Next Generation Monitoring: Tier 2 Experience

R Fay, J Bland and S Jones
Department of Physics, University of Liverpool, Liverpool, L69 7ZE, UK
fay@hep.ph.liv.ac.uk

Abstract. Monitoring IT infrastructure is essential for maximizing availability and minimizing disruption by detecting failures and developing issues. The HEP group at Liverpool have recently updated our monitoring infrastructure with the goal of increasing coverage, improving visualization capabilities, and streamlining configuration and maintenance. Here we present a summary of Liverpool’s experience, the monitoring infrastructure, and the tools used to build it. In brief, system checks are configured in Puppet using Hiera, and managed by Sensu, replacing Nagios. Centralised logging is managed with Elasticsearch, together with Logstash and Filebeat. Kibana provides an interface for interactive analysis, including visualization and dashboards. Metric collection is also configured in Puppet, managed by collectd and stored in Graphite, with Grafana providing a visualization and dashboard tool. The Uchiwa dashboard for Sensu provides a web interface for viewing infrastructure status. Alert capabilities are provided via external handlers. A custom alert handler is in development to provide an easily configurable, extensible and maintainable alert facility.

1. Introduction
Monitoring services are used at Liverpool to assess the state of HEP services, the infrastructure used to host those services, and the state of some external dependencies (such as external network connectivity). The objectives of the monitoring infrastructure are:

- to provide an accurate and current overview of the state of the HEP IT infrastructure;
- to provide additional visualization of metrics and logs;
- to provide alerts when any state changes; and
- to be easily maintainable as the HEP IT infrastructure evolves.

2. Overview
The monitoring service infrastructure is comprised of five processes that collectively meet the objectives. These are as follows:

2.1. Checks
Checks are tests on services and hosts to determine state. Some run on service hosts (e.g. to check if an http service is running), while others run externally to the service host (e.g. to check an http service is externally available). The results are recorded, typically on a single server, displayed via a dashboard, and used to trigger alerts.
2.2. **Metrics**

Metrics are statistics that provide data such as CPU load, free disk space, system temperatures, etc. These are gathered from hosts, collated, and can be displayed via a dashboard. They are subsequently used for diagnosis, analysis, and identification of trends for prediction.

2.3. **Logs**

Logs refers to the log files produced by systems and applications, which can be used for diagnosis and fault-finding. This process covers how the logs are stored and accessed (e.g. through centralized log collection), and analysis of these logs.

2.4. **Alerts**

Alerts are notifications, typically of state changes, indicating a possible need for intervention.

2.5. **Management**

The management process consists of the service and/or processes used to manage the configuration, and distributed configuration where appropriate, of the other monitoring processes. It is vital in maintaining the service, ensuring it remains current.

3. **Previous Monitoring Setup**

The previous monitoring setup mainly used Nagios [1] and Ganglia [2], managed with Puppet [3], together with locally developed scripts.

3.1. **Checks**

Nagios, a widely used open source monitoring application, was the primary service used to provide checks at Liverpool. In Nagios, checks are typically defined on the server and actively run by the server according to a schedule. ‘Passive’ checks can also be defined; these are still defined on the server, but receive their results from external applications or processes. At Liverpool, all checks were configured directly on the Nagios server rather than being managed by Puppet as, while it would have been possible to configure them via Puppet, this would have come with some overhead and offered little advantage (see 3.5). All checks defined in Nagios were ‘active’ checks, with the exception of additional checks made using Testnodes [4], a separate Python-implemented program that periodically ran a check script installed on each worker node to ensure fitness for running Grid jobs, and automatically marked the node offline in the batch queue if the node failed the check. As well as controlling the online/offline status of cluster nodes, Testnodes recorded its output in Nagios by submitting its results as the status for passive checks defined on the Nagios server.

3.2. **Metrics**

Ganglia is a BSD-licensed scalable monitoring application, developed for HPC systems. At Liverpool, Ganglia was used to collect and view metrics (e.g. CPU load, memory in use, free diskspace). In addition, the open-source network monitoring and graphing application Cacti [5] was used to monitor network traffic on switches, with the open source network visualisation Weathermap [6] plugin being used to provide a visual representation of network activity.

3.3. **Logs**

Command-line tools and scripts were used to access and analyse logs directly on the individual hosts where the logs were generated, and over centralised backup of logs where appropriate. While generally effective, this required any specific analysis to be scripted, and visualisation, in particular, required significant effort with limited reusability.
3.4. Alerts
Nagios was configured to generate email alerts. These were based on the combination of contacts, contact groups, notification periods, template configuration, and per-service overrides. Default settings were generally configured with low thresholds to ensure alerts were raised in a timely fashion.

