

# DevOps

Л06. Ansible - advanced

П06. Vars, handlers, debug, vault, j2 templates

Виктор Моисеев  
+7-902-83-145-30  
[t.me/v\\_paranoid](https://t.me/v_paranoid)  
[victorparanoid@gmail.com](mailto:victorparanoid@gmail.com)

# План курса

1. Введение в DevOps
2. Базовое администрирование Linux
3. Системы контроля версионности кода (git)
4. Оркестровка (Ansible)
5. Контейнеризация (docker)
6. Микросервисная архитектура и оркестровка контейнеров (k8s)
7. Непрерывная интеграция и доставка (CI/CD, Github Actions, ArgoCD)
8. Инфраструктура как код (IaC, Terraform)
9. Мониторинг (Prometheus)

# 04. Ansible - advanced

1. Vault
2. Debug
3. Vars
4. Loops and conditionals
5. Handlers
6. Error handling
7. Jinja Templates

404 amarao 2 июл 2020 в 13:21

**Основы Ansible, без которых ваши плейбуки — комок слипшихся макарон**

 12 мин  142K

# Ansible – Playbook



---

```
- name: play 1
  hosts: webserver
  remote_user: runner
  become: true
  become_method: sudo
  gather_facts: no

tasks:
- name: install Nginx
  ansible.builtin.apt:
    name: nginx-extras
    update_cache: yes
- name: Start Nginx
  service: nginx
  state: enabled
- name: Print message
  ansible.builtin.debug:
    msg: Hello world
```

В какого юзера логиниться

Повышать ли привилегии и как

Собирать ли информацию о системе

Отладка на контрольной ноде

# Ansible – Inventory

```
inventory.ini
```

```
[webservers]
```

```
192.0.2.50
```

```
192.0.2.51
```

```
192.0.2.52
```

```
[db]
```

```
10.0.5.1
```

```
10.0.5.2
```

```
10.0.5.3
```

```
inventory.yml
```

```
myhosts:
```

```
  hosts:
```

```
    my_host_01:
```

```
      ansible_host: 192.0.2.50
```

```
    my_host_02:
```

```
      ansible_host: 192.0.2.51
```

```
    my_host_03:
```

```
      ansible_host: 192.0.2.52
```

# Ansible – Пример

Тестовый прогон без изменений:

```
ansible-playbook ./playbook.yml -i ./inventory.ini --check --diff
```

Продуктовый прогон:

```
ansible-playbook ./playbook.yml -i ./inventory.ini
```

**--ask-become-pass** – если на удаленной стороне требуется ввод пароля

# Ansible Vault

Инструмент для управления секретами

Зашифровать(расшифровать):

- файл
- строку

```
victor@ubuntu:~/lab-ansible-6$ ansible-vault
usage: ansible-vault [-h] [--version] [-v] {create,decrypt,edit,view,encrypt,encrypt_string,rekey}
ansible-vault: error: the following arguments are required: action

usage: ansible-vault [-h] [--version] [-v] {create,decrypt,edit,view,encrypt,encrypt_string,rekey}
encryption/decryption utility for Ansible data files

positional arguments:
  {create,decrypt,edit,view,encrypt,encrypt_string,rekey}
    create              Create new vault encrypted file
    decrypt             Decrypt vault encrypted file
    edit               Edit vault encrypted file
    view              View vault encrypted file
    encrypt            Encrypt YAML file
    encrypt_string      Encrypt a string
    rekey              Re-key a vault encrypted file

options:
  --version            show program's version number, config file location, config
                      location, executable location and exit
  -h, --help          show this help message and exit
  -v, --verbose        Causes Ansible to print more debug messages. Adding multiple
                      the builtin plugins currently evaluate up to -vvvvvv. A reasonable
                      connection debugging might require -vvvv. This argument may be used
                      multiple times.

See 'ansible-vault <command> --help' for more information on a specific command.
```

