**Mininet安裝教學**

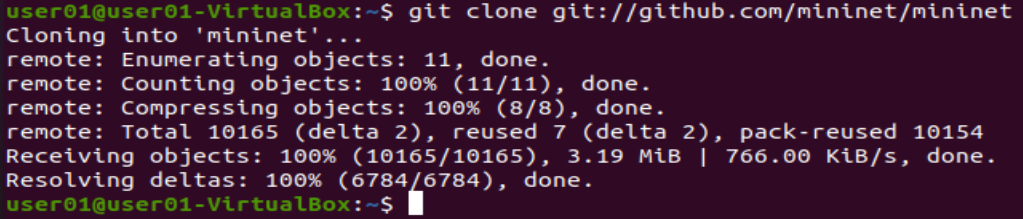
**從source code安裝Mininet（比較推薦）**

**Step1.** git clone 下載 mininet

$ git clone git://github.com/mininet/mininet.git

（沒有git 先安裝git sudo apt install git

建議做必要的升級 sudo apt-get update ）



clone 下來應該都是最新版的，如果需要安裝舊版本可以照下面作法

**Step 1.1** 選擇舊版 mininet

進入 mininet repo

列出所有 version tag

git checkout 到該版本

回上層目錄

$ cd mininet

$ git tag

$ git checkout -b 2.2.1 2.2.1 (可以選更高的版本 建議 2.5.0)

$ cd ..

**Step2.** 安裝 mininet

預設是 -a 全部安裝

-h 可列出所有可用選項

-s mydir 可指定安裝資料夾，需放在所有選項的最前面

$ mininet/util/install.sh [options] #建議options 選 -n3V 2.5.0

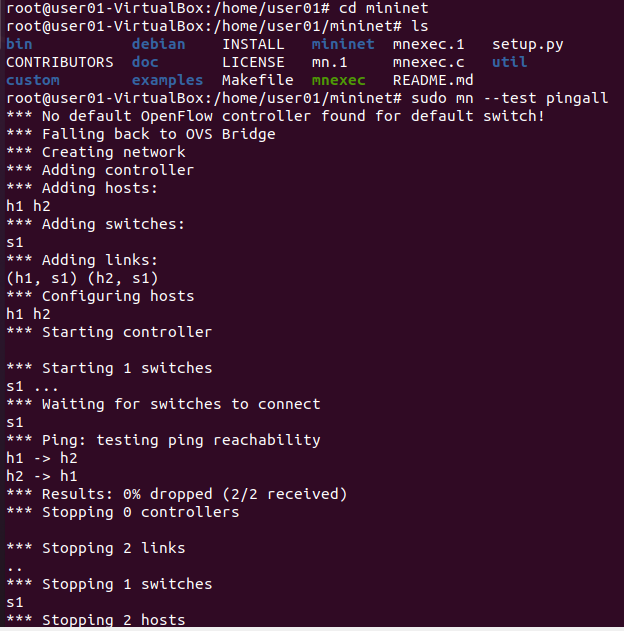
即

$ mininet/util/install.sh -n3V 2.5.0

（各參數n：安裝mininet 核心文件和依賴 3：支持OpenFlow V：**OpenvSwitch** 的版本） 時間比較久

**Step3.** 測試 Mininet

$ sudo mn --test pingall #驗證所有節點的連通性



# Mininet Walkthrough

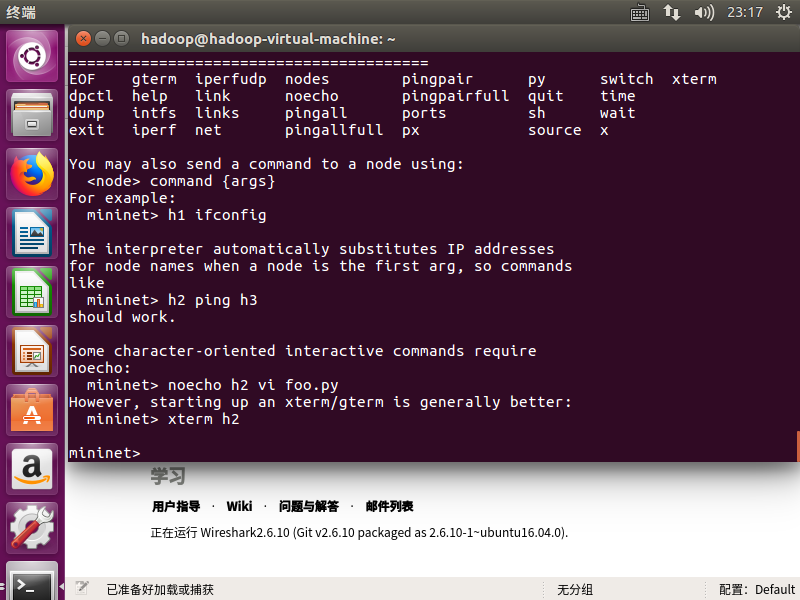
**$ sudo mn -h**

### Start Wireshark

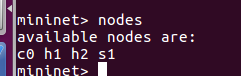
**$ sudo wireshark &**

#### 1621869359(1)

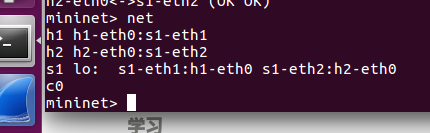
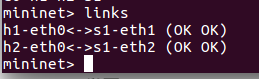
**mininet> help**

Display nodes:

**mininet> nodes**

Display links:

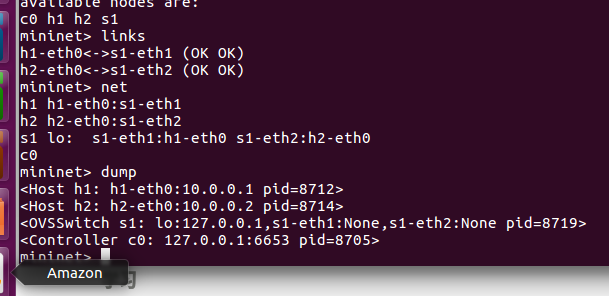
**mininet> net**

Dump information about all nodes:

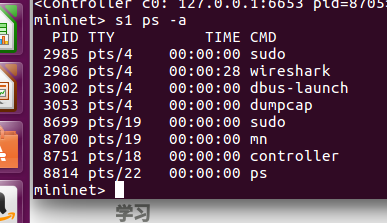
**mininet> dump**

**mininet> h1 ifconfig -a**

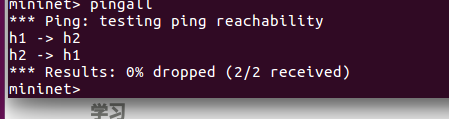
**mininet> s1 ifconfig -a**

**mininet> h1 ps -a**

**mininet> s1 ps -a**

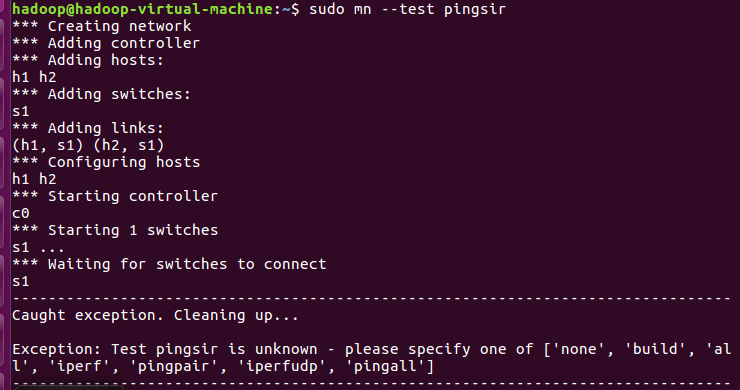
****

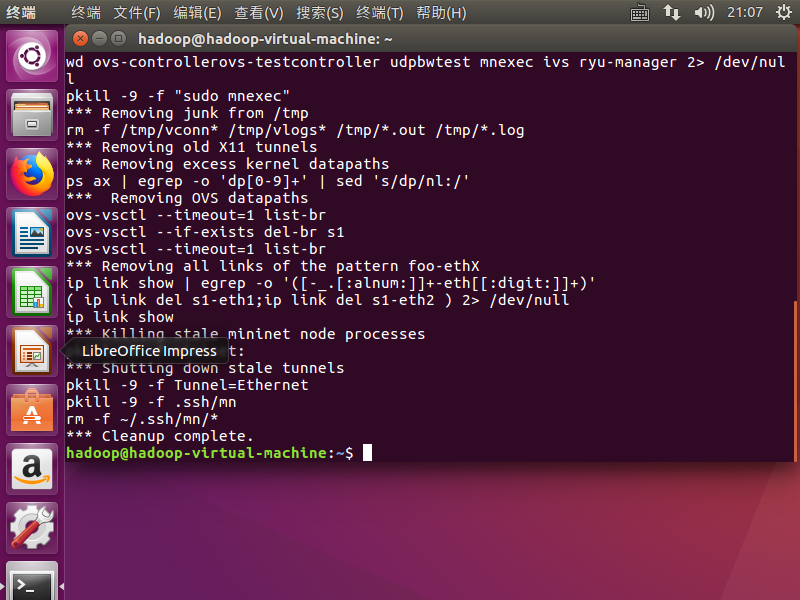
**Test connectivity between hosts**

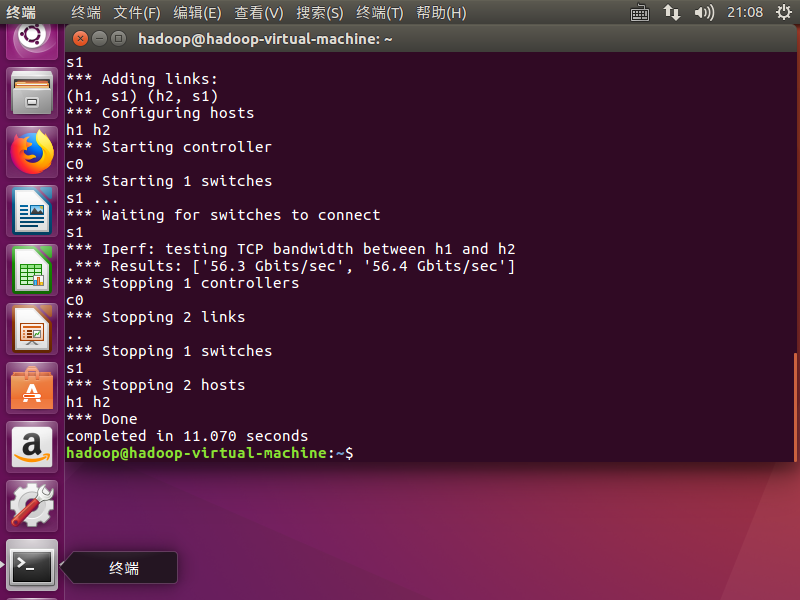
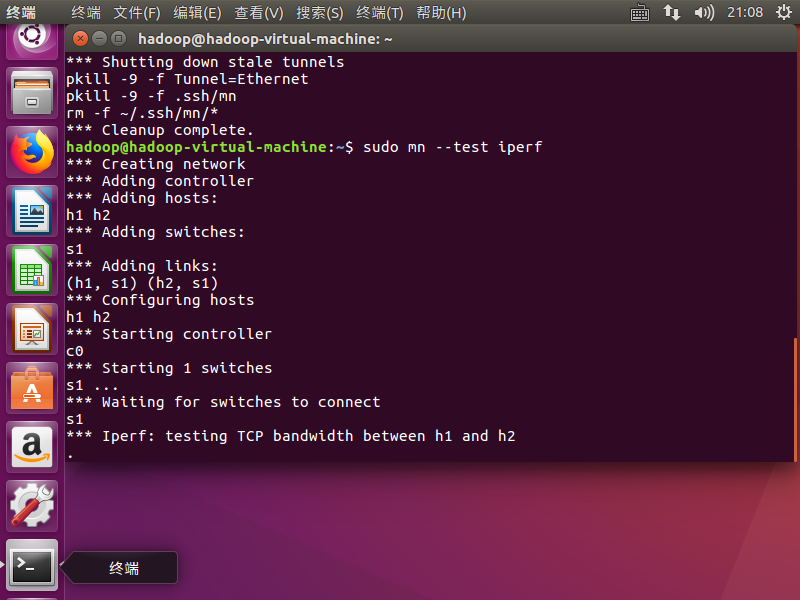
**mininet> pingall**

