puppet 安装与使用 -- 安装

网上参考，推荐一个不错的 puppet 文章站点 http://www.mysqlops.com/category/puppet ，百度搜 puppet 大部分好的文章还是出自这里，下面根据我的操作，记录下安装步骤和遇到的问题

Puppet 工作流程

1. Puppet 客户端请求节点配置 Puppet master 进行 SSL 认证
2. 将信息日志写入 cactlog
3. Puppet 解析器解释代码 （包括错误检查以及语法检查） ,并写入日志。
4. 事务处理（例如是否有权限，是否要同步文件，是否有 collection 资源）
5. 客户端发送日志
6. Ssl 认证授权结束，整个流程结束。

Puppet 安装

我的安装环境操作系统 cenos5.2 32 位

server	puppet1	192.168.140.78
clinet	puppet2	192.168.140.79
clinet	puppet3	192.168.140.80

安装前要注意：

由于 SSL 证书依赖时间同步。请注意服务端与客户端保持一致。推荐使用 ntp 同步时间。

由于基于主机名，推荐使用 FQDN，标准格式。认证过后请不要随便修改主机名。（我的主机名以上面 puppet1、2、3为例）

安装步骤（使用 **yum** 安装比较方便）

1). 配置源 /etc/yum.repos.d/ 下 puppetlabs.repo,如下：

```
[puppetlabs-products]
name=Puppet Labs Products $releasever - $basearch
baseurl=http://yum.puppetlabs.com/el/$releasever/products/$basearch
gpgkey=http://yum.puppetlabs.com/RPM-GPG-KEY -puppetlabs
enabled=1
gpgcheck=1

[puppetlabs-deps]
name=Puppet Labs Dependencies $releasever - $basearch
baseurl=http://yum.puppetlabs.com/el/$releasever/dependencies/$basearch
gpgkey=http://yum.puppetlabs.com/RPM-GPG-KEY -puppetlabs
enabled=1
gpgcheck=1
```

```
[puppetlabs-products-source]
name=Puppet Labs Products $releasever - $basearch - Source
baseurl=http://yum.puppetlabs.com/el/$releasever/products/SRPMS
gpgkey=http://yum.puppetlabs.com/RPM-GPG-KEY   -puppetlabs
failovermethod=priority
enabled=0
gpgcheck=1

[puppetlabs-deps-source]
name=Puppet Labs Source Dependencies $releasever - $basearch - Source
baseurl=http://yum.puppetlabs.com/el/$releasever/dependencies/SRPMS
gpgkey=http://yum.puppetlabs.com/RPM-GPG-KEY   -puppetlabs
enabled=0
pgcheck=1
```

2). yum install ruby ruby-libs ruby-shadow （Puppet 是用 ruby 语言写的，所以要安装 ruby 环境，服务器端与客户端都要安装 ）

yum -y install puppet puppet-server facter （安装 puppetmaster ） Puppet 客户端， 只需要使用 yum -y install puppet 即可 .

注意：

客户端和服务器端版本要一致。如果版本不一致的话，那么高版本的只能是 puppet server, 另一台只能为 puppet 客户端， 也就是说 **puppet** 服务端的版本可以 大于或者等于客户端版本，不可以小与，因为有些同学有些是 yum 安装的

```
[root@puppet1 ~]# rpm -aq | grep puppet
puppet-2.7.12-1.el5
puppet-server-2.7.12-1.el5
[root@puppet1 ~]#
```

测试安装

1). 在 puppet 安装完成后，主机名使用 fqdn, 用 dns 解析最好，如没有，可以使用 hosts，我用 hosts 解析，如下图。

```
[root@puppet1 ~]# more /etc/hosts
# Do not remove the following line, or various programs
# that require network functionality will fail.
127.0.0.1       localhost.localdomain   localhost puppet1
::1     localhost6.localdomain6 localhost6
192.168.140.78  puppet1
192.168.140.79  puppet2
192.168.140.80  puppet3
```

每台机子修改相对应 hostname

source /etc/sysconfig/network

使用 ntpdate 使每台机子时间同步

2). 启动 puppetmaster，检查防火墙确保 8140端口开放，在客户端运行命令测试：

/etc/init.d/puppetmaster start #(puppetmaster/puppet1 上运行)

puppetd --test --server puppet1 (# 客户端 puppet2/puppet3 上运行)

会有如下提示：

warning: peer certificate won't be verified in this SSL sessionExiting; no certificate found and

waitforcert is disabled

不用担心，这是因为第一次运行，服务端没有给客户端签名　　.

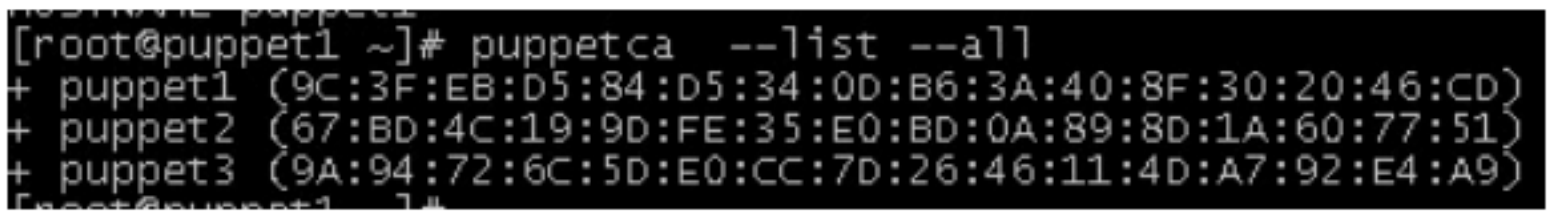
3). 在 puppetmaster(puppet1)执行

puppetca --list 查看有没有来验证签名的客户端

puppetca -s puppet2 给客户端 puppet2签名验证

puppetca -s -a　　给所有来服务端请求验证的客户端签名

4). 在服务端查看验证签名　,注意前面的 +号，说明已经签名



小结（列出一些别的命令）

使用 yum 的安装与配置比较方便，但　　puppet 有很多的资源模块需要进一步研究，下面列出几个服务端和客户端常用命令

1).puppetrun 用于连接客户端，　强制运行本地配置文件　,在默认安装 puppet 后 puppetrun 是不能使用，需要配置，如下：

客户端

/etc/puppet/puppet.conf 下添加

[puppetd]

