ARCHITECTURE-day01

大型架构及配置技术

# ansible基础

## ansible简介

### 什么是ansible？

ansible是2013年推出的一款IT自动化和DevOps软件，2015年被RedHat收购。是基于Python研发，糅合很多老运维工具的优点，实现了批量操作系统配置，批量程序部署，批量运行命令等功能

### 为什么选择ansible?

选择一款配置管理软件，从以下几点权衡利弊:

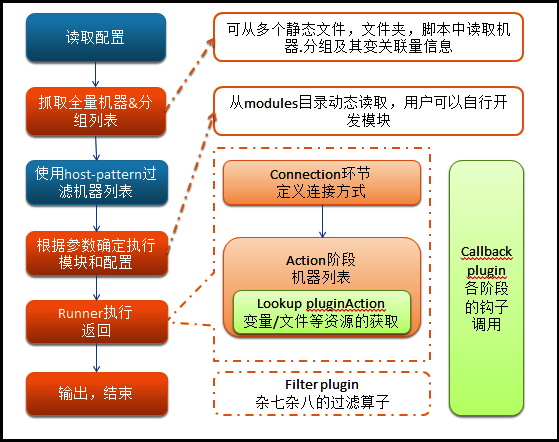
活跃度（社区）、学习成本、使用成本、编码语言、性能、使用是否广泛



|  |  |  |
| --- | --- | --- |
| ansible优点 | ansible特性 | ansible可以实现 |
| 只需要SSH和Python即可使用  无客户端  snsible功能强大，模块丰富  上手容易，门槛低  基于Python开发，做二此开发更容易  使用公司比较多，社区活跃 | 模块化设计，调用特定的模块完成特定任务  基于Python语言实现  - paramiko  - PyYAML  - jinja2  其模块支持JSON等标准输出格式，可以采用任何变成语言重写  部署简单  主从模式工作  支持自定义模块  支持playbook  易于使用  支持对层部署  支持异构IT环境 | 自动化部署APP  自动化管理配置项  自动化持续交付  自动化（AWS）运服务管理 |

## 工作流程

### ansible大体执行过程



## ansible安装

### 软件依赖关系

对管理主机：要求Python2.6或Python2.7

### ansible使用以下模块，都需要安装

- paramiko

- PyYAML

- jinja2

- httplib2

- six

### 对于被托管主机

- ansible默认通过SSH西一管理主机

- 被管理主机要开启ssh服务，允许ansible主机登录

- 在托管节点上也需要安装Python2.5或以上的版本

- 如果托管节点上开启了SElinux,需要安装libselinux-python

## 安装ansible

ansible可以基于源码运行

### 源码安装

– pip,需要配置扩展软件包源extras

– git

yum install epel-release

yum install git python2-pip

– pip安装依赖模块

pip install paramiko PyYAML Jinja2 httplib2 six

ansible源码下载

– git clone git://github.com/ansible/ansible.git

– yum install python-setuptools python-devel

– python setup.py build

– python setup.py install

pip方式安装

– pip install ansible

### yum扩展源安装简单,自动解决依赖关系(推荐)

– http://mirror.centos.org/.../.../extras/

– yum install ansible

**安**装完成以后验证

– ansible --version

### 环境准备：

|  |  |
| --- | --- |
| 2G mem 2CPU 16G disk | |
| 1台ansible | 192.168.1.10 |
| 2台web | web1 192.168.1.11  web2 192.168.1.12 |
| 2台db | db1 192.168.1.21  db2 192.168.1.22 |
| 1 cache | 192.168.1.31 |
| yum源：  [CentOS]  name=CentOS  baseurl=ftp://192.168.1.254/CentOS-1807  gpgcheck=1  enabled=1  [ansible]  name=ansible  baseurl=ftp://192.168.1.254/ansible  gpgcheck=0  enabled=1 | **DNS本地解析文件：**  [root@ansible ~]# vim /etc/hosts  192.168.1.10 ansible  192.168.1.11 web1  192.168.1.12 web2  192.168.1.21 db1  192.168.1.22 db2  192.168.1.31 cache |

