

Helm Chart

운영가이드

Technical Document



Version 0.1
(2023. 07.07)

목차

내용

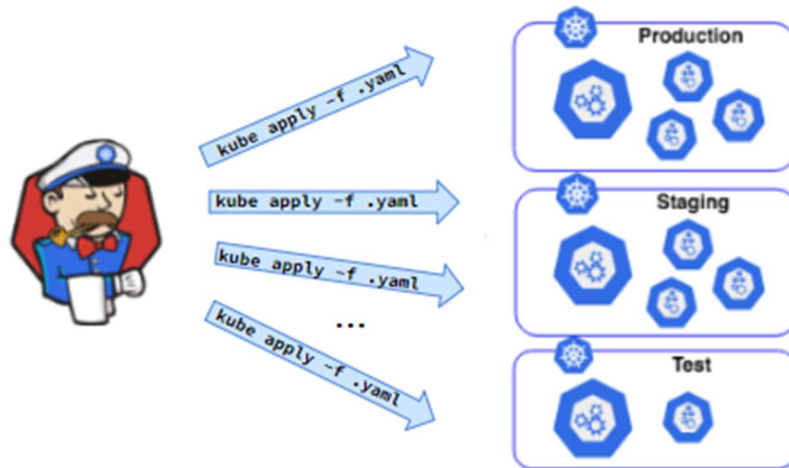
1. Helm Chart 검토배경.....	3
1.1. Helm Chart 란?.....	3
1.2. helm과 kubectl 커맨드의 역할.....	4
2. Helm CLI 설치.....	5
2.1. 설치전 선행 작업.....	5
2.1.1. 클러스터 구성.....	5
2.1.2. kubectl CLI.....	6
2.2. helm CLI 설치.....	6
3. 차트 설치.....	7
3.1. 차트 디렉토리 생성.....	7
3.1.1. helm create 로디렉토리생성.....	7
3.1.2. 디렉토리 구조 직접 생성.....	8
3.2. helm 커맨드로차트설치.....	9
3.2.1. 차트 디펜던시 빌드.....	9
3.2.2. 차트 검증.....	9
3.2.3. 차트 설치.....	10
3.2.4. 차트 업그레이드.....	10
3.2.5. 차트 버전 롤백.....	11
3.2.5. 차트 삭제.....	11
3.3. 차트 릴리즈 변경 사항 확인.....	12
0. Reference.....	13

날짜	수정이력	작성자
2023-07-07	초안생성	이준호
2023-07-10	초안작성완료	이준호

1. Helm Chart 검토배경

컨테이너 기반 MSA 환경 서비스를 쿠버네티스로 구축 하려면, 수많은 컴포넌트/매니페스트를 yaml 파일로 정의해야 하며, kubectl 커맨드를 각 컴포넌트 yaml 파일에 개별적으로 실행하여 생성 또는 수정하는 반복적인 작업이 필요 합니다.

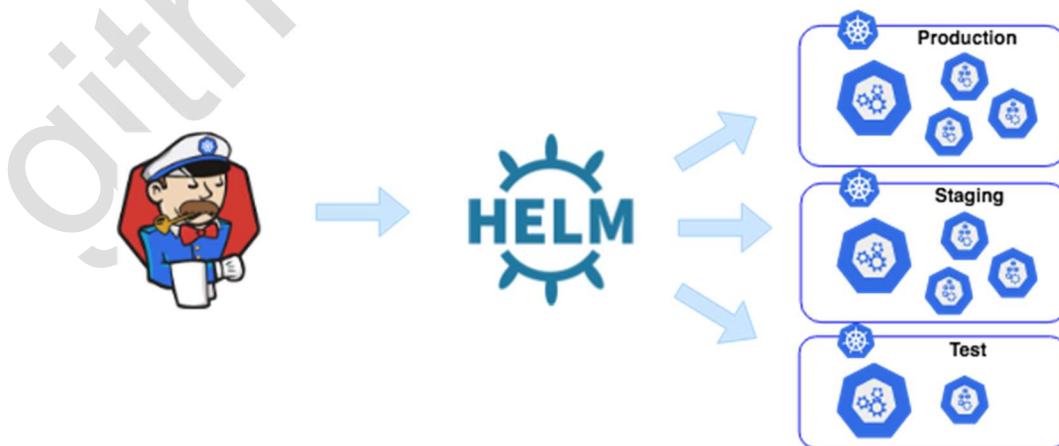
(kubectl: 쿠버네티스 클러스터의 API Server 와 통신하는 CLI 툴)



<kubectl 로 컴포넌트 생성 시 반복작업으로 인한 비효율 발생>

1.1. Helm Chart 란?

Helm Chart 패키지 매니저는 하나의 커맨드라인으로 차트를 설치하여, 다수의 서비스 컴포넌트를 한번에 생성할 수 있으며, 변경사항이 발생했을 때, helm upgrade 로 빠르게 반영할 수 있습니다. 또한, 버전 관리를 통해, 배포 건에 대한 손쉬운 rollback, upgrade, delete 등 의 작업이 가능합니다.



