链滴

Ansible

作者：wysl
原文链接：https://ld246.com/article/1566184321418
来源网站：链滴

title: Ansible
categories:

- 自动化运维
- 配置管理工具

tags: []
toc: true
author: 一剑风
date: 2018-11-11 02:28:00
permalink:
description:
image:

机器配置

主控端 192.168.195.100：
CentOS7.0 python2.7.5

Linux被控制端 192.168.195.101：
Centos7.0 SSH

Windows被控制端 192.168.195.132：
Win7 PoweShell 4.0

<!--more-->

主控制端安装说明

使用yum安装

{% codeblock lang:shell %}

下载阿里云yum源到本机

wget -P /etc/yum.repos.d/ http://mirrors.aliyun.com/repo/Centos-7.repo
yum clean all
yum makecache
yum install ansible -y
{% endcodeblock %}

<p>显示如下则安装成功

已安装:
  ansible.noarch 0:2.4.2.0-2.el7
作为依赖被安装:
  PyYAML.x86_64 0:3.10-11.el7 libyaml.x86_64 0:0.1.4-11.el7_0
  python-babel.noarch 0:0.9.6-8.el7 python-httplib2.noarch 0:0.9.2-1.el7
  python-jinja2.noarch 0:2.7.2-2.el7 python-markupsafe.x86_64 0:0.11-10.el7
  python-paramiko.noarch 0:2.1.1-9.el7 python-passlib.noarch 0:1.6.5-2.el7
  python2-jmespath.noarch 0:0.9.0-3.el7 sshpass.x86_64 0:1.06-2.el7
</p>

windows的管理模块安装

{% codeblock lang:shell %}

安装pip

curl https://bootstrap.pypa.io/get-pip.py -o get-pip.py

python get-pip.py

安装pywinrm模块

pip install pywinrm

如果上行命令提醒requests无法卸载则执行

pip install --ignore-installed requests && pip install pywinrm

{% endcodeblock %}

添加被控制端

{% codeblock lang:shell %}

添加主机ip并分组

cat >> /etc/ansible/hosts <<EOF

linux分组:

[linux]

192.168.195.101

windwos分组并设定:

[win]

192.168.195.132

[win:vars]

ansible_user=Administrator
ansible_password=admin
ansible_port=5986
ansible_connection=winrm
ansible_winrm_server_cert_validation=ignore
EOF
{% endcodeblock %}

测试控制Linux端是否可用，并设定免密登录：

需先设定好Linux被控制端（点击跳转）
{% codeblock lang:shell %}

测试是否能连通，输入对方root密码，返回"pong" 则为通

ansible 192.168.195.101 -m ping -k

设定使用ssh-key连接，设定文件名和密码

ssh-keygen
ssh-copy-id -i .ssh/id_rsa.pub root@192.168.195.101

测试免密登录是否可用,和批量测试linux分组下的机器是可用

ansible 192.168.195.101 -m ping
ansible linux -m ping
{% endcodeblock %}

测试控制Windows端是否可用：

需先设定好Windows被控制端（点击跳转）

```
# 测试是否能连通，输入对方root密码，返回"pong" 则为连通
ansible 192.168.195.132 -m win_ping

# 测试windwos分组下的机器是否可用
ansible win -m win_ping

# 测试命令行是否可用
ansible win -m win_shell -a "ping baidu.com -n 1"
```

对Linux控制常用命令

{% codeblock lang:shell %}

ping

ansible 192.168.195.101 -m ping

command 模块（有局限性，对管道和重定向之类有局性，建议用shell）

#1. chdir 先cd再操作

ansible linux -a 'chdir=/tmp ls'

#2. creates XX已经存在将不执行操作

ansible linux -a 'creates=/tmp/1 ls'

#3. removes XX不存在将不执行

ansible linux -a 'removes=/tmp/1 ls'

shell 模块

ansible linux -m shell -a 'echo $HOSTNAME'

#1. chdir 先cd再操作

ansible linux -m shell -a 'chdir=/tmp ls'

