

Linux Networking

Network Interface + DNS Tools

LinuXpert Systems

WhatsApp: 98841 65649, 90808 01382

E-mail: baskar@linuxpert.in

TABLE OF CONTENTS

1. ifconfig 1

2. ip 17

3. route 24

4. hostnamectl 32

5. host 47

6. dig 52

7. nslookup 58

8. resolvectl 65

1. ifconfig

The **`ifconfig`** command is a traditional Linux networking utility used to **configure, display, and manage network interfaces**. It can be used to view interface information, assign IP addresses, change interface status, configure network masks, and modify other interface parameters.

`ifconfig` was historically one of the primary Linux networking commands. On modern Linux systems, it has largely been replaced by the **`ip` command** from the **`iproute2`** package. However, **`ifconfig`** is still found on many systems and remains important for understanding older Linux installations and legacy administration practices.

On Ubuntu and many other distributions, **`ifconfig`** is provided by the **`net-tools`** package.

Syntax

```
ifconfig [interface]
```

For configuring an interface:

```
ifconfig interface [options] [address]
```

General format

```
ifconfig interface options
```

Example:

```
ifconfig eth0
```

or:

```
ifconfig eth0 192.168.1.100 netmask 255.255.255.0
```

Common Uses

Command	Purpose
`ifconfig`	Display information about all active interfaces
`ifconfig -a`	Display information about all interfaces
`ifconfig eth0`	Display information about a specific interface
`ifconfig eth0 up`	Enable an interface
`ifconfig eth0 down`	Disable an interface
`ifconfig eth0 IP`	Assign an IPv4 address
`ifconfig eth0 netmask MASK`	Configure subnet mask
`ifconfig eth0 mtu SIZE`	Configure MTU
`ifconfig eth0 hw ether MAC`	Change MAC address
`ifconfig eth0 broadcast ADDRESS`	Configure broadcast address

Frequently Used Commands

1. Display Network Interfaces

```
ifconfig
```

Example output:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.20  netmask 255.255.255.0
        broadcast 192.168.1.255
        ether 00:11:22:33:44:55
        RX packets 12543  bytes 1543210
        TX packets 10342  bytes 1234567
```

Displays information such as:

- Interface name
- Interface status
- IPv4 address
- Netmask
- Broadcast address
- MAC address
- MTU
- RX statistics

- TX statistics
- Packet errors
- Dropped packets

Important:

Without **`-a`,** traditional **`ifconfig`** normally displays only interfaces that are currently active.

---

2. Display All Network Interfaces

```
ifconfig -a
```

Unlike:

```
ifconfig
```

the **`-a`** option displays **all interfaces**, including interfaces that are currently down.

Example:

```
eth0
lo
wlan0
```

This is particularly useful when an interface does not appear in the normal **`ifconfig`** output.

---

3. Display a Specific Interface

```
ifconfig eth0
```

Example:

```
eth0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.1.20
        netmask 255.255.255.0
        broadcast 192.168.1.255
```

Displays configuration and statistics for **`eth0`**.

4. Bring a Network Interface Up

```
sudo ifconfig eth0 up
```

Enables the `eth0` interface.

Equivalent modern command:

```
sudo ip link set eth0 up
```

---

5. Bring a Network Interface Down

```
sudo ifconfig eth0 down
```

Disables the `eth0` interface.

Equivalent modern command:

```
sudo ip link set eth0 down
```

---

6. Assign an IP Address

```
sudo ifconfig eth0 192.168.1.100
```

Assigns:

```
192.168.1.100
```

to `eth0`.

The command can also be used with a netmask:

```
sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0
```

---

7. Configure IP Address and Netmask

```
sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0
```

This configures:

```
IP address : 192.168.1.100
Netmask    : 255.255.255.0
```

The CIDR equivalent is:
`192.168.1.100/24`

8. Configure Broadcast Address

```
sudo ifconfig eth0 broadcast 192.168.1.255
```

Sets the broadcast address of `eth0`.

You can specify IP address, netmask, and broadcast together:

```
sudo ifconfig eth0 192.168.1.100 \
    netmask 255.255.255.0 \
    broadcast 192.168.1.255
```

---

9. Change the MTU

MTU means **Maximum Transmission Unit**.

Display the current MTU:

```
ifconfig eth0
```

Example:

```
mtu 1500
```

Change it:

```
sudo ifconfig eth0 mtu 1400
```

This sets the MTU to:

```
1400 bytes
```

Modern equivalent:

```
sudo ip link set eth0 mtu 1400
```

---

10. Change the MAC Address

`ifconfig` can be used on systems where the driver supports it to change the hardware/MAC address.

First bring the interface down:

```
sudo ifconfig eth0 down
```

Change the MAC address:

```
sudo ifconfig eth0 hw ether 02:11:22:33:44:55
```

Bring the interface up:

```
sudo ifconfig eth0 up
```

Verify:

```
ifconfig eth0
```

Note:

Changing a MAC address may not be supported by every network driver, and NetworkManager or other network-management software may subsequently restore the original address.

---

11. Configure the Network Mask

```
sudo ifconfig eth0 netmask 255.255.255.0
```

Sets the subnet mask to:

```
255.255.255.0
```

Equivalent CIDR prefix:

```
/24
```

---

12. Configure Point-to-Point Address

For point-to-point interfaces, **`ifconfig`** supports the `pointopoint` option.

Example:

```
sudo ifconfig ppp0 10.0.0.1 pointopoint 10.0.0.2
```

This is primarily associated with interfaces such as PPP.

13. Display Interface Statistics

```
ifconfig eth0
```

Example:

```
RX packets  12543  bytes 1543210
RX errors   0      dropped 2
TX packets  10342  bytes 1234567
TX errors   0      dropped 0
```

The statistics provide information about:

RX

Receive statistics:

- Received packets
- Received bytes
- Receive errors
- Dropped packets

TX

Transmit statistics:

- Transmitted packets
- Transmitted bytes
- Transmit errors
- Dropped packets

These statistics can help diagnose network problems.

---

14. Display Loopback Interface

```
ifconfig lo
```

Typical output:

```
lo: flags=73<UP,LOOPBACK,RUNNING>  mtu 65536
        inet 127.0.0.1
        netmask 255.0.0.0
```

The loopback interface is normally named:

`lo`

Its IPv4 address is:

`127.0.0.1`

---

15. Configure Loopback Interface

Normally the loopback interface is automatically configured by the operating system.

For example:

```
sudo ifconfig lo 127.0.0.1
```

However, manually modifying the loopback interface is generally unnecessary and can disrupt local networking if done incorrectly.

---

16. Configure an Alias Interface

Traditional `ifconfig` supports interface aliases.

For example:

```
sudo ifconfig eth0:0 192.168.1.101
```

This creates an additional logical interface/address associated with **`eth0`**.

Another example:

```
sudo ifconfig eth0:1 192.168.1.102
```

Historically this was used to assign multiple IP addresses to an interface.

Modern Linux practice: use **`ip addr add`** instead:

```
sudo ip addr add 192.168.1.101/24 dev eth0
```

17. Remove an IP Address

Traditional `ifconfig` **does not** provide a direct equivalent to:

```
ip addr del 192.168.1.100/24 dev eth0
```

Instead, administrators historically used commands such as:

```
sudo ifconfig eth0 0.0.0.0
```

or:

```
sudo ifconfig eth0 down
```

depending on the intended result.

For modern Linux systems, **`ip addr del`** or **`ip addr flush`** is preferable.

---

18. Flush Interface Configuration

`ifconfig` does **not** have a direct equivalent to the modern:

```
sudo ip addr flush dev eth0
```

The **`ip`** command provides much more precise address-management capabilities.

This is one of the important differences between **`ifconfig`** and **`ip`**.

Important Options

Option	Meaning
`-a`	Display all interfaces
`up`	Enable interface
`down`	Disable interface
`netmask`	Specify subnet mask
`broadcast`	Specify broadcast address
`mtu`	Set Maximum Transmission Unit
`hw ether`	Set hardware/MAC address
`pointopoint`	Configure point-to-point address
`add`	Add an address in supported configurations
`del`	Delete an address in supported configurations

Practical Examples

View all active interfaces