## Part 2: Advanced Startup Options

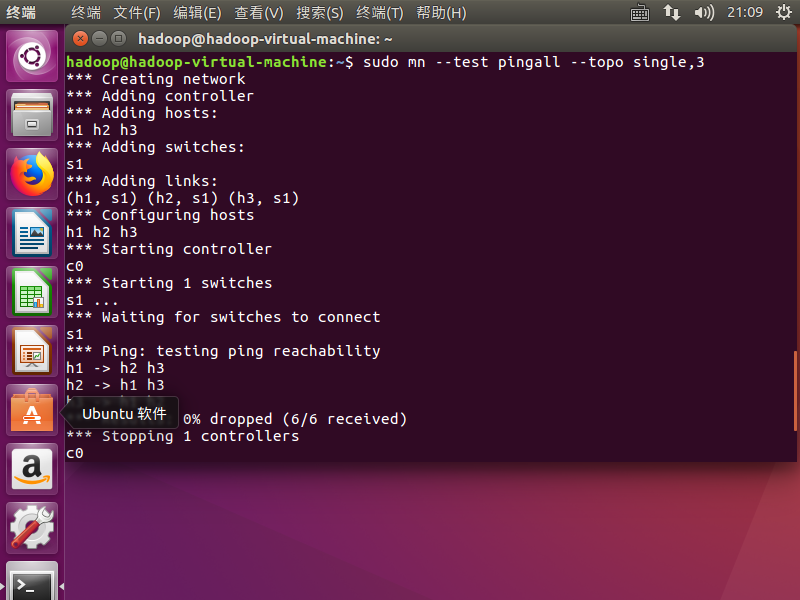
### Run a Regression Test

**$ sudo mn --test pingpair**

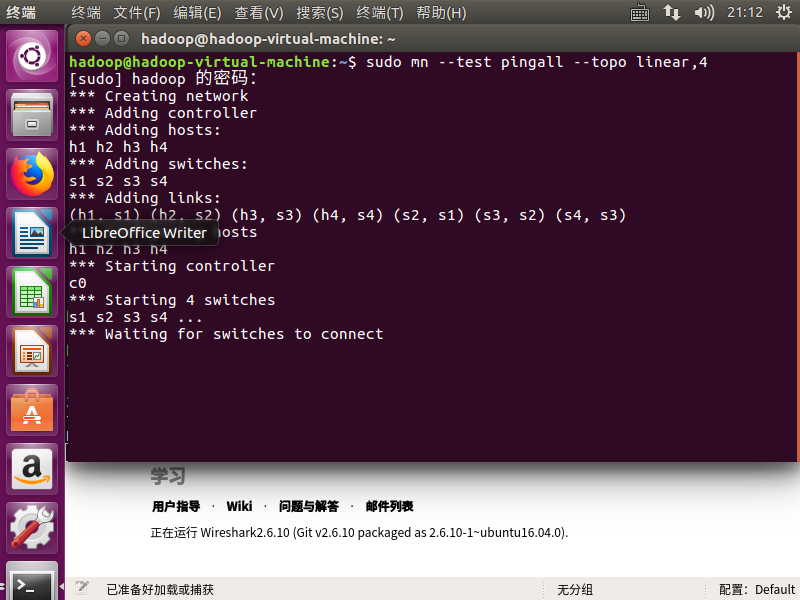
****

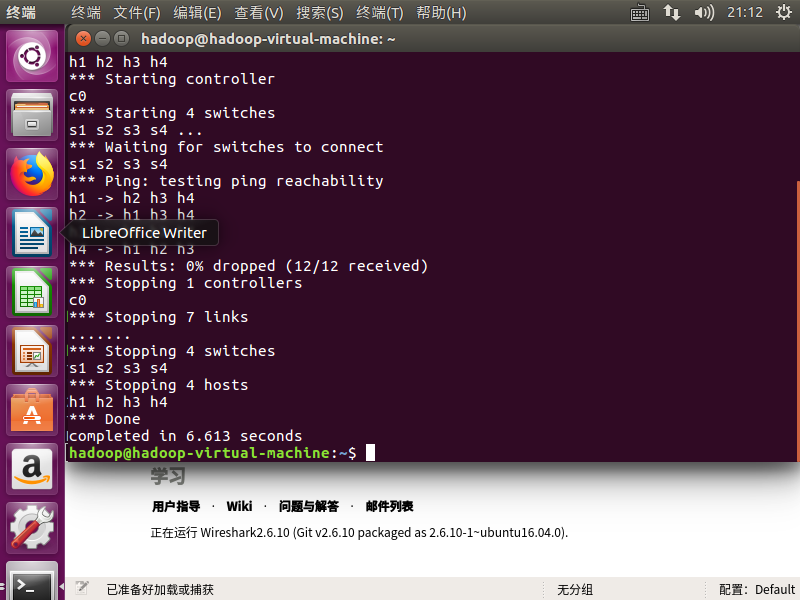
**$ sudo mn --test iperf**

### Changing Topology Size and Type

**$ sudo mn --test pingall --topo single,3**

****

**$ sudo mn --test pingall --topo linear,4**

****

Parametrized topologies are one of Mininet’s most useful and powerful features.

