🧩 ConfigMap и Secret

Цель:

Научиться хранить и подключать конфигурации и секреты в Kubernetes.

🔹 Проблема

- Конфигурации часто "зашиты" в код или Pod манифест.
- Изменение требует пересборки образа.
- Секреты могут случайно попасть в Git.

Решение:

Хранить конфигурации и секреты отдельно — через **ConfigMap** и **Secret**.

⚙️ ConfigMap — что это?

ConfigMap — объект Kubernetes для хранения *неконфиденциальных данных* (настроек, путей, переменных).

Хранит пары `ключ: значение`.

Используется для:

- Переменных окружения
- Конфигурационных файлов
- Аргументов команд

🧱 Создание ConfigMap

Из файла:

```
kubectl create configmap app-config --from-file=config.yaml
```

Из переменных окружения:

```
kubectl create configmap app-config --from-env-file=.env
```

Из литералов:

```
kubectl create configmap app-config --from-literal=APP_MODE=prod
```

Пример манифеста ConfigMap

```yaml
apiVersion: v1
kind: ConfigMap
metadata:
  name: app-config
data:
  APP_MODE: "production"
  LOG_LEVEL: "info"
```

🔌 Подключение ConfigMap

1. Как переменные окружения

```yaml
envFrom:
  - configMapRef:
      name: app-config
```

2. Как volume

```yaml
volumes:
  - name: config-vol
    configMap:
      name: app-config
volumeMounts:
  - name: config-vol
    mountPath: /app/config
```

🧠 Пример использования

```
env:
  - name: APP_MODE
    valueFrom:
      configMapKeyRef:
        name: app-config
        key: APP_MODE
```

Проверка:

```
kubectl exec pod -- printenv | grep APP_MODE
```

🔒 Secret — что это?

Secret — объект Kubernetes для хранения чувствительных данных:

- пароли
- токены
- ключи

Отличия от ConfigMap:

- Данные кодируются **base64**
- Ограничен доступ (`kubectl get secret` не показывает содержимое)
- Хранятся в **etcd** в закодированном виде

🧱 Создание Secret

Из литералов:

```
kubectl create secret generic secret-db \
  --from-literal=username=admin \
  --from-literal=password=Pa55w0rd
```

Из файла:

```
kubectl create secret generic tls-secret \
  --from-file=tls.crt --from-file=tls.key
```

Пример манифеста Secret

```yaml
apiVersion: v1
kind: Secret
metadata:
  name: secret-db
type: Opaque
data:
  username: YWRtaW4=
  password: UGE1NXcwcmQ=
```

🔌 Подключение Secret

Как env vars

```yaml
envFrom:
  - secretRef:
      name: secret-db
```

Как volume

```yaml
volumes:
  - name: secret-vol
    secret:
      secretName: secret-db
volumeMounts:
  - name: secret-vol
    mountPath: /app/secret
```

🧪 Проверка Secret

Показать значение:

```
kubectl get secret secret-db -o yaml
```

Декодировать:

```
echo "YWRtaW4=" | base64 --decode
```

🧍 Практика (обзор)

1. Создать ConfigMap (3 способа)
2. Подключить как env vars и volume
3. Создать Secret (из literal)
4. Подключить Secret в Pod
5. Проверить доступ из контейнера

⚔️ Сравнение ConfigMap vs Secret

Характеристика	ConfigMap	Secret
Тип данных	Неконфиденциальные	Конфиденциальные
Кодирование	Нет	Base64
Использование	Настройки, пути	Пароли, токены
Подключение	env, volume	env, volume
Защита	Нет	RBAC, encryption

🧰 Безопасность Secret

- Ограничить доступ через **RBAC**
- Включить **encryption at rest** для etcd
- Использовать внешние менеджеры секретов:
  - AWS Secrets Manager
  - HashiCorp Vault
  - External Secrets Operator

✅ Итоги

- **ConfigMap** — хранит некритичные конфигурации
- **Secret** — хранит чувствительные данные
- Оба можно подключать как:
  - env vars
  - volumes

🏠 Домашнее задание

Создать Pod, который:

1. Подключает ConfigMap как env vars
2. Подключает Secret как volume
3. Проверить значения внутри контейнера через `kubectl exec`