

# 自动化运维工具Ansible详细部署（转）



帅T

关注



0.239

2017.03.27 10:17:39 字数 3,433 阅读 1,122

## 一、基础介绍

=====

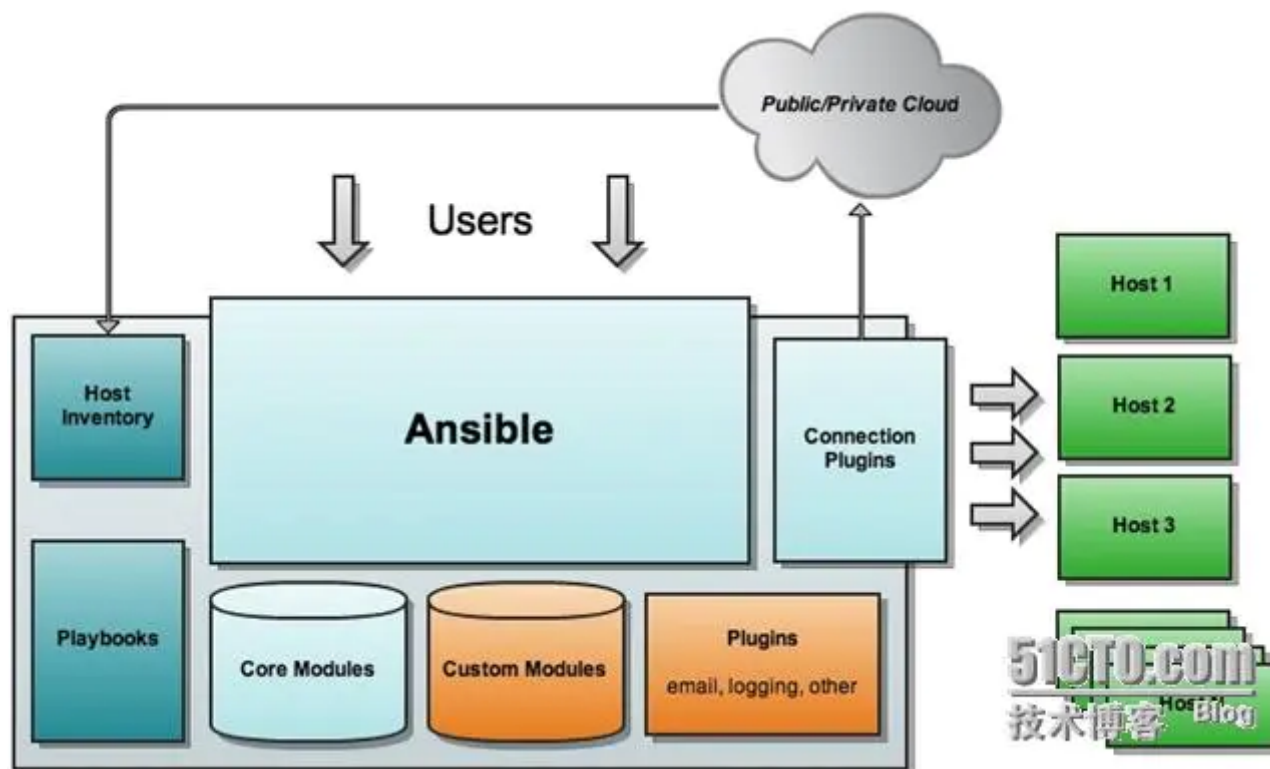
### 1、简介

ansible是新出现的自动化运维工具，基于Python开发，集合了众多运维工具（puppet、cfengine、chef、func、fabric）的优点，实现了批量系统配置、批量程序部署、批量运行命令等功能。ansible是基于模块工作的，本身没有批量部署的能力。真正具有批量部署的是ansible所运行的模块，ansible只是提供一种框架。主要包括：

- (1)、连接插件connection plugins：负责和被监控端实现通信；
- (2)、host inventory：指定操作的主机，是一个配置文件里面定义监控的主机；
- (3)、各种模块核心模块、command模块、自定义模块；
- (4)、借助于插件完成记录日志邮件等功能；

(5)、playbook：剧本执行多个任务时，非必需可以让节点一次性运行多个任务。

## 2、总体架构



## 3、特性

(1)、no agents：不需要在被管控主机上安装任何客户端；

(2)、no server：无服务器端，使用时直接运行命令即可；

(3)、modules in any languages：基于模块工作，可使用任意语言开发模块；

(4)、yaml, not code: 使用yaml语言定制剧本playbook;

