Asynchronous actions and polling

By default Ansible runs tasks synchronously, holding the connection to the remote node open until the action is completed. This means within a playbook, each task blocks the next task by default, meaning subsequent tasks will not run until the current task completes. This behavior can create challenges. For example, a task may take longer to complete than the SSH session allows for, causing a timeout. Or you may want a long-running process to execute in the background while you perform other tasks concurrently. Asynchronous mode lets you control how long-running tasks execute.

- Asynchronous ad hoc tasks
- Asynchronous playbook tasks
  - Avoid connection timeouts: poll > 0
  - Run tasks concurrently: poll = 0

Asynchronous ad hoc tasks

You can execute long-running operations in the background with :ref:`ad hoc tasks <intro_adhoc>`. For example, to execute `long_running_operation` asynchronously in the background, with a timeout (`-B`) of 3600 seconds, and without polling (`-P`):

> **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_async.rst, line 14); *backlink***
>
> Unknown interpreted text role "ref".

```
$ ansible all -B 3600 -P 0 -a "/usr/bin/long_running_operation --do-stuff"
```

To check on the job status later, use the `async_status` module, passing it the job ID that was returned when you ran the original job in the background:

```
$ ansible web1.example.com -m async_status -a "jid=488359678239.2844"
```

Ansible can also check on the status of your long-running job automatically with polling. In most cases, Ansible will keep the connection to your remote node open between polls. To run for 30 minutes and poll for status every 60 seconds:

```
$ ansible all -B 1800 -P 60 -a "/usr/bin/long_running_operation --do-stuff"
```

Poll mode is smart so all jobs will be started before polling begins on any machine. Be sure to use a high enough `--forks` value if you want to get all of your jobs started very quickly. After the time limit (in seconds) runs out (`-B`), the process on the remote nodes will be terminated.

Asynchronous mode is best suited to long-running shell commands or software upgrades. Running the copy module asynchronously, for example, does not do a background file transfer.

Asynchronous playbook tasks

:ref:`Playbooks <working_with_playbooks>` also support asynchronous mode and polling, with a simplified syntax. You can use asynchronous mode in playbooks to avoid connection timeouts or to avoid blocking subsequent tasks. The behavior of asynchronous mode in a playbook depends on the value of *poll*.

> **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_async.rst, line 39); *backlink***
>
> Unknown interpreted text role "ref".

Avoid connection timeouts: poll > 0

If you want to set a longer timeout limit for a certain task in your playbook, use `async` with `poll` set to a positive value. Ansible will still block the next task in your playbook, waiting until the async task either completes, fails or times out. However, the task will only time out if it exceeds the timeout limit you set with the `async` parameter.

To avoid timeouts on a task, specify its maximum runtime and how frequently you would like to poll for status:

```
---

- hosts: all
  remote_user: root
```

```
  tasks:

  - name: Simulate long running op (15 sec), wait for up to 45 sec, poll every 5 sec
    ansible.builtin.command: /bin/sleep 15
    async: 45
    poll: 5
```

Note

The default poll value is set by the :ref:`DEFAULT_POLL_INTERVAL` setting. There is no default for the async time limit. If you leave off the 'async' keyword, the task runs synchronously, which is Ansible's default.

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_async.rst, line 63); *backlink*

Unknown interpreted text role "ref".

Note

As of Ansible 2.3, async does not support check mode and will fail the task when run in check mode. See :ref:`check_mode_dry` on how to skip a task in check mode.

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_async.rst, line 69); *backlink*

Unknown interpreted text role "ref".

Note

When an async task completes with polling enabled, the temporary async job cache file (by default in ~/.ansible_async/) is automatically removed.

Run tasks concurrently: poll = 0

If you want to run multiple tasks in a playbook concurrently, use `async` with `poll` set to 0. When you set `poll: 0`, Ansible starts the task and immediately moves on to the next task without waiting for a result. Each async task runs until it either completes, fails or times out (runs longer than its `async` value). The playbook run ends without checking back on async tasks.

To run a playbook task asynchronously:

```
---

- hosts: all
  remote_user: root

  tasks:

  - name: Simulate long running op, allow to run for 45 sec, fire and forget
    ansible.builtin.command: /bin/sleep 15
    async: 45
    poll: 0
```

Note

Do not specify a poll value of 0 with operations that require exclusive locks (such as yum transactions) if you expect to run other commands later in the playbook against those same resources.

Note

Using a higher value for `--forks` will result in kicking off asynchronous tasks even faster. This also increases the efficiency of polling.

Note

When running with `poll: 0`, Ansible will not automatically cleanup the async job cache file. You will need to manually clean this up with the :ref:`async_status <async_status_module>` module with `mode: cleanup`.

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_async.rst, line 105); *backlink*

Unknown interpreted text role "ref".

If you need a synchronization point with an async task, you can register it to obtain its job ID and use the :ref:`async_status <async_status_module>` module to observe it in a later task. For example:

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_async.rst, line 109); *backlink*

Unknown interpreted text role "ref".

```
- name: Run an async task
  ansible.builtin.yum:
    name: docker-io
    state: present
  async: 1000
  poll: 0
  register: yum_sleeper

- name: Check on an async task
  async_status:
    jid: "{{ yum_sleeper.ansible_job_id }}"
  register: job_result
  until: job_result.finished
  retries: 100
  delay: 10
```

Note

If the value of `async:` is not high enough, this will cause the "check on it later" task to fail because the temporary status file that the `async_status:` is looking for will not have been written or no longer exist

Note

Asynchronous playbook tasks always return changed. If the task is using a module that requires the user to annotate changes with `changed_when`, `creates`, and so on, then those should be added to the following `async_status` task.

To run multiple asynchronous tasks while limiting the number of tasks running concurrently:

```
#####################
# main.yml
#####################
- name: Run items asynchronously in batch of two items
  vars:
    sleep_durations:
      - 1
      - 2
      - 3
      - 4
      - 5
    durations: "{{ item }}"
  include_tasks: execute_batch.yml
  loop: "{{ sleep_durations | batch(2) | list }}"

#####################
# execute_batch.yml
#####################
- name: Async sleeping for batched_items
  ansible.builtin.command: sleep {{ async_item }}
  async: 45
  poll: 0
  loop: "{{ durations }}"
  loop_control:
    loop_var: "async_item"
  register: async_results
```

```
- name: Check sync status
  async_status:
    jid: "{{ async_result_item.ansible_job_id }}"
  loop: "{{ async_results.results }}"
  loop_control:
    loop_var: "async_result_item"
  register: async_poll_results
  until: async_poll_results.finished
  retries: 30
```

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_async.rst, line 180)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`playbooks_strategies`
       Options for controlling playbook execution
   :ref:`playbooks_intro`
       An introduction to playbooks
   `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
```