ntopng development in NetBeans IDE

Version 1.0

Aug 2015

Simone Mainardi, mainardi@ntop.org

Index

Preface

NetBeans is a powerful, widespread Integrated Development Environment (IDE), that supports multiple languages including C and C++. Features such as syntax highlighting, code completion, integrated versioning and debugging make this IDE a very effective tool.
This guide describes how to set up NetBeans to develop and debug ntopng on a Unix operating system such as Linux, and Mac OS X.

Download and Installation

NetBeans

NetBeans can be downloaded from the official website https://netbeans.org/downloads/. There are many flavors ready to be downloaded. If available disk space is not a concern, you are encouraged to get the 'All' technologies bundle that has extra support for HTML. Otherwise, the 'C/C++' bundle may suffice.

NetBeans IDE Download Bundles

Supported technologies *	Java SE	Java EE	C/C++	HTML5 & PHP	All
NetBeans Platform SDK	●	●			●
Java SE	●	●			●
Java FX	●	●			●
Java EE		●			●
Java ME					–
HTML5		●		●	●
Java Card™ 3 Connected					–
C/C++			●		●
Groovy					●
PHP				●	●
Bundled servers					
GlassFish Server Open Source Edition 4.1		●			●
Apache Tomcat 8.0.15		●			●
	Download	Download	Download	Download	Download
	Free, 105 MB	Free, 222 MB	Free, 72 MB	Free, 72 MB	Free, 243 MB

Installation is straightforward. Simply click on the download archive and follow the instructions.

Alternatively, you may install NetBeans via your distribution package manager — e.g., apt-get, rpm, brew — but changes are that the package will not contain the latest version.

ntopng

ntopng source code is available at https://github.com/ntop/ntopng. You can clone the repository in a local copy using git.

```
git clone https://github.com/ntop/ntopng.git
```

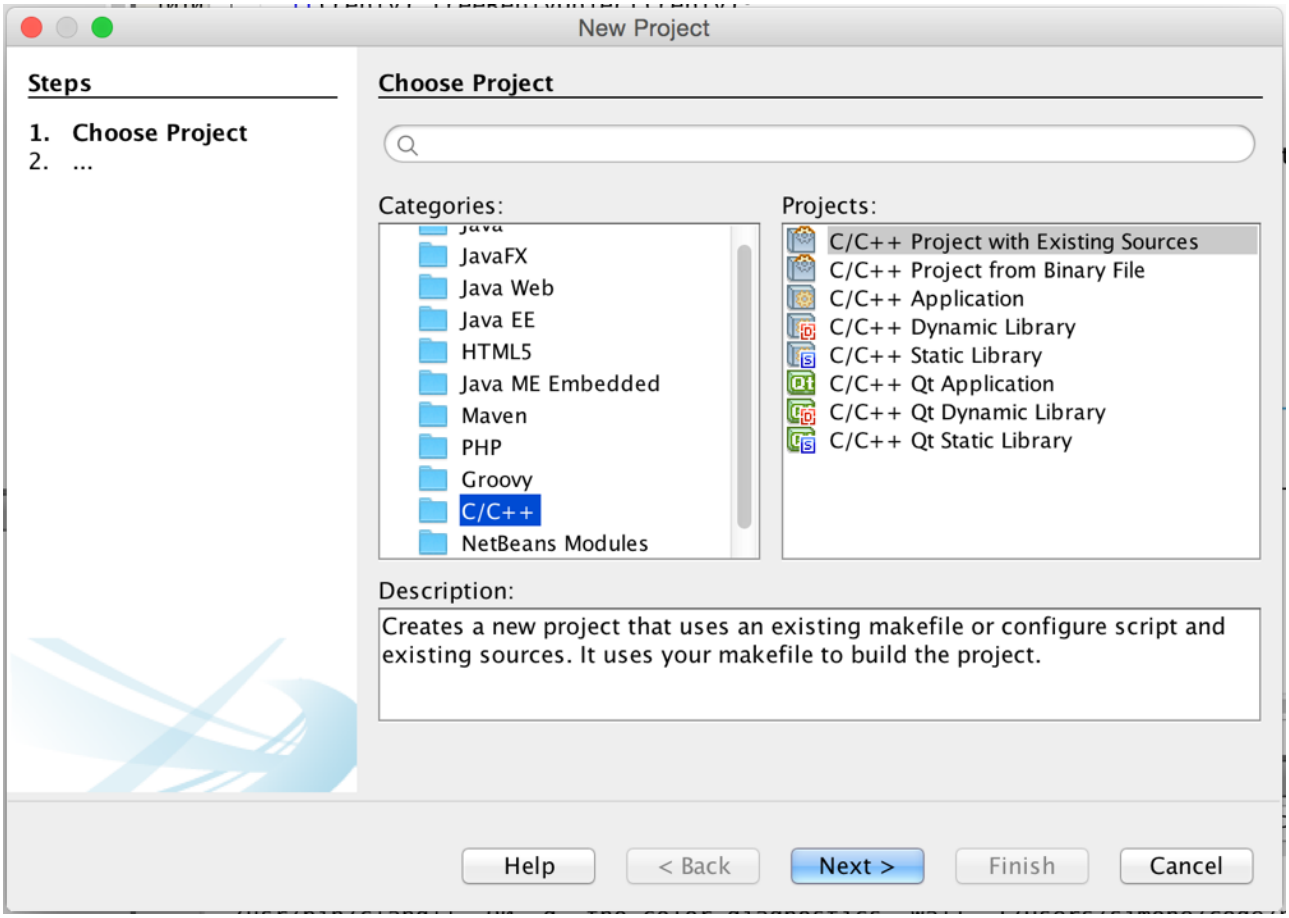
Once downloaded, a Makefile for the project has to be generated. cd into ntopng folder and run

```
Simones-MBP:code simone$ cd ntopng/
Simones-MBP:ntopng simone$ ./autogen.sh && ./configure
```

