Blocks

Blocks create logical groups of tasks. Blocks also offer ways to handle task errors, similar to exception handling in many programming languages.

- Grouping tasks with blocks
- Handling errors with blocks

Grouping tasks with blocks

All tasks in a block inherit directives applied at the block level. Most of what you can apply to a single task (with the exception of loops) can be applied at the block level, so blocks make it much easier to set data or directives common to the tasks. The directive does not affect the block itself, it is only inherited by the tasks enclosed by a block. For example, a *when* statement is applied to the tasks within a block, not to the block itself.

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_blocks.rst`, line 17)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: YAML
 :emphasize-lines: 3
 :caption: Block example with named tasks inside the block

  tasks:
    - name: Install, configure, and start Apache
      block:
        - name: Install httpd and memcached
          ansible.builtin.yum:
            name:
            - httpd
            - memcached
            state: present

        - name: Apply the foo config template
          ansible.builtin.template:
            src: templates/src.j2
            dest: /etc/foo.conf

        - name: Start service bar and enable it
          ansible.builtin.service:
            name: bar
            state: started
            enabled: True
      when: ansible_facts['distribution'] == 'CentOS'
      become: true
      become_user: root
      ignore_errors: yes
```

In the example above, the 'when' condition will be evaluated before Ansible runs each of the three tasks in the block. All three tasks also inherit the privilege escalation directives, running as the root user. Finally, `ignore_errors: yes` ensures that Ansible continues to execute the playbook even if some of the tasks fail.

Names for blocks have been available since Ansible 2.3. We recommend using names in all tasks, within blocks or elsewhere, for better visibility into the tasks being executed when you run the playbook.

Handling errors with blocks

You can control how Ansible responds to task errors using blocks with `rescue` and `always` sections.

Rescue blocks specify tasks to run when an earlier task in a block fails. This approach is similar to exception handling in many programming languages. Ansible only runs rescue blocks after a task returns a 'failed' state. Bad task definitions and unreachable hosts will not trigger the rescue 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_blocks.rst`, line 60)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: YAML
 :emphasize-lines: 3,14
 :caption: Block error handling example

  tasks:
  - name: Handle the error
    block:
      - name: Print a message
        ansible.builtin.debug:
          msg: 'I execute normally'

      - name: Force a failure
        ansible.builtin.command: /bin/false

      - name: Never print this
        ansible.builtin.debug:
          msg: 'I never execute, due to the above task failing, :-('
    rescue:
      - name: Print when errors
        ansible.builtin.debug:
          msg: 'I caught an error, can do stuff here to fix it, :-)'
```

You can also add an `always` section to a block. Tasks in the `always` section run no matter what the task status of the previous block is.

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_blocks.rst`, line 85)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: YAML
 :emphasize-lines: 2,13
 :caption: Block with always section

  - name: Always do X
    block:
      - name: Print a message
        ansible.builtin.debug:
          msg: 'I execute normally'

      - name: Force a failure
        ansible.builtin.command: /bin/false

      - name: Never print this
        ansible.builtin.debug:
          msg: 'I never execute :-('
    always:
      - name: Always do this
        ansible.builtin.debug:
          msg: "This always executes, :-)"
```

Together, these elements offer complex 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_blocks.rst`, line 108)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: YAML
 :emphasize-lines: 2,13,24
 :caption: Block with all sections

 - name: Attempt and graceful roll back demo
   block:
     - name: Print a message
       ansible.builtin.debug:
         msg: 'I execute normally'

     - name: Force a failure
       ansible.builtin.command: /bin/false

     - name: Never print this
       ansible.builtin.debug:
         msg: 'I never execute, due to the above task failing, :-('
   rescue:
     - name: Print when errors
       ansible.builtin.debug:
         msg: 'I caught an error'

     - name: Force a failure in middle of recovery! >:-)
       ansible.builtin.command: /bin/false

     - name: Never print this
       ansible.builtin.debug:
         msg: 'I also never execute :-('
   always:
     - name: Always do this
       ansible.builtin.debug:
         msg: "This always executes"
```

The tasks in the `block` execute normally. If any tasks in the block return `failed`, the `rescue` section executes tasks to recover from the error. The `always` section runs regardless of the results of the `block` and `rescue` sections.

If an error occurs in the block and the rescue task succeeds, Ansible reverts the failed status of the original task for the run and continues to run the play as if the original task had succeeded. The rescued task is considered successful, and does not trigger `max_fail_percentage` or `any_errors_fatal` configurations. However, Ansible still reports a failure in the playbook statistics.

You can use blocks with `flush_handlers` in a rescue task to ensure that all handlers run even if an error occurs:

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_blocks.rst`, line 146)

Error in "code-block" directive: unknown option: "emphasize-lines".

```
.. code-block:: YAML
 :emphasize-lines: 3,12
 :caption: Block run handlers in error handling

  tasks:
    - name: Attempt and graceful roll back demo
      block:
        - name: Print a message
          ansible.builtin.debug:
            msg: 'I execute normally'
          changed_when: yes
          notify: run me even after an error

        - name: Force a failure
          ansible.builtin.command: /bin/false
      rescue:
        - name: Make sure all handlers run
          meta: flush_handlers
  handlers:
     - name: Run me even after an error
       ansible.builtin.debug:
         msg: 'This handler runs even on error'
```

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_blocks.rst`, line 170)

Unknown directive type "versionadded".

```
.. versionadded:: 2.1
```

Ansible provides a couple of variables for tasks in the `rescue` portion of a block:

ansible_failed_task
: The task that returned 'failed' and triggered the rescue. For example, to get the name use `ansible_failed_task.name`.

ansible_failed_result
: The captured return result of the failed task that triggered the rescue. This would equate to having used this var in the `register` 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_blocks.rst, line 180)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`playbooks_intro`
       An introduction to playbooks
   :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
```