#2. creates XX已经存在将不执行操作

ansible linux -m shell -a 'creates=/tmp/1 ls'

#3. removes XX不存在将不执行

ansible linux -m shell -a 'removes=/tmp/1 ls'

script 远程执行脚本（不需推送）

ansible linux -m script -a './host.sh'

copy 复制文件到被控制端

#1. backup 覆盖并备份原文件

ansible linux -m copy -a 'src=./110.txt dest=/tmp/110.txt backup=yes'

#2. mode/owner修改权限和所有者

ansible linux -m copy -a 'src=./110.txt dest=/tmp/110.txt mode=644 owner=test'

#3. content 覆盖修改文件内容，不存在则创建

ansible linux -m copy -a 'content="test\n\content" dest=/tmp/110.txt mode=644'

fetch 抓取控制端上的文件到本机（单文件）结果包含源录结构

ansible linux -m fetch -a 'src=/var/log/zabbix/zabbix_agentd.log dest=/tmp'

file 模块

#1. state=directory 可递归创建文件夹
ansible linux -m file -a 'path=/tmp/file/file state=directory'
#2. state=touch 创建文件/更新现有文件的修改时间
ansible linux -m file -a 'path=/tmp/file/file1 state=touch'
#3. state=link 创建软链接
ansible linux -m file -a 'src=/tmp/file1 dest=/tmp/file1.link state=link'
#4. state=absent 可递归删除文件/文件夹的内容/软链接
ansible linux -m file -a 'path=/tmp/file/file1 state=absent'
ansible linux -m file -a 'path=/tmp/file state=absent'

cron 计划任务

#1. name 创建计划任务并指定名称
ansible linux -m cron -a 'minute=* weekday=1-5 job="/tmp/hostname.sh" name=showhost'
#2. 禁用计划任务
ansible linux -m cron -a 'disabled=true job="/tmp/hostname.sh" name=showhost'
#3. 删除计划任务
ansible linux -m cron -a 'job="/tmp/hostname.sh" name=showhost state=absent'

yum 管理yum

#1. name=XXX 默认从yum源安装指定软件
ansible linux -m yum -a 'name=XXX,YYY,ZZZ'
#2. state=absent 卸载包
ansible linux -m yum -a 'name=XXX state=absent'
#3. 安装RPM包，并且忽略gpg_check
ansible linux -m yum -a 'name=/tmp/nginx....rpm disable_gpg_check=yes'
#4. 更新yum缓存并安装
ansible linux -m yum -a 'name=XXX update_cache=yes

service 管理服务

#1. state=started/stopped/restarted/reloaded 管理服务
ansible linux -m service -a 'name=zabbix-agent state=started'
#2. enabled 设置开启自启
ansible linux -m service -a 'name=zabbix-agent enabled=yes'

user 管理用户

#1. 新增nginx账户设定为系统用户UID=80,不自动登录，默认加目录为/var/nginx，设定附加组

ansible linux -m user -a 'name=nginx shell=/sbin/nologin system=yes home=/var/nginx groups=root,bin uid=80'

#2. 删除nginx账户，并删除home目录

ansible linux -m user -a 'name=nginx state=absent remove=yes'

group 管理组

#1. 创建组 并设为系统组，指定gid=80

ansible linux -m group -a 'name=nginx system=yes gid=80'

#2. 删除组

ansible linux -m group -a 'name=nginx state=absent'

