**YAML Syntax**

We use YAML because it is easier for humans to read and write than other common data formats like XML or JSON. Further, there are libraries available in most programming languages for working with YAML.

**YAML Basics**

For Ansible, nearly every YAML file starts with a list. Each item in the list is a list of key/value pairs, commonly called a “hash” or a “dictionary”. So, we need to know how to write lists and dictionaries in YAML.

There’s another small quirk to YAML. All YAML files (regardless of their association with Ansible or not) can optionally begin with --- and end with .... This is part of the YAML format and indicates the start and end of a document.

All members of a list are lines beginning at the same indentation level starting with a "- " (a dash and a space):

**---**

*# A list of tasty fruits*

fruits:

- Apple

- Orange

- Strawberry

- Mango

**...**

A dictionary is represented in a simple key: value form (the colon must be followed by a space):

*# An employee record*

martin:

name: Martin D'vloper

job: Developer

skill: Elite

More complicated data structures are possible, such as lists of dictionaries, dictionaries whose values are lists or a mix of both:

*# Employee records*

- martin:

name: Martin D'vloper

job: Developer

skills:

- python

- perl

- pascal

- tabitha:

name: Tabitha Bitumen

job: Developer

skills:

- lisp

- fortran

- erlang

Dictionaries and lists can also be represented in an abbreviated form if you really want to:

**---**

martin: {name: Martin D'vloper, job: Developer, skill: Elite}

fruits: ['Apple', 'Orange', 'Strawberry', 'Mango']

These are called “Flow collections”.

Ansible doesn’t really use these too much, but you can also specify a boolean value (true/false) in several forms:

create\_key: yes

needs\_agent: no

knows\_oop: True

likes\_emacs: TRUE

uses\_cvs: false

Values can span multiple lines using | or >. Spanning multiple lines using a “Literal Block Scalar” | will include the newlines and any trailing spaces. Using a “Folded Block Scalar” > will fold newlines to spaces; it’s used to make what would otherwise be a very long line easier to read and edit. In either case the indentation will be ignored. Examples are:

include\_newlines: |

exactly as you see

will appear these three

lines of poetry

fold\_newlines: >

this is really a

single line of text

despite appearances

While in the above > example all newlines are folded into spaces, there are two ways to enforce a newline to be kept:

fold\_some\_newlines: >

a

b

c

d

e

f

same\_as: "a b\nc d\n e\nf\n"

Let’s combine what we learned so far in an arbitrary YAML example. This really has nothing to do with Ansible, but will give you a feel for the format:

**---**

*# An employee record*

name: Martin D'vloper

job: Developer

skill: Elite

employed: True

foods:

- Apple

- Orange

- Strawberry

- Mango

languages:

perl: Elite

python: Elite

pascal: Lame

education: |

4 GCSEs

3 A-Levels

BSc in the Internet of Things

That’s all you really need to know about YAML to start writing *Ansible* playbooks.

**Gotchas**

While you can put just about anything into an unquoted scalar, there are some exceptions. A colon followed by a space (or newline) : `` is an indicator for a mapping. A space followed by the pound sign `` # starts a comment.

Because of this, the following is going to result in a YAML syntax error:

foo: somebody said I should put a colon here: so I did

windows\_drive: c:

...but this will work:

windows\_path: c:\windows

You will want to quote hash values using colons followed by a space or the end of the line:

foo: 'somebody said I should put a colon here: so I did'

windows\_drive: 'c:'

...and then the colon will be preserved.

Alternatively, you can use double quotes:

foo: "somebody said I should put a colon here: so I did"

windows\_drive: "c:"

The difference between single quotes and double quotes is that in double quotes you can use escapes:

foo: "a \t TAB and a \n NEWLINE"

The list of allowed escapes can be found in the YAML Specification under “Escape Sequences” (YAML 1.1) or “Escape Characters” (YAML 1.2).

The following is invalid YAML:

foo: "an escaped \' single quote"

Further, Ansible uses “{{ var }}” for variables. If a value after a colon starts with a “{”, YAML will think it is a dictionary, so you must quote it, like so:

foo: "{{ variable }}"

If your value starts with a quote the entire value must be quoted, not just part of it. Here are some additional examples of how to properly quote things:

foo: "{{ variable }}/additional/string/literal"

foo2: "{{ variable }}\\backslashes\\are\\also\\special\\characters"

foo3: "even if it's just a string literal it must all be quoted"

Not valid:

foo: "E:\\path\\"rest\\of\\path

In addition to ' and " there are a number of characters that are special (or reserved) and cannot be used as the first character of an unquoted scalar: [] {} > | \* & ! % # ` @ ,.

You should also be aware of ? : -. In YAML, they are allowed at the beginning of a string if a non-space character follows, but YAML processor implementations differ, so it’s better to use quotes.

In Flow Collections, the rules are a bit more strict:

a scalar in block mapping: this } is [ all , valid

flow mapping: { key: "you { should [ use , quotes here" }

Boolean conversion is helpful, but this can be a problem when you want a literal *yes* or other boolean values as a string. In these cases just use quotes:

non\_boolean: "yes"

other\_string: "False"

YAML converts certain strings into floating-point values, such as the string *1.0*. If you need to specify a version number (in a requirements.yml file, for example), you will need to quote the value if it looks like a floating-point value:

version: "1.0"