

# Red Hat Enterprise Linux Automation with Ansible

Version 9.0 ▾

Translations ▾



☆ Bookmark this page

|  |  |  |  |
|--|--|--|--|
| P  | (/rol/app/courses/rh294-9.0/pages/pr01)    | (/rol/app/courses/rh294-9.0/pages/pr01s02) | (/rol/app/courses/rh294-9.0/pages/pr01s03) |
| (/rol/app/courses/rh294-9.0/pages/pr01)    | (/rol/app/courses/rh294-9.0/pages/pr01s04) | (/rol/app/courses/rh294-9.0/pages/pr01s05) | (/rol/app/courses/rh294-9.0/pages/pr01s06) |
| (/rol/app/courses/rh294-9.0/pages/ch01s03) | (/rol/app/courses/rh294-9.0/pages/ch01s04) | (/rol/app/courses/rh294-9.0/pages/ch01s05) | (/rol/app/courses/rh294-9.0/pages/ch01s06) |
| (/rol/app/courses/rh294-9.0/pages/ch02)    | (/rol/app/courses/rh294-9.0/pages/ch02s02) | (/rol/app/courses/rh294-9.0/pages/ch02s03) | (/rol/app/courses/rh294-9.0/pages/ch02s04) |
| (/rol/app/courses/rh294-9.0/pages/ch02s04) | (/rol/app/courses/rh294-9.0/pages/ch02s05) | (/rol/app/courses/rh294-9.0/pages/ch02s06) | (/rol/app/courses/rh294-9.0/pages/ch02s07) |
| (/rol/app/courses/rh294-9.0/pages/ch02s07) | (/rol/app/courses/rh294-9.0/pages/ch02s08) | (/rol/app/courses/rh294-9.0/pages/ch02s09) | (/rol/app/courses/rh294-9.0/pages/ch02s10) |
| (/rol/app/courses/rh294-9.0/pages/ch02s10) | (/rol/app/courses/rh294-9.0/pages/ch02s11) | (/rol/app/courses/rh294-9.0/pages/ch02s12) | (/rol/app/courses/rh294-9.0/pages/ch02s13) |
| (/rol/app/courses/rh294-9.0/pages/ch03s03) | (/rol/app/courses/rh294-9.0/pages/ch03s04) | (/rol/app/courses/rh294-9.0/pages/ch03s05) | (/rol/app/courses/rh294-9.0/pages/ch03s06) |
| (/rol/app/courses/rh294-9.0/pages/ch03s06) | (/rol/app/courses/rh294-9.0/pages/ch03s07) | (/rol/app/courses/rh294-9.0/pages/ch03s08) | (/rol/app/courses/rh294-9.0/pages/ch03s09) |
| (/rol/app/courses/rh294-9.0/pages/ch04)    | (/rol/app/courses/rh294-9.0/pages/ch04s02) | (/rol/app/courses/rh294-9.0/pages/ch04s03) | (/rol/app/courses/rh294-9.0/pages/ch04s04) |
| (/rol/app/courses/rh294-9.0/pages/ch04s04) | (/rol/app/courses/rh294-9.0/pages/ch04s05) | (/rol/app/courses/rh294-9.0/pages/ch04s06) | (/rol/app/courses/rh294-9.0/pages/ch04s07) |
| (/rol/app/courses/rh294-9.0/pages/ch04s07) | (/rol/app/courses/rh294-9.0/pages/ch04s08) | (/rol/app/courses/rh294-9.0/pages/ch04s09) | (/rol/app/courses/rh294-9.0/pages/ch04s10) |
| (/rol/app/courses/rh294-9.0/pages/ch05s02) | (/rol/app/courses/rh294-9.0/pages/ch05s03) | (/rol/app/courses/rh294-9.0/pages/ch05s04) | (/rol/app/courses/rh294-9.0/pages/ch05s05) |
| (/rol/app/courses/rh294-9.0/pages/ch05s05) | (/rol/app/courses/rh294-9.0/pages/ch05s06) | (/rol/app/courses/rh294-9.0/pages/ch05s07) | (/rol/app/courses/rh294-9.0/pages/ch05s08) |
