# 自动批量部署维护系统

## 摘要：

在云存储或云计算应用环境中，测试人员或系统部署人员通常需要部署几十台、数百台或更多节点程序，如果手动一台台部署，这工作量非常庞大，而且全手动部署极易出错。

本系统主要解决大批量部署程序及维护问题。适用于批量安装、卸载、升级、更新程序或配置，批量执行shell命令等场景。

部署系统基于puppet系统上改进完成，脚本语言选用bash shell，系统仅使用puppet的命令发起及文件同步功能，其他部分都由shell脚本实现。

## 应用环境：

系统要求：支持bash shell的Linux系统

网络要求：同一网络内能相互访问

所需服务：网络内需有台dns服务、ntp校时服务，可与puppet master安装在同一台

被控PC所需安装软件包：puppet(并已于puppet master通讯)、git、telnet、ntp、ruby

## 功能特点：

Puppet系统介绍：

Puppet提供一个简洁的但是强大的框架来完成系统管理任务，而且安全，client与master之间通讯都是基于ssl和证书，只有经过master证书认证的client才可以与master通讯。

Puppet是一个CS架构的配置管理工具，在中央服务器上安装puppet-server服务(puppet master)，在需要被管理的目标服务器上安装puppet客户端软件(puppet client)。当客户端连接上服务端后定义在服务器上的配置文件会被编译，然后在客户端上运行。客户端会定时和服务器通信一次(默认半小时)，确认配置信息的更新情况，配置文件将会被重新编译并分发到客户端执行。当然，也可以在服务器上主动触发更新指令来强制各客户端进行配置更新。

Puppet提供file、package、exec、user、crontab、service等资源类型，利用这些资源你可以自由同步文件、安装软件包、执行外部命令、添加系统用户、添加计划任务、开关系统服务等。

但整个系统缺点也很明显：

1. 如果客户端相互间差异性较大，则需要在服务端配置文件中对不同pc进行个别分配，势必导致配置难以维护。
2. 客户端执行的任务是由服务端的配置文件决定，如果服务端先后发起的几次指令不同，则需每次修改配置文件。
3. Puppet master主动发起同步方式是一台台串行发送消息让客户端来同步，当前面的客户端收到同步信号连接上服务端来同步文件时，服务端系统压力增大，如果服务端发送同步消息还没结束，则会影响发送同步消息给客户端甚至可能导致发送同步消息超时。
4. 服务端发起的同步指令，客户端返回值只会记录到reports目录下日志文件中，不能直观显示每台客户端是否成功或失败原因。

本系统针对以上几项缺点进行了优化，主要特点有：

易用性：

1. 批量部署只需执行一个shell命令就可完成。
2. 准备步骤简单，只需配置KV任务表，指定哪台机子执行什么任务。
3. 使用git管理版本，可追溯查询不同运行阶段原版本信息等。

扩展性：

1. 服务端使用标准命令，PC间差异使用其他独立shell脚本来完成不同配置。
2. 模块化管理shell命令，易于扩展。
3. 新加入设备只需配置KV任务表即可开启相应服务。

对比puppet优势：

1. 无需手动修改site.pp文件来更改执行不同任务。
2. 使用git来转移众多客户端来同步文件的压力，还可使用多个git。
3. 服务端发起同步可直接显示错误返回号，方便定位问题。

## 组成：

方案图：

同步命令文件并执行

下载软件及sh文件

Server

GIT

Client

Client

Client

Client

Client

Client

Client

Client

Client

Client

命令文件

软件

Shell

系统分为3个部分(如图)：

1. 服务端(puppet master)：
   1. 命令发起端，存放指令集。
2. 客户端(puppet)：
   1. 接受PM发送过来的指令
   2. 向PM同步文件
   3. 解析指令
   4. 从Git拉取相应文件
   5. 执行指令
3. Git端
   1. 软件存放。
   2. 基础shell命令文件存放。
   3. 任务表，配置不同PC执行的任务。

系统流程图：案例：

GIT

3.要求客户端同步

4.客户端发起同步：

客户端根据site.pp中配置同步文件及执行命令。

6.解析KV文件，执行命令

7.发回返回值

客户端

1.命令发起

2.修改site.pp

5．从git上更新必要shell脚本、KV任务表及软件包

目的：机房有50台PC，需部署整套云存储环境(1台zk及CN服务器，49台SN)

准备步骤：

1. 向Git服务器上传最新版本ZK、CN、SN必要软件包。
2. 上传软件对应的默认配置文件及特殊配置脚本。
3. 修改KV文件，分配哪台PC运行ZK、CN服务，哪些运行SN程序。

