Cobbler Documentation

Release 3.2.2

Enno Gotthold

Oct 06, 2021

Contents

Cobbler is a provisioning (installation) and update server. It supports deployments via PXE (network booting), virtualization (Xen, QEMU/KVM, or VMware), and re-installs of existing Linux systems. The latter two features are enabled by usage of 'Koan' on the remote system. Update server features include yum mirroring and integration of those mirrors with automated installation files. Cobbler has a command line interface, WebUI, and extensive Python and XML-RPC APIs for integration with external scripts and applications.

If you want to explore tools or scripts which are using Cobbler please use the GitHub Topic: https://github.com/topics/cobbler

Here you should find a comprehensive overview about the usage of Cobbler.

CHAPTER 1

Quickstart

Cobbler can be a somewhat complex system to get started with, due to the wide variety of technologies it is designed to manage, but it does support a great deal of functionality immediately after installation with little to no customization needed. Before getting started with Cobbler, you should have a good working knowledge of PXE as well as the automated installation methodology of your chosen distribution(s).

We will assume you have successfully installed Cobbler, please refer to the *Installation Guide* for instructions for your specific operating system. Finally, this part guide will focus only on the CLI application.

1.1 Preparing your OS

1.1.1 SELinux

Before getting started with Cobbler, it may be convenient to either disable SELinux or set it to "permissive" mode, especially if you are unfamiliar with SELinux troubleshooting or modifying SELinux policy. Cobbler constantly evolves to assist in managing new system technologies, and the policy that ships with your OS can sometimes lag behind the feature-set we provide, resulting in AVC denials that break Cobbler's functionality.

1.1.2 Firewall

TBD

1.2 Changing settings

Before starting the *cobblerd* service, there are a few things you should modify.

Settings are stored in `/etc/cobbler/settings.yaml`. This file is a YAML formatted data file, so be sure to take care when editing this file as an incorrectly formatted file will prevent *cobblerd* from running.

1.2.1 Default encrypted password

This setting controls the root password that is set for new systems during the handsoff installation.

```
default_password_crypted: "$1$bfI7WLZz$PxXetL97LkScqJFxnW7KS1"
```

You should modify this by running the following command and inserting the output into the above string (be sure to save the quote marks):

```
$ openssl passwd -1
```

1.2.2 Server and next_server

The `server` option sets the IP that will be used for the address of the Cobbler server. **DO NOT** use 0.0.0.0, as it is not the listening address. This should be set to the IP you want hosts that are being built to contact the Cobbler server on for such protocols as HTTP and TFTP.

```
server: 127.0.0.1
```

The `next_server` option is used for DHCP/PXE as the IP of the TFTP server from which network boot files are downloaded. Usually, this will be the same IP as the server setting.

```
next_server: 127.0.0.1
```

1.3 DHCP management and DHCP server template

In order to PXE boot, you need a DHCP server to hand out addresses and direct the booting system to the TFTP server where it can download the network boot files. Cobbler can manage this for you, via the `manage_dhcp` setting:

```
manage_dhcp: 0
```

Change that setting to 1 so Cobbler will generate the `dhcpd.conf` file based on the `dhcp.template` that is included with Cobbler. This template will most likely need to be modified as well, based on your network settings:

```
$ vi /etc/cobbler/dhcp.template
```

For most uses, you'll only need to modify this block:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
    option routers             192.168.1.1;
    option domain-name-servers 192.168.1.210,192.168.1.211;
    option subnet-mask         255.255.255.0;
    filename                   "/pxelinux.0";
    default-lease-time         21600;
    max-lease-time             43200;
    next-server                $next_server;
}
```

No matter what, make sure you do not modify the `next-server $next_server;` line, as that is how the next_server setting is pulled into the configuration. This file is a cheetah template, so be sure not to modify anything starting after this line:

```
#for dhcp_tag in $dhcp_tags.keys():
```

Completely going through the `dhcpd.conf` configuration syntax is beyond the scope of this document, but for more information see the man page for more details:

```
$ man dhcpd.conf
```

1.4 Notes on files and directories

Cobbler makes heavy use of the `/var` directory. The `/var/www/cobbler/distro_mirror` directory is where all of the distribution and repository files are copied, so you will need 5-10GB of free space per distribution you wish to import.

If you have installed Cobbler onto a system that has very little free space in the partition containing `/var`, please read the *Relocating your installation* section of the Installation Guide to learn how you can relocate your installation properly.

1.5 Starting and enabling the Cobbler service

Once you have updated your settings, you're ready to start the service:

```
$ systemctl start cobblerd.service
$ systemctl enable cobblerd.service
$ systemctl status cobblerd.service
```

If everything has gone well, you should see output from the status command like this:

```
cobblerd.service - Cobbler Helper Daemon
    Loaded: loaded (/lib/systemd/system/cobblerd.service; enabled)
      Active: active (running) since Sun, 17 Jun 2012 13:01:28 -0500; 1min 44s ago
    Main PID: 1234 (cobblerd)
      CGroup: name=systemd:/system/cobblerd.service
              └ 1234 /usr/bin/python /usr/bin/cobblerd -F
```

1.6 Checking for problems and your first sync

Now that the *cobblerd* service is up and running, it's time to check for problems. Cobbler's check command will make some suggestions, but it is important to remember that these are mainly only suggestions and probably aren't critical for basic functionality. If you are running iptables or SELinux, it is important to review any messages concerning those that check may report.

```
$ cobbler check
The following are potential configuration items that you may want to fix:

1. ....
2. ....
```

Restart *cobblerd* and then run `cobbler sync` to apply changes.

If you decide to follow any of the suggestions, such as installing extra packages, making configuration changes, etc., be sure to restart the *cobblerd* service as it suggests so the changes are applied.

Once you are done reviewing the output of `cobbler check`, it is time to synchronize things for the first time. This is not critical, but a failure to properly sync at this point can reveal a configuration problem.

```
$ cobbler sync
task started: 2012-06-24_224243_sync
task started (id=Sync, time=Sun Jun 24 22:42:43 2012)
running pre-sync triggers
...
rendering DHCP files
generating /etc/dhcp/dhcpd.conf
cleaning link caches
running: find /var/lib/tftpboot/images/.link_cache -maxdepth 1 -type f -links 1 -
↪exec rm -f '{}' ';'
```

(continues on next page)

(continued from previous page)

```
received on stdout:
received on stderr:
running post-sync triggers
running python triggers from /var/lib/cobbler/triggers/sync/post/*
running python trigger cobbler.modules.sync_post_restart_services
running: dhcpd -t -q
received on stdout:
received on stderr:
running: service dhcpd restart
received on stdout:
received on stderr:
running shell triggers from /var/lib/cobbler/triggers/sync/post/*
running python triggers from /var/lib/cobbler/triggers/change/*
running python trigger cobbler.modules.scm_track
running shell triggers from /var/lib/cobbler/triggers/change/*
*** TASK COMPLETE ***
```

Assuming all went well and no errors were reported, you are ready to move on to the next step.

1.7 Importing your first distribution

Cobbler automates adding distributions and profiles via the `cobbler import` command. This command can (usually) automatically detect the type and version of the distribution your importing and create (one or more) profiles with the correct settings for you.

1.7.1 Download an ISO image

In order to import a distribution, you will need a DVD ISO for your distribution.

Note: You must use a full DVD, and not a "Live CD" ISO. For this example, we'll be using the Fedora 17 x86_64 ISO.

Warning: When running Cobbler via systemd, you cannot mount the ISO to `/tmp` or a sub-folder of it because we are using the option *Private Temporary Directory*, to enhance the security of our application.

Once this file is downloaded, mount it somewhere:

```
$ mount -t iso9660 -o loop,ro /path/to/isos/Fedora-17-x86_64-DVD.iso /mnt
```

1.7.2 Run the import

You are now ready to import the distribution. The name and path arguments are the only required options for import:

```
$ cobbler import --name=fedora17 --arch=x86_64 --path=/mnt
```

The `--arch` option need not be specified, as it will normally be auto-detected. We're doing so in this example in order to prevent multiple architectures from being found.

Listing objects

If no errors were reported during the import, you can view details about the distros and profiles that were created during the import.

```
$ cobbler distro list
$ cobbler profile list
```

The import command will typically create at least one distro/profile pair, which will have the same name as shown above. In some cases (for instance when a Xen-based kernel is found), more than one distro/profile pair will be created.

Object details

The report command shows the details of objects in Cobbler:

```
$ cobbler distro report --name=fedora17-x86_64
Name                           : fedora17-x86_64
Architecture                   : x86_64
TFTP Boot Files                : {}
Breed                          : redhat
Comment                        :
Fetchable Files                : {}
Initrd                         : /var/www/cobbler/distro_mirror/fedora17-x86_64/
↪images/pxeboot/initrd.img
Kernel                         : /var/www/cobbler/distro_mirror/fedora17-x86_64/
↪images/pxeboot/vmlinuz
Kernel Options                 : {}
Kernel Options (Post Install)  : {}
Automatic Installation Template Metadata : {'tree': 'http://@@http_server@@/cblr/
↪links/fedora17-x86_64'}
Management Classes             : []
OS Version                     : fedora17
Owners                         : ['admin']
Red Hat Management Key         : <<inherit>>
Red Hat Management Server      : <<inherit>>
Template Files                 : {}
```

As you can see above, the import command filled out quite a few fields automatically, such as the breed, OS version, and initrd/kernel file locations. The "Automatic Installation Template Metadata" field (`--autoinstall_meta` internally) is used for miscellaneous variables, and contains the critical "tree" variable. This is used in the automated installation templates to specify the URL where the installation files can be found.

Something else to note: some fields are set to `<<inherit>>`. This means they will use either the default setting (found in the settings file), or (in the case of profiles, sub-profiles, and systems) will use whatever is set in the parent object.

Creating a system

Now that you have a distro and profile, you can create a system. Profiles can be used to PXE boot, but most of the features in Cobbler revolve around system objects. The more information you give about a system, the more Cobbler will do automatically for you.

First, we'll create a system object based on the profile that was created during the import. When creating a system, the name and profile are the only two required fields:

```
$ cobbler system add --name=test --profile=fedora17-x86_64
$ cobbler system list
test
```

(continues on next page)

(continued from previous page)

```
$ cobbler system report --name=test
Name                           : test
TFTP Boot Files                : {}
Comment                        :
Enable gPXE?                   : 0
Fetchable Files                : {}
Gateway                        :
Hostname                       :
Image                          :
IPv6 Autoconfiguration         : False
IPv6 Default Device            :
Kernel Options                 : {}
Kernel Options (Post Install)  : {}
Automatic Installation Template: <<inherit>>
Automatic Installation Template Metadata: {}
Management Classes             : []
Management Parameters          : <<inherit>>
Name Servers                   : []
Name Servers Search Path       : []
Netboot Enabled                : True
Owners                         : ['admin']
Power Management Address       :
Power Management ID            :
Power Management Password      :
Power Management Type          : ipmilanplus
Power Management Username      :
Profile                        : fedora17-x86_64
Proxy                          : <<inherit>>
Red Hat Management Key         : <<inherit>>
Red Hat Management Server      : <<inherit>>
Repos Enabled                  : False
Server Override                : <<inherit>>
Status                         : production
Template Files                 : {}
Virt Auto Boot                 : <<inherit>>
Virt CPUs                      : <<inherit>>
Virt Disk Driver Type          : <<inherit>>
Virt File Size(GB)             : <<inherit>>
Virt Path                      : <<inherit>>
Virt RAM (MB)                  : <<inherit>>
Virt Type                      : <<inherit>>
```

The primary reason for creating a system object is network configuration. When using profiles, you're limited to DHCP interfaces, but with systems you can specify many more network configuration options.

So now we'll setup a single, simple interface in the `192.168.1/24` network:

```
$ cobbler system edit --name=test --interface=eth0 --mac=00:11:22:AA:BB:CC --ip-
↪address=192.168.1.100 --netmask=255.255.255.0 --static=1 --dns-name=test.
↪mydomain.com
```

The default gateway isn't specified per-NIC, so just add that separately (along with the hostname):

```
$ cobbler system edit --name=test --gateway=192.168.1.1 --hostname=test.mydomain.
↪com
```

The `--hostname` field corresponds to the local system name and is returned by the `hostname` command. The `--dns-name` (which can be set per-NIC) should correspond to a DNS A-record tied to the IP of that interface. Neither are required, but it is a good practice to specify both. Some advanced features (like configuration management) rely on the `--dns-name` field for system record look-ups.

Whenever a system is edited, Cobbler executes what is known as a "lite sync", which regenerates critical files like

the PXE boot file in the TFTP root directory. One thing it will **NOT** do is execute service management actions, like regenerating the `dhcpd.conf` and restarting the DHCP service. After adding a system with a static interface it is a good idea to execute a full `cobbler sync` to ensure the dhcpd.conf file is rewritten with the correct static lease and the service is bounced.

CHAPTER 2

Install Guide

Setting up and running *cobblerd* is not a easy task. Knowledge in apache configuration (setting up ssl, virtual hosts, apache module and wsgi) is needed. Certificates and some server administration knowledge is required too.

Cobbler is available for installation in several different ways, through packaging systems for each distribution or directly from source.

Cobbler has both definite and optional prerequisites, based on the features you'd like to use. This section documents the definite prerequisites for both a basic installation and when building/installing from source.

2.1 Prerequisites

2.1.1 Packages

Please note that installing any of the packages here via a package manager (such as dnf/yum or apt) can and will require a large number of ancilary packages, which we do not document here. The package definition should automatically pull these packages in and install them along with Cobbler, however it is always best to verify these requirements have been met prior to installing Cobbler or any of its components.

First and foremost, Cobbler requires Python. Since 3.0.0 you will need Python 3. Cobbler also requires the installation of the following packages:

- createrepo_c
- httpd / apache2
- xorriso
- mod_wsgi / libapache2-mod-wsgi
- mod_ssl / libapache2-mod-ssl
- python-cheetah
- python-netaddr
- python-simplejson
- python-librepo
- PyYAML / python-yaml

- rsync
- syslinux
- tftp-server / atftpd
- dnf-plugins-core

If you decide to use the LDAP authentication, please also install manually in any case:

- python3-ldap3 (or via PyPi: ldap3)

Cobbler-web only has one other requirement besides Cobbler itself:

- Django / python-django

Koan can be installed apart from Cobbler, and has only the following requirement (besides python itself of course):

- python-simplejson

Note: Not installing all required dependencies will lead to stacktraces in your Cobbler installation.

2.1.2 Source

Note: Please be aware that on some distributions the python packages are named differently. On Debian based systems everything which is named `something-devel` is named `something-dev` there. Also please remember that the case of some packages is slightly different.

Warning: Some distributions still have Python 2 available. It is your responsibility to adjust the package names to Python3.

Installation from source requires the following additional software:

- git
- make
- python3-devel (on Debian based distributions `python3-dev`)
- python3-Cheetah3
- python3-future
- python3-Sphinx
- python3-coverage
- openssl
- apache2-devel (and thus apache2)
- A TFTP server

2.2 Installation

Cobbler is available for installation for many Linux variants through their native packaging systems. However, the Cobbler project also provides packages for all supported distributions which is the preferred method of installation.

2.2.1 Packages

We leave packaging to downstream; this means you have to check the repositories provided by your distribution vendor. However we provide docker files for

- CentOS 7
- CentOS 8
- Debian 10 Buster

which will give you packages which will work better then building from source yourself.

2.2.2 Packages from source

For some platforms it's also possible to build packages directly from the source tree.

2.3 RPM

```
$ make rpms
... (lots of output) ...
Wrote: /path/to/cobbler/rpm-build/cobbler-3.0.0-1.fc20.src.rpm
Wrote: /path/to/cobbler/rpm-build/cobbler-3.0.0-1.fc20.noarch.rpm
Wrote: /path/to/cobbler/rpm-build/koan-3.0.0-1.fc20.noarch.rpm
Wrote: /path/to/cobbler/rpm-build/cobbler-web-3.0.0-1.fc20.noarch.rpm
```

As you can see, an RPM is output for each component of Cobbler, as well as a source RPM. This command was run on a system running Fedora 20, hence the fc20 in the RPM name - this will be different based on the distribution you're running.

2.4 DEB

To install Cobbler from source on a Debian-Based system, the following steps need to be made (tested on Debian Buster):

```
$ apt-get -y install make git
$ apt-get -y install python3-yaml python3-cheetah python3-netaddr python3-
↪simplejson
$ apt-get -y install python3-future python3-distro python3-setuptools python3-
↪sphinx python3-coverage
$ apt-get -y install pyflakes3 python3-pycodestyle
$ apt-get -y install apache2 libapache2-mod-wsgi-py3
$ apt-get -y install atftpd
# In case you want cobbler-web
$ apt-get -y install python3-django

$ a2enmod proxy
$ a2enmod proxy_http
$ a2enmod rewrite

$ ln -s /srv/tftp /var/lib/tftpboot

$ systemctl restart apache2
```

Change all `/var/www/cobbler` in `/etc/apache2/conf.d/cobbler.conf` to `/usr/share/cobbler/webroot/` Init script: - add Required-Stop line - path needs to be `/usr/local/...` or fix the install location

2.5 Source

The latest source code is available through git:

```
$ git clone https://github.com/cobbler/cobbler.git
$ cd cobbler
```

The release30 branch corresponds to the official release version for the 3.0.x series. The master branch is the development series, and always uses an odd number for the minor version (for example, 3.1.0).

When building from source, make sure you have the correct prerequisites. The Makefile uses a script called *distro_build_configs.sh* which sets the correct environment variables. Be sure to source it if you do not use the Makefile. If all prerequisites are met, you can install Cobbler with the following command:

```
$ make install
```

This command will rewrite all configuration files on your system if you have an existing installation of Cobbler (whether it was installed via packages or from an older source tree).

To preserve your existing configuration files, snippets and automatic installation files, run this command:

```
$ make devinstall
```

To install the Cobbler web GUI, use these steps:

1. Copy the systemd service file for *cobblerd* from `/etc/cobbler/cobblerd.service` to your systemd unit directory (`/etc/systemd/system`) and adjust `ExecStart` from `/usr/bin/cobblerd` to `/usr/local/bin/cobblerd`.
2. Install `apache2-mod_wsgi-python3` or the package responsible for your distro. (On Debian: `libapache2-mod-wsgi-py3`)
3. Enable the proxy module of Apache2 (`a2enmod proxy` or something similar) if not enabled.
4. `make webtest`
5. Copy `templates` and `cobbler.wsgi` from the `web` folder to `/usr/share/cobbler/web`.
6. Copy `settings.py` from `cobbler/web` to `/usr/share/cobbler/web` and adjust the `SECRET_KEY` there.
7. Restart Apache and `cobblerd`.

This will do a full install, not just the web GUI. `make webtest` is a wrapper around `make devinstall`, so your configuration files will also be saved when running this command. Be adviced that we don't copy the service file into the correct directory and that the path to the binary may be wrong depending on the location of the binary on your system. Do this manually and then you should be good to go. The same is valid for the Apache webserver config.

2.6 Relocating your installation

Often folks don't have a very large `/var` partition, which is what Cobbler uses by default for mirroring install trees and the like.

You'll notice you can reconfigure the webdir location just by going into `/etc/cobbler/settings.yaml`, but it's not the best way to do things – especially as the packaging process does include some files and directories in the stock path. This means that, for upgrades and the like, you'll be breaking things somewhat. Rather than attempting to reconfigure Cobbler, your Apache configuration, your file permissions, and your SELinux rules, the recommended course of action is very simple.

1. Copy everything you have already in `/var/www/cobbler` to another location – for instance, `/opt/cobbler_data`

2. Now just create a symlink or bind mount at `/var/www/cobbler` that points to `/opt/cobbler_data`.

Done. You're up and running.

If you decided to access Cobbler's data store over NFS (not recommended) you really want to mount NFS on `/var/www/cobbler` with SELinux context passed in as a parameter to mount versus the symlink. You may also have to deal with problems related to rootsquash. However if you are making a mirror of a Cobbler server for a multi-site setup, mounting read only is OK there.

Also Note: `/var/lib/cobbler` can not live on NFS, as this interferes with locking ("flock") Cobbler does around it's storage files.

CHAPTER 3

Cobbler CLI

This page contains a description for commands which can be used from the CLI.

3.1 General Principles

This should just be a brief overview. For the detailed explanations please refer to Readthedocs.

3.1.1 Distros, Profiles and Systems

Cobbler has a system of inheritance when it comes to managing the information you want to apply to a certain system.

3.1.2 Images

3.1.3 Repositories

3.1.4 Management Classes

3.1.5 Deleting configuration entries

If you want to remove a specific object, use the remove command with the name that was used to add it.

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file remove --
↪name=string
```

3.1.6 Editing

If you want to change a particular setting without doing an `add` again, use the `edit` command, using the same name you gave when you added the item. Anything supplied in the parameter list will overwrite the settings in the existing object, preserving settings not mentioned.

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file edit --name=string␣
↪[parameterlist]
```

3.1.7 Copying

Objects can also be copied:

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file copy --
↪name=oldname --newname=newname
```

3.1.8 Renaming

Objects can also be renamed, as long as other objects don't reference them.

```
cobbler distro|profile|system|repo|image|mgmtclass|package|file rename --
↪name=oldname --newname=newname
```

3.2 CLI-Commands

Short Usage: `cobbler command [subcommand] [--arg1=value1] [--arg2=value2]`

Long Usage:

```
cobbler <distro|profile|system|repo|image|mgmtclass|package|file> ...␣
↪[add|edit|copy|get-autoinstall*|list|remove|rename|report] [options|--help]
cobbler <aclsetup|buildiso|import|list|replicate|report|reposync|sync|validate-
↪autoinstalls|version|signature|get-loaders|hardlink> [options|--help]
```

3.2.1 Cobbler distro

This first step towards configuring what you want to install is to add a distribution record to Cobbler's configuration.

If there is an rsync mirror, DVD, NFS, or filesystem tree available that you would rather `import` instead, skip down to the documentation about the `import` command. It's really a lot easier to follow the import workflow – it only requires waiting for the mirror content to be copied and/or scanned. Imported mirrors also save time during install since they don't have to hit external install sources.

If you want to be explicit with distribution definition, however, here's how it works:

```
$ cobbler distro add --name=string --kernel=path --initrd=path [--kernel-
↪options=string] [--kernel-options-post=string] [--autoinstall-meta=string] [--
↪arch=i386|x86_64|ppc|ppc64] [--breed=redhat|debian|suse] [--template-
↪files=string]
```

Name	Description
arch	Sets the architecture for the PXE bootloader and also controls how Koan's `--replace-self` option will operate. The default setting (`standard`) will use `pxelinux`. Set to `ppc` and `ppc64` to use `yaboot`. `x86` and `x86_64` effectively do the same thing as standard. If you perform a `cobbler import`, the arch field will be auto-assigned.
autoinstall-meta	This is an advanced feature that sets automatic installation template variables to substitute, thus enabling those files to be treated as templates. Templates are powered using Cheetah and are described further along in this manpage as well as on the Cobbler Wiki. Example: `--autoinstall-meta="foo=bar baz=3 asdf"` See the section on "Kickstart Templating" for further information.
boot-files	TFTP Boot Files (Files copied into tftpboot beyond the kernel/initrd).
boot-loader	Boot loader (Network installation boot loader). Valid options are <<inherit>>, *grub*, *pxelinux*, *yaboot*, *ipxe*.
breed	Controls how various physical and virtual parameters, including kernel arguments for automatic installation, are to be treated. Defaults to `redhat`, which is a suitable value for Fedora and CentOS as well. It means anything Red Hat based. There is limited experimental support for specifying "debian", "ubuntu", or "suse", which treats the automatic installation template file as a preseed/autoyast file format and changes the kernel arguments appropriately. Support for other types of distributions is possible in the future. See the Wiki for the latest information about support for these distributions. The file used for the answer file, regardless of the breed setting, is the value used for `--autoinstall` when creating the profile.
comment	Simple attach a description (Free form text) to your distro.
fetchable-files	Fetchable Files (Templates for tftp or wget/curl)
initrd	An absolute filesystem path to a initrd image.
kernel	An absolute filesystem path to a kernel image.
kernel-options	Sets kernel command-line arguments that the distro, and profiles/systems depending on it, will use. To remove a kernel argument that may be added by a higher Cobbler object (or in the global settings), you can prefix it with a `!`. Example: `--kernel-options="foo=bar baz=3 asdf !gulp"` This example passes the arguments `foo=bar baz=3 asdf` but will make sure `gulp` is not passed even if it was requested at a level higher up in the Cobbler configuration.
kernel-options-post	This is just like `--kernel-options`, though it governs kernel options on the installed OS, as opposed to kernel options fed to the installer. The syntax is exactly the same. This requires some special snippets to be found in your automatic installation template in order for this to work. Automatic installation templating is described later on in this document. Example: `noapic`
mgmt-classes	Management Classes (Management classes for external config management).
name	A string identifying the distribution, this should be something like `rhel6`.
os-version	Generally this field can be ignored. It is intended to alter some hardware setup for virtualized instances when provisioning guests with Koan. The valid options for `--os-version` vary depending on what is specified for `--breed`. If you specify an invalid option, the error message will contain a list of valid OS versions that can be used. If you don't know the OS version or it does not appear in the list, omitting this argument or using `other` should be perfectly fine. If you don't encounter any problems with virtualized instances, this option can be safely ignored.
owners	Users with small sites and a limited number of admins can probably ignore this option. All Cobbler objects (distros, profiles, systems, and repos) can take a –owners parameter to specify what Cobbler users can edit particular objects.This only applies to the Cobbler WebUI and XML-RPC interface, not the "cobbler" command line tool run from the shell. Furthermore, this is only respected by the `authz_ownership` module which must be enabled in `/etc/cobbler/modules.conf`. The value for `--owners` is a space separated list of users and groups as specified in `/etc/cobbler/users.conf`. For more information see the users.conf file as well as the Cobbler Wiki. In the default Cobbler configuration, this value is completely ignored, as is `users.conf`.
redhat-management-key	Management Classes (Management classes for external config management).

