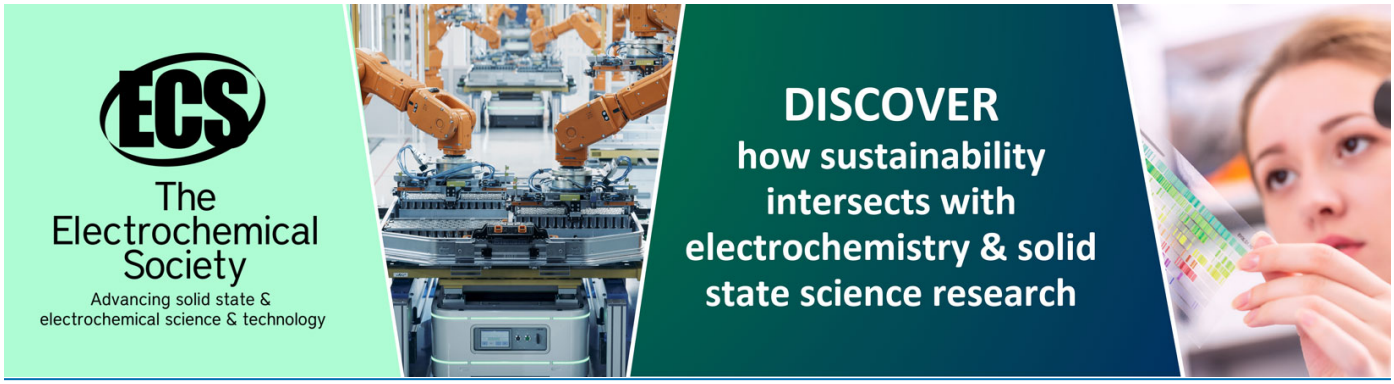
Tier-2 System Administration: A Comprehensive Approach



Journal of Physics: Conference Series **513** (2014) 062005 doi:10.1088/1742-6596/513/6/062005

Tier-2 System Administration: A Comprehensive Approach

J Bland[1], R B Fay, S H Jones and M J Norman

Department of Physics, University of Liverpool, Liverpool, L69 7ZE, UK

E-mail: jbland@hep.ph.liv.ac.uk

Abstract. Liverpool is consistently amongst the top Tier-2 sites in Europe in terms of efficiency and cluster utilisation. This paper will cover the work performed at Liverpool over the last six years to maximise and maintain efficiency and productivity at their Tier 2 site, with an overview of the tools used (including established, emerging, and locally developed solutions) for monitoring, testing, installation, configuration, ticketing, logging, and administration, along with the techniques for management, operations, and optimisation that tie them together into a comprehensive, scalable, and sustainable approach to Tier 2 administration.

1. Introduction

It is easy to take an ad hoc approach to site administration, employing whatever tools seem appropriate at any given moment. However, as a long-term strategy this can lead to constant "fire-fighting" potentially resulting in loss of efficiency in terms of both system and human resources.

Inevitably this will result in periods of downtime. While such fire-fighting cannot be avoided entirely, it is by employing a comprehensive framework approach that we at Liverpool have sought to keep such disruption to a minimum. This has benefitted the users of the Liverpool Tier2 site, which include many of the WLCG Virtual Organisations such as ATLAS and LHCb, who require continuous operation and accessibility.

1.1. Goals

The aims of a comprehensive approach are to ensure that all site administration activity and consequent effects on other activities is managed, not just by the individual but by all those with collective responsibility. To this end, it is necessary to ensure an appropriate set of tools and procedures are in place to cover the following areas: Communication, Management, Monitoring, Documentation, Control and Protection.

1.2. Principles

To realise these objectives, the procedures and tools at our disposal need to be employed in the appropriate manner. Where practical, system changes (upgrades, new installations, etc) should be tested before being fully deployed and any changes should have a rollback path where practical. Fundamental issues should be addressed first, with initially only sufficient regard to possible future optimisations to avoid inadvertent preclusion, optimising then taking place where time allows. Any

[1] To whom any correspondence should be addressed.

tools and procedures that are implemented should be sustainable in respect of demands on human and technical resources. Sites should avoid solutions that involve becoming locked in to a particular solution where practical, whether it is a software package, piece of hardware or a vendor.

2. Communication

Communication is essential for coordinating and directing effort (both locally at a site and across the global community). In conjunction with documentation, it provides the means for discussion, decision-making, and dissemination of information.

2.1. Email

Email is a long established communication medium and it can be assumed that any user or system will be capable of both sending and receiving it. Email communications will include system reports and warnings, communications from the global community including required updates, user tickets and other operational issues. It is a very capable medium for communications that don't require immediate response such as detailed technical discussions, or enquiries to support personnel who are widely dispersed in location and availability. It is easy for any person or system to access and can maintain threads for future reference.