# Ansible Vault – шифровать файл

```
$ cat test
```

```
123123123
```

```
$ ansible-vault encrypt test
```

```
New Vault password:
```

```
Confirm New Vault password:
```

```
Encryption successful
```

```
$ cat test
```

```
$ANSIBLE_VAULT;1.1;AES256
```

```
33336663623131353631383162386532623539653230313636616237643633633761383133643638  
6531623836653134393239376162333662393366393266610a316664316438663263323130373835  
64326465636230346133363734356165373262303230316631643761353832386133386363643335  
3334666361346430340a333235363232343562313861633065666131303666303837313136336638  
6437
```

```
$ ansible-vault decrypt test
```

```
Vault password:
```

```
Decryption successful
```

```
$ cat test
```

```
123123123
```



# Ansible Vault – шифровать строку

```
$ ansible-vault encrypt_string
```

```
New Vault password:
```

```
Confirm New Vault password:
```

```
Reading plaintext input from stdin. (ctrl-d to end input, twice if your content does not  
already have a newline)
```

```
SUPERSECRET
```

```
Encryption successful
```

```
!vault |
```

```
$ANSIBLE_VAULT;1.1;AES256
```

```
643036366533316334336564637613432613561373539313535636363343632613766613439
```

```
261626563353934660237643165396266336230383662626364386534373862303465363434
```

```
$ ansible-vault encrypt_string SUPERSECRET
```

```
New Vault password:
```

```
Confirm New Vault password:
```

```
Encryption successful
```

```
!vault |
```

```
$ANSIBLE_VAULT;1.1;AES256
```

```
6465373431353363653432313238626561353834373832393635663235383937356265386562
```

```
373630346136306331313461343232626465336531310a386462333134633062313436343730
```

# Ansible Vault – как пользоваться

Пример инвентаря в YAML без секретов

```
victor@ubuntu:~/lab-ansible-6$ cat inventory.yml
---
webservers:
  hosts:
    local_host:
      ansible_host: 127.0.0.1
    ubuntu2:
      ansible_host: 10.0.2.16

db:
  hosts:
    ubuntu5:
      ansible_host: 10.0.2.5

all_hosts:
  children:
    webservers:
    db:
```

# Ansible Vault – как пользоваться

Пример инвентаря в YAML с секретом

```
$ ansible-playbook playbook.yml -i inventory.yml --ask-vault-pass
```

```
GNU nano 7.2                                inventory.yml
---
webservers:
  hosts:
    local_host:
      ansible_host: 127.0.0.1
      my_local_secret: !vault |
        $ANSIBLE_VAULT;1.1;AES256
        65663964326437633839386361366164383330633638363463396165306639613732663931343263
        3334343831623431653733326430363064386239306363650a303462356433343464333732616238
        39313930623563313537346433366236323236353834356634643539346336343361313530356365
        3733373738343761340a373336346338316332613330656439616662656137316339346531323935
        3136
    ubuntu2:
      ansible_host: 10.0.2.16
```

# Ansible – отработка ошибок

Не считать ошибку этой задачи как повод прекратить список задач хоста:

```
- name: Do not count this as a failure
  ansible.builtin.command: /bin/false
  ignore_errors: true
```

Переопределить, что мы считаем за ошибку в конкретной задаче:

```
- name: Fail task when the command error output prints FAILED
  ansible.builtin.command: /usr/bin/example-command -x -y -z
  register: command_result
  failed_when: "'FAILED' in command_result.stderr"
```

*Аналогично можно переопределить Changed*

# Ansible.builtin.debug - msg

Вывод диагностических сообщений

```
GNU nano 7.2                                playbook_debug.yml
---
- name: Example debug tasks
  hosts: local_host
  become: Yes
  remote_user: runner
  become_method: sudo
  gather_facts: No

  tasks:
  - ansible.builtin.debug: { msg: "HELLO WORLD!" }
```

```
victor@ubuntu:~/lab-ansible-6$ ansible-playbook playbook_debug.yml -i inventory.yml
Vault password:

PLAY [Example debug tasks] *****

TASK [ansible.builtin.debug] *****
ok: [local_host] => {
    "msg": "HELLO WORLD!"
}
```

# Ansible.builtin.debug - vars

Вывод диагностических сообщений

```
$ ansible-playbook playbook_debug.yml -i inventory.yml --ask-vault-pass
```

```
tasks:
- ansible.builtin.debug: { msg: "HELLO WORLD!" }

- ansible.builtin.debug:
  var: my_local_secret
```

```
TASK [ansible.builtin.debug] *****
ok: [local_host] => {
  "my_local_secret": "SUPERSECRET"
}
```