### 安装：

[root@ansible ~]# yum install ansible

[root@ansible ~]# ansible --version

# 主机管理

## 主机定义与分组

### 安装ansible之后可以做一些简单的任务

ansible配置文件查找顺序

– 首先检测ANSIBLE\_CONFIG变量定义的配置文件

– 其次检查当前目录下的 ./ansible.cfg 文件

– 再次检查当前用户家目录下 ~/ansible.cfg 文件

– 最后检查/etc/ansible/ansible.cfg文件

/etc/ansible/ansible.cfg是ansible的默认配置文件路径

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

[root@ansible ~]# grep -Pv "^(#|$)" /etc/ansible/ansible.cfg

[defaults]

inventory = /etc/ansible/hosts

roles\_path = /etc/ansible/roles:/usr/share/ansible/roles

host\_key\_checking = False

[inventory]

[privilege\_escalation]

[paramiko\_connection]

[ssh\_connection]

[persistent\_connection]

[accelerate]

[selinux]

[colors]

[diff]

### ansible.cfg 配置文件

– inventory定义托管主机地址配置文件

– 先编辑/etc/ansible/hosts文件,写入远程主机的地址。

### 格式

– # 表示注释

[组名称]

主机名称或ip地址,登录用户名,密码、端口等信息

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

[web]

web1

web2

[mysql]

db1

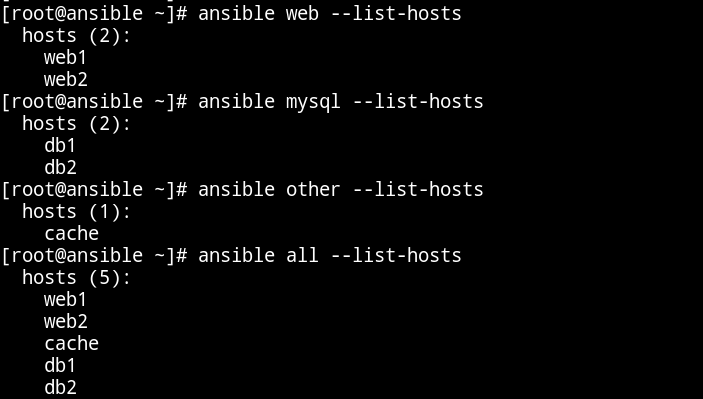
db2

[other]

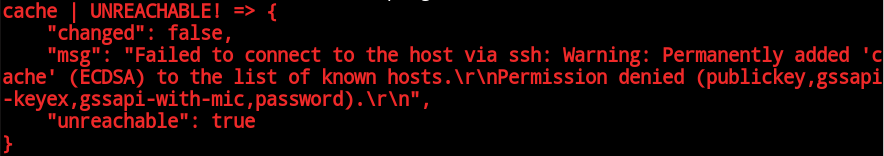
cache

### 测试

– ansible [组名称] --list-hosts



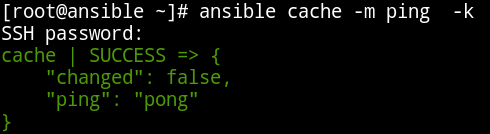
[root@ansible ~]# ansible cache -m ping //检测ansible模块中的ping模块



**检查的是ssh的连通性，与ping无关**

[root@ansible ~]# ansible cache -m ping -k

SSH password: //输入密码



禁用ping

[root@web1 ~]# sysctl -a | grep icmp\_echo\_ignore

[root@web1 ~]# sysctl -w net.ipv4.icmp\_echo\_ignore\_all=1

**启用ping**

[root@web1 ~]# sysctl -w net.ipv4.icmp\_echo\_ignore\_all=0

## 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别名不同,可以通过此变量设置

– ansible\_ssh\_port

– ssh端口号:如果不是默认的端口号,通过此变量设置

– 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(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,如:local,ssh或paramiko, 1.2以前默认使用paramiko,1.2以后默认使用'smart' ,它会根据是否支持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

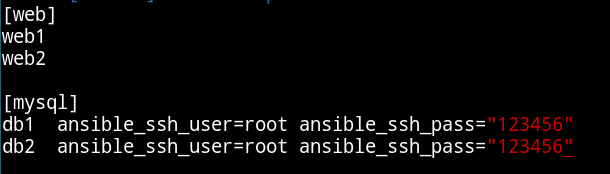
– 目标系统的shell类型,默认情况下,命令的执行使用'sh'语法,可设置为'csh'或'fish'

**– ansible\_python\_interpreter**

– 目标主机的python路径,适用情况:系统中有多个Python,或者命令路径不是"/usr/bin/python"