| (/rol/app/courses/rh294-9.0/pages/ch06s02) | (/rol/app/courses/rh294-9.0/pages/ch06s03) | (/rol/app/courses/rh294-9.0/pages/ch06s04) | (/rol/app/courses/rh294-9.0/pages/ch06s05) |
| (/rol/app/courses/rh294-9.0/pages/ch06s05) | (/rol/app/courses/rh294-9.0/pages/ch06s06) | (/rol/app/courses/rh294-9.0/pages/ch06s07) | (/rol/app/courses/rh294-9.0/pages/ch06s08) |
| (/rol/app/courses/rh294-9.0/pages/ch07s02) | (/rol/app/courses/rh294-9.0/pages/ch07s03) | (/rol/app/courses/rh294-9.0/pages/ch07s04) | (/rol/app/courses/rh294-9.0/pages/ch07s05) |
| (/rol/app/courses/rh294-9.0/pages/ch07s05) | (/rol/app/courses/rh294-9.0/pages/ch07s06) | (/rol/app/courses/rh294-9.0/pages/ch07s07) | (/rol/app/courses/rh294-9.0/pages/ch07s08) |
| (/rol/app/courses/rh294-9.0/pages/ch07s08) | (/rol/app/courses/rh294-9.0/pages/ch07s09) | (/rol/app/courses/rh294-9.0/pages/ch07s10) | (/rol/app/courses/rh294-9.0/pages/ch07s11) |
| (/rol/app/courses/rh294-9.0/pages/ch07s11) | (/rol/app/courses/rh294-9.0/pages/ch07s12) | (/rol/app/courses/rh294-9.0/pages/ch07s13) | (/rol/app/courses/rh294-9.0/pages/ch07s14) |
| (/rol/app/courses/rh294-9.0/pages/ch08s02) | (/rol/app/courses/rh294-9.0/pages/ch08s03) | (/rol/app/courses/rh294-9.0/pages/ch08s04) | (/rol/app/courses/rh294-9.0/pages/ch08s05) |
| (/rol/app/courses/rh294-9.0/pages/ch08s05) | (/rol/app/courses/rh294-9.0/pages/ch08s06) | (/rol/app/courses/rh294-9.0/pages/ch08s07) | (/rol/app/courses/rh294-9.0/pages/ch08s08) |
| (/rol/app/courses/rh294-9.0/pages/ch09s02) | (/rol/app/courses/rh294-9.0/pages/ch09s03) | (/rol/app/courses/rh294-9.0/pages/ch09s04) | (/rol/app/courses/rh294-9.0/pages/ch09s05) |
| (/rol/app/courses/rh294-9.0/pages/ch09s05) | (/rol/app/courses/rh294-9.0/pages/ch09s06) | (/rol/app/courses/rh294-9.0/pages/ch09s07) | (/rol/app/courses/rh294-9.0/pages/ch09s08) |
| (/rol/app/courses/rh294-9.0/pages/ch09s08) | (/rol/app/courses/rh294-9.0/pages/ch09s09) | (/rol/app/courses/rh294-9.0/pages/ch09s10) | (/rol/app/courses/rh294-9.0/pages/ch09s11) |
| (/rol/app/courses/rh294-9.0/pages/ch09s11) | (/rol/app/courses/rh294-9.0/pages/ch09s12) | (/rol/app/courses/rh294-9.0/pages/ch09s13) | (/rol/app/courses/rh294-9.0/pages/ch09s14) |
| (/rol/app/courses/rh294-9.0/pages/ch10s02) | (/rol/app/courses/rh294-9.0/pages/ch10s03) | (/rol/app/courses/rh294-9.0/pages/ch10s04) | (/rol/app/courses/rh294-9.0/pages/ch10s05) |
| (/rol/app/courses/rh294-9.0/pages/ch10s05) | (/rol/app/courses/rh294-9.0/pages/ch10s06) | (/rol/app/courses/rh294-9.0/pages/ch10s07) | (/rol/app/courses/rh294-9.0/pages/ch10s08) |

