

MININET Simulation

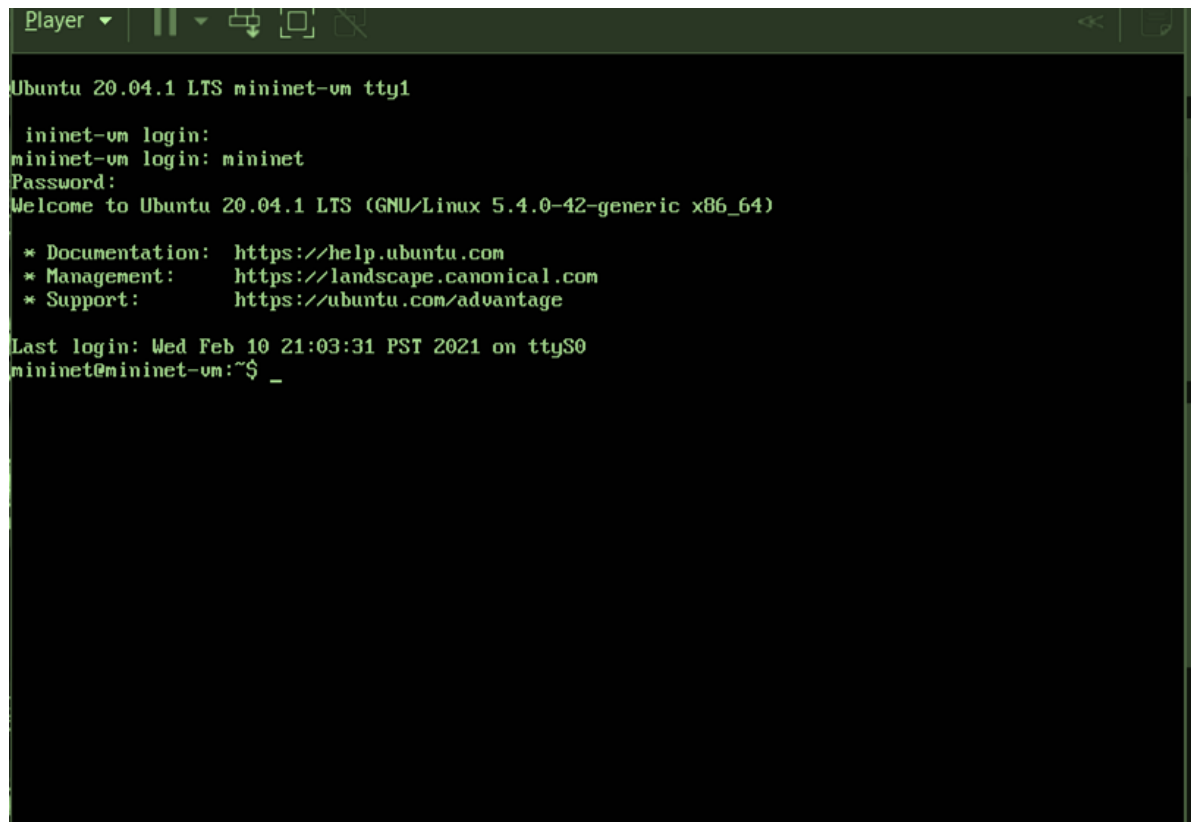
INSTALLATION

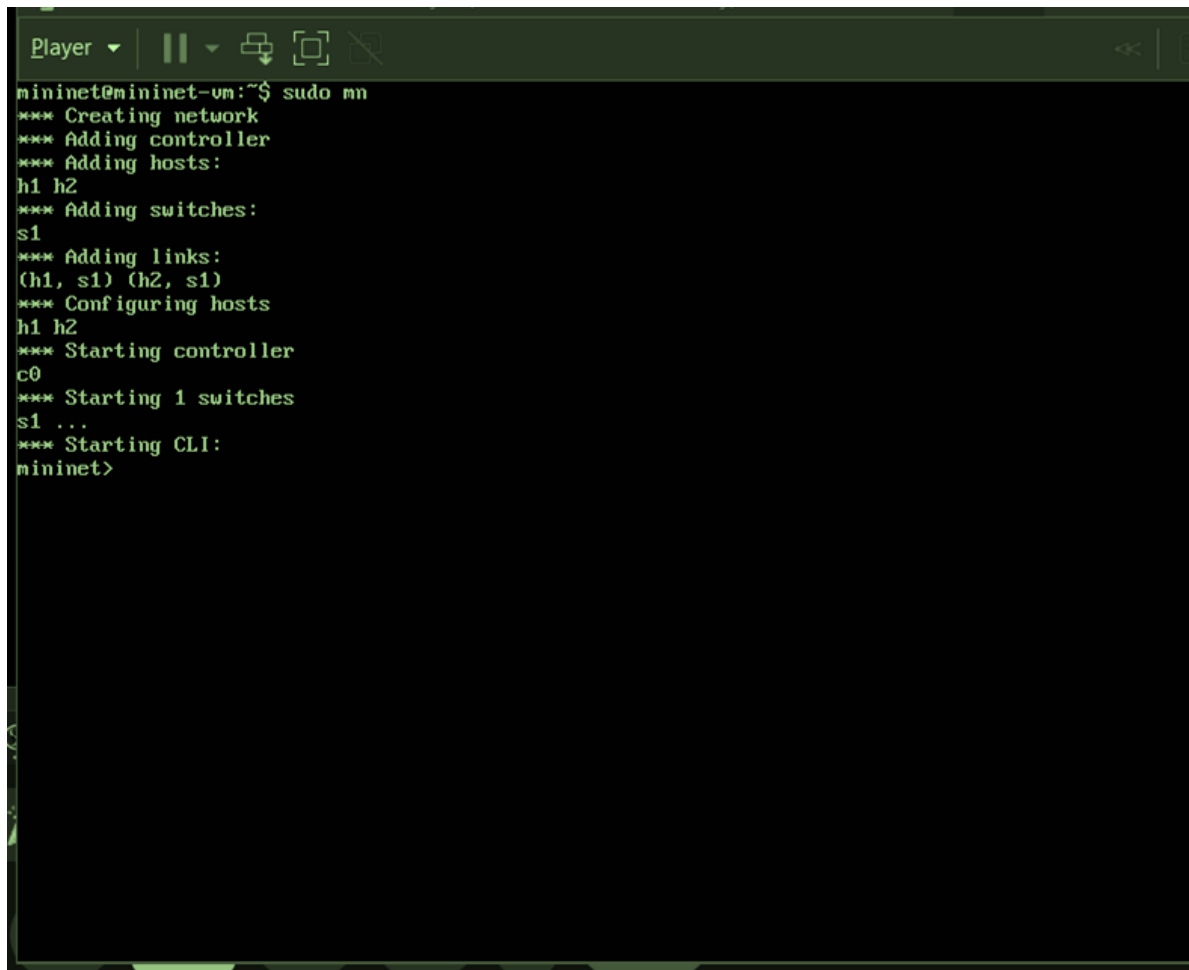
```
user@test:~$ git clone https://github.com/mininet/mininet
Cloning into 'mininet'...
remote: Enumerating objects: 11, done.
remote: Counting objects: 100% (11/11), done.
remote: Compressing objects: 100% (9/9), done.
remote: Total 10165 (delta 2), reused 3 (delta 1), pack-reused 10154
Receiving objects: 100% (10165/10165), 3.19 MiB | 5.15 MiB/s, done.
Resolving deltas: 100% (6784/6784), done.
user@test:~$ mininet/util/install.sh -a
Detected Linux distribution: Ubuntu 20.04 focal amd64
sys.version_info(major=2, minor=7, micro=18, releaselevel='final', serial=0)
Detected Python (python) version 2
Installing all packages except for -eix (doxypy, ivs, nox-classic)...
Install Mininet-compatible kernel if necessary
```

