Helm Chart
운영 가이드

Technical Document

Version 0.1
(2023. 07.07)

목차

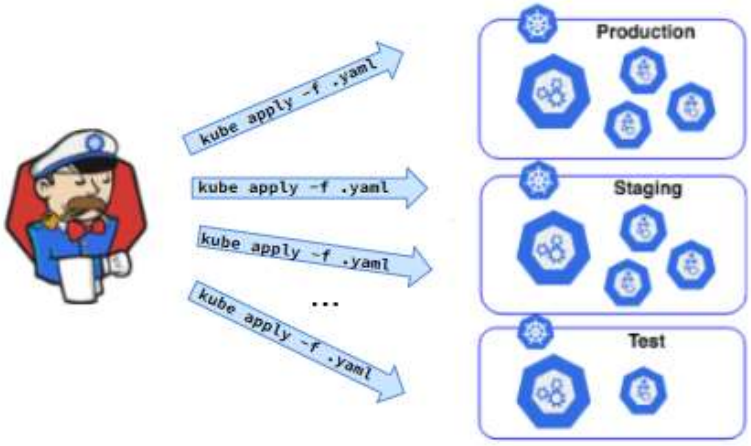
내용

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

1. Helm Chart 검토배경

컨테이너 기반 MSA 서비스를 쿠버네티스로 구축 하려면,수많은 컴포넌트를 yaml 파일로 정의해야 하며, kubectl 커맨드를 각 컴포넌트 yaml 파일에 개별적으로 실행하여 생성 또는 수정하는 반복적인 작업이 필요합니다.

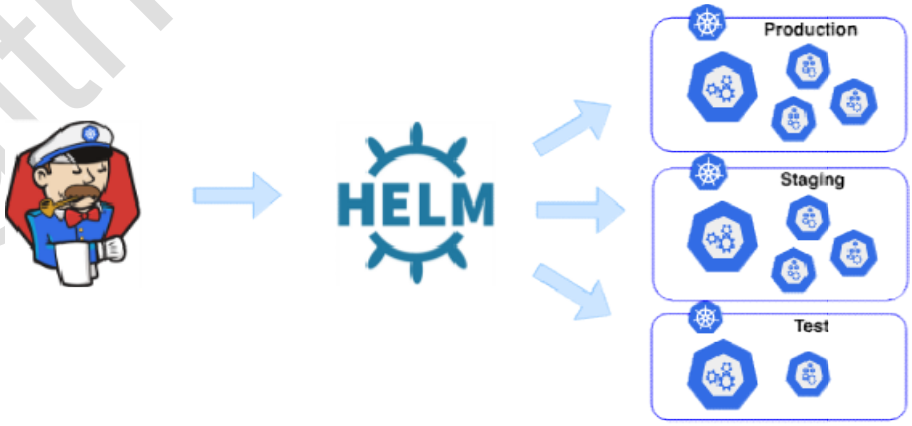
(kubectl: 쿠버네티스 클러스터의 API Server 와 통신하는 client CLI tool)



<kubectl 로 컴포넌트 생성시 반복작업으로 인한 비효율 발생>

1.1. Helm Chart 란?

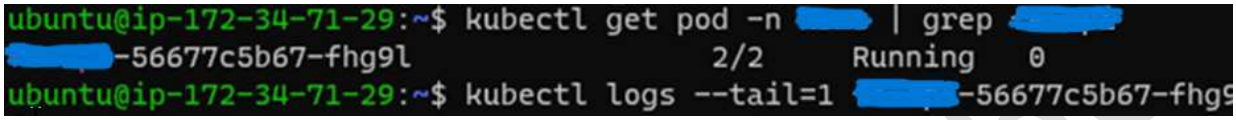
Helm Chart 패키지 매니저는 하나의 커맨드라인으로 차트를 설치하여, 다수의 서비스 컴포넌트를 한번에 생성할 수 있으며, 변경사항이 발생했을때, helm upgrade 로 빠르게 반영할 수 있습니다. 또한, 버전 관리를 통해, 배포 건에 대한 손쉬운 rollback, upgrade, delete 등의 작업이 가능합니다.



