自动化运维Ansible实践

# 第 1 章 Ansible概述

## IT自动化的好处

### 团队影响

* 节省时间，提高工作效率
* 消除重复任务
* 更少的错误风险
* 改善协作和工作满意度

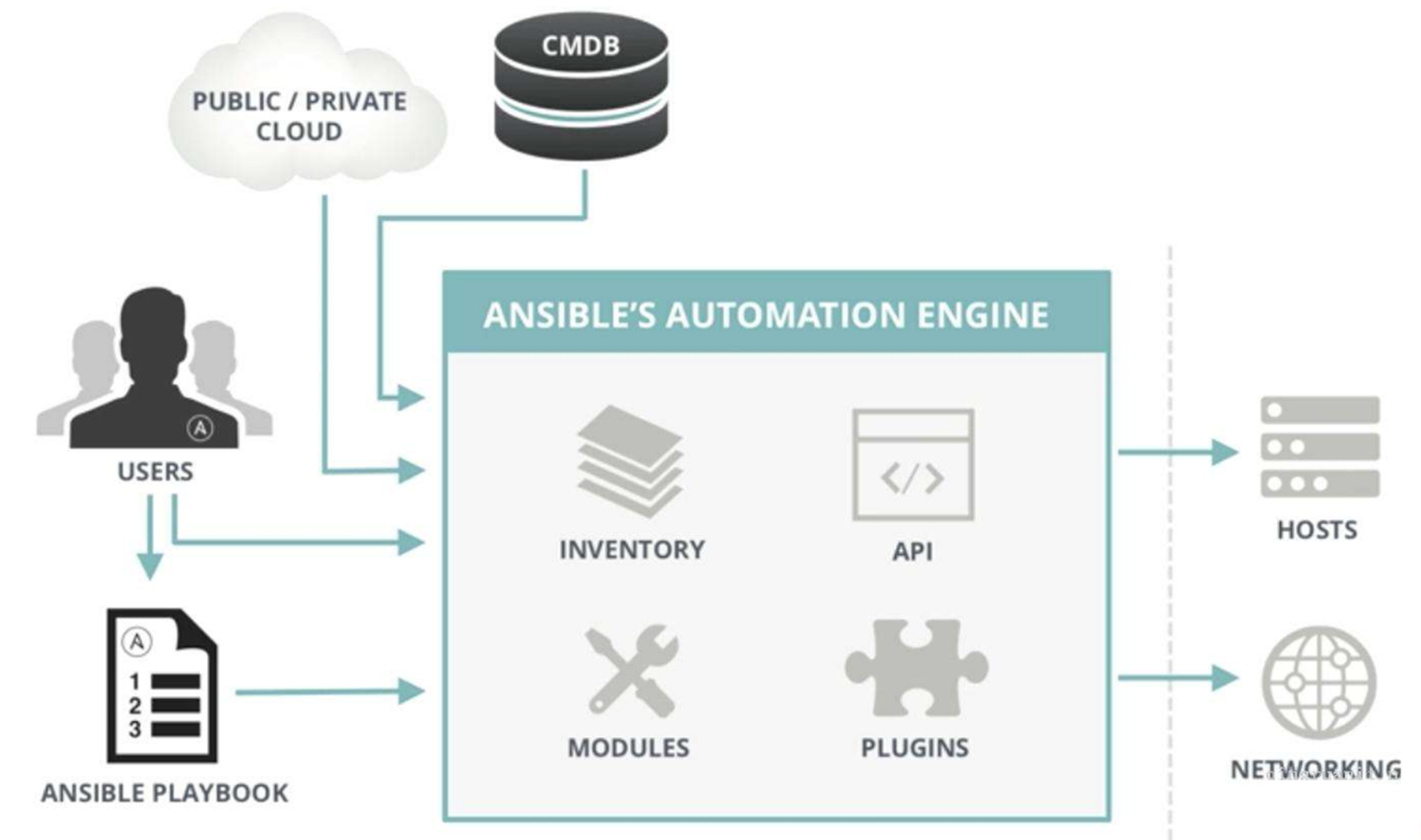
### 企业影响

* 克服复杂性
* 更多创新资源
* 加强问责制和合规性

## Ansible是什么

Ansible是一种IT自动化工具。它可以配置系统，部署软件以及协调更高级的IT任务，例如持续部署，滚动更新。Ansible适用于管理企业IT基础设施，从具有少数主机的小规模到数千个实例的企业环境。Ansible也是一种简单的自动化语言，可以完美地描述IT应用程序基础结构。

