## Contents

1 Table of Contents

1.1 Install .................................................. 3
1.2 Quickstart ............................................. 3
1.3 Command Reference ................................. 9
1.4 Configuration ....................................... 13
1.5 Anatomy Of The Application .................... 16
1.6 Gateway To Other Services ...................... 35
1.7 API Documentation ................................. 44
1.8 Docker ............................................... 45
1.9 F.A.Q ................................................. 46
1.10 Support .............................................. 46

Index ....................................................... 47
Glances is a cross-platform monitoring tool which aims to present a maximum of information in a minimum of space through a curses or Web based interface. It can adapt dynamically the displayed information depending on the terminal size.

It can also work in client/server mode. Remote monitoring could be done via terminal, Web interface or API (XML-RPC and RESTful).

Glances is written in Python and uses the psutil library to get information from your system.

Stats can also be exported to external time/value databases.
1.1 Install

Glances is on PyPI. By using PyPI, you are sure to have the latest stable version.

To install, simply use pip:

\texttt{pip install glances}

\textit{Note:} Python headers are required to install psutil. For example, on Debian/Ubuntu you need to install first the \texttt{python-dev} package. For Fedora/CentOS/RHEL install first \texttt{python-devel} package. For Windows, just install psutil from the binary installation file.

You can also install the following libraries in order to use optional features (like the Web interface, export modules…):

\texttt{pip install glances[all]}

To upgrade Glances and all its dependencies to the latest versions:

\texttt{pip install --upgrade glances}
\texttt{pip install --upgrade psutil}
\texttt{pip install --upgrade glances[all]}

For additional installation methods, read the official \texttt{README} file.

1.2 Quickstart

This page gives a good introduction in how to get started with Glances. Glances offers 3 modes:

- Standalone
- Client/Server
- Web server
1.2.1 Standalone Mode

If you want to monitor your local machine, open a console/terminal and simply run:

```
$ glances
```

Glances should start (press ‘q’ or ‘ESC’ to exit):

```
```

It is also possible to display RAW JSON stats directly to stdout using:

```
$ glances --stdout cpu.user,mem.used,load
```

```
cpu.user: 30.7
mem.used: 3278204928
load: {'cpucore': 4, 'min1': 0.21, 'min5': 0.4, 'min15': 0.27}
```

```
cpu.user: 3.4
mem.used: 3275251712
load: {'cpucore': 4, 'min1': 0.19, 'min5': 0.39, 'min15': 0.27}
```

... or in a CSV format thanks to the stdout-csv option:

```
$ glances --stdout-csv now,cpu.user,mem.used,load
```

```
now,cpu.user,mem.used,load.cpucore,load.min1,load.min5,load.min15
2018-12-08 22:04:20 CEST,7.3,5948149760,4,1.04,0.99,1.04
2018-12-08 22:04:23 CEST,5.4,5949136896,4,1.04,0.99,1.04
```

Note: It will display one line per stat per refresh.
1.2.2 Client/Server Mode

If you want to remotely monitor a machine, called server, from another one, called client, just run on the server:

```bash
server$ glances -s
```

and on the client:

```bash
client$ glances -c @server
```

where @server is the IP address or hostname of the server.

In server mode, you can set the bind address with `-B ADDRESS` and the listening TCP port with `-p PORT`.

In client mode, you can set the TCP port of the server with `-p PORT`.

Default binding address is `0.0.0.0` (Glances will listen on all the available network interfaces) and TCP port is 61209.

In client/server mode, limits are set by the server side.

You can set a password to access to the server using the `--password`. By default, the username is `glances` but you can change it with `--username`.

If you want, the SHA password will be stored in `username.pwd` file. Next time your run the server/client, password will not be asked. To set a specific username you can used the `-u <username>` option.

It is also possible to set the password in the Glances configuration file:

```ini
[passwords]
# Define the passwords list
# Syntax: host=password
# Where: host is the hostname
# password is the clear password
# Additionally (and optionally) a default password could be defined
localhost=mylocalhostpassword
default=mydefaultpassword
```

Central client

Glances can centralize available Glances servers using the `--browser` option. The server list can be statically defined via the configuration file (section `[serverlist]`).

Example:

```ini
[serverlist]
# Define the static servers list
server_1_name=xps
server_1_alias=xps
server_1_port=61209
server_2_name=win
server_2_port=61235
```
Glances can also detect and display all Glances servers available on your network via the zeroconf protocol (not available on Windows):

To start the central client, use the following option:

```
client$ glances --browser
```

**Note:** Use `--disable-autodiscover` to disable the auto discovery mode.

When the list is displayed, you can navigate through the Glances servers with up/down keys. It is also possible to sort the server using: - ‘1’ is normal (do not sort) - ‘2’ is using sorting with ascending order (ONLINE > SNMP > PROTECTED > OFFLINE > UNKNOWN) - ‘3’ is using sorting with descending order (UNKNOWN > OFFLINE > PROTECTED > SNMP > ONLINE)

**SNMP**

As an experimental feature, if Glances server is not detected by the client, the latter will try to grab stats using the SNMP protocol:

```
client$ glances -c @snmpserver
```

**Note:** Stats grabbed by SNMP request are limited and OS dependent. A SNMP server should be installed and configured...

**IPv6**

Glances is IPv6 compatible. Just use the `-B ::` option to bind to all IPv6 addresses.
1.2.3 Web Server Mode

If you want to remotely monitor a machine, called server, from any device with a web browser, just run the server with the `-w` option:

```
server$ glances -w
```

then on the client enter the following URL in your favorite web browser:

```
http://@server:61208
```

where `@server` is the IP address or hostname of the server.

To change the refresh rate of the page, just add the period in seconds at the end of the URL. For example, to refresh the page every 10 seconds:

```
http://@server:61208/10
```

The Glances web interface follows responsive web design principles.

Here’s a screenshot from Chrome on Android:
1.3 Command Reference

1.3.1 Command-Line Options

- **h**, --help
  show this help message and exit

- **V**, --version
  show program’s version number and exit

- **d**, --debug
  enable debug mode

- **C**, CONF_FILE, --config CONF_FILE
  path to the configuration file

- **--modules-list**
  display modules (plugins & exports) list and exit

- **--disable-plugin PLUGIN**
  disable PLUGIN (comma separated list)

- **--enable-plugin PLUGIN**
  enable PLUGIN (comma separated list)

- **--stdout PLUGINS_STATS**
  display stats to stdout (comma separated list of plugins/plugins.attribute)

- **--export EXPORT**
  enable EXPORT module (comma separated list)

- **--export-csv-file EXPORT_CSV_FILE**
  file path for CSV exporter

- **--export-json-file EXPORT_JSON_FILE**
  file path for JSON exporter

- **--disable-process**
  disable process module (reduce Glances CPU consumption)

- **--disable-webui**
  disable the Web UI (only the RESTful API will respond)

- **--light**, --enable-light
  light mode for Curses UI (disable all but top menu)

- **-0**, --disable-irix
  task’s CPU usage will be divided by the total number of CPUs

- **-1**, --percpu
  start Glances in per CPU mode

- **-2**, --disable-left-sidebar
  disable network, disk I/O, FS and sensors modules

