BigCloud Enterprise Linux 6.5

运维手册

BCLinux 6.5 集群管理

本文介绍如何在**BCLinux 6**中使用运维管理工具ansible。

1 Ansible简介

Ansible 是个什么东西呢？官方的title是“Ansible is Simple IT Automation”——简单的自动化IT工具。

这个工具的目标有这么几项：自动化部署APP；自动化管理配置项；自动化的持续交互；自动化的（AWS）云服务管理。所有的这几个目标**从本质上来说都是在一个台或者几台服务器上，执行一系列的命令而已**。通俗的说就是批量的在远程服务器上执行命令 。

当然，最主要的是它是基于 paramiko 开发的。这个paramiko是什么呢？它是一个纯Python实现的ssh协议库。因此fabric和ansible还有一个共同点就是不需要在远程主机上安装client/agents，因为它们是基于ssh来和远程主机通讯的。简单归纳一下：

- 基于 Python paramiko 开发，分布式，无需客户端，轻量级，配置语法使用 YMAL 及 Jinja2模板语言，更强的远程命令执行操作
- ansible是一款功能强大的自动化运维工具，集合了puppet、cfengine、chef、func、fabric等功能；
- ansible能实现多节点发布和远程任务执行等功能，可满足日常自动化运维需求；
- 简单的来讲，Ansible其实就是实现了能够在其他设备上运行命令，并将结果信息取回的这一功能；

1.1 特性

- No agents：不需要在被管控主机上安装任意客户端（因为它们是基于ssh来和远程主机通讯的）；
- No server：无服务器端，使用时直接运行命令即可；
- Modules in any languages：基于模块工作，可使用任意语言开发模块

- YAML，not code：使用yaml语言定制剧本playbook；
- SSH by default：基于SSH工作；
- Strong multi-tier solution：可实现多级指挥；

1.2 基本架构

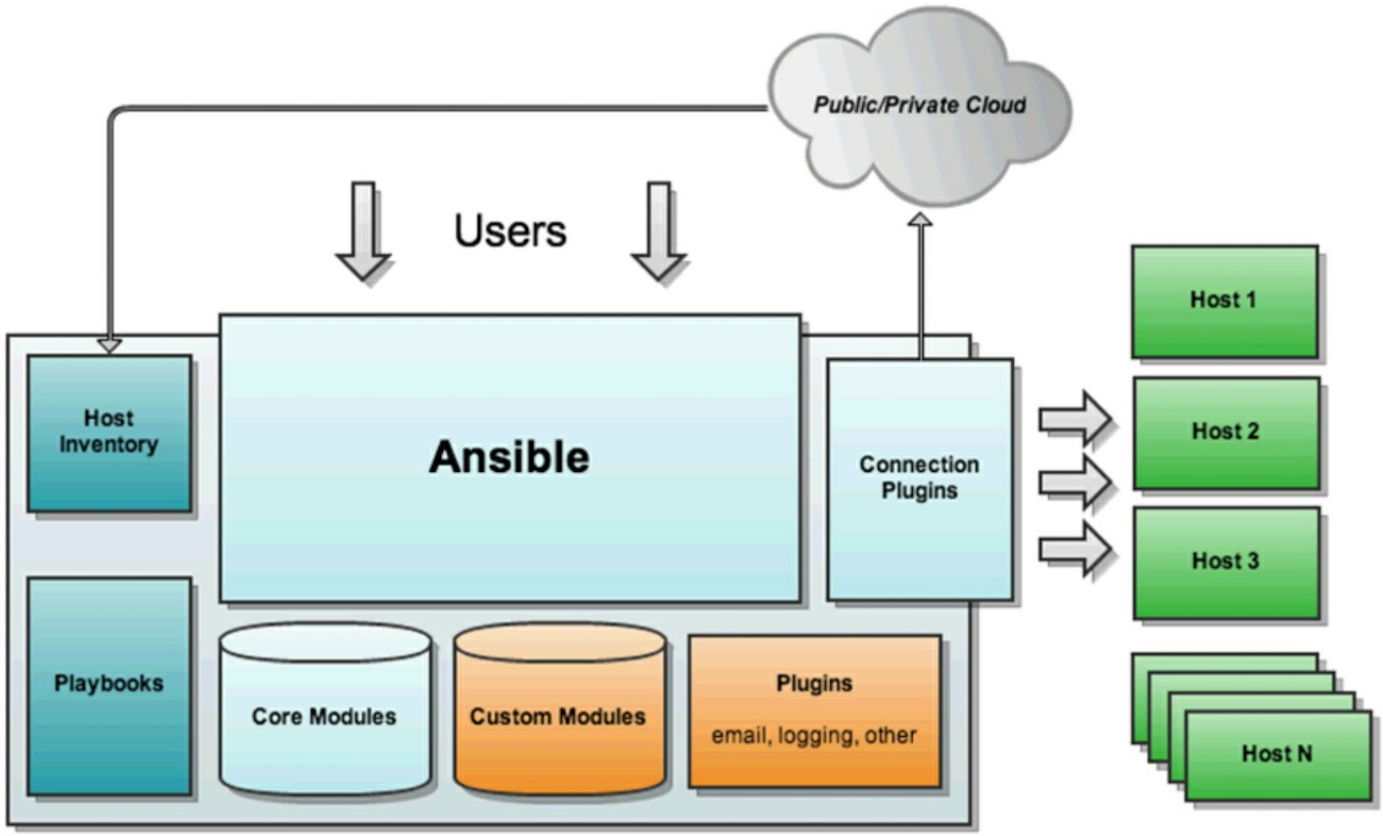


图1：Ansible基本架构图

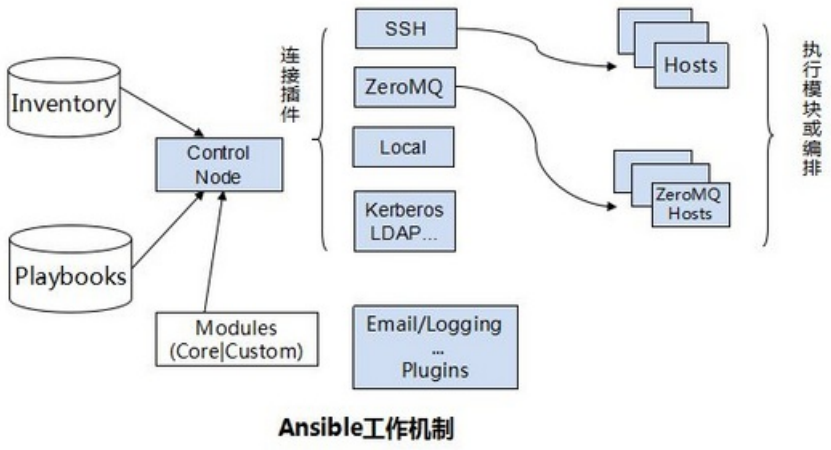


图2：Ansible工作机制

- 核心：ansible
- 核心模块（Core Modules）：这些都是ansible自带的模块，可以通过`ansible-doc -l`查看
- 扩展模块（Custom Modules）：如果核心模块不足以完成某种功能，可以添加扩展模块
- 插件（Plugins）：完成模块功能的补充
- 剧本（Playbooks）：ansible的任务配置文件，将多个任务定义在剧本中，由ansible自动执行
- 连接插件（Connectior Plugins）：ansible基于连接插件连接到各个主机上，虽然ansible是使用ssh连接到各个主机的，但是它还支持其他的连

接方法，所以需要有连接插件

- 主机清单（Host Inventory）：定义ansible管理的主机，默认在`/etc/ansible/hosts`文件中配置；

2 服务的安装

问题1：需要安装哪些东西？安装到哪里？

Ansible默认情况下，通过SSH协议来管理机器。

一旦安装Ansible，它不会创建任何的数据库，并且不会有守护进程在后台一直保持运行。你只需要在一台机器上安装它，它就可以从该中心节点管理当前网络的所有机器。当Ansible管理远程计算机时，它不会在其上安装或运行任何软件，所以当需要对Ansible进行更新时是非常方便的。

