链滴

ansible 的 playbook（2）之创建可重复使用的 playbook-1

作者：SmiteLi

原文链接：https://ld246.com/article/1538282435726

来源网站：链滴

一、Creating Reusable Playbooks

- Including and Importing
- Roles

While it is possible to write a playbook in one very large file (and you might start out learning playbooks this way), eventually you' ll want to reuse files and start to organize things. In Ans ble, there are three ways to do this: includes, imports, and roles.

Includes and imports (added in Ansible version 2.4) allow users to break up large playbooks in o smaller files, which can be used across multiple parent playbooks or even multiple times wi hin the same Playbook.

Roles allow more than just tasks to be packaged together and can include variables, handlers, or even modules and other plugins. Unlike includes and imports, roles can also be uploaded nd shared via Ansible Galaxy.

Dynamic vs. Static

Ansible has two modes of operation for reusable content: dynamic and static.

In Ansible 2.0, the concept of dynamic includes was introduced. Due to some limitations with making all includes dynamic in this way, the ability to force includes to be static was introduc d in Ansible 2.1. Because the include task became overloaded t encompass both static and dynamic syntaxes, and because the default behavior of an include could change based on other options set on the Task, Ansible 2.4 introduces the concept of i clude vs. import.

If you use any import* Task (import_playbook, import_tasks, etc.), it will be static. If you use an include* Task (include_tasks, include_role, etc.), it will be dynamic.

The bare include task (which was used for both Task files and Playbook-level includes) is still vailable, however it is now considered deprecated.

Differences Between Static and Dynamic

The two modes of operation are pretty simple:

- Ansible pre-processes all static imports during Playbook parsing time.
- Dynamic includes are processed during runtime at the point in which that task is encounter d.

When it comes to Ansible task options like tags and conditional statements (when:):

- For static imports, the parent task options will be copied to all child tasks contained within he import.
- For dynamic includes, the task options will only apply to the dynamic task as it is evaluate , and will not be copied to child tasks.

Note

Roles are a somewhat special case. Prior to Ansible 2.3, roles were always statically included v a the special roles: option for a given play and were always executed first before any other pl y tasks (unless pre_tasks were used). Roles can still be used this way, however, Ansible 2.3 intr duced the include_role option to allow roles to be executed inline with other tasks.

Tradeoffs and Pitfalls Between Includes and Imports

Using include* vs. import* has some advantages as well as some tradeoffs which users should consider when choosing to use each:

The primary advantage of using include* statements is looping. When a loop is used with an i clude, the included tasks or role will be executed once for each item in the loop.

Using include* does have some limitations when compared to import* statements:

- Tags which only exist inside a dynamic include will not show up in --list-tags output.
- Tasks which only exist inside a dynamic include will not show up in --list-tasks output.
- You cannot use notify to trigger a handler name which comes from inside a dynamic incl de (see note below).
- You cannot use --start-at-task to begin execution at a task inside a dynamic include.

Using import* can also have some limitations when compared to dynamic includes:

- As noted above, loops cannot be used with imports at all.
- When using variables for the target file or role name, variables from inventory sources (host group vars, etc.) cannot be used.

Note:

Regarding the use of notify for dynamic tasks: it is still possible to trigger the dynamic include itself, which would result in all tasks within the include being run.