# Puppet+MCollective 説明
# 参考資料

# 目次

# 1はじめに

この説明書は、本演習で用いた Puppet と MCollective の設定を説明するための参考資料です。
Puppet と MCollective を使って、本演習のシステムを自動構成するための設定を説明します。

# 2前提条件

1. 各々サーバに Puppet と MCollective が稼働していること
2. Puppet の設定手順が実行されていること
3. MCollective の設定手順が実行されていること

設定対象ファイル
PuppetMaster
　/etc/puppet/manifests/site.pp

PuppetAgent
　/etc/default/puppet

MCollectiveClient
　/etc/mcollective/client.cfg

MCollectiveServer
　/etc/mcollective/server.cfg

# 3Puppet Master の設定

Puppet は独自の宣言型言語によってシステムの構成を記述します。この言語で書かれたシステム構成ファイルのことを「マニフェスト」と呼びます。ファイルの拡張子に.pp を指定したテキストファイルがマニフェストファイルとなります。マニフェストには大きく分けて「リソース」とリソースをどのサーバで構成するのかを指定する「クラス」を記述します。

Puppet の管理対象を「リソース」と呼びます。このリソースを複数まとめた構成を「クラス」と呼びます。以下ではリソースとクラスについて説明します。
（参考: http://docs.puppetlabs.com/guides/language_guide.html ）

## 3.1.リソース定義

ここでは標準で定義されている File リソースと Exec リソースを使い設定をおこないます。

最初の「file」、「exec」は、リソースの種類をあらわすリソースタイプです。続いて、ダブルクォーテーションで囲まれている値は、実行先の対象リソースです。File リソースタイプは owner,group,ensure,mode,content、Exec リソースタイプは cwd,path がリソース固有のパラメータです。

設定ファイル

```
deploy# /etc/puppet/manifests/site.pp
```

### 3.1.1.File リソース

PuppetMaster にある、設定ファイルをコピーして構成をする。

設定項目：Gamglia-monitor のデフォルト設定ファイルを定義する。

```
file{"/etc/ganglia/gmond.conf":
  owner   => root,
  group   => root,
  ensure  => file,
  mode    => 644,
  content => template('/etc/puppet/templates/etc/ganglia/gmond.conf.erb')
}
```

設定項目：Mcollective の facts ファイルを定義する。

```
file{"/etc/mcollective/facts.yaml":
            owner    => root,
            group    => root,
            mode     => 400,
            loglevel => debug
            content  => inline_template("<%= scope.to_hash.reject { |k,v| k.to_s
=~ /(uptime_seconds|timestamp|free)/ }.to_yaml %>")
}
```

## 3.1.2.設定項目の説明

owner:
　　ファイルのオーナーを指定します。

group:
　　ファイルのグループを指定します。

ensure:
　　PuppetMaster から管理対象となるファイルどのような形で存在するか指定します。
　　file の他にシンボリックリンクを指定する link やディレクトリを指定する directory 設定などがあります。

mode:
　　ファイルの読み書きモードを指定します。

content:
　　PuppetMaster にある、構成する設定ファイルの元ファイルを指定します。
　　Ruby の ERB 形式でテンプレート化し、テンプレートにあてはめる値をマニフェスト側で指定することもできます。

## 3.1.3.Exec リソース

PuppetAgent でコマンドを実行して構成をする

設定項目:keijiban の war の配布

```
exec{"scp -r -oStrictHostKeyChecking=no
deploy.nii.localdomain:/var/tmp/keijiban.war
/var/lib/tomcat6/webapps/keijiban.war":
        cwd => "/var/lib/tomcat6/webapps/",
        path => ["/usr/bin", "/usr/sbin/"]
}
```

## 3.1.4.設定項目の説明

cwd:
　　実行する場所を指定します。

path:
　　実行するコマンドの場所を指定します。

## 3.2.クラス定義

複数のリソースを束ねたクラスは class 文で定義します。

設定項目:Ganglia-monitor のクラス定義

```
class gmond {
  file{"/etc/ganglia/gmond.conf":
    owner   => root,
    group   => root,
    ensure  => file,
    mode    => 644,
    content => template('/etc/puppet/templates/etc/ganglia/gmond.conf.erb')
  }
}
```

クラスを読む込むときは include 文を用いて設定します。

設定項目:クラスの読み込み

```
include gmond
```

## 3.3.define 定義

引数として値を渡す場合は define 文で定義します。

設定項目:Nginx のクラス定義

```
define nginx::register($content) {
  $ipset = $content

  file{"/etc/nginx/sites-available/default":
    owner   => root,
    group   => root,
    ensure  => file,
    mode    => 644,
    content => template('/etc/puppet/templates/etc/nginx/nginx.conf.erb')
  }
}
```

define を読む込むに名前空間と引数を設定することができます。

設定項目:define の読み込み

```
nginx::register{"Nginx": content => $nginx_ipset }
```

## 3.4.変数

マニフェスト内部で動的に変わる値は先頭に$をつけた名前を指定することで変数を定義することができます。変数は自分で宣言して使う変数と予約された変数があります。

自分で宣言した変数

```
$cluster_name = “Web”
```

予約された変数

```
$hostname
```

予約された変数は facter コマンドを Puppet Agent が起動しているサーバで実行することによって閲覧できます。

```
deploy# facter

architecture => x86_64
domain => nii.localdomain
facterversion => 1.5.9
fqdn => deploy.nii.localdomain
hardwareisa => x86_64
hardwaremodel => x86_64
hostname => deploy
…以下省略
```

