Software deployment and service administration with Quattor

Quattor @ LCG T2 workshop, 16/6/06

German Cancio

CERN/IT

Outline

Software Deployment

Service Configuration

Outline

Software Deployment:

- SWRep and SPMA overview
- SWRep/SPMA vs. APT/yum

Service Configuration

SWRep (Software Repository)

- Universal repository for storing Software:
  - Extendable to multiple platforms and packagers
    (RH Linux RPM.. Solaris PKG, others like Debian pkg)
  - Stores multiple package versions/releases per platform
- Management ("product maintainers") interface:
  - ACL based mechanism to grant/deny modification rights
    (packages associated to "areas")
  - Uses SOAP since Quattor 1.2
- Client access: via standard protocols
  - **HTTP**, AFS/NFS, FTP
- Replication for load balancing/redundancy: using standard tools
  - Locally: Apache mod_proxy ; squid (as used at CERN)
  - Remotely: Rsync (distributed T2's)

SPMA (Software Package Manager Agent)

- Can manage either *all* or a *subset* of packages on the nodes
  - On production nodes: *full control* - wipe out unknown packages, (re)install missing ones.
  - On development nodes: non-intrusive, configurable management of system and security updates.
- Package *manager*, not only *upgrader*
  - Can roll back package versions
  - Transactional verification of operations
- Scalability:
  - Supports HTTP (also FTP, AFS/NFS) and forward/reverse proxies
  - Time smearing and package pre-caching
- Portability via generic plug-in framework
  - System packager specific transactional interface (RPMT, PKGT)
- Multiple repositories can be accessed (eg. division/experiment specific)

SPMA (II)

- SPMA functionality:
  1. Compares the packages currently installed on the local node with the packages listed in the configuration profile
  2. Computes the necessary install/deinstall/upgrade operations
  3. Invokes the packager (rpmt/pkgt) with the right operation transaction set

- The SPM is driven via a local configuration file
  - An NCM *component* (ncm-spma) generates/updates this configuration file out of CDB information
  - `/var/lib/spma-target.cf`

SPMA (III)

RPMT

- RPMT (RPM transactions) is a small tool on top of the RPM libraries, which allows for multiple simultaneous package operations resolving dependencies (unlike plain RPM)
  - Example: ‘upgrade X, deinstall Y, downgrade Z, install T’ and verify/resolve appropriate dependencies
- Does use basic RPM library calls, no added intelligence

SPMA/SWRep vs. other package management tools

- Quattor does not impose using any packaging tool.
  - **SPMA**
  - **APT** (and **yum**, not described here as very similar to APT)
- SPMA advantages:
  - **Declarative**: Keep list of packages in CDB templates. Packages are updated/downgraded as required.
    - SPMA behaves stateless.
  - Every node may have a completely **different setup** in terms of installed packages and versions.
    - Not "go for the latest"
  - Explicit **separation of Package depot** (SWRep) and **configuration** (in CDB)
    - You can have multiple versions (old, production, new, beta) of packages in the repository without causing client updates
  - SPMA supports **rollbacks**. Just change the packages/versions in CDB and SPMA will take care to downgrade/upgrade/install/remove whatever is required.
  - SPMA supports multiple **simultaneous package versions** on one node.

SPMA/SWRep vs. other package management tools

- APT advantages:
  - **Standard tool** shipped with Scientific Linux
    - LCG and Quattor offer APT/yum repositories for downloading software
  - **Dependency resolution**
    - But may decide to resolve differently than what you want! Eg. More than one package 'providing' a dependency.
    - Requires to set 'priorities' on repositories.
  - Nice **GUIs** (synaptic), easier to use.
- APT and SPMA/SWRep should not be used in parallel on the same node.
  - However, it is possible to bootstrap the Quattor server with APT, and then use SPMA to manage your farms and servers
- The choice between SPMA and APT/yum will depend on the complexity of your environment and/or the control level you need.
  - CERN example: SPMA for Computer Centre ("full" mode for batch nodes, "light" mode for development nodes)
  - APT for desktops

SW configuration in CDB with SPMA

Some PAN functions are provided for **manipulating the package list in the profile**, which are used in the templates:

- `pkg_add("packagename",["version-release","arch"]);`
  - Adds a package to the profile (version-release and arch are optional)
- `pkg_del("packagename",["version-release"]);`
  - Remov
- `pkg_repl`
  - Replac not sp
- It is impo functions
  - Eg. 'pk from t
  - Useful

```
template pro_software_packages_i386_sl3;
…
# take defaults - no specific version-release
"/software/packages" = pkg_add("4Suite");
"/software/packages" = pkg_add("ElectricFence");
"/software/packages" = pkg_add("GConf2");
"/software/packages" = pkg_add("GConf2-devel");
"/software/packages" = pkg_add("ImageMagick");
"/software/packages" = pkg_add("Omni");
"/software/packages" = pkg_add("Omni-foomatic");
"/software/packages" = pkg_add("PyXML");
"/software/packages" = pkg_add("SDL");
"/software/packages" = pkg_add("SDL-devel");
"/software/packages" = pkg_add("SysVinit");
…

# specific version requested
"/software/packages"=pkg_add("mypackage","1.10.15-1","noarch");
"/software/packages"=pkg_add("other_package","0.0.14-1","i386");

# remove an "inherited" package
"/software/packages"=pkg_del("somepkg");
```

Running SPMA

- The SPMA configuration is generated/updated by running an NCM component
  - `ncm-ncd --configure spma` : *updates* `/var/lib/spma-target.cf` *and* `/etc/spma.conf` *if needed*
  - The NCM component updates the SPMA config files, and can
    autom
- SPMA is
  - `# spm`
- Most imp
  - **`--noa`**
  - `--ver`
  - `--use`
  - `--use`

```
# spma

[INFO] SPMA version 1.10.10 started by root at: Fri May  5 13:30:03 2006
[INFO] using local package cache in: /var/spma-cache/
[INFO] proxy server activated, type: reverse
       proxy server(s): lxc1m991
[INFO] active proxy found: lxc1m991
[INFO] examining local installations..
[INFO] reading target configuration ..
[INFO] executing operations..
[INFO] The following package operations are required:
       replace  - php 4.3.2 26.ent i386  with
       http://swrep/swrep/i386_slc3/ php 4.3.2 30.ent i386
       replace  - php-mysql 4.3.2 26.ent i386  with
       http://swrep/swrep/i386_slc3/ php-mysql 4.3.2 30.ent i386
       replace  - php-oci8 4.3.2 26.ent i386  with
       http://swrep/swrep/i386_slc3/ php-oci8 4.3.2 30.ent i386
[INFO] Please be patient... 3 operation(s) to verify/execute.
[OK]   SPMA finished successfully.
```

- The **typical sequence** of operation is:
  1. *update templates in CDB via* `cdbop`
  2. *run* `ncm-ncd --configure spma`

Outline

Software Deployment

Service Configuration:

- NCM (Node Configuration Manager) overview
- Some example components

What are components? (1/2)

- “Components” (like SUE “features” or LCFG ‘objects’) are responsible for updating local config files, and notifying services if needed
- Components do only *configure* the system (unlike LCFG!)
  - Usually, this implies regenerating and/or updating local config files (eg. `/etc/sshd_config`)
- Use standard system facilities (SysV scripts) for *managing* services
  - Components can notify services using SysV scripts when their configuration changes.
- Components can be run
  - Manually (via `ncm-ncd`)
  - via hooks (cron, boot time, etc)
  - automatically*:* register their interest in configuration entries or subtrees, and get invoked in case of changes (via `ncm-cdispd`)
- Possible to define configuration dependencies between components
  - Eg. configure *SPMA* before *GRUB*
  - Components won’t run if a pre-dependency is *unsatisfied* (eg. failing prerequisite component)

What are components? (2/2)

- Components are written as Perl OO class instances
  - But don't worry, no OO knowledge needed for writing them, just some Perl.
- Each component is packaged as an individual RPM.

- Each component can provide two methods:
- **`Configure():`**
  - invoked when there was a CDB configuration change or on startup
  - *Mandatory method*
- **`Unconfigure():`**
  - invoked when a component is to be removed
  - *Optional method* – most of the components don't need to implement it.

Component (simplified) example

```
sub Configure {
  my ($self,$config) = @_;
  # 1. access configuration information
  my $arch=$config->getValue('/system/architecture'); # NVA API
  $self->Fail ("not supported") unless ($arch eq 'i386');
  # 2. (re)generate and/or update local config file(s)
  open (myconfig,'/etc/myconfig'); …
  # 3. notify affected (SysV) services if required
  if ($changed) {
    system('/sbin/service myservice reload'); …
  }
}
```

Existing components

> 100 NCM configuration components are available:

- Configure basic Quattor and core system services
  - Quattor services: **ccm, spma, cdp**
  - System services: **accesscontrol, accounts, autofs, cron, filecopy, grub, iptables, ldconf, lmsensors, logrotate, mailaliases, network, netdriver, nfs, ntpd, portmap, profile, serialclient, smartd, ssh, sysctl**
- Configure advanced system services
  - Including **castor, chkconfig, fiberchannel, gdmconf, ipmi, lsfclient, named, quota, screensaver, sysacct**
  - These would need more testing outside CERN
- Configure Grid services
  - **bdiicfg, ceinfo, cliconfig, cmnconfig, condorconfig, edglcg, gip, globuscfg, gridmapdir, guiconfig, infoproviders, lbconfig, lcas, lcgbdii, lcginfo, lcmaps, mkgridmap, myproxy, pbsclient, rgmaproducer, rm, uicmnconfig, wlconfig, yaim**

