

## DIT's clusters management with AutoYaST

HPC-CH meeting, Lausanne, 20/05/2010

---

Jacques.Menu@epfl.ch

<http://hpc-dit.epfl.ch>

May 20, 2010

YaST (*Yet another Setup Tool*) is the management tool provided by SuSE Linux, be it openSUSE or SuSE Linux Enterprise Server (SLES). SuSE was bought by Novell some years ago.

SLES 10 is used on DIT's general purpose clusters and the BlueGene/P's front-ends, at level SP1 or SP2 depending on the machine.

AutoYaST complements YaST to handle the automatic installation of bunches of machines.

We present below how we use AutoYaST to handle the Callisto, Antares & Vega clusters at DIT. The configurations samples are taken from Vega.

## Contents

<b>1</b>	<b>Documentation and versions</b>	<b>4</b>
<b>2</b>	<b>YaST possibilities</b>	<b>4</b>
2.1	Novell registration . . . . .	7
2.2	System patch . . . . .	7
2.3	Installation of software supplied by Novell . . . . .	7
<b>3</b>	<b>Installation server</b>	<b>8</b>
<b>4</b>	<b>Compute nodes installation strategy</b>	<b>8</b>
<b>5</b>	<b>Compute nodes boot order</b>	<b>9</b>
<b>6</b>	<b>PXE boot configuration files</b>	<b>10</b>
<b>7</b>	<b>AutoYaST control files</b>	<b>15</b>
<b>8</b>	<b>Node-specific settings</b>	<b>19</b>
8.1	“chroot-scripts” section . . . . .	21
8.2	“init-scripts” section . . . . .	21
8.3	Specific software installation scripts . . . . .	24
8.4	SSH matters . . . . .	25
<b>9</b>	<b>Conclusion</b>	<b>27</b>

## List of Figures

1	"YaST" screen	5
2	"YaST2" screen	6
3	PXE configuration file	11
4	PXE message file	12
5	Control file structure	14
6	Bootloader section	17
7	Network interfaces section	18
8	"postinstall.sh"	20
9	"chroot-scripts"	22
10	"init-scripts"	23
11	"postinstall_20_Atlas"	24
12	"postinstall_01_NodeConfig"	25

## 1 Documentation and versions

The documentation is available online at URL:

<http://chorgan.provo.novell.com/YaST/doc/SLES10/autoinstall/index.html>

as well as on the machines in file `/usr/share/doc/packages/autoYaST2/autoYaST.pdf`.

YaST exists in semi-graphic and graphic modes, launched with “`yast`” and “`yast2`” respectively. Both offer the same possibilities, displayed in “groups”, as depicted by figures 1 and 2.

The version of YaST shown here is the one available on SLES 10. There are user interface variations on other versions.

One can manage the various groups without displaying all of them first, thru an argument to the command.

Thus, “`/sbin/yast2 hwinfo`” leads directly to the “Hardware” group window.

## 2 YaST possibilities

The basic features are similar to the ones provided by equivalent tools provided on distributions such as Mandriva, Red Hat and Ubuntu for example.

One can supply a string in the “*Filter*” input box to control the amount of information displayed in the right pane of the window.

In the “*Hardware*” group, all the information about the hardware of the machine can be accessed, with vendors, models, serial numbers and versions.



Figure 1: “YaST” screen



Figure 2: "YaST2" screen

Under “*Security and Users*”, one can manage users and groups, define the constraints on passwords such as length and validity limit. That's where the firewall can be setup, too, and certificates managed.

## 2.1 Novell registration

An *activation code* is to be purchased from a Novell partner to access the patches and software evolutions. This button allows one to register the activation key.

## 2.2 System patch

The button “*Patch CD Update*” copies the available patches on the machine and installs them. One can compare the currently installed versions with the ones on Novell servers, as well as their dependencies. The files belonging to the installed versions are also shown.

In menu “*Extras*”, the command “*Show products*” displays the various service packs available. Vega relies on DIT's SAN infrastucture, that limits it to SP2 at the time of writing, but migration to SP3 is proposed, for example.

## 2.3 Installation of software supplied by Novell

That's done with button “*Software Management*” in group “*Software*”. The dependencies are taken into account, which may lead to a choice by the user in certain circumstances.

### 3 Installation server

One should setup how to access so-called *software catalogs*: they're available on CD, DVD and from FTP, HTTP, HTTPS or SMB/CIFS file servers.

These software catalogs are handled thru the “*Installation Source*” button in the “*Software*” group.

