01.Ansible 快速入门

1.Ansible基础概述

1.什么是Ansible

- 基础批量管理工具
  - 1 Xshell/SecureCRT
  - 2 密钥认证（免密码登录）+命令pssh
  - 3 密钥认证（免密码登录）+ 脚本
  - 4 批量管理工具：**Ansible**/SaltStack

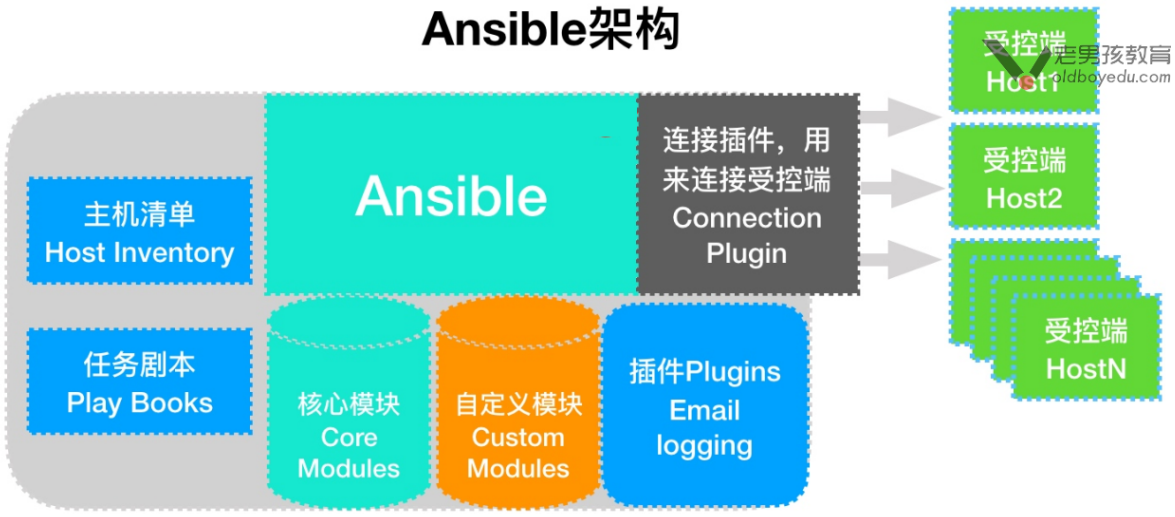
自动化本质:完成重复性工作,减少占用时间， 核心:标准化

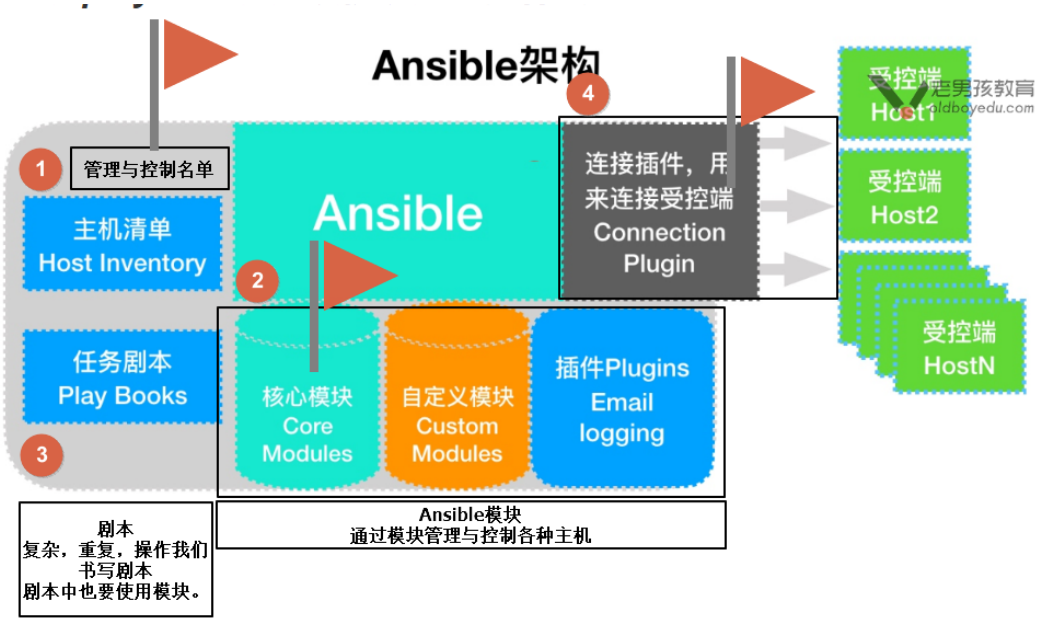
Ansible是一个IT自动化的配置管理工具，自动化主要体现在Ansible集成了丰富模块，丰富的功能的组件，可以通过一个命令行完成一系列的操作。进而能减少我们重复性的工作和维护成本，以提高工作的效率。

