Table of contents

- Table of contents
- Настройка
    - Подготовка узлов
    - inventory
    - Первая команда ansible .
    - ansible.cfg
    - первые команды
- Понимание ad-hoc модулей и playbook
- Модули
    - Вызов модуля
    - Получение справки по модулю
- Playbook
    - Создание и вызов playbook
    - Расширенная диагностика запуска playbook
    - Использование переменных
        - Обьявление переменных внутри playbook
        - Обьявление переменных внутри отдельного файла для группы хостов
        - Приоритеты переменных
            - Приоритеты
- Ansible Facts
    - Удобная команда, чтобы посмотреть значения факта для набора серверов
- Ansible vault - средство для хранения секретов
    - Создание защифрованного хранилища
    - Edit encypted file
    - Демонстрация использования.
- Лучшие практики по организации файловой структуры:
- Управление задачами
    - Register (Результат команды помещаем в переменную)
    - Циклы:
    - When (условие)
        - Пример установки пакетов только если дистрибутив линукса из списка допустимых:
        - Пример установки пакета с использованием цикла.
        - Пример рестартуем сервис, если команда вернула return code 0
        - пример с несколькими условиями
    - Handler
- Шаблоны (Templates)
- Операции с файлами
    - Сложный пример (не проверено)
- Ansible roles
    - (Ansible Galaxy)
    - Создание кастомных ролей
    - Использование системных ролей

Настройка

Подготовка узлов

- Для примера у нас будет одна управляющая нода `ansible` и две дочерние.
- Создадим на всех нодах пользователя `ansible` и дадим ему права запрашивать `sudo` без ввода пароля.
- Скопируем через `ssh-copy-id` сертификат для аутентификации по сертификатую См. Руководство по linux.

Также пропишем в `/etc/hosts` ip адреса (если они есть) управляемых нод для того, чтобы использовать имена.

```
192.168.0.32 ansible1
192.168.0.33 ansible2
```

inventory

Создадим `inventory` файл на управляющей ноде в домашнем каталоге пользователя.

```
[webservers]
ansible1
ansible2

[dbservers]
ansible1
ansible2

[lamp]
ansible1
```

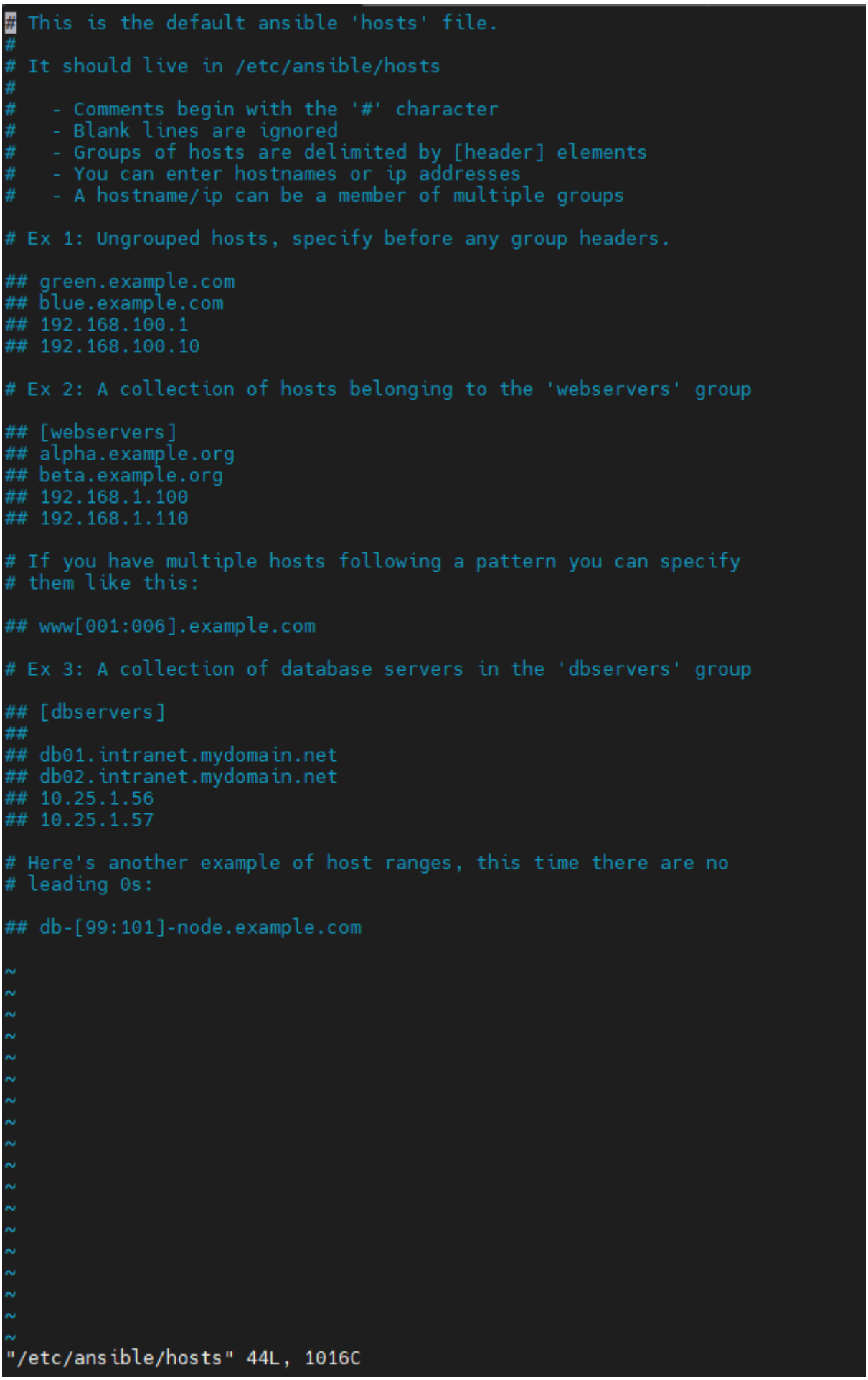
- можно делать группы серверов. Есть группа `all`, куда входят все сервера.
- эталонный инвентори файл `/etc/ansible/hosts` с примерами, как писать inventory. Видно, что можно писать хосты, ip адреса, а также группы серверов , диапазоны в названии. Не рекомендуется править его, лучше создать собственный inventory.
- Выше речь шла про статический inventory- список хостов. Для небольшого окружения он вполне подходит. Часто бывает полезным получать список хостов извне. Ansible поддерживает это. В документации есть набор скриптов для различного окружения, например получить список хостов ec2 Amazon.

Можно создавать вложенные группы черед nodes:children. Например

```
ansible1
ansible2

[lamp]
```

```
ansible1

[file]
ansible2

[nodes:children]
lamp
file
```

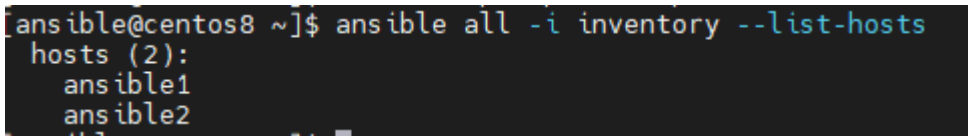
```
ansible1.example.com
ansible2.example.com

[lamp]
ansible1.example.com

[file]
ansible2.example.com

[nodes:children]
lamp
file
```



Первая команда ansible .

Показать все имена хостов из инвентори. Указываем файл инвентори через `-i`

```
ansible all -i inventory --list-hosts
```

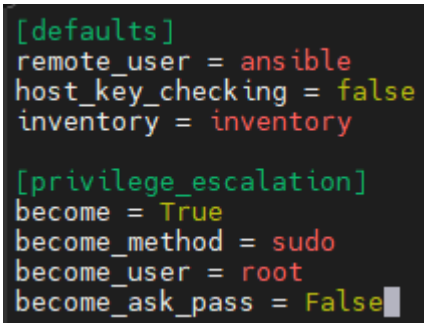


ansible.cfg

Файл настроек Ansible `/etc/ansible/ansible.cfg`

Рекомендуется создать собственный файл в каталоге пользователя с таким же именем

```
[defaults]
remote_user = ansible
host_key_checking = false
inventory = inventory

[privilege_escalation]
become = True
become_method = sudo
become_user = root
become_ask_pass = False
```



первые команды

Запускаем из домашней папки пользователя ansible

```
ansible all -m command -a "useradd bob"
```

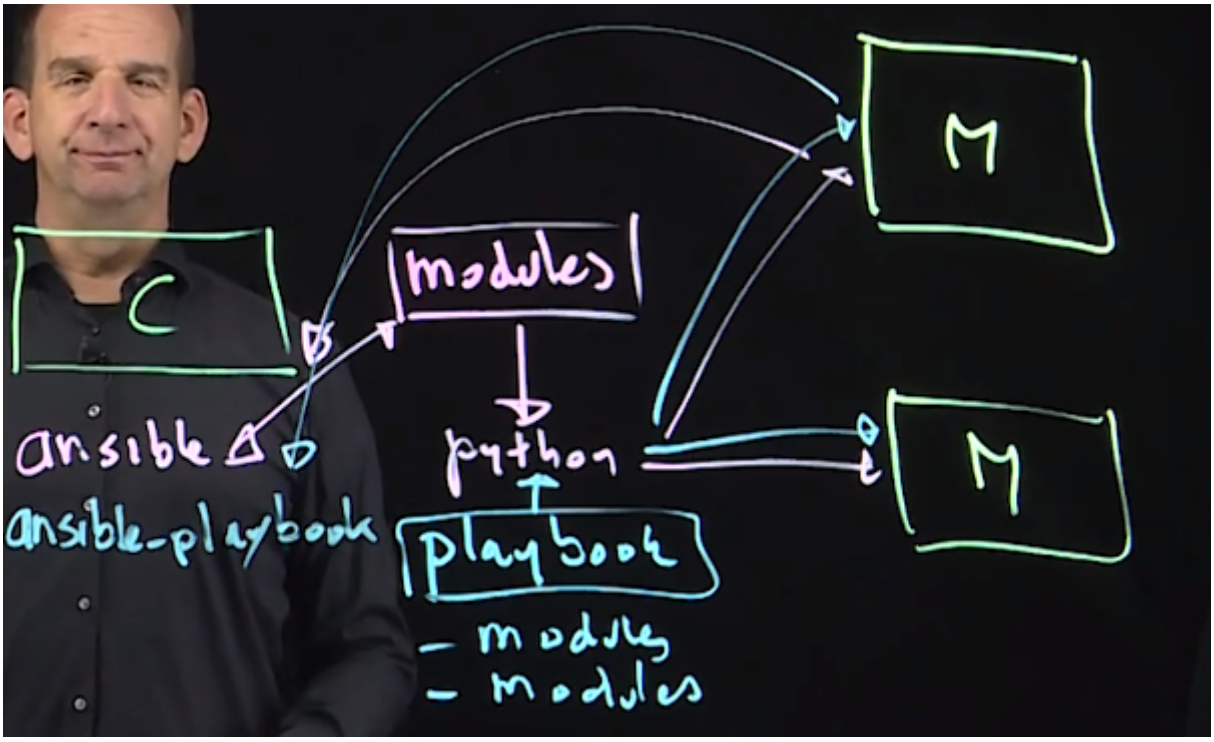
- `All` группа серверов
- `-m` модуль. В данном случае модуль command
- `-a` агрументы для модуля

Итого создадим на серверах пользователя bob

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible all -m command -a "useradd bob"
ansible1 | CHANGED | rc=0 >>

ansible2 | CHANGED | rc=0 >>
```

Понимание ad-hoc модулей и playbook

Через `m` можем вызвать 1 модуль. Но часто конфигурация это вызов нескольких действий. В этом случае делают плейбуки, которые вызывают несколько модулей.



Модули

Вызов модуля

Пример простого вызова (ad-hoc) модуля.