3.2.2 Cobbler profile

A profile associates a distribution to additional specialized options, such as a installation automation file. Profiles are the core unit of provisioning and at least one profile must exist for every distribution to be provisioned. A profile might represent, for instance, a web server or desktop configuration. In this way, profiles define a role to be performed.

```
$ cobbler profile add --name=string --distro=string [--autoinstall=path] [--kernel-
↪options=string] [--autoinstall-meta=string] [--name-servers=string] [--name-
↪servers-search=string] [--virt-file-size=gigabytes] [--virt-ram=megabytes] [--
↪virt-type=string] [--virt-cpus=integer] [--virt-path=string] [--virt-
↪bridge=string] [--server] [--parent=profile] [--filename=string]
```

Arguments are the same as listed for distributions, save for the removal of “arch” and “breed”, and with the additions listed below:

Name	Description
autoinstall	Local filesystem path to a automatic installation file, the file must reside under `/var/lib/cobbler/templates`
autoinstall_meta	Automatic Installation Metadata (Ex: *dog=fang agent=86*).
bootfiles	TFTP Boot Files (Files copied into tftpboot beyond the kernel/initrd).
comment	Simple attach a description (Free form text) to your distro.
dhcp_tag	DHCP Tag (see description in system).
distro	The name of a previously defined Cobbler distribution. This value is required.
enable_gpxe	Enable gPXE? (Use gPXE instead of PXELINUX for advanced booting options)
enable_menu	Enable PXE Menu? (Show this profile in the PXE menu?)
fetchable_files	Fetchable Files (Templates for tftp or wget/curl)
filename	This parameter can be used to select the bootloader for network boot. If specified, this must be a path relative to the TFTP servers root directory. (e.g. grub/grubx64.efi) For most use cases the default bootloader is correct and this can be omitted
name	A descriptive name. This could be something like `rhel5webservers` or `f9desktops`.
name_servers	If your nameservers are not provided by DHCP, you can specify a space separated list of addresses here to configure each of the installed nodes to use them (provided the automatic installation files used are installed on a per-system basis). Users with DHCP setups should not need to use this option. This is available to set in profiles to avoid having to set it repeatedly for each system record.
name_servers_search	You can specify a space separated list of domain names to configure each of the installed nodes to use them as domain search path. This is available to set in profiles to avoid having to set it repeatedly for each system record.
next_server	To override the Next server.
owners	Users with small sites and a limited number of admins can probably ignore this option. All objects (distros, profiles, systems, and repos) can take a –owners parameter to specify what Cobbler users can edit particular objects.This only applies to the Cobbler WebUI and XML-RPC interface, not the "cobbler" command line tool run from the shell. Furthermore, this is only respected by the `authz_ownership` module which must be enabled in `/etc/cobbler/modules.conf`. The value for `--owners` is a space separated list of users and groups as specified in `/etc/cobbler/users.conf`. For more information see the users.conf file as well as the Cobbler Wiki. In the default Cobbler configuration, this value is completely ignored, as is `users.conf`.
parent	This is an advanced feature. Profiles may inherit from other profiles in lieu of specifying `--distro`. Inherited profiles will override any settings specified in their parent, with the exception of `--autoinstall-meta` (templating) and `--kernel-options` (kernel options), which will be blended together. Example: If profile A has `--kernel-options="x=7 y=2"`, B inherits from A, and B has `--kernel-options="x=9 z=2"`, the actual kernel options that will be used for B are `x=9 y=2 z=2`. Example: If profile B has `--virt-ram=256` and A has `--virt-ram=512`, profile B will use the value 256. Example: If profile A has a `--virt-file-size=5` and B does not specify a size, B will use the value from A.
proxy	Proxy URL.
redhat_management_key	Management Classes (Management classes for external config management).
repos	This is a space delimited list of all the repos (created with `cobbler repo add` and updated with `cobbler reposync`) that this profile can make use of during automated installation. For example, an example might be `--repos="fc6i386updates fc6i386extras"` if the profile wants to access these two mirrors that are already mirrored on the Cobbler server. Repo management is described in greater depth later in the manpage.
server	This parameter should be useful only in select circumstances. If machines are on a subnet that cannot

3.2.3 Cobbler system

System records map a piece of hardware (or a virtual machine) with the Cobbler profile to be assigned to run on it. This may be thought of as choosing a role for a specific system.

Note that if provisioning via Koan and PXE menus alone, it is not required to create system records in Cobbler, though they are useful when system specific customizations are required. One such customization would be defining the MAC address. If there is a specific role intended for a given machine, system records should be created for it.

System commands have a wider variety of control offered over network details. In order to use these to the fullest possible extent, the automatic installation template used by Cobbler must contain certain automatic installation snippets (sections of code specifically written for Cobbler to make these values become reality). Compare your automatic installation templates with the stock ones in /var/lib/cobbler/templates if you have upgraded, to make sure you can take advantage of all options to their fullest potential. If you are a new Cobbler user, base your automatic installation templates off of these templates.

Read more about networking setup at: https://cobbler.readthedocs.io/en/release28/4_advanced/advanced%20networking.html

Example:

```
$ cobbler system add --name=string --profile=string [--mac=macaddress] [--ip-
↪address=ipaddress] [--hostname=hostname] [--kernel-options=string] [--
↪autoinstall-meta=string] [--autoinstall=path] [--netboot-enabled=Y/N] [--
↪server=string] [--gateway=string] [--dns-name=string] [--static-routes=string] [-
↪-power-address=string] [--power-type=string] [--power-user=string] [--power-
↪pass=string] [--power-id=string]
```

Adds a Cobbler System to the configuration. Arguments are specified as per "profile add" with the following changes:

Name	Description
autoinstall	While it is recommended that the `--autoinstall` parameter is only used within for the "profile add" command, there are limited scenarios when an install base switching to Cobbler may have legacy automatic installation files created on aper-system basis (one automatic installation file for each system, nothing shared) and may not want to immediately make use of the Cobbler templating system. This allows specifying a automatic installation file for use on a per-system basis. Creation of a parent profile is still required. If the automatic installation file is a filesystem location, it will still be treated as a Cobbler template.
autoinstall-meta	Automatic Installation Metadata (Ex: *dog=fang agent=86*).
boot-files	TFTP Boot Files (Files copied into tftpboot beyond the kernel/initrd).
boot-loader	Boot loader (Network installation boot loader). Valid options are <<inherit>>, *grub*, *pxelinux*, *yaboot*, *ipxe*.
comment	Simple attach a description (Free form text) to your distro.

Continued on next page

Table 1 – continued from previous page

Name	Description
dhcp-tag	If you are setting up a PXE environment with multiple subnets/gateways, and are using Cobbler to manage a DHCP configuration, you will probably want to use this option. If not, it can be ignored. By default, the dhcp tag for all systems is "default" and means that in the DHCP template files the systems will expand out where $insert_cobbler_systems_definitions is found in the DHCP template. However, you may want certain systems to expand out in other places in the DHCP config file. Setting `--dhcp-tag=subnet2` for instance, will cause that system to expand out where $insert_cobbler_system_definitions_subnet2 is found, allowing you to insert directives to specify different subnets (or other parameters) before the DHCP configuration entries for those particular systems. This is described further on the Cobbler Wiki.
dns-name	If using the DNS management feature (see advanced section – Cobbler supports auto-setup of BIND and dnsmasq), use this to define a hostname for the system to receive from DNS. Example: `--dns-name=mycomputer.example.com` This is a per-interface parameter. If you have multiple interfaces, it may be different for each interface, for example, assume a DMZ / dual-homed setup.
enable-gpxe	Enable gPXE? (Use gPXE instead of PXELINUX for advanced booting options)
fetchable-files	Fetchable Files (Templates for tftp or wget/curl)
filename	This parameter can be used to select the bootloader for network boot. If specified, this must be a path relative to the TFTP servers root directory. (e.g. grub/grubx64.efi) For most use cases the default bootloader is correct and this can be omitted
gateway and netmask	If you are using static IP configurations and the interface is flagged `--static=1`, these will be applied. Netmask is a per-interface parameter. Because of the way gateway is stored on the installed OS, gateway is a global parameter. You may use `--static-routes` for per-interface customizations if required.
hostname	This field corresponds to the hostname set in a systems `/etc/sysconfig/network` file. This has no bearing on DNS, even when manage_dns is enabled. Use `--dns-name` instead for that feature. This parameter is assigned once per system, it is not a per-interface setting.

Continued on next page

Table 1 – continued from previous page

Name	Description
interface	By default flags like `--ip`, `--mac`, `--dhcp-tag`, `--dns-name`, `--netmask`, `--virt-bridge`, and `--static-routes` operate on the first network interface defined for a system (eth0). However, Cobbler supports an arbitrary number of interfaces. Using `--interface=eth1` for instance, will allow creating and editing of a second interface. Interface naming notes: Additional interfaces can be specified (for example: eth1, or any name you like, as long as it does not conflict with any reserved names such as kernel module names) for use with the edit command. Defining VLANs this way is also supported, of you want to add VLAN 5 on interface eth0, simply name your interface eth0.5. Example: cobbler system edit –name=foo –ip-address=192.168.1.50 –mac=AA:BB:CC:DD:EE:A0 cobbler system edit –name=foo –interface=eth0 –ip-address=10.1.1.51 –mac=AA:BB:CC:DD:EE:A1 cobbler system report foo Interfaces can be deleted using the –delete-interface option. Example: cobbler system edit –name=foo –interface=eth2 –delete-interface

Continued on next page

Table 1 – continued from previous page

Name	Description
interface-type, interface-master, bonding-opts, bridge-opts	One of the other advanced networking features supported by Cobbler is NIC bonding, bridging and BMC. You can use this to bond multiple physical network interfaces to one single logical interface to reduce single points of failure in your network, to create bridged interfaces for things like tunnels and virtual machine networks, or to manage BMC interface by DHCP. Supported values for the `--interface-type` parameter are "bond", "bond_slave", "bridge", "bridge_slave","bonded_bridge_slave" and "bmc". If one of the "_slave" options is specified, you also need to define the master-interface for this bond using `--interface-master=INTERFACE`. Bonding and bridge options for the master-interface may be specified using `--bonding-opts="foo=1 bar=2"` or `--bridge-opts="foo=1 bar=2"`. Example: **cobbler system edit –name=foo** –interface=eth0 –mac=AA:BB:CC:DD:EE:00 –interface-type=bond_slave –interface-master=bond0 **cobbler system edit –name=foo** –interface=eth1 –mac=AA:BB:CC:DD:EE:01 –interface-type=bond_slave –interface-master=bond0 **cobbler system edit –name=foo** –interface=bond0 –interface-type=bond –bonding-opts="mode=active-backup miimon=100" –ip-address=192.168.0.63 –netmask=255.255.255.0 –gateway=192.168.0.1 –static=1 More information about networking setup is available at https://github.com/cobbler/cobbler/wiki/Advanced-networking To review what networking configuration you have for any object, run "cobbler system report" at any time: Example: cobbler system report –name=foo
if-gateway	If you are using static IP configurations and have multiple interfaces, use this to define different gateway for each interface. This is a per-interface setting.

Continued on next page

Table 1 – continued from previous page

Name	Description
ip-address, ipv6-address	If Cobbler is configured to generate a DHCP configuration (see advanced section), use this setting to define a specific IP for this system in DHCP. Leaving off this parameter will result in no DHCP management for this particular system. Example: `--ip-address=192.168.1.50` If DHCP management is disabled and the interface is labelled `--static=1`, this setting will be used for static IP configuration. Special feature: To control the default PXE behavior for an entire subnet, this field can also be passed in using CIDR notation. If `--ip` is CIDR, do not specify any other arguments other than `--name` and `--profile`. When using the CIDR notation trick, don't specify any arguments other than `--name` and `--profile`, as they won't be used.
kernel-options	Sets kernel command-line arguments that the distro, and profiles/systems depending on it, will use. To remove a kernel argument that may be added by a higher Cobbler object (or in the global settings), you can prefix it with a `!`. Example: `--kernel-options="foo=bar baz=3 asdf !gulp"` This example passes the arguments `foo=bar baz=3 asdf` but will make sure `gulp` is not passed even if it was requested at a level higher up in the Cobbler configuration.
kernel-options-post	This is just like `--kernel-options`, though it governs kernel options on the installed OS, as opposed to kernel options fed to the installer. The syntax is exactly the same. This requires some special snippets to be found in your automatic installation template in order for this to work. Automatic installation templating is described later on in this document. Example: `noapic`
mac, mac-address	Specifying a mac address via `--mac` allows the system object to boot directly to a specific profile via PXE, bypassing Cobbler's PXE menu. If the name of the Cobbler system already looks like a mac address, this is inferred from the system name and does not need to be specified. MAC addresses have the format AA:BB:CC:DD:EE:FF. It's highly recommended to register your MAC addresses in Cobbler if you're using static addressing with multiple interfaces, or if you are using any of the advanced networking features like bonding, bridges or VLANs. Cobbler does contain a feature (enabled in /etc/cobbler/settings.yaml) that can automatically add new system records when it finds profiles being provisioned on hardware it has seen before. This may help if you do not have a report of all the MAC addresses in your datacenter/lab configuration.

Continued on next page

Table 1 – continued from previous page

Name	Description
mgmt-classes	Management Classes (Management classes for external config management).
mgmt-parameters	Management Parameters which will be handed to your management application. (Must be valid YAML dictionary)
name	The system name works like the name option for other commands. If the name looks like a MAC address or an IP, the name will implicitly be used for either `--mac` or `--ip` of the first interface, respectively. However, it's usually better to give a descriptive name – don't rely on this behavior. A system created with name "default" has special semantics. If a default system object exists, it sets all undefined systems to PXE to a specific profile. Without a "default" system name created, PXE will fall through to local boot for unconfigured systems. When using "default" name, don't specify any other arguments than `--profile`, as they won't be used.
name-servers	If your nameservers are not provided by DHCP, you can specify a space separated list of addresses here to configure each of the installed nodes to use them (provided the automatic installation files used are installed on a per-system basis). Users with DHCP setups should not need to use this option. This is available to set in profiles to avoid having to set it repeatedly for each system record.
name-servers-search	You can specify a space separated list of domain names to configure each of the installed nodes to use them as domain search path. This is available to set in profiles to avoid having to set it repeatedly for each system record.
netboot-enabled	If set false, the system will be provisionable through Koan but not through standard PXE. This will allow the system to fall back to default PXE boot behavior without deleting the Cobbler system object. The default value allows PXE. Cobbler contains a PXE boot loop prevention feature (pxe_just_once, can be enabled in /etc/cobbler/settings.yaml) that can automatically trip off this value after a system gets done installing. This can prevent installs from appearing in an endless loop when the system is set to PXE first in the BIOS order.
next-server	To override the Next server.

Continued on next page

Table 1 – continued from previous page

Name	Description
owners	Users with small sites and a limited number of admins can probably ignore this option. All objects (distros, profiles, systems, and repos) can take a –owners parameter to specify what Cobbler users can edit particular objects.This only applies to the Cobbler WebUI and XML-RPC interface, not the "cobbler" command line tool run from the shell. Furthermore, this is only respected by the `authz_ownership` module which must be enabled in `/etc/cobbler/modules.conf`. The value for `--owners` is a space separated list of users and groups as specified in `/etc/cobbler/users.conf`. For more information see the users.conf file as well as the Cobbler Wiki. In the default Cobbler configuration, this value is completely ignored, as is `users.conf`.
power-address, power-type, power-user, power-pass, power-id, power-options, power-identity-file	Cobbler contains features that enable integration with power management for easier installation, reinstallation, and management of machines in a datacenter environment. These parameters are described online at *power-management*. If you have a power-managed datacenter/lab setup, usage of these features may be something you are interested in.
profile	The name of Cobbler profile the system will inherite its properties.
proxy	Proxy URL.
redhat- management-key	Management Classes (Management classes for external config management).
repos-enabled	If set true, Koan can reconfigure repositories after installation. This is described further on the Cobbler Wiki,https://github.com/cobbler/cobbler/wiki/Manage-yum-repos.
static	Indicates that this interface is statically configured. Many fields (such as gateway/netmask) will not be used unless this field is enabled. This is a per-interface setting.
static-routes	This is a space delimited list of ip/mask:gateway routing information in that format. Most systems will not need this information. This is a per-interface setting.
virt-auto-boot	(Virt-only) Virt Auto Boot (Auto boot this VM?).
virt-bridge	(Virt-only) This specifies the default bridge to use for all systems defined under this profile. If not specified, it will assume the default value in the Cobbler settings file, which as shipped in the RPM is `xenbr0`. If using KVM, this is most likely not correct. You may want to override this setting in the system object. Bridge settings are important as they define how outside networking will reach the guest. For more information on bridge setup, see the Cobbler Wiki, where there is a section describing Koan usage.
virt-cpus	(Virt-only) How many virtual CPUs should Koan give the virtual machine? The default is 1. This is an integer.

Continued on next page

Table 1 – continued from previous page

Name	Description
virt-disk-driver	(Virt-only) Virt Disk Driver Type (The on-disk format for the virtualization disk). Valid options are <<inherit>>, *raw*, *qcow2*, *qed*, *vdi*, *vmdk*
virt-file-size	(Virt-only) How large the disk image should be in Gigabytes. The default is 5. This can be a comma separated list (ex: `5,6,7`) to allow for multiple disks of different sizes depending on what is given to `--virt-path`. This should be input as a integer or decimal value without units.
virt-path	(Virt-only) Where to store the virtual image on the host system. Except for advanced cases, this parameter can usually be omitted. For disk images, the value is usually an absolute path to an existing directory with an optional filename component. There is support for specifying partitions `/dev/sda4` or volume groups `VolGroup00`, etc. For multiple disks, separate the values with commas such as `VolGroup00,VolGroup00` or `/dev/sda4,/dev/sda5`. Both those examples would create two disks for the VM.
virt-ram	(Virt-only) How many megabytes of RAM to consume. The default is 512 MB. This should be input as an integer without units.
virt-type	(Virt-only) Koan can install images using either Xen paravirt (`xenpv`) or QEMU/KVM (`qemu`). Choose one or the other strings to specify, or values will default to attempting to find a compatible installation type on the client system("auto"). See the "Koan" manpage for more documentation. The default `--virt-type` can be configured in the Cobbler settings file such that this parameter does not have to be provided. Other virtualization types are supported, for information on those options (such as VMware), see the Cobbler Wiki.

3.2.4 Cobbler repo

Repository mirroring allows Cobbler to mirror not only install trees ("cobbler import" does this for you) but also optional packages, 3rd party content, and even updates. Mirroring all of this content locally on your network will result in faster, more up-to-date installations and faster updates. If you are only provisioning a home setup, this will probably be overkill, though it can be very useful for larger setups (labs, datacenters, etc).

```
$ cobbler repo add --mirror=url --name=string [--rpmlist=list] [--creatrepo-
↪flags=string] [--keep-updated=Y/N] [--priority=number] [--arch=string] [--mirror-
↪locally=Y/N] [--breed=yum|rsync|rhn] [--mirror_type=baseurl|mirrorlist|metalink]
```

Name	Description
apt-components	Apt Components (apt only) (ex: main restricted universe)
apt-dists	Apt Dist Names (apt only) (ex: precise precise-updates)
arch	Specifies what architecture the repository should use. By default the current system arch (of the server) is used,which may not be desirable. Using this to override the default arch allows mirroring of source repositories(using `--arch=src`).
breed	Ordinarily Cobbler's repo system will understand what you mean without supplying this parameter, though you can set it explicitly if needed.
comment	Simple attach a description (Free form text) to your distro.
createrepo-flags	Specifies optional flags to feed into the createrepo tool, which is called when `cobbler reposync` is run for the given repository. The defaults are `-c cache`.
keep-updated	Specifies that the named repository should not be updated during a normal "cobbler reposync". The repo may still be updated by name. The repo should be synced at least once before disabling this feature. See "cobbler reposync" below.
mirror	The address of the yum mirror. This can be an `rsync://`-URL, an ssh location, or a `http://` or `ftp://` mirror location. Filesystem paths also work. The mirror address should specify an exact repository to mirror – just one architecture and just one distribution. If you have a separate repo to mirror for a different arch, add that repo separately. Here's an example of what looks like a good URL: • `rsync://yourmirror.example.com/fedora-linux-core/updates/6/i386` (for rsync protocol) • `http://mirrors.kernel.org/fedora/extras/6/i386/` (for http) • `user@yourmirror.example.com/fedora-linux-core/updates/6/i386` (for SSH) Experimental support is also provided for mirroring RHN content when you need a fast local mirror. The mirror syntax for this is `--mirror=rhn://channel-name` and you must have entitlements for this to work. This requires the Cobbler server to be installed on RHEL 5 or later. You will also need a version of `yum-utils` equal or greater to 1.0.4.
mirror-locally	When set to `N`, specifies that this yum repo is to be referenced directly via automatic installation files and not mirrored locally on the Cobbler server. Only `http://` and `ftp://` mirror urls are supported when using `--mirror-locally=N`, you cannot use filesystem URLs.
name	This name is used as the save location for the mirror. If the mirror represented, say, Fedora Core 6 i386 updates, a good name would be `fc6i386updates`. Again, be specific. This name corresponds with values given to the `--repos` parameter of `cobbler profile add`. If a profile has a `--repos`-value that matches the name given here, that repo can be automatically set up during provisioning (when supported) and installed systems will also use the boot server as a mirror (unless `yum_post_install_mirror` is disabled in the settings file). By default the provisioning server will act as a mirror to systems it installs, which may not be desirable for laptop configurations, etc. Distros that can make use of yum repositories during automatic installation include FC6 and later, RHEL 5 and later, and derivative distributions. See the documentation on `cobbler profile add` for more information.
owners | Users with small sites and a limited number of admins can probably ignore this option. All objects (distros, profiles, systems, and repos) can take a –owners parameter to specify what Cobbler users can edit particular objects.This only applies to the Cobbler WebUI and XML-RPC interface, not the "cobbler" command line tool run from the shell. Furthermore, this is only respected by the `authz_ownership` module which must be enabled in `/etc/cobbler/modules.conf`. The value for `--owners` is a space separated list of users and groups as specified in `/etc/cobbler/users.conf`.	

```
$ cobbler repo autoadd
```

Add enabled yum repositories from `dnf repolist --enabled` list. The repository names are generated using the <repo id>-<releasever>-<arch> pattern (ex: fedora-32-x86_64). Existing repositories with such names are not overwritten.

