컨피그맵과 시크릿: 애플리케이션 설정

컨테이너화된 애플리케이션 설정

1. 컨테이너에 명령줄 인자 전달
2. 각 컨테이너를 위한 사용자 정의 환경변수 지정
3. 특수한 유형의 볼륨을 통해 설정 파일을 컨테이너에 마운트

컨테이너에 명령줄 인자 전달

- 컨테이너에서 실행하는 전체 명령이 명령어와 인자의 두 부분으로 구성되어 있다.

도커와 쿠버네티스의 실행파일과 인자를 지정하는 방법 비교

Aa Name	컨테이너 안에서 실행되는 실행파일	실행 파일에 전달되는 인자
Docker	ENTRYPOINT	CMD
k8s	command	args

컨테이너의 환경변수 설정

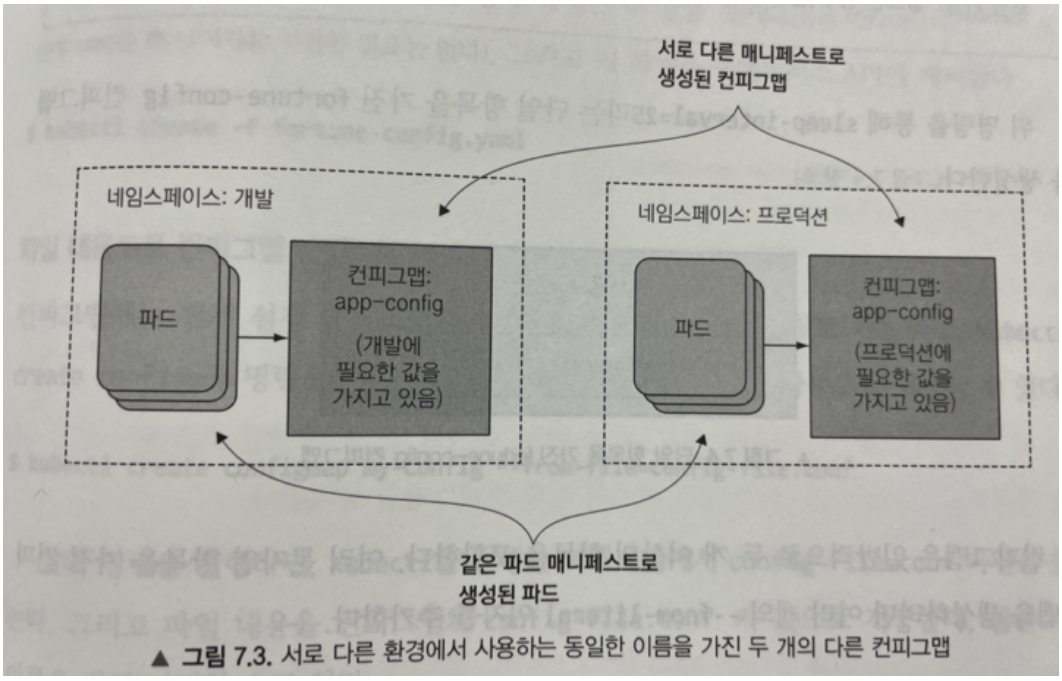
- 파드의 각 컨테이너를 위한 환경변수 리스트를 지정할 수 있다.
- 현재는 파드 수준에서 환경변수를 설정하고 컨테이너 상속받는 옵션이 존재하지 않는다.
- 파드를 만들 때 환경변수를 컨테이너 정의에 포함해 스크립트에 전달할 수 있다.

```
kind: Pod
spec:
  containers:
  - image: luksa/fortune:env
    # 환경변수 목록에 단일 변수 추가. 컨테이너 정의 안에 설정.
    env:
    - name: INTERVAL
      value: "30"
    name: html-generator
```

- 파드 정의에 하드코딩된 값을 가져오는 것은 효율적이지만, 프로덕션과 개발을 위해 서로 분리된 파드 정의가 필요하다는 것을 뜻한다.
  ⇒ 여러 환경에서 동일한 파드 정의를 재사용하려면 파드 정의에서 설정을 분리하는 것이 좋다.
  ⇒ configMap 사용.