2.2. Chat

For more immediate communication, such as for on-line meetings or immediate support, chat or video conferencing clients are more appropriate than email. There are numerous examples such as IRC [1], Skype [2], Seevogh [3] and Vidyo [4]. The wide diversity can be problematic compared to the ubiquity of email, groups must decide upon one protocol to prevent personnel being excluded and it is easy for threads of discussion to become lost or forgotten, similar to face-to-face discussions.

2.3. Issue-tracking

To ensure that problems aren't forgotten and left unresolved, it is essential to keep track of issues arising from hardware, services or user requests. We have used RT [5] successfully for bug tracking and helpdesk ticketing. We have found that automatic integration with monitoring systems such as Nagios [6] or the Nagios fork Icinga [7] makes operational issues more manageable and easier to assign to an appropriate system manager.

2.4. Face to face

Discussions, both in formal meetings and informal conversations, are still required from time to time. Although they can be efficient for relaying information and stimulating ideas they should have notes taken where possible and follow-ups (e.g. via email) to ensure the results are not forgotten or lost.

3. Management

With focus on the management of technical items rather than the human aspects, this covers the tools and procedures used to administrate infrastructure, including the definition and configuration of OS and services.

3.1. Provision and Configuration

Liverpool has been using Puppet [8] for over 5 years for package management, service configuration and control. The reduction in administrator overhead through automation of general system configuration and management is a large initial gain. Combined with version control the system states can be rolled forwards or backwards when required. Through extensions and external commands Puppet can control all aspects of system configuration. Tier 2 sites maintain large clusters of systems, with rapidly changing requirements, and regular upgrades. This can cause a disruptive drain on administrator time unless configuration management tools such as Puppet are used.

3.2. Package Management

Yum [9] provides a very flexible and extensible package management system. At Liverpool we mirror regularly used repositories locally for performance and control reasons, as well as maintaining local repositories for specific site or group configurations. Pakiti [10] can be used to check the status of package installation.

3.3. Middleware Configuration

Some middleware components have complex, site-specific configurations. Site administrators do not necessarily have the time, experience or information necessary to do this themselves and hence require tools written by experts to perform these tasks. This can reduce duplication of effort at every site.

Traditionally Yaim [11] has been the common configuration tool for WLCG software although this functionality is being ported to Puppet. There is still a need for specialized tools in some aspects of the middleware configuration, such as Vomssnooper [12] for maintaining consistent and accurate VO configuration at sites.

3.4. Code Management

Good code management tools are essential not just for code development but also in site configuration. The same tools, such as Subversion [13] or Git [14], can be used, thus utilizing existing knowledge and systems, both in local site repositories, which can be managed with tools such as GitLab [15], and in public repositories e.g. GitHub [16]. Such tools provide the benefits of configuration backups, change history and version control.

4. Monitoring

Monitoring at Liverpool is roughly divided between the broad gathering of metrics and targeted testing. Metrics can be used to detect problems and known issues, while targeted testing can raise alerts for general issues or for specific tests. Tools used for monitoring include Nagios and Testnodes [17], which is a locally developed application for assessing worker node fitness. It is also necessary to monitor the environmental conditions on both the system and site level (e.g. to detect overheating resulting from broken cooling).

4.1. Metrics

Ganglia [18] is a widely used metric gathering tool that collects and collates general system metrics both now and in the past. It can be a valuable tool for detecting and tracking performance problems or system issues over time. Many performance or general issues at Liverpool (such as network bottlenecks or CPU utilization) have been detected and followed using Ganglia and related plugins like Monami [19]. Newer technologies e.g. Graphite [20] offer even more flexibility in gathering, storing and displaying the metrics.

4.2. Monitoring, Testing, Alerting

General site monitoring is via Nagios, giving a framework to regularly test hosts and services, directly or through user-extensible plugins. Early warnings provided by these tests help ensure problems aren't overlooked, particularly when combined with other management tools such as tickets. Some systems require more regular, tightly scheduled tests, e.g. using Testnodes to check for worker node problems before they cause job failures.

5. Documentation

Good documentation ensures that information necessary to understand and hence administer the site is available both now and in the future. The process draws upon the outputs of site-related communication and management, but requires additional infrastructure and procedures to render it comprehensive. It is all too common for vital documentation to be left until a later that never comes ("I'll do it when it's finished").

Local procedure at Liverpool aims to ensure a basic level of documentation is achieved in-line with most activities (comments in Puppet configuration, code, commits) reducing the effort required to keep full documentation up to date. An easily accessible iterative interface for code allows documentation to be kept current in step with maintenance activity.

Many of the tools and sources of documentation have already been covered such as E-mail, RT, Puppet, Subversion or Gitlab. The information in these sources should be distilled into a format that is easily accessible such as a document management system e.g. TWiki [21]. A collaborative tool like TWiki allows anyone to create and update documents. This is essential to prevent information becoming lost or stale.