listen = true

/etc/puppet/auth.conf 在 path / 　auth any 字样上面添加

path /run

method save

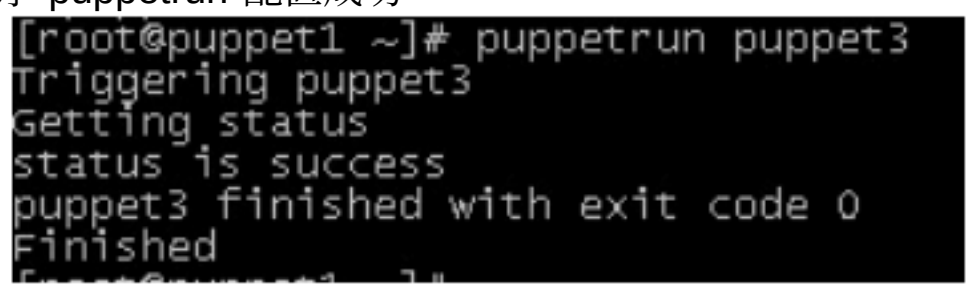
allow *

/etc/puppet/namespaceauth.conf 下

[puppetrunner]

allow *

在服务端执行 puppetrun puppet3（客户端名 ,需客户端服务 puppet 启动） 返回状态有 code 0，说明 puppetrun 配置成功



2).客户端启动方式

一般采用 /etc/init.d/puppet start 这样启动 这样默认客户端 30分钟会向服务端发请求，拉取配置文件，　30分可以修改　/etc/puppet/puppet.conf 中 puppetd 下 runinterval 参数值 单位为秒

如采用 --no-client 参数启动，及永不会向服务端主动请求，命令如下：

puppetd --no-client　--verbose --server puppet1 --no-daemonize --listen

还可以加 --debug 进入调试

3).服务端命令

puppetmasterd --genconfig > /etc/puppet/puppet.conf.default 可以生成默认配置文件

puppetmasterd -d -v --no-daemonize 调试启动模式

4).查看所有配置选项

puppet -configprint --all

puppet 安装与使用 -- 配置文件及常用资源

安装完 puppet 后，然后需要了解下 puppet 配置文件和一些常用资源的使用

1、默认情况下，配置文件位于 /etc/puppet 目录下

```
[root@puppet1 ~]# ls /etc/puppet
auth.conf  fileserver.conf  manifests  modules  namespaceauth.conf  puppet.conf
[root@puppet1 ~]#
```

moudules 和 namsapceauth.conf 在 yum 安装后没有，可自行创建

1).auth.conf 和 namespaceauth.conf 应该是一些认证，里面配置有先生顺序

2).modules 目录，里面存放一些自定义的模块，在 manifests 中调用，在 modules 某模块结构如下：

```
manifests   #存放代码，另外 init.pp 必须存在
files       #存放用于同步到客户端的文件
templates   #存放模板
```

3).manifests 目录，里面存放 site.pp 程序主入口

4).fileserver.conf ，文件服务配置文件，格式如下：

```
[tools]
path /etc/puppet/modules/
allow
```

5). puppet.conf，服务启动主配置文件

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

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/puppet

logdir 日志目录，默认在 $vardir/log

rundir puppet PID 目录，默认在 $vardir/run

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 $confdir/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

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

puppet 命名空间选项
server puppet puppet 服务器，默认为 puppet
runinterval seconds puppet 应用配置的时间间隔，默认 1800秒（0.5小时）
puppetport port 后台进程执行的端口，默认 8139

6).autosign.conf，自动答名，格式如下：

*	表示所有
puppet*	puppet 开头的所有
192.168.140.0/24	网段

事先要在 puppet.conf 中[puppetmaster] 下填加
autosign=true
autosing=/etc/puppet/autosign.conf 这两行

2、Puppet 能管理的资源有很多，常见的也是常用的，有 file,exec,package,service,cron 等（各个资源参数可以用法使用下面命令可以查到，简单说下常用的参数）

Puppet describe --list 查询所支持的资源
Puppet describe file 查看该资源的用法

1).**puppet file** 资源管理

【**puppet file** 参数介绍】

backup

决定文件的内容在被修改前是否进行备份 . 利用 filebucket 对文件进行备份 ,按文件的 md5sum 进行归类 ,便于恢复文件的时候找到文件 .可以把文件备份到 puppet 客户端 ,也可以通过设置 backpup => bucket_name 把文件备份到网络上的其他机器 . 如果 backup 的值是一个点号 ”.”开头的字符串 ,puppet 会把文件备份在同一目录下 ,备份文件的扩展名就是 bakcup 里面的那个字符串 .如果设置 backup => false , 该文件不做备份 .

checksum

怎样检查文件是否被修改 ,这个状态用来在复制文件的时候使用 , 这里有几种检测方式 , 包括 md5 ,mtime,time,timestamp 等.

