大数据环境下海量服务器如何运维

谷歌、Facebook 等大厂一个运维人员管理的服务器在上万台左右，这么多的服务器使用手工的方法去维护是很难做到的，那么他们是怎么运维这么多台服务器的呢？真相只有一个：使用自动化运维工具。大数据运维同样也是如此。

目前主流的自动化运维管理工具有 Puppet、Saltstack、Ansible 等，它们各有优缺点，这里我们选用 Ansible 作为大数据运维平台的自动化运维工具。

Ansible 是基于 Python 语言开发的，只需要在一台普通的服务器上运行即可，不需要在客户端服务器上安装客户端。因为 Ansible 是基于 SSH 远程管理，而 Linux 服务器基本都开启了 SSH 服务，所以 Ansible 不需要为配置工作添加额外的支持。

Ansible 命令行模式的使用

Ansible 执行自动化任务，分为以下两种执行模式：

（1）ad-hoc（单个模块），单条命令的批量执行，或者叫命令行模式；

（2）playbook，为面向对象的编程，可以把多个想要执行的任务放到一个 playbook 中，当然多个任务在事物逻辑上最好是有上下关联的，通过多个任务可以完成一个总体的目标。

命令行模式一般用于测试、临时应用等场景，而 playbook 方式，主用用于正式环境，通过编写 playbook 文件，可实现固定的、批量的对系统或服务进行配置以及维护工作。

本课时将从头讲 Ansible，使用时需要注意两个概念：管理机和远程主机。管理机是安装 Ansible 的机器，远程主机是 Ansible 批量操作的对象，可以是一个或一组主机。Ansible 通过管理机发出批量操作远程主机的指令，这些指令在每个远程主机上依次执行。

1. Ansible 的执行流程与配置文件

Ansible 的安装非常简单，执行如下命令即可：

复制[root@master ~]# yum install epel-release

然后即可通过 yum 工具安装 Ansible：

复制[root@master ~]# yum install ansible

安装好的 Ansible 配置文件位于 /etc/ansible 目录下，需要重点关注的有 ansible.cfg、hosts 文件。

（1）hosts 文件（以下 hosts 文件均指 /etc/ansible/hosts 文件）

该文件用来定义 Ansible 批量操作的主机列表，主机列表有多种书写方式，最简单的格式如下：

复制[webservers]
ixdba1.net
ixdba2.net

[dbservers]
db.ixdba1.net
db.ixdba2.net
中括号中的名字代表组名，可以根据需求将庞大的主机分成具有标识的组。比如上面分了两个组 webservers 和 dbservers 组。

主机（hosts）部分可以使用域名、主机名、IP 地址表示；当然使用前两者时，需要主机能反解析到相应的 IP 地址，一般此类配置中多使用 IP 地址；未分组的机器需保留在 hosts 的顶部。

也可在 hosts 文件中，指定主机的范围，示例如下：

复制[web]
www[01:50].ixdba.net
[db]
db[a:f].ixdba.ent
这个配置中，web 主机组的主机为 www01.ixdba.net、www02.ixdba.net、www03.ixdba.net 等以此类推，一直到 www50.ixdba.net。下面的 db 组中的 a:f 表示从 a 到 f 的字符。

在 hosts 文件中，还可以使用变量，变量分为主机变量和组变量两种类型，常用的变量如下表所示：

例如，在 hosts 中可以这么使用变量：

复制[test]
192.168.1.1 ansible_ssh_user=root ansible_ssh_pass='abc123'
192.168.1.2 ansible_ssh_user=breeze ansible_ssh_pass='123456'
（2）ansible.cfg 文件

此文件定义了 Ansible 主机的默认配置熟悉，比如默认是否需要输入密码、是否开启 sudo 认证、action_plugins 插件的位置、hosts 主机组的位置、是否开启 log 功能、默认端口、key 文件位置等。一般情况下这个文件无需修改，保存默认即可。

注意：host_key_checking 表示是否关闭第一次使用 Ansible 连接客户端时 yes/no 的连接确认提示，False 表示关闭，我们只需要去掉此选项的注释即可。这个问题其实是 SSH 连接的问题，因为 Linux 下的主机在第一次 SSH 连接到一个新的主机时，一般会需要 yes/no 的连接确认，这在自动化运维中是不需要的，因此需要禁止这种确认。在 Ansible 中通过设置 host_key_checking 为 False 就可以避免这种情况。