```
Ifconfig
```

View all interfaces, including inactive ones

```
ifconfig -a
```

Check `eth0`

```
ifconfig eth0
```

Check Wi-Fi interface

```
ifconfig wlan0
```

Enable Ethernet

```
sudo ifconfig eth0 up
```

Disable Ethernet

```
sudo ifconfig eth0 down
```

Assign an IP address

```
sudo ifconfig eth0 192.168.1.100
```

Assign IP address and netmask

```
sudo ifconfig eth0 192.168.1.100 netmask 255.255.255.0
```

Set broadcast address

```
sudo ifconfig eth0 broadcast 192.168.1.255
```

Change MTU

```
sudo ifconfig eth0 mtu 1400
```

Display interface statistics

```
ifconfig eth0
```

Change MAC address

```
sudo ifconfig eth0 down
sudo ifconfig eth0 hw ether 02:11:22:33:44:55
sudo ifconfig eth0 up
```

`ifconfig` (vs) `ip`

Task	ifconfig	Modern ip
Display interfaces	`ifconfig`	`ip link`
Display IP addresses	`ifconfig`	`ip addr`
Display all interfaces	`ifconfig -a`	`ip link show`
Enable interface	`ifconfig eth0 up`	`ip link set eth0 up`
Disable interface	`ifconfig eth0 down`	`ip link set eth0 down`
Assign IP	`ifconfig eth0 192.168.1.100`	`ip addr add 192.168.1.100/24 dev eth0`
Configure netmask	`ifconfig eth0 netmask ...`	CIDR prefix in ip addr
Configure MTU	`ifconfig eth0 mtu 1400`	`ip link set eth0 mtu 1400`
Change MAC	`ifconfig eth0 hw ether ...`	`ip link set eth0 address ...`
Remove IP	Limited/indirect	`ip addr del ...`
Flush IP addresses	No direct equivalent	`ip addr flush dev eth0`

Task	ifconfig	Modern ip
Routing table	Not its primary function	`ip route`
ARP/Neighbour table	Not its primary function	`ip neigh`
Routing rules	No	`ip rule`
Network monitoring	No comparable unified command	`ip monitor`

Installation

On Debian/Ubuntu systems, **`ifconfig`** is normally supplied by the **`net-tools`** package.

Install it with:

```
sudo apt install net-tools
```

Check its location:

```
which ifconfig
```

Typical output:

```
/usr/sbin/ifconfig
```

Check the version/help information:

```
ifconfig --help
```

or:

```
man ifconfig
```

Important Limitations of `ifconfig`

Although **`ifconfig`** remains useful for learning and legacy administration, it has several limitations compared with **`ip`**.

---

1. It is primarily an interface configuration tool

`ifconfig` does not provide the comprehensive networking object model offered by **`ip`**.

2. Routing is handled separately

For example, traditional Linux systems use:

```
route
```

whereas modern Linux uses:

```
ip route
```

---

3. Neighbour/ARP management is separate

Traditional command:

```
arp
```

Modern command:

```
ip neigh
```

---

4. Limited IPv6 and advanced networking functionality

The `ip` command provides much more comprehensive support for:

- IPv4
- IPv6
- Routing
- Policy routing
- Network namespaces
- Tunnels
- VLANs
- Bridges
- Neighbours
- Multicast
- Interface properties

Common `ifconfig` Troubleshooting Examples

Check whether an interface exists

```
ifconfig -a
```

Check whether an interface is UP

```
ifconfig eth0
```

Look for:

```
UP
RUNNING
```

Check the assigned IP address

```
ifconfig eth0
```

Look for:

```
inet 192.168.1.20
```

Check MAC address

```
ifconfig eth0
```

Look for:

```
ether 00:11:22:33:44:55
```

Check packet errors

```
ifconfig eth0
```

Look for:

```
RX errors
TX errors
```

and:
```
dropped
```

Check MTU

```
ifconfig eth0
```

Look for:

```
mtu 1500
```

Important Note for Modern Linux

`ifconfig` should now generally be considered a **legacy networking command**.

The recommended modern approach is:

```
ifconfig       → ip addr / ip link
route          → ip route
arp            → ip neigh
netstat        → ss / ip
```

For example:

Old:

```
ifconfig eth0 192.168.1.100 netmask 255.255.255.0
```

Modern:

```
sudo ip addr add 192.168.1.100/24 dev eth0
```

Similarly:

Old:

```
ifconfig eth0 up
```

Modern:

```
sudo ip link set eth0 up
```

Summary

The **`ifconfig`** command is a traditional Linux utility for **viewing and configuring network interfaces**. It can display IP/MAC addresses and interface statistics, enable or disable interfaces, configure IP addresses, subnet masks, broadcast addresses and MTU values, and perform certain hardware-level interface configurations.

Although **`ifconfig`** is still useful for **Linux education, legacy systems, troubleshooting and understanding historical Linux networking**, the **`ip` command is the preferred modern tool** because it provides a unified and much more powerful interface for managing Linux networking.

Key commands to remember

```
ifconfig

ifconfig -a

ifconfig eth0

sudo ifconfig eth0 up

sudo ifconfig eth0 down

sudo ifconfig eth0 192.168.1.100

sudo ifconfig eth0 netmask 255.255.255.0

sudo ifconfig eth0 mtu 1400
```

2. ip

The **`ip`** command is a powerful Linux networking utility used to **configure, monitor, and troubleshoot network interfaces, IP addresses, routing tables, ARP tables, and tunnels**. It is part of the **iproute2** package and is the modern replacement for older networking commands such as **`ifconfig`**, **`route`**, **`arp`**, and **`netstat`** (for many tasks).

Syntax

```
ip [OPTIONS] OBJECT COMMAND
```

- **OBJECT** – The networking object you want to work with.
- **COMMAND** – The action to perform.

General format:

```
ip object command
```

Example:

```
ip addr show
```

Common Objects

Object	Purpose
addr	Manage IP addresses
link	Manage network interfaces
route	Manage routing table
neigh	Manage ARP/Neighbour table
rule	Manage routing rules
tunnel	Manage IP tunnels
maddr	Manage multicast addresses
monitor	Monitor network changes

Frequently Used Commands

1. Display IP Addresses

```
ip addr
```

or

```
ip address show
```

Example output:

```
2: eth0:
    inet 192.168.1.20/24
```

Shows

- IPv4 address
- IPv6 address
- MAC address
- Interface status

Shortcut:

```
ip a
```

---

2. Show Network Interfaces

```
ip link
```

Shortcut:

```
ip l
```

Example:

```
2: eth0: <UP,BROADCAST,RUNNING>
```

Displays

- Interface name
- MTU
- MAC address
- Interface state

3. Bring Interface Up

```
sudo ip link set eth0 up
```

---

4. Bring Interface Down

```
sudo ip link set eth0 down
```

---

5. Assign an IP Address

```
sudo ip addr add 192.168.1.100/24 dev eth0
```

Adds

```
192.168.1.100
```

to interface

```
eth0
```

---

6. Remove an IP Address

```
sudo ip addr del 192.168.1.100/24 dev eth0
```

---

7. Show Routing Table

```
ip route
```

Shortcut:

```
ip r
```

Example:

```
default via 192.168.1.1 dev eth0
```

Shows

- Default gateway
- Network routes
- Gateway information

8. Add a Route

```
sudo ip route add 10.0.0.0/24 via 192.168.1.1
```

---

9. Delete a Route

```
sudo ip route del 10.0.0.0/24
```

---

10. Display ARP/Neighbour Table

```
ip neigh
```

Example:

```
192.168.1.1 dev eth0 lladdr 00:11:22:33:44:55 REACHABLE
```

Shows

- IP address
- MAC address
- Status

---

11. Monitor Network Changes

```
ip monitor
```

Displays changes such as:

- Interface up/down
- Address changes
- Route changes

---

12. Display Statistics

```
ip -s link
```

Shows

- RX packets
- TX packets

- Errors
- Dropped packets

---

13. Show Only IPv4 Addresses

```
ip -4 addr
```

---

14. Show Only IPv6 Addresses

```
ip -6 addr
```

---

15. Show Specific Interface

```
ip addr show eth0
```

or

```
ip a show eth0
```

---

16. Flush All IP Addresses

```
sudo ip addr flush dev eth0
```

Removes all IP addresses assigned to the interface.

Common Shortcuts

Full Command	Shortcut
ip address	ip a
ip link	ip l
ip route	ip r
ip neighbour	ip n

Important Options

Option	Meaning
`-4`	IPv4 only
`-6`	IPv6 only
`-s`	Show statistics
`-d`	Detailed information
`-br`	Brief output

Example:

```
ip -br addr
```

Output:

```
lo        UNKNOWN 127.0.0.1/8
eth0      UP      192.168.1.20/24
wlan0     DOWN
```

Practical Examples

View all network interfaces

```
ip link show
```

Check your IP address

```
ip addr show
```

Display the default gateway

```
ip route
```

Enable a network interface

```
sudo ip link set wlan0 up
```

Disable Wi-Fi interface

```
sudo ip link set wlan0 down
```

Add a temporary IP address

```
sudo ip addr add 10.0.0.50/24 dev eth0
```

Remove the temporary IP

```
sudo ip addr del 10.0.0.50/24 dev eth0
```

Advantages of `ip` Over Older Commands

Old Command	Modern ip Equivalent
`ifconfig`	`ip addr`, `ip link`
`route`	`ip route`
`arp`	`ip neigh`
`netstat -rn`	`ip route`
`ifconfig eth0 up`	`ip link set eth0 up`

Common Use Cases

- Viewing IP and MAC addresses
- Enabling or disabling network interfaces
- Configuring temporary IP addresses
- Viewing and modifying routing tables
- Managing ARP (neighbour) entries
- Monitoring network configuration changes
- Troubleshooting connectivity issues
- Managing IPv4 and IPv6 networking

Summary

The **`ip`** command is the standard Linux tool for network configuration and diagnostics. It combines the functionality of several older networking utilities into a single, consistent interface, making it the preferred command for managing interfaces, IP addresses, routes, neighbour tables, and other networking components on modern Linux systems.

3. route

The **`route`** command is a traditional Linux networking utility used to **display and manipulate the IP routing table**. It allows administrators to view how packets are routed between the local system and remote networks, and to add, delete, or modify routing entries.

It is part of the older **net-tools** package. On modern Linux systems, the **`ip route`** command from the **`iproute2`** package is preferred and provides more advanced routing capabilities.

Syntax

```
route [OPTIONS] [COMMAND] [TARGET] [OPTIONS]
```

Common forms include:

```
route
route -n
route -e
route add ...
route del ...
```

Main Uses

The route command can be used to:

- Display the kernel's IPv4 routing table
- Add a static route
- Delete an existing route
- Configure a default gateway
- Route traffic through a specific network interface
- Specify a gateway for a particular destination network
- Troubleshoot routing problems

1. Display the Routing Table

```
route
```

Example:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
default         192.168.1.1     0.0.0.0         UG    100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

Important columns

Column	Meaning
Destination	Destination network or host
Gateway	Gateway used to reach the destination
Genmask	Network mask
Flags	Properties of the route
Metric	Route preference/cost
Ref	Reference count
Use	Number of times route has been used
Iface	Network interface used

---

2. Display Routing Table Without DNS Resolution

```
route -n
```

Example:

```
Kernel IP routing table
Destination     Gateway         Genmask         Flags Metric Ref    Use Iface
0.0.0.0         192.168.1.1     0.0.0.0         UG    100    0        0 eth0
192.168.1.0     0.0.0.0         255.255.255.0   U     100    0        0 eth0
```

The `-n` option prevents **`route`** from attempting to resolve IP addresses into hostnames.

This is particularly useful when troubleshooting because it is **faster and avoids DNS-related delays**.

3. Display Extended Routing Information

```
route -e
```

This displays additional information such as the interface and metric.

Example:

```
Kernel IP routing table
Destination Gateway Genmask Flags Metric Ref Use Iface
default 192.168.1.1 0.0.0.0 UG 100 0 0 eth0
192.168.1.0 0.0.0.0 255.255.255.0 U 100 0 0 eth0
```

Adding Routes

4. Add a Route to a Network

```
sudo route add -net 192.168.10.0 netmask 255.255.255.0 dev eth0
```

This tells the system:

To reach `192.168.10.0/24`, use `eth0`.

Equivalent modern command:

```
sudo ip route add 192.168.10.0/24 dev eth0
```

---

5. Add a Route Through a Gateway

```
sudo route add -net 192.168.10.0 netmask 255.255.255.0 gw 192.168.1.254
```

Traffic destined for `192.168.10.0/24` will be sent through gateway:

```
192.168.1.254
```

Modern equivalent:

```
sudo ip route add 192.168.10.0/24 via 192.168.1.254
```

6. Add a Host Route

A route can be created for a **single IP address**.

```
sudo route add -host 192.168.10.50 gw 192.168.1.254
```

This creates a route specifically for:

```
192.168.10.50
```

Modern equivalent:

```
sudo ip route add 192.168.10.50 via 192.168.1.254
```

Default Gateway

7. Add a Default Route

```
sudo route add default gw 192.168.1.1
```

This establishes:

```
Destination: 0.0.0.0
Gateway:     192.168.1.1
```

A default route is used when no more specific route exists.

Modern equivalent:

```
sudo ip route add default via 192.168.1.1
```

---

8. Add Default Route Through an Interface

```
sudo route add default dev eth0
```

This tells the kernel to use `eth0` for the default route.

Deleting Routes

9. Delete a Network Route

```
sudo route del -net 192.168.10.0 netmask 255.255.255.0
```

Modern equivalent:

```
sudo ip route del 192.168.10.0/24
```

---

10. Delete a Host Route

```
sudo route del -host 192.168.10.50
```

---

11. Delete the Default Gateway

```
sudo route del default
```

or:

```
sudo route del default gw 192.168.1.1
```

Modern equivalent:

```
sudo ip route del default
```

Useful Route Flags

The `Flags` column can contain several letters.

Flag	Meaning
U	Route is up
G	Route uses a gateway
H	Route is to a host
D	Route was dynamically created
M	Route was modified
R	Route was reinstated
!	Reject route

For example:

`UG`

means:

- `U` → route is up
- `G` → route uses a gateway

A typical default route therefore appears as:

```
default  192.168.1.1  0.0.0.0  UG
```

Practical Examples

Example 1: Find the Default Gateway

```
route -n
```

Look for:

```
Destination  Gateway
0.0.0.0      192.168.1.1
```

Here:

```
Default Gateway = 192.168.1.1
```

Example 2: Find Which Interface Handles a Network

```
route -n
```

For:

```
192.168.10.0  0.0.0.0  255.255.255.0  U  100  0  0  eth1
```

the network:

```
192.168.10.0/24
```

is reached through:

```
eth1
```

Example 3: Route a Private Network Through Another Router

Suppose the system has:

```
Local network:    192.168.1.0/24
Gateway:          192.168.1.1
Remote network:   10.10.10.0/24
Remote router:    192.168.1.254
```

Add:

```
sudo route add -net 10.10.10.0 netmask 255.255.255.0 gw 192.168.1.254
```

Now traffic to:

```
10.10.10.0/24
```

will be forwarded to:

```
192.168.1.254
```

`route` (vs) `ip route`

For modern Linux administration, **`ip route`** should generally be preferred.

Task	Traditional	Modern
Display routes	`route`	`ip route`
Display numeric routes	`route -n`	`ip route`
Add network route	`route add -net ...`	`ip route add ...`
Delete route	`route del ...`	`ip route del ...`
Add default route	`route add default gw ...`	`ip route add default via ...`
Route to host	`route add -host ...`	`ip route add <IP>/32 ...`

For example:

```
route -n
```

is normally replaced by:

```
ip route
```

and:

```
sudo route add -net 10.10.10.0 netmask 255.255.255.0 gw 192.168.1.254
```

is replaced by:

```
sudo ip route add 10.10.10.0/24 via 192.168.1.254
```

Important note

`route` is primarily a **legacy IPv4 routing utility**. The **`ip`** command supports IPv4 and IPv6 and provides substantially more powerful routing functionality, including policy routing, multiple routing tables, route metrics, ECMP, and other advanced networking features.

For a modern Linux networking use **`route` as a traditional / legacy command** and learn **`ip route` as the current standard**.

4. hostnamectl

The **`hostnamectl`** command is a Linux system administration utility used to **view and configure the system hostname and related system information**. It is part of the **systemd** system and is commonly available on modern Linux distributions such as Ubuntu, Debian, Fedora, RHEL, Rocky Linux, and Arch Linux.

It provides a convenient way to manage the system's **static hostname, transient hostname, pretty hostname, and machine identification information**.

Syntax

```
hostnamectl [OPTIONS] [COMMAND]
```

Common commands

```
hostnamectl

hostnamectl status

hostnamectl hostname

hostnamectl hostname NEW_HOSTNAME

hostnamectl set-hostname NEW_HOSTNAME

hostnamectl set-hostname NEW_HOSTNAME --static

hostnamectl set-hostname NEW_HOSTNAME --pretty

hostnamectl set-hostname NEW_HOSTNAME --transient
```

1. Display hostname and system information

Simply running **`hostnamectl`** displays information about the system.

```
hostnamectl
```

Example:

```
 Static hostname: guru-linux
       Icon name: computer
         Chassis: desktop
      Machine ID: 8a7c9d...
         Boot ID: 3f21ab...
Operating System: Ubuntu 22.04.5 LTS
          Kernel: Linux 5.15.0-139-generic
    Architecture: x86-64
 Hardware Vendor: HP
  Hardware Model: ProLiant DL380 Gen10 Plus
Firmware Version: U30
```

This is equivalent to:

```
hostnamectl status
```

---

2. Display only the hostname

```
hostnamectl hostname
```

Example:

```
guru-linux
```

This is useful in scripts when only the hostname is required.

---

3. Change the hostname

The simplest way to change the system hostname is:

```
sudo hostnamectl set-hostname guru-server
```

Verify:

```
hostnamectl
```

or:

```
hostname
```

The new hostname normally takes effect immediately without requiring a reboot.

---

4. Set a static hostname

A **static hostname** is the persistent hostname stored by the system.

```
sudo hostnamectl set-hostname guru-server --static
```

or:

```
sudo hostnamectl set-hostname guru-server
```

The static hostname is normally stored in:

```
/etc/hostname
```

You can examine it with:

```
cat /etc/hostname
```

---

5. Set a pretty hostname

A **pretty hostname** is a descriptive, human-friendly name. Unlike a conventional hostname, it can contain spaces and other characters.

Example:

```
sudo hostnamectl set-hostname "Guru Linux Educational Server" --pretty
```

Display it:

```
hostnamectl
```

You may see:

```
Pretty hostname: Guru Linux Educational Server
```

The pretty hostname is intended primarily for presentation to users.

6. Set a transient hostname

A **transient hostname** is a temporary hostname that can be supplied dynamically, for example through network configuration.

You can set it with:

```
sudo hostnamectl set-hostname temporary-server --transient
```

Check:

```
hostnamectl
```

A transient hostname generally does **not** provide the same persistent configuration as a static hostname.

---

7. Set all hostname types

You can specify multiple hostname types:

```
sudo hostnamectl set-hostname guru-server --static --pretty
```

For example:

```
sudo hostnamectl set-hostname "Guru Linux Server" --pretty
```

and:

```
sudo hostnamectl set-hostname guru-server --static
```

---

8. Reset the hostname

To remove the configured static hostname:

```
sudo hostnamectl set-hostname ""
```

This resets the hostname configuration so that the system can obtain a hostname from another source, if available.

9. Display operating-system information

`hostnamectl` can also display OS information:

```
hostnamectl
```

Typical information includes:

```
Operating System: Ubuntu 22.04.5 LTS
Kernel: Linux 5.15.0-139-generic
Architecture: x86-64
```

For a more focused view, you can also use:

```
hostnamectl status
```

---

10. Display kernel information

The kernel information is included in:

```
hostnamectl
```

For example:

```
Kernel: Linux 5.15.0-139-generic
```

For detailed kernel information, however, the traditional command is:

```
uname -a
```

---

11. Display machine architecture

`hostnamectl` displays the architecture:

```
hostnamectl
```

Example:

```
Architecture: x86-64
```

Another command is:

```
uname -m
```

Example:

```
x86_64
```

12. Display chassis information

On systems where the information is available:

```
hostnamectl
```

may display:

```
Chassis: desktop
```

Possible chassis types include:

```
desktop
laptop
server
tablet
handset
vm
container
```

This information is obtained from system hardware/virtualization information.

---

13. Display hardware vendor and model

On supported systems:

```
hostnamectl
```

may show:

```
Hardware Vendor: HP
Hardware Model: ProLiant DL380 Gen10 Plus
```

This can be useful for quickly identifying a physical server.

---

14. Display machine ID

`hostnamectl` can display the unique **machine ID**:

```
hostnamectl
```

Example:

```
Machine ID: 8a7c9d2b4e...
```

The machine ID is normally associated with:

```
/etc/machine-id
```

You can also display it using:

```
cat /etc/machine-id
```

Important

The machine ID is **not the same as the hostname**.

The hostname identifies the system on a network or to users, while the machine ID is a persistent identifier for the installed system.

---

15. Display boot ID

`hostnamectl` can display the current boot ID:

```
hostnamectl
```

Example:

```
Boot ID: 3f21ab...
```

The boot ID changes each time the system boots.

It can also be obtained with:

```
cat /proc/sys/kernel/random/boot_id
```

---

16. Change hostname and update `/etc/hostname`

For example:

```
sudo hostnamectl set-hostname webserver01
```

Then:

```
cat /etc/hostname
```

Output:

```
webserver01
```

This is preferable to manually editing `/etc/hostname` on a system managed by systemd.

17. Check the current hostname

There are several ways to check the hostname:

```
hostname
```

```
hostnamectl hostname
```

```
cat /etc/hostname
```

The first two are particularly convenient for command-line administration.

---

18. Hostname and `/etc/hosts`

Changing the hostname does not necessarily mean that every hostname-related configuration has been automatically changed.

For example:

```
hostnamectl set-hostname server01
```

You should inspect:

```
cat /etc/hostname
```

and:

```
cat /etc/hosts
```

A typical `/etc/hosts` file may contain:

```
127.0.0.1       localhost
127.0.1.1       server01
```

For servers, DNS and **`/etc/hosts`** configuration should be considered separately from the system hostname.

19. Hostname containing a domain name

A hostname can be configured as:

```
sudo hostnamectl set-hostname server01.example.com
```

Check:

```
hostnamectl hostname
```

Output:

```
server01.example.com
```

However, it is important to distinguish between a **hostname/FQDN** and the DNS configuration of the machine. Setting a hostname does not automatically create a DNS record.

---

20. Change hostname on a server

For example, suppose a server currently has:

```
ubuntu
```

and you want to rename it:

```
sudo hostnamectl set-hostname digimat-server
```

Verify:

```
hostnamectl
```

Then:

```
hostname
```

Output:

```
digimat-server
```

---

21. Change hostname on a desktop

Example:

```
sudo hostnamectl set-hostname guru-desktop
```

Verify:

```
hostnamectl status
```

22. Change hostname for a laboratory computer

For an educational laboratory, computers could be named systematically:

```
sudo hostnamectl set-hostname lab-pc01
```

Other machines:

```
lab-pc01
lab-pc02
lab-pc03
...
lab-pc50
```

This makes system identification much easier when managing a computer laboratory.

---

23. Change hostname remotely using SSH

Suppose you are connected to a remote server:

```
ssh admin@192.168.1.50
```

You can change its hostname:

```
sudo hostnamectl set-hostname server01
```

Then verify:

```
hostnamectl hostname
```

Be aware that your existing SSH session may continue to work even though the hostname has changed.

---

24. Using `hostnamectl` with systemd

`hostnamectl` communicates with the systemd hostnamed service, commonly:

```
systemd-hostnamed
```

You can check its status with:

```
systemctl status systemd-hostnamed
```

On many systems, **`systemd-hostnamed`** is activated on demand rather than continuously running.

25. Help

Display help:

```
hostnamectl --help
```

or:

```
hostnamectl -h
```

This displays the available commands and options.

---

26. Man page

For detailed documentation:

```
man hostnamectl
```

You can search within the manual:

```
/hostname
```

or:

```
/set-hostname
```

---

27. Common options

Some useful options include:

Option	Purpose
`-h`	Display help
`--help`	Display help
`--version`	Display version information
`--no-ask-password`	Do not prompt for authentication
`--static`	Work with the static hostname
`--transient`	Work with the transient hostname
`--pretty`	Work with the pretty hostname
`--json`	Produce JSON-formatted output on supported systemd versions

28. JSON output

On sufficiently recent systemd versions, structured output can be requested:

```
hostnamectl --json=pretty
```

or:

```
hostnamectl --json=short
```

This can be useful when hostname/system information needs to be processed by scripts or other programs.

---

29. Useful hostname-related commands

`hostnamectl` is not the only hostname utility in Linux.

`hostname`

Display or temporarily change the hostname:

```
hostname
```

`hostnamectl`

Systemd-based hostname management:

```
hostnamectl
```

`uname`

Display kernel and system information:

```
uname -a
```

`dnsdomainname`

Display the DNS domain name:

```
dnsdomainname
```

`domainname`

Display or configure the NIS domain name:

```
domainname
```

30. `hostname` (vs) `hostnamectl`

Feature	`hostname`	`hostnamectl`
Display hostname	Yes	Yes
Persistent hostname management	Limited	Yes
Static hostname	Basic	Yes
Pretty hostname	No	Yes
Transient hostname	No	Yes
Machine ID	No	Yes
Boot ID	No	Yes
OS information	No	Yes
Kernel information	No	Yes
Hardware information	No	Yes
systemd integration	No	Yes

For modern systemd-based distributions, **`hostnamectl` is generally the preferred tool for persistent hostname administration**.

---

31. Practical examples

Example 1 — Find the current hostname

```
hostnamectl hostname
```

Example 2 — Rename a computer

```
sudo hostnamectl set-hostname lab-pc01
```

Example 3 — Rename a server

```
sudo hostnamectl set-hostname webserver01
```

Example 4 — Set a pretty hostname

```
sudo hostnamectl set-hostname "Guru Linux Training Server" –pretty
```

Example 5 — Set a static hostname

```
sudo hostnamectl set-hostname guru-server –static
```

Example 6 — Set a transient hostname

```
sudo hostnamectl set-hostname temporary-server –transient
```

Example 7 — Display complete information

```
hostnamectl status
```

Example 8 — Display machine ID

```
hostnamectl | grep "Machine ID"
```

Example 9 — Display kernel information

```
hostnamectl | grep "Kernel"
```

Example 10 — Display operating system

```
hostnamectl | grep "Operating System"
```

Summary

`hostnamectl` is a **systemd-based utility for managing and inspecting the hostname and basic system identity information**.

The most important commands to remember are:

```
hostnamectl
```

Display hostname and system information.

```
hostnamectl hostname
```

Display only the hostname.

```
sudo hostnamectl set-hostname NEW_HOSTNAME
```

Change the persistent hostname.

```
sudo hostnamectl set-hostname "Pretty Name" --pretty
```

Set a human-readable hostname.

```
hostnamectl --help
```

Display available options.

```
man hostnamectl
```

Read the complete manual.

Quick reference

```
hostnamectl
hostnamectl status
hostnamectl hostname
sudo hostnamectl set-hostname NAME
sudo hostnamectl set-hostname NAME --static
sudo hostnamectl set-hostname "PRETTY NAME" --pretty
sudo hostnamectl set-hostname NAME --transient
hostnamectl --help
hostnamectl --version
man hostnamectl
```

Key point: `hostnamectl` is more than a command for changing the hostname—it provides a unified interface for **hostname management and basic system identification on systemd-based Linux systems**.

5. host

The **`host`** command is a **DNS (Domain Name System) lookup utility** used to find information about domain names and IP addresses. It helps network administrators and users troubleshoot DNS-related issues.

Syntax

```
host [OPTIONS] NAME [SERVER]
```

- **NAME** – Domain name or IP address to query.
- **SERVER** *(optional)* – Specify the DNS server to use instead of the system's default.

Common Uses

1. Find the IP address of a domain

```
host google.com
```

Example Output

```
google.com has address 142.250.183.206
google.com has IPv6 address 2404:6800:4009:81c::200e
google.com mail is handled by 10 smtp.google.com.
```

Explanation

- Shows the IPv4 address (A record).
- Shows the IPv6 address (AAAA record).
- Shows the mail server (MX record).

---

2. Reverse DNS Lookup (IP → Hostname)

```
host 8.8.8.8
```

Example Output

```
8.8.8.8.in-addr.arpa domain name pointer dns.google.
```

This tells you the hostname associated with the IP address.

3. Query a Specific DNS Server

```
host google.com 8.8.8.8
```

Queries Google's public DNS server instead of the default system DNS server.

---

4. Display MX (Mail Exchange) Records

```
host -t MX gmail.com
```

Example:

```
gmail.com mail is handled by 5 gmail-smtp-in.l.google.com.
```

Shows the mail servers responsible for receiving email.

---

5. Display NS (Name Server) Records

```
host -t NS example.com
```

Example:

```
example.com name server a.iana-servers.net.
example.com name server b.iana-servers.net.
```

Lists the authoritative DNS servers.

---

6. Display TXT Records

```
host -t TXT google.com
```

TXT records often contain:

- SPF records
- Domain verification
- Security information

---

7. Display SOA (Start of Authority)

```
host -t SOA example.com
```

Example output:

```
example.com has SOA record sns.dns.icann.org. noc.dns.icann.org. ...
```

Shows information about the DNS zone administrator and zone configuration.

---

8. Display IPv6 Address

```
host -t AAAA google.com
```

Returns only IPv6 addresses.

---

9. Display IPv4 Address

```
host -t A google.com
```

Returns only IPv4 addresses.

Common DNS Record Types

Record	Purpose
A	IPv4 address
AAAA	IPv6 address
MX	Mail server
NS	Name server
TXT	Text information (SPF, verification, etc.)
SOA	Start of Authority
CNAME	Alias for another domain
PTR	Reverse DNS (IP → Hostname)

Useful Options

Option	Description
`-t TYPE`	Specify DNS record type (A, MX, NS, TXT, etc.)
`-a`	Query all available record types (verbose)
`-v`	Verbose output
`-W TIME`	Set query timeout (seconds)
`-R NUMBER`	Number of retries

Practical Examples

Check a website's IP