*2.Ansible可以完成哪些功能呢

1) ✲✲✲✲✲执行命令： **批量执行**远程**命令**,并发，可以对N多台主机同时进行命令的执行。
2) ✲✲✲✲配置管理： **批量配置**软件服务，可以进行自动化的方式配置和管理服务。
3) **二次开发**： 实现软件开发功能，jumpserver底层使用ansible来实现的自动化管理。
4) ✲✲✲✲剧本（playbook): **编排高级**的IT任务，Ansible的Playbook是一门编程语言，可以用来描绘一套IT架构*。

4.Ansible的架构中的控制节点、被控制节点、inventroy、ad-hoc、playbook、连接协议这些是什么?





3.Ansible安装配置

3.1 Ansible环境准备

ansible环境准备	功能	ip地址
m01	ansible	10.0.0.61/172.16.1.61
web01		10.0.0.7/172.16.1.7
backup		10.0.0.41/172.16.1.41
nfs01		10.0.0.31/172.16.1.31
db01		10.0.0.51/172.16.1.51

3.2 环境准备与检查

```
#01 用户与端口
[root@m01 ~]# whoami
root
[root@m01 ~]# ss -lntup |grep ssh
tcp    LISTEN     0      128          *:22
*:*                   users:(("sshd",pid=1093,fd=3))
tcp    LISTEN     0      128       [::]:22
[::]:*                   users:(("sshd",pid=1093,fd=4))
[root@m01 ~]#

#02 防火墙与selinux
[root@m01 ~]# getenforce
Disabled
[root@m01 ~]# systemctl is-active firewalld.service
iptables.service
unknown
inactive
```

3.3 Ansible安装

1) Ansible急速安装

```
#01 配置 epel 源


#02 安装ansible

[root@manager ~]# yum install ansible -y
```

```
#查看ansible的版本
[root@manager ~]# ansible --version
ansible 2.8.0
  config file = /etc/ansible/ansible.cfg   #配置文件的位置
  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 11 2018, 07:36:10)
[GCC 4.8.5 20150623 (Red Hat 4.8.5-28)]


# 配置文件
 /etc/ansible/ansible.cfg

#




# ansible <host-pattern> [options]
--version   #ansible版本信息
-v          #显示详细信息  用于排错   -vvv    -vvvv debug模式

-i          #inentory 文件 主机清单文件路径，默认是
在/etc/ansible/hosts

-m          #使用的模块名称，默认使用command模块      module

-a          #使用的模块参数，模块的具体动作           action 模
块中的具体功能/参数

-k          #提示输入ssh密码，而不使用基于ssh的密钥认证  ansible
默认时候 基于密钥认证

-C          #模拟执行测试，但不会真的执行   --check 检查语法
(playbook(剧本中必备))
```

```
-T              #执行命令的超时   timeout 超时时间
```

2) Ansible核心选项

Ansible选项	含义	应用场景
--version	**ansible版本信息**	
-v	--verbose 显示过程 -v -vv -vvvv 排错	排错
-i	inventory 指定主机清单位置 默认是/etc/ansible/hosts	指定主机清单，剧本
-m	module 指定模块	
-a	action 模块中的具体的功能（动作）	
-k	指定被控的节点的密码	如果没有配置秘钥认证
-C	--check 检查与测试 （书写剧本playbook必备）	书写剧本检查语法
-T	timeout 超时时间， 默认是10s	
-f	修改并发数，默认是5	

3) Ansible主配置文件

*2.Ansible的配置文件可以存放在任何位置，但配置文件有读取顺序，先Ansible配置做一个基本**了解。***

ansible的配置文件有查找顺序：

1) 最先查找$ANSIBLE_CONFIG变量对应的位置和文件

2) 其次查找当前目录下ansible.cfg

3) 然后查找用户家目录下的.ansible.cfg

*4) 最后查找/etc/ansible/ansible.cfg(**默认**)*

Ansible.cfg 配置文件		
inventory	默认的主机清单的位置	
remote_tmp	默认的远程的临时目录的位置 ~/.ansible/tmp	

Ansible.cfg 配置文件		
local_tmp	默认的本地的临时 目录 ~/.ansible/tmp	
forks	并发数量 默认是5	
sudo_user	使用sudo的时候切换为哪个用户（root)	
ask_sudo_pass	True 强制让你输入sudo用户的密码 （当前用户密码)	
ask_pass	True 如果没有配置秘钥认证，是否询问密码	
remote_port	远程的ssh服务的端口号 默认是22	
host_key_checking	⚠修改为False 连接前是否验证主机信息	
log_path	/var/log/ansible.log 指定日志路径及名字	

