

Contents

- ConfigMap, Secret 생성하고 사용하기

ConfigMap, Secret 생성하고 사용하기

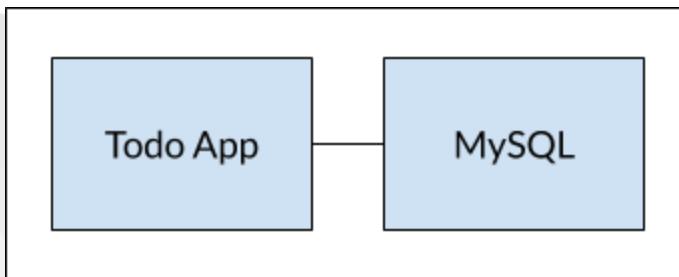
ToDo App을 또 다른 방법으로 실행해보려고 합니다.

Docker Network 실습에서 했던 방법과 동일하게

- MySQL
- Todo App

을 각각 실행해서 연동하려고 합니다.

이렇게요.



기본적인 구조와 방법은 Docker Network 실습과 같고, Kubernetes환경에 맞게 리소스들을 생성해서 해볼게요.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

먼저 MySQL을 실행합니다.

Docker에서는 아래와 같이 했습니다.

```
ubuntu@ip-10-0-1-14:~$ docker run -d \
--network todo-app --network-alias mysql \
--volume todo-mysql-data:/var/lib/mysql \
--env MYSQL_ROOT_PASSWORD=secret \
--env MYSQL_DATABASE=todos \
--env LANG=C.UTF-8 \
--name my-mysql \
mysql:5.7 \
--character-set-server=utf8mb4 \
--collation-server=utf8mb4_unicode_ci
c9d83cbd2ac8941da32d8d64103223fe1c6937c9c28507c6e19ed91fca740c98
```

환경변수를 이용해서 MySQL 설정을 하고, user-defined bridge 네트워크를 이용했습니다.

쿠버네티스에서는 설정에 필요한 환경변수를 ConfigMap과 Secret으로 만들어서 사용해볼게요.

- **ConfigMap**으로 만들 환경변수
 - MYSQL_DATABASE
 - LANG
- **Secret**으로 만들 환경변수
 - MYSQL_ROOT_PASSWORD

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

ConfigMap과 Secret은 각각 아래와 같이 준비합니다.

```
apiVersion: v1
kind: ConfigMap
metadata:
  name: mysql-config
data:
  database: "todos"
  lang: "C.UTF-8"
```

파일명은 mysql-configmap.yaml로 합니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: mysql-secret
type: Opaque
data:
  password: c2VjcmV0
```

파일명은 mysql-secret.yaml로 합니다.

password("c2VjcmV0")는 "secret"을 base64 encoding한 것입니다.
base64 encoding은 아래처럼 하면 됩니다.

```
ubuntu@ip-172-31-20-30:~$ echo -n 'secret' | base64
c2VjcmV0
```

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

ConfigMap과 Secret을 위한 파일들이 준비됐으면, 다음은 pvc를 위한 파일도 하나 준비해주세요.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: mysql-pvc
spec:
  storageClassName: standard
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 3Gi
```

파일명은 mysql-pvc.yaml로 합니다.

그리고, 이제 다들 아시겠지만 Service도 만들어야하니 준비해 주시구요.

```
apiVersion: v1
kind: Service
metadata:
  name: todo-mysql-svc
spec:
  type: ClusterIP
  selector:
    app: todo-mysql
  ports:
    - protocol: TCP
      port: 3306
      targetPort: 3306
```

파일명은 mysql-clusterip-service.yaml로 합니다.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

마지막으로 Deployment를 준비합니다.

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: todo-mysql-deployment
  labels:
    app: mysql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: todo-mysql
  template:
    metadata:
      labels:
        app: todo-mysql
    spec:
      containers:
        - name: todo-mysql-pod
          image: mysql:5.7
          env:
            - name: MYSQL_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mysql-secret
                  key: password
            - name: MYSQL_DATABASE
              valueFrom:
                configMapKeyRef:
                  name: mysql-config
                  key: database
            - name: LANG
              valueFrom:
                configMapKeyRef:
                  name: mysql-config
                  key: lang
          ports:
            - containerPort: 3306
          args: [--character-set-server=utf8mb4, --collation-server=utf8mb4_unicode_ci]
          volumeMounts:
            - mountPath: /var/lib/mysql
              name: mysql-storage
          volumes:
            - name: mysql-storage
              persistentVolumeClaim:
                claimName: mysql-pvc
```

파일명은 mysql-deployment.yaml로 합니다.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

이제 하나씩 생성해줍니다.

모두 다섯 개의 리소스를 생성해야 하니 천천히 생성해주세요.

