Controlling playbook execution: strategies and more

By default, Ansible runs each task on all hosts affected by a play before starting the next task on any host, using 5 forks. If you want to change this default behavior, you can use a different strategy plugin, change the number of forks, or apply one of several keywords like `serial`.

Selecting a strategy

The default behavior described above is the :ref:`linear strategy<linear_strategy>`. Ansible offers other strategies, including the :ref:`debug strategy<debug_strategy>` (see also :ref:`playbook_debugger`) and the :ref:`free strategy<free_strategy>`, which allows each host to run until the end of the play as fast as it can:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 13); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 13); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 13); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 13); *backlink*

Unknown interpreted text role "ref".

```
- hosts: all
  strategy: free
  tasks:
  # ...
```

You can select a different strategy for each play as shown above, or set your preferred strategy globally in `ansible.cfg`, under the `defaults` stanza:

```
[defaults]
strategy = free
```

All strategies are implemented as :ref:`strategy plugins<strategy_plugins>`. Please review the documentation for each strategy plugin for details on how it works.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 29); *backlink*

Unknown interpreted text role "ref".

Setting the number of forks

If you have the processing power available and want to use more forks, you can set the number in `ansible.cfg`:

```
[defaults]
forks = 30
```

or pass it on the command line: *ansible-playbook -f 30 my_playbook.yml*.

Using keywords to control execution

In addition to strategies, several :ref:`keywords<playbook_keywords>` also affect play execution. You can set a number, a percentage, or a list of numbers of hosts you want to manage at a time with `serial`. Ansible completes the play on the specified number or percentage of hosts before starting the next batch of hosts. You can restrict the number of workers allotted to a block or task with `throttle`. You can control how Ansible selects the next host in a group to execute against with `order`. You can run a task on a single host with `run_once`. These keywords are not strategies. They are directives or options applied to a play, block, or task.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 45); *backlink*

Unknown interpreted text role "ref".

Other keywords that affect play execution include `ignore_errors`, `ignore_unreachable`, and `any_errors_fatal`. These options are documented in :ref:`playbooks_error_handling`.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 47); *backlink*

Unknown interpreted text role "ref".

Setting the batch size with `serial`

By default, Ansible runs in parallel against all the hosts in the :ref:`pattern <intro_patterns>` you set in the `hosts:` field of each play. If you want to manage only a few machines at a time, for example during a rolling update, you can define how many hosts Ansible should manage at a single time using the `serial` keyword:

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 54); *backlink*

Unknown interpreted text role "ref".

```
---
- name: test play
  hosts: webservers
  serial: 3
  gather_facts: False

  tasks:
    - name: first task
      command: hostname
    - name: second task
      command: hostname
```

In the above example, if we had 6 hosts in the group 'webservers', Ansible would execute the play completely (both tasks) on 3 of the hosts before moving on to the next 3 hosts:

System Message: WARNING/2 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 73)

Cannot analyze code. No Pygments lexer found for "ansible-output".

```
.. code-block:: ansible-output

    PLAY [webservers] ****************************************

    TASK [first task] ****************************************
    changed: [web3]
    changed: [web2]
    changed: [web1]
```

```
TASK [second task] ***************************************
changed: [web1]
changed: [web2]
changed: [web3]

PLAY [webservers] ****************************************

TASK [first task] ****************************************
changed: [web4]
changed: [web5]
changed: [web6]

TASK [second task] ***************************************
changed: [web4]
changed: [web5]
changed: [web6]

PLAY RECAP ***********************************************
web1       : ok=2    changed=2    unreachable=0    failed=0
web2       : ok=2    changed=2    unreachable=0    failed=0
web3       : ok=2    changed=2    unreachable=0    failed=0
web4       : ok=2    changed=2    unreachable=0    failed=0
web5       : ok=2    changed=2    unreachable=0    failed=0
web6       : ok=2    changed=2    unreachable=0    failed=0
```

Note

Setting the batch size with `serial` changes the scope of the Ansible failures to the batch size, not the entire host list. You can use :ref:`ignore_unreachable <ignore_unreachable>` or :ref:`max_fail_percentage <maximum_failure_percentage>` to modify this behavior.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 109); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 109); *backlink*

Unknown interpreted text role "ref".