默认的检测是用 md5

content

把文件的内容设置为 content 参数后面的字符串 , 新行 ,tab,空格可用 escaped syntax 表示

ensure

如果文件本来不存在是否要新建文件 ,可以设置的值是 absent 和 present,file 和 directory. 如果指定 present,就会检查该文件是否存在 , 如果不存在就新建该文件 ,如果指定是 absent, 就会删除该文件 (如果 recurse => true ,就会删除目录).

force

force 当前的唯一作用是用在把一个目录变成一个链接 ,可用的值是 ture 和 false

group

指定那个该文件的用户组 ,值可以是 gid 或者组名

ignore

当用 recursion 方法复制一个目录的时候 ,可以用 ignore 来设定过滤条件 ,符合过滤条件的文件不被复制 . 使用 ruby 自带的匹配法则 .因此 shell 级别的过滤表达式完全支持 , 例如 [a-g]*

links

定义操作符合链接文件 . 可以设置的值是 follow 和 manage; 文件拷贝的时候 ,设置 follow, 会拷贝文件的内容 ,而不是只拷贝符合链接本身 ,如果设置成 manage ,会拷贝符合链接本身 .

Mode

mode 用于设置文件的权限

owner

设置文件的属主

path

指定要管理文件的路径 ,必须用引号引起来 , 这也是一个资源的 namevar ,通常 path 等于资源的 title

recurse

设置是否以及如何进行递归操作 , 可以设置的值是 false,true ,inf ,remote

source

拷贝一个文件覆盖当前文件 ,用 checksum 来判断是否有必要进行复制 ,可以设置的值是一个引用的完整的文件路径 ,或者是 URI, 当前支持的 URI 只有 puppet 和 file ; 这是一个对文件常用的操作 ,可以让 puppet 修改系统的配置文件 .

target

是为创建链接的。可以设置的值为 notlink.

type

检查文件是否只读

2).puppet exec资源管理

【**puppet exec** 参数介绍】

command:将会被执行的命令，必须为被执行命令的绝对路径，或者得提供该命令的搜

索路径。如果命令被成功执行，所有的输出会被记录在实例的正常（ normal）日志里，但是如果命令执行失败（既返回值与我们所指定的不同） ，那么所有的输出会在错误（ err）日志中被记录。

这个是 exec 资源类型的名变量（ namevar)。

creates:

指定命令所生成的文件。 如果提供了这个参数， 那么命令只会在所指定的文件不存在的情况的被执行：

cwd:

指定命令执行的目录。如果目录不存在，则命令执行失败。

Group

定义运行命令的用户组。在不同的平台下的运行的结果无法确定，由于不同用户运行命令的时候，变量是不变的，所以这是平台的问题，而不是 Ruby 或 Puppet 的问题。

logoutput

是否记录输出。默认会根据 exec 资源的日志等级（ loglevel） 来记录输出。若定义为 on_failure，则仅在命令返回错误的时候记录输出。可取的值为： true，false 和其他合法的日志等级。

Onlyif

如果这个参数被设定了，则 exec 只会在 onlyif 设定的命令返回 0时才执行。例如：

exec { "logrotate": path => "/usr/bin:/usr/sbin:/bin", onlyif => "test `du /var/log/messages | cut -f1` -gt 100000" }

只有在 test 返回 true 的时候 logrotate 才会被运行。

需要注意的是 onlyif 定义的命令跟主命令遵循同样的规则，也就是说如果 path 没有被设置的话，需要使用绝对路径。

除此之外， onlyif 还可以接受数组做为其值，例如：

onlyif => ["test -f /tmp/file1", "test -f /tmp/file2"]

上面的代码限定了只有在所有数组中的条件返回 true 时 exec 才会被执行。 “

path

命令执行的搜索路径。如果 path 没有被定义，命令需要使用绝对路径。路径可以以数组或以冒号分隔的形式来定义。

refresh

定义如何更新命令。当 exec 收到一个来自其他资源的事件时，默认只会重新执行一次命令。不过这个参数允许你定义更新时执行不同的命令。

refreshonly

该属性可以使命令变成仅刷新触发的， 也就是说只有在一个依赖的对象被改变时， 命令才会被执行。 仅当命令与其他对象有依赖关系时， 这个参数才有意义。 当你要触发某个行为时，会显得很有用：

Pull down the main aliases file file { "/etc/aliases": source => "puppet://server/module/aliases" }

Rebuild the database, but only when the file changes exec { newaliases: path => ["/usr/bin", "/usr/sbin"], subscribe => File["/etc/aliases"], refreshonly => true }

要注意的是只有 subscribe 和 notify 可以促发行为，而不是 require，所以在使用 refreshonly

时，只有同时使用 subscribe 或 notify 才有意义。有效的值为 true, false。

unless
如果这个变量被指定了，那么 exec 会执行，除非 unless 所设定的命令返回 0。？

user
定义运行命令的用户。 注意如果你使用了这个参数，那么任何的错误输出不会在当下被捕捉，这是 Ruby 的一个 bug。

3).puppet package资源管理

【puppet package 参数介绍】

adminfile
软件包管理器，通常是为了安装软件包。这个参数只适用于 Solarias

allowcdrom
告诉 apt 允许使用 cdrom 作为软件源 ， 可以设置成 false 或者 true

category
软件包设置的一个只读的属性

configfiles
是否保留或者替换软件的配置文件 ,大多数软件不支持这个参数 ,可设置的值是 false,true

description
描述软件包 ,软件包设置的一个只读属性

ensure
设置该软件包应该在什么状态 . installed 表示要安装该软件 ,也可以写成 present; absent 表示反安装该软件 ,pureged 表示干净的移除该软件 ,latest 表示安装软件包的最新版本 .