Upon successful completion of both autogen.sh and configure, you will find a Makefile file in the root of the project. This file contains all the necessary information to build ntopng in your system. NetBeans will use this file to compile and build the software.

Importing ntopng in NetBeans

ntopng can be imported in NetBeans as C/C++ project with existing sources. Open NetBeans and select File -> New Project -> C/C++ Project with existing sources, then click Next.



You will be prompted to specify the folder containing the sources. Click ‘Browse’ and navigate to the ntopng folder. Leave default settings for ‘Build Host’, ’Tools Collection’ and ‘Configuration Mode’ and confirm.
NetBeans will start to compile the sources. Compilation process and status from are shown on a tabbed window at the bottom of NetBeans. Once completed, a ‘MAKE SUCCESSFUL’ message should be displayed.

Now you are ready to develop ntopng. Navigate the project tree on the left side of the IDE. Open files, edit them, and compile the project by clicking on the top toolbar *Hammer* button. However, there is still some work to do in order to run ntopng. Simply clicking on the run *Play* button will not suffice since ntopng requires administrative privileges to start. The following section will discuss how to grant ntopng the necessary privileges.

Running ntopng in NetBeans

Grant ntopng administrative privileges

ntopng requires administrative privileges to run, since it has to access network interfaces. From the command line, one can simply run ntopng with *sudo* and insert the root password when prompted. This is not possible in NetBeans.
Therefore, it is necessary to tell sudo to execute ntopng without prompting for passwords. Open a shell and type *sudo visudo* to configure super user preferences.

```
Simones-MBP:ntopng simone$ sudo visudo
```

A text editor will open with a preference file. At the very bottom of the file add the following line

```
<your_user> ALL=NOPASSWD: <full_path_to_ntopng_executable>
```

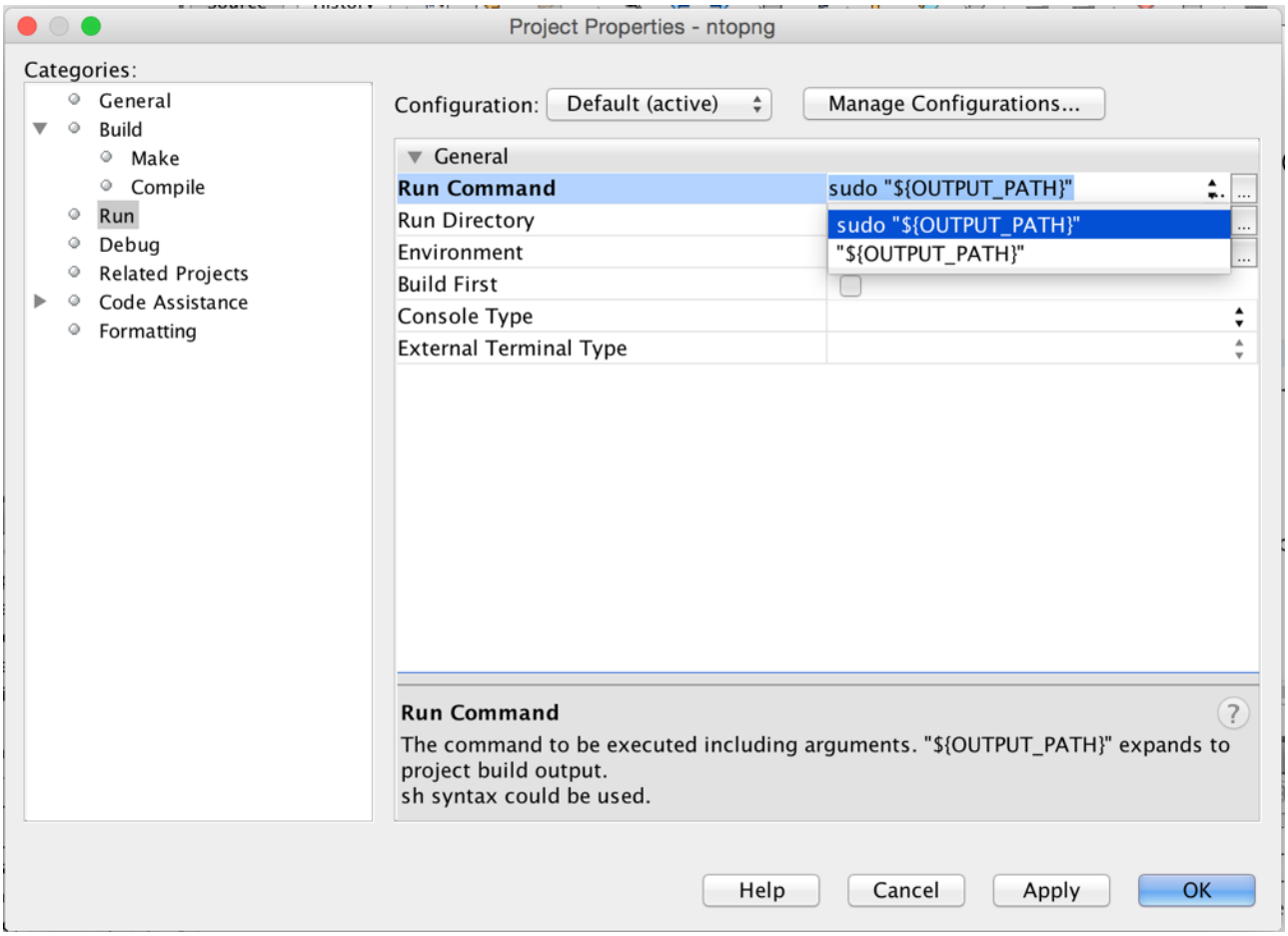
Then save and close. Replace <your_user> with the user that runs NetBeans and <full_path_to_ntopng_executable> with the absolute path of the ntopng executable. The executable is called *ntopng* and can be found in the root directory of the ntopng project folder. In my case, I added the following line

```
simone ALL=NOPASSWD: /Users/simone/code/ntopng/ntopng
```