[Previous](#)[Next](#)

## Guided Exercise: Handling Task Failure

Explore different ways to handle task failure in an Ansible Playbook.

### Outcomes

- Ignore failed commands during the execution of playbooks.



- Force execution of handlers.
- Override what constitutes a failure in tasks.
- Override the changed state for tasks.
- Implement `block`, `rescue`, and `always` in playbooks.

As the student user on the workstation machine, use the `lab` command to prepare your system for this exercise.

This command prepares your environment and ensures that all required resources are available.

```
[student@workstation ~]$ lab start control-errors
```

## Instructions

1. On the workstation machine, change to the `/home/student/control-errors` directory.

```
[student@workstation ~]$ cd ~/control-errors
[student@workstation control-errors]$
```

2. The `lab` command created an Ansible configuration file as well as an inventory file, which contains the `servera.lab.example.com` server in the `databases` group. Review the file before proceeding.

```
[student@workstation control-errors]$ cat inventory
[databases]
servera.lab.example.com
```

3. Create a playbook named `playbook.yml` that contains a play with two tasks. Write the first task with a deliberate error to cause failure.

- 3.1. Open the playbook in a text editor. Define three variables: `web_package` with a value of `http`, `db_package` with a value of `mariadb-server`, and `db_service` with a value of `mariadb`. These variables are used to install the required packages and start the server.

The `http` value is an intentional error in the package name. The (intentionally incorrect) should consist of the following content:

```
---
- name: Task Failure Exercise
  hosts: databases
  vars:
    web_package: http
    db_package: mariadb-server
    db_service: mariadb
```

- 3.2. Define two tasks that use the `ansible.builtin.dnf` module and the two variables, `web_package` and `db_package`. The tasks should install the required packages and read as follows:

```
tasks:
  - name: Install {{ web_package }} package
    ansible.builtin.dnf:
      name: "{{ web_package }}"
      state: present

  - name: Install {{ db_package }} package
    ansible.builtin.dnf:
      name: "{{ db_package }}"
      state: present
```

4. Run the playbook and watch the output of the play.

```
[student@workstation control-errors]$ ansible-navigator run -m stdout playbook.yml

PLAY [Task Failure Exercise] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Install http package] *****
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failures": ["No package http available."], "msg": "Failed to
install some of the specified packages", "rc": 1, "results": []}

PLAY RECAP *****
servera.lab.example.com : ok=1   changed=0   unreachable=0   failed=1   skipped=0   rescued=0   ignored=0
Please review the log for errors.
```

The task failed because there is no existing package called http. Because the first task failed, the second task did not run.

5. Update the first task to ignore any errors by adding the `ignore_errors` keyword. The tasks should consist of the following content:

```
tasks:
  - name: Install {{ web_package }} package
    ansible.builtin.dnf:
      name: "{{ web_package }}"
      state: present
      ignore_errors: true

  - name: Install {{ db_package }} package
    ansible.builtin.dnf:
      name: "{{ db_package }}"
      state: present
```

6. Run the playbook again and watch the output of the play.

```
[student@workstation control-errors]$ ansible-navigator run -m stdout playbook.yml

PLAY [Task Failure Exercise] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Install http package] *****
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failures": ["No package http available."], "msg": "Failed to
install some of the specified packages", "rc": 1, "results": []}
...ignoring

TASK [Install mariadb-server package] *****
changed: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com : ok=3   changed=1   unreachable=0   failed=0   skipped=0   rescued=0   ignored=1
```

Although the first task failed, Ansible executed the second one.

7. In this step, you set up a `block` keyword, so that you can experiment with how they work.
- 7.1. Update the playbook by nesting the first task in a `block` clause. Remove the line that sets `ignore_errors: true`. The block should consist of the following content:

```
- name: Attempt to set up a webserver
  block:
    - name: Install {{ web_package }} package
      ansible.builtin.dnf:
        name: "{{ web_package }}"
        state: present
```

- 7.2. Nest the task that installs the mariadb-server package in a rescue clause. If the task listed in the block clause fails, then this task runs. The block clause should consist of the following content:

```
rescue:
  - name: Install {{ db_package }} package
    ansible.builtin.dnf:
      name: "{{ db_package }}"
      state: present
```

