# CentOS6.7下Ansible部署 | 一路向北的博客

http://www.showerlee.com/archives/1649

[Ansible](http://www.showerlee.com/archives/tag/ansible)是一种集成IT系统的配置管理, 应用部署, 执行特定任务的开源平台. 它基于Python语言实现, 部署只需在主控端部署[Ansible](http://www.showerlee.com/archives/tag/ansible)环境, 被控端无需安装代理工具, 只需打开SSH, 让主控端通过SSH秘钥认证对其进行所有的管理监控操作. 相对于SaltStack, 它除了利用SSH安全传输, 无需在客户端进行任何配置, 而且它有一个很庞大的用户群体以及丰富的API, 相对适合部署到数量比较大且对系统软件安装要求比较严格的集群中.

更多配置参考: [https://github.com/ansible](https://github.com/ansible/)

官方文档: [http://docs.ansible.com/ansible](http://docs.ansible.com/ansible/)

本文将帮助大家如何快速部署一个[Ansible](http://www.showerlee.com/archives/tag/ansible)平台.

安装环境:

System: Centos 6.7 x64

Master: master.example.com

Minion: client01.example.com

Minion: client02.example.com

**一. 环境部署及安装**

1. 关闭iptables和SELINUX

# service iptables stop

# setenforce 0

# vi /etc/sysconfig/selinux

...

SELINUX=disabled

...

2. Master端安装EPEL第三方yum源

# rpm -Uvh <http://ftp.linux.ncsu.edu/pub/epel/6/i386/epel-release-6-8.noarch.rpm>

3.安装[Ansible](http://www.showerlee.com/archives/tag/ansible)

# yum install ansible -y

4.添加环境变量以便vi能正常显示中文注释.

# vi /etc/profile

添加:

export LC\_ALL=en\_US.UTF-8

export LANG=en\_US.UTF-8

export LANGUAGE=en\_US.UTF-8

# source /etc/profile

**二. 初始配置**

1. 修改主机及组配置

# cd /etc/ansible

# cp hosts hosts.bak

# cat /dev/null > hosts

# vi /etc/ansible/hosts

[webservers]

client01.example.com

client02.example.com

[nginx01]

client01.example.com

[nginx02]

client02.example.com

2.配置SSH秘钥认证

# yum install ssh\* -y

# ssh-keygen -t rsa

Generating public/private rsa key pair.

Enter file in which to save the key (/root/.ssh/id\_rsa):

Created directory '/root/.ssh'.

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:

24:13:34:e9:71:2b:20:0b:48:a6:86:9a:1d:1b:1d:26 [root@master.example.com](mailto:root@master.example.com)

The key's randomart image is:

+--[ RSA 2048]----+

|ooE o.+. |

|\* .+..oo. |

|oooo.ooo.. |

|oo.+ o+. |

|o o .S |

| |

| |

| |

| |

+-----------------+

同步公钥文件id\_rsa.pub到目标主机

# ssh-copy-id -i /root/.ssh/id\_rsa.pub [root@client01.example.com](mailto:root@client01.example.com)

# ssh-copy-id -i /root/.ssh/id\_rsa.pub [root@client02.example.com](mailto:root@client02.example.com)

校验SSH免密码配置是否成功.

# ssh [root@client02.example.com](mailto:root@client02.example.com)

如直接进入则配置完成.

3.定义主机与组

所有定义的主机与组规则都在/etc/[Ansible](http://www.showerlee.com/archives/tag/ansible)/hosts下.

常见的写法:

192.168.1.21:2135 定义一个IP为192.168.1.21, SSH端口为2135的主机.

jumper ansible\_ssh\_port=22 ansible\_ssh\_host=192.168.1.50 定义一个别名为jumper, SSH端口为22, IP为192.168.1.50的主机.

组成员主机名称范例:

[webservers]

www[001:006].example.com

[dbservers]

db-[a:f].example.com

4.定义主机变量

主机可以指定变量, 后面可以供Playbooks调用

[atlanta]

host1 http\_port=80 maxRequestsPerChild=808

host2 http\_port=8080 maxRequestsPerChild=909

5.定义组变量

[atlanta]

host1

host2

[atlanta:vars]

ntp\_server=ntp.atlanta.example.com

proxy=proxy.atlanta.example.com

6.匹配目标

重启webservers组所有SSH服务.

# ansible webservers -m service -a "name=sshd state=restarted"

client01.example.com | success >> {

"changed": true,

"name": "sshd",

"state": "started"

}

client02.example.com | success >> {

"changed": true,

"name": "sshd",

"state": "started"

}

**三. Ansible常用模块及API**

1.远程命令模块

command: 执行远程主机SHELL命令:

# ansible webservers -m command -a "free -m"

client01.example.com | success | rc=0 >>

total used free shared buffers cached

Mem: 996 108 887 0 7 41

-/+ buffers/cache: 58 937

Swap: 1023 0 1023

client02.example.com | success | rc=0 >>

total used free shared buffers cached

Mem: 996 108 888 0 7 41

-/+ buffers/cache: 58 937

Swap: 1023 0 1023

script: 远程执行MASTER本地SHELL脚本.(类似scp+shell)

# echo "df -h" > ~/test.sh

# ansible webservers -m script -a "~/test.sh"

client01.example.com | success >> {

"changed": true,

"rc": 0,

"stderr": "OpenSSH\_5.3p1, OpenSSL 1.0.1e-fips 11 Feb 2013\ndebug1: Reading configuration data /etc/ssh/ssh\_config\r\ndebug1: Applying options for \*\r\ndebug1: auto-mux: Trying existing master\r\ndebug1: mux\_client\_request\_session: master session id: 2\r\ndebug1: mux\_client\_request\_session: master session id: 2\r\nShared connection to client01.example.com closed.\r\n",

"stdout": "Filesystem Size Used Avail Use% Mounted on\r\n/dev/sda3 6.6G 815M 5.5G 13% /\r\ntmpfs 499M 0 499M 0% /dev/shm\r\n/dev/sda1 190M 27M 154M 15% /boot\r\n"

}

client02.example.com | success >> {

"changed": true,

"rc": 0,

"stderr": "OpenSSH\_5.3p1, OpenSSL 1.0.1e-fips 11 Feb 2013\ndebug1: Reading configuration data /etc/ssh/ssh\_config\r\ndebug1: Applying options for \*\r\ndebug1: auto-mux: Trying existing master\r\ndebug1: mux\_client\_request\_session: master session id: 2\r\ndebug1: mux\_client\_request\_session: master session id: 2\r\nShared connection to client02.example.com closed.\r\n",

"stdout": "Filesystem Size Used Avail Use% Mounted on\r\n/dev/sda3 6.6G 815M 5.5G 13% /\r\ntmpfs 499M 0 499M 0% /dev/shm\r\n/dev/sda1 190M 27M 154M 15% /boot\r\n"

}

2. copy模块

实现主控端向目标主机拷贝文件, 类似scp功能.

