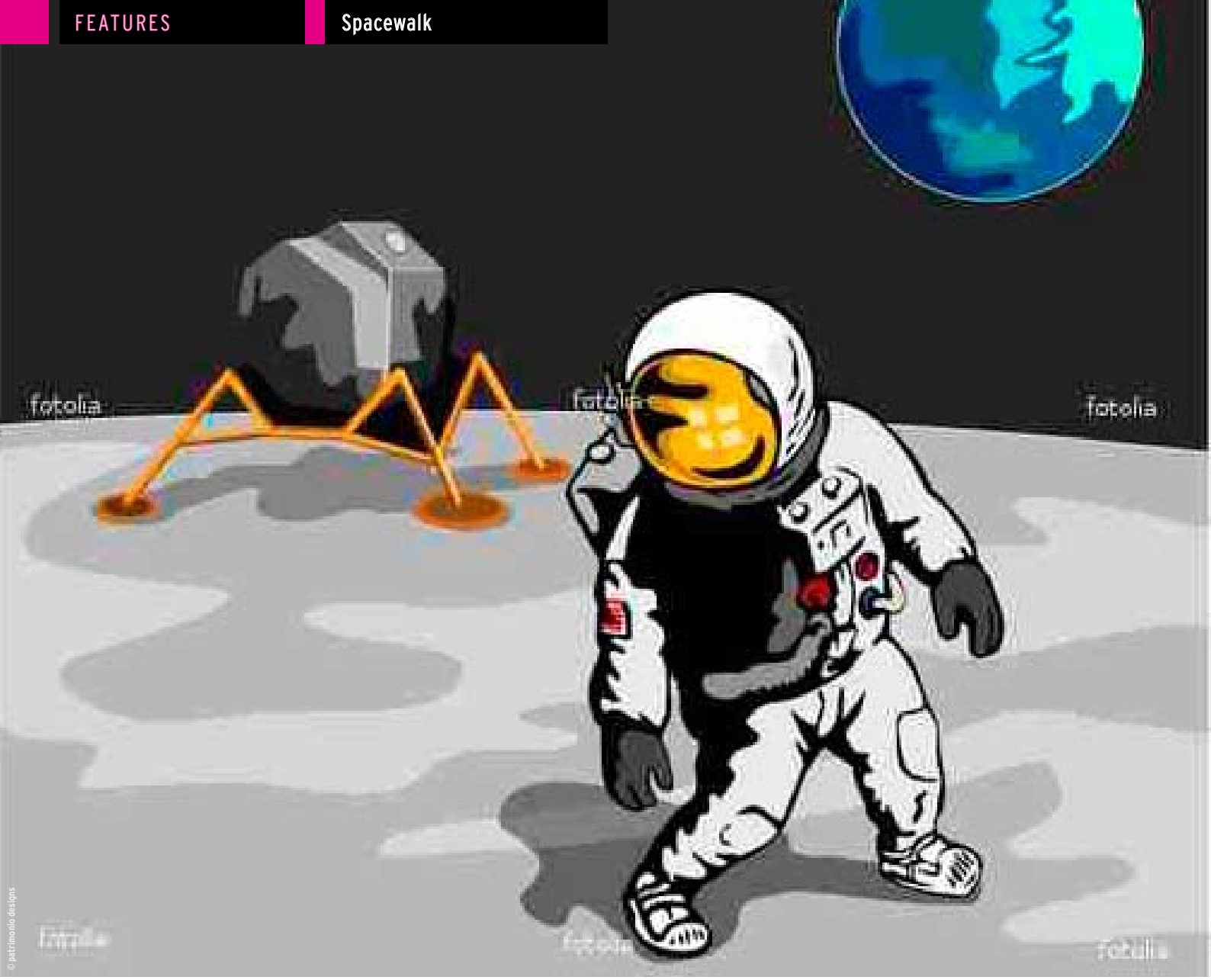

Managing Linux systems with Spacewalk

Moon Landing

When your system landscape reaches a certain size, managing Linux systems manually is time-consuming and impractical. Enter Spacewalk: an open source tool that takes the footwork out of network management.

