



运维自动化之ANSIBLE



讲师：王晓春

本章内容



- ◆ 运维自动化发展历程及技术应用
- ◆ Ansible命令使用
- ◆ Ansible常用模块详解
- ◆ YAML语法简介
- ◆ 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

运维自动化发展历程及技术应用



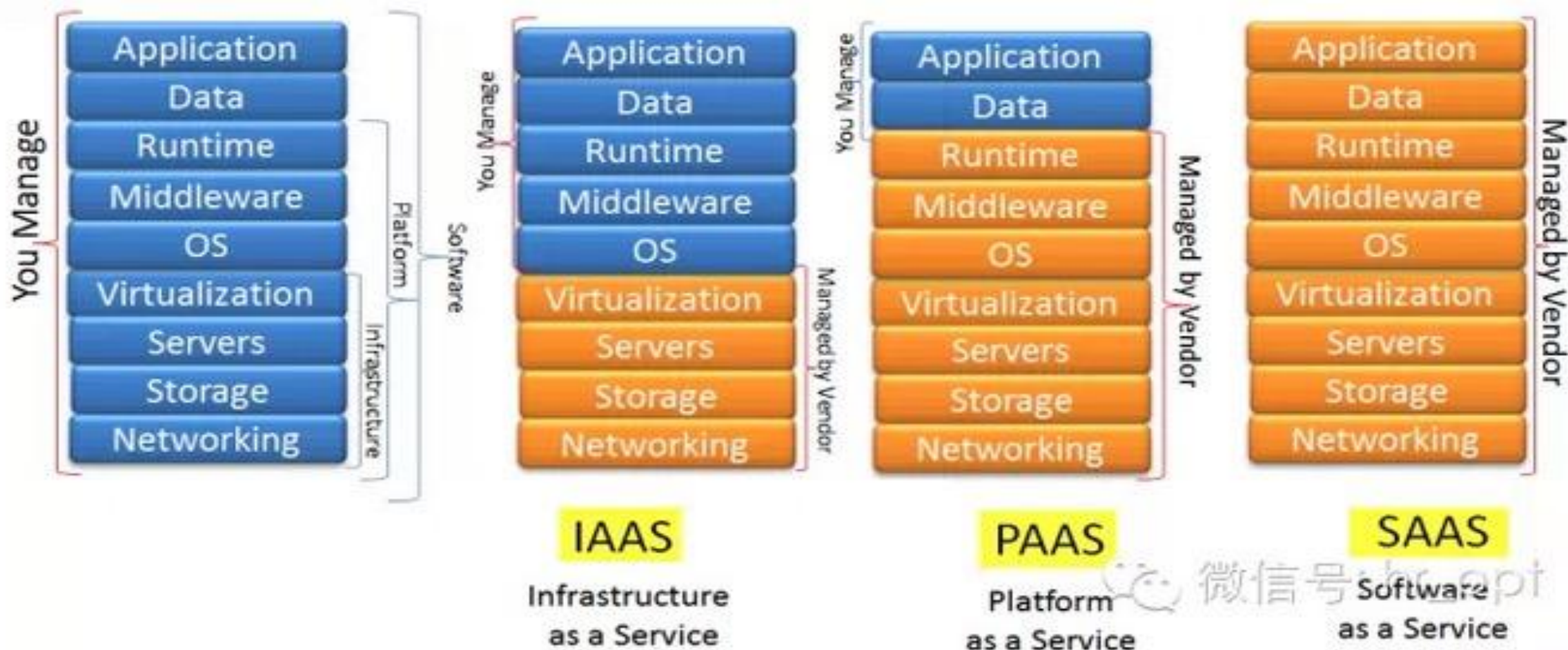
马哥教育
IT 人的高薪职业学院



运维自动化发展历程及技术应用



马哥教育
IT 人的高薪职业学院



云计算运维工程师核心职能



马哥教育
IT 人的高薪职业学院

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。

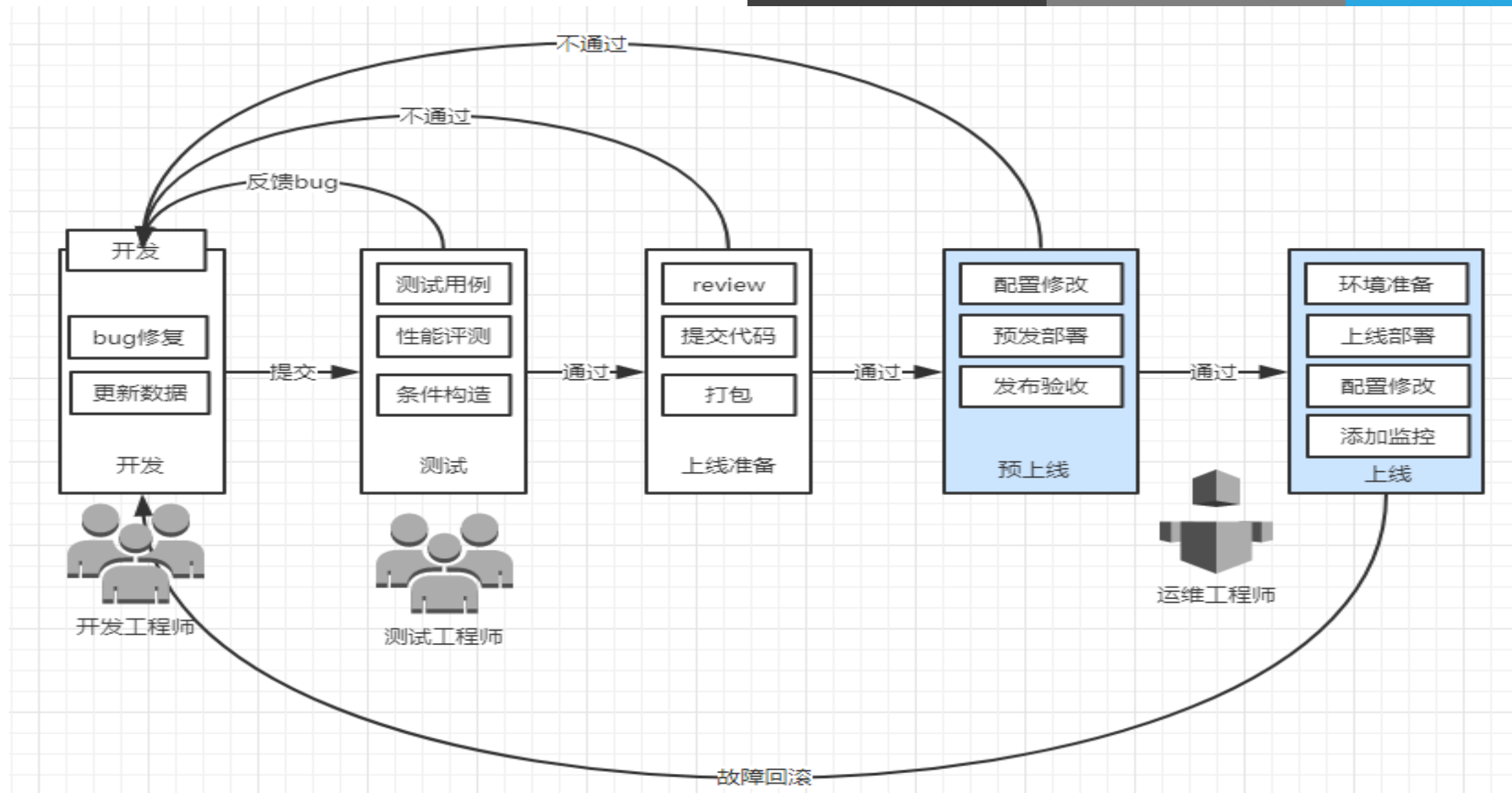
性能、效率优化

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。

Linux运维工程师职能划分



马哥教育
IT 人的高薪职业学院



◆ Dev开发环境

使用者：程序员

功能：程序员开发软件，测试BUG的环境

管理者：程序员

◆ 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

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

1、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

2、通常测试环境有多少套和产品线数量保持一样

◆ 发布环境：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）

◆ 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

◆ 灰度环境（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

