ATLAS TDAQ System Administration: Master of Puppets

S Ballestrero[1], F Brasolin[2], D Fazio[3], C Gament[3,4], C J Lee[5,8], D A Scannicchio[6], M S Twomey[7]

[1] University of Johannesburg, South Africa
[2] Istituto Nazionale di Fisica Nucleare Sezione di Bologna, Italy
[3] CERN, Switzerland
[4] Polytechnic University of Bucharest, Romania
[5] University of Cape Town, South Africa
[6] University of California Irvine, United States of America
[7] University of Washington, United States of America

E-mail: atlas-tdaq-sysadmins@cern.ch

Abstract. Within the ATLAS detector, the Trigger and Data Acquisition system is responsible for the online processing of data streamed from the detector during collisions at the Large Hadron Collider at CERN. The online farm is comprised of ∼4000 servers processing the data read out from ∼100 million detector channels through multiple trigger levels. Configuring of these servers is not an easy task, especially since the detector itself is made up of multiple different sub-detectors, each with their own particular requirements. The previous method of configuring these servers, using Quattor and a hierarchical scripts system was cumbersome and restrictive. A better, unified system was therefore required to simplify the tasks of the TDAQ Systems Administrators, for both the local and net-booted systems, and to be able to fulfil the requirements of TDAQ, Detector Control Systems and the sub-detectors groups. Various configuration management systems were evaluated, though in the end, Puppet was chosen as the application of choice and was the first such implementation at CERN.

ATL-DAQ-PROC-2016-041
20 December 2016

1. Introduction

In this paper, the Trigger, Data and Acquisition (TDAQ) [1] Systems Administrators (SysAdmins) of the ATLAS [2] experiment at the LHC will explain the implementation of a Configuration Management Systems (CMS) for use within the ATLAS Technical and Control Network (ATCN), and the General Public Network (GPN) at CERN. Including the reasoning behind the choices made by the SysAdmins and how it has evolved over the years.

2. When

This project initially started in 2010. At that time the de facto CMS in use within CERN was Quattor [3] and the SysAdmins started evaluating other options as a replacement. While the SysAdmins initially began to work on the conversion with Scientific Linux 5 (SLC5) [4], it was not until systems were migrated to Scientific Linux 6 (SLC6) in 2013 that the SysAdmins were able to completely dismiss Quattor. Over the next two years, many further additions and

[8] Primary Author

tweaks were added, and the Sysadmins are now able to completely re-install a local-boot system in under an hour, build a new PXE image for deployment in the same time, and constantly ensure the sanity of the computing farms.

3. What

Various CMS's were evaluated in order to try streamline the task of managing and controlling such a large number of servers. These included Puppet [5], Chef [6] and CF Engine [7], which were the main competing products available at the time. In the end, Puppet was chosen as the application of choice. This is believed to be the first such implementation at CERN. During the "Long Shutdown 1" (LS1), CERN IT also adopted Puppet as their CMS of choice and today many other WLCG [8] applications have also moved towards Puppet modules that are used to not only install, but also configure their applications.

Quattor was deeply ingrained into many of the systems. Initially with SLC5 the migration to puppet was done on local installations and only on servers that provided core services such as DNS, DHCP, NFS, etc. This period allowed the SysAdmins to not only learn how to use Puppet effectively, but also to pre-plan for the migration to SLC6 which would be needed in the Large Hadron Collider (LHC) Run 2. Most of the computing systems within ATLAS make use of a "net-booted" system, where a basic image is supplied via PXE to a node which uses this image to boot. Many of these nodes are also diskless systems, meaning the entire operating system had to be loaded into memory. Once the computer had booted, a complicated system of hierarchical bash scripts called "BootWithMe" (BWM) was used to configure each node, based on the requirements of each sub-detector.

With SLC6 an entire new boot system was developed. Part of the reasoning behind this was due to the memory limitations mentioned previously. SLC6 was larger than the previous operating systems and this meant that there would be less free RAM available for the machines to use. The method used was one very similar to the live image CD's used by Linux. Most of the underlying filesystem is stored on a Local File Server (LFS), which is provided to each machine via a Read Only Network File System (NFS). This allowed the SysAdmins to free up a very large portion of RAM that would otherwise be wasted on applications that were common to all systems.

