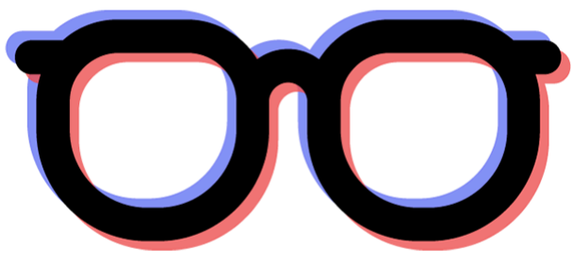
mas… 9 Branches 46 Tags Go to file Go to file Code

AnalogJ Merge pull request #725 from pabsi/706-add-wait-time-between-checks-f…

affe05e · last month 817 Commits

.github	fixing github actions.	3 months ago
collector	Change to time.Seconds	last month
docker	fix amd64 s6_arch.	8 months ago
docs	Update INSTALL_HUB_SPOKE.md	5 months ago
rootfs/etc	fixing cron in #602	8 months ago
webapp	Adjust null input response, and tests	5 months ago
.dockerignore	trying to fix docker image builds (take …	2 years ago
.gitattributes	Update .gitattributes	4 years ago
.gitignore	Fix some development issues	11 months ago
CONTRIBUTING.md	regenerate package-lock with angular …	last year
LICENSE	add license and logo link.	4 years ago
Makefile	simplify docker image build	last year
README.md	Merge pull request #529 from KaeTuu…	11 months ago
REFERENCES.md	init	4 years ago
example.collector.yaml	Add a wait between disks checks	last month
example.scrutiny.yaml	fix Shoutrrr discord notification url stru…	7 months ago
go.mod	reverted accidental bump of spf13/vipe…	4 months ago
go.sum	reverted accidental bump of spf13/vipe…	4 months ago
packagr.yml	fix packagr config.	2 years ago

README MIT license



scrutiny

CI no status codecov 31% license MIT godoc reference go report A release v0.8.1

WebUI for smartd S.M.A.R.T monitoring

> NOTE: Scrutiny is a Work-in-Progress and still has some rough edges.

About

Hard Drive S.M.A.R.T Monitoring, Historical Trends & Real World Failure Thresholds