# Ansible.builtin.debug - verbosity

Ограничение на уровень детализации логов (verbosity), при котором выводить отладку

```
$ ansible-playbook playbook_debug.yml -i inventory.yml --ask-vault-pass -vv
```

```
tasks:
- ansible.builtin.debug: { msg: "HELLO WORLD!" }

- ansible.builtin.debug:
  var: my_local_secret
  verbosity: 2
```

```
TASK [ansible.builtin.debug] *****
task path: /home/victor/lab-ansible-6/playbook_debug.yml:12
ok: [localhost] => {
  "my_local_secret": "SUPERSECRET"
}
```



# Ansible - условия «when»

Задача выполняется, если выполнено условие.  
Можем оперировать фактами и переменными.

```
- name: Print the gateway for each host when defined
  ansible.builtin.debug:
    msg: System {{ inventory_hostname }} has gateway {{ ansible_default_ipv4.gateway }}
  when: ansible_default_ipv4.gateway is defined
```

```
- name: List contents of directory
  ansible.builtin.command: ls mydir
  register: contents

- name: Check contents for emptiness
  ansible.builtin.debug:
    msg: "Directory is empty"
  when: contents.stdout == ""
```

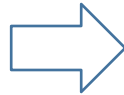


# Ansible – циклы «loop»

Задача выполняется в цикле по элементам списка

```
---
- name: Example debug tasks
  hosts: db
  become: Yes
  remote_user: runner
  become_method: sudo
  gather_facts: No

  tasks:
    - ansible.builtin.debug:
        msg: "{{ item }}"
      loop:
        - Apples
        - Oranjes
        - Raindrops
        - Sunshowers
```



```
victor@ubuntu:~/lab-ansible-6$ ansible-playbook playb
PLAY [Example debug tasks] *****

TASK [ansible.builtin.debug] *****
ok: [ubuntu5] => (item=Apples) => {
    "msg": "Apples"
}
ok: [ubuntu5] => (item=Oranjes) => {
    "msg": "Oranjes"
}
ok: [ubuntu5] => (item=Raindrops) => {
    "msg": "Raindrops"
}
ok: [ubuntu5] => (item=Sunshowers) => {
    "msg": "Sunshowers"
}
```

# Ansible – циклы «loop»

По списку словарей

```
- name: Add several users
  ansible.builtin.user:
    name: "{{ item.name }}"
    state: present
    groups: "{{ item.groups }}"
  loop:
    - { name: 'testuser1', groups: 'wheel' }
    - { name: 'testuser2', groups: 'root' }
```

или по элементам словаря

```
- name: Using dict2items
  ansible.builtin.debug:
    msg: "{{ item.key }} - {{ item.value }}"
  loop: "{{ tag_data | dict2items }}"
  vars:
    tag_data:
      Environment: dev
      Application: payment
```

# Ansible – register (fact)

Теперь несколько хостов >

Задача для каждого хоста>

Задача для каждого хоста>

Задача для каждого хоста>

Зарегистрировать **факт** хоста

Задача для каждого хоста>

Показать **факт** из хоста

```
GNU nano 7.2                                playbook_debug.yml
---
- name: Example debug tasks
  hosts: webserver
  become: Yes
  remote_user: runner
  become_method: sudo
  gather_facts: No

  tasks:
  - ansible.builtin.debug: { msg: "HELLO WORLD!" }

  - ansible.builtin.debug:
    var: my_local_secret

  - name: Get addresses
    ansible.builtin.shell: ip addr
    register: result

  - name: Print return information from the previous task
    ansible.builtin.debug:
    var: result
```