问题2：应该安装哪个版本？

Ansible的发布周期通常是四个月。 由于这种短的发布周期，小错误一般会在下一版本被修复。 对于一些主要的缺陷会有单独的维护版本（这种情况比较少见）。

首先，对于使用RHEL、CentOS、Fedora、Debian或Ubuntu的用户，我们推荐用户直接使用安装包来安装Ansible；其次，用户可以通过“pip”这个Python包管理器来安装Ansible。如果没有必要，不推荐用户直接使用最新的开发版本的源码来安装。

问题3：管理节点和被控节点的软件版本依赖需求？

- 管理节点：

目前Ansible可以从任何一台机器安装了Python2.6或2.7的机器上运行（不支持Windows操作系统）。

- 被控节点：

被控节点不需要安装任何软件，你只需要确保被控节点与管理节点之间能够建立通信，通常可以使用ssh，对于文件的传输，Ansib默认使用sftp，但也可以使用scp，这些方式只需要在ansible.cfg文件中进行配置即可。同时，安装的python版本要高于2.4，如果版本低于2.5则需要安装`python-simplejson`。

如果被控节点中启用了SELinux，则在使用Ansible的 copy/file/template 相关的功能之前，还需要安装`libselinux-python`，

如果一些系统安装了Python3（Python3与Python2.x差距较大），Ansible还未支持，一个解决办法就是安装完Ansible之后，设置`ansible_python_interpreter = /usr/bin/python2`来指向 Python2.x 。

如果你想要通过Ansible来监控这些远程系统，则只需要保证这些系统上安装了Python2.x软件包即可，具体可以通过`ansible myhost --sudo -m raw -a "yum install -y python2 python-simplejson"` 命令来完成。

添加EPEL仓库

我们这里是BCLinux 6，所以设置如下仓库：

```
[root@bclinux65 ~]# rpm -ivh http://mirrors.sohu.com/fedora-epel/6/x86_64/epel-release-6-8.noarch.rpm
```

如果是其他版本，到http://mirrors.sohu.com/fedora-epel/下选择合适的仓库。

安装Ansible

```
 [root@bclinux65 ~]# yum install ansible
```

安装后，查看Ansible的版本：

```
[root@bclinux65 ~]#  ansible --version
 ansible 1.9.4
 configured module search path = None.
```

查看Ansible相关文件含义（重要）：

```
[root@bclinux65 ~]#  man  ansible

...省略的信息

FILES
       /etc/ansible/hosts — Default inventory file

       /usr/share/ansible/ — Default module library

       /etc/ansible/ansible.cfg — Config file, used if present

       ~/.ansible.cfg — User config file, overrides the default config if present

ENVIRONMENT
       The following environment variables may be specified.

       ANSIBLE_HOSTS  — Override the default ansible hosts file

       ANSIBLE_LIBRARY — Override the default ansible module library path

       ANSIBLE_CONFIG — Override the default ansible config file

...省略的信息
```

3 Ansible使用

Ansible作为一款运维管理工具，适用于管理整个集群，所以在使用Ansible之前，我们需要确保管理节点与被控节点之间的ssh通信正常。

3.1 Ansible配置

3.1.1 SSH互信

在服务器端生成公钥（如果.ssh目录下已经有了id_rsa秘钥对，则略过这一步）：

```
[root@bclinux65 ~]# ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /root/.ssh/id_rsa.
Your public key has been saved in /root/.ssh/id_rsa.pub.
The key fingerprint is:
4f:2b:0b:66:2a:c5:f0:cf:03:46:9a:b2:9a:ec:ea:7f root@bclinux65.bclinux.novalocal
The key's randomart image is:
+--[ RSA 2048]----+
|                 |
|                 |
|                 |
|  . .            |
|   B    S .      |
|. o *    o .     |
| o o ++ . o      |
|o..  E+. o       |
|O+.oo  ..        |
+-----------------+
```

把生成的ansible公钥复制到待管理的节点中：

```
[root@bclinux65 ~]# ssh-copy-id -i root@10.133.17.101
/usr/bin/ssh-copy-id: INFO: attempting to log in with the new key(s), to filter out any that are already installed
/usr/bin/ssh-copy-id: INFO: 1 key(s) remain to be installed -- if you are prompted now it is to install the new keys
root@10.133.17.101's password:

Number of key(s) added: 1

Now try logging into the machine, with:   "ssh 'root@10.133.17.101'"
and check to make sure that only the key(s) you wanted were added.
```

注意：在首次连接或者重装系统之后会出现检查 keys 的提示：

```
The authenticity of host '10.133.17.101 (10.133.17.101)' can't be established.
ECDSA key fingerprint is 05:51:e5:c4:d4:66:9b:af:5b:c9:ba:e9:e6:a4:2b:fe.
Are you sure you want to continue connecting (yes/no)?
```

解决办法:

```
vim /etc/ansible/ansible.cfg 或者 ~/.ansible.cfg
[defaults]
host_key_checking = False
```

也可以通过设置系统环境变量来禁止这样的提示：

```
export ANSIBLE_HOST_KEY_CHECKING=False
```

这样我们在服务器端输入 ssh root@10.133.17.101就可以以root身份无密码远程登录到10.133.17.101了。

> 注意：很多时候，一些童鞋使用`scp`来把公钥文件传递到远程主机上，然后再给`cat`到远程主机的`authorized_keys`文件中，但是这种情况是属于你 **手动创建** 了`authorized_keys`文件，这个时候这个文件的 **权限** 跟你系统定义的`umask`是相关联的，所以也就导致了你已经上传公钥到远程主机但是依旧不能通过公钥来登录远程主机。这是`authorized_keys`**文件权限**问题导致的。

> 如果你的机器不行，那就尝试着将`authorized_keys`权限改为`600`即可。

3.1.2 配置文件的设置

ansible安装之后的配置文件位于/etc/ansible目录下，共有两个配置文件：

- ansible.cfg：主配置文件；
- hosts：主机清单；

a、查看配置文件cfg常用参数：

```
hostfile  = /etc/ansible/hosts      #默认目标主机文件
library = /usr/share/ansible        #库文件，装载模块时候其模块装载的路径，因为ansible是模块化的工具
module_name =command                #模块的名称，意思是没有使用指定模块的时候其默认使用的模块，默认是命令，意为只执行命令
forks = 5                           #启动的子进程
remote_port  = 22                   #被管理主机的端口，默认为22，如果ssh端口有变动则需要修改此参数
sudo_user = root                    #如果以普通用户登录进行sudo那么sudo的默认账户是root
```

b、hosts配置文件：

一般按照安全性要求，应该禁止 root 登录，禁用密码登录，一律使用证书登录。 ansible 支持通过 sudo 来执行。所有管理主机，每个被管理主机都可以使用主机名或者ip地址都可以。**使用中括号分组，紧跟着中括号下面的都是同一组内被管理的主机**。

hosts配置文件中参数简介：

```
ansible_ssh_user                    #指定用于管理远程主机的账号
ansible_ssh_host                    #指定被管理的主机
ansible_ssh_port                    #指定ssh的端口
ansible_ssh_private_key_file        #指定key文件
host_key_checking=False             #当第一次连接远程主机时会提示yes/no，忽略此检查
```

资源管理清单默认在`/etc/ansbile/hosts`中配置；也可以自定义在命令行用`-i`指定配置文件地址：`ansible all -i /etc/ansible/hosts1 -m ping`。

例如：打开`/etc/ansible/hosts`，加入维护的主机ip（注意，此时你的ssh公钥必须要已经加入到了该目标IP机器中）：

```
[test-servers]
 10.133.17.101
```

/etc/ansible/hosts 主机清单配置格式如下

```
[dbservers]                                                 [] 表示主机的分组名 ， 可以按照功能、系统等进行分类，便于对某些主机或者某一组功能相同的主机进行操作

192.168.1.12

one.example.com

badwolf.example.com:5309                                     支持指定 SSH 端口 5309

jumper ansible_ssh_port=5555 ansible_ssh_host=192.168.1.50     设置主机别名为 jumper

www[01:50].example.com                                      支持通配符匹配 www01 www02 …www50

[databases]

db-[a:f].example.com                                         支持字母匹配 a b c…f
```

为某主机指定连接类型和连接用户

```
[zhao]

Localhost  ansible_connection=local

other1.example.com ansible_connection=ssh ansible_ssh_user=deploy

other2.example.com ansible_connection=ssh ansible_ssh_user=deploy

hosts 文件支持一些特定指令，上面已经使用了其中几个，所有支持的指令如下：
```