Readme
MIT license
Activity
5.5k stars
33 watching
175 forks
Report repository

Releases 40

v0.8.1 Latest
on Apr 8

+ 39 releases

Sponsor this project

AnalogJ Jason Kulatunga

https://paypal.me/analogj/usd10

Learn more about GitHub Sponsors

Packages 1

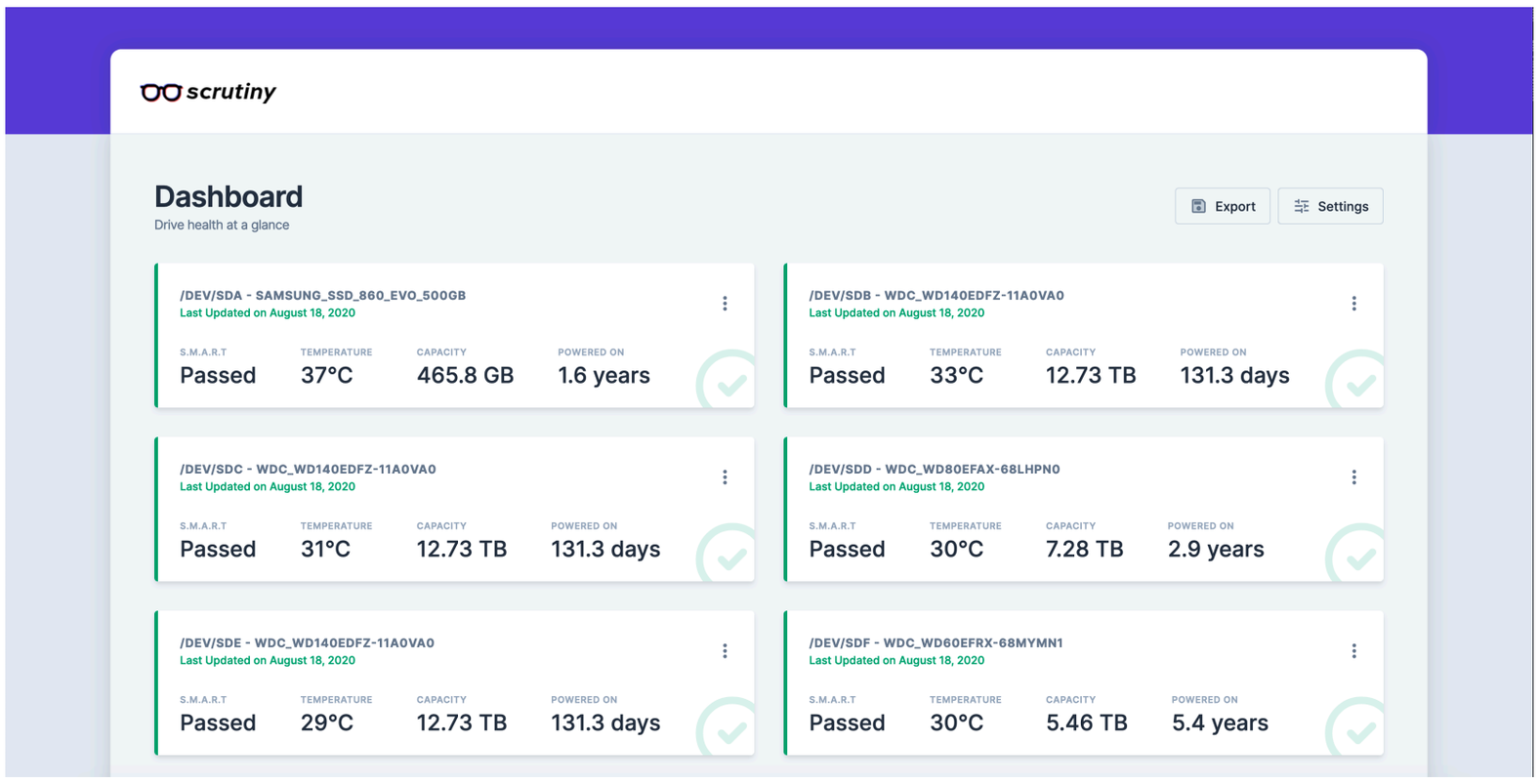
scrutiny

Contributors 37

+ 23 contributors

Languages

Go 79.2% **HTML** 18.7%
Other 2.1%



Introduction

If you run a server with more than a couple of hard drives, you're probably already familiar with S.M.A.R.T and the `smartd` daemon. If not, it's an incredible open source project described as the following:

> smartd is a daemon that monitors the Self-Monitoring, Analysis and Reporting Technology (SMART) system built into many ATA, IDE and SCSI-3 hard drives. The purpose of SMART is to monitor the reliability of the hard drive and predict drive failures, and to carry out different types of drive self-tests.

These S.M.A.R.T hard drive self-tests can help you detect and replace failing hard drives before they cause permanent data loss. However, there's a couple issues with `smartd`:

- There are more than a hundred S.M.A.R.T attributes, however `smartd` does not differentiate between critical and informational metrics
- `smartd` does not record S.M.A.R.T attribute history, so it can be hard to determine if an attribute is degrading slowly over time.
- S.M.A.R.T attribute thresholds are set by the manufacturer. In some cases these thresholds are unset, or are so high that they can only be used to confirm a failed drive, rather than detecting a drive about to fail.
- `smartd` is a command line only tool. For head-less servers a web UI would be more valuable.

Scrutiny is a Hard Drive Health Dashboard & Monitoring solution, merging manufacturer provided S.M.A.R.T metrics with real-world failure rates.