By Thorsten Scherf

Spacewalk [1] is the open source derivative of the popular Red Hat Network Satellite Server. Red Hat published the source code for the server in the summer of 2008, and the community has now released version 1.0. The application's core tasks include RPM package software provisioning, managing configuration files, and kickstart trees, thus supporting the installation of bare-metal systems.

The approach that Spacewalk uses is quite simple: Before a system can access Spacewalk's resources, it first has to register with the server. Registration can be based either on a username/password combination or an activation key that is pregenerated by the Spacewalk server. After registration, the system appears in the server's web GUI.

If the server has more resources, you can assign them to the system at this point. Resources include software packages or configuration files that are normally organized in channels. A system always has exactly one base channel with optional subchannels. The base channel contains the RPM-based operating system, such as Red Hat Enterprise Linux, Fedora, or CentOS. The subchannels contain additional software packages that are independent of the operating system, such as the Red Hat Cluster Suite or the 389 Directory Server.

Spacewalk can clone existing channels and create new channels from scratch. This feature gives you full control of the software stack that you provide via Spacewalk. Configuration channels help you distribute the

configuration files for the software packages. Spacewalk also keeps older versions of the files to let you roll back to a previous version at any time if the need arises.
The software packages or configuration files can be installed either via the target system or centrally in the Spacewalk web front end. To avoid spending too much time on the installation of a large number of systems, you can assign systems to logical groups and apply the installation of a resource to a group. For example, it might make sense to assign all your web servers to a *WWW-Server* group in Spacewalk. When a new version of the web server software is released, you would simply tell Spacewalk to apply the update to the group, automatically updating all the systems belonging to the group.
The installation uses polling by default; in other words, the client systems query the server at a predefined interval (which defaults to four hours) to see if new actions have been defined since the last poll. If so, Spacewalk then runs these actions. As an alternative, you can trigger the installation of software packages and other actions using a push approach. The client system and the Spacewalk server talk to each other constantly using the Jabber protocol. Any new actions you define are immediately run on the client by Spacewalk.

Ground Control

Communications are always from the client to the server; this is important with respect to firewall rules. A list of the network ports you need to enable can be found online [2]. Besides software package or configuration file installation, actions can also run arbitrary commands on the individual systems via the Spacewalk server. For example, after creating a new configuration file for your web servers and distributing it to the systems, you need to restart the web server process to parse the new configuration instructions. Instead of logging in to each individual system or using a `for` loop, simply issue the restart command centrally on the Spacewalk server.
Installing new systems is also quite simple. Spacewalk has the installation files you need in the form of kickstart trees. The installation candidate uses a boot medium such as a CD, a USB stick, or a PXE-capable network card to contact the server. The First-Stage Installer, which is part of the installation medium, defines which server will handle the installation.
The remaining installation steps are handled by the Second-Stage Installer, located on the Spacewalk server and transferred to the client system when the installation starts. If you want to automate the installation fully, define the kickstart file location in the boot medium. The kickstart file is a kind of answer file that describes the properties of the installation candidate, such as partitioning, software, language, and firewall settings. Of course, you can create a kickstart file on the Spacewalk server and just include a link to the file on the boot medium.
Spacewalk can manage any RPM-based distribution. You even have the option of operating client systems across multiple organizations. Using the web interface, the administrator creates various organizations and assigns a certain number of system entitlements to them. Entitlements are linked to certificates that Spacewalk automatically generates during the installation. You can then add users to the organizations.
If a client is registered with a user account from a specific organization, the system is assigned to this organization. When users from the organization logs into the Spacewalk server, they will only see the systems in their own organization. This feature is useful if you manage multiple departments and prefer to manage the systems in the individual departments separately. You just assign them to different organizations, which, of course, you need to create up front.

