

# Ansible

- [Установка и настройка](#)
  - [Установка](#)
  - [Настройка ansible](#)
- [Запуск и элементы управления ansible скриптами](#)
- [Модули](#)
  - [Общая информация](#)
  - [Системные модули и скрипты](#)
  - [Git и pip](#)
  - [Docker](#)
- [Роли](#)

????????? ? ???????????

Установка и настройка

????????

### На Debian-подобных системах:

```
apt-get install ansible
```

### Настройка управляемых хостов

На каждом хосте:

1. Если sudo нет, то

```
apt install sudo
```

2. Если создается новый пользователь, то

```
sudo useradd -G sudo -s /bin/bash ansibleuser
```

3. Если существующий пользователь, то добавить в группу sudo:

```
usermod -aG sudo sergey
```

4. Разрешить пользователю повышать привилегии без ввода пароля. Создать файл

```
sudo nano /etc/sudoers.d/ansibleuser
```

5. И добавить текст

```
ansibleuser ALL=(ALL) NOPASSWD:ALL
```

Теперь можно использовать в скриптах `become: true`

### Настройка сервера

```
ssh-keygen  
ssh-copy-id username@remote_host
```

1. Создать ключ доступа
2. Скопировать на каждый управляемый хост, используя логин на хост. Удобнее, если для задач управления на всех хостах сделать одинаковый логин.

В настройке каждого хоста нужно использовать соответствующий логин, для которого настроен ключ доступа.

# ????????? ansible

## Приоритет поиска файла настроек

1. ANSIBLE\_CONFIG (environment variable if set)
2. ansible.cfg (in the current directory)
3. ~/.ansible.cfg (in the home directory)
4. /etc/ansible/ansible.cfg (default)

## Генерация настроек

```
ansible-config init --disabled > ansible.cfg
```

## С плагинами

```
ansible-config init --disabled -t all > ansible.cfg
```

# и ; комментарии. Но ; обычно используется для комментария значения по умолчанию.

## Файл инвентаризации

hosts

Размещение файла может задаваться переменной окружения \$ANSIBLE\_HOSTS, либо ключ -i при запуске

После : пишется тип данных для группы. Например группа [example], [example:vars] это переменные

:vars переменные для группы

:children группы-потомки

## Пример

```
[example] группа
www.example.com сервер

# Group 'multi' with all servers
[multi:children]
example

#5 servers in one line [a:z] or [A:Z], or numbers with specific digits, such as [001:250].
[dyngroup]
Node[0:4].lab.edu

# Variables that will be applied to all servers
```

```
[multi:vars]
ansible_ssh_user=vagrant
```

### Файлы размещения переменных:

Файлы с переменными групп хранятся в директории "group\_vars/имя\_группы";

Файлы с переменными хостов в директории "hosts\_vars/имя\_хоста";

Формат файла одинаковый, дочерние переменные заменяются на родительские.

```
---
ansible_user=setup
ansible_private_ssh_key=/home/user/ansible.key
```

Может применяться и плоское определение, и вложенное, тогда в скрипте {{ db.user }}:

```
db: lll
  user: ppp
```

### Динамический реестр:

Должен поддерживать следующий интерфейс:

```
--host=hostname - реестр выдает список в JSON { "ansible_ssh_host": "127.0.0.1",
"ansible_ssh_port": 2200, "ansible_ssh_user": "vagrant" }
--list реестр выдает список групп {"staging": [ "ontaro.example.com",
"quebec.example.com"], "vagrant": [ "vagrant1", "vagrant2", "vagrant3"]}
"_meta" : {"hostvars" : {"vagrant1" : {"ansible_ssh_host": "127.0.0.1", "ansible_ssh_port":
2222, "ansible_ssh_user": "vagrant"}, "vagrant2": {"ansible_ssh_host": "127.0.0.1",
"ansible_ssh_port": 2200, "ansible_ssh_user": "vagrant"}}}
```

Если создать папку inventory и добавить в ansible.cfg параметр hostfile = inventory то статический hosts и динамический будут объединены