3.2.5 Cobbler image

Example:

```
$ cobbler image
```

3.2.6 cobbler mgmtclass

Management classes allows Cobbler to function as an configuration management system. Cobbler currently supports the following resource types:

1. Packages
2. Files

Resources are executed in the order listed above.

```
$ cobbler mgmtclass add --name=string --comment=string [--packages=list] [--
↪files=list]
```

Name	Description
classname	Class Name (Actual Class Name (leave blank to use the name field)).
comment	A comment that describes the functions of the management class.
files	Specifies a list of file resources required by the management class.
name	The name of the mgmtclass. Use this name when adding a management class to a system, profile, or distro. To add a mgmtclass to an existing system use something like (`cobbler system edit --name="madhatter" --mgmt-classes="http mysql"`).
packages	Specifies a list of package resources required by the management class.

3.2.7 Cobbler package

Package resources are managed using `cobbler package add`

Actions:

Name	Description
install	Install the package. [Default]
uninstall	Uninstall the package.

Attributes:

Name	Description
installer	Which package manager to use, valid options [rpm|yum].
name	Cobbler object name.
version	Which version of the package to install.

Example:

```
$ cobbler package add --name=string --comment=string [--action=install|uninstall] --
↪installer=string [--version=string]
```

3.2.8 Cobbler file

Actions:

Name	Description
create	Create the file. [Default]
remove	Remove the file.

Attributes:

Name	Description
group	The group owner of the file.
mode	Permission mode (as in chmod).
name	Name of the cobbler file object
path	The path for the file.
template	The template for the file.
user	The user for the file.

Example:

```
$ cobbler file add --name=string --comment=string [--action=string] --mode=string --
↪group=string --owner=string --path=string [--template=string]
```

3.2.9 cobbler aclsetup

Example:

```
$ cobbler aclsetup
```

3.2.10 Cobbler buildiso

Example:

```
$ cobbler buildiso
```

3.2.11 Cobbler import

Note: When running Cobbler via systemd, you cannot mount the ISO to `/tmp` or a sub-folder of it because we are using the option *Private Temporary Directory*, to enhance the security of our application.

Example:

```
$ cobbler import
```

3.2.12 Cobbler list

This list all the names grouped by type. Identically to `cobbler report` there are subcommands for most of the other Cobbler commands. (Currently: distro, profile, system, repo, image, mgmtclass, package, file)

```
$ cobbler list
```

3.2.13 Cobbler replicate

Cobbler can replicate configurations from a master Cobbler server. Each Cobbler server is still expected to have a locally relevant `/etc/cobbler/cobbler.conf` and `modules.conf`, as these files are not synced.

This feature is intended for load-balancing, disaster-recovery, backup, or multiple geography support.

Cobbler can replicate data from a central server.

Objects that need to be replicated should be specified with a pattern, such as `--profiles="webservers* dbservers*"` or `--systems="*.example.org"`. All objects matched by the pattern, and all dependencies of those objects matched by the pattern (recursively) will be transferred from the remote server to the central server. This is to say if you intend to transfer `*.example.org` and the definition of the systems have not changed, but a profile above them has changed, the changes to that profile will also be transferred.

In the case where objects are more recent on the local server, those changes will not be overridden locally.

Common data locations will be rsync'ed from the master server unless `--omit-data` is specified.

To delete objects that are no longer present on the master server, use `--prune`.

Warning: This will delete all object types not present on the remote server from the local server, and is recursive. If you use prune, it is best to manage Cobbler centrally and not expect changes made on the slave servers to be preserved. It is not currently possible to just prune objects of a specific type.

Example:

```
$ cobbler replicate --master=cobbler.example.org [--distros=pattern] [--
↪profiles=pattern] [--systems=pattern] [--repos-pattern] [--images=pattern] [--
↪prune] [--omit-data]
```

3.2.14 Cobbler report

This lists all configuration which Cobbler can obtain from the saved data. There are also `report` subcommands for most of the other Cobbler commands (currently: distro, profile, system, repo, image, mgmtclass, package, file).

```
$ cobbler report --name=[object-name]
```

–name=[object-name]

Optional parameter which filters for object with the given name.

3.2.15 Cobbler reposync

Example:

```
$ cobbler reposync [--only=ONLY] [--tries=TRIES] [--no-fail]
```

Cobbler reposync is the command to use to update repos as configured with `cobbler repo add`. Mirroring can take a long time, and usage of cobbler reposync prior to usage is needed to ensure provisioned systems have the files they need to actually use the mirrored repositories. If you just add repos and never run `cobbler`

`reposync`, the repos will never be mirrored. This is probably a command you would want to put on a crontab, though the frequency of that crontab and where the output goes is left up to the systems administrator.

For those familiar with dnf's reposync, cobbler's reposync is (in most uses) a wrapper around the `dnf reposync` command. Please use `cobbler reposync` to update cobbler mirrors, as dnf's reposync does not perform all required steps. Also cobbler adds support for rsync and SSH locations, where as dnf's reposync only supports what dnf supports (http/ftp).

If you ever want to update a certain repository you can run: `cobbler reposync --only="reponame1" ...`

When updating repos by name, a repo will be updated even if it is set to be not updated during a regular reposync operation (ex: `cobbler repo edit -name=reponame1 -keep-updated=0`). Note that if a cobbler import provides enough information to use the boot server as a yum mirror for core packages, cobbler can set up automatic installation files to use the cobbler server as a mirror instead of the outside world. If this feature is desirable, it can be turned on by `setting yum_post_install_mirror` to `True` in `/etc/cobbler/settings.yaml` (and running `cobbler sync`). You should not use this feature if machines are provisioned on a different VLAN/network than production, or if you are provisioning laptops that will want to acquire updates on multiple networks.

The flags `--tries=N` (for example, `--tries=3`) and `--no-fail` should likely be used when putting reposync on a crontab. They ensure network glitches in one repo can be retried and also that a failure to synchronize one repo does not stop other repositories from being synchronized.

3.2.16 Cobbler sync

The sync command is very important, though very often unnecessary for most situations. It's primary purpose is to force a rewrite of all configuration files, distribution files in the TFTP root, and to restart managed services. So why is it unnecessary? Because in most common situations (after an object is edited, for example), Cobbler executes what is known as a "lite sync" which rewrites most critical files.

When is a full sync required? When you are using `manage_dhcpd` (Managing DHCP) with systems that use static leases. In that case, a full sync is required to rewrite the `dhcpd.conf` file and to restart the dhcpd service.

Cobbler sync is used to repair or rebuild the contents `/tftpboot` or `/var/www/cobbler` when something has changed behind the scenes. It brings the filesystem up to date with the configuration as understood by Cobbler.

Sync should be run whenever files in `/var/lib/cobbler` are manually edited (which is not recommended except for the settings file) or when making changes to automatic installation files. In practice, this should not happen often, though running sync too many times does not cause any adverse effects.

If using Cobbler to manage a DHCP and/or DNS server (see the advanced section of this manpage), sync does need to be run after systems are added to regenerate and reload the DHCP/DNS configurations.

The sync process can also be kicked off from the web interface.

Example:

```
$ cobbler sync
```

3.2.17 Cobbler validate-autoinstalls

Example:

```
$ cobbler validate-autoinstalls
```

3.2.18 Cobbler version

Example:

```
$ cobbler version
```

3.2.19 Cobbler signature

Example:

```
$ cobbler signature
```

3.2.20 Cobbler get-loaders

Example:

```
$ cobbler get-loaders
```

3.2.21 Cobbler hardlink

Example:

```
$ cobbler hardlink
```

3.3 EXIT_STATUS

Cobbler's command line returns a zero for success and non-zero for failure.

3.4 Additional Help

We have a Gitter Channel and you also can ask questions as GitHub issues. The IRC Channel on Freenode (#cobbler) is not that active but sometimes there are people who can help you.

The way we would prefer are GitHub issues as they are easily searchable.

CHAPTER 4

Cobblerd

Cobbler - a provisioning and update server

4.1 Preamble

We will refer to *cobblerd* here as "cobbler" because *cobblerd* is short for cobbler-daemon which is basically the server. The CLI will be referred to as Cobbler-CLI and Koan as Koan.

4.2 Description

Cobbler manages provisioning using a tiered concept of Distributions, Profiles, Systems, and (optionally) Images and Repositories.

Distributions contain information about what kernel and initrd are used, plus metadata (required kernel parameters, etc).

Profiles associate a Distribution with an automated installation template file and optionally customize the metadata further.

Systems associate a MAC, IP, and other networking details with a profile and optionally customize the metadata further.

Repositories contain yum mirror information. Using cobbler to mirror repositories is an optional feature, though provisioning and package management share a lot in common.

Images are a catch-all concept for things that do not play nicely in the "distribution" category. Most users will not need these records initially and these are described later in the document.

The main advantage of cobbler is that it glues together many disjoint technologies and concepts and abstracts the user from the need to understand them. It allows the systems administrator to concentrate on what he needs to do, and not how it is done.

This manpage will focus on the cobbler command line tool for use in configuring cobbler. There is also mention of the Cobbler WebUI which is usable for day-to-day operation of Cobbler once installed/configured. Docs on the API and XML-RPC components are available online at https://cobbler.github.io or https://cobbler.readthedocs.io.

Most users will be interested in the Web UI and should set it up, though the command line is needed for initial configuration – in particular `cobbler check` and `cobbler import`, as well as the repo mirroring features. All of these are described later in the documentation.

4.3 Setup

After installing, run `cobbler check` to verify that cobbler's ecosystem is configured correctly. Cobbler check will direct you on how to modify it's config files using a text editor.

Any problems detected should be corrected, with the potential exception of DHCP related warnings where you will need to use your judgement as to whether they apply to your environment. Run `cobbler sync` after making any changes to the configuration files to ensure those changes are applied to the environment.

It is especially important that the server name field be accurate in `/etc/cobbler/settings.yaml`, without this field being correct, automatic installation trees will not be found, and automated installations will fail.

For PXE, if DHCP is to be run from the cobbler server, the DHCP configuration file should be changed as suggested by `cobbler check`. If DHCP is not run locally, the `next-server` field on the DHCP server should at minimum point to the cobbler server's IP and the filename should be set to `pxelinux.0`. Alternatively, cobbler can also generate your DHCP configuration file if you want to run DHCP locally – this is covered in a later section. If you don't already have a DHCP setup managed by some other tool, allowing cobbler to manage your DHCP environment will prove to be useful as it can manage DHCP reservations and other data. If you already have a DHCP setup, moving an existing setup to be managed from within cobbler is relatively painless – though usage of the DHCP management feature is entirely optional. If you are not interested in network booting via PXE and just want to use Koan to install virtual systems or replace existing ones, DHCP configuration can be totally ignored. Koan also has a live CD (see Koan's manpage) capability that can be used to simulate PXE environments.

4.4 Autoinstallation (Autoyast/Kickstart)

For help in building kickstarts, try using the `system-config-kickstart` tool, or install a new system and look at the `/root/anaconda-ks.cfg` file left over from the installer. General kickstart questions can also be asked at kickstart-list@redhat.com. Cobbler ships some autoinstall templates in /etc/cobbler that may also be helpful.

For AutoYaST guides and help please refer to the opensuse project.

Also see the website or documentation for additional documentation, user contributed tips, and so on.

4.5 Options

-B –daemonize If you pass no options this is the default one. The Cobbler-Server runs in the background.

-F –no-daemonize The Cobbler-Server runs in the foreground.

-f –log-file Choose a destination for the logfile (currently has no effect).

-l –log-level Choose a loglevel for the application (currently has no effect).

CHAPTER 5

Cobbler Configuration

There are two main settings files which are located per default at `/etc/cobbler/`:

- The file `settings.yaml` is following YAML specification.
- The file `modules.conf` is following INI specification.

Note: Since we are cleaning a lot of tech-debt this may change over time. We are trying to find the balance which format is the best for us to handle in the code and the best for admins to handle in the config files.

Warning: If you are using `allow_dynamic_settings`, then the comments in the YAML file will vanish after the first change due to the fact that PyYAML doesn't support comments (Source)

There are additional configuration file locations which need to follow the YAML Syntax. These are loaded from the `include` directory in the `settings.yaml` file. Any key specified in one of these files overwrites values from the main file.

Warning: When using `allow_dynamic_settings` the values are only persisted in the file `settings.yaml`. This may lead to a non expected behaviour after `cobblerd` restarts. This is a known issue.

5.1 Updates to the yaml-settings-file

Starting with 3.2.1:

- We require the extension `.yaml` on our settings file to indicate the format of the file to editors and comply to standards of the YAML specification.
- We require the usage of booleans in the format of `True` and `False`. If you have old integer style booleans with `1` and `0` this is fine but you may should convert them as soon as possible. We may decide in a future version to enforce our new way in a stricter manner. Automatic conversion is only done on a best-effort/available-resources basis.
- We enforce the types of values to the keys. Additional unexpected keys will throw errors. If you have those used in Cobbler please report this in our issue tracker. We have decided to go this way to be able to rely

on the existence of the values. This gives us the freedom to write less access checks to the settings without loosing stability.

5.2 settings.yaml

5.2.1 allow_duplicate_hostnames

If `True`, Cobbler will allow insertions of system records that duplicate the `--dns-name` information of other system records. In general, this is undesirable and should be left False.

default: `False`

5.2.2 allow_duplicate_ips

If `True`, Cobbler will allow insertions of system records that duplicate the IP address information of other system records. In general, this is undesirable and should be left False.

default: `False`

5.2.3 allow_duplicate_macs

If `True`, Cobbler will allow insertions of system records that duplicate the mac address information of other system records. In general, this is undesirable.

default: `False`

5.2.4 allow_dynamic_settings

If `True`, Cobbler will allow settings to be changed dynamically without a restart of the `cobblerd` daemon. You can only change this variable by manually editing the settings file, and you MUST restart `cobblerd` after changing it.

default: `False`

5.2.5 always_write_dhcp_entries

Always write DHCP entries, regardless if netboot is enabled.

default: `False`

5.2.6 anamon_enabled

By default, installs are *not* set to send installation logs to the Cobbler server. With `anamon_enabled`, automatic installation templates may use the `pre_anamon` snippet to allow remote live monitoring of their installations from the Cobbler server. Installation logs will be stored under `/var/log/cobbler/anamon/`.

Note: This does allow an XML-RPC call to send logs to this directory, without authentication, so enable only if you are ok with this limitation.

---

default: `False`

5.2.7 auth_token_expiration

How long the authentication token is valid for, in seconds.

default: `3600`

5.2.8 authn_pam_service

If using authn_pam in the `modules.conf`, this can be configured to change the PAM service authentication will be tested against.

default: `"login"`

5.2.9 autoinstall_snippets_dir

This is a directory of files that Cobbler uses to make templating easier. See the Wiki for more information. Changing this directory should not be required.

default: `/var/lib/cobbler/snippets`

5.2.10 autoinstall_templates_dir

This is a directory of files that Cobbler uses to make templating easier. See the Wiki for more information. Changing this directory should not be required.

default: `/var/lib/cobbler/templates`

5.2.11 bind_chroot_path

Set to path of bind chroot to create bind-chroot compatible bind configuration files. This should be automatically detected.

default: `""`

5.2.12 bind_master

Set to the ip address of the master bind DNS server for creating secondary bind configuration files.

default: `127.0.0.1`

5.2.13 boot_loader_conf_template_dir

Location of templates used for boot loader config generation.

default: `"/etc/cobbler/boot_loader_conf"`

5.2.14 bootloaders_dir

The location where Cobbler searches for the bootloaders to copy into the web directory.

default: `/var/lib/cobbler/loaders`

5.2.15 grubconfig_dir

The location where Cobbler searches for GRUB configuration files.

default: `/var/lib/cobbler/grub_config`

5.2.16 build_reporting_*

Email out a report when Cobbler finishes installing a system.

- enabled: Set to `true` to turn this feature on
- email: Which addresses to email
- ignorelist: TODO
- sender: Optional
- smtp_server: Used to specify another server for an MTA.
- subject: Use the default subject unless overridden.

defaults:

```
build_reporting_enabled: false
build_reporting_sender: ""
build_reporting_email: [ 'root@localhost' ]
build_reporting_smtp_server: "localhost"
build_reporting_subject: ""
build_reporting_ignorelist: [ "" ]
```

5.2.17 buildisodir

Used for caching the intermediate files for ISO-Building. You may want to use a SSD, a tmpfs or something which does not persist across reboots and can be easily thrown away but is also fast.

default: `/var/cache/cobbler/buildiso`

5.2.18 cache_enabled

If `cache_enabled` is `True`, a cache will keep converted records in memory to make checking them faster. This helps with use cases like writing out large numbers of records. There is a known issue with cache and remote XML-RPC API calls. If you will use Cobbler with config management or infrastructure-as-code tools such as Terraform, it is recommended to disable by setting to `False`.

default: `True`

5.2.19 cheetah_import_whitelist

Cheetah-language autoinstall templates can import Python modules. while this is a useful feature, it is not safe to allow them to import anything they want. This whitelists which modules can be imported through Cheetah. Users can expand this as needed but should never allow modules such as subprocess or those that allow access to the filesystem as Cheetah templates are evaluated by `cobblerd` as code.

default:

- `random`
- `re`
- `time`
- `netaddr`

5.2.20 client_use_https

If set to `True`, all commands to the API (not directly to the XML-RPC server) will go over HTTPS instead of plain text. Be sure to change the `http_port` setting to the correct value for the web server.

default: `False`

5.2.21 client_use_localhost

If set to `True`, all commands will be forced to use the localhost address instead of using the above value which can force commands like `cobbler sync` to open a connection to a remote address if one is in the configuration and would traceback.

default: `False`

5.2.22 cobbler_master

Used for replicating the Cobbler instance.

default: `""`

5.2.23 convert_server_to_ip

Convert hostnames to IP addresses (where possible) so DNS isn't a requirement for various tasks to work correctly.

default: `False`

5.2.24 createrepo_flags

Default `createrepo_flags` to use for new repositories.

default: `"-c cache -s sha"`

5.2.25 default_autoinstall

If no autoinstall template is specified to profile add, use this template.

default: `/var/lib/cobbler/autoinstall_templates/default.ks`

5.2.26 default_name_*

Configure all installed systems to use these name servers by default unless defined differently in the profile. For DHCP configurations you probably do **not** want to supply this.

defaults:

```
default_name_servers: []
default_name_servers_search: []
```

5.2.27 default_ownership

if using the `authz_ownership` module, objects created without specifying an owner are assigned to this owner and/or group.

default:

- `admin`

5.2.28 default_password_crypted

Cobbler has various sample automatic installation templates stored in `/var/lib/cobbler/autoinstall_templates/`. This controls what install (root) password is set up for those systems that reference this variable. The factory default is "cobbler" and Cobbler check will warn if this is not changed. The simplest way to change the password is to run `openssl passwd -1` and put the output between the `""`.

default: `"$1$mF86/UHC$WvcIcX2t6crBz2onWxyac."`

5.2.29 default_template_type

The default template type to use in the absence of any other detected template. If you do not specify the template with `#template=<template_type>` on the first line of your templates/snippets, Cobbler will assume try to use the following template engine to parse the templates.

Note: Over time we will try to deprecate and remove Cheetah3 as a template engine. It is hard to package and there are fewer guides then with Jinja2. Making the templating independent of the engine is a task which complicates the code. Thus, please try to use Jinja2. We will try to support a seamless transition on a best-effort basis.

Current valid values are: `cheetah`, `jinja2`

default: `"cheetah"`

5.2.30 default_virt_bridge

For libvirt based installs in Koan, if no virt-bridge is specified, which bridge do we try? For EL 4/5 hosts this should be `xenbr0`, for all versions of Fedora, try `virbr0`. This can be overridden on a per-profile basis or at the Koan command line though this saves typing to just set it here to the most common option.

default: `xenbr0`

5.2.31 default_virt_disk_driver

The on-disk format for the virtualization disk.

default: `raw`

5.2.32 default_virt_file_size

Use this as the default disk size for virt guests (GB).

default: `5`

5.2.33 default_virt_ram

Use this as the default memory size for virt guests (MB).

default: `512`

5.2.34 default_virt_type

If Koan is invoked without `--virt-type` and no virt-type is set on the profile/system, what virtualization type should be assumed?

Current valid values are:

- `xenpv`
- `xenfv`
- `qemu`
- `vmware`

NOTE: this does not change what `virt_type` is chosen by import.

default: `xenpv`

5.2.35 enable_gpxe

Enable gPXE booting? Enabling this option will cause Cobbler to copy the `undionly.kpxe` file to the TFTP root directory, and if a profile/system is configured to boot via gPXE it will chain load off `pxelinux.0`.

Note: We now gPXE is not active anymore and try to transition the code, settings and guide we have to iPXE.

---

default: `False`

5.2.36 enable_menu

Controls whether Cobbler will add each new profile entry to the default PXE boot menu. This can be overridden on a per-profile basis when adding/editing profiles with `--enable-menu=False/True`. Users should ordinarily leave this setting enabled unless they are concerned with accidental reinstall from users who select an entry at the PXE boot menu. Adding a password to the boot menus templates may also be a good solution to prevent unwanted reinstallations.

default: `True`

5.2.37 http_port

Change this port if Apache is not running plain text on port 80. Most people can leave this alone.

default: `80`

5.2.38 include

Include other configuration snippets with this regular expression. This is a list of folders.

default: `[ "/etc/cobbler/settings.d/*.settings" ]`

5.2.39 iso_template_dir

Folder to search for the ISO templates. These will build the boot-menu of the built ISO.

default: `/etc/cobbler/iso`

5.2.40 jinja2_includedir

This is a directory of files that Cobbler uses to include files into Jinja2 templates. Per default this settings is commented out.

default: `/var/lib/cobbler/jinja2`

5.2.41 kernel_options

Kernel options that should be present in every Cobbler installation. Kernel options can also be applied at the distro/profile/system level.

default: `{}`

5.2.42 ldap_*

Configuration options if using the authn_ldap module. See the Wiki for details. This can be ignored if you are not using LDAP for WebUI/XML-RPC authentication.

defaults:

```
ldap_server: "ldap.example.com"
ldap_base_dn: "DC=example,DC=com"
ldap_port: 389
ldap_tls: true
ldap_anonymous_bind: true
ldap_search_bind_dn: ''
ldap_search_passwd: ''
ldap_search_prefix: 'uid='
ldap_tls_cacertfile: ''
ldap_tls_keyfile: ''
ldap_tls_certfile: ''
```

5.2.43 bind_manage_ipmi

When using the Bind9 DNS server, you can enable or disable if the BMCs should receive own DNS entries.

default: `False`

5.2.44 manage_dhcp

Set to `True` to enable Cobbler's DHCP management features. The choice of DHCP management engine is in `/etc/cobbler/modules.conf`.

default: `True`

5.2.45 manage_dns

Set to `True` to enable Cobbler's DNS management features. The choice of DNS management engine is in `/etc/cobbler/modules.conf`.

default: `False`

5.2.46 manage_*_zones

If using BIND (named) for DNS management in `/etc/cobbler/modules.conf` and `manage_dns` is enabled (above), this lists which zones are managed. See *DNS configuration management* for more information.

defaults:

```
manage_forward_zones: []
manage_reverse_zones: []
```

5.2.47 manage_genders

Whether or not to manage the genders file. For more information on that visit: github.com/chaos/genders

default: `False`

5.2.48 manage_rsync

Set to `True` to enable Cobbler's RSYNC management features.

default: `False`

5.2.49 manage_tftpd

Set to `True` to enable Cobbler's TFTP management features. The choice of TFTP management engine is in `/etc/cobbler/modules.conf`.

default: `True`

5.2.50 mgmt_*

Cobbler has a feature that allows for integration with config management systems such as Puppet. The following parameters work in conjunction with `--mgmt-classes` and are described in further detail at *Configuration Management Integrations*.

```
mgmt_classes: []
mgmt_parameters:
    from_cobbler: true
```

5.2.51 next_server

If using Cobbler with `manage_dhcp`, put the IP address of the Cobbler server here so that PXE booting guests can find it. If you do not set this correctly, this will be manifested in TFTP open timeouts.

default: `127.0.0.1`

5.2.52 nsupdate_enabled

This enables or disables the replacement (or removal) of records in the DNS zone for systems created (or removed) by Cobbler.