namevar
该资源的 namevar ； 软件包的名字

4).puppet service资源管理

【puppet service 参数介绍】

binary ：可执行文件。这个只用在不支持 init 脚本的情况下。

control ：用来管理服务的，通常是用在 hp-unix 系统上。

enable ：服务是否随开机而启动，可设置的值为 true,false.

ensure： 服务是否运行，可设置的值为 running,stopped, 也可以用 true,false.

hasrestart： 服务的 init 脚本是否支持 restart 参数，可设置的值为 true,false.

hasstatus ：服务的 init 脚本是否支持 status 参数，可设置的值为 ture,false.

manifest ：Specify a command to config a service,or a path to a manifest to do so. 配置服务的命令，或者指定路径。

name： 要运行的服务名字。

path： 指定查找 init 脚本的路径。

pattern ：搜索进程表匹配字符串，用于不支持 init 的脚本。

provider ：The specific backend for provider to use ， 可设置的值有 base, daemontools,init 等

restart ：重启服务。

Start： 开启服务

status ：服务运行状态

stop： 停止服务

5).puppet cron资源管理

【**puppet cron** 参数】

command ：

crontab 要执行的命令 ， 环境变量按照系统本地规则进行管理 ,推荐使用绝对路径 .

ensure：指定该资源是否启用 ,可设置成 true 或 false

environment：在 crontab 环境里面指定环境变量 ,例如 PATH=/bin:/usr/bin:/usr/sbin.

hour： 运行 crontab 的小时 ,可设置成 0-23

minute：运行 crontab 的分钟 ,可设置成 0-59

month： 设置 crontab 运行的月份 ,1-12

monthday：一个月份中的日子 ,1-31

name：

该 crontab 的名字 ,这个名字用于管理员区分不同的 crontab,以及 puppet 管理各种资源关系 .

user：

把该 crontab 加到那个用户的 crontab 列表 ,默认是运行 puppet 的用户

weekday ：

运行 crontab 的星期数 ,0-7，周日是为 0.

3、简单使用例子

puppet 程序的主要入口在 /etc/puppet/manifests/notes.pp 中，

修改 notes.pp ， 如下配置

```
node default {
        file {"/tmp/test":
                ensure => "directory";
                "/tmp/test/test.sh":
                ensure => "present",
                mode => 744,
                owner => root,
                content => "#/bin/bash\necho test | mail -s test wangliqiang@kingsoft.com\n"}

        cron {"crontab":
                command => "/tmp/test/test.sh",
                user => root,
                minute => 59,
                hour => 7}

    exec {"test":
                command => "echo test>/tmp/test/test",
                user => "root",
                path => "/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin"}
```

```
        service {["smb","nfs"]:
                ensure => "stopped"}

        package {"screen":
                ensure => "installed"}
}
```

上面简单例子解释： 对于默认节点， 在 /tmp/ 下创建 test 目录，在 test 目录下再建立文件 test.sh, 内容为 #/bin/bash echo test | mail -s test wangliqiang@kingsoft.com ，属主为 root, 属性 744，并在每天 7 点 59 分执行此脚本，测试 exec 资源，在 /tmp/test 下用命令建立 test 文件，内容为 test,使 smb,nfs 服务处于停止状态，安装 screen 包

验证：在服务端 puppet1 上执行 puppetrn puppet3 后，到客户端 puppet3 上查看

```
[root@puppet3 ~]# cd /tmp/test/
[root@puppet3 test]# ls
test  test.sh
[root@puppet3 test]# cat test
test
[root@puppet3 test]# cat test.sh
#/bin/bash
echo test | mail -s test wangliqiang@kingsoft.com
[root@puppet3 test]# crontab -l
# HEADER: This file was autogenerated at Fri Mar 23 15:05:29 +0800 2012 by puppet.
# HEADER: While it can still be managed manually, it is definitely not recommended.
# HEADER: Note particularly that the comments starting with 'Puppet Name' should
# HEADER: not be deleted, as doing so could cause duplicate cron jobs.
# Puppet Name: crontab
59 7 * * * /tmp/test/test.sh
[root@puppet3 test]# service smb status
smbd is stopped
nmbd is stopped
[root@puppet3 test]# service nfs status
rpc.mountd is stopped
nfsd is stopped
[root@puppet3 test]#
```