**示例：**

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

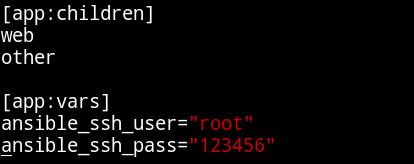


[root@ansible ~]# ansible web -m ping //测试失败

[root@ansible ~]# ansible mysql -m ping //测试成功

**示例：**

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



[root@ansible ~]# ansible app -m ping //测试成功

## 动态主机

### 无限可能

– ansible Inventory包含静态和动态的Inventory,静态Inventory指在文件/etc/ansible/hosts中指定的主机和组,动态Inventory指通过外部脚本获取主机列表,按照其要求格式返回给ansilbe命令

### Json

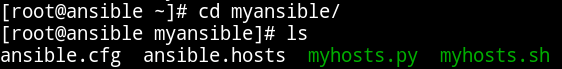
– JSON(JavaScript Object Notation,JavaScript对象表示法),一种基于文本独立于语言的轻量级数据交换格式

### 注意事项:

**– 主机部分必须是列表格式**

**– Hostdata行,其中的"hosts" 部分可以省略,但使用时,必须是"hosts"**

### 脚本输出主机列表



[root@ansible myansible]# cat ansible.cfg

[defaults]

inventory = myhosts.py

host\_key\_checking = False

[root@ansible myansible]# vim myhosts.py

#!/usr/bin/python

import json

hostlist = {}

hostlist["bb"] = ["web1", "db1","cache"]

hostlist["aa"] = {

"hosts" : ["web2", "db2"],

"vars" : {

"ansible\_ssh\_user":"root","ansible\_ssh\_pass":"123456"

}

}

print( json.dumps(hostlist))

[root@ansible myansible]# ansible all -m ping

**//bb集合中的主机会测试失败，aa集合中的主机成功**

# 批量执行

## ansible命令基础

### ansible <host-pattern> [options]

– host-pattern 主机或定义的分组

– -M 指定模块路径

– -m 使用模块,默认command模块

– -a or --args 模块参数

– -i inventory 文件路径,或可执行脚本

– -k 使用交互式登录密码

– -e 定义变量

– -v 详细信息,-vvvv开启debug模式

列出**要执行的主机**

– ansible all --list-hosts

批量检测主机

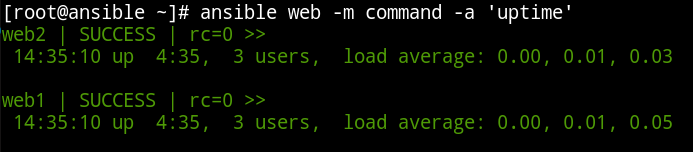
– ansible all -m ping

批量执行命令

– ansible all -m command -a 'id' -k

**在所有web组的主机上执行uptime命令查询负载：**

[root@ansible ~]# ansible web -m command -a 'uptime'



### 批量部署证书文件

• 每次交互输入密码太麻烦

• 密码写入配置文件安全性很差

• 不同主机不同密码

• 使用key方式认证

• 给所有主机部署公钥

ansible all -m authorized\_key -a "user=root exclusive=true

manage\_dir=true key='$(< /root/.ssh/authorized\_keys)'" -k -v

exclusive=true 强制覆盖

manage\_dir=true 目录不存在时自行创建

**生成公钥：**

[root@ansible ~]# cd /root/.ssh/

[root@ansible .ssh]# ssh-keygen -t rsa -b 2048 -N ''

**批量部署公钥：**

[root@ansible .ssh]# ansible all -m authorized\_key -a "user=root exclusive=true manage\_dir=true key='$(< /root/.ssh/id\_rsa.pub)'" -k

# 模块

## ansible-doc和ping模块

### • ansible-doc

– 模块的手册相当与shell的man,很重要

– ansible-doc -l 列出所有模块

– ansible-doc modulename 查看帮助

### • ping 模块

– 测试网络连通性, ping模块没有参数

– 注:测试ssh的连通性

ansible host-pattern -m ping

## command模块

### • command模块

**– 默认模块,远程执行命令**

**– 用法**

ansible host-pattern -m command -a '[args]'

**– 查看所有机器负载**

ansible all -m command -a 'uptime'

**– 查看日期和时间**

ansible all -m command -a 'date +%F\_%T'