Note: There are additional settings needed when enabling this. Due to the limited number of resources, this won't be done until 3.3.0. Thus please expect to run into troubles when enabling this setting.

default: `False`

5.2.53 nsupdate_log

The logfile to document what records are added or removed in the DNS zone for systems.

Note: The functionality this settings is related to is currently not tested due to tech-debt. Please use it with caution. This note will be removed once we were able to look deeper into this functionality of Cobbler.

- Required: No
- Default: `/var/log/cobbler/nsupdate.log`

5.2.54 nsupdate_tsig_algorithm

Note: The functionality this settings is related to is currently not tested due to tech-debt. Please use it with caution. This note will be removed once we were able to look deeper into this functionality of Cobbler.

- Required: No
- Default: `hmac-sha512`

5.2.55 nsupdate_tsig_key

Note: The functionality this settings is related to is currently not tested due to tech-debt. Please use it with caution. This note will be removed once we were able to look deeper into this functionality of Cobbler.

- Required: No
- Default: `[]`

5.2.56 power_management_default_type

Settings for power management features. These settings are optional. See *Power Management* to learn more.

Choices (refer to the fence-agents project for a complete list):

- apc_snmp
- bladecenter
- bullpap
- drac
- ether_wake
- ilo
- integrity
- ipmilan
- ipmilanplus
- lpar
- rsa
- virsh
- wti

default: `ipmilanplus`

5.2.57 proxy_url_ext

External proxy which is used by the following commands: `get-loaders`, `reposync`, `signature update`

defaults:

```
http: http://192.168.1.1:8080
https: https://192.168.1.1:8443
```

5.2.58 proxy_url_int

Internal proxy which is used by systems to reach Cobbler for kickstarts.

e.g.: `proxy_url_int:  http://10.0.0.1:8080`

default: `""`

5.2.59 puppet_auto_setup

If enabled, this setting ensures that puppet is installed during machine provision, a client certificate is generated and a certificate signing request is made with the puppet master server.

default: `False`

5.2.60 puppet_parameterized_classes

Choose whether to enable puppet parameterized classes or not. Puppet versions prior to 2.6.5 do not support parameters.

default: `True`

5.2.61 puppet_server

Choose a `--server` argument when running puppetd/puppet agent during autoinstall.

default: `'puppet'`

5.2.62 puppet_version

Let Cobbler know that you're using a newer version of puppet. Choose version 3 to use: 'puppet agent'; version 2 uses status quo: 'puppetd'.

default: `2`

5.2.63 puppetca_path

Location of the puppet executable, used for revoking certificates.

default: `"/usr/bin/puppet"`

5.2.64 pxe_just_once

If this setting is set to `True`, Cobbler systems that pxe boot will request at the end of their installation to toggle the `--netboot-enabled` record in the Cobbler system record. This eliminates the potential for a PXE boot loop if the system is set to PXE first in it's BIOS order. Enable this if PXE is first in your BIOS boot order, otherwise leave this disabled. See the manpage for `--netboot-enabled`.

default: `True`

5.2.65 nopxe_with_triggers

If this setting is set to `True`, triggers will be executed when systems will request to toggle the `--netboot-enabled` record at the end of their installation.

default: `True`

5.2.66 redhat_management_permissive

If using `authn_spacewalk` in `modules.conf` to let Cobbler authenticate against Satellite/Spacewalk's auth system, by default it will not allow per user access into Cobbler Web and Cobbler XML-RPC. In order to permit this, the following setting must be enabled HOWEVER doing so will permit all Spacewalk/Satellite users of certain types to edit all of Cobbler's configuration. these roles are: `config_admin` and `org_admin`. Users should turn this on only if they want this behavior and do not have a cross-multi-org separation concern. If you have a single org in your satellite, it's probably safe to turn this on and then you can use CobblerWeb alongside a Satellite install.

default: `False`

5.2.67 redhat_management_server

This setting is only used by the code that supports using Uyuni/SUSE Manager/Spacewalk/Satellite authentication within Cobbler Web and Cobbler XML-RPC.

default: `"xmlrpc.rhn.redhat.com"`

5.2.68 redhat_management_key

Specify the default Red Hat authorization key to use to register system. If left blank, no registration will be attempted. Similarly you can set the `--redhat-management-key` to blank on any system to keep it from trying to register.

default: `""`

5.2.69 register_new_installs

If set to `True`, allows `/usr/bin/cobbler-register` (part of the Koan package) to be used to remotely add new Cobbler system records to Cobbler. This effectively allows for registration of new hardware from system records.

default: `False`

5.2.70 remove_old_puppet_certs_automatically

When a puppet managed machine is reinstalled it is necessary to remove the puppet certificate from the puppet master server before a new certificate is signed (see above). Enabling the following feature will ensure that the certificate for the machine to be installed is removed from the puppet master server if the puppet master server is running on the same machine as Cobbler. This requires `puppet_auto_setup` above to be enabled

default: `False`

5.2.71 replicate_repo_rsync_options

Replication rsync options for repos set to override default value of `-avzH`.

default: `"-avzH"`

5.2.72 replicate_rsync_options

replication rsync options for distros, autoinstalls, snippets set to override default value of `-avzH`.

default: `"-avzH"`

5.2.73 reposync_flags

Flags to use for yum's reposync. If your version of yum reposync does not support `-l`, you may need to remove that option.

default: `"-l -n -d"`

5.2.74 reposync_rsync_flags

Flags to use for rysync's reposync. If archive mode (-a,–archive) is used then createrepo is not ran after the rsync as it pulls down the repodata as well. This allows older OS's to mirror modular repos using rsync.

default: `"-rltDv --copy-unsafe-links"`

5.2.75 restart_*

When DHCP and DNS management are enabled, `cobbler sync` can automatically restart those services to apply changes. The exception for this is if using ISC for DHCP, then OMAPI eliminates the need for a restart. `omapi`, however, is experimental and not recommended for most configurations. If DHCP and DNS are going to be managed, but hosted on a box that is not on this server, disable restarts here and write some other script to ensure that the config files get copied/rsynced to the destination box. This can be done by modifying the restart services trigger. Note that if `manage_dhcp` and `manage_dns` are disabled, the respective parameter will have no effect. Most users should not need to change this.

defaults:

```
restart_dns: true
restart_dhcp: true
```

5.2.76 run_install_triggers

Install triggers are scripts in `/var/lib/cobbler/triggers/install` that are triggered in autoinstall pre and post sections. Any executable script in those directories is run. They can be used to send email or perform other actions. They are currently run as root so if you do not need this functionality you can disable it, though this will also disable `cobbler status` which uses a logging trigger to audit install progress.

default: `true`

5.2.77 scm_track_*

enables a trigger which version controls all changes to `/var/lib/cobbler` when add, edit, or sync events are performed. This can be used to revert to previous database versions, generate RSS feeds, or for other auditing or backup purposes. Git and Mercurial are currently supported, but Git is the recommend SCM for use with this feature.

default:

```
scm_track_enabled: false
scm_track_mode: "git"
scm_track_author: "cobbler <cobbler@localhost>"
scm_push_script: "/bin/true"
```

5.2.78 serializer_pretty_json

Sort and indent JSON output to make it more human-readable.

default: `False`

5.2.79 server

This is the address of the Cobbler server – as it is used by systems during the install process, it must be the address or hostname of the system as those systems can see the server. if you have a server that appears differently to different subnets (dual homed, etc), you need to read the `--server-override` section of the manpage for how that works.

default: `127.0.0.1`

5.2.80 sign_puppet_certs_automatically

When puppet starts on a system after installation it needs to have its certificate signed by the puppet master server. Enabling the following feature will ensure that the puppet server signs the certificate after installation if the puppet master server is running on the same machine as Cobbler. This requires `puppet_auto_setup` above to be enabled.

default: `false`

5.2.81 signature_path

The `cobbler import` workflow is powered by this file. Its location can be set with this config option.

default: `/var/lib/cobbler/distro_signatures.json`

5.2.82 signature_url

Updates to the signatures may happen more often then we have releases. To enable you to import new version we provide the most up to date signatures we offer on this like. You may host this file for yourself and adjust it for your needs.

default: `https://cobbler.github.io/signatures/3.0.x/latest.json`

5.2.83 tftpboot_location

This variable contains the location of the tftpboot directory. If this directory is not present Cobbler does not start.

Default: `/srv/tftpboot`

5.2.84 virt_auto_boot

Should new profiles for virtual machines default to auto booting with the physical host when the physical host reboots? This can be overridden on each profile or system object.

default: `true`

5.2.85 webdir

Cobbler's web directory. Don't change this setting – see the Wiki on "relocating your Cobbler install" if your /var partition is not large enough.

default: `@@webroot@@/cobbler`

5.2.86 webdir_whitelist

Directories that will not get wiped and recreated on a `cobbler sync`.

default:

```
webdir_whitelist:
  - misc
  - web
  - webui
  - localmirror
  - repo_mirror
  - distro_mirror
  - images
  - links
  - pub
  - repo_profile
  - repo_system
  - svc
  - rendered
  - .link_cache
```

5.2.87 xmlrpc_port

Cobbler's public XML-RPC listens on this port. Change this only if absolutely needed, as you'll have to start supplying a new port option to Koan if it is not the default.

default: `25151`

5.2.88 yum_distro_priority

The default yum priority for all the distros. This is only used if yum-priorities plugin is used. 1 is the maximum value. Tweak with caution.

default: `true`

5.2.89 yum_post_install_mirror

`cobbler repo add` commands set Cobbler up with repository information that can be used during autoinstall and is automatically set up in the Cobbler autoinstall templates. By default, these are only available at install time. To make these repositories usable on installed systems (since Cobbler makes a very convenient mirror) set this to `True`. Most users can safely set this to `True`. Users who have a dual homed Cobbler server, or are installing laptops that will not always have access to the Cobbler server may wish to leave this as `False`. In that case, the Cobbler mirrored yum repos are still accessible at `http://cobbler.example.org/cblr/repo_mirror` and YUM configuration can still be done manually. This is just a shortcut.

default: `True`

5.2.90 yumdownloader_flags

Flags to use for yumdownloader. Not all versions may support `--resolve`.

default: `"--resolve"`

5.3 modules.conf

If you have own custom modules which are not shipped with Cobbler directly you may have additional sections here.

5.3.1 authentication

What users can log into the WebUI and Read-Write XML-RPC?

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:

- *Web-Interface*
- https://cobbler.readthedocs.io/en/release28/5_web-interface/security_overview.html
- https://cobbler.readthedocs.io/en/release28/5_web-interface/web_authentication.html#defer-to-apache-kerberos
- https://cobbler.readthedocs.io/en/release28/5_web-interface/web_authentication.html#ldap

default: `authn_configfile`

5.3.2 authorization

Once a user has been cleared by the WebUI/XML-RPC, what can they do?

Choices:

- authz_allowall – full access for all authenticated 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. Configuration file does but does not support object ownership which is useful as an additional layer of control.

For more information:

- *Web-Interface*
- https://cobbler.readthedocs.io/en/release28/5_web-interface/security_overview.html
- https://cobbler.readthedocs.io/en/release28/5_web-interface/web_authentication.html

default: `authz_allowall`

5.3.3 dns

Chooses the DNS management engine if `manage_dns` is enabled in `/etc/cobbler/settings.yaml`, which is off by default.

Choices:

- manage_bind – default, uses BIND/named
- manage_dnsmasq – uses dnsmasq, also must select dnsmasq for DHCP below
- manage_ndjbdns – uses ndjbdns

NOTE: More configuration is still required in `/etc/cobbler`

For more information see *DNS configuration management*.

default: `manage_bind`

5.3.4 dhcp

Chooses the DHCP management engine if `manage_dhcp` is enabled in `/etc/cobbler/settings.yaml`, which is off by default.

Choices:

- manage_isc – default, uses ISC dhcpd
- manage_dnsmasq – uses dnsmasq, also must select dnsmasq for DNS above

NOTE: More configuration is still required in `/etc/cobbler`

For more information see *DHCP Management*.

default: `manage_isc`

5.3.5 tftpd

Chooses the TFTP management engine if `manage_tftpd` is enabled in `/etc/cobbler/settings.yaml`, which is **on** by default.

Choices:

- manage_in_tftpd – default, uses the system's TFTP server
- manage_tftpd_py – uses Cobbler's TFTP server

default: `manage_in_tftpd`

CHAPTER 6

User Guide

6.1 Web-Interface

Please be patient until we have time to rework this section or please file a PR for this section.

The standard login for the WebUI can be read below. We would recommend to change this as soon as possible!

Username: `cobbler` Password: `cobbler`

6.1.1 Old Release 2.8.x

https://cobbler.readthedocs.io/en/release28/web-interface.html

6.1.2 Old GitHub-Wiki Entry

Most of the day-to-day actions in cobbler's command line can be performed in Cobbler's Web UI.

With the web user interface (WebUI), you can:

- View all of the cobbler objects and the settings
- Add and delete a system, distro, profile, or system
- Run the equivalent of a `cobbler sync`
- Edit kickstart files (which must be in `/etc/cobbler` and `/var/lib/cobbler/kickstarts`)

You cannot (yet):

- Auto-Import media
- Auto-Import a rsync mirror of install trees
- Do a `cobbler reposync` to mirror or update yum content
- Do a `cobbler validateks`

The WebUI can be very good for day-to-day configuring activities, but the CLI is still required for basic bootstrapping and certain other activities.

The WebUI is intended to be self-explanatory and contains tips and explanations for nearly every field you can edit. It also contains links to additional documentation, including the Cobbler manpage documentation in HTML format.

Who logs in and what they can access is controlled by Web Authentication and Web Authorization. The default options are mostly good for getting started, but for safety reasons the default authentication is "denyall" so you will at least need to address that.

Basic Setup

1. You must have installed the cobbler-web package
2. Your `/etc/httpd/conf.d/cobbler_web.conf` should look something like this:

```
# This configuration file enables the cobbler web interface (django version)
# Force everything to go to https
RewriteEngine on
RewriteCond %{HTTPS} off
RewriteCond %{REQUEST_URI} ^/cobbler_web
RewriteRule (.*) https://%{HTTP_HOST}%{REQUEST_URI}

WSGIScriptAlias /cobbler_web /usr/share/cobbler/web/cobbler.wsgi

# The following Directory Entry in Apache Configs solves 403 Forbidden errors.
<Directory "/usr/share/cobbler/web">
  Order allow,deny
  Allow from all
</Directory>

# Display Cobbler Themes + Logo graphics.
<Directory "/var/www/cobbler_webui_content">
Order allow,deny
Allow from all
</Directory>
```

3. Your `/etc/cobbler/modules.conf` should look something like this:

```
[authentication]
module = authn_configfile

[authorization]
module = authz_allowall
```

4. You should change the password for the 'cobbler' username, see *Managing users.digest*.
5. If this is not a new install, your Apache configuration for Cobbler might not be current.

```
cp /etc/httpd/conf.d/cobbler.conf.rpmnew /etc/httpd/conf.d/cobbler.conf
```

6. Now restart Apache and `cobblerd`.

```
/sbin/service cobblerd restart
/sbin/service httpd restart
```

7. If you use SELinux, you may also need to set the following, so that the WebUI can connect with the XMLRPC:

```
setsebool -P httpd_can_network_connect true
```

Basic setup (2.2.x and higher)

In addition to the steps above, cobbler 2.2.x has a requirement for `mod_wsgi` which, when installed via EPEL, will be disabled by default. Attempting to start httpd will result in:

```
Invalid command 'WSGIScriptAliasMatch', perhaps misspelled \
  or defined by a module not included in the server configuration
```

You can enable this module by editing `/etc/httpd/conf.d/wsgi.conf` and un-commenting the "LoadModule wsgi_module modules/mod_wsgi.so" line.

Next steps

It should be ready to go. From your web browser visit the URL on your bootserver that resembles:

```
https://bootserver.example.com/cobbler_web
```

and log in with the username (usually cobbler) and password that you set earlier.

Should you ever need to debug things, see the following log files:

```
/var/log/httpd/error_log
/var/log/cobbler/cobbler.log
```

Managing users.digest

Cobbler authenticates all WebUI logins through `cobblerd`, which uses a configurable authentication mechanism. You may wish to adjust that for your environment. For instance, if in `modules.conf` above you choose to stay with the `authentication.configfile` module, you may want to add your system administrator usernames to the digest file.

Because the generated password isn't supported by the `htdigest` command you have to generate the entries yourself, and to generate the password hashes it is recommended to use either `openssl` or Python directly.

The entry format should be, where `Cobbler` is the realm:

```
username:realm:hash
```

Example using `openssl 1.1.1` or later:

```
printf "foobar" | openssl dgst -sha3-512
```

It is possible with `openssl` to generate hashes for the following hash algorithms which are configurable: blake2b512, blake2s256, shake128, shake256, sha3-224m sha3-256, sha3-384, sha3-512

Example using Python (using the python interactive shell):

```
import hashlib
hashlib.sha3_512("<PASSWORD>".encode('utf-8')).hexdigest()
```

Python of course will always have all possible hash algorithms available which are valid in the context of Cobbler.

Both examples return the same result when executed with the same password. The file itself is structured according to the following: `<USERNAME>:<REALM>:<PASSWORDHASH>`. Normally `<REALM>` will be `Cobbler`. Other values are currently not valid. Please add the user, realm and passwordhash with your preferred editor. Normally there should be no need to restart cobbler when a new user is added, removed or the password is changed. The authentication process reads the file every time a user is authenticated.

You may also want to refine for authorization settings.

Before Cobbler 3.1.2 it was recommended to do edit the file `users.digest` with the following command. Since `md5` is not FIPS compatible from Cobbler 3.1.3 and onwards this is not possible anymore. The file was also just

read once per Cobbler start and thus a change of the data requires that Cobbler is restarted that it picks up these changes.

```
htdigest /etc/cobbler/users.digest "Cobbler" <username>
```

Rewrite Rule for secure-http

To redirect access to the WebUI via HTTPS on an Apache webserver, you can use the following rewrite rule, probably at the end of Apache's `ssl.conf`:

```
### Force SSL only on the WebUI
<VirtualHost *:80>
    <LocationMatch "^/cobbler_web/*">
       RewriteEngine on
       RewriteRule ^(.*) https://%{SERVER_NAME}/%{REQUEST_URI} [R,L]
   </LocationMatch>
</VirtualHost>
```

6.2 Configuration Management Integrations

Cobbler contains features for integrating an installation environment with a configuration management system, which handles the configuration of the system after it is installed by allowing changes to configuration files and settings.

Resources are the lego blocks of configuration management. Resources are grouped together via Management Classes, which are then linked to a system. Cobbler supports two (2) resource types. Resources are configured in the order listed below.

The initial provisioning of client systems with cobbler is just one component of their management. We also need to consider how to continue to manage them using a configuration management system (CMS). Cobbler can help you provision and introduce a CMS onto your client systems.

One option is cobbler's own lightweight CMS. For that, see the document *Built-In Configuration Management*.

Here we discuss the other option: deploying a CMS such as cfengine3, puppet, bcfg2, Chef, etc.

Cobbler doesn't force you to chose a particular CMS (or to use one at all), though it helps if you do some things to link cobbler's profiles with the "profiles" of the CMS. This, in general, makes management of both a lot easier.

Note that there are two independent "variables" here: the possible client operating systems and the possible CMSes. We don't attempt to cover all details of all combinations; rather we illustrate the principles and give a small number of illustrative examples of particular OS/CMS combinations. Currently cobbler has better support for Red Hat based OSes and for Puppet so the current examples tend to deal with this combination.

6.2.1 Background considerations

Machine lifecycle

A typical computer has a lifecycle something like:

- installation
- initial configuration
- ongoing configuration and maintenance
- decommissioning

Typically installation happens once. Likewise, the initial configuration happens once, usually shortly after installation. By contrast ongoing configuration evolves over an extended period, perhaps of several years. Sometimes part of that ongoing configuration may involve re-installing an OS from scratch. We can regard this as repeating the earlier phase.

We need not consider decommissioning here.

Installation clearly belongs (in our context) to Cobbler. In a complementary manner, ongoing configuration clearly belongs to the CMS. But what about initial configuration?

Some sites consider their initial configuration as the final phase of installation: in our context, that would put it at the back end of Cobbler, and potentially add significant configuration-based complication to the installation-based Cobbler set-up.

But it is worth considering initial configuration as the first step of ongoing configuration: in our context that would put it as part of the CMS, and keep the Cobbler set-up simple and uncluttered.

Local package repositories

Give consideration to:

- local mirrors of OS repositories
- local repository of local packages
- local repository of pick-and-choose external packages

In particular consider having the packages for your chosen CMS in one of the latter.

Package management

Some sites set up Cobbler always to deploy just a minimal subset of packages, then use the CMS to install many others in a large-scale fashion. Other sites may set up Cobbler to deploy tailored sets of packages to different types of machines, then use the CMS to do relatively small-scale fine-tuning of that.

6.2.2 General scheme

We need to consider getting Cobbler to install and automatically invoke the CMS software.

Set up Cobbler to include a package repository that contains your chosen CMS:

```
cobbler repo add ...
```

Then (illustrating a Red Hat/Puppet combination) set up the kickstart file to say something like:

```
%packages
puppet

%post
/sbin/chkconfig --add puppet
```

The detail may need to be more substantial, requiring some other associated local packages, files and configuration. You may wish to manage this through [Kickstart snippets](Kickstart Snippets).

David Lutterkort has a walkthrough for kickstart. While his example is written for Red Hat (Fedora) and Puppet, the principles are useful for other OS/CMS combinations.

6.2.3 Built-In Configuration Management

Cobbler is not just an installation server, it can also enable two different types of ongoing configuration management system (CMS):

- integration with an established external CMS such as cfengine3, bcfg2, Chef, or puppet, discussed [elsewhere](Using cobbler with a configuration management system);
- its own, much simpler, lighter-weight, internal CMS, discussed here.

Setting up

Cobbler's internal CMS is focused around packages and templated configuration files, and installing these on client systems.

This all works using the same Cheetah-powered templating engine used in Kickstart Templating, so once you learn about the power of treating your distribution answer files as templates, you can use the same templating to drive your CMS configuration files.

For example:

```
cobbler profile edit --name=webserver --template-files=/srv/cobbler/x.template=/
↪etc/foo.conf
```

A client system installed via the above profile will gain a file `/etc/foo.conf` which is the result of rendering the template given by `/srv/cobbler/x.template`. Multiple files may be specified; each `template=destination` pair should be placed in a space-separated list enclosed in quotes:

```
--template-files="srv/cobbler/x.template=/etc/xfile.conf srv/cobbler/y.template=/
↪etc/yfile.conf"
```

Template files

Because the template files will be parsed by the Cheetah parser, they must conform to the guidelines described in Kickstart Templating. This is particularly important when the file is generated outside a Cheetah environment. Look for, and act on, Cheetah 'ParseError' errors in the Cobbler logs.

Template files follows general Cheetah syntax, so can include Cheetah variables. Any variables you define anywhere in the cobbler object hierarchy (distros, profiles, and systems) are available to your templates. To see all the variables available, use the command:

```
cobbler profile dumpvars --name=webserver
```

Cobbler snippets and other advanced features can also be employed.

Ongoing maintenance

Koan can pull down files to keep a system updated with the latest templates and variables:

```
koan --server=cobbler.example.org --profile=foo --update-files
```

You could also use `--server=bar` to retrieve a more specific set of templating. Koan can also autodetect the server if the MAC address is registered.

Further uses

This Cobbler/Cheetah templating system can serve up templates via the magic URLs (see "Leveraging Mod Python" below). To do this ensure that the destination path given to any `--template-files` element is relative, not absolute; then Cobbler and Koan won't download those files.

For example, in:

```
cobbler profile edit --name=foo --template-files="/srv/templates/a.src=/etc/foo/a.
↪conf /srv/templates/b.src=1"
```

Cobbler and koan would automatically download the rendered `a.src` to replace the file `/etc/foo/a.conf`, but the `b.src` file would not be downloaded to anything because the destination pathname `1` is not absolute.

This technique enables using the Cobbler/Cheetah templating system to build things that other systems can fetch and use, for instance, BIOS config files for usage from a live environment.

