Automate and manage systems installation with Cobbler

Easily set up and administer a network installation environment with Cobbler

Paulo de Rezende Pinatti June 07, 2013

Cobbler simplifies system provisioning by centralizing the tasks that are involved in setting up and administering an installation server. This article discusses some of Cobbler's features, how to install it, and how to create a configuration suitable for automatically installing multiple client machines.

Setting up a network environment can involve many steps until everything is ready to start the installation. You must:

- Configure services such as DHCP, TFTP, DNS, HTTP, FTP, and NFS
- Fill individual client machine entries in DHCP and TFTP configuration files
- Create automatic deployment files (such as kickstart and autoinst)
- Extract installation media to HTTP/FTP/NFS repositories.

The process is not straightforward, and manual registration of each client machine that must be provisioned can be annoying. Any parameter change to the provisioning of a machine — such as a different operating system to be used — demands a manual intervention in configuration and possibly automatic deployment files. When the number of machines increases, elements like the TFTP directory can get messy unless you pay close attention to the files organization.

Cobbler addresses these shortcomings by creating a central point of management for all aspects of machine provisioning. It can reconfigure services, create repositories, extract operating system media, act or integrate with a configuration management system, control power management, and more. Cobbler creates a layer of abstraction where you can run commands like "add new repository" or "change client machine operating system." Cobbler takes care of everything — creating or updating configuration files, restarting services, or extracting media to newly created directories. The intention is to hide all the system-related issues so you can focus on the task itself.

This article covers how Cobbler is designed, its main features, and examples of how to use those features to provision machines quickly and easily. First, its features.

Tivoli Configuration Manager

IBM Tivoli® Configuration Manager is an integrated software distribution and asset management suite that consists of two main components, Software Distribution and Inventory, and various services. Tivoli Configuration Manager controls configuration, distribution, change, version, and asset management in a multiplatform environment.

What the tool offers

With Cobbler, you can install machines without manual intervention. Cobbler sets up a PXE boot environment (it also supports PowerPC by using yaboot) and controls all aspects that are related to installation, such as network boot services (DHCP and TFTP) and repository mirroring. When you want to install a new machine, Cobbler:

- Uses a previously defined template to configure the DHCP service (if manage DHCP is enabled)
- Mirrors a repository (yum or rsync) or extracts a media to register a new operating system
- Creates an entry in the DHCP configuration file for the machine to be installed with the parameters that you specify (IP and MAC addresses)
- Creates the appropriate PXE files under the TFTP service directory
- Restarts the DHCP service to reflect the changes
- Restarts the machine to begin installation, if power management was enabled

Cobbler has a good range of distribution support: Red Hat, Fedora, CentOS, Debian, Ubuntu, and SuSE. When you add an operating system (usually by using an ISO file), Cobbler knows how to extract the appropriate files and adjust the network services to boot the machines correctly.

Cobbler can work with kickstart templating. Systems that are based on Red Hat or Fedora use kickstart files to automate the installation process. By using templates, you can have base kickstart templates and then define how variables are replaced in them for a profile or machine configuration. For example, a template can contain two variables, *$domain* and *$machine_name*. In a Cobbler configuration, a profile specifies `domain=mydomain.com`, and each machine that uses this profile specifies its name in the *machine_name* variable. All machines in that profile are installed with the same kickstart and configured for `domain=mydomain.com`, while each has its own machine name. You can still use the kickstart template to install other machines in different domains and with different machine names.

To help manage systems, Cobbler can connect to various power management environments through *fence scripts*. Cobbler supports apc_snmp, bladecenter, bullpap, drac, ether_wake, ilo, integrity, ipmilan, ipmitool, lpar, rsa, virsh, and wti. To reinstall a machine, run a `reboot system foo` command, and Cobbler runs the appropriate fence script with the necessary credentials and information (such as the machine slot number) for you.

In addition to these features, you can use a configuration management system (CMS). You have two choices: an internal system in the tool, or an integration with an existing external CMS such as Chef or Puppet. With the internal system, you can specify file templates, which are processed according to configuration parameters (the same way the kickstart templates are) and copied

to the location you determine. This function is useful when you must deploy configuration files automatically to specific machines.

