Kadepoy installation HOW-TO

Benjamin DEXHEIMER

0.1

Abstract

This HOW-TO deals about the installation methods of Kadeploy. Firstly we would explain how-to perform a parallel installation aside from the production instance of a previously installed Kadeploy, and secondly how to perform a production installation of Kadeploy 2.1.7.

Table of Contents

1. How-to perform a parallel installation

This part explains how-to have at the same time a testing-purpose instance of Kadeploy aside from the production instance.

1.1. Pre-requisites

The Kadeploy commands suite cannot handle a customized configuration path, which is set by default at `/etc/kadeploy`. It implies to use a second server so as to have at the same time a production and a testing-purpose instance of Kadeploy.

From now, the frontend machine hosting the production-class Kadeploy will be designated by *frontend-prod* and the frontend hosting the testing version will be designated by *frontend-dev*.

Kadeploy needs several resources :

- rsh client
- DHCP server
- PXE booting environment (usually from Syslinux project) for booting nodes
- TFTP server, and therefore a writeable access to root TFTP directory served by tftp daemon (e.g. `/var/lib/tftpboot` and its sub-directories)
- MySQL database
- File-system level read access to Grid'5000 deployable system images.

1.1.1. rsh client

Kadeploy requires the usage of a legacy rsh client. This could be somewhat problematic because rsh is not installed by default anymore in modern Linux distributions, although rsh responds at CLI (but it's a symbolic link pointing to ssh client). In debian-like systems, you have to install the package *rsh-client* to obtain the real rsh client.

1.1.2. DHCP

Because of nature of DHCP, it's rather difficult to have side by side 2 servers handling the same set of nodes but with different configuration. That's why nodes used during Kadeploy tests will be booted with the same DHCP as for production use.

1.1.3. TFTP and PXELinux

Because of previous DHCP configuration, the testing instance of Kadeploy will use the same TFTP server and therefore, the testing and production instances need to write PXE files at the same place.

The steps to follow are :

1. Install a NFS server on frontend hosting TFTP service.
2. Make export root TFTP directory (cf `/etc/exports`) from this frontend.
3. On frontend-dev : import the previously exported root TFTP directory.

The PXE environment is usually set by the cluster sysadmin in accordance with the TFTP installation layout. PXE related files and directories are likely to be located under TFTP root directory (cf `pxelinux.0` bootloader and `pxelinux.cfg` directory containing the boot configuration for the nodes). If TFTP root directory is NFS-mounted by frontend-dev, it will have access to PXE ressources, too.

1.1.4. MySQL database

Kadeploy needs an access as root and deploy user to a MySQL instance. Make sure your MySQL running server is reachable from frontend-dev using the MySQL root user. For instance, the following command should be working from frontend-dev :

```
mysql -u root -p -h mysql.<site>.grid5000.fr
```

1.1.5. Grid'5000 images

Kadeploy needs a direct file-system access to deployable system images. The `/grid5000` directory have to be NFS mounted by fronted-dev.

1.2. Kadeploy 2.1.7 installation

The frontend considered for this installation is *frontend-dev*.

Please follow the instructions below :

1. Retrieve and unpack the tarball archive of Kadeploy 2.1.7 (`kadeploy-2.1.7.tar.gz`) into a safe location (e.g. your home directory).

   ```
   cd <path_to_unpack_kadeploy_archive> && tar -xvzf kadeploy-2.1.7.tar.gz
   ```
2. Edit the heading variables of `Makefile` located at the top of Kadeploy archive.

   a. Especially, the following variables :

   - **DISTRIB** : your distribution name. It sets an installation path for Perl library used by Kadeploy. Recognized distributions are : debian4 (Debian Etch 4.x), Fedora Core 4.

Another distribution will set to default pathname.
- **PREFIX** : the PREFIX for installation pathnames.

b. This variables may be of interest, too :

- **KADEPLOYCONFDIR** : where to place configuration files.
- **KADEPLOYHOMEDIR** : where Kadeploy files are installed.
- **DEPLOYUSER** and **DEPLOYGROUP** : username and groupname of the identity used by Kadeploy to perform his work (instead of using the super-privileged user *root*). The values *deploy* for both variables are set by default.

3. Install Kadeploy by using the command as *root* :

```
make kadeploy_install
```

4. Make sure that UID/GID used for the *deploy* user on frontend-dev are the same that those on frontend-prod. Otherwise, make changes accordingly on frontend-dev to get same UID/GID values on both machines. It's important for later file-system accesses
5. Get TFTP root directory and Grid'5000 deployable images repository NFS mounted on frontend-dev.
6. Get your Kadeploy 2.1.7 configuration ready. A quick way to achieve this is to copy into the test kadeploy configuration directory all production configuration files and modify them. Some informations into this files will be needed by the next step.
7. Setup the Kadeploy database. The goal is to have side by side the production DB and the test DB. Informations will be sync from production to test DB to get the test DB initialized and working-ready. The command to interact with the Kadeploy DB is **kadatabase**. It will ask for several informations, as MySQL login and password, database names.

   a. Issue the following commands :

   - **kadatabase --create-db-deploy-217** : create a Kadeploy 2.1.7 empty database.
   - **kadatabase --add-deploy-db-user** : add the deploy DB user into the MySQL instance. This is the mysql user used by Kadeploy for getting access to his database. It may be different from the production deploy DB user.
   - **kadatabase --dup-production-db** : duplicate the production DB into the test DB. Both DB names will be asked during operation. After that, you will have a working-ready test DB.
   - **kadatabase --dup-deployment-rights** : duplicate deployment rights into test DB. To be used each time a OAR submission have been issued from the OAR frontend. It gives the test kadeploy rights to deploy on the reserved nodes.

8. That's all folks !

Note

- Remember to issue kadeploy commands from frontend-dev when you want to test the 2.1.7 version.
- Remember to issue **kadatabase --dup-deployment-rights** each time you get a OAR reservation.

2. How-to perform a migration from 2.1.6 to 2.1.7

TODO.