




 [meetecho](#) / [janus-gateway](#)

Janus WebRTC Server

 [janus.conf.meetecho.com](#) GPL-3.0 License 5.8k stars  1.9k forks Star Watch ▾<> **Code** Issues 20 Pull requests 9 Actions Security Insights master ▾

...

**jmfotokite** Use crypto safe random numbers (#2738) ...

✓ 23 days ago ⌚ 5,006

[View code](#)

# Janus WebRTC Server

License GPLv3

 janus-ci passing

coverity passed

oss-fuzz fuzzing

Janus is an open source, general purpose, WebRTC server designed and developed by [Meetecho](#). This version of the server is tailored for Linux systems, although it can be compiled for, and installed on, MacOS machines as well. Windows is not supported, but if that's a requirement, Janus is known to work in the "Windows Subsystem for Linux" on Windows 10: do **NOT** trust repos that provide .exe builds of Janus, they are not official and will not be supported.

For some online demos and documentations, make sure you pay the [project website](#) a visit!

If you have questions on Janus, or wish to discuss Janus with us and other users, please join our [meetecho-janus](#) Google Group. If you encounter bugs, please submit an issue on [GitHub](#): make sure you read the [guidelines](#) before opening an issue, though.

# Dependencies

---

To install it, you'll need to satisfy the following dependencies:

- [Jansson](#)
- [libconfig](#)
- [libnice](#) (at least v0.1.16 suggested, v0.1.18 recommended)
- [OpenSSL](#) (at least v1.0.1e)
- [libsrt](#) (at least v2.x suggested)
- [usrctp](#) (only needed if you are interested in Data Channels)
- [libmicrohttpd](#) (at least v0.9.59; only needed if you are interested in REST support for the Janus API)
- [libwebsockets](#) (only needed if you are interested in WebSockets support for the Janus API)
- [cmake](#) (only needed if you are interested in WebSockets and/or BoringSSL support, as they make use of it)
- [rabbitmq-c](#) (only needed if you are interested in RabbitMQ support for the Janus API or events)
- [paho.mqtt.c](#) (only needed if you are interested in MQTT support for the Janus API or events)
- [nanomsg](#) (only needed if you are interested in Nanomsg support for the Janus API)
- [libcurl](#) (only needed if you are interested in the TURN REST API support)

A couple of plugins depend on a few more libraries:

- [Sofia-SIP](#) (only needed for the SIP plugin)
- [libopus](#) (only needed for the AudioBridge plugin)
- [libogg](#) (needed for the VoiceMail plugin and/or post-processor, and optionally AudioBridge and Streaming plugins)
- [libcurl](#) (only needed if you are interested in RTSP support in the Streaming plugin or in the sample Event Handler plugin)
- [Lua](#) (only needed for the Lua plugin)

Additionally, you'll need the following libraries and tools:

- [GLib](#)
- [zlib](#)
- [pkg-config](#)
- [gengetopt](#)

All of those libraries are usually available on most of the most common distributions. Installing these libraries on a recent Fedora, for instance, is very simple:

```
yum install libmicrohttpd-devel jansson-devel \
    openssl-devel libsrtplib-devel sofia-sip-devel glib2-devel \
    opus-devel libogg-devel libcurl-devel pkgconfig gengetopt \
    libconfig-devel libtool autoconf automake
```

Notice that you may have to `yum install epel-release` as well if you're attempting an installation on a CentOS machine instead.

On Ubuntu or Debian, it would require something like this:

```
aptitude install libmicrohttpd-dev libjansson-dev \
    libssl-dev libsrtplib-dev libsofia-sip-ua-dev libglib2.0-dev \
    libopus-dev libogg-dev libcurl4-openssl-dev liblua5.3-dev \
    libconfig-dev pkg-config gengetopt libtool automake
```

- *Note:* please notice that libopus may not be available out of the box on your distro. In that case, you'll have to [install it manually](#).