## Ansible架构



## 先来认识一下Ansible

# 第 2 章 Ansible安装与配置

## Ansible使用要求

### 服务端要求

* Python2.6/2.7/3.x
* RedHat，Debian，CentOS，OS X等。不支持Windows

### 被管理端要求

* OpenSSH
* Python2.6/2.7/3.x

## 安装Ansible

* yum install ansible （推荐）
* pip install ansible
* https://releases.ansible.com/ansible or https://github.com/ansible/ansible.git

## 配置文件

# vi /etc/ansible/ansible.cfg   
[defaults]  
  
inventory = /etc/ansible/hosts  
  
forks = 5  
  
become = root  
  
remote\_port = 22  
  
host\_key\_checking = False  
  
timeout = 10  
  
log\_path = /var/log/ansible.log  
  
private\_key\_file = /root/.ssh/id\_rsa

## Inventory（主机清单）

# 示例1：未分组的主机  
green.example.com  
blue.example.com  
192.168.100.1  
192.168.100.10  
   
# 示例2：属于webservers组主机集合  
[webservers]  
alpha.example.org  
beta.example.org  
192.168.1.100  
192.168.1.110  
www[001:006].example.com   
   
示例3：属于dbservers组主机集合  
[dbservers]  
db01.intranet.mydomain.net  
db02.intranet.mydomain.net  
10.25.1.56  
10.25.1.57  
db-[99:101]-node.example.com

主机和主机组变量：  
  
[webservers]  
192.168.1.10 ansible\_ssh\_user=root ansible\_ssh\_pass='123456’ http\_port=80  
192.168.1.11 ansible\_ssh\_user=root ansible\_ssh\_pass='123456’ http\_port=80  
  
[webservers:vars]  
http\_port=8080  
server\_name=www.ctnrs.com  
  
组变量分解到单个文件：  
  
# cat /etc/ansible/group\_vars/webservers.yml   
http\_port: 8080  
server\_name: www.ctnrs.com

# 第 3 章 ad-hoc命令

## 命令行工具常用选项,可以使用ansible --help查看帮助

格式：ansible <host-pattern> [ options ]  
  
选项：  
-a MODULE\_ARGS, --args=MODULE\_ARGS 模块参数  
-C, --check 运行检查，不执行任何操作  
-e EXTRA\_VARS, --extra-vars=EXTRA\_VARS 设置附加变量 key=value  
-f FORKS, --forks=FORKS 指定并行进程数量，默认5  
-i INVENTORY, --inventory=INVENTORY 指定主机清单文件路径  
--list-hosts 输出匹配的主机列表，不执行任何操作  
-m MODULE\_NAME, --module-name=MODULE\_NAME 执行的模块名，默认command  
--syntax-check 语法检查playbook文件，不执行任何操作  
-t TREE, --tree=TREE 将日志输出到此目录  
-v, --verbose 详细信息，-vvv更多, -vvvv debug  
--version 查看程序版本  
  
连接选项：控制谁连接主机和如何连接  
-k, --ask-pass 请求连接密码  
--private-key=PRIVATE\_KEY\_FILE, --key-file=PRIVATE\_KEY\_FILE 私钥文件  
-u REMOTE\_USER, --user=REMOTE\_USER 连接用户，默认None  
-T TIMEOUT, --timeout=TIMEOUT 覆盖连接超时时间，默认10秒  
  
提权选项：控制在目标主机以什么用户身份运行  
-b, --become 以另一个用户身份操作  
--become-method=BECOME\_METHOD 提权方法，默认sudo  
--become-user=BECOME\_USER 提权后的用户身份，默认root  
-K, --ask-become-pass 提权密码

sudo操作实例

ansible webservers -a "ls /root" -u tuwei -k --become -K

需要输入两次密码，一次是连接远程主机，一次是sudo提权

没有配置root用户，连接远程主机普通用户

[root@salt-master ansible]# ansible webservers -a "pwd" -u tuwei -k

SSH password:

192.168.132.14 | SUCCESS | rc=0 >>

/home/tuwei

192.168.132.16 | SUCCESS | rc=0 >>

/home/tuwei

## SSH密码认证

[webservers]  
192.168.1.10:22 ansible\_ssh\_user=root ansible\_ssh\_pass=’123456’  
192.168.1.11:22 ansible\_ssh\_user=root ansible\_ssh\_pass=’123456’

## SSH秘钥对认证

[webservers]  
192.168.1.10:22 ansible\_ssh\_user=root ansible\_ssh\_key=/root/.ssh/id\_rsa  
192.168.1.11:22 ansible\_ssh\_user=root

配置了密码或者秘钥认证后，在使用ansible执行命令时无需交互。

主机清单已经配置

例如：[root@salt-master ~]# ansible webservers -a "df -h"

192.168.132.14 | SUCCESS | rc=0 >>

Filesystem Size Used Avail Use% Mounted on

/dev/sda3 48G 4.5G 41G 11% /

tmpfs 383M 0 383M 0% /dev/shm

/dev/sda1 194M 29M 155M 16% /boot

192.168.132.16 | SUCCESS | rc=0 >>

Filesystem Size Used Avail Use% Mounted on

/dev/sda3 18G 5.6G 12G 34% /

tmpfs 284M 0 284M 0% /dev/shm

/dev/sda1 190M 82M 99M 46% /boot

# 第 4 章 Ansible常用模块

ansible帮助说明可以参考网站https://docs.ansible.com/

## 执行shell命令（command和shell）

默认为command，可以省略。但如果命令中有特殊字符，如<> |等，则需要使用shell模块，该模块可以执行所有shell命令

[root@salt-master ~]# ansible webservers -m shell -a "netstat -tnlp|grep httpd"

192.168.132.14 | SUCCESS | rc=0 >>

tcp 0 0 :::80 :::\* LISTEN 1384/httpd

192.168.132.16 | SUCCESS | rc=0 >>

tcp 0 0 :::80 :::\* LISTEN 1974/httpd

## ping模块

用来测试主机是否是通的，不涉及参数。

[root@salt-master ~]# ansible webservers -m ping

192.168.132.16 | SUCCESS => {

"changed": false,

"ping": "pong"

}

192.168.132.14 | SUCCESS => {

"changed": false,

"ping": "pong"

}

## script模块

在目标主机上执行管理端的shell脚本。

ansible 192.168.132.16 -m script -a '/root/test.sh'

## 文件传输（copy和file）

### copy模块主要是用来复制文件到远程目标主机。

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。

ansible webservers -m copy -a "src=/tools/nagios-3.5.1.tar.gz dest=/tmp"

[root@salt-master tools]# ansible webservers -a "ls /tmp"

192.168.132.16 | SUCCESS | rc=0 >>

VMwareDnD

VMwareTools-10.1.15-6627299.tar.gz

\_cafenv-appconfig\_

ansible\_ztwPs\_

nagios-3.5.1.tar.gz

test.sh

vmware-root

vmware-tools-distrib

192.168.132.14 | SUCCESS | rc=0 >>

ansible\_VohgWG

cmdserver.sock

nagios-3.5.1.tar.gz

test.sh

### file模块主要用于远程主机上的文件操作，包括以下选项：

force:需要在两种情况下强制创建软链接

1.源文件不存在但之后会建立的情况下

2.目标软连接已经存在，需要取消之前的软链接，然后创建新的软连接，有两个选项：yes|no

group：定义文件/目标的属组

mode:定义文件/目录的权限

owenr: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：如果文件不存在，创建目录

file：即使文件不存在，也不会被创建

link:创建软连接

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

实例：

创建目录

ansible webservers -m file -a "path=/tmp/abc123 state=directory"

创建文件

ansible webservers -m file -a "path=/tmp/abc123 state=touch"

## cron模块

cron模块，用来管理计划任务， 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：用来指定一个计划任务文件，也就是将计划任务写到远程主机上/etc/cron.d目录下，

