gLite Configuration and Deployment Models

JRA1 Integration Cluster

www.eu-egee.org

Information Society

Overview

- Deployment modules
- Configuration model
- Deployment models
- Short-term changes
- Some ideas for the future

gLite Deployment Modules

- **The gLite Deployment Modules are essentially meta-packages containing:**
  - A list of dependencies (normally in the form of RPM dependencies)
  - One (sometimes more) configuration file per module
  - One (sometimes more) configuration script per module
- **In the current form:**
  - The configuration file is encoded in XML and has an associated schema
  - The configuration script is in written in python
- **Each module corresponds to some high-level logical functionality: a service (WMS, R-GMA, VOMS, FTS, etc), a client (I/O Client, R-GMA Clients, etc), a group of other modules (UI and WN) or a utility package (gLite Configuration and gLite Security Utilities)**
- **Each service has a version number independent from the gLite distribution version number and the version number is automatically published in R-GMA to be used by service discovery**

Deployment Modules List

- AMGA
- BD-II
- CE
- Configuration
- DGAS Client
- DGAS Server
- DPM Disk Server
- DPM MySQL
- DPM Oracle
- Fireman MySQL
- Fireman Oracle
- File Transfer Service
- File Transfer Agents
- File Transfer Service Client
- GPBox
- I/O Client
- I/O Server
- Job Provenance
- LFC MySQL
- LFC Oracle
- LFC Client
- Logging and Bookkeeping
- R-GMA Server
- R-GMA Client
- R-GMA Servicetool
- R-GMA Gin
- Security Utilities
- Service Discovery
- Stand-alone Metadata Catalog
- Torque Client
- Torque Server
- User Interface
- VOMS Server and Admin MySQL
- VOMS Server and Admin Oracle
- Worker Node
- WMS

The Configuration Files

- **They contain a list of parameters categorized in User, Advanced and System**
- **User parameters in the templates have a value of 'changeme'. All 'changeme' values have to be replaced with real values. Advanced and System parameters have always default values that do not need to be modified in most cases**
- **The files are encoded in XML:**
  - Disadvantages: it's XML, less human readable/writable than key-value pairs
  - Advantages: it's XML, it easily allows structures and hierarchies of data, can be validated for type errors, can be easily transformed to many other formats as needed and used to automatically prepare documentation, web pages, configuration monitoring tools, etc

Configuration File Example

```
<config>

    <mysql.root.password scope="user"
            description="The mysql root password"
            value="verysecret"/>

    <enable.purchasing.from.rgma scope="advanced"
            description="Enable the R-GMA purchaser"
            value="true"/>

    <condor.daemonlist scope="system"
            description="List of the condor daemons to start"
            value="MASTER, SCHEDD, COLLECTOR, NEGOTIATOR"/>

</config>
```

Configuration Model

- **The standard configuration model consists of the following three steps:**
  - Using the configuration templates in $GLITE_LOCATION/etc/config/templates generate the configuration files by replacing any ‘changeme’ value of the User parameters with appropriate values. Optionally modify the default values of the Advanced and System parameters
  - Run the configuration script with the --configure option
  - Run the configuration script with the --start option

Deployment Models

- **There are four main deployment models**
- **Local configuration files**
  - Consists in generating the configuration files from the templates and storing them in $GLITE_LOCATION/etc/config. Overriding values can also be stored in /etc/glite.conf and $HOME/.glite/glite.conf
- **Site configuration files**
  - Consists in generating the configuration files from the templates and storing them on a web server. It is best used together with the XInclude file inclusion mechanism
- **Quattor and other management systems**
  - Consists in using Quattor to generate the configuration files and issue the configure/start/stop commands
- **The Configuration Web Service**
  - The configuration parameter are stored in a database backend, administrators and modules can manipulate the values using a secure web service client

Site Configuration Files

```
<siteconfig xmlns:xi="http://www.w3.org/2001/XInclude">

    <!-- Global parameters -->
    <xi:include href="http://roc.domain.org/glite-global-1.4.1.cfg.xml" />
    <xi:include href="http://site.domain.org/glite-site-1.4.1.cfg.xml" />

    <!-- VO List -->
    <xi:include href="http://roc.domain.org/vo-list-20-10-2005.xml" />

    <!-- A service-->
    <node name="server1.domain.com, server2.domain.com" type="A gLite
          Service Type">
          <xi:include href="glite-service-x.y.z.cfg.xml" />
    </node>

    <!-- Another service-->
    <node name="server3.domain.com" type="Another gLite Service Type">
          <xi:include href="glite-another-service-x.y.z.cfg.xml" />
    </node>

</siteconfig>
```