- **-3**, --disable-quicklook
  disable quick look module

- **-4**, --full-quicklook
  disable all but quick look and load
-5, --disable-top
  disable top menu (QuickLook, CPU, MEM, SW AP and LOAD)
-6, --meangpu
  start Glances in mean GPU mode
--enable-history
  enable the history mode
--disable-bold
  disable bold mode in the terminal
--disable-bg
  disable background colors in the terminal
--enable-process-extended
  enable extended stats on top process
-c CLIENT, --client CLIENT
  connect to a Glances server by IPv4/IPv6 address, hostname or hostname:port
-s, --server
  run Glances in server mode
--browser
  start the client browser (list of servers)
--disable-autodiscover
  disable autodiscover feature
-p PORT, --port PORT
  define the client/server TCP port [default: 61209]
-B BIND_ADDRESS, --bind BIND_ADDRESS
  bind server to the given IPv4/IPv6 address or hostname
--username
  define a client/server username
--password
  define a client/server password
--snmp-community SNMP_COMMUNITY
  SNMP community
--snmp-port SNMP_PORT
  SNMP port
--snmp-version SNMP_VERSION
  SNMP version (1, 2c or 3)
--snmp-user SNMP_USER
  SNMP username (only for SNMPv3)
--snmp-auth SNMP_AUTH
  SNMP authentication key (only for SNMPv3)
--snmp-force
  force SNMP mode
-t TIME, --time TIME
  set refresh time in seconds [default: 3 sec]
-w, --webserver
  run Glances in web server mode (bottle lib needed)

--cached-time  CACHED_TIME
  set the server cache time [default: 1 sec]

open-web-browser
  try to open the Web UI in the default Web browser

-q, --quiet
  do not display the curses interface

-f  PROCESS_FILTER, --process-filter  PROCESS_FILTER
  set the process filter pattern (regular expression)

--process-short-name
  force short name for processes name

--hide-kernel-threads
  hide kernel threads in process list (not available on Windows)

-b, --byte
  display network rate in byte per second

--diskio-show-ramfs
  show RAM FS in the DiskIO plugin

--diskio-iops
  show I/O per second in the DiskIO plugin

--fahrenheit
  display temperature in Fahrenheit (default is Celsius)

--fs-free-space
  display FS free space instead of used

--theme-white
  optimize display colors for white background

--disable-check-update
  disable online Glances version check

1.3.2 Interactive Commands

The following commands (key pressed) are supported while in Glances:

**ENTER**  Set the process filter

**Note:** On macOS please use **CTRL-H** to delete filter.

Filter is a regular expression pattern:

- gnome: matches all processes starting with the gnome string
- .*gnome.*: matches all processes containing the gnome string

a  Sort process list automatically

  - If CPU >70%, sort processes by CPU usage
  - If MEM >70%, sort processes by MEM usage
• If CPU iowait > 60%, sort processes by I/O read and write

A Enable/disable Application Monitoring Process
b Switch between bit/s or Byte/s for network I/O
B View disk I/O counters per second
c Sort processes by CPU usage
d Show/hide disk I/O stats
D Enable/disable Docker stats
e Enable/disable top extended stats
E Erase current process filter
f Show/hide file system and folder monitoring stats
F Switch between file system used and free space
g Generate graphs for current history
h Show/hide the help screen
i Sort processes by I/O rate
I Show/hide IP module
k Show/hide TCP connections
l Show/hide log messages
m Sort processes by MEM usage
M Reset processes summary min/max
n Show/hide network stats
N Show/hide current time
p Sort processes by name
q|ESC|CTRL–C Quit the current Glances session
Q Show/hide IRQ module
r Reset history
R Show/hide RAID plugin
s Show/hide sensors stats
t Sort process by CPU times (TIME+)
T View network I/O as combination
u Sort processes by USER
U View cumulative network I/O
w Delete finished warning log messages
W Show/hide Wifi module
x Delete finished warning and critical log messages
z Show/hide processes stats
0  Enable/disable Irix/Solaris mode
   Task’s CPU usage will be divided by the total number of CPUs
1  Switch between global CPU and per-CPU stats
2  Enable/disable left sidebar
3  Enable/disable the quick look module
4  Enable/disable all but quick look and load module
5  Enable/disable top menu (QuickLook, CPU, MEM, SWAP and LOAD)
6  Enable/disable mean GPU mode
   /  Switch between process command line or command name
In the Glances client browser (accessible through the --browser command line argument):

  ENTER  Run the selected server
  UP     Up in the servers list
  DOWN   Down in the servers list
  q|ESC   Quit Glances

1.4 Configuration

No configuration file is mandatory to use Glances.
Furthermore a configuration file is needed to access more settings.

1.4.1 Location

Note: A template is available in the /usr{,/local}/share/doc/glances (Unix-like) directory or directly on GitHub.

You can put your own glances.conf file in the following locations:

<table>
<thead>
<tr>
<th>Platform</th>
<th>Directory</th>
</tr>
</thead>
<tbody>
<tr>
<td>Linux, SunOS</td>
<td>~/.config/glances, /etc/glances</td>
</tr>
<tr>
<td>*BSD</td>
<td>~/.config/glances, /usr/local/etc/glances</td>
</tr>
<tr>
<td>macOS</td>
<td>~/Library/Application Support/glances, /usr/local/etc/glances</td>
</tr>
<tr>
<td>Windows</td>
<td>%APPDATA%\glances</td>
</tr>
</tbody>
</table>

- On Windows XP, %APPDATA% is: C:\Documents and Settings\<USERNAME>\Application Data.
- On Windows Vista and later: C:\Users\<USERNAME>\AppData\Roaming.

User-specific options override system-wide options and options given on the command line override either.
1.4.2 Syntax

Glances reads configuration files in the *ini* syntax.

A first section (called global) is available:

```ini
[gLOBAL]
# Does Glances should check if a newer version is available on PyPI?
check_update=true
```

Each plugin, export module and application monitoring process (AMP) can have a section. Below an example for the CPU plugin:

```ini
[cpu]
disable=False
user_careful=50
user_critical=90
iowait_careful=50
iowait_warning=70
iowait_critical=90
system_careful=50
system_warning=70
system_critical=90
steal_careful=50
steal_warning=70
steal_critical=90
```

an InfluxDB export module:

```ini
[influxdb]
# Configuration for the --export influxdb option
# https://influxdb.com/
host=localhost
port=8086
user=root
password=root
db=glances
prefix=localhost
#tags=foo:bar,spam:eggs
```

or a Nginx AMP:

```ini
[amp_nginx]
# Nginx status page should be enable (https://easyengine.io/tutorials/nginx/status-page/)
enable=true
regex=/usr/sbin/nginx
refresh=60
one_line=false
status_url=http://localhost/nginx_status
```

With Glances 3.0 or higher it is also possible to use dynamic configuration value using system command. For example, if you to set the prefix of an InfluxDB export to the current hostname, use:

```ini
[influxdb]
...prefix=`hostname`
```
Or if you want to add the Operating System name as a tag:

```
[influxdb]
...
tag=system:`uname -a`
```

### 1.4.3 Logging

Glances logs all of its internal messages to a log file.