创建一个文件对应的计划任务。 day：日（1-31，<em>，</em>/2,……）

hour：小时（0-23，<em>，</em>/2，……） minute：分钟（0-59，<em>，</em>/2，……）

month：月（1-12，<em>，</em>/2，……）

weekday：周（0-7，\*，……）

job：要执行的任务，依赖于state=present

name：定义定时任务的描述信息

special\_time： 特殊的时间范围，参数：reboot（重启时）,annually（每年）,monthly（每月）,

weekly（每周）,daily（每天）,hourly（每小时）

state：确认该任务计划是创建还是删除,有两个值可选，分别是present和absent，present表示创建定时任务，

absent表示删除定时任务，默认为present。

user：以哪个用户的身份执行job指定的任务。

使用示例：

ansible 192.168.132.14 -m cron -a 'name="job for reboot" special\_time=reboot job="/data/bootservice.sh"'

ansible 192.168.132.14 -m cron -a 'name="test" weekday="1" minute=40 hour=19 user="root" job="sh /server/scripts/test.sh"'

查看[root@mail scripts]# crontab -l

#Ansible: test

40 19 \* \* 1 sh /server/scripts/test.sh

ansible 192.168.132.14 -m cron -a 'backup="True" name="autobackup" weekday="6" minute=30 hour=1 user="root" job="/home/ixdba/backup.sh"'

ansible 192.168.132.14 -m cron -a 'name="checkhttp" minute=30 hour=12 user="root" job="/home/ixdba/check\_http.sh" cron\_file="check\_http\_for\_ansible" '

ansible 192.168.132.14 -m cron -a 'name="yum autoupdate" state=absent'

## service模块

用于管理远程主机上的服务，该模块包含如下选项：

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

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)

实例：

ansible 192.168.132.14 -m service -a "name=httpd state=stopped"

关闭httpd服务

ansible 192.168.132.14 -m service -a "name=httpd enabled=yes"

## yum模块

进行软件包安装。

实例：state可以是present，表示指定安装，或者是latest即安装最新版

absent表示卸载

ansible 192.168.132.14 -m yum -a "name=lrzsz state=latest"

## user模块

创建用户相关

ansible 192.168.132.14 -m user -a "name=test password=123456"新建用户

ansible 192.168.132.14 -m user -a "name=test password=123456 shell=/sbin/nologin" 创建用户，不让登陆

ansible 192.168.132.14 -m user -a "name=test state=absent" 删除用户

## git模块

从源代码管理系统部署，要使用git，需要目标主机已安装git

ansible 192.168.132.14 -m git -a "repo=https://github.com/ansible/ansible.git dest=/tmp/ansible"

## setup模块

ansible 192.168.132.14 -m setup 输出系统信息，在写playbook时用到

ansible 192.168.132.14 -m setup -a "filter=ansible\_nodename" 利用filter进行输出信息过滤。

# 第 5 章 Playbook基本使用

## 使用Playbook的好处

特点

* 易读的编排语言
* 适合配置管理和应用部署
* 非常适合部署复杂的工作

## 先来认识一下Playbook

自动部署Nginx

# main.yml  
---  
- hosts: webservers  
 vars:  
 hello: Ansible  
   
 tasks:  
 - name: Add repo   
 yum\_repository:  
 name: nginx  
 description: nginx repo  
 baseurl: http://nginx.org/packages/centos/7/$basearch/  
 gpgcheck: no  
 enabled: 1  
 - name: Install nginx  
 yum:  
 name: nginx  
 state: latest  
 - name: Copy nginx configuration file  
 copy:  
 src: ./site.conf  
 dest: /etc/nginx/conf.d/site.conf  
 - name: Start nginx  
 service:  
 name: nginx  
 state: started  
 - name: Create wwwroot directory  
 file:  
 dest: /var/www/html  
 state: directory  
 - name: Create test page index.html  
 shell: echo "hello {{hello}}" > /var/www/html/index.html  
  
  
# site.conf  
server {  
 listen 80;  
 server\_name www.ctnrs.com;  
 location / {  
 root /var/www/html;  
 index index.html;  
 }  
}

## .YAML语法

* 缩进表示层级关系
* 不支持制表符“tab”缩进，使用空格缩进
* 通常开头缩进 2 个空格
* 字符后缩进 1 个空格，如冒号、逗号等
* “---” 表示YAML格式，一个文件的开始
* “#”注释

playbook帮助 ansible-playbook --help

在执行前可以先检查语法ansible-playbook nginx.yml --syntax-check

## .Playbook文件结构

---  
- name: play1  
 hosts: webservers  
 remote\_user: root  
 vars:  
 var\_name: value  
 tasks:  
 - name: echo  
 shell: "echo {{var\_name}}"  
   
