通过 MCollective 更加安全地实现 puppet 的推送更新功能

1 介绍

1.1 Mcollective 介绍

MCollective 是一个构建服务器编排(Server Orchestration)和并行工作执行系统的框架。首先，MCollective 是一种针对服务器集群进行可编程控制的系统管理解决方案。在这一点上，它的功能类似：Func，Fabric 和 Capistrano。

其次，MCollective 的设计打破基于中心存储式系统和像 SSH 这样的工具，不再仅仅痴迷于 SSH 的 For 循环。它使用发布订阅中间件(Publish Subscribe Middleware)这样的现代化 工具和通过目标数据(meta data)而不是主机名(hostnames)来实时发现网络资源这样的现代化理念。提供了一个可扩展的而且迅速的并行执行环境。

MCollective 工具为命令行界面，但它可与数千个应用实例进行通信，而且传输速度惊人。无论部署的实例位于什么位置，通信都能以线速进行传输，使用的是一个类似多路传送的推送信息系统。MCollective 工具没有可视化用户界面，用户只能通过检索来获取需要应用的实例。Puppet Dashboard 提供有这部分功能。

MCollective 特点：

- 能够与小到大型服务器集群交互
- 使用广播范式(broadcast paradigm)来进行请求分发，所有服务器会同时收到请求，而只有与请求所附带的过滤器匹配的服务器才会去执行这些请求。没有中心数据库来进行同步，网络是唯一的真理
- 打破了以往用主机名作为身份验证手段的复杂命名规则。使用每台机器自身提供的丰富的目标数据来定位它们。目标数据来自于：Puppet, Chef, Facter, Ohai 或者自身提供的插件
- 使用命令行调用远程代理
- 能够写自定义的设备报告
- 大量的代理来管理包，服务和其他来自于社区的通用组件
- 允许写 SimpleRPC 风格的代理、客户端和使用 Ruby 实现 Web UIs
- 外部可插件化(pluggable)实现本地需求
- 中间件系统已有丰富的身份验证和授权模型，利用这些作为控制的第一道防线。
- 重用中间件来做集群、路由和网络隔离以实现安全和可扩展安装。

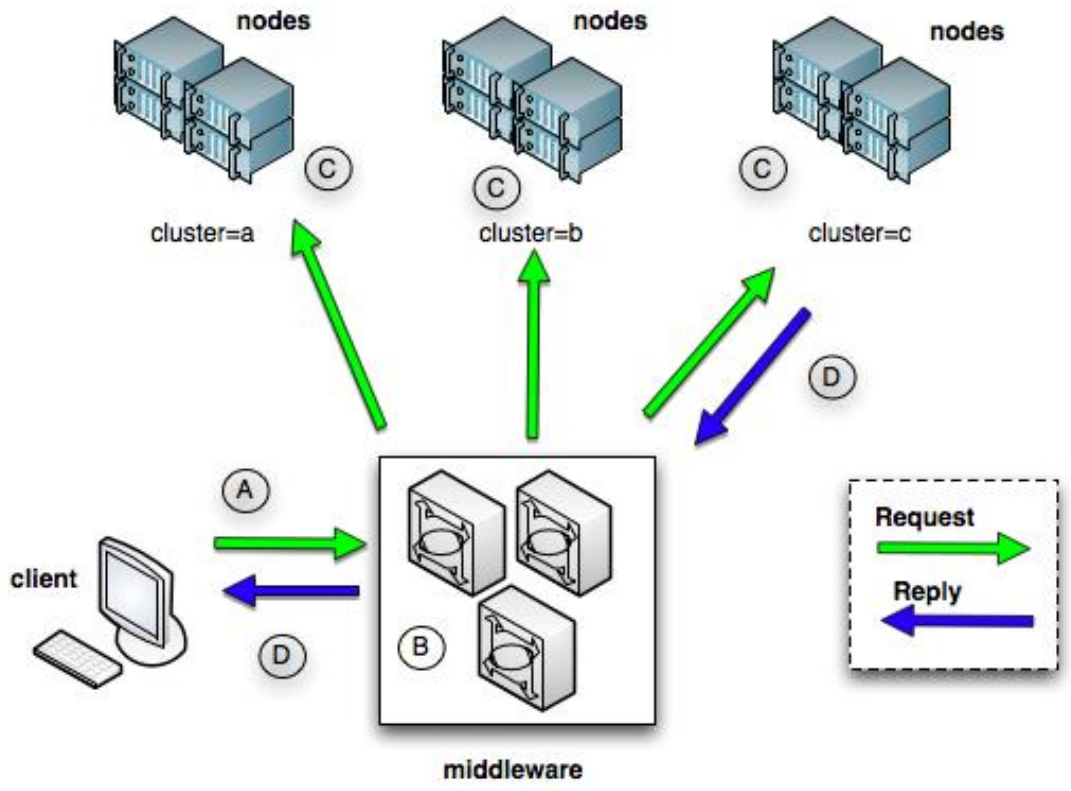
MCollective 就是一个框架，一个空壳。它除了 MCO 命令之外都可以被替换被自定义。
备注：更多信息请参考 http://docs.puppetlabs.com/

