🔐 What is a Kubernetes Secret?

A Secret is a Kubernetes object used to store sensitive data such as:

Passwords

API keys

TLS certificates

DB credentials

Unlike ConfigMap, a Secret ensures that data is stored base64-encoded, and is handled more securely by Kubernetes (e.g., not shown in logs, limited RBAC access).

A screenshot of a computer

AI-generated content may be incorrect.

**🧠 How Secrets Work**

1. You create a Secret object with key-value pairs.
2. Kubernetes stores it encoded (base64).
3. You inject it into pods via:
   * Environment variables
   * Mounted volumes as files

**🧪 Your Use Case: Spring Boot App**

You had this:

env:

- name: SPRING\_DATASOURCE\_USERNAME

value: postgres

- name: SPRING\_DATASOURCE\_PASSWORD

value: admin@123

We will move this to a **Kubernetes Secret**.

**🛠️ Step-by-Step Implementation**

**✅ Step 1: Create Secret YAML (spring-db-secret.yaml)**

Use stringData (Kubernetes will convert it to base64):

apiVersion: v1

kind: Secret

metadata:

name: spring-db-secret

namespace: default

type: Opaque

stringData:

SPRING\_DATASOURCE\_USERNAME: postgres

SPRING\_DATASOURCE\_PASSWORD: admin@123

kubectl apply -f spring-db-secret.yaml

✅ Step 2: Reference Secret in Deployment YAML

env:

- name: SPRING\_DATASOURCE\_USERNAME

valueFrom:

secretKeyRef:

name: spring-db-secret

key: SPRING\_DATASOURCE\_USERNAME

- name: SPRING\_DATASOURCE\_PASSWORD

valueFrom:

secretKeyRef:

name: spring-db-secret

key: SPRING\_DATASOURCE\_PASSWORD

✅ Step 3: Full Deployment Snippet (Secrets + ConfigMap)

env:

- name: SPRING\_APPLICATION\_NAME

valueFrom:

configMapKeyRef:

name: spring-config

key: SPRING\_APPLICATION\_NAME

- name: SERVER\_PORT

valueFrom:

configMapKeyRef:

name: spring-config

key: SERVER\_PORT

- name: SPRING\_DATASOURCE\_URL

valueFrom:

configMapKeyRef:

name: spring-config

key: SPRING\_DATASOURCE\_URL

- name: SPRING\_DATASOURCE\_USERNAME

valueFrom:

secretKeyRef:

name: spring-db-secret

key: SPRING\_DATASOURCE\_USERNAME

- name: SPRING\_DATASOURCE\_PASSWORD

valueFrom:

secretKeyRef:

name: spring-db-secret

key: SPRING\_DATASOURCE\_PASSWORD

🔍 Step 4: Validate the Secret

kubectl get secret spring-db-secret -o yaml

# Decode value

kubectl get secret spring-db-secret -o jsonpath="{.data.SPRING\_DATASOURCE\_PASSWORD}" | base64 –decode

**⚙️ Optional: Mount Secrets as Files**

You can mount the secret into a pod as files instead of env vars:

volumeMounts:

- name: db-secret-volume

mountPath: "/etc/db-secrets"

readOnly: true

volumes:

- name: db-secret-volume

secret:

secretName: spring-db-secret

A screenshot of a computer secret

AI-generated content may be incorrect.

**🔁 What is Secret Auto-Rotation?**

**Secret Auto-Rotation** is the **automatic update and propagation** of credentials (like DB passwords, API tokens, certificates) **without manual intervention** or redeploying the application.

A screenshot of a computer

AI-generated content may be incorrect.

1. **Manual Rotation + Sync to Key Vault**

**2. Automated Rotation Using Azure Automation or Logic Apps**

You can build an Azure Automation Runbook (PowerShell or Python) or Logic App that:

1. Periodically logs into PostgreSQL (using admin creds).
2. Rotates the user password.
3. Updates the new password in Azure Key Vault.
4. Triggers a refresh in Kubernetes via External Secrets Operator.