While `libnice` is typically available in most distros as a package, the version available out of the box in Ubuntu is known to cause problems. As such, we always recommend manually compiling and installing the master version of libnice. To build libnice, you need Python 3, Meson and Ninja:

```
git clone https://gitlab.freedesktop.org/libnice/libnice
cd libnice
meson --prefix=/usr build && ninja -C build && sudo ninja -C build install
```

- *Note:* Make sure you remove the distro version first, or you'll cause conflicts between the installations. In case you want to keep both for some reason, for custom installations of libnice you can also run `pkg-config --cflags --libs nice` to make sure Janus can find the right installation. If that fails, you may need to set the `PKG_CONFIG_PATH` environment variable prior to compiling Janus, e.g., `export PKG_CONFIG_PATH=/path/to/libnice/lib/pkgconfig`

In case you're interested in compiling the sample Event Handler plugin, you'll need to install the development version of libcurl as well (usually `libcurl-devel` on Fedora/CentOS, `libcurl4-openssl-dev` on Ubuntu/Debian).

If your distro ships a pre-1.5 version of libsrtp, you'll have to uninstall that version and [install 1.5.x, 1.6.x or 2.x manually](#). In fact, 1.4.x is known to cause several issues with WebRTC. While 1.5.x is supported, we recommend installing 2.x instead. Notice that the following steps are for version 2.2.0, but there may be more recent versions available:

## ☰ README.md

```
cd libsrtp-2.2.0
./configure --prefix=/usr --enable-openssl
make shared_library && sudo make install
```

Notice that the `--enable-openssl` part is *important*, as it's needed for AES-GCM support. As an alternative, you can also pass `--enable-nss` to have libsrtp use NSS instead of OpenSSL. A failure to configure libsrtp with either might cause undefined references when starting Janus, as we'd be trying to use methods that aren't there.

The Janus configure script autodetects which one you have installed and links to the correct library automatically, choosing 2.x if both are installed. If you want 1.5 or 1.6 to be picked (which is NOT recommended), pass `--disable-libsrtp2` when configuring Janus to force it to use the older version instead.

- *Note:* when installing libsrtp, no matter which version, you may need to pass `--libdir=/usr/lib64` to the configure script if you're installing on a x86\_64 distribution.

If you want to make use of BoringSSL instead of OpenSSL (e.g., because you want to take advantage of `--enable-dtls-settimeout`), you'll have to manually install it to a specific location. Use the following steps:

```
git clone https://boringssl.googlesource.com/boringssl
cd boringssl
# Don't barf on errors
sed -i s/" -Werror"/g CMakeLists.txt
# Build
mkdir -p build
cd build
cmake -DCMAKE_CXX_FLAGS="-lrt" ..
make
cd ..
# Install
sudo mkdir -p /opt/boringssl
sudo cp -R include /opt/boringssl/
sudo mkdir -p /opt/boringssl/lib
sudo cp build/ssl/libssl.a /opt/boringssl/lib/
sudo cp build/crypto/libcrypto.a /opt/boringssl/lib/
```

Once the library is installed, you'll have to pass an additional `--enable-boringssl` flag to the configure script, as by default Janus will be built assuming OpenSSL will be used. By default, Janus expects BoringSSL to be installed in `/opt/boringssl` -- if it's installed in another location, pass the path to the configure script as such: `--enable-boringssl=/path/to/boringssl` If you were using OpenSSL and want to switch to BoringSSL, make sure you also do a `make clean` in the Janus folder before compiling with the new BoringSSL support. If you enabled BoringSSL support and also want Janus to detect and react to DTLS timeouts with faster retransmissions, then pass `--enable-dtls-settimeout` to the configure script too.

For what concerns `usrctp`, which is needed for Data Channels support, it is usually not available in repositories, so if you're interested in them (support is optional) you'll have to install it manually. It is a pretty easy and standard process:

```
git clone https://github.com/sctplab/usrctp
cd usrctp
./bootstrap
./configure --prefix=/usr --disable-programs --disable-inet --disable-
inet6
make && sudo make install
```

- *Note:* you may need to pass `--libdir=/usr/lib64` to the configure script if you're installing on a `x86_64` distribution.