该实例实现~/test.sh文件至webservers组目标主机/tmp下, 并更新文件owner和group

# ansible webservers -m copy -a "src=~/test.sh dest=/tmp/ owner=root group=root mode=0755"

# ansible webservers -m copy -a "src=~/test.sh dest=/tmp/ owner=root group=root mode=0755"

client01.example.com | success >> {

"changed": true,

"checksum": "c989bd551bfa8c755f6cacacb90c5c509432110e",

"dest": "/tmp/test.sh",

"gid": 0,

"group": "root",

"md5sum": "69a238d8cb3c5f979252010b3299e524",

"mode": "0755",

"owner": "root",

"size": 6,

"src": "/root/.ansible/tmp/ansible-tmp-1445322165.21-234077402845688/source",

"state": "file",

"uid": 0

}

client02.example.com | success >> {

"changed": true,

"checksum": "c989bd551bfa8c755f6cacacb90c5c509432110e",

"dest": "/tmp/test.sh",

"gid": 0,

"group": "root",

"md5sum": "69a238d8cb3c5f979252010b3299e524",

"mode": "0755",

"owner": "root",

"size": 6,

"src": "/root/.ansible/tmp/ansible-tmp-1445322165.2-164402895387597/source",

"state": "file",

"uid": 0

}

3.stat模块

获取远程文件状态信息, 包括atime, ctime, mtime, md5, uid, gid等信息.

# ansible webservers -m stat -a "path=/etc/sysctl.conf"

client02.example.com | success >> {

"changed": false,

"stat": {

"atime": 1445312213.9599864,

"checksum": "704d7d26321b453d973939ee41aaf9861e238a78",

"ctime": 1444969315.401,

"dev": 2051,

"exists": true,

"gid": 0,

"gr\_name": "root",

"inode": 130328,

"isblk": false,

"ischr": false,

"isdir": false,

"isfifo": false,

"isgid": false,

"islnk": false,

"isreg": true,

"issock": false,

"isuid": false,

"md5": "9ce78fbee91a542ca29d3e7945486e27",

"mode": "0644",

"mtime": 1437725687.0,

"nlink": 1,

"path": "/etc/sysctl.conf",

"pw\_name": "root",

"rgrp": true,

"roth": true,

"rusr": true,

"size": 998,

"uid": 0,

"wgrp": false,

"woth": false,

"wusr": true,

"xgrp": false,

"xoth": false,

"xusr": false

}

}

client01.example.com | success >> {

"changed": false,

"stat": {

"atime": 1445312212.9747968,

"checksum": "704d7d26321b453d973939ee41aaf9861e238a78",

"ctime": 1444969315.401,

"dev": 2051,

"exists": true,

"gid": 0,

"gr\_name": "root",

"inode": 130328,

"isblk": false,

"ischr": false,

"isdir": false,

"isfifo": false,

"isgid": false,

"islnk": false,

"isreg": true,

"issock": false,

"isuid": false,

"md5": "9ce78fbee91a542ca29d3e7945486e27",

"mode": "0644",

"mtime": 1437725687.0,

"nlink": 1,

"path": "/etc/sysctl.conf",

"pw\_name": "root",

"rgrp": true,

"roth": true,

"rusr": true,

"size": 998,

"uid": 0,

"wgrp": false,

"woth": false,

"wusr": true,

"xgrp": false,

"xoth": false,

"xusr": false

}

}

4.get\_url模块

实现在远程主机下载指定URL到本地.

# ansible webservers -m get\_url -a "url=http://www.showerlee.com dest=/tmp/index.html mode=0400 force=yes"

client02.example.com | success >> {

"changed": true,

"checksum": "470d6ab960810153bb8149c3754b0e8a2d89209d",

"dest": "/tmp/index.html",

"gid": 0,

"group": "root",

"md5sum": "009949f770f35a4ea82105e5e923abcb",

"mode": "0400",

"msg": "OK (unknown bytes)",

"owner": "root",

"sha256sum": "",

"size": 81635,

"src": "/tmp/tmpa44PoE",

"state": "file",

"uid": 0,

"url": "http://www.showerlee.com"

}

client01.example.com | success >> {

"changed": true,

"checksum": "9b1afd16f97c07638965ba0c5cf01037af00a38a",

"dest": "/tmp/index.html",

"gid": 0,

"group": "root",

"md5sum": "5a935e77927286dfcb7a0190e8af461b",

"mode": "0400",

"msg": "OK (unknown bytes)",

"owner": "root",

"sha256sum": "",

"size": 81679,

"src": "/tmp/tmp5WHuj0",

"state": "file",

"uid": 0,

"url": "http://www.showerlee.com"

}

5.yum模块

Linux包管理平台操作,  常见都会有yum和apt, 此处会调用yum管理模式

# ansible servers -m yum -a "name=curl state=latest"

client01.example.com | success >> {

"changed": false,

"msg": "",

"rc": 0,

"results": [

"All packages providing curl are up to date"

]

}

client02.example.com | success >> {

"changed": false,

"msg": "",

"rc": 0,

"results": [

"All packages providing curl are up to date"

]

}

6. cron模块

远程主机crontab配置

# ansible webservers -m cron -a "name='check dir' hour='5,2' job='ls -alh > /dev/null'"

client02.example.com | success >> {

"changed": true,

"jobs": [

"check dir"

]

}

client01.example.com | success >> {

"changed": true,

"jobs": [

"check dir"

]

}

7.service模块

远程主机系统服务管理

# ansible webservers -m service -a "name=crond state=stopped"

# ansible webservers -m service -a "name=crond state=restarted"

# ansible webservers -m service -a "name=crond state=reloaded"

8.user服务模块

远程主机系统用户管理

添加用户:

# ansible webservers -m user -a "name=johnd comment='John Doe'"

删除用户:

# ansible webservers -m user -a "name=johnd state=absent remove=yes"

**四. playbook介绍**

playbook是一个不同于使用Ansible命令行执行方式的模式, 其功能是将大量命令行配置集成到一起形成一个可定制的多主机配置管理部署工具.