- ◆ 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：在调度器上下线一批主机(标记为maintanance状态) --> 关闭服务 --> 部署新版本的应用程序 --> 启动服务 --> 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

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



- ◆ 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

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？



◆ Ansible

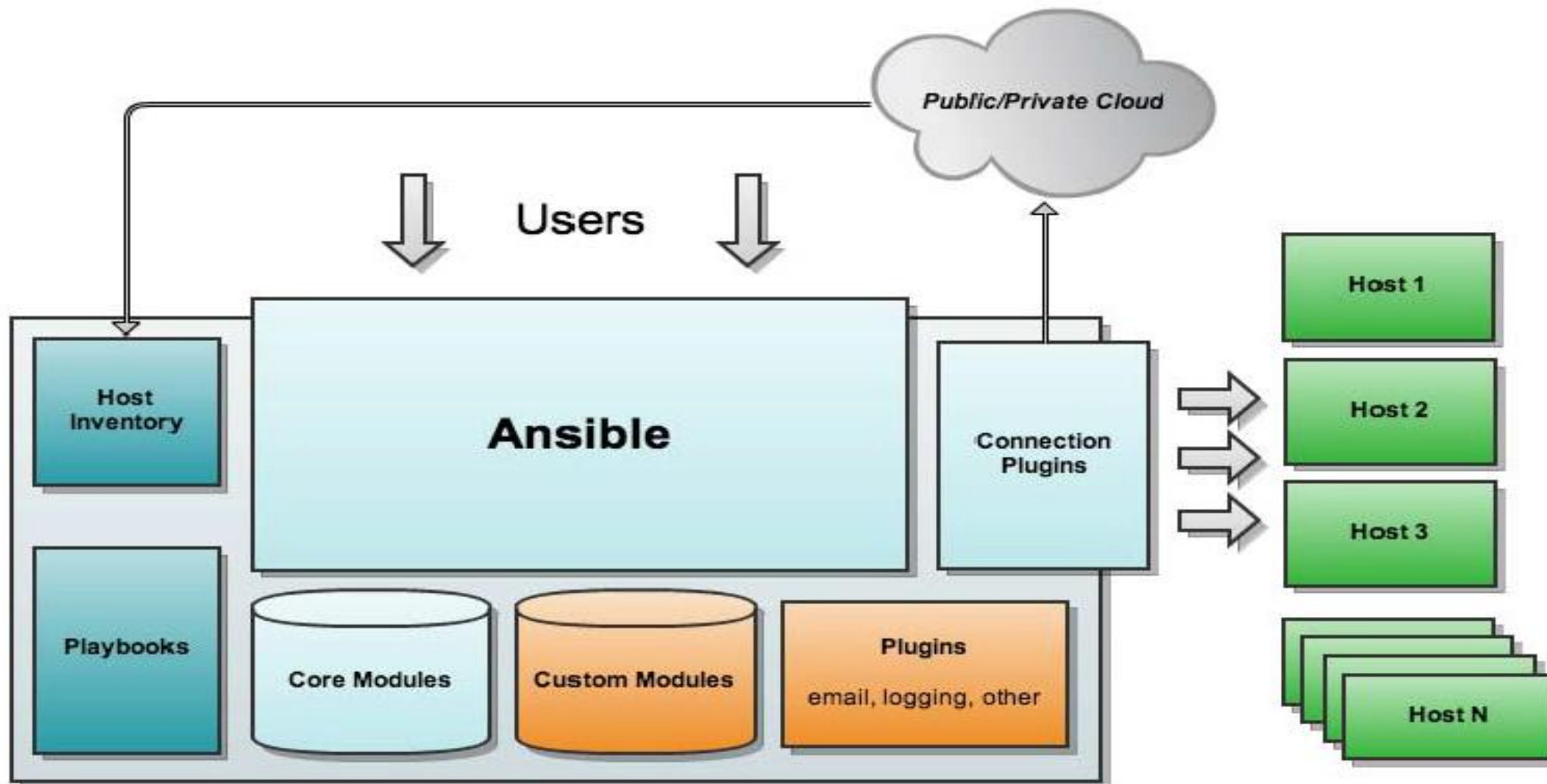
- 创始人，Michael DeHaan（Cobbler 与 Func 的作者）
- 2012-03-09，发布0.0.1版，红帽收购
- 2015-10-17，Red Hat宣布收购

◆ 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 |

- ◆ 模块化：调用特定的模块，完成特定任务
- ◆ 有Paramiko，PyYAML，Jinja2（模板语言）三个关键模块
- ◆ 支持自定义模块
- ◆ 基于Python语言实现
- ◆ 部署简单，基于python和SSH(默认已安装)，agentless
- ◆ 安全，基于OpenSSH
- ◆ 支持playbook编排任务
- ◆ 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
- ◆ 无需代理不依赖PKI（无需ssl）
- ◆ 可使用任何编程语言写模块
- ◆ 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

