链滴

ansible 的 inventory

作者：SmiteLi
原文链接：https://ld246.com/article/1567415549588
来源网站：链滴

一、work with inventory

Ansible可同时处理基础架构中的多个系统。它通过选择Ansible库存中列出的系统部分来实现这一点该系统默认保存在该位置/etc/ansible/hosts。您可以使用命令行上的选项指定其他清单文件。-i <path>

此库存不仅可配置，而且您还可以同时使用多个库存文件，并从动态或云源或不同格式（YAML，ini）中提取库存，如使用动态库存中所述。Ansible在2.4版本中引入了库存插件，使其具有灵活性和可制性。

库存基础：主机和组

清单文件可以采用多种格式之一，具体取决于您拥有的库存插件。对于此示例，格式为/etc/ansible/hosts类似INI（Ansible的默认值之一），如下所示：

```
mail.example.com

[webservers]
foo.example.com
bar.example.com

[dbservers]
one.example.com
two.example.com
three.example.com
```

括号中的标题是组名，用于分类系统并决定您在什么时间和目的控制哪些系统。

YAML版本看起来像：

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
```

多个组中的主机

您可以将系统放在多个组中，例如，服务器既可以是Web服务器，也可以是特定数据中心。例如，您以创建跟踪的组：

什么 - 应用程序，堆栈或微服务。（例如，数据库服务器，Web服务器等）。

Where - 数据中心或区域，与本地DNS，存储等通信（例如，东，西）。

何时 - 开发阶段，避免测试生产资源。（例如，prod，test）。

扩展之前的YAML库存以包括什么，何时和何处：

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
    east:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    west:
      hosts:
        bar.example.com:
        three.example.com:
    prod:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
    test:
      hosts:
        bar.example.com:
        three.example.com:
```

你可以看到，one.example.com在存在着dbservers，east和prod团体。

你也可以使用嵌套组来简化prod并test在此盘点，对于相同的结果：

```
all:
  hosts:
    mail.example.com:
  children:
    webservers:
      hosts:
        foo.example.com:
        bar.example.com:
    dbservers:
      hosts:
        one.example.com:
        two.example.com:
        three.example.com:
    east:
      hosts:
        foo.example.com:
        one.example.com:
        two.example.com:
```

```
  west:
    hosts:
      bar.example.com:
      three.example.com:
  prod:
    children:
      east:
  test:
    children:
      west:
```

如果您确实有多个组中的系统，请注意变量将来自它们所属的所有组。变量优先级在变量优先级中详说明：我应该在哪里放置变量？。

主机和非标准端口

如果您拥有在非标准SSH端口上运行的主机，则可以在主机名后面添加带冒号的端口号。SSH配置文中列出的端口不会与paramiko连接一起使用，但会与openssh连接一起使用。

为了使事情明确，建议您在默认端口上没有运行时设置它们：

```
badwolf.example.com:5309
```

假设您只有静态IP，并且想要设置一些存在于主机文件中的别名，或者您正在通过隧道连接。您还可通过变量描述主机：

在INI：

```
jumper ansible_port=5555 ansible_host=192.0.2.50
```

在YAML：

```
...
  hosts:
    jumper:
      ansible_port: 5555
      ansible_host: 192.0.2.50
```

在上面的示例中，尝试对主机别名“jumper”（可能甚至不是真正的主机名）进行回复，将在端口555上联系192.0.2.50。请注意，这是使用库存文件的一个功能来定义一些特殊变量。一般来说，这不定义描述系统策略的变量的最佳方式，但我们将在稍后分享建议。

注意

使用key=value语法以INI格式传递的值根据声明的位置进行不同的解释。*当与主机内联声明时，INI被解释为Python文字结构（字符串，数字，元组，列表，dicts，布尔值，无）。主机线key=value行接受多个参数。因此，他们需要一种方法来指示空间是值的一部分而不是分隔符。*在:vars节中声时，INI值被解释为字符串。例如，var=FALSE将创建一个等于'FALSE'的字符串。与主机行不同，:var段每行只接受一个条目，所以后面的所有内容都是=必须是条目的值。*不要依赖于定义期间设置的类，在使用变量时，请务必确保在需要时使用过滤器指定类型。*考虑将YAML格式用于库存源，以避免淆变量的实际类型。YAML清单插件一致且正确地处理变量值。

如果要添加大量遵循类似模式的主机，则可以执行此操作而不是列出每个主机名：

在INI：

```
[webservers]
www[01:50].example.com
```

在YAML：

```
...
  webservers:
    hosts:
      www[01:50].example.com:
```

对于数字模式，可以根据需要包含或删除前导零。范围包括在内。您还可以定义字母范围：

```
[databases]
db-[a:f].example.com
```