它通过YAML格式定义, 可以实现向多台主机的分发应用部署.

以下给大家详细介绍一个针对nginx嵌套复用结构的playbook部署实例:

1. 构建目录结构

# cd /etc/ansible/

# mkdir group\_vars

# mkdir roles

2.定义host

# vi /etc/ansible/hosts

[webservers]

client01.example.com

client02.example.com

[nginx01]

client01.example.com

[nginx02]

client02.example.com

3.定义变量

# vi /etc/ansible/group\_vars/nginx01

worker\_processes: 4

num\_cpus: 4

max\_open\_file: 65506

root: /data

remote\_user: root

# vi /etc/ansible/group\_vars/nginx02

worker\_processes: 2

num\_cpus: 2

max\_open\_file: 35506

root: /www

remote\_user: root

Tips:这里在group\_vars下定义的文件名必须对应hosts文件下的group标签, 通过这里定义的不同参数从而部署不同类型的主机配置.

4.创建roles入口文件

# vi /etc/ansible/site.yml

- hosts: webservers

roles:

- base\_env

- hosts: nginx01

roles:

- nginx01

- hosts: nginx02

roles:

- nginx02

Tips: 这里的**roles:**下的字符串需对应roles目录下的目录名.

5.定义全局role base\_env

创建目录结构

# mkdir -p /etc/ansible/roles/base\_env/tasks

# vi /etc/ansible/roles/base\_env/tasks/main.yml

# 将EPEL的yum源配置文件传送到客户端

- name: Create the contains common plays that will run on all nodes

copy: src=epel.repo dest=/etc/yum.repos.d/epel.repo

- name: Create the GPG key for EPEL

copy: src=RPM-GPG-KEY-EPEL-6 dest=/etc/pki/rpm-gpg

# 关闭SELINUX

- name: test to see if selling is running

command: getenforce

register: sestatus

changed\_when: false

# 删除iptables默认规则并保存

- name: remove the default iptables rules

command: iptables -F

- name: save iptables rules

command: service iptables save

将对应需要拷贝到远程的文件复制到base\_env/files目录下

# mkdir -p  /etc/ansible/roles/base\_env/files

# cp /etc/yum.repos.d/epel.repo /etc/ansible/roles/base\_env/files

# cp /etc/pki/rpm-gpg/RPM-GPG-KEY-EPEL-6 /etc/ansible/roles/base\_env/files

6. 定义nginx01和ngnix02 role

创建目录结构

# mkdir -p /etc/ansible/roles/nginx{01,02}

# mkdir -p /etc/ansible/roles/nginx01/tasks

# mkdir -p /etc/ansible/roles/nginx02/tasks

# vi /etc/ansible/roles/nginx01/tasks/main.yml

# 安装nginx最新版本

- name: ensure nginx is at the latest version

yum: pkg=nginx state=latest

# 将nginx配置文件传送到远程目录

- name: write the nginx config file

template: src=nginx.conf dest=/etc/nginx/nginx.conf

notify: restart nginx # 重启nginx

# 创建nginx根目录

- name: Create Web Root

file: dest={{ root }} mode=775 state=directory owner=nginx group=nginx

notify: reload nginx

- name: ensure nginx is running

service: name=nginx state=restarted

# cp /home/ansible/roles/nginx01/tasks/main.yml /home/ansible/roles/nginx02/tasks/main.yml

7.定义files

# mkdir -p /etc/ansible/roles/nginx01/templates

# mkdir -p /etc/ansible/roles/nginx02/templates

# vi /etc/ansible/roles/nginx01/templates/nginx.conf

# For more information on configuration, see:

user nginx;

worker\_processes {{ worker\_processes }};

{% if num\_cpus == 2 %}

worker\_cpu\_affinity 01 10;

{% elif num\_cpus == 4 %}

worker\_cpu\_affinity 1000 0100 0010 0001;

{% elif num\_cpus >= 8 %}

worker\_cpu\_affinity 00000001 00000010 00000100 00001000 00010000 00100000 01000000 10000000;

{% else %}

worker\_cpu\_affinity 1000 0100 0010 0001;

{% endif %}

worker\_rlimit\_nofile {{ max\_open\_file }};

error\_log /var/log/nginx/error.log;

#error\_log /var/log/nginx/error.log notice;

#error\_log /var/log/nginx/error.log info;

pid /var/run/nginx.pid;

events {

worker\_connections {{ max\_open\_file }};

}

http {

include /etc/nginx/mime.types;

default\_type application/octet-stream;

log\_format main '$remote\_addr - $remote\_user [$time\_local] "$request" '

'$status $body\_bytes\_sent "$http\_referer" '

'"$http\_user\_agent" "$http\_x\_forwarded\_for"';

access\_log /var/log/nginx/access.log main;

sendfile on;

#tcp\_nopush on;

#keepalive\_timeout 0;

keepalive\_timeout 65;

#gzip on;

# Load config files from the /etc/nginx/conf.d directory

# The default server is in conf.d/default.conf

#include /etc/nginx/conf.d/\*.conf;

server {

listen 80 default\_server;

server\_name \_;

#charset koi8-r;

#access\_log logs/host.access.log main;

location / {

root {{ root }};

index index.html index.htm;

}

error\_page 404 /404.html;

location = /404.html {

root /usr/share/nginx/html;

}

# redirect server error pages to the static page /50x.html

#

error\_page 500 502 503 504 /50x.html;

location = /50x.html {

root /usr/share/nginx/html;

}

}

}

Tip: worker\_processes, num\_cpus, max\_open\_file, root等参数会调用group\_vars目录下配置文件中相应的变量值

# cp /etc/ansible/roles/nginx01/templates/nginx.conf  /etc/ansible/roles/nginx02/templates/nginx.conf

8.执行playbook

# ansible-playbook -i /etc/ansible/hosts /etc/ansible/site.yml -f 10

Tips: -f 为启动10个并行进程执行playbook, -i 定义inventory host文件, site.yml 为入口文件

PLAY [webservers] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client02.example.com]

ok: [client01.example.com]

TASK: [base\_env | Create the contains common plays that will run on all nodes] \*\*\*

ok: [client01.example.com]

ok: [client02.example.com]

TASK: [base\_env | Create the GPG key for EPEL] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client02.example.com]

ok: [client01.example.com]

