

- fact简介
- 自定义fact
 - 1. 手动设置fact
 - 2. 使用set_fact模块定义新的变量
- 手动采集fact
- 启用fact缓存
 - 1. Json文件fact缓存后端
 - 2. Redis fact缓存后端
 - 3. Memcached fact缓存后端
- 关闭fact

fact简介

ansible有一个模块叫setup，用于获取远程主机的相关信息，并可以将这些信息作为变量在playbook里进行调用。而setup模块获取这些信息的方法就是依赖于fact。

```
# ansible test -m setup
10.1.61.187 | SUCCESS => {
    "ansible_facts": {
        "ansible_all_ipv4_addresses": [
            "10.1.61.187"
        ],
        "ansible_all_ipv6_addresses": [
            "fe80::f816:3eff:fe4f:6611"
        ],
        "ansible_apparmor": {
            "status": "disabled"
        },
        "ansible_architecture": "x86_64",
        "ansible_bios_date": "04/01/2014",
        "ansible_bios_version": "Ubuntu-1.8.2-1ubuntu1~cloud0",
        ...
    }
}
```

setup获取的这些信息，都是可用于该主机的变量。

setup模块其他用法示例：

```
# 查看主机内存信息
ansible 10.212.52.252 -m setup -a 'filter=ansible_*_mb'

# 查看地接口为eth0-2的网卡信息
ansible 10.212.52.252 -m setup -a 'filter=ansible_eth[0-2]'

# 将所有主机的信息输入到/tmp/facts目录下，每台主机的信息输入到主机名文件中 (/etc/ansible/hosts
```

```
里的主机名)
ansible all -m setup --tree /tmp/facts
```

自定义fact

1. 手动设置fact

ansible除了能获取到预定义的fact的内容,还支持手动为某个主机定制fact。称之为本地fact。本地fact默认存放于被控端的 /etc/ansible/facts.d 目录下, 如果文件为 ini 格式或者 json 格式, ansible会自动识别。以这种形式加载的fact是key为 ansible_local 的特殊变量。

下面是一个简单的示例, 在ansibler主控端定义一个 ini 格式的custom.fact文件内容如下:

```
[general]
package = httpd
service = httpd
state = started
```

然后我们编写一个playbook文件名为setup_facts.yml内容如下:

```
---
- name: Install remote facts
  hosts: test
  vars:
    remote_dir: /etc/ansible/facts.d
    facts_file: custom факт
  tasks:
    - name: Create the remote directory
      file:
        state: directory
        recurse: yes
        path: "{{ remote_dir }}"
    - name: Install the new facts
      copy:
        src: "{{ facts_file }}"
        dest: "{{ remote_dir }}"
```

执行该playbook, 完成facts的推送:

```
ansible-playbook setup_facts.yml
```

此时, 我们可以在被控端看到新的facts已经生成:

```
# ansible test -m setup
10.1.61.187 | SUCCESS => {
```

```

"ansible_facts": {

    ...output omitted...

    "ansible_local": {
        "custom": {
            "general": {
                "package": "httpd",
                "service": "httpd",
                "state": "started"
            }
        }
    },
    ...output omitted...
}

```

我们可以写一个简单的playbook来使用这些facts:

```

- name: Install Apache and starts the service
  hosts: test
  tasks:
    - name: Install the required package
      yum:
        name: "{{ ansible_facts.ansible_local.custom.general.package }}"
        state: latest
    - name: Start the service
      service:
        name: "{{ ansible_facts.ansible_local.custom.general.service }}"
        state: "{{ ansible_facts.ansible_local.custom.general.state }}"

```

2. 使用set_fact模块定义新的变量

`set_fact` 模块可以自定义facts，这些自定义的facts可以通过`template`或者变量的方式在playbook中使用。如果你想要获取一个进程使用的内存的百分比，则必须通过`set_fact`来进行计算之后得出其值，并将其值在playbook中引用。

下面是一个`set_fact`模块的应用示例：

```

- name: set_fact example
  hosts: test
  tasks:
    - name: Calculate InnoDB buffer pool size
      set_fact: innodb_buffer_pool_size_mb="{{ ansible_memtotal_mb / 2 | int }}"
    - debug: var=innodb_buffer_pool_size_mb

```

执行playbook如下：

```

# ansible-playbook set_fact_ex.yaml

PLAY [set_fact example]
*****
***** TASK [Gathering Facts]
*****
***** ok: [10.1.61.187]

TASK [Calculate InnoDB buffer pool size]
*****
***** ok: [10.1.61.187]

TASK [debug]
*****
***** ok: [10.1.61.187] => {
    "innodb_buffer_pool_size_mb": "3911.0"
}

PLAY RECAP
*****
***** 10.1.61.187 : ok=3      changed=0      unreachable=0      failed=0      skipped=0
rescued=0      ignored=0

```

这种设置方式只在当前playbook当中有效

手动采集fact

通常情况下，我们在运行play的时候，ansible会先尝试ssh到被控端采集fact，如果此时，被控制端的ssh还没有完全启动，就会导致整个play执行失败。这个时候，我们可以先显式的关闭fact采集，然后在task中通过wait_for等待被控端ssh端口被正常监听，再在task中使用setup模块来手动采集fact：

```

- name: Deploy apps
  hosts: webservers
  gather_facts: False
  tasks:
    - name: wait for ssh to be running
      local_action: wait_for port=22 host="{{ inventory_hostname }}" search_regex=OpenSSH
    - name: gather facts
      setup:
        .....

```

启用fact缓存

如果在play中需要引入fact，则可以开启fact缓存。fact缓存目前支持三种存储方式，分别为JSON、memcached、redis。

1. Json文件fact缓存后端

使用JSON文件作为fact缓存后端的时候，ansible将会把采集的fact写入到控制主机的文件中。

ansible.cfg配置如下：

```
[defaults]
gathering = smart
#缓存时间，单位为秒
fact_caching_timeout = 86400
fact_caching = jsonfile
#指定ansible包含fact的json文件位置，如果目录不存在，会自动创建
fact_caching_connection = /tmp/ansible_fact_cache
```

选项说明：

- gathering：是否启用fact，有三个选项：
 - smart：默认收集facts，但在facts已有的情况下就不收集，即使用facts缓存
 - implicit：默认收集facts，要禁止收集，必须显式的申明：gather_facts: false
 - explicit：默认不收集，要收集，必须显示的申明：gather_facts: true
- fact_caching_timeout：缓存时间，单位为s
- fact_caching：缓存的方式，支持jsonfile、redis、memcached
- fact_caching_connection：指定ansible缓存fact的连接方式，如果是jsonfile，则指定jsonfile的缓存路径

2. Redis fact缓存后端

使用redis作为fact缓存后端，需要在控制主机上安装redis服务并保持运行。需要安装python操作redis的软件包。

ansible.cfg配置如下：

```
[defaults]
gathering = smart
fact_caching_timeout = 86400
fact_caching = redis
fact_caching_connection = 127.0.0.1:6379:0
```

3. Memcached fact缓存后端

使用memcached作为fact缓存后端，需要在控制主机上安装Memcached服务并保持运行，需要安装python操作memcached的软件包。

ansible.cfg配置如下：

```
[defaults]
gathering = smart
fact_caching_timeout = 86400
fact_caching = memcached
fact_caching_connection = 127.0.0.1:11211
```

关闭fact

如果不想从fact中获取变量，或者说整个playbook当中都没有使用到fact变量，可以通过如下方法关闭fact以提升执行效率：

```
- hosts: test
gather_facts: no
```

也可以在ansible.cfg中添加如下配置：

```
[defaults]
gathering = explicit
```