- name: play2  
 hosts: webservers  
 remote\_user: root  
 vars:  
 var\_name: value  
 tasks:  
 - name: echo  
 shell: "echo {{var\_name}}"  
   
   
 ---  
- hosts: webservers  
 remote\_user: root  
 vars:  
 var\_name: value  
 tasks:  
 - name: echo  
 shell: "echo {{var\_name}}"

## .在变更时执行操作（handlers）

notify：在任务结束时触发

handlers：由特定条件触发Tasks

---  
hosts: webservers  
 gather\_facts: no  
  
 tasks:  
 - name: Copy nginx configuration file  
 copy:  
 src: ./site.conf  
 dest: /etc/nginx/conf.d/site.conf  
 notify:  
 - reload nginx ------》当配置文件改变时通知重启nginx服务  
  
 handlers:  
 - name: reload nginx  
 service: name=nginx state=reloaded

## .任务控制（tags）

在每一个任务中添加tags标签，可以根据指定的tags运行相应的任务。

- hosts: webservers

gather\_facts: no

vars:

hello: Ansible

tasks:

- name: Add repo

yum\_repository:

name: nginx

description: nginx repo

baseurl: http://nginx.org/packages/centos/6/$basearch/

gpgcheck: no

enabled: 1

- name: Install nginx

yum:

name: nginx

state: latest

tags: install

- name: Copy nginx configuration file

copy:

src: ./site.conf

dest: /etc/nginx/conf.d/site.conf

tags: congiuration

- name: Start nginx

service:

name: nginx

state: started

- name: Create wwwroot directory

file:

dest: /var/www/html

state: directory

- name: Create test page index.html

shell: echo "hello {{hello}}" > /var/www/html/index.html

指定tags运行：

[root@salt-master ansible-playbook]# ansible-playbook nginx.yml --tags "congiuration"

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

TASK [Copy nginx configuration file] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [192.168.132.16]

ok: [192.168.132.14]

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

192.168.132.14 : ok=1 changed=0 unreachable=0 failed=0

192.168.132.16 : ok=1 changed=0 unreachable=0 failed=0

## .Playbook文件调试

语法检查：ansible-playbook main.yml --syntax-check  
  
打印语句：  
---  
- hosts: webservers  
 tasks:  
 - debug:  
 msg: {{group\_names}}  
 - debug:   
 msg: {{inventory\_hostname}}  
 - debug:  
 msg: {{ansible\_hostname}}

## .案例：自动部署Tomcat

- hosts: 192.168.132.16

gather\_facts: no

vars:

tomcat\_version: 8.5.37

tomcat\_install\_dir: /usr/local

tasks:

- name: Install jdk1.8

yum: name=java-1.8.0-openjdk state=present

- name: Download tomcat

get\_url: url=http://mirrors.hust.edu.cn/apache/tomcat/tomcat-8/v{{ tomcat\_version }}/bin/apache-tomcat-{{ tomcat\_version }}.tar.gz dest=/tmp

- name: Unarchive tomcat-{{ tomcat\_version }}.tar.gz

unarchive:

src: /tmp/apache-tomcat-{{ tomcat\_version }}.tar.gz

dest: "{{ tomcat\_install\_dir }}"

copy: no

- name: Start tomcat

shell: cd {{ tomcat\_install\_dir }} &&

mv apache-tomcat-{{ tomcat\_version }} tomcat8 &&

cd tomcat8/bin && nohup ./startup.sh &

# 第 6 章 Playbook定义变量与使用

## .命令行

## .在Inventory中定义

## .在Playbook中定义

# 在Playbook中定义变量  
---  
hosts: webservers  
 gather\_facts: no  
 vars:  
 var\_name: value  
 var\_name: value  
 tasks:  
 - name: hello  
 shell: "echo {{var\_name}}"

## .在Role中定义

## .注册变量（register）

# 注册变量  
---  
hosts: webservers   
 gather\_facts: no  
 tasks:  
 - name: Get date   
 command: date +"%F\_%T"  
 register: date\_output  
 - name: Echo date\_output  
 command: touch /tmp/{{date\_output.stdout}}