The activation code is used to create an account on Novell servers, which is accessible thru a URL such as

```
https://5e2705k97ce548ckb7a104e022760b75@nu.novell.com//repo/$RCE/SLES10-SP2-Updates/sles-10-x86_64/
```

The “*Installation Into Directory*” button in group “*Software*” is used to create a local repository. One chooses the installation source (a Novell server or a locally mounted DVD, for example) and the destination directory.

Vega is an HTTPS client of “SLES10-SP2-Pool” and “SLES10-SP2-Updates” on Novell servers, and it accesses “/exports/cluster/suse/SLES-10-SP2/CD1/” on its own disks.

### 4 Compute nodes installation strategy

A compute node is installed when the machine is first set to production work, and afterwards in case serious problems are encountered on the node.

The installation time ranges from 15 to 25 minutes, depending on the particular machine and the size of the software to be installed.

We prefer not to clone an existing “standard” compute node. We thus get a freshly formatted disk with all the software installed from the reference configuration, without anything left over from history.



We use:

- a configuration server, accessible thru NFS on the same subnet as the compute nodes. In practice, the machine's front-end is used;
- so-called XML "*control files*": they describe in a structured way how to install the nodes;
- a reference directory on the front-end, from which files are copied over to the nodes after basic software installation.

In the case of Vega, the installation server is accessed thru "nfs://172.30.1.1/exports/cluster/suse/SLES-10-SP2/CD1". This directory contains the image of the first installation CD.

The sub-directory "*suse/x86\_64*" contains the ".rpm" packages for the nodes' architecture, as well as "*MD5SUMS*", the MD5 checksum pour each of them.

## 5 Compute nodes boot order

The nodes of Vega were supplied by Dell. The boot sequence is:

1. local DVD;
2. NIC1 interface;
3. NIC2 interface ;
4. internal hard disk.

When no DVD is present, they boot via PXE and fetch their configuration from the front-end.

There are then two cases:

Node marked as...	Action
<code>boot</code>	Boot from internal hard disk, nothing special
<code>install</code>	Fetch the installation configuration from the front-end thru TFTP, and launch the installation process

## 6 PXE boot configuration files

These files are named after the IP address of the corresponding node, written in hexadecimal without dots.

The “`gethostip`” comes in handy:

```
root@node22:~ > /usr/bin/gethostip node23
node23 172.30.2.23 AC1E0217
```

```
root@vega:/opt/cluster/tftpboot/pxelinux.cfg > cat AC1E0217
DEFAULT install
PROMPT 1
TIMEOUT 20
DISPLAY message/AC1E0217

LABEL boot
    LOCALBOOT 0

LABEL install
    KERNEL linux
    APPEND initrd=initrd ~>
    ~> textmode=1 splash=0 showopts netdevice=eth0 hostip=172.30.2.23 ~>
    ~> netmask=255.255.0.0 gateway=172.30.1.1 netwait=120 ~>
    ~> install=nfs://172.30.1.1/exports/cluster/suse/SLES-10-SP2/CD1 ~>
    ~> autoYaST=nfs://172.30.1.1/exports/cluster/autoYaST/node23.xml

LABEL rescue
    KERNEL linux
    APPEND initrd=initrd splash=0 rescue=1 showopts ~>
    ~> install=nfs://172.30.1.1/exports/cluster/suse/SLES-10-SP2/CD1

LABEL memtest
    kernel memdisk
    append initrd=dalco/mem165.img
```

Figure 3: PXE configuration file

```
root@vega:/opt/cluster/tftpboot/message > cat AC1E0217
VEGA boot menu for node23 [AC1E0217, 172.30.2.23]
```

```
-----
Possible actions:
```


```
boot      - boot the node from its internal hard disk
install   - install the node, scratching everything
rescue    - boot in rescue mode
memtest   - run memtest on the node
```

```
Default: install
```

```
-----
Enter an action in the next 20 seconds (default: install)
```

Figure 4: PXE message file

Vega' node number 23 has address "172.30.2.23". Its configuration file is shown at figure 3.

 The "netwait" option is needed on Vega, to let the IP stack start before trying to access the NFS server.

The message at boot time is show at figure 4.

These configuration and message files are created by a script whose argument is the default action, i.e "install" or "boot".

After the installation is performed, the action for the given node is set back to "boot" for the forthcoming reboots. This is done by re-creating the files above with "boot" as the argument.

