Basics

This section is dedicated to the basic tasks we can perform in ubuntu Linux

- How to add user in Ubuntu Lniux
- Install Docker Engine on Ubuntu
- Configuring networks

How to add user in Ubuntu Lniux

Ubuntu-logo.png

At some point in your Ubuntu Linux server or desktop administration journey, you will need to create a user.
Usually, we create users in the servers for administration on more than one person or a service account that
needs to be created in order for the service to only run as that user. This creates a security barrier in the services you run.
Here we have steps on how to complete this function. Follow the instructions and if you encounter an error, please email support.

> **This tutorial assumes that:**
> You know the IP address of the machine intended for changes, and
> can Login the machine with **Telnet**, **Serial** or, **SSH**

> **Note:**
> Only "sudo" can create a user, So, the first thing we have to do is check if the user you login with is a "sudo".
> Check with the command `sudo -l`

1. Check "sudo" privileges by running `sudo -l`

code:

```
mslsadmin@remote01:~$ sudo -l
[sudo] password for mslsadmin:
Matching Defaults entries for mslsadmin on remote01:
    env_reset, mail_badpass,
secure_path=/usr/local/sbin\:/usr/local/bin\:/usr/sbin\:/usr/bin\:/sbin\:/bin\:/snap/bin,
use_pty
```

```
User mslsadmin may run the following commands on remote01:
    (ALL : ALL) ALL
mslsadmin@remote01:~$
```

2. Use the `sudo adduser` command to add the new user and follow the prompt:

code:

```
mslsadmin@remote01:~$ sudo adduser test
Adding user `test' ...
Adding new group `test' (1001) ...
Adding new user `test' (1001) with group `test' ...
Creating home directory `/home/test' ...
Copying files from `/etc/skel' ...
```

3. Continue the prompt and create a password for the new user :

code:

```
New password: ****
Retype new password:*******
passwd: password updated successfully
Changing the user information for test
```

> note:
> While entering the password you might not see that keys are actually being entered. This is completely normal and may differ from other Linux distributions or console applications.

4. Next the `adduser` prompt will ask you for more information about the user, Enter relevant information or, if you want to skip the prompt, leave the options blank and press ENTER.

code:

```
Changing the user information for test
Enter the new value, or press ENTER for the default
        Full Name []: Test User
        Room Number []:
        Work Phone []:
        Home Phone []: d
        Other []:
```

5. Confirm your changes and choose "Y"

code:

```
Is the information correct? [Y/n] Y
```

Congratulations! You successfully created an user in Ubuntu Linux

Install Docker Engine on Ubuntu

To get started with Docker Engine on Ubuntu, make sure you meet the prerequisites, and then follow the installation steps.

Prerequisites

> **Note**
>
> If you use ufw or firewalld to manage firewall settings, be aware that when you expose container ports using Docker, these ports bypass your firewall rules. For more information, refer to Docker and ufw.

OS requirements

To install Docker Engine, you need the 64-bit version of one of these Ubuntu versions:

- Ubuntu Lunar 23.04
- Ubuntu Kinetic 22.10
- Ubuntu Jammy 22.04 (LTS)
- Ubuntu Focal 20.04 (LTS)

Docker Engine for Ubuntu is compatible with x86_64 (or amd64), armhf, arm64, s390x, and ppc64le (ppc64el) architectures.

Uninstall old versions

Before you can install Docker Engine, you must first make sure that any conflicting packages are uninstalled.

Distro maintainers provide an unofficial distributions of Docker packages in APT. You must uninstall these packages before you can install the official version of Docker Engine.

The unofficial packages to uninstall are:

- `docker.io`
- `docker-compose`
- `docker-doc`
- `podman-docker`

Moreover, Docker Engine depends on `containerd` and `runc`. Docker Engine bundles these dependencies as one bundle: `containerd.io`. If you have installed the `containerd` or `runc` previously, uninstall them to avoid conflicts with the versions bundled with Docker Engine.

Run the following command to uninstall all conflicting packages:

```
$ for pkg in docker.io docker-doc docker-compose podman-docker containerd runc; do sudo apt-get remove $pkg; done
```

`apt-get` might report that you have none of these packages installed.