DEBUG messages can be logged using the `-d` option on the command line.

By default, the `glances-USERNAME.log` file is under the temporary directory:

<table>
<thead>
<tr>
<th>*nix</th>
<th>/tmp</th>
</tr>
</thead>
<tbody>
<tr>
<td>Windows</td>
<td>%TEMP%</td>
</tr>
</tbody>
</table>

- On Windows XP, `%TEMP%` is: C:\Documents and Settings\<USERNAME>\Local Settings\Temp.
- On Windows Vista and later: C:\Users\<USERNAME>\AppData\Local\Temp.

If you want to use another system path or change the log message, you can use your own logger configuration. First of all, you have to create a `glances.json` file with, for example, the following content (JSON format):

```json
{
  "version": 1,
  "disable_existing_loggers": "False",
  "root": {
    "level": "INFO",
    "handlers": ["file", "console"
  },
  "formatters": {
    "standard": {
      "format": "%(asctime)s -- %(levelname)s -- %(message)s"
    },
    "short": {
      "format": "%(levelname)s: %(message)s"
    },
    "free": {
      "format": "%(message)s"
    }
  },
  "handlers": {
    "file": {
      "level": "DEBUG",
      "class": "logging.handlers.RotatingFileHandler",
      "formatter": "standard",
      "filename": "/var/tmp/glances.log"
    },
    "console": {
      "level": "CRITICAL",
      "class": "logging.StreamHandler",
      "formatter": "free"
    }
  },
  "loggers": {
```

(continues on next page)
and start Glances using the following command line:

```
LOG_CFG=<path>/glances.json glances
```

Note: Replace `<path>` by the folder where your `glances.json` file is hosted.

## 1.5 Anatomy Of The Application

This document is meant to give an overview of the Glances interface.

Legend:

<table>
<thead>
<tr>
<th>Color</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>GREEN</td>
<td>OK</td>
</tr>
<tr>
<td>BLUE</td>
<td>CAREFUL</td>
</tr>
<tr>
<td>MAGENTA</td>
<td>WARNING</td>
</tr>
<tr>
<td>RED</td>
<td>CRITICAL</td>
</tr>
</tbody>
</table>

Note: Only stats with colored background will be shown in the alert view.
1.5.1 Header

The header shows the hostname, OS name, release version, platform architecture IP addresses (private and public) and system uptime. Additionally, on GNU/Linux, it also shows the kernel version.

In client mode, the server connection status is also displayed.

Connected:

Disconnected:

If you are hosted on an OpenStack instance, some additional information can be displayed (AMI-ID, region).

1.5.2 Quick Look

The quicklook plugin is only displayed on wide screen and proposes a bar view for CPU and memory (virtual and swap).

If the per CPU mode is on (by clicking the 1 key):

In the Curses/terminal interface, it is also possible to switch from bar to sparkline using ‘S’ hot key or –sparkline command line option (ned the sparklines Python lib on your system). Please be aware that sparklines use the Glances history and will not be available if the history is disabled from the command line.
Note: Limit values can be overwritten in the configuration file under the [quicklook] section.

You can also configure the percentage char used in the terminal user interface.

```
[quicklook]
# Graphical percentage char used in the terminal user interface (default is |)
percentage_char=@
```

### 1.5.3 CPU

The CPU stats are shown as a percentage or values and for the configured refresh time. The total CPU usage is displayed on the first line.

If enough horizontal space is available, extended CPU information are displayed.

A character is also displayed just after the CPU header and shows the trend value:

<table>
<thead>
<tr>
<th>Trend</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>CPU value is equal to the mean of the six latests refreshes</td>
</tr>
<tr>
<td>\</td>
<td>CPU value is lower than the mean of the six latests refreshes</td>
</tr>
<tr>
<td>/</td>
<td>CPU value is higher than the mean of the six latests refreshes</td>
</tr>
</tbody>
</table>

CPU stats description:

- **user**: percent time spent in user space. User CPU time is the time spent on the processor running your program’s code (or code in libraries).
- **system**: percent time spent in kernel space. System CPU time is the time spent running code in the Operating System kernel.
- **idle**: percent of CPU used by any program. Every program or task that runs on a computer system occupies a certain amount of processing time on the CPU. If the CPU has completed all tasks it is idle.
- **nice (linux)**: percent time occupied by user level processes with a positive nice value. The time the CPU has spent running users’ processes that have been *niced*.
- **irq (linux, *BSD)**: percent time spent servicing/handling hardware/software interrupts. Time servicing interrupts (hardware + software).
- **iowait (linux)**: percent time spent by the CPU waiting for I/O operations to complete.
• **steal** (*Linux*): percentage of time a virtual CPU waits for a real CPU while the hypervisor is servicing another virtual processor.

• **ctx_sw**: number of context switches (voluntary + involuntary) per second. A context switch is a procedure that a computer’s CPU (central processing unit) follows to change from one task (or process) to another while ensuring that the tasks do not conflict.

• **inter**: number of interrupts per second.

• **sw_inter**: number of software interrupts per second. Always set to 0 on Windows and SunOS.

• **syscal**: number of system calls per second. Do not displayed on Linux (always 0).

To switch to per-CPU stats, just hit the 1 key:

![CPU Usage Table](image)

By default, **steal** CPU time alerts aren’t logged. If you want that, just add to the configuration file:

```
[cpu]
steal_log=True
```

Legend:

<table>
<thead>
<tr>
<th>CPU (user/system)</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;50%</td>
<td>OK</td>
</tr>
<tr>
<td>&gt;50%</td>
<td>CAREFUL</td>
</tr>
<tr>
<td>&gt;70%</td>
<td>WARNING</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>CRITICAL</td>
</tr>
</tbody>
</table>

**Note:** Limit values can be overwritten in the configuration file under the [cpu] and/or [percpu] sections.

### 1.5.4 GPU

**Note:** You need to install the `nvidia-ml-py3` library on your system.

The GPU stats are shown as a percentage of value and for the configured refresh time. The total GPU usage is displayed on the first line, the memory consumption on the second one.

![GPU Usage](image)

If you click on the 6 short key, the per-GPU view is displayed:
Note: You can also start Glances with the `--meangpu` option to display the first view by default.

You can change the threshold limits in the configuration file:

```plaintext
[gpu]
# Default processor values if not defined: 50/70/90
proc_careful=50
proc_warning=70
proc_critical=90
# Default memory values if not defined: 50/70/90
mem_careful=50
mem_warning=70
mem_critical=90
```

Legend:

<table>
<thead>
<tr>
<th>GPU (PROC/MEM)</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;50%</td>
<td>OK</td>
</tr>
<tr>
<td>&gt;50%</td>
<td>CAREFUL</td>
</tr>
<tr>
<td>&gt;70%</td>
<td>WARNING</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>CRITICAL</td>
</tr>
</tbody>
</table>

1.5.5 Load

Availability: Unix and Windows with a PsUtil version >= 5.6.2

On the No Sheep blog, Zachary Tirrell defines the load average on GNU/Linux operating system:

“In short it is the average sum of the number of processes waiting in the run-queue plus the number currently executing over 1, 5, and 15 minutes time periods.”