1.2 Middleware（RabbitMQ、ActiveMQ）介绍

RabbitMQ 是一个实现了高级消息排队协议（AMQP）的消息队列服务。RabbitMQ 基于 OTP(Open Telecom Platform，开发电信平台)进行构建，并使用 Erlang 语言和运行时环境来实现。 ActiveMQ 是 Apache 出品，最流行的，能力强劲的开源消息总线。 ActiveMQ 是一个完全支持 JMS1.1 和 J2EE 1.4 规范的 JMS Provider 实现

备注：MCollective 是基于 Apache ActiveMQ 中间件来进行开发和测试的，然而其对 java 和 XML 格式的配置文件的依赖使我们将更多的注意力和兴趣转移到 RabbitMQ 中间件服务上。如果考虑到性能和扩展性，部署 ActivemMQ 是一个更好的选择。

1.3 工作原理图



备注：更多详细信息请参考
http://docs.puppetlabs.com/mcollective/reference/basic/messageflow.html

2 安装和配置 Middleware

备注：ActiveMQ 和 RabbitMQ 选其一进行部署即可

2.1 安装和配置 ActiveMQ

2.1.1 安装 ActiveMQ

```
[root@puppetserver rpms]# yum install tanukiwrapper activemq
activemq-info-provider
```

2.1.2 配置 ActiveMQ

主要配置 MCollective 连接的端口以及账号、密码及权限

```
[root@puppetserver rpms]# vim /etc/activemq/activemq.xml
…
<simpleAuthenticationPlugin>
           <users>
<!--             <authenticationUser
username="${activemq.username}"
password="${activemq.password}" groups="admins,everyone"/> -->
#禁用
             <authenticationUser username="mcollective"
password="secret" groups="mcollective,admins,everyone"/>  #配置
通信的账号及密码
           </users>
         </simpleAuthenticationPlugin>
…
<authorizationPlugin>  #配置权限，默认即可
           <map>
             <authorizationMap>
               <authorizationEntries>
                 <authorizationEntry queue=">" write="admins"
read="admins" admin="admins" />
                 <authorizationEntry topic=">" write="admins"
read="admins" admin="admins" />
                 <authorizationEntry topic="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
                 <authorizationEntry topic="mcollective.>"
write="mcollective" read="mcollective" admin="mcollective" />
                 <authorizationEntry
topic="ActiveMQ.Advisory.>" read="everyone" write="everyone"
admin="everyone"/>
               </authorizationEntries>
             </authorizationMap>
           </map>
         </authorizationPlugin>
…
```

```
<transportConnectors>
          <transportConnector name="openwire"
uri="tcp://0.0.0.0:61616"/>
          <transportConnector name="stomp+nio"
uri="stomp://0.0.0.0:61613"/> #配置通信协议为 stomp，监听 61613 端
口
       </transportConnectors>
…
```

2.1.3 启动 ActiveMQ

```
[root@puppetserver rpms]# /etc/rc.d/init.d/activemq start
Starting ActiveMQ Broker...
[root@puppetserver rpms]# chkconfig activemq on
[root@puppetserver rpms]# netstat -nlatp | grep 61613  #查看监
听端口
tcp        0      0 :::61613                    :::*
LISTEN      33805/java
```