TASK: [base\_env | test to see if selling is running] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client01.example.com]

ok: [client02.example.com]

TASK: [base\_env | remove the default iptables rules] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [client02.example.com]

changed: [client01.example.com]

TASK: [base\_env | save iptables rules] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [client01.example.com]

changed: [client02.example.com]

PLAY [nginx01] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client01.example.com]

TASK: [nginx01 | ensure nginx is at the latest version] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client01.example.com]

TASK: [nginx01 | write the nginx config file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client01.example.com]

TASK: [nginx01 | Create Web Root] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client01.example.com]

TASK: [nginx01 | ensure nginx is running] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [client01.example.com]

PLAY [nginx02] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

GATHERING FACTS \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client02.example.com]

TASK: [nginx02 | ensure nginx is at the latest version] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client02.example.com]

TASK: [nginx02 | write the nginx config file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client02.example.com]

TASK: [nginx02 | Create Web Root] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [client02.example.com]

TASK: [nginx02 | ensure nginx is running] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [client02.example.com]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

client01.example.com : ok=11 changed=3 unreachable=0 failed=0

client02.example.com : ok=11 changed=3 unreachable=0 failed=0

最终部署目录结构如下

# tree /etc/ansible/

/etc/ansible/

├── ansible.cfg

├── group\_vars

│ ├── nginx01

│ └── nginx02

├── hosts

├── hosts.bak

├── roles

│ ├── base\_env

│ │ ├── files

│ │ │ ├── epel.repo

│ │ │ └── RPM-GPG-KEY-EPEL-6

│ │ └── tasks

│ │ └── main.yml

│ ├── nginx01

│ │ ├── tasks

│ │ │ └── main.yml

│ │ └── templates

│ │ └── nginx.conf

│ └── nginx02

│ ├── tasks

│ │ └── main.yml

│ └── templates

│ └── nginx.conf

└── site.yml

11 directories, 13 files

到此, 部署nignx到两台远程webserver服务器全部完成.

# Jenkins+Ansible+Gitlab自动化部署三剑客 | 一路向北的博客

http://www.showerlee.com/archives/1880

最近一直在学习[Ansible](http://www.showerlee.com/archives/tag/ansible)的一些playbook的写法, 所以一直没有怎么更新, 想到目前大家对诸如saltstack, docker, [Ansible](http://www.showerlee.com/archives/tag/ansible)等自动化部署相关的工具很感兴趣, 但又苦于没有可学习的中文实例, 这里我就把我这几个月所接触到目前国外比较流行的部署经验给大家分享一下.

首先给大家介绍的是Ansible, 恩, 重要的问题说三遍, 不是Saltstack, Ansible作为一个python写的自动化部署工具, 确实较之前我所接触的Chef, saltstack, puppet更有自己的一些优势, 首先就是agentless, 无需在Linux client安装任何服务即可无缝连接Linux default ssh端口进行部署(windows需要安装winrm 开启ssh服务), 这点其实我觉得非常重要, 可以想象很多公司本身是对network管理非常严格的, 在部署一个产品的同时你需要考虑很多时间成本, 使用其他部署工具本身非常棘手的问题就是去申请开端口, client量少的话, 我们可以去等, 多的话本身你去request, waiting, unblock port等等long long process.... 最后会耗费很长时间. 这个对很多产品本身就是很致命的. 不推荐Saltstack的原因也是因为其需要在每台agent逐一去安装client service并测试, 这本身就会耗费一些时间成本.

其他呢? 其实我觉得就是容易上手, 语法简单, 有现成模板让你去学习, 加之是我们非常喜爱的python语法, why not?

Jenkins不用我多说, 估计懂行的人都在用它, 开源, 轻量级, 兼容性和扩展性强, 直观的GUI管理这都是它的优势, 配合Ansible我觉得用起来会非常easy going.

最后提一下Gitlab, 为什么要用Gitlab? 他作为一个代码版本控制系统和部署有什么关系呢? 其实这里就涉及一个我们Ansible playbook管理问题, 试想我们需要维护一个公司庞大的server集群, 我们所有需要部署的机器或者产品会对应我们相对的部署脚本, 我们使用的Ansible playbook如果只是保存在Ansible Server的具体某个目录, 这本身就不便于我们进行编写维护更新(想想每次都跑到远程去编写playbook或者每次在本地编写好后再upload到远程我都会脑补数以万计的草泥马从我眼前呼啸而来).

这里Gitlab就给我们提供一个非常方便以及直观的Playbook management. 我们需要做的其实就是在Gitlab去建立一个对应产品或者server的playbook仓库, 然后我们在本地写好后直接commit到这个仓库, 最后在部署的时候, 去让Jenkins pull这个playbook到其workspace, 并作为一个Job去run这个playbook, 这样是不是很规范, 而且便于管理?

当然Ansible本身企业版Tower也会提供一个类似管理并维护playbook以及监控ansible本身running process的GUI管理系统, 用起来也很不错, 但作为收费版本, 我们在这里就不做过多阐述了.

这里我推荐Jenkins和Ansible可以安装到同一个环境作为部署server, Gitlab作为版本控制系统可单独部署在另一台server.

总结:

Jenkins首先从Gitlab去抓取我们写好的具体产品的playbook, 并使用virtualenv下的Ansible相关命令, 保证我们在一个clean的环境下使用stable version去批量部署我们的产品到远程client.

Let's go.....

一. 安装环境

System: CentOS 6.7 x64 (deploy.example.com)

Jenkins: Jenkins ver. 1.650

Ansible: Ansible 2.1.0

Gitlab: GitLab 7.14.3

二. Jenkins配置

我们创建deploy用户作为jenkins\_user, workspace为deploy家目录下的jenkins目录.

# su - root

# adduser deploy

# wget -O /etc/yum.repos.d/jenkins.repo <http://pkg.jenkins-ci.org/redhat/jenkins.repo>

# rpm --import <https://jenkins-ci.org/redhat/jenkins-ci.org.key>

# yum install jenkins -y

# vi /etc/sysconfig/jenkins

...