<helm 으로 컴포넌트 생성>from boxbeat.com

1.2. helm 과 kubectl 커맨드의 역할

Helm 은 kubectl 커맨드만으로서비즈니스별컴포넌트생성, 수정,
삭제하던반복적인작업을대체함과동시에, Chart 패키지의버전관리및롤백등의역할을합니다.
그외에, 자원의조회및로그확인,
컨테이너내부진입등디버깅은기존에사용하던 kubectl 로수행합니다. 이와같이목적에따라,
helm 과 kubectl 커맨드를함께사용해야합니다.

```
ubuntu@ip-172-34-71-29:~$ kubectl get pod -n krms | grep fe-api
fe-api-56677c5b67-fhg9l          2/2      Running    0
ubuntu@ip-172-34-71-29:~$ kubectl logs --tail=1 fe-api-56677c5b67-fhg9l
{"time": "2023-07-10T05:11:53.271852538Z", "id": "74139878-829b-4016-807f-fet", "method": "GET", "uri": "/api/v4//doc.json", "user_agent": "Mozilla/5.0 (ecko) Chrome/114.0.0.0 Safari/537.36", "status": 200, "error": "", "latency": 0.0001}
ubuntu@ip-172-34-71-29:~$ kubectl exec -it fe-api-56677c5b67-fhg9l -n krms
kubectl exec [POD] [COMMAND] is DEPRECATED and will be removed in a future version. Use kubectl exec [POD] -- [COMMAND] instead.
/ # env
FILE_AUTH_USER=
JWT_REFRESH_KEY=refreshKaiA@
FE_AUTH_SERVICE_HOST_NAME=https://lghv.rmsinfo.net
RDB_CHARSET=utf8
RDB_LOG_HOST=lghv-mariadb.rmsinfo.net
DF_API_PORT_9200_TCP_PORT=9200
```

<kubectl 커맨드>

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm upgrade my-release parent-chart/
Release "my-release" has been upgraded. Happy Helming!
NAME: my-release
LAST DEPLOYED: Mon Jul 10 05:20:52 2023
NAMESPACE: default
STATUS: deployed
REVISION: 4
TEST SUITE: None
ubuntu@ip-172-34-71-29:~/tst-helm$ helm diff revision my-release 3 4
default, df-root, Deployment (apps) has changed:
# Source: my-chart/charts/df-root/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
```

<helm 커맨드>

2. Helm CLI 설치

Helm 차트를 설치하기 전에 쿠버네티스 클러스터 구성이 필요합니다. 클러스터가 구성되면, `kubectl` 이 설치된 머신에 `helm` 설치를 진행합니다. Helm 은 기본적으로 `kubectl` CLI 툴에 설정된 동일 클러스터를 바라보도록 자동 설정됩니다.

2.1. 설치 전 선행 작업

Helm 을 설치하기 전에 쿠버네티스 클러스터 구성과 클러스터와 통신하기 위한 `kubectl` 커맨드라인 툴이 설치되어 있어야 합니다.

2.1.1. 클러스터 구성

* 쿠버네티스 클러스터 구성에는 다음과 같은 방법이 존재합니다 :

1. CSP (AWS, Google Cloud, Azure) 가 제공하는 관리형 쿠버네티스 서비스 (대표적으로 EKS, GKE, AKS) 를 활용하여 클러스터 구성.
2. On-premise 또는 EC2 인스턴스에 `microk8s` 로 경량화 클러스터 구성.

본 문서에서는 AWS EC2 우분투 Linux 머신에 `microk8s` 를 설치하여, 클러스터를 구성하였습니다. CentOS 에서도, 설치가 가능하며 `microk8s` 를 구성하기 위해서는, `snap` 패키지 설치가 선행되어야 합니다.