备注：更多详细配置信息请参考

http://docs.puppetlabs.com/mcollective/reference/plugins/connector_activemq.html

2.2 安装和配置 RabbitMQ

2.2.1 安装 RabbitMQ

```
[root@puppetserver rpms]# yum install erlang #RabbitMQ 依赖
erlang 语言，需要安装大概 65 个左右的 erlang 依赖包
[root@puppetserver rpms]# yum install rabbitmq-server
[root@puppetserver rpms]# ll
/usr/lib/rabbitmq/lib/rabbitmq_server-3.1.5/plugins/ #默认已经
安装了 stomp 插件，老版本需要下载安装
-rw-r--r-- 1 root root 242999 Aug 24 17:42 amqp_client-
3.1.5.ez
-rw-r--r-- 1 root root  85847 Aug 24 17:42 rabbitmq_stomp-
3.1.5.ez
…
```

2.2.2 启动 rabbitmq-server

```
[root@puppetserver rpms]# /etc/rc.d/init.d/rabbitmq-server
start #启动 rabbitmq 服务
Starting rabbitmq-server: SUCCESS
rabbitmq-server.
[root@puppetserver rpms]# /etc/rc.d/init.d/rabbitmq-server
status #查看 rabbitmq 状态
Status of node rabbit@puppetserver ...
```

```
[{pid,43198},
 {running_applications,[{rabbit,"RabbitMQ","3.1.5"},
                        {mnesia,"MNESIA  CXC 138 12","4.5"},
                        {os_mon,"CPO  CXC 138 46","2.2.7"},
                        {xmerl,"XML parser","1.2.10"},
                        {sasl,"SASL  CXC 138 11","2.1.10"},
                        {stdlib,"ERTS  CXC 138 10","1.17.5"},
                        {kernel,"ERTS  CXC 138 10","2.14.5"}]},
 {os,{unix,linux}},
 {erlang_version,"Erlang R14B04 (erts-5.8.5) [source] [64-bit]
[rq:1] [async-threads:30] [kernel-poll:true]\n"},
 {memory,[{total,27101856},
          {connection_procs,2648},
          {queue_procs,5296},
          {plugins,0},
          {other_proc,9182320},
          {mnesia,57456},
          {mgmt_db,0},
          {msg_index,21848},
          {other_ets,765504},
          {binary,3296},
          {code,14419185},
          {atom,1354457},
          {other_system,1289846}]},
 {vm_memory_high_watermark,0.4},
 {vm_memory_limit,838362726},
 {disk_free_limit,1000000000},
 {disk_free,15992676352},
 {file_descriptors,[{total_limit,924},
                    {total_used,3},
                    {sockets_limit,829},
                    {sockets_used,1}]},
 {processes,[{limit,1048576},{used,122}]},
 {run_queue,0},
 {uptime,4}]
...done.
[root@puppetserver rpms]# netstat -nlp | grep beam #默认监听端口
为 5672
tcp        0      0 0.0.0.0:44422               0.0.0.0:*
LISTEN      43198/beam
tcp        0      0 :::5672                     :::*
LISTEN      43198/beam
```

2.2.3 配置 RabbitMQ

2.2.3.1 加载 amqp_client 和 rabbit_stomp 插件

```
[root@puppetserver sbin]#ln -s
/usr/lib/rabbitmq/lib/rabbitmq_server-3.1.5/sbin/rabbitmq-
plugins /usr/sbin/rabbitmq-plugins #创建命令 rabbitmq-plugins 的
软连接
[root@puppetserver sbin]# ln -s
/usr/lib/rabbitmq/lib/rabbitmq_server-3.1.5/sbin/rabbitmq-env
/usr/sbin/rabbitmq-env #创建命令 rabbitmq-env 的软连接
[root@puppetserver sbin]# rabbitmq-plugins enable
rabbitmq_stomp #开启 rabbitmq_stomp 插件
The following plugins have been enabled:
  amqp_client
  rabbitmq_stomp
Plugin configuration has changed. Restart RabbitMQ for changes
to take effect.
[root@puppetserver sbin]# /etc/rc.d/init.d/rabbitmq-server
restart
Restarting rabbitmq-server: SUCCESS
rabbitmq-server.
[root@puppetserver rabbitmq]# tailf
/var/log/rabbitmq/rabbit\@puppetserver.log #可以从日志看到 stomp
插件加载成功
=INFO REPORT==== 3-Oct-2013::20:25:18 ===
started STOMP TCP Listener on [::]:61613

=INFO REPORT==== 3-Oct-2013::20:25:18 ===
Server startup complete; 2 plugins started.
 * amqp_client
 * rabbitmq_stomp
 *
```