2. commands 模块
命令行下执行 ansible，基本格式如下：

复制ansible 主机或组 -m 模块名 -a '模块参数' ansible参数
其中：

主机或组，在 /etc/ansible/hosts 里进行指定；
模块名，可以通过 ansible-doc -l 查看目前安装的模块，默认不指定时，使用的是 command 模块；
模块参数，可以通过 “ansible-doc 模块名”查看具体用法。
ansible 常用的参数如下表所示：

下面看几个使用 command 模块的例子：

复制ansible 172.16.213.157 -m command -a 'pwd'
ansible 172.16.213.157 -m command -a 'chdir=/tmp/ pwd'
ansible 172.16.213.157 -m command -a 'chdir=/var/www tar zcvf /data/html.tar.gz html'

ansible 172.16.213.157 -m command -a 'creates=/tmp/tmp.txt date'
ansible 172.16.213.157 -m command -a 'removes=/tmp/tmp.txt date'
ansible 172.16.213.157 -m command -a 'ps -ef|grep sshd'（此命令会执行失败）
上面的例子是对主机 172.16.213.157 进行的操作，在实际应用中需要替换为主机组。另外，还用到了 command 模块的几个选项：

creates，后跟一个文件名，当远程主机上存在这个文件时，该命令不执行，否则执行；
chdir，在执行指令之前，先切换到该指定的目录；
removes，后跟一个文件名，当该文件存在时，该选项执行，否则不执行。
注意：commands 模块的执行，在远程主机上，需有 Python 环境的支持。该模块通过在 -a 参数后面跟上要在远程机器上执行的命令即可完成远程操作，不过命令里如果带有特殊字符（“<”、“>”、“|”、“&”等），则执行不成功，也就是 commands 模块不支持这些特殊字符。上面最后那个例子无法执行成功就是这个原因。

3. shell 模块
shell 模块的功能和用法与 command 模块一样，不过 shell 模块执行命令的时候使用的是/bin/sh，该模块可以执行任何命令。看下面几个例子：

复制ansible 172.16.213.233 -m shell -a 'ps -ef|grep sshd'（此命令可执行成功）
ansible 172.16.213.233 -m shell -a 'sh /tmp/install.sh >/tmp/install.log'
最后这个例子是执行远程机器上的脚本，其路径为 /tmp/install.sh（远程主机上的脚本，非本机的），然后将执行命令的结果存放在远程主机路径 /tmp/install.log 中，注意在进行保存文件的时候，写上全路径，否则就会保存在登录之后的默认路径中。官方文档表示 command 用起来更安全，更有可预知性，但从我使用角度来说，并没发现有多大差别。

4. raw 模块和 script 模块
raw 模块功能与 command 和 shell 模块类似，shell 能够完成的操作，raw 也都能完成。不同的是，raw 模块不需要远程主机上的 Python 环境。

Ansible 要执行自动化操作，需在管理机上安装 Ansible，客户机上安装 Python，如果客户机上没有安装，那么 command、shell 模块将无法工作，但 raw 可以正常工作。因此，若有的机器没有装 Python，或者装的版本在 2.4 以下，就可以使用 raw 模块来装 Python、python-simplejson 等。

若有些机器根本安装不了 Python 的话（如交换机、路由器等），那么，直接用 raw 模块是最好的选择。下面看几个例子：

复制[root@localhost ansible]#ansible 172.16.213.107 -m raw -a "ps -ef|grep sshd|awk '{print \$2}'"
[root@localhost ansible]#ansible 172.16.213.107 -m raw -a "yum -y install python26" -k
script 模块是将管理端的 shell 脚本拷贝到被管理的远程主机上执行，其原理是先将 shell 复制到远程主机，再在远程主机上执行。此模块的执行，不需要远程主机上的 Python 环境。看下面这个例子：

复制[root@localhost ansible]# ansible 172.16.213.233 -m script -a 'sh /mnt/install1.sh >/tmp/install1.log'
脚本 /tmp/install1.sh 在管理端本机上，script 模块执行的时候将脚本传送到远程的 172.16.213.233 主机中，然后执行这个脚本，同时，将执行的输出日志文件保存在远程主机对应的路径 /tmp/install.log 下，这里保存日志文件的时候，最好用全路径。

5. file 模块、copy 模块与 synchronize 模块
file 模块功能强大，主要用于远程主机上的文件或目录操作，该模块包含如下选项：

下面来看几个使用示例。
（1）创建一个不存在的目录，并进行递归授权：

复制[root@localhost ansible]# ansible 172.16.213.233 -m file -a "path=/mnt/abc123 state=directory"
[root@localhost ansible]# ansible 172.16.213.233 -m file -a "path=/mnt/abc123 owner=nobody group=nobody mode=0644 recurse=yes"
[root@localhost ansible]# ansible 172.16.213.233 -m file -a "path=/mnt/ansibletemp owner=sshd group=sshd mode=0644 state=directory "
（2）创建一个文件（如果不存在），并进行授权：