```
ubuntu@ip-172-31-20-30:~$ kubectl apply -f mysql-configmap.yaml
configmap/mysql-config created
ubuntu@ip-172-31-20-30:~$ kubectl apply -f mysql-secret.yaml
secret/mysql-secret created
ubuntu@ip-172-31-20-30:~$ kubectl apply -f mysql-pvc.yaml
persistentvolumeclaim/mysql-pvc created
ubuntu@ip-172-31-20-30:~$ kubectl apply -f mysql-clusterip-service.yaml
service/todo-mysql-svc created
ubuntu@ip-172-31-20-30:~$ kubectl apply -f mysql-deployment.yaml
deployment.apps/todo-mysql-deployment created
```

명령어 : `kubectl apply -f mysql-configmap.yaml`

명령어 : `kubectl apply -f mysql-secret.yaml`

명령어 : `kubectl apply -f mysql-pvc.yaml`

명령어 : `kubectl apply -f mysql-clusterip-service.yaml`

명령어 : `kubectl apply -f mysql-deployment.yaml`

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

ConfigMap이 잘 생성됐나 볼까요?

```
ubuntu@ip-172-31-20-30:~$ kubectl get configmaps
NAME          DATA   AGE
kube-root-ca.crt   1    3d9h
mysql-config     2    6m23s
ubuntu@ip-172-31-20-30:~$ kubectl describe configmaps mysql-config
Name:         mysql-config
Namespace:    default
Labels:       <none>
Annotations: <none>

Data
=====
lang:
-----
C.UTF-8
database:
-----
todos

BinaryData
=====

Events:  <none>
```

명령어 : `kubectl get configmaps`

명령어 : `kubectl describe configmaps mysql-config`

lang(C.UTF-8)과 database(todos)두 개의 데이터가 보입니다.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

Secret도 볼게요.

```
ubuntu@ip-172-31-20-30:~$ kubectl get secrets
NAME          TYPE           DATA  AGE
default-token-r7g6g  kubernetes.io/service-account-token  3      3d9h
mysql-secret   Opaque          1      6m21s
ubuntu@ip-172-31-20-30:~$ kubectl describe secrets mysql-secret
Name:         mysql-secret
Namespace:    default
Labels:       <none>
Annotations: <none>

Type:        Opaque

Data
=====
password:  6 bytes
```

명령어 : `kubectl get secrets`

명령어 : `kubectl describe secrets mysql-secret`

Secret의 Data는 값이 보이지는 않네요.

Secret0|니까요. -_-

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

Pod도 확인해보세요. (Environment, Mounts, Volumes 부분을 잘 보세요.)

```
ubuntu@ip-172-31-20-30:~$ kubectl get pod
NAME                      READY   STATUS    RESTARTS   AGE
todo-mysql-deployment-78dd847547-xcl4d4   1/1     Running   0          13m
ubuntu@ip-172-31-20-30:~$ kubectl describe pod todo-mysql-deployment-78dd847547-xcl4d4
Name:           todo-mysql-deployment-78dd847547-xcl4d4
...
Containers:
  todo-mysql-pod:
    ...
    ...
  Environment:
    MYSQL_ROOT_PASSWORD: <set to the key 'password' in secret 'mysql-secret'>      Optional: false
    MYSQL_DATABASE:       <set to the key 'database' of config map 'mysql-config'>  Optional: false
    LANG:                <set to the key 'lang' of config map 'mysql-config'>    Optional: false
  Mounts:
    /var/lib/mysql from mysql-storage (rw)
    /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-pls6z (ro)
...
Volumes:
  mysql-storage:
    Type:      PersistentVolumeClaim (a reference to a PersistentVolumeClaim in the same namespace)
    ClaimName: mysql-pvc
    ReadOnly:   false
...

```

명령어 : `kubectl get pod`

명령어 : `kubectl describe pod [POD-NAME]`

[POD-NAME]에는 MySQL POD의 이름을 넣어주세요.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

이제 두 번째 워크로드인 ToDo App을 실행해볼까요?

MySQL과 마찬가지로 ConfigMap과 Secret을 사용하고, 외부에서 접속을 해야하니 Ingress까지 만들어볼게요.

이번엔 둘째 하나의 yaml파일로 만들어 보겠습니다.