Be aware that Load on Linux, BSD and Windows are different things, high load on BSD does not means high CPU load. The Windows load is emulated by the PsUtil lib (see load on Windows)

Glances gets the number of CPU core to adapt the alerts. Alerts on load average are only set on 15 minutes time period. The first line also displays the number of CPU core.

Legend:
<table>
<thead>
<tr>
<th>Load avg</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;0.7*core</td>
<td>OK</td>
</tr>
<tr>
<td>&gt;0.7*core</td>
<td>CAREFUL</td>
</tr>
<tr>
<td>&gt;1*core</td>
<td>WARNING</td>
</tr>
<tr>
<td>&gt;5*core</td>
<td>CRITICAL</td>
</tr>
</tbody>
</table>

Note: Limit values can be overwritten in the configuration file under the [load] section.

1.5.6 Memory

Glances uses two columns: one for the RAM and one for the SWAP.

If enough space is available, Glances displays extended information for the RAM:

A character is also displayed just after the MEM header and shows the trend value:

<table>
<thead>
<tr>
<th>Trend</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>-</td>
<td>MEM value is equal to the mean of the six latests refreshes</td>
</tr>
<tr>
<td>\</td>
<td>MEM value is lower than the mean of the six latests refreshes</td>
</tr>
<tr>
<td>/</td>
<td>MEM value is higher than the mean of the six latests refreshes</td>
</tr>
</tbody>
</table>

Alerts are only set for used memory and used swap.

Legend:

<table>
<thead>
<tr>
<th>RAM/Swap</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;50%</td>
<td>OK</td>
</tr>
<tr>
<td>&gt;50%</td>
<td>CAREFUL</td>
</tr>
<tr>
<td>&gt;70%</td>
<td>WARNING</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>CRITICAL</td>
</tr>
</tbody>
</table>

Note: Limit values can be overwritten in the configuration file under the [memory] and/or [memswap] sections.
1.5.7 Network

Glances displays the network interface bit rate. The unit is adapted dynamically (bit/s, kbit/s, Mbit/s, etc).

If the interface speed is detected (not on all systems), the defaults thresholds are applied (70% for careful, 80% warning and 90% critical). It is possible to define this percents thresholds from the configuration file. It is also possible to define per interface bit rate thresholds. In this case thresholds values are define in bps.

Additionally, you can define:

- a list of network interfaces to hide
- per-interface limit values
- aliases for interface name

The configuration should be done in the [network] section of the Glances configuration file.

For example, if you want to hide the loopback interface (lo) and all the virtual docker interface (docker0, docker1, ...):

```plaintext
[network]
# Default bitrate thresholds in % of the network interface speed
# Default values if not defined: 70/80/90
rx_careful=70
rx_warning=80
rx_critical=90
tx_careful=70
tx_warning=80
tx_critical=90

# Define the list of hidden network interfaces (comma-separated regexp)
hide=docker.*,lo

# WLAN 0 alias
wlan0_alias=Wireless IF

# It is possible to overwrite the bitrate thresholds per interface
# WLAN 0 Default limits (in bits per second aka bps) for interface bitrate
wlan0_rx_careful=4000000
wlan0_rx_warning=5000000
wlan0_rx_critical=6000000
wlan0_rx_log=True
wlan0_tx_careful=700000
wlan0_tx_warning=900000
wlan0_tx_critical=1000000
wlan0_tx_log=True
```
1.5.8 Connections

This plugin display extended information about network connections.

The states are the following:

- Listen: all ports created by server and waiting for a client to connect
- Initialized: All states when a connection is initialized (sum of SYN_SENT and SYN_RECEIVED)
- Established: All established connections between a client and a server
- Terminated: All states when a connection is terminated (FIN_WAIT1, CLOSE_WAIT, LAST_ACK, FIN_WAIT2, TIME_WAIT and CLOSE)
- Tracked: Current number and maximum Netfilter tracker connection (nf_conntrack_count/nf_conntrack_max)

The configuration should be done in the [connections] section of the Glances configuration file.

By default the plugin is disabled. Please change your configuration file as following to enable it

```
[connections]
disable=False
# nf_conntrack thresholds in %
nf_conntrack_percent_careful=70
nf_conntrack_percent_warning=80
nf_conntrack_percent_critical=90
```

1.5.9 Wi-Fi

Availability: Linux
Glances displays the Wi-Fi hotspot names and signal quality. If Glances is ran as root, then all the available hotspots are displayed.

**Note:** You need to install the `wireless-tools` package on your system.

In the configuration file, you can define signal quality thresholds:

- "**Poor**" quality is between -100 and -85dBm
- "**Good**" quality between -85 and -60dBm
- "**Excellent**" between -60 and -40dBm

It’s also possible to disable the scan on a specific interface from the configuration file ([wifi] section). For example, if you want to hide the loopback interface (lo) and all the virtual docker interfaces:

```
[wifi]
hide=lo,docker.*
# Define SIGNAL thresholds in dBm (lower is better...)
careful=-65
warning=-75
critical=-85
```

You can disable this plugin using the `--disable-wifi` option or by hitting the `W` key from the user interface.

### 1.5.10 Ports

**Availability:** All

This plugin aims at providing a list of hosts/port and URL to scan.

You can define ICMP or TCP ports scans and URL (head only) check.

The list should be defined in the `[ports]` section of the Glances configuration file.

```
[ports]
# Ports scanner plugin configuration
# Interval in second between two scans
refresh=30
# Set the default timeout (in second) for a scan (can be overwrite in the scan list)
timeout=3
# If port_default_gateway is True, add the default gateway on top of the scan list
port_default_gateway=True
```
# Define the scan list (1 < x < 255)
# port_x_host (name or IP) is mandatory
# port_x_port (TCP port number) is optional (if not set, use ICMP)
# port_x_description is optional (if not set, define to host:port)
# port_x_timeout is optional and overwrite the default timeout value
# port_x_rtt_warning is optional and defines the warning threshold in ms
#
port_1_host=192.168.0.1
port_1_port=80
port_1_description=Home Box
port_1_timeout=1
port_2_host=www.free.fr
port_2_description=My ISP
port_3_host=www.google.com
port_3_description=Internet ICMP
port_3_rtt_warning=1000
port_4_host=www.google.com
port_4_description=Internet Web
port_4_port=80
port_4_rtt_warning=1000
#
# Define Web (URL) monitoring list (1 < x < 255)
# web_x_url is the URL to monitor (example: http://my.site.com/folder)
# web_x_description is optional (if not set, define to URL)
# web_x_timeout is optional and overwrite the default timeout value
# web_x_rtt_warning is optional and defines the warning respond time in ms
#
web_1_url=https://blog.nicolargo.com
web_1_description=My Blog
web_1_rtt_warning=3000
web_2_url=https://github.com
web_3_url=http://www.google.fr
web_3_description=Google Fr

## 1.5.11 Disk I/O

Glances displays the disk I/O throughput. The unit is adapted dynamically.

You can display:

- bytes per second (default behavior / Bytes/s, KBytes/s, MBytes/s, etc)
- requests per second (using –diskio-iops option or B hotkey)

There is no alert on this information.

