Sometimes you want a task to run only when a change is made on a machine. For example, you may want to restart a service if a task updates the configuration of that service, but not if the configuration is unchanged. Ansible uses handlers to address this use case. Handlers are tasks that only run when notified. Each handler should have a globally unique name.

* Handlers always run in the order they are defined, not in the order listed in the notify-statement. This is also the case for handlers using listen.
* Handler names and listen topics live in a global namespace.
* Handler names are templatable and listen topics are not.
* Use unique handler names. If you trigger more than one handler with the same name, the first one(s) get overwritten. Only the last one defined will run.
* You can notify a handler defined inside a static include.
* You cannot notify a handler defined inside a dynamic include.

**Handlers example**

We will write a playbook ansible-handlers.yml to install httpd package and then use a handler to start the httpd service. This handler will be executed only when change is made i.e. the dependent task reports changed=1 or higher value.

---

- name: Handlers Example

hosts: server1

gather\_facts: false

tasks:

- name: Install httpd latest version

yum:

name: httpd

state: latest

become: true

notify: restart\_httpd

handlers:

- name: restart\_httpd

become: true

service:

name: httpd

state: started

**NOTE:**

You have to take care of the indentation for handlers as it should start from the same line as your tasks.

In this playbook I have created a task which will install httpd rpm to the latest available version using sudo privilege. Next I have created a handler which will take care of starting the service. The handler will be executed using the value of "name" of the handler i.e. "restart\_httpd" using the notify keyword.

I have used notify with the name of our handler so this will check the changed mode of the task and accordingly will decide to execute the handler.

Let us execute the playbook:

[ansible@controller ~]$ ansible-playbook ansible-handlers.yml

PLAY [Handlers Example] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Install httpd latest version] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1]

RUNNING HANDLER [restart\_httpd] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server1 : ok=2 changed=2 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

From the output we know that our TASK and HANDLER, both have executed successfully.

If we re-run the same playbook to check if handler is executed:

[ansible@controller ~]$ ansible-playbook ansible-handlers.yml

PLAY [Handlers Example] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Install httpd latest version] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server1]

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server1 : ok=1 changed=0 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

This time there was no change performed, because httpd was already in installed state.

**Flush handlers and control when handler runs**

By default, handlers run after all the tasks in a particular play have been completed. This approach is efficient, because the handler only runs once, regardless of how many tasks notify it. For example, if multiple tasks update a configuration file and notify a handler to restart Apache, Ansible only bounces Apache once to avoid unnecessary restarts.

In this sample playbook I have 2 tasks which will print some text on the console using debug module. I have also added one handler which will provide the date and time of execution. Now I am calling this handler from both the tasks:

---

- name: Handlers Example

hosts: server1

gather\_facts: false

become: true

tasks:

- name: print message-1

debug:

msg: "First Message"

changed\_when: true

notify: run\_handler

- name: print message-2

debug:

msg: "Second Message"

changed\_when: true

notify: run\_handler

handlers:

- name: run\_handler

debug:

msg: "Today's date and time: {{ '%d-%m-%Y %H:%M:%S' | strftime }}"

Let's verify the output if the handler is executed for both the tasks:

[ansible@controller ~]$ ansible-playbook ansible-handlers.yml

PLAY [Handlers Example] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [print message-1] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1] => {

"msg": "First Message"

}

TASK [print message-2] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1] => {

"msg": "Second Message"

}

RUNNING HANDLER [run\_handler] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server1] => {

"msg": "Today's date and time: 25-09-2020 09:55:03"

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server1 : ok=3 changed=2 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

So the handler was called only once irrespective of the fact that both the tasks were calling the handler.

Now this is the default behaviour but **if you need handlers to run before the end of the play**, add a task to flush them using the meta module, which executes Ansible actions. I have updated my playbook to perform sleep on the remote server and then observe the time with the handler to make sure we have new instance of handler being executed after every task:

---

- name: Handlers Example

hosts: server1

gather\_facts: false

become: true

tasks:

- name: Sleep for 2 seconds

command: sleep 2

notify: run\_handler

- name: Flush handlers

meta: flush\_handlers

- name: Sleep for 5 seconds

command: sleep 5

notify: run\_handler

- name: Flush handlers

meta: flush\_handlers

- name: Sleep for 7 seconds

command: sleep 7

notify: run\_handler

handlers:

- name: run\_handler

debug:

msg: "Today's date and time: {{ '%d-%m-%Y %H:%M:%S:%s' | strftime }}"

* Here my first task will sleep for 2 minutes and then execute the handler
* Then we flush the handler and call the second task which will sleep for 5 seconds and call the handler
* Then we will again flush the handler and sleep for another 7 seconds before calling the handler for third time
* In the handler I have used strftime from our Jinja syntax to get the time of execution for the handler

Let us execute the playbook:

[ansible@controller ~]$ ansible-playbook ansible-handlers.yml

PLAY [Handlers Example] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Sleep for 2 seconds] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1]

RUNNING HANDLER [run\_handler] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server1] => {

"msg": "Today's date and time: 25-09-2020 10:05:45:1601028345"

}

TASK [Sleep for 5 seconds] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1]

RUNNING HANDLER [run\_handler] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server1] => {

"msg": "Today's date and time: 25-09-2020 10:05:51:1601028351"

}

TASK [Sleep for 7 seconds] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1]

RUNNING HANDLER [run\_handler] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server1] => {

"msg": "Today's date and time: 25-09-2020 10:05:58:1601028358"

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server1 : ok=6 changed=3 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

So now our handler has executed three times because we were flushing the handler after every task execution and the time of the handler execution is also different so life is good.

Advertisement

**Using variables with handlers**

You may want your Ansible handlers to use variables. For example, if the name of a service varies slightly by distribution, you want your output to show the exact name of the restarted service for each target machine. Avoid placing variables in the name of the handler. Since handler names are templated early on, Ansible may not have a value available for a handler name like this:

handlers:

# This handler name may cause your play to fail!

- name: Restart "{{ web\_service\_name }}"

If the variable used in the handler name is not available, **the entire play fails**. Changing that variable mid-play will not result in newly created handler.

So we should always use a default variable when in such scenarios as I have used in the sample playbook:

---

- name: Handlers Example

hosts: server1

gather\_facts: false

become: true

vars:

pkg: httpd

tasks:

- name: Installing vsftpd

debug:

msg: "restarting vsftp"

changed\_when: true

notify: restart vsftpd

handlers:

- name: "restart {{ pkg | default('vsftpd')}}"

debug:

msg: "Restarting {{ pkg | default('vsftpd')}}"

Now I have defined a pkg variable but if for some reason handlers fails to get this value then it will consider the default variable value instead of failing the entire play:

Let us execute this play:

[ansible@controller ~]$ ansible-playbook ansible-handlers.yml

PLAY [Handlers Example] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TASK [Installing vsftpd] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

changed: [server1] => {

"msg": "restarting vsftp"

}

RUNNING HANDLER [restart vsftpd] \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

ok: [server1] => {

"msg": "Restarting vsftpd"

}

PLAY RECAP \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

server1 : ok=2 changed=1 unreachable=0 failed=0 skipped=0 rescued=0 ignored=0

So our playbook was successfully executed the task and the handler.