Installation

Spacewalk can be installed on Red Hat Enterprise (RHEL) [3], Fedora [4], or CentOS [3] Linux. Note that Spacewalk does need a current Java Runtime Version 1.6.0 or newer. You can use the Open JDK for this; Fedora includes it out of the box. Admins on RHEL or CentOS can retrieve the package via the additional EPEL (Extra Packages for Enterprise Linux) software repository.
Besides the Java package, an Oracle 10g database is also required for installing Spacewalk. Oracle XE provides a free version of the database. The developers are currently working hard on implementing support for an open source database after identifying PostgreSQL as the best alternative to Oracle. As of this writing it is hard to say when official support for PostgreSQL will be available, but it makes sense to check the roadmap [5] or the mailing lists [6] at regular intervals.

Oracle XE

After installing the repository RPM for your distribution, the first step is to install Oracle Express, which you can download for free [7]. You will need version 10.2.0.1. Besides the database, you also need the oracle-instantclient-basic and oracle-instantclient-sqlplus, which you can then install with Yum:

```
yum localinstall --nogpgcheck ➥
  oracle-xe-univ*.rpm
oracle-instantclient-basic*.rpm
oracle-instantclient-sqlplus*.rpm
```

Before configuring the database, you should make sure that your hostname points to the correct IP address in your `/etc/hosts` to avoid problems

Listing 1: Oracle Listener Configuration

```
cat >> /etc/tnsnames.ora << 'EOF'
XE =
    (DESCRIPTION =
       (ADDRESS_LIST =
          (ADDRESS = (PROTOCOL = TCP)(HOST = localhost)
           (PORT = 1521))
       )
       (CONNECT_DATA =
          (SERVICE_NAME = xe)
       )
    )
EOF
```

with the Oracle Listener configuration later on. Use the following parameters for the configuration:

```
HTTP port for Oracle Application Express: 9055
Database listener port: 1521
Password for SYS/SYSTEM: Password
Start at boot: y
```

The default HTTP port for the Oracle Express application (8080) is already occupied by the Tomcat application server, so you need to choose an alternative port to avoid conflicts.
To talk to the database, you need to configure the listener in the /etc/tnsnames.ora file (Listing 1).
Now you just need to make a few changes to the database. To do this, log in to the database with sqlplus and create a spacewalk user, to which you could assign a password of spacewalk (Listing 2).
The standard configuration of Oracle Express supports a maximum of 40 simultaneous connections, which is not enough for Spacewalk operations. The instructions in Listing 3 change the limit to a maximum of 400 connections.
Now you need to restart the database by giving the /sbin/service oracle-xe command.

Spacewalk Setup

The next step is to install the Spacewalk server. To do so, you need to include the Spacewalk repository as described previously. You should have a spacewalk.repo file that points to the appropriate repository in /etc/yum.repos.d/. The following command starts the installation:

```
yum install spacewalk-oracle
```

Because this package depends on all the other Spacewalk packages, the package manager will automatically download and install the dependencies in the next step. Then you can configure the application interactively with the setup tool or with the use of an answer file (Listing 4).
Pass the file in to the setup tool as follows:

```
spacewalk-setup --disconnected --answer-file=answerfile
```

The configuration can take some time to complete as the process sets up the database tables. The setup tool then launches all the required services. You can manually restart using the /usr/sbin/rhn-satellite tool.
To configure the system, launch the Spacewalk web interface via its URL (*http://spacewalk.server.tld*). Besides contact information, you can also set the password for the Spacewalk administrator here.

