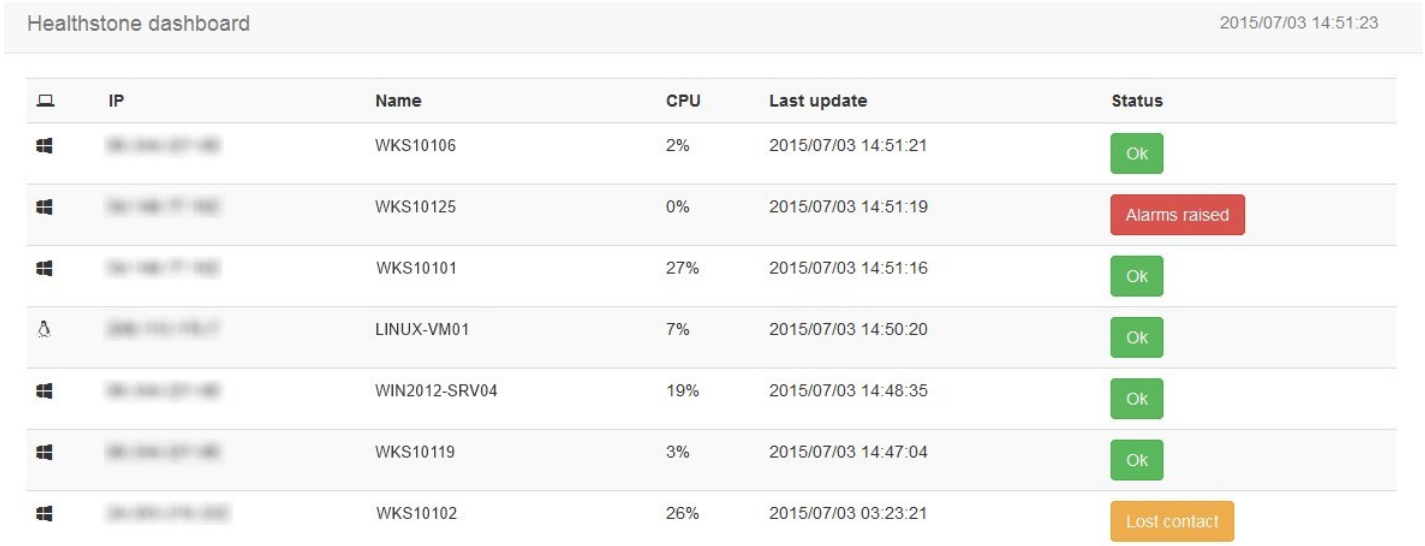

Healthstone Monitoring System

Patrick Lambert

http://healthstone.ca

v1.1.0

Contents

1 Introduction **2**

2 Windows client **2**

2.1 Installation 2

2.2 Troubleshooting 3

2.3 Configuration 3

3 Linux client **6**

3.1 Installation 6

3.2 Troubleshooting 6

3.3 Configuration 6

4 Dashboard **7**

4.1 Installation 7

4.2 Troubleshooting 8

4.3 Configuration 8

5 Conclusion **8**

6 License **9**

1 Introduction

Healthstone is an open source and lightweight system monitoring solution able to run dozens of customizable health checks. The client component runs as a Windows service or Linux cron job and can notify the Event Log, send emails, file a NodePoint ticket, run a custom program, or send a notification using Amazon SNS or Pushbullet in case any health check fails. It runs on any 64 bits Windows XP, 7, 8, 10, Server 2008 or Server 2012 system, or a Linux system. It can be used to monitor servers or the state of Windows desktops in any environment, along with other services such as IIS, SQL Server, etc. It also includes a server component which provides an optional dashboard against which the Healthstone clients can register.

A typical scenario would be to have a number of Windows desktops in a LAN configuration. For this, you can easily place the Windows client files on a central network location, and run the installation through Group Policy. You may also have some Linux and Windows servers spread through various datacenters and cloud locations, where you can manually install clients. Finally, you can select one server to run the dashboard, against which all of your clients will register, and where you can monitor the health of your whole network from a central location.

You may configure each client to have the ability to send you notifications through a number of means, or have them only report to the dashboard, and have your dashboard report to you if any of the clients raised alarms.