```
ansible all -m command -a "yum install -y mc"
```

```
[ansible@centos8 ~]$ ansible all -m command -a "yum install -y mc"
```

Пример вызова модуля с аргументами

```
ansible all -m user -a "name=linda shell=/bin/bash"
```

```
[ansible@centos8 ~]$ ansible all -m user -a "name=linda shell=/bin/bash"
ansible2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1003,
    "home": "/home/linda",
    "name": "linda",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1003
}
ansible1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "comment": "",
    "create_home": true,
    "group": 1003,
    "home": "/home/linda",
    "name": "linda",
    "shell": "/bin/bash",
    "state": "present",
    "system": false,
    "uid": 1003
}
```

Модуль command не позволяют делать piping, но есть модуль shell

```
ansible all -m shell -a "cat /etc/passwd | grep ansible"
```

```
[ansible@centos8 ~]$ ansible all -m shell -a "cat /etc/passwd | grep ansible"
ansible2 | CHANGED | rc=0 >>
ansible:x:1001:1001::/home/ansible:/bin/bash
ansible1 | CHANGED | rc=0 >>
ansible:x:1001:1001::/home/ansible:/bin/bash
```

Выше был пример вызова пакетного менеджера `yum`. Есть модуль `package` который может тоже самое, только через модуль

```
ansible all -m package -a "name=vsftpd state=installed"
```

```
[ansible@centos8 ~]$ ansible all -m package -a "name=vsftpd state=installed
ansible2 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Installed: vsftpd-3.0.3-32.el8.x86_64"
    ]
}
ansible1 | CHANGED => {
    "ansible_facts": {
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": true,
    "msg": "",
    "rc": 0,
    "results": [
        "Installed: vsftpd-3.0.3-32.el8.x86_64"
    ]
}
```

Стейт можно указать `absent` или `present` или

```
value of state must be one of: absent, installed, latest, present,
```

Получение справки по модулю

Вывести список всех модулей (их очень много - много экранов):

```
ansible-doc -l | less
```

```
[ansible@centos8 ~]$ ansible-doc -l | wc
   3387   27022  690975
```

```
a10_server                                      Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' server object
a10_server_axapi3                               Manage A10 Networks AX/SoftAX/Thunder/vThunder devices
a10_service_group                               Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' service groups
a10_virtual_server                              Manage A10 Networks AX/SoftAX/Thunder/vThunder devices' virtual servers
aci_aaa_user                                    Manage AAA users (aaa:User)
aci_aaa_user_certificate                        Manage AAA user certificates (aaa:UserCert)
aci_access_port_block_to_access_port            Manage port blocks of Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:PortBlk)
aci_access_port_to_interface_policy_leaf_profile Manage Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:RsAccBaseGrp, infra:PortBlk)
aci_access_sub_port_block_to_access_port        Manage sub port blocks of Fabric interface policy leaf profile interface selectors (infra:HPortS, infra:SubPortBlk)
aci_aep                                         Manage attachable Access Entity Profile (AEP) objects (infra:AttEntityP, infra:ProvAcc)
aci_aep_to_domain                               Bind AEPs to Physical or Virtual Domains (infra:RsDomP)
aci_ap                                          Manage top level Application Profile (AP) objects (fv:Ap)
aci_bd                                          Manage Bridge Domains (BD) objects (fv:BD)
aci_bd_subnet                                   Manage Subnets (fv:Subnet)
aci_bd_to_l3out                                 Bind Bridge Domain to L3 Out (fv:RsBDToOut)
aci_config_rollback                             Provides rollback and rollback preview functionality (config:ImportP)
aci_config_snapshot                             Manage Config Snapshots (config:Snapshot, config:ExportP)
aci_contract                                    Manage contract resources (vz:BrCP)
aci_contract_subject                            Manage initial Contract Subjects (vz:Subj)
aci_contract_subject_to_filter                  Bind Contract Subjects to Filters (vz:RsSubjFiltAtt)
aci_domain                                      Manage physical, virtual, bridged, routed or FC domain profiles (phys:DomP, vmm:DomP, l2ext:DomP, l3ext:DomP, fc:DomP)
aci_domain_to_encap_pool                        Bind Domain to Encap Pools (infra:RsVlanNs)
aci_domain_to_vlan_pool                         Bind Domain to VLAN Pools (infra:RsVlanNs)
aci_encap_pool                                  Manage encap pools (fvns:VlanInstP, fvns:VxlanInstP, fvns:VsanInstP)
aci_encap_pool_range                            Manage encap ranges assigned to pools (fvns:EncapBlk, fvns:VsanEncapBlk)
aci_epg                                         Manage End Point Groups (EPG) objects (fv:AEPg)
aci_epg_monitoring_policy                       Manage monitoring policies (mon:EPGPol)
aci_epg_to_contract                             Bind EPGs to Contracts (fv:RsCons, fv:RsProv)
aci_epg_to_domain                               Bind EPGs to Domains (fv:RsDomAtt)
aci_fabric_node                                 Manage Fabric Node Members (fabric:NodeIdentP)
aci_fabric_scheduler                            This modules creates ACI schedulers
aci_filter                                      Manages top level filter objects (vz:Filter)
aci_filter_entry                                Manage filter entries (vz:Entry)
aci_firmware_group                              This module creates a firmware group
aci_firmware_group_node                         This modules adds and remove nodes from the firmware group
aci_firmware_policy                             This creates a firmware policy
aci_firmware_source                             Manage firmware image sources (firmware:OSource)
aci_interface_policy_cdp                        Manage CDP interface policies (cdp:IfPol)
aci_interface_policy_fc                         Manage Fibre Channel interface policies (fc:IfPol)
aci_interface_policy_l2                         Manage Layer 2 interface policies (l2:IfPol)
```

В примере выше добавляли пользователя через модуль `command`. Но есть модуль для работы с пользователями. Поищем его: `ansible-doc -l | grep user`

```
ansible-doc -l | grep user
```

```
user                                    Manage user accounts
vertica_user                            Adds or removes Vertica database users and assigns roles
vmware_about_facts                      Provides information about VMware server to which user is connecting to
vmware_about_info                       Provides information about VMware server to which user is connecting to
```

По каждому модулю есть справка, например `ansible-doc user`

```
> USER    (/usr/lib/python3.6/site-packages/ansible/modules/system/user.py)

        Manage user accounts and user attributes. For Windows targets, use the [win_user] module instead.

  * This module is maintained by The Ansible Core Team
OPTIONS (= is mandatory):

- append
        If `yes', add the user to the groups specified in `groups'.
        If `no', user will only be added to the groups specified in `groups', removing them from all other
        Mutually exclusive with `local'
        [Default: False]
        type: bool

- authorization
        Sets the authorization of the user.
        Does nothing when used with other platforms.
        Can set multiple authorizations using comma separation.
        To delete all authorizations, use `authorization=''.
        Currently supported on Illumos/Solaris.
        [Default: (null)]
        type: str
        version_added: 2.8

- comment
        Optionally sets the description (aka `GECOS') of user account.
        [Default: (null)]
        type: str

- create_home
        Unless set to `no', a home directory will be made for the user when the account is created or if th
        Changed from `createhome' to `create_home' in Ansible 2.5.
        (Aliases: createhome)[Default: True]
```

Обязательные аргументы при вызове отмечены символом `=`

```
- move_home
        If set to `yes' when used with `home: ', attempt
        exists.
        [Default: False]
        type: bool

= name
        Name of the user to create, remove or modify.
        (Aliases: user)
        type: str
```

Внизу есть примеры. Правда они для плейбуков в `yml` формате

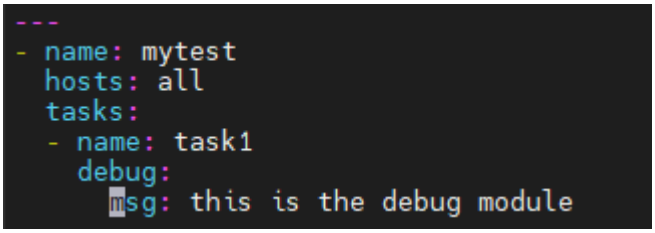
```
EXAMPLES:

- name: Add the user 'johnd' with a specific uid and a primary group of 'admin'
  user:
    name: johnd
    comment: John Doe
    uid: 1040
    group: admin
```

Playbook

Создание и вызов playbook

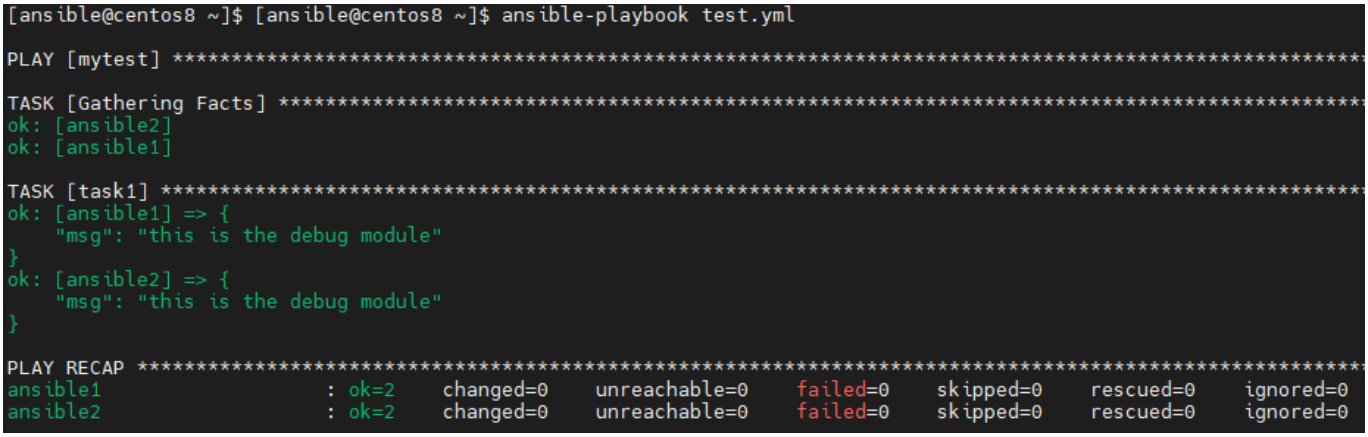
Пример:

```
---
- name: mytest
  hosts: all
  tasks:
  - name: task1
    debug:
      msg: this is the debug module
```



Используем плагин debug который просто напишет сообщение в консоль:

```
ansible-playbook test.yml
```



Пример посложнее:

```
---
- name: deploy vsftpd
  hosts: ansible1
  tasks:
  - name: install vsftpd
    yum: name=vsftpd
  - name: enable vsftpd
    service: name=vsftpd enabled=true
```

```
    - name: create readme file
      copy:
        content: "welcome to this FTP server"
        dest: /var/ftp/pub/README
        force: no
        mode: 0444
```

Тут главное заметить, что аргументы можно передавать как одной строкой сразу после модуля, так и делать полноценную yml структуру.

```
[ansible@centos8 ~]$ vim vsftpd.yml
---
- name: deploy vsftpd
  hosts: ansible1
  tasks:
  - name: install vsftpd
    yum: name=vsftpd
  - name: enable vsftpd
    service: name=vsftpd enabled=true
  - name: create readme file
    copy:
      content: "welcome to this FTP server"
      dest: /var/ftp/pub/README
      force: no
      mode: 0444
```

Первый раз

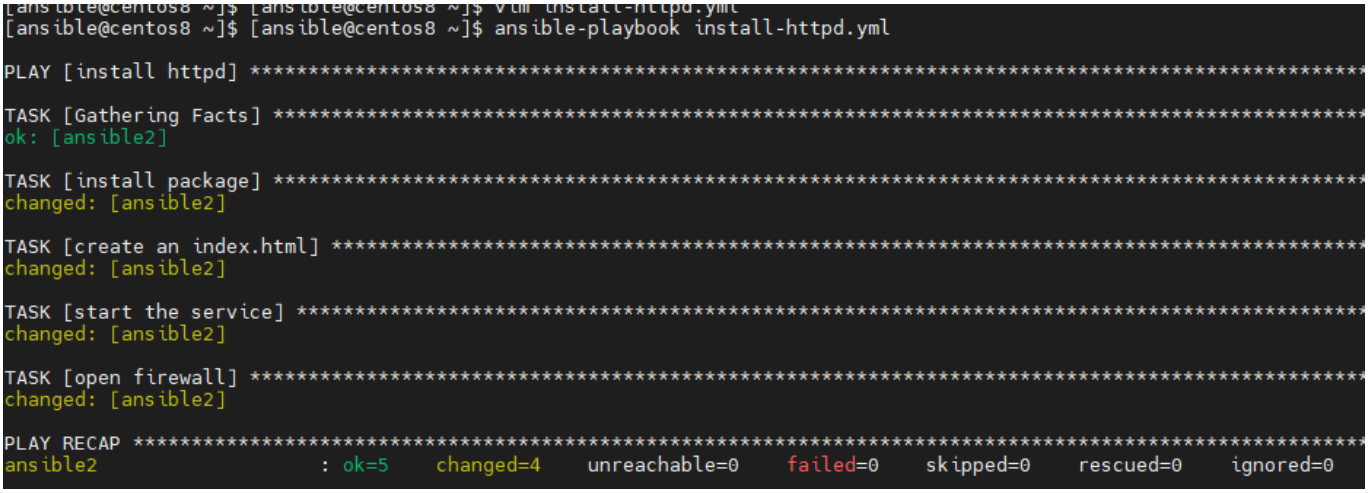
```
[ansible@centos8 ~]$ ansible-playbook vsftpd.yml

PLAY [deploy vsftpd] ***********************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [ansible1]

TASK [install vsftpd] **********************************************************************************************
ok: [ansible1]

TASK [enable vsftpd] ***********************************************************************************************
changed: [ansible1]

TASK [create readme file] ******************************************************************************************
changed: [ansible1]

PLAY RECAP *********************************************************************************************************
ansible1                   : ok=4    changed=2    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Второй раз

```
[ansible@centos8 ~]$ ansible-playbook vsftpd.yml

PLAY [deploy vsftpd] ***********************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [ansible1]

TASK [install vsftpd] **********************************************************************************************
ok: [ansible1]

TASK [enable vsftpd] ***********************************************************************************************
ok: [ansible1]

TASK [create readme file] ******************************************************************************************
ok: [ansible1]

PLAY RECAP *********************************************************************************************************
ansible1                   : ok=4    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Установка httpd

```yaml
---
- name: install httpd
  hosts: ansible2
  tasks:
    - name: install package
      package:
        name: httpd
        state: present
    - name: create an index.html
      copy:
        content: "welcome to this webserver"
        dest: /var/www/html/index.html
    - name: start the service
      service:
        name: httpd
        state: started
        enabled: true
    - name: open firewall
      firewalld:
        service: http
        permanent: yes
        state: enabled
```



Удаление httpd

```yaml
---
- name: remove httpd
  hosts: ansible2
  tasks:
    - name: close firewall
      firewalld:
        service: htpd
        permanent: yes
        state: disabled
    - name: remove file
      file:
        path: /var/www/html/index.html
```

```
      state: absent
  - name: remove package
    package:
      name: httpd
      state: absent
```

Расширенная диагностика запуска playbook

Ключ `-v` позволяет увидеть расширенный вывод при запуске плейбука.

