Journal of Physics: Conference Series **396** (2012) 042027 doi:10.1088/1742-6596/396/4/042027

Scientific Cluster Deployment and Recovery – Using puppet to simplify cluster management

Val Hendrix[1], Doug Benjamin[2], Yushu Yao[1]

[1]Lawrence Berkeley National Lab, 1 Cyclotron Road, Berkeley, CA, USA 94720
[2]Duke University Durham, NC 27708

Abstract.

Deployment, maintenance and recovery of a scientific cluster, which has complex, specialized services, can be a time consuming task requiring the assistance of Linux system administrators, network engineers as well as domain experts. Universities and small institutions that have a part-time FTE with limited time for and knowledge of the administration of such clusters can be strained by such maintenance tasks.

This current work is the result of an effort to maintain a data analysis cluster (DAC) with minimal effort by a local system administrator. The realized benefit is the scientist, who is the local system administrator, is able to focus on the data analysis instead of the intricacies of managing a cluster. Our work provides a cluster deployment and recovery process (CDRP) based on the puppet configuration engine allowing a part-time FTE to easily deploy and recover entire clusters with minimal effort.

Puppet is a configuration management system (CMS) used widely in computing centers for the automatic management of resources. Domain experts use Puppet's declarative language to define reusable modules for service configuration and deployment.

Our CDRP has three actors: domain experts, a cluster designer and a cluster manager. The domain experts first write the puppet modules for the cluster services. A cluster designer would then define a cluster. This includes the creation of cluster roles, mapping the services to those roles and determining the relationships between the services. Finally, a cluster manager would acquire the resources (machines, networking), enter the cluster input parameters (hostnames, IP addresses) and automatically generate deployment scripts used by puppet to configure it to act as a designated role. In the event of a machine failure, the originally generated deployment scripts along with puppet can be used to easily reconfigure a new machine.

The cluster definition produced in our CDRP is an integral part of automating cluster deployment in a cloud environment. Our future cloud efforts will further build on this work.

1. Introduction

Deployment, maintenance and recovery of a scientific cluster, which has complex, specialized services, can be a time consuming task requiring the assistance of Linux system administrators, network engineers as well as domain experts. Scientific data analysis clusters (DAC) at universities and small institutions often have a part-time FTE with limited knowledge of the administration of such clusters. These cluster managers are often scientists who are wearing an additional system administrator's hat. This extra duty of cluster maintenance diverts the scientist from her main focus, data analysis.

The aim of this current work is to ease the maintenance burden of the scientist turned system administrator. This paper is divided into three parts. Section 2 describes puppet, the

central tool in this process, and it's basic architecture. Section 3 explains our general process of turnkey cluster deployment. The last section, Section 4, walks through a real-world example of an ATLAS [2] Tier 3 analysis cluster deployment.

2. Puppet

The central tool in our cluster deployment and recovery process (CDRP) is an open-source configuration management system (CMS) called Puppet [6]. Puppet is used widely in computing centers for the automatic management of resources. It allows an IT manager to standardize the configuration of resources by going beyond managing system packages for individual systems and providing a way to manage distributed systems. In a distributed system, each node plays a role with services that are interdependent. This is typical of a DAC which usually has services such as data management, job submission/execution and user authentication. Configuring and managing these services requires the system administration to have specialized knowledge of each software package.

Puppet uses a high-level declarative language to create a formal description of the state a system needs to be in. These formal descriptions, called manifests, describe the deployment and configuration of system services. They are packaged in modules and can be deployed in a version control system such as git [4] and subversion [8]. This allows those with specialized knowledge of a particular service to write and publish puppet modules for reuse. In the context of configuring the services of a DAC, these individuals are considered *domain experts*.

Puppet has a client-server architecture. The puppet server contains puppet manifests which describe node configurations where puppet clients may connect, receive and apply their target state.

3. Cluster Deployment & Recovery Process

As we said previously, the object of this work is to allow a scientist (*cluster manager*) to maintain a DAC with minimal effort using our CDRP. Our method, based on the puppet CMS, allows a part-time FTE to easily deploy and recover nodes for entire clusters with minimal effort. This section reviews the components, actors and process steps involved in our CDRP.

3.1. The Components

A *cluster manager* (the scientist) uses a **cluster template** and a **cluster tool** to generate a **cluster definition**. The cluster definition is then used for a turnkey deployment of a DAC(see Fig 1).