# Ansible – register – Модули обычно что-то RETURN

Ansible Community Documentation

Python 3 Support

Interpreter Discovery

Releases and maintenance

Testing Strategies

Sanity Tests

Frequently Asked Questions

Glossary

Ansible Reference: Module Utilities

Special Variables

Red Hat Ansible Automation Platform

Ansible Automation Hub

Logging Ansible output

ROADMAPS

Ansible Roadmap

ansible-core Roadmaps

Red Hat  
Ansible Automation  
Platform

Key	Description
<code>cmd</code> <code>string</code>	The command executed by the task. <b>Returned:</b> always <b>Sample:</b> <code>"rabbitmqctl join_cluster rabbit@master"</code>
<code>stdout</code> <code>string</code>	The command standard output. <b>Returned:</b> always <b>Sample:</b> <code>"Clustering node rabbit@slave1 with rabbit@master \u0266"</code>
<code>stdout_lines</code> <code>list / elements=string</code>	The command standard output split in lines. <b>Returned:</b> always <b>Sample:</b> <code>["\u0266Clustering node rabbit@slave1 with rabbit@master \u0266"]</code>
<code>msg</code> <code>boolean</code>	<code>changed</code> <b>Returned:</b> always <b>Sample:</b> <code>true</code>
<code>rc</code> <code>integer</code>	The command return code (0 means success). <b>Returned:</b> always <b>Sample:</b> <code>0</code>



# Ansible.builtin.debug – register (fact)

```
victor@ubuntu:~/lab-ansible-6$ ansible-playbook playbook_debug.yml -i inventory.yml --ask-vault-pass
Vault password:
```

```
PLAY [Example debug tasks] *****
```

```
TASK [ansible.builtin.debug] *****
```

```
ok: [local_host] => {
  "msg": "HELLO WORLD!"
}
```

```
ok: [ubuntu2] => {
  "msg": "HELLO WORLD!"
}
```

```
TASK [ansible.builtin.debug] *****
```

```
ok: [ubuntu2] => {
  "my_local_secret": "VARIABLE IS NOT DEFINED!"
}
```

```
ok: [local_host] => {
  "my_local_secret": "SUPERSECRET"
}
```

```
TASK [Get addresses] *****
```

```
changed: [ubuntu2]
changed: [local_host]
```

```
TASK [Print return information from the previous task] *****
```

```
ok: [local_host] => {
  "result": {
    "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "cmd": "ip addr"
```

# Ansible.builtin.debug – register (fact)

```
TASK [Print return information from the previous task] *****
ok: [local_host] => {
  "result": {
    "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python3"
    },
    "changed": true,
    "cmd": "ip addr",
    "delta": "0:00:00.003766",
    "end": "2024-10-01 15:42:52.436713",
    "failed": false,
    "msg": "",
    "rc": 0,
    "start": "2024-10-01 15:42:52.432947",
    "stderr": "",
    "stderr_lines": [],
    "stdout": "1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN
0:00:00:00:00:00 brd 00:00:00:00:00:00\n      inet 127.0.0.1/8 scope host lo\n      vali
t6 ::1/128 scope host noprefixroute \n      valid_lft forever preferred_lft forever\n2
mtu 1500 qdisc pfifo_fast state UP group default qlen 1000\n      link/ether 08:00:27:17:
.15/24 brd 10.0.2.255 scope global enp0s3\n      valid_lft forever preferred_lft forev
ope link \n      valid_lft forever preferred_lft forever",
    "stdout_lines": [
      "1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group",
      "    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00",
      "    inet 127.0.0.1/8 scope host lo",
      "        valid_lft forever preferred_lft forever",
      "    inet6 ::1/128 scope host noprefixroute ",
```

# Ansible Variables

Посмотрим все переменные (факты) хоста

```
ansible-playbook playbook_debug.yml -i inventory.yml --ask-vault-pass -vvvv
```

```
- name: Display all variables/facts known for a host
  ansible.builtin.debug:
    var: hostvars[inventory_hostname]
    verbosity: 4
```

# Ansible Variables

Посмотрим все переменные (факты) хоста

```
ansible-playbook playbook_debug.yml -i inventory.yml --ask-vault-pass -vvvv
```

```
- name: Display all variables/facts known for a host
  ansible.builtin.debug:
    var: hostvars[inventory_hostname]
    verbosity: 4
```

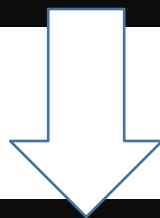
```
TASK [Display all variables/facts known for a host] *****
task path: /home/victor/lab-ansible-6/playbook_debug.yml:28
Trying secret <ansible.parsing.vault.PromptVaultSecret object at 0x762...
ok: [ubuntu2] => {
  "hostvars[inventory_hostname]": {
    "ansible_check_mode": false,
    "ansible_config_file": null,
    "ansible_diff_mode": false,
    "ansible_facts": {
      "discovered_interpreter_python": "/usr/bin/python3"
    },
    "ansible_forks": 5
  }
}
```




