开 源 自 动 化 配 置 管 理 工 具 Puppet 入 门 教 程

系统管理员经常陷入一系列的重复任务中：如升级软件包、管理配置文件、系统服务、cron 任务以及添加新的配置、修复错误等。这些任务通常是重复低效的，解决这类任务的第一反应是让他们自动化，于是出现了定制脚本。由于环境复杂，定制脚本和应用程序一再被重复开发，并且很难适合多种平台，灵活性和功能也很难保证，于是像 Puppet 这样的自动化配置管理工具便出现了。

在开源世界里，有很多配置工具可供选择，这个领域一些关键的产品有：

Puppet():

Ruby 写成的配置管理工具，使用 C/S 架构，使用 declarative language 配置客户端。

Cfengine():

最先发布的开源配置工具之一， 1993 年发布，同样是 C/S 架构，通常应用于教育机构。

LCFG():

C/S 架构的配置管理工具，使用 XML定义配置。

Bcfg2

Python 编写的 C/S 架构的配置管理工具，使用规格书和客户机响应配置目标主机。

本文档致力于描述使用 Puppet 管理你的主机、应用程序、后台程序和各种服务。

Puppet 简介：

1. Puppet 的用途

Puppet 是开源的基于 Ruby 的系统配置管理工具， 依赖于 C/S 的部署架构。 主要开发者是 Luke Kanies ， 遵循 GPLv2版权协议。从 1997 年开始 Kanies 参与 UNIX 的系统管理工作，Puppet 的开发源于这些经验。因为对已有的配置工具不甚满意，从 2001 年到 2005 年间，Kanies 开始在 Reductive 实验室从事工具的开发。很快， Reductive 实验室发布了他们的旗舰产品—— Puppet 。

2. Pupput 的特性

许多系统配置管理工具工作的方式非常类似， 如 cfengine 。是什么让 Puppet 与众不同

Puppet 的语法允许你创建一个单独脚本，用来在你所有的目标主机上建立一个用户。所有的目标主机会依次使用适用于本地系统的语法解释和执行这个模块。举例：如果这个配置是在 Red Hat 服务器上执行，建立用户使用 useradd 命令 ; 如果这个配置是在 FreeBSD 主机上执行，使用的是 adduser 命令。

Puppet 另一个卓越的地方是它的灵活性。源于开源软件的天性，你可以自由的获得 Puppet 的源码， 如果你遇到问题并且有能力的话， 你可以修改或者加强 Puppet 的代码去适用于你的环境。另外，社区开发者和捐献者还在不断增强 Puppet 的功能。一个大的开发者和用户社区也致力于提供 Puppet 的文档和技术支持。

Puppet 也是易于扩展的。定制软件包的支持功能和特殊的系统环境配置能够快速简单的添加进 Puppet 的安装程序中。

3. Puppet 的工作模式

Puppet 是一个 C/S 架构的配置管理工具，在中央服务器上安装 puppet-server 软件包（被称作 Puppet master）。在需要管理的目标主机上安装 puppet 客户端软件（被称作 Puppet Client）。当客户端连接上 Puppet master 后，定义在 Puppet master 上的配置文件会被编译，然后在客户端上运行。每个客户端默认每半个小时和服务器进行一次通信，确认配置信息的更新情况。如果有新的配置信息或者配置信息已经改变，配置将会被重新编译并发布到各客户端执行。也可以在服务器上主动触发一个配置信息的更新，强制各客户端进行配置。如果客户端的配置信息被改变了，它可以从服务器获得原始配置进行校正。

4. Puppet 的未来

最后，Puppet 是一个年轻的工具，仍然处于开发和发展中。Puppet 社区快速壮大，并且许多新的想法不断融入，促使开发、更新和模块每天都在呈现。

安装配置：

1. Puppet 在 RedHat/CentOS 系统上安装

Puppet 是基于 Ruby 写成的，所以安装前要准备好 Ruby 环境。在中心的 Server 上安装 puppet-server 包，并运行 puppetmasterd 进程；在被管理机上安装 puppet 包，并运行 puppetd 进程。另外，在每台主机上配置好自己的 hostname，之后每台机器要以 hostname 区分。

1). 安装 ruby 环境：

```
1. yum install ruby ruby-rdoc
```

复制代码

2). 安装 puppet

Server 端安装：

```
1. yum install puppet-server
2. chkconfig –level 2345 puppetmaster on
```

复制代码

修改 hosts，添加下面行：

```
1. Vi /etc/hosts
```

复制代码

客户端安装：

```
1. yum install puppet
```

```
2.  chkconfig  –level 2345 puppet on
```