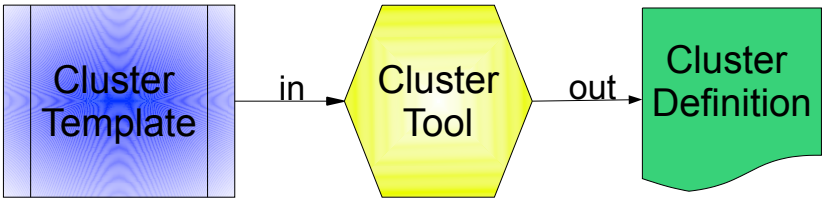

Figure 1: A cluster deployment begins with creating a cluster definition from a cluster template

3.1.1. Cluster Template A **cluster template** is created by a *cluster designer*. It is a set of cluster node roles defined by service definitions and their relationships. It contains

- configuration parameters

- templates
- role definitions

A node *role* is the designated function of a cluster node. It is defined by the services it provides, the services it is a client of and how those services interact. A *service* is managed by a puppet manifest which defines the target state of a node with its declarative language.

3.1.2. Cluster Definition A **cluster definition** is used to deploy a scientific DAC. It consists of a complete puppet server configuration, kickstart install scripts [5] as well as other other configuration scripts. The content of the definition depends on the cluster it defines.

3.1.3. Cluster Tool The **cluster tool** is a script that understands a cluster template. It transforms the cluster template into a cluster definition.

3.2. The Actors
There are several actors in this process from the domain expert who understand the intricacies of the clusters services to the end user who submits analysis jobs on the cluster. Table 1 describes the cluster actors involved with a DAC.

Actor	Definition
Domain Expert	A domain expert writes puppet modules for configuring and managing services used by DACs.
Cluster Designer	A cluster designer creates a cluster template with the puppet modules designed by the domain expert.
Cluster Manager	A cluster manager procures hardware, sets up networking according to the cluster topology. and uses the cluster template to create a cluster definition which is used to configure and deploy a DAC.
Cluster User	A cluster user logs into a deployed cluster and performs a task. In the case of DAC, a user would run analysis jobs.

Table 1: The actors involved with DACs

3.3. The Process
The CDRP for creating a DAC has four main steps (see table 2) and involves three actors: domain expert, cluster designer and cluster manager. In this paper, we will focus on steps 2 and 3 from table 2.

CDRP Steps
1. Write puppet modules
2. Design a cluster template
3. Define a cluster
4. Deploy a cluster

Table 2: Steps for creating and deploying a DAC

Table 2 step 1 of the CDRP starts with the domain experts writing the puppet manifests. The puppet manifests are package in puppet modules and published in a central repository. These puppet modules are used by the cluster designer to create a cluster template.

In table 2 step 2, a cluster designer creates a cluster template. A cluster template has a particular directory structure that is understood by a cluster tool. It contains configuration parameters, templates, node roles and puppet modules. The cluster designer should also provide easy to follow installation instructions for his cluster template.

Once a cluster template has been designed, a cluster manager uses it to create a definition for her local cluster (see table 2 step 3). She first has to set up her cluster hardware then she must edit the cluster configuration parameters by adding specific information about her cluster such as ip addresses and domain names.

The final step in the CDRP (see table 2 step 4) is deploying the cluster. Once the cluster manager has confirmed her cluster configuration, she generates the cluster definition using a cluster tool. The cluster manager then follows the cluster designer's instructions for cluster deployment. These instructions should consist of setting up a puppet server and installing the node operating systems via kickstart installation. Configuration of the nodes is performed automatically upon reboot by the puppet client daemon.

4. ATLAS Tier 3 Analysis Cluster, A Real World Example

An ATLAS[2] Tier-3 [9] data analysis cluster (AT3-DAC) is designed to support the Physics goals of the institutions in which they reside. They typically consist of a batch cluster that allows physicists to perform n-tuple analysis, develop analysis code and run small local test jobs before submitting larger jobs to Tier-1/2 sites via the grid .

4.1. Roles and Services

A typical local batch cluster has up to six different node roles supporting nine major services. Our current work has added one extra cluster service, the puppet server. (see table 3).

Service	Function
condor	job submission/execution
cvmfs	ATLAS software distribution
dnsmasq	small scale dns service
ganglia	distributed monitoring system for HPC clusters
ldap	user authentication
nfs	distributed filesystem
puppet	automated & centralized configuration management
pxe	Preboot eXecution Environment
squid	proxy server and web cache
virt-manager	manage virtual machines

Table 3: Cluster Services.

