CloudBees.

YAML Tutorial: Everything You Need to Get Started in Minutes

Written by: erik

December 11, 2018

7 min read

YAML Ain't Markup Language (YAML) is a serialization language that has steadily increased in popularity over the last few years. It's often used as a format for configuration files, but its object serialization abilities make it a viable replacement for languages like JSON. This YAML tutorial will demonstrate the language syntax with a guide and some simple coding examples in Python. YAML has broad language support and maps easily into native data structures. It's also easy to for humans to read, which is why it's a good choice for configuration. The YAML acronym was shorthand for Yet Another Markup Language. But the maintainers renamed it to YAML Ain't Markup Language to place more emphasis on its data-oriented features.

YAML Tutorial Quick Start: A Simple File

Let's take a look at a YAML file for a brief overview.

```
---
 doe: "a deer, a female deer"
 ray: "a drop of golden sun"
 pi: 3.14159
 xmas: true
 french-hens: 3
 calling-birds:
   - huey
   - dewey
   - louie
   - fred
 xmas-fifth-day:
   calling-birds: four
   french-hens: 3
   golden-rings: 5
   partridges:
     count: 1
     location: "a pear tree"
   turtle-doves: two
```

The file starts with three dashes. These dashes indicate the start of a new YAML document. YAML supports multiple documents, and compliant parsers will recognize each set of dashes as the beginning of a new one. Next, we see the construct that makes up most of a typical YAML document: a key-value pair. **Doe** is a key that points to a string value: **a deer, a female deer.** YAML supports more than just string values. The file starts with six key-value pairs. They have four different data types. **Doe** and **ray** are strings. **Pi** is a floating-point number. **Xmas** is a boolean. **French-hens** is an integer. You can enclose

strings in single or double-quotes or no quotes at all. YAML recognizes unquoted numerals as integers or floating point. The seventh item is an array. **Calling-birds** has four elements, each denoted by an opening dash. I indented the elements in **calling-birds** with two spaces. Indentation is how YAML denotes nesting. The number of spaces can vary from file to file, but tabs are not allowed. We'll look at how indentation works below. Finally, we see **xmas-fifth-day**, which has five more elements inside it, each of them indented. We can view **xmas-fifth-day** as a dictionary that contains two string, two integers, and another dictionary. YAML supports nesting of key-values, and mixing types. Before we take a deeper dive, let's look at how this document looks in JSON. I'll throw it in this handy JSON to YAML converter.

```
{
  "doe": "a deer, a female deer",
  "ray": "a drop of golden sun",
  "pi": 3.14159,
  "xmas": true,
  "french-hens": 3,
  "calling-birds": [
     "huey",
     "dewey",
     "louie",
     "fred"
  ],
  "xmas-fifth-day": {
  "calling-birds": "four",
  "french-hens": 3,
  "golden-rings": 5,
  "partridges": {
    "count": 1,
    "location": "a pear tree"
  },
  "turtle-doves": "two"
  }
}
```

JSON and YAML have similar capabilities, and you can convert most documents between the formats.

Outline Indentation and Whitespace

Whitespace is part of YAML's formatting. Unless otherwise indicated, newlines indicate the end of a field. You structure a YAML document with indentation. The indentation level can be one or more spaces. The specification forbids tabs because tools treat them differently. Consider this document. The items inside **stuff** are indented with two spaces.

```
foo: bar
     pleh: help
     stuff:
       foo: bar
       bar: foo
```

Let's take a look at how a simple python script views this document. We'll save it as a file named **foo.yaml**. The PyYAML package will map a YAML file stream into a dictionary. We'll iterate through the outermost set of keys and values and print the key and the string representation of each value. You can find a processor for your favorite platform here.

```
import yaml

if __name__ == '__main__':

    stream = open("foo.yaml", 'r')
    dictionary = yaml.load(stream)
    for key, value in dictionary.items():
        print (key + " : " + str(value))
```

The output is:

```
foo : bar
pleh : help
stuff : {'foo': 'bar', 'bar': 'foo'}
```

When we tell python to print a dictionary as a string, it uses the inline syntax we'll see below. We can see from the output that our document is a python dictionary with two strings and another dictionary nested inside it. YAML's simple nesting gives us the power to build sophisticated objects. But that's only the beginning.

Comments

Comments begin with a pound sign. They can appear after a document value or take up an entire line.