(5)、ssh by default: 基于SSH工作;

(6)、strong multi-tier solution: 可实现多级指挥。

#### **4、优点**

(1)、轻量级, 无需在客户端安装agent, 更新时, 只需在操作机上进行一次更新即可;

(2)、批量任务执行可以写成脚本, 而且不用分发到远程就可以执行;

(3)、使用python编写, 维护更简单, ruby语法过于复杂;

(4)、支持sudo。

#### **5、任务执行流程**



说明：

(1)、以上内容大多是基于他人分享的基础上总结而来，学习借鉴之用；

(2)、本次安装基于 CentOS 6.4系统环境。

=====

=====

## 二、Ansible基础安装与配置

=====

=====

### 1、Ansible基础安装

#### (1)、python2.7安装

<https://www.python.org/ftp/python/2.7.8/Python-2.7.8.tgz>

```
# tar xvf Python-2.7.8.tgz
```

```
# cd Python-2.7.8
```

```
# ./configure --prefix=/usr/local
```

```
# make --jobs=`grep processor/proc/cpuinfo | wc -l`
```

```
# make install
```

```
##将python头文件拷贝到标准目录，以避免编译ansible时，找不到所需的头文件
```

```
# cd /usr/local/include/python2.7
```

```
# cp -a ./* /usr/local/include/
```

```
##备份旧版本的python，并符号链接新版本的python
```

```
# cd /usr/bin
```

```
# mv python python2.6
```

```
# ln -s /usr/local/bin/python
```

```
##修改yum脚本，使其指向旧版本的python，已避免其无法运行
```

```
# vim /usr/bin/yum
```

```
#!/usr/bin/python --> #!/usr/bin/python2.6
```

(2)、setuptools**模块安装**

<https://pypi.python.org/packages/source/s/setuptools/setuptools-7.0.tar.gz>

```
# tar xvzf setuptools-7.0.tar.gz
```

```
# cd setuptools-7.0
```

```
# python setup.py install
```

### (3)、pycrypto模块安装

<https://pypi.python.org/packages/source/p/pycrypto/pycrypto-2.6.1.tar.gz>

```
# tar xvzf pycrypto-2.6.1.tar.gz
```

```
# cd pycrypto-2.6.1
```

```
# python setup.py install
```

### (4)、PyYAML模块安装

<http://pyyaml.org/download/libyaml/yaml-0.1.5.tar.gz>

```
# tar xvzf yaml-0.1.5.tar.gz
```

```
# cd yaml-0.1.5
```

```
# ./configure --prefix=/usr/local
```

```
# make --jobs=`grep processor/proc/cpuinfo | wc -l`
```

```
# make install
```

<https://pypi.python.org/packages/source/P/PyYAML/PyYAML-3.11.tar.gz>

```
# tar xvzf PyYAML-3.11.tar.gz
```

```
# cd PyYAML-3.11
```

```
# python setup.py install
```

#### (5)、Jinja2**模块安装**

<https://pypi.python.org/packages/source/M/MarkupSafe/MarkupSafe-0.9.3.tar.gz>

```
# tar xvzf MarkupSafe-0.9.3.tar.gz
```

```
# cd MarkupSafe-0.9.3
```

```
# python setup.py install
```

<https://pypi.python.org/packages/source/J/Jinja2/Jinja2-2.7.3.tar.gz>

```
# tar xvzf Jinja2-2.7.3.tar.gz
```

```
# cd Jinja2-2.7.3
```

```
# python setup.py install
```

#### (6)、paramiko**模块安装**

<https://pypi.python.org/packages/source/e/ecdsa/ecdsa-0.11.tar.gz>

```
# tar xvzf ecdsa-0.11.tar.gz
```

```
# cd ecdsa-0.11
```

```
# python setup.py install
```

<https://pypi.python.org/packages/source/p/paramiko/paramiko-1.15.1.tar.gz>

```
# tar xvzf paramiko-1.15.1.tar.gz
```

```
# cd paramiko-1.15.1
```

```
# python setup.py install
```

(7)、simplejson**模块安装**

<https://pypi.python.org/packages/source/s/simplejson/simplejson-3.6.5.tar.gz>

```
# tar xvzf simplejson-3.6.5.tar.gz
```

```
# cd simplejson-3.6.5
```

```
# python setup.py install
```

