Gmail **Syed Asif <sydasif78@gmail.com>**

[PyNet Learning Python] - Lesson7 / Jinja2, YAML and JSON

1 message

Kirk Byers <support@twb-tech.com> Tue, Oct 20, 2020 at 8:00 PM
To: sydasif78@gmail.com

Syd

Note: There is a table of contents for each video at the bottom of this email including timestamps to where various content is located. This should be helpful in navigating the videos.

In this email of Learning Python we are going to cover the following:

1. **Jinja2 Basics**
   Video link https://vimeo.com/257997257
   Length is 7 minutes

2. **Jinja2 For-Loops and Conditionals**
   Video link https://vimeo.com/257999160
   Length is 9 minute

3. **Jinja2 and CSV**
   Video link https://vimeo.com/258142987
   Length is 5 minutes

4. **Jinja2 Dictionaries and Nested Loops**
   Video link https://vimeo.com/258145504
   Length is 11 minutes

5. **YAML Basics**
   Video link https://vimeo.com/258161182
   Length is 9 minutes

6. **YAML Part2**
   Video link https://vimeo.com/258169427
   Length is 10 minutes

7. **Using Python to Write YAML**
   Video link https://vimeo.com/258171559
   Length is 3 minutes

8. **JSON**
   Video link https://vimeo.com/258178243

Length is 5 minutes

9. **Managing Data Structures**
   Video link https://vimeo.com/258181273
   Length is 5 minutes

Additional Content:

Jinja2 Documentation

YAML Syntax Basics

Exercises

Reference code for these exercises is posted on GitHub at:
https://github.com/ktbyers/pynet/tree/master/learning_python/lesson7

1a. Use Jinja2 templating to render the following:

```
vlan
   name
```

Your template should be inside of your Python program for simplicity.

The output from processing your template should be as follows. This should be printed to stdout.

```
vlan 400
   name red400
```

1b. Using a conditional in a Jinja2 template, generate the following output:

```
crypto isakmp policy 10
 encr aes
 authentication pre-share
 group 5
crypto isakmp key my_key address 1.1.1.1 no-xauth
crypto isakmp keepalive 10 periodic
```

The encryption of aes, and the Diffie-Hellman group should be variables in the template.

Additionally this entire ISAKMP section should only be added if the isakmp_enable variable is set to True.

Your template should be inside your Python program for simplicity.

1c. Using Jinja2 templating and a for-loop inside the template, generate the following configuration snippet:

```
vlan 501
   name blue501
vlan 502
   name blue502
vlan 503
   name blue503
vlan 504
   name blue504
vlan 505
   name blue505
vlan 506
   name blue506
vlan 507
   name blue507
vlan 508
   name blue508
```

Your template should be inside your Python program for simplicity.

It is fine for your VLAN IDs to be out of order in the generated configuration (for example, VLAN ID 508 can come before VLAN ID 504).

2. Using Python and Jinja2 templating generate the following OSPF configuration:

```
interface vlan 1
   ip ospf priority 100

router ospf 10
   passive-interface default
   no passive-interface Vlan1
   no passive-interface Vlan2
   network 10.10.10.0/24 area 0.0.0.0
   network 10.10.20.0/24 area 0.0.0.0
   network 10.10.30.0/24 area 0.0.0.0
   max-lsa 12000
```

The following items should be variables in your Jinja2 template:

```
ospf_process_id
ospf_priority
ospf_active_interfaces (i.e. the non-passive interfaces)
ospf_area0_networks (the three networks that are specified as belonging
to area0)
```

Your template should be in an external file.

Your template should also use a conditional to control whether this is output or not:

```
interface vlan 1
   ip ospf priority 100
```

If the 'ospf_priority variable is defined', then include that section. If 'ospf_priority' is not defined then only include the 'router ospf 10' section.

3a. Create a YAML file that defines a list of interface names. Use the expanded form of YAML.

Use a Python script to read in this YAML list and print it to the screen.

The output of your Python script should be:

```
['Ethernet1', 'Ethernet2', 'Ethernet3', 'Ethernet4', 'Ethernet5',
'Ethernet6', 'Ethernet7', 'Management1', 'Vlan1']
```

3b. Expand the data structure defined earlier in exercise3a. This time you should have an 'interfaces' key that refers to a dictionary.

Use Python to read in this YAML data structure and print this to the screen.