In our ATLAS Tier 3 (AT3) example, the puppet server is hosted on the HEAD node but a site can choose to add a new PUPPET node role (see table 4). Node roles can be unique meaning that there is only one instance of the node otherwise their are multiple instances. In this example, there are multiple instances of INTERACTIVE and WORKER nodes. All the other nodes are unique. Depending on a site's available physical resources, a node role may be

virtual. Here we have two virtual nodes, LDAP and PROXY, that are hosted by the HEAD node.

Node Role	Servers	Clients	Unique?	Virtual?
HEAD	condor, dnsmasq, puppet, pxe, virt-manager	ganglia, ldap, nfs	Yes	No
NFS	nfs, ganglia	pxe	Yes	No
LDAP	ldap	ganglia, puppet, pxe	Yes	Yes
PROXY	squid	ganglia, puppet, pxe	Yes	Yes
INTERACTIVE	–	condor, cvmfs, ganglia, ldap, puppet, pxe, nfs	No	No
WORKER	–	condor, cvmfs, ganglia, ldap, puppet, pxe, nfs	No	No

Table 4: Cluster Node Roles and Services.

4.2. ATLAS Tier 3 Cluster Tool
ATLAS Tier 3 Cluster tool (`at3c`) [1] is a set of python commands that facilitates configuration and deployment of AT3-DACs. It is a command line tool for creating a *cluster definition* from a pre-existing *cluster template*. A cluster definition is a set of files and templates with a special directory structure. Table 5 defines the directory structure of an AT3-DAC template:

Cluster Template Structure

• `at3c-config.cfg`: A *file* containing parameters for the cluster configuration. Used by `at3c config` sub-command
• `at3c-modules.cfg`: A *file* that is used by the `at3c modules` subcommand. This tells the `modules` sub command where to find the puppet modules for this cluster configuration
• `config`: A *directory* containing templates and files for the cluster configuration. It has three sub-directories: `role`, `module` and `node`.
  – `role`: this *directory* contains the configuration files for the cluster roles, services and their relationships.
  – `module`: this *directory* contains templates and files for puppet module customization
  – `node`: this *directory* contains template files.
• `deploy`: A *directory* where the definition of the puppet server is built by the cluster manager. This configuration is then deployed to the puppet server.

Table 5: The directory structure of an AT3-DAC template

There are three main functions in `at3c`: `config`, `deploy` and `modules`. The `config` command is used to generate configuration files from templates. This is the first step when creating the cluster definition from a cluster template. Before running this command, the cluster

manager should review the configuration parameters in `at3c-config.cfg`. This file resides in the top level of the cluster template directory. The `deploy` command is used to copy the puppet server configuration to the puppet server. The `modules` command uses subversion to check out puppet modules to the puppet server. These puppet modules are defined in `at3c-modules.cfg` which resides in the top level of the cluster template directory structure.

4.3. Installing a new Cluster
The first step in deploying a new AT3-DAC (see CDRP Step 4 in Table 2) is to procure the hardware, reserve the ip addresses/domain names and then set up the network like the graph below. Note that the cluster has public and private networks (see figure 2). The following sections describe the general workflow involved in configuring, deploying and recovering nodes in an AT3-DAC using `at3c`.