Images, containers, volumes, and networks stored in `/var/lib/docker/` aren't automatically removed when you uninstall Docker. If you want to start with a clean installation, and prefer to clean up any existing data, read the uninstall Docker Engine section.

Installation methods

You can install Docker Engine in different ways, depending on your needs:

- Docker Engine comes bundled with Docker Desktop for Linux. This is the easiest and quickest way to get started.
- Set up and install Docker Engine from Docker's `apt` repository.
- Install it manually and manage upgrades manually.
- Use a convenience script. Only recommended for testing and development environments.

Install using the apt repository

Before you install Docker Engine for the first time on a new host machine, you need to set up the Docker repository. Afterward, you can install and update Docker from the repository.

Set up the repository

1. Update the `apt` package index and install packages to allow `apt` to use a repository over HTTPS:

   ```
   $ sudo apt-get update
   $ sudo apt-get install ca-certificates curl gnupg
   ```

2. Add Docker's official GPG key:

   ```
   $ sudo install -m 0755 -d /etc/apt/keyrings
   $ curl -fsSL https://download.docker.com/linux/ubuntu/gpg | sudo gpg --dearmor -o
   /etc/apt/keyrings/docker.gpg
   $ sudo chmod a+r /etc/apt/keyrings/docker.gpg
   ```

3. Use the following command to set up the repository:

   ```
   $ echo \
     "deb [arch="$(dpkg --print-architecture)" signed-by=/etc/apt/keyrings/docker.gpg]
   https://download.docker.com/linux/ubuntu \
     "$(. /etc/os-release && echo "$VERSION_CODENAME")" stable" | \
     sudo tee /etc/apt/sources.list.d/docker.list > /dev/null
   ```

   > **Note**
   > If you use an Ubuntu derivative distro, such as Linux Mint, you may need to use `UBUNTU_CODENAME` instead of `VERSION_CODENAME`.

4. Update the `apt` package index:

   ```
   $ sudo apt-get update
   ```

Install Docker Engine

1. Install Docker Engine, containerd, and Docker Compose.

Latest Specific version

To install the latest version, run:

```
$ sudo apt-get install docker-ce docker-ce-cli containerd.io docker-buildx-plugin docker-
compose-plugin
```

---

2. Verify that the Docker Engine installation is successful by running the `hello-world` image.

   ```
   $ sudo docker run hello-world
   ```

   This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

You have now successfully installed and started Docker Engine.

> **Tip**
>
> Receiving errors when trying to run without root?
>
> The `docker` user group exists but contains no users, which is why you’re required to use `sudo` to run Docker commands. Continue to Linux postinstall to allow non-privileged users to run Docker commands and for other optional configuration steps.

Upgrade Docker Engine

To upgrade Docker Engine, follow the installation instructions, choosing the new version you want to install.

Install from a package

If you can't use Docker's `apt` repository to install Docker Engine, you can download the `deb` file for your release and install it manually. You need to download a new file each time you want to upgrade Docker Engine.

1. Go to `https://download.docker.com/linux/ubuntu/dists/`open_in_new.
2. Select your Ubuntu version in the list.
3. Go to `pool/stable/` and select the applicable architecture (`amd64`, `armhf`, `arm64`, or `s390x`).
4. Download the following `deb` files for the Docker Engine, CLI, containerd, and Docker Compose packages:
   - `containerd.io_<version>_<arch>.deb`
   - `docker-ce_<version>_<arch>.deb`
   - `docker-ce-cli_<version>_<arch>.deb`
   - `docker-buildx-plugin_<version>_<arch>.deb`
   - `docker-compose-plugin_<version>_<arch>.deb`

5. Install the `.deb` packages. Update the paths in the following example to where you downloaded the Docker packages.

   ```
   $ sudo dpkg -i ./containerd.io_<version>_<arch>.deb \
     ./docker-ce_<version>_<arch>.deb \
     ./docker-ce-cli_<version>_<arch>.deb \
     ./docker-buildx-plugin_<version>_<arch>.deb \
     ./docker-compose-plugin_<version>_<arch>.deb
   ```

   The Docker daemon starts automatically.
6. Verify that the Docker Engine installation is successful by running the `hello-world` image.