JENKINS\_HOME="/home/deploy/jenkins"

JENKINS\_USER="deploy"

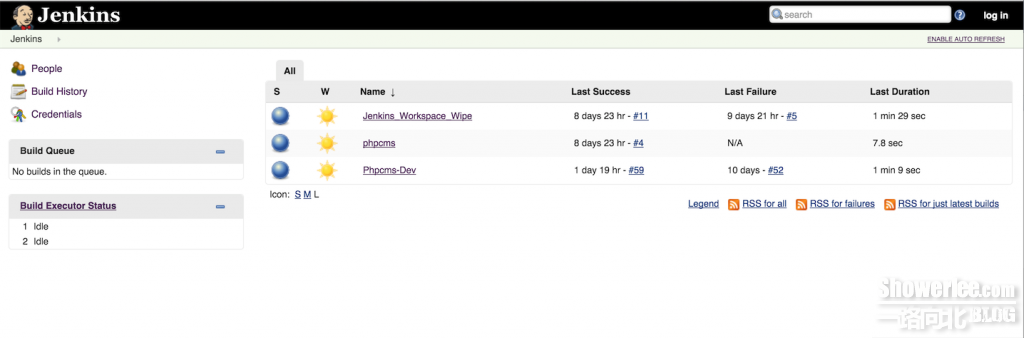
...

# service jenkins start

浏览器访问Jenkins页面

[http://deploy.example.com:8080](http://deploy.example.com:8080/)

安装完成, 以下是我已经配置好的一些Jenkins Job.

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-0.png)

这里我们使用一个国内PHP网站模板phpcms作为我们需要部署的产品进行本次范例演示, 在进行最终的Build前我们需要做一些准备工作, 稍后我们会回到这个界面.

三. Ansible配置

这里我们需要配置virtualenv去隔离我们ansible的发行版本为最新版本2.1.0, 默认pip或者yum安装的1.9版本因为BUG以及对windows不兼容的原因, 这里不推荐使用.

配置步骤传送门: <http://www.showerlee.com/archives/1862>

Ansible-playbook范例传送门: <http://www.showerlee.com/archives/1649>

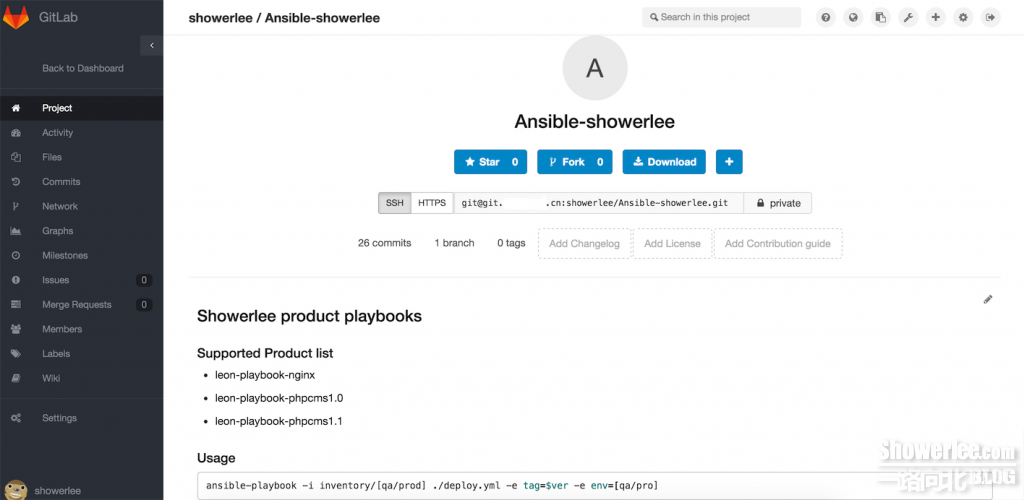
四. Gitlab配置

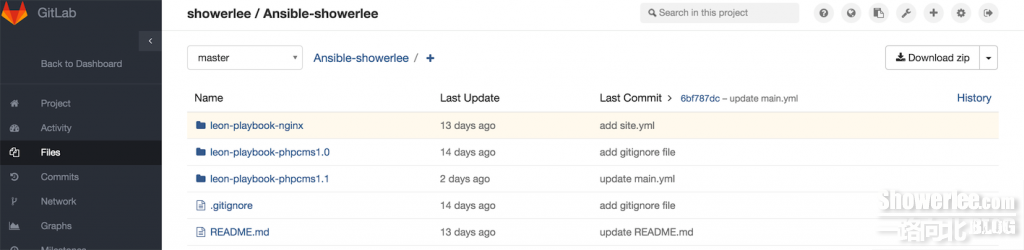
部署并使用传送门: <http://www.showerlee.com/archives/1285>

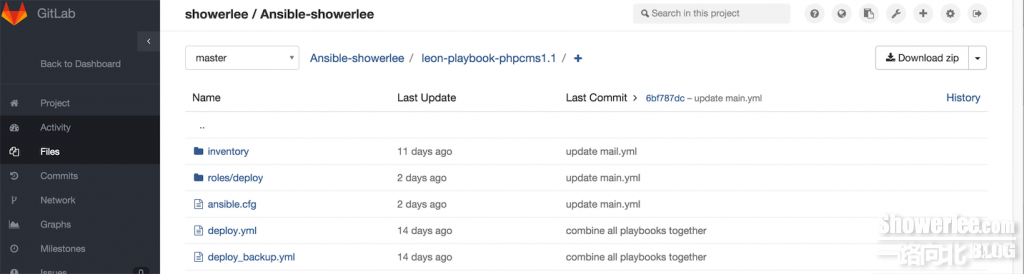
我们最终会创建一个ansible playbook仓库 git@git.example.cn:showerlee/Ansible-showerlee.git, 并在本地编写好我们的规则, 最终commit到这个仓库, 以便Jenkins去调用我们的部署规则.

这里博主单独clone出来一份部署phpcms的playbook仓库, 算是给大家的福利:

<https://git.yanwenbo.cn/showerlee/leon-playbook-phpcms1.1>

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-2.png)

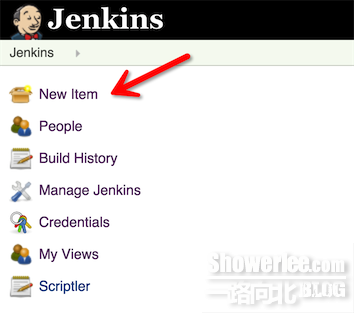
[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-3.png)

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-4.png)

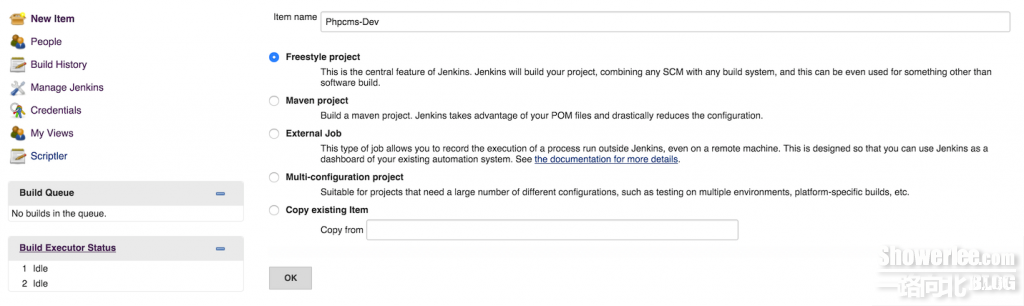
五.最终部署

准备工作完毕, 我们接下来给大家介绍Jenkins Job配置.

