Controlling where tasks run: delegation and local actions

By default Ansible gathers facts and executes all tasks on the machines that match the `hosts` line of your playbook. This page shows you how to delegate tasks to a different machine or group, delegate facts to specific machines or groups, or run an entire playbook locally. Using these approaches, you can manage inter-related environments precisely and efficiently. For example, when updating your webservers, you might need to remove them from a load-balanced pool temporarily. You cannot perform this task on the webservers themselves. By delegating the task to localhost, you keep all the tasks within the same play.

Tasks that cannot be delegated

Some tasks always execute on the controller. These tasks, including `include`, `add_host`, and `debug`, cannot be delegated.

Delegating tasks

If you want to perform a task on one host with reference to other hosts, use the `delegate_to` keyword on a task. This is ideal for managing nodes in a load balanced pool or for controlling outage windows. You can use delegation with the :ref:`serial <rolling_update_batch_size>` keyword to control the number of hosts executing at one time:

> **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_delegation.rst`, line 21); *backlink*
>
> Unknown interpreted text role "ref".

```
---
- hosts: webservers
  serial: 5

  tasks:
    - name: Take out of load balancer pool
      ansible.builtin.command: /usr/bin/take_out_of_pool {{ inventory_hostname }}
      delegate_to: 127.0.0.1

    - name: Actual steps would go here
      ansible.builtin.yum:
        name: acme-web-stack
        state: latest

    - name: Add back to load balancer pool
      ansible.builtin.command: /usr/bin/add_back_to_pool {{ inventory_hostname }}
      delegate_to: 127.0.0.1
```

The first and third tasks in this play run on 127.0.0.1, which is the machine running Ansible. There is also a shorthand syntax that you can use on a per-task basis: `local_action`. Here is the same playbook as above, but using the shorthand syntax for delegating to 127.0.0.1:

```
---
# ...

  tasks:
    - name: Take out of load balancer pool
      local_action: ansible.builtin.command /usr/bin/take_out_of_pool {{ inventory_hostname }}

# ...

    - name: Add back to load balancer pool
      local_action: ansible.builtin.command /usr/bin/add_back_to_pool {{ inventory_hostname }}
```

You can use a local action to call 'rsync' to recursively copy files to the managed servers:

```
---
# ...

  tasks:
    - name: Recursively copy files from management server to target
      local_action: ansible.builtin.command rsync -a /path/to/files {{ inventory_hostname }}:/path/to/targ
```

Note that you must have passphrase-less SSH keys or an ssh-agent configured for this to work, otherwise rsync asks for a passphrase.

To specify more arguments, use the following syntax:

```
---
# ...

  tasks:
    - name: Send summary mail
      local_action:
        module: community.general.mail
        subject: "Summary Mail"
        to: "{{ mail recipient }}"
        body: "{{ mail_body }}"
      run_once: True
```

Note

- The *ansible_host* variable and other connection variables, if present, reflects information about the host a task is delegated to, not the inventory_hostname.

Warning

Although you can `delegate_to` a host that does not exist in inventory (by adding IP address, DNS name or whatever requirement the connection plugin has), doing so does not add the host to your inventory and might cause issues. Hosts delegated to in this way do not inherit variables from the "all" group', so variables like connection user and key are missing. If you must `delegate_to` a non-inventory host, use the :ref:`add host module <add_host_module>`.

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_delegation.rst, line 93); *backlink*

Unknown interpreted text role "ref".

Delegation and parallel execution

By default Ansible tasks are executed in parallel. Delegating a task does not change this and does not handle concurrency issues (multiple forks writing to the same file). Most commonly, users are affected by this when updating a single file on a single delegated to host for all hosts (using the `copy`, `template`, or `lineinfile` modules, for example). They will still operate in parallel forks (default 5) and overwrite each other.

This can be handled in several ways:

```
- name: "handle concurrency with a loop on the hosts with `run_once: true`"
  lineinfile: "<options here>"
  run_once: true
  loop: '{{ ansible_play_hosts_all }}'
```

By using an intermediate play with *serial: 1* or using *throttle: 1* at task level, for more detail see :ref:`playbooks_strategies`

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_delegation.rst, line 112); *backlink*

Unknown interpreted text role "ref".

Delegating facts

Delegating Ansible tasks is like delegating tasks in the real world - your groceries belong to you, even if someone else delivers them to your home. Similarly, any facts gathered by a delegated task are assigned by default to the *inventory_hostname* (the current host), not to the host which produced the facts (the delegated to host). To assign gathered facts to the delegated host instead of the current host, set `delegate_facts` to `true`:

```
---
- hosts: app_servers

  tasks:
    - name: Gather facts from db servers
      ansible.builtin.setup:
      delegate_to: "{{ item }}"
      delegate_facts: true
      loop: "{{ groups['dbservers'] }}"
```

This task gathers facts for the machines in the dbservers group and assigns the facts to those machines, even though the play targets the app_servers group. This way you can lookup *hostvars['dbhost1']['ansible_default_ipv4']['address']* even though dbservers

were not part of the play, or left out by using *--limit*.

Local playbooks

It may be useful to use a playbook locally on a remote host, rather than by connecting over SSH. This can be useful for assuring the configuration of a system by putting a playbook in a crontab. This may also be used to run a playbook inside an OS installer, such as an Anaconda kickstart.

To run an entire playbook locally, just set the `hosts:` line to `hosts: 127.0.0.1` and then run the playbook like so:

```
ansible-playbook playbook.yml --connection=local
```

Alternatively, a local connection can be used in a single playbook play, even if other plays in the playbook use the default remote connection type:

```
---
- hosts: 127.0.0.1
  connection: local
```

Note

If you set the connection to local and there is no ansible_python_interpreter set, modules will run under /usr/bin/python and not under {{ ansible_playbook_python }}. Be sure to set ansible_python_interpreter: "{{ ansible_playbook_python }}" in host_vars/localhost.yml, for example. You can avoid this issue by using `local_action` or `delegate_to: localhost` instead.

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_delegation.rst, line 163)

Unknown directive type "seealso".

```
.. seealso::

   :ref:`playbooks_intro`
       An introduction to playbooks
   :ref:`playbooks_strategies`
       More ways to control how and where Ansible executes
   `Ansible Examples on GitHub <https://github.com/ansible/ansible-examples>`_
       Many examples of full-stack deployments
   `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
```