您还可以基于每个主机选择连接类型和用户：

```
[targets]
localhost              ansible_connection=local
other1.example.com     ansible_connection=ssh       ansible_user=mpdehaan
other2.example.com     ansible_connection=ssh       ansible_user=mdehaan
```

如上所述，在库存文件中设置这些只是一种简写，稍后我们将讨论如何将它们存储在'host_vars'目录的单个文件中。

将变量分配给一台机器：主机变量

如上所述，很容易将变量分配给稍后将在playbooks中使用的主机：

```
[atlanta]
host1 http_port=80 maxRequestsPerChild=808
host2 http_port=303 maxRequestsPerChild=909
```

YAML版本：

```
atlanta:
  host1:
    http_port: 80
    maxRequestsPerChild: 808
  host2:
    http_port: 303
    maxRequestsPerChild: 909
```

为多台机器分配变量：组变量

变量也可以同时应用于整个组：

INI方式：

```
[atlanta]
host1
host2

[atlanta:vars]
ntp_server=ntp.atlanta.example.com
proxy=proxy.atlanta.example.com
```

YAML版本：

```
atlanta:
  hosts:
    host1:
    host2:
  vars:
    ntp_server: ntp.atlanta.example.com
    proxy: proxy.atlanta.example.com
```

请注意，这只是一次将变量应用于多个主机的便捷方式; 即使您可以按组定位主机，在执行播放之前变量始终会展平到主机级别。

继承变量值：组的组变量

您可以使用:childrenINI中的后缀或children:YAML中的条目来创建组。您可以使用:vars或将变量应于这些组的组vars::

```
[atlanta]
host1
host2

[raleigh]
host2
host3

[southeast:children]
atlanta
raleigh

[southeast:vars]
some_server=foo.southeast.example.com
halon_system_timeout=30
self_destruct_countdown=60
escape_pods=2

[usa:children]
southeast
northeast
southwest
northwest
all:
  children:
    usa:
      children:
        southeast:
          children:
            atlanta:
              hosts:
                host1:
                host2:
            raleigh:
              hosts:
                host2:
                host3:
```

```
  vars:
    some_server: foo.southeast.example.com
    halon_system_timeout: 30
    self_destruct_countdown: 60
    escape_pods: 2
northeast:
northwest:
southwest:
```

如果您需要存储列表或哈希数据，或者希望将主机和组特定变量与库存文件分开，请参阅下一节。儿团体有几个要注意的属性：

作为子组成员的任何主机都自动成为父组的成员。

子组的变量将具有更高的优先级（覆盖）父组的变量。

群组可以有多个父母和孩子，但不是循环关系。

主机也可以在多个组，但只会有一个主机的情况下，合并来自多个组的数据。

默认组

有两个默认组：all和ungrouped。all包含每个主机。 ungrouped包含除了之外没有其他组的所有主all。每个主机始终属于至少2个组。虽然all并且ungrouped始终存在，但它们可以是隐含的，而不会现在组列表中group_names。

组织主机和组变量

虽然您可以将变量存储在主库存文件中，但存储单独的主机和组变量文件可以帮助您更轻松地跟踪变值。

主机和组变量可以存储在相对于库存文件的单个文件中（不是目录，它始终是文件）。

这些变量文件采用YAML格式。有效的文件扩展名包括“.yml”， “。yaml”， “。json”或没有文扩展名。如果您是YAML的新手，请参阅YAML语法。

比方说，您可以将库存文件保存在/etc/ansible/hosts。你有一个名为'foosball'的主持人，它是两个组的成员：'raleigh'和'webservers'。该主机将在以下位置使用YAML文件中的变量：

```
/etc/ansible/group_vars/raleigh # can optionally end in '.yml', '.yaml', or '.json'
/etc/ansible/group_vars/webservers
/etc/ansible/host_vars/foosball
```

例如，假设您有按数据中心分组的主机，并且每个数据中心使用一些不同的服务器。'raleigh'组的组件'/ etc / ansible / group_vars / raleigh'中的数据可能如下所示：

---

ntp_server: acme.example.org

database_server: storage.example.org

如果这些文件不存在，这是可以的，因为这是一个可选功能。

作为高级用例，您可以创建以组或主机命名的目录，Ansible将按字典顺序读取这些目录中的所有文。“raleigh”组的一个例子：

/etc/ansible/group_vars/raleigh/db_settings

/etc/ansible/group_vars/raleigh/cluster_settings

“raleigh”组中的所有主机都将具有这些文件中定义的变量。当单个文件开始太大或者想要在组的变的一部分上使用Ansible Vault时，这对于保持变量有条理非常有用。