2.2.3.2 创建 rabbitmq.config 配置文件，修改监听端口为 61613

```
[root@puppetserver rpms]# vim /etc/rabbitmq/rabbitmq.config
[
       {stomp,[ {tcp_listeners, [61613]} ]} #设置 connector 为
stomp，监听端口为 61613

].
[root@puppetserver rpms]# /etc/rc.d/init.d/rabbitmq-server
restart
Restarting rabbitmq-server: SUCCESS
rabbitmq-server.
[root@puppetserver rpms]# netstat -nlp | grep beam  #默认监听端
口为 61613
```

```
tcp        0      0 0.0.0.0:56532               0.0.0.0:*
LISTEN      1906/beam.smp
tcp        0      0 :::61613                    :::*
LISTEN      1906/beam.smp
tcp        0      0 :::5672                     :::*
LISTEN      1906/beam.smp
```

2.2.3.3 删除默认账户 guest，为 MCollective 创建账户“mcollective”并设置密码为“secret”，然后设置权限。

```
[root@puppetserver rpms]# rabbitmqctl delete_user guest
Deleting user "guest" ...
...done.
[root@puppetserver rpms]# rabbitmqctl add_user mcollective
secret
Creating user "mcollective" ...
...done.
[root@puppetserver rpms]# rabbitmqctl set_permissions -p "/"
mcollective  ".*" ".*" ".*"
Setting permissions for user "mcollective" in vhost "/" ...
...done.
[root@puppetserver sbin]# rabbitmqctl list_users #查看监听用户
Listing users ...
mcollective []
...done.
```

备注：RabbitMQ 拥有一个默认的 guest 账户，它默认对消息队列拥有全部权限。出于安全方面的考虑，建议删除这个账户。

更多详细配置信息请参考 http://www.rabbitmq.com/admin-guide.html

更多详细配置信息请参考：

http://docs.puppetlabs.com/mcollective/reference/plugins/connector_rabbitmq.html

3 安装和配置 MCollective

3.1 安装 MCollective

3.1.1 测试端安装 MCollective 客户端

```
[root@puppetserver rpms]# yum install mcollective-common
mcollective-client #依赖包 rubygem-stomp
```

3.1.2 节点安装 MCollective 服务端

```
[root@agent1 ~]# yum install mcollective  mcollective-common #
依赖 rubygem-stomp、rubygems 和 ruby 相关包
```

3.2 配置 MCollective

3.2.1 测试端配置 MCollective 客户端

```
[root@puppetserver rpms]# vim /etc/mcollective/client.cfg
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /usr/libexec/mcollective
logger_type = console
loglevel = warn
# Plugins
securityprovider = psk
plugin.psk = a36cd839414370e10fd281b8a38a4f48 #MCollective 通信
共享密钥，和 MCollective 服务端保持一致
connector = stomp  #通信协议
plugin.stomp.host = 192.168.100.110  #Middleware 地址
plugin.stomp.port = 61613  #Middleware 监听端口
plugin.stomp.user = mcollective  #Middleware 通信账号
plugin.stomp.password = secret #Middleware 通信密码

# Facts
factsource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
```