- 主配置文件超级详解

```
[root@manager ~]# cat /etc/ansible/ansible.cfg

#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     = ~/.ansible/tmp          #临时py文件存放在远程主机目录
#local_tmp      = ~/.ansible/tmp          #本机的临时执行目录

#forks          = 5                       #默认并发数

#sudo_user      = root                    #默认sudo用户
```

```
#ask_sudo_pass = True                          #每次执行是否询问
sudo的ssh密码
#ask_pass      = True                          #每次执行是否询问
ssh密码

#remote_port    = 22                           #远程主机端口
ssh端口号
host_key_checking = False                      #跳过检查主机指纹
log_path = /var/log/ansible.log                #ansible日志


#如果是普通用户则需要配置提权
[privilege_escalation]
#become=True
#become_method=sudo
#become_user=root
#become_ask_pass=False



[root@m01 ~]# egrep -v  '#|^$'  /etc/ansible/ansible.cfg
[defaults]
host_key_checking = False
log_path = /var/log/ansible.log
[inventory]
[privilege_escalation]
[paramiko_connection]
[ssh_connection]
[persistent_connection]
[accelerate]
[selinux]
[colors]
[diff]
#
```

4.Ansible Inventory

- 主机清单

- 1某一台主机、多台
- 2给主机进行分组
- 3划分子组

Inventory文件中填写需要被管理主机与主机组信息(逻辑上定义)。默认Inventory文件在/etc/ansible/hosts。当然也可以自定义，然后使用-i指定Inventory文件位置。下面通过几个场景演示，如何配置Inventory文件

4.1 场景一、基于密码连接 ✨✨✨

```
[root@oldboy.com ~]# cat /etc/ansible/hosts

#方式一、主机+端口+密码 ※※※※※
[webservers]
10.0.0.31 ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='123456'
10.0.0.41 ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='123456'

#方式二、主机+端口+密码
[webservers]
web[01:04].oldboy.com ansible_ssh_pass='123456'

#方式三、主机+端口+密码
[webservers]
web[1:2].oldboy.com

[webservers:vars]     #给webservers 主机组 设置共用 变量
variables
```

```
ansible_ssh_pass='123456'

root@m01 ~]# cat  /etc/ansible/hosts
[web]
172.16.1.7    ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
172.16.1.8    ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
172.16.1.9    ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
172.16.1.10   ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
[root@m01 ~]# ansible  172.16.1.7 -m command  -a
'hostname'
172.16.1.7 | CHANGED | rc=0 >>
web01
[root@m01 ~]# ansible  web -m command  -a 'hostname'
172.16.1.7 | CHANGED | rc=0 >>
web01
172.16.1.10 | CHANGED | rc=0 >>
web04
172.16.1.8 | CHANGED | rc=0 >>
web02
172.16.1.9 | CHANGED | rc=0 >>
web03



[root@m01 ~]# tail -4   /etc/hosts
172.16.1.7   web1.oldboy.com
172.16.1.8   web2.oldboy.com
172.16.1.9   web3.oldboy.com
172.16.1.10  web4.oldboy.com
[root@m01 ~]# cat  /etc/ansible/hosts
[web]
172.16.1.7    ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
172.16.1.8    ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
172.16.1.9    ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
```

```
172.16.1.10   ansible_ssh_port=22 ansible_ssh_user=root
ansible_ssh_pass='1'
[webserver]
web[1:4].oldboy.com ansible_ssh_pass='1'


[root@m01 ~]# ansible webserver  -m  ping
web3.oldboy.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
web2.oldboy.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
web4.oldboy.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
web1.oldboy.com | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
[root@m01 ~]#

[root@m01 ~]# ansible webserver  -a   'hostname'
web3.oldboy.com | CHANGED | rc=0 >>
web03
web1.oldboy.com | CHANGED | rc=0 >>
```

```
web01
web4.oldboy.com | CHANGED | rc=0 >>
web04
web2.oldboy.com | CHANGED | rc=0 >>
web02



[root@m01 ~]# cat /etc/ansible/hosts
# This is the default ansible 'hosts' file.
#
# It should live in /etc/ansible/hosts
#
#   - Comments begin with the '#' character
#   - Blank lines are ignored
#   - Groups of hosts are delimited by [header] elements
#   - You can enter hostnames or ip addresses
#   - A hostname/ip can be a member of multiple groups

# Ex 1: Ungrouped hosts, specify before any group headers.

## green.example.com
## blue.example.com
## 192.168.100.1
## 192.168.100.10

# Ex 2: A collection of hosts belonging to the
'webservers' group

## [webservers]
## alpha.example.org
## beta.example.org
## 192.168.1.100
## 192.168.1.110

# If you have multiple hosts following a pattern you can
specify
# them like this:

## www[001:006].example.com
```

```
# Ex 3: A collection of database servers in the
'dbservers' group

## [dbservers]
##
## db01.intranet.mydomain.net
## db02.intranet.mydomain.net
## 10.25.1.56
## 10.25.1.57

# Here's another example of host ranges, this time there
are no
# leading 0s:

## db-[99:101]-node.example.com
```

