Overview

- Configure networking and routing in Gnu/Linux:
  - Basic networking from command line
  - DHCP server
  - Basic firewall rules
  - NAT
Ethernet Interfaces

- Ethernet interfaces are identified by the system using the naming convention of ethX, where X represents a numeric value.
- The first Ethernet interface is typically identified as eth0, the second as eth1, and all others should move up in numerical order.
- To quickly identify all available Ethernet interfaces, you can use the `ifconfig` command as shown below.
  - `ifconfig -a | grep eth`
  - `eth0 Link encap:Ethernet  HWaddr 00:15:c5:4a:16:5a`
**Lshw command**

- Lshw shows a single Ethernet interface with the logical name of eth0 along with bus information, driver details and all supported capabilities.

```
lshw -class network
  *-network
    description: Ethernet interface
    product: BCM4401-B0 100Base-TX
    vendor: Broadcom Corporation
    physical id: 0
    bus info: pci@0000:03:00.0
    logical name: eth0
    version: 02
    serial: 00:15:c5:4a:16:5a
    size: 10MB/s
    capacity: 100MB/s
    width: 32 bits
    clock: 33MHz
    capabilities: (snipped for brevity)
    configuration: (snipped for brevity)
    resources: irq:17 memory:ef9fe000-ef9fffffff
```
Ethernet Interface Logical Names

- Interface logical names are configured in the file /etc/udev/rules.d/70-persistent-net.rules.

- If you would like control which interface receives a particular logical name, find the line matching the interfaces physical MAC address and modify the value of NAME=ethX to the desired logical name. Reboot the system to commit your changes.

- SUBSYSTEM=="net", ACTION=="add", DRIVERS=="*", ATTR{address}=="00:15:c5:4a:16:5a", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth0"

- SUBSYSTEM=="net", ACTION=="add", DRIVERS=="*", ATTR{address}=="00:15:c5:4a:16:5b", ATTR{dev_id}=="0x0", ATTR{type}=="1", KERNEL=="eth*", NAME="eth1"
Temporary IP Address Assignment

- For temporary network configurations, you can use standard commands such as `ip`, `ifconfig` and `route`, which are also found on most other GNU/Linux operating systems.

- These commands allow you to configure settings which take effect immediately, however they are not persistent and will be lost after a reboot.

- To temporarily configure an IP address, you can use the `ifconfig` command in the following manner.

- Just modify the IP address and subnet mask to match your network requirements.

  - `ifconfig eth0 10.0.0.100 netmask 255.255.255.0`
Verify the IP address configuration

To verify the IP address configuration of eth0, you can use the `ifconfig` command in the following manner.

```
ifconfig eth0
```

```
eth0   Link encap:Ethernet  HWaddr 00:15:c5:4a:16:5a
inet addr:10.0.0.100  Bcast:10.0.0.255  Mask:255.255.255.0
inet6 addr: fe80::215:c5ff:fe4a:165a/64 Scope:Link
UP  BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
RX packets:466475604 errors:0 dropped:0 overruns:0 frame:0
TX packets:403172654 errors:0 dropped:0 overruns:0 carrier:0
    collisions:0 txqueuelen:1000
RX bytes:2574778386 (2.5 GB)  TX bytes:1618367329 (1.6 GB)
Interrupt:16
```
Default gateway

- To configure a default gateway, you can use the route command in the following manner. Modify the default gateway address to match your network requirements.

  `sudo route add default gw 10.0.0.1 eth0`

- To verify your default gateway configuration, you can use the route command in the following manner.

  `route -n`

<table>
<thead>
<tr>
<th>Destination</th>
<th>Gateway</th>
<th>Genmask</th>
<th>Flags</th>
<th>Metric</th>
<th>Ref</th>
<th>Use</th>
<th>Iface</th>
</tr>
</thead>
<tbody>
<tr>
<td>10.0.0.0</td>
<td>0.0.0.0</td>
<td>255.255.255.0</td>
<td>U</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>eth0</td>
</tr>
<tr>
<td>0.0.0.0</td>
<td>10.0.0.1</td>
<td>0.0.0.0</td>
<td>UG</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>eth0</td>
</tr>
</tbody>
</table>
Route examples

- route add -net 127.0.0.0 netmask 255.0.0.0 dev lo
  - adds the normal loopback entry, using netmask 255.0.0.0 and associated with the "lo" device (assuming this device was previously set up correctly with ifconfig(8)).