(8)、ansible**安装**

<https://github.com/ansible/ansible/archive/v1.7.2.tar.gz>



```
# tar xvzf ansible-1.7.2.tar.gz
```

```
# cd ansible-1.7.2
```

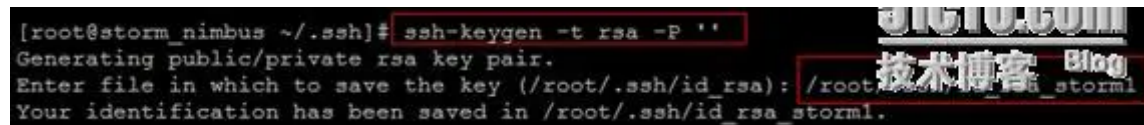
```
# python setup.py install
```

## 2、Ansible配置

### (1)、SSH免密钥登录设置

##生成公钥/私钥

```
# ssh-keygen -t rsa -P ''
```



```
[root@storm_nimbus ~/.ssh]# ssh-keygen -t rsa -P ''
Generating public/private rsa key pair.
Enter file in which to save the key (/root/.ssh/id_rsa): /root/.ssh/id_rsa_storm1
Your identification has been saved in /root/.ssh/id_rsa_storm1.
```

##写入信任文件（将/root/.ssh/id\_rsa\_storm1.pub分发到其他服务器，并在所有服务器上执行如下指令）：

```
# cat /root/.ssh/id_rsa_storm1.pub >> /root/.ssh/authorized_keys
```

```
# chmod 600 /root/.ssh/authorized_keys
```

### (2)、ansible配置

```
# mkdir -p /etc/ansible
```

```
# vim /etc/ansible/ansible.cfg
```

```
.....
```

```
remote_port = 36000
```

```
private_key_file = /root/.ssh/id_rsa_storm1
```

```
.....
```

```
##主机组定义
```

```
# vim /etc/ansible/hosts
```

```
[storm_cluster]
```

```
10.223.55.100
```

```
10.223.55.101
```

```
10.223.38.226
```

```
10.223.38.227
```

```
10.223.39.216
```


```
10.223.25.123
```

**(3)、简单测试**

```
# ansible storm_cluster -m command -a 'uptime'
```

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m command -a 'uptime'
```

paramiko: The authenticity of host '[10.223.55.100]:36000' can't be established.  
The ssh-rsa key fingerprint is d2b1d47971f689403ble2985815da  
Are you sure you want to continue connecting (yes/no)?




说明：第一次运行时，需要输入一下“yes”【进行公钥验证】，后续无需再次输入。

##再次运行

```
# ansible storm_cluster -m command -a 'uptime'
```

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m command -a 'uptime'
```

Host	Uptime	Load Average
10.223.55.101	20:04:19 up 31 days, 5:20, 2 users	5.83, 5.23, 5.71
10.223.55.100	20:04:19 up 31 days, 5:20, 2 users	5.09, 3.82, 3.45
10.223.38.226	20:04:19 up 31 days, 5:20, 1 user	2.41, 3.03, 2.97
10.223.38.227	20:04:19 up 31 days, 5:20, 1 user	6.04, 5.09, 4.19
10.223.25.123	20:04:19 up 31 days, 5:20, 1 user	2.35, 3.37, 3.61
10.223.39.216	20:04:19 up 31 days, 5:20, 1 user	2.68, 3.37, 3.61



### 3、常用模块使用

#### (1)、setup

##用来查看远程主机的一些基本信息

# ansible storm\_cluster -m setup

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m setup
10.223.38.226 | success >> {
  "ansible_facts": {
    "ansible_all_ipv4_addresses": [
      "10.223.38.226"
    ],
    "ansible_all_ipv6_addresses": [],
    "ansible_architecture": "x86_64",
    "ansible_bios_date": "01/23/2014",
    "ansible_bios_version": "1.00",
    "ansible_cmdline": {
      "console": "tty0",
      "crashkernel": "512M-2G:64M,2G-12G:128M",
      "i8042.noaux": true,
      "ro": true,
      "root": "/dev/sda1"
    }
  }
}
```

51CTO.com  
技术博客 Blog

(2)、ping

##用来测试远程主机的运行状态

# ansible storm\_cluster -m ping

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m ping
10.223.38.226 | success >> {
  "changed": false,
  "ping": "pong"
}

10.223.55.100 | success >> {
  "changed": false,
  "ping": "pong"
}

10.223.38.227 | success >> {
  "changed": false,
  "ping": "pong"
}
```

