🚀 Kubernetes YAML Resources – A Detailed Guide

This repository is a collection of all my hands-on Kubernetes YAML files I created while learning the foundational concepts. Each file reflects an individual concept and its implementation using YAML.

📂 Folder Structure

```
k8-resources/
├── volumes/ # Volume-related examples
├── 01-namespace.yaml # Create custom namespaces
├── 02-pod.yaml # Basic pod definition
├── 03-multi-container.yaml # Pod with multiple containers
├── 04-labels.yaml # Add metadata labels
├── 05-annotations.yaml # Add non-identifying metadata
├── 06-env.yaml # Set environment variables in containers
├── 07-resources.yaml # Resource requests and limits
├── 08-config-map.yaml # Configuration management
├── 09-pod-config.yaml # Inject configuration into pods
├── 10-secret.yaml # Store sensitive data
├── 11-pod-secret.yaml # Use secrets inside pods
├── 12-service.yaml # ClusterIP service
├── 13-node-port.yaml # NodePort service
├── 14-load-balancer.yaml # LoadBalancer service (for cloud)
├── 15-replica-set.yaml # Pod replication
├── 16-deployment.yaml # Deployment with rolling updates
├── 17-statefulset.yaml # For stateful applications like databases
├── 18-service-account.yaml # ServiceAccounts
├── 19-init-container.yaml # Pre-task containers
├── 20-daemon-set.yaml # Run pods on every node
```

📘 YAMLs Explained with Examples

📦 01 - Namespace
Purpose: To isolate resources under a specific environment.
Example:
Creates a namespace called `dev-environment`.

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

🛠️ 02 - Pod
Purpose: Smallest deployable unit in Kubernetes.
Example:
Defines a simple Pod running a single `nginx` container.

```yaml
apiVersion: v1
kind: Pod
metadata:
  name: nginx-pod
spec:
  containers:
```

```
  - name: nginx
    image: nginx:latest
```

🥣 03 - Multi-Container Pod
Purpose: Run multiple containers inside the same Pod (e.g., sidecars).
Example:
One app container + one sidecar container running BusyBox.

```
apiVersion: v1
kind: Pod
metadata:
  name: multi-container-pod
spec:
  containers:
  - name: nginx
    image: nginx:latest
  - name: sidecar
    image: busybox:latest
```

🏷️ 04 - Labels
Purpose: Add identifying metadata to Kubernetes objects.
Example:
Labels like `app=myapp`, `env=dev` for organizing and selecting resources.

```
apiVersion: v1
kind: Pod
metadata:
  name: labeled-pod
  labels:
    app: myapp
    env: dev
spec:
  containers:
  - name: nginx
    image: nginx:latest
```

📜 05 - Annotations
Purpose: Attach non-identifying metadata to objects.
Example:
Annotate a Pod with a description or URL for documentation.

```
apiVersion: v1
kind: Pod
metadata:
  name: annotated-pod
  annotations:
    description: "This is an nginx pod"
spec:
  containers:
  - name: nginx
    image: nginx:latest
```

🌱 06 - Environment Variables
Purpose: Inject runtime config values into containers.
Example:
Set variables like `ENV=development` directly in the pod spec.

```
apiVersion: v1
kind: Pod
metadata:
  name: env-var-pod
```

```
spec:
  containers:
  - name: nginx
    image: nginx:latest
    env:
    - name: ENV
      value: development
```

🎛️ 07 - Resource Limits
Purpose: Prevent containers from using too much CPU/memory.
Example:
Request 128Mi memory, limit to 256Mi. CPU request 250m, limit 500m.

```
apiVersion: v1
kind: Pod
metadata:
  name: resource-limits-pod
spec:
  containers:
  - name: nginx
    image: nginx:latest
    resources:
      requests:
        memory: "128Mi"
        cpu: "250m"
      limits:
        memory: "256Mi"
        cpu: "500m"
```

🗂️ 08 - ConfigMap
Purpose: Manage environment-specific or app settings outside code.
Example:
Create a ConfigMap `app-config` with a key `APP_MODE=production`.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_MODE: production
```

🧩 09 - Use ConfigMap in Pod
Purpose: Inject ConfigMap values into Pods.
Example:
Use `envFrom` or mount as volume.

```
apiVersion: v1
kind: Pod
metadata:
  name: configmap-pod
spec:
  containers:
  - name: nginx
    image: nginx:latest
    envFrom:
    - configMapRef:
        name: app-config
```

🔐 10 - Secret
Purpose: Store sensitive values securely (base64 encoded).

Example:
Create a secret with a username/password for a database.