6. Control

Control provides the means by which actions are ultimately carried out on the site, both directly by administrators, and indirectly, e.g. arising from monitoring and management tools. It is not practical to manage systems purely from the physical console on anything beyond the smallest site. Consequently it is necessary to have the tools (together with the information, see Section 5) to manipulate the site hardware and environment as necessary. Given the inherent power of these tools, they must be clear to use (with a well-developed UI) and secure.

6.1. Low-level system control

IPMI [22] is a standard hardware control and monitoring protocol. Servers and worker node power and system health can be monitored and alerts generated when metrics exceed thresholds. This can be integrated with central monitoring systems e.g. Nagios and Ganglia. Most implementations allow command line access as well as graphical interfaces through utilities or web interfaces. They allow management of complete clusters from a single interface and the GUIs can also provide a virtual console interface for remote administration.

6.2. General system access

Accessing systems remotely to run commands and control GUIs is often necessary, even if it's only for debugging purposes. Systems without IPMI interfaces can be accessed remotely through KVMoIP switches where graphical interfaces are required or when the system is without networking. Network access through SSH provides a secure connection and can be used interactively or in scripts. Parallel execution of commands is necessary when managing large clusters. Liverpool uses a locally developed 'parallel' script but general alternatives include ClusterShell [23].

6.3. Other tools

Batch systems and other services provide their own tools for managing nodes e.g. Torque's [24] pbsnodes command.

7. Protection

Protection covers the measures taken to ensure the site's robustness in the face of failures and disasters. This includes both initial design and specification to ensure high availability, along with security procedures and backups. Design and specification includes, where practical, putting in redundancy on a supply level e.g. UPSes on two electrical feeds and network resilience, system level e.g. disks and PSUs, and site level e.g. fail-over servers. Liverpool takes backups of site configuration, logs, databases and important user data regularly using rdiff-backup [25] for space efficient incremental backups. This data is stored both off-system and off-site. The procedures used for protection should inform recovery procedures (see Section 5).

7.1. Security

Maintaining a secure system is essential for protection of resources and data. Access to resources and systems should require appropriate authorization. Network services should be firewalled by default

and access limited to groups or systems that are authorized. Linux provides firewalling at the system level through iptables [26], although firewalls are not recommended for WAN connections due to performance issues. System access should be monitored e.g. with logwatch [27] or Nagios plugins, and alerts generated. Security issues can escalate and spread quickly hence they need to be discovered and fixed early.

Ensuring system packages and configurations are kept up to date and patched in a timely fashion helps prevent future problems through flaws that are forgotten or overlooked being combined with new flaws in harmful ways.

8. Conclusion

Following a comprehensive approach to site management has allowed the Liverpool Tier2 site to reduce downtime and the impact on administrator effort. The savings in administrator effort have been reinvested in improving site efficiency and creating tools like VomsSnooper and Testnodes. Newer and more effective technologies will continue to be evaluated and added to the current framework.

References

[1] Oikarinen J and Reed D 1993 Internet Relay Chat Protocol *RFC 1459*
[2] Skype http://www.skype.com/
[3] Seevogh http://research.seevogh.com/
[4] Joao Fernandes *et al* 2010 Virtuality and efficiency – overcoming past antinomy in the remote collaboration experience *J. Phys.: Conf. Ser.* **219** 082006
[5] RT http://www.bestpractical.com/rt/
[6] Nagios http://www.nagios.org/
[7] Icinga http://www.icinga.org
[8] Val Hendrix *et al* 2012 Scientific Cluster Deployment and Recovery – Using puppet to simplify cluster management *J. Phys.: Conf. Ser.* **396** 042027
[9] Yum http://yum.baseurl.org/
[10] Pakiti http://pakiti.sourceforge.net/
[11] YAIM https://twiki.cern.ch/twiki/bin/view/EGEE/YAIM
[12] Stephen Jones 2013 VomsSnooper - a tool for managing VOMS records *CHEP2013*
[13] Subversion http://subversion.apache.org/
[14] Michal Husejko 2013 Collaboration platform @CERN : Self-service for software development tools *CHEP2013*
[15] GitLab http://gitlab.org/
[16] GitHub https://github.com/
[17] Robert Fay 2013 Testnodes – a Lightweight Node-Testing Infrastructure *CHEP2013*
[18] Ganglia http://ganglia.sourceforge.net/
[19] A P Millar *et al* 2008 Monitoring with MonAMI: a case study *J. Phys.: Conf. Ser.* **119** 062037
[20] Graphite http://graphite.wikidot.com/
[21] Nir Amram *et al* 2010 The use of the TWiki Web in ATLAS *J. Phys.: Conf. Ser.* **219** 082008
[22] IPMI http://www.intel.com/content/www/us/en/servers/ipmi/ipmi-home.html
[23] ClusterShell http://cea-hpc.github.io/clustershell/
[24] Torque http://www.adaptivecomputing.com/products/open-source/torque/
[25] rdiff-backup http://rdiff-backup.nongnu.org/
[26] iptables http://www.netfilter.org/projects/iptables/
[27] Logwatch http://logwatch.sourceforge.net/