With the koan client, Cobbler can provision virtual machines and reinstall systems from the client side. I do not discuss the configuration management and koan features because they are beyond the scope of this article. However, they are useful features that are worth exploring. (For more information, see Related topics.)

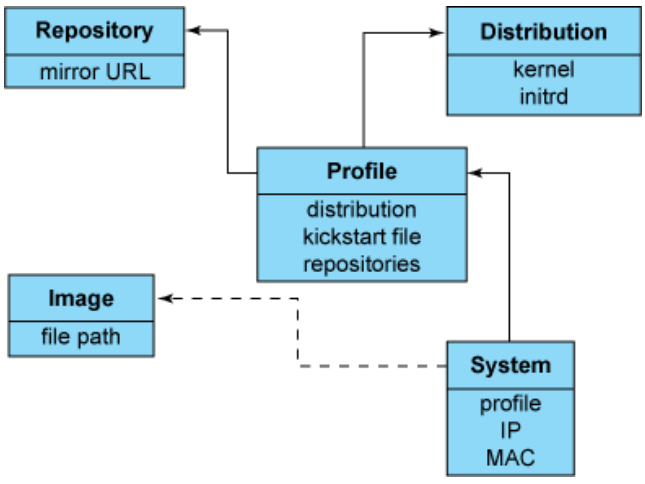
How Cobbler is designed

The configuration structure of Cobbler is based on a set of registered objects. Each object represents an entity that is associated with another entity (the object points to another object, or another object points to it). When one object points to another object, it inherits the data of the pointed object and can override or add more specific information. The following types of objects are defined:

- **Distribution**: Represents an operating system. It carries information about the kernel and initrd, plus other data such as kernel parameters.
- **Profile**: Points to a distribution, a kickstart file, and possibly repositories, plus other data such as more specific kernel parameters.
- **System**: Represents the machine to be provisioned. It points to a profile or an image and contains information about IP and MAC addresses, power management (address, credentials, type), and more specialized data.
- **Repository**: Holds mirroring information for a yum or rsync repository.
- **Image**: Can replace a distribution object for files that do not fit in this category (for example, cannot be divided in kernel and initrd).

Based on what objects are registered and the association between them, Cobbler knows how to change the file system to reflect the configuration. Because the internals of the system configuration are abstracted, you can focus just on what you want to do.

Figure 1. Cobbler objects relationship



Using Cobbler

The options to configure and use Cobbler include command line, API, XML-RPC, and web UI. I primarily focus on the command-line option.

EPEL packages

Cobbler packages for RHEL and CentOS are distributed through EPEL (Extra Packages for Enterprise Linux), a special interest group of the Fedora community. EPEL maintains a set of additional packages for Enterprise Linux, including the Red Hat Enterprise Linux (RHEL), CentOS, and Scientific Linux (SL) distributions. Find more information at the Fedora Project.

Cobbler has packages for the Fedora, RHEL, CentOS, Ubuntu, and OpenSuse distributions (see EPEL packages). I base the instructions on a Fedora system — you can easily adjust them to another system if you prefer.

First, install the tool and the fence-agents package, which Cobbler uses for power management activities. As the root user, issue the `install` command of the distribution's package manager:

```
yum -y install cobbler fence-agents
```

Once the packages are installed, start the necessary services:

```
service cobblerd start
service httpd start
```

To test whether Cobbler is running correctly, type `cobbler check`. This command reports configuration points that might need adjustment. Don't worry about them; the command is just a guidance to help users. You will configure the relevant aspects on the next steps (the only exceptions are the SELinux warnings, adjust them by following the instructions from Cobbler). If you happen to get a connection error message, verify that the services — and the SELinux logs, if enabled — were correctly started. You might need to set the SELinux boolean `httpd_can_network_connect_cobbler` to true (`setsebool -P httpd_can_network_connect_cobbler 1`). If so, remember to restart the services afterward.

Configuring Cobbler

The main Cobbler configuration file is /etc/cobbler/settings. Open it with your text editor, and set the following options:

- manage_dhcp: 1
- manage_dns: 1
- manage_tftpd: 1
- restart_dhcp: 1
- restart_dns: 1
- pxe_just_once: 1
- next_server: <server's IP address>
- server: <server's IP address>

The options `manage_*` and `restart_*` are self-explanatory. The option `next_server` is used in the DHCP configuration file to tell the machines what is the address of the server that provides the boot file. The option `server` is used during the machine installation to refer to the Cobbler server address. Finally, the option `pxe_just_once` prevents installation loops in machines that are configured to always boot from the network. When this option is activated, the machine tells Cobbler that the installation is finished. Cobbler changes the netboot flag of the system object to false, forcing the machine to boot from the local disk. You use this option later in the example.