```
ansible-playbook -v ./uninstall-httpd.yml
```

Причем, есть до 4-х уровней детализации. Т.е `-vv`, `-vvv`, `-vvvv`. Последний вообще очень подробный.

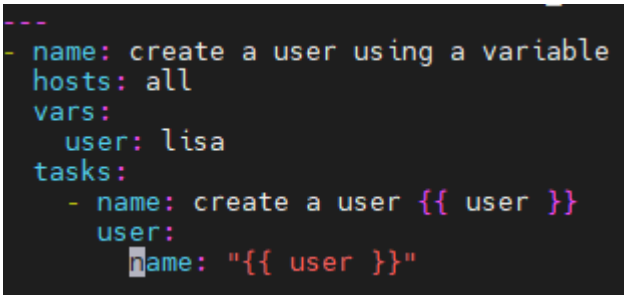
```
[ansible@centos8 ~]$ ansible-playbook -vv ./uninstall-httpd.yml
ansible-playbook 2.9.16
  config file = /home/ansible/ansible.cfg
  configured module search path = ['/home/ansible/.ansible/plugins/modules', '/usr/share/ansible
  ansible python module location = /usr/lib/python3.6/site-packages/ansible
  executable location = /usr/bin/ansible-playbook
  python version = 3.6.8 (default, Aug 24 2020, 17:57:11) [GCC 8.3.1 20191121 (Red Hat 8.3.1-5)]
Using /home/ansible/ansible.cfg as config file
Skipping callback 'actionable', as we already have a stdout callback.
Skipping callback 'counter_enabled', as we already have a stdout callback.
Skipping callback 'debug', as we already have a stdout callback.
Skipping callback 'dense', as we already have a stdout callback.
Skipping callback 'dense', as we already have a stdout callback.
Skipping callback 'full_skip', as we already have a stdout callback.
Skipping callback 'json', as we already have a stdout callback.
Skipping callback 'minimal', as we already have a stdout callback.
Skipping callback 'null', as we already have a stdout callback.
Skipping callback 'oneline', as we already have a stdout callback.
Skipping callback 'selective', as we already have a stdout callback.
Skipping callback 'skippy', as we already have a stdout callback.
Skipping callback 'stderr', as we already have a stdout callback.
Skipping callback 'unixy', as we already have a stdout callback.
Skipping callback 'yaml', as we already have a stdout callback.

PLAYBOOK: uninstall-httpd.yml **************************************************************
1 plays in ./uninstall-httpd.yml

PLAY [remove httpd] ************************************************************************

TASK [Gathering Facts] *********************************************************************
task path: /home/ansible/uninstall-httpd.yml:2
ok: [ansible2]
META: ran handlers

TASK [close firewall] **********************************************************************
task path: /home/ansible/uninstall-httpd.yml:5
ok: [ansible2] => {"changed": false, "msg": "Permanent operation"}

TASK [remove file] *************************************************************************
task path: /home/ansible/uninstall-httpd.yml:10
ok: [ansible2] => {"changed": false, "path": "/var/www/html/index.html", "state": "absent"}

TASK [remove package] **********************************************************************
```

Использование переменных

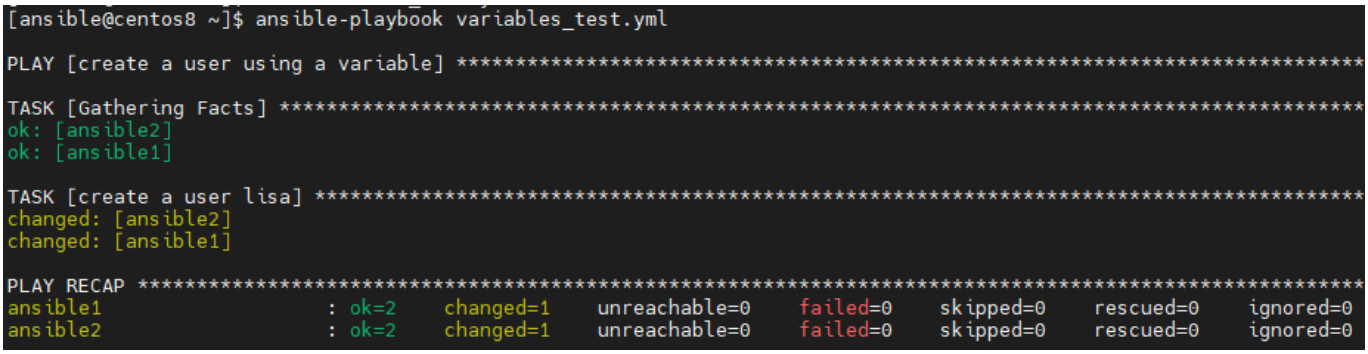
Обьявление переменных внутри playbook

Внутри playbook можно использовать переменные.

```
---
- name: create a user using a variable
  hosts: all
  vars:
    user: lisa
  tasks:
    - name: create a user {{ user }}
      user:
        name: "{{ user }}"
```



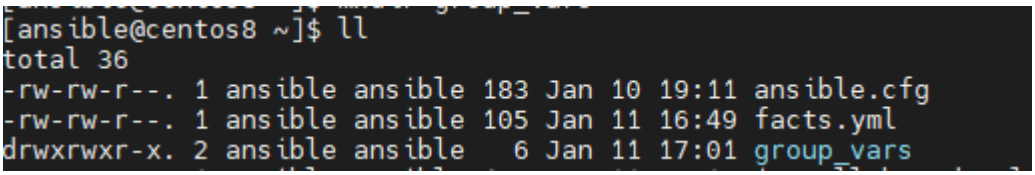
Тут важно, что в name ссылка на переменную в фигурных скобках . Это важно, когда в строчке встречается переменная и она идет первой



Обьявление переменных внутри отдельного файла для группы хостов

- Переменные можно задавать в отдельном файле для группы хостов.
- Каталог для создания файла с переменными `group_vars`

```
mkdir group_vars
```



- Создадим в нем файле для группы серверов `lamp` и укажем в нем переменные:

```
[ansible@centos8 ~]$ ll ./group_vars/
total 4
-rw-rw-r--. 1 ansible ansible 38 Jan 11 17:02 lamp
```

```
web_package: httpd
web_service: httpd
```

- Добавим группу `lamp` в inventory

```
[ansible@centos8 ~]$ cat inventory
[webservers]
ansible1
ansible2

[dbservers]
ansible1
ansible2

[lamp]
ansible1
```

И напишем простой плейбук:

```
---
- name: configure web services
  hosts: lamp
  tasks:
    - name: this is the {{ web_package }} package
      debug:
        msg: "Installing {{ web_package }}"
    - name: this is the {{ web_service }} service
      debug:
        msg: "Starting {{ web_service }}"
```

```
---
- name: configure web services
  hosts: lamp
  tasks:
    - name: this is the {{ web_package }} package
      debug:
        msg: "Installing {{ web_package }}"
    - name: this is the {{ web_service }} service
      debug:
        msg: "Starting {{ web_service }}"
```

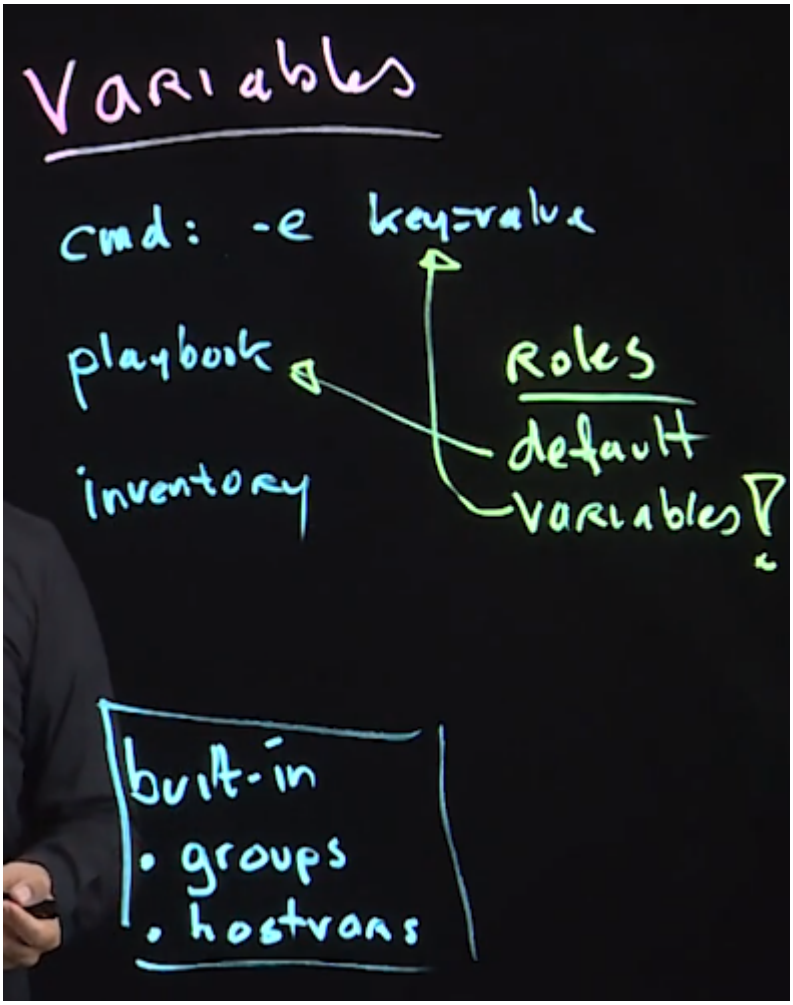
- Запустим плейбук

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook site.yml

PLAY [configure web services] **************************************************

TASK [Gathering Facts] *********************************************************
ok: [ansible1]

TASK [this is the httpd package] ***********************************************
ok: [ansible1] => {
    "msg": "Installing httpd"
}

TASK [this is the httpd service] ***********************************************
ok: [ansible1] => {
    "msg": "Starting httpd"
}

PLAY RECAP *********************************************************************
ansible1                   : ok=3    changed=0    unreachable=0    failed=0
```

- Видим, что переменные подставились из файла.

Приоритеты переменных

- Можно объявлять переменные для конкретного хоста в каталоге `host_vars`



- Есть встроенные переменные, переменные уровня группы `groups` , а также `hostvars` (список переменных окружения на хосте)

Приоритеты

- передаются из командной строки под ключом `-e key=value`
- Переменные на уровне плейбука. Они имеют приоритет выше чем дефолтные переменные группы.
- Переменные на уровне группы
- Наконец последние по приоритету это переменны из inventory.

Выше был пример с переменными из группы. Покажем, что переменные из командной строки имеют выше приоритет.

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook site.yml

PLAY [configure web services] ***************************************************

TASK [Gathering Facts] **********************************************************
ok: [ansible1]

TASK [this is the httpd package] ************************************************
ok: [ansible1] => {
    "msg": "Installing httpd"
}

TASK [this is the httpd service] ************************************************
ok: [ansible1] => {
    "msg": "Starting httpd"
}

PLAY RECAP **********************************************************************
ansible1                   : ok=3    changed=0    unreachable=0    failed=0

[ansible@centos8 ~]$ ansible-playbook site.yml -e "web_package=apache"

PLAY [configure web services] ***************************************************

TASK [Gathering Facts] **********************************************************
ok: [ansible1]

TASK [this is the apache package] ***********************************************
ok: [ansible1] => {
    "msg": "Installing apache"
}

TASK [this is the httpd service] ************************************************
ok: [ansible1] => {
    "msg": "Starting httpd"
}

PLAY RECAP **********************************************************************
ansible1                   : ok=3    changed=0    unreachable=0    failed=0
```

Теперь добавим переменную в плейбук. Покажем что переменные в плейбуке имеют приоритет над приоритетами группы

```
---
- name: configure web services
  hosts: lamp
  vars:
    web_package: nginx
  tasks:
    - name: this is the {{ web_package }} package
      debug:
        msg: "Installing {{ web_package }}"
    - name: this is the {{ web_service }} service
      debug:
        msg: "Starting {{ web_service }}"
```

```
[ansible@centos8 ~]$ ansible-playbook site.yml

PLAY [configure web services] ********************

TASK [Gathering Facts] ***************************
ok: [ansible1]

TASK [this is the nginx package] *****************
ok: [ansible1] => {
    "msg": "Installing nginx"
}

TASK [this is the httpd service] *****************
ok: [ansible1] => {
    "msg": "Starting httpd"
}

PLAY RECAP ***************************************
ansible1                   : ok=3    changed=0
```

Ansible Facts

Каждый раз при исполнении плейбука, ансибле запускает фазу `gathering facts`. Во время нее собирается много информации о хосте, которая в дальнейшем может быть использована в плейбуке. Эта информация присваивается переменной `ansible_facts`

Простой пример:

```
- name: show facts
  hosts: all
  tasks:
    - name: pring facts
      debug:
        var: ansible_facts
```