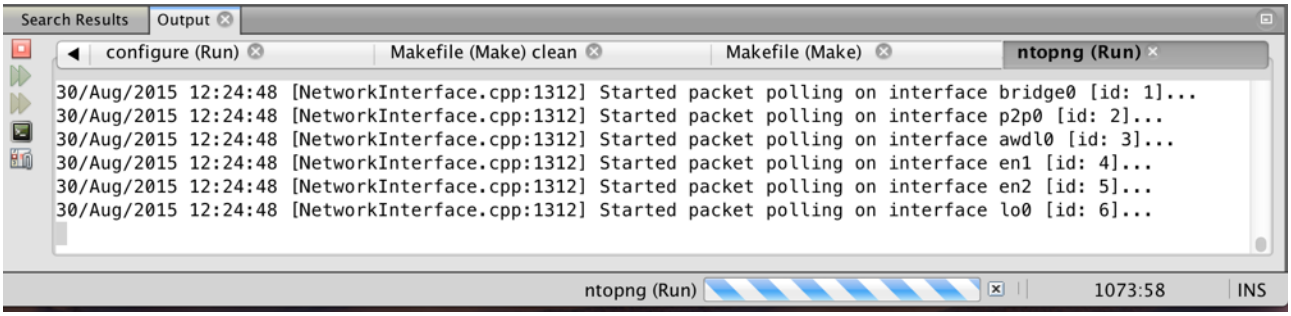
Now ntopng can be executed with administrative privileges without passwords. To verify, open a new shell terminal, go to the ntopng project, and run *sudo ntopng*. It should start without asking for super user password.The next step consists in telling NetBeans to run ntopng using *sudo*.

Run sudo ntopng in NetBeans

Right click on the ntopng project, and select Set Configuration -> Customize. Then, select the category ‘Run’ from the tree on the left, and add prepend the word *sudo* to the Run command. On Mac OS X, run defaults to NetBeans variable ${OUTPUT_PATH}. This may not be true for other operating systems. Nevertheless, you should not worry about it. Prepending the word sudo to the default command may suffice. One done, Click ‘Apply’, and confirm with ‘OK’.