   ```
   $ sudo service docker start
   $ sudo docker run hello-world
   ```

   This command downloads a test image and runs it in a container. When the container runs, it prints a confirmation message and exits.

You have now successfully installed and started Docker Engine.

> **Tip**
>
> Receiving errors when trying to run without root?
>
> The `docker` user group exists but contains no users, which is why you're required to use `sudo` to run Docker commands. Continue to Linux postinstall to allow non-privileged users to run Docker commands and for other optional configuration steps.

Upgrade Docker Engine

To upgrade Docker Engine, download the newer package files and repeat the installation procedure , pointing to the new files.

Install using the convenience script

Docker provides a convenience script at https://get.docker.com/open_in_new to install Docker into development environments non-interactively. The convenience script isn't recommended for production environments, but it's useful for creating a provisioning script tailored to your needs. Also refer to the install using the repository steps to learn about installation steps to install using the package repository. The source code for the script is open source, and you can find it in the `docker-install` repository on GitHubopen_in_new.

Always examine scripts downloaded from the internet before running them locally. Before installing, make yourself familiar with potential risks and limitations of the convenience script:

- The script requires `root` or `sudo` privileges to run.
- The script attempts to detect your Linux distribution and version and configure your package management system for you.
- The script doesn't allow you to customize most installation parameters.
- The script installs dependencies and recommendations without asking for confirmation. This may install a large number of packages, depending on the current configuration of your host machine.
- By default, the script installs the latest stable release of Docker, containerd, and runc. When using this script to provision a machine, this may result in unexpected major version upgrades of Docker. Always test upgrades in a test environment before deploying to your production systems.
- The script isn't designed to upgrade an existing Docker installation. When using the script to update an existing installation, dependencies may not be updated to the expected version, resulting in outdated versions.

> Tip: preview script steps before running
>
> You can run the script with the `--dry-run` option to learn what steps the script will run when invoked:
>
> ```
> $ curl -fsSL https://get.docker.com -o get-docker.sh
> $ sudo sh ./get-docker.sh --dry-run
> ```

This example downloads the script from https://get.docker.com/open_in_new and runs it to install the latest stable release of Docker on Linux:

```
$ curl -fsSL https://get.docker.com -o get-docker.sh
$ sudo sh get-docker.sh
Executing docker install script, commit: 7cae5f8b0decc17d6571f9f52eb840fbc13b2737
<...>
```

You have now successfully installed and started Docker Engine. The `docker` service starts automatically on Debian based distributions. On `RPM` based distributions, such as CentOS, Fedora, RHEL or SLES, you need to start it manually using the appropriate `systemctl` or `service` command. As the message indicates, non-root users can't run Docker commands by default.

> **Use Docker as a non-privileged user, or install in rootless mode?**
>
> The installation script requires `root` or `sudo` privileges to install and use Docker. If you want to grant non-root users access to Docker, refer to the post-installation steps for Linux. You can also install Docker without `root` privileges, or configured to run in rootless mode. For instructions on running Docker in rootless mode, refer to run the Docker daemon as a non-root user (rootless mode).

Install pre-releases

Docker also provides a convenience script at https://test.docker.com/open_in_new to install pre-releases of Docker on Linux. This script is equal to the script at `get.docker.com`, but configures your package manager to use the test channel of the Docker package repository. The test channel includes both stable and pre-releases (beta versions, release-candidates) of Docker. Use this script to get early access to new releases, and to evaluate them in a testing environment before they're released as stable.

To install the latest version of Docker on Linux from the test channel, run:

```
$ curl -fsSL https://test.docker.com -o test-docker.sh
$ sudo sh test-docker.sh
```

Upgrade Docker after using the convenience script

If you installed Docker using the convenience script, you should upgrade Docker using your package manager directly. There's no advantage to re-running the convenience script. Re-running it can cause issues if it attempts to re-install repositories which already exist on the host machine.

Uninstall Docker Engine

1. Uninstall the Docker Engine, CLI, containerd, and Docker Compose packages:

   ```
   $ sudo apt-get purge docker-ce docker-ce-cli containerd.io docker-buildx-plugin
   docker-compose-plugin docker-ce-rootless-extras
   ```

2. Images, containers, volumes, or custom configuration files on your host aren't automatically removed. To delete all images, containers, and volumes:

   ```
   $ sudo rm -rf /var/lib/docker
   $ sudo rm -rf /var/lib/containerd
   ```

You have to delete any edited configuration files manually.

