MCOLLECTIVE SERVER CONFIG

概述

此文档描述 MCollective 2.0.0 及以后的 MCollective Server 配置，之前版本的 MCollective 可能会缺少某些特性。

主配置文件

默认情况下，MCollective Server 的配置文件是`/etc/mcollective/server.cfg`。它包含 MCollective 的核心设置以及不同种类插件的设置。

此文件包含敏感证书信息，应该确保 root 用户或者是能够运行此 deamon 的用户可读。

文件格式

每行由一组`Key=Value`设置组成

单行注释，以#开头，等号前后空格可选

```
# setting = value
connector = activemq
```

对于布尔值：MCollective 的配置代码没有一致的布尔值处理方式，大多数核心设置接受 1/0 和 y/n，但是都不会接受 true/false。另外，每个插件处理布尔值方式不同，其中一些不能很好处理核心设置接受的 y/n 设置。

插件配置文件

MCollective 的很多配置命名为`plugin.<NAME>.<SETTING_NAME>`。这些设置可以可选的放在单独的文件里面，这些单独文件在`/etc/mcollective/plugin.d/`目录下。

然后将该插件设置移到外部配置文件`/etc/mcollective/plugin.d/<NAME>.cfg`中。就可以只使用`<SETTING_NAME>`段作为设置名称。如下：

```
# /etc/mcollective/server.cfg
plugin.puppet.splay = true
```

等价设置：

```
# /etc/mcollective/plugin.d/puppet.cfg
splay = true
```

注意这个设置对 `plugin.psk` 无效。因为它没有`<SETTING_NAME>`段。一个设置至少有三段才能将其放置在 `plugin.cfg` 的文件里。

最佳实践

对于 MCollective 配置文件，可以使用像 Puppet 这样的配置管理进行管理。虽然部署环境中大多数机器是相同的，但是部分机器还是相互不同，而要手动维护这些差异比较困难。因此建议使用 Puppet 对配置文件进行管理。

如果你的部署是简单的而且也没有太多的分支机构来进行管理，比如一个组管理 MCollective 核心另一组管理部分代理插件。这种情况可以使用一个简单的模板来进行管理配置文件。

如果你的部署环境较复杂，就需要将 MCollective 设置作为一个个资源单独进行管理，这是在单个文件下划分责任的唯一可行方式。

下面的例子展示了如何使用 Puppet 的 file_inline 类型来管理 MCollective 配置文件：

```
    # /etc/puppet/modules/mcollective/manifests/setting.pp
    define mcollective::setting ($setting = $title, $target =
'/etc/mcollective/server.cfg', $value) {
      validate_re($target, '\/(plugin\.d\/[a-z]+|server)\.cfg\Z')
      $regex_escaped_setting = regsubst($setting, '\.', '\\.', 'G') # assume
dots are the only regex-unsafe chars in a setting name.

      file_line {"mco_setting_${title}":
        path  => $target,
        line  => "${setting} = ${value}",
        match => "^ *${regex_escaped_setting} *=.*$",
      }
    }

    # /etc/puppet/modules/mcollective_core/manifests/server/connector.pp
    # ...
    # Connector settings:
    mcollective::setting {
      'connector':
        value => 'activemq';
      'direct_addressing':
        value => '1';
      'plugin.activemq.pool.size':
        value => '1';
      'plugin.activemq.pool.1.host':
        value => $activemq_server;
      'plugin.activemq.pool.1.port':
```

```
      value => '61614';
    'plugin.activemq.pool.1.user':
      value => $activemq_user;
    'plugin.activemq.pool.1.password':
      value => $activemq_password;
    'plugin.activemq.pool.1.ssl':
      value => '1';
    'plugin.activemq.pool.1.ssl.fallback':
      value => '1';
  }
  # ...
```