It’s possible to define:

- a list of disks to hide
• aliases for disk name

under the [diskio] section in the configuration file.

For example, if you want to hide the loopback disks (loop0, loop1, ...) and the specific sda5 partition:

```
[diskio]
hide=sda5,loop.*
```

### 1.5.12 File System

Glances displays the used and total file system disk space. The unit is adapted dynamically.

Alerts are set for used disk space.

Legend:

<table>
<thead>
<tr>
<th>Disk usage</th>
<th>Status</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;50%</td>
<td>OK</td>
</tr>
<tr>
<td>&gt;50%</td>
<td>CAREFUL</td>
</tr>
<tr>
<td>&gt;70%</td>
<td>WARNING</td>
</tr>
<tr>
<td>&gt;90%</td>
<td>CRITICAL</td>
</tr>
</tbody>
</table>

**Note:** Limit values can be overwritten in the configuration file under the [filesystem] section.

By default, the plugin only displays physical devices (hard disks, USB keys). To allow other file system types, you have to enable them in the configuration file. For example, if you want to allow the zfs file system:

```
[fs]
allow=zfs
```

Also, you can hide mount points as well (in the following /boot):

```
[fs]
hide=/boot.*
```

**RAID**

*Availability: Linux*

Thanks to the pymdstat library, if a RAID controller is detected on your system, its status will be displayed as well:
1.5.13 Folders

The folders plugin allows user, through the configuration file, to monitor size of a predefined folders list.

If the size cannot be computed, a '?' (non-existing folder) or a '! ' (permission denied) is displayed.

Each item is defined by:

- **path**: absolute path to monitor (mandatory)
- **careful**: optional careful threshold (in MB)
- **warning**: optional warning threshold (in MB)
- **critical**: optional critical threshold (in MB)
- **refresh**: interval in second between two refresh (default is 30 seconds)

Up to 10 items can be defined.

For example, if you want to monitor the /tmp folder every minute, the following definition should do the job:

```
[folders]
folder_1_path=/tmp
folder_1_careful=2500
folder_1_warning=3000
folder_1_critical=3500
folder_1_refresh=60
```

In client/server mode, the list is defined on the server side.

**Warning**: Do NOT define folders containing lot of files and subfolders or use an huge refresh time…

1.5.14 IRQ

*Availability: Linux*

This plugin is disable by default, please use the –enable-irq option to enable it.
Glances displays the top 5 interrupts rate.

This plugin is only available on GNU/Linux (stats are grabbed from the `/proc/interrupts` file).

**Note:** `/proc/interrupts` file doesn’t exist inside OpenVZ containers.

How to read the information:

- The first column is the IRQ number / name
- The second column says how many times the CPU has been interrupted during the last second

### 1.5.15 Sensors

**Availability:** Linux

Glances can displays the sensors information using `psutil` and/or `hddtemp`.

There is no alert on this information.

**Note:** Limit values and sensors alias names can be defined in the configuration file under the `[sensors]` section.

**Note:** This plugin is disabled by default in the configuration file.

### 1.5.16 Processes List

Compact view:
The process view consists of 3 parts:

- Processes summary
- Monitored processes list (optional)
- Processes list
The processes summary line displays:

- Tasks number (total number of processes)
- Threads number
- Running tasks number
- Sleeping tasks number
- Other tasks number (not running or sleeping)
- Sort key

By default, or if you hit the a key, the processes list is automatically sorted by:

- CPU: if there is no alert (default behavior)
- CPU: if a CPU or LOAD alert is detected
- MEM: if a memory alert is detected
- DISK I/O: if a CPU iowait alert is detected

The number of processes in the list is adapted to the screen size.
### Columns display

<table>
<thead>
<tr>
<th>Column</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU%</td>
<td>% of CPU used by the process. If Irix/Solaris mode is off, the value is divided by logical core number.</td>
</tr>
<tr>
<td>MEM%</td>
<td>% of MEM used by the process.</td>
</tr>
<tr>
<td>VIRT</td>
<td>Virtual Memory Size. The total amount of virtual memory used by the process. It includes all code, data and shared libraries plus pages that have been swapped out and pages that have been mapped but not used.</td>
</tr>
<tr>
<td>RES</td>
<td>Resident Memory Size. The non-swapped physical memory a process is using (what’s currently in the physical memory).</td>
</tr>
<tr>
<td>PID</td>
<td>Process ID</td>
</tr>
<tr>
<td>USER</td>
<td>User ID</td>
</tr>
<tr>
<td>THR</td>
<td>Threads number of the process</td>
</tr>
<tr>
<td>TIME+</td>
<td>Cumulative CPU time used by the process</td>
</tr>
<tr>
<td>NI</td>
<td>Nice level of the process</td>
</tr>
<tr>
<td>S</td>
<td>Process status. The status of the process:</td>
</tr>
<tr>
<td></td>
<td>• R: running or runnable (on run queue)</td>
</tr>
<tr>
<td></td>
<td>• S: interruptible sleep (waiting for an event)</td>
</tr>
<tr>
<td></td>
<td>• D: uninterruptible sleep (usually I/O)</td>
</tr>
<tr>
<td></td>
<td>• Z: defunct (“zombie”) process</td>
</tr>
<tr>
<td></td>
<td>• T: traced/stopped by job control signal</td>
</tr>
<tr>
<td></td>
<td>• X: dead (should never be seen)</td>
</tr>
<tr>
<td>R/s</td>
<td>Per process I/O read rate in B/s</td>
</tr>
<tr>
<td>W/s</td>
<td>Per process I/O write rate in B/s</td>
</tr>
<tr>
<td>COMMAND</td>
<td>Process command line or command name. User can switch to the process name by pressing on the '/' key</td>
</tr>
</tbody>
</table>

### Process filtering

It’s possible to filter the processes list using the ENTER key.

Filter syntax is the following (examples):

- **python**: Filter processes name or command line starting with `python` (regexp)
- **.*python.***: Filter processes name or command line containing `python` (regexp)
- **username:nicolargo**: Processes of nicolargo user (key:regexp)
- **cmdline:\/usr\/bin.***: Processes starting by `/usr/bin`
Extended info

In standalone mode, additional information are provided for the top process:

<table>
<thead>
<tr>
<th>CPU affinity</th>
<th>Number of cores used by the process</th>
</tr>
</thead>
<tbody>
<tr>
<td>Memory info</td>
<td>Extended memory information about the process For example, on Linux: swap, shared, text, lib, data and dirty</td>
</tr>
<tr>
<td>Open</td>
<td>The number of threads, files and network sessions (TCP and UDP) used by the process</td>
</tr>
<tr>
<td>IO nice</td>
<td>The process I/O niceness (priority)</td>
</tr>
</tbody>
</table>

The extended stats feature can be enabled using the `--enable-process-extended` option (command line) or the e key (curses interface).

**Note:** Limit for CPU and MEM percent values can be overwritten in the configuration file under the [processlist] section. It is also possible to define limit for Nice values (comma separated list). For example: `nice_warning=-20,-19,-18`

### 1.5.17 Monitored Processes List

**Warning:** The monitored processes list feature has been removed. Use the new Application Monitoring Process (AMP) instead.