- route add -net 192.56.76.0 netmask 255.255.255.0 dev eth0
  - adds a route to the local network 192.56.76.x via "eth0". The word "dev" can be omitted here.

- route del default
  - deletes the current default route, which is labeled "default" or 0.0.0.0 in the destination field of the current routing table.

- route add default gw mango-gw
  - adds a default route (which will be used if no other route matches). All packets using this route will be gatewayed through "mango-gw". The device which will actually be used for that route depends on how we can reach "mango-gw" - the static route to "mango-gw" will have to be set up before.
Route examples

- `route add ipx4 s10`
  - Adds the route to the "ipx4" host via the SLIP interface (assuming that "ipx4" is the SLIP host).

- `route add -net 192.57.66.0 netmask 255.255.255.0 gw ipx4`
  - This command adds the net "192.57.66.x" to be gatewayed through the former route to the SLIP interface.

- `route add -net 224.0.0.0 netmask 240.0.0.0 dev eth0`
  - This is an obscure one documented so people know how to do it. This sets all of the class D (multicast) IP routes to go via "eth0". This is the correct normal configuration line with a multicasting kernel.

- `route add -net 10.0.0.0 netmask 255.0.0.0 reject`
  - This installs a rejecting route for the private network "10.x.x.x."
Configure DNS

- If you require DNS for your temporary network configuration, you can add DNS server IP addresses in the file `/etc/resolv.conf`

- The example below shows how to enter two DNS servers to `/etc/resolv.conf`, which should be changed to servers appropriate for your network.

  nameserver 8.8.8.8

  nameserver 8.8.4.4
Purge configuration

- If you no longer need this configuration and wish to purge all IP configuration from an interface, you can use the `ip` command with the `flush` option as shown below.
  - `ip addr flush eth0`
Interface configuration file

- The interface configuration file is `/etc/network/interfaces`.

- The following is an example of how the interface identified as eth0 could be permanently configured with a port speed of 1000Mb/s running in full duplex mode.
  - `auto eth0`
  - `iface eth0 inet static`
  - `pre-up /usr/sbin/ethtool -s eth0 speed 1000 duplex full`
Dynamic IP Address Assignment (DHCP Client)

- To configure your server to use DHCP for dynamic address assignment, add the dhcp method to the inet address family statement for the appropriate interface in the file `/etc/network/interfaces`.

- The example below assumes you are configuring your first Ethernet interface identified as eth0.
  - `auto eth0`
  - `iface eth0 inet dhcp`

- By adding an interface configuration as shown above, you can manually enable the interface through the `ifup` command which initiates the DHCP process via `dhclient`.
  - `ifup eth0`

- To manually disable the interface, you can use the `ifdown` command, which in turn will initiate the DHCP release process and shut down the interface.
  - `ifdown eth0`
Static IP Address Assignment

- To configure your system to use a static IP address assignment, add the static method to the inet address family statement for the appropriate interface in the file `/etc/network/interfaces`.

- The example below assumes you are configuring your first Ethernet interface identified as `eth0`.

- Change the address, netmask, and gateway values to meet the requirements of your network.
  - `auto eth0`
  - `iface eth0 inet static`
  - `address 10.0.0.100`
  - `netmask 255.255.255.0`
  - `gateway 10.0.0.1`
Loopback Interface

- The loopback interface is identified by the system as lo and has a default IP address of **127.0.0.1**
- It can be viewed using the `ifconfig` command.

```bash
ifconfig lo
lo Link encap:Local Loopback
inet addr:127.0.0.1 Mask:255.0.0.0
inet6 addr: ::1/128 Scope:Host
UP LOOPBACK RUNNING MTU:16436 Metric:1
RX packets:2718 errors:0 dropped:0 overruns:0 frame:0
TX packets:2718 errors:0 dropped:0 overruns:0 carrier:0
collisions:0 txqueuelen:0
RX bytes:183308 (183.3 KB) TX bytes:183308 (183.3 KB)
```

- By default, there should be two lines in `/etc/network/interfaces` responsible for automatically configuring your loopback interface. An example of the two default lines are shown below
  - `auto lo`
  - `Iface lo inet loopback`
DNS Client Configuration

- To configure your system to use DNS for name resolution, add the IP addresses of the DNS servers that are appropriate for your network in the file `/etc/resolv.conf`.