完整示例

下面的例子展示了一个完整的 MCollective Server 配置文件所有重要设置。

```
# /etc/mcollective/server.cfg

# Connector settings (required):
# ------------------------------

connector = activemq
direct_addressing = 1

# ActiveMQ connector settings:
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = middleware.example.net
plugin.activemq.pool.1.port = 61614
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = secret
plugin.activemq.pool.1.ssl = 1
plugin.activemq.pool.1.ssl.ca = /var/lib/puppet/ssl/certs/ca.pem
plugin.activemq.pool.1.ssl.cert =
/var/lib/puppet/ssl/certs/web01.example.com.pem
plugin.activemq.pool.1.ssl.key =
/var/lib/puppet/ssl/private_keys/web01.example.com.pem
plugin.activemq.pool.1.ssl.fallback = 0

# RabbitMQ connector settings:
plugin.rabbitmq.vhost = /mcollective
plugin.rabbitmq.pool.size = 1
plugin.rabbitmq.pool.1.host = middleware.example.net
# ... etc., similar to activemq settings
```

```
# Security plugin settings (required):
# ------------------------------------

securityprovider = ssl

# SSL plugin settings:
plugin.ssl_client_cert_dir = /etc/mcollective.d/clients
plugin.ssl_server_private = /etc/mcollective.d/server_private.pem
plugin.ssl_server_public = /etc/mcollective.d/server_public.pem

# PSK plugin settings:
plugin.psk = j9q8kx7fnuied9e

# Facts, identity, and classes (recommended):
# -------------------------------------------

factsource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
fact_cache_time = 300

identity = web01.example.com

classesfile = /var/lib/puppet/state/classes.txt

# Subcollectives (optional):
# --------------------------

collectives = mcollective,uk_collective
main_collective = mcollective

# Registration (optional):
# ------------------------

registerinterval = 300
registration = agentlist
registration_collective = mcollective

# Auditing (optional):
# --------------------

rpcaudit = 1
rpcauditprovider = logfile
plugin.rpcaudit.logfile = /var/log/mcollective-audit.log
```

```
# Authorization (optional):
# -------------------------

rpcauthorization = 1
rpcauthprovider = action_policy

# Logging:
# -------

logger_type = file
loglevel = info
keeplogs = 5
max_log_size = 2097152
logfacility = user

# Platform defaults:
# ------------------

daemonize = 1
libdir = /usr/libexec/mcollective
ssl_cipher = aes-256-cbc
```

必须配置

connector plugin 和 security plugin 是必须配置。

CONNECTOR 设置

示例配置如下：

```
connector = activemq
direct_addressing = 1

# ActiveMQ connector settings:
plugin.activemq.pool.size = 1
plugin.activemq.pool.1.host = middleware.example.net
plugin.activemq.pool.1.port = 61614
plugin.activemq.pool.1.user = mcollective
plugin.activemq.pool.1.password = secret
plugin.activemq.pool.1.ssl = 1
plugin.activemq.pool.1.ssl.ca = /var/lib/puppet/ssl/certs/ca.pem
```

```
plugin.activemq.pool.1.ssl.cert =
/var/lib/puppet/ssl/certs/web01.example.com.pem
plugin.activemq.pool.1.ssl.key =
/var/lib/puppet/ssl/private_keys/web01.example.com.pem
plugin.activemq.pool.1.ssl.fallback = 0

# RabbitMQ connector settings:
plugin.rabbitmq.vhost = /mcollective
plugin.rabbitmq.pool.size = 1
plugin.rabbitmq.pool.1.host = middleware.example.net
plugin.rabbitmq.pool.1.port = 61613
# ... etc., similar to activemq settings
```

MCollective 总是需要 connector 插件，connector 插件由部署所选用中间件所决定，每个插件会有相应的所需要的设置。

共享配置

所有的 servers 和 clients 都必须配置相同的 connector 插件，而且它们的配置必需兼容。

使用所选中间件的 connector 插件

主机名和端口必须匹配所使用的中间件。用户名和密码必须能够正确登入中间件。如果使用 CA-verified TLS，证书必须是由中间件所使用的相同 CA 签名。

```
connector
```

使用哪个 connector 插件，这由使用的中间件决定。

默认：activemq

允许值：activemq,rabbitmq 或者其他第三方 connector 插件的名称。注意插件名称大写没有关系，MCollective 会在加载时标准化。

```
direct_addressing
```

无论你的中间件是否支持直接点到点消息。这通常应该被设置启用，但默认却是未启用。内建的 activemq 和 rabbitmq connector 都支持直接寻址，redis connector 也支持。但是旧的 stomp 不支持。

默认值：0

允许值：0,1,y,n。注意不要使用 true 或 false。

ActiveMQ Connector 设置

ActiveMQ 是 MCollective 主要推荐使用的中间件。可以使用多个 ActiveMQ connector 作为 failover pool。如果只有一个 server，那么设置 pool size 为 1。下面列举部分设置，详细设置请参见 Apache ActiveMQ 官方文档。