```
- name: show facts
  hosts: all
  tasks:
    - name: pring facts
      debug:
        var: ansible_facts
```

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook facts.yml

PLAY [show facts] ***********************************************************

TASK [Gathering Facts] ******************************************************
ok: [ansible2]
ok: [ansible1]

TASK [pring facts] **********************************************************
ok: [ansible1] => {
    "ansible_facts": {
        "all_ipv4_addresses": [
            "192.168.0.32",
            "10.0.2.15"
        ],
        "all_ipv6_addresses": [
            "fe80::a00:27ff:fee3:35ec"
        ],
        "ansible_local": {},
        "apparmor": {
            "status": "disabled"
        },
        "architecture": "x86_64",
        "bios_date": "12/01/2006",
        "bios_version": "VirtualBox",
        "cmdline": {
```

```yaml
---
- name: show ip of hosts
  hosts: all
  tasks:
    - name: show IP address
      debug:
        msg: IP address {{ansible_facts.default_ipv4.address}}
```


```yaml
---
- name: show ip of hosts
  hosts: all
  tasks:
    - name: show IP address
      debug:
        msg: IP address {{ansible_facts.default_ipv4.address}}
```

```
[ansible@centos8 ~]$ ansible-playbook show_ip.yml

PLAY [show ip of hosts] ********************************************

TASK [Gathering Facts] *********************************************
ok: [ansible2]
ok: [ansible1]

TASK [show IP address] *********************************************
ok: [ansible1] => {
    "msg": "IP address 10.0.2.15"
}
ok: [ansible2] => {
    "msg": "IP address 10.0.2.15"
}

PLAY RECAP *********************************************************
ansible1                   : ok=2    changed=0    unreachable=0
ansible2                   : ok=2    changed=0    unreachable=0
```

Удобная команда, чтобы посмотреть значения факта для набора серверов

```
ansible all -m setup -a "filter=ansible_distribution"
```

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible all -m setup -a "filter=ansible_distribution"
ansible2 | SUCCESS => {
    "ansible_facts": {
        "ansible_distribution": "CentOS",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
ansible1 | SUCCESS => {
    "ansible_facts": {
        "ansible_distribution": "CentOS",
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

```
[ansible@centos8 ~]$ ansible all -m setup -a "filter=ansible_memfree_mb"
ansible2 | SUCCESS => {
    "ansible_facts": {
        "ansible_memfree_mb": 1103,
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
ansible1 | SUCCESS => {
    "ansible_facts": {
        "ansible_memfree_mb": 1129,
        "discovered_interpreter_python": "/usr/libexec/platform-python"
    },
    "changed": false
}
```

Ansible vault - средство для хранения секретов

Создание защифрованного хранилища

```
ansible-vault create secret.yml
```

```
[vagrant@centos8 ~]$ ansible-vault create secret.yml
New Vault password:
```

- Введем пароль для хранилища, затем откроется редактор с пустым файлом, добавим туда те свойства, которые хотим хранить внутри хранилища

```
2. ansible_virtual_machine
username: lisa
pwhash: password
~
~
~
```

- И сохраним файл
- Вот содержимое файла:

```
[vagrant@centos8 ~]$ cat secret.yml
$ANSIBLE_VAULT;1.1;AES256
36623061303037316163633562653138643031346435326630303565323638353531366434636636
64623839616132353663393466646236636266646164656 50a383136643764643666386438333636
64613230333362626566313333663536636623835386666616261363266326265376536333232643 8
64353361623934306 20a6332333666636134616132353764393634313639613930323035356165 37
32383339383461643062353535643663666236333030366436336636326463396533383132373637
643562623566656534306636643862366238363665336335633 7
```

Edit encypted file

View encrypted file

```
ansible-vault view ./secret.yml
```

Edit encrypted file

```
ansible-vault edit ./secret.yml
```

Демонстрация использования.

Создадим простой плейбук:

```
---
- name: create a user
  hosts: all
  vars_files:
    - secret.yml
  tasks:
    - name: creating user
```

```
        user:
          name: "{{ username }}"
          password: "{{ pwhash }}"
```

```
---
- name: create a user
  hosts: all
  vars_files:
    - secret.yml
  tasks:
    - name: creating user
      user:
        name: "{{ username }}"
        password: "{{ pwhash }}"
```

```
[ansible@centos8 ~]$ ansible-playbook create-user.yml
ERROR! Attempting to decrypt but no vault secrets found
```

```
[ansible@centos8 ~]$ ansible-playbook --ask-vault-pass create-user.yml
Vault password:

PLAY [create a user] ***********************************************************

TASK [Gathering Facts] *********************************************************
ok: [ansible1]
ok: [ansible2]

TASK [creating user] ***********************************************************
[WARNING]: The input ******** appears not to have been hashed. The '********
work properly.
changed: [ansible1]
changed: [ansible2]

PLAY RECAP *********************************************************************
ansible1                   : ok=2    changed=1    unreachable=0    failed=0
ansible2                   : ok=2    changed=1    unreachable=0    failed=0
```

- Передавать пароли так нельзя. Т.к их будет видно в shadow

```
[ansible@centos8 ~]$ ansible ansible1 -m command -a "grep lisa /etc/shadow"
ansible1 | CHANGED | rc=0 >>
lisa:password:18640:0:99999:7:::
```

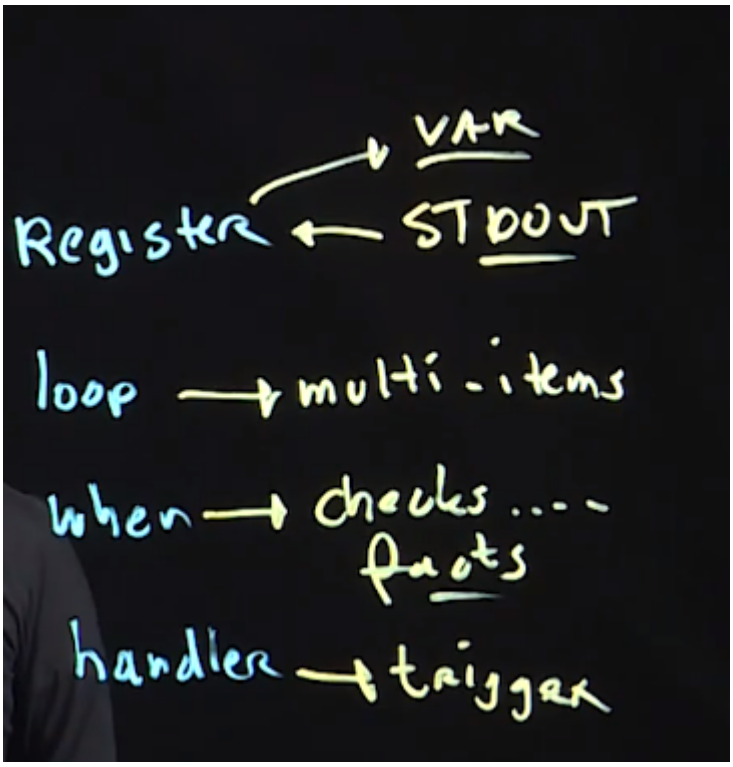
Лучшие практики по организации файловой структуры:

https://docs.ansible.com/ansible/2.8/user_guide/playbooks_best_practices.html#directory-layout

The top level of the directory would contain files and directories like so:

```
production                # inventory file for production servers
staging                   # inventory file for staging environment

group_vars/
   group1.yml             # here we assign variables to particular groups
   group2.yml
host_vars/
   hostname1.yml          # here we assign variables to particular systems
   hostname2.yml

library/                  # if any custom modules, put them here (optional)
module_utils/             # if any custom module_utils to support modules, put them here (optional)
filter_plugins/           # if any custom filter plugins, put them here (optional)

site.yml                  # master playbook
webservers.yml            # playbook for webserver tier
dbservers.yml             # playbook for dbserver tier

roles/
    common/               # this hierarchy represents a "role"
        tasks/            #
            main.yml      #  <-- tasks file can include smaller files if warranted
        handlers/         #
            main.yml      #  <-- handlers file
        templates/        #  <-- files for use with the template resource
            ntp.conf.j2   #  <------- templates end in .j2
        files/            #
            bar.txt       #  <-- files for use with the copy resource
            foo.sh        #  <-- script files for use with the script resource
        vars/             #
            main.yml      #  <-- variables associated with this role
        defaults/         #
            main.yml      #  <-- default lower priority variables for this role
        meta/             #
            main.yml      #  <-- role dependencies
        library/          # roles can also include custom modules
        module_utils/     # roles can also include custom module_utils
        lookup_plugins/   # or other types of plugins, like lookup in this case

    webtier/              # same kind of structure as "common" was above, done for the webtier role
    monitoring/           # ""
    fooapp/               # ""
```

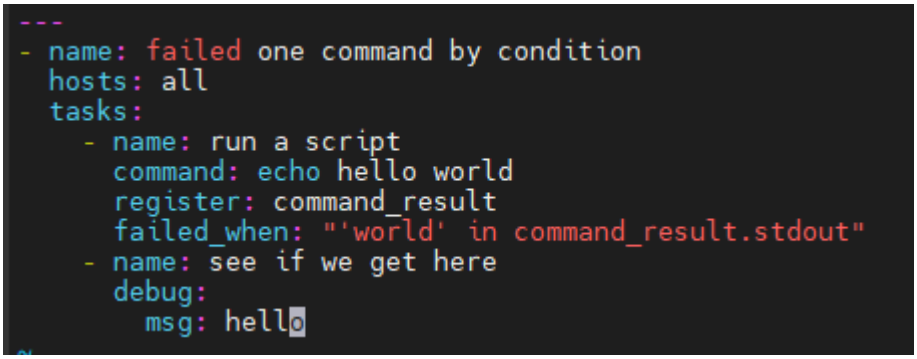
Управление задачами



Register (Результат команды помещаем в переменную)

Пример показывает, вызывали команду, сохранили результат stdout в переменную, далее можем использовать `failed_when` - условие определения неуспешности команды. Все последующие команды будут отменны.

```
---
- name: failed one command by condition
  hosts: all
  tasks:
    - name: run a script
      command: echo hello world
      register: command_result
      failed_when: "'world' in command_result.stdout"
    - name: see if we get here
      debug:
        msg: hello
```



```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook register-example.yml

PLAY [failed one command by condition] *****************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [ansible1]
ok: [ansible2]

TASK [run a script] ************************************************************************************************
fatal: [ansible2]: FAILED! => {"changed": true, "cmd": ["echo", "hello", "world"], "delta": "0:00:00.004979", "end": "2021-01-21
 16:37:45.165015", "failed_when_result": true, "rc": 0, "start": "2021-01-21 16:37:45.160036", "stderr": "", "stderr_lines": [],
 "stdout": "hello world", "stdout_lines": ["hello world"]}
fatal: [ansible1]: FAILED! => {"changed": true, "cmd": ["echo", "hello", "world"], "delta": "0:00:00.004627", "end": "2021-01-21
 16:37:45.169482", "failed_when_result": true, "rc": 0, "start": "2021-01-21 16:37:45.164855", "stderr": "", "stderr_lines": [],
 "stdout": "hello world", "stdout_lines": ["hello world"]}

PLAY RECAP *********************************************************************************************************
ansible1                   : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
ansible2                   : ok=1    changed=0    unreachable=0    failed=1    skipped=0    rescued=0    ignored=0
```

Если напишем `Ignore_errors` тогда попадем во второй шаг

```yaml
---
- name: failed one command by condition
  hosts: all
  tasks:
    - name: run a script
      command: echo hello world
      register: command_result
      failed_when: "'world' in command_result.stdout"
      ignore_errors: yes
    - name: see if we get here
      debug:
        msg: hello
```

```yaml
---
- name: demonstrating failed_when
  hosts: all
  tasks:
    - name: run a script
      command: echo hello world
      ignore_errors: yes
      register: command_result
      failed_when: "'world' in command_result.stdout"
    - name: see if we get here
      debug:
        msg: hello
```

```
[ansible@centos8 ~]$ ansible-playbook register-example.yml ple.yml

PLAY [failed one command by condition] *****************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [ansible1]
ok: [ansible2]

TASK [run a script] ************************************************************************************************
fatal: [ansible2]: FAILED! => {"changed": true, "cmd": ["echo", "hello", "world"], "delta": "0:00:00.002869", "end": "20
 16:39:50.041372", "failed_when_result": true, "rc": 0, "start": "2021-01-21 16:39:50.038503", "stderr": "", "stderr_lin
 "stdout": "hello world", "stdout_lines": ["hello world"]}
...ignoring
fatal: [ansible1]: FAILED! => {"changed": true, "cmd": ["echo", "hello", "world"], "delta": "0:00:00.003616", "end": "20
 16:39:50.051202", "failed_when_result": true, "rc": 0, "start": "2021-01-21 16:39:50.047586", "stderr": "", "stderr_lin
 "stdout": "hello world", "stdout_lines": ["hello world"]}
...ignoring

TASK [see if we get here] ******************************************************************************************
ok: [ansible1] => {
    "msg": "hello"
}
ok: [ansible2] => {
    "msg": "hello"
}

PLAY RECAP *********************************************************************************************************
ansible1                   : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1
ansible2                   : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=1
```

Циклы:

Записали переменную в виде массива. Далее с помощью loop итерируемся по нему и подставляем в item значение из массива.

```yaml
---
- name: start some services
  hosts: all
  vars:
    my_services:
      - crond
      - sshd
  tasks:
    - name: start some services
      service:
        name: "{{ item }}"
        state: started
      loop: "{{ my_services }}"
```

```yaml
---
- name: start some services
  hosts: all
  vars:
    my_services:
      - crond
      - sshd
  tasks:
    - name: start some services
      service:
        name: "{{ item }}"
        state: started
      loop: "{{ my_services }}"
~
```

```
[ansible@centos8 ~]$ ansible-playbook loops.yml

PLAY [start some services] ***************************************************************************************************

TASK [Gathering Facts] *******************************************************************************************************
ok: [ansible2]
ok: [ansible1]

TASK [start some services] ***************************************************************************************************
ok: [ansible1] => (item=crond)
ok: [ansible2] => (item=crond)
ok: [ansible1] => (item=sshd)
ok: [ansible2] => (item=sshd)

PLAY RECAP *******************************************************************************************************************
ansible1                   : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ansible2                   : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

When (условие)

Выполняем команды, если выполнилось условие. `When` использует массив фактов, которые собираеются на фазе `gathering facts` или проверки переменных, которые ранее поместили через `register`