执行任务：

在puppet master端运行/etc/puppet下的create.sh文件。

注：如果中途任意步骤出错都有对应的错误号返回给服务端，以便定位问题。(详细错误代码见附录中错误代码表)

Puppet master与PC客户端交互流程：

结束后，返回结果

只下载final目录对应软件，客户端开始执行命令

解析KV,再下载相应软件包

Puppet

Puppet目录

shell目录

从Git服务器上，下载shell目录

根据site.pp文件,同步puppet目录

./create.sh

要求puppet同步

同步

Puppet Master

Puppet目录

Puppet Master

Puppet目录

Puppet

Puppet目录

Puppet

Git

shell目录

final目录

Puppet Master

Puppet目录

更改site.pp

Git

shell目录

final目录

Puppet

Puppet目录

shell目录

CN

PC端执行指令流程：

CN服务开启

Telnet成功，运行程序,删除tmp文件夹,信息写入State文件

1、Config/default拷贝至软件目录

2、运行

Config/special下shell脚本更改配置

Puppet

Puppet目录

shell目录

CN

tmp

W

Puppet

Puppet目录

Shell

CN

tmp

W

/Config

Puppet

Puppet目录

Shell

CN

Workspace

/Config

Puppet

ZK

创建workspace目录

并把程序移动

解压缩到tmp目录下

Puppet

Puppet目录

shell目录

CN

Puppet

Puppet目录

shell目录

CN

tmp

telnet其他程序

案例2：

目的：删除所有SN上的运行日志。

执行命令：

在puppet master端运行/etc/puppet下，输入

./cmd.sh “rm –fr /home/testuser/workspace/skyFs-controller/log” SN

将命令返回值传回PM

执行完成后，删除tmp.sh

解析KV文件，判断自己是否是SN，不是则退出，并返回给PM

将命令写入tmp.sh文件

./cmd.sh “rm –fr /home/testuser/workspace/skyFs-controller/log” SN

同步puppet目录及shell目录

Puppet Master

Puppet

Puppet目录

shell目录

Puppet

Puppet目录

shell目录

Tmp.sh

KV

LOG

LOG

Puppet

Puppet目录

shell目录

Puppet Master

Puppet目录

## 附录：

命令说明：

Puppet Master服务器，/etc/puppet/目录下，主要sh文件说明：

注：圆括号内参数可不填

Create.sh:

参数：(主机名/SN/CN/CLT\_Master/CLT\_Snode)

作用：新安装软件，如机子/home/testuser目录有State文件，则不执行。

Clear.sh

参数：soft/disk/alldisk (主机名/SN/CN/CLT\_Master/CLT\_Snode)

作用：清除软件，清盘功能

Update.sh

参数：soft/config (主机名/SN/CN/CLT\_Master/CLT\_Snode)

作用：更新程序。更新配置文件。

Rebuild.sh

参数：soft (主机名/SN/CN/CLT\_Master/CLT\_Snode)

作用：删除原程序，新安装程序。

Cmd.sh

参数：命令 (主机名/SN/CN/CLT\_Master/CLT\_Snode)

例：./cmd.sh “rm –fr /home/testuser/workspace/skyFs-controller/log/\*” SN

作用：执行简单单行shell命令。

Restart.sh

参数：(主机名/SN/CN/CLT\_Master/CLT\_Snode)

作用：根据State文件重新软件。

其他sh文件：

gitPullFiles.sh：

参数：无

作用：更新files/puppet下的git库。

Sync.sh:

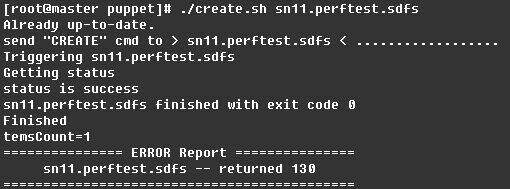
参数：(主机名)

作用：发送同步命令。

getReport.sh:

参数：整形

作用：获取客户端运行错误返回值。

例：

Puppet master端(/etc/puppet目录)配置：

1. 接口shell命令：用于命令发起。
2. manifests/site.pp文件：
   1. site.pp文件告诉Puppet去哪里寻找并载入指定的客户端配置。
3. Files目录
   1. Master目录下存放基础配置文件，如：ssh密钥、挂载硬盘、启动ntp校时服务等
   2. Puppet目录存放将同步到客户端上运行的shell文件，用于解析服务端发出的标准命令。