Login Details

Username: mininet

Password: mininet

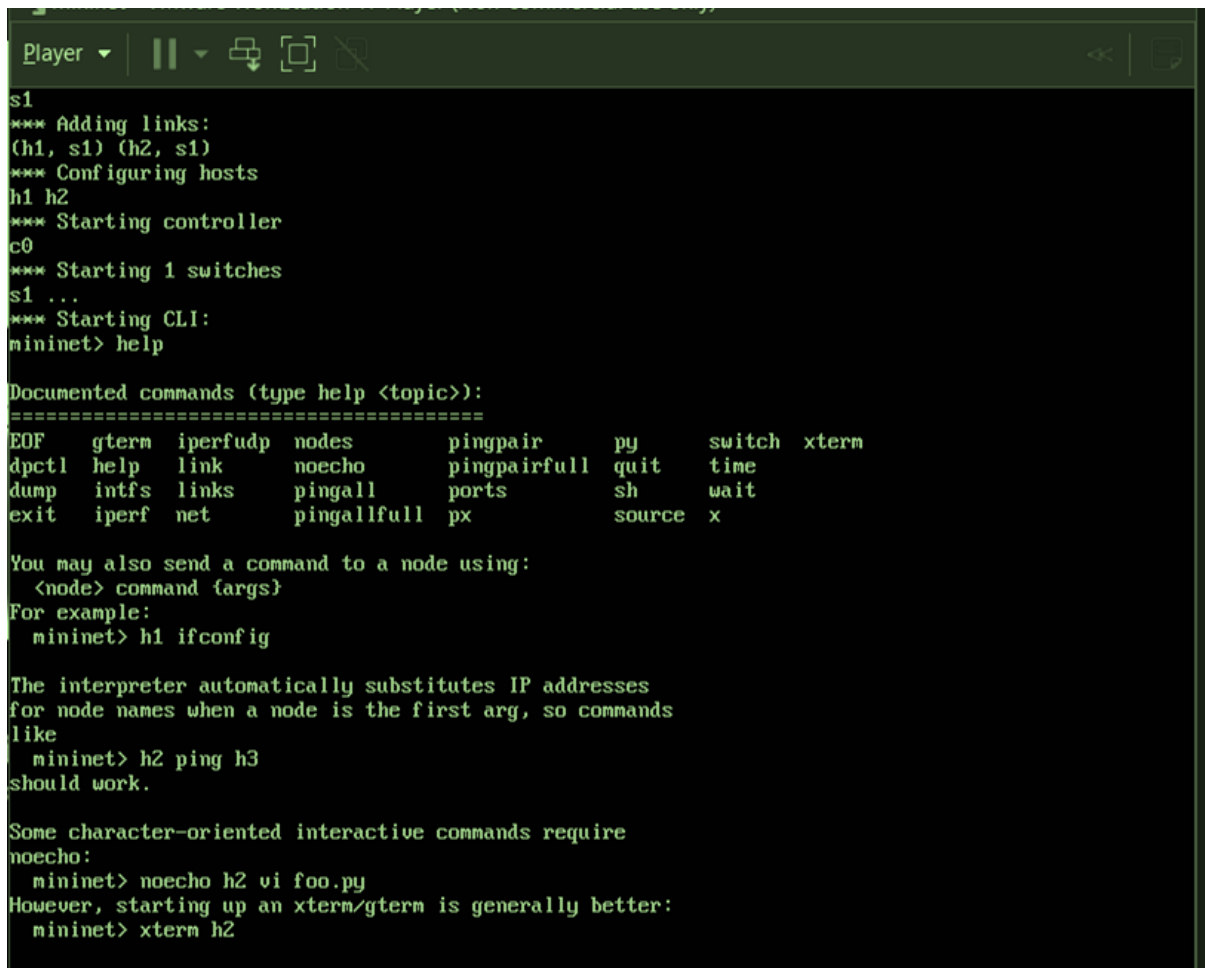
A screenshot of a terminal window titled 'Ubuntu 20.04.1 LTS mininet-vm tty1'. The terminal shows the login process for the 'mininet-vm' user. The prompt is 'mininet-vm login:', followed by the username 'mininet-vm login: mininet' and the password prompt 'Password:'. After the password is entered, the system displays the Ubuntu welcome message: 'Welcome to Ubuntu 20.04.1 LTS (GNU/Linux 5.4.0-42-generic x86_64)'. Below this, there are links for documentation, management, and support. The last login information is shown: 'Last login: Wed Feb 10 21:03:31 PST 2021 on ttyS0'. The prompt then changes to 'mininet@mininet-vm:~\$'.



```
mininet@mininet-vm:~$ sudo mn
*** Creating network
*** Adding controller
*** Adding hosts:
h1 h2
*** Adding switches:
s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet>
```