```
# snap 패키지 설치
# 1. 우분투 환경
sudo apt update
sudo apt install snapd

# 2. CentOS 환경
sudo dnf install epel-release -y
sudo dnf update
sudo dnf -y install snapd
```

```
# microk8s 설치 (snap 패키지로 설치)
sudo snap install microk8s --classic --channel=1.27

# 권한 설정
sudo usermod -a -G microk8s $USER
sudo chown -f -R $USER ~/.kube
```

```
# 쿠버네티스클러스터상태조회
microk8s status --wait-ready
```

```
# 쿠버네티스노드조회
microk8s kubectl get nodes
```

2.1.2. kubectl CLI

microk8s 클러스터인경우기본적으로설치가되어있으며, 별도의설치가필요없습니다.

클러스터와통신할수있는 CLI

클라이언트로서특정클러스터를바라볼수있도록설정한이후, 커맨드라인을통해쿠버네티스자원을생성, 수정, 삭제, 조회할수있다.

```
# microk8s 사용시기본적으로설치되어있음.
microk8s kubectl version

# microk8s 이외의환경은, 클러스터구성후 kubectl 바이너리다운로드하여설치
curl -O https://s3.us-west-2.amazonaws.com/amazon-eks/1.26.4/2023-05-11/bin/linux/amd64/kubectl

chmod +x ./kubectl
mkdir -p $HOME/bin && cp ./kubectl $HOME/bin/kubectl
export PATH=$HOME/bin:$PATH
echo 'export PATH=$HOME/bin:$PATH' >> ~/.bashrc
source ~/.bashrc
kubectl version --short --client
```

2.2. helm CLI 설치

microk8s 기반클러스터의경우 add-on 으로 helm3 를추가하여손쉽게사용할수있으며, 이외의환경에서는 get_helm.sh 설치스크립트를다운받아설치진행합니다.

```
# 1. microk8s 환경
microk8s enable helm3

# 2. microk8s 클러스터이외의환경에서는, shell 스크립트로설치
curl -fsSL -o get_helm.sh
https://raw.githubusercontent.com/helm/helm/main/scripts/get-helm-3
chmod 700 get_helm.sh
./get_helm.sh
```

3. 차트 설치

3.1. 차트 디렉토리 생성

```
ubuntu@ip-172-34-71-29:~$ ls tst-helm/  
001.df 002.common 004.fe 005.sfg 006.sg 007.task lib-chart parent-chart  
ubuntu@ip-172-34-71-29:~$
```

<tst-helm 이라는 차트 디렉토리를 생성>

Helm 은 Chart 라는 패키지 단위를 사용합니다. Chart 는 Helm 이 애플리케이션을 정의하는 단위이며, 차트를 생성하기 위해서는 먼저 '디렉토리'를 생성하여, 템플릿과 서비스별 value 를 yaml 파일 형식으로 정의해야 합니다. 이후에, helm install 커맨드로 생성한 디렉토리를 참조하여 차트의 릴리즈를 생성합니다.

* 디렉토리 구조를 만드는 방법은 두 가지가 있습니다:

1. helm create 커맨드 실행

2. 디렉토리 구조 직접 생성

차트 하나에 하나의 서비스를 정의하는 경우 helm create 로 간단하게 생성 가능하지만, MSA 환경의 경우 다수의 서비스와 이들의 공통된 구조를 base chart 로 관리할 수 있도록 디렉토리 구조를 직접 정의하고, 생성해야 합니다.

3.1.1. helm create 로 디렉토리 생성

```
ubuntu@ip-172-34-71-29:~$ helm create my-chart  
Creating my-chart  
ubuntu@ip-172-34-71-29:~$ ls my-chart/  
Chart.yaml charts templates values.yaml  
ubuntu@ip-172-34-71-29:~$
```

하나의 서비스를 차트로 정의하는 경우, helm create 커맨드를 사용하여, 손쉽게 디렉토리 구조를 만들 수 있으며, 환경에 맞게 template 과 value 내용이 담긴 yaml 파일을 수정하여 사용합니다.