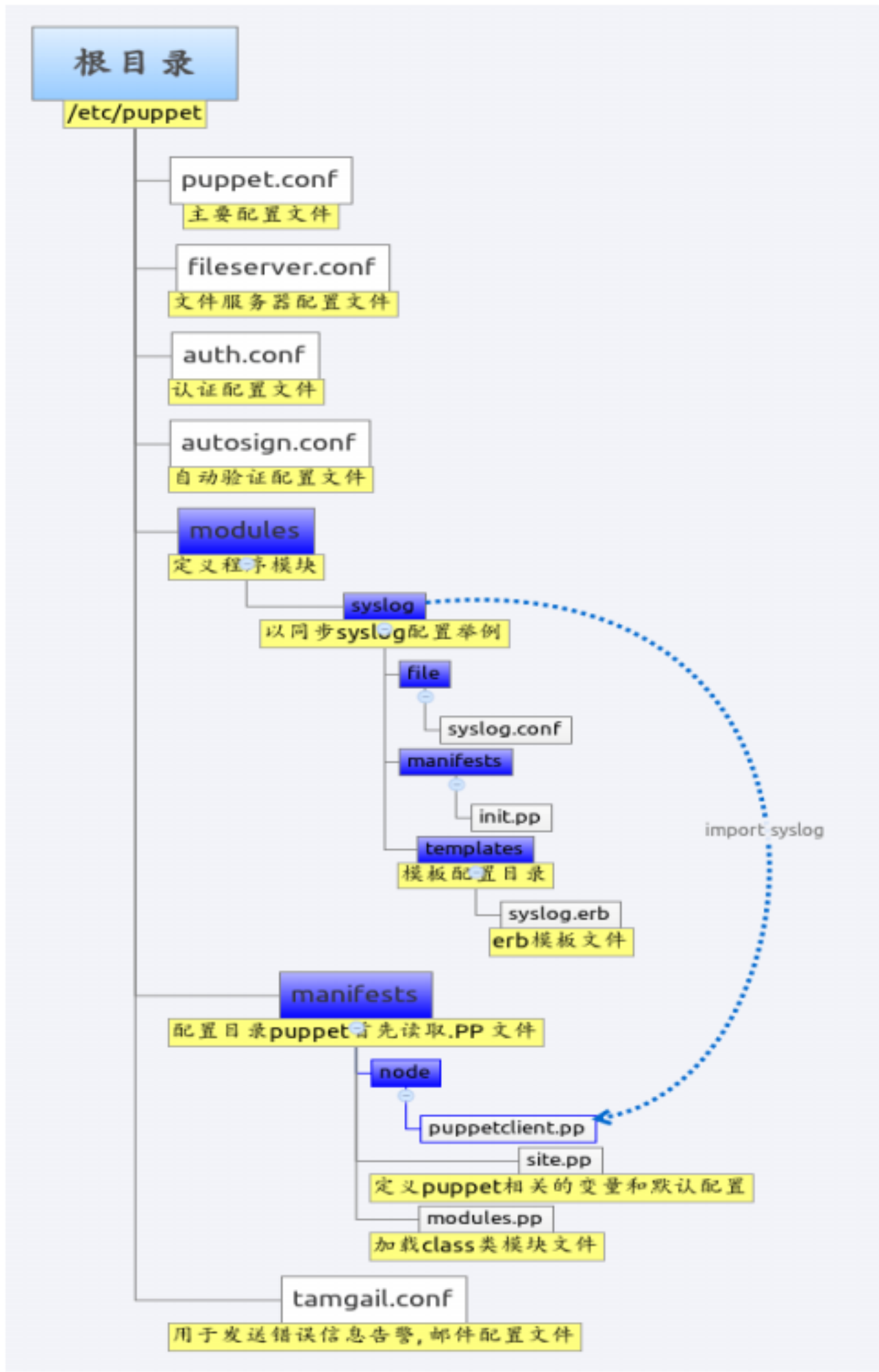
puppet 安装与使用 -- 模块结构（ iptables 与 rsync 模块）

模块结构 3 ^& I; o6 Q# Q5 g

一个模块就是一个 /etc/puppet/modules 目录下面的一个目录和它的子目录，在 puppet 的主文件 site.pp 里面用 import modulename 可以插入模块。新版本的 puppet 可以自动插入 /etc/puppet/modules 目录下的模块。引入模块，可以结构化代码，便于分享和管理。

例如关于 apache的所有配置都写到 apache 模块下面。一个模块目录下面通常包括三个目录： files，manifests，templates。manifests 里面必须要包括一个 init.pp 的文件，这是该模块的初始（入口）文件，导入一个模块的时候，会从 init.pp 开始执行。可以把所有的代码都写到 init.pp 里面，也可以分成多个 pp 文件， init 再去包含其他文件。 files 目录是该模块的文件发布目录， puppet 提供一个文件分发机制， 类似 rsync 的模块。 templates 目录包含 erb 模型文件，这个和 file 资源的 template 属性有关。

网上摘个图，还不错



如上结构类似，自己也可以跟着创建相应目录，如下图，我的测试机器上目录结构：

```
[root@puppet1 ~]# cd /etc/puppet
[root@puppet1 puppet]# tree -L 3
.
|-- auth.conf
|-- fileserver.conf
|-- manifests
|   |-- modules.pp
|   |-- nodes.pp
|   |-- site.pp
|   `-- site.pp1
|-- modules
|   |-- concat
|   |   |-- COPYING
|   |   |-- README.md
|   |   |-- lib
|   |   `-- pkg
|   |-- cron
|   |   |-- files
|   |   `-- manifests
|   |-- iptables
|   |   |-- files
|   |   `-- manifests
|   `-- rsync
|       |-- COPYING
|       |-- README.md
|       |-- lib
|       |-- manifests
|       |-- pkg
|       |-- spec
|       `-- templates
|-- namespaceauth.conf
`-- puppet.conf

17 directories, 12 files
[root@puppet1 puppet]#
```

结构与上叙述完全相同，每个模块下 manifests 下总会有一个 init.pp 文件，总规是一个套一个

iptalbes 模块例子使用

结构与上，在 /etc/puppet/modules 模块下增加 rsync 模块

```
[root@puppet1 rsync]# cd /etc/puppet/modules/iptables/
[root@puppet1 iptables]# tree
.
|-- files
|   `-- iptables.sh
`-- manifests
    |-- init.pp
    `-- init.pp1

2 directories, 3 files
[root@puppet1 iptables]#
```