VO List

```
<siteconfig xmlns:xi="http://www.w3.org/2001/XInclude">

    <vo name="EGEE">
        <voms.vo.name scope="user"
                description="The VO name"
                value="EGEE">
        <voms.vomsnode scope="user"
                description="The VOMS Server hosting this VO"
                value="kuiken.nikhef.nl">
    </vo>

    <vo name="dteam">
        <voms.vo.name scope="user"
                description="The VO name"
                value="dteam">
        <voms.vomsnode scope="user"
                description="The VOMS Server hosting this VO"
                value="voms.cern.ch">
    </vo>

</siteconfig>
```

VO List Override

- **Sites or individual services may want to override a VO list**
- **There are two ways**
  - Define a custom VO List and use it instead of the complete one
  - Use the <volist> tag to include or exclude VOs

```
<volist name="myVOList1">
   <include name="EGEE">
</volist>

<volist name="myVOList2">
   <exclude name="EGEE">
</volist>
```

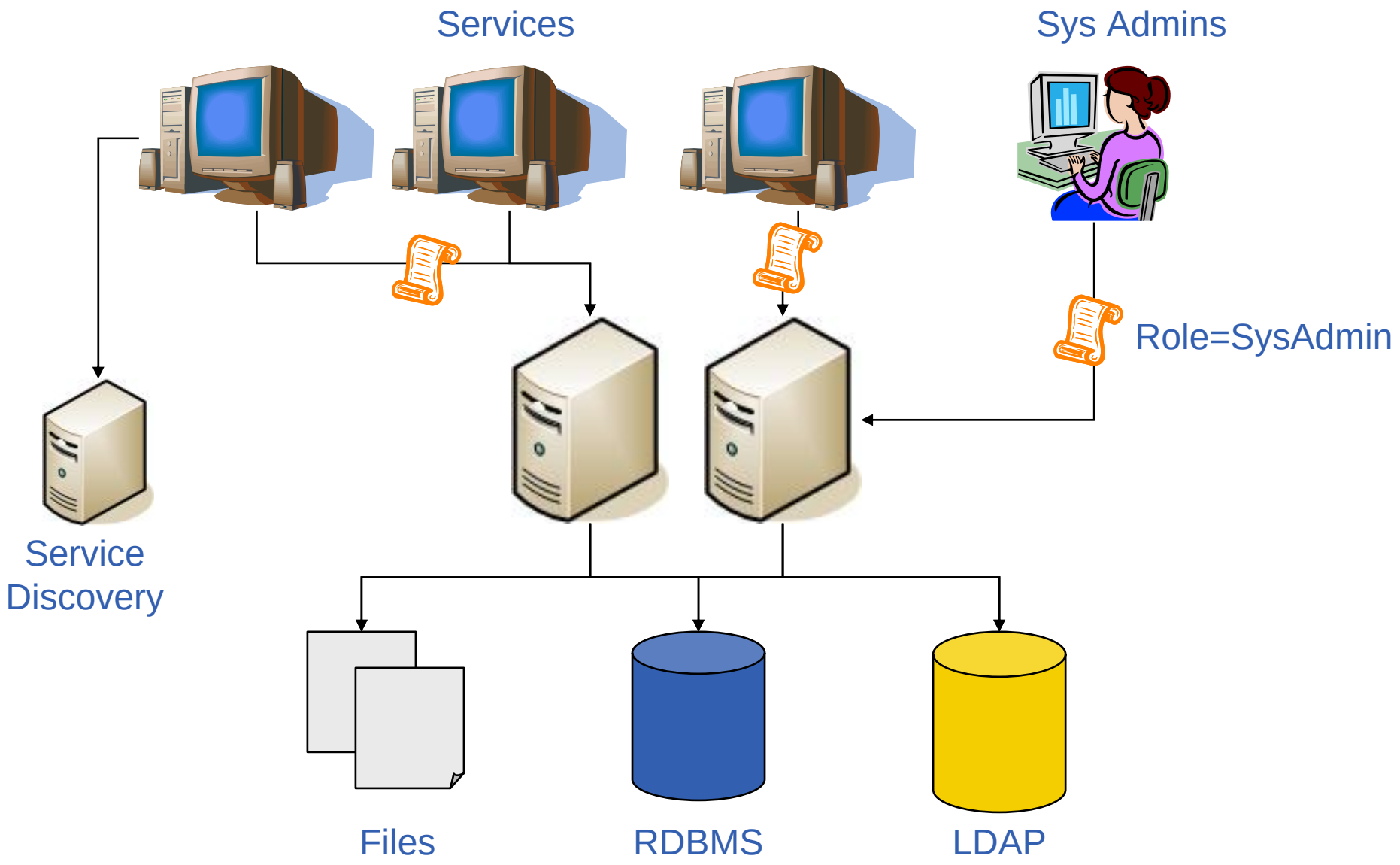
Example

http://glite.web.cern.ch/glite/siteconfig/default.asp

Quattor

- Since the start of gLite being able to integrate with high-level management systems has been a must
- We have started with Quattor
- The Quattor templates for each deployment module are automatically generated during the build process from the known dependencies list
- The gLite configuration files are transformed into pan format using an XSLT script. This also can be included in the build
- An NCM component that creates the gLite configuration files and triggers the configuration scripts is being finalized.
- This is being progressively used to deploy gLite in the JRA1 testbed known as the Prototype

Configuration Web Service



Changes in 1.5.0

- **The new VO management model and the VO lists will be introduced**
- **The servicetool instances in the services configuration files will disappear from the standard templates. They will still be usable for advanced tuning, but the configuration scripts will set all default values**
- **The file-based service discovery instances in the client will disappear from the standard templates. They will still be usable for advanced tuning ro troubleshooting, but rgma or bdii will be the recommended methods**
- **Some highly requested changes like:**
  - the possibility of having well-formatted subfiles in the site configuration model
  - Support for different versions of the python XML libraries (for example for running on Fedora)

More short-term changes

- **Integration of the current configuration model with YAIM**
  - Quickest option: YAIM can be used in the same way as Quattor is used to generate the XML files and replace the changeme values with user-defined values ans then run the python script