清单通配模式介绍，通配具有规则特征的主机或者主机名

```
one.example.com

*.example.com

192.168.1.100

192.168.1.*
```

通配两个组的所有主机，组名之间通过冒号隔开，表示OR的意思

```
web

web:db
```

非模式匹配：表示在 web组不在db组的主机

```
web:!db
```

交集匹配：表示同时都在 web 和db组的主机

```
web:&db
```

匹配一个组的特定编号的主机（先后顺序 0 到…），匹配 web组的第 1 个主机

```
web[0]
```

匹配 web组的第 1 个到第 25 个主机

```
web [0-25]                                                官网文档是" : "表示范围，测试发现应该使用" - " , 注意不要和匹配多个主机组混淆
```

组合匹配：在 web或者 db组中，必须还存在于test1组中，但是不在test2组中

```
web:db:&test1:!test2
```

3.2 Ansible命令

3.2.1 命令格式：

```
ansible <host-pattern> [-f forks] [-m module_name] [-a args]
```

参数说明：

```
host-pattern                                    表示`/etc/ansible/hosts`中配置的主机，`all`表示所有主机；

-v,-verbose                                     详细模式，如果命令执行成功，输出详细的结果 (-vv -vvv -vvvv)

-i PATH,-inventory=PATH                         指定 host 文件的路径，默认是在 /etc/ansible/hosts

-f NUM,-forks=NUM                               表示我们可以明确指明启动多少个子进程去连接这些主机默认不定义为5个forks          如果节点比较多，可
以将forks参数调大

-m NAME,-module-name=NAME                       指定使用的 module 名称，每个模块都需要接受特定参数,默认是 command；可以用 `ansible-doc -l`
 来查看已有的模块，通过 `ansible-doc  -s  模块名` 来查看该模块的哪些参数可以使用；
-a,MODULE_ARGS                                  指定 module 模块的参数
-M DIRECTORY,-module-path=DIRECTORY             指定 module 的目录来加载 module ，默认是 /usr/share/ansible,


-k,-ask-pass                                    提示输入 ssh 的密码，而不是使用基于 ssh 的密钥认证

-sudo                                           当使用普通用户登陆管理时，可以在命令末尾加上此参数以获取root权限；如：ansible all -m ping -s
udo
-K,-ask-sudo-pass                               提示输入 sudo 密码，与 -sudo 一起使用
-u USERNAME,-user=USERNAME                      指定连接远程主机时的用户，默认为root

-C,-check                                       测试此命令执行会改变什么内容，不会真正的去执行
```

3.2.2 简单测试

ping远程服务器（使用ping模块，无模块参数）

```
[root@bclinux65 ~]# ansible test-servers -m ping
10.133.17.101 | success >> {
    "changed": false,
    "ping": "pong"
}
```

查看远程服务器系统负载（使用command模块在**远程节点**上执行命令，无模块参数）

```
[root@bclinux65 ~]# ansible test-servers -m command -a uptime
10.133.17.101 | success | rc=0 >>
 17:01:07 up  6:13,  1 user,  load average: 0.00, 0.00, 0.00
```

查看远程服务器操作系统的发行版号

```
[root@bclinux65 ~]# ansible test-servers -m command -a "uname -r"
10.133.17.101 | success | rc=0 >>
2.6.32-431.el6.x86_64
```

为远程服务器添加用户

```
[root@bclinux65 ~]# ansible test-servers -m command -a "useradd mark"
10.133.17.101 | success | rc=0 >>
```

3.3 Ansible常用模块

Ansible通过模块的方式来完成一些远程的管理工作。可以通过`ansible-doc -l`查看所有模块，可以使用`ansible-doc -s module`来查看某个模块的参数，下面列出一些常用的模块：

a、setup 模块

可以用来查看远程主机的一些基本信息：

```
ansible test-servers -i /etc/ansible/hosts -m setup
```

b、ping 模块

可以用来测试远程主机的运行状态：

```
ansible test-servers -m ping
```

c、command：在远程主机上执行命令

command模块包含如下选项：

```
creates：一个文件名，当该文件存在，则该命令不执行

free_form：要执行的linux指令

chdir：在执行指令之前，先切换到该指定的目录

removes：一个文件名，当该文件不存在，则该选项不执行

executable：切换shell来执行指令，该执行路径必须是一个绝对路径
```

示例：

```
ansible test-servers -a "/sbin/reboot"
```

d、shell 模块

默认情况下，**ansible使用的module 是 command，这个模块并不支持 shell 变量和管道等**，若想使用shell 来执行模块请使用-m 参数指定 shell 模块,但是值得注意的是普通的命令执行模块是通过python的ssh执行。

使用shell模块在远程主机上执行命令:

```
ansible test-servers -m shell -a 'echo $TERM'
```

e、service 模块

用于管理服务，该模块包含如下选项：

```
arguments：给命令行提供一些选项

enabled：是否开机启动  yes|no

name：必选项，服务名称

pattern：定义一个模式，如果通过status指令来查看服务的状态时，没有响应，就会通过ps指令在进程中根据该模式进行查找，如果匹配到，则认为该服务依然在运行

runlevel：运行级别

sleep：如果执行了restarted，在则stop和start之间沉睡几秒钟

state：对当前服务执行启动，停止、重启、重新加载等操作（started,stopped,restarted,reloaded）
```

示例：

```
启动web 组所有主机的 httpd服务
ansible web -m service -a "name=httpd state=started"

重启 web 组所有主机的 httpd服务
ansible web -m service -a "name=httpd state=restarted"

关闭web组所有主机的 httpd服务
ansible web -m service -a "name=httpd state=stopped"
```

f、cron 模块

用于管理计划任务，包含如下选项：

```
backup：对远程主机上的原任务计划内容修改之前做备份

cron_file：如果指定该选项，则用该文件替换远程主机上的cron.d目录下的用户的任务计划

day：日（1-31，*，*/2,……）

hour：小时（0-23，*，*/2，……）

minute：分钟（0-59，*，*/2，……）

month：月（1-12，*，*/2，……）

weekday：周（0-7，*，……）

job：要执行的任务，依赖于state=present

name：该任务的描述

special_time:指定执行时间，参数：reboot,yearly,annually,monthly,weekly,daily,hourly

state：确认该任务计划是创建还是删除

user：以哪个用户的身份执行
```

示例：

```
ansible test -m cron -a 'name="check dirs" hour="5,2" job="ls -alh > /dev/null"'  每天5点、2点

ansible test -m cron -a 'name="a job for reboot" special_time=reboot job="/some/job.sh"'  每次reboot之后执行

ansible test -m cron -a 'name="yum autoupdate" weekday="2" minute=0 hour=12 user="root" job="YUMINTERACTIVE=0 /usr/sbin/yum-autoupdate" cron_file=ansible_yum-autoupdate' 每周2的12点

ansilbe test -m cron -a 'cron_file=ansible_yum-autoupdate state=absent' 删除定时计划
```

g、raw 模块

Raw 也是命令执行模块，类似于command，而raw模块则可以传递管道，是直接用ssh模块进行执行，通常用在客户机还没有python的环境的时候。

使用raw模块在远程主机上执行命令:

```
ansible test-servers -m raw -a 'echo $TERM'
```

h、copy 模块

实现主控端向目标主机拷贝文件，类似于scp的功能，copy模块包含如下选项：

```
backup：在覆盖之前将原文件备份，备份文件包含时间信息。有两个选项：yes|no

content：用于替代"src",可以直接设定指定文件的值

dest：必选项。要将源文件复制到的远程主机的绝对路径，如果源文件是一个目录，那么该路径也必须是个目录

directory_mode：递归的设定目录的权限，默认为系统默认权限

force：如果目标主机包含该文件，但内容不同，如果设置为yes，则强制覆盖，如果为no，则只有当目标主机的目标位置不存在该文件时，才复制。默认为yes

others：所有的file模块里的选项都可以在这里使用

src：要复制到远程主机的文件在本地的地址，可以是绝对路径，也可以是相对路径。如果路径是一个目录，它将递归复制。在这种情况下，如果路径使用”/”来结尾，则只复制目录里的内容，如果没有使用”/”来结尾，则包含目录在内的整个内容全部复制，类似于rsync。
```