3.2.2 节点配置 MCollective 服务端

```
[root@agent1 rpms]# vim /etc/mcollective/server.cfg
topicprefix = /topic/
main_collective = mcollective
collectives = mcollective
libdir = /usr/libexec/mcollective  #存放 plugins 的位置
logfile = /var/log/mcollective.log
loglevel = info
daemonize = 1
# Plugins
securityprovider = psk
plugin.psk = a36cd839414370e10fd281b8a38a4f48 #MCollective 通信
共享密钥，和 MCollective 客户端保持一致
connector = stomp #通信协议
plugin.stomp.host =  192.168.100.110  #Middleware 地址
plugin.stomp.port =  61613  #Middleware 监听端口
plugin.stomp.user = mcollective  #Middleware 通信账号
plugin.stomp.password = secret #Middleware 通信密码
# Facts
factsource = yaml
```

```
plugin.yaml = /etc/mcollective/facts.yaml
[root@agent1 ~]# /etc/rc.d/init.d/mcollective start
Starting mcollective:
[  OK  ]
[root@agent1 ~]# chkconfig mcollective on
[root@agent1 ~]#
```

3.3 测试 Mcollective 与 Middleware 通信

```
[root@puppetserver rpms]# mco ping  #检查所有存活的节点
agent2.kisspuppet.com                    time=119.98 ms
agent1.kisspuppet.com                    time=159.31 ms

---- ping statistics ----
2 replies max: 159.31 min: 119.98 avg: 139.64
[root@puppetserver rpms]# mco find
agent1.kisspuppet.com
agent2.kisspuppet.com
```

4 Mcollective 插件的安装及测试

MCollective 可以使用多种方式进行扩展。最普遍的一种扩展 MCollective 的方式就是重用已经写好的 agent 插件。这些小的 Ruby 库可以让 MCollective 在整个集群中执行自定义的命令。

一个 agent 插件通常包含一个 Ruby 库，它必须被分发到所有运行 MCollective agent 的节点上。另外，一个数据定义文件（DDL）提供了插件接受的传入参数的具体描述，整个 DDL 文件需要放在 MCollective 客户端系统上。最后，一个使用指定的 agent 插件运行 MCollective 的脚步也需要被安装到所有的 MCollective 客户端系统上。

备注：更多插件可以在 https://github.com/puppetlabs/mcollective-plugins 找到。

4.1 安装 puppet agent 插件

MCollective 本身并不包含一个可以立即使用的 Puppet agent 插件，需要安装使用。这一插件可以让操作员在需要时运行 Puppet agent。他不需要等待 Puppet agent 的默认运行间隔，也不需要使用其他工具来开始这些任务

4.1.1 安装 MCollective 的 Agent 插件

```
[root@agent1 rpms]# yum install mcollective-puppet-agent
mcollective-puppet-common
```

```
[root@agent1 rpms]# ll
/usr/libexec/mcollective/mcollective/agent/
total 36
-rw-r--r-- 1 root root 1033 May 21 01:34 discovery.rb
-rw-r--r-- 1 root root 8346 May 14 07:28 puppet.ddl
-rw-r--r-- 1 root root 7975 May 14 07:25 puppet.rb
-rw-r--r-- 1 root root 5999 May 21 01:34 rpcutil.ddl
-rw-r--r-- 1 root root 3120 May 21 01:34 rpcutil.rb
[root@puppetserver rpms]# yum install mcollective-puppet-
client mcollective-puppet-common
[root@puppetserver rpms]# ll
/usr/libexec/mcollective/mcollective/agent/
total 28
-rw-r--r-- 1 root root 1033 May 21 01:34 discovery.rb
-rw-r--r-- 1 root root 8346 May 14 07:28 puppet.ddl
-rw-r--r-- 1 root root 5999 May 21 01:34 rpcutil.ddl
-rw-r--r-- 1 root root 3120 May 21 01:34 rpcutil.rb
```

4.1.2 载入 Agent 插件

```
[root@puppetserver rpms]# mco  #客户端默认在自动载入
The Marionette Collective version 2.2.4
usage: /usr/bin/mco command <options>
Known commands:
   completion          facts               find
   help                inventory           ping
   plugin              puppet              rpc
Type '/usr/bin/mco help' for a detailed list of commands and
'/usr/bin/mco help command'
to get detailed help for a command
[root@agent1 ~]# /etc/rc.d/init.d/mcollective restart
Shutting down mcollective:
[  OK  ]
Starting mcollective:
[  OK  ]
```