컨피그맵

- 쿠버네티스에서는 설정 옵션을 컨피그맵이라 부르는 별도 오브젝트로 분리할 수 있다.
- 짧은 문자열에서 전체 설정 파일에 이르는 값을 가지는 키/값 쌍으로 구성된 맵.



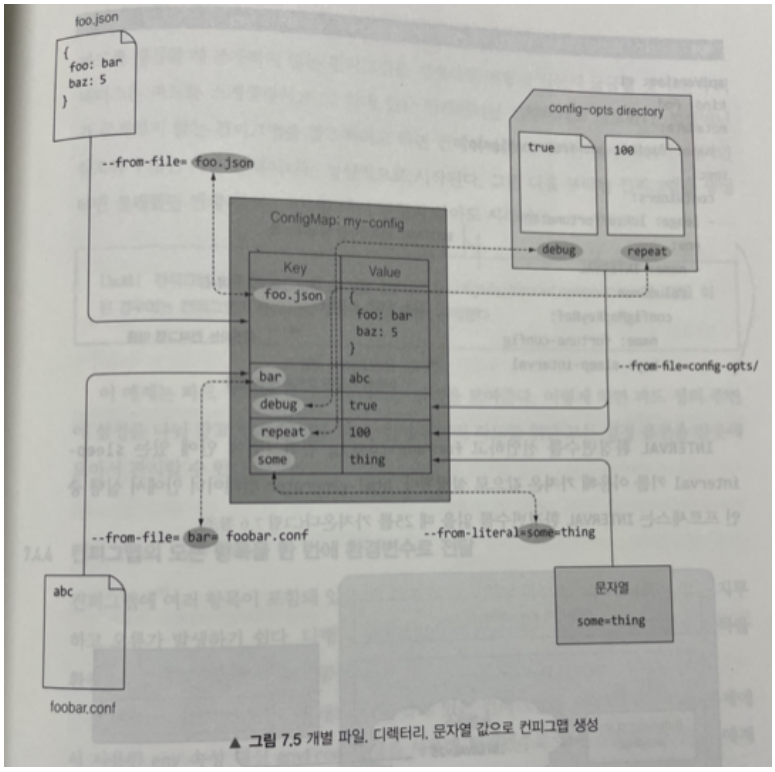
▲ 그림 7.3. 서로 다른 환경에서 사용하는 동일한 이름을 가진 두 개의 다른 컨피그맵

- 각각 다른 환경에대해 동일한 이름으로 컨피그맵에 관한 매니페스트를 유지할 수 있다.
  ⇒ 모든 환경에서 동일한 파드 정의를 사용해 각 환경에서 서로 다른 설정을 사용할 수 있다.

▼ 컨피그맵 생성

```
kubectl create configmap my-config
  --from-file=foo.json              # 단일파일
```

```
  --from-file=bar=foobar.conf      # 사용자 정의 키 밑에 파일 저장
  --from-fole=config-opts          # 전체 디렉터리
  --from-literal=some=thing        # 문자열 값
```



▲ 그림 7.5 개별 파일, 디렉터리, 문자열 값으로 컨피그맵 생성

▼ 환경변수를 컨피그맵에서 가져오는 파드 선언

```
# 환경변수를 컨피그맵에서 가져오는 파드 선언
apiVersion: v1
kind: Pod
metadata:
  name: fortune-env-from-configmap
spec:
  containers:
  - image: luksa/fortune:env
    env:
    # INTERVAL 환경변수를 설정하는 중
    - name: INTERVAL
      # 고정 값을 설정하는 대신 컨피그맵 키에서 값을 가져와 초기화한다.
      valueFrom:
        configMapKeyRef:
          name: fortune-config
          # 컨피그맵에서 해당 키 아래에 저장된 값으로 변수 설정
          key: sleep-interval
```

▼ **컨피그맵의 모든 항목을 한 번에 환경변수로 전달**

env 속성 대신 envFrom 속성을 사용해 환경변수로 모두 노출 가능.

▼ **컨피그맵 항목을 명령줄 인자로 전달**

$(ENVVARIABLENAME) 문법을 사용해 쿠버네티스가 해당 변수의 값을 인자에 주입한다.

