Quick Start

You'll first want to make sure you have a decent grasp of the conceptual overview, and then install WireGuard. After that, read onwards here.

Side by Side Video

Before explaining the actual comands in detail, it may be extremely instructive to first watch them being used by two peers being configured side by side:

0:00 / 2:25

Or individually, a single configuration looks like:

Command-line Interface

A new interface can be added via `ip-link(8)`, which should automatically handle module loading:

```
# ip link add dev wg0 type wireguard
```

(Non-Linux users will instead write `wireguard-go wg0`.)

An IP address and peer can be assigned with `ifconfig(8)` or `ip-address(8)`

```
# ip address add dev wg0 192.168.2.1/24
```

Or, if there are only two peers total, something like this might be more desirable:

```
# ip address add dev wg0 192.168.2.1 peer 192.168.2.2
```

The interface can be configured with keys and peer endpoints with the included `wg(8)` utility:

```
# wg setconf wg0 myconfig.conf
```

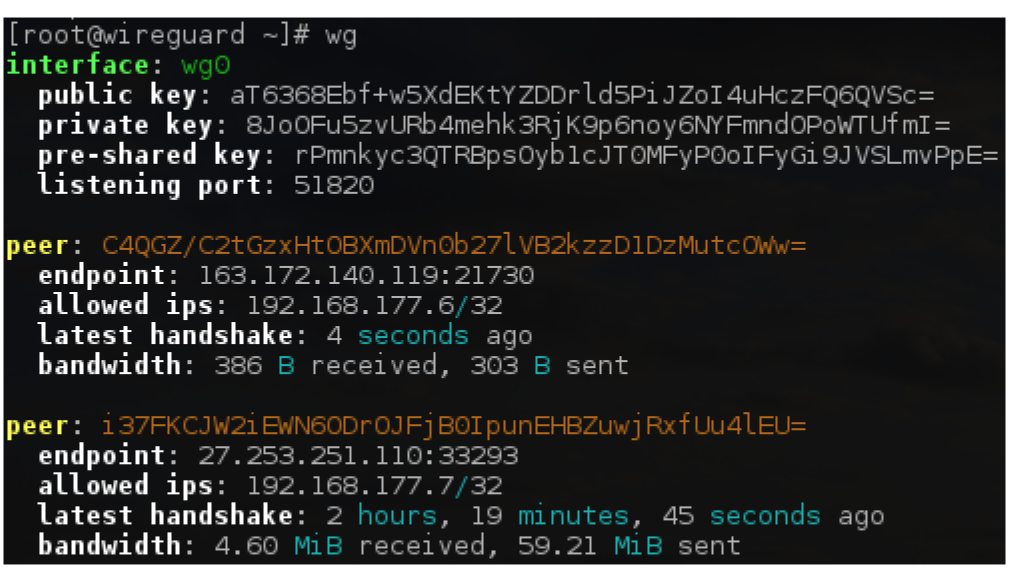
or

```
# wg set wg0 listen-port 51820 private-key /path/to/private-key peer ABCDEF...
allowed-ips 192.168.88.0/24 endpoint 209.202.254.14:8172
```

Finally, the interface can then be activated with `ifconfig(8)` or `ip-link(8)`:

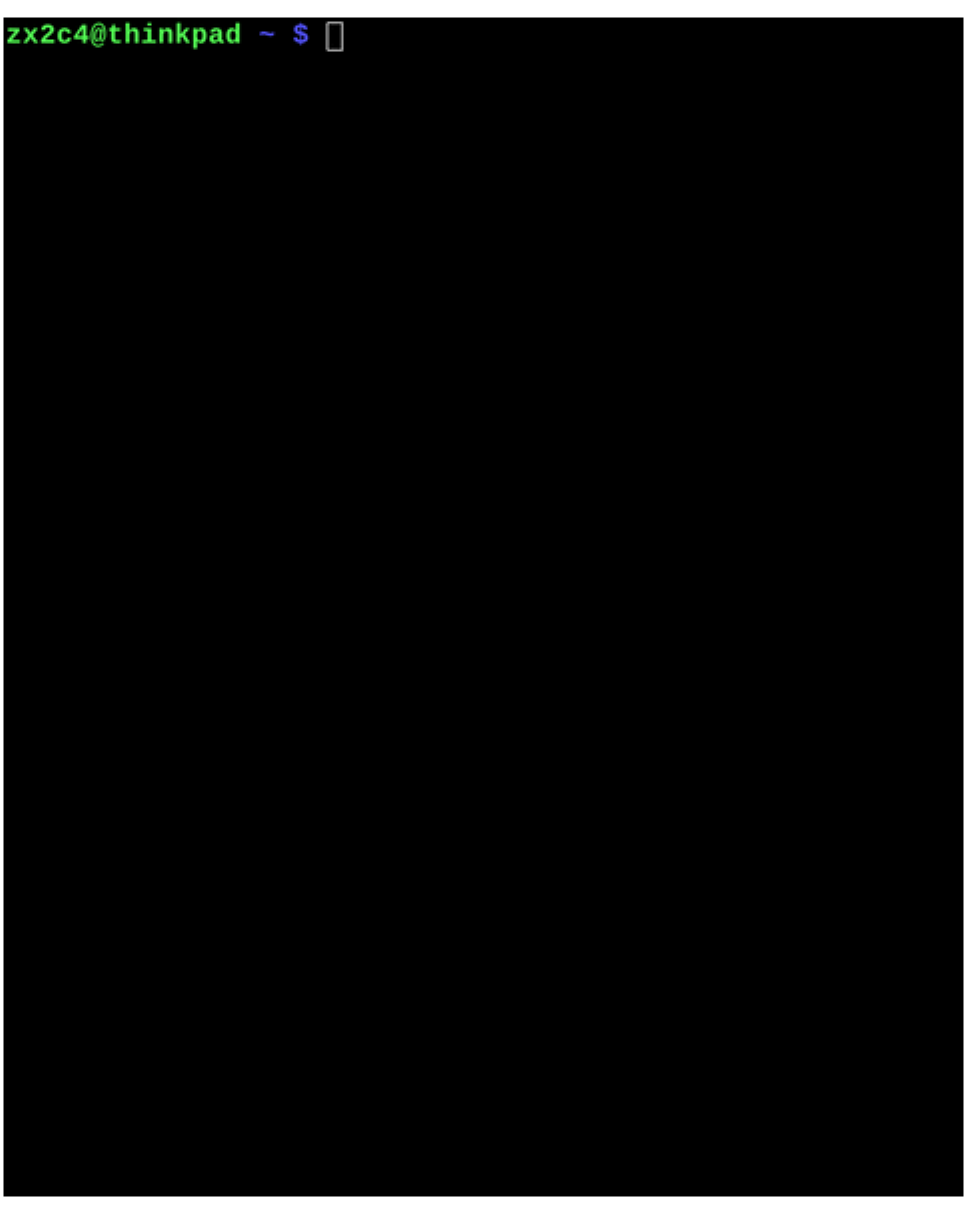
```
# ip link set up dev wg0
```

There are also the `wg show` and `wg showconf` commands, for viewing the current configuration. Calling `wg` with no arguments defaults to calling `wg show` on all WireGuard interfaces.



Consult the man page of `wg(8)` for more information.

Much of the routine bring-up and tear-down dance of `wg(8)` and `ip(8)` can be automated by the included `wg-quick(8)` tool:



Key Generation

WireGuard requires base64-encoded public and private keys. These can be generated using the `wg(8)` utility:

```
$ umask 077
$ wg genkey > privatekey
```

This will create `privatekey` on stdout containing a new private key.

You can then derive your public key from your private key:

```
$ wg pubkey < privatekey > publickey
```

This will read `privatekey` from stdin and write the corresponding public key to `publickey` on stdout.

Of course, you can do this all at once:

```
$ wg genkey | tee privatekey | wg pubkey > publickey
```

NAT and Firewall Traversal Persistence

By default, WireGuard tries to be as silent as possible when not being used; it is not a chatty protocol. For the most part, it only transmits data when a peer wishes to send packets. When it's not being asked to send packets, it stops sending packets until it is asked again. In the majority of configurations, this works well. However, when a peer is behind NAT or a firewall, it might wish to be able to receive incoming packets even when it is not sending any packets. Because NAT and stateful firewalls keep track of "connections", if a peer behind NAT or a firewall wishes to receive incoming packets, he must keep the NAT/firewall mapping valid, by periodically sending keepalive packets. This is called *persistent keepalives*. When this option is enabled, a keepalive packet is sent to the server endpoint once every *interval* seconds. A sensible interval that works with a wide variety of firewalls is 25 seconds. Setting it to 0 turns the feature off, which is the default, since most users will not need this, and it makes WireGuard slightly more chatty. This feature may be specified by adding the `PersistentKeepalive =` field to a peer in the configuration file, or setting `persistent-keepalive` at the command line. If you don't need this feature, don't enable it. But if you're behind NAT or a firewall and you want to receive incoming connections long after network traffic has gone silent, this option will keep the "connection" open in the eyes of NAT.

Demo Server

After installing WireGuard, if you'd like to try sending some packets through WireGuard, you may use, for testing purposes only, the script in `contrib/ncat-client-server/client.sh`.

```
$ sudo contrib/examples/ncat-client-server/client.sh
```

This will automatically setup interface `wg0`, through a very insecure transport that is only suitable for demonstration purposes. You can then try loading the hidden website or sending pings:

```
$ chromium http://192.168.4.1
$ ping 192.168.4.1
```

If you'd like to redirect your internet traffic, you can run it like this:

```
$ sudo contrib/examples/ncat-client-server/client.sh default-route
$ curl zx2c4.com/ip
163.172.161.0
demo.wireguard.com
curl/7.49.1
```

By connecting to this server, you acknowledge that you will not use it for any abusive or illegal purposes and that your traffic may be monitored.

Debug Info

If you're using the Linux kernel module and your kernel supports dynamic debugging, you can get useful runtime output by enabling dynamic debug for the module:

```
# modprobe wireguard && echo module wireguard +p > /sys/kernel/debug/dynamic_de
bug/control
```

If you're using a userspace implementation, set the environment variable `export LOG_LEVEL=verbose`.