```
host openai.com
```

Check mail servers

```
host -t MX yahoo.com
```

Find authoritative DNS servers

```
host -t NS ubuntu.com
```

Reverse lookup

```
host 1.1.1.1
```

Use Cloudflare DNS

```
host google.com 1.1.1.1
```

Install `host`

Ubuntu/Debian

```
sudo apt install dnsutils
```

RHEL/CentOS/Rocky Linux

```
sudo dnf install bind-utils
```

or

```
sudo yum install bind-utils
```

Exit Status

Exit Code	Meaning
0	Query successful
1	Usage error
2	DNS lookup failed
3	DNS server returned an error

When to Use `host`

Use `host` when you need to:

- Verify that a domain resolves to the correct IP address.
- Troubleshoot DNS resolution problems.
- Find the mail servers (MX records) for a domain.
- Identify the authoritative name servers.
- Perform reverse DNS lookups (IP to hostname).
- Query specific DNS record types.

`host` (vs) `dig` (vs) `nslookup`

Feature	host	dig	nslookup
Ease of use			
Detailed DNS information			
Reverse lookups			
Script-friendly			Limited
Commonly used for troubleshooting			Legacy

Summary:

- **`host`** is ideal for quick, simple DNS lookups.
- **`dig`** provides detailed DNS information and is preferred for advanced troubleshooting.
- **`nslookup`** is older and still available on many systems, but **`host`** and **`dig`** are generally recommended on modern Linux distributions.

6. dig

The **`dig`** (Domain Information Groper) command is a Linux/Unix network utility used to **query DNS (Domain Name System) servers**. It helps you retrieve DNS records such as IP addresses, mail servers, name servers, and more.

It is one of the most commonly used tools by system administrators and network engineers for DNS troubleshooting.

Syntax

```
dig [OPTIONS] [DOMAIN] [RECORD_TYPE]
```

or

```
dig @DNS_SERVER DOMAIN RECORD_TYPE
```

- `@DNS_SERVER` – Specifies the DNS server to query.
- `DOMAIN` – The domain name to look up.
- `RECORD_TYPE` – Type of DNS record (A, AAAA, MX, NS, TXT, etc.).

Common DNS Record Types

Record	Purpose
A	Maps a domain to an IPv4 address
AAAA	Maps a domain to an IPv6 address
MX	Mail server records
NS	Name server records
TXT	Text records (SPF, DKIM, verification)
CNAME	Alias to another hostname
SOA	Start of Authority record
PTR	Reverse DNS lookup

Examples

1. Lookup a Domain

```
dig google.com
```

Example output:

```
;; QUESTION SECTION:
;google.com.    IN    A

;; ANSWER SECTION:
google.com. 300 IN A 142.250.190.14
```

This returns the IPv4 address of `google.com`.

---

2. Query an A Record

```
dig google.com A
```

Output:

```
google.com. 300 IN A 142.250.190.14
```

Shows the IPv4 address.

---

3. Query an IPv6 Address

```
dig google.com AAAA
```

Output:

```
google.com. 300 IN AAAA 2404:6800:4009:80d::200e
```

Shows the IPv6 address.

---

4. Find Mail Servers

```
dig gmail.com MX
```

Example:

```
gmail.com. 300 IN MX 10 smtp.google.com.
```

Lists mail servers responsible for receiving email.

5. Find Name Servers

```
dig google.com NS
```

Output:

```
google.com. IN NS ns1.google.com.
google.com. IN NS ns2.google.com.
```

Shows the authoritative name servers.

---

6. Query TXT Records

```
dig google.com TXT
```

Displays TXT records, which may include:

- SPF
- Domain verification
- Other text information

---

7. Reverse DNS Lookup

```
dig -x 8.8.8.8
```

Output:

```
8.8.8.8.in-addr.arpa. PTR dns.google.
```

Converts an IP address to its hostname.

---

8. Use a Specific DNS Server

```
dig @8.8.8.8 google.com
```

Queries Google's public DNS server instead of the system's default DNS server.

9. Short Output

```
dig +short google.com
```

Output:

```
142.250.190.14
```

Displays only the answer without additional details.

---

10. Trace DNS Resolution

```
dig +trace google.com
```

Shows how the DNS query is resolved step-by-step, from the root servers to the authoritative name server.

---

11. Query All Available Records

```
dig google.com ANY
```

Note: Many DNS servers restrict `ANY` queries, so the results may be incomplete or empty.

Understanding the Output

Example:

```
; <<>> DiG 9.18 <<>> google.com

;; QUESTION SECTION:
;google.com. IN A

;; ANSWER SECTION:
google.com. 300 IN A 142.250.190.14

;; AUTHORITY SECTION:
...

;; ADDITIONAL SECTION:
...

;; Query time: 20 msec
;; SERVER: 192.168.1.1#53
;; WHEN: Thu Jul 27 10:15:10 IST 2026
```

Meaning of the sections:

Section	Description
QUESTION	The DNS query that was sent
ANSWER	The requested DNS record(s)
AUTHORITY	Information about the authoritative DNS servers
ADDITIONAL	Extra records that help resolve the query
Query time	Time taken to receive the response
SERVER	DNS server that answered the query
WHEN	Time the query was performed

Useful Options

Option	Description
`+short`	Display only the answer
`+trace`	Trace DNS delegation from the root servers
`+noall +answer`	Show only the answer section
`-x`	Perform a reverse DNS lookup
`@server`	Query a specific DNS server
`+tcp`	Use TCP instead of UDP
`+stats`	Show query statistics

Practical Uses

- Troubleshoot DNS resolution issues.
- Verify that DNS records have propagated after updates.
- Check mail server (MX) records.
- Validate SPF, DKIM, and domain verification TXT records.
- Test responses from different DNS servers.
- Perform reverse DNS lookups.
- Diagnose DNS configuration problems.

Install `dig`

Ubuntu / Debian

```
sudo apt install dnsutils
```

RHEL / CentOS / Rocky / AlmaLinux

```
sudo dnf install bind-utils
```

or

```
sudo yum install bind-utils
```

Verify Installation

```
dig -v
```

Summary

Feature	Description
Command	`dig`
Full Form	Domain Information Groper
Purpose	Query DNS servers and retrieve DNS records
Common Uses	DNS troubleshooting, domain lookups, reverse lookups, mail server checks
Common Record Types	A, AAAA, MX, NS, TXT, CNAME, SOA, PTR
Most Useful Option	+short for concise output

7. nslookup

The **`nslookup`** command is a network administration tool used to query the **Domain Name System (DNS)**. It helps you find information about domain names, IP addresses, mail servers, name servers, and other DNS records.

It is commonly used for:

- Troubleshooting DNS issues
- Verifying DNS records
- Finding the IP address of a website
- Performing reverse DNS lookups

Syntax

```
nslookup [options] [domain-name | IP-address] [DNS-server]
```

Basic Examples

1. Find the IP address of a domain

```
nslookup google.com
```

Example output:

```
Server:         192.168.1.1
Address:        192.168.1.1#53

Non-authoritative answer:
Name:   google.com
Address: 142.250.183.110
```

This command asks your configured DNS server to resolve `google.com` into an IP address.

2. Reverse DNS Lookup

Find the domain name associated with an IP address.