事先要把给客户端定制好的 iptables.sh 脚本放入 files 下，编写 init.pp

```
[root@puppet1 iptables]# cd manifests/
[root@puppet1 manifests]# more init.pp
class iptables {
        file
          { "/root/iptables":
          ensure => "directory";
         "/root/iptables/iptables.sh":
          mode => 0755,
          ensure => present,
          source => "puppet://$fileserver/iptables/iptables.sh",
          notify => Exec["sh-iptables"],
          }
          exec
          { "sh-iptables":
            cwd => "/root/iptables",
            command => "sh iptables.sh",
            user => "root",
            path => "/usr/local/sbin:/usr/local/bin:/sbin:/bin:/usr/sbin:/usr/bin",
          }
}
[root@puppet1 manifests]#
```

创建一个 class 类，在模块 modules.pp 中调用即可，主入口为 site.pp，所以在 site.pp 要调用 modules.pp

上图中难理解的就是 source => "puppet://$fileserver/iptables/iptables.sh"，这个是 puppet 代码中就写好的，每个模块 files 下面的文件，都可以写成 puppet://$fileserver/ 模块名 /files 文件下的文件名 " 来调用 notify 指定了顺序，在执行 exec 前执行 file，因为 puppet 程序里面定义的资源是同时执行的，不分先后，所以控制先后顺序必须的利用某个参数

模块写好后，看 /etc/puppet 下文件 内容，如下：

在 modules.pp 中 import " 模块名 ”，在 site.pp 里面 import modules.pp

```
[root@puppet1 manifests]# cd /etc/puppet
[root@puppet1 puppet]# ls
auth.conf  fileserver.conf  manifests  modules  namespaceauth.conf  puppet.conf
[root@puppet1 puppet]# cd  manifests/
[root@puppet1 manifests]# cat modules.pp
import "iptables"
import "rsync"
import "cron"
[root@puppet1 manifests]# cat site.pp
Exec { path => "/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin" }
import "nodes.pp"
import "modules.pp"
[root@puppet1 manifests]#
```

还有个节点 pp，可义为 nodes.pp，在 init.pp 里也有 import,nodes.pp 里面可以这样写

```
node default {
                include 'iptables'}
```

对默认节点使用 iptables,要结某个节点使用 iptables，可以 node ' 节点名 ' {}

验证测试

rsync 模块例子使用

Puppet 结合 rsync 同步文件是参考网上一篇文章，模块是由别人写的，可以下载自由使用，但测试后，发现 rsync 客户端同步还是有问题，所以自己在里面填了部分 client，可以实现文件同步

参考文章网址： http://www.mysqlops.com/2012/02/20/puppet-rsync.html#comment-1564

当然， 同步文件也可以使用 puppet 文件服务器， 要用到 fileserver.conf 文件 ，同步小文件可以，但考虑到要同步大文件还是考虑用 rsync 效率要高点

根据文章中，先下载两个模块到 modules

cd /etc/puppet/modules/

git clone https://github.com/onyxpoint/pupmod-concat && mv pupmod-concat concat

git clone https://github.com/onyxpoint/pupmod-rsync && mv pupmod-rsync rsync

Git 如果没有装的话， yum 下， git 好象类似 svn（有待研究）， git 时可能会报个错，如下解决即可

Cloning into cancan …

error: SSL certificate problem, verify that the CA cert is OK. Details:

error:14090086:SSL routines:SSL3_GET_SERVER_CERTIFICA TE:certificate verify failed while

。。。。。

解决办法：

git config --global http.sslV erify false

下载后，就该配　rsync 服务了，我选择　puppet1(server）作为 rsync-server,在 nodes.pp 里面如下配置

```
node 'puppet1'{
        include 'rsync::server'
        rsync::server::global { 'global':
        address => '192.168.140.78'
        }
        rsync::server::section { 'default':
        comment => 'the default file path',
        path => '/usr/local/src',
        hosts_allow => '192.168.140.79'
        }
        rsync::server::section { 'test':
        comment => 'Test comment',
        path => '/usr/local/src',
        hosts_allow => '192.168.140.79,192.168.140.80',
        outgoing_chmod => 'o-w'
        }

}
```

在 puppet1运行　puppetd --test --server puppet1 , 即可生成　rsyncd.conf 文件，启动　rsync 服务（注意：防火墙中要过滤　873服务端口）

```
[root@puppet1 manifests]# more /etc/rsyncd.conf
pid file = /var/run/rsyncd.pid
syslog facility = daemon
port = 873
address = 192.168.140.78
[default]
comment = the default file path
path = /usr/local/src
use chroot = false
max connections = 0
max verbosity = 1
lock file = /var/run/rsyncd.lock
read only = true
write only = false
list = false
uid = root
gid = root
outgoing chmod = o-w
ignore nonreadable = true
transfer logging = true
log format = "%o %h [%a] %m (%u) %f %l"
dont compress = *.gz *.tgz *.zip *.z *.rpm *.deb *.iso *.bz
hosts allow = 192.168.140.79
hosts deny = *
[test]
comment = Test comment
path = /usr/local/src
use chroot = false
max connections = 0
max verbosity = 1
lock file = /var/run/rsyncd.lock
read only = true
write only = false
list = false
```

rsyncd 的配置文件　应该大家不默认，不多说了

服务端配置好后，　需要配置客户端了，　但按上面那个链接来做，　我没有成功，　可能跟版本有关，大家可以尝试下他的做法，我现在共享下我的做法

cd /etc/puppet/modules/rsync/manifests/

进到下载的模块后，有　init.pp　server　server.pp 3个文件，观察里面内容，可自己编写 client.pp　client/rs.pp

```
[root@puppet1 manifests]# cd /etc/puppet/modules/rsync/manifests/
[root@puppet1 manifests]# more client/rs.pp
define rsync::client::rs ($title,$rserver,$source,$target ) {
exec {"$title":
command => "rsync -av $rserver::$source $target"
}
}

[root@puppet1 manifests]# more client.pp
class rsync::client inherits rsync {
}
[root@puppet1 manifests]#
```