2 Windows client

The Healthstone Windows client is a small binary along with a configuration file. The binary runs as a Windows service and checks the current system for all configured health checks on a regular interval. This service is only a few KBs in size and takes a few MBs of system memory to run. On startup, it reads the configuration file from a standard location at `C:\Program Files\healthstone\healthstone.cfg` and follows those settings until stopped. Note that you can modify the location of this file by changing the Registry key `HKLM/SOFTWARE/Healthstone`. This can be useful if you wish to use a single config file for all your Windows systems, accessible on a mapped drive.

When the Windows service notices a health check failed (or even on success, should you configure it that way), a number of notifications can be sent. This can include email notifications, creating a NodePoint ticket, sending a PushBullet notification to your phone, or notifying a Healthstone dashboard. Note that this package is all you need to monitor a Windows system, but the dashboard component is a useful companion to run if you wish to keep track of multiple systems.

Along with the service file and configuration file, an installation batch script is provided, and this document.

2.1 Installation

The client component can run on any 64 bits Windows host and requires Microsoft .NET Framework 3.5 to be installed.

1. Download ans extract `healthstone-client-win64.zip` onto each host you want to monitor, or to a central accessible network location.
2. Edit `healthstone.cfg` and customize the checks you want to run.
3. Run `install.bat` as a local administrator on each host to copy the files and install the service.

The install file will go through several steps. First, it will copy the binary and configuration files to `%PROGRAMFILES%\healthstone`. Then, it will add the location of the configuration file to the Registry. Finally, it will create and start the Windows service.

2.2 Troubleshooting

You can look at the Event Log to see whether the service started correctly, and if any error occurred. At every interval, a new entry will be added with a result of the health checks. Note that all of the configuration file entries must be present, do not remove any line. If the service fails to start, an error message should be shown in the Event Log. A typical reason may be if the configuration file cannot be found or is corrupted. You can also try to manually run the binary file to see whether it launches. This can be useful to know if you're missing the .NET Framework, for example.

2.3 Configuration

Configuration for the Windows service is done in the `heaslthstone.cfg` file. Note that you will need to restart the service after any change. This file comes with sane default values, but you still should go over all the options to configure them depending on your needs.

- Interval (in seconds) to run checks. [number]
  `Interval: 300`
- Always raise an alarm, regardless of results. [true|false]
  `AlwaysRaise: false`
- Show full system information, not just failed checks. [true|false]
  `Verbose: false`
- Custom text to add at the end of all reports. [text]
  `CustomText:`
- Raise errors instead of warnings in the Event Log. [true|false]
  `RaiseErrors: false`
- Raise an alarm if a test query fails or is not supported on the system. [true|false]
  `RaiseQueryFailures: false`
- Log debug information about notifications, used to troubleshoot failed notifications. [true|false]
  `NotifyDebug: false`
- Address of a Healthstone dashboard. [url|false]
  `NotifyHealthstoneDashboard: false`
- File a NodePoint ticket. [true|false]
  `NotifyNodePoint: false`
- NodePoint server address. [url]
  `NotifyNodePointAddress: http://10.10.106.2/nodepoint`
- NodePoint API key. [text]
  `NotifyNodePointKey: XXXXXXXXXXXXX`
- NodePoint product id. [number]
  `NotifyNodePointProduct: 1`

- Send a Pushbullet notification. [true|false]
  `NotifyPushbullet: false`
- Pushbullet key. [text]
  `NotifyPushbulletKey: XXXXXXXXXXXXXX`
- Send mail notifications. [true|false]
  `NotifyEmail: false`
- SMTP server name. [hostname]
  `NotifyEmailServer: smtp.example.com`
- SMTP server port. [number]
  `NotifyEmailPort: 25`
- SMTP from address. [email]
  `NotifyEmailFrom: noreply@example.com`
- SMTP to addresses. Separate email addresses with spaces. [email email email]
  `NotifyEmailTo: tyrion.lannister@example.com`
- Run custom program. Report content will be passed as command line arguments. [program|false]
  `NotifyProgram: false`
- Send Amazon SNS notification to a topic ARN. [topicarn|false]
  `NotifyAWSTopic: false`
- AWS region where the SNS topic is defined. [region]
  `NotifyAWSRegion: us-west-2`
- AWS IAM access key ID with access to publish to that topic. [key]
  `NotifyAWSKey: XXXXXXXXX`
- AWS IAM secret with access to publish to that topic. [secret]
  `NotifyAWSSecret: XXXXXXXXXXXXXXXXXXXXX`
- Check whether Data Execution Prevention is set. [true|false]
  `CheckDEP: true`
- Check for a specific time zone. Number, in hours, the system is offset from Greenwich Mean Time (GMT). Example: PST is -8. [number|false]
  `CheckTimeZone: false`
- Check for a specific code page. Example: Hebrew is 1255. [number|false]
  `CheckCodePage: false`
- Check if free physical memory is lower than x megabytes. [number|false]
  `CheckFreeMemory: 200`
- Check whether the system rebooted in the past x hours [number|false]
  `CheckLastBoot: false`
- Check if the system locale is set to a specific value. Example: EN-US is 0409. [number|false]
  `CheckLocale: false`
- Check overall system status as reported by the Windows subsystem. [true|false]
  `CheckSystemStatus: true`
- Check if there are more than x processes running on the system. [number|false]
  `CheckProcessCount: 200`