## 3.5.条件分岐

マニフェストの中で、if や case 文を利用することができます。case 文の条件に変数を利用することで、ホスト別に必要なクラスのみを読む設定を書くことができます。
Puppet Agent が facter の情報を Puppet Master に送信し、Puppet Master がマニフェストの設定にしたがって構成情報を作成するため、LB、Web、DB 別に構成を分ける使い方に利用できます。

設定項目:条件分岐と変数名を使いクラスの読み込みを変更する。

```
case $hostname {
          'lb': {
             include nginx
             include mcollective_server
          }
      'web': {
            include war_file
            include mcollective_server
          }
      'db': {
            include mcollective_server
          }
}
```

## 3.6.ERB

ERB(eRuby、embedded Ruby)とは、Ruby の周辺技術の一つで、HTML や任意のプレインテキストへ Ruby スクリプトを埋め込む事を可能とする技術である。

下の様な形式で Ruby のコードを記述することができる

```
<% (Ruby のコード) %>
```

また、下の様な形式で記述すると、変数やコードの文字列表現をテキストに埋め込むことができる

```
<%= (Ruby の変数) %>
<%= (Ruby のコード) %>
```

例)

```
<% list = ["aa", "bb"] -%>
<% list.each do |element| -%>
<%= (Ruby のコード) %>
<% end -%>
```

結果)

```
aa
bb
```

### 3.6.1.Puppet における ERB

template リソースによって指定されたファイルが erb 形式ならば、Puppet Agent のリクエスト情報から Puppet Master で Ruby を実行し、結果を反映したファイルで Puppet Agent に応答する。

# 4Puppet Agent の設定

PuppetAgent の設定をします。
PuppetAgent は puppetd コマンドを実行することでデーモンとして起動し PuppetMaster にマニフェストの取得を依頼します。また、MCollectiveAgent の Puppet プラグインを利用することで、MColleciveClient からのリクエストを受け付けます。

設定ファイル

```
deploy# vi /etc/default/puppet
```

設定項目:PuppetAgent の設定をする。

```
# Start puppet on boot?
START=yes

# Startup options
DAEMON_OPTS="--server deploy.nii.localdomain --
logdest=/var/log/puppet/puppet.log -d"
```

## 4.1.設定項目の説明

START
yes を指定することで、Boot 時に自動的に PuppetAgent が起動します。

DAEMON_OPTS
puppet デーモンが起動するときのオプションを指定します。
--server は PuppetMaster のアドレスを指定します。
--logdest は PuppetAgent のログの場所を指定します。
-d はデバック情報を表示します。

# 5MCollectiveClient の設定

MCollectiveClient の設定をします。
MCollectieServer にインストールされている MColletiveAgent の「pupptd」を使うことにより MCollective と Puppet を連携し、システム構成の自動化を行うことができます。

設定ファイル

```
deploy# vi /etc/mcollective/client.cfg
```

設定項目:メッセージキューサーバを設定する。

```
connector = stomp
plugin.stomp.host = localhost
plugin.stomp.port = 61613
plugin.stomp.user = mcollective
plugin.stomp.password = secret
```

## 5.1.設定項目の説明

connector
　　プラグインのコネクタ名を指定します。

plugin.stomp.host
　　メッセージキューサーバのホストを指定します。
　　今回は Monitor サーバとメッセージキューサーバが同じサーバにあるため、localhost を指定しています。

plugin.stomp.port
　　メッセージキューサーバのポートを指定します。

plugin.stomp.user
　　メッセージキューサーバのユーザーを指定します。
　　メッセージキューサーバに StompServer を使う場合はなにを指定してもよいです。
　　メッセージキューサーバに ActiveMQ を指定する場合は必要です。

plugin.stomp.password
　　メッセージキューサーバのパスワードを指定します。
　　メッセージキューサーバに StompServer を使う場合はなにを指定してもよいです。
　　メッセージキューサーバに ActiveMQ を指定する場合は必要です。

# 6MCollectiveServer の設定

MCollectiveServer の設定をします。
以下では、MCollectiveClient から PuppetAgent を実行する仕掛の設定の説明をします。

設定ファイル

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

設定項目:メッセージキューサーバを設定する。

```
connector = stomp
plugin.stomp.host = deploy.nii.localdomain
plugin.stomp.port = 61613
plugin.stomp.user = mcollective
plugin.stomp.password = secret
```

## 6.1.設定項目の説明

connector
　　プラグインのコネクタ名を指定します。

plugin.stomp.host
　　メッセージキューサーバのホストを指定します。

plugin.stomp.port
　　client.cfg と同じ

plugin.stomp.user
　　client.cfg と同じ

plugin.stomp.password
　　client.cfg と同じ

# 6.2.MCollective Plugin の設定

## 6.2.1.MCollectiveServer+PuppetClient の設定

MCollectiveServer に PuppetServer の参照先を設定します。

設定ファイル

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

設定項目:MCollective と Puppet を連携する設定をする。

```
plugin.puppetd.puppetd = /usr/sbin/puppetd
plugin.puppetd.lockfile = /var/lib/puppet/state/puppetdlock
plugin.puppetd.statefile = /var/lib/puppet/state/state.yaml
plugin.puppet.pidfile = /var/run/puppet/agent.pid
plugin.puppet.summary = /var/lib/puppet/state/last_run_summary.yaml
```

## 6.2.2.設定項目の説明

plugin.puppetd.puppetd
　　PuppetClient の実行ファイルを指定します。

plugin.puppetd.lockfile
　　PuppetAgent のロックファイルを指定します。
　　以下のコマンドを実行したときに作成されます。

　　deploy# mco puppetd disable

plugin.puppetd.statefile
　　Puppet の状態の記録するファイルの格納先を指定します。

plugin.puppet.pidfile
　　Puppet の PID ファイルの場所を指定します。

plugin.puppet.summary
　　最後に実行された時間を記録したファイルの格納先を指定します。