Configuring only some services

Cobbler can be configured to manage only certain services if that fits your needs. For example, you might administer a file server, but another machine in the network that you cannot access serves DHCP. In this case, the DHCP administrator sets the filename option to ask for a boot file (pxelinux.0 for x86 or yaboot for PowerPC) and the `next_server` option to point to the IP address of your file server. The machines that are booted in the network ask for the boot file from your server. You then configure Cobbler to manage the TFTP service and register the machines in the tool so that it serves the appropriate files to those machines.

You might not activate all the options (see Configuring only some services). In this example, configure Cobbler to manage all the services, because that's a common scenario and demonstrates how to configure them.

Now that Cobbler knows which services to manage, tell it which programs to use. The options are:

- DHCP: ISC dhcpd or dnsmasq
- DNS: BIND or dnsmasq
- TFTP: in.tftpd or cobbler's internal TFTP

Using dnsmasq for DHCP and DNS is a good idea because configuring it is easy. Use in.tftpd because it's the system default. Edit the file /etc/cobbler/modules.conf with the settings in Listing 1:

Listing 1. Configuration settings

```
[dns]
module = manage_dnsmasq

[dhcp]
module = manage_dnsmasq

[tftpd]
module = manage_in_tftpd
```

Cobbler uses a template to create the configuration files for the services. Editing the dnsmasq template at /etc/cobbler/dnsmasq.template is necessary to adjust network information such as the gateway address and IP range to be used. Assume the server that runs Cobbler is also the gateway, and our IP range is 192.168.122.5-192.168.122.254, enter this line in the file:

```
dhcp-range=192.168.122.5,192.168.122.254,255.255.255.0
```

Usually you want to block unregistered clients from booting from the server. To do that, add the parameter `dhcp-ignore=tag:!known`. (In older versions, the syntax might be different: `dhcp-`

`ignore=#known`. If in doubt, you can insert both versions.) The file content is similar to the code in Listing 2:

Listing 2. dnsmasq template file

```
# Cobbler generated configuration file for dnsmasq
# $date
#

# resolve.conf .. ?
#no-poll
#enable-dbus
read-ethers
addn-hosts = /var/lib/cobbler/cobbler_hosts

dhcp-range=192.168.122.5,192.168.122.254,255.255.255.0
dhcp-ignore=tag:!known
dhcp-option=3,$next_server
dhcp-lease-max=1000
dhcp-authoritative
dhcp-boot=pxelinux.0
dhcp-boot=net:normalarch,pxelinux.0
dhcp-boot=net:ia64,$elilo

$insert_cobbler_system_definitions
```

Cobbler is almost ready for use. Restart the service and synchronize the changes to the filesystem for them to take effect. Remember also to restart the `xinetd` service to make TFTP available. Run the commands:

```
service cobblerd restart
cobbler sync
service xinetd restart
```

You can add distributions and repositories, create profiles, and register systems. Remember to verify that your firewall configuration allows traffic on the ports that the network services TFTP, DHCP, and HTTP/HTTPS use.

Installing Fedora 17 systems

Prepare Cobbler to install Fedora 17 systems with two available options: Xfce or GNOME desktop. To add the Fedora installation tree, first download the ISO media and then run the following commands to mount the media and extract its content. (In some systems, Cobbler might not see the mounted directory content and fails to import the media. If this problem happens to you, restart the Cobbler service after the mount command.):

```
mount -o loop /Fedora-17-x86_64-DVD.iso /mnt/iso
cobbler import --arch=x86_64 --path=/mnt/iso --name=Fedora17
```

Cobbler now copies the media content to the filesystem. Be patient: The operation can take a while to complete. The command `cobbler import` is handy because it automatically creates a distribution and a profile object for you. You can also point Cobbler directly to a network repository. The result is similar to Listing 3:

Listing 3. Result of the `cobbler import` command

```
cobbler distro report
Name                           : Fedora17-x86_64
Architecture                   : x86_64
TFTP Boot Files                : {}
Breed                          : redhat
Comment                        :
Fetchable Files                : {}
Initrd                         : /var/www/cobbler/ks_mirror/Fedora17-x86_64/images/
                                  pxeboot/initrd.img
Kernel                         : /var/www/cobbler/ks_mirror/Fedora17-x86_64/images/
                                  pxeboot/vmlinuz
Kernel Options                 : {}
Kernel Options (Post Install)  : {}
Kickstart Metadata             : {'tree': 'http://@@http_server@@/cblr/links/
                                  Fedora17-x86_64'}
Management Classes             : []
OS Version                     : generic26
Owners                         : ['admin']
Red Hat Management Key         : <<inherit>>
Red Hat Management Server      : <<inherit>>
Template Files                 : {}

cobbler profile report

Name                           : Fedora17-x86_64
TFTP Boot Files                : {}
Comment                        :
DHCP Tag                       : default
Distribution                   : Fedora17-x86_64
Enable gPXE?                   : 0
Enable PXE Menu?               : 1
Fetchable Files                : {}
Kernel Options                 : {}
Kernel Options (Post Install)  : {}
Kickstart                      :
Kickstart Metadata             : {}
Management Classes             : []
Management Parameters          : <<inherit>>
Name Servers                   : []
Name Servers Search Path       : []
Owners                         : ['admin']
Parent Profile                 :
Proxy                          :
Red Hat Management Key         : <<inherit>>
Red Hat Management Server      : <<inherit>>
Repos                          : []
Server Override                : <<inherit>>
Template Files                 : {}
Virt Auto Boot                 : 1
Virt Bridge                    : xenbr0
Virt CPUs                      : 1
Virt Disk Driver Type          : raw
Virt File Size(GB)             : 5
Virt Path                      :
Virt RAM (MB)                  : 512
Virt Type                      : qemu
```

This profile is the parent of the two other profiles that you will create for each desktop you want installed.

Before you do that, however, consider that you have a yum repository with more packages to use in the installations. To do that, create a repository object:

```
cobbler repo add --arch=x86_64 --name=Flash-plugin \
  --mirror=http://linuxdownload.adobe.com/linux/x86_64/
cobbler reposync
cobbler repo report
```

For the yum repository URL, Cobbler accepts http://, ftp://, rsync://, filesystem paths, and ssh locations (by using private key-based authentication). The `reposync` operation is important because it copies files from the remote repository. If you create the repository object but do not run `reposync`, your repository is empty and your installation is likely to fail.

To complete the repository activation, associate the repository to a profile. Associate it to the Fedora profile with the command:

```
cobbler profile edit --name=Fedora17-x86_64 --repos=Flash-plugin
```

Creating profiles

The next step is to create the profiles. For an automatic installation, specify a kickstart file with the kickstart template feature. Create a simple kickstart that is based on the files available from /var/lib/cobbler/kickstarts. Then, in the %packages section, define a variable *$desktop_pkg_group*, which later is substituted to determine which desktop packages are installed. Listing 4 shows the content of the kickstart file:

Listing 4. Contents of the kickstart file

```
# System bootloader configuration
bootloader --location=mbr
# Partition clearing information
clearpart --all --initlabel
# Run the Setup Agent on first boot
firstboot --disable
# Activate X
xconfig --startxonboot
# Use network installation
url --url=$tree
# additional repostories get added here
$yum_repo_stanza
# Reboot after installation
reboot
# System keyboard
keyboard us
# System language
lang en_US
# System timezone
timezone  America/New_York
# Root password
rootpw --iscrypted $default_password_crypted
# Install OS instead of upgrade
install
# Clear the Master Boot Record
zerombr
# Allow anaconda to partition the system as needed
autopart

%packages
@base
@base-x
firefox
flash-plugin
```

```
$desktop_pkg_group
%end

%post
# create a default user to log in X
useradd desktop-user
passwd -d desktop-user

# adds the yum repositories to the installed system
$yum_config_stanza
# cobbler final steps
$SNIPPET('kickstart_done')
%end
```

The variables that start with *$* are replaced by the Cheetah program (see Related topics), which Cobbler uses to process its templates. If you are familiar with Cheetah templates, the rules are the same. To learn more about the internal Cobbler variables such as *$yum_config_stanza*, look at the available kickstarts from /var/lib/cobbler/kickstarts.