Components and CDB configuration

- Components can have "private" configuration entries, including:

```
/software/components/<name>/active (bool)                  <- component active?
                            dispatch (bool)                <- run automatically via cdispd?
                            dependencies/pre  (string[])   <- run components before
                            dependencies/post  (string[])  <- run components after
                            foo/...                           (component specific)
                            bar/...                           (component specific)
```

- Components can access configuration information *anywhere* in the node profile `(/system/.., /software/.., /hardware/..)`
  - Useful to share common configuration entries between components
  - Eg. `/system/kernel/version`

- All components need to declare their "private" config data types, and can define default values

```
pro_declaration_component_<component>.tpl    <- structure
pro_software_component_<component>.tpl       <- default values
```

Example components (I)

ncm-grub:

- Functionality
  - configures the GRUB boot loader.
  - Uses the ‘grubby’ command line tool.
  - Won’t change grub config if inconsistencies found.
- Most important config parameters:

`/system/kernel/version` *(string): kernel version to be used.*

- More info:
  - `man ncm-grub`

Example components (II)

ncm-cron:

- Functionality
  - Adds/removes cron entries.
  - Places them under /etc/cron.d with a log file in /var/log.
  - Respects existing cron.d entries.
- Most important config parameters:

```
/software/components/cron/entries/list/name  (string) cron entry name (eg.
                                                                  "example")
                                       user  (string) user (eg. "root")
                                       frequency  (string) eg. "* 1 * * *"
                                       command  (string) "/bin/myexec"
```

- More info:
  - `man ncm-cron`

Example components (III)

<u>ncm-accounts:</u>

- Functionality
  - Controls the /etc/passwd, /etc/group, (/etc/shadow) files.
  -

- Most important config parameters:

```
/software/components/accounts/rootpwd    (string) crypted root password.
                                shadowpwd  (boolean) use /etc/shadow.
For every user:
/software/components/accounts/users/<user>/comment  (string) comment field
                                       <user>/uid  (string) groups it belongs to
                                       <user>/passsword  (str) crypted password
                                       <user>/createHome  (bool) make homedir?
```

- More info:
  - `man ncm-accounts`

How to run components? (I)

Manually:

- `ncm-ncd` (Node Configuration Deployer):
  - framework a
    or manually)
  - dependency
  - Invoke it (or

```
# ncm-ncd -config
# ncm-ncd --confi

# ncm-ncd --uncon
# ncm-ncd --list
```

  - You should r
  - A logfile dire

```
# /var/log/ncm/nc
# /var/log/ncm/co
```

```
# ncm-ncd --configure grub
```

```
# ncm-ncd --list

active components found inside profile /software/components:
name            file?  predeps                        postdeps
-------------------------------------------------------------------
libterm:        yes
lbclient:       yes
zephyrclt:      yes
ssh:            yes
afsclt:         yes
spma:           yes
cdp:            yes
consoleclient:  yes
sendmail:       yes
sysctl:         yes
access_control:yes
ntpd:           yes
krb5clt:        yes
regisclient:    yes
smartd:         yes
grub:           yes
lsfclient:      yes    chkconfig
quota:          yes
srvtab:         yes    afsclt
rpmverify:      yes
snmp:           yes
cron:           yes
chkconfig:      yes
fmonagent:      yes
```

How to run components? (II)

Automatically (default!):

- ncm-cdispd (Configuration Dispatch Daemon)
  - Monitors the config profile, and invokes registered components via ncm-ncd if there were changes
  - Looks up for changes for every component in the following entry:

```
/software/components/<component>/...
```

  - Additional entries to watch can be configured (eg. /system/kernel/version for the grub component)
  - All operations are logged:

```
/var/log/ncm-cdispd.log
```

ncm-query

- Use **ncm-query** for to visualize component configuration information on the target node:

```
# ncm-query --component <component>
```

```
# ncm-query -dump /path/in/configuration/tree
```

```
# ncm-query --component spma

[INFO] Subtree: /software/components/spma
+-spma
  $ active : (boolean) 'true'
  $ headnode : (boolean) 'true'
  $ proxy : (string) 'yes'


# ncm-query --dump /system/kernel/version

[INFO] Subtree: /system/kernel/version
$ version : (string) '2.4.21-40.EL.cernsmp'
```

quattor

http://quattor.org