- You can also add an optional DNS suffix search-lists to match your network domain names.

- Below is an example of a typical configuration of `/etc/resolv.conf` for a server on the domain "example.com" and using two public DNS servers.
  - `search example.com`
  - `nameserver 8.8.8.8`
  - `nameserver 8.8.4.4`
Static Hostnames

- Static hostnames are locally defined hostname-to-IP mappings located in the file `/etc/hosts`.
- Entries in the hosts file will have precedence over DNS by default. This means that if your system tries to resolve a hostname and it matches an entry in `/etc/hosts`, it will not attempt to look up the record in DNS.
- The following is an example of a hosts file where a number of local servers have been identified by simple hostnames, aliases and their equivalent Fully Qualified Domain Names (FQDN's).
  - 127.0.0.1 localhost
  - 127.0.1.1 ubuntu-server
  - 10.0.0.11 server1 vpn server1.example.com
  - 10.0.0.12 server2 mail server2.example.com
  - 10.0.0.13 server3 www server3.example.com
  - 10.0.0.14 server4 file server4.example.com
Name Service Switch Configuration

- The order in which your system selects a method of resolving hostnames to IP addresses is controlled by the Name Service Switch (NSS) configuration file `/etc/nsswitch.conf`.

- Typically static hostnames defined in the systems `/etc/hosts` file have precedence over names resolved from DNS.

- The following is an example of the line responsible for this order of hostname lookups in the file `/etc/nsswitch.conf`:

  ```
  hosts:  files mdns4_minimal [NOTFOUND=return] dns mdns4
  (mdns4: multicast DNS service discovery)
  ```
Bridging

- Bridging multiple interfaces is a more advanced configuration, but is very useful in multiple scenarios.

- One scenario is setting up a bridge with **multiple network interfaces**, then using a firewall to filter traffic between two network segments.

- Another scenario is using bridge on a system with one interface to **allow virtual machines** direct access to the outside network.

- The following **example** covers the latter scenario.

- Before configuring a bridge you will need to install the bridge-utils package. To **install the package**, in a terminal enter:
  
  - `apt-get install bridge-utils`
Bridging example

- configure the bridge by editing `/etc/network/interfaces`:
  
  ```
  auto lo
  iface lo inet loopback
  auto br0
  iface br0 inet static
    address 192.168.0.10
    network 192.168.0.0
    netmask 255.255.255.0
    broadcast 192.168.0.255
    gateway 192.168.0.1
  bridge_ports eth0
  bridge_fd 9
  bridge_hello 2
  bridge_maxage 12
  bridge_stp off
  ```

- Enter the appropriate values for your physical interface and network
Install DHCP server

- At a terminal prompt, enter the following command to install dhcpd:
  - `apt-get install dhcp3-server`
- You will probably need to change the default configuration by editing `/etc/dhcp3/dhcpd.conf` to suit your needs and particular configuration.
- You also need to edit `/etc/default/dhcp3-server` to specify the interfaces dhcpd should listen to. By default it listens to `eth0`.
- **NOTE:** `dhcpd's messages are being sent to syslog`. Look there for diagnostics messages.
DHCP configuration example

- Most commonly, what you want to do is assign an IP address randomly. This can be done with settings as follows:

```
# Sample /etc/dhcpd.conf
# (add your comments here)
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.1.255;
option routers 192.168.1.254;
option domain-name-servers 192.168.1.1, 192.168.1.2;
option domain-name "mydomain.example";

subnet 192.168.1.0 netmask 255.255.255.0 {
  range 192.168.1.10 192.168.1.100;
  range 192.168.1.150 192.168.1.200;
}
```
Example explanation

- The previous example result in the DHCP server giving a client an **IP address from the range** 192.168.1.10-192.168.1.100 or 192.168.1.150-192.168.1.200.

- It will **lease an IP** address for 600 seconds if the client doesn't ask for a specific time frame.

- Otherwise the **maximum (allowed) lease** will be 7200 seconds.