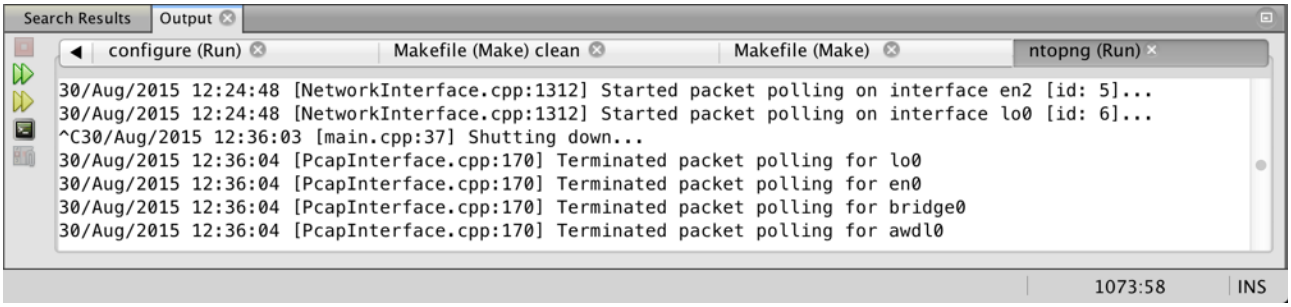
At this point we are ready to run ntopng within NetBeans. Click on the green Play button on the top toolbar. A tabbed window ‘Output’ should pop at the bottom of the IDE.



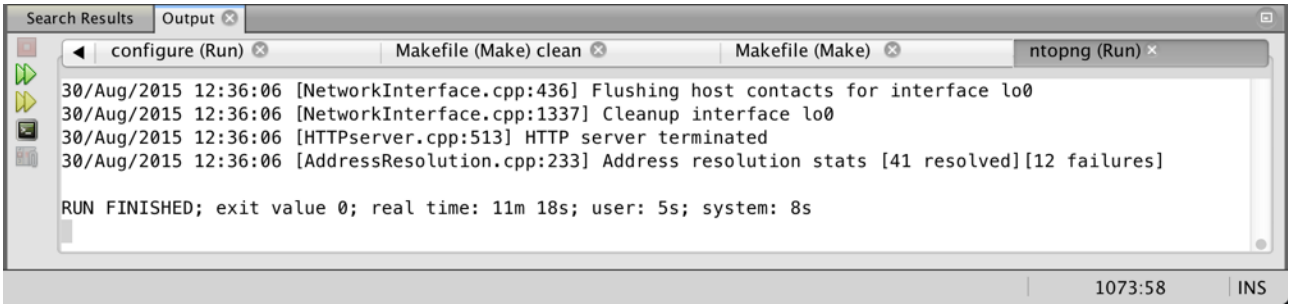
Hooray, ntopng has successfully started within the IDE. Try to fire up a browser and point it to localhost:3000. ntopng instance should serve your requests and print additional output to the window.

Stop ntopng instance running in NetBeans

To stop ntopng, hitting the red square 'stop' button may not be enough. Recall that ntopng runs with administrative privileges and hence, NetBeans — that is run by an unprivileged user — may not have sufficient privileges to stop it.
Therefore, to make sure ntopng is successfully stopped, a CTRL-C command has to be injected into the 'Output' window. Click somewhere on the window to make it active and then, hit CTRL +C.



'Shutting down' will be displayed and ntopng will start to gracefully shut down. Eventually, a RUN FINISHED message will pop to confirm the software has been terminated.



Debugging ntopng in NetBeans

In order to debug ntopng, the GNU gdb debugger has to be installed on the system. Debugger installation falls outside the scope of this guide. For the installation please have a look at tutorials specific for your operating system. The most straightforward way to install gdb is to install it via your package manager, e.g., apt-get install gdb. Once installed, the debugger will be available as an executable in the system. Type `which gdb` in a terminal to see the full path of the executable.

```
Simones-MBP:tmp simone$ which gdb
```

A path should be output. Typical installation paths are

```
/opt/local/bin/gdb
/bin
/usr/bin
```

If you can't find gdb, try with `which ggdb`. Some package managers such as mac ports install it under the name of ggdb. In my case, I have gdb installed with the name ggdb in /opt/local/bin.

```
Simones-MBP:tmp simone$ which ggdb
/opt/local/bin/ggdb
Simones-MBP:tmp simone$ /opt/local/bin/ggdb
GNU gdb (GDB) 7.9.1
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/
gpl.html>
[…]
(gdb) …
```

Note for Mac OS X users: If you are on Mac OS X, you have to certify the debugger in order for it to take control of other applications. Please refer to this guide: http://ntraft.com/installing-gdb-on-os-x-mavericks/.

Once the full path to the debugger executable is known, a one-line wrapper script for the debugger has to be created. ntopng requires administrative privileges and the debugger must have such privileges to take control of it. Go to the ntopng project folder, cd into tools, and touch a file called gdbsudo

```
Simones-MBP:code simone$ cd ntopng
Simones-MBP:ntopng simone$ cd tools/
Simones-MBP:tools simone$ touch gdbsudo
```

Now edit the file gdbsudo and add the following contents:

```
#!/bin/bash
echo "I'm going to call sudo ggdb with options " $*
sudo /opt/local/bin/ggdb $*
```

Make sure to *change* the full path to the debugger so that it **matches** the path output by the `which gdb` command as discussed above. This one-line wrapper simply calls the debugger with sudo, and forwards any argument to it.