The SysAdmins create this image in a "Change Root" (CHROOT) environment. Puppet is then used to install all the packages required and deploys a very basic configuration that would be required by all systems. Once the system has booted, an INIT-V script is used which runs *puppet apply* from an NFS mount. This applies the final configuration required by the machine and ensures all the services it needs are started and working before releasing the machine to be used in production. A sane state of the machine is ensured via a cron job which runs every hour on these systems.

Local installed machines make use of a daemon and a Puppet server. After the Linux Anaconda kickstart is finished, Puppet is installed in post installation scripts and started via an INIT-V service. This makes sure that everything is configured as required. Only after a three successful runs of Puppet, the machine is rebooted and released for production. Puppet is started at boot in daemon mode, which then runs every 30 minutes and ensures that the correct configurations are applied.

Figures 1 and 2 show the workflow of the new design using Puppet at the heart of the process.

4. Why

The previous method of configuring these servers, using Quattor and a hierarchical scripts system, was cumbersome and restrictive. Systems Administrators needed to know multiple different scripting languages in order to make simple changes, and configuration files were all kept in multiple places. Puppet instead allowed a single application, which could be used on

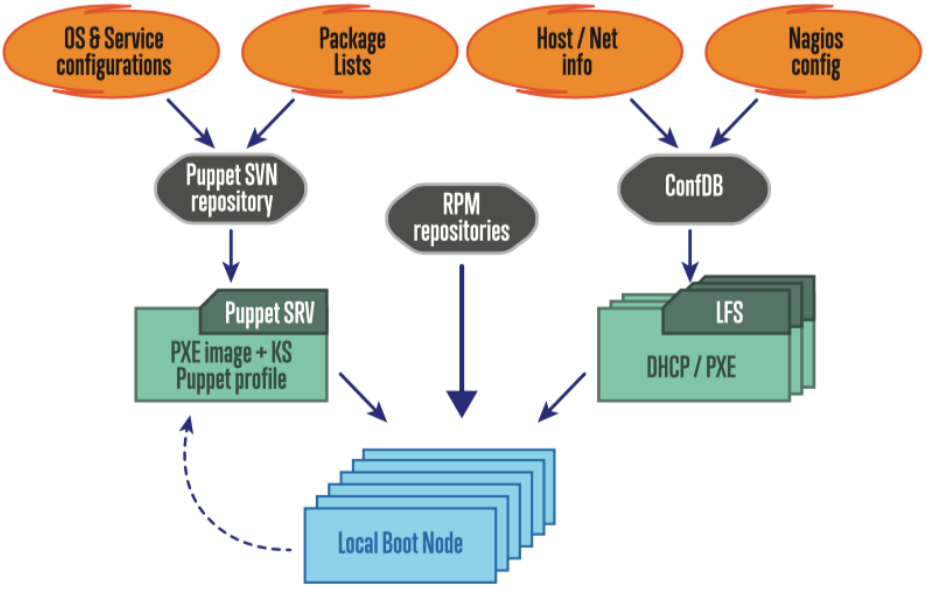

Figure 1. Puppet local-booted workflow.