- 7.3. Finally, add an always clause to start the database server upon installation using the ansible.builtin.service module. The always clause should consist of the following content:

```
always:
  - name: Start {{ db_service }} service
    ansible.builtin.service:
      name: "{{ db_service }}"
      state: started
```

- 7.4. The completed task should consist of the following content:

```
tasks:
  - name: Attempt to set up a webserver
    block:
      - name: Install {{ web_package }} package
        ansible.builtin.dnf:
          name: "{{ web_package }}"
          state: present
    rescue:
      - name: Install {{ db_package }} package
        ansible.builtin.dnf:
          name: "{{ db_package }}"
          state: present
    always:
      - name: Start {{ db_service }} service
        ansible.builtin.service:
          name: "{{ db_service }}"
          state: started
```

8. Run the playbook again and observe the output.

- 8.1. Run the playbook. The task in the block that makes sure web\_package is installed fails, which causes the task in the rescue block to run. The task in the always block then runs.

```
[student@workstation control-errors]$ ansible-navigator run -m stdout playbook.yml

PLAY [Task Failure Exercise] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Install http package] *****
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failures": ["No package http available."], "msg": "Failed to install some of the specified packages", "rc": 1, "results": []}

TASK [Install mariadb-server package] *****
ok: [servera.lab.example.com]

TASK [Start mariadb service] *****
changed: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=1    ignored=0
```

- 8.2. Edit the playbook, correcting the value of the web\_package variable to read httpd. This causes the task in the block to succeed the next time you run the playbook.

```
vars:
  web_package: httpd
  db_package: mariadb-server
  db_service: mariadb
```

- 8.3. Run the playbook again. This time, the task in the block does not fail. This causes the task in the `rescue` section to be ignored. The task in the `always` section still runs.

```
[student@workstation control-errors]$ ansible-navigator run -m stdout playbook.yml

PLAY [Task Failure Exercise] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Install httpd package] *****
changed: [servera.lab.example.com]

TASK [Start mariadb service] *****
ok: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com : ok=3    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

9. This step explores how to control the condition that causes a task to be reported as "changed" for a managed host.

- 9.1. Edit the playbook to add two tasks to the start of the play, preceding the `block` clause. The first task uses the `ansible.builtin.command` module to run the `date` command and register the result in the `command_result` variable. The second task uses the `ansible.builtin.debug` module to print the standard output of the first task's command.

```
tasks:
  - name: Check local time
    ansible.builtin.command: date
    register: command_result

  - name: Print local time
    ansible.builtin.debug:
      var: command_result.stdout
```

- 9.2. Run the playbook. You should see that the first task, which runs the `ansible.builtin.command` module, reports `changed`, even though it did not change the remote system; it only collected information about the time. That is because the `ansible.builtin.command` module cannot tell the difference between a command that collects data and a command that changes state.

```
[student@workstation control-errors]$ ansible-navigator run -m stdout playbook.yml

PLAY [Task Failure Exercise] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Check local time] *****
changed: [servera.lab.example.com]

TASK [Print local time] *****
ok: [servera.lab.example.com] => {
  "command_result.stdout": "Tue Jul  5 03:04:51 PM EDT 2022"
}

TASK [Install httpd package] *****
ok: [servera.lab.example.com]

TASK [Start mariadb service] *****
ok: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com : ok=5    changed=1    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

If you run the playbook again, the `Check local time` task returns `changed` again.

- 9.3. That `ansible.builtin.command` task should not report `changed` every time it runs because it is not changing the managed host. Because you know that the task never changes a managed host, add the line `changed_when: false` to the task to suppress the change.

```
tasks:
  - name: Check local time
    ansible.builtin.command: date
    register: command_result
    changed_when: false

  - name: Print local time
    ansible.builtin.debug:
      var: command_result.stdout
```

- 9.4. Run the playbook again and notice that the task now reports `ok`, but the task is still being run and is still saving the time in the variable.

```
[student@workstation control-errors]$ ansible-navigator run -m stdout playbook.yml

PLAY [Task Failure Exercise] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Check local time] *****
ok: [servera.lab.example.com]

TASK [Print local time] *****
ok: [servera.lab.example.com] => {
  "command_result.stdout": "Tue Jul  5 03:06:43 PM EDT 2022"
}

TASK [Install httpd package] *****
ok: [servera.lab.example.com]

TASK [Start mariadb service] *****
ok: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=0    ignored=0
```

0. As a final exercise, edit the playbook to explore how the `failed_when` keyword interacts with tasks.

- 10.1. Edit the `Install {{ web_package }}` package task so that it reports as having failed when `web_package` has the value `httpd`. Because this is the case, the task reports a failure when you run the play.