Если включить сбор фактов gather\_facts: True то можно во время выполнения задач группировать хосты (например debian/centos)

Использование динамического реестра (в примере my\_dynamic\_inventory.py должен быть исполняемым скриптом)

```
ansible-playbook -i my_dynamic_inventory.py my_playbook.yml
```

В Ссылках есть материал по динамическим реестрам

### Предустановленные переменные

Переменная	Использование
ansible_user	пользователь, от имени которого выполняются задачи на хостах данной группы. Значение в файле задач игнорируется.

ansible_host	Сопоставление ip-имени
ansible_port	Порт доступа по ssh
ansible_connection	Может быть ssh, local, docker (запуск команд непосредственно на контейнере)
ansible_become	+ sudo
ansible_become_user	sudo -> another user
ansible_ssh_private_key_file	адрес ключа

**Ссылки:**

[Описание основного config файла](#)

[Динамический реестр](#)

?????? ? ??????????

????????????? ansible ??????????????

### Запуск одного действия

ansible [host/group/all] [action parameters]

#### Действия

-a "shell command" shell command

-m exec\_name installed module

-i <filename> inventory file

--become все команды + sudo

--become-user user1 обязательно с become, переключается на выбранного пользователя

-f 1 использование одного потока. По умолчанию процессы параллельно для всех хостов.

#### Пример:

```
ansible multi -a "hostname"
```

### Запуск через ansible playbook

Список действий сохраняется в файле \*.yaml (playbook)

```
ansible-playbook filename.yaml
```

### Структура playbook файла

```
---
- hosts: [host ip | host group | all]
  remote_user: []
  become: true
  tasks:
    - [task 1]
    - [task n]
---
- hosts: [host ip | host group | all]
  remote_user: []
  become: true
  tasks:
    - [task 1]
    - [task n]
```

### Параметры задачи

gather\_facts: False Проводить сбор данных по умолчанию  
ignore\_errors: True Прекращать или нет работу при ошибке

## Переменные из модулей

```
---
- name: Check if host is up and running
  hosts: all
  become: false
  gather_facts: false

  tasks:
    - name: Ping the remote server
      ping:
        register: result
        until: result is succeeded
        retries: 3
        delay: 10

    - name: Get system uptime
      command: uptime
      when: result is succeeded
      register: uptime_result

    - name: Show system uptime
      debug:
        var: uptime_result.stdout
      when: result is succeeded
```

10: полученное значение из модуля ping сохраняется в переменной result

11: данное задание выполняется до тех пор, пока result не будет succeeded, но с ограничением кол-ва попыток (12) и задержкой между попытками (13)

17: задание 15 выполняется если result succeeded

18: полученное значение заносится в переменную uptime\_result

21: вывод значения uptime\_result

## Условия (handlers)

```
when: ansible_os_family == "Debian"
```

## Получить список фактов:

```
- name: Show facts available on the system
  ansible.builtin.debug:
  var: ansible_facts
```

В папке /etc/ansible/facts.d/\*.fact файлы с локальными фактами о машине. Обращение к фактам в yaml:

```
ansible_local.<fact file name>.<fact group inside file>.<varname>
```

Создание новой переменной:

```
set_fact:<valname>=<...>
```

## Циклы (loops)

Цикл с объявленной переменной:

```
- name: use apt to install multiple apps
  apt:
  name: '{{ app }}'
  state: latest
  update_cache: yes
  vars:
    app:
    - htop
    - mc
    - nload
  become: yes
```

Цикл с необъявленной переменной:

```
- name: use apt to install multiple apps
  apt:
  name: '{{ item }}'
  state: latest
  update_cache: yes
  loop:
  - htop
  - mc
```

```
- nload
become: yes
```

## Цикл с переменной-словарем

```
- name: install python packages
  pip: name={{item.name}} version={{item.version}}
  become: True
  with_items:
    - {name=mezzanine, version=1.2.1}
    - {name=guincorn, version=3.2.9}
```