Configuring networks

Ubuntu ships with a number of graphical utilities to configure your network devices. This document is geared toward server administrators and will focus on managing your network on the command line.

Ethernet interfaces

Ethernet interfaces are identified by the system using predictable network interface names. These names can appear as *eno1* or *enp0s25*. However, in some cases an interface may still use the kernel *eth#* style of naming.

Identify Ethernet interfaces

To quickly identify all available Ethernet interfaces, you can use the ip command as shown below.

```
ip a
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
2: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default qlen 1000
    link/ether 00:16:3e:e2:52:42 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.102.66.200/24 brd 10.102.66.255 scope global dynamic eth0
       valid_lft 3257sec preferred_lft 3257sec
    inet6 fe80::216:3eff:fee2:5242/64 scope link
       valid_lft forever preferred_lft forever
```

Another application that can help identify all network interfaces available to your system is the `lshw` command. This command provides greater details around the hardware capabilities of specific adapters. In the example below, `lshw` shows a single Ethernet interface with the logical name of *eth4* along with bus information, driver details and all supported capabilities.

```
sudo lshw -class network
  *-network
       description: Ethernet interface
       product: MT26448 [ConnectX EN 10GigE, PCIe 2.0 5GT/s]
       vendor: Mellanox Technologies
       physical id: 0
       bus info: pci@0004:01:00.0
       logical name: eth4
       version: b0
       serial: e4:1d:2d:67:83:56
       slot: U78CB.001.WZS09KB-P1-C6-T1
       size: 10Gbit/s
       capacity: 10Gbit/s
       width: 64 bits
       clock: 33MHz
       capabilities: pm vpd msix pciexpress bus_master cap_list ethernet physical fibre
10000bt-fd
       configuration: autonegotiation=off broadcast=yes driver=mlx4_en driverversion=4.0-0
duplex=full firmware=2.9.1326 ip=192.168.1.1 latency=0 link=yes multicast=yes port=fibre
speed=10Gbit/s
       resources: iomemory:24000-23fff irq:481 memory:3fe200000000-3fe2000fffff
memory:240000000000-240007ffffff
```

Ethernet Interface logical names

Interface logical names can also be configured via a Netplan configuration. If you would like control which interface receives a particular logical name use the `match` and `set-name` keys. The `match` key is used to find an adapter based on some criteria like MAC address, driver, etc. The `set-name` key can be used to change the device to the desired logical name.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth_lan0:
      dhcp4: true
      match:
        macaddress: 00:11:22:33:44:55
      set-name: eth_lan0
```

Ethernet Interface settings

`ethtool` is a program that displays and changes Ethernet card settings such as auto-negotiation, port speed, duplex mode, and Wake-on-LAN. The following is an example of how to view the supported features and configured settings of an Ethernet interface.

```
sudo ethtool eth4
Settings for eth4:
    Supported ports: [ FIBRE ]
    Supported link modes:   10000baseT/Full
    Supported pause frame use: No
    Supports auto-negotiation: No
    Supported FEC modes: Not reported
    Advertised link modes:  10000baseT/Full
    Advertised pause frame use: No
    Advertised auto-negotiation: No
    Advertised FEC modes: Not reported
    Speed: 10000Mb/s
    Duplex: Full
    Port: FIBRE
    PHYAD: 0
    Transceiver: internal
    Auto-negotiation: off
    Supports Wake-on: d
    Wake-on: d
    Current message level: 0x00000014 (20)
                   link ifdown
    Link detected: yes
```

IP addressing

The following section describes the process of configuring your system’s IP address and default gateway needed for communicating on a local area network and the Internet.

Temporary IP address assignment

For temporary network configurations, you can use the `ip` command which is also found on most other GNU/Linux operating systems. The `ip` command allows 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 `ip` command in the following manner. Modify the IP address and subnet mask to match your network requirements.

```
sudo ip addr add 10.102.66.200/24 dev enp0s25
```

The `ip` can then be used to set the link up or down.