## .系统信息变量（facts）

# 系统变量  
---  
hosts: webservers   
 tasks:  
 - name: Get hostname  
 debug: msg={{ansible\_hostname}}

# 第 7 章 Playbook文件复用

## .include & import 区别

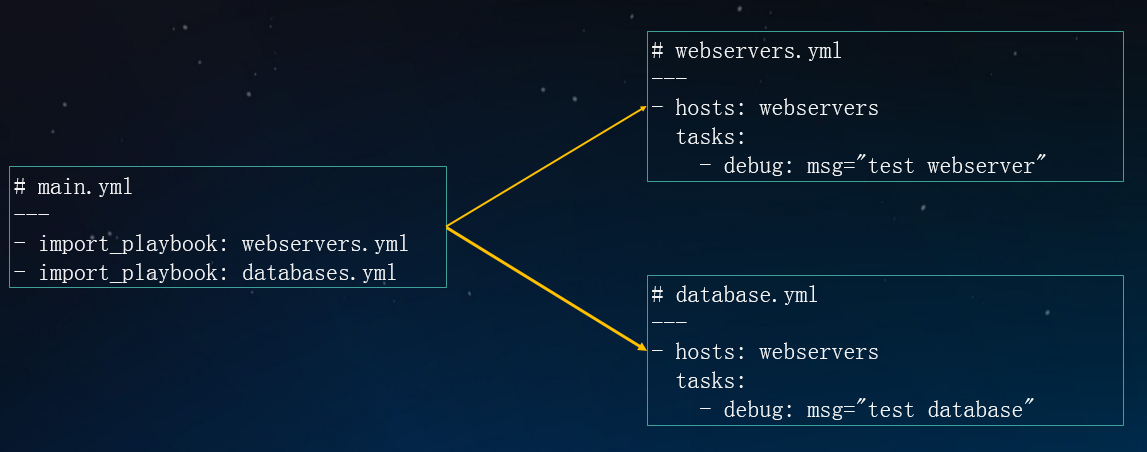
include\*（动态）：在运行时导入

* --list-tags，--list-tasks不会显示到输出
* 不能使用notify触发来自include\*内处理程序名称（handlers）

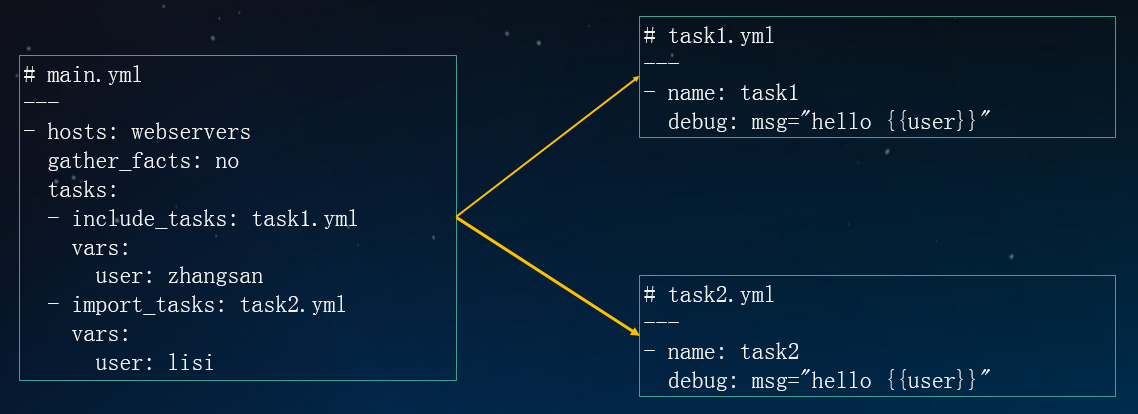
import\*（静态）：在Playbook解析时预先导入

* 不能与循环一起使用
* 将变量用于目标文件或角色名称时，不能使用inventory（主机/主机组等）中的变量

## .import\_playbook



## .include*tasks & import*tasks



# 第 8 章 Playbook流程控制

## .条件

- hosts: webservers  
   
 tasks:  
 - name: Host 192.168.132.16 run this task  
 debug: msg="{{ansible\_default\_ipv4.address}}"  
 when: ansible\_default\_ipv4.address == '192.168.132.16'