3.5. Management
Puppet was used to manage the installation and configuration of client packages and some configuration (including the Nagios client software package, the Testnodes client script, and the Ganglia client and its configuration). The Nagios server was not managed by Puppet, and instead the Nagios configuration was manually edited directly on the Nagios server. While Puppet management was considered, it would essentially have been a matter of implementing the Nagios configuration in Puppet syntax on the Puppet server (as opposed to configuring Nagios directly on the Nagios server). This would have come with the overhead of translating the existing Nagios configuration (Nagios having been introduced at Liverpool before Puppet), and the additional procedural step of manually running, or waiting for a scheduled run, of the Puppet client on the Nagios server following any changes or updates to Nagios configuration. As only one Nagios server was in use at Liverpool, the benefit of re-using Puppet configuration for multiple systems was not a factor. The primary advantage of Puppet configuration, therefore, would have been ease of reinstallation in the event of a server rebuild, but a comprehensive backup policy at Liverpool made that non-essential and overall the overheads were seen to outweigh any possible benefits.

3.6. Summary
The monitoring setup worked well for the most part, enabling Liverpool to identify and respond to issues rapidly, and consistently attain a high standard of availability and efficiency. However, over time the Nagios configuration became increasingly awkward to maintain; with the check (Nagios) configuration entirely separate to the definition of services (typically in Puppet), introducing new services required configuration changes to be made and tested in two separate locations. Ensuring the consistency of service and check specification was seen to become, over time, an increasingly onerous task. The increasing use of virtualisation within HEP, on both servers and on worker nodes, introduced an increasingly dynamic element to the configuration of services and monitoring, exacerbating the issues caused by the separation of the two.

Additionally, adjusting the configuration in Nagios to ensure alerts were generated promptly to ensure rapid intervention, but not at such low thresholds as to result in an overwhelming volume of alerts, proved difficult in practice. With configuration being based on the combination of contacts, contact groups, notification periods, template configuration and per-service overrides, the default was inevitably set to err on the side of too many alerts, with the notion that this would be refined for particular service/host checks depending on their characteristics. However, in practice, the alerts were too often left at the default level due to time constraints and the cumbersome process of making adjustments for individual services and hosts, resulting in an unnecessarily high volume of alerts.

4. New Monitoring Setup
The new monitoring setup was designed to address the key issues identified with the original monitoring infrastructure; specifically, to more closely unify the management of the service configuration with the monitoring configuration, to improve log analysis and visualization capabilities, and to provide more streamlined adjustment of monitoring alerts.

4.1. Checks
Sensu [7] is a monitoring platform designed to replace Nagios, featuring monitoring plugin compatibility allowing existing Nagios checks to be easily ported to Sensu. Sensu offers two types of checks; centrally defined and scheduled checks (called ‘subscription’ checks, as they are shared with
the client by a publish/subscribe model), and ‘standalone’ checks that are defined, scheduled, and executed by the local client. The Liverpool configuration mainly uses ‘standalone’ checks. This allows checks to be defined in Puppet on a client, rather than server, basis offering significant management advantages over the previous Puppet and Nagios combination (see 4.5). For monitored systems where it is not possible to run a Sensu client (e.g. network switches), additional ‘standalone’ checks are defined on the Sensu server to check the system via SNMP, for example. Subscriptions are also defined, but only used for adding temporary, non-Puppet defined, checks. The Uchiwa [8] dashboard is used to view the current status of checks as recorded by the Sensu server. Old checks can also be deleted from Sensu via the Uchiwa dashboard. It is also possible to script automated removal of old check results as systems are retired.

4.2. Metrics

Metrics are collected with collectd [9], configured on nodes by Puppet. The metrics are then sent to Graphite [10] and stored with Graphite’s Carbon storage application. Grafana [11] provides visualisation. Sensu checks can be configured to run on the Graphite server, querying stored metrics, e.g. referring to rolling averages. Compared to Ganglia, collectd with Graphite & Grafana requires significantly more installation and configuration for collection of basic site metrics. Collectd with Graphite & Grafana is, however, easier to expand and considerably more powerful than Ganglia. We are continuing to use Ganglia, along with Cacti and Weathermap for network traffic metrics, alongside collectd and Graphite and Grafana as the latter continues to be expanded.

