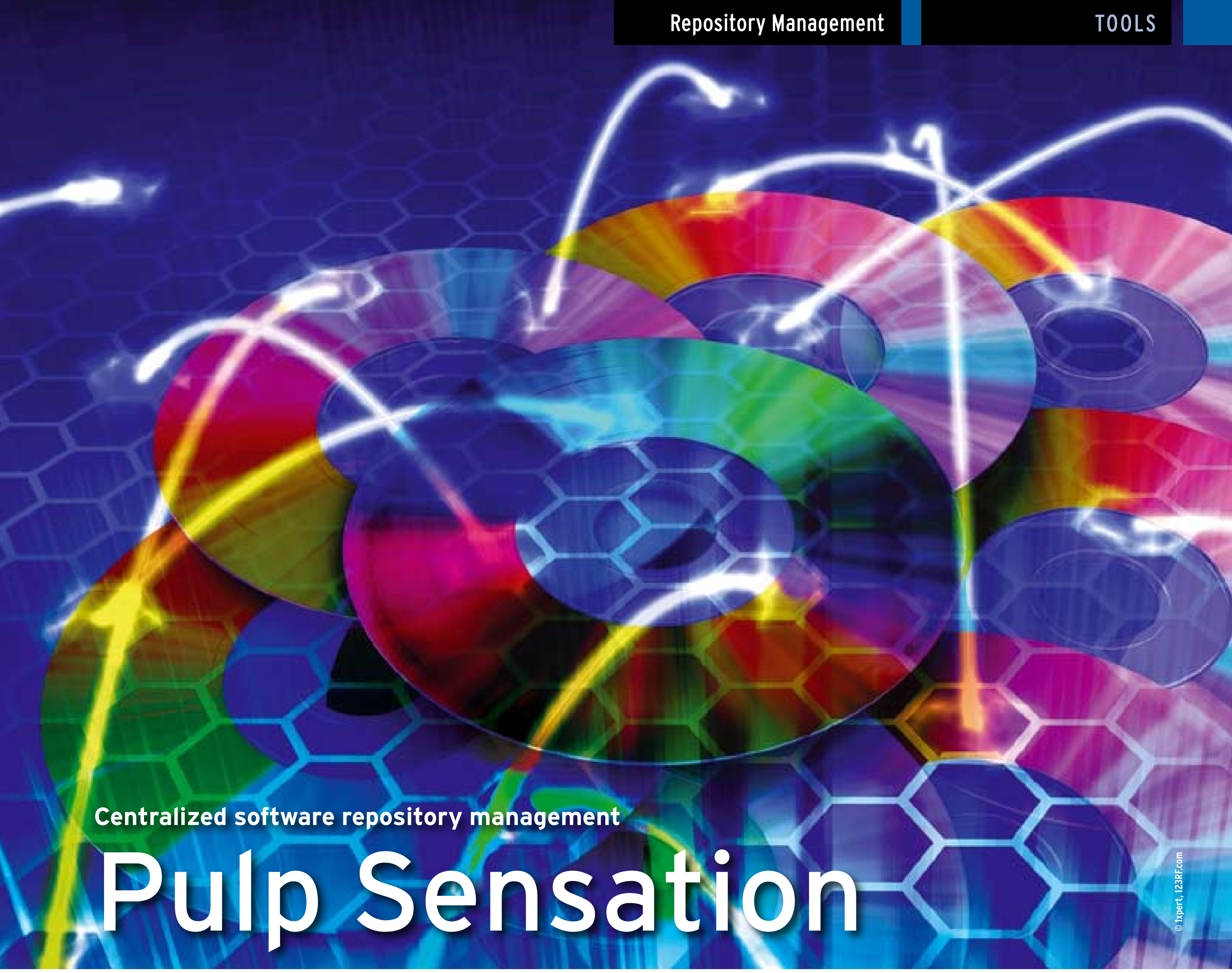

Centralized software repository management

Pulp Sensation

If you use free software, you will likely need to manage multiple software repositories and keep all the information up to date. Pulp gives you a centralized approach to repository management. By Thorsten Scherf

Pulp helps administrators consolidate multiple software repositories centrally. To allow this to happen, Pulp can access a variety of sources; for example, you can just as easily query a regular Yum server as you can access the Red Hat Network or software packages hosted by a web server.

Pulp [1] doesn't act as a proxy between the clients and the various repositories; instead, it mirrors the packages locally and updates the packages at regular intervals. Obviously, this means that you need to provide a large amount of storage space up front.

That said, Pulp can shift individual repositories out to external servers, which are referred to as content delivery servers. To do so, the central server simply routes client requests to external servers, which then respond to the requests. This approach makes it pretty easy to establish geographically distributed systems. Thanks to highly granular access rules, Pulp makes sure that clients can only query specific repositories. User authentication can take place against an LDAP server, which makes it possible to deploy Pulp in very large environments.

