**Controlling playbook execution: strategies and more**[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#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](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#selecting-a-strategy)
* [Setting the number of forks](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#setting-the-number-of-forks)
* [Using keywords to control execution](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#using-keywords-to-control-execution)
  + [Setting the batch size with serial](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#setting-the-batch-size-with-serial)
  + [Restricting execution with throttle](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#restricting-execution-with-throttle)
  + [Ordering execution based on inventory](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#ordering-execution-based-on-inventory)
  + [Running on a single machine with run\_once](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#running-on-a-single-machine-with-run-once)

[**Selecting a strategy**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#id1)[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#selecting-a-strategy)

The default behavior described above is the [linear strategy](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/linear_strategy.html#linear-strategy). Ansible offers other strategies, including the [debug strategy](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/debug_strategy.html#debug-strategy) (see also [Debugging tasks](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_debugger.html#playbook-debugger)) and the [free strategy](https://docs.ansible.com/ansible/latest/collections/ansible/builtin/free_strategy.html#free-strategy), which allows each host to run until the end of the play as fast as it can:

**-** **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 [strategy plugins](https://docs.ansible.com/ansible/latest/plugins/strategy.html#strategy-plugins). Please review the documentation for each strategy plugin for details on how it works.

[**Setting the number of forks**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#id2)[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#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**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#id3)[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#using-keywords-to-control-execution)

In addition to strategies, several [keywords](https://docs.ansible.com/ansible/latest/reference_appendices/playbooks_keywords.html#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.

Other keywords that affect play execution include ignore\_errors, ignore\_unreachable, and any\_errors\_fatal. These options are documented in [Error handling in playbooks](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_error_handling.html#playbooks-error-handling).

[**Setting the batch size with serial**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#id4)[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#setting-the-batch-size-with-serial)

By default, Ansible runs in parallel against all the hosts in the [pattern](https://docs.ansible.com/ansible/latest/inventory_guide/intro_patterns.html#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:

**---**

**-** **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:

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

**TASK** **[**first task**]** **\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**changed:** **[web1]**

**changed:** **[web3]**

**changed:** **[web2]**

**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 **skipped=**0 **rescued=**0 **ignored=**0

web2 **:** **ok=**2 **changed=**2 **unreachable=**0 **failed=**0 **skipped=**0 **rescued=**0 **ignored=**0

web3 **:** **ok=**2 **changed=**2 **unreachable=**0 **failed=**0 **skipped=**0 **rescued=**0 **ignored=**0

web4 **:** **ok=**2 **changed=**2 **unreachable=**0 **failed=**0 **skipped=**0 **rescued=**0 **ignored=**0

web5 **:** **ok=**2 **changed=**2 **unreachable=**0 **failed=**0 **skipped=**0 **rescued=**0 **ignored=**0

web6 **:** **ok=**2 **changed=**2 **unreachable=**0 **failed=**0 **skipped=**0 **rescued=**0 **ignored=**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 [ignore\_unreachable](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_error_handling.html" \l "ignore-unreachable) or [max\_fail\_percentage](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_error_handling.html" \l "maximum-failure-percentage) to modify this behavior.

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**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#id5)[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#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**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#id6)[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#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**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#id7)[**ℑ**](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_strategies.html#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 [delegation](https://docs.ansible.com/ansible/latest/playbook_guide/playbooks_delegation.html#playbooks-delegation), the action will be executed on the delegated host, but the information is still that of the original host in the task.

**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.