```
ip link set dev enp0s25 up
ip link set dev enp0s25 down
```

To verify the IP address configuration of `enp0s25`, you can use the `ip` command in the following manner:

```
ip address show dev enp0s25
10: enp0s25: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc noqueue state UP group default
qlen 1000
    link/ether 00:16:3e:e2:52:42 brd ff:ff:ff:ff:ff:ff link-netnsid 0
    inet 10.102.66.200/24 brd 10.102.66.255 scope global dynamic eth0
       valid_lft 2857sec preferred_lft 2857sec
    inet6 fe80::216:3eff:fee2:5242/64 scope link
       valid_lft forever preferred_lft forever6
```

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

```
sudo ip route add default via 10.102.66.1
```

You can also use the `ip` command to verify your default gateway configuration, as follows:

```
ip route show
default via 10.102.66.1 dev eth0 proto dhcp src 10.102.66.200 metric 100
10.102.66.0/24 dev eth0 proto kernel scope link src 10.102.66.200
10.102.66.1 dev eth0 proto dhcp scope link src 10.102.66.200 metric 100
```

If you require DNS for your temporary network configuration, you can add DNS server IP addresses in the file `/etc/resolv.conf`. In general, editing `/etc/resolv.conf` directly is not recommended, but this is a temporary and non-persistent configuration. The example below shows how to enter two DNS servers to `/etc/resolv.conf`, which should be changed to servers appropriate for your network. A more lengthy description of the proper (persistent) way to do DNS client configuration is in a following section.

```
nameserver 8.8.8.8
nameserver 8.8.4.4
```

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:

```
ip addr flush eth0
```

> **Note**
> Flushing the IP configuration using the `ip` command does not clear the contents of `/etc/resolv.conf`. You must remove or modify those entries manually (or re-boot), which should also cause `/etc/resolv.conf`, which is a symlink to `/run/systemd/resolve/stub-resolv.conf`, to be re-written.

Dynamic IP address assignment (DHCP client)

To configure your server to use DHCP for dynamic address assignment, create a Netplan configuration in the file `/etc/netplan/99_config.yaml`. The following example assumes you are configuring your first Ethernet interface identified as `enp3s0`.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp3s0:
      dhcp4: true
```

The configuration can then be applied using the `netplan` command:

```
sudo netplan apply
```

Static IP address assignment

To configure your system to use static address assignment, create a `netplan` configuration in the file `/etc/netplan/99_config.yaml`. The example below assumes you are configuring your first Ethernet interface identified as `eth0`. Change the `addresses`, `routes`, and `nameservers` values to meet the requirements of your network.

```
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      addresses:
        - 10.10.10.2/24
      routes:
        - to: default
          via: 10.10.10.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [10.10.10.1, 1.1.1.1]
```

The configuration can then be applied using the `netplan` command.

```
sudo netplan apply
```

> **NOTE**
> `netplan` in Ubuntu Bionic 18.04 LTS doesn’t understand the “`to: default`” syntax to specify a default route, and should use the older `gateway4: 10.10.10.1` key instead of the whole `routes:` block.

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 `ip` command.

```
ip address show lo
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
       valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
       valid_lft forever preferred_lft forever
```

Name resolution

Name resolution (as it relates to IP networking) is the process of mapping hostnames to IP addresses, and vice-versa, making it easier to identify resources on a network. The following section will explain how to properly configure your system for name resolution using DNS and static hostname records.

DNS client configuration

Traditionally, the file `/etc/resolv.conf` was a static configuration file that rarely needed to be changed, or it automatically changed via DCHP client hooks. `systemd-resolved` handles nameserver configuration, and it should be interacted with through the `systemd-resolve` command. Netplan configures `systemd-resolved` to generate a list of nameservers and domains to put in `/etc/resolv.conf`, which is a symlink:

```
/etc/resolv.conf -> ../run/systemd/resolve/stub-resolv.conf
```

To configure the resolver, add the IP addresses of the appropriate nameservers for your network to the `netplan` configuration file. You can also add optional DNS suffix search-lists to match your network domain names. The resulting file might look like the following:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses:
        - 192.168.0.100/24
      routes:
        - to: default
          via: 192.168.0.1
      nameservers:
          search: [mydomain, otherdomain]
          addresses: [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

The *search* option can also be used with multiple domain names so that DNS queries will be appended in the order in which they are entered. For example, your network may have multiple sub-domains to search; a parent domain of *`example.com`*, and two sub-domains, *`sales.example.com`* and *`dev.example.com`*.

