动态主机管理模块

add_host

在playbook执行的过程中，动态的添加主机到指定的主机组中
常用参数：

- groups：添加主机至指定的组
- name：要添加的主机名或IP地址

示例：

```
- name: add a host to group webservers
  hosts: webservers
  tasks:
    - add_host:
        name: "{{ item }}"
        group: webservers
        foo=42 #添加主机到webservers组中，主机的变量foo的值为42
      loop:
        - host1
        - host2

    - debug:
        msg: "{{ groups.webservers }}"
```

group_by

在playbook执行的过程中，动态的创建主机组

示例：

```
- name: Create operating system group
  hosts: all
  tasks:
```

```
    #在playbook中设置一个新的主机组
    - group_by:
        key: "os_{{ ansible_distribution }}"

- name: Run on CentOS hosts only
  hosts: os_CentOS
  tasks:
    - name: Install Apache
      yum:
        name: httpd
        state: latest

- name: Run on Ubuntu hosts only
  hosts: os_Ubuntu
  tasks:
    - name: Install Apache
      apt:
        name: apache2
        state: latest
```

动态inventory管理

动态inventory简介

在前面我们所有的选取主机组的操作都是通过维护inventory文件来完成的。而事实上，当在大规模应用当中，如果主机达成千上万台，这个时候还手动维护inventory文件将会给运维工作带来巨大的挑战。

在这种大规模的应用场景中，通常的做法是，将所有的主机都存储在cmdb当中。当需要对某一组主机或者某一类型的主机执行相应操作时，通过cmdb将相应主机取出来，动态的生成inventory，然后交由ansible处理即可。

所以其实Ansible Inventory包含静态inventory和动态inventory两部分。而我们前面通过手动在inventory文件中维护主机列表的方式即称之为静态inventory。而动态inventory则是指通过外部脚本获取主机列表，并按照ansible所要求的格式返回给ansible指令的操作方式。

动态inventory一般都会结合cmdb或者云计算平台等获取主机信息，由于主机资源一般会动态的进行增减，而这些系统一般会智能更新。我们需要通过这些工具提供的api或者接入库查询等方式返回主机列表。

动态inventory脚本规约

动态inventory脚本最终返回的满足ansible输出格式的json数据。ansible对于使用什么语言来实现动态inventory没有要求。但脚本必须支持两个参数：

- --list：用于返回所有的主机组信息，每个组所包含的主机列表hosts、子组列表children、主机变量列表vars都应该是字典形式的，而 `_meta` 则用于存放主机变量

- --host <host>：返回指定主机的变量列表，也可返回一个空字典

动态inventory脚本示例

```
#cat dynamic_inventory.py

#!/usr/bin/env python
# coding: utf-8
import os
import sys
import argparse

try:
    import json
except ImportError:
    import simplejson as json

class ExampleInventory(object):

    def __init__(self):
        self.inventory = {}
        self.read_cli_args()

        # Called with `--list`.
        if self.args.list:
            self.inventory = self.inventory_groups()
        # Called with `--host [hostname]`.
        elif self.args.host:
            # Not implemented, since we return _meta info `--list`.
            self.inventory = self.inventory_hosts(self.args.host)
        # If no groups or vars are present, return empty inventory.
        else:
            self.inventory = self.empty_inventory()

        print json.dumps(self.inventory);

    # Example inventory for testing.

    def inventory_groups(self):
        return {
            "webserver":  # 定义webserver组
            {
                "hosts": ["10.10.0.112"],  # webserver 组内主机
                "vars": {  # 组变量
                    "ansible_ssh_pass": "123456",
                    "ansible_ssh_port": "27100"
                },
                "children": ["dbserver"]  # 定义子组
            },
            "dbserver":
            {
                "hosts": ["10.10.0.109"],
                "vars": {
                    "ansible_ssh_pass": "123456",
                    "ansible_ssh_port": "27100"
```

```
                    }
            },
            '_meta': {
                # 定义主机变量
                'hostvars': {
                    '10.10.0.112': {
                        'host_specific_var': 'foo'
                    },
                    '10.10.0.109': {
                        'host_specific_var': 'bar'
                    }
                }
            }
        }

    def inventory_hosts(self,host):
        if host == "10.10.0.112":
            return {
              '_meta': {
                'hostvars': {
                    '10.10.0.112': {
                        'host_specific_var': 'foo'
                    }
                }
              }
            }
        elif host == "10.10.0.109":
            return {
              '_meta': {
                'hostvars': {
                    '10.10.0.109': {
                        'host_specific_var': 'bar'
                    }
                }
              }
            }
        else:
            return {'_meta': {'hostvars': {}}}

    # Read the command line args passed to the script.
    def read_cli_args(self):
        parser = argparse.ArgumentParser()
        parser.add_argument('--list', help="list hosts", action = 'store_true')
        parser.add_argument('--host', help="display hostvars for host",action = 'store')
        self.args = parser.parse_args()

# Get the inventory.
ExampleInventory()
```

脚本需要设置 `x` 权限，否则ansible会提示没有权限调用：

```
chmod +x ./dynamic_inventory.py
```

执行该脚本，返回如下：

```
#  ./dynamic_inventory.py --list
{"webserver": {"hosts": ["10.10.0.112"], "vars": {"ansible_ssh_port": "27100",
"ansible_ssh_pass": "123456"}, "children": ["dbserver"]}, "_meta": {"hostvars":
{"10.10.0.112": {"host_specific_var": "foo"}, "10.10.0.109": {"host_specific_var":
"bar"}}}, "dbserver": {"hosts": ["10.10.0.109"], "vars": {"ansible_ssh_port": "27100",
"ansible_ssh_pass": "123456"}}}

# ./dynamic_inventory.py --host 10.10.0.109
{"_meta": {"hostvars": {"10.10.0.109": {"host_specific_var": "bar"}}}}

# ./dynamic_inventory.py --host 192.168.0.1
{"_meta": {"hostvars": {}}}
```

通过ansible操作示例如下：

```
# ansible -i dynamic_inventory.py webserver --list-hosts
  hosts (1):
    10.10.0.112

# ansible -i dynamic_inventory.py all --list-hosts
  hosts (2):
    10.10.0.112
    10.10.0.109
```