```
apiVersion: v1
kind: Secret
metadata:
  name: todo-secret
type: Opaque
data:
  mysql-user: cm9vdA==
  mysql-password: c2VjcmV0
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: todo-config
data:
  mysql-host: "todo-mysql-svc.default.svc.cluster.local"
  mysql-db: "todos"
---
apiVersion: v1
kind: Service
metadata:
  name: todo-app-svc
spec:
  type: NodePort
  selector:
    app: todo-app
  ports:
    - protocol: TCP
      port: 3000
      targetPort: 3000
      nodePort: 30007
```

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

앞장에 이어서 계속.

```
---  
apiVersion: apps/v1  
kind: Deployment  
metadata:  
  name: todo-app-deployment  
  labels:  
    app: todo-app  
spec:  
  replicas: 3  
  strategy:  
    type: RollingUpdate  
  selector:  
    matchLabels:  
      app: todo-app  
  template:  
    metadata:  
      labels:  
        app: todo-app  
    spec:  
      containers:  
        - name: todo-app  
          image: [USER-NAME]/todo-app:1.0.0  
          env:  
            - name: MYSQL_HOST  
              valueFrom:  
                configMapKeyRef:  
                  name: todo-config  
                  key: mysql-host  
            - name: MYSQL_DB  
              valueFrom:  
                configMapKeyRef:  
                  name: todo-config  
                  key: mysql-db  
            - name: MYSQL_USER  
              valueFrom:  
                secretKeyRef:  
                  name: todo-secret  
                  key: mysql-user  
            - name: MYSQL_PASSWORD  
              valueFrom:  
                secretKeyRef:  
                  name: todo-secret  
                  key: mysql-password  
      ports:  
        - containerPort: 3000  
---
```

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

앞장에 이어서 계속.

```
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: todo-app-ingress
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /
spec:
  rules:
    - host: todo-app.info
      http:
        paths:
          - path: /
            pathType: Prefix
            backend:
              service:
                name: todo-app-svc
                port:
                  number: 3000
```

파일명은 `todo-all.yaml`로 합니다.

[USER-NAME]에는 여러분의 정보로 채워넣어 주세요. (58번째 라인 -> Deployment의 `.spec.containers[0].image`입니다.)

파일이 좀 길죠?

교재 `hands_on_files` 디렉토리에 `todo-all.yaml`이라는 이름으로 미리 만들어 놓았으니, 그걸 사용하셔도 됩니다.

하나의 yaml 파일 안에 여러개의 K8s [Manifest](#)를 정의할 때는, `---`을 구분자로 해서 여러개를 담으면 됩니다.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

생성하기 전에 한 가지 더 해줘야 할 일이 있습니다.

image([USER-NAME]/todo-app:1.0.0)를 여러분의 private repository에서 pull해야하기 때문에, 자격증명을 해야합니다.
여러가지 방법이 있지만, 간단하게 [커맨드 라인에서 자격 증명을 통하여 시크릿 생성하기](#) 방법으로 해볼게요.

```
ubuntu@ip-172-31-20-30:~$ kubectl create secret docker-registry regcred --docker-server=https://index.docker.io/v1/ --docker-username=rogallo --docker-password=XXXXXX  
secret/regcred created
```

명령어 :

```
kubectl create secret docker-registry regcred --docker-server=https://index.docker.io/v1/ --docker-username=[USER-NAME] --docker-password=[PASSWORD]
```

[USER-NAME]과 [PASSWORD]는 여러분의 정보로 채워넣어 주세요.

이것도 많이 쓰이는 Secret의 용도 중 하나입니다.
조회도 한 번 해보세요.

```
ubuntu@ip-172-31-20-30:~$ kubectl describe secrets regcred  
Name:         regcred  
Namespace:    default  
Labels:       <none>  
Annotations:  <none>  
  
Type:  kubernetes.io/dockerconfigjson  
  
Data  
=====  
.dockerconfigjson:  114 bytes
```

명령어 : `kubectl describe secrets regcred`

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

이제 ToDo App을 생성해볼게요.
생성은 아래처럼 한 번에 됩니다.