示例：

```
ansible test-servers -m copy -a "src=/etc/hosts dest=/tmp/hosts" #拷贝本地的/etc/hosts 文件到 test-servers 主机组所有主机的/tmp/hosts（空目录除外）

ansible test -m copy -a "src=/srv/myfiles/foo.conf dest=/etc/foo.conf owner=foo group=foo mode=0644"
ansible test -m copy -a "src=/mine/ntp.conf dest=/etc/ntp.conf owner=root group=root mode=644 backup=yes"
ansible test -m copy -a "src=/mine/sudoers dest=/etc/sudoers validate='visudo -cf %s'"
```

i、file模块

file 模块称之为文件属性模块，主要用于设置文件属性，包含如下选项：

```
force：需要在两种情况下强制创建软链接，一种是源文件不存在但之后会建立的 情况下；另一种是目标软链接已存在,需要先取消之前的软链，然后创建新的软链，有两个选项：yes|no

group：定义文件/目录的属组

mode：定义文件/目录的权限

owner：定义文件/目录的属主

path：必选项，定义文件/目录的路径

recurse：递归的设置文件的属性，只对目录有效

src：要被链接的源文件的路径，只应用于state=link的情况

dest：被链接到的路径，只应用于state=link的情况

state：
    directory：如果目录不存在，创建目录
    file：即使文件不存在，也不会被创建
    link：创建软链接
    hard：创建硬链接
    touch：如果文件不存在，则会创建一个新的文件，如果文件或目录已存在，则更新其最后修改时间
    absent：删除目录、文件或者取消链接文件
```

示例：

```
ansible test-servers -m file -a "src=/etc/fstab dest=/tmp/fstab state=link"
ansible test-servers -m file -a "path=/tmp/fstab state=absent"
ansible test-servers -m file -a "path=/tmp/test state=touch"
```

j、template 模块

根据官方的翻译是： template 使用了 Jinjia2 格式作为文件模板，进行文档内变量的替换的模块。他的每次使用都会被 ansible 标记为 changed 状态。

k、stat 模块

获取远程文件状态信息，包含atime、ctime、mtime、md5、uid、gid等

```
ansible test-servers -m stat -a "path=/tmp/zhao/a.txt"
```

l、包管理模块

apt 、yum模块分别用于管理ubuntu系列和redhat系列系统软件包

- 安装nginx软件包

```
ansible test-servers -m yum -a "name=nginx state=present"
ansible test-servers -m apt -a "name=nginx state=present"
```

- 安装包到一个特定的版本

  ansible test-servers -m yum -a "name=nginx-1.6.2 state=present"
  ansible test-servers -m apt -a "name=nginx-1.6.2 state=present"

- 指定某个源仓库安装某软件包

  ansible test-servers -m yum -a "name=php55w enablerepo= remi state=present"

- 更新一个软件包是最新版本

  ansible test-servers -m yum -a "name=nginx state=latest"
  ansible test-servers -m apt -a "name=nginx state=latest"

- 卸载一个软件

  ansible test-servers -m yum -a "name=nginx state=absent"
  ansible test-servers - m apt -a "name=nginx state=absent"

Ansible 支持很多操作系统的软件包管理，使用时 -m 指定相应的软件包管理工具模块，如果没有这样的模块，可以自己定义类似的模块或者使用 command 模块来安装软件包。

m、User 模块

使用 user 模块对于创建新用户和更改、删除已存在用户非常方便

创建一个zhao用户并更新密码(密码必须为加密过的字符串)

```
ansible all -m user -a "name=zhao password= $6$YyF5qLN8$edF1l.d/xcd9kv4ZQD/VVq5g2Uavlwoo/l.W4YVIQgsNghN4CbJKSEdZ5ihxztkYJ.bZV2PCP6MnGOioSLqUK."
```

n、删除zhao用户

```
ansible all -m user -a "name=zhao state=absent"
```

3.4 Playbook 解析

> playbook是由一个或多个“play”组成的列表。play的主要功能在于将事先归并为一组的主机装扮成事先通过ansible中的task定义好的角色。从根本上来讲，所谓task无非是调用ansible的一个module。将多个play组织在一个playbook中，即可以让它们联合起来按事先编排的机制完成某一任务。简单来说，**playbook是一个非常简单的配置管理和多主机部署系统**，不同于任何已经存在的模式，可作为一个适合部署复杂应用程序的基础。

> Playbook可以定制配置，可以按照指定的操作步骤有序执行，支持同步和异步方式。值得注意的是**playbook是通过YAML格式来进行描述定义的**。基本的YMAL语法请参考这里。

3.4.1 Playbook 组成

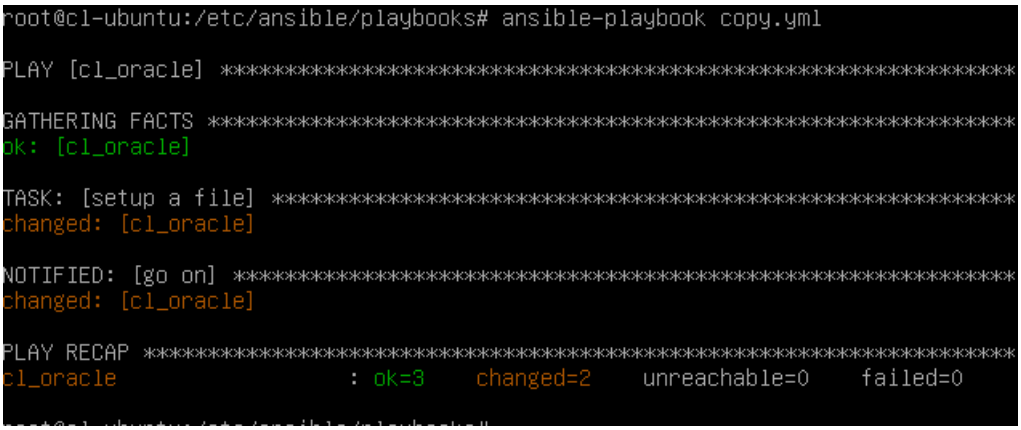
- Target section 定义将要执行 playbook 的远程主机组
- Variable section 定义 playbook 运行时需要使用的变量
- Task section 定义将要在远程主机上执行的任务列表
- Handler section 定义 task 执行完成以后需要调用的任务

以下是一个Playbook：

```
—
- hosts: web
  vars:
  worker_processes: 4
  max_open_file: 65535
  remote_user: deploy

  tasks:
    - name: ensure nginx is at the latest version
      yum: pkg=nginx state=latest
        - name: write the nginx config file
          template: src=/data/ansible/template/nginx.j2 dest=/etc/nginx.conf

      notify:
        - restart nginx
            - name: ensure nginx is running
              service: name=nginx state=started

  handlers:
    - name: restart nginx
      service: name=nginx state=restarted
```

通过下面这个例子我们来简单了解一下：

```
—

- hosts: cl_oracle
  remote_user: root
  tasks:
     - name: setup a file
       command: mkdir /home/test
       sudo: yes
       notify: go on
  handlers:
     - name: go on
       command: echo 'yeah'
~
~
~
~
~
~
~
~
~
~
~
```




```
hosts：定义远程的主机组

user：执行该任务组的用户

remote_user：连接远程主机组的用户

sudo：如果设置为yes，执行该任务组的用户在执行任务的时候，获取root权限

sudo_user：如果你设置user为tom，sudo为yes，sudo_user为jerry，则tom用户则会获取jerry用户的权限

connection：通过什么方式连接到远程主机，默认为ssh

gather_facts：除非你明确说明不需要在远程主机上执行setup模块，否则默认会自动执行。如果你确实不需要setup模块所传递过来的变量，你可以启用该选项。
```

在这里需要注意，handler部分的任务必须在tasks中进行声明，如果tasks没有进行声明，那么即使playbook中有handler模块也不会执行。

a、主机和用户

在playbook中的每一个play都可以选择在哪些机器和以什么用户完成，`hosts`一行可以是一个主机组或者主机或者主机也可以是多个，中间以冒号分隔，可以参考前面提到的清单通配模式：其中remote_user表示执行的用户账号

```
—
- hosts: web
  remote_user: root
```

