📘 Comprehensive YAML Guide for DevOps Engineers

A complete beginner-friendly and practical guide to understanding and mastering YAML.

📚 Table of Contents

---

1. YAML Fundamentals

What is YAML?

- YAML stands for **"YAML Ain't Markup Language"**.
- It's a **human-readable** data format used mainly for **configuration files** in DevOps, Kubernetes, CI/CD pipelines, and more.

🔹 **Basic Syntax:**

```
# A simple key-value pair
name: John

# Hierarchical structure using indentation
person:
  name: Alice
  age: 25
```

🔹 **Key Features:**

- **Indentation:** Use **spaces only** (not tabs). Standard is 2 spaces per level.
- **Comments:** Start with #
- **Multiple Documents:** Separate with ---

---

2. Basic Data Types

🔹 **Strings:**

```
plain: Hello World
quoted: "Special characters like : or &"
multiline: |
  This is a multi-line
  string that keeps
  line breaks.
```

🔹 **Numbers:**

```
integer: 10
float: 3.1415
scientific: 1.5e+10
```

- **Booleans:**

```
true_values: [true, True, yes, on]
false_values: [false, False, no, off]
```

- **Null Values:**

```
null_value: null
another_null: ~
```

---

3. Collections in YAML

- **Lists (Arrays):**

```
fruits:
  - Apple
  - Banana
  - Mango

# Inline list
colors: [red, green, blue]
```

- **Dictionaries (Maps):**

```
employee:
  name: John Doe
  age: 30
  department: IT
```

- **Nested Structures:**

```
company:
  name: TechCorp
  departments:
    devops:
      head: Alice
      size: 10
    qa:
      head: Bob
      size: 5
```

---

4. Intermediate YAML Concepts

- **Anchors (&) and Aliases (*)**

```yaml
defaults: &defaults
  host: localhost
  port: 5432

dev:
  <<: *defaults
  db: dev_db

prod:
  <<: *defaults
  db: prod_db
```

- **Complex Nesting:**

```yaml
project:
  name: InfraProject
  team:
    - name: Alice
      skills: [Terraform, AWS]
    - name: Bob
      skills: [Docker, Kubernetes]
```

---

5. Advanced YAML Techniques

- **Merge Keys:**

```yaml
base: &base
  logging: true
  retries: 3

service:
  <<: *base
  name: myservice
```

- **Explicit Data Typing:**

```yaml
version: !!str 1.0
enabled: !!bool "yes"
created: !!timestamp 2024-01-01T00:00:00Z
```

- **Flow vs Block Styles:**

```yaml
# Block style (more readable)
items:
  - One
  - Two

# Flow style (compact)
items: [One, Two]
```

---

6. YAML for Kubernetes

- **Deployment:**

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 2
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:1.14.2
          ports:
            - containerPort: 80
```

- **Service:**

```yaml
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
  type: LoadBalancer
```

---

7. YAML for CI/CD Pipelines

- **GitLab CI:**

```yaml
stages:
  - build
  - test
  - deploy

build_job:
  stage: build
  script:
    - echo "Building..."
```

- **GitHub Actions:**

```yaml
name: CI Pipeline

on: [push]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v2
```

```
      - name: Run tests
        run: echo "Running tests..."
```

---

8. YAML for Docker Compose *(New)*

```yaml
version: "3.8"
services:
  web:
    image: nginx
    ports:
      - "80:80"
  app:
    build: .
    depends_on:
      - db
  db:
    image: postgres
    environment:
      POSTGRES_PASSWORD: example
```

---

9. YAML Best Practices

✅ Use **2 spaces** (no tabs)

✅ Group related settings

✅ Comment your YAML files for clarity

✅ Break large files into smaller reusable ones

✅ Use **Helm templates** for Kubernetes when configs grow complex

✅ Always use version control (like Git)

---

10. Common Pitfalls

❌ Mixing **tabs and spaces** (always use spaces)

❌ Forgetting **quotes** for special characters (like : or &)

❌ **Overusing anchors and aliases** (can hurt readability)

❌ Repeating keys silently overrides values

❌ Flat, **unstructured large files** reduce clarity

---

11. Validation and Tools

🧪 **Linting and Validation:**

```
# Lint YAML file
yamllint file.yaml

# Validate Kubernetes manifest
kubeval deployment.yaml

# Convert JSON to YAML
python -c 'import sys, yaml, json;
print(yaml.dump(json.loads(sys.stdin.read())))' < file.json >
file.yaml
```

🔧 **Online Validators:**

- yamllint.com

🛠️ **IDE Support:**

- **VS Code** (with Red Hat YAML extension)
- **IntelliJ IDEA**
- **Sublime Text** (YAML plugins)

---

12. Extras and Pro Tips

✅ **Multiline Folded Block:**

```
text: >
  This is a long sentence
  that will be folded into
  a single line.
```

✅ **Environment Variables in Templates:**

```
env:
```

```
  - name: APP_ENV
    valueFrom:
      configMapKeyRef:
        name: app-config
        key: env
```

✅ **JSON Compatibility:**

- YAML is a **superset of JSON**. Valid JSON is valid YAML.

---

13. Real-World YAML Examples *(New)*

🔹 **Helm values.yaml Example:**

```
replicaCount: 2
image:
  repository: myapp
  tag: latest
  pullPolicy: IfNotPresent

service:
  type: ClusterIP
  port: 80
```

🔹 **Azure DevOps Pipeline:**

```
trigger:
  branches:
    include:
      - main

jobs:
- job: Build
  pool:
    vmImage: 'ubuntu-latest'
  steps:
    - script: echo "Building the project"
```

✅ **Tip for Freshers**: Always practice by editing and creating your own YAML files. Start with small ones like .gitlab-ci.yml or docker-compose.yml, then work your way into Kubernetes and Helm templates.