1.2- YAML Structure

YAML was said to mean Yet Another Markup Language, referencing its purpose as a markup language with yet another construct, but it was then repurposed as YAML Ain't Markup Language, a recursive acronym, to distinguish its purpose as data-oriented, rather than document markup. YAML is adapted globally by multiple technologies to write codes. There are other available Markup Languages like JSON, XML, etc., but they are little complex in nature and need investment of time to understand. On the other hand, YAML is human readable, very simple, easy to learn Markup Language. It has been adapted by product development organizations very quickly. YAML is case sensitive like Unix Operating Systems. This reference is taken from our earlier book "**Automation with Ansible – By Yogesh Raheja"** as YAML is the default language used by Ansible for writing Ansible codes called Playbooks. In Puppet Hiera files are using YAML structure, so it's worth to mention the ground rules for creating YAML files. This reference will help you to understand the 3 most important considerations while writing YAML codes.
Wiki reference for YAML "https://en.wikipedia.org/wiki/YAML "

There are only three very simple rules to remember while writing YAML for Ansible Playbooks.

1. Rule 1 – Indentation: YAML uses a fixed indentation scheme to represent relationships between data layers. YAML requires that the indentation for each level consists of exactly two spaces (best practices); otherwise, we can use any equal number of spaces. Do not use tabs.

2. Rule 2 – Colons: Dictionary keys are represented in YAML as strings terminated by a trailing colon. Values are represented by either a string following the colon, separated by a space, example: Key: Value. Alternatively, a value can be associated with a key through indentation, i.e., key followed by a colon, then next line with two spaces and a value;

example:

Key:

Value

Note: In Python, the above maps to: {'key': 'value'}

3. Rule 3 – Dashes: To represent a list of items, always use a single dash followed by a space. Several elements are a part of the same list, as a function of their having the same level of indentation. Example:

– list_value_one

– list_value_two

– list_value_three

Sometimes you come across different terms like key/value pairs, hash and/or dictionary. In YAML, these three terms are same; so do not get confused while following other documentations.

Note: Lists can be the value of a key−value pair. This is quite common in YAML coees. You will observe similar pattern while creating YAML codes.

my_dictionary:

– list_value_one

– list_value_two

– list_value_three

Like any other language, YAML also provides the flexibility to add comments by using # and then comment. YAML also provides one additional feature of indicating start --- (three dashes) and end… (three dots) of a YAML file. This is an optional feature, but you generally see the start --- (three dashes) format in almost all available YAML fils. In fact, it is a good practice to follow.

Let us write a simple YAML file and use all of the mentioned rules and best practices.

```
[root@puppetmaster ~]# cat /var/tmp/test.yml
---
# Yogesh Raheja and Dennis McCarthy are writing their first test YAML file


# Rule number three “dashes”
Chapters:
  - one
  - two
  - three
  - four


# Rule number two “colons”
firstname: yogesh
secondname: raheja


# Rule number first “indentation”
name:
  yogeshraheja


# Yogesh Raheja and Dennis McCarthy are ending their first test YAML file
...
[root@puppetmaster ~]#
```

There are multiple methods to check the valid syntax of a YAML file, which is used to ensure the correct YAML file. So far we have just understood the YAML structure. The method below demonstrates valid YAML files.

Option 1: Paste in your YAML code online at http://www.yamllint.com/; click Go and verify the validation of your YAML file.

Option 2: Use Python to check and validate your YAML code. Run below command to check YAML syntax. If there are no errors in the code, Python will print the code contents in JSON format.

```
[root@puppetmaster ~]#  python -c 'import yaml, sys; print yaml.load(sys.stdin)' < /var/tmp/test.yml
{'Chapters': ['one', 'two', 'three', 'four'], 'name': 'yogeshraheja', 'firstname': 'yogesh', 'secondname': 'raheja'}
[root@puppetmaster ~]#
```