马哥教育
IT人的高薪职业学院

运维自动化之ANSIBLE

讲师：王晓春

运维自动化之ANSIBLE

本章内容

- 运维自动化发展历程及技术应用
- Ansible架构和相关命令使用
- Ansible常用模块详解
- Ansible playbook基础
- Playbook变量、tags、handlers使用
- Playbook模板 templates
- Playbook条件判断 when
- Playbook字典 with_items
- Ansible Roles

1 自动化运维应用场景

1.1 云计算运维工程师核心职能

平台架构组建

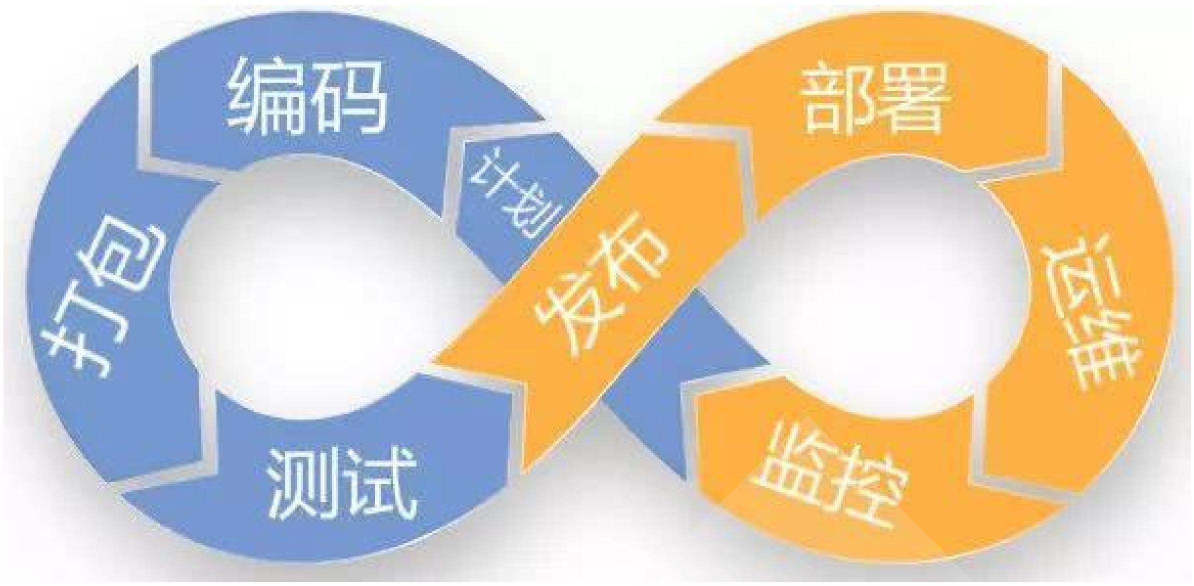
负责参与并审核架构设计的合理性和可运维性，搭建运维平台技术架构，通过开源解决方案，以确保在产品发布之后能高效稳定的运行，保障并不断提升服务的可用性，确保用户数据安全，提升用户体验。

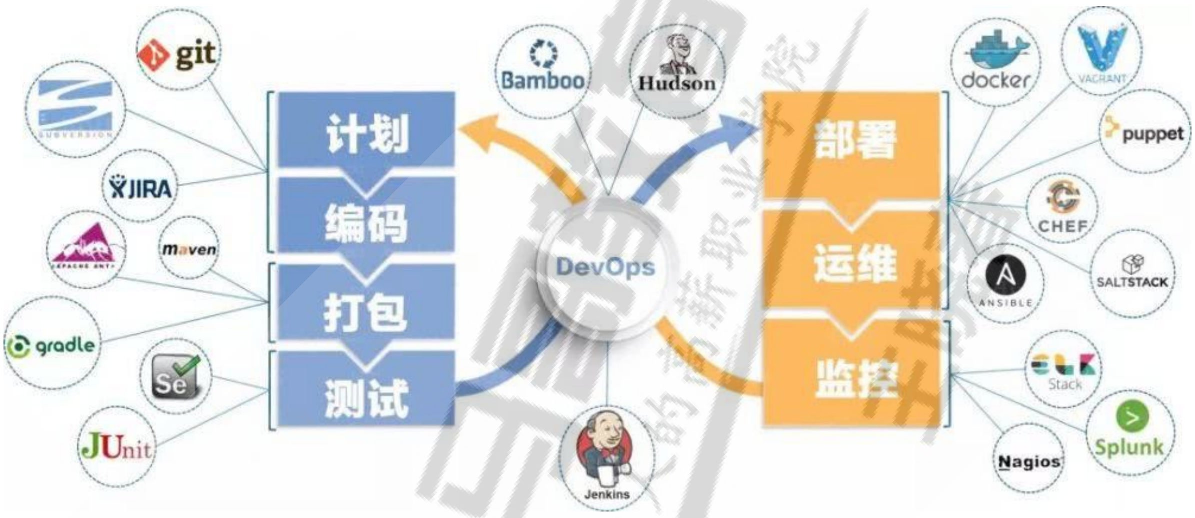
日常运营保障

负责用运维技术或者运维平台确保产品可以高效的发布上线，负责保障产品7*24H稳定运行，在此期间对出现的各种问题可以快速定位并解决；在日常工作中不断优化系统架构和部署的合理性，以提升系统服务的稳定性。

性能、效率优化

用自动化的工具/平台提升软件在研发生命周期中的工程效率。不断优化系统架构、提升部署效率、优化资源利用率支持产品的不断迭代，需要不断的进行架构优化调整。以确保整个产品能够在功能不断丰富和复杂的条件下，同时保持高可用性。

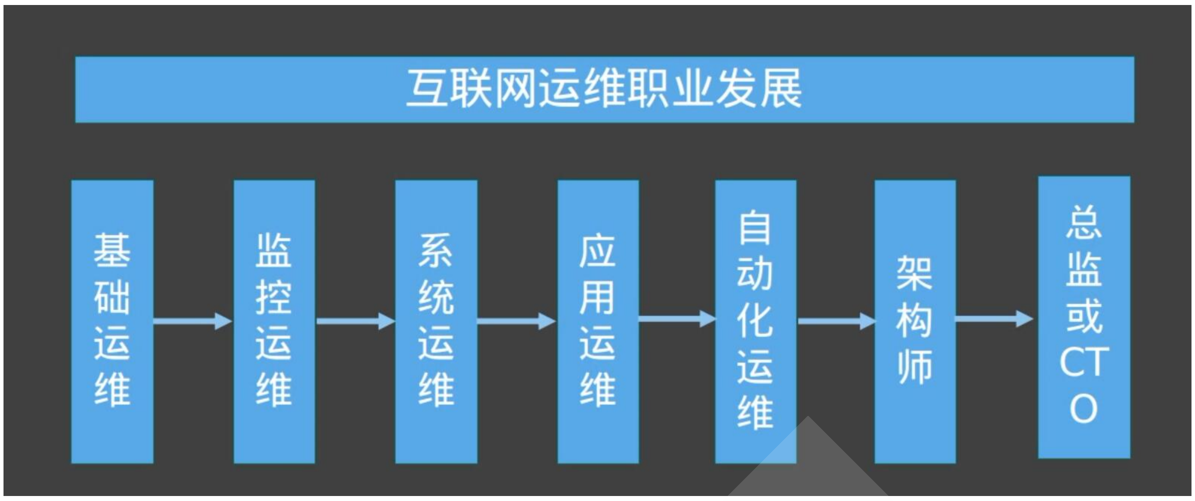




相关工具

- 代码管理（SCM）：GitHub、GitLab、BitBucket、SubVersion
- 构建工具：maven、Ant、Gradle
- 自动部署：Capistrano、CodeDeploy
- 持续集成（CI）：Jenkins、Travis
- 配置管理：Ansible、SaltStack、Chef、Puppet
- 容器：Docker、Podman、LXC、第三方厂商如AWS
- 编排：Kubernetes、Core、Apache Mesos
- 服务注册与发现：Zookeeper、etcd、Consul
- 脚本语言：python、ruby、shell
- 日志管理：ELK、Logentries
- 系统监控：Prometheus、Zabbix、Datadog、Graphite、Ganglia、Nagios
- 性能监控：AppDynamics、New Relic、Splunk
- 压力测试：JMeter、Blaze Meter、loader.io
- 应用服务器：Tomcat、JBoss、IIS
- Web服务器：Apache、Nginx
- 数据库：MySQL、Oracle、PostgreSQL等关系型数据库；mongoDB、redis等NoSQL数据库
- 项目管理（PM）：Jira、Asana、Taiga、Trello、Basecamp、Pivotal Tracker

1.2 运维职业发展路线

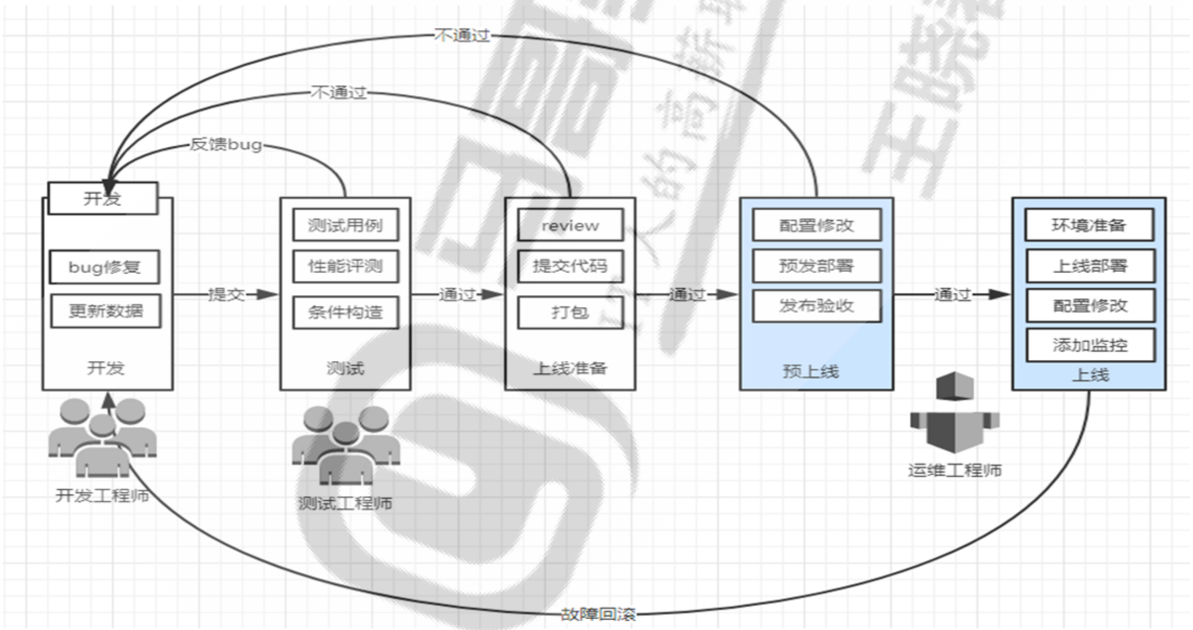


运维的未来是什么?

一切皆自动化

"运维的未来是，让研发人员能够借助工具、自动化和流程，并且让他们能够在运维干预极少的情况下部署和运营服务，从而实现自助服务。每个角色都应该努力使工作实现自动化。"——《运维的未来》

1.3 企业实际应用场景分析



1.3.1 Dev开发环境

使用者：程序员

功能：程序员个人的办公电脑或项目的开发测试环境，部署开发软件，测试个人或项目整体的BUG的环境

管理者：程序员

1.3.2 测试环境

使用者：QA测试工程师

功能：测试经过Dev环境测试通过的软件的功能和性能，判断是否达到项目的预期目标，生成测试报告

管理者：运维

说明：测试环境往往有多套,测试环境满足测试功能即可，不宜过多

- 测试人员希望测试环境有多套,公司的产品多产品线并发，即多个版本，意味着多个版本同步测试
- 通常测试环境有多少套和产品线数量保持一样

1.3.3 预发布环境

使用者：运维

功能：使用和生产环境一样的数据库，缓存服务等配置，测试是否正常

1.3.4 发布环境

包括代码发布机，有些公司为堡垒机（安全屏障）

使用者：运维

功能：发布代码至生产环境

管理者：运维（有经验）

发布机：往往需要有2台（主备）

1.3.5 生产环境

使用者：运维，少数情况开放权限给核心开发人员，极少数公司将权限完全开放给开发人员并其维护

功能：对用户提供公司产品的服务

管理者：只能是运维

生产环境服务器数量：一般比较多，且应用非常重要。往往需要自动工具协助部署配置应用

1.3.6 灰度环境

属于生产环境的一部分

使用者：运维

功能：在全量发布代码前将代码的功能面向少量精准用户发布的环境,可基于主机或用户执行灰度发布

案例：共100台生产服务器，先发布其中的10台服务器，这10台服务器就是灰度服务器

管理者：运维

灰度环境：往往该版本功能变更较大，为保险起见特意先让一部分用户优化体验该功能，待这部分用户使用没有重大问题的时候，再全量发布至所有服务器

1.4 程序发布

程序发布要求：

- 不能导致系统故障或造成系统完全不可用
- 不能影响用户体验

预发布验证：

新版本的代码先发布到服务器（跟线上环境配置完全相同，只是未接入到调度器）

灰度发布：

基于主机，用户，业务

发布路径：

```
/webapp/tuangou
/webapp/tuangou-1.1
/webapp/tuangou-1.2
```

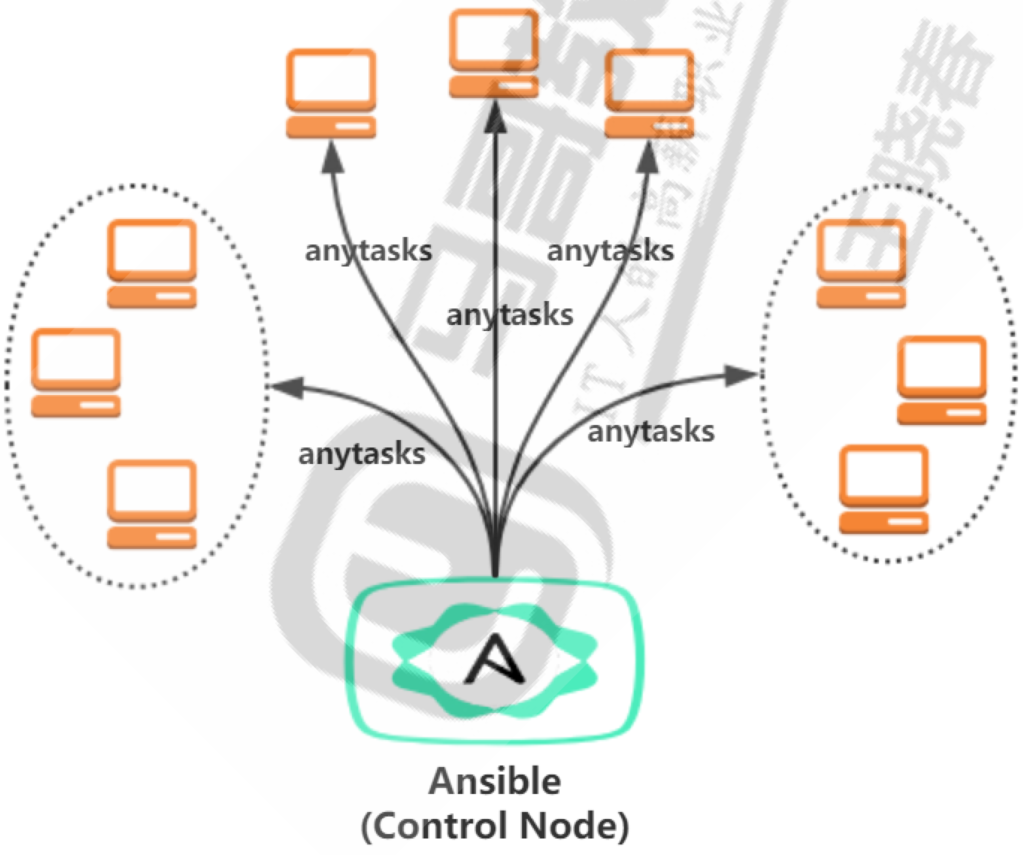
发布过程：

1. 在调度器上下线一批主机(标记为maintenance 状态)
2. 关闭服务
3. 部署新版本的应用程序
4. 启动服务
5. 在调度器上启用这一批服务器

自动化灰度发布：

- 脚本
- 发布平台

1.5 自动化运维应用场景

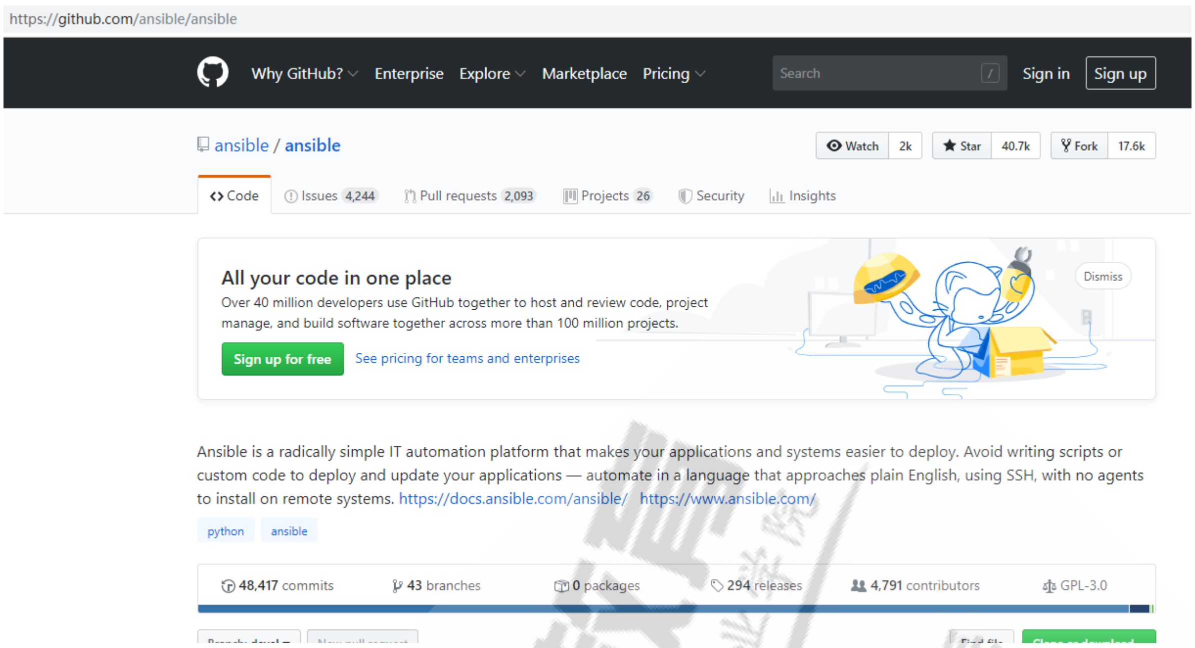


- 文件传输
- 应用部署
- 配置管理
- 任务流编排

1.6 常用自动化运维工具

- Ansible：python，Agentless，中小型应用环境
- Saltstack：python，一般需部署agent，执行效率更高
- Puppet：ruby, 功能强大，配置复杂，重型，适合大型环境

- Fabric：python，agentless
- Chef：ruby，国内应用少
- Cfengine
- func


https://github.com/ansible/ansible

ansible / ansible

All your code in one place

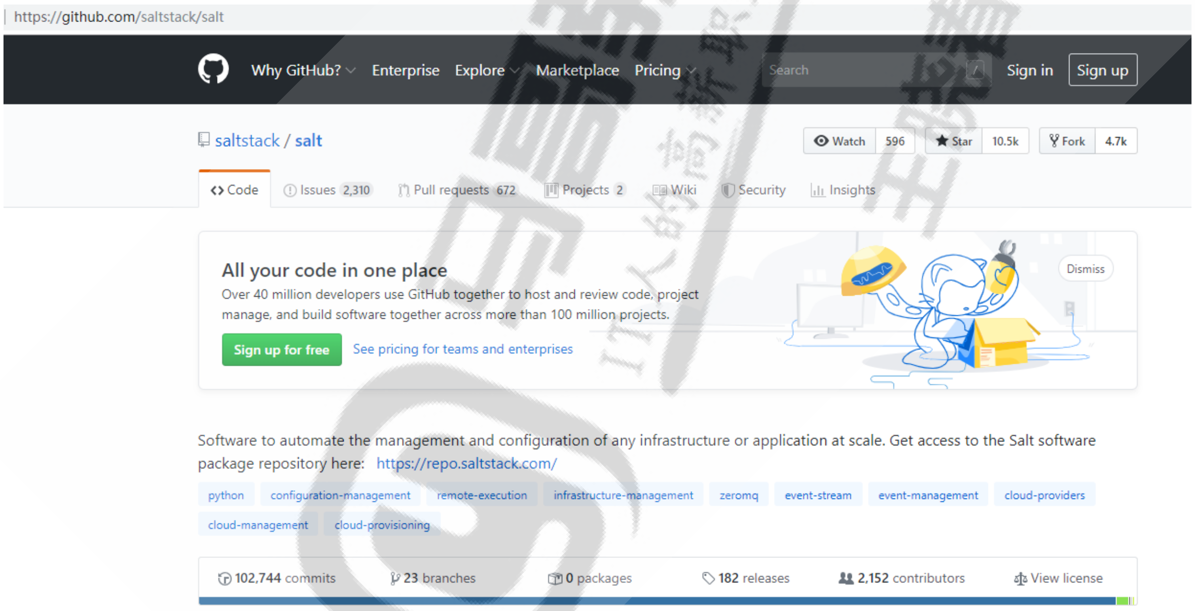
Over 40 million developers use GitHub together to host and review code, project manage, and build software together across more than 100 million projects.

Sign up for free See pricing for teams and enterprises

Ansible is a radically simple IT automation platform that makes your applications and systems easier to deploy. Avoid writing scripts or custom code to deploy and update your applications — automate in a language that approaches plain English, using SSH, with no agents to install on remote systems. https://docs.ansible.com/ansible/ https://www.ansible.com/

python ansible

48,417 commits 43 branches 0 packages 294 releases 4,791 contributors GPL-3.0


https://github.com/saltstack/salt

saltstack / salt

All your code in one place

Over 40 million developers use GitHub together to host and review code, project manage, and build software together across more than 100 million projects.

Sign up for free See pricing for teams and enterprises

Software to automate the management and configuration of any infrastructure or application at scale. Get access to the Salt software package repository here: https://repo.saltstack.com/

python configuration-management remote-execution infrastructure-management zeromq event-stream event-management cloud-providers cloud-management cloud-provisioning

102,744 commits 23 branches 0 packages 182 releases 2,152 contributors View license

同类自动化工具GitHub关注程度（2016-07-10）

自动化运维工具	Watch（关注）	Star（点赞）	Fork（复制）	Contributors(贡献者)
Ansible	1387	17716	5356	1428
Saltstack	530	6678	3002	1520
Puppet	463	4044	1678	425
Chef	383	4333	1806	464
Fabric	379	7334	1235	116

2 Ansible 介绍和架构

公司计划在年底做一次大型市场促销活动，全面冲刺下交易额，为明年的上市做准备。公司要求各业务组对年底大促做准备，运维部要求所有业务容量进行三倍的扩容，并搭建出多套环境可以共开发和测试人员做测试，运维老大为了在年底有所表现，要求运维部门同学尽快实现，当你接到这个任务时，有没有更快的解决方案？

2.1 Ansible发展史

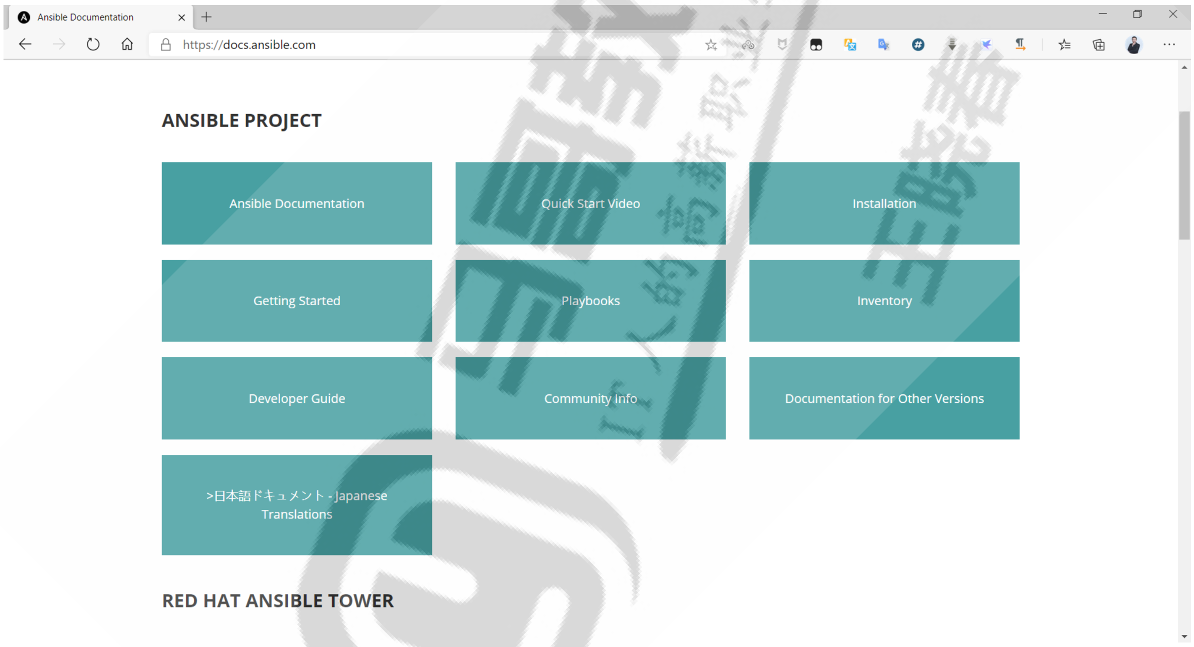
作者：Michael DeHaan（ Cobbler 与 Func 作者）

ansible 的名称来自科幻小说《安德的游戏》中跨越时空的即时通信工具，使用它可以在相距数光年的距离，远程实时控制前线的舰队战斗

2012-03-09，发布0.0.1版，2015-10-17，Red Hat宣布1.5亿美元收购

官网：https://www.ansible.com/

官方文档：https://docs.ansible.com/



2.2. ansible 功能

- 批量执行远程命令,可以对远程的多台主机同时进行命令的执行
- 批量安装和配置软件服务，可以对远程的多台主机进行自动化的方式配置和管理各种服务
- 编排高级的企业级复杂的IT架构任务, Ansible的Playbook和role可以轻松实现大型的IT复杂架构
- 提供自动化运维工具的开发API, 有很多运维工具,如jumpserver就是基于 ansible 实现自动化管理功能

2.3 Ansible 特性

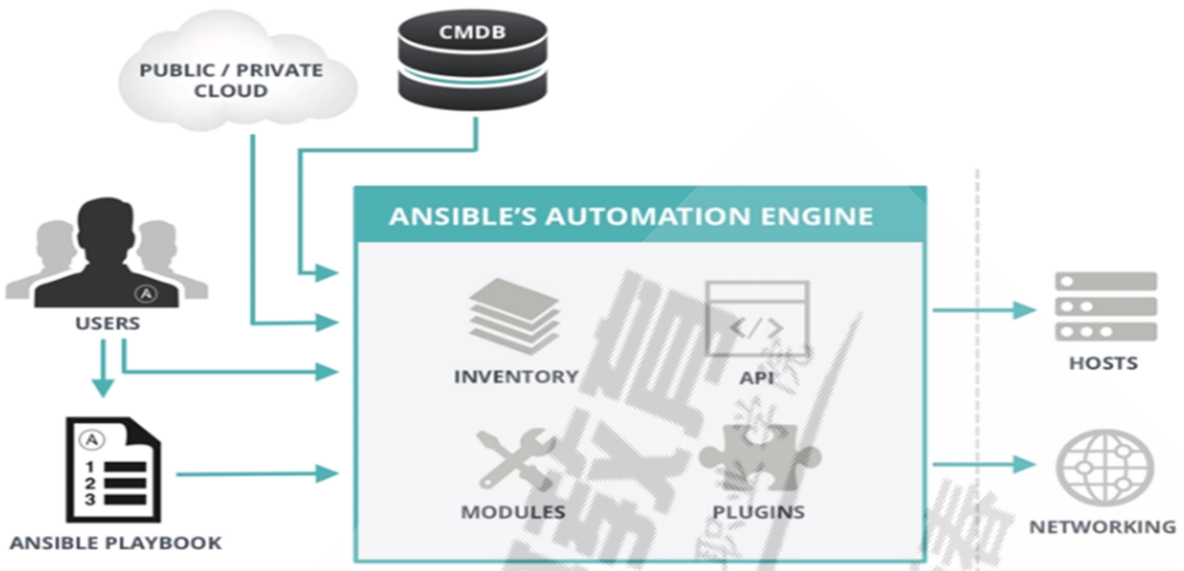
- 模块化：调用特定的模块完成特定任务，支持自定义模块，可使用任何编程语言写模块
- Paramiko（python对ssh的实现），PyYAML，Jinja2（模板语言）三个关键模块
- 基于Python语言实现
- 部署简单，基于python和SSH(默认已安装)，agentless，无需代理不依赖PKI（无需ssl）
- 安全，基于OpenSSH
- 幂等性：一个任务执行1遍和执行n遍效果一样，不因重复执行带来意外情况,此特性非绝对