## Проверка ошибок выполнения

Проверяет условия и останавливает выполнение при нарушении

```
- name: Проверка значения переменной
  assert:
    that:
      - my_variable == "expected_value"
```

## Примеры из документации:

```
- name: A single condition can be supplied as string instead of list
  ansible.builtin.assert:
    that: "ansible_os_family != 'RedHat'"

- name: Use yaml multiline strings to ease escaping
  ansible.builtin.assert:
    that:
      - "'foo' in some_command_result.stdout"
      - number_of_the_counting == 3
      - >
        "reject" not in some_command_result.stderr

- name: After version 2.7 both 'msg' and 'fail_msg' can customize failing assertion
  message
  ansible.builtin.assert:
    that:
      - my_param <= 100
```

```
- my_param >= 0
fail_msg: "'my_param' must be between 0 and 100"
success_msg: "'my_param' is between 0 and 100"
```

```
- name: Please use 'msg' when ansible version is smaller than 2.7
```

```
ansible.builtin.assert:
```

```
  that:
```

```
    - my_param <= 100
```

```
    - my_param >= 0
```

```
  msg: "'my_param' must be between 0 and 100"
```

```
- name: Use quiet to avoid verbose output
```

```
ansible.builtin.assert:
```

```
  that:
```

```
    - my_param <= 100
```

```
    - my_param >= 0
```

```
  quiet: true
```

??????

## Модули

????? ????????????

Модули - созданный на python скрипт, упрощающий конкретную задачу.

Модули возвращают значения, название модуля должно быть под name. Есть общие переменные и специфичные для модулей

### Справка по модулю

```
ansible-doc apt
```

### Список модулей

```
ansible-doc -l
```

### Пример проверки необходимости перезагрузки

```
- name: check if reboot is required
  become: yes
  become_method: sudo
  shell: "[ -f /var/run/reboot-required ]"
  failed_when: False
  register: reboot_required
  changed_when: reboot_required.rc == 0
  notify: reboot

handlers:
- name: reboot
  command: shutdown -r now "Ansible triggered reboot after system updated"
  async: 0
  poll: 0
  ignore_errors: true
```

### Ссылки

[Информация на русском языке о модулях](#)

?????????? ???? ? ???? ?

### Установка и/или проверка установки apt пакета

**Название модуля:** apt

**Переменные:**

- name: ntp #имя проверяемого и устанавливаемого модуля
- state: present #состояние после завершения
- update\_cache: yes # обновлять ли кэш

```
- name: Install module
  apt:
    name: ntp
    state: present
    update_cache: yes
```

### Добавить ключ стороннего репозитория

```
- name: Add Docker GPG key
  apt_key:
    url: https://download.docker.com/linux/ubuntu/gpg
```

### Добавить сторонний репозиторий

```
- name: Add Docker repository
  apt_repository:
    repo: deb [arch=amd64] https://download.docker.com/linux/ubuntu bionic stable
```

## Обновить apt кэш

```
- name: Update apt cache
  apt: update_cache=yes
```

## Копирование файлов

### Копирование с локального на удаленный

```
- name: Copy from server to client
  copy:
    src: /home/user/file.txt
    dest: /home/setup/file.txt
    owner: foo
    group: foo
    mode: '0644'
```

### Копирование с удаленного на локальный

```
- name: Copy from client to server
  fetch:
    src: /var/log/access.log
    dest: /var/log/fetched
    flat: true удалить структуру родительских папок для файла
```

## Создание пользователя и группы

### Создание пользователя

```
- name: Create and/or check presence user
  user:
    name: install
    state: present
    shell: /bin/bash
    group: sudo
    system: yes
```

```
hidden: yes
ssh_key_file: .ssh/id_rsa
expires: -1
```