4.3. Logs

Logs are collected with Filebeat [12], processed with Logstash [13], and stored in Elasticsearch [14], all configured with Puppet. Kibana [15] provides an interface for searching and visualisation of logs. While Elasticsearch and Logstash modules from the Puppet Forge help with installation and configuration, the process is relatively complex. For logs where pre-written grok filters (used to parse logs into fields capable of being indexed and searched) are either not available or unsuitable, defining efficient and correct grok filters is a non-trivial task requiring a degree of expertise. The ELK stack is also relatively demanding in terms of hardware resources, potentially requiring scaling up to a cluster of multiple nodes as the volume of logs processed through it increases. Once configured, ELK provides analytical and visualisation capabilities that are hard to match.

4.4. Alerts

Sensu allows the definition of handlers to process alerts from checks. Liverpool currently has a simple email handler defined. This allows state changes from checks to be raised simply and directly via email, but it offers little fine-grained control, short of adding manual definitions and control code directly to the handler script. Consequently, we are developing a custom alert handler, named Lerts to process Sensu events and replace Nagios notifications. As mentioned in 3.6, we found with a sufficiently large infrastructure, even one that's in principle largely static, that a static configuration of event handling is difficult to accurately and comprehensively define, resulting in excessive alerts (which can be overlooked due to volume) or inadequate alerting. Software that does not allow simple tweaking, instead requiring relatively complex static configuration defined in text files to be edited, tested, and services subsequently restarted, can become unwieldy, resulting in a potential degradation of the service over time. ‘Lerts is designed to provide dynamic adjustable handling of events, configurable on both a system and individual user level, allowing alert handling to be adjusted easily over time to reflect evolving requirements and circumstances.

4.5. Management

Puppet continues to be used to manage the monitoring services, but has been updated to use Hiera to provide a hierarchical configuration. This allows common definitions based on operating system, for example, to be easily defined. In addition, ‘standalone’ Sensu checks can be defined with Hiera
configuration, on a group or individual node level. This allows both monitoring checks and the services they monitor to be included in one single location. This means that if (for example) a new virtualised Scientific Linux 6 system is created, it will automatically receive the configuration for all services and associated monitoring defined on a site and OS level without any system specific Puppet configuration required at all. This simplifies the initial process of creating the appropriate definitions across the HEP infrastructure, the process of ensuring the integrity of checks and services, and the process of maintaining configuration over time. It would also be possible to combine monitoring and service configuration in ‘wrapper’ modules, allowing both to be defined for a specific host or group in one package. This would help to ensure the coverage of monitoring checks to services by effectively automatically including checks when services are included, but would require additional effort to create the wrapper modules initially.

5. Conclusions and Future
A solution using configuration management (such as Puppet) with standalone checks defined on systems (as enabled by Sensu) enables services and checks to be defined in one place, simplifying configuration and reducing the likelihood of omissions. Graphite and Grafana are initially more demanding to set up for a basic configuration than Ganglia, but can be more easily extended and offer significantly greater visualisation capabilities. Centralised logging with ELK enables faster and more comprehensive debugging and analysis, but can be demanding in resources, and complex to set up initially. The assessed combination of software overall offers great potential for monitoring and analysis, but at a potentially higher cost in terms of manpower and resources for initial setup which may not be available at all Tier 2 sites; for effective widespread adoption, it is likely that collaborative, maintained effort would be necessary to reduce overheads through development and provision of common installation instructions and configurations.

References
[1] Nagios https://www.nagios.org/
[2] Ganglia http://ganglia.info/
[3] Puppet https://puppet.com/
[4] Fay R and Bland J 2014 Testnodes: a Lightweight Node-Testing Infrastructure J. Phys.: Conf. Ser. Proceedings of CHEP2013. 513 doi:10.1088/1742-6596/513/6/062013 http://iopscience.iop.org/article/10.1088/1742-6596/513/6/062013
[5] Cacti http://www.cacti.net/
[6] Weathermap https://network-weathermap.com/
[7] Sensu https://sensuapp.org/
[8] Uchiwa https://uchiwa.io/
[9] Collectd https://collectd.org/
[10] Graphite https://graphiteapp.org/
[11] Grafana http://grafana.org/
[12] Filebeat https://www.elastic.co/products/beats/filebeat
[13] Logstash https://www.elastic.co/products/logstash
[14] Elasticsearch https://www.elastic.co/products/elasticsearch
[15] Kibana https://www.elastic.co/products/kibana