4.1.3 验证 Agent 插件是否被载入

```
[root@puppetserver rpms]# mco inventory agent1.kisspuppet.com
#查看节点 agent1 是否已经载入 puppet 插件
Inventory for agent1.kisspuppet.com:
   Server Statistics:
                  Version: 2.2.4
               Start Time: Thu Oct 03 16:09:03 +0800 2013
              Config File: /etc/mcollective/server.cfg
              Collectives: mcollective
```

```
             Main Collective: mcollective
                  Process ID: 8902
              Total Messages: 3
     Messages Passed Filters: 3
           Messages Filtered: 0
            Expired Messages: 0
                Replies Sent: 2
        Total Processor Time: 0.46 seconds
                 System Time: 0.12 seconds
   Agents:
      discovery       puppet          rpcutil
   Data Plugins:
      agent           fstat           puppet    #已经载入 puppet 插
件
      resource
   Configuration Management Classes:
      No classes applied
   Facts:
      mcollective => 1
```

4.1.4 从 MCollective 中运行 Puppet

在运行命令之前，可以在节点查看 puppet 日志和 puppetd 服务的启停来判断命令是否调用了 puppetd 进程。

```
[root@puppetserver ~]# mco puppet  --noop --verbose status #查
看节点 agent 守护进程状态
Discovering hosts using the mc method for 2 second(s) .... 2

 *
[ ============================================================
> ] 2 / 2

   agent2.kisspuppet.com: Currently stopped; last completed run
9 hours 35 minutes 36 seconds ago
   agent1.kisspuppet.com: Currently stopped; last completed run
9 hours 35 minutes 34 seconds ago

Summary of Applying:

   false = 2

Summary of Daemon Running:

   stopped = 2
```

```
Summary of Enabled:

   enabled = 2
[root@puppetserver rpms]# mco puppet -v runonce
Discovering hosts using the mc method for 2 second(s) .... 2
 *
[ ============================================================
> ] 2 / 2
agent1.kisspuppet.com                       : OK
    {:summary=>      "Started a background Puppet run using the
'puppet agent --onetime --daemonize --color=false --splay --
splaylimit 30' command"}
agent2.kisspuppet.com                       : OK
    {:summary=>      "Started a background Puppet run using the
'puppet agent --onetime --daemonize --color=false --splay --
splaylimit 30' command"}
---- rpc stats ----
          Nodes: 2 / 2
     Pass / Fail: 2 / 0
      Start Time: Thu Oct 03 16:12:03 +0800 2013
  Discovery Time: 2007.23ms
      Agent Time: 3591.72ms
      Total Time: 5598.94ms
```

备注：当使用 MCollective 运行 Puppet 时，要求在所有被管理的节点上 Puppet agent 守护进程都需要被关闭。在每次使用 `mco puppet -v runonce` 命令调用 `puppetd agent` 时，MCollective 都会产生一个新的 Puppet 进程。这个进程会和任何已经运行的 Puppet agent 守护进程产生功能性的重复。

当 Puppet 使用`--runonce` 参数运行时，agent 会在后台运行。所以虽然 MCollective 成功运行了 Puppet，但实际上的 Puppet agent 运行可能 http://kisspuppet.com/2013/11/10/my-fact/并不成功。需要查看 Puppet 报告来确定每一个 Puppet agent 运行的结果。MCollective 返回的 OK 值表示 MCollective 服务器成功地启动了 puppetd 进程并且没有得到任何输出。

4.2 安装 facter 插件（测试多次发现存在不稳定性）

注意：通过 facter 插件获取节点 facter 变量信息不是很稳定，因此可将节点 facts 信息通过 `inline_template` 写入`/etc/mcollective/facts.yaml` 中，并在 `/etc/mcollective/server.cfg` 中设置 `factsource = yaml`，这样 MCollective 客户端只需要每次读取这个文件中的 facter 变量即可。而且在本地目录