```
---
- name: test command result
  hosts: all
  tasks:
    - name: check a fact
      debug:
        msg: echo today is 26
      when: '"26" in ansible_date_time.day'
```

```
---
- name: test command result
  hosts: all
  tasks:
    - name: check a fact
      debug:
        msg: echo today is 26
      when: '"26" in ansible_date_time.day'
~
```

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook check_when.yml

PLAY [test command result] ***************************************************************************************************

TASK [Gathering Facts] *******************************************************************************************************
ok: [ansible1]
ok: [ansible2]

TASK [check a fact] **********************************************************************************************************
ok: [ansible1] => {
    "msg": "echo today is 26"
}
ok: [ansible2] => {
    "msg": "echo today is 26"
}

PLAY RECAP *******************************************************************************************************************
ansible1                   : ok=2    changed=0    unreachable=0    failed=0    skipped=0
ansible2                   : ok=2    changed=0    unreachable=0    failed=0    skipped=0
```

- Поменяем дату и увидим, что условие не выполнилось и команда не выполнилась.

```yaml
---
- name: test command result
  hosts: all
  tasks:
    - name: check a fact
      debug:
        msg: echo today is 26
      when: '"27" in ansible_date_time.day'
```

```yaml
---
- name: test command result
  hosts: all
  tasks:
    - name: check a fact
      debug:
        msg: echo today is 26
      when: '"27" in ansible_date_time.day'
```

```
[ansible@centos8 ~]$ ansible-playbook check_when.yml

PLAY [test command result] *****************************************************

TASK [Gathering Facts] *********************************************************
ok: [ansible1]
ok: [ansible2]

TASK [check a fact] ************************************************************
skipping: [ansible1]
skipping: [ansible2]

PLAY RECAP *********************************************************************
ansible1                   : ok=1    changed=0    unreachable=0    failed=0    skipped=1
ansible2                   : ok=1    changed=0    unreachable=0    failed=0    skipped=1
```

- Важно понимать что есть статегии сбора тасков. Можно посмотреть их описание в файле конфигурации `/etc/ansible/ansible.cfg`

```
# plays will gather facts by default, which contain information about
# the remote system.
#
# smart - gather by default, but don't regather if already gathered
# implicit - gather by default, turn off with gather_facts: False
# explicit - do not gather by default, must say gather_facts: True
#gathering = implicit
```

- Если поставить `explicit` факты будут собираться только если выставлена опция `gathering_facts: True`. По-умолчанию поведение `implicitit` , т.е собирается всегда, если не передана опция не собирать.
- Помним что мы создавали файл локальных конфигураций для пользователя в его домашней директории. Добавим туда строчку включающую `explicity` режим сбора фактов.

```
[defaults]
remote_user = vagrant
host_key_checking = false
inventory = inventory
```

```
gathering = explicit

[privilege_escalation]
become = True
become_method = sudo
become_user = root
become_ask_pass = False
```

```
[defaults]
remote_user = ansible
host_key_checking = false
inventory = inventory
gathering = explicit

[privilege_escalation]
become = True
become_method = sudo
become_user = root
become_ask_pass = False
```

- Если запустим опять плейбук, получим ошибку. Т.к факты не собрались и нет такой переменной с датой.

```
[ansible@centos8 ~]$ ansible-playbook check_when.yml

PLAY [test command result] *****************************************

TASK [check a fact] ************************************************
fatal: [ansible1]: FAILED! => {"msg": "The conditional check '\"27\" in ansib
e evaluating conditional (\"27\" in ansible_date_time.day): 'ansible_date_tim
e/ansible/check_when.yml': line 5, column 7, but may\nbe elsewhere in the fil
ending line appears to be:\n\n  tasks:\n    - name: check a fact\n      ^ her
fatal: [ansible2]: FAILED! => {"msg": "The conditional check '\"27\" in ansib
e evaluating conditional (\"27\" in ansible_date_time.day): 'ansible_date_tim
e/ansible/check_when.yml': line 5, column 7, but may\nbe elsewhere in the fil
ending line appears to be:\n\n  tasks:\n    - name: check a fact\n      ^ her

PLAY RECAP *********************************************************************
ansible1                   : ok=0    changed=0    unreachable=0    failed=1
ansible2                   : ok=0    changed=0    unreachable=0    failed=1
```

- Чтобы починить, явно указать на сбор фактов в плейбуке

```
---
- name: test command result
  hosts: all
  gather_facts: yes
  tasks:
    - name: check a fact
      debug:
        msg: echo today is 26
      when: '"26" in ansible_date_time.day'
```

```
---
- name: test command result
  hosts: all
  gather_facts: yes
  tasks:
    - name: check a fact
      debug:
        msg: echo today is 26
      when: '"27" in ansible_date_time.day'
~
```

- Починилось

```
[ansible@centos8 ~]$ ansible-playbook check_when.yml

PLAY [test command result] *****************************************************

TASK [Gathering Facts] *********************************************************
ok: [ansible2]
ok: [ansible1]

TASK [check a fact] ************************************************************
skipping: [ansible1]
skipping: [ansible2]

PLAY RECAP *********************************************************************
ansible1                   : ok=1    changed=0    unreachable=0    failed=0
ansible2                   : ok=1    changed=0    unreachable=0    failed=0
```

Пример установки пакетов только если дистрибутив линукса из списка допустимых:

```yaml
---
- name: when demo
  gather_facts: yes
  hosts: all
  vars:
    supported_distros:
      - RedHat
      - CentOS
      - Fedora
  tasks:
    - name: install RH family specific packages
      yum:
        name: nginx
        state: present
      when: ansible_distribution in supported_distros
```

```
---
- name: when demo
  gather_facts: yes
  hosts: all
  vars:
    supported_distros:
      - RedHat
      - CentOS
      - Fedora
  tasks:
    - name: install RH family specific packages
      yum:
        name: nginx
        state: present
      when: ansible_distribution in supported_distros
```

```
[ansible@centos8 ~]$ ansible-playbook distro.yml

PLAY [when demo] ***************************************************************

TASK [Gathering Facts] *********************************************************
ok: [ansible1]
ok: [ansible2]

TASK [install RH family specific packages] *************************************
changed: [ansible1]
changed: [ansible2]

PLAY RECAP *********************************************************************
ansible1                   : ok=2    changed=1    unreachable=0    failed=0
ansible2                   : ok=2    changed=1    unreachable=0    failed=0
```

Пример установки пакета с использованием цикла.

Если есть определенная точка монтирования и места на ней достаточно, тогда устанавливаем пакет

```
---
- name: condiionals test
  gather_facts: yes
  hosts: all
  tasks:
  - name: install vsftpd if sufficient space on /var/ftp
    package:
      name: vsftpd
      state: latest
    with_items: "{{ ansible_mounts }}"
    when: item.mount == "/var/ftp" and item.size_available > 100000
```

```yaml
---
- name: condiionals test
  gather_facts: yes
  hosts: all
  tasks:
  - name: install vsftpd if sufficient space on /var/ftp
    package:
      name: vsftpd
      state: latest
    with_items: "{{ ansible_mounts }}"
    when: item.mount == "/var/ftp" and item.size_avaliable > 100000
```

```
[ansible@centos8 ~]$ ansible-playbook ifsize.yml

PLAY [condiionals test] ********************************************************************************************************

TASK [Gathering Facts] *********************************************************************************************************
ok: [ansible1]
ok: [ansible2]

TASK [install vsftpd if sufficient space on /var/ftp] **************************************************************************
skipping: [ansible1] => (item={'mount': '/', 'device': '/dev/mapper/cl_centos8-root', 'fstype': 'xfs', 'options': 'rw,seclabe
elatime,attr2,inode64,noquota', 'size_total': 53660876800, 'size_available': 51389255680, 'block_size': 4096, 'block_total':
00800, 'block_available': 12546205, 'block_used': 554595, 'inode_total': 26214400, 'inode_available': 26148167, 'inode_used':
233, 'uuid': 'f7d1b888-09fa-4a51-b153-10634842e02c'})
skipping: [ansible1] => (item={'mount': '/vagrant', 'device': '/vagrant', 'fstype': 'vboxsf', 'options': 'rw,nodev,relatime',
ize_total': 240054693888, 'size_available': 118532132864, 'block_size': 4096, 'block_total': 58607103, 'block_available': 289
09, 'block_used': 29668594, 'inode_total': 1000, 'inode_available': 1000, 'inode_used': 0, 'uuid': 'N/A'})
skipping: [ansible1] => (item={'mount': '/boot', 'device': '/dev/sda1', 'fstype': 'ext4', 'options': 'rw,seclabel,relatime',
ze_total': 1023303680, 'size_available': 806756352, 'block_size': 4096, 'block_total': 249830, 'block_available': 196962, 'bl
_used': 52868, 'inode_total': 65536, 'inode_available': 65226, 'inode_used': 310, 'uuid': '936904fb-f0ef-4b77-b46e-01c89f9035
})
skipping: [ansible2] => (item={'mount': '/', 'device': '/dev/mapper/cl_centos8-root', 'fstype': 'xfs', 'options': 'rw,seclabe
elatime,attr2,inode64,noquota', 'size_total': 53660876800, 'size_available': 51404271616, 'block_size': 4096, 'block_total':
00800, 'block_available': 12549871, 'block_used': 550929, 'inode_total': 26214400, 'inode_available': 26148739, 'inode_used':
661, 'uuid': 'f7d1b888-09fa-4a51-b153-10634842e02c'})
skipping: [ansible2] => (item={'mount': '/vagrant', 'device': '/vagrant', 'fstype': 'vboxsf', 'options': 'rw,nodev,relatime',
ize_total': 240054693888, 'size_available': 118532132864, 'block_size': 4096, 'block_total': 58607103, 'block_available': 289
09, 'block_used': 29668594, 'inode_total': 1000, 'inode_available': 1000, 'inode_used': 0, 'uuid': 'N/A'})
skipping: [ansible2] => (item={'mount': '/boot', 'device': '/dev/sda1', 'fstype': 'ext4', 'options': 'rw,seclabel,relatime',
ze_total': 1023303680, 'size_available': 806756352, 'block_size': 4096, 'block_total': 249830, 'block_available': 196962, 'bl
_used': 52868, 'inode_total': 65536, 'inode_available': 65226, 'inode_used': 310, 'uuid': '936904fb-f0ef-4b77-b46e-01c89f9035
})

PLAY RECAP *********************************************************************************************************************
ansible1                   : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
ansible2                   : ok=1    changed=0    unreachable=0    failed=0    skipped=1    rescued=0    ignored=0
```

- Посмотрим какие точки монтирования есть на машинах

```
ansible all -m command -a "df -h"
```

```
[ansible@centos8 ~]$ ansible all -m command -a "df -h"
ansible2 | CHANGED | rc=0 >>
Filesystem                      Size  Used Avail Use% Mounted on
devtmpfs                        900M     0  900M   0% /dev
tmpfs                           917M     0  917M   0% /dev/shm
tmpfs                           917M   17M  901M   2% /run
tmpfs                           917M     0  917M   0% /sys/fs/cgroup
/dev/mapper/cl_centos8-root      50G  2.2G   48G   5% /
vagrant                         224G  114G  111G  51% /vagrant
/dev/sda1                       976M  140M  770M  16% /boot
tmpfs                           184M     0  184M   0% /run/user/1000
tmpfs                           184M     0  184M   0% /run/user/1001
ansible1 | CHANGED | rc=0 >>
Filesystem                      Size  Used Avail Use% Mounted on
devtmpfs                        900M     0  900M   0% /dev
tmpfs                           917M     0  917M   0% /dev/shm
tmpfs                           917M   17M  901M   2% /run
tmpfs                           917M     0  917M   0% /sys/fs/cgroup
/dev/mapper/cl_centos8-root      50G  2.2G   48G   5% /
vagrant                         224G  114G  111G  51% /vagrant
/dev/sda1                       976M  140M  770M  16% /boot
tmpfs                           184M     0  184M   0% /run/user/1000
tmpfs                           184M     0  184M   0% /run/user/1001
```

- Поменяем на `/vagrant`

```yaml
---
- name: condiionals test
  gather_facts: yes
  hosts: all
  tasks:
  - name: install vsftpd if sufficient space on /var/ftp
    package:
      name: vsftpd
      state: latest
    with_items: "{{ ansible_mounts }}"
    when: item.mount == "/vagrant" and item.size_available > 100000
```

```yaml
---
- name: condiionals test
  gather_facts: yes
  hosts: all
  tasks:
  - name: install vsftpd if sufficient space on /var/ftp
    package:
      name: vsftpd
      state: latest
    with_items: "{{ ansible_mounts }}"
    when: item.mount == "/vagrant" and item.size_avaliable > 100000
```

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook ifsize.yml

PLAY [condiionals test] ********************************************************************************************

TASK [Gathering Facts] *********************************************************************************************
ok: [ansible1]
ok: [ansible2]