Be careful with your indentation to ensure that the keyword is correctly set on the task.

```
block:
  - name: Install {{ web_package }} package
    ansible.builtin.dnf:
      name: "{{ web_package }}"
      state: present
      failed_when: web_package == "httpd"
```

- 10.2. Run the playbook.

```
[student@workstation control-errors]$ ansible-navigator run -m stdout playbook.yml

PLAY [Task Failure Exercise] *****

TASK [Gathering Facts] *****
ok: [servera.lab.example.com]

TASK [Check local time] *****
ok: [servera.lab.example.com]

TASK [Print local time] *****
ok: [servera.lab.example.com] => {
  "command_result.stdout": "Tue Jul  5 03:08:41 PM EDT 2022"
}

TASK [Install httpd package] *****
fatal: [servera.lab.example.com]: FAILED! => {"changed": false, "failed_when_result": true, "msg": "Nothing to do", "rc": 0, "results": []}

TASK [Install mariadb-server package] *****
ok: [servera.lab.example.com]

TASK [Start mariadb service] *****
ok: [servera.lab.example.com]

PLAY RECAP *****
servera.lab.example.com : ok=5    changed=0    unreachable=0    failed=0    skipped=0    rescued=1    ignored=0
```

Look carefully at the output. The `failed_when` keyword changes the status that the task reports *after* the task runs; it does not change the behavior of the task itself.

However, the reported failure might change the behavior of the rest of the play. Because that task was in a block and reported that it failed, the `Install mariadb-server package` task in the block's `rescue` section was run.

## Finish

On the workstation machine, change to the student user home directory and use the `lab` command to complete this exercise. This step is important to ensure that resources from previous exercises do not impact upcoming exercises.

```
[student@workstation ~]$ lab finish control-errors
```

</rol/app/courses/rh294-9.0/pages/ch04s05>


</rol/app/cou>



Red Hat  
Learning Community

Discuss Red Hat Enterprise Linux Automation with Ansible  
(<https://learn.redhat.com/t5/RH294-Red-Hat-Linux-Automation/gh-p/RH294RedHatLinuxAutomationwithAnsible>)









Red Hat Linux Automation with Ansible (RH294)

Haley\_Ruccio Jul 24, 2023

Learn how to automate Linux system administration tasks with Red Hat Ansible...




 3  1  980




curl finds no route to host?

TPeters Jul 23, 2023

I am doing the self-paced course RH294 . With two exercises (Ch.3 final Lab and Ch....




 1  6  1702



Welcome to the Red Hat Linux Automation with Ansible (RH294)...

cschunke Jul 18, 2023

We are excited to launch a space dedicated to the Red Hat Training course Red Hat...

 8  1  351

Revision: rh294-9.0-4e8f1e1



- Privacy Policy ([http://s.bl-1.com/h/cZrgWbQn?url=https://www.redhat.com/en/about/privacy-policy?extIdCarryOver=true&sc\\_cid=701f2000001D8QoAAK](http://s.bl-1.com/h/cZrgWbQn?url=https://www.redhat.com/en/about/privacy-policy?extIdCarryOver=true&sc_cid=701f2000001D8QoAAK))

Terms of Use (<https://www.redhat.com/en/about/terms-use>)

Release Notes (<https://learn.redhat.com/t5/Red-Hat-Learning-Subscription/bg-p/RedHatLearningSubscriptionGroupblog-board>)
- Red Hat Training Policies (<http://s.bl-1.com/h/cZrb2DXG?url=https://www.redhat.com/en/about/red-hat-training-policies>)

All policies and guidelines (<https://www.redhat.com/en/about/all-policies-guidelines>)

Cookie preferences