- 支持playbook编排任务，YAML格式，编排任务，支持丰富的数据结构
- 较强大的多层解决方案 role

2.4 Ansible 架构

2.4.1 Ansible 组成

组合INVENTORY、API、MODULES、PLUGINS的绿框，为ansible命令工具，其为核心执行工具



- INVENTORY：Ansible管理主机的清单/etc/anaible/hosts
- MODULES：Ansible执行命令的功能模块，多数为内置核心模块，也可自定义
- PLUGINS：模块功能的补充，如连接类型插件、循环插件、变量插件、过滤插件等，该功能不常用
- API：供第三方程序调用的应用程序编程接口

2.4.2 Ansible 命令执行来源

- USER 普通用户，即SYSTEM ADMINISTRATOR
- PLAYBOOKS：任务剧本（任务集），编排定义Ansible任务集的配置文件，由Ansible顺序依次执行，通常是JSON格式的YML文件
- CMDB（配置管理数据库） API 调用
- PUBLIC/PRIVATE CLOUD API调用
- USER-> Ansible Playbook -> Ansibile

2.4.3 注意事项

- 执行ansible的主机一般称为管理端, 主控端，中控，master或堡垒机
- 主控端Python版本需要2.6或以上
- 被控端Python版本小于2.4，需要安装python-simplejson
- 被控端如开启SELinux需要安装libselinux-python
- windows 不能做为主控端

3 Ansible 安装和入门

3.1 Ansible安装

ansible的安装方法有多种

官方文档

```
https://docs.ansible.com/ansible/latest/installation_guide/intro_installation.ht
ml
```

下载

```
https://releases.ansible.com/ansible/
```

pip 下载

```
https://pypi.org/project/ansible/
```

3.1.1 包安装方式

```
#CentOS 的EPEL源的rpm包安装
[root@centos ~]#yum install ansible

#ubuntu 安装
[root@ubuntu ~]#apt -y install  ansible
```

范例：查看ansible版本

```
[root@centos8 ~]#yum info ansible
Last metadata expiration check: 0:06:46 ago on Thu 21 Jan 2021 07:36:23 PM CST.
Available Packages
Name         : ansible
Version      : 2.9.16
Release      : 1.el8
Architecture : noarch
Size         : 17 M
Source       : ansible-2.9.16-1.el8.src.rpm
Repository   : epel
Summary      : SSH-based configuration management, deployment, and task execution
system
URL          : http://ansible.com
License      : GPLv3+
Description  : Ansible is a radically simple model-driven configuration
management,
             : multi-node deployment, and remote task execution system. Ansible
works
             : over SSH and does not require any software or daemons to be
installed
             : on remote nodes. Extension modules can be written in any language
and
             : are transferred to managed machines automatically.

[root@centos7 ~]#yum info ansible
Available Packages
Name        : ansible
Arch        : noarch
Version     : 2.9.16
Release     : 1.el7
Size        : 17 M
Repo        : epel/7/x86_64
```

```
Summary     : SSH-based configuration management, deployment, and task execution
system
URL         : http://ansible.com
License     : GPLv3+
Description : Ansible is a radically simple model-driven configuration
management,
            : multi-node deployment, and remote task execution system. Ansible
works
            : over SSH and does not require any software or daemons to be
installed
            : on remote nodes. Extension modules can be written in any language
and
            : are transferred to managed machines automatically.

root@ubuntu2004:~# apt show ansible
Package: ansible
Version: 2.9.6+dfsg-1
Priority: optional
Section: universe/admin
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Harlan Lieberman-Berg <hlieberman@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 58.0 MB
Depends: python3-cryptography, python3-jinja2, python3-yaml, python3:any,
openssh-client | python3-paramiko, python3-crypto, python3-distutils, python3-
dnspython, python3-httplib2, python3-netaddr
Recommends: python3-argcomplete, python3-jmespath, python3-kerberos, python3-
libcloud, python3-selinux, python3-winrm, python3-xmltodict
Suggests: cowsay, sshpass
Homepage: https://www.ansible.com
Download-Size: 5,794 kB
APT-Sources: http://mirrors.aliyun.com/ubuntu focal/universe amd64 Packages
Description: Configuration management, deployment, and task execution system
 Ansible is a radically simple model-driven configuration management,
 multi-node deployment, and remote task execution system. Ansible works
 over SSH and does not require any software or daemons to be installed
 on remote nodes. Extension modules can be written in any language and
 are transferred to managed machines automatically.

[root@ubuntu1804 ~]#apt show ansible
Package: ansible
Version: 2.5.1+dfsg-1ubuntu0.1
Priority: optional
Section: universe/admin
Origin: Ubuntu
Maintainer: Ubuntu Developers <ubuntu-devel-discuss@lists.ubuntu.com>
Original-Maintainer: Harlan Lieberman-Berg <hlieberman@debian.org>
Bugs: https://bugs.launchpad.net/ubuntu/+filebug
Installed-Size: 26.9 MB
Depends: python-cryptography, python-jinja2, python-paramiko, python-pkg-
resources, python-yaml, python:any (<< 2.8), python:any (>= 2.7.5-5~), python-
crypto, python-httplib2, python-netaddr
Recommends: python-jmespath, python-kerberos, python-libcloud, python-selinux,
python-winrm (>= 0.1.1), python-xmltodict
Suggests: cowsay, sshpass
Homepage: https://www.ansible.com
Download-Size: 3,198 kB
```

```
APT-Sources: http://mirrors.aliyun.com/ubuntu bionic-security/universe amd64
Packages
Description: Configuration management, deployment, and task execution system
 Ansible is a radically simple model-driven configuration management,
 multi-node deployment, and remote task execution system. Ansible works
 over SSH and does not require any software or daemons to be installed
 on remote nodes. Extension modules can be written in any language and
 are transferred to managed machines automatically.

N: There is 1 additional record. Please use the '-a' switch to see it
```

范例：ubuntu18.04安装最新版的ansible

```
[root@ubuntu1804 ~]#apt update
[root@ubuntu1804 ~]#apt install software-properties-common
[root@ubuntu1804 ~]#apt-add-repository --yes --update ppa:ansible/ansible
[root@ubuntu1804 ~]#apt install ansible
[root@ubuntu1804 ~]#ansible --version
ansible 2.9.17
  config file = /etc/ansible/ansible.cfg
  configured module search path = [u'/root/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/dist-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.17 (default, Sep 30 2020, 13:38:04) [GCC 7.5.0]
```

范例： CentOS8 安装ansible

```
[root@centos8 ~]#yum install ansible -y
[root@centos8 ~]#ansible --version
ansible 2.9.16
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Apr 16 2020, 01:36:27) [GCC 8.3.1 20191121
(Red Hat 8.3.1-5)]
```

3.1.2 编译安装

```
yum -y install python-jinja2 PyYAML python-paramiko python-babel python-crypto

wget https://releases.ansible.com/ansible/ansible-1.5.4.tar.gz

tar xf ansible-1.5.4.tar.gz
cd ansible-1.5.4
python setup.py build
python setup.py install
mkdir /etc/ansible
cp -r examples/* /etc/ansible
```

3.1.3 Git方式

```
git clone git://github.com/ansible/ansible.git --recursive
cd ./ansible
source ./hacking/env-setup
```

3.1.4 pip 安装

pip 是安装Python包的管理器，类似 yum

```
[root@centos7 ~]#yum install python-pip
[root@centos7 ~]#pip install  --upgrade pip
[root@centos7 ~]#pip install ansible --upgrade
[root@centos7 ~]#ansible --version
/usr/lib64/python2.7/site-packages/cryptography/__init__.py:39:
CryptographyDeprecationWarning: Python 2 is no longer supported by the Python
core team. Support for it is now deprecated in cryptography, and will be removed
in a future release.
  CryptographyDeprecationWarning,
ansible 2.9.12
  config file = None
  configured module search path = [u'/root/.ansible/plugins/modules',
u'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python2.7/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 2.7.5 (default, Apr  2 2020, 13:16:51) [GCC 4.8.5 20150623
(Red Hat 4.8.5-39)]

[root@centos7 ~]#ll /opt/etc/ansible/ansible.cfg
-rw-r--r-- 1 wang bin 19980 Aug 11 21:34 /opt/etc/ansible/ansible.cfg
```

3.1.5 确认安装

```
[root@ansible ~]#ansible --version
ansible 2.9.5
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Nov 21 2019, 19:31:34) [GCC 8.3.1 20190507
(Red Hat 8.3.1-4)]
```

3.2 Ansible 相关文件

3.2.1 配置文件

- /etc/ansible/ansible.cfg　主配置文件，配置ansible工作特性,也可以在项目的目录中创建此文件,当前目录下如果也有ansible.cfg,则此文件优先生效,建议每个项目目录下,创建独有的ansible.cfg文件
- /etc/ansible/hosts　主机清单
- /etc/ansible/roles/　存放角色的目录

3.2.2 ansible 主配置文件

Ansible 的配置文件可以放在多个不同地方,优先级从高到低顺序如下

```
ANSIBLE_CONFIG          #环境变量,注意此项用 ansible --version 看不到,但可以生效
./ansible.cfg           #当前目录下的ansible.cfg
~/.ansible.cfg          #当前用户家目录下的.ansible.cfg
/etc/ansible/ansible.cfg  #系统默认配置文件
```

Ansible 的默认配置文件 `/etc/ansible/ansible.cfg`,其中大部分的配置内容无需进行修改

```
[defaults]
#inventory      = /etc/ansible/hosts    #主机列表配置文件
#library  = /usr/share/my_modules/      #库文件存放目录
#remote_tmp  = $HOME/.ansible/tmp       #临时py命令文件存放在远程主机目录
#local_tmp      = $HOME/.ansible/tmp    #本机的临时命令执行目录
#forks          = 5             #默认并发数
#sudo_user      = root          #默认sudo 用户
#ask_sudo_pass = True           #每次执行ansible命令是否询问ssh密码
#ask_pass      = True
#remote_port    = 22
#host_key_checking = False      #检查对应服务器的host_key，建议取消此行注释,实现第一次连
接自动信任目标主机
#log_path=/var/log/ansible.log  #日志文件，建议启用
#module_name = command          #默认模块，可以修改为shell模块

[privilege_escalation]          #普通用户提权配置
#become=True
#become_method=sudo
#become_user=root
#become_ask_pass=False
```

范例：当前目录下的ansible的配置文件优先生效

```
[root@ansible ~]#ansible --version
ansible 2.9.17
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Apr 16 2020, 01:36:27) [GCC 8.3.1 20191121
(Red Hat 8.3.1-5)]

[root@ansible ~]#cp /etc/ansible/ansible.cfg  .

[root@ansible ~]#ansible --version
ansible 2.9.17
  config file = /root/ansible.cfg  #注意配置文件路径
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Apr 16 2020, 01:36:27) [GCC 8.3.1 20191121
(Red Hat 8.3.1-5)]
[root@ansible ~]#
```

3.2.3 inventory 主机清单文件

ansible的主要功用在于批量主机操作，为了便捷地使用其中的部分主机，可以在inventory 主机清单文件中将其分组组织

默认的inventory file为`/etc/ansible/hosts`

inventory file可以有多个，且也可以通过Dynamic Inventory来动态生成

注意: 生产建议在每个项目目录下创建项目独立的hosts文件

官方文档:

```
https://docs.ansible.com/ansible/latest/user_guide/intro_inventory.html
```

主机清单文件格式

inventory文件遵循INI文件风格，中括号中的字符为组名。可以将同一个主机同时归并到多个不同的组中

此外，当如若目标主机使用了非默认的SSH端口，还可以在主机名称之后使用冒号加端口号来标明

如果主机名称遵循相似的命名模式，还可以使用列表的方式标识各主机

Inventory 参数说明

```
ansible_ssh_host #将要连接的远程主机名.与你想要设定的主机的别名不同的话,可通过此变量设置.
ansible_ssh_port #ssh端口号.如果不是默认的端口号,通过此变量设置.这种可以使用 ip:端口
192.168.1.100:2222
ansible_ssh_user #默认的 ssh 用户名
ansible_ssh_pass #ssh 密码(这种方式并不安全,我们强烈建议使用 --ask-pass 或 SSH 密钥)
ansible_sudo_pass #sudo 密码(这种方式并不安全,我们强烈建议使用 --ask-sudo-pass)
ansible_sudo_exe (new in version 1.8) #sudo 命令路径(适用于1.8及以上版本)
ansible_connection #与主机的连接类型.比如:local, ssh 或者 paramiko. Ansible 1.2 以前默
认使用 paramiko.1.2 以后默认使用 'smart','smart' 方式会根据是否支持 ControlPersist, 来
判断'ssh' 方式是否可行.
ansible_ssh_private_key_file #ssh 使用的私钥文件.适用于有多个密钥,而你不想使用 SSH 代理的
情况.
ansible_shell_type #目标系统的shell类型.默认情况下,命令的执行使用 'sh' 语法,可设置为
'csh' 或 'fish'.
ansible_python_interpreter #目标主机的 python 路径.适用于的情况: 系统中有多个 Python,
或者命令路径不是"/usr/bin/python",比如  \*BSD, 或者 /usr/bin/python 不是 2.X 版本的
Python.之所以不使用 "/usr/bin/env" 机制,因为这要求远程用户的路径设置正确,且要求 "python"
可执行程序名不可为 python以外的名字(实际有可能名为python26).与
ansible_python_interpreter 的工作方式相同,可设定如 ruby 或 perl 的路径....
```

范例:

```
ntp.magedu.com
[webservers]
www1.magedu.com:2222
www2.magedu.com

[dbservers]
db1.magedu.com
db2.magedu.com
db3.magedu.com
#或者
db[1:3].magedu.com
```

范例: 组嵌套

```
[webservers]
www[1:100].example.com

[dbservers]
db-[a:f].example.com

[appservers]
10.0.0.[1:100]

#定义testsrvs组中包括两个其它分组,实现组嵌套
[testsrvs:children]
webservers
dbservers
```

范例:

```
[test]
10.0.0.8  ansible_connection=local   #指定本地连接,无需ssh配置

#ansible_connection=ssh 需要StrictHostKeyChecking no
10.0.0.7  ansible_connection=ssh  ansible_ssh_port=2222  ansible_ssh_user=wang
ansible_ssh_password=magedu
10.0.0.6  ansible_connection=ssh  ansible_ssh_user=root
 ansible_ssh_password=123456

#执行ansible命令时显示别名,如web01
[websrvs]
web01 ansible_ssh_host=10.0.0.101
web02 ansible_ssh_host=10.0.0.102
[websrvs:vars]
ansible_ssh_password=magedu

some_host         ansible_ssh_port=2222     ansible_ssh_user=manager
aws_host          ansible_ssh_private_key_file=/home/example/.ssh/aws.pem
freebsd_host      ansible_python_interpreter=/usr/local/bin/python
ruby_module_host  ansible_ruby_interpreter=/usr/bin/ruby.1.9.3
```

3.3 Ansible相关工具

/usr/bin/ansible	主程序，临时命令执行工具
/usr/bin/ansible-doc	查看配置文档，模块功能查看工具,相当于man

- /usr/bin/ansible-playbook　　定制自动化任务，编排剧本工具,相当于脚本
- /usr/bin/ansible-pull　　　　远程执行命令的工具
- /usr/bin/ansible-vault　　　　文件加密工具
- /usr/bin/ansible-console　　基于Console界面与用户交互的执行工具
- /usr/bin/ansible-galaxy　　　下载/上传优秀代码或Roles模块的官网平台

利用ansible实现管理的主要方式：

- Ansible Ad-Hoc 即利用ansible命令，主要用于临时命令使用场景
- Ansible playbook 主要用于长期规划好的，大型项目的场景，需要有前期的规划过程

ansible 使用前准备

ansible 相关工具大多数是通过ssh协议，实现对远程主机的配置管理、应用部署、任务执行等功能

建议：使用此工具前，先配置ansible主控端能基于密钥认证的方式联系各个被管理节点

范例：利用sshpass批量实现基于key验证脚本1

```
[root@centos8 ~]#vim /etc/ssh/ssh_config
#修改下面一行
StrictHostKeyChecking no

[root@centos8 ~]#cat hosts.list
10.0.0.18
10.0.0.28
[root@centos8 ~]#vim  push_ssh_key.sh
#!/bin/bash
rpm -q sshpass &> /dev/null || yum -y install sshpass
[ -f /root/.ssh/id_rsa ] || ssh-keygen -f /root/.ssh/id_rsa  -P ''

export SSHPASS=magedu

while read IP;do
    sshpass -e ssh-copy-id  -o StrictHostKeyChecking=no $IP
done < hosts.list
```

范例：实现基于key验证的脚本2

```
[root@centos8 ~]#cat ssh_key.sh
#!/bin/bash
#
#********************************************************************
#Author:         wangxiaochun
#QQ:             29308620
#Date:           2020-08-11
#FileName：       ssh_key.sh
#URL:            http://www.wangxiaochun.com
#Description：         The test script
#Copyright (C):     2020 All rights reserved
#********************************************************************

IPLIST="
10.0.0.8
10.0.0.18
10.0.0.7
10.0.0.6
10.0.0.200"
```

```
rpm -q sshpass &> /dev/null || yum -y install sshpass
[ -f /root/.ssh/id_rsa ] || ssh-keygen -f /root/.ssh/id_rsa  -P ''

export SSHPASS=123456

for IP in $IPLIST;do
    { sshpass -e ssh-copy-id -o StrictHostKeyChecking=no $IP; } &
done
wait
```

3.3.1 ansible-doc

此工具用来显示模块帮助,相当于man

格式

```
ansible-doc [options] [module...]
-l, --list          #列出可用模块
-s, --snippet       #显示指定模块的playbook片段
```

范例: 查看帮助

```
[root@ansible ~]#ansible-doc --help
usage: ansible-doc [-h] [--version] [-v] [-M MODULE_PATH]
                   [--playbook-dir BASEDIR]
                   [-t
{become,cache,callback,cliconf,connection,httpapi,inventory,lookup,netconf,shell
,module,strategy,vars}]
                   [-j] [-F | -l | -s | --metadata-dump]
                   [plugin [plugin ...]]

plugin documentation tool

positional arguments:
  plugin                Plugin

optional arguments:
  --metadata-dump       **For internal testing only** Dump json metadata for
                        all plugins.
  --playbook-dir BASEDIR
                        Since this tool does not use playbooks, use this as a
                        substitute playbook directory.This sets the relative
                        path for many features including roles/ group_vars/
                        etc.
  --version             show program's version number, config file location,
                        configured module search path, module location,
                        executable location and exit
  -F, --list_files      Show plugin names and their source files without
                        summaries (implies --list)
  -M MODULE_PATH, --module-path MODULE_PATH
                        prepend colon-separated path(s) to module library (def
                        ault=~/.ansible/plugins/modules:/usr/share/ansible/plu
                        gins/modules)
  -h, --help            show this help message and exit
  -j, --json            Change output into json format.
  -l, --list            List available plugins
  -s, --snippet         Show playbook snippet for specified plugin(s)
  -t
{become,cache,callback,cliconf,connection,httpapi,inventory,lookup,netconf,shell
,module,strategy,vars}, --type
{become,cache,callback,cliconf,connection,httpapi,inventory,lookup,netconf,shell
,module,strategy,vars}
                        Choose which plugin type (defaults to "module").
                        Available plugin types are : ('become', 'cache',
                        'callback', 'cliconf', 'connection', 'httpapi',
                        'inventory', 'lookup', 'netconf', 'shell', 'module',
                        'strategy', 'vars')
  -v, --verbose         verbose mode (-vvv for more, -vvvv to enable
                        connection debugging)

See man pages for Ansible CLI options or website for tutorials
https://docs.ansible.com
```

范例:

```
#列出所有模块
ansible-doc -l
#查看指定模块帮助用法
ansible-doc ping
#查看指定模块帮助用法
ansible-doc -s  ping
```

范例:

```
[root@ansible ~]#date
Wed Jun 17 16:08:09 CST 2020
[root@ansible ~]#ansible --version
ansible 2.9.9
  config file = /etc/ansible/ansible.cfg
  configured module search path = ['/root/.ansible/plugins/modules',
'/usr/share/ansible/plugins/modules']
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible
  python version = 3.6.8 (default, Apr 16 2020, 01:36:27) [GCC 8.3.1 20191121
(Red Hat 8.3.1-5)]
[root@ansible ~]#ansible-doc  -l|wc -l
3387
```

范例: 查看指定的插件

```
[root@ansible ~]#ansible-doc -t connection -l
[root@ansible ~]#ansible-doc -t lookup  -l
```

3.3.2 ansible

Ansible Ad-Hoc 的执行方式的主要工具就是 ansible

格式:

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

选项说明:

```
--version           #显示版本
-m module           #指定模块，默认为command
-v                  #详细过程 -vv  -vvv更详细
--list-hosts        #显示主机列表，可简写 --list
-C, --check         #检查，并不执行
-T, --timeout=TIMEOUT #执行命令的超时时间，默认10s

-k, --ask-pass      #提示输入ssh连接密码，默认Key验证
-u, --user=REMOTE_USER #执行远程执行的用户,默认root
-b, --become            #代替旧版的sudo 切换
--become-user=USERNAME  #指定sudo的runas用户，默认为root
-K, --ask-become-pass  #提示输入sudo时的口令
-f FORKS, --forks FORKS #指定并发同时执行ansible任务的主机数
```

范例: 将普通用户提升权限

```
#先在被控制端sudo授权
[root@centos8 ~]#grep wang /etc/sudoers
wang    ALL=(ALL)  NOPASSWD: ALL

#以wang的用户连接用户,并利用sudo代表mage执行whoami命令
[root@ansible ~]#ansible 10.0.0.8 -m shell -a 'whoami' -u wang -k  -b --become-
user=mage
SSH password: #输入远程主机wang用户ssh连接密码
10.0.0.8 | CHANGED | rc=0 >>
mage
```

ansible的Host-pattern

用于匹配被控制的主机的列表

All： 表示所有Inventory中的所有主机

范例

```
ansible all -m ping
```

*:通配符

```
ansible  "*"  -m ping
ansible  192.168.1.* -m ping
ansible  "srvs"  -m ping
ansible "10.0.0.6 10.0.0.7" -m ping
```

或关系

```
ansible "websrvs:appsrvs"  -m ping
ansible "192.168.1.10:192.168.1.20"  -m ping
```

逻辑与

```
#在websrvs组并且在dbsrvs组中的主机
ansible "websrvs:&dbsrvs" -m ping
```

逻辑非

```
#在websrvs组，但不在dbsrvs组中的主机
#注意：此处为单引号
ansible 'websrvs:!dbsrvs' -m ping
```

综合逻辑

```
ansible 'websrvs:dbsrvs:&appsrvs:!ftpsrvs' -m ping
```

正则表达式

```
ansible "websrvs:dbsrvs" -m ping
ansible "~(web|db).*\.magedu\.com" -m ping
```

范例:

```
[root@kube-master1 ~]#ansible 'kube*:etcd:!10.0.0.101' -a reboot && reboot
```

范例:

```
[root@centos8 ~]#ansible all  --list-hosts
  hosts (3):
    10.0.0.6
    10.0.0.7
    10.0.0.8
[root@centos8 ~]#ansible websrvs  --list-hosts
  hosts (3):
    10.0.0.6
    10.0.0.7
    10.0.0.8
[root@centos8 ~]#ansible appsrvs  --list-hosts
  hosts (2):
    10.0.0.7
    10.0.0.8
[root@centos8 ~]#ansible "appsrvs:dbsrvs"  --list-hosts
  hosts (3):
    10.0.0.7
    10.0.0.8
    10.0.0.6
[root@centos8 ~]#ansible "dbsrvs"  --list-hosts
  hosts (2):
    10.0.0.6
    10.0.0.7
[root@centos8 ~]#ansible appsrvs  --list-hosts
```

```
  hosts (2):
    10.0.0.7
    10.0.0.8
[root@centos8 ~]#ansible "appsrvs:dbsrvs"  --list-hosts
  hosts (3):
    10.0.0.7
    10.0.0.8
    10.0.0.6
[root@centos8 ~]#ansible "appsrvs:&dbsrvs"  --list-hosts
  hosts (1):
    10.0.0.7

#引用!号时,不要用双引号,而使用单引号
[root@centos8 ~]#ansible "appsrvs:!dbsrvs"  --list-hosts
-bash: !dbsrvs: event not found

[root@centos8 ~]#ansible 'appsrvs:!dbsrvs' --list-hosts
  hosts (1):
    10.0.0.8
```

范例: 并发执行控制

```
#分别执行下面两条命令观察结果
[root@ansible ~]#ansible websrvs -a 'sleep 5' -f10
[root@ansible ~]#ansible websrvs -a 'sleep 5' -f1
```

ansible命令执行过程

1. 加载自己的配置文件,默认/etc/ansible/ansible.cfg
2. 加载自己对应的模块文件，如：command
3. 通过ansible将模块或命令生成对应的临时py文件，并将该文件传输至远程服务器的对应执行用户$HOME/.ansible/tmp/ansible-tmp-数字/XXX.PY文件
4. 给文件+x执行
5. 执行并返回结果
6. 删除临时py文件，退出

ansible 的执行状态：

```
[root@centos8 ~]#grep -A 14 '\[colors\]' /etc/ansible/ansible.cfg
[colors]
#highlight = white
#verbose = blue
#warn = bright purple
#error = red
#debug = dark gray
#deprecate = purple
#skip = cyan
#unreachable = red
#ok = green
#changed = yellow
#diff_add = green
#diff_remove = red
#diff_lines = cyan
```

- 绿色：执行成功并且不需要做改变的操作
- 黄色：执行成功并且对目标主机做变更

- 红色：执行失败

ansible使用范例

```
#以wang用户执行ping存活检测
ansible all -m ping -u wang  -k
#以wang sudo至root执行ping存活检测
ansible all -m ping -u wang -k -b
#以wang sudo至mage用户执行ping存活检测
ansible all -m ping -u wang -k -b --become-user=mage
#以wang sudo至root用户执行ls
ansible all -m command  -u wang -a 'ls /root' -b --become-user=root -k -K
```

3.3.3 ansible-console

此工具可交互执行命令，支持tab，ansible 2.0+新增

提示符格式：

```
执行用户@当前操作的主机组 (当前组的主机数量)[f:并发数]$
```

常用子命令：

- 设置并发数：forks n 例如：forks 10
- 切换组：cd 主机组 例如：cd web
- 列出当前组主机列表：list
- 列出所有的内置命令：?或help

范例

```
[root@ansible ~]#ansible-console
Welcome to the ansible console.
Type help or ? to list commands.

root@all (3)[f:5]$ ping
10.0.0.7 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
10.0.0.6 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
10.0.0.8 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false,
    "ping": "pong"
}
```

```
root@all (3)[f:5]$ list
10.0.0.8
10.0.0.7
10.0.0.6
root@all (3)[f:5]$ cd websrvs
root@websrvs (2)[f:5]$ list
10.0.0.7
10.0.0.8
root@websrvs (2)[f:5]$ forks 10
root@websrvs (2)[f:10]$ cd appsrvs
root@appsrvs (2)[f:5]$ yum name=httpd state=present
root@appsrvs (2)[f:5]$ service name=httpd state=started
```

3.3.4 ansible-playbook

此工具用于执行编写好的 playbook 任务

范例：

```
ansible-playbook hello.yml
cat  hello.yml
---
#hello world yml file
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: hello world
      command: /usr/bin/wall hello world
```

3.3.5 ansible-vault

此工具可以用于加密解密yml文件

格式：

```
ansible-vault [create|decrypt|edit|encrypt|rekey|view]
```

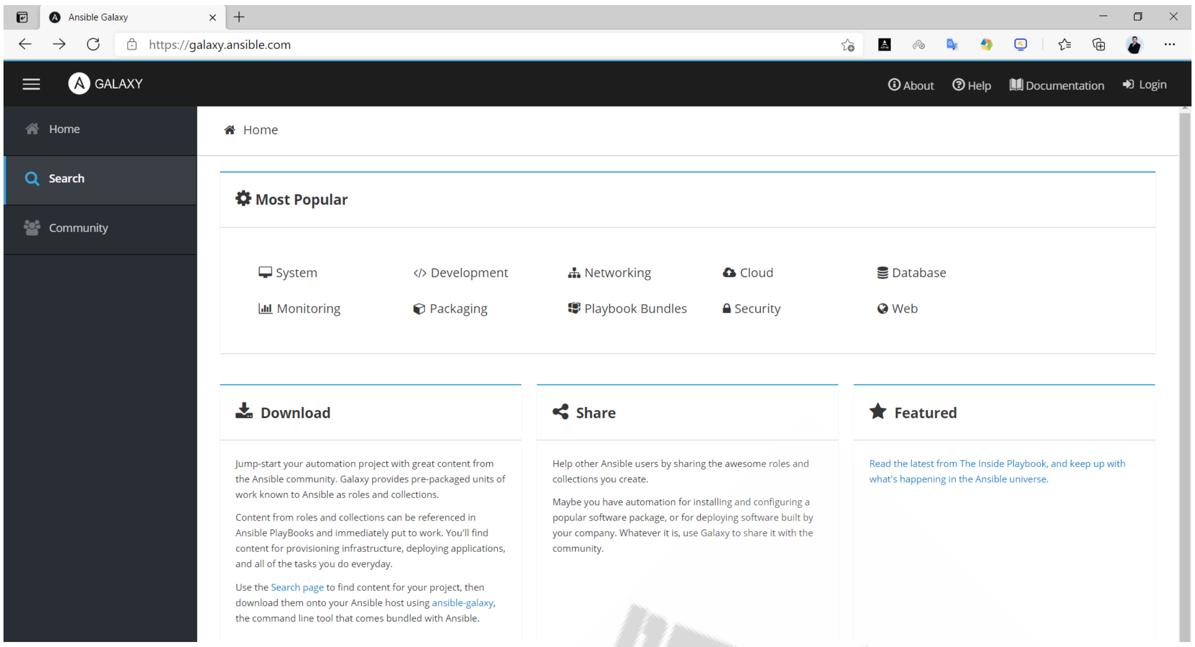
范例

```
ansible-vault encrypt hello.yml     #加密
ansible-vault decrypt hello.yml     #解密
ansible-vault view hello.yml        #查看
ansible-vault edit  hello.yml       #编辑加密文件
ansible-vault rekey  hello.yml      #修改口令
ansible-vault create new.yml        #创建新文件
```

3.3.6 ansible-galaxy

Galaxy 是一个免费网站, 类似于github网站, 网站上发布了很多的共享的roles角色。

Ansible 提供了ansible-galaxy命令行工具连接 https://galaxy.ansible.com 网站下载相应的roles, 进行 init(初始化、search(查拘、install(安装、 remove(移除)等操作。



Download

Jump-start your automation project with great content from the Ansible community. Galaxy provides pre-packaged units of work known to Ansible as roles and collections.

