Status Page Review

Contents

Introduction

As interest and use of Apertis grows it is becoming increasingly important to show the health of the Apertis infrastructure. This enables users to proactively discover the health of the resources provided by Apertis and determine if any issues they may be having are due to Apertis or their infrastructure.

Terminology and concepts

- **Hosted**: Service provided by an external provider that can typically be accessed over the internet.
- **Self-hosted**: Service installed and run from computing resources directly owned by the user.

Use cases

- A developer is releasing a new version of a package they maintain, but the upload to OBS is failing and they need to find out if it is a misconfiguration

on their part or if the OBS service actually down.

Non-use cases

- Providing the Apertis system administrators with a granular over-view of the infrastructure state.

Requirements

- An automated system monitoring status of user accessible resources provided by the Apertis platform.
- The system displays a simple indication of the availability of the resources.
- The chosen system appears to be actively maintained:
  - Hosted services have activity on their website in the last six months
  - Self-hosted projects show signs of activity in the six months
- (Optional) The system is hosted on a distinct infrastructure to reduce shared infrastructure that could lead to inaccurate results.

Existing systems

Numerous externally hosted services and open source projects are available which provide the functionality required to show a status page.

Self-hosted

The self-hosted options fall into 2 categories:

- **Static**: The status page is generated to html pages, stored on a web server which then provides the latest status page when requested.
- **Dynamic**: The page is generated via a web scripting language on the server and served to the user per request.

These include the following options:

Static

- Statusfy[1]
- ClearStatus[2]
- CState[3]
- status.sh[4]
- upptime[5]

[1] https://marquez.co/statusfy
[2] https://github.com/weeblrpress/clearstatus/
[3] https://github.com/cstate/cstate
[4] https://github.com/Cyclenerd/static_status
[5] https://upptime.js.org/

Dynamic

- Cachet[6]
- Gatus[7]

Hosted

Many of the hosted services understandably charge a fee to provide a status page. A small number have free options which provide a basic service. As we are looking for a simple option and as a self-hosted option is expected to cost us very little once setup, we will only be considering the free services. The following options have been found:

- Better Uptime[8]
- Freshstatus[9]
- HetrixTools[10]
- Instatus[11]
- Nixstats[12]
- Pagefate[13]
- Squadcast[14]
- StatusKit[15]
- StatusCake[16]
- UptimeRobot[17]

Approach

As there are an abundance of tools and services available which provide status page functionality, choosing from these existing solutions will be preferred over a home grown solution, assuming that one can be found to fit our requirements, with a home grown solution only considered if none of the existing solutions are appropriate. Our approach is to:

- Determine services that need to be monitored, this will be critical to discount some of the free services that limit the number of services that cam be monitored.
- Each option will be evaluated against the following criteria:

[6] http://cachethq.io/
[7] https://github.com/TwinProduction/gatus
[8] https://betteruptime.com/status-page
[9] https://www.freshworks.com/status-page/
[10] https://hetrixtools.com/pricing/uptime-monitor/
[11] https://instatus.com/
[12] https://nixstats.com/
[13] https://pagefate.com/
[14] https://www.squadcast.com/
[15] https://statuskit.com/
[16] https://www.statuscake.com/features/uptime/
[17] https://uptimerobot.com/status-page/

- Tool provides automated update to status of monitored services
- Tool can be used to monitor all services that we wish to monitor (preferably with some capacity to monitor more in the future if desired).
- Simple interface, providing clear picture of status.
- The tool is actively maintained, either appearing to have active contributions or in the case of services activity on its website.