The output of your Python script should look as follows (in other words, your YAML data structure should yield the following when read by Python). You YAML data structure should be written in expanded YAML format.

```
{'interfaces': {
    'Ethernet1': {'mode': 'access', 'vlan': 10},
    'Ethernet2': {'mode': 'access', 'vlan': 20},
    'Ethernet3': {'mode': 'trunk',
                  'native_vlan': 1,
                  'trunk_vlans': 'all'}
    }
}
```

4. Take the YAML file and corresponding data structure that you defined in exercise3b:

```
{'interfaces': {
    'Ethernet1': {'mode': 'access', 'vlan': 10},
    'Ethernet2': {'mode': 'access', 'vlan': 20},
    'Ethernet3': {'mode': 'trunk',
                  'native_vlan': 1,
                  'trunk_vlans': 'all'}
    }
}
```

From this YAML data input source, use Jinja templating to generate the following configuration output:

```
interface Ethernet1
  switchport mode access
  switchport access vlan 10
interface Ethernet2
  switchport mode access
  switchport access vlan 20
interface Ethernet3
  switchport mode trunk
  switchport trunk native vlan 1
  switchport trunk allowed vlan all
```

The following should all be variables in your Jinja template (the names may be different than below, but they should be variabilized and not be hard-coded in your template).

```
interface_name
switchport_mode
access_vlan
native_vlan
trunk_vlans
```

All your Jinja2 variables should be retrieved from your YAML file.

This exercise might be challenging.

CLASS OUTLINE

```
1. Jinja2 Basics (VIDEO1)
   A. What is Jinja2?   [00:11]
```

```
      1. Jinja2 is also used in Ansible   [2:11]
   B. Jinja2 basic use   [2:22]
      1. Cisco configuration example   [2:33]
      2. Variables embedded using    [3:57]
      3. Expanding to more variables   [5:17]

2. Jinja2 For-loops and Conditionals (VIDEO2)
   A. Decoupling the template from the code   [00:08]
   B. Jinja2 and for-loops   [2:10]
      1. Embedding a for-loop in the template   [2:40]
      2. Controlling whitespace   [5:16]
   C. Jinja2 and conditionals   [6:35]
      1. How much logic in a Jinja2 template   [7:47]

3. Jinja2 and a CSV File (VIDEO3)
   A. Reading data from an external CSV source   [00:06]
      1. Using CSV DictReader   [1:12]
      2. Adding additional CSV rows   [3:17]

4. Jinja2 dictionaries and nested for-loops (VIDEO4)
   A. Looping over Jinja2 dictionaries using .items()   [1:22]
   B. Looping over Jinja2 lists   [3:24]
   C. Jinja2 nested for-loops   [4:31]
   D. Jinja2 nested conditionals   [7:58]
   E. Jinja2 dictionaries accessing keys   [9:36]

5. YAML Basics (VIDEO5)
   A. Introduction to YAML   [00:04]
      1. Why do we care about YAML?   [00:18]
      2. What is serialization?   [00:38]
      3. YAML is used in Ansible and Salt   [2:27]
   B. Basic YAML
      1. Creating a list   [2:57]
      2. Creating a dictionary   [6:20]

6. YAML Part2 (VIDEO6)
   A. YAML basic data types   [00:12]
      1. Strings   [00:15]
      2. Quoting strings   [00:43]
      3. Booleans   [1:22]
      4. Integers   [3:16]
   B. YAML and complex data structures   [3:43]
      1. Dictionaries containing dictionaries   [3:52]
      2. Dictionaries containing lists   [6:21]
   C. YAML expanded form and compressed form   [8:03]
      1. YAML is a superset of JSON   [8:17]

7. Writing YAML using Python (VIDEO7)
```

```
8. JSON (VIDEO8)
   A. What is JSON?   [00:02]
   B. Comparing Python to JSON   [2:09]
   C. Writing JSON out to a file   [1:44]
   D. Reading JSON in Python   [3:39]
   E. Talking about APIs and JSON
      1. Why do we care about JSON?   [4:29]
      2. Some APIs that use JSON (Arista eAPI, NX-API)   [4:35]
      3. REST APIs frequently use JSON   [4:49]

9. Managing Data Structures (VIDEO9)
   A. Complex data structures   [00:35]
      1. Dictionaries that contain lists or dictionaries
      2. Lists that contains dictionaries or lists
   B. Handling complex data structures   [1:19]
      1. Peeling back a layer at a time   [2:03]
         a. Dictionaries   [2:17]
         b. Lists   [3:34]
   C. Creating complex data structures in Python   [4:51]
```

Kirk Byers
https://pynet.twb-tech.com