Content from roles and collections can be referenced in Ansible PlayBooks and immediately put to work. You'll find content for provisioning infrastructure, deploying applications, and all of the tasks you do everyday.

Use the Search page to find content for your project, then download them onto your Ansible host using ansible-galaxy, the command line tool that comes bundled with Ansible.

Share

Help other Ansible users by sharing the awesome roles and collections you create.

Maybe you have automation for installing and configuring a popular software package, or for deploying software built by your company. Whatever it is, use Galaxy to share it with the community.

Featured

Read the latest from The Inside Playbook, and keep up with what's happening in the Ansible universe.

范例:

```
#搜索项目
[root@ansible ~]#ansible-galaxy  search lamp
#列出所有已安装的galaxy
ansible-galaxy list
#安装galaxy,默认下载到~/.ansible/roles下
ansible-galaxy install geerlingguy.mysql
ansible-galaxy install geerlingguy.redis
#删除galaxy
ansible-galaxy remove geerlingguy.redis
```

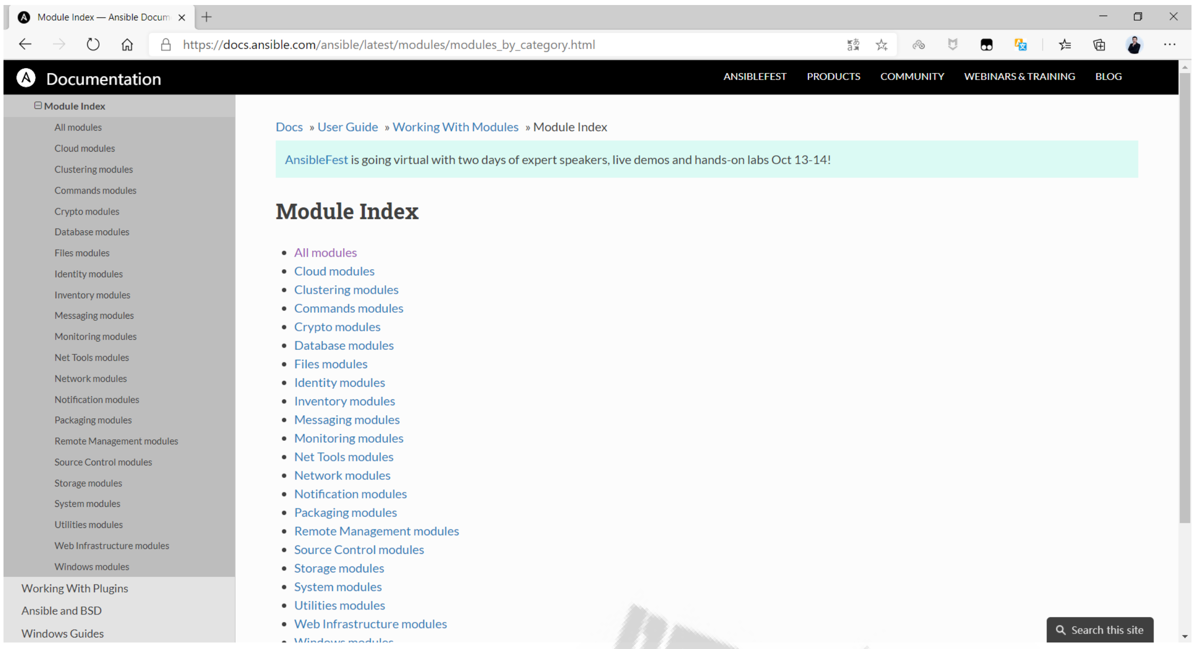
3.4 Ansible常用模块

2015年底270多个模块，2016年达到540个，2018年01月12日有1378个模块，2018年07月15日1852个模块,2019年05月25日（ansible 2.7.10）时2080个模块，2020年03月02日有3387个模块

虽然模块众多，但最常用的模块也就2，30个而已，针对特定业务只用10几个模块

常用模块帮助文档参考：

```
https://docs.ansible.com/ansible/2.9/modules/modules_by_category.html
https://docs.ansible.com/ansible/2.9/modules/list_of_all_modules.html


https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html
https://docs.ansible.com/ansible/latest/modules/modules_by_category.html
```


Module Index — Ansible Docum
https://docs.ansible.com/ansible/latest/modules/modules_by_category.html
Documentation
ANSIBLEFEST
PRODUCTS
COMMUNITY
WEBINARS & TRAINING
BLOG
Search this site

Module Index
- All modules
- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Net Tools modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules
- System modules
- Utilities modules
- Web Infrastructure modules
- Windows modules

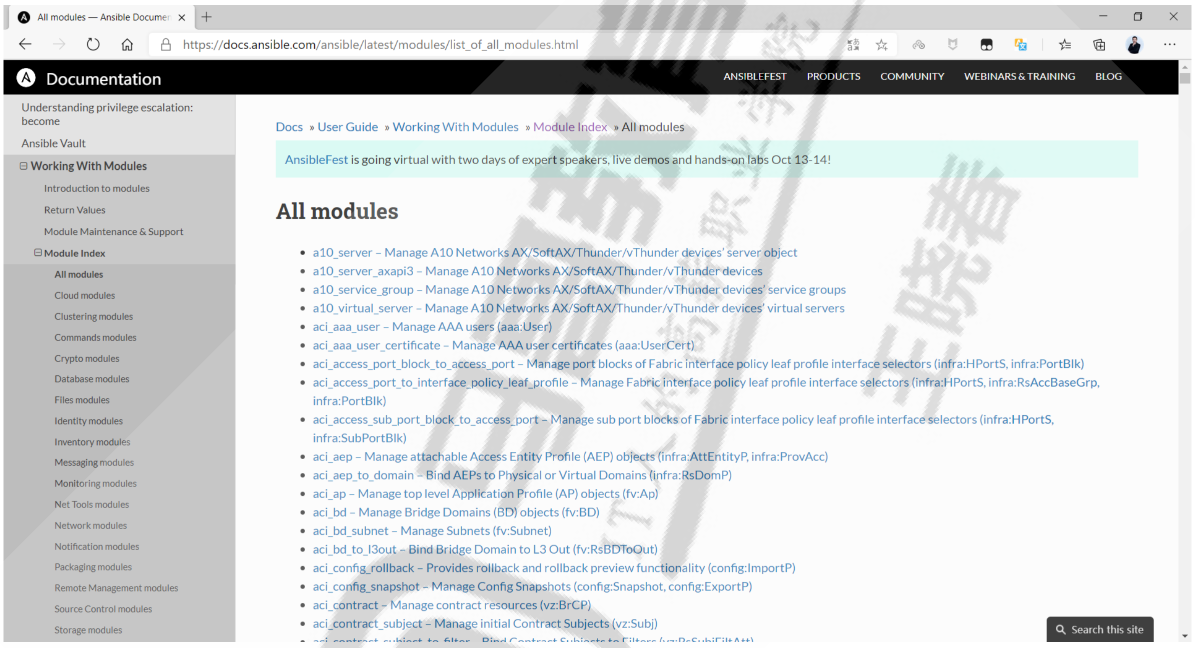
Working With Plugins
Ansible and BSD
Windows Guides

Docs » User Guide » Working With Modules » Module Index

AnsibleFest is going virtual with two days of expert speakers, live demos and hands-on labs Oct 13-14!

Module Index

- All modules
- Cloud modules
- Clustering modules
- Commands modules
- Crypto modules
- Database modules
- Files modules
- Identity modules
- Inventory modules
- Messaging modules
- Monitoring modules
- Net Tools modules
- Network modules
- Notification modules
- Packaging modules
- Remote Management modules
- Source Control modules
- Storage modules
- System modules
- Utilities modules
- Web Infrastructure modules


All modules — Ansible Documen
https://docs.ansible.com/ansible/latest/modules/list_of_all_modules.html
Documentation
ANSIBLEFEST
PRODUCTS
COMMUNITY
WEBINARS & TRAINING
BLOG
Search this site

Understanding privilege escalation: become
Ansible Vault
Working With Modules
Introduction to modules
Return Values
Module Maintenance & Support
Module Index
All modules
Cloud modules
Clustering modules
Commands modules
Crypto modules
Database modules
Files modules
Identity modules
Inventory modules
Messaging modules
Monitoring modules
Net Tools modules
Network modules
Notification modules
Packaging modules
Remote Management modules
Source Control modules
Storage modules

Docs » User Guide » Working With Modules » Module Index » All modules

AnsibleFest is going virtual with two days of expert speakers, live demos and hands-on labs Oct 13-14!

All modules

- a10_server – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' server object
- a10_server_axapi3 – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices
- a10_service_group – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' service groups
- a10_virtual_server – Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' virtual servers
- aci_aaa_user – Manage AAA users (aaa:User)
- aci_aaa_user_certificate – Manage AAA user certificates (aaa:UserCert)
- aci_access_port_block_to_access_port – Manage port blocks of Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:PortBlk)
- aci_access_port_to_interface_policy_leaf_profile – Manage Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:RsAccBaseGrp, infra:PortBlk)
- aci_access_sub_port_block_to_access_port – Manage sub port blocks of Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:SubPortBlk)
- aci_aep – Manage attachable Access Entity Profile (AEP) objects (infra:AttEntityP, infra:ProvAcc)
- aci_aep_to_domain – Bind AEPs to Physical or Virtual Domains (infra:RsDomP)
- aci_ap – Manage top level Application Profile (AP) objects (fv:Ap)
- aci_bd – Manage Bridge Domains (BD) objects (fv:BD)
- aci_bd_subnet – Manage Subnets (fv:Subnet)
- aci_bd_to_l3out – Bind Bridge Domain to L3 Out (fv:RsBDToOut)
- aci_config_rollback – Provides rollback and rollback preview functionality (config:ImportP)
- aci_config_snapshot – Manage Config Snapshots (config:Snapshot, config:ExportP)
- aci_contract – Manage contract resources (vz:BrCP)
- aci_contract_subject – Manage initial Contract Subjects (vz:Subj)

3.4.1 Command 模块

功能：在远程主机执行命令，此为默认模块，可忽略 -m 选项

注意：此命令不支持 $VARNAME < > | ; & 等，可能用shell模块实现

注意：此模块不具有幂等性

范例：

```
[root@ansible ~]#ansible websrvs -m command -a 'chdir=/etc cat centos-release'
10.0.0.7 | CHANGED | rc=0 >>
CentOS Linux release 7.7.1908 (Core)
10.0.0.8 | CHANGED | rc=0 >>
CentOS Linux release 8.1.1911 (Core)
[root@ansible ~]#ansible websrvs -m command -a 'chdir=/etc creates=/data/f1.txt
cat centos-release'
10.0.0.7 | CHANGED | rc=0 >>
CentOS Linux release 7.7.1908 (Core)
10.0.0.8 | SUCCESS | rc=0 >>
skipped, since /data/f1.txt exists
```

```
[root@ansible ~]#ansible websrvs -m command -a 'chdir=/etc removes=/data/f1.txt
cat centos-release'
10.0.0.7 | SUCCESS | rc=0 >>
skipped, since /data/f1.txt does not exist
10.0.0.8 | CHANGED | rc=0 >>
CentOS Linux release 8.1.1911 (Core)


ansible websrvs -m command -a 'service vsftpd start'
ansible websrvs -m command -a 'echo magedu |passwd --stdin wang'
ansible websrvs -m command -a 'rm -rf /data/'
ansible websrvs -m command -a 'echo hello > /data/hello.log'
ansible websrvs -m command -a "echo $HOSTNAME"
```

3.4.2 Shell 模块

功能：和command相似，用shell执行命令,支持各种符号,比如:*,$, >

注意：此模块不具有幂等性

范例：

```
[root@ansible ~]#ansible websrvs -m shell -a "echo $HOSTNAME"
10.0.0.7 | CHANGED | rc=0 >>
ansible
10.0.0.8 | CHANGED | rc=0 >>
ansible
[root@ansible ~]#ansible websrvs -m shell -a 'echo $HOSTNAME'
10.0.0.7 | CHANGED | rc=0 >>
centos7.wangxiaochun.com
10.0.0.8 | CHANGED | rc=0 >>
centos8.localdomain

[root@ansible ~]#ansible websrvs -m shell -a 'echo centos | passwd --stdin wang'
10.0.0.7 | CHANGED | rc=0 >>
Changing password for user wang.
passwd: all authentication tokens updated successfully.
10.0.0.8 | CHANGED | rc=0 >>
Changing password for user wang.
passwd: all authentication tokens updated successfully.
[root@ansible ~]#ansible websrvs -m shell -a 'ls -l /etc/shadow'
10.0.0.7 | CHANGED | rc=0 >>
---------- 1 root root 889 Mar  2 14:34 /etc/shadow
10.0.0.8 | CHANGED | rc=0 >>
---------- 1 root root 944 Mar  2 14:34 /etc/shadow
[root@ansible ~]#ansible websrvs -m shell -a 'echo hello > /data/hello.log'
10.0.0.7 | CHANGED | rc=0 >>

10.0.0.8 | CHANGED | rc=0 >>

[root@ansible ~]#ansible websrvs -m shell -a 'cat  /data/hello.log'
10.0.0.7 | CHANGED | rc=0 >>
hello
10.0.0.8 | CHANGED | rc=0 >>
hello
```

注意：调用bash执行命令 类似 cat /tmp/test.md | awk -F'|' '{print $1,$2}' &> /tmp/example.txt 这些复杂命令，即使使用shell也可能会失败，解决办法：写到脚本时，copy到远程，执行，再把需要的结果拉回执行命令的机器

范例：将shell模块代替command，设为模块

```
[root@ansible ~]#vim /etc/ansible/ansible.cfg
#修改下面一行
module_name = shell
```

3.4.3 Script 模块

功能：在远程主机上运行ansible服务器上的脚本(无需执行权限)

注意：此模块不具有幂等性

范例：

```
ansible websrvs  -m script -a /data/test.sh
```

3.4.4 Copy 模块

功能：从ansible服务器主控端复制文件到远程主机

注意: src=file 如果是没指明路径,则为当前目录或当前目录下的files目录下的file文件

```
#如目标存在，默认覆盖，此处指定先备份
ansible websrvs -m copy -a "src=/root/test1.sh dest=/tmp/test2.sh    owner=wang 
mode=600 backup=yes"
#指定内容，直接生成目标文件
ansible websrvs -m copy -a "content='test line1\ntest line2\n' 
dest=/tmp/test.txt"

#复制/etc目录自身,注意/etc/后面没有/
ansible websrvs -m copy -a "src=/etc dest=/backup"

#复制/etc/下的文件，不包括/etc/目录自身,注意/etc/后面有/
ansible websrvs -m copy -a "src=/etc/ dest=/backup"
```

3.4.5 Get_url 模块

功能: 用于将文件从http、https或ftp下载到被管理机节点上

常用参数如下:

```
url: 下载文件的URL,支持HTTP，HTTPS或FTP协议
dest: 下载到目标路径（绝对路径），如果目标是一个目录，就用服务器上面文件的名称，如果目标设置了名
称就用目标设置的名称
owner：指定属主
group：指定属组
mode：指定权限
force: 如果yes，dest不是目录，将每次下载文件，如果内容改变，替换文件。如果否，则只有在目标不存
在时才会下载该文件
checksum:  对目标文件在下载后计算摘要，以确保其完整性
           示例: checksum="sha256:D98291AC[...]B6DC7B97",
                checksum="sha256:http://example.com/path/sha256sum.txt"
url_username: 用于HTTP基本认证的用户名。 对于允许空密码的站点，此参数可以不使用
`url_password'
url_password: 用于HTTP基本认证的密码。 如果未指定`url_username'参数，则不会使用
`url_password'参数
validate_certs：如果“no”，SSL证书将不会被验证。 适用于自签名证书在私有网站上使用
timeout: URL请求的超时时间,秒为单位
```

范例:

```
[root@ansible ~]#ansible websrvs -m get_url -a
'url=http://nginx.org/download/nginx-1.18.0.tar.gz
dest=/usr/local/src/nginx.tar.gz
checksum="md5:b2d33d24d89b8b1f87ff5d251aa27eb8"'
```

3.4.6 Fetch 模块

功能：从远程主机提取文件至ansible的主控端，copy相反，目前不支持目录

范例：

```
ansible websrvs -m fetch -a 'src=/root/test.sh dest=/data/scripts'
```

范例：

```
[root@ansible ~]#ansible   all -m  fetch -a 'src=/etc/redhat-release
dest=/data/os'
[root@ansible ~]#tree /data/os/
/data/os/
├── 10.0.0.6
|   └── etc
|       └── redhat-release
├── 10.0.0.7
|   └── etc
|       └── redhat-release
└── 10.0.0.8
    └── etc
        └── redhat-release

6 directories, 3 files
```

3.4.7 File 模块

功能：设置文件属性,创建软链接等

范例:

```
#创建空文件
ansible all -m  file  -a 'path=/data/test.txt state=touch'
ansible all -m  file  -a 'path=/data/test.txt state=absent'
ansible all -m file -a "path=/root/test.sh owner=wang mode=755"
#创建目录
ansible all -m file -a "path=/data/mysql state=directory owner=mysql
group=mysql"
#创建软链接
ansible all -m file -a 'src=/data/testfile  path|dest|name=/data/testfile-link
state=link'
#创建目录
ansible all -m file  -a 'path=/data/testdir state=directory'

#递归修改目录属性,但不递归至子目录
ansible all -m file -a "path=/data/mysql state=directory owner=mysql
group=mysql"

#递归修改目录及子目录的属性
ansible all -m file -a "path=/data/mysql state=directory owner=mysql group=mysql
recurse=yes"
```

3.4.8 stat 模块

功能：检查文件或文件系统的状态

注意：对于Windows目标，请改用win_stat模块

选项:

```
path：文件/对象的完整路径（必须）
```

常用的返回值判断:

```
exists： 判断是否存在
isuid： 调用用户的ID与所有者ID是否匹配
```

范例:

```
[root@ansible ~]#ansible 127.0.0.1 -m stat -a 'path=/etc/passwd'
127.0.0.1 | SUCCESS => {
    "changed": false,
    "stat": {
        "atime": 1614601466.7493012,
        "attr_flags": "",
        "attributes": [],
        "block_size": 4096,
        "blocks": 8,
        "charset": "us-ascii",
        "checksum": "8f7a9a996d24de98bf1eab4a047f8e89e9c708cf",
        "ctime": 1614334259.4498665,
        "dev": 2050,
        "device_type": 0,
        "executable": false,
        "exists": true,
        "gid": 0,
        "gr_name": "root",
        "inode": 134691833,
        "isblk": false,
        "ischr": false,
        "isdir": false,
        "isfifo": false,
        "isgid": false,
        "islnk": false,
        "isreg": true,
        "issock": false,
        "isuid": false,
        "mimetype": "text/plain",
        "mode": "0000",
        "mtime": 1614334259.4498665,
        "nlink": 1,
        "path": "/etc/passwd",
        "pw_name": "root",
        "readable": true,
        "rgrp": false,
        "roth": false,
        "rusr": false,
        "size": 1030,
        "uid": 0,
        "version": "671641160",
        "wgrp": false,
        "woth": false,
        "writeable": true,
        "wusr": false,
        "xgrp": false,
        "xoth": false,
        "xusr": false
    }
}
```

案例:

```
- name: install  | Check if  file is already configured.
  stat: path={{ nginx_file_path }}
  connection: local
  register: nginx_file_result
- name: install | Download nginx file
  get_url: url={{ nginx_file_url }} dest={{ software_files_path }}
validate_certs=no
  connection: local
  when:, not. nginx_file_result.stat.exists
```

范例:

```
[root@ansible ansible]#cat stat.yml
---
```

```
- hosts: websrvs

  tasks:
    - name: check file
      stat: path=/data/mysql
      register: st
    - name: debug
      debug:
        msg: "/data/mysql is not exist"
      when: not st.stat.exists
 [root@ansible ansible]#ansible-playbook    stat.yml

PLAY [websrvs]
********************************************************************************
***************************************

TASK [Gathering Facts]
********************************************************************************
*******************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [check file]
********************************************************************************
************************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [debug]
********************************************************************************
*****************************************
ok: [10.0.0.7] => {
    "msg": "/data/mysql is not exist"
}
ok: [10.0.0.8] => {
    "msg": "/data/mysql is not exist"
}

PLAY RECAP
********************************************************************************
*******************************************
10.0.0.7                   : ok=3    changed=0    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
10.0.0.8                   : ok=3    changed=0    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
```

3.4.9 unarchive 模块

功能：解包解压缩

实现有两种用法：

1、将ansible主机上的压缩包传到远程主机后解压缩至特定目录，设置copy=yes,此为默认值,可省略
2、将远程主机上的某个压缩包解压缩到指定路径下，设置copy=no

常见参数：

```
copy：默认为yes，当copy=yes，拷贝的文件是从ansible主机复制到远程主机上，如果设置为copy=no，
会在远程主机上寻找src源文件
remote_src：和copy功能一样且互斥，yes表示在远程主机，不在ansible主机，no表示文件在ansible
主机上
src：源路径，可以是ansible主机上的路径，也可以是远程主机(被管理端或者第三方主机)上的路径，如果
是远程主机上的路径，则需要设置copy=no
dest：远程主机上的目标路径
mode：设置解压缩后的文件权限
```

范例:

```
ansible all -m unarchive -a 'src=/data/foo.tgz dest=/var/lib/foo owner=wang
group=bin'
ansible all -m unarchive -a 'src=/tmp/foo.zip dest=/data copy=no mode=0777'
ansible all -m unarchive -a 'src=https://example.com/example.zip dest=/data
copy=no'

ansible websrvs -m unarchive -a
'src=https://releases.ansible.com/ansible/ansible-2.1.6.0-0.1.rc1.tar.gz
dest=/data/   owner=root remote_src=yes'
ansible websrvs -m unarchive -a 'src=http://nginx.org/download/nginx-
1.18.0.tar.gz dest=/usr/local/src/ copy=no'
```

3.4.10 Archive 模块

功能：打包压缩保存在被管理节点

范例:

```
ansible websrvs -m archive  -a 'path=/var/log/ dest=/data/log.tar.bz2 format=bz2
owner=wang mode=0600'
```

3.4.11 Hostname 模块

功能：管理主机名

范例:

```
ansible node1 -m hostname -a "name=websrv"
ansible 10.0.0.18 -m hostname -a 'name=node18.magedu.com'
```

3.4.12 Cron 模块

功能：计划任务
支持时间：minute，hour，day，month，weekday

范例:

```
#备份数据库脚本
[root@centos8 ~]#cat /root/mysql_backup.sh
#!/bin/bash
mysqldump -A -F --single-transaction --master-data=2 -q -uroot |gzip >
/data/mysql_`date +%F_%T`.sql.gz
```

```
#创建任务
ansible 10.0.0.8 -m cron -a 'hour=2 minute=30 weekday=1-5 name="backup mysql"
job=/root/mysql_backup.sh'
ansible websrvs   -m cron -a "minute=*/5 job='/usr/sbin/ntpdate ntp.aliyun.com
&>/dev/null' name=Synctime"

#禁用计划任务
ansible websrvs   -m cron -a "minute=*/5 job='/usr/sbin/ntpdate 172.20.0.1
&>/dev/null' name=Synctime disabled=yes"

#启用计划任务
ansible websrvs   -m cron -a "minute=*/5 job='/usr/sbin/ntpdate 172.20.0.1
&>/dev/null' name=Synctime disabled=no"

#删除任务
ansible websrvs -m cron -a "name='backup mysql' state=absent"
ansible websrvs -m cron -a 'state=absent name=Synctime'
```

3.4.13 Yum 和 Apt 模块

功能:

yum 管理软件包，只支持RHEL，CentOS，fedora，不支持Ubuntu其它版本

apt 模块管理 Debian 相关版本的软件包

范例:

```
ansible websrvs -m yum -a 'name=httpd state=present'  #安装
ansible websrvs -m yum -a 'name=nginx state=present enablerepo=epel'  #启用epel源
进行安装
ansible websrvs -m yum -a 'name=* state=lastest exclude=kernel*,foo*' #升级除
kernel和foo开头以外的所有包

ansible websrvs -m yum -a 'name=httpd state=absent'   #删除

[root@ansible ~]#ansible websrvs -m yum -a 'name=sl,cowsay'
```

范例:

```
[root@ansible ~]#ansible websrvs -m yum -a
"name=https://mirror.tuna.tsinghua.edu.cn/zabbix/zabbix/5.2/rhel/7/x86_64/zabbix-
agent-5.2.5-1.el7.x86_64.rpm"
```