Leveraging Mod Python

All template files are generated dynamically at run-time. If a change is made to a template, a `--ks-meta` variable or some other variable in cobbler, the result of template rendering will be different on subsequent runs. This is covered in more depth in the Developer documentation.

Possible future developments

- Serving and running scripts via `--update-files` (probably staging them through `/var/spool/koan`).
- Auto-detection of the server name if `--ip` is registered.

6.2.4 Terraform Provider

This is developed and maintained by the Cobbler community. You will find more information in the docs under https://registry.terraform.io/providers/cobbler/cobbler/latest/docs.

The code for the Terraform-Provider can be found at: https://github.com/cobbler/terraform-provider-cobbler

6.2.5 Ansible

Although we currently can not provide something official we can indeed link some community work here:

- https://github.com/ac427/my_cm
- https://github.com/AnKosteck/ansible-cluster
- https://github.com/osism/ansible-cobbler
- https://github.com/hakoerber/ansible-roles

6.2.6 Saltstack

Although we currently can not provide something official we can indeed link some community work here:

- https://github.com/hakoerber/salt-states/tree/master/cobbler

6.2.7 Vagrant

Although we currently can not provide something official we can indeed link some community work here:

- https://github.com/davegermiquet/vmwarevagrantcobblercentos
- https://github.com/dratushnyy/tools
- https://github.com/mkusanagi/cobbler-kickstart-playground

6.2.8 Puppet

There is also an example of Puppet deploying Cobbler: https://github.com/gothicfann/puppet-cobbler

This example is relatively advanced, involving Cobbler "mgmt-classes" to control different types of initial configuration. But if instead you opt to put most of the initial configuration into the Puppet CMS rather than here, then things could be simpler.

Keeping Class Mappings In Cobbler

First, we assign management classes to distro, profile, or system objects.

```
cobbler distro edit --name=distro1 --mgmt-classes="distro1"
cobbler profile add --name=webserver --distro=distro1 --mgmt-classes="webserver␣
↪likes_llamas" --autoinstall=/etc/cobbler/my.ks
cobbler system edit --name=system --profile=webserver --mgmt-classes="orange" --
↪dns-name=system.example.org
```

For Puppet, the `--dns-name` (shown above) must be set because this is what puppet will be sending to cobbler and is how we find the system. Puppet doesn't know about the name of the system object in cobbler. To play it safe you probably want to use the FQDN here (which is also what you want if you were using Cobbler to manage your DNS, which you don't have to be doing).

External Nodes

For more documentation on Puppet's external nodes feature, see https://docs.puppetlabs.com.

Cobbler provides one, so configure puppet to use `/usr/bin/cobbler-ext-nodes`:

```
[main]
external_nodes = /usr/bin/cobbler-ext-nodes
```

Note: if you are using puppet 0.24 or later then you will want to also add the following to your configuration file.

```
node_terminus = exec
```

You may wonder what this does. This is just a very simple script that grabs the data at the following URL, which is a URL that always returns a YAML document in the way that Puppet expects it to be returned. This file contains all the parameters and classes that are to be assigned to the node in question. The magic URL being visited is powered by Cobbler.

```
http://cobbler/cblr/svc/op/puppet/hostname/foo
```

(for developer information about this magic URL, visit https://fedorahosted.org/cobbler/wiki/ModPythonDetails)

And this will return data such as:

```
---
classes:
    - distro1
    - webserver
    - likes_llamas
    - orange
parameters:
    tree: 'http://.../x86_64/tree'
```

Where do the parameters come from? Everything that cobbler tracks in `--ks-meta` is also a parameter. This way you can easily add parameters as easily as you can add classes, and keep things all organized in one place.

What if you have global parameters or classes to add? No problem. You can also add more classes by editing the following fields in `/etc/cobbler/settings.yaml`:

```
# cobbler has a feature that allows for integration with config management
# systems such as Puppet.  The following parameters work in conjunction with

# --mgmt-classes  and are described in furhter detail at:
# https://fedorahosted.org/cobbler/wiki/UsingCobblerWithConfigManagementSystem
mgmt_classes: []
mgmt_parameters:
   from_cobbler: 1
```

Alternate External Nodes Script

Attached at `puppet_node.py` is an alternate external node script that fills in the nodes with items from a manifests repository (at `/etc/puppet/manifests/`) and networking information from cobbler. It is configured like the above from the puppet side, and then looks for `/etc/puppet/external_node.yaml` for cobbler side configuration. The configuration is as follows.

```
base: /etc/puppet/manifests/nodes
cobbler: <%= cobbler_host %>
no_yaml: puppet::noyaml
no_cobbler: network::nocobbler
bad_yaml: puppet::badyaml
unmanaged: network::unmanaged
```

The output for network information will be in the form of a pseudo data structure that allows puppet to split it apart and create the network interfaces on the node being managed.

6.2.9 cfengine support

Documentation to be added

6.2.10 bcfg2 support

Documentation to be added

6.2.11 Chef

Documentation to be added.

There is some integration information on bootstrapping chef clients with cobbler in [this blog article](http://blog.milford.io/2012/03/getting-a-basic-cobbler-server-going-on-centos/)

6.2.12 Conclusion

Hopefully this should get you started in linking up your provisioning configuration with your CMS implementation. The examples provided are for Puppet, but we can (in the future) presumably extend `--mgmt-classes` to work with other tools... Just let us know what you are interested in, or perhaps take a shot at creating a patch for it.

6.2.13 Attachments

- [puppet_node.py](/cobbler/attachment/wiki/UsingCobblerWithConfigManagementSystem/puppet_node.py) (2.5 kB) -Alternate External Nodes Script, added by shenson on 12/09/10 20:33:36.

6.3 Automatic Windows installation with Cobbler

One of the challenges for creating your own Windows network installation scenario with Cobbler is preparing the necessary files in a Linux environment. However, generating the necessary binaries can be greatly simplified by using the cobbler post trigger on the sync command. Below is an example of such a trigger, which prepares the necessary files for legacy BIOS mode boot. Boot to UEFI Mode with iPXE is simpler and can be implemented by replacing the first 2 steps and several others with creating an iPXE boot menu.

Trigger `sync_post_wingen.py`:

- some of the files are created from standard ones (pxeboot.n12, bootmgr.exe) by directly replacing one string with another directly in the binary
- in the process of changing the `bootmgr.exe` file, the checksum of the PE file will change and it needs to be recalculated. The trigger does this with `python-pefile`
- `python3-hivex` is used to modify Windows boot configuration data (BCD). For pxelinux distro boot_loader in BCD, paths to `winpe.wim` and `boot.sdi` are generated as `/winos/<distro_name>/boot`, and for iPXE - `\Boot`.
- uses `wimlib-tools` to replace `startnet.cmd startup` script in WIM image

Windows answer files (`autounattended.xml`) are generated using Cobbler templates, with all of its conditional code generation capabilities, depending on the Windows version, architecture (32 or 64 bit), installation profile, etc.

startup scripts for WIM images (startnet.cmd) and a script that is launched after OS installation (`post_install.cmd`) are also generated from templates

Post-installation actions such as installing additional software, etc., are performed using the Automatic Installation Template (`win.ks`).

A logically automatic network installation of Windows 7 and newer can be represented as follows:

PXE + Legacy BIOS Boot

```
Original files: pxeboot.n12 → bootmgr.exe → BCD → winpe.wim → startnet.cmd →␣
↪autounattended.xml
Cobbler profile 1: pxeboot.001 → boot001.exe → 001 → wi001.wim → startnet.cmd →␣
↪autounatten001.xml → post_install.cmd profile_name
...
```

iPXE + UEFI BIOS Boot

```
Original files: ipxe-x86_64.efi → wimboot → bootx64.efi → BCD → winpe.wim →␣
↪startnet.cmd → autounattended.xml
Cobbler profile 1: ipxe-x86_64.efi → wimboot → bootx64.efi → 001 → wi001.wim →␣
↪startnet.cmd → autounatten001.xml → post_install.cmd profile_name
...
```

For older versions (Windows XP, 2003):

```
Original files: pxeboot.n12 → setupldr.exe → winnt.sif → post_install.cmd␣
↪profile_name
Cobbler profile <xxx>: pxeboot.<xxx> → setup<xxx>.exe → wi<xxx>.sif → post_
↪install.cmd profile_name
```

6.3.1 Preparing for an unattended network installation of Windows

- `dnf install python3-pefile python3-hivex wimlib-utils`
- In the server's tftp directory, create a directory winos

```
mkdir /var/lib/tftpboot/winos
```

and copy the Windows distributions there:

```
dr-xr-xr-x. 1 root   root         200 Mar 23  2017 Win10_EN-x64
dr-xr-xr-x. 1 root   root         238 Aug  7  2015 Win2012-Server_EN-x64
dr-xr-xr-x. 1 root   root         220 May 17  2019 Win2016-Server_EN-x64
drwxr-xr-x. 1 root   root         236 Dec  3 22:42 Win2019-Server_EN-x64
dr-xr-xr-x. 1 root   root         788 Aug  8  2015 Win2k3-Server_EN-x64
dr-xr-xr-x. 1 root   root         196 Sep 24  2017 Win2k8-Server_EN-x64
dr-xr-xr-x. 1 root   root         132 Aug  8  2015 Win7_EN-x64
dr-xr-xr-x. 1 root   root         238 Aug  7  2015 Win8_EN-x64
dr-xr-xr-x. 1 root   root         456 Aug  8  2015 WinXp_EN-i386
```

Copy the following files to the distributions directories (for Windows 7 and newer):

PXE + Legacy BIOS Boot

- pxeboot.n12
- bootmgr.exe
- boot/BCD
- boot/boot.sdi

iPXE + UEFI BIOS Boot

- ipxe-x86_64.efi
- ipxe-x86_64.efi
- wimboot
- boot/bootx64.efi
- boot/BCD
- boot/boot.sdi

- Share `/var/lib/tftpboot/winos` via Samba:

```
vi /etc/samba/smb.conf
        [WINOS]
        path = /var/lib/tftpboot/winos
        guest ok = yes
        browseable = yes
        public = yes
        writeable = no
        printable = no
```

- You can use `tftpd.rules` to indicate the actual locations of the bootmgr.exe and BCD files generated by the trigger.

```
cp /usr/lib/systemd/system/tftp.service /etc/systemd/system
```

Replace the line in the `/etc/systemd/system/tftp.service`

```
ExecStart=/usr/sbin/in.tftpd -s /var/lib/tftpboot
    to:
ExecStart=/usr/sbin/in.tftpd -m /etc/tftpd.rules -s /var/lib/tftpboot
```

Create a file /etc/tftpd.rules:

```
vi /etc/tftpd.rules
rg  \\                                  / # Convert backslashes to slashes
r   (BOOTFONT\.BIN)                 /winos/\1
r   (/Boot/Fonts/)(.*)                  /winos/Fonts/\2

r   (ntdetect\....)                 /winos/\1

r   (wine.\.sif)                        /WinXp_EN-i386/\1
r   (xple.)                             /WinXp_EN-i386/\1
r   (/WinXp...-i386/)(.*)               /winos\1\L\2

r   (wi2k.\.sif)                        /Win2k3-Server_EN-x64/\1
r   (w2k3.)                             /Win2K3-Server_EN-x64/\1
r   (/Win2k3-Server_EN-x64/)(.*)        /winos\1\L\2

r   (boot7e.\.exe)                      /winos/Win7_EN-x64/\1
r   (/Boot/)(7E.)                       /winos/Win7_EN-x64/boot/\2

r   (boot28.\.exe)                      /winos/Win2k8-Server_EN-x64/\1
r   (/Boot/)(28.)                       /winos/Win2k8-Server_EN-x64/boot/\2

r   (boot9r.\.exe)                      /winos/Win2019-Server_EN-x64/\1
r   (/Boot/)(9r.)                       /winos/Win2019-Server_EN-x64/boot/\2

r   (boot6e.\.exe)                      /winos/Win2016-Server_EN-x64/\1
r   (/Boot/)(6e.)                       /winos/Win2016-Server_EN-x64/boot/\2

r   (boot2e.\.exe)                      /winos/Win2012-Server_EN-x64/\1
r   (/Boot/)(2e.)                       /winos/Win2012-Server_EN-x64/boot/\2

r   (boot81.\.exe)                      /winos/Win8_EN-x64/\1
r   (/Boot/)(B8.)                       /winos/Win8_EN-x64/boot/\2

r   (boot1e.\.exe)                      /winos/Win10_EN-x64/\1
r   (/Boot/)(1E.)                       /winos/Win10_EN-x64/boot/\2
```

- Add information about Windows distributions to the `distro_signatures.json` file

```
"windows": {
 "2003": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub"]}
 },
 "2008": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub","ipxe"]}
 },
 "2012": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub","ipxe"]}
 },
 "2016": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub","ipxe"]}
 },
 "2019": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub","ipxe"]}
 },
 "XP": {
  "supported_arches":["i386","x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub"]}
```

(continues on next page)

(continued from previous page)

```
 },
 "7": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub","ipxe"]}
 },
 "8": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub","ipxe"]}
 },
 "10": {
  "supported_arches":["x86_64"],
  "boot_loaders":{"x86_64":["pxelinux","grub","ipxe"]}
 }
},
```

- Add trigger /usr/lib/python3.9/site-packages/cobbler/modules/sync_post_wingen.py

6.3.2 Cobbler Windows Templates

- `/var/lib/tftpboot/winos/startnet.template` is used to generate /Windows/System32/startnet.cmd script in WIM image.

Example:

```
wpeinit

ping 127.0.0.1 -n 10 >nul
md \tmp
cd \tmp
ipconfig /all | find "DHCP Server" > dhcp
ipconfig /all | find "IPv4 Address" > ipaddr
FOR /F "eol=- tokens=2 delims=:" %%i in (dhcp) do set dhcp=%%i
FOR  %%i in (%dhcp%) do set dhcp=%%i
FOR /F "eol=- tokens=2 delims=:(" %%i in (ipaddr) do set ipaddr=%%i

net use y: \\@@http_server@@\Public /user:install install
#set $distro_dir = '\\\\' + $http_server + '\\WINOS\\' + $distro_name
net use z: $distro_dir /user:install install
set exit_code=%ERRORLEVEL%
IF %exit_code% EQU 0 GOTO GETNAME
echo "Can't mount network drive"
goto EXIT

:GETNAME
y:\windows\bind\nslookup.exe %ipaddr% | find "name =" > wsname
for /f "eol=- tokens=2 delims==" %%i in (wsname) do echo %%i > ws
for /f "eol=- tokens=1 delims=." %%i in (ws) do set wsname=%%i
FOR  %%i in (%wsname%) do set wsname=%%i

#set $unattended = "set UNATTENDED_ORIG=Z:\\sources\\" + $kernel_options["sif"]
$unattended
set UNATTENDED=X:\tmp\autounattended.xml

echo off
FOR /F "tokens=1 delims=!" %%l in (%UNATTENDED_ORIG%) do (
   IF "%%l"==""            <ComputerName>*</ComputerName>" (
     echo             ^<ComputerName^>%wsname%^<^/ComputerName^>>> %UNATTENDED%
   ) else (
     echo %%l>> %UNATTENDED%
   )
```

(continues on next page)

(continued from previous page)

```
)
echo on

:INSTALL
set n=0
z:\sources\setup.exe /unattend:%UNATTENDED%
set /a n=n+1
ping 127.0.0.1 -n 5 >nul
IF %n% lss 20 goto INSTALL

:EXIT
```

- Templates `/var/lib/tftpboot/winos/{winpe7,winpe8 }.template` are standard or customized WIM PE images. The trigger copies to the directory of the corresponding distro and changes the contents of `startnet.cmd` based on the corresponding template and Cobbler profile. winpe7 is used for Windows 7 and Windows 2008 Server, and winpe8 for newer versions.
- `/var/lib/tftpboot/winos/win_sif.template` is used to generate `/var/lib/tftpboot/winos/<distro_name>/sources/autounattended.xml` in case of Windows 7 and newer or winnt.sif for Windows XP, 2003

Example:

```
#if $arch == 'x86_64'
        #set $win_arch = 'amd64'
#else if $arch == 'i386'
        #set $win_arch = 'i386'
#end if

#set $OriSrc = '\\\\' + $http_server + '\\WINOS\\' + $distro_name + '\\' + $win_
↪arch
#set $DevSrc = '\\Device\\LanmanRedirector\\' + $http_server + '\\WINOS\\' +
↪$distro_name

#if $distro_name in ( 'WinXp_EN-i386', 'Win2k3-Server_EN-x64' )
[Data]
floppyless = "1"
msdosinitiated = "1"
; Needed for second stage
OriSrc="$OriSrc"
OriTyp="4"
LocalSourceOnCD=1
DisableAdminAccountOnDomainJoin=0
AutomaticUpdates="No"
Autopartition="0"
UnattendedInstall="Yes"
<..>
[GuiRunOnce]
"%Systemdrive%\post_install.cmd @@profile_name@@"
<..>
#else if $distro_name in ('Win7_EN-x64', 'Win2k8-Server_EN-x64', 'Win2012-Server_
↪EN-x64', 'Win2016-Server_EN-x64', 'Win2019-Server_EN-x64', 'Win8_EN-x64', 'Win10_
↪EN-x64' )
<?xml version="1.0" encoding="utf-8"?>
<unattend xmlns="urn:schemas-microsoft-com:unattend">
#if $distro_name in ( 'Win2012-Server_EN-x64' )
    <servicing>
        <package action="configure">
<..>
            </DiskConfiguration>
            <ImageInstall>
```

(continues on next page)

(continued from previous page)

```
                <OSImage>
                    <InstallFrom>
                        <Credentials>
                            <Domain></Domain>
                        </Credentials>
                        <MetaData wcm:action="add">
                            <Key>/IMAGE/NAME</Key>
#else if $distro_name in ( 'Win7_EN-x64' )
                            <Value>Windows 7 PROFESSIONAL</Value>
#else if $distro_name in ( 'Win2k8-Server_EN-x64' )
                            <Value>Windows Server 2008 R2 SERVERENTERPRISE</Value>
<..>
        <component name="Microsoft-Windows-PnpCustomizationsWinPE"␣
↪processorArchitecture="amd64" publicKeyToken="31bf3856ad364e35" language="neutral
↪" versionScope="nonSxS" xmlns:wcm="http://schemas.microsoft.com/WMIConfig/2002/
↪State" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
            <DriverPaths>
#if $distro_name in ( 'Win2012-Server_EN-x64', 'Win8_EN-x64' )
                <PathAndCredentials wcm:action="add" wcm:keyValue="1">
                    <Path>\\@@http_server@@\WINOS\Drivers\CHIPSET\Win8</Path>
                </PathAndCredentials>
<..>
            <FirstLogonCommands>
                <SynchronousCommand wcm:action="add">
                    <RequiresUserInput>false</RequiresUserInput>
                    <Order>1</Order>
                    <CommandLine>c:\post_install.cmd @@profile_name@@</CommandLine>
                </SynchronousCommand>
            </FirstLogonCommands>
<..>
```

- The `post_inst_cmd.template` is used to generate a script that is launched after OS installation in the <FirstLogonCommands> `autounattended.xml` section, or [GuiRunOnce] in `winnt.sif`

Example:

```
%systemdrive%
CD %systemdrive%\TMP >nul 2>&1
$SNIPPET('my/win_wait_network_online')
wget.exe http://@@http_server@@/cblr/svc/op/ks/profile/%1
MOVE %1 install.cmd
todos.exe install.cmd
start /wait install.cmd
DEL /F /Q libeay32.dll >nul 2>&1
DEL /F /Q libiconv2.dll >nul 2>&1
DEL /F /Q libintl3.dll >nul 2>&1
DEL /F /Q libssl32.dll >nul 2>&1
DEL /F /Q wget.exe >nul 2>&1
DEL /F /Q %0 >nul 2>&1
```

For the script to work, you need to place the following files in the /var/lib/tftpboot/winos/<distro_name>/OEM/$1/TMP directory:

```
ls -l '/var/lib/tftpboot/winos/Win10_EN-x64/$OEM$/$1/TMP'
total 2972
-rwxr-xr-x. 1 root root 1177600 Sep  4  2008 libeay32.dll
-rwxr-xr-x. 1 root root 1008128 Mar 15  2008 libiconv2.dll
-rwxr-xr-x. 1 root root  103424 May  6  2005 libintl3.dll
-rwxr-xr-x. 1 root root  232960 Sep  4  2008 libssl32.dll
-rwxr-xr-x. 1 root root    4880 Oct 26  1999 sleep.exe
-rwxr-xr-x. 1 root root   52736 Oct 27  2013 todos.exe
-rwxr-xr-x. 1 root root  449024 Dec 31  2008 wget.exe
```

The `win_wait_network_online` snippet might look something like this:

```
:wno10
set n=0

:wno20
ping @@http_server@@ -n 3
set exit_code=%ERRORLEVEL%

IF %exit_code% EQU 0 GOTO wno_exit
set /a n=n+1
IF %n% lss 30 goto wno20
pause
goto wno10

:wno_exit
```

- `win.ks` - Automatic Installation Template, which is specified for the Cobbler profile in `cobbler profile add/edit --autoinstall=win.ks ..` command.