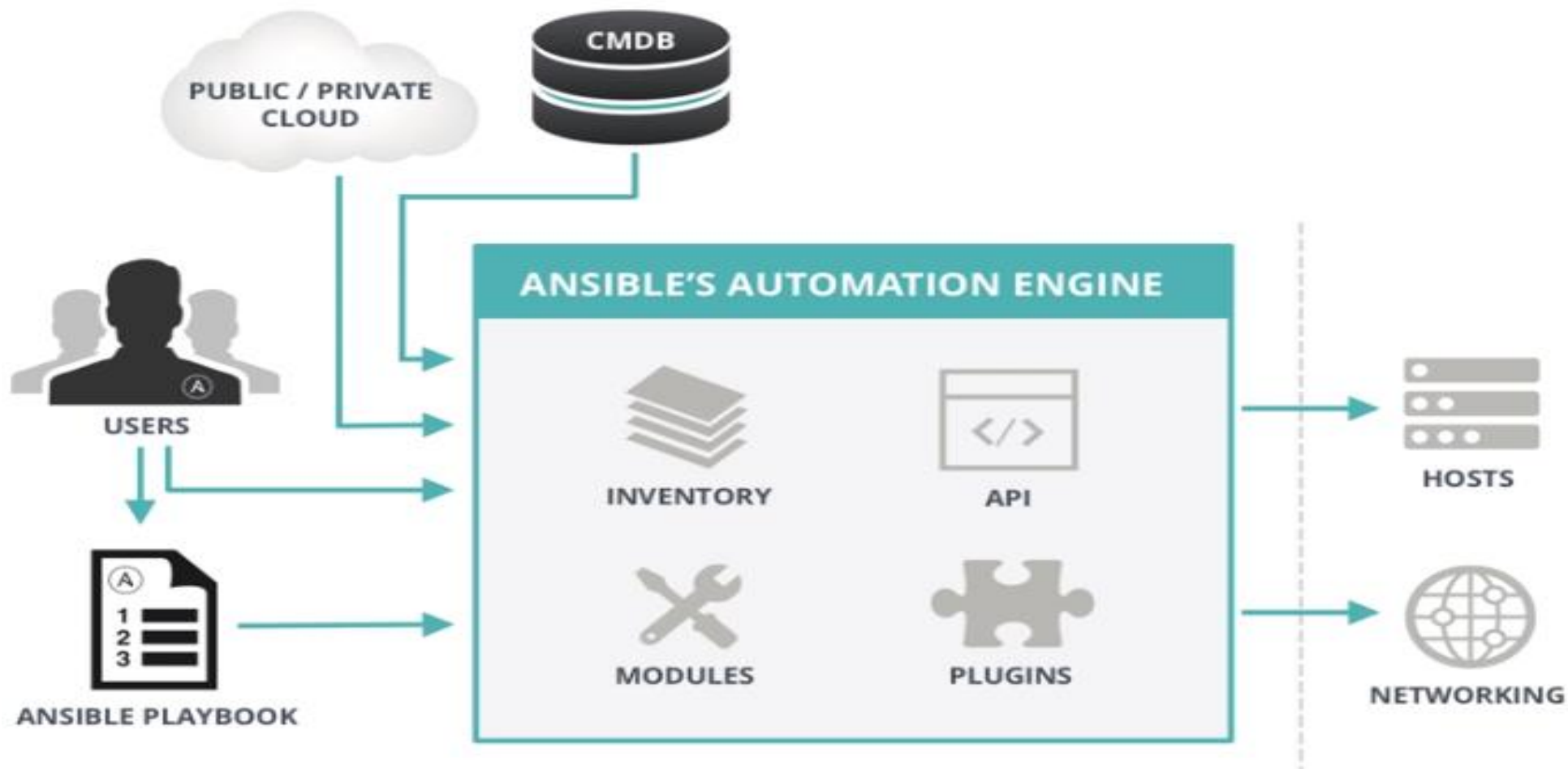
ansible架构



Ansible工作原理



马哥教育
IT 人的高薪职业学院



Ansible主要组成部分

- ◆ ANSIBLE 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
- ◆ 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
- ◆ ANSIBLE：组合INVENTORY、API、MODULES、PLUGINS的绿框，可以理解是ansible命令工具，其为核心执行工具

◆ Ansible命令执行来源：

- USER，普通用户，即SYSTEM ADMINISTRATOR
- CMDB（配置管理数据库）API 调用
- PUBLIC/PRIVATE CLOUD API调用
- USER-> Ansible Playbook -> Ansible

◆ 利用ansible实现管理的方式：

- Ad-Hoc 即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，需要有前提的规划

- ◆ Ansible-playbook (剧本) 执行过程 :
 - 将已有编排好的任务集写入Ansible-Playbook
 - 通过ansible-playbook命令分拆任务集至逐条ansible命令 , 按预定规则逐条执行
- ◆ Ansible主要操作对象 :
 - HOSTS主机
 - NETWORKING网络设备
- ◆ 注意事项
 - 执行ansible的主机一般称为主控端 , 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

◆ rpm包安装: EPEL源

```
yum install ansible
```

◆ 编译安装:

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

```
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
```

◆ Git方式:

```
git clone git://github.com/ansible/ansible.git --recursive
```

```
cd ./ansible
```

```
source ./hacking/env-setup
```

◆ pip安装： pip是安装Python包的管理器，类似yum

```
yum install python-pip python-devel
```

```
yum install gcc glibc-devel zlib-devel rpm-build openssl-devel
```

```
pip install --upgrade pip
```

```
pip install ansible --upgrade
```

◆ 确认安装： ansible --version

◆ 配置文件

/etc/ansible/ansible.cfg 主配置文件，配置ansible工作特性

/etc/ansible/hosts 主机清单

/etc/ansible/roles/ 存放角色的目录

◆ 程序

/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

/usr/bin/ansible-galaxy 下载/上传优秀代码或Roles模块的官网平台

/usr/bin/ansible-playbook 定制自动化任务，编排剧本工具
/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