提示：group_vars/和host_vars/目录可以存在于playbook目录或inventory目录中。如果两个路径存在，则playbook目录中的变量将覆盖inventory目录中设置的变量。

提示：该ansible-playbook命令默认查找当前工作目录中的playbooks。其他Ansible命令（例如ansile，ansible-console等）将只查找group_vars/和host_vars/在库存目录中，除非你提供--playbook-ir的命令行选项。

提示：将库存文件和变量保存在git仓库（或其他版本控制）中是跟踪库存和主机变量更改的绝佳方法。

如何合并变量

默认情况下，在运行播放之前将变量合并/展平到特定主机。这使得Ansible专注于主机和任务，因此组在库存和主机匹配之外并不存在。默认情况下，Ansible会覆盖变量，包括为组和/或主机定义的变（请参阅DEFAULT_HASH_BEHAVIOUR）。顺序/优先级是（从最低到最高）：

所有组（因为它是所有其他组的“父”）

父母组

儿童组

主办

当合并相同父/子级别的组时，它将按字母顺序完成，并且加载的最后一个组将覆盖先前的组。例如，_group将与b_group合并，b_group变量匹配将覆盖a_group中的变量。

版本2.4中的新功能。

从Ansible 2.4版开始，用户可以使用组变量ansible_group_priority来更改同级别组的合并顺序（在析父/子顺序之后）。数字越大，合并越晚，优先级越高。1如果未设置，此变量默认为。例如：

a_group:

testvar: a

ansible_group_priority: 10

b_group：

testvar: b

在这个例子中，如果两个组具有相同的优先级，结果通常是，但由于我们给出了更高的优先级，结果是。testvar == ba_grouptestvar == a

注意

ansible_group_priority 只能在库存源中设置，而不能在group_vars /中设置，因为变量用于加载group_vars。

使用多个库存来源

作为高级用例，您可以通过从命令行提供多个库存参数或通过配置来同时定位多个库存源（目录，动库存脚本或库存插件支持的文件） ANSIBLE_INVENTORY。当您希望针对特定操作同时针对正常分的环境（如分段和生产）时，此功能非常有用。

从命令行定位两个源，如下所示：

ansible-playbook get_logs.yml -i staging -i production

请记住，如果库存中存在可变冲突，则根据变量如何合并和变量优先级中描述的规则来解决它们：我该在哪里放置变量？。合并顺序由库存源参数的顺序控制。如果[all:vars]在暂存库存中定义，但生产存定义，则将运行该手册。如果播放剧本，结果将被颠倒 。myvar = 1myvar = 2myvar = 2-i production -i staging

使用目录聚合库存源

您还可以通过在目录下组合多个库存源和源类型来创建库存。这对于组合静态和动态主机以及将它们为一个库存进行管理非常有用。以下清单将清单插件源，动态清单脚本和文件与静态主机组合在一起：

inventory/

openstack.yml # configure inventory plugin to get hosts from Openstack cloud

dynamic-inventory.py # add additional hosts with dynamic inventory script

static-inventory # add static hosts and groups

group_vars/

all.yml # assign variables to all hosts

您可以像这样定位此库存目录：

ansible-playbook example.yml -i inventory

如果存在可变冲突或组与其他库存源的依赖关系，则控制库存源的合并顺序会很有用。库存根据文件按字母顺序合并，因此可以通过向文件添加前缀来控制结果：

inventory/

01-openstack.yml # configure inventory plugin to get hosts from Openstack cloud

02-dynamic-inventory.py # add additional hosts with dynamic inventory script

03-static-inventory # add static hosts

group_vars/

all.yml # assign variables to all hosts

如果01-openstack.yml定义了组，定义和定义，则将运行该剧本。myvar = 1all02-dynamic-inventory.pymyvar = 203-static-inventorymyvar = 3myvar = 3

有关库存插件和动态库存脚本的更多详细信息，请参阅库存插件和使用动态库存。

连接到主机：行为库存参数

如上所述，设置以下变量可控制Ansible与远程主机的交互方式。

主机连接：

注意

Ansible不会公开一个通道，允许用户和ssh进程之间的通信在使用ssh连接插件时手动接受密码来解密sh密钥（这是默认设置）。使用的ssh-agent强烈建议。