### • command模块注意事项:

– 该模块通过-a跟上要执行的命令可以直接执行,若命令里有如下字符则执行不成功

– "<" , ">" , "|" , "&"

– 该模块不启动shell直接在ssh进程中执行,所有使用到shell的命令执行都会失败

– 下列命令执行会失败

ansible all -m command -a 'ps aux|grep ssh'

ansible all -m command -a 'set'

## shell|raw模块

### • shell | raw 模块

– shell 模块用法基本和command一样,区别是shell模块是通过/bin/sh进行执行命令,可以执行任意命令

– raw模块,用法和shell模块一样 ,可以执行任意命令

– 区别是raw没有chdir、creates、removes参数

### 示例：commond、shell、raw

[root@ansible ~]# ansible cache -m command -a "ps -ef | grep ssh "

**//执行失败**

[root@ansible ~]# ansible cache -m shell -a "ps -ef | grep ssh "

**//执行成功**

**使用模块给web1，db1添加用户z3，并设置密码**

[root@ansible ~]# ansible web1,db1 -m shell -a "useradd z3 && echo 123 | passwd --stdin z3" //使用shell模块

[root@ansible ~]# ansible web1,db1 -m raw -a "useradd z3 && echo 123 | passwd --stdin z3" //使用raw模块

**使用command模块会失败**

## script模块

### • script模块

– 在本地写脚本,然后使用script模块批量执行

ansible t1 -m script -a 'urscript'

– 注意:该脚本包含但不限于shell脚本,只要指定Sha-bang解释器的脚本都可运行

### 示例：给web组添加用户li4，要求li4与z3不能同时出现在同一台主机上

[root@ansible ~]# cat auser.sh

#!/bin/bash

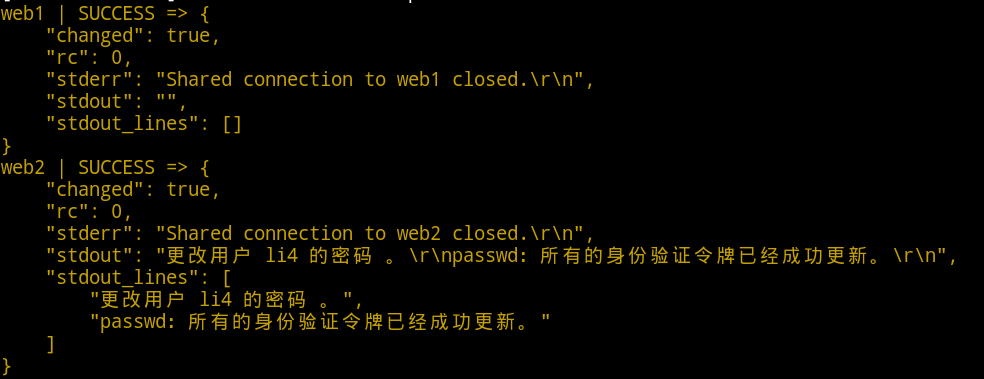
if ! $(id z3 &> /dev/null );then

useradd li4

echo 123 | passwd --stdin li4

fi

[root@ansible ~]# ansible web -m script -a 'auser.sh'



注：因为web1上之前创建了z3用户，所以命令不会在web1上执行，web则相反。

## copy模块

### • copy 模块

– 复制文件到远程主机

– src:复制本地文件到远程主机,绝对路径和相对路径都可,路径为目录时会递归复制。若路径以"/"结尾,只复制目录里的内容,若不以"/"结尾,则复制包含目录在内的整个内容,类似于rsync

– dest:必选项。远程主机的绝对路径,如果源文件是一个目录,那该路径必须是目录

– backup:覆盖前先备份原文件,备份文件包含时间信息。有两个选项:yes|no

– force:若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,设为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

– 复制文件

ansible t1 -m copy -a 'src=/root/alog dest=/root/a.log'

– 复制目录

ansible t1 -m copy -a 'src=urdir dest=/root/'

### 示例：

[root@ansible ~]# ansible all -m copy -a 'src=/etc/resolv.conf dest=/etc/resolv.conf' //将本地的DNS文件批量拷贝到其他主机

[root@ansible ~]# ansible all -m shell -a 'cat /etc/resolv.conf'