*4.2 场景二、基于密钥连接，需要先创建公钥和私钥，并下发公钥至被控端 ✲ ✲ ✲ ✲ ✲

```
[root@manager ~]# ssh-copy-id -i ~/.ssh/id_rsa.pub
root@172.16.1.7
[root@manager ~]# ssh-copy-id -i ~/.ssh/id_rsa.pub
root@172.16.1.8

[root@m01 ~]# for n  in {5..10}  31 41 51   ;  do  ssh
172.16.1.$n hostname    ;done
lb01
ssh: connect to host 172.16.1.6 port 22: No route to host
web01
web02
web03
web04
nfs01
backup
db01
[root@m01 ~]# cat  hosts
```

```
[lb]
172.16.1.5
172.16.1.6
[web]
172.16.1.7
172.16.1.8
172.16.1.9
172.16.1.10
[data]
172.16.1.31
172.16.1.41
172.16.1.51

[root@m01 ~]# ansible all -i hosts   -a 'hostname'
172.16.1.9 | CHANGED | rc=0 >>
web03
172.16.1.10 | CHANGED | rc=0 >>
web04
172.16.1.8 | CHANGED | rc=0 >>
web02
172.16.1.7 | CHANGED | rc=0 >>
web01
172.16.1.5 | CHANGED | rc=0 >>
lb01
172.16.1.31 | CHANGED | rc=0 >>
nfs01
172.16.1.41 | CHANGED | rc=0 >>
backup
172.16.1.51 | CHANGED | rc=0 >>
db01
172.16.1.6 | UNREACHABLE! => {
    "changed": false,
    "msg": "Failed to connect to the host via ssh: ssh:
connect to host 172.16.1.6 port 22: No route to host",
    "unreachable": true
}
```

```
#方式一、主机+端口+密钥
[root@manager ~]# cat hosts
[webservers]
172.16.1.7
172.16.1.8

#方式二、别名+主机+端口+密钥
[root@manager ~]# cat hosts
[webservers]
web01 ansible_ssh_host=172.16.1.7 ansible_ssh_port=22
web02 ansible_ssh_host=172.16.1.8
```

*4.3 场景三、主机组使用方式 * ✳✳✳✳✳

- 主机清单中
  - 直接书写主机ip/域名
  - **主机分组**
  - 主机分组：**子组**

```
[lbservers] #定义lbservers组
172.16.1.5
172.16.1.6

[webservers] #定义webserver组
172.16.1.7
172.16.1.8

[servers:children]  #定义servers组包括两个子组
[lbservers,webserver]
lbservers
webserver

servers组   包含的lb和web两个组的内容
```

```
[backup]
172.16.1.41
[db]
172.16.1.51
[nfs]
172.16.1.31

[data:children]   #data 主机组    backup   db nfs 作为子组
backup
db
nfs


[root@m01 ~]# cat  /etc/ansible/hosts
[web]
172.16.1.7

[backup]
172.16.1.41

[nfs]
172.16.1.31

[db]
172.16.1.51

[data:children]
#组的名字
backup
nfs
db

[root@m01 ~]# ansible nfs -m ping
[root@m01 ~]# ansible nfs -m ping
172.16.1.31 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

```
[root@m01 ~]# ansible data  -m ping
172.16.1.41 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
172.16.1.31 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
172.16.1.51 | SUCCESS => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/bin/python"
    },
    "changed": false,
    "ping": "pong"
}
```

4.4 了解：列出每个主机组下面的主机情况

```
[root@manager ~]# ansible lbservers  -i ./hosts --list-hosts
  hosts (1):
    web01
[root@manager ~]# ansible webservers  -i ./hosts --list-hosts
  hosts (1):
    web02
[root@manager ~]# ansible servers -i ./hosts --list-hosts
  hosts (2):
    web01
    web02
[root@manager ~]# ansible all  -i ./hosts --list-hosts
  hosts (3):
```

```
web01
web02
web03
```

5. 总结

- Ansible架构-**主机清单，模块，剧本** ✨✨✨
- Ansible部署：yum/apt
- Ansible命令的**选项与配置**✨✨✨
- Ansible Inventory 主机清单使用： **分组，子组** ✨✨✨✨✨