{% endcodeblock %}

对windwos控制常用命令

```
# ping
ansible win -m win_ping

# powershell
ansible win -m win_shell -a "ping baidu.com -n 1"

# win_shell 调用powershell下载bat 并执行
ansible all -m win_shell -a "powershell -command Invoke-WebRequest -Uri https://*/1.bat -OtFile c:\1.bat;c:\1.bat"

# win_stat 查看文件/文件夹信息
ansible all -m win_stat -a path="c:\110"
ansible all -m win_stat -a path="c:\110\putty.exe"

# win_copy 从本地复制文件到被控制端并备份,默认在内容不同时才传输
ansible all -m win_copy -a "src=1.txt dest=c:\1.txt backup=yes"

# win_file 文件/文件夹操作,state=directory/touch/absent
#1. state=directory 可递归创建文件夹
ansible all -m win_file -a "path=c:\1\2\3 state=directory"
#2. state=touch 创建文件/更新现有文件的修改时间
ansible all -m win_file -a "path=c:\1\2\3.txt state=touch"
#3. state=absent 可递归删除文件/文件夹的内容
ansible all -m win_file -a "path=c:\1\2\3.txt state=absent"
ansible all -m win_file -a "path=c:\1 state=absent"

# win_file_version 查看exe/dll文件版本
ansible all -m win_file_version -a "path=c:\windows\system32\cmd.exe"
ansible all -m win_file_version -a "path=c:\windows\system32\onex.dll"

# win_service 查看/控制服务
ansible all -m win_service -a "name=spooler"
#1. start_mode=auto/delayed/disabled/manual[自动/延迟/禁止/移除] 修改启动模式
ansible all -m win_service -a "name=spooler start_mode=auto"
#2. state=started/stopped/restarted[启动/停止/重启]
```

```
ansible all -m win_service -a "name=spooler state=started"

# win_unzip 解压zip档案至指定文件夹，文件夹不存则自动创建，默认覆盖已有文件
ansible all -m win_unzip -a "src=c:\1.zip dest=c:\2"

# win_lineinfile 在文件末尾插入字符串，如果检查到字符串存在，则不执行
ansible all -m win_lineinfile -a "path=c:\1\1.bat line=qqqq"
#1. win_lineinfile 修改文件，默认修改最后检查到的一行，若没有匹配到则在末尾插入
ansible all -m win_lineinfile -a "path=c:\1\1.bat regexp=11111 line=qqqq"
#1.1. backrefs=yes 若没有匹配到则不执行任何操作
ansible all -m win_lineinfile -a "path=c:\1\1.bat regexp=2q3w line=qqqq backrefs=yes"
#2. state=absent 全局检出匹配项，并移除
ansible all -m win_lineinfile -a "path=c:\1\1.bat regexp=^1 state=absent"

# win_firewall_rule 配置防火墙
ansible all -m win_firewall_rule -a "name='open 3333' localport=3333 action=allow direction=
n protocol=tcp state=present enable=yes"
```

被控制端说明

<apsn id="linux">

linux

1.被控制端必须使用SSH

2.注意selinux和iptables/firewall的设定

{% codeblock lang:shell %}

修改/etc/ssh/sshd_config加速查询

vim /etc/ssh/sshd_config

禁用SSH DNS查找,关闭/注释使用基于 GSSAPI 的用户证

GSSAPIAuthentication no

GSSAPICleanupCredentials no

UseDNS no

切换回shell重启SSH服务

systemctl restart sshd

<apsn id="windows">

windows

1.网络位置使用家庭网络/工作网络

2.windows大部分执行需要使用Aministrator，启用Administrator用户 并设定密码

3.下载安装：

.NET Framework 4.5

powershell4.0

使用Powershell 3.0或以上作为脚本运行的命令行：

```
# 查看当前系统powershell 版本
$get-host

# 设定允许运行不受信任的ps1脚本
set-ExecutionPolicy RemoteSigned

# 新建临时文件夹，下载执行脚本检查powershell，安装WinRM和开放端口5986
mkdir c:\tmp
cd c:\tmp
Invoke-WebRequest -Uri https://raw.githubusercontent.com/ansible/ansible/devel/examples/
cripts/ConfigureRemotingForAnsible.ps1 -OutFile ConfigureRemotingForAnsible.ps1
powershell -ExecutionPolicy RemoteSigned .\ConfigureRemotingForAnsible.ps1 -SkipNetwor
ProfileCheck

# 查看确认powershell端口监听状态
winrm enumerate winrm/config/listener
```

---

playbook

{% codeblock lang:shell %}

使用playbooks命令

ansible-playbook xxx.yml

#1. 使用标签指定运行其中一部分

ansible-playbook [tags],[tags] xxx.yml

#2. setup 示例查看被控制端指定的信息

ansible linux -m setup -a 'filter=ansible_all_ipv4_addresses'