Software Channels

The next step is to set up an initial software channel for the client systems. When you register a client, you must specify exactly one base channel for the client; it will use this channel to retrieve its operating system packages and their updates. Of course, you can set up subchannels for the base channel and assign the subchannels to clients as needed. After doing so, you can use the subchannels to distribute more RPM packages to the systems. The packages can be your own creations or RPMs from other repositories.
The easiest approach to setting up a software channel is to use the web interface (*Channels* | *Manage Software Channels* | *Create*; Figure 1). Thanks to the Spacewalk API, you can also script this process [8]. Call the script as follows:

```
create_channel.py --label=fedora-12-i386 --name "Fedora 12 32-bit" --summary "32-bit Fedora 12 channel"
```

In the script, you need to provide the Fully Qualified Domain Name (FQDN) for the Spacewalk server and the user account for creating the channels, such as the Spacewalk administrator account created previously. The *Users* tab also gives you the option of creating more users with specific privileges (Figure 2).
The channel you set up should now be visible in the Channels tab of the web interface but will not contain any software packages. Although you can upload software packages to the server in several ways, the method you choose will depend on whether the packages are available locally (e.g., DVD) or you want to synchronize a remote Yum repository with the Spacewalk server. If you choose the local upload, you can use the

Listing 2: Creating the Spacewalk User

```
sqlplus 'sys@xe as sysdba'
SQL> create user spacewalk identified by spacewalk
     default tablespace users;
SQL> grant dba to spacewalk;
SQL> quit
```

Listing 3: Oracle Tuning

```
sqlplus spacewalk/spacewalk@xe
SQL> alter system set processes = 400 scope=spfile;
SQL> alter system set "_optimizer_filter_pred_pullup"
     =false scope=spfile;
SQL> alter system set "_optimizer_cost_based_
     transformation"=off scope=spfile;
SQL> quit
```

Figure 1: The easiest approach to setting up a software channel is to use the web graphical interface.

Figure 2: Assigning individual users different privileges on the Spacewalk server.

`rhnpush` tool, which you launch as follows:

```
rhnpush -v --channel=fedora-13-i386 ➥
  --server=http://localhost/APP ➥
  --dir=/path/to/the/packages
```

To synchronize with a remote software repository, you simply need to specify the URL for the remote repository in the software channel properties in the web interface (*Channels* | *Manage Software Channels* | *Fedora 12 32-bit*). Synchronization can take a while to happen. Your other option here is the spacewalk-repo-sync command-line tool that downloads software packages from a Yum repository to your own Spacewalk server. To keep the server up to date, you can use cron to run a script [9] at regular intervals. This script will check your configured software sources and automatically download any new packages. This approach removes the need for manual synchronization.

Incidentally, you can use the method discussed here to set up subchannels, too. Note that any RPM packages you build yourself must be digitally signed. Both the Spacewalk server and the Yum client application will reject unsigned packages by default. Although you can disable this feature, it makes more sense to work with digital signatures for security reasons. The `rpm --resign RPM package` command will sign the package for you; you must have GPG keys in place for the RPM tool. The `~/.rpmmacros` file tells you the name and location of the key (Listing 5).

To allow client systems to verify packages signed with this key, you need to deposit the public key on the Spacewalk server, preferably in `/var/www/html/pub`, which any client can access. The following command exports the public key from the GPG keyring:

```
gpg --armor --export tscherf@redhat.com > ➥
    /var/www/html/pub/rpm-gpg-key
```

To allow the existing client systems to access the software packages you just uploaded, you need to register them with the Spacewalk server. Start by installing the Spacewalk Client Repository RPM on the clients. Fedora 12 systems have a matching RPM [10], as do RHEL5 and CentOS5 [11]. On RHEL and CentOS, you also need to install the RPM for the EPEL repository [12] because the client tool dependencies might not resolve correctly otherwise. The following command installs the Yum file on a 32-bit Fedora 12 system:

```
rpm -Uvh http://spacewalk.redhat.com/ ➥
  yum/1.0/Fedora/12/i386/spacewalk- ➥
  client-repo-1.0-2.fc12.noarch.rpm
```

Then, use Yum to install the client tools:

```
yum install rhn-client-tools ➥
  rhn-check rhn-setup rhnsd m2crypto ➥
  yum-rhn-plugin
```

The easiest approach to registering a system on the server is to run the `rhnreg_ks` tool, which expects a registration key. You need to create the key up front on the Spacewalk server (*Systems* | *Activation Key* | *Create Key*). When you create a key, you can bind various resources to it, such as the Fedora 12 software channel just created here, or various configuration channels, if you have created some (Figure 3). Also, you can assign system groups to the key. Systems that use this key to register are granted access to the associated resources. To do so, specify the key you created during the registration process:

Figure 3: Various resources can be bound to the registration key. Systems that use the key are given access to the associated resources.