```
apiVersion: v1
kind: Secret
metadata:
  name: db-secret
type: Opaque
data:
  username: c3VwZXI=   # 'super' in base64
  password: cGFzc3dvcmQ=   # 'password' in base64
```

🔐 11 - Use Secret in Pod
Purpose: Consume secret securely via environment variable or volume.
Example:
Use `valueFrom.secretKeyRef` to set DB credentials.

```
apiVersion: v1
kind: Pod
metadata:
  name: secret-pod
spec:
  containers:
  - name: nginx
    image: nginx:latest
    env:
    - name: DB_USERNAME
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: username
    - name: DB_PASSWORD
      valueFrom:
        secretKeyRef:
          name: db-secret
          key: password
```

🔌 12 - Service (ClusterIP)
Purpose: Expose Pods within the cluster (internal access).
Example:
Defines a service that points to app Pods on port 8080.

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

🌐 13 - NodePort
Purpose: Make services available on a specific port on each node.
Example:
Expose the service at `NodeIP:30007`.

```
apiVersion: v1
kind: Service
```

```
metadata:
  name: nodeport-service
spec:
  selector:
    app: myapp
  ports:
  - protocol: TCP
    port: 8080
    targetPort: 80
    nodePort: 30007
  type: NodePort
```

🌍 14 - LoadBalancer
Purpose: Automatically get a public IP (in cloud platforms).
Example:
Type: LoadBalancer — only works on cloud like AWS, GCP, Azure.

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

♻️ 15 - ReplicaSet
Purpose: Ensure a fixed number of identical Pods are always running.
Example:
Run 3 replicas of a Pod with label `app=myapp`.

```
apiVersion: apps/v1
kind: ReplicaSet
metadata:
  name: myapp-replicaset
spec:
  replicas: 3
  selector:
    matchLabels:
      app: myapp
  template:
    metadata:
      labels:
        app: myapp
    spec:
      containers:
      - name: nginx
        image: nginx:latest
```

🚀 16 - Deployment
Purpose: Manage rolling updates and rollbacks of Pods.
Example:
Defines 2 replicas and rolling update strategy.

```
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:latest
```

17 - StatefulSet

Purpose: For stateful apps needing stable network/storage.
Example:
Use for MySQL, Cassandra, etc., with stable identity.

```
apiVersion: apps/v1
kind: StatefulSet
metadata:
  name: mysql-statefulset
spec:
  serviceName: "mysql"
  replicas: 3
  selector:
    matchLabels:
      app: mysql
  template:
    metadata:
      labels:
        app: mysql
    spec:
      containers:
      - name: mysql
        image: mysql:5.7
```

18 - ServiceAccount

Purpose: Assign identity to Pods for access control.
Example:
Create a custom service account and use in Pod spec.

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: my-service-account
---
apiVersion: v1
kind: Pod
metadata:
  name: pod-with-service-account
spec:
  serviceAccountName: my-service-account
  containers:
  - name: nginx
    image: nginx:latest
```

19 - Init Container

Purpose: Run setup tasks before main container starts.
Example:
Wait for DB readiness or pre-create config files.

```
apiVersion: v1
```

```
kind: Pod
metadata:
  name: init-container-pod
spec:
  initContainers:
  - name: init-script
    image: busybox
    command: ["sh", "-c", "sleep 10"]
  containers:
  - name: nginx
    image: nginx:latest
```

🌐 20 - DaemonSet
Purpose: Ensure one Pod runs on every (or selected) Node.
Example:
Used for log shippers, monitoring agents, etc.

```
apiVersion: apps/v1
kind: DaemonSet
metadata:
  name: fluentd
spec:
  selector:
    matchLabels:
      name: fluentd
  template:
    metadata:
      labels:
        name: fluentd
    spec:
      containers:
      - name: fluentd
        image: fluent/fluentd:v1.12-1
```

📦 Volumes Directory

This folder contains hands-on examples related to Kubernetes volumes such as:

- emptyDir
- hostPath
- configMap as volume
- secret as volume
- PersistentVolume
- PersistentVolumeClaim

Each helps manage persistent or temporary storage in Pods.

✅ Summary

Resource	Purpose
Pod	Smallest deployable unit
ConfigMap	Externalize app configs
Secret	Store encrypted sensitive data
Service	Expose Pods within/externally
ReplicaSet	Maintain desired Pod count
Deployment	Automate updates and rollback
StatefulSet	Use for stateful apps like DB
DaemonSet	Run one Pod per Node

Init Containers	Run pre-setup jobs
ServiceAccount	Pod identity and access control