每个任务都可以定义自己的用户

```
—
- hosts: web
  remote_user: root
  tasks:
    - name: test connection
      ping:
      remote_user: yourname
```

在 playbook中使用sudo

```
—
- hosts: web
  remote_user: yourname
  sudo: yes
```

在一个任务中使用sudo

```
—
- hosts: web
  remote_user: yourname

  tasks:
   - service: name=nginx state=started
     sudo: yes
```

登陆后 sudo 到其他用户执行

```
—
- hosts: web
  remote_user: yourname
  sudo: yes
  sudo_user: postgres
```

注释：在使用 sudo_user 切换到非 root 用户时，Ansible 会将模块参数（非密码选项参数）记录到/tmp 下的一个临时随机文件，命令执行完后会删除；当 sudo 到 root 或者普通用户登陆时并不记录

b、任务列表

所有定义的任务列表(tasks list)，playbook将按照定义的配置文件自上而下的顺序执行，定义的主机都将得到相同的任务，但是执行的返回结果不一定

保存一致，取决于主机的环境及程序包状态。建议每个任务事件都要定义一个name标签，好处是增强可读性，也便于观察结果输出时了解运行的位置。下面就一service模块为例来定义一个任务，service: key=value参数，具体请参考模块的详细介绍

```
tasks:
- name: ensure nginx is running
service: name=nginx state=started
```

command 和 shell 模块不需要增加 key

```
tasks:
- name: disable selinux
command: setenforce 0
```

command 和shell模块关注命令或者脚本执行后返回值，如果命令成功执行返回结果不是0的情况下可以使用以下方法：

```
tasks:
- name: disable selinux
command: setenforce 0
ignore_errors: True
```

如果在任务中参数过长可以回车使用空格缩进

```
tasks:
- name: Copy ansible inventory file to client
copy: src=/etc/ansible/hosts dest=/tmp/hosts
owner=root group=root mode=0644
```

c、变量的使用

如何创建一个有效的变量名

变量名应该由字母、数组和下划线组成，以字母开头。例如：foo_port、foo5 就是很好的变量名，而 foo-port、foo port、foo.port、12 都是无效的变量名

在 playbook 中如何定义变量

```
- hosts: webservers
vars:
http_port: 80
```

变量这一块就不多提了，针对这一块了解甚微，如有需要请自行查找，这里简单介绍一下jinja2过滤器中的register关键字，register关键字的作用是将命令执行的结果保存为变量，结果会因为模块不同而不同，在运行ansible-playbook时增加-v参数就可以看到results可能的值。比如：

```
—
- hosts: local
remote_user: root
tasks:
- name: Get server Time
shell: date +%Y-%m-%d_%H_%M
register : Time
ignore_errors: True
- name: Create a file
shell: touch /tmp/zhao{{Time.stdout}}.txt
```

执行结果如下：

```
$ ansible-playbook 2.yml -v

PLAY [local] ******************************************************************

GATHERING FACTS ***************************************************************

ok: [192.168.1.52]

TASK: [Get server Time] *******************************************************

changed: [192.168.1.52] => {"changed": true, "cmd": "date +%Y-%m-%d_%H_%M" , "delta": "0:00:00.002363", "end": "2015-06-19 1
1:06:12.359652", "rc": 0, "start": "2015-06-19 11:06:12.357289", "stderr": "", " stdout": "2015-06-19_11_06 ", "warnings": [
]}

TASK: [Create a file] *********************************************************

changed: [192.168.1.52] => {"changed": true, "cmd": "touch /tmp/zhao2015-06-19_11_06.txt" , "delta": "0:00:00.002386", "end"
: "2015-06-19 11:06:12.631204", "rc": 0, "start": "2015-06-19 11:06:12.628818", "stderr": "", "stdout": "", "warnings": ["Co
nsider using file module with state=touch rather than running touch"]}

PLAY RECAP ********************************************************************

192.168.1.52               : ok=3    changed=2    unreachable=0    failed=0
```

通过上面这个例子我们可以看到task的执行输出和facts一样的，我们可以通过jinja2模板获取变量值： {{Time.stdout}} 、 {{Time.cmd}} 等等

d、Notify 和 Handles

用于当关注的资源发生变化时采取一定的操作。

“notify”这个action可用于在每个play的最后被触发，这样可以避免多次有改变发生时每次都执行指定的操作，取而代之，仅在所有的变化发生完成后一次性地执行指定操作。在notify中列出的操作称为handler，也即notify中调用handler中定义的操作。

```
- name: template configuration file

template: src=template.j2 dest=/etc/foo.conf

notify:

- restart memcached

- restart apache
```

handler 是task列表，这些task与前述的task并没有本质上的不同。

```
handlers:

- name: restart memcached

service:  name=memcached state=restarted

- name: restart apache

service: name=apache state=restarted
```

e、角色定义使用

Roles 在Ansible1.2+版本中所被支持，主要是为了更好的组织playbooks。

举例说明：

```
site.yml
webservers.yml
fooservers.yml
roles/
common/
files/
templates/
tasks/
handlers/
vars/
defaults/
meta/
webservers/
files/
templates/
tasks/
handlers/
vars/
defaults/
meta/
```

在 playbook 中可以这样使用 roles

```
—
- hosts: webservers

roles:
- common
- webservers
```

roles 目录结构说明：

tasks 、 handlers 、 vars （只对当前 role 有效）、 meta （定义 role 间的直接依赖关系）目录内存在

main.yml 文件时会将对应的任务、处理、变量和 meta 添加到 play

files 存放文件， ansible 默认会从这里找文件，对应 task 里面的 copy 、 script 模块

template 存放模板，对应 task 里面的 template 模块

tasks 存放任务， include 默认读取这里的任务

defaults 默认的变量存放位置，使用 /devaults/main.yml, 相对其他参数变量设置的优先级最低

注释： Ansible1.4+ 以后的版本，可以通过 roles_path 参数配置 roles 路径，多路径使用冒号分隔，可以将常见角色集中存放，指定 roles 路径，这样多个 playbook 可以共用

3.4.2 运行 Playbook

查看模块执行成功与否的详细信息

$ansible-playbook playbook.yml -v

查看一个 playbook 中都会对哪些主机产生影响

$ansible-playbook playbook.yml --list-hosts

查看都有哪些任务要执行

$ansible-playbook playbook.yml --list-tasks

3.4.3 Playbook示例

下面的格式很重要,冒号后面一定要有空格,tasks下面一定要有缩进,并且每行对齐

```
- hosts: all          #执行的主机,前面有横杠表示可以有多个
 remote_user: root     #使用哪个用户的身份在各远程主机上执行命令,可以定义全局的,也可以定义单个任务要使用的用户
 tasks:               #需要执行的任务,因为有子任务,所以下面的任务需要缩进
   - name: add a group  #第一个执行的任务的名字,类似注释信息
     group: gid=1001 name=testymal system=no  #具体的任务本身
   - name: excute a command
     command: /bin/date
```

以上的执行顺序是：把一个任务在所有主机上执行一遍,再把第二个任务在所有主机上执行,如果在执行任务的过程中发生错误,将回滚所有可回滚的任务

ansible的handler：

```
- hosts: all
  remote_user: root
  tasks:
    - name: ensure apache latest version
      yum: name=httpd state=laster
    - name: apache configure file
      copy: src=/root/httpd.conf dest=/etc/httpd/conf force=yes
      notify:                          #调用下面的handlers
        - restart httpd
      handlers:                        #定义handlers要执行的动作
        - name: restart httpd
      service: name=httpd state=restarted
```

ansible的playbooks实现安装heartbeat，heartbeat.yaml：

```
- hosts: hbhosts
  remote_user: root
  tasks:
    - name: ensure heartbeat latest version
      yum: name=heartbeat state=present          #安装最新版本
    - name: authkeys configure file
      copy: src=/root/hb_conf/authkeys dest=/etc/ha.d/authkeys  #复制本地文件到远程主机
    - name: authkeys mode 600
          file: path=/etc/ha.d/authkeys mode=600 #改文件权限,可以使用ansible-doc -s file查看
          notify:                                #调用下面的handlers
            - restart heartbeat
        - name: ha.cf configure file
          copy: src=/root/hb_conf/ha.cf dest=/etc/ha.d/ha.cf     #复制本地文件到远程主机
          notify:                                #调用下面的handlers
            - restart heartbeat                  #调用下面的handlers
      handlers:                                  #定义handlers的动作,这个的缩进与tasks一致
        - name: restart heartbeat
      service: name=heartbeat state=restarted
```