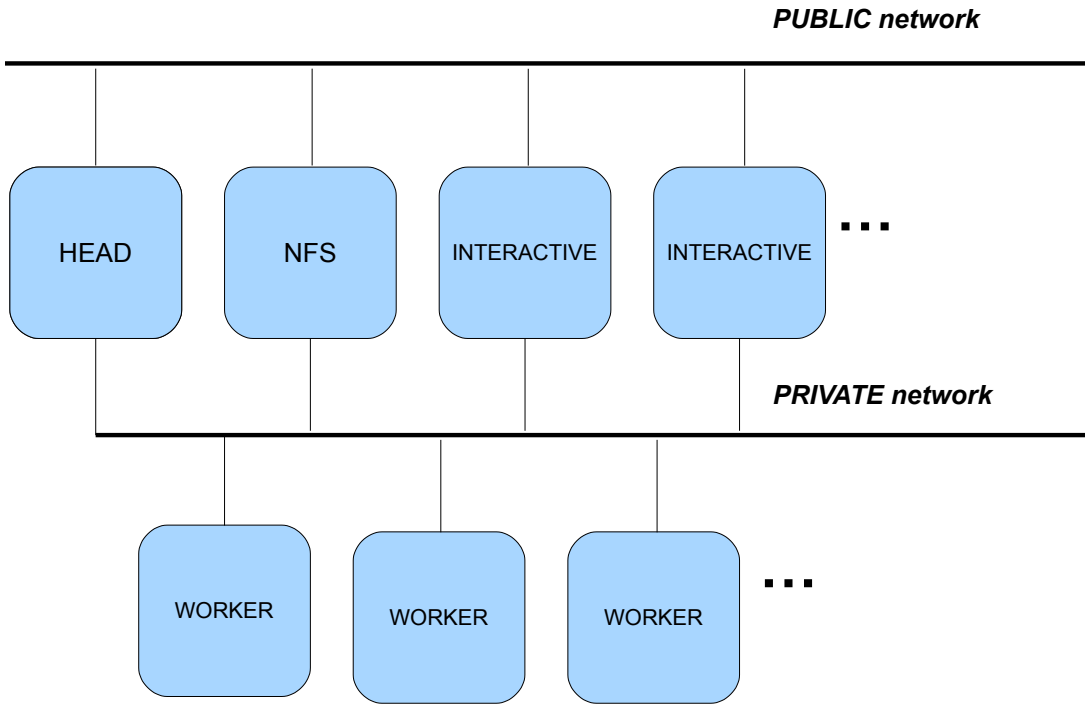

Figure 2: Baseline ATLAS Tier 3 Networking

4.3.1. Create Cluster Definition The cluster definition can be created on any computer that can run python 2.4.1[7]. A cluster manager must first review the required cluster configuration parameters. These parameters are defined by the cluster designer and can be found in

`at3c-config.cfg` at the root of the cluster template directory.

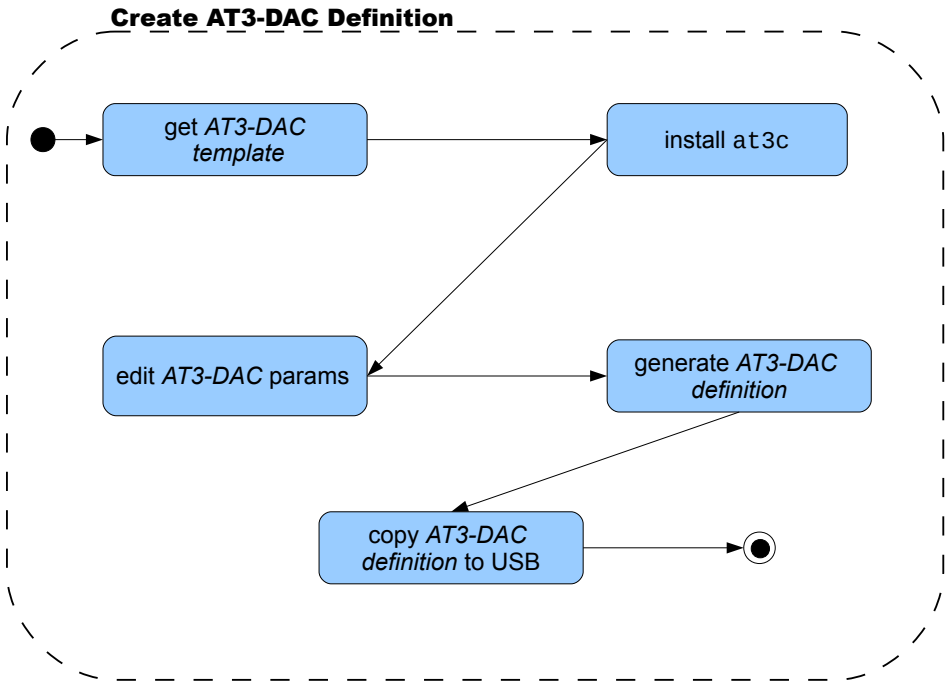

Figure 3: Creating a cluster definition

(i) Check out a cluster template from a remote repository
(ii) Install `at3c` command line tool
(iii) Edit cluster parameters in `at3c-config.cfg`
(iv) Generate the cluster definition using the `at3c config`
(v) Copy the cluster definition to a bootable USB stick

4.3.2. Deploy Cluster Deploying the cluster is a straight forward process for each node. Using the bootable USB stick, the cluster manager installs the operating system via kickstart installation. After the operating system installation is complete, the system is rebooted and the

puppet client daemon configures the system.