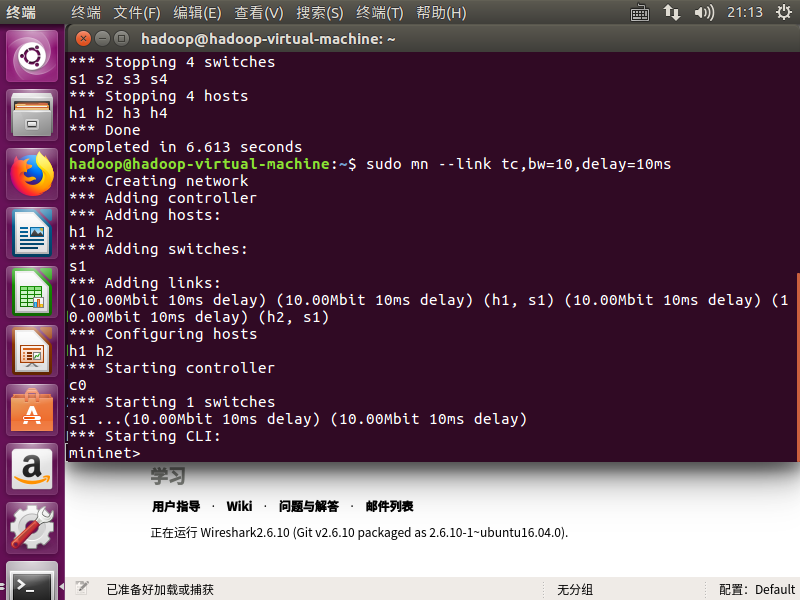
### Link variations

**$ sudo mn --link tc,bw=10,delay=10ms**

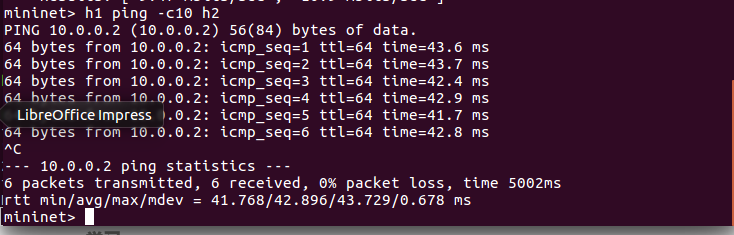
**mininet> iperf**

**...**

**mininet> h1 ping -c10 h2**

****

****

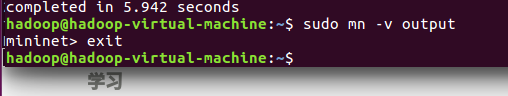
****

### Adjustable Verbosity

**$ sudo mn -v debug**

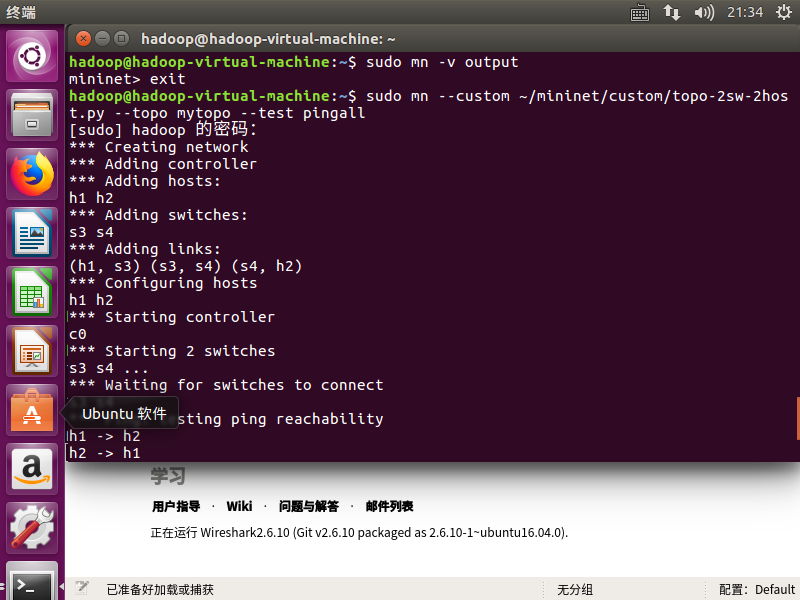
**...**

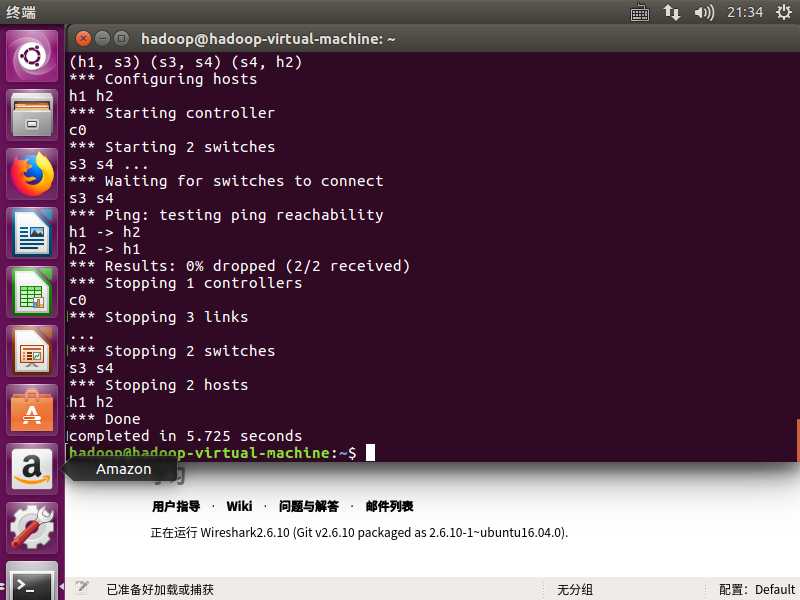
**mininet> exit**

****

Lots of extra detail will print out. Now try **output**, a setting that prints CLI output and little else:

**$ sudo mn -v output**

****

****

**mininet> exit**

### Custom Topologies

**$ sudo mn --custom ~/mininet/custom/topo-2sw-2host.py --topo mytopo --test pingall**

### ID = MAC

**$ sudo mn**

**...**

**mininet> h1 ifconfig**

**h1-eth0 Link encap:Ethernet HWaddr f6:9d:5a:7f:41:42**

**inet addr:10.0.0.1 Bcast:10.255.255.255 Mask:255.0.0.0**

**UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1**

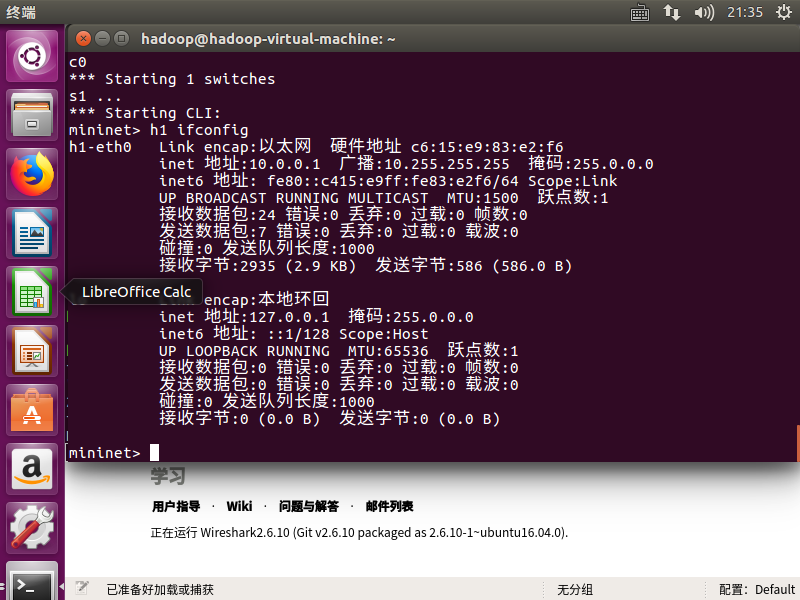
**RX packets:6 errors:0 dropped:0 overruns:0 frame:0**

**TX packets:6 errors:0 dropped:0 overruns:0 carrier:0**

**collisions:0 txqueuelen:1000**

**RX bytes:392 (392.0 B) TX bytes:392 (392.0 B)**

**mininet> exit**

****

After:

**$ sudo mn --mac**

**...**

**mininet> h1 ifconfig**

**h1-eth0 Link encap:Ethernet HWaddr 00:00:00:00:00:01**

**inet addr:10.0.0.1 Bcast:10.255.255.255 Mask:255.0.0.0**

**UP BROADCAST RUNNING MULTICAST MTU:1500 Metric:1**

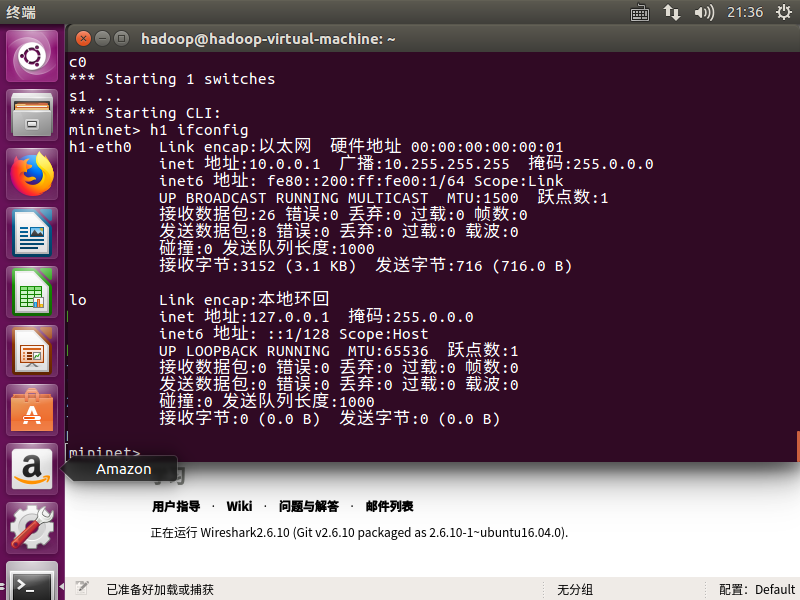
**RX packets:0 errors:0 dropped:0 overruns:0 frame:0**

**TX packets:0 errors:0 dropped:0 overruns:0 carrier:0**

**collisions:0 txqueuelen:1000**

**RX bytes:0 (0.0 B) TX bytes:0 (0.0 B)**

**mininet> exit**

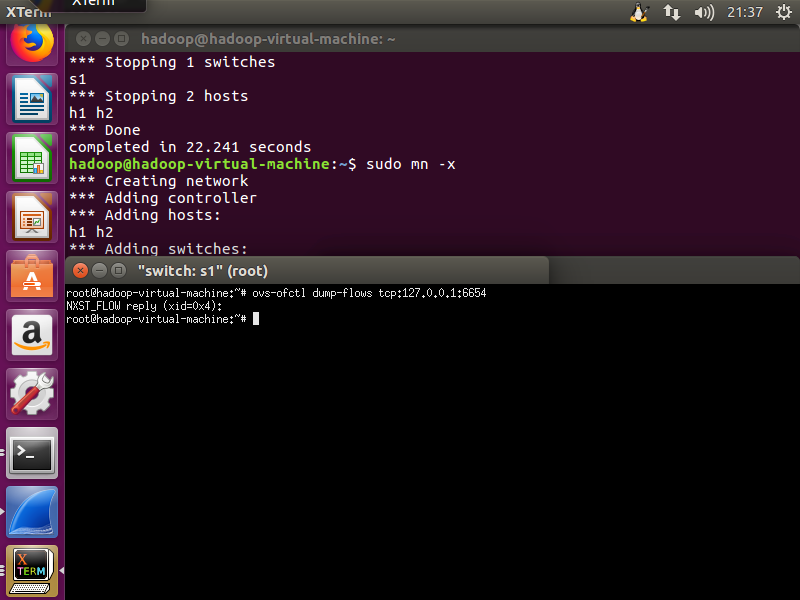
****

### XTerm Display

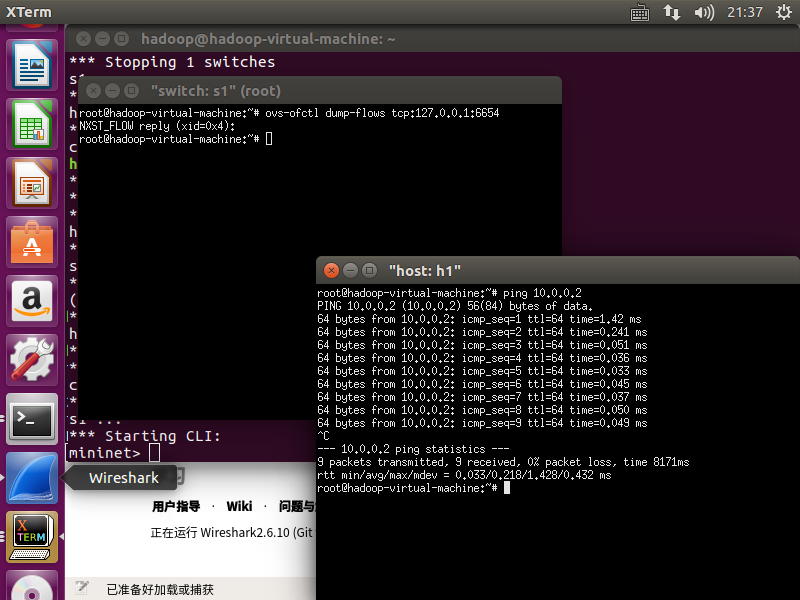
For more complex debugging, you can start Mininet so that it spawns one or more xterms.

To start an **xterm** for every host and switch, pass the **-x** option:

**$ sudo mn -x**

****

**# ovs-ofctl dump-flows tcp:127.0.0.1:6654**

****

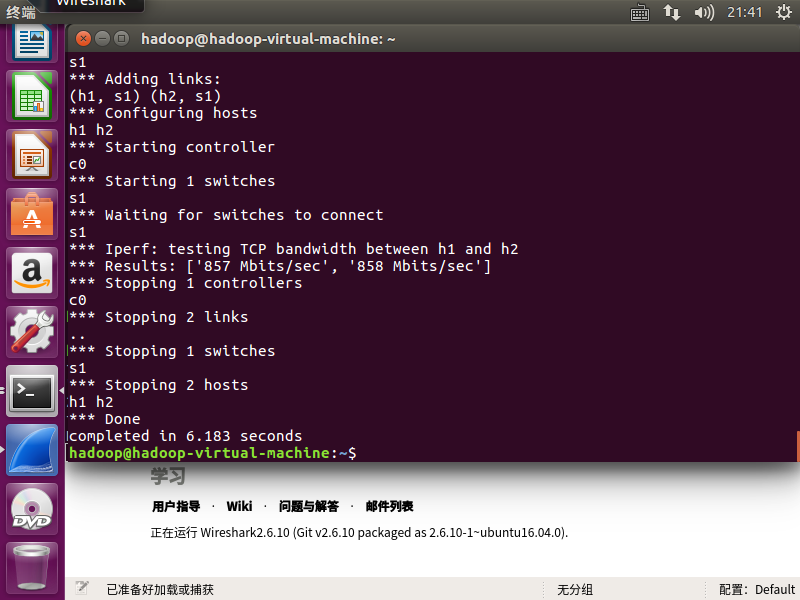
**# ping 10.0.0.2**

**mininet> exit**

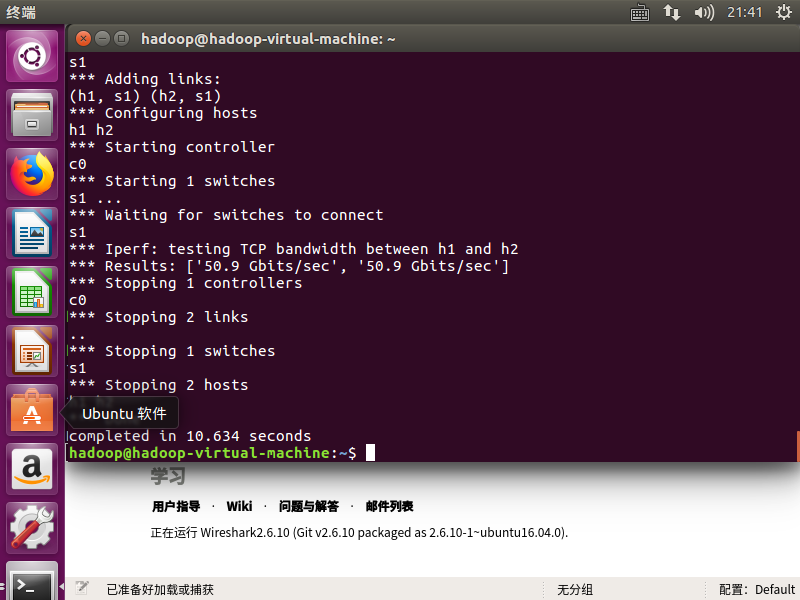
### Other Switch Types

Other switch types can be used. For example, to run the user-space switch:

**$ sudo mn --switch user --test iperf**

****

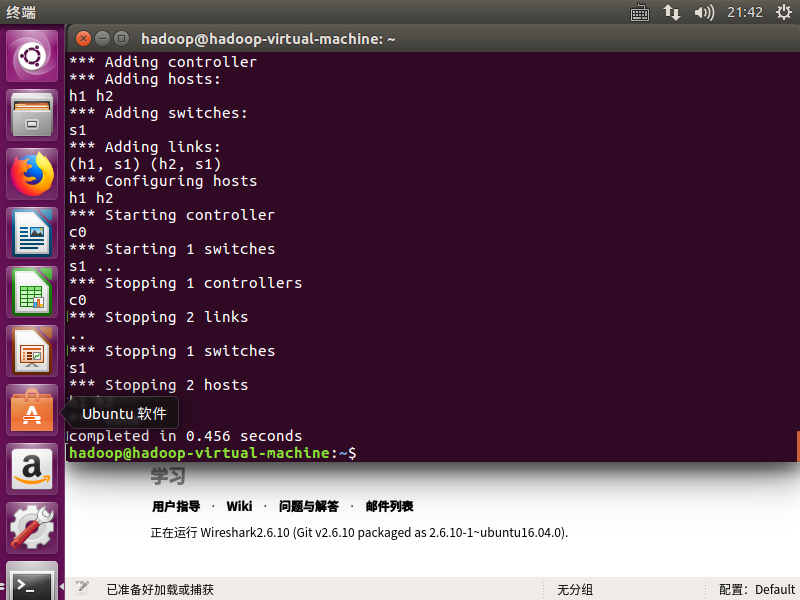
**$ sudo mn --switch ovsk --test iperf**

****

### Mininet Benchmark

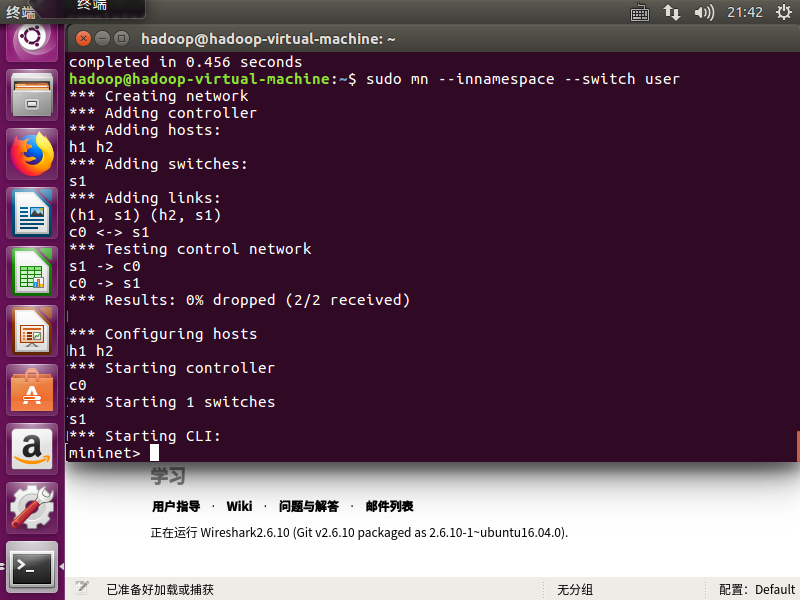
To record the time to set up and tear down a topology, use test ‘none’:

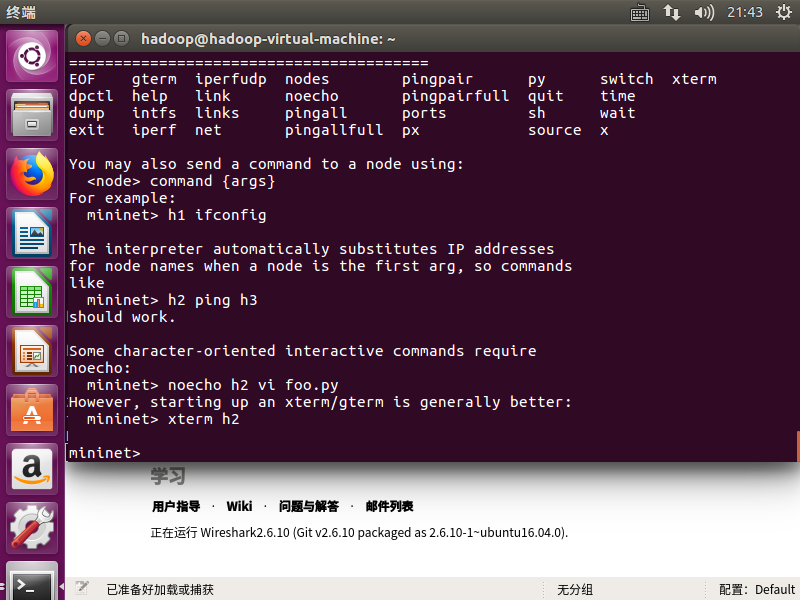
**$ sudo mn --test none**

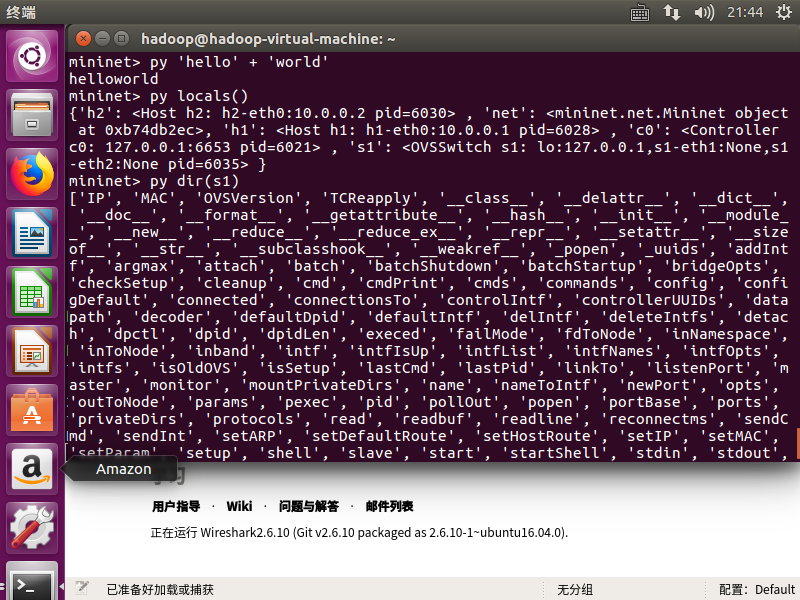
****

### Everything in its own Namespace (user switch only)

**$ sudo mn --innamespace --switch user**

****

****

****

**mininet> exit**

## Part 3: Mininet Command-Line Interface (CLI) Commands

### Display Options

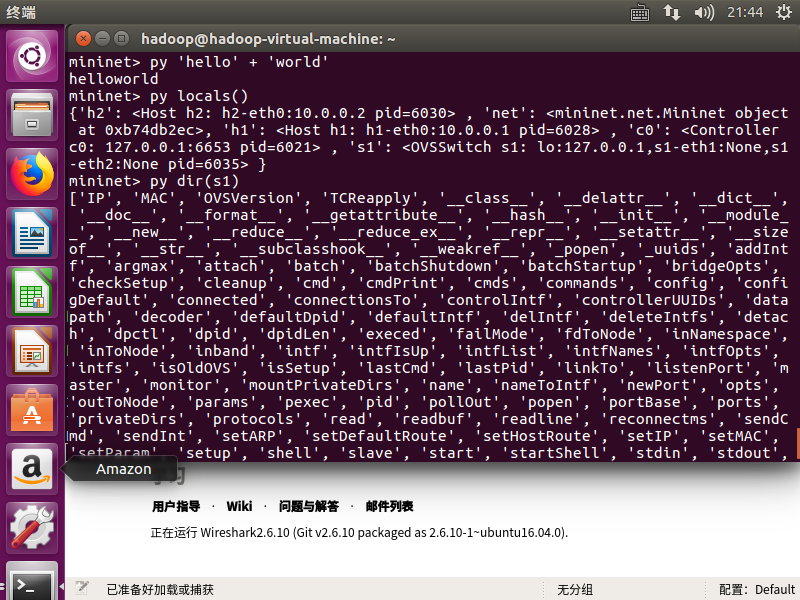
To see the list of Command-Line Interface (CLI) options, start up a minimal topology and leave it running. Build the Mininet:

**$ sudo mn**

Display the options:

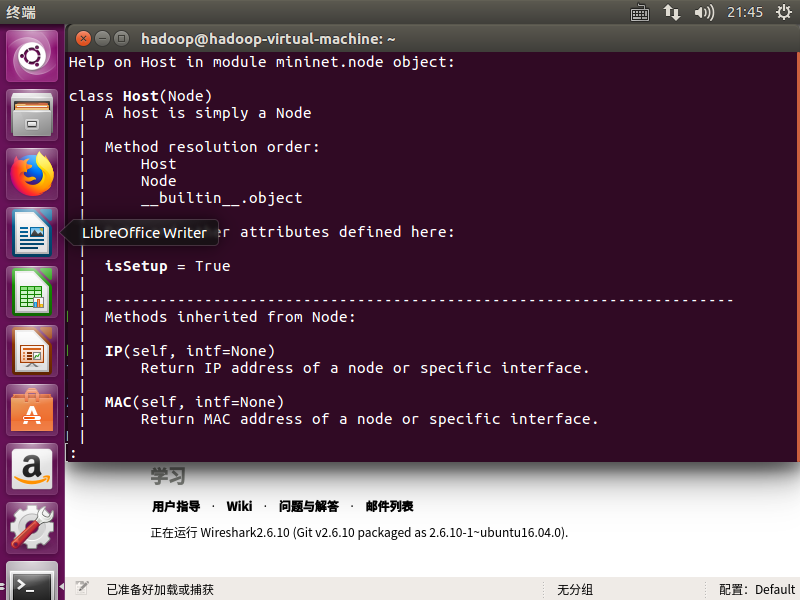
**mininet> help**

### Python Interpreter

**mininet> py 'hello ' + 'world'**

**mininet> py locals()**

****

****

**mininet> py dir(s1)**

**mininet> py help(h1) (Press "q" to quit reading the documentation.)**

****

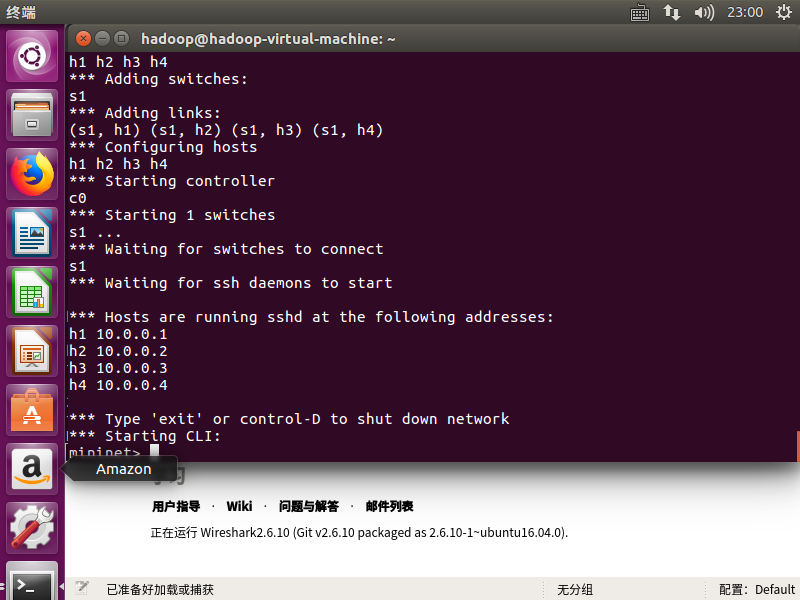
**mininet> py h1.IP()**

### Link Up/Down

For fault tolerance testing, it can be helpful to bring links up and down.