```
nslookup 8.8.8.8
```

Output:

```
8.8.8.8.in-addr.arpa
name = dns.google.
```

---

3. Query a Specific DNS Server

Instead of using your system's default DNS server:

```
nslookup google.com 8.8.8.8
```

Here,

- Domain: `google.com`
- DNS Server: `8.8.8.8` (Google DNS)

---

4. Find Mail Servers (MX Records)

```
nslookup -type=mx gmail.com
```

Output:

```
gmail.com mail exchanger = 5 gmail-smtp-in.l.google.com
```

This shows the mail servers responsible for receiving email for the domain.

---

5. Find Name Servers (NS Records)

```
nslookup -type=ns ubuntu.com
```

Output:

```
ubuntu.com
nameserver = ns1.canonical.com
nameserver = ns2.canonical.com
```

6. Find IPv6 Address (AAAA Record)

```
nslookup -type=aaaa google.com
```

Example:

```
google.com
Address: 2404:6800:4007:80d::200e
```

---

7. Find Canonical Name (CNAME)

```
nslookup -type=cname www.microsoft.com
```

---

8. Find TXT Records

```
nslookup -type=txt google.com
```

TXT records are often used for:

- SPF
- DKIM
- Domain verification
- Other metadata

---

9. Find SOA Record

```
nslookup -type=soa example.com
```

Displays:

- Primary DNS server
- Administrator email
- Serial number
- Refresh interval
- Retry interval
- Expiry time

Interactive Mode

Start interactive mode:

```
nslookup
```

You'll see:

```
>
```

Now enter commands:

```
> google.com
> set type=mx
> gmail.com
> set type=ns
> ubuntu.com
> exit
```

Useful Interactive Commands

Command	Description
`set type=mx`	Query MX records
`set type=ns`	Query NS records
`set type=txt`	Query TXT records
`set type=soa`	Query SOA records
`server 8.8.8.8`	Change DNS server
`exit`	Exit interactive mode

Common Record Types

Record	Purpose
A	IPv4 address
AAAA	IPv6 address
MX	Mail server
NS	Name server
TXT	Text information
CNAME	Alias
PTR	Reverse lookup
SOA	Start of Authority

Useful Options

Option	Description
`-type=a`	IPv4 address lookup
`-type=aaaa`	IPv6 address lookup
`-type=mx`	Mail servers
`-type=ns`	Name servers
`-type=txt`	TXT records
`-type=soa`	SOA record
`-debug`	Display detailed query information
`-timeout=<seconds>`	Set query timeout
`-retry=<count>`	Number of retries

Real-World Use Cases

Check whether a website's DNS is working

```
nslookup example.com
```

Verify email server configuration

```
nslookup -type=mx company.com
```

Verify domain ownership records

```
nslookup -type=txt company.com
```

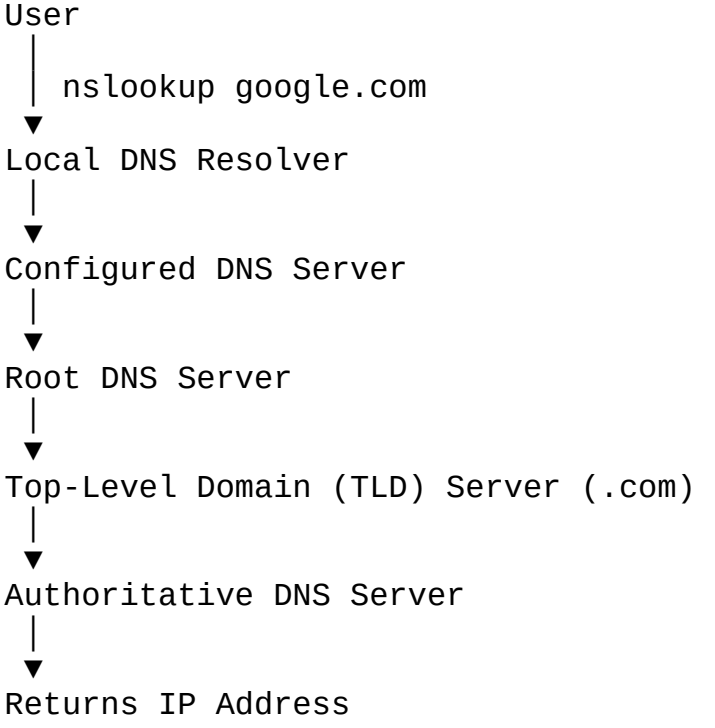
Troubleshoot using a public DNS server

```
nslookup github.com 1.1.1.1
```

Reverse lookup an IP address

```
nslookup 192.168.1.100
```

How `nslookup` Works



Exit Status

`nslookup` prints results directly to the terminal. If a lookup fails, you may see messages such as:

```
** server can't find example.invalid: NXDOMAIN
```

or

```
connection timed out; no servers could be reached
```

These indicate either that the domain does not exist or that the DNS server could not be contacted.

Notes

- **`nslookup`** is available on Linux, macOS, and Windows.
- On many modern Linux distributions, **`nslookup`** is provided by the **`dnsutils`** package (Debian/Ubuntu) or **`bind-utils`** (RHEL/CentOS/Fedora).
- Although still widely used, tools such as **`dig`** and **`host`** are generally recommended for more detailed DNS diagnostics because they provide more comprehensive output.

Summary

Feature	Description
Command	`nslookup`
Purpose	Query DNS records
Used For	DNS troubleshooting and verification
Interactive Mode	Yes
Reverse Lookup	Yes
Common Records	A, AAAA, MX, NS, TXT, CNAME, PTR, SOA
Specify DNS Server	Yes
Typical Users	Network administrators, system administrators, developers

8. resolvectl

1. What is resolvectl ?

`resolvectl` is the command-line tool used to **query and inspect the DNS resolver managed by `systemd-resolved`**.

On Ubuntu 22.04, it is particularly useful for understanding:

- Which DNS servers your system is using
- Which network interface is handling DNS
- Whether DNS resolution is working
- DNS cache statistics
- DNS search domains
- Per-interface DNS configuration
- DNSSEC status
- DNS resolution problems

For example:

```
resolvectl status
```

shows the DNS configuration of your system.

---

2. Why learn `resolvectl`?

DNS problems can be confusing because several components are involved:

```
Application
    ↓
glibc / NSS
    ↓
systemd-resolved
    ↓
DNS server
    ↓
Internet
    ↓
Authoritative DNS server
```

`resolvectl` helps you investigate the part between your Linux system and its configured DNS servers.

It is especially useful when:

Some websites resolve while others don't.

---

3. Check whether `resolvectl` is installed

```
which resolvectl
```

Check:

```
resolvectl --version
```

On Ubuntu 22.04, it is normally installed as part of the `systemd` packages.

---

4. Check DNS configuration

The most important command is:

```
resolvectl status
```

Example:

```
Global
       Protocols: -LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
resolv.conf mode: stub

Link 2 (eno1)
     Current Scopes: DNS
     Protocols: +DefaultRoute +LLMNR -mDNS -DNSOverTLS DNSSEC=no/unsupported
     DNS Servers: 203.153.41.28 203.153.44.44
     DNS Domain: domain.name
```

This tells you:

- DNS is being handled by `systemd-resolved`
- `eno1` is currently used for DNS
- DNS servers are:

  ```
  203.153.41.28
  203.153.44.44
  ```

- DNSSEC is disabled/unsupported
- LLMNR is enabled on that interface

5. Understand `resolv.conf mode: stub`

If you see:

```
resolv.conf mode: stub
```

Ubuntu is normally using the local resolver address:

```
127.0.0.53
```

Check:

```
cat /etc/resolv.conf
```

You may see:

```
nameserver 127.0.0.53
options edns0
search domain.name
```

The important concept is:

```
Application
  ↓
127.0.0.53:53
  ↓
systemd-resolved
  ↓
203.153.41.28 / 203.153.44.44
  ↓
DNS
```

---

6. Query a domain

One of the most useful commands is:

```
resolvectl query google.com
```

Example:

```
google.com: 142.251.x.x
             -- link: eno1
```

Try:

```
resolvectl query web.arattai.in
```

This directly tests whether the system's resolver can resolve the hostname.

7. Compare `resolvectl` with `dig`

You can use both:

```
resolvectl query web.arattai.in
```

and:

```
dig web.arattai.in
```

The difference is important.

`resolvectl`

Tests the **systemd-resolved path**.

`dig`

Can query a specific DNS server directly.

For example:

```
dig web.arattai.in @203.153.41.28
```

So you can determine where the problem occurs.

---

8. Query a specific DNS server

`resolvectl` can also query a specific DNS server:

```
resolvectl query google.com --legend=no
```

However, for directly comparing arbitrary DNS servers, `dig` is generally more convenient:

```
dig google.com @8.8.8.8
```

and:

```
dig google.com @1.1.1.1
```

Therefore, a good Linux student should learn **both** `resolvectl` **and** `dig`.

9. Display DNS statistics

```
resolvectl statistics
```

Example information includes:

```
Transactions
Current Transactions: ...
Total Transactions: ...
Cache
Current Cache Size: ...
Cache Hits: ...
Cache Misses: ...
```

This helps you understand how the local DNS cache is behaving.

---

10. Flush the DNS cache

If you suspect stale DNS information:

```
sudo resolvectl flush-caches
```

Then test:

```
resolvectl query example.com
```

You can check statistics again:

```
resolvectl statistics
```

A useful troubleshooting sequence is:

```
sudo resolvectl flush-caches
resolvectl query example.com
```

---

11. Check DNS cache statistics

```
resolvectl statistics
```

Look particularly at:

```
Cache Hits
Cache Misses
```

A cache hit means the resolver was able to answer from its local cache.

12. Check a particular network interface

Your machine has interfaces such as:

```
eno1
wlxf0090d40fad3
virbr0
```

You can inspect DNS configuration for an interface:

```
resolvectl status eno1
```

Depending on the systemd version, you can also use:

```
resolvectl dns eno1
```

For example:

```
203.153.41.28
203.153.44.44
```

---

13. Display DNS servers configured for an interface

```
resolvectl dns
```

Example:

```
Link 2 (eno1): 203.153.41.28 203.153.44.44
```

This is useful when a machine has several interfaces.

---

14. Display DNS domains

```
resolvectl domain
```

You might see:

```
Link 2 (eno1): domain.name
```

DNS domains can influence which interface is used for particular DNS queries.

15. Configure DNS temporarily

You can configure DNS for a particular interface:

```
sudo resolvectl dns eno1 8.8.8.8 1.1.1.1
```

Then check:

```
resolvectl status
```

You should see:

```
DNS Servers: 8.8.8.8 1.1.1.1
```

This is useful for **testing** whether your existing DNS servers are causing the problem.

Important

This configuration may not survive a network restart or reboot because **NetworkManager/netplan** may reapply its own DNS configuration.

For permanent configuration, configure DNS through your network-management system rather than relying on **`resolvectl`** `dns`.

---

16. Restore automatic DNS

If you manually configured DNS:

```
sudo resolvectl revert eno1
```

This tells `systemd-resolved` to revert the per-interface DNS settings.

Then:

```
resolvectl status
```

---

17. Test DNSSEC

Check:

```
resolvectl status
```

You may see:

```
DNSSEC=no/unsupported
```

To investigate DNSSEC with `dig`:

```
dig example.com +dnssec
```

`resolvectl` can also show DNSSEC-related resolver status.

DNSSEC problems can sometimes cause:

```
SERVFAIL
```

for particular domains.

---

18. Check DNS-over-TLS

Your output showed:

```
DNSOverTLS
```

If configured, this can provide encrypted DNS communication.

Check:

```
resolvectl status
```

You may see:

```
DNSOverTLS=no
```

or:

```
DNSOverTLS=yes
```

You can configure it for an interface, for example:

```
sudo resolvectl dnsovertls eno1 yes
```

But don't enable this merely for troubleshooting unless you specifically want to test DNS-over-TLS.

---

19. Check LLMNR

Your output showed:

```
+LLMNR
```

LLMNR stands for:

Link-Local Multicast Name Resolution

Check the status with:

```
resolvectl status
```

You may see:

```
LLMNR=yes
```

LLMNR is mainly useful for local-network hostname resolution and is separate from normal Internet DNS.

---

20. Check mDNS

You may see:

```
-mDNS
```

mDNS means:

Multicast DNS

It is commonly associated with `.local` hostnames and services such as Bonjour/Avahi.

For example:

```
resolvectl query printer.local
```

If mDNS is disabled, `.local` resolution may not work through this mechanism.

---

21. Monitor DNS troubleshooting

For a domain that is failing:

```
resolvectl query failing-site.com
```

Then compare:

```
dig failing-site.com
```

and:

```
dig failing-site.com @8.8.8.8
```

This gives you a useful diagnostic matrix:

Test	Purpose
`resolvectl query`	Test systemd-resolved
`dig domain`	Test normal DNS resolver path
`dig @DNS-server`	Test a specific DNS server
`dig @8.8.8.8`	Compare against Google DNS
`dig @1.1.1.1`	Compare against Cloudflare DNS

---

22. The current DNS troubleshooting example

We currently have:

```
eno1
↓
203.153.41.28
203.153.44.44
```

We already established:

```
dig web.arattai.in @203.153.41.28
```

returns:

```
NOERROR
```

and:

```
74.188.224.19
```

Now test through `systemd-resolved`:

```
resolvectl query web.arattai.in
```

If it returns:

```
web.arattai.in: 74.188.224.19
```

then:

```
Application
    ↓
systemd-resolved
    ↓
203.153.41.28
    ↓
web.arattai.in
    ↓
74.188.224.19
```

is working.

At that point, the problem is **probably beyond DNS**.

then move to:

```
nc -vz -w 5 74.188.224.19 443
```

followed by:

```
curl -4 -v https://web.arattai.in
```

and, if necessary:

```
sudo tcpdump -i eno1 -nn host 74.188.224.19 and tcp port 443
```

This creates a very useful troubleshooting chain:

```
resolvectl
   ↓
DNS resolution
   ↓
nc
   ↓
TCP connectivity
   ↓
tcpdump
   ↓
Packet-level analysis
   ↓
curl
   ↓
TLS / HTTP
```

23. `resolvectl` (vs) `dig`

This distinction is important for Linux networking students.

Command	Main purpose
`resolvectl status`	View system DNS configuration
`resolvectl query`	Query through systemd-resolved
`resolvectl statistics`	Resolver/cache statistics
`resolvectl flush-caches`	Clear DNS cache
`resolvectl dns`	View/configure DNS servers
`resolvectl domain`	View DNS domains
`resolvectl revert`	Revert interface DNS configuration
`dig`	Detailed DNS queries
`host`	Simple DNS lookup
`nslookup`	Interactive DNS lookup

A good rule is:

Use `resolvectl` to understand Linux resolver; use `dig` to investigate DNS itself.

---

24. Most useful commands to remember

For everyday troubleshooting, these are the commands should be learnt first:

View DNS configuration

```
resolvectl status
```

Resolve a hostname

```
resolvectl query google.com
```

View DNS servers

```
resolvectl dns
```

View DNS statistics

```
resolvectl statistics
```

Flush DNS cache

```
sudo resolvectl flush-caches
```

Test a particular interface

```
resolvectl status eno1
```

Temporarily change DNS

```
sudo resolvectl dns eno1 8.8.8.8 1.1.1.1
```

Restore interface DNS

```
sudo resolvectl revert eno1
```

---

25. A practical student exercise

Use a example website and determine **where DNS resolution fails**.

Exercise

```
resolvectl status
```

Then:

```
resolvectl query example.com
```

Then:

```
dig example.com
```

Then:

```
dig example.com @8.8.8.8
```

Then:

```
dig example.com @1.1.1.1
```

Finally:

```
sudo resolvectl flush-caches
resolvectl query example.com
```

Answer the following:

1. Which DNS server is being used?
2. Does `systemd-resolved` resolve the domain?
3. Does the ISP DNS resolver resolve it?
4. Does Google DNS resolve it?
5. Does Cloudflare DNS resolve it?
6. Is the problem local, with the DNS resolver, or with the domain itself?

That makes **`resolvectl`** much more than a command to memorize—it teaches students **how Linux DNS resolution actually works**.

The key lesson

`resolvectl` is the first tool to reach for when you want to understand what DNS configuration your Linux machine is actually using.

And when combined with **`dig`** + **`nc`** + **`tcpdump`** + **`curl`**, it gives students a very powerful methodology for diagnosing real-world network problems.