TASK [install vsftpd if sufficient space on /var/ftp] **************************************************************
skipping: [ansible1] => (item={'mount': '/', 'device': '/dev/mapper/cl_centos8-root', 'fstype': 'xfs', 'options': 'rw,se
elatime,attr2,inode64,noquota', 'size_total': 53660876800, 'size_available': 51389157376, 'block_size': 4096, 'block_tot
00800, 'block_available': 12546181, 'block_used': 554619, 'inode_total': 26214400, 'inode_available': 26148167, 'inode_u
233, 'uuid': 'f7d1b888-09fa-4a51-b153-10634842e02c'})
skipping: [ansible2] => (item={'mount': '/', 'device': '/dev/mapper/cl_centos8-root', 'fstype': 'xfs', 'options': 'rw,se
elatime,attr2,inode64,noquota', 'size_total': 53660876800, 'size_available': 51404210176, 'block_size': 4096, 'block_tot
00800, 'block_available': 12549856, 'block_used': 550944, 'inode_total': 26214400, 'inode_available': 26148739, 'inode_u
661, 'uuid': 'f7d1b888-09fa-4a51-b153-10634842e02c'})
ok: [ansible2] => (item={'mount': '/vagrant', 'device': '/vagrant', 'fstype': 'vboxsf', 'options': 'rw,nodev,relatime',
tal': 240054693888, 'size_available': 118532132864, 'block_size': 4096, 'block_total': 58607103, 'block_available': 2893
lock_used': 29668594, 'inode_total': 1000, 'inode_available': 1000, 'inode_used': 0, 'uuid': 'N/A'})
skipping: [ansible2] => (item={'mount': '/boot', 'device': '/dev/sda1', 'fstype': 'ext4', 'options': 'rw,seclabel,relati
ze_total': 1023303680, 'size_available': 806756352, 'block_size': 4096, 'block_total': 249830, 'block_available': 196962
_used': 52868, 'inode_total': 65536, 'inode_available': 65226, 'inode_used': 310, 'uuid': '936904fb-f0ef-4b77-b46e-01c89
})
ok: [ansible1] => (item={'mount': '/vagrant', 'device': '/vagrant', 'fstype': 'vboxsf', 'options': 'rw,nodev,relatime',
tal': 240054693888, 'size_available': 118532132864, 'block_size': 4096, 'block_total': 58607103, 'block_available': 2893
lock_used': 29668594, 'inode_total': 1000, 'inode_available': 1000, 'inode_used': 0, 'uuid': 'N/A'})
skipping: [ansible1] => (item={'mount': '/boot', 'device': '/dev/sda1', 'fstype': 'ext4', 'options': 'rw,seclabel,relati
ze_total': 1023303680, 'size_available': 806756352, 'block_size': 4096, 'block_total': 249830, 'block_available': 196962
_used': 52868, 'inode_total': 65536, 'inode_available': 65226, 'inode_used': 310, 'uuid': '936904fb-f0ef-4b77-b46e-01c89
})

PLAY RECAP *********************************************************************************************************
ansible1                   : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
ansible2                   : ok=2    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

Пример рестартуем сервис, если команда вернула return code 0

```yaml
---
- name: restart sshd only if crond is running
  gather_facts: yes
  hosts: all
  tasks:
    - name: get the crond server status
      command: /usr/bin/systemctl is-active crond
      ignore_errors: yes
      register: result
    - name: restart sshd based on crond status
      service:
        name: sshd
        state: restarted
      when: result.rc == 0
```

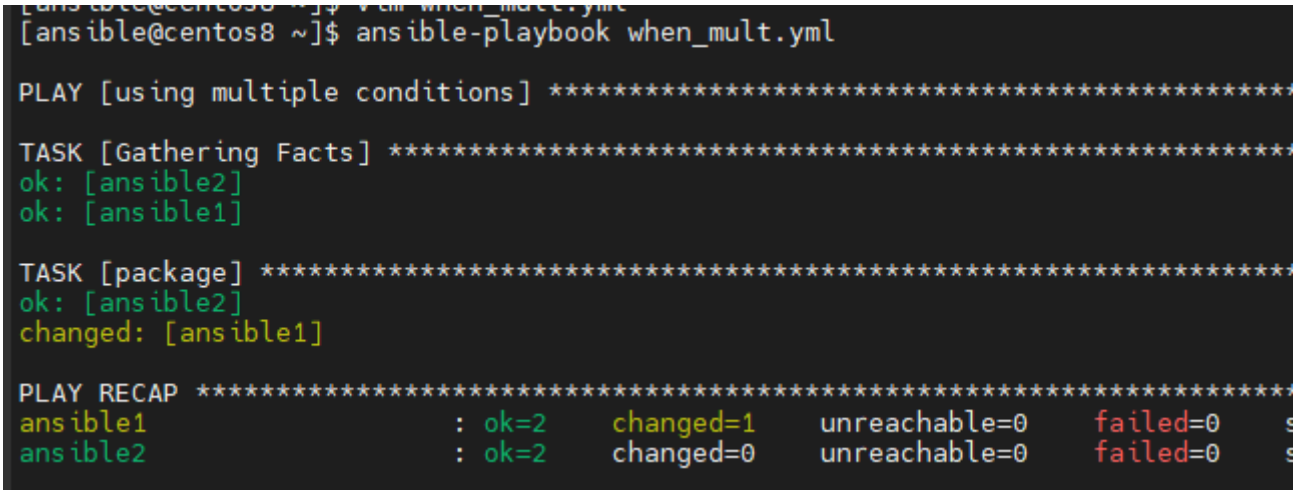

```yaml
- name: restart sshd only if crond is running
  gather_facts: yes
  hosts: all
  tasks:
    - name: get the crond server status
      command: /usr/bin/systemctl is-active crc
      ignore_errors: yes
      register: result
    - name: restart sshd based on crond status
      service:
        name: sshd
        state: restarted
      when: result.rc == 0
```

```
[ansible@centos8 ~]$ vim restart.yml
[ansible@centos8 ~]$ ansible-playbook restart.yml

PLAY [restart sshd only if crond is running] ***********************************

TASK [Gathering Facts] *********************************************************
ok: [ansible1]
ok: [ansible2]

TASK [get the crond server status] *********************************************
changed: [ansible2]
changed: [ansible1]

TASK [restart sshd based on crond status] **************************************
changed: [ansible2]
changed: [ansible1]

PLAY RECAP *********************************************************************
ansible1                   : ok=3    changed=2    unreachable=0    failed=0
ansible2                   : ok=3    changed=2    unreachable=0    failed=0
```

пример с несколькими условиями

```yaml
---
- name: using multiple conditions
  gather_facts: yes
  hosts: all
  tasks:
    - package:
        name: httpd
        state: installed
      when:
        - ansible_distribution == "CentOS"
        - ansible_memfree_mb > 256
```


```yaml
---
- name: using multiple conditions
  gather_facts: yes
  hosts: all
  tasks:
    - package:
        name: httpd
        state: installed
      when:
        - ansible_distribution == "CentOS"
        - ansible_memfree_mb > 256
```



и еще условие посложнее

```
---
- name: using multipe conditions
  gather_facts: yes
  hosts: all
  tasks:
    - package:
        name: httpd
        state: removed
      when: >
        ( ansible_distribution == "RedHat" and
          ansible_memfree_mb > 512 )
        or
        ( ansible_distribution == "CentOS" and
          ansible_memfree_mb < 100000 )
```

```
---
- name: using multipe conditions
  gather_facts: yes
  hosts: all
  tasks:
    - package:
        name: httpd
        state: removed
      when: >
        ( ansible_distribution == "RedHat" and
          ansible_memfree_mb > 512 )
        or
        ( ansible_distribution == "CentOS" and
          ansible_memfree_mb < 100000 )
```

```
[ansible@centos8 ~]$ ansible-playbook when_mult_complex.yml

PLAY [using multipe conditions] ************************************************

TASK [Gathering Facts] *********************************************************
ok: [ansible1]
ok: [ansible2]

TASK [package] *****************************************************************
changed: [ansible1]
changed: [ansible2]

PLAY RECAP *********************************************************************
ansible1                   : ok=2    changed=1    unreachable=0    failed=0
ansible2                   : ok=2    changed=1    unreachable=0    failed=0
```

Handler

В случае если все таски успешно пройдены, то словом `notify` запускаем `handler`

```
---
- name: set up web server
  hosts: all
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: latest
    - name: copy index.html
      copy:
        src: /tmp/index.html
        dest: /var/www/html/index.html
      notify:
        - restart_web
    - name: copy nothing - intended to fail
      copy:
        src: /tmp/nothing
        dest: /var/www/html/nothing.html
  handlers:
    - name: restart_web
      service:
        name: httpd
        state: restarted
```

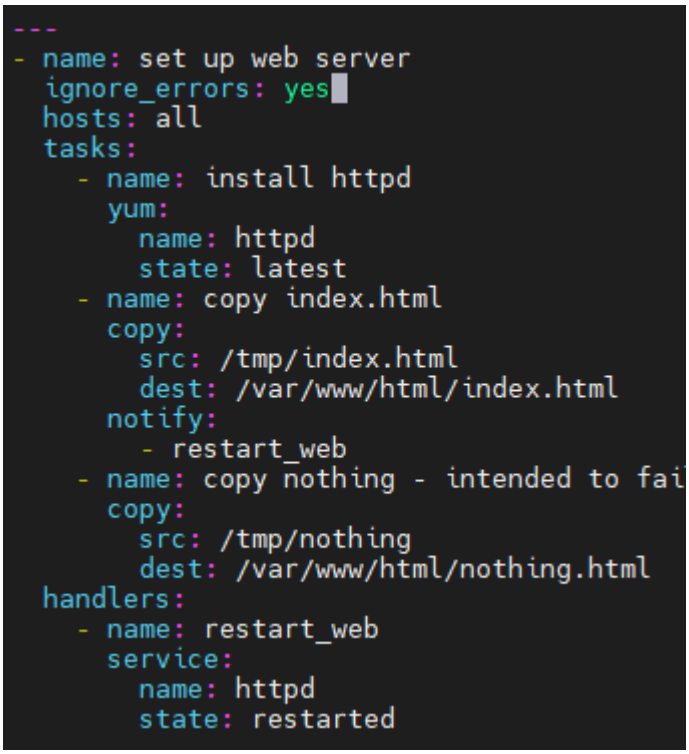
```yaml
---
- name: set up web server
  hosts: all
  tasks:
    - name: install httpd
      yum:
        name: httpd
        state: latest
    - name: copy index.html
      copy:
        src: /tmp/index.html
        dest: /var/www/html/index.html
      notify:
        - restart_web
    - name: copy nothing - intended to fail
      copy:
        src: /tmp/nothing
        dest: /var/www/html/nothing.html
  handlers:
    - name: restart_web
      service:
        name: httpd
        state: restarted
```

```
[ansible@centos8 ~]$ ansible-playbook handlers.yml

PLAY [set up web server] ********************************************************

TASK [install httpd] ************************************************************
changed: [ansible1]
changed: [ansible2]

TASK [copy index.html] **********************************************************
changed: [ansible2]
changed: [ansible1]

TASK [copy nothing - intended to fail] ******************************************
An exception occurred during task execution. To see the full traceback, use -vvv. The er
xpect the file to exist on the remote, see the remote_src option
fatal: [ansible1]: FAILED! => {"changed": false, "msg": "Could not find or access '/tmp/
you are using a module and expect the file to exist on the remote, see the remote_src op
An exception occurred during task execution. To see the full traceback, use -vvv. The er
xpect the file to exist on the remote, see the remote_src option
fatal: [ansible2]: FAILED! => {"changed": false, "msg": "Could not find or access '/tmp/
you are using a module and expect the file to exist on the remote, see the remote_src op

RUNNING HANDLER [restart_web] ***************************************************

PLAY RECAP **********************************************************************
ansible1                   : ok=2    changed=2    unreachable=0    failed=1    skipped=0
ansible2                   : ok=2    changed=2    unreachable=0    failed=1    skipped=0
```

- Видно, что т.к вторая таска упала, хендлер не запустился.
- Поправим плейбук

```yaml
---
- name: set up web server
  ignore_errors: yes
  hosts: all
  tasks:
```

```
    - name: install httpd
      yum:
        name: httpd
        state: latest
    - name: copy index.html
      copy:
        src: /tmp/index.html
        dest: /var/www/html/index.html
      notify:
        - restart_web
    - name: copy nothing - intended to fail
      copy:
        src: /tmp/nothing
        dest: /var/www/html/nothing.html
  handlers:
    - name: restart_web
      service:
        name: httpd
        state: restarted
```