1.创建一个new item

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-5.png)

2. 创建一个freestyle Job, 命名规则"产品名-环境", 这里我们为Phpcms-Dev

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-6.png)

3. Job配置

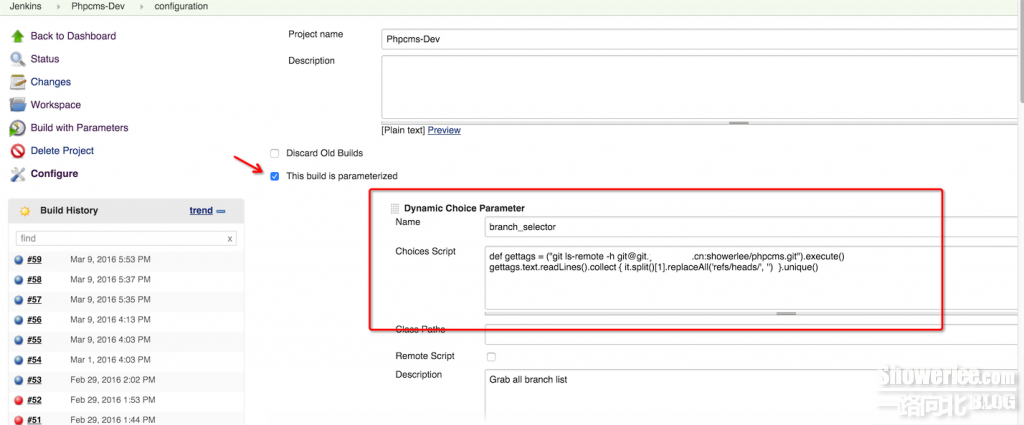
1). 定制Build参数.

这里Dynamic Choice Parameter用来通过Groovy脚本来抓取这个git仓库的所有branch, 并作为一个多选项, 方便我们在最终Build前去选择我们需要的这个产品Branch分支.

Groovy抓取Git branch代码:

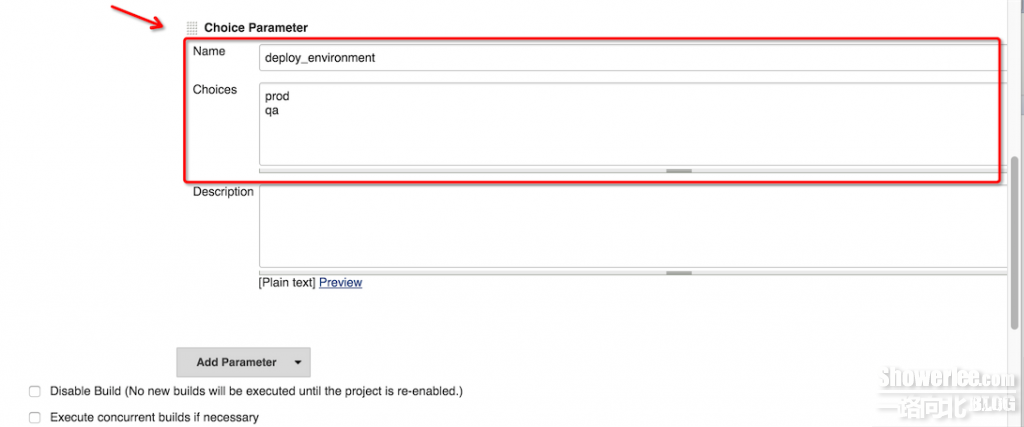
def gettags = ("git ls-remote -h [git@git.yanwenbo.cn](mailto:git@git.yanwenbo.cn):showerlee/phpcms.git").execute()

gettags.text.readLines().collect { it.split()[1].replaceAll('refs/heads/', '') }.unique()

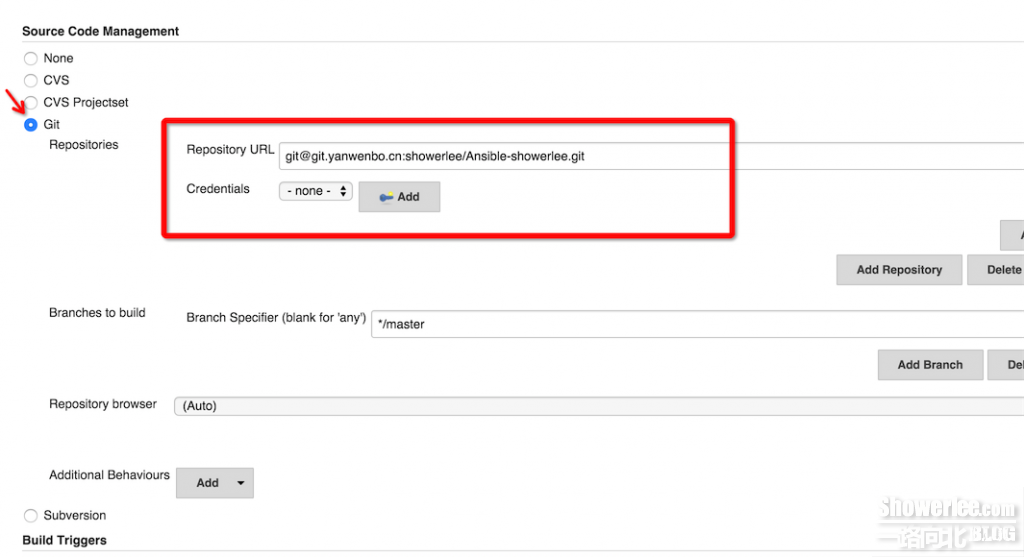
[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-11.png)

Choice Parameter也是用来给我们Job定制Build前的可选参数, 不过这里的参数可以直接写死

deploy\_environment为我们的参数名, 定义我们的部署环境名, prod, qa为我们具体的可选项, 定义我们产品的两个环境.

