YAML

YAML means **yet another markup language**.

YAML has gained a lot of popularity over the last few years as it became part of crucial DevOps tools, technologies and processes such as Ansible, Kubernetes, CI/CD pipelines and so on.

YAML was created specifically for common use cases such as:

- Configuration files
- Log files
- Inter-process messaging
- Cross-language data sharing
- Object persistence
- Complex data structures

Why you should use YAML:

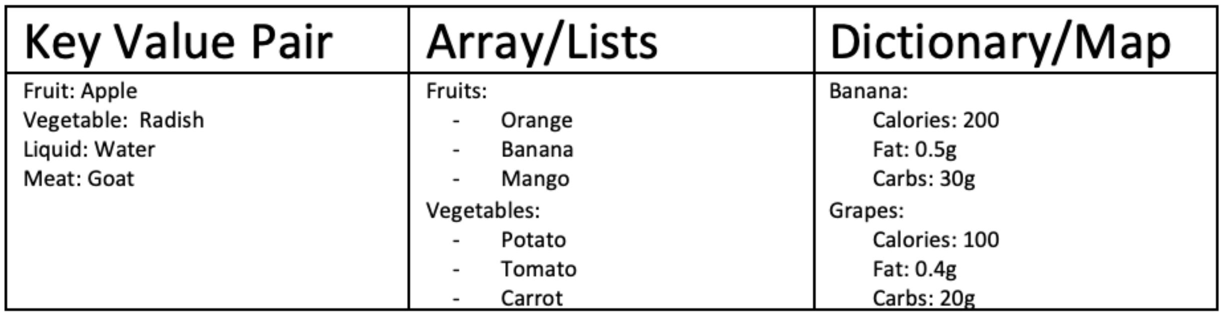
There are a few advantages to using YAML files:

1. They are easily readable by humans. YAML files are expressive and extensible.
2. They are easy to implement and use.
3. They are easily portable between programming languages.
4. They match the native data structures of agile languages.
5. YAML files have a consistent model to support generic tools.
6. They support one-pass processing.
7. They are convenient to use, so you no longer need to add all of your parameters to the command line.
8. You can perform maintenance. YAML files can be added to the source control to track the changes.
9. They are flexible. You can create much complex structures using YAML than you can use on command line

The structure of a YAML file:

The following are the building blocks of a YAML file:

1. Key Value Pair — The basic type of entry in a YAML file is of a key value pair. After the Key and colon there is a space and then the value.
2. Arrays/Lists — Lists would have a number of items listed under the name of the list. The elements of the list would start with a -. There can be a n of lists, however the indentation of various elements of the array matters a lot.
3. Dictionary/Map — A more complex type of YAML file would be a Dictionary and Map.



Key Value Pair	Array/Lists	Dictionary/Map
Fruit: Apple	Fruits:	Banana:
Vegetable: Radish	- Orange	Calories: 200
Liquid: Water	- Banana	Fat: 0.5g
Meat: Goat	- Mango	Carbs: 30g
	Vegetables:	Grapes:
	- Potato	Calories: 100
	- Tomato	Fat: 0.4g
	- Carrot	Carbs: 20g

Comments in YAML:

Comments in YAML can be defined by placing a hash in front of an item '#'. Comments can be made at the start of a line of anywhere in the line.

Here's an example.

```
#Configuring the port
  ports:
  - port: 8080        #service port
    targetPort: 8080   #Pod Port
    nodePort: 30012  #Node Port from the range - 30000-32767
```

Datatypes:

YAML has three types of data types:

1. Scalar
2. List
3. Dictionary

Scalar data type:

Scalar is a simple data type. In YAML, scalar means a simple value for a key. The value of the scalar can be integer, float, Boolean, and string. Scalar data types are classified into two data types:

- Numeric Data type
- String

Numeric Data type:

There are three types of numeric data type:

- Integer
- Floating point numbers
- Booleans

An **Integer data type** can be decimal, octal, or hexadecimal.

Boolean:

Boolean value can be True/False or Yes/No or On/Off.

str1: **null**
b1:True

We can write a multi-line string in a single line using > symbol. In this, a newline character(\n) will be ignored.

```
str: >
    this is a multi-line string it
    spans more than one
    line
```

Collections:

YAML files are made up of three core components: Mappings, lists, and scalars.

Mappings are simple key-value pairs, like ip: 10.0.4.0.

Lists work like any plain-text bulleted list, with each item on a new line and starting with a dash.

Finally, a *scalar* is something that is a string, boolean, or number; an item on a list is its own scalar, while both the key and value of a key-value pair are individual scalars.

Mappings and lists can also be combined.

Lists:

We can define the list in a single line as follows:

```
---
items: [6, 7, 8, 9, 10]
name: [six, seven, eight, nine, ten]
```

This style is known as block style. We can put the above list in multiple lines as follows:

```
---
items:
  - 6
  - 7
  - 8
name:
  - "six"
  - "seven"
  - "eight"
  - "nine"
```

This style is known as flow style. A list that contains complex objects needs multiple lines.

Dictionaries:

If we want to write a complex YAML file which holds the complex data structure, we will use dictionaries. It is a collection of key: value pairs and each of the key: value pairs can be nested with a lot of options.

Example:

```
---
student1: "john"
hobbies:
  - music
  - reading
  - dancing
```

In the above example, student is the first key, and john is the value. Hobbies are the second key, but it is nested, which means it contains a list of values. The value of the key can again be a key: value pair, which we will see in the next example.

Styles:

Yaml can be written in 2 styles.

There are two types of styles in which we can write the YAML.

- Block styles
- Flow styles

Block style Example:

```
host: comcast34
datacenter:
  location: canada
  cab: 15
animals:
  - dog
  - cat
  - mouse
```

Flow Style:

Flow style is an extension of JSON. It is used to allow YAML and JSON to work together. Flow style is less human-readable, but sometimes they are better for the computer that processing our YAML. Flow style is used to fold the long line of content.

Example:

```
host: comcast42
datacenter: {location: canada , cab: 15}
animals: [dog , cat , mouse]
```