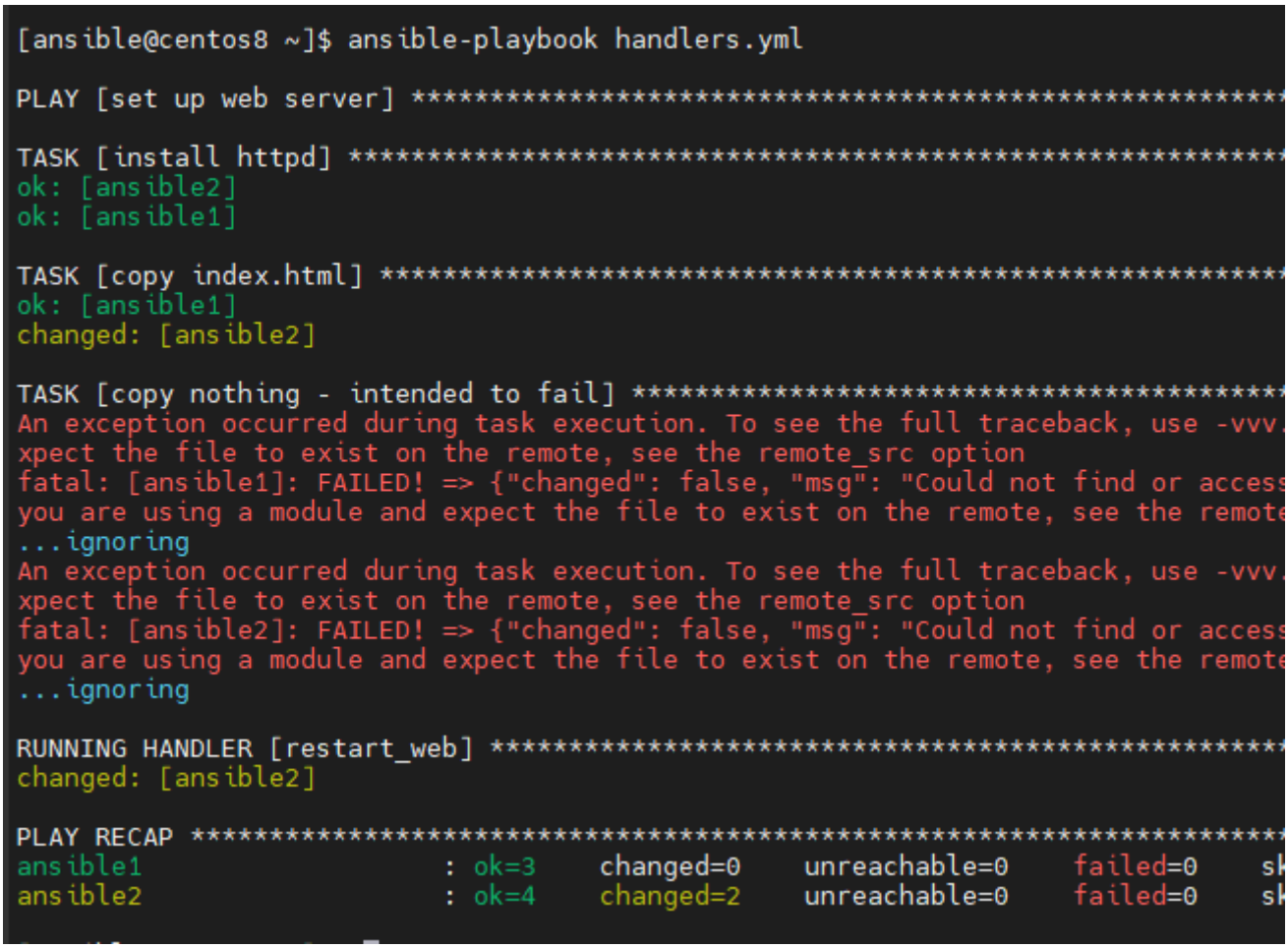
```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook handlers.yml

PLAY [set up web server] ******************************************************

TASK [install httpd] **********************************************************
ok: [ansible2]
ok: [ansible1]

TASK [copy index.html] ********************************************************
ok: [ansible2]
ok: [ansible1]

TASK [copy nothing - intended to fail] ****************************************
An exception occurred during task execution. To see the full traceback, use -vvv. The e
xpect the file to exist on the remote, see the remote_src option
fatal: [ansible1]: FAILED! => {"changed": false, "msg": "Could not find or access '/tmp
you are using a module and expect the file to exist on the remote, see the remote_src o
...ignoring
An exception occurred during task execution. To see the full traceback, use -vvv. The e
xpect the file to exist on the remote, see the remote_src option
fatal: [ansible2]: FAILED! => {"changed": false, "msg": "Could not find or access '/tmp
you are using a module and expect the file to exist on the remote, see the remote_src o
...ignoring

PLAY RECAP ********************************************************************
ansible1                   : ok=3    changed=0    unreachable=0    failed=0    skipped=
ansible2                   : ok=3    changed=0    unreachable=0    failed=0    skipped=
```

- Видим, что все таски завершились и ошибка игнорируется. Но хендлер не запустился. Почему? Потому что хедлер запускается только если таска, которая вызывает `notify` должна менять состояние хоста, если было бы `changed`, то хендлер бы вызвался. В принципе логично, т.к не было копирования файла (файл уже был скопирован ранее) нет необходимости в перезагрузке сервиса.
- Удалим файл из одного из хостов

```
ansible ansible2 -a "rm /var/www/html/index.html"
```

```
[ansible@centos8 ~]$ ansible ansible2 -a "rm /var/www/html/index.html"
[WARNING]: Consider using the file module with state=absent rather than running 'rm'.
is insufficient you can add 'warn: false' to this command task or set 'command_warnings
this message.
ansible2 | CHANGED | rc=0 >>
```

- Тут не указываем модуль, т.к если не передан модуль, значит по умолчанию модуль command
- Запустим плейбук заново



- И ура. У нас вызвался хендлер который рестартанул сервис.

Шаблоны (Templates)

```
---
- name: configure VSFTPD using a template
  hosts: all
  vars:
    anonymous_enable: yes
    local_enable: yes
    write_enable: yes
    anon_upload_enable: yes
  tasks:
    - name: install vsftpd
      yum:
        name: vsftpd
    - name: use template to copy FPT config
      template:
        src: vsftpd.j2
        dest: /etc/vsftpd/vsftpd.conf
```

```
---
- name: configure VSFTPD using a template
  hosts: all
  vars:
    anonymous_enable: yes
    local_enable: yes
    write_enable: yes
    anon_upload_enable: yes
  tasks:
    - name: install vsftpd
      yum:
        name: vsftpd
    - name: use template to copy FPT config
      template:
        src: vsftpd.j2
        dest: /etc/vsftpd/vsftpd.conf
```

- Шаблоны располагаются в папке `/templates`
- Шаблон в нашем случае выглядит так

```
anonymous_enable={{ anonymous_enable }}
local_enable={{ local_enable }}
write_enable={{ write_enable }}
anon_upload_enable={{ anon_upload_enable }}
dirmessage_enable=YES
xferlog_enable=YES
connect_from_port_20=YES
pam_service_name=vsftpd
userlist_enable=YES
#my IP address={{ ansible_facts['default_ipv4']['address'] }}
```

- Тут можно использовать переменные в том числе и из `ansible_facts`. Можно их писать через точку, а можно через квадратные скобки.

Операции с файлами

```
---
- name: file copy modules
  hosts: all
  tasks:
    - name: copy file demo
      copy:
        src: /etc/hosts
        dest: /tmp
    - name: add some lines to /tmp/hosts
      blockinfile:
        path: /tmp/hosts
        block: |
          192.168.4.110 host1.example.com
          192.168.4.120 hosts2.example.com
        state: present
    - name: verify file checksum
      stat:
        path: /tmp/hosts
```

```
        checksum_algorithm: md5
      register: result
    - debug:
        msg: "The checksum of /tmp/hosts is {{ result.stat.checksum }}"
    - name: fetch a file
      fetch:
        src: /tmp/hosts
        dest: /tmp
```

```
---
- name: file copy modules
  hosts: all
  tasks:
    - name: copy file demo
      copy:
        src: /etc/hosts
        dest: /tmp
    - name: add some lines to /tmp/hosts
      blockinfile:
        path: /tmp/hosts
        block: |
          192.168.4.110 host1.example.com
          192.168.4.120 hosts2.example.com
        state: present
    - name: verify file checksum
      stat:
        path: /tmp/hosts
        checksum_algorithm: md5
      register: result
    - debug:
        msg: "The checksum of /tmp/hosts is {{ result.stat.checksum }}"
    - name: fetch a file
      fetch:
        src: /tmp/hosts
        dest: /tmp
```

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook files.yml

PLAY [file copy modules] ********************************************************************

TASK [copy file demo] ***********************************************************************
changed: [ansible1]
changed: [ansible2]

TASK [add some lines to /tmp/hosts] *********************************************************
changed: [ansible2]
changed: [ansible1]

TASK [verify file checksum] *****************************************************************
ok: [ansible1]
ok: [ansible2]

TASK [debug] ********************************************************************************
ok: [ansible1] => {
    "msg": "The checksum of /tmp/hosts is 6f015bffd258c48650cd54fb08bf05b1"
}
ok: [ansible2] => {
    "msg": "The checksum of /tmp/hosts is 6f015bffd258c48650cd54fb08bf05b1"
}

TASK [fetch a file] *************************************************************************
changed: [ansible1]
changed: [ansible2]

PLAY RECAP **********************************************************************************
ansible1                   : ok=5    changed=3    unreachable=0    failed=0    skipped=0
ansible2                   : ok=5    changed=3    unreachable=0    failed=0    skipped=0
```

- Т.к запускали на нескольких хостах файлы скопируются в `/tmp` но для каждого хоста будет создан отдельный каталог

```
[ansible@centos8 ~]$ ls /tmp
ansible1  index.html
ansible2  systemd-private-e0a4f669
```

```
[ansible@centos8 ~]$ cat /tmp/ansible1/tmp/hosts
127.0.0.1   localhost localhost.localdomain localhost4 localhost4.localdomain4
::1         localhost localhost.localdomain localhost6 localhost6.localdomain6

127.0.0.1 centos8.localdomain
192.168.0.32 ansible1
192.168.0.33 ansible2
# BEGIN ANSIBLE MANAGED BLOCK
192.168.4.110 host1.example.com
192.168.4.120 hosts2.example.com
# END ANSIBLE MANAGED BLOCK
```

Добавились и комменты к блоку добавления через ansible

Сложный пример (не проверено)

```yaml
---
- name: create some users
  hosts: file
  vars_files:
    - vars/users
    - vars/groups
  tasks:
  - name: add groups
    group:
      name: "{{ item.groupname }}"
    loop: "{{ usergroups }}"
  - name: add users
    user:
      name: "{{ item.username }}"
      groups: "{{ item.groups }}"
    loop: "{{ users }}"
  - name: add SSH public keys
    authorized_key:
      user: "{{ item.username }}"
      key: "{{ lookup('file', 'files/'+ item.username + '/id_rsa.pub') }}"
    loop: "{{ users }}"
  - name: add sales group members to sudo
    copy:
      content: "%sales ALL=(ALL) NOPASSWD: ALL"
      dest: /etc/sudoers.d/sales
      mode: 0440
  - name: disable root SSH login
    lineinfile:
      dest: /etc/ssh/sshd_config
      regexp: "^PermitRootLogin"
      line: "PermitRootLogin no"
    notify: restart sshd
```

```yaml
handlers:
- name: restart sshd
  service:
    name: sshd
    state: restarted
```

```
[ansible@control lesson7]$ tree files/
files/
├── linda
│   └── id_rsa.pub
├── lisa
│   └── id_rsa.pub
├── lori
│   └── id_rsa.pub
└── lucy
    └── id_rsa.pub
```

```
[ansible@control lesson7]$ tree vars
vars
├── defaults
├── groups
└── users
```

```
[ansible@control lesson7]$ cat vars/users
---
users:
  - username: linda
    groups: sales
  - username: lori
    groups: sales
  - username: lisa
    groups: account
  - username: lucy
    groups: account
```

```
    groups: account
[ansible@control lesson7]$ cat vars/groups
usergroups:
  - groupname: sales
  - groupname: account
```

```
TASK [Gathering Facts] *********************************************************
ok: [ansible2.example.com]

TASK [add groups] **************************************************************
changed: [ansible2.example.com] => (item={'groupname': 'sales'})
changed: [ansible2.example.com] => (item={'groupname': 'account'})

TASK [add users] ***************************************************************
changed: [ansible2.example.com] => (item={'username': 'linda', 'groups': 'sales'})
changed: [ansible2.example.com] => (item={'username': 'lori', 'groups': 'sales'})
changed: [ansible2.example.com] => (item={'username': 'lisa', 'groups': 'account'})
changed: [ansible2.example.com] => (item={'username': 'lucy', 'groups': 'account'})

TASK [add SSH public keys] *****************************************************
changed: [ansible2.example.com] => (item={'username': 'linda', 'groups': 'sales'})
changed: [ansible2.example.com] => (item={'username': 'lori', 'groups': 'sales'})
changed: [ansible2.example.com] => (item={'username': 'lisa', 'groups': 'account'})
changed: [ansible2.example.com] => (item={'username': 'lucy', 'groups': 'account'})

TASK [add sales group members to sudo] *****************************************
changed: [ansible2.example.com]

TASK [disable root SSH login] **************************************************
changed: [ansible2.example.com]

RUNNING HANDLER [restart sshd] *************************************************
changed: [ansible2.example.com]