- The server will also "advise" the client that it should use 255.255.255.0 as its **subnet mask**, 192.168.1.255 as its **broadcast address**, 192.168.1.254 as the **router/gateway** and 192.168.1.1 and 192.168.1.2 as its **DNS servers**.
Basic Iptables Commands

• Typing `sudo iptables -L` lists your current rules in iptables.
• If you have just set up your server, you will have no rules, and you should see

  Chain INPUT (policy ACCEPT)
  target    prot opt source          destination

  Chain FORWARD (policy ACCEPT)
  target    prot opt source          destination

  Chain OUTPUT (policy ACCEPT)
  target    prot opt source          destination
Allowing Established Sessions

- We can **allow established sessions** to receive traffic:
  ```bash
  iptables -A INPUT -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
  ```

- The above rule has **no spaces** either side of the comma in `ESTABLISHED,RELATED`

- If the line above doesn't work, you may be on a castrated VPS whose provider has not made available the extension, in which case an inferior version can be used as last resort:
  ```bash
  iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
  ```
Allowing Incoming Traffic on Specific Ports

- You could start by blocking traffic, but you might be working over SSH, where you would need to allow SSH before blocking everything else.

- To allow incoming traffic on the default SSH port (22), you could tell iptables to allow all TCP traffic on that port to come in.

  ```bash
  iptables -A INPUT -p tcp --dport ssh -j ACCEPT
  ```

- Referring back to the list above, you can see that this tells iptables
  - append this rule to the input chain (-A INPUT) so we look at incoming traffic
  - check to see if it is TCP (-p tcp).
  - if so, check to see if the input goes to the SSH port (--dport ssh).
  - if so, accept the input (-j ACCEPT).
Check Iptable rules

- Checking our rules, we have

```
sudo iptables -L
```

Chain INPUT (policy ACCEPT)

```
target prot opt source destination
ACCEPT all -- anywhere anywhere state RELATED,ESTABLISHED
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
ACCEPT tcp -- anywhere anywhere tcp dpt:www
```

- We have specifically allowed tcp traffic to the ssh and web ports, but as we have not blocked anything, all traffic can still come in
Blocking Traffic

- Once a decision is made to accept a packet, no more rules affect it. As our rules allowing ssh and web traffic come first, as long as our rule to block all traffic comes after them, we can still accept the traffic we want. All we need to do is put the rule to block all traffic at the end.

```bash
iptables -A INPUT -j DROP
iptables -L
Chain INPUT (policy ACCEPT)
target prot opt source destination
ACCEPT all -- anywhere anywhere
ACCEPT tcp -- anywhere anywhere tcp dpt:ssh
ACCEPT tcp -- anywhere anywhere tcp dpt:www
DROP all -- anywhere anywhere
```

- Because we didn't specify an interface or a protocol, any traffic for any port on any interface is blocked, except for web and ssh.
Saving iptables

- If you were to reboot your machine right now, your iptables configuration would disappear.
- You can save the configuration, and have it start up automatically.
- To save the configuration, you can use
  - `iptables-save`
  - `Iptables-restore`
- Save your firewall rules to a file
  - `sh -c "iptables-save > /etc/iptables.rules"`
Apply rules at startup

- When you have found out the interface you are using open your `/etc/network/interfaces` and modify it:

  ```bash
  auto eth0
  iface eth0 inet dhcp
  pre-up iptables-restore < /etc/iptables.rules
  ```
Configure NAT

- Configure iptables for NAT translation so that packets can be correctly routed through the Ubuntu gateway.

  ```shell
  iptables -A FORWARD -o eth0 -i eth1 -s 192.168.0.0/24 -m conntrack --ctstate NEW -j ACCEPT
  
  iptables -A FORWARD -m conntrack --ctstate ESTABLISHED,RELATED -j ACCEPT
  
  iptables -A POSTROUTING -t nat -j MASQUERADE
  ```

- The first rule allows forwarded packets (initial ones).
- The second rule allows forwarding of established connection packets (and those related to ones that started).
- The third rule does the NAT.
Enable routing

- Configure the gateway for routing between two interfaces by enabling IP forwarding:
  
  sh -c "echo 1 > /proc/sys/net/ipv4/ip_forward"

- Edit `/etc/sysctl.conf`, and (up to Ubuntu 10.04) add these lines:
  
  - `net.ipv4.conf.default.forwarding=1`
  - `net.ipv4.conf.all.forwarding=1`

- From Ubuntu 10.10 onwards, it suffices to edit `/etc/sysctl.conf` and uncomment:
  
  - `#net.ipv4.ip_forward=1`
End of lesson