### (3)、file

##设置文件的属性

相关选项如下：

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

group：定义文件/目录的属组

mode：定义文件/目录的权限

owner：定义文件/目录的属主

path：必选项，定义文件/目录的路径

recurse：递归设置文件的属性，只对目录有效

src: 被链接的源文件路径, 只应用于state=link的情况

dest: 被链接到的路径, 只应用于state=link的情况

state:

directory: 如果目录不存在, 就创建目录

file: 即使文件不存在, 也不会被创建

link: 创建软链接

hard: 创建硬链接

touch: 如果文件不存在, 则会创建一个新的文件, 如果文件或目录已存在, 则更新其最后修改时间

absent: 删除目录、文件或者取消链接文件

示例:

##远程文件符号链接创建

```
# ansible storm_cluster -m file -a "src=/etc/resolv.conf dest=/tmp/resolv.conf state=link"
```

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m file -a "src=/etc/resolv.conf dest=/tmp/resolv.conf state=link"
10.223.55.100 | success >> |
  "changed": true,
  "dest": "/tmp/resolv.conf",
  "gid": 0,
  "group": "root",
  "mode": "0777",
  "owner": "root",
  "size": 14,
  "src": "/etc/resolv.conf",
```

##远程文件信息查看

# ansible storm\_cluster -m command -a "ls -al /tmp/resolv.conf"

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m command -a "ls -al /tmp/resolv.conf"
10.223.38.226 | success | rc=0 >>
lrwxrwxrwx 1 root root 16 Nov 15 20:16 /tmp/resolv.conf -> /etc/resolv.conf

10.223.55.100 | success | rc=0 >>
lrwxrwxrwx 1 root root 16 Nov 15 20:16 /tmp/resolv.conf -> /etc/resolv.conf

10.223.38.227 | success | rc=0 >>
lrwxrwxrwx 1 root root 16 Nov 15 20:16 /tmp/resolv.conf -> /etc/resolv.conf

10.223.55.101 | success | rc=0 >>
lrwxrwxrwx 1 root root 16 Nov 15 20:16 /tmp/resolv.conf -> /etc/resolv.conf

10.223.25.123 | success | rc=0 >>
lrwxrwxrwx 1 root root 16 Nov 15 20:16 /tmp/resolv.conf -> /etc/resolv.conf

10.223.39.216 | success | rc=0 >>
lrwxrwxrwx 1 root root 16 Nov 15 20:16 /tmp/resolv.conf -> /etc/resolv.conf
```

51CTO.com  
技术博客 Blog

##远程文件符号链接删除

# ansible storm\_cluster -m file -a "path=/tmp/resolv.conf state=absent"

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m file -a "path=/tmp/resolv.conf state=absent"
10.223.38.227 | success => |
    "changed": true,
```

##远程文件信息查看

# ansible storm\_cluster -m command -a "ls -al /tmp/resolv.conf"

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m command -a "ls -al /tmp/resolv.conf"
10.223.38.227 | FAILED | rc=3 >>
ls: cannot access /tmp/resolv.conf: No such file or directory

10.223.38.100 | FAILED | rc=3 >>
ls: cannot access /tmp/resolv.conf: No such file or directory


10.223.38.101 | FAILED | rc=3 >>
ls: cannot access /tmp/resolv.conf: No such file or directory

10.223.38.225 | FAILED | rc=3 >>
ls: cannot access /tmp/resolv.conf: No such file or directory

10.223.38.226 | FAILED | rc=3 >>
ls: cannot access /tmp/resolv.conf: No such file or directory

10.223.38.228 | FAILED | rc=3 >>
ls: cannot access /tmp/resolv.conf: No such file or directory

10.223.38.229 | FAILED | rc=3 >>
ls: cannot access /tmp/resolv.conf: No such file or directory
```



说明：如上显示，代表文件或链接已经删除。

(4)、copy

##复制文件到远程主机

相关选项如下：

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

content：用于替代“src”，可以直接设定指定文件的值



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

directory\_mode: 递归设定目录的权限, 默认为系统默认权限

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

others: 所有的file模块里的选项都可以在这里使用

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

示例:

```
##将本地文件"/etc/ansible/ansible.cfg"复制到远程服务器
```

```
# ansible storm_cluster -m copy -a "src=/etc/ansible/ansible.cfg dest=/tmp/ansible.cfg  
owner=root group=root mode=0644"
```