After you create the file, copy it to /var/lib/cobbler/kickstarts (if you don't, Cobbler might fail to use it). Cobbler knows the value of *$desktop_pkg_group* because you define it when you create the profiles with the `--ksmeta` option. With this option, you determine the value to use when you replace a variable in the kickstart template. The commands in Listing 5 create the Xfce and GNOME profiles:

Listing 5. Commands to create the Xfce and GNOME profiles

```
cobbler profile add --name=Fedora17-xfce \
                    --ksmeta='desktop_pkg_group=@xfce-desktop' \
                    --kickstart=/var/lib/cobbler/kickstarts/example.ks \
                    --parent=Fedora17-x86_64
cobbler profile add --name=Fedora17-gnome \
                    --ksmeta='desktop_pkg_group=@gnome-desktop' \
                    --kickstart=/var/lib/cobbler/kickstarts/example.ks \
                    --parent=Fedora17-x86_64
cobbler profile report
```

The `--parent` parameter tells these profiles to inherit from the Fedora profile. The profiles use the Fedora distribution and the additional Flash-plugin repository. To make sure that everything is correct, you can verify the kickstart content after it processes. The result is similar to Listing 6:

Listing 6. Verifying kickstart content

```
cobbler profile getks --name=Fedora17-xfce

# System bootloader configuration
bootloader --location=mbr
# Partition clearing information
clearpart --all --initlabel
# Run the Setup Agent on first boot
firstboot --disable
# Activate X
xconfig --startxonboot
# Use network installation
url --url=http://192.168.122.1/cblr/links/Fedora17-x86_64
# additional repostories get added here
repo --name=Flash-plugin --baseurl=http://192.168.122.1/cobbler/repo_mirror/Flash-plugin
repo --name=source-1 --baseurl=http://192.168.122.1/cobbler/ks_mirror/Fedora17-x86_64
```

```
# Reboot after installation
reboot
# System keyboard
keyboard us
# System language
lang en_US
# System timezone
timezone  America/New_York
# Root password
rootpw --iscrypted $1$mF86/UHC$WvcIcX2t6crBz2onWxyac.
# Install OS instead of upgrade
install
# Clear the Master Boot Record
zerombr
# Allow anaconda to partition the system as needed
autopart

%packages
@base
@base-x
firefox
flash-plugin
@xfce-desktop
%end

%post
# create a user we can use to log on X
useradd desktop-user
passwd -d desktop-user

# adds the yum repositories to the installed system
wget "http://192.168.122.1/cblr/svc/op/yum/profile/Fedora17-xfce" \
   --output-document=/etc/yum.repos.d/cobbler-config.repo

# cobbler final steps


wget "http://192.168.122.1/cblr/svc/op/ks/profile/Fedora17-xfce" -O /root/cobbler.ks
wget "http://192.168.122.1/cblr/svc/op/trig/mode/post/profile/Fedora17-xfce" -O /dev/null
%end
```

The variable *$dekstop_pkg_group* was correctly replaced by *@xfce-desktop*, which tells the Anaconda installer to install the Xfce desktop package group.

Associating machines with the profiles

You are almost ready to start an installation. The last step is to associate the machines (one for each desktop) with the profiles you want them installed with. The commands are in Listing 7:

Listing 7. Associating machines with their profiles

```
cobbler system add --name=desktop-xfce-1 \
                   --profile=Fedora17-xfce \
                   --mac=52:54:00:b8:5e:8f \
                   --ip-address=192.168.122.10
cobbler system add --name=desktop-gnome-1 \
                   --profile=Fedora17-gnome \
                   --mac=52:54:00:88:f3:44 \
                   --ip-address=192.168.122.11
cobbler system report
```

The power management feature in Cobbler can power on, power off, and reboot the machines for you. This function is also useful when you have many machines and must organize power

management information such as user and passwords for each because Cobbler registers them in its base. Suppose machine desktop-xfce-1 is in an IBM Bladecenter at bay 2, and desktop-gnome-1 is a machine that is managed with an RSA board. You can set up your systems as shown in Listing 8:

Listing 8. Adding power management information

```
cobbler system edit --name=desktop-xfce-1 \
                    --power-type=bladecenter \
                    --power-id=2 \
                    --power-user=admin_user \
                    --power-pass=admin_password \
                    --power-address=192.168.122.2
cobbler system edit --name=desktop-gnome-1 \
                    --power-type=rsa \
                    --power-user=rsa_user \
                    --power-pass=rsa_password \
                    --power-address=192.168.122.3
```

Remember to apply all the changes to the filesystem:

```
cobbler sync
```

Finally, you can install the machines.