plugin.activemq.pool.size – How many ActiveMQ servers to attempt to use. *Default:* (nothing)
plugin.activemq.pool.1.host – The hostname of the first ActiveMQ server. (Note that additional servers use the same settings as the first, incrementing the number.) *Default:* (nothing)
plugin.activemq.pool.1.port – The Stomp port of the first ActiveMQ server. *Default:* 61613 or6163, depending on the MCollective version.
plugin.activemq.pool.1.user – The ActiveMQ user account to connect as. If the STOMP_USERenvironment variable is set, MCollective will use its value instead of this setting.
plugin.activemq.pool.1.password – The password for the user account being used. If theSTOMP_PASSWORD environment variable is set, MCollective will use its value instead of this setting.
plugin.activemq.pool.1.ssl – Whether to use TLS when connecting to ActiveMQ. *Default:* 0;*allowed:* 1/0, true/false, yes/no
plugin.activemq.pool.1.ssl.fallback – *(When ssl == 1)* Whether to allow unverified TLS if the ca/cert/key settings aren't set. *Default:* 0; *allowed:* 1/0, true/false, yes/no
plugin.activemq.pool.1.ssl.ca – *(When ssl == 1)* The CA certificate to use when verifying ActiveMQ's certificate. Must be the path to a .pem file. *Default:* (nothing)
plugin.activemq.pool.1.ssl.cert – *(When ssl == 1)* The certificate to present when connecting to ActiveMQ. Must be the path to a .pem file. *Default:* (nothing)
plugin.activemq.pool.1.ssl.key – *(When ssl == 1)* The private key corresponding to this node's certificate. Must be the path to a .pem file. *Default:* (nothing)

RabbiMQ connector 设置

Rabbitmq 使用类似 activemq connector 的设置，同样使用.pool.1.host 这样的配置名称。

安全插件设置

示例配置如下：

```
securityprovider = ssl

# SSL plugin settings:
plugin.ssl_client_cert_dir = /etc/mcollective.d/clients
plugin.ssl_server_private = /etc/mcollective.d/server_private.pem
plugin.ssl_server_public = /etc/mcollective.d/server_public.pem

# PSK plugin settings:
plugin.psk = j9q8kx7fnuied9e
```

MCollective 总是需要使用安全插件（虽然它们被称为安全插件，它们实际处理地更多，包括消息序列化）每个插件都有些其他配置。

共享配置
所有的 servers 和 clients 都必须使用相同的安全插件，而且它们的配置需要兼容。

可以写新的自己的安全插件，但是大多数人都使用下列三者之一：

SSL：大多数用户最好的选择，当结合 connector 插件上的 TLS 时提供很好的安全性

PSK：安全性很弱，但是容易配置，对于概念验证的部署很好

AES：配置很复杂，而且在大型网络环境下带来很大的性能消耗。只符合特定用例，比如中间件上不能配置 TLS

对于 MCollective 整个安全性配置，请参考《MCollective 安全综述》文档。

`securityprovider`

使用哪个安全插件

默认：psk

允许值：ssl, psk, aes_security,或者任何一个第三方安全插件的名称。注意插件名称大写没有关系，MCollective 会在加载时标准化。

SSL 安全插件配置

注意：安全插件需要管理和分发 SSL 证书。大致来看：
所有的 servers 共享一个“server”密钥对。它们所有必须拥有一份公钥和私钥的拷贝
所有的 admin 用户都必须拥有一个 server 公钥
所有的 admin 用户拥有它们自己的 client 密钥对
所有的 servers 必须拥有一份每个被授权 client 公钥的拷贝

所有的设置没有默认值，而且都必须设置 SSL 才能工作

plugin.ssl_server_private – The path to the server private key file, which must be in .pemformat.
plugin.ssl_server_public – The path to the server public key file, which must be in .pem format.
plugin.ssl_client_cert_dir – A directory containing every authorized client public key.

PSK 安全插件配置

注意：只有使用此插件的证书使用单个共享密码，所有的 servers 和 admin 用户都拥有一份拷贝
plugin.psk – The shared passphrase. If the MCOLLECTIVE_PSK environment variable is set, MCollective will use its value instead of this setting.

示例配置如下：

```
factsource = yaml
plugin.yaml = /etc/mcollective/facts.yaml
fact_cache_time = 300

identity = web01.example.com

classesfile = /var/lib/puppet/state/classes.txt
```

MCollective clients 使用过滤来发现节点信息和限制命令使用范围。这些过滤能使用每个机器上的大量元数据信息，包括 facts, identity 和 classes。

Facts：关于机器软硬件信息的 key/value 数据集合

Identity：节点名称

Classes：应用到此节点的 Puppet Classes 或者 chef cookbooks。Classes 非常有用是因为它们描述了此节点的 roles 信息

这些设置都不是必须的，但是却不能少了它们。

identity

节点名或者 identity。对于所每个节点需要唯一，但是不是必须的

默认值：ruby 方法 Socket.gethostname 的返回值，这通常是 server 的 FQDN

允许值：匹配正则表达式：`/\A[\w\.\-]+\Z/`

classesfile

配置管理工具应用的一个文件，每行包含一个此节点的 class 信息。比如 Puppet 能够自动写一个 class 文件。

默认值：`/var/lib/puppet/state/classes.txt`

factsource

使用哪个 fact 插件

MCollective 包含一个 fact 插件称为 yaml。大多数使用默认配置，其他 fact 插件包括 Facter 和 Ohai 等位于 plugin 目录下。

