

Enabling IPv6 in Linux

Ibrahim Haddad

Most of today's Internet uses IPv4, or Internet Protocol version 4, which has proven to be robust, easily implemented and interoperable, and has stood the test of scaling to the size of today's Internet by using different mechanisms such as NAT. However, the initial design of IPv4 did not take into consideration several issues that are of importance today and suffers from problems in various areas: address space limitations, inefficient routing, lack of support for security, lack of autoconfiguration, lack of QoS support, and poor mobility support.

The Internet Engineering Task Force (IETF) had two options. The first option was to fix IPv4 and risk the continued degradation of the Internet model, which would lead to more complex and volatile network services, lower performance, and less robust, less secure, and less manageable networks. The second option was to replace IPv4 with a newer version and enable simple and stable network services, higher performance, and more robust, secure, and manageable networks. They wanted to avoid doing changes now to IPv4 and then repeat the same exercise a few years later on. Therefore, they made the decision to design a new protocol. As a result, the IETF defined IPv6 to fix the problems in IPv4 and to add many improvements for future networks.

The design philosophy of IPv6 is a scalable protocol that provides a large address space with a simple structure, an original end-to-end environment, a NAT-free network, fast processing, and many features needed by current and future applications. Migrating from IPv4 to IPv6, and IPv6 deployment should not be expensive. IPv6 should inter-operate with IPv4 and provide tools and mechanisms needed by hosts running different IP versions to communicate with each other, and to enable applications to work with both IP versions.

IPv6 Features

IPv6 was designed with enhanced features compared to IPv4. Its major features are:

1. Large Address Space

IPv6 provides a 128-bit address field. This extended address space is very essential, as IP addresses will be assigned to mobile phones, home appliances, motor vehicles, and other equipment. In addition, with such a huge address space, we can create multi-level hierarchies of addresses, simplifying routing problems — requiring simpler routing algorithms and less space needed for routing tables.

2. New Types of Addresses

IPv6 introduces the concept of scoped addresses and defines three types of addresses: unicast (global, link local, site local), multicast, and anycast.

An IPv6 unicast address identifies a single interface. A packet sent to a unicast address is delivered to the interface identified by that address. Three types of unicast addresses exist:

1. A global unicast address is used for point-to-point communication.
2. A link local unicast address allows packets to traverse on only one link or segment. Routers will not forward packets with link local unicast addresses.
3. A site local unicast address limits the scope of packet delivery to your intranet. The edge router connecting your internal network to the external network will not forward packets with site local unicast addresses to the external network.

An IPv6 multicast address delivers copies of one source packet to recipients. In the IPv6 multicast address, you can specify multicast scope, which can be node-local, link-local, site-local, or global.

Enabling IPv6 in Linux

Ibrahim Haddad

An IPv6 anycast address identifies a set of interfaces typically belonging to different nodes. A packet sent to an anycast address is delivered to one of the interfaces identified by that address. Anycast differs from multicast in that it delivers a message to any one of the nodes in a group. When one node — often the nearest node in the group — receives the message, anycast is finished.

Autoconfiguration

IPv6 provides hosts with the ability to configure themselves automatically without the use of a stateful configuration protocol. A host can also use router discovery to determine the addresses of routers, additional addresses, and other configuration parameters. This feature allows hosts to discover automatically all the information they need to connect to the Internet, without any human intervention.

New Streamlined Header Format

IPv6 has a new 40-bytes header (as shown in Figure 1) with the following fields:

- Version, 4 bits that identify the version of the Internet Protocol.
- Traffic class, 8 bits that identify different classes or priorities.
- Flow label, 20 bits used by a source node to identify packets that belong to the same flow.
- Payload length, 16 bits containing the length of the IPv6 payload.
- Next header, 8 bits that indicate to the router which extension header to expect next. If there are no more extension headers, the next header field indicates the upper layer header.
- Hop limit, indicating the maximum number of hops allowed.
- Source address, 128 bits containing the address of the source node sending the packet.
- Destination address, 128 bits containing the final destination node address for the packet.