Listing 4: Answer File

```
admin-email = root@localhost
ssl-set-org = Tuxgeek Org
ssl-set-org-unit = Tuxgeek OU
ssl-set-city = Essen
ssl-set-state = NRW
ssl-set-country = DE
ssl-password = spacewalk
ssl-set-email = root@localhost
ssl-config-sslvhost = Y
db-backend=oracle
db-user=spacewalk
db-password=spacewalk
db-sid=xe
db-host=localhost
db-port=1521
db-protocol=TCP
enable-tftp=Y
```

```
rhnreg_ks --serverUrl=➥
  http://spacewalk.server.tld/XMLRPC ➥
  --activationkey=key
```

If all of this worked correctly, you will see the system in the `Systems` tab of the server web interface. Viewing the system's properties should also show you the configured software channel. The easiest approach to checking whether access to the channel is working is to install a package from the channel. If this doesn't work, one possible issue could be that the client system is not using the Spacewalk server's CA certificate. The certificate is stored in *http://spacewalk.server.tld/pub/* on the server and must be stored in `/usr/share/rhn` on the client side. The `/etc/sysconfig/rhn/up2date` file needs a reference to the certificate.

As before, you need to enter the name of the Spacewalk server. You only need to perform these steps on systems you have already installed. Any that you install from scratch via the Spacewalk server are automatically registered with the server as part of the installation process and can thus access the server immediately (Figure 4).

Figure 4: After completing the registration, the system appears in the Spacewalk server's web interface.

Kickstart Installation

To automate the installation of new client systems, you need two pieces of information on the Spacewalk server. One of them is a kickstart file with details of how to install the new system, including partitioning, the software selection, and other settings that you would need to provide for a manual install. The easiest way to create a kickstart file is to select *Systems* | *Kickstart* | *Profiles* in the web front end.

After checking out the overview of existing profiles, you can also create a new profile. The kickstart distribution must be specified as part of the profile file. This does not mean the RPM files that belong to the distribution you want to install, such as Fedora 12, but the basic installation files, like the Anaconda tool.

The software repositories you synchronized earlier will not normally provide a kickstart distribution, and this means creating the distribution on the Spacewalk server. Again, just navigate to *Systems* | *Kickstart* | *Distributions* in the web interface and point to the required files. The easiest way to provide the files is to mount an installation CD/DVD for your preferred distribution via the loopback device:

```
mount -o loop ➥
  /var/iso-images/Fedora-23-i386-DVD.iso ➥
  /var/distro-trees/Fedora-12
```

When you create a Fedora 12 kickstart distribution, you simply point the Spacewalk server to the `/var/distro-trees/Fedora-12` directory. If all of this works out, just point to the distribution you created when you made the kickstart file. When a client system is installed from scratch, it will automatically pick up the right files from this source.

Although there are a number of ways to install a Fedora 12 system from scratch, the easiest approach is to point any client PXE requests by your clients to the Spacewalk server with the `next-server` command. Thanks to Cobbler [13] integration, the Spacewalk server has a TFTP server and any kickstart profiles that you have set up. To confirm this, you can type `cobbler profile list` at the command line.

When you boot a client system via a PXE-capable network card, you will automatically see a list of the existing

Figure 5: The system properties give you a neat option for handling a variety of administrative tasks for a system via the Spacewalk server.

Listing 5: GPG Configuration for RPM

```
cat .rpmmacros
%_signature gpg
%_gpg_name Thorsten Scherf <tscherf@redhat.com>
```

kickstart profiles. To install the client, simply select the required profile from the list. The client is then automatically registered on the Spacewalk server. Existing systems can easily be reinstalled using:

```
koan --replace-self ➥
     --server=Spacewalk-Server ➥
     --profile=Kickstart-Profile
```

This creates an entry in the system's bootloader menu and automatically selects the entry when the system reboots.

