

YAML is a straightforward, domain-specific language that can represent just about anything hierarchically. The user explicitly defines the structure of an object, process or workflow in terms easily understood by even novice coders. The time it takes to learn YAML is paid back in capabilities for configuration management, automation and orchestration.

YAML's structure is case-sensitive and similar to a markup language, but it is not one -- its name stands for [YAML Ain't Markup Language](#), after all. YAML has only a few different types of objects, called *constructs*: scalars, sequences or lists, and dictionaries or mappings. Admins can create complex structures through a system of embedded constructs.

While YAML is a mechanism to [describe server configurations](#), container deployments and other IT operations, that's not all it can do. You can express nearly everything there is to know about someone with a simple YAML file. A person has many different attributes, such as a first name, last name, occupation, home and more. To humanize coding in a way that will help you learn YAML, this article covers how I created a file called AdamBertram.yaml.

Scalars

The most straightforward YAML construct is a [scalar](#), which is a singular string or a number. Scalars don't contain multiple elements. A scalar is represented by a type followed by a colon and then the value. As you'll see below, these scalars are essentially mappings hiding under another name. Scalars represented this way are not typically used.

```
---  
integer: 2  
boolean: true  
string: "26"
```

©2018 TECHTARGET. ALL RIGHTS RESERVED. TechTarget

Strings are enclosed in quotes. Quotes are also required for values that contain spaces, colons or other special characters.

To represent a plain-text file as YAML, always include three dashes at the top of the file, as displayed in each snippet below. This indicates to YAML-reading utilities that the following text is code.

Mappings

Because scalar values aren't labeled, *mappings* are more common and integral when you learn YAML. Every person has a first and last name -- unless you're [Prince](#) -- so YAML represents each of these attributes with a key for the label and value, as shown below.

```
---
first_name: Adam
last_name: Bertram
hair_color: Brown
married: true
dog_count: 2
```

Lists and sequences

Lists, or *sequences*, are akin to arrays in computer programming that contain a label to represent one or more elements inside of a list. In this example, I have two dogs, so I'll use a list in YAML to describe their names.

```
---
first_name: Adam
last_name: Bertram
hair_color: Brown
married: true
dog_count: 2
dogs:
  - Elliott
  - Brody
```

Because the file is called AdamBertram.yaml, the structure of the file makes it clear that Adam Bertram has brown hair and is married with two dogs named Elliott and Brody.

Dictionaries

In YAML, a *dictionary* is a list that contains [key-value pairs](#), or mappings, instead of single elements. Some users refer to dictionaries as *hashtables*.


A dictionary enables the code writer to include additional attributes about the dogs listed in AdamBertram.yaml. Instead of just referencing a string as their first names, I can introduce a dictionary of mappings to describe each dog better. For example, all dogs have a breed and a color.

A single dog is represented below:

```
dog:
  name: Elliott
  breed: Shih-Tzu
  color: black/white
```

But I have two dogs, and YAML is hierarchical. So, I need a parent list called *dogs*, to which I add each dog and its attributes.


```
dogs:
  dog1:
    name: Elliott
    breed: Shih-Tzu
    color: black/white
  dog2:
    name: Brody
    breed: Shih-Tzu
    color: black/white
```

©2009 TECHTARGET. ALL RIGHTS RESERVED. 

YAML syntax can get complicated: Nearly any construct can be embedded within another.

Here's what the AdamBertram.yaml file looks like now, with the *dogs* parent list:

```
---
first_name: Adam
last_name: Bertram
hair_color: Brown
married: true
dog_count: 2
dogs:
  dog1:
    name: Elliott
    breed: Shih-Tzu
    color: black/white
  dog2:
    name: Brody
    breed: Shih-Tzu
    color: black/white
```

©2009 TECHTARGET. ALL RIGHTS RESERVED. 

Following this format, I can improve upon this file and add mappings, lists or dictionaries anywhere necessary by adding more descriptions.