{% endcodeblock %}

playbook 使用yaml 语言格式

指定执行被控制端

- hosts: linux

指定被控端使用什么用户执行

remote_user: root

task任务列表开始标头

tasks:

指定某一个Action的任务名

```
- name: task1
```

模块: 执行命令

```
  shell: ls
```

设定标签，不同的action标签可以一样，执行会按顺序都行

```
  tags: acttag1
```

第二个Action任务

```
- name: copy file
```

模块: 执行命令

```
  copy: src=/tmp/1.txt dest=/tmp backup=yes
```

如果发生改变 则引导触发器执行动作,可配置多个

```
  notify:
    - for 2nd action1
    - for 2nd action2
```

设定标签

```
  tags: acttag2
```

handlers触发器

触发器标头

handlers:

设定触发器任务名1

- name: for 2nd action1

模块: 执行命令

shell cat /tmp/1.txt

设定触发器任务名2

- name: for 2nd action2

模块: 执行命令

shell echo $? > /tmp/testlogs.log

{% endcodeblock %}

playbook中的变量

#在yml外部赋值变量

- hosts: linux

remote_user: root

tasks:

- name: install package

设定变量

yum: name=两个花括号中一个变量package

- name: start service
  service: name=两个花括号中一个变量package state=started enabled=yes

-e 执行对应标签的playbook内容

ansible-playbook -e 'package=httpd' xxx.yml

{% endcodeblock %}

#在yml内部定义变量赋值

- hosts: linux

remote_user: root

vars:

  - package1: xxx
  - package2: yyy

tasks:

  - name: install package

设定变量

```
yum: name=两个花括号中一个变量package1
- name: start service
  service: name=两个花括号中一个变量package2 state=started enabled=yes
```

{% endcodeblock %}

设定专门存放变量的文件

{% codeblock lang:yaml %}

创建一个yml文件存放变量:

vim vartest.yml

var1: xxxx

var2: yyyy

playbook调用此yml中的变量

- host: linux

romote_user: root

vars_files:

  - vartest.yml

tasks:

  - name: xxxxxx

module: name=两个花括号中一个变量var1

{% endcodeblock %}

{% codeblock lang:yaml %}

with_items 循环迭代

vim xxx.yml

.....

tasks:

- name: create some files

file: name=/tmp/两个花括号中一个变量item

with_items:

  - file1
  - file2
  - file3

- name: install some packages

yum: name=两个花括号中一个变量item

with_items:

  - httpd
  - zabbix

{% endcodeblock %}

{% codeblock lang:yaml %}

with_items 循环迭代嵌套

vim xxx.yml

.....

tasks:

- name: add some groups

group: name=两个花括号中一个变量item

with_items:

  - group1
  - group2
  - group3

- name: add some users

users: name=两个花括号中一个变量item.name 两个花括号中一个变量item.froup

with_items:

- { name: "user1", group: "group1"}
  - { name: "user2", group: "group2"}
  - { name: "user3", group: "group3"}

{% endcodeblock %}

template 配置模板

{% codeblock lang:yaml %}

使用jinj2格式使用，文件为.j2

建议位置/etc/ansible/templates/xxx.j2

copy 配置文件，使用变量运算赋值

示例设定naginx 进程数从setup取得cpu变量

vim ./templates/nginx.conf.j2 dest=/etc/nginx/nginx.conf

worker_processes 两个花括号中一个变量ansible_processor_vcpus*2;

在yplaybook使用template

vim xxx.yml

.....

-name: change conf file use tmplate

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

.....

使用判断when 做判断执行示例

.....

tasks:

- name:

template:

when: 某变量 == xxx

notify: zzz

- name:

template:

when: 某变量 == yyy

notify: zzz

handlers:

● name:

service: name=aaaa states=restarted

{% endcodeblock %}

命令以及配置参考内容

Ansible中文权威指南

ansible主机配置文件/etc/ansible/hosts

ansible主配置文件/etc/ansible/ansible.cfg