复制[root@localhost ansible]# ansible 172.16.213.233 -m file -a "path=/mnt/syncfile.txt mode=0444"

（3）创建一个软连接（将 /etc/ssh/sshd_config 软连接到 /mnt/sshd_config）：

复制[root@localhost ansible]#ansible 172.16.213.233 -m file -a "src=/etc/ssh/sshd_config dest=/mnt/sshd_config owner=sshd state=link"

（4）删除一个压缩文件：

复制[root@localhost ansible]#ansible 172.16.213.233 -m file -a "path=/tmp/backup.tar.gz state=absent"

（5）创建一个文件：

复制[root@localhost ansible]#ansible 172.16.213.233 -m file -a "path=/mnt/ansibletemp state=touch"

接着继续来看 copy 模块，此模块用来复制文件到远程主机，copy 模块包含的选项如下表所示：

下面是几个例子。

（1）拷贝文件并进行权限设置。

复制[root@localhost ansible]#ansible 172.16.213.233 -m copy -a 'src=/etc/sudoers dest=/mnt/sudoers owner=root group=root mode=440 backup=yes'

copy 默认会对存在的备份文件进行覆盖，通过 backup=yes 参数可以在覆盖前，对之前的文件进行自动备份。

（2）拷贝文件之后进行验证。

复制[root@localhost ansible]#ansible 172.16.213.233 -m copy -a "src=/etc/sudoers dest=/mnt/sudoers validate='visudo -cf %s'"

这里用了 validate 参数，表示在复制之前验证要拷贝的文件是否正确。如果验证通过则复制到远程主机上，%s 是一个文件路径的占位符，在文件被复制到远程主机之前，它会被替换为 src 后面的文件。

（3）拷贝目录并进行递归设定目录的权限。

复制[root@localhost ansible]#ansible 172.16.213.233 -m copy -a 'src=/etc/yum dest=/mnt/ owner=hadoop group=hadoop directory_mode=644'
[root@localhost ansible]#ansible 172.16.213.233 -m copy -a 'src=/etc/yum/ dest=/mnt/bak owner=hadoop group=hadoop directory_mode=644'
上面这两个命令执行是有区别的，第一个是拷贝管理机的 /etc/yum 目录到远程主机的 /mnt 目录下；第二个命令是拷贝管理机 /etc/yum 目录下的所有文件或子目录到远程主机的 /mnt/bak 目录下。

copy 模块拷贝小文件还可以，如果拷贝大文件或者目录的话，速度很慢，不建议使用。此时推荐使用 synchronize 模块，此模块通过调用 rsync 进行文件或目录同步，同步速度很快，还指出增量同步，该模块常用的选项如下表所示：

下面看几个例子。

（1）同步本地的 /mnt/rpm 到远程主机

复制172.16.213.77 的 /tmp 目录下。
ansible 172.16.213.77 -m synchronize -a 'src=/mnt/rpm dest=/tmp'
（2）将远程主机 172.16.213.77 上 /mnt/a 文件拷贝到本地的 /tmp 目录下。

复制ansible 172.16.213.77 -m synchronize -a 'mode=pull src=/mnt/a dest=/tmp'
6. cron 模块、yum 模块与 service 模块
cron 模块用于管理计划任务，常用选项含义如下表所示：

下面是几个示例。

（1）系统重启时执行 /data/bootservice.sh 脚本。

复制ansible 172.16.213.233 -m cron -a 'name="job for reboot"
special_time=reboot job="/data/bootservice.sh"'

此命令执行后，会在 172.16.213.233 的 crontab 中写入“@reboot /data/bootservice.sh”，通过“crontab -l ”可以查看到。

（2）表示在每周六的 1:20 分执行"yum -y update"操作。

复制ansible 172.16.213.233 -m cron -a 'name="yum autoupdate" weekday="6" minute=20 hour=1 user="root" job="yum -y update"'
（3）表示在每周六的 1:30 分以 root 用户执行 "/home/ixdba/backup.sh" 脚本。

复制ansible 172.16.213.233 -m cron -a 'backup="True" name="autobackup" weekday="6" minute=30 hour=1 user="root" job="/home/ixdba/backup.sh"'
（4）会在 /etc/cron.d 创建一个 check_http_for_ansible 文件，表示每天的 12：30 分通过 root 用户执行 /home/ixdba/check_http.sh 脚本。