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m copy -a "src=/etc/ansible/ansible.cfg dest=/tmp/ansible.cfg owner=root group=root mode=0644"
10.223.38.227: success: >> |
10.223.38.227: success: >> |
```

##远程文件信息查看

# ansible storm\_cluster -m command -a "ls -al /tmp/ansible.cfg"

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m command -a "ls -al /tmp/ansible.cfg"
10.223.38.226 | success | rc=0 >>
-rw-r--r-- 1 root root 7484 Nov 15 20:31 /tmp/ansible.cfg

10.223.55.101 | success | rc=0 >>
-rw-r--r-- 1 root root 7484 Nov 15 20:31 /tmp/ansible.cfg

10.223.55.100 | success | rc=0 >>
-rw-r--r-- 1 root root 7484 Nov 15 20:31 /tmp/ansible.cfg

10.223.38.227 | success | rc=0 >>
-rw-r--r-- 1 root root 7484 Nov 15 20:31 /tmp/ansible.cfg

10.223.25.123 | success | rc=0 >>
-rw-r--r-- 1 root root 7484 Nov 15 20:31 /tmp/ansible.cfg

10.223.39.216 | success | rc=0 >>
-rw-r--r-- 1 root root 7484 Nov 15 20:31 /tmp/ansible.cfg
```

51CTO.com  
技术博客 Blog

(5)、command

##在远程主机上执行命令

相关选项如下:

creates: 一个文件名, 当该文件存在, 则该命令不执行

free\_form: 要执行的linux指令

chdir: 在执行指令之前, 先切换到该目录

removes: 一个文件名, 当该文件不存在, 则该选项不执行

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

示例:

```
# ansible storm_cluster -m command -a "uptime"
```

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m command -a "uptime"
10.223.38.226 | success | rc=0 >>
 20:37:38 up 31 days,  5:53,  1 user,  load average: 3.23, 3.34, 3.17


10.223.55.100 | success | rc=0 >>
 20:37:38 up 31 days,  5:53,  2 users, load average: 4.49, 4.29, 3.96

10.223.38.227 | success | rc=0 >>
 20:37:38 up 31 days,  5:53,  1 user,  load average: 3.83, 3.76, 3.65

10.223.95.101 | success | rc=0 >>
 20:37:38 up 31 days,  5:53,  2 users, load average: 4.62, 4.60, 4.77

10.223.39.216 | success | rc=0 >>
 20:37:39 up 31 days,  5:53,  1 user, load average: 5.48, 5.48, 5.48

10.223.25.123 | success | rc=0 >>
 20:37:39 up 31 days,  5:53,  1 user, load average: 4.97, 5.79, 5.05
```



(6)、shell

##切换到某个shell执行指定的指令, 参数与command相同。

与command不同的是, 此模块可以支持命令管道, 同时还有另一个模块也具备此功能: raw

示例:

##先在本机创建一个SHELL脚本

```
# vim /tmp/rocketzhang_test.sh
```

```
#!/bin/sh
```

```
date +%F_%H:%M:%S
```

```
#chmod +x /tmp/rocketzhang_test.sh
```


```
##将创建的脚本文件分发到远程
```

```
# ansible storm_cluster -m copy -a "src=/tmp/rocketzhang_test.sh  
dest=/tmp/rocketzhang_test.sh owner=root group=root mode=0755"
```

```
##远程执行
```

```
# ansible storm_cluster -m shell -a "/tmp/rocketzhang_test.sh"
```

```
[root@storm_nimbus /etc/ansible]# ansible storm_cluster -m shell -a "/tmp/rocketzhang_test.sh"
10.223.38.227 | success | rc=0 >>
2014-11-15_20:48:22
10.223.38.226 | success | rc=0 >>
2014-11-15_20:48:22
10.223.55.101 | success | rc=0 >>
2014-11-15_20:48:22
10.223.55.100 | success | rc=0 >>
2014-11-15_20:48:22
10.223.39.216 | success | rc=0 >>
2014-11-15_20:48:23
10.223.25.123 | success | rc=0 >>
2014-11-15_20:48:23
```



## (7)、更多模块

其他常用模块，比如：service、cron、yum、synchronize就不一一例举，可以结合自身的系统环境进行测试。

service: 系统服务管理

cron: 计划任务管理

yum: yum软件包安装管理

synchronize: 使用rsync同步文件

user: 系统用户管理

group: 系统用户组管理

更多模块可以参考:

#ansible-doc -l