复制代码

修改 hosts，添加下面行：

```
1.  Vi /etc/hosts
```

复制代码

3). 启动 puppet

Server 端首次运行前，编辑 /etc/puppet/manifests/site.pp 文件，内容可以用最基本的：

```
1.  # Create “/tmp/testfile” if it doesn’t exist.
2.  class test_class {
3.  file {  “/tmp/testfile”:
4.  ensure => present,
5.  mode => 644,
6.  owner => root,
7.  group => root
8.  }
9.  }
10.  # tell puppet on which client to run the class
11.  include test_class
12.  }
```

复制代码

启动 Server 端：

```
1.  service puppetmaster start
```

复制代码

启动客户端：

```
1.  /etc/init.d/puppet once -v
```

复制代码

这时客户机会去连 server ，但是由于连接是在 ssl 上的，而 Server 还没有 sign 过客户端的 cert ，客户机被断开。

到 Server 端执行：puppetca -list ，会显示等待签名的客户端的主机名， 执行：puppetca -sign < 客户端主机名 > 即可为其签名。

```
1.  puppetca -list
```

复制代码

这时再到客户机上启动 puppetd，即可看到客户在正常地连接 server ，并且应用 Server 上为客户端定制的配置策略。

启动客户端：

```
1.  /etc/init.d/puppet once -v
```

复制代码

4). 测试：

也可以将日志直接打印到终端上进行测试：

Server 端： puppetmasterd -d – no-daemonize -v – trace

客户端： puppetd – test – trace – debug

2. puppet 配置文件

主配置文件 (puppet.conf) ：

1). 配置文件命名空间：

main 通用配置选项

puppetd 客户端配置选项

puppetmasterd 服务端配置选项

2). main 命名空间选项：

confdir 配置文件目录，默认在 /etc/puppet

vardir 动态数据目录，默认在 /var/lib/puppet

logdir 日志目录，默认在 /var/log/log

rundir puppet PID 目录，默认在 /var/run/puppet

statedir state 目录，默认在 $vardir/state

statefile state 文件，默认在 $statedir/state.yaml

ssldir SSL 证书目录，默认在 $vardir/ssl

trace 发生错误时显示跟踪信息，默认 false

filetimeout 检测配置文件状态改变的时间周期，单位秒，默认 15 秒

syslogfacility 指定 syslog 功能为 user 级，默认为 daemon级

3). puppetmasterd 命名空间选项 ：

user 后台进程执行的用户

group 后台进程执行的组

mainfestdir mainfests 文件存储目录，默认为 $confdir/mainfests

mainfest mainfest 站点文件的名字，默认为 site.pp

bindaddress 后台进程绑定的网卡地址接口

masterport 后台进程执行的端口，默认为 8140

4). puppet 命名空间选项 ：

server puppet puppet 服务器名，默认为 puppet

runinterval seconds puppet 应用配置的时间间隔，默认 1800 秒(0.5 小时)

puppetdlockfie file puppet lock 文件位置，默认 $statedir/puppetdlock

puppetport port 后台进程执行的端口，默认 8139

文件服务配置文件 (fileserver.conf) ：

```
1.  [files]

2.  path /var/lib/puppet/files
```

复制代码

path 定义文件存放路径，通过 allow/deny 来控制访问权限。

3. puppet 命令集

1). puppet 用于执行用户所写独立的 mainfests 文件

```
1.  # puppet -l /tmp/manifest.log manifest.pp
```

复制代码

2). puppetd 运行在被管理主机上的客户端程序

复制代码

3). puppetmasterd 运行在管理机上的服务器程序

```
1.  # puppetmasterd
```

复制代码

4). puppetca puppet 认证程序

```
1.  # puppetca -l
```

复制代码

5). puppetrun 用于连接客户端，强制运行本地配置文件

```
# puppetrun -p 10 –host host1 –host host2 -t remotefile -t webserver
```

复制代码

6). filebucket 客户端用于发送文件到 puppet file bucket 的工具

```
# filebucket -b /tmp/filebucket /my/file
```

复制代码

7). ralsh 转换配置信息到 puppet 配置代码

```
# ralsh user luke
user { ‘luke’:
home => ‘/home/luke’,
uid => ‘100′,
ensure => ‘present’,
comment => ‘Luke Kanies,,,’,
gid => ‘1000′,
shell => ‘/bin/bash’,
groups => ['sysadmin','audio','video','puppet']
}
```

复制代码

8). puppetdoc 打印 puppet 参考文档

```
# puppetdoc -r type > /tmp/type_reference.rst
# puppetdoc –outputdir /tmp/rdoc –mode rdoc /path/to/manifests
# puppetdoc /etc/puppet/manifests/site.pp
```

复制代码