Features

Scrutiny is a simple but focused application, with a couple of core features:

- Web UI Dashboard - focused on Critical metrics
- `smartd` integration (no re-inventing the wheel)
- Auto-detection of all connected hard-drives
- S.M.A.R.T metric tracking for historical trends
- Customized thresholds using real world failure rates
- Temperature tracking
- Provided as an all-in-one Docker image (but can be installed manually)
- Configurable Alerting/Notifications via Webhooks
- (Future) Hard Drive performance testing & tracking

Getting Started

RAID/Virtual Drives

Scrutiny uses `smartctl --scan` to detect devices/drives.

- All RAID controllers supported by `smartctl` are automatically supported by Scrutiny.
  - While some RAID controllers support passing through the underlying SMART data to `smartctl` others do not.
  - In some cases `--scan` does not correctly detect the device type, returning incomplete SMART data. Scrutiny supports overriding detected device type via the config file: see example.collector.yaml
- If you use docker, you **must** pass though the RAID virtual disk to the container using `--device` (see below)
  - This device may be in `/dev/*` or `/dev/bus/*`.
  - If you're unsure, run `smartctl --scan` on your host, and pass all listed devices to the container.

See docs/TROUBLESHOOTING_DEVICE_COLLECTOR.md for help

Docker

If you're using Docker, getting started is as simple as running the following command:

> See docker/example.omnibus.docker-compose.yml for a docker-compose file.

```
docker run -it --rm -p 8080:8080 -p 8086:8086 \
  -v `pwd`/scrutiny:/opt/scrutiny/config \
  -v `pwd`/influxdb2:/opt/scrutiny/influxdb \
  -v /run/udev:/run/udev:ro \
  --cap-add SYS_RAWIO \
  --device=/dev/sda \
  --device=/dev/sdb \
  --name scrutiny \
  ghcr.io/analogj/scrutiny:master-omnibus
```

- `/run/udev` is necessary to provide the Scrutiny collector with access to your device metadata
- `--cap-add SYS_RAWIO` is necessary to allow `smartctl` permission to query your device SMART data
  - NOTE: If you have **NVMe** drives, you must add `--cap-add SYS_ADMIN` as well. See issue #26
- `--device` entries are required to ensure that your hard disk devices are accessible within the container.
- `ghcr.io/analogj/scrutiny:master-omnibus` is a omnibus image, containing both the webapp server (frontend & api) as well as the S.M.A.R.T metric collector. (see below)

Hub/Spoke Deployment

In addition to the Omnibus image (available under the `latest` tag) you can deploy in Hub/Spoke mode, which requires 3 other Docker images:

- `ghcr.io/analogj/scrutiny:master-collector` - Contains the Scrutiny data collector, `smartctl` binary and cron-like scheduler. You can run one collector on each server.
- `ghcr.io/analogj/scrutiny:master-web` - Contains the Web UI and API. Only one container necessary
- `influxdb:2.2` - InfluxDB image, used by the Web container to persist SMART data. Only one container necessary See docs/TROUBLESHOOTING_INFLUXDB.md

> See docker/example.hubspoke.docker-compose.yml for a docker-compose file.

```
docker run --rm -p 8086:8086 \
  -v `pwd`/influxdb2:/var/lib/influxdb2 \
  --name scrutiny-influxdb \
  influxdb:2.2

docker run --rm -p 8080:8080 \
  -v `pwd`/scrutiny:/opt/scrutiny/config \
  --name scrutiny-web \
  ghcr.io/analogj/scrutiny:master-web

docker run --rm \
  -v /run/udev:/run/udev:ro \
  --cap-add SYS_RAWIO \
  --device=/dev/sda \
  --device=/dev/sdb \
  -e COLLECTOR_API_ENDPOINT=http://SCRUTINY_WEB_IPADDRESS:8080 \
  --name scrutiny-collector \
  ghcr.io/analogj/scrutiny:master-collector
```

Manual Installation (without-Docker)

While the easiest way to get started with Scrutiny is using Docker, it is possible to run it manually without much work. You can even mix and match, using Docker for one component and a manual installation for the other.

See docs/INSTALL_MANUAL.md for instructions.