```
# 1. shell 스크립트로 설치  
helm create [DIR-NAME]  
tree DIR-NAME
```



```

DIR-NAME
├── Chart.yaml
├── charts
├── templates
│   ├── NOTES.txt
│   ├── _helpers.tpl
│   ├── deployment.yaml
│   ├── hpa.yaml
│   ├── ingress.yaml
│   ├── service.yaml
│   ├── serviceaccount.yaml
│   └── tests
│       └── test-connection.yaml
└── values.yaml

```

3.1.2. 디렉토리 구조 직접 생성

다수의서비스들을정의해야하는경우, helm

create 이생성하는디렉토리구조를수정할필요가있습니다. 생성한디렉토리내부에 **parent-chart** 라는디렉토리의 Chart.yaml 에각서비스들의디렉토리위치와버전을명시해야합니다.

각서비스들은서브차트로정의되어, parent-chart 를빌드하면, 서브차트들이 .tgz 파일로빌드됩니다.

lib-chart 라는디렉토리에는, 공통된템플릿 (Go template) 을정의하며, 서비스들이이를참조하게됩니다.

```

tree tst-helm/

tst-helm
├── 001.df
│   ├── 001.df-root (마이크로서비스 1)
│   │   ├── Chart.yaml
│   │   ├── templates
│   │   │   ├── deployment.yaml
│   │   │   └── service.yaml
│   │   └── values.yaml
│   └── 002.df-cwmp-parser (마이크로서비스 2)
│       ├── Chart.yaml
│       ├── templates
│       │   ├── deployment.yaml
│       │   └── service.yaml
│       └── values.yaml
...
└── lib-chart
    ├── Chart.yaml
    └── templates
        ├── _configmap.yaml
        ├── _deployment.yaml
        └── _service.yaml

```

```
└── _vs.yaml
└── parent-chart
    ├── Chart.yaml
    └── charts
```

3.2. helm 커맨드로차트설치

위에서디렉토리구조가만들어졌다면, helm install 커맨드로차트의
'릴리즈'를설치할수있습니다. 설치하기에앞서,
서비스간디펜던시가있다면디펜던시빌드를해야하며, 차트의디렉토리구조나,
템플릿등에문제가없는지검증하고생성될결과를 yaml 로미리출력해볼수있습니다.

3.2.1. 차트 디펜던시 빌드

```
# parent-chart 의 Chart.yaml 에정의된서비스들의디펜던시를참고하여,  
# child 서비스들과 lib-chart 를서브차트패키지 .tgz 로빌드  
helm dep build parent-chart/  
  
# 빌드후변경사항 (values.yaml 또는서비스추가/수정/삭제가발생) 업데이트  
helm dep update parent-chart/
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm dep build parent-chart/  
Saving 35 charts  
Deleting outdated charts  
ubuntu@ip-172-34-71-29:~/tst-helm$ ls -al parent-chart/charts/  
total 156  
drwxr-xr-x 2 ubuntu ubuntu 4096 Jul 10 06:31 .  
drwxr-xr-x 3 ubuntu ubuntu 4096 Jul 10 06:31 ..  
-rw-rw-r-- 1 ubuntu ubuntu  986 Jul 10 06:31 df-api-0.1.0.tgz  
-rw-rw-r-- 1 ubuntu ubuntu 6582 Jul 10 06:31 df-cwmp-engine-0.1.0.tgz  
-rw-rw-r-- 1 ubuntu ubuntu  845 Jul 10 06:31 df-cwmp-parser-0.1.0.tgz
```

3.2.2. 차트 검증

```
# 차트포맷등오류검증  
helm lint parent-chart/  
  
# 템플릿에 values.yaml 에서정의한값을대입하여실제 Render 된결과출력  
helm template parent-chart/
```

```
# helm template 과비슷하지만, Render 된객체들이 valid 한
# 쿠버네티스객체인지검증
# 차트설치없이실행될결과를출력하여에러로그등을확인
helm install --dry-run my-release parent-chart
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm lint parent-chart/
==> Linting parent-chart/
[INFO] Chart.yaml: icon is recommended
[INFO] values.yaml: file does not exist
[WARNING] templates/: directory not found

1 chart(s) linted, 0 chart(s) failed
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm install --dry-run my-release parent-chart/
NAME: my-release
LAST DEPLOYED: Mon Jul 10 06:33:00 2023
NAMESPACE: default
STATUS: pending-install
REVISION: 1
TEST SUITE: None
HOOKS:
MANIFEST:
---
# Source: my-chart/charts/fe-api/templates/configmap.yaml
apiVersion: v1
kind: ConfigMap
data:
  id_rsa: |
    -----BEGIN OPENSSH PRIVATE KEY-----
```

3.2.3. 차트 설치

