Kadepoy 2.1.7 installation HOW-TO

Benjamin DEXHEIMER

Emmanuel JEANVOINE

0.2

Abstract

This HOW-TO deals about installation process of Kadeploy. In particular, the process for a parallel installation of Kadeploy aside from an already working (but older) instance of Kadeploy would be explained. Then some generic points related to installation process will be exposed.

Table of Contents

1. Pre-requisites before performing a duplicate installation of Kadeploy

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
- Perl modules
- File-system level read access to Grid'5000 deployable system images.

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.

1.2. 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.

1.3. 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.

1.4. 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
```

1.4.1. In case of parallel installation

If you're planning for a parallel installation, you have to setup a testing-purpose Kadeploy DB aside from the production DB. The test DB would be hosted on the same MySQL server than the production DB. The script used to setup the test DB expects to find both databases on the same MySQL server.

1.5. Perl modules

Kadeploy is fully written en Perl, and therefore is based upon some Perl modules. You have to ensure that the packages listed below (Debian names) are installed on the deployment frontend (the machine hosting the Ka* commands) :

- `libterm-readkey-perl`
- `libdbi-perl`
- `libdbd-mysql-perl`
- `libfile-chdir-perl`

1.6. 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.

2. Key points to be aware of

2.1. Makefile usage

Kadeploy's makefile offers 3 targets.

- **install** : use this target to install Kadeploy on the deployment frontend (may it be production or test).

  It must run as root.

  The following actions are performed on the target system :

  1. Create if necessary user and group *deploy*.
  2. Copies commands, libraries, configuration files and manpages to correct places.
  3. Do the sudowrapper configuration (in `/etc/sudoers`) and create corresponding symlinks to Kadeploy utility commands.
- **uninstall** : use this target to uninstall a previous installation of *Kadeploy 2.1.7*.

  It must be run as root.

  The following actions are performed on the target system

  1. Removes all Kadeploy files (commands, libraries, configuration files and manpages).
  2. Removes the sudowrapper configuration (in `/etc/sudoers`).
- **dist** : use this target from the SVN root directory (`trunk/`). It packs all Kadeploy files into a tgz typed archive. This archive will be used later to install Kadeploy on a deployment frontend.

  In particular, it generates this HOW-TO and manpages.

2.2. Setup of database

Various operations on Kadeploy internal database should be done with the **kadatabase** command. When using this command, you should keep in mind that point :

- DB settings should have be done on Kadeploy main configuration file (`/etc/kadeploy/deploy.conf`).

  Specifically, some informations like DB hostname, DB name, deploy username and password are expected by kadatabase to be found in this file. Ensure to have a working configuration before doing any DB setup.

It supports the following options :

- **--create-db-deploy-217** : creates a Kadeploy 2.1.7 compliant DB, which is empty from any data.

  ### Warning

  this option is destructive if you're giving a name already used by a previously existing DB.
- **--dup-production-db** : duplicates data from a production DB into a testing DB. It asks for both DB names.

  ### Note

  It duplicates only static data (not history), that is said : disk, environment, node and partition tables.
- **--dup-deployment-rights** : duplicates deployment data rights from production DB into testing DB.

  ### Note

  It is useful when testing Kadeploy 2.1.7 when OAR is still connected to a production working Kadeploy 2.1.6.
- **--db-migration-216-to-217** : changes a Kadeploy 2.1.6 DB schema into a Kadeploy 2.1.7 compliant DB schema.
- **--add-deploy-db-user** : adds the deploy user into the user table and grants him corrects privileges.
- **--del-deploy-db-user** : removes the deploy user from the user table and flushes his privileges.
- **--purge-test-db** : removes all data from the testing DB (disk, environment, node, partition and rights tables).
- **--drop-db-deploy** : drops the specified DB (may it be production or testing).
- **--clean-db-deploy** : removes all history data from the production DB (deployed and deployment tables).

Others options are no longer used and may be deprecated ; please do not use them.

2.3. Recommanded installation layout

You may choose the installation PREFIX of your choice. A standard choice will be `/usr/local`.

the KADEPLOYHOMEDIR will default to <PREFIX>/kadeploy-2.1.7. It's generally a good idea to leave it unchanged.

DEPLOYUSER and DEPLOYGROUP will default to *deploy*. To keep compliance with habits from Kadeploy 2.1.6, it's generally a good idea to leave them unchanged.

KADEPLOYCONFDIR must be leaved unchanged. Kadeploy commands will search their configuration file into `/etc/kadeploy`. This directory is unfortunately hard-coded and cannot be easily changed without

code refactoring. By default, Kadeploy 2.1.7 creates a `/etc/kadeploy-2.1.7` configuration directory and makes a symlink `/etc/kadeploy` onto it. It preserves a previously existing `/etc/kadeploy` directory (from a 2.1.6 release for instance) by renaming it into `/etc/kadeploy.old`.

if your DISTRIB is not in the Makefile, some lines should be added to set up some PATHS which are distribution dependant. Please ask Kadeploy staff for extensive explanations.

2.4. Rights access to directories

When a user issues a *kadeploy* command, it has been run in fact as the *deploy* user, defined in `Makefile` and `/etc/kadeploy/deploy.conf`. It's mandatory that some specific directories will be open for read/write access to that system user, created at installation time.

In particular, make sure that the directories below are opened for R/W access to user *deploy*:

- `/etc/kadeploy`
- `/var/lib/tftpboot` (usually used on Debian for TFTP root directory configuration) and the subdirectories `boot/`, `boot/cache`, and `pxelinux.cfg/`

It means that this directories must be owned by the *deploy* user.

2.4.1. In case of parallel installation

You will be driven to align the *deploy* UID/GID of frontend-dev on those of frontend-prod. Do not forget to check directories listed above with the UID/GID updated.

3. How-to perform a parallel installation

The frontend considered for this operation 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 :

   - **KADEPLOYHOMEDIR** : where Kadeploy files are installed. By default it's compound by the PREFIX and the path string `kadeploy-<VERSION>`.

That variable is tied to the one specified in main Kadeploy's configuration file `deploy.conf` : *kadeploy2_directory*. Please ensure the same value has been entered for boths variables otherwise some path will be missing at run time.
- **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 value *deploy* is set by default for both variables.

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

```
make 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. To setup the second Kadeploy database, 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've done an OAR reservation.

4. Collateral fittings

Several operations must be performed to get a fully working Kadeploy 2.1.7.

- The deployment kernel must be rebuilt thanks to the script provided in the `addons/deployment_kernel_generation` directory of the Kadeploy's distribution. Warning, the deployment kernel must not be compiled with the SMP support, otherwhise kexec will not work.

- The configuration files `/etc/kadeploy/deploy.conf` and `/etc/kadeploy/deploy-cluster.conf` must be updated according the example provided in the `confs` directory of the distribution. In particular, pay attention to the options **use_kexec_by_default**, **default_target_device** and **default_target_partition**.
- The preinstall file must be modified according the files given in the `addons/preinstallation` directory of the distribution.
- The SSH public key of the user deploy must be added in the root's `authorized_keys` on the production environment of the nodes (this is used for the kexec optimization).
- The SSH public key of the user root of the production environment must be added in the root's `authorized_keys` on the production environment of the nodes (this is used for the kexec optimization).
- The files located in the `scripts/scripts_used_to_deploy` directory of the distribution must be copied in the `/usr/local/bin` directory of the production environment on the nodes.