Kubernetes YAML File: Data Types and Structures

In Kubernetes, YAML (`.yml` or `.yaml`) files are used to define the desired state of cluster resources. These YAML files describe different Kubernetes objects such as Deployments, Services, ConfigMaps, and Secrets. The structure of a Kubernetes YAML file follows a specific format and contains different types of data.

Basic Structure of a Kubernetes YAML File

A Kubernetes YAML file consists of:

1. ``: Defines the Kubernetes API version for the resource.
2. ``: Specifies the type of resource (e.g., `Pod`, `Service`, `Deployment`).
3. ``: Contains metadata like name, labels, and namespace.
4. ``: Describes the desired state of the resource.

Example:

```
apiVersion: v1
kind: Pod
metadata:
  name: my-pod
  labels:
    app: my-app
spec:
  containers:
    - name: my-container
      image: nginx
```

Types of Data in Kubernetes YAML

Kubernetes YAML files contain different types of data:

1. Scalars (Simple Values)

Scalars include string, integer, float, and boolean values.

Examples:

String

```
 name: "my-service"
```

-

Integer
replicas: 3

-

Float
cpu: 0.5

-

Boolean
enabled: true

-

---

2. Lists (Arrays)

Lists in Kubernetes YAML are used for specifying multiple values.

Example:

```
containers:
  - name: app-container
    image: my-app:v1
  - name: sidecar-container
    image: logging-agent:v1
```

Here, `containers` is a list containing two items.

---

3. Maps (Dictionaries)

Maps contain key-value pairs and are used for structuring complex data.

Example:

```
metadata:
  name: my-deployment
  labels:
    app: my-app
    version: v1
```

Here, `metadata` is a map containing keys (`name`, `labels`), and `labels` is another nested map.

---

4. Multi-line Strings

YAML allows multi-line strings using `|` (literal block) and `>` (folded block).

Literal Block (``): Preserves new lines.

```
config: |
 server {
   listen 80;
   server_name example.com;
 }
```

-

Folded Block (``): Converts new lines into spaces.

```
message: >
 This is a long message
 that will be collapsed into a single line.
```

- Output: `"This is a long message that will be collapsed into a single line."`

---

5. Environment Variables (Key-Value Pairs)

Environment variables can be defined inside containers.

Example:

```
env:
  - name: DATABASE_URL
    value: "postgres://user:pass@db:5432/mydb"
  - name: DEBUG
    value: "true"
```

---

6. References (Using ConfigMaps & Secrets)

Instead of hardcoding values, Kubernetes allows referencing external resources like ConfigMaps and Secrets.

Example: Using ConfigMap

```
env:
  - name: APP_CONFIG
    valueFrom:
      configMapKeyRef:
```

```
      name: my-config
      key: app_settings
```

Example: Using Secret

```
env:
  - name: DB_PASSWORD
    valueFrom:
      secretKeyRef:
        name: db-secret
        key: password
```

Types of Kubernetes YAML Files

Different types of Kubernetes resources have different YAML file formats.

1. Deployment (Manages Pods)

Defines a set of identical Pods with scaling and rolling updates.

Example:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: my-deployment
spec:
  replicas: 3
  selector:
    matchLabels:
      app: my-app
  template:
    metadata:
      labels:
        app: my-app
    spec:
      containers:
        - name: my-container
          image: my-app:v1
```

---

2. Service (Exposes Pods)

Defines networking rules for exposing applications.

Example:

```
apiVersion: v1
kind: Service
metadata:
  name: my-service
spec:
  selector:
    app: my-app
  ports:
    - protocol: TCP
      port: 80
      targetPort: 8080
  type: LoadBalancer
```

---

3. ConfigMap (Stores Configuration Data)

Holds key-value configuration that can be injected into Pods.

Example:

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: my-config
data:
  app_settings: "production"
  log_level: "info"
```

---

4. Secret (Stores Sensitive Data)

Stores encrypted data such as passwords and API keys.

Example:

```
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
data:
  password: cGFzc3dvcmQ=  # Base64 encoded value
```

---

5. PersistentVolumeClaim (Storage for Pods)

Requests persistent storage for a Pod.

Example:

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

---

6. Ingress (Manages External Access)

Defines HTTP/HTTPS routing to services.

Example:

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: my-ingress
spec:
  rules:
    - host: my-app.example.com
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: my-service
                port:
                  number: 80
```

Conclusion

Kubernetes YAML files contain different types of data, including scalars, lists, maps, and references. Understanding these structures helps in defining deployments, services, configurations, and networking efficiently.