Example:

```
$SNIPPET('my/win_wait_network_online')

set n=0

:mount_y
net use y: \\@@http_server@@\Public /user:install install
set exit_code=%ERRORLEVEL%

IF %exit_code% EQU 0 GOTO mount_z
set /a n=n+1
IF %n% lss 20 goto mount_y
PAUSE
goto mount_y

set n=0

:mount_z
net use z: \\@@http_server@@\winos /user:install install
set exit_code=%ERRORLEVEL%

IF %exit_code% EQU 0 GOTO mount_exit
set /a n=n+1
IF %n% lss 20 goto mount_z
PAUSE
goto mount_z

:mount_exit
if exist %systemdrive%\TMP\stage.dat goto flag005
echo 0 > %systemdrive%\TMP\stage.dat

$SNIPPET('my/win_check_virt')

#if $distro_name in ( 'WinXp_EN-i386', 'Win2k3-Server_EN-x64' )
z:\Drivers\wsname.exe /N:$DNS /NOREBOOT
#else
REM pause
#end if
echo Windows Registry Editor Version 5.00 > %systemdrive%\TMP\install.reg
echo [HKEY_CURRENT_USER\Software\Microsoft\Windows\CurrentVersion\RunOnce] >>
↪%systemdrive%\TMP\install.reg
echo "DD"="C:\\TMP\\install.cmd" >> %systemdrive%\TMP\install.reg
```

(continues on next page)

(continued from previous page)

```
$SNIPPET('my/win_install_drivers')

#if $distro_name == 'Win2k3-Server_EN-x64'
start /wait z:\Win2K3-Server_EN-x64\cmpnents\r2\setup2.exe /q /a /sr
start /wait y:\Windows\Win2003\IE8-WindowsServer2003-x64-ENU.exe /passive /update-
↪no /norestart
if %virt% equ NO REG IMPORT y:\Windows\Win2003\vm.reg
#end if
REG IMPORT %systemdrive%\TMP\install.reg
net use Y: /delete
net use Z: /delete
%systemdrive%\TMP\sleep.exe 10
exit

:flag005
for /f "tokens=*" %%i in (%systemdrive%\TMP\stage.dat) do set stage=%%i
echo 1 > %systemdrive%\TMP\stage.dat
REG IMPORT %systemdrive%\TMP\install.reg
if %stage% neq 0 goto flag010
net use Y: /delete
net use Z: /delete
shutdown -r -f -t 5
exit

:flag010
if %stage% gtr 1 goto flag020
echo 2 > %systemdrive%\TMP\stage.dat

$SNIPPET('my/winzip')
$SNIPPET('my/winrar')
$SNIPPET('my/win_install_chrome')
$SNIPPET('my/win_install_ffox')
$SNIPPET('my/win_install_adacr')
#if $distro_name in ( 'WinXp_EN-i386', 'Win2k3-Server_EN-x64' )
$SNIPPET('my/win_install_office_2007')
#else if $distro_name in (  'Win7_EN-x64', 'Win8_EN-x64' )
$SNIPPET('my/win_install_office_2010')

< .. >

Title Cleaning Temp files
DEL "%systemroot%\*.bmp" >nul 2>&1
DEL "%systemroot%\Web\Wallpaper\*.jpg" >nul 2>&1
DEL "%systemroot%\system32\dllcache\*.scr" >nul 2>&1
DEL "%systemroot%\system32\*.scr" >nul 2>&1
DEL "%AllUsersProfile%\Start Menu\Windows Update.lnk" >nul 2>&1
DEL "%AllUsersProfile%\Start Menu\Set Program Access and Defaults.lnk" >nul 2>&1
DEL "%AllUsersProfile%\Start Menu\Windows Catalog.lnk" >nul 2>&1
DEL "%systemdrive%\Microsoft Office*.txt" >nul 2>&1
net user aspnet /delete >nul 2>&1
REM %systemdrive%\TMP\sleep.exe 60
net use Y: /delete
net use Z: /delete

shutdown -r -f -t 30
RD /S /Q %systemdrive%\DRIVERS\ >nul 2>&1
if not defined stage DEL /F /Q %systemdrive%\post_install.cmd
DEL /F /S /Q %systemdrive%\TMP\*.*
exit
```

- Add Windows to the network installation menu in the `/etc/cobbler/boot_loader_conf/pxedefault.template` file:

```
menu begin Windows
MENU TITLE Windows
        label Win10_EN-x64
                MENU INDENT 5
                MENU LABEL Win10_EN-x64
                kernel /winos/Win10_EN-x64/win10a.0
        label  Win10-profile1
                MENU INDENT 5
                MENU LABEL  Win10-profile1
                kernel /winos/Win10_EN-x64/win10b.0
        label  Win10-profile2
                MENU INDENT 5
                MENU LABEL  Win10-profile2
                kernel /winos/Win10_EN-x64/win10c.0
        label Win2016-Server_EN-x64
                MENU INDENT 5
                MENU LABEL Win2016-Server_EN-x64
                kernel /winos/Win2016-Server_EN-x64/win6ra.0
< .. >
        label returntomain
                menu label Return to ^main menu.
                menu exit
menu end
```

Or create an iPXE boot menu

```
#!ipxe
< .. >
kernel http://<http_server>/winos/wimboot
initrd --name bootx64.efi   http://<http_server>/winos/Win10_EN-x64/EFI/Boot/
↪bootx64.efi bootx64.efi
initrd --name bcd           http://<http_server>/winos/Win10_EN-x64/boot/1Ea
↪      bcd
initrd --name boot.sdi      http://<http_server>/winos/Win10_EN-x64/boot/boot.sdi
↪      boot.sdi
initrd --name winpe.wim     http://<http_server>/winos/Win10_EN-x64/boot/winpe.wim
↪      winpe.wim
boot
< .. >
```

6.3.3 Final steps

- Restart the services:

```
systemctl restart cobblerd
systemctl restart tftpd
systemctl restart smb
systemctl restart nmb
```

- add distros:

```
cobbler distro add -name=Win10_EN-x64 \
--kernel=/var/lib/tftpboot/winos/Win10_EN-x64/pxeboot.n12 \
--initrd=/var/lib/tftpboot/winos/boot/boot.sdi \
--boot-loader=pxelinux \
--arch=x86_64 --breed=windows -os-version=10 \
--kernel-options='post_install=/var/lib/tftpboot/winos/Win10_EN-x64/sources/$OEM$/
↪$1/post_install.cmd'
```

- and profiles:

```
cobbler profile add --name=Win10_EN-x64 --distro=Win10_EN-x64 --autoinstall=win.ks␣
↪\
--kernel-options='pxeboot=win10a.0, bootmgr=boot1ea.exe, bcd=1Ea,winpe=winpe.wim,␣
↪sif=autounattended.xml'

cobbler profile add --name=Win10-profile1 --parent=Win10_EN-x64 \
--kernel-options='pxeboot=win10b.0, bootmgr=boot1eb.exe, bcd=1Eb,winpe=winp1.wim,␣
↪sif=autounattende1.xml'

cobbler profile add --name=Win10-profile2 --parent=Win10_EN-x64 \
--kernel-options='pxeboot=win10c.0, bootmgr=boot1ec.exe, bcd=1Ec,winpe=winp2.wim,␣
↪sif=autounattende2.xml'
```

- cobbler sync
- Install Windows

6.4 Extending Cobbler

This section covers methods to extend the functionality of Cobbler through the use of *Triggers* and *Modules*, as well as through extensions to the Cheetah templating system.

6.4.1 Triggers

About

Cobbler triggers provide a way to tie user-defined actions to certain Cobbler commands – for instance, to provide additional logging, integration with apps like Puppet or cfengine, set up SSH keys, tieing in with a DNS server configuration script, or for some other purpose.

Cobbler Triggers should be Python modules written using the low-level Python API for maximum speed, but could also be simple executable shell scripts.

As a general rule, if you need access to Cobbler's object data from a trigger, you need to write the trigger as a module. Also never invoke Cobbler from a trigger, or use Cobbler XMLRPC from a trigger. Essentially, Cobbler triggers can be thought of as plugins into Cobbler, though they are not essentially plugins per se.

Trigger Names (for Old-Style Triggers)

Cobbler script-based triggers are scripts installed in the following locations, and must be made `chmod +x`.

- `/var/lib/cobbler/triggers/add/system/pre/*`
- `/var/lib/cobbler/triggers/add/system/post/*`
- `/var/lib/cobbler/triggers/add/profile/pre/*`
- `/var/lib/cobbler/triggers/add/profile/post/*`
- `/var/lib/cobbler/triggers/add/distro/pre/*`
- `/var/lib/cobbler/triggers/add/distro/post/*`
- `/var/lib/cobbler/triggers/add/repo/pre/*`
- `/var/lib/cobbler/triggers/add/repo/post/*`
- `/var/lib/cobbler/triggers/sync/pre/*`
- `/var/lib/cobbler/triggers/sync/post/*`
- `/var/lib/cobbler/triggers/install/pre/*`

- `/var/lib/cobbler/triggers/install/post/*`

And the same as the above replacing "add" with "remove".

Pre-triggers are capable of failing an operation if they return anything other than 0. They are to be thought of as "validation" filters. Post-triggers cannot fail an operation and are to be thought of notifications.

We may add additional types as time goes on.

Pure Python Triggers

As mentioned earlier, triggers can be written in pure Python, and many of these kinds of triggers ship with Cobbler as stock.

Look in your `site-packages/cobbler/modules` directory and cat "`install_post_report.py`" for an example trigger that sends email when a system finished installation.

Notice how the trigger has a register method with a path that matches the shell patterns above. That's how we find out the type of trigger.

You will see the path used in the trigger corresponds with the path where it would exist if it was a script – this is how we know what type of trigger the module is providing.

The Simplest Trigger Possible

1. Create `/var/lib/cobbler/triggers/add/system/post/test.sh`.
2. `chmod +x` the file.

```
#!/bin/bash
echo "Hi, my name is $1 and I'm a newly added system"
```

However that's not very interesting as all you get are the names passed across. For triggers to be the most powerful, they should take advantage of the Cobbler API – which means writing them as a Python module.

Performance Note

If you have a very large number of systems, using the Cobbler API from scripts with old style (non-Python modules, just scripts in `/var/lib/cobbler/triggers`) is a very very bad idea. The reason for this is that the Cobbler API brings the Cobbler engine up with it, and since it's a seperate process, you have to wait for that to load. If you invoke 3000 triggers editing 3000 objects, you can see where this would get slow pretty quickly. However, if you write a modular trigger (see above) this suffers no performance penalties – it's crazy fast and you experience no problems.

Permissions

The `/var/lib/cobbler/triggers` directory is only writeable by root (and are executed by Cobbler on a regular basis). For security reasons do not loosen these permissions.

Example trigger for resetting Cfengine keys

Here is an example where Cobbler and cfengine are running on two different machines and XMLRPC is used to communicate between the hosts.

Note that this uses the Cobbler API so it's somewhat inefficient – it should be converted to a Python module-based trigger. If it would be a pure Python modular trigger, it would fly.

On the Cobbler box: `/var/lib/cobbler/triggers/install/post/clientkeys.py`

```
#!/usr/bin/python

import socket
import xmlrpclib
import sys
from cobbler import api
cobbler_api = api.BootAPI()
systems = cobbler_api.systems()
box = systems.find(sys.argv[2])
server = xmlrpclib.ServerProxy("http://cfengine:9000")
server.update(box.get_ip_address())
```

On the cfengine box, we run a daemon that does the following (along with a few steps to update our `ssh_known_hosts`- file):

```
#!/usr/bin/python

import SimpleXMLRPCServer
import os


class Keys(object):
    def update(self, ip):
        try:
            os.unlink('/var/cfengine/ppkeys/root-%s.pub' % ip)
        except OSError:
            pass


keys = Keys()
server = SimpleXMLRPCServer.SimpleXMLRPCServer(("cfengine", 9000))
server.register_instance(keys)
server.serve_forever()
```

See Also

- Post by Ithiriel: Writing triggers

6.4.2 Modules

Certain Cobbler features can be user extended (in Python) by Cobbler users.

These features include storage of data (serialization), authorization, and authentication. Over time, this list of module types will grow to support more options. *Triggers* are basically modules.

See Also

- The Cobbler command line itself (it's implemented in Cobbler modules so it's easy to add new commands)

Python Files And modules.conf

To create a module, add a Python file in `/usr/lib/python$version/site-packages/cobbler/modules`. Then, in the appropriate part of `/etc/cobbler/modules.conf`, reference the name of your module so Cobbler knows that you want to activate the module.

(*Triggers* that are Python modules, as well as CLI Python modules don't need to be listed in this file, they are auto-loaded)

An example from the serializers is:

```
[serializers]
settings = serializer.file
```

The format of `/etc/cobbler/modules.conf` is that of Python's ConfigParser module.

A setup file consists of sections, lead by a "[section]" header, and followed by "name: value" entries with continuations and such in the style of RFC 822.

Each module, regardless of it's nature, must have the following function that returns the type of module (as a string) on an acceptable load (when the module can be loaded) or raises an exception otherwise.

The trivial case for a cli module is:

```
def register():
    return "cli"
```

Other than that, modules do not have a particular API signature – they are "Duck Typed" based on how they are employed. When starting a new module, look at other modules of the same type to see what functions they possess.

6.4.3 Cheetah Macros

Cobbler uses Cheetah for its templating system, it also wants to support other choices and may in the future support others.

It is possible to add new functions to the templating engine, much like snippets that provide the ability to do macro-based things in the template. If you are new to Cheetah, see the documentation at Cheetah User Guide and pay special attention to the `#def` directive.

To create new functions, add your Cheetah code to `/etc/cobbler/cheetah_macros`. This file will be sourced in all Cheetah templates automatically, making it possible to write custom functions and use them from this file.

You will need to restart `cobblerd` after changing the macros file.

6.5 Terraform Provider for Cobbler

First have a brief look at Introduction to Terraform.

Next check out the Cobbler Provider official documentation.

- On GitHub: https://github.com/cobbler/terraform-provider-cobbler
- Releases: https://github.com/cobbler/terraform-provider-cobbler/releases

6.5.1 Why Terraform for Cobbler

Note: This document is written with Cobbler 3.2 and higher in mind, so the examples used here can not be used for Cobbler 2.x and `terraform-provider-cobbler` version 1.1.0 (and older).

There are multiple ways to add new systems, profiles, distro's into Cobbler, eg. through the web-interface or using shell-scripts on the Cobbler-host itself.

One of the main advantages of using the Terraform Provider for Cobbler is speed: you do not have to login into the web-interface or SSH to the host itself and adapt shell-scripts. When Terraform is installed on a VM or your local computer, it adds new assets through the Cobbler API.

6.5.2 Configure Cobbler

Configure Cobbler to have **caching disabled**.

In file `/etc/cobbler/settings`, set `cache_enabled: 0`.

6.5.3 Install Terraform

Terraform comes as a single binary, written in Go. Download an OS-specific package to install on your local system via the Terraform downloads. Unpack the ZIP-file and move the binary-file into `/usr/local/bin`.

Make sure you're using at least **Terraform v0.14 or higher**. Check with `terraform version`:

```
$ terraform version
Terraform v0.14.5
```

Install terraform-provider-cobbler

Since Terraform version 0.13, you can use the Cobbler provider via the Terraform provider registry.

After setting up a Cobbler Terraform repository for the first time, run `terraform init` in the **basedir**, so the Cobbler provider gets installed automatically in `tf_cobbler/.terraform/providers`.

```
$ terraform init

Initializing the backend...

Initializing provider plugins...
- Reusing previous version of cobbler/cobbler from the dependency lock file
- Installing cobbler/cobbler v2.0.2...
- Installed cobbler/cobbler v2.0.2 (self-signed, key ID B2677721AC1E7A84)

Partner and community providers are signed by their developers.
If you'd like to know more about provider signing, you can read about it here:
https://www.terraform.io/docs/plugins/signing.html

Terraform has made some changes to the provider dependency selections recorded
in the .terraform.lock.hcl file. Review those changes and commit them to your
version control system if they represent changes you intended to make.

Terraform has been successfully initialized!

You may now begin working with Terraform. Try running "terraform plan" to see
any changes that are required for your infrastructure. All Terraform commands
should now work.

If you ever set or change modules or backend configuration for Terraform,
rerun this command to reinitialize your working directory. If you forget, other
commands will detect it and remind you to do so if necessary.
```

If you ever run into this error: `Error: Could not load plugin`, re-run `terraform init` in the **basedir** to reinstall / upgrade the Cobbler provider.

When you initialize a Terraform configuration for the first time with Terraform 0.14 or later, Terraform will generate a new `.terraform.lock.hcl` file in the current working directory. You should include the lock file in your version control repository to ensure that Terraform uses the same provider versions across your team and in ephemeral remote execution environments.

6.5.4 Repository setup & configurations

Create a git repository (for example `tf_cobbler`) and use a phased approach of software testing and deployment in the DTAP-style:

- **development** - holds development systems
- **test** - holds test systems
- **staging** - holds staging / acceptance systems
- **production** - holds production systems
- **profiles** - holds system profiles
- **templates** - holds kickstarts and preseed templates
- **snippets** - holds Cobbler snippets (written in Python Cheetah or Jinja2)
- **distros** - holds OS distributions

The directory-tree would look something like this:

```
├── .gitignore
├── .terraform
│   └── prioviders
├── .terraform.lock.hcl
├── README.md
├── templates
│   ├── main.tf
│   ├── debian10.seed
│   ├── debian10_VMware.seed
│   ├── ...
├── staging
│   ├── db-staging
│   ├── lb-staging
│   ├── web-staging
│   └── ...
├── development
├── production
│   ├── database
│   ├── load_balancer
│   ├── webserver
│   ├── ...
├── set_links.sh
├── snippets
│   ├── partitioning-VMware.file
│   ├── main.tf
│   ├── ...
├── test
│   └── web-test
│   ├── ...
├── distros
│   └── distro-debian10-x86_64.tf
├── profiles
│   └── profile-debian10-x86_64.tf
├── terraform.tfvars
├── variables.tf
└── versions.tf
```

Each host-subdirectory consists of a Terraform-file named `main.tf`, one **symlinked** directory `.terraform` and files **symlinked** from the root: `versions.tf`, `variables.tf`. `.terraform.lock.hcl` and `terraform.tfvars`:

```
tf_cobbler/production/webserver
.
├── .terraform -> ../../.terraform
├── .terraform.lock.hcl -> ../../.terraform.lock.hcl
├── main.tf
├── terraform.tfstate
├── terraform.tfstate.backup
├── terraform.tfvars -> ../../terraform.tfvars
├── variables.tf -> ../../variables.tf
└── versions.tf -> ../../versions.tf
```

The files `terraform.tfstate` and `terraform.tfstate.backup` are the state files once Terraform has run succesfully.

File `versions.tf`

The block in this file specifies the required provider version and required Terraform version for the configuration.

```
terraform {
  required_version = ">= 0.14"
  required_providers {
    cobbler = {
      source = "cobbler/cobbler"
      version = "~> 2.0.1"
    }
  }
}
```

Credentials

You must add the `cobbler_username`, `cobbler_password` and the `cobbler_url` to the Cobbler API into a new file named `terraform.tfvars` in the basedir of your repo.

File `terraform.tfvars`

```
cobbler_username = "cobbler"
cobbler_password = "<the Cobbler-password>"
cobbler_url      = "https://cobbler.example.com/cobbler_api"
```

Terraform automatically loads `.tfvars`-files to populate variables defined in `variables.tf`.

> **Warning:** When using a git repo, do not (force) push the file `terraform.tfvars`, since it contains login credentials!

File `variables.tf`

Tip: We recommend you always add variable descriptions. You never know who'll be using your code, and it'll make their (and your) life a lot easier if every variable has a clear description. Comments are fun too.

Excerpt from: James Turnbull, "The Terraform Book."

```
variable "cobbler_username" {
  description = "Cobbler admin user"
  default     = "some_user"
}

variable "cobbler_password" {
  description = "Password for the Cobbler admin"
  default     = "some_password"
}

variable "cobbler_url" {
  description = "Where to reach the Cobbler API"
  default     = "http://some_server/cobbler_api"
}

provider "cobbler" {
  username = var.cobbler_username
  password = var.cobbler_password
  url      = var.cobbler_url
}
```

Example configuration - system

This is the `main.tf` for system `webserver`, written in so called HCL (HashiCorp Configuration Language). It has been cleaned up with the terraform fmt command, to rewrite Terraform configuration files to a canonical format and style:

Important: Make sure there is only **ONE** gateway defined on **ONE** interface!

```
resource "cobbler_system" "webserver" {
  count            = "1"
  name             = "webserver"
  profile          = "debian10-x86_64"
  hostname         = "webserver.example.com"       # Use FQDN
  autoinstall      = "debian10_VMware.seed"
  # NOTE: Extra spaces at the end are there for a reason!
  # When reading these resource states, the terraform-provider-cobbler
  # parses these fields with an extra space. Adding an extra space in the
  # next 2 lines prevents Terraform from constantly changing the resource.
  kernel_options   = "netcfg/choose_interface=eth0 "
  autoinstall_meta = "fs=ext4 swap=4096 "
  status           = "production"
  netboot_enabled  = "1"

  # Backend interface ############################
  interface {
    name          = "ens18"
    mac_address   = "0C:C4:7A:E3:C3:12"
    ip_address    = "10.11.15.106"
    netmask       = "255.255.255.0"
    dhcp_tag      = "grqproduction"
    dns_name      = "webserver.example.org"
    static_routes = ["10.11.14.0/24:10.11.15.1"]
    static        = true
    management    = true
  }

  # Public interface #############################
```

(continues on next page)

(continued from previous page)

```
  interface {
    name        = "ens18.15"
    mac_address = "0C:C4:7A:E3:C3:12"
    ip_address  = "127.28.15.106"
    netmask     = "255.255.255.128"
    gateway     = "127.28.15.1"
    dns_name    = "webserver.example.com"
    static      = true
  }
}
```

Example configuration - snippet

This is the `main.tf` for a snippet:

```
resource "cobbler_snippet" "partitioning-VMware" {
  name = "partitioning-VMware"
  body = file("partitioning-VMware.file")
}
```

In the same folder a file named `partitioning-VMware.file` holds the actual snippet.

Example configuration - repo

```
resource "cobbler_repo" "debian10-x86_64" {
  name           = "debian10-x86_64"
  breed          = "apt"
  arch           = "x86_64"
  apt_components = ["main universe"]
  apt_dists      = ["buster buster-updates buster-security"]
  mirror         = "http://ftp.nl.debian.org/debian/"
}
```

Example configuration - distro

```
resource "cobbler_distro" "debian10-x86_64" {
  name            = "debian10-x86_64"
  breed           = "debian"
  os_version      = "buster"
  arch            = "x86_64"
  kernel          = "/var/www/cobbler/distro_mirror/debian10-x86_64/install.amd/
↪linux"
  initrd          = "/var/www/cobbler/distro_mirror/debian10-x86_64/install.amd/
↪initrd.gz"
}
```

Example configuration - profile

```
resource "cobbler_profile" "debian10-x86_64" {
  name               = "debian10-x86_64"
  distro             = "debian10-x86_64"
  autoinstall        = "debian10.seed"
  autoinstall_meta   = "release=10 swap=2048"
  kernel_options     = "fb=false ipv6.disable=1"
```

(continues on next page)

(continued from previous page)

```
  name_servers        = ["1.1.1.1", "8.8.8.8"]    # Should be a list
  name_servers_search = ["example.com"]
  repos               = ["debian10-x86_64"]
}
```

Example configuration - combined

It is also possible to combine multiple resources into one file. For example, this will combine an Ubuntu Bionic distro, a profile and a system:

```
resource "cobbler_distro" "foo" {
    name = "foo"
    breed = "ubuntu"
    os_version = "bionic"
    arch = "x86_64"
    boot_loader = "grub"
    kernel = "/var/www/cobbler/distro_mirror/Ubuntu-18.04/install/netboot/ubuntu-
↪installer/amd64/linux"
    initrd = "/var/www/cobbler/distro_mirror/Ubuntu-18.04/install/netboot/ubuntu-
↪installer/amd64/initrd.gz"
  }

  resource "cobbler_profile" "foo" {
    name = "foo"
    distro = "foo"
  }

  resource "cobbler_system" "foo" {
    name = "foo"
    profile = "foo"
    name_servers = ["8.8.8.8", "8.8.4.4"]
    comment = "I'm a system"
    interface {
      name = "ens18"
      mac_address = "aa:bb:cc:dd:ee:ff"
      static = true
      ip_address = "1.2.3.4"
      netmask = "255.255.255.0"
    }
    interface {
      name = "ens19"
      mac_address = "aa:bb:cc:dd:ee:fa"
      static = true
      ip_address = "1.2.3.5"
      netmask = "255.255.255.0"
    }
  }
```

File `set_links.sh`

The file `set_links.sh` is used to symlink to the default variables. We need these in every subdirectory.

```
#!/bin/sh

ln -s ../../variables.tf
ln -s ../../versions.tf
ln -s ../../.terraform
ln -s ../../terraform.tfvars
ln -s ../../.terraform.lock.hcl
```

Adding a new system

```
git pull --rebase <-- Refresh the repository

mkdir production/hostname
cd production/hostname

vi main.tf          <-- Add a-based configuration as described above.

../../set_links.sh  # This will create symlinks to .terraform, variables.tf and␣
↪terraform.tfvars

terraform fmt       <-- Rewrites the file "main.tf" to canonical format.

terraform validate  <-- Validates the .tf file (optional).

terraform plan      <-- Create the execution plan.

terraform apply     <-- Apply changes, eg. add this system to the (remote) Cobbler.
```

When `terraform apply` gives errors it is safe to run `rm terraform.tfstate*` in the "hostname" directory and run `terraform apply` again.

6.6 API

Cobbler also makes itself available as an XML-RPC API for use by higher level management software. Learn more at https://cobbler.github.io

6.7 Triggers

Triggers provide a way to integrate Cobbler with arbitrary 3rd party software without modifying Cobbler's code. When adding a distro, profile, system, or repo, all scripts in `/var/lib/cobbler/triggers/add` are executed for the particular object type. Each particular file must be executable and it is executed with the name of the item being added as a parameter. Deletions work similarly – delete triggers live in `/var/lib/cobbler/triggers/delete`. Order of execution is arbitrary, and Cobbler does not ship with any triggers by default. There are also other kinds of triggers – these are described on the Cobbler Wiki. For larger configurations, triggers should be written in Python – in which case they are installed differently. This is also documented on the Wiki.

6.8 Images

Cobbler can help with booting images physically and virtually, though the usage of these commands varies substantially by the type of image. Non-image based deployments are generally easier to work with and lead to more sustainable infrastructure. Some manual use of other commands beyond of what is typically required of Cobbler may be needed to prepare images for use with this feature.

6.9 Power Management

Cobbler contains a power management feature that allows the user to associate system records in Cobbler with the power management configuration attached to them. This can ease installation by making it easy to reassign systems to new operating systems and then reboot those systems.