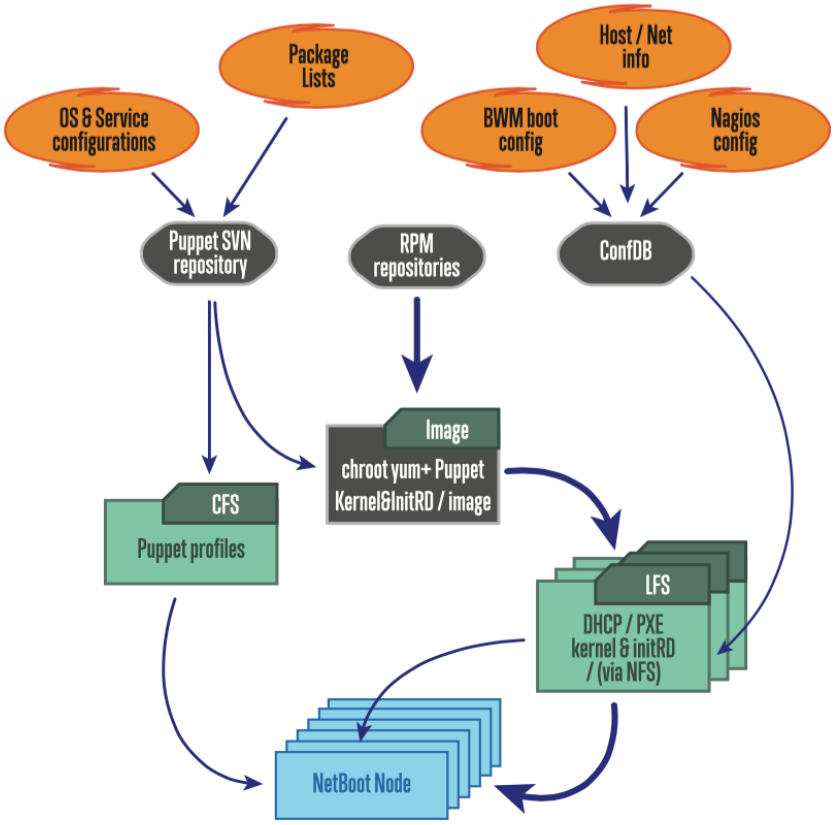

Figure 2. Puppet Net-Booted workflow.

```
### MySQL server daemon and basic config
class mysql::server ($poolsize="256M") {
    $mysqlserver_package      = hiera('mysqlserver_package')
    $mysql_service            = hiera('mysql_service')
    pkg {"$mysqlserver_package":ensure=>installed;} ->
    service {
        "$mysql_service":
            enable=>true,ensure=>running,hasstatus=>true,
    }
    ## Some basic performance settings that should be fine for all
    augeas {
        "mysql_server":
            incl => "/etc/my.cnf",
            lens => "Mysql.lns",
            changes => [
                "set target[ . = 'mysqld' ]/max_allowed_packet 32M",
                "set target[ . = 'mysqld' ]/innodb_file_per_table 1",
                "set target[ . = 'mysqld' ]/innodb_buffer_pool_size $poolsize".
            ],
            require=>Package["$mysqlserver_package"],
            notify=>Service["$mysql_service"]
    }
}
```

Figure 3. Simplistic mysql Puppet code.

both local and net-booted systems, and a single programming language (Puppet DSL). While not a very advanced language, it allows for easy training of new systems administrators. Code is re-usable and also much easier to maintain. In the beginning there were not many tools available for large farms. While applications like Quattor were widely accepted by the HEP community, due to resources and other restrictions, development was always going to be slow. When Puppet became more widely used and accepted by the IT industry, it rapidly exceeded the capabilities of the custom tools. This allows us to profit from a much larger pool of expertise and concentrate on other issues.

One of the advantages of Puppet is that it has a large online community of people who are able to help with issues. There are also modules that are pre-built by community members that are available on the Puppet forge. Unfortunately this is also sometimes a problem. Many modules are built to accommodate every possible operating system and configuration system, and many times this can dramatically increase the complication. For example, the MySQL module available on the Puppet forge, currently contains 32980 lines of code. While this module is incredibly robust and constantly being updated, it would be excessive to use. The SysAdmins are able to accomplish what is needed in only 21 lines of code, as can be seen in Figure 3. Due to this and the unique nature of the ATLAS experiment, many of the modules have to be custom designed. Due to resources such as the Puppet forge, the modules used do not need to be built from the ground up, and existing modules can be stripped down for use, and this drastically helps speed up the workflow process.

5. Where

In the GPN there are two Puppet Master servers running in the TestBed for daemonised Puppet runs, servicing ∼140 local-boot systems. For net-booted systems there is an NFS server which provides the manifests which are used to applying catalogs to ∼250 machines. The Puppet Masters are run on virtual machines on separate virtual hosts. One machine is used for production systems, while the second is for development. They have been created as identical Certificate Authorities for certificate management. This allows the SysAdmins to use

the development server directly on production machines if needed, without the need to deploy new certificates. In the case of a critical failure of one host, all that is needed is to move the main alias to the other machine and it will instantly take over the role of the production host. The net-booted systems are also configured to use an alias of the Central File Server (CFS) which deploys the NFS mount to all systems using this infrastructure. This mounted area contains two separate directory structures for production and development and changes can be made easily to one without any effect on the other.