◆ Inventory 主机清单

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 file中将其分组命名

◆ 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

主机清单inventory

- ◆ /etc/ansible/hosts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

ntp.magedu.com

[webservers]

www1.magedu.com:2222

www2.magedu.com

[dbservers]

db1.magedu.com

db2.magedu.com

db3.magedu.com

主机清单inventory



◆ 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

◆ 示例：

[websrvs]

www[01:100].example.com

[dbsrvs]

db-[a:f].example.com

◆ Ansible 配置文件/etc/ansible/ansible.cfg （一般保持默认）

◆ [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 , 建议取消注释
#log_path=/var/log/ansible.log #日志文件
#module_name    = command         #默认模块
```

◆ Ansible系列命令

ansible ansible-doc ansible-playbook ansible-vault
ansible-console ansible-galaxy ansible-pull

◆ ansible-doc: 显示模块帮助

ansible-doc [options] [module...]

-a 显示所有模块的文档

-l, --list 列出可用模块

-s, --snippet 显示指定模块的playbook片段

示例：

ansible-doc -l 列出所有模块

ansible-doc ping 查看指定模块帮助用法

ansible-doc -s ping 查看指定模块帮助用法

- ◆ ansible通过ssh实现配置管理、应用部署、任务执行等功能，建议配置ansible端能基于密钥认证的方式联系各被管理节点
- ◆ `ansible <host-pattern> [-m module_name] [-a args]`
 - `--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`
 - `-k, --ask-pass` 提示输入ssh连接密码，默认Key验证
 - `-K, --ask-become-pass` 提示输入sudo时的口令
 - `-C, --check` 检查，并不执行
 - `-T, --timeout=TIMEOUT` 执行命令的超时时间，默认10s
 - `-u, --user=REMOTE_USER` 执行远程执行的用户
 - `-b, --become` 代替旧版的sudo 切换
 - `--become-user=USERNAME` 指定sudo的runas用户，默认为root

ansible的Host-pattern

◆ ansible的Host-pattern

匹配主机的列表

- All : 表示所有Inventory中的所有主机

```
ansible all -m ping
```

- * :通配符

```
ansible "*" -m ping
```

```
ansible 192.168.1.* -m ping
```

```
ansible "*srvs" -m ping
```

- 或关系

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

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

ansible的Host-pattern



◆ 逻辑与

```
ansible "websrvs:&dbsrvs" -m ping
```

在websrvs组并且在dbsrvs组中的主机

◆ 逻辑非

```
ansible 'websrvs:!dbsrvs' -m ping
```

在websrvs组，但不在dbsrvs组中的主机

注意：此处为单引号

◆ 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
```

◆ 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，sleep 0退出

◆ 执行状态：

- 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

◆ 示例

- 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
```

- ◆ Command：在远程主机执行命令，默认模块，可忽略-m选项
 - `ansible srvs -m command -a 'service vsftpd start'`
 - `ansible srvs -m command -a 'echo magedu |passwd --stdin wang'`
 - 此命令不支持 `$VARNAME < > | ; &` 等，用shell模块实现
- ◆ Shell：和command相似，用shell执行命令
 - `ansible srv -m shell -a 'echo magedu |passwd -stdin wang'`
 - 调用bash执行命令 类似 `cat /tmp/stanley.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
- ◆ Script：在远程主机上运行ansible服务器上的脚本
 - `-a "/PATH/TO/SCRIPT_FILE "`
 - `ansible webservs -m script -a /data/f1.sh`

◆ Copy : 从服务器复制文件到客户端

➤ `ansible srv -m copy -a "src=/root/f1.sh dest=/tmp/f2.sh owner=wang mode=600 backup=yes"`

如目标存在，默认覆盖，此处指定先备份

➤ `ansible srv -m copy -a "content= 'test content\n' dest=/tmp/f1.txt"`
指定内容，直接生成目标文件

◆ Fetch : 从客户端取文件至服务器端，copy相反，目录可先tar

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

◆ File : 设置文件属性

➤ `ansible srv -m file -a "path=/root/a.sh owner=wang mode=755 "`

➤ `ansible web -m file -a 'src=/app/testfile dest=/app/testfile-link state=link'`

◆ Hostname : 管理主机名

➤ `ansible node1 -m hostname -a "name=websrv"`

◆ Cron : 计划任务

支持时间 : minute , hour , day , month , weekday

➤ `ansible srv -m cron -a "minute=*/5 job= '/usr/sbin/ntpdate 172.16.0.1 &>/dev/null' name=Synctime"` 创建任务

➤ `ansible srv -m cron -a 'state=absent name=Synctime'` 删除任务

◆ Yum : 管理包

➤ `ansible srv -m yum -a 'name=httpd state=present'` 安装

➤ `ansible srv -m yum -a 'name=httpd state=absent'` 删除

◆ Service : 管理服务

- `ansible srv -m service -a 'name=httpd state=stopped'`
- `ansible srv -m service -a 'name=httpd state=started enabled=yes'`
- `ansible srv -m service -a 'name=httpd state=reloaded'`
- `ansible srv -m service -a 'name=httpd state=restarted'`

◆ User : 管理用户

- `ansible srv -m user -a 'name=user1 comment="test user" uid=2048 home=/app/user1 group=root '`
- `ansible srv -m user -a 'name=sysuser1 system=yes home=/app/sysuser1 '`
- `ansible srv -m user -a 'name=user1 state=absent remove=yes '`
删除用户及家目录等数据

◆ Group : 管理组

- `ansible srv -m group -a "name=testgroup system=yes "`
- `ansible srv -m group -a "name=testgroup state=absent"`

◆ ansible-galaxy

- 连接 <https://galaxy.ansible.com> 下载相应的roles
- 列出所有已安装的galaxy
 - ansible-galaxy list
- 安装galaxy
 - ansible-galaxy install geerlingguy.redis
- 删除galaxy
 - ansible-galaxy remove geerlingguy.redis

◆ ansible-pull

推送命令至远程，效率无限提升，对运维要求较高

◆ ansible-playbook

执行playbook

示例：ansible-playbook hello.yml

```
cat hello.yml
```

```
#hello world yml file
```

```
- hosts: webservs
```

```
  remote_user: root
```

```
  tasks:
```

```
    - name: hello world
```

```
      command: /usr/bin/wall hello world
```

◆ ansible-vault

- 功能：管理加密解密yaml文件
- `ansible-vault [create|decrypt|edit|encrypt|rekey|view]`
- `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

◆ Ansible-console : 2.0+新增, 可交互执行命令, 支持tab

➤ root@test (2)[f:10] \$

执行用户@当前操作的主机组 (当前组的主机数量)[f:并发数]\$

➤ 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

➤ 示例:

```
root@all (2)[f:5]$ list
```

```
root@all (2)[f:5]$ cd appsrvs
```

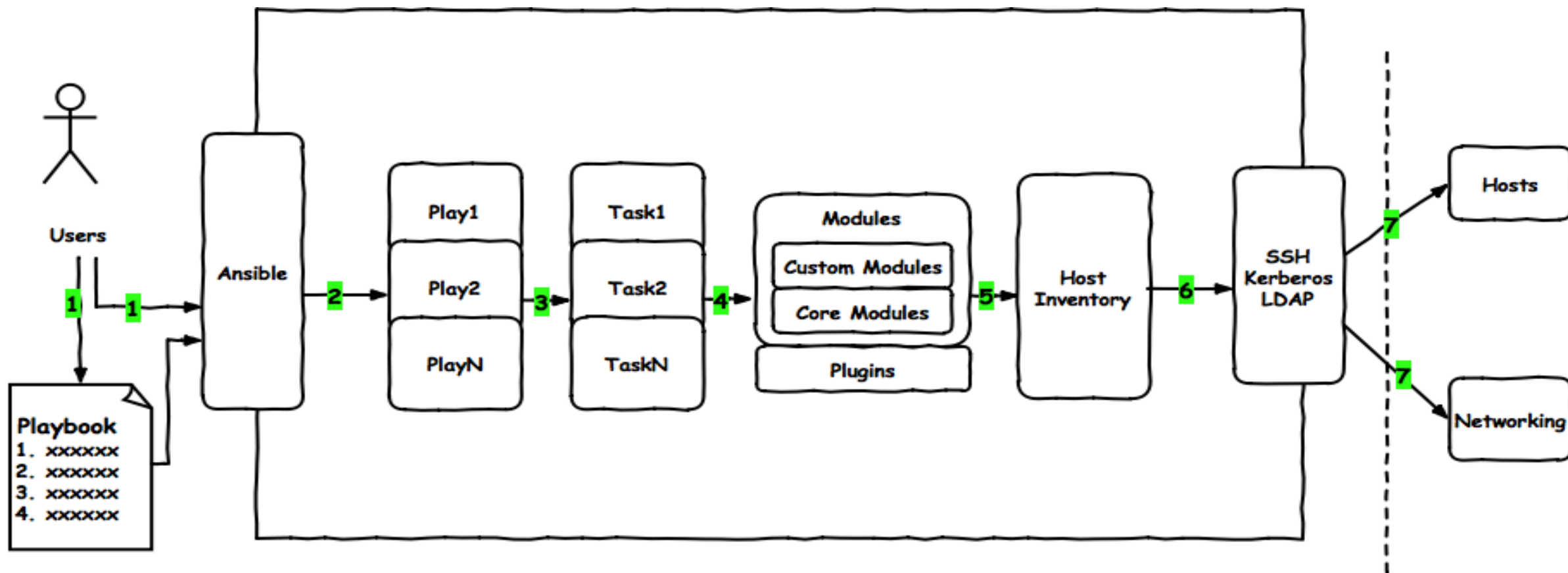
```
root@appsrvs (2)[f:5]$ list
```

```
root@appsrvs (2)[f:5]$ yum name=httpd state=present
```

```
root@appsrvs (2)[f:5]$ service name=httpd state=started
```

- ◆ playbook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采用YAML语言编写

playbook



- ◆ 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
- ◆ YAML Ain't Markup Language，即YAML不是XML。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）
- ◆ 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
- ◆ 更多的内容及规范参见<http://www.yaml.org>

- ◆ 在单一档案中，可用连续三个连字号(——)区分多个档案。另外，还有选择性的连续三个点号(...)用来表示档案结尾
- ◆ 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，k/v的值均需大小写敏感
- ◆ k/v的值可同行写也可换行写。同行使用:分隔
- ◆ v可是个字符串，也可是另一个列表
- ◆ 一个完整的代码块功能需最少元素需包括 name: task
- ◆ 一个name只能包括一个task
- ◆ YAML文件扩展名通常为yml或yaml

◆ List：列表，其所有元素均使用“-”打头

◆ 示例：

```
# A list of tasty fruits
```

- Apple
- Orange
- Strawberry
- Mango

◆ Dictionary：字典，通常由多个key与value构成

◆ 示例：

An employee record

name: Example Developer

job: Developer

skill: Elite

也可以将key:value放置于{}中进行表示，用,分隔多个key:value

◆ 示例：

An employee record

{name: Example Developer, job: Developer, skill: Elite}

- ◆ YAML的语法和其他高阶语言类似，并且可以简单表达清单、散列表、标量等数据结构。其结构（ Structure ）通过空格来展示，序列（ Sequence ）里的项用"-"来代表，Map里的键值对用":"分隔

- ◆ 示例

```
name: John Smith
```

```
age: 41
```

```
gender: Male
```

```
spouse:
```

```
  name: Jane Smith
```

```
  age: 37
```

```
  gender: Female
```

```
children:
```

```
  - name: Jimmy Smith
```

```
    age: 17
```

```
    gender: Male
```

```
  - name: Jenny Smith
```

```
    age 13
```

```
    gender: Female
```

- ◆ Hosts 执行的远程主机列表
- ◆ Tasks 任务集
- ◆ Varniables 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notity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

```
ansible-playbook -t tagname useradd.yml
```

◆ 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

➤ 可以是如下形式：

one.example.com

one.example.com:two.example.com

192.168.1.50

192.168.1.*

➤ Webservs:dbsrvs 或者，两个组的并集

➤ Webservs:&dbsrvs 与，两个组的交集

➤ webserver:!phoenix 在websrvs组，但不在dbsrvs组

示例: - hosts: websrvs : dbsrvs

- ◆ `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: webservs`

- `remote_user: root`

- `tasks:`

- `name: test connection`

- `ping:`

- `remote_user: magedu`

- `sudo: yes`

- `sudo_user:wang`

默认sudo为root

sudo为wang

◆ task列表和action

- play的主体部分是task list。task list中的各任务按次序逐个在hosts中指定的所有主机上执行，即在所有主机上完成第一个任务后，再开始第二个任务
- 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

- ◆ tasks : 任务列表

- ◆ 两种格式 :

 - (1) action: module arguments

 - (2) module: arguments 建议使用

 - 注意 : shell和command模块后面跟命令 , 而非key=value

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

- ◆ 任务可以通过“tags”打标签 , 可在ansible-playbook命令上使用-t指定进行调用
示例 :

 - tasks:

 - name: disable selinux

 - command: /sbin/setenforce 0

- ◆ 如果命令或脚本的退出码不为零，可以使用如下方式替代

tasks:

- name: run this command and ignore the result
shell: /usr/bin/somecommand || /bin/true

- ◆ 或者使用ignore_errors来忽略错误信息

tasks:

- name: run this command and ignore the result
shell: /usr/bin/somecommand
ignore_errors: True

运行playbook



◆ 运行playbook的方式

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

◆ 常见选项

| | |
|---------------------------|----------------------|
| <code>--check -C</code> | 只检测可能会发生的改变，但不真正执行操作 |
| <code>--list-hosts</code> | 列出运行任务的主机 |
| <code>--list-tags</code> | 列出tag |
| <code>--list-tasks</code> | 列出task |
| <code>--limit 主机列表</code> | 只针对主机列表中的主机执行 |
| <code>-v -vv -vvv</code> | 显示过程 |

◆ 示例