System Management

All of the systems registered on the Spacewalk server retrieve their software packages from this source, with no need to access external repositories. This method not only improves your security posture but also saves network bandwidth. With a registered system, you can customize various settings in the *System Properties* section (Figure 5).

For example, you can assign new software or configuration channels, compare the installed software with profiles on other systems, or create snapshots as a backup that you can roll back later. Additionally, you can install new software or distribute configuration files from a centralized location.

Thanks to the ability to assign registered systems to groups, you can point and click to do this for a large number of systems. The `rhnsd` service on the systems queries the Spacewalk server at predefined intervals to check for new actions, such as software installations.

When a system finds an action, it then executes it. If the `osad` service is enabled on the system, you can even run actions immediately without waiting for the polling interval to elapse. The client and the server then use the Jabber protocol for a continuous exchange.

Finally, don't forget the feature-rich Spacewalk API, which is accessible at *http://*Servername*/rhn/apidoc/index.jsp* on the installed server. This tool gives you access to a plethora of functions that are not available in the web interface.

The API can be accessed with XML-RPC, which makes it perfect for your own Perl or Python scripts. A Python script [8] for creating a software channel is just one example of accessing the Spacewalk server via the API (Figure 6).

Conclusions

Spacewalk gives administrators a very powerful tool for managing large-scale Linux landscapes. It facilitates many daily tasks, such as the installation of software updates or uploading of configuration files. Advanced features, such as channel cloning, make it possible to put any software through a quality assurance process before rolling it out to your production systems. Thanks to the comprehensive API, many tasks can also be scripted. ■

Figure 6: An XMPRPM interface opens up a huge selection of Spacewalk server functions via the programmable API.

Info

[1] Spacewalk project homepage: [https://fedorahosted.org/spacewalk]

[2] Spacewalk network ports: [http://magazine.redhat.com/2008/09/30/tips-and-tricks-what-tcpip-ports-are-required-to-be-open-on-an-rhn-satellite-proxy-or-client-system/]

[3] RHEL5, CentOS5 Spacewalk Server Repository RPM: [http://spacewalk.redhat.com/yum/1.0/RHEL/5/i386/spacewalk-repo-1.0-2.el5.noarch.rpm]

[4] Fedora12 Spacewalk Server Repository RPM: [http://spacewalk.redhat.com/yum/1.0/Fedora/12/i386/spacewalk-repo-1.0-2.fc12.noarch.rpm]

[5] Spacewalk Roadmap: [http://fedorahosted.org/spacewalk/roadmap]

[6] Spacewalk mailing list: [http://www.redhat.com/spacewalk/communicate.html#lists]

[7] Oracle XE: [http://www.oracle.com/technology/software/products/database/xe/htdocs/102xelinsoft.html]

[8] Spacewalk API script for creating a software channel: [http://fedorahosted.org/spacewalk/attachment/wiki/UploadFedoraContent/create_channel.py]

[9] Repository sync: [http://fedorahosted.org/spacewalk/attachment/wiki/UploadFedoraContent/sync_repos.py]

[10] Fedora12 Spacewalk Client Repository RPM: [http://spacewalk.redhat.com/yum/1.0/Fedora/12/i386/spacewalk-client-repo-1.0-2.fc12.noarch.rpm]

[11] RHEL5 and CentOS5 Client Repository RPM: [http://spacewalk.redhat.com/yum/1.0/RHEL/5/i386/spacewalk-client-repo-1.0-2.el5.noarch.rpm]

[12] EPEL Repository: [http://download.fedora.redhat.com/pub/epel/5/i386/epel-release-5-3.noarch.rpm]

[13] Cobbler: [https://fedorahosted.org/cobbler/]

The Author

Thorsten Scherf is a Senior Consultant for Red Hat EMEA. You can meet him as a speaker at conferences. He is also a keen marathon runner whenever time permits.