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-8.png)2). 源代码管理

我们可以利用Jenkins内置的Source Code Management工具去抓取远程Git或者SVN仓库的代码到本地, 这里我们抓取存放在我们Gitlab上的Playbook到Jenkins的workspace目录, 用来进行后续部署工作, 这个仓库如需认证, 我们可以在Credentials add这个仓库的用户账号密码, 其余均保持默认即可(默认Jenkins default不支持Git, 需要到其后台安装Git插件)

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-9.png)3). Execute shell进行最终的CLI部署.

这个Build模块下的Execute shell方法是Jenkins比较常用并非常核心的功能, 用来执行我们部署过程中核心的命令.

开头和结尾的set +x, set -x用来打开和关闭该部分的扩展参数及命令

开启virtualenv和加载ansible环境变量

# source /home/deploy/.virtualenv/bin/activate

# . /home/deploy/.virtualenv/ansible/hacking/env-setup -q

进入该Job的workspace目录下保存该playbook的仓库子目录下, 检查ansible版本, 并执行最终的部署命令.

cd $WORKSPACE/leon-playbook-phpcms1.1

ansible --version

ansible-playbook -i inventory/$deploy\_environment ./deploy.yml -e project=phpcms -e branch=$branch\_selector -e env=$deploy\_environment

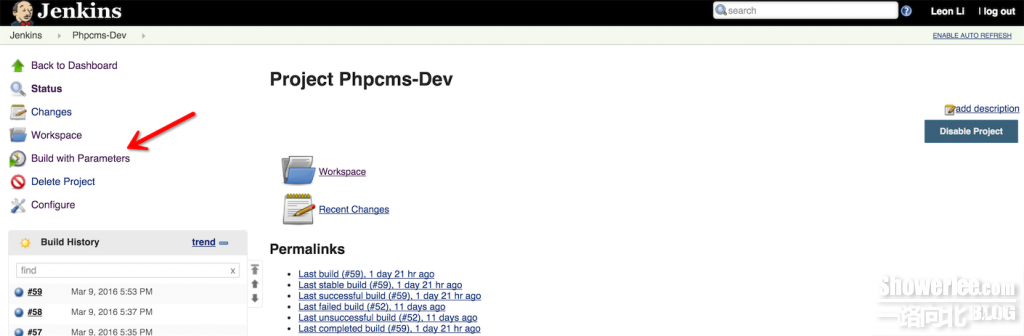
注: -i 用来自定义ansible host文件路径, ./deploy.yml为ansible-playbook入口文件, -e 后可跟给当前session添加的环境变量.

这里$deploy\_environment $branch\_selector 为该Job定义好的可选参数, 详见3-1)

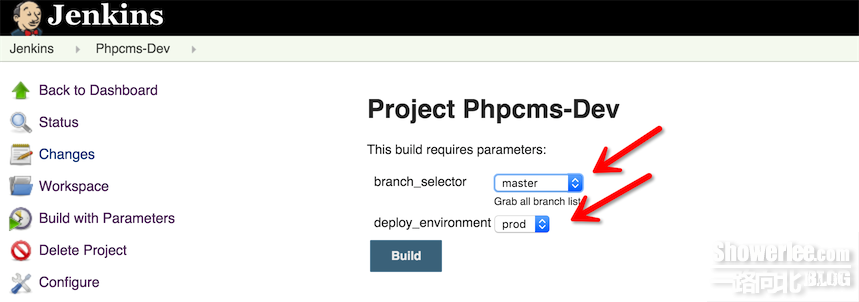
[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-10.png)

配置完毕后, save保存.

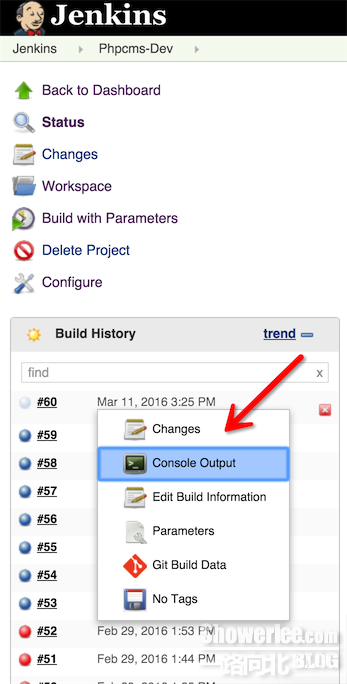
4. 执行Job.

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-12.png)

选择master分支和prod环境

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-13.png)

查看该Job最终的console output, 也就是显示我们实际在CLI下的输出结果.

[](http://www.showerlee.com/wp-content/uploads/2016/03/QQ20160311-14.png)

Console Output

Started by user Leon Li

Building in workspace /home/deploy/jenkins/workspace/Phpcms-Dev

> git rev-parse --is-inside-work-tree # timeout=10

Fetching changes from the remote Git repository

> git config remote.origin.url [git@git.yanwenbo.cn](mailto:git@git.yanwenbo.cn):showerlee/Ansible-showerlee.git # timeout=10

Fetching upstream changes from [git@git.yanwenbo.cn](mailto:git@git.yanwenbo.cn):showerlee/Ansible-showerlee.git

> git --version # timeout=10

> git fetch --tags --progress [git@git.yanwenbo.cn](mailto:git@git.yanwenbo.cn):showerlee/Ansible-showerlee.git +refs/heads/\*:refs/remotes/origin/\*

> git rev-parse refs/remotes/origin/master^{commit} # timeout=10

> git rev-parse refs/remotes/origin/origin/master^{commit} # timeout=10

Checking out Revision 6bf787dcad68219d8eee09cecb83cbca36edbef1 (refs/remotes/origin/master)

> git config core.sparsecheckout # timeout=10

> git checkout -f 6bf787dcad68219d8eee09cecb83cbca36edbef1

> git rev-list 6bf787dcad68219d8eee09cecb83cbca36edbef1 # timeout=10

[Phpcms-Dev] $ /bin/sh -xe /tmp/hudson7452069223867148990.sh

+ set +x

ansible 2.1.0 (devel 6ddea3e915) last updated 2016/02/16 16:13:32 (GMT +800)

lib/ansible/modules/core: (detached HEAD 8d126bd877) last updated 2016/02/16 16:19:09 (GMT +800)

lib/ansible/modules/extras: (detached HEAD f6c5ed987f) last updated 2016/02/16 16:19:40 (GMT +800)

config file = /home/deploy/jenkins/workspace/Phpcms-Dev/leon-playbook-phpcms1.1/ansible.cfg

configured module search path = /home/deploy/active-ansible-modules/

PLAY \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [setup] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [127.0.0.1]