Evaluation Report

Monitored services

The following services could be monitored to gauge the status of the Apertis project:

- **GitLab**: This is the main service used by Apertis developers which hosts the source code used and developed as part of the project.
- **Website**: This is the main site at www.apertis.org[18]. This is hosted by GitLab pages which is a distinct from the main GitLab service.
- **APT repositories**: This service hosts the `.deb` packages that are build by the Apertis project. This is required in order to build images or update/extend existing apt based installations.
- **Artifacts hosting**: This is where the images built by Apertis are stored along with the OSTree repositories. This service is therefore important for anyone wanting to install a fresh copy of Apertis or update one based on OSTree.
- **OBS**: Apertis utilizes Collabora's instance of the Open Build Service. This performs compilation of the source into `.deb` packages. Whilst this will not be directly interacted with by most users, it is required to be available for updates to be generated when releases are made to packages in GitLab and there may be some cases where advanced users may need access to OBS.
- **LAVA**: Apertis utilizes Collabora's instance of LAVA. This is primarily used to test images built by Apertis and is thus a critical part of the automated QA infrastructure.
- **lavaphabbridge**: This records the outcome of LAVA runs and displays the test cases used for QA.
- **hawkBit**: This is a deployment management system that is being integrated into Apertis. It provides both a web UI and rest API. Both of these should be monitored.

Whilst this list could arguably be reduced a little to just target core services, it would be prudent to choose a service that would allow Apertis room to grow and add services that need monitoring.

[18] http://www.apertis.org

Tool comparison

The following table was created whilst evaluating the options listed under existing systems. To save time, where it was apparent that the option was not going to meet the initial criteria, no further attempt was made to evaluate later criterion, hence the lack of answers on less suitable options.

Tool	Hosting	Automated	8+ Services?	Simplicity	Activity
UptimeRobot[19]	Service	Yes	Yes - 50	Simple	Active
status.sh[20]	Self	Yes	Yes - Unlimited	Simple	Active
Gatus[21]	Self	Yes	Yes - Unlimited	Simple	Active
Better Uptime[22]	Service	Yes	Yes - 10	Moderate	Active
upptime[23]	Self	Yes	Yes - Unlimited	Moderate	Active
HetrixTools[24]	Service	Yes	Yes - 15	Complex	?
StatusCake[25]	Service	Yes	Yes - 10	?	Active
Pagefate[26]	Service	?	?	-	-
Nixstats[27]	Service	?	No - 5	-	-
Statusfy[28]	Self	No	Yes - Unlimited	-	-
ClearStatus[29]	Self	No	Yes - Unlimited	-	-
CState[30]	Self	No	Yes - Unlimited	-	-
Cachet[31]	Self	No	yes - Unlimited	-	-
Freshstatus[32]	Service	No - Requires freshping	-	-	-
Instatus[33]	Service	No - Requires extra service	-	-	-
Squadcast[34]	Service	No	?	-	-
StatusKit[35]	Service	No	?	-	-

Recommendation

Based on the above evaluation, the top 4 options would appear to be:

[19] https://uptimerobot.com/status-page/
[20] https://github.com/Cyclenerd/static_status
[21] https://github.com/TwinProduction/gatus
[22] https://betteruptime.com/status-page
[23] https://upptime.js.org/
[24] https://hetrixtools.com/uptime-monitor/
[25] https://www.statuscake.com/features/uptime/
[26] https://pagefate.com/
[27] https://nixstats.com/
[28] https://marquez.co/statusfy
[29] https://github.com/weeblrpress/clearstatus/
[30] https://github.com/cstate/cstate
[31] http://cachethq.io/
[32] https://www.freshworks.com/status-page/
[33] https://instatus.com/
[34] https://www.squadcast.com/
[35] https://statuskit.com/

- Better Uptime
- Gatus
- status.sh
- UptimeRobot

The choice can be further slimmed by making a decision between a service and a self-hosted solution.

A self-hosted solution has the advantage that it will remain available long-term, not being reliant on an outside provider, however they will also require maintenance and up keep. An externally provided service has the advantage that it is hosted on distinct infrastructure from that hosting the other Apertis services and thus less likely to be made unavailable by a fault affecting the whole platform. An external service is also likely to provide a more independent and reliable evaluation of the platform status.

Based on this our recommendation would be to utilize UptimeRobot to provide a status page for Apertis.

Risks

- UptimeRobot stops providing free service: In the event that the free service ceases to be offered or changes such that it is no longer suitable to Apertis, it would appear to be fairly trivial to migrate to an alternative service or decide to self-host.