```
ansible-playbook file.yml --check 只检测
ansible-playbook file.yml
ansible-playbook file.yml --limit webservs
```

Playbook VS ShellScripts

◆ SHELL脚本

```
#!/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
service httpd start
chkconfig httpd on
```

◆ Playbook定义

```
- hosts: all
  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
```

示例

示例：sysuser.yml

- hosts: all
- remote_user: root

tasks:

- name: create mysql user
 - user: name=mysql system=yes uid=36
- name: create a group
 - group: name=httpd system=yes

Playbook示例

示例 : httpd.yml

- hosts: webservs
remote_user: root

tasks:

- name: Install httpd
yum: name=httpd state=present
- name: Install configure file
copy: src=files/httpd.conf dest=/etc/httpd/conf/
- name: start service
service: name=httpd state=started enabled=yes

handlers和notify结合使用触发条件

◆ Handlers

是task列表，这些task与前述的task并没有本质上的不同,用于当关注的资源发生变化时，才会采取一定的操作

- ◆ Notify此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

Playbook中handlers使用

- hosts: webservs
remote_user: root
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
 - name: ensure apache is running
service: name=httpd state=started enabled=yes
- handlers:
- name: restart httpd
service: name=httpd status=restarted

示例



```
- hosts: webservs
  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: config
copy: src=/root/config.txt dest=/etc/nginx/nginx.conf

notify:

- 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

Playbook中tags使用

示例：httpd.yml

- hosts: webservs
remote_user: root
tasks:
 - name: Install httpd
yum: name=httpd state=present
 - name: Install configure file
copy: src=files/httpd.conf dest=/etc/httpd/conf/
tags: conf
 - name: start httpd service
tags: service
service: name=httpd state=started enabled=yes

ansible-playbook -t conf httpd.yml

Playbook中变量使用



- ◆ 变量名：仅能由字母、数字和下划线组成，且只能以字母开头
- ◆ 变量来源：
 - 1 ansible setup facts 远程主机的所有变量都可直接调用
 - 2 在/etc/ansible/hosts中定义
 - 普通变量：主机组中主机单独定义，优先级高于公共变量
 - 公共（组）变量：针对主机组中所有主机定义统一变量
 - 3 通过命令行指定变量，优先级最高
 - `ansible-playbook -e varname=value`
 - 4 在playbook中定义
 - vars:
 - var1: value1
 - var2: value2
 - 5 在独立的变量YAML文件中定义
 - 6 在role中定义

◆ 变量命名

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：key=value

示例：http_port=80

◆ 变量调用方式：

- 通过{{ variable_name }} 调用变量，且变量名前后必须有空格，有时用“{{ variable_name }}" 才生效
- ansible-playbook -e 选项指定
ansible-playbook test.yml -e "hosts=www user=magedu"

示例：使用setup变量

示例：var.yml

```
- hosts: webservs
  remote_user: root
```

tasks:

```
- name: create log file
  file: name=/var/log/ {{ ansible_fqdn }} state=touch
```

```
ansible-playbook var.yml
```

示例：变量



示例：var.yml

- hosts: webservs

remote_user: root

tasks:

- name: install package

- yum: name={{ pkname }} state=present

ansible-playbook -e pkname=httpd var.yml

示例：变量

示例：var.yml

- hosts: webservs

remote_user: root

vars:

- username: user1

- groupname: group1

tasks:

- name: create group

- group: name={{ groupname }} state=present

- name: create user

- user: name={{ username }} state=present

ansible-playbook var.yml

ansible-playbook -e "username=user2 groupname=group2" var2.yml

◆ 主机变量

可以在inventory中定义主机时为其添加主机变量以便于在playbook中使用

◆ 示例：

[webservs]

www1.magedu.com http_port=80 maxRequestsPerChild=808

www2.magedu.com http_port=8080 maxRequestsPerChild=909

◆ 组变量

组变量是指赋予给指定组内所有主机上的在playbook中可用的变量

◆ 示例：

```
[webservs]
```

```
www1.magedu.com
```

```
www2.magedu.com
```

```
[webservs:vars]
```

```
ntp_server=ntp.magedu.com
```

```
nfs_server=nfs.magedu.com
```

示例：变量

◆ 普通变量

```
[webservs]
```

```
192.168.99.101 http_port=8080 hname=www1
```

```
192.168.99.102 http_port=80 hname=www2
```

◆ 公共（组）变量

```
[webservs:vars]
```

```
http_port=808
```

```
mark= "_"
```

```
[webservs]
```

```
192.168.99.101 http_port=8080 hname=www1
```

```
192.168.99.102 http_port=80 hname=www2
```

```
ansible webservs -m hostname -a 'name={{ hname }}{{ mark }}{{ http_port }}'
```

◆ 命令行指定变量：

```
ansible webservs -e http_port=8000 -m hostname -a
```

```
'name={{ hname }}{{ mark }}{{ http_port }}'
```

使用变量文件

- ◆ cat vars.yml
 - var1: httpd
 - var2: nginx

- ◆ cat var.yml
 - hosts: web
 - remote_user: root
 - vars_files:
 - vars.yml
 - tasks:
 - name: create httpd log
 - file: name=/app/{{ var1 }}.log state=touch
 - name: create nginx log
 - file: name=/app/{{ var2 }}.log state=touch

- ◆ 文本文件，嵌套有脚本（使用模板编程语言编写）
- ◆ Jinja2语言，使用字面量，有下面形式
 - 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

templates



- ◆ templates功能：根据模块文件动态生成对应的配置文件
 - templates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目录同级，目录结构如下：