6.10 Non-import (manual) workflow

The following example uses a local kernel and initrd file (already downloaded), and shows how profiles would be created using two different automatic installation files – one for a web server configuration and one for a database server. Then, a machine is assigned to each profile.

```
cobbler check
cobbler distro add --name=rhel4u3 --kernel=/dir1/vmlinuz --initrd=/dir1/initrd.img
cobbler distro add --name=fc5 --kernel=/dir2/vmlinuz --initrd=/dir2/initrd.img
cobbler profile add --name=fc5webservers --distro=fc5-i386 --autoinstall=/dir4/
↪kick.ks --kernel-options="something_to_make_my_gfx_card_work=42 some_other_
↪parameter=foo"
cobbler profile add --name=rhel4u3dbservers --distro=rhel4u3 --autoinstall=/dir5/
↪kick.ks
cobbler system add --name=AA:BB:CC:DD:EE:FF --profile=fc5-webservers
cobbler system add --name=AA:BB:CC:DD:EE:FE --profile=rhel4u3-dbservers
cobbler report
```

6.11 Repository Management

6.11.1 REPO MANAGEMENT

This has already been covered a good bit in the command reference section.

Yum repository management is an optional feature, and is not required to provision through Cobbler. However, if Cobbler is configured to mirror certain repositories, it can then be used to associate profiles with those repositories. Systems installed under those profiles will then be autoconfigured to use these repository mirrors in `/etc/yum.repos.d`, and if supported (Fedora Core 6 and later) these repositories can be leveraged even within Anaconda. This can be useful if (A) you have a large install base, (B) you want fast installation and upgrades for your systems, or (C) have some extra software not in a standard repository but want provisioned systems to know about that repository.

Make sure there is plenty of space in Cobbler's webdir, which defaults to `/var/www/cobbler`.

```
cobbler reposync [--only=ONLY] [--tries=N] [--no-fail]
```

Cobbler reposync is the command to use to update repos as configured with "cobbler repo add". Mirroring can take a long time, and usage of Cobbler reposync prior to usage is needed to ensure provisioned systems have the files they need to actually use the mirrored repositories. If you just add repos and never run "cobbler reposync", the repos will never be mirrored. This is probably a command you would want to put on a crontab, though the frequency of that crontab and where the output goes is left up to the systems administrator.

For those familiar with dnf's reposync, Cobbler's reposync is (in most uses) a wrapper around the dnf reposync command. Please use "cobbler reposync" to update Cobbler mirrors, as dnf's reposync does not perform all required steps. Also Cobbler adds support for rsync and SSH locations, where as dnf's reposync only supports what yum supports (http/ftp).

If you ever want to update a certain repository you can run:

```
cobbler reposync --only="reponame1" ...
```

When updating repos by name, a repo will be updated even if it is set to be not updated during a regular reposync operation (ex: `cobbler repo edit --name=reponame1 --keep-updated=False`).

Note that if a Cobbler import provides enough information to use the boot server as a yum mirror for core packages, Cobbler can set up automatic installation files to use the Cobbler server as a mirror instead of the outside world. If this feature is desirable, it can be turned on by setting `yum_post_install_mirror` to `True` in `/etc/cobbler/settings.yaml` (and running `cobbler sync`). You should not use this feature if machines are

provisioned on a different VLAN/network than production, or if you are provisioning laptops that will want to acquire updates on multiple networks.

The flags `--tries=N` (for example, `--tries=3`) and `--no-fail` should likely be used when putting reposync on a crontab. They ensure network glitches in one repo can be retried and also that a failure to synchronize one repo does not stop other repositories from being synchronized.

6.11.2 Importing trees

Cobbler can auto-add distributions and profiles from remote sources, whether this is a filesystem path or an rsync mirror. This can save a lot of time when setting up a new provisioning environment. Import is a feature that many users will want to take advantage of, and is very simple to use.

After an import is run, Cobbler will try to detect the distribution type and automatically assign automatic installation files. By default, it will provision the system by erasing the hard drive, setting up eth0 for DHCP, and using a default password of "cobbler". If this is undesirable, edit the automatic installation files in `/etc/cobbler` to do something else or change the automatic installation setting after Cobbler creates the profile.

Mirrored content is saved automatically in `/var/www/cobbler/distro_mirror`.

Example 1: `cobbler import --path=rsync://mirrorserver.example.com/path/ --name=fedora --arch=x86`

Example 2: `cobbler import --path=root@192.168.1.10:/stuff --name=bar`

Example 3: `cobbler import --path=/mnt/dvd --name=baz --arch=x86_64`

Example 4: `cobbler import --path=/path/to/stuff --name=glorp`

Example 5: `cobbler import --path=/path/where/filer/is/mounted --name=anyname --available-as=nfs://nfs.example.org:/where/mounted/`

Once imported, run a `cobbler list` or `cobbler report` to see what you've added.

By default, the rsync operations will exclude content of certain architectures, debug RPMs, and ISO images – to change what is excluded during an import, see `/etc/cobbler/rsync.exclude`.

Note that all of the import commands will mirror install tree content into `/var/www/cobbler` unless a network accessible location is given with `--available-as`. –available-as will be primarily used when importing distros stored on an external NAS box, or potentially on another partition on the same machine that is already accessible via `http://` or `ftp://`.

For import methods using rsync, additional flags can be passed to rsync with the option `--rsync-flags`.

Should you want to force the usage of a specific Cobbler automatic installation template for all profiles created by an import, you can feed the option `--autoinstall` to import, to bypass the built-in automatic installation file auto-detection.

6.11.3 Repository mirroring workflow

The following example shows how to set up a repo mirror for all enabled Cobbler host repositories and two additional repositories, and create a profile that will auto install those repository configurations on provisioned systems using that profile.

```
cobbler check
# set up your cobbler distros here.
cobbler autoadd
cobbler repo add --mirror=http://mirrors.kernel.org/fedora/core/updates/6/i386/ --
↪name=fc6i386updates
cobbler repo add --mirror=http://mirrors.kernel.org/fedora/extras/6/i386/ --
↪name=fc6i386extras
cobbler reposync
cobbler profile add --name=p1 --distro=existing_distro_name --autoinstall=/etc/
↪cobbler/kickstart_fc6.ks --repos="fc6i386updates fc6i386extras"
```

(continues on next page)

(continued from previous page)

6.11.4 Import Workflow

Import is a very useful command that makes starting out with Cobbler very quick and easy.

This example shows how to create a provisioning infrastructure from a distribution mirror or DVD ISO. Then a default PXE configuration is created, so that by default systems will PXE boot into a fully automated install process for that distribution.

You can use a network rsync mirror, a mounted DVD location, or a tree you have available via a network filesystem.

Import knows how to autodetect the architecture of what is being imported, though to make sure things are named correctly, it's always a good idea to specify `--arch`. For instance, if you import a distribution named "fedora8" from an ISO, and it's an x86_64 ISO, specify `--arch=x86_64` and the distro will be named "fedora8-x86_64" automatically, and the right architecture field will also be set on the distribution object. If you are batch importing an entire mirror (containing multiple distributions and arches), you don't have to do this, as Cobbler will set the names for things based on the paths it finds.

```
cobbler check
cobbler import --path=rsync://yourfavoritemirror.com/rhel/5/os/x86_64 --name=rhel5␣
↪--arch=x86_64
# OR
cobbler import --path=/mnt/dvd --name=rhel5 --arch=x86_64
# OR (using an external NAS box without mirroring)
cobbler import --path=/path/where/filer/is/mounted --name=anyname --available-
↪as=nfs://nfs.example.org:/where/mounted/
# wait for mirror to rsync...
cobbler report
cobbler system add --name=default --profile=name_of_a_profile1
cobbler system add --name=AA:BB:CC:DD:EE:FF --profile=name_of_a_profile2
cobbler sync
```

6.12 Virtualization

For Virt, be sure the distro uses the correct kernel (if paravirt) and follow similar steps as above, adding additional parameters as desired:

```
cobbler distro add --name=fc7virt [options...]
```

Specify reasonable values for the Virt image size (in GB) and RAM requirements (in MB):

```
cobbler profile add --name=virtwebservers --distro=fc7virt --autoinstall=path --
↪virt-file-size=10 --virt-ram=512 [...]
```

Define systems if desired. Koan can also provision based on the profile name.

```
cobbler system add --name=AA:BB:CC:DD:EE:FE --profile=virtwebservers [...]
```

If you have just installed Cobbler, be sure that the *cobblerd* service is running and that port 25151 is unblocked.

See the manpage for Koan for the client side steps.

6.13 Autoinstallation

6.13.1 Automatic installation templating

The `--autoinstall_meta` options above require more explanation.

If and only if `--autoinstall` options reference filesystem URLs, `--autoinstall-meta` allows for templating of the automatic installation files to achieve advanced functions. If the `--autoinstall-meta` option for a profile read `--autoinstall-meta="foo=7 bar=llama"`, anywhere in the automatic installation file where the string `$bar` appeared would be replaced with the string "llama".

To apply these changes, `cobbler sync` must be run to generate custom automatic installation files for each profile/system.

For NFS and HTTP automatic installation file URLs, the `--autoinstall_meta` options will have no effect. This is a good reason to let Cobbler manage your automatic installation files, though the URL functionality is provided for integration with legacy infrastructure, possibly including web apps that already generate automatic installation files.

Templated automatic files are processed by the templating program/package Cheetah, so anything you can do in a Cheetah template can be done to an automatic installation template. Learn more at https://cheetahtemplate.org/users_guide/intro.html

When working with Cheetah, be sure to escape any shell macros that look like `$(this)` with something like `\$(this)` or errors may show up during the sync process.

The Cobbler Wiki also contains numerous Cheetah examples that should prove useful in using this feature.

Also useful is the following repository: https://github.com/FlossWare/cobbler

6.13.2 Automatic installation snippets

Anywhere a automatic installation template mentions `SNIPPET::snippet_name`, the file named `/var/lib/cobbler/snippets/snippet_name` (if present) will be included automatically in the automatic installation template. This serves as a way to recycle frequently used automatic installation snippets without duplication. Snippets can contain templating variables, and the variables will be evaluated according to the profile and/or system as one would expect.

Snippets can also be overridden for specific profile names or system names. This is described on the Cobbler Wiki.

6.13.3 Kickstart validation

To check for potential errors in kickstarts, prior to installation, use `cobbler validateks`. This function will check all profile and system kickstarts for detectable errors. Since pykickstart is not future-Anaconda-version aware, there may be some false positives. It should be noted that `cobbler validateks` runs on the rendered kickstart output, not kickstart templates themselves.

6.14 Network Topics

6.14.1 PXE Menus

Cobbler will automatically generate PXE menus for all profiles it has defined. Running `cobbler sync` is required to generate and update these menus.

To access the menus, type `menu` at the `boot:` prompt while a system is PXE booting. If nothing is typed, the network boot will default to a local boot. If "menu" is typed, the user can then choose and provision any Cobbler profile the system knows about.

If the association between a system (MAC address) and a profile is already known, it may be more useful to just use `system add` commands and declare that relationship in Cobbler; however many use cases will prefer having a PXE system, especially when provisioning is done at the same time as installing new physical machines.

If this behavior is not desired, run `cobbler system add --name=default --profile=plugh` to default all PXE booting machines to get a new copy of the profile `plugh`. To go back to the menu system, run `cobbler system remove --name=default` and then `cobbler sync` to regenerate the menus.

When using PXE menu deployment exclusively, it is not necessary to make Cobbler system records, although the two can easily be mixed.

Additionally, note that all files generated for the PXE menu configurations are templatable, so if you wish to change the color scheme or equivalent, see the files in `/etc/cobbler`.

6.14.2 Default PXE Boot behavior

What happens when PXE booting a system when Cobbler has no record of the system being booted?

By default, Cobbler will configure PXE to boot to the contents of `/etc/cobbler/default.pxe`, which (if unmodified) will just fall through to the local boot process. Administrators can modify this file if they like to change that behavior.

An easy way to specify a default Cobbler profile to PXE boot is to create a system named `default`. This will cause `/etc/cobbler/default.pxe` to be ignored. To restore the previous behavior do a `cobbler system remove` on the `default` system.

```
cobbler system add --name=default --profile=boot_this
cobbler system remove --name=default
```

As mentioned in earlier sections, it is also possible to control the default behavior for a specific network:

```
cobbler system add --name=network1 --ip-address=192.168.0.0/24 --profile=boot_this
```

6.14.3 PXE boot loop prevention

If you have your machines set to PXE first in the boot order (ahead of hard drives), change the `pxe_just_once` flag in `/etc/cobbler/settings.yaml` to 1. This will set the machines to not PXE on successive boots once they complete one install. To re-enable PXE for a specific system, run the following command:

```
cobbler system edit --name=name --netboot-enabled=1
```

6.14.4 Automatic installation tracking

Cobbler knows how to keep track of the status of automatic installation of machines.

```
cobbler status
```

Using the status command will show when Cobbler thinks a machine started automatic installation and when it finished, provided the proper snippets are found in the automatic installation template. This is a good way to track machines that may have gone interactive (or stalled/crashed) during automatic installation.

6.15 Boot CD

Cobbler can build all of it's profiles into a bootable CD image using the `cobbler buildiso` command. This allows for PXE-menu like bring up of bare metal in environments where PXE is not possible. Another more advanced method is described in the Koan manpage, though this method is easier and sufficient for most applications.

6.15.1 DHCP Management

Cobbler can optionally help you manage DHCP server. This feature is off by default.

Choose either `management = isc_and_bind` in `/etc/cobbler/dhcp.template` or `management = "dnsmasq"` in `/etc/cobbler/modules.conf`. Then set `manage_dhcp=1` in `/etc/cobbler/settings.yaml`.

This allows DHCP to be managed via "cobbler system add" commands, when you specify the mac address and IP address for systems you add into Cobbler.

Depending on your choice, Cobbler will use `/etc/cobbler/dhcpd.template` or `/etc/cobbler/dnsmasq.template` as a starting point. This file must be user edited for the user's particular networking environment. Read the file and understand how the particular app (ISC dhcpd or dnsmasq) work before proceeding.

If you already have DHCP configuration data that you would like to preserve (say DHCP was manually configured earlier), insert the relevant portions of it into the template file, as running `cobbler sync` will overwrite your previous configuration.

By default, the DHCP configuration file will be updated each time `cobbler sync` is run, and not until then, so it is important to remember to use `cobbler sync` when using this feature.

If omapi_enabled is set to 1 in `/etc/cobbler/settings.yaml`, the need to sync when adding new system records can be eliminated. However, the OMAPI feature is experimental and is not recommended for most users.

6.15.2 DNS configuration management

Cobbler can optionally manage DNS configuration using BIND and dnsmasq.

Choose either `management = isc_and_bind` or `management = dnsmasq` in `/etc/cobbler/modules.conf` and then enable `manage_dns` in `/etc/cobbler/settings.yaml`.

This feature is off by default. If using BIND, you must define the zones to be managed with the options `manage_forward_zones` and `manage_reverse_zones`. (See the Wiki for more information on this).

If using BIND, Cobbler will use `/etc/cobbler/named.template` and `/etc/cobbler/zone.template` as a starting point for the `named.conf` and individual zone files, respectively. You may drop zone-specific template files in `/etc/cobbler/zone_templates/name-of-zone` which will override the default. These files must be user edited for the user's particular networking environment. Read the file and understand how BIND works before proceeding.

If using dnsmasq, the template is `/etc/cobbler/dnsmasq.template`. Read this file and understand how dnsmasq works before proceeding.

All managed files (whether zone files and `named.conf` for BIND, or `dnsmasq.conf` for dnsmasq) will be updated each time `cobbler sync` is run, and not until then, so it is important to remember to use `cobbler sync` when using this feature.

6.16 Containerization

We have a test-image which you can find in the Cobbler repository and an old image made by the community: https://github.com/osism/docker-cobbler

CHAPTER 7

Developer Guide

7.1 Patch process

You'd like to contribute features or fixes to Cobbler? Great! We'd love to have them.

It is highly recommended that you have a GitHub account if you would like to contribute code. Create an account, log in, and then go to https://github.com/cobbler/cobbler to "fork" the project.

Create a new branch named after the feature you are working on. Do the work on your local machine, please make sure your work passes Cobbler's coding standards by using `make qa`. Only then push to your personal GitHub branch (e.g. https://github.com/yourname/cobbler).

Then use the "submit pull request" feature of GitHub to request that the official repo pull in your changes. Be sure to include a full description of what your change does in the comments, including what you have tested (and other things that you may have not been able to test well and need help with).

If the patch needs more work, we'll let you know in the comments.

Do not mix work on different features in different pull requests/branches if at all possible as this makes it difficult to take only some of the work at one time, and to quickly slurp in some changes why others get hammered out.

Once we merge in your pull request, you can remove the branch from your repo if you like. The AUTHORS file is created automatically when we release.

7.2 Setup

The preferred development platform is the latest openSUSE Leap or Tumbleweed. You'll also have to disable SELinux to get Cobbler up and running.

For CentOS you will need the EPEL repository: `http://download.fedoraproject.org/pub/epel/7/x86_64/repoview/epel-release.html`

Install development dependencies:

```
# yum install git make openssl python-sphinx python36-coverage python36-devel␣
↪python36-distro python36-future python36-pyflakes python36-pycodestyle python36-
↪setuptools rpm-build
```

Install runtime dependencies:

```
# yum install httpd mod_wsgi python36-PyYAML python36-netaddr python36-simplejson
# pip3 install Cheetah3
```

Initially, to run Cobbler without using packages:

```
# git clone https://github.com/<your username>/cobbler.git
# cd cobbler
# make install
```

For each successive run, do not run make install again. To avoid blowing away your configuration, run:

```
# make webtest
```

This will install Cobbler and restart apache/cobblerd, but move your configuration files and settings aside and restore them, rather than blindly overwriting them.

You can now run Cobbler commands and access the web interface.

7.3 Tests

We are using pytest and are executing our tests inside Docker because of the high overhead (TFTP, Apache 2, . . .), this also has the advantage that we can easily debug the tests locally.

7.4 Build RPMs/DEBs using Docker

1. Make sure docker and docker-compose are installed
2. Use docker-compose to build rpms for the various distros
3. RPMs are in rpm-build/

7.5 Branches

Cobbler has a development branch called “master” (where the action is), and branches for all releases that are in maintenance mode. All work on new features should be done against the master branch. If you want to address bugs then please target the latest release branch, the maintainers will then cherry-pick those changes into the master branch.

```
# git branch -r
# git checkout <branch>
# git checkout -b <new branch name>
```

7.6 Standards

We’re not overly picky, but please follow the python PEP8 standards we want to adhere to (see Makefile).

- Always use under_scores, not camelCase.
- Always four (4) spaces, not tabs.
- Avoid one line if statements.
- Validate your code by using `make qa`.
- Keep things simple, keep in mind that this is a tool for sysadmins and not python developers.

- Use modules that are easily available (e.g. EPEL) but preferably in the base OS, otherwise they have to be packaged with the app, which usually runs afoul of distribution packaging guidelines.
- Cobbler is since the 3.x.x release Python3 only.
- Koan has no new release currently but starting with the next we will also only support Python3.
- Older releases will of course stay with Python2.

You're also welcome to hang out in #cobbler and #cobbler-devel on irc.freenode.net, as there are folks around to answer questions, etc. But it isn't that active anymore please drop also in our Cobbler Gitter channel there we will probably answer faster.

7.7 Contributing to the website

The GitHub-based git repository for the https://cobbler.github.io website itself is at https://github.com/cobbler/cobbler.github.io.

If you want to contribute changes to the website, you will need Jekyll (http://jekyllrb.com).

You will probably want to:

- edit the files as markdown
- run the docker container
- check if your changes didn't break anything

7.8 Debugging

If you need to debug a remote process, rpdb provides some very nice capabilities beyond the standard python debugger, just insert a `import rpdb; rpdb.set_trace()` on the desired line run cobbler and then do a `nc 127.0.0.1 4444`.

CHAPTER 8

cobbler package

8.1 Subpackages

8.1.1 cobbler.actions package

Submodules

cobbler.actions.acl module

cobbler.actions.buildiso module

cobbler.actions.check module

cobbler.actions.dlcontent module

Downloads bootloader content for all arches for when the user doesn't want to supply their own.

class cobbler.actions.dlcontent.**ContentDownloader**(*collection_mgr*, *logger=None*)

Bases: object

run(*force: bool = False*)

Download bootloader content for all of the latest bootloaders, since the user has chosen to not supply their own. You may ask "why not get this from yum", we also want this to be able to work on Debian and further do not want folks to have to install a cross compiler. For those that don't like this approach they can still source their cross-arch bootloader content manually.

Parameters **force** – If the target path should be overwritten, even if there are already files present.

cobbler.actions.hardlink module

cobbler.actions.litesync module

cobbler.actions.log module

Copyright 2009, Red Hat, Inc and Others Bill Peck <bpeck@redhat.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.log.**LogTool**(*collection_mgr*, *system*, *api*, *logger=None*)

Bases: object

Helpers for dealing with System logs, anamon, etc..

clear()

Clears the system logs

cobbler.actions.replicate module

cobbler.actions.report module

cobbler.actions.reposync module

cobbler.actions.status module

Reports on automatic installation activity by examining the logs in /var/log/cobbler.

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.actions.status.**CobblerStatusReport**(*collection_mgr*, *mode*, *logger=None*)

Bases: object

catalog(*profile_or_system: str*, *name: str*, *ip*, *start_or_stop: str*, *ts: float*)

Add a system to cobbler status.

Parameters

- **profile_or_system** – This can be system or profile.

- **name** – The name of the object.
- **ip** – The ip of the system to watch.
- **start_or_stop** – This parameter may be `start` or `stop`
- **ts** – Don't know what this does.

get_printable_results()
Convert the status of Cobbler from a machine readable form to human readable.

Returns A nice formatted representation of the results of `cobbler status`.

process_results()
Look through all systems which were collected and update the status.

Returns Return `ip_data` of the object.

run()
Calculate and print a automatic installation status report.

scan_logfiles()
Scan the install log-files - starting with the oldest file.

cobbler.actions.sync module

Module contents

8.1.2 cobbler.cobbler_collections package

Submodules

cobbler.cobbler_collections.collection module

cobbler.cobbler_collections.distros module

cobbler.cobbler_collections.files module

cobbler.cobbler_collections.images module

cobbler.cobbler_collections.manager module

cobbler.cobbler_collections.mgmtclasses module

cobbler.cobbler_collections.packages module

cobbler.cobbler_collections.profiles module

cobbler.cobbler_collections.repos module

cobbler.cobbler_collections.systems module

Module contents

8.1.3 cobbler.items package

Submodules

cobbler.items.distro module

cobbler.items.file module

cobbler.items.image module

cobbler.items.item module

cobbler.items.mgmtclass module

cobbler.items.package module

cobbler.items.profile module

cobbler.items.repo module

cobbler.items.system module

Module contents

8.1.4 cobbler.modules package

Subpackages

cobbler.modules.authentication package

Submodules

cobbler.modules.authentication.configfile module

cobbler.modules.authentication.denyall module

Authentication module that denies everything. Used to disable the WebUI by default.

`cobbler.modules.authentication.denyall.`**`authenticate`**`(`*api_handle*, *username*, *password*`)` → bool

Validate a username/password combo, returning True/False

Thanks to http://trac.edgewall.org/ticket/845 for supplying the algorithm info.

`cobbler.modules.authentication.denyall.`**`register`**`()` → str

The mandatory Cobbler module registration hook.

cobbler.modules.authentication.ldap module

Authentication module that uses ldap Settings in /etc/cobbler/authn_ldap.conf Choice of authentication module is in /etc/cobbler/modules.conf

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.authentication.ldap.**authenticate**(*api_handle*, *username*, *password*) → bool

Validate an LDAP bind, returning whether the authentication was successful or not.

Parameters

- **api_handle** – The api instance to resolve settings.
- **username** – The username to authenticate.
- **password** – The password to authenticate.

Returns True if the ldap server authentication was a success, otherwise false.

cobbler.modules.authentication.ldap.**register**() → str

The mandatory Cobbler module registration hook.

Returns Always "authn"

Return type str

cobbler.modules.authentication.pam module

Authentication module that uses /etc/cobbler/auth.conf Choice of authentication module is in /etc/cobbler/modules.conf

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

PAM python code based on the pam_python code created by Chris AtLee: http://atlee.ca/software/pam/

#———————————————— pam_python (c) 2007 Chris AtLee <chris@atlee.ca> Licensed under the MIT license: http://www.opensource.org/licenses/mit-license.php

PAM module for python

Provides an authenticate function that will allow the caller to authenticate a user against the Pluggable Authentication Modules (PAM) on the system.