An alternative would be to use the following option, which we don't use:

```
<second_stage config:type="boolean">false<second_stage>
```

```
<?xml version="1.0"?>
<!DOCTYPE profile>

<profile xmlns="http://www.suse.com/1.0/YaST2ns" xmlns:config="http://www.suse.com/1.0/configns">

  <bootloader>
    <device_map config:type="list">
      <device_map_entry>
        <firmware>hd0</firmware>
        <linux>/dev/sda</linux>
      </device_map_entry>
    </device_map>

    <global>
      <activate>true</activate>
      <boot_root>true</boot_root>
      <default>SUSE Linux Enterprise Server 10 SP2</default>
      <generic_mbr>true</generic_mbr>
      <gfxmenu>/boot/message</gfxmenu>
      <lines_cache_id>2</lines_cache_id>
      <timeout config:type="integer">8</timeout>
    </global>
    ... ..
  </bootloader>
  ... ..
</profile>
```

Figure 5: Control file structure

## 7 AutoYaST control files

These files, such as “`node23.xml`”, are essentially the same for all compute nodes. The only differences are the hostname “`node23`” and the IP address “`172.30.2.23`”.

They have the structure shown at figure 5.

These files are generated by a script too. One can check they are well-formed with the “`xmllint`” utility, that is part of package “`libxml2`”.

The model they're synthesized from can be obtained initially on a ready to use node.

The user interface to do this is not over-intuitive, however:

- in YaST, choose command “*Create Reference Profile*” from menu “*Tools*”;
- this brings up a window in which additional resources can be selected, such as NFS client and NTP configuration;
- button “*Create*” created the XML code describing the operating system and the additional resources selected. In case one selects a not yet installed resource, YaST proposes to install it on the fly;
- in menu “*View*”, the command “*Source*” displays this XML code;
- one still has to save this code to a file thru command “*Save as...*” in menu “*File*”, with directory “`/var/lib/autoinstall/r`” proposed by default.

An alternative is to choose, in menu “*Tools*”, the command “*Check Validity of Profile*”: the effect is to create a control file in “`/tmp`”, as:

```
root@vega:/var/cache/zmd > ll /tmp/YaST2-31366-dkPnh9/valid.xml
148 -rw-r--r-- 1 root root 147424 May  5 16:50 /tmp/YaST2-31366-dkPnh9/valid.xml
```

and to test it with “`xmllint`”.

One can then use this file at will.

The available **bootloader options** are shown at figure 6.

The “*Network*” section contains DNS and routing information among others.  
The **interfaces settings** are presented at figure 7.



```
<loader_type>grub</loader_type>
<sections config:type="list">
  <section>
    <append>resume=/dev/sda2 splash=silent showopts</append>
    <image>/boot/vmlinuz-2.6.16.60-0.21-smp</image>
    <initial>1</initial>
    <initrd>/boot/initrd-2.6.16.60-0.21-smp</initrd>
    <kernel>/boot/vmlinuz</kernel>
    <lines_cache_id>0</lines_cache_id>
    <name> SUSE Linux Enterprise Server 10 SP2</name>
    <original_name>linux</original_name>
    <root>/dev/sda1</root>
    <type>image</type>
    <vgamode>0x317</vgamode>
  </section>
  <section>
    ... ..
    <name> Failsafe -- SUSE Linux Enterprise Server 10 SP2</name>
    <original_name>failsafe</original_name>
    ... ..
  </section>
</sections>
```

Figure 6: Bootloader section

```
<networking>
  <dhcp_options>
    <dhclient_additional_options></dhclient_additional_options>
    <dhclient_client_id></dhclient_client_id>
    <dhclient_hostname_option>AUTO</dhclient_hostname_option>
  </dhcp_options>

  <interfaces config:type="list">
    <interface>
      <bootproto>static</bootproto>
      <device> eth0</device>
      <ipaddr>172.30.2.23</ipaddr>
      <name>Broadcom NetXtreme II BCM5708 Gigabit Ethernet</name>
      <netmask>255.255.0.0</netmask>
      <startmode>auto</startmode>
      <usercontrol>no</usercontrol>
    </interface>
  </interfaces>
  ... ..
</networking>
```

Figure 7: Network interfaces section

## 8 Node-specific settings

AutoYaST offers means to execute scripts to do such settings, at four different stages of the installation process:

Scripts	Executed...
pre-scripts	At the very beginning, right after hardware detection, but before installation actually starts
chroot-scripts	Before the just-installed system is rebooted
post-scripts	Right after this first reboot, before the operating system services are started
init-scripts	After these services have started

We use scripts to **fine-tune** the compute nodes **just before and just after** the first reboot of the newly-installed system.

They're in charge of installing all complementary software beyond the operating system, such as compilers, libraries and specific software, as shown at figures 9 and 10.