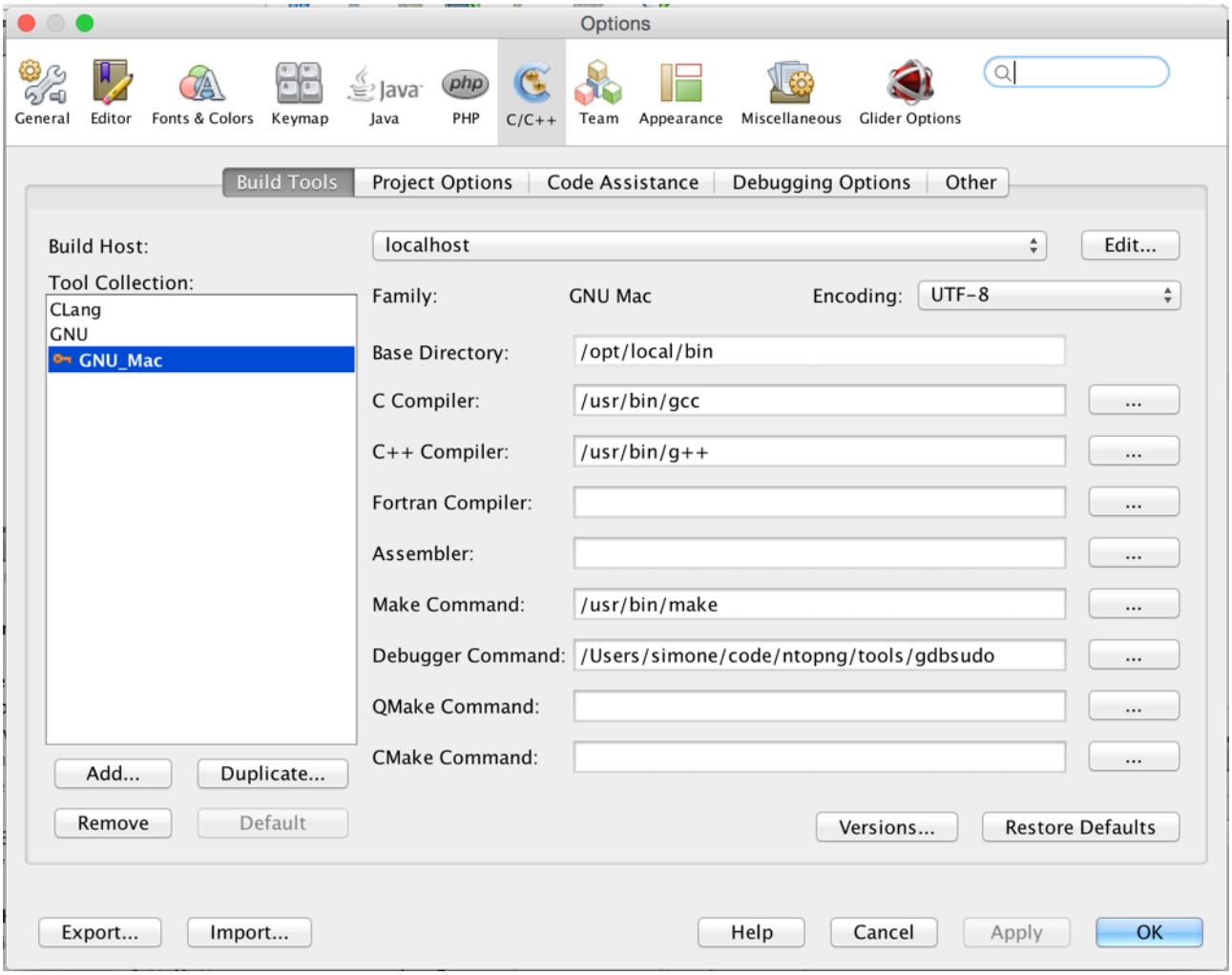
gdbsudo would prompt for the administrator password when executed. Since we want to call it from NetBeans, we have to tell *sudo* not to ask for passwords when running the debugger.
In a shell, type *sudo visudo* (see section 'Grant ntopng administrative privileges' above) and, at the very bottom of the file, add

```
<your_user> ALL=NOPASSWD: <full_path_to_the_debugger>
```

Again, make sure to change <full_path_to_the_debugger> so that it matches the path output by the `which gdb` command as discussed above. Also substitute <your_user> with the user name NetBeans is run with. In my case, I added the following line