范例:

```
[root@centos8 ~]#ansible 10.0.0.100 -m apt -a
'name=bb,sl,cowsay,cmatrix,oneko,hollywood,boxes,libaa-bin,x11-apps'
[root@centos8 ~]#ansible websrvs -m apt -a 'name=rsync,psmisc state=absent'
```

范例: 查看包

```
[root@ansible ~]#ansible localhost -m yum -a "list=tree"
localhost | SUCCESS => {
    "ansible_facts": {
        "pkg_mgr": "dnf"
    },
    "changed": false,
    "msg": "",
    "results": [
        {
            "arch": "x86_64",
            "epoch": "0",
            "name": "tree",
            "nevra": "0:tree-1.7.0-15.el8.x86_64",
            "release": "15.el8",
            "repo": "@System",
            "version": "1.7.0",
            "yumstate": "installed"
        },
        {
            "arch": "x86_64",
            "epoch": "0",
            "name": "tree",
            "nevra": "0:tree-1.7.0-15.el8.x86_64",
            "release": "15.el8",
            "repo": "BaseOS",
            "version": "1.7.0",
            "yumstate": "available"
        }
    ]
}
```

3.4.14 yum_repository 模块

```
- name: Add multiple repositories into the same file (1/2)
  yum_repository:
    name: epel
    description: EPEL YUM repo
    file: external_repos
    baseurl: https://download.fedoraproject.org/pub/epel/$releasever/$basearch/
    gpgcheck: no

- name: Add multiple repositories into the same file (2/2)
  yum_repository:
    name: rpmforge
    description: RPMforge YUM repo
    file: external_repos
    baseurl: http://apt.sw.be/redhat/el7/en/$basearch/rpmforge
    mirrorlist: http://mirrorlist.repoforge.org/el7/mirrors-rpmforge
    enabled: no

 - name: Remove repository from a specific repo file
   yum_repository:
     name: epel
     file: external_repos
     state: absent
```

范例：创建和删除仓库

```
[root@ansible ~]#cat  yum_repo.yml
- hosts: websrvs
```

```
  tasks:
    - name: Add multiple repositories into the same file
      yum_repository:
        name: test
        description: EPEL YUM repo
        file: external_repos
        baseurl:
https://download.fedoraproject.org/pub/epel/$releasever/$basearch/
        gpgcheck: no


[root@ansible ~]#ansible-playbook  yum_repo.yml
[root@web1 ~]#cat /etc/yum.repos.d/external_repos.repo
[test]
baseurl = https://download.fedoraproject.org/pub/epel/$releasever/$basearch/
gpgcheck = 0
name = EPEL YUM repo



[root@ansible ~]#cat remove_yum_repo.yml
- hosts: websrvs
  tasks:
    - name: remove repo
      yum_repository:
        name: test
        file: external_repos
        state: absent

[root@ansible ~]#ansible-playbook  remove_yum_repo.yml
```

3.4.15 Service 模块

功能：管理服务

范例：

```
ansible all -m service -a 'name=httpd state=started enabled=yes'
ansible all -m service -a 'name=httpd state=stopped'
ansible all -m service -a 'name=httpd state=reloaded'
ansible all -m shell -a "sed -i 's/^Listen 80/Listen 8080/'
/etc/httpd/conf/httpd.conf"
ansible all -m service -a 'name=httpd state=restarted'
```

3.4.16 User 模块

功能：管理用户

范例：

```
#创建用户
ansible all -m user -a 'name=user1 comment="test user" uid=2048 home=/app/user1
group=root'
```

```
ansible all -m user -a 'name=nginx comment=nginx uid=88 group=nginx
groups="root,daemon" shell=/sbin/nologin system=yes create_home=no
home=/data/nginx non_unique=yes'

#remove=yes表示删除用户及家目录等数据,默认remove=no
ansible all -m user -a 'name=nginx state=absent remove=yes'

#生成123456加密的密码
ansible localhost -m debug -a "msg={{ '123456'|
password_hash('sha512','salt')}}"
localhost | SUCCESS => {
    "msg": "$6$salt$MktMKPZJ6t59GfxcJU20DwcwQzfMvOlHFVZiOVD71w."
}
#用上面创建的密码创建用户
ansible websrvs -m user -a 'name=test
password="$6$salt$MktMKPZJ6t59GfxcJU20DwcwQzfMvOlHFVZiOVD71w."'


#创建用户test,并生成4096bit的私钥
ansible websrvs -m user -a 'name=test generate_ssh_key=yes ssh_key_bits=4096
ssh_key_file=.ssh/id_rsa'
```

3.4.17 Group 模块

功能：管理组

范例：

```
#创建组
ansible websrvs -m group  -a 'name=nginx gid=88 system=yes'
#删除组
ansible websrvs -m group  -a 'name=nginx state=absent'
```

3.4.18 Lineinfile 模块

ansible在使用sed进行替换时，经常会遇到需要转义的问题，而且ansible在遇到特殊符号进行替换时，存在问题，无法正常进行替换 。其实在ansible自身提供了两个模块：lineinfile模块和replace模块，可以方便的进行替换

一般在ansible当中去修改某个文件的单行进行替换的时候需要使用lineinfile模块

regexp参数 ：使用正则表达式匹配对应的行，当替换文本时，如果有多行文本都能被匹配，则只有最后面被匹配到的那行文本才会被替换，当删除文本时，如果有多行文本都能被匹配，这么这些行都会被删除。

如果想进行多行匹配进行替换需要使用replace模块

功能：相当于sed，可以修改文件内容

范例：

```
ansible websrvs -m  lineinfile -a "path=/etc/httpd/conf/httpd.conf
regexp='^Listen' line='Listen 80'"

ansible all -m   lineinfile -a "path=/etc/selinux/config regexp='^SELINUX='
line='SELINUX=disabled'"

ansible all -m lineinfile  -a 'dest=/etc/fstab state=absent regexp="^#"'
```

3.4.19 Replace 模块

该模块有点类似于sed命令，主要也是基于正则进行匹配和替换，建议使用

范例:

```
ansible all -m replace -a "path=/etc/fstab regexp='^(UUID.*)' replace='#\1'"
ansible all -m replace -a "path=/etc/fstab regexp='^#(UUID.*)' replace='\1'"
```

3.4.20 SELinux 模块

该模块管理 SELInux 策略

范例:

```
[root@ansible ~]#ansible 10.0.0.8 -m selinux -a 'state=disabled'
[WARNING]: SELinux state temporarily changed from 'enforcing' to 'permissive'.
State change will take effect next reboot.
10.0.0.8 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "configfile": "/etc/selinux/config",
    "msg": "Config SELinux state changed from 'enforcing' to 'disabled'",
    "policy": "targeted",
    "reboot_required": true,
    "state": "disabled"
}

[root@centos8 ~]#grep -v '#' /etc/selinux/config
SELINUX=disabled
SELINUXTYPE=targeted

[root@centos8 ~]#getenforce
Permissive
```

3.4.21 reboot 模块

```
[root@ansible ~]#ansible websrvs -m reboot
```

3.4.22 mount 挂载和卸载

功能: 挂载和卸载文件系统

范例:

```
#临时挂载
mount websrvs -m mount -a 'src="UUID=b3e48f45-f933-4c8e-a700-22a159ec9077"
path=/home fstype=xfs opts=noatime state=present'

#临时取消挂载
mount websrvs -m mount -a 'path=/home fstype=xfs opts=noatime state=unmounted'


#永久挂载
ansible websrvs -m mount -a 'src=10.0.0.8:/data/wordpress path=/var/www/html/wp-
content/uploads opts="_netdev" state=mounted'

#永久卸载
ansible websrvs -m mount -a 'src=10.0.0.8:/data/wordpress path=/var/www/html/wp-
content/uploads  state=absent'
```

3.4.23 Setup 模块

功能： setup 模块来收集主机的系统信息，这些 facts 信息可以直接以变量的形式使用，但是如果主机较多，会影响执行速度

可以使用`gather_facts: no`来禁止 Ansible 收集 facts 信息

范例:

```
ansible all -m setup
ansible all -m setup -a "filter=ansible_nodename"
ansible all -m setup -a "filter=ansible_hostname"
ansible all -m setup -a "filter=ansible_domain"
ansible all -m setup -a "filter=ansible_memtotal_mb"
ansible all -m setup -a "filter=ansible_memory_mb"
ansible all -m setup -a "filter=ansible_memfree_mb"
ansible all -m setup -a "filter=ansible_os_family"
ansible all -m setup -a "filter=ansible_distribution_major_version"
ansible all -m setup -a "filter=ansible_distribution_version"
ansible all -m setup -a "filter=ansible_processor_vcpus"
ansible all -m setup -a "filter=ansible_all_ipv4_addresses"
ansible all -m setup -a "filter=ansible_architecture"
ansible all -m setup -a "filter=ansible_uptime_seconds"
ansible all -m setup -a "filter=ansible_processor*"
ansible all -m setup -a 'filter=ansible_env'
```

范例:

```
[root@ansible ~]#ansible all  -m  setup -a 'filter=ansible_python_version'
10.0.0.7 | SUCCESS => {
    "ansible_facts": {
        "ansible_python_version": "2.7.5",
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}
10.0.0.6 | SUCCESS => {
    "ansible_facts": {
        "ansible_python_version": "2.6.6",
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false
}
10.0.0.8 | SUCCESS => {
    "ansible_facts": {
        "ansible_python_version": "3.6.8",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
[root@ansible ~]#
```

范例：取IP地址

```
#取所有IP
ansible 10.0.0.101 -m setup -a 'filter=ansible_all_ipv4_addresses'
10.0.0.101 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "192.168.0.1",
            "192.168.0.2",
            "192.168.64.238",
            "192.168.13.36",
            "10.0.0.101",
            "172.16.1.0",
            "172.17.0.1"
        ]
    },
    "changed": false
}

#取默认IP
ansible all -m setup -a 'filter="ansible_default_ipv4"'
10.0.0.101 | SUCCESS => {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "10.0.0.101",
            "alias": "eth0",
            "broadcast": "10.0.0.255",
            "gateway": "10.0.0.2",
            "interface": "eth0",
            "macaddress": "00:0c:29:e8:c7:9b",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "10.0.0.0",
            "type": "ether"
        }
    },
    "changed": false
}
```

3.4.24 debug 模块

此模块可以用于输出信息,并且通过 msg 定制输出的信息内容

注意: msg后面的变量有时需要加 " " 引起来

范例: debug 模块默认输出Hello world

```
[root@ansible ~]#ansible 10.0.0.18 -m debug
10.0.0.18 | SUCCESS => {
    "msg": "Hello world!"
}


[root@ansible ansible]#cat debug.yml
---
- hosts: websrvs

  tasks:
    - name: output Hello world
      debug:

#默认没有指定msg,默认输出"Hello world!"
[root@ansible ansible]#ansible-playbook  debug.yml

PLAY [websrvs]
********************************************************************************
***************************************

TASK [Gathering Facts]
********************************************************************************
*******************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [output variables]
********************************************************************************
******************************
ok: [10.0.0.7] => {
    "msg": "Hello world!"
}
ok: [10.0.0.8] => {
    "msg": "Hello world!"
}

PLAY RECAP
********************************************************************************
*******************************************
10.0.0.7                   : ok=2    changed=0    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
10.0.0.8                   : ok=2    changed=0    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
```

范例: 利用debug 模块输出变量

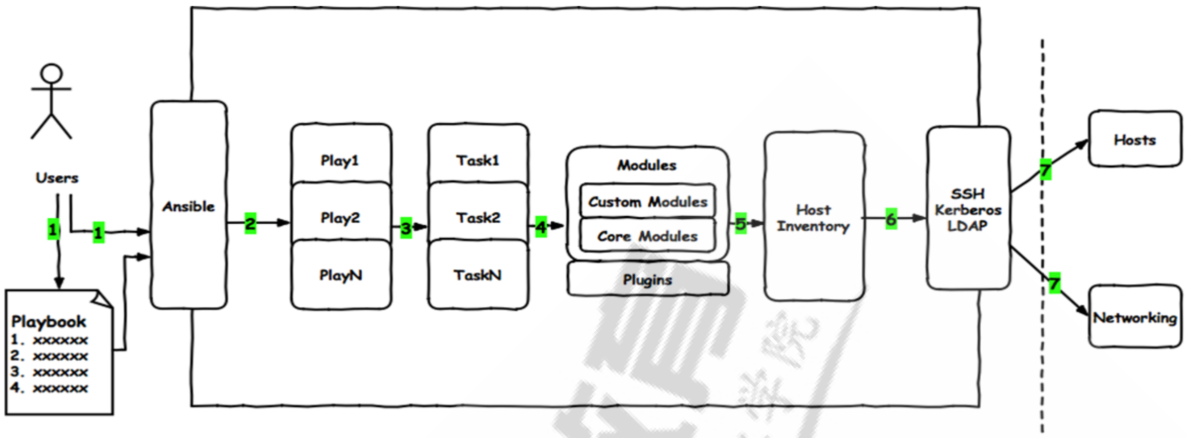
```
[root@centos8 ~]#cat debug.yaml
---
- hosts: websrvs

  tasks:
    - name: output variables
      debug:
        msg: Host "{{ ansible_nodename }}" Ip  "{{ ansible_default_ipv4.address
}}"
[root@centos8 ~]#ansible-playbook  debug.yaml

PLAY [websrvs]
********************************************************************************
***************************************

TASK [Gathering Facts]
********************************************************************************
*******************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [output variables]
********************************************************************************
******************************
ok: [10.0.0.7] => {
    "msg": "Host \"centos7.wangxiaochun.com\" Ip  \"10.0.0.7\""
}
ok: [10.0.0.8] => {
    "msg": "Host \"centos8.wangxiaochun.com\" Ip  \"10.0.0.8\""
}

PLAY RECAP
********************************************************************************
*******************************************
10.0.0.7                   : ok=2    changed=0    unreachable=0    failed=0    
 skipped=0    rescued=0    ignored=0
10.0.0.8                   : ok=2    changed=0    unreachable=0    failed=0    
 skipped=0    rescued=0    ignored=0
```

范例: 显示字符串特定字符

```
# cat debug.yml
- hosts: all
  gather_facts: no
  vars:
    a: "12345"
  tasks:
  - debug:
      msg: "{{a[2]}}"

#定义了一个字符串变量a，如果想要获取a字符串的第3个字符，则可以使用"a[2]"获取，索引从0开始，执行上例playbook，debug的输出信息如下：

TASK [debug] *************************
ok: [test71] => {
    "msg": "3"
}
```

4 Playbook

4.1 playbook介绍



- playbook 剧本是由一个或多个"play"组成的列表
- play的主要功能在于将预定义的一组主机，装扮成事先通过ansible中的task定义好的角色。Task实际是调用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，按事先编排的机制执行预定义的动作
- Playbook 文件是采用YAML语言编写的

4.2 YAML 语言

4.2.1 YAML 语言介绍

YAML：YAML Ain't Markup Language，即YAML不是标记语言。不过，在开发的这种语言时，YAML的意思其实是："Yet Another Markup Language"（仍是一种标记语言）

YAML是一个可读性高的用来表达资料序列的格式。YAML参考了其他多种语言，包括：XML、C语言、Python、Perl以及电子邮件格式RFC2822等。Clark Evans在2001年在首次发表了这种语言，另外Ingy döt Net与Oren Ben-Kiki也是这语言的共同设计者,目前很多最新的软件比较流行采用此格式的文件存放配置信息，如:ubuntu，anisble，docker，kubernetes等

YAML 官方网站：http://www.yaml.org

ansible 官网: https://docs.ansible.com/ansible/latest/reference_appendices/YAMLSyntax.html

4.2.2 YAML 语言特性

- YAML的可读性好
- YAML和脚本语言的交互性好
- YAML使用实现语言的数据类型
- YAML有一个一致的信息模型
- YAML易于实现
- YAML可以基于流来处理
- YAML表达能力强，扩展性好

4.2.3 YAML语法简介

- 在单一文件第一行，用连续三个连字号"-" 开始，还有选择性的连续三个点号(...)用来表示文件的结尾
- 次行开始正常写Playbook的内容，一般建议写明该Playbook的功能
- 使用#号注释代码
- 缩进必须是统一的，不能空格和tab混用
- 缩进的级别也必须是一致的，同样的缩进代表同样的级别，程序判别配置的级别是通过缩进结合换行来实现的
- YAML文件内容是区别大小写的，key/value的值均需大小写敏感
- 多个key/value可同行写也可换行写，同行使用，分隔
- key后面冒号要加一个空格 比如: key: value
- value可是个字符串，也可是另一个列表
- YAML文件扩展名通常为yml或yaml

4.2.4 支持的数据类型

YAML 支持以下常用几种数据类型：

- 标量：单个的、不可再分的值
- 对象：键值对的集合，又称为映射（mapping）/ 哈希（hashes） / 字典（dictionary）
- 数组：一组按次序排列的值，又称为序列（sequence） / 列表（list）

4.2.4.1 scalar 标量

key对应value

```
name: wang
age: 18
```

使用缩进的方式

```
name:
  wang
age:
  18
```

标量是最基本的，不可再分的值，包括：

- 字符串
- 布尔值
- 整数
- 浮点数
- Null
- 时间
- 日期

4.2.4.2 Dictionary 字典

字典由多个key与value构成，key和value之间用 ： 分隔, 并且 : 后面有一个空格，所有k/v可以放在一行，或者每个 k/v 分别放在不同行

格式

```
account: { name: wang, age: 30 }
```

使用缩进方式

```
account:
  name: wang
  age: 18
```

范例:

```
#不同行
# An employee record
name: Example Developer
job: Developer
skill: Elite(社会精英)


#同一行,也可以将key:value放置于{}中进行表示，用,分隔多个key:value
# An employee record
{name: "Example Developer", job: "Developer", skill: "Elite"}
```

4.2.4.2 List 列表

列表由多个元素组成，每个元素放在不同行，且元素前均使用"-"打头，并且 - 后有一个空格, 或者将所有元素用 [ ] 括起来放在同一行

格式

```
course: [ linux , golang , python ]
```

也可以写成以 - 开头的多行

```
course:
  - linux
  - golang
  - python
```

数据里面也可以包含字典

```
course:
  - linux: manjaro
  - golang: gin
  - python: django
```

范例:

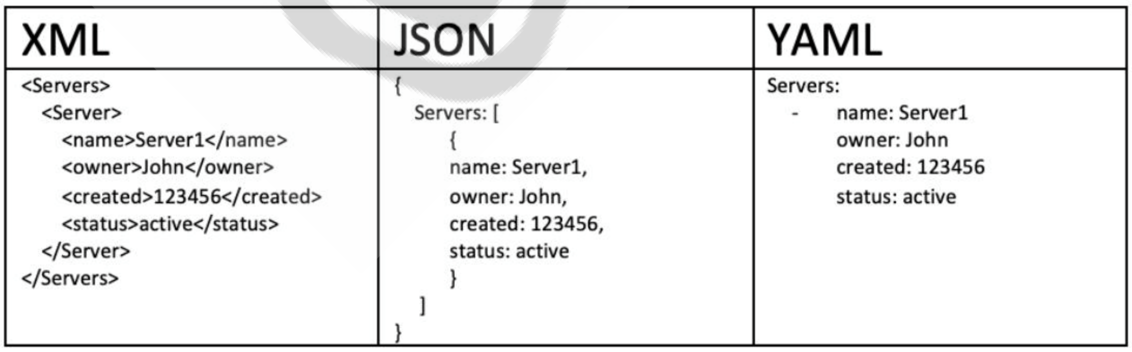
```
#不同行,行以-开头,后面有一个空格
# A list of tasty fruits
- Apple
- Orange
- Strawberry
- Mango

#同一行
[Apple,Orange,Strawberry,Mango]
```

范例：YAML 表示一个家庭

```
name: John Smith
age: 41
gender: Male
spouse: { name: Jane Smith, age: 37, gender: Female } # 写在一行里
  name: Jane Smith    #也可以写成多行
  age: 37
  gender: Female
children: [ {name: Jimmy Smith,age: 17, gender: Male}, {name: Jenny Smith, age:
13, gender: Female}, {name: hao Smith, age: 20, gender: Male } ]  #写在一行
  - name: Jimmy Smith     #写在多行,更为推荐的写法
    age: 17
    gender: Male
  - {name: Jenny Smith, age: 13, gender: Female}
  - {name: hao Smith, age: 20, gender: Male }
```

4.2.5 三种常见的数据格式

- XML：Extensible Markup Language，可扩展标记语言，可用于数据交换和配置
- JSON：JavaScript Object Notation, JavaScript 对象表记法，主要用来数据交换或配置，不支持注释
- YAML：YAML Ain't Markup Language YAML 不是一种标记语言， 主要用来配置，大小写敏感，不支持tab



XML

```
<Servers>
  <Server>
    <name>Server1</name>
    <owner>John</owner>
    <created>123456</created>
    <status>active</status>
  </Server>
</Servers>
```

JSON

```
{
  Servers: [
    {
    name: Server1,
    owner: John,
    created: 123456,
    status: active
    }
  ]
}
```

YAML

```
Servers:
  -    name: Server1
       owner: John
       created: 123456
       status: active
```

可以用工具互相转换，参考网站：

https://www.json2yaml.com/

http://www.bejson.com/json/json2yaml/

4.3 Playbook 核心组件

官方文档

```
https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.
html#playbook-keywords
```

一个playbook 中由多个组件组成,其中所用到的常见组件类型如下:

- Hosts 执行的远程主机列表
- Tasks 任务集,由多个task的元素组成的列表实现,每个task是一个字典,一个完整的代码块功能需最少元素需包括 name 和 task,一个name只能包括一个task
- Variables 内置变量或自定义变量在playbook中调用
- Templates 模板，可替换模板文件中的变量并实现一些简单逻辑的文件
- Handlers 和 notify 结合使用，由特定条件触发的操作，满足条件方才执行，否则不执行
- tags 标签 指定某条任务执行，用于选择运行playbook中的部分代码。ansible具有幂等性，因此会自动跳过没有变化的部分，即便如此，有些代码为测试其确实没有发生变化的时间依然会非常地长。此时，如果确信其没有变化，就可以通过tags跳过此些代码片断

4.3.1 hosts 组件

Hosts：playbook中的每一个play的目的都是为了让特定主机以某个指定的用户身份执行任务。hosts用于指定要执行指定任务的主机，须事先定义在主机清单中

```
one.example.com
one.example.com:two.example.com
192.168.1.50
192.168.1.*
Websrvs:dbsrvs      #或者，两个组的并集
Websrvs:&dbsrvs     #与，两个组的交集
webservers:!dbsrvs  #在websrvs组，但不在dbsrvs组
```

案例:

```
- hosts: websrvs:appsrvs
```

4.3.2 remote_user 组件

remote_user: 可用于Host和task中。也可以通过指定其通过sudo的方式在远程主机上执行任务，其可用于play全局或某任务；此外，甚至可以在sudo时使用sudo_user指定sudo时切换的用户

```
- hosts: websrvs
  remote_user: root

  tasks:
    - name: test connection
      ping:
      remote_user: magedu
      sudo: yes                #默认sudo为root
      sudo_user:wang           #sudo为wang
```

4.3.3 task列表和action组件

play的主体部分是task list，task list中有一个或多个task,各个task 按次序逐个在hosts中指定的所有主机上执行，即在所有主机上完成第一个task后，再开始第二个task
task的目的是使用指定的参数执行模块，而在模块参数中可以使用变量。模块执行是幂等的，这意味着多次执行是安全的，因为其结果均一致
每个task都应该有其name，用于playbook的执行结果输出，建议其内容能清晰地描述任务执行步骤。如果未提供name，则action的结果将用于输出

task两种格式：

```
action: module arguments  #示例: action: shell wall hello
module: arguments      #建议使用  #示例: shell: wall hello
```

注意：shell和command模块后面跟命令，而非key=value

范例:

```
[root@ansible ansible]#cat hello.yaml
---
# first yaml file
- hosts: websrvs
  remote_user: root
  gather_facts: no   #不收集系统信息,提高执行效率

  tasks:
    - name: test network connection
      ping:
    - name: excute command
      command: wall "hello world!"
```

范例：

```
---
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: install httpd
      yum: name=httpd
    - name: start httpd
      service: name=httpd state=started enabled=yes
```

4.3.4 其它组件说明

某任务的状态在运行后为changed时，可通过"notify"通知给相应的handlers任务

还可以通过"tags"给task 打标签，可在ansible-playbook命令上使用-t指定进行调用

4.3.5 ShellScripts VS Playbook 案例

```
#SHELL脚本实现
#!/bin/bash
# 安装Apache
yum install --quiet -y httpd
# 复制配置文件
cp /tmp/httpd.conf /etc/httpd/conf/httpd.conf
cp/tmp/vhosts.conf /etc/httpd/conf.d/
# 启动Apache，并设置开机启动
systemctl enable --now httpd

#Playbook实现
---
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: "安装Apache"
      yum: name=httpd
    - name: "复制配置文件"
      copy: src=/tmp/httpd.conf dest=/etc/httpd/conf/
    - name: "复制配置文件"
      copy: src=/tmp/vhosts.conf dest=/etc/httpd/conf.d/
    - name: "启动Apache，并设置开机启动"
      service: name=httpd state=started enabled=yes
```

4.4 playbook 命令

格式

```
ansible-playbook <filename.yml> ... [options]
```

常见选项

```
--syntax-check      #语法检查,可缩写成--syntax, 相当于bash -n
-C --check          #模拟执行,只检测可能会发生的改变，但不真正执行操作,dry run
--list-hosts        #列出运行任务的主机
--list-tags         #列出tag
--list-tasks        #列出task
--limit 主机列表      #只针对主机列表中的特定主机执行
-i INVENTORY        #指定主机清单文件,通常一个项对应一个主机清单文件
--start-at-task START_AT_TASK #从指定task开始执行,而非从头开始,START_AT_TASK为任务的name
-v -vv  -vvv        #显示过程
```

范例:

```
[root@ansible ansible]#cat hello.yml
---
- hosts: websrvs
  tasks:
    - name: hello
      command: echo "hello ansible"
[root@ansible ansible]#ansible-playbook  hello.yml
[root@ansible ansible]#ansible-playbook -v  hello.yml
```

范例

```
ansible-playbook  file.yml  --check #只检测
ansible-playbook  file.yml
ansible-playbook  file.yml  --limit websrvs
```

4.5 Playbook 初步

4.5.1 利用 playbook 创建 mysql 用户

范例：mysql_user.yml

```
---
- hosts: dbsrvs
  remote_user: root
  gather_facts: no

  tasks:
    - {name: create group, group: name=mysql system=yes gid=306}
    - name: create user
      user: name=mysql shell=/sbin/nologin system=yes group=mysql uid=306
home=/data/mysql create_home=no
```

4.5.2 利用 playbook 安装 nginx

范例：install_nginx.yml

```
---
# install nginx
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: add group nginx
      group: name=nginx state=present
    - name: add user nginx
      user: name=nginx state=present group=nginx
    - name: Install Nginx
      yum: name=nginx state=present
    - name: web page
      copy: src=files/index.html dest=/usr/share/nginx/html/index.html
    - name: Start Nginx
      service: name=nginx state=started enabled=yes
```

4.5.3 利用 playbook 安装和卸载 httpd

范例：install_httpd.yml

```
[root@centos8 ansible]#cat install_httpd.yml
---
#install httpd
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: Install httpd
      yum: name=httpd
    - name: Modify config list port
      lineinfile:
        path: /etc/httpd/conf/httpd.conf
        regexp: '^Listen'
        line: 'Listen 8080'
    - name: Modify config data1
      lineinfile:
        path: /etc/httpd/conf/httpd.conf
        regexp: '^DocumentRoot "/var/www/html"'
        line: 'DocumentRoot "/data/html"'
    - name: Modify config data2
      lineinfile:
        path: /etc/httpd/conf/httpd.conf
        regexp: '^<Directory "/var/www/html">'
        line: '<Directory "/data/html">'
    - name: Mkdir website dir
      file: path=/data/html state=directory
    - name: Web html
      copy: src=files/index.html dest=/data/html/
    - name: Start service
      service: name=httpd state=started enabled=yes


ansible-playbook   install_httpd.yml --limit 10.0.0.8
```