All of the Puppet manifests are versioned using subversion (SVN) and can quickly be updated or merged between the machines. There are four separate subversion areas: GPN local, GPN net-boot, ATCN local, ATCN net-boot. When new configuration changes have been successfully tested and deployed in the TestBed, they are merged to a checkout of the ATCN subversion. Wrapper scripts have been created to allow the SysAdmins to do this merging and allow not only the merging of modules between GPN and ATCN, but also common modules that are used for local and net-booted systems.

Once SVN merge has been made, the Puppet Masters in the ATCN are updated, which has an identical layout with respect to the Puppet Masters and CFS in GPN, except in ATCN there are ∼500 local-boot and ∼3000 net-booted systems that are managed by these servers and locations.

6. How

Puppetising the "beast" that is ATLAS is far from an easy task. Even now while everything is currently in production, there are still many changes that can, and need, to be done. During the first half of 2016 we again started migrating from Puppet 2.7 to Puppet 3.3. This required many changes to all the code to comply with the changes and deprecated features in ∼20000 lines of code. The gain however was a 50% to 60% reduction of catalog compilation time, which meant quicker deployment of configurations to the farm. During the second half of 2016, requests were made to upgrade the Detector Control Systems to the next OS that is to be used by CERN based on CentOS 7 (CC7) [9]. The main reason for remaining on Puppet 3.3 was due to the packages that are available in the CERN repositories. For CC7 Puppet 3.6 is available, so a decision was made to bring version 3.6 into the personal repositories used for SLC6. This is due to the fact that while a Puppet client can be a version lower than the Master server, it is not recommended the other way around. The SysAdmins also took the opportunity to move to the new environment system and fixed deprecated features that would be required for Puppet 3.8, meaning that as soon as the latest version are available in the CERN repository, the change should be able to be made easily without any major code and configuration changes. This was again done on all the local-boot systems, and a new Net-Booted image was deployed starting in November 2016.

7. Puppet DataBase (PuppetDB)

As Puppet has been advancing, many new features have been added by the developers. One of these was PuppetDB [10] available in Puppet 3.0. PuppetDB collects data generated by Puppet. This includes various facts about the machine on which Puppet is running, resources that are configured during a Puppet run, the latest catalog of every node and up to 14 days of event reports. Previously the SysAdmins used an OpenSource application developed by Puppet Labs called Puppet Dashboard. This application is however no longer maintained and the database was badly designed. For example, the Dashboard MySQL database in ATCN was around 70 GB, with only seven days of reports. PuppetDB on the other hand is currently 8.6 GB using a PostgreSQL database and minimal CPU load. It also enables advanced Puppet features like exported resources, and can be the foundation for other applications that use Puppet's data. This database is parsed, on an hourly basis, via a cron job and python scripts, extracting

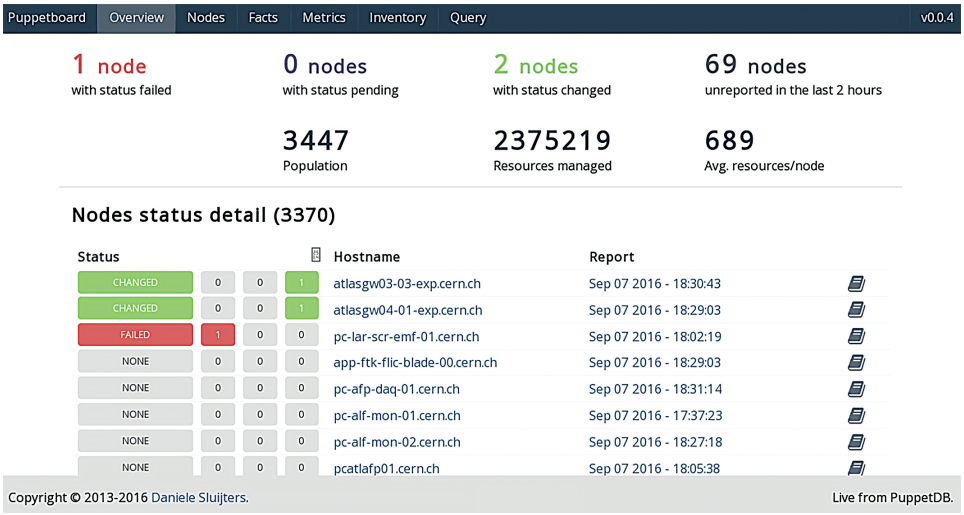

Figure 4. Puppet Board.