4 参考资料

- ansible官网
- Ansible入门与进阶
- ansible及ansible-palybook使用详解
- Ansible 中文手册
- Ansible模块知多少

BCLinux 6.5 缺陷跟踪

1 ABRT介绍

ABRT的全称是Automatic Bug Reporting Tool，它提供了一种机制，当系统中出现应用程序crash或者内核oops和crash时，可以被检测到，并且自动收集相关的信息，发送到远程的问题跟踪服务器。

ABRT有很多组件：

- abrtd

这是ABRT的后台服务，以root用户运行在后台，检测系统中出现的异常情况。

- abrt-gui

ABRT相关的图形服务，当crash发生时，以图形界面的形式通知用户，并提供了图形化的配置工具，对abrt做配置。

- abrt-cli

类似于abrt-gui提供的功能，只不过是命令行形式的。

- abrt-ccpp

这是ABRT提供的addon软件包，提供了对C/C++问题的分析支持。

- abrt-python

这是ABRT提供的addon软件包，提供了对Python问题的分析支持。

- abrt-oops

这是ABRT提供的addon软件包，提供了对内核oops问题的分析支持。

- abrt-vmcore

这是ABRT提供的addon软件包，提供了对内核crash问题的分析支持。

2 ABRT软件包安装

2.1 安装abrt软件包

```
[root@promote ~]# yum install abrt -y
```

2.2 开启abrt服务

```
[root@promote ~]# chkconfig --list abrtd
abrtd           0:off   1:off   2:off   3:on    4:off   5:on    6:off
[root@promote ~]# service abrtd start
Starting abrt daemon:                                      [  OK  ]
[root@promote ~]# service abrtd status
abrtd (pid  7827) is running...
[root@promote ~]#
```

2.3 安装ABRT的addon软件包

```
[root@promote ~]# yum install abrt-addon-* -y
```

2.4 开启ABRT的addon服务

```
[root@promote ~]# service abrt-oops start
[root@promote ~]# service abrt-vmcore start
[root@promote ~]# service abrt-ccpp start
```

2.5 安装abrt-cli

```
[root@promote ~]# yum install abrt-cli -y
```

3 ABRT配置

ABRT检测到crash之后，会主动发送邮件到邮箱。默认发送到bclinux@bigcloudsys.com，这部分的配置文件如下，用户可以根据自己的需要进行修改：

```
[root@promote ~]# cat /etc/libreport/plugins/mailx.conf
# Uncomment and specify these parameters if you want to use
# reporter-mailx tool outside of libreport's GUI
# (i.e. from command line or in custom scripts)
# and you don't want to specify parameters in every tool invocation.
#
# String parameters:
Subject="[BC-Linux 6.5 Abrt] One crash has beed detected!"
EmailFrom="root@localhost"
EmailTo="bclinux@bigcloudsys.com"
#
# Boolean parameter:
# SendBinaryData=yes/no
```

BCLinux 6.5 升级服务

1 YUM介绍

YUM是一个软件包管理工具，广泛应用在Redhat、Fedora和CentOS发行版中。BCLinux基于CentOS做了深度定制，也使用YUM作为其包管理工具。YUM基于RPM包进行管理，可以从指定服务器自动下载RPM包并安装，并可以自动解决RPM包之间的依赖关系，将依赖包一并下载安装。

2 YUM服务搭建

YUM服务是基于HTTP协议的，为了搭建YUM服务，首先需要搭建一个HTTP服务。

2.1 安装httpd

```
[root@localhost ~]# yum install httpd -y
```

2.2 启动httpd服务

```
[root@localhost ~]# chkconfig --level 345 httpd on
[root@localhost ~]# service  httpd start
Starting httpd:
```

2.3 确保httpd服务已开启

```
[root@localhost ~]# ss -tln | grep 80
LISTEN     0      128                    :::80                   :::*
[root@localhost ~]#
```

2.4 关闭防火墙

```
[root@localhost ~]# chkconfig --level 2345 iptables off
[root@localhost ~]# service  iptables stop
iptables: Setting chains to policy ACCEPT: filter          [  OK  ]
iptables: Flushing firewall rules:                         [  OK  ]
iptables: Unloading modules:                               [  OK  ]
```

3 同步BCLinux YUM源

BCLinux的YUM源服务器域名地址是http://mirrors.bclinux.org/，其中的bclinux目录下包含了BCLinux各个版本的YUM源以及安装镜像文件。

在本地搭建YUM源，需要将BCLinux的源服务器内容同步下来。为了满足同步需求，服务器开启了rsync服务，不过这个服务只对认证用户开放。当前，网站没有开通注册功能，需要同步的用户可以发送邮件到邮箱team@bclinux.org申请，审核通过后会分配账号给予权限。

下面介绍如何与BCLinux 源服务器做同步。

3.1 安装rsync

```
[root@localhost html]# yum install rsync -y
```

3.2 同步BCLinux源

这里以bclinux用户，需要输入认证密码才可以开启同步操作：

```
[root@localhost html]# pwd
/var/www/html
[root@localhost html]# rsync -avlP --delete rsync://bclinux@mirrors.bclinux.org/mirrors/bclinux .
Password:
```

4 对客户端做配置

安装BCLinux 6后，默认的YUM配置是BCLinux的官方源服务器载软件包。此时需要对YUM的配置文件做修改，使得其指向本地的YUM服务器。根据现网的环境不同，可以采用不同的方法：

- 如果在内网中配置了DNS服务器，修改DNS配置，将mirrors.bclinux.org和本地YUM源服务器的IP地址做绑定。
- 否则，修改客户端的/etc/hosts文件，添加如下内容：

```
x.x.x.x     mirrors.bclinux.org
```

x.x.x.x指的是本地YUM源的IP地址。

BCLinux 6.5 网络安装

1 PXE介绍

PXE(Pre-boot Execution Environment)是由Intel公司开发的技术，支持主机（客户机）通过远程主机提供的安装服务安装操作系统。
PXE的基本工作原理如下：

1. PXE客户机从支持PXE的网卡启动，向本网络中的DHCP服务器获取IP地址。
2. DHCP服务器返回分配给客户机的IP地址以及pxelinux.0文件的存放位置。
3. PXE客户机向本网络中的TFTP服务器索取pxelinux.0文件，并执行。
4. 从TFTP服务器获取pxelinux的配置文件pxelinux.cfg。
5. 客户机选择启动项，从TFTP服务器下载内核文件和initrd。
6. 加载内核，并启动安装过程。

为了提供PXE安装服务，需要在本地网络中开启如下服务：

1. DHCP服务
   为待安装机器分配IP地址，以及提供PXE安装所需要的其他信息。
2. TFTP服务
   用于提供PXE安装所需要的bootloader，内核镜像等文件。
3. HTTP(S)、NFS、FTP服务
   提供安装过程中所使用的安装源，支持HTTP(S)、NFS和FTP多种协议，本文以HTTP服务为例。

2 服务的搭建

2.1 DHCP服务搭建

2.1.1 安装dhcp软件包

```
[root@promote ~]# yum install dhcp -y
```

2.1.2 修改dhcp配置文件/etc/dhcp/dhcpd.conf

```
option space pxelinux;
option pxelinux.magic code 208 = string;
option pxelinux.configfile code 209 = text;
option pxelinux.pathprefix code 210 = text;
option pxelinux.rebootime code 211 = unsigned integer 32;

subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers 192.168.1.1;
        range 192.168.1.2 192.168.1.254;

        class "pxeclients" {
                match if substring (option vendor-classidentifier, 0, 9) = "PXEClient";
                next-server 192.168.1.1;

                if option arch = 00:06 {
                        filename "bootia32.efi";
                } else if option arch = 00:07 {
                        filename "bootx64.efi";
                } else {
                        filename "pxelinux.0";
                }
        }
}
```