# Ansible Variables – подстановка (интерполяция) в синтаксисе jinja2

При использовании обязательно заключать j2 ссылки в кавычки!

```
- ansible.builtin.debug:
  msg: "Band: {{ band }} Release: {{ release }}"
vars:
  band: TOOL
  release: Lateralus
```



```
TASK [ansible.builtin.debug] *****
ok: [ubuntu5] => {
  "msg": "Band: TOOL Release: Lateralus"
}
```



Где же живут  
переменные?

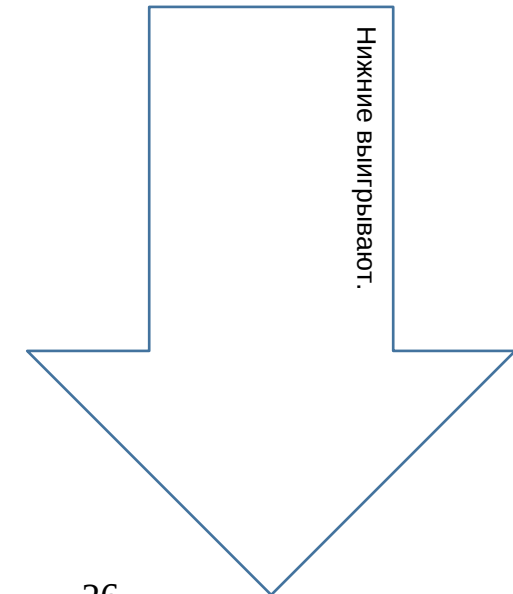
# Ansible Variables - precedence

1. command line values (for example, `-u my_user`, these are not variables)
2. role defaults (as defined in [Role directory structure](#)) <sup>1</sup>
3. inventory file or script group vars <sup>2</sup>
4. inventory group\_vars/all <sup>3</sup>
5. playbook group\_vars/all <sup>3</sup>
6. inventory group\_vars/\* <sup>3</sup>
7. playbook group\_vars/\* <sup>3</sup>
8. inventory file or script host vars <sup>2</sup>
9. inventory host\_vars/\* <sup>3</sup>
10. playbook host\_vars/\* <sup>3</sup>
11. host facts / cached set\_facts <sup>4</sup>
12. play vars
13. play vars\_prompt
14. play vars\_files
15. role vars (as defined in [Role directory structure](#))
16. block vars (only for tasks in block)
17. task vars (only for the task)
18. include\_vars
19. set\_facts / registered vars
20. role (and include\_role) params
21. include params
22. extra vars (for example, `-e "user=my_user"`)(always win precedence)

Где же живут  
переменные?  
Ответ: ВЕЗДЕ )

## Variable precedence: Where should I put a variable?

You can set multiple variables with the same name in many different places. When you do this, Ansible loads every possible variable it finds, and then chooses the variable to apply based on variable precedence. In other words, the different variables will override each other in a certain order.



# Ansible Variables - scopes

- Global: this is set by config, environment variables and the command line
- Play: each play and contained structures, vars entries (vars; vars\_files; vars\_prompt), role defaults and vars.
- Host: variables directly associated to a host, like inventory, include\_vars, facts or registered task outputs

Ансибл постоянно подгружает и дополняет переменные на всех уровнях, при этом выигрывают более поздние переменные (новые затирают старые), но только в пределах своего Scope.

В результате выигрывают:

- Более новые
- С более точным scope
- Более явно объявленные (default проигрывает)

*Фасералт: одновременно могут существовать переменные с одинаковым именем в разных scope*

# Ansible Variables - lifetime

Переменные Ансибла живут:

- Вечно, если они заданы в `inventory` и `group_vars`.
- До окончания `play`, если заданы в ролях или переменных `play`.
- До окончания `task`, если это переменные задачи.
- До окончания плейбуки, если это факт. (именно этим отличаются переменные и факты — факты живут дольше).