只执行该任务，跳过其他主机。

根据不同发行版安装apache服务

- hosts: webservers  
   
 tasks:  
 - name: Update apache version - yum  
 yum: name=httpd state=present  
 when: ansible\_pkg\_mgr == 'yum'  
 notify: restart httpd  
   
 - name: Update apache version - apt  
 apt: name=apache2 state=present update\_cache=yes  
 when: ansible\_pkg\_mgr == 'apt'  
 notify: restart apache2  
   
 handlers:  
 - name: restart httpd  
 service: name=httpd state=restared  
 handlers:  
 - name: restart apache2  
 service: name=apache2 state=restared

tasks:  
 - name: "shut down CentOS 6 and Debian 7 systems"  
 command: /sbin/shutdown -t now  
 when: (ansible\_distribution == "CentOS" and ansible\_distribution\_major\_version == "6") or  
 (ansible\_distribution == "Debian" and ansible\_distribution\_major\_version == "7")

tasks:  
 - name: "shut down CentOS 6 systems"  
 command: /sbin/shutdown -t now  
 when:  
 - ansible\_distribution == "CentOS"  
 - ansible\_distribution\_major\_version == "6"

## .循环

- name: with\_list  
 debug:  
 msg: "{{ item }}"  
 with\_list:  
 - one  
 - two  
  
- name: with\_list -> loop  
 debug:  
 msg: "{{ item }}"  
 loop:  
 - one  
 - two

- name: with\_items  
 debug:  
 msg: "{{ item }}"  
 with\_items: "{{ items }}"  
  
- name: with\_items -> loop  
 debug:  
 msg: "{{ item }}"  
 loop: "{{ items|flatten(levels=1) }}"

创建用户：

- hosts: webservers

gather\_facts: no

tasks:

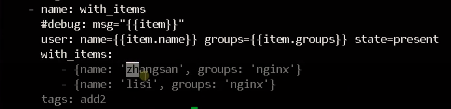
- name: with\_list

user: name={{item}} state=present

with\_list:

- test1

- test2



# 第 9 章 Playbook模板（jinja2）

## .条件和循环

# test.yml   
---  
- hosts: webservers  
 vars:  
 hello: Ansible  
   
 tasks:  
 - template: src=f.j2 dest=/tmp/f.j2  
  
  
# f.j2   
{% set list=['one', 'two', 'three'] %}  
   
{% for i in list %}  
 {% if i == 'two' %}  
 -> two  
 {% elif loop.index == 3 %}  
 -> 3  
 {% else %}  
 {{i}}  
 {% endif %}  
{% endfor %}   
   
{{ hello }}

{% set dict={'zhangsan': '26', 'lisi': '25'} %}  
{% for key, value in dict.iteritems() %}  
 {{key}} -> {{value}}  
{% endfor %}

## .案例：管理Nginx配置文件

# main.yml   
---  
- hosts: webservers  
 gather\_facts: no  
 vars:  
 http\_port: 80  
 server\_name: www.ctnrs.com  
   
 tasks:  
 - name: Copy nginx configuration file   
 template: src=site.conf.j2 dest=/etc/nginx/conf.d/www.ctnrs.com.conf  
 notify: reload nginx  
   
 handlers:  
 - name: reload nginx  
 service: name=nginx state=reloaded

# site.conf.j2   
{% set list=[10, 12, 13, 25, 31] %}  
upstream {{server\_name}} {  
 {% for i in list %}  
 server 192.168.1.{{i}}:80;  
 {% endfor %}  
}  
server {  
 listen {{ http\_port }};  
 server\_name {{ server\_name }};  
   
 location / {  
 proxy\_pass http://{{server\_name}};  
 }   
}

# 第 10 章 角色（roles）

## .Roles目录结构

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

* tasks - 包含角色要执行的主要任务列表
* handlers - 包含角色使用的处理程序
* defaults - 角色默认的变量
* vars - 角色其他的变量
* files - 角色部署时用到的文件
* templates - 角色部署时用到的模板
* meta - 角色定义的一些元数据