ansible_connection

连接类型到主机。这可以是任何ansible连接插件的名称。SSH协议类型是smart，ssh或paramiko。认是智能的。基于非SSH的类型将在下一节中介绍。

一般所有连接：

ansible_host

要连接的主机的名称，如果与您希望提供给它的别名不同。

ansible_port

连接端口号，如果不是默认值（ssh为22）

ansible_user

连接到主机时要使用的用户名

ansible_password

用于向主机进行身份验证的密码（从不以纯文本格式存储此变量;始终使用保管库。请参阅变量和保管）

特定于SSH连接：

ansible_ssh_private_key_file

ssh使用的私钥文件。如果使用多个密钥并且您不想使用SSH代理，则很有用。

ansible_ssh_common_args

此设置始终附加到sftp，scp和ssh的默认命令行。ProxyCommand用于为特定主机（或组）配置a 。

ansible_sftp_extra_args

此设置始终附加到默认的sftp命令行。

ansible_scp_extra_args

此设置始终附加到默认scp命令行。

ansible_ssh_extra_args

此设置始终附加到默认的ssh命令行。

ansible_ssh_pipelining

确定是否使用SSH流水线。这可以覆盖中的pipelining设置ansible.cfg。

ansible_ssh_executable（在2.2版中添加）

此设置将覆盖使用系统ssh的默认行为。这可以覆盖中的ssh_executable设置ansible.cfg。

特权升级（详见Ansible Privilege Escalation）：

ansible_become

等同于ansible_sudo或ansible_su允许强制权限升级

ansible_become_method

允许设置权限提升方法

ansible_become_user

等同于ansible_sudo_user或ansible_su_user允许通过权限提升设置您成为的用户

ansible_become_password

等效于ansible_sudo_password或ansible_su_password允许您设置权限提升密码（永远不要将此变存储为纯文本;始终使用保管库。请参阅变量和保管库）

ansible_become_exe

等效于ansible_sudo_exe或ansible_su_exe允许您为所选的升级方法设置可执行文件

ansible_become_flags

等效于ansible_sudo_flags或ansible_su_flags允许您设置传递给选定升级方法的标志。这也可以ansible.cfg在sudo_flags选项中全局设置

远程主机环境参数：

ansible_shell_type

目标系统的shell类型。除非已将ansible_shell_executable设置为非Bourne（sh）兼容shell，否则应使用此设置 。默认情况下，命令使用sh-style语法格式化。将此设置为csh或fish将导致在目标系统执行的命令遵循这些shell的语法。

ansible_python_interpreter

目标主机python路径。这对于具有多个Python或不位于/ usr / bin / python的系统（如* BSD）或/ sr / bin / python 不是2.X系列Python的系统非常有用。我们不使用/ usr / bin / env机制，因为它要设置远程用户的路径，并假设python可执行文件名为python，其中可执行文件可能被命名为pytho 2.6。

ansible _ * _解释

适用于任何东西，如ruby或perl，就像ansible_python_interpreter一样。这取代了将在该主机上运的模块。

版本2.1中的新功能。

ansible_shell_executable

这将设置ansible控制器将目标机器上使用的壳，将覆盖executable在ansible.cfg其中默认为 / bin / s的。如果无法使用/ bin / sh（即/ bin / sh未安装在目标计算机上或无法从sudo运行），您应该只更它。

Ansible-INI主机文件中的示例：

some_host ansible_port=2222 ansible_user=manager

aws_host ansible_ssh_private_key_file=/home/example/.ssh/aws.pem

freebsd_host ansible_python_interpreter=/usr/local/bin/python

ruby_module_host ansible_ruby_interpreter=/usr/bin/ruby.1.9.3

非SSH连接类型

如上一节所述，Ansible通过SSH执行playbooks，但不限于此连接类型。使用主机特定参数ansible_cnnection=<connector>，可以更改连接类型。以下非基于SSH的连接器可用：

本地

此连接器可用于将剧本部署到控制机器本身。

搬运工人

此连接器使用本地Docker客户端将playbook直接部署到Docker容器中。此连接器处理以下参数：

ansible_host

要连接的Docker容器的名称。

ansible_user

要在容器内操作的用户名。用户必须存在于容器内。

ansible_become

如果设置为true对become_user将被用于在容器内运行。

ansible_docker_extra_args

可以是一个包含Docker理解的任何其他参数的字符串，它们不是特定于命令的。此参数主要用于配要使用的远程Docker守护程序。

以下是如何立即部署到创建的容器的示例：

```
- name: create jenkins container
  docker_container:
    docker_host: myserver.net:4243
    name: my_jenkins
    image: jenkins

- name: add container to inventory
  add_host:
    name: my_jenkins
    ansible_connection: docker
    ansible_docker_extra_args: "--tlsverify --tlscacert=/path/to/ca.pem --tlscert=/path/to/client
cert.pem --tlskey=/path/to/client-key.pem -H=tcp://myserver.net:4243"
    ansible_user: jenkins
  changed_when: false

- name: create directory for ssh keys
  delegate_to: my_jenkins
  file:
    path: "/var/jenkins_home/.ssh/jupiter"
    state: directory
```