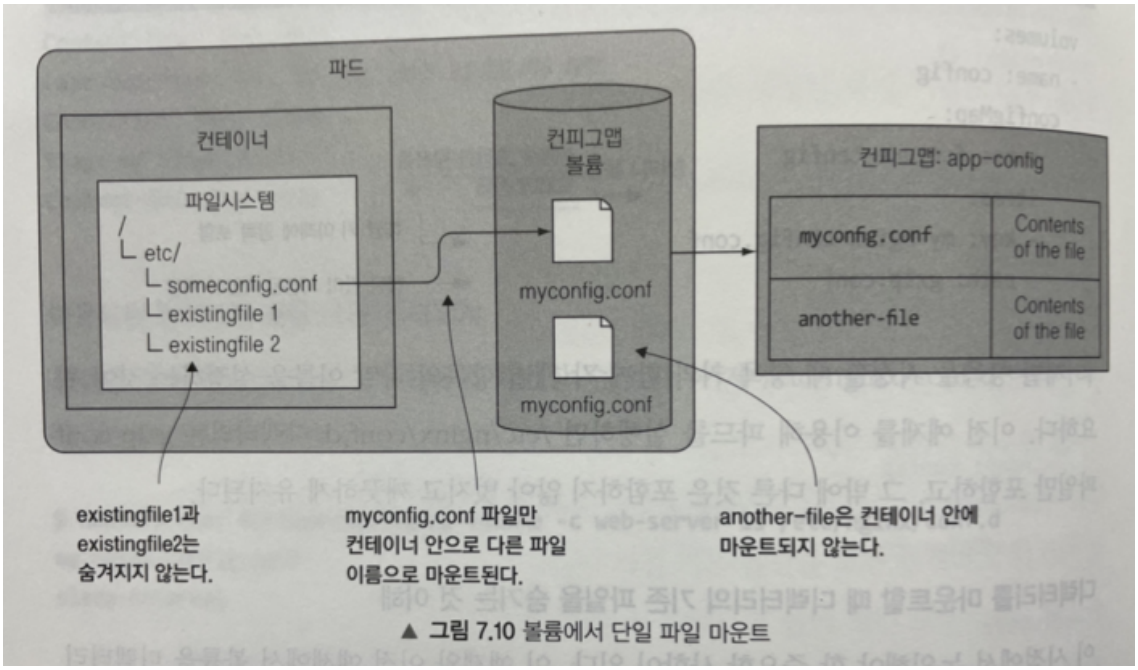
▼ **컨피그맵 볼륨을 사용해 컨피그맵 항목을 파일로 노출**

- 컨피그맵 볼륨은 파일로 컨피그맵의 각 항목을 노출한다.
- 컨테이너에서 실행 중인 프로세스는 이 파일 내용을 읽어 각 항목의 값을 얻을 수 있다.
- 컨피그맵 볼륨을 컨피그맵 항목의 일부만으로 채울 수 있다.

```
# 지정한 컨피그맵 항목을 파일 디렉터리에 마운트한 파드

volumes:
- name: config
  configMap:
    name: fortune-config
    items:      # 볼륨에 포함할 항목을 조회해 선택
    - key: my-nginx-config.conf    # 해당 키 아래에 항목 포함
      path: gzip.conf     #  항목 갑이 지정된 파일에 저장
```

- **디렉터리 안에 다른 파일을 숨기지 않고 개별 컨피그맵 항목을 파일로 마운트**



▲ 그림 7.10 볼륨에서 단일 파일 마운트

subPath 속성으로 디렉터리에 있는 다른 파일에 영향을 주지 않고 마운트할 수 있다.

- 기본적으로 컨피그맵 볼륨의 모든 파일 권한은 644(-rw-r-r—)로 설정된다.
  볼륨 정의에서 defaultMode 속성을 설정해 변경할 수 있다.

▼ 애플리케이션을 재시작하지 않고 애플리케이션 설정 업데이트

- 컨피그맵을 업데이트하면, 이를 참조하는 모든 볼륨의 파일이 업데이트된다.
- 컨피그맵이 업데이트되면 쿠버네티스는 새 디렉터리를 생성하고, 모든 파일을 여기에 쓴 다름 심볼릭 링크가 새 디렉터리를 가리키도록 해 모든 파일을 한번에 효과적으로 변경한다.

시크릿

- 자격증명, 개임 암호화 키와 같은 민감한 정보를 포함하는 설정을 보관하고 배포하기 위한 오브젝트.
- 키-값 쌍을 가진 맵.
- 쿠버네티스는 시크릿에 접근해야 하는 파드가 실행되고 있는 노드에만 개별 시크릿을 배포해 시크릿을 안전하게 유지한다.