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

Templates示例



◆ 示例：利用templates 同步nginx配置文件

准备templates/nginx.conf.j2文件

```
vim temnginx.yml
```

```
- hosts: webservs
  remote_user: root
```

```
tasks:
```

```
- name: template config to remote hosts
```

```
  template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf
```

```
ansible-playbook temnginx.yml
```

Playbook中template变更替换

- ◆ 修改文件nginx.conf.j2 下面行为

```
worker_processes {{ ansible_processor_vcpus }};
```

- ◆ cat temnginx2.yml

```
- hosts: webservs
  remote_user: root
```

```
tasks:
```

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

```
ansible-playbook temnginx2.yml
```

Playbook中template算术运算

◆ 算法运算：

◆ 示例：

```
vim nginx.conf.j2
```

```
worker_processes {{ ansible_processor_vcpus**2 }};
```

```
worker_processes {{ ansible_processor_vcpus+2 }};
```


- ◆ 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when语句实现，在task中使用，jinja2的语法格式
- ◆ when语句
在task后添加when子句即可使用条件测试；when语句支持Jinja2表达式语法
- ◆ 示例：
tasks:
 - name: "shutdown RedHat flavored systems"
command: /sbin/shutdown -h now
when: ansible_os_family == "RedHat"

示例：when条件判断



```
- hosts: webservs
  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"
```

示例：when条件判断

◆ 示例：

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"

迭代：with_items



- ◆ 迭代：当有需要重复性执行的任务时，可以使用迭代机制
 - 对迭代项的引用，固定变量名为“item”
 - 要在task中使用with_items给定要迭代的元素列表
 - 列表格式：
 - 字符串
 - 字典

◆ 示例：

- name: add several users

- user: name={{ item }} state=present groups=wheel

- with_items:

- testuser1

- testuser2

◆ 上面语句的功能等同于下面的语句：

- name: add user testuser1

- user: name=testuser1 state=present groups=wheel

- name: add user testuser2

- user: name=testuser2 state=present groups=wheel

示例：迭代

◆ 示例：将多个文件进行copy到被控端

- hosts: testsrv

remote_user: root

tasks

- name: Create rsyncd config

copy: src={{ item }} dest=/etc/{{ item }}

with_items:

- rsyncd.secrets

- rsyncd.conf

示例：迭代



```
- hosts: webservs
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
```

示例：迭代

- hosts : webservs
remote_user: root

tasks

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

示例：迭代嵌套子变量

```
- hosts : webservs
remote_user: root
tasks:
  - name: add some groups
    group: name={{ item }} state=present
    with_items:
      - group1
      - group2
      - group3
  - name: add some users
    user: name={{ item.name }} group={{ item.group }} state=present
    with_items:
      - { name: 'user1', group: 'group1' }
      - { name: 'user2', group: 'group2' }
      - { name: 'user3', group: 'group3' }
```

Playbook中template for if



马哥教育

IT 人的高薪职业学院

```
{% for vhost in nginx_vhosts %}  
server {  
listen {{ vhost.listen | default('80 default_server') }};
```

```
{% if vhost.server_name is defined %}  
server_name {{ vhost.server_name }};  
{% endif %}
```

```
{% if vhost.root is defined %}  
root {{ vhost.root }};  
{% endif %}
```

示例



// temnginx.yml

```
- hosts: testweb
  remote_user: root
  vars:
    nginx_vhosts:
      - listen: 8080
```

//templates/nginx.conf.j2

```
{% for vhost in nginx_vhosts %}
server {
    listen {{ vhost.listen }}
}
{% endfor %}
```

生成的结果

```
server {
    listen 8080
}
```

示例



// temnginx.yml

- hosts: mageduweb

remote_user: root

vars:

nginx_vhosts:

- web1

- web2

- web3

tasks:

- name: template config

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

// templates/nginx.conf.j2

{% for vhost in
nginx_vhosts %}

server {

listen {{ vhost }}

}

{% endfor %}

生成的结果：

server {

listen web1

}

server {

listen web2

}

server {

listen web3

}

示例

// temnginx.yml

```
- hosts: mageduweb
  remote_user: root
  vars:
    nginx_vhosts:
      - web1:
          listen: 8080
          server_name: "web1.magedu.com"
          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
      template: src=nginx.conf.j2 dest=/etc/nginx/nginx.conf
```



// templates/nginx.conf.j2

```
{% for vhost in nginx_vhosts %}
server {
    listen {{ vhost.listen }}
    server_name {{ vhost.server_name }}
    root {{ vhost.root }}
}
```

{% endfor %}

生成结果：

```
server {
    listen 8080
    server_name web1.magedu.com
    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/
}
```

示例



// temnginx.yml

```
- hosts: mageduweb
  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.conf.j2 dest=/etc/nginx/nginx.conf
```

// templates/nginx.conf.j2

```
{% for vhost in nginx_vhosts %}
server {
    listen {{ vhost.listen }}
    {% if vhost.server_name is defined %}
    server_name {{ vhost.server_name }}
    {% endif %}
    root {{ vhost.root }}
}
{% 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/
}
```