The software also has impressive reporting capabilities. Administrators not only have access to the history feature that shows which client system accessed which repository and when, they can also view the package status of the clients at any time. Thus, it is quite easy to discover systems that have not yet installed an important update package. If you are interested in experimenting, you will probably appreciate the fact that Pulp offers an API that supports convenient scripting of regularly recurring tasks. Figure 1 shows the architecture of a Pulp environment.

Installation

If you are interested in deploying Pulp, you will find prebuilt RPM packages for Fedora, Red Hat Enterprise Linux, and CentOS [2]. You can access the source code via the Git repository [3]. After downloading the Yum repository configuration file, you can proceed to install the server in a normal way with Yum (Listing 1). After initialization, the MongoDB database server is then ready for use. Make sure the `/etc/pulp/pulp.conf` configuration file includes the correct server name and the client systems have access to ports 5672, 5674, and 443.

The client systems also need access to the Pulp repository that I mentioned previously. The easiest way to achieve this is to specify the details during the install. If you use Kickstart to automate the installation of systems, you can install and configure the required client software at the same time. The systems will then be ready for use directly after the installation without requiring any manual configuration.

The client package goes by the name of `pulp-client` and should be installed on the server, too, under ideal circumstances. This step is necessary for administrative work. The important thing is to modify the `/etc/pulp/client.conf` configuration file for the client software so that all references to the server use the FQDN (fully qualified domain name); otherwise, server certificate validation will fail (Listing 2). The certificate contains the server's FQDN. If the client attempts to use any other name to access the server, the software will complain that the name is incorrect. After this, you need to launch the client agent (`service pulp-agent start`).

If you intend to deploy a content delivery server in addition to the Pulp server, you need to install the pulp-cds package to support it. Then, you need to configure the server via the `/etc/pulp/cds.conf` configuration file and type

```
service pulp-cds start
```

to launch it.

Users and Roles

Pulp comes with a default administrative account. It is a very good idea to change this immediately:

```
# pulp-admin -u admin -p admin user update ↩
        --username admin --password
Enter new password for user admin:
Re-enter new password for user admin:
Successfully updated [ admin ]
```

Every administrative action on the server relies on previous authentication. To avoid being prompted for your username and password whenever you attempt to execute a command, log in to the server and store the user credentials (in the form of a user certificate) in `~/.pulp`. For example:

```
# pulp-admin auth login -u admin -p password
User credentials successfully stored at ↩
  [/home/pulp-user/.pulp]
```

New users can be added, deleted, or modified with the `user` command. To grant the user specific rights, you can run the `permission grant` command and modify rights accordingly. Instead of assigning rights to individual users, you can assign users to specific

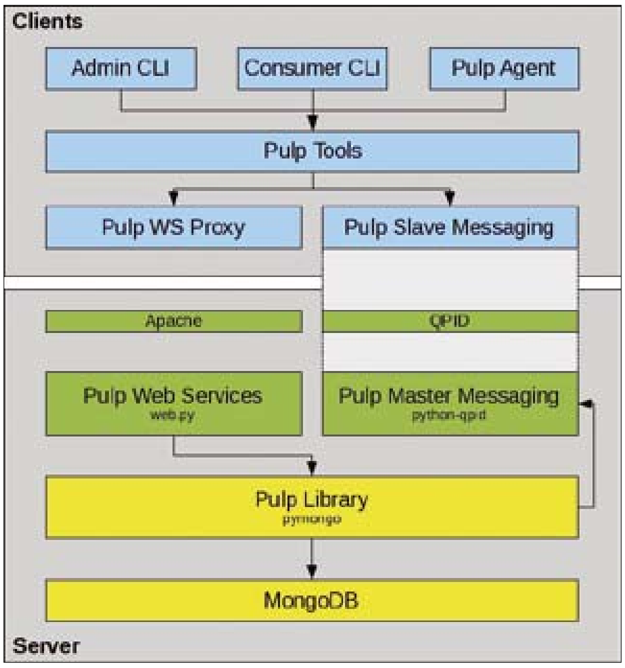

Figure 1: Many components interact in Pulp.

roles and then assign rights to these roles. This approach makes it easier to group individual users with similar rights. Pulp features a `role show` command that lists the roles, and `role info` lets you list the rights for a specific role (Listing 3).