If you have multiple domains you wish to search, your configuration might look like the following:

```
network:
  version: 2
  renderer: networkd
  ethernets:
    enp0s25:
      addresses:
        - 192.168.0.100/24
      routes:
        - to: default
          via: 192.168.0.1
      nameservers:
          search: [example.com, sales.example.com, dev.example.com]
          addresses: [1.1.1.1, 8.8.8.8, 4.4.4.4]
```

If you try to ping a host with the name `server1`, your system will automatically query DNS for its Fully Qualified Domain Name (FQDN) in the following order:

1. `server1.example.com`
2. `server1.sales.example.com`
3. `server1.dev.example.com`

If no matches are found, the DNS server will provide a result of *notfound* and the DNS query will fail.

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. In some configurations, especially when Internet access is not required, servers that communicate with a limited number of resources can be conveniently set to use static hostnames instead of 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 server1.example.com vpn
10.0.0.12   server2 server2.example.com mail
10.0.0.13   server3 server3.example.com www
10.0.0.14   server4 server4.example.com file
```

> **Note**
> In this example, notice that each of the servers were given aliases in addition to

> their proper names and FQDN’s. *Server1* has been mapped to the name *vpn*, *server2* is referred to as *mail*, *server3* as *www*, and *server4* as *file*.

Name Service Switch (NSS) 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`. As mentioned in the previous section, 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
```

- **`files`** first tries to resolve static hostnames located in `/etc/hosts`.
- **`mdns4_minimal`** attempts to resolve the name using Multicast DNS.
- **`[NOTFOUND=return]`** means that any response of `notfound` by the preceding `mdns4_minimal` process should be treated as authoritative and that the system should not try to continue hunting for an answer.
- **`dns`** represents a legacy unicast DNS query.
- **mdns4** represents a multicast DNS query.

To modify the order of these name resolution methods, you can simply change the `hosts:` string to the value of your choosing. For example, if you prefer to use legacy unicast DNS versus multicast DNS, you can change the string in `/etc/nsswitch.conf` as shown below:

```
hosts:          files dns [NOTFOUND=return] mdns4_minimal mdns4
```

Bridging multiple interfaces

Bridging 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:

Configure the bridge by editing your `netplan` configuration found in `/etc/netplan/`, entering the appropriate values for your physical interface and network:

```
network:
  version: 2
  renderer: networkd
```

```
    ethernets:
      enp3s0:
        dhcp4: no
    bridges:
      br0:
        dhcp4: yes
        interfaces:
          - enp3s0
```

Now apply the configuration to enable the bridge:

```
sudo netplan apply
```

The new bridge interface should now be up and running. The `brctl` provides useful information about the state of the bridge, controls which interfaces are part of the bridge, etc. See `man brctl` for more information.

networkd-dispatcher for hook scripts

Users of the former `ifupdown` may be familiar with using hook scripts (e.g., pre-up, post-up) in their interfaces file. Netplan configuration does not currently support hook scripts in its configuration definition.

Instead, to achieve this functionality with the `networkd` renderer, users can use networkd-dispatcher. The package provides both users and packages with hook points when specific network states are reached, to aid in reacting to network state.

> **Note**:
> If you are on Desktop (not Ubuntu Server) the network is driven by Network Manager - in that case you need NM Dispatcher scripts instead.

The Netplan FAQ has a great table that compares event timings between `ifupdown`/`systemd-networkd`/`network-manager`.

It is important to be aware that these hooks run asynchronously; i.e. they will not block transition into another state.

The Netplan FAQ also has an example on converting an old `ifupdown` hook to `networkd-dispatcher`.

Resources

- The Ubuntu Wiki Network page has links to articles covering more advanced network configuration.
- The Netplan website has additional examples and documentation.
- The Netplan man page has more information on Netplan.
- The systemd-resolved man page has more information on systemd-resolved service.
- For more information on *bridging* see the netplan.io examples page