- Check if a list of processes are running. Use the binary names, separated with spaces. [process process process|false]
  `CheckProcesses: false`
- Check if an Anti Virus product is installed. Only works on workstations, not servers. [true|false]
  `CheckAntiVirus: true`
- Check if Anti Virus is disabled or out of date. Only works on workstations, not servers. [true|false]
  `CheckAntiVirusState: true`
- Check if the Public firewall is on. [true|false]
  `CheckFirewallPublic: true`
- Check if the Private firewall is on. [true|false]
  `CheckFirewallPrivate: true`
- Check if the Domain firewall is on. [true|false]
  `CheckFirewallDomain: true`
- Check whether the computer booted normally, or in safe mode. [true|false]
  `CheckBootState: true`
- Check whether the machine's DNS hostname is set to a specific value. [text|false]
  `CheckHostName: false`
- Check if the machine is part of a specific domain or workgroup name. [text|false]
  `CheckDomainName: false`
- Check if the CPU temperature is above x degrees Celsius. Does not work with all systems. [number|false]
  `CheckTemperature: false`
- Check if the CPU load is above x%. [number|false]
  `CheckCpuLoad: 90`
- Check if a specific IP address is reachable using ICMP. [ip|false]
  `CheckNetConnectivity: false`
- Check if the latency with the address in CheckNetConnectivity is above x ms. [number|false]
  `CheckNetLatency: 500`
- Check whether a HTTP URL is working. Will fail if the host is down, page missing, unauthorized, internal server error, etc. [url|false]
  `CheckNetHttp: false`
- Check if an ODBC database connection can be made. Example: Driver=SQL Server; Server=myServerAddress; Database=myDataBase; Uid=myUsername; Pwd=myPassword; [connection string|false]
  `CheckODBC: false`
- Check whether Windows Updates are set to install automatically. [true|false]
  `CheckWUEnabled: true`
- Check if Windows Updates were installed successfully less than x hours ago. [number|false]
  `CheckWULast: false`
- Check if specific Windows Updates are installed. List the Microsoft knowledge base (KB123456) IDs separated with spaces. [kb kb kb|false]
  `CheckWUHotFixes: false`
- Check if any drive has less than x megabytes of space free. [number|false]
  `CheckDiskSpace: 500`

- Check if custom TCP ports are closed on the local system. Separate ports with spaces. [number number number|false]
  `CheckCustomPort: false`
- Check if the time is in sync (<5mins) with this NTP server. [server|false]
  `CheckTime: false`

3 Linux client

The Linux client is a smaller and less fully featured client compared with the Windows client, but should run on a wider array of systems. It supports basic system health monitoring such as whether the host is up, CPU usage, along with a number of notification options.

Like the Windows version, it can be run by itself, configured to notify you when a health check fails. Or, you may pair it with a Healthstone dashboard to have centralized monitoring.

3.1 Installation

The client component can run on any Linux host and requires Python 3.x to be installed.

1. Download and extract `healthstone-client-linux.zip` onto each host you want to monitor, or to a central accessible network location.
2. Edit `healthstone.py` and customize the checks you want to run.
3. Run `sudo ./install.sh` to install the script on the local system.

The installation script will copy the Python file to `/usr/bin/healthstone.py` and add a crontab entry. This assumes that you can run it as a root. Note that Healthstone can run as a regular user as well. Simply edit the script to copy it to a location you have access to instead. The default crontab is also set to 5 minutes. Modify that time if you're going to change the *Interval* value in the configuration.

3.2 Troubleshooting

The install script defaults to adding a 5 minutes crontab for running Healthstone. If you modify the interval value in `healthstone.py` you should modify that value in the install script as well. You should remove the old crontab entry using `crontab -e` if you run the install script more than once. Note that being told that *There is no root crontab* is not an error. Finally, if your Python 3.x installation is not in your PATH or at `/usr/bin/python3` then you may have to edit the crontab.