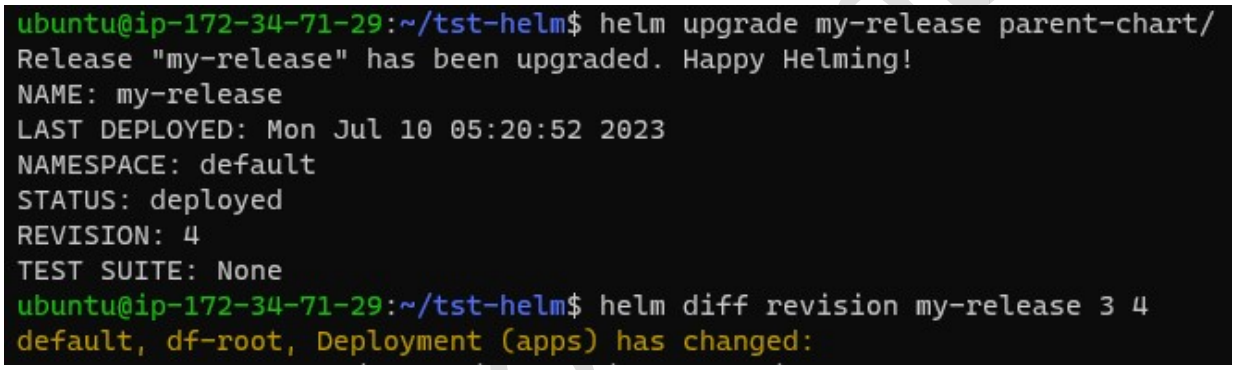
<helm 으로컴포넌트생성>*from boxbeat.com*

1.2. helm 과 kubectl 커맨드의역할

Helm 은 kubectl 커맨드만으로 서비스 컴포넌트 생성, 수정, 삭제하던 반복적인 작업을 대체함과 동시에, Chart 패키지의 버전 관리 및 롤백 등의 역할을 합니다. 그외에, 자원의 조회 및 로그 확인, 컨테이너 내부 진입 등 디버깅은 기존에 사용하던 kubecstl 로 수행합니다. 이와 같이 목적에 따라, helm 과 kubectl 커맨드를 함께 사용해야 합니다.



<kubectl 커맨드>



<helm 커맨드>

2. Helm CLI 설치

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

2.1. 설치 전 선행 작업

Helm 을설치하기전에쿠버네티스클러스터구성과클러스터와통신하기위한 kubectl 커맨드라인툴이설치되어있어야합니다.

2.1.1. 클러스터 구성

* 쿠버네티스클러스터구성에는다음과같은방법이존재합니다 :

1. 관리형 쿠버네티스서비스 : CSP (AWS, Google Cloud, Azure)가 제공하는 EKS, GKE, AKS 를 사용하여 클러스터 구성.
2. On-premise/EC2 에 microk8s 또는 minikube 로 경량화 클러스터 구성.

본 문서에서는 AWS EC2 우분투 Linux 머신에 microk8s 를설치하여,
클러스터를구성하였습니다. microk8s 는 snap 패키지로 설치 할 수 있습니다.

```
# snap 패키지설치
# 1. 우분투환경
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 패키지로설치)
sudo snap install microk8s --classic --channel=1.27

# 권한설정
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 사용시 기본적으로 설치되어 있음.
microk8s kubectl version
```

```
# microk8s 이외의환경은,클러스터구성후 kubectl 바이너리 다운로드하여 설치
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

<tst-helm 이라는차트디렉토리를생성>

Helm 은 Chart 라는패키지단위를사용합니다. Chart 는 Helm 이 애플리케이션을정의하는단위이며, 차트를생성하기위해서는먼저'디렉토리'를생성하여, 템플릿과서비스별 value 를 yaml 파일형식으로정의해야합니다. 이후에,helm install 커맨드로생성한디렉토리를참조하여차트의릴리즈를생성합니다.

* 디렉토리구조를만드는방법은두가지가있습니다:

1.helm create 커맨드실행

2.디렉토리구조직접생성

차트하나에하나의서비스를정의하는경우 helm create 로간단하게생성가능하지만, MSA 환경의경우다수의서비스와이들의공통된구조를 base chart 로관리할수있도록디렉토리구조를직접정의하고, 생성해야합니다.

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:~$
```