The same applies for `libwebsockets`, which is needed for the optional WebSockets support. If you're interested in supporting WebSockets to control Janus, as an alternative (or replacement) to the default plain HTTP REST API, you'll have to install it manually:

```
git clone https://libwebsockets.org/repo/libwebsockets
cd libwebsockets
# If you want the stable version of libwebsockets, uncomment the next line
# git checkout v3.2-stable
mkdir build
cd build
# See https://github.com/meetecho/janus-gateway/issues/732 re: LWS_MAX_SMP
# See https://github.com/meetecho/janus-gateway/issues/2476 re:
LWS_WITHOUT_EXTENSIONS
cmake -DLWS_MAX_SMP=1 -DLWS_WITHOUT_EXTENSIONS=0 -
DCMAKE_INSTALL_PREFIX=/usr -DCMAKE_C_FLAGS="-fpic" ..
make && sudo make install
```

- *Note:* if `libwebsockets.org` is unreachable for any reason, replace the first line with this:

git clone <https://github.com/warmcat/libwebsockets.git>

The same applies for Eclipse Paho MQTT C client library, which is needed for the optional MQTT support. If you're interested in integrating MQTT channels as an alternative (or replacement) to HTTP and/or WebSockets to control Janus, or as a carrier of Janus Events, you can install the latest version with the following steps:

```
git clone https://github.com/eclipse/paho.mqtt.c.git
cd paho.mqtt.c
make && sudo make install
```

- *Note:* you may want to set up a different install path for the library, to achieve that, replace the last command by 'sudo prefix=/usr make install'.

In case you're interested in Nanomsg support, you'll need to install the related C library. It is usually available as an easily installable package in pretty much all repositories. The following is an example on how to install it on Ubuntu:

```
aptitude install libnanomsg-dev
```

Finally, the same can be said for rabbitmq-c as well, which is needed for the optional RabbitMQ support. In fact, several different versions of the library can be found, and the versions usually available in most distribution repositories are not up-do-date with respect to the current state of the development. As such, if you're interested in integrating RabbitMQ queues as an alternative (or replacement) to HTTP and/or WebSockets to control Janus, you can install the latest version with the following steps:

```
git clone https://github.com/alanxz/rabbitmq-c
cd rabbitmq-c
git submodule init
git submodule update
mkdir build && cd build
cmake -DCMAKE_INSTALL_PREFIX=/usr ..
make && sudo make install
```

- *Note:* you may need to pass `--libdir=/usr/lib64` to the configure script if you're installing on a x86\_64 distribution.

To conclude, should you be interested in building the Janus documentation as well, you'll need some additional tools too:

- [Doxygen](#)

- [Graphviz](#)

On Fedora:

```
yum install doxygen graphviz
```

On Ubuntu/Debian:

```
aptitude install doxygen graphviz
```

## Compile

---

Once you have installed all the dependencies, get the code:

```
git clone https://github.com/meetecho/janus-gateway.git
cd janus-gateway
```

Then just use:

```
sh autogen.sh
```

to generate the configure file. After that, configure and compile as usual to start the whole compilation process:

```
./configure --prefix=/opt/janus
make
make install
```

Since Janus requires configuration files for both the core and its modules in order to work, you'll probably also want to install the default configuration files to use, which you can do this way:

```
make configs
```

Remember to only do this once, or otherwise a subsequent `make configs` will overwrite any configuration file you may have modified in the meanwhile.

If you've installed the above libraries but are not interested, for instance, in Data Channels, WebSockets, MQTT and/or RabbitMQ, you can disable them when configuring:

```
./configure --disable-websockets --disable-data-channels --disable-rabbitmq --disable-mqtt
```

There are configuration flags for pretty much all external modules and many of the features, so you may want to issue a `./configure --help` to dig through the available options. A summary of what's going to be built will always appear after you do a configure, allowing you to double check if what you need and don't need is there.