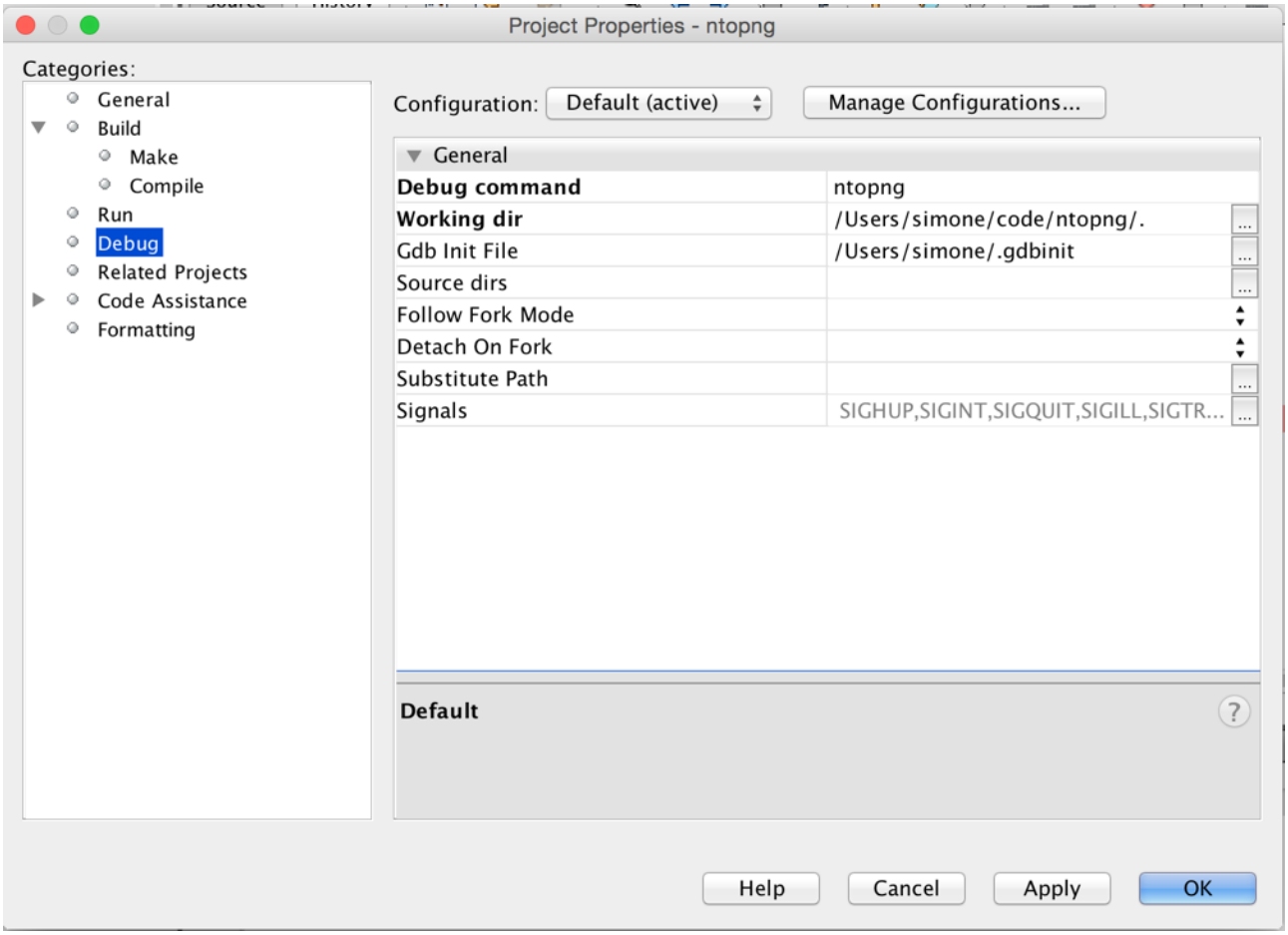
```
simone ALL=NOPASSWD: /opt/local/bin/ggdb
```

Now we are ready to tell NetBeans to use the one-line wrapper as debugger for ntopng. Open NetBeans Preferences, tab C/C++ and specify the absolute path to the one-line wrapper in the 'Debugger Command'.