  - Another option: Replace one system with the other
  - Best option: leverage the functionality and advantages of both systems to define a new common method. It may take a while, requires more changes in all components

Ideas for the future

- **Look what else is done around and see if we can use it: WSDM, CDDLM, CIM, etc**
- **Clearly separate configuration and deployment**
  - Configuration is about providing information
  - Deployment is about using information to perform actions
- **They are currently mixed. The scripts contains the information and act on it. They are not self descriptive**
- **As done with the configuration parameters, the configuration actions should be described in a standard way**
- **Configuration steps information should be self-descriptive and it should be possible to:**
  - directly generate documentation from it
  - detect deployment conflicts before deploying
  - avoid duplication of actions

Example

```
<checks>

    <file path="/file.path" perms="0755" user="root" group="root">
            <action name="create" value="true"/>
    </file>
    <dir path="/dir.path" perms="0755" user="root" group="root"/>
            <action name="create" value="true"/>
    </dir>

    <user name="wmsuser" group="gm">
            <action name="create" value="true"/>
    </user>

    <hostcerts>
            <certificate path="/etc/grid-security/hostcert.pem"/>
            <key path="/etc/grid-security/hostkey.pem"/>
    </hostcerts>

    <X509_CERT_DIR path="/etc/grid-security">
            <action name="define" value="false"/>
    </X509_CERT_DIR>

    <param name="SandBoxDir" value="/tmp/glite">
            <action name="set" file="$GLITE_LOCATION/etc/config/glite-wms.cfg.xml"
                    format="xml" xpath="/parameters">
            <action name="set" file="$GLITE_LOCATION/etc/glite_wms.conf"
                    format="classad" section="workload_manager">
    </param>

</checks>
```

Ideas for the future

- This model is language-independent and portable
- In any case the current models do not scale: integrators and deployment teams cannot keep providing and maintaining scripts for each new or changed component, port it to different platforms, etc
- The essential configuration information is with the developers, they should maintain it and deliver the standard config files (parameters and list of actions) with the components
- Integrators should use the information to create basic validated packages removing conflicts and duplications
- Deployment people should instantiate the information to the specific environment they have to support