```
ubuntu@ip-172-31-20-30:~$ kubectl apply -f todo-all.yaml
secret/todo-secret created
configmap/todo-config created
service/todo-app-svc created
deployment.apps/todo-app-deployment created
ingress.networking.k8s.io/todo-app-ingress created
```

명령어 : `kubectl apply -f todo-all.yaml`

앞서 MySQL에서 한 것과 비슷하게 ConfigMap, Secret, Pod도 확인해보세요.
명령어만 알려드릴게요.

명령어 : `kubectl describe configmaps todo-config`

명령어 : `kubectl describe secrets todo-secret`

명령어 : `kubectl describe pod [POD-NAME]`

[POD-NAME]에는 ToDo App POD중 하나의 이름을 넣어주세요.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

POD의 환경변수를 확인하려면 아래처럼도 가능합니다.

```
ubuntu@ip-172-31-20-30:~$ kubectl exec todo-app-deployment-df65dc8b6-dmv2m -- env
PATH=/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin
HOSTNAME=todo-app-deployment-df65dc8b6-dmv2m
TERM=xterm
MYSQL_HOST=todo-mysql-svc.default.svc.cluster.local
MYSQL_DB=todos
MYSQL_USER=root
MYSQL_PASSWORD=secret
TODO_MYSQL_SVC_PORT_3306_TCP_PROTO=tcp
TODO_MYSQL_SVC_PORT_3306_TCP_PORT=3306
TODO_MYSQL_SVC_PORT_3306_TCP_ADDR=10.105.99.97
TODO_APP_SVC_SERVICE_PORT=3000
TODO_APP_SVC_PORT=tcp://10.98.198.46:3000
TODO_MYSQL_SVC_SERVICE_PORT=3306
TODO_APP_SVC_PORT_3000_TCP=tcp://10.98.198.46:3000
TODO_APP_SVC_PORT_3000_TCP_PORT=3000
TODO_APP_SVC_PORT_3000_TCP_ADDR=10.98.198.46
TODO_MYSQL_SVC_SERVICE_HOST=10.105.99.97
TODO_MYSQL_SVC_PORT=tcp://10.105.99.97:3306
TODO_MYSQL_SVC_PORT_3306_TCP=tcp://10.105.99.97:3306
TODO_APP_SVC_SERVICE_HOST=10.98.198.46
TODO_APP_SVC_PORT_3000_TCP_PROTO=tcp
...
...생략...
```

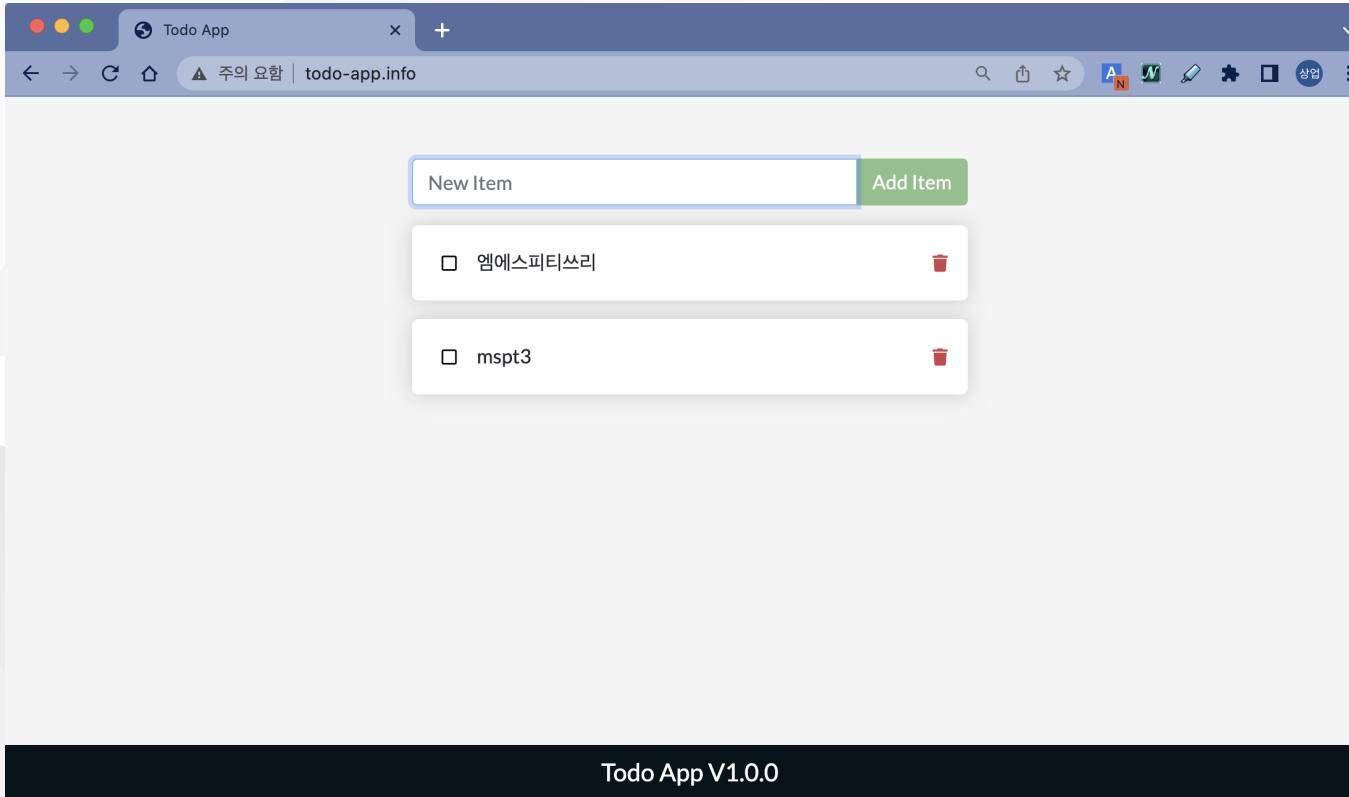
명령어 : `kubectl exec -it [POD-NAME] -- env`

[POD-NAME]에는 ToDo App POD중 하나의 이름을 넣어주세요.

- `kubectl exec` 명령어의 사용방법은 [동작중인 컨테이너의 셸에 접근하기](#)를 참고하세요.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

브라우저에서 <http://todo-app.info/> 로 접속해서 테스트도 해보시구요.



Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

MySQL DB의 테이블에 잘 저장됐는지도 확인해보세요.