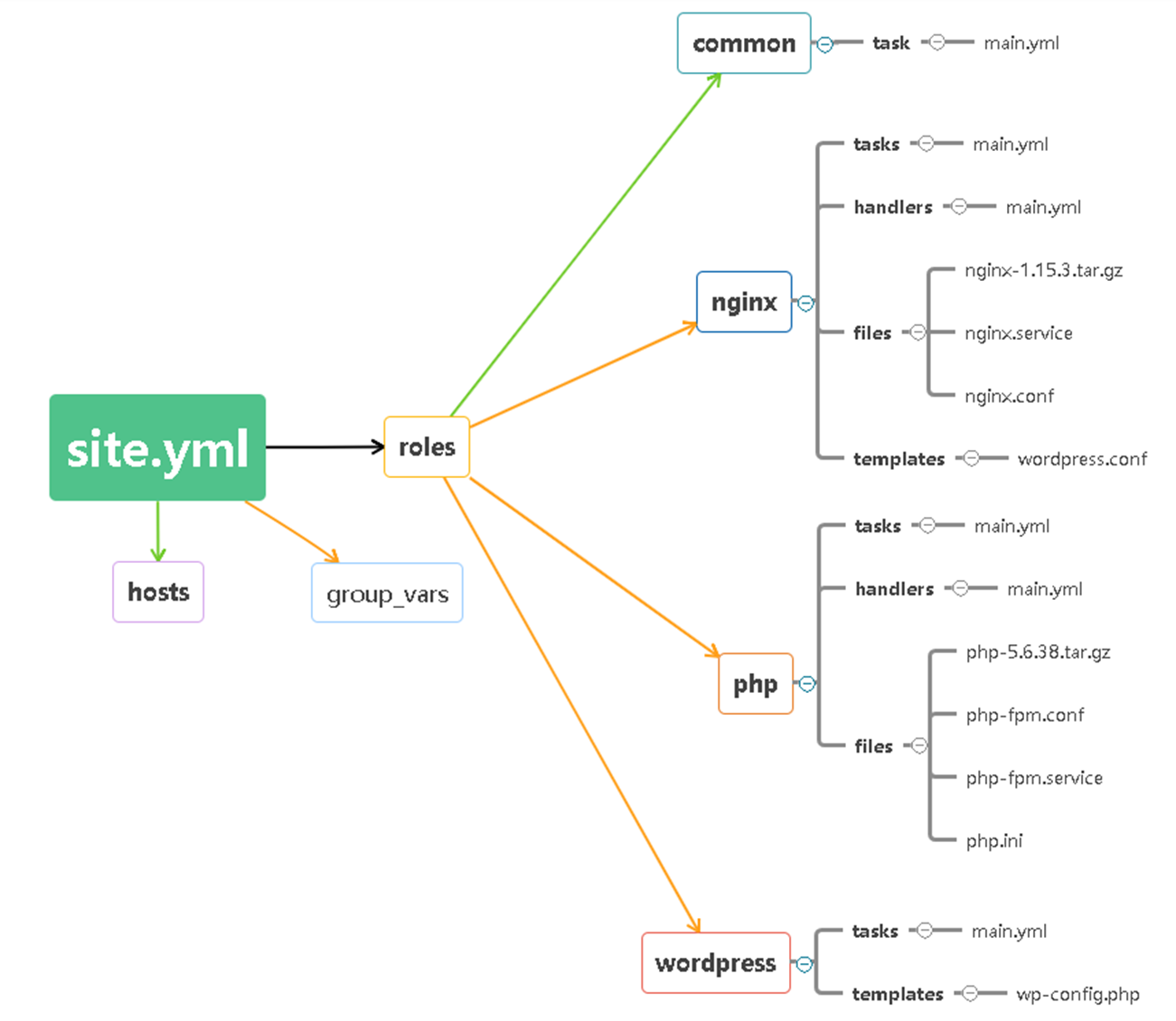
## .Roles基本使用

---  
- hosts: webservers  
 roles:  
 - common  
 - nginx  
 - php

---  
- hosts: webservers  
 roles:  
 - common  
 - role: nginx  
 vars:  
 dir: '/opt/a'  
 app\_port: 5000  
 - role: php  
 vars:  
 dir: '/opt/b'  
 app\_port: 5001

---  
- hosts: webservers  
 roles:  
 - role: common  
 tags: ["common"]  
 - role: nginx  
 tags: ["nginx"]  
 - role: php  
 tags: ["php"]

## .案例：部署Web服务器



最佳实践：https://docs.ansible.com/ansible/latest/user*guide/playbooks*best\_practices.html

示例参考：https://github.com/ansible/ansible-examples