Opening mininet topology

Sudo mn



```

s1
*** Adding links:
(h1, s1) (h2, s1)
*** Configuring hosts
h1 h2
*** Starting controller
c0
*** Starting 1 switches
s1 ...
*** Starting CLI:
mininet> help

Documented commands (type help <topic>):
=====
EOF      gterm  iperfudp  nodes      pingpair    py      switch  xterm
dpctl    help   link      noecho     pingpairfull  quit    time
dump     intfs  links     pingall    ports        sh      wait
exit     iperf  net       pingallfull  px          source  x

You may also send a command to a node using:
  <node> command {args}
For example:
  mininet> h1 ifconfig

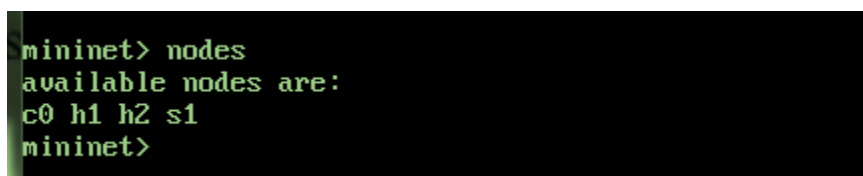
The interpreter automatically substitutes IP addresses
for node names when a node is the first arg, so commands
like
  mininet> h2 ping h3
should work.

Some character-oriented interactive commands require
noecho:
  mininet> noecho h2 vi foo.py
However, starting up an xterm/gterm is generally better:
  mininet> xterm h2

```

Command: help or ?

Shows the operations that we can perform



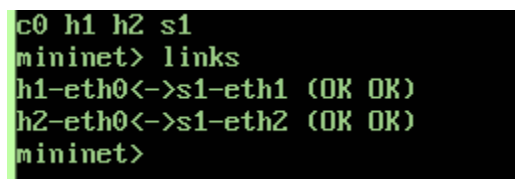
```

mininet> nodes
available nodes are:
c0 h1 h2 s1
mininet>

```

Command nodes

Shows the connected devices



```

c0 h1 h2 s1
mininet> links
h1-eth0<->s1-eth1 (OK OK)
h2-eth0<->s1-eth2 (OK OK)
mininet>

```

Command: links

Shows the connections between nodes, which node is connected to other and by which interface

```
mininet> net
h1 h1-eth0:s1-eth1
h2 h2-eth0:s1-eth2
s1 lo: s1-eth1:h1-eth0 s1-eth2:h2-eth0
c0
mininet>
```

Command: net

Show the links between the nodes

```
mininet> h1 ifconfig
h1-eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST> mtu 1500
    inet 10.0.0.1 netmask 255.0.0.0 broadcast 10.255.255.255
    ether 32:f4:91:d6:13:50 txqueuelen 1000 (Ethernet)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

lo: flags=73<UP,LOOPBACK,RUNNING> mtu 65536
    inet 127.0.0.1 netmask 255.0.0.0
    loop txqueuelen 1000 (Local Loopback)
    RX packets 0 bytes 0 (0.0 B)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0

mininet>
```

Command: h1 ifconfig

Show the description of h1 like ip address netmask connected ports and so on

```
mininet> h1 ping h2
PING 10.0.0.2 (10.0.0.2) 56(84) bytes of data.
64 bytes from 10.0.0.2: icmp_seq=1 ttl=64 time=3.26 ms
64 bytes from 10.0.0.2: icmp_seq=2 ttl=64 time=0.547 ms
64 bytes from 10.0.0.2: icmp_seq=3 ttl=64 time=0.085 ms
64 bytes from 10.0.0.2: icmp_seq=4 ttl=64 time=0.087 ms
64 bytes from 10.0.0.2: icmp_seq=5 ttl=64 time=0.086 ms
64 bytes from 10.0.0.2: icmp_seq=6 ttl=64 time=0.087 ms
64 bytes from 10.0.0.2: icmp_seq=7 ttl=64 time=0.087 ms
^C
--- 10.0.0.2 ping statistics ---
7 packets transmitted, 7 received, 0% packet loss, time 6120ms
rtt min/avg/max/mdev = 0.085/0.606/3.264/1.096 ms
mininet> _
```

Command: h1 ping h2

Tests the connection of node h1 with node h2 and in this case it is successful connection.