//批量查看

## lineinfile|replace模块

### • lineinfile | replace 模块

– 类似sed的一种行编辑替换模块

– path 目的文件

– regexp 正则表达式

– line 替换后的结果

ansible t1 -m lin einfile -a 'path="/etc/selinux/config"

regexp="^SELINUX=" line="SELINUX=disabled"'

– 替换指定字符

ansible t1 -m replace -a 'path="/etc/selinux/config"

regexp="^(SELINUX=).\*" replace="\1disabled"'

### 示例：

### lineinfile

[root@ansible ~]# ansible all -m lineinfile -a 'path="/etc/resolv.conf" regexp=".\*(\.114){3}" line="nameserver 8.8.8.8"'

[root@ansible ~]# ansible all -m shell -a 'cat /etc/resolv.conf'

### replace

[root@ansible ~]# ansible all -m replace -a 'path="/etc/resolv.conf" regexp="8.8$" replace="4.4"'

[root@ansible ~]# ansible all -m replace -a 'path=/etc/resolv.conf regexp="^(nameserver).\*" replace="\1 192.168.1.254"'

## yum模块

### • yum模块

– 使用yum包管理器来管理软件包

– config\_file:yum的配置文件

– disable\_gpg\_check:关闭gpg\_check

– disablerepo:不启用某个源

– enablerepo:启用某个源

– name:要进行操作的软件包名字,也可传递一个url或一个本地的rpm包的路径

– state:状态(present,absent,latest)

– 删除软件包

ansible t1 -m yum -a 'name="lrzsz" state=absent'

– 删除多个软件包

ansible t1 -m yum -a 'name="lrzsz,lftp" state=absent'

– 安装软件包

ansible t1 -m yum -a 'name="lrzsz"'

– 安装多个软件包

ansible t1 -m yum -a 'name="lrzsz,lftp"'

### 示例：

安装软件：

[root@ansible ~]# ansible all -m yum -a 'name=httpd state=present'

或者

[root@ansible ~]# ansible all -m yum -a 'name=httpd state=installed'

**移出软件：**

[root@ansible ~]# ansible all -m yum -a 'name=httpd state=absent'

或者

[root@ansible ~]# ansible all -m yum -a "name=httpd state=removed"

**查看软件是否安装**

[root@ansible ~]# ansible all -m shell -a 'rpm -qa httpd'

**注：若同时管理多个软件，name字段只需要用逗号隔开即可**

## service模块

### • service模块

– name:必选项,服务名称

– enabled:是否开机启动 yes|no

– sleep:执行restarted,会在stop和start之间沉睡几秒钟

– state:对当前服务执行启动,停止、重启、重新加载等操作(started,stopped,restarted,reloaded)

ansible t1 -m service -a 'name="sshd" enabled="yes" state="started"'

### 示例：

[root@ansible ~]# ansible all -m shell -a 'ss -natpul '

[root@ansible ~]# ansible all -m service -a 'name=chronyd state=stopped'

[root@ansible ~]# ansible all -m service -a 'name=chronyd enabled=yes state=started'

## setup模块

### • setup模块

– 主要用于获取主机信息,playbooks里经常会用的另一个参数gather\_facts与该模块相关,setup模块下经常用的是filter参数

– filter过滤所需信息

ansible t1 -m setup -a 'filter=ansible\_distribution'

### 示例：

[root@ansible ~]# ansible all -m setup -a 'filter=ansible\_distribution'

[root@ansible ~]# ansible web1 -m setup //查看系统相关的所有信息

### 综合练习：

### 1. 安装Apache并修改监听端口为8080

2. 修改ServerName配置,执行apachectl -t命令不报错

3. 设置默认主页hello world

4. 启动服务并设开机自启

[root@ansible ~]# ansible all -m yum -a 'name="httpd" state="installed"'

[root@ansible ~]# ansible all -m replace -a 'path=/etc/httpd/conf/httpd.conf regexp="^(Listen).\*" replace="\1 8080"'

[root@ansible ~]# ansible all -m lineinfile -a 'path=/etc/httpd/conf/httpd.conf regexp="^(#ServerName).\*" line="ServerName [www.example.com:8080"'](http://www.example.com:8080\"')

[root@ansible ~]# ansible all -m shell -a 'echo hello world > /var/www/html/index.html'

[root@ansible ~]# ansible all -m service -a 'name=httpd state=started enabled=yes'

[root@ansible ~]# ansible all -m shell -a 'apachectl -t'

[root@ansible ~]# curl <http://192.168.1.11:8080>