默认值：yaml

允许值：安装的 fact 插件的名称。注意名称与大小写无关。

```
plugin.yaml
```

当 factsource==yaml 时

fact file 加载默认 yaml fact 插件

默认值：(nothing)

参考值：`/etc/mcollective/facts.yaml`

有效值：单个文件路径，或者多个路径以(冒号：linux)/(分号；windows)隔开

注意：默认的 yaml fact 插件从一个文件红读取缓存，通常应该是一个简单的 YAML 哈希。如果指定多个文件，它们会被合并，如果有值冲突后面的值会覆盖前面的值

默认 facts.yaml 文件时空的，可将 Facter 信息导入进来，使用 Puppet 实现：

```
# /etc/puppet/manifests/site.pp
    file{"/etc/mcollective/facts.yaml":
       owner    => root,
       group    => root,
       mode     => 400,
       loglevel => debug, # reduce noise in Puppet reports
       content  => inline_template("<%= scope.to_hash.reject { |k,v| k.to_s
=~ /(uptime_seconds|timestamp|free)/ }.to_yaml %>"), # exclude rapidly
changing facts
    }
```

```
fact_cache_time
```

fact 结果缓存时间(以秒计)，过期后会从 fact 文件中重新读取

默认值：300

可选特性

SUBCOLLECTIVES

示例配置如下：

```
collectives = mcollective,uk_collective
main_collective = mcollective
```

Subcollectives 为部署提供了另一种划分服务器集群方式。这种方式很有用，因为中间件能够知道它们，通过它们能够启用流量控制和访问控制。在多数据中心部署环境下，这能够节省带宽成本和提供额外安全。

Subcollective 成员被管理在 server 端，通过每个 server 的配置文件。一个指定的 server 可以加入任何 collectives，而且也会响应任何来自于此 collective 的命令。

共享配置
如果使用 subcollective(除了默认的 mcollective collective 之外)，中间件必须被配置来允许这些 collective 之间的流量传输。
client 端 subcollective 的配置是表示此 client 能够允许连接的 subcollectives，server 端配置 subcollectives 是表示此节点属于哪些 subcollectives。

collectives

逗号隔开的参数列表，表示此 server 属于的 subcollectives

默认值：mcollective

参考值：mcollective,　uk_collective

main_collective

此节点的 main collective。目前，这个设置仅用于 registration_collective 设置的默认值

默认值：collectives 设置的第一个值，通常是 mcollective

高级配置

LOGGING

示例配置如下：

```
logger_type = file
loglevel = info
logfile = /var/log/mcollective.log
keeplogs = 5
max_log_size = 2097152
logfacility = user
```

MCollective Server Daemon 会向自己的日志文件记录 log 日志或者使用 syslog。如果你以前台任务运行，它也可以直接向控制台输出日志。

如果使用 log 文件记录日志，下面的配置也许会使用到，其中一些是使用 syslog。

logger_type

MCollective Server Daemon 记录 log 的方式

默认值：file

允许值：file，syslog，console

loglevel

记录 log 的等级

默认值：info

允许值：按顺序依次是：fatal，error，warn，info，debug

logfile

如果 logger_type==file 时记录日志的文件

默认值：(nothing)

参考值：`/var/log/mcollective.log`

keeplogs

当 logger_type==file 时 log 文件发生旋转的最大字节

默认值：2097152

logfacility

当 logger_type==syslog 时使用 syslog 的设施

默认值：user

PLATFORM DEFAULT(平台默认)

示例配置如下：

```
daemonize = 1
libdir = /usr/libexec/mcollective
ssl_cipher = aes-256-cbc
```

这些设置通常不应该由用户所修改，但是它们的值根据平台不同而有所不同。安装 MCollective 时会创建一个配置文件包含这些平台相关的配置选项及其值。

daemonize

是否在后台 fork 并且 run MCollective Server Daemon

这取决于平台初始化系统，例如最新的 Ubuntu 发行版需要设置此项为 False，而 RHEL-drived 系统需要设置此项为 True

默认值：0

允许值：1,0，y，n

libdir

插件目录，一般是一个单个目录或者一系列目录(冒号隔开*nix/分号隔开 windows)

如果 Server.cfg 中此设置没有值，那么 MCollective 将无法工作。

默认值：(nothing，同样在软件安装时默认配置文件下通常设置有此项平台相关值)

参考值：`/usr/libexec/mcollective:/opt/mcollective`

ssl_cipher

用于加密的密码，这与使用 AES 安全插件相关。这个设置应该是个标准 OpenSSL 密码字符串。

默认值：`aes-256-cbc`