- 노드 자체적으로 시크릿을 항상 메모리에만 저장되게 하고 물리 저장소에 기록되지 않도록 한다.

▼ 시크릿 유형

- docker-registry: 도커 레지스트리를 사용하기 위함.
- tls: TLS 통신을 위함.
- generic

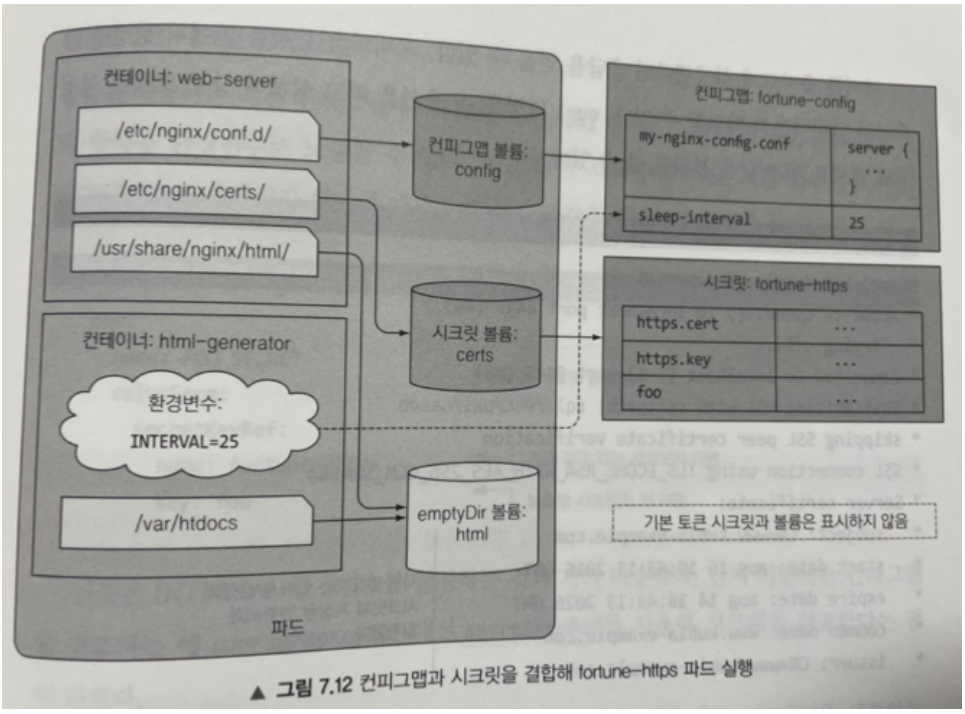
▼ 시크릿 생성

```
kubectl create secret generic fortune-https --from-file=https.key
  --from-file=htttps.cert --from-file=foo
```

▼ 컨피그맵과 시크릿 비교

시크릿 항목의 내용은 Base64 인코딩 문자열로 표시되고, 컨피그맵의 내용은 일반 텍스트로 표시된다.

▼ 파드에서 시크릿 사용



▲ 그림 7.12 컨피그맵과 시크릿을 결합해 fortune-https 파드 실행

- secret 볼륨은 시크릿 파일을 저장하는데 인메모리 파일시스템(tmpfs)을 사용한다. tmpfs를 사용하는 이유는 민감한 데이터를 노출시킬 수도 있는 디스크에 저장하지 않기 위해서이다.
- 환경변수로 시크릿 항목을 노출
  ⇒ configMapKeyRef 대신 secretKeyRef를 사용해 시크릿을 참조한다.

```
env:
- name: FOO_SECRET
  valueFrom:
    secretKeyRef:          # 변수는 시크릿항목에서 설정된다.
      name: fortune-https # 키를 갖고 있는 시크릿의 이름
      key: foo             # 노출할 시크릿의 키 이름
```

▼ 이미지를 가져올 때 사용하는 시크릿 이해

- 도커 허브에서 프라이빗 이미지를 사용한다.

▼ 프라이빗 저장소를 사용하는 파드를 실행하려면

- 도커 레지스트리 자격증명을 가진 시크릿 생성
- 파드 매니페스트 안에 imagePullSecrets 필드에 해당 시크릿 참조.