If Doxygen and graphviz are available, the process can also build the documentation for you. By default the compilation process will not try to build the documentation, so if you instead prefer to build it, use the `--enable-docs` configuration option:

```
./configure --enable-docs
```

You can also selectively enable/disable other features (e.g., specific plugins you don't care about, or whether or not you want to build the recordings post-processor). Use the `--help` option when configuring for more info.

## Building on FreeBSD

- *Note:* `rtp_forward` of streams only works streaming to IPv6, because of #2051 and thus the feature is not supported on FreeBSD at the moment.

When building on FreeBSD you can install the dependencies from ports or packages, here only `pkg` method is used. You also need to use `gmake` instead of `make`, since it is a GNU makefile. `./configure` can be run without arguments since the default prefix is `/usr/local` which is your default `LOCALBASE`. Note that the `configure.ac` is coded to use `openssl` in base. If you wish to use `openssl` from ports or any other `ssl` you must change `configure.ac` accordingly.

```
pkg install libsrtp2 libusrsctp jansson libnice libmicrohttpd
libwebsockets curl opus sofia-sip libogg jansson libnice libconfig \
libtool gmake autoconf autoconf-wrapper glib gengetopt
```

## Building on MacOS

While most of the above instructions will work when compiling Janus on MacOS as well, there are a few aspects to highlight when doing that.



First of all, you can use `brew` to install most of the dependencies:

```
brew install jansson libnice openssl srtp libsrtp libmicrohttpd \
    libwebsockets cmake rabbitmq-c sofia-sip opus libogg curl glib \
    libconfig pkg-config gengetopt autoconf automake libtool
```

For what concerns `libwebsockets`, though, make sure that the installed version is higher than `2.4.1`, or you might encounter the problems described in [this post](#). If `brew` doesn't provide a more recent version, you'll have to install the library manually.

Notice that you may need to provide a custom `prefix` and `PKG_CONFIG_PATH` when configuring Janus as well, e.g.:

```
./configure --prefix=/usr/local/janus
PKG_CONFIG_PATH=/usr/local/opt/openssl/lib/pkgconfig
```

Everything else works exactly the same way as on Linux.

## Configure and start

To start the server, you can use the `janus` executable. There are several things you can configure, either in a configuration file:

```
<installdir>/etc/janus/janus.jcfg
```

or on the command line:

```
<installdir>/bin/janus --help
```

```
Usage: janus [OPTIONS]...
```

<code>-h, --help</code>	Print help and exit
<code>-V, --version</code>	Print version and exit
<code>-b, --daemon</code>	Launch Janus in background as a daemon (default=off)
<code>-p, --pid-file=path</code>	Open the specified PID file when starting Janus (default=none)
<code>-N, --disable-stdout</code>	Disable stdout based logging (default=off)
<code>-L, --log-file=path</code>	Log to the specified file (default=stdout only)
<code>-H --cwd-path</code>	Working directory for Janus daemon process (default=/)