To disable both halves of a virtual ethernet pair:

**mininet> link s1 h1 down**

****

**mininet> link s1 h1 up**

**mininet> xterm h1 h2**

## Part 4: Python API Examples

### SSH daemon per host

One example that may be particularly useful runs an SSH daemon on every host:

**$ sudo ~/mininet/examples/sshd.py**

From another terminal, you can ssh into any host and run interactive commands:

**$ ssh 10.0.0.1**

**$ ping 10.0.0.2**

**...**

**$ exit**

Exit SSH example mininet:

**$ exit**

## Part 5: Walkthrough Complete!

### Next Steps to mastering Mininet

## Appendix: Supplementary Information

These are not required, but you might find them useful to skim.

### Using a Remote Controller

Note: this step is not part of the default walkthrough; it is primarily useful if you have a controller running outside of the VM, such as on the VM host, or a different physical PC. The OpenFlow tutorial uses ***controller --remote*** for starting up a simple learning switch that you create using a controller framework like POX, NOX, Beacon or Floodlight.

When you start a Mininet network, each switch can be connected to a remote controller - which could be in the VM, outside the VM and on your local machine, or anywhere in the world.

This setup may be convenient if you already have a custom version of a controller framework and development tools (such as Eclipse) installed on the local machine, or you want to test a controller running on a different physical machine (maybe even in the cloud).**controller listening port**

If you want to try this, fill in the host IP and/or listening port:

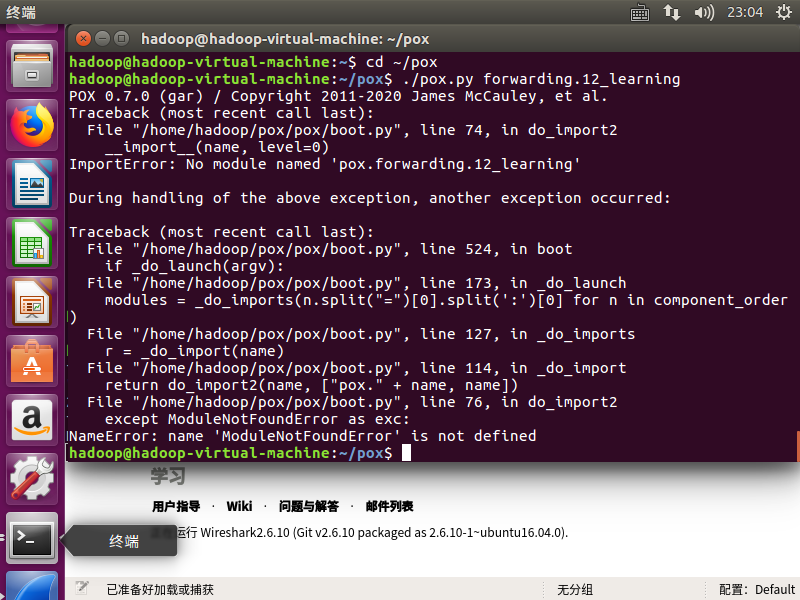
**$ sudo mn --controller=remote,ip=[controller IP],port=[controller listening port]**

For example, to run POX’s sample learning switch, you could do something like

**$ cd ~/pox**

**$ ./pox.py forwarding.l2\_learning**

**$ sudo mn --controller=remote,ip=127.0.0.1,port=6633**

****

### Ryu

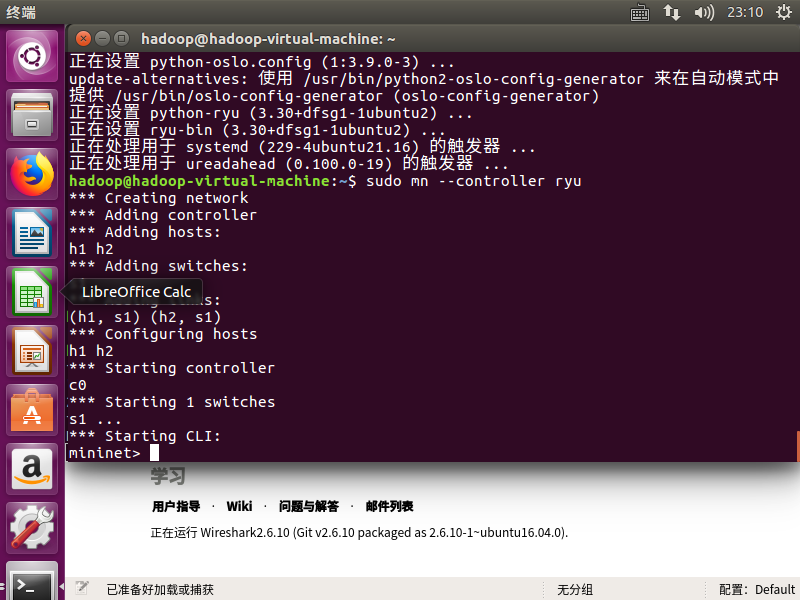
**$ sudo pip3 install ryu # install ryu if it's not already installed**

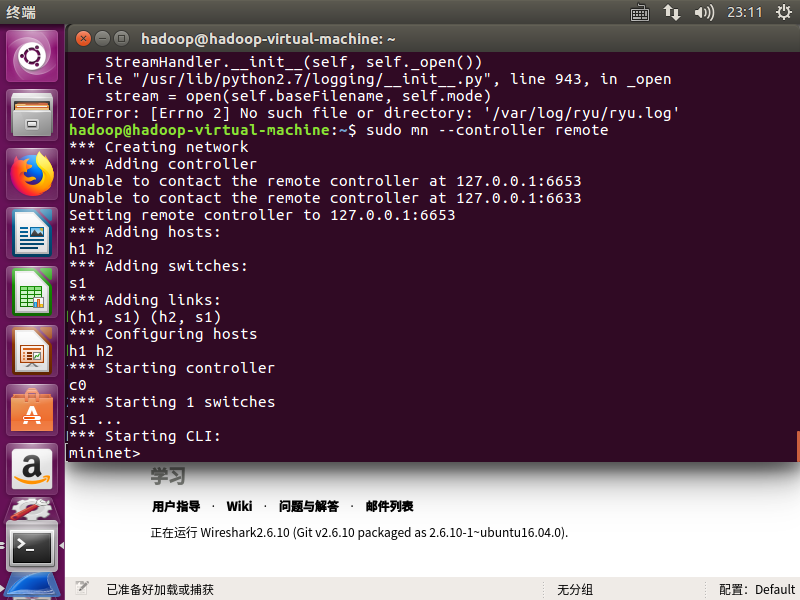
**$ sudo mn --controller ryu**

**$ sudo mn --controller,ryu.app.simple\_switch\_13**

**$ ryu run ryu.app.simple\_switch**

**$ sudo mn --controller remote**

****

****