范例：remove_httpd.yml

```
#remove_httpd.yml
---
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: remove httpd package
      yum: name=httpd state=absent
    - name: remove apache user
      user: name=apache state=absent
    - name: remove config file
      file: name=/etc/httpd  state=absent
    - name: remove web html
      file: name=/data/html/ state=absent
```

4.5.4 利用 playbook 安装 MySQL 5.6

范例：安装mysql-5.6.46-linux-glibc2.12

```
[root@ansible ~]#ls -l /data/ansible/files/mysql-5.6.46-linux-glibc2.12-
x86_64.tar.gz
-rw-r--r-- 1 root root 403177622 Dec  4 13:05 /data/ansible/files/mysql-5.6.46-
linux-glibc2.12-x86_64.tar.gz

[root@ansible ~]#cat /data/ansible/files/my.cnf
[mysqld]
socket=/tmp/mysql.sock
user=mysql
symbolic-links=0
datadir=/data/mysql
innodb_file_per_table=1
log-bin
pid-file=/data/mysql/mysqld.pid

[client]
port=3306
socket=/tmp/mysql.sock

[mysqld_safe]
log-error=/var/log/mysqld.log

[root@ansible ~]#cat /data/ansible/files/secure_mysql.sh
#!/bin/bash
/usr/local/mysql/bin/mysql_secure_installation <<EOF

y
magedu
magedu
y
y
y
y
EOF


[root@ansible ~]#tree /data/ansible/files/
/data/ansible/files/
├── my.cnf
├── mysql-5.6.46-linux-glibc2.12-x86_64.tar.gz
└── secure_mysql.sh

0 directories, 3 files


[root@ansible ~]#cat /data/ansible/install_mysql.yml
---
# install mysql-5.6.46-linux-glibc2.12-x86_64.tar.gz
- hosts: dbsrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: install packages
      yum: name=libaio,perl-Data-Dumper,perl-Getopt-Long
    - name: create mysql group
      group: name=mysql gid=306
    - name: create mysql user
      user: name=mysql uid=306 group=mysql shell=/sbin/nologin system=yes
create_home=no home=/data/mysql
    - name: copy tar to remote host and file mode
      unarchive: src=/data/ansible/files/mysql-5.6.46-linux-glibc2.12-
x86_64.tar.gz dest=/usr/local/ owner=root group=root
    - name: create linkfile  /usr/local/mysql
      file: src=/usr/local/mysql-5.6.46-linux-glibc2.12-x86_64
dest=/usr/local/mysql state=link
    - name: data dir
      shell: chdir=/usr/local/mysql/  ./scripts/mysql_install_db --
datadir=/data/mysql --user=mysql
      tags: data
    - name: config my.cnf
      copy: src=/data/ansible/files/my.cnf  dest=/etc/my.cnf
    - name: service script
      shell: /bin/cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
    - name: enable service
      shell: /etc/init.d/mysqld start;chkconfig --add mysqld;chkconfig mysqld on

      tags: service
    - name: PATH variable
      copy: content='PATH=/usr/local/mysql/bin:$PATH'
dest=/etc/profile.d/mysql.sh
    - name: secure script
      script: /data/ansible/files/secure_mysql.sh
      tags: script
```

范例：install_mariadb.yml

```
---
#Installing MariaDB Binary Tarballs
- hosts: dbsrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: create group
      group: name=mysql gid=27 system=yes
    - name: create user
      user: name=mysql uid=27 system=yes group=mysql shell=/sbin/nologin
home=/data/mysql create_home=no
    - name: mkdir datadir
      file: path=/data/mysql owner=mysql group=mysql state=directory
    - name: unarchive package
      unarchive: src=/data/ansible/files/mariadb-10.2.27-linux-x86_64.tar.gz
dest=/usr/local/ owner=root group=root
    - name: link
      file: src=/usr/local/mariadb-10.2.27-linux-x86_64 path=/usr/local/mysql
state=link
    - name: install database
```

```
      shell: chdir=/usr/local/mysql   ./scripts/mysql_install_db --
datadir=/data/mysql --user=mysql
    - name: config file
      copy: src=/data/ansible/files/my.cnf  dest=/etc/ backup=yes
    - name: service script
      shell: /bin/cp  /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
    - name: start service
      service: name=mysqld state=started enabled=yes
    - name: PATH variable
      copy: content='PATH=/usr/local/mysql/bin:$PATH'
dest=/etc/profile.d/mysql.sh
```

4.6 忽略错误 ignore_errors

如果一个task出错,默认将不会继续执行后续的其它task

利用 ignore_errors: yes 可以忽略此task的错误,继续向下执行playbook其它task

```
[root@ansible ansible]#cat test_ignore.yml
---
- hosts: websrvs

  tasks:
    - name: error
      command: /bin/false
      ignore_errors: yes
    - name: continue
      command: wall continue
```

4.7 Playbook中使用handlers和notify

Handlers本质是task list ，类似于MySQL中的触发器触发的行为，其中的task与前述的task并没有本质上的不同，主要用于当关注的资源发生变化时，才会采取一定的操作。而Notify对应的action可用于在每个play的最后被触发，这样可避免多次有改变发生时每次都执行指定的操作，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

注意:

- 如果多个task通知了相同的handlers， 此handlers仅会在所有tasks结束后运行一 次。
- 只有notify对应的task发生改变了才会通知handlers， 没有改变则不会触发handlers
- handlers 是在所有前面的tasks都成功执行才会执行,如果前面任何一个task失败,会导致handler跳过执行,可以使用force_handlers: yes 强制执行handler

案例:

```
---
- hosts: websrvs
  remote_user: root
  gather_facts: no
  tasks:
    - name: Install httpd
      yum: name=httpd state=present
    - name: Install configure file
      copy: src=files/httpd.conf dest=/etc/httpd/conf/
      notify:
        - restart httpd
        - wall
    - name: ensure apache is running
      service: name=httpd state=started enabled=yes

  handlers:
    - name: restart httpd
      service: name=httpd state=restarted
    - name: wall
      command: wall "The config file is changed"
```

案例:

```
---
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: add group nginx
      user: name=nginx state=present
    - name: add user nginx
      user: name=nginx state=present group=nginx
    - name: Install Nginx
      yum: name=nginx state=present
    - name: config
      copy: src=/root/config.txt dest=/etc/nginx/nginx.conf
      notify: ["Restart Nginx","Check Nginx Process"] #或者下面格式
        - Restart Nginx
        - Check Nginx Process

   handlers:
     - name: Restart Nginx
       service: name=nginx state=restarted enabled=yes
     - name: Check Nginx process
       shell: killall -0 nginx &> /tmp/nginx.log
```

范例: 强制调用handlers

```
- hosts: websrvs
  force_handlers: yes #无论task中的任何一个task失败,仍强制调用handlers
  tasks:
    - name: config file
      copy: src=nginx.conf  dest=/etc/nginx/nginx.conf
      notify: restart nginx
    - name: install package
      yum: name=no_exist_package
  handlers:
    - name: restart nginx
      service: name=nginx state=restarted
```

4.8 Playbook中使用tags组件

官方文档:

```
https://docs.ansible.com/ansible/latest/user_guide/playbooks_tags.html
```

在playbook文件中，可以利用tags组件，为特定 task 指定标签，当在执行playbook时，可以只执行特定tags的task,而非整个playbook文件

可以一个task对应多个tag,也可以多个task对应一个tag

还有另外3个特殊关键字用于标签, tagged, untagged 和 all,它们分别是仅运行已标记，只有未标记和所有任务。

 案例:

```
vim  httpd.yml
---
# tags example
- hosts: websrvs
  remote_user: root
  gather_facts: no

  tasks:
    - name: Install httpd
      yum: name=httpd state=present
    - name: Install configure file
      copy: src=files/httpd.conf dest=/etc/httpd/conf/
      tags: [ conf,file ] #写在一行
        - conf     #写成多行
        - file
    - name: start httpd service
      tags: service
      service: name=httpd state=started enabled=yes

[root@ansible ~]#ansible-playbook --list-tags  httpd.yml
[root@ansible ~]#ansible-playbook -t conf,service  httpd.yml
[root@ansible ~]#ansible-playbook --skip-tags conf  httpd.yml
[root@ansible ~]#ansible-playbook   httpd.yml  --skip-tags untagged
```

4.9 Playbook中使用变量

Playbook中同样也支持变量

变量名：仅能由字母、数字和下划线组成，且只能以字母开头

变量定义：

```
variable=value
variable: value
```

范例：

```
http_port=80
http_port: 80
```

变量调用方式：

通过`{{ variable_name }}`调用变量，且变量名前后建议加空格，有时用"{{ variable_name }}"才生效

变量来源：

1. ansible 的 setup facts 远程主机的所有变量都可直接调用
2. 通过命令行指定变量，优先级最高

   ```
   ansible-playbook -e varname=value test.yml
   ```

3. 在playbook文件中定义

   ```
   vars:
     var1: value1
     var2: value2
   ```

4. 在独立的变量YAML文件中定义

   ```
   - hosts: all
     vars_files:
       - vars.yml
   ```

5. 在主机清单文件中定义

   主机（普通）变量：主机组中主机单独定义，优先级高于公共变量
   组（公共）变量：针对主机组中所有主机定义统一变量

6. 在项目中针对主机和主机组定义

   在项目目录中创建 host_vars和group_vars目录

7. 在role中定义

变量的优先级从高到低如下

```
-e 选项定义变量 -->playbook中vars_files --> playbook中vars变量定义 -->host_vars/主机名文件 -->主机清单中主机变量--> group_vars/主机组名文件-->group_vars/all文件--> 主机清单单组变量
```

4.9.1 使用 setup 模块中变量

本模块自动在playbook调用，不要用ansible命令调用,生成的系统状态信息, 并存放在facts变量中

facts 包括的信息很多,如: 主机名,IP,CPU,内存,网卡等

facts 变量的实际使用场景案例

- 通过facts变量获取被控端CPU的个数信息,从而生成不同的Nginx配置文件
- 通过facts变量获取被控端内存大小信息,从而生成不同的memcached的配置文件
- 通过facts变量获取被控端主机名称信息,从而生成不同的Zabbix配置文件
- ....

案例：使用setup变量

```
ansible 10.0.0.101 -m setup -a 'filter="ansible_default_ipv4"'
10.0.0.101 | SUCCESS => {
    "ansible_facts": {
        "ansible_default_ipv4": {
            "address": "10.0.0.101",
            "alias": "eth0",
            "broadcast": "10.0.0.255",
            "gateway": "10.0.0.2",
            "interface": "eth0",
            "macaddress": "00:0c:29:e8:c7:9b",
            "mtu": 1500,
            "netmask": "255.255.255.0",
            "network": "10.0.0.0",
            "type": "ether"
        }
    },
    "changed": false
}


[root@centos8 ~]#ansible 10.0.0.8 -m setup -a "filter=ansible_nodename"
10.0.0.8 | SUCCESS => {
    "ansible_facts": {
        "ansible_nodename": "centos8.magedu.org",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

范例:

```
---
#var1.yml
- hosts: all
  remote_user: root
  gather_facts: yes

  tasks:
    - name: create log file
      file: name=/data/{{ ansible_nodename }}.log state=touch owner=wang
mode=600


[root@ansible ~]#ansible-playbook  var.yml
```

范例: 显示 eth0 网卡的 IP 地址

```
[root@ansible ansible]#cat show_ip.yml
- hosts: websrvs

  tasks:
    - name: show eth0 ip address {{ ansible_facts["eth0"]["ipv4"]["address"] }}
      debug:
        msg: IP address {{ ansible_eth0.ipv4.address }}
        #msg: IP address {{ ansible_facts["eth0"]["ipv4"]["address"] }}
        #msg: IP address {{ ansible_facts.eth0.ipv4.address }}
        #msg: IP address {{ ansible_default_ipv4.address }}
        #msg: IP address {{ ansible_eth0.ipv4.address }}
        #msg: IP address {{ ansible_eth0.ipv4.address.split('.')[-1] }}  #取IP中的
最后一个数字

[root@ansible ansible]#ansible-playbook -v  show_ip.yml
```

范例:

```
[root@centos8 ~]#cat test.yml
---
- hosts: websrvs

  tasks:
    - name: test var
      file: path=/data/{{ ansible_facts["eth0"]["ipv4"]["address"] }}.log
state=touch
      #file: path=/data/{{ ansible_eth0.ipv4.address }}.log state=touch  #和上面效
果一样
[root@centos8 ~]#ansible-playbook  test.yml

[root@centos8 ~]#ll /data/10.0.0.8.log
-rw-r--r-- 1 root root 0 Oct 16 18:22 /data/10.0.0.8.log
```

4.9.2 在playbook 命令行中定义变量

范例:

```
vim var2.yml
---
- hosts: websrvs
  remote_user: root
  tasks:
    - name: install package
      yum: name={{ pkname }} state=present

[root@ansible ~]#ansible-playbook  -e pkname=httpd  var2.yml
```

范例:

```
#也可以将多个变量放在一个文件中
[root@ansible ~]#cat vars
pkname1: memcached
pkname2: vsftpd

[root@ansible ~]#vim var2.yml
---
- hosts: websrvs
  remote_user: root
  tasks:
    - name: install package {{ pkname1 }
      yum: name={{ pkname1 }} state=present
    - name: install package {{ pkname2 }
      yum: name={{ pkname2 }} state=present
```

```
[root@ansible ~]#ansible-playbook  -e pkname1=memcached -e pkname2=httpd
var5.yml
[root@ansible ~]#ansible-playbook  -e '@vars' var2.yml
```

4.9.3 在playbook文件中定义变量

范例:

```
[root@ansible ~]#vim var3.yml
---
- hosts: websrvs
  remote_user: root
  vars:
    username: user1
    groupname: group1

  tasks:
    - name: create group {{ groupname }}
      group: name={{ groupname }} state=present
    - name: create user {{ username }}
      user: name={{ username }} group={{ groupname }} state=present

[root@ansible ~]#ansible-playbook -e "username=user2 groupname=group2"  var3.yml
```

范例：变量的相互调用

```
[root@ansible ~]#cat var4.yaml
---
  - hosts: websrvs
    remote_user: root
    vars:
      collect_info: "/data/test/{{ansible_default_ipv4['address']}}/"

    tasks:
      - name: create IP directory
        file: name="{{collect_info}}" state=directory


#执行结果
tree /data/test/
/data/test/
└── 10.0.0.102

1 directory, 0 files
```

范例: 变量的相互调用

```
[root@ansible ansible]#cat var2.yml
---
- hosts: websrvs
  vars:
    suffix: "txt"
    file: "{{ ansible_nodename }}.{{suffix}}"

  tasks:
    - name: test var
      file: path="/data/{{file}}" state=touch
```

范例: 安装多个包

```
[root@ansible ~]#cat install.yml
- hosts: websrvs
  vars:
    web: httpd
    db: mariadb-server

  tasks:
    - name: install {{ web }} {{ db }}
      yum:
        name:
          - "{{ web }}"
          - "{{ db }}"
        state: latest

[root@ansible ~]#cat install2.yml
- hosts: websrvs
  tasks:
    - name: install packages
      yum: name={{ pack }}
      vars:
        pack:
          - httpd
          - memcached
```

范例：安装指定版本的MySQL

```
[root@ansible ansible]#cat install_mysql.yml
---
# install mysql-5.6.46-linux-glibc2.12-x86_64.tar.gz
- hosts: dbsrvs
  remote_user: root
  gather_facts: no
  vars:
    version: "mysql-5.6.46-linux-glibc2.12-x86_64"
    suffix: "tar.gz"
    file: "{{version}}.{{suffix}}"

  tasks:
    - name: install packages
      yum: name=libaio,perl-Data-Dumper,perl-Getopt-Long
    - name: create mysql group
      group: name=mysql gid=306
    - name: create mysql user
      user: name=mysql uid=306 group=mysql shell=/sbin/nologin system=yes
create_home=no home=/data/mysql
    - name: copy tar to remote host and file mode
      unarchive: src=/data/ansible/files/{{file}} dest=/usr/local/ owner=root
group=root
    - name: create linkfile  /usr/local/mysql
      file: src=/usr/local/{{version}} dest=/usr/local/mysql state=link
    - name: data dir
      shell: chdir=/usr/local/mysql/  ./scripts/mysql_install_db --
datadir=/data/mysql --user=mysql
      tags: data
    - name: config my.cnf
      copy: src=/data/ansible/files/my.cnf  dest=/etc/my.cnf
    - name: service script
      shell: /bin/cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
    - name: enable service
      shell: /etc/init.d/mysqld start;chkconfig --add mysqld;chkconfig mysqld on

      tags: service
    - name: PATH variable
      copy: content='PATH=/usr/local/mysql/bin:$PATH'
dest=/etc/profile.d/mysql.sh
    - name: secure script
      script: /data/ansible/files/secure_mysql.sh
      tags: script
```

4.9.4 使用变量文件

可以在一个独立的playbook文件中定义变量，在另一个playbook文件中引用变量文件中的变量，比playbook中定义的变量优化级高

```
vim vars.yml
---
# variables file
package_name: mariadb-server
service_name: mariadb


vim  var5.yml
---
#install package and start service
- hosts: dbsrvs
  remote_user: root
  vars_files:
    - vars.yml

  tasks:
    - name: install package
      yum: name={{ package_name }}
      tags: install
    - name: start service
      service: name={{ service_name }} state=started enabled=yes
```

范例：

```
cat  vars2.yml
---
var1: httpd
var2: nginx

cat  var6.yml
---
- hosts: web
  remote_user: root
  vars_files:
    - vars2.yml

   tasks:
     - name: create httpd log
       file: name=/app/{{ var1 }}.log state=touch
     - name: create nginx log
       file: name=/app/{{ var2 }}.log state=touch
```

4.9.5 针对主机和主机组的变量

4.9.5.1 在主机清单中针对所有项目的主机和主机分组的变量

所有项目的主机变量

在inventory 主机清单文件中为指定的主机定义变量以便于在playbook中使用

范例：

```
[websrvs]
www1.magedu.com http_port=80 maxRequestsPerChild=808
www2.magedu.com http_port=8080 maxRequestsPerChild=909
```

所有项目的组（公共）变量

在inventory 主机清单文件中赋予给指定组内所有主机上的在playbook中可用的变量，如果和主机变是同名，优先级低于主机变量

范例：

```
[websrvs:vars]
http_port=80
ntp_server=ntp.magedu.com
nfs_server=nfs.magedu.com

[all:vars]
# --------- Main Variables ---------------
# Cluster container-runtime supported: docker, containerd
CONTAINER_RUNTIME="docker"

# Network plugins supported: calico, flannel, kube-router, cilium, kube-ovn
CLUSTER_NETWORK="calico"

# Service proxy mode of kube-proxy: 'iptables' or 'ipvs'
```

```
PROXY_MODE="ipvs"

# K8S Service CIDR, not overlap with node(host) networking
SERVICE_CIDR="192.168.0.0/16"

# Cluster CIDR (Pod CIDR), not overlap with node(host) networking
CLUSTER_CIDR="172.16.0.0/16"

# NodePort Range
NODE_PORT_RANGE="20000-60000"

# Cluster DNS Domain
CLUSTER_DNS_DOMAIN="magedu.local."
```

范例:

```
[root@ansible ~]#vim /etc/ansible/hosts

[websrvs]
10.0.0.8 hname=www1 domain=magedu.io
10.0.0.7 hname=www2

[websrvs:vars]
mark="-"

[all:vars]
domain=magedu.org


[root@ansible ~]#ansible  websrvs  -m hostname -a 'name={{ hname }}{{ mark }}{{ domain }}'

#命令行指定变量:
[root@ansible ~]#ansible  websrvs  -e domain=magedu.cn -m hostname -a  'name={{ hname }}{{ mark }}{{ domain }}'
```

范例: k8s 的ansible 变量文件

```
[etcd]
10.0.0.104 NODE_NAME=etcd1
10.0.0.105 NODE_NAME=etcd2
10.0.0.106 NODE_NAME=etcd3

[kube-master]
10.0.0.103 NEW_MASTER=yes
10.0.0.101
10.0.0.102

[kube-node]
10.0.0.109 NEW_NODE=yes
10.0.0.107
10.0.0.108

[harbor]

[ex-lb]
```

```
10.0.0.111 LB_ROLE=master EX_APISERVER_VIP=10.0.0.100 EX_APISERVER_PORT=8443
10.0.0.112 LB_ROLE=backup EX_APISERVER_VIP=10.0.0.100 EX_APISERVER_PORT=8443

[chrony]

[all:vars]
CONTAINER_RUNTIME="docker"
CLUSTER_NETWORK="calico"
PROXY_MODE="ipvs"
SERVICE_CIDR="192.168.0.0/16"
CLUSTER_CIDR="172.16.0.0/16"
NODE_PORT_RANGE="20000-60000"
CLUSTER_DNS_DOMAIN="magedu.local."
bin_dir="/usr/bin"
ca_dir="/etc/kubernetes/ssl"
base_dir="/etc/ansible"
```

4.9.5.2 针对当前项目的主机和主机组的变量

上面的方式是针对所有项目都有效,而官方更建议的方式是使用ansible特定项目的主机变量和组变量

生产建议在项目目录中创建额外的两个变量目录,分别是host_vars和group_vars

host_vars下面的文件名和主机清单主机名一致,针对单个主机进行变量定义,格式:host_vars/hostname

group_vars下面的文件名和主机清单中组名一致, 针对单个组进行变量定义,格式: gorup_vars/groupname

group_vars/all文件内定义的变量对所有组都有效

范例: 特定项目的主机和组变量

```
[root@ansible ansible]#pwd
/data/ansible

[root@ansible ansible]#mkdir host_vars
[root@ansible ansible]#mkdir group_vars

[root@ansible ansible]#cat host_vars/10.0.0.8
id: 2
[root@ansible ansible]#cat host_vars/10.0.0.7
id: 1
[root@ansible ansible]#cat group_vars/websrvs
name: web
[root@ansible ansible]#cat group_vars/all
domain: magedu.org

[root@ansible ansible]#tree host_vars/ group_vars/
host_vars/
├── 10.0.0.7
└── 10.0.0.8
group_vars/
├── all
└── websrvs

0 directories, 4 files

[root@ansible ansible]#cat test.yml
```

```
- hosts: websrvs

  tasks:
    - name: get variable
      command: echo "{{name}}{{id}}.{{domain}}"
      register: result
    - name: print variable
      debug:
        msg: "{{result.stdout}}"

[root@ansible ansible]#ansible-playbook  test.yml

PLAY [websrvs]
********************************************************************************
***************************************

TASK [Gathering Facts]
********************************************************************************
*******************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [get variable]
********************************************************************************
**********************************
changed: [10.0.0.7]
changed: [10.0.0.8]

TASK [print variable]
********************************************************************************
********************************
ok: [10.0.0.7] => {
    "msg": "web1.magedu.org"
}
ok: [10.0.0.8] => {
    "msg": "web2.magedu.org"
}

PLAY RECAP
********************************************************************************
*******************************************
10.0.0.7                   : ok=3    changed=1    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
10.0.0.8                   : ok=3    changed=1    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
```

4.9.6 register 注册变量

在playbook中可以使用register将捕获命令的输出保存在临时变量中，然后使用debug模块进行显示输出

范例: 利用debug 模块输出变量

```
[root@centos8 ~]#cat register1.yml
- hosts: dbsrvs
```

```
  tasks:
    - name: get variable
      shell: hostname
      register: name
    - name: "print variable"
      debug:
        msg: "{{ name }}"                #输出register注册的name变量的全部信息,注意
变量要加" "引起来
        #msg: "{{ name.cmd }}"           #显示命令
        #msg: "{{ name.rc }}"            #显示命令成功与否
        #msg: "{{ name.stdout }}"        #显示命令的输出结果为字符串形式
        #msg: "{{ name.stdout_lines }}"    #显示命令的输出结果为列表形式
        #msg: "{{ name.stdout_lines[0] }}" #显示命令的输出结果的列表中的第一个元素
        #msg: "{{ name['stdout_lines'] }}" #显示命令的执行结果为列表形式

#说明
第一个 task 中，使用了 register 注册变量名为 name ；当 shell 模块执行完毕后，会将数据放到该
变量中。
第二给 task 中，使用了 debug 模块，并从变量name中获取数据。

[root@centos8 ~]#ansible-playbook  register1.yml

PLAY [dbsrvs]
********************************************************************************
****************************************

TASK [Gathering Facts]
********************************************************************************
*******************************
ok: [10.0.0.7]
ok: [10.0.0.18]

TASK [get variable]
********************************************************************************
**********************************
changed: [10.0.0.7]
changed: [10.0.0.18]

TASK [print variable]
********************************************************************************
********************************
ok: [10.0.0.7] => {
    "msg": {
        "changed": true,
        "cmd": "hostname",
        "delta": "0:00:00.003054",
        "end": "2021-01-27 20:30:39.261396",
        "failed": false,
        "rc": 0,
        "start": "2021-01-27 20:30:39.258342",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "centos7.wangxiaochun.com",
        "stdout_lines": [
            "centos7.wangxiaochun.com"
        ]
    }
}
ok: [10.0.0.18] => {
    "msg": {
        "changed": true,
        "cmd": "hostname",
        "delta": "0:00:00.004216",
        "end": "2021-01-27 20:30:39.369274",
        "failed": false,
        "rc": 0,
        "start": "2021-01-27 20:30:39.365058",
        "stderr": "",
        "stderr_lines": [],
        "stdout": "centos8.wangxiaochun.com",
        "stdout_lines": [
            "centos8.wangxiaochun.com"
        ]
    }
}

PLAY RECAP
********************************************************************************
*******************************************
10.0.0.18                  : ok=3    changed=1    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
10.0.0.7                   : ok=3    changed=1    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
```

范例:使用 register 注册变量创建文件

```
[root@centos8 ~]#cat register2.yml
- hosts: dbsrvs

  tasks:
    - name: get variable
      shell: hostname
      register: name
    - name: create file
      file: dest=/tmp/{{ name.stdout }}.log state=touch
[root@centos8 ~]#ansible-playbook  register2.yml

[root@centos7 ~]#ll /tmp/centos7.wangxiaochun.com.log
-rw-r--r-- 1 root root 0 Jan 27 20:29 /tmp/centos7.wangxiaochun.com.log
```

范例: register和debug模块

```
[root@centos8 ~]#cat debug_test.yml
---
- hosts: dbsrvs
  tasks:
    - shell: echo hello world
      register: say_hi
    - shell: "awk -F: 'NR==1{print $1}' /etc/passwd"
      register: user
    - debug: var=say_hi.stdout    #自定义输出变量代替msg
    - debug: var=user.stdout      #自定义输出变量代替msg
```

```
[root@centos8 ~]#ansible-playbook  debug_test.yml

PLAY [dbsrvs]
*******************************************************************************
*****************************************************************************

TASK [Gathering Facts]
*******************************************************************************
********************************************************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [shell]
*******************************************************************************
******************************************************************************
changed: [10.0.0.7]
changed: [10.0.0.8]

TASK [shell]
*******************************************************************************
******************************************************************************
changed: [10.0.0.7]
changed: [10.0.0.8]

TASK [debug]
*******************************************************************************
******************************************************************************
ok: [10.0.0.8] => {
    "say_hi.stdout": "hello world"
}
ok: [10.0.0.7] => {
    "say_hi.stdout": "hello world"
}

TASK [debug]
*******************************************************************************
******************************************************************************
ok: [10.0.0.8] => {
    "user.stdout": "root"
}
ok: [10.0.0.7] => {
    "user.stdout": "root"
}

PLAY RECAP
*******************************************************************************
*******************************************************************************
*
10.0.0.7                   : ok=5    changed=2    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
10.0.0.8                   : ok=5    changed=2    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
```

范例：安装启动服务并检查

```
[root@ansible ansible]#cat service.yml
---
- hosts: websrvs
  vars:
    package_name: nginx
    service_name: nginx
  tasks:
    - name: install {{ package_name }}
      yum: name={{ package_name }}
    - name: start {{ service_name }}
      service: name={{ service_name }} state=started enabled=yes
    - name: check
      shell: ps axu|grep {{ service_name }}
      register: check_service
    - name: debug
      debug:
        msg: "{{ check_service.stdout_lines }}"
[root@ansible ansible]#ansible-playbook  service.yml

PLAY [websrvs]
********************************************************************************
***************************************