Starting the installation

You're ready to boot the machines and install them. They must be configured to boot from the network — if they aren't, they might boot from the hard disk and never start the installation. If you activated power management, Cobbler can reboot the machine for you so you can start the installation with a simple command:

```
cobbler system reboot --name=desktop-xfce-1
```

This command makes Cobbler connect to the Bladecenter with the credentials you specified and issue a reboot command for blade 2. The blade restarts with a network boot and receives the boot files from Cobbler. The installation takes place automatically, and the process ends when the Fedora login screen is displayed.

Cobbler even easier: the web interface

You might want to easily visualize the Cobbler objects and reuse object values for repetitive tasks on a daily basis. Cobbler offers a useful web interface so you can do that. To use it, start by installing its package:

```
yum -y install cobbler-web
```

After you install the package, configure the Cobbler authorization and authentication system so you can log in. The configuration is available in the file /etc/cobbler/modules.conf, and it looks like the code in Listing 9:

Listing 9. Default Cobbler authorization and authentication system configuration

```
# authentication:
# what users can log into the WebUI and Read-Write XMLRPC?
# choices:
#    authn_denyall    -- no one (default)
#    authn_configfile -- use /etc/cobbler/users.digest (for basic setups)
#    authn_passthru   -- ask Apache to handle it (used for kerberos)
#    authn_ldap       -- authenticate against LDAP
#    authn_spacewalk  -- ask Spacewalk/Satellite (experimental)
#    authn_pam        -- use PAM facilities
#    authn_testing    -- username/password is always testing/testing (debug)
#    (user supplied)  -- you may write your own module
# WARNING: this is a security setting,  do not choose an option blindly.
# for more information:
# https://github.com/cobbler/cobbler/wiki/Cobbler-web-interface
# https://github.com/cobbler/cobbler/wiki/Security-overview
# https://github.com/cobbler/cobbler/wiki/Kerberos
# https://github.com/cobbler/cobbler/wiki/Ldap

[authentication]

module = authn_denyall

# authorization:
# once a user has been cleared by the WebUI/XMLRPC, what can they do?
# choices:
#    authz_allowall   -- full access for all authneticated users (default)
#    authz_ownership  -- use users.conf, but add object ownership semantics
#    (user supplied)  -- you may write your own module
# WARNING: this is a security setting  do not choose an option blindly.
# If you want to further restrict cobbler with ACLs for various groups,
# pick authz_ownership.  authz_allowall does not support ACLs.  configfile
# does but does not support object ownership which is useful as an additional
# layer of control.

# for more information:
# https://github.com/cobbler/cobbler/wiki/Cobbler-web-interface
# https://github.com/cobbler/cobbler/wiki/Security-overview
# https://github.com/cobbler/cobbler/wiki/Web-authorization

[authorization]

module = authz_allowall
```

The help comments in Listing 9 show that authentication options such as LDAP, PAM, and configuration file are available. Because PAM is fairly common, use that for authentication. In the authorization section, define which users have clearance to use the tool. Set the module value to `authz_ownership` so you can specify in the users.conf file who are able to access the web interface. The configuration looks like the code in Listing 10:

Listing 10. Authentication and authorization configuration for the Cobbler web interface

```
[authentication]

module = authn_pam

[authorization]
module = authz_ownership
```

Save the file. Next, you need a system user named `myuser` (if you don't have one, create it with `useradd myuser && passwd myuser`). Then, open the file /etc/cobbler/users.conf and add `myuser` to the admins group (this group has full access to objects), as in Listing 11:

Listing 11. Adding `myuser` to the admins group in the authorization file

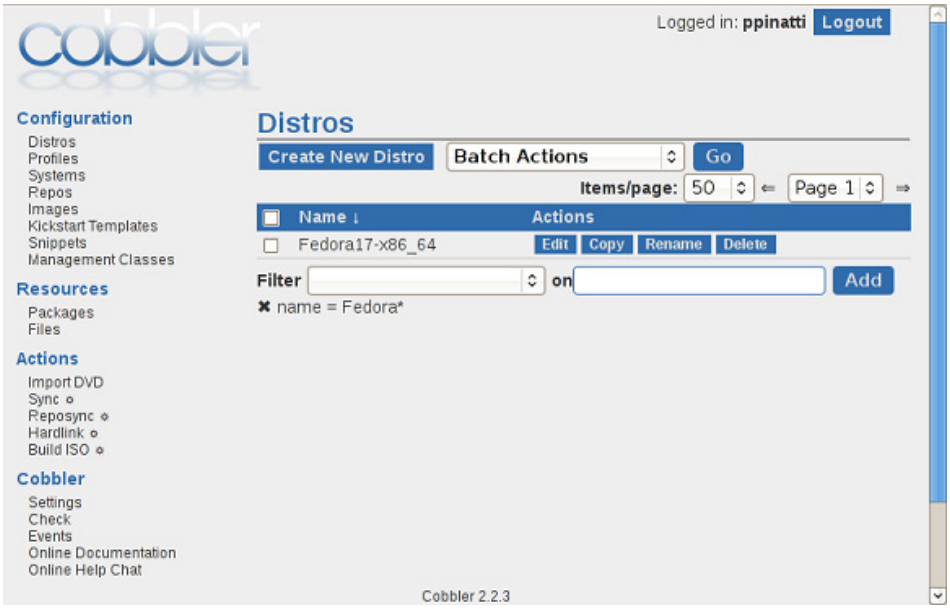
```
# Cobbler WebUI / Web Services authorization config file
#
# NOTICE:
# this file is only used when /etc/cobbler/modules.conf
# specifies an authorization mode of either:
#
#   (A) authz_configfile
#   (B) authz_ownership
#
# For (A), any user in this file, in any group, are allowed
# full access to any object in cobbler configuration.
#
# For (B), users in the "admins" group are allowed full access
# to any object, otherwise users can only edit an object if
# their username/group is listed as an owner of that object. If a
# user is not listed in this file they will have no access.
#
#     cobbler command line example:
#
#     cobbler system edit --name=server1 --owner=dbas,mac,pete,jack
#
# NOTE:  yes, you do need the equal sign after the names.
# don't remove that part.  It's reserved for future use.

[admins]

myuser = ""
```

The configuration is done. Now restart the Cobbler and Apache services to apply the changes:

```
service cobblerd restart
service httpd restart
```

The web interface is straightforward (see Figure 2): The menu on the left shows the configuration classes (such as repositories, systems, distributions, and profiles), resources (for configuration management), and actions (import, sync). Click a configuration class to list all objects on the right side of the screen. You can apply list filters and do different actions through the buttons (Edit, Copy, Rename, Delete) next to each item.

Figure 2. Cobbler web interface



Conclusion

Cobbler, a tool for automating and facilitating systems installations, uses network boot to control and initiate the installations. Other Cobbler features include repository mirroring, kickstart templating, and connectivity with power management systems.

A description of the configuration objects of the tool and how they relate to each other demonstrates Cobbler's internal design. The structure, which is based on associated objects and inheritance between them, provides good abstraction and reuse, which facilitates the task of configuring an environment for systems installation.

You learned how to install and configure Cobbler, plus how to use its commands to create a configuration suitable for automatically installing machines. Finally, you learned to install and configure the web interface as a practical alternative to perform daily activities.

Cobbler can simplify the life of system administrators. The features not explored in the article, including the XML-RPC API, configuration management, and the koan client, make Cobbler even more powerful and point the way to further exploration.

Related topics

- "Linux initial RAM disk (initrd) overview" (M. Tim Jones, developerWorks, July 2006): Learn about the initrd anatomy, creation, and use in the Linux boot process.
- Cheetah: Find out more about this open source template engine and code-generation tool that is used primarily for web development.
- The developerWorks Linux zone has more resources for Linux developers, including programmers who are new to Linux.
- The Linux technical library: Find more Linux articles, including the most popular.
- Download IBM developer kits: Update your system and get the latest tools and technologies here.