回到 notes.pp 对节点 puppet2做下面配置,

```
node 'puppet2'{
        include 'rsync::client'
        rsync::client::rs {"client1":
        title => "client1",
        rserver => '192.168.140.78',
        source => 'test',
        target => '/tmp/test'
        }
        rsync::client::rs {"client2":
        title => "client2",
        rserver => '192.168.140.78',
        source => 'default',
        target => '/tmp/test1'
        }
}
```

注：上面　rsync 在服务端配置两模块,　default 和 test，对应路径都为　puppet1 下/usr/local/src 在 notes.pp 对节点　2 中，我写的是定义的模块名，所以意思就是把　puppet1/usr/local/src 下的文件同步到　puppet2 下的 /tmp/test /tmp/test1

验证测试：

在 puppet1 上，有两文件

```
[root@puppet1 ~]# ls /usr/local/src/
123.tar.gz  nload-0.7.3.tar.gz
[root@puppet1 ~]#
```

在 puppet2 上

同步成功

Puppet 功能非常强大，自身包括了很多的资源，根据自己的爱好和自己工作实际所需，可以有选择有研究，还有 cron 也是很好用的，并且配置也是比较简单的，模块文档先写到这里，后续的模块研究希望可以和大家一起研究

Puppet dashboard　　安装

puppet dashboard是 **GUI**（图形用户界面）方式管理　**puppet**，可以查看　**puppet** 运行日志

安装所需支持的　rpm 包（在　Puppet Server　端操作）

```
yum -y install openssl-devel gcc-c++ zlib-devel readline-devel mysql mysql-server mysql-devel
```

安装 ruby

```
wget http://rubyenterpriseedition.googlecode.com/files/ruby-enterprise-1.8.7-2011.03.tar.gz
tar xvzf ruby-enterprise-1.8.7-2011.03
./ruby-enterprise-1.8.7-2011.03/installer
```

需要耐心等待一段时间。出现选择安装路径。本人写的是　/usr/local/ruby

```
echo "export PATH=/usr/local/ruby/bin/:$PATH" >> /etc/profile
source /etc/profile
```

```
[root@puppet1 ~]# ruby -v
ruby 1.8.7 (2011-02-18 patchlevel 334) [i686-linux], MBARI 0x8770, Ruby
[root@puppet1 ~]#
```

安装 rubygems

```
wget http://production.cf.rubygems.org/rubygems/rubygems-1.3.6.tgz
tar zxvf rubygems-1.3.6.tgz ; ruby rubygems-1.3.6/setup.rb
gem install mysql
gem install rake
```

```
[root@puppet1 ~]# gem -v
1.3.6
[root@puppet1 ~]#
```

注：ruby 与 gem在默认安装的 5.2 系统上，yum的版本太低，如不升级会出现下面错误，yum 在此应用不太方便，版本不同，后现会出现各种报错，所以采用编译会好点，参照的这篇文档很顺利，没有太大问题

```
[root@puppet1 ~]# gem -v
1.3.1
[root@puppet1 ~]# gem install rake
ERROR:  Error installing rake:
        rake requires Ruby version >= 1.8.6
[root@puppet1 ~]#
```

```
[root@puppet1 ruby-1.9.1-p376]# gem install rake
ERROR:  Error installing rake:
        rake requires RubyGems version >= 1.3.2
Updating class cache with 1381 classes...
[root@puppet1 ruby-1.9.1-p376]#
```

安装 Puppet dashboard

```
wget http://puppetlabs.com/downloads/dashboard/puppet-dashboard-1.2.2.tar.gz
```

tar xzvf puppet-dashboard-1.2.2.tar.gz

mv puppet-dashboard-1.2.2 /usr/local/puppet-dashboard

chown -R puppet.puppet /usr/local/puppet-dashboard

创建 mysql 数据库

```
mysql >create database dashboard character set utf8;
mysql >grant all on dashboard.* to dashboard@'localhost' identified by 'dashboard';
mysql >flush privileges;
```

配置 puppet dashboard

```
cd /usr/local/puppet-dashboard
cp config/database.yml.example config/database.yml
cp config/settings.yml.example config/settings.yml
```

修改 config/database.yml 如下

```
production:
database: dashboard
username: dashboard
password: dashboard
encoding: utf8
adapter: mysql
```

导入 mysql

```
[root@puppet1 ~]# cd /usr/local/puppet-dashboard/
[root@puppet1 puppet-dashboard]# rake RAILS_ENV=production db:migrate
```

启动 puppet dashboard

./script/server -e production (可面可加 -p 端口和 -d 选项)