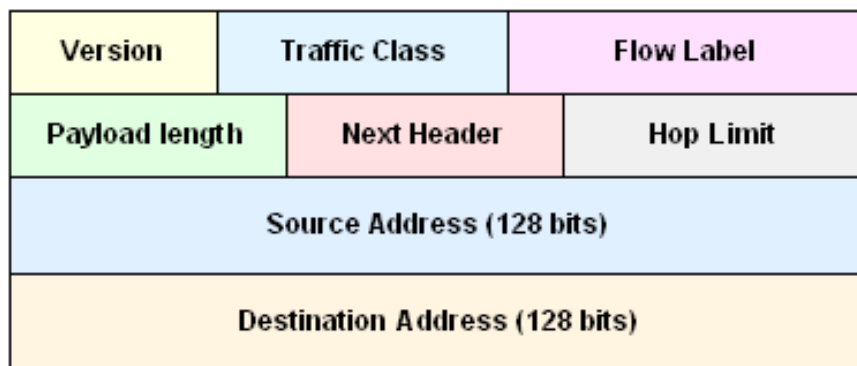


Figure 1
The IPv6 Header

In addition, IPv6 is much more flexible in its support of options through extension headers. Extension headers encode optional Internet-layer information. They are placed between the IPv6 header and the upper layer header in a packet and are chained together using the next header field in the IPv6 header. There are six different extension headers: Hop-by-hop Options, Destination Options, Routing,

Enabling IPv6 in Linux

Ibrahim Haddad

Fragment, Authentication, and Encapsulated Security. The next header field indicates to the router which extension header to expect next. If there are no more extension headers, the next header field indicates the upper-layer header (TCP header, UDP header, ICMPv6 header, an encapsulated IP packet, or other items).

Better Network Management

IPv6 provides enhancements that allow better network management such as network renumbering, which make it simpler to move a whole network to a new ISP by reconfiguring the router with the new routing prefix from the new ISP.

Improved Mobility Support

Mobility support in IPv6 allows transparent routing of IPv6 packets to mobile nodes, taking advantage of the design of a new version of IP.

Support for IPsec

The IETF has mandated support for Internet Protocol Security (Ipsec) with IPv6 so it will not be an optional extension, as was the case with IPv4.

QoS

The IETF has specified two approaches, integrated services and differentiated services, to provide guaranteed and selectable Quality of Service (QoS) over the Internet. In addition, IPv6 provides flow labels that can be used to provide QoS by identifying the packets as belonging to a flow. These labels can be used in conjunction with a hop-by-hop routing extension header (allowing predefined routes) and the priority field (allowing for QoS). The flow label also serves as a key in the router cache to reduce the amount of processing. When a router first receives a datagram, it can cache the flow label and next hop so as to save time when the next datagram arrives with the same flow label. This technique reduces router processing time considerably.

As a result, IPv6 is feature-rich, fixing many of the problems of IPv4 and adding much new functionality. To read more on these features, please refer to my IPv6 Essentials article.

Supporting IPv6 in the Linux Kernel

IPv6 support can be enabled as a built-in kernel feature or as a loadable module. First, we will demonstrate how to support IPv6 as a built-in feature of the kernel, and then as a kernel module that can be loaded and unloaded as needed. For the purpose of this article, we will be using a Linux test machine running Red Hat 9.0 and the official kernel version 2.6.0-test9, which was available at the time of writing this article (November 2003). However, you can follow the same steps for virtually any kernel version.

Download and Unpack

You need to download the latest kernel version and uncompress it; alternatively, you can use whatever kernel version is already available on your system. I would assume, though, that you'd want to try the latest and greatest version. Download the compressed source and uncompress it under /usr/src:

```
% cd /usr/src
% bunzip2 linux-2.6.0-test9.tar.bz2
% tar -xvf linux-2.6.0-test9.tar
```

A directory called linux-2.6.0-test9 will be created.

Enabling IPv6 in Linux

Ibrahim Haddad

Pre-Configuration Steps

Before you can configure the kernel, you need to perform a few steps:

Delete the symbolic link to the older kernel source tree:

```
% rm linux
```

Create a symbolic link to the new kernel source tree:

```
% ln -s linux-2.6.0-test9 linux
```

Move into the kernel source directory:

```
% cd linux
```

Make sure to clean up any existing .o files and old dependencies:

```
% make mrproper
```

Configuring the Kernel

You can use make config, make menuconfig, or make xconfig to configure the kernel. We'll use the latter here. There are three areas where you need to update the kernel configuration; we will explain all of them.

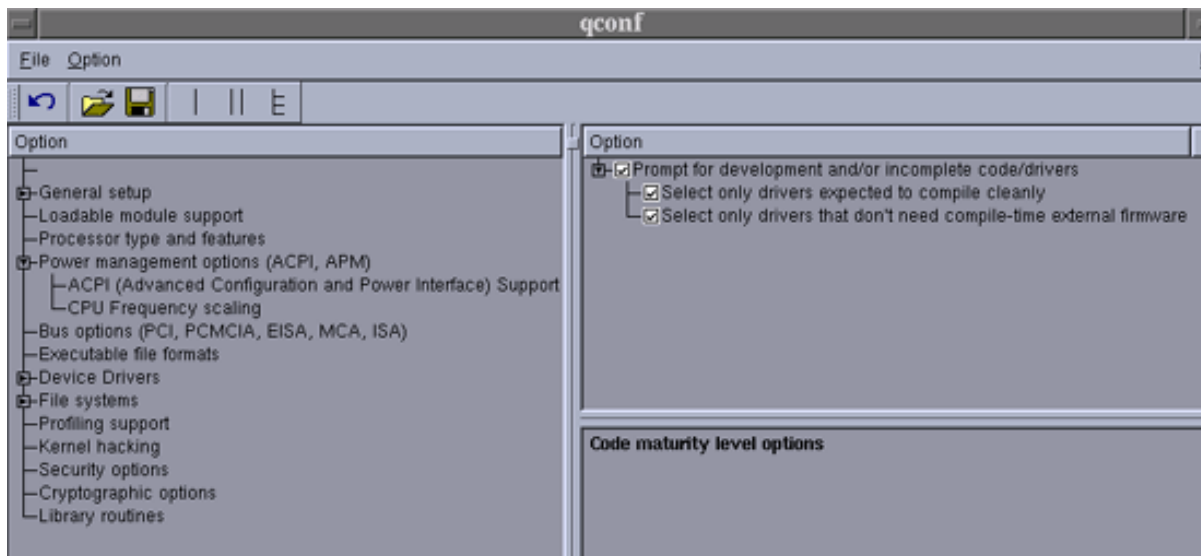


Figure 2
Code Maturity Level Options

Since IPv6 is still an experimental feature, you need to enable the "Prompt for development and/or incomplete code/driver" option to be able to have the option to activate IPv6 support in a later configuration option (see Figure 2).

Enabling IPv6 in Linux

Ibrahim Haddad

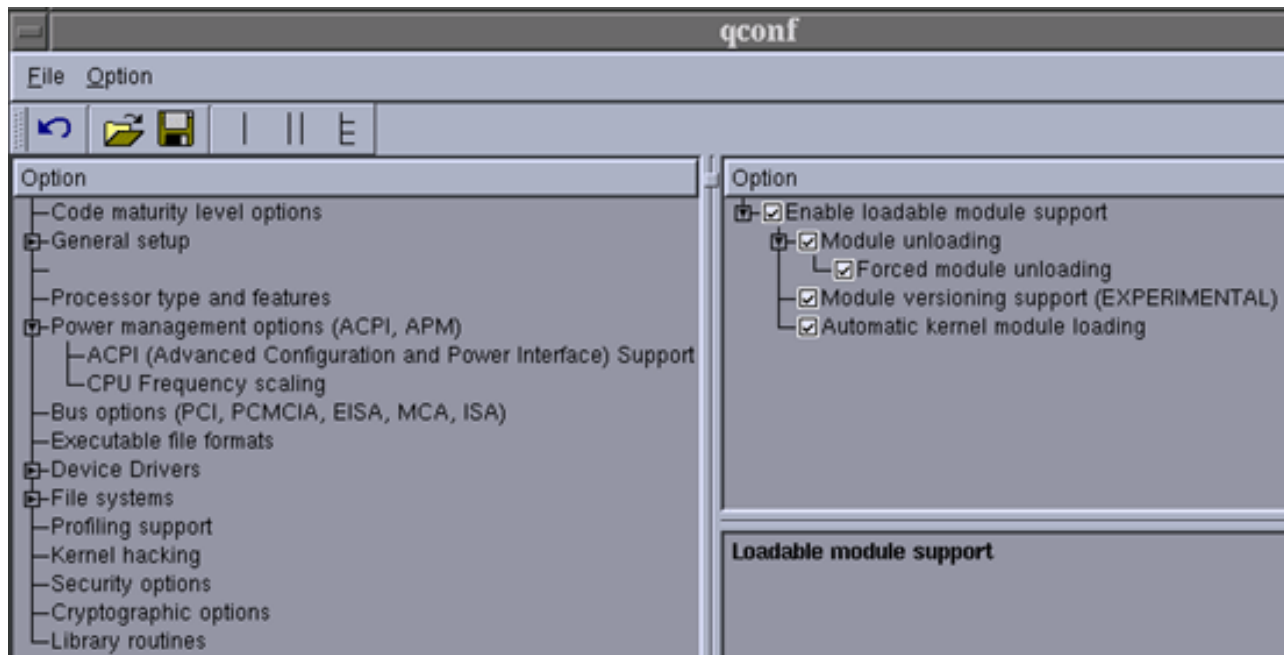


Figure 3
Loadable Module Support

In most systems, the option "Enable loadable module support" is enabled by default, as shown in Figure 3. You can also enable the following options:

Module Unloading and Forced Module Unloading. Enable these options if you want to be able to unload a module or force a module to unload, even if the kernel believes it is unsafe.

Module Versioning Support (EXPERIMENTAL). If you enable this option, it will be possible sometimes to load modules compiled against a different kernel version. This is not required in our case since you are compiling the IPv6 module with the same kernel you are running. However, you can enable it.

Automatic Kernel Module Loading. If you enable this option, then when needed, the kernel will be able to load modules for itself: when a part of the kernel needs a module, it will run modprobe with the appropriate arguments, thereby loading the module if it is available.

To enable support for IPv6, you need to enable the "IPv6 protocol (EXPERIMENTAL)" either as a built-in kernel feature or as a module, as shown in Figure 4.

Enabling IPv6 in Linux

Ibrahim Haddad

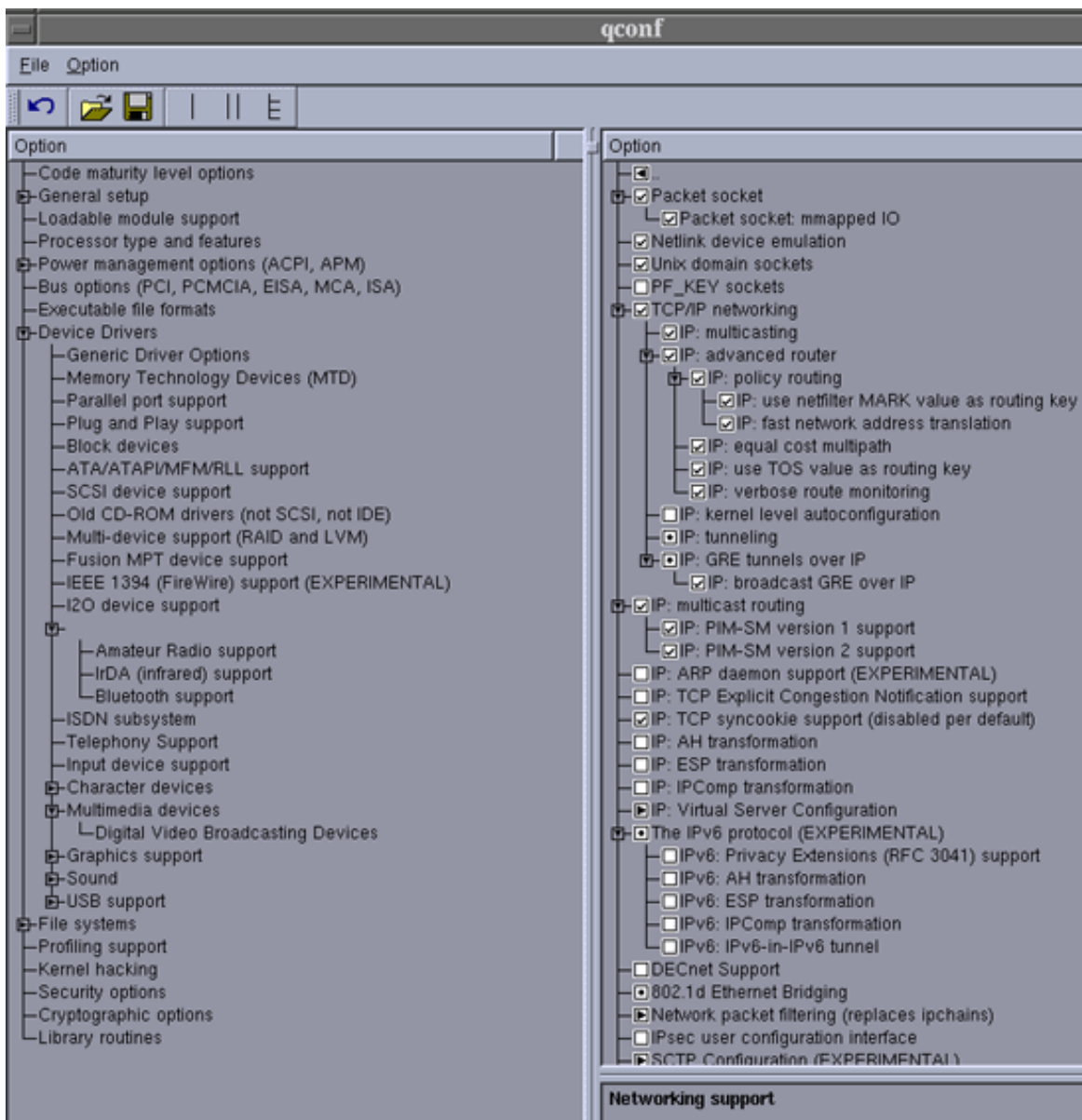


Figure 4
Networking Support Options

Compiling and Installing the New Kernel

After enabling these options, save the configuration and exit the kernel configuration tool. Now you are ready to compile the kernel.

```
% make bzImage
```

The result of the compilation will be a new compressed kernel image created in /usr/src/linux/arch/i386/boot/ (or the appropriate architecture-specific directory for your computer). If you compiled some features as loadable modules, you should compile and install the modules:

```
% make modules
```

Enabling IPv6 in Linux

Ibrahim Haddad

```
% make modules_install
```

You can now copy the new kernel image bzImage and System.map to your boot directory:

```
% cp arch/i386/boot/bzImage /boot/vmlinuz-2.6.0-test9-ipv6
% cp /usr/src/linux/System.map /boot/System.map-2.6.0-test9-ipv6
% rm /boot/System.map
% ln -fs /boot/System.map-2.6.0-test9-ipv6 /boot/System.map
```

If you configured IPv6 support as a module, a module will be created in /lib/modules/linux-2.6.0-test9/kernel/net/ipv6. The module is called ipv6.o.

Updating your Bootloader Configuration

If you use LILO, you need to add a new entry in the LILO configuration file (/etc/lilo.conf) for your freshly-compiled kernel. At boot time, you'll see this entry in the list of kernels to boot. A sample entry will look like this:

```
Image=/boot/vmlinuz-2.6.0-test9-ipv6
label=2.6.0-test9-ipv6
root=/dev/hda1
read-only
```

Make sure that the root directive refers to the correct partition on your system. Next, run /sbin/lilo to install the bootloader with the new configuration options.

After following these steps, you will have a boot-time entry called 2.6.0-test9-ipv6. We recommend that you do not set this new image as the default at first; it is best to try it and make sure it works before you set it as a default boot image or delete a known-good entry.

On the other hand, if you use grub as your bootloader, add an entry to the /etc/grub.conf configuration file as follows:

```
title 2.6.0-test9-ipv6
root (hd0,0)
kernel /vmlinuz-2.6.0-test9-ipv6 ro root=/dev/hda1
```

Please remember that you need to update this entry to reflect your disk partitioning.

Rebooting with the New Kernel

You are now ready to reboot your Linux machine with the new kernel:

```
% shutdown -r now
```

When the bootloader prompt comes up, choose to boot with 2.6.0-test9-ipv6. After rebooting, if you compiled IPv6 as a module, load that module:

```
% insmod ipv6
```

Testing your Configuration

Now you can verify the network interfaces on your Linux machine by typing ifconfig at the command prompt.

Enabling IPv6 in Linux

Ibrahim Haddad

```
[root@pgtest1-ibha root]# ifconfig
```

```
eth0      Link encap:Ethernet  HWaddr 00:50:8B:F1:6A:D0
          inet addr:192.168.15.70  Bcast: 192.168.15.255
Mask:255.255.255.0
          inet6 addr: fec0::1:250:8bff:fef1:6ad0/64 Scope:Site
          inet6 addr: fe80::250:8bff:fef1:6ad0/10 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:28134 errors:0 dropped:0 overruns:0 frame:0
          TX packets:4351 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:100
          RX bytes:1719007 (1.6 Mb)  TX bytes:352632 (344.3 Kb)
          Interrupt:5 Base address:0x1000

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:394 errors:0 dropped:0 overruns:0 frame:0
          TX packets:394 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:47176 (46.0 Kb)  TX bytes:47176 (46.0 Kb)
```

```
[root@pgtest1-ibha root]#
```

You can also test by pinging your IPv6 loopback address and your link local address.

```
[root@pgtest1-ibha root]# ping6 ::1
```

```
PING ::1(::1) from ::1 : 56 data bytes
64 bytes from ::1: icmp_seq=1 ttl=64 time=0.052 ms
64 bytes from ::1: icmp_seq=2 ttl=64 time=0.046 ms
64 bytes from ::1: icmp_seq=3 ttl=64 time=0.044 ms
64 bytes from ::1: icmp_seq=4 ttl=64 time=0.047 ms
64 bytes from ::1: icmp_seq=5 ttl=64 time=0.046 ms
--- ::1 ping statistics ---
5 packets transmitted, 5 received, 0% loss, time 3997ms
rtt min/avg/max/mdev = 0.044/0.047/0.052/0.002 ms
```

```
[root@pgtest1-ibha root]#
```

```
[root@pgtest1-ibha root]# ping6 fec0::1:250:8bff:fef1:6ad0
```

```
PING fec0::1:250:8bff:fef1:6ad0(fec0::1:250:8bff:fef1:6ad0) from ::1 : 56
data bytes
64 bytes from fec0::1:250:8bff:fef1:6ad0: icmp_seq=1 ttl=64 time=0.043 ms
64 bytes from fec0::1:250:8bff:fef1:6ad0: icmp_seq=2 ttl=64 time=0.057 ms
64 bytes from fec0::1:250:8bff:fef1:6ad0: icmp_seq=3 ttl=64 time=0.047 ms
--- fec0::1:250:8bff:fef1:6ad0 ping statistics ---
3 packets transmitted, 3 received, 0% loss, time 2000ms
rtt min/avg/max/mdev = 0.043/0.049/0.057/0.005 ms
```

Enabling IPv6 in Linux

Ibrahim Haddad

```
[root@pgtest1-ibha root]#
```

If you try to ssh over IPv6, you'll see something like this:

```
[root@pgtest1-ibha root]# ssh -6 fec0::1:250:8bff:fe1:6ad0
```

```
The authenticity of host 'fec0::1:250:8bff:fe1:6ad0
(fec0::1:250:8bff:fe1:6ad0)' can't be established.
RSA key fingerprint is b6:1b:d2:62:03:0c:d4:a0:61:46:50:9b:6e:03:5b:05.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added 'fec0::1:250:8bff:fe1:6ad0' (RSA) to the list
of known hosts.
root@fec0::1:250:8bff:fe1:6ad0's password:
Last login: Thu Nov  6 20:09:08 2003 from 142.133.100.75
```

```
[root@pgtest1-ibha root]#
```

Since you do not yet have a global IPv6 address, you are limited to test with your own machine and the machines sitting on the same local link (vlan).

Conclusion

This article introduced IPv6, along with a step-by-step tutorial on supporting IPv6 on your Linux machine. An upcoming article will demonstrate how to connect your Linux machine, now IPv6-enabled, to the IPv6 Internet, which will also allow you to have all kinds of testing and functionalities over IPv6.