## **Lab: Env variables:**

A **ConfigMap** in Kubernetes is used to store **non-confidential configuration data** in key-value pairs.

* Useful for app configs like DB URLs, feature flags, or properties.
* Can be used as **environment variables** or **files inside a container**.

## **🔹 1. Create a ConfigMap (YAML)**

Here’s a simple ConfigMap storing some app settings:

apiVersion: v1

kind: ConfigMap

metadata:

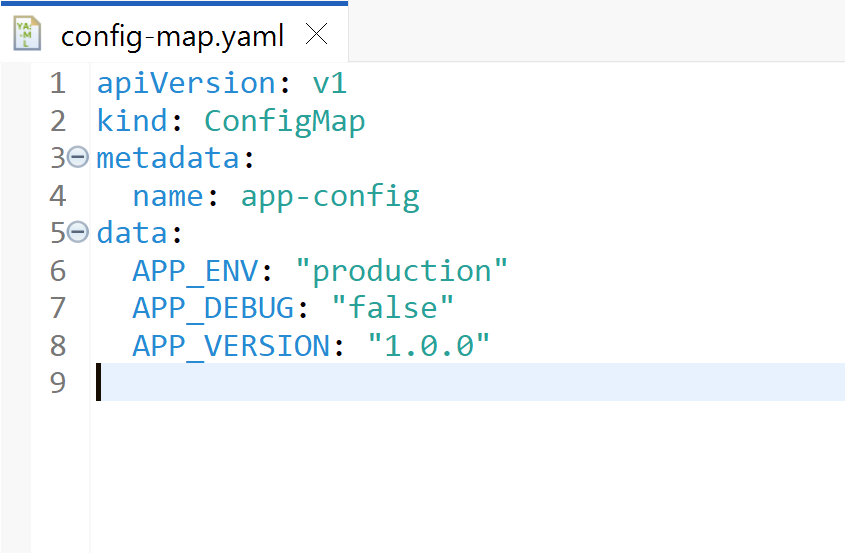
name: app-config

data:

APP\_ENV: "production"

APP\_DEBUG: "false"

APP\_VERSION: "1.0.0"



👉 Save this as config-map.yaml and apply:

kubectl apply -f config-map.yaml

Check it:

kubectl get configmap app-config -o yaml



**🔹 Use ConfigMap in a Pod (as Environment Variables)**

apiVersion: v1

kind: Pod

metadata:

name: demo-pod

spec:

containers:

- name: demo-container

image: nginx

envFrom:

- configMapRef:

name: app-config # reference ConfigMap

ports:

- containerPort: 80

👉 Now, inside this Pod, the container will have env vars:

* $APP\_ENV=production
* $APP\_DEBUG=false
* $APP\_VERSION=1.0.0

**Open a shell inside the Pod**

kubectl exec -it demo-pod -- /bin/sh

**Print environment variables**

Once inside the container shell, run:

printenv | grep APP\_

You should see:

APP\_ENV=production

APP\_DEBUG=false

APP\_VERSION=1.0.0

**Exit container**

exit

⚡ Tip: If you just want a quick check without entering the shell:

kubectl exec demo-pod -- printenv

## **2. Expose postgres env variables as config map**

**Step 1: Create a ConfigMap**

apiVersion: v1

kind: ConfigMap

metadata:

name: postgres-config

data:

POSTGRES\_DB: "postgresdb"

POSTGRES\_USER: "postgres"

POSTGRES\_PASSWORD: "postgres"

PGDATA: "/var/lib/postgresql/data/pgdata"

**Step 2: Update Deployment to use ConfigMap**

apiVersion: apps/v1

kind: Deployment

metadata:

name: postgres

spec:

replicas: 1

selector:

matchLabels:

app: postgres

template:

metadata:

labels:

app: postgres

spec:

containers:

- name: postgres-container

image: postgres:17-alpine

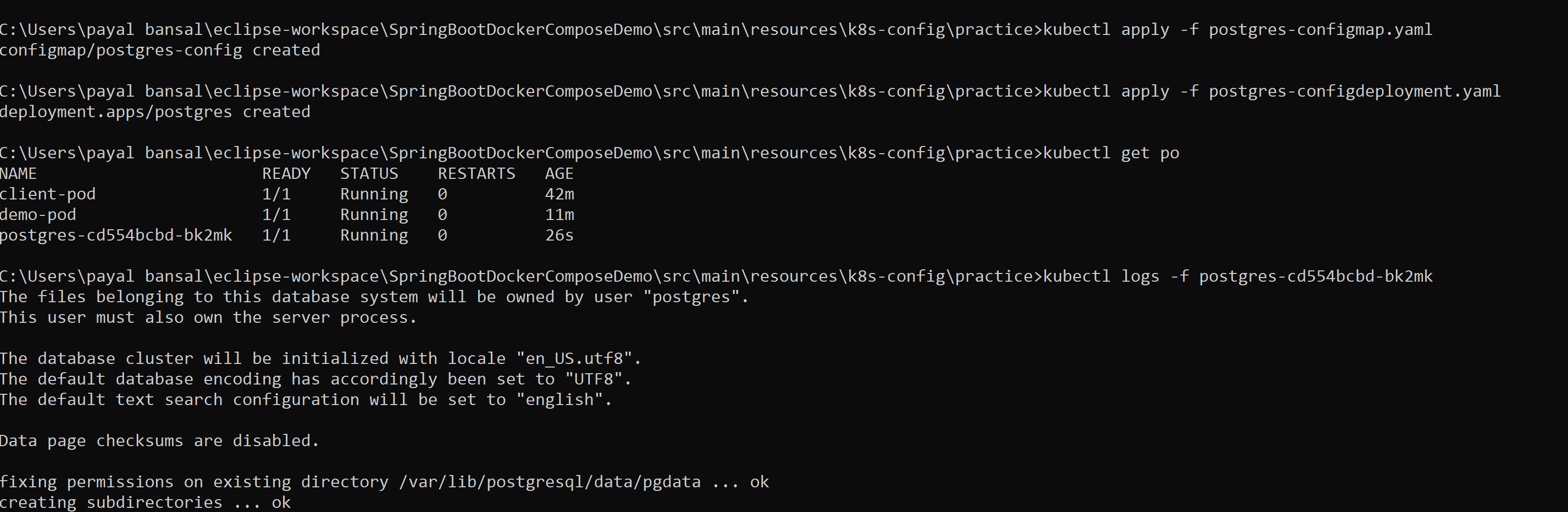
ports:

- containerPort: 5432

envFrom: # <---- load all keys from ConfigMap

- configMapRef:

name: postgres-config



**How it works:**

* The **ConfigMap** stores all your environment variables.
* envFrom.configMapRef injects all key-value pairs as environment variables into the container.
* Your container now sees:
* $POSTGRES\_DB=postgresdb
* $POSTGRES\_USER=postgres
* $POSTGRES\_PASSWORD=postgres
* $PGDATA=/var/lib/postgresql/data/pgdata

Passwords (like POSTGRES\_PASSWORD) are **better stored in Secrets** instead of ConfigMaps, because ConfigMaps are not encrypted.

## **Lab3: Use secret Map instead of configmap for db password:**

Use ConfigMap for non-sensitive values and Secret for the password

* Use **ConfigMap** for non-sensitive configs (POSTGRES\_DB, POSTGRES\_USER, PGDATA).
* Use **Secret** for sensitive configs (POSTGRES\_PASSWORD).

**Step 1: ConfigMap (non-sensitive values)**

apiVersion: v1

kind: ConfigMap

metadata:

name: postgres-config

data:

POSTGRES\_DB: "postgresdb"

POSTGRES\_USER: "postgres"

PGDATA: "/var/lib/postgresql/data/pgdata"

**Step 2: Secret (sensitive value)**

apiVersion: v1

kind: Secret

metadata:

name: postgres-secret

type: Opaque

stringData:

POSTGRES\_PASSWORD: "postgres"

(stringData is convenient — you don’t need to manually base64 encode the password; Kubernetes handles it.)

**Step 3: Deployment referencing both**

apiVersion: apps/v1

kind: Deployment

metadata:

name: postgres

spec:

replicas: 1

selector:

matchLabels:

app: postgres

template:

metadata:

labels:

app: postgres

spec:

containers:

- name: postgres-container

image: postgres:17-alpine

ports:

- containerPort: 5432

envFrom:

- configMapRef: # Load all keys from ConfigMap

name: postgres-config

- secretRef: # Load password from Secret

name: postgres-secret

**What happens now:**

Inside the container you’ll have:

$POSTGRES\_DB=postgresdb

$POSTGRES\_USER=postgres

$PGDATA=/var/lib/postgresql/data/pgdata

$POSTGRES\_PASSWORD=postgres

You can also check what envs are injected **from outside**, without entering the Pod:

kubectl describe pod postgres-5c4f77f4d6-abcde <pod name>

Scroll down to Environment Variables:

yaml

Copy code

Environment:

POSTGRES\_DB: postgresdb

POSTGRES\_USER: postgres

POSTGRES\_PASSWORD: <set to the key 'POSTGRES\_PASSWORD' in secret 'postgres-secret'> Optional: false

PGDATA: /var/lib/postgresql/data/pgdata