Working with command output and prompts in network modules

Conditionals in networking modules

Ansible allows you to use conditionals to control the flow of your playbooks. Ansible networking command modules use the following unique conditional statements.

- `eq` - Equal
- `neq` - Not equal
- `gt` - Greater than
- `ge` - Greater than or equal
- `lt` - Less than
- `le` - Less than or equal
- `contains` - Object contains specified item

Conditional statements evaluate the results from the commands that are executed remotely on the device. Once the task executes the command set, the `wait_for` argument can be used to evaluate the results before returning control to the Ansible playbook.

For example:

```
---
- name: wait for interface to be admin enabled
  arista.eos.eos_command:
      commands:
          - show interface Ethernet4 | json
      wait_for:
          - "result[0].interfaces.Ethernet4.interfaceStatus eq connected"
```

In the above example task, the command `show interface Ethernet4 | json` is executed on the remote device and the results are evaluated. If the path `(result[0].interfaces.Ethernet4.interfaceStatus)` is not equal to "connected", then the command is retried. This process continues until either the condition is satisfied or the number of retries has expired (by default, this is 10 retries at 1 second intervals).

The commands module can also evaluate more than one set of command results in an interface. For instance:

```
---
- name: wait for interfaces to be admin enabled
  arista.eos.eos_command:
      commands:
          - show interface Ethernet4 | json
          - show interface Ethernet5 | json
      wait_for:
          - "result[0].interfaces.Ethernet4.interfaceStatus eq connected"
          - "result[1].interfaces.Ethernet5.interfaceStatus eq connected"
```

In the above example, two commands are executed on the remote device, and the results are evaluated. By specifying the result index value (0 or 1), the correct result output is checked against the conditional.

The `wait_for` argument must always start with result and then the command index in `[]`, where `0` is the first command in the commands list, `1` is the second command, `2` is the third and so on.

Handling prompts in network modules

Network devices may require that you answer a prompt before performing a change on the device. Individual network modules such as :ref:`cisco.ios.ios_command <ansible_collections.cisco.ios.ios_command_module>` and :ref:`cisco.nxos.nxos_command <ansible_collections.cisco.nxos.nxos_command_module>` can handle this with a `prompt` parameter.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\user_guide\[ansible-devel][docs][docsite][rst][network][user_guide]network_working_with_command_output.rst`, line 77); *backlink*

Unknown interpreted text role "ref".

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\user_guide\[ansible-devel][docs][docsite][rst][network][user_guide]network_working_with_command_output.rst`, line 77); *backlink*

Unknown interpreted text role "ref".

Note

`prompt` is a Python regex. If you add special characters such as `?` in the `prompt` value, the prompt won't match and you will get a timeout. To avoid this, ensure that the `prompt` value is a Python regex that matches the actual device prompt. Any special characters must be handled correctly in the `prompt` regex.

You can also use the :ref:`ansible.netcommon.cli_command <ansible_collections.ansible.netcommon.cli_command_module>` to handle multiple prompts.

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\user_guide\[ansible-devel][docs][docsite][rst][network][user_guide]network_working_with_command_output.rst`, line 83); ***backlink***

Unknown interpreted text role "ref".

```
---
- name: multiple prompt, multiple answer (mandatory check for all prompts)
  ansible.netcommon.cli_command:
    command: "copy sftp sftp://user@host//user/test.img"
    check_all: True
    prompt:
      - "Confirm download operation"
      - "Password"
      - "Do you want to change that to the standby image"
    answer:
      - 'y'
      - <password>
      - 'y'
```

You must list the prompt and the answers in the same order (that is, prompt[0] is answered by answer[0]).

In the above example, `check_all: True` ensures that the task gives the matching answer to each prompt. Without that setting, a task with multiple prompts would give the first answer to every prompt.

In the following example, the second answer would be ignored and `y` would be the answer given to both prompts. That is, this task only works because both answers are identical. Also notice again that `prompt` must be a Python regex, which is why the `?` is escaped in the first prompt.

```
---
 - name: reboot ios device
   ansible.netcommon.cli_command:
     command: reload
     prompt:
       - Save\?
       - confirm
     answer:
       - y
       - y
```

System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\ansible-devel\docs\docsite\rst\network\user_guide\[ansible-devel][docs][docsite][rst][network][user_guide]network_working_with_command_output.rst`, line 120)

Unknown directive type "seealso".

```
.. seealso::

  `Rebooting network devices with Ansible <https://www.ansible.com/blog/rebooting-network-devices-with-an
      Examples using ``wait_for``, ``wait_for_connection``, and ``prompt`` for network devices.

  `Deep dive on cli_command <https://www.ansible.com/blog/deep-dive-on-cli-command-for-network-automation
      Detailed overview of how to use the ``cli_command``.
```