/etc/puppet/

Manifests/

Site.pp

Files/

Master/

Puppet/

接口Shell命令

基础配置文件

命令解析sh文件

Git目录结构：

Shell.git及Final.git或beta.git都可以分布与不同的git服务器上只需在配置文件中更改git路径即可。

1. shell目录：
   1. basic：用于存放
   2. 软件名命名的文件夹：
      1. Default：存放默认配置文件。
      2. Special：读取本机信息来配置配置文件的shell脚本。
   3. KV：任务表(配置hostname与执行任务的关系，结构详见附录中KV表结构)
2. Final目录：
   1. 以不同软件命令文件夹存放不同软件包。
3. Beta目录：
   1. 与Final相同，不同的是存放的是测试版本软件包。

Git

Shell/

Final/

CLT/

CN/

SN/

ZK/

Basic/

CLT/

CN/

SN/

ZK/

KV

Default/

Special/

Shell方法文件

Config/

skyFS-mapreduce.zip

skyFS-controller.zip

skyFS-storage.zip

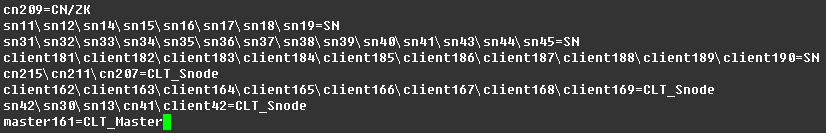
Zookeeper.zip

KV任务表结构:

Hostname=任务名

主机hostname获取方式：在PC输入

主机名间使用”\”分隔，右边任务名使用”/”分隔。



State文件：

作用：记录当前主机运行的软件、git版本、开启的时间

结构：

[软件名1]

date=软件1开始运行时间

git=软件1git版本号

[软件名2]

date=软件2开始运行时间

git=软件2git版本号

错误代码表：

|  |  |  |
| --- | --- | --- |
| 错误号 | 模块 | 原因 |
| 130 | Create.sh | 客户端State文件已存在 |
| 131 | Update.sh | 参数错误—没有指定参数 |
| 132 | Update.sh | 读取State文件错误 |
| 133 | Update.sh | 参数错误 |
| 134 | Update.sh | 没有找到State文件 |
| 135 | Rebuild.sh | 参数错误—没有指定参数 |
| 136 | Clear.sh | 参数错误—没有指定参数 |
| 137 | Clear.sh | 参数错误 |
| 138 | InitShPath.sh | 创建shell目录错误 |
| 139 | Cmd.sh | 参数错误—没有指定参数 |
| 255 | Git\_pull.sh | 参数错误—没有指定参数 |
| 254 | Git\_pull.sh | Git pull error |
| 253 | analyzeJob.sh | 参数错误—没有指定参数 |
| 252 | initState.sh | State文件错误 |
| 251 | nameToPath.sh | 参数错误—没有指定参数 |
| 250 | nameToPath.sh | 参数错误 |
| 249 | readConfig.sh | 参数错误—没有指定参数 |
| 248 | Run.sh | 参数错误—没有指定参数 |
| 247 | Run.sh | 启动软件错误 |
| 246 | Unzip.sh | 参数错误—没有指定参数 |
| 245 | Unzip.sh | Tar解压缩错误 |
| 244 | Unzip.sh | Unzip解压缩错误 |
| 243 | Update\_config.sh | 参数错误—没有指定参数 |
| 242 | Update\_config.sh | 没有找到etc目录 |
| 241 | Update\_soft.sh | 参数错误—没有指定参数 |
| 240 | Update\_soft.sh | 没有找到lib目录 |
| 239 | writeState.sh | 参数错误—没有指定参数 |
| 238 | configSoft.sh/update\_config.sh | 拷贝ZK默认配置文件错误 |
| 237 | configSoft.sh/update\_config.sh | 拷贝默认配置文件错误 |
| 236 | Clear\_soft.sh | 没有找到State文件 |
| 235 | Clear\_soft.sh | 读取State文件 |
| 234 | Git\_pull.sh | 创建Git目录错误 |
| 233 | analyzeJob.sh | KV表中没有hostname对应的任务 |
| 232 | isSoftRunning.sh | 参数错误—没有指定参数 |
| 231 | isSoftRunning.sh | 软件不在运行中 |
| 230 | jobWhoDo.sh | 参数错误—没有指定参数 |
| 229 | jobWhoDo.sh | 任务未找到对应的PC |
| 228 | isSoftRunning.sh | telnet错误 |
| 227 | restartSoft.sh | 参数错误—没有指定参数 |
| 226 | initStateValue.sh | 参数错误—没有指定参数 |
| 225 | initWorkspace.sh | 无法创建workspace目录 |