You can also specify a percentage with the `serial` keyword. Ansible applies the percentage to the total number of hosts in a play to determine the number of hosts per pass:

```
---
- name: test play
  hosts: webservers
  serial: "30%"
```

If the number of hosts does not divide equally into the number of passes, the final pass contains the remainder. In this example, if you had 20 hosts in the webservers group, the first batch would contain 6 hosts, the second batch would contain 6 hosts, the third batch would contain 6 hosts, and the last batch would contain 2 hosts.

You can also specify batch sizes as a list. For example:

```
---
- name: test play
  hosts: webservers
  serial:
    - 1
    - 5
    - 10
```

In the above example, the first batch would contain a single host, the next would contain 5 hosts, and (if there are any hosts left), every following batch would contain either 10 hosts or all the remaining hosts, if fewer than 10 hosts remained.

You can list multiple batch sizes as percentages:

```
---
```

```
- name: test play
  hosts: webservers
  serial:
    - "10%"
    - "20%"
    - "100%"
```

You can also mix and match the values:

```
---
- name: test play
  hosts: webservers
  serial:
    - 1
    - 5
    - "20%"
```

> **Note**
>
> No matter how small the percentage, the number of hosts per pass will always be 1 or greater.

Restricting execution with `throttle`

The `throttle` keyword limits the number of workers for a particular task. It can be set at the block and task level. Use `throttle` to restrict tasks that may be CPU-intensive or interact with a rate-limiting API:

```
tasks:
- command: /path/to/cpu_intensive_command
  throttle: 1
```

If you have already restricted the number of forks or the number of machines to execute against in parallel, you can reduce the number of workers with `throttle`, but you cannot increase it. In other words, to have an effect, your `throttle` setting must be lower than your `forks` or `serial` setting if you are using them together.

Ordering execution based on inventory

The `order` keyword controls the order in which hosts are run. Possible values for order are:

inventory:
: (default) The order provided by the inventory for the selection requested (see note below)

reverse_inventory:
: The same as above, but reversing the returned list

sorted:
: Sorted alphabetically sorted by name

reverse_sorted:
: Sorted by name in reverse alphabetical order

shuffle:
: Randomly ordered on each run

> **Note**
>
> the 'inventory' order does not equate to the order in which hosts/groups are defined in the inventory source file, but the 'order in which a selection is returned from the compiled inventory'. This is a backwards compatible option and while reproducible it is not normally predictable. Due to the nature of inventory, host patterns, limits, inventory plugins and the ability to allow multiple sources it is almost impossible to return such an order. For simple cases this might happen to match the file definition order, but that is not guaranteed.

Running on a single machine with `run_once`

If you want a task to run only on the first host in your batch of hosts, set `run_once` to true on that task:

```
---
# ...

  tasks:

    # ...

    - command: /opt/application/upgrade_db.py
      run_once: true

    # ...
```

Ansible executes this task on the first host in the current batch and applies all results and facts to all the hosts in the same batch. This approach is similar to applying a conditional to a task such as:

```
- command: /opt/application/upgrade_db.py
  when: inventory_hostname == webservers[0]
```

However, with `run_once`, the results are applied to all the hosts. To run the task on a specific host, instead of the first host in the batch, delegate the task:

```
- command: /opt/application/upgrade_db.py
  run_once: true
  delegate_to: web01.example.org
```

As always with :ref:`delegation <playbooks_delegation>`, the action will be executed on the delegated host, but the information is still that of the original host in the task.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 231); *backlink*

Unknown interpreted text role "ref".

Note

When used together with `serial`, tasks marked as `run_once` will be run on one host in *each* serial batch. If the task must run only once regardless of `serial` mode, use `when: inventory_hostname == ansible_play_hosts_all[0]` construct.

Note

Any conditional (in other words, *when:*) will use the variables of the 'first host' to decide if the task runs or not, no other hosts will be tested.

Note

If you want to avoid the default behavior of setting the fact for all hosts, set `delegate_facts: True` for the specific task or block.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\user_guide\[ansible-devel][docs][docsite][rst][user_guide]playbooks_strategies.rst`, line 243)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`about_playbooks`
       An introduction to playbooks
   :ref:`playbooks_delegation`
       Running tasks on or assigning facts to specific machines
   :ref:`playbooks_reuse_roles`
       Playbook organization by roles
   `User Mailing List <https://groups.google.com/group/ansible-devel>`_
       Have a question?  Stop by the google group!
   :ref:`communication_irc`
       How to join Ansible chat channels
```