因为修改 database.yml 中为 production ，所以启动时要加 -e production ，默认不加启动为 development

在浏览器上输入 192.168.140.78:3000 会出现图形界面

puppet dashboard 识别客户端

Puppet client 配置 /etc/puppet/puppet.conf [agent] 下添加

```
report = true
```

重启服务 /etc/init.d/puppet restart

Puppet Server 配置 /etc/puppet/puppet.conf [main] 下添加

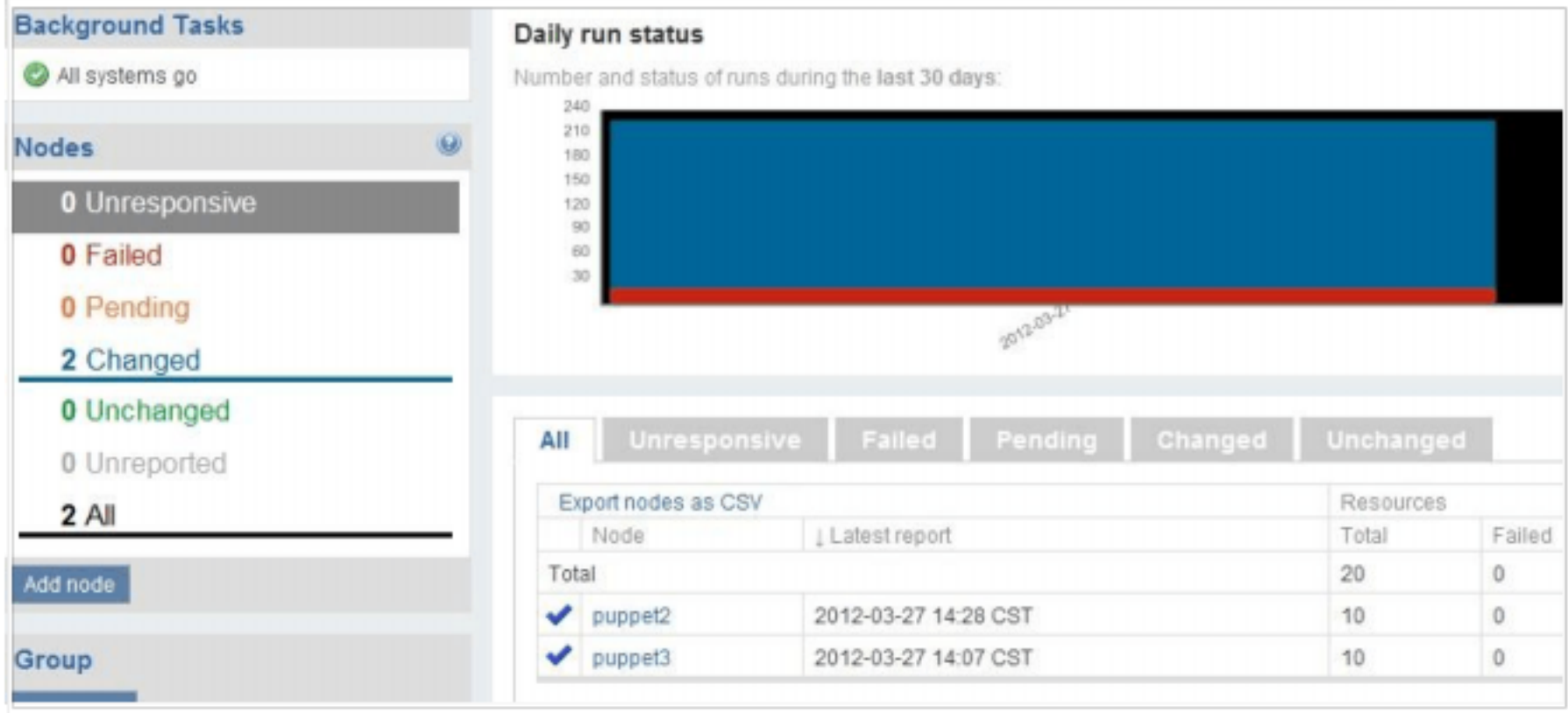
reports = store, http

重启服务 /etc/init.d/puppetmaster restart

识别

```
[root@puppet1 puppet-dashboard]# env RAILS_ENV=production script/delayed_job -p dashboard -n 4 -m start
```

在浏览器中访问如下图，对于每个节点获取服务器资源后，都能看到是成功还是失败的状态：



注：有关常用 Puppet Dashboard 操作命令可见参考文档中

参考文档： http://www.inanu.net/post/725.html

http://notry.blog.51cto.com/3909869/714003

puppet dashboard 启动脚本 （可以网上复制一个，一大堆，复制回来根据自己的情况要改下）

```
[root@puppet1 ~]# cat /etc/init.d/puppet-dashboard |grep -v ^# | grep -v ^$
. /etc/rc.d/init.d/functions
if [ -f /etc/sysconfig/puppet-dashboard ]; then
    . /etc/sysconfig/puppet-dashboard
fi
prog=puppet-dashboard
DASHBOARD_HOME=/usr/local/puppet-dashboard
RETVAL=0
start() {
        echo -n $"Starting $prog: "
        /usr/local/ruby/bin/ruby ${DASHBOARD_HOME}/script/server -e production >/dev/null 2>&1 &
        RETVAL=$?
        if [ $RETVAL = 0 ]
        then
            echo $! > /var/run/puppet-dashboard.pid
            echo_success
        else
            echo_failure
        fi
        echo
        return $RETVAL
}
stop() {
        echo -n $"Stopping $prog: "
        killproc puppet-dashboard
        RETVAL=$?
        echo
        [ $RETVAL = 0 ] && rm -f /var/run/puppet-dashboard.pid
}
case "$1" in
    start)
        start
        ;;
    stop)
        stop
        ;;
    restart)
        stop
        start
        ;;
    *)
        echo $"Usage: $prog {start|stop|restart}"
        exit 1
esac
exit $RETVAL
[root@puppet1 ~]#
```

注：红色部分根据自己的安装情况来修改