## Создание группы

```
- name: Create check group
group:
  name: clustergroup
  state: present
  gid: 1040
```

## Управление сервисами (daemon)

```
- name: Update sysctl
sysctl:
  name: net.ipv4.ip_forward
  value: 1
  sysctl_set: yes
  state: present
  reload: yes
```

```
- name: Set or check service
service:
  name: ntp
  state: started
  enabled: yes
```

```
- name: Set check daemon starting
systemd:
  name: ntp
  state: started
  enabled: yes
  masked: no
  daemon_reload: yes
  register: systemd
```

cron:

## Скрипты и консольные команды

```
- name: Raw command
raw: echo "this was written by a raw Ansible module!!" >> ~/raw.txt
```

```
- name: Executing script
shell: ./shell_script.sh >> ~/shell.txt
  args:
  chdir: /usr/local/
  creates: ~/shell.txt
  executable: /bin/csh
```

```
- name: Executing python script
script: ./shell_script.py --some-argumets "42"
  args:
  creates: ~/shell.txt
  executable: python
```

Для исполнения expect скриптов нужно сначала проверить и установить пакет expect

```
- name: Expect module
expect:
  command: passwd user1
  responses:
  (?i)password: "Ju5tAn07herP@55w0rd":
```

# Git ? pip

## Git

```
- name: Clone update repo
git:
  repo: https://github.com/ansible/ansible.git
  dest: /usr/local/ansible
  clone: yes
  update: yes
```

## Pip

Из официального репозитория:

```
- name: Install python package
pip:
  name: numpy
  version: 0.3
```

Из внешнего источника

```
- name: install a python library from a github
pip:
  name: https://github.com/jakubroztocil/httpie
```

# Docker

## Необходимые модули:

```
pip install 'docker-py>=1.7.0'
```

```
pip install 'docker-compose>=1.7.0'
```

ansible-container позволяет работать с docker без dockerfile

### Создание контейнера

```
- name: create a container
  docker_container:
    name: debianlinux
    image: debian:9
    pull: yes
    state: present
```

### Запуск контейнера

```
- name: start a container
  docker_container:
    name: debianlinux
    state: started
    devices:
      - "/dev/sda:/dev/xvda:rwm"
```

### Остановка контейнера

```
- name: stop a container
  docker_container:
    name: debianlinux
    state: stopped
```

### Удаление образа из локального хранилища

```
- name: remove a container image
  docker_image:
    name: labimages/ubuntu
    state: absent
    tag: lab16
```

## Авторизация на docker hub

```
- name: login to DockerHub
  docker_login:
    username: labuser1
    password: "L@bp@55w0rd"
    email: user1@lab.edu
```

## Скачать образ из docker hub

```
- name: pull a container image
  docker_image:
    name: ubuntu:18.04
    pull: yes
```

## Сохранить образ в docker hub

```
- name: push a container image to docker hub
  docker_image:
    name: labimages/ubuntu
    repository: labimages/ubuntu
    tag: lab18
    push: yes
```

# ????

## Структура роли и добавление в playbook

1. Создается папка roles, внутри папки с названиями ролей, внутри каждой папки - defaults files handlers meta templates tasks vars
2. в playbook добавляется

```
roles:  
  - имя
```

## Назначения папок

Папка	Назначение
defaults	позволяет устанавливать переменные по умолчанию для включенных или зависимых ролей.
files	содержит статические файлы и файлы сценариев, которые могут быть скопированы на удаленный сервер или выполнены на нём.
handlers	все обработчики, которые ранее были в вашем плейбуке, теперь могут быть добавлены в каталог.
meta	для метаданных роли, которые используются для управления зависимостями. Например, вы можете определить список ролей, которые должны быть применены до вызова текущей роли.
templates	для шаблонов, которые генерируют файлы на удаленных хостах.
tasks	содержит один или несколько файлов с задачами, которые определяются в разделе tasks обычного плейбука Ansible. Эти задачи могут напрямую ссылаться на файлы и шаблоны, содержащиеся в соответствующих каталогах внутри роли, без необходимости указывать полный путь к файлу.
vars	переменные для роли могут быть указаны в файлах внутри каталога, а затем ссылаться на них в другом месте роли.

## Ссылки:

[Пример преобразования одного файла в роль](#)