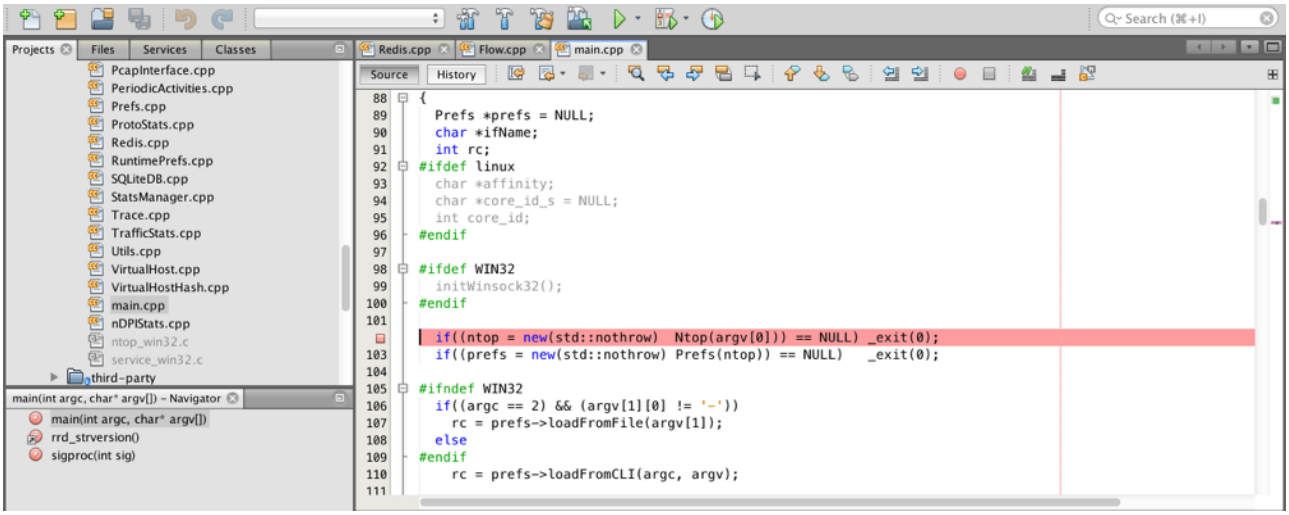


Make sure you are editing the right Tool Collection on the left. That is, the tool collection used to build and debug ntopng.
In my case, I created a new collection ‘GNU_Mac’ to work with ntopng. This collection has base directory /opt/local/bin. With the default /usr/bin/ NetBeans was complaining about not finding tools (e.g., wget, pkg-config). I changed to /opt/local/bin since mac ports install tools and utilities there. You don’t have to worry about changing C and C++ compliers. ntopng has its own Makefile that is able to automatically detect them.
Click ‘Apply’ and then ‘OK’.

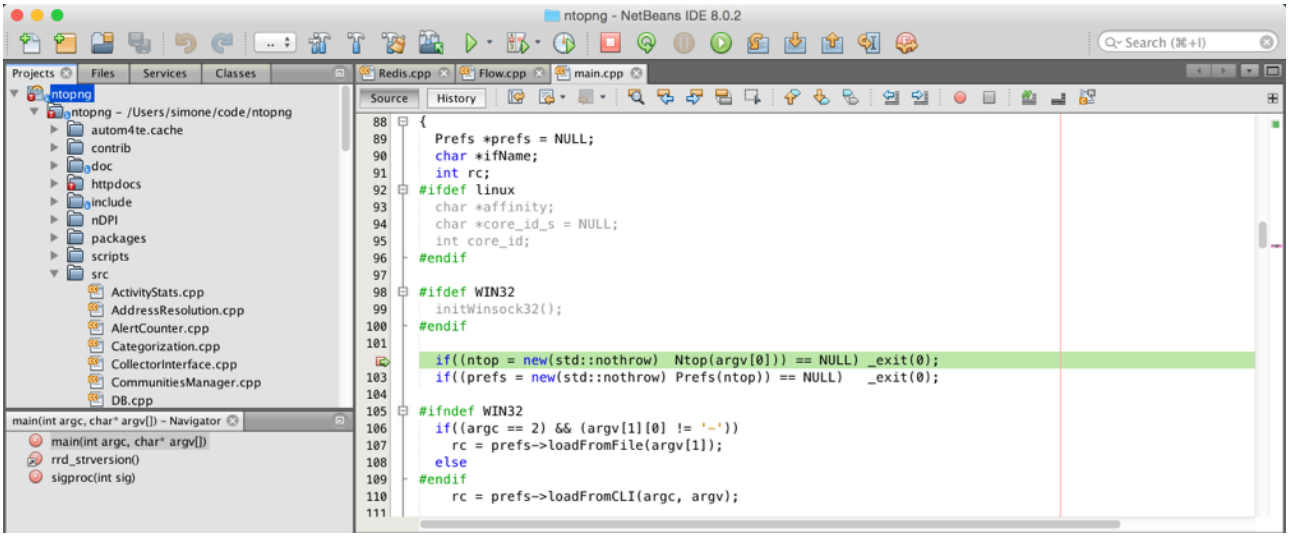
Now right click on ntopng project, ‘Set Configuration’ -> ‘Customize…’, and select the ‘Debug’ Category on the left. Write ntopng in the field ‘Debug command’, click ‘Apply’ and confirm with ‘OK’.