### 1.5.18 Applications Monitoring Process

Thanks to Glances and its AMP module, you can add specific monitoring to running processes. AMPs are defined in the Glances [configuration file](http://glances.readthedocs.io/en/stable/config.html).

You can disable AMP using the `--disable-amps` option or pressing the A key.

#### Simple AMP

For example, a simple AMP that monitor the CPU/MEM of all Python processes can be defined as follows:

```
[amp_python]
enable=true
regex=.*python.*
refresh=3
```

Every 3 seconds (refresh) and if the enable key is true, Glances will filter the running processes list thanks to the `.*python.*` regular expression (regex).

The default behavior for an AMP is to display the number of matching processes, CPU and MEM:
You can also define the minimum (`countmin`) and/or maximum (`countmax`) process number. For example:

```
[amp_python]
enable=true
regex=.*python.*
refresh=3
countmin=1
countmax=2
```

With this configuration, if the number of running Python scripts is higher than 2, then the AMP is displayed with a purple color (red if less than `countmin`):

![Python AMP example](image)

### User defined AMP

If you need to execute a specific command line, you can use the `command` option. For example, if you want to display the Dropbox process status, you can define the following section in the Glances configuration file:

```
[amp_dropbox]
# Use the default AMP (no dedicated AMP Python script)
enable=true
regex=.*dropbox.*
refresh=3
one_line=false
command=dropbox status
countmin=1
```

The `dropbox status` command line will be executed and displayed in the Glances UI:

![Dropbox AMP example](image)

You can force Glances to display the result in one line setting `one_line` to true.

### Embedded AMP

Glances provides some specific AMP scripts (replacing the `command` line). You can write your own AMP script to fill your needs. AMP scripts are located in the `amps` folder and should be named `glances_*`.py. An AMP script define an Amp class (`GlancesAmp`) with a mandatory `update` method. The update method call the `set_result` method to set the AMP return string. The return string is a string with one or more line (n between lines). To enable it, the configuration file section should be named `[amp_*]`.

For example, if you want to enable the Nginx AMP, the following definition should do the job (Nginx AMP is provided by the Glances team as an example):

```
[amp_nginx]
enable=true
regex=/usr/sbin/nginx
refresh=60
one_line=false
status_url=http://localhost/nginx_status
```

1.5. Anatomy Of The Application
In client/server mode, the AMP list is defined on the server side.

### 1.5.19 events

Events list is displayed in the bottom of the screen if and only if:

- at least one **WARNING** or **CRITICAL** alert was occurred
- space is available in the bottom of the console/terminal

Each event message displays the following information:

1. start datetime
2. duration if alert is terminated or **ongoing** if the alert is still in progress
3. alert name
4. \{min,avg,max\} values or number of running processes for monitored processes list alerts

### 1.5.20 Docker

If you use **Docker**, Glances can help you to monitor your containers. Glances uses the Docker API through the **docker-py** library.

You can install this dependency using:

```bash
pip install glances[docker]
```

It is possible to define limits and actions from the configuration file under the **[docker]** section:
Glances Documentation, Release 3.1.3

You can use all the variables ({{foo}}) available in the Docker plugin.

1.5.21 Actions

Glances can trigger actions on events.

By action, we mean all shell command line. For example, if you want to execute the foo.py script if the last 5 minutes load are critical then add the _action line to the Glances configuration file:

```
[load]
critical=5.0
critical_action=python /path/to/foo.py
```

All the stats are available in the command line through the use of the {{mustache}} syntax. Another example would be to create a log file containing used vs total disk space if a space trigger warning is reached:

```
[fs]
warning=70
warning_action=echo {{mnt_point}} {{used}}/{{size}} > /tmp/fs.alert
```

Note: You can use all the stats for the current plugin. See https://github.com/nicolargo/glances/wiki/The-Glances-2.x-API-How-to for the stats list.

It is also possible to repeat action until the end of the alert. Keep in mind that the command line is executed every refresh time so use with caution:

```
[load]
critical=5.0
critical_action_repeat=/home/myhome/bin/bipper.sh
```

1.6 Gateway To Other Services

Glances can exports stats to a CSV file. Also, it can act as a gateway to providing stats to multiple services (see list below).
1.6.1 CSV

It's possible to export stats to a CSV file.

```
$ glances --export csv --export-csv-file /tmp/glances.csv
```

CSV file description:
- first line: Stats description (header)
- others lines: Stats (data)

By default, data will be append any existing CSV file.

If the header did not match with a previous one, an error is logged.

The --export-csv-overwrite tag should be used if you want to delete the existing CSV file when Glances starts.

It is possible to remove some exported data using the --disable-plugin tag:

```
$ glances --export csv --export-csv-file /tmp/glances.csv --disable-plugin load,swap
```

1.6.2 Cassandra

You can export statistics to a Cassandra or Scylla server. The connection should be defined in the Glances configuration file as following:

```
[cassandra]
host=localhost
port=9042
protocol_version=3
keyspace=glances
replication_factor=2
table=localhost
```

and run Glances with:

```
$ glances --export cassandra
```

The data model is the following:

```
CREATE TABLE <table> (plugin text, time timeuuid, stat map<text,float>, PRIMARY KEY (plugin, time))
```

Only numerical stats are stored in the Cassandra table. All the stats are converted to float. If a stat cannot be converted to float, it is not stored in the database.

1.6.3 CouchDB

You can export statistics to a CouchDB server. The connection should be defined in the Glances configuration file as following:

```
[couchdb]
host=localhost
port=5984
user=root
password=root
db=glances
```

and run Glances with:

```
$ glances --export couchdb
```

Documents are stored in native JSON format. Glances adds "type" and "time" entries:

- **type**: plugin name
- **time**: timestamp (format: "2016-09-24T16:39:08.524828Z")

Example of Couch Document for the load stats:

```
{
    "_id": "36cbbad81453c53ef08804cb2612d5b6",
    "_rev": "1-382400899bec5615cabb99aa34df49fb",
    "min15": 0.33,
    "time": "2016-09-24T16:39:08.524828Z",
    "min5": 0.4,
    "cpucore": 4,
    "load_warning": 1,
    "min1": 0.5,
    "history_size": 28800,
    "load_critical": 5,
    "type": "load",
    "load_careful": 0.7
}
```

You can view the result using the CouchDB utils URL: http://127.0.0.1:5984/_utils/database.html?glances.