/var/lib/puppet/yaml/facts/也会生成一份节点的 facter 信息，模块部分信息如下：

```
class mcollective::facter {
  file{"/etc/mcollective/facts.yaml":
    owner    => root,
    group    => root,
    mode     => 0440,
    loglevel => debug, # reduce noise in Puppet reports
    content  => inline_template('<%= scope.to_hash.reject
{ |k,v| k.to_s =~
/(uptime.*|path|timestamp|free|.*password.*|.*psk.*|.*key)/ }.
to_yaml %>'),
  }
}
```

```
[root@agent1 ~]# yum install mcollective-facter-facts
[root@agent1 rpms]# ll
/usr/libexec/mcollective/mcollective/facts/
total 12
-rw-r--r-- 1 root root  422 Feb 21  2013 facter_facts.ddl
-rw-r--r-- 1 root root  945 Feb 21  2013 facter_facts.rb
-rw-r--r-- 1 root root 1530 May 21 01:34 yaml_facts.rb
```

```
[root@agent1 ~]# vim /etc/mcollective/server.cfg
…
# Facts
#factsource = yaml #注释掉
factsource = facter
plugin.yaml = /etc/mcollective/facts.yaml
[root@agent1 rpms]# /etc/rc.d/init.d/mcollective restart
Shutting down mcollective:
[  OK  ]
Starting mcollective:
[  OK  ]
```

```
[root@puppetserver rpms]# mco inventory agent1.kisspuppet.com
#查看节点 agent1 是否加载了 facts 插件
```

```
Inventory for agent1.kisspuppet.com:

   Server Statistics:
                  Version: 2.2.4
               Start Time: Thu Oct 03 16:31:47 +0800 2013
              Config File: /etc/mcollective/server.cfg
              Collectives: mcollective
          Main Collective: mcollective
               Process ID: 9485
           Total Messages: 37
      Messages Passed Filters: 33
            Messages Filtered: 4
            Expired Messages: 0
                Replies Sent: 32
       Total Processor Time: 0.74 seconds
                System Time: 0.21 seconds

   Agents:
      discovery       puppet          rpcutil

   Data Plugins:
      agent           fstat           puppet
      resource

   Configuration Management Classes:
      No classes applied
   Facts:  #可以看到获取的节点 facter 信息（获取信息需要一些等待时
间）
      architecture => x86_64
      augeasversion => 0.10.0
      bios_release_date => 07/02/2012
      bios_vendor => Phoenix Technologies LTD
      bios_version => 6.00
      blockdevice_fd0_size => 4096
     …
      uptime_days => 0
      uptime_hours => 20
      uptime_seconds => 74506
      uuid => 564DFBAB-CADC-FC69-36CA-955BFDB30F43
      virtual => vmware
```

```
[root@puppetserver rpms]# mco facts lsbdistdescription -v  #使
用 mco facts 命令对操作系统类型进行显示
Discovering hosts using the mc method for 2 second(s) .... 2
Report for fact: lsbdistdescription
       Red Hat Enterprise Linux Server release 5.7
(Tikanga)found 1 times
           agent2.kisspuppet.com
       Red Hat Enterprise Linux Server release 5.8
(Tikanga)found 1 times
           agent1.kisspuppet.com
---- rpc stats ----
          Nodes: 2 / 2
     Pass / Fail: 2 / 0
      Start Time: Thu Oct 03 16:59:04 +0800 2013
  Discovery Time: 2004.83ms
      Agent Time: 67.32ms
      Total Time: 2072.15ms
```

```
[root@puppetserver rpms]# mco facts lsbdistdescription  #使用
mco facts 命令对操作系统类型进行统计
Report for fact: lsbdistdescription
       Red Hat Enterprise Linux Server release 5.7
(Tikanga)found 1 times
       Red Hat Enterprise Linux Server release 5.8
(Tikanga)found 1 times
Finished processing 2 / 2 hosts in 79.15 ms
[root@puppetserver rpms]# mco facts -v --with-fact
hostname='agent1' memoryfree #查看主机 agent1 的剩余内存
Discovering hosts using the mc method for 2 second(s) .... 1
Report for fact: memoryfree
       795.13 MB                              found 1 times
           agent1.kisspuppet.com
---- rpc stats ----
          Nodes: 1 / 1
     Pass / Fail: 1 / 0
      Start Time: Thu Oct 03 17:02:13 +0800 2013
  Discovery Time: 2005.65ms
      Agent Time: 49.37ms
      Total Time: 2055.03ms
```