PLAY RECAP *********************************************************************
ansible2.example.com       : ok=7    changed=6    unreachable=0    failed=0    skipped=0
```

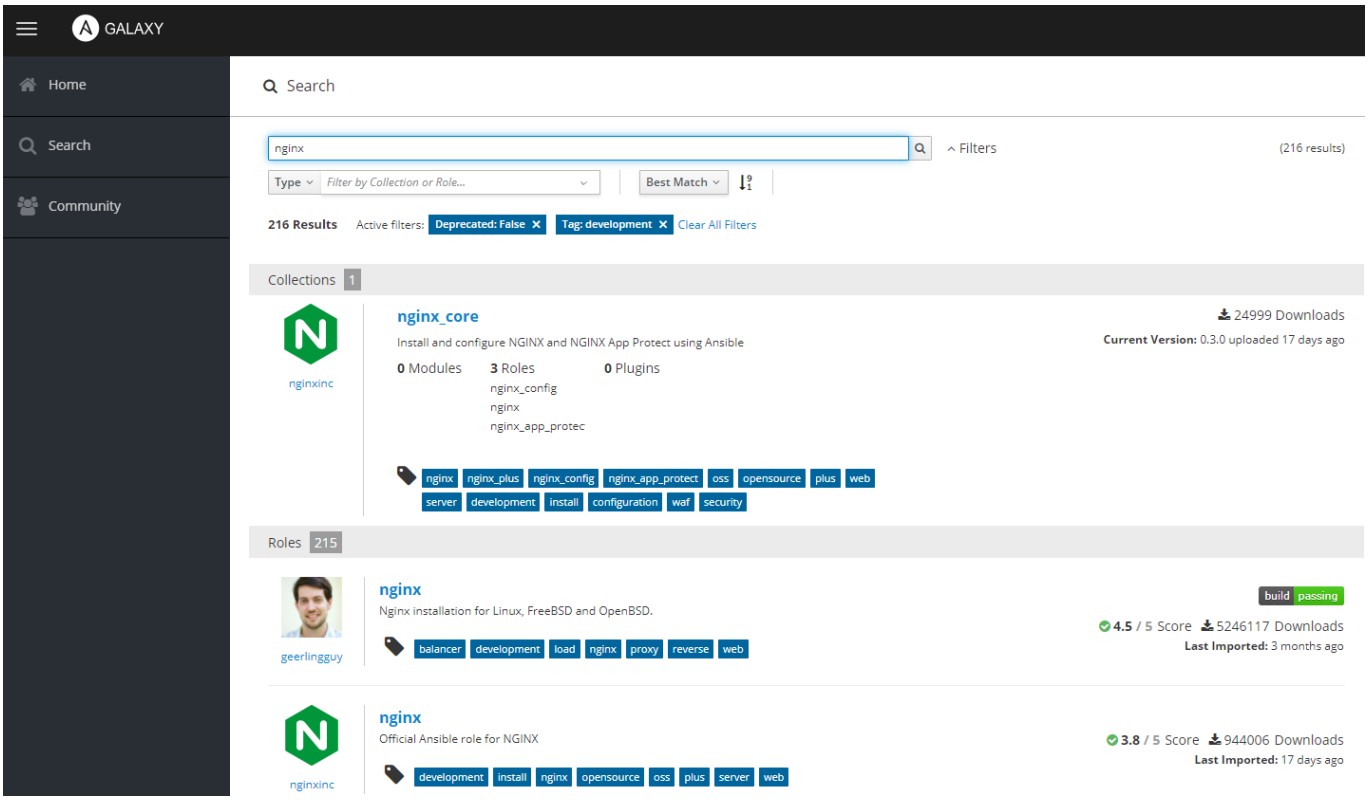
Ansible roles

Роли позволяют существенно облегчить работу с ansible

(Ansible Galaxy)

Существует сайт `ansible galaxy` https://galaxy.ansible.com/

где есть всевозможные плейбуки написанные другими.



можно поискать командой:

```
ansible-galaxy seach name_of_roles
```

```
[ansible@centos8 vagrant]$ ansible-galaxy search nginx

Found 1533 roles matching your search. Showing first 1000.

 Name                                                  Description
 ----                                                  -----------
 0x0i.prometheus                                       Prometheus - a multi-dimensional time-series data monitoring and
 0x5a17ed.ansible_role_netbox                          Installs and configures NetBox, a DCIM suite, in a production set
 1davidmichael.ansible-role-nginx                      Nginx installation for Linux, FreeBSD and OpenBSD.
 1it.sudo                                              Ansible role for managing sudoers
 1mr.zabbix_host                                       configure host zabbix settings
 1nfinitum.php                                         PHP installation role.
 2goobers.jellyfin                                     Install Jellyfin on Debian.
 2kloc.trellis-monit                                   Install and configure Monit service in Trellis.
 4linuxdevops.web-balancer                             Instalacao e Configuracao do servidor Nginx Load Balancer
 aadl.docker-nginx-alpine                              Ansible role to manage and run the alpine nginx docker container.
 aalaesar.install_nextcloud                            Add a new Nextcloud instance in your infrastructure. The role man
 aalaesar.upgrade-nextcloud                            Upgrade an Nextcloud instance in your infrastructure. The role ma
 AAROC.discourse-sso                                   This is a role to deploy the [Discourse](http://www.discourse.org
```

есть команда ansible-galaxy которая может скачать роль (набор файлов плейбука) с этого сайта.

```
ansible-galaxy install geerlingguy.nginx
```

```
[ansible@centos8 vagrant]$ ansible-galaxy install geerlingguy.nginx
```

```
[ansible@centos8 ~]$ ll ./.ansible/roles/geerlingguy.nginx/
total 16
drwxrwxr-x. 2 ansible ansible    22 Jan 28 06:45 defaults
drwxrwxr-x. 2 ansible ansible    22 Jan 28 06:45 handlers
-rw-rw-r--. 1 ansible ansible  1080 Jul 22  2020 LICENSE
drwxrwxr-x. 2 ansible ansible    50 Jan 28 06:45 meta
drwxrwxr-x. 3 ansible ansible    21 Jan 28 06:45 molecule
-rw-rw-r--. 1 ansible ansible 11497 Jul 22  2020 README.md
drwxrwxr-x. 2 ansible ansible   210 Jan 28 06:46 tasks
drwxrwxr-x. 2 ansible ansible    64 Jan 28 06:45 templates
drwxrwxr-x. 2 ansible ansible   101 Jan 28 06:45 vars
[ansible@centos8 ~]$
```

Для использование роли нужно написать плейбук и вызвать роль.

```yaml
---
- name: use galxy nginx role
  gather_facts: yes
  hosts: all
  roles:
    - role: geerlingguy.nginx
```

```yaml
---
- name: use galxy nginx role
  hosts: all
  roles:
    - role: geerlingguy.nginx
```

```
[ansible@centos8 ~]$ [ansible@centos8 ~]$ ansible-playbook install-nginx.yml

PLAY [use galxy nginx role] ****************************************************

TASK [Gathering Facts] *********************************************************
ok: [ansible1]
ok: [ansible2]

TASK [geerlingguy.nginx : Include OS-specific variables.] **********************
ok: [ansible1]
ok: [ansible2]

TASK [geerlingguy.nginx : Define nginx_user.] **********************************
ok: [ansible1]
ok: [ansible2]

TASK [geerlingguy.nginx : include_tasks] ***************************************
included: /home/ansible/.ansible/roles/geerlingguy.nginx/tasks/setup-RedHat.yml for

TASK [geerlingguy.nginx : Enable nginx repo.] **********************************
changed: [ansible2]
changed: [ansible1]

TASK [geerlingguy.nginx : Ensure nginx is installed.] **************************
ok: [ansible2]
ok: [ansible1]

TASK [geerlingguy.nginx : include_tasks] ***************************************
skipping: [ansible1]
skipping: [ansible2]

TASK [geerlingguy.nginx : include_tasks] ***************************************
skipping: [ansible1]
skipping: [ansible2]

TASK [geerlingguy.nginx : include_tasks] ***************************************
skipping: [ansible1]
skipping: [ansible2]

TASK [geerlingguy.nginx : include_tasks] ***************************************
skipping: [ansible1]
skipping: [ansible2]

TASK [geerlingguy.nginx : include_tasks] ***************************************
skipping: [ansible1]
skipping: [ansible2]

TASK [geerlingguy.nginx : Remove default nginx vhost config file (if configured).]
skipping: [ansible1]
skipping: [ansible2]
```

Создание кастомных ролей

```
ansible-galaxy init hello
```

```
[ansible@centos8 ~]$ ansible-galaxy init hello
- Role hello was created successfully
```

```
[ansible@centos8 ~]$ cd hello/
[ansible@centos8 hello]$ tree
.
├── defaults
│   └── main.yml
├── files
├── handlers
│   └── main.yml
├── meta
│   └── main.yml
├── README.md
├── tasks
│   └── main.yml
├── templates
├── tests
│   ├── inventory
│   └── test.yml
└── vars
    └── main.yml
```

Использование системных ролей

Почти каждый дистрибутив линукса имеет набор системных ролей. В нем роли для администрирования этого дистрибутива. Если приходится поддерживать множество серверов они могут быть очень полезны - так например есть роли для настройки сети.

```
sudo yum search system-role
sudo yum install rhel-system-roles
```

```
[ansible@centos8 motd]$ sudo yum search system-role
Last metadata expiration check: 0:17:02 ago on Thu 28 Jan 2021 07:08:45 PM UTC.
================================================= Name Matched: system-role ====
rhel-system-roles.noarch : Set of interfaces for unified system management
```

```
[ansible@centos8 motd]$ sudo yum install rhel-system-roles
Last metadata expiration check: 0:17:35 ago on Thu 28 Jan 202
Dependencies resolved.
================================================================
 Package                            Architecture
================================================================
Installing:
 rhel-system-roles                  noarch

Transaction Summary
================================================================
Install  1 Package

Total download size: 503 k
Installed size: 3.1 M
Is this ok [y/N]: y
Downloading Packages:
rhel-system-roles-1.0-20.el8.noarch.rpm
----------------------------------------------------------------
Total
Running transaction check
Transaction check succeeded.
Running transaction test
Transaction test succeeded.
Running transaction
  Preparing        :
  Installing       : rhel-system-roles-1.0-20.el8.noarch
  Verifying        : rhel-system-roles-1.0-20.el8.noarch

Installed:
  rhel-system-roles-1.0-20.el8.noarch

Complete!
```

Посмотрим содержимое установочного пакета

```
sudo rpm -ql rhel-system-roles
```

```
[ansible@centos8 motd]$ sudo rpm -ql rhel-system-roles
/usr/share/ansible
/usr/share/ansible/roles
/usr/share/ansible/roles/linux-system-roles.certificate
/usr/share/ansible/roles/linux-system-roles.kdump
/usr/share/ansible/roles/linux-system-roles.kernel_settings
/usr/share/ansible/roles/linux-system-roles.logging
/usr/share/ansible/roles/linux-system-roles.metrics
/usr/share/ansible/roles/linux-system-roles.nbde_client
/usr/share/ansible/roles/linux-system-roles.nbde_server
/usr/share/ansible/roles/linux-system-roles.network
/usr/share/ansible/roles/linux-system-roles.postfix
/usr/share/ansible/roles/linux-system-roles.selinux
/usr/share/ansible/roles/linux-system-roles.storage
/usr/share/ansible/roles/linux-system-roles.timesync
/usr/share/ansible/roles/linux-system-roles.tlog
/usr/share/ansible/roles/rhel-system-roles.certificate
/usr/share/ansible/roles/rhel-system-roles.certificate/.flake8
/usr/share/ansible/roles/rhel-system-roles.certificate/.github
/usr/share/ansible/roles/rhel-system-roles.certificate/.github/CODEOWNERS
/usr/share/ansible/roles/rhel-system-roles.certificate/.gitignore
/usr/share/ansible/roles/rhel-system-roles.certificate/.lgtm.yml
/usr/share/ansible/roles/rhel-system-roles.certificate/.pre-commit-config.yaml
/usr/share/ansible/roles/rhel-system-roles.certificate/.pydocstyle
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis/config.sh
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis/custom.sh
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis/custom_pylint.py
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis/preinstall
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis/runblack.sh
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis/runcoveralls.sh
/usr/share/ansible/roles/rhel-system-roles.certificate/.travis/runflake8.sh
```

```
[ansible@centos8 motd]$ cd /usr/share/ansible/roles/
[ansible@centos8 roles]$ ll
total 36
lrwxrwxrwx.  1 root root   29 Sep 25 20:04 linux-system-roles.certificate
lrwxrwxrwx.  1 root root   23 Sep 25 20:04 linux-system-roles.kdump -> rhe
lrwxrwxrwx.  1 root root   33 Sep 25 20:04 linux-system-roles.kernel_setti
lrwxrwxrwx.  1 root root   25 Sep 25 20:04 linux-system-roles.logging -> r
lrwxrwxrwx.  1 root root   25 Sep 25 20:04 linux-system-roles.metrics -> r
lrwxrwxrwx.  1 root root   29 Sep 25 20:04 linux-system-roles.nbde_client
lrwxrwxrwx.  1 root root   29 Sep 25 20:04 linux-system-roles.nbde_server
lrwxrwxrwx.  1 root root   25 Sep 25 20:04 linux-system-roles.network -> r
lrwxrwxrwx.  1 root root   25 Sep 25 20:04 linux-system-roles.postfix -> r
lrwxrwxrwx.  1 root root   25 Sep 25 20:04 linux-system-roles.selinux -> r
lrwxrwxrwx.  1 root root   25 Sep 25 20:04 linux-system-roles.storage -> r
lrwxrwxrwx.  1 root root   26 Sep 25 20:04 linux-system-roles.timesync ->
lrwxrwxrwx.  1 root root   22 Sep 25 20:04 linux-system-roles.tlog -> rhel
drwxr-xr-x. 12 root root 4096 Jan 28 19:26 rhel-system-roles.certificate
drwxr-xr-x.  9 root root  156 Jan 28 19:26 rhel-system-roles.kdump
drwxr-xr-x. 11 root root 4096 Jan 28 19:26 rhel-system-roles.kernel_settin
drwxr-xr-x.  9 root root 4096 Jan 28 19:26 rhel-system-roles.logging
drwxr-xr-x. 14 root root 4096 Jan 28 19:26 rhel-system-roles.metrics
drwxr-xr-x. 11 root root 4096 Jan 28 19:26 rhel-system-roles.nbde_client
drwxr-xr-x. 10 root root 4096 Jan 28 19:26 rhel-system-roles.nbde_server
drwxr-xr-x. 10 root root 4096 Jan 28 19:26 rhel-system-roles.network
drwxr-xr-x.  6 root root  114 Jan 28 19:26 rhel-system-roles.postfix
drwxr-xr-x.  8 root root  138 Jan 28 19:26 rhel-system-roles.selinux
drwxr-xr-x. 11 root root 4096 Jan 28 19:26 rhel-system-roles.storage
drwxr-xr-x. 11 root root  187 Jan 28 19:26 rhel-system-roles.timesync
drwxr-xr-x. 10 root root 4096 Jan 28 19:26 rhel-system-roles.tlog
```