📄 YAML Full Syntax Guide

> **YAML = YAML Ain't Markup Language**
> Used widely in: **Kubernetes**, **GitHub Actions**, **Docker Compose**, **Ansible**, and more.

🧠 Basics of YAML

- 🧱 **Indentation**: Always use `2 spaces` – never use tabs ❌
- 💡 YAML is case-sensitive
- 📄 File extensions: `.yaml` or `.yml`

🏷️ Basic Key-Value Pairs

```yaml
name: Darshan Vasani
role: DevOps Engineer
active: true
age: 23
```

🔢 Data Types

```yaml
string_value: "Hello YAML"
int_value: 25
float_value: 12.34
boolean_true: true
boolean_false: false
null_value: null    # or ~ or ""
```

📋 Comments

```yaml
# This is a comment in YAML
```

📃 Strings

```yaml
plain_string: Hello
single_quoted: 'Don''t forget me'    # escape single quote by doubling it
```

```yaml
double_quoted: "This is a string with \n newline"

multiline_string: |
  This is a multi-line string
  which preserves line breaks.
  Useful for writing big texts.

folded_string: >
  This is a folded
  multi-line string that becomes
  a single line with spaces.
```

📚 Lists (Arrays)

```yaml
languages:
  - JavaScript
  - Python
  - Go
  - Rust

short_list: [Red, Green, Blue]
```

🧱 Dictionaries (Maps / Objects)

```yaml
database:
  host: localhost
  port: 5432
  username: user
  password: pass
```

🧬 Nested Structures

```yaml
infrastructure:
  cloud: AWS
  region: ap-south-1
  services:
    - EC2
    - S3
    - Lambda
```

⚙️ Anchors (&) and Aliases (*)

```yaml
defaults: &default_settings
  retries: 3
  timeout: 30

production:
  <<: *default_settings
  timeout: 60  # override
```

🧪 Boolean Gotchas (YAML 1.1 vs 1.2)

- YAML 1.1 treats `yes`, `no`, `on`, `off` as booleans (❌ unpredictable)
- YAML 1.2 strictly uses `true` and `false` (✅ recommended)

```yaml
bool_true: true
bool_false: false
```

📦 Kubernetes Pod Example

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: my-nginx-pod
  labels:
    app: nginx
spec:
  containers:
    - name: nginx
      image: nginx:1.29
      ports:
        - containerPort: 80
```

🐳 Docker Compose Example

```yaml
version: "3.9"
services:
  web:
    image: nginx:alpine
    ports:
      - "8080:80"
  db:
    image: postgres:15
    environment:
      POSTGRES_USER: admin
      POSTGRES_PASSWORD: secret
      POSTGRES_DB: mydb
```

⚙️ GitHub Actions CI/CD Workflow

```
name: Node CI

on:
  push:
    branches: [main]
  pull_request:
    branches: [main]

jobs:
  build:
    runs-on: ubuntu-latest
    steps:
      - name: Checkout code
        uses: actions/checkout@v3

      - name: Set up Node.js
        uses: actions/setup-node@v3
        with:
          node-version: 20

      - name: Install dependencies
        run: npm install

      - name: Run tests
        run: npm test
```

✅ DOs and ❌ DON'Ts

✅ **DOs**	❌ **DON'Ts**
Use **2-space** indentation	❌ Never use **tabs**
Quote strings starting with `!`, `@`, `#`, or numbers	❌ Don't rely on implicit parsing
Use hyphens for lists	❌ Don't align values manually

```
# ❌ Incorrect
key:    value

# ✅ Correct
key: value
```

📎 Tips & Tools

- ✅ Use linters like `yamllint` or `kubeval`
- 🧠 Install VS Code extensions:
    - "YAML" by Red Hat
    - "Kubernetes" by Microsoft

🚀 Keep YAMLing Like a Pro!

With great indentation comes great configuration power. Happy YAMLing! ✨ Made with ❤️ by **Darshan Vasani**

🧱 1. Pod (nginx example)

```
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
  labels:
    app: nginx
spec:
  containers:
    - name: nginx-container
      image: nginx:latest
      ports:
        - containerPort: 80
```

🚀 2. Deployment

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: nginx
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
      - name: nginx
        image: nginx:1.25
        ports:
        - containerPort: 80
```

🌐 3. Service (ClusterIP, NodePort, LoadBalancer)

```
apiVersion: v1
kind: Service
metadata:
  name: nginx-service
spec:
  type: NodePort  # Can be ClusterIP, NodePort, LoadBalancer
  selector:
    app: nginx
  ports:
    - protocol: TCP
      port: 80
      targetPort: 80
      nodePort: 30036  # Only for NodePort
```

📦 4. ConfigMap

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_ENV: production
  DB_HOST: localhost
```

🔐 5. Secret (base64 encoded)

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
data:
  username: dXNlcm5hbWU=  # base64 encoded
  password: cGFzc3dvcmQ=  # base64 encoded
```

To encode:

```bash
echo -n 'username' | base64
```

🗂️ 6. PersistentVolume (Static Provisioning)

```yaml
apiVersion: v1
kind: PersistentVolume
metadata:
  name: pv-example
spec:
  capacity:
    storage: 1Gi
  accessModes:
    - ReadWriteOnce
  hostPath:
    path: /mnt/data
```

📁 7. PersistentVolumeClaim

```yaml
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-example
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

🧾 8. Job

```
apiVersion: batch/v1
kind: Job
metadata:
  name: simple-job
spec:
  template:
    spec:
      containers:
      - name: hello
        image: busybox
        command: ["echo", "Hello from Kubernetes Job"]
      restartPolicy: Never
```

🔁 9. CronJob

```
apiVersion: batch/v1
kind: CronJob
metadata:
  name: hello-cron
spec:
  schedule: "*/5 * * * *"  # Every 5 minutes
  jobTemplate:
    spec:
      template:
        spec:
          containers:
          - name: hello
            image: busybox
            args:
            - /bin/sh
            - -c
            - date; echo Hello from cron job
          restartPolicy: OnFailure
```

🌍 10. Ingress (with NGINX controller)

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: example-ingress
  annotations:
```

```yaml
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: nginx-service
                port:
                  number: 80
```

📂 11. Namespace

```yaml
apiVersion: v1
kind: Namespace
metadata:
  name: dev-namespace
```

📎 12. ServiceAccount

```yaml
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-serviceaccount
  namespace: default
```

📚 13. RBAC (Role, RoleBinding)

```yaml
# Role (access within a namespace)
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  namespace: default
  name: pod-reader
rules:
  - apiGroups: [""]
```

```
    resources: ["pods"]
    verbs: ["get", "watch", "list"]
```

```
# RoleBinding
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: read-pods-binding
  namespace: default
subjects:
  - kind: ServiceAccount
    name: my-serviceaccount
    namespace: default
roleRef:
  kind: Role
  name: pod-reader
  apiGroup: rbac.authorization.k8s.io
```

🧹 14. Clean-up Job (TTL)

```
apiVersion: batch/v1
kind: Job
metadata:
  name: ttl-cleanup-job
spec:
  ttlSecondsAfterFinished: 60
  template:
    spec:
      containers:
        - name: cleanup
          image: busybox
          command: ["sh", "-c", "echo Cleaning... && sleep 10"]
      restartPolicy: Never
```

📌 Notes:

- All YAMLs are compatible with `kubectl apply -f file.yml`
- Always validate using:

```
kubectl apply -f file.yml --dry-run=client -o yaml
```

- You can use `kubectl explain` for any field:

```
kubectl explain deployment.spec.template.spec.containers
```