2.1.3 开启dhcp服务

```
[root@promote ~]# chkconfig --level 345 dhcpd on
[root@promote ~]# service dhcpd start
Starting dhcpd:                                            [  OK  ]
```

2.1.4 查看dhcp服务是否开启

```
[root@promote ~]# ss -uln | grep 67
UNCONN      0      0                  *:67                                *:*
```

2.2 TFTP服务搭建

2.2.1 安装tftp-server和syslinux-tftpboot软件包

```
[root@promote ~]# yum install tftp-server syslinux-tftpboot -y
```

2.2.2 将syslinux-tftpboot软件包中的镜像文件，拷贝到tftp根目录下

```
[root@promote tftpboot]# cp -f /tftpboot/* /var/lib/tftpboot/
```

2.2.3 挂载BCLinux 6光盘，拷贝其中的内核镜像和initrd文件到tftp根目录下。假定/mount_point是光盘的挂载目录

```
[root@promote ~]# mkdir -p /var/lib/tftpboot/bclinux6/
[root@promote ~]# cp /mount_point/images/pxeboot/{vmlinuz,initrd.img} /var/lib/tftp/bclinux6/
```

2.2.4 创建pxelinux配置文件

PXE安装默认会从pxelinux.cfg目录下获取PXE的配置文件，首先创建这个目录：

```
[root@promote ~]# mkdir /var/lib/tftpboot/pxelinux.cfg/
```

默认配置文件名是default，如果需要对不同的客户机给于不同的配置文件，可以用IP地址作为文件名。例如对于IP地址是10.0.0.1的客户机而言，针对它的配置文件名称是0A000001。下面是一个典型的配置文件内容：

```
[root@promote ~]# cat /var/lib/tftpboot/pxelinux.cfg/default
default vesamenu.c32
prompt 1
timeout 600
display boot.msg

label linux
    menu label ^Install or upgrade an existing system
    menu default
    kernel bclinux6/vmlinuz
    append initrd=bclinux6/initrd.img
label vesa
    menu label Install system with ^basic video driver
    kernel bclinux6/vmlinuz
    append initrd=bclinux6/initrd.img xdriver=vesa nomodeset
label rescue
    menu label ^Rescue installed system
    kernel bclinux6/vmlinuz
    append initrd=bclinux6/initrd.img rescue
label local
    menu label Boot from ^local drive
    localboot 0xffff
label memtest86
    menu label ^Memory test
    kernel memtest
    append -
```

将boot menu中需要的文件拷贝到tftp目录下：

```
[root@promote ~]# cp /mount_point/isolinux/{boot.msg,memtest} /var/lib/tftpboot/
```

2.2.5 开启tftp服务

将/etc/xinetd.d/tftp文件中的disable选项改成no，重启xinet服务：

```
[root@promote ~]# service xinetd restart
Stopping xinetd:                                           [  OK  ]
Starting xinetd:                                           [  OK  ]
```

2.2.6 查看tftp服务是否开启

```
[root@promote ~]# ss -uln | grep 69
UNCONN      0      0                 *:69                              *:*
```

2.2.7 在本网段内找一台机器，测试tftp服务是否可以正常访问。

2.3 HTTP服务搭建

2.3.1 安装httpd以及相关依赖软件包

```
[root@promote ~]# yum install httpd -y
```

2.3.2 将安装光盘中的内容拷贝到/var/www/html/bclinux6/目录下

```
[root@promote ~]# mkdir /var/www/html/bclinux6/
[root@promote ~]# cp -rf /mount_point/* /var/www/html/bclinux6/
```

2.3.3 开启httpd服务

```
[root@promote ~]# chkconfig --level 345 httpd on
[root@promote ~]# service  httpd start
Starting httpd:                                            [  OK  ]
```

2.3.4 查看httpd服务是否开启

```
[root@promote ~]# ss -tln | grep 80
LISTEN     0      128                    :::80                      :::*
```

2.3.5 在本网段内找一台机器，测试http服务是否可以正常访问。

确保上述服务的端口在防火墙中是放行的，也可以将防火墙关闭：

```
[root@promote ~]# service iptables stop
iptables: Setting chains to policy ACCEPT: filter          [  OK  ]
iptables: Flushing firewall rules:                         [  OK  ]
iptables: Unloading modules:                               [  OK  ]
```

3 自动化安装

按照前面的步骤，在本地局域网内搭建上述服务后，将客户机重启并从网卡启动，就会启动PXE安装。在安装过程中，需要对待安装系统做一些配置，例如时区、磁盘分区等。为了实现这部分的自动化，可以使用kickstart安装方式。

kickstart可以解析配置文件，其中可以给出了安装过程中所有需要回答的问题。安装程序解析这个配置文件，就可以实现完全的自动化安装。

为了实现kickstart安装，需要完成以下步骤。

3.1 创建kickstart文件

kickstart文件是一个文本文件，通过关键字指定了安装过程中需要配置的选项。可以使用图形化的工具Kickstart Configurator生成kickstart文件，也可以手动写。在系统安装结束后，安装程序也会创建一个kickstart文件在/root/目录下，文件名是anaconda-ks.cfg，这个文件保存的是安装系统时的配置参数。

由于kickstart支持的选项非常多，这里就不一一列举，可以参考kickstart选项说明。

3.2 使得安装源可以被访问

无论使用kickstart安装还是其他方式安装，都必须访问安装源。安装源可以在光盘上，也可以在磁盘和网络上，其内容包含了待安装的软件包，一般是从安装光盘中复制而来的。

3.3 使得kickstart文件可以被访问

在使用PXE安装时，为了使用自动化安装，需要访问kickstart文件。此时需要将kickstart文件放置在网络服务器上，在安装过程中通过网络获取文件，开启自动化安装。

现在支持三种形式的网络服务：nfs、http、https，分别需要在启动参数中添加如下配置项：

> ks=nfs:<server>:/<path>

> ks=http://<server>/<path>

> ks=https://<server>/<path>

对于PXE安装，启动的配置文件是pxelinux.cfg/default。假设ks文件被放置HTTP服务上的根目录下，pxelinux.cfg/default文件需要被修改为如下内容：

```
[root@promote ~]# cat /var/lib/tftpboot/pxelinux.cfg/default
default vesamenu.c32
prompt 1
timeout 600
display boot.msg

label linux
    menu label ^Install or upgrade an existing system
    menu default
    kernel bclinux6/vmlinuz
    append initrd=bclinux6/initrd.img ks=http://192.168.1.1/ks.cfg
label vesa
    menu label Install system with ^basic video driver
    kernel bclinux6/vmlinuz
    append initrd=bclinux6/initrd.img xdriver=vesa nomodeset ks=http://192.168.1.1/ks.cfg
label rescue
    menu label ^Rescue installed system
    kernel bclinux6/vmlinuz
    append initrd=bclinux6/initrd.img rescue
label local
    menu label Boot from ^local drive
    localboot 0xffff
label memtest86
    menu label ^Memory test
    kernel memtest
    append -
```