TASK [deploy : Backup current source code] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [127.0.0.1]

cmd: mv /data/deploy\_dir/phpcms /data/deploy\_dir/phpcms\_master\_1457681152

start: 2016-03-11 15:25:54.774716

end: 2016-03-11 15:25:54.927415

delta: 0:00:00.152699

TASK [deploy : Get new source code] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [127.0.0.1]

TASK [deploy : Check if caches/configs/database.php exists] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [127.0.0.1]

TASK [deploy : Check if test\_dir exists] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [127.0.0.1]

TASK [deploy : debug] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [127.0.0.1] => {

"msg": "/data/deploy\_dir/phpcms\_master\_1457681152/caches/configs/database.php exists"

}

msg: /data/deploy\_dir/phpcms\_master\_1457681152/caches/configs/database.php exists

TASK [deploy : debug] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [127.0.0.1] => {

"msg": "/data/deploy\_dir/phpcms\_master\_1457681152/test\_dir exists"

}

msg: /data/deploy\_dir/phpcms\_master\_1457681152/test\_dir exists

TASK [deploy : Copy remote necessary original config to new release when Product env] \*\*\*

changed: [127.0.0.1] => (item={u'name': u'db\_config', u'dir': u'caches/configs/database.php'})

changed: [127.0.0.1] => (item={u'name': u'version\_config', u'dir': u'caches/configs/version.php'})

msg: All items completed

results: [

{

"src": "/data/deploy\_dir/phpcms\_master\_1457681152/caches/configs/database.php",

"changed": true,

"group": "deploy",

"uid": 606,

"dest": "/data/deploy\_dir/phpcms/caches/configs/database.php",

"checksum": "91869c2faa244f8c5de8a586636c6b4f3c0a2817",

"md5sum": "fd88a78a4629bca012a79d22fdcecadd",

"owner": "deploy",

"\_ansible\_no\_log": false,

"item": {

"name": "db\_config",

"dir": "caches/configs/database.php"

},

"state": "file",

"gid": 608,

"mode": "0644",

"invocation": {

"module\_args": {

"src": "/data/deploy\_dir/phpcms\_master\_1457681152/caches/configs/database.php",

"directory\_mode": null,

"force": true,

"remote\_src": true,

"dest": "/data/deploy\_dir/phpcms/caches/configs/database.php",

"selevel": null,

"seuser": null,

"setype": null,

"group": null,

"content": null,

"serole": null,

"original\_basename": null,

"delimiter": null,

"mode": "0644",

"regexp": null,

"owner": null,

"follow": false,

"validate": null,

"backup": false

}

},

"size": 302

},

{

"src": "/data/deploy\_dir/phpcms\_master\_1457681152/caches/configs/version.php",

"changed": true,

"group": "deploy",

"uid": 606,

"dest": "/data/deploy\_dir/phpcms/caches/configs/version.php",

"checksum": "d0eaedb46a36303eb3f3e2a77cc2a623062eff3c",

"md5sum": "7917d8199b7c6d5bc87ff3035a72670e",

"owner": "deploy",

"\_ansible\_no\_log": false,

"item": {

"name": "version\_config",

"dir": "caches/configs/version.php"

},

"state": "file",

"gid": 608,

"mode": "0644",

"invocation": {

"module\_args": {

"src": "/data/deploy\_dir/phpcms\_master\_1457681152/caches/configs/version.php",

"directory\_mode": null,

"force": true,

"remote\_src": true,

"dest": "/data/deploy\_dir/phpcms/caches/configs/version.php",

"selevel": null,

"seuser": null,

"setype": null,

"group": null,

"content": null,

"serole": null,

"original\_basename": null,

"delimiter": null,

"mode": "0644",

"regexp": null,

"owner": null,

"follow": false,

"validate": null,

"backup": false

}

},

"size": 127

}

]

TASK [deploy : Copy dir test\_dir to new release when Product env] \*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [127.0.0.1]

cmd: cp -a /data/deploy\_dir/phpcms\_master\_1457681152/test\_dir /data/deploy\_dir/phpcms/

start: 2016-03-11 15:26:16.966237

end: 2016-03-11 15:26:17.069705

delta: 0:00:00.103468

TASK [deploy : Get php version] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [127.0.0.1 -> localhost]

cmd: python /home/deploy/jenkins/workspace/Phpcms-Dev/leon-playbook-phpcms1.1/roles/deploy/files/get\_php\_version.py [http://www.showerlee.com](http://www.showerlee.com/)

start: 2016-03-11 15:26:17.468311

end: 2016-03-11 15:26:51.560313

delta: 0:00:34.092002

stdout: PHP/5.4.13

TASK [deploy : debug] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [127.0.0.1] => {

"msg": {

"changed": true,

"cmd": "python /home/deploy/jenkins/workspace/Phpcms-Dev/leon-playbook-phpcms1.1/roles/deploy/files/get\_php\_version.py [http://www.showerlee.com](http://www.showerlee.com/)",

"delta": "0:00:34.092002",

"end": "2016-03-11 15:26:51.560313",

"rc": 0,

"start": "2016-03-11 15:26:17.468311",

"stderr": "",

"stdout": "PHP/5.4.13",

"stdout\_lines": [

"PHP/5.4.13"

],

"warnings": []

}

}

msg: {

"changed": true,

"end": "2016-03-11 15:26:51.560313",

"stdout": "PHP/5.4.13",

"cmd": "python /home/deploy/jenkins/workspace/Phpcms-Dev/leon-playbook-phpcms1.1/roles/deploy/files/get\_php\_version.py [http://www.showerlee.com](http://www.showerlee.com/)",

"start": "2016-03-11 15:26:17.468311",

"delta": "0:00:34.092002",

"stderr": "",

"rc": 0,

"stdout\_lines": [

"PHP/5.4.13"

],

"warnings": []

}

TASK [deploy : debug] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [127.0.0.1] => {

"msg": "PHP/5.4.13"

}

msg: PHP/5.4.13

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

127.0.0.1 : ok=12 changed=5 unreachable=0 failed=0

Finished: SUCCESS

这样我们就利用Jenkins+Ansible+Gitlab, 成功部署phpcms到远程Client.