information about successful and failed Puppet runs for all systems. The results are tabularised and sent via email, which allows quick and easy analysis of problems. Another use case currently in production is to query the database using known facts or resources to generate a list of machine names, e.g.: all machines that have the *httpd* service running and are on machines from a specific manufacturer. This list can then be used to perform tasks on only those machines. All of this information is also quickly viewable, using an open source GUI called Puppet Board as can be seen in figure 4

8. EXTDATA vs Hiera

When designing Puppet modules or code of any kind, it is good practise to minimise repetition of code. This makes the code easier to maintain, as well as making the modules simpler to use across multiple platforms. This is done by using parameters or variables that are external to the modules. *extlookup* was introduced into Puppet 2.6.0 as a hierarchical method to retrieve this information. *extlookup* searches for a specific fact value, found in a list of files. Precedence for the search order of these files is defined in the sites main configuration file. The downside to EXTDATA is that this information is kept in comma separated data files and structured data is not supported. Searches only return the first match and it can not concatenate a list from the entire hierarchy of data. With Puppet 3.0, Hiera was introduced, a pluggable, hierarchal database that can query YAML, JSON file and Puppet manifests for this information. Hiera data is completely separated from the Puppet code, supports arrays and hashes and allows for dynamic searches based on "facts", which are per-node variables made available to the manifests by Facter [11]. This means it is much easier to design and re-use modules since no hardcoded variables are found in the code, which avoids repetition and allows multiple levels where default data can be over ridden. With Puppet 3.6 currently installed as the default, the SysAdmins have started to migrate to Hiera as a means for separating the data from the Puppet manifests. The use of JSON and YAML files makes it much easier to read and understand what and when data is being used. A full migration to Hiera has not yet been done; it will require a number of changes to existing code by removing the previous *extlookup* functions and variables within the manifests before EXTDATA can be completely dismissed. This is planned for 2017.

9. Conclusion

Puppet in its current state has taken around 6 years to produce a highly complex yet effective system that is capable of maintaining a sane state of ∼4000 machines under the control of the ATLAS TDAQ SysAdmins. The migration to Puppet 3.6 to prepare for CC7 was much easier than the initial migration and changes from SLC5 to SLC6 due to previous experience of the many other migrations, but mostly due to Puppet, which allows properly designed modules to be used on any system. There is however still much to be done, such as the migration from EXTDATA to Hiera, which will allow us to keep all site-specific data out of the manifests, making it much easier to maintain. Something to consider is that while a net-booted system is still much quicker to "reboot", the difference is getting smaller, and a full installation of a new system can be done in around thirty minutes. Local installed systems do not require the complexity of the net-booted infrastructure and has many benefits, such as allowing us to install approved packages that are required, from CERN repositories, without the need to create an entire new net-booted image that has to be deployed across the entire farm. This will allow for much more flexibility and better support of the various sub-detectors.

References

[1] ATLAS Collaboration 2003 *ATLAS high-level trigger, data-acquisition and controls: Technical Design Report* Technical Design Report ATLAS (Geneva: CERN) URL `http://cds.cern.ch/record/616089`

[2] ATLAS Collaboration 2008 *The ATLAS Experiment at the CERN Large Hadron Collider JINST* **3** S08003

[3] Quattor Overview `http://www.quattor.org/documentation/2012/06/19/documentation-overview.html`

[4] Scientific Linux Cern `http://linux.web.cern.ch/linux/scientific.shtml`

[5] PuppetLabs `https://puppetlabs.com/puppet/what-is-puppet`

[6] Chef `http://www.opscode.com/chef/`

[7] CFEngine `http://cfengine.com/product/what-is-cfengine/`

[8] Worldwide LHC Computing GRID `http://wlcg-public.web.cern.ch/about` accessed: 2015-04-16

[9] CERN CentOS 7, `http://linux.web.cern.ch/linux/centos7/`

[10] PuppetDB `https://docs.puppet.com/puppetdb/`

[11] Facter `https://docs.puppet.com/facter/`