### 1.6.4 Elasticsearch

You can export statistics to an Elasticsearch server. The connection should be defined in the Glances configuration file as following:

```
[elasticsearch]
host=localhost
port=9200
index=glances
```

and run Glances with:

```
$ glances --export elasticsearch
```

The stats are available through the elasticsearch API. For example, to get the CPU system stats:

```
$ curl http://172.17.0.2:9200/glances/cpu/system
{
    "_index": "glances",
    "_type": "cpu",
    "_id": "system",
    "_version": 28,
    "found": true,
    "_source": {
        "timestamp": "2016-02-04T14:11:02.362232",
        "value": "2.2"
    }
}
```
1.6.5 Graph

You can generate dynamic graphs (SVG format) in a target folder. The generation starts every time the ‘g’ key is pressed in the CLI interface.

```
[graph]
# Configuration for the --export graph option
# Set the path where the graph (.svg files) will be created
# Can be overwrite by the --graph-path command line option
path=/tmp
# It is possible to generate the graphs automatically by setting the
# generate_every to a non zero value corresponding to the seconds between
# two generation. Set it to 0 to disable graph auto generation.
generate_every=60
# See followings configuration keys definitions in the Pygal lib documentation
width=800
height=600
style=DarkStyle
```

and run Glances with:

```
$ glances --export graph --export-graph-path /tmp
```

Example of output (load graph)

1.6.6 InfluxDB

You can export statistics to an InfluxDB server (time series server). The connection should be defined in the Glances configuration file as following:

```
[influxdb]
host=localhost
port=8086
protocol=http
user=root
password=root
db=glances
tags=foo:bar,spam:eggs
```

and run Glances with:

```
$ glances --export influxdb
```

Glances generates a lot of columns, e.g., if you have many running Docker containers, so you should use the tsm1 engine in the InfluxDB configuration file (no limit on columns number).

Note: if you want to use SSL, please set ‘protocol=https’.

Grafana

For Grafana users, Glances provides a dedicated dashboard.
To use it, just import the file in your Grafana web interface.

1.6.7 JSON

It’s possible to export stats to a JSON file.

```sh
$ glances --export json --export-json-file json /tmp/glances.json
```

1.6.8 Kafka

You can export statistics to a Kafka server. The connection should be defined in the Glances configuration file as following:

```conf
[kafka]
host=localhost
port=9092
topic=glances
#compression=gzip
```

Note: you can enable the compression but it consume CPU on your host.

and run Glances with:

1.6. Gateway To Other Services
Stats are sent in native JSON format to the topic:

- **key**: plugin name
- **value**: JSON dict

Example of record for the memory plugin:

```
ConsumerRecord(topic=u'glances', partition=0, offset=1305, timestamp=1490460592248,
→timestamp_type=0, key='mem', value=u'{"available": 2094710784, "used": 5777428480,
→"cached": 2513543168, "mem_careful": 50.0, "percent": 73.4, "free": 2094710784,
→"mem_critical": 90.0, "inactive": 2361626624, "shared": 475504640, "history_size":
→28800.0, "mem_warning": 70.0, "total": 7872139264, "active": 4834361344, "buffers":
→160112640}', checksum=214895201, serialized_key_size=3, serialized_value_size=303)
```

Python code example to consume Kafka Glances plugin:

```
from kafka import KafkaConsumer
import json

consumer = KafkaConsumer('glances', value_deserializer=json.loads)
for s in consumer:
    print(s)
```

### 1.6.9 MQTT

You can export statistics to an MQTT server. The connection should be defined in the Glances configuration file as following:

```
[mqtt]
host=localhost
port=883
user=glances
password=glances
topic=glances
```

and run Glances with:

```
$ glances --export mqtt
```

### 1.6.10 OpenTSDB

You can export statistics to an OpenTSDB server (time series server). The connection should be defined in the Glances configuration file as following:

```
[opentsdb]
host=localhost
port=4242
prefix=glances
tags=foo:bar, spam:eggs
```

and run Glances with:
### 1.6.11 Prometheus

You can export statistics to a Prometheus server through an exporter. When the `--export-prometheus` is used, Glances creates a Prometheus exporter listening on `<host:port>` (define in the Glances configuration file).

```ini
[prometheus]
host=localhost
port=9091
prefix=glances
labels=src:glances
```

Note: you can use dynamic fields for the label (ex: `labels=system:uname -s`) and run Glances with:

```bash
$ glances --export prometheus
```

You can check that Glances exports the stats using this URL: [http://localhost:9091](http://localhost:9091)

In order to store the metrics in a Prometheus server, you should add this exporter to your Prometheus server configuration with the following lines (in the prometheus.yml configuration file):
scrape_configs:
- job_name: 'glances_exporter'
  scrape_interval: 5s
  static_configs:
  - targets: ['localhost:9091']

1.6.12 RabbitMQ

You can export statistics to a RabbitMQ server (AMQP Broker). The connection should be defined in the Glances configuration file as following:

[rabbitmq]
host=localhost
port=5672
user=glances
password=glances
queue=glances_queue

and run Glances with:

$ glances --export rabbitmq

1.6.13 RESTful

You can export statistics to a RESTful JSON server. All the available stats will be exported in one big (~15 KB) POST request to the RESTful endpoint.

The RESTful endpoint should be defined in the Glances configuration file as following:

[restful]
# Configuration for the --export-restful option
# Example, export to http://localhost:6789/
host=localhost
port=6789

(continues on next page)
URL Syntax:

```
http://localhost:6789/
```

and run Glances with:

```
$ glances --export restful
```

Glances will generate stats as a big JSON dictionary (see example here).

### 1.6.14 Riemann

You can export statistics to a Riemann server (using TCP protocol). The connection should be defined in the Glances configuration file as following:

```
[riemann]
host=localhost
port=5555
```

and run Glances with:

```
$ glances --export riemann
```

### 1.6.15 StatsD

You can export statistics to a StatsD server (welcome to Graphite!). The connection should be defined in the Glances configuration file as following:

```
[statsd]
host=localhost
port=8125
prefix=glances
```

**Note:** The `prefix` is optional (glances by default)

and run Glances with:

```
$ glances --export statsd
```

Glances will generate stats as:
'glances.cpu.user': 12.5,
'glances.cpu.total': 14.9,
'glances.load.cpucore': 4,
'glances.load.min1': 0.19,
...

## 1.6.16 ZeroMQ

You can export statistics to a ZeroMQ server.

The connection should be defined in the Glances configuration file as following:

```
[zeromq]
host=127.0.0.1
port=5678
prefix=G
```

Glances **envelopes** the stats before publishing it. The message is composed of three frames:

1. the prefix configured in the [zeromq] section (as STRING)
2. the Glances plugin name (as STRING)
3. the Glances plugin stats (as JSON)

Run Glances with:

```
$ glances --export zeromq
```

Following is a simple Python client to subscribe to the Glances stats:

```python
# -*- coding: utf-8 -*-
#
# ZeroMQ subscriber for Glances
#

import json
import zmq

context = zmq.Context()

subscriber = context.socket(zmq.SUB)
subscriber.setsockopt(zmq.SUBSCRIBE, 'G')
subscriber.connect("tcp://127.0.0.1:5678")

while True:
    _, plugin, data_raw = subscriber.recv_multipart()
    data = json.loads(data_raw)
    print('{} => {}'.format(plugin, data))

subscriber.close()
context.term()
```

## 1.7 API Documentation

Glances provides an [XML-RPC server](#) and a [RESTful-JSON API](#) which can be used by other clients.
API documentation is available at:


### 1.8 Docker

Glances can be installed through Docker, allowing you to run it without installing all the python dependencies directly on your system. Once you have [docker installed](https://docs.docker.com/install/), you can

Get the Glances container:

```
docker pull nicolargo/glances
```

Run the container in **console mode**:

```
docker run --rm -v /var/run/docker.sock:/var/run/docker.sock:ro --pid host --network
-\rightarrow host -it docker.io/nicolargo/glances
```

Additionally, if you want to use your own glances.conf file, you can create your own Dockerfile:

```
FROM nicolargo/glances
COPY glances.conf /glances/conf/glances.conf
CMD python -m glances -C /glances/conf/glances.conf $GLANCES_OPT
```

Alternatively, you can specify something along the same lines with docker run options:

```
docker run -v `pwd`/glances.conf:/glances/conf/glances.conf -v /var/run/docker.sock:/var/run/docker.sock:ro --pid host -it docker.io/nicolargo/glances
```

Where ``pwd`/glances.conf` is a local directory containing your glances.conf file.

