

id: installation title: Installation allowDiscussion: None subject: description: Instructions on how to install Puppet. contributors: creators: luke effectiveDate: 2005/08/29 18:38:45.808 GMT-5 expirationDate: None language: en rights: creation_date: 2005/08/29 15:51:40.818 GMT-5 modification_date: 2005/08/30 13:12:40.144 GMT-5 layout: document_view Content-Type: text/x-rst

Getting the Files

You will need to install Puppet on all machines that will use it, including both clients and servers.

Prerequisites

The only prerequisite for Puppet that doesn't come as part of the Ruby standard library is [factor](#), which is also [developed](#) by Reductive Labs.

All other prerequisites for Puppet are Ruby libraries, and they should all come with any standard Ruby 1.8.2 install. The other prerequisites, should your OS not come with the complete standard library, are:

- base64
- cgi
- digest/md5
- etc
- fileutils
- ipaddr
- openssl
- strscan
- syslog
- uri
- webrick
- xmlrpc

Factor

First install factor:

```
# get the latest tarball
wget http://reductivelabs.com/downloads/factor/factor-latest.tgz

# untar and install it
gzip -d -c factor-latest.tgz | tar xf -
cd factor-*
sudo ruby install.rb # or become root and run install.rb
```

There are also gems available in the [download](#) directory.

Install Puppet

Using the same mechanism, install the puppet libraries and executables:

```
# get the latest tarball
wget http://reductivelabs.com/downloads/puppet/puppet-latest.tgz

# untar and install it
gzip -d -c puppet-latest.tgz | tar xf -
cd puppet-*
sudo ruby install.rb # or become root and run install.rb
```

Alternative: Using RubyGems

You can also use Reductive Labs' Gems server to install Facter and Puppet:

```
gem install --remote --source http://reductivelabs.com/downloads facter
gem install --remote --source http://reductivelabs.com/downloads puppet
```

For more information on RubyGems, see the *Gems User Guide*.

Building the Server

Create Your Site Manifest

Because the Puppet language is declarative, it does not make as much sense to speak of “executing” Puppet programs, or to describe them as “scripts”. We choose to use the word *manifest* to describe Puppet programs, and we speak of *applying* those manifests to the local system. Thus, a *manifest* is a text document written in the Puppet language and meant to result in a desired configuration.

Puppet is written with the assumption that you will have one central manifest capable of configuring your entire network, which we call the *site manifest*. You could have multiple, separate site manifests if you wanted, but at this point each of them would need their own servers.

For more information on how to create the site manifest, see the [Language Reference](#) and the [Library Reference](#).

Puppet will look for your site manifest in `/etc/puppet/manifests/site.pp`, so create `/etc/puppet/manifests` and add your manifest, along with any files it includes, to that directory. It is highly recommended that you use some kind of [version control](#) on your manifests.

Start the Central Daemon

Most sites should only need a single central server. Reductive Labs will soon publish a document describing how to build puppet architectures with failover capabilities and architectures that are capable of handling large loads, but for now only a single server is supported.

Decide which machine you want to be your central server; this is where you will be starting `puppetmasterd`.

The best way to start any daemon is using your local server's service management system, often in the form of `init` scripts. Eventually Puppet will ship with an appropriate script for each platform, but in the meantime you can either create your own, using an existing script as an example, or simply run without one (not recommended).

The daemon should start just fine with no arguments:

```
/usr/bin/puppetmasterd
```

It will automatically create all necessary certificates, directories, and files. If you want the daemon to also function as a file server, so your clients can copy files from it, you will need to create a [fileservers configuration file](#).

Verifying Installation

To verify that your daemon is working as expected, pick a single client to use as a testbed. Once Puppet is installed on that machine, run a single client against the central server to verify that everything is working appropriately. You should start the first client in verbose mode, with the `--waitforcert` flag enabled:

```
puppetd --server myserver.domain.com --waitforcert 60 --verbose
```

The default server for `puppetd` is `puppet`, so you could just create a CNAME of that to whatever server is running `puppetmasterd`.

In running the client, you should see a message that the client did not receive a certificate (this message will repeat every 60 seconds with the above command). This is normal, since your server is not autosigning certificates as a security precaution. On your server, list the waiting certificates:

```
puppetca --list
```

You should see the name of the test client. Now go ahead and sign the certificate:

```
puppetca --sign mytestclient.domain.com
```

Within 60 seconds, your test client should receive its certificate from the server, receive its configuration, apply it locally, and exit normally.

Finishing Installation

Once you have verified that one client can connect to your server and is applying your configuration as you expect, configure each of your clients to connect to the server on the schedule you desire. It is planned to make `puppetd` a long-running process, but for now, install a cron job to run puppet on the schedule you want:

```
# in root's crontab, probably /var/spool/cron/crontabs/root
0,30 * * * * /usr/bin/puppetd --logdest syslog
```

The above example sends log messages to syslog; the default is to log them locally in `/var/puppet`.

Beta Notes

There are some important notes to keep in mind about using the beta of Puppet:

- Files are currently automatically reread when they are changed, within a timeout of 60 seconds.
- Patches are not only welcome, they're encouraged.