```
ubuntu@ip-172-31-20-30:~$ kubectl exec -it todo-mysql-deployment-78dd847547-xcld4 -- mysql -p todos
Enter password:
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

Welcome to the MySQL monitor. Commands end with ; or \g.
Your MySQL connection id is 11
Server version: 5.7.41 MySQL Community Server (GPL)

Copyright (c) 2000, 2023, Oracle and/or its affiliates.

Oracle is a registered trademark of Oracle Corporation and/or its
affiliates. Other names may be trademarks of their respective
owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> select * from todo_items;
+-----+-----+-----+
| id      | name          | completed |
+-----+-----+-----+
| a5964fe2-4992-4a4d-ac25-d44503a4013b | 엠에스피티쓰리 |          0 |
| 0e7219c5-5843-40f3-beb0-9fc7875b118a | mspt3         |          0 |
+-----+-----+-----+
2 rows in set (0.00 sec)

mysql> exit;
Bye
```

명령어 : `kubectl exec -it [POD-NAME] -- mysql -p todos`
[POD-NAME]에는 MySQL POD의 이름을 넣어주세요.

Docker & Kubernetes - [Hands-on] 12. Kubernetes Configuration

이것저것 확인해보시고, 마지막은 생성된 리소스들을 정리해주세요.

```
ubuntu@ip-172-31-20-30:~$ kubectl delete -f todo-all.yaml
secret "todo-secret" deleted
configmap "todo-config" deleted
service "todo-app-svc" deleted
deployment.apps "todo-app-deployment" deleted
ingress.networking.k8s.io "todo-app-ingress" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete secrets regcred
secret "regcred" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f mysql-deployment.yaml
deployment.apps "todo-mysql-deployment" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f mysql-clusterip-service.yaml
service "todo-mysql-svc" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f mysql-pvc.yaml
persistentvolumeclaim "mysql-pvc" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f mysql-secret.yaml
secret "mysql-secret" deleted
ubuntu@ip-172-31-20-30:~$ kubectl delete -f mysql-configmap.yaml
configmap "mysql-config" deleted
```

명령어 : `kubectl delete -f todo-all.yaml`

명령어 : `kubectl delete secret regcred`

명령어 : `kubectl delete -f mysql-deployment.yaml`

명령어 : `kubectl delete -f mysql-clusterip-service.yaml`

명령어 : `kubectl delete -f mysql-pvc.yaml`

명령어 : `kubectl delete -f mysql-secret.yaml`

명령어 : `kubectl delete -f mysql-configmap.yaml`

이번 실습은 여기까지입니다. _↗(。|_)|_。)