Хорошее описание:

<https://habr.com/ru/articles/512036/>

# Ansible Variables – использование значения

Переменные просто «лежат» в виде неинтерполированных строк (включая j2-ссылки), до тех пор, пока Ансиблу не потребуется «использовать значение» конкретной переменной.

Перед использованием будет проведена (рекурсивная) интерполяция j2.

```
---
- hosts: localhost
  tasks:
    - set_fact:
        foo: '{{ foo + 1 }}'
    - debug:
        msg: '{{foo}}'

vars:
  foo: 1
```

Хорошее описание:

<https://habr.com/ru/articles/512036/>

# Ansible - Handlers

После всех задач «flush-атся» хэндлеры:

- Если задача делала notify  
И
- Если задача вернула CHANGED  
ТО
- будет запущен хэндлер с этим именем

```
tasks:
- name: Template configuration file
  ansible.builtin.template:
    src: template.j2
    dest: /etc/foo.conf
  notify:
    - Restart apache
    - Restart memcached

handlers:
- name: Restart memcached
  ansible.builtin.service:
    name: memcached
    state: restarted

- name: Restart apache
  ansible.builtin.service:
    name: apache
    state: restarted
```

# П06. Деплой веб-сервера с помощью Ansible

Сделать ansible-плейбук, который на целевых хостах:

- устанавливает сканер NMAP
- копирует файл со списком целей для сканирования
- запускает NMAP и сканирует цели
- Собирает результаты сканирования (stdout)
- выводит результаты сканирования

Подготовка аналогична  
предыдущей лабе

## Заготовка списка целей сканирования

```
GNU nano 7.2 targets.txt
student.psu.ru
etis.psu.ru
www.psu.ru
```



## Установка NMAP

```
- name: Install NMAP
  apt:
    name: nmap
    state: latest
    update_cache: True
```

## Копирование списка целей

```
- name: Copy nmap targets list
  copy:
    src: ./targets.txt
    dest: ~/targets.txt
    force: yes
    mode: 0644
```

Запуска сканирования  
с выводом результатов в stdout в greppable-формате

```
- name: Run nmap  
  shell: "nmap -iL ~/targets.txt -p 80 -oG -"  
  register: nmap_result
```

## Вывод результатов

```
- name: Nmap output  
  debug:  
    msg: "{{ nmap_result.stdout_lines }}"
```

Пример результата последней задачи:

```
TASK [Nmap output] *****
ok: [10.0.2.16] => {
  "msg": [
    "# Nmap 7.94SVN scan initiated Tue Oct  1 20:06:57 2024 as: nmap -iL /root/tar
oG -",
    "Host: 212.192.83.80 (helios.psu.ru)\tStatus: Up",
    "Host: 212.192.83.80 (helios.psu.ru)\tPorts: 80/open/tcp//http//",
    "Host: 212.192.80.42 (etis.psu.ru)\tStatus: Up",
    "Host: 212.192.80.42 (etis.psu.ru)\tPorts: 80/filtered/tcp//http//",
    "Host: 212.192.64.44 (www.psu.ru)\tStatus: Up",
    "Host: 212.192.64.44 (www.psu.ru)\tPorts: 80/open/tcp//http//",
    "# Nmap done at Tue Oct  1 20:06:58 2024 -- 3 IP addresses (3 hosts up) scanned
s"
  ]
}
ok: [10.0.2.15] => {
  "msg": [
    "# Nmap 7.94SVN scan initiated Tue Oct  1 20:06:57 2024 as: nmap -iL /root/tar
oG -",
    "Host: 212.192.83.80 (helios.psu.ru)\tStatus: Up",
    "Host: 212.192.83.80 (helios.psu.ru)\tPorts: 80/open/tcp//http//",
    "Host: 212.192.80.42 (etis.psu.ru)\tStatus: Up"
```

```
victor@ubuntu:~/lab-ansible$ ansible-playbook playbook.yml -i inventory.ini

PLAY [Deploy nginx] *****
***

TASK [Gathering Facts] *****
***
ok: [10.0.2.16]

TASK [Install nginx] *****
***
ok: [10.0.2.16]

TASK [Copy index] *****
***
changed: [10.0.2.16]

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