4 外部引用

1. kickstart选项

BCLinux 6.5 运维监控

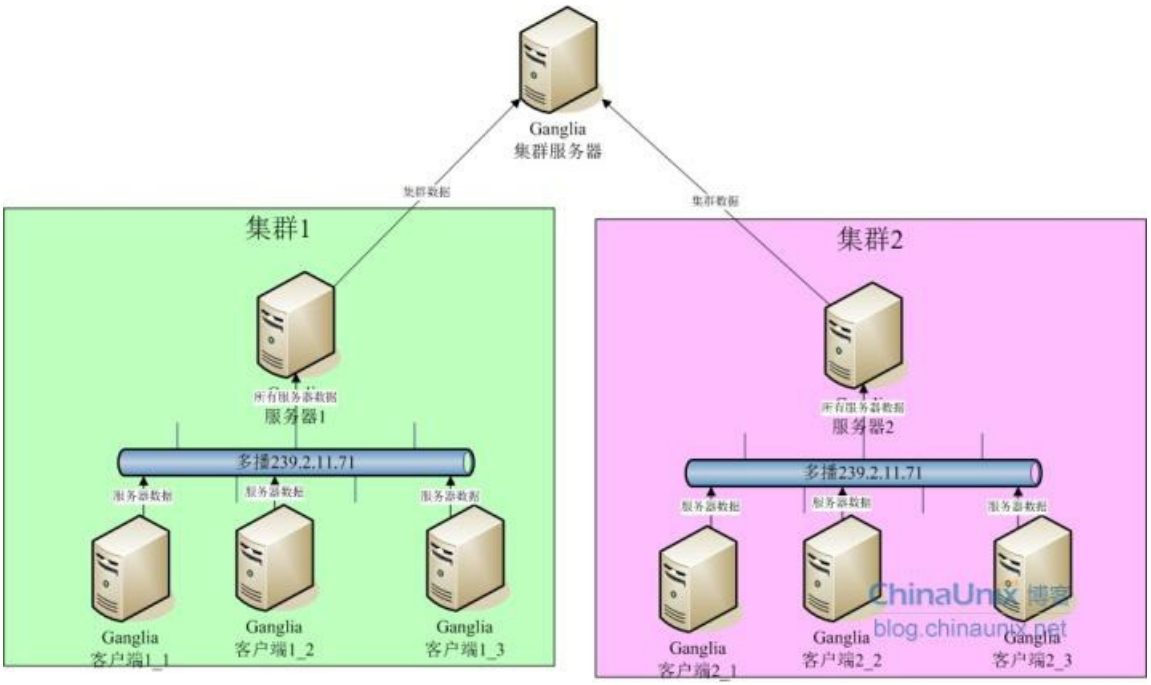
本文介绍如何在**BCLinux 6**中使用ganglia进行运维监控。

1 Ganglia概述

Ganglia是UC Berkeley发起的一个开源集群监视项目，是一个监控服务器、集群的开源软件，设计用于测量数以千计的节点，能够用曲线图表现最近一个小时，最近一天，最近一周，最近一月，最近一年的服务器或者集群的cpu负载、内存、网络、硬盘等指标。

Ganglia的核心包含**gmond**、**gmetad**以及一个**Web前端**。主要是用来监控系统性能，如：cpu 、mem、硬盘利用率、I/O负载、网络流量情况等，通过曲线很容易见到每个节点的工作状态，对合理调整、分配系统资源，提高系统整体性能起到重要作用。

Ganglia的强大在于：ganglia服务端能够**通过一台客户端收集到同一个网段的所有客户端的数据**，ganglia集群服务端能够通过一台服务端收集到它下属的所有客户端数据。这个体系设计表示一台服务器能够通过不同的分层能够管理上万台机器。这个功能是其他mrtg,nagios,cacti所不能比拟。



Ganglia的扩展插件非常好写，无论用何种语言(shell,php,pyton)都可以写,只要把最终结果传给gmetric就可以，这样在web上就可以看到对应的数据。

2 Ganglia安装

EPEL(Extra Packages for Enterprise Linux),这是针对RHEL设计的软件仓库，在这个仓库中有很多免费的常用软件，由Fedora项目维护，如果使用的是BCLinux、RHEL、CentOS、Scientific等RHEL系列的linux，可以非常方便的使用EPEL的yum源。

2.1 添加epel仓库

```
rpm -ivh http://mirrors.sohu.com/fedora-epel/6/x86_64/epel-release-6-8.noarch.rpm
```

导入key：

```
rpm --import /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6
```

2.2 安装ganglia

1. 服务端依赖包： Ganglia里的ganglia
2. 客户端： ganglia里的gmond
3. Web端： Apache

2.2.1 服务端安装

```
yum -y install ganglia
```

安装完成后重启服务；

2.2.2 安装客户端gmond

在每台需要监控的机器上，安装gmond：

```
yum -y install ganglia-gmond
```

安装完成后重启服务；

2.2.3 服务器端Apache

```
yum -y install apache
```

安装完成后重启服务；

3 ganglia的配置

3.1 服务端配置

在/etc/ganglia/gmetad.conf中进行下面的配置

```
data_source "ShaQi" hdp1 hdp2 hdp3
setuid_username "apache" （可以不设置）
```

3.2 客户端配置

在/etc/ganglia/gmond.conf中进行下面的配置

```
cluster {
  name = "ShaQi"
  owner = "apache"
  latlong = "unspecified"
  url = "unspecified"
}
```

其中`name`需要与`gmetad.conf`中`data_source`中的“ShaQi”匹配

`owner`需要与`setuid_username`中的值对应(如果没有设置，可以不用修改)

修改了setuid_username后需要对权限进行更改:

```
Chown -R apache:apache /var/lib/ganglia/rrds
```

否则使用`service gmetad status`时出现`gmetad dead but subsys locked`

3.3 Apache的服务端配置

/etc/httpd/conf.d/ganglia.conf

代码如下:

```
Alias /ganglia /usr/share/ganglia

<Location /ganglia>

Order deny,allow

Allow from all

</Location>
```

4 ganglia的启动

启动服务端gmetad并且设为开机启动

```
service gmetad start

chkconfig gmetad on
```

通过telnet localhost 8651进行测试

启动客户端的gmond并设为开机启动

```
service gmond start

chkconfig gmond on
```

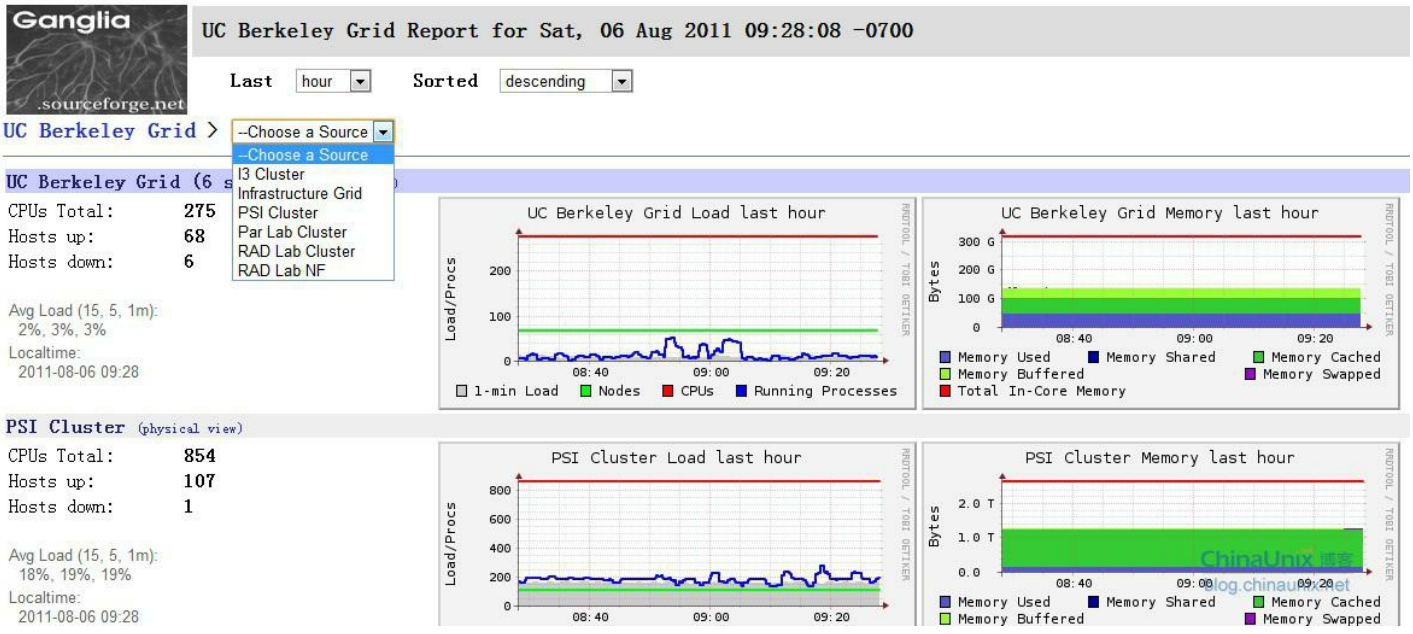
通过telnet localhost 8649进行测试

启动apache服务端

```
service httpd restart
```

5 ganglia的测试

通过浏览器访问：http://service_ip/ganglia加载下面界面：



6 参考

- 开源监控软件ganglia安装手册
- Ganglia在CentOS 6.5的安装