4.3 使用元数据定位主机

4.3.1 使用默认 facter 元数据定位主机

4.3.1.1 触发所有系统为 RedHat，版本为 5.7 的所有节点 puppetd 守护进程

```
[root@puppetserver rpms]# mco puppet -v runonce   rpc --np -F
operatingsystemrelease='5.7' -F operatingsystem='RedHat'
Discovering hosts using the mc method for 2 second(s) .... 1
agent2.kisspuppet.com                    : OK
    {:summary=>      "Started a background Puppet run using the
'puppet agent --onetime --daemonize --color=false --splay --
splaylimit 30' command"}
---- rpc stats ----
          Nodes: 1 / 1
     Pass / Fail: 1 / 0
      Start Time: Thu Oct 03 17:03:56 +0800 2013
  Discovery Time: 2008.09ms
      Agent Time: 1187.69ms
      Total Time: 3195.78ms
```

4.3.1.2 触发所有系统为 RedHat，kernel 版本为 2.6.18 的所有节点 puppetd 守护进程

```
[root@puppetserver rpms]# mco puppet -v runonce   rpc --np -F
kernelversion='2.6.18'  -F operatingsystem='RedHat'
Discovering hosts using the mc method for 2 second(s) .... 2
agent2.kisspuppet.com                    : OK
    {:summary=>      "Started a background Puppet run using the
'puppet agent --onetime --daemonize --color=false --splay --
splaylimit 30' command"}
agent1.kisspuppet.com                    : OK
    {:summary=>      "Started a background Puppet run using the
'puppet agent --onetime --daemonize --color=false --splay --
splaylimit 30' command"}
---- rpc stats ----
          Nodes: 2 / 2
     Pass / Fail: 2 / 0
      Start Time: Thu Oct 03 17:06:15 +0800 2013
  Discovery Time: 2004.32ms
      Agent Time: 1308.34ms
      Total Time: 3312.66ms
```

4.3.2 使用自定义 facter 元数据定位主机

备注：使用自定义 facter 元数据可以更加灵活的定位主机，如何定义 fact 可参考博文《通过自定义 fact 增强 MCollective 推送更新元数据的灵活性》

4.3.2.1 在 agent1 上定义 facter my*apply1* 和 my*apply2*

```
[root@agent1 mcollective]# facter -p | grep my_apply
my_apply1 => apache
my_apply2 => mysql
```

4.3.2.2 在 agent2 上定义 facter my*apply2 和 my*apply3

```
[root@agent2 mcollective]# facter -p | grep my_apply
my_apply2 => mysql
my_apply3 => php
```

4.3.2.3 在 MCollective 客户端测试节点自定义 facter 是否正确

```
[root@puppetserver facter]# mco inventory
agent1.kisspuppet.com  | grep my_apply
     my_apply1 => apache
     my_apply2 => mysql
[root@puppetserver facter]# mco inventory
agent2.kisspuppet.com  | grep my_apply
     my_apply2 => mysql
     my_apply3 => php
```

4.3.2.4 通过自定义 facter 定位主机触发更新

```
[root@puppetserver facter]# mco puppet -v runonce  mco facts -v
--with-fact  my_apply3='php' #筛选节点 facter 变量 my_apply3=php
的主机进行触发 puppetd 守护进程
Discovering hosts using the mc method for 2 second(s) .... 1

 *
[ ============================================================
> ] 1 / 1
agent2.kisspuppet.com                    : OK
    {:summary=>     "Started a background Puppet run using the
'puppet agent --onetime --daemonize --color=false --splay --
splaylimit 30' command"}
---- rpc stats ----
          Nodes: 1 / 1
     Pass / Fail: 1 / 0
      Start Time: Thu Oct 03 23:33:54 +0800 2013
  Discovery Time: 2005.35ms
      Agent Time: 1078.86ms
      Total Time: 3084.21ms
```

为了能够和大家更好的交流和学习 Puppet，本人 2014 年又新开辟了微信公众号进行交流学习，目前已经有 300 多人同时收听，喜欢 Puppet 的大神们可自行加入哦。

如果你有好的有关 Puppet 的咨询也可以给我投稿，投稿地址：admin@kisspuppet.com

微信公众号：“puppet2014”，可搜索加入，也可以扫描以下二维码