YAML File

A YAML file is a plain text file with the .yaml or .yml extension.

YAML (YAML Ain't Markup Language) is a human-readable data serialization format commonly used for configuration files (e.g., in Kubernetes, Docker, CI/CD pipelines).

It's:

- Easier to read and write than JSON or XML.
- Used in many DevOps tools (Kubernetes, Ansible, Docker Compose, etc.).
- Space-sensitive, so indentation is critical.
-

Comparing YAML with XML & JSON

take the same configuration example and represent it in YAML, JSON, and XML.

A user profile containing name, age, and a list of skills.

1. YAML

```yaml
name: joshi
age: 29
skills:
  - Python
  - Kubernetes
  - Docker
```

2. JSON

```json
{
  "name": "joshi",
  "age": 29,
  "skills": [
    "Python",
    "Kubernetes",
    "Docker"
  ]
}
```

3. XML

```xml
<user>
  <name>joshi</name>
  <age>29</age>
```

```
  <skills>
    <skill>Python</skill>
    <skill>Kubernetes</skill>
    <skill>Docker</skill>
  </skills>
</user>
```

Feature	YAML	JSON	XML
Syntax	Indentation-based	Curly braces	Angle brackets
Readability	High	Moderate	Low
Used for	Config files	APIs	Data transfer
Comments	Yes (#)	No	No

Key Rules for Writing YAML Files

1. Use Spaces, Not Tabs

- YAML does not support tabs, only spaces for indentation.
- Standard indentation = 2 spaces (some projects may use 4, but consistency is key).

```
#Correct way
name: joshi
age: 30

#Incorrect way
name: joshi
  age: 30  #Indentation mismatch
```

2. Key-Value pairs

- All data is stored as key-value pairs. A colon : separates key and value, followed by a space.

```
name: joshi
age: 29
city: Bidar
```

3. Representing Lists/Arrays

- Use a dash - followed by a space for each item in the list.

```
# fruits is list of string values
```

```yaml
fruits:
  - apple
  - banana
  - orange
  - grapes
```

4. Representing Dictionaries (Maps/Objects)

- Dictionaries are collections of key-value pairs.

```yaml
#person is a key mapping to nested object

person:
  name: joshi
  age: 29
  place: bidar
```

5. Nested Structures (Indentation Matters)

- Dictionary inside a dictionary

```yaml
# employee is top level key, contain nested person dictionary

employee:
  person:
    name: joshi
    age: 29
    place: Bidar
```

- List inside a dictionary:

```yaml
person:
  name: joshi
  age: 29
  skills: [python, js]
  places: [Bidar, Bagalkote, Bangalore]

#In above example the use of square brackets ([ ]) for lists is perfectly valid in
YAML.

employee:
  name: joshi
  skills:
    - python
    - js
  places:
    - Bidar
```

```
    - Bagalkote
    - Bengaluru
#This is more readable, especially for longer or nested items.
```

- List of dictionaries:

```
services:
  - name: hello
    port: 8000
  - name: hi
    port: 8001
```

6. Multiline Strings Use | for preserving line breaks, and > for folding lines into one.

- | Literal Block (preserves new lines):

```
description: |
  This is a service
  that exposes the web app
  on port 80.
```

- > Folded Block (new line became space):

```
description: >
  This is a service
  that exposes the web app
  on port 80.
```

7. Comments in YAML Use # to add comments. YAML ignores anything after #.

```
# This is a comment
name: Alex  # Inline comment
```

Summary

Concept	Description
YAML	Human-readable data format
Indentation	Defines structure (no tabs!)
Key-Value Pairs	Basic data structure (`key: value`)
List	Ordered collection using `-`

Concept	Description
Dictionary	Unordered key-value pairs
Nested Data	Indent to show nesting
Comments	Start with `#`

Common yaml mistakes to avoise

Mistake	Description	Fix
Using tabs	YAML doesn't support tabs	Use **spaces** only
Inconsistent indentation	Indentation levels must match	Use consistent **2-space** indentation
Missing space after `:`	`key:value` is invalid	Use `key: value`
Trailing comma	YAML doesn’t use commas like JSON	Don’t use commas in lists or maps

YAML data structures overview

Structure	Description	Type
Scalars	Represents single, indivisible values such as strings, numbers, booleans, or null. Examples: `"John"`, `42`, `true`, `null`. Scalars are the simplest data type in YAML.	Primitive
List (Array)	An **ordered** collection of items, where order matters. Items are listed with a dash `-` and are typically homogeneous but can contain mixed types. Example: a list of fruits or ports.	Ordered
Dictionary (Map)	An **unordered** collection of key-value pairs. Each key maps to a value which can be any data type (scalar, list, dictionary). Used to represent objects or entities with attributes. Example: a person with name, age, and city.	Unordered
Nested structures	Complex data formed by combining lists and dictionaries inside each other. Examples include dictionaries containing lists, lists of dictionaries, or multiple levels of nesting, allowing modeling of complex configurations or data hierarchies.	Composite

```yaml
apiVersion: v1           # Scalar
kind: Pod                # Scalar
metadata:                # Dictionary
  name: nginx-pod        # Scalar
  labels:                # Dictionary
    app: nginx           # Scalar
spec:                    # Dictionary
  containers:            # List
    - name: nginx        # Scalar (in dict)
      image: nginx:latest # Scalar
      ports:             # List
```

```
        - containerPort: 80  # Scalar in dict in list
```

---

yaml used to represent config data data structure formats : XML, JSON, yaml (how to write all using same example) How to write YAML file (key rules)

- how to represent array/list in yaml
- how to represent dictionary
- Ensure space is correct

Dictionary is unordered collection List is ordered collection

represents a comment

list example, dictionary example, dict in dict