-i, --interface=ipaddress	Interface to use (will be the public IP)
-P, --plugins-folder=path	Plugins folder (default=./plugins)
-C, --config=filename	Configuration file to use
-F, --configs-folder=path	Configuration files folder (default=./conf)
-c, --cert-pem=filename	DTLS certificate
-k, --cert-key=filename	DTLS certificate key
-K, --cert-pwd=text	DTLS certificate key passphrase (if needed)
-S, --stun-server=address:port	STUN server(:port) to use, if needed (e.g., Janus behind NAT, default=none)
-1, --nat-1-1=ip	Public IP to put in all host candidates, assuming a 1:1 NAT is in place (e.g., Amazon EC2 instances, default=none)
-2, --keep-private-host	When nat-1-1 is used (e.g., Amazon EC2 instances), don't remove the private host, but keep both to simulate STUN
(default=off)	
-E, --ice-enforce-list=list	Comma-separated list of the only interfaces to use for ICE gathering; partial strings are supported (e.g., eth0 or eno1,wlan0, default=none)
-X, --ice-ignore-list=list	Comma-separated list of interfaces or IP addresses to ignore for ICE gathering; partial strings are supported (e.g., vmnet8,192.168.0.1,10.0.0.1 or vmnet,192.168., default=vmnet)
-6, --ipv6-candidates	Whether to enable IPv6 candidates or not (experimental) (default=off)
-0, --ipv6-link-local	Whether IPv6 link-local candidates should be gathered as well (default=off)
-l, --libnice-debug	Whether to enable libnice debugging or not (default=off)
-f, --full-trickle	Do full-trickle instead of half-trickle (default=off)
-I, --ice-lite	Whether to enable the ICE Lite mode or not (default=off)
-T, --ice-tcp	Whether to enable ICE-TCP or not (warning: only works with ICE Lite) (default=off)
-Q, --min-nack-queue=number	Minimum size of the NACK queue (in ms) per user for retransmissions, no matter the RTT
-t, --no-media-timer=number	Time (in s) that should pass with no media (audio or video) being received before Janus notifies you about this
-W, --slowlink-threshold=number	Number of lost packets (per s) that should trigger a 'slowlink' Janus API event to users

```

-r, --rtp-port-range=min-max  Port range to use for RTP/RTCP (only
available                                                                if the installed
libnice supports it)
-B, --twcc-period=number      How often (in ms) to send TWCC feedback back
to senders, if negotiated (default=200ms)
-n, --server-name=name       Public name of this Janus instance
                             (default=MyJanusInstance)
-s, --session-timeout=number  Session timeout value, in seconds
(default=60)
-m, --reclaim-session-timeout=number
                             Reclaim session timeout value, in seconds
                             (default=0)
-d, --debug-level=1-7        Debug/logging level (0=disable debugging,
                             7=maximum debug level; default=4)
-D, --debug-timestamps       Enable debug/logging timestamps
(default=off)
-o, --disable-colors         Disable color in the logging (default=off)
-M, --debug-locks            Enable debugging of locks/mutexes (very
                             verbose!) (default=off)
-a, --apisecret=randomstring API secret all requests need to pass in
order to be accepted by Janus (useful when
wrapping Janus API requests in a server, none by
                             default)
-A, --token-auth             Enable token-based authentication for all
                             requests (default=off)
-e, --event-handlers         Enable event handlers (default=off)
-w, --no-webrtc-encryption   Disable WebRTC encryption, so no DTLS or
SRTP (only for debugging!) (default=off)

```

Options passed through the command line have the precedence on those specified in the configuration file. To start the server, simply run:

```
<installdir>/bin/janus
```

This will start the server, and have it look at the configuration file.

Make sure you have a look at all of the configuration files, to tailor Janus to your specific needs: each configuration file is documented, so it shouldn't be hard to make changes according to your requirements. The repo comes with some defaults (assuming you issues `make configs` after installing the server) that tend to make sense for generic deployments, and also includes some sample configurations for all the plugins (e.g., web servers to listen on, conference rooms to create, streaming mountpoints to make available at startup, etc.).

To test whether it's working correctly, you can use the demos provided with this package in the `html` folder: these are exactly the same demos available online on the [project website](#). Just copy the file it contains in a webserver, or use a userspace webserver to serve the files in the `html` folder (e.g., with php or python), and open the `index.html` page in either Chrome or Firefox. A list of demo pages exploiting the different plugins will be available. Remember to edit the transport/port details in the demo JavaScript files if you changed any transport-related configuration from its defaults. Besides, the demos refer to the pre-configured plugin resources, so if you add some new resources (e.g., a new videoconference) you may have to tweak the demo pages to actually use them.

## Documentation

---

Janus is thoroughly documented. You can find the current documentation, automatically generated with Doxygen, on the [project website](#).

## Help us!

---

Any thought, feedback or (hopefully not!) insult is welcome!

Developed by [@meetecho](#)

## Releases

 54 tags

## Packages

No packages published

## Used by 62



## Contributors 236



+ 225 contributors

## Languages