◆ roles

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

◆ 复杂场景：建议使用roles，代码复用度高

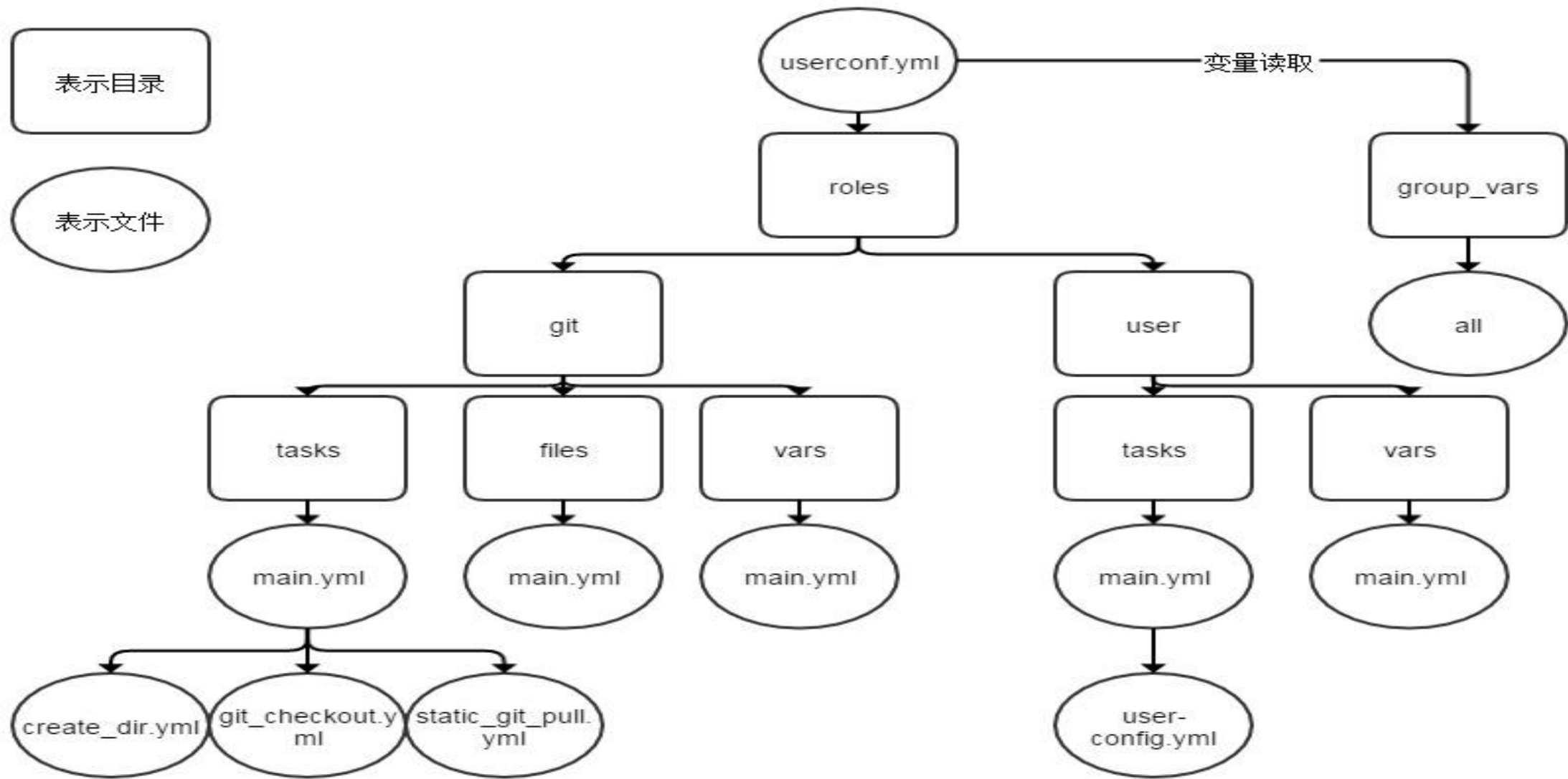
- 变更指定主机或主机组
- 如命名不规范维护和传承成本大
- 某些功能需多个Playbook，通过includes即可实现

Roles



- ◆ 角色(roles) : 角色集合
 - roles/
 - mysql/
 - httpd/
 - nginx/
 - memcached/

Ansible Roles目录编排



roles目录结构



- ◆ 每个角色，以特定的层级目录结构进行组织

- ◆ roles目录结构：

 - playbook.yml

 - roles/

 - project/

 - tasks/

 - files/

 - vars/

 - templates/

 - handlers/

 - default/ 不常用

 - meta/ 不常用

- ◆ /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; 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; 其它的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

- ◆ 创建role的步骤
- ◆ (1) 创建以roles命名的目录
- ◆ (2) 在roles目录中分别创建以各角色名称命名的目录，如webservers等
- ◆ (3) 在每个角色命名的目录中分别创建files、handlers、meta、tasks、templates和vars目录；用不到的目录可以创建为空目录，也可以不创建
- ◆ (4) 在playbook文件中，调用各角色

针对大型项目使用Roles进行编排



roles目录结构：

playbook.yml

roles/

project/

tasks/

files/

vars/

templates/

handlers/

default/ # 不经常用

meta/ # 不经常用

示例：

nginx-role.yml

roles/

└─ nginx

├─ files

│ └─ main.yml

├─ tasks

│ └─ groupadd.yml

│ └─ install.yml

│ └─ main.yml

│ └─ restart.yml

│ └─ useradd.yml

└─ vars

└─ main.yml

示例



◆ roles的示例如下所示：

site.yml

webservers.yml

dbservers.yml

roles/

common/

files/

templates/

tasks/

handlers/

vars/

meta/

webservers/

files/

templates/

tasks/

handlers/

vars/

meta/

playbook调用角色

◆ 调用角色方法1：

- hosts: webservs

remote_user: root

roles:

- mysql
- memcached
- nginx

◆ 调用角色方法2：

传递变量给角色

- hosts:

remote_user:

roles:

- mysql

- { role: nginx, username: nginx }

键role用于指定角色名称

后续的k/v用于传递变量给角色

◆ 调用角色方法3：还可基于条件测试实现角色调用

roles:

- { role: nginx, username: nginx, when: ansible_distribution_major_version == '7' }

完整的roles架构



马哥教育

IT 人的高薪职业学院

```
// nginx-role.yml 顶层任务调用yaml文件
```

```
---
```

```
- hosts: testweb
  remote_user: root
```

```
roles:
```

- role: nginx
- role: httpd 可执行多个role

```
cat roles/nginx/tasks/main.yml
```

```
---
```

- include: groupadd.yml
- include: useradd.yml
- include: install.yml
- include: restart.yml
- include: filecp.yml

```
// roles/nginx/tasks/groupadd.yml
```

```
---
```

- name: add group nginx
 user: name=nginx state=present

```
cat roles/nginx/tasks/filecp.yml
```

```
---
```

- name: file copy
 copy: src=tom.conf dest=/tmp/tom.conf

以下文件格式类似：

useradd.yml,install.yml,restart.yml

```
ls roles/nginx/files/
tom.conf
```

roles playbook tags使用

◆ roles playbook tags使用

```
ansible-playbook --tags="nginx,httpd,mysql" nginx-role.yml
```

```
// nginx-role.yml
```

```
---
```

```
- hosts: testweb
```

```
  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: marridb ,tags: ['mysql', 'db'] }
- { role: php }

- ◆ <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>

- ◆ 博客 : <http://mageedu.blog.51cto.com>
- ◆ 主页 : <http://www.magedu.com>
- ◆ QQ : 1661815153, 113228115
- ◆ QQ群 : 203585050, 279599283

祝大家学业有成

谢 谢

咨询热线 400-080-6560