```
[root@storm_nimbus /etc/ansible]# ansible-doc -l
```

[http://docs.ansible.com/modules\\_by\\_category.html](http://docs.ansible.com/modules_by_category.html)

<http://www.ansible.cn/docs/>

(国内的一个镜像站点，避免被墙^\_^)

### (8)、一些概念补充

**playbook的组成：**playbook是由一个或多个“play”组成的列表，可以让它们联同起来按事先编排的机制执行；所谓task无非是调用ansible的一个module，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；

**执行模型：**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。在顺序运行某playbook时，如果中途发生错误，所有已执行任务都将回滚，因此，在修改playbook后重新执行一次即可；

**task组成：**每个task都应该有其name，用于playbook的执行结果输出，建议其内容尽可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；

**notify指定handler的执行机制：**“notify”这个action可用于在每个play的最后被触发，在notify中列出的操作称为handler，仅在所有的变化发生完成后一次性地执行指定操作。

=====

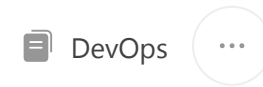
=====

### 三、后续工作

=====

=====

- 1、深入学习ansible的playbook以及扩展模块；
- 2、结合业务环境，初步实现基础监控，以取代目前调用自动化部署平台API的方式；
- 3、尝试自动化运维工具saltstack，并将其与ansible进行对比。



"小礼物走一走，来简书关注我"

赞赏支持

还没有人赞赏，支持一下



帅T

总资产2 (约0.21元) 共写了3.2W字 获得61个赞 共27个粉丝

关注

写下你的评论...

全部评论 0 只看作者

按时间倒序 按时间正序

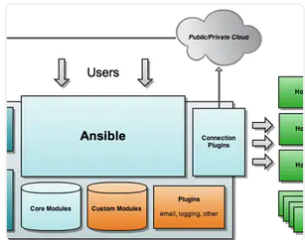
推荐阅读

自动化运维工具Ansible使用介绍

本文主要内容均收集于网络上的博文资料，仅以此文作为学习总结。BTW，目前Ansible对python3的支持还不是...

qiuyi943 阅读 14,779 评论 1 赞 14

更多精彩内容>



2017 11-17 ansible应用

一.ansible (1) ansible: ansible是一款新出现的自动化运维系统，基于python开发并集...

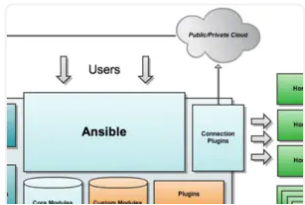
楠人帮 阅读 1,319 评论 0 赞 8

```
Manage A10 Networks AX/SoftAX/Thunder/vThunder dev
Manage A10 Networks devices' service groups
Manage A10 Networks devices' virtual servers
Sets and retrieves file ACL information
add a host (and alternatively a group) to the ansible
Notify airbrake about app deployments
Manages alternative programs for common commands
Set and/or get members' attributes of an Apache httpd
enables/disables a module of the Apache2 webserver
Manages apt packages
Manages apt packages
Add or remove an apt key
Add and remove APT repositories
apt rpm package manager
Manage access-lists on a Cisco ASA
Run arbitrary commands on Cisco ASA devices
Manage Cisco ASA configuration sections
Assembles a configuration file from fragments
Asserts given expressions are true
Obtain status of asynchronous Task
```

ansible自动化运维工具

(一)架构 Ansible CoreModulesCore Modules(核心模块)Customed Modul...

uangianlap 阅读 1,617 评论 1 赞 4





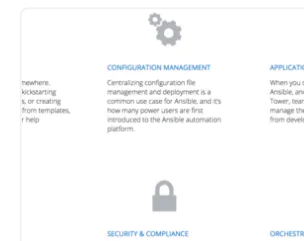


## 《Ansible\_Up-And-Running》笔记1-Ansible超详细使用指南

在项目中有很多地方用到ansible。最初使用ansible只是为了方便代码部署和模板配置，毕竟手动去30+台机器...



\_七把刀\_ 阅读 9,485 评论 6 赞 81



2017.3.21

今天做捕梦网的时候，一直在想男朋友，想我们之间的事情，我觉得自己受到的不尊重和伤害，我问自己为什么，我们之间就总是...



小酒窝天使 阅读 48 评论 0 赞 0