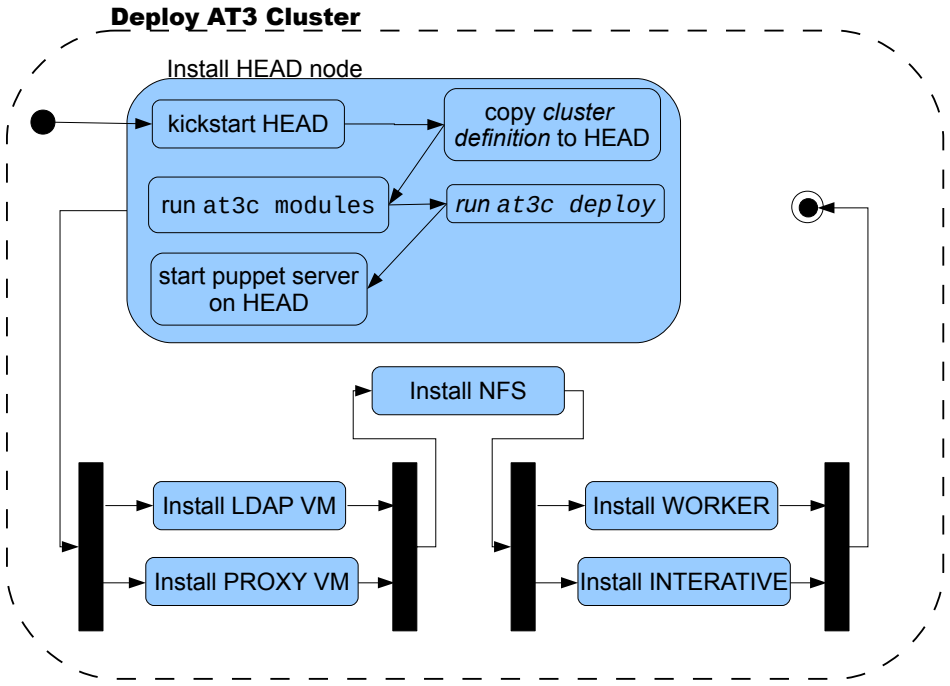

Figure 4: Deploying a cluster

(i) Install HEAD node
  (a) Kickstart installation of HEAD node using USB stick generated by `at3c`
  (b) Copy cluster definition to HEAD node
  (c) Run `at3c modules` to checkout puppet modules from remote repositories
  (d) Run `at3c deploy` to deploy puppet server configuration
  (e) Start the puppet server on the HEAD node

(ii) Install virtual machines in parallel
  (a) Kickstart install PROXY virtual machine (On reboot, puppet will configure the node)
  (b) Kickstart install LDAP virtual machine

(iii) Kickstart install NFS node (On reboot, puppet will configure the node)

(iv) Kickstart WORKER and INTERACTIVE nodes (On reboot, puppet will configure the node)

(v) Start the puppet agent on the head node

4.3.3. Recover a Node The recovery of a failed node is a very simple process. The cluster manager simply needs to replace the failed hardware and perform the same steps she did when deploying the cluster.

(i) Replace failed machine with new hardware

(ii) Kickstart installation of node using USB stick generated with `at3c`

(iii) On reboot, the puppet client will get the configuration from the puppet server

5. Conclusion

In this work, we described a method for deploying a DAC using puppet CMS. Puppet CMS provides an efficient way for cluster managers to maintain the state of nodes in a cluster. This method of cluster deployment and recovery allows part-time FTE system administrators, who are often times a DAC user, to spend less time on cluster management. The ATLAS Tier 3 Cluster tool is an example of how one might automatically generate a cluster definition that facilitates cluster deployment and maintenance. The reduction of time spent doing cluster deployment and maintenance will result in more science results being produced from the DACs.

6. References

[1] http://svnweb.cern.ch/guest/atustier3/at3c/trunk/
[2] http://atlas.web.cern.ch/Atlas/Collaboration/
[3] http://research.cs.wisc.edu/condor/
[4] http://git-scm.com/
[5] http://docs.redhat.com/docs/en-US/Red_Hat_Enterprise_Linux/6/html/Installation_Guide/ch-kickstart2.html
[6] http://puppetlabs.com
[7] http://www.python.org/
[8] http://subversion.apache.org/
[9] http://www.usatlas.bnl.gov/computing/meet/docs/Tier3_Whitepaper.pdf
[10] http://xrootd.slac.stanford.edu/