-linux-gnu/libc.so.6(pthread_kill+0x12c)[0x7da08ec969fc]
/lib/x86_64-linux-gnu/libc.so.6(raise+0x16)[0x7da08ec42476]
/lib/x86_64-linux-gnu/libc.so.6(abort+0xd3)[0x7da08ec287f3]
/lib/x86_64-linux-gnu/libstdc++.so.6(+0xa2b9e)[0x7da08f0a2b9e]
/lib/x86_64-linux-gnu/libstdc++.so.6(+0xae20c)[0x7da08f0ae20c]
/lib/x86_64-linux-gnu/libstdc++.so.6(+0xae277)[0x7da08f0ae277]
/home/demo/ns-3-allinone/ns-3.35/build/lib/libns3.35-core-debug.so(_ZN3ns36Config7ConnectENSt7__cxx1112basic_stringIcSt11char_traitsIcESaIcEEERKNS_12CallbackBaseE+0x339)[0x7da091e051d1]
/home/demo/ns-3-allinone/ns-3.35/build/contrib/leo/examples/ns3.35-leo-bulk-send-debug(+0xb3cb)[0x634d086863cb]
/lib/x86_64-linux-gnu/libc.so.6(+0x29d90)[0x7da08ec29d90]
/lib/x86_64-linux-gnu/libc.so.6(__libc_start_main+0x80)[0x7da08ec29e40]
/home/demo/ns-3-allinone/ns-3.35/build/contrib/leo/examples/ns3.35-leo-bulk-send-debug(+0x8615)[0x634d08683615]
Command ['/home/demo/ns-3-allinone/ns-3.35/build/contrib/leo/examples/ns3.35-leo-bulk-send-debug'] exited with code 1
```

# NS3

---

- [Introduction to NS3](#)
- Lab2 related code
  - [ns-3.35/contrib/leo/examples/leo-bulk-send-example.cc](https://ns-3.35/contrib/leo/examples/leo-bulk-send-example.cc)
- Some important folders
  - examples
  - helper
  - Model
- Reference
  - [NS3 API documentation](#)
  - [NS3 tutorial](#)



A file explorer view showing the structure of the NS3 project. It lists several folders and files. The folders are 'data', 'doc', 'examples', 'helper', 'model', and 'test', each preceded by a right-pointing chevron and a blue folder icon. Below the folders are five files: '.gitignore', 'LICENSE', 'README.rst', and 'wscript', each preceded by a document icon.

- > data
- > doc
- > examples
- > helper
- > model
- > test
- .gitignore
- LICENSE
- README.rst
- wscript

# Run Example Code (1/4)

---

- Modify `calculate_delay.cc` in `/ns-3-allinone/ns-3.35/contrib/leo/examples`
  - It is a modified version of `leo-bulk-send.cc`
  - Modifications:
    - Fixed segmentation fault
    - Set up satellite and user configuration
    - Set up transmission configuration
    - Found satellite position

# Run Example Code (2/4)

- Modify the `wscript` file in `/ns-3-allinone/ns-3.35/contrib/leo/examples`

```
obj = bid.create_ns3_program('calculate_delay',  
                             [the module you need])  
obj.source = 'calculate_delay.cc'
```

- Execute `calculate_delay.cc`

```
./waf --run calculate_delay
```

- Output

Current simulation time                      Sequence number of the packet

```
+2.33088e+09ns: /NodeList/25/$ns3::TcpL4Protocol/SocketList/0/Tx 157:0x60263ce28d40 1026  
+2.41715e+09ns: /NodeList/26/$ns3::TcpL4Protocol/SocketList/0/Rx 157:0x60263ce28e40:1026
```

# Run Example Code (3/4)

---

- **TODO: Task1**

- Complete TODOs in `calculate_delay.cc`
- Parse the send time and arrival time of each packet
- Store the send time and arrival time for the same sequency number
- Calculate the average end-to-end delay of all the packets
- Output format:

```
Packet average end-to-end delay is 2.5s
```

# Run Example Code (4/4)

---

- **TODO: Task2**
  - Modify `calculate_delay.cc` to change the sending data rate of the source
  - Repeat task1 and observe whether the end-to-end delay changes accordingly