하나의서비스를차트로정의하는경우, helm create 커맨드를사용하여, 손쉽게디렉토리구조를만들수있으며, 환경에맞게 template 과 value 내용이담긴 yaml 파일을수정하여사용합니다.

```
# 1. shell 스크립트로설치
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 라는 디렉토리의 Chart.yaml 에각서비스들의디렉토리위치와버전을명시해야합니다. 각서비스들은서브차트로정의되어, parent-chart 를빌드하면,서브차트들이.tgz 파일로빌드됩니다.

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

```
tree tst-helm/

tst-helm
├── 001.fe
│   ├── 001.fe-api (마이크로서비스 1)
│   │   ├── Chart.yaml
│   │   ├── templates
│   │   │   ├── deployment.yaml
│   │   │   └── service.yaml
│   │   └── values.yaml
│   ├── 002.fe-auth (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 커맨드로 차트 설치

위에서 디렉토리 구조가 만들어졌다면, 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
```

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
```

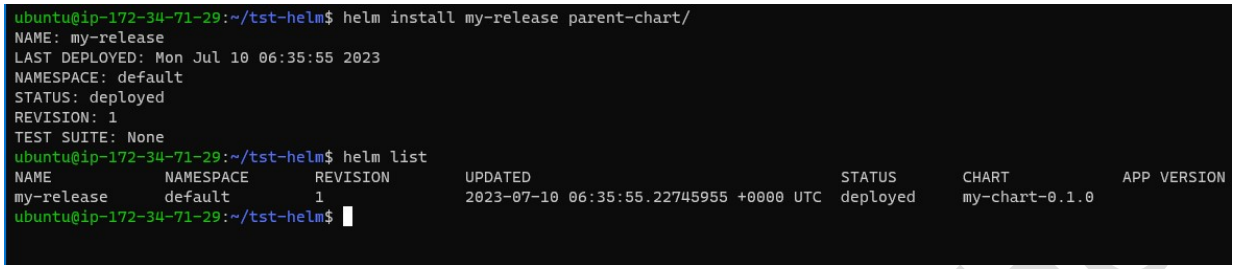
```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm lint parent-chart/
==> Linting parent-chart/
[INFO] Chart.yaml: icon is recommended
[INFO] values.yaml: file does not exist
[WARNING] templates/: directory not found

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:
---
```