Implemented using ctypes, so no compilation is necessary.

class cobbler.modules.authentication.pam.**PamConv**

Bases: _ctypes.Structure

wrapper class for pam_conv structure

appdata_ptr

Structure/Union member

conv
Structure/Union member

class cobbler.modules.authentication.pam.**PamHandle**
Bases: _ctypes.Structure

wrapper class for pam_handle_t

handle
Structure/Union member

class cobbler.modules.authentication.pam.**PamMessage**
Bases: _ctypes.Structure

wrapper class for pam_message structure

msg
Structure/Union member

msg_style
Structure/Union member

class cobbler.modules.authentication.pam.**PamResponse**
Bases: _ctypes.Structure

wrapper class for pam_response structure

resp
Structure/Union member

resp_retcode
Structure/Union member

cobbler.modules.authentication.pam.**authenticate**(*api_handle*, *username: str*, *password: str*) → bool

Parameters **api_handle** – Used for resolving the the pam service name and getting the Logger.

:param username:The username to log in with. :param password: The password to log in with. :returns: True if the given username and password authenticate for the given service. Otherwise False

cobbler.modules.authentication.pam.**register**() → str
The mandatory Cobbler module registration hook.

cobbler.modules.authentication.passthru module

cobbler.modules.authentication.spacewalk module

cobbler.modules.authentication.testing module

Authentication module that denies everything. Unsafe demo. Allows anyone in with testing/testing.

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.authentication.testing.**authenticate**(*api_handle*, *username: str*, *password: str*) → bool

Validate a username/password combo, returning True/False

Thanks to http://trac.edgewall.org/ticket/845 for supplying the algorithm info.

Parameters

- **api_handle** – This parameter is not used currently.
- **username** – The username which should be checked.
- **password** – The password which should be checked.

Returns True if username is "testing" and password is "testing". Otherwise False.

cobbler.modules.authentication.testing.**register**() → str

The mandatory Cobbler module registration hook.

Returns Always "authn"

Return type str

Module contents

cobbler.modules.authorization package

Submodules

cobbler.modules.authorization.allowall module

Authorization module that allows everything, which is the default for new Cobbler installs.

Copyright 2007-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.authorization.allowall.**authorize**(*api_handle*, *user*, *resource*, *arg1=None*, *arg2=None*) → bool

Validate a user against a resource. NOTE: acls are not enforced as there is no group support in this module

Parameters

- **api_handle** – This parameter is not used currently.
- **user** – This parameter is not used currently.
- **resource** – This parameter is not used currently.
- **arg1** – This parameter is not used currently.
- **arg2** – This parameter is not used currently.

Returns Always True

Return type bool

cobbler.modules.authorization.allowall.**register**() → str
The mandatory Cobbler module registration hook.

Returns Always "authz"

Return type str

cobbler.modules.authorization.configfile module

Authorization module that allow users listed in /etc/cobbler/users.conf to be permitted to access resources. For instance, when using authz_ldap, you want to use authn_configfile, not authz_allowall, which will most likely NOT do what you want.

This software may be freely redistributed under the terms of the GNU general public license.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA.

cobbler.modules.authorization.configfile.**authorize**(*api_handle*, *user*, *resource*, *arg1=None*, *arg2=None*) → int
Validate a user against a resource. All users in the file are permitted by this module.

Parameters

- **api_handle** – This parameter is not used currently.
- **user** – The user to authorize.
- **resource** – This parameter is not used currently.
- **arg1** – This parameter is not used currently.
- **arg2** – This parameter is not used currently.

Returns "0" if no authorized, "1" if authorized.

cobbler.modules.authorization.configfile.**register**() → str
The mandatory Cobbler module registration hook.

Returns Always "authz".

cobbler.modules.authorization.ownership module

Authorization module that allow users listed in /etc/cobbler/users.conf to be permitted to access resources, with the further restriction that Cobbler objects can be edited to only allow certain users/groups to access those specific objects.

Copyright 2008-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.authorization.ownership.**authorize**(*api_handle*, *user*, *resource*, *arg1=None*, *arg2=None*) → Union[bool, int]
Validate a user against a resource. All users in the file are permitted by this module.

Parameters

- **api_handle** – The api to resolve required information.
- **user** – The user to authorize to the resource.
- **resource** – The resource the user is asking for access. This is something abstract like a remove operation.
- **arg1** – This is normally the name of the specific object in question.
- **arg2** – This parameter is pointless currently. Reserved for future code.

Returns `True` or `1` if okay, otherwise `False`.

`cobbler.modules.authorization.ownership.`**register**`()` → str
The mandatory Cobbler module registration hook.

Returns Always "authz"

Module contents

cobbler.modules.installation package

Submodules

cobbler.modules.installation.post_log module

(C) 2008-2009, Red Hat Inc. Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

`cobbler.modules.installation.post_log.`**register**`()` → str
The mandatory Cobbler module registration hook.

`cobbler.modules.installation.post_log.`**run**(*api*, *args*, *logger*) → int

Parameters

- **api** – This parameter is unused currently.
- **args** – An array of three elements. Type (system/profile), name and ip. If no ip is present use a `?`.
- **logger** – This parameter is unused currently.

Returns Always 0

cobbler.modules.installation.post_power module

class `cobbler.modules.installation.post_power.`**reboot**(*api*, *target*)
Bases: `threading.Thread`

run()
Method representing the thread's activity.

You may override this method in a subclass. The standard run() method invokes the callable object passed to the object's constructor as the target argument, if any, with sequential and keyword arguments taken from the args and kwargs arguments, respectively.

cobbler.modules.installation.post_power.**register**() → str
The mandatory Cobbler module registration hook.

cobbler.modules.installation.post_power.**run**(*api*, *args*, *logger*) → int
Obligatory trigger hook.

Parameters

- **api** – The api to resolve information with.
- **args** – This is an array containing two objects. 0: String with the content "target" or "profile". 1: The name of target or profile
- **logger** – Unused parameter for this hook.

Returns 0 on success.

cobbler.modules.installation.post_puppet module

cobbler.modules.installation.post_report module

cobbler.modules.installation.pre_clear_anamon_logs module

(C) 2008-2009, Red Hat Inc. James Laska <jlaska@redhat.com> Bill Peck <bpeck@redhat.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.modules.installation.pre_clear_anamon_logs.**register**() → str
This pure python trigger acts as if it were a legacy shell-trigger, but is much faster. The return of this method indicates the trigger type.

Returns Always: "/var/lib/cobbler/triggers/install/pre/*"

cobbler.modules.installation.pre_clear_anamon_logs.**run**(*api*, *args*, *logger*) → int

The list of args should have one element:

- 1: the name of the system or profile

Parameters

- **api** – The api to resolve metadata with.
- **args** – This should be a list as described above.
- **logger** – This parameter is unused currently.

Returns "0" on success.

cobbler.modules.installation.pre_log module

cobbler.modules.installation.pre_log.**register**() → str
This pure python trigger acts as if it were a legacy shell-trigger, but is much faster. The return of this method indicates the trigger type.

Returns Always: "/var/lib/cobbler/triggers/install/pre/*"

cobbler.modules.installation.pre_log.**run**(*api*, *args: list*, *logger*) → int
The method runs the trigger, meaning this logs that an installation has started.

The list of args should have three elements:

- 0: system or profile
- 1: the name of the system or profile
- 2: the ip or a "?"

Parameters

- **api** – This parameter is currently unused.
- **args** – Already described above.
- **logger** – This parameter is currently unused.

Returns A "0" on success.

cobbler.modules.installation.pre_puppet module

Module contents

cobbler.modules.managers package

Submodules

cobbler.modules.managers.bind module

cobbler.modules.managers.dnsmasq module

cobbler.modules.managers.genders module

cobbler.modules.managers.import_signatures module

cobbler.modules.managers.in_tftpd module

cobbler.modules.managers.isc module

cobbler.modules.managers.ndjbdns module

cobbler.modules.managers.tftpd_py module

Module contents

cobbler.modules.serializers package

Submodules

cobbler.modules.serializers.file module

cobbler.modules.serializers.mongodb module

Module contents

Submodules

cobbler.modules.nsupdate_add_system_post module

cobbler.modules.nsupdate_add_system_post.**nslog**(*msg*)
Log a message to the logger.

Parameters **msg** – The message to log.

cobbler.modules.nsupdate_add_system_post.**register**() → str
This method is the obligatory Cobbler registration hook.

Returns The trigger name or an empty string.

cobbler.modules.nsupdate_add_system_post.**run**(*api*, *args*, *logger*)
This method executes the trigger, meaning in this case that it updates the dns configuration.

Parameters

- **api** – The api to read metadata from.
- **args** – Metadata to log.
- **logger** – The logger to audit the action with.

Returns "0" on success or a skipped task. If the task failed or problems occurred then an exception is raised.

cobbler.modules.nsupdate_delete_system_pre module

cobbler.modules.nsupdate_delete_system_pre.**nslog**(*msg*)
Log a message to the logger.

Parameters **msg** – The message to log.

cobbler.modules.nsupdate_delete_system_pre.**register**() → str
This method is the obligatory Cobbler registration hook.

Returns The trigger name or an empty string.

Return type str

cobbler.modules.nsupdate_delete_system_pre.**run**(*api*, *args*, *logger*)
This method executes the trigger, meaning in this case that it updates the dns configuration.

Parameters

- **api** – The api to read metadata from.
- **args** – Metadata to log.
- **logger** – The logger to audit the action with.

Returns "0" on success or a skipped task. If the task failed or problems occurred then an exception is raised.

cobbler.modules.scm_track module

cobbler.modules.sync_post_restart_services module

Module contents

8.1.5 cobbler.web package

Subpackages

cobbler.web.templatetags package

Submodules

cobbler.web.templatetags.site module

class cobbler.web.templatetags.site.**And**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**BaseCalc**(*var1*, *var2=None*, *negate=False*)
Bases: object

calculate(*var1*, *var2*)

resolve(*context*)

resolve_vars(*context*)

class cobbler.web.templatetags.site.**Equals**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**Greater**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**GreaterOrEqual**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**IfParser**(*tokens*)
Bases: object

at_end()

create_var(*value*)

error_class
alias of builtins.ValueError

get_token()

get_var()

parse()

tokens

class cobbler.web.templatetags.site.**In**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**Or**(*var1*, *var2=None*, *negate=False*)
Bases: *cobbler.web.templatetags.site.BaseCalc*

calculate(*var1*, *var2*)

class cobbler.web.templatetags.site.**SmartIfNode**(*var*, *nodelist_true*, *nodelist_false=None*)
Bases: django.template.base.Node

get_nodes_by_type(*nodetype*)
Return a list of all nodes (within this node and its nodelist) of the given type

render(*context*)
Return the node rendered as a string.

class cobbler.web.templatetags.site.**SmartIfTests**(*methodName='runTest'*)
Bases: unittest.case.TestCase

assertCalc(*calc*, *context=None*)
Test a calculation is True, also checking the inverse "negate" case.

assertCalcFalse(*calc*, *context=None*)
Test a calculation is False, also checking the inverse "negate" case.

setUp()
Hook method for setting up the test fixture before exercising it.

test_and()

test_boolean()

test_equals()

test_greater()

test_greater_or_equal()

test_in()

test_or()

test_parse_bits()

class cobbler.web.templatetags.site.**TemplateIfParser**(*parser*, **args*, ***kwargs*)
Bases: *cobbler.web.templatetags.site.IfParser*

create_var(*value*)

error_class
alias of django.template.exceptions.TemplateSyntaxError

class cobbler.web.templatetags.site.**TestVar**(*value*)
Bases: object

A basic self-resolvable object similar to a Django template variable. Used to assist with tests.

resolve(*context*)

cobbler.web.templatetags.site.**ifinlist**(*parser*, *token*)
A smarter {% if %} tag for django templates.

While retaining current Django functionality, it also handles equality, greater than and less than operators. Some common case examples:

```
{% if articles|length >= 5 %}...{% endif %}
{% if "ifnotequal tag" != "beautiful" %}...{% endif %}
```

Arguments and operators _must_ have a space between them, so `{% if 1>2 %}` is not a valid smart if tag.

All supported operators are: `or`, `and`, `in`, = (or ==), `!=`, >, >=, < and <=.

cobbler.web.templatetags.site.**listsort**(*value*)

cobbler.web.templatetags.site.**register = <django.template.library.Library object>**

A smarter {% if %} tag for django templates.

While retaining current Django functionality, it also handles equality, greater than and less than operators. Some common case examples:

```
{% if articles|length >= 5 %}...{% endif %}
{% if "ifnotequal tag" != "beautiful" %}...{% endif %}
```

cobbler.web.templatetags.site.**smart_if**(*parser*, *token*)

A smarter {% if %} tag for django templates.

While retaining current Django functionality, it also handles equality, greater than and less than operators. Some common case examples:

```
{% if articles|length >= 5 %}...{% endif %}
{% if "ifnotequal tag" != "beautiful" %}...{% endif %}
```

Arguments and operators _must_ have a space between them, so `{% if 1>2 %}` is not a valid smart if tag.

All supported operators are: `or`, `and`, `in`, = (or ==), `!=`, >, >=, < and <=.

Module contents

Submodules

cobbler.web.field_ui_info module

Describes additional web UI properties of Cobbler fields defined in item_*.py.

Copyright 2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

cobbler.web.manage module

cobbler.web.settings module

cobbler.web.urls module

cobbler.web.views module

Module contents

8.2 Submodules

8.3 cobbler.api module

8.4 cobbler.autoinstall_manager module

8.5 cobbler.autoinstallgen module

8.6 cobbler.cexceptions module

Custom exceptions for Cobbler

Copyright 2006-2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

exception cobbler.cexceptions.**CX**(*value*, **args*)
Bases: *cobbler.cexceptions.CobblerException*

This is a general exception which get's thrown often inside Cobbler.

exception cobbler.cexceptions.**CobblerException**(*value*, **args*)
Bases: Exception

This is the default Cobbler exception where all other exceptions are inheriting from.

exception cobbler.cexceptions.**FileNotFoundException**(*value*, **args*)
Bases: *cobbler.cexceptions.CobblerException*

This means that the required file was not found during the process of opening it.

exception cobbler.cexceptions.**NotImplementedException**(*value*, **args*)
Bases: *cobbler.cexceptions.CobblerException*

On the command line interface not everything is always implemented. This is the exception which stated this.

8.7 cobbler.cli module

8.8 cobbler.clogger module

Python standard logging doesn't super-intelligent and won't expose filehandles, which we want. So we're not using it.

Copyright 2009, Red Hat, Inc and Others Michael DeHaan <michael.dehaan AT gmail>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.clogger.**Logger**(*logfile=None*)

Bases: object

Logger class for Cobbler which is wrapped around the Python3 standard logger.

Please don't use this. Utilize the standard logger from Python3 so we can get rid of this eventually.

critical(*msg: str*)

A critical message which is related to a problem which will halt Cobbler.

Parameters **msg** – The message to be logged.

debug(*msg: str*)

A message which is useful for finding errors or performance problems. Should not be visible in the production usage of Cobbler.

Parameters **msg** – The message to be logged.

error(*msg: str*)

An error message which means that Cobbler will not halt but the future actions may not be executed correctly.

Parameters **msg** – The message to be logged.

flat(*msg: str*)

This uses the print function from the std library. Avoid using this. This is only used for the report command in `cobbler/actions/report.py`

Parameters **msg** – The message to be logged.

info(*msg: str*)

An informational message which should be written to the target log.

Parameters **msg** – The message to be logged.

warning(*msg: str*)

A warning message which could possibly indicate performance or functional problems.

Parameters **msg** – The message to be logged.

8.9 cobbler.cobblerd module

8.10 cobbler.configgen module

8.11 cobbler.download_manager module

Cobbler DownloadManager

Copyright 2018, Jorgen Maas <jorgen.maas@gmail.com>

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 51 Franklin Street, Fifth Floor, Boston, MA 02110-1301 USA

class cobbler.download_manager.**DownloadManager**(*collection_mgr*, *logger=None*)
Bases: object

download_file(*url*, *dst*, *proxies=None*, *cert=None*)
Donwload a file from a URL and save it to any disc location.

Parameters

- **url** – The URL the request.
- **dst** – The destination file path.
- **proxies** – Override the default Cobbler proxies.
- **cert** – Override the default Cobbler certs.

urlread(*url*, *proxies=None*, *cert=None*)
Read the content of a given URL and pass the requests. Response object to the caller.

Parameters

- **url** – The URL the request.
- **proxies** – Override the default Cobbler proxies.
- **cert** – Override the default Cobbler certs.

Returns The Python `requests.Response` object.

8.12 cobbler.field_info module

Deprecated fields that have been renamed, but we need to account for them appearing in older datastructs that may not have been saved since the code change.

8.13 cobbler.module_loader module

8.14 cobbler.power_manager module

8.15 cobbler.remote module

8.16 cobbler.resource module

8.17 cobbler.serializer module

8.18 cobbler.services module

8.19 cobbler.settings module

8.20 cobbler.templar module

8.21 cobbler.template_api module

8.22 cobbler.tftpgen module

8.23 cobbler.utils module

8.24 cobbler.validate module

cobbler.validate.**hostname**(*dnsname: str*) → str
Validate the DNS name.

Parameters **dnsname** – Hostname or FQDN

Returns dnsname

Raises ***CX*** – If the Hostname/FQDN is not a string or in an invalid format.

cobbler.validate.**ipv4_address**(*addr: str*) → str
Validate an IPv4 address.

Parameters **addr** – IPv4 address

Returns str addr or CX

cobbler.validate.**ipv4_netmask**(*addr: str*) → str
Validate an IPv4 netmask.

Parameters **addr** – IPv4 netmask

Returns str addr or CX

cobbler.validate.**ipv6_address**(*addr: str*) → str
Validate an IPv6 address.

Parameters **addr** – IPv6 address

Returns The IPv6 address.

cobbler.validate.**mac_address**(*mac: str*, *for_item=True*) → str
Validate as an Eternet MAC address.

Parameters

- **mac** – MAC address
- **for_item** – If the check should be performed for an item or not.

Returns str mac or CX

cobbler.validate.**name_servers**(*nameservers: Union[str, list], for_item: bool = True*) → Union[str, list]
Validate nameservers IP addresses, works for IPv4 and IPv6

Parameters

- **nameservers** – string or list of nameserver addresses
- **for_item** – enable/disable special handling for Item objects

Returns The list of valid nameservers.

cobbler.validate.**name_servers_search**(*search: Union[str, list], for_item: bool = True*) → Union[str, list]
Validate nameservers search domains.

Parameters

- **search** – One or more search domains to validate.
- **for_item** – (enable/disable special handling for Item objects)

Returns The list of valid nameservers.

cobbler.validate.**object_name**(*name: str*, *parent: str*) → str
Validate the object name.

Parameters

- **name** – object name
- **parent** – Parent object name

Returns name or CX

cobbler.validate.**validate_autoinstall_script_name**(*name: str*) → bool
This validates if the name given for the script is valid in the context of the API call made. It will be handed to tftpgen.py#generate_script in the end.

Parameters **name** – The name of the script. Will end up being a filename. May have an extension but should never be a path.

Returns If this is a valid script name or not.

cobbler.validate.**validate_obj_id**(*object_id: str*) → bool

Parameters **object_id** –

Returns True in case it is one, False otherwise.

cobbler.validate.**validate_obj_name**(*object_name: str*) → bool

Parameters **object_name** –

Returns

cobbler.validate.**validate_obj_type**(*object_type: str*) → bool

Parameters **object_type** –

Returns

cobbler.validate.**validate_uuid**(*possible_uuid: str*) → bool
Validate if the handed string is a valid UUIDv4.

Parameters **possible_uuid** – The str with the UUID.

Returns True in case it is one, False otherwise.

8.25 cobbler.yumgen module

8.26 Module contents

CHAPTER 9

Release Notes for Cobbler 3.0.0

9.1 Enhancements

- Use new dracut ip option for configuring static interfaces (koan).
- Add a whitelist of directories in order to persist a `cobbler sync`.
- Add proxy support for get-loaders, signature update and reposync.
- Add initial support for DJBDNS.
- Enable external YUM repo mirroring through a proxy server.
- DHCP configuration now also supports the per interface gateway setting.
- A new interface_type `BMC` was added which also can be managed with DHCP.
- Yaboot was updated to 1.3.17.
- Add ability to have per-profile/per-system `next_server` values (#1196).
- Add `--graphics` option to Koan.
- Improved input validation and error handling.
- Support `virtio26` for generic QEMU fallback in Koan.
- Debian network config: add support for tagged vlan only bonding interfaces.
- Documentation has been converted into rST and is now included with the source tree.
- Integrated pyflakes into the build system and resolved hundreds of issues.
- Integrated pep8 (coding style) into the build system and resolved thousands of issues.
- Add a new field to the system type `ipv6_prefix` (#203).
- Minor update to CSS; make better use of screen (tables) (cobbler-web).
- Add support for an empty system status.
- If `dns-name` is specified, set it as DHCP hostname in preference to the `hostname` field.
- Allow user to choose whether or not to delete item(s) recursively (cobbler-web).
- Set ksdevice kernel option to MAC address for ppc systems as bootif is not used by yaboot.

- Return to list of snippets/kickstarts when snippet/kickstart is saved (cobbler-web).
- Layout in snippet/kickstart edit form has been improved (cobbler-web).
- Better handling of copy/remove actions for subprofiles (API and cobbler-web).
- Make kickstart selectable from a pulldown list in cobbler-web (#991).

9.2 Bugfixes

- Changed Apache configuration directory in Ubuntu 14.04 (#1208).
- build_reporting no longer fails with an empty string in ignorelist (#1248).
- Kickstart repo statement, filter invalid values: `gpgcheck`, `gpgkey` and `enabled` (#323).
- Several improvements to Debian/Ubuntu packaging.
- Some class/method names have been changed to make the code more intuitive for developers.
- Remove `root=` argument in Koan when using grubby and replace-self to avoid booting the current OS.
- Exit with an error if the cobblerd executable can't be found (#1108, #1135).
- Fix cobbler sync bug by xmlrpclib returning NoneType object.
- Dont send the Puppet environment when system status is empty (#560).
- Cobbler-web kept only the most recent interface change (#687).
- Fix broken gitdate, gitstamp values in `/etc/cobbler/version`.
- Prevent disappearing profiles after cobblerd restart (#1030).
- Add missing icons to cobbler_web/content (#679).
- cobbler-ext-nodes was broken with `mgmt_classes` defined at the profile level (#790).
- Properly name the VLAN interface in the manual page.
- Fix wrong address of the Free Software Foundation.
- Remove legacy (EL5/6) cruft from the RPM specfile.
- Koan: use the print function instead of the print statement.
- Minor improvement to LDAP configuration (#217).
- Improvements to the unittest framework.
- Removed several unused functions from utils.
- List of authors is now automagically generated.

9.3 Upgrade notes

- Support for LDAP configuration through Koan has been removed.
- Support for redhat_management (Spacewalk/Satelite) has been moved to contrib. Users of this functionality should checkout contrib/redhat-management/README.
- Monit support has been removed; you really need to use a CMS to manage your services.
- Support for remote kickstart templates and files been removed (eg. kickstart=http://).
- All object names are now validated like that of the system object.
- The use of `parent` and `distro` on subprofiles are now mutually exclusive.
- Support for s390/s390x has been removed.

- Support for ia64 (Itanium) has been removed.
- Support for the MySQL backend has been removed.
- Support for deprecated fieldnames (`subnet`, `bonding_master`, `bonding`) has been removed.
- Cobbler now requires python 2.7 and Koan now requires python 2.6.
- Red Hat specific default kernel options have been removed from the settings file.
- Support for Func integration has been moved to contrib. Users of this functionality should checkout contrib/func/README.
- Deprecated Koan LiveCD: moved to contrib.

CHAPTER 10

Limitations and Surprises

10.1 Templating

Before templates are passed to Jinja or Cheetah there is a pre-processing of templates happening. During pre-processing Cobbler replaces variables like `@@my_key@@` in the template. Those keys are currently limited by the regex of `\S`, which translates to `[^ \t\n\r\f\v]`.

CHAPTER 11

Indices and tables

- genindex
- modindex
- search

Python Module Index

C

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W