The `“/exports/cluster”` volume is mounted temporarily from front-end `172.30.1.1` on `“/opt/cluster”`, so as to execute:

```
/opt/cluster/admin/postinstall.sh node23 172.30.2.23
```

The script “`postinstall.sh`”, for example, browses the directory that contains it to execute all the scripts therein whose name match a specific pattern. It is presented at figure 8.

```
# loop through the postinstall_files and execute them
cd $CDIR/admin
for SCRIPT in $(ls postinstall_[0-9][0-9]* | sort)
do
    echo "#####"
    echodate "### $SCRIPT starting"
    $SCRIPT
    echodate "### $SCRIPT finished"
done
```

Figure 8: “`postinstall.sh`”

## 8.1 "chroot-scripts" section

For "chroot-scripts", the boolean value "`chrooted`" indicates when the script is executed:

<code>"chrooted"</code>	Meaning
false (default)	The installed system is still mounted on " <code>/mnt</code> ", and the bootloader is not yet installed
true	The "chroot" on " <code>/mnt</code> " has been done, the bootloader is installed, and the installed system is up and running.  There's no longer any need to prefix directories with " <code>/mnt</code> "

The "chroot-script" we use is shown at figure 9.

The "`CDATA`" tag contains the code to be executed.

## 8.2 "init-scripts" section

The one we use is shown at figure 10.

```
<scripts>
<chroot-scripts config:type="list">
  <script>
    <chrooted config:type="boolean">>true</chrooted>
    <debug config:type="boolean">>true</debug>
    <feedback config:type="boolean">>false</feedback>
    <filename>postinstall.sh</filename>
    <interpreter>shell</interpreter>
    <source><![CDATA[echo "##### Running postinstall.sh"
      hostname >/tmp/hostname.out
      set >/tmp/set.out
      mkdir -p /opt/cluster
      mount -t nfs -o nolock 172.30.1.1:/exports/cluster /opt/cluster
      /opt/cluster/admin/postinstall.sh node23 172.30.2.23
      umount /opt/cluster
    ]]></source>
  </script>
</chroot-scripts>
```

Figure 9: "chroot-scripts"

```
<init-scripts config:type="list">
  <script>
    <debug config:type="boolean">>true</debug>
    <filename>postinstall2.sh</filename>
    <source><![CDATA[echo "##### Running postinstall2.sh"
      mount -t nfs -o nolock 172.30.1.1:/exports/cluster /opt/cluster
      /opt/cluster/admin/postinstall2.sh
      umount /opt/cluster
      mount -at nfs
    ]]></source>
  </script>
</init-scripts>
</scripts>
```

Figure 10: "init-scripts"

### 8.3 Specific software installation scripts

In order to add software to the current configuration, one just needs to place an archive in “/opt/cluster/spool” and a script like “postinstall\_20\_Atlas” dans “/opt/cluster/admin”. It is presented at figure 11.

```
root@vega:/opt/cluster/admin > cat postinstall_20_Atlas
#!/bin/bash

# AutoYaST Post-processing

set -x -v

#-----
# Installing Atlas

ARCHIVE=Atlas.tar.gz

cd /

tar xfp /opt/cluster/spool/$ARCHIVE
```

Figure 11: “postinstall\_20\_Atlas”



Another script copies configuration files from a **reference directory**. These include “/etc/passwd” and “/etc/group”, as their contents evolves over time, and the links to users home directories from “/home”. It is presented at figure 12.

```
root@vega:/opt/cluster/admin > cat postinstall_01_NodeConfig
#!/bin/bash
# AutoYaST Post-processing

set -x -v

#-----
# Copy node-config files

cp -drp /opt/cluster/spool/node-config/* /
```

Figure 12: “postinstall\_01\_NodeConfig”

## 8.4 SSH matters

We handle SSH keys in the following way:

- when a node is installed, nothing special is done for its SSH server key. The latter is thus created afresh;
- there is no local user password on the compute nodes for regular users: an SSH key pair is created on the front-end in their shared home directory, and the public key is copied to “`authorized_keys`”;
- the compute nodes’ “`root`” account is local. The “`.ssh`” folder is copied from the reference directory.

## 9 Conclusion

We've been using AutoYaST for several years to manage DIT's general purpose clusters.

The setup described above use has been supplied in the first place by Dalco's Jonas Lehmann and Beat Rubischon, for Mizar and Alcor respectively. These two machines have been decommissioned recently. Useful suggestions were also made by IBM's Markus Bärtschi when Callisto was installed afterwards.

The experience is very positive from our point of view:

- the tools supplied by Novell do what is expected from them, without any problem;
- compute nodes installation is done with a couple of scripts that create the PXE configuration and message files, as well as AutoYaST control files, from models;
- a couple of other scripts do compute node specific complementary setup after the disk has been formatted and the operating system has been installed;
- all these scripts are written in Bash and Perl, and can be taken over by any system administrator if needed;
- there's no syndrom in which the management tool would be the main problem because we would have to circumvent its limitations or add missing features.