Repositories and Distributions

To populate the server with data, you need to synchronize the required repositories. Pulp's definition of a repository is a collection of software RPMs and installation files (distributions), such as kernel and initrd. The following command creates and synchronizes a repository:

```
# pulp-admin repo create ↩
    --id Fedora14-x86_64 ↩
    --feed yum:http://download.fedoraproject.↩
      org/pub/fedora/linux/releases/14/↩
      Everything/x86_64/os/
Successfully created repository [ example ]
# pulp-admin repo sync --id Fedora14-x86_64
```

Pulp shows the synchronization status when you enter:

```
pulp-admin repo status --id Fedora14-x86_64
```

Besides this, other useful commands let you modify the repository. For example, you can create filters so cer-

Listing 1: Installation with Yum

```
# yum install pulp
# service pulp-server init
# service pulp-server start
# chkconfig pulp-server on
```

Listing 2: Server Name in the Configuration

```
# grep ipa1 /etc/pulp/client.conf
host = ipa1.virt.tuxgeek.de
baseurl = https://ipa1.virt.tuxgeek.de/pulp/repos
keyurl = http://ipa1.virt.tuxgeek.de/pulp/gpg
ksurl = http://ipa1.virt.tuxgeek.de/pulp/ks
```

Listing 3: Using Roles to Assign Rights

```
# pulp-admin role list
+------------------------------------------+
              Available Roles
+------------------------------------------+
  super-users
  consumer-users

# pulp-admin role info --role super-users
+------------------------------------------+
       Role Information for super-users
+------------------------------------------+
Name                    super-users
Users                   admin
Permissions:
  /          CREATE, READ, UPDATE, DELETE, EXECUTE
```

tain packages are not synchronized, you can schedule automatic updates using the `--schedule` option, and you can enable these directly when you create a repository or make the changes retroactively. The `repo update` lets you retroactively modify individual configuration settings.
Pulp outputs a list of all distributions when you type `distribution list` (Listing 4). When you install a new system, you can specify the URL for the required distribution to tell the installation program to download the required kernel, initrd, and installation files directly.

Pulp Consumer

Once you have the pulp-client package installed and configured on the client, you need to register the client with the Pulp server. This is handled by a call to

```
pulp-client consumer create --id client1
```

As with package installation, you can automate this step – for example, as part of a Kickstart profile. The

```
pulp-client repository list
```

command then gives the client a list of all the repositories available on the server (Listing 5). The `Sync Schedule` line uses cron format to define the time at which the repository is updated.
For the client to install packages from this repository, you need to bind the client to the corresponding repository. Pulp includes the

```
pulp-client consumer bind --repoid Fedora14-x86_64
```

command, which tells Pulp to create a new `pulp.conf` configuration file below `/etc/yum.repos.d` to point to the new repository on the server (Listing 6). If the client binds to other repositories on the server, the bindings are all stored in the same Yum configuration file. Running `yum repolist` on the client confirms that it can install packages on the Yum repository you just configured on the Pulp server.
To help you keep track of which clients access which repositories, Pulp has a history that lists all instances of client access. For example,

```
pulp-admin consumer info --id ipa2
```

shows the configuration of the individual repositories and which individual packages are installed from them. Grouping individual clients is also extremely useful. The option for this is `consumergroup create`. Instead of running a specific action against a single client, Pulp runs the action against all members of the group. For example, if you want to install the `pidgin` package on all clients belonging to the `desktop-clients` group, you would enter:

```
pulp-admin package install -n pidgin --consumergroupid desktop-clients
```

Of course, this means all members of the group need access to the repository on which the package is located.

Conclusions

Pulp is a comprehensive management tool for software repositories. If you do not need a graphical interface, Pulp can help you to manage huge numbers of client systems, even if they are geographically distributed. The comprehensive reporting functions let you keep track of your systems, and if you enjoy automating processes, you will appreciate the Pulp's REST API.
The data Pulp expects or returns (e.g., when you create a new repository) is in JSON format [4]. Any state-of-the-art scripting language, such as Perl or Python, will offer a matching encoder/decoder. ■

Listing 4: Distributions

```
pulp-admin distribution list
+------------------------------------------+
       List of Available Distributions
+------------------------------------------+

Id                    Fedora14-x86_64
Description           Kickstart Trees for Fedora14-x86_64
URL                   http://ipa1.virt.tuxgeek.de/pulp/ks/released/fedora/14/x86_64/os
```

Listing 5: Overview of Repositories

```
# pulp-client repo list
+------------------------------------------+
       List of Available Repositories
+------------------------------------------+

Id                    Fedora14-x86_64
Name                  Fedora14-x86_64
FeedURL               http://download.fedoraproject.org/
pub/fedora/linux/releases/14/Everything/x86_64/os/
FeedType              yum
Arch                  noarch
Sync Schedule         10 0 * * *
Packages              10
Files                 0
Distributions         None
Publish               True
Clones                []
Groups                None
Filters               []
```

Listing 6: Pulp Creating a New Yum File

```
#
# Pulp Repositories
# Managed by Pulp client
#
[Fedora14-x86_64]
name = Fedora14-x86_64
enabled = 1
sslverify = 0
gpgcheck = 1
baseurl = https://ipa1.virt.tuxgeek.de/pulp/repos/pub/
fedora/linux/releases/14/Everything/x86_64/os
```

Info

[1] **Pulp project:** [http://pulpproject.org/]

[2] **Pulp download:** [https://fedorahosted.org/pulp/wiki/UGInstallation]

[3] **Git repository:** [http://git.fedorahosted.org/git/?p=pulp.git]

[4] **JSON project:** [http://www.json.org/]