复制ansible 172.16.213.233 -m cron -a 'name="checkhttp" minute=30 hour=12 user="root" job="/home/ixdba/check_http.sh" cron_file="check_http_for_ansible" '
（5）删除一个计划任务。

复制ansible 172.16.213.233 -m cron -a 'name="yum update" state=absent'
接着，再看看 yum 模块的使用，此模块用来通过 yum 包管理器来管理软件包，常用选项以及含义如下表所示：

下面是几个示例。

（1）通过 yum 安装 Redis。

复制ansible 172.16.213.77 -m yum -a "name=redis state=installed"
（2）通过 yum 卸载 Redis。

复制ansible 172.16.213.77 -m yum -a "name=redis state=removed"
（3）通过 yum 安装 Redis 最新版本，并设置 yum 源。

复制ansible 172.16.213.77 -m yum -a "name=redis state=latest enablerepo=epel"

（4）通过指定地址的方式安装 bash。

复制ansible 172.16.213.78 -m yum -a
"name=http://mirrors.aliyun.com/centos/7.4.1708/os/x86_64/Packages/bash-4.2.46-28.el7.x86_64.rpm" state=present'
最后看看 service 模块，此模块用于管理远程主机上的服务，该模块包含如下选项：

下面是几个使用示例。

（1）启动 httpd 服务。

复制ansible 172.16.213.233 -m service -a "name=httpd state=started"
（2）设置 httpd 服务开机自启。

复制ansible 172.16.213.233 -m service -a "name=httpd enabled=yes"
7. setup 模块获取 Ansible facts 信息
Ansible facts 是远程主机上的系统信息，主要包含 IP 地址、操作系统版本、网络设备、Mac 地址、内存、磁盘、硬件等信息，这些信息根据远程主机的信息来作为执行条件操作的场景，非常有用。比如，我们可以根据远程主机的操作系统版本，选择安装不同版本的软件包，或者收集远程主机上每个主机的主机名、IP 地址等信息。

那么如何获取 Ansible facts 信息呢，其实，Ansible 提供了一个 setup 模块来收集远程主机的系统信息，这些 facts 信息可以直接以变量的形式使用。

下面是两个使用的例子。

（1）查看主机内存信息。

复制[root@localhost ~]# ansible 172.16.213.77 -m setup -a 'filter=ansible_*_mb'
（2）查看接口为 eth0-2 的网卡信息。

复制[root@localhost ~]# ansible 172.16.213.77 -m setup -a 'filter=ansible_em[1-2]'

在后面 ansible-playbook 内容中会讲到的 playbooks 脚本中，经常会用到一个参数 gather_facts，其与该模块相关。gather_facts 默认值为 yes，也就是说，在使用 Ansible 对远程主机执行任何一个 playbook 之前，总会先通过 setup 模块获取 facts，并将信息暂存在内存中，直到该 playbook 执行结束为止。

8. user 模块与 group 模块

user 模块请求的是 useradd、userdel、usermod 三个指令；group 模块请求的是 groupadd、groupdel、groupmod 三个指令，常用的选项如下表所示：

下面看几个使用例子。

（1）创建一个用户 usertest1。

复制ansible 172.16.213.77 -m user -a "name=usertest1"

（2）创建用户 usertest2，并设置附加组。

复制ansible 172.16.213.77 -m user -a "name=usertest2 groups=admins,developers"

（3）删除用户 usertest1 的同时，删除用户根目录。

复制ansible 172.16.213.77 -m user -a "name=usertest1 state=absent remove=yes"

（4）批量修改用户密码。

复制[root@localhost ~]# echo "linux123www" | openssl passwd -1 -salt $(< /dev/urandom tr -dc '[:alnum:]' | head -c 32) -stdin
1yjJ74Wid$xOQUaaHzA8EwWU2kG6SRB1
[root@localhost ~]# ansible 172.16.213.77 -m user -a 'name=usertest2 password="1yjJ74Wid$xOQUaaHzA8EwWU2kG6SRB1"'

其中：

-1 表示采用的是 MD5 加密算法；

-salt 指定 salt 值，在使用加密算法进行加密时，即使密码一样，由于 salt 不一样，所以计算出来的 hash 值也不一样，除非密码一样，salt 值也一样，计算出来的 hash 值才一样；

“< /dev/urandom tr -dc ’[:alnum:]’ | head -c 32”产生一个随机的 salt；
passwd 的值不能是明文，passwd 关键字后面应该是密文，密文会被保存在 /etc/shadow 文件中。