Run the container in **Web server mode** (notice the `GLANCES_OPT` environment variable setting parameters for the glances startup command):

```
docker run -d --restart="always" -p 61208-61209:61208-61209 -e GLANCES_OPT="-w" -v /var/run/docker.sock:/var/run/docker.sock:ro --pid host docker.io/nicolargo/glances
```

Note: if you want to see the network interface stats within the container, add --net=host --privileged

You can also include Glances container in your own `docker-compose.yml`. Here’s a realistic example including a “traefik” reverse proxy serving an “whoami” app container plus a Glances container, providing a simple and efficient monitoring webui.

```
version: '3'

services:
  reverse-proxy:
    image: traefik:alpine
    command: --api --docker
    ports:
      - "80:80"
      - "8080:8080"
    volumes:
      - /var/run/docker.sock:/var/run/docker.sock
```

(continues on next page)
whoami:
  image: emilevauge/whoami
  labels:
    - "traefik.frontend.rule=Host:whoami.docker.localhost"

monitoring:
  image: nicolargo/glances:latest-alpine
  restart: always
  pid: host
  volumes:
    - /var/run/docker.sock:/var/run/docker.sock
  environment:
    - "GLANCES_OPT=-w"
  labels:
    - "traefik.port=61208"
    - "traefik.frontend.rule=Host:glances.docker.localhost"

1.9 F.A.Q

Any encoding issue ?
Try to run Glances with the following command line:

    LANG=en_US.UTF-8 LC_ALL= glances

1.10 Support

To post a question about Glances use cases, please post it to the official Q&A forum.
To report a bug or a feature request use the GitHub issue tracker.
Feel free to contribute!
Symbols
-`-browser` command line option, 10
-`-cached-time` [CACHE_TIME] command line option, 11
-`-disable-autodiscover` command line option, 10
-`-disable-bg` command line option, 10
-`-disable-bold` command line option, 10
-`-disable-check-update` command line option, 11
-`-disable-plugin` [PLUGIN] command line option, 9
-`-disable-process` command line option, 9
-`-disable-webui` command line option, 9
-`-diskio-iops` command line option, 11
-`-diskio-show-ramfs` command line option, 11
-`-enable-history` command line option, 10
-`-enable-plugin` [PLUGIN] command line option, 9
-`-enable-process-extended` command line option, 10
-`-export` [EXPORT] command line option, 9
-`-export-csv-file` [EXPORT_CSV_FILE] command line option, 9
-`-export-json-file` [EXPORT_JSON_FILE] command line option, 9
-`-fahrenheit` command line option, 11
-`-fs-free-space` command line option, 11
-`-hide-kernel-threads` command line option, 11
-`-light`, `-enable-light` command line option, 9
-`-modules-list` command line option, 9
-`-password` command line option, 10
-`-process-short-name` command line option, 11
-`-snmp-auth` [SNMP_AUTH] command line option, 10
-`-snmp-community` [SNMP_COMMUNITY] command line option, 10
-`-snmp-force` command line option, 10
-`-snmp-port` [SNMP_PORT] command line option, 10
-`-snmp-user` [SNMP_USER] command line option, 10
-`-snmp-version` [SNMP_VERSION] command line option, 10
-`-stdout` [PLUGINS_STATS] command line option, 9
-`-theme-white` command line option, 11
-`-username` command line option, 10
-`-0`, `-disable-irix` command line option, 9
-`-1`, `-percpu` command line option, 9
-`-2`, `-disable-left-sidebar` command line option, 9
-`-3`, `-disable-quicklook` command line option, 9
-`-4`, `-full-quicklook` command line option, 9
-`-5`, `-disable-top` command line option, 9
-6, -meanpu
  command line option, 10
-B BIND_ADDRESS, -bind BIND_ADDRESS
  command line option, 10
-C CONF_FILE, -config CONF_FILE
  command line option, 9
-V, -version
  command line option, 9
-b, -byte
  command line option, 11
-c CLIENT, -client CLIENT
  command line option, 10
-d, -debug
  command line option, 9
-f PROCESS_FILTER, -process-filter
  PROCESS_FILTER
  command line option, 11
-h, -help
  command line option, 9
-p PORT, -port PORT
  command line option, 10
-q, -quiet
  command line option, 11
-s, -server
  command line option, 10
-t TIME, -time TIME
  command line option, 10
-w, -webserver
  command line option, 10

C
  command line option
  -browser, 10
  -cached-time CACHED_TIME, 11
  -disable-autodiscover, 10
  -disable-bg, 10
  -disable-bold, 10
  -disable-check-update, 11
  -disable-plugin PLUGIN, 9
  -disable-process, 9
  -disable-webui, 9
  -diskio-iops, 11
  -diskio-show-ramfs, 11
  -enable-history, 10
  -enable-plugin PLUGIN, 9
  -enable-process-extended, 10
  -export EXPORT, 9
  -export-csv-file EXPORT_CSV_FILE, 9
  -export-json-file EXPORT_JSON_FILE, 9
  -fahrenheit, 11
  -fs-free-space, 11
  -hide-kernel-threads, 11
  -light, -enable-light, 9
  -modules-list, 9
  -password, 10
  -process-short-name, 11
  -snmp-auth SNMP_AUTH, 10
  -snmp-community SNMP_COMMUNITY, 10
  -snmp-force, 10
  -snmp-port SNMP_PORT, 10
  -snmp-user SNMP_USER, 10
  -snmp-version SNMP_VERSION, 10
  -stdout PLUGINS_STATS, 9
  -theme-white, 11
  -username, 10
  -0, -disable-irix, 9
  -1, -percpu, 9
  -2, -disable-left-sidebar, 9
  -3, -disable-quicklook, 9
  -4, -full-quicklook, 9
  -5, -disable-top, 9
  -6, -meanpu, 10
  -B BIND_ADDRESS, -bind
    BIND_ADDRESS, 10
  -C CONF_FILE, -config CONF_FILE, 9
  -V, -version, 9
  -b, -byte, 11
  -c CLIENT, -client CLIENT, 10
  -d, -debug, 9
  -f PROCESS_FILTER, -process-filter
    PROCESS_FILTER, 11
  -h, -help, 9
  -p PORT, -port PORT, 10
  -q, -quiet, 11
  -s, -server, 10
  -t TIME, -time TIME, 10
  -w, -webserver, 10
  open-web-browser, 11

O
  open-web-browser
  command line option, 11