3.3 Configuration

Configuration for the Linux client is done at the top of the `healthstone.py` file.

- Interval (in seconds) configured in your crontab between runs [number]
  `Interval = 300`

- Check for acceptable CPU threshold [number|False]
  `CheckCPU = 90`
- Check if a specific process is running [process name|False]
  `CheckProcess = False`
- Check if used disk space is above x percent [number|False]
  `CheckDiskSpace = 90`
- Notify a Healthstone dashboard [url|False]
  `NotifyDashboardURL = "http://localhost/healthstone"`
- Write alarms to a log file [filename|False]
  `NotifyFile = False`
- Run a custom shell command when alarms fail [command|False]
  `NotifyProgram = False`
- Send a Pushbullet notification [API key|False]
  `NotifyPushbullet = False`

4 Dashboard

The dashboard is a Python script that runs on an IIS or Apache web server and awaits connections from clients. Once a client connects, it stores the data it receives, and optionally sends a notification if the client reported health check failures or stops checking in. You can connect to the dashboard web site and login using the access code configured inside of the script, then view a table of all the clients that reported in. By clicking on the status icon, you can also view more details about a specific host, such as a graph of past CPU utilization, a history of alarms and lost contact, along with the last check in time.

The dashboard also distinguishes between Linux and Windows hosts, and tracks hosts based on the hostname, not the IP address. This means a single host will keep its entry even if its IP address changes.

4.1 Installation

The server component runs the dashboard against which Healthstone clients can register. It is not mandatory, and Healthstone clients can be run without a dashboard. It requires Python 3.x and runs on IIS or Apache with CGI and ISAPI support enabled.

- Download `healthstone-server.zip` onto your server and unzip it where you want the files to live, for example `C:\healthstone`.
- Edit `dashboard.py` and configure the access code and notification settings you want.
- Run `setup.bat` as a local administrator on Windows, or create a virtual site linking to the `www` folder on Linux, making sure the script has write access to `../db`.

Since the dashboard component runs on both Windows and Linux systems, the installation process may vary based on your particular scenario. On Windows, the `setup.bat` script should handle most default configurations. It will create a local user to run under and add a virtual server under IIS. On Linux, the script should be placed in a publicly available location. A `.htaccess` file is provided to configure an Apache web server. All data received from clients is kept in a SQLite database under the `..\db` folder.

4.2 Troubleshooting

If you get an error loading the site, make sure the rewrite engine is turned on in your web server configuration, so the web configuration file is read. You may need to edit the `web.config` (Windows) or `.htaccess` (Linux) files manually depending on your situation. If your Python 3.x installation path is different, you may need to edit those files, along with the `dashboard.py` file. If you get a 404 error on IIS, make sure you allow the script in *ISAPI and CGI restrictions* inside of IIS Manager.

4.3 Configuration

The dashboard configuration is located at the top of the `dashboard.py` file. Note that dashboard notifications are triggered if any of the clients that check in fail a health check, or stop checking in.

- Access code to access the dashboard [string]
  `AccessCode = "1234"`
- Send notifications for systems that lose contacts [True|False]
  `NotifyOnLostContact = True`
- Send notifications for systems that raise alarms [True|False]
  `NotifyOnAlarms = True`
- Send a Pushbullet notification using this API key [API key|False]
  `NotifyPushbullet = False`
- Create a NodePoint ticket using this URL, API key, product and release numbers [url|False]
  ```
  NotifyNodePointURL = False
  NotifyNodePointKey = "XXXXXXX"
  NotifyNodePointProduct = "1"
  NotifyNodePointRelease = "1.0"
  ```
- Send an email notification using this SMTP server, To address, and From address [smtp server|False]
  ```
  NotifySMTPServer = False
  NotifySMTPFrom = "me@example.com"
  NotifySMTPTo = "you@example.com"
  ```

5 Conclusion

Healthstone is provided as a free open source solution. The source code is available to be viewed and modified from https://github.com/dendory/Healthstone. Out of the box, it is able to fully monitor a network of machines and report health status through a number of means. If you need more features, you can easily add them to the code.

Should you have questions or comments, feel free to contact the author at `dendory@live.ca`.

6 License

The MIT License (MIT)

Copyright © 2014-2015 Patrick Lambert

Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY, WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.