Usage

Once scrutiny is running, you can open your browser to `http://localhost:8080` and take a look at the dashboard.

If you're using the omnibus image, the collector should already have run, and your dashboard should be populate with every drive that Scrutiny detected. The collector is configured to run once a day, but you can trigger it manually by running the command below.

For users of the docker Hub/Spoke deployment or manual install: initially the dashboard will be empty. After the first collector run, you'll be greeted with a list of all your hard drives and their current smart status.

```
docker exec scrutiny /opt/scrutiny/bin/scrutiny-collector-metrics run
```

Configuration

By default Scrutiny looks for its YAML configuration files in `/opt/scrutiny/config`

There are two configuration files available:

- Webapp/API config via `scrutiny.yaml` - example.scrutiny.yaml.
- Collector config via `collector.yaml` - example.collector.yaml.

Neither file is required, however if provided, it allows you to configure how Scrutiny functions.

Cron Schedule

Unfortunately the Cron schedule cannot be configured via the `collector.yaml` (as the collector binary needs to be trigged by a scheduler/cron). However, if you are using the official `ghcr.io/analogj/scrutiny:master-collector` or `ghcr.io/analogj/scrutiny:master-omnibus` docker images, you can use the `COLLECTOR_CRON_SCHEDULE` environmental variable to override the default cron schedule (daily @ midnight - `0 0 * * *`).

```
docker run -e COLLECTOR_CRON_SCHEDULE="0 0 * * *" ...
```

Notifications

Scrutiny supports sending SMART device failure notifications via the following services:

- Custom Script (data provided via environmental variables)
- Email
- Webhooks
- Discord
- Gotify
- Hangouts
- IFTTT
- Join
- Mattermost
- ntfy
- Pushbullet
- Pushover
- Slack
- Teams
- Telegram
- Tulip

Check the `notify.urls` section of example.scrutiny.yml for examples.

For more information and troubleshooting, see the TROUBLESHOOTING_NOTIFICATIONS.md file

Testing Notifications

You can test that your notifications are configured correctly by posting an empty payload to the notifications health check API.

```
curl -X POST http://localhost:8080/api/health/notify
```

Debug mode & Log Files

Scrutiny provides various methods to change the log level to debug and generate log files.

Web Server/API

You can use environmental variables to enable debug logging and/or log files for the web server:

```
DEBUG=true
SCRUTINY_LOG_FILE=/tmp/web.log
```

You can configure the log level and log file in the config file:

```yaml
log:
  file: '/tmp/web.log'
  level: DEBUG
```

Or if you're not using docker, you can pass CLI arguments to the web server during startup:

```
scrutiny start --debug --log-file /tmp/web.log
```

Collector

You can use environmental variables to enable debug logging and/or log files for the collector:

```
DEBUG=true
COLLECTOR_LOG_FILE=/tmp/collector.log
```

Or if you're not using docker, you can pass CLI arguments to the collector during startup:

```
scrutiny-collector-metrics run --debug --log-file /tmp/collector.log
```

Supported Architectures

Architecture Name	Binaries	Docker
linux-amd64	✅	✅
linux-arm-5	✅	
linux-arm-6	✅	
linux-arm-7	✅	web/collector only. see #236
linux-arm64	✅	✅
freebsd-amd64	✅	
macos-amd64	✅	✅
macos-arm64	✅	✅
windows-amd64	✅	WIP, see #15
windows-arm64	✅	

Contributing

Please see the CONTRIBUTING.md for instructions for how to develop and contribute to the scrutiny codebase.

Work your magic and then submit a pull request. We love pull requests!

If you find the documentation lacking, help us out and update this README.md. If you don't have the time to work on Scrutiny, but found something we should know about, please submit an issue.

Versioning

We use SemVer for versioning. For the versions available, see the tags on this repository.

Authors

Jason Kulatunga - Initial Development - @AnalogJ

Licenses

- MIT
- Logo: Glasses by matias porta lezcano

Sponsors

Scrutiny is only possible with the help of my Github Sponsors.