```
# 차트에해당하는 release 를생성하고, 쿠버네티스자원생성
helm install my-release parent-chart/
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm install my-release parent-chart/
NAME: my-release
LAST DEPLOYED: Mon Jul 10 06:35:55 2023
NAMESPACE: default
STATUS: deployed
REVISION: 1
TEST SUITE: None
ubuntu@ip-172-34-71-29:~/tst-helm$ helm list
NAME                NAMESPACE    REVISION    UPDATED                               STATUS    CHART          APP VERSION
my-release          default       1           2023-07-10 06:35:55.22745955 +0000 UTC deployed   my-chart-0.1.0
ubuntu@ip-172-34-71-29:~/tst-helm$
```

3.2.4. 차트 업그레이드

처음차트의릴리즈를설치하고, 추후수정사항이생기면, 차트 dependency 를먼저업데이트하고,

차트릴리즈를업그레이드합니다.

```
# 서비스별디펜던시가있는경우업그레이드전, 디펜던시를업데이트합니다.
```

```
helm dep update parent-chart
```

```
# 차트업그레이드 (템플릿혹은 value 정의값들등을반영)
```

```
helm upgrade my-release parent-chart/
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ vim 001.df/001.df-root/values.yaml
ubuntu@ip-172-34-71-29:~/tst-helm$ helm dep update parent-chart/
Saving 35 charts
Deleting outdated charts
ubuntu@ip-172-34-71-29:~/tst-helm$ helm upgrade my-release parent-chart/
Release "my-release" has been upgraded. Happy Helming!
NAME: my-release
LAST DEPLOYED: Mon Jul 10 06:37:52 2023
NAMESPACE: default
STATUS: deployed
REVISION: 2
TEST SUITE: None
ubuntu@ip-172-34-71-29:~/tst-helm$
```

3.2.5. 차트 버전 롤백

```
# 특정버전으로차트를롤백: 롤백시에도버전 1 씩증가
```

```
helm rollback my-release VERSION_NO
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm list
NAME                NAMESPACE    REVISION    UPDATED                               STATUS    CHART
my-release          default       2           2023-07-10 06:37:52.188639213 +0000 UTC deployed  my-chart-0.1.0
ubuntu@ip-172-34-71-29:~/tst-helm$ helm rollback my-release 1
Rollback was a success! Happy Helming!
ubuntu@ip-172-34-71-29:~/tst-helm$ helm list
NAME                NAMESPACE    REVISION    UPDATED                               STATUS    CHART
my-release          default       3           2023-07-10 06:38:19.914178631 +0000 UTC deployed  my-chart-0.1.0
ubuntu@ip-172-34-71-29:~/tst-helm$
```

3.2.6. 차트 삭제

```
# 차트의 my-release 릴리즈를삭제: 쿠버네티스자원도함께삭제가됩니다.
```

```
helm uninstall my-release
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm list
NAME                NAMESPACE    REVISION    UPDATED                               STATUS    CHART    APP VERSION
my-release          default       4           2023-07-10 05:20:52.860328329 +0000 UTC deployed  my-chart-0.1.0
ubuntu@ip-172-34-71-29:~/tst-helm$ helm uninstall my-release
ubuntu@ip-172-34-71-29:~/tst-helm$ kubectl get pod -n default
No resources found in default namespace.
```

3.3. 차트 릴리즈 변경 사항 확인

처음 차트 릴리즈를 생성하고, 차트에 변경이 생겨서 `values` 또는, 차트 템플릿을 수정하고, `helm upgrade` 를 진행하면, 새로운 버전의 릴리즈가 생성됩니다. 릴리즈 버전 사이의 변경 사항을 확인할 수 있는 `helm diff` 플러그인을 설치하여 사용할 수 있습니다.

```
# helm diff 플러그인 설치
helm plugin install https://github.com/databus23/helm-diff

# values.yaml 변경 후 업그레이드를 하고 helm diff 로 변경 사항 확인
helm dep update my-release parent-chart
helm upgrade my-release parent-chart
helm diff revision my-release 1 2
```

```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm diff revision my-release 2 3
default, df-root, Deployment (apps) has changed:
# Source: my-chart/charts/df-root/templates/deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: df-root
  namespace: default
  labels:
    app: df-root
    version: v3.1
spec:
-   replicas: 2
+   replicas: 1
  selector:
```

0. Reference

- <https://boxboat.com/2018/09/19/helm-and-kubernetes-deployments/>
- <https://helm.sh/>
- <https://devops.stackexchange.com/questions/13379/use-one-helm-chart-for-all-microservices>