3.2.3. 차트 설치

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

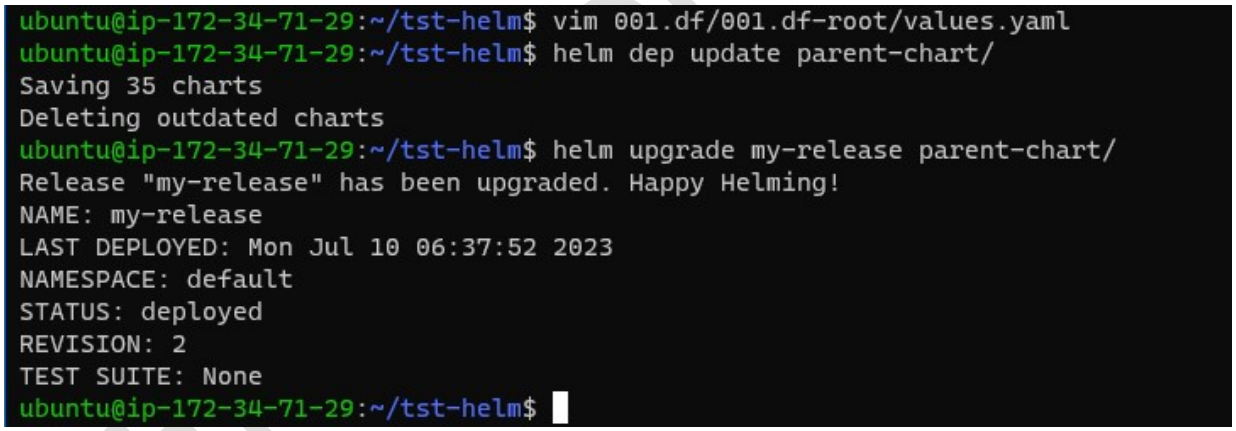


3.2.4.차트 업그레이드

처음차트의릴리즈를설치하고,추후수정사항이생기면,차트 dependency 를먼저업데이트하고, 차트릴리즈를업그레이드합니다.

```
# 서비스별디펜던시가있는경우업그레이드전,디펜던시를업데이트합니다.
helm dep update parent-chart

# 차트업그레이드 (템플릿혹은 value 정의값들등을반영)
helm upgrade my-release parent-chart/
```



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
RSION
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
RSION
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
```

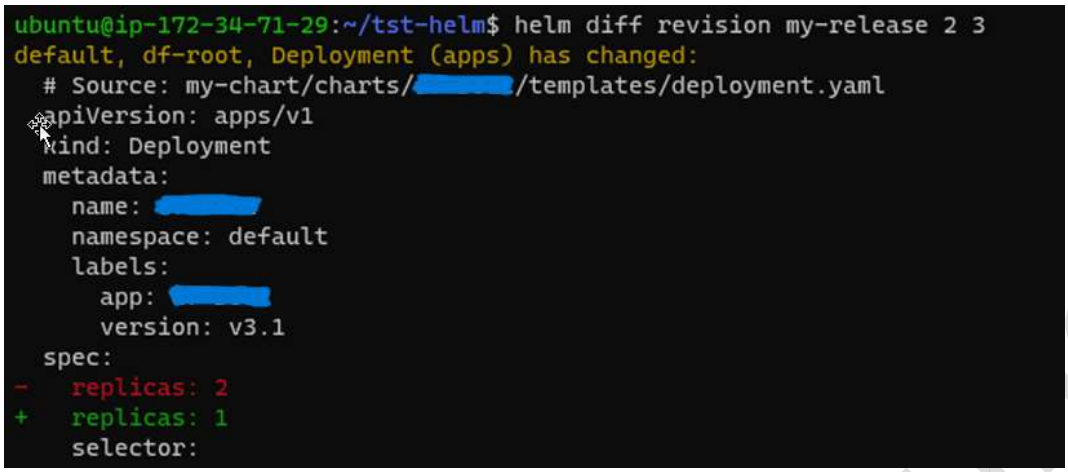
```
ubuntu@ip-172-34-71-29:~/tst-helm$ helm list
NAME            NAMESPACE       REVISION        UPDATED                                 STATUS          CHART             APP VERSIO
my-release      default         4               2023-07-10 05:20:52.860328329 +0000 UTC deployed        my-chart-0.1.0
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 변경사항 확인

처음차트릴리즈를생성하고,차트에변경이생겨서 values 또는,차트템플릿을수정하고, helm upgrade 를진행하면,새로운버전의릴리즈가생성됩니다.릴리즈버전사이의변경사항을확인할수있는 helm diff 플러그인을설치하여사용할수있습니다.

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

# values.yaml 변경후업그레이드를하고 helm diff 로변경사항확인
helm dep update my-release parent-chart
helm upgrade my-release parent-chart
helm diff revision my-release 1 2
```



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