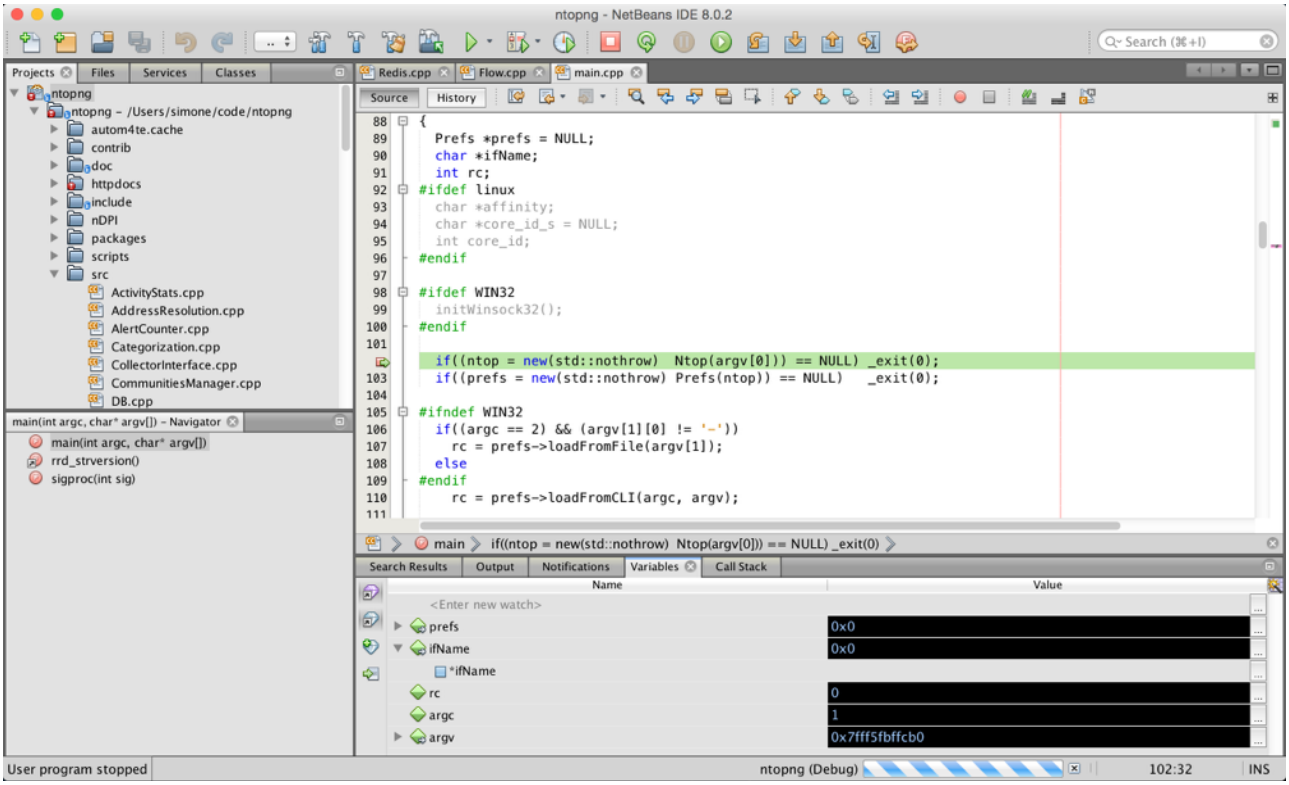
Wow. We are ready to debug ntopng! Open main.cpp source code file in src/ and add a break point somewhere in the main() function. Breakpoints are set by clicking on line numbers and are shown as small red squares.



Click on the 'Debug Project' button on the top toolbar, the first button on the right of the green Play button. NetBeans should open a tabbed window showing ntopng output. Once the breakpoint is reached, line highlighting will switch to green.



The debugger is ready. Hover the mouse on variables to see their current values. Similarly, select the 'Variables' tab to have an overall image of the current state.
To stop the debugger, click on the Output window and inject a CTRL-C (see section 'Stop ntopng instance running in NetBeans') or hit the stop red square button.


ntopng - NetBeans IDE 8.0.2
Projects
Files
Services
Classes
ntopng
ntopng – /Users/simone/code/ntopng
autom4te.cache
contrib
doc
httpdocs
include
nDPI
packages
scripts
src
ActivityStats.cpp
AddressResolution.cpp
AlertCounter.cpp
Categorization.cpp
CollectorInterface.cpp
CommunitiesManager.cpp
DB.cpp
main(int argc, char* argv[]) – Navigator
main(int argc, char* argv[])
rrd_strversion()
sigproc(int sig)
Redis.cpp
Flow.cpp
main.cpp
Source
History
{
Prefs *prefs = NULL;
char *ifName;
int rc;
#ifdef linux
char *affinity;
char *core_id_s = NULL;
int core_id;
#endif
#ifdef WIN32
initWinsock32();
#endif
if((ntop = new(std::nothrow) Ntop(argv[0])) == NULL) _exit(0);
if((prefs = new(std::nothrow) Prefs(ntop)) == NULL) _exit(0);
#ifndef WIN32
if((argc == 2) && (argv[1][0] != '-'))
rc = prefs->loadFromFile(argv[1]);
else
#endif
rc = prefs->loadFromCLI(argc, argv);
main
if((ntop = new(std::nothrow) Ntop(argv[0])) == NULL) _exit(0)
Search Results
Output
Notifications
Variables
Call Stack
Name
Value
<Enter new watch>
prefs
0x0
ifName
0x0
*ifName
rc
0
argc
1
argv
0x7fff5fbffcb0
User program stopped
ntopng (Debug)
102:32
INS