Challenge Exercise 03

Using Ansible Jinja Filters on Network Device Data

Prepared by:

Network to Code, LLC

2020

Exercise Overview

This exercise requires you to work with various Ansible Jinja2 filters to process structured data that you have received from a network device. Quite often you will need to search through or filter large amounts of data and prepare the results for printing or further processing.

You will get started by loading the JSON output of a **show interfaces** command on an NX-OS switch. This output is provided as a file named **show_int_nxos.json** which should be placed next to the exercise playbook named **pb_filters_challenge.yml** .

This is an "offline" task and you do not need to deploy configurations to real devices. You will be executing the playbook locally and use debug messages to print the results.

IMPORTANT: You can solve all tasks using built-in Ansible filters only. You do not need to use any custom filters, loops/conditionals or python code.

For reference, you will need to use the following documentation:

- Ansible Filters https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters.html
- Jinja2 Filters https://jinja.palletsprojects.com/en/2.11.x/templates/#builtin-filters
- The ipaddr Filter https://docs.ansible.com/ansible/latest/user_guide/playbooks_filters_ipaddr.html#playbooks-filters-ipaddr
- Jinja2 Tests for expressions https://jinja.palletsprojects.com/en/2.11.x/templates/#list-of-builtin-tests

If you get stuck, take a look at the **Using Jinja2 Filters bonus lab** from the Network Programmability and Automation Course - you can find it here. Remember you need to be logged in to Github to open the private course repository!

In the folder with the challenge files, open the **pb_filters_challenge.yml** file in your editor and start implementing the tasks below.

Task 1 Extracting Interface Names and Attributes

In this task you will be extracting various properties from a large list of interfaces. Use the debug module to print the required information in the steps below. The focus is on using filters and applying them to data!

Step 1 - Fill in the first task. From the **show_int** variable, extract the total number of interfaces on the switch.

Step 2 - Set a new fact called **intf_lis**t that contains **only** the list of interfaces from the nested structure contained by the **show_int** variable.

Step 3 - Print a list of all interface names contained in **intf_list**. Do not print any other additional data.
Sample output: "List of interface names: ['mgmt0', 'Ethernet1/1', 'Ethernet1/2']"

Step 4 - Print the list from Step 3 as comma-separated values.
Sample output: "Comma-separated interface names: mgmt0, Ethernet1/1, Ethernet1/2"

Step 5 - Print a list containing the hardware addresses (**eth_hw_addr** field) of all the interfaces. Ensure they are in all **UPPERCASE** letters.
Sample output: "All MAC addresses: ['92AD.0005.0000', '92AD.0005.0008', '92AD.0005.0009']"

Step 6 - Convert the MAC list from Step 5 from Cisco format into Linux format.
Sample output: "All MAC addresses: ['92:ad:00:05:00:00', '92:ad:00:05:00:08', '92:ad:00:05:00:09']"

Hint: The hwaddr filter (documented here) can do the conversion. Use the **map** filter to apply **hwaddr** to the whole list at once.

Task 2 Working with IP addresses

Here you will be practicing the use of the **ipaddr** family of filters that perform all the necessary subnetting arithmetic for you. The referenced ansible documentation has all you need.

Step 1 - Set a new fact called **mgmt0_prefix** containing the IP address and the mask of the interface sourced from the **intf_list** data.
Sample target value: mgmt0_prefix = "10.0.0.4/24"

Step 2 - Print the usable IP range calculated from the **mgmt0_prefix** variable.
Sample output: "Usable IP range: 10.0.0.1-10.0.0.254"

Step 3 - Print the broadcast IP range for the **mgmt0_prefix**.
Sample output: "Broadcast address: 10.0.0.255"

Step 4 - Print the 555th usable IP in the **mgmt0_prefix** subnet.

Bonus Task 3

This **optional bonus task** is a little bit more difficult. You will need to perform conditional filtering of data using arbitrary custom fields and values (e.g. admin_state and “up” or “down”).
It requires the usage of the **selectattr** filter together with the **map** filter you used in **Task 1**. Remember you can chain multiple filters together to get closer and closer to the desired result.

Task 1 - Count the number of interfaces (from the **intf_list** variable) that are in Admin Up state (**admin_state** attribute).
Desired output: "Number of interfaces in Admin/UP state: 111"

Task 2 - Print a comma-separated list of all the interface names in Admin/UP state.
Desired output: "Interfaces in Admin/UP state: mgmt0, Ethernet1/1, Ethernet1/2, ... <output snipped>"

Task 3 - Print a comma-separated list of all the interface names in Admin/UP (“admin_state” attribute) **AND** Operational/UP (“state” attribute).
Desired output: "Interfaces in Admin/UP Operational/UP state: mgmt0, Ethernet1/1, Ethernet1/2, Ethernet1/3, Ethernet1/4, Ethernet1/5, Ethernet1/6, Ethernet1/7, Ethernet1/8"

HINT: You may also solve this section using the json_query filter instead of **selectattr** & **map**. Both approaches are valid.

Submitting Your Solution

You can submit your solution at any time over the next week to the following email address: baxter-enablement@networktocode.com. While this is optional, the target should be to submit by next Monday August 10th.

If you have any questions on the challenge or want to do a 1:1 session or a group discussion on the challenge, email the question or schedule a Zoom session by also emailing: baxter-enablement@networktocode.com.

Reference Enablement Material

The following bite sized session can be used as reference:

- Network Automation and Programmability Course - Ansible Filters Labs

Tips

- Some of the tasks require using multiple filters together in what's called a chain.
- While you can ultimately solve each task with one filter chain, it may be useful to first take it step by step with separate debugs for each additional filter you're adding.