TASK [Gathering Facts]
********************************************************************************
*******************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [install nginx]
********************************************************************************
*********************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [start nginx]
********************************************************************************
***********************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [check]
********************************************************************************
*****************************************
changed: [10.0.0.7]
changed: [10.0.0.8]

TASK [debug]
********************************************************************************
*****************************************
ok: [10.0.0.7] => {
    "msg": [
        "root       6725  0.0  0.1 105500  1980 ?        Ss   12:14   0:00
nginx: master process /usr/sbin/nginx",
        "nginx      6726  0.0  0.2 105968  2920 ?        S    12:14   0:00
nginx: worker process",
        "root       7382  0.0  0.1 113280  1188 pts/1    S+   12:16   0:00 
/bin/sh -c ps axu|grep nginx",
        "root       7384  0.0  0.0 113280   184 pts/1    R+   12:16   0:00 
/bin/sh -c ps axu|grep nginx"
    ]
}
ok: [10.0.0.8] => {
    "msg": [
        "root      31912  0.0  0.2 117748  2164 ?        Ss   12:14   0:00 
nginx: master process /usr/sbin/nginx",
        "nginx     31913  0.0  0.8 149292  7968 ?        S    12:14   0:00 
nginx: worker process",
        "nginx     31914  0.0  0.8 149292  7968 ?        S    12:14   0:00 
nginx: worker process",
        "root      32897  0.0  0.3  12696  2972 pts/1    S+   12:16   0:00 
/bin/sh -c ps axu|grep nginx",
        "root      32899  0.0  0.1  12108  1100 pts/1    S+   12:16   0:00 grep 
nginx"
    ]
}

PLAY RECAP 
********************************************************************************
*******************************************
10.0.0.7                   : ok=5    changed=1    unreachable=0    failed=0    
 skipped=0    rescued=0    ignored=0   
10.0.0.8                   : ok=5    changed=1    unreachable=0    failed=0    
 skipped=0    rescued=0    ignored=0
```

范例: 批量修改主机名

```
[root@ansible ansible]#cat hostname.yml
- hosts: websrvs
  vars:
    host: web
    domain: magedu.org

  tasks:
    - name: get variable
      shell: echo $RANDOM | md5sum | cut -c 1-8
      register: get_random
    - name: print variable
      debug:
        msg: "{{ get_random.stdout }}"
    - name: set hostname
      hostname: name={{ host }}-{{ get_random.stdout }}.{{ domain }}
[root@ansible ansible]#ansible-playbook  hostname.yml
```

4.9.7 实战案例: 利用ansible 的 playbook 批量部署MySQL5.7 或 8.0

4.9.7.1 案例1: 利用register 注册变量批量部署MySQL5.7 或8.0

```
[root@centos8 ~]#mkdir -p /data/ansible/files

[root@centos8 ~]#cd /data/ansible

[root@centos8 ansible]#tree
.
├── files
│   ├── my.cnf
│   ├── mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz
│   └── mysql-8.0.23-linux-glibc2.12-x86_64.tar.xz
└── install_mysql5.7or8.0.yml

1 directory, 4 files
[root@centos8 ansible]#cat files/my.cnf
[mysqld]
server-id=1
log-bin
datadir=/data/mysql
socket=/data/mysql/mysql.sock

log-error=/data/mysql/mysql.log
pid-file=/data/mysql/mysql.pid
[client]
socket=/data/mysql/mysql.sock
[root@centos8 ansible]#cat install_mysql5.7or8.0.yml
---
# install mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz
# install mysql-8.0.23-linux-glibc2.12-x86_64.tar.xz
- hosts: dbsrvs
  remote_user: root
  gather_facts: no
  vars:
    mysql_version: 5.7.33
    mysql_file: mysql-{{mysql_version}}-linux-glibc2.12-x86_64.tar.gz
    mysql_root_password: Magedu@123

  tasks:
    - name: install packages
      yum:
        name:
          - libaio
          - numactl-libs
          - MySQL-python
        state: latest
    - name: create mysql group
      group: name=mysql gid=306
    - name: create mysql user
      user: name=mysql uid=306 group=mysql shell=/sbin/nologin system=yes
create_home=no home=/data/mysql
    - name: copy tar to remote host and file mode
```

```
      unarchive: src=/data/ansible/files/{{mysql_file}} dest=/usr/local/
owner=root group=root
    - name: create linkfile  /usr/local/mysql
      file: src=/usr/local/mysql-{{ mysql_version }}-linux-glibc2.12-x86_64
dest=/usr/local/mysql state=link
    - name: data dir
      shell: /usr/local/mysql/bin/mysqld --initialize --user=mysql --
datadir=/data/mysql
      tags: data
    - name: config my.cnf
      copy: src=/data/ansible/files/my.cnf  dest=/etc/my.cnf
    - name: service script
      shell: /bin/cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
    - name: PATH variable
      copy: content='PATH=/usr/local/mysql/bin:$PATH'
dest=/etc/profile.d/mysql.sh
    - name: enable service
      shell: chkconfig --add mysqld;/etc/init.d/mysqld start
      tags: service
    - name: get password
      shell: awk '/A temporary password/{print $NF}' /data/mysql/mysql.log
      register: password
    - name: change password
      #debug:
      #  msg: "{{ password.stdout }}"
      shell: /usr/local/mysql/bin/mysqladmin  -uroot -p'{{password.stdout}}'
password {{mysql_root_password}}

[root@centos8 ansible]#ansible-playbook  install_mysql5.7or8.0.yml
#注意:第一次执行无法修改密码,需要在目标主机上执行下面操作后再一次playbook
[root@centos8 ansible]#ansible dbsrvs -m shell -a "service mysqld stop ;rm -rf
/data/mysql/*"

#再次执行成功
[root@centos8 ansible]#ansible-playbook  install_mysql5.7or8.0.yml
```

4.9.7.2 案例2: 变量实现部署 MySQL

```
[root@centos8 ~]#grep ^inventory /etc/ansible/ansible.cfg
inventory      = /data/ansible/hosts

[root@centos8 ~]#mkdir -p /data/ansible/files
[root@centos8 ~]#cd /data/ansible
[root@centos8 ansible]#tree
.
├── files
│   ├── my.cnf
│   ├── mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz
│   └── mysql-8.0.23-linux-glibc2.12-x86_64.tar.xz
├── hosts
└── install_mysql5.7or8.0-v2.yml

1 directory, 5 files

[root@centos8 ansible]#cat /data/ansible/hosts
```

```
[dbsrvs]
10.0.0.1[7:8]

[root@centos8 ansible]#cat files/my.cnf
[mysqld]
server-id=1
log-bin
datadir=/data/mysql
socket=/data/mysql/mysql.sock

log-error=/data/mysql/mysql.log
pid-file=/data/mysql/mysql.pid
[client]
socket=/data/mysql/mysql.sock

[root@centos8 ansible]#cat install_mysql5.7or8.0-v2.yml
---
# install mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz
# install mysql-8.0.23-linux-glibc2.12-x86_64.tar.xz
- hosts: dbsrvs
  remote_user: root
  gather_facts: no
  vars:
    mysql_version: 8.0.23
    mysql_file: mysql-{{mysql_version}}-linux-glibc2.12-x86_64.tar.xz
    mysql_root_password: 123456

  tasks:
    - name: install packages
      yum:
        name:
          - libaio
          - numactl-libs
        state: latest
    - name: create mysql group
      group: name=mysql gid=306
    - name: create mysql user
      user: name=mysql uid=306 group=mysql shell=/sbin/nologin system=yes
create_home=no home=/data/mysql
    - name: copy tar to remote host and file mode
      unarchive: src=/data/ansible/files/{{mysql_file}} dest=/usr/local/
owner=root group=root
    - name: create linkfile  /usr/local/mysql
      file: src=/usr/local/mysql-{{ mysql_version }}-linux-glibc2.12-x86_64
dest=/usr/local/mysql state=link
    - name: data dir
      shell: /usr/local/mysql/bin/mysqld --initialize-insecure --user=mysql --
datadir=/data/mysql
      tags: data
    - name: config my.cnf
      copy: src=/data/ansible/files/my.cnf  dest=/etc/my.cnf
    - name: service script
      shell: /bin/cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
    - name: PATH variable
      copy: content='PATH=/usr/local/mysql/bin:$PATH'
dest=/etc/profile.d/mysql.sh
    - name: enable service
```

```
      shell: chkconfig --add mysqld;/etc/init.d/mysqld start
      tags: service
    - name: change password
      shell: /usr/local/mysql/bin/mysqladmin  -uroot  password
{{mysql_root_password}}
```

4.9.7.3 案例3: 部署MySQL 5.7

```
[root@centos8 ansible]#cat files/my.cnf
[mysqld]
server-id=1
log-bin
datadir=/data/mysql
socket=/data/mysql/mysql.sock

log-error=/data/mysql/mysql.log
pid-file=/data/mysql/mysql.pid
[client]
socket=/data/mysql/mysql.sock

[root@centos8 ansible]#cat mysql_install.yml
---
- hosts: dbsrvs
  vars:
    password: 123456
  tasks:
    - name: download mysql-package
      get_url:
        url: http://mirrors.163.com/mysql/Downloads/MySQL-5.7/mysql-5.7.31-linux-
glibc2.12-x86_64.tar.gz
        dest: /usr/local/mysql-5.7.31-linux-glibc2.12-x86_64.tar.gz
        force: yes

    - name: tar mysql-package
      unarchive:
        src: /usr/local/mysql-5.7.31-linux-glibc2.12-x86_64.tar.gz
        dest: /usr/local
        owner: root
        group: root
        mode: 0755
        copy: no

    - name: create linkfile /usr/local/mysql
      file:
        src: /usr/local/mysql-5.7.31-linux-glibc2.12-x86_64
        dest: /usr/local/mysql
        state: link

    - name: create bin link
      shell: "ln -s /usr/local/mysql/bin/* /usr/bin/"
      ignore_errors: yes

    - name: copy my.cnf
      copy:
        src: files/my.cnf
        dest: /etc/my.cnf
    - name: install packages
      yum:
        name: libaio,perl-Data-Dumper,perl-Getopt-Long
        state: present

    - name: create mysql group
      group:
        name: mysql
        gid: 306

    - name: create mysql user
      user:
        name: mysql
        uid: 306
        group: mysql
        shell: /sbin/nologin
        system: yes

    - name: crate work directory
      file:
        path: /data/mysql
        state: directory
        mode: 0755
        owner: mysql
        group: mysql

    - name: Initialization mysql
      shell: "mysqld --initialize --user=mysql --datadir=/data/mysql"
      ignore_errors: yes

    - name: serivce script
      shell: "/bin/cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld;chkconfig --add mysqld;chkconfig mysqld on"

    - name: start service
      service:
        name: mysqld
        state: started
        enabled: yes

    - name: set root password
      shell: "mysqladmin -uroot -p\"`awk '/A temporary password/ {print $NF}'
/data/mysql/mysql.log`\" password {{ password }}"
      register: error
      ignore_errors: yes

    - name: retry set new password
      shell: "mysqladmin -uroot -p'{{ password }}' password {{ password }}"
      when: error.failed
```

4.9.7.4 案例4: 利用shell 脚本部署 MySQL

```
[root@centos8 ansible]# mkdir files
[root@centos8 ansible]# ls files/
mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz

[root@centos8 ansible]# vim files/my.cnf
[mysqld]
server-id=1
log-bin
datadir=/data/mysql
socket=/data/mysql/mysql.sock

log-error=/data/mysql/mysql.log
pid-file=/data/mysql/mysql.pid
[client]
socket=/data/mysql/mysql.sock

[root@centos8 ansible]# vim vars.yml
---
# variables file
mysql_version: 5.7.33

[root@centos8 ansible]# cat files/set_password.sh
#!/bin/bash
MYSQL_ROOT_PASSWORD=123456
MYSQL_OLDPASSWORD=`awk '/A temporary password/{print $NF}'
/data/mysql/mysql.log`
mysqladmin  -uroot -p$MYSQL_OLDPASSWORD password $MYSQL_ROOT_PASSWORD
&>/dev/null

[root@centos8 ansible]# vim install_mysql5.7.yml
---
# install mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz
- hosts: dbsrvs
  remote_user: root
  gather_facts: yes
  vars_files:
    - vars.yml

  tasks:
    - name: install packages for centos7
      yum: name=libaio,perl-Data-Dumper
      when: ansible_facts['distribution_major_version'] == "7"
    - name: install packages for centos8
      yum: name=libaio,perl-Data-Dumper,ncurses-compat-libs
      when: ansible_facts['distribution_major_version'] == "8"
    - name: cteate mysql group
      group: name=mysql gid=306
    - name: create mysql user
      user: name=mysql uid=306 group=mysql shell=/sbin/nologin system=yes
create_home=no home=/data/mysql
    - name: copy tar to remote host and file mode
      unarchive: src=/data/ansible/files/mysql-{{mysql_version}}-linux-
glibc2.12-x86_64.tar.gz dest=/usr/local/ owner=root group=root
    - name: create linkfile /usr/local/mysql
```

```
      file: src=/usr/local/mysql-{{mysql_version}}-linux-glibc2.12-x86_64
dest=/usr/local/mysql state=link
    - name: PATH variable
      copy: content='PATH=/usr/local/mysql/bin:$PATH'
dest=/etc/profile.d/mysql.sh
    - name: PATH variable entry
      shell: . /etc/profile.d/mysql.sh
    - name: config my.cnf
      copy: src=/data/ansible/files/my.cnf dest=/etc/my.cnf
    - name: data dir
      shell: chdir=/usr/local/mysql ./bin/mysqld --initialize --user=mysql --
datadir=/data/mysql
    - name: service script
      shell: /bin/cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld
    - name: enable service
      shell: /etc/init.d/mysqld start;chkconfig --add mysqld;chkconfig mysqld on
      tags: service
    - name: set mysql user password
      script: /data/ansible/files/set_password.sh
      tags: script

[root@centos8 ansible]# tree
.
├── files
│   ├── my.cnf
│   ├── mysql-5.7.33-linux-glibc2.12-x86_64.tar.gz
│   └── set_password.sh
├── install_mysql5.7.yml
└── vars.yml

1 directory, 5 files


[root@centos8 ansible]# ansible-playbook install_mysql5.7.yml
```

4.10 template 模板

模板是一个文本文件，可以做为生成文件的模版，并且模板文件中还可嵌套jinja2语法

4.10.1 jinja2语言

Jinja2 是一个现代的，设计者友好的，仿照 Django 模板的 Python 模板语言。 它速度快，被广泛使用，并且提供了可选的沙箱模板执行环境保证安全:

特性:

- 沙箱中执行
- 强大的 HTML 自动转义系统保护系统免受 XSS
- 模板继承

- 及时编译最优的 python 代码
- 可选提前编译模板的时间
- 易于调试。异常的行数直接指向模板中的对应行。
- 可配置的语法

官方网站:

```
http://jinja.pocoo.org/
https://jinja.palletsprojects.com/en/2.11.x/
```

官方中文文档

```
http://docs.jinkan.org/docs/jinja2/
https://www.w3cschool.cn/yshfid/
```

jinja2 语言支持多种数据类型和操作:

字面量，如: 字符串：使用单引号或双引号,数字：整数，浮点数

列表：[item1, item2, ...]

元组：(item1, item2, ...)

字典：{key1:value1, key2:value2, ...}

布尔型：true/false

算术运算：+, -, *, /, //, %, **

比较操作：==, !=, >, >=, <, <=

逻辑运算：and，or，not

流表达式：For，If，When

字面量：

表达式最简单的形式就是字面量。字面量表示诸如字符串和数值的 Python 对象。如"Hello World"
双引号或单引号中间的一切都是字符串。无论何时你需要在模板中使用一个字符串（比如函数调用、过滤器或只是包含或继承一个模板的参数），如42，42.23
数值可以为整数和浮点数。如果有小数点，则为浮点数，否则为整数。在 Python 里， 42 和 42.0 是不一样的

算术运算：

Jinja 允许用计算值。支持下面的运算符

+：把两个对象加到一起。通常对象是素质，但是如果两者是字符串或列表，你可以用这 种方式来衔接它们。无论如何这不是首选的连接字符串的方式！连接字符串见 ~ 运算符。 {{ 1 + 1 }} 等于 2

-：用第一个数减去第二个数。 {{ 3 - 2 }} 等于 1

/：对两个数做除法。返回值会是一个浮点数。 {{ 1 / 2 }} 等于 0.5

//：对两个数做除法，返回整数商。 {{ 20 // 7 }} 等于 2

%：计算整数除法的余数。 {{ 11 % 7 }} 等于 4

*：用右边的数乘左边的操作数。 {{ 2 * 2 }} 会返回 4 。也可以用于重 复一个字符串多次。 {{ '=' * 80 }} 会打印 80 个等号的横条\

：取左操作数的右操作数次幂。 {{ 23 }} 会返回 8

比较操作符

== 比较两个对象是否相等
!= 比较两个对象是否不等

> 如果左边大于右边，返回 true
>= 如果左边大于等于右边，返回 true
< 如果左边小于右边，返回 true
<= 如果左边小于等于右边，返回 true

逻辑运算符

对于 if 语句，在 for 过滤或 if 表达式中，它可以用于联合多个表达式

and 如果左操作数和右操作数同为真，返回 true

or 如果左操作数和右操作数有一个为真，返回 true

not 对一个表达式取反

(expr)表达式组

true / false true 永远是 true ，而 false 始终是 false

4.10.2 template

template功能：可以根据和参考模块文件，动态生成相类似的配置文件
template文件必须存放于templates目录下，且命名为 .j2 结尾
yaml/yml 文件需和templates目录平级，目录结构如下示例：

```
./
 ├── temnginx.yml
 └── templates
        └── nginx.conf.j2
```

范例：利用template 同步nginx配置文件

```
#准备templates/nginx.conf.j2文件
[root@ansible ~]#vim temnginx.yml
---
- hosts: websrvs
  remote_user: root

  tasks:
    - name: template config to remote hosts
      template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf

 [root@ansible ~]#ansible-playbook temnginx.yml
```

template变更替换

范例：

```
#修改文件nginx.conf.j2
[root@ansible ~]#mkdir templates
[root@ansible ~]#vim templates/nginx.conf.j2
......
worker_processes {{ ansible_processor_vcpus }};
......
```

```
[root@ansible ~]#vim temnginx2.yml
---
- hosts: websrvs
  remote_user: root

  tasks:
    - name: install nginx
      yum: name=nginx
    - name: template config to remote hosts
      template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf
    - name: start service
      service: name=nginx state=started enabled=yes

[root@ansible ~]#ansible-playbook temnginx2.yml
```

template算术运算

范例：

```
vim nginx.conf.j2
worker_processes {{ ansible_processor_vcpus**2 }};
worker_processes {{ ansible_processor_vcpus+2 }};
```

范例：

```
[root@ansible ansible]#vim templates/nginx.conf.j2
worker_processes {{ ansible_processor_vcpus**3 }};

[root@ansible ansible]#cat templnginx.yml
---
- hosts: websrvs
  remote_user: root

  tasks:
    - name: install nginx
      yum: name=nginx
    - name: template config to remote hosts
      template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf
      notify: restart nginx
    - name: start service
      service: name=nginx state=started enabled=yes

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


[root@ansible ~]#-playbook  templnginx.yml --limit 10.0.0.8
```

4.10.3 template中使用流程控制 for 和 if

template中也可以使用流程控制 for 循环和 if 条件判断，实现动态生成文件功能

4.10.3.1 for 循环

格式

```
{% for i in EXPR %}
  ...
{% endfor %}

示例:
{% for i in range(1,10) %}
 server_name web{{i}};
{% endfor %}
```

范例

```
#temlnginx2.yml
---
- hosts: websrvs
  remote_user: root
  vars:
    nginx_vhosts:
      - 81
      - 82
      - 83
  tasks:
    - name: template config
      template: src=nginx.conf2.j2 dest=/data/nginx.conf

#templates/nginx.conf2.j2
{% for vhost in  nginx_vhosts %}
server {
   listen {{ vhost }}
}
{% endfor %}

ansible-playbook -C  templnginx2.yml  --limit 10.0.0.8

#生成的结果:
server {
   listen 81
}
server {
   listen 82
}
server {
   listen 83
}
```

范例:

```
#temlnginx3.yml
---
- hosts: websrvs
  remote_user: root
  vars:
    nginx_vhosts:
      - listen: 8080
  tasks:
    - name: config file
      template: src=nginx.conf3.j2 dest=/data/nginx3.conf

#templates/nginx.conf3.j2
{% for vhost in nginx_vhosts %}
server {
  listen {{ vhost.listen }}
}
{% endfor %}


[root@ansible ~]#ansible-playbook   templnginx3.yml  --limit 10.0.0.8

#生成的结果
server {
  listen 8080
}
```

范例:

```
#templnginx4.yml
- hosts: websrvs
  remote_user: root
  vars:
    nginx_vhosts:
      - listen: 8080
        server_name: "web1.magedu.com"
        root: "/var/www/nginx/web1/"
      - listen: 8081
        server_name: "web2.magedu.com"
        root: "/var/www/nginx/web2/"
      - {listen: 8082, server_name: "web3.magedu.com", root:
"/var/www/nginx/web3/"}
  tasks:
    - name: template config
      template: src=nginx.conf4.j2 dest=/data/nginx4.conf


# templates/nginx.conf4.j2
{% for vhost in nginx_vhosts %}
server {
   listen {{ vhost.listen }}
   server_name {{ vhost.server_name }}
   root {{ vhost.root }}
}
{% endfor %}


[root@ansible ~]#ansible-playbook  templnginx4.yml --limit 10.0.0.8
```

```
#生成结果:
server {
    listen 8080
    server_name web1.magedu.com
    root /var/www/nginx/web1/
}
server {
    listen 8081
    server_name web2.magedu.com
    root /var/www/nginx/web2/
}
server {
    listen 8082
    server_name web3.magedu.com
    root /var/www/nginx/web3/
}
```

范例:

```
#nginx.conf5.j2
upstream webservers {
{% for in range(1,11) %}
 server 10.0.0.{{i}}:{{http_port}}
{% endfor %}

server {
    listen {{http_port}};
    server_name {{server_name}};
    location / {
        proxy_pass http://webservers;
    }
}

#templnginx5.yml
- hosts: websrvs
  vars:
    http_port: 80
    server_name: www.magedu.org
  tasks:
    - name:install nginx
      yum: name=nginx
    - name: config file
      template: src=nginx.conf5.j2 dest=/etc/nginx/conf.d/web_proxy.conf
    - name: start nginx
      service: name=nginx state=started
```

范例:

```
#nginx.conf6.j2
upstream webservers {
{% for in groups['webservers] %}
 server {{i}}:{{http_port}}
{% endfor %}

vim hosts
[webservers]
10.0.0.101
10.0.0.102
```

4.10.3.2 if 条件判断

在模版文件中还可以使用 if条件判断，决定是否生成相关的配置信息

范例：

```
#templnginx6.yml
- hosts: websrvs
  remote_user: root
  vars:
    nginx_vhosts:
      - web1:
        listen: 8080
        root: "/var/www/nginx/web1/"
      - web2:
        listen: 8080
        server_name: "web2.magedu.com"
        root: "/var/www/nginx/web2/"
      - web3:
        listen: 8080
        server_name: "web3.magedu.com"
        root: "/var/www/nginx/web3/"
  tasks:
    - name: template config to
      template: src=nginx.conf5.j2 dest=/data/nginx5.conf


#templates/nginx.conf6.j2
{% for vhost in  nginx_vhosts %}
server {
   listen {{ vhost.listen }}
   {% if vhost.server_name is defined %}
server_name {{ vhost.server_name }}    #注意缩进
   {% endif %}
root  {{ vhost.root }}                 #注意缩进
}
{% endfor %}

#生成的结果
server {
   listen 8080
   root  /var/www/nginx/web1/
}
server {
   listen 8080
   server_name web2.magedu.com
   root  /var/www/nginx/web2/
}
server {
   listen 8080
   server_name web3.magedu.com
   root  /var/www/nginx/web3/
}
```

范例：生成keepalived配置文件

```
vrrp_instrance VI_1 {
{% if ansible_fqdn == "ka1" %}
  state MASTER
  priority 100
{% elif ansible_fqdn == "ka2" %}
  state SLAVE
  priority 80
{% endif% }
......
}
```

4.11 使用循环迭代

迭代：当有需要重复性执行的任务时，可以使用迭代机制

4.11.1 迭代 with_items(loop)

对迭代项的引用，固定内置变量名为"item"

要在task中使用with_items给定要迭代的元素列表

注意: ansible2.5版本后,可以用loop代替with_items

列表元素格式：

- 字符串
- 字典

范例：

```
---
- hosts: websrvs
  remote_user: root

  tasks:
    - name: add several users
      user: name={{ item }} state=present groups=wheel
      with_items:
        - testuser1
        - testuser2
        - testuser3

#上面语句的功能等同于下面的语句
    - name: add several users
```

```
      user: name=testuser1 state=present groups=wheel
    - name: add several users
      user: name=testuser2 state=present groups=wheel
    - name: add several users
      user: name=testuser3 state=present groups=wheel
```

范例：卸载 mariadb

```
---
#remove mariadb server
- hosts: appsrvs:!10.0.0.8
  remote_user: root

  tasks:
    - name: stop service
      shell: /etc/init.d/mysqld stop
    - name:  delete files and dir
      file: path={{item}} state=absent
      with_items:
        - /usr/local/mysql
        - /usr/local/mariadb-10.2.27-linux-x86_64
        - /etc/init.d/mysqld
        - /etc/profile.d/mysql.sh
        - /etc/my.cnf
        - /data/mysql
    - name: delete user
      user: name=mysql state=absent remove=yes
```

范例：

```
---
- hosts：websrvs
  remote_user: root

  tasks
    - name: install some packages
      yum: name={{ item }} state=present
      with_items:
        - nginx
        - memcached
        - php-fpm        -
```

范例：

```
---
- hosts: websrvs
  remote_user: root
  tasks:
    - name: copy file
      copy: src={{ item }} dest=/tmp/{{ item }}
      with_items:
        - file1
        - file2
        - file3
    - name: yum install httpd
      yum: name={{ item }}  state=present
      with_items:
        - apr
        - apr-util
        - httpd
```

4.11.2 迭代嵌套子变量

在迭代中，还可以嵌套子变量，关联多个变量在一起使用

示例：

```
---
- hosts: websrvs
  remote_user: root

  tasks:
    - name: add some groups
      group: name={{ item }} state=present
      with_items:
        - nginx
        - mysql
        - apache
    - name: add some users
      user: name={{ item.user }} group={{ item.group }} uid={{item.uid}}
state=present
      with_items:
        - { user: 'nginx', group: 'nginx',uid: "80" }
        - { user: 'mysql', group: 'mysql' ,uid: "3306"}
        - { user: 'apache', group: 'apache',uid: "8080"}
```

范例：

```
[root@ansible ~]#cat with_item2.yml
---
- hosts: websrvs
  remote_user: root

  tasks:
    - name: add some groups
      group: name={{ item }} state=present
      with_items:
        - g1
        - g2
        - g3
    - name: add some users
      user: name={{ item.name }} group={{ item.group }} home={{ item.home }}
create_home=yes state=present
      with_items:
        - { name: 'user1', group: 'g1', home: '/data/user1' }
        - { name: 'user2', group: 'g2', home: '/data/user2' }
        - { name: 'user3', group: 'g3', home: '/data/user3' }
```

范例:

```
#ansible-doc file
- name: Create two hard links
  file:
    src: '/tmp/{{ item.src }}'
    dest: '{{ item.dest }}'
    state: hard
  loop:
    - { src: x, dest: y }
    - { src: m, dest: n }
```

范例:

```
- hosts: websrvs
  vars:
    rsyncd_conf: /etc/rsync.conf
    rsync_pass: /etc/rsync.pass
  tasks:
  - name: Configure Rsyncd Service
    template: src={{ item.src }} dest={{ item.dest }} mode={{ item.mode }}
    with items:
      - {src: './rsyncd.conf.j2', dest: {{ rsyncd_conf }}, mode: 0644 }
      - {src: './rsync.pass.j2', dest: {{ rsync_pass }}, mode: 0600 }
```

范例: 批量修改用户密码

```
---
  - hosts: ssh-host
    gather_facts: false
    tasks:
    - name: change user passwd
      user: name={{ item.name }} password={{ item.chpass |
password_hash('sha512') }}  update_password=always
      with_items:
           - { name: 'root', chpass: '123456' }
           - { name: 'app', chpass: '654321' }
```

4.11.3 until 循环

范例: until 循环

```
#until为false时才会执行循环,为true则退出循环
[root@ansible ansible]#cat until.yml
- hosts: localhost
  gather_facts: false

  tasks:
    - debug: msg="until"
      until: false
      retries: 3 #默认值即为3次
      delay: 1

[root@ansible ansible]#ansible-playbook  until.yml
```

```
PLAY [localhost]
****************************************************************************************************************

TASK [debug]
********************************************************************************************************************
FAILED - RETRYING: debug (3 retries left).Result was: {
    "attempts": 1,
    "changed": false,
    "msg": "until",
    "retries": 4
}
FAILED - RETRYING: debug (2 retries left).Result was: {
    "attempts": 2,
    "changed": false,
    "msg": "until",
    "retries": 4
}
FAILED - RETRYING: debug (1 retries left).Result was: {
    "attempts": 3,
    "changed": false,
    "msg": "until",
    "retries": 4
}
fatal: [localhost]: FAILED! => {
    "msg": "until"
}

PLAY RECAP
**********************************************************************************************************************
localhost                  : ok=0    changed=0    unreachable=0    failed=1
 skipped=0    rescued=0    ignored=0
```

4.11.4 with_lines 逐行处理

范例: with_lines 逐行处理

```
[root@ansible ansible]#cat with_lines.yml
- hosts: localhost
  tasks:
    - debug: msg={{ item }}
      with_lines: ps aux
```

4.12 playbook使用 when

when语句可以实现条件测试。如果需要根据变量、facts或此前任务的执行结果来做为某task执行与否的前提时要用到条件测试,通过在task后添加when子句即可使用条件测试，jinja2的语法格式

范例：条件判断

```
---
- hosts: websrvs
  remote_user: root
  tasks:
    - name: "shutdown RedHat flavored systems"
      command: /sbin/shutdown -h now
      when: ansible_os_family == "RedHat"
```

范例：对主机名进行条件判断

```
---
- hosts: websrvs
  remote_user: root
  tasks:
    - name: install nginx
      yum: name=nginx
      when: ansible_fqdn is match ("web*")
```

范例： 判断服务状态决定是否重新启动

```
---
- hosts: websrvs
  tasks :
    - name: Check nginx Service #检查nginx服务是否是活动的
      command: systemctl is-active nginx
      ignore_ errors: yes
      register: check_nginx
    - name: Httpd Restart #如果check nginx执行命令结果成功,即check_nginx.rc等于0, 则执行重启nginx,否则跳过
      service: name=nginx state=restarted
      when: check_nginx.rc == 0
```

范例：分组判断

```
tasks:
  - name: "shut down CentOS 6 and Debian 7 systems"
    command: /sbin/shutdown -t now
    when: (ansible_facts['distribution'] == "CentOS" and
ansible_facts['distribution_major_version'] == "6") or
(ansible_facts['distribution'] == "Debian" and
ansible_facts['distribution_major_version'] == "18")
```

范例: when的列表形式表示 and 关系

```
---
#关闭CentOS 7 版本的主机
- hosts: all

  tasks:
    - name: "shut down CentOS 7 systems"
      reboot:
      when:
        - ansible_facts['distribution'] == "CentOS"
        - ansible_facts['distribution_major_version'] == "7"
```

范例: 判断是否定义

```
- hosts: localhost
  tasks:
  - debug: msg="undefined"
    #when: foo is  defined
    when: bar is undefined
```

范例: 和循环一起使用

```
- hosts: localhost
  tasks:
  - debug: msg="item > 3"
    with_items: [1,2,3,4,5]
    when: item > 3
```

范例: 判断执行状态

```
---
- hosts: localhost
  tasks:
  - command: /bin/true

    register: result
    ignore_errors: True
  - debug: msg="failed"
    when: result is failed
  - debug: msg="succeeded"
    when: result is succeeded
  - debug: msg="skipped"
    when: result is skipped
```

范例: failed_when 满足条件时，使任务失败,和when功能相反

```
tasks:
- command: echo failed
  register: result
  failed_when: "'failed' in result.stdout"
  #failed_when: false 不满足条件,任务正常执行
  #failed_when: true 满足条件,使用任务失败
- debug: msg="echo failed_when"
```

范例:

```
---
- hosts: websrvs
  remote_user: root
  tasks:
    - name: add group nginx
      tags: user
      user: name=nginx state=present
    - name: add user nginx
      user: name=nginx state=present group=nginx
    - name: Install Nginx
      yum: name=nginx state=present
    - name: restart Nginx
      service: name=nginx state=restarted
      when: ansible_distribution_major_version == "6"
```

范例:

```
---
- hosts: websrvs
  remote_user: root
  tasks:
    - name: install conf file to centos7
      template: src=nginx.conf.c7.j2 dest=/etc/nginx/nginx.conf
      when: ansible_distribution_major_version == "7"
    - name: install conf file to centos6
      template: src=nginx.conf.c6.j2 dest=/etc/nginx/nginx.conf
      when: ansible_distribution_major_version == "6"
```

4.13 分组 block

当想在满足一个条件下，执行多个任务时，就需要分组了。而不再每个任务都是用when

```
[root@ansible ansible]#cat  block.yml
---
- hosts: localhost

  tasks:
    - block:
        - debug: msg="first"
        - debug: msg="second"
      when:
        - ansible_facts['distribution'] == "CentOS"
        - ansible_facts['distribution_major_version'] == "8"

#相当于下面写法
---
- hosts: localhost

  tasks:
     - debug: msg="first"
       when:
         - ansible_facts['distribution'] == "CentOS"
         - ansible_facts['distribution_major_version'] == "8"
     - debug: msg="second"
       when:
         - ansible_facts['distribution'] == "CentOS"
         - ansible_facts['distribution_major_version'] == "8"
```

4.14 changed_when

4.14.1 关闭 changed 状态

当确定某个task不会对被控制端做修改时但执行结果却显示是黄色的changed状态,可以通过 changed_when: false 关闭changed状态

```
[root@ansible ansible]#cat test_changed.yml
---
- hosts: websrvs

  tasks:
    - name: check sshd service
      shell: ps aux| grep sshd
      changed_when: false  #关闭changed状态
```

4.14.2 利用 changed_when 检查task返回结果

changed_when 检查task返回结果,决定是否继续向下执行

```
[root@ansible ansible]#cat test_changed_when.yml
---
- hosts: websrvs
  tasks:
    - name: install nginx
      yum: name=nginx
    - name: config file
      template: src="nginx.conf.j2" dest="/etc/nginx/nginx.conf"
      notify: restart nginx
    - name: check config
      shell: /usr/sbin/nginx -t
      register: check_nginx_config
      changed_when:
        - (check_nginx_config.stdout.find('successful')) #如果执行结果中有successful字符串,则继续执行,如果没有则停止向下执行
        - false                                          #nginx -t 每次成功执行是changed状态,关闭此changed状态
    - name: start service
      service: name=nginx state=started enabled=yes
  handlers:
    - name: restart nginx
      service: name=nginx state=restarted
```

4.15 滚动执行

管理节点过多导致的超时问题解决方法

默认情况下，Ansible将尝试并行管理playbook中所有的机器。对于滚动更新用例，可以使用serial关键字定义Ansible一次应管理多少主机，还可以将serial关键字指定为百分比，表示每次并行执行的主机数占总数的比例

范例：

```
#vim test_serial.yml
---
- hosts: all
  serial: 2  #每次只同时处理2个主机,将所有task执行完成后,再选下2个主机再执行所有task,直至所有主机
  gather_facts: False

  tasks:
    - name: task one
      comand: hostname
    - name: task two
      command: hostname
```

范例:

```
- name: test serail
  hosts: all
  serial: "20%"   #每次只同时处理20%的主机
```

范例:

```
[root@ansible ansible]#cat test_serial.yml
---
- hosts: websrvs
  serial: 1

  tasks:
    - name: task1
      shell: wall "{{ansible_nodename}} is  running  task1"
    - name: task2
      shell: wall "{{ansible_nodename}} is  running  task2"
    - name: task3
      shell: wall "{{ansible_nodename}} is  running  task3"
```

4.16 委派至其它主机执行

利用委托技术,可以在非当前被控主机的其它主机上执行指定操作

范例:

```
[root@ansible ~]#cat delegate.yml
#在10.0.0.8上执行hostname -I,而非当前主机localhost
- hosts: localhost

  tasks:
    - name: show ip address
      command: hostname -I
      delegate_to: 10.0.0.8
```

范例:

```
#在本地执行ifconfig,而非10.0.0.8
[root@ansible ~]#cat delegate2.yml
- hosts: 10.0.0.8

  tasks:
    - name: show ip address
      local_action: command ifconfig
```

4.18 只执行一次

利用 run_once 指令可以只执行一次,而非在所有被控主机都执行

```
[root@ansible ~]#cat run_once.yml
- hosts: websrvs
  tasks:
    - command: hostname
      run_once: true


[root@ansible ~]#ansible-playbook run_once.yml  --list-hosts

playbook: run_once.yml

  play #1 (websrvs): websrvs    TAGS: []
    pattern: ['websrvs']
    hosts (2):
      10.0.0.8
      10.0.0.7
[root@ansible ~]#ansible-playbook run_once.yml

PLAY [websrvs]
********************************************************************************
***************************************

TASK [Gathering Facts]
********************************************************************************
*******************************
ok: [10.0.0.7]
ok: [10.0.0.8]

TASK [command]
********************************************************************************
***************************************
changed: [10.0.0.7]

PLAY RECAP
********************************************************************************
*******************************************
10.0.0.7                   : ok=2    changed=1    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
10.0.0.8                   : ok=1    changed=0    unreachable=0    failed=0
 skipped=0    rescued=0    ignored=0
```

4.19 环境变量

临时修改环境变量

```
[root@ansible ~]#cat  environment.yml
- hosts: localhost
  tasks:
    - shell: echo $PATH
      environment:
        PATH: /usr/local/app/bin:{{ ansible_env.PATH }}
[root@ansible ~]#ansible-playbook  environment.yml -v
```

4.20 wait_for 等待条件再执行

```
#等待端口可用，才能执行任务
#暂停10s等待端口80打开,否则出错
wait_for: port=80 delay=10

#等待直到锁定文件被删除
wait_for: path=/var/lock/file.lock state=absent
```

4.21 yaml文件的相互调用

利用include 或 include_tasks 可以在某个task中调用其它的只有task内容的yaml文件

```
[root@ansible ansible]#cat a.yml
---
- hosts: websrvs

  tasks:
    - name: run a job
      command: wall run a job
    - name: excute b.yml
      include: b.yml     #调用另一个yaml文件
      #include_tasks: b.yml #另一种写法
[root@ansible ansible]#cat b.yml
- name: run b job
  command: wall run b job
```

也可以将多个包含完整内容的yml文件由一个yml统一调用

```
[root@ansible ansible]#cat total_tasks.yml
- import_playbook: tasks1.yml
- import_playbook: tasks2.yml

[root@ansible ansible]#cat tasks1.yml
---
- hosts: websrvs
  tasks:
    - name: run task1 job
      command: wall run task1 job
[root@ansible ansible]#cat tasks2.yml
---
- hosts: websrvs
  tasks:
    - name: run task2 job
      command: wall run task2 job
```

4.22 实战案例

4.22.1 利用playbook 实现批量编译安装部署 httpd-2.4

```
[root@ansible ansible]#cat install_httpd.yml
---
- hosts: websrvs
  remote_user: root
  vars:
    download_dir: /usr/local/src
    install_dir: /apps/httpd
    httpd_version: httpd-2.4.46
    apr_version: apr-1.7.0
    apr_util_version: apr-util-1.6.1
    httpd_url: https://mirrors.tuna.tsinghua.edu.cn/apache/httpd
    apr_url: https://mirrors.tuna.tsinghua.edu.cn/apache/apr

  tasks:
  - name: install packages
    yum: name=gcc,make,pcre-devel,openssl-devel,expat-devel,bzip2
state=installed
  - name: download httpd file
    unarchive: src="{{ httpd_url }}/{{ httpd_version }}.tar.bz2" dest={{
download_dir }} owner=root remote_src=yes
  - name: download apr file
    unarchive: src="{{ apr_url }}/{{ apr_version }}.tar.bz2" dest={{
download_dir }} owner=root remote_src=yes
  - name: download apr_util file
    unarchive: src="{{ apr_url }}/{{ apr_util_version }}.tar.bz2" dest={{
download_dir }} owner=root remote_src=yes
  - name: prepare apr dir
    shell: chdir={{ download_dir }}   mv {{ apr_version }} {{ download_dir }}/{{
httpd_version }}/srclib/apr
  - name: prepare apr_util dir
    shell: chdir={{ download_dir }}   mv {{ apr_util_version }} {{ download_dir
}}/{{ httpd_version }}/srclib/apr-util
  - name: build httpd
    shell: chdir={{ download_dir }}/{{ httpd_version }}  ./configure --prefix={{
install_dir }} --enable-so  --enable-ssl  --enable-cgi  --enable-rewrite  --
with-zlib  --with-pcre --with-included-apr  --enable-modules=most  --enable-
mpms-shared=all  --with-mpm=prefork &&  make -j {{ ansible_processor_vcpus }} &&
make install
  - name: create group
    group: name=apache gid=80 system=yes
  - name: create user
    user: name=apache uid=80 group=apache shell=/sbin/nologin system=yes
create_home=no home={{ install_dir }}/conf/httpd
  - name: set httpd  user
    lineinfile: path={{ install_dir }}/conf/httpd.conf regexp='^User' line='User
apache'
  - name: set httpd group
    lineinfile: path={{ install_dir }}/conf/httpd.conf regexp='^Group'
line='Group apache'
  - name: set variable PATH
```

```
    shell: echo PATH={{ install_dir }}/bin:$PATH >> /etc/profile.d/httpd.sh
  - name: prepare service file
    template: src=./httpd.service.j2 dest=/usr/lib/systemd/system/httpd.service
  - name: start service
    service: name=httpd state=started enabled=yes


[root@ansible ansible]#cat httpd.service.j2
[Unit]
Description=The Apache HTTP Server
After=network.target remote-fs.target nss-lookup.target
Documentation=man:httpd(8)
Documentation=man:apachectl(8)

[Service]
Type=forking
#EnvironmentFile=/etc/sysconfig/httpd
ExecStart={{ install_dir }}/bin/apachectl start
#ExecStart={{ install_dir }}/bin/httpd $OPTIONS -k start
ExecReload={{ install_dir }}/bin/apachectl graceful
#ExecReload={{ install_dir }}/bin/httpd $OPTIONS -k graceful
ExecStop={{ install_dir }}/bin/apachectl stop
KillSignal=SIGCONT
PrivateTmp=true
[Install]
WantedBy=multi-user.target

[root@ansible ansible]#ansible-playbook   install_httpd.yml
```

4.22.2 利用 playbook 安装 docker

```
[root@centos8 ansible]# vim /etc/ansible/hosts
[ubuntu]
10.0.0.100
10.0.0.200


[root@centos8 ansible]# vim vars.yml
docker_version: 5:19.03.15~3-0~ubuntu-
ubuntu1804: bionic
ubuntu2004: focal


[root@centos8 ansible]# vim files/daemon.json
{
  "registry-mirrors": ["https://si7y70hh.mirror.aliyuncs.com"]

}


[root@centos8 ansible]# vim install_docker.yml
---
#install docker
- hosts: ubuntu
  remote_user: root
  vars_files:
    - vars.yml

  tasks:
    - name: install packages
      apt: name=apt-transport-https,ca-certificates,curl,software-properties-
common state=present
    - name: import key
      shell: curl -fsSL https://mirrors.tuna.tsinghua.edu.cn/docker-
ce/linux/ubuntu/gpg | sudo apt-key add -
    - name: import installation source on ubuntu1804
      shell: add-apt-repository  "deb [arch=amd64]
https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu  {{ubuntu1804}}
stable"
      when:
        - ansible_facts['distribution_major_version'] == "18"
    - name: import installation source on ubuntu2004
      shell: add-apt-repository  "deb [arch=amd64]
https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu  {{ubuntu2004}}
stable"
      when:
        - ansible_facts['distribution_major_version'] == "20"
    - name: install docker for ubuntu1804
      apt: name=docker-ce={{docker_version}}{{ubuntu1804}},docker-ce-cli=
{{docker_version}}{{ubuntu1804}}
      when:
        - ansible_facts['distribution_major_version'] == "18"
    - name: install docker for ubuntu2004
      apt: name=docker-ce={{docker_version}}{{ubuntu2004}},docker-ce-cli=
{{docker_version}}{{ubuntu2004}}
      when:
        - ansible_facts['distribution_major_version'] == "20"
    - name: mkdir /etc/docker
      file: path=/etc/docker state=directory
    - name: aliyun Mirror acceleration
      copy: src=/data/ansible/files/daemon.json dest=/etc/docker/
    - name: load daemon
      shell: systemctl daemon-reload
    - name: start docker
      service: name=docker state=started enabled=yes


[root@centos8 ansible]# ansible-playbook install_dokcer.yml

#验证安装是否成功
[root@centos8 ansible]# ansible ubuntu -a "docker version"
```

4.22.3 利用 playbook 安装 docker harbor

```
[root@centos8 ansible]# vim vars.yml
docker_version: 5:19.03.15~3-0~ubuntu-
ubuntu1804: bionic
ubuntu2004: focal
docker_compose_version: 1.27.4
harbor_version: 1.7.6
```

```
[root@centos8 ansible]# vim files/harbor.sh
#!/bin/bash

IPADDR=`hostname -I|awk '{print $1}'`
HARBOR_ADMIN_PASSWORD=123456
sed -i.bak -e 's/^hostname =.*/hostname = '$IPADDR'/' -e
's/^harbor_admin_password =.*/harbor_admin_password = '$HARBOR_ADMIN_PASSWORD'/'
/apps/harbor/harbor.cfg


[root@centos8 files]# vim files/harbor.service
[Unit]
Description=Harbor
After=docker.service systemd-networkd.service systemd-resolved.service
Requires=docker.service
Documentation=http://github.com/vmware/harbor

[Service]
Type=simple
Restart=on-failure
RestartSec=5
ExecStart=/usr/bin/docker-compose -f /apps/harbor/docker-compose.yml up
ExecStop=/usr/bin/docker-compose -f /apps/harbor/docker-compose.yml down

[Install]
WantedBy=multi-user.target

[root@centos8 ansible]# vim files/daemon.json
{
  "registry-mirrors": ["https://si7y70hh.mirror.aliyuncs.com"]

}


[root@centos8 ansible]# ls files/
daemon.json  docker-compose-Linux-x86_64-1.27.4  harbor-offline-installer-
v1.7.6.tgz  harbor.service  harbor.sh

[root@centos8 ansible]# vim install_docker_harbor.yml
---
#install docker harbor
- hosts: ubuntu
  remote_user: root
  vars_files:
    - vars.yml

  tasks:
    - name: install packages
      apt: name=apt-transport-https,ca-certificates,curl,software-properties-
common state=present
    - name: import key
      shell: curl -fsSL https://mirrors.tuna.tsinghua.edu.cn/docker-
ce/linux/ubuntu/gpg | sudo apt-key add -
    - name: import installation source on ubuntu1804
      shell: add-apt-repository  "deb [arch=amd64]
https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu  {{ubuntu1804}}
stable"
      when:
        - ansible_facts['distribution_major_version'] == "18"
    - name: import installation source on ubuntu2004
      shell: add-apt-repository  "deb [arch=amd64]
https://mirrors.tuna.tsinghua.edu.cn/docker-ce/linux/ubuntu  {{ubuntu2004}}
stable"
      when:
        - ansible_facts['distribution_major_version'] == "20"
    - name: install docker for ubuntu1804
      apt: name=docker-ce={{docker_version}}{{ubuntu1804}},docker-ce-cli=
{{docker_version}}{{ubuntu1804}}
      when:
        - ansible_facts['distribution_major_version'] == "18"
    - name: install docker for ubuntu2004
      apt: name=docker-ce={{docker_version}}{{ubuntu2004}},docker-ce-cli=
{{docker_version}}{{ubuntu2004}}
      when:
        - ansible_facts['distribution_major_version'] == "20"
    - name: mkdir /etc/docker
      file: path=/etc/docker state=directory
    - name: aliyun Mirror acceleration
      copy: src=/data/ansible/files/daemon.json dest=/etc/docker/
    - name: load daemon
      shell: systemctl daemon-reload
    - name: start docker
      service: name=docker state=started enabled=yes
    - name: install compose
      copy: src=/data/ansible/files/docker-compose-Linux-x86_64-
{{docker_compose_version}} dest=/usr/bin/docker-compose
    - name: set excute permission
      file: path=/usr/bin/docker-compose mode=755
    - name: mkdir /apps
      file: path=/apps state=directory
    - name: unarchive  harbor package
      unarchive: src=/data/ansible/files/harbor-offline-installer-
v{{harbor_version}}.tgz dest=/apps/
    - name: set harbor.cfg
      script: /data/ansible/files/harbor.sh
    - name: install python
      apt: name=python
    - name: install harbor
      shell: /apps/harbor/install.sh
    - name: copy harbor.service
      copy: src=/data/ansible/files/harbor.service dest=/lib/systemd/system/
    - name: service enable
      shell: systemctl daemon-reload;systemctl enable harbor

[root@centos8 ansible]# ansible-playbook install_docker_harbor.yml
```

5 roles 角色

角色是ansible自1.2版本引入的新特性，用于层次性、结构化地组织playbook。roles能够根据层次型结构自动装载变量文件、tasks以及handlers等。要使用roles只需要在playbook中使用include指令即可。简单来讲，roles就是通过分别将变量、文件、任务、模板及处理器放置于单独的目录中，并可以便捷地include它们的一种机制。角色一般用于基于主机构建服务的场景中，但也可以是用于构建守护进程等场景中

运维复杂的场景：建议使用 roles，代码复用度高

roles：多个角色的集合目录， 可以将多个的role，分别放至roles目录下的独立子目录中,如下示例

```
roles/
    mysql/
    nginx/
    tomcat/
    redis/
```

默认roles存放路径

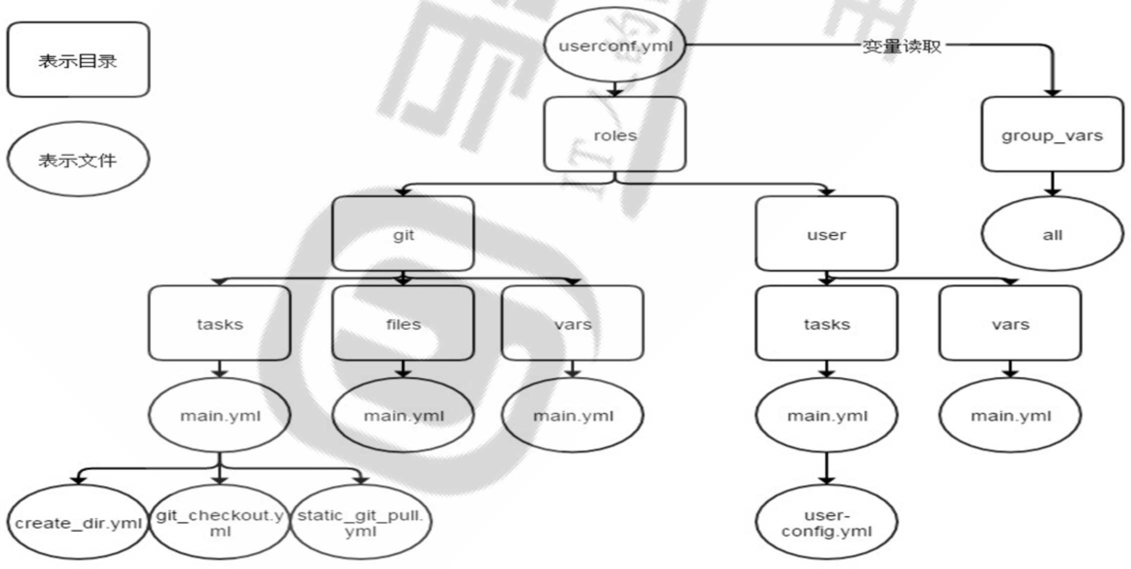
```
/root/.ansible/roles
/usr/share/ansible/roles
/etc/ansible/roles
```

官方文档:

```
https://docs.ansible.com/ansible/latest/user_guide/playbooks_reuse_roles.html
```

5.1 Ansible Roles目录编排

roles目录结构如下所示



每个角色，以特定的层级目录结构进行组织

roles目录结构：

```
playbook1.yml
playbook2.yml
roles/
  project1/
    tasks/
    files/
    vars/
    templates/
    handlers/
    default/
    meta/
  project2/
    tasks/
    files/
    vars/
    templates/
    handlers/
    default/
    meta/
```

Roles各目录作用

roles/project/ :项目名称,有以下子目录

- files/ ： 存放由copy或script模块等调用的文件
- templates/：template模块查找所需要模板文件的目录
- tasks/：定义task,role的基本元素，至少应该包含一个名为main.yml的文件；其它的文件需要在此文件中通过include进行包含
- handlers/：至少应该包含一个名为main.yml的文件；此目录下的其它的文件需要在此文件中通过include进行包含
- vars/：定义变量，至少应该包含一个名为main.yml的文件；此目录下的其它的变量文件需要在此文件中通过include进行包含
- meta/：定义当前角色的特殊设定及其依赖关系,至少应该包含一个名为main.yml的文件，其它文件需在此文件中通过include进行包含
- default/：设定默认变量时使用此目录中的main.yml文件，比vars的优先级低

5.2 创建 role

创建role的步骤

```
1 创建role的目录结构.在以roles命名的目录下分别创建以各角色名称命名的目录，如mysql等,在每个角色命名的目录中分别创建相关的目录和文件,比如tasks、files、handlers、templates和vars等目录；用不到的目录可以创建为空目录，也可以不创建
2 编写和准备role的功能文件
3 编写playbook文件调用需要的角色应用于指定的主机
```

针对大型项目使用Roles进行编排

范例：利用 ansible-galaxy 创建角色目录的结构

```
[root@ansible ansible]#ansible-galaxy  init test_role
- Role test_role was created successfully
[root@ansible ansible]#tree test_role/
test_role/
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
|   ├── inventory
|   └── test.yml
└── vars
    └── main.yml

8 directories, 8 files
```

范例：roles的目录结构

```
nginx-role.yml
roles/
└── nginx
     ├── files
     |    └── nginx.conf
     ├── tasks
     |    ├── groupadd.yml
     |    ├── install.yml
     |    ├── main.yml
     |    ├── restart.yml
     |    └── useradd.yml
     └── vars
          └── main.yml
```

5.3 playbook 调用角色

调用角色方法1：

```
---
- hosts: websrvs
  remote_user: root
  roles:
    - mysql
    - memcached
    - nginx
```

调用角色方法2：

键role用于指定角色名称，后续的k/v用于传递变量给角色

```
---
- hosts: all
  remote_user: root
  roles:
    - role: mysql
      username: mysql
    - { role: nginx, username: nginx }
```

调用角色方法3：

还可基于条件测试实现角色调用

```
---
- hosts: all
  remote_user: root
  roles:
    - { role: nginx, username: nginx, when: ansible_distribution_major_version ==
'7'  }
```

范例:

```
---
- hosts: webservers
  roles:
    - common
    - role: foo_app_instance
      vars:
        dir: '/opt/a'
        app_port: 5000
      tags: typeA
    - role: foo_app_instance
      vars:
        dir: '/opt/b'
        app_port: 5001
      tags: typeB
```

范例:

```
---
- hosts: webservers
  roles:
    - { role: foo, vars: { message: "first" } }
    - { role: foo, vars: { message: "second" } }
```

5.4 roles 中 tags 使用

```
[root@ansible ~]#vi app-role.yml
---
#可以有多个play
- hosts: lbserver
  roles:
    - role: haproxy
    - role: keepalived

- hosts: appsrvs
  remote_user: root
  roles:
    - { role: nginx ,tags: [ 'nginx', 'web' ] ,when:
ansible_distribution_major_version == "6" }
    - { role: httpd ,tags: [ 'httpd', 'web' ]  }
    - { role: mysql ,tags: [ 'mysql', 'db' ] }
    - role: mariadb
```

```
      tags:
        - mariadb
        - db
  tags: app  #play的tag

[root@ansible ~]#ansible-playbook --tags="nginx,mysql" app-role.yml
```

5.5 实战案例

5.5.1 案例1：实现 httpd 角色

```
#创建角色相关的目录
[root@ansible ~]#mkdir -pv /data/ansible/roles/httpd/{tasks,handlers,files}

#创建角色相关的文件
[root@ansible ~]#cd /data/ansible/roles/httpd/

#main.yml 是task的入口文件
[root@ansible ~]#vim tasks/main.yml
- include: group.yml
- include: user.yml
- include: install.yml
- include: config.yml
- include: index.yml
- include: service.yml

[root@ansible ~]#vim  tasks/group.yml
- name: create apache group
  group: name=apache system=yes gid=80

[root@ansible ~]#vim  tasks/user.yml
- name: create apache user
  user: name=apache system=yes shell=/sbin/nologin home=/var/www/ uid=80
group=apache

[root@ansible ~]#vim tasks/install.yml
- name: install httpd package
  yum: name=httpd

[root@ansible ~]#vim tasks/config.yml
- name: config file
  copy: src=httpd.conf dest=/etc/httpd/conf/ backup=yes
  notify: restart

[root@ansible ~]# tasks/index.yml
- name: index.html
  copy: src=index.html dest=/var/www/html/

[root@ansible ~]#vim tasks/service.yml
- name: start service
  service: name=httpd state=started enabled=yes

[root@ansible ~]#vim handlers/main.yml
- name: restart
  service: name=httpd state=restarted
```

```
#在files目录下准备两个文件
[root@ansible ~]#ls files/
httpd.conf index.html

[root@ansible ~]#tree /data/ansible/roles/httpd/
/data/ansible/roles/httpd/
├── files
│   ├── httpd.conf
│   └── index.html
├── handlers
│   └── main.yml
└── tasks
    ├── config.yml
    ├── group.yml
    ├── index.yml
    ├── install.yml
    ├── main.yml
    ├── service.yml
    └── user.yml

3 directories, 10 files


#在playbook中调用角色
[root@ansible ~]#vim  /data/ansible/role_httpd.yml
---
# httpd role
- hosts: websrvs
  remote_user: root

  roles:
    - httpd

#运行playbook
[root@ansible ~]#ansible-playbook  /data/ansible/role_httpd.yml
```

5.5.2 案例2：实现 nginx 角色

```
[root@ansible ~]#mkdir -pv
/data/ansible/roles/nginx/{tasks,handlers,templates,vars}

#创建task文件
[root@ansible ~]#cd /data/ansible/roles/nginx/

[root@ansible nginx]#vim tasks/main.yml
- include: install.yml
- include: config.yml
- include: index.yml
- include: service.yml

[root@ansible nginx]#vim  tasks/install.yml
- name: install
  yum: name=nginx

[root@ansible nginx]#vim tasks/config.yml
```

```
- name: config file for centos7
  template: src=nginx7.conf.j2 dest=/etc/nginx/nginx.conf
  when: ansible_distribution_major_version=="7"
  notify: restart
- name: config file for centos8
  template: src=nginx8.conf.j2 dest=/etc/nginx/nginx.conf
  when: ansible_distribution_major_version=="8"
  notify: restart

#跨角色调用文件
[root@ansible nginx]#vim  tasks/index.yml
- name: index.html
  copy: src=roles/httpd/files/index.html dest=/usr/share/nginx/html/

[root@ansible nginx]#vim tasks/service.yml
- name: start service
  service: name=nginx state=started enabled=yes

#创建handler文件
[root@ansible nginx]#cat handlers/main.yml
- name: restart
  service: name=nginx state=restarted

#创建两个template文件
[root@ansible nginx]#cat templates/nginx7.conf.j2
...省略...
user {{user}};
worker_processes {{ansible_processor_vcpus+3}};   #修改此行
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
...省略...


[root@ansible nginx]#cat templates/nginx8.conf.j2
...省略...
user {{user}};
worker_processes {{ansible_processor_vcpus**3}};  #修改此行
error_log /var/log/nginx/error.log;
pid /run/nginx.pid;
...省略...

#创建变量文件
[root@ansible nginx]#vim vars/main.yml
user: daemon


#目录结构如下

[root@ansible ~]#tree /data/ansible/roles/nginx/
/data/ansible/roles/nginx/
├── handlers
│   └── main.yml
├── tasks
│   ├── config.yml
│   ├── file.yml
│   ├── install.yml
│   ├── main.yml
│   └── service.yml
```

```
├── templates
|   ├── nginx7.conf.j2
|   └── nginx8.conf.j2
└── vars
    └── main.yml

4 directories, 9 files


#在playbook中调用角色
[root@ansible ~]#vim /data/ansible/role_nginx.yml
---
#nginx role
- hosts: websrvs

  roles:
    - role: nginx

#运行playbook
[root@ansible ~]#ansible-playbook  /data/ansible/role_nginx.yml
```

5.5.3 案例3：实现 memcached 角色

```
[root@ansible ~]#mkdir -pv  /data/ansible/roles/memcached/{tasks,templates}

[root@ansible ~]#cd /data/ansible/roles/memcached
[root@ansible memcached]#vim tasks/main.yml
- include: install.yml
- include: config.yml
- include: service.yml

[root@ansible memcached]#vim tasks/install.yml
- name: install
  yum: name=memcached

[root@ansible memcached]#vim tasks/config.yml
- name: config file
  template: src=memcached.j2  dest=/etc/sysconfig/memcached

[root@ansible memcached]#vim tasks/service.yml
- name: service
  service: name=memcached state=started enabled=yes

[root@ansible memcached]#vim templates/memcached.j2
PORT="11211"
USER="memcached"
MAXCONN="1024"
CACHESIZE="{{ansible_memtotal_mb//4}}"
OPTIONS=""

[root@ansible memcached]#tree /data/ansible/roles/memcached/
/data/ansible/roles/memcached/
├── tasks
|   ├── config.yml
|   ├── install.yml
|   ├── main.yml
|   └── service.yml
└── templates
    └── memcached.j2

2 directories, 5 files

[root@ansible ~]#vim /data/ansible/role_memcached.yml
---
- hosts: appsrvs

  roles:
    - role: memcached


[root@ansible ~]#ansible-play /data/ansible/role_memcached.yml
```

5.5.4 案例4：实现MySQL5.7 或8.0 的角色

```
[root@ansible mysql]#pwd
/data/ansible/roles/mysql

[root@ansible mysql]#tree /data/ansible/roles/mysql/
.
├── files
|   ├── my.cnf
|   └── mysql-8.0.23-linux-glibc2.12-x86_64.tar.xz
├── tasks
|   ├── config.yml
|   ├── data.yml
|   ├── group.yml
|   ├── install.yml
|   ├── linkfile.yml
|   ├── main.yml
|   ├── path.yml
|   ├── script.yml
|   ├── secure.yml
|   ├── service.yml
|   ├── unarchive.yml
|   └── user.yml
└── vars
    └── main.yml

3 directories, 15 files

[root@ansible mysql]#cat /data/ansible/roles/mysql/files/my.cnf
[mysqld]
server-id=1
log-bin
datadir=/data/mysql
socket=/data/mysql/mysql.sock

log-error=/data/mysql/mysql.log
pid-file=/data/mysql/mysql.pid
[client]
socket=/data/mysql/mysql.sock
```

```
[root@ansible mysql]#cat /data/ansible/roles/mysql/vars/main.yml
mysql_version: 8.0.23
mysql_file: mysql-{{mysql_version}}-linux-glibc2.12-x86_64.tar.xz
mysql_root_password: 123456


[root@ansible mysql]#cat /data/ansible/roles/mysql/tasks/main.yml
- include: install.yml
- include: group.yml
- include: user.yml
- include: unarchive.yml
- include: linkfile.yml
- include: data.yml
- include: config.yml
- include: script.yml
- include: path.yml
- include: service.yml
- include: secure.yml


[root@ansible mysql]#cat /data/ansible/roles/mysql/tasks/install.yml
- name: install packages
  yum:
    name:
      - libaio
      - numactl-libs
[root@ansible mysql]#cat tasks/group.yml
- name: create mysql group
  group: name=mysql gid=306
[root@ansible mysql]#cat tasks/user.yml
- name: create mysql user
  user: name=mysql uid=306 group=mysql shell=/sbin/nologin system=yes
create_home=no home=/data/mysql
[root@ansible mysql]#cat tasks/unarchive.yml
- name: copy tar to remote host and file mode
  unarchive: src={{mysql_file}} dest=/usr/local/ owner=root group=root
[root@ansible mysql]#cat tasks/linkfile.yml
- name: create linkfile  /usr/local/mysql
  file: src=/usr/local/mysql-{{ mysql_version }}-linux-glibc2.12-x86_64
dest=/usr/local/mysql  state=link
[root@ansible mysql]#cat tasks/data.yml
- name: data dir
  shell: /usr/local/mysql/bin/mysqld --initialize-insecure --user=mysql --
datadir=/data/mysql
  tags: data
[root@ansible mysql]#cat tasks/config.yml
- name: config my.cnf
  copy: src=/data/ansible/files/my.cnf  dest=/etc/my.cnf
[root@ansible mysql]#cat tasks/script.yml
- name: service script
  shell: /bin/cp /usr/local/mysql/support-files/mysql.server /etc/init.d/mysqld
[root@ansible mysql]#cat tasks/path.yml
- name: PATH variable
  copy: content='PATH=/usr/local/mysql/bin:$PATH' dest=/etc/profile.d/mysql.sh
[root@ansible mysql]#cat tasks/service.yml
- name: enable service
  shell: chkconfig --add mysqld;/etc/init.d/mysqld start
  tags: service
[root@ansible mysql]#cat tasks/secure.yml
- name: change password
  shell: /usr/local/mysql/bin/mysqladmin  -uroot  password
{{mysql_root_password}}


[root@ansible ansible]#cat /data/ansible/role_mysql.yml
---
- hosts: dbsrvs
  remote_user: root
  gather_facts: no

  roles:
    - mysql

[root@ansible ansible]#ansible-playbook   role_mysql.yml
```

5.5.5 案例5：实现MySQL 5.6 的角色

```
[root@ansible ~]#cat /data/ansible/roles/mysql/files/my.cnf
[mysqld]
socket=/tmp/mysql.sock
user=mysql
symbolic-links=0
datadir=/data/mysql
innodb_file_per_table=1
log-bin
pid-file=/data/mysql/mysqld.pid

[client]
port=3306
socket=/tmp/mysql.sock

[mysqld_safe]
log-error=/var/log/mysqld.log

[root@ansible ~]#cat /data/ansible/roles/mysql/files/secure_mysql.sh
#!/bin/bash
/usr/local/mysql/bin/mysql_secure_installation <<EOF

y
magedu
magedu
y
y
y
y
EOF

[root@ansible ~]#chmod +x  /data/ansible/roles/mysql/files/secure_mysql.sh

[root@ansible ~]#ls /data/ansible/roles/mysql/files/
my.cnf  mysql-5.6.46-linux-glibc2.12-x86_64.tar.gz  secure_mysql.sh
```

```
[root@ansible ~]#cat /data/ansible/roles/mysql/vars/main.yml
mysql_version: 5.6.46-linux-glibc2.12-x86_64
mysql_file: mysql-{{mysql_version}}.tar.gz


[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/main.yml
- include: install.yml
- include: group.yml
- include: user.yml
- include: unarchive.yml
- include: link.yml
- include: data.yml
- include: config.yml
- include: service.yml
- include: path.yml
- include: secure.yml

[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/install.yml
- name: install packages
  yum: name=libaio,perl-Data-Dumper,perl-Getopt-Long
[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/group.yml
- name: create mysql group
  group: name=mysql gid=306
[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/user.yml
- name: create mysql user
  user: name=mysql uid=306 group=mysql shell=/sbin/nologin system=yes
create_home=no home=/data/mysql
[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/unarchive.yml
- name: copy tar to remote host and file mode
  unarchive: src={{mysql_file}} dest=/usr/local/ owner=root group=root
[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/link.yml
- name: mkdir /usr/local/mysql
  file: src=/usr/local/{{mysql_version}} dest=/usr/local/mysql state=link
[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/data.yml
- name: data dir
  shell: chdir=/usr/local/mysql/  ./scripts/mysql_install_db --
datadir=/data/mysql --user=mysql
[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/config.yml
- name: config my.cnf
  copy: src=my.cnf  dest=/etc/my.cnf
[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/service.yml
- name: service script
  shell: /bin/cp /usr/local/mysql/support-files/mysql.server
/etc/init.d/mysqld;chkconfig --add mysqld;chkconfig mysqld on;/etc/init.d/mysqld
start

[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/path.yml
- name: PATH variable
  copy: content='PATH=/usr/local/mysql/bin:$PATH' dest=/etc/profile.d/mysql.sh

[root@ansible ~]#cat /data/ansible/roles/mysql/tasks/secure.yml
- name: secure script
  script: secure_mysql.sh

[root@ansible ~]#tree /data/ansible/roles/mysql/
/data/ansible/roles/mysql/
├── files
│   ├── my.cnf
```

```
|   ├── mysql-5.6.46-linux-glibc2.12-x86_64.tar.gz
|   └── secure_mysql.sh
└── tasks
    ├── config.yml
    ├── data.yml
    ├── group.yml
    ├── install.yml
    ├── link.yml
    ├── main.yml
    ├── path.yml
    ├── secure.yml
    ├── service.yml
    ├── unarchive.yml
    └── user.yml

2 directories, 14 files

[root@ansible ~]#cat /data/ansible/mysql_roles.yml
- hosts: dbsrvs
  remote_user: root

  roles:
    - {role: mysql,tags: ["mysql","db"]}
    - {role: nginx,tage: ["nginx","web"]}

[root@ansible ~]#ansible-playbook -t mysql /data/ansible/mysql_roles.yml
```

5.5.6 案例6：调用变量

```
[root@ansible ~]#vim /data/ansible/roles/test_role/tasks/main.yml
- name: Include OS-specific variables
  include_vars: {{ ansible_os_family}}.yml
....
[root@ansible ~]#ls /data/ansible/roles/test_role/vars/
Archlinux.yml Debian.yml FreeBSD.yml OpenBSD.ymL RedHat.yml
```

5.5.7 案例7：实现多角色的选择

```
[root@ansible ~]#vim /data/ansible/role_httpd_nginx.yml
---
- hosts: websrvs

  roles:
    - {role: httpd,tags: [httpd,web], when:
ansible_distribution_major_version=="7" }
    - {role: nginx,tags: [nginx,web], when:
ansible_distribution_major_version=="8" }

[root@ansible ~]#ansible-playbook -t nginx /data/ansible/role_httpd_nginx.yml
```

5.5.8 案例8: 依赖其它角色

```
[root@ansible roles]#vim wordpress/meta/main.yml
dependenices:
  - role: nginx
  - role: php-fpm
  - role: mysql
```

6 ansible tower 介绍

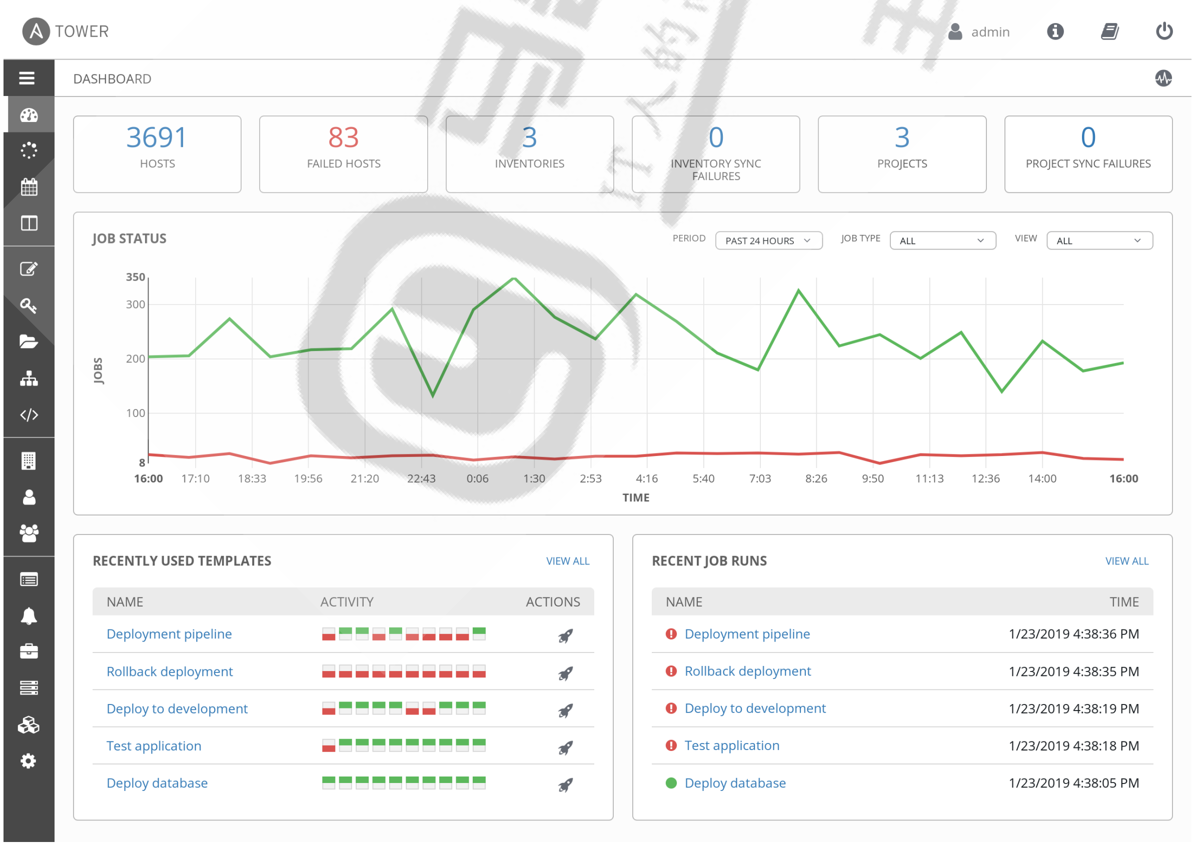
公司中实现运维自动化的架构中主要用到ansible，但是ansible脚本在部署服务器指令行中显得不太直观

Ansible Tower (最早以前叫'AWX')是能够帮助IT团队更容易使用Ansible的解决方案。

Ansible Tower是一个图形化基于WEB的任务调度，复杂服务部署，IT自动化的一个管理平台，属于发布配置管理系统，支持Api及界面操作，基于Django编写

Tower允许对用户进行权限控制，即使某用户不能传送某SSH凭证，你也可以通过Tower来对该用户共享该凭证。我们可以通过图形化界面来管理 Inventory，也可以对各种各样的云资源做同步。Tower可以记录所有job的日志，也可以与LDAP集成，并且拥有强大的可浏览的REST API。Tower也提供了命令行工具，可以与Jenkins轻松集成。Provisioning回调对自动伸缩拓扑图提供了强大的支持。

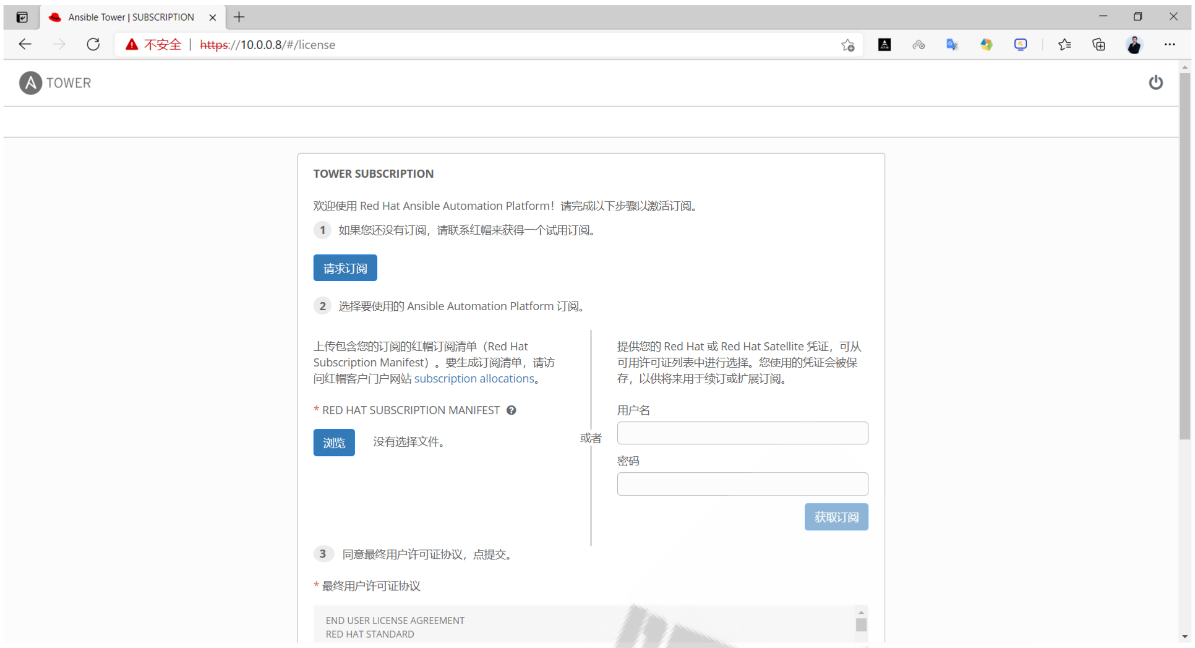
可以在`Ansible Tower 页面 <http://ansible.com/tower>`_了解Tower更多的功能，并且了解如何下载使用。Tower的免费版本最多支持10个节点，并且Ansible公司会提供强大的支持。可以支持用Ansible playbook来安装Tower



NAME	ACTIVITY	ACTIONS
Deployment pipeline	🟥🟩🟩🟥🟩🟥🟥🟥🟥🟩	
Rollback deployment	🟥🟥🟥🟥🟥🟥🟥🟥🟥🟥	
Deploy to development	🟥🟩🟩🟩🟩🟥🟥🟩🟩🟩	
Test application	🟥🟩🟩🟩🟩🟩🟩🟩🟩🟩	
Deploy database	🟩🟩🟩🟩🟩🟩🟩🟩🟩🟩	

NAME	TIME
Deployment pipeline	1/23/2019 4:38:36 PM
Rollback deployment	1/23/2019 4:38:35 PM
Deploy to development	1/23/2019 4:38:19 PM
Test application	1/23/2019 4:38:18 PM
Deploy database	1/23/2019 4:38:05 PM

安装完成后的登录初始界面,需要订阅获取license文件才能使用


Ansible Tower | SUBSCRIPTION
不安全 | https://10.0.0.8/#/license
TOWER

TOWER SUBSCRIPTION

欢迎使用 Red Hat Ansible Automation Platform！请完成以下步骤以激活订阅。

1 如果您还没有订阅，请联系红帽来获得一个试用订阅。

请求订阅

2 选择要使用的 Ansible Automation Platform 订阅。

上传包含您的订阅的红帽订阅清单（Red Hat Subscription Manifest）。要生成订阅清单，请访问红帽客户门户网站 subscription allocations。

* RED HAT SUBSCRIPTION MANIFEST

浏览 没有选择文件。

或者

提供您的 Red Hat 或 Red Hat Satellite 凭证，可从可用许可证列表中进行选择。您使用的凭证会被保存，以供将来用于续订或扩展订阅。

用户名

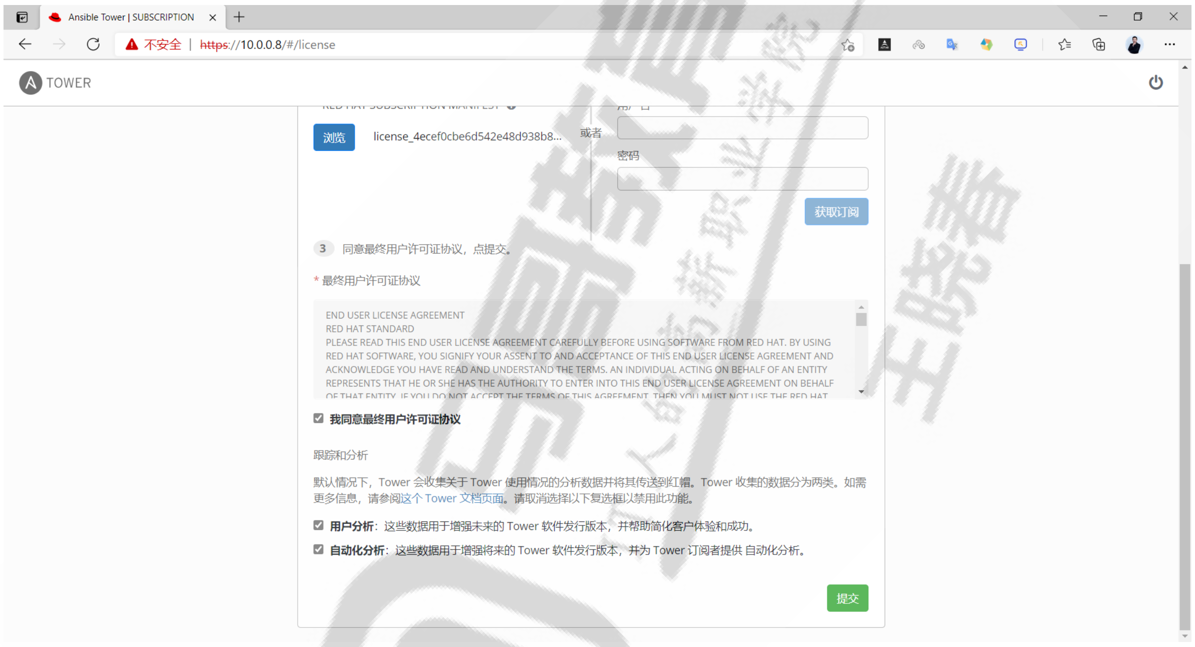
密码

获取订阅

3 同意最终用户许可证协议，点提交。

* 最终用户许可证协议

END USER LICENSE AGREEMENT
RED HAT STANDARD


Ansible Tower | SUBSCRIPTION
不安全 | https://10.0.0.8/#/license
TOWER

浏览 license_4ecef0cbe6d542e48d938b8...

或者

用户名

密码

获取订阅

3 同意最终用户许可证协议，点提交。

* 最终用户许可证协议

END USER LICENSE AGREEMENT
RED HAT STANDARD
PLEASE READ THIS END USER LICENSE AGREEMENT CAREFULLY BEFORE USING SOFTWARE FROM RED HAT. BY USING RED HAT SOFTWARE, YOU SIGNIFY YOUR ASSENT TO AND ACCEPTANCE OF THIS END USER LICENSE AGREEMENT AND ACKNOWLEDGE YOU HAVE READ AND UNDERSTAND THE TERMS. AN INDIVIDUAL ACTING ON BEHALF OF AN ENTITY REPRESENTS THAT HE OR SHE HAS THE AUTHORITY TO ENTER INTO THIS END USER LICENSE AGREEMENT ON BEHALF OF THAT ENTITY. IF YOU DO NOT ACCEPT THE TERMS OF THIS AGREEMENT, THEN YOU MUST NOT USE THE RED HAT

[✓] **我同意最终用户许可证协议**

跟踪和分析

默认情况下，Tower 会收集关于 Tower 使用情况的分析数据并将其传送到红帽。Tower 收集的数据分为两类。如需更多信息，请参阅这个 Tower 文档页面。请取消选择以下复选框以禁用此功能。

[✓] **用户分析**：这些数据用于增强未来的 Tower 软件发行版本，并帮助简化客户体验和成功。

[✓] **自动化分析**：这些数据用于增强将来的 Tower 软件发行版本，并为 Tower 订阅者提供 自动化分析。

提交

7 ansible 推荐学习资料

```
http://galaxy.ansible.com
https://galaxy.ansible.com/explore#/
http://github.com/
http://ansible.com.cn/
https://github.com/ansible/ansible
https://github.com/ansible/ansible-examples
```

祝大家学业有成

谢　谢

讲师：王晓春
邮箱：29308620@qq.com
电话：400-080-6560