Installation and basic setup of
Gigapxy (version 1.0-2.x and higher)

Install the package, get a license

Download the package suitable for your system from the website, then untar it and execute the installation script as root. For instance, on a Debian-derived Linux distro:

```
~/tmp$ tar -xzvf gigapxy_1.0.2-12-master.x86_64.deb.tar.gz
gigapxy.deb
install-deb.sh

~/tmp$ sudo ./install-deb.sh gigapxy.deb
Preparing to unpack gigapxy.deb ...
Unpacking gigapxy (1.0-2.12) ...
Setting up gigapxy (1.0-2.12) ...
…
~/tmp$
```

Your copy is not licensed yet, so if you run **gng**, you'll see the following message:

```
~/tmp$ gng -V
gng (Gigapxy relay engine) 1.0-2.12 (master) regular [Debian 8 Linux
(amd64) - 3.16.0-4-amd64/x86_64] (undefined) 74501f5c/0; built on
2018-12-13
==
= ERROR: License not found, please contact support@gigapxy.com ==
==
License validation failed, application will quit.
```

A license is issued based on a key, generated by Gigapxy. The key ties it up to the server the application would run on. Get the key as:

```
~/tmp$ gng -K
System key:             [213a0aca4c1c6a90ca5ca2d9f6c49e758f65291dcf1d34fe]
(0x204a)
```

NB: **`sudo gng -K`** under *FreeBSD*

Copy the full output (including the value in round brackets) into an email to request a demo license from support@gigapxy.com. Your license will arrive in a tarball containing `gigapxy.lic` - the license file. Copy `gigapxy.lic` to `/etc` directory and run `gng` module to check that the license works.

```
~/tmp$ gng -V
gng (Gigapxy relay engine) 1.0-2.12 (master) regular [Debian 8 Linux
(amd64) - 3.16.0-4-amd64/x86_64] (2019-12-31) 74501f5c/253082; built on
2018-12-13
```

NB: **`sudo -u gigaplus gxng -V`** under *FreeBSD*

You're now licensed till January 31st, 2019 `(2019-12-31)` - but **please** make sure to **renew in advance**.

NB: If license check fails (**err = -5**): make sure UDP `port 123 (NTP)` **is open** for requests in both directions.

Update the default configuration (gigapxy.conf)

Default configuration file is automatically stored as `/etc/gigapxy.conf (/usr/local/etc` under `FreeBSD)` during installation. You might need to update it for the application to run. Use any text editor you like.

Before we proceed to editing, I suggest reading (at least) the Gigapxy core manual:

```
$ man gigapxy
```

Let's edit the config (I suggest you save the original first):

```
$ cp /etc/gigapxy.conf ~/tmp/
$ vim ~/tmp/gigapxy.conf
```

There'll be a few sections to address, related to different modules.

If wonder about the meaning of a parameter with no adjacent comment, read a respective man page or study a commented config file:

```
$ man gws.conf
$ man gng.conf
$ less /usr/share/doc/gigapxy/examples/gigapxy-commented.conf
```

NB: Use `/usr/local/share/doc/`… on FreeBSD.

CONFIG: Request listeners (ws.listener)

Make sure you agree with the default access ports and the **network interfaces** the requests would be available from. Mind that **admin** requests might be sensitive to allow on all interfaces. Use your judgement.

`Example`:

```
   listener: {
        admin = { ifc = "lo"; port = "4047"; default_af = "inet";  };
        user  = { ifc = "eth0"; port = "4046"; default_af = "inet"; };
    };
```

In the above example we allow admin requests only on the local interface (same box) and let user requests come from eth0, ports not changed.

CONFIG: Multicast interface (ws.multicast_ifc)

This is set to "**all**" by default, but should be a distinct network interface instead; otherwise the OS will pick one for you and will assume that all multicast data should originate there.

`Example`:

```
multicast_ifc  = "eth1";
```

Before moving on to the `ng.*(gxng)` section of the config, please read the documentation.

```
$ man gng
$ man gng.conf
```

CONFIG: Buffer subsystem (ng.bufd.*)

Buffer subsystem is responsible for caching HTTP-stream data. Each `gng` can allocate up to **max_unit_count** buffers, each of **max_unit_size** bytes, forming a *common pool*. `bufd` has a lot of parameters to tweak, so a special guide has been created to explain and illustrate them. Please study the guide and read the comments in the default config.

CONFIG: Finalizing

Save `gigapxy.conf` and copy it over to `/etc` or `/usr/local/etc` (on FreeBSD).

```
$ sudo cp gigapxy.conf /etc
```

Launch with a single gng

Before any channels are streamed, we need to make sure the config is valid and the application would start. The installation provides **`gigapxy.sh`** script to automate ***stop/start/status*** operations on the application's modules. The location is /usr/share/gigapxy/scripts/gigapxy.sh unde Linux (/usr/local/ under FreeBSD). Let's use the script to verify the configuration.

```
$ ln -s /usr/share/gigapxy/scripts/gigapxy.sh
$ sudo ./gigapxy.sh start
[GWS] is not running.
Starting GWS ...
Starting GNG #1 ...
Pausing for 1 second(s).
GWS [2550] STARTED
GNG [2560] STARTED
$
```

The configuration works, but you should by no means consider it final. However, we can use the running single-gng setup to verify that streaming also works.

Test a stream

Any HTTP client can now request a stream from your running **`gws`**. Pick a multicast channel and issue a udpxy-style request (for this example, the gws listens on **192.168.2.20:4046** and the address of the multicast channel is **224.0.2.26:5050**):

```
$ sudo wget -O /dev/null http://192.168.2.20:4046/udp/224.0.2.26:5050
```

If you see `wget(1)` receiving the stream, your basic setup is complete. Re-read the docs and start working on your **real-world configuration** (**bufd** section is particularly important).

If something went wrong, proceed to **Troubleshooting**.

Troubleshooting

Things can go wrong at any of the steps we've covered so far. Just like this:

```
$ sudo ./gigapxy.sh start
[GWS] is not running.
Starting GWS ...
Starting GNG #1 ...
Pausing for 1 second(s).
GWS [] FAILED
$
```

Check the logs

The first place to check is the **log** for the failed component. If log does not present a clue or is empty, it would make sense to run the modules (**gws**, **gng**) manually, in **DEBUG** mode.

Enable core dumps

Let's first enable core dumps for all components. In `gigapxy.conf` we make sure that core dumps can be generated for suid-executables and enable core dumps in the console. **NB**: Please, don't forget to uncomment **both** for **gws** and **gng**.

```
enforce_core_dumps = true;
```

```
$ ls -ld /opt/gigapxy
drwxr-xr-x 2 gigapxy gigapxy 4096 Dec 14 21:03 /opt/gigapxy
$ cd /opt/gigapxy/
$ ulimit -c unlimited
```

Run components manually

Let's launch **gws** manually (from `/opt/gigapxy`, under user **gigapxy**):

```
$ sudo -u gigapxy gws -vvv -l gws-manual.log
```

If something goes wrong, the module will exit (or crash/core-dump), if not, it will most likely wait for events to process, indicating it in the log.

If needed, it's as easy to start a **gng** the same way (in a different terminal window):

```
$ sudo -u gigapxy gng -vvv -l gng-manual.log
```

If either of the modules core-dumps, this script (run as a super-user) may help to generate cores with meaningful names:

```
$ cat corep.sh
# @(#) Makes sure cores are dumped in a proper format.

echo '1' > /proc/sys/fs/suid_dumpable
echo "%e.%p.core" > /proc/sys/kernel/core_pattern
$
```

In case of a crash, gather relevant logs and cores and **tar.gz** them along with `gigapxy.conf`. If cores are present, include `/usr/bin/gigapxy` executable in the tarball.

Cannot connect to gws(1)

If you simply cannot connect to the primary port (usually `4046`), corresponding to `gws(1)`, then the first thing to check is whether `gws(1)` receives your requests. See if `gws.log` holds the clue to that. If no requests come through, it might be the firewall (allow `4046/tcp`).

Probe the stream

Another thing to consider, when things don't work out with a channel is: *how is the stream doing*? See if it runs at all, probe it with `ffprobe` to see if its format is ok. Don't probe from your desktop, do it from the server where the issue is, please.

Seek technical support

If all of the above fails, write to support (support@